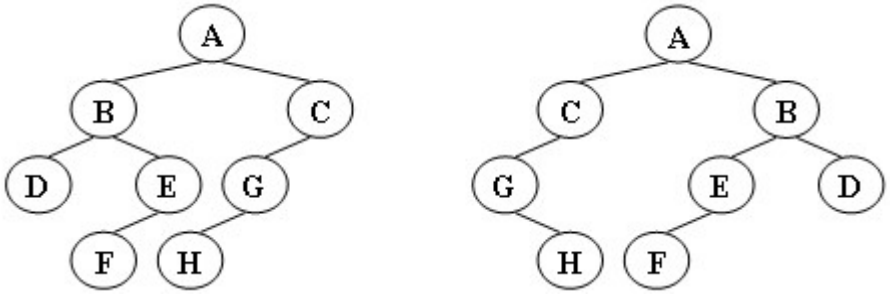
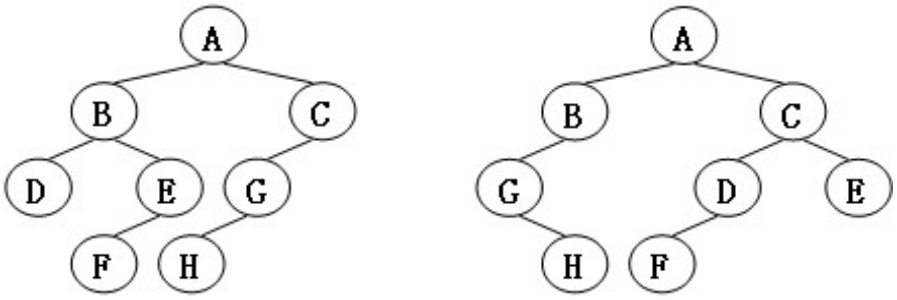
青少年软件编程（C 语言）等级考试试卷（七级）

1 树的同构

给定两棵树 T_1 和 T_2。如果 T_1 可以通过若干次左右孩子互换就变成 T_2，则我们称两棵树是“同构”的。例如图 1 给出的两棵树就是同构的，因为我们把其中一棵树的结点 A、B、G 的左右孩子互换后，就得到另外一棵树。而图 2 就不是同构的。



| 图 1 |



|图 2|

现给定两棵树，请你判断它们是否是同构的。

时间限制：1000
内存限制：65536

输入

输入给出 2 棵二叉树的信息。对于每棵树，首先在一行中给出一个非负整数 n(≤ 10)，即该树的结点数（此时假设结点从 0 到 n -1 编号）；随后 n 行，第 i 行对应编号第 i 个结点，给出该结点中存储的 1 个英文大写字母、其左孩子结点的编号、右孩子结点的编号。

如果孩子结点为空，则在相应位置上给出 “-”。给出的数据间用一个空格分隔。注意：题目保证每个结点中存储的字母是不同的。

输出
如果两棵树是同构的，输出“Yes”，否则输出“No”。

样例输入

```
样例#1（对应图 1）：

8
A 1 2
B 3 4
C 5 -
D - -
E 6 -
G 7 -
F - -
H - -
8
G - 4
B 7 6
F - -
A 5 1
H - -
C 0 -
D - -
E 2 -


样例#2（对应图 2）：

8
B 5 7
F - -
A 0 3
C 6 -
H - -
```

```
D - -
G 4 -
E 1 -
8
D 6 -
B 5 -
E - -
H - -
C 0 2
G - 3
F - -
A 1 4
```

样例输出

```
样例#1：
Yes


样例#2：
No
```

参考代码：

```
#include <iostream>
#include <vector>
using namespace std;

struct Node {
    char c;
    int l, r;
};

bool isIsomorphic(Node t1[], Node t2[], int i, int j) {
    if (t1[i].c != t2[j].c) return false;
    if (t1[i].l == -1 && t2[j].l != -1 || t1[i].r == -1 && t2[j].r != -1) return false;
    if (t1[i].l != -1 && t2[j].l != -1 && !isIsomorphic(t1, t2, t1[i].l, t2[j].l)) return false;
    if (t1[i].r != -1 && t2[j].r != -1 && !isIsomorphic(t1, t2, t1[i].r, t2[j].r)) return false;
    return true;
```

```
}

int main() {
    int n1, n2;
    cin >> n1;
    vector<Node> t1(n1 + 1);
      for (int i = 0; i < n1; ++i) cin >> t1[i].c >> t1[i].l >> t1[i].r, t1[i].l = (t1[i].l == '-') ? -1 : stoi(string(1, t1[i].l)), t1[i].r = (t1[i].r == '-') ? -1 : stoi(string(1, t1[i].r));
    cin >> n2;
    vector<Node> t2(n2 + 1);
      for (int i = 0; i < n2; ++i) cin >> t2[i].c >> t2[i].l >> t2[i].r, t2[i].l = (t2[i].l == '-') ? -1 : stoi(string(1, t2[i].l)), t2[i].r = (t2[i].r == '-') ? -1 : stoi(string(1, t2[i].r));
    cout << (isIsomorphic(t1, t2, 0, 0) ? "Yes" : "No") << endl;
}
```

2 网红点打卡攻略

一个旅游景点，如果被带火了的话，就被称为“网红点”。大家来网红点游玩，俗称“打卡”。在各个网红点打卡的快（省）乐（钱）方法称为“攻略”。你的任务就是从一大堆攻略中，找出那个能在每个网红点打卡仅一次、并且路上花费最少的攻略。

时间限制：1000
内存限制：65536

输入
首先第一行给出两个正整数：网红点的个数 N（1 < N ≤ 200）和网红点之间通路的条数 M。随后 M 行，每行给出有通路的两个网红点、以及这条路上的旅行花费（为正整数），格式为“网红点 1 网红点 2 费用”，其中网红点从 1 到 N 编号；同时也给出你家到某些网红点的花费，格式相同，其中你家的编号固定为 0。 再下一行给出一个正整数 K，是待检验的攻略的数量。随后 K 行，每行给出一条待检攻略，格式为： n V_1 V_2 … V_n 其中 n (≤ 200) 是攻略中的网红点数，V_i 是路径上的网红点编号。这里假设你从家里出发，从 V_1 开始打卡，最后从 V_n 回家。

输出
在第一行输出满足要求的攻略的个数。 在第二行中，首先输出那个能在每个网红点打卡仅一次、并且路上花费最少的攻略的序号（从 1 开始），然后输出这个攻略的总路费，其间以一个空格分隔。如果这样的攻略不唯一，则输出序号最小的那个。 题目保证至少存在一个有效攻略，并且总路费不超过 10^9。

样例输入

```
6 13
0 5 2
6 2 2
```

```
6 0 1
3 4 2
1 5 2
2 5 1
3 1 1
4 1 2
1 6 1
6 3 2
1 2 1
4 5 3
2 0 2
7
6 5 1 4 3 6 2
6 5 2 1 6 3 4
8 6 2 1 6 3 4 5 2
3 2 1 5
6 6 1 3 4 5 2
7 6 2 1 3 4 5 2
6 5 2 1 4 3 6
```

样例输出

```
3
5 11
```

提示

样例说明： 第 2、3、4、6 条都不满足攻略的基本要求，即不能做到从家里出发，在每个网红点打卡仅一次，且能回到家里。所以满足条件的攻略有 3 条。 第 1 条攻略的总路费是：(0->5) 2 + (5->1) 2 + (1->4) 2 + (4->3) 2 + (3->6) 2 + (6->2) 2 + (2->0) 2 = 14； 第 5 条攻略的总路费同理可算得：1 + 1 + 1 + 2 + 3 + 1 + 2 = 11，是一条更省钱的攻略； 第 7 条攻略的总路费同理可算得：2 + 1 + 1 + 2 + 2 + 2 + 1 = 11，与第 5 条花费相同，但序号较大，所以不输出。

参考代码:

```
#include <iostream>
#include <vector>
#include <climits>
using namespace std;
```

```
const int INF = 1e9;

int main() {
    int N, M;
    cin >> N >> M;
    vector<vector<int>> g(N + 1, vector<int>(N + 1, INF));
    for (int i = 0; i <= N; ++i) g[i][i] = 0;
    for (int i = 0; i < M; ++i) {
        int u, v, w;
        cin >> u >> v >> w;
        g[u][v] = g[v][u] = w;
    }
    for (int k = 0; k <= N; ++k)
        for (int i = 0; i <= N; ++i)
            for (int j = 0; j <= N; ++j)
                g[i][j] = min(g[i][j], g[i][k] + g[k][j]);
    int K;
    cin >> K;
    int cnt = 0, best = -1, minCost = INF;
    for (int i = 1; i <= K; ++i) {
        int n;
        cin >> n;
        vector<int> p(n);
        for (int &x : p) cin >> x;
        bool ok = true;
        int cost = 0;
        for (int j = 1; j < n; ++j) if (g[p[j - 1]][p[j]] == INF) { ok = false; break; } else cost +=
g[p[j - 1]][p[j]];
        if (!ok || g[0][p[0]] == INF || g[p[n - 1]][0] == INF) continue;
        cost += g[0][p[0]] + g[p[n - 1]][0];
        vector<bool> vis(N + 1, false);
        for (int x : p) if (vis[x]) { ok = false; break; } else vis[x] = true;
        if (!ok) continue;
        cnt++;
        if (cost < minCost) minCost = cost, best = i;
    }
    cout << cnt << endl << best << " " << minCost << endl;
}
```

3 树的偏斜度

对于一棵二叉树，令 n_L 表示仅有左孩子的结点的个数，令 n_R 表示仅有右孩子的结点的个数。这棵树的“偏斜度”定义为 $D_S = n_L - n_R$。本题就请你计算任一棵给定二叉树的 D_S。

时间限制：1000
内存限制：65536

输入
输入在第一行给出正整数 n (≤ 10^3)，为二叉树中结点个数。随后两行先后给出这棵树的后序遍历和中序遍历序列，键值为 1 到 n 的整数。同行数字间以空格分隔。

输出
在一行中按以下格式输出树的偏斜度： $D_S = n_L - n_R$

样例输入

```
7
1 2 7 5 4 3 6
1 2 3 4 7 5 6
```

样例输出

```
2 = 3 - 1
```

参考代码:

```
#include <iostream>
#include <vector>
using namespace std;

pair<int, int> calcSkew(vector<int>& post, vector<int>& in, int ps, int pe, int is, int ie) {
    if (ps > pe) return {0, 0};
    int root = post[pe], idx = is;
    while (in[idx] != root) idx++;
    pair<int, int> left = calcSkew(post, in, ps, ps + idx - is - 1, is, idx - 1);
    pair<int, int> right = calcSkew(post, in, ps + idx - is, pe - 1, idx + 1, ie);
    return {left.first  + right.first  + (idx > is && idx == ie), left.second  + right.second  + (idx < ie
&& idx == is)};
}

int main() {
    int n;
```

```
    cin >> n;
    vector<int> post(n), in(n);
    for (int &x : post) cin >> x;
    for (int &x : in) cin >> x;
    pair<int, int> res = calcSkew(post, in, 0, n - 1, 0, n - 1);
    cout << res.first  - res.second  << " = " << res.first  << " - " << res.second  << endl;
}
```

4 是不是堆

二叉堆可以用一棵完全二叉树来实现，但完全二叉树不一定满足堆的性质。本题就请你判断一棵给定的完全二叉树是不是堆。

时间限制：1000
内存限制：65536

输入
输入在一行中给出两个正整数：m（≤ 100）是将要测试的完全二叉树的数量；n（1 < n ≤ 1000）是完全二叉树中的结点数。 随后 m 行，每行给出 n 个互不相同的键值（均在整型范围内），为完全二叉树的层序遍历序列。

输出
对输入的每棵完全二叉树，如果它是最大堆（大顶堆），就在一行中输出 `Max Heap`；如果是最小堆（小顶堆），则输出 `Min Heap`；如果根本不是堆，则输出 `Not Heap`。然后在下一行输出这棵树的后序遍历序列。同行数字间以 1 个空格分隔，行首尾不得有多余空格。

样例输入

```
3 8
98 72 86 60 65 12 23 50
8 38 25 58 52 82 70 60
10 28 15 12 34 9 8 56
```

样例输出

```
Max Heap
50 60 65 72 12 23 86 98
Min Heap
60 58 52 38 82 70 25 8
```

```
Not Heap
56 12 34 28 9 8 15 10
```

参考代码:

```
#include <iostream>
#include <vector>
using namespace std;

string checkHeap(const vector<int>& tree) {
    int n = tree.size();
    for (int i = 0; i < n; ++i) {
        int l = 2 * i + 1, r = 2 * i + 2;
        if (l < n && tree[i] > tree[l]) return "Min Heap";
        if (r < n && tree[i] > tree[r]) return "Min Heap";
        if (l < n && tree[i] < tree[l]) return "Max Heap";
        if (r < n && tree[i] < tree[r]) return "Max Heap";
    }
    return "Not Heap";
}

void printPostOrder(const vector<int>& tree, int i, vector<int>& res) {
    if (i >= tree.size())  return;
    printPostOrder(tree, 2 * i + 1, res);
    printPostOrder(tree, 2 * i + 2, res);
    res.push_back(tree[i]);
}

int main() {
    int m, n;
    cin >> m >> n;
    while (m--) {
        vector<int> tree(n);
        for (int &x : tree) cin >> x;
        cout << checkHeap(tree) << endl;
        vector<int> res;
        printPostOrder(tree, 0, res);
        for (int i = 0; i < res.size();  ++i) cout << res[i] << (i == res.size()  - 1 ? '\n' : ' ');
    }
}
```