15-210: Parallel and Sequential DSA April 11, 2022

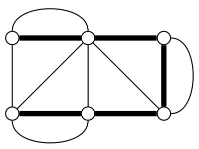
Lecture 21 : Minimum Spanning Trees

Lecturer: Umut Acar

Scribe: Rajeev Godse

1 Motivation

- Graphs can have a lot of edges. $|E|$ is only bounded by $\binom{|V|}{2} = \Theta(V^2)$.
- Sometimes, you don't need that many edges. You would rather reduce the number or weight of edges as much as possible without harming connectivity.
- **Spanning trees** achieve connectivity with the minimum number of edges ($|V| - 1$).
  - Given $G = (V, E)$, a spanning tree is a structure $T = (V, E')$ such that $E' \subseteq E$ and T is a tree.
  - Example (bolded edges form a spanning tree):



  - In general, there are many spanning trees over a given graph.
- **Minimum spanning trees** achieve connectivity with minimum weight.
  - Given weighted graph $G = (V, E)$ with weight function $w : E \to \mathbb{R}$, a minimum spanning tree is a structure $T = (V, E')$ where E' is the subset of E with minimal weight such that T is a tree.

2 Properties of (M)STs

2.1 Edge Replacment Lemma

- Suppose we want to insert an edge (u, v) into a spanning tree T, both over a graph G.
- There is a unique path from u to v via T. We can add in (u, v) and remove any of the edges of the path and we still have a spanning tree.
  - This doesn't change connectivity: any path that traverses the removed edge can be replaced by the inserted edge and the remainder of the path.
  - This doesn't introduce any cycles: paths are still unique.
- Alternatively, we can say that if we remove an edge from a spanning tree T, taking any edge between the two connected components that partition the graph yields an MST.

3 Algorithms

3.1 Spanning Trees

3.1.1 Build a Tree

- **Algorithm**:
  1. $T = \varnothing$
  2. Repeat:
     (a) Select an edge e.
     (b) If e doesn't create a cycle, then $T = T \cup \{e\}$
- **Correctness**: Adding an edge to a graph either creates a cycle or reduces the number of connected components by 1. If it's possible to select a spanning tree T, then edges will be selected until the number of connected components reaches 1.
- **Implementation**: Keep track of the connected components (using some disjoint-set data structure). Each time we consider an edge, check if it goes between two different connected components. If so, merge the components and add the edge. Otherwise, it introduces a cycle: reject.
- **Runtime**: If we use sets/tables, then the work and span is clearly $\Theta(m \log n)$. If we use the union-find data structure from 122, we can get down to $\Theta(m\alpha(n))$, where α is the inverse Ackermann function.

3.1.2 Grow a Tree

- **Algorithm**:
  1. Do graph search and return the tree edges.
- **Runtime**: Work: $O(m)$, Span: $O(d)$ (if BFS is used).

3.1.3 Contraction

- **Algorithm**:
  1. Repeat:
     (a) Have each vertex, in parallel, choose an edge incident on it.
     (b) Perform tree contraction on the chosen edges. Add the tree edges to the MST and continue with the contracted graph.
- **Runtime**: Work: $O(m \log n)$, Span: $O(\log^3 n)$, but can be squeezed down to $O(\log^2 n)$ with a smarter implementation (see Boruvka's algorithm).

3.2 Minimum Spanning Trees

3.2.1 Boruvka's Algorithm

- **Algorithm**:
  1. Each vertex, in parallel, picks the minimum weight edge incident on it.
  2. Contract the produced trees and recurse.
- **Correctness** is clear when we show that we don't get a cycle from our choice:

  - AFSOC that the choice of minimum weight vertices produces a cycle. The cycle has the same number of vertices as edges, so each edge must belong to a distinct vertex. Each edge is also incident on two vertices in the cycle. So each edge must be the minimum weight edge for one of the vertices it is incident on, but not the other. Thus, as we traverse the cycle in the direction of the minimum weight edges, the edge weights must be strictly decreasing. But arriving at the initial edge gives us that the edge has a smaller weight than itself, a contradiction. So the choice cannot produce a cycle.

- **Implementation**: We're not so happy with the extra log factor. Instead of performing tree contraction, we just do one round of star contraction on the forest produced by the edge selection. This still gives us a constant fraction reduction, so we get $O(\log^2 n)$ span and lose the contraction within the contraction.

3.2.2 Kruskual's Algorithm

- **Algorithm**: Build a tree, but select the edge with minimum weight.

3.2.3 Prim's Algorithm

- **Algorithm**: Grow a tree, but select the edge with minimum weight in the frontier.