NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

SC1007 Data Structures and Algorithms **2022/23 Semester 2**

Lab 7: Graph Traversal and Backtracking

School of Computer Science and Engineering *Nanyang Technological University*

Q1 Write a function DFS_I() to do a depth search from a input vertex v. The labels of v are from 1 to $|V|$. The algorithm will visit the neighbor nodes in ascending order. The function prototype is given as follows:

```
void DFS_I (Graph g, int v);
```

The structure of a graph is given below.

```
typedef struct _listnode
{
    int vertex;
  struct _listnode *next;
} ListNode;
typedef ListNode StackNode;

typedef struct _graph{
    int V;
    int E;
    int **matrix;
}Graph;

typedef struct _linkedlist
{
  int size;
  ListNode *head;
} Stack;
```

```
void DFS_I(int start, int n, int graph[][n],int visited[]){
    int stack[n];
    int top = -1;

    visited[start] = 1;
    stack[++top] = start;
    while(top>=0){
        int current = stack[top--];
        printf("%d ",current);

        for(int i=0;i<n;i++){
            if(graph[current][i] ==1 && visited[i] ==0){
                visited[i] = 1;
                stack[++top] = i;
}}}}
```

```
DFS_i(Graph g, int v){
    Stack s;
    s.size =0;
    s.head = NULL;
    int w;

    push(&s,v);
    g.visited[v-1] = 1;
    printf("%d ",v);

    int i;
    int stopNode;

    while(!isEmptyStack(s)){
      w = peek(s);
      stopNode = 0;
      for(i=0;i<g.V;i++){
        if(g.matrix[w-1][i] ==1
&&g.visited[i]==0){
            push(&s,i+1);
            g.visited[i] = 1;
            printf("%d ", i+1);
            stopNode = 1;
            break;

    if(stopNode ==0)
      pop(&s);
```

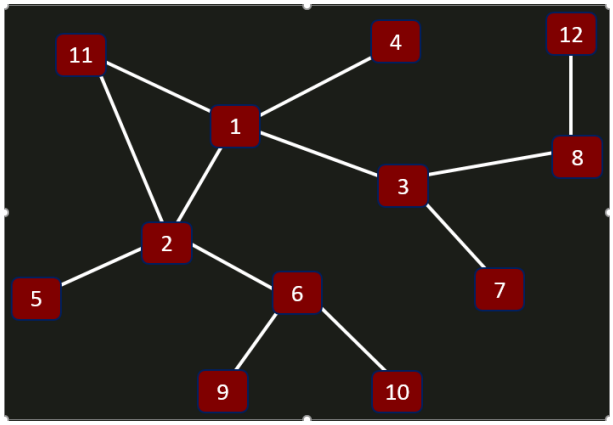
A test sample graph and its expected output are given below: The start vertex for DFS is vertex 11.

```
void DFS_R(int start, int n, int graph[][n], int visited[]){
    visited[start] = 1;
    printf("%d ",start);

    for(int i=0;i<n;i++){
        if(graph[start][i] ==1 && visited[i] ==0){
            DFS_R(i,n,graph,visited);
```

```
DFS_R(Graph g, int v){
  int i;
  g.visited[v-1] = 1;
  printf("%d ", v);

  for(i = 0;i<g.V;i++){
    if(g.matrix[v-1][i]==1&&g.visited[i]==0){
      DFS+R(g,i+1);
```



The expected output: 5 9 10 6 2 7 12 8 3 4 1 11

Q2 Rewrite a depth search algorithm in a recursive approach. The function prototype is given as follows:

```
void DFS_R (Graph_DFS g, int v);
```

Q3 Write a function, nQueens(), to print out all the possible solutions of the N-queen problem.

```
int nQueens(int** board, int N, int col);
```

The number of possible solutions to different n are:

n	number of possible solutions
4	2
5	10
6	4
7	40
8	92
10	724
12	14200

There is no known formula for the exact number of solutions but the grown rate is extremely high.

```
int nQueens(int **board,int N, int col):

    int count = 0;

    if(col ==N){
        printf("%d", ++count);
        for i
            for j
                printf("%d", board[i][j]);
        return;
     }

    for i
       if(is safe(board,col,i)){
            board[col][i] = 1;
            nQueens(board,N,col+1);
            board[col][i] = 0;
    }}}
```

```
int is_safe(int board[N][N], int row, int col){
    for i
        if(board[i][col] ==1){
            return 0;

    for i=row;j=col;i>=0&&j>=0;i--;j--
        if(board[i][j] ==1){
            return 0;

    for(i=row;j=col;i>=0&&j<N;i--;j++){
        if(board[i][j] == 1){
            return 0;

return 1;
```