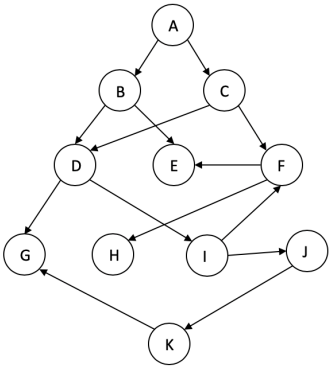
Informatics II
Exercise 12 / **Solution**

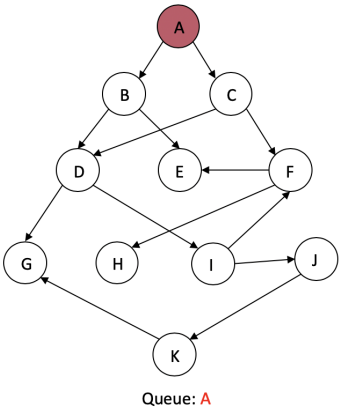
May 19, 2020

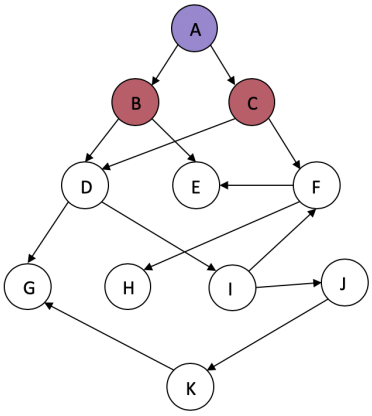
Graphs(BFS, DFS)

Task 1. Given connected graph, write in what order nodes will be visited if BFS algorithm is used. It is assumed that the direction (order) of visiting the neighbors of a node is in the alphabetical order of their labels. First two steps of the solution are presented. Nodes colored in blue denotes visited nodes, while nodes colored in red denotes nodes that are in queue to be visited.

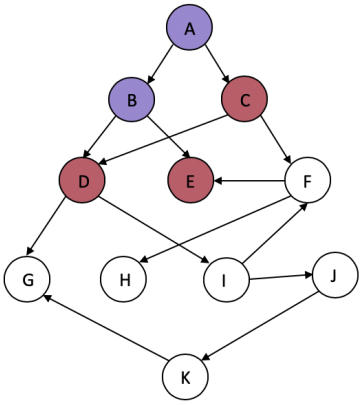


Solution

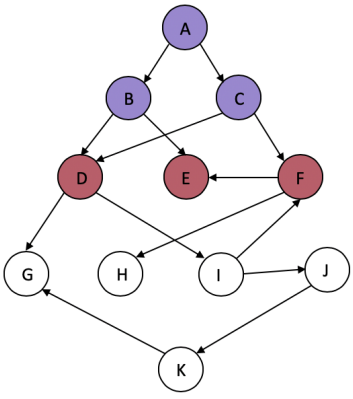




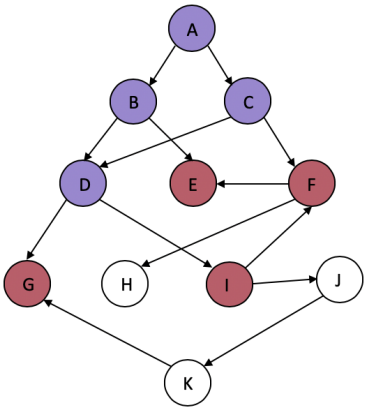
Queue: B, C



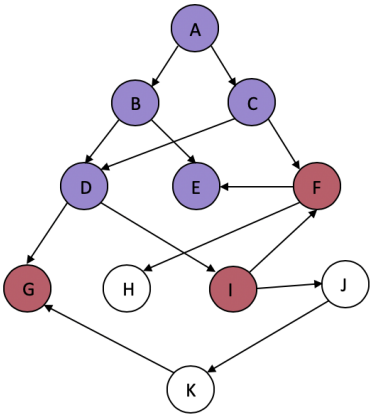
Queue: C, D, E



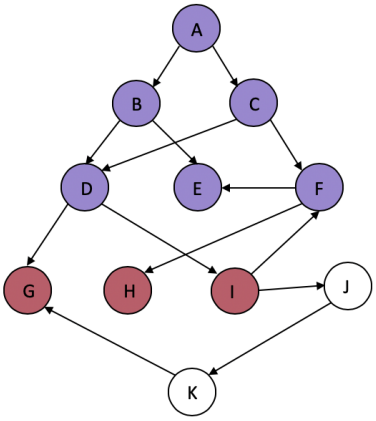
Queue: D, E, F



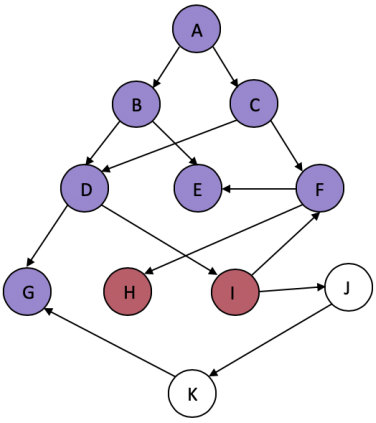
Queue: E, F, G, I



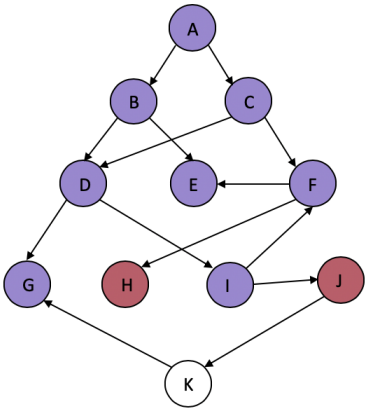
Queue: F, G, I



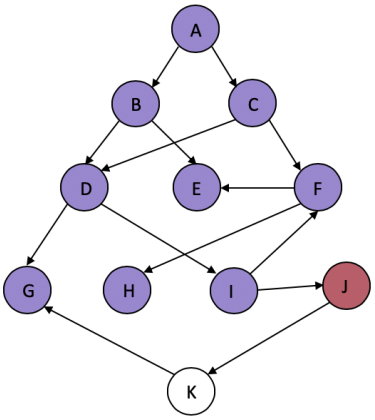
Queue: G, I, H



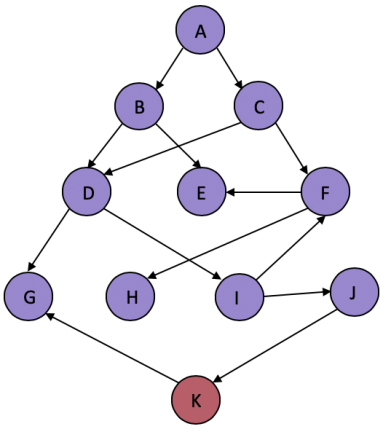
Queue: I, H



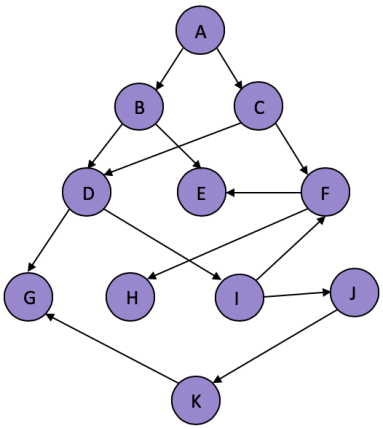
Queue: H, J



Queue: J



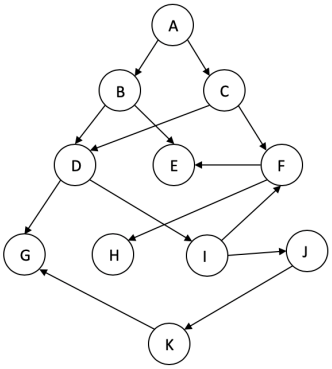
Queue: K



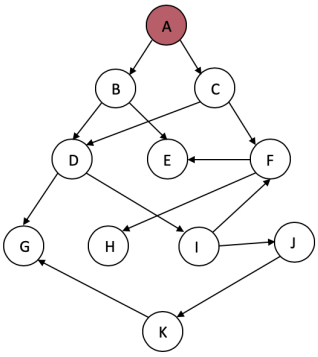
Queue:

The order in which nodes will be visited is: A, B, C, D, E, F, G, I, H, J, K

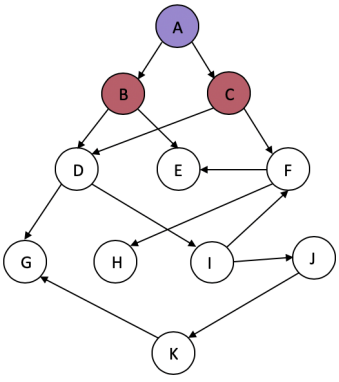
Task 2. Given connected graph, same as in `Task 1.`, write in what order nodes will be visited if DFS algorithm is used. It is assumed that the direction (order) of visiting the neighbors of a node is in the alphabetical order of their labels. First two steps of the solution are presented. Nodes colored in blue denotes visited nodes, while nodes colored in red denotes nodes that are in stack to be visited.



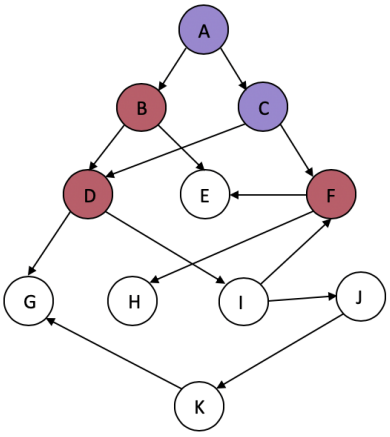
Solution



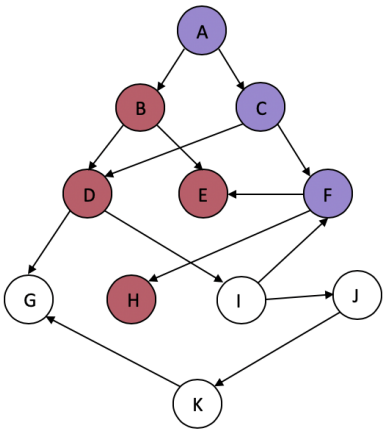
Stack: A



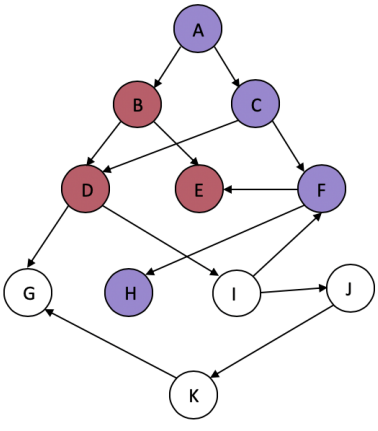
Stack: B, C



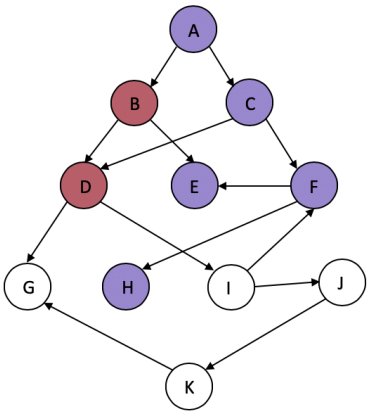
Stack: B, D, F



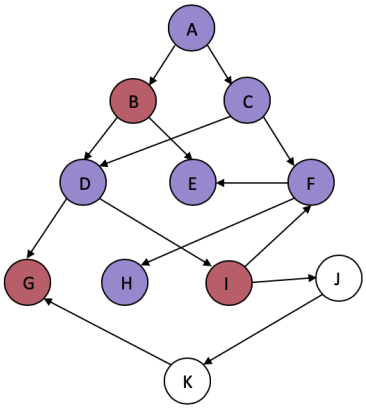
Stack: B, D, E, H



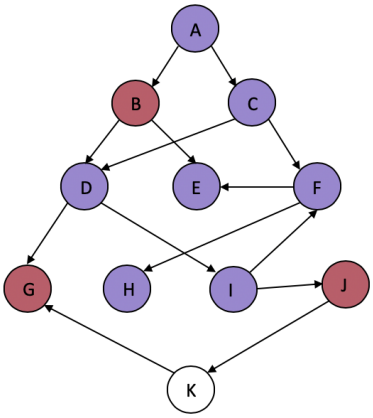
Stack: B, D, E



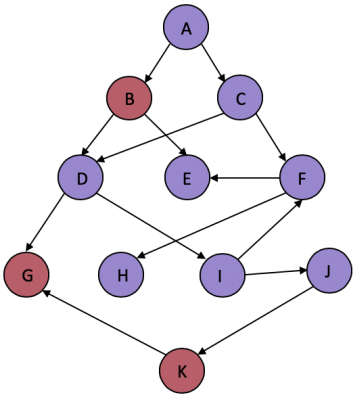
Stack: B, D



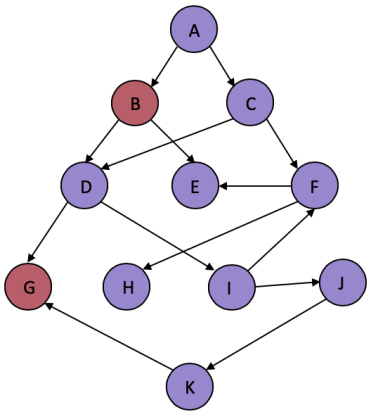
Stack: B, G, I



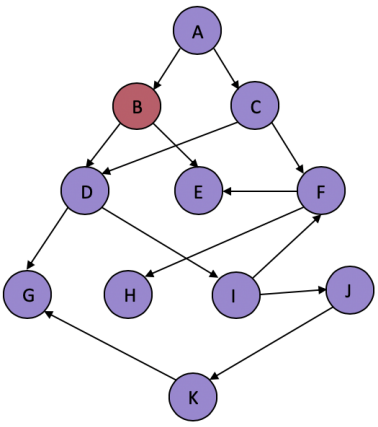
Stack: B, G, J



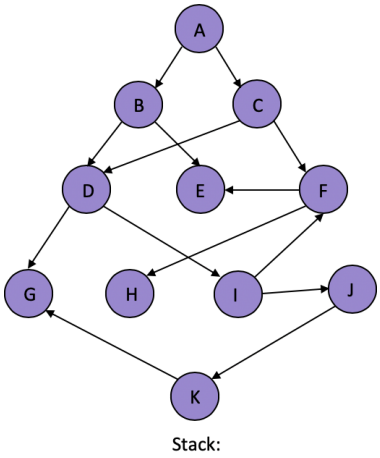
Stack: B, G, K



Stack: B, G



Stack: B

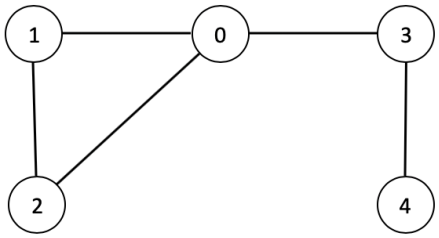


The order in which nodes will be visited is: A, C, F, H, E, D, I, J, K, G, B

Task 3. This task is about detecting cycle in graph using Breadth-first search (BFS). For this task you are given a framework in the file `task3_framework.c`. Queue that will be used in BFS is implemented as a linked list where each element of the list is of the following type:

```
1 struct node {
2     int data;
3     struct node* next;
4 };
```

An undirected graph G has cycle if one can reach the same node following the path in graph. Use Breadth-first search (BFS) to check whether the given graph G has cycles in it or not. Your task is to implement function `bool isSubgraphCyclic(struct Graph* graph, int s)` that returns true if subgraph reachable from node `s` has cycle in it. Example graph can be seen in Figure and it has cycle $0 - 1 - 2 - 0$




```
1 bool isSubgraphCyclic(struct Graph* graph, int s){
2     int V = graph->numVertices;
3     int parent[V];
4     int i = 0;
```

```
5     for (i=0; i<V; i++) {
6         parent[i] = −1;
7     }
8     struct node *Queue = NULL;
9     add(&Queue, s);
10    graph−>visited[s] = true;
11    while (Queue != NULL) {
12        int u = Queue−>data;
13        del(&Queue, &u);
14
15        struct node* tmp = graph−>adjLists[u];
16        while (tmp != NULL) {
17            if (!graph−>visited[tmp−>data]) {
18                graph−>visited[tmp−>data] = true;
19                add(&Queue, tmp−>data);
20                parent[tmp−>data] = u;
21            }
22            else if (parent[u] != tmp−>data) {
23                return true;
24            }
25            tmp = tmp−>next;
26        }
27    }
28    return false;
29 }
30
31 // Function to check cyclic in case of of disconnected graph
32 bool isGraphCyclic(struct Graph* graph)
33 {
34    int V = graph−>numVertices;
35    int i = 0;
36    for (i = 0; i < V; i++) {
37        if (!graph−>visited[i] && isSubgraphCyclic(graph, i))
38            return true;
39    }
40    return false;
41 }
```

Task 4. This task is about implementing depth-first search. For this task you are given a framework in the file `task4_framework.c`.
Stack that will be used in depth-first search is implemented as a linked list where each element of the list is of the following type:

```
1 struct node {
2     int vertex;
3     struct node* next;
4 };
```

Your task is to implement function `void DFS(struct Graph * graph, int vertex)` that prints in what order elements will be visited if DFS algorithm is used. Graph is represented by adjacency matrix and it contains 4 vertices.

```
1  void DFS(struct Graph * graph, int vertex) {
2    struct node* adjList = graph->adjLists[vertex];
3    struct node* temp = adjList;
4
5    graph->visited[vertex] = 1;
6    printf("Visited %d \n", vertex);
7
8    while (temp != NULL) {
9      int connectedVertex = temp->vertex;
10
11     if (graph->visited[connectedVertex] == 0) {
12       DFS(graph, connectedVertex);
13     }
14     temp = temp->next;
15   }
16 }
```