Overview of Chapter

- Ground work
  - Generic minimum spanning tree method
- Compare two algorithms
  - Kruskal's algorithm
  - Prim's algorithm
- Ground work

Minimum Spanning Trees (Chapter 23)

- Connected Undirected graph
- Edges have weights
- Vertices and edges
- Minimum spanning tree
- Forests minimum weight subset of edges that connects every vertex
- Spanning tree

Questions:

- How fast is this used for?
- When might this be used for?
- Do we have to add the acyclic?
- Hence the use of the term spanning tree
- Does not need to be a path
- Must be connected
- Connected Unirected Graph

P. Heeman, 2017

© P. Heeman, 2017
Growing a Min-Spanning Tree

- Such an edge is called a safe edge.
- At each step, determine an edge (u, v) that we can add to \( V \), where \( V \) is a subset of some minimum spanning tree.
- Loop invariant: \( \emptyset = G \) (a set of edges) starting with \( V \).

Generic Method for Growing a Tree

Let \( G = (V, E) \) with a weight function \( w : E \to \mathbb{R} \).
Theorem 23.1: In Plain English

- Will it be greedy?
- Will find minimum spanning tree

Any algorithm that follows Theorem 23.1:

- Do we evolve the cut or pick a fresh cut each time?
- How should we pick cut?
- Is it hard to find such a cut?

Questions:

- Is it easy to find a safe edge?
- Find edge of min weight edge that crosses cut
- Pick some vertices to make a cut that respects A
- If you have A, a partial min-spanning tree for G

Theorem 23.1

Let $G = (V, E)$ be a connected, undirected graph with $\omega : E \rightarrow \mathbb{R}$.

Let $A$ be a subset of $E$ that is included in some minimum spanning tree for $G$.

Let $C = (F, A)$ be a connected, undirected graph with $\omega : F \rightarrow \mathbb{R}$.

Finding Safe Edges

- Edge is a safe edge if its weight is min of any edge
- A cut respects a set of edges if no edge in $F$ crosses the cut
- Edge crosses cut $F$ if one vertex in $F$ and the other in $A$ is in a partition of $A - S$.
- Definition:
Another way to look at the Algorithm

- Define \( G_A = (V, A) \)
  - \( G_A \) is a forest
    - Each connected component in \( G_A \) is a tree (no cycles, connected)
- Algorithm
  - Start with \( A = \emptyset \).
    - \( G_A \) has \(|V|\) trees
  - Any safe edge \((u, v)\) will connect two distinct components of \( G_A \).
    - \( |A| = 0 \) if \( A \) has no trees

Proof (Not by contradiction)

Let \( T \) be a min-spanning tree that includes \( A \)

Let \((S, V-S)\) be a cut that respects \( A \).

Let \((u, v)\) be a light edge that crosses the cut.

Case 1:
\((u, v) \in T\).
Done

Case 2:
\((u, v) \) forms a cycle with edges on simple path from \( u \) to \( v \) in \( T \).
Hence, at least one other edge \((x, y)\) in \( T \) must cross cut

Create \( T' \) by removing \((x, y)\) and adding \((u, v)\).

\( T' \) will be a spanning tree.

Since \((u, v)\) is lighter, its weight must be same as \((x, y)\).

So \( T' \) will also be a min-spanning tree.

Form a cycle with edges on simple path from \( u \) to \( v \) in \( T \).

Case 2: \( \exists (a'\cdot n) \in \mathbb{E} \).

Done
Overview

Growing a Min Spanning Tree

Prim's Algorithm

Kruskal's Algorithm

Questions

• What are the choices left?
• How does this restrict the cut used?
• How does this restrict the previous algorithm?
• What is the cut used by the light edge crossing?

Corollary 23.2

Let $G = (V, E)$ be a connected, undirected graph with $w : E \to \mathbb{R}$.

Let $A$ be a subset of $E$ that is in some min spanning tree of $G$.

Let $C$ be a connected component of $G_A$.

If $(u, v)$ is a light edge between $C$ and some other component in $G$,

then $(u, v)$ is safe for $A$.
Kruskal's Algorithm

1. INIT-SET(P) for each vertex $v \in G$
2. Sort the edges of $G$ into non-decreasing order by weight $w$
3. MAKE-SET($u$) for each vertex $v \in G$
4. Sort the edges of $G$ into non-decreasing order by weight $w$
5. FOR each edge $(u,v) \in G$: taken in non-decreasing order by weight
6. IF FIND-SET($u$) $\neq$ FIND-SET($v$) THEN
7. UNION($u$, $v$)
8. RETURN $A$

What data structures does it need?

Can this be proved correct by Corollary 23.2?

Pick edge $(u,v)$ of min weight connecting any two trees in $G$

MST-KRUSKAL($G$)

Code
Running Time

- Overall $O(\log V)$ operations
- From Chapter 21: $O(E \log E)$ operations
- So $O(E)$ operations
- Graph is fully connected so

```
|A| > |E|

$\lambda \geq \log E = \log |E| \geq \log |V| - 1$
```

- $|E| \geq |V| - 1$
- $|E| < |V|^2$

Set operations:

- make-set $O(1)$
- find-set $O(1)$
- union $O(1)$
- union $O(1)$
- make-set $O(1)$

For each edge $e \in E$, taken in nondecreasing order by weight $w$:

```
1. make-set $V \in E$
2. find-set $V \in E$
3. union
4. sort the edges of $E$ in nonincreasing order by weight $w$
5. for each edge $e = u \rightarrow v$ in nonincreasing order by weight $w$
6. return
```

Example
Prim's Algorithm

- Can easily be \( O(VE) \)
- Go through all edges to find min connecting \( A \) to a vertex not in \( A \)
- To determine next vertex to add:
  - Add a light edge that connects \( A \) to a new vertex
  - Scan with an arbitrary vertex
  - Edges in a always form a single tree

Overview

- Growing a Min Spanning Tree
- Prim's Algorithm
- Kruskal's Algorithm
\[
\begin{align*}
&(a, n) \cdot n = \omega \cdot a, \\
n = n \cdot a, \\
\omega \cdot a > (a, n) \cdot n & \Rightarrow \quad \text{for each } a \in \emptyset \quad \text{and } a \in \emptyset \\
\text{for each } a \in \emptyset & \Rightarrow \text{EXTRACT-MIN}(\emptyset) = n \\
\emptyset \neq \emptyset & \Rightarrow \\
\emptyset, \emptyset & = 0 \\
0 = \omega \cdot n & = \text{NIL} \\
\text{NIL} = n & = \omega \cdot n \\
\omega \cdot n & = \text{NIL} \\
\emptyset & = \emptyset \\
\text{for each } & \text{ not in } A \\
\text{MST-PRI(M}(G, n, u), w) \\
\end{align*}
\]

A Better Way

- Need to update weights: min-priority queue
  - If it does, update its weight/edge
  - For all vertices adjacent to \( u \), see if \( u \) provides better way to \( A \) (via \( n \))
  - When new vertex \( u \) added to \( A \)
    - No edge to \( A \): use \( \infty \)
    - Keep its weight/edge to any vertex in \( A \)
    - Keep its min weight/edge to any vertex in \( A \)
  - For each vertex not in \( A \)
Running Time

Example