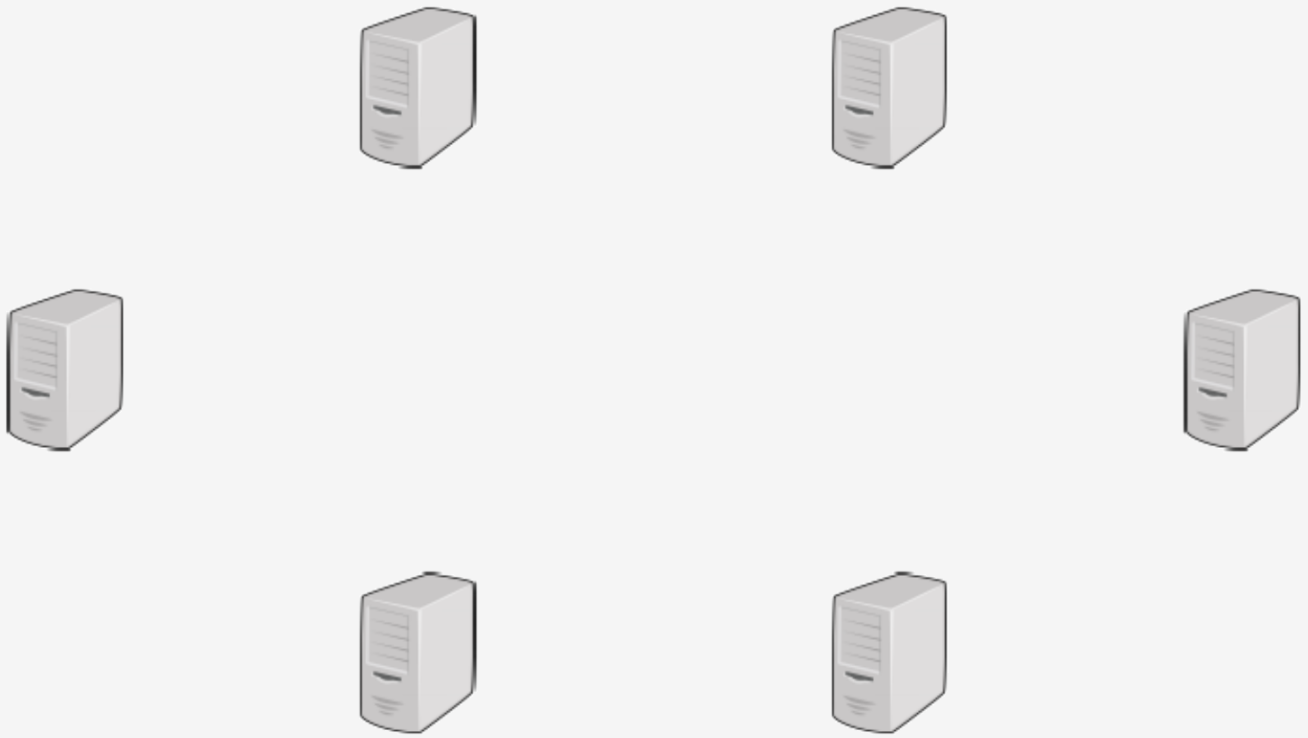
Design and Analysis of Algorithms

06-04
Minimum Spanning Tree

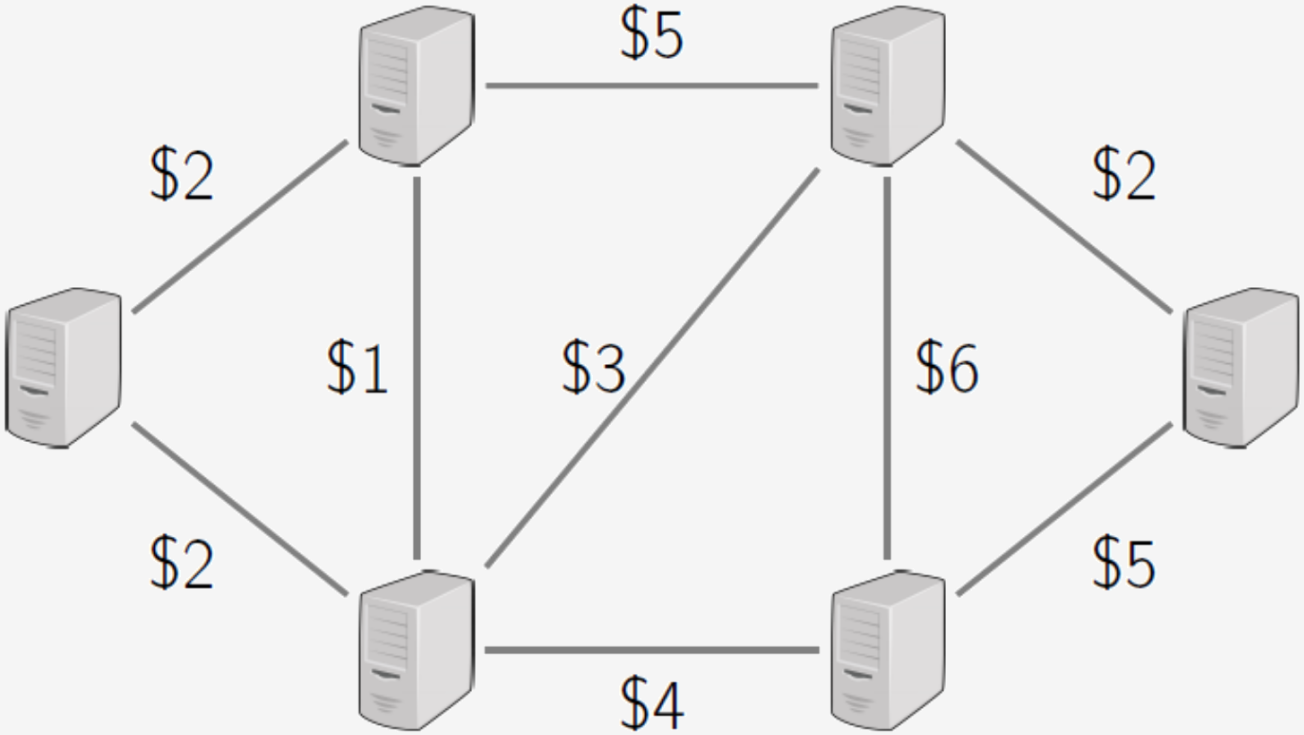
Kruskal's Algorithm and Prim's Algorithm

Imran Ihsan
Assistant Professor, Department of Computer Science
Air University, Islamabad, Pakistan
www.imranihsan.com

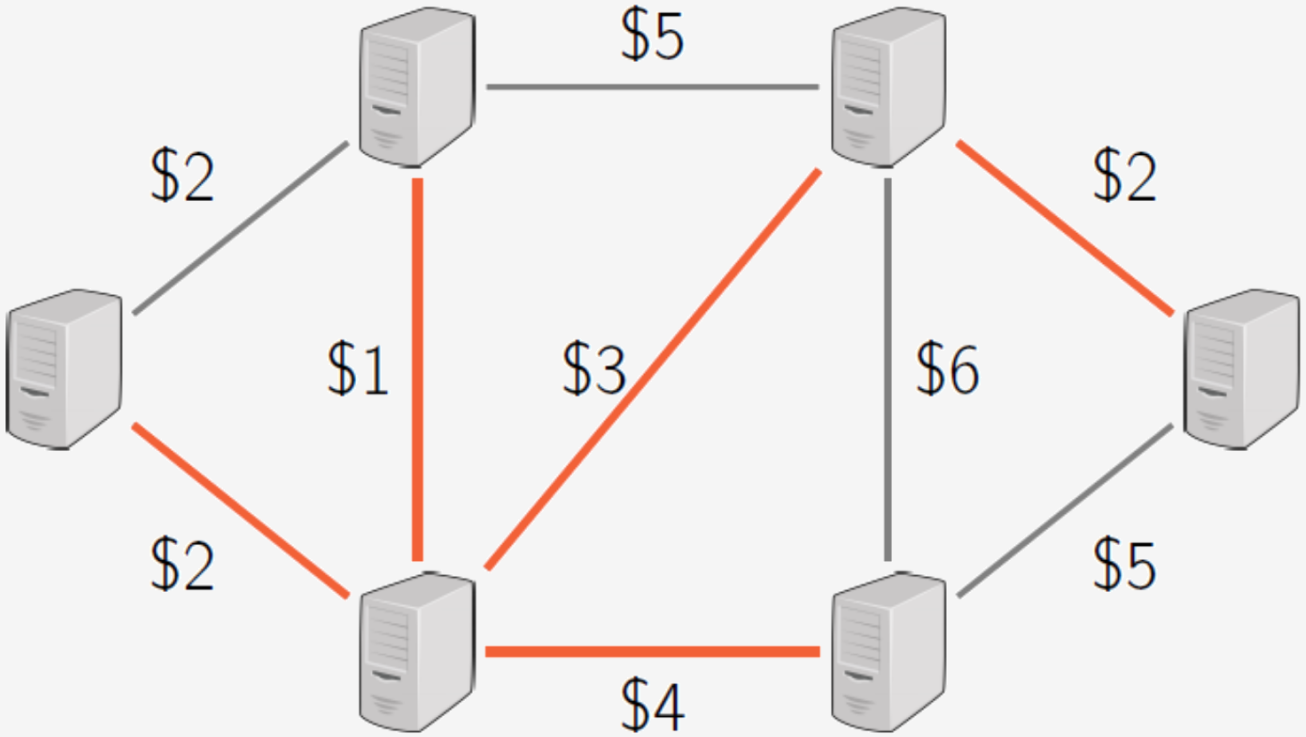
Connecting Computers by Wires



Connecting Computers by Wires

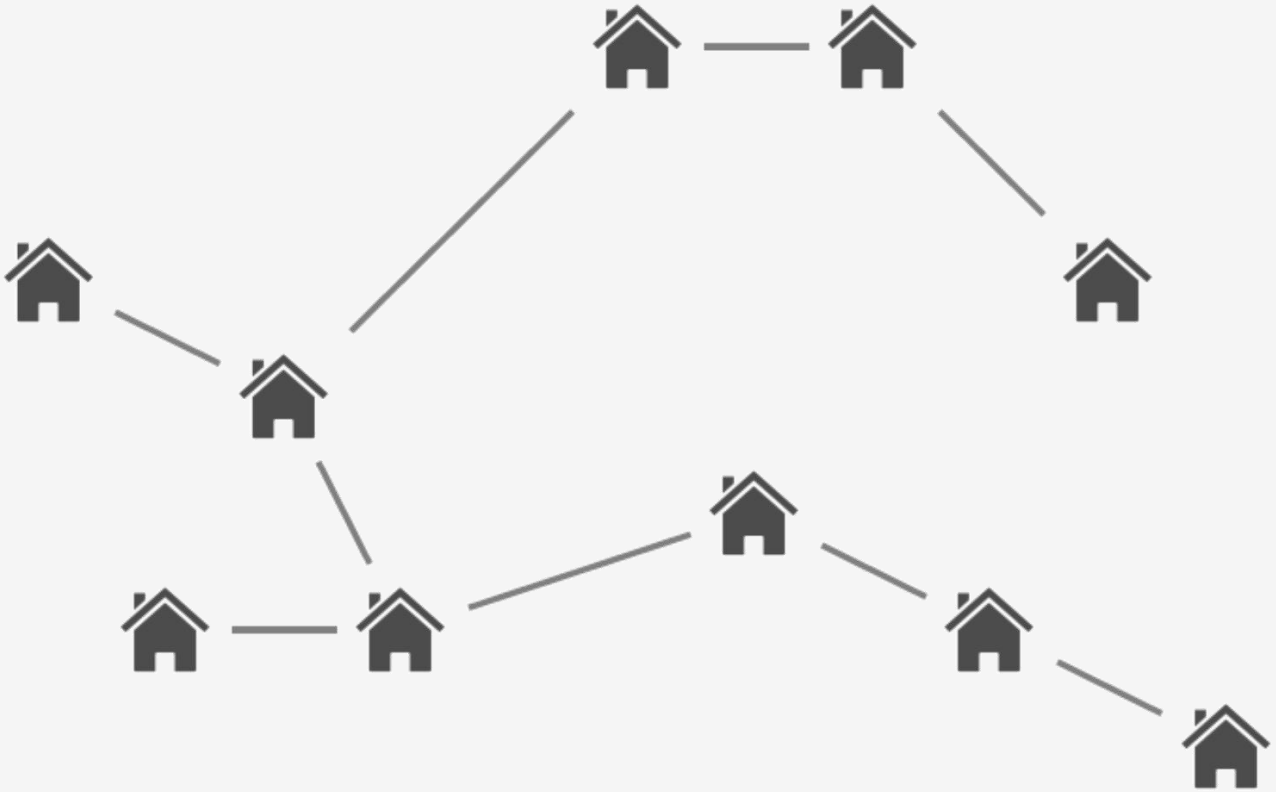


Connecting Computers by Wires



Building Roads

Building Roads



Minimum spanning tree (MST)

Input:

A connected, undirected graph G = (V , E) with positive edge weights.

Output:

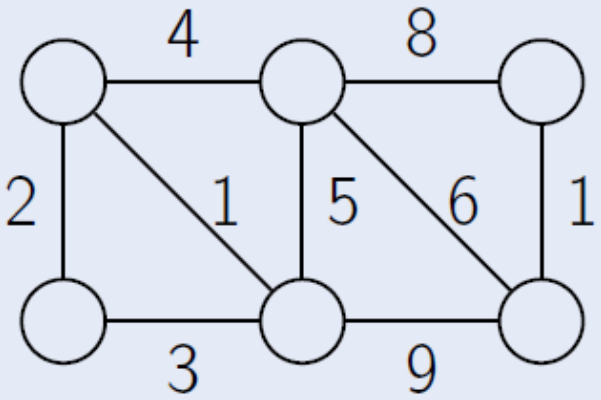
A subset of edges E’ ⊆ E of minimum total weight such that the graph (V , E’) is connected.

Remark

The set E’ always forms a tree.

Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle

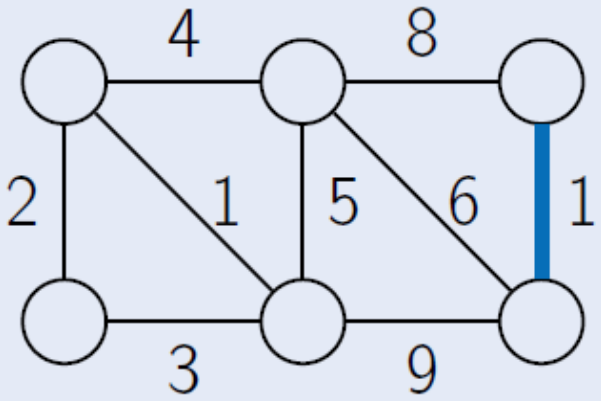


Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge

Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle

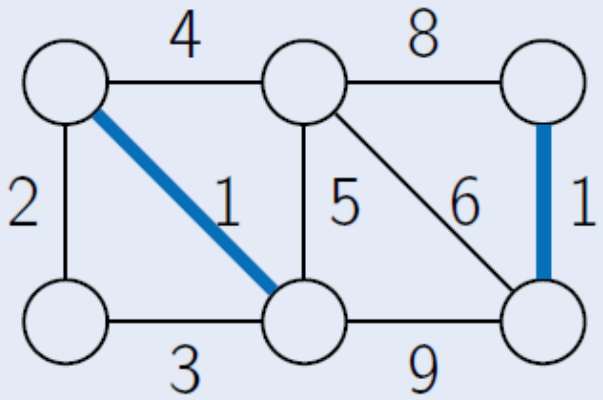


Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge

Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle

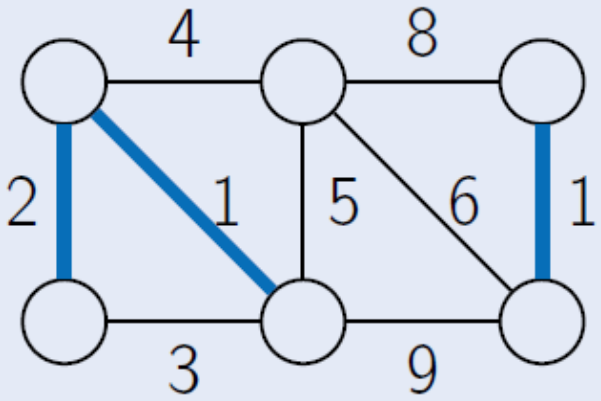


Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge

Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle

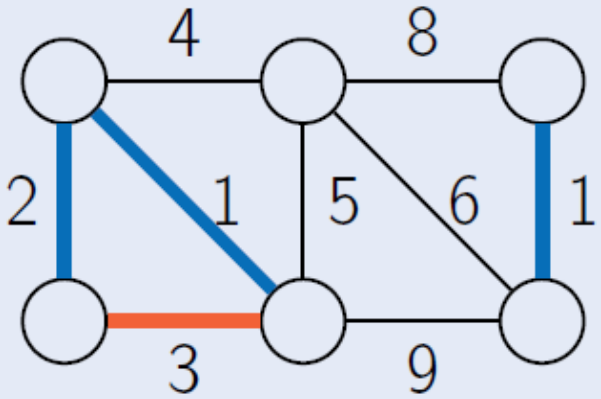


Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge

Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle

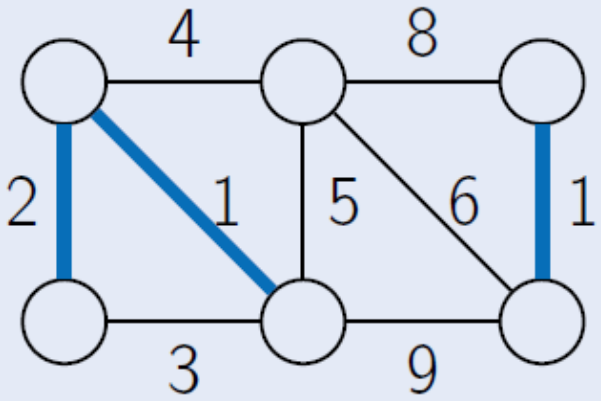


Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge

Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle

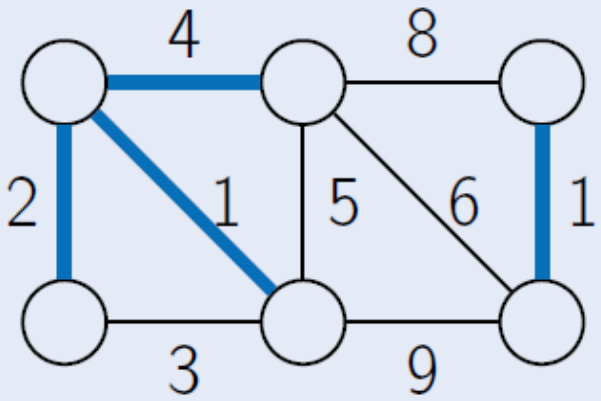


Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge

Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle

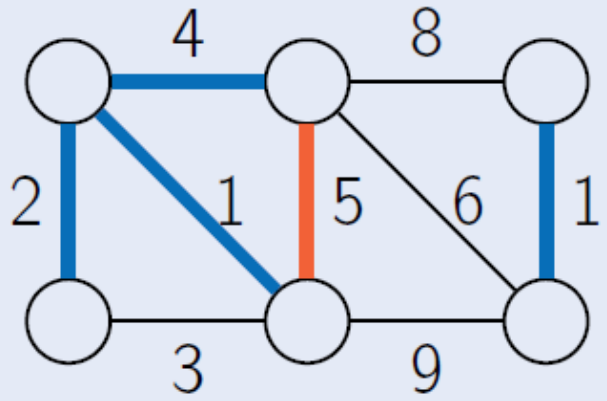


Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge

Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle

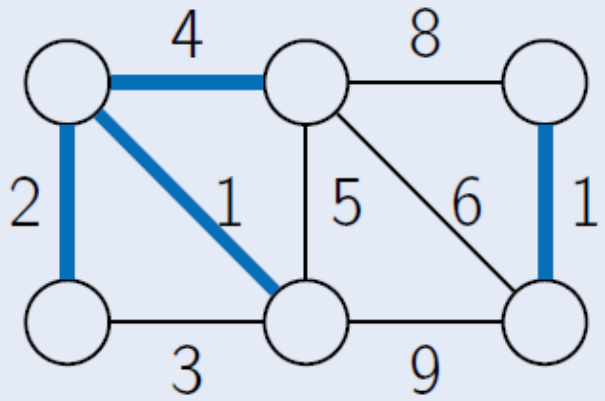


Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge

Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle

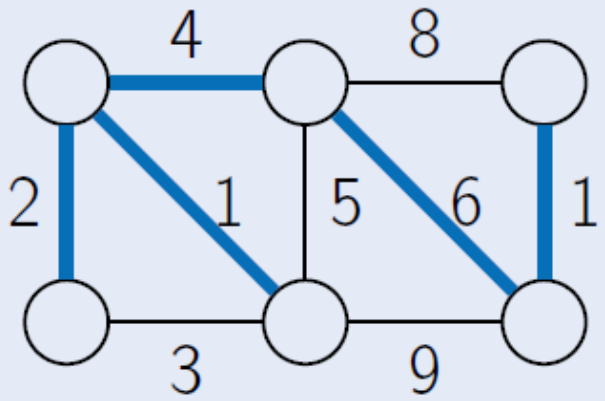


Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge

Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle

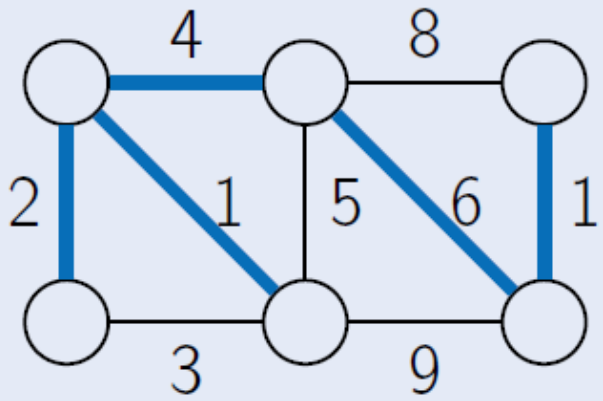


Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge

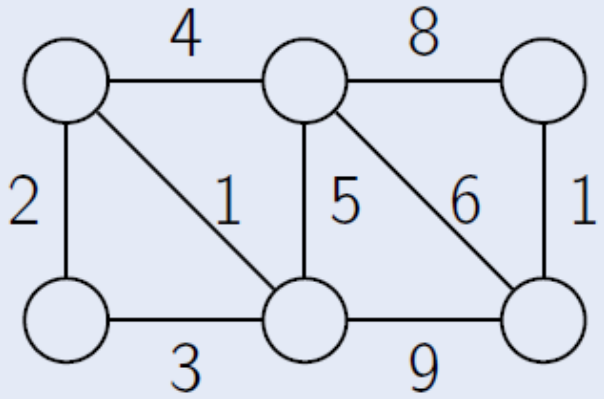
Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle



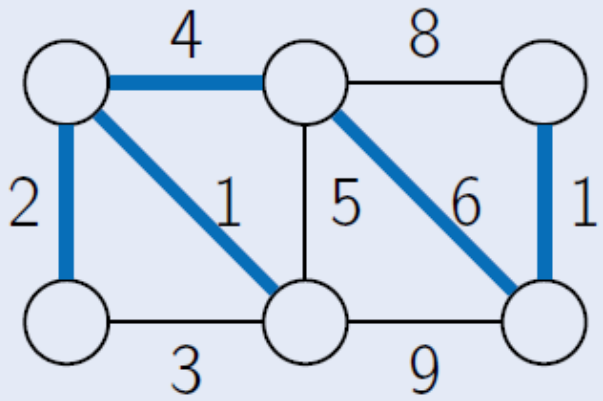
Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge



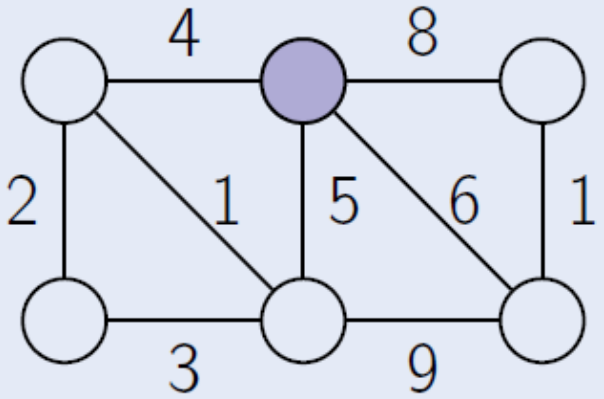
Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle



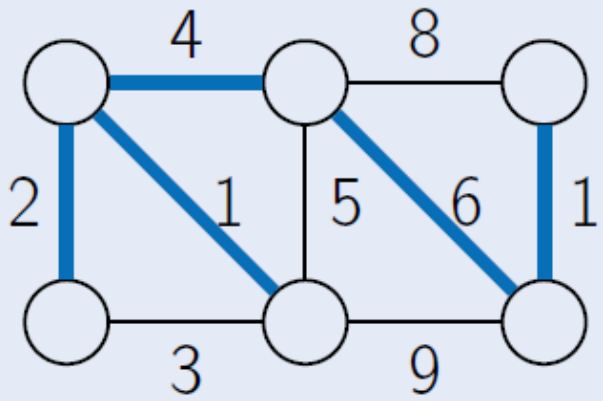
Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge



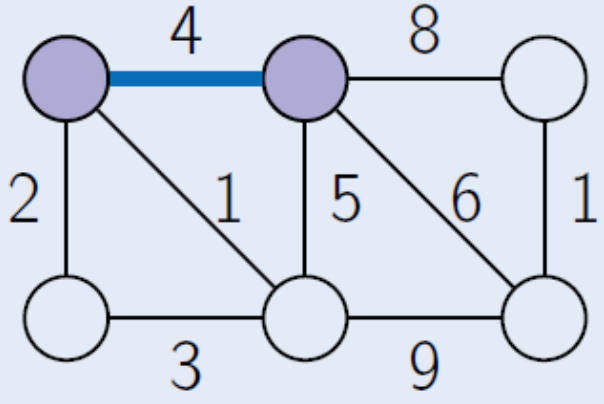
Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle



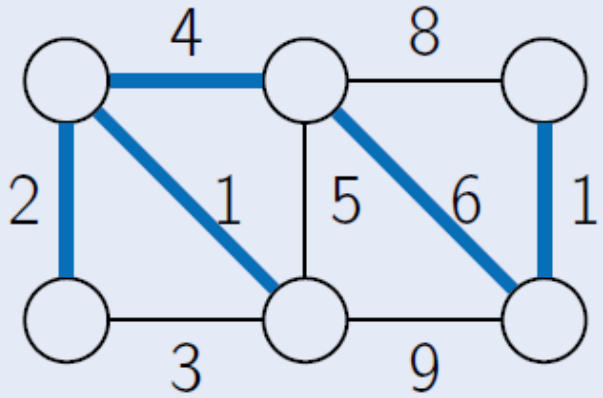
Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge



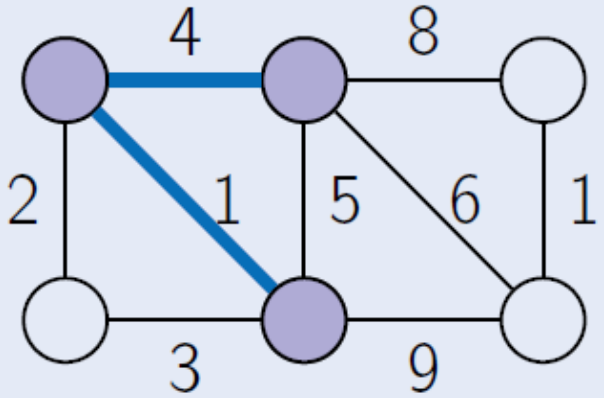
Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle



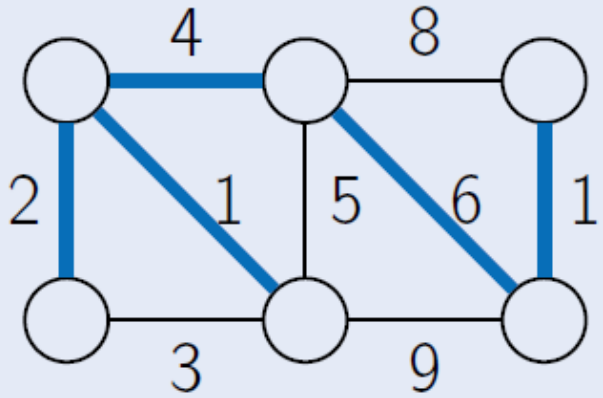
Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge



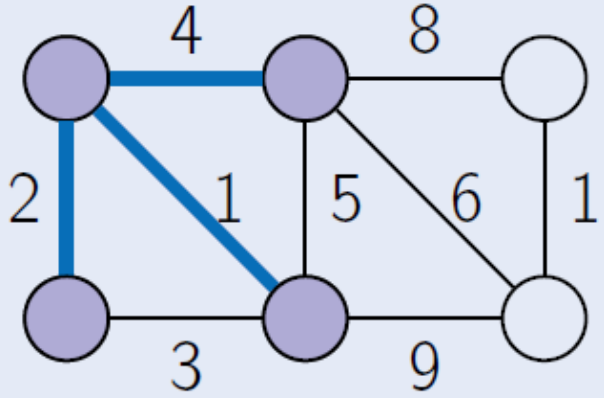
Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle



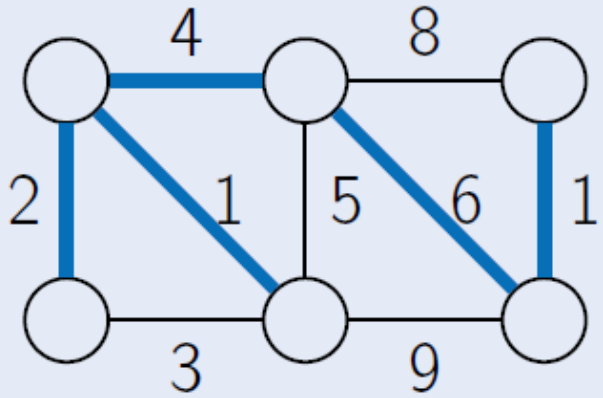
Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge



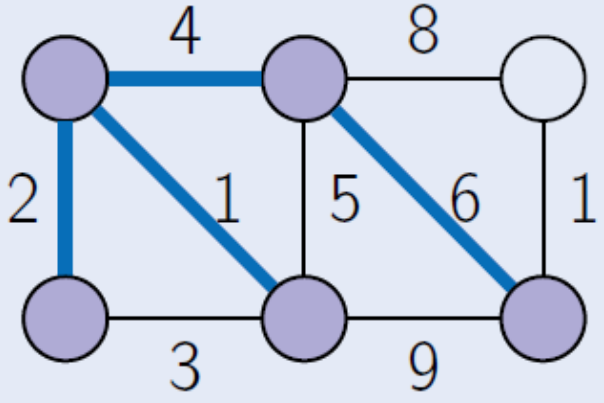
Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle



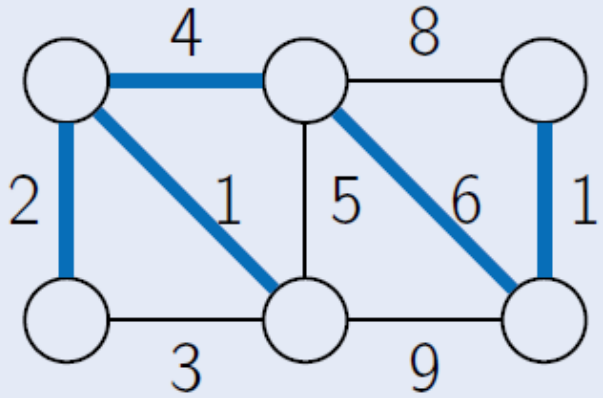
Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge



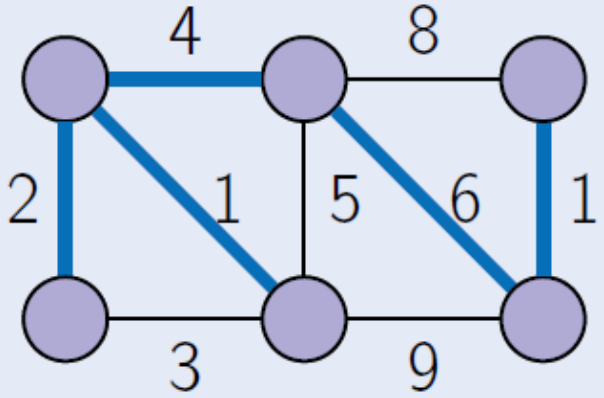
Kruskal's Algorithm

repeatedly add the next lightest edge if this doesn't produce a cycle



Prim's Algorithm

repeatedly attach a new vertex to the current tree by a lightest edge



Kruskal's Algorithm

Algorithm:

Repeatedly add to X the next lightest edge e that doesn't produce a cycle

At any point of time, the set X is a forest, that is, a collection of trees

Use disjoint sets data structure

Initially, each vertex lies in a separate set each set is the set of vertices of a connected component

To check whether the current edge {u, v} produces a cycle, we check whether u and v belong to the same set

Kruskal's Algorithm

```
Kruskal(G)
   for all u ∈ V :
       MakeSet(v)
   X ← empty set
   sort the edges E by weight
   for all {u, v} ∈ E in non-decreasing weight order:
       if Find(u) ≠ Find(v):
           add {u, v} to X
           Union(u, v)
   return X.
```

Prim's Algorithm

X is always a subtree, grows by one edge at each iteration

We add a lightest edge between a vertex of the tree and a vertex not in the tree

Very similar to Dijkstra's algorithm

Prim's Algorithm

```
Prim(G)
    for all u ∈ V:
        cost[u] ← ∞, parent[u] ← nil
    pick any initial vertex u0
    cost[u0] ← 0
    PrioQ ← MakeQueue(V) {priority is cost}
    while PrioQ is not empty:
        v ← ExtractMin(PrioQ)
        for all {v, z} ∈ E:
            If z ∈ PrioQ and cost[z] > w(v, z):
                cost[z] ← w(v, z), parent[z] ← v
                ChangePriority(PrioQ, z, cost[z]).
```

Summary

Kruskal:

Repeatedly add the next lightest edge if this doesn't produce a cycle; use disjoint sets to check whether the current edge joins two vertices from different components.

Prim:

Repeatedly attach a new vertex to the current tree by a lightest edge; use priority queue to quickly find the next lightest edge.