1 pA

給定 n 個數字以及一差距 k 任兩數差距在 k 之內可建一條邊問總共有幾群
策略: sort 之後若和前一個數字超過 k 則 ans++

```cpp
#include <bits/stdc++.h>
using namespace std;
int Solve(){
    int n, m;
    scanf("%d%d", &n, &m);
    vector<int> vec(n);
    for(int i = 0 ; i < n ; i++)
        scanf("%d", &vec[i]);
    sort(vec.begin(), vec.end());
    int ans = 1;
    for(int i = 1 ; i < n ; i++)
        if(vec[i] - vec[i-1] > m)
            ans++;
    return ans;
}
int main(){
    int n;
    scanf("%d", &n);
    for(int i = 1 ; i <= n ; i++){
        printf("Case #%d: %d\n", i, Solve());
    }
    return 0;
}
```

2 pB

給定一張圖問圖上有多少個兩點數相同之完全圖有一條邊連起來

策略: 先做一次 Disjoint Set 將點分群，若群中點的個數為偶數且只有兩個點度數是 n(兩完全圖相連點) 則此群可能為要求在跑一次 Disjoint Set 但這次忽略兩完全圖的相連邊若 Disjoint Set 的群數為 2 則 ans++

```
#include <bits/stdc++.h>
using namespace std;
struct DS{
    int v[128], n;
    void init(int _n=128){
        for(int i = 0 ; i < 128 ; i++)
            v[i] = i;
        n = _n;
    }
    int find(int x){
        return x == v[x] ? v[x] : v[x] = find(v[x]);
    }
    void merge(int x, int y){
        x = find(x), y = find(y);
        if(x != y){
            v[x] = y;
            n--;
        }
    }
};

int Solve(){
    int ans = 0;
    int n, m;
    scanf("%d%d", &n, &m);
    DS ds;
    ds.init();
    bool go[128] = {};
    vector<int> ed[128];
    for(int i = 0 ; i < m ; i++){
        int x, y;
        scanf("%d%d", &x, &y);
        ed[x].push_back(y);
        ed[y].push_back(x);
        ds.merge(x, y);
    }
    for(int i = 1 ; i <= n ; i++){
        int dsn = ds.find(i);
        vector<int> p;
        if(!go[dsn]){
            go[dsn] = true;
            for(int j = 1 ; j <= n ; j++){
                if(dsn == ds.find(j))
                    p.push_back(j);
            }
            if(p.size() & 1)
                continue;
            vector<int> d;
            bool flag = true;
            for(int j = 0 ; j < (int)p.size() ; j++){
                if(ed[p[j]].size() == p.size() / 2)
                    d.push_back(p[j]);
                else if(ed[p[j]].size() == (p.size() / 2) - 1){
                } else {
```

```
55                      flag = false;
56                  }
57              }
58              if(!flag || d.size() != 2)
59                  continue;
60
61              DS ds2;
62              ds2.init(p.size());
63              for(int j = 0 ; j < (int)p.size() ; j++){
64                  int u = p[j];
65                  for(int k = 0 ; k < (int)ed[u].size() ; k++){
66                      int v = ed[u][k];
67                      if((u == d[0] && v == d[1]) || (u==d[1] && v==d[0]))
68                          continue;
69                      ds2.merge(u, v);
70                  }
71              }
72              if(ds2.n == 2)
73                  ans++;
74          }
75      }
76      return ans;
77  }
78
79  int main(){
80      int n;
81      scanf("%d", &n);
82      for(int i = 1 ; i <= n ; i++){
83          printf("Case #%d: %d\n", i, Solve());
84      }
85      return 0;
86  }
```

3 pD

一間公司要徵才總共有 n 個人去，而你是第 p 個人徵才的規則是直接拒絕前 k 個人，之後遇到第一個比前一個大的就上了問你你有多少種組合能上

策略: DP，保持遞增，紀錄 [pos][rank] 時有多少種組合，然後用排列組合計算即可得到答案。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
#define MOD 1000000007
#define MAX 512
ll dp[MAX][MAX];
ll f[MAX];
int Solve(){
    memset(dp, 0, sizeof(dp));
    ll ans = 0, now;
    int n, k, s;
    scanf("%d%d%d", &n, &k, &s);
    for(int i = 1 ; i <= n ; i++)
        dp[k][i] = i != s;
    for(int i = k + 1 ; i < n ; i++){
        now = 0;
        for(int j = 1 ; j <= n ; j++){
            dp[i][j] = 0;
            if( j == s ) continue;
            if( j == s + 1 ){
                if(j > 2)
                    dp[i][j] = dp[i][j-2] + dp[i-1][j-2];
            } else {
                if(j > 1)
                    dp[i][j] = dp[i][j-1] + dp[i-1][j-1];
            }
            dp[i][j] %= MOD;
            if(j > s)
                now = (now+dp[i-1][j]) % MOD;
        }
        ans = (ans + (now*f[n-i+k-1]) % MOD) % MOD;
    }
    now = 0;
    for(int i = 1 ; i <= n ; i++){
        if(i == s) continue;
        now = (now+dp[n-1][i]) % MOD;
    }
    ans = ans + ((now*f[k-1])%MOD)%MOD;
    return ans % MOD;
}
int main(){
    f[0]=1;
    for(int i = 1 ; i < MAX ; i++)
        f[i] = (f[i-1] * i) % MOD;
    int n;
    scanf("%d", &n);
    for(int i = 1 ; i <= n ; i++){
        printf("Case #%d: %d\n", i, Solve());
    }
    return 0;
}
```

4 pE

給你一棵樹，接下來有兩種操作 1) 砍掉一條邊 2) 詢問兩點是否連通

策略: 使用 Disjoint Set，一開始先把沒被砍的邊都處理好，接著把操作反著做。測資中有一條邊不只被砍一次，所以要等到真的是第一次被砍才把邊建回去。

```
#include <bits/stdc++.h>
using namespace std;
namespace DS{
    int v[32768];
    void init(){
        for(int i = 0 ; i < 32768 ; i++)
            v[i] = i;
    }
    int find(int x){
        return x == v[x] ? v[x] : v[x] = find(v[x]);
    }
    void merge(int x, int y){
        if(x == 0 || y == 0){
            return;
        }
        v[find(x)] = find(y);
    }
};
struct Q{
    int t;
    int x, y;
    Q(){}
    Q(int _t, int _x, int _y) : t(_t), x(_x), y(_y) {}
};
void Solve(){
    int n, q;
    scanf("%d%d", &n, &q);
    DS::init();
    int v[32768];
    for(int i = 1 ; i <= n ; i++)
        scanf("%d", &v[i]);
    vector<Q> query;
    int ok[32768];
    fill(ok, ok+32768, 0);
    char s[2];
    int x, y;
    for(int i = 0 ; i < q ; i++){
        scanf("%s", s);
        if(s[0] == 'Q'){
            scanf("%d%d", &x, &y);
            query.push_back(Q(1, x, y));
        } else {
            scanf("%d", &x);
            ok[x]++;
            query.push_back(Q(0, x, 0));
        }
    }
    for(int i = 1 ; i <= n ; i++){
        if(!ok[i])
            DS::merge(i, v[i]);
    }
    stack<int> ans;
    for(int i = (int)query.size() - 1; i >= 0 ; i--)
        if(query[i].t){
            ans.push(DS::find(query[i].x) == DS::find(query[i].y));
```

```
56          } else {
57              ok[query[i].x]--;
58              if(!ok[query[i].x])
59                  DS::merge(query[i].x, v[query[i].x]);
60          }
61      while(ans.size()){
62          printf("%s\n", ans.top()?"YES":"NO");
63          ans.pop();
64      }
65  }
66
67  int main(){
68      //freopen("pc.in", "r", stdin);
69      int t;
70      scanf("%d", &t);
71      for(int i = 1 ; i <= t ; i++){
72          printf("Case #%d:\n", i);
73          Solve();
74      }
75      return 0;
76  }
```

5 pF

砍龍需要兩把劍問最少要準備幾把第一把固定長度 Ai 第二把長度可以為 Bi Ci

策略: 先把第一把丟入 set 跟 multiset 中答案至少為 set.size() 把接著考慮第二把先將 multiset 減掉自己的第一把查在 multiset 中是否有符合 Bi Ci 的若有則不處理若無則保留起來處理。

```cpp
#include <bits/stdc++.h>
using namespace std;
#define MAX 1048576
#define f first
#define s second
typedef pair<int, int> PII;
int d[MAX][3];
int ok[MAX];

bool cmp(const PII &a, const PII &b){
    if(a.f == b.f) return a.s < b.s;
    return a.f < b.f;
}

int Solve(){
    multiset<int> ms;
    set<int> s;
    int n, ans;
    memset(ok, 0, sizeof(ok));
    scanf("%d", &n);
    for(int i = 0 ; i < n ; i++){
        scanf("%d%d%d", &d[i][0], &d[i][1], &d[i][2]);
        s.insert(d[i][0]);
        ms.insert(d[i][0]);
    }
    vector<PII> p;
    ans = s.size();
    for(int i = 0 ; i < n ; i++){
        ms.erase(d[i][0]);
        multiset<int>::iterator it = ms.lower_bound(d[i][1]);
        if(it != ms.end() && d[i][2] >= *it)
            ok[i] = true;
        ms.insert(d[i][0]);
    }
    for(int i = 0 ; i < n ; i++){
        if(ok[i]) continue;
        p.push_back(PII(d[i][1], d[i][2]));
    }
    sort(p.begin(), p.end(), cmp);
    int right = 0;
    for(int i = 0 ; i < (int)p.size() ; i++){
        right = min(right, p[i].s);
        if(right >= p[i].f) continue;
        ans++;
        right = p[i].s;
    }
    return ans;
}

int main(){
    int n;
    scanf("%d", &n);
    for(int i = 1 ; i <= n ; i++){
        printf("Case #%d: %d\n", i, Solve());
```

```
55      }
56      return 0;
57  }
```

6 pH

有兩隊要吃蛋糕，每隊 n 個人，每個人最多可以吃 k 個，總共 m 個蛋糕吃完蛋糕的那隊獲勝
策略: 先將連續的人壓縮起來，利用 dp[pos][remain] 紀錄化搜索如果接下來對方有一個可能會輸則自己必贏

```
#include <bits/stdc++.h>
using namespace std;
#define MAX 1024
int dp[MAX * 2][MAX * 4];
int s[MAX*2];
int n, m, k;
char str[MAX*2];

bool ok(int p, int x){
    if(dp[p][x] != -1)
        return dp[p][x];
    if(s[p] * k >= x)
        return dp[p][x] = 1;
    int l = x - s[p] * k;
    int r = x - s[p];
    for(int i = l ; i <= r ; i++)
        if(!ok(p+1, i))
            return dp[p][x] = 1;
    return dp[p][x] = 0;
}

char Solve(){
    memset(dp, -1, sizeof(dp));
    memset(s, 0, sizeof(s));
    scanf("%d%d%d", &n, &m, &k);
    scanf("%s", str);
    n *= 2;
    char last = 'C';
    int now = -1;
    for(int i = 0 ; i < n ; i++){
        if(str[i] == last){
            s[now]++;
        } else {
            now++;
            s[now] = 1;
        }
        last = str[i];
    }
    now++;
    bool ans = ok(0, m);
    return ans ? str[0] : ((str[0] - 'A') ^ 1) + 'A';
}

int main(){
    int n;
    scanf("%d", &n);
    for(int i = 1 ; i <= n ; i++)
        printf("Case #%d: %c\n", i, Solve());
    return 0;
}
```