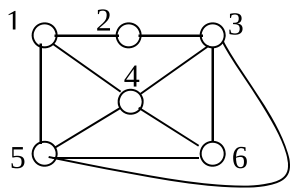
## Tutorial 11

1. **Find in the following array the index of the element of 22 by applying the binary search algorithm. How many times is the binary search algorithm faster than the linear search in this case?**

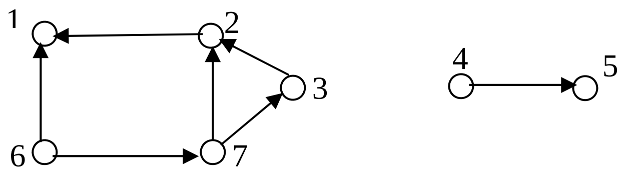
| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **2** | **4** | **5** | **7** | **8** | **9** | **12** | **14** | **17** | **19** | **22** | **25** | **27** | **28** | **33** | **37** |

2. **Write an algorithm for the recursive implementation of binary search.**

3. **Draw adjacency lists for the graph**




4. **Draw adjacency lists for the digraph**




5. **Write an algorithm that prints the indegree and outdegree of every vertex in a digraph, where the digraph is represented using adjacency lists.**

## Questions of Tutorial 12

1. **List the order in which the vertices of the following graph are visited when the depth-first-search algorithm is executed with the start vertex = 4. Assume that the vertices are listed in increasing order in each adjacency list.**

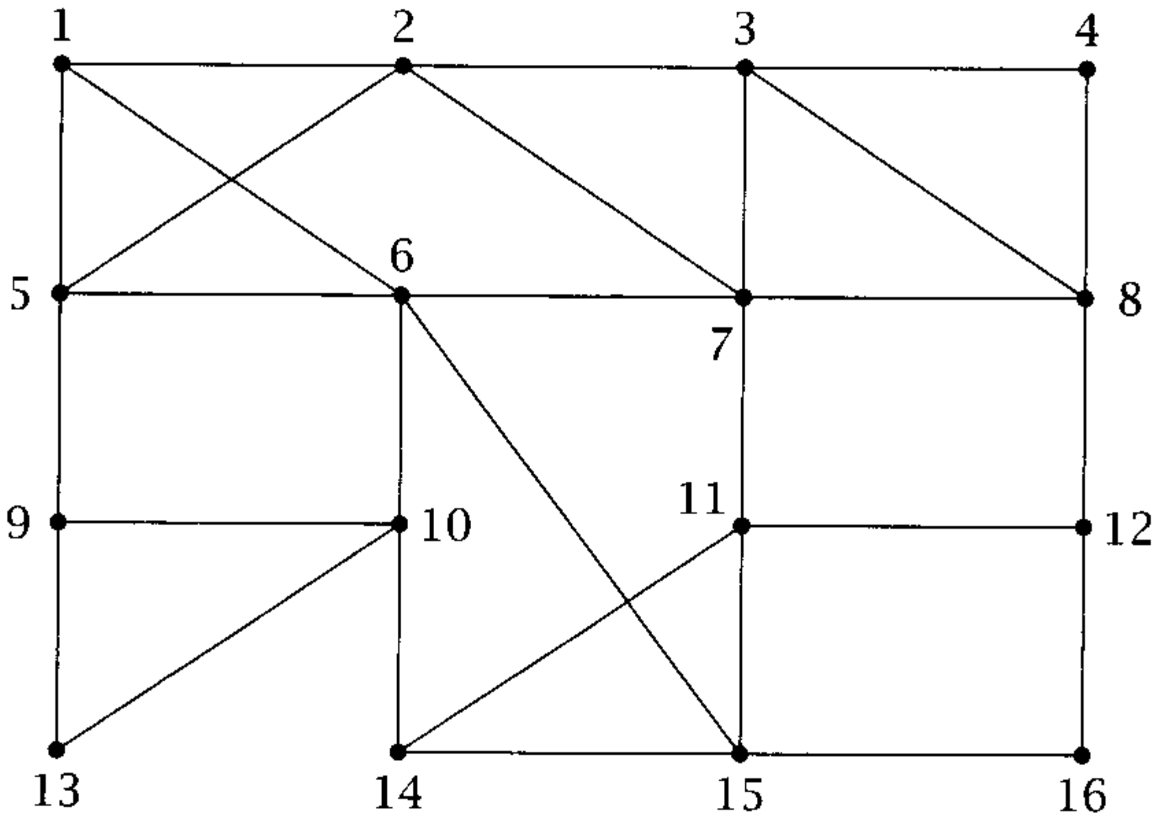



**Figure 1**

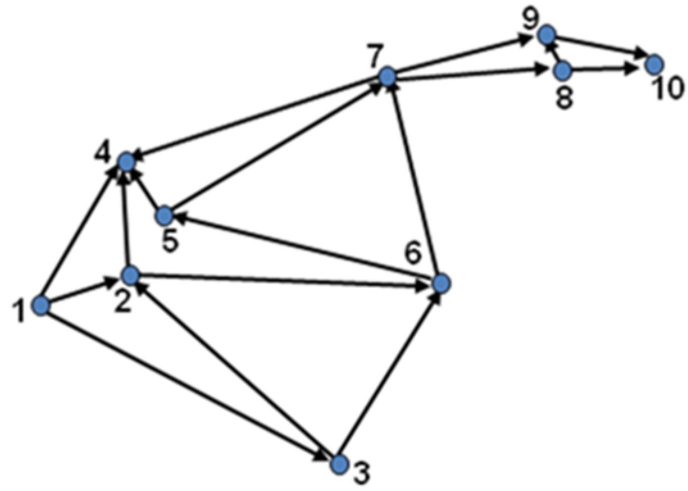
2. **List the order in which the vertices of the graph shown in Figure 1 are visited when the breadth-first-search algorithm is executed with the start vertex = 4. Assume that the vertices are listed in increasing order in each adjacency list.**

3. **Let *T* and *T'* are two spanning trees of a connected Graph *G*. Suppose that an edge *e* is in *T* but not in *T'*. Show that there is an edge *e'* is in *T'* but not in *T* so that $(T-\{e\})\cup\{e'\}$ and $(T'-\{e'\})\cup\{e\}$ are spanning trees of *G*.**

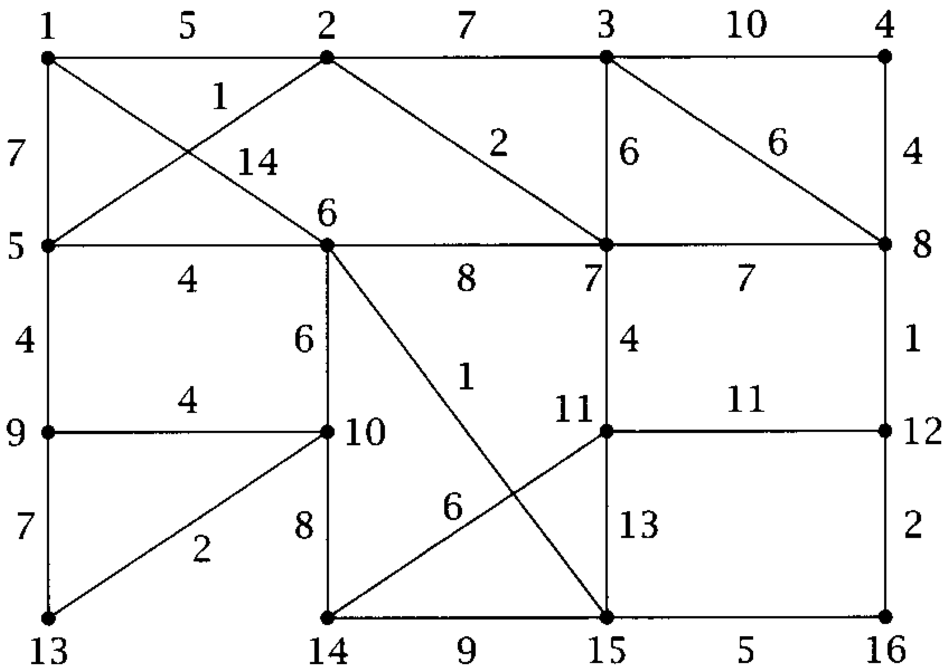
4. **Write a stack based non-recursive algorithm implementation of the depth-first-search approach.**

**Questions of Tutorial 13**

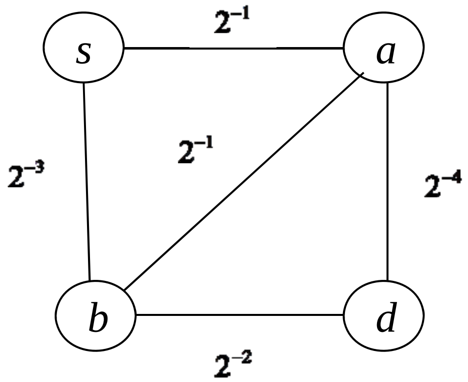
1. **Show one topological sort of the following DAG.**




2. **Find the minimum spanning tree of the following graph by using *Kruskal*'s algorithm.**



3. **Consider a weighted graph $G$. If $(u,v)$ is an edge in the graph, let $w(u,v)$ denote its weight. Suppose that all edge weights are between 0 and 1. Given a starting vertex $s$ and a destination vertex $d$, we wish to find a path $(s,u_1,u_2,\ldots,u_m,d)$ with *maximum edge weight product* $w(s,u_1)\times w(u_1,u_2)\times\cdots\times w(u_m,d)$. Suggest an efficient algorithm to do this and trace its steps for the following graph.**




4. **Let $G$ be a connected weighted graph and $v$ be a vertex in $G$. Suppose that the weights of the edges incident on $v$ are distinct. Let $e$ be the edge of minimum weight incident on $v$. Show that $e$ is contained in every minimal spanning tree.**

5. **Suppose that $G=(V,E)$ is a tree represented by an adjacency list. Write in pseudo-code an algorithm that constructs the adjacency list for a new graph $G'=(V,E')$ with the same set of vertices $V$ as $G$, and with edges between any two vertices if and only if they are 2 hops away in $G$, i.e., $G'$ contains the edge $(u,v)$ if and only there is a path of length 2 in $G$ connecting $u$ and $v$.**