Minimum Spanning Trees

- Subgraph that contains all $N$ vertices of the original graph but only $E=N-1$ edges
- Examples for non-directed and unweighted graphs:

Algorithms to create a MST can be derived from DFS while keeping track of the path

One can use either DFS or BFS, but we need to record the edges
Weighted Graphs: Motivations

- Weighted graphs → edges have weights
  - Distance
  - Dollars cost of traversal
  - Delay, etc.
- Weights make algorithms more 'interesting'

**Minimum spanning tree (MST):**

- Find cheapest set of flights that connect all airports
- Cheapest installation for cable TV that connects a set of cities
- Etc

**Shortest Path Problem:**

- What is the shortest (or cheapest) distance from one vertex to another
- Etc.
Weighted Graphs and MST

- Find a set of edges with minimum total weight that connect all nodes.
  - Not easy to 'see' the correct solution. Example:

- **Idea:**
  - Add edges that are cheap
  - Do not add edges between nodes that have already been connected (avoid redundancy)

- **Two basic algorithms** (there are greedy algorithms...making locally optimal choices at each stage)
  - Prim's algorithm (textbook)
  - Kruskal's algorithm
MST- Prim

- Initialize the tree with a single vertex (chosen arbitrarily). Then Repeat until all vertices are in the tree:
  - 1- Find all the edges from the newest vertex to other vertices that are not in the tree. Put these edges in a priority queue (descending order)
  - 2- Pick the edge with lowest weight, and add this edge and its destination vertex to the tree.

- Example

List of edges:
- AB6, AD4,
- BC10, BE7, BD7
- CD8, CE5, CF6
- DE12, EF7

MST → 5 edges (with minimum total weight)
MST- Prim

- Start with A
  - Tree: A
  - PQ: AB6, AD4

- Dequeue → AD4

- D is current node
  - Tree: A D
  - PQ: DE12, DC8, AB6
  - Remark: DB7 not in PQ since AB6 already connects to B

(we make sure there is no other edges going to the same destination. If there is, we keep the one with smallest weight)
MST- Prim

- Dequeue → AB6
  - B is current node
    - Tree: A D B
    - PQ: DC8, BE7
    - Remark: BC10, DE12 not in PQ

- Dequeue → BE7
  - E is current node
    - Tree: A D B E
    - PQ: EF7, EC5
    - Remark: DC8 not in PQ
MST - Prim

- Dequeue → EC5
  - C is current node
    - Tree: A D B E C
    - PQ: CF6
    - Remark: EF7 not in PQ
- Dequeue → CF6
  - F is current node
    - Tree: A D B E C F
  - Done

Total weight: 4 + 6 + 7 + 5 + 6 = 28
class Edge{
    public int srcVert;
    public int destVert;
    public int distance;
    public Edge(int sv, int dv, int d){
        srcVert=sv;
        destVert=dv;
        distance=d;
    }
}

class Vertex {
    private char label;//label e.g. A,B
    public boolean isInTree;
    public Vertex (char c) {
        label = c;
        isInTree = false;
    }
}

class PriorityQ{
    private Edge[] array; // ideally one should use a heap
    private int size;
    ...
    // List of methods to implement:
    // insert, removeMin
    ...
}

class Graph {
    // Define:
    // adjacency matrix,
    // List of vertices, priorityQ, etc.
    // Two important methods:
    // 1- mstw
    // 2- putInPQ (used by mstw)
    }
    }

Complete code p681 Textbook (not optimal)
Todo:
Java applet
GraphW.html
MST- Prim (full PQ implementation)

A in Tree
AB6,AD4
Dequeue → AD4

D in Tree
DE12,DC8,DB7,AB6
Dequeue → AB6

B in Tree
DE12,BC10,DC8,DB7,BE7
Dequeue → BE7

E in Tree
DE12,BC10,DC8,DB7,EF7,EC5
Dequeue → EC5

C in Tree
DE12,BC10,DC8,DB7,EF7,CF6
Dequeue → CF6

F in Tree
Done Tree{A,D,B,E,C,F}
MST- Kruskal

- Assign each vertex to its own set
- Add all edges to Priority Q
- For each edge in PriorityQ (dequeue)
  - If the edge connects to different sets (and does not form a cycle)- add edge to tree
  - Merge sets

**Example**

- 6 vertices:
  - A B C D E F
- Priority Q
  - DE12, BC10, CD8, BE7
  - BD7, EF7, AB6, CF6, CE5, AD4
MST- Kruskal

- DE12, BC10, CD8, BE7, BD7, EF7, AB6, CF6, CE5, AD4
MST - Kruskal - other example

Step 1-
Find least-cost edge SWF-LGA: $49

<table>
<thead>
<tr>
<th></th>
<th>ALB</th>
<th>BDL</th>
<th>LGA</th>
<th>JFK</th>
<th>EWR</th>
<th>HPN</th>
<th>PVD</th>
<th>BOS</th>
<th>MHT</th>
<th>BTV</th>
<th>SWF</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALB</td>
<td>0</td>
<td>0</td>
<td>75</td>
<td>123</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>257</td>
<td>59</td>
</tr>
<tr>
<td>BDL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>167</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>82</td>
<td>0</td>
<td>147</td>
</tr>
<tr>
<td>LGA</td>
<td>75</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>63</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>180</td>
<td>49</td>
</tr>
<tr>
<td>JFK</td>
<td>123</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>55</td>
<td>95</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EWR</td>
<td>0</td>
<td>167</td>
<td>63</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>288</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HPN</td>
<td>0</td>
<td>0</td>
<td>55</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>225</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>PVD</td>
<td>0</td>
<td>0</td>
<td>95</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>110</td>
<td>0</td>
<td>202</td>
</tr>
<tr>
<td>BOS</td>
<td>82</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>225</td>
<td>0</td>
<td>0</td>
<td>99</td>
<td>0</td>
<td>160</td>
<td>0</td>
</tr>
<tr>
<td>MHT</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>288</td>
<td>0</td>
<td>110</td>
<td>99</td>
<td>0</td>
<td>0</td>
<td>139</td>
<td>0</td>
</tr>
<tr>
<td>BTV</td>
<td>257</td>
<td>147</td>
<td>180</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>202</td>
<td>0</td>
<td>139</td>
<td>0</td>
</tr>
<tr>
<td>SWF</td>
<td>59</td>
<td>0</td>
<td>49</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>160</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
MST- Kruskal- other example

Step 2-
Next least-cost edge JFK-HPN: $55

Step3 and Step4
MST- Kruskal- other example

Step 5-
Next least-cost edge ALB-LGA: $75
But causes a loop ! So we skip
MST - Kruskal - other example
MST- Result
Total Cost: $874