Minimum Spanning Trees (MSTs)

G is an undirected, connected graph, with a weight $w(u, v)$ associated with each edge.

Spanning tree is a subset of edges $T \subseteq E$ that forms a tree (acyclic and connect all vertices).

Minimum: with smallest total weights

Generic MST(G, W):

A = $\emptyset$

while A is not a spanning tree:

  find $(u, v) \in E$ safe for A

  $A = A \cup \{(u, v)\}$ // add it to A

return A

Remark: By loop invariant, there exist some safe edges.

Definitions:

- A cut of $G = (V, E)$ is a partition of V into $(S, V - S)$.
- Edge (u, v) crosses a cut $(S, V - S)$ if $u \in S$ and $v \in V - S$ or vice versa.
- Cut respects A if no edge in A crosses the cuts.
- (u, v) is a light edge crossing the cut if it has minimum weight of any edge entering the cut.

MST Safe Edge Theorem: Let $A \subseteq E$ be included in some MST, $(S, V - S)$ be a cut respecting A, and (u, v) a light edge crossing $(S, V - S)$. Then (u, v) is safe for A.

Proof: Let T be an MST containing A. If $(u, v) \in T$, we're done.

Assume $(u, v) \notin T$. Since T is a spanning tree, there is a path connecting u and v. Since $u \in S$ and $v \in V - S$, there exists an edge (x, y) on that path that crosses $(S, V - S)$.

Since (u, v) is a light edge, $w(u, v) \leq w(x, y)$.

$T' = T - \{(u, v)\} + \{(x, y)\}$ is also a spanning tree, $w(T') - w(x, y) + w(u, v) \leq w(T)$.

Therefore, (u, v) is safe.

Kruskal's algorithm

Consider edges by non-decreasing weights. Any edge that connects two connected components is added to A.

Correctness: such an edge is a light edge for the cut $(C, V - C)$ and therefore is safe.

MST-KRUSKAL($G = (V, E)$, w)

A = $\emptyset$

for $v \in V$:

$\mathrm{MAKESET}(v)$

Sort E in non-decreasing order of weights

for $(u, v) \in E$:

if $\mathrm{FINDSET}(u) \neq \mathrm{FINDSET}(v)$: // u, v in different connected components

$A = A \cup \{(u, v)\}$

$\mathrm{UNION}(u, v)$

return A

Runtime: $|V| \times \mathrm{MAKESET}$; sort $|E|$ numbers; $|E| \times \mathrm{FINDSET}$; $|V| \times \mathrm{UNION}$

Using Disjoint-set data structure:

$\mathrm{MAKESET} \quad O(1)$

$\mathrm{FINDSET} \quad O(log|V|)$

$\mathrm{MERGE} \quad O(log|V|)$

Total runtime: $O(V + ElogE + ElogV + VlogV) = O(ElogE) = O(ElogV)$

(Since $O(logE) \leq O(logV^2) = O(2logV) = O(logV)$)

Prim’s algorithm

Idea: we grow a tree starting from some root r. Each time we find the vertex outside the tree that is connected by the lightest edge, and add it (because it is safe by the theorem).

MST-PRIM($G = (V, E)$, w, r)

for $u \in V$:

u.key = ∞

$u.\pi$ = NIL

r.key = 0

Q = priority-queue(V) // waiting list

while $Q \neq \emptyset$:

u = EXTRACTMIN(Q) // u added to tree with edge $(u.\pi, u)$

for $v \in Adj[u]$:

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

$v.\pi = u$

DECREASEKEY($Q, v, w[u, v]$) // $v.key = w(u, v)$

Return the tree $\{(u.\pi, u)|u \in V - r\}$

The developing tree at any is given by $A = \{(u.\pi, u)|u \in V - r - Q\}$

Runtime: $|V| \times$insert; $|V| \times$EXTRACT; $|E| \times$DECREASEKEY

$O(V + VlogV + ElogV) = O(ElogV)$

With Fibonacci heaps, EXTRACTMIN is still $O(logV)$, but DECREASEKEY is only $O(1)$ (amortized).

So total runtime is $O(V + VlogV + E) = O(E + VlogV)$

which is generally better than Kruskal's algorithm.