

Algorithms and Data Structures



COMP261 **Minimum Spanning Trees**

Yi Mei

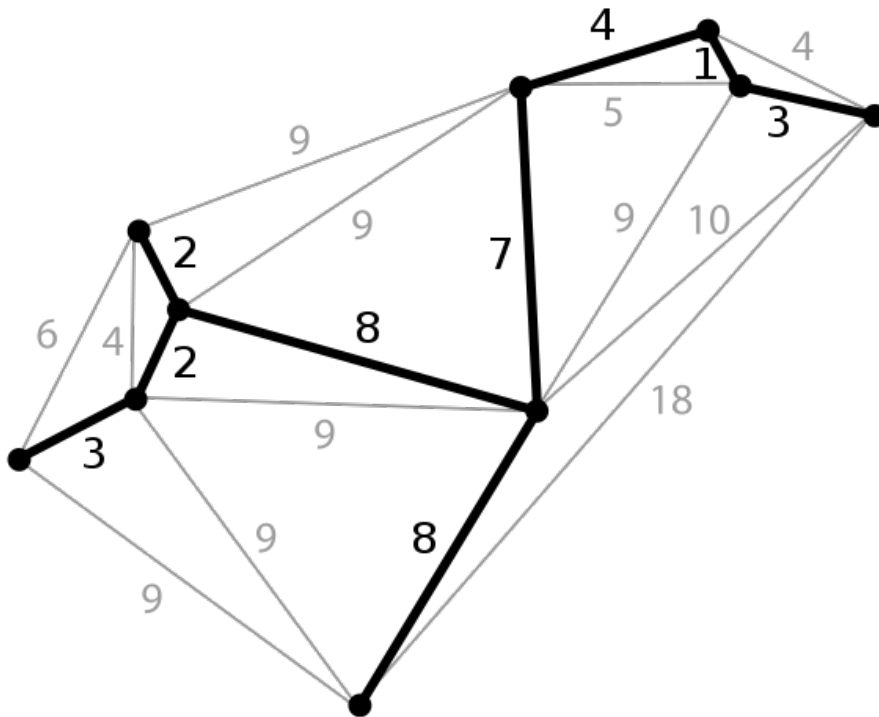
yi.mei@ecs.vuw.ac.nz

Outline

- Why Minimum Spanning Tree
- What is Minimum Spanning Tree
- Finding Minimum Spanning Tree
 - Prim's Algorithm
 - Kruskal's Algorithm

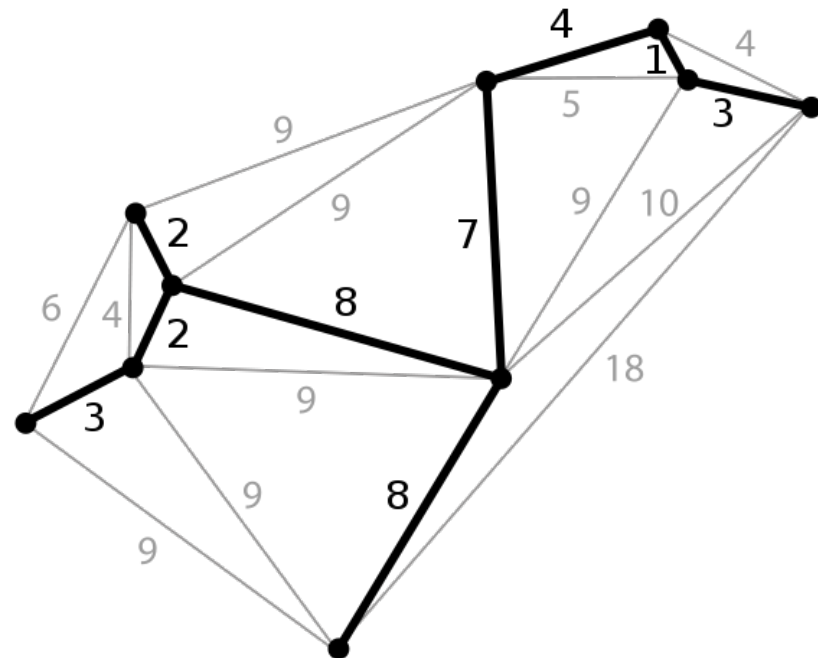
Why Minimum Spanning Tree

- Many applications
 - Telecommunication (cover the network with minimum cost)
 - Cluster analysis (e.g. gene expression clustering)
 - Image segmentation



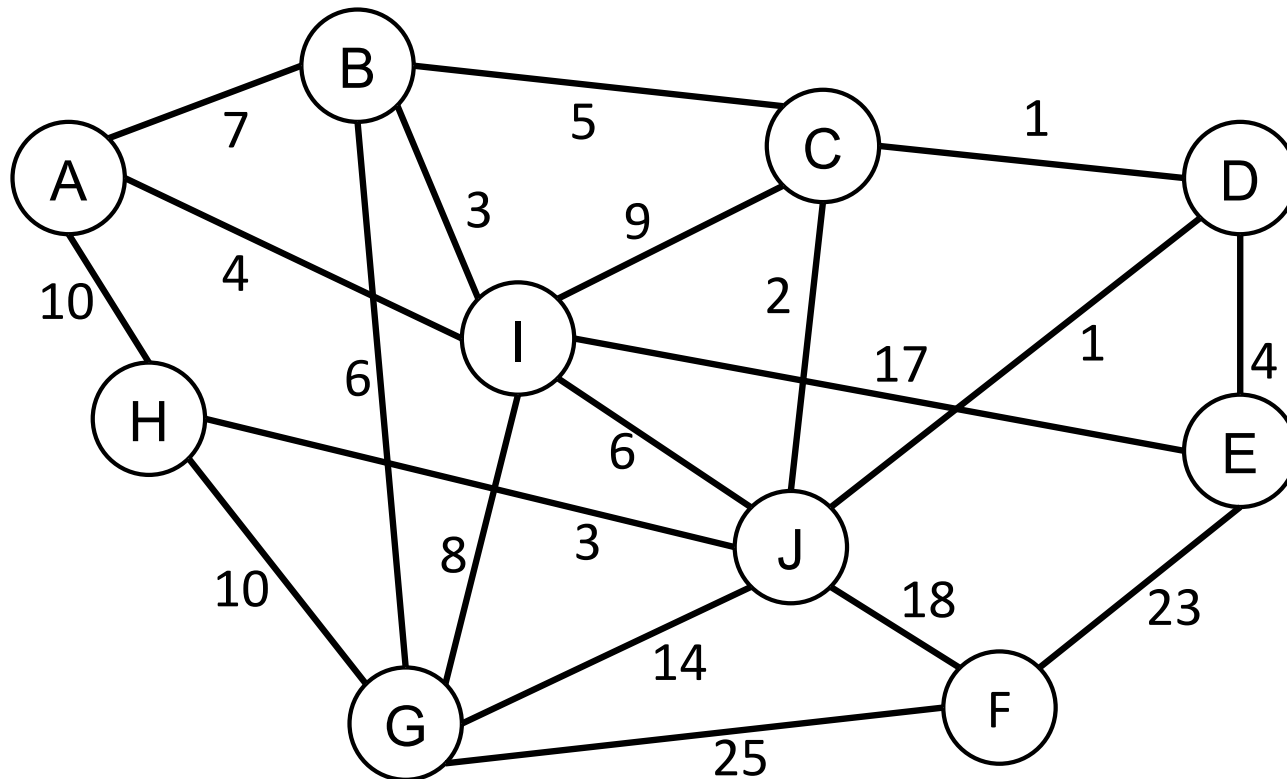
Minimum Spanning Tree

- Given a **connected, undirected, weighted** graph
- A **spanning tree** is a subgraph that contains **all the nodes** but is a **tree (no cycle)**
 - A spanning tree does not require a weighted graph
- A **minimum spanning tree (MST)** is a spanning tree with the **minimum total weight** among all the spanning trees, i.e. its total weight is no greater than the total weight of any other spanning tree
- Two algorithms to find MST
 - Prim's Algorithm
 - Kruskal's Algorithm
 - Both can prove **correctness**



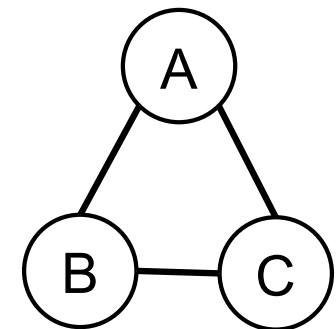
Prim's Algorithm

- Grow the tree from a root node
 - Randomly select a **root node**, initialise a single-node tree
 - Repeatedly **add one node** outside the tree into the tree until all the nodes are in the tree
 - Add **a new edge**: **one node in the tree, the other outside the tree**
 - The added edge has the **minimum weight**



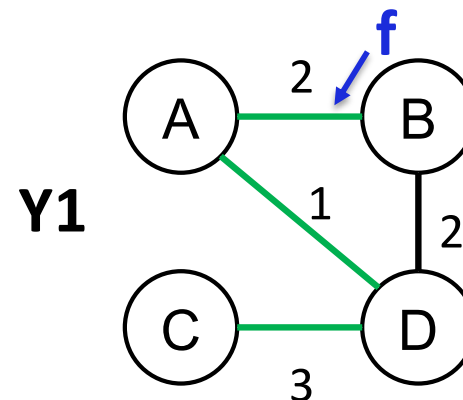
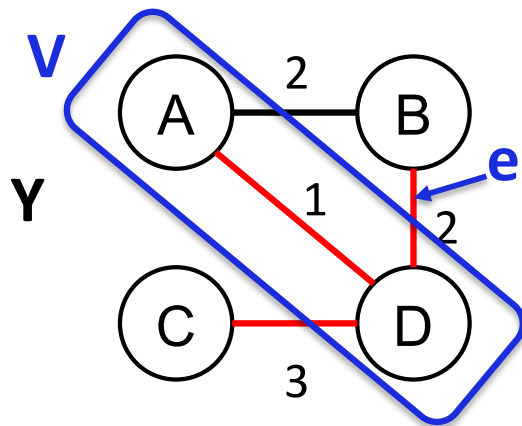
Correctness of Prim's Algorithm

- **Theorem:** the tree obtained by Prim's Algorithm is a MST
- **Proof:**
 - The tree **contains all the nodes**
 - Algorithm does not stop until all the nodes have been added
 - The tree has **no cycle**
 - If there is a cycle, then when adding the last edge, the two nodes of the edge are already in the tree (conflicting with the algorithm)



Correctness of Prim's Algorithm

- **Theorem:** the tree obtained by Prim's Algorithm is a MST
- **Proof:**
 - The tree has the **minimum total weight**
 - Let **Y** be the tree obtained by Prim's Algorithm
 - **Y_1** be the MST, **Y_1 not equal to Y**
 - Let **e** be the first edge that is not in Y_1 during the algorithm, **V** be the set of nodes in the tree before adding e. We have another edge **f** in **Y_1** , with one node in V, the other outside V.
 - Since Prim's algorithm selects e rather than f, then **$w(e) \leq w(f)$**
 - If we **remove f** from Y_1 , **add e** to Y_1 , we get Y_2 with **$w(Y_2) \leq w(Y_1)$**
 - Repeat this, ... **$Y_n = Y$** , and **$w(Y_n) \leq w(Y_1)$** . So Y is also a MST



Prim's Algorithm

Given: a connected undirected weight graph

Initialise fringe to have a root node with costToTree = 0 and a dummy edge, all nodes are unvisited;

Repeat until all nodes are visited {

 Choose from fringe the unvisited node (n^*) with minimum costToTree;

 Add the corresponding edge to the spanning tree, set n^* as visited

for each (edge (n^* , n') with one end-node n^*) {

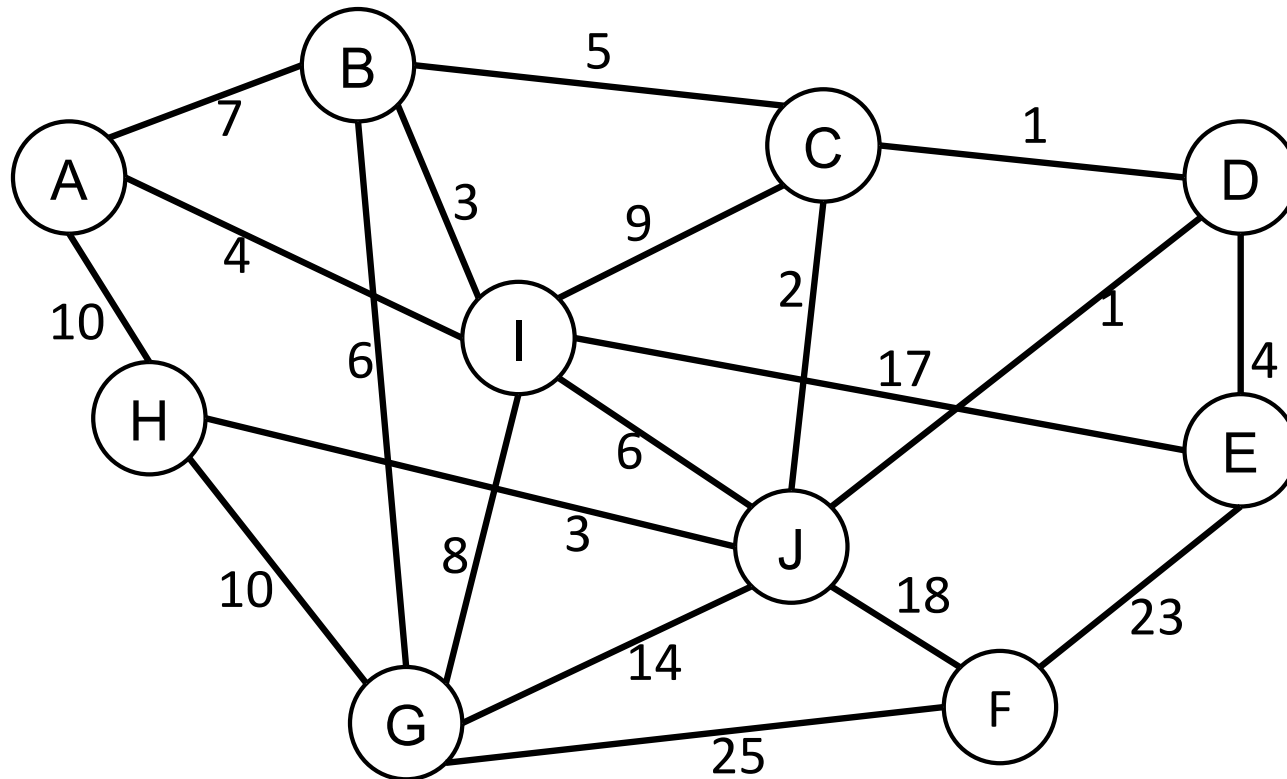
if (n' is not visited) **then** add $\langle n', (n^*, n'), cost(n^*, n') \rangle$ into the fringe;

 }

}

Kruskal's Algorithm

- Merge trees
 - Initially, each node is a **single-node tree**
 - At each step, **merge two trees into one**
 - The merge cost is the **minimum** (min-cost edge)

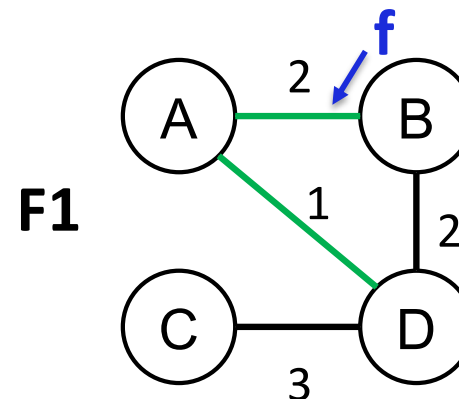
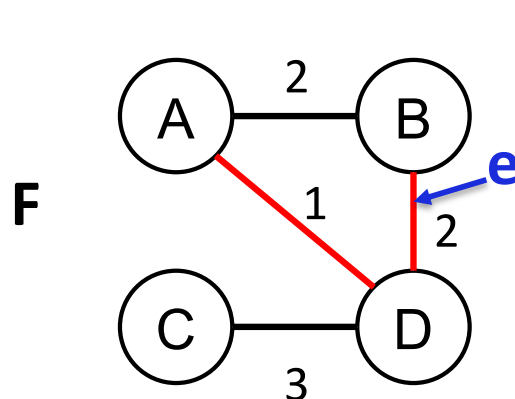


Correctness of Kruskal's Algorithm

- **Theorem:** the tree obtained by Kruskal's Algorithm is a MST
- **Proof:**
 - The tree cannot have cycle (always merge two tree into one)
 - The tree has all the nodes
 - Therefore, the tree is a spanning tree
- If F is the set of edges chosen at any stage of the algorithm, then there is some minimum spanning tree that contains F .

Correctness of Kruskal's Algorithm

- If F is the set of edges chosen at any stage of the algorithm, then there is some minimum spanning tree that contains F .
 - At the beginning, F is empty, so MST contains F
 - If F is contained in a MST, then $F + e$ is also contained in a MST, where e is the new edge added by Kruskal's algorithm
 - If not, then there is another tree $F1$ with the same nodes, and is contained in a MST.
 - There exists an edge f in $F1$ but not in F .
 - $F1 - f + e$ is another tree, and $\text{cost}(F1 - f + e) = \text{cost}(F1) - \text{cost}(f) + \text{cost}(e) \leq \text{cost}(F1)$.
 - $F = F1 - f + e$ is contained in a MST



Kruskal's Algorithm

Given: a connected undirected weight graph (N nodes, M edges)

Set forest as N node sets, each containing a node;

Set fringe as a priority queue of all the edges $\langle n1, n2, \text{length} \rangle$;

Set tree as an empty set of edges;

Repeat until forest contains only one tree or edges is empty {

 Get and remove $\langle n1^*, n2^*, \text{length}^* \rangle$ as the edge with **minimum length** from fringe;

If ($n1^*$ and $n2^*$ are in different sets in forest) {

 Merge the two sets in forest;

 Add the edge to tree;

 }

}

return tree;

Complexity of Prim's and Kruskal's Alg

- Depends on data structure
 - If using adjacency matrix for graph?
 - If using priority queue?
- Can we do better in Kruskal's algorithm?
 - Efficiently check two nodes are in different sets in forest
 - Efficiently merge two trees
 - A new data structure: disjoint sets

Summary

- Minimum Spanning Tree (MST)
- Two algorithms for finding MST
 - Prim's algorithm
 - Kruskal's algorithm
- Correctness
- Complexity
- Next lecture: disjoint set for efficient Kruskal's algorithm