简介：

2020/3/30 下午2:30-6:00

给妹妹们进行了一次快乐的打码训练

难度对标NOIp普及组，~~对标优秀初中生~~

大胆猜一下有木有一等奖的水平?

下面是题目的分析和题解：

T1：

名称：Gold

来源：NOIp 普及组 2015 T1

难度：入门

知识点：模拟

做法：

100分：模拟嘛，根据题意编写就行啦

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

int main() {
    int i,j,n,sum=0,c=0;
    cin >> n;
    for (i=1; c<n; i++) {
        for (j=1; j<=i; j++) {
            sum += i;
            c++;
            if (c == n) break;
        }
    }
    cout << sum << endl;
}
```

T2：

名称：Minions Have Spawned

来源：普及组模拟题

难度：普及-

知识点：排序、搜索

做法：

40分：根据数据范围提示，40%数据m=0；没有英雄组合带来的附加值。因此从大到小排个序，选取前六个英雄即可

100分：因为n,m<30，因此dfs暴力搜索所有的情况，判断出最大值也是可行滴

时间复杂度：

$$O(n^6)$$

```
#include <iostream>
#include <algorithm>
#include <map>
#include <set>
using namespace std;

map<string,int> ma; // 给每一个英雄匹配一个编号

struct node{ // 保存英雄和能力值
    int s,v;
}a[32];

struct node2{ // 保存英雄组合加成
    int a,b,c;
}b[32];

int n,m,ans;
set<int> v;

void dfs(int k,int c,int s) {
    if (c == 6) { // 凑齐了6个英雄
        for (int i=0; i<m; i++)
            if(v.count(b[i].a) and v.count(b[i].b)) // 判断英雄里面有没有加分组合
                s += b[i].c;
        ans = max(ans,s);
        return;
    }
    for (int i = k+1; i<n; i++) {
        v.insert(i);
        dfs(i,c+1,s+a[i].v);
        v.erase(i); // 回溯
    }
}

int main() {
    int i,j;
    string s,s2;
    cin >> n >> m;
    for (i=0; i<n; i++) {
        cin >> s >> a[i].v;
        ma[s] = i;
        a[i].s = i;
    }
    for (i=0; i<m; i++) {
        cin >> s >> s2 >> b[i].c;
        b[i].a = ma[s];
        b[i].b = ma[s2];
    }
    dfs(-1,0,0); // 提问：这里第一个参数为什么是-1呢
    cout << ans << endl;
}
```

T3：

名称：Symmetric Binary Tree

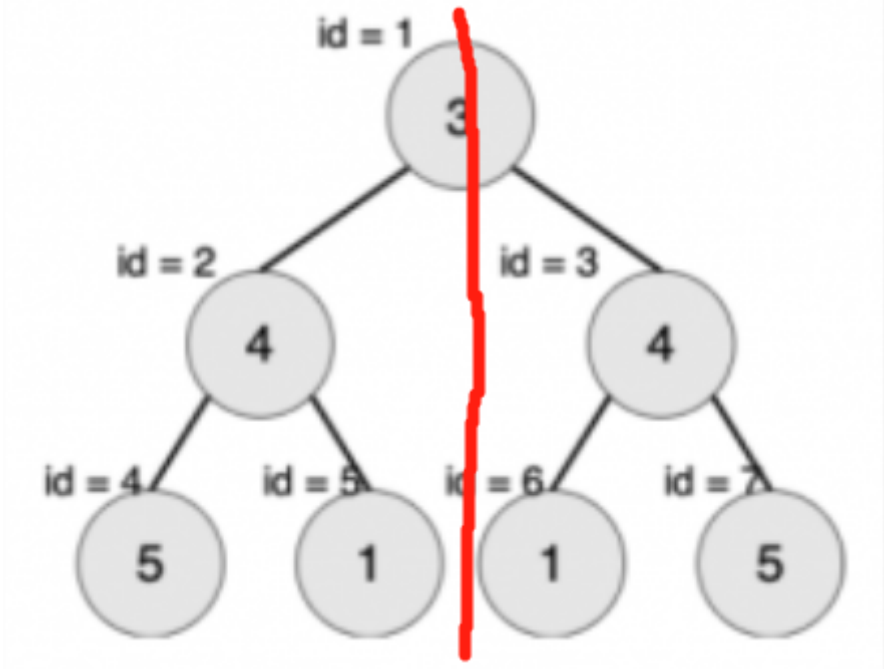
来源：NOIp 普及组 2018 T4

难度：普及+/提高

知识点：树形结构、递归

做法：

判断对称二叉树：由题目的定义可知，对称二叉树沿中心线对称



因此可以考虑用树的递归判断：

如果根节点的左右节点存在且相等，就判断

1、根左节点的左节点和根右节点的右节点是否存在且相等

2、根左节点的右节点和根右节点的左节点是否存在且相等

之后递归上述过程，直到判断出是不是对称二叉树

查找最大对称二叉树：遍历每一个子节点，通过上述递归判断是不是对称二叉树，是的话递归算出该树的节点数量，取最大值输出即可

时间复杂度：

$$O(nlogn)$$

混分：

测试点1~3：根节点的左子树所有节点没有右孩子，右子树所有节点没有左孩子，因此递归判断的过程可以没有"判断2"

测试点9~12：输入是满二叉树，递归判断可以没有"判断一边存在子节点一边不存在"的情况

测试点17~20：所有点的点权相等，递归判断可以没有"判断点权是否相等"

测试点21~25：因为数据量比较大，nlogn的算法比较极限，可以优化程序卡一下常数、记忆化搜索优化计算该树的节点数量的递归，~~当然也可以开O2吸氧过~~

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

int n,c,sum;
int a[1000005],l[1000005],r[1000005]; // a:记录点权,l,r:记录左右子节点

int cou(int k) { // 统计以k为节点的子树的节点数量
    int num = 0;
    if (l[k] != -1) num += cou(l[k]); // 递归统计左子树节点数量
    if (r[k] != -1) num += cou(r[k]); // 递归统计右子树节点数量
    return num+1; // 总数=左+右+根节点
}

void dfs(int p,int q) { // 判断是不是对称二叉树
    if(p == -1 and q == -1) return;
    if(p == -1 or q == -1 or a[p] != a[q]) { // 如果一边存在子节点一边不存在，或者点权不相等
        c = 0; // 标识符:不是对称二叉树
        return;
    }
    dfs(l[p],r[q]); // 递归根左节点的左节点和根右节点的右节点
    dfs(r[p],l[q]); // 递归根左节点的右节点和根右节点的左节点
}

int main() {
    int i,j;
    cin >> n;
    for (i=1; i<=n; i++) cin >> a[i];
    for (i=1; i<=n; i++) cin >> l[i] >> r[i];
    for (i=1; i<=n; i++) {
        c = 1;
        dfs(i,i);
        if (c == 1) sum = max(sum,cou(i)); // 如果是对称二叉树，则统计节点数，取最大值
    }
    cout << sum << endl;
}
```

T4:

名称：Program Automatic Analysis

来源：NOI 2015 Day1 T1

难度：提高+/省选-

知识点：并查集、离散化

做法：

看了一遍题目其实有比较明显的并查集意思

输入x1=x2其实就是合并操作，并查集把x1和x2合并成一组就可以啦

输入x1!=x2其实就是查询操作，只需要判断x1和x2是否在同一组就可以啦

不在同一组就没关系，在同一组则产生矛盾

当然，所有的判断应该在所有的合并之后

如果先判断不在同一组而后面又合并成一组，就会产生矛盾

所以需要先排序一下，合并操作放前面

这样子应该能拿下70分

而对于后面的三个测试点，普通的并查集需要声明10^9大小的数组，显然会RE或者MLE

因为n只有10^5，因此用map代替掉普通的并查集数组即可实现离散化

最后想拿100分的话，还需要尽可能优化时间避免TLE

第一个是使用unordered_map代替map（需要C++11）

前者本质是哈希表，后者本质是红黑树

哈希表的查询能达到O(1)级别，比红黑树O(logn)要快

第二个就是输入输出优化

scanf和printf比cin、cout要快，更快的话可以手写快读

~~第三个就是开吸氧优化啦~~

```
#include <iostream>
#include <algorithm>
#include <unordered_map>
#include <cstdio>
using namespace std;

unordered_map<int, int> m; // map代替并查集数组

int find(int k) { // 并查集标配
    return m[k] == k ? k : m[k] = find(m[k]);
}

struct node { // 记录合并和查询操作
    int a,b,l; // a,b是两端,l=1为合并,l=0为查询
};

int cmp(node a,node b) {
    return a.l>b.l; // 结构体排序,合并操作排前面
}

int main() {
    int i,j,n,t;
    scanf("%d",&t);
    while (t--) {
        m.clear(); // 先清零，排除前面测试样例的影响
        j = 0; // 判断位
        scanf("%d",&n);
        node s[n];
        for (i=0; i<n; i++) scanf("%d%d%d",&s[i].a,&s[i].b,&s[i].l);
        sort(s,s+n,cmp); // 排序
        for (i=0; i<n; i++) {
            if (s[i].l == 1) {
                if (m.count(s[i].a) == 0) m[s[i].a] = s[i].a; // 声明并查集数组
                if (m.count(s[i].b) == 0) m[s[i].b] = s[i].b;
                m[find(s[i].a)] = find(s[i].b); // 合并
```

```
            }
            else {
                if (m.count(s[i].a) == 0) m[s[i].a] = s[i].a;
                if (m.count(s[i].b) == 0) m[s[i].b] = s[i].b;
                if (find(s[i].a) == find(s[i].b)) { // 查询
                    j = 1;
                    break;
                }
            }
        }
        cout << (j ? "NO" : "YES") << endl;
    }
}
```

题目PDF：快乐的打码[3.30]