1 A New Problem – MST

For this class, let $G$ be an undirected weighted graph.

- **Defn:** a spanning tree on $G$ is a subset of $G$’s edges that (1) forms a tree (i.e. no cycles) and (2) touches every vertex of $G$.

- Notice that a spanning tree has exactly one path between any two vertices in $G$. Adding any edge of $G$ to such a tree creates a cycle.

- A minimum spanning tree for $G$ is a spanning tree $T$ that minimizes the sum of its edge weights

\[ W(T) = \sum_{(u,v) \in T} w(u, v). \]

- Could be more than one minimum spanning tree (same weight)

2 How to Find MST

- Will use greedy algorithm due to Prim

- Start from some (any) vertex.

- Build up spanning tree $T$, one vertex at a time.

- At each step, add to $T$ the lowest-weight edge in $G$ that does not create a cycle.

- Equivalently, connect to vertex not in $T$ that is closest to $T$. 
3 Is Greed Correct?

Greedy strategy finds a spanning tree, but why is it minimum? Can prove inductively on number of vertices added to “in-progress” tree $T$.

**Thm:** The tree $T$ built by greedy method is always a subtree of some minimum spanning tree for $G$.

- **Base**: when $T$ has one vertex (no edges), trivially true.
- **Ind**: Suppose current tree $T$ is contained in an MST $T_0$.
- Let $(u, v)$ be lowest-weight edge connecting a vertex $u \in T$ to a vertex $v \notin T$.
- If edge $(u, v)$ is in $T_0$, then $T \cup (u, v)$ contained in $T_0$, and we are done.
- Otherwise, $T_0$ contains some other path $P$ connecting $u$ to $v$ (because it spans $G$).

- Path $P$ must contain an edge $(x, y)$ connecting some vertex $x \in T$ to a vertex $y \notin T$ (since $u$ is in $T$ but $v$ is not).
• If we remove \((x, y)\) from \(T_0\) and add \((u, v)\), claim that resulting graph \(T'\) is a new minimum spanning tree.

• *Spanning*: if vertices \(p, q\) were connected by a path through \((x, y)\), they are still connected, now by a path through \((u, v)\).

• *Tree*: adding \((u, v)\) to \(T_0\) cannot create more than one cycle, which was broken by removing \((x, y)\).

• *Minimum*: by assumption, \(w(x, y) \geq w(u, v)\).

• Hence, if we remove \((x, y)\) from \(T_0\) and add \((u, v)\), we have

\[
W(T') = W(T_0) - w(x, y) + w(u, v) \leq W(T_0)
\]

• Conclude that \(T'\) is also an MST, and it contains \(T \cup (u, v)\). QED

4 **Making the Greedy Algorithm Efficient**

How can we efficiently find closest unconnected vertex to \(T\) at each step of Prim’s algorithm?

• Maintain priority queue of unconnected vertices

• Vertex’s key is weight of its lowest-weight edge (cheapest connection) to \(T\)

• As vertex \(v\) is added to \(T\), can update connections for all neighbors in \(\text{Adj}[v]\) using \text{decreaseKey}.

Example:
Does this algorithm look familiar to anyone?

5 Pseudocode

Build MST $T$ in graph $G$ starting from vertex $s$.

```
PRIM($G$, $s$)
    for $u \in V$ do
        $u$.distance $\leftarrow \infty$
        $u$.parent $\leftarrow$ null
        $Q$.insert($u$, $\infty$)
    $T$ $\leftarrow$ $\emptyset$
    $s$.distance $\leftarrow 0$
    $Q$.decreaseKey($s$, 0)

    while $Q$ is not empty do
        $u$ $\leftarrow$ $Q$.extractMin()
        if $u$.distance $= \infty$
            stop
        $T$ $\leftarrow$ $T$ $\cup$ ($u$.parent, $u$)
        for $v \in \text{Adj}[u]$ do
            if $Q$.decreaseKey($v$, $w(u, v)$)
                $v$.distance $\leftarrow w(u, v)$
                $v$.parent $\leftarrow u$
```

6 Efficiency

Analysis is identical to that for Dijkstra’s algorithm. So is the running time.

- Binary heap: $O(m \log n)$
- Fibonacci heap: $O(n \log n + m)$