CRACK A HACK

Road Maintenance

Course: ALGORITHMIC PROBLEM SOLVING

Course code: 17ECSE309

by:

Pooja S Galer

USN: 01FE15BCS130

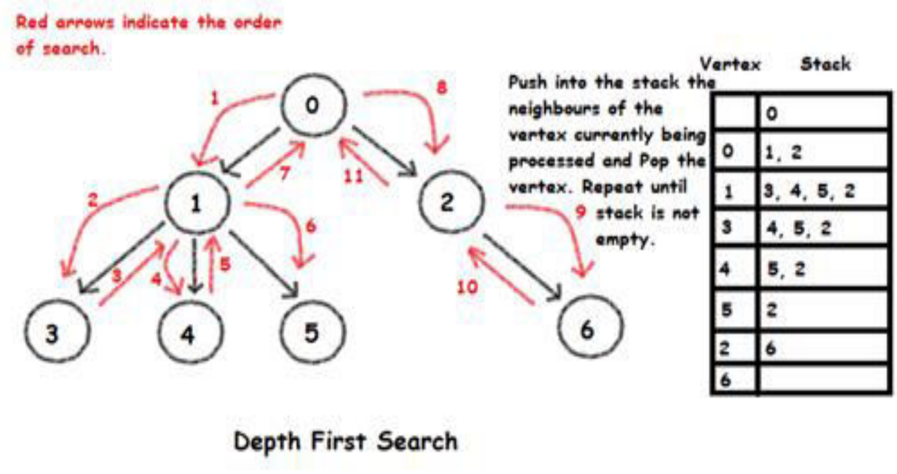
1. Introduction

Byteland has N cities (numbered from 1 to N) and N-1 bidirectional roads. A *path* is comprised of 1 or more connected roads. It is guaranteed that there is a path from any city to any other city.

Steven is a road maintenance worker in Byteland. He is required to maintain *exactly* M paths on any given workday. He *cannot* work on the same road twice in one day (so no 2 paths can contain the same 2 roads). Steven can start his workday in any city and, once he has finished maintaining a path, teleport to his next starting city.

The problem is related to Depth First Search (DFS) for given M level from every node to M-1 nodes in all directions. Here, we travel in same direction until M levels are met, then the starting node changed.

2. Example



Red arrows indicate the order of search.

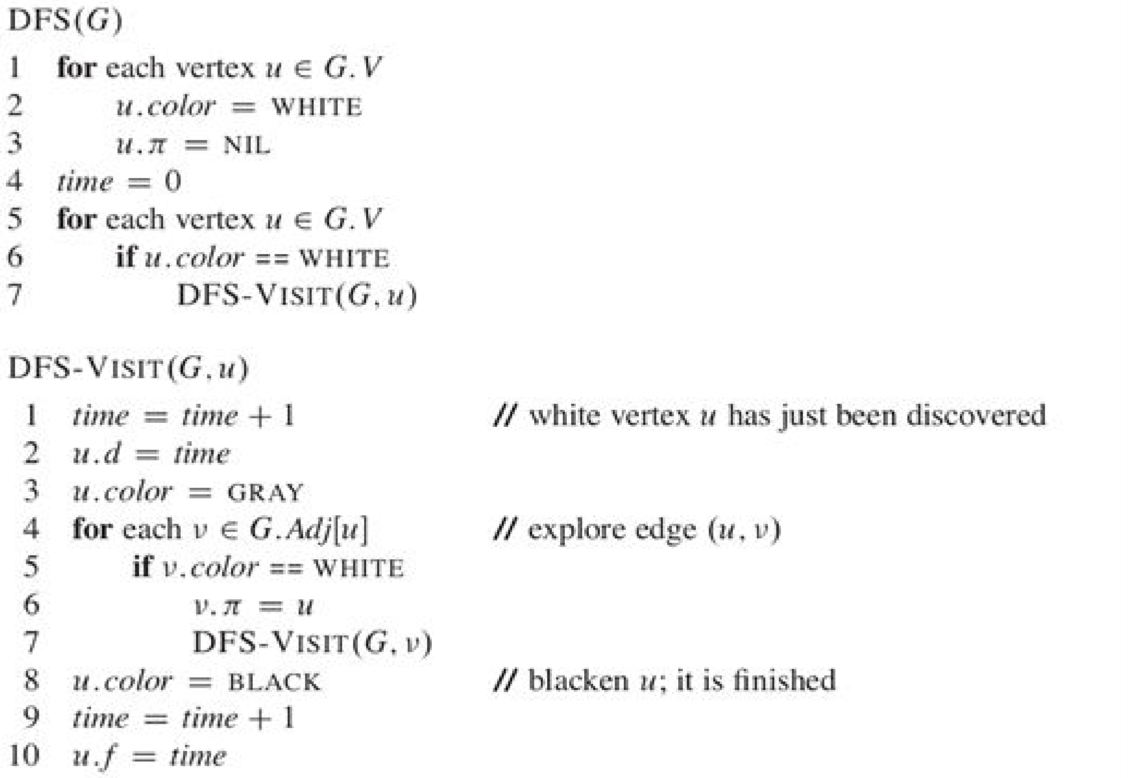
Push into the stack the neighbours of the vertex currently being processed and Pop the vertex. Repeat until stack is not empty.

Vertex	Stack
	0
0	1, 2
1	3, 4, 5, 2
3	4, 5, 2
4	5, 2
5	2
2	6
6	

Depth First Search

Figure 2.1: DFS Algorithm example

3. Algorithm



```
DFS($G$)
for each vertex $u \in G.V$
    $u.color$ = WHITE
    $u.\pi$ = NIL
$time = 0$
for each vertex $u \in G.V$
    if $u.color$ == WHITE
        DFS-VISIT($G, u$)

DFS-VISIT($G, u$)
$time = time + 1$                // white vertex $u$ has just been discovered
$u.d = time$
$u.color$ = GRAY
for each $v \in G.Adj[u]$        // explore edge $(u, v)$
    if $v.color$ == WHITE
        $v.\pi = u$
        DFS-VISIT($G, v$)
$u.color$ = BLACK                // blacken $u$; it is finished
$time = time + 1$
$u.f = time$
```

Figure 3.1: DFS Algorithm Pseudocode

4. Code (in C)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
typedef struct way{
  int x;
  int w;
  struct way *next;
} path;
#define MOD 1000000007
#define MAX 100000
int m;
int tp[MAX]={0};
long long dp[6][6][MAX]={0};
```

```
path *table[MAX]={0};
void dfs(int x){
int i,j,k,l;
 long long t[6][6];
 path *p;
 tp[x]=1;
 dp[0][0][x]=1;
 for(p=table[x];p;p=p->next)
  if(!tp[p->x]){
   dfs(p->x);
   memset(t,0,sizeof(t));
   for(i=0;i<=m;i++)
    for(j=0;i+j<=m+1;j++)
     for(k=0;k<=i;k++)
      for(l=0;l<=j;l++){
       if(i+j<=m){
        t[k][i+j]=(t[k][i+j]+dp[k][i][x]*dp[l][j][p->x])%MOD;
        if(k)
         t[k-1][i+j]=(t[k-1][i+j]+dp[k][i][x]*dp[l][j][p->x]%MOD*k)%MOD;
        if(k+1<=i+j)
         t[k+1][i+j]=(t[k+1][i+j]+dp[k][i][x]*dp[l][j][p->x]%MOD*l)%MOD;
       }
       if(i+j && k)
        t[k-1][i+j-1]=(t[k-1][i+j-1]+dp[k][i][x]*dp[l][j][p->x]%MOD*k*l)%MOD;
       if(i+j+1<=m)
        t[k+1][i+j+1]=(t[k+1][i+j+1]+dp[k][i][x]*dp[l][j][p->x])%MOD;
      }
   for(i=0;i<=m;i++)
    for(j=0;j<=m;j++)
     dp[i][j][x]=t[i][j]%MOD;
  }
 return;
}

void addto(int x,int y,int w){
 path *t=(struct way*)malloc(sizeof(path));
 t->x=y;
 t->w=w;
 t->next=table[x];
 table[x]=t;
 return;
}
int main(){
 int N,x,y,i;
 long long order;
 scanf("%d%d",&N,&m);
 for(i=0;i<N-1;i++){
 scanf("%d%d",&x,&y);
 addto(x-1,y-1,1);
```

```
  dfs(0);
  for(i=ans=0;i<=m;i++)
    order=(order+dp[i][m][0])%MOD;
  printf("%lld",order);
  return 0;
}
```

5. Time Complexity

The time complexity of the above code is $O(|N-1|+|N|)$, Where ‘N-1’ is the number of edges(roads) and ‘N’ is the number of cities.

6. Applications

- Finding Connectivity in graphs.
- Topological ordering of jobs based on dependencies
- Planarity Testing

7. References

1) https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/
2) https://www.programiz.com/dsa/graph-dfs