Minimum Spanning Trees

Given a connected, undirected graph $G = (V, E)$ with edge weights $w(u,v)$ for each edge $(u,v) \in E$, the minimum spanning tree (MST) $T = (V, E')$ of $G$, $E' \subseteq E$, is an acyclic, connected graph such that $w(t) = \sum_{(u,v) \in E'} w(u, v)$ is minimized.

Example

```
    a     b     c
   3 - 4
 G:  2   1
    d
```

Applications

Circuit wiring: connecting common pins with minimal wire

Networking

Growing a Minimal Spanning Tree

Greedy approach

Given $A \subseteq T = \text{MST}(G)$, determine a minimum $(u,v)$ to add to $A$ such that $A \cup \{(u,v)\} \subseteq T$

Greedy-MST($G,w$)

$A = \{\}$
while A is not a spanning tree ; includes all vertices of G
find a safe edge \((u,v)\) for A
\[ A = A \cup \{(u,v)\} \]
return A

What is a “safe” edge?
A safe edge is an edge connecting a vertex in \(A \subseteq T\) to a vertex in \(G\) that is not in \(A\) such that \(A \cup \text{safe edge} \subseteq \text{MST}\).

Definitions:

A ____ \((S, V-S)\) of an undirected graph \(G = (V, E)\) is a partition of \(V\).

An edge \((u,v) \in E\) _________ the cut \((S, V-S)\) if \(u \in S\) and \(v \in V-S\).

(a,b) and (b,d) cross the cut
Definitions

A cut _______ the set $A$ of edges if no edge in $A$ crosses the cut.

\[
\begin{array}{ccc}
\text{a} & \text{3} & \text{b} \\
\text{2} & & \text{1}
\end{array}
\]

\[
\begin{array}{ccc}
\text{c} & \text{4} & \text{d}
\end{array}
\]

$A = (V, E)$, $V = \{a, b, c, d\}$, $E = \{(a,b), (b,c)\}$

An edge is a _______ crossing a cut if its weight is the minimum of any edge crossing the cut.

\[
\begin{array}{ccc}
\text{a} & \text{3} & \text{b} \\
\text{2} & & \text{1}
\end{array}
\]

\[
\begin{array}{ccc}
\text{c} & \text{4} & \text{d}
\end{array}
\]

(b,d) is the light edge

Theorem 24.1

Given a connected, undirected graph $G = (V, E)$ with edge weights $w$, $A \subseteq \text{MST}(G)$, cut $(S, V-S)$ that respects $A$, and light edge $(u,v)$ crossing $(S, V-S)$, then $(u,v)$ is a safe edge.

Proof: Assume $T = \text{MST}(G)$ contains edge $(x,y)$ crossing $(S, V-S)$. Note that $(x,y)$ must be on a unique path connecting $u$ to $v$. Edge $(u,v)$ would form a cycle. Removing $(x,y)$ breaks $T$ in 2 parts, but $(u,v)$ reconnects
them.

$T'$ is the new resulting MST.

Since $(u,v)$ is a light edge, then $T' = T - \{(x,y)\} \cup \{(u,v)\}$ is also MST($G$).

Note that this is true because $(u,v)$ and $(x,y)$ cross the same cut and $(u,v)$ is safe, $w(u,v) \leq w(x,y)$, $w(T') = w(T) - w(x,y) + w(u,v) \leq w(T)$.

Since $(x,y) \notin A$ ($(S, V-S)$ respects $A$), then $A \cup \{(u,v)\} \subseteq T' = \text{MST}(G)$. Thus, $(u,v)$ is a safe edge.

---

**Corollary 24.2**

Given $A \subseteq \text{MST}(G)$ and a connected component $C$ of the forest $G_A(V, A)$, if $(u,v)$ is a light edge connecting $C$ to some other component in $G_A$, then $(u,v)$ is safe for $A$.

**Algorithm:**

1. Find two unconnected components of $G$.
2. Connect them using a light edge.

---

**Kruskal’s Algorithm**

Kruskal’s Algorithm

repeat
find a light edge (u,v) between two unconnected components
A = A ∪ {(u,v)}
until all edges have been considered

• Sort the edges by weight
• Use disjoint sets for speed (union by rank and path compression)

MST-Kruskal(G, w) ; G = (V, E)
1 A = {}
2 foreach v in V ; O(V)
3 MakeSet(v)
4 sort edges E by nondecreasing weight w ; O(E lg E)
5 foreach edge (u,v) in E, in order ; m = |E| operations
6 if FindSet(u) ≠ FindSet(v) ; n = |V| keys
7 then A = A ∪ {(u,v)} ; O(m α(m,n))
8 Union(u,v) ; O(E α(E,V))
9 return A ; α(E,V) = O(lg E)

T(V,E) = O(V) + O(E lg E) + O(E lg E), V = O(E)
         = O(E lg E)
Example

\[
\text{Sorted } E = \{(b,d), (a,d), (a,b), (b,c)\}
\]

Prim’s Algorithm

Prim’s Algorithm
repeat
find minimal edge \((u,v)\) connecting \(A\) to a vertex not in \(A\)
\[ A = A \cup \{(u,v)\} \]
until all vertices are in \(A\)

Implementation
Maintain a priority queue \(Q\) of vertices of the form

| parent | points to neighbor vertex in \(A\) along smallest edge |
| key    | weight of smallest edge |
| (in \(Q\)) | true or false |
Starting from some root vertex \( r \)
Update key and parent slots of vertices on \( A \) adjacent to \( r \)
Extract minimum-key vertex \( v \) from those adjacent to \( r \)
\( r = V \)

**Pseudocode**

\[
\text{MST-Prim}(G, w, r)
\]

1. \( \text{foreach } v \text{ in } V \) \; \( \text{O(V), BuildHeap} \)
2. \( \text{key}(v) = \infty \) \; \( \text{Fibonacci Heap } O(E + V \lg V) \)
3. \( \text{(inQ}(v) = \text{true}) \)
4. \( \text{Insert}(Q, v) \)
5. \( \text{key}(r) = 0 \)
6. \( \text{parent}(r) = \text{NIL} \)
7. \( \text{while } Q \neq \text{NIL} \) \; \( \text{O(V)} \)
8. \( u = \text{Extract-Min}(Q) \) \; \( \text{O(lg V)} \)
9. \( \text{(inQ}(u) = \text{false}) \) \; \( \text{Fibonacci Heap } O(\text{lg V}) \)
10. \( \text{foreach } v \text{ in } \text{Adj}(u) \) \; \( \text{O(E) total} \)
11. \( \text{if inQ}(v) \text{ and } w(u,v) < \text{key}(v) \) \; \( 2 \mid E \mid \)
12. \( \text{then parent}(v) = u \) \; \( \text{O(lg V), DecreaseKey} \)
13. \( \text{key}(v) = w(u,v) \) \; \( \text{O(V lg V + E lg V)} \)

\[
\text{O(V lg V + E lg V) = O(E lg V)}
\]
\[
\text{Fibonacci Heap: } O(E + V \lg V)
\]

**Example**

\[
\text{MST-Prim}(G, w, r)
\]
Applications