Lecture

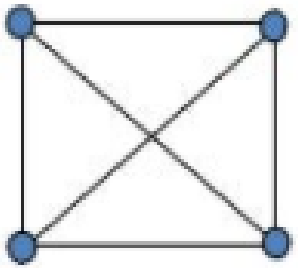
Minimum Spanning trees

Prim’s algorithm

Spanning Trees

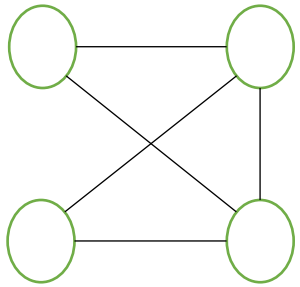
A connected sub graph H of given Graph G is said to be spanning tree iff
1. H contains all vertices of G.
2. H should contain (n-1) edges if G contains n vertices.

Question 1 Find number of spanning trees for the following graph:



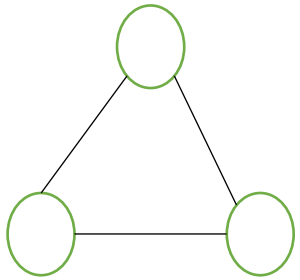
Spanning Trees

Question 2 Find number of spanning trees for the following graph:



Spanning Trees

Question 3 Find number of spanning trees for the following graph:



Spanning Trees

The number of spanning trees in a complete graph:

$$\mathbf{ST}(K_n) = n^{n-2}$$

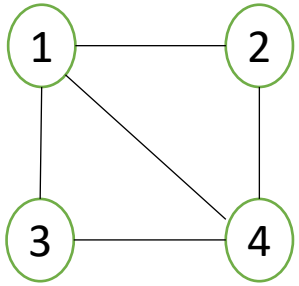
$\mathbf{ST}(K_2) =$

$\mathbf{ST}(K_3) =$

$\mathbf{ST}(K_4) =$

$\mathbf{ST}(K_5) =$

Spanning Trees: Kirchoff's Theorem



Find number of spanning trees for the following graph:

Step 1: Find adjacency matrix for the given graph.

$$M = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Step 2: Replace all diagonal 0's by its' degree and non-diagonal 1's by -1.

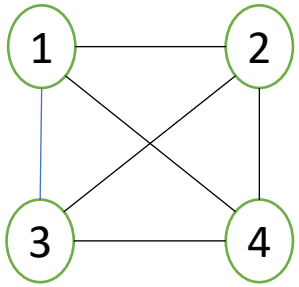
$$M = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & 0 & -1 \\ -1 & 0 & 2 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$

Step 3: Number of spanning tree = co-factor of any element.

$$\text{Co-factor } (M_{11}) = \begin{vmatrix} 2 & 0 & -1 \\ 0 & 2 & -1 \\ -1 & -1 & 3 \end{vmatrix}$$

$$= 2(6-1)-1\,(2) = 8$$

Spanning Trees: Kirchoff's Theorem



Find number of spanning trees for the following graph:

Step 1: Find adjacency matrix for the given graph.

$$M = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

Step 2: Replace all diagonal 0's by its' degree and non-diagonal 1's by -1.

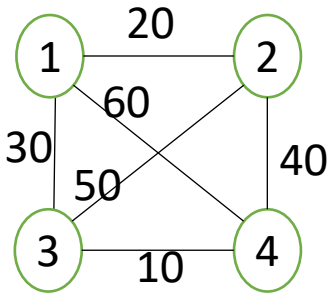
$$M = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 3 & -1 & -1 \\ -1 & -1 & 3 & -1 \\ -1 & -1 & -1 & 3 \end{bmatrix}$$

Step 3: Number of spanning tree = co-factor of any element.

$$\text{Co-factor } (M_{11}) = \begin{vmatrix} 3 & -1 & -1 \\ -1 & 3 & -1 \\ -1 & -1 & 3 \end{vmatrix}$$

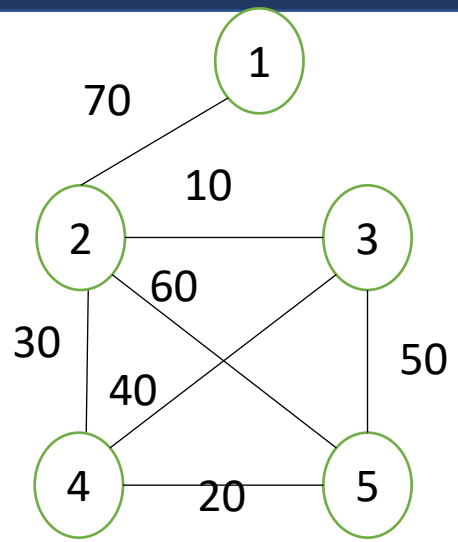
$$= 3(9-1) + 1\,(-3-1) - 1(1+3) = 16$$

Minimum Cost Spanning Tree



Using Prim's and Kruskal's algorithm we can find out the minimum cost spanning tree

Minimum Cost Spanning Tree: Prim's Algorithm

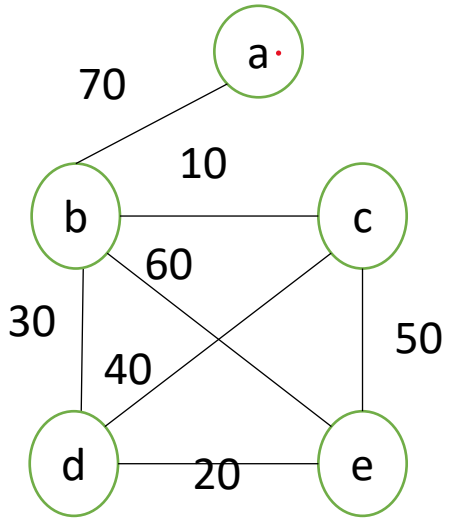


Step 1: Choose any vertex and find adjacent of that vertex and take minimum of them

Step 2: Find adjacency vertices of new vertex and take minimum of these adjacent and previous.

Step 3: Repeat step 2 until all the vertices are not covered.

Minimum Cost Spanning Tree: Prim's Algorithm



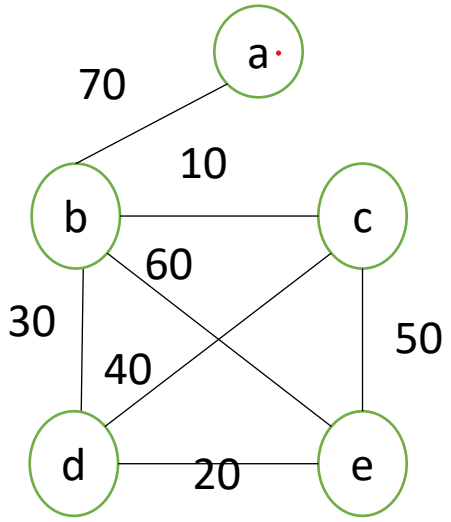
The edges in set A always form a single tree

Starts from an arbitrary "root": $V_A = \{a\}$

At each step:

- Find a light edge crossing $(V_A, V - V_A)$
- Add this edge to A
- Repeat until the tree spans all vertices

Minimum Cost Spanning Tree: Prim's Algorithm



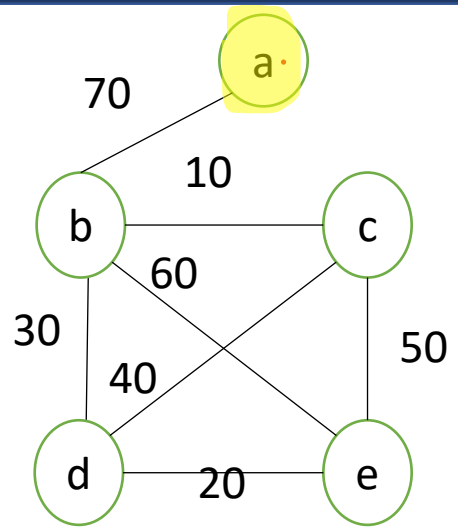
The edges in set A always form a single tree

Starts from an arbitrary "root": $V_A = \{a\}$

At each step:

- Find a light edge crossing $(V_A, V - V_A)$
- Add this edge to A
- Repeat until the tree spans all vertices

Minimum Cost Spanning Tree: Prim's Algorithm



Nodes distances in priority Queue : 0 ∞ ∞ ∞ ∞

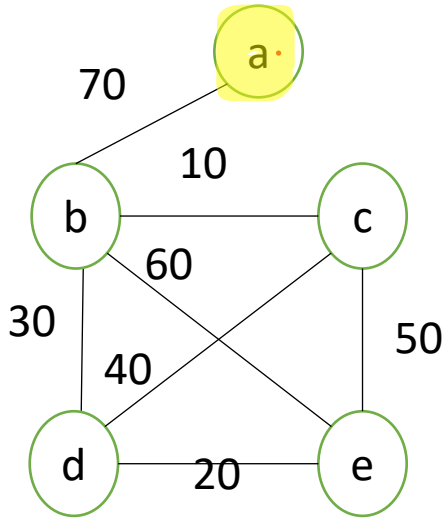
Q = {a, b, c, d, e}

$V_A = \varnothing$

Extract-MIN(Q) $\Rightarrow$ a

V_A = {a}

Minimum Cost Spanning Tree: Prim's Algorithm



key [b] = 70 π [b] = a

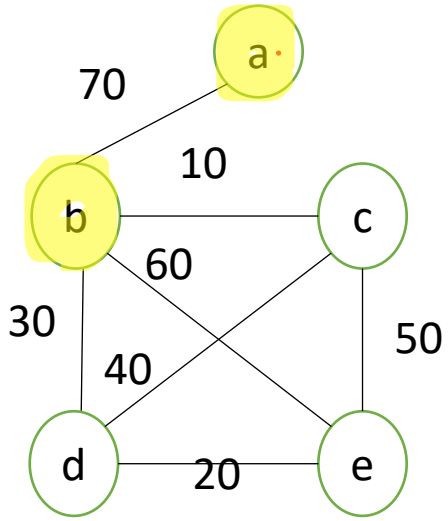
70 ∞ ∞ ∞ ∞

Q = {b, c, d, e} V_A = {a}

Extract-MIN(Q) $\Rightarrow$ b

V_A = {a, b}

Minimum Cost Spanning Tree: Prim's Algorithm



key [b] = 70 π [b] = a

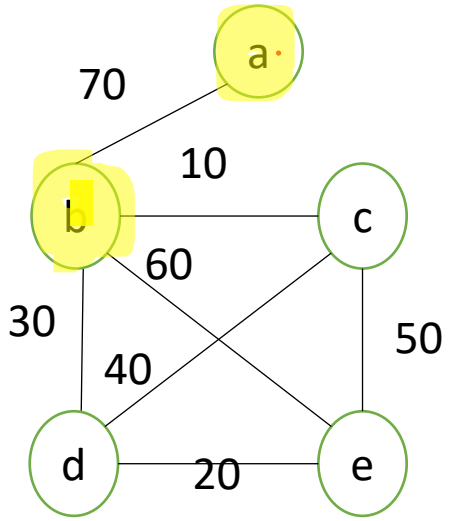
70 ∞ ∞ ∞ ∞

Q = {b, c, d, e} V_A = {a}

Extract-MIN(Q) $\Rightarrow$ b

V_A = {a, b}

Minimum Cost Spanning Tree: Prim's Algorithm



key [c] = 10 π [c] = b

key [d] = 30 π [d] = b

key [e] = 60 π [e] = b

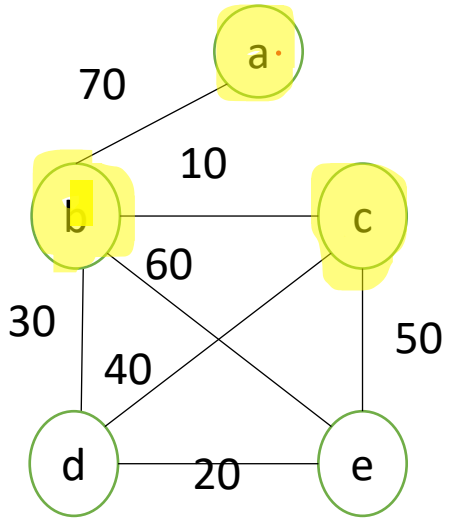
10, 30, 60

Q = {c, d, e} V_A = {a, b}

Extract-MIN(Q) $\Rightarrow$ c

V_A = {a, b, c}

Minimum Cost Spanning Tree: Prim's Algorithm



key [c] = 10 π [c] = b

key [d] = 30 π [d] = b

key [e] = 60 π [e] = b

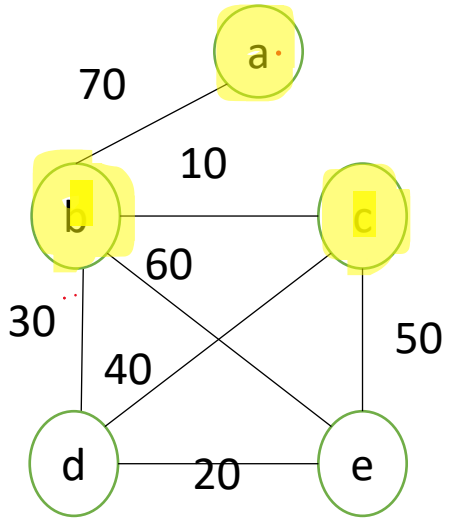
10, 30, 60

Q = {c, d, e} V_A = {a, b}

Extract-MIN(Q) $\Rightarrow$ c

V_A = {a, b, c}

Minimum Cost Spanning Tree: Prim's Algorithm



key [d] = 30 π [d] = b

key [e] = 50 π [e] = c

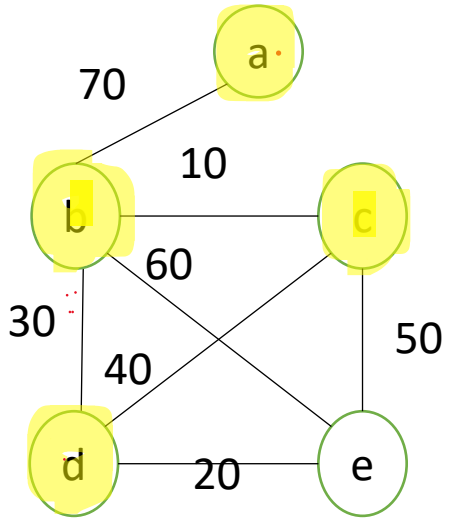
30, 50

Q = { d, e} V_A = {a, b, c}

Extract-MIN(Q) $\Rightarrow$ d

V_A = {a, b, c, d}

Minimum Cost Spanning Tree: Prim's Algorithm



key [d] = 30 π [d] = b

key [e] = 50 π [e] = c

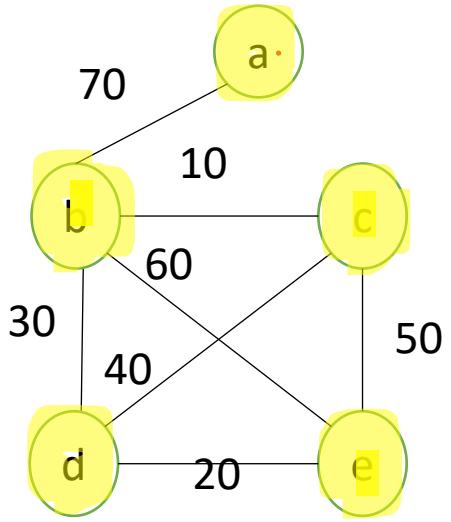
30, 50

Q = { d, e} V_A = {a, b, c}

Extract-MIN(Q) $\Rightarrow$ d

V_A = {a, b, c, d}

Minimum Cost Spanning Tree: Prim's Algorithm



key [e] = 20 π [e] = d

20

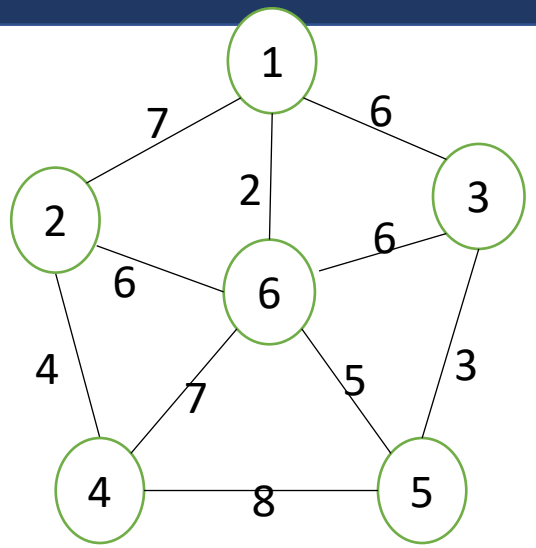
Q = { e} V_A = {a, b, c, d}

Extract-MIN(Q) $\Rightarrow$ e

V_A = {a, b, c, d, e}

Q = $\varnothing$

Minimum Cost Spanning Tree: Prim's Algorithm

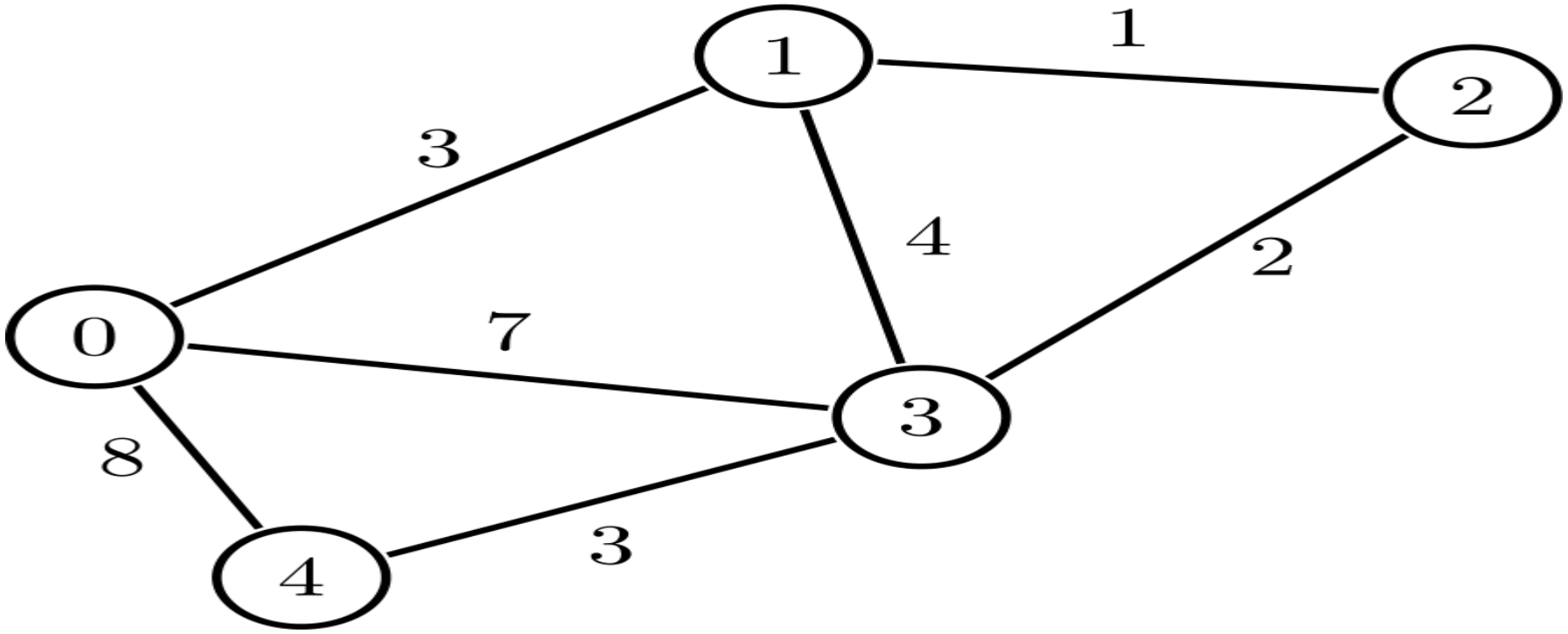


Minimum Cost Spanning Tree: Prim's Algorithm

```
MST-PRIM(G, w, r)
 1  for each u ∈ G.V
 2      u.key = ∞
 3      u.π = NIL
 4  r.key = 0
 5  Q = G.V
 6  while Q ≠ ∅
 7      u = EXTRACT-MIN(Q)
 8      for each v ∈ G.Adj[u]
 9          if v ∈ Q and w(u, v) < v.key
10              v.π = u
11              v.key = w(u, v)
```

Time Complexity:
Using Binary Min Heap: $O[(V + E)logV]$
Using Fibonacci Min Heap: $O[E + VlogV]$
Using Binomial Min Heap: $O[(V + E)]$

Minimum Cost Spanning Tree: Prim's Algorithm



Thank You