

Minimum Spanning Trees

Given an undirected graph with weights on the edges

To find a spanning tree with minimum possible weight

Tree:

An undirected graph with no a cycle.

A tree with n vertices has $n-1$ edges.

Given

$$G = (V, E)$$

$$w: E \rightarrow \mathbf{R}$$

To find a spanning tree T of G such that

$$\sum_{e \in E(T)} w(e)$$

is minimum possible.

Spanning tree $T : V(T) = V(G)$

Simple greedy method

Sort edges so that $w(e_1) \leq w(e_2) \leq \dots \leq w(e_m)$.

$E(T) \leftarrow \{e_1\}$

For $i = 2, \dots, m$, add e_i to $E(T)$ if doing so does not create a cycle in $E(T)$.

Justifying first selection :

Let T_1 be a spanning tree not containing e_1 .

Then $T_1 \cup \{e_1\}$ contains a unique cycle and e_1 is on the cycle.

Removing any e , $e \neq e_1$, on the cycle results in a new spanning tree with weight

$$w(T) + w(e_1) - w(e) \leq w(T)$$

since $w(e_1) \leq w(e)$.

A general lemma

$$G = (V, E)$$

A cut : $(U, V - U)$.

An edge $e = (x, y)$ crosses the cut if $x \in U, y \in V - U$.

$$w : E \rightarrow \mathbf{R}$$

Let $A \subseteq E$ so that no edge of A crosses the cut.

Let $e \in E$ be a lightest edge crossing the cut.

Let Γ_A be the set of all spanning trees which contain A .

Then there is some $T \in \Gamma_A$ which contains e such that

$$w(T) \leq w(T'), \forall T' \in \Gamma_A.$$

Simple greedy as a corollary

$$G = (V, E)$$

$$e_1 = (x, y)$$

$$\text{Cut} : (U, V - U), U = \{x\}.$$

A : the empty set.

Proof of general lemma

Suppose $T \in \Gamma_A$ does not contain e .

Then $T \cup \{e\}$ contains a unique cycle and e is on the cycle.

Call the cycle C .

e crosses the cut \Rightarrow there is at least one other edge e' on C that also crosses the cut.

$T - \{e'\} \cup \{e\}$ is also a spanning tree, with weight

$$w(T) - w(e') + w(e) \leq w(T).$$

Kruskal's algorithm

Order $E : w(e_1) \leq w(e_2) \leq \dots w(e_m)$.

$\Gamma \leftarrow$ the empty set (initially).

For $i = 1$ to m , include e_i if it connects two trees in (V, Γ) .

Inductively, if e_i connects two trees in (V, Γ) , then it is a lightest edge crossing a cut where the two trees lie on different sides of the cut and Γ respects the cut.

How efficiently can the for - statement be performed?

Maintaining sets

Each set has a representative.

Initially, every vertex forms a singleton set.

$$p(v)=v, n(v)=1, \text{child}(v)=\text{nil}, \text{next}(v)=\text{nil}.$$

Inductively, set S has a representative v .

$$p(u)=v \text{ for all } u \text{ in } S.$$

$$n(v)=|S|.$$

Suppose an edge (x,y) is examined.

If $p(x) \neq p(y)$, then (x, y) connects two trees, hence we merge $\text{set}(x)$ and $\text{set}(y)$:

Say $n(p(x)) \leq n(p(y))$, then merge $\text{set}(x)$ to $\text{set}(y)$,

so that after the merge,

$p(z) = p(y)$ for all $z \in \text{set}(x)$.

Time for this : $O(| \text{set}(x) |)$

Time complexity of Kruskal

$$\begin{aligned} \text{Total set - updates} &= \sum_x \# \text{ times } p(x) \text{ is changed} \\ &\leq \sum_x \log n = O(n \log n) \end{aligned}$$

Every time $p(x)$ is changed, x is found in a set at least twice as large as the set before.

Sort edges: $O(m \log m)$

For statement: $O(m) + O(n \log n)$

Overall: $O(m \log m)$

Prim's algorithm

Start with any vertex v_0 .

Inductively, choose $(u, v) \in E$ of min. possible wt such that u is on T and v is not. $\dots\dots(*)$

Why it works?

Inductively apply General Lemma to :

$A = E(T)$ and the cut $(V(T), V - V(T))$

To implement (*):

Maintain a priority queue Q of vertices which are not on T .

Initially, $Q \leftarrow V - \{v_0\}$;

$\text{key}(v) \leftarrow \infty$; $\text{key}(v_0) \leftarrow 0$.

$p(v) \leftarrow v$ [parent pointer]

Inductively, for v not on T , hence $v \in Q$:

$\text{key}(v) = \min\{w(v, u) \mid u \in V(T)\}$

(*)while $Q \neq \text{empty}$ do the following

$u \leftarrow \text{extract - min}(Q)$; [u is the new node to join T]

Update from u : for each v adj to u , if $v \in Q$

[hence not on T] and $w(u, v) < \text{key}(v)$,

then $\text{key}(v) \leftarrow w(u, v)$ and $p(v) \leftarrow u$

Time complexity

Fibonacci heap to maintain priority queue.

$O(|V|)$ extract-min.

$O(|E|)$ updates.

So, $O(|E| + |V|\log|V|)$