## COURSE HANDOUT

| | |
|---|---|
| Course Code | ACSC13 |
| Course Name | Design and Analysis of Algorithms |
| Class / Semester | IV SEM |
| Section | A-SECTION |
| Name of the Department | CSE-CYBER SECURITY |
| Employee ID | IARE11023 |
| Employee Name | Dr K RAJENDRA PRASAD |
| Topic Covered | Single source shortest paths. |
| Course Outcome/s | Make Use of appropriate tree traversal techniques for findingshortest path |
| Handout Number | 30 |
| Date | 8 April, 2023 |

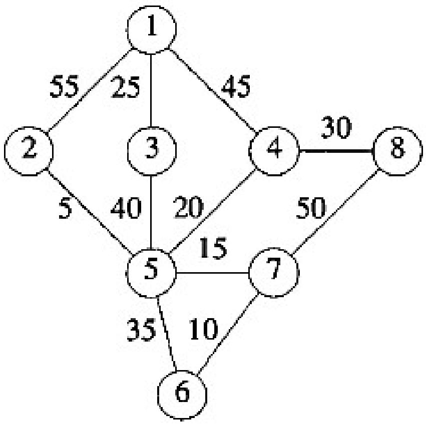
**Content about topic covered: Single source shortest path**

**Single source shortest path**

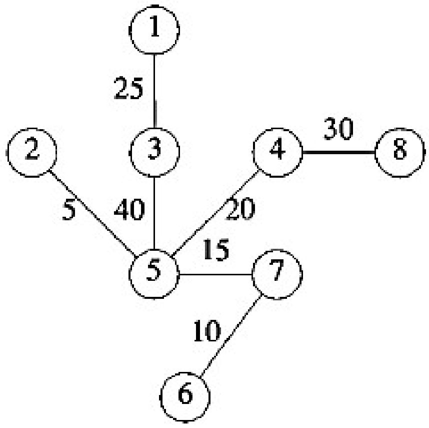
We are given a directed graph G = (V, E), a weighting function cost for the edges of G, and a source vertex $V_0$. The problem is to determine the shortest paths from $V_0$ to all the remaining vertices of G. It is assumed that all the weights are +ve.

```
Algorithm ShortestPaths(v, cost, dist, n)
// dist[j], 1 ≤ j ≤ n, is set to the length of the shortest
// path from vertex v to vertex j in a digraph G with n
// vertices. dist[v] is set to zero. G is represented by its
// cost adjacency matrix cost[1 : n, 1 : n].
{
    for i := 1 to n do
    { // Initialize S.
        S[i] := false; dist[i] := cost[v, i];
    }
    S[v] := true; dist[v] := 0.0; // Put v in S.
    for num := 2 to n − 1 do
    {
        // Determine n − 1 paths from v.
        Choose u from among those vertices not
        in S such that dist[u] is minimum;
        S[u] := true; // Put u in S.
        for (each w adjacent to u with S[w] = false) do
            // Update distances.
            if (dist[w] > dist[u] + cost[u, w])) then
                    dist[w] := dist[u] + cost[u, w];
    }
}
```

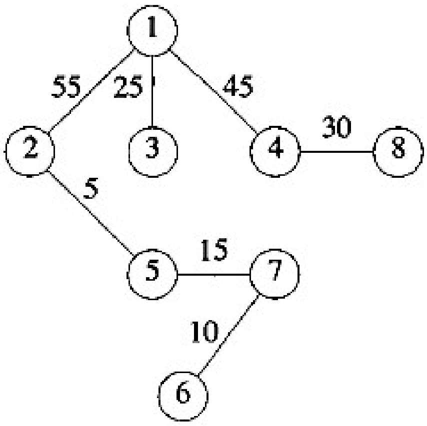
Eg: A graph, minimum cost spanning tree and shortest path spanning tree from vertex 1 are shown below:




(a) A Graph




(b) Minimum cost spanning tree




(c) Shortest path spanning tree from vertex 1.