第 1 章 atcode 061

1.1 arc061

1.1.1 c- Many Formulass

dfs, 这样去枚举每一个括号的位置

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
ll ans = 0;
string S;
void dfs(int n, ll v, ll sum)
{
    if (S.length() == n) {
        sum += v;
        ans += sum;
        return;
    }
    ll next = v * 10 + S[n] - '0';
    dfs(n + 1, 0, sum + next);
    dfs(n + 1, next, sum);
}
int main()
{
    cin >> S;
    dfs(0, 0, 0);
    cout << ans / 2 << endl;
    return 0;
}
```

1.1.2 d- Many Formulass

很明显一个点被染色只有可能对周围的点产生影响，所以暴力修改就行了

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<ll, ll> pll;
int main()
{
    ll H, W, N;
    cin >> H >> W >> N;
    vector<ll> a(N), b(N);
    for (int i = 0; i < N; i++)
        cin >> a[i] >> b[i];
    map<pll, ll> mp;
    for (int i = 0; i < N; i++) {
        for (int j = -2; j <= 0; j++) {
            for (int k = -2; k <= 0; k++) {
                ll nx = a[i] + j, ny = b[i] + k;
                if (nx <= 0 || nx >= H - 1 || ny <= 0 || ny >= W - 1)
                    continue;
                mp[pll(nx, ny)]++;
            }
```

```
        }
    }
    ll ans[10] = { 0 };
    for (auto it = mp.begin(); it != mp.end(); it++) {
        ans[it->second]++;
        //cout<<it->first.first<<" "<<it->first.second<<" "<<it->second<<endl;
    }
    ll sum = (H - 2) * (W - 2);
    for (auto v : ans) {
        sum -= v;
    }
    ans[0] += sum;
    for (auto v : ans) {
        cout << v << endl;
    }
    return 0;
}
```

1.1.3 e- Snuke's Subway Trip

我们把单条边拆掉，拆成三条边，中间放两个虚点，规定虚点之间的费用为 0，实点到虚点的费用为 1，那么就可以直接跑 spfa 了，最后答案除 2 就行（这样子拆边的话一条路径的花费是 2，所以要除一下）

至于虚点的编号可以用 map 来取，为什么不能直接 tot++ 呢？因为如果直接加点的编号，那么对于同个公司的情况就没法求了——你每一次到达的都是一个新点。

```
#include <bits/stdc++.h>
using namespace std;
#define N 5000010
#define inf 0x3f3f3f3f
int head[N], cnt;
int d[N], vis[N], q[N];
int n, m;
struct node {
    int to, nxt, v;
} e[N];
void ins(int u, int v, int w)
{
    e[++cnt].to = v;
    e[cnt].nxt = head[u];
    e[cnt].v = w;
    head[u] = cnt;
}
map<pair<int, int>, int> mp;
int tot = 0;
int get_num(int x, int y)
{
    if (!mp.count(make_pair(x, y)))
        mp[make_pair(x, y)] = ++tot;
    return mp[make_pair(x, y)];
}
void spfa()
{
    for (int i = 1; i <= tot; i++)
        d[i] = inf;
    vis[1] = q[1] = 1;
    d[1] = 0;
```

```
    int l = 1, r = 2;
    while (l < r) {
        int u = q[l++];
        vis[u] = 0;
        for (int i = head[u]; i; i = e[i].nxt) {
            int v = e[i].to;
            if (d[v] > d[u] + e[i].v) {
                d[v] = d[u] + e[i].v;
                if (!vis[v])
                    vis[v] = 1, q[r++] = v;
            }
        }
    }
    if (d[n] == inf)
        puts("-1");
    else
        printf("%d\n", d[n] / 2);
}
int main()
{
    scanf("%d%d", &n, &m);
    tot = n;
    for (int i = 1; i <= m; i++) {
        int x, y, c;
        scanf("%d%d%d", &x, &y, &c);
        int n1 = get_num(x, c), n2 = get_num(y, c);
        ins(x, n1, 1);
        ins(n1, x, 1);
        ins(n1, y, 1);
        ins(y, n1, 1);
        ins(n1, n2, 0);
        ins(n2, n1, 0);
    }
    spfa();
}
```

1.1.4 f- Many Formulass

组合数加逆元求 (留个坑，题解后补)

```
#include <bits/stdc++.h>
using namespace std;
#define N 5010
#define ll long long
const int mod = 1e9 + 7;
int n, m, k;
ll fac[N], ifac[N];
ll power(ll a, ll b)
{
    ll base = a, ans = 1;
    while (b) {
        if (b & 1)
            ans = (ans * base) % mod;
        base = (base * base) % mod;
        b >>= 1;
    }
    return ans;
}
ll mul(ll x, ll y)
```

```
{
    return (1ll * x * y) % mod;
}
ll inv(ll x)
{
    return power(x, mod - 2) % mod;
}
int main()
{
    scanf("%d%d%d", &n, &m, &k);

    fac[0] = 1;
    for (int i = 1; i < N; i++)
        fac[i] = mul(fac[i - 1], i);
    for (int i = 0; i < N; i++)
        ifac[i] = inv(fac[i]);

    ll ans = 0, sum = n + k + m;
    for (int i = n; i <= sum; i++) { // 取出n张A牌
        for (int j = 0; j <= m; j++) { //B牌数量
            int C = i - n - j; //C牌数量
            if (C < 0 || C > k)
                continue;
            ll tmp = mul(fac[i - 1], mul(ifac[n - 1], mul(ifac[j], ifac[C])));
            tmp = mul(tmp, power(3, sum - i));
            ans = (ans + tmp) % mod;
        }
    }
    printf("%lld\n", ans);
}
```

第 2 章 atcode 062

2.1 arc062

2.1.1 c-AtCoDeer and Election Report

每一次票数都是增加的，所以找到约去的公因子就好。

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    long long N;
    cin >> N;
    vector<long long> T, A;
    long long t = 1, a = 1;
    for (long long i = 0; i < N; i++) {
        long long x, y;
        cin >> x >> y;
        T.push_back(x);
        A.push_back(y);
    }
    for (long long i = 0; i < N; i++) {
        long long temp = max((t + T[i] - 1) / T[i], (a + A[i] - 1) / A[i]);
        t = temp * T[i];
        a = temp * A[i];
    }
    cout << a + t << endl;
    //system("pause");
    return 0;
}
```

2.1.2 d-Iroha and a Grid

就是一个只有石头和布的石头剪刀布... 然后任何时候石头都得比剪刀出的次数多

一个特别好猜的结论，只要能出布那就出布，因为不管怎么样出布都稳赚不赔，所以按这个结论直接模拟就好

拿两个 cur 一个维护布数一个维护石头数就行了

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define N 100010
char s[N];
int cur1, cur2, ans;
int main()
{
    scanf("%s", s + 1);
    int n = strlen(s + 1);
    if (s[1] == 'p')
        ans = -1;
    cur2 = 1;
```

```
    for (int i = 2; i <= n; i++) {
        if (s[i] == 'p') {
            if (cur1 < cur2) {
                cur1++;
            } else
                ans--, cur2++;
        } else {
            if (cur1 < cur2) {
                cur1++;
                ans++;
            } else
                cur2++;
        }
    }
    printf("%d\n", ans);
    //system("pause");
    return 0;
}
```

第 3 章 atcode 063

3.1 arc063

3.1.1 c-1D Reversi

黑白棋，div3 的签到题

```
#include<bits/stdc++.h>
#define INF 0x3f3f3f3f
using namespace std;
int main() {
    string S; cin >> S;
    int k=S.length();
    int tmp = 1;
    for(int i = 0; i < k-1;i++){
        if(S[i] != S[i+1]) tmp++;
    }
    cout << tmp-1 << endl;
    return 0;
}
```

3.1.2 d-An Invisible Hand

题目的意思就是说一个商人可以从一个地方买苹果，然后再下不知道几个地方卖出去，每个地方都有个苹果的价值（且不相等），他想取得最大的利润，（毕竟商人）。而另一个竞争对手想要阻止他，哪怕只令他少赚一块钱，他可以任意修改地方苹果售价，但是要付出相等的代价。求最小的代价。

那我们只需要求出第一个商人最大价值出现了几次（因为地方售价不相等，所以可以不会出现改一个地方售价影响两个最大价值的情况），然后改动他卖出或者出售地方售价就 ok，毕竟求最小那么我们就只改动 1 就好，那么最小代价就变成了，最大利润出现的次数。

```
#include <bits/stdc++.h>
using namespace std;
#define N 100010
#define inf 0x3f3f3f3f
#define ll long long
int n, t;
int a[N];
int main()
{
    scanf("%d%d", &n, &t);
    for (int i = 1; i <= n; i++)
        scanf("%d", &a[i]);
    int ans = 0, mx = 0;
    int tot = 0;
    for (int i = n; i; i--) {
        mx = max(a[i], mx);
        ans = max(mx - a[i], ans);
    }
    mx = 0;
```

```
    for (int i = n; i; i--) {
        mx = max(a[i], mx);
        if (mx - a[i] == ans)
            tot++;
    }
    printf("%d\n", tot);
    return 0;
}
```

3.1.3 e-Painting Graphs with AtCoDeer

堆 + 递推

```
#include <bits/stdc++.h>
using namespace std;
#define N 100010
#define inf 0x3f3f3f3f
int n, K, d[N];
int head[N << 1], cnt, vis[N];
struct edge {
    int to, nxt;
} e[N << 1];
priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> q;
void ins(int u, int v)
{
    e[++cnt].to = v;
    e[cnt].nxt = head[u];
    head[u] = cnt;
}
int main()
{
    for (int i = 0; i < N; i++)
        d[i] = inf;
    scanf("%d", &n);
    for (int i = 1; i < n; i++) {
        int u, v;
        scanf("%d%d", &u, &v);
        ins(u, v);
        ins(v, u);
    }
    scanf("%d", &K);
    for (int i = 1; i <= K; i++) {
        int p, v;
        scanf("%d%d", &v, &p);
        d[v] = p;
        q.push(make_pair(p, v));
    }
    while (!q.empty()) {
        int u = q.top().second, val = q.top().first;
        q.pop();
        for (int i = head[u]; i; i = e[i].nxt) {
            int v = e[i].to;
            if (d[v] == inf)
                d[v] = val + 1, q.push(make_pair(d[v], v));
            if (abs(d[v] - d[u]) != 1) {
                puts("No");
                return 0;
            }
        }
```

```
    }
    puts("Yes");
    for (int i = 1; i <= n; i++) {
        printf("%d\n", d[i]);
    }
    return 0;
}
```

第 4 章 atcode 064

4.1 arc064

4.1.1 c-Boxes and Candies

贪心题目，意思是相邻；两个相加的和不能超过 x, 答案是最少减去多少。

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int main()
{
    ll N, x;
    cin >> N >> x;
    vector<ll> a(N);
    for (auto& in : a)
        cin >> in;
    ll ans = 0;
    for (int i = 1; i < N; i++) {
        if (a[i] > x) {
            ans += a[i] - x;
            a[i] = x;
        }
        if (a[i - 1] > x) {
            ans += a[i - 1] - x;
            a[i - 1] = x;
        }
        ll tmp = a[i] + a[i - 1];
        if (tmp > x) {
            ans += tmp - x;
            a[i] -= tmp - x;
        }
    }
    cout << ans << endl;
    return 0;
}
```

4.1.2 d-An Ordinary Game

这题结论要从奇偶性入手：

首先可以发现最后的字符串一定是形如“ababab”这样子由两个字符交替组成的, 所以我们可以根据最后的这个字符串的奇偶性入手来判断谁赢谁输, 如果字符串的第一个字符和最后一个字符相同，那么最后的字符串是奇数的，否则是偶数的, 然后再根据原串的奇偶性就可以判断出答案了.

```
#include <bits/stdc++.h>
using namespace std;
#define N 100010
char arr[N];
int main()
{
    int cnt = 0;
```

```
    scanf("%s", arr + 1);
    int n = strlen(arr + 1), k = n & 1;
    puts(abs(k - (arr[n] == arr[1])) & 1 ? "First" : "Second");
}
```

第 5 章 atcode 065

5.1 arc065

5.1.1 c-Boxes and Candies

采取逆序贪心，否则要多考虑好几种情况（从前往后贪心的话不能无脑选“dreamer”，”er” 可能为”erase”/”eraser” 的前缀）

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    string S; cin >> S;
    string s[4] = {"dream" , "dreamer", "erase" ,"eraser"};
    int at = (int)S.length();
    while(at > 0){
        bool f = false;
        for(int i = 0; i < 4;i++){
            if(at < s[i].length()) continue;
            string tmp = S.substr(at-s[i].length(),s[i].length());
            if(tmp == s[i]){
                f = true;
                at -= s[i].length();
                break;
            }
        }
        if(!f){
            cout << "NO" << endl;
            return 0;
        }
    }
    cout << "YES" << endl;
    return 0;
}
```

5.1.2 d-Connectivity

并查集

```
#include <bits/stdc++.h>
#define INF 0x3f3f3f3f
using namespace std;
typedef long long ll;
const int N = 2e5 + 10;
int pre1[N], pre2[N];
int find(int x, int* pre)
{
    return pre[x] == x ? pre[x] : pre[x] = find(pre[x], pre);
}
int join(int x, int y, int* pre)
{
    int px = find(x, pre);
    int py = find(y, pre);
    if (px != py) {
        pre[px] = py;
```

```
    }
}
int main()
{
    int n, k, l;
    cin >> n >> k >> l;
    for (int i = 1; i <= n; i++)
        pre1[i] = i, pre2[i] = i;
    for (int i = 1; i <= k; i++) {
        int x, y;
        cin >> x >> y;
        join(x, y, pre1);
    }
    for (int i = 1; i <= l; i++) {
        int x, y;
        cin >> x >> y;
        join(x, y, pre2);
    }
    for (int i = 1; i <= n; i++) {
        find(i, pre1);
        find(i, pre2);
    }
    map<pair<int ,int>,int> mp;
    for(int i=1;i<=n;i++){
        mp[make_pair(pre1[i],pre2[i])]++;
    }
    for(int i=1;i<=n;i++){
        cout<<mp[make_pair(pre1[i],pre2[i])]<<"␣";
    }
    cout<<endl;
    system("pause");
}
```