Introduction to Graphs

Graphs

Introduction to Graphs

- We begin a major new topic: Graphs.
- Graphs are important discrete structures because they are a flexible mathematical model for many application problems.

Introduction to Graphs

Any time there is

- a set of objects and
- there is some sort of "connection" or "relationship" or "interaction" between pairs of objects,

a graph is a good way to model this.

Introduction to Graphs

Examples:

- computer and communication networks
- transportation networks, e.g., roads
- VLSI, logic circuits
- surface meshes for shape description in computer-aided desigr and GIS

Introduction to Graphs

- transportation networks, e.g., roads
- VLSI, logic circuits
- surface meshes for shape description in computer-aided desigr and GIS
- precedence constraints in scheduling systems.

Graphs and Digraphs

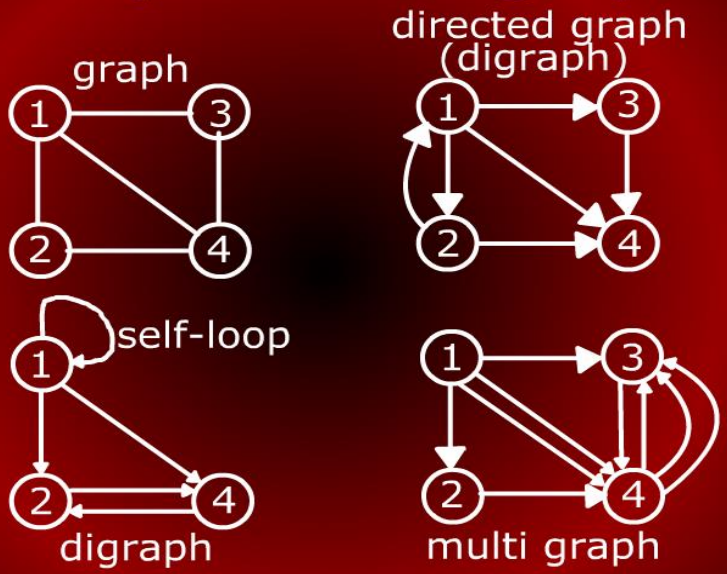
A graph G = (V, E) consists of

- a finite set of vertices V (or nodes) and
- E, a binary relation on V called edges.

Graphs and Digraphs

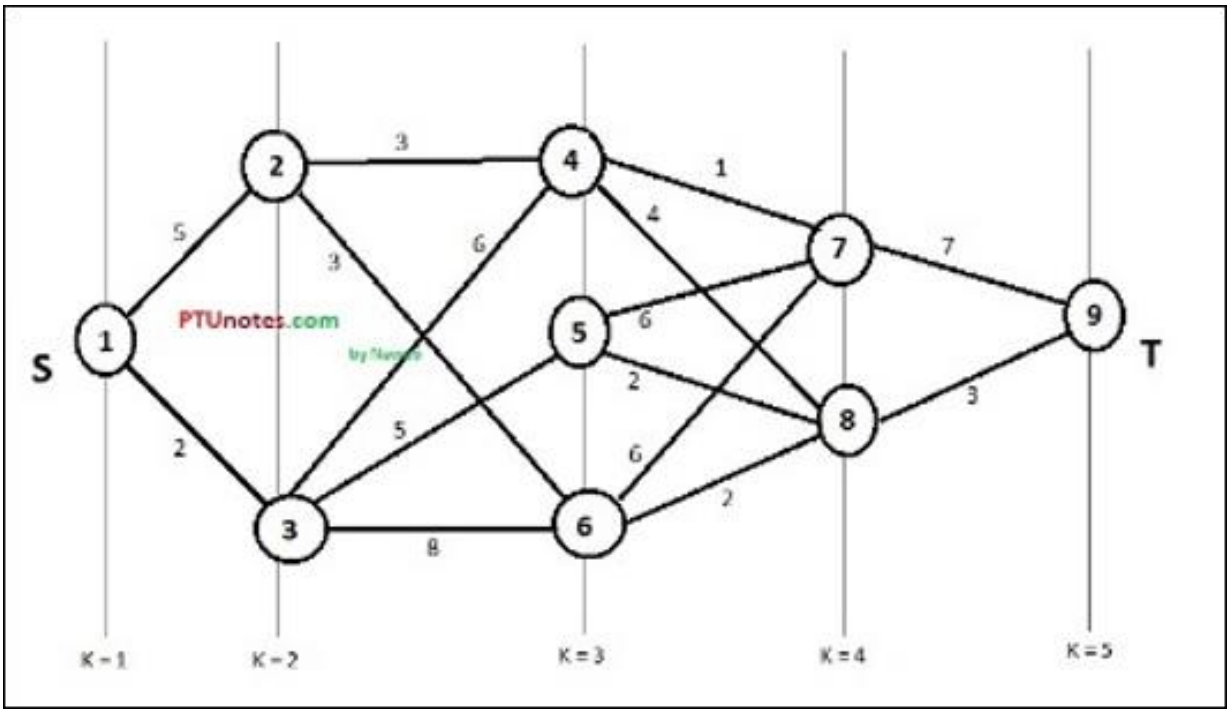
- E is a set of pairs from V.
- If a pair is *ordered*, we have a *directed* graph.
- For *unordered* pair, we have an *undirected* graph.

Graphs and Digraphs



Multistage Graph (Shortest Path) $O(|V| + |E|)$

The multistage graph problem is finding the path with minimum cost from source S to sink T.



We have the formula:

$$\text{cost(Stage,Vertex)=min\{c(Vertex,l)+cost(Stage+1,l)\}}$$

According to the formula, we have to calculate the cost (stage, vertex) using the following steps, the small c denote the cost of the edge, and l denote the destination stage of the current vertex.

1. Create a table

Nodes									
Cost									
Destination									

In Figure K = Stage, can be denoted by any alphabet; here I am using K

Where S is the starting point T is the ending point and nodes are Vertex

The multistage graph problem can be solved in two ways using dynamic programming

- Forward approach
- Backward approach

We use the Backward Approach and fill the above table

As we move from backward we start from K=5

- **Stage-5**

$Cost(5,9) = 0$

The cost of vertex 9 is 0 as it is terminal and if you start from one node and end on the same node then its cost would be zero and the destination would be the same as started.

Nodes									9
Cost									0
Destination									9

- **Stage = 4**

In stage 4 we have two nodes so we calculate their costs individually one for vertex 7 and one for vertex 8

$Cost(4,7) = 7$

$Cost(4,8) = 3$

If you move from node 7 to node 9, the cost is written on the transition arrow between two nodes)

Nodes							7	8	9
Cost							7	3	0
Destination							9	9	9

- **Stage = 3**

If you may observe that on stage 3 there are three nodes and each node have more than one transition connecting with node 7 and 8, now remember that we only calculate the cost between stage-3 nodes and stage-4 nodes, the rest of the nodes (7, 8, and 9) we already calculated and filled the above table accordingly, so we don't need to re-calculate them.

Secondly, if a node has more than one outgoing transition then we calculate their cost individually and pick the minimum cost among them to fill the above table, respectively.

Now apply the formula:

$Cost(3,4) =$	min	$c(4,7) + cost(4,7)$ $1 + 7 = 8$	
		$c(4,8) + cost(4,8)$ $4 + 3 = 7$	✓

$Cost(3,5) =$	min	$c(5,7) + cost(4,7)$ $6 + 7 = 13$	
		$c(5,8) + cost(4,8)$ $2 + 3 = 5$	✓

$Cost(3,6) =$	min	$c(6,7) + cost(4,7)$ $6 + 7 = 13$	
		$c(6,8) + cost(4,8)$ $2 + 3 = 5$	✓

Nodes				4	5	6	7	8	9
Cost				7	5	5	7	3	0
Destination				8	8	8	9	9	9

- **Stage = 2**

$Cost(2,2) =$	min	$c(2,4) + cost(3,4)$ $3 + 7 = 10$	
		$c(2,6) + cost(3,6)$ $3 + 5 = 8$	✓

$Cost(2,3) =$	min	$c(3,4) + cost(3,4)$ $6 + 7 = 13$	
		$c(3,5) + cost(3,5)$ $5 + 5 = 10$	✓
		$c(3,6) + cost(3,6)$ $8 + 5 = 13$	

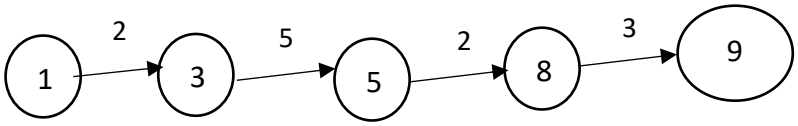
Nodes		2	3	4	5	6	7	8	9
Cost		8	10	7	5	5	7	3	0
Destination		6	5	8	8	8	9	9	9

- **Stage = 1**

$Cost(1,1) =$	min	$c(1,2) + cost(2,2)$ $5 + 8 = 13$	
		$c(1,3) + cost(2,3)$ $2 + 10 = 12$	✓

Nodes	1	2	3	4	5	6	7	8	9
Cost	12	8	10	7	5	5	7	3	0
Destination	3	6	5	8	8	8	9	9	9

Shortest Path:



// Graph stored in the form of an adjacency Matrix

e.g.

$nodes \rightarrow$ $\downarrow$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	INF	5	2	INF	INF	INF	INF	INF	INF
2	INF	INF	INF	3	INF	3	INF	INF	INF
3	INF	INF	INF	6	5	8	INF	INF	INF
4	INF	INF	INF	INF	INF	INF	1	4	INF
5	INF	INF	INF	INF	INF	INF	6	2	INF
6	INF	INF	INF	INF	INF	INF	6	2	INF
7	INF	INF	INF	INF	INF	INF	INF	INF	7
8	INF	INF	INF	INF	INF	INF	INF	INF	3
9	INF	INF	INF	INF	INF	INF	INF	INF	INF

```
int[][] graph = new int[][]{
        {INF,  5,   2,   INF, INF, INF, INF, INF, INF},
        {INF,  INF, INF, 3,   INF, 3,   INF, INF, INF},
        {INF,  INF, INF, 6,   5,   8,   INF, INF, INF},
        {INF,  INF, INF, INF, INF, INF, 1,   4,   INF},
        {INF,  INF, INF, INF, INF, INF, 6,   2,   INF},
        {INF,  INF, INF, INF, INF, INF, 6,   2,   INF},
        {INF,  INF, INF, INF, INF, INF, INF, INF, 7},
        {INF,  INF, INF, INF, INF, INF, INF, INF, 3}};
```

1. $N \leftarrow 9$ // number of nodes
2. $INF \leftarrow \infty$

$Shortest\_Distance\ (int\ [\ ][\ ]\ graph)$

1. $int[\ ]\ dist \leftarrow N$ //is going to shortest distance from node 1 to node N-1
2. $dist[N-1] \leftarrow 0;$
3. $for\ i \leftarrow N-2\ downto\ 1$
4. $dist[i] \leftarrow INF;$
5. $for\ j \leftarrow i\ to\ N-1$
6. $if\ graph[i][j] = INF\ then$
7. $continue;$ //immediate jump for next iteration
8. $end\ if$
9. $dist[i] = Math.\min(dist[i], graph[i][j] + dist[j])$ //we apply recursive equation to distance to target through j and compare with minimum distance so far
10. $end\ for$
11. $end\ for$
12. $return\ dist[0];$

Spanning Tree (Greedy Approach)

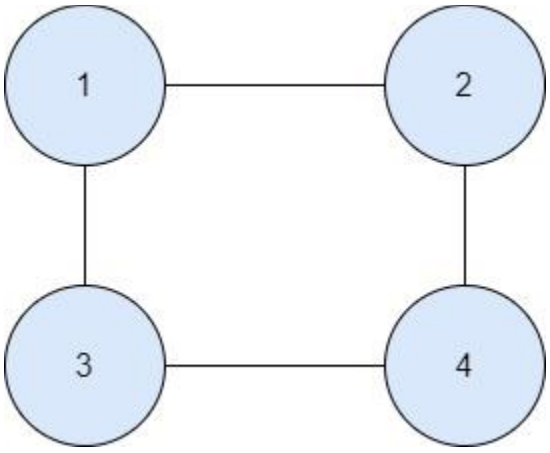
A connected sub-graph 'S' of Graph G(V,E) is said to be spanning if and if only (iff):

1. 'S' should contain all vertices of 'G'

2. 'S' should contain $(|V| - 1)$ edge's

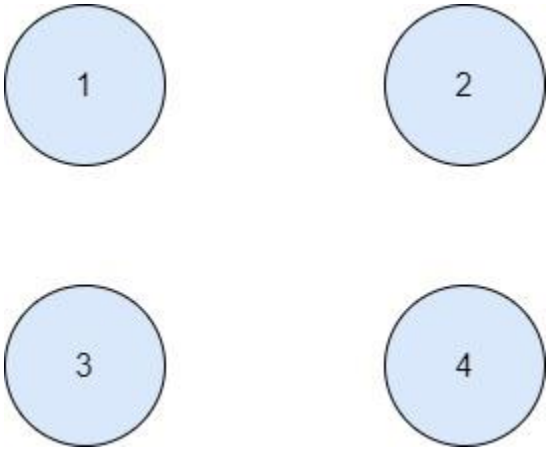
3. As it is tree so there is no cycle between vertices

For Example: The below Graph ‘G’

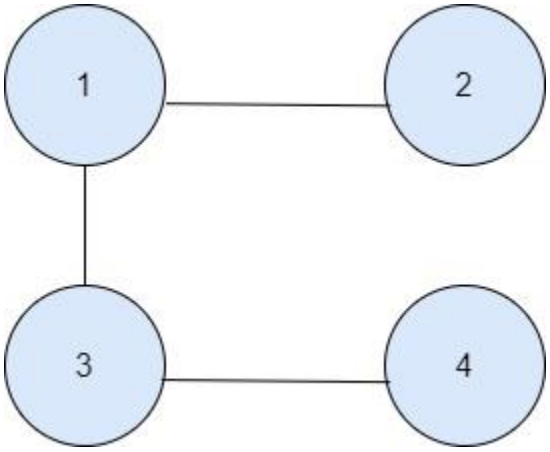


Now create the spanning tree

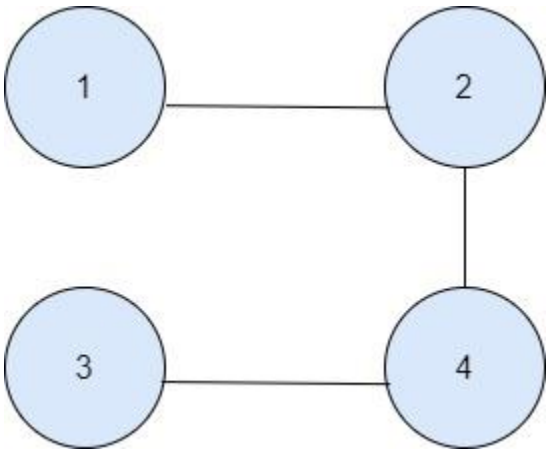
1. Should contain all vertex



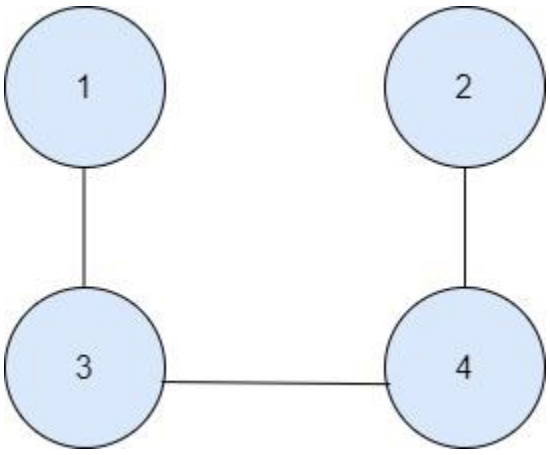
2. Should contain edges $(|V| - 1)$



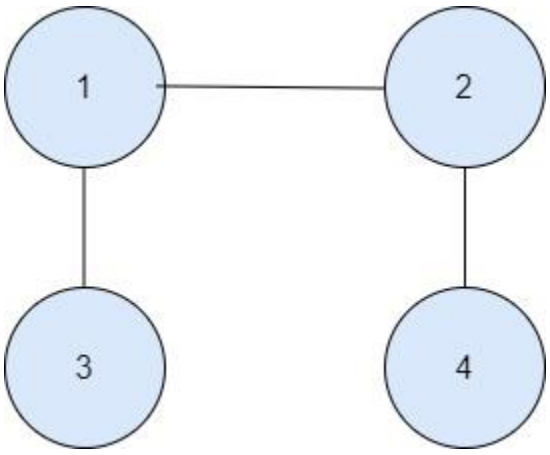
Or



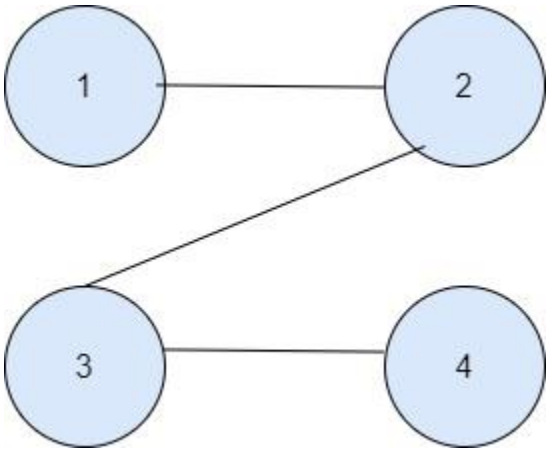
OR



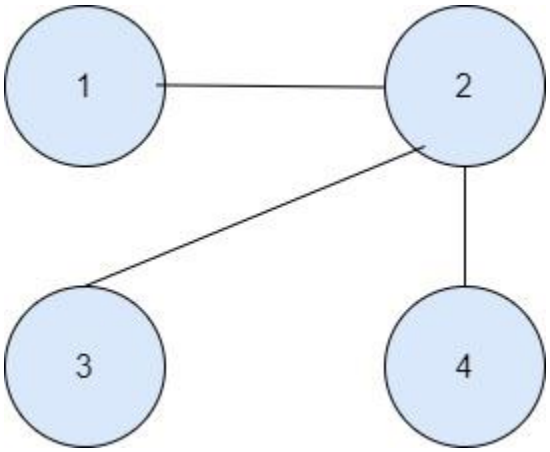
OR



OR



OR

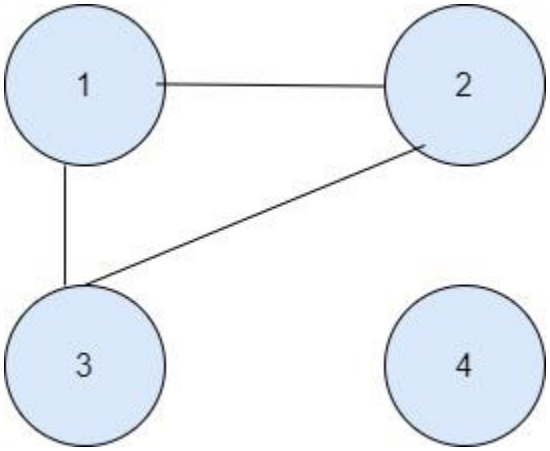


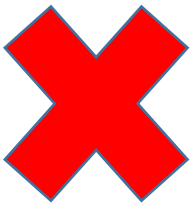
Wrong Spanning Tree would be:

1. Containing isolated vertices

2. Containing cycle between vertices

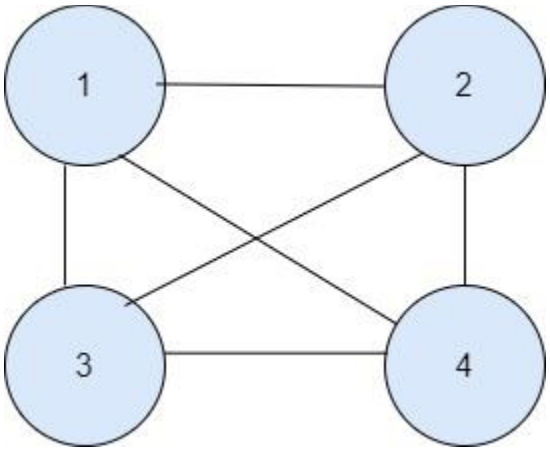
e.g





Possible questions may occur:

1. Find out the number of spanning tree can be possible of the given graph below:



You may apply formula to calculate the possible spanning tree if and only if the given graph is complete graph as given above.

$$\boldsymbol{n^{n-2}}$$

Where n is the number of vertices in the graph.

If the given Graph is not a complete graph then we use Kirchhoff's theorem.

2. Find the minimum cost spanning tree

This is to find the shortest path in the spanning tree using Krushal Algorithm and Prim's Algorithm.