Minimum Spanning Tree

Outline and Reading

- Minimum Spanning Trees (§12.7)
  - Definitions
  - Cycle property
  - Partition property
- Prim-Jarník's Algorithm (§12.7.2)
- Kruskal's Algorithm (§12.7.1)

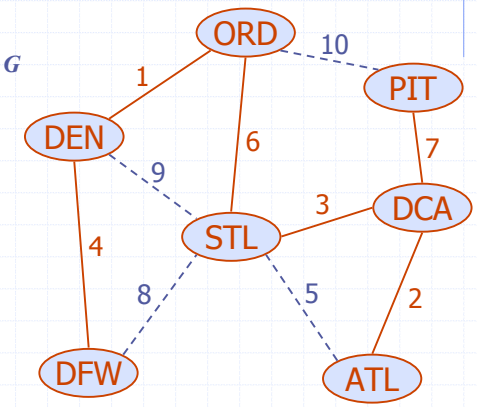
Minimum Spanning Tree

Spanning subgraph
- Subgraph of a graph G containing all the vertices of G

Spanning tree
- Spanning subgraph that is itself a (free) tree

Minimum spanning tree (MST)
- Spanning tree of a weighted graph with minimum total edge weight

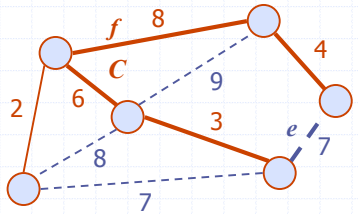
- Applications
  - Communications networks
  - Transportation networks



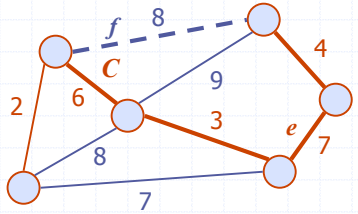
Cycle Property

Cycle Property:
- Let T be a minimum spanning tree of a weighted graph G
- Let e be an edge of G that is not in T and C let be the cycle formed by e with T
- For every edge f of C, $weight(f) \leq weight(e)$

Proof:
- By contradiction
- If $weight(f) > weight(e)$ we can get a spanning tree of smaller weight by replacing e with f



Replacing f with e yields a better spanning tree



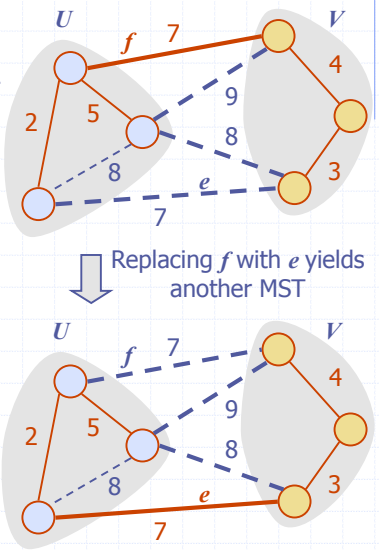
Partition Property

Partition Property:

- Consider a partition of the vertices of G into subsets U and V
- Let e be an edge of minimum weight across the partition
- There is a minimum spanning tree of G containing edge e

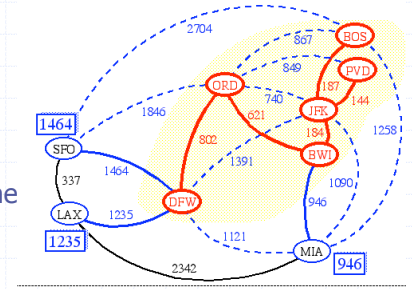
Proof:

- Let T be an MST of G
- If T does not contain e, consider the cycle C formed by e with T and let f be an edge of C across the partition
- By the cycle property, $weight(f) \leq weight(e)$
- Thus, $weight(f) = weight(e)$
- We obtain another MST by replacing f with e



Prim-Jarnik's Algorithm (§ 12.7.2)

- Similar to Dijkstra's algorithm (for a connected graph)
- We pick an arbitrary vertex s and we grow the MST as a cloud of vertices, starting from s
- We store with each vertex v a label $d(v)$ = the smallest weight of an edge connecting v to a vertex in the cloud
- At each step:
  - We add to the cloud the vertex u outside the cloud with the smallest distance label
  - We update the labels of the vertices adjacent to u



Prim-Jarnik's Algorithm (cont.)

- A priority queue stores the vertices outside the cloud
  - Key: distance
  - Element: vertex
- Locator-based methods
  - $insert(k,e)$ returns a locator
  - $replaceKey(l,k)$ changes the key of an item
- We store three labels with each vertex:
  - Distance
  - Parent edge in MST
  - Locator in priority queue

```
Algorithm PrimJarnikMST(G)
    Q ← new heap-based priority queue
    s ← a vertex of G
    for all v ∈ G.vertices()
        if v = s
            setDistance(v, 0)
        else
            setDistance(v, ∞)
        setParent(v, ∅)
        l ← Q.insert(getDistance(v), v)
        setLocator(v,l)
    while ¬Q.isEmpty()
        u ← Q.removeMin()
        for all e ∈ G.incidentEdges(u)
            z ← G.opposite(u,e)
            r ← weight(e)
            if r < getDistance(z)
                setDistance(z,r)
                setParent(z,e)
                Q.replaceKey(getLocator(z),r)
```