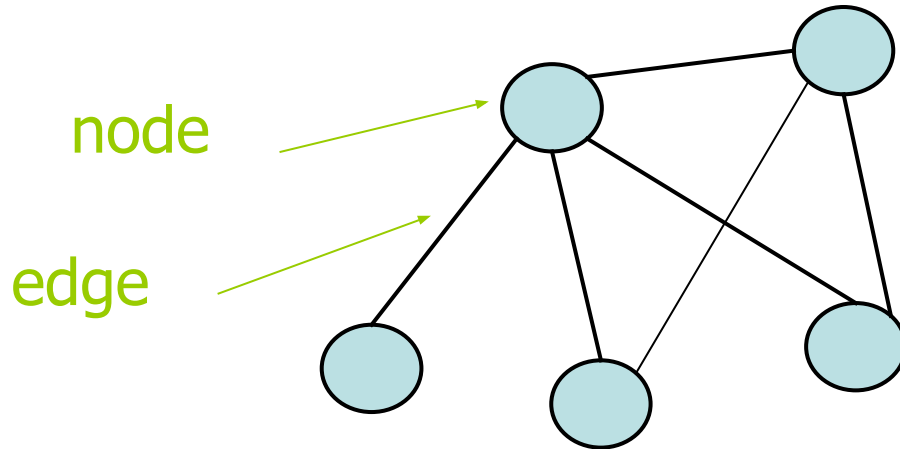


Graphs and Spanning Trees

**Computer Science and Engineering
Indian Institute of Information Technology Sri City**

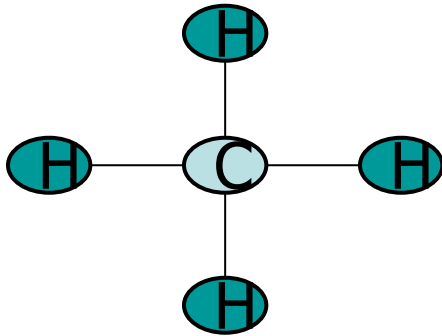
What is a graph?

- Graph represents relationship among the data items
- A graph G consists of
 - a set V of nodes (vertices)
 - a set E of edges (each edge connects two nodes)

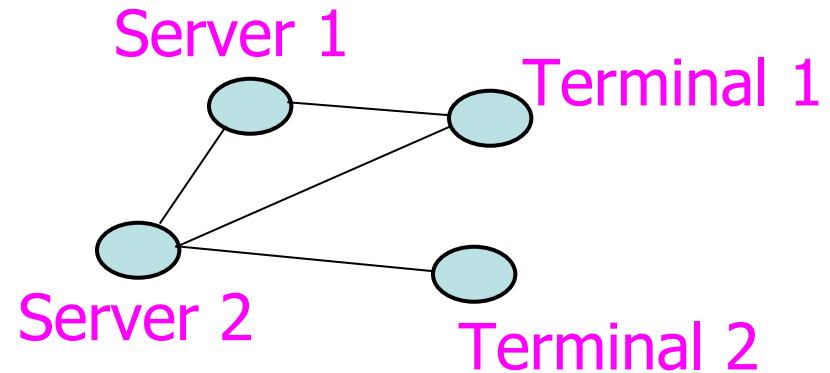


Examples of graphs

Molecular Structure



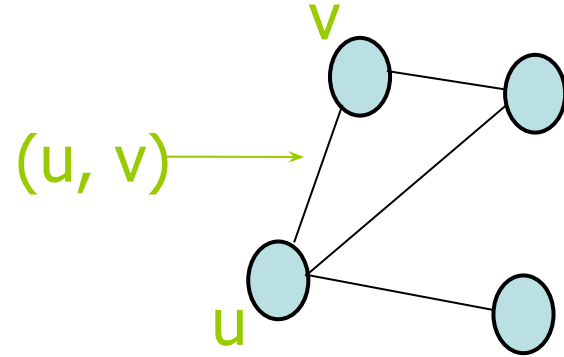
Computer Network



Other examples: electrical and communication networks, airline routes, flow chart, etc.

Formal Definition of graph

- The set of nodes is denoted as V
- For any nodes u and v , if u and v are connected by an edge, such edge is denoted as (u, v)
- The set of edges is denoted as E
- A **graph** G is defined as a pair (V, E)

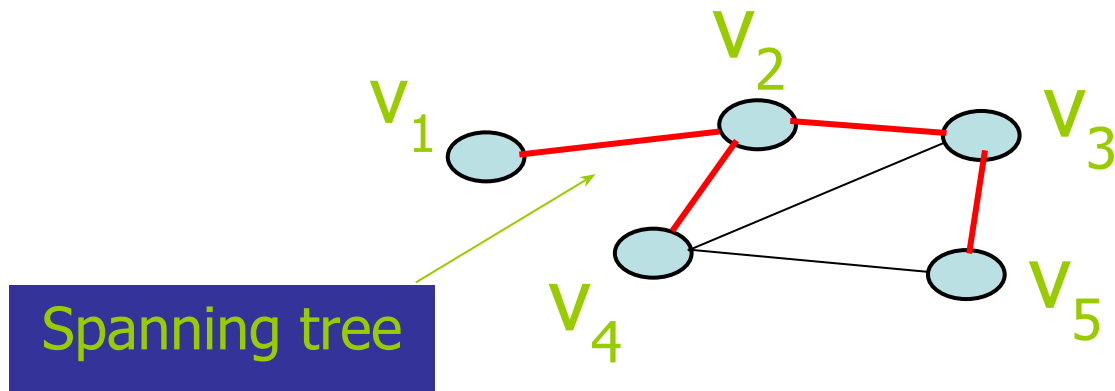


Problems related to Graph

- Spanning Tree
- Minimum Spanning Tree
- Shortest Path

Spanning Tree

- Given a connected undirected graph G , a **spanning tree** of G is a subgraph of G that contains all of G 's nodes and enough of its edges to form a tree.



$$E = \{(v_1, v_2), (v_2, v_3), (v_2, v_4), (v_3, v_5)\}$$

DFS spanning tree

Algorithm dfsSpanningTree(v)

mark v as visited;

for (each unvisited node u adjacent to v) {

 mark the edge from u to v;

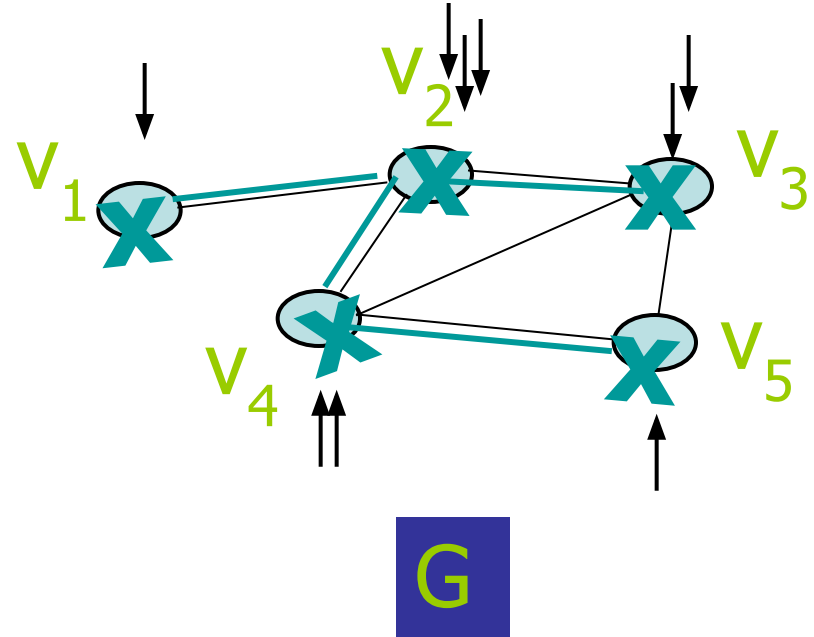
 dfsSpanningTree(u);

}

- Similar to DFS, the spanning tree edges can be generated based on BFS traversal.

Generating SP based on DFS

	stack
v_3	v_3
v_2	v_3, v_2
v_1	v_3, v_2, v_1
backtrack	v_3, v_2
v_4	v_3, v_2, v_4
v_5	v_3, v_2, v_4, v_5
backtrack	v_3, v_2, v_4
backtrack	v_3, v_2
backtrack	v_3
backtrack	empty

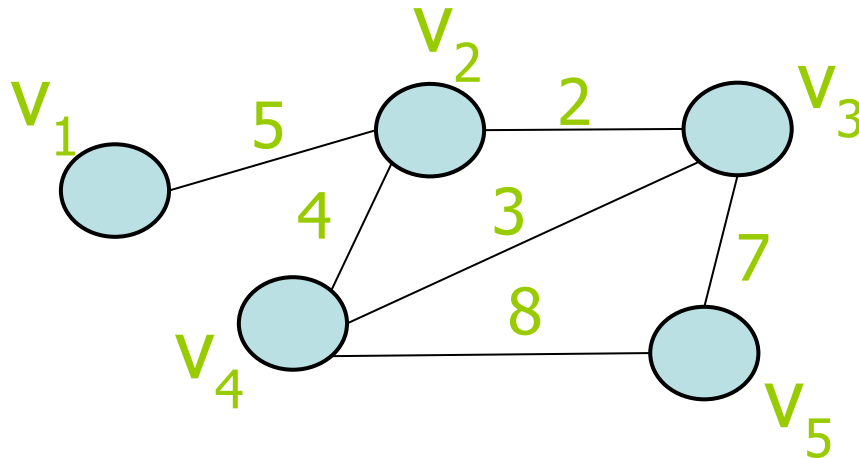


Minimum Spanning Tree (MST)

- Consider a connected undirected graph where
 - Each node x represents a country x
 - Each edge (x, y) has a number which measures the cost of placing telephone line between country x and country y
- **Problem:** connecting all countries while minimizing the total cost
- **Solution:** find a spanning tree with minimum total weight, that is, **minimum spanning tree**

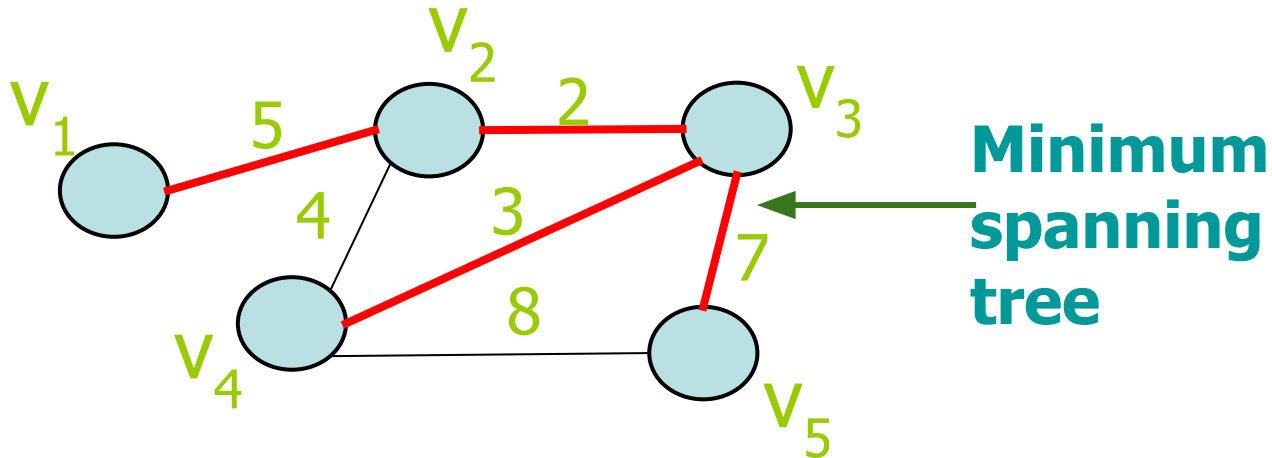
Formal definition of MST

- Given a connected undirected graph G .
- Let T be a spanning tree of G .
- $\text{cost}(T) = \sum_{e \in T} \text{weight}(e)$
- MST is a spanning tree T which minimizes $\text{cost}(T)$



Formal definition of MST

- Given a connected undirected graph G .
- Let T be a spanning tree of G .
- $\text{cost}(T) = \sum_{e \in T} \text{weight}(e)$
- MST is a spanning tree T which minimizes $\text{cost}(T)$

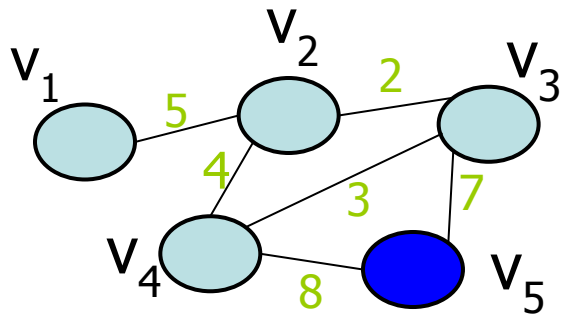


Prim's algorithm

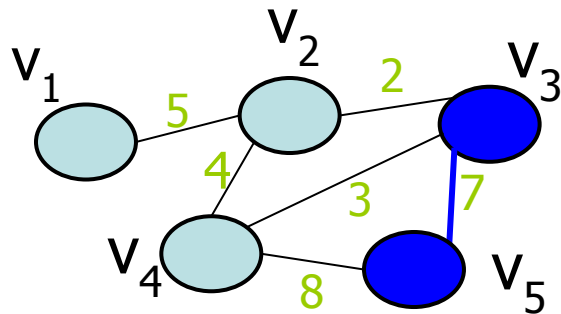
Algorithm PrimAlgorithm(**v**)

- Mark node **v** as visited and include it in the minimum spanning tree;
- while (there are unvisited nodes)
 - {
 - find the minimum edge (**v**, **u**) between a visited node **v** and an unvisited node **u**;
 - mark **u** as visited;
 - add both **v** and (**v**, **u**) to the minimum spanning tree;
 - }

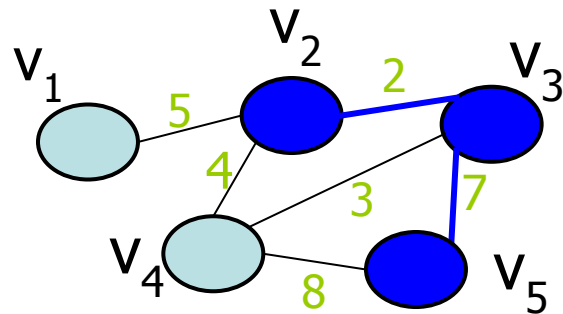
Prim's algorithm



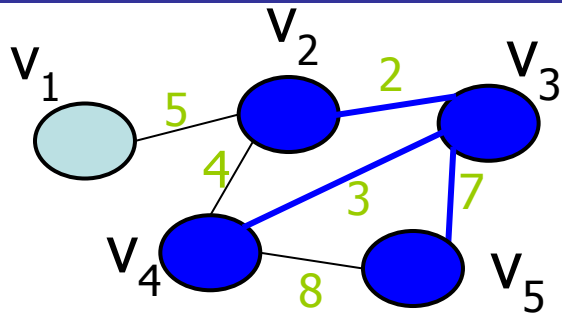
Start from v_5 , find the minimum edge attach to v_5



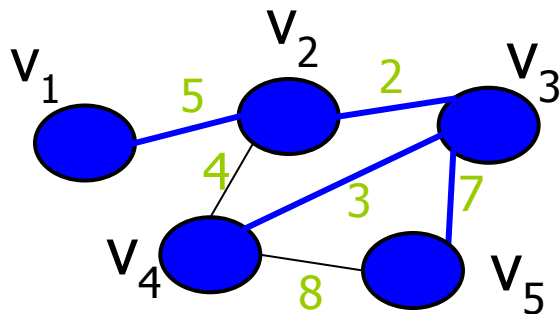
Find the minimum edge attach to v_3 and v_5



Find the minimum edge attach to v_2 , v_3 and v_5



Find the minimum edge attach to v_2 , v_3 , v_4 and v_5

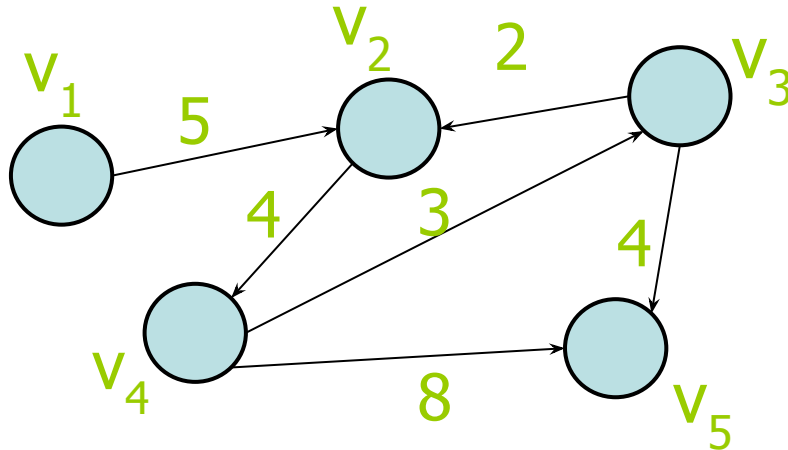


Shortest path

- Consider a weighted directed graph
 - Each node x represents a city x
 - Each edge (x, y) has a number which represent the cost of traveling from city x to city y
- **Problem**: find the minimum cost to travel from city x to city y
- **Solution**: find the **shortest path** from x to y

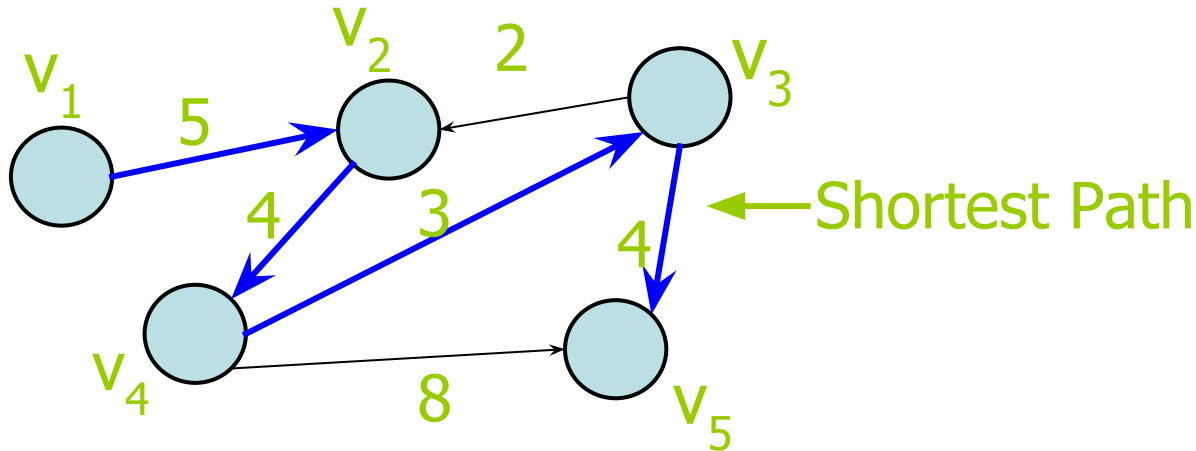
Formal definition of shortest path

- Given a weighted directed graph G .
- Let P be a path of G from x to y .
- $\text{cost}(P) = \sum_{e \in P} \text{weight}(e)$
- The shortest path is a path P which minimizes $\text{cost}(P)$



Formal definition of shortest path

- Given a weighted directed graph G .
- Let P be a path of G from x to y .
- $\text{cost}(P) = \sum_{e \in P} \text{weight}(e)$
- The shortest path is a path P which minimizes $\text{cost}(P)$



Dijkstra's algorithm

- Let **G** be a graph
- Each edge **(u, v)** has a weight **$w(u, v) > 0$**
- Find the shortest path starting from **v_1** to any node **v_i**
- Let **VS** be a subset of nodes in **G**
- Let **cost[v_i]** be the weight of the shortest path from **v_1** to **v_i** that passes through nodes in **VS** only

Dijkstra's algorithm

Algorithm shortestPath()

n = number of nodes in the graph;

for $i = 1$ to n

$\text{cost}[v_i] = w(v_1, v_i);$

$VS = \{ v_1 \};$

for step = 2 to n {

 find the smallest $\text{cost}[v_i]$ s.t. v_i is not in VS ;

 include v_i to VS ;

for (all nodes v_j not in VS) {

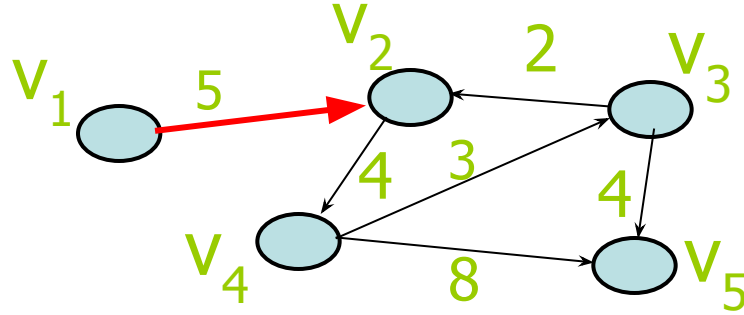
 if ($\text{cost}[v_j] > \text{cost}[v_i] + w(v_i, v_j)$)

$\text{cost}[v_j] = \text{cost}[v_i] + w(v_i, v_j);$

 }

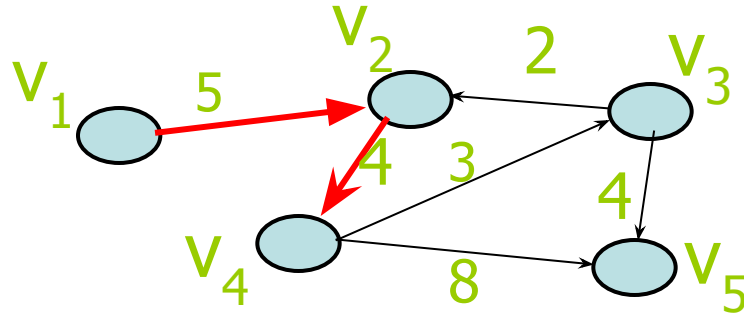
}

Dijkstra's algorithm



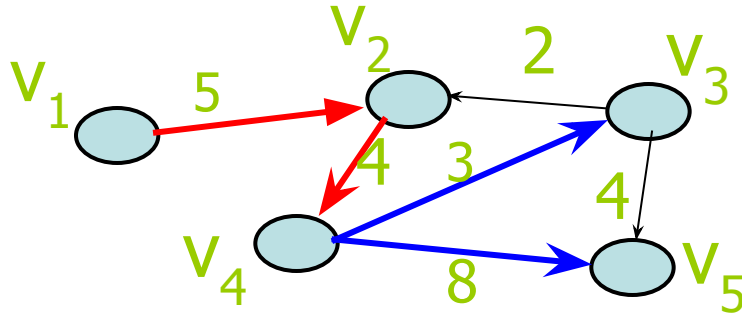
	v	VS	cost[v ₁]	cost[v ₂]	cost[v ₃]	cost[v ₄]	cost[v ₅]
1		[v ₁]	0	5	∞	∞	∞

Dijkstra's algorithm



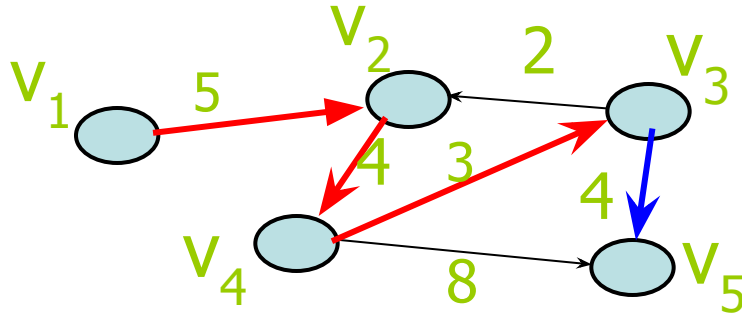
	v	VS	cost[v ₁]	cost[v ₂]	cost[v ₃]	cost[v ₄]	cost[v ₅]
1		[v ₁]	0	5	∞	∞	∞
2	v ₂	[v ₁ , v ₂]	0	5	∞	9	∞

Dijkstra's algorithm



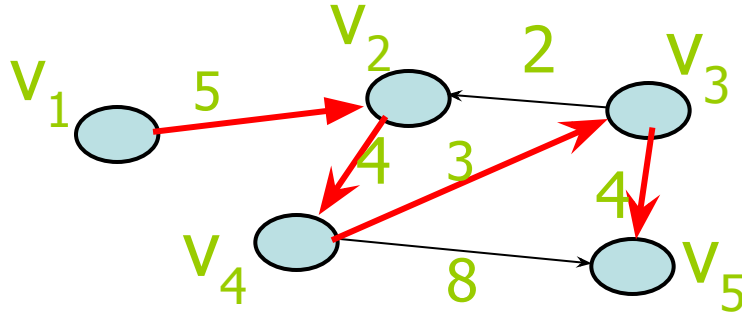
	v	VS	cost[v ₁]	cost[v ₂]	cost[v ₃]	cost[v ₄]	cost[v ₅]
1		[v ₁]	0	5	∞	∞	∞
2	v ₂	[v ₁ , v ₂]	0	5	∞	9	∞
3	v ₄	[v ₁ , v ₂ , v ₄]	0	5	12	9	17

Dijkstra's algorithm



	v	VS	cost[v ₁]	cost[v ₂]	cost[v ₃]	cost[v ₄]	cost[v ₅]
1		[v ₁]	0	5	∞	∞	∞
2	v ₂	[v ₁ , v ₂]	0	5	∞	9	∞
3	v ₄	[v ₁ , v ₂ , v ₄]	0	5	12	9	17
4	v ₃	[v ₁ , v ₂ , v ₄ , v ₃]	0	5	12	9	16

Dijkstra's algorithm



	v	VS	cost[v ₁]	cost[v ₂]	cost[v ₃]	cost[v ₄]	cost[v ₅]
1		[v ₁]	0	5	∞	∞	∞
2	v ₂	[v ₁ , v ₂]	0	5	∞	9	∞
3	v ₄	[v ₁ , v ₂ , v ₄]	0	5	12	9	17
4	v ₃	[v ₁ , v ₂ , v ₄ , v ₃]	0	5	12	9	16
5	v ₅	[v ₁ , v ₂ , v ₄ , v ₃ , v ₅]	0	5	12	9	16

Summary

- We have studied some basic concepts and algorithms
 - **Spanning Tree**
 - **Minimum Spanning Tree**
 - **Shortest Path**