CS 380
ALGORITHM DESIGN AND ANALYSIS

Lecture 21: Minimal Spanning Trees
Kruskal’s, Prim’s algorithms
Text Reference: Chapter 23

Spanning Tree

• What are the edges you need to keep the graph connected?
  • If you remove any edge, the graph becomes disconnected
• Minimum Spanning Tree
  • minimize the total weight of the edges
• Problem: Minimal set of roads needed to connect cities
Minimum Spanning Tree

• Model as a graph:
  • Undirected graph $G = (V, E)$
  • Weight $w(u, v)$ on each edge $(u, v)$ in $E$
  • Find $T$ that is a subset of $E$ such that
    • $T$ connects all vertices and
    
    \[ w(T) = \sum_{(u,v) \in T} w(u,v) \]
    is minimized

A spanning tree whose weight is minimum over all spanning trees is called a **minimum spanning tree**

• Example:

![Graph with edges and weights](image)
Growing an MST

• Properties of an MST:
  • It has $|V|-1$ edges
  • It has no cycles
  • It might not be unique

• Building up a Solution
  • We will build a set $A$ of edges
  • Initially $A$ has no edges
  • As we add edges we maintain the invariant:
    • Loop Invariant: $A$ is a subset of MST
  • Add only edges that maintain the invariant. If $A$ is a subset of MST, an edge $(u, v)$ is safe for $A$ if and only if $A \cup \{(u, v)\}$ is also a subset of some MST. So, we will add only safe edges.

Generic MST Algorithm

```
GENERIC-MST(G, w)
A = ∅
while A is not a spanning tree
    find an edge $(u, v)$ that is safe for $A$
    $A = A \cup \{(u, v)\}$
return A
```
Finding a Safe Edge

• How do we find safe edges?
• Edge (c,f) - Is it safe for A?

Intuitively: Let S (subset of V) be any set of vertices that includes c but not f (f is in V-S). In any MST, there has to be one edge that connects S with V-S. Why not choose the edge with the minimum weight?
Definitions

• Let S be a subset of V and A be a subset of E
  • A cut \((S, V - S)\) is a partition of vertices into disjoint sets \(S\) and \(V - S\)
  • Edge \((u,v)\) in E crosses cut \((S,V-S)\) if one endpoint is in \(S\) and the other is in \(V-S\)
  • A cut respects A if and only if no edge in A crosses the cut
  • An edge is a light edge crossing a cut if and only if its weight is minimum over all edges crossing the cut

Theorem

• Let A be a subset of some MST, \((S,V-S)\) be a cut that respects A, and \((u,v)\) be a light edge crossing \((S,V-S)\).
  • Then…the edge \((u,v)\) is safe for A.
**Generic-MST**

- So, in a generic MST
  - A is a forest containing connected components. Initially, each component is a single vertex
  - Any safe edge merges two of these components into one. Each component is a tree
  - Since an MST has exactly $|V|-1$ edges, the for loop iterates $|V|-1$ times. Equivalently, after adding $|V|-1$ safe edges, we’re down to just one component

**Kruskal’s Algorithm**

- $G = (V,E)$ is a connected, undirected, weighted graph. $w:E\rightarrow\mathbb{R}$ weight function
  - Starts with each vertex being its own component
  - Repeatedly merges two components into one by choosing the light edge that connects them
  - Scans the set of edges in monotonically increasing order by weight
  - Uses a disjoint-set data structure to determine whether an edge connects vertices in different components
Kruskal(V,E,w)

Kruskal(G, w)

A = ∅

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

\( \text{MAKE-SET}(v) \)

sort the edges of \( G.E \) into nondecreasing order by weight \( w \)

for each \( (u, v) \) taken from the sorted list

if \( \text{FIND-SET}(u) \neq \text{FIND-SET}(v) \)

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

\( \text{UNION}(u, v) \)

return \( A \)

Example
Prim’s Algorithm

- Builds one tree, so $A$ is always a tree
- Starts from an arbitrary “root” $r$
- If $V_A =$ vertices included in the tree $A$, at each step, find a light edge crossing the cut $(V_A, V-V_A)$. Add this edge to $A$

How to Find a Light Edge Quickly

- Use a priority queue $Q$:
  - Each object is a vertex in $V-V_A$
  - Key of $v$ is minimum weight of any edge $(u,v)$, where $u$ is in $V_A$
  - Then the vertex returned by EXTRACT-MIN is $v$ such that there exists $u$ in $V_A$ and $(u,v)$ is a light edge crossing $(V_A, V-V_A)$
  - Key of $v$ is infinity if $v$ is not adjacent to any vertices in $V_A$
Prim’s Algorithm

- The edges of A will form a rooted tree with root r:
  - r is given as an input to the algorithm, but it can be any vertex
  - Each vertex knows its parent in the tree by the attribute \( \pi[v] = \text{parent of } v \). \( \pi[v] = \text{NIL} \) if \( v = r \) or \( v \) has no parent

- The minimum spanning tree is:

\[
A = \{(v, v.\pi) : v \in V - \{r\}\}
\]

PRIM(G,w,r)

```pseudo
PRIM(G, w, r)
Q = \emptyset
for each \( u \in G.V \)
    \( u.key = \infty \)
    \( u.\pi = \text{NIL} \)
    INSERT(Q, u)
DECREASE-KEY(Q, r, 0) \quad // r.key = 0
while Q \neq \emptyset
    u = EXTRACT-MIN(Q)
    for each v \in G.Adj[u]
        if v \in Q and w(u, v) < v.key
            \( v.\pi = u \)
            DECREASE-KEY(Q, v, w(u, v))
```
Example