Lab 4 - Minimum Spanning Trees

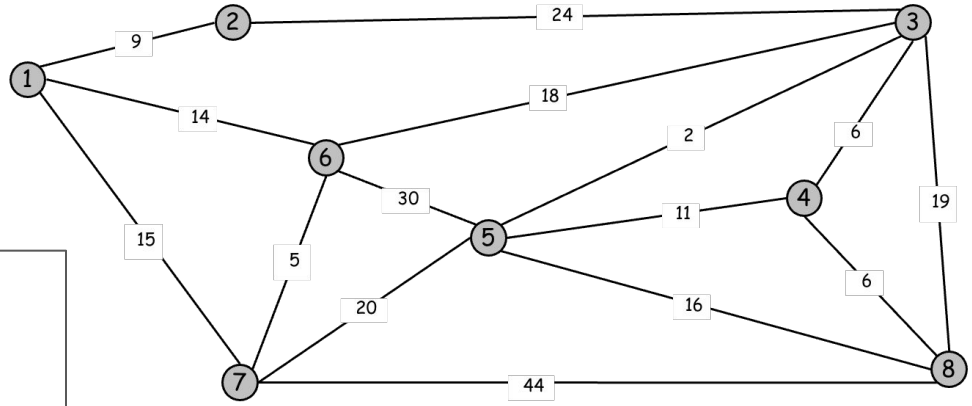
Oct 2, 2025

Kruskal’s Algorithm

MST = {}

Iteratively insert edges from E in order of increasing cost. If an edge will cause a cycle, do not add it.

Kruskal's Algorithm




```
kMST(G=(V, E)){
  T = {}
  place each vertex in a set by itself
  sort E in increasing order by weight
  for each ((u, v) in E in sorted order){
    // u and v in different sets
    if (find(u) != find(v)) {
      add (u, v) to T
      union(u, v)
    }
  }
}
```

- Clearly state what checks are performed
- Contents of the data structure at each iteration
- Current state of the solution

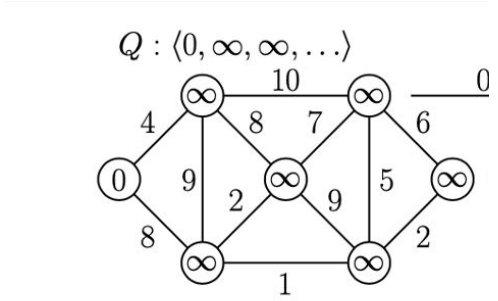
Prim’s Algorithm

Prim’s Algorithm: start with a root node s and try to greedily grow the tree outward. At each step, add the node that can be attached as cheaply as possible.

Maintains a set of $S \subseteq V$ on which a spanning tree has been constructed so far. At each iteration, we grow S by one node, adding the node with the minimum “attachment cost”

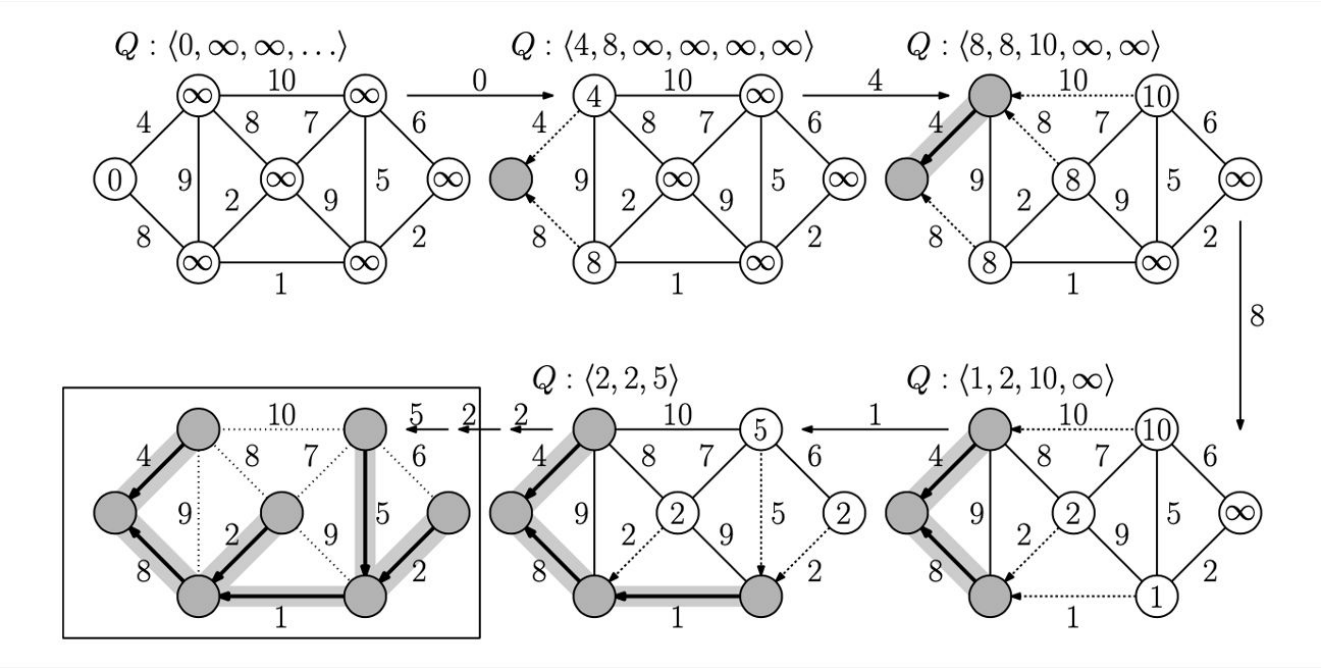
Prim's Algorithm

```
PrimMST(G, s){
  for each (v in V) {key[v] = ∞; mark[v] = F}
  key[s] = 0; pred[s] = null
  Q = priority queue of all vertices v keyed by key[v]
  while (Q is not empty) {
    u = extractMin from Q
    for each (v in Adj[u]) {
      if (!mark[v] && w(u, v) < key[v]) {
        key[v] = w(u, v)
        decrease v's key value in Q to key[v]
        pred[v] = u //keeps track of the tree
      }
    }
    mark[u] = T // finished
  }
}
```



- Clearly state what checks are performed
- Contents of the data structure at each iteration
- Current state of the solution

Prim's Algorithm



Shortest Path Tree

A Shortest Path Tree is a spanning tree T of G, such that the path distance from root v to any other vertex u in T is the shortest path distance from v to u in G

Midterm Discussion

Quiz on divide and conquer in a few weeks

- Tentative: Oct 29th
- Midterm: 20% of overall grade
- Quiz: 5% of overall grade