Faster MST Algorithms

Using Boruvka Step

Liwen Ouyang

Department of Computer Science
University of Massachusetts Amherst

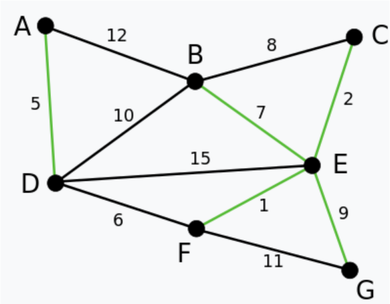
Apr 28 2020

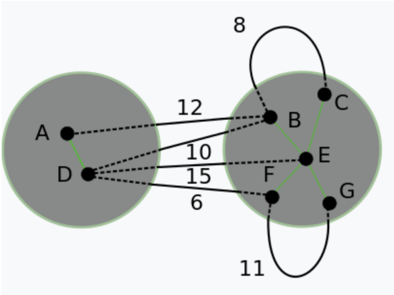
Outline

Boruvka Step

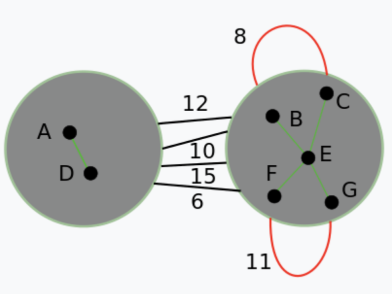
- Input: a weighted undirected graph $G = \{V, E\}$
- 1. For each $v \in V$, we select an edge e with the smallest weight incident to v, and thus get a subgraph G_1
  2. Create a multigraph G_m by contracting all nodes that are connected by the edges we selected in 1 into supernodes.
  3. Get a contracted graph G' by deleting all self loops and non-minimal repetitive edges in G_m
- Output: the set of edges selected in 1 and a contracted graph G'

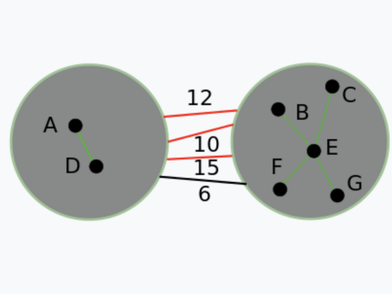
An Example





An Example





Key Points

- The edges we choose in 1 are in MST, and the non-minimal repetitive edges we delete in 3 cannot be in MST by the cut property.
- The edges that form self loop cannot be in MST by cycle property.
- After one Boruvka step, the number of nodes in G' is at most $\frac{1}{2}$ of G.
- Can be done in $O(m\alpha(n))$ using UnionFind with path compression and union by rank. Essentially $O(m)$

Boruvka's Algorithm: Iterative Boruvka Step

- If we iteratively apply Boruvka's step on a connected graph G until the contracted graph G' returned by it has only 1 node, we get a MST of G.
- Recall every Boruvka step reduce the number of nodes by at least a factor of 2: the algorithm has a runtime of $O(m \log n)$
- No better than Prim/Kruskal, but Boruvka step is used in most faster MST algorithms as a subroutine.

A Faster Deterministic Algorithm

Use Brouvka step with Prim's algorithm

procedure $\mathrm{MST}(G)$
  $G', E \leftarrow$ apply Boruvka step to G $\log\log n$ times.
  $T' \leftarrow$ apply Prim's algorithm to G'
  return $T' \cup E$
end procedure

Runtime

This has a runtime of $O(m \log\log n)$, which is better. More on next slide.

Runtime Analysis

- After applying $\log \log n$ Boruvka step , the number of nodes in G' should be $O(\frac{n}{2^{\log \log n}}) = O(\frac{n}{\log n})$
- Running $\log \log n$ times Brouvka step gives $O(m \log \log n)$
- Consider the runtime of Prim's algorithm on G' with at most $\frac{n}{\log n}$ nodes using Fibonacci heap: It should be: $O(m + \frac{n}{\log n} \log n) = O(m + n)$
- Combine these two together we achieved a runtime of $O(m \log \log n)$

A Randomized Algorithm with Linear Expected Runtime

General Idea

1. Get a contracted graph G', and selected edges E by applying Boruvka step to G 2 times
2. Create a subgraph H of G' by selecting each edge in G' with a probability of 0.5 , and recursivly apply the algorithm to H to get a minimum spanning forrest F of H.
3. Remove some edges that cannot be in the MST of G' from G' (F-heavy)
4. Recursively call the algorithm on G' to get its MST T'
5. Return $T' \cup E$

- This algorithm was developed by David Karger, Philip Klein, and Robert Tarjan.
- It requires a linear-time MST verification subroutine for step 3
- The correctness is evident by induction if we view step 2,3 as some procedures to accelerate the speed of computing the MST of G'.
- It has an expected runtime of $O(m)$.
- In the worst case this algorithm has the same runtime as Boruvka's algorithm. This is easy to see. If the randomness of step 2 and 3 does not help, the whole algorithm is basically a recursive implementation of Boruvka's algorithm

Summary

- Boruvka step: a way to calculate the edges that belong to MST and contract the graph in the meantime.
- A deterministic MST algorithm: A hybrid approach using Boruvka step with Prim's algorithm with a runtime of $O(m \log \log n)$
- Linear-time randomized algorithm: very hard to implement because of the linear-time MST algorithm it requires.

Outlook

There's a possible implementation of the MST verification algorithm by Hagerup.

For Further Reading I

DAVID R. KARGER, PHILIP N. KLEIN, ROBERT E. TARJAN
A Randomized Linear-Time Algorithm to Find Minimum Spanning Trees.
Journal of the Association for Computing Machinery March 1995

Torben Hagerup
An Even Simpler Linear-Time Algorithm for Verifying Minimum Spanning Trees
Graph-Theoretic Concepts in Computer Scince