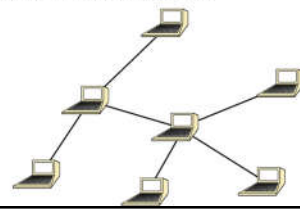
# Minimum Spanning Trees

Dr. G P Gupta

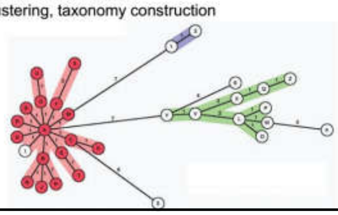
## Minimum Spanning Trees

- Find a minimum-cost set of edges that connect all vertices of a graph
- Applications
  - Connect “nodes” with a minimum of “wire”
    - Networking
    - Circuit design



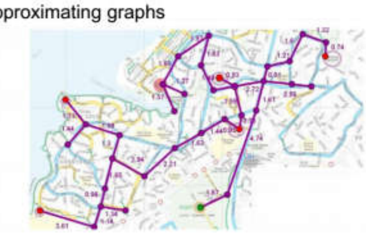
## Minimum Spanning Trees

- Find a minimum-cost set of edges that connect all vertices of a graph
- Applications
  - Collect nearby nodes
    - Clustering, taxonomy construction



## Minimum Spanning Trees

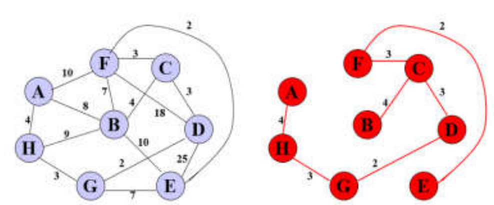
- Find a minimum-cost set of edges that connect all vertices of a graph
- Applications
  - Approximating graphs



## Minimum Spanning Trees

- A tree is an acyclic, undirected, connected graph
- A spanning tree of a graph is a tree containing all vertices from the graph
- A minimum spanning tree is a spanning tree, where the sum of the weights on the tree’s edges are minimal

## Minimum Spanning Trees




- A minimum spanning tree is a spanning tree, where the sum of the weights on the tree’s edges are minimal

## Minimum Spanning Trees

- Problem formulation
  - Given an undirected, weighted graph $G = (V, E)$ with weights $w(u, v)$ for each edge $(u, v) \in E$
  - Find an acyclic subset $T \subseteq E$ that connects all of the vertices $V$ and minimizes the total weight:

$$w(T) = \sum_{(u,v)\in T} w(u, v)$$

  - The minimum spanning tree is $(V, T)$
    - Minimum spanning tree may be not unique (can be more than one)

## Minimum Spanning Trees

- **Kruskal's Algorithms** and
- **Prim's Algorithms**
- Both Kruskal's and Prim's Algorithms ***work with undirected graphs***
- Both are ***greedy algorithms*** that produce optimal solutions
  - The greedy strategy advocates making the choice that is the best at the moment.

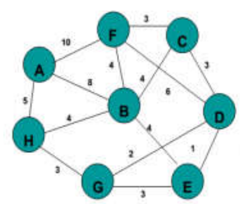
## Kruskal's algorithm

MST-KRUSKAL$(G, w)$

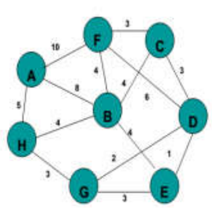
```
A = ∅
for each vertex v ∈ G.V
    MAKE-SET(v)
sort the edges of G.E into nondecreasing order by weight w
for each edge (u, v) ∈ G.E, taken in nondecreasing order by weight
    if FIND-SET(u) ≠ FIND-SET(v)
        A = A ∪ {(u, v)}
        UNION(u, v)
return A
```

## Kruskal's algorithm

- Two steps:
  - Sort edges by increasing edge weight
  - Select the first $|V|$ - 1 edges that do not generate a cycle




## Kruskal's algorithm: Example

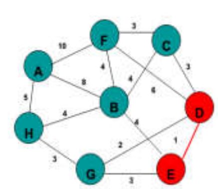



Sort the edges by increasing edge weight

| *edge* | $d_v$ | |
|---|---|---|
| (D,E) | 1 | |
| (D,G) | 2 | |
| (E,G) | 3 | |
| (C,D) | 3 | |
| (G,H) | 3 | |
| (C,F) | 3 | |
| (B,C) | 4 | |

| *edge* | $d_v$ | |
|---|---|---|
| (B,E) | 4 | |
| (B,F) | 4 | |
| (B,H) | 4 | |
| (A,H) | 5 | |
| (D,F) | 6 | |
| (A,B) | 8 | |
| (A,F) | 10 | |

## Kruskal's algorithm: Example

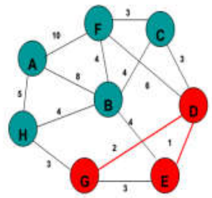



Select first $|V|$–1 edges which do not generate a cycle

| *edge* | $d_v$ | |
|---|---|---|
| (D,E) | 1 | √ |
| (D,G) | 2 | |
| (E,G) | 3 | |
| (C,D) | 3 | |
| (G,H) | 3 | |
| (C,F) | 3 | |
| (B,C) | 4 | |

| *edge* | $d_v$ | |
|---|---|---|
| (B,E) | 4 | |
| (B,F) | 4 | |
| (B,H) | 4 | |
| (A,H) | 5 | |
| (D,F) | 6 | |
| (A,B) | 8 | |
| (A,F) | 10 | |

## Kruskal's algorithm: Example

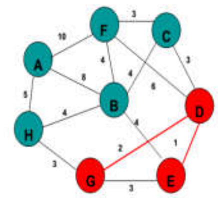



Select first |V|–1 edges which do not generate a cycle

| edge | $d_v$ | |
|---|---|---|
| (D,E) | 1 | √ |
| (D,G) | 2 | √ |
| (E,G) | 3 | |
| (C,D) | 3 | |
| (G,H) | 3 | |
| (C,F) | 3 | |
| (B,C) | 4 | |

| edge | $d_v$ | |
|---|---|---|
| (B,E) | 4 | |
| (B,F) | 4 | |
| (B,H) | 4 | |
| (A,H) | 5 | |
| (D,F) | 6 | |
| (A,B) | 8 | |
| (A,F) | 10 | |

## Kruskal's algorithm: Example



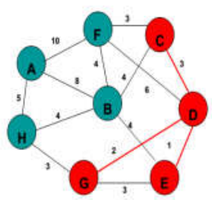

Select first |V|–1 edges which do not generate a cycle

| edge | $d_v$ | |
|---|---|---|
| (D,E) | 1 | √ |
| (D,G) | 2 | √ |
| (E,G) | 3 | χ |
| (C,D) | 3 | |
| (G,H) | 3 | |
| (C,F) | 3 | |
| (B,C) | 4 | |

| edge | $d_v$ | |
|---|---|---|
| (B,E) | 4 | |
| (B,F) | 4 | |
| (B,H) | 4 | |
| (A,H) | 5 | |
| (D,F) | 6 | |
| (A,B) | 8 | |
| (A,F) | 10 | |

Accepting edge (E,G) would create a cycle

## Kruskal's algorithm: Example

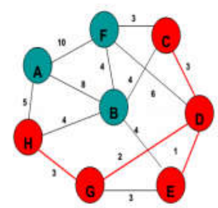



Select first |V|–1 edges which do not generate a cycle

| edge | $d_v$ | |
|---|---|---|
| (D,E) | 1 | √ |
| (D,G) | 2 | √ |
| (E,G) | 3 | χ |
| (C,D) | 3 | √ |
| (G,H) | 3 | |
| (C,F) | 3 | |
| (B,C) | 4 | |

| edge | $d_v$ | |
|---|---|---|
| (B,E) | 4 | |
| (B,F) | 4 | |
| (B,H) | 4 | |
| (A,H) | 5 | |
| (D,F) | 6 | |
| (A,B) | 8 | |
| (A,F) | 10 | |

## Kruskal's algorithm: Example

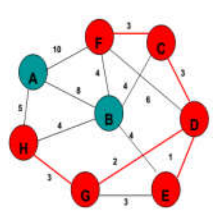



Select first |V|–1 edges which do not generate a cycle

| edge | $d_v$ | |
|---|---|---|
| (D,E) | 1 | √ |
| (D,G) | 2 | √ |
| (E,G) | 3 | χ |
| (C,D) | 3 | √ |
| (G,H) | 3 | √ |
| (C,F) | 3 | |
| (B,C) | 4 | |

| edge | $d_v$ | |
|---|---|---|
| (B,E) | 4 | |
| (B,F) | 4 | |
| (B,H) | 4 | |
| (A,H) | 5 | |
| (D,F) | 6 | |
| (A,B) | 8 | |
| (A,F) | 10 | |

## Kruskal's algorithm: Example

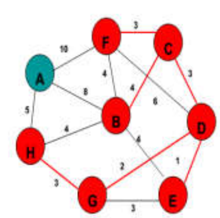



Select first |V|–1 edges which do not generate a cycle

| edge | $d_v$ | |
|---|---|---|
| (D,E) | 1 | √ |
| (D,G) | 2 | √ |
| (E,G) | 3 | χ |
| (C,D) | 3 | √ |
| (G,H) | 3 | √ |
| (C,F) | 3 | √ |
| (B,C) | 4 | |

| edge | $d_v$ | |
|---|---|---|
| (B,E) | 4 | |
| (B,F) | 4 | |
| (B,H) | 4 | |
| (A,H) | 5 | |
| (D,F) | 6 | |
| (A,B) | 8 | |
| (A,F) | 10 | |

## Kruskal's algorithm: Example

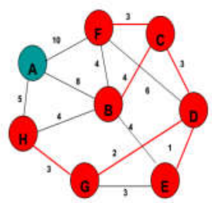



Select first |V|–1 edges which do not generate a cycle

| edge | $d_v$ | |
|---|---|---|
| (D,E) | 1 | √ |
| (D,G) | 2 | √ |
| (E,G) | 3 | χ |
| (C,D) | 3 | √ |
| (G,H) | 3 | √ |
| (C,F) | 3 | √ |
| (B,C) | 4 | √ |

| edge | $d_v$ | |
|---|---|---|
| (B,E) | 4 | |
| (B,F) | 4 | |
| (B,H) | 4 | |
| (A,H) | 5 | |
| (D,F) | 6 | |
| (A,B) | 8 | |
| (A,F) | 10 | |

## Kruskal's algorithm: Example

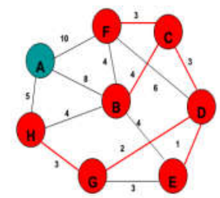



Select first |V|–1 edges which do not generate a cycle

| edge | $d_v$ | |
|---|---|---|
| (D,E) | 1 | √ |
| (D,G) | 2 | √ |
| (E,G) | 3 | χ |
| (C,D) | 3 | √ |
| (G,H) | 3 | √ |
| (C,F) | 3 | √ |
| (B,C) | 4 | √ |

| edge | $d_v$ | |
|---|---|---|
| (B,E) | 4 | χ |
| (B,F) | 4 | |
| (B,H) | 4 | |
| (A,H) | 5 | |
| (D,F) | 6 | |
| (A,B) | 8 | |
| (A,F) | 10 | |

## Kruskal's algorithm: Example

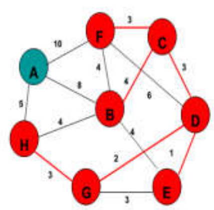



Select first |V|–1 edges which do not generate a cycle

| edge | $d_v$ | |
|---|---|---|
| (D,E) | 1 | √ |
| (D,G) | 2 | √ |
| (E,G) | 3 | χ |
| (C,D) | 3 | √ |
| (G,H) | 3 | √ |
| (C,F) | 3 | √ |
| (B,C) | 4 | √ |

| edge | $d_v$ | |
|---|---|---|
| (B,E) | 4 | χ |
| (B,F) | 4 | χ |
| (B,H) | 4 | |
| (A,H) | 5 | |
| (D,F) | 6 | |
| (A,B) | 8 | |
| (A,F) | 10 | |

## Kruskal's algorithm: Example

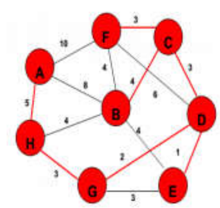



Select first |V|–1 edges which do not generate a cycle

| edge | $d_v$ | |
|---|---|---|
| (D,E) | 1 | √ |
| (D,G) | 2 | √ |
| (E,G) | 3 | χ |
| (C,D) | 3 | √ |
| (G,H) | 3 | √ |
| (C,F) | 3 | √ |
| (B,C) | 4 | √ |

| edge | $d_v$ | |
|---|---|---|
| (B,E) | 4 | χ |
| (B,F) | 4 | χ |
| (B,H) | 4 | χ |
| (A,H) | 5 | |
| (D,F) | 6 | |
| (A,B) | 8 | |
| (A,F) | 10 | |

## Kruskal's algorithm: Example

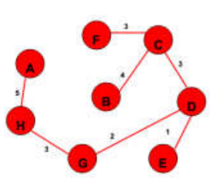



Select first |V|–1 edges which do not generate a cycle

| edge | $d_v$ | |
|---|---|---|
| (D,E) | 1 | √ |
| (D,G) | 2 | √ |
| (E,G) | 3 | χ |
| (C,D) | 3 | √ |
| (G,H) | 3 | √ |
| (C,F) | 3 | √ |
| (B,C) | 4 | √ |

| edge | $d_v$ | |
|---|---|---|
| (B,E) | 4 | χ |
| (B,F) | 4 | χ |
| (B,H) | 4 | χ |
| (A,H) | 5 | √ |
| (D,F) | 6 | |
| (A,B) | 8 | |
| (A,F) | 10 | |

## Kruskal's algorithm: Example




Select first |V|–1 edges which do not generate a cycle

| edge | $d_v$ | |
|---|---|---|
| (D,E) | 1 | √ |
| (D,G) | 2 | √ |
| (E,G) | 3 | χ |
| (C,D) | 3 | √ |
| (G,H) | 3 | √ |
| (C,F) | 3 | √ |
| (B,C) | 4 | √ |

| edge | $d_v$ | |
|---|---|---|
| (B,E) | 4 | χ |
| (B,F) | 4 | χ |
| (B,H) | 4 | χ |
| (A,H) | 5 | √ |
| (D,F) | 6 | not considered |
| (A,B) | 8 | not considered |
| (A,F) | 10 | not considered |

**Done**

**Total Cost = *21***

## Analysis of Kruskal's algorithm

- Depends on how we implement the disjoint-set data structure.

## Kruskal's algorithm

MST-KRUSKAL$(G, w)$

1 $A = \emptyset$
2 **for** each vertex $v \in G.V$
3 MAKE-SET$(v)$
4 sort the edges of $G.E$ into nondecreasing order by weight $w$
5 **for** each edge $(u, v) \in G.E$, taken in nondecreasing order by weight
6 **if** FIND-SET$(u) \neq$ FIND-SET$(v)$
7 $A = A \cup \{(u, v)\}$
8 UNION$(u, v)$
9 **return** $A$

## Analysis of Kruskal

- Lines 1-3 (initialization): O(V)
- Line 4 (sorting): O(E lg E)
- Lines 6-8 (set-operation): O(E log E)
- **Total:** O(E log E)

## Prim's algorithm

- Work with nodes (instead of edges)

**Two steps**

- Select node with minimum distance
- Update distances of adjacent, unselected nodes

•

the attribute $v.key$ is the minimum weight of any edge connecting $v$ to a vertex in the tree; by convention, $v.key = \infty$ if there is no such edge.

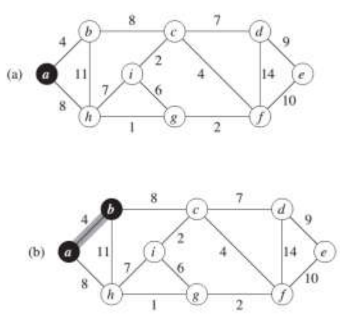
attribute $v.\pi$ names the parent of $v$ in the tree.

## Prim's algorithm

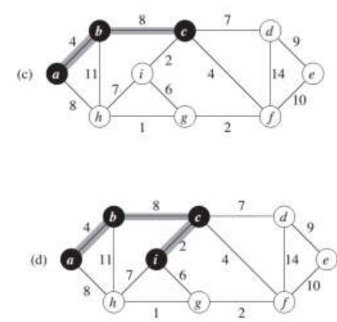
MST-PRIM$(G, w, r)$

**for** each $u \in G.V$
  $u.key = \infty$
  $u.\pi =$ NIL
$r.key = 0$
$Q = G.V$
**while** $Q \neq \emptyset$
  $u =$ EXTRACT-MIN$(Q)$
  **for** each $v \in G.Adj[u]$
    **if** $v \in Q$ and $w(u, v) < v.key$
      $v.\pi = u$
      $v.key = w(u, v)$

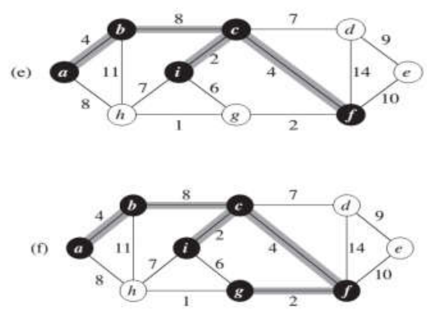
## Prim's algorithm: Example



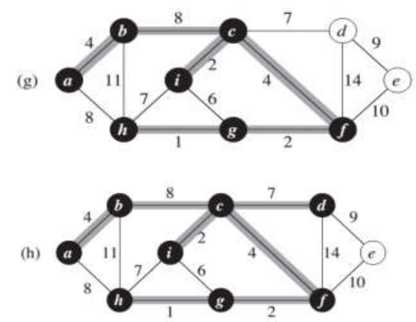
## Prim's algorithm: Example



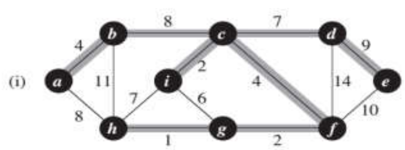
## Prim's algorithm: Example



## Prim's algorithm: Example



## Prim's algorithm: Example



At the end, $\{(v, \pi[v])\}$ forms the MST.

## Analysis of Prim

- Extracting the vertex from the queue: $O(\lg n)$
- For each incident edge, decreasing the key of the neighboring vertex: $O(\lg n)$ where $n = |V|$
- The other steps are constant time.
- The overall running time is, where $e = |E|$

$$T(n) = \sum_{u \in V}(\lg n + \deg(u) \lg n) = \sum_{u \in V}(1 + \deg(u)) \lg n$$
$$= \lg n \, (n + 2e) = O((n + e) \lg n)$$

**Essentially same as Kruskal's:** ***O*((*n*+*e*) lg *n*)** time