

Color Language

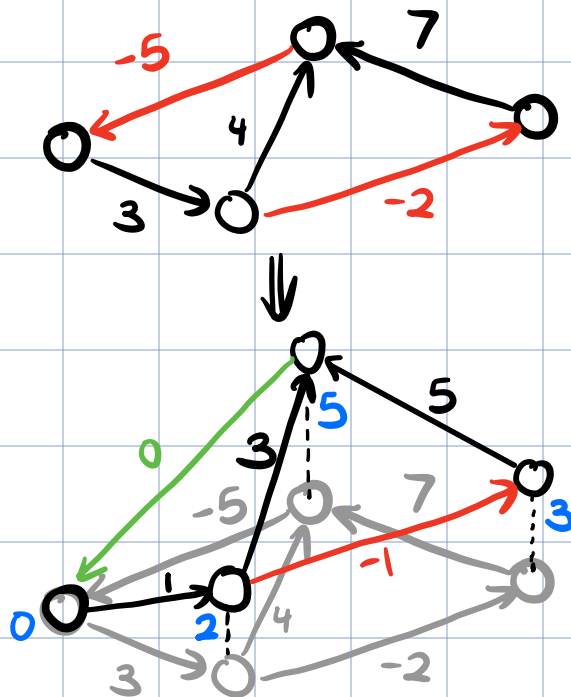
Black - text & generic edge

Red - negative edges, vert's, & problems

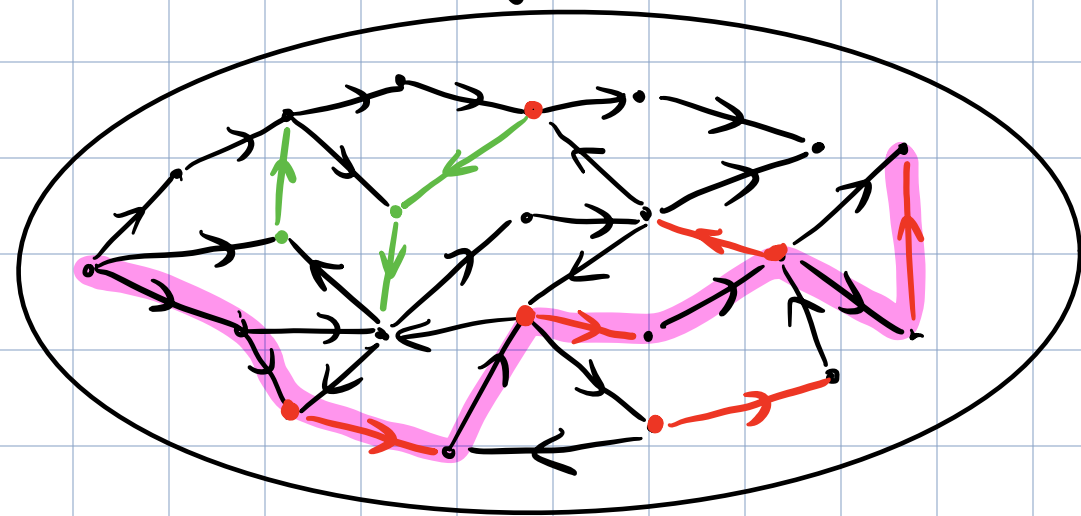
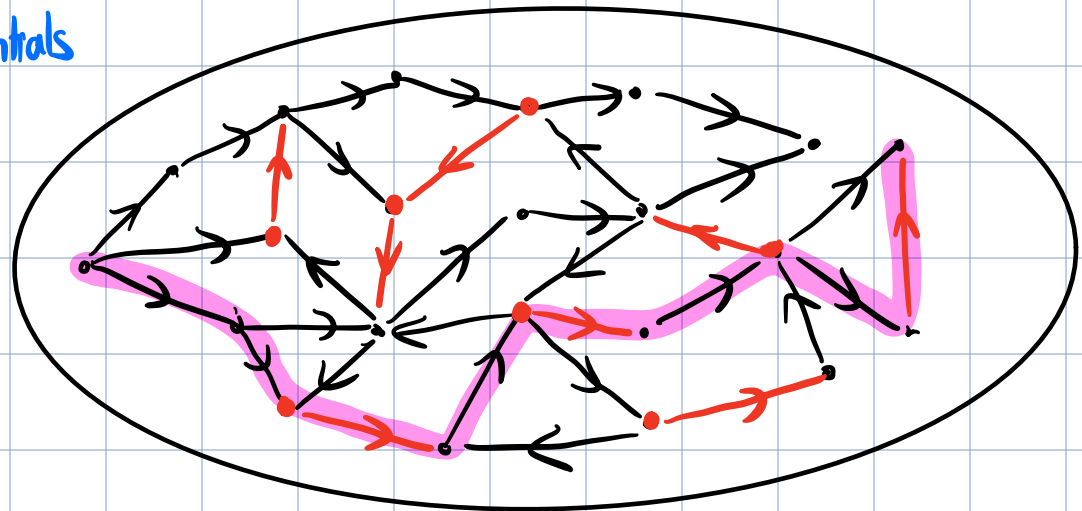
Blue - Remarks & annotations; potentials

Green - Neutralized arcs

Purple - Shortest path (tree)

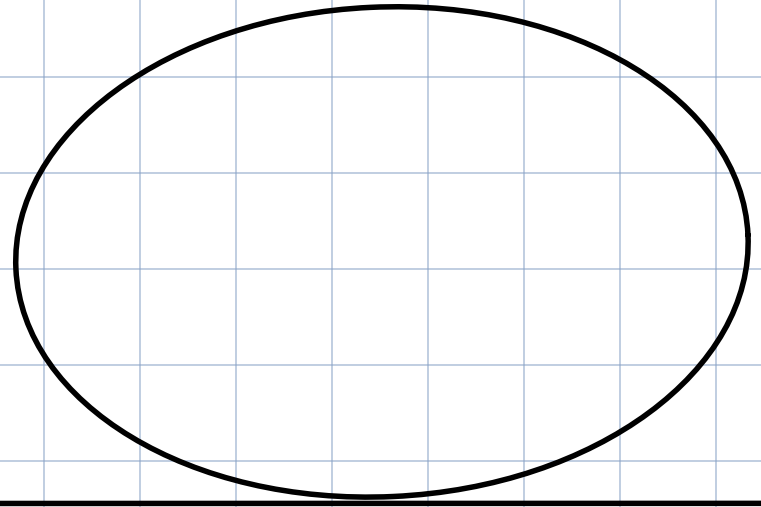


Ex.

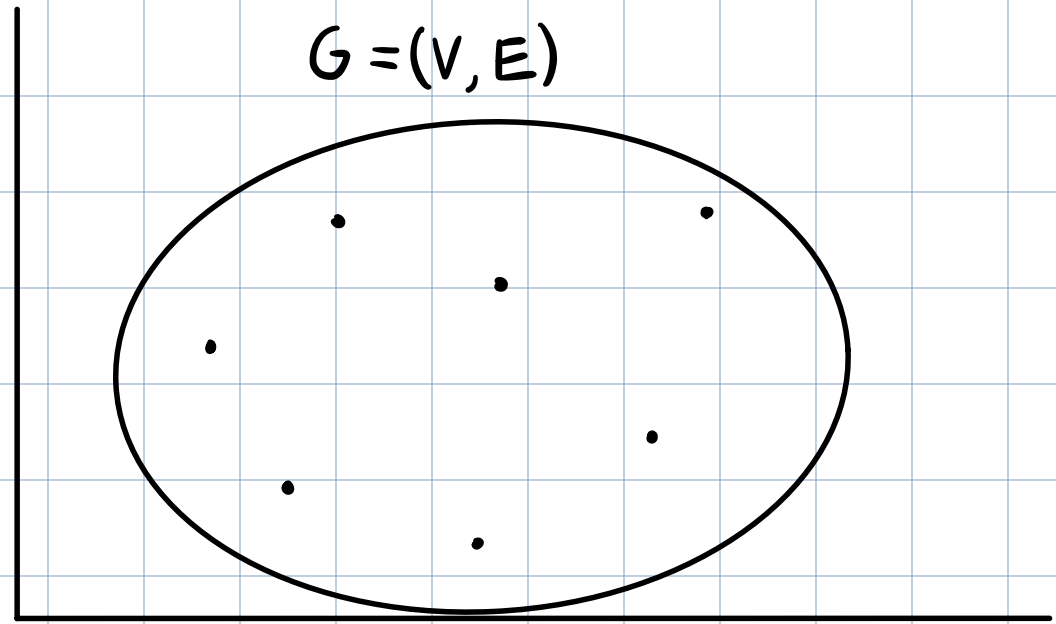


Vertices & Edges; $n=|V|, m=|E|$

$G=(V, E)$

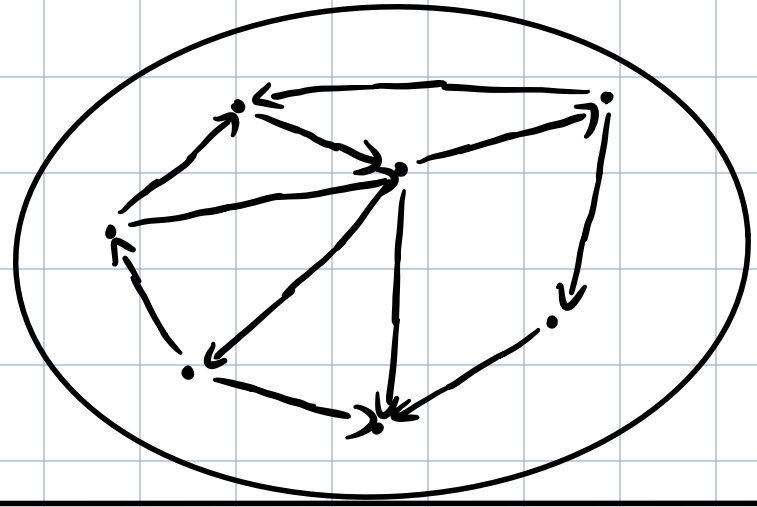


Vertices & Edges; $n = |V|$, $m = |E|$



Vertices & Edges; $n=|V|, m=|E|$

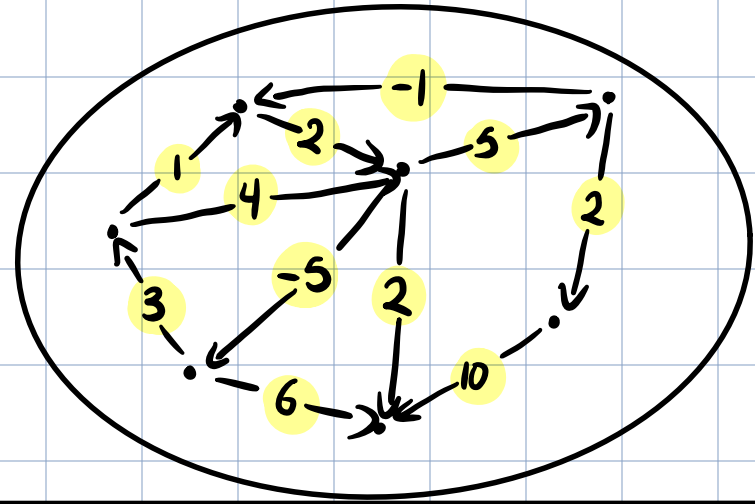
$G=(V,E)$



Vertices & Edges; $n=|V|, m=|E|$

Edge weights $l:E \rightarrow \mathbb{R}$

$G=(V,E)$

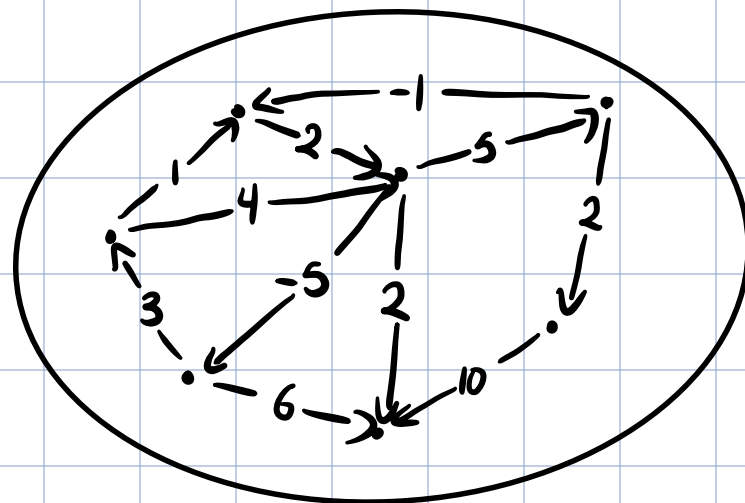


Vertices & Edges; $n=|V|, m=|E|$

Edge weights $l: E \rightarrow \mathbb{R}$

(s,t) -walks W ; $l(W) := \sum_{e \in W} l(e)$

$G=(V,E)$

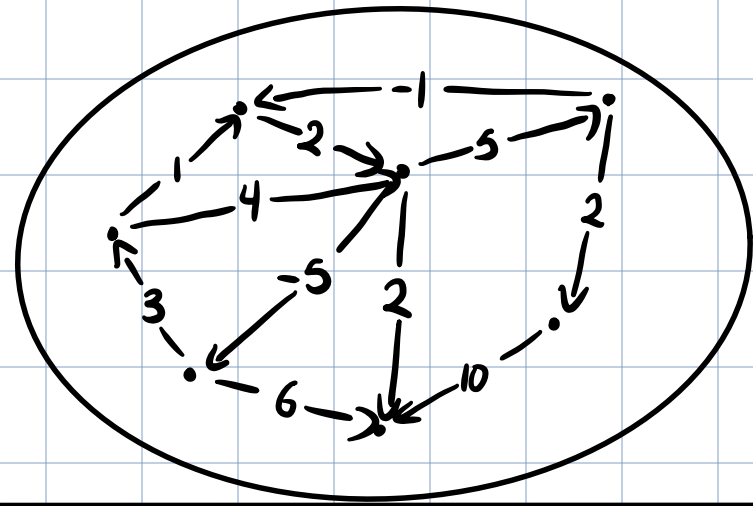


Vertices & Edges; $n=|V|, m=|E|$

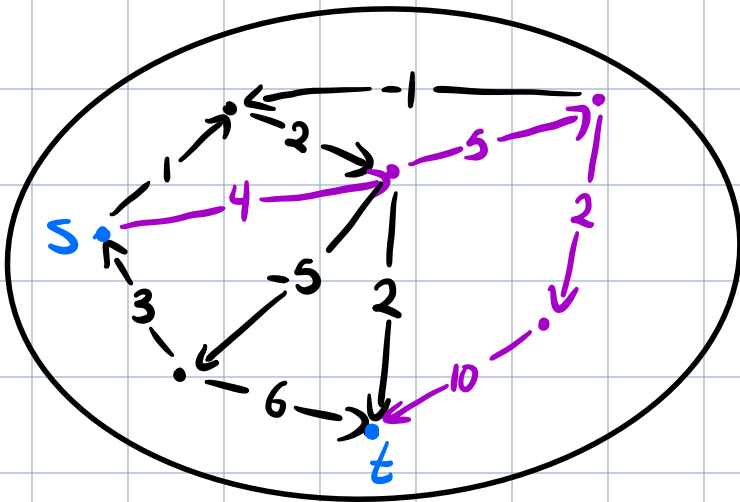
Edge weights $l: E \rightarrow \mathbb{R}$

(s,t) -walks W ; $l(W) := \sum_{e \in W} l(e)$

$G=(V,E)$



$l(W_1) =$

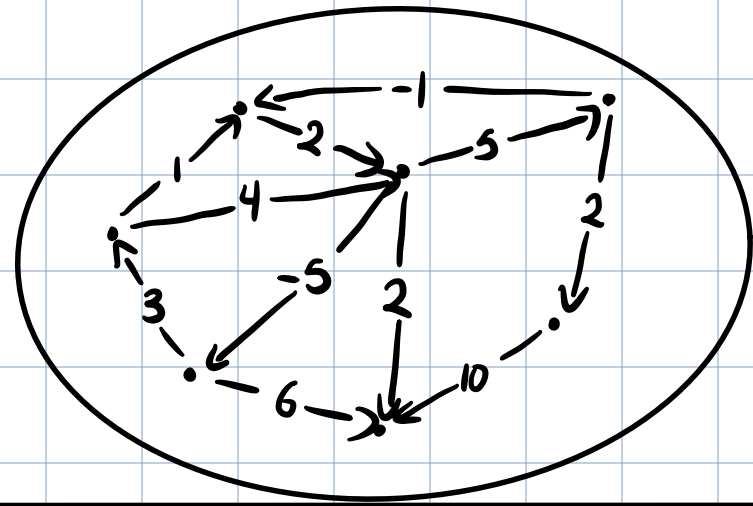


Vertices & Edges; $n=|V|, m=|E|$

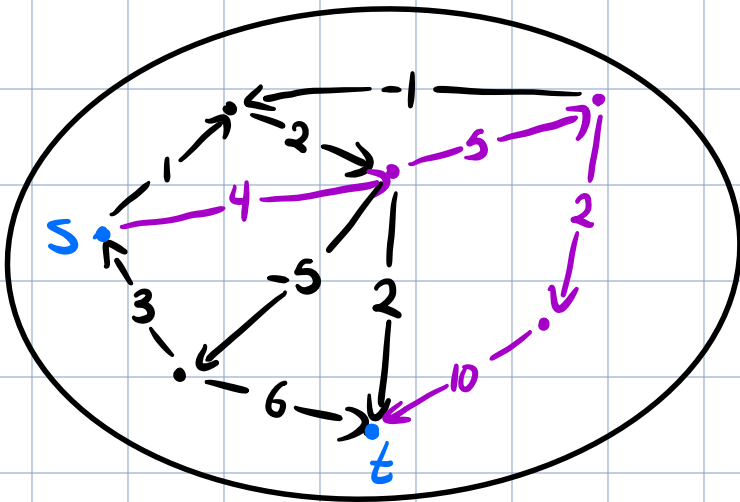
Edge weights $l: E \rightarrow \mathbb{R}$

(s,t) -walks W ; $l(W) := \sum_{e \in W} l(e)$

$G=(V,E)$



$$l(W_1) = 21$$

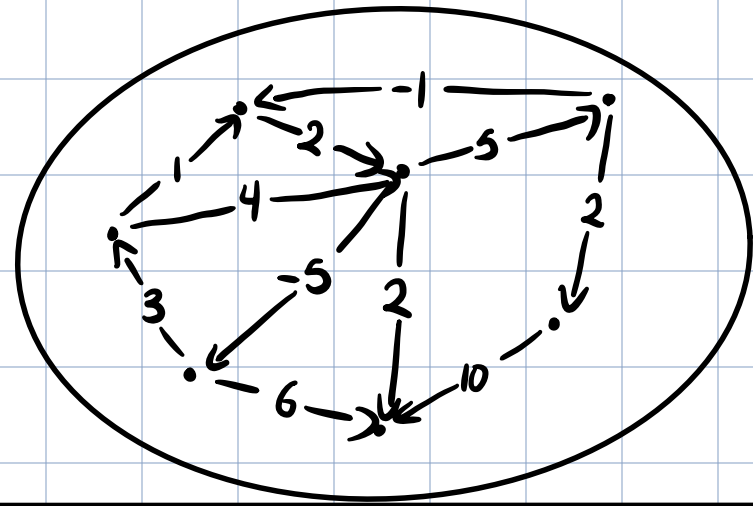


Vertices & Edges; $n=|V|, m=|E|$

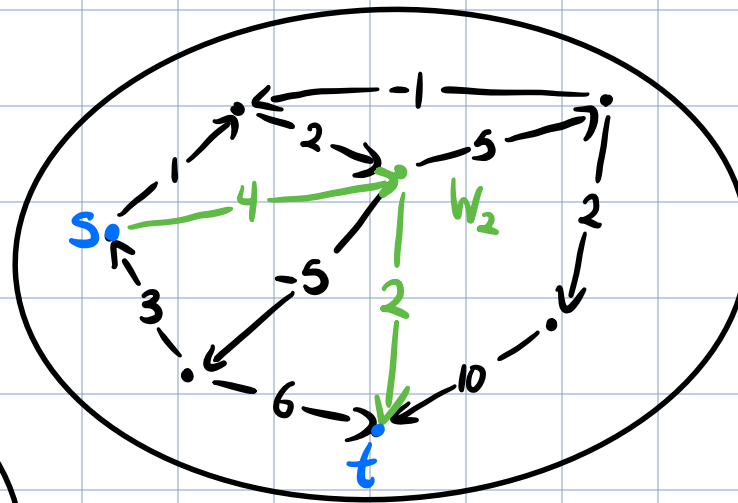
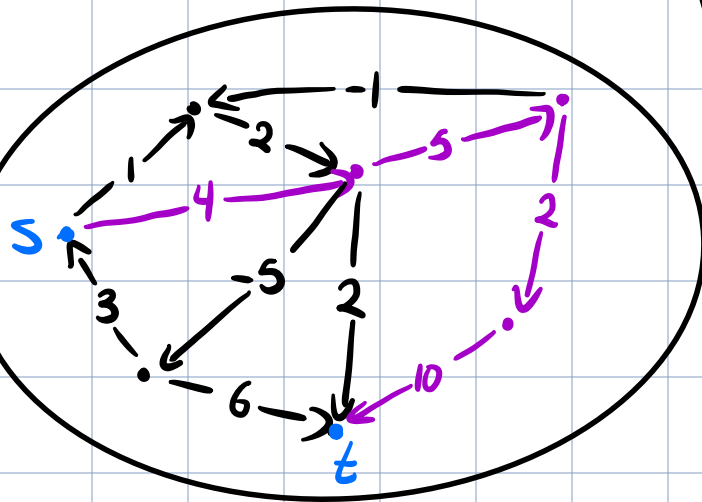
Edge weights $l:E \rightarrow \mathbb{R}$

(s,t) -walks W ; $l(W) := \sum_{e \in W} l(e)$

$G=(V,E)$



$$l(W_1) = 21$$



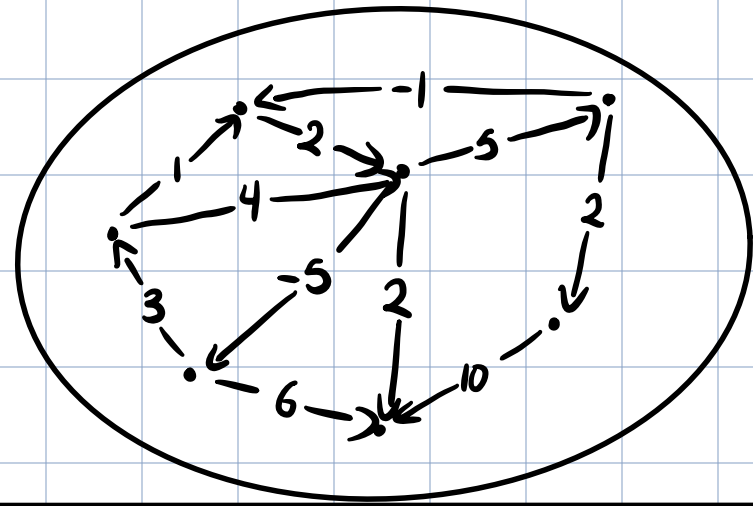
$$l(W_2) =$$

Vertices & Edges; $n=|V|, m=|E|$

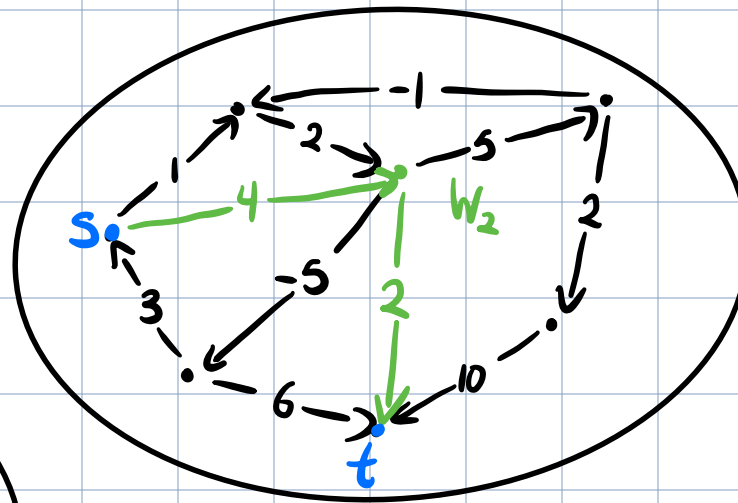
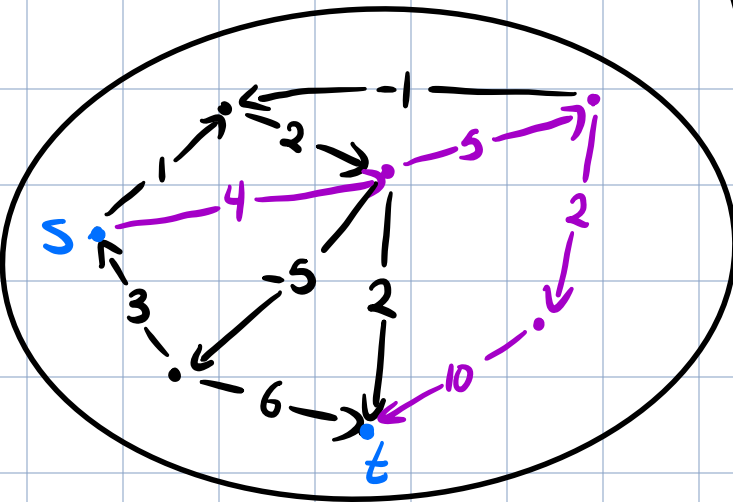
Edge weights $l: E \rightarrow \mathbb{R}$

(s,t) -walks W ; $l(W) := \sum_{e \in W} l(e)$

$G=(V,E)$



$$l(W_1) = 21$$



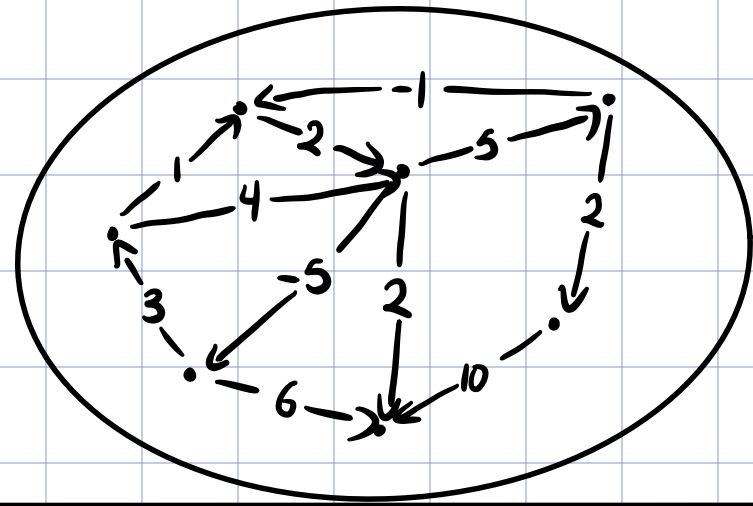
$$l(W_2) = 6$$

Vertices & Edges; $n=|V|, m=|E|$

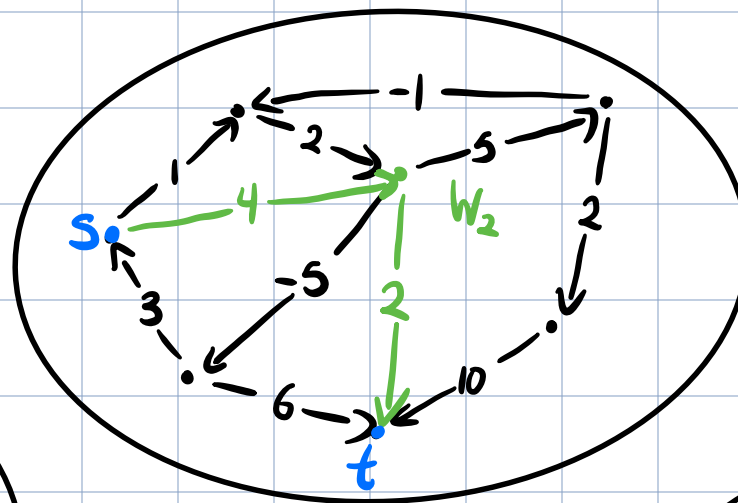
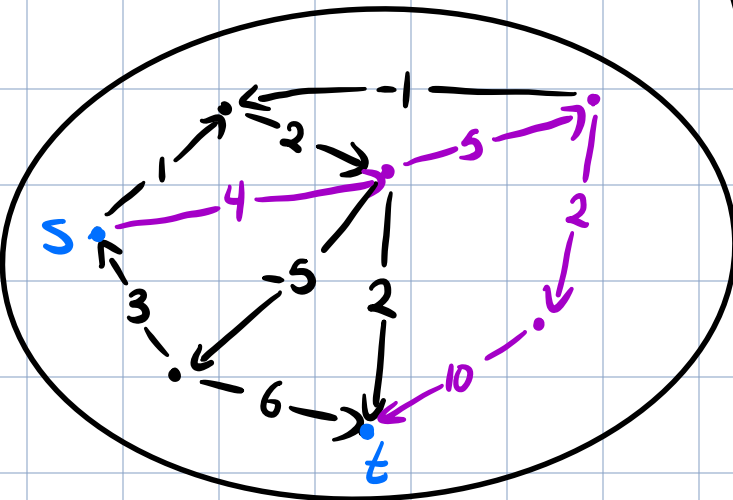
Edge weights $l: E \rightarrow \mathbb{R}$

(s,t) -walks W ; $l(W) := \sum_{e \in W} l(e)$

$G=(V,E)$

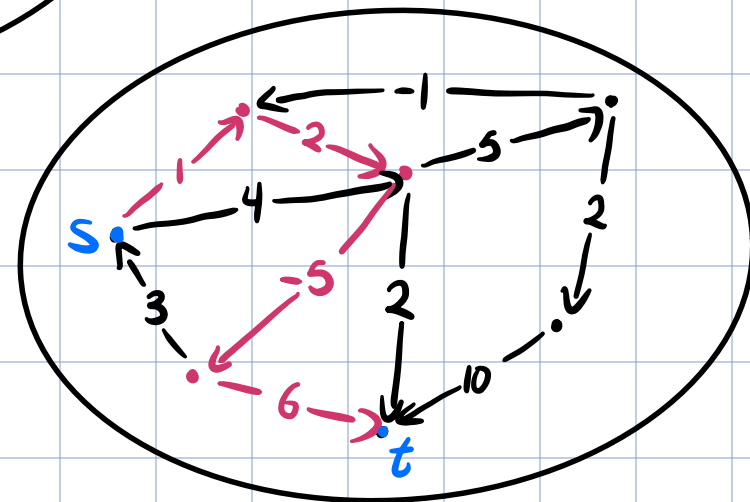


$$l(W_1) = 21$$



$$l(W_2) = 6$$

$$l(W_3) =$$

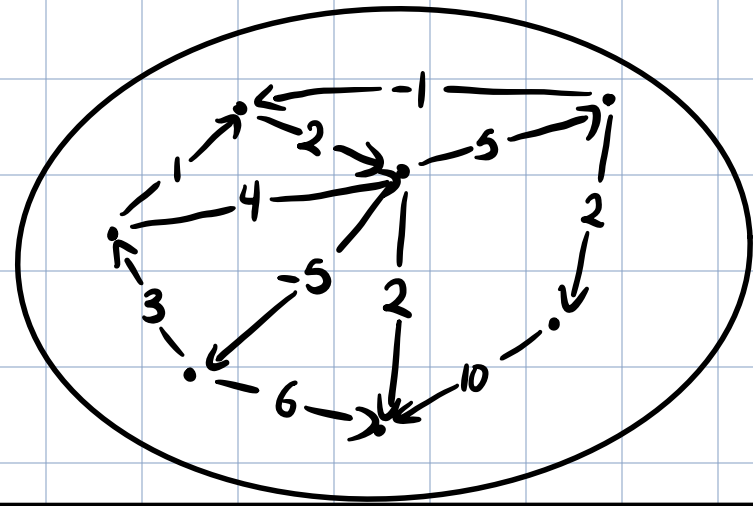


Vertices & Edges; $n=|V|, m=|E|$

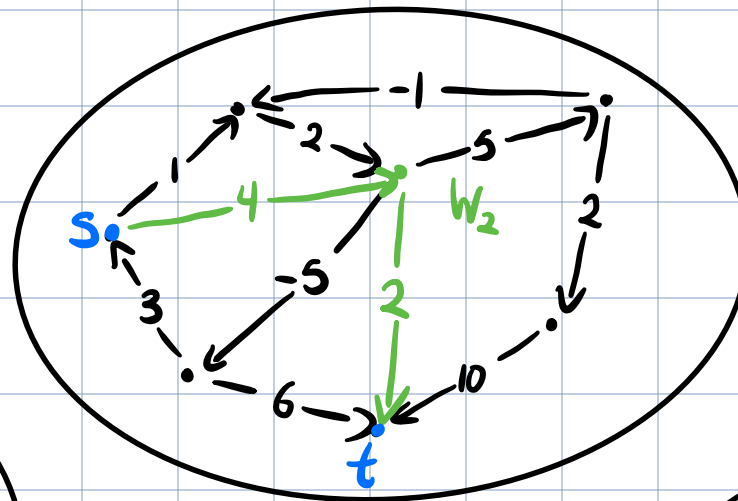
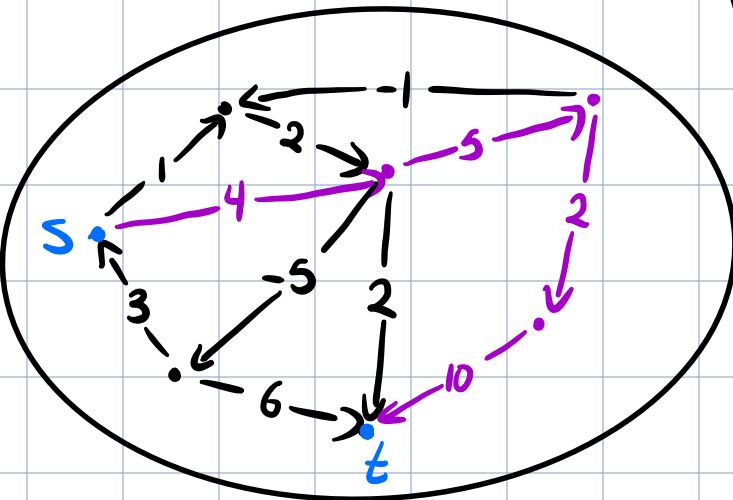
Edge weights $l:E \rightarrow \mathbb{R}$

(s,t) -walks W ; $l(W) := \sum_{e \in W} l(e)$

$G=(V,E)$

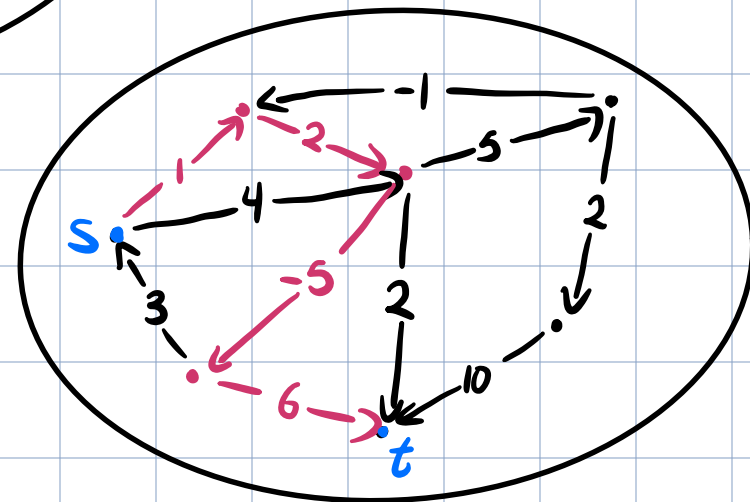


$$l(W_1) = 21$$

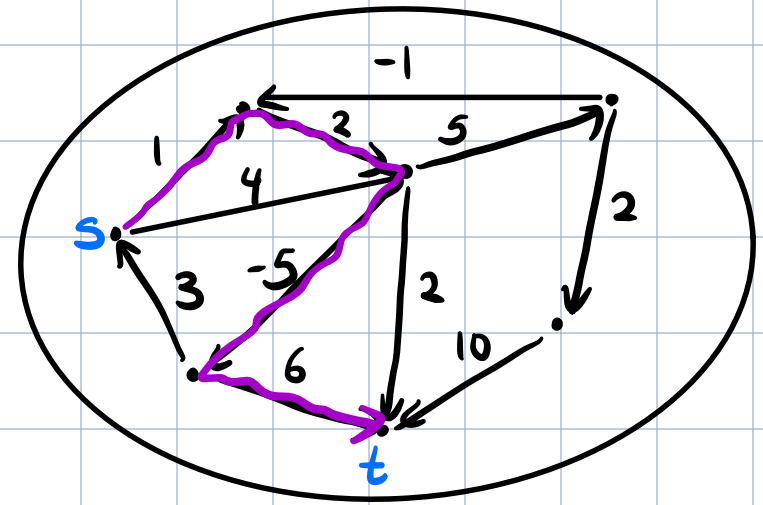


$$l(W_2) = 6$$

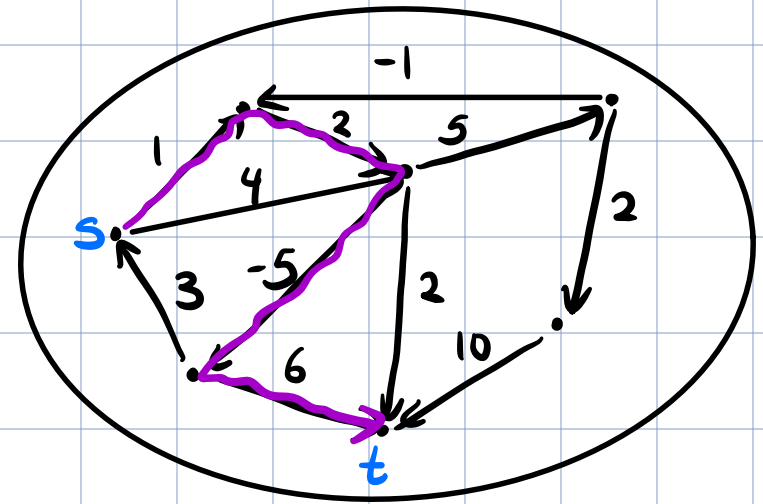
$$l(W_3) = 4$$



$$d(s,t) := \min \{ l(w) \mid w \text{ an } (s,t)\text{-walk} \}$$



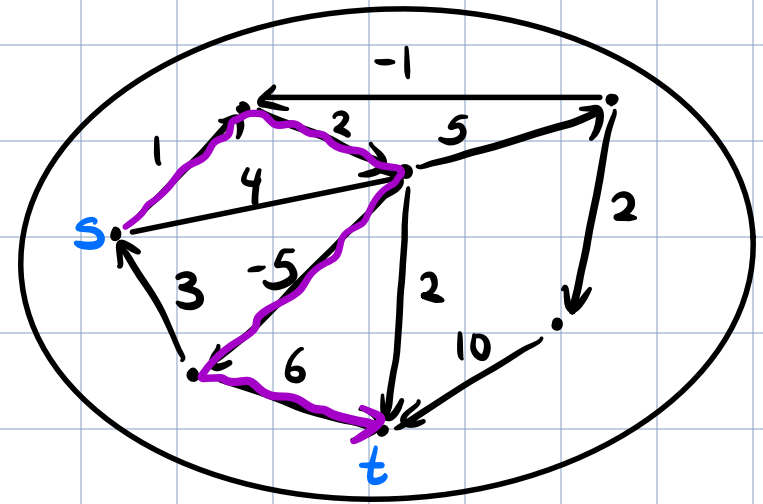
$$d(s,t) := \min \{ l(w) \mid w \text{ an } (s,t)\text{-walk} \}$$



$$d(s,t) = 2$$

$$d(s,t) := \min \{ l(w) \mid w \text{ an } (s,t)\text{-walk} \}$$

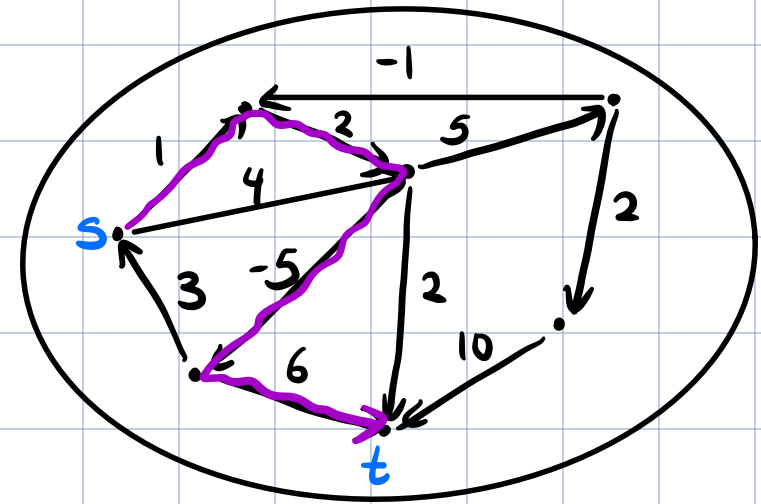
Triangle inequality: $d(s,u) + d(u,t) \geq d(s,t)$



$$d(s,t) = 2$$

$$d(s,t) := \min \{ l(w) \mid w \text{ an } (s,t)\text{-walk} \}$$

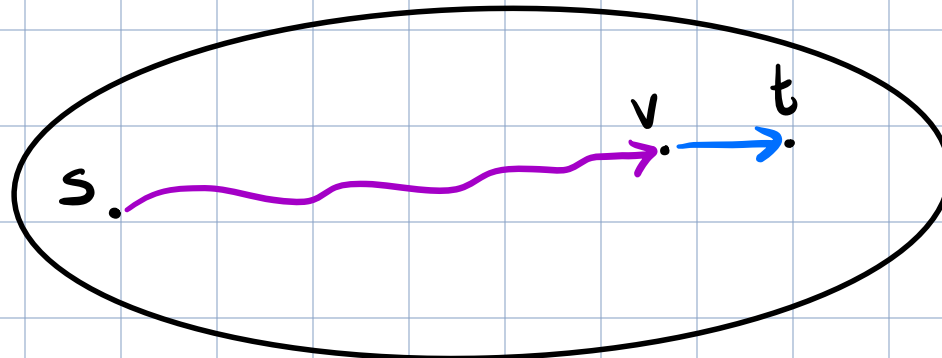
Triangle inequality: $d(s,u) + d(u,t) \geq d(s,t)$



$$d(s,t) = 2$$

"Recurrence"

$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$

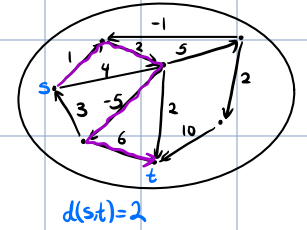
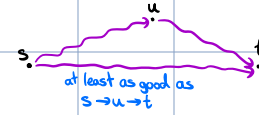


Aside 1

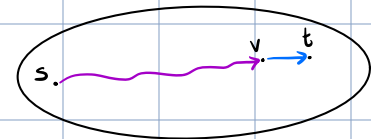
$\min \{l(w) \mid w \text{ an } (s,t)\text{-walk}\}$ may be ill-defined!

$$d(s,t) := \min \{l(w) \mid w \text{ an } (s,t)\text{-walk}\}$$

Triangle inequality: $d(s,u) + d(u,t) \geq d(s,t)$



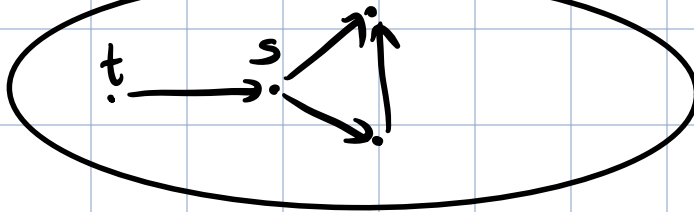
"Recurrence"
$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$



Aside 1

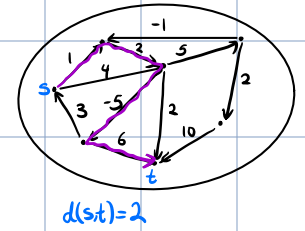
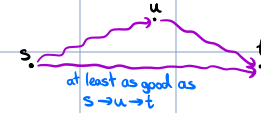
$\min \{l(w) \mid w \text{ an } (s,t)\text{-walk}\}$ may be ill-defined!

Reach:



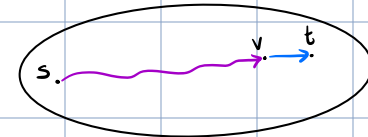
$$d(s,t) := \min \{l(w) \mid w \text{ an } (s,t)\text{-walk}\}$$

Triangle inequality: $d(s,u) + d(u,t) \geq d(s,t)$



$$d(s,t) = 2$$

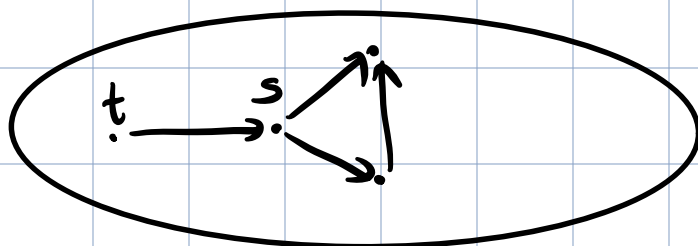
"Recurrence"
$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$



Aside 1

$\min \{l(w) \mid w \text{ an } (s,t)\text{-walk}\}$ may be ill-defined!

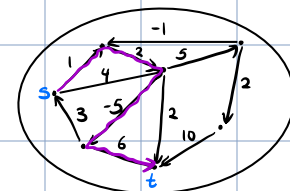
Reach:



$$d(s,t) = \infty \quad \text{"min } \emptyset = \infty \text{"}$$

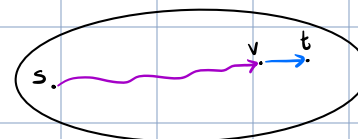
$$d(s,t) := \min \{l(w) \mid w \text{ an } (s,t)\text{-walk}\}$$

Triangle inequality: $d(s,u) + d(u,t) \geq d(s,t)$



$$d(s,t) = 2$$

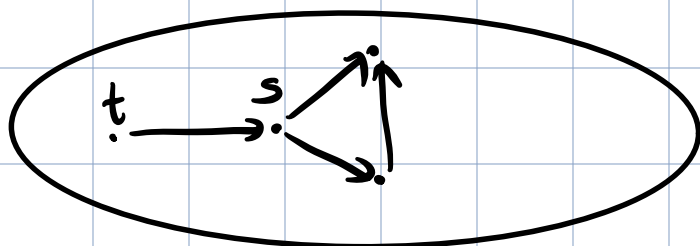
"Recurrence"
$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$



Aside 1

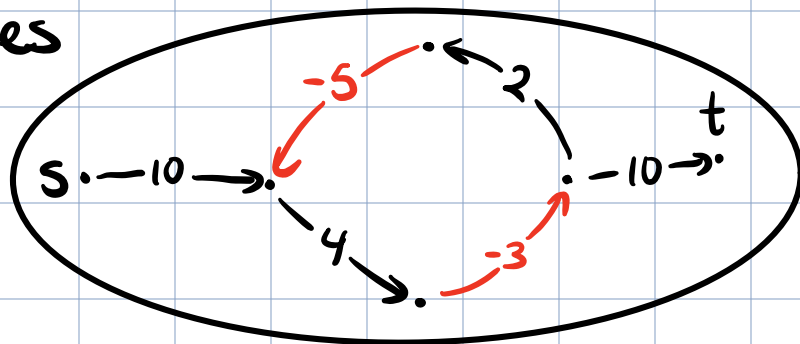
$\min \{l(w) \mid w \text{ an } (s,t)\text{-walk}\}$ may be ill-defined!

Reach:



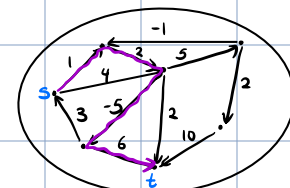
$$d(s,t) = \infty \quad \text{"min } \emptyset = \infty \text{"}$$

Negative cycles



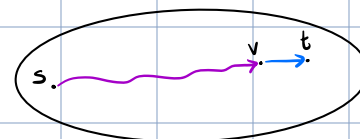
$$d(s,t) := \min \{l(w) \mid w \text{ an } (s,t)\text{-walk}\}$$

Triangle inequality: $d(s,u) + d(u,t) \geq d(s,t)$



$$d(s,t) = 2$$

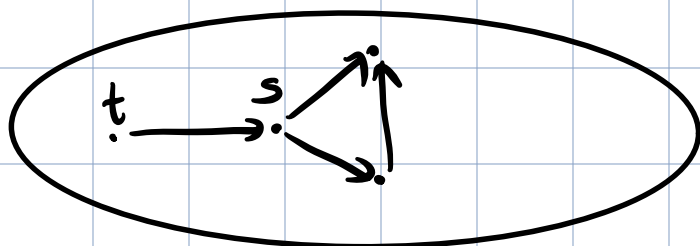
"Recurrence"
$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$



Aside 1

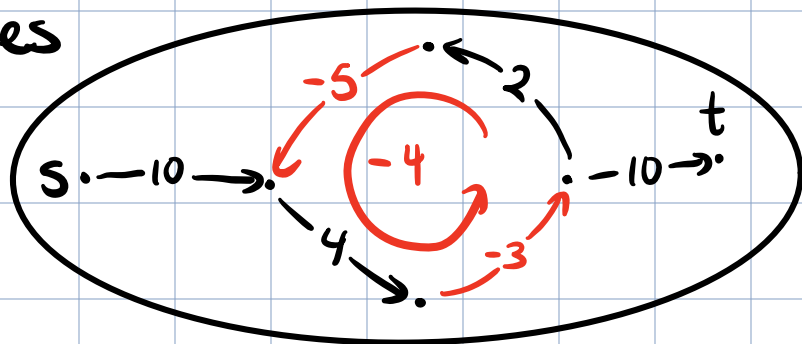
$\min \{l(w) \mid w \text{ an } (s,t)\text{-walk}\}$ may be ill-defined!

Reach:



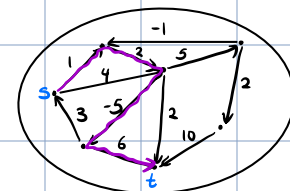
$$d(s,t) = \infty \quad \text{"min } \emptyset = \infty \text{"}$$

Negative cycles



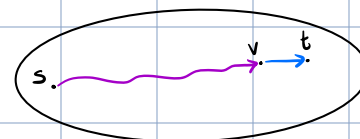
$$d(s,t) := \min \{l(w) \mid w \text{ an } (s,t)\text{-walk}\}$$

Triangle inequality: $d(s,u) + d(u,t) \geq d(s,t)$



$$d(s,t) = 2$$

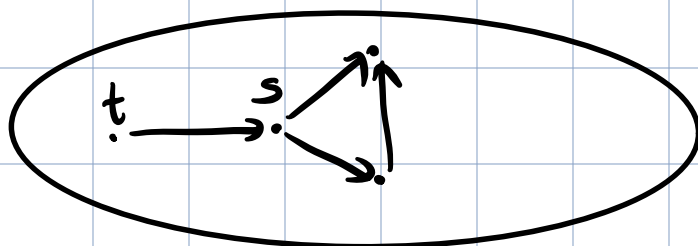
"Recurrence"
$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$



Aside 1

$\min \{l(w) \mid w \text{ an } (s,t)\text{-walk}\}$ may be ill-defined!

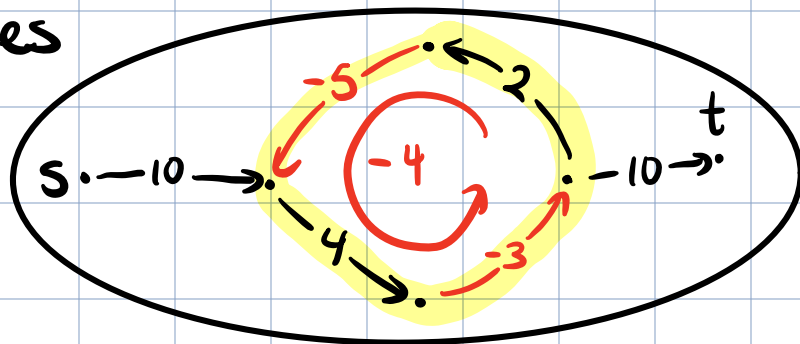
Reach:



$$d(s,t) = \infty \quad \text{"min } \emptyset = \infty \text{"}$$

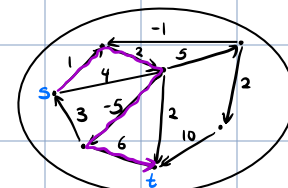
Negative cycles

$$d(s,t) = -\infty$$



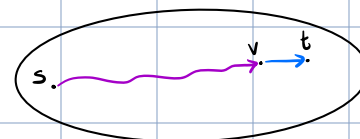
$$d(s,t) := \min \{l(w) \mid w \text{ an } (s,t)\text{-walk}\}$$

Triangle inequality: $d(s,u) + d(u,t) \geq d(s,t)$



$$d(s,t) = 2$$

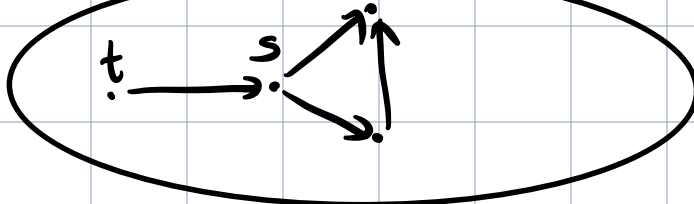
"Recurrence"
$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$



Aside 1

$\min \{l(w) \mid w \text{ an } (s,t)\text{-walk}\}$ may be ill-defined!

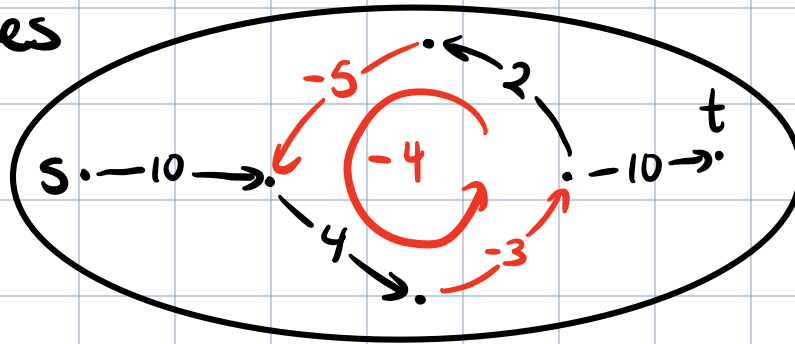
Reach:



$$d(s,t) = \infty \quad \text{"min } \emptyset = \infty \text{"}$$

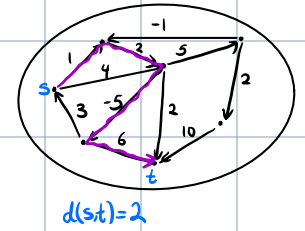
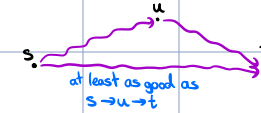
Negative cycles

$$d(s,t) = -\infty$$

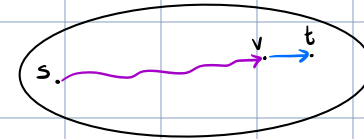


$$d(s,t) := \min \{l(w) \mid w \text{ an } (s,t)\text{-walk}\}$$

$$\text{Triangle inequality: } d(s,u) + d(u,t) \geq d(s,t)$$



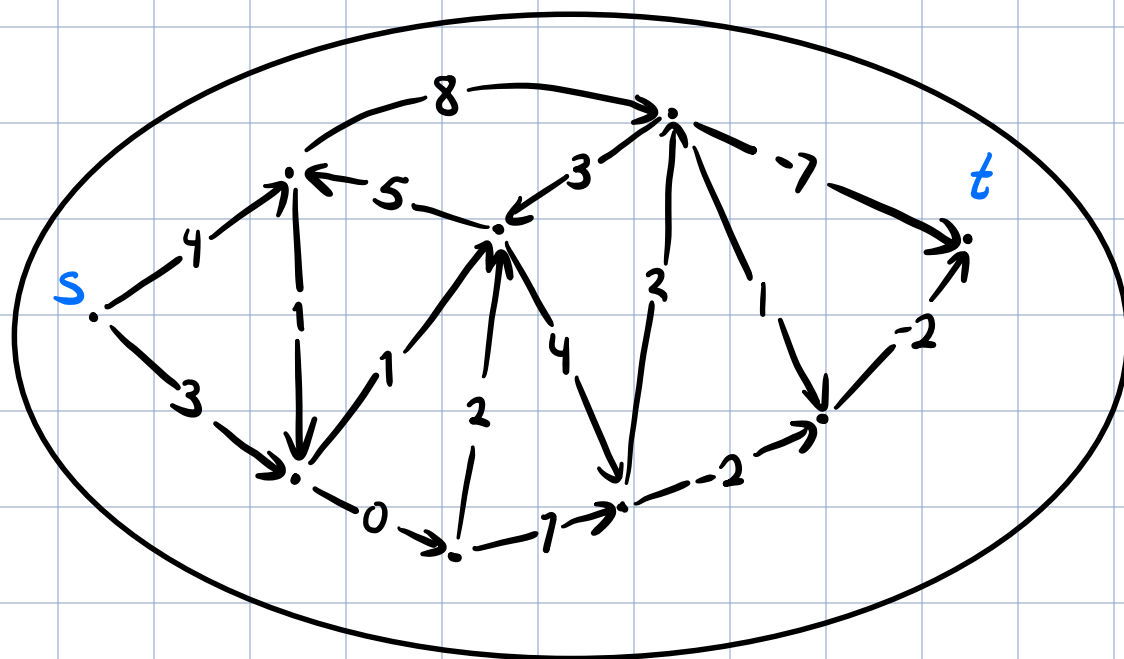
$$\text{"Recurrence" } d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$



We won't worry too much about these cases

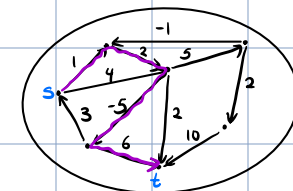
Aside 2

Distances give paths



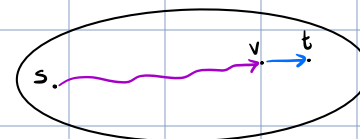
$$d(s,t) := \min \{ l(w) \mid w \text{ an } (s,t)\text{-walk} \}$$

Triangle inequality: $d(s,u) + d(u,t) \geq d(s,t)$



$$d(s,t) = 2$$

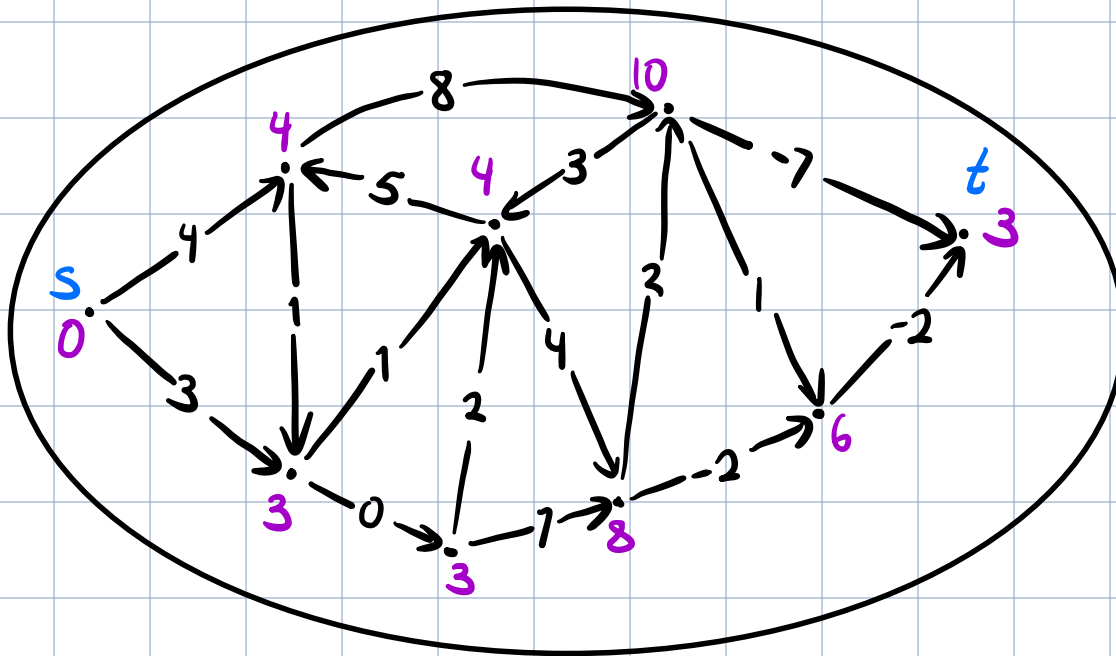
"Recurrence"
$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$



Aside 2

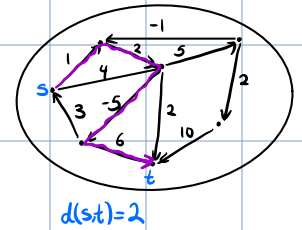
Distances give paths

Given $d(s, \cdot)$, what is the shortest (s, t) path?

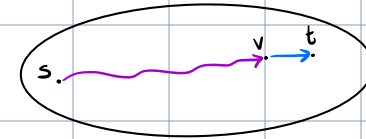


$$d(s, t) := \min \{ l(w) \mid w \text{ an } (s, t)\text{-walk} \}$$

Triangle inequality: $d(s, u) + d(u, t) \geq d(s, t)$



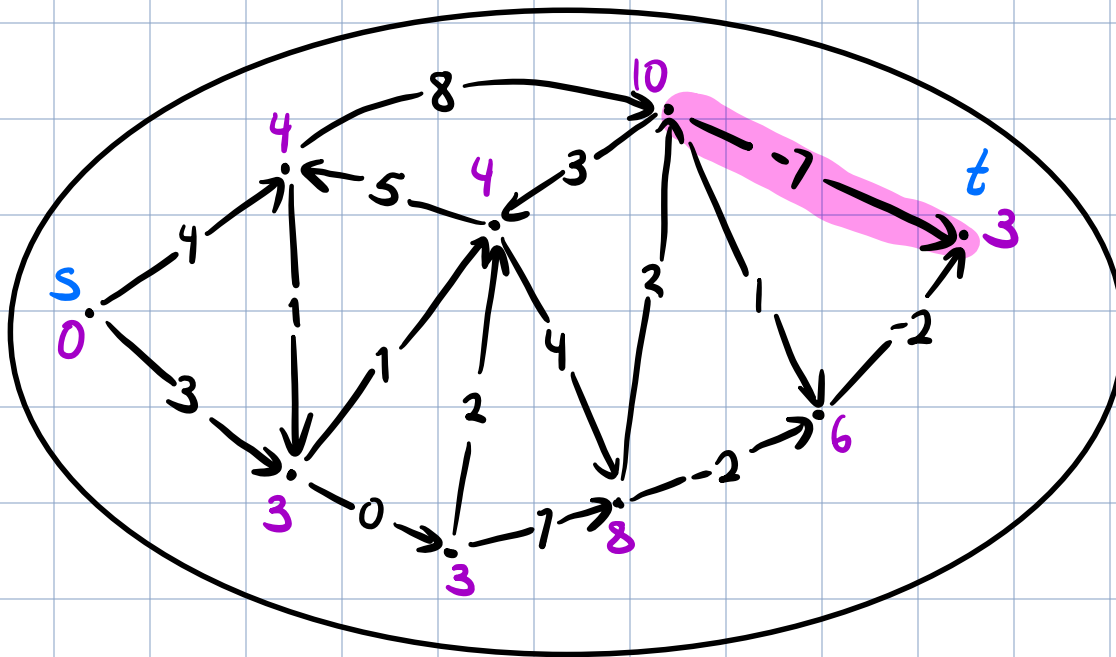
"Recurrence"
$$d(s, t) = \begin{cases} 0 & \text{if } s = t \\ \min_{(v, t) \in E} d(s, v) + l(v, t) & \text{otherwise} \end{cases}$$



Aside 2

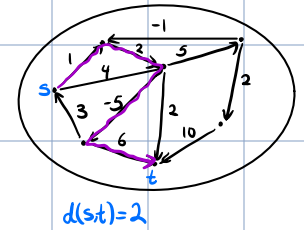
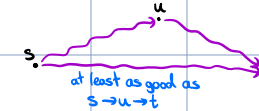
Distances give paths

Given $d(s, \cdot)$, what is the shortest (s, t) path?

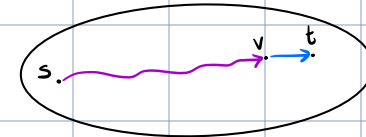


$$d(s, t) := \min \{ l(w) \mid w \text{ an } (s, t)\text{-walk} \}$$

Triangle inequality: $d(s, u) + d(u, t) \geq d(s, t)$



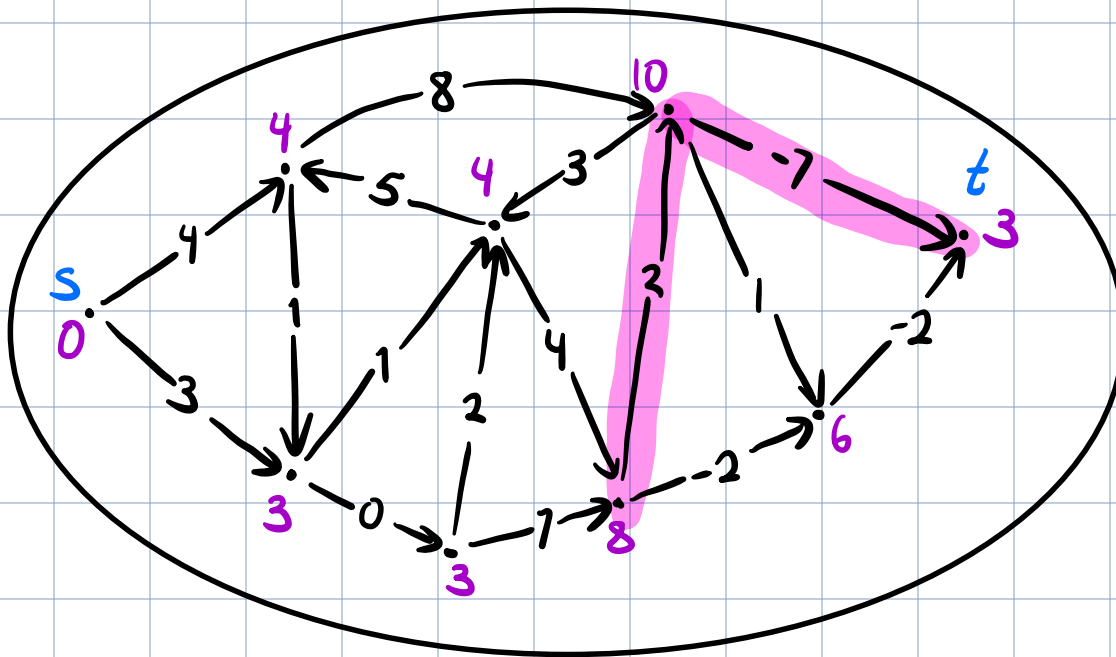
"Recurrence"
$$d(s, t) = \begin{cases} 0 & \text{if } s = t \\ \min_{(v, t) \in E} d(s, v) + l(v, t) & \text{otherwise} \end{cases}$$



Aside 2

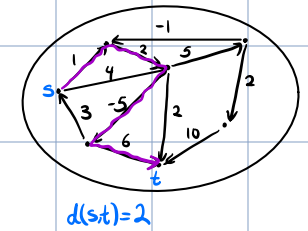
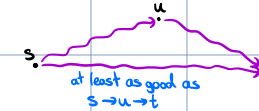
Distances give paths

Given $d(s, \cdot)$, what is the shortest (s, t) path?

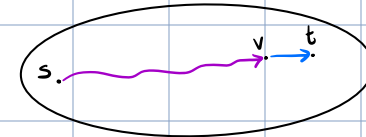


$$d(s, t) := \min \{ l(w) \mid w \text{ an } (s, t)\text{-walk} \}$$

$$\text{Triangle inequality: } d(s, u) + d(u, t) \geq d(s, t)$$



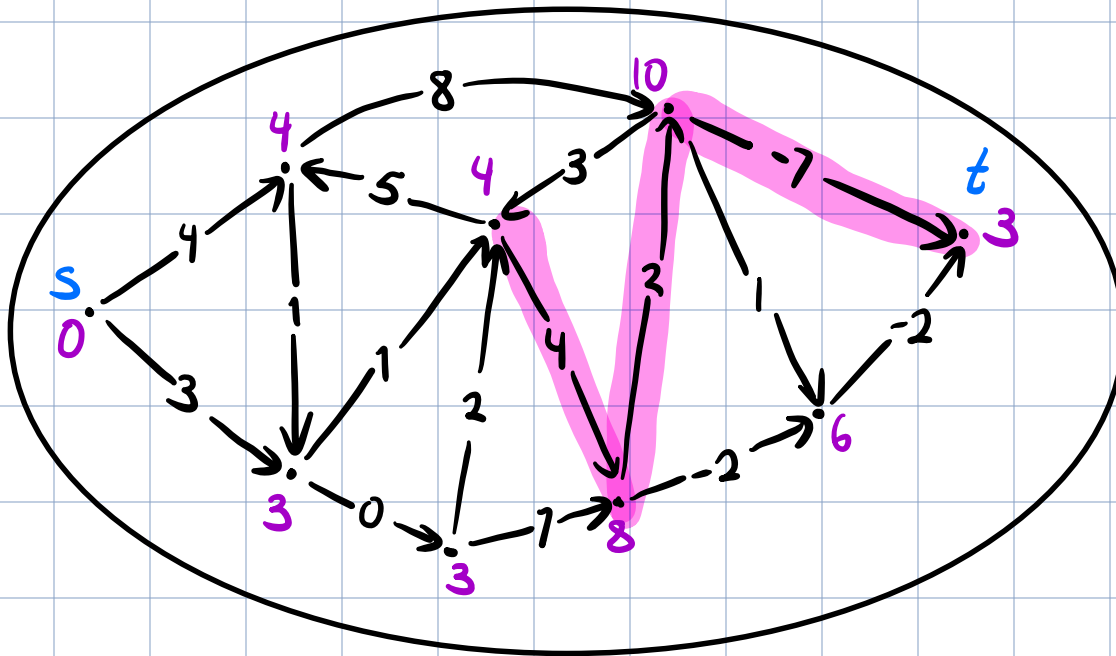
$$\text{"Recurrence"} \quad d(s, t) = \begin{cases} 0 & \text{if } s = t \\ \min_{(v, t) \in E} d(s, v) + l(v, t) & \text{otherwise} \end{cases}$$



Aside 2

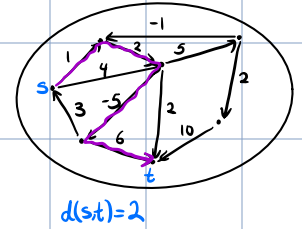
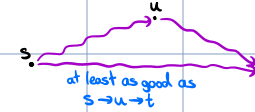
Distances give paths

Given $d(s, \cdot)$, what is the shortest (s, t) path?

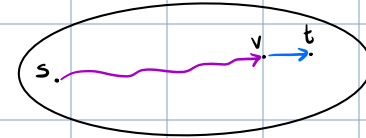


$$d(s, t) := \min \{ l(w) \mid w \text{ an } (s, t)\text{-walk} \}$$

$$\text{Triangle inequality: } d(s, u) + d(u, t) \geq d(s, t)$$



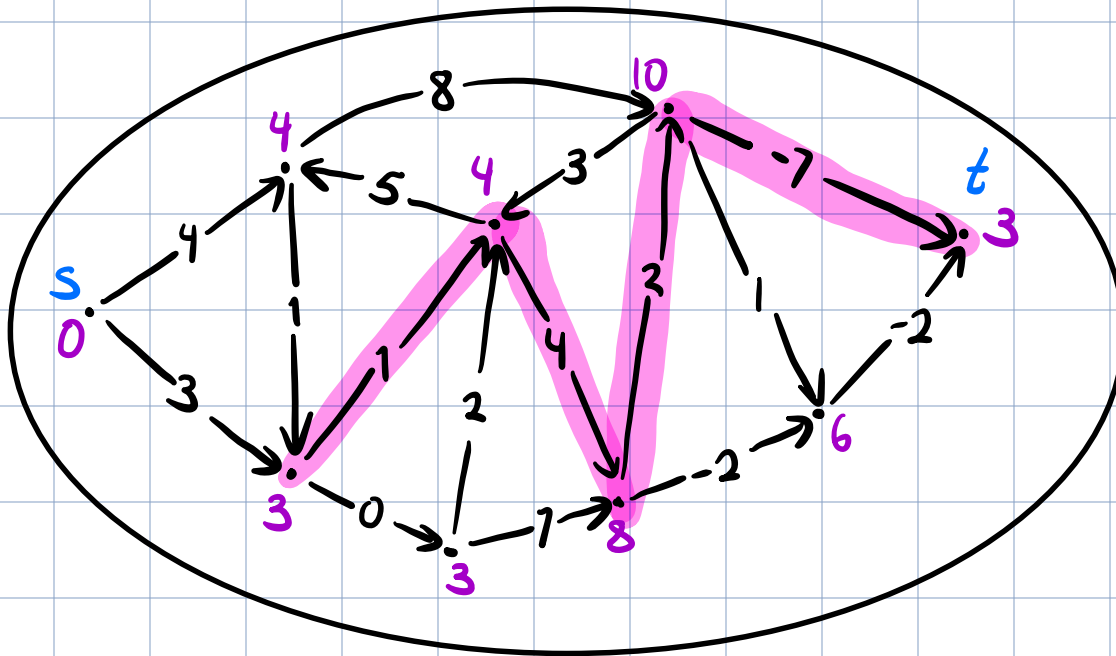
$$\text{"Recurrence"} \quad d(s, t) = \begin{cases} 0 & \text{if } s = t \\ \min_{(v, t) \in E} d(s, v) + l(v, t) & \text{otherwise} \end{cases}$$



Aside 2

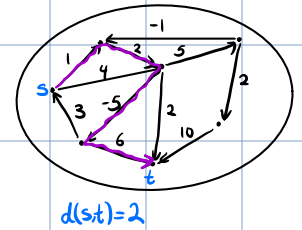
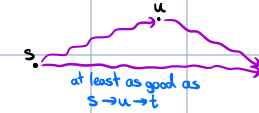
Distances give paths

Given $d(s, \cdot)$, what is the shortest (s, t) path?

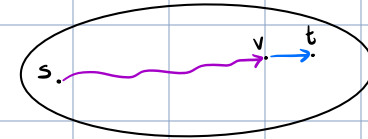


$$d(s, t) := \min \{ l(w) \mid w \text{ an } (s, t)\text{-walk} \}$$

$$\text{Triangle inequality: } d(s, u) + d(u, t) \geq d(s, t)$$



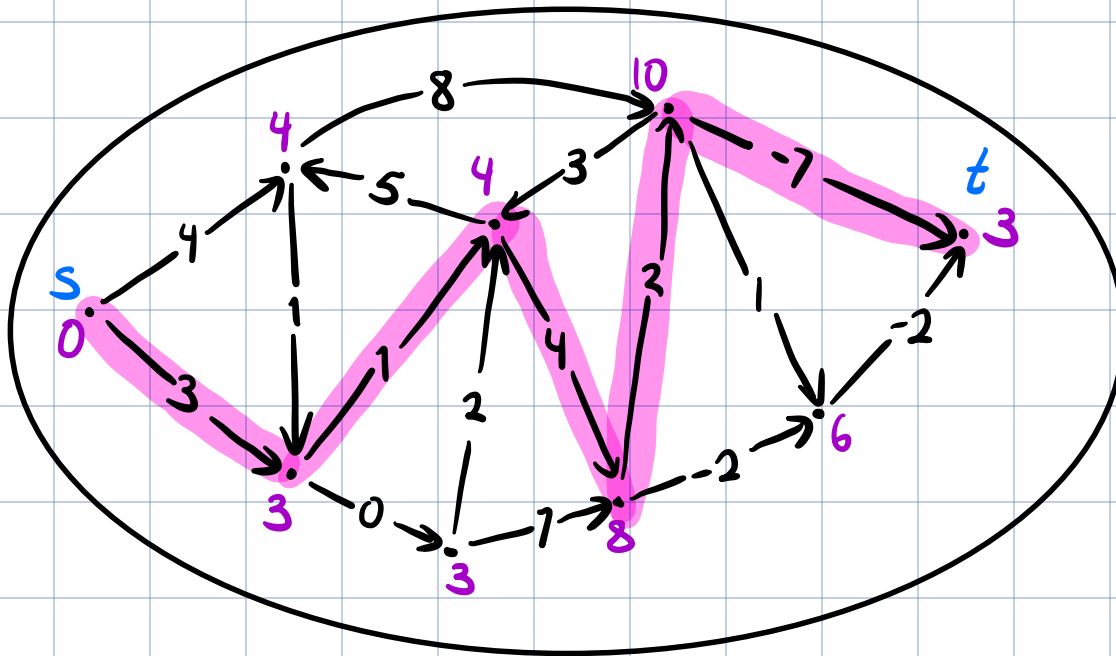
$$\text{"Recurrence"} \quad d(s, t) = \begin{cases} 0 & \text{if } s = t \\ \min_{(v, t) \in E} d(s, v) + l(v, t) & \text{otherwise} \end{cases}$$



Aside 2

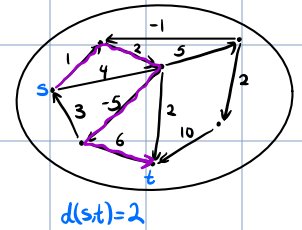
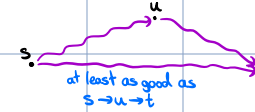
Distances give paths

Given $d(s, \cdot)$, what is the shortest (s, t) path?

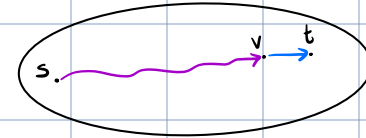


$$d(s, t) := \min \{ l(w) \mid w \text{ an } (s, t)\text{-walk} \}$$

$$\text{Triangle inequality: } d(s, u) + d(u, t) \geq d(s, t)$$



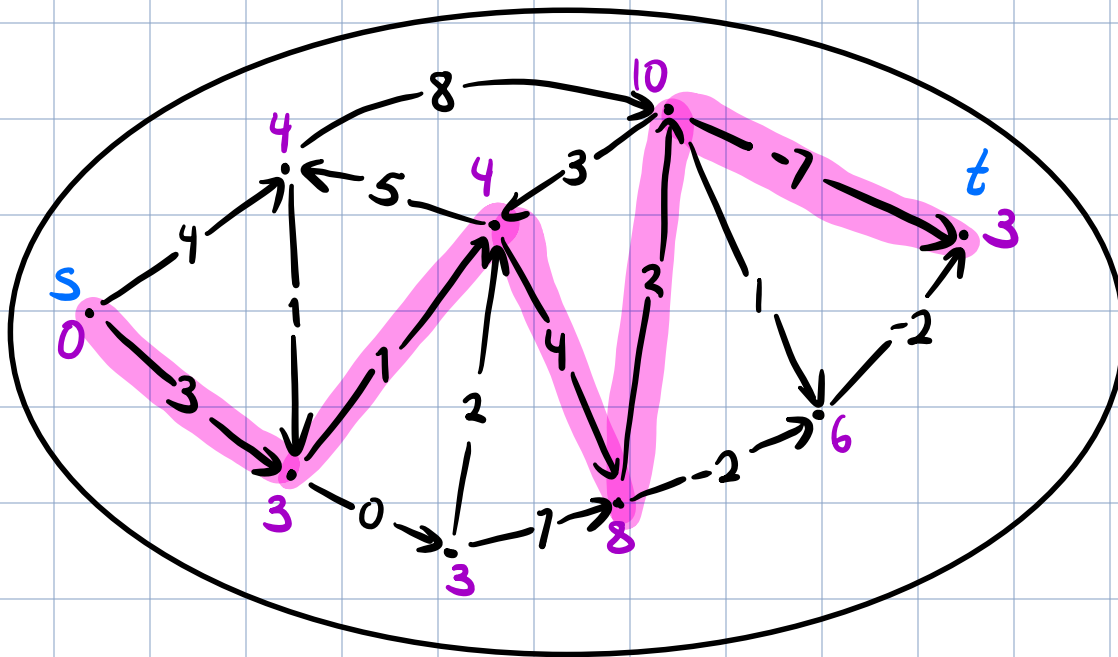
$$\text{"Recurrence"} \quad d(s, t) = \begin{cases} 0 & \text{if } s = t \\ \min_{(v, t) \in E} d(s, v) + l(v, t) & \text{otherwise} \end{cases}$$



Aside 2

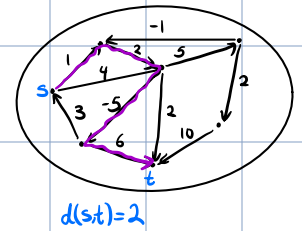
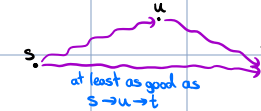
Distances give paths

Given $d(s, \cdot)$, what is the shortest (s, t) path?

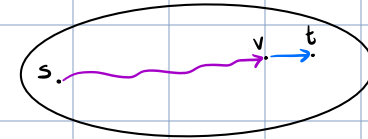


$$d(s, t) := \min \{ l(w) \mid w \text{ an } (s, t)\text{-walk} \}$$

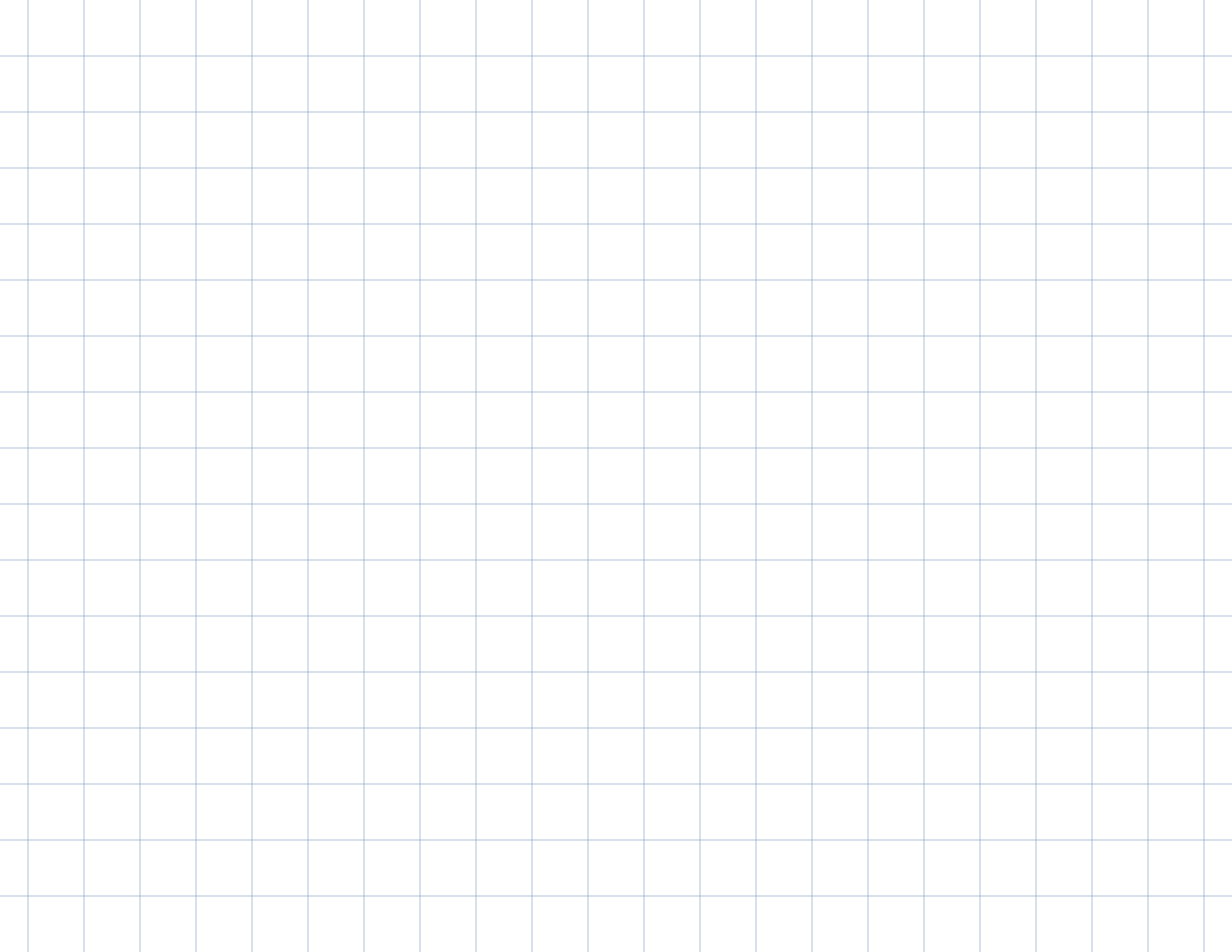
Triangle inequality: $d(s, u) + d(u, t) \geq d(s, t)$



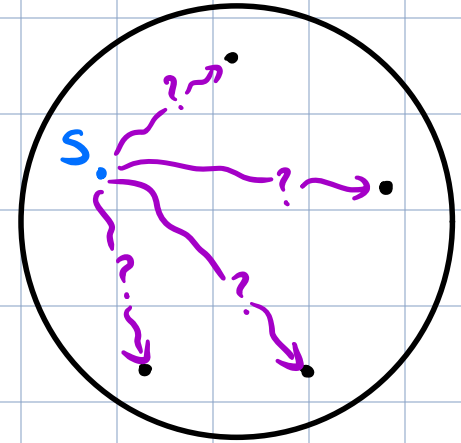
"Recurrence"
$$d(s, t) = \begin{cases} 0 & \text{if } s = t \\ \min_{(v, t) \in E} d(s, v) + l(v, t) & \text{otherwise} \end{cases}$$



So we will only focus on finding distances

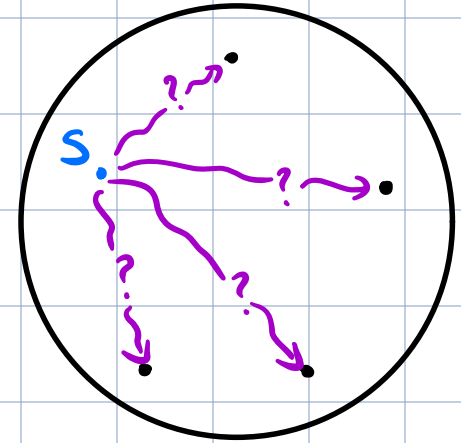


Single Source Shortest Path (SSSP): Find $d(s, \cdot)$



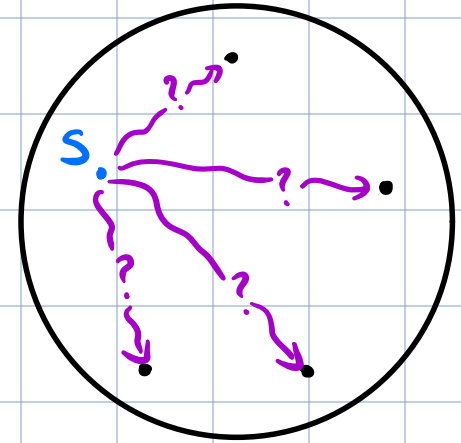
Single Source Shortest Path (SSSP): Find $d(s, \cdot)$

Bellman-Ford:



Single Source Shortest Path (SSSP): Find $d(s, \cdot)$

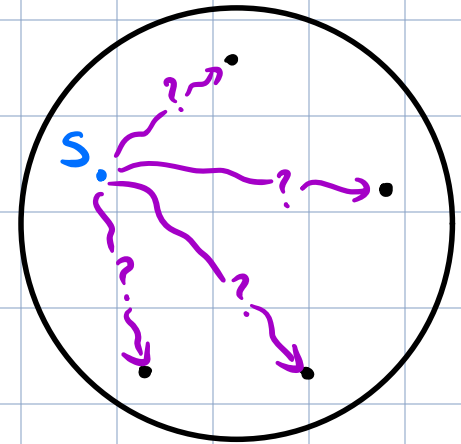
Bellman-Ford: $O(nm)$



Single Source Shortest Path (SSSP): Find $d(s, \cdot)$

Bellman-Ford: $O(nm)$

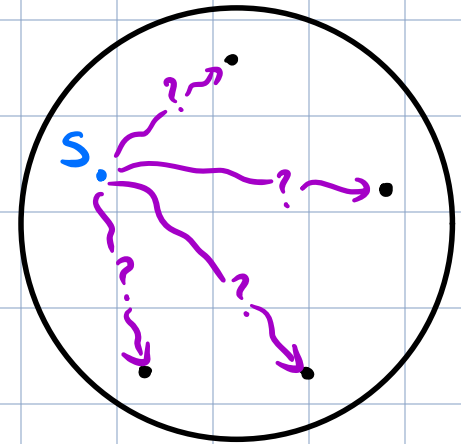
If $l(e) \geq 0$, Dijkstra's:



Single Source Shortest Path (SSSP): Find $d(s, \cdot)$

Bellman-Ford: $O(nm)$

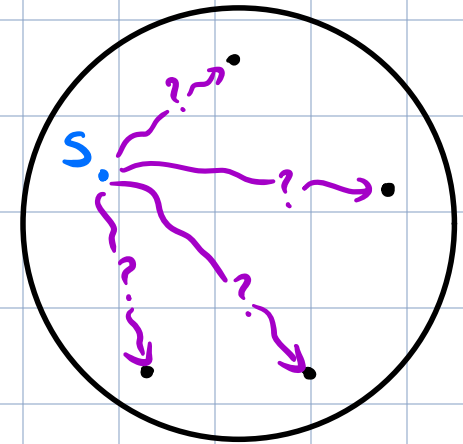
If $l(e) \geq 0$, Dijkstra's: $O(m + n \log n)$



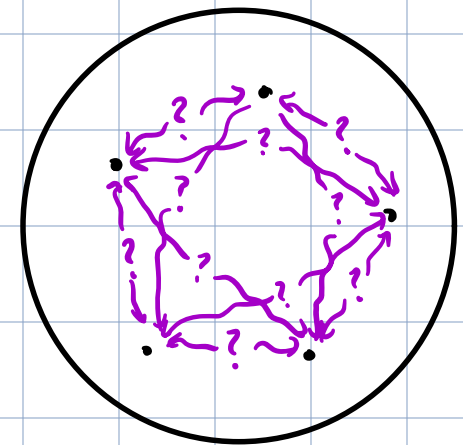
Single Source Shortest Path (SSSP): Find $d(s, \cdot)$

Bellman-Ford: $O(nm)$

If $l(e) \geq 0$, Dijkstra's: $O(m + n \log n)$



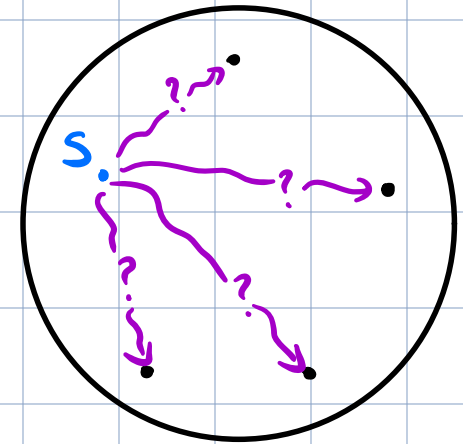
All Pairs Shortest Path (APSP): Find $d(\cdot, \cdot)$



Single Source Shortest Path (SSSP): Find $d(s, \cdot)$

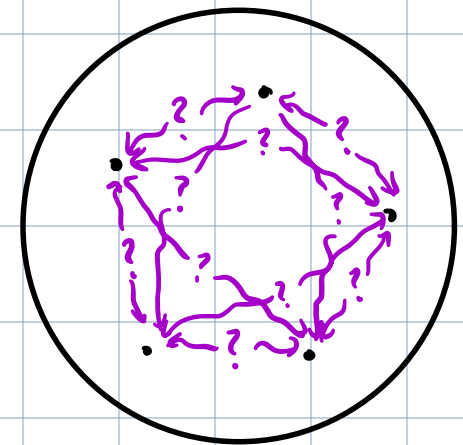
Bellman-Ford: $O(nm)$

If $l(e) \geq 0$, Dijkstra's: $O(m + n \log n)$



All Pairs Shortest Path (APSP): Find $d(\cdot, \cdot)$

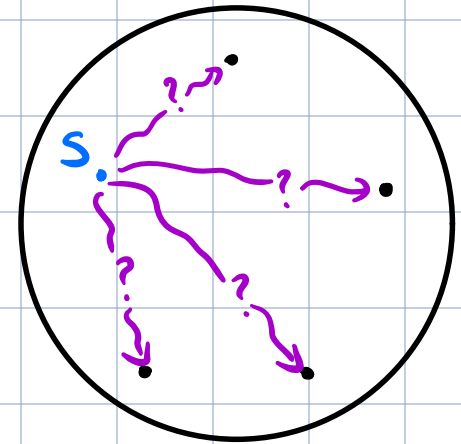
Floyd-Warshall:



Single Source Shortest Path (SSSP): Find $d(s, \cdot)$

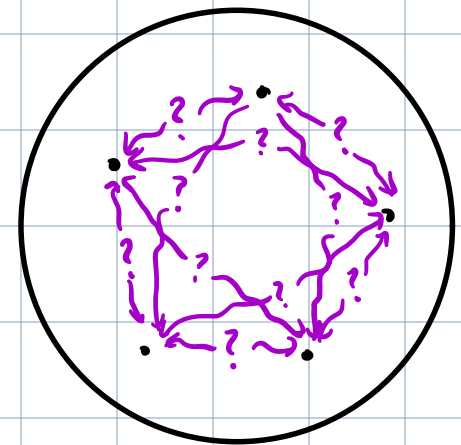
Bellman-Ford: $O(nm)$

If $l(e) \geq 0$, Dijkstra's: $O(m + n \log n)$



All Pairs Shortest Path (APSP): Find $d(\cdot, \cdot)$

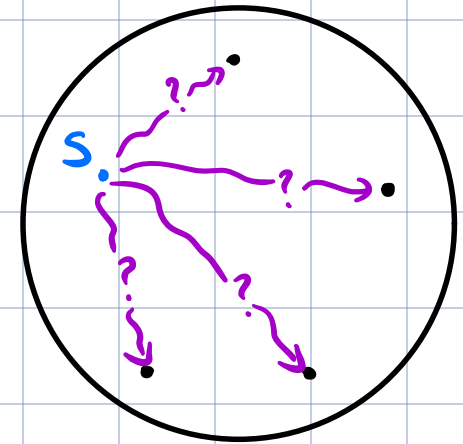
Floyd-Warshall: $O(n^3)$



Single Source Shortest Path (SSSP): Find $d(s, \cdot)$

Bellman-Ford: $O(nm)$

If $l(e) \geq 0$, Dijkstra's: $O(m + n \log n)$

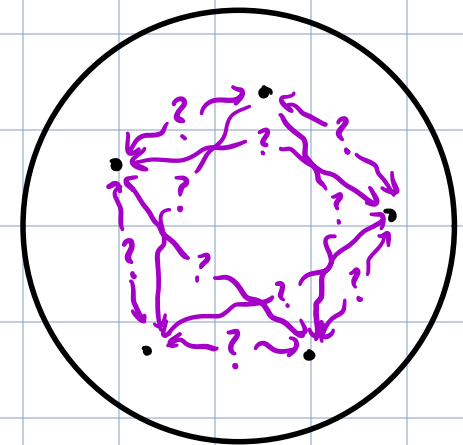


All Pairs Shortest Path (APSP): Find $d(\cdot, \cdot)$

Floyd-Warshall: $O(n^3)$

Johnson's Algorithm:

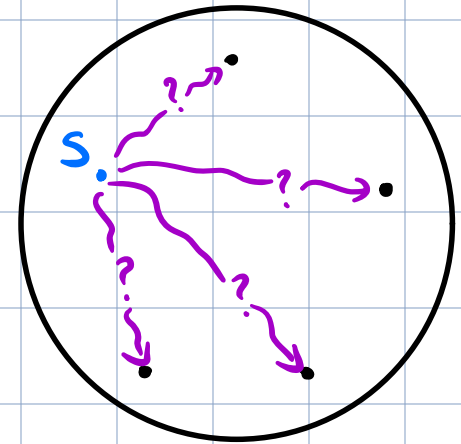
← soon!



Single Source Shortest Path (SSSP): Find $d(s, \cdot)$

Bellman-Ford: $O(nm)$

If $l(e) \geq 0$, Dijkstra's: $O(m + n \log n)$

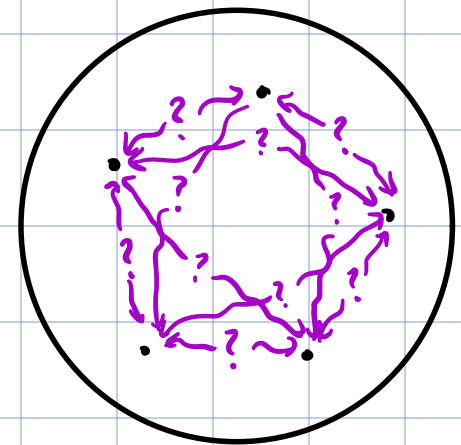


All Pairs Shortest Path (APSP): Find $d(\cdot, \cdot)$

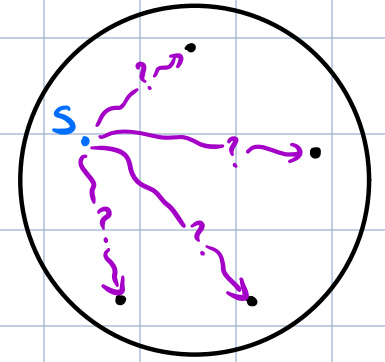
Floyd-Warshall: $O(n^3)$

Johnson's Algorithm: $O(nm + n^2 \log n)$

← soon!

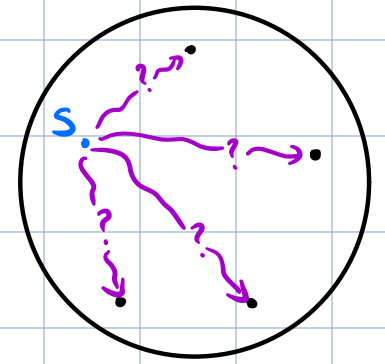


Bellman-Ford: Stretching the distance recurrence

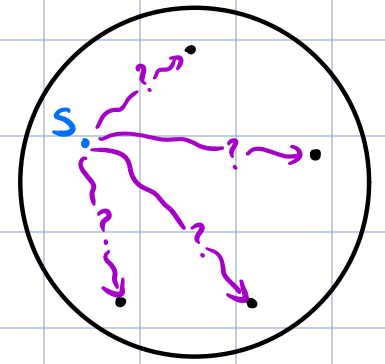


Bellman-Ford: Stretching the distance recurrence

$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$



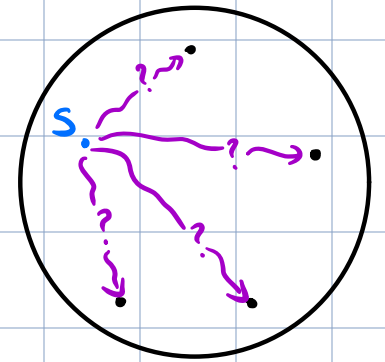
Bellman-Ford: Stretching the distance recurrence



$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$

$$\text{dist}(v,i) := \min \{ l(W) \mid W \text{ an } (s,v)\text{-walk with at most } i \text{ edges} \}$$

Bellman-Ford: Stretching the distance recurrence

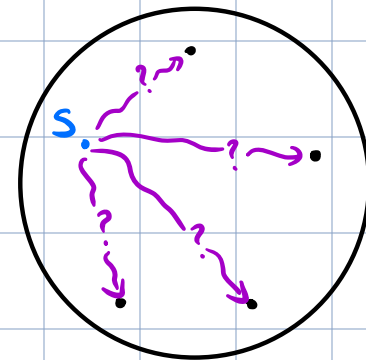


$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$

$$\text{dist}(v,i) := \min \{ l(W) \mid W \text{ an } (s,v)\text{-walk with at most } i \text{ edges} \}$$

$$\text{dist}(v,i) = \begin{cases} 0 & \text{if } i=0 \text{ \& } s=v \\ \infty & \text{if } i=0 \text{ \& } s \neq v \\ \min_{(u,v) \in E} \text{dist}(v,i-1) + l(u,v) & \text{otherwise} \end{cases}$$

Bellman-Ford: Stretching the distance recurrence

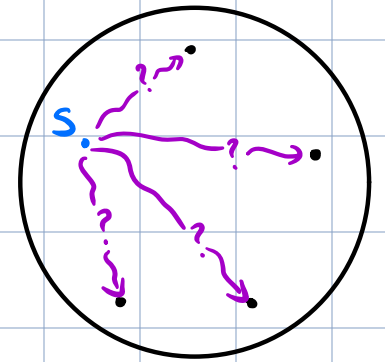


$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$

$$\text{dist}(v,i) := \min \{ l(W) \mid W \text{ an } (s,v)\text{-walk with at most } i \text{ edges} \}$$

$$\text{dist}(v,i) = \begin{cases} 0 & \text{if } i=0 \text{ \& } s=v \\ \infty & \text{if } i=0 \text{ \& } s \neq v, \text{ } d(s,t) = \text{dist}(t,n-1) \\ \min_{(u,v) \in E} \text{dist}(v,i-1) + l(u,v) & \text{otherwise} \end{cases}$$

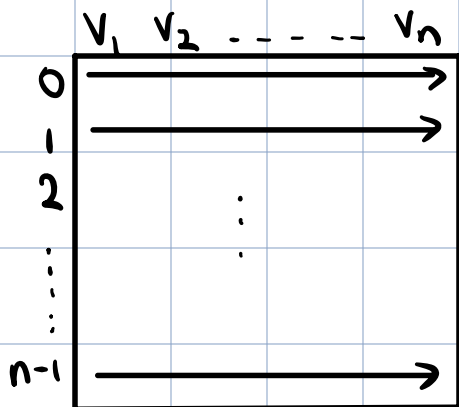
Bellman-Ford: Stretching the distance recurrence



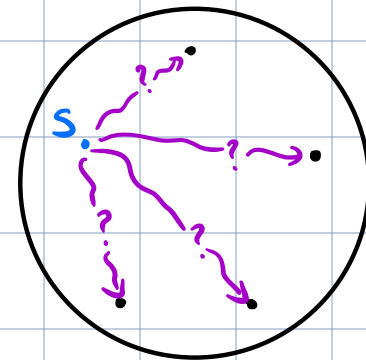
$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$

$$\text{dist}(v,i) := \min \{ l(W) \mid W \text{ an } (s,v)\text{-walk with at most } i \text{ edges} \}$$

$$\text{dist}(v,i) = \begin{cases} 0 & \text{if } i=0 \text{ \& } s=v \\ \infty & \text{if } i=0 \text{ \& } s \neq v, \text{ } d(s,t) = \text{dist}(t, n-1) \\ \min_{(u,v) \in E} \text{dist}(v, i-1) + l(u,v) & \text{otherwise} \end{cases}$$



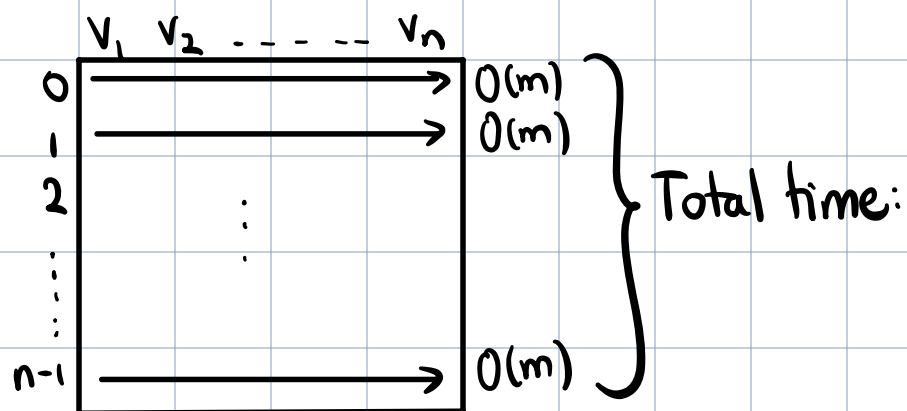
Bellman-Ford: Stretching the distance recurrence



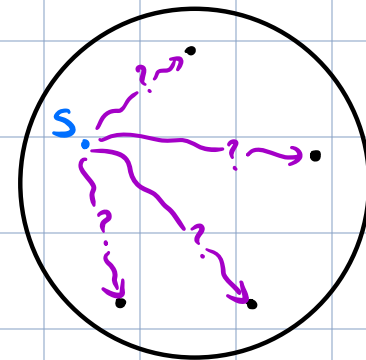
$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$

$$\text{dist}(v,i) := \min \{ l(W) \mid W \text{ an } (s,v)\text{-walk with at most } i \text{ edges} \}$$

$$\text{dist}(v,i) = \begin{cases} 0 & \text{if } i=0 \text{ \& } s=v \\ \infty & \text{if } i=0 \text{ \& } s \neq v, \text{ } d(s,t) = \text{dist}(t, n-1) \\ \min_{(u,v) \in E} \text{dist}(v, i-1) + l(u,v) & \text{otherwise} \end{cases}$$



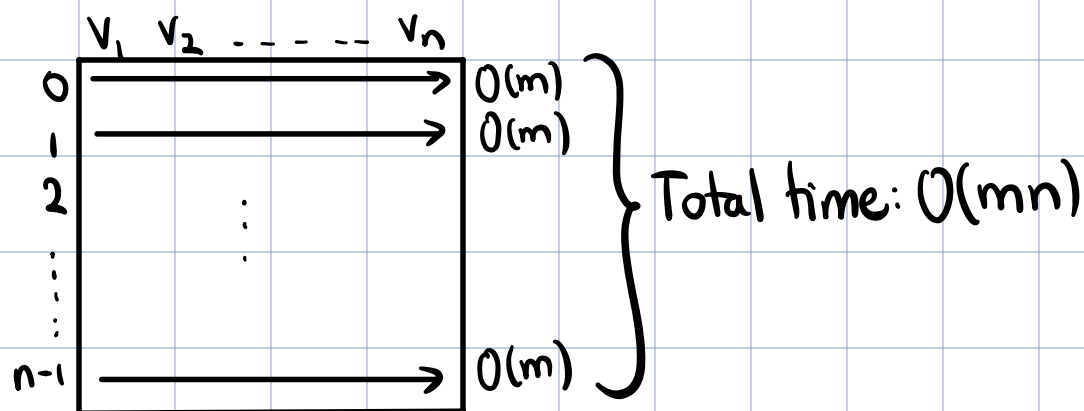
Bellman-Ford: Stretching the distance recurrence



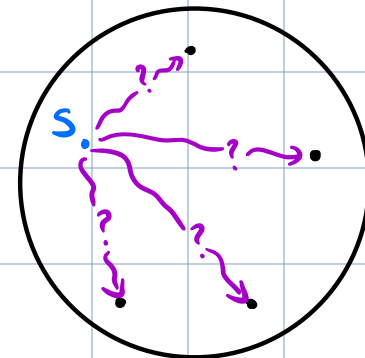
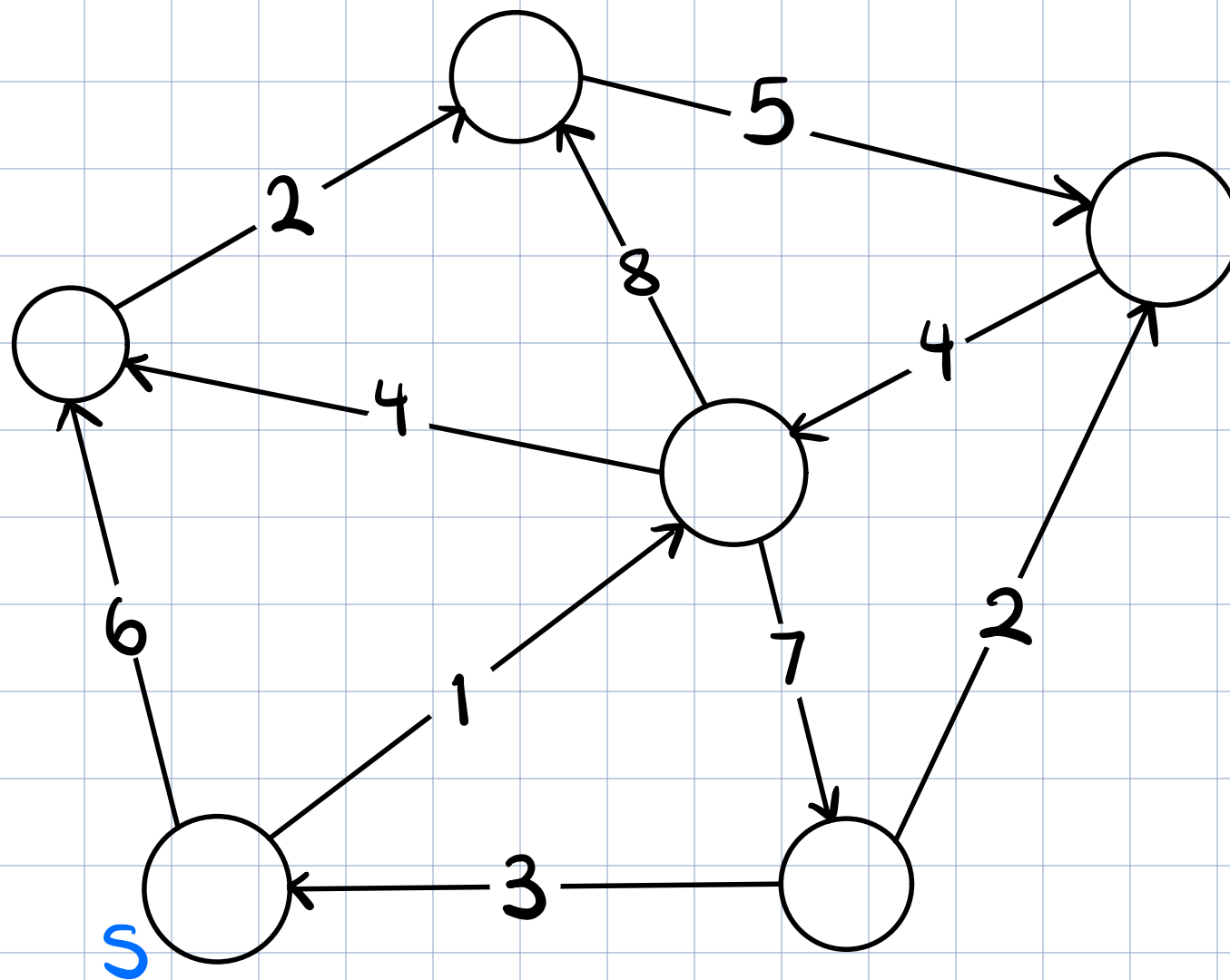
$$d(s,t) = \begin{cases} 0 & \text{if } s=t \\ \min_{(v,t) \in E} d(s,v) + l(v,t) & \text{otherwise} \end{cases}$$

$$\text{dist}(v,i) := \min \{ l(W) \mid W \text{ an } (s,v)\text{-walk with at most } i \text{ edges} \}$$

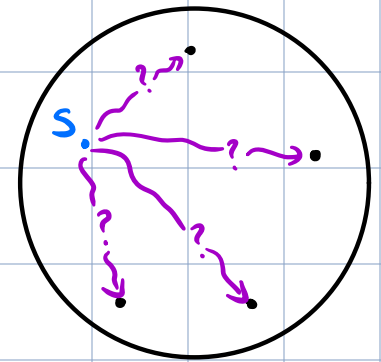
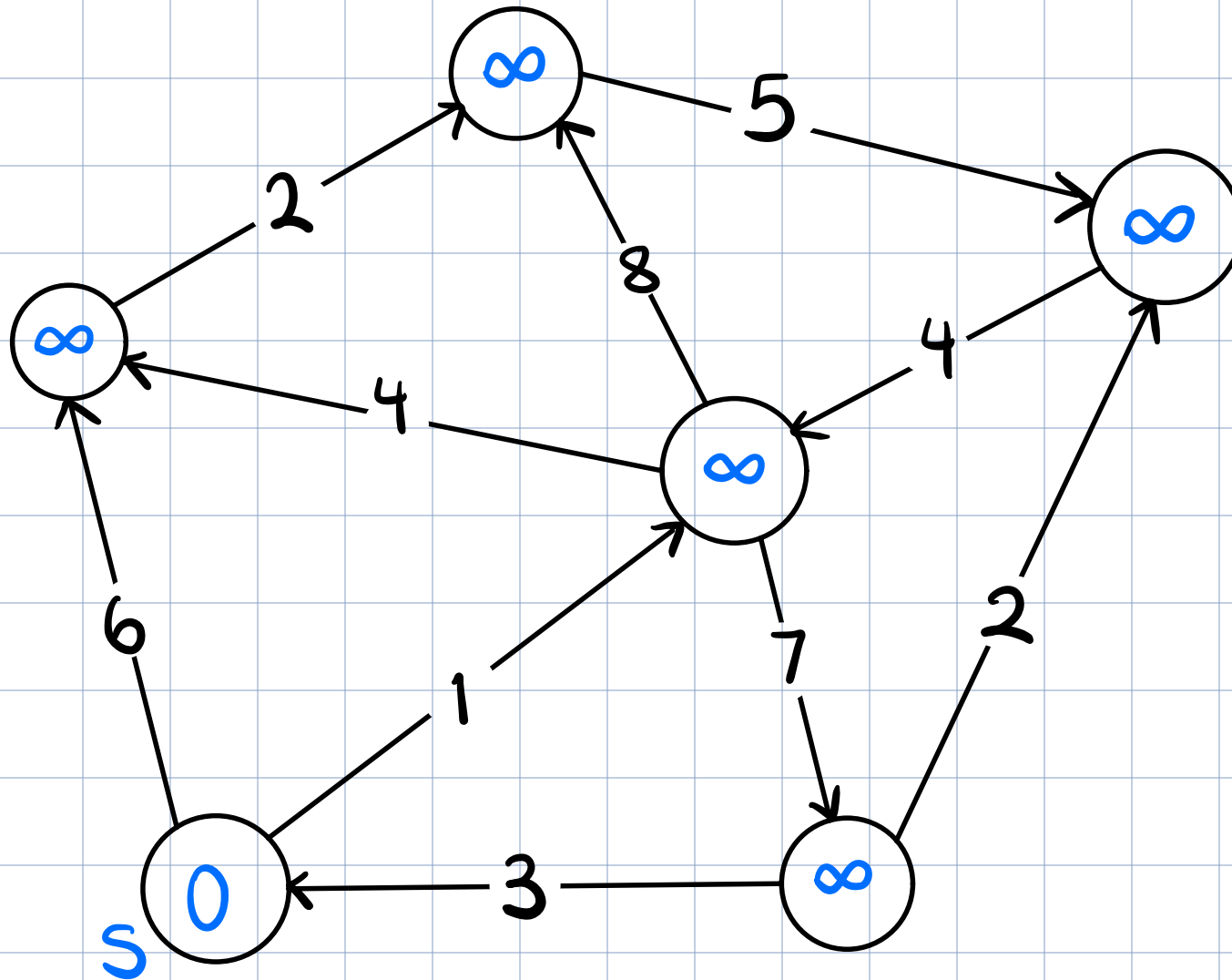
$$\text{dist}(v,i) = \begin{cases} 0 & \text{if } i=0 \text{ \& } s=v \\ \infty & \text{if } i=0 \text{ \& } s \neq v, \text{ } d(s,t) = \text{dist}(t, n-1) \\ \min_{(u,v) \in E} \text{dist}(v, i-1) + l(u,v) & \text{otherwise} \end{cases}$$



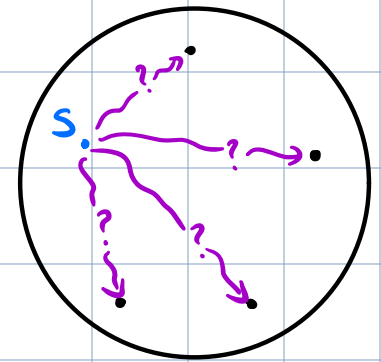
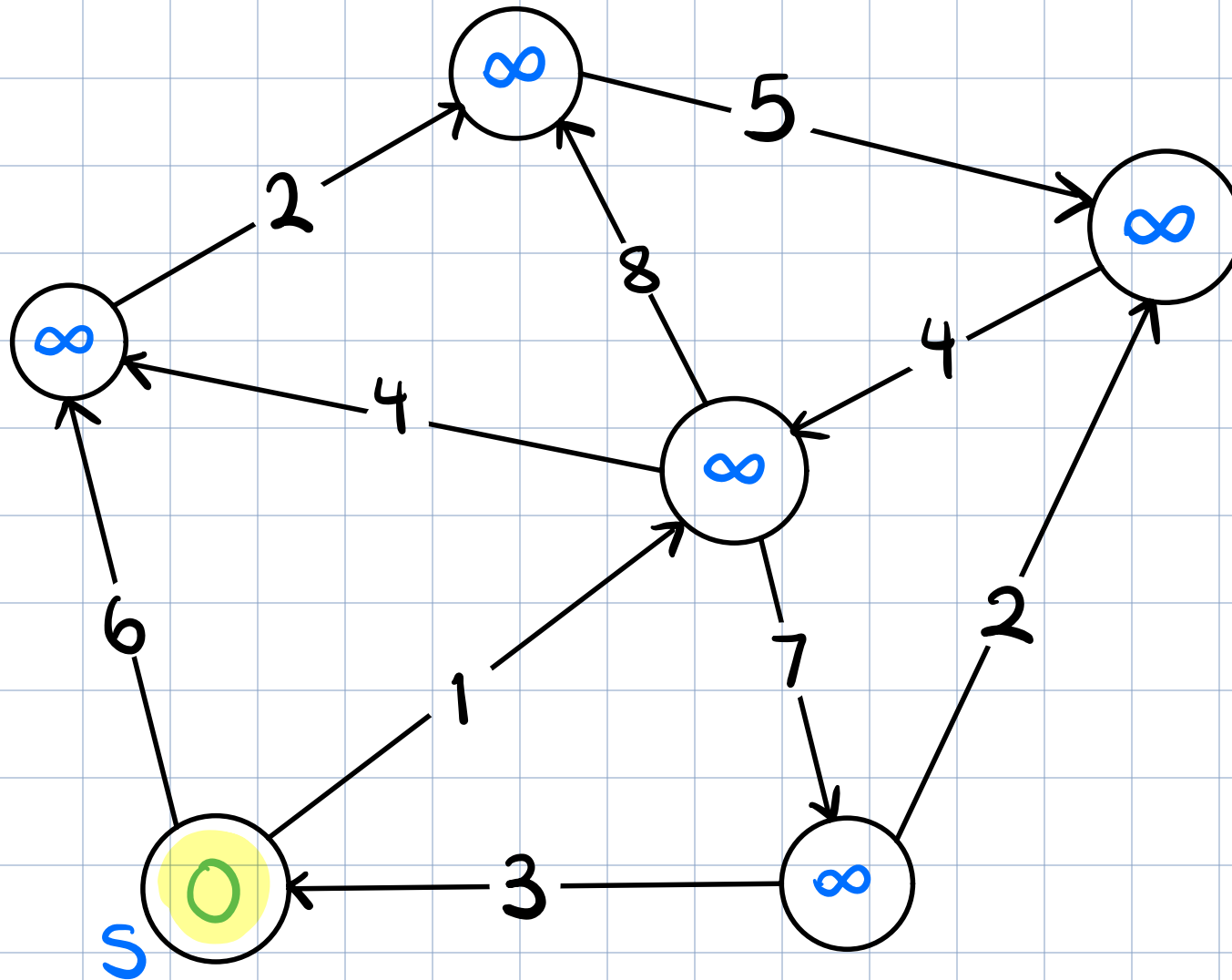
Dijkstra's



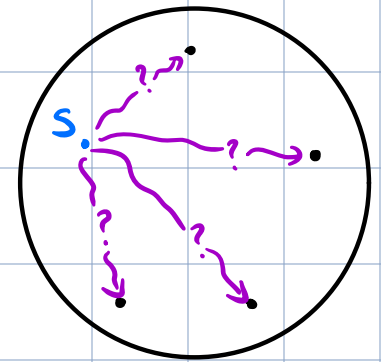
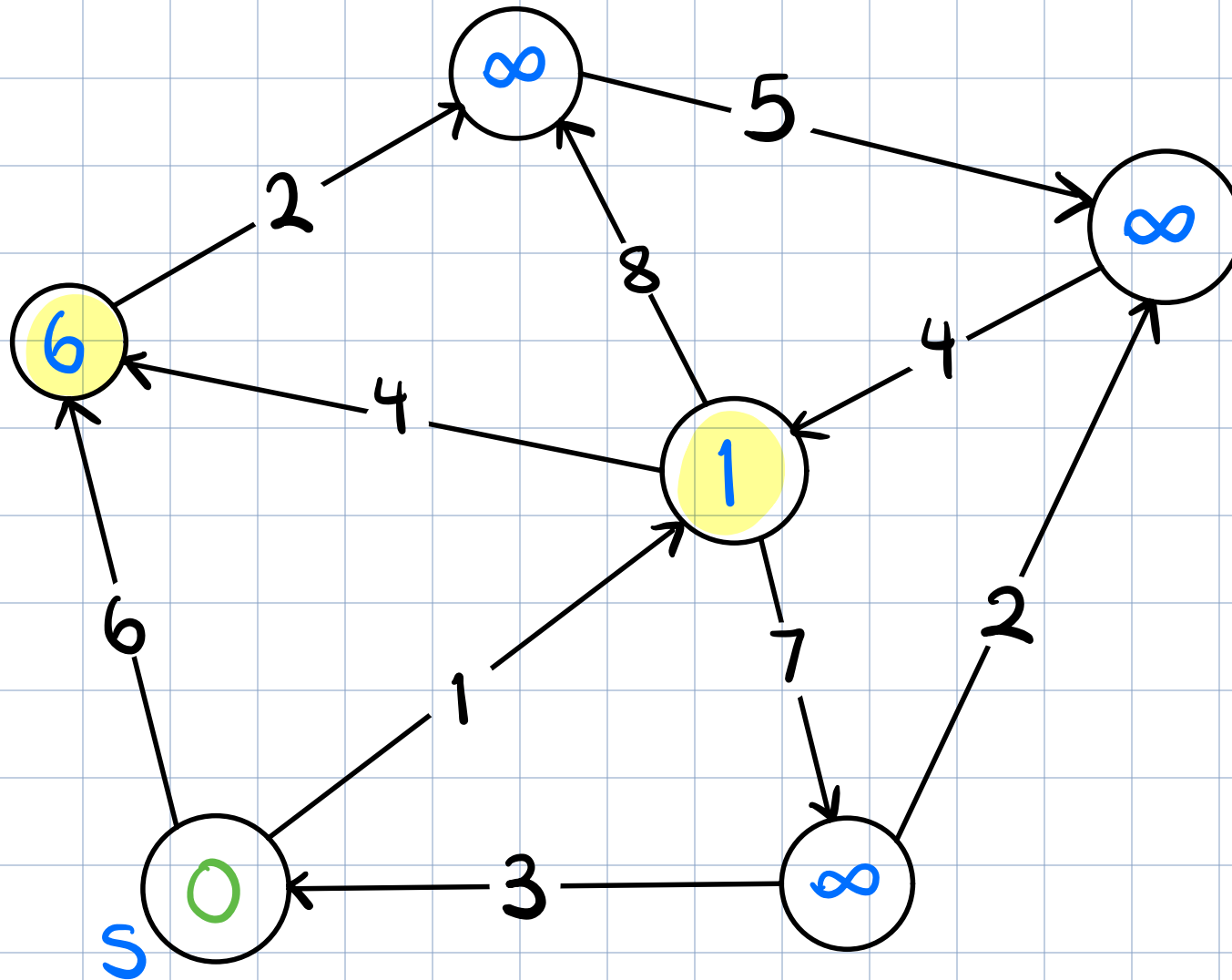
Dijkstra's



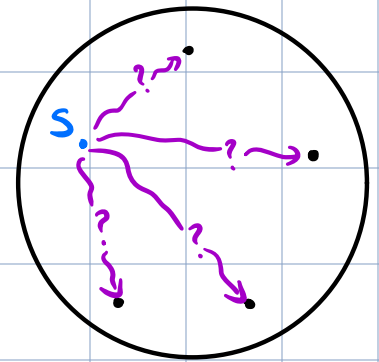
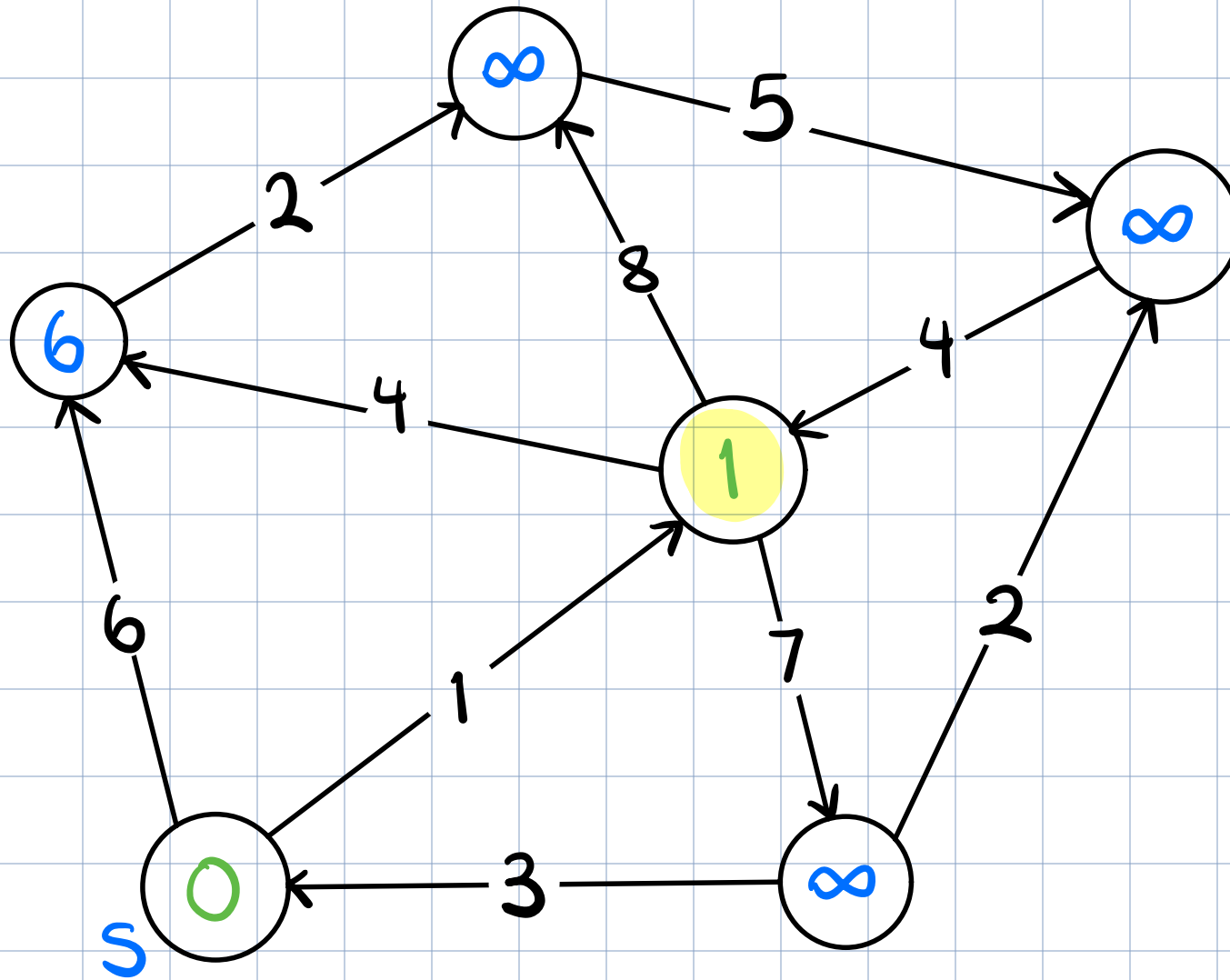
Dijkstra's



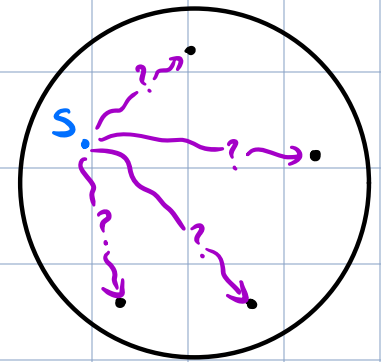
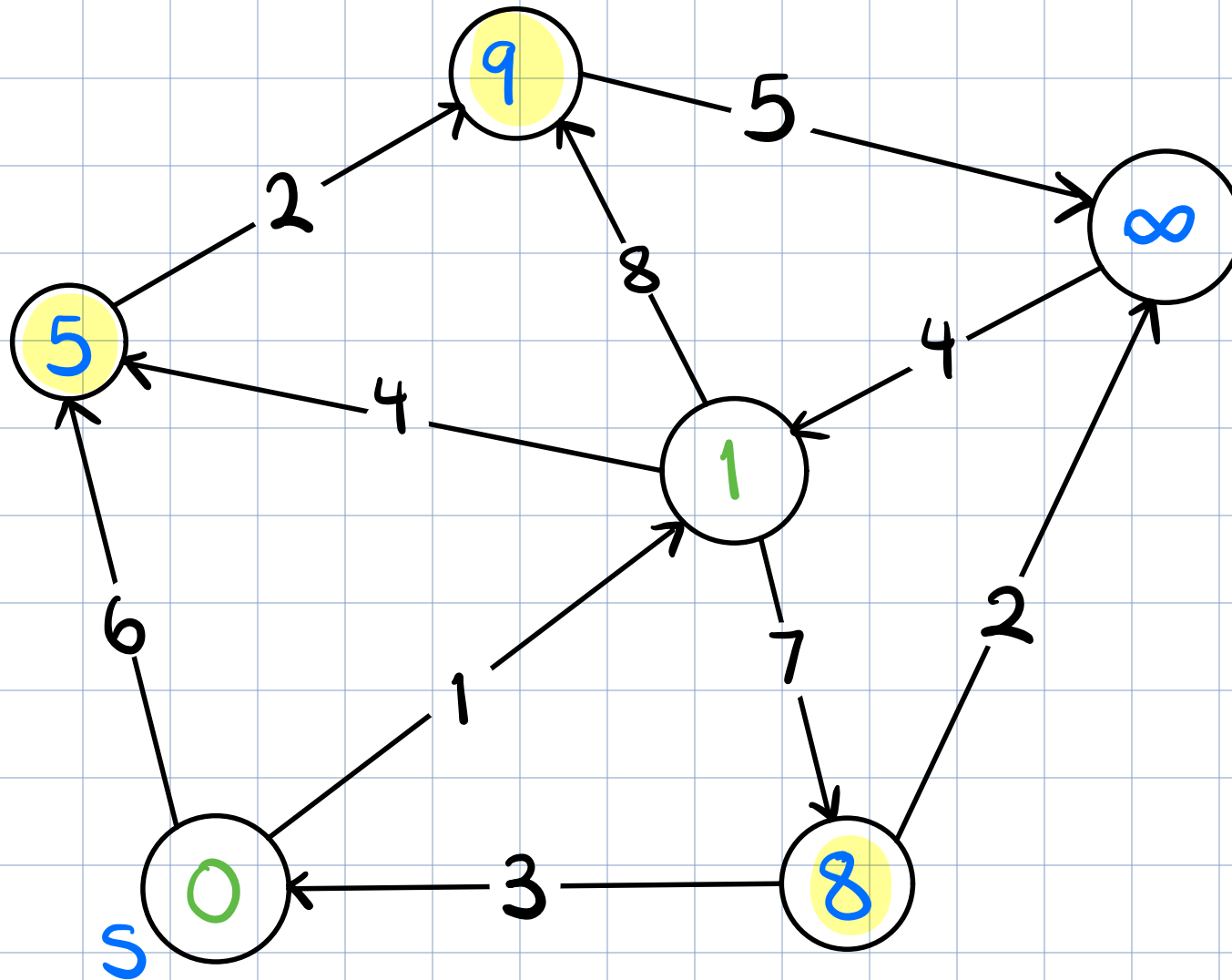
Dijkstra's



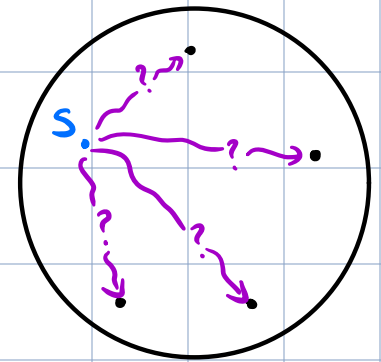
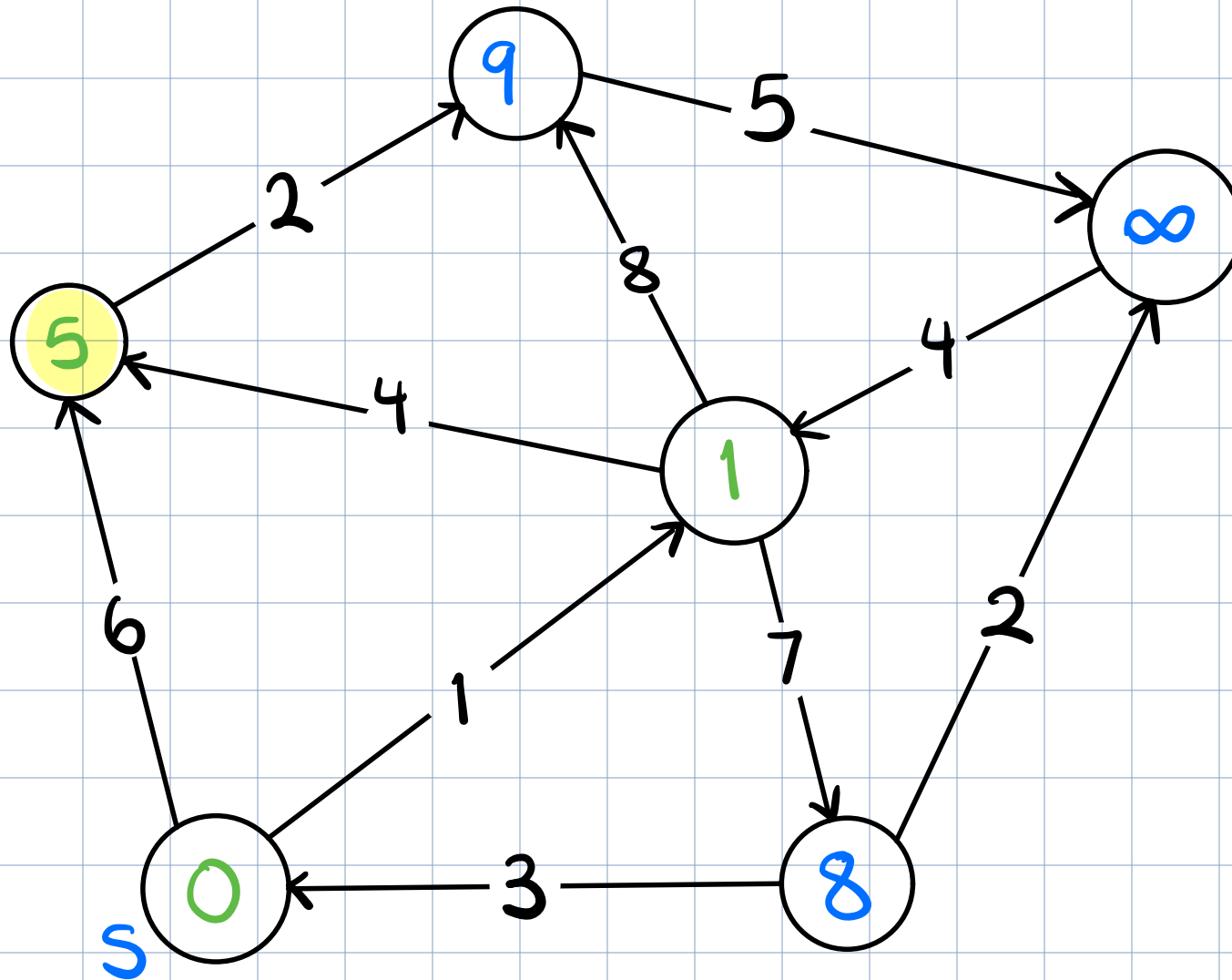
Dijkstra's



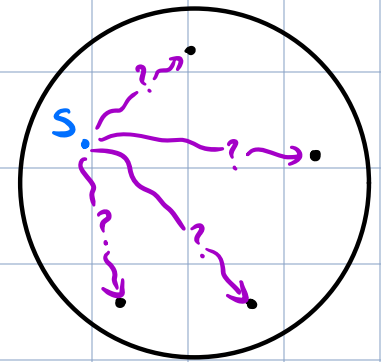
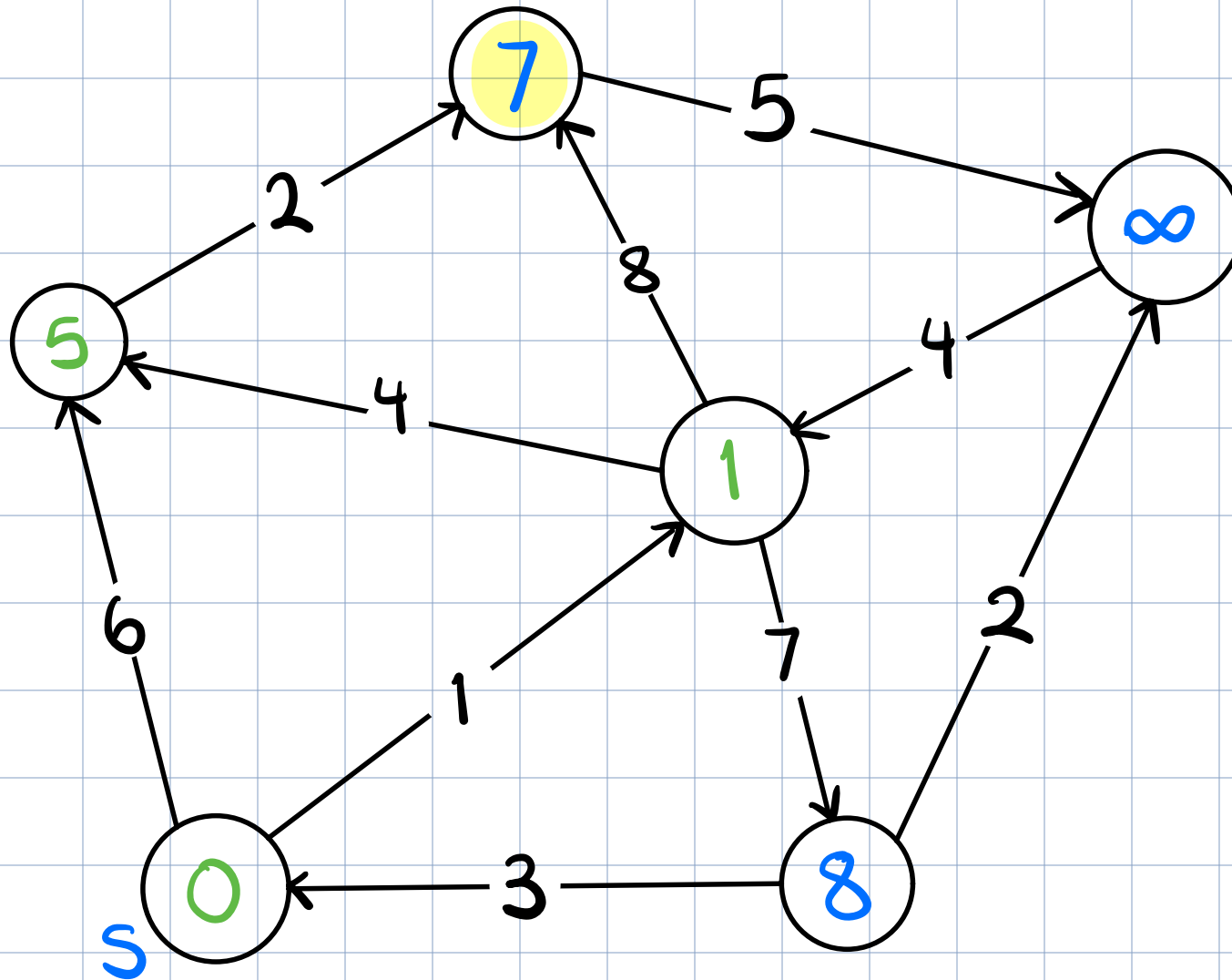
Dijkstra's



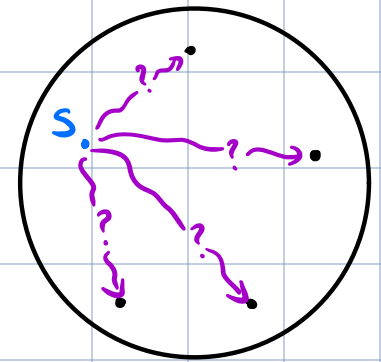
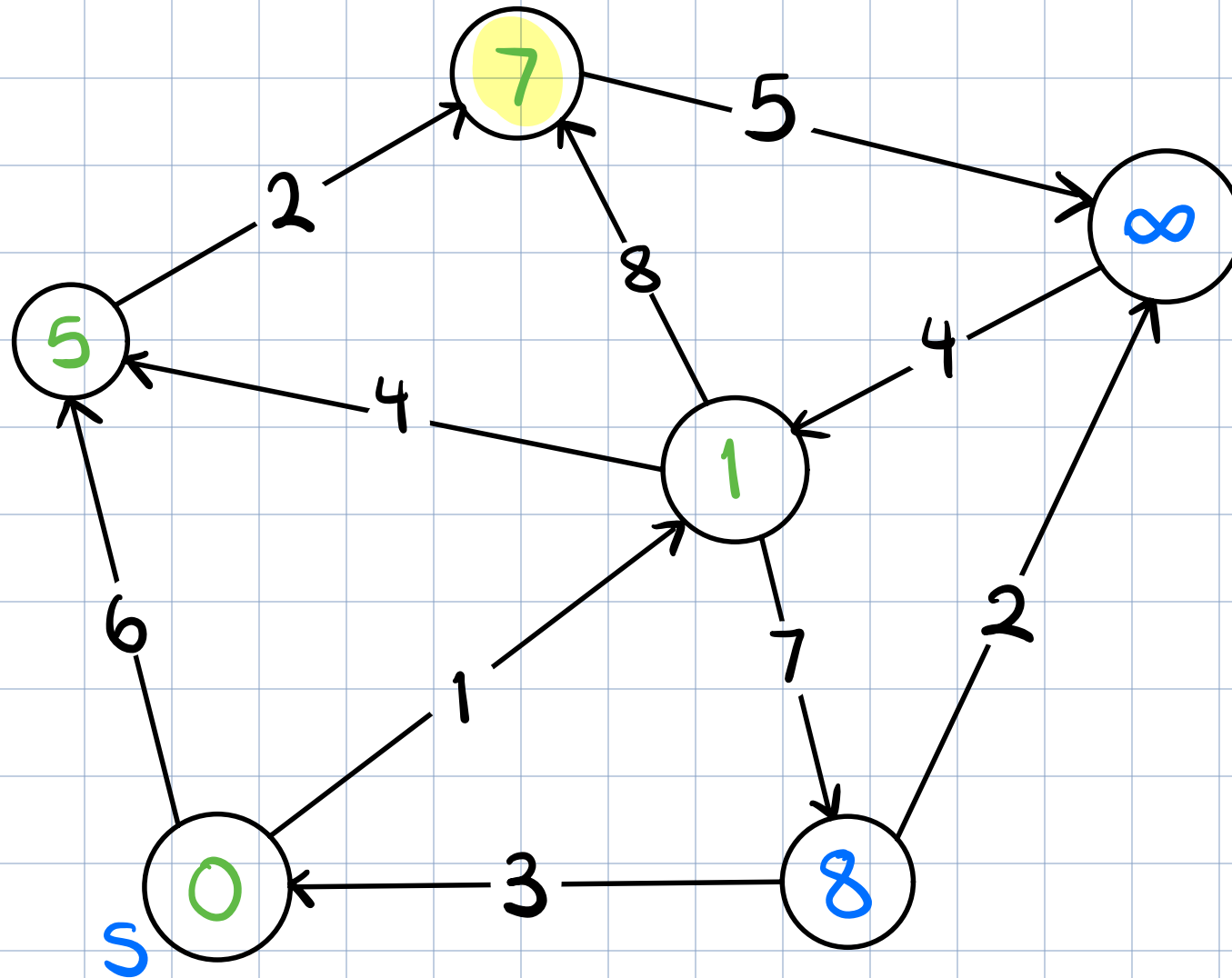
Dijkstra's



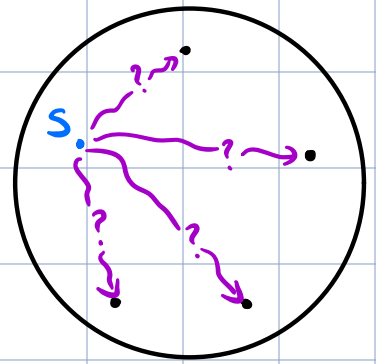
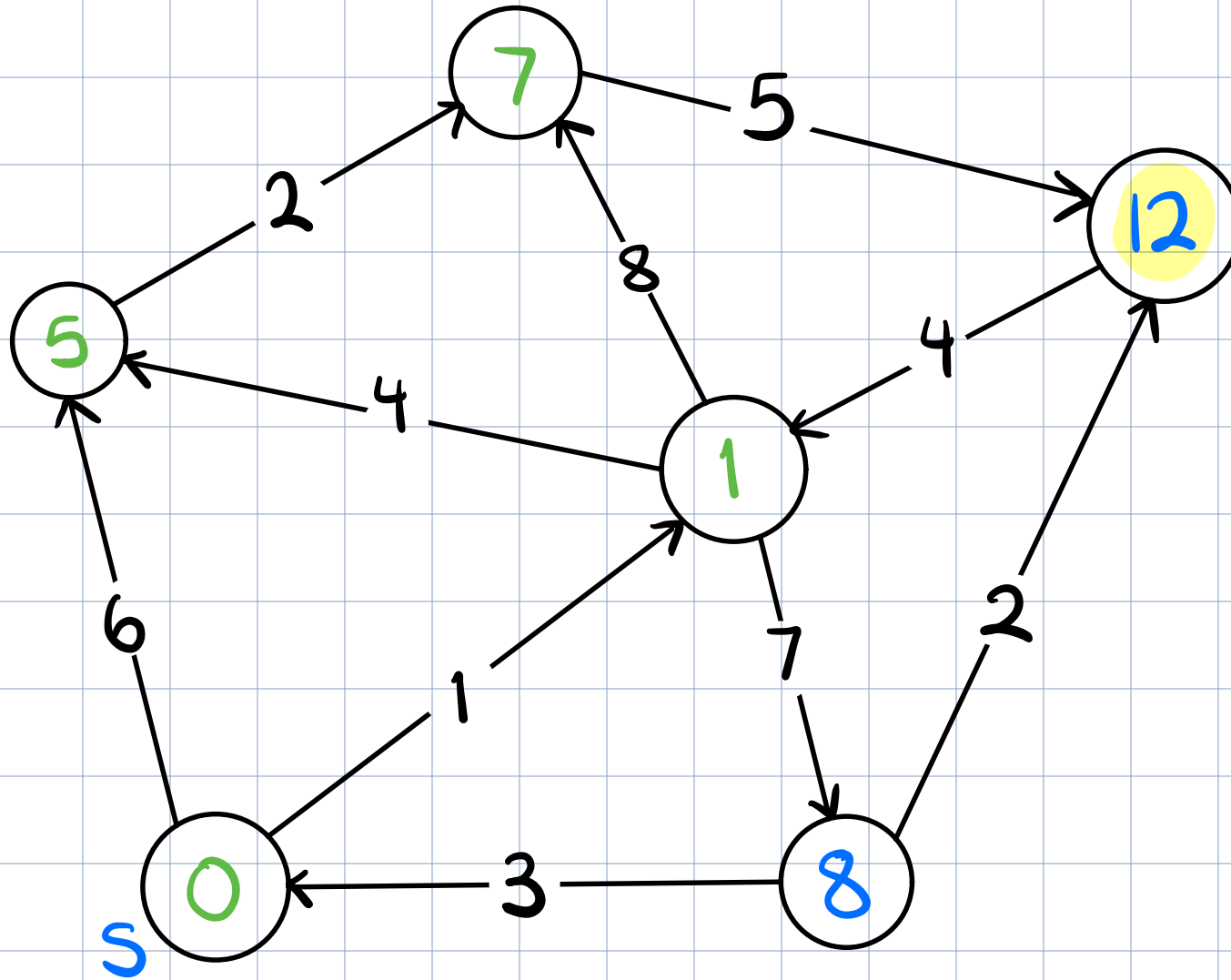
Dijkstra's



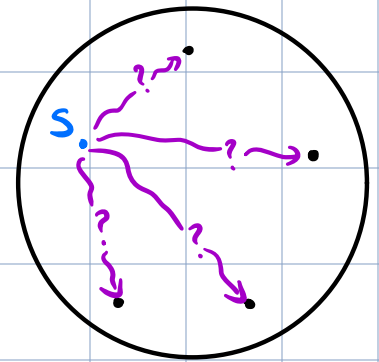
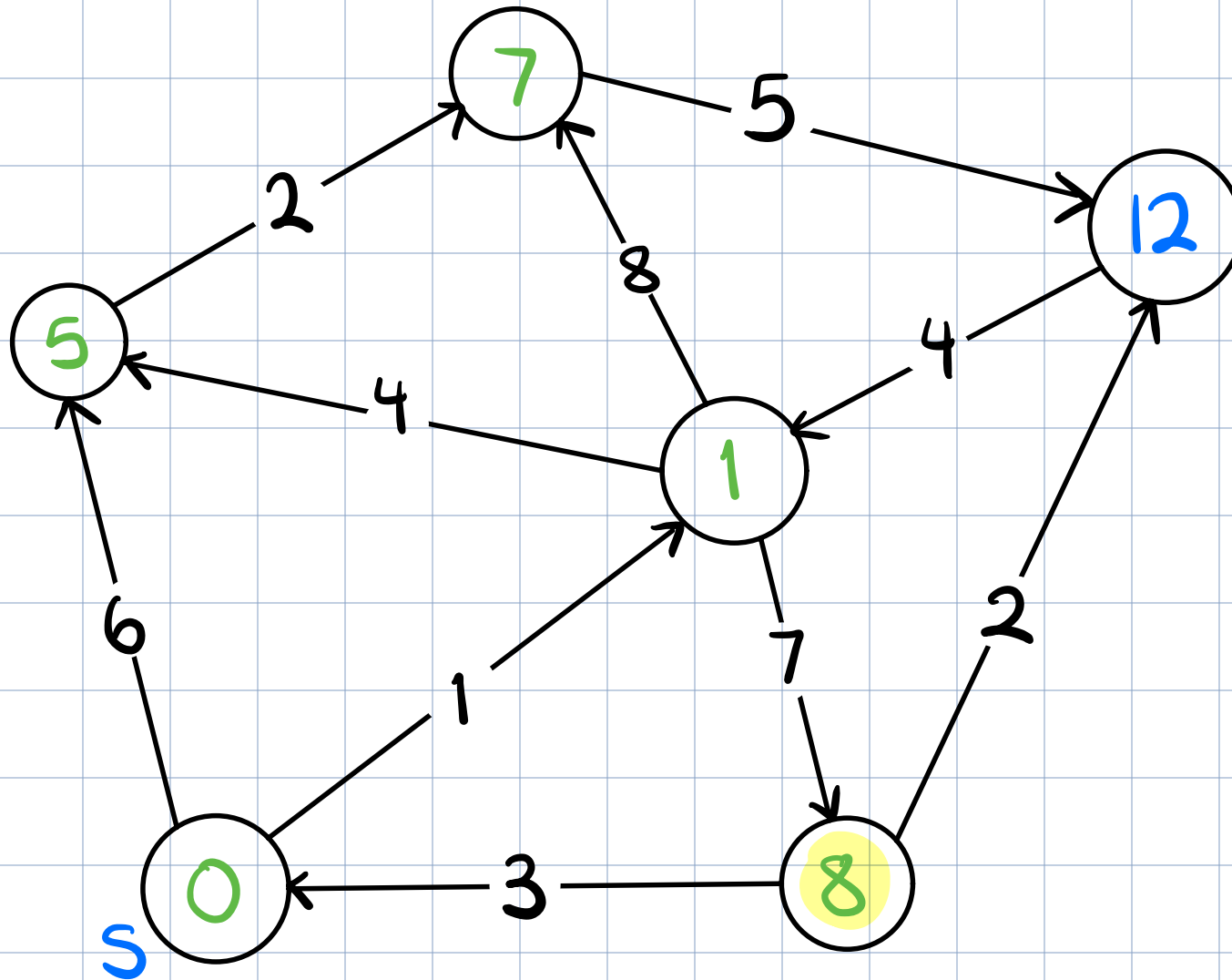
Dijkstra's



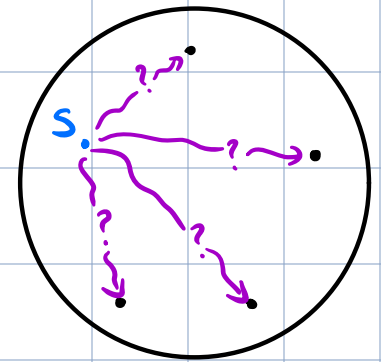
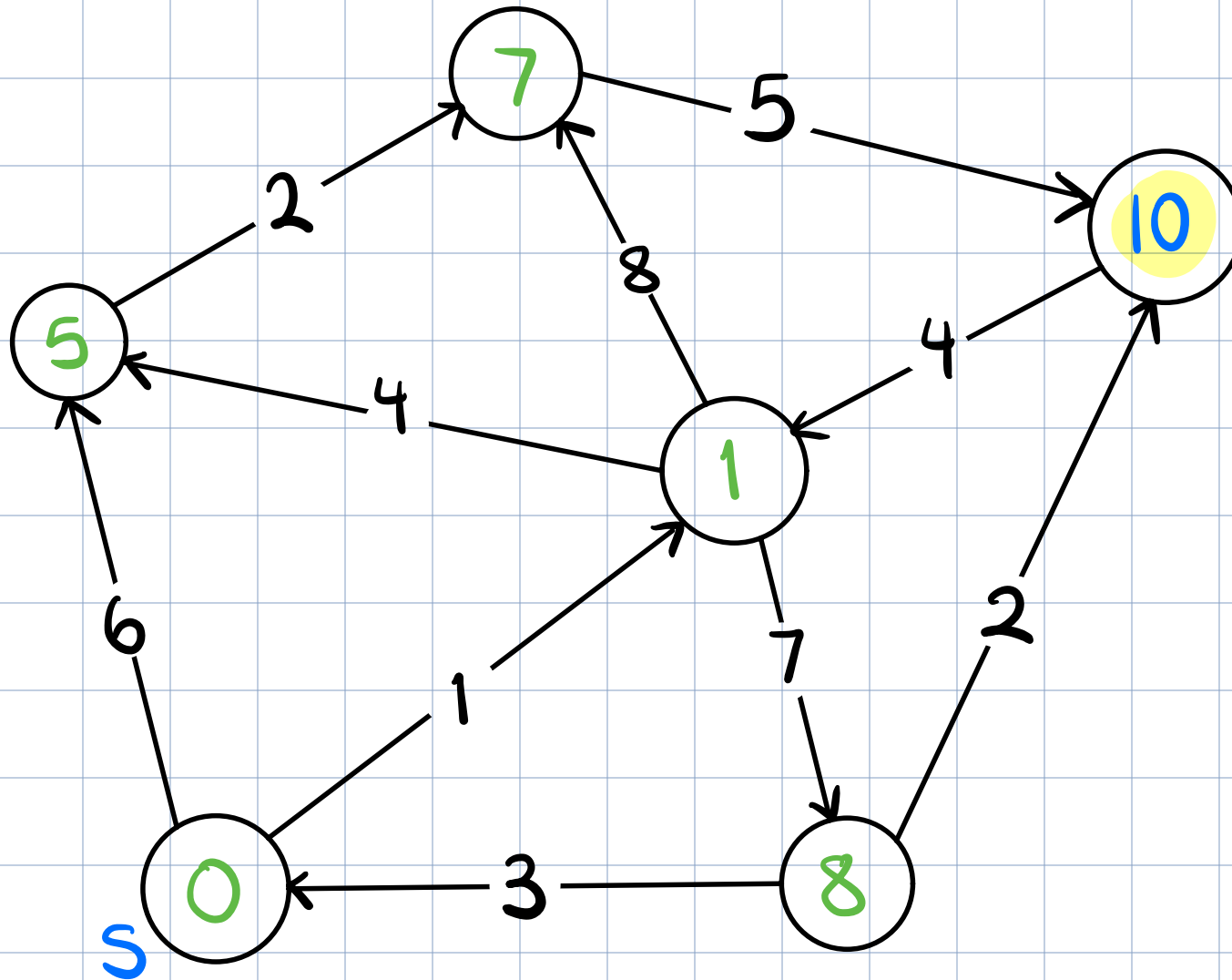
Dijkstra's



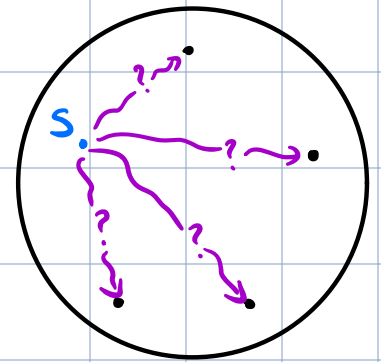
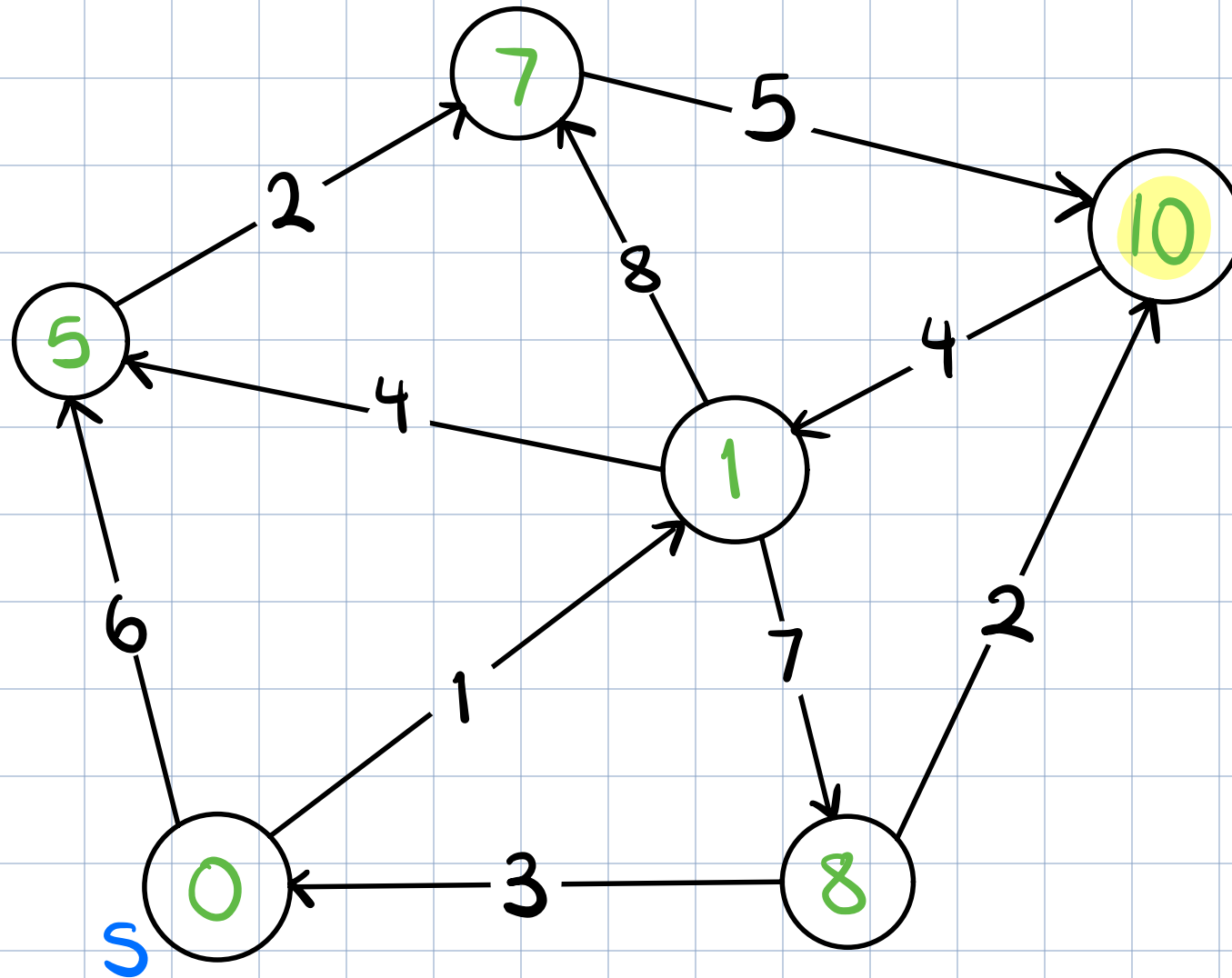
Dijkstra's



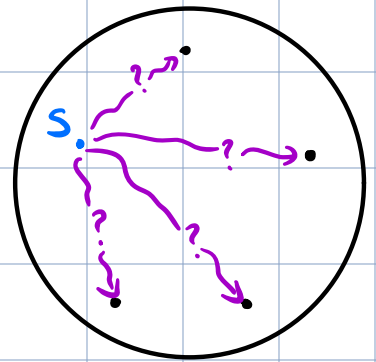
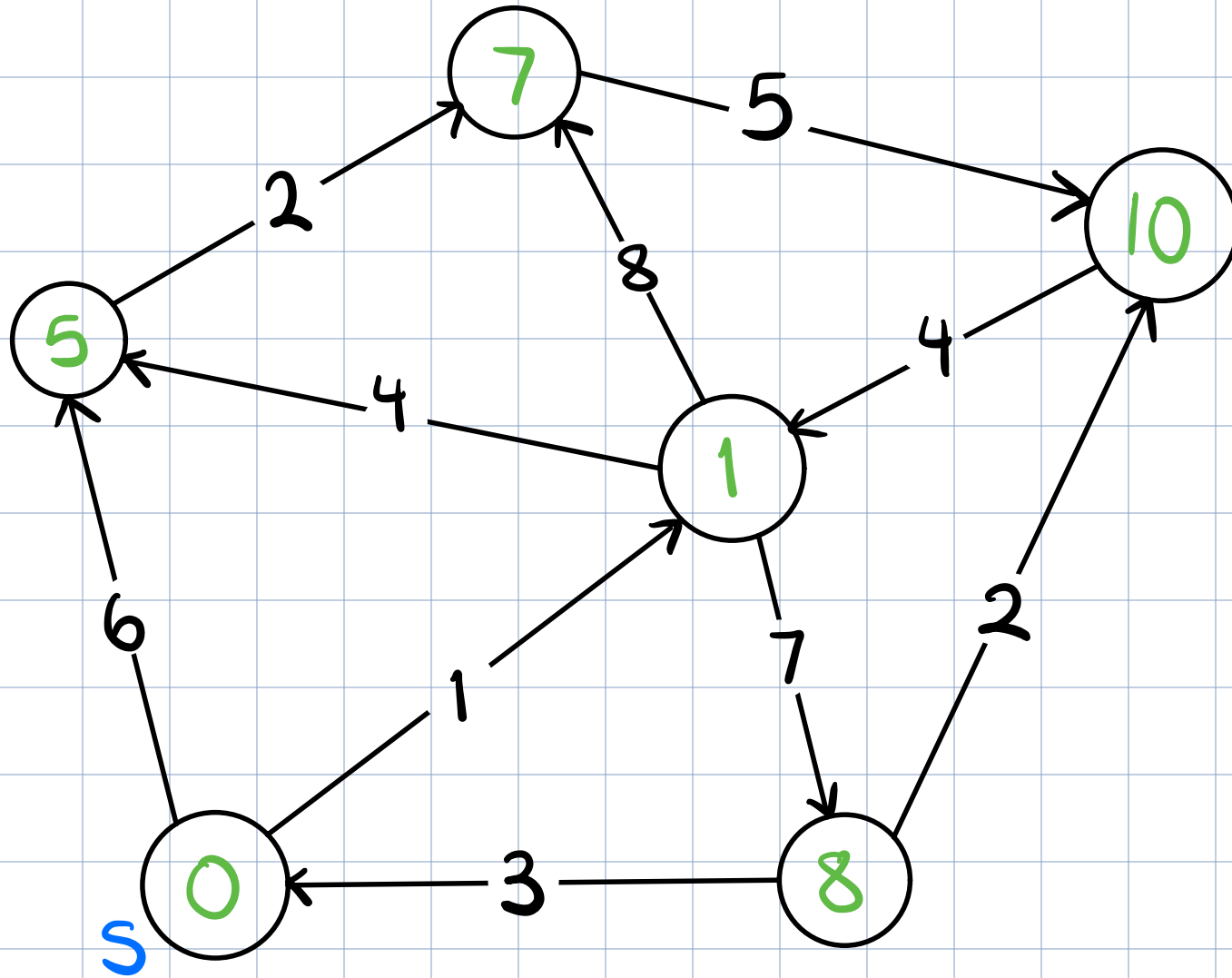
Dijkstra's



Dijkstra's



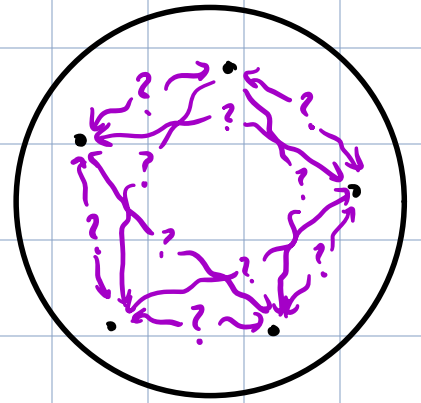
Dijkstra's



Total: $O(m + n \log n)$

Floyd-Warshall: Magic DP

$$\text{dist}(s, t, k) := \min \{ l(W) \mid W \text{ a } (s, t) \text{ walk using only vertices in } \{s, t, v_1, \dots, v_k\} \}$$

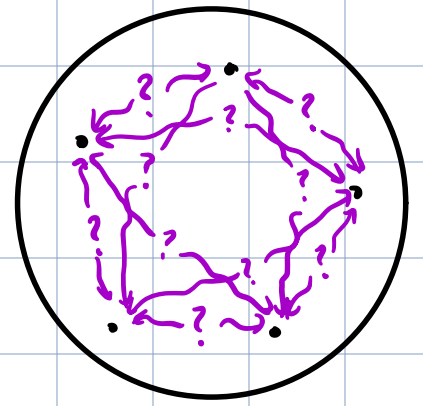


$$\text{dist}(s, t, k) = \begin{cases} 0 & \text{if } s = t \\ l(s, t) & \text{if } k = 0 \text{ \& } (s, t) \in E \\ \infty & \text{if } k = 0 \text{ \& } (s, t) \notin E \\ \min \{ d(s, t, k-1), d(s, v_k, k-1) + d(v_k, t, k-1) \} & \text{otherwise} \end{cases}$$

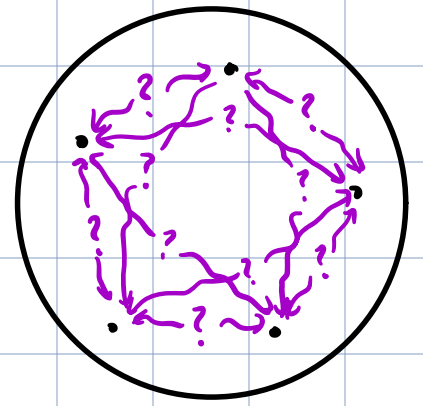
(either v_k included or not)

$$\text{Time} = n^3 \text{ subproblems} \cdot O(1) = O(n^3)$$

Question: Can we solve APSP faster than $O(n^3)$?

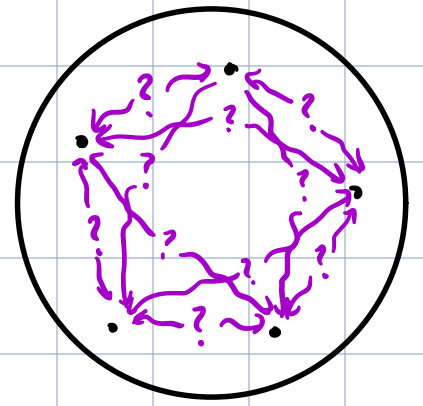


Question: Can we solve APSP faster than $O(n^3)$?



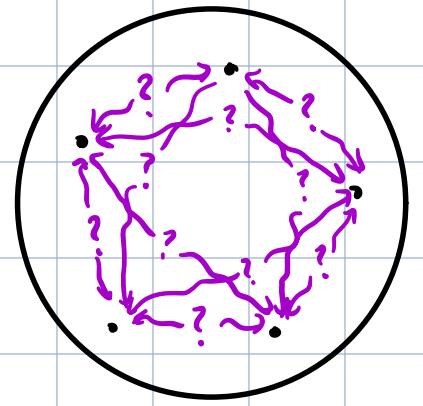
If weights are ≥ 0

Question: Can we solve APSP faster than $O(n^3)$?

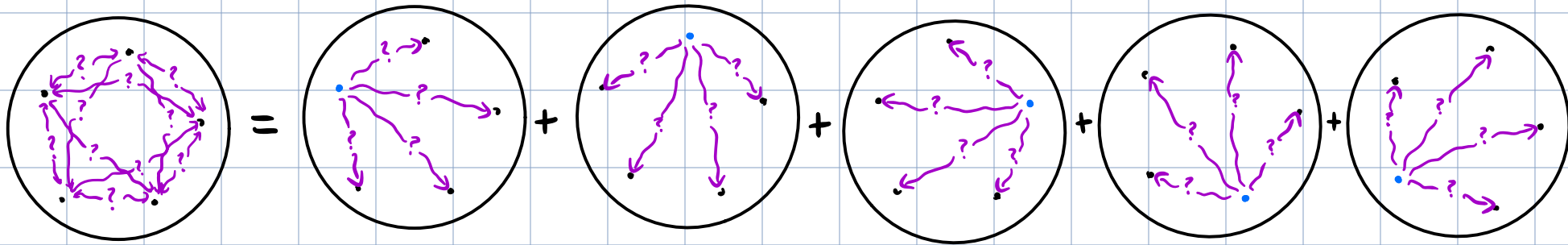


If weights are ≥ 0 , yes: Dijkstra from each $v \in V$!

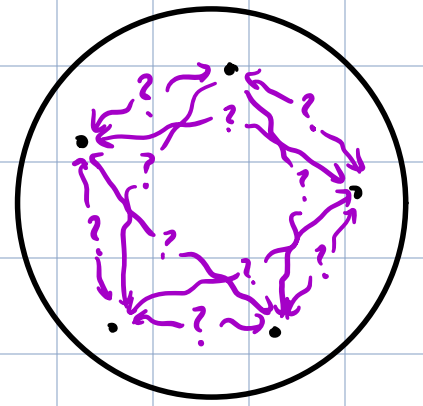
Question: Can we solve APSP faster than $O(n^3)$?



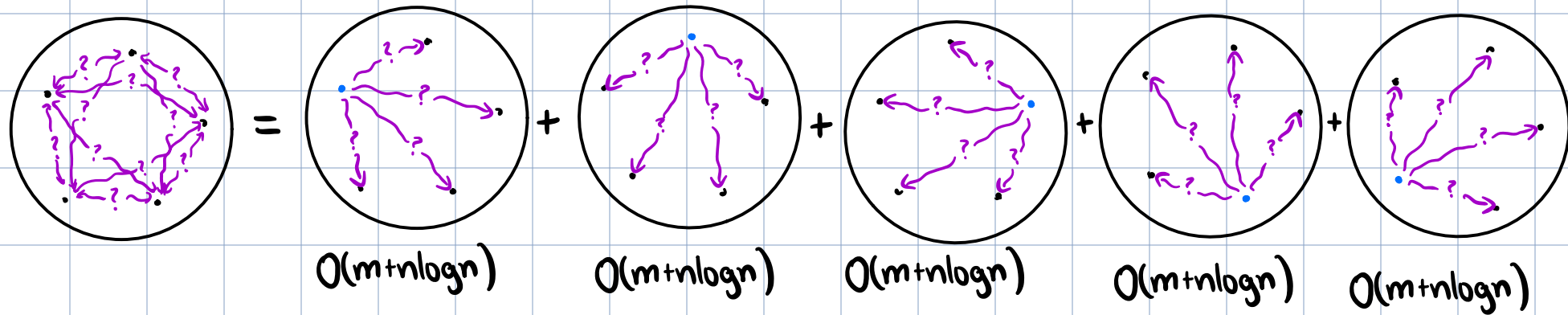
If weights are ≥ 0 , yes: Dijkstra from each $v \in V$!



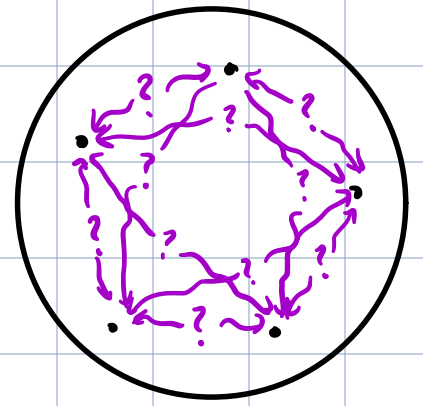
Question: Can we solve APSP faster than $O(n^3)$?



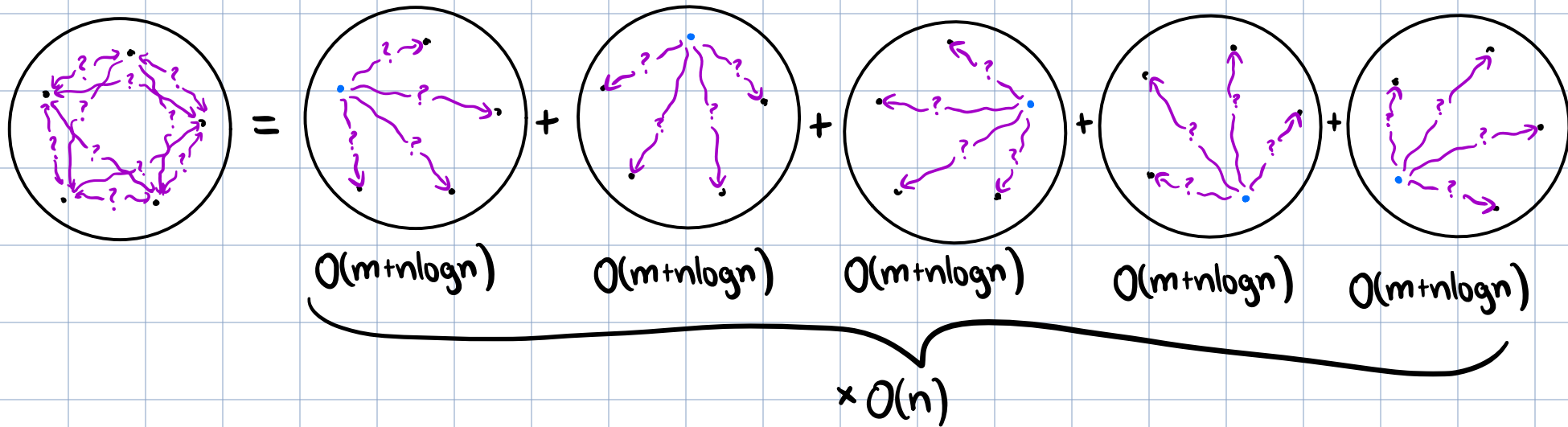
If weights are ≥ 0 , yes: Dijkstra from each $v \in V$!



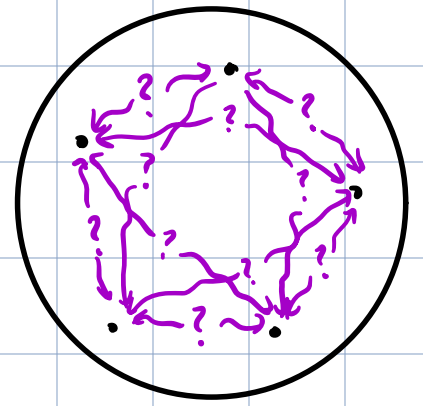
Question: Can we solve APSP faster than $O(n^3)$?



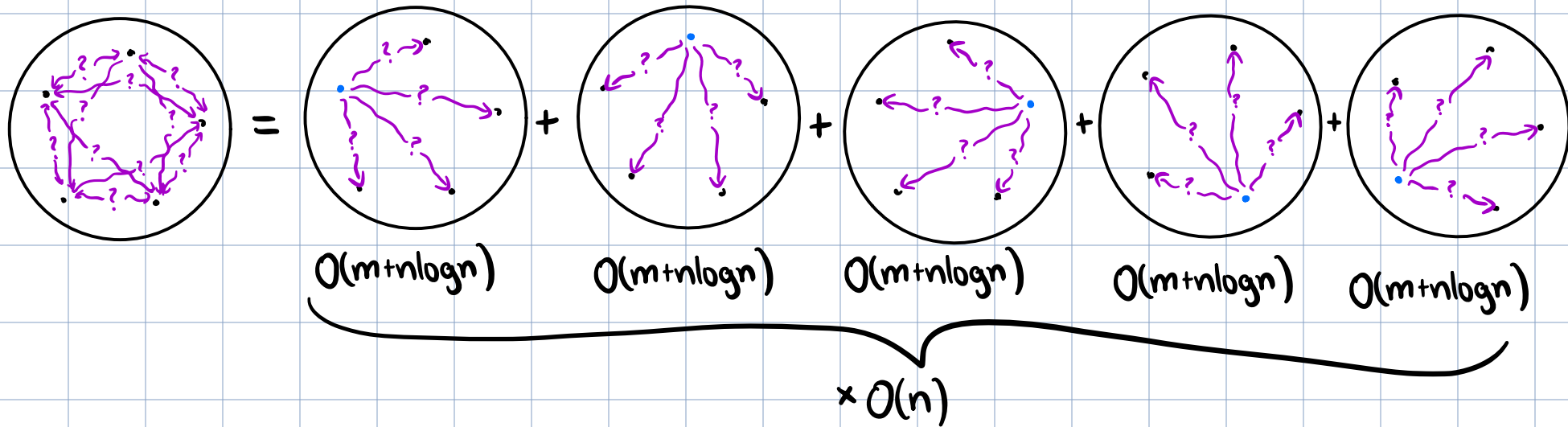
If weights are ≥ 0 , yes: Dijkstra from each $v \in V$!



Question: Can we solve APSP faster than $O(n^3)$?

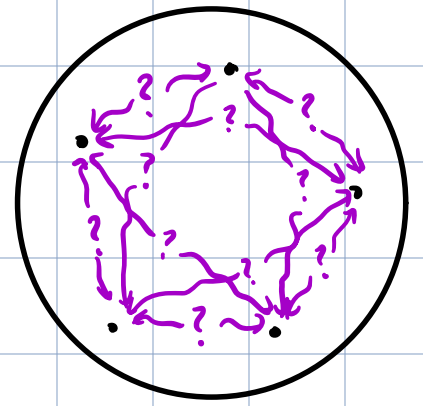


If weights are ≥ 0 , yes: Dijkstra from each $v \in V$!

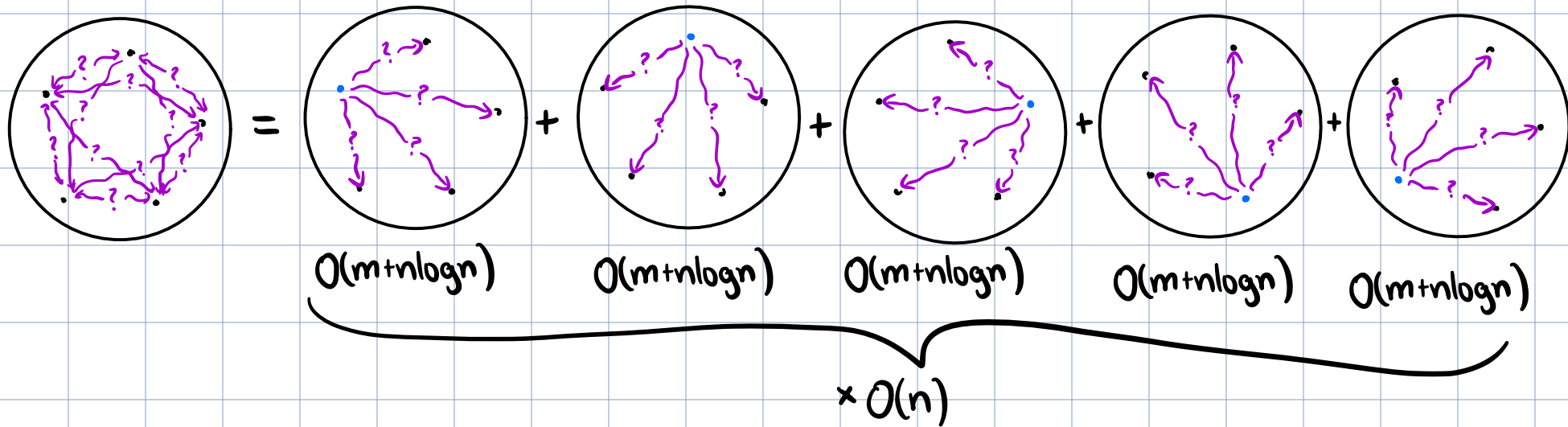


Total: $O(mn + n^2 \log n) \ll O(n^3)$ if G is sparse

Question: Can we solve APSP faster than $O(n^3)$?

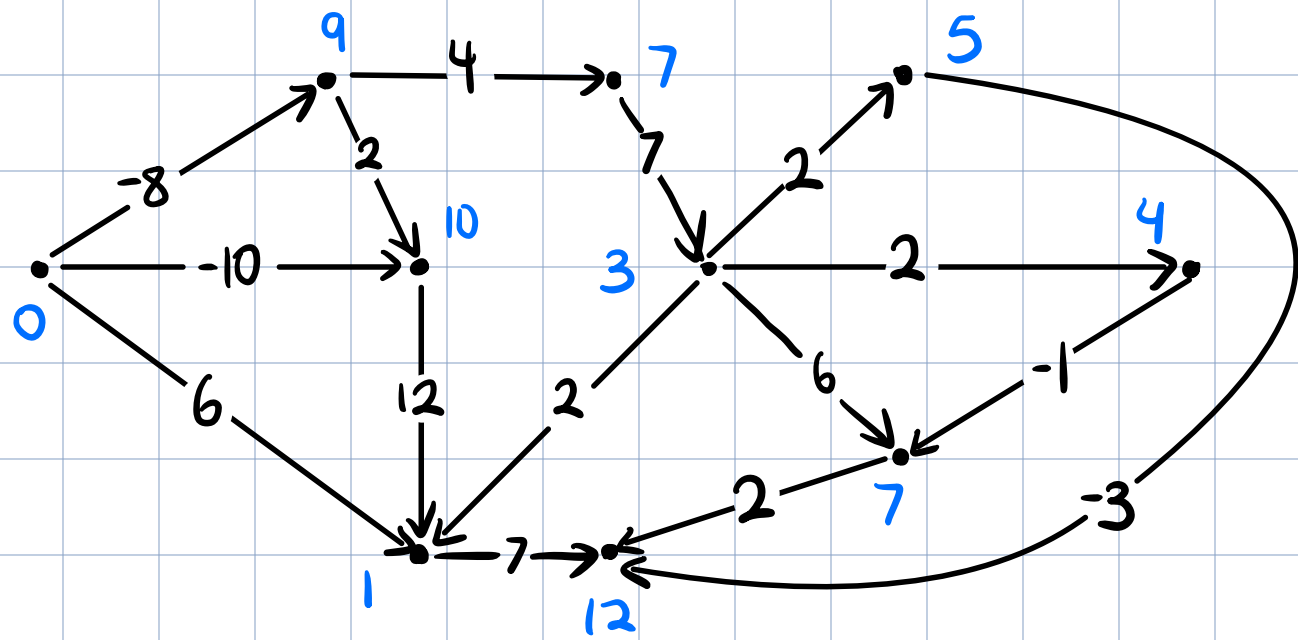
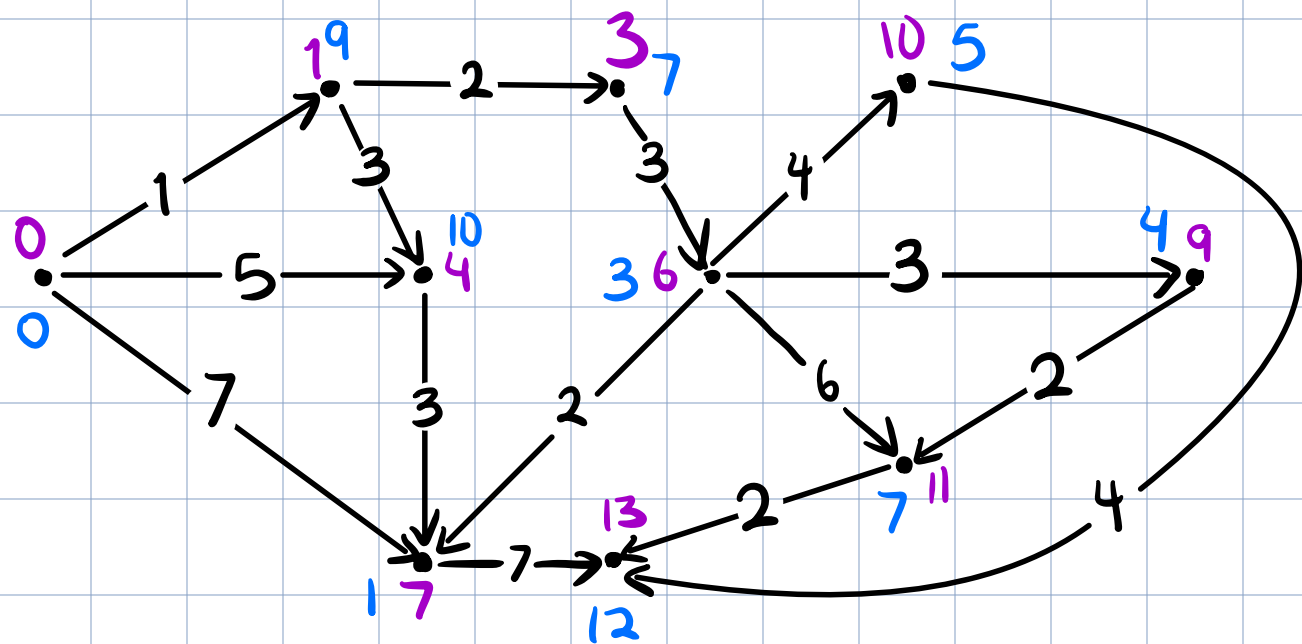


If weights are ≥ 0 , yes: Dijkstra from each $v \in V$!

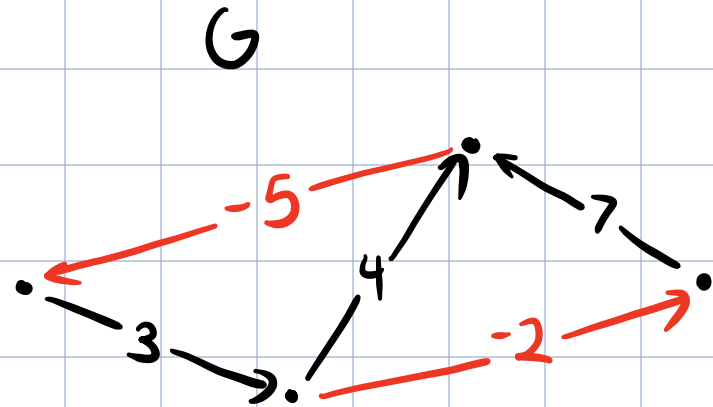


Total: $O(mn + n^2 \log n) \ll O(n^3)$ if G is sparse

Can we make this work in general?



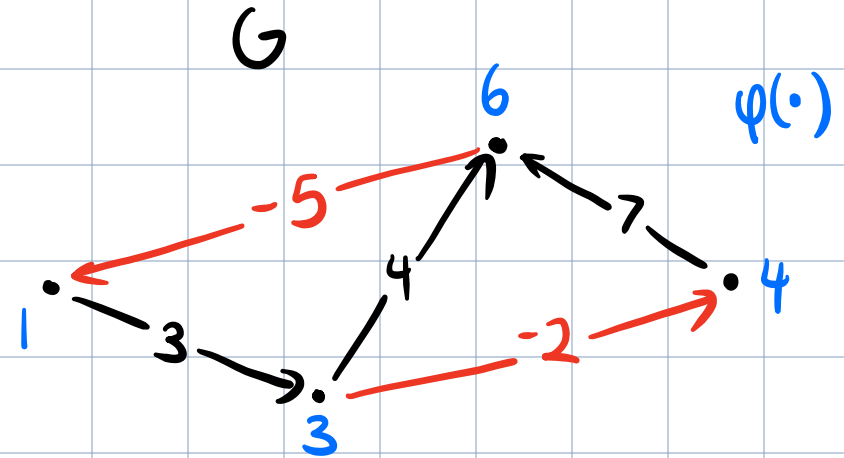
Potentials



Potentials

Say we have vertex weights

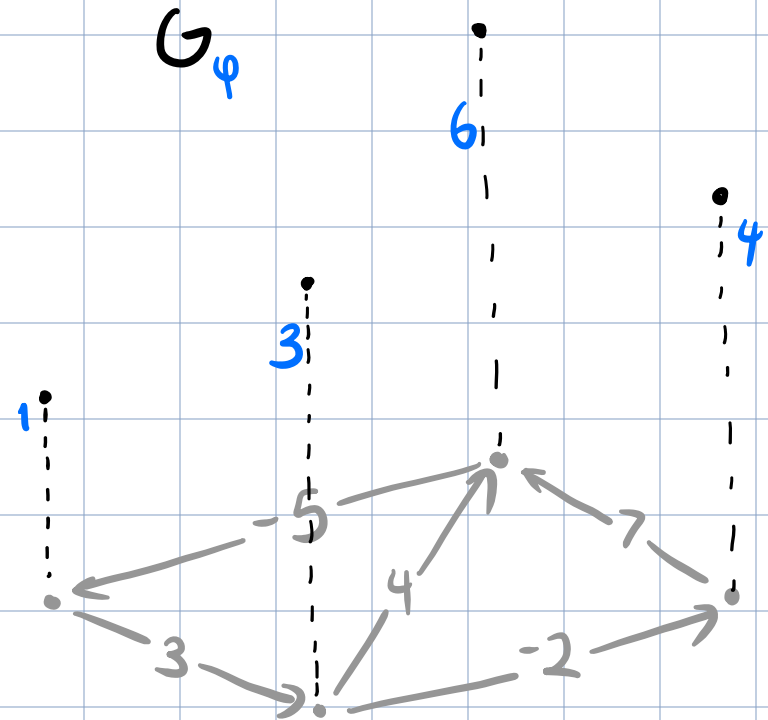
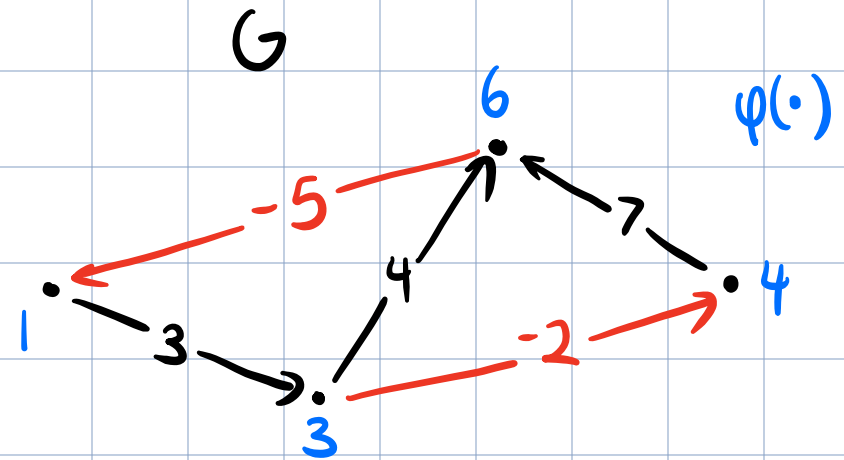
$$\varphi: V \rightarrow \mathbb{R}$$



Potentials

Say we have vertex weights

$$\varphi: V \rightarrow \mathbb{R}$$



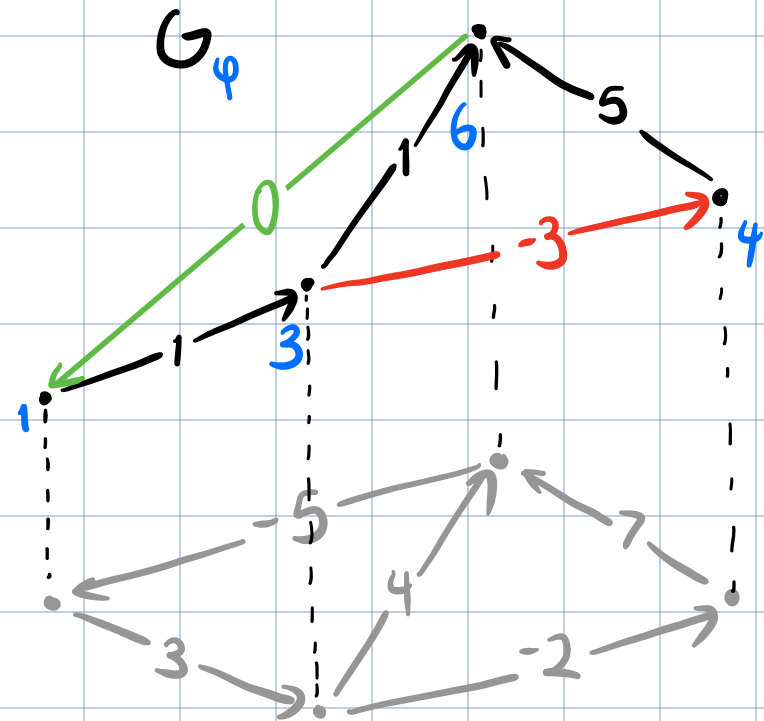
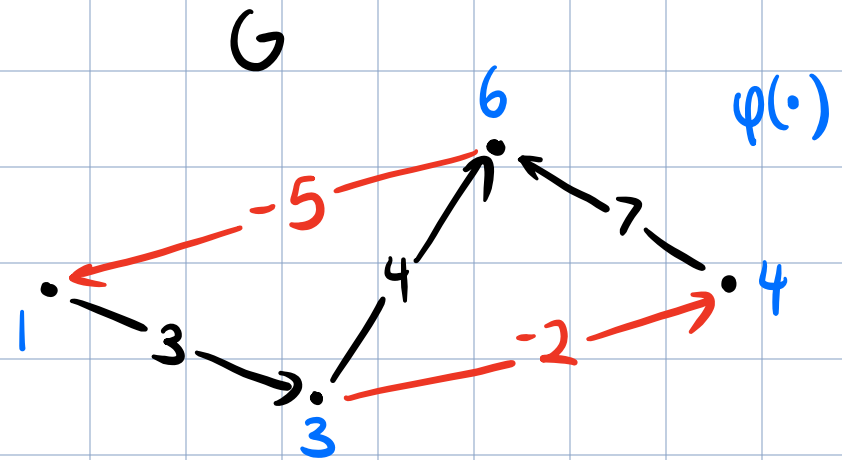
Potentials

Say we have vertex weights
 $\varphi: V \rightarrow \mathbb{R}$

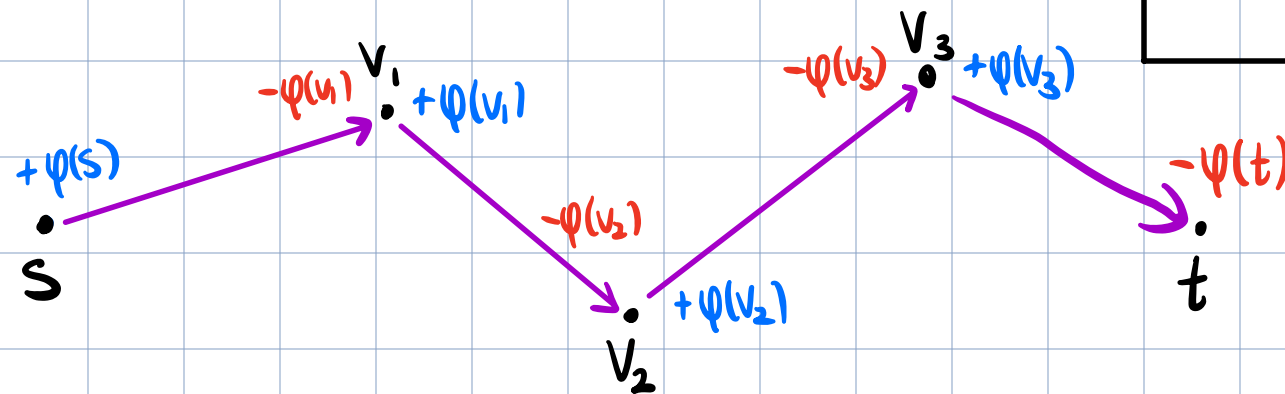
Define reduced weights l_φ by

$$l_\varphi(u, v) = l(u, v) + \varphi(u) - \varphi(v)$$

"Subtract uphill change/
add downhill change"



(s,t) -walks W have length



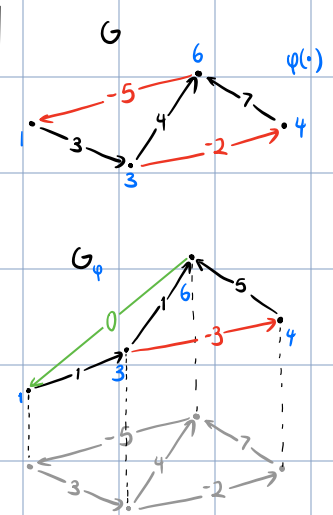
Potentials

Say we have vertex weights
 $\varphi: V \rightarrow \mathbb{R}$

Define reduced weights l_φ by

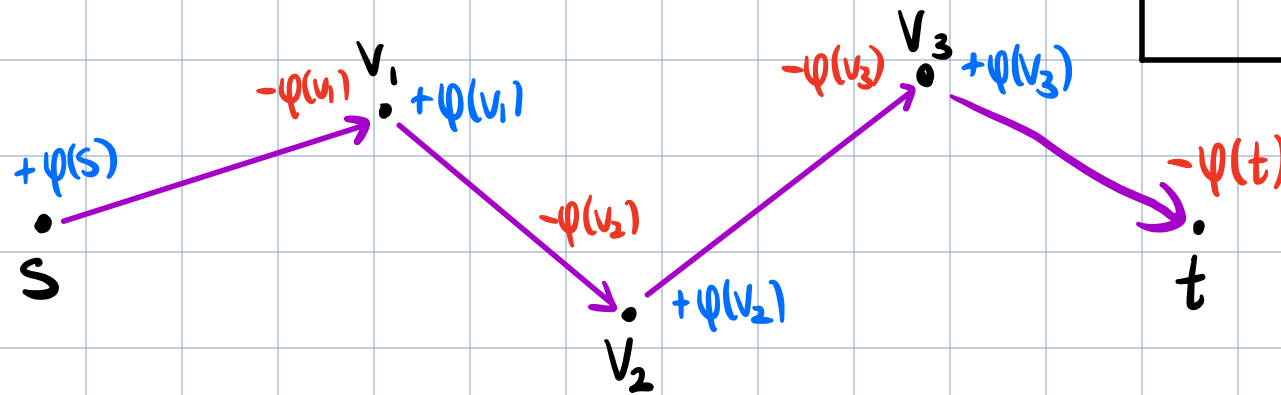
$$l_\varphi(u,v) = l(u,v) + \varphi(u) - \varphi(v)$$

"Subtract uphill change/
 add downhill change"



(s,t) -walks W have length

$$l_\varphi(W) = l(W) + \varphi(s) - \varphi(t)$$



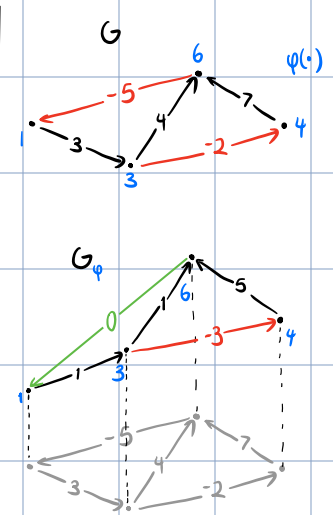
Potentials

Say we have vertex weights
 $\varphi: V \rightarrow \mathbb{R}$

Define reduced weights l_φ by

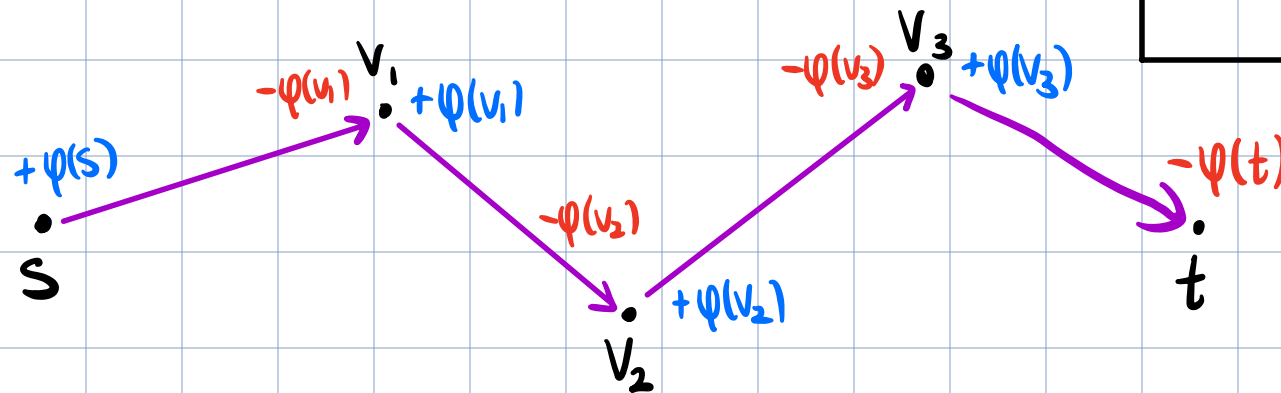
$$l_\varphi(u,v) = l(u,v) + \varphi(u) - \varphi(v)$$

"Subtract uphill change/
 add downhill change"

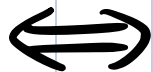


(s,t) -walks W have length

$$l_\varphi(W) = l(W) + \varphi(s) - \varphi(t)$$



W a shortest (s,t) -walk in G



W a shortest (s,t) -walk in G_φ !

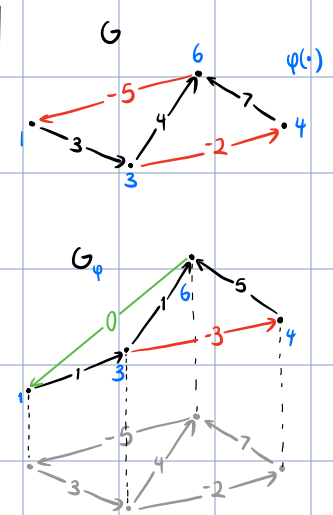
Potentials

Say we have vertex weights
 $\varphi: V \rightarrow \mathbb{R}$

Define reduced weights l_φ by

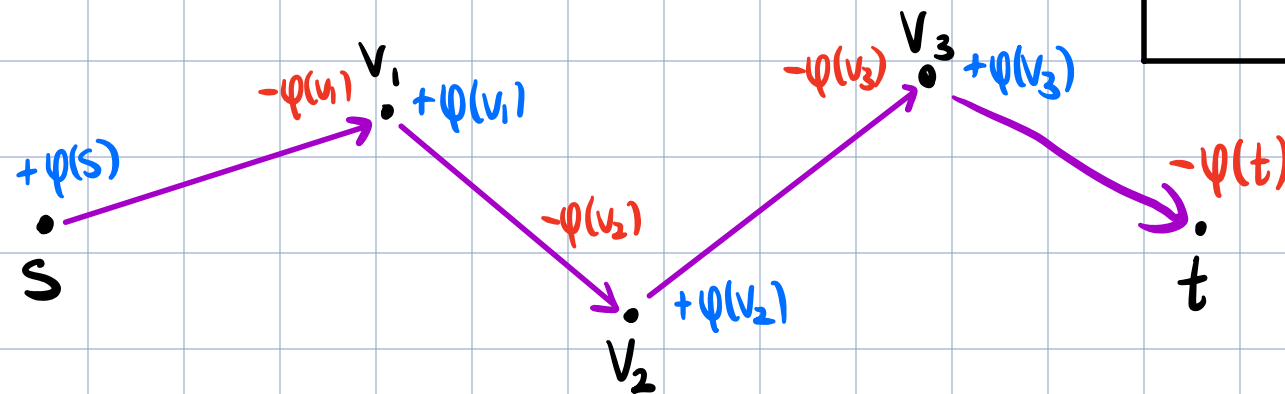
$$l_\varphi(u,v) = l(u,v) + \varphi(u) - \varphi(v)$$

"Subtract uphill change/
 add downhill change"

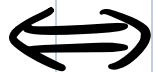


(s,t) -walks W have length

$$l_\varphi(W) = l(W) + \varphi(s) - \varphi(t)$$



W a shortest (s,t) -walk in G



W a shortest (s,t) -walk in G_φ !

$$\left. \begin{array}{l} W \text{ a shortest } (s,t)\text{-walk in } G \\ \iff \\ W \text{ a shortest } (s,t)\text{-walk in } G_\varphi! \end{array} \right\} d_{G_\varphi}(s,t) = d(s,t) + \varphi(s) - \varphi(t)$$

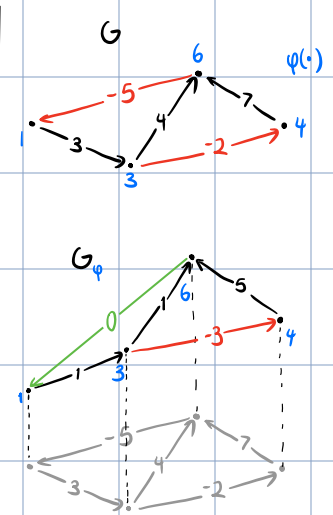
Potentials

Say we have vertex weights
 $\varphi: V \rightarrow \mathbb{R}$

Define reduced weights l_φ by

$$l_\varphi(u,v) = l(u,v) + \varphi(u) - \varphi(v)$$

"Subtract uphill change/
 add downhill change"



(s,t) -walks W have length

$$l_\varphi(W) = l(W) + \varphi(s) - \varphi(t)$$

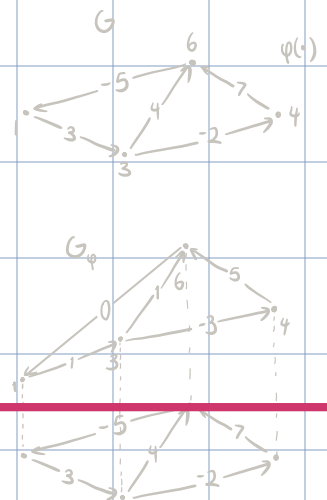
Potentials

Say we have vertex weights
 $\varphi: V \rightarrow \mathbb{R}$

Define reduced weights l_φ by

$$l_\varphi(u,v) = l(u,v) + \varphi(u) - \varphi(v)$$

"Subtract uphill change/
 add downhill change"



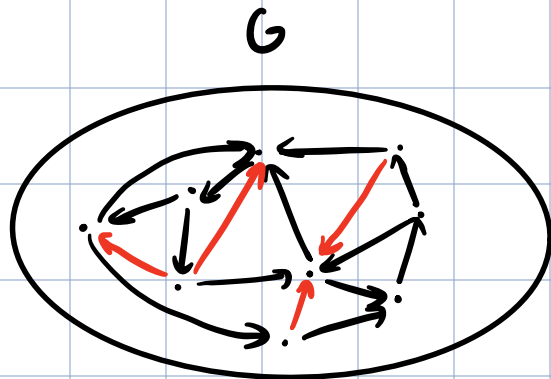
**Distances in G_φ give
 distances in G**

W a shortest (s,t) -walk in G

$$d_{G_\varphi}(s,t) = d(s,t) + \varphi(s) - \varphi(t)$$

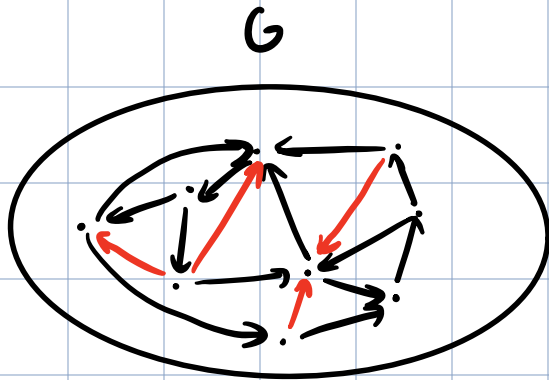
$\iff$
 W a shortest (s,t) -walk in G_φ !

Neutralizing G



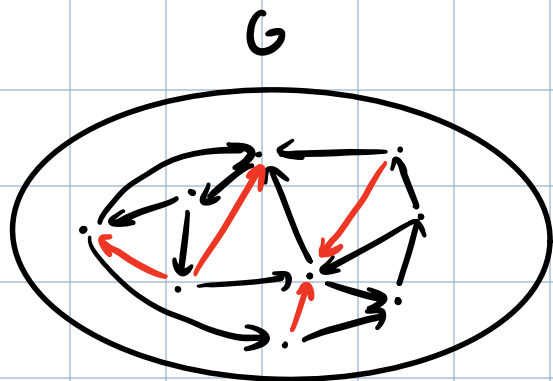
Neutralizing G

Suppose some vertex s can reach all $v \in V$.



Neutralizing G

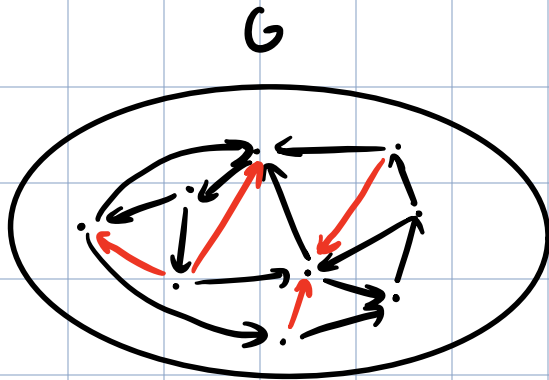
Suppose some vertex s can reach all $v \in V$. Let $\varphi(v) := d(s, v)$. Then,



Neutralizing G

Suppose some vertex s can reach all $v \in V$. Let $\varphi(v) := d(s, v)$. Then,

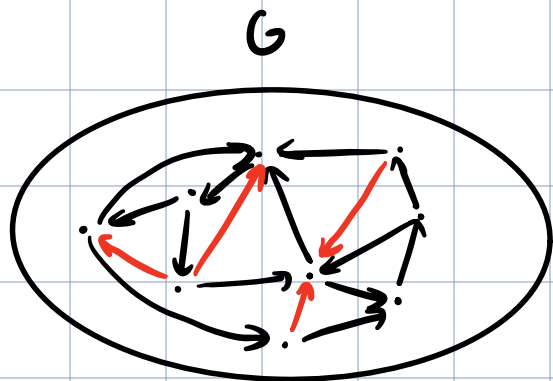
$$l_{\varphi}(u, v) =$$



Neutralizing G

Suppose some vertex s can reach all $v \in V$. Let $\varphi(v) := d(s, v)$. Then,

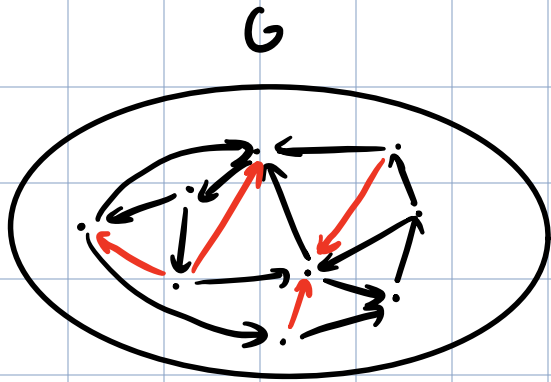
$$l_{\varphi}(u, v) = l(u, v) + \varphi(u) - \varphi(v)$$



Neutralizing G

Suppose some vertex s can reach all $v \in V$. Let $\varphi(v) := d(s, v)$. Then,

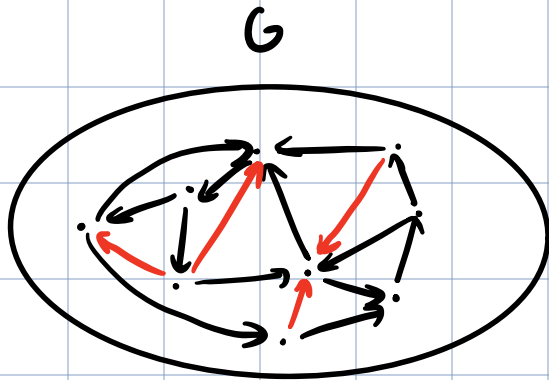
$$l_{\varphi}(u, v) = l(u, v) + \varphi(u) - \varphi(v) = l(u, v) + d(s, u) - d(s, v)$$



Neutralizing G

Suppose some vertex s can reach all $v \in V$. Let $\varphi(v) := d(s, v)$. Then,

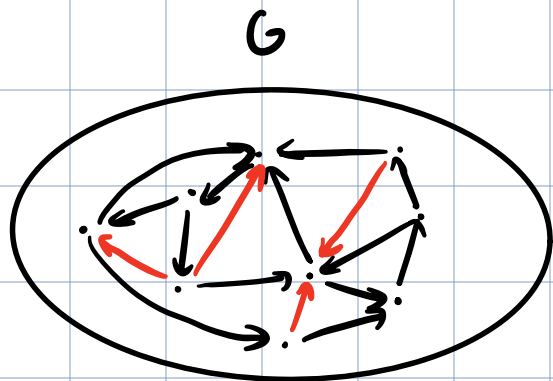
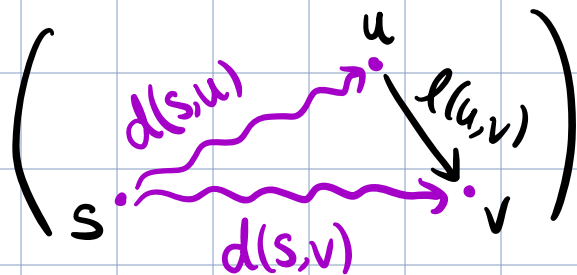
$$l_{\varphi}(u, v) = l(u, v) + \varphi(u) - \varphi(v) = l(u, v) + d(s, u) - d(s, v)$$



Neutralizing G

Suppose some vertex s can reach all $v \in V$. Let $\varphi(v) := d(s, v)$. Then,

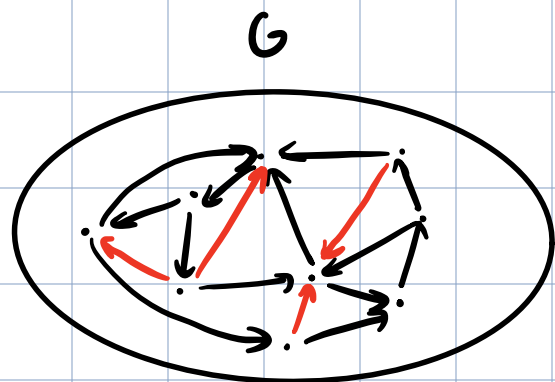
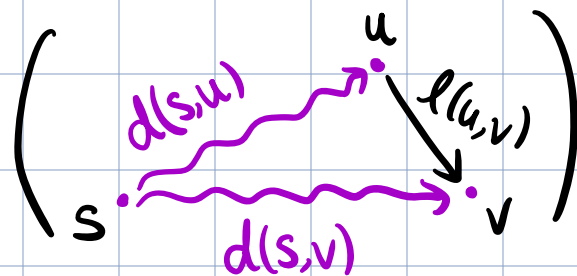
$$l_{\varphi}(u, v) = l(u, v) + \varphi(u) - \varphi(v) = l(u, v) + d(s, u) - d(s, v) \geq 0$$



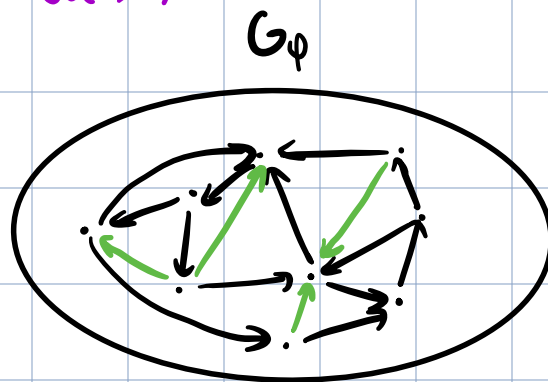
Neutralizing G

Suppose some vertex s can reach all $v \in V$. Let $\varphi(v) := d(s, v)$. Then,

$$l_{\varphi}(u, v) = l(u, v) + \varphi(u) - \varphi(v) = l(u, v) + d(s, u) - d(s, v) \geq 0$$



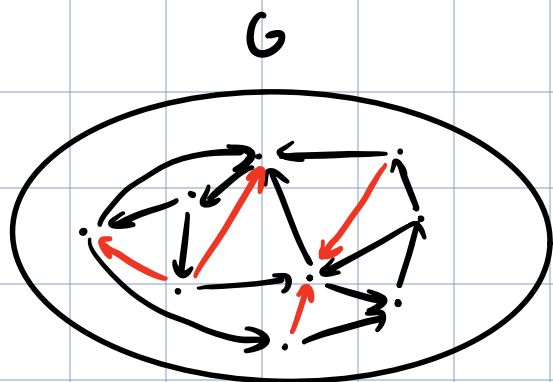
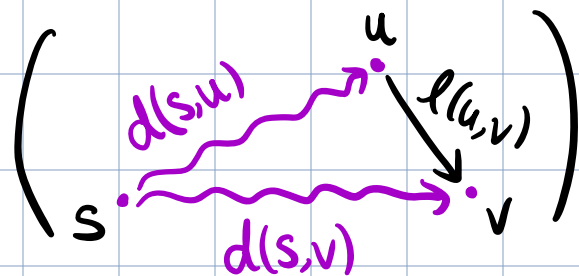
$$\varphi = d(s, \cdot)$$



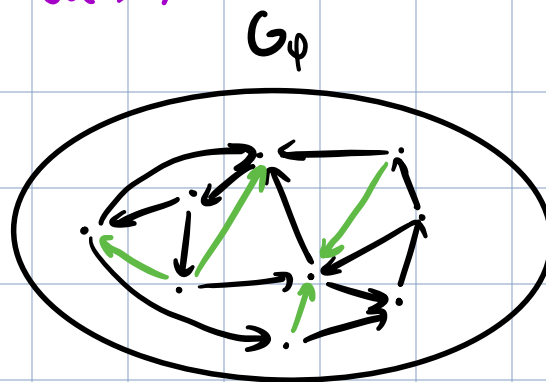
Neutralizing G

Suppose some vertex s can reach all $v \in V$. Let $\varphi(v) := d(s, v)$. Then,

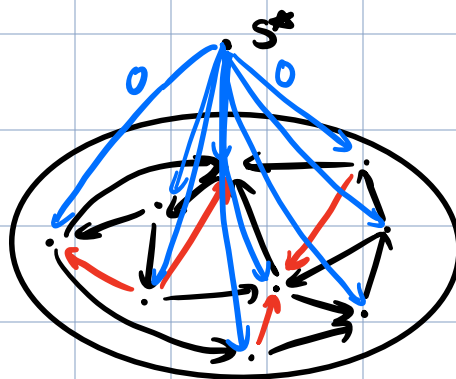
$$l_\varphi(u, v) = l(u, v) + \varphi(u) - \varphi(v) = l(u, v) + d(s, u) - d(s, v) \geq 0$$



$$\varphi = d(s, \cdot)$$



How to pick s ?



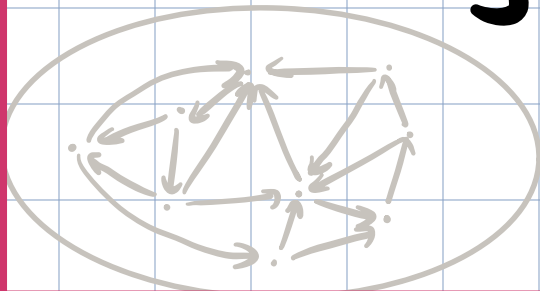
$$\begin{aligned} d(s^*, v) &= d(v, v) \\ &= \min_{u \in V} d(u, v) \end{aligned}$$

Neutralizing G

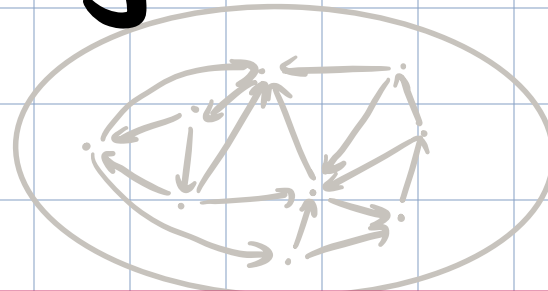
Suppose some vertex s can reach all $v \in V$. Let $\varphi(v) := d(s, v)$. Then,

$$l_\varphi(u, v) = l(u, v) + \varphi(u) - \varphi(v) = l(u, v) + d(s, u) - d(s, v) \geq 0$$

Distance potentials make
all edges nonnegative



$$\varphi = d(s, \cdot)$$

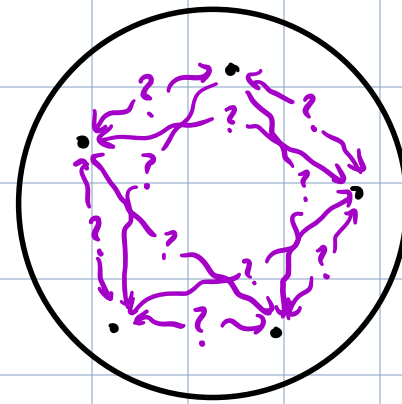
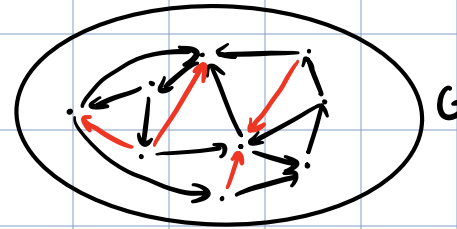


How to pick s ?



$$\begin{aligned} d(s^*, v) &= d(v, v) \\ &= \min_{u \in V} d(u, v) \end{aligned}$$

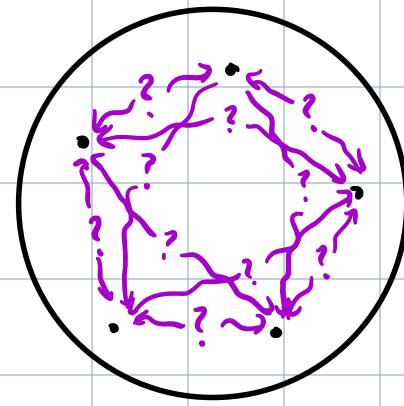
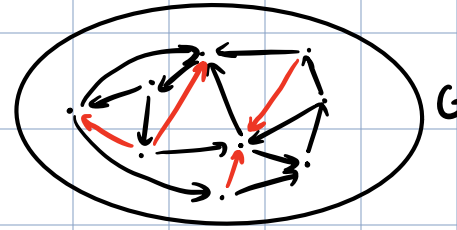
Johnson's Algo



Johnson's Algo

"neutralize G "

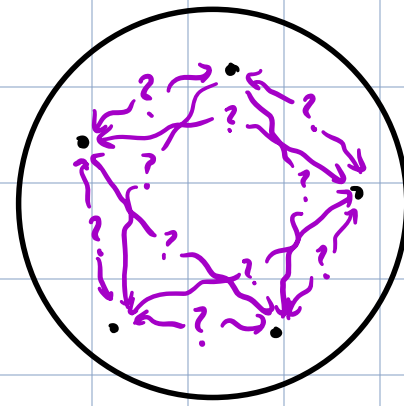
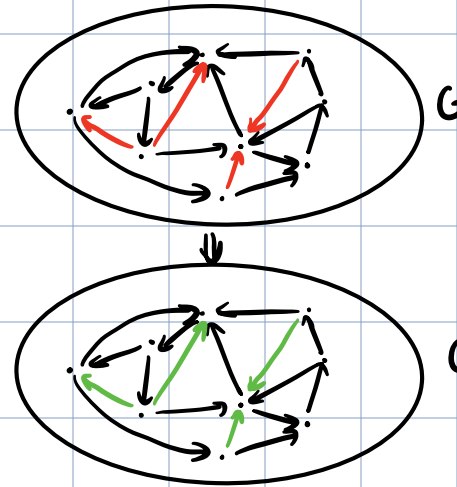
1. Compute $d(V, \cdot)$ for all v



Johnson's Algo

"neutralize G "

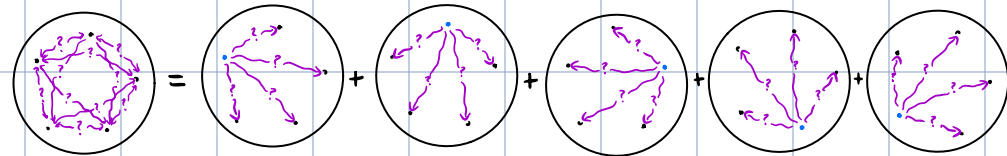
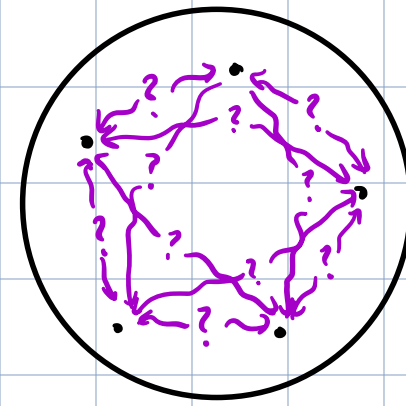
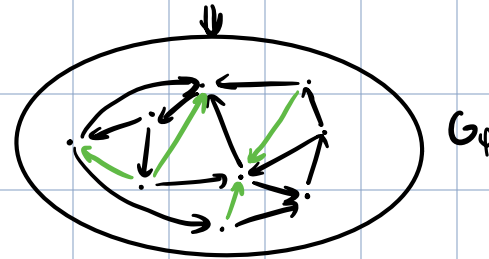
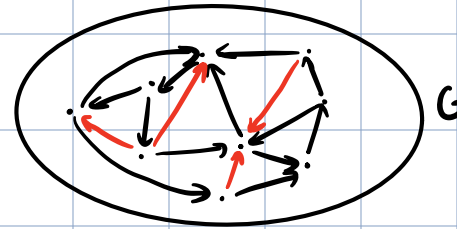
1. Compute $d(V, \cdot)$ for all v
2. Apply $\varphi(v) = d(V, v)$ to G



Johnson's Algo

"neutralize G "

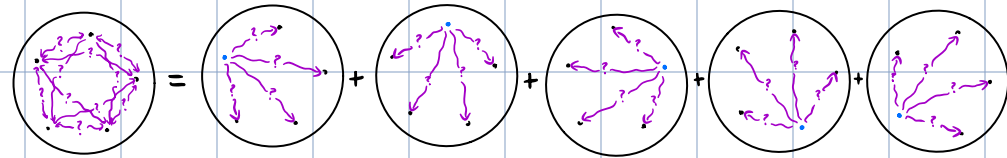
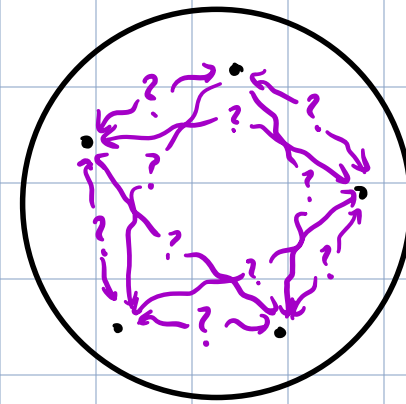
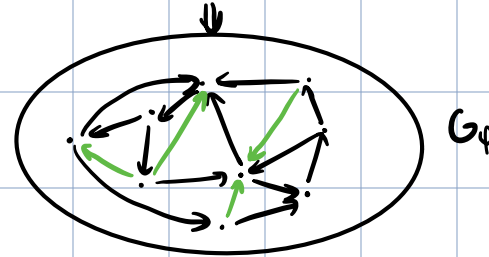
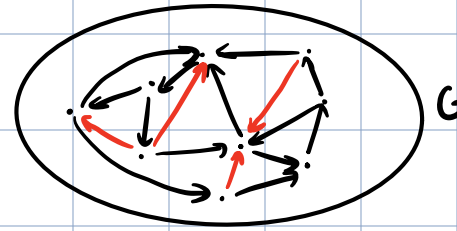
1. Compute $d(V, \cdot)$ for all v
2. Apply $\varphi(v) = d(V, v)$ to G
3. Dijkstra's from each v in G_φ



Johnson's Algo

"neutralize G "

1. Compute $d(V, \cdot)$ for all v
2. Apply $\varphi(v) = d(V, v)$ to G
3. Dijkstra's from each v in G_φ
4. Translate distances back.

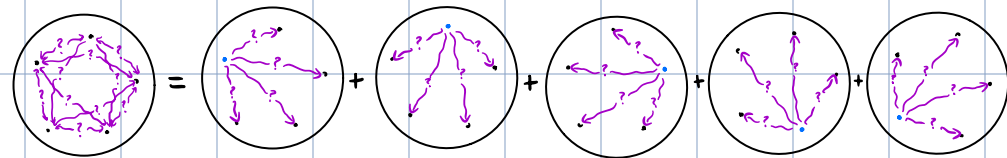
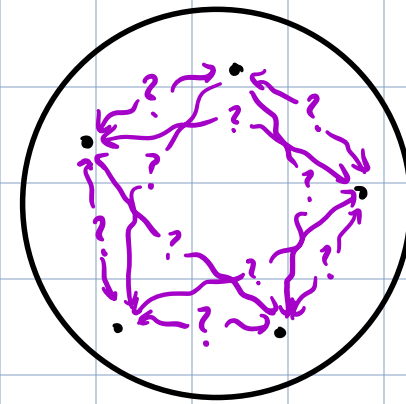
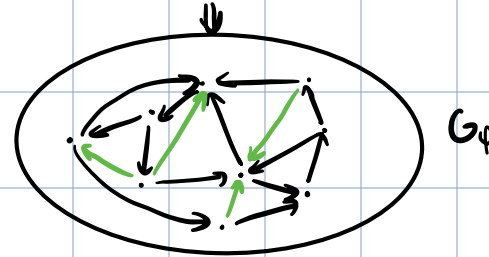
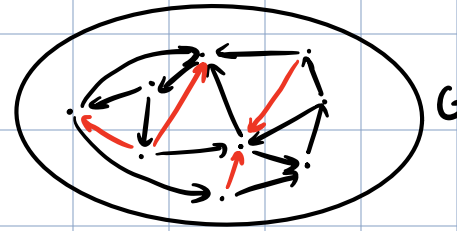


Total:

Johnson's Algo

"neutralize G "

1. Compute $d(V, \cdot)$ for all v
2. Apply $\varphi(v) = d(V, v)$ to G
3. Dijkstra's from each v in G_φ
4. Translate distances back.



$$\text{Total: } O(mn + n^2 \log n)$$

History



History

Bellman-Ford

|
1955



History

Bellman-Ford

1955



Dijkstra's

1984



History



Dijkstra's in $O(m+n \log n)$

Bellman-Ford

1955

1984



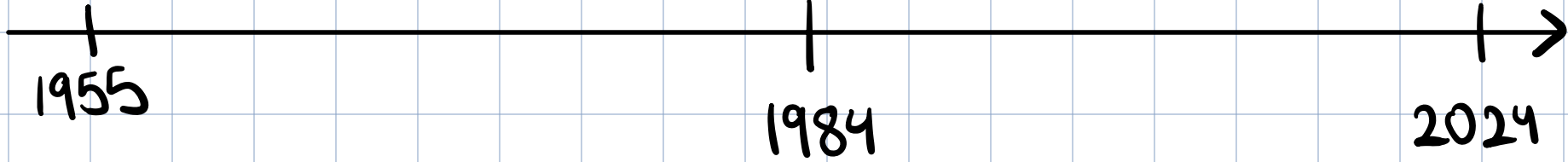
History



Dijkstra's in $O(m+n \log n)$

Bellman-Ford

Fineman's Algo



Housekeeping

Housekeeping

Who cares about logs?

Housekeeping

Who cares about logs? $O(n^2 \log^5 n)$

Housekeeping

"soft O"

Who cares about logs? $O(n^2 \log^5 n) \Rightarrow \tilde{O}(n^2)$

Housekeeping

"soft O"

Who cares about logs? $O(n^2 \log^5 n) \Rightarrow \tilde{O}(n^2)$

- Ex. Dijkstra's runs in $\tilde{O}(m)$

Housekeeping

"soft O "

Who cares about logs? $O(n^2 \log^5 n) \Rightarrow \tilde{O}(n^2)$

- Ex. Dijkstra's runs in $\tilde{O}(m)$

G has $k < n$ negative edges.

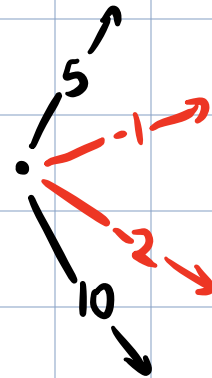
Housekeeping

"soft O"

Who cares about logs? $O(n^2 \log^5 n) \Rightarrow \tilde{O}(n^2)$

- Ex. Dijkstra's runs in $\tilde{O}(m)$

G has $k < n$ negative edges.



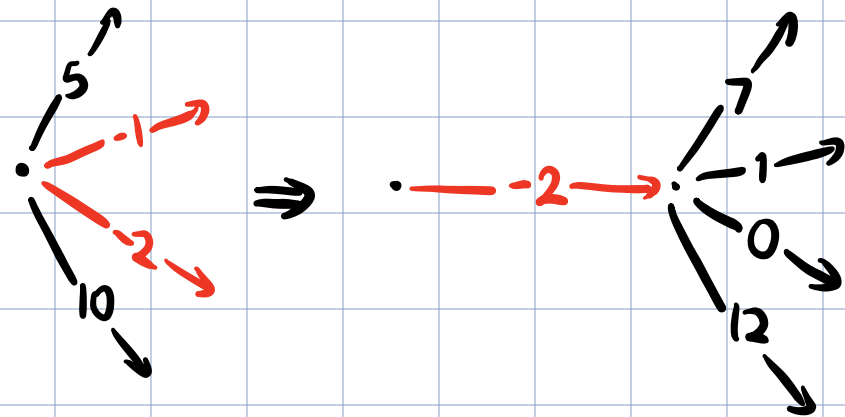
Housekeeping

"soft 0"

Who cares about logs? $O(n^2 \log^5 n) \Rightarrow \tilde{O}(n^2)$

- Ex. Dijkstra's runs in $\tilde{O}(m)$

G has $k < n$ negative edges.



Better than Bellman-Ford

(s,t) h -hop distance:

Better than Bellman-Ford

(s,t) h -hop distance:

$$d^h(s,t)$$

Better than Bellman-Ford

(s,t) h -hop distance:

$$d^h(s,t) := \min \{ l(W) \mid W \text{ an } (s,t)\text{-walk with at most } h \text{ edges} \}$$

Better than Bellman-Ford

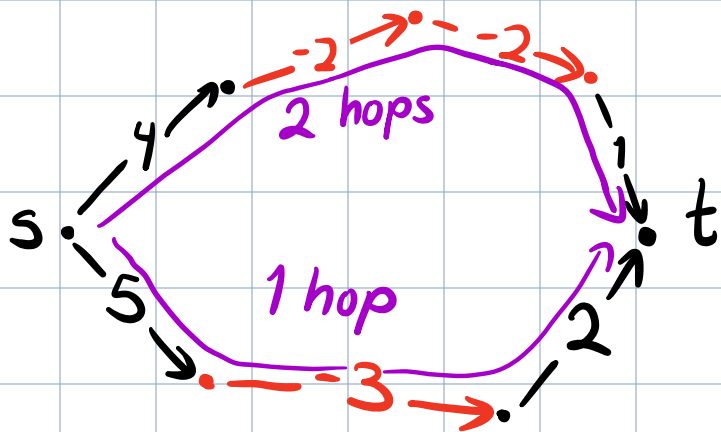
(s,t) h -hop distance:

$$d^h(s,t) := \min \{ l(W) \mid W \text{ an } (s,t)\text{-walk with at most } h \text{ "h hops" negative edges} \}$$

Better than Bellman-Ford

(s,t) h -hop distance:

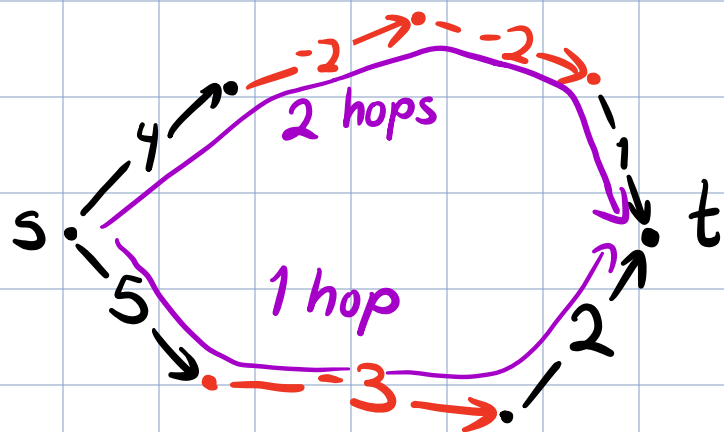
$$d^h(s,t) := \min \{ l(W) \mid W \text{ an } (s,t)\text{-walk with at most } h \text{ "h hops" negative edges} \}$$



Better than Bellman-Ford

(s,t) h -hop distance:

$$d^h(s,t) := \min \{ l(W) \mid W \text{ an } (s,t)\text{-walk with at most } h \text{ "h hops" negative edges} \}$$



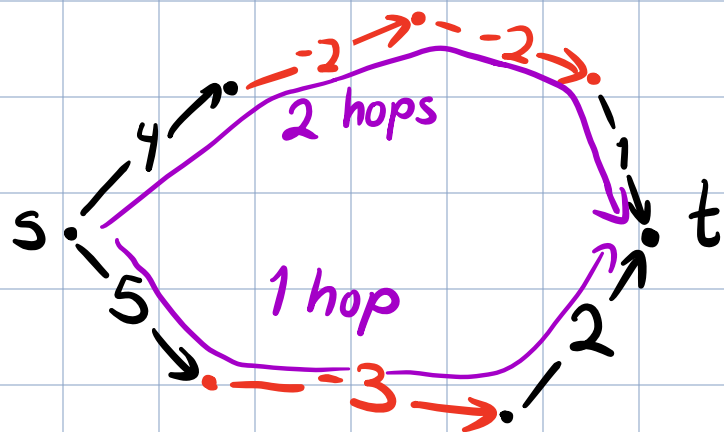
$$d^1(s,t) =$$

$$d^2(s,t) =$$

Better than Bellman-Ford

(s,t) h -hop distance:

$$d^h(s,t) := \min \{ l(W) \mid W \text{ an } (s,t)\text{-walk with at most } h \text{ "h hops" negative edges} \}$$



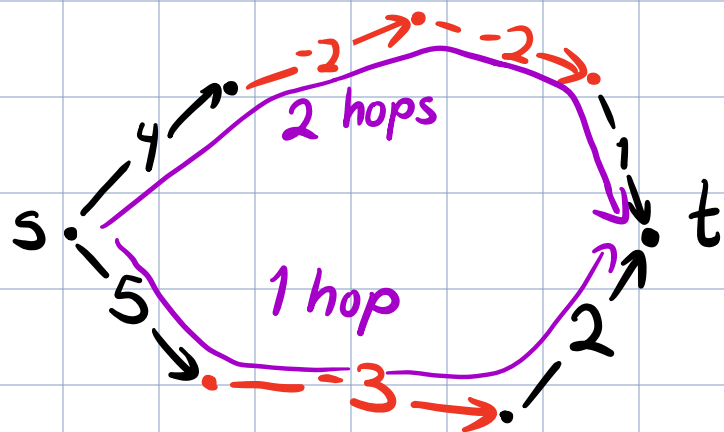
$$d^1(s,t) = 4$$

$$d^2(s,t) =$$

Better than Bellman-Ford

(s,t) h -hop distance:

$$d^h(s,t) := \min \{ l(W) \mid W \text{ an } (s,t)\text{-walk with at most } h \text{ "h hops" negative edges} \}$$



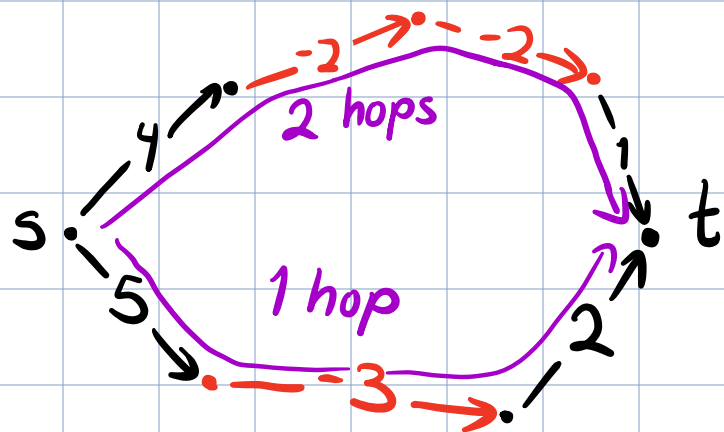
$$d^1(s,t) = 4$$

$$d^2(s,t) = 1$$

Better than Bellman-Ford

(s,t) h -hop distance:

$$d^h(s,t) := \min \{ l(W) \mid W \text{ an } (s,t)\text{-walk with at most } h \text{ "h hops" negative edges} \}$$



$$d^1(s,t) = 4$$

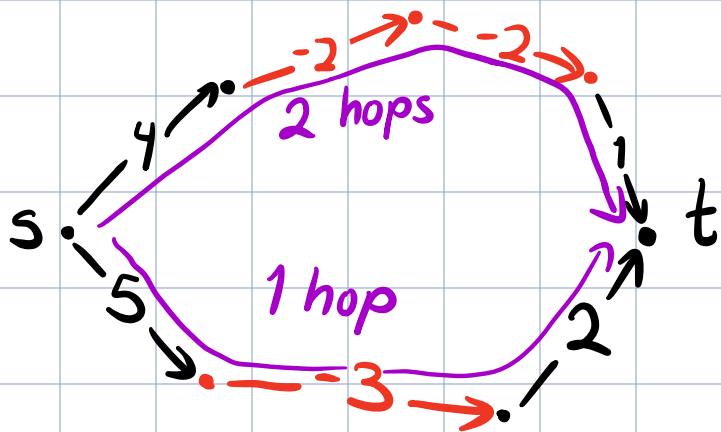
$$d^2(s,t) = 1$$

If G has $k \ll n$ negative edges, then $d(s,t) = d^k(s,t)$

Better than Bellman-Ford

(s,t) h -hop distance:

$$d^h(s,t) := \min \{ l(W) \mid W \text{ an } (s,t)\text{-walk with at most } h \text{ "h hops" negative edges} \}$$



$$d^1(s,t) = 4$$

$$d^2(s,t) = 1$$

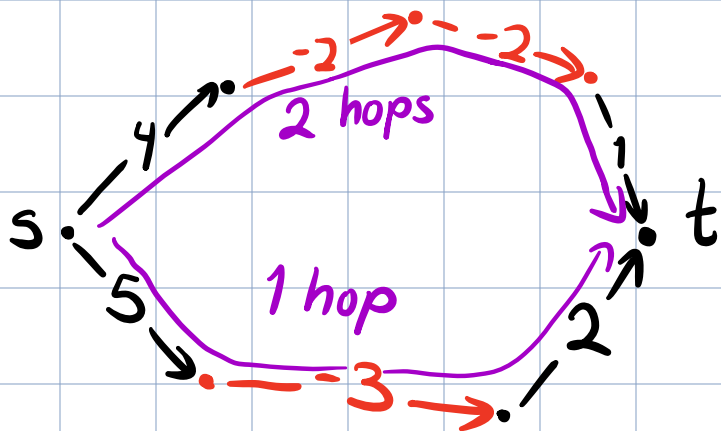
If G has $k \ll n$ negative edges, then $d(s,t) = d^k(s,t)$

Exercise: Compute $d^h(s, \cdot)$ in $O(h(m + n \log n)) = \tilde{O}(hm)$ time

Better than Bellman-Ford

(s,t) h -hop distance:

$$d^h(s,t) := \min \{ l(W) \mid W \text{ an } (s,t)\text{-walk with at most } h \text{ "h hops" negative edges} \}$$



$$d^1(s,t) = 4$$

$$d^2(s,t) = 1$$

If G has $k \ll n$ negative edges, then $d(s,t) = d^k(s,t)$

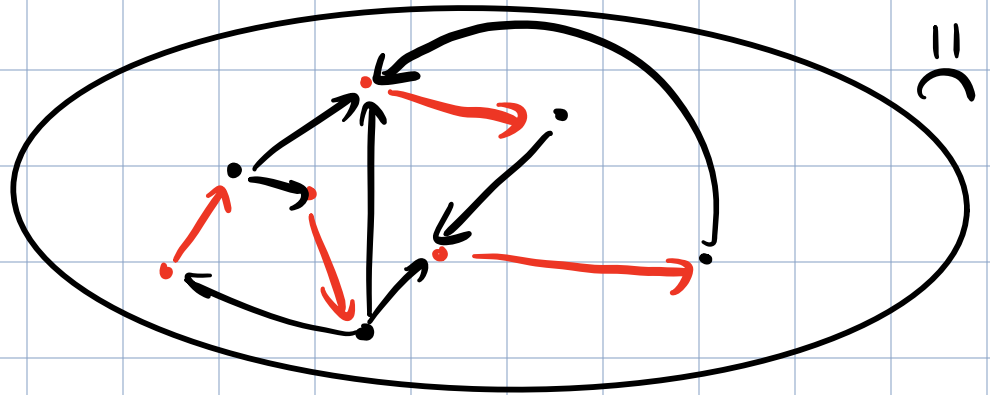
Exercise: Compute $d^h(s, \cdot)$ in $O(h(m + n \log n)) = \tilde{O}(hm)$ time

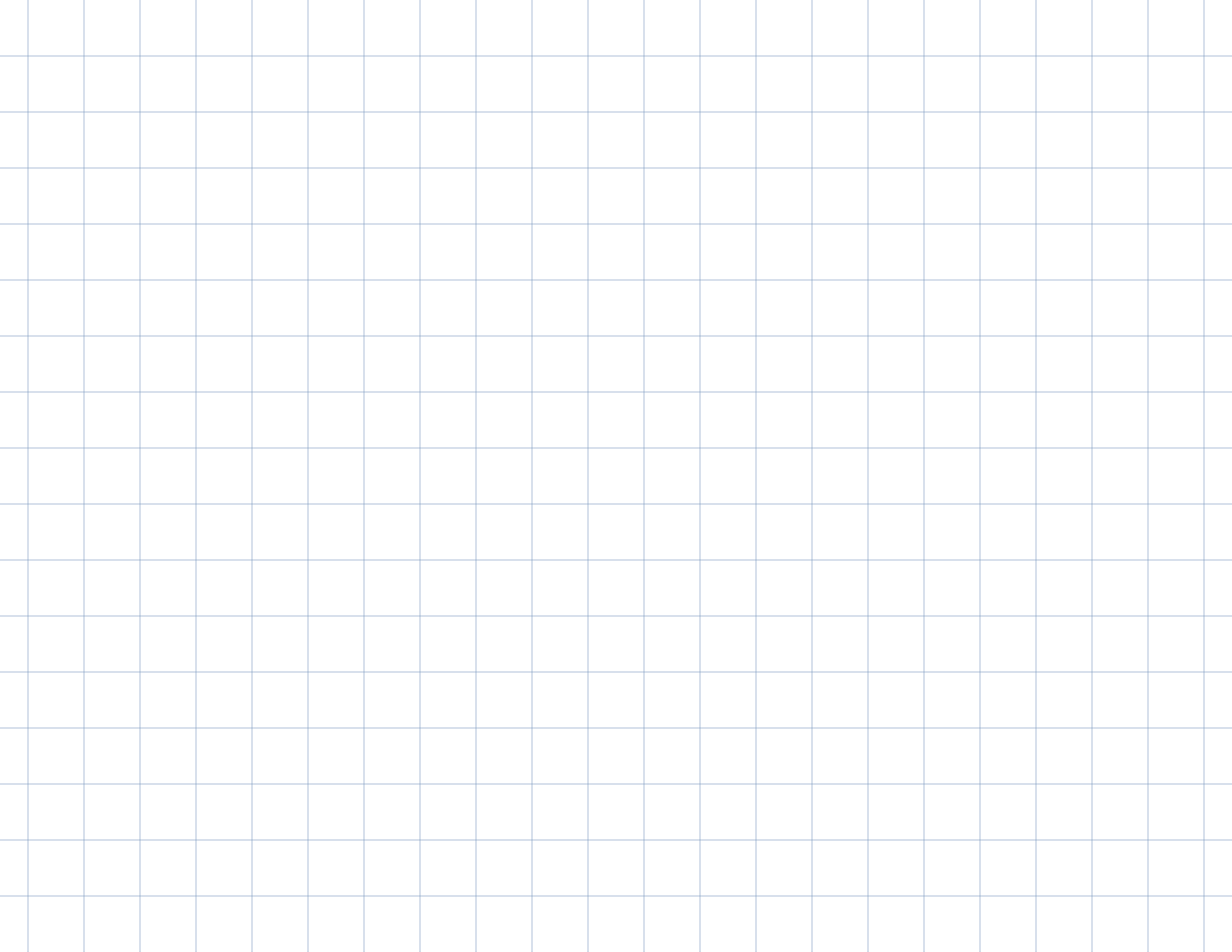
Hint: $\text{dist}(v, i) := \min \{ l(W) \mid W \text{ an } (s,v)\text{-walk with at most } i \text{ negative edges} \}$

Can compute $d^k(s, \cdot) = d(s, \cdot)$ in $\tilde{O}(km)$ time.

Prev
Slide

Problem: What if $k \approx n$?





(s,t) -walks W have length

$$l_p(W) = l(W) + \varphi(s) - \varphi(t)$$

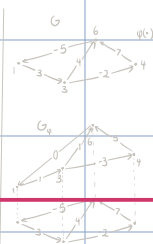
Potentials

Say we have vertex weights
 $\varphi: V \rightarrow \mathbb{R}$

Define reduced weights l_p by

$$l_p(u,v) = l(u,v) + \varphi(u) - \varphi(v)$$

"Subtract uphill change/
add downhill change"



Distances in G_p give

distances in G

W a shortest (s,t) -walk in G

$$d_{G_p}(s,t) = d(s,t) + \varphi(s) - \varphi(t)$$

W a shortest (s,t) -walk in G_p !

(s,t) -walks W have length

$$l_p(W) = l(W) + \varphi(s) - \varphi(t)$$

Distances in G_φ give
distances in G

W a shortest (s,t) -walk in G

$$d_{G_\varphi}(s,t) = d(s,t) + \varphi(s) - \varphi(t)$$

W a shortest (s,t) -walk in $G_\varphi!$

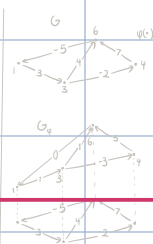
Potentials

Say we have vertex weights
 $\varphi: V \rightarrow \mathbb{R}$

Define reduced weights l_p by

$$l_p(u,v) = l(u,v) + \varphi(u) - \varphi(v)$$

"Subtract uphill change/
add downhill change"

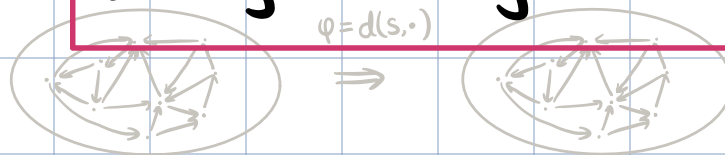


Neutralizing G

Suppose some vertex s can reach all $v \in V$. Let $\varphi(v) := d(s,v)$. Then,

$$l_p(u,v) = l(u,v) + \varphi(u) - \varphi(v) = l(u,v) + d(s,u) - d(s,v) \geq 0$$

Distance potentials make
all edges nonnegative



How to pick s ?



$$d(s^*, v) = d(v, v) = \min_{u \in V} d(u, v)$$

(s,t) -walks W have length

$$l_\varphi(W) = l(W) + \varphi(s) - \varphi(t)$$

Distances in G_φ give
distances in G

If a shortest (s,t) -walk in G
 $\iff$
 If a shortest (s,t) -walk in G_φ !

$$d_{G_\varphi}(s,t) = d(s,t) + \varphi(s) - \varphi(t)$$

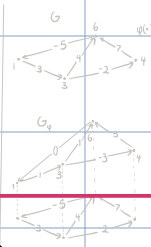
Potentials

Say we have vertex weights
 $\varphi: V \rightarrow \mathbb{R}$

Define reduced weights l_φ by

$$l_\varphi(u,v) = l(u,v) + \varphi(u) - \varphi(v)$$

"Subtract uphill change/
add downhill change"



Neutralizing G

Suppose some vertex s can reach all $v \in V$. Let $\varphi(v) := d(s,v)$. Then,

$$l_\varphi(u,v) = l(u,v) + \varphi(u) - \varphi(v) = l(u,v) + d(s,u) - d(s,v) \geq 0$$

Distance potentials make
all edges nonnegative



How to pick s ?



$$d(s^*, v) = d(v, v) = \min_{u \in V} d(u, v)$$

Can compute $d^h(s, \cdot)$ in $O(h(m + n \log n)) = \tilde{O}(hm)$ time

(s,t) -walks W have length

$$l_p(W) = l(W) + \varphi(s) - \varphi(t)$$

Distances in G_φ give
distances in G

W a shortest (s,t) -walk in G

W a shortest (s,t) -walk in G_φ !

$$d_{G_\varphi}(s,t) = d(s,t) + \varphi(s) - \varphi(t)$$

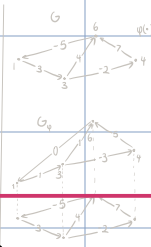
Potentials

Say we have vertex weights
 $\varphi: V \rightarrow \mathbb{R}$

Define reduced weights l_p by

$$l_p(u,v) = l(u,v) + \varphi(u) - \varphi(v)$$

"Subtract uphill change/
add downhill change"



Neutralizing G

Suppose some vertex s can reach all $v \in V$. Let $\varphi(v) := d(s,v)$. Then,

$$l_p(u,v) = l(u,v) + \varphi(u) - \varphi(v) = l(u,v) + d(s,u) - d(s,v) \geq 0$$

Distance potentials make
all edges nonnegative



How to pick s ?



$$d(s^*, v) = d(v, v) = \min_{u \in V} d(u, v)$$

Can compute $d^h(s, \cdot)$ in $O(h(m + n \log n)) = \tilde{O}(hm)$ time

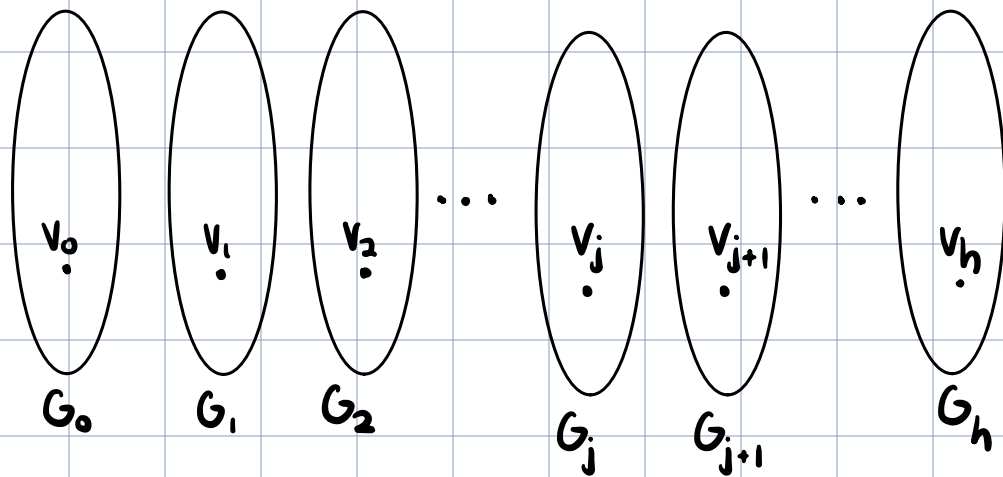
Goal: SSSP faster than $\tilde{O}(mk)$

Hop Reducers Layered graph H

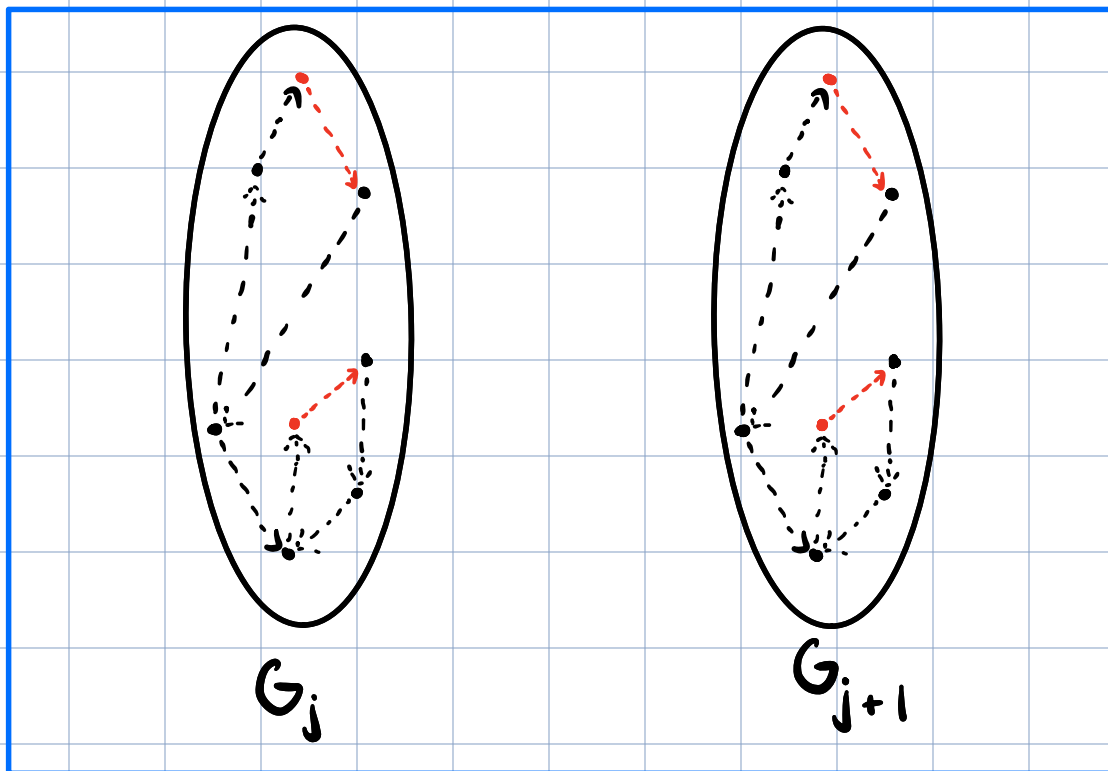
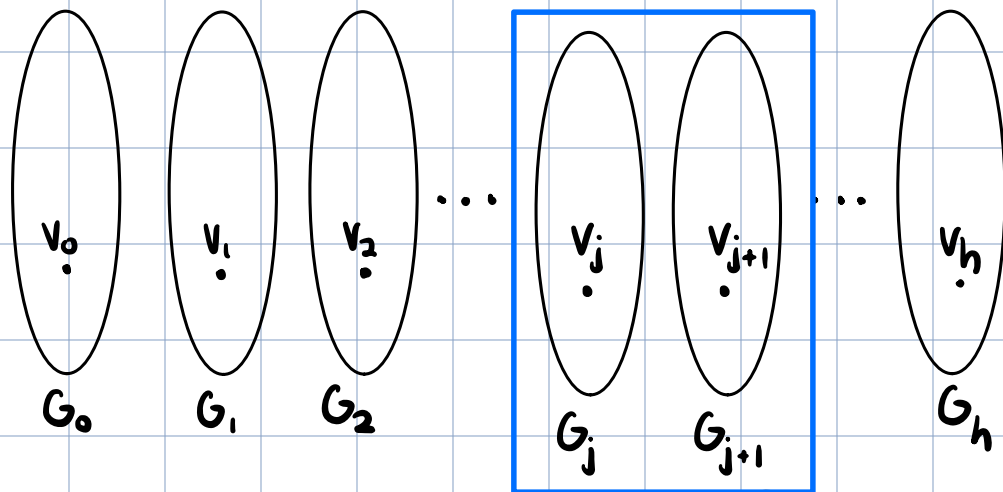


G_0

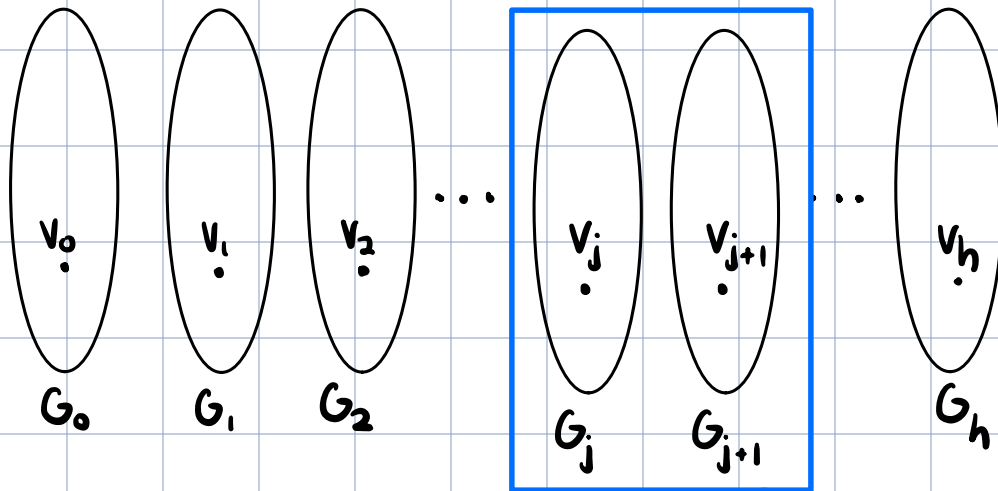
Hop Reducers Layered graph H



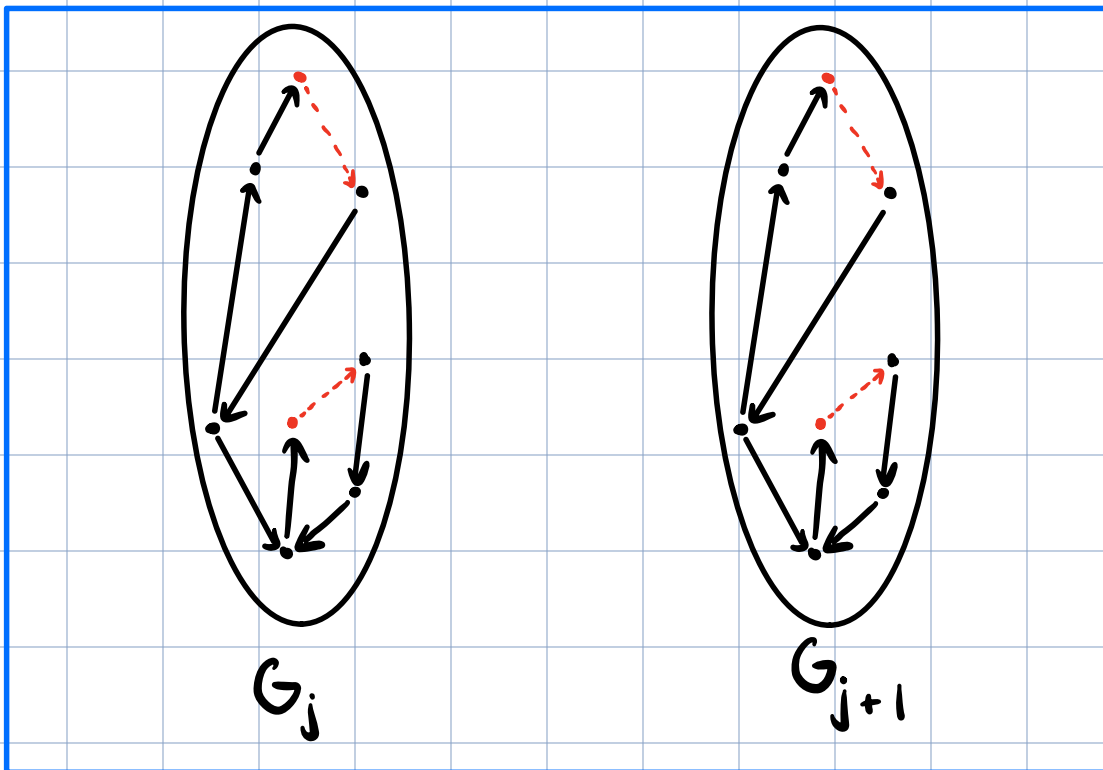
Hop Reducers Layered graph H



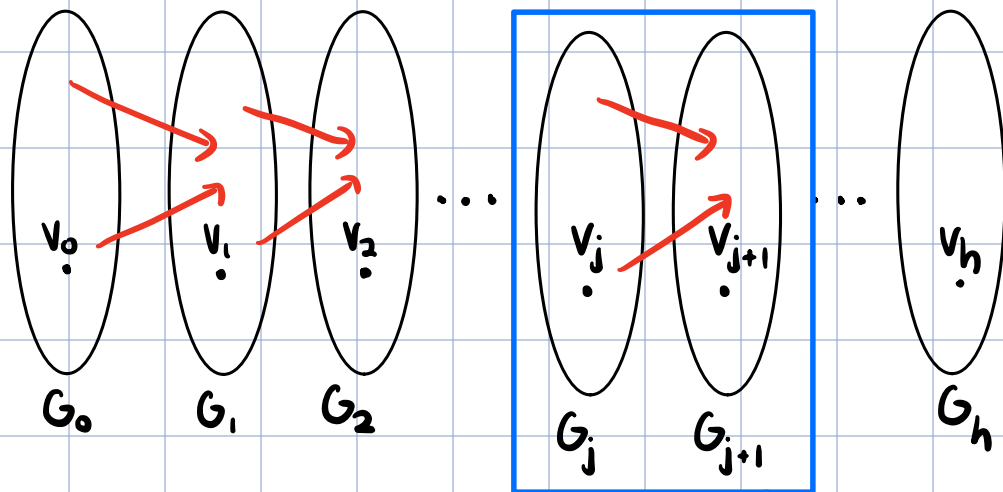
Hop Reducers Layered graph H



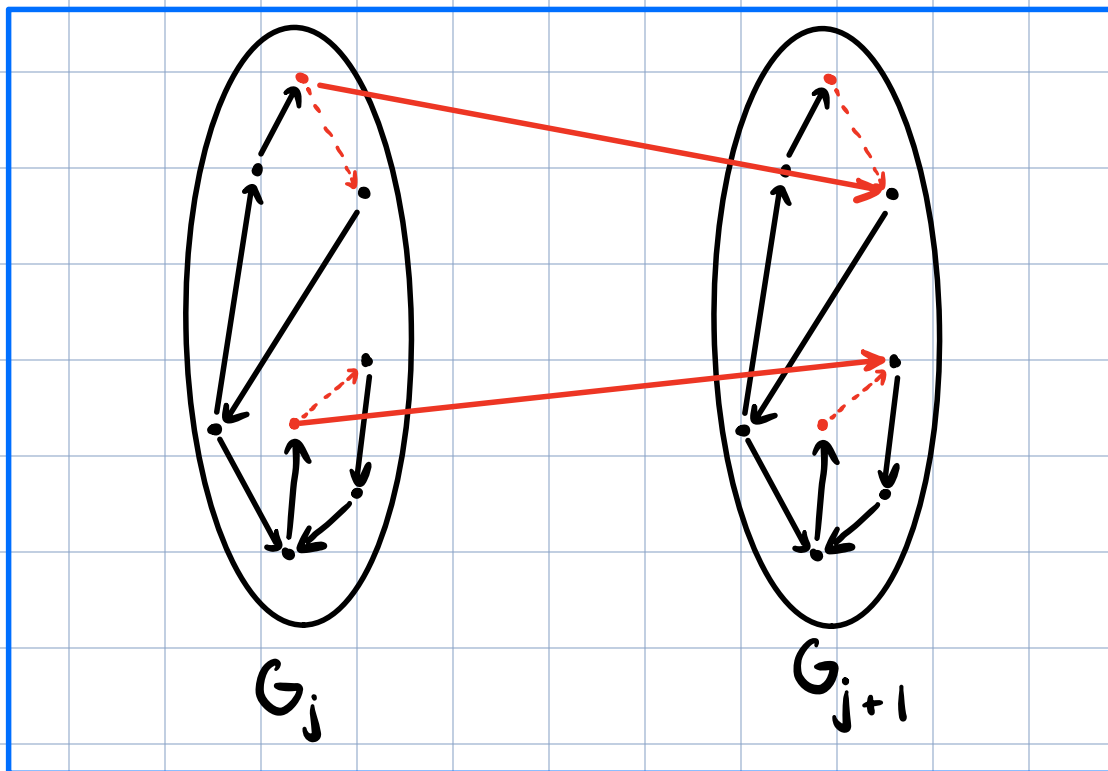
• Add ≥ 0 edges within layer



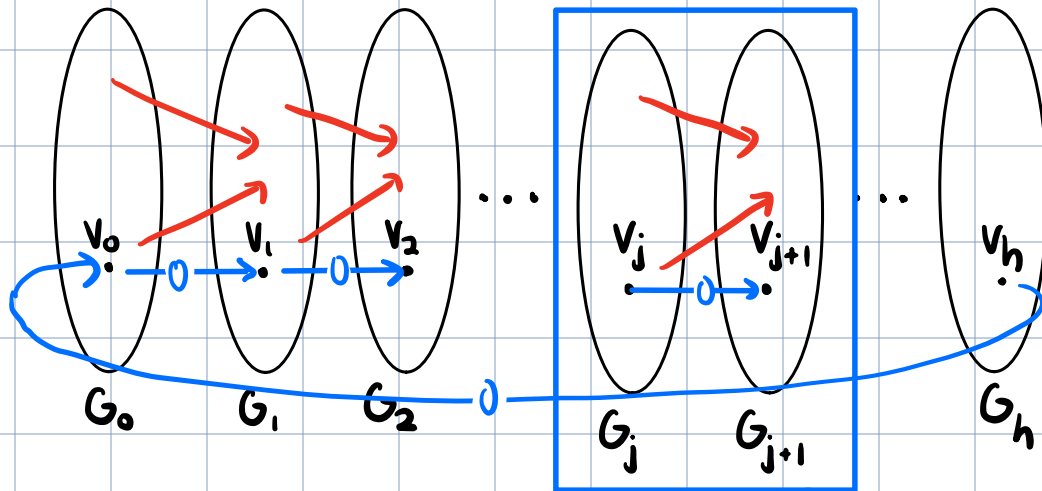
Hop Reducers Layered graph H



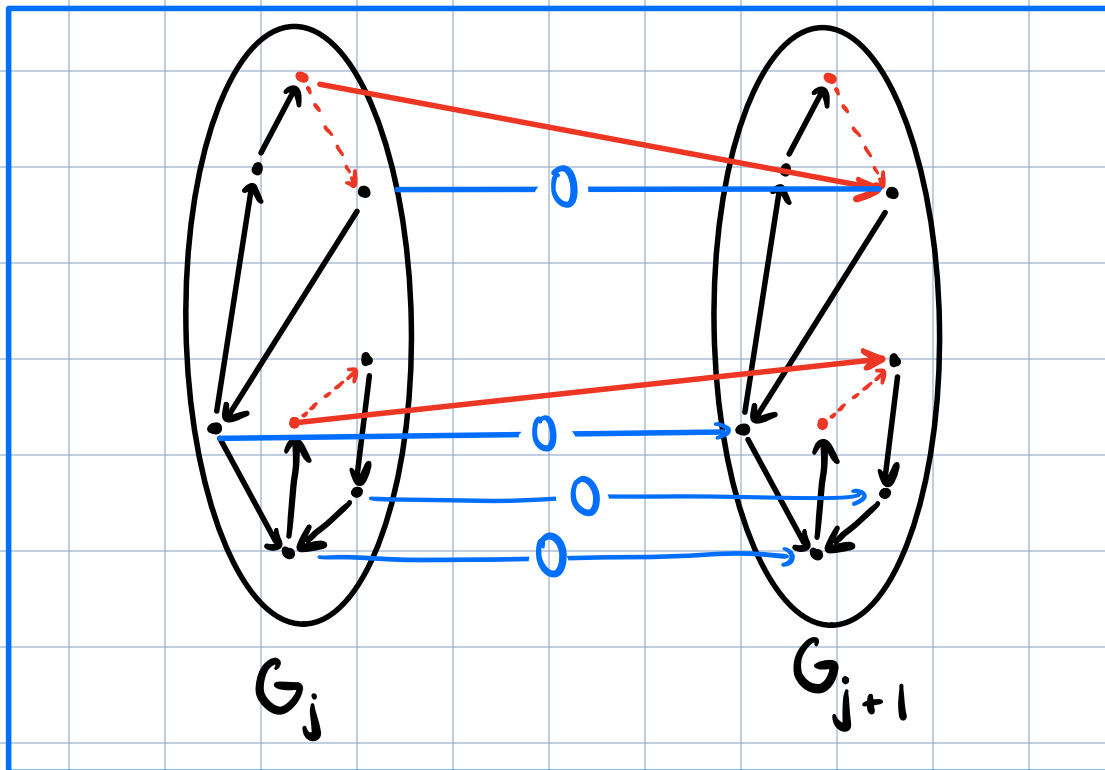
- Add ≥ 0 edges within layer
- Neg edges go forward one layer



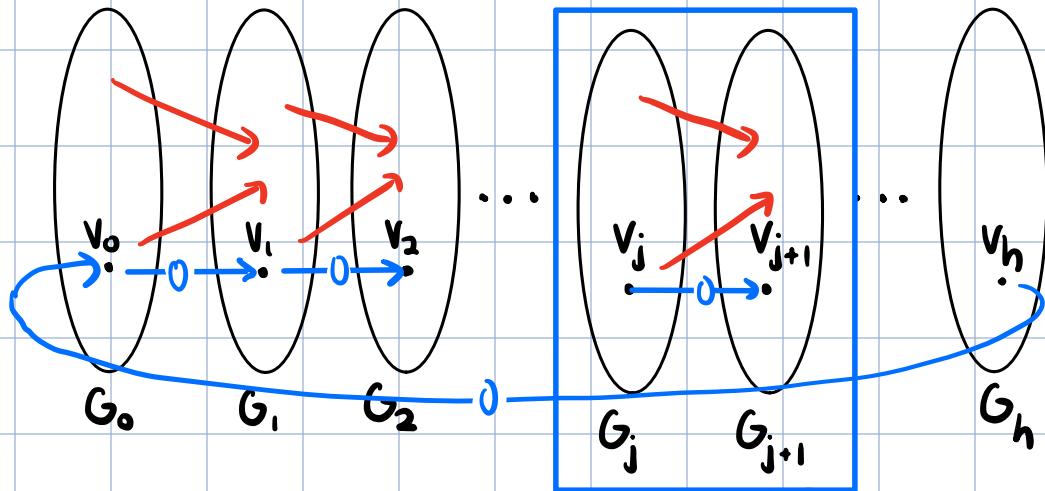
Hop Reducers Layered graph H



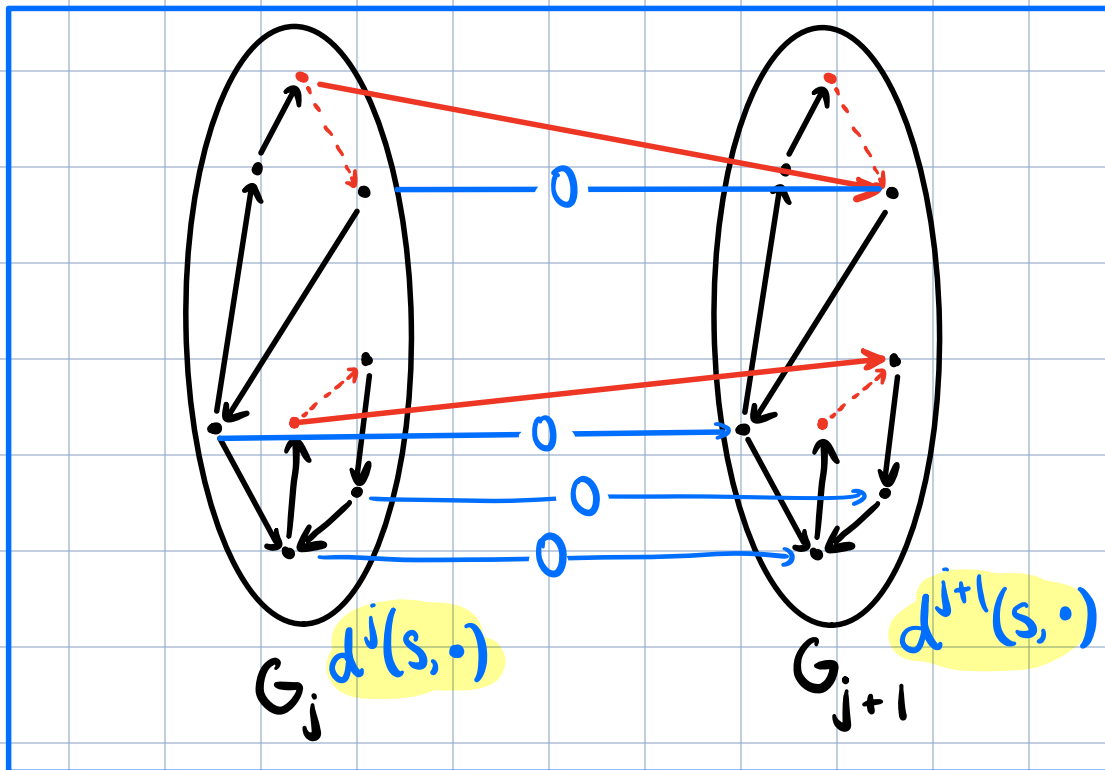
- Add ≥ 0 edges within layer
- Neg edges go forward one layer
- Length 0 edges going forward and looping back



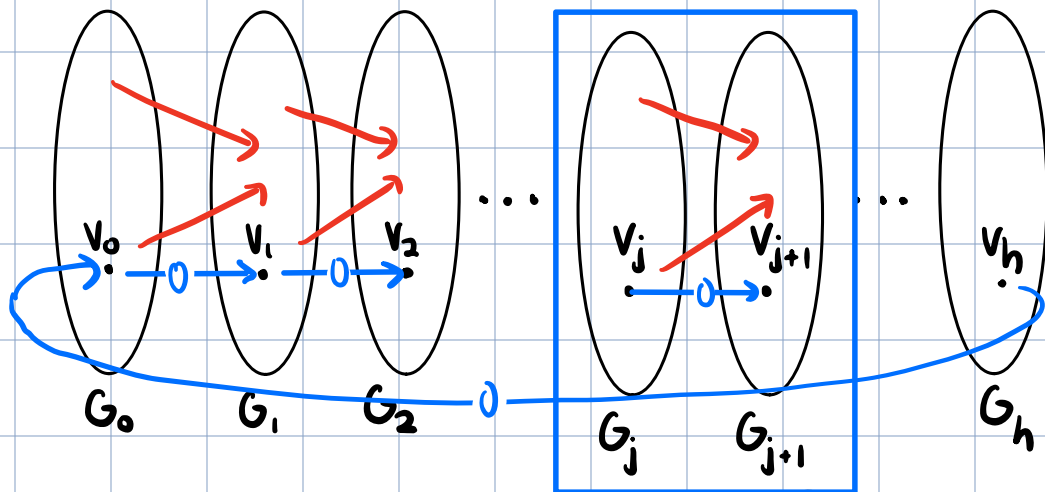
Hop Reducers Layered graph H



- Add ≥ 0 edges within layer
- Neg edges go forward one layer
- Length 0 edges going forward and looping back
- Apply potential $\varphi(v_j) = d^j(s, v)$

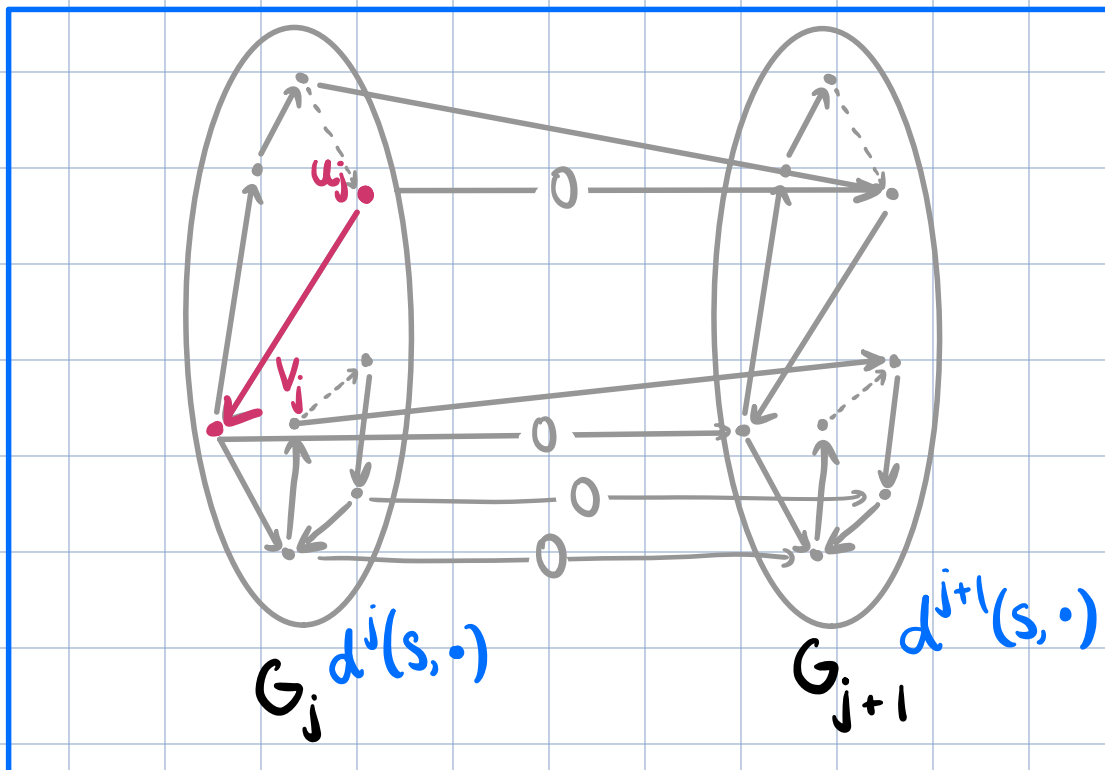


Hop Reducers Layered graph H

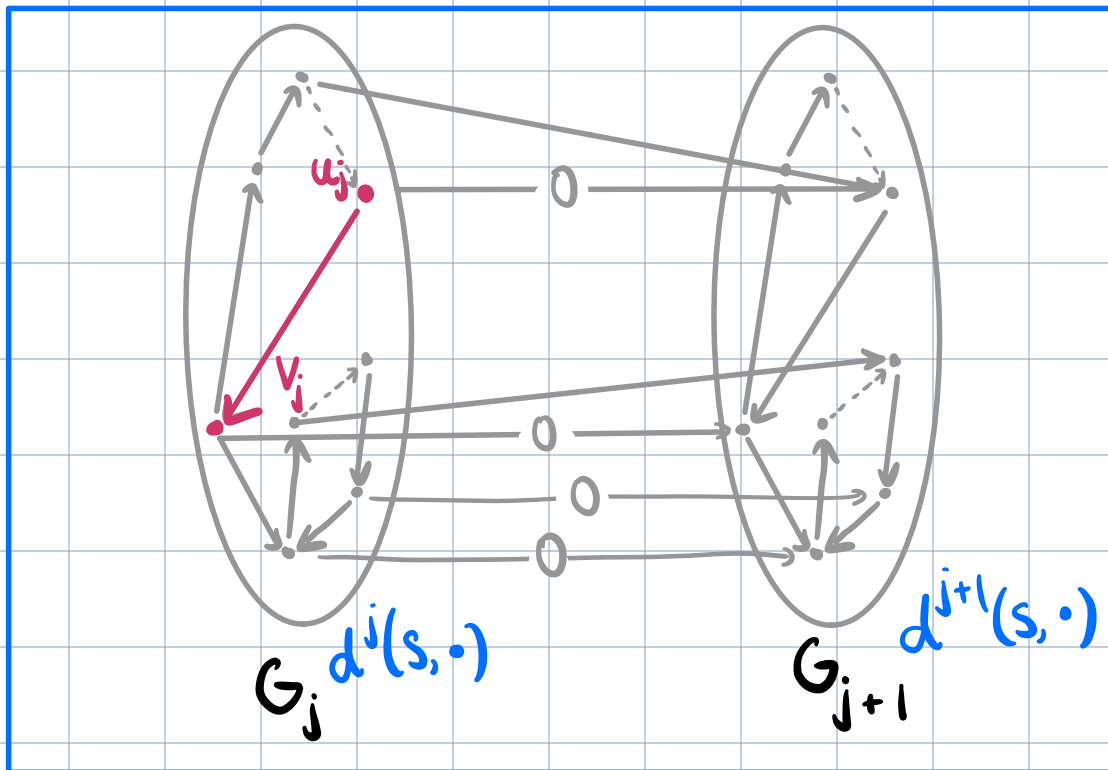
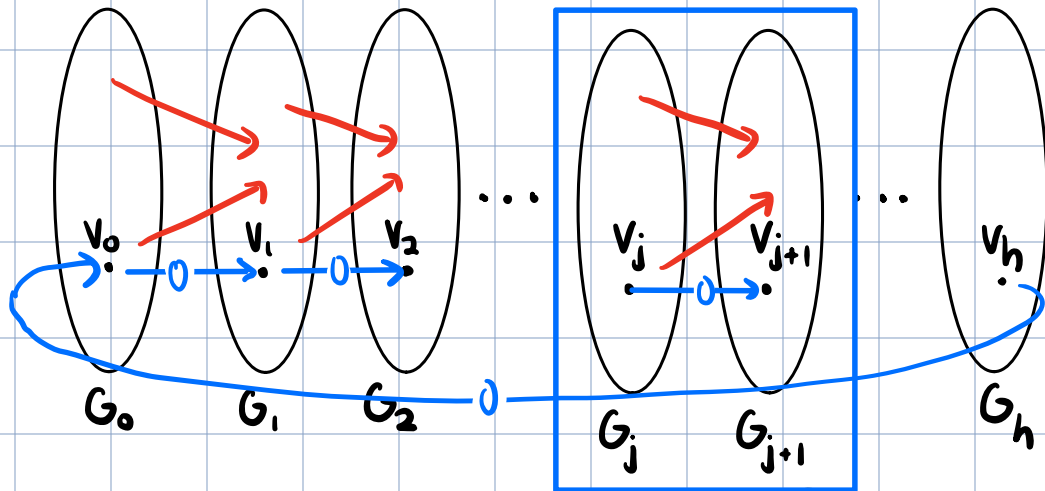


- Add ≥ 0 edges within layer
- Neg edges go forward one layer
- Length 0 edges going forward and looping back
- Apply potential $\varphi(v_j) = d^j(s, v)$
- ≥ 0 edges (u_j, v_j) :

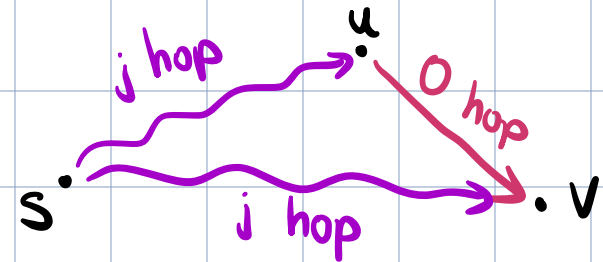
$$l(u, v) + d^j(s, u) - d^j(s, v)$$



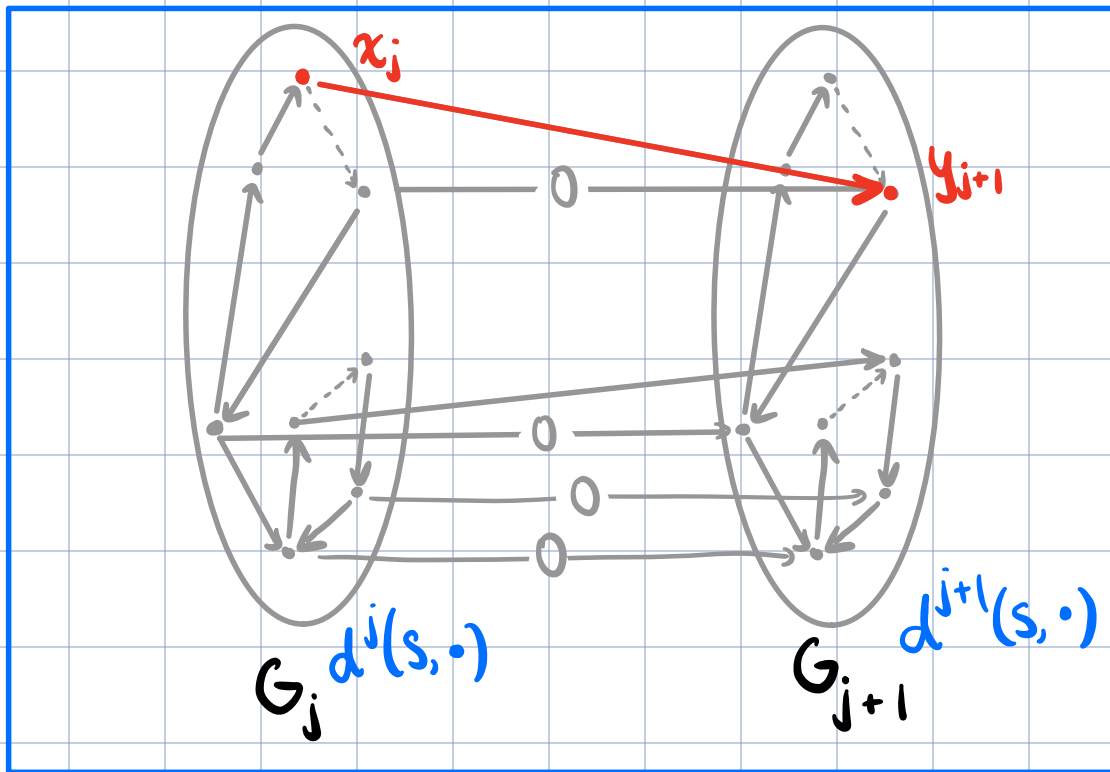
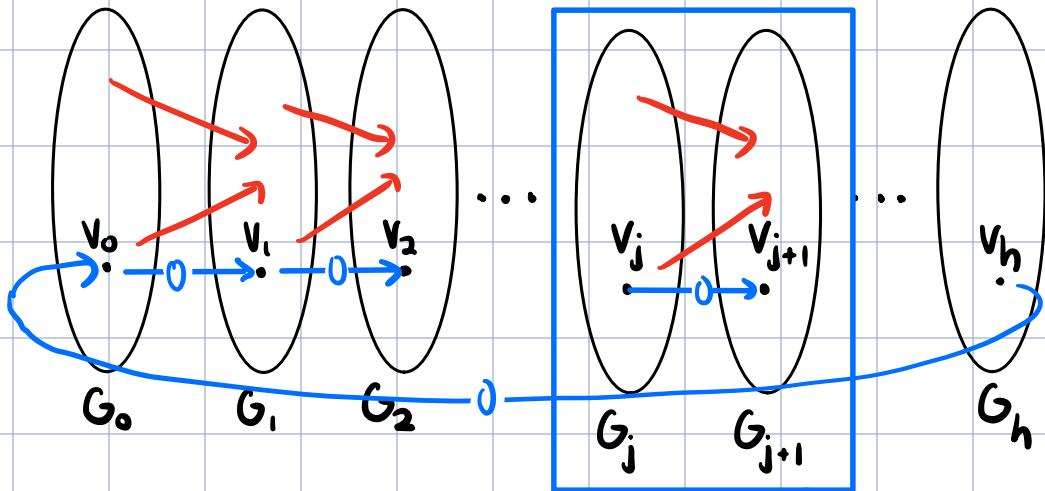
Hop Reducers Layered graph H



- Add ≥ 0 edges within layer
 - Neg edges go forward one layer
 - Length 0 edges going forward and looping back
 - Apply potential $\varphi(v_j) = d^j(s, v)$
 - ≥ 0 edges (u_j, v_j) :
- $$l(u, v) + d^j(s, u) - d^j(s, v) \geq 0$$

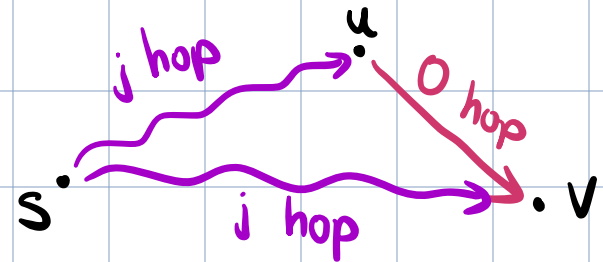


Layered graph H



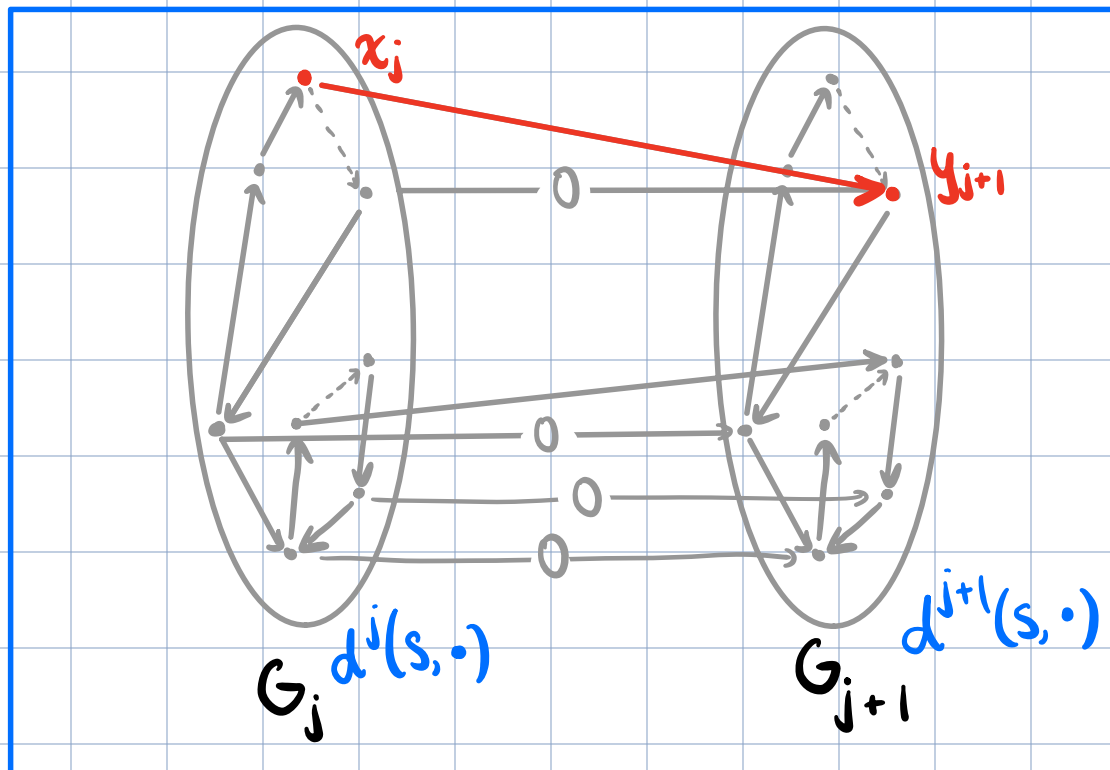
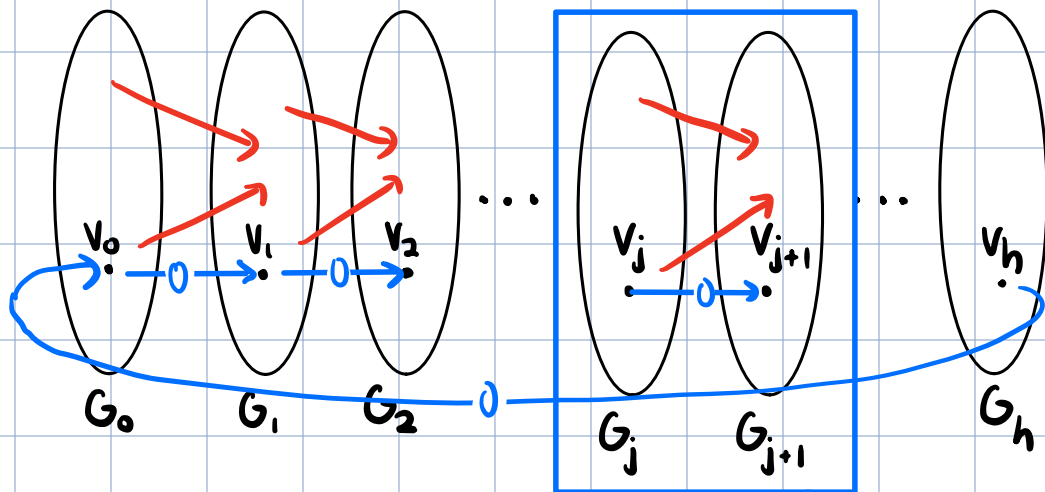
- Add ≥ 0 edges within layer
- Neg edges go forward one layer
- Length 0 edges going forward and looping back
- Apply potential $\varphi(v_j) = d^j(s, v)$
- ≥ 0 edges (u_j, v_j) :

$$l(u, v) + d^j(s, u) - d^j(s, v) \geq 0$$



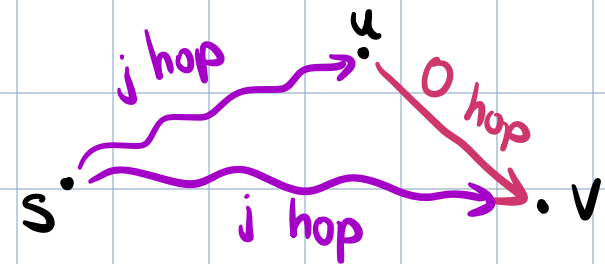
- Neg edges (x_j, y_{j+1}) :
 $l(x, y) + d^j(s, x) - d^{j+1}(s, y)$

Hop Reducers Layered graph H



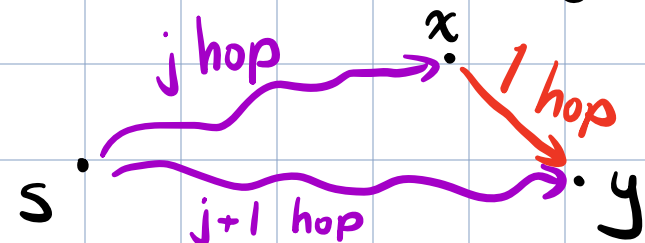
- Add ≥ 0 edges within layer
- Neg edges go forward one layer
- Length 0 edges going forward and looping back
- Apply potential $\varphi(v_j) = d^j(s, v)$
- ≥ 0 edges (u_j, v_j) :

$$l(u, v) + d^j(s, u) - d^j(s, v) \geq 0$$

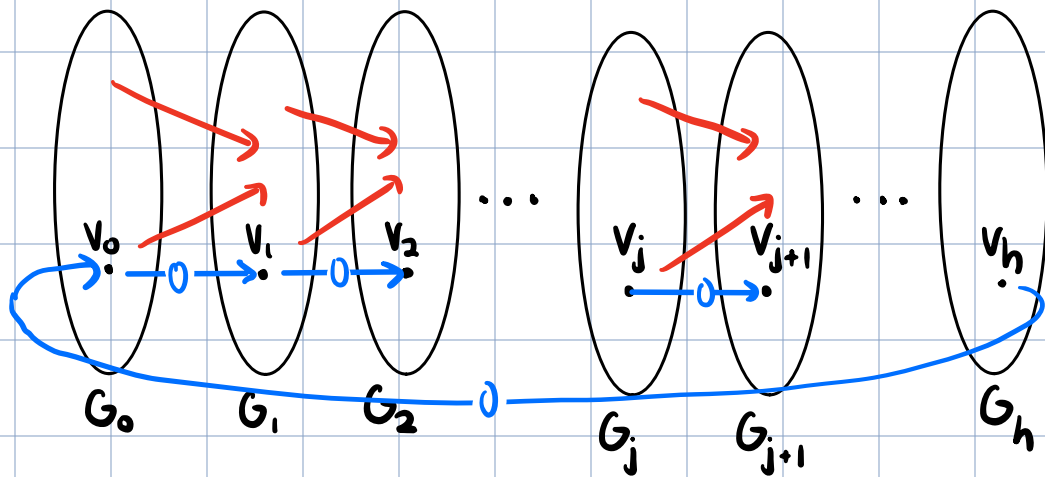


- Neg edges (x_j, y_{j+1}) :

$$l(x, y) + d^j(s, x) - d^{j+1}(s, y) \geq 0$$

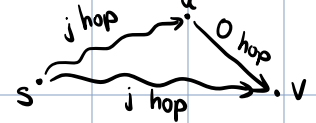


Hop Reducers Layered graph H



- Add ≥ 0 edges within layer
- Neg edges go forward one layer
- Length 0 edges going forward and looping back
- Apply potential $\phi(v_j) = d^j(s, v)$
- ≥ 0 edges (u_j, v_j) :

$$l(u, v) + d^j(s, u) - d^j(s, v) \geq 0$$

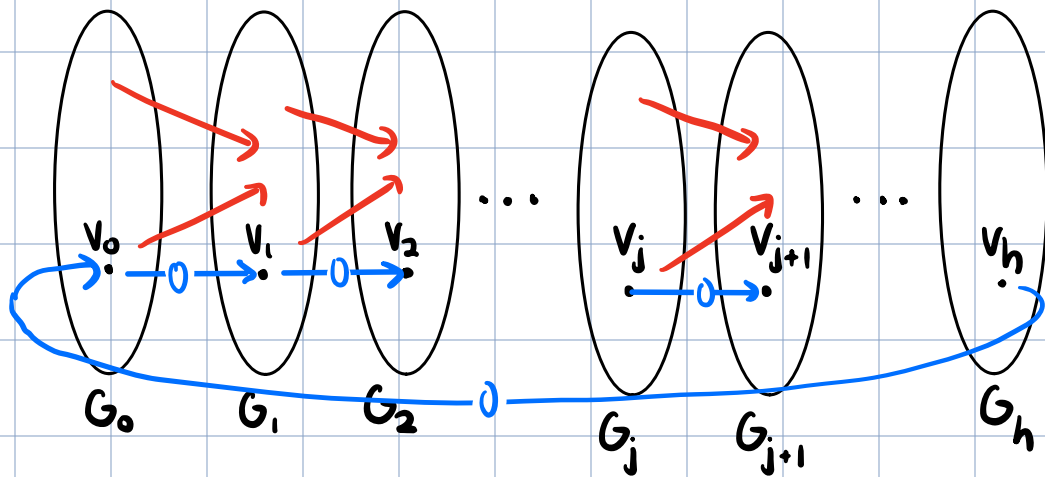


- Neg edges (x_j, y_{j+1}) :

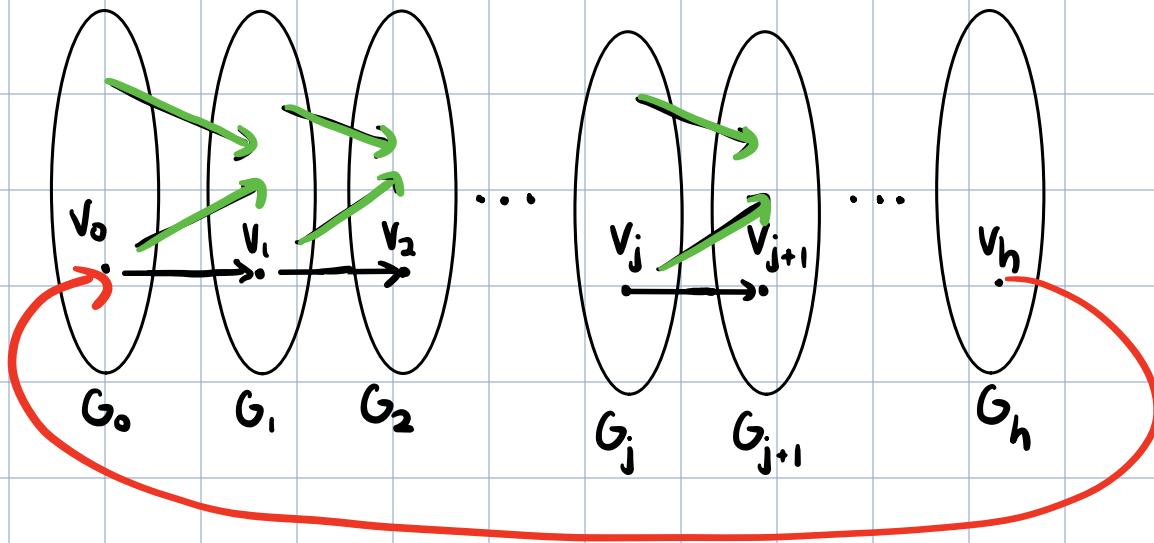
$$l(x, y) + d^j(s, x) - d^{j+1}(s, y) \geq 0$$



Hop Reducers Layered graph H

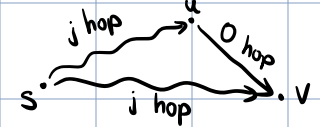


H_φ $\Downarrow$ reweight



- Add ≥ 0 edges within layer
- Neg edges go forward one layer
- Length 0 edges going forward and looping back
- Apply potential $\varphi(v_j) = d^j(s, v)$
- ≥ 0 edges (u_j, v_j) :

$$l(u, v) + d^j(s, u) - d^j(s, v) \geq 0$$

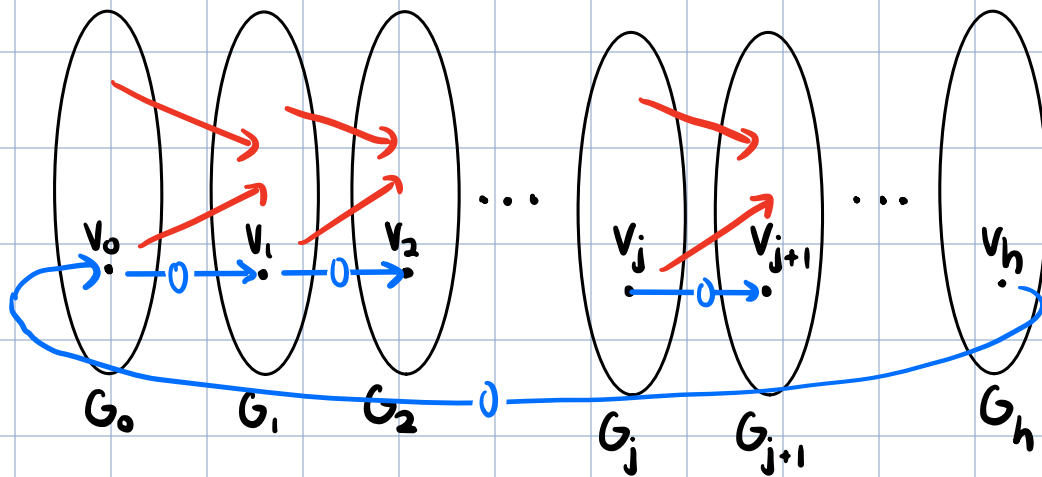


- Neg edges (x_j, y_{j+1}) :

$$l(x, y) + d^j(s, x) - d^{j+1}(s, y) \geq 0$$

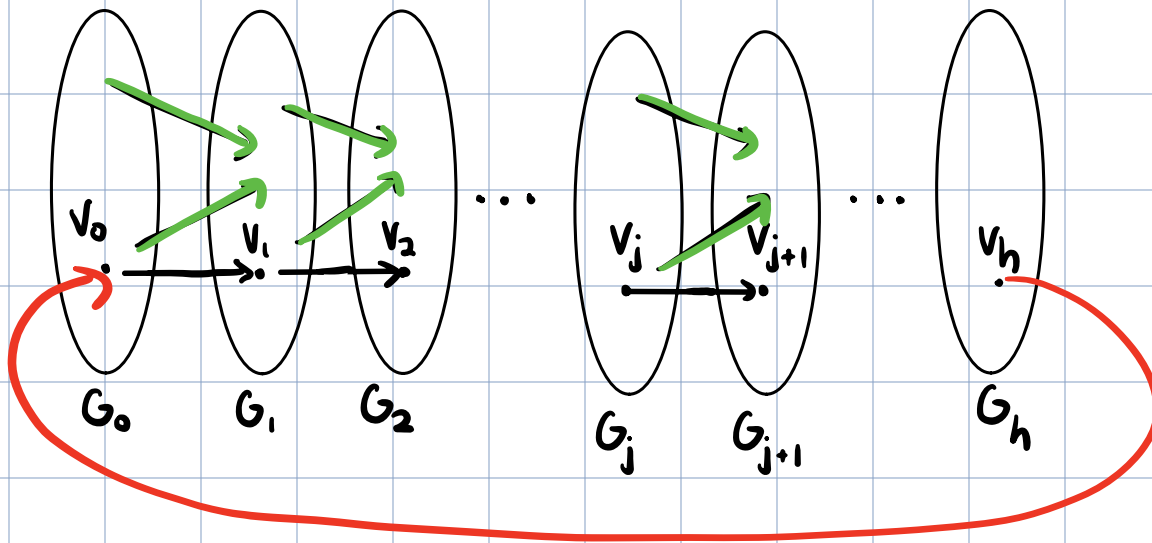


Hop Reducers Layered graph H



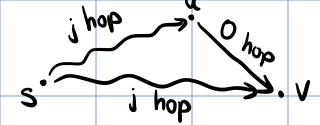
↓ reweight

H_φ



- Add ≥ 0 edges within layer
- Neg edges go forward one layer
- Length 0 edges going forward and looping back
- Apply potential $\varphi(v_j) = d^j(s, v)$
- ≥ 0 edges (u_j, v_j) :

$$l(u, v) + d^j(s, u) - d^j(s, v) \geq 0$$



- Neg edges (x_j, y_{j+1}) :

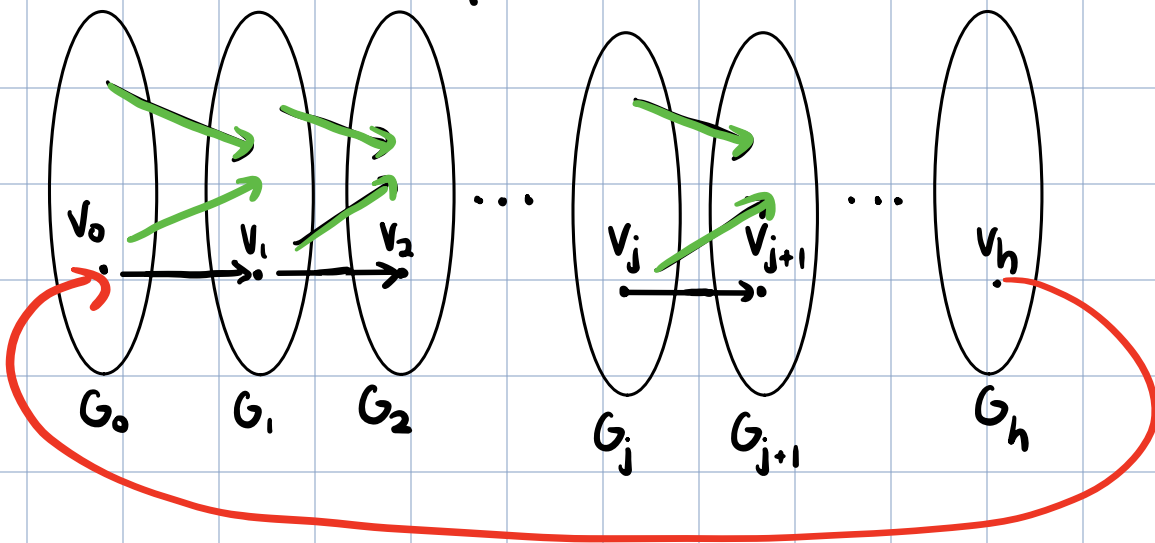
$$l(x, y) + d^j(s, x) - d^{j+1}(s, y) \geq 0$$



The only negative edges are the "loopback" edges.

Hop Reducers

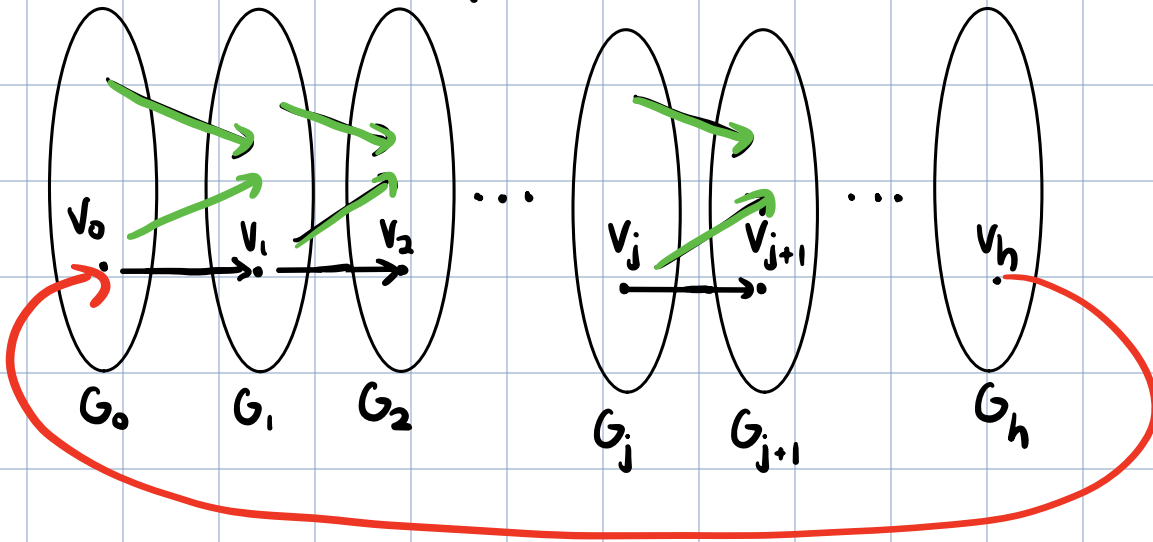
H_φ



The only negative edges are the "loopback" edges.

Hop Reducers

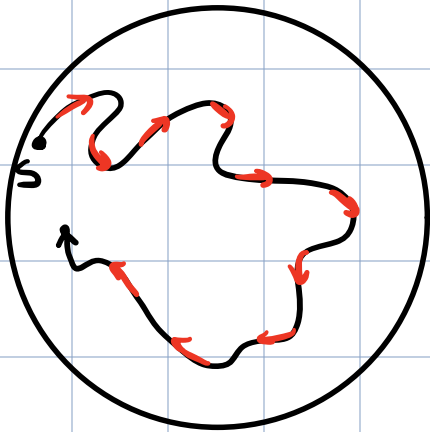
H_φ



The only negative edges are the "loopback" edges.

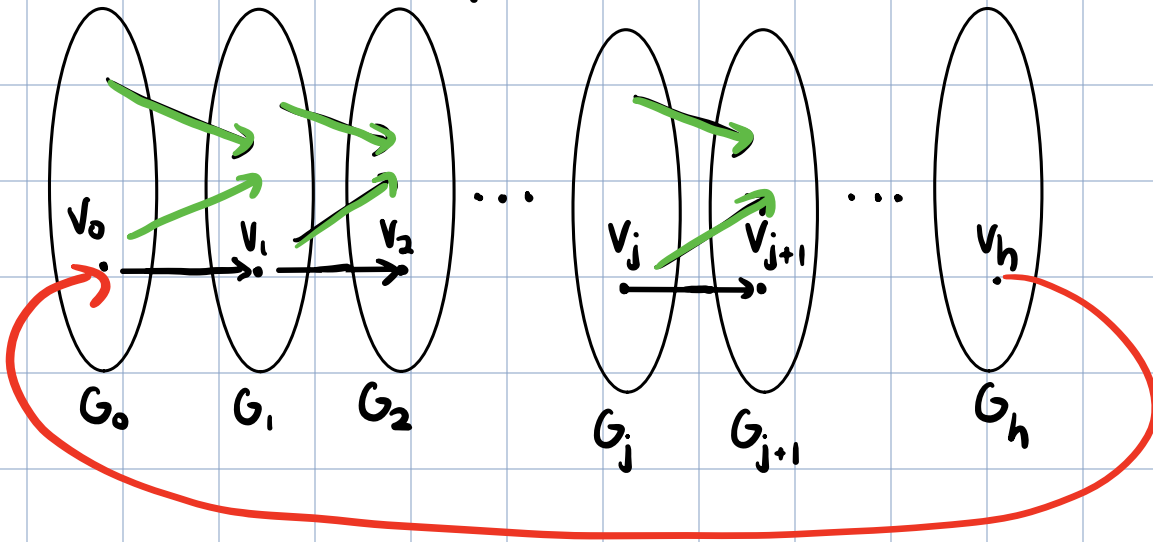
Mapping Paths

W some η -hop walk in G



Hop Reducers

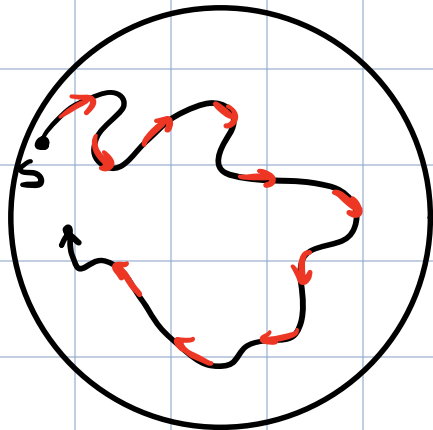
H_φ



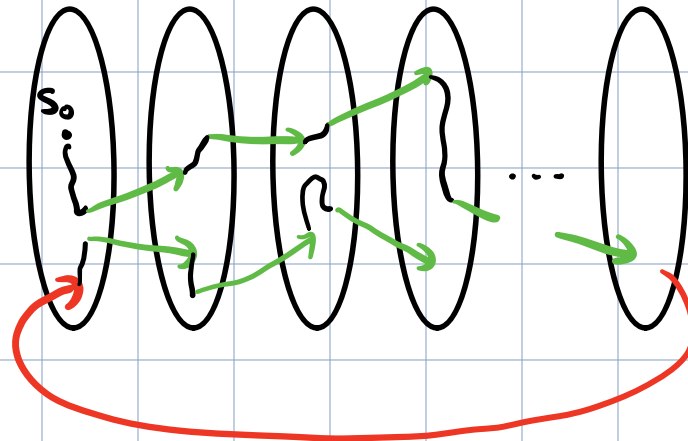
The only negative edges are the "loopback" edges.

Mapping Paths

W some η -hop walk in $G \implies W'$ in H_φ

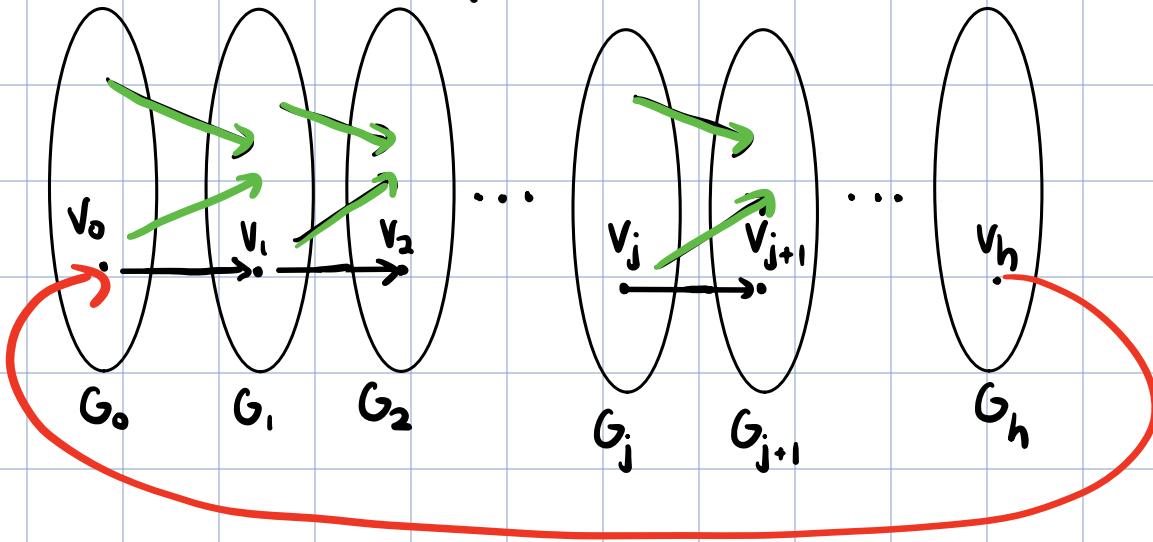


$\implies$



Hop Reducers

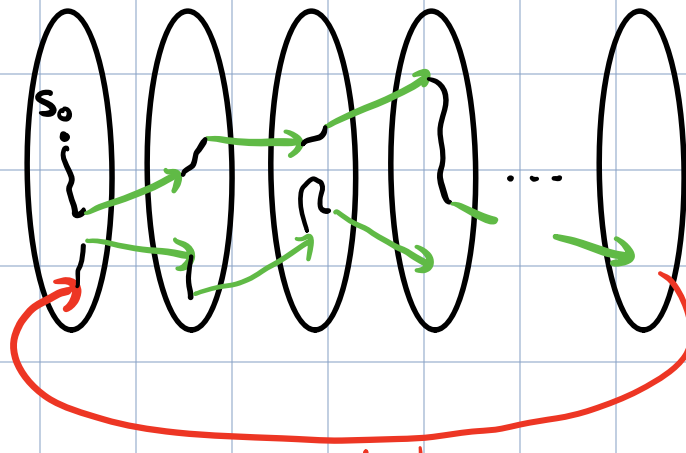
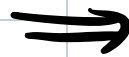
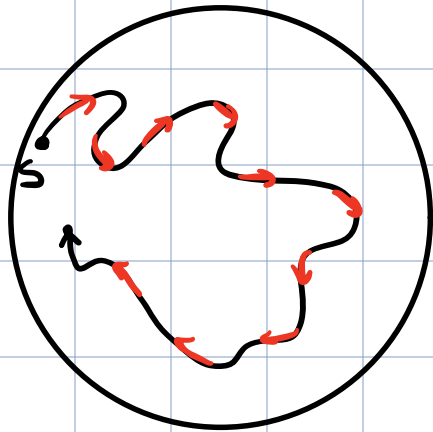
H_ψ



The only negative edges are the "loopback" edges.

Mapping Paths

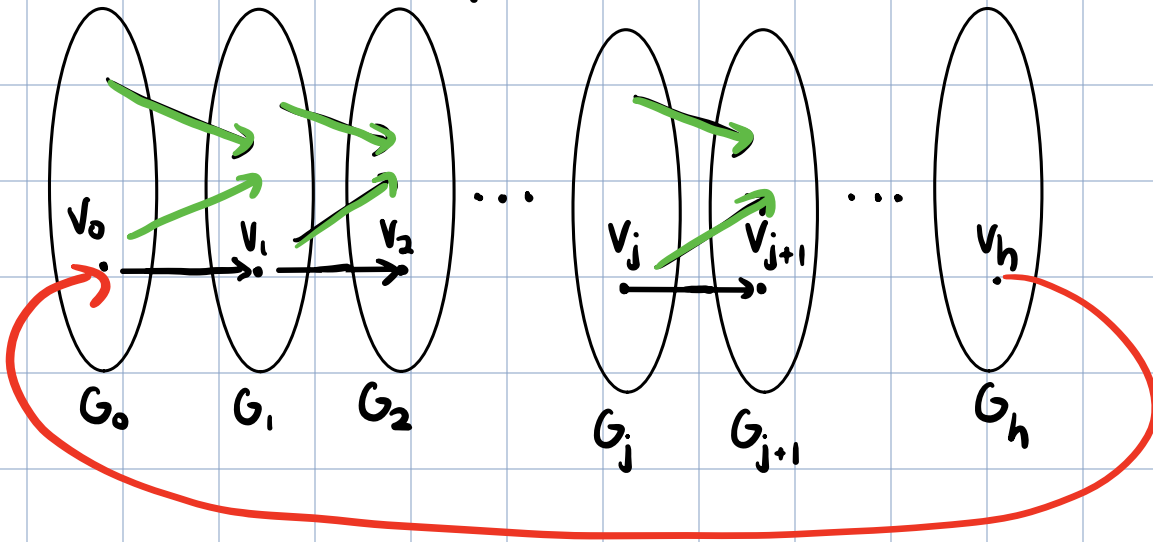
W some η -hop walk in $G \implies W'$ in H_ψ



loop once every h hops

Hop Reducers

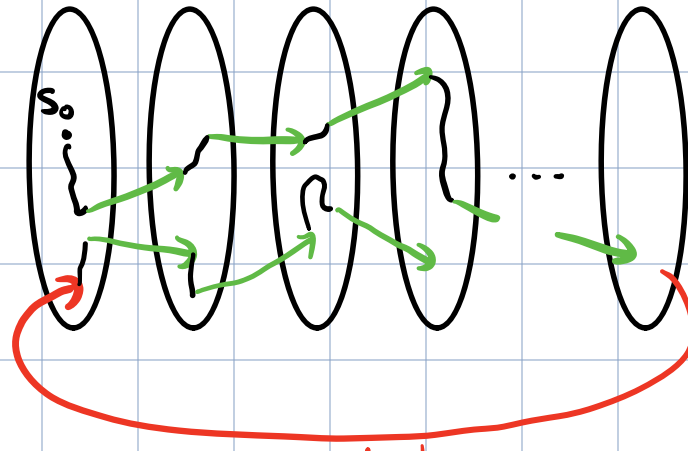
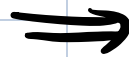
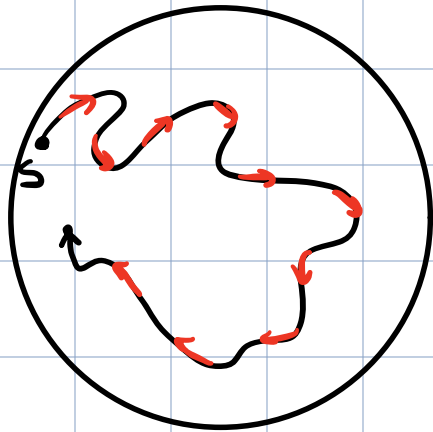
H_φ



The only negative edges are the "loopback" edges.

Mapping Paths

W some η -hop walk in $G \implies W'$ in H_φ using only $\frac{\eta}{h}$ hops!



loop once every h hops

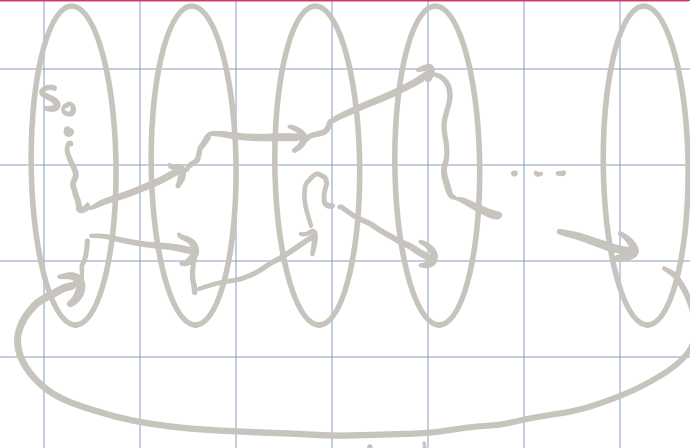
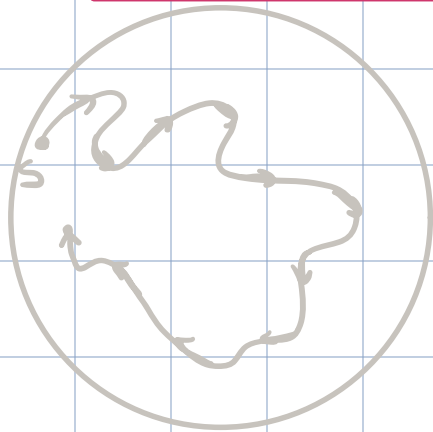
Hop Reducers

Shortest walks in G
take $\leq k$ hops, so
Shortest walks in H_ψ
take $\leq \frac{k}{h}$ hops

The only negative edges
are the "loopback" edges.

Mapping Paths

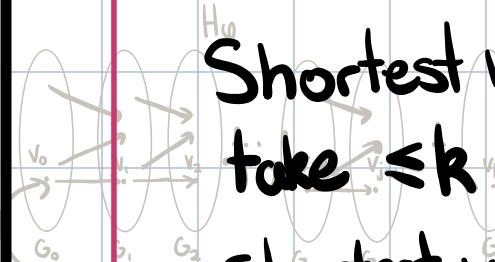
W some η -hop walk in $G \Rightarrow W'$ in H_ψ using only $\frac{\eta}{h}$ hops!



loop once every h hops

An Algorithm?

Hop Reducers



Shortest walks in G
take $\leq k$ hops, so
Shortest walks in H_ψ

The only negative edges
are the "loopback" edges.

Mapping Paths

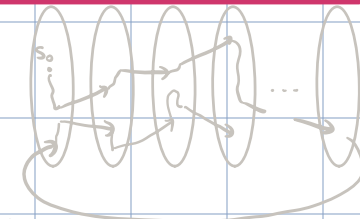
W some n -hop walk in G

take $\leq \frac{n}{h}$ hops

$\Rightarrow W$ in H_ψ using only $\frac{n}{h}$ hops!



$\Rightarrow$

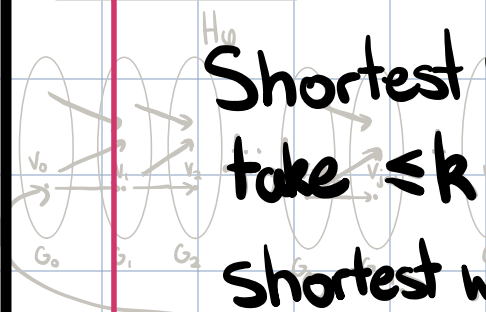


loop once every h hops

An Algorithm?

1. Pick h

Hop Reducers



Shortest walks in G
take $\leq k$ hops, so
Shortest walks in H_ψ
take $\leq \frac{k}{h}$ hops

The only negative edges
are the "loopback" edges.

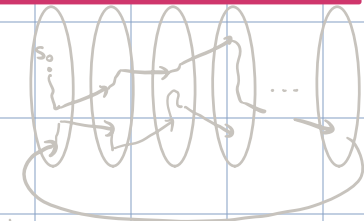
Mapping Paths

W some n -hop walk in G

$\Rightarrow W$ in H_ψ using only $\frac{n}{h}$ hops!



$\Rightarrow$

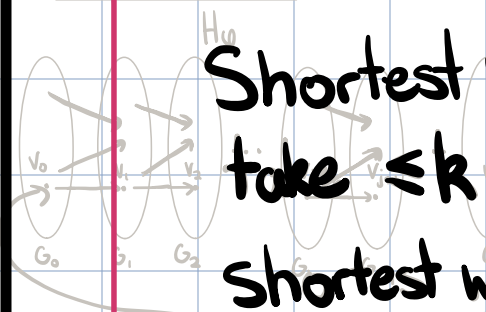


loop once every h hops

An Algorithm?

1. Pick h
2. Construct H

Hop Reducers



Shortest walks in G
take $\leq k$ hops, so
Shortest walks in H_ψ
take $\leq \frac{k}{h}$ hops

The only negative edges
are the "loopback" edges.

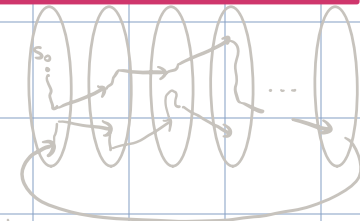
Mapping Paths

W some n -hop walk in G

$\Rightarrow W$ in H_ψ using only $\frac{n}{h}$ hops!



$\Rightarrow$

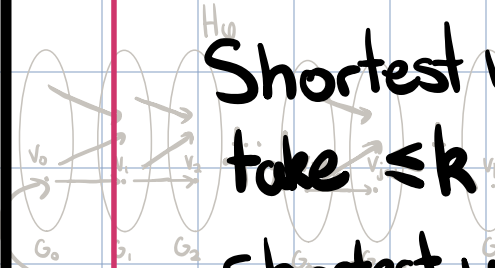


loop once every h hops

An Algorithm?

1. Pick h
2. Construct H
3. Compute φ & reweight H

Hop Reducers



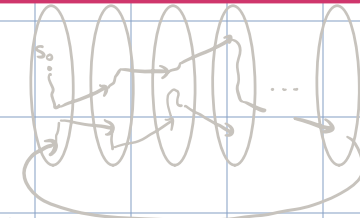
The only negative edges are the "loopback" edges.

Mapping Paths

W some n -hop walk in G



$\Rightarrow$

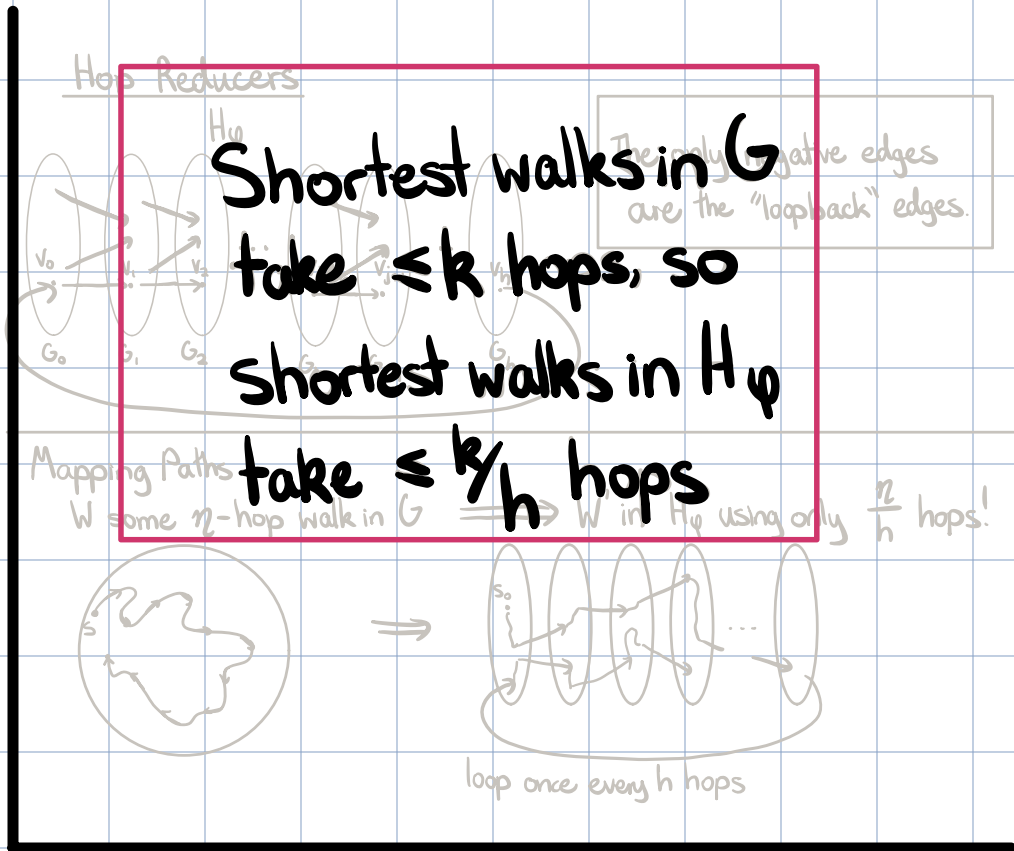


loop once every h hops

W in H_φ using only $\frac{n}{h}$ hops!

An Algorithm?

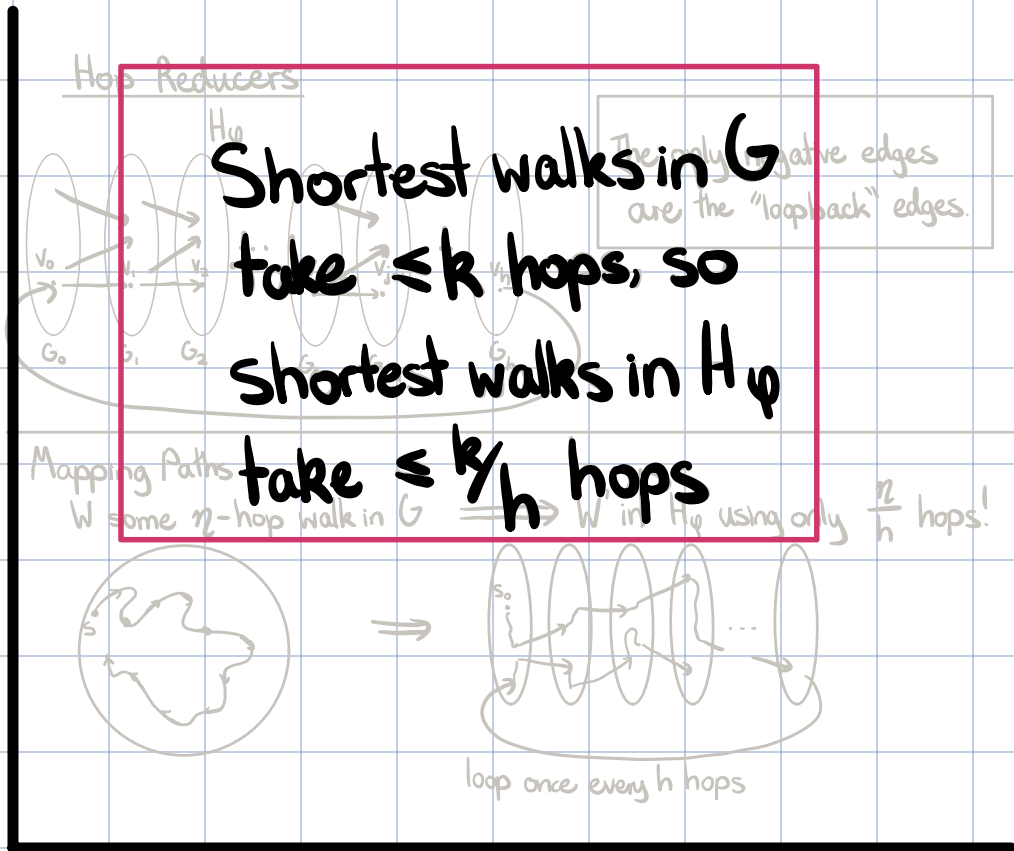
1. Pick h
2. Construct H
3. Compute φ & reweight H
4. Find distances in G with k/h -hop distances in H



An Algorithm?

1. Pick h
2. Construct H
3. Compute φ & reweight H
4. Find distances in G with k/h -hop distances in H

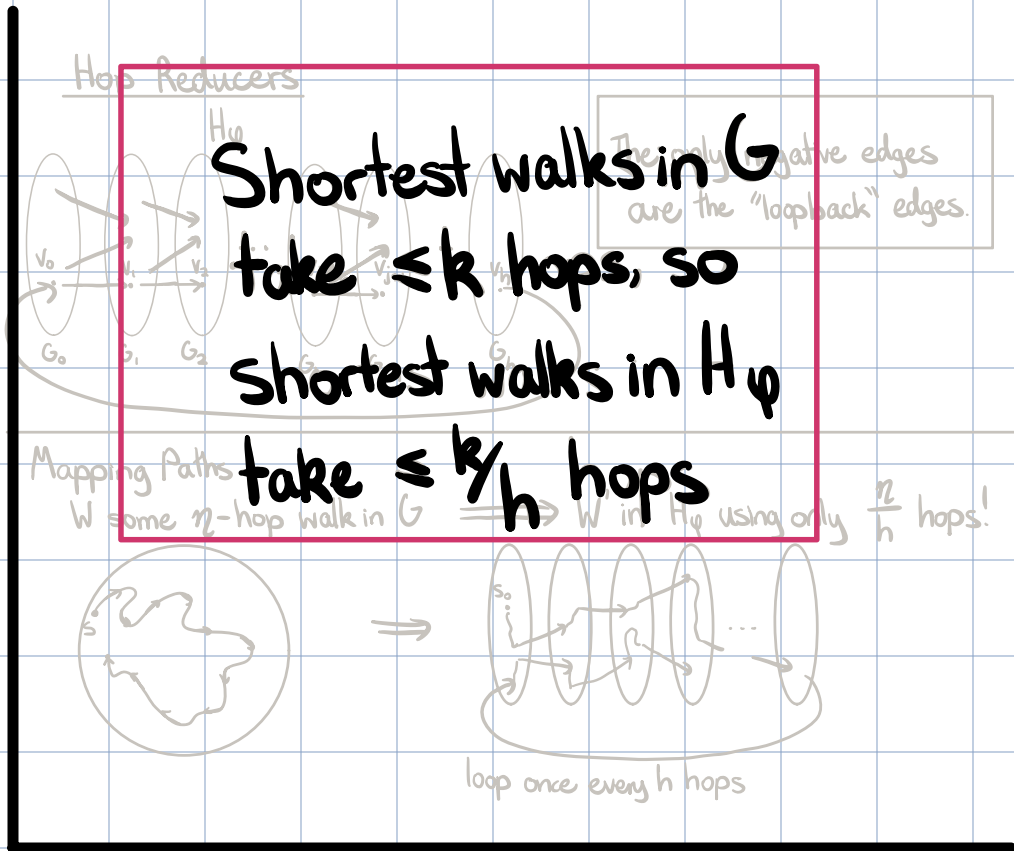
Running Time:



An Algorithm?

1. Pick h
2. Construct H
3. Compute φ & reweight H
4. Find distances in G with k/h -hop distances in H

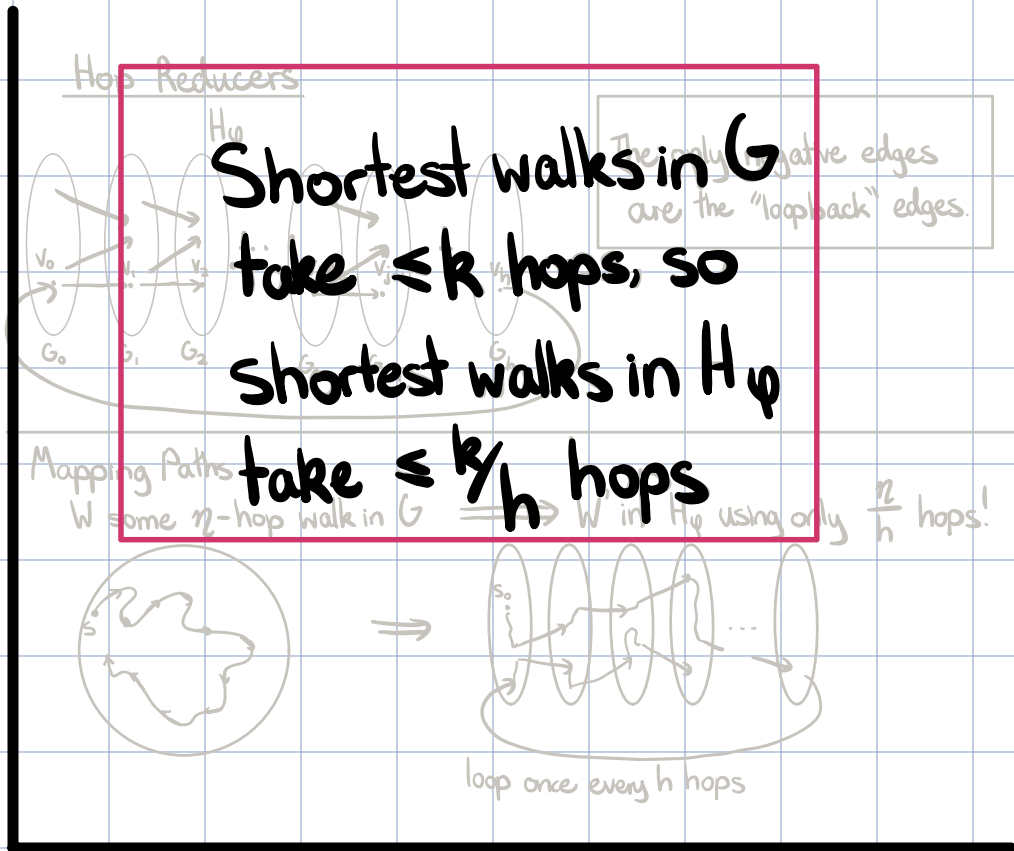
Running Time: $\tilde{O}(\frac{k}{h} |H|)$



An Algorithm?

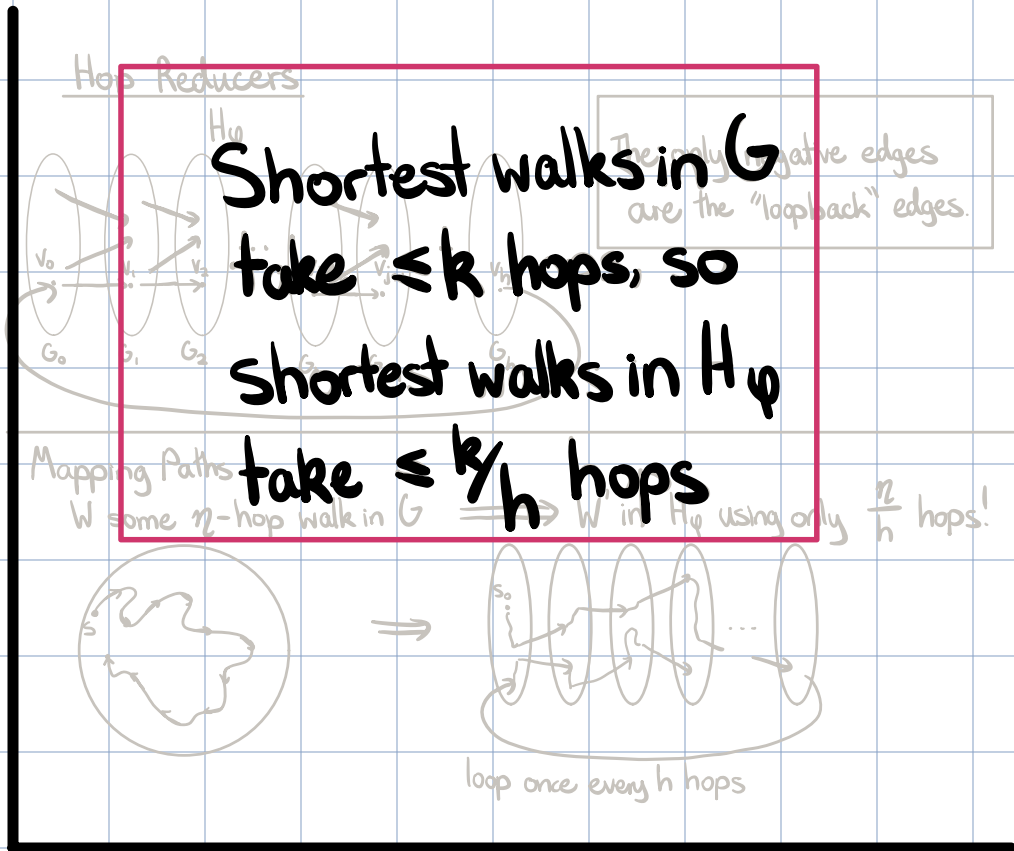
1. Pick h
2. Construct H
3. Compute φ & reweight H
4. Find distances in G with k/h -hop distances in H

Running Time: $\tilde{O}\left(\frac{k}{h} |H|\right) = \tilde{O}\left(\frac{k}{h} (hm)\right)$



An Algorithm?

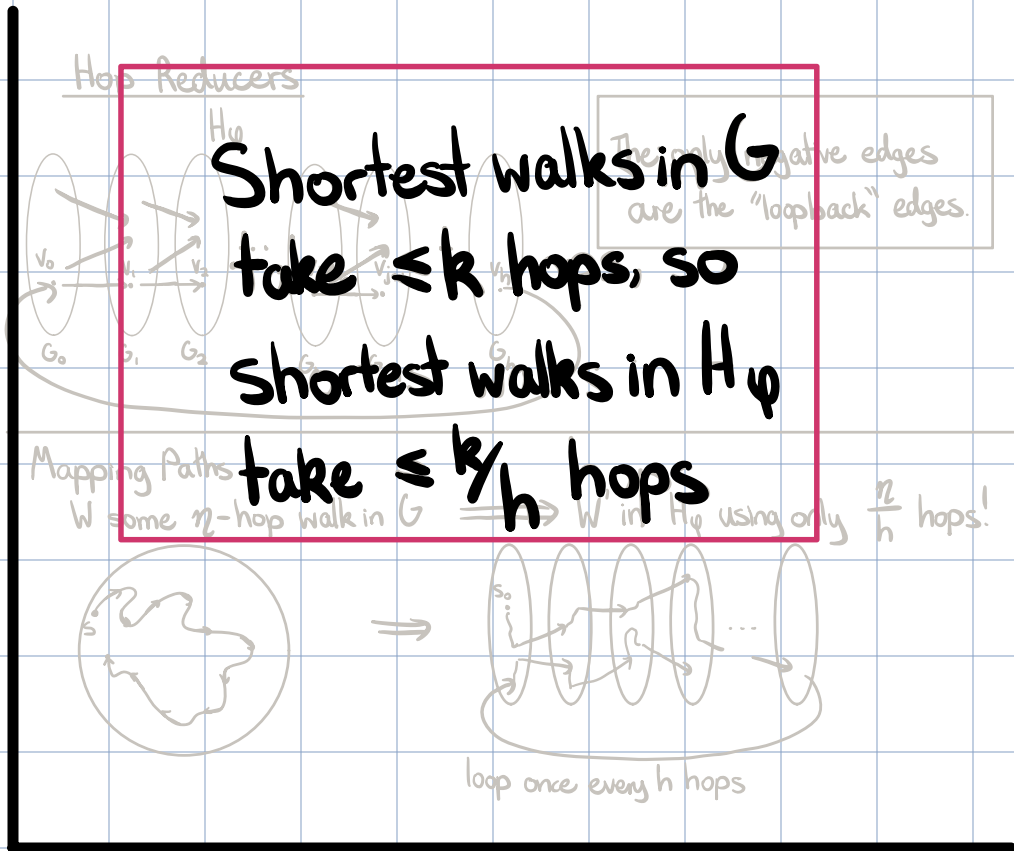
1. Pick h
2. Construct H
3. Compute φ & reweight H
4. Find distances in G with k/h -hop distances in H



$$\text{Running Time: } \tilde{O}\left(\frac{k}{h} |H|\right) = \tilde{O}\left(\frac{k}{h} (hm)\right) = \tilde{O}(km)$$

An Algorithm?

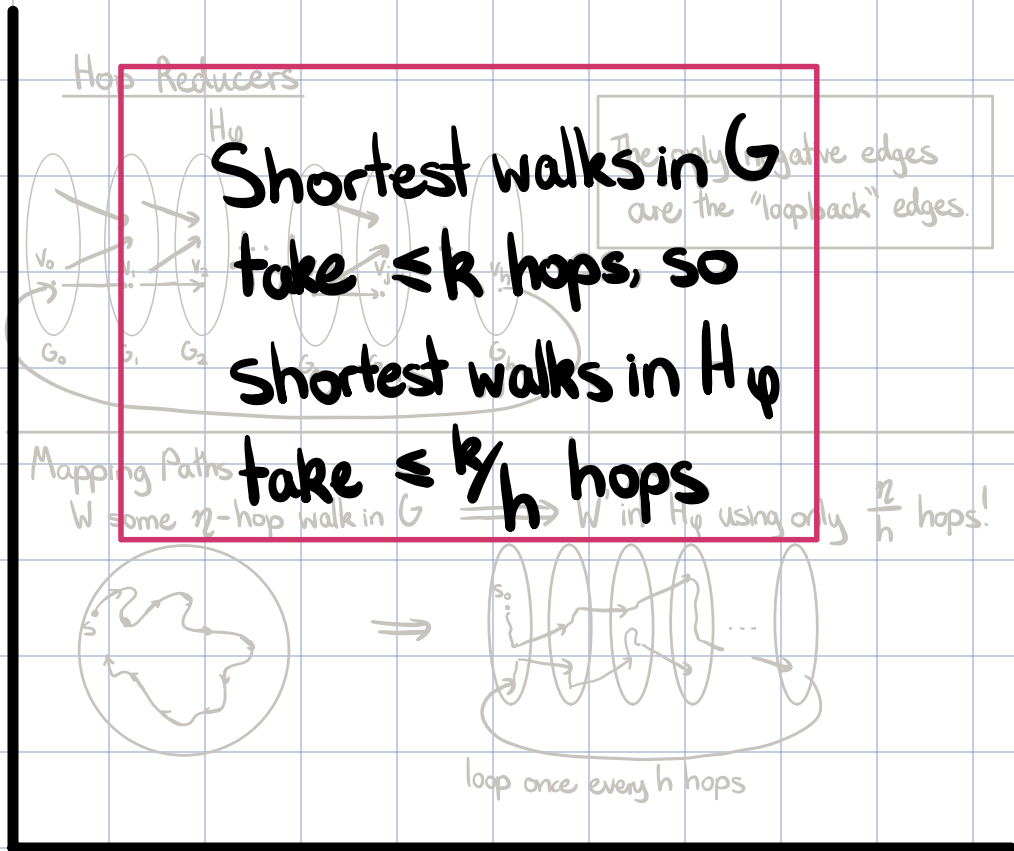
1. Pick h
2. Construct H
3. Compute φ & reweight H
4. Find distances in G with k/h -hop distances in H



Running Time: $\tilde{O}\left(\frac{k}{h} |H|\right) = \tilde{O}\left(\frac{k}{h} (hm)\right) = \tilde{O}(km)$ 😞

An Algorithm?

1. Pick h
2. Construct H
3. Compute φ & reweight H
4. Find distances in G with k/h -hop distances in H



Running Time: $\tilde{O}\left(\frac{k}{h} |H|\right) = \tilde{O}\left(\frac{k}{h} (hm)\right) = \tilde{O}(km)$ 😞

What went wrong? Hops $\downarrow$ by h but graph $\uparrow$ by h .

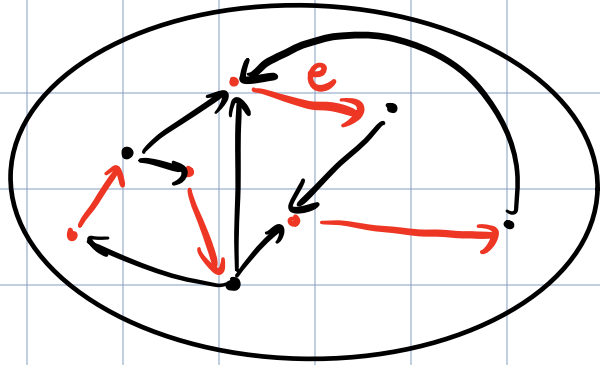
Incremental Elimination

Idea: Neutralize one edge at a time

Incremental Elimination

Idea: Neutralize one edge at a time

G

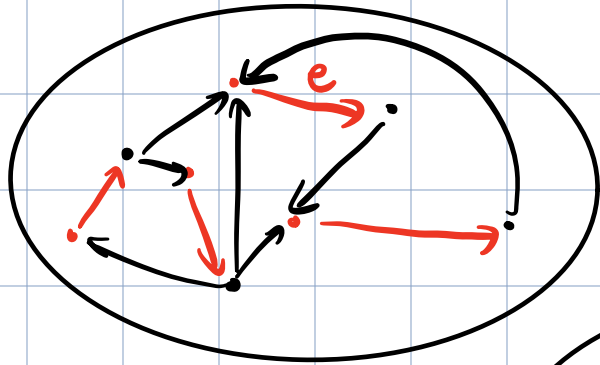


Pick neg
edge e

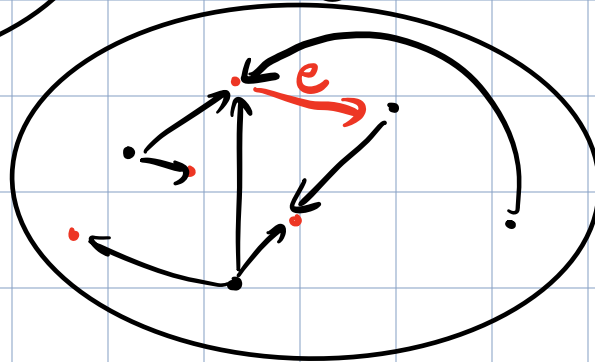
Incremental Elimination

Idea: Neutralize one edge at a time

G



G_e



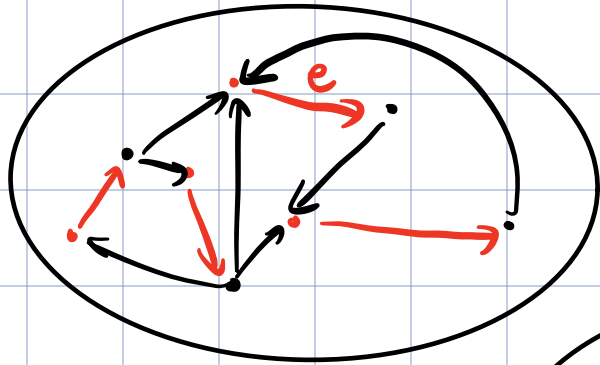
Pick neg
edge e

Remove other
neg. edges

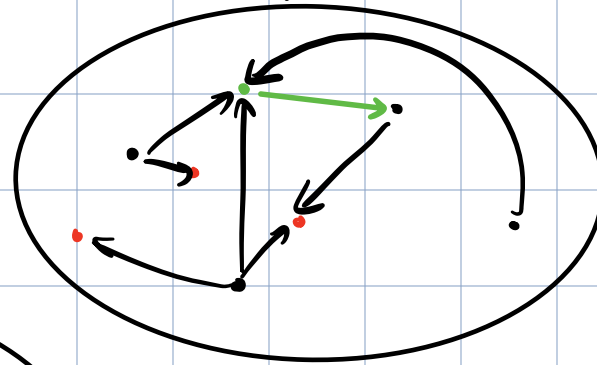
Incremental Elimination

Idea: Neutralize one edge at a time

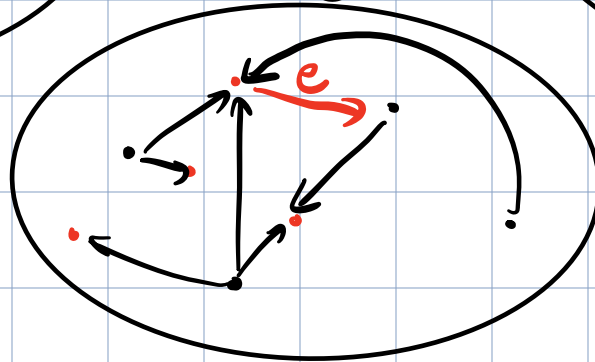
G



$G_{e,\varphi}$



G_e



Pick neg
edge e

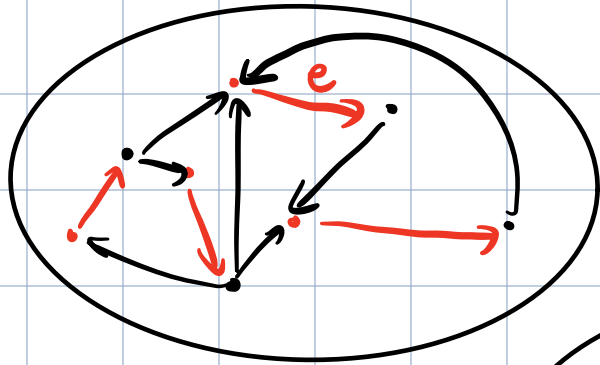
Remove other
neg. edges

Neutralize G_e
with $d'(s, \cdot)$

Incremental Elimination

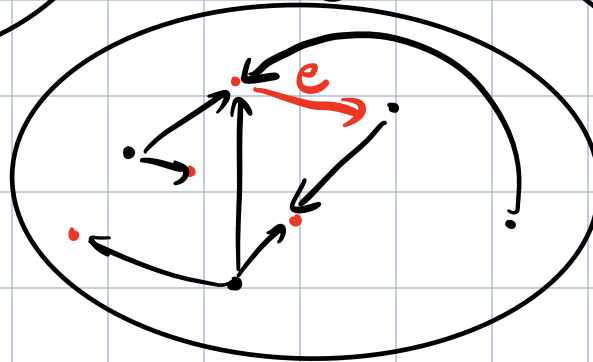
Idea: Neutralize one edge at a time

G



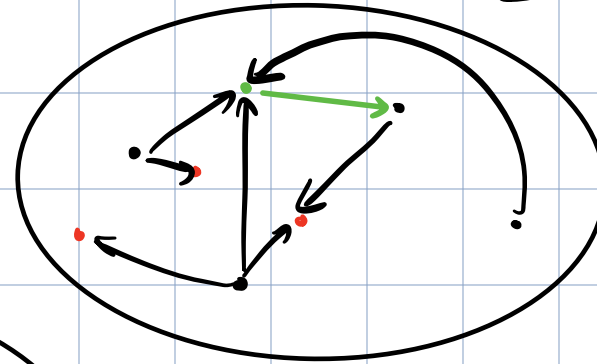
Pick neg
edge e

G_e



Remove other
neg. edges

$G_{e,\psi}$



Neutralize G_e
with $d'(s, \cdot)$

Neutralizing G
Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,

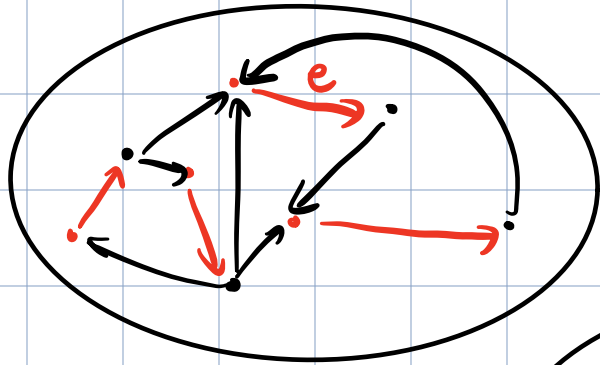
$$d(u, v) = d(u, s) + d(s, v) - \psi(v) = d(u, s) + d(s, v) - d(s, v) \geq 0$$
 Distance potentials make all edges nonnegative
 $\psi = d(s, \cdot)$

How to pick s ? $d(s^*, v) = d(s, v) = \min_{u \in V} d(u, v)$

Incremental Elimination

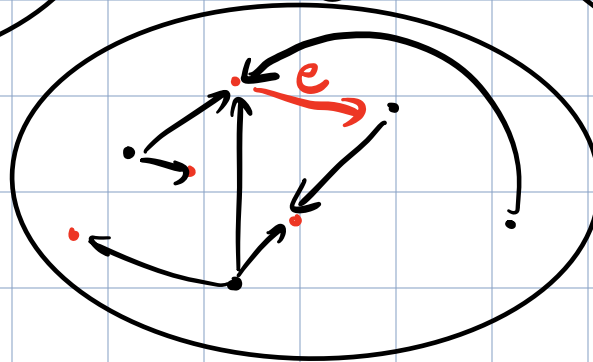
Idea: Neutralize one edge at a time

G



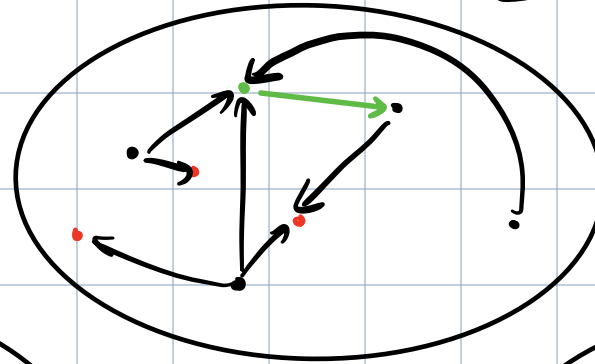
Pick neg
edge e

G_e



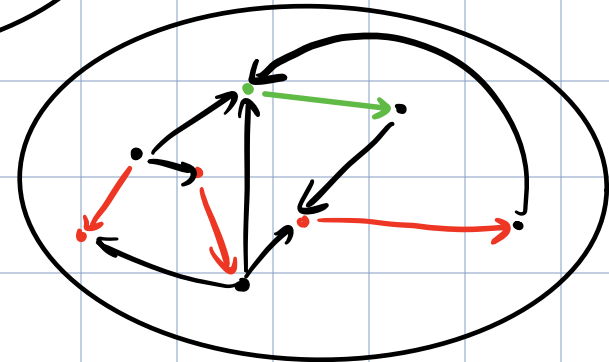
Remove other
neg. edges

$G_{e,\psi}$



Neutralize G_e
with $d'(s, \cdot)$

G_ψ



Add edges
back

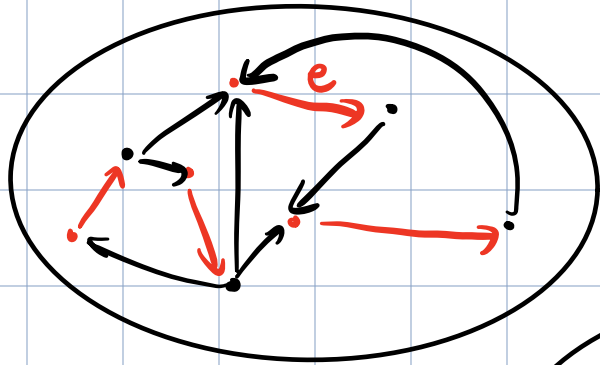
Neutralizing G
Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,
 $d'(u, v) = d(u, v) + \psi(u) - \psi(v) = d(u, v) + d(s, u) - d(s, v) \geq 0$
Distance potentials make
all edges nonnegative
 $\psi = d(s, \cdot)$

How to pick s ? $d(s^*, v) = d(s, v)$
 $= \min_{u \in V} d(u, v)$

Incremental Elimination

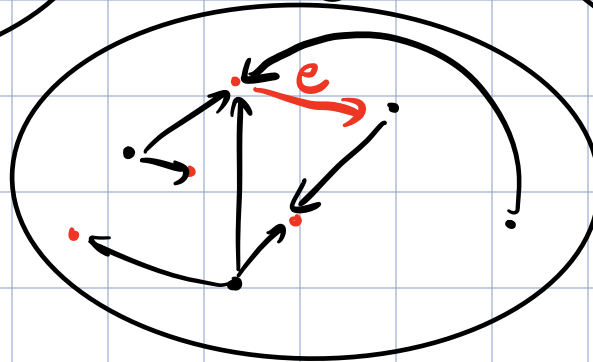
Idea: Neutralize one edge at a time

G



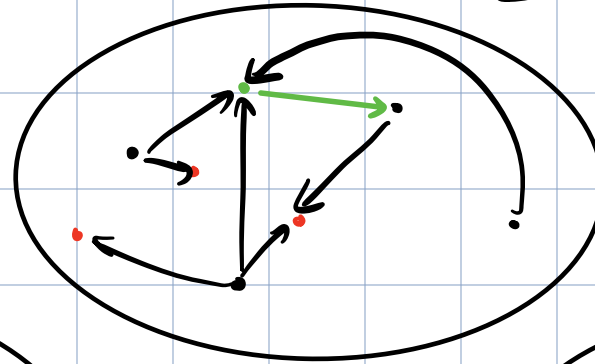
Pick neg
edge e

G_e



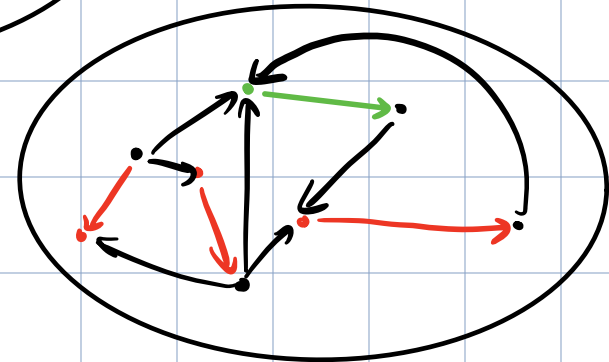
Remove other
neg. edges

$G_{e,\psi}$



Neutralize G_e
with $d'(s, \cdot)$

G_ψ



Add edges
back

Repeat until all edges are nonnegative

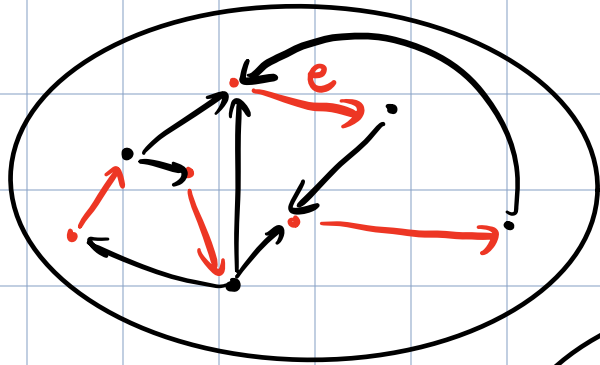
Neutralizing G
Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,
 $d(u, v) = d(u, s) + d(s, v) - d(s, u) = \psi(v) - \psi(u) + d(s, u) \geq 0$
Distance potentials make
all edges nonnegative
 $\psi = d(s, \cdot)$

How to pick s ? $d(s^*, v) = d(v, v) = 0$
 $= \min_{u \in V} d(u, v)$

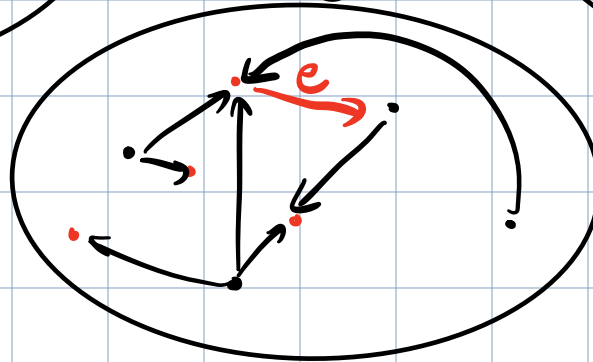
Incremental Elimination

Idea: Neutralize one edge at a time

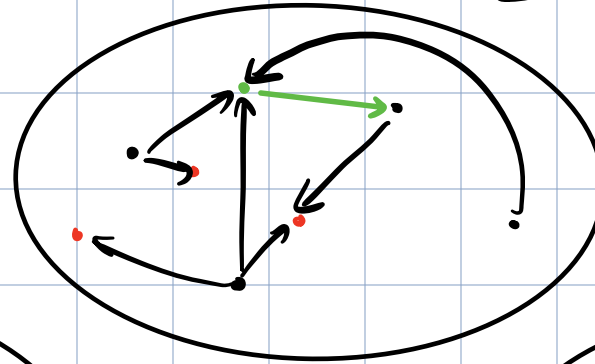
G



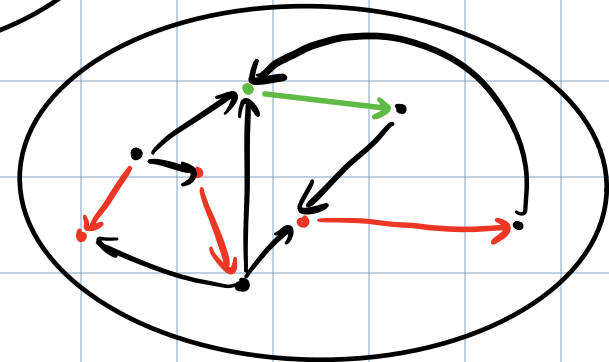
G_e



$G_{e,\psi}$



G_ψ



Pick neg
edge e

Remove other
neg. edges

Neutralize G_e
with $d'(s, \cdot)$

Add edges
back

Repeat until all edges are nonnegative

Running Time:

Neutralizing G
Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,
 $d'(u, v) = d(u, v) + \psi(u) - \psi(v) = d(u, v) + d(s, u) - d(s, v) \geq 0$
Distance potentials make
all edges nonnegative
 $\psi = d(s, \cdot)$

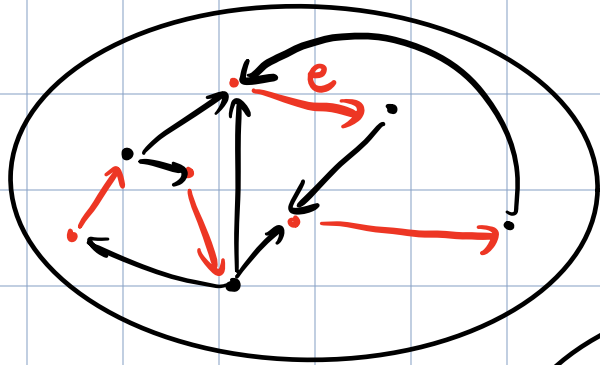
How to pick s ? $d(s^*, v) = d(s, v)$
 $= \min_{u \in V} d(u, v)$

A small diagram showing a graph with a source vertex s and its distance potentials to other vertices. Another small diagram shows a graph with a source vertex s and its distance potentials to other vertices.

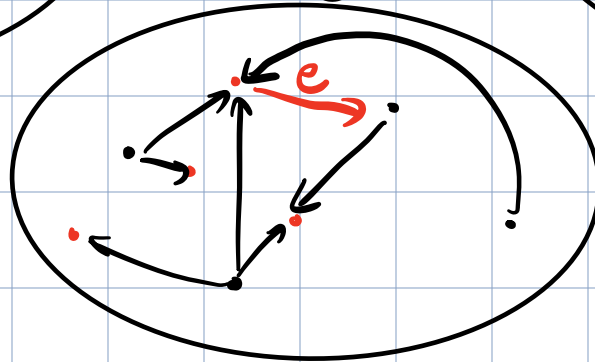
Incremental Elimination

Idea: Neutralize one edge at a time

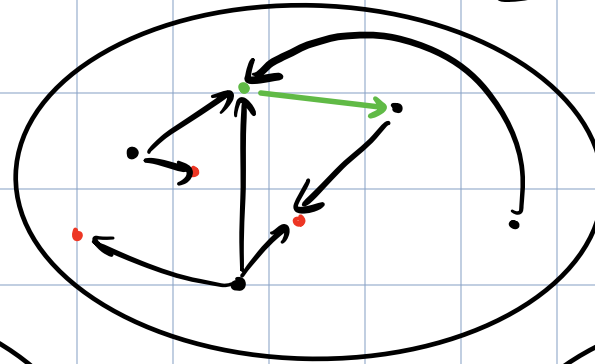
G



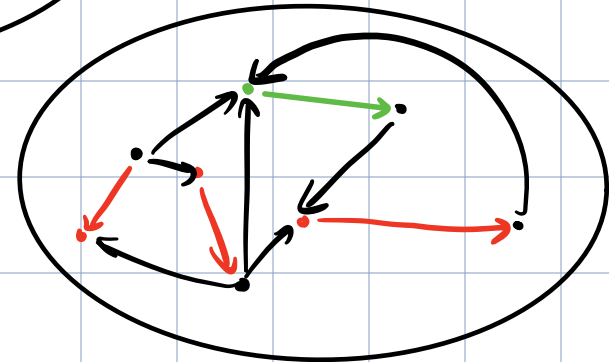
G_e



$G_{e,\psi}$



G_ψ



Pick neg
edge e

Remove other
neg. edges

Neutralize G_e
with $d'(s, \cdot)$

Add edges
back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m)$ per edge

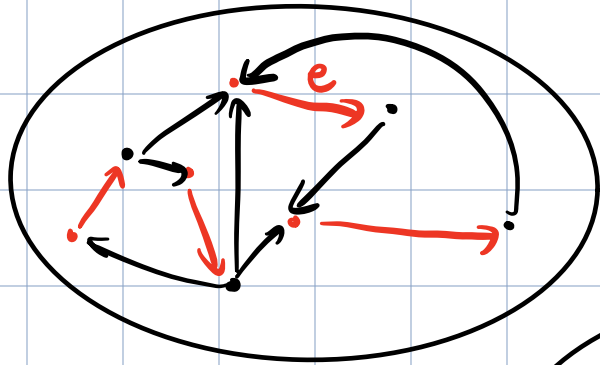
Neutralizing G
Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,
 $d(u, v) = d(u, s) + d(s, v) - d(s, u) = \psi(v) - \psi(u) + d(u, v) \geq 0$
Distance potentials make
all edges nonnegative
 $\psi = d(s, \cdot)$

How to pick s ?
 $d(s^*, v) = d(v, v)$
 $= \min_{u \in V} d(u, v)$

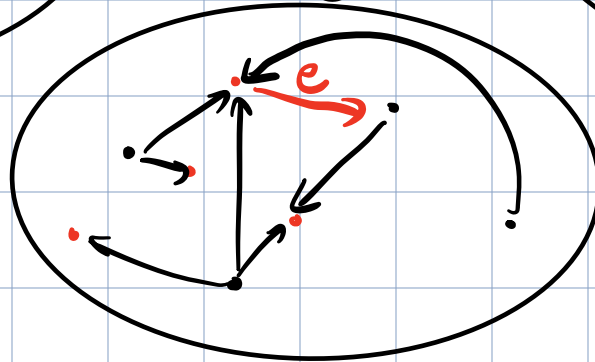
Incremental Elimination

Idea: Neutralize one edge at a time

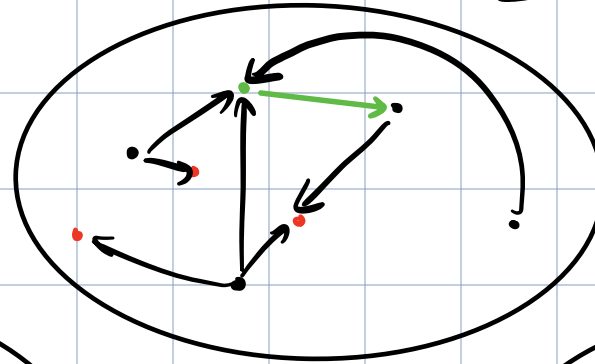
G



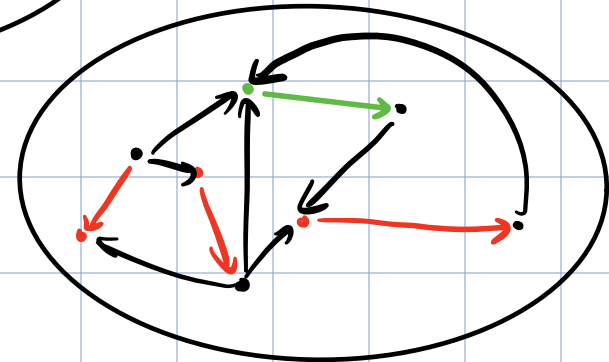
G_e



$G_{e,\psi}$



G_ψ



Pick neg
edge e

Remove other
neg. edges

Neutralize G_e
with $d'(s, \cdot)$

Add edges
back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m)$ per edge $\times k$ edges

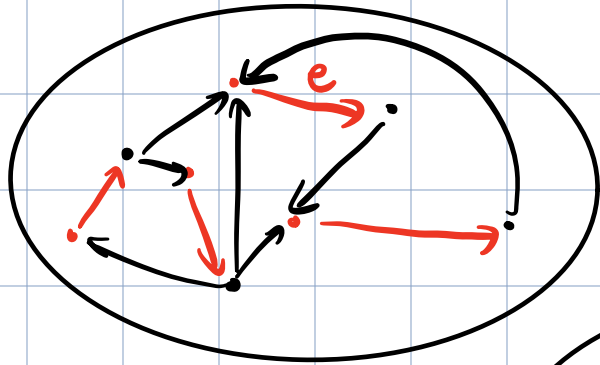
Neutralizing G
Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,
 $d(u, v) = d(u, s) + d(s, v) - d(s, u) = \psi(v) - \psi(u) + d(s, u) \geq 0$
Distance potentials make
all edges nonnegative
 $\psi = d(s, \cdot)$

How to pick s ? $d(s^*, v) = d(s, v)$
 $= \min_{u \in V} d(u, v)$

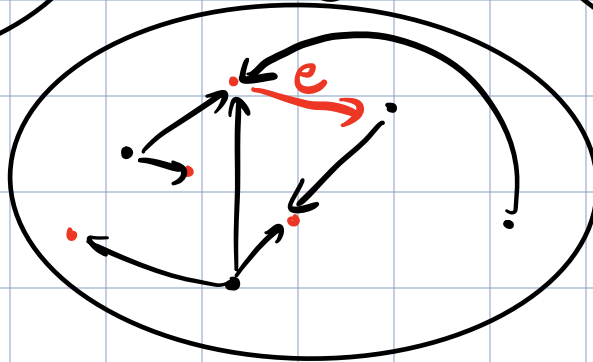
Incremental Elimination

Idea: Neutralize one edge at a time

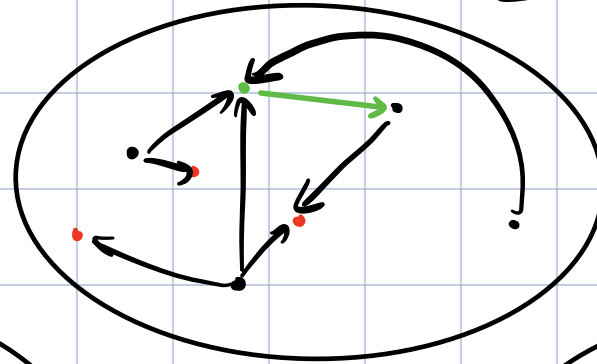
G



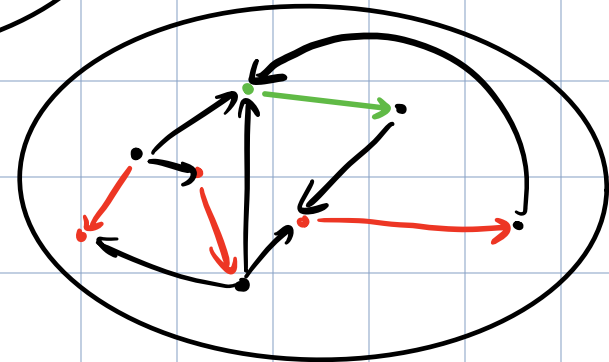
G_e



$G_{e,\psi}$



G_ψ



Pick neg
edge e

Remove other
neg. edges

Neutralize G_e
with $d'(s, \cdot)$

Add edges
back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m)$ per edge $\times k$ edges = $\tilde{O}(km)$

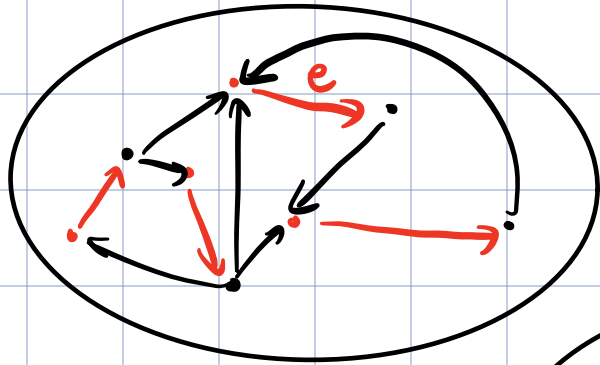
Neutralizing G
Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,
 $d(u, v) = d(u, s) + d(s, v) = \psi(u) + \psi(v) - d(s, u) \geq 0$
Distance potentials make
all edges nonnegative
 $\psi = d(s, \cdot)$

How to pick s ? $d(s^*, v) = d(v, v)$
 $= \min_{u \in V} d(u, v)$

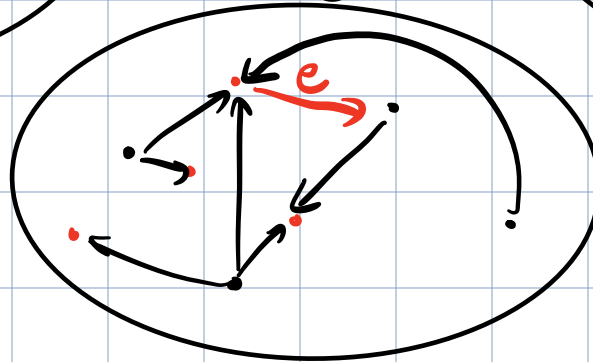
Incremental Elimination

Idea: Neutralize one edge at a time

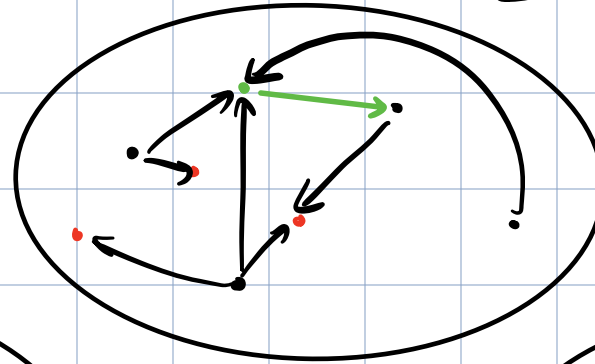
G



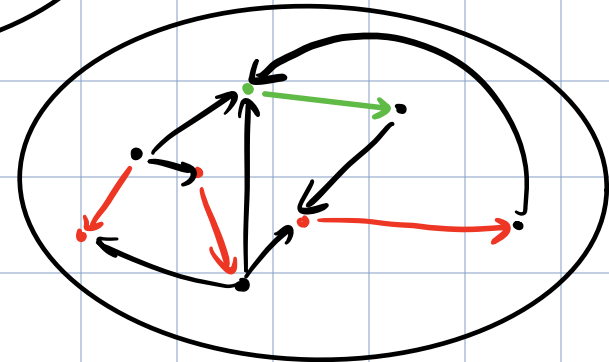
G_e



$G_{e,\psi}$



G_ψ



Pick neg
edge e

Remove other
neg. edges

Neutralize G_e
with $d'(s, \cdot)$

Add edges
back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m)$ per edge $\times k$ edges = $\tilde{O}(km)$



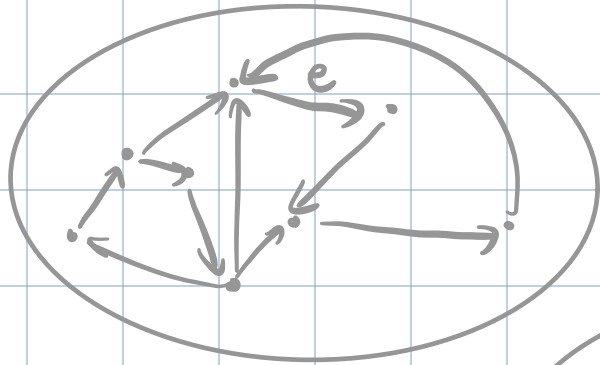
Neutralizing G
Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,
 $d(u, v) = \min_{u \rightarrow v} d(u, v) = \min_{u \rightarrow v} (d(u, s) + d(s, v)) \geq \psi(v) - \psi(u) \geq 0$
Distance potentials make all edges nonnegative
 $\psi = d(s, \cdot)$

How to pick s ? $d(s^*, v) = d(s, v)$
 $= \min_{u \in V} d(u, v)$

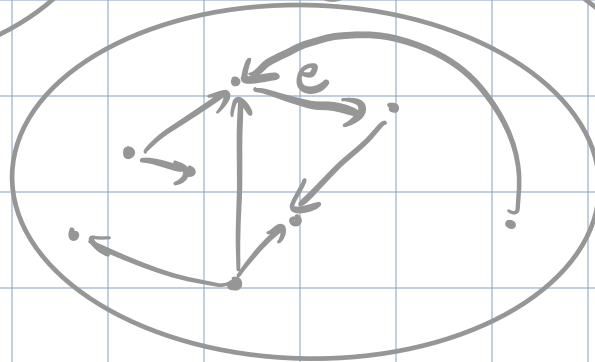
Incremental Elimination

Idea: Neutralize one edge at a time

G



G_e



Pick neg
edge e

Remove other
neg. edges

Neutralize G_e
with $d'(s, \cdot)$

Add edges
back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m)$ per edge $\times k$ edges = $\tilde{O}(km)$

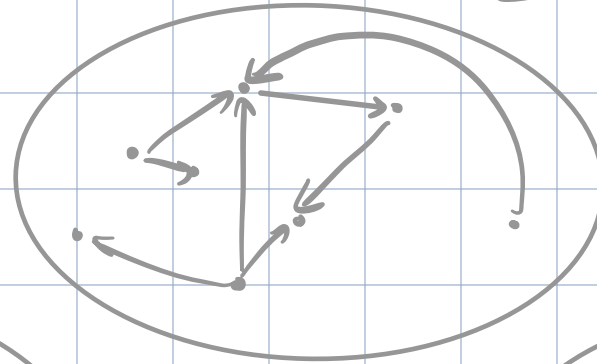


Neutralizing G
Suppose some vertex s can reach all $v \in V$. Let $\varphi(v) = d(s, v)$. Then,

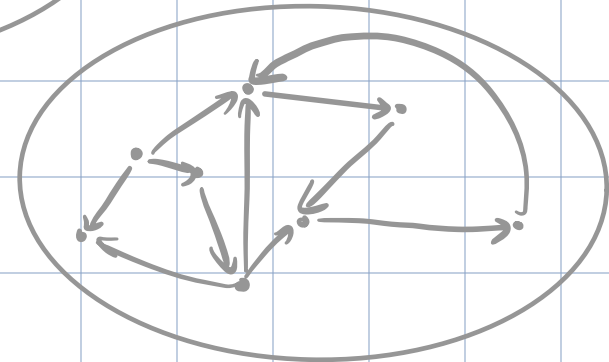
$$f_e(u, v) = f(u, v) + \varphi(u) - \varphi(v) = f(u, v) + d(s, u) - d(s, v) \geq 0$$
 Distance potentials make all edges nonnegative
 $\varphi = d(s, \cdot)$
 How to pick s ? $d(s^*, v) = d(v, v) = 0$
 $= \min_{u \in V} d(u, v)$

The diagram shows a graph with a source vertex s (indicated by a star) and other vertices. Arrows represent edges, and labels $d(s, v)$ indicate the distance from s to each vertex v . A box highlights the text 'Distance potentials make all edges nonnegative'.

$G_{e, \varphi}$



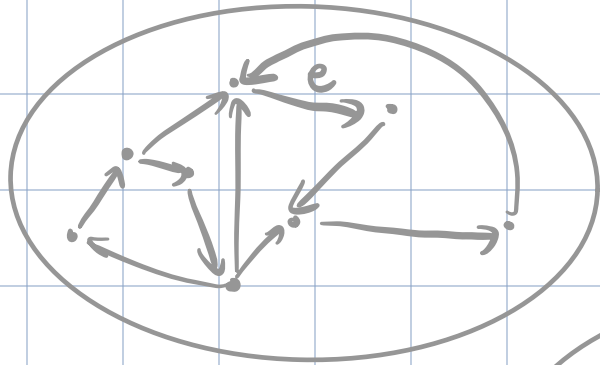
G_φ



Incremental Elimination

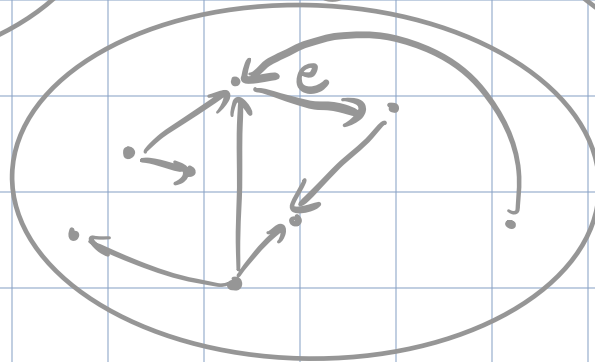
Idea: Neutralize one **edge set** at a time

G



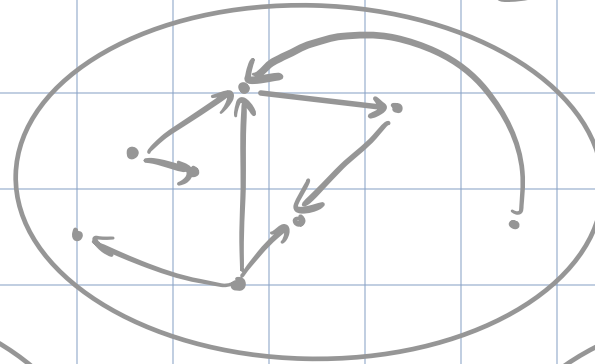
Pick neg
edge e

G_e



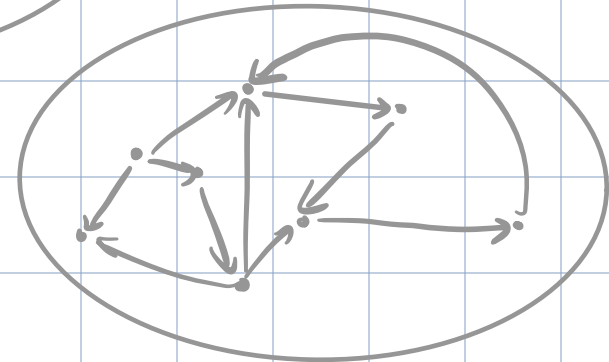
Remove other
neg. edges

$G_{e,\psi}$



Neutralize G_e
with $d'(s, \cdot)$

G_ψ



Add edges
back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m)$ per edge $\times k$ edges = $\tilde{O}(km)$

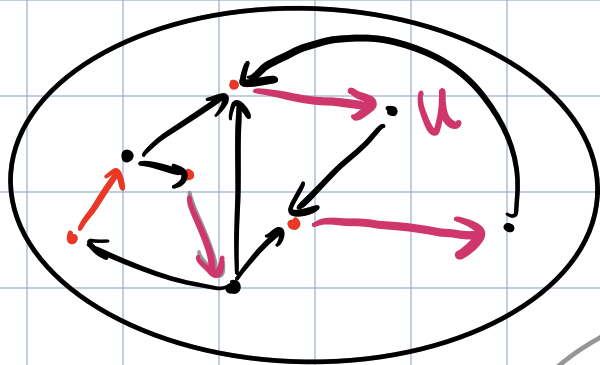


Neutralizing G
Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,
 $\ell_e(u, v) = \ell(u, v) + \psi(u) - \psi(v) = \ell(u, v) + d(s, u) - d(s, v) \geq 0$
Distance potentials make all edges nonnegative
 $\psi = d(s, \cdot)$
How to pick s ? $d(s^*, v) = d(v, v) = 0$
 $= \min_{u \in V} d(u, v)$

Incremental Elimination

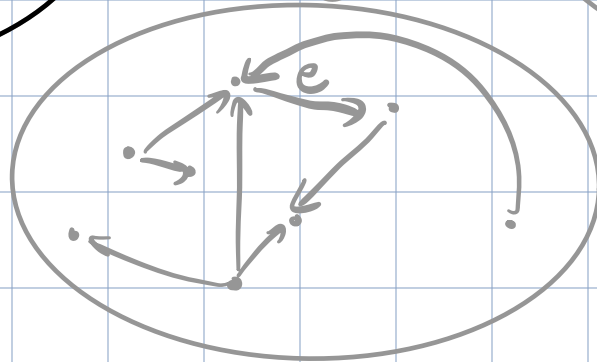
Idea: Neutralize one **edge set** at a time

G



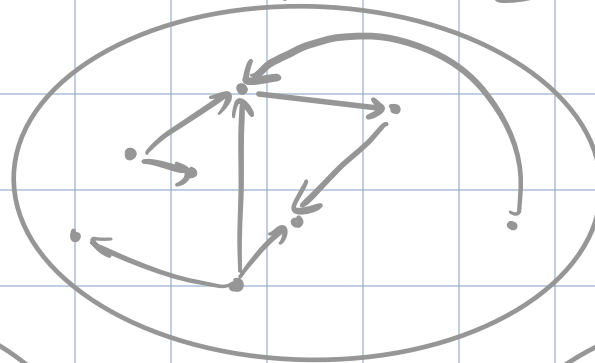
Pick neg
edge set **U**

G_e



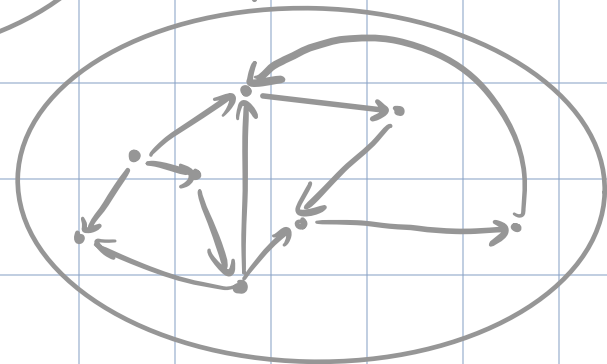
Remove other
neg. edges

G_{e,ψ}



Neutralize G_e
with d'(s, ·)

G_ψ



Add edges
back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m)$ per edge $\times k$ edges = $\tilde{O}(km)$

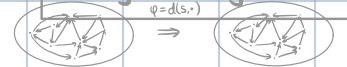


Neutralizing G

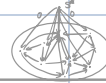
Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,

$$f_e(u, v) = f(u, v) + \psi(u) - \psi(v) = f(u, v) + d(s, u) - d(s, v) \geq 0$$

Distance potentials make
all edges nonnegative



How to pick s ?

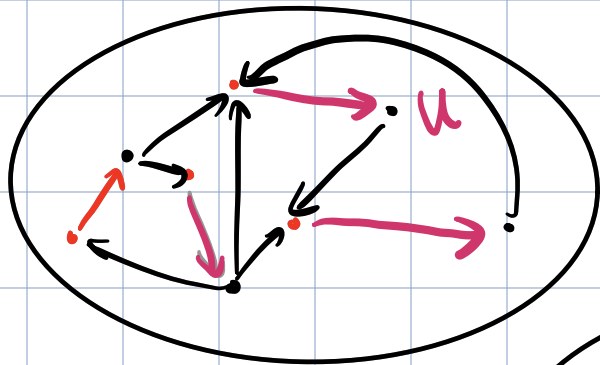


$$d(s^*, v) = d(v, v) = \min_{u \in V} d(u, v)$$

Incremental Elimination

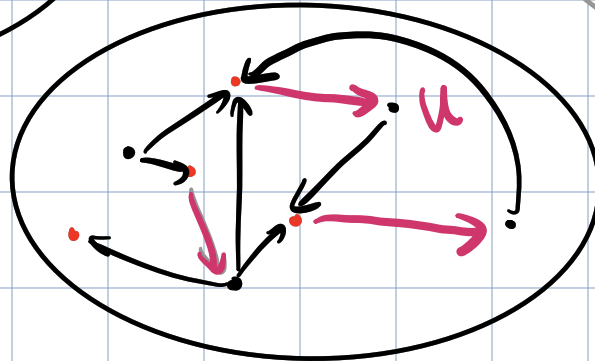
Idea: Neutralize one **edge set** at a time

G



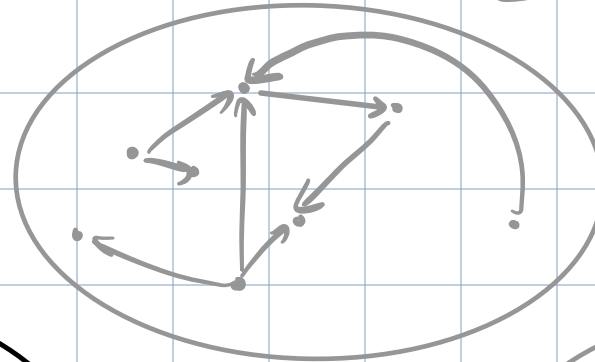
Pick neg
edge set **U**

G_U



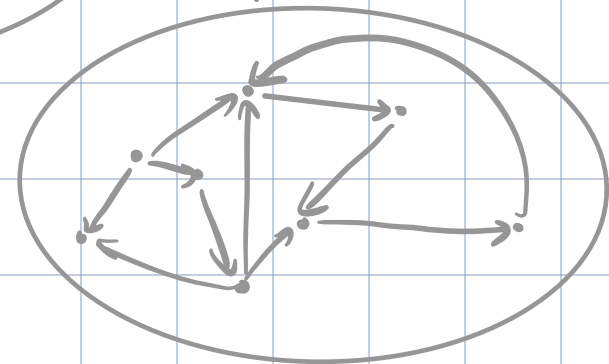
Remove other
neg. edges

G_{E,φ}



Neutralize G_E
with d'(s, ·)

G_φ



Add edges
back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m)$ per edge $\times k$ edges = $\tilde{O}(km)$



Neutralizing G

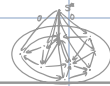
Suppose some vertex s can reach all $v \in V$. Let $\phi(v) = d(s, v)$. Then,

$$f_e(u, v) = f(u, v) + \phi(u) - \phi(v) = f(u, v) + d(s, u) - d(s, v) \geq 0$$

Distance potentials make
all edges nonnegative



How to pick s ?

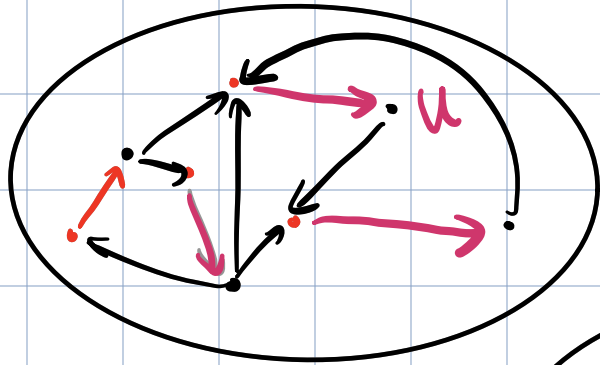


$$d(s^*, v) = d(s, v) = \min_{u \in V} d(u, v)$$

Incremental Elimination

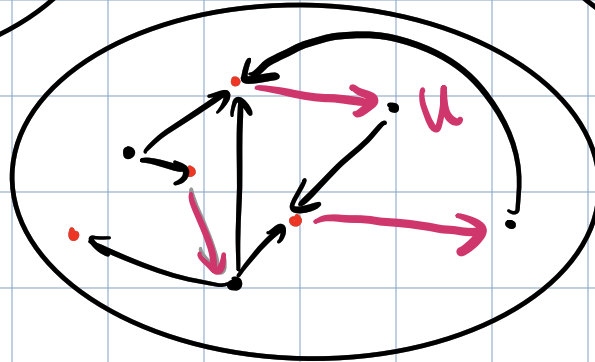
Idea: Neutralize one **edge set** at a time

G



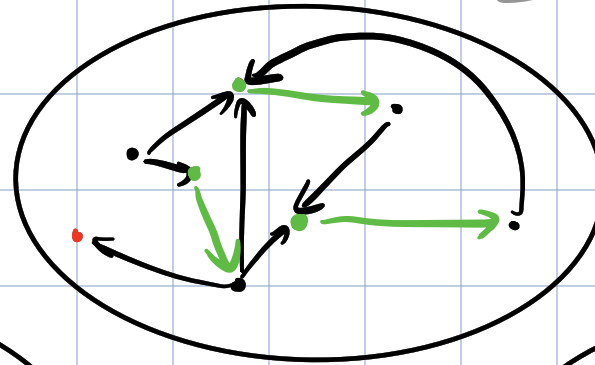
Pick neg
edge set **U**

G_U



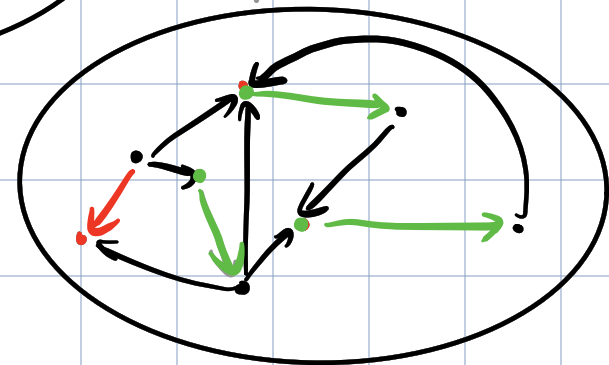
Remove other
neg. edges

G_{U,φ}



Neutralize G_U
with **d^{|U|}(s, ·)**

G_φ



Add edges
back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m)$ per edge $\times k$ edges = $\tilde{O}(km)$



Neutralizing G

Suppose some vertex s can reach all $v \in V$. Let $\phi(v) = d(s, v)$. Then,

$$f_e(u, v) = f(u, v) + \phi(u) - \phi(v) = f(u, v) + d(s, u) - d(s, v) \geq 0$$

Distance potentials make all edges nonnegative

$\phi = d(s, \cdot)$

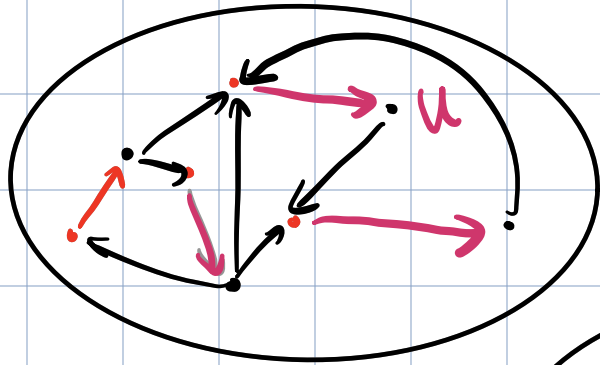
How to pick s ?

$d(s^*, v) = d(v, v)$
 $= \min_{u \in V} d(u, v)$

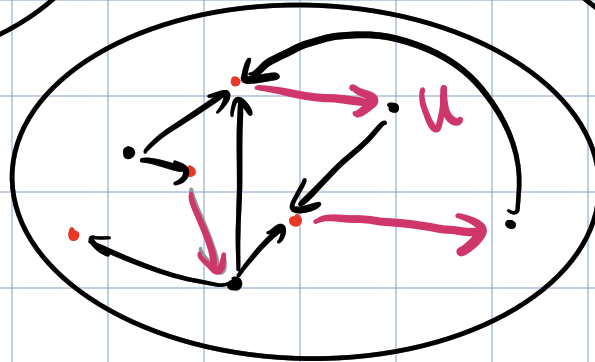
Incremental Elimination

Idea: Neutralize one **edge set** at a time

G



G_U



Pick neg
edge set **U**

Remove other
neg. edges

Neutralize G_U
with **$d^{|U|}(s, \cdot)$**

Add edges
back

Repeat until all edges are nonnegative

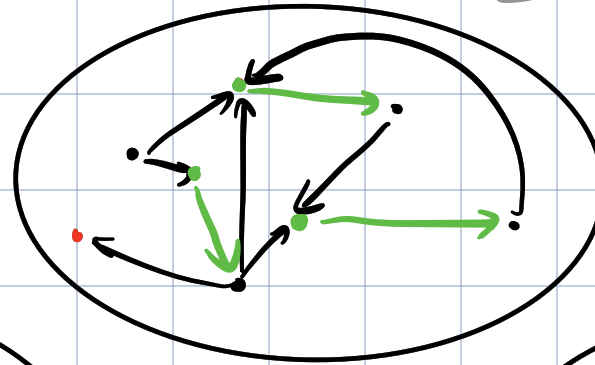
Running Time: **$\tilde{O}(m|U|)$** per edge set $\times \frac{k}{|U|}$ edges = **$\tilde{O}(km)$**

Neutralizing G
Suppose some vertex s can reach all $v \in V$. Let $\varphi(v) = d(s, v)$. Then,
 $\ell_g(u, v) = \ell(u, v) + \varphi(u) - \varphi(v) = \ell(u, v) + d(s, u) - d(s, v) \geq 0$
Distance potentials make
all edges nonnegative
 $\varphi = d(s, \cdot)$

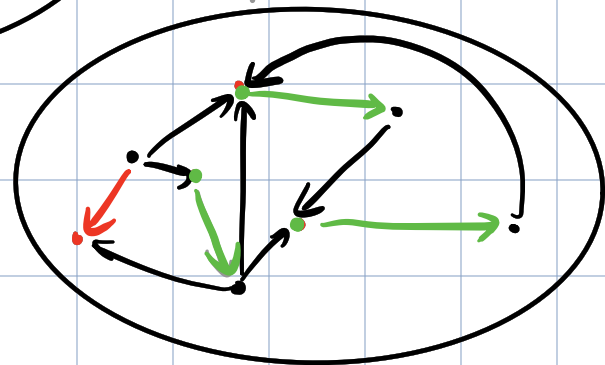
How to pick s ? $d(s^*, v) = d(v, v) = \min_{u \in V} d(u, v)$

The diagram shows a graph with a source vertex s (red dot) and other vertices. Green arrows represent the shortest paths from s to each vertex, illustrating the distance potential $\varphi(v) = d(s, v)$.

G_{U, φ}



G _{φ}

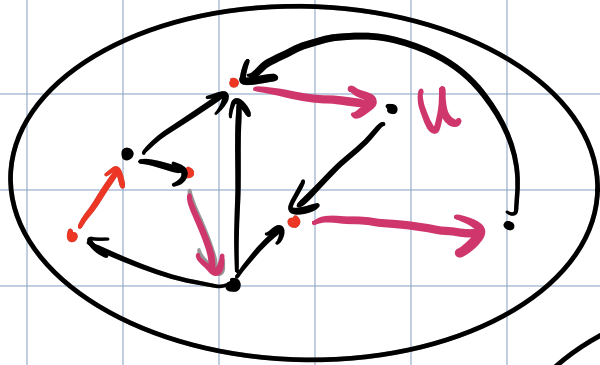


Incremental Elimination

Idea: Neutralize one edge set at a time

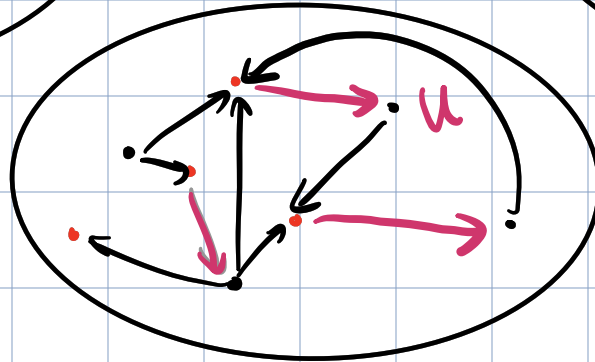
Neutralizing G
 Suppose some vertex s can reach all $v \in V$. Let $\varphi(v) = d(s, v)$. Then,
 $\ell_e(u, v) = \ell(u, v) + \varphi(u) - \varphi(v) = \ell(u, v) + d(s, u) - d(s, v) \geq 0$
 Distance potentials make all edges nonnegative
 $\varphi = d(s, \cdot)$
 How to pick s ? $d(s^*, v) = d(v, v) = \min_{u \in V} d(u, v)$

G



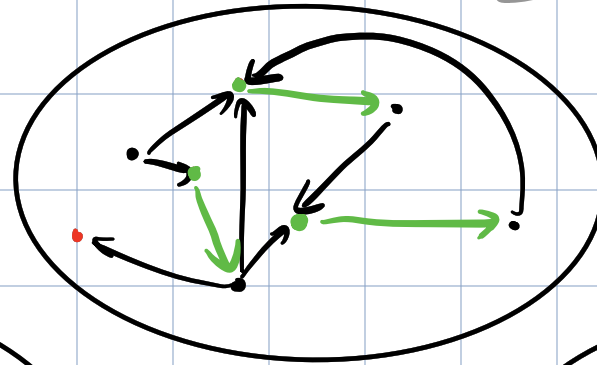
Pick neg edge set U

G_U



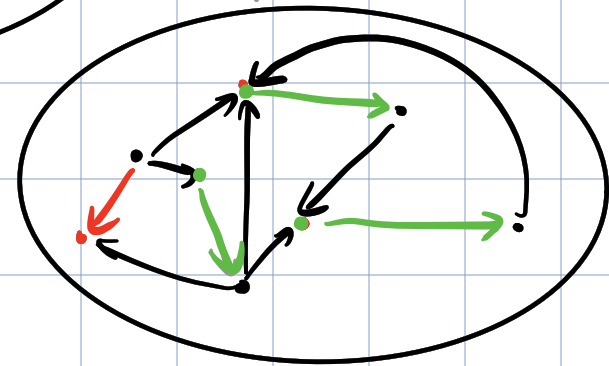
Remove other neg. edges

$G_{U, \varphi}$



Neutralize G_U with $d^{|U|}(s, \cdot)$

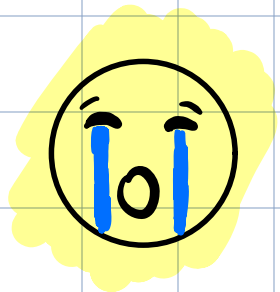
G_φ



Add edges back

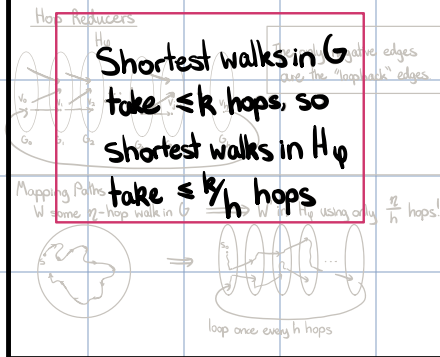
Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m|U|)$ per edge set $\times \frac{k}{|U|}$ edges = $\tilde{O}(km)$



An Algorithm?

1. Pick h
2. Construct H
3. Compute φ & reweight H
4. Find distances in G with k_h -hop distances in H

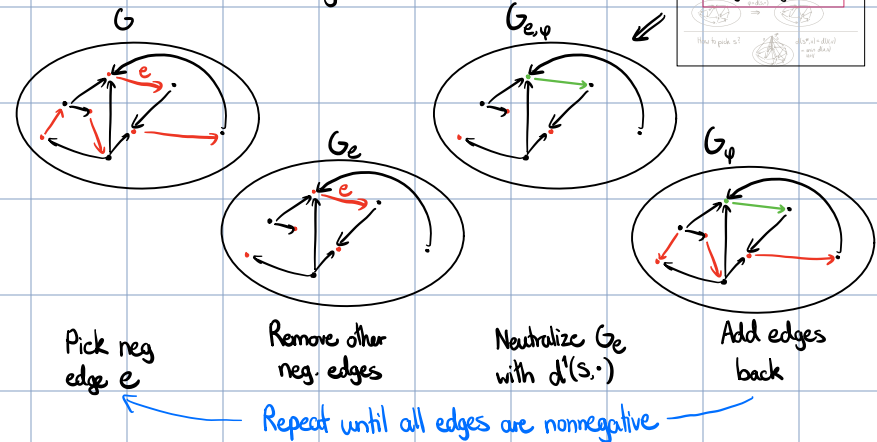


Running Time: $\tilde{O}(\frac{k}{h} |H|) = \tilde{O}(\frac{k}{h} (hm)) = \tilde{O}(km)$ ☹️

What went wrong? Hops $\downarrow$ by h but graph $\uparrow$ by h .

Incremental Elimination

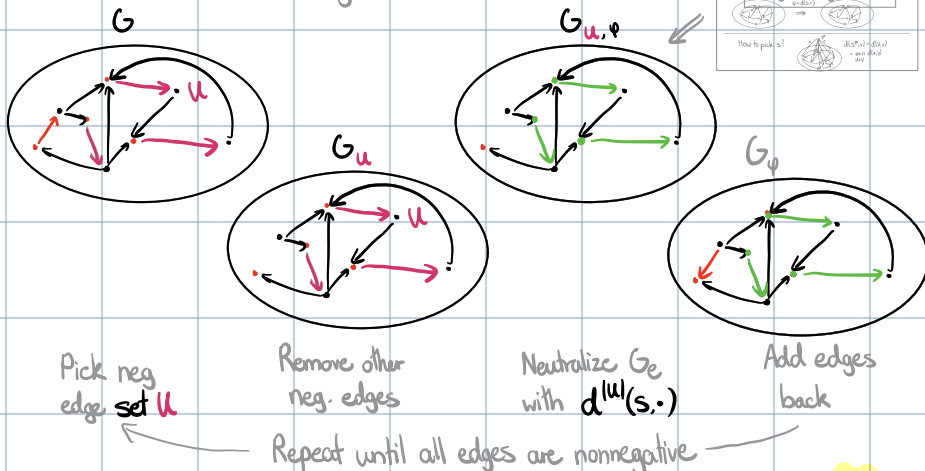
Idea: Neutralize one edge at a time



Running Time: $\tilde{O}(m)$ per edge $\times k$ edges = $\tilde{O}(km)$ ☹️

Incremental Elimination

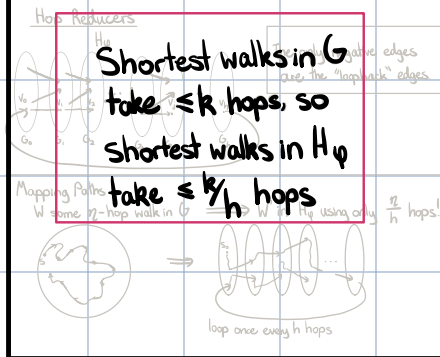
Idea: Neutralize one edge set at a time



Running Time: $\tilde{O}(m|U|)$ per edge set $\times \frac{k}{|U|}$ edges = $\tilde{O}(km)$ 😊

An Algorithm?

1. Pick h
2. Construct H
3. Compute φ & reweight H
4. Find distances in G with k_h -hop distances in H

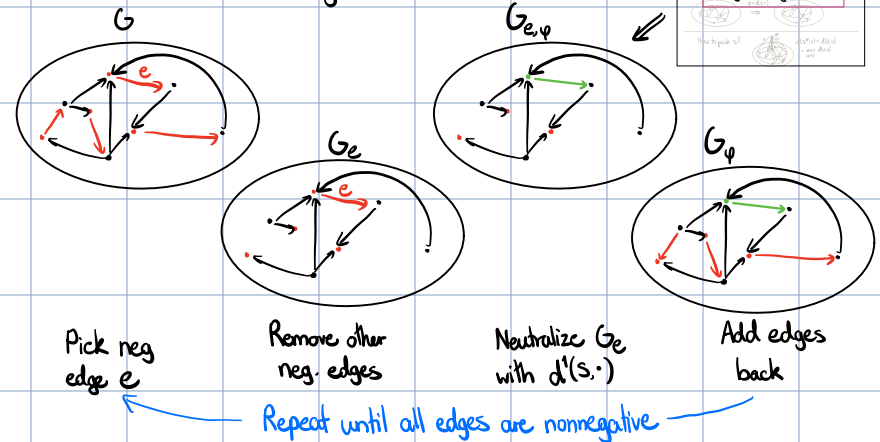


Running Time: $\tilde{O}(\frac{k}{h} |H|) = \tilde{O}(\frac{k}{h} (hm)) = \tilde{O}(km)$ ☹️

What went wrong? Hops $\downarrow$ by h but graph $\uparrow$ by h .

Incremental Elimination

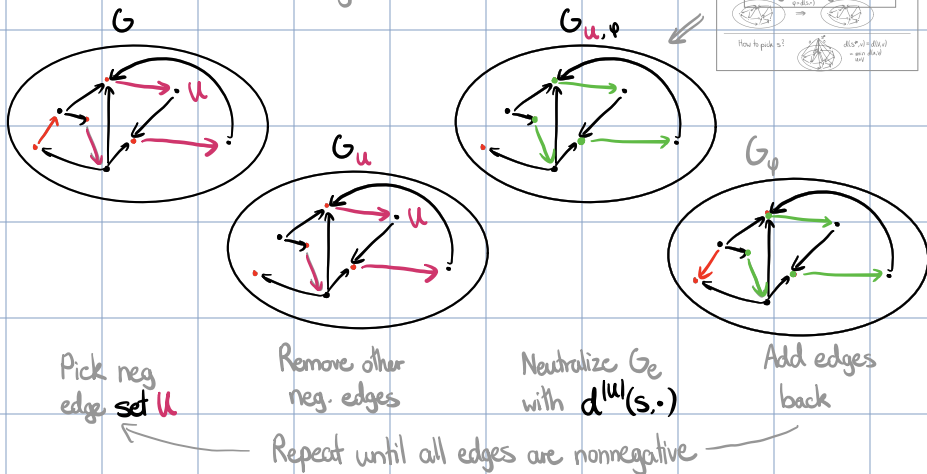
Idea: Neutralize one edge at a time



Running Time: $\tilde{O}(m)$ per edge $\times k$ edges = $\tilde{O}(km)$ ☹️

Incremental Elimination

Idea: Neutralize one edge set at a time



Running Time: $\tilde{O}(m|U|)$ per edge set $\times \frac{k}{|U|}$ edges = $\tilde{O}(km)$ 😊

Overarching Problem:
Spending $\tilde{O}(m)$ time on avg
per negative edge

End Goal: Spend $\tilde{O}(\frac{m}{h})$ time per negative edge.

End Goal: Spend $\tilde{O}(\frac{m}{h})$ time per negative edge.

But how?

End Goal: Spend $\tilde{O}(\frac{m}{h})$ time per negative edge.

But how?



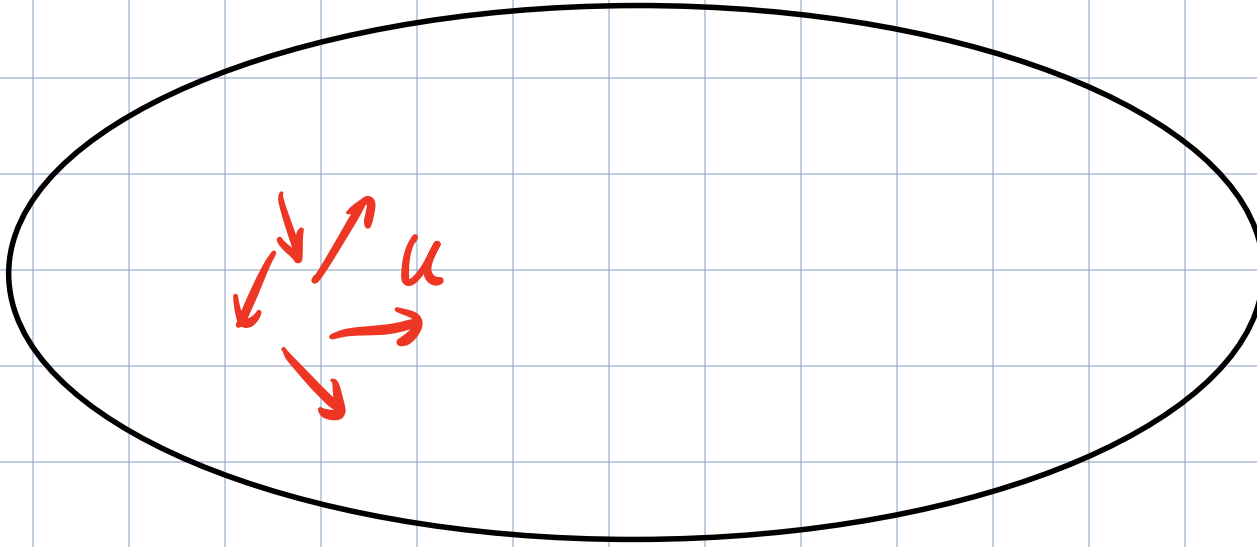
End Goal: Spend $\tilde{O}(\frac{m}{h})$ time per negative edge.

But how?

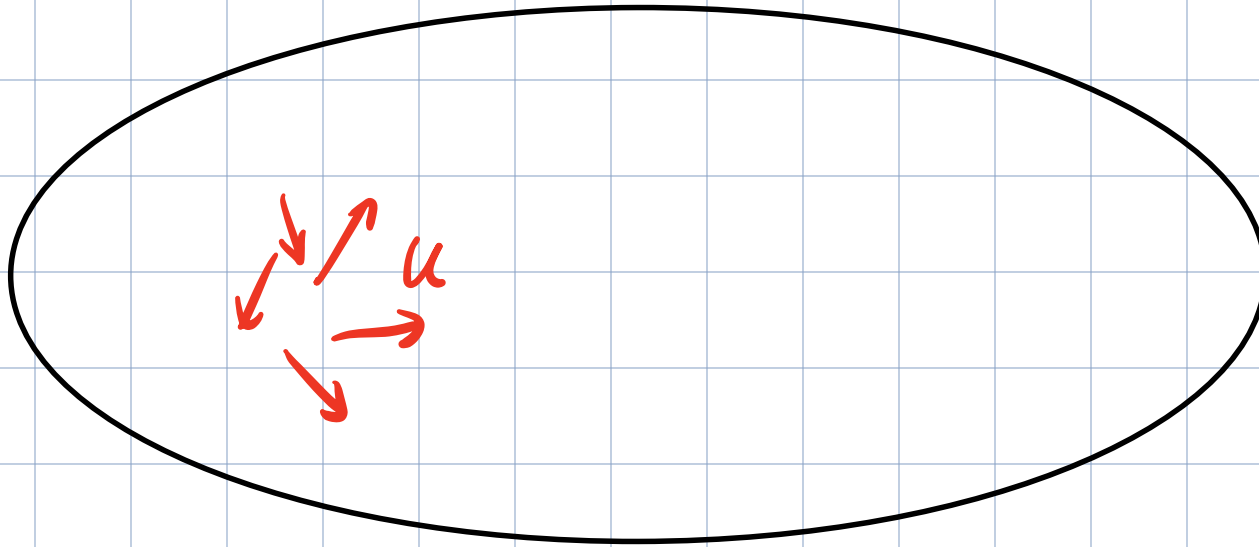


Professor
Jeremy Fineman

Remote Sets

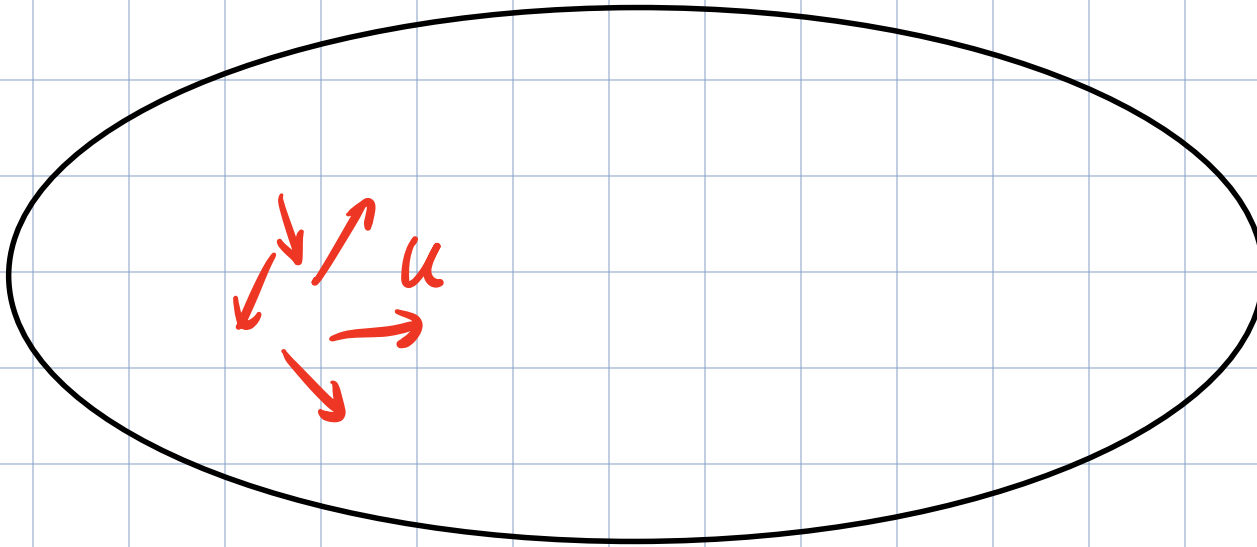


Remote Sets



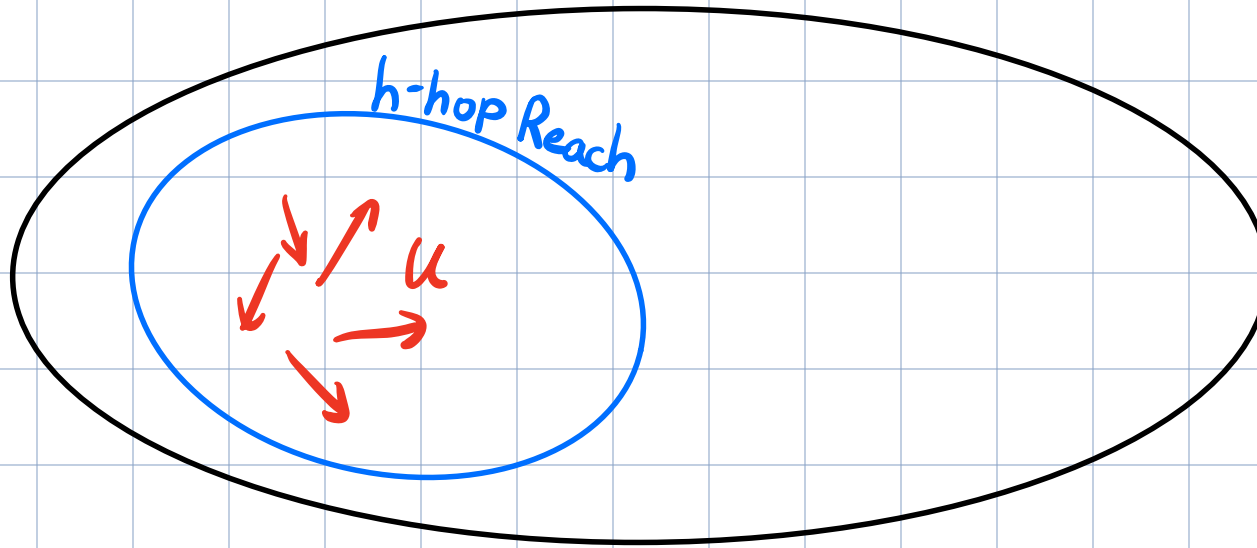
For a set of neg. edges
 u

Remote Sets



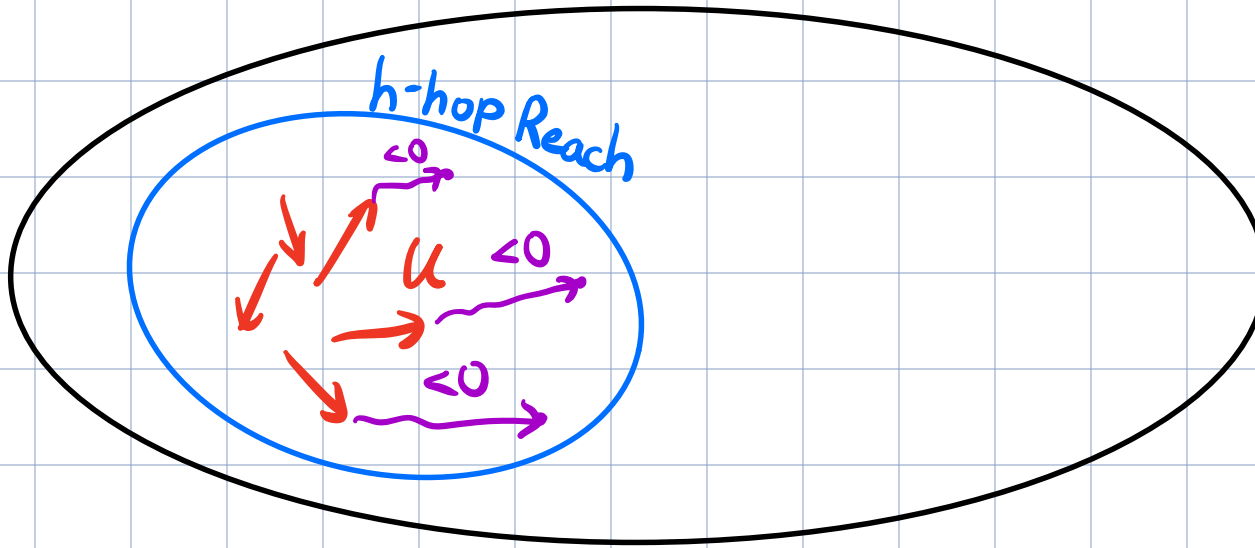
For a set of neg. edges U , the h -hop reach of U is the set of $v \in V$ where $d^h(u, v) < 0$ for some $u \in U$.

Remote Sets



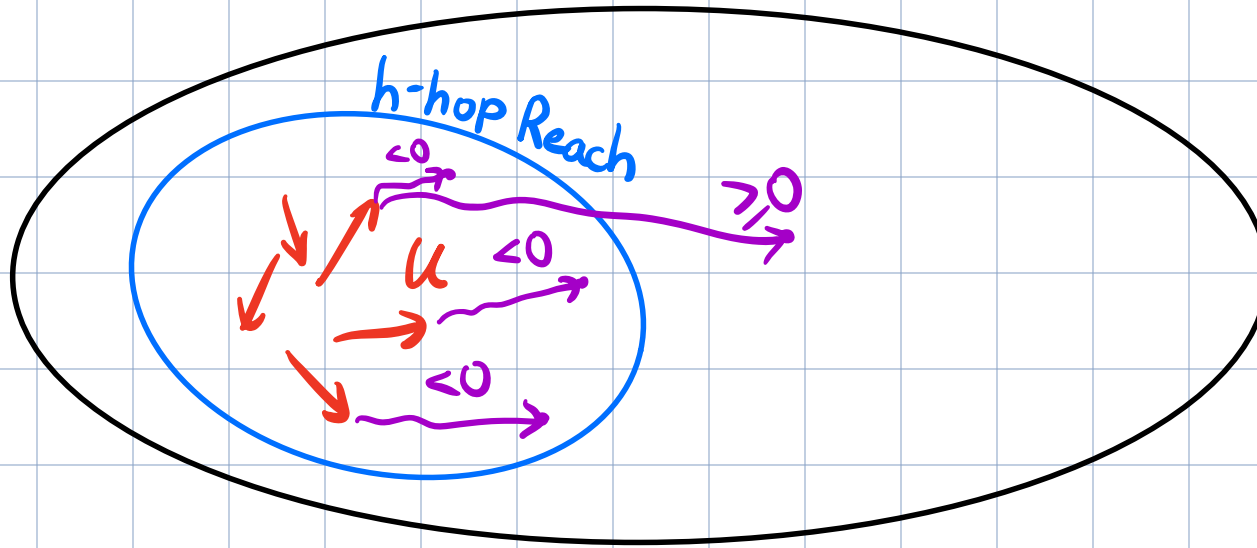
For a set of neg. edges U , the h -hop reach of U is the set of $v \in V$ where $d^h(u, v) < 0$ for some $u \in U$.

Remote Sets



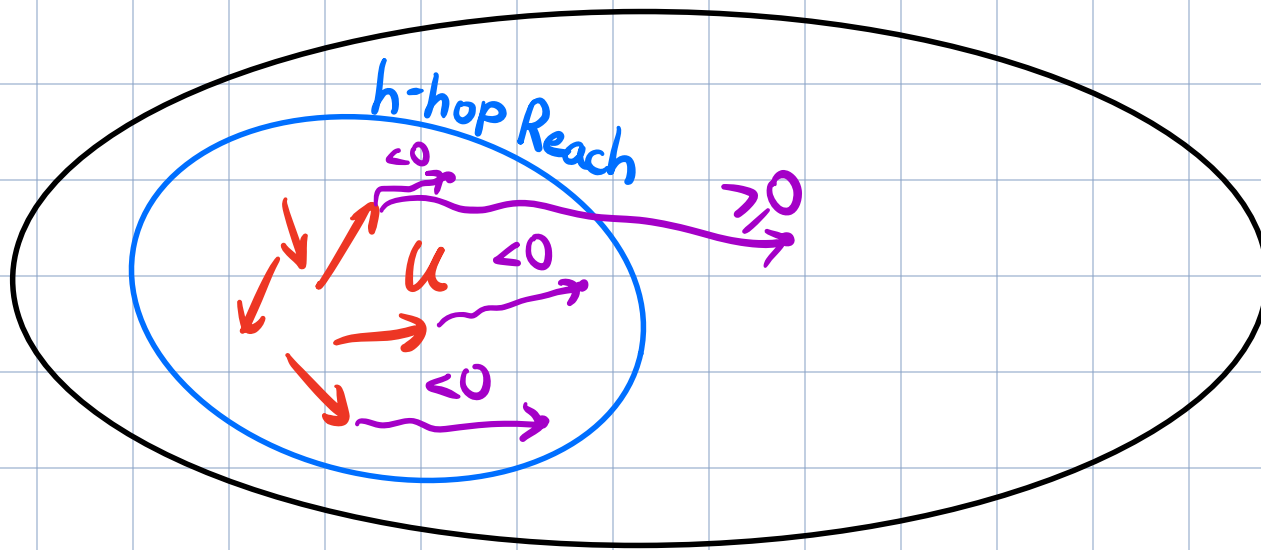
For a set of neg. edges U , the h -hop reach of U is the set of $v \in V$ where $d^h(u, v) < 0$ for some $u \in U$.

Remote Sets



For a set of neg. edges U , the h -hop reach of U is the set of $v \in V$ where $d^h(u, v) < 0$ for some $u \in U$.

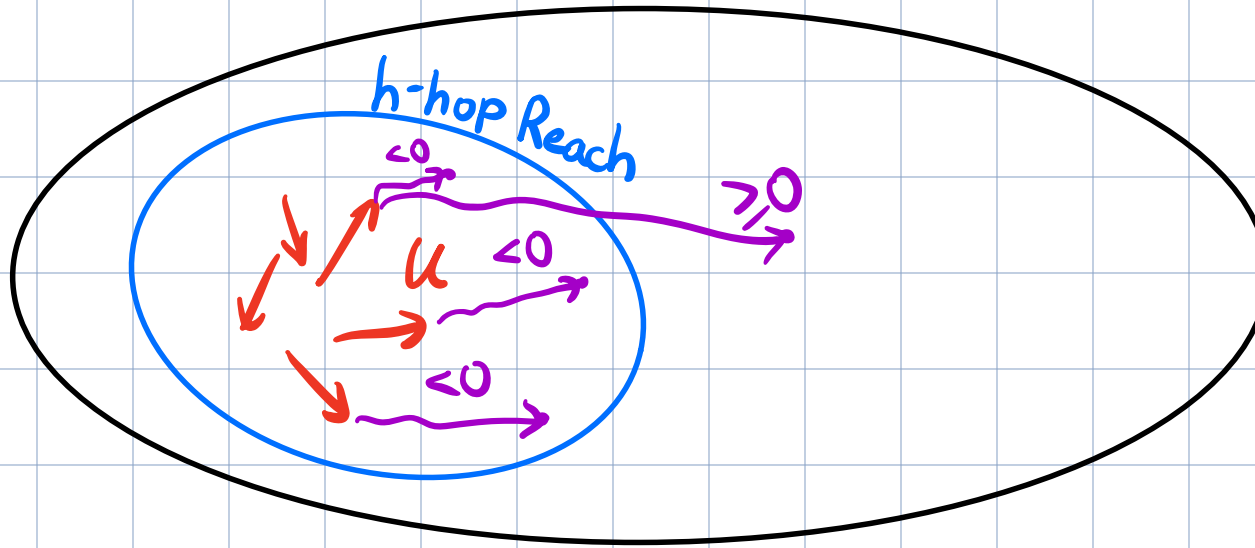
Remote Sets



For a set of neg. edges U , the h -hop reach of U is the set of $v \in V$ where $d^h(u, v) < 0$ for some $u \in U$.

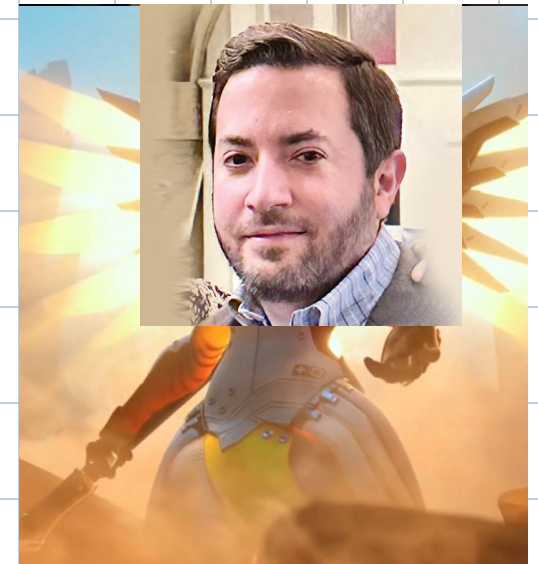
U is remote if its h -hop reach is $\frac{n}{h}$.

Remote Sets

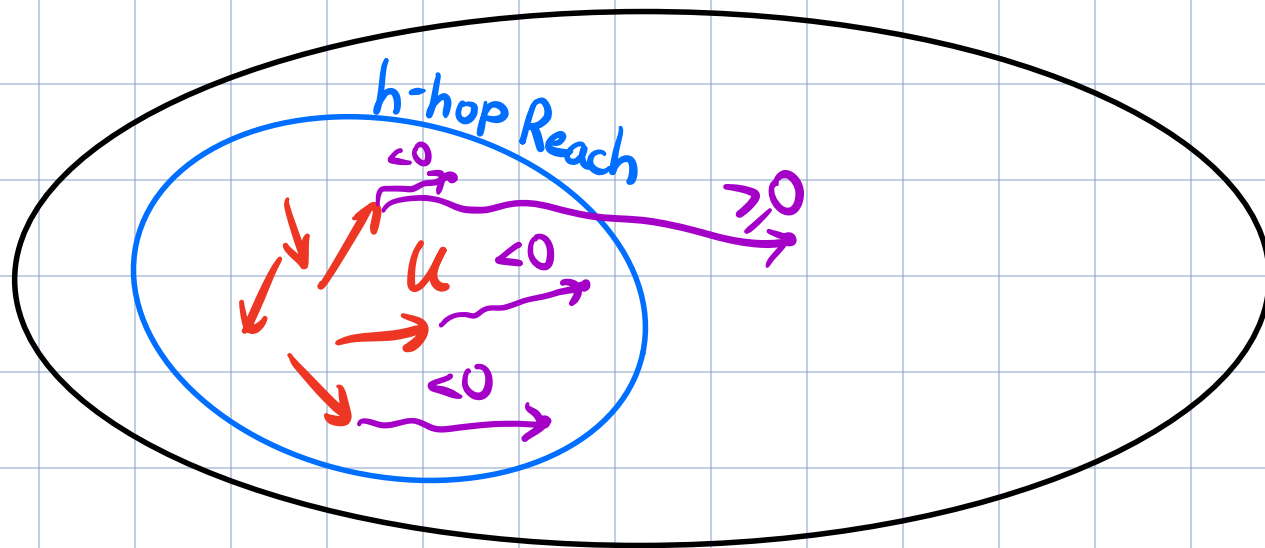


For a set of neg. edges U , the h -hop reach of U is the set of $v \in V$ where $d^h(u, v) < 0$ for some $u \in U$.

U is remote if its h -hop reach is $\frac{n}{h}$.



Remote Sets



For a set of neg. edges U , the h -hop reach of U is the set of $v \in V$ where $d^h(u, v) < 0$ for some $u \in U$.

U is remote if its h -hop reach is $\frac{n}{h}$.

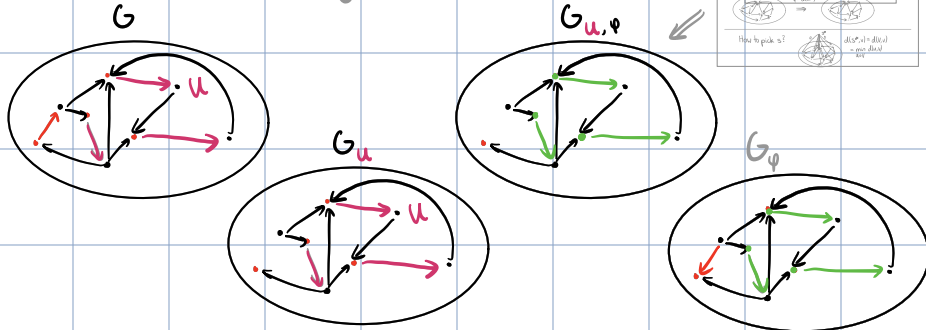
I can find a big h -hop remote set



Incremental Elimination

Idea: Neutralize one edge set at a time

Shortest path to
 Suppose some vertices u are not at v (let $q(u,v)$ be the
 length of the shortest path from u to v). Then
 Distance potentials make
 all edges nonnegative
 $d(u,v) = d(u,s) + d(s,v)$
 How to pick s ?
 $d(u,s) + d(s,v) \rightarrow \min_s$



Pick neg
 edge set U

Remove other
 neg. edges

Neutralize G_u
 with $d^{(u)}(s, \cdot)$

Add edges
 back

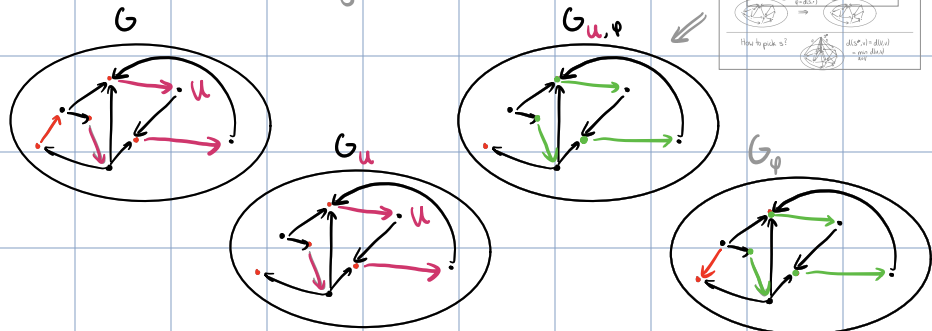
Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m|U|)$ per edge set $\times \frac{k}{|U|}$ edges = $\tilde{O}(km)$ 😭

one set
 at a time

Incremental Elimination

Idea: Neutralize one edge set at a time



Pick neg edge set U

Remove other neg. edges

Neutralize G_u with $d^{(u)}(s, \cdot)$

Add edges back

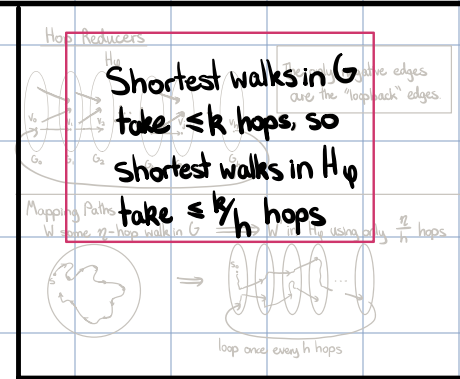
Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m|U|)$ per edge set $\times \frac{k}{|U|}$ edges = $\tilde{O}(km)$ 😭

one set
at a time

An Algorithm?

1. Pick h
2. Construct H
3. Compute φ & reweight H
4. Find distances in G with k/h -hop distances in H



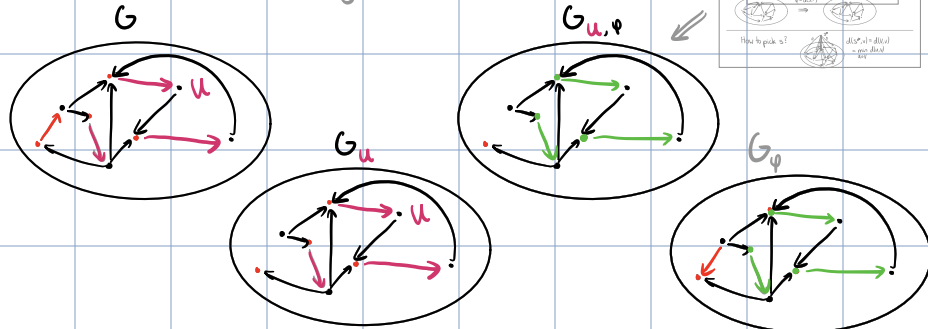
Running Time: $\tilde{O}(\frac{k}{h}|H|) = \tilde{O}(\frac{k}{h}(hm)) = \tilde{O}(km)$ 😞

What went wrong? Hops $\downarrow$ by h but graph $\uparrow$ by h .

reducing hops

Incremental Elimination

Idea: Neutralize one edge set at a time



Pick neg edge set U

Remove other neg. edges

Neutralize G_u with $d^{(u)}(s, \cdot)$

Add edges back

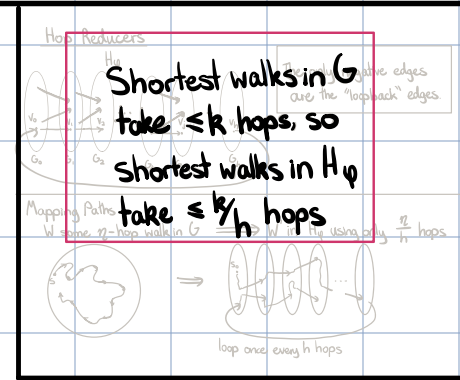
Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m|U|)$ per edge set $\times \frac{k}{|U|}$ edges = $\tilde{O}(km)$ 😭

one set at a time

An Algorithm?

1. Pick h
2. Construct H
3. Compute φ & reweight H
4. Find distances in G with k/h -hop distances in H



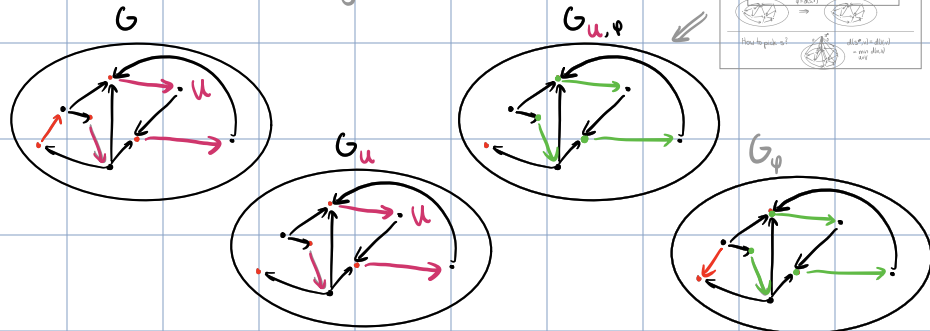
Running Time: $\tilde{O}(\frac{k}{h}|H|) = \tilde{O}(\frac{k}{h}(hm)) = \tilde{O}(km)$ 😞

What went wrong? Hops $\downarrow$ by h but graph $\uparrow$ by h .

reducing hops

Incremental Elimination

Idea: Neutralize one edge set at a time



Pick neg edge set U

Remove other neg. edges

Neutralize G_u with $d^{|U|}(s, \cdot)$

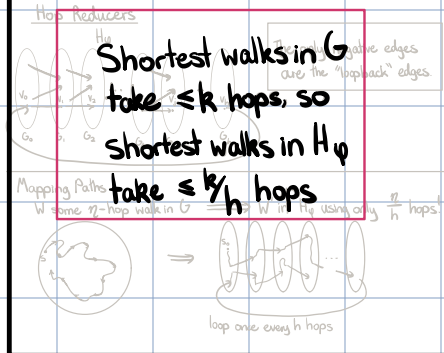
Add edges back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m|U|)$ per edge set $\times \frac{k}{|U|}$ edges = $\tilde{O}(km)$ 😭

An Algorithm?

1. Pick h
2. Construct H
3. Compute φ & reweight H
4. Find distances in G with k/h -hop distances in H



Running Time: $\tilde{O}(\frac{k}{h} |H|) = \tilde{O}(\frac{k}{h} (hm)) = \tilde{O}(km)$ 😞

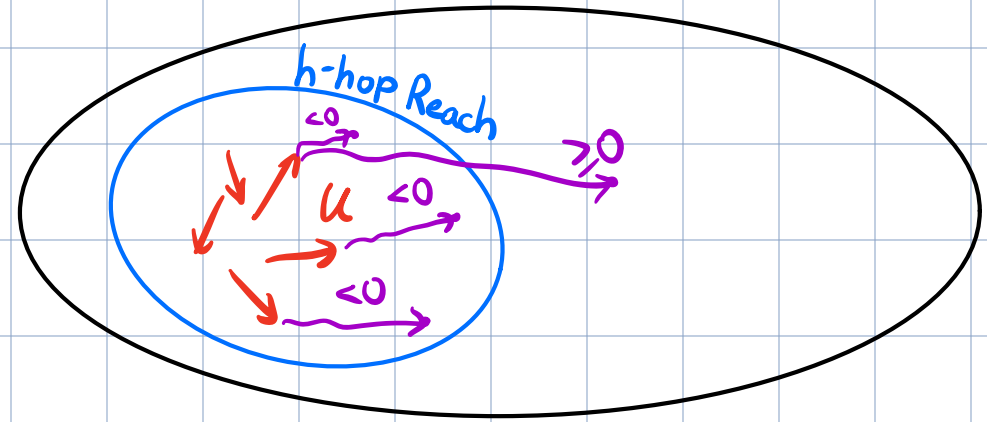
What went wrong? Hops $\downarrow$ by h but graph $\uparrow$ by h .

one set
at a time

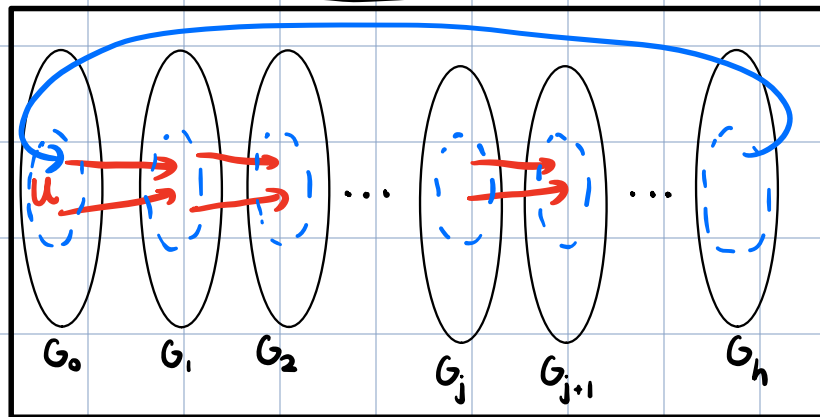
reducing hops

Make an h -hop reducer
for a remote set.

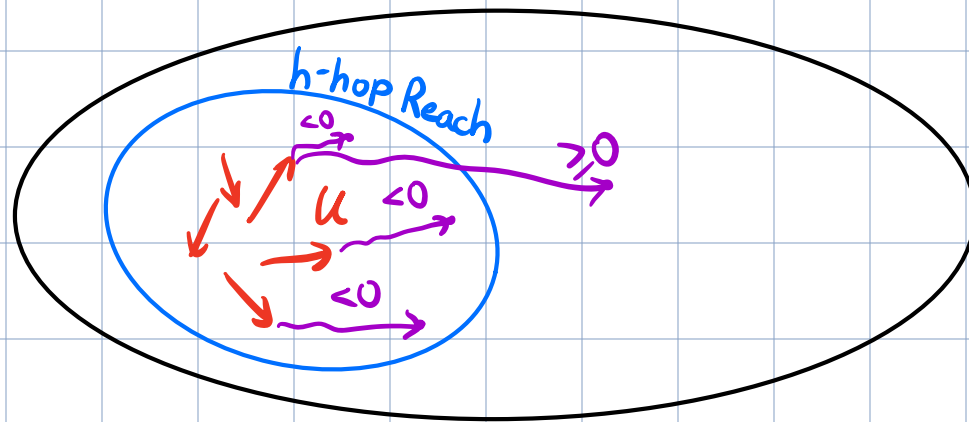
G



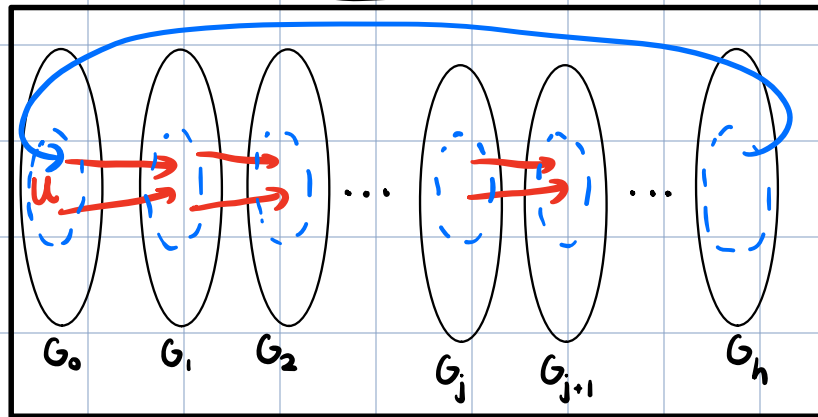
Old H



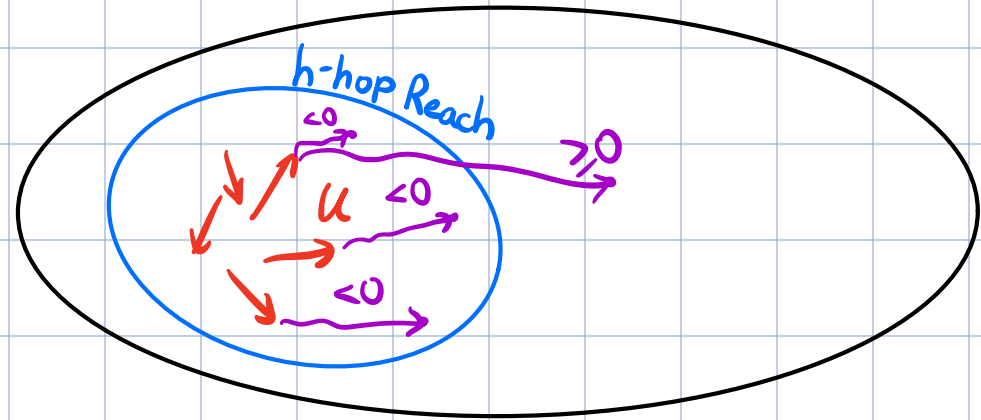
G



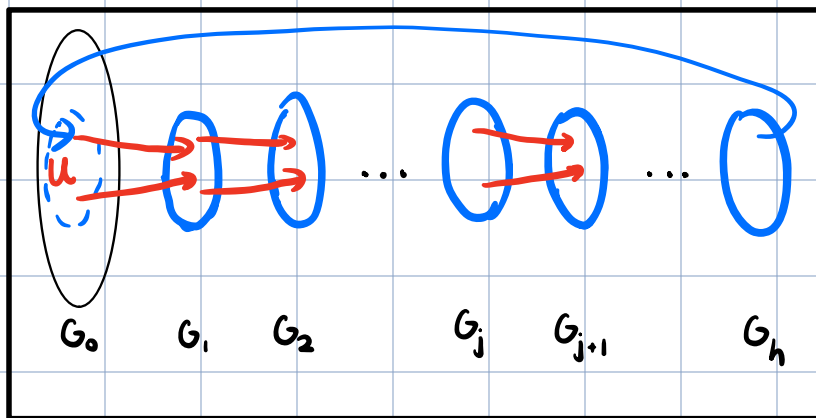
Old H



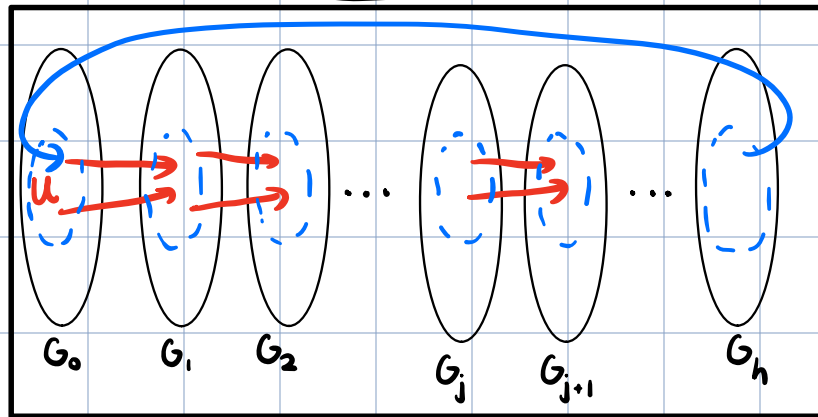
G



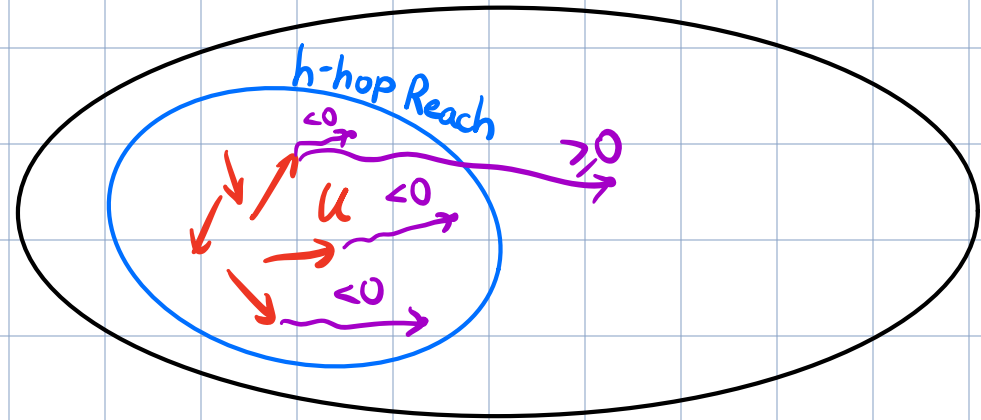
New H: Only copy reach



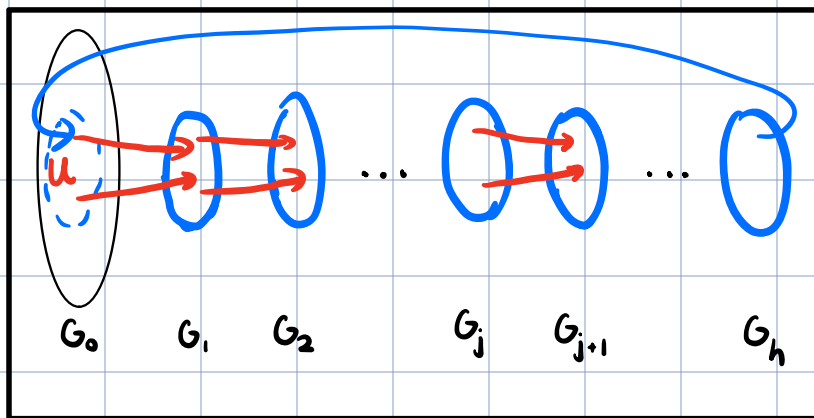
Old H



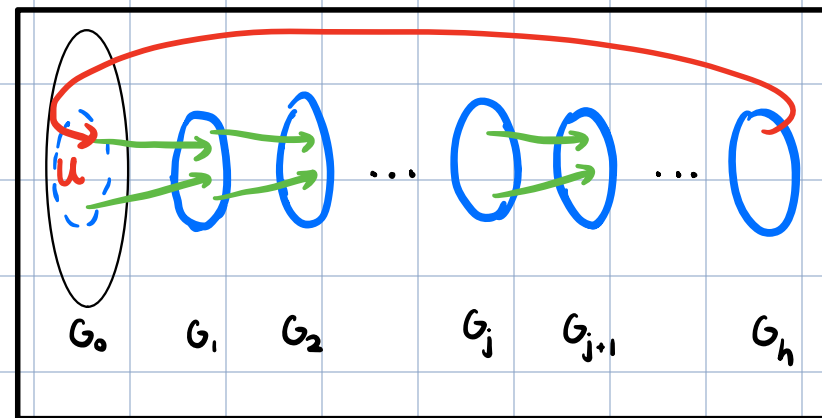
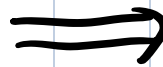
G



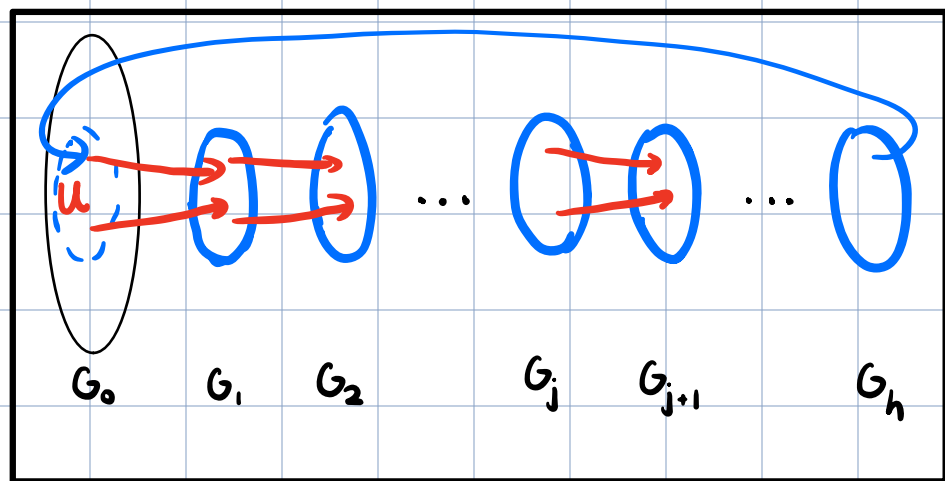
New H: Only copy reach



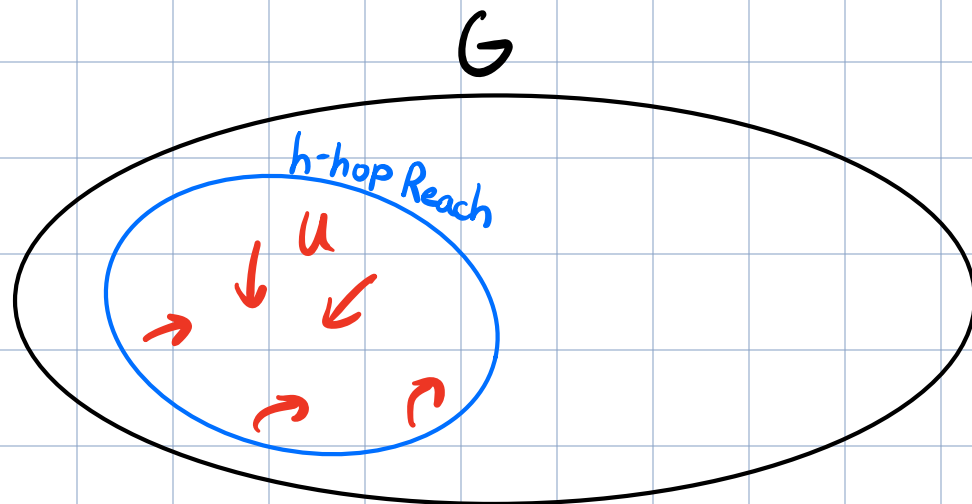
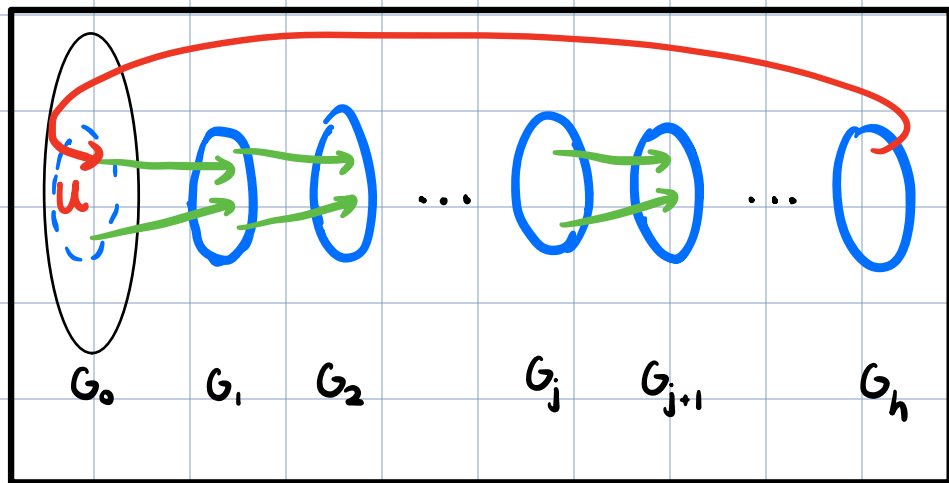
$$\varphi(v_j) = d^j(V, v)$$



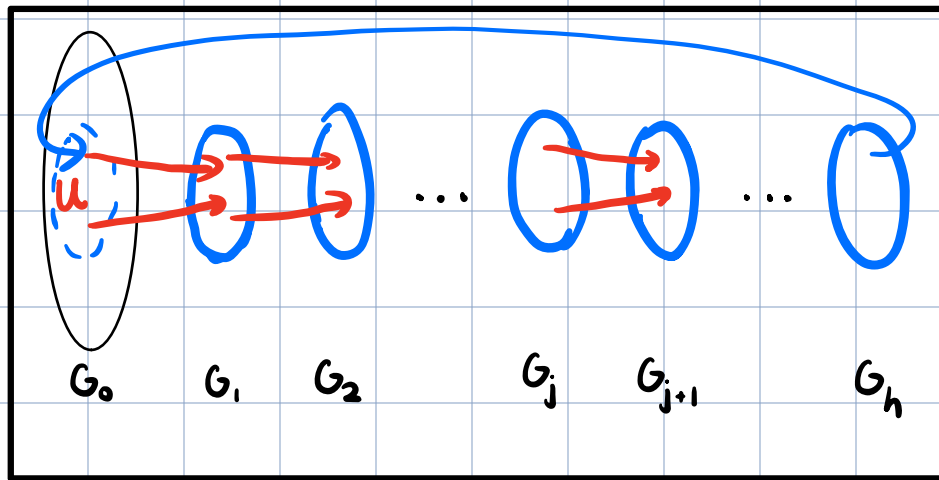
New H: Only copy reach



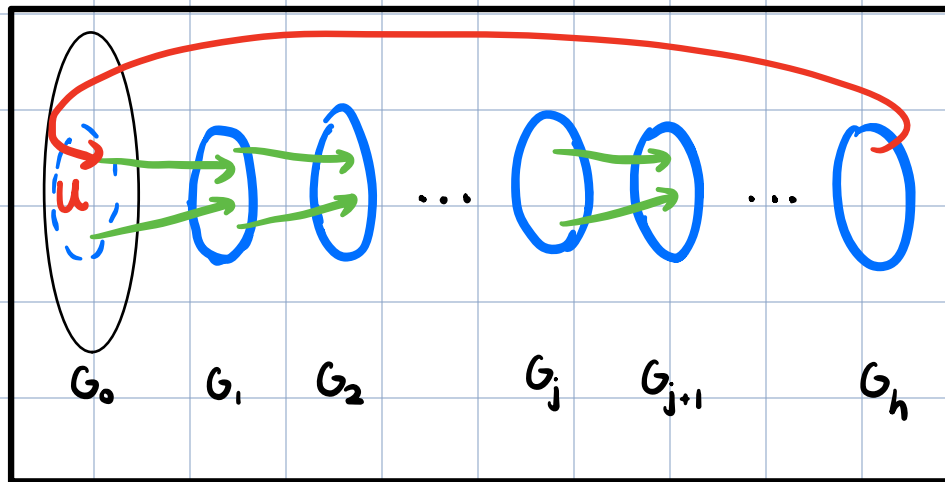
$\Downarrow \varphi$



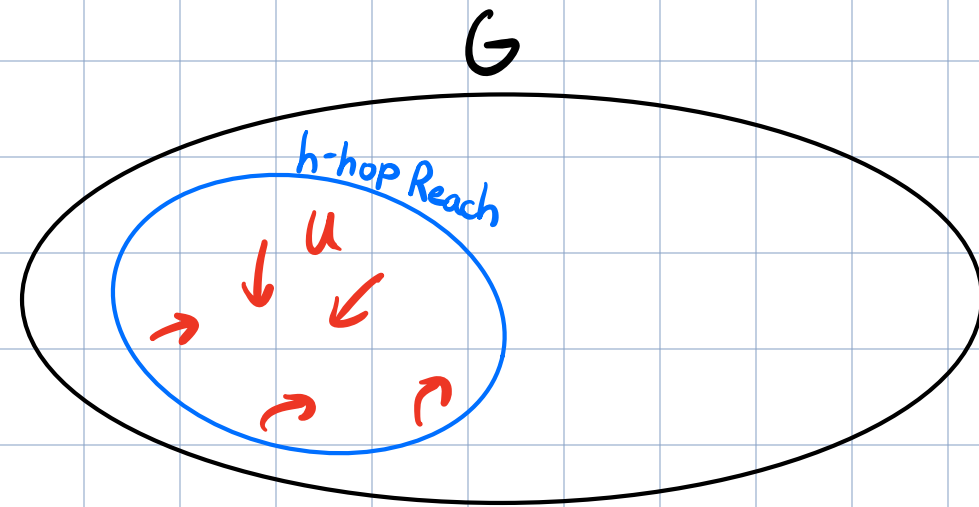
New H: Only copy reach



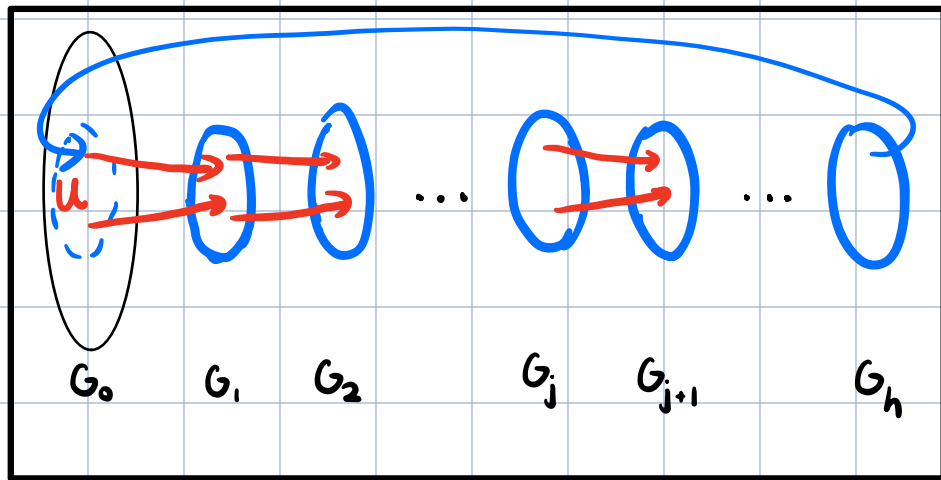
$\Downarrow \varphi$



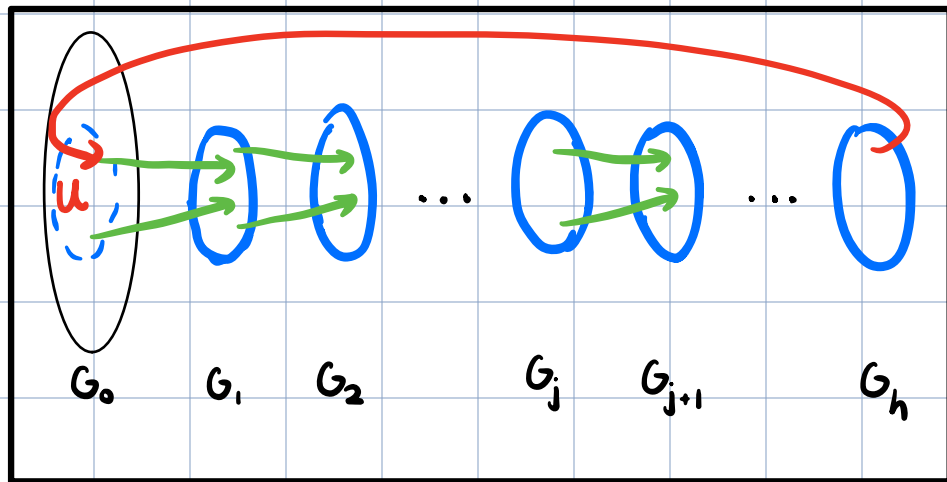
Now h has
edges.



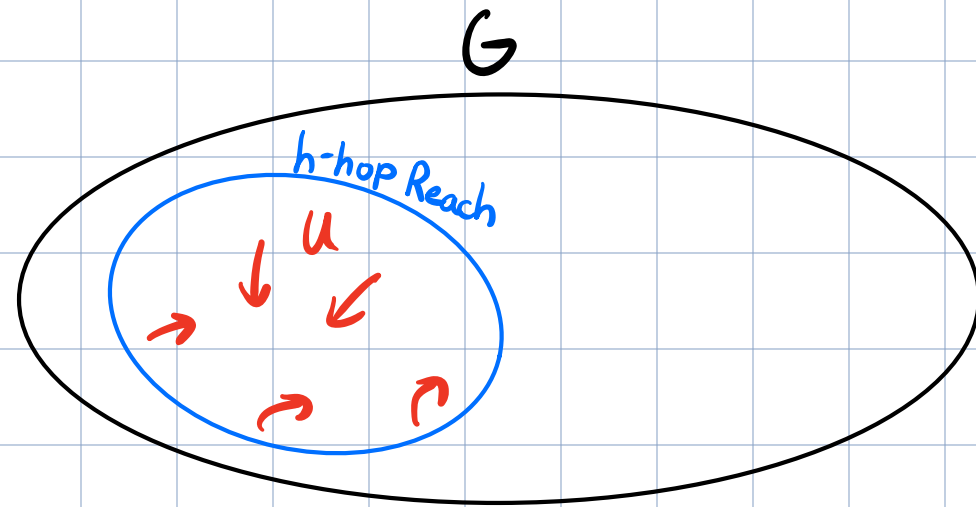
New H: Only copy reach



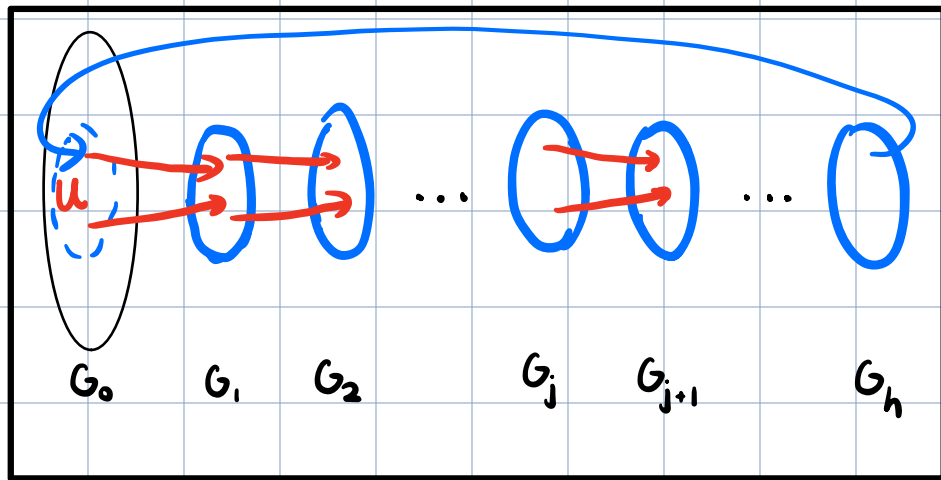
$\Downarrow \varphi$



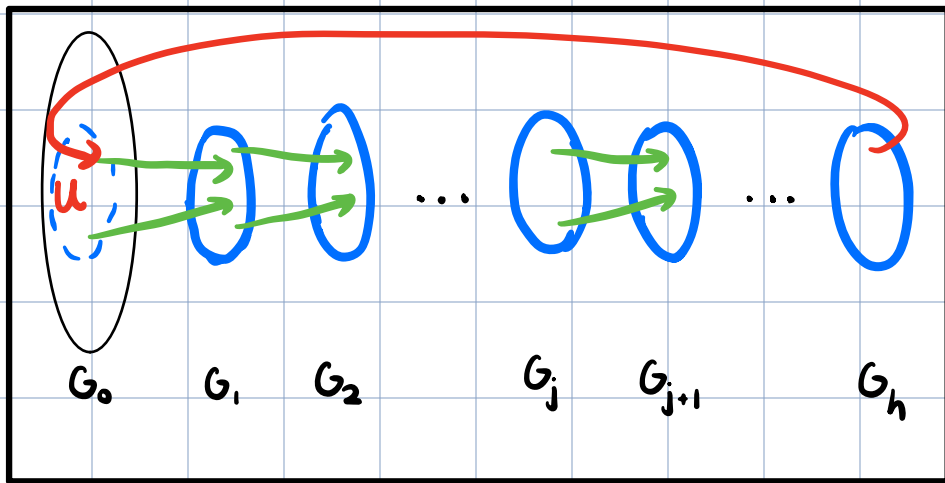
Now h has $m + h\left(\frac{m}{h}\right) = O(m)$ edges.



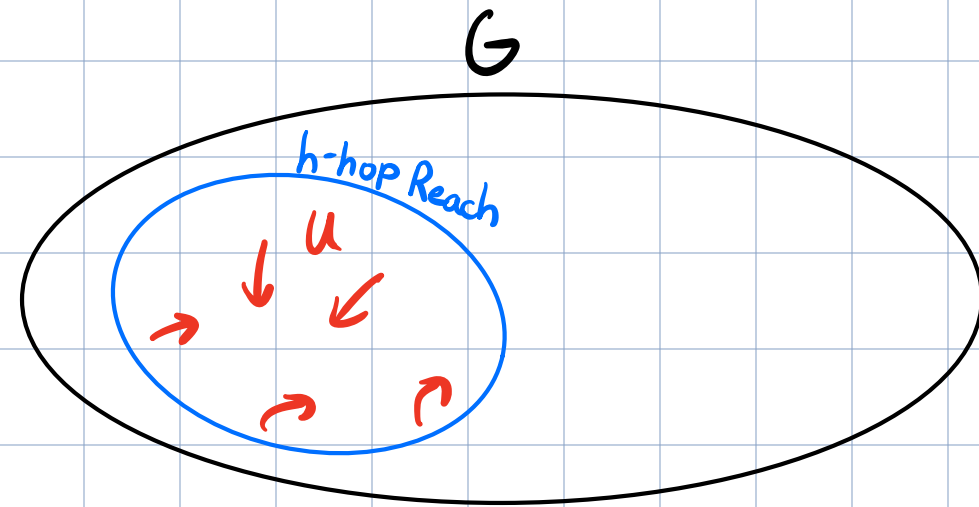
New H: Only copy reach



$\Downarrow \varphi$

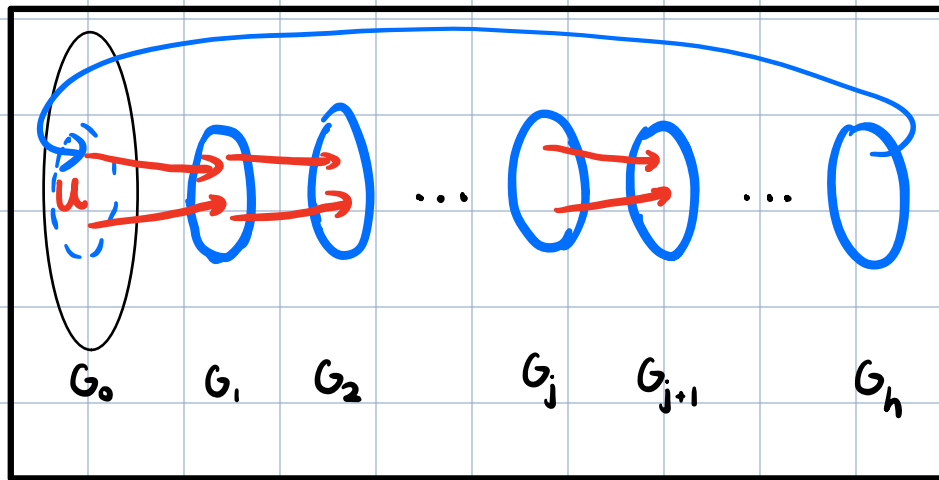


Now h has $m + h\left(\frac{m}{h}\right) = O(m)$ edges.

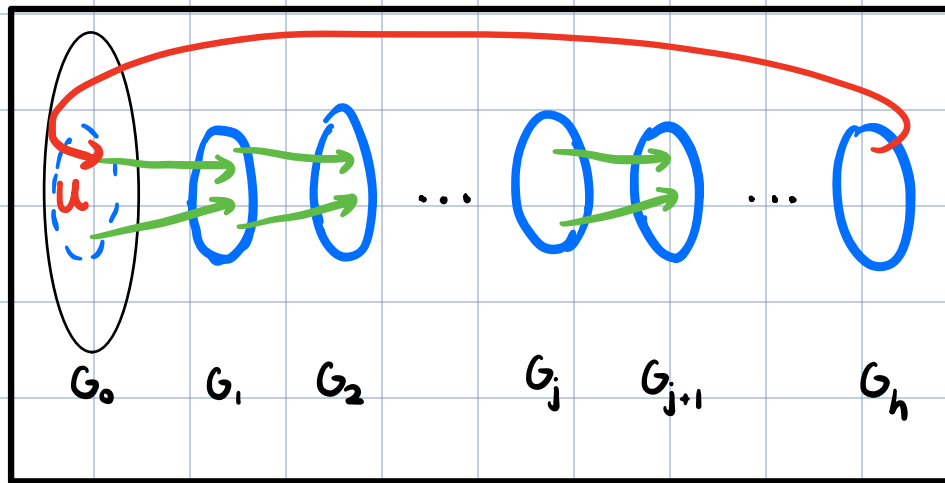


Does this reduce hops by factor of h ?

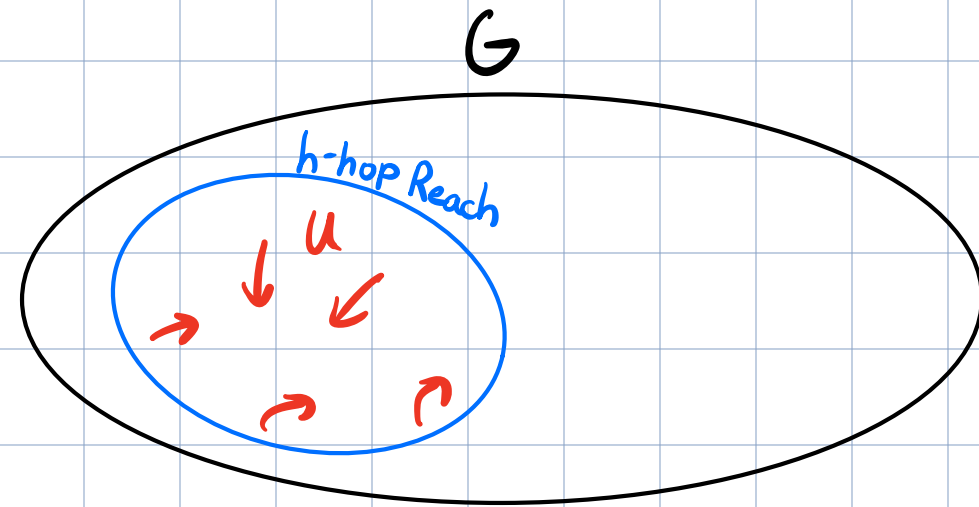
New H: Only copy reach



$\Downarrow \varphi$

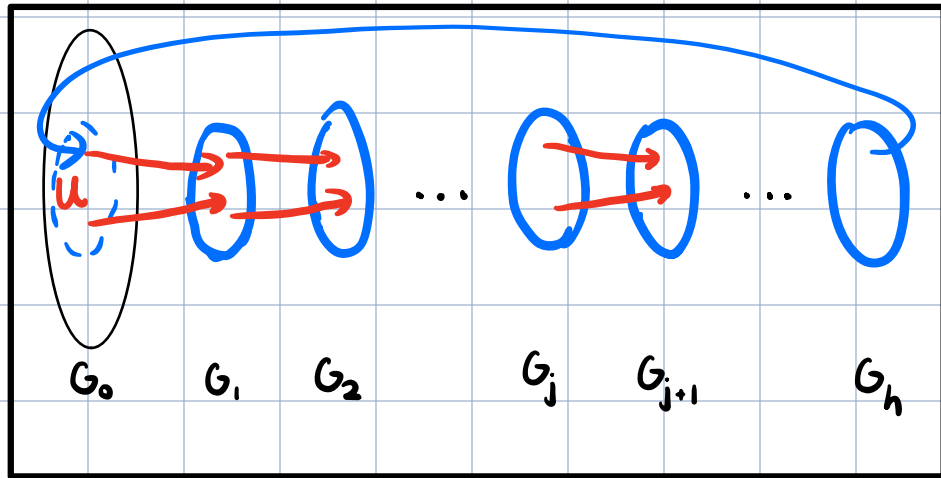


Now h has $m + h\left(\frac{m}{h}\right) = O(m)$ edges.

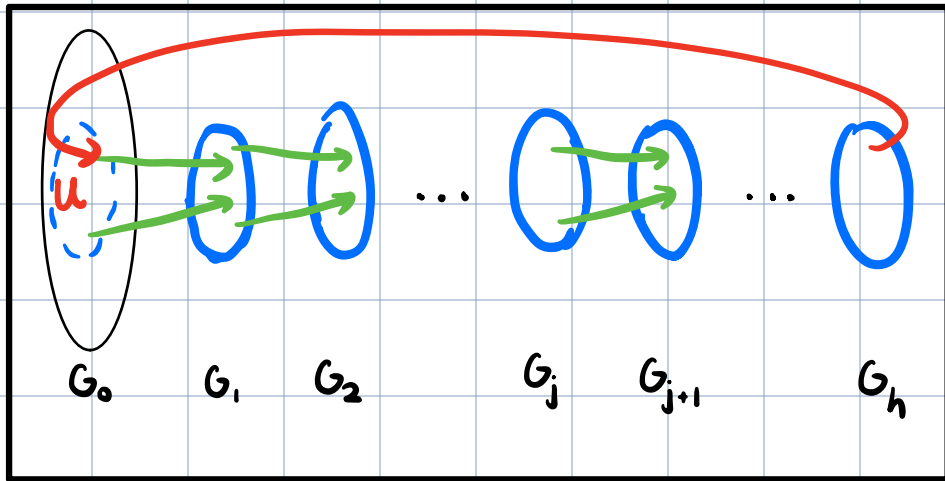


Does this reduce hops by factor of h ? No! Can't map some paths.

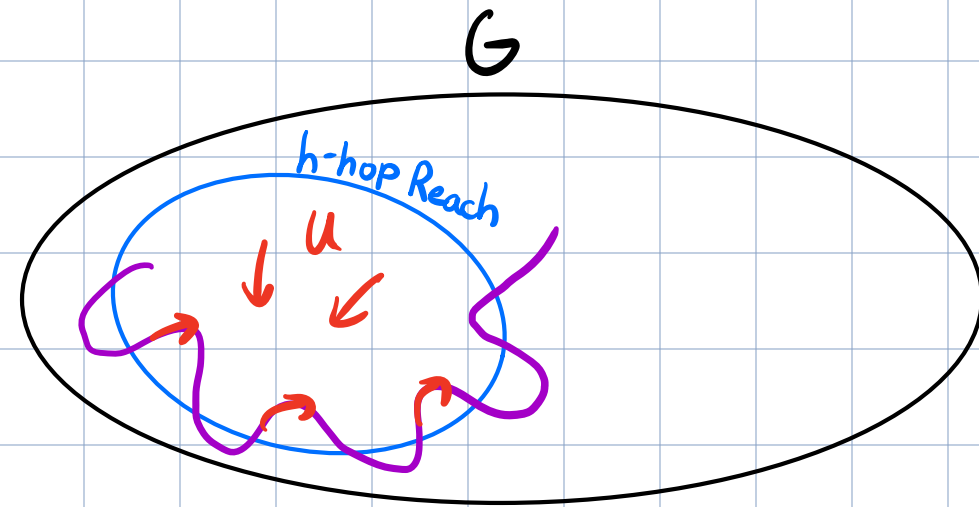
New H: Only copy reach



$\Downarrow \varphi$

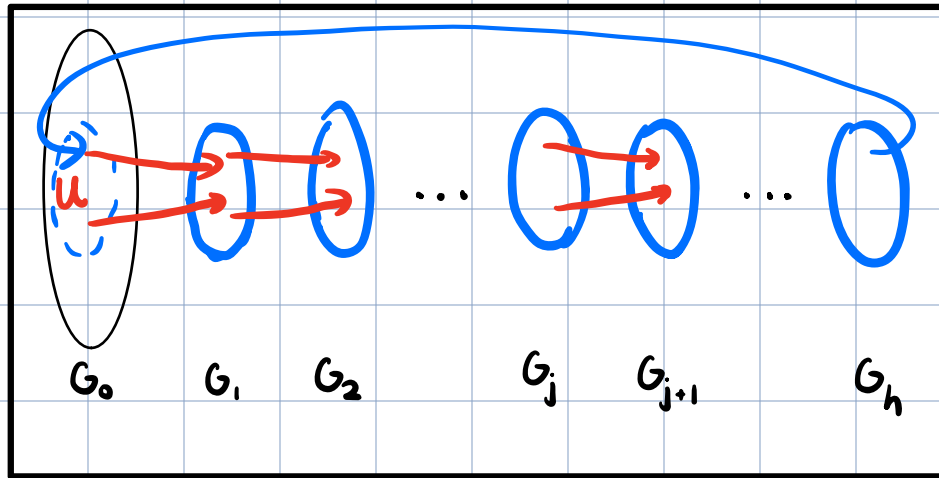


Now h has $m + h\left(\frac{m}{h}\right) = O(m)$ edges.

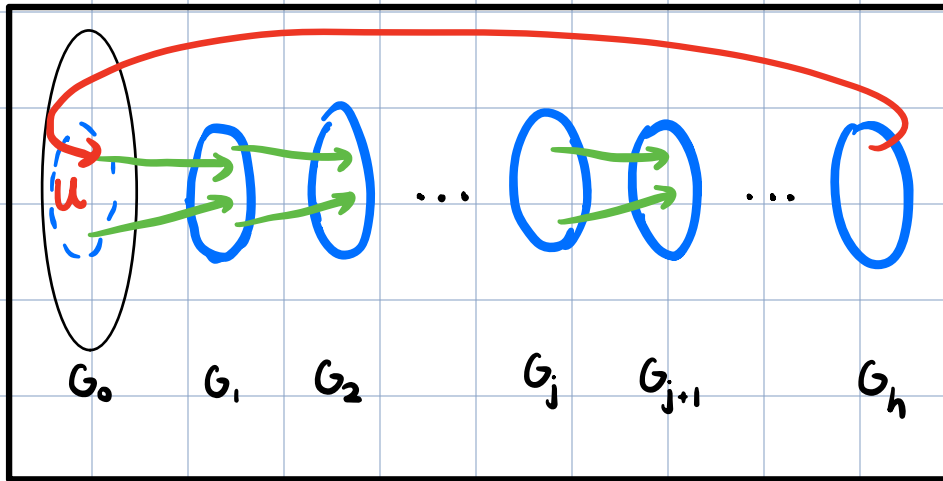


Does this reduce hops by factor of h ? No! Can't map some paths.

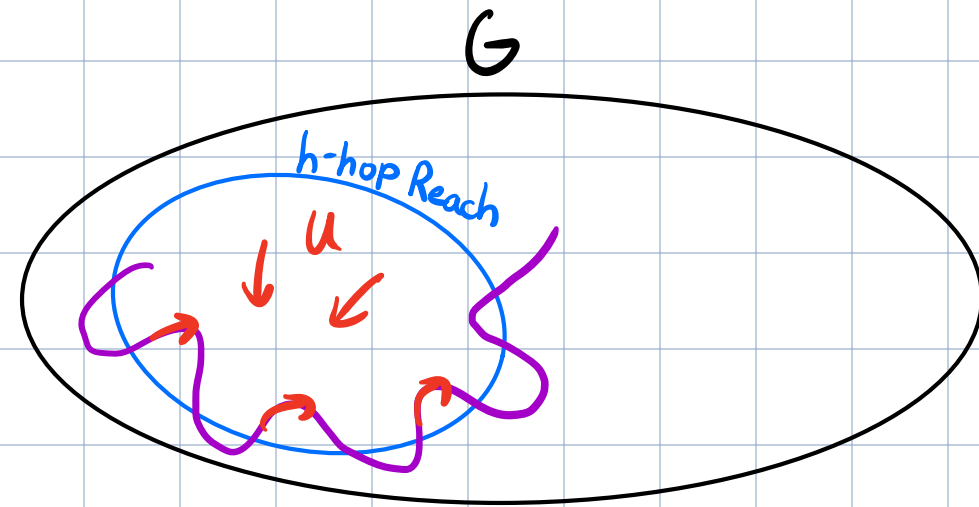
New H: Only copy reach



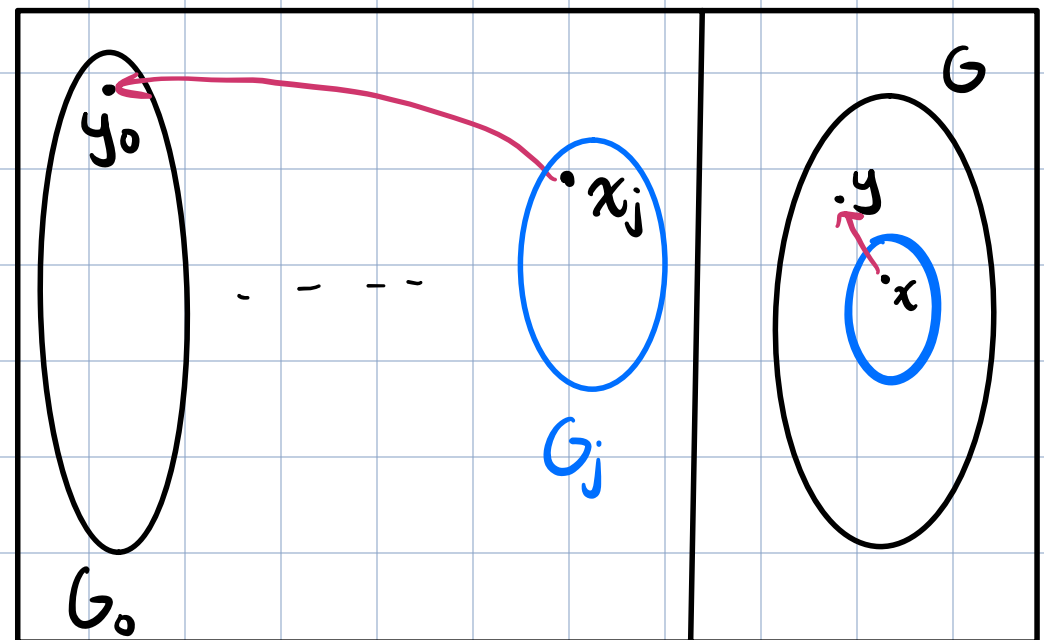
$\Downarrow \varphi$

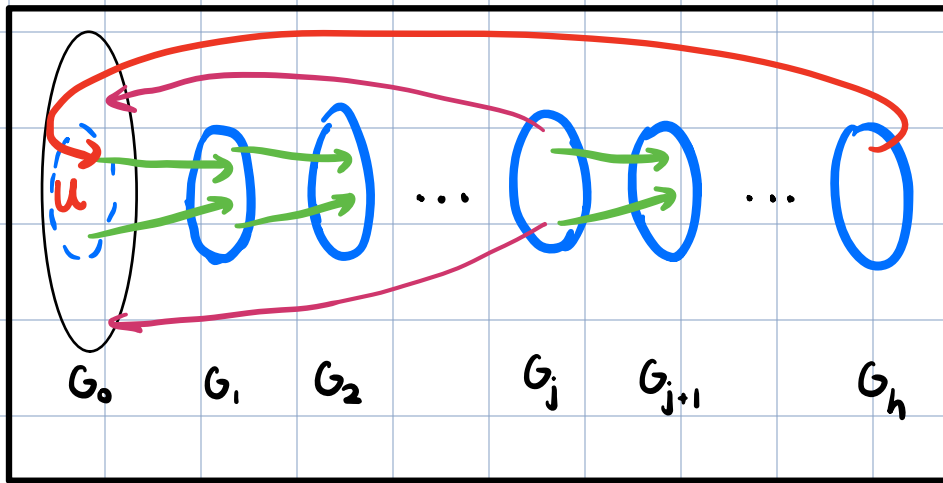


Now h has $m + h\left(\frac{m}{h}\right) = O(m)$ edges.

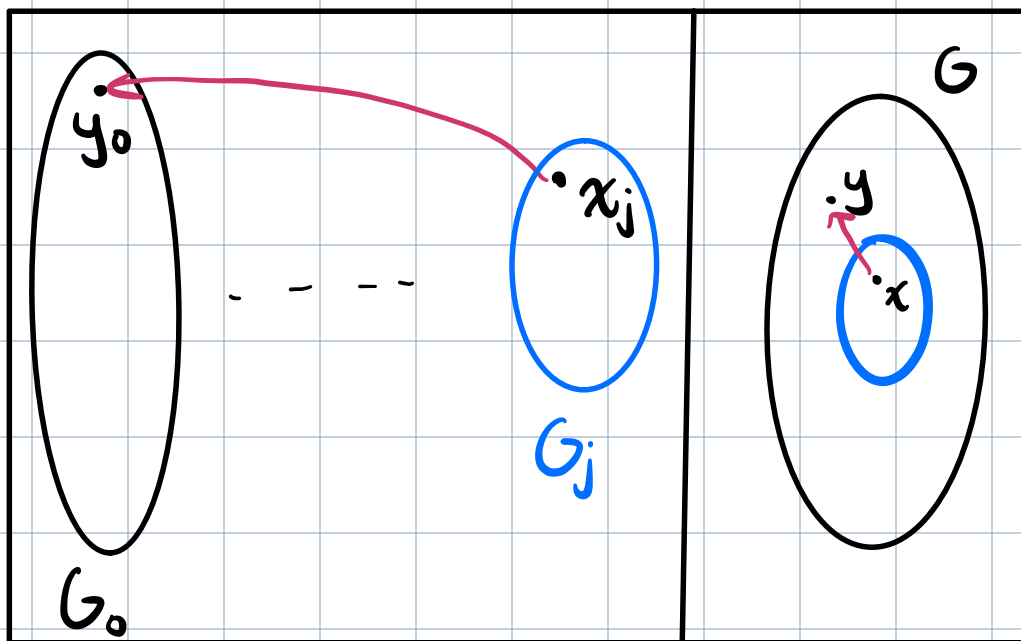


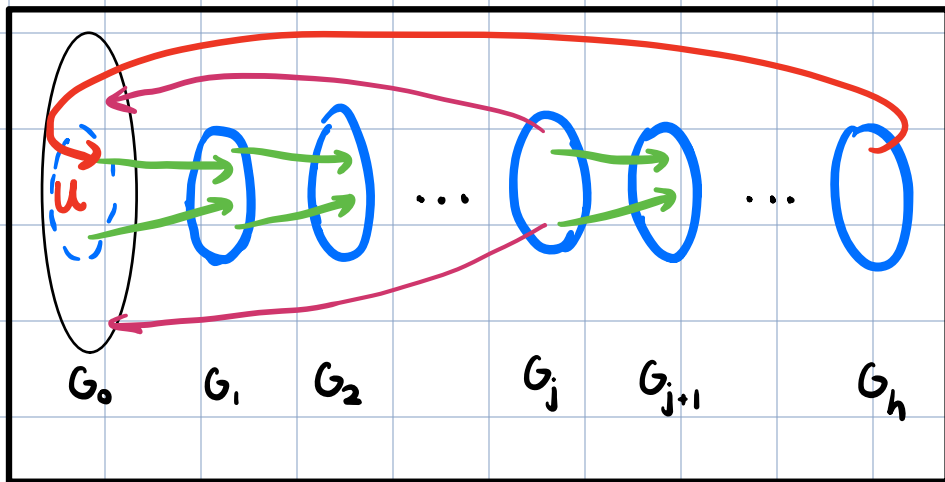
Does this reduce hops by factor of h ? No! Can't map some paths. Solution: "exit edges"





Solution: "exit edges"

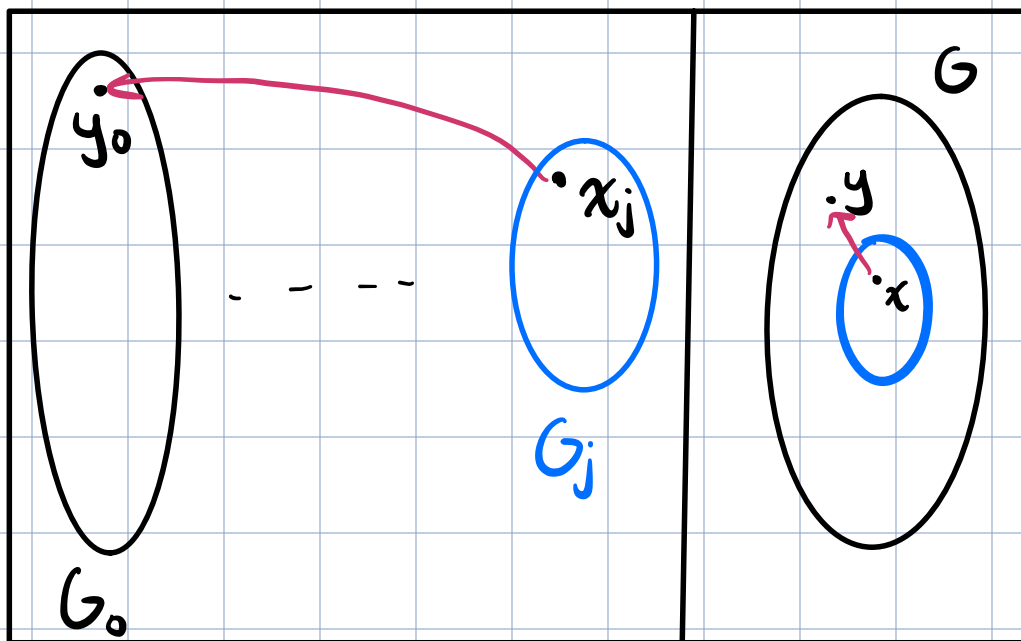


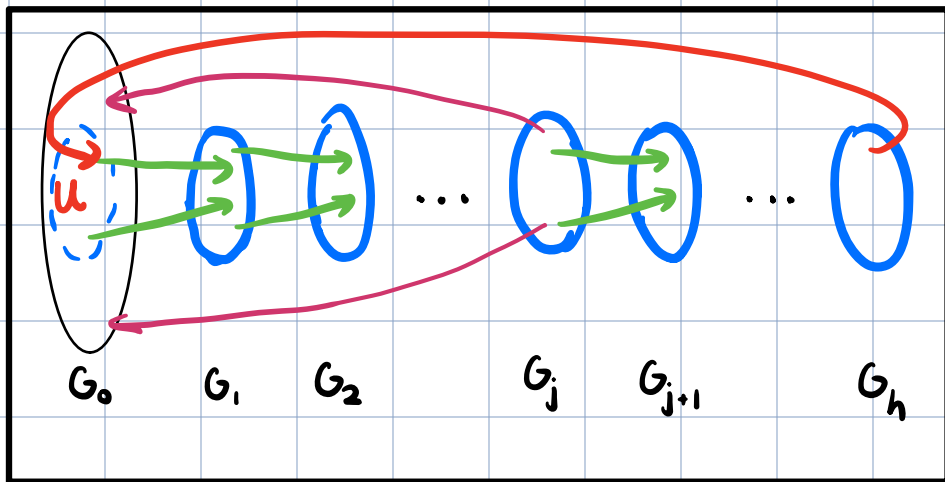


Reweight of exit edge is

$$l_p(x_j, y_0) =$$

Solution: "exit edges"

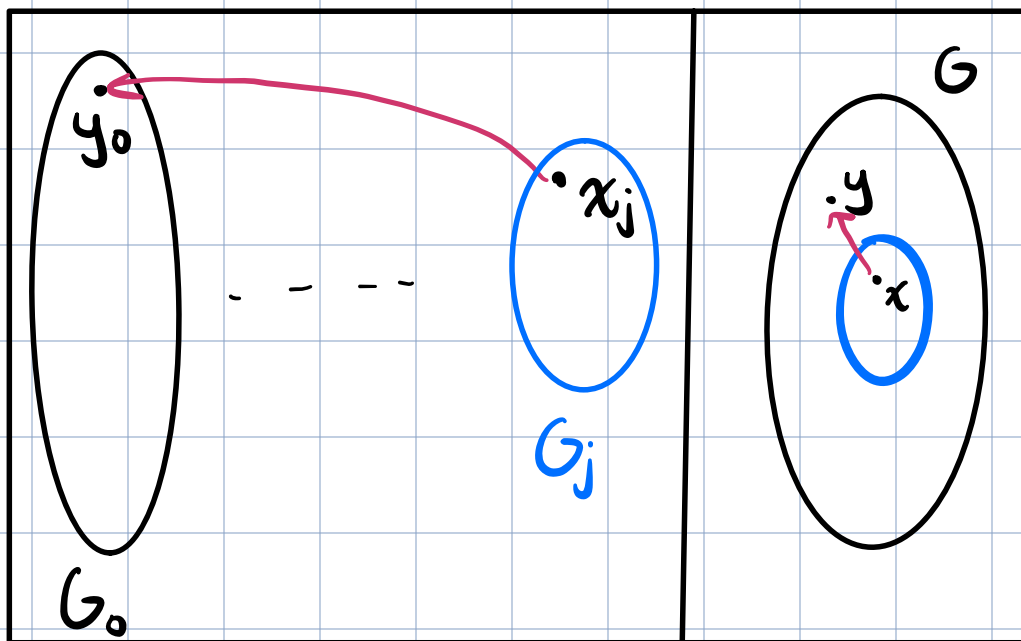


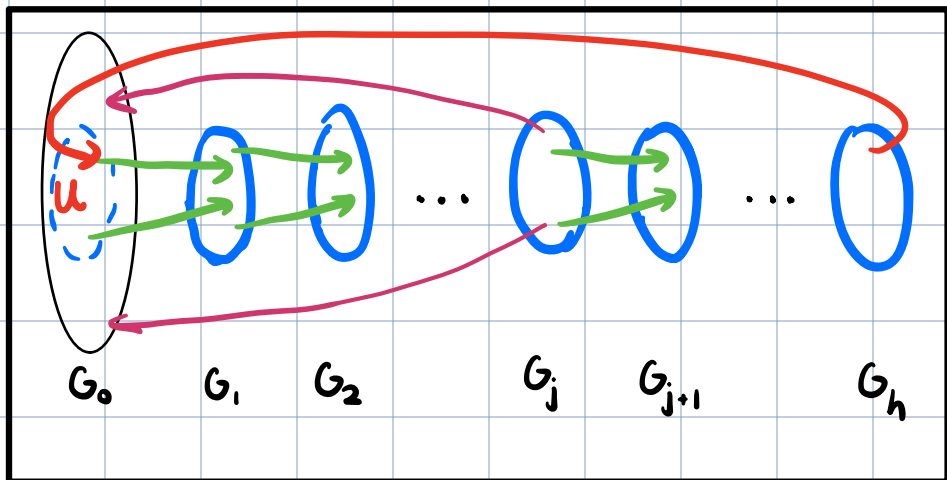


Reweight of exit edge is

$$l_p(x_j, y_0) = l(x, y) + d^j(v, x) - d^0(v, y)$$

Solution: "exit edges"

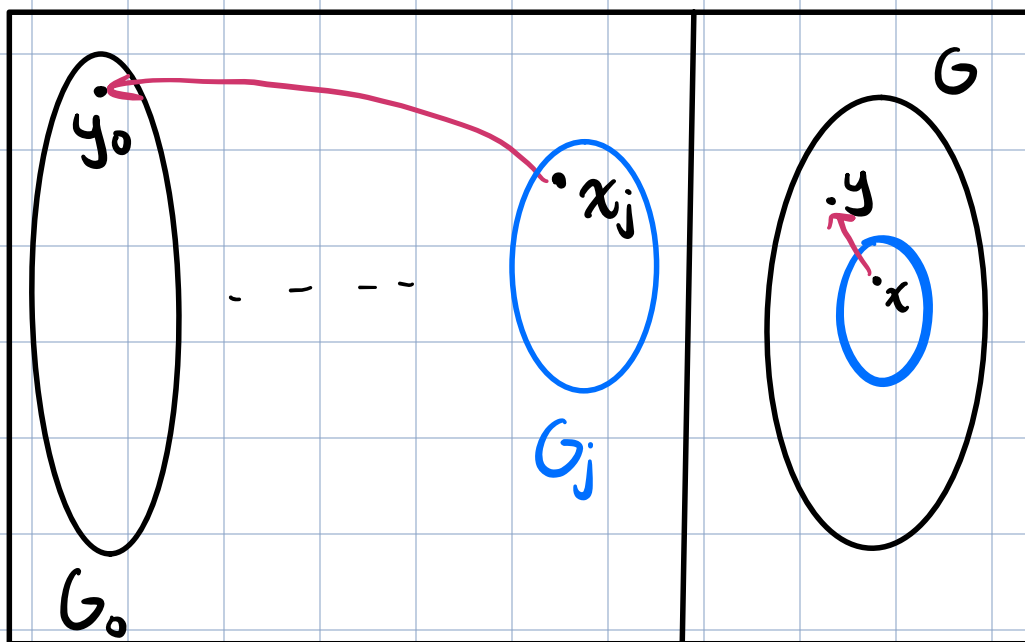


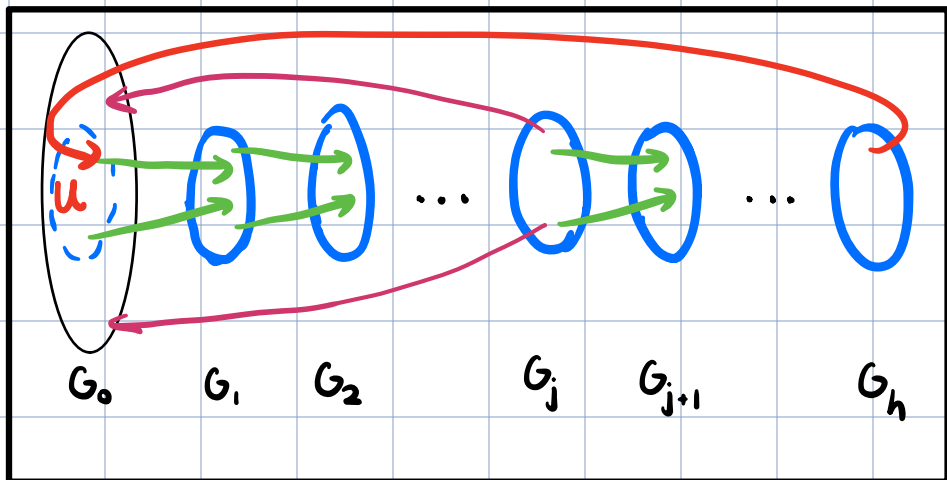


Reweight of exit edge is

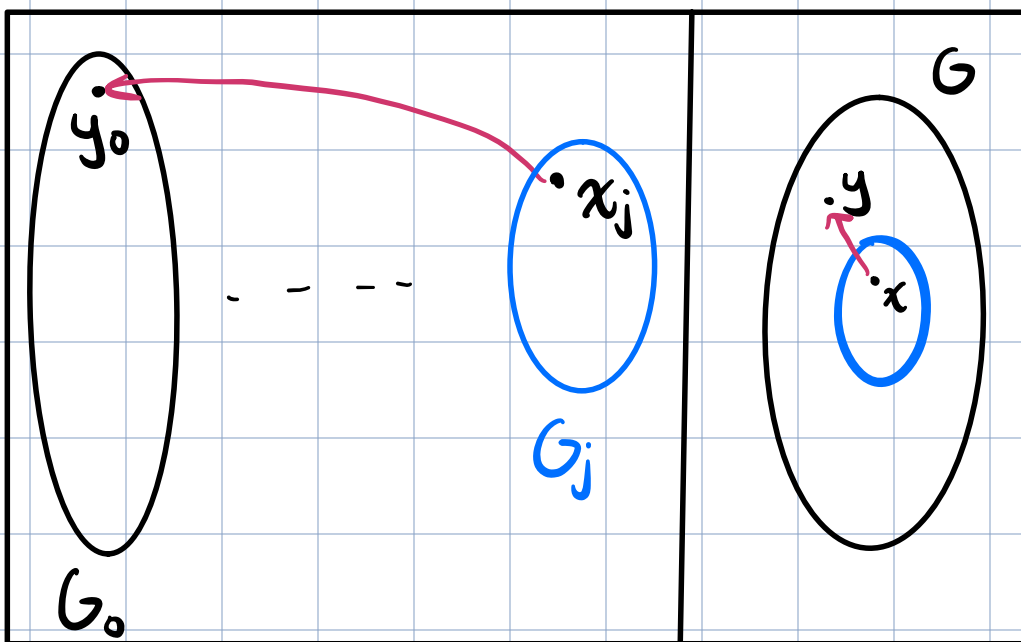
$$l_p(x_j, y_0) = l(x, y) + d^j(v, x) - \cancel{d^0(v, y)}$$

Solution: "exit edges"



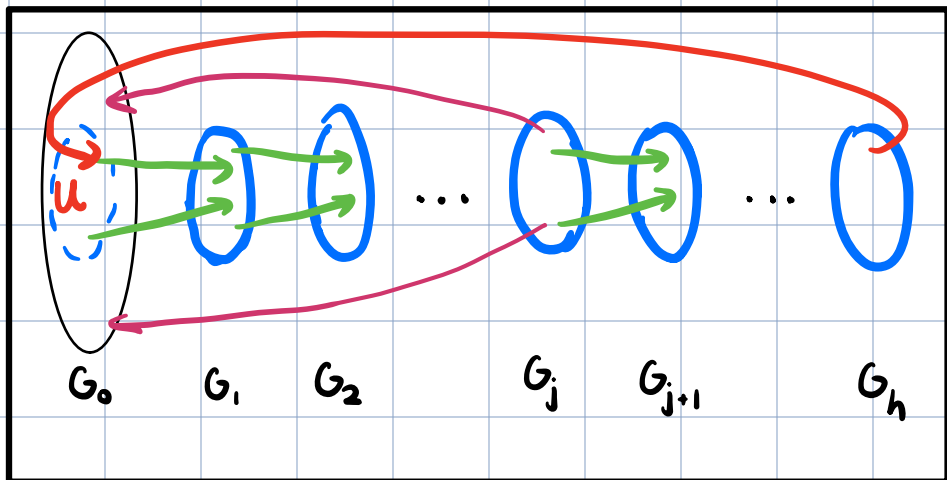


Solution: "exit edges"

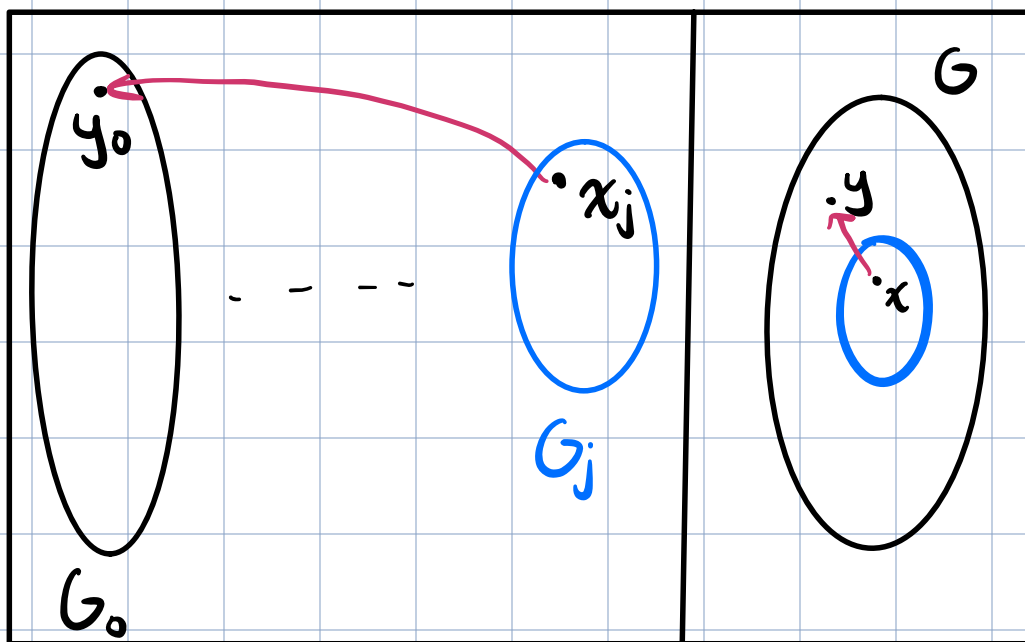


Reweight of exit edge is

$$l_p(x_j, y_0) = l(x, y) + d^j(V, x) - \cancel{d^0(V, y)}^0 \\ \geq d^j(V, y) \geq 0$$

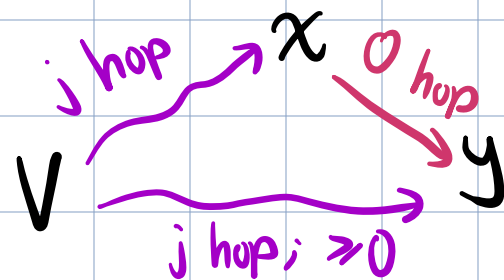


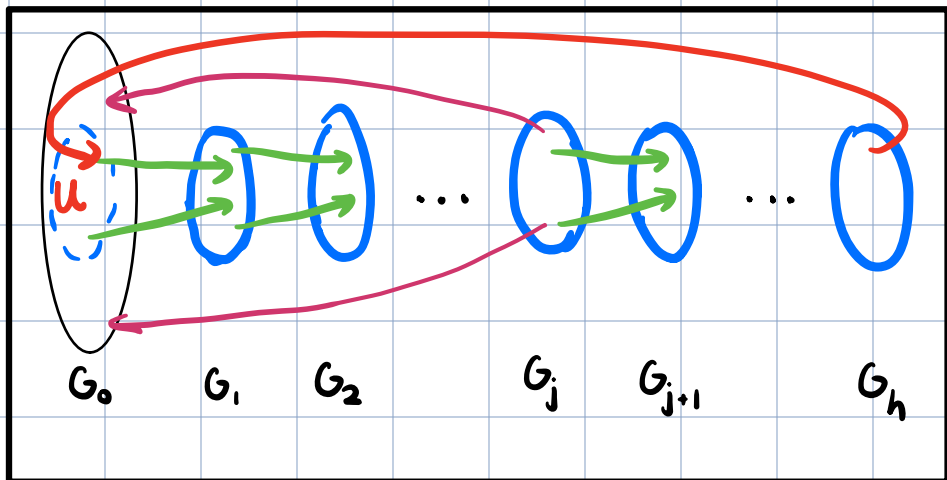
Solution: "exit edges"



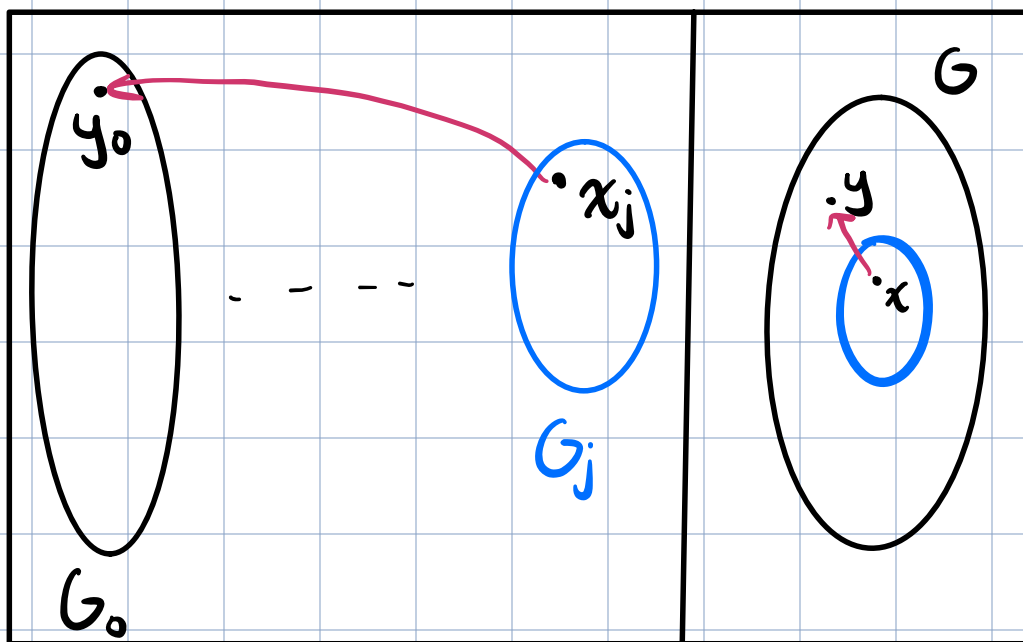
Reweight of exit edge is

$$l_p(x_j, y_0) = l(x, y) + d^j(V, x) - \cancel{d^0(V, y)} \\ \geq d^j(V, y) \geq 0$$



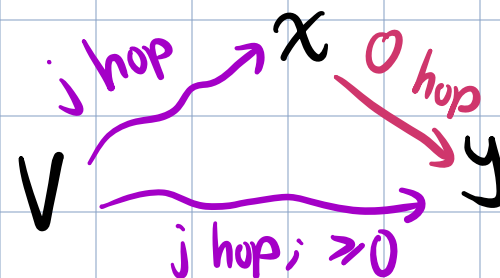


Solution: "exit edges"

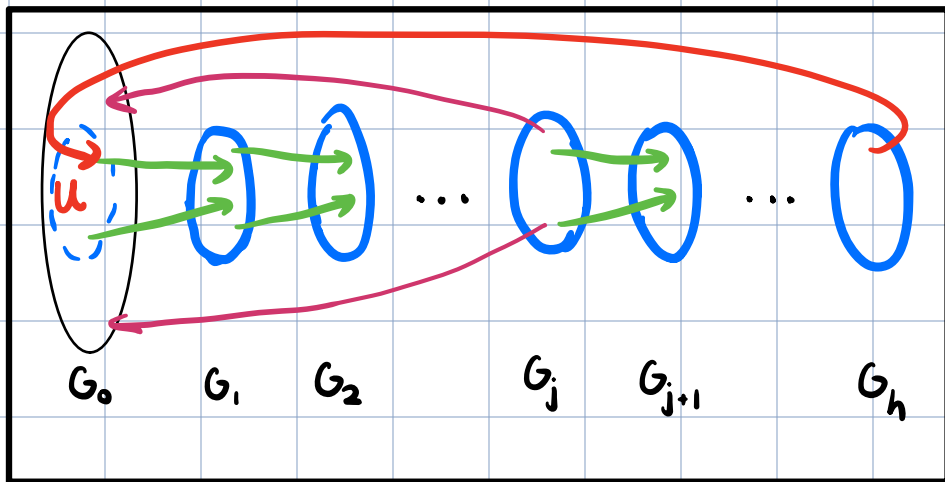


Reweight of exit edge is

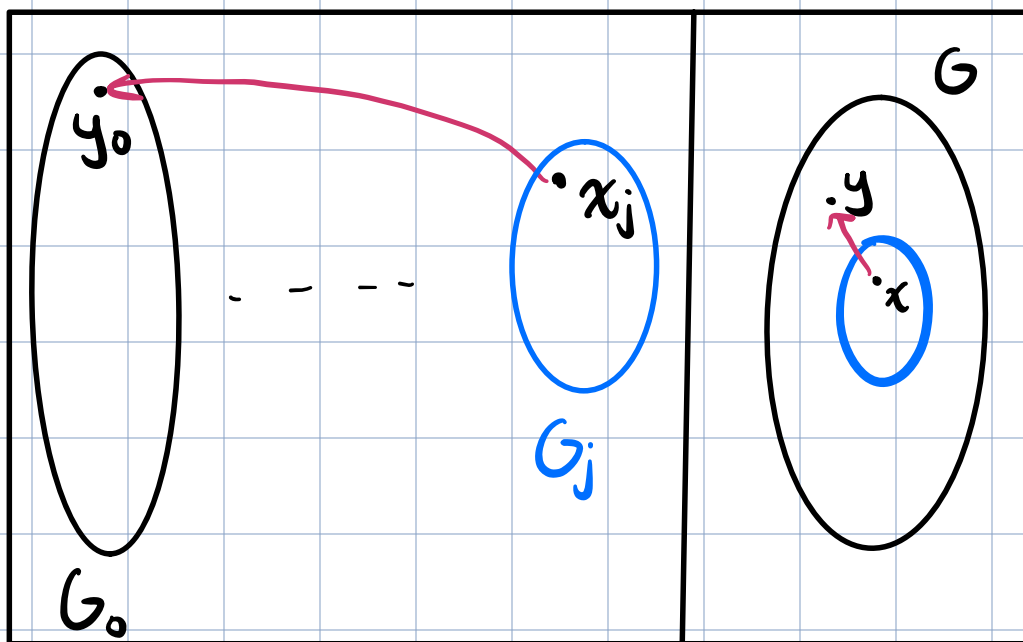
$$l_p(x_j, y_0) = l(x, y) + d^j(V, x) - \cancel{d^0(V, y)} \\ \geq d^j(V, y) \geq 0$$



All exit arcs are nonnegative

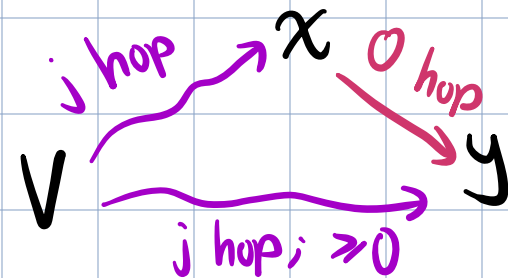


Solution: "exit edges"



Reweight of exit edge is

$$l_p(x_j, y_0) = l(x, y) + d^j(V, x) - \cancel{d^0(V, y)} \\ \geq d^j(V, y) \geq 0$$

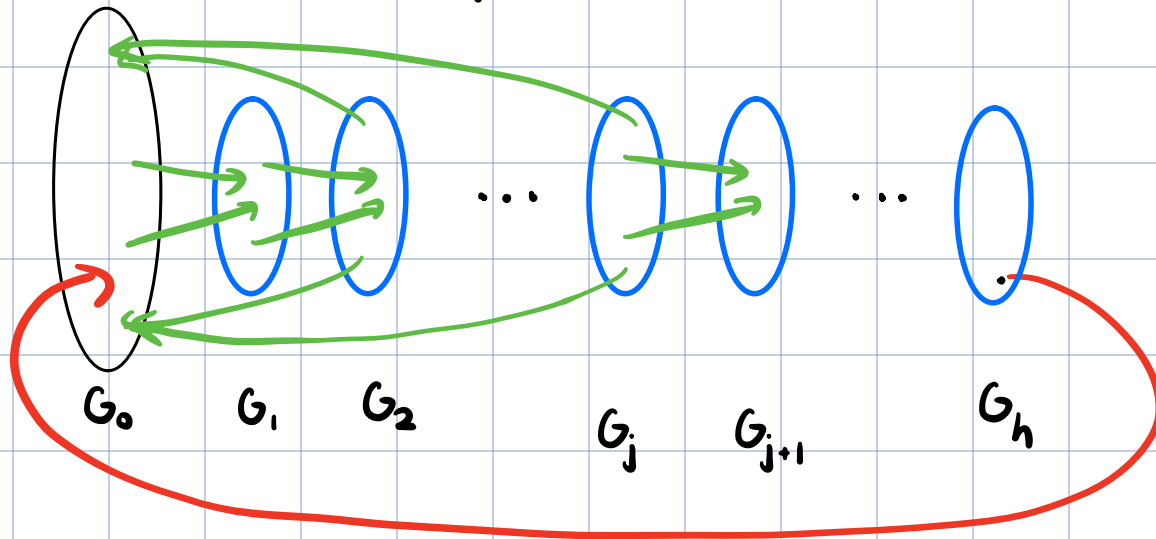


All exit arcs are
nonnegative
⇓

The only negative edges
are the "loopback" edges.

Hop Reducers, Again

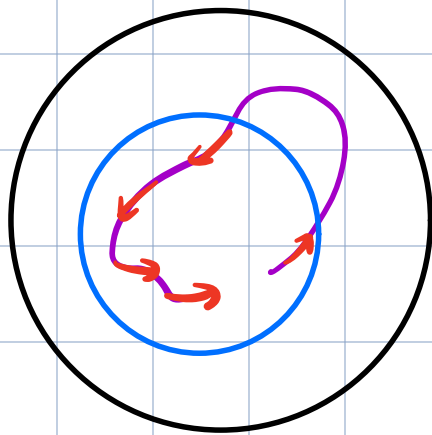
H_ψ



The only negative edges are the "loopback" edges.

Mapping Paths

W some η -hop walk in $G \implies W'$ in H_ψ using only $\frac{\eta}{h}$ hops



loop back every h hops

Hop Reducers, Again

Shortest walks in G

use $\leq |W|$ hops, so

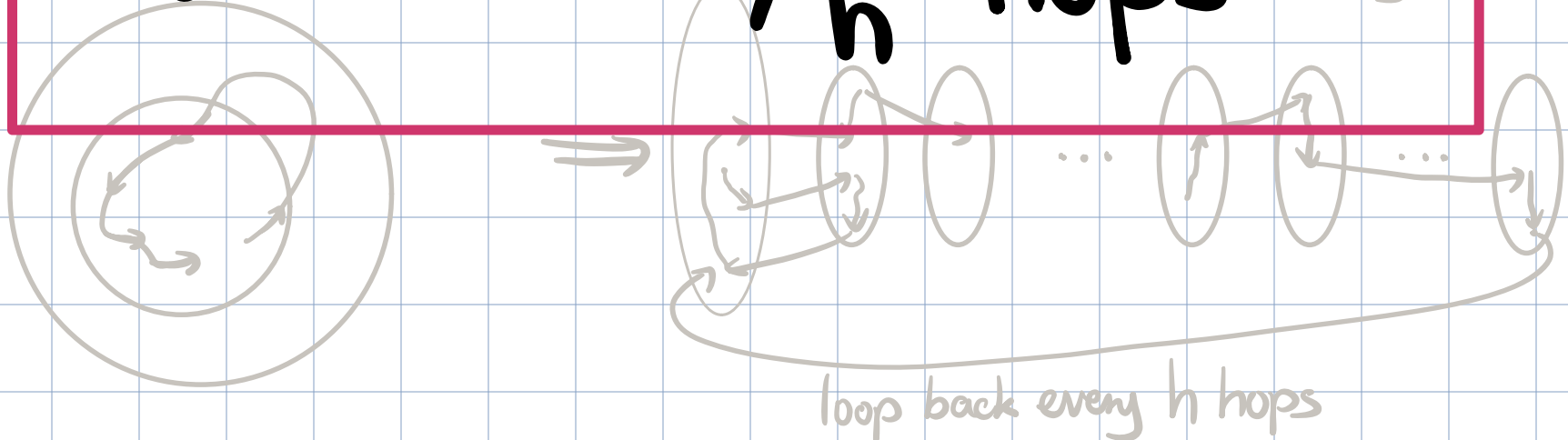
shortest walks in H_q

use $\leq |W|/h$ hops

The only negative edges are the "loopback" edges.

Mapping Paths

W some walk in $G \Rightarrow W$ in H_q using only $\frac{|W|}{h}$ hops



Hop Reducers, Again

Shortest walks in G

use $\leq |U|$ hops, so

shortest walks in H_φ

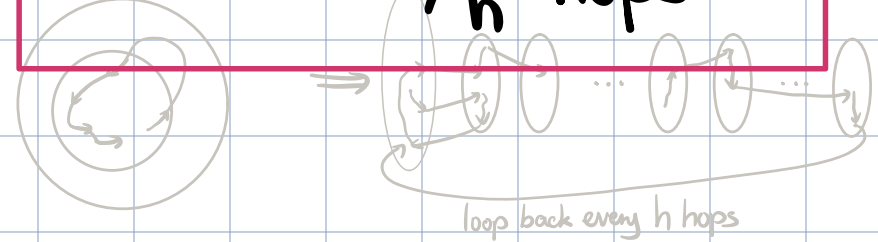
use $\leq |U|/h$ hops

The only negative edges
are the "loopback" edges.



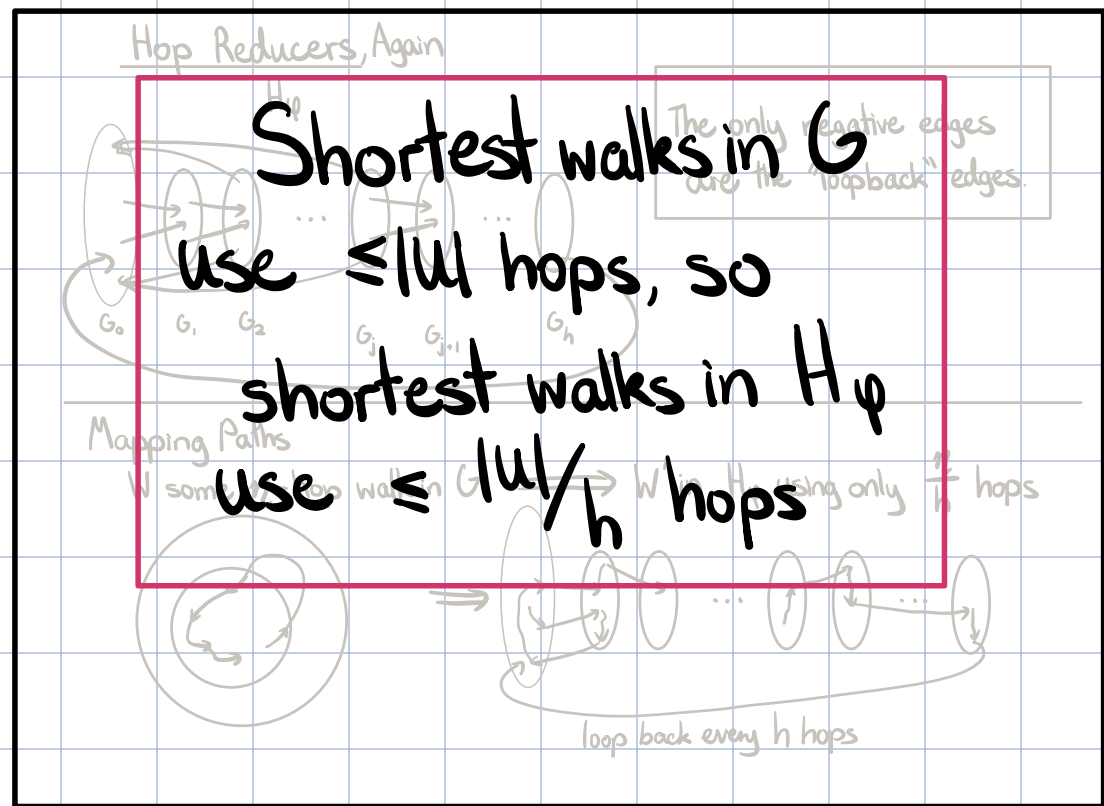
Mapping Paths

W some walk in G $\Rightarrow W$ in H using only $\frac{|U|}{h}$ hops



H_φ has $\tilde{O}(m)$ edges

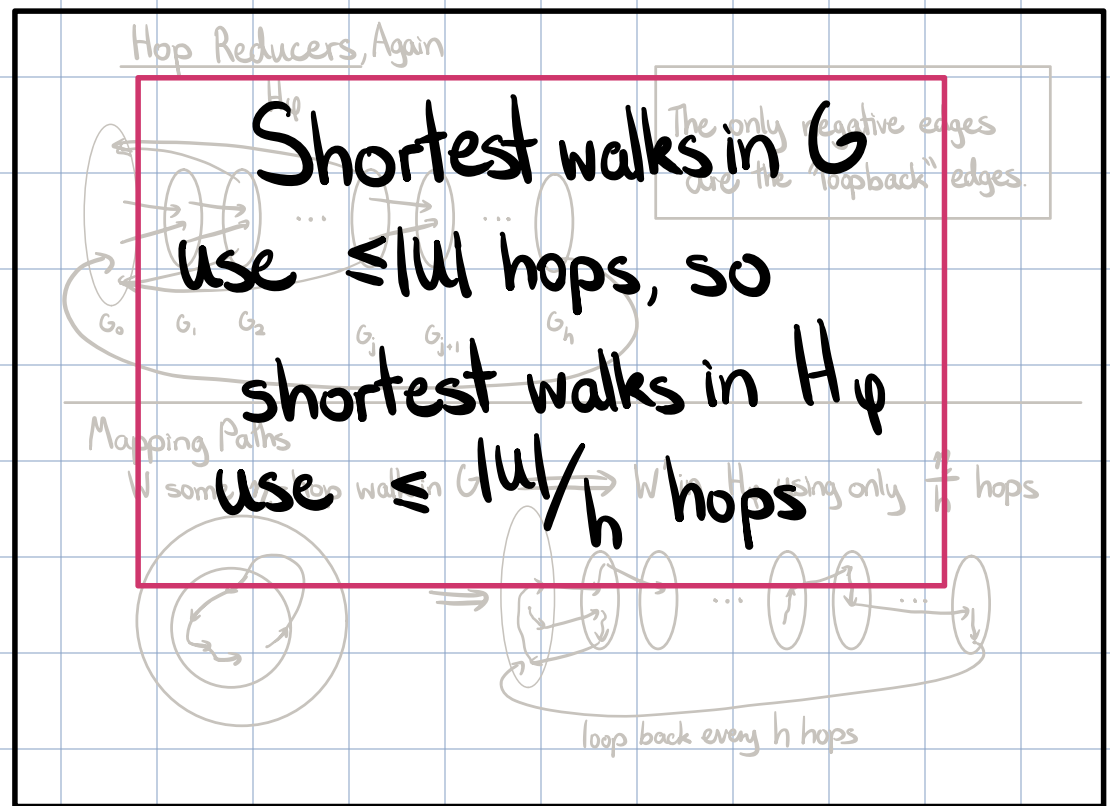
Make U nonneg in



H_φ has $\tilde{O}(m)$ edges

Make U nonneg in

$\tilde{O}(\frac{|U|}{h} m)$ time



H_φ has $\tilde{O}(m)$ edges

Make U nonneg in

$\tilde{O}(\frac{|U|}{h} m)$ time

$\tilde{O}(\frac{m}{h})$ per edge!

Hop Reducers, Again

Shortest walks in G

The only negative edges are the "loopback" edges.

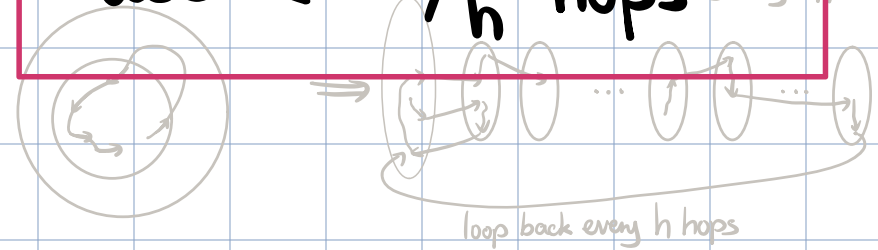
use $\leq |U|$ hops, so

shortest walks in H_φ

use $\leq |U|/h$ hops

Mapping Paths

W some $|U|$ hop walk in $G \Rightarrow W$ in H_φ using only $\frac{|U|}{h}$ hops

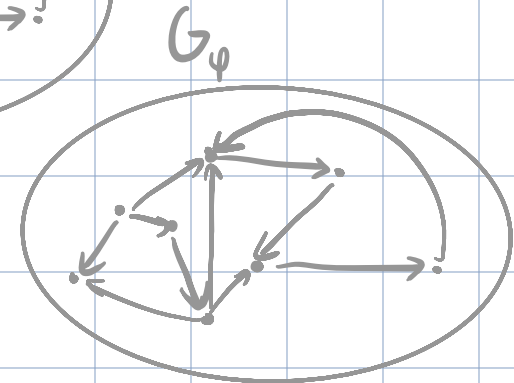
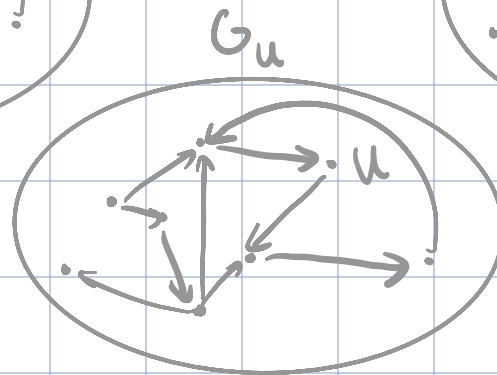
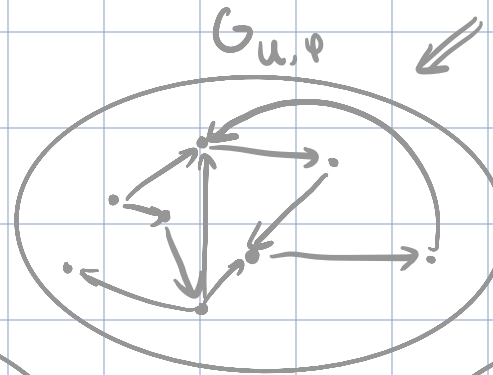
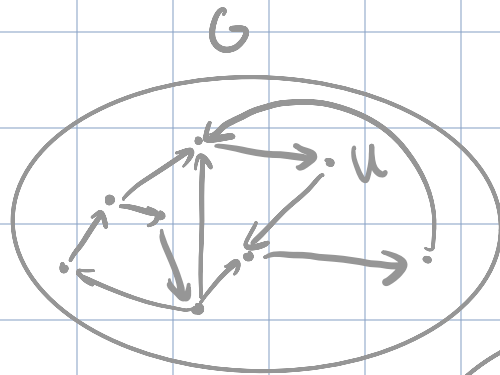


Final Algorithm

Incremental Elimination

Idea: Neutralize one edge set at a time

Neutralizing G
 Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,
 $\delta_e(\psi(u) = d(s, u), \psi(v) = d(s, v)) = (d(s, u) - d(s, v)) \geq 0$
 Distance (potentials) make
 all edges nonnegative
 $\psi = d(s, \cdot)$
 How to pick s ? $d(s^*, v) = d(V, v)$
 $= \min_{u \in V} d(u, v)$



Pick neg
edge set U

Remove other
neg. edges

Neutralize G_u
with $d^{(U)}(s, \cdot)$

Add edges
back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m|U|)$ per edge set $\times \frac{k}{|U|}$ edges = $\tilde{O}(km)$

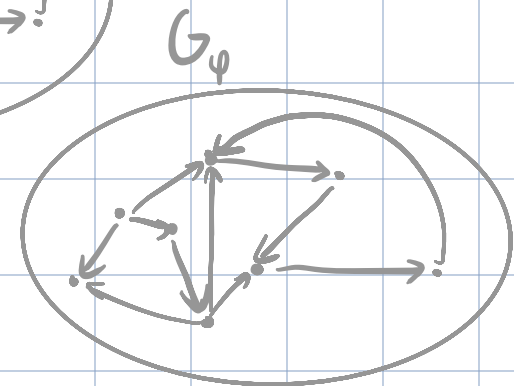
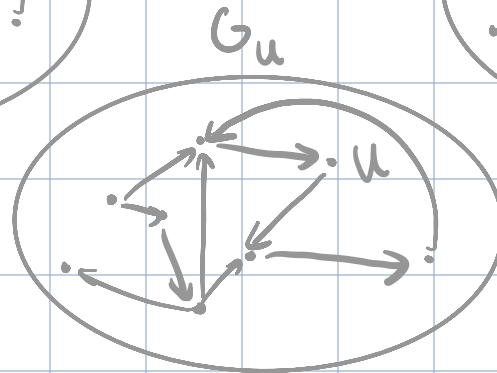
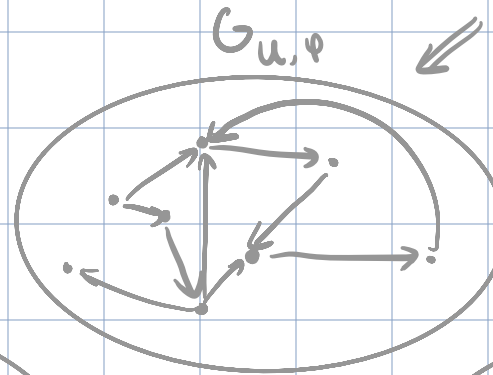
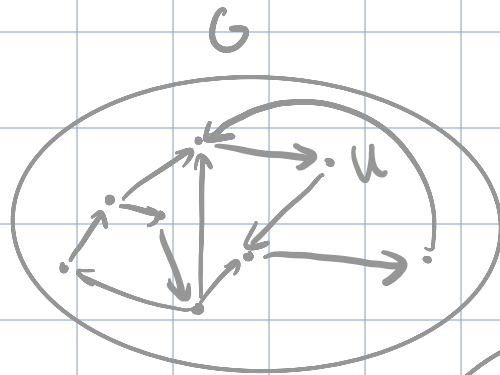


Final Algorithm

Incremental Elimination

Idea: Neutralize one edge set at a time

Neutralizing G
 Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,
 $\delta_e(\psi(u) - d(u, u) - \psi(v) + d(v, v)) = d(u, v) - d(s, u) - d(s, v) \geq 0$
 Distance (potentials) make all edges nonnegative
 $\psi = d(s, \cdot)$
 How to pick s ? $d(s^*, v) = d(v, v) = \min_{u \in V} d(u, v)$



Pick remote
neg edge set u

Remove other
neg. edges

Neutralize G_u
with $d^{(u)}(s, \cdot)$

Add edges
back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m|u|)$ per edge set $\times \frac{k}{|u|}$ edges = $\tilde{O}(km)$

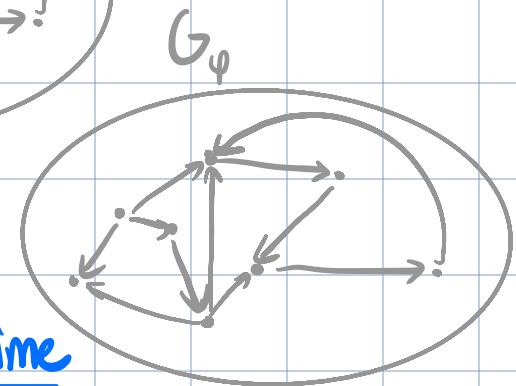
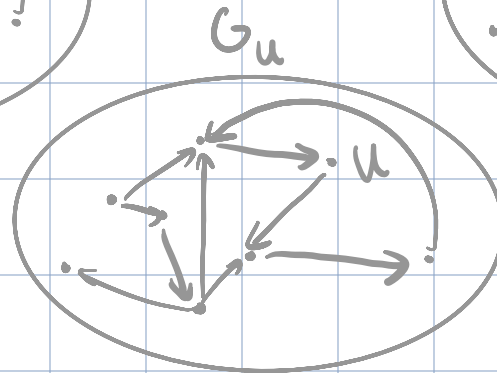
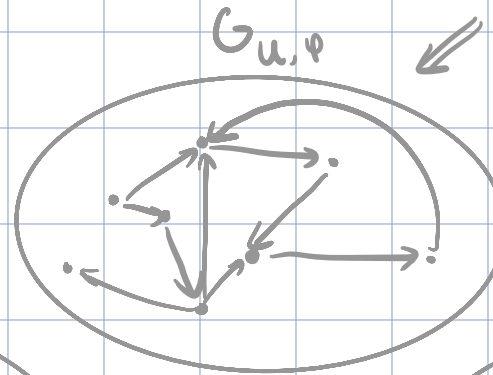
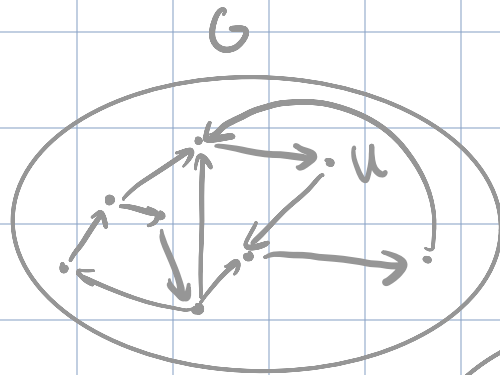


Final Algorithm

Incremental Elimination

Idea: Neutralize one edge set at a time

Neutralizing G
 Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,
 $\delta_e(\psi(u) = d(s, u), \psi(v) = d(s, v)) = (d(s, u) - d(s, v)) \geq 0$
 Distance (potentials) make
 all edges nonnegative
 $\psi = d(s, \cdot)$
 How to pick s ? $d(s^*, v) = d(V, v)$
 $= \min_{u \in V} d(u, v)$



Pick remote
neg edge set u

Remove other
neg. edges

In $\tilde{O}(\frac{m|u|}{h})$ time

Neutralize G_e
with $d^{(u)}(s, \cdot)$

Add edges
back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(m|u|)$ per edge set $\times \frac{k}{|u|}$ edges = $\tilde{O}(km)$

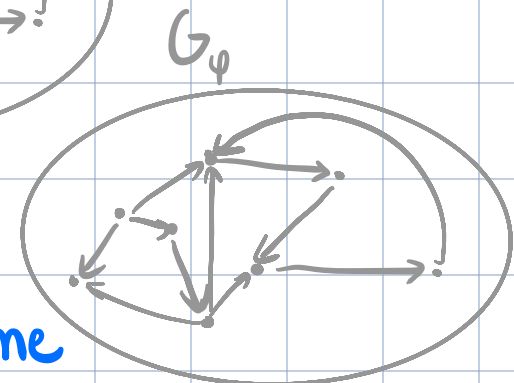
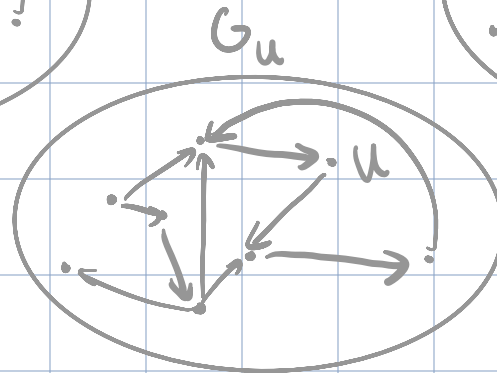
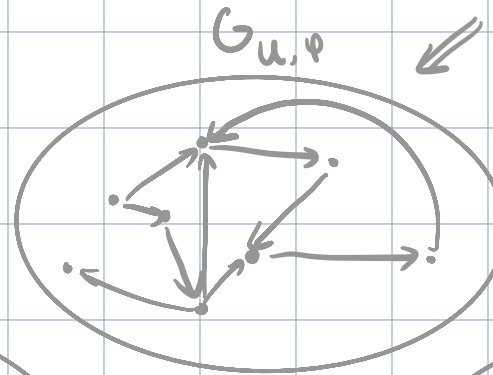
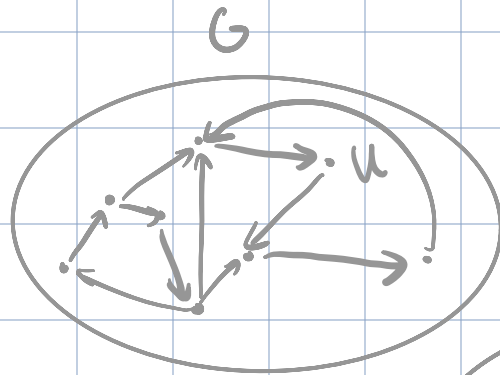


Final Algorithm

Incremental Elimination

Idea: Neutralize one edge set at a time

Neutralizing G
 Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,
 $\delta_e(\psi(u) = d(s, u), \psi(v) = d(s, v)) = (d(s, u) - d(s, v)) \geq 0$
 Distance (potentials) make
 all edges nonnegative
 $\psi = d(s, \cdot)$
 How to pick s ? $d(s^*, v) = d(V, v)$
 $= \min_{u \in V} d(u, v)$



Pick remote
neg edge set U

Remove other
neg. edges

In $\tilde{O}(m)$ time

Neutralize G_e
with $d^{(U)}(s, \cdot)$

Add edges
back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}\left(\frac{m|U|}{h}\right)$ per edge set $\times \frac{k}{|U|}$ edges sets $= \tilde{O}(km)$

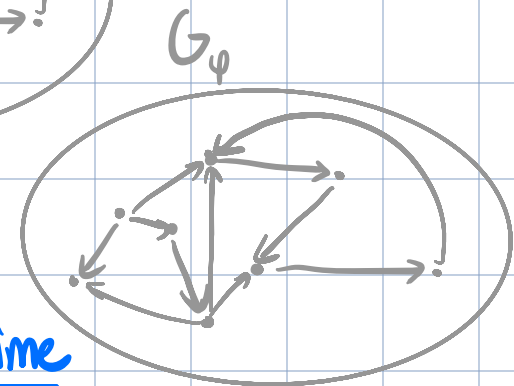
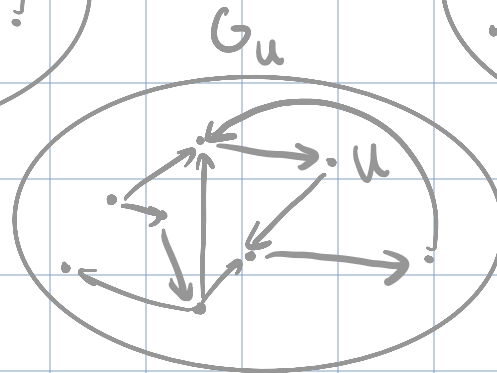
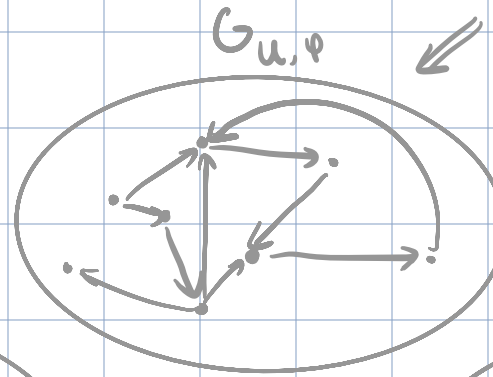
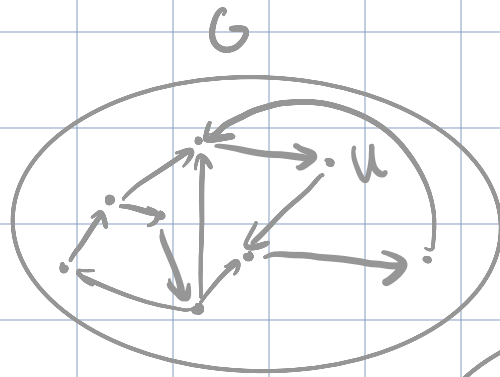


Final Algorithm

Incremental Elimination

Idea: Neutralize one edge set at a time

Neutralizing G
 Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,
 $\delta_e(\psi(u) = d(s, u), \psi(v) = d(s, v)) = (d(s, u) - d(s, v)) \geq 0$
 Distance (potentials) make all edges nonnegative
 $\psi = d(s, \cdot)$
 How to pick s ? $d(s^*, v) = d(V, v) = \min_{u \in V} d(u, v)$



Pick remote neg edge set U

Remove other neg. edges

In $\tilde{O}(\frac{m|U|}{h})$ time

Neutralize G_u with $d^{(U)}(s, \cdot)$

Add edges back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(\frac{m|U|}{h})$ per edge set $\times \frac{k}{|U|}$ edge sets $= \tilde{O}(km)$

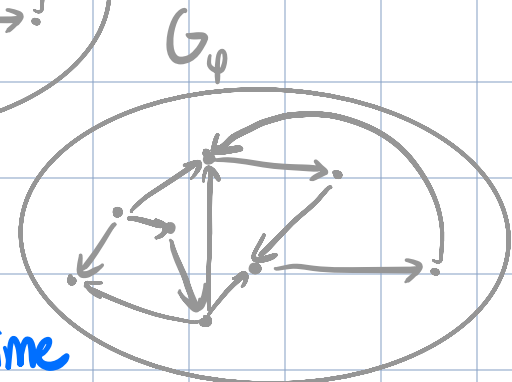
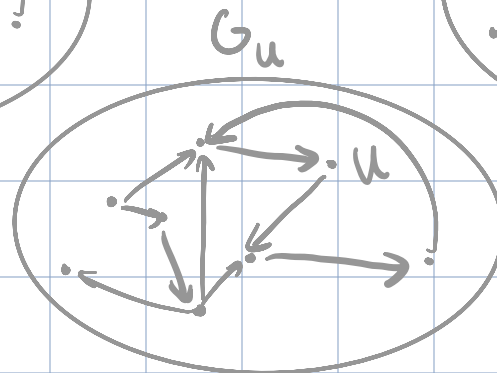
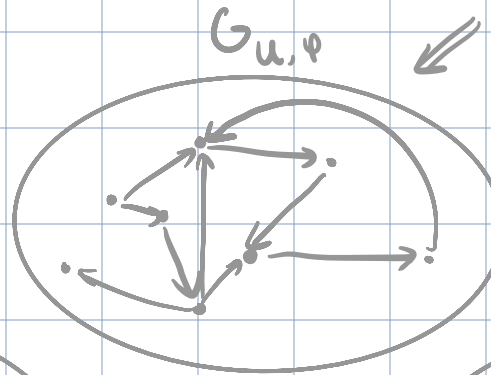
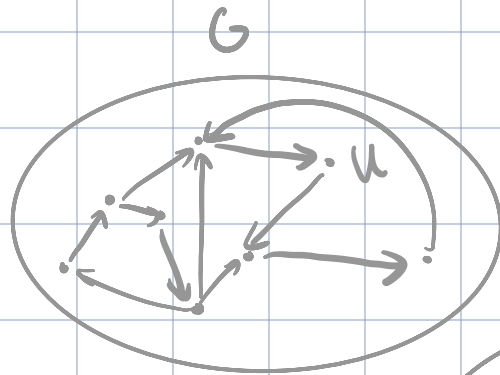


Final Algorithm

Incremental Elimination

Idea: Neutralize one edge set at a time

Neutralizing G
 Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,
 $\delta_{\psi}(u, v) = d(u, v) + \psi(u) - \psi(v) = d(u, v) + d(s, u) - d(s, v) \geq 0$
 Distance (potentials) make
 all edges nonnegative
 $\psi = d(s, \cdot)$
 How to pick s ? $d(s^*, v) = d(V, v)$
 $= \min_{u \in V} d(u, v)$



Pick remote
neg edge set U

Remove other
neg. edges

In $\tilde{O}(\frac{m|U|}{h})$ time

Neutralize G_U
with $d^{(U)}(s, \cdot)$

Add edges
back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(\frac{m|U|}{h})$ per edge set $\times \frac{k}{|U|}$ edge sets $= \tilde{O}(\frac{km}{h})$

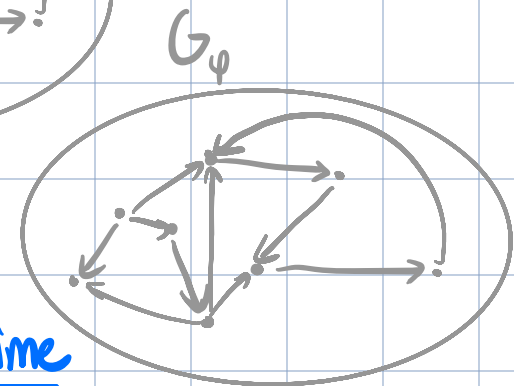
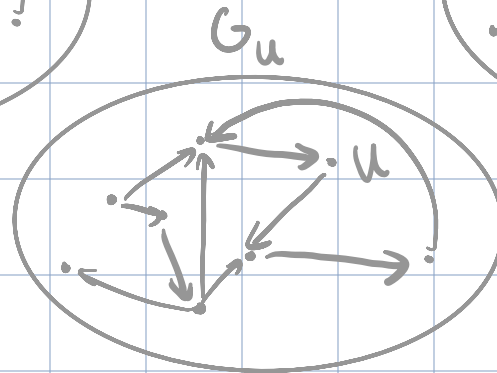
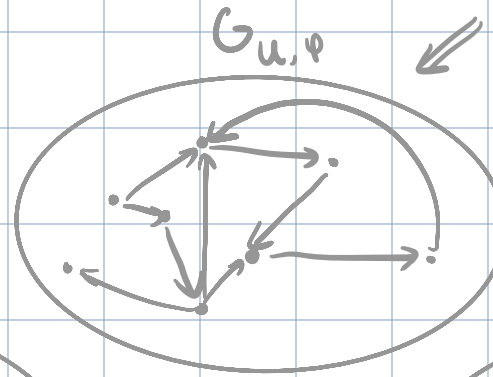
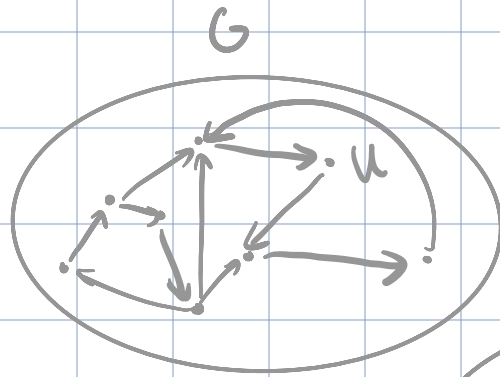


Final Algorithm

Incremental Elimination

Idea: Neutralize one edge set at a time

Neutralizing G
 Suppose some vertex s can reach all $v \in V$. Let $\psi(v) = d(s, v)$. Then,
 $\delta_{\psi}(u, v) = d(u, v) + \psi(u) - \psi(v) = d(u, v) + d(s, u) - d(s, v) \geq 0$
 Distance (potentials) make
 all edges nonnegative
 $\psi = d(s, \cdot)$
 How to pick s ? $d(s^*, v) = d(V, v)$
 $= \min_{u \in V} d(u, v)$



Pick remote
neg edge set U

Remove other
neg. edges

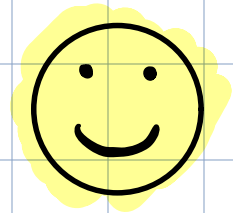
In $\tilde{O}(\frac{m|U|}{h})$ time

Neutralize G_U
with $d^{(U)}(s, \cdot)$

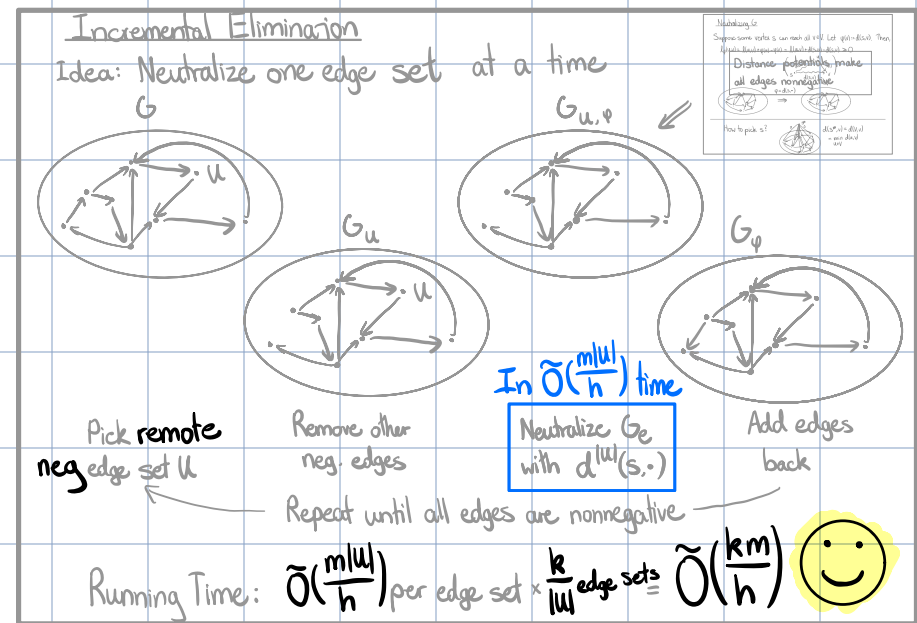
Add edges
back

Repeat until all edges are nonnegative

Running Time: $\tilde{O}(\frac{m|U|}{h})$ per edge set $\times \frac{k}{|U|}$ edge sets $= \tilde{O}(\frac{km}{h})$



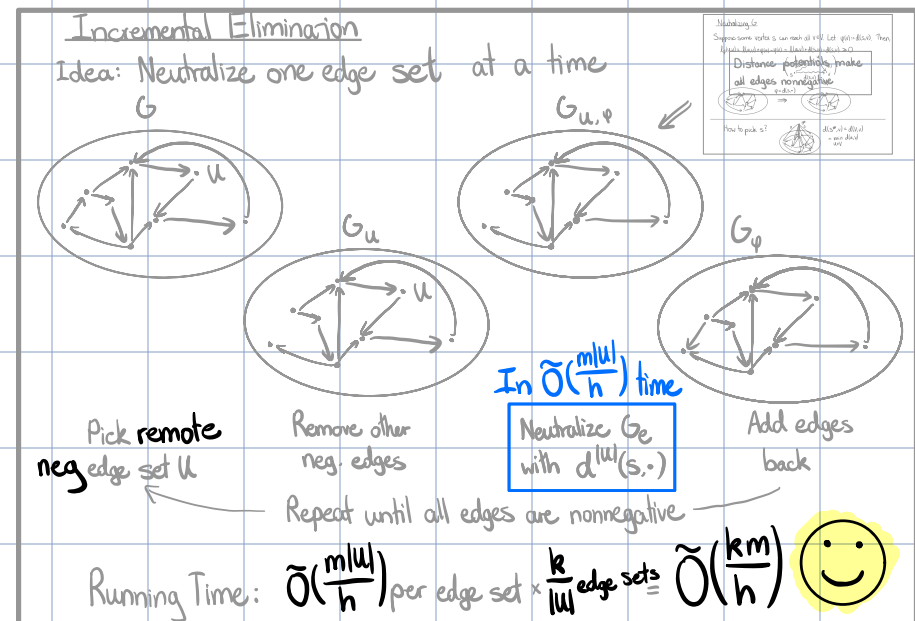
The Actual Running Time



I can find a big h -hop remote set



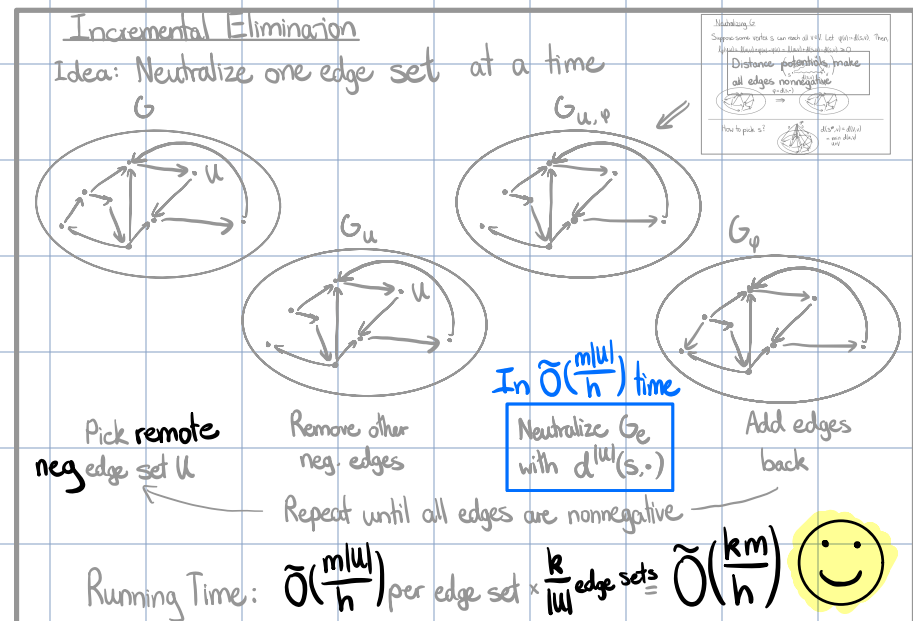
The Actual Running Time



I can find a big h -hop remote set of size $\sqrt[3]{k}$



The Actual Running Time



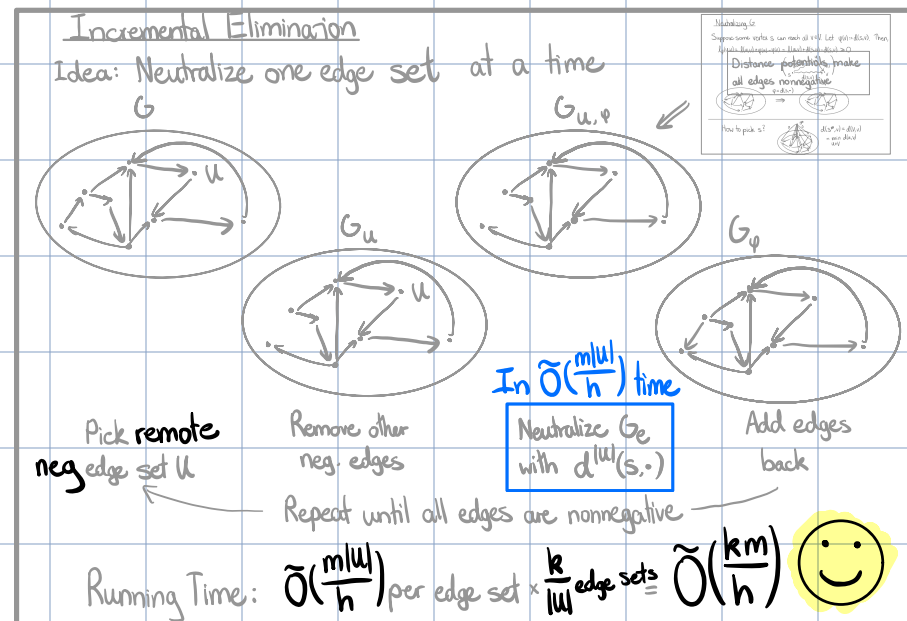
I can find a big h -hop remote set of size $\sqrt[3]{k}$ in $\tilde{O}(h^2 m)$ time



The Actual Running Time

Per iteration:

$$mh^2 + \frac{m|u|}{h}$$



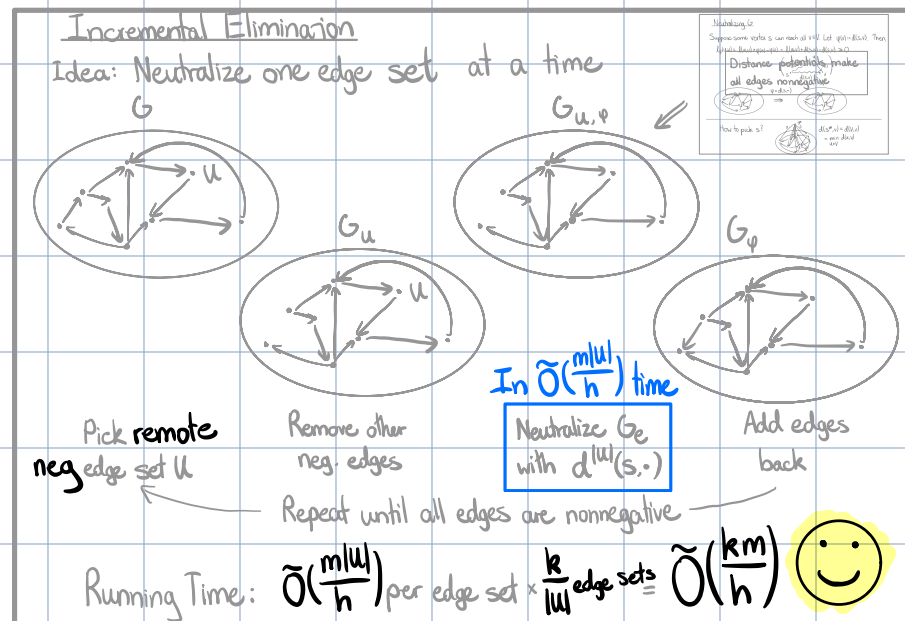
I can find a big h -hop remote set of size $\sqrt[3]{k}$ in $\tilde{O}(h^2 m)$ time



The Actual Running Time

Per iteration:

$$mh^2 + \frac{m|U|}{h} = mh^2 + \frac{m^3 \sqrt[3]{k}}{h}$$



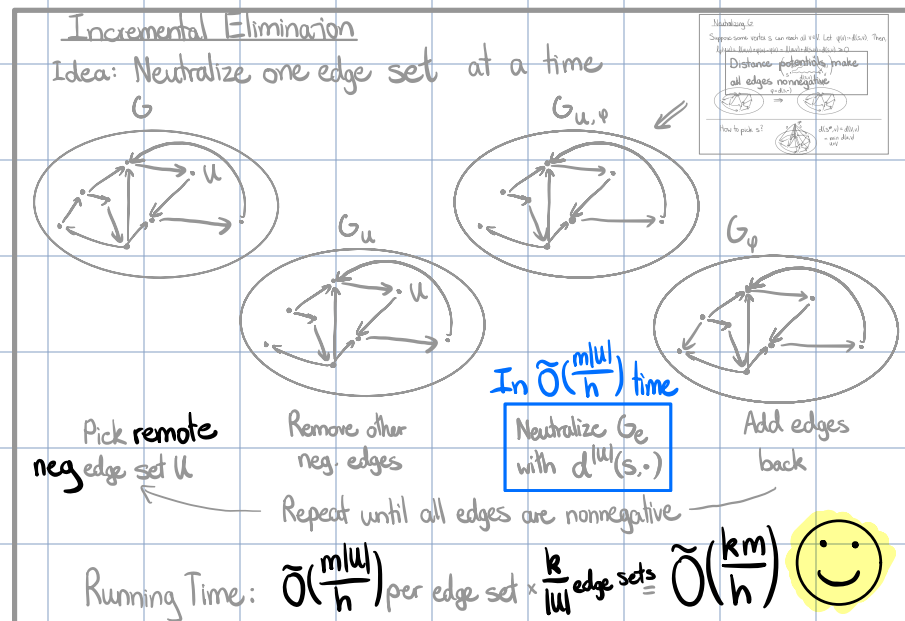
I can find a big h -hop remote set of size $\sqrt[3]{k}$ in $\tilde{O}(h^2 m)$ time



The Actual Running Time

Per iteration:

$$mh^2 + \frac{m|U|}{h} = mh^2 + \frac{m^3 \sqrt[3]{k}}{h}$$



I can find a big h -hop remote set of size $\sqrt[3]{k}$ in $\tilde{O}(h^2 m)$ time

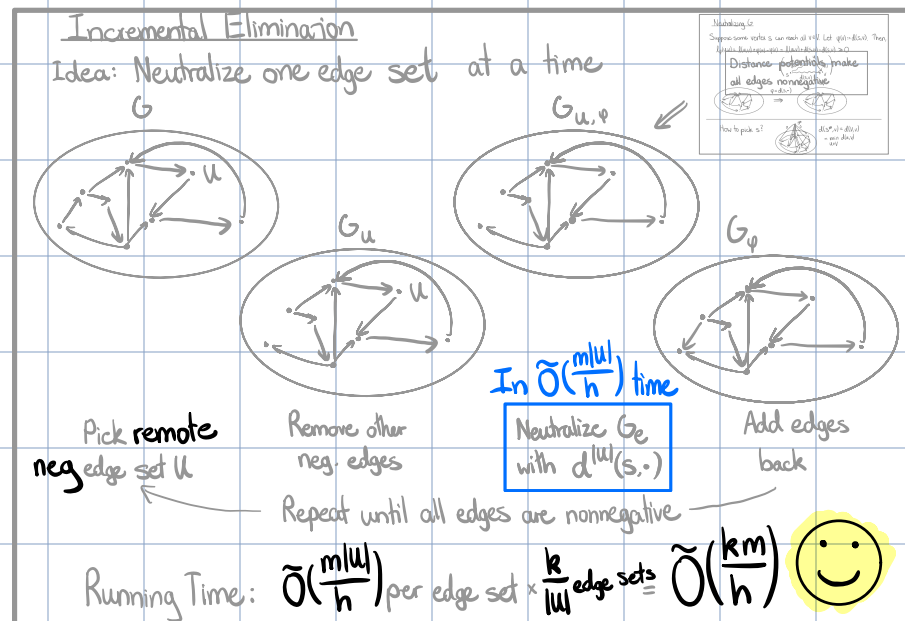


The Actual Running Time

Per iteration:

$$mh^2 + \frac{m|U|}{h} = mh^2 + \frac{m^3 \sqrt{k}}{h}$$

Make h as big as possible:



I can find a big h -hop remote set of size $\sqrt[3]{k}$ in $\tilde{O}(h^2 m)$ time

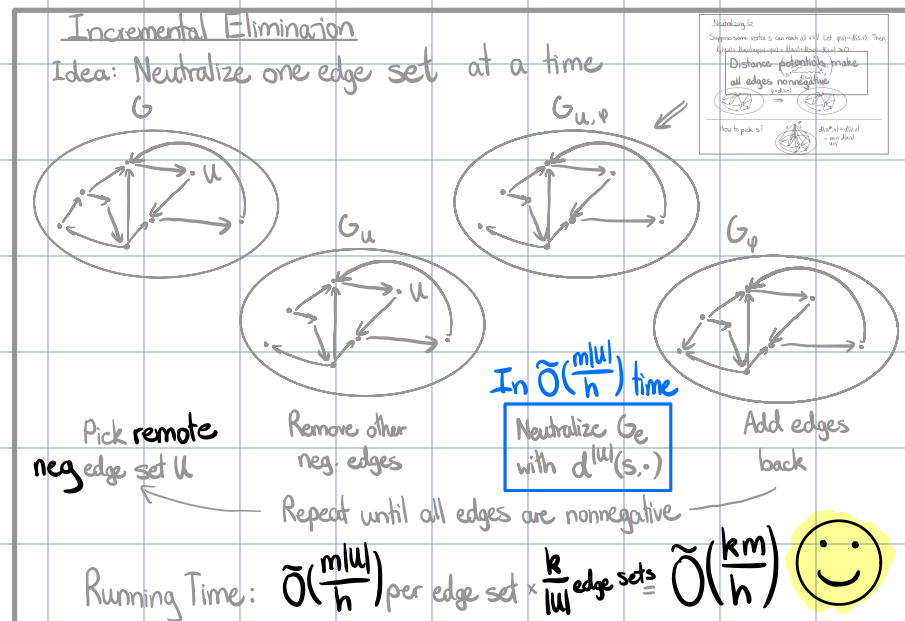


The Actual Running Time

Per iteration:

$$mh^2 + \frac{m|U|}{h} = mh^2 + \frac{m^{\frac{3}{4}}k}{h}$$

Make h as big as possible: $h = k^{\frac{1}{9}}$



I can find a big h -hop remote set of size $\sqrt[3]{k}$ in $\tilde{O}(h^2 m)$ time



The Actual Running Time

Per iteration:

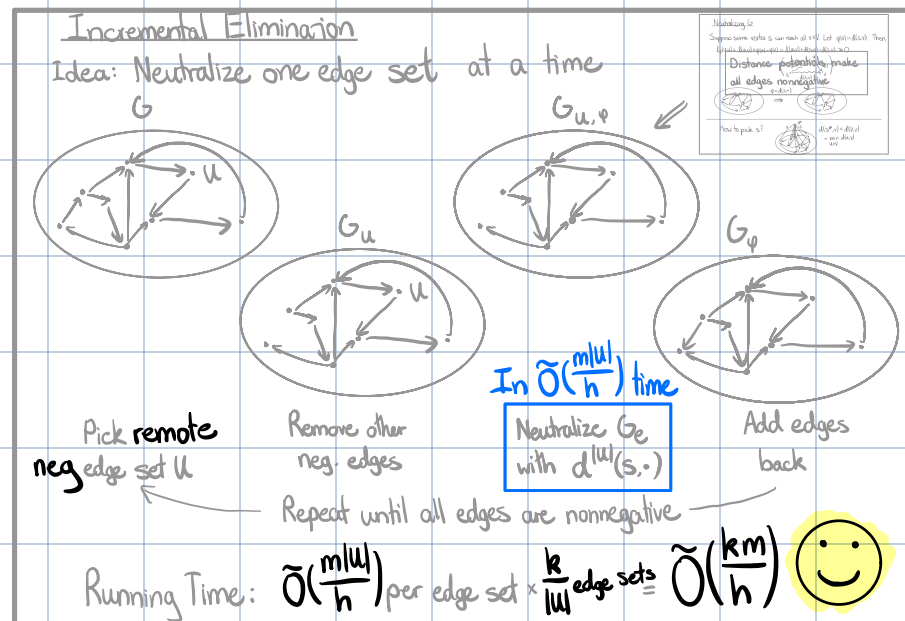
$$mh^2 + \frac{m|u|}{h} = mh^2 + \frac{m^{\frac{3}{4}}k}{h}$$

Make h as big as possible: $h = k^{\frac{1}{9}}$

Final Time

$$\tilde{O}(mk^{\frac{8}{9}})$$

I can find a big h -hop
remote set of size $\sqrt[3]{k}$
in $\tilde{O}(h^2 m)$ time



New Developments

[Fin24] $\tilde{O}(mk^{8/9})$

New Developments

$$[\text{Fin24}] \quad \tilde{O}(mk^{8/9})$$

$$[\text{HJQ25}] \quad \tilde{O}(mk^{4/5})$$

New Developments

[Fin24] $\tilde{O}(mk^{8/9})$

[HJQ25] $\tilde{O}(mk^{4/5})$

Bigger remote set

New Developments

[Fin24] $\tilde{O}(mk^{8/9})$

[HJQ25] $\tilde{O}(mk^{4/5})$ Bigger remote set

[HJQ26] $\tilde{O}(mk^{3/4} + m^{4/5}k)$

New Developments

[Fin24] $\tilde{O}(mk^{8/9})$

[HJQ25] $\tilde{O}(mk^{4/5})$ Bigger remote set

[HJQ26] $\tilde{O}(mk^{3/4} + m^{4/5}k)$

[QT26] $\tilde{O}(mk^{1/\sqrt{2}})$ [or $\tilde{O}(mk^{0.7})$]

New Developments

$$[\text{Fin24}] \quad \tilde{O}(mk^{8/9})$$

$$[\text{HJQ25}] \quad \tilde{O}(mk^{4/5}) \quad \text{Bigger remote set}$$

$$[\text{HJQ26}] \quad \tilde{O}(mk^{3/4} + m^{4/5}k)$$

$$[\text{QT26}] \quad \tilde{O}(mk^{1/\sqrt{2}}) \quad [\text{or } \tilde{O}(mk^{0.7})]$$

$$[\text{LLRZ26}] \quad \tilde{O}(n^{2.5})$$

New Developments

$$[\text{Fin24}] \quad \tilde{O}(mk^{8/9})$$

$$[\text{HJQ25}] \quad \tilde{O}(mk^{4/5}) \quad \text{Bigger remote set}$$

$$[\text{HJQ26}] \quad \tilde{O}(mk^{3/4} + m^{4/5}k)$$

$$[\text{QT26}] \quad \tilde{O}(mk^{1/\sqrt{2}}) \quad [\text{or } \tilde{O}(mk^{0.7})]$$

$$[\text{LLRZ26}] \quad \tilde{O}(n^{2.5})$$

$$[\text{LLZ26}] \quad \tilde{O}(n^{2+o(1)})$$