```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long

```

## Normal Dfs:

```cpp
vector<int> visited(100100 , 0) , adj[100100];

void dfs(int u){
    visited[u] = 1;
    for(auto v : adj[u]){
        if(!visited[v])
            dfs(v);
    }
}

```

## Dfs for grid:

```cpp
//for four direction:   S    E     N     w
            int dx[] = {1 , 0 , -1 ,  0};
            int dy[] = {0 , 1 ,  0 , -1};
//for eight direction:  S   SE  E    NE   N    NW   W    SW
//          int dx[] = {1 , 1 , 0 , -1 , -1 , -1 ,  0 ,  1};
//          int dy[] = {0 , 1 , 1 ,  1 ,  0 , -1 , -1 , -1};

vector<vector<int>> visited_(1000 , vector<int>(1000));
int n , m;

void dfs(int x , int y){
    visited_[x][y] = 1;
    for(int i = 0 ; i < 8/4 ; i++){ //check 8/4
        int xx = x+dx[i];
        int yy = y+dy[i];
        if(xx>=0 && xx<n && yy>=0 && yy<m && !visited_[xx][yy])
            dfs(xx , yy);
    }
}

```

## Dfs for Tree:

```cpp
vector<int> adj[100100];

void dfs(int u , int parent){
    for(auto v : adj[u])
        if(v!=parent)
            dfs(v , u);
}


```

## Dfs order:

```cpp
vector<int> visited(100100) , a(100100) , in(100100) , out(100100);
int cnt = 0;

void dfs_order(int u){
    visited[u] = 1;
    in[u] = ++cnt;
    a[cnt] = u;
    for(auto v : adj[u]){
        if(!visited[v])
            dfs_order(v);
    }
    out[u] = cnt;
}
```