题意

给一棵n个结点的无根树，给长度为m的序列a（ai∈[1,n])
从1号节点出发，经过每条边恰好两次,问能否按a序列的先后顺序访问a的每个结点

分析

将1看作根节点，转化成有根树
从1开始dfs用bitset类型的变量bb[i] 标记第i个结点为根节点的子树的所有结点
bb[now] = bb[to1] | bb[to2] | bb[to3] ... 再将now标记
to为now 的儿子结点
搜索的时候，走过的边标记，只走没标记过的边，当前需要找的点是a[x] ,则每次找到a[x]所在的子树进行搜索，如果找到a[x] 则x++,知道x>m返回
如果搜索结束，x<=m 则无法按给定顺序搜索，输出NO，x>m 输出YES
bitset用法

定义一个**bitset**

```
// constructing bitsets
#include <iostream>       // std::cout
#include <string>         // std::string
#include <bitset>         // std::bitset

int main ()
{
  std::bitset<16> foo;
  std::bitset<16> bar (0xfa2);
  std::bitset<16> baz (std::string("0101111001"));

  std::cout << "foo: " << foo << '\n';
  std::cout << "bar: " << bar << '\n';
  std::cout << "baz: " << baz << '\n';

  return 0;
}
```

输出结果：

```
foo: 0000000000000000
bar: 0000111110100010
baz: 0000000101111001
```

bitset的运算

bitset的运算就像一个普通的整数一样，可以进行与(&)、或(|)、异或(^)、左移(<<)、右移(>>)等操作。

```
// bitset operators
#include <iostream>       // std::cout
#include <string>         // std::string
#include <bitset>         // std::bitset

int main ()
{
  std::bitset<4> foo (std::string("1001"));
  std::bitset<4> bar (std::string("0011"));

  std::cout << (foo^=bar) << '\n';       // 1010 (XOR,assign)
  std::cout << (foo&=bar) << '\n';       // 0010 (AND,assign)
  std::cout << (foo|=bar) << '\n';       // 0011 (OR,assign)

  std::cout << (foo<<=2) << '\n';        // 1100 (SHL,assign)
  std::cout << (foo>>=1) << '\n';        // 0110 (SHR,assign)

  std::cout << (~bar) << '\n';           // 1100 (NOT)
  std::cout << (bar<<1) << '\n';         // 0110 (SHL)
  std::cout << (bar>>1) << '\n';         // 0001 (SHR)

  std::cout << (foo==bar) << '\n';       // false (0110==0011)
  std::cout << (foo!=bar) << '\n';       // true  (0110!=0011)

  std::cout << (foo&bar) << '\n';        // 0010
  std::cout << (foo|bar) << '\n';        // 0111
  std::cout << (foo^bar) << '\n';        // 0101

  return 0;
}
```

上面代码的输出结果见注释。（注意，这段代码涉及赋值操作）

bitset的相关函数

对于一个叫做foo的bitset：

`foo.size()` 返回大小（位数）

`foo.count()` 返回1的个数

`foo.any()` 返回是否有1

`foo.none()` 返回是否没有1

`foo.set()` 全都变成1

`foo.set(p)` 将第p + 1位变成1

`foo.set(p, x)` 将第p + 1位变成x

`foo.reset()` 全都变成0

`foo.reset(p)` 将第p + 1位变成0

`foo.flip()` 全都取反

`foo.flip(p)` 将第p + 1位取反
`foo.to_ulong()` 返回它转换为unsigned long的结果，如果超出范围则报错
`foo.to_ullong()` 返回它转换为unsigned long long的结果，如果超出范围则报错
`foo.to_string()` 返回它转换为string的结果

来源： https://www.cnblogs.com/RabbitHu/p/bitset.html

代码

```
#include<algorithm>
#include<bitset>
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<cmath>
#include<deque>
#include<iostream>
#include<map>
#include<queue>
#include<set>
#include<stack>
#include<string>
#include<vector>
#include<list>
#define For(i,a,b) for(int i=(a); i<=(b) ; i++)
#define _For(i,a,b) for(int i=(a); i>=(b) ; i--)
#define Memset(a,b); memset((a),(b),sizeof((a)));
#define Cout(a,b);  printf("%d",(a));printf(b);
#define Coutc(a,b);  printf("%c",(a));printf(b);
#define Couts(a,b);  printf("%s",(a));printf(b);
using namespace std;
const int INF = 0x3f3f3f3f;
typedef  long long LL;typedef  unsigned long long ULL;typedef  long double LDB;
inline LL CinLL(){LL x=0,f=1;char ch=getchar();while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}return x*f;}
inline int Cin(){int x=0,f=1;char ch=getchar();while(!isdigit(ch)){if(ch=='-')f=-1;ch=getchar();}while(isdigit(ch))x=x*10+ch-'0',ch=getchar();return f*x;}
int e_book[105][105];
bitset<105>bb[100];
int fa[105];
int e[105][105];
```

```cpp
int n,m;
int a[105];
void dfs(int now)
{
    For(i,1,n)
    {
        if(i == fa[now]) continue;
        if(e[now][i] == 1)
        {
            fa[i] = now;
            dfs(i);
            bb[now] |= bb[i];
        }
    }
    bb[now].set(now);
}
int tot = 1;
void dfs2(int now)
{
    if(a[tot] == now) tot++;
    if(tot > m) return;
    For(i,1,n)
    {
        if(i == fa[now]) continue;
        if(e[now][i] == 1 && e_book[now][i] == 0 && bb[i][a[tot]] == 1)
        {
            e_book[now][i] = 1;
            dfs2(i);
            if(tot > m) return ;
            i = 0;
        }
    }
}
int main()
{
    ios::sync_with_stdio(false);
    int _;cin>>_;
    while(_--)
    {
        cin>>n;
        int x,y;
        Memset(e,0);
        For(i,1,n-1) {
            cin>>x>>y;
            e[x][y] = e[y][x] = 1;
        }
```

```
77          dfs(1);
78          cin>>m;
79          For(i,1,m) cin>>a[i];
80          tot = 1;
81          dfs2(1);
82          if(tot > m) cout<<"YES"<<endl;
83          else cout<<"NO"<<endl;
84          For(i,1,n) bb[i].reset();
85          Memset(e_book,0)
86          Memset(fa,0);
87      }
88  }
```