Minimum Spanning Trees

Weighted Graphs
Minimum Spanning Trees
Kruskal’s Algorithm
Prim’s Algorithm
A weighted graph is a graph in which each edge \((u, v)\) has a weight \(w(u, v)\). Each weight is a real number. Weights can represent distance, cost, time, capacity, etc.
Optimization Problems for Weighted Graphs:
Find a minimum spanning tree.
Find shortest paths between vertices.
Maximize flow from a source to a sink.
Traveling salesman problem.

Variations on Problems:
Undirected/directed graphs
Negative weights
A spanning tree $T$ of a weighted, undirected graph $G$ is a subset of edges that form a (free) tree. If $G$ has $|V|$ vertices, then $T$ has $|V| - 1$ edges.

A minimum spanning tree (MST) is a spanning tree with a minimum sum of weights. Consider any cut of vertices $S$ and $V - S$. Let $E'$ be the edges between $S$ and $S - V$.

Theorem: A MST has a minimum weight edge (called a light edge) from $E'$. 
Proof:

Suppose $T$ is a spanning tree with no such edge. We can show $T$ cannot be a MST.

Let $(u, v)$ be a minimum weight edge from $E'$. $T$ has a path between $u$ and $v$. Some edge $(x, y)$ on this path is in $E'$. Replacing $(x, y)$ with $(u, v)$ decreases the sum. Implies $T$ is not a minimum spanning tree.

Algorithm idea:
Find light edges without making cycles.
Kruskal’s Algorithm

MST-KRUSKAL(G, w)

T ← ∅

for each vertex v ∈ G.V
    MAKE-SET(v)

for each edge (u, v) ∈ G.E in ascending order
    if FIND-SET(u) ≠ FIND-SET(v)
        T ← T ∪ {(u, v)}
        UNION(u, v)
        exit loop if all vertices are unioned

return T
Kruskal example

weight graphs
wg problems
minimum spanning
trees
mst analysis
kruskal
kruskal example
kruskal analysis
prim
prim example
prim analysis
Kruskal Analysis

Proof of correctness: \((u, v)\) is a light edge between \(\text{FIND-SET}(u)\) and the other vertices.

Use disjoint-set forest for union-find, almost \(O(E)\). Use \(O(E \lg E)\) sorting algorithm or binary heap. \(\text{MST-KRUSKAL}\) is \(O(E \lg E) = O(E \lg V)\).
Prims’s Algorithm

**MST-PRIM**\((G, w, r)\)

for each vertex \(v\) in \(G\)

\[v.key \leftarrow \infty\]

\(r.key \leftarrow 0\)

\(r.\pi \leftarrow \text{NIL}\)

\(Q \leftarrow G.V\)

while \(Q\) is not empty

\[u \leftarrow \text{EXTRACT-MIN}(Q)\]

for each \(v \in G.Adj[u]\)

if \(v \in Q\) and \(w(u, v) < v.key\)

\[v.\pi \leftarrow u\]

\(\text{DECREASE-KEY}(Q, v, w(u, v))\)
Prim Example

(a) Prim Example

(b) Kruskal Example

(c) Prim Analysis

(d) Kruskal Analysis
Prim Analysis

Proof of correctness:
Q is the vertices not yet added to the tree.
If v ∈ Q, then (v.π, v) is the smallest edge from v to V − Q, and v.key is its weight.
The minimum value in Q must be a light edge between Q and V − Q.

Use binary heap for priority queue, O(V) elts.
O(V) Extract-Mins is O(V lg V).
O(E) Decrease-Keys is O(E lg V).
MST-Prim is O(E lg V) = O(E lg E).
Using Fibonacci heaps makes it O(E + V lg V).