Linear time computation for Minimum Spanning Trees

Ian Chen

Problem Statement

- Input: $G = (V, E)$, undirected graph with distinct weights.
- Output: $T \subset G$ spanning tree minimizing the sum of all edge weights

Heavy and light edges (1/2)

- An edge is *light* if it is the minimum weight edge across some bipartition
- An edge is *heavy* if it is the heaviest weight edge in a cycle

Heavy and light edges (1/2)

- An edge is *light* if it is the minimum weight edge across some bipartition
- An edge is *heavy* if it is the heaviest weight edge in a cycle

How are these related to minimum spanning trees?

Heavy and light edges (2/2)

Algorithm	Light	Heavy
Kruskals	Globally light edge not making cycle	Adding an edge forms a cycle
Prims	Lightest edge crossing frontier	
Boruvkas	Lightest edge on partition boundaries	
Chazelle	Approximate lightest edge crossing frontier	Adding an edge forms cycle
Karger-Klein-Tarjan	Lightest edges on partition boundaries	Detect heavy edges wrt a subgraph

Table: Heavy and light edge classification for selected MST algorithms

Preliminaries

Verifying an MST

Finding an MST

MST Verification

Theorem

Let T be a spanning tree of G. T is the (unique) MST if and only if for all edges $uv \in G$

$$w(uv) \geq \left\|\pi_{uv}^{T}\right\|_{\infty} \tag{1}$$

where π_{uv}^{T} is the unique path betwen u and v in T, and $\|\cdot\|_{\infty}$ is the heaviest edge in the subgraph.

MST Verification

- How do we check whether Equation 1 holds?

MST Verification

- How do we check whether Equation 1 holds?
  - Reduce the problem to a full-branching tree
  - Binary search and clever dynamic programming

Boruvka Trees

- Vertices in level i of the Boruvka tree is the partition of G after i Boruvka steps
- Edges are between adjacent levels whoose vertices are not disjoint (take the heaviest edge)

Boruvka Trees

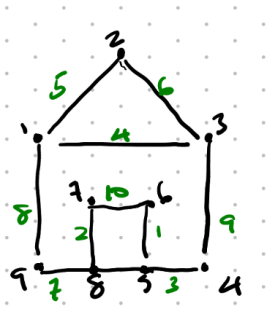

Figure: Example of Boruvka Tree construction (a), the input graph

Boruvka Trees

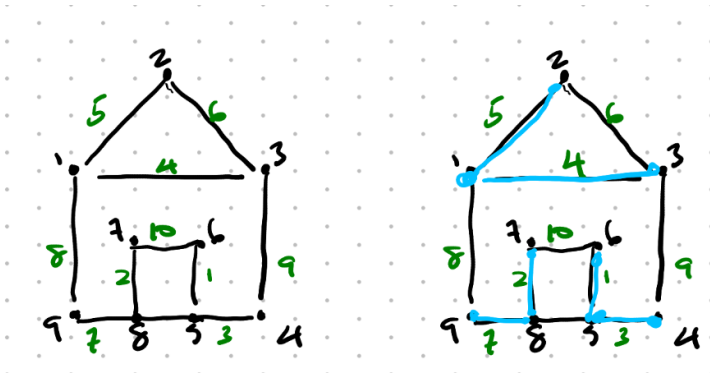

Figure: Example of Boruvka Tree construction (b), one step of Boruvka

Boruvka Trees

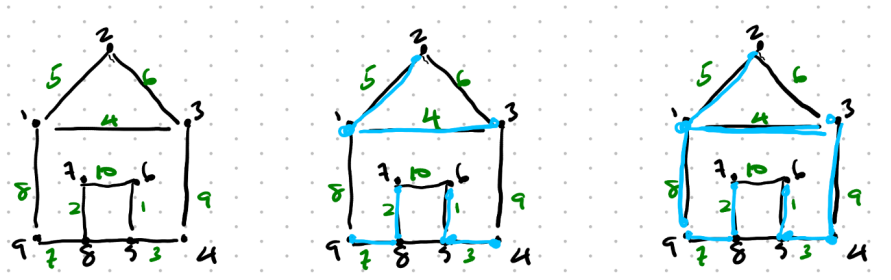

Figure: Example of Boruvka Tree construction (c), two step of Boruvka. Boruvka finds the minimum spanning tree.

Boruvka Trees

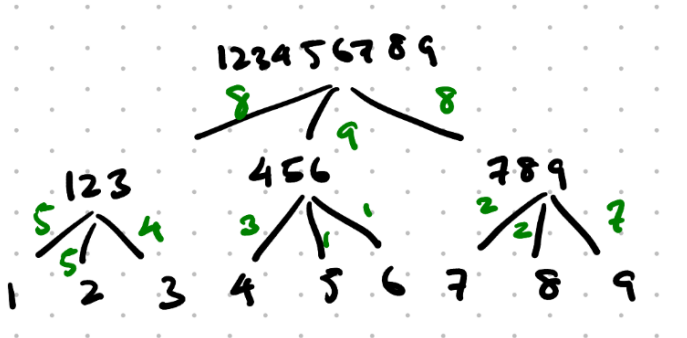

Figure: Example of Boruvka Tree construction (d), the witness tree.

Boruvka Trees

- All internal nodes have ≥ 2 children
- All leaves are at same depth
- $\left\|\pi^T_{uv}\right\|_\infty = \left\|\pi^B_{uv}\right\|_\infty$

Querying the Boruvka Tree

- We have m queries (one for each edge in G)
- Denote $\pi_{ij}^B\big|_v$ be the path π_{ij}^B restricted to $v \rightsquigarrow B.root$.
- Let $f_v : (i,j) \mapsto \left\| \pi_{ij}^B\big|_v \right\|_\infty$ for all (i,j) query.
- f_v is updated from $f_{v.parent}$
  - Need to update all paths with weight less than $w(v, v.parent)$
  - Keep f_v sorted, binary search to determine which paths update!

Querying the Boruvka Tree

- We have m queries (one for each edge in G)
- Denote $\pi_{ij}^B\big|_v$ be the path π_{ij}^B restricted to $v \rightsquigarrow B.\mathit{root}$.
- Let $f_v : (i,j) \mapsto \left\| \pi_{ij}^B\big|_v \right\|_\infty$ for all (i,j) query.
- f_v is updated from $f_{v.\mathit{parent}}$
  - Need to update all paths with weight less than $w(v, v.\mathit{parent})$
  - Keep f_v sorted, binary search to determine which paths update!

Querying the Boruvka Tree

- We have m queries (one for each edge in G)
- Denote $\pi_{ij}^B\big|_v$ be the path π_{ij}^B restricted to $v \rightsquigarrow B.\mathit{root}$.
- Let $f_v : (i,j) \mapsto \left\|\pi_{ij}^B\big|_v\right\|_\infty$ for all (i,j) query.
- f_v is updated from $f_{v.\mathit{parent}}$
  - Need to update all paths with weight less than $w(v, v.\mathit{parent})$
  - Keep f_v sorted, binary search to determine which paths update!

Querying the Boruvka Tree

- We have m queries (one for each edge in G)
- Denote $\pi_{ij}^B\big|_v$ be the path π_{ij}^B restricted to $v \rightsquigarrow B.\mathit{root}$.
- Let $f_v : (i,j) \mapsto \left\| \pi_{ij}^B\big|_v \right\|_\infty$ for all (i,j) query.
- f_v is updated from $f_{v.\mathit{parent}}$
  - Need to update all paths with weight less than $w(v, v.\mathit{parent})$
  - Keep f_v sorted, binary search to determine which paths update!

Querying the Boruvka Tree

- We have m queries (one for each edge in G)
- Denote $\pi_{ij}^B\big|_v$ be the path π_{ij}^B restricted to $v \rightsquigarrow B.\mathit{root}$.
- Let $f_v : (i,j) \mapsto \left\| \pi_{ij}^B\big|_v \right\|_\infty$ for all (i,j) query.
- f_v is updated from $f_{v.\mathit{parent}}$
  - Need to update all paths with weight less than $w(v, v.\mathit{parent})$
  - Keep f_v sorted, binary search to determine which paths update!

Analysis of MST Verification

- Number of comparisons is bounded by $\sum_v \log(|f_v| + 1)$
- L_i vertices on level i, $\ell_i = |L_i|$, so f_v are disjoint for $v \in L_i$
- The log function is convex, so apply Jensen's inequality

$$\sum_{v \in L_i} \log(|f_v| + 1) \leq \ell_i \log\left(1 + \frac{1}{\ell_i} \sum_{v \in L_i} |f_v|\right) \leq \ell_i \log\left(1 + \frac{m}{\ell_i}\right)$$

- Summing over all levels,

$$\sum_i \ell_i \log\left(1 + \frac{m}{\ell_i}\right) \leq 3n + n \log\left(1 + \frac{m}{n}\right) \leq 4n + m$$

Analysis of MST Verification

- Number of comparisons is bounded by $\sum_v \log(|f_v| + 1)$
- L_i vertices on level i, $\ell_i = |L_i|$, so f_v are disjoint for $v \in L_i$
- The log function is convex, so apply Jensen's inequality

$$\sum_{v \in L_i} \log(|f_v| + 1) \leq \ell_i \log\left(1 + \frac{1}{\ell_i} \sum_{v \in L_i} |f_v|\right) \leq \ell_i \log\left(1 + \frac{m}{\ell_i}\right)$$

- Summing over all levels,

$$\sum_i \ell_i \log\left(1 + \frac{m}{\ell_i}\right) \leq 3n + n \log\left(1 + \frac{m}{n}\right) \leq 4n + m$$

Analysis of MST Verification

- Number of comparisons is bounded by $\sum_v \log(|f_v| + 1)$
- L_i vertices on level i, $\ell_i = |L_i|$, so f_v are disjoint for $v \in L_i$
- The log function is convex, so apply Jensen's inequality

$$\sum_{v \in L_i} \log(|f_v| + 1) \leq \ell_i \log\left(1 + \frac{1}{\ell_i} \sum_{v \in L_i} |f_v|\right) \leq \ell_i \log\left(1 + \frac{m}{\ell_i}\right)$$

- Summing over all levels,

$$\sum_i \ell_i \log\left(1 + \frac{m}{\ell_i}\right) \leq 3n + n \log\left(1 + \frac{m}{n}\right) \leq 4n + m$$

Analysis of MST Verification

- Number of comparisons is bounded by $\sum_v \log(|f_v| + 1)$
- L_i vertices on level i, $\ell_i = |L_i|$, so f_v are disjoint for $v \in L_i$
- The log function is convex, so apply Jensen's inequality

$$\sum_{v \in L_i} \log(|f_v| + 1) \leq \ell_i \log\left(1 + \frac{1}{\ell_i} \sum_{v \in L_i} |f_v|\right) \leq \ell_i \log\left(1 + \frac{m}{\ell_i}\right)$$

- Summing over all levels,

$$\sum_i \ell_i \log\left(1 + \frac{m}{\ell_i}\right) \leq 3n + n \log\left(1 + \frac{m}{n}\right) \leq 4n + m$$

Preliminaries

Verifying an MST

Finding an MST

Strategy

- Sufficient to find a linear time algorithm that reduces the number of edges/nodes by constant factor.
- To reduce the number of nodes, can run Boruvka iterations
- To reduce the number of edges, (???)

Strategy

- Sufficient to find a linear time algorithm that reduces the number of edges/nodes by constant factor.
- To reduce the number of nodes, can run Boruvka iterations
- To reduce the number of edges, (???)

Strategy

- If e is heavy in a $H \subset G$, then e is heavy in G.
- We say e is H-heavy if e is heavy in $H + e$ (otherwise, e is H-light).
- What subgraph is easy to detect H-heavy edges?

Sampling lemma

Lemma

Let H be constructed by picking each edge from G with probability $1/2$. Let F be the minimum spanning forest of H. Then, in expectation, G has at most $2n$ F-light edges.

Sampling lemma

Proof.

- Label the edges e_i by increasing weight, and let H_i be the subgraph $H \cap \{ e_j \,|\, j \leq i \}$ and F_i be the MSF of H_i.
- If e_i is F-light, then $e_i = F_i \setminus F_{i-1}$ with probability $1/2$.
- The number of edges in F is less than n
- Therefore, in expectation G has at most $2n$ F-light edges.

□

Sampling lemma

Proof.

- Label the edges e_i by increasing weight, and let H_i be the subgraph $H \cap \{ e_j \,|\, j \leq i \}$ and F_i be the MSF of H_i.
- If e_i is F-light, then $e_i = F_i \setminus F_{i-1}$ with probability $1/2$.
- The number of edges in F is less than n
- Therefore, in expectation G has at most $2n$ F-light edges.

□

Sampling lemma

Proof.

- Label the edges e_i by increasing weight, and let H_i be the subgraph $H \cap \{ e_j \,|\, j \leq i \}$ and F_i be the MSF of H_i.
- If e_i is F-light, then $e_i = F_i \setminus F_{i-1}$ with probability $1/2$.
- The number of edges in F is less than n
- Therefore, in expectation G has at most $2n$ F-light edges.

□

Sampling lemma

Proof.

- Label the edges e_i by increasing weight, and let H_i be the subgraph $H \cap \{ e_j \,|\, j \leq i \}$ and F_i be the MSF of H_i.
- If e_i is F-light, then $e_i = F_i \setminus F_{i-1}$ with probability $1/2$.
- The number of edges in F is less than n
- Therefore, in expectation G has at most $2n$ F-light edges.

□

KKT algorithm

1. If $|G| = 1$, uncontract any components and return the MST.
2. $G' \leftarrow G$ after 2 Boruvka iterations
3. Subsample a graph $H \subset G'$ with probability $1/2$
4. Find the MSF F of H
5. $G \leftarrow G'$ after removing all F-heavy edges
6. Goto step 1

Analysis

Theorem

Let $T(n, m)$ be the runtime of the KKT algorithm. Then, $\mathbf{E}(T(n, m)) = O(n + m)$.

Proof.

$$T(n, m) = T(n/4, \tilde{m}_1) + T(n/4, \tilde{m}_2) + O(n + m)$$

where $\tilde{m}_1 = |E(H)|$, $\tilde{m}_2$ is the number of F-light edges. Observe that $\mathbf{E}(\tilde{m}_1) = m/2$ and $\mathbf{E}(\tilde{m}_2) \leq 2 \cdot n/4 = n/2$.

□

Analysis

Theorem

Let $T(n, m)$ be the runtime of the KKT algorithm. Then, $\mathbf{E}(T(n, m)) = O(n + m)$.

Proof.

Supposing that $\mathbf{E}(T(n', m'))$ is linear from $n' < n$ and $m' < m$, then

$$\begin{aligned}
\mathbf{E}(T(n, m)) &= \mathbf{E}(T(n/4, \tilde{m}_1)) + \mathbf{E}(T(n/4, \tilde{m}_2)) + O(n + m) \\
&= \mathbf{E}(T(n/4, \mathbf{E}(\tilde{m}_1))) + \mathbf{E}(T(n/4, \mathbf{E}(\tilde{m}_2))) + O(n + m) \\
&= \mathbf{E}(T(n/4, m/2)) + \mathbf{E}(T(n/4, n/2)) + O(n + m) \\
&= O(n + m)
\end{aligned}$$

□

Bibliography I

- David R. Karger, Philip N. Klein, and Robert E. Tarjan. 1995. A randomized linear-time algorithm to find minimum spanning trees. J. ACM 42, 2 (March 1995), 321–328. https://doi.org/10.1145/201019.201022
- Komlós, J. Linear verification for spanning trees. Combinatorica 5, 57–65 (1985). https://doi.org/10.1007/BF02579443
- King, V. (1995). A simpler minimum spanning tree verification algorithm. In: Akl, S.G., Dehne, F., Sack, JR., Santoro, N. (eds) Algorithms and Data Structures. WADS 1995. Lecture Notes in Computer Science, vol 955. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-60220-8_83