Spanning Tree

- What is spanning tree?
  - Subgraph that contains all the vertices of the original graph and is a tree

- Property
  - Often, a graph has many different spanning trees.

- Cost of spanning tree: the sum of the edge costs in the spanning tree: $10 + 8 + 13 + 15 + 11 = 57$

Minimum Spanning Tree (MSP)

- Definition
  - A spanning tree with the minimum cost

- Graph
  - Cost: $3 + 8 + 7 + 9 + 11 = 28$

- Minimum spanning tree
  - Minimum spanning trees are useful in constructing networks: a minimum spanning tree gives the way to connect a set of points with the smallest total amount of wire
  - Algorithms: Prim’s algorithm, Kruskal’s algorithms

Prim’s Algorithm

- Prim’s algorithm builds the MST one vertex at a time.
- Prim’s algorithm keeps track of:
  - $U$, the set of vertices included in the MST so far, and
  - $V - U$, the set of vertices not in the MST yet and, for each such vertex $v$ in $V - U$, the cheapest edge connecting to $v$ any vertex in $U$.

- Procedure
  - Begin with $U = \{V_0\}$
  - At each step
    - Look at all the vertices in $V - U$
    - Choose the vertex $v_j$ with the cheapest edge $(v_i, v_j)$ ($v_i \in U$ and $v_j \in V$)
    - Add $v_j$ to $U$
  - After $k$ steps (where $k$ is the number of vertices), the chosen vertices and their cheapest edges form a MST for the graph.
Prim’s Algorithm

- Key procedure in each step
  - Find a vertex in \( V - U \) with the cheapest edge connecting it to any vertex in \( U \):
  - Keep the vertices in \( V - U \) in a priority queue.
  - The priority of a vertex \( v \) in the priority queue is the cost of the cheapest edge connecting \( v \) and some vertex in \( U \).
  - If there is no such edge, then \( v \)’s priority is infinity.
  - Record the edge that gives a vertex its priority, because when a vertex is removed from the priority queue, its recorded edge becomes a spanning tree edge.
- Goal
  - A cheapest edge: the lowest cost is the highest priority
  - Use a min heap.

An Example

• Starting from A

<table>
<thead>
<tr>
<th>Vertex</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>1</td>
<td>2</td>
<td>inf</td>
<td>inf</td>
<td>inf</td>
</tr>
<tr>
<td>Edge</td>
<td>(A,B)</td>
<td>(A,C)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

An Example

• Add the first priority vertex, B.

<table>
<thead>
<tr>
<th>Vertex</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority</td>
<td>2</td>
<td>inf</td>
<td>1</td>
<td>inf</td>
</tr>
<tr>
<td>Edge</td>
<td>(A,C)</td>
<td>-</td>
<td>(B,E)</td>
<td>-</td>
</tr>
</tbody>
</table>
An Example

Vertex | C | D | F
---|---|---|---
Priority | 1 | inf | 2
Edge | (A,C) | - | (E,F)

An Example

Vertex | D | F
---|---|---
Priority | 2 | 1
Edge | (C,D) | (E,F)

An Example

Vertex | D
---|---
Priority | 1
Edge | (C,D)

An Example

Vertex | D | F
---|---|---
Priority | 1 | 1
Edge | (C,D) | (E,F)
Kruskal’s Algorithm

• Kruskal’s algorithm builds the MST one vertex at a time.
• Create a forest $F$ (a set of trees), where each vertex in the graph is a separate tree
• Create a set $S$ containing all the edges in the graph
  • while $S$ is nonempty and $F$ is not yet spanning
    ▪ Remove an edge with minimum weight from $S$
    ▪ If that edge connects two different trees, then add it to the forest, combining two trees into a single tree
    ▪ otherwise discard that edge.
• At the termination of the algorithm, the forest forms a minimum spanning forest of the graph.

An Example

• Find a minimum spanning tree of the following weighted graph using Kruskal’s algorithm.

An Example

• Start with the edge with minimum weight.

An Example

• Add the first priority edge as long as it does not create a cycle.
An Example

Edge (A,C) (B,C) (C,E) (C,D) (C,F) (E,F)
Priority 1 4 2 5 6 2

Analysis of MST Algorithms

• Greedy algorithm:
  ▪ Making the locally optimal choice at each stage
  ▪ No backtracking

• Time complexity
  ▪ Prim’s algorithm
    ▪ $O(|V|^2)$ with adjacency matrix
    ▪ $O(|E|\log|V|)$ with binary heap and adjacency list
  ▪ $O(|E| + |V|\log|V|)$ with Fibonacci heap and adjacency list
  ▪ Kruskal’s algorithm: $O(|E|\log|V|)$
Hamiltonian Cycle

- A cycle that goes through every vertex exactly once

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Traveling Salesperson Problem (TSP)

- Definition
  - The problem of finding the minimum cost Hamiltonian circuit in a weighted graph

- Properties
  - Intractable problem
    - NP: There is no polynomial time algorithm.
    - Many other problems are in this class.
  - Time complexity of TSP $> O(n^k)$ where $n$ is the number of vertices
  - Time complexity of naïve algorithm: $O(n!)$
  - Time complexity of the best known algorithm: $O(2^n)$
  - Greedy algorithm
    - Time complexity: $O(n^2)$
    - Not necessarily optimal