#include **<iostream>**#include **<stdint.h>**#include **<vector>**#include **<cstring>  
  
using namespace** std;  
  
*//const long long mx=(1<<63)-1;  
//long long int h[mx][64];* **bool** visited[104];  
**int** parent[100];  
**int** start[100];  
**int** finish[100];  
**int** counter=0;  
vector<**int**> arr[100];  
**bool** dfs(**int** u,**int** t){  
 **if**(u==t){  
 start[u]=counter++;  
 finish[u]=counter++;  
 **return true**;  
 }  
 start[u]=counter++;  
 visited[u]=**true**;  
 **for**(**int** i=0;i<arr[u].size();i++ ){  
 **int** v=arr[u][i];  
 **if**(!visited[v]){  
 parent[v]=u;  
 dfs(v,t);  
 }  
 }  
 finish[u]=counter++;  
 **return false**;  
}  
**void** print\_path(**int** s,**int** t){  
 **if**(s==t){  
 cout<<t<<**" -> "**;  
 **return**;  
 }  
 print\_path(s,parent[t]);  
 cout<<t<<**" -> "**;  
}  
  
**int** main() {  
 **int** n;cin>>n;  
 **for**(**int** i=0;i<n;i++){  
 **int** u;cin>>u;  
 **int** v;cin>>v;  
 arr[u].push\_back(v);  
 }  
 *//memset(visited,false, sizeof(visited));* **for**(**int** i=0;i<n;i++){  
 parent[i]=i;  
 visited[i]=**false**;  
 start[i]=-1;  
 finish[i]=-1;  
 }  
 dfs(0,n-1);  
 print\_path(0,n-1);  
 **for**(**int** i=0;i<n;i++){  
 cout<<endl<<start[i]<<**" - "**<<finish[i]<<endl;  
 }  
 **return** 0;  
}

1. #include <iostream>
2. #include <map>
3. #include <queue>
5. **using** **namespace** std;
7. map<string ,bool> map1;
8. string start1,end1;
10. int mybfs(){
11. queue<pair<int,string > >pq;
12. pq.push(make\_pair(0,start1));
13. while(! pq.empty()){
15. pair<int , string > tmp;
16. tmp=pq.front();
18. pq.pop();
19. if(map1[tmp.second]==1)continue;
20. if(tmp.second==end1)return tmp.first;
21. map1[tmp.second]=1;
22. for(int i=0;i<3;i++){
23. string s=tmp.second;
24. if(s[i]=='z'){
25. s[i]='a';
26. }else{
27. s[i]=(char)((int)s[i]+1);
28. }
29. pq.push(make\_pair(tmp.first+1,s));
31. s=tmp.second;
32. if(s[i]=='a'){
33. s[i]='z';
34. }else{
35. s[i]=(char)((int)s[i]-1);
36. }
37. pq.push(make\_pair(tmp.first+1,s));
39. }
40. }
41. return -1;
42. }

void Graph::BFS(int s)

{

    bool \*visited = new bool[V];

    for(int i = 0; i < V; i++)

        visited[i] = false;

    list<int> queue;

    visited[s] = true;

    queue.push\_back(s);

    list<int>::iterator i;

    while(!queue.empty())

    {        s = queue.front();

        cout << s << " ";

        queue.pop\_front();

        for (i = adj[s].begin(); i != adj[s].end(); ++i)

        {

            if (!visited[\*i])

            {

                visited[\*i] = true;

                queue.push\_back(\*i);

            }

        }

    }

}