# 1 pA

照著題目講的做

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
void die(){
    cout << "TLE" << endl;
    exit(0);
}
void mpow(ll x, ll y, ll m){
    ll now = 1;
    for(int i = 0 ; i < y ; i++){
        now *= x;
        if(now > m)
            die();
    }
}
int main(){
    ll m, n, t, now = 1;
    cin >> m >> n >> t;
    if(t == 1){
        for(ll i = 1 ; i <= n ; i++){
            now *= i;
            if(now > m)
                die();
        }
    } else if(t == 2){
        mpow(2, n, m);
    } else if(t == 3){
        mpow(n, 4, m);
    } else if(t == 4){
        mpow(n, 3, m);
    } else if(t == 5){
        mpow(n, 2, m);
    } else if(t == 6){
        if(double(n) * double(log2(n)) > m)
            die();
    } else if(t == 7){
        if(n > m)
            die();
    }
    cout << "AC" << endl;
}
```

# 2 pB

經典小遊戲 2048，純粹模擬提。這邊的實作了一個方向的旋轉，將所有數字移到最左邊，跟合成數字。

```cpp
#include <bits/stdc++.h>
using namespace std;
int t[4][4], d;
void move(){
    for(int k = 0 ; k < 4 ; k++){
        for(int i = 0 ; i < 4 ; i++){
            for(int j = 0 ; j < 3 ; j++){
                if(!t[i][j])
                    swap(t[i][j], t[i][j+1]);
            }
        }
    }
}
void add(){
    for(int i = 0 ; i < 4 ; i++){
        for(int j = 0 ; j < 3 ; j++){
            if(t[i][j] == t[i][j+1]){
                t[i][j] *= 2;
                t[i][j+1] = 0;
            }
        }
    }
}
void trans(int x){
    for(int k = 0 ; k < x ; k++){
        for(int i = 0 ; i < 4 ; i++){
            for(int j = 0 ; j < i ; j++)
                swap(t[i][j], t[j][i]);
        }
        for(int i = 0 ; i < 2 ; i++){
            for(int j = 0 ; j < 4 ; j++)
                swap(t[i][j], t[3-i][j]);
        }
    }
}

int main(){
    for(int i = 0 ; i < 4 ; i++)
        for(int j = 0 ; j < 4 ; j++)
            cin >> t[i][j];
    cin >> d;
    trans(d);
    move();
    add();
    move();
    trans(4-d);
    for(int i = 0 ; i < 4 ; i++){
        for(int j = 0 ; j < 4 ; j++)
            cout << t[i][j] << ' ';
        cout << endl;
    }
    return 0;
}
```

# 3 pC

給定一張有向圖，問從 s->t 的最短路徑有幾條。

採用 dijkstra 來紀錄到點 x 的最短距離，以及方法數。如果發現有更好的方法能到就將方法歸零。如果當前距離恰為最短距離，則加上來源點的方法數。

```cpp
#include <bits/stdc++.h>
using namespace std;
#define f first
#define s second
typedef pair<int, int> PII;
#define MAX 16384
vector<PII> ed[MAX];
PII dis[MAX];

int main(){
    ios_base::sync_with_stdio(0), cin.tie(0);
    fill(dis, dis+MAX, PII(1029384756, 0));
    int n, m;
    cin >> n >> m;
    for(int i = 0 ; i < m ; i++){
        int a, b, c;;
        cin >> a >> b >> c;
        ed[a].push_back(PII(b, c));
    }
    int start, end;
    cin >> start >> end;
    dis[start] = PII(0, 1);
    priority_queue<PII, vector<PII>, greater<PII> > PQ;
    PQ.push(PII(dis[start].f, start));
    while(PQ.size()){
        PII tmp = PQ.top(); PQ.pop();
        if(tmp.f > dis[end].f) continue;
        if(tmp.f > dis[tmp.s].f) continue;
        for(int i = 0 ; i < (int)ed[tmp.s].size() ; i++){
            int nx = ed[tmp.s][i].f;
            int nd = dis[tmp.s].f + ed[tmp.s][i].s;
            if(nd < dis[nx].f){
                dis[nx].f = nd;
                dis[nx].s = 0;
                PQ.push(PII(nd, nx));
            }
            if(nd == dis[nx].f)
                dis[nx].s += dis[tmp.s].s;
        }
    }
    cout << dis[end].s << endl;
    return 0;
}
```

# 4 pD

給定二維平面上的一堆整數點，問這些整數點的最小生成樹。
從給定的所有點做 BFS，如果碰到其他點則建一條邊，最後使用 Kruskal 求解。

```
#include <bits/stdc++.h>
using namespace std;
#define f first
#define s second
#define MAX 1024
typedef pair<int, int> PII;
struct ED{
    int a, b, c;
    ED(){}
    ED(int _a, int _b, int _c) : a(_a), b(_b), c(_c) {}
    bool operator<(const ED &E) const {
        return c < E.c;
    }
};
namespace DS{
    int v[131072];
    void init(){
        for(int i = 0 ; i < 131072 ; i++)
            v[i] = i;
    }
    int find(int x){
        return x == v[x] ? v[x] : v[x] = find(v[x]);
    }
    bool merge(int x, int y){
        x = find(x), y = find(y);
        if(x == y)
            return false;
        v[x] = y;
        return true;
    }
};
int t[MAX][MAX];
int d[4][2] = {{1, 0}, {-1, 0}, {0, 1}, {0, -1}};
int main(){
    memset(t, -1, sizeof(t));
    int n;
    scanf("%d", &n);
    vector<PII> p(n);
    vector<ED> ed;
    queue<PII> Q;
    for(int i = 0 ; i < n ; i++){
        scanf("%d%d", &p[i].f, &p[i].s);
        Q.push(PII(p[i].f, p[i].s));
        t[p[i].f][p[i].s] = i;
    }
    while(Q.size()){
        PII now = Q.front(); Q.pop();
        for(int i = 0 ; i < 4 ; i++){
            int nx = now.f + d[i][0];
            int ny = now.s + d[i][1];
            if(nx < 0 || ny < 0 || nx >= MAX || ny >= MAX) continue;
            if(t[nx][ny] == -1){
                t[nx][ny] = t[now.f][now.s];
                Q.push(PII(nx, ny));
            } else {
                int u = t[now.f][now.s];
```

```
57                  int v = t[nx][ny];
58                  int dis = abs(p[u].f - p[v].f) + abs(p[u].s - p[v].s);
59                  ed.push_back(ED(u, v, dis));
60              }
61          }
62      }
63      sort(ed.begin(), ed.end());
64      long long ans = 0;
65      DS::init();
66      for(int i = 0 ; i < (int)ed.size() ; i++){
67          if(DS::merge(ed[i].a, ed[i].b))
68              ans += ed[i].c;
69      }
70      printf("%lld\n", ans);
71      return 0;
72  }
```

# 5 pE

給定二維平面上多個矩形的值，問最大值為多少及其面積。
將所有座標離散化，搭配線段數求解。

```cpp
#include <bits/stdc++.h>
using namespace std;
#define MAX 131072
#define SMAX 262144
typedef long long ll;

struct S{
    int x, sy, ey, v;
    S(){}
    S(int _x, int _sy, int _ey, int _v):
        x(_x), sy(_sy), ey(_ey), v(_v){}
    bool operator<(const S &b) const {
        return x < b.x;
    }
};
vector<int> x, y;
namespace Seg{
    struct Node{
        int v, maxv, sz;
    };
    Node v[SMAX * 2];

    void init(){
        for(int i = 2 * SMAX - 1 ; i >= SMAX ; i--){
            v[i].v = v[i].maxv = 0;
            v[i].sz = 0;
        }
        for(int i = 1 ; i < (int)y.size() ; i++){
            v[i+SMAX-1].sz = y[i] - y[i-1];
        }
        for(int i = SMAX - 1 ; i > 0 ; i--){
            v[i].v = v[i].maxv = 0;
            v[i].sz = v[2*i].sz + v[2*i+1].sz;
        }
    }
    void update(int now, int l, int r, int ql, int qr, int add){
        if(r <= ql || qr <= l) return;
        if(ql <= l && r <= qr){
            v[now].v += add;
            v[now].maxv += add;
            return;
        }
        int mid = (l+r) / 2;
        update(now*2, l, mid, ql, qr, add);
        update(now*2+1, mid, r, ql, qr, add);
        v[now].maxv = max(v[now*2].maxv, v[now*2+1].maxv) + v[now].v;
        if(v[now*2].maxv == v[now*2+1].maxv)
            v[now].sz = v[now*2].sz + v[now*2+1].sz;
        else if(v[now*2].maxv > v[now*2+1].maxv)
            v[now].sz = v[now*2].sz;
        else
            v[now].sz = v[now*2+1].sz;
    }
};
void Solve(){
    x.clear(), y.clear();
```

```
57      int n;
58      scanf("%d", &n);
59      int sx, sy, ex, ey, v;
60      vector<S> s;
61      for(int i = 0 ; i < n ; i++){
62          scanf("%d%d%d%d%d", &sx, &sy, &ex, &ey, &v);
63          x.push_back(sx), x.push_back(ex);
64          y.push_back(sy), y.push_back(ey);
65          s.push_back(S(sx, sy, ey, v));
66          s.push_back(S(ex, sy, ey, -v));
67      }
68      sort(x.begin(), x.end());
69      sort(y.begin(), y.end());
70      x.resize(unique(x.begin(), x.end()) - x.begin());
71      y.resize(unique(y.begin(), y.end()) - y.begin());
72      Seg::init();
73      sort(s.begin(), s.end());
74      ll maxv = 0, maxsz = 0;
75      for(int i = 0 ; i < (int)s.size() ; i++){
76          if(i){
77              if(s[i].x != s[i-1].x){
78                  if(maxv < Seg::v[1].maxv){
79                      maxv = Seg::v[1].maxv;
80                      maxsz = 0;
81                  }
82                  if(maxv == Seg::v[1].maxv)
83                      maxsz += ll(Seg::v[1].sz) * ll(s[i].x - s[i-1].x);
84              }
85          }
86          sy = lower_bound(y.begin(), y.end(), s[i].sy) - y.begin();
87          ey = lower_bound(y.begin(), y.end(), s[i].ey) - y.begin();
88          Seg::update(1, 0, SMAX, sy, ey, s[i].v);
89      }
90      printf("%lld %lld\n", maxv, maxsz);
91  }
92
93  int main(){
94      int t;
95      scanf("%d", &t);
96      while(t--)
97          Solve();
98      return 0;
99  }
```

# 6 pH

給定一場比賽勝負關係以及有誤的排名保證正確的排名唯一問至少要修改幾次才會改對
解題概念：Topological sort
做出排名後用 LIS 求出最長的部份 2 * n - LIS.size() 即是答案

```cpp
#include <bits/stdc++.h>
using namespace std;
#define f first
#define s second
typedef long long ll;
typedef pair<ll, ll> PLL;
map<ll, ll> deg;
map<ll, vector<ll> > ed;
int main(){
    ios_base::sync_with_stdio(0), cin.tie(0);
    ll n, m, d;
    cin >> n >> m >> d;
    for(int i = 0 ; i < m ; i++){
        ll a, b;
        cin >> a >> b;
        deg[b]++;
        deg[a] = deg[a];
        ed[a].push_back(b);
    }
    vector<PLL> rank(n);
    map<ll, ll> rrank;
    for(int i = 0 ; i < n ; i++)
        cin >> rank[i].f;
    queue<ll> Q;
    for(map<ll, ll>::iterator x = deg.begin() ; x != deg.end() ; x++){
        if( !(*x).second )
            Q.push((*x).first);
    }
    while(Q.size()){
        rrank[Q.front()] = rrank.size();
        ll x = Q.front(); Q.pop();
        vector<ll> &E = ed[x];
        for(int i = 0 ; i < (int)E.size() ; i++){
            deg[E[i]]--;
            if(!deg[E[i]])
                Q.push(E[i]);
        }
    }
    for(int i = 0 ; i < (int)rank.size() ; i++){
        rank[i].s = rrank[rank[i].f];
    }
    vector<ll> LIS;
    int start = 0;
    for( ; start < (int)rank.size() ; start++){
        if(rank[start].s){
            break;
        }
    }
    if(start == (int)rank.size()){
        cout << 2 * (int)rank.size() << endl;
        return 0;
    }
    LIS.push_back(rank[start].s);
    for(int i = start + 1 ; i < (int)rank.size() ; i++){
        if(rank[i].s == 0) continue;
```

```
56          if(rank[i].s > LIS.back())
57              LIS.push_back(rank[i].s);
58          else
59              *lower_bound(LIS.begin(), LIS.end(), rank[i].s) = rank[i].s;
60      }
61      int ans = (rank.size() - LIS.size()) * 2;
62      cout << ans << endl;
63
64      return 0;
65  }
```

# 7 pI

給定一數列，問有多少數符合：大於左邊所有數且小於右邊所有數。

解題概念：從左而右做 max，從右而左做 min，l[i-1] < v[i] && v[i] < r[i+1] 即為要求的數。

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

int main(){
    ios_base::sync_with_stdio(0), cin.tie(0);
    int n;
    cin >> n;
    vector<int> v(n);
    for(int i = 0 ; i < n ; i++)
        cin >> v[i];
    vector<int> l(n), r(n);
    l[0] = v[0];
    r[n-1] = v[n-1];
    for(int i = 1 ; i < n ; i++)
        l[i] = max(v[i], l[i-1]);
    for(int i = n - 2 ; i >= 0 ; i--)
        r[i] = min(v[i], r[i+1]);
    int ans = 0;
    for(int i = 1 ; i < n - 1 ; i++)
        if(l[i-1] < v[i] && v[i] < r[i+1])
            ans++;
    if(v[0] < r[1])
        ans++;
    if(l[n-2] < v[n-1])
        ans++;
    cout << ans << endl;

    return 0;
}
```

# 8 pJ

給定兩棵二元樹，問有多少子樹的組成元素相同

解題概念：做 hash 然後 count 數量，hash 部份這邊採用一般的加法，以及相乘並且 mod 一質數，並且每個值也事先做平移。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int, int> PII;
struct S{
    ll s[2];
    S(){}
    S(int a, int b){
        s[0] = a;
        s[1] = b;
    }
    S& operator=(const S& b){
        for(int i = 0 ; i < 3 ; i++)
            s[i] = b.s[i];
        return (*this);
    }
    S operator+(const S& b) const {
        S re;
        re.s[0] = s[0] + b.s[0];
        re.s[1] = (s[1] * b.s[1]) % 1000000007;
        return re;
    }
};
int _hash(int x){
    return (x + 10007) % 100019;
}
set<PII> s;
string str[2];
int ans;
int p;
S build(int t){
    S re;
    if(str[t][p] == '('){
        ++p;
        S x = build(t);
        ++p;
        S y = build(t);
        re = x + y;
        ++p;
    }
    if(isdigit(str[t][p])){
        int x = 0;
        while(isdigit(str[t][p])){
            x = x * 10 + str[t][p++] - '0';
        }
        x = _hash(x);
        re = S(x, x);
    }
    if(!t){
        s.insert(PII(re.s[0], re.s[1]));
    } else {
        ans += s.count(PII(re.s[0], re.s[1]));
    }
    return re;
}
```

```
56 int main(){
57     int n;
58     cin >> n;
59     cin >> str[0] >> str[1];
60     build(0);
61     p = 0;
62     build(1);
63     cout << ans << endl;
64 }
```