W3U1: Minimum Spanning Trees

C Pandu Rangan

Recap

- SSSP and APSP

Outcomes

- Minimum Spanning Trees

MST Basics

Let $G = (V, E)$ be a

Connected,

Undirected,

Edge-weighted graph

- The weight of a n edge $e \in E$ is denoted by $w(e)$. The weights are arbitrary real numbers.
- A Tree is a connected graph that has no cycles (acyclic).
- A tree $T = (V', E')$ is said to be a spanning for a graph $G = (V, E)$ if $V' = V$ and $E' \subseteq E$.

MST Basics (contd)

In other words, T is a spanning tree for G if T and G have same vertex set and each edge of T is an edge of G.
Cost of a spanning tree T is defined by $w(T) = \sum_{e \in T} W(e)$.
A spanning tree T of G is said to be a Minimum Cost Spanning Tree or simply Minimum Spanning Tree (MST) if $w(T) \leq w(T')$ for any spanning tree of T' of G.

MST Basics (contd)

- We are interested in designing algorithms to find a minimum for a given spanning tree connected, weighted graph G.
- It can be shown that if all edge weights of G are distinct, then G has a unique minimum spanning tree.
- From now on, we assume that edge weights are distinct numbers.

Properties of Spanning Trees

Properties of Spanning Trees (contd)

Properties of Spanning Trees (contd)

MST – Prim's Algorithm

- We start with a general result that allows us to build the edge set of a MST in a variety of ways. Each way of building T leads to a different algorithm for obtaining a MST.
- Let $G = (V, E)$ be a Connected, Undirected, weighted graph.
- A cut is a partition of V into two disjoint sets $(S, V - S)$. An edge (x, y) where $x, y \in S$ or $x, y \in V - S$ is called an internal edge of the cut and an edge (x, y) with $x \in S$ and $y \in V - S$ is called crossing edge of the cut.
- Let $A \subseteq E$ be a set of edges. A cut $(S, V - S)$ respects A if no edge of A is crossing edge of the cut. That is, all edges of A are internal edges with respect to the cut $(S, V - S)$.

Theorem

- Let $G = (V, E)$ be a Connected, weighted, and undirected graph.
- Let $A \subseteq E$ be a set of edges of some minimum spanning tree of G.
- Assume that $(S, V - S)$ is a cut respecting A and let e be a crossing edge with minimum weight. (that is $w(e) \leq w(e')$ for any crossing edge e' of the cut $(s, V - s)$).
- Thus, $A \cup \{e\}$ is a subset of the edge set of some MST of G. (This MST may be different from the MST that contained A.

Template for MST Algorithms

- Based on this interesting 'mathematical' result, we may devise a 'computational scheme' that may be stated as follows:

$A = \emptyset$

For $i = 1 \; to \; n - 1$

Find a cut $(S, V - S)$ respecting A.

Find a minimum weight crossing edge e of the cut $(S, V - S)$

$A = A \cup \{e\}$

Return $T = (V, A)$ \\ T is a MST of G

Remark

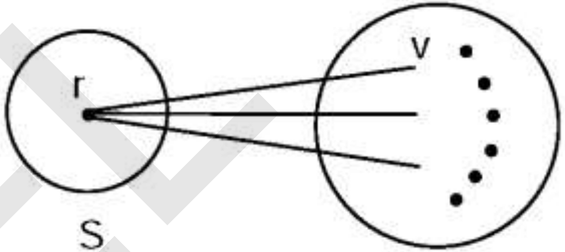
In the first step in the for loop, we may choose any cut respecting A.

So, we may choose a respecting cut that is easy to compute. Again, in every iteration way may freshly construct a respecting cut for the updated A.

However, we must explore the possibilities of updating the cut $(S, V - S)$ respecting A to a cut $(S', V - S')$ respecting $A \cup \{e\}$.

Thank You

- When A is empty, the simplest cut is just a vertex in S, say r, and rest of the vertices in $V - S$. The minimum cost crossing edge is an edge (r, v) such that $w(r, v) \leq w(r, x)\ \forall x \in Adj(r)\ in\ V - s$

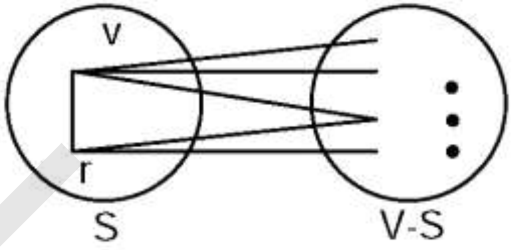


We now update

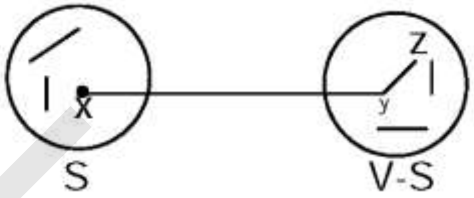
$A = A \cup \{r, v\} = \emptyset \cup \{r, v\} = \{(r, v)\}$

How do we update the current cut to respect the updated A?

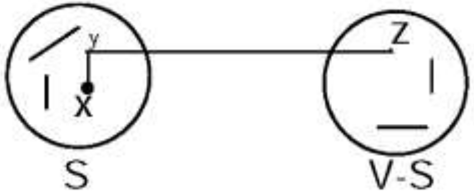
Simple idea – move $v\ to\ S$!



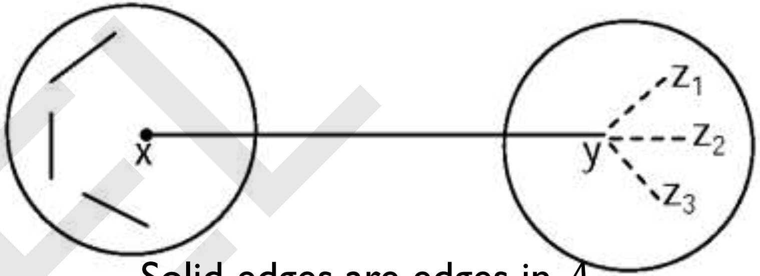
- This move will convert the crossing edge (r, v) to an internal edge with respect to the new cut and thus the new cut would be a cut respecting current A.
- Notice that the set of crossing edges has become bigger and it is a bit more complex to find the minimum weight crossing the edge now.
- Moreover, there is a minor issue in simply moving the vertex v from $V - s$ to s



- Suppose current minimum weight crossing edge is (x, y). The edges of A are shown in the picture. If we move y to s, the edge (y, z) in A becomes a crossing edge and hence the new cut is NOT a respecting cut!
- Thus, while moving y may make (x, y) an internal edge, it may make some other edge of A a crossing one!
- Thus, simply moving y when (x, y) is a minimum weight crossing edge may not work!

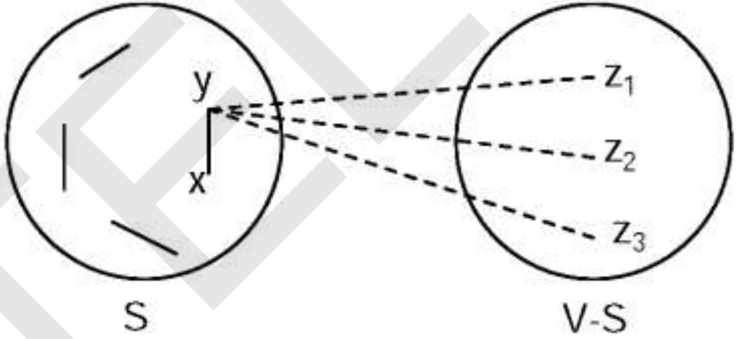


- Fortunately, there is a simple fix to this problem. We maintain all edges of A as internal to S and NO EDGE of A is in $V - s$.
- In this case, there is no problem in moving any vertex from $V - s$ to s.
- Now, moving $y\ to\ s$ makes (x, y) internal to s and new crossing edges are NOT in A. Hence, new cut respects A and has all edges of A internal to s.



Solid edges are edges in A
dotted edges are NOT in A.

- The new crossing edges created by moving $y\ to\ s$ are NOT in A. The new cut respects A and has all edges of A internal to S.

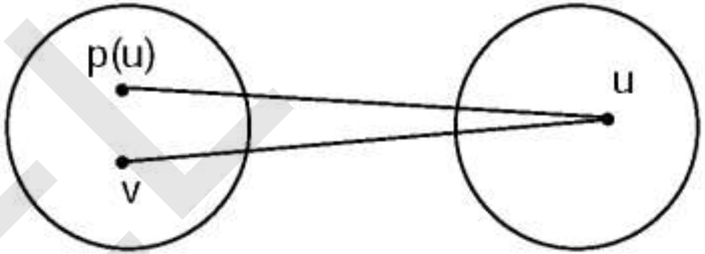


- The first cut, namely $(\{r\}, V - \{r\})$ has the property that all edges of A are internal to $\{r\}$ because $A = \emptyset$!
- Thus, we may identify the minimum weight crossing edge (u, v) where $u \in S, v \in V - s$ and simply move v from V-s to s.
- Now we turn our attention to finding minimum weight crossing edge.
- In the absence of any other information, minimum weight crossing edge may be identified only by checking all crossing edges and this could take $O(m)$ time. Since we perform $(n-1)$ iteration, the complexity of such a naïve algorithm will be $O(nm)$.

- Suppose we maintain for every vertex v in $V - s$ a minimum weight crossing edge (v, n). Then the overall minimum weight crossing edge can be identified by looking at only edges of the type (v, u) for $v \in V - S$.
- Thus, by examining only $|V - S|$ edges we identify the minimum weight crossing edge. Since $|V - S| \leq (n - 1)$ this is a significant improvement over $O(m)$ naïve approach.
- Let us introduce some notations to present our approach in a cleaner way.

- For every vertex v in $V - S$, let (v, v')be a minimum weight crossing edge, incident on v. Here $v' \in S$. That is $w(v, v') \leq w(v, u)$ for all crossing edge (v, u) incident on v.
- We call (v') as a partner of v in S and denote it by $p(v)$. The weight of the crossing edge $(v, p(v))$ is denoted by $c(v)$. Thus, for each vertex v, we maintain $p(v)$ and $c(v)$ and now, it is easy to determine the minimum weight crossing edge.
- In fact, let v be a vertex in $V - S$ -→

- $c(v) \leq c(u) \forall u \in V - S$. Then, clearly, $\left(v, p(v')\right)$ is the minimum weight crossing edge. Hence, we add $(v, p(v))$ to A and move v to S.
- After moving v to S, the values $p(u)$ and $c(u)$ must be updated for $u \in V - S$.
- The entity $p(u)$ needs an update only if $(u, v) \in E$ and $w(u, v) < w(u, p(v))$. If $(u, v) \in E$ and $w(u, v) \subset w(u, p(n))$
- $p(u)$ is updated to v.



High level pseudocode:

$A = \emptyset, S = \{r\}, V - S = V - \{r\}$

For all $v \in V$,

$p(v) = \text{NULL}$

$c(v) = \infty$

For each $u \in V - S$

If $(n, u) \in E$

$p(u) = r;$

$c(u) = w(r, u)$

while $S \neq v$

Find a vertex v in $V - S$ with minimum $c(v) value$
Move v to S and add $(v, p(v))$ to A
Update $c(u)$ and $p(u)$ values for all $u \in V - S$

Pseudocode in more detail"

Prim $\left(G = (v, E, w)\right)$

\\$r \in V. r$ is arbitrary.

\\$T = (v, A), A$ is implicitly represented by $p(\,)$.
$A\{\left(v, p(v)\right)|$

$v \in V - r\}$

$S = \{r\}$ \\\\($S, V - S$ is the current cut)

$p(v) = \text{NULL}\ \forall\ v \epsilon V$

$c(v) = \infty \forall\ v \epsilon V$

For all $u \in V - S$

If $(u, r) \in E$

$p(u) = r$

$c(u) = w(u, r)$ while ($V - S$ is non empty)

W_U_

- What are we giving in next presentation.

Thank You