Minimum Spanning Tree

Ch 24
Hewett
Minimum Spanning Tree (MST)

- **Spanning Tree** – A tree that connects all vertices
- **MST** is a spanning tree $T$, where
  \[ w(T) = \sum_{(u,v) \in T} w(u, v) \text{ is minimum (over all spanning trees)} \]
- **MST Problem**: Find an MST of a given undirected graph $G$

MST is not unique
Characteristics

- **MST problem** – optimal substructure => Greedy Strategy
- If all the edges have distinct weights => unique MST

\[ w(T_1) = 42 \]

\[ w(T_2) = 47 \]

\( T_1 \) is MST

\( T_1 - \{9\} \) is MST for \( G - \{9\} \)

\( T_2 \) is not MST

\( T_2 - \{2\} \) is not MST for \( G - \{2\} \)
Terminologies

- Let $T$ be an MST of $G$ and $A \subseteq T$.
  $(u, v)$ is safe if $A \cup \{(u, v)\} \subseteq$ some MST of $G$.
  E.g., suppose $T$ is the only MST of $G$.
  Edge weight 10 is not safe for any $A \subseteq T$. 

\[ G \quad \text{and} \quad T \quad \text{such that} \quad w(T) = 42 \]
Terminologies

- A **cut** of $G = (V, E)$ is a partition of $V$ into $(S, V-S)$
- A cut of $G$ respects $A \subseteq E$ if no edge in $A$ crosses the partitions
- A cross edge with minimum weight is **light** edge

The cut respects $A$ (blue shade) since no edge in $A$ crosses the cut

Cross edges – edges with weight 3, 10, 2
Light edge – edge with weight 2
Greedy Strategy

• Each step add a “safe” edge to a subset of MST found so far. I.e., MST grows one edge at a time
• How to find “safe” edge?

**Theorem:** $T$ is a MST that includes a subset of edges, $A \subseteq T$. For any cut $(S, V-S)$ of connected undirected graph $G = (V, E)$ that respects $A$, the light edge $(u, v)$ crossing the cut is safe for $A$.

Cut: $S = \{a, b, c, g\}$ and $V-S = \{e, f, h\}$

$A = \{(a, c), (c, g)\}$

The cut respects $A$

$(d, e), (b, e), (c, f)$ are cross edges but only $(c, f)$ is safe since it is light edge
Kruskal’s Algorithm

• Each step add to the forest, the edge connecting two trees in the forest with minimal weight (light edge). The connected forest is MST

```
MST-Kruskal (G, w)
1   A ← φ
2   for each vertex v ∈ V[G]
3     do Make-set(v)    Each vertex is a set
4   sort the edges of E by non-decreasing weight w
5   for each edge (u, v) ∈ E in the sorted list
6     do if Find-set(u) ≠ Find-set(v)
7        then A ← A ∪ {(u, v)}    Pick the min edge
8        Union(u, v)
9   return A
```

Pick the cross edge
Kruskal’s Algorithm

Rank of weights: 2, 2, 3, 3, 3, 8, 8, 9, 15
Consider \((c, f)\). Find-set\((c)\) = \{"c\} ≠ \{"f\} = Find-set(f)

\[ A = \{(c, f)\}. \] Similarly for \((a, b)\) ⇒ \[ A = \{(c, f), (a, b)\} \]
Consider \((b, c)\). Find-set\((b)\) = \{"a, b\} ≠ \{"c, f\} = Find-set (c)

Thus, add \((b, c)\) to \(A\).
Next for \((a, c)\), Find-set\((a)\) = Find-set\((c)\). Don’t add \((a, c)\) to \(A\)
Kruskal’s Algorithm

Rank of weights: 2, 2, 3, 3, 3, 8, 8, 9, 15

Another solution
Analysis of MST-Kruskal

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

1. \(A \leftarrow \emptyset\)
2. for each vertex \(v \in V[G]\)
3. \hspace{1em} do Make-set(\(v\))
4. sort the edges of \(E\) by non-decreasing weight \(w\)
5. for each edge \((u, v) \in E\) in the sorted list
6. \hspace{1em} do if Find-set(\(u\)) \(\neq\) Find-set(\(v\))
7. \hspace{2em} then \(A \leftarrow A \cup \{(u, v)\}\)
8. \hspace{2em} \text{Union}(u, v)
9. return \(A\)

\(O(V)\) \hspace{2.5cm} O(E \, lg(V))
Given $d$. Cut: $S = \{d\}$ and $V-\{S\}$

Choose light edge from all cross edges: $(d, e), (d, b)$

Cut: $S = \{d, e\}$ and $V-\{S\}$

Choose light edge from all cross edges: $(d, b), (e, b), (e, f)$

Cut: $S = \{d, e, b\}$ and $V-\{S\}$

Choose light edge from all cross edges: $(b, a), (b, c), (e, f)$
Prim’s Algorithm

• Analysis (see text)
  • Depends on how priority Q is implemented
    • Unsorted array $\sim O(V^2)$
    • Binary Heap $\sim O(E \lg V)$