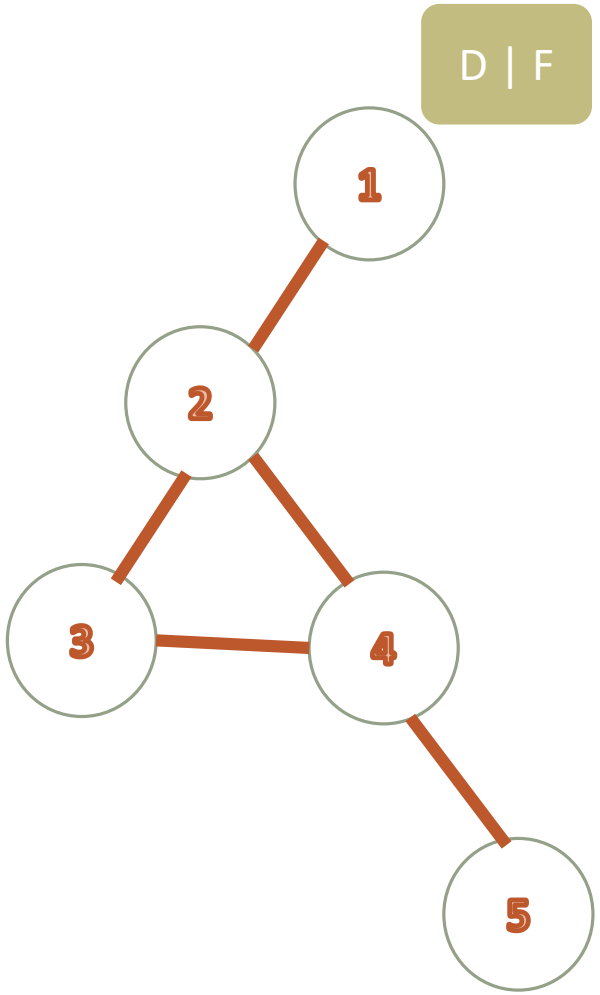
Graph Traversal

DEPTH FIRST SEARCH

Depth-First Search: The Code

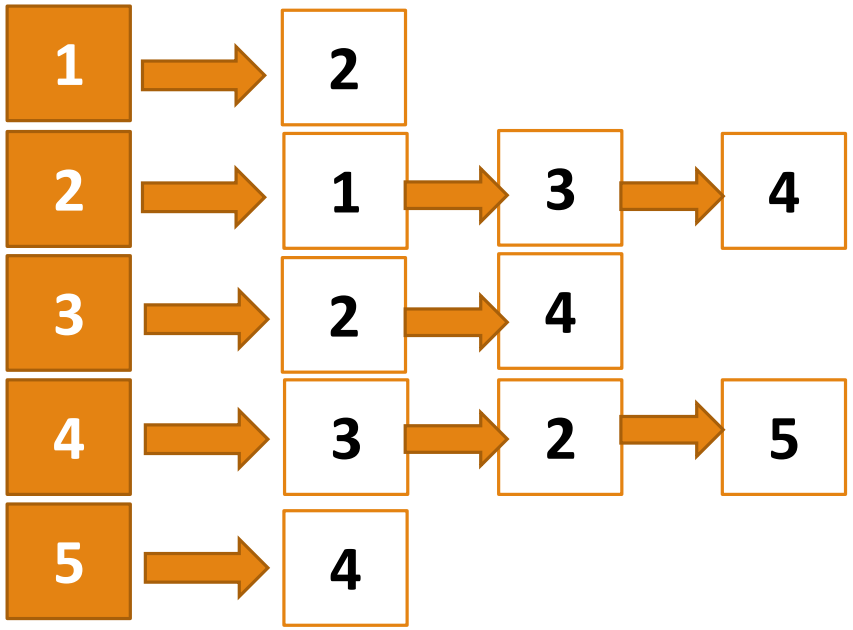
```
DFS(G)
{
   for each vertex u ∈ G->V
   {
      u->color = WHITE;
   }
   time = 0;
   for each vertex u ∈ G->V
   {
      if (u->color == WHITE)
         DFS_Visit(u);
   }
}
```

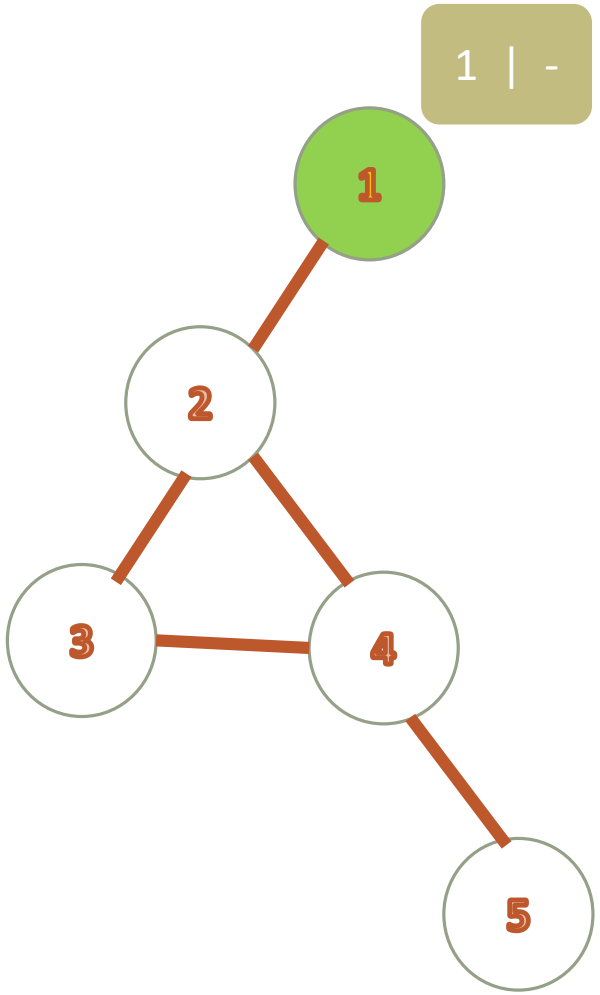
```
DFS_Visit(u)
{
   u->color = GREY;
   time = time+1;
   u->d = time;
   for each v ∈ u->Adj[]
   {
      if (v->color == WHITE)
         DFS_Visit(v);
   }
   u->color = BLACK;
   time = time+1;
   u->f = time;
}
```



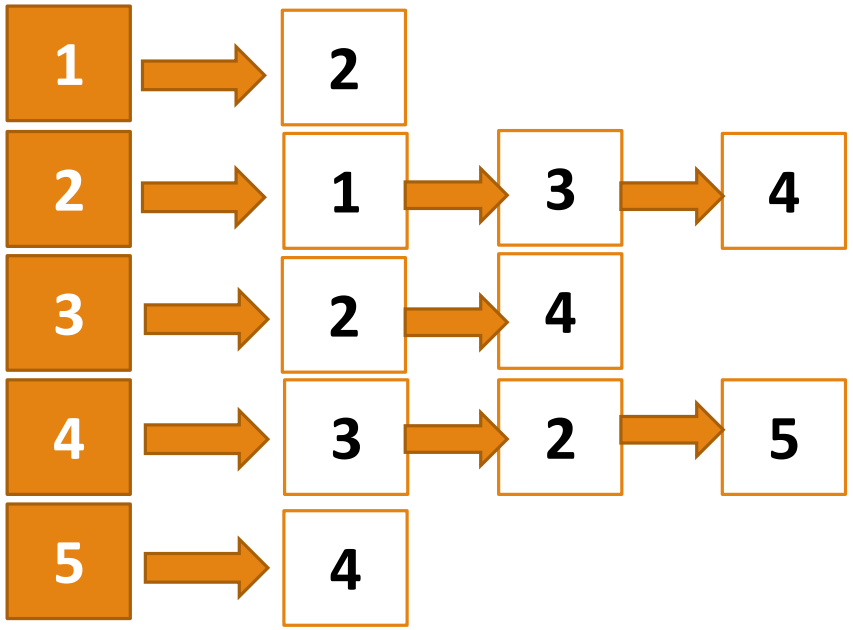
	1	2	3	4	5
Visited	0	0	0	0	0

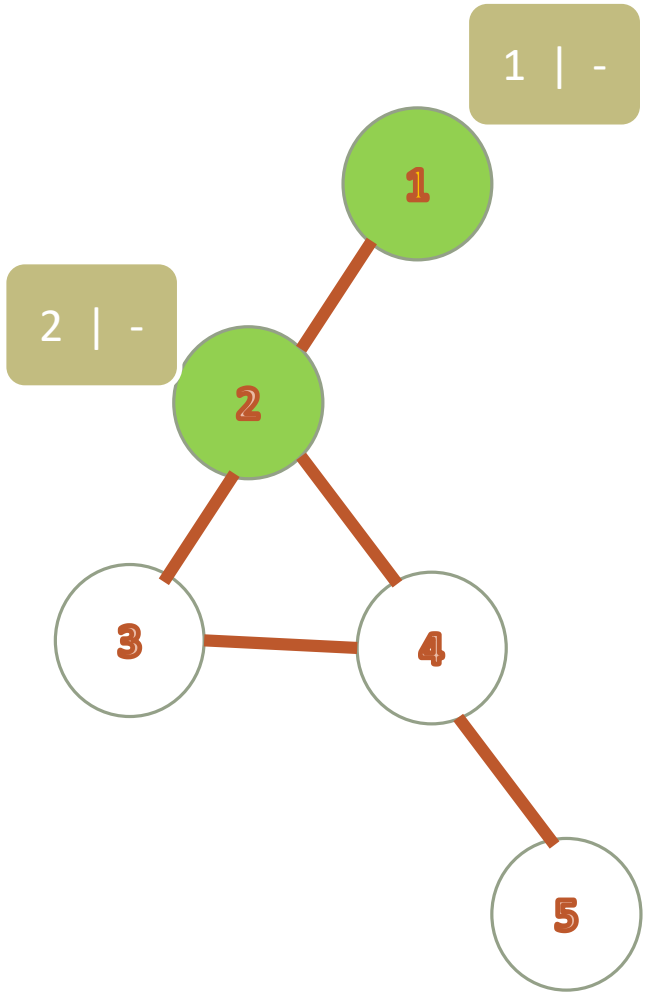
D = Discover Time
F = Finishing Time



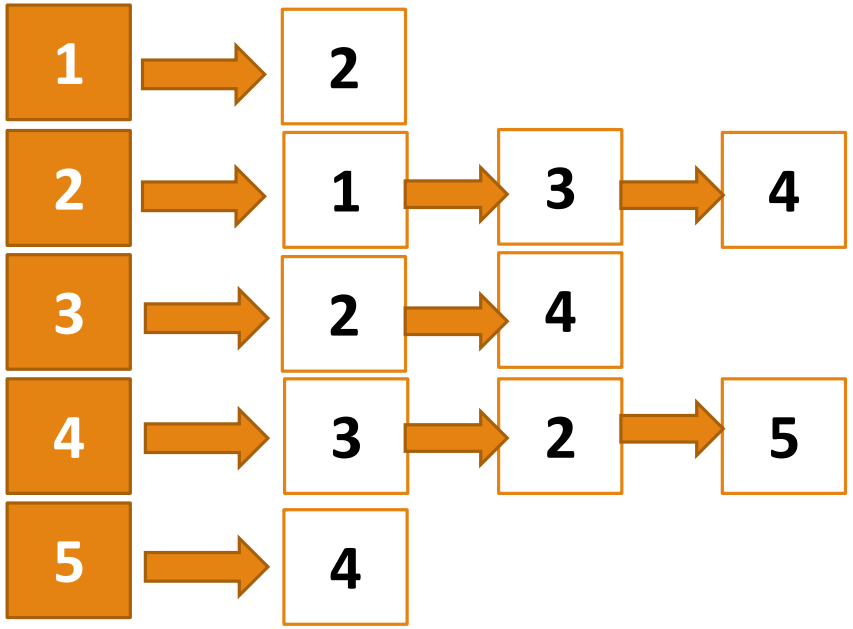


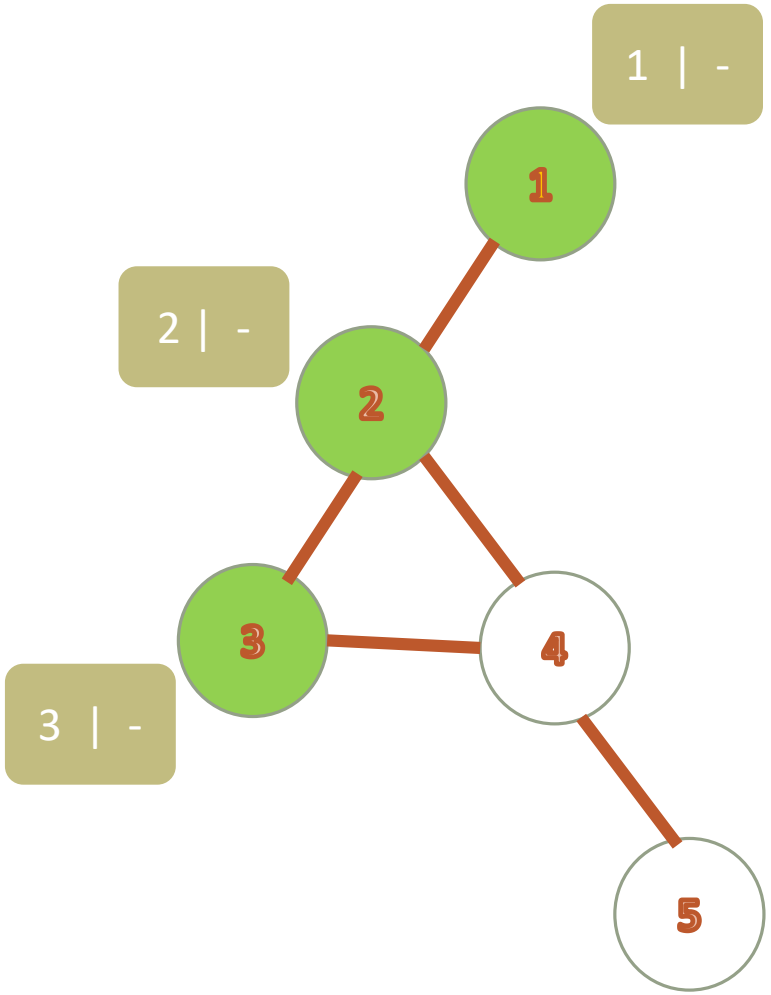
	1	2	3	4	5
Visited	1	0	0	0	0



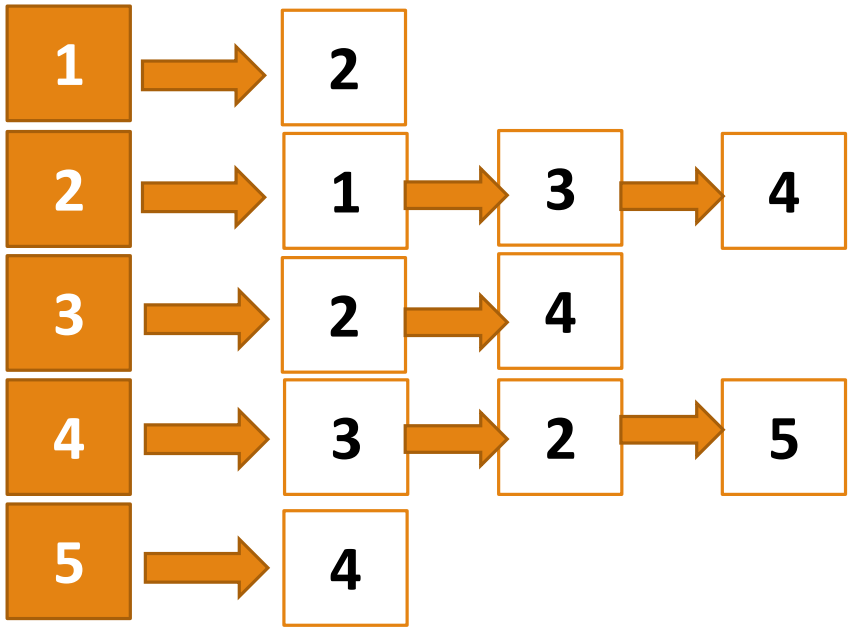


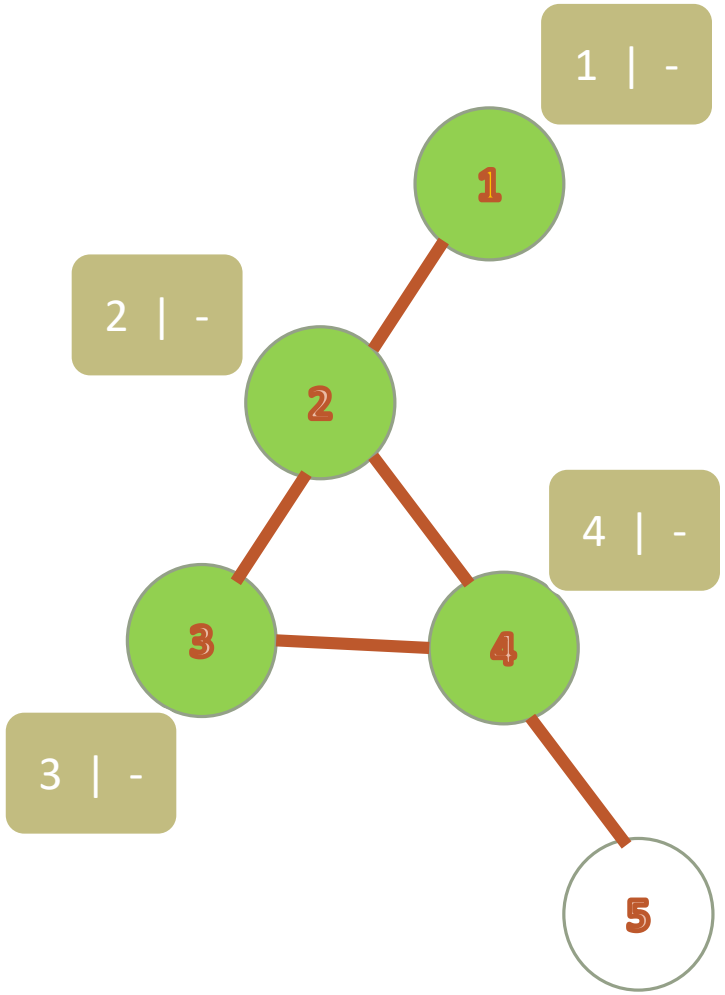
	1	2	3	4	5
Visited	1	1	0	0	0



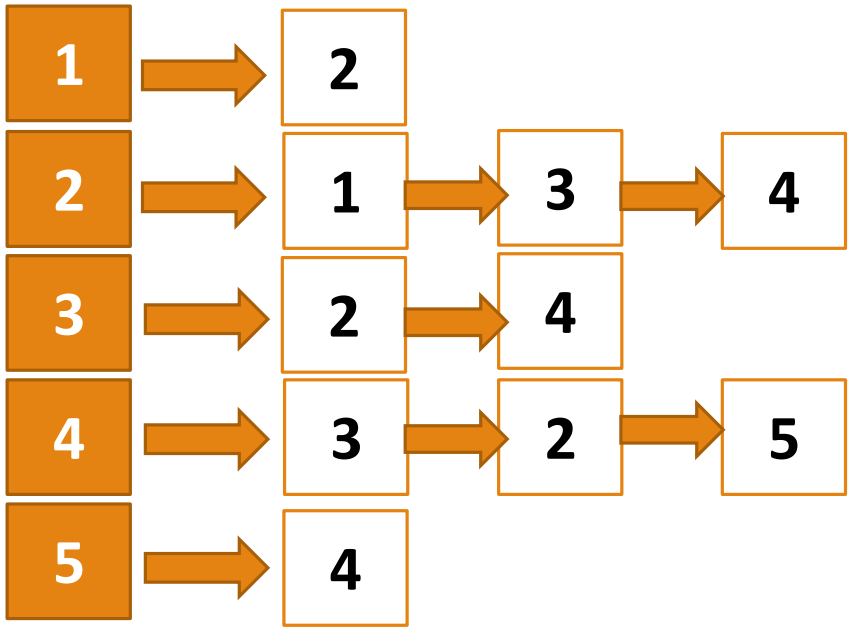


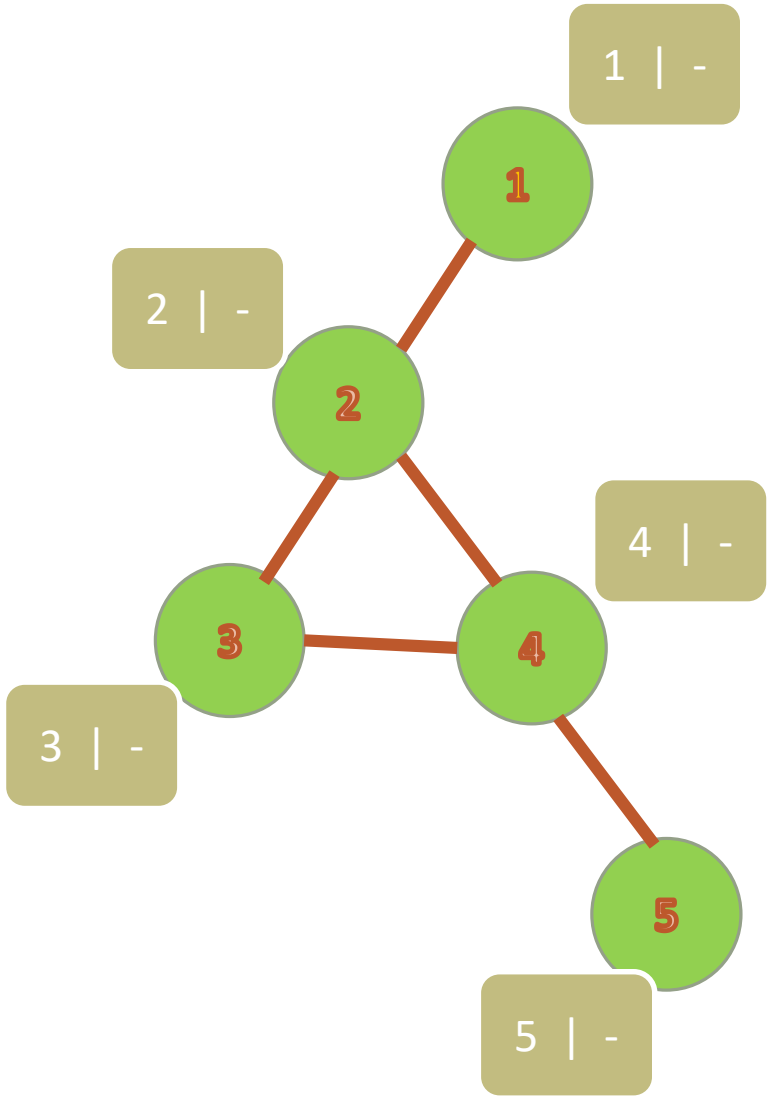
	1	2	3	4	5
Visited	1	1	1	0	0



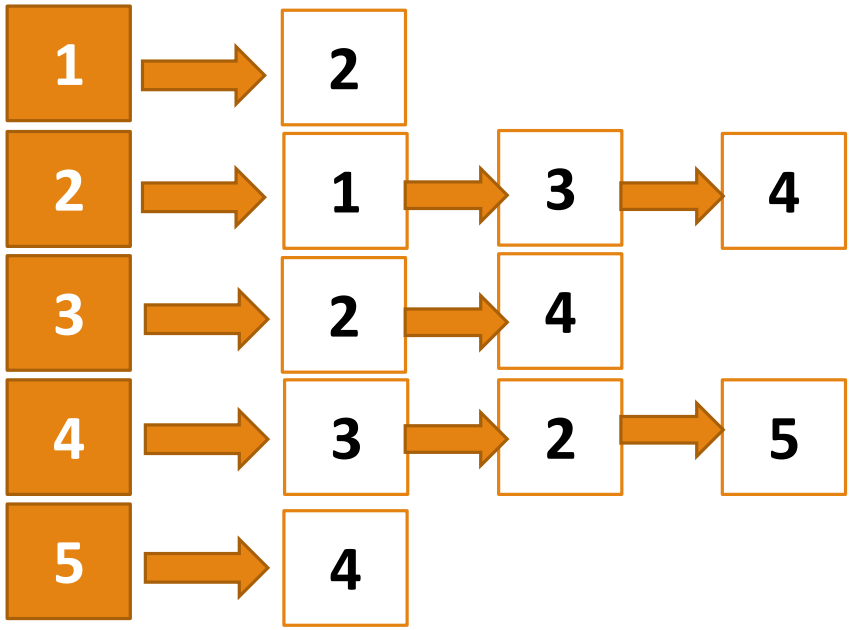


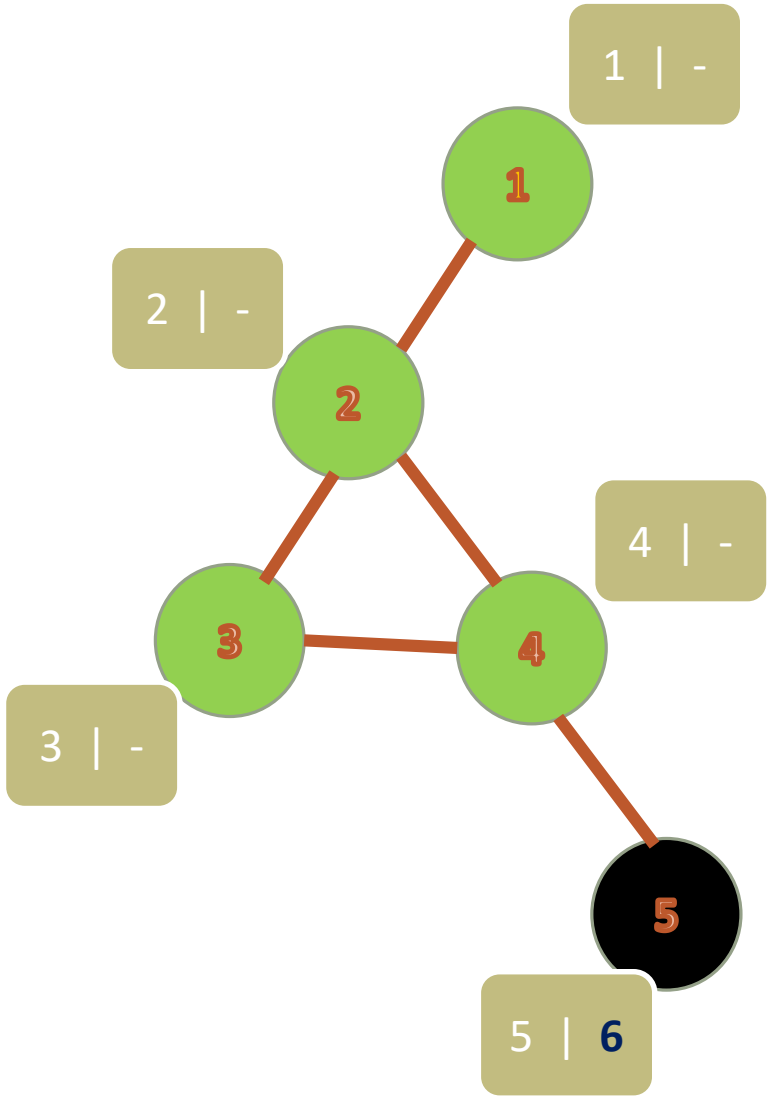
	1	2	3	4	5
Visited	1	1	1	1	0



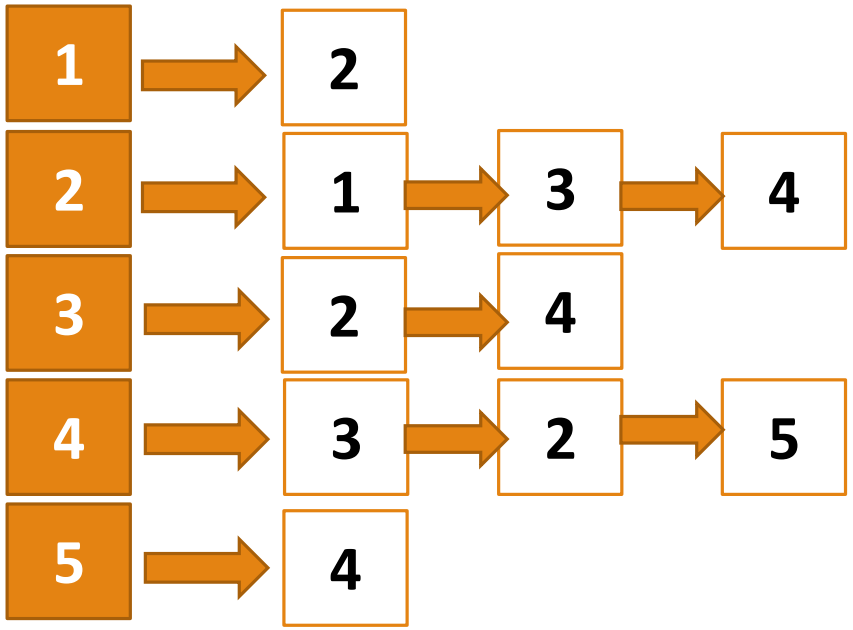


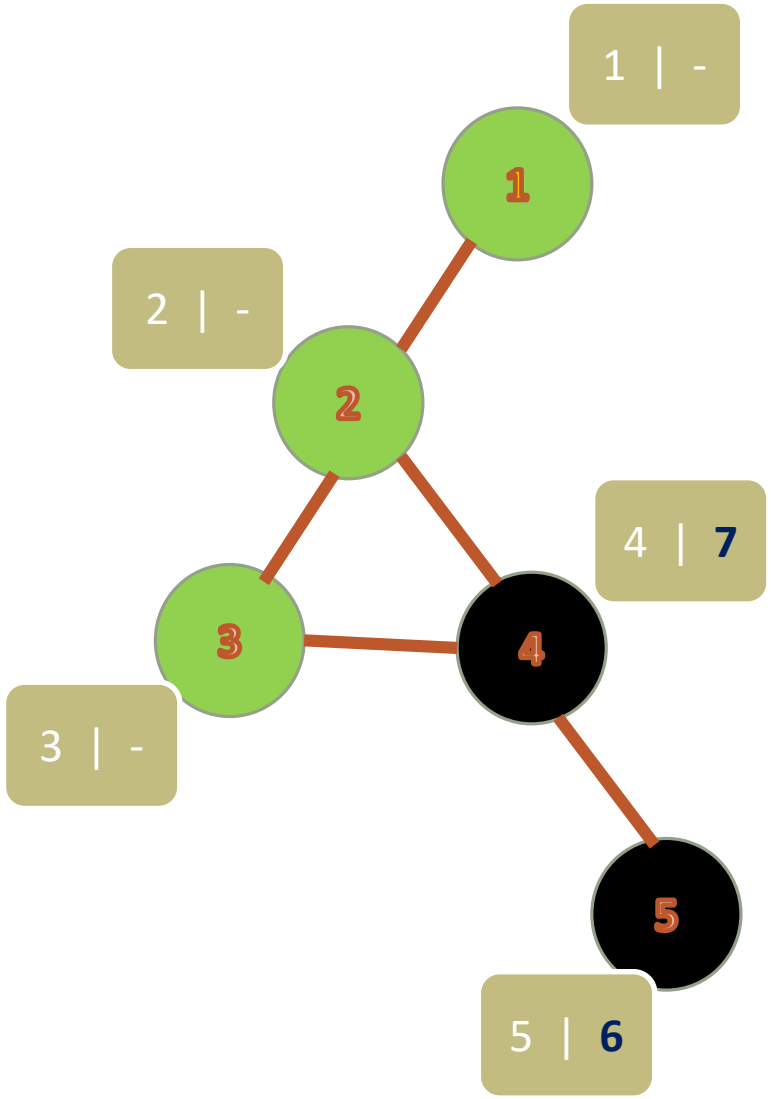
	1	2	3	4	5
Visited	1	1	1	1	1



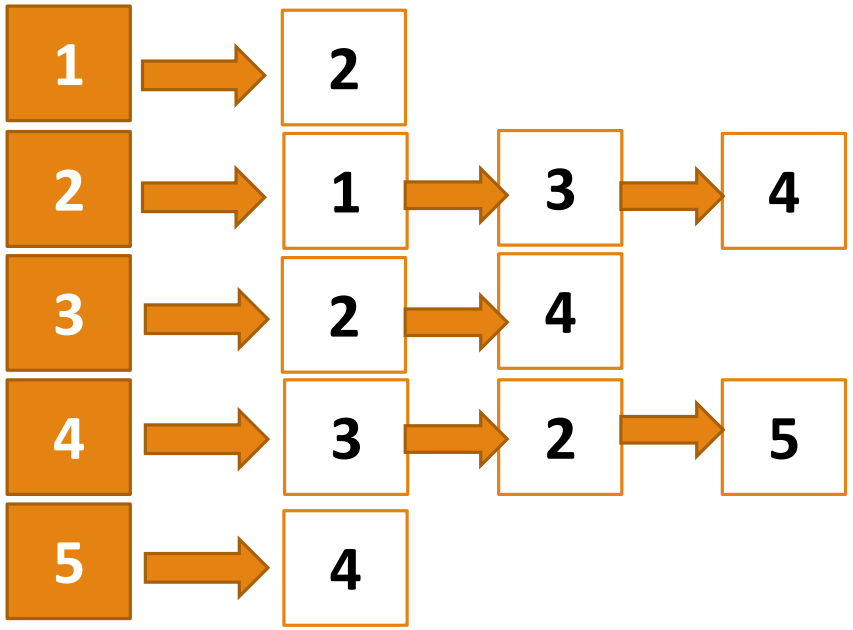


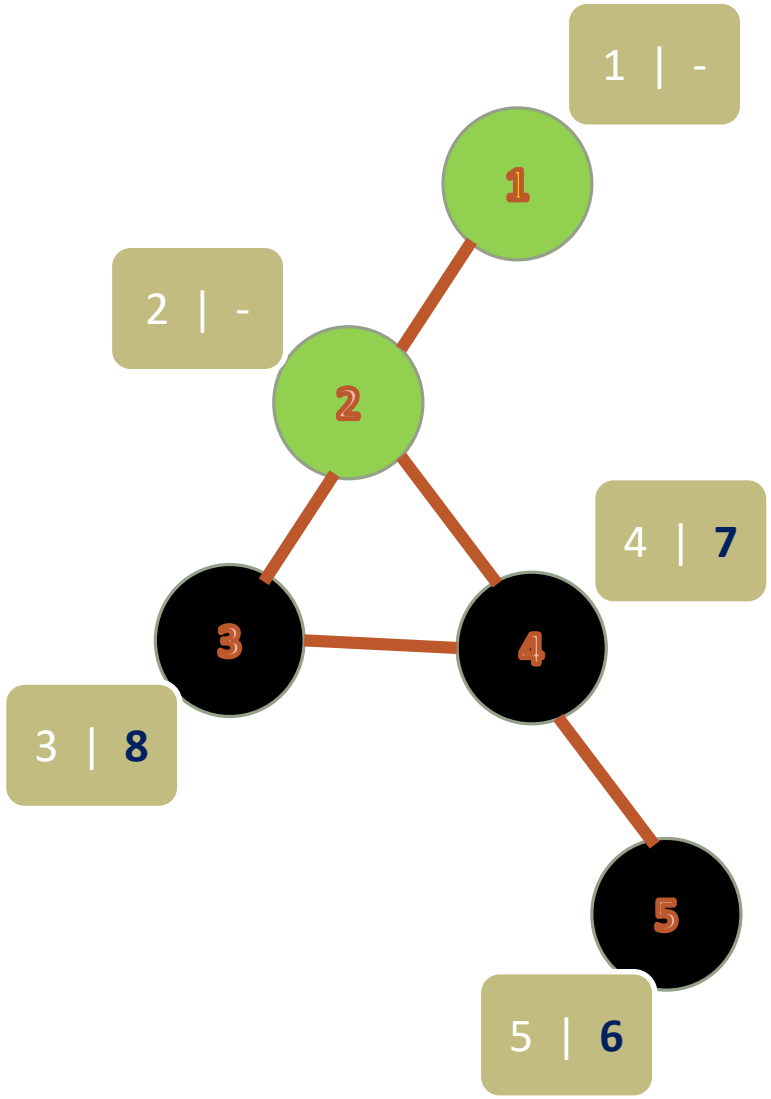
	1	2	3	4	5
Visited	1	1	1	1	-1



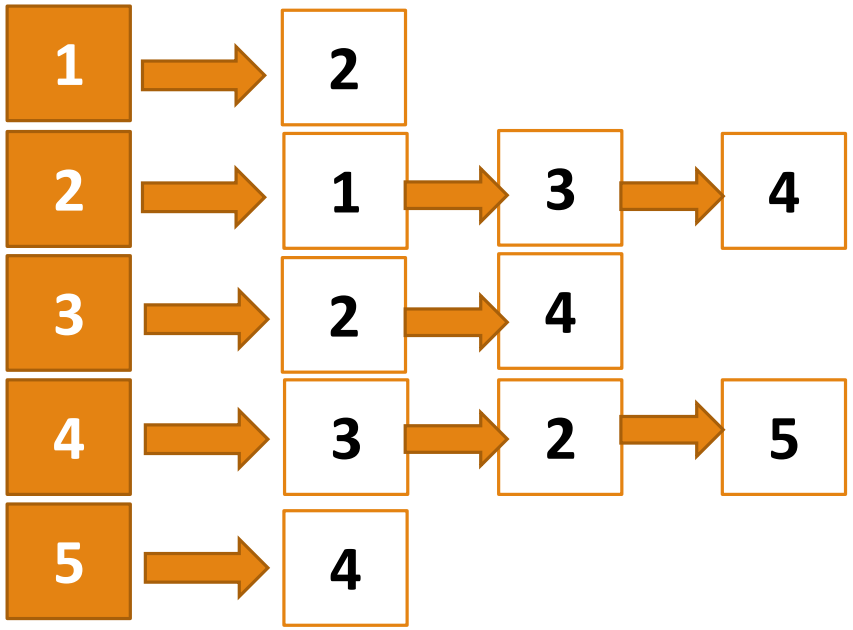


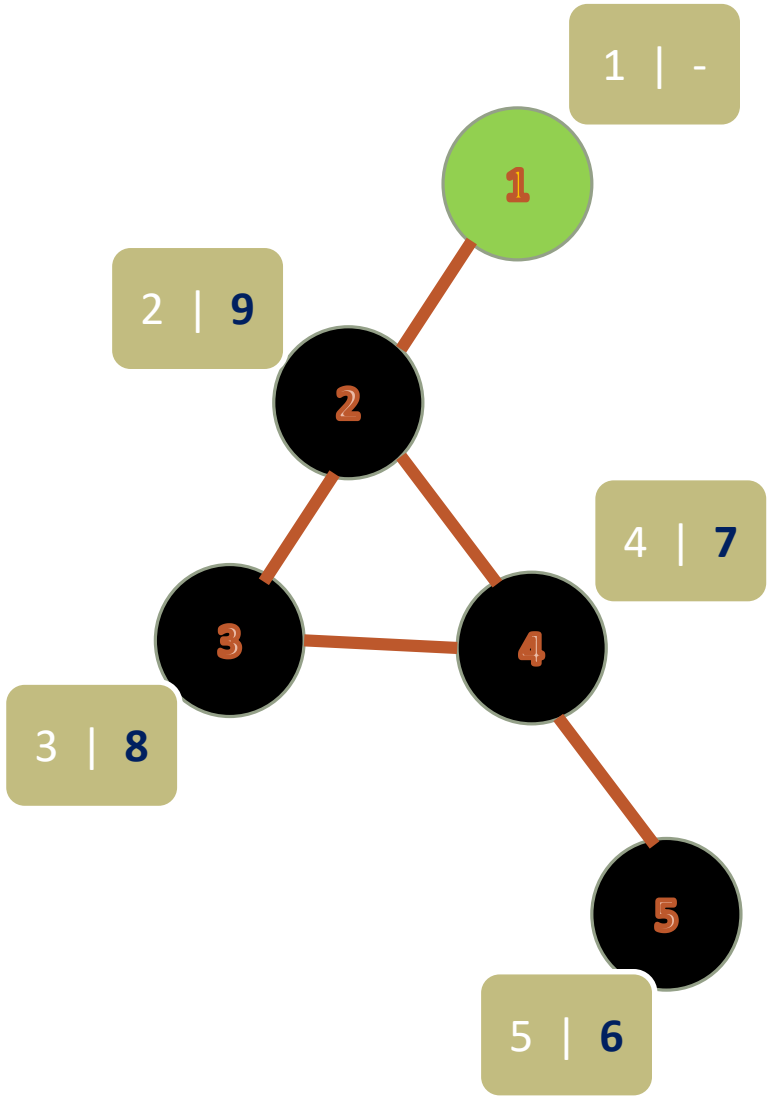
	1	2	3	4	5
Visited	1	1	1	-1	-1



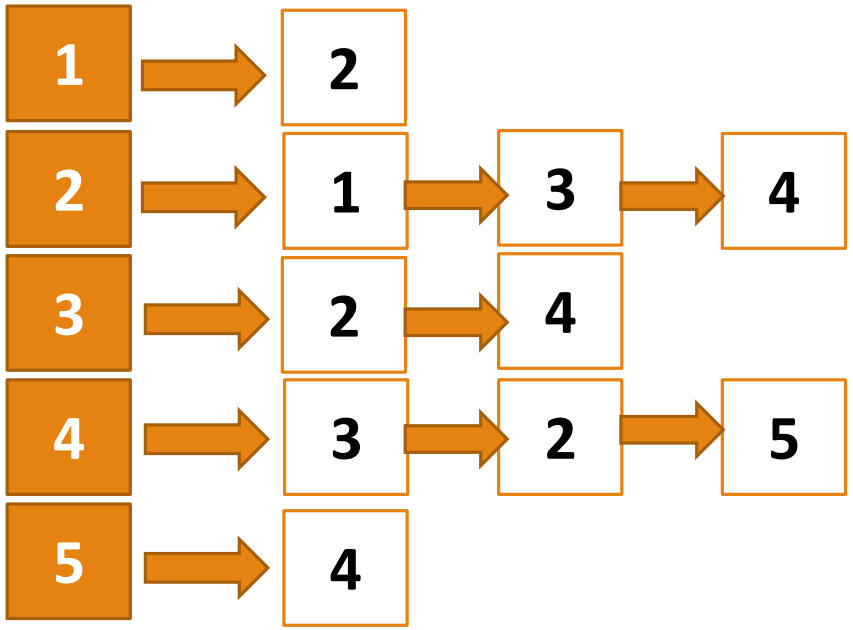


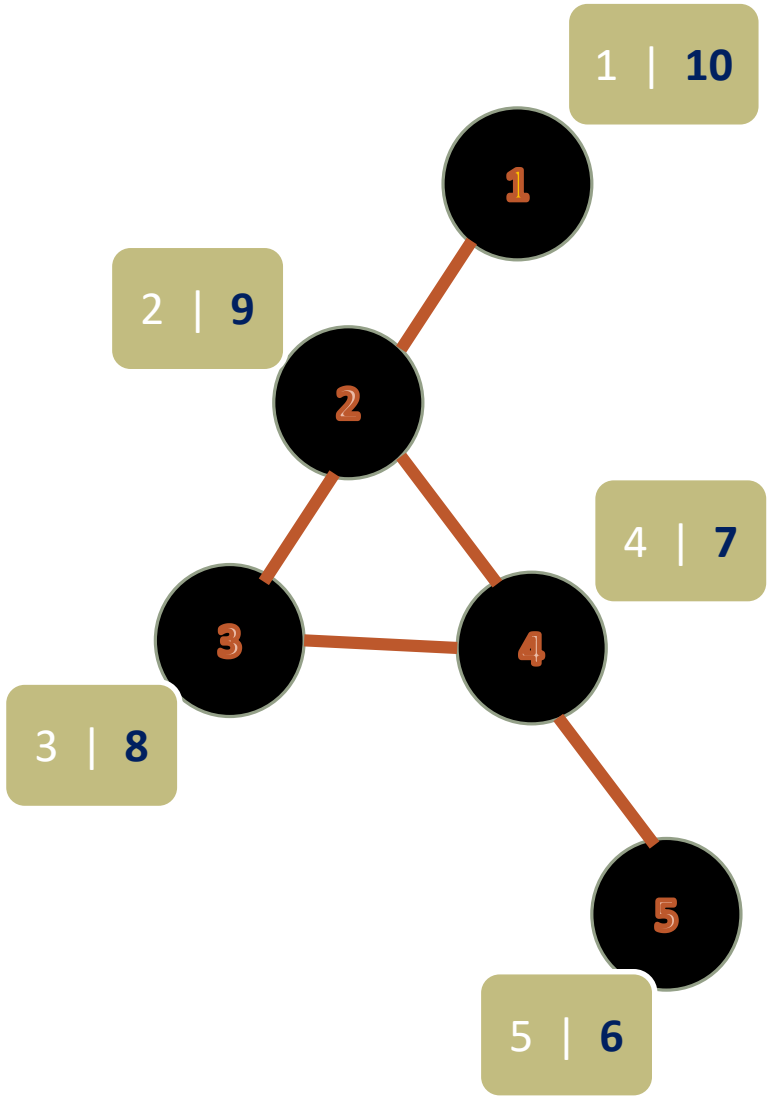
	1	2	3	4	5
Visited	1	1	-1	-1	-1



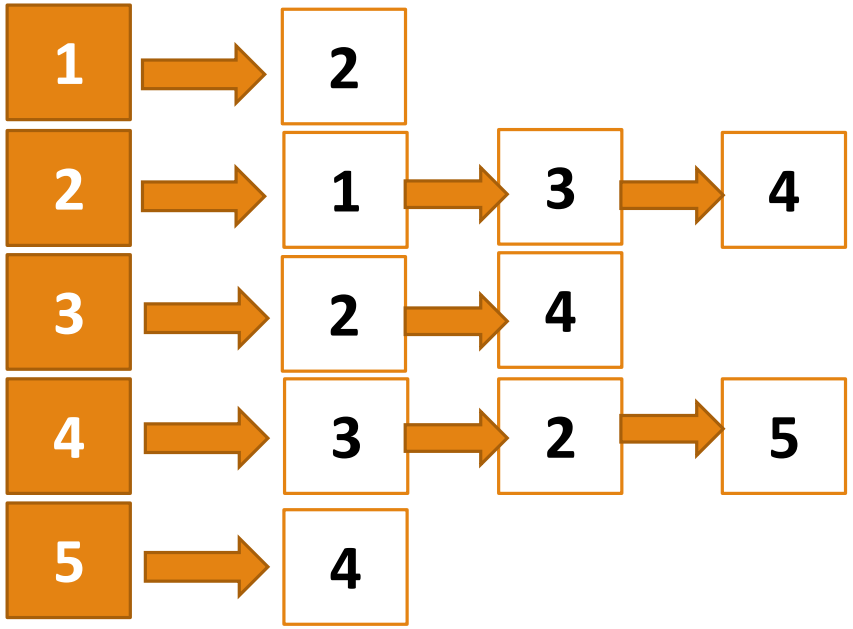


	1	2	3	4	5
Visited	1	-1	-1	-1	-1





	1	2	3	4	5
Visited	-1	-1	-1	-1	-1



DFS Implementation

- ***Vector in C++ STL*:** Vectors are same as dynamic arrays with the ability to resize itself automatically when an element is inserted or deleted, with their storage being handled automatically by the container. Vector elements are placed in contiguous storage so that they can be accessed and traversed using iterators.
- ***memset()*** is used to fill a block of memory with a particular value.

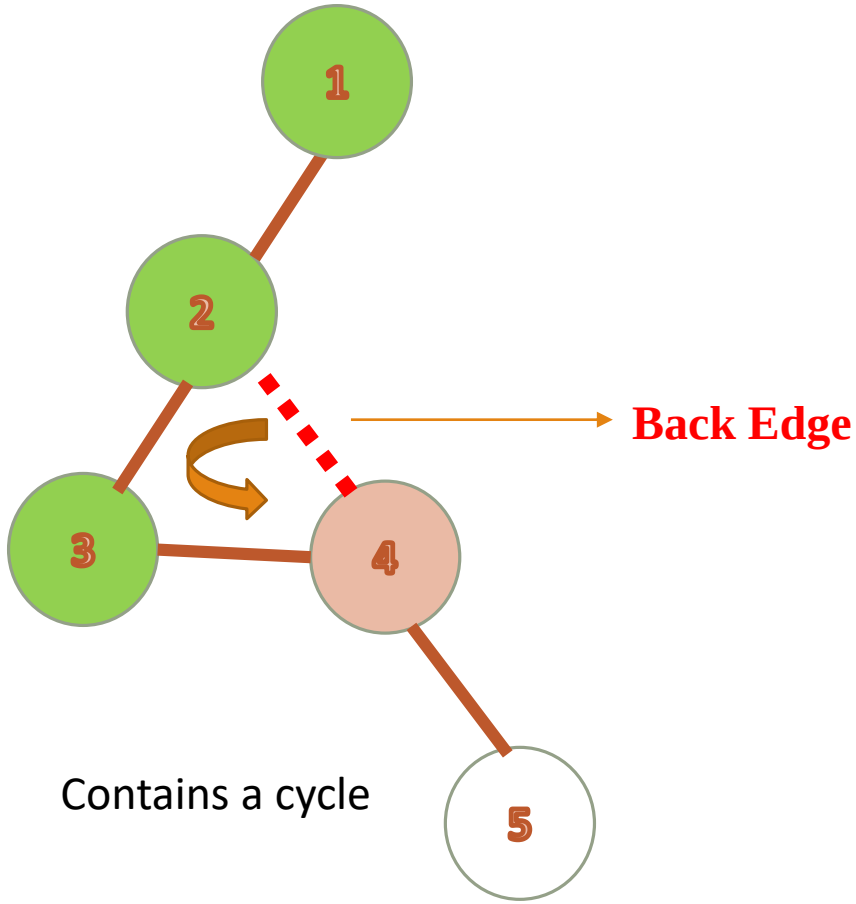
DFS Application

1. **Minimum Spanning Tree:** If we perform DFS on unweighted graph, then it will generate minimum spanning tree.
2. **Detecting a Cycle in a Graph:** A graph has a cycle if we found a back edge during DFS. Therefore, we should run DFS for the graph and verify for back edges.
3. **Path Finding:** We can specialize in the DFS algorithm to search a path between two given vertices ***u*** and ***v***.
4. **Bipartite graph checking:** A bipartite graph is possible if the graph coloring is possible using two colors such that vertices in a set are colored with the same color. So, to check if the graph is bipartite or not we can use DFS.

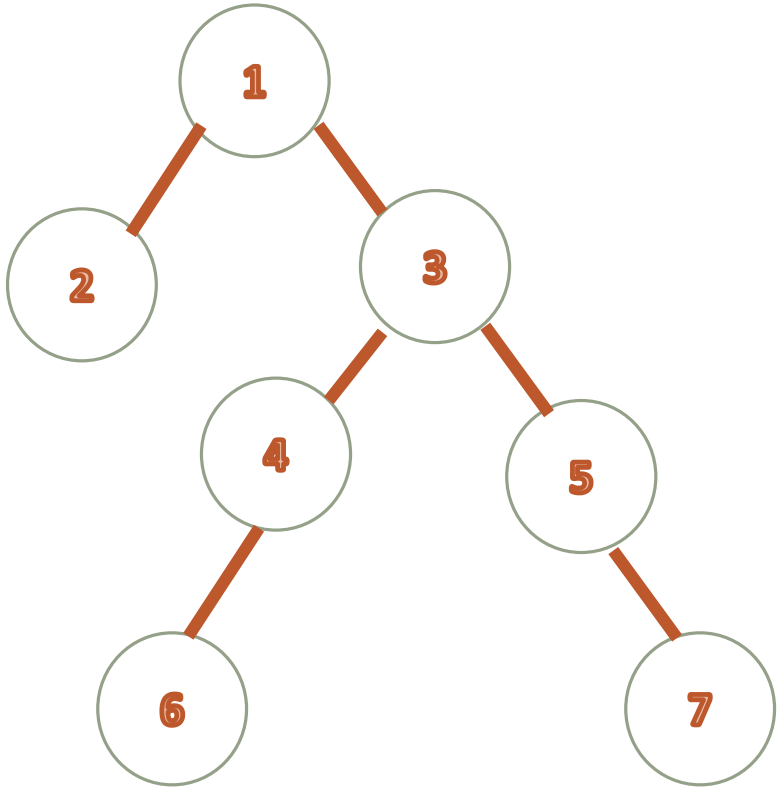
DFS Application

5. **Topological Sorting:** It is primarily used for scheduling jobs from the given dependencies among the group of jobs. In computer science, it is used in instruction scheduling, data serialization, logic synthesis, determining the order of compilation tasks.
6. **Strongly Connected Components:** Using DFS, we can find strongly connected components of a graph. If there is a path from each vertex to every other vertex, that is strongly connected.
7. **Solving Puzzles with Only One Solution:** DFS algorithm can be easily adapted to search all solutions to a maze by including nodes on the existing path in the visited set.

Detecting Cycle



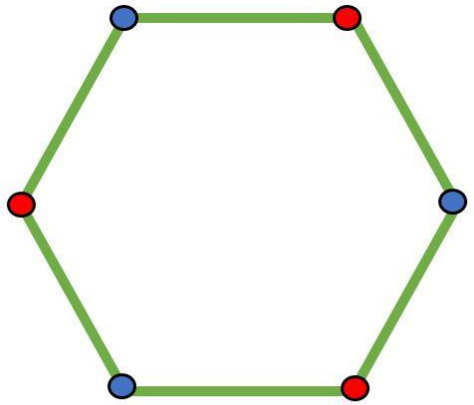
Contains a cycle



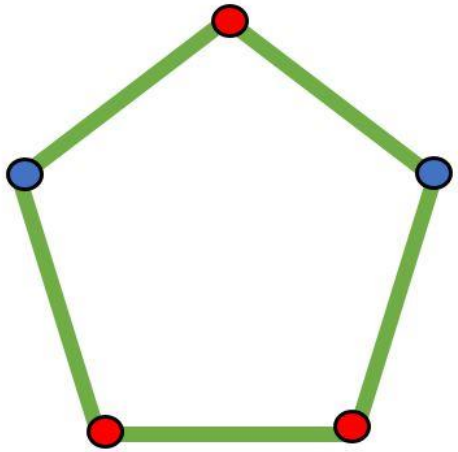
Does not contain any cycle

Bi-coloring

Given a graph, want to give each node a color such that no two neighbors have the same color. Goal is to do color a graph with two colors if possible, else say impossible. Easy !!



Cycle graph of length 6



Cycle graph of length 5

Thank You