2023ACM新生赛讲义

A 数一数三角形

在此选取了一位去吃饭了的同学的代码，虽然AC了，但真的AC了嘛?

```cpp
#include <iostream>
using namespace std;
typedef long long LL;
const int N = 200010;
int n;
LL a[N], sum;
int main()
{
    scanf("%d", &n);
    for (int i = 1; i <= n; i ++ )
    {
        scanf("%lld", &a[i]);
        sum += a[i];
    }
    printf("%lld", sum / 3);
}
```

请你构造出一组Hack样例

B 作文批改

> 如果你AC了这道题，你就真的会了这道题嘛?

按照题意**有逻辑**模拟即可

```cpp
#include<bits/stdc++.h>
using namespace std;
#define LL long long
#define PII pair<int, int>
#define PLL pair<LL, LL>
#define xx first
#define yy second
#define endl '\n'
const int N = 2e5 + 10;
void solve(){
    string a;
    int n, flag = 0, cnt = 0;
    cin >> n;
    for(int j = 0;j < n;j++){
        cin >> a;
        for(int i = 0;i < a.size();i++){
            if(i == 0){
                if(a == "i" || a == "i.")cnt++;
                else if(a == "I" || a == "I.")continue;
```

```
                else if(flag == 0 && a[i] <= 'z' && a[i] >= 'a'){
                    cnt++;
                }
                else if(flag == 1 && a[i] <= 'Z' && a[i] >= 'A'){
                    cnt++;
                }
            }
            else{
                if(a[i] <= 'Z' && a[i] >= 'A')cnt++;
            }
        }
        if(flag == 0)flag = 1;
        if(a[a.size() - 1] == '.')flag = 0;
    }
    cout << cnt;
}
int main(){
    int T = 1;
//    cin >> T;
    while(T--){
        solve();
    }
}
```

C 找找ACM

- 能暴力吗?
- 考虑边界了吗?
- 边界考虑对了吗?

```
#include<bits/stdc++.h>
using namespace std;
#define LL long long
#define PII pair<int, int>
#define PLL pair<LL, LL>
#define xx first
#define yy second
#define endl '\n'
const int N = 2e6 + 10;
int sum[N];
void solve(){
    int n, m;
    string s;
    cin >> n >> m;
    cin >> s;

    for(int i = 0; i + 2 < s.size(); i++)
        if(s.substr(i, 3) == "acm") sum[i + 2]++;

    for(int i = 2; i < n; ++i) sum[i] += sum[i - 1];

    while(m--)
    {
```

```
        int l, r;
        cin >> l >> r;
        cout << sum[r] - sum[l + 1] << endl;//! ! ! !
    }
}
int main(){
    std::cin.tie(0);
    std::ios::sync_with_stdio(0);
    int T = 1;
//    cin >> T;
    while(T--){
        solve();
    }
}
```

D 字符串相约

- 你可以任选一个字符串都能修改
- 你可以两个字符串一起修改
- 什么时候不能成功?

- 注意到字符串首尾一定分别为 0 和 1，如果 等，则一定形如 `000011111`。判断两字符串是否在相同位置拥有 `01` 字串即可。
- 为什么呢?

E 简单DP

dp，转移方程为 $dp[i] = dp[i] \pm dp[i-x]$。这里给出代码，暂不要求掌握

```
#include<bits/stdc++.h>
using namespace std;
#define LL long long
#define PII pair<int, int>
#define PLL pair<LL, LL>
#define xx first
#define yy second
#define endl '\n'
const int N = 2e5 + 10;
const int MOD = 998244353;
int f[N];
void solve(){
    f[0] = 1;
    int n, m;
    cin >> n >> m;
    while(n --){
        char op;
        int x;
        cin >> op >> x;
        if(op == '+'){
            for(int i = m;i >= x;i--){
                (f[i] +=f[i - x]) %= MOD;
```

```
            }
        }
        else{
            for(int i = x;i <= m;i++){
                (f[i] += MOD - f[i - x]) %= MOD;
            }
        }
        cout << f[m] << endl;
    }

}
int main(){
    int T = 1;
//    cin >> T;
    while(T--){
        solve();
    }
}
```

F 两点距离

- 最大的最小值、最小的最大值，我们应该想到二分
- check怎么写?
- 先将点按 x 排序，固定一个点，枚举其余满足 $|x_i - x_j| \geq mid$ 的点，判断 $|y_i - y_j|$ 的最大值是否也大于等于 mid。
- 时间复杂度是?
- 这里给出一份错误代码，哪里错了?
-

```
#include<bits/stdc++.h>
using namespace std;
#define LL long long
#define PII pair<int, int>
#define PLL pair<LL, LL>
#define xx first
#define yy second
#define endl '\n'
const int N = 2e5 + 10;
PII a[N];
int n;
bool check(int x){
    for(int i = 0, j = 0;i < n;i++){
        for(;j < i && a[i].xx - a[j].xx >= x;j++){
            if(a[i].yy - a[j].yy >= x || a[j].yy - a[i].yy >=
x)return true;
        }
    }
    return false;
}
void solve(){
    cin >> n;
    for(int i = 0;i < n;i++)cin >> a[i].xx >> a[i].yy;
```

```
    sort(a, a + n);
    int l = 0, r = 4;
    while(l < r){
        int mid = l + r + 1 >> 1;
        if(check(mid))l = mid;
        else r = mid - 1;
    }
    cout << l;
}
int main(){
    int T = 1;
//    cin >> T;
    while(T--){
        solve();
    }
}
```

G 每日一颗?

线段树、平衡树、对顶堆

不要求掌握

H 找0

签到题，注意0的特判即可

I 找0_pro

当然，你可以写高精度，那么只能写高精度吗?

J 黎明之剑

遍历整个棋盘，从未访问过的 `#` 或 `?` 开始跑 dfs，更新答案

没学过的dfs的同学暂不要求掌握

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
const int MAXN = 5e2 + 10;

int n, m;
int g[MAXN][MAXN];
int ans1, ans2;

int dir[4][2] = {{0, 1}, {0, -1}, {1, 0}, {-1, 0}};
bool legal(int x, int y)
{
    if(g[x][y] == 0) return false;
    if(x < 1 || x > n || y < 1 || y > m) return false;
    return true;
}
```

```cpp

void dfs(int x, int y, int &res1, int &res2)
{
    if(!legal(x, y)) return;
    ++res1;
    res2 += g[x][y];
    g[x][y] = 0;

    for(int i = 0; i < 4; ++i)
    {
        int nx = x + dir[i][0], ny = y + dir[i][1];
        dfs(nx, ny, res1, res2);
    }
}

signed main()
{
    cin >> n >> m;
    for(int i = 1; i <= n; ++i)
    {
        char ch = getchar();
        for(int j = 1; j <= m; ++j)
        {
            ch = getchar();
            if(ch == '*') g[i][j] = 0;
            else if(ch == '#') g[i][j] = 1;
            else g[i][j] = 2;
        }
    }

    int res1 = 0, res2 = 0;
    for(int i = 1; i <= n; ++i)
    {
        for(int j = 1; j <= m; ++j)
        {
            res1 = res2 = 0;
            if(g[i][j]) dfs(i, j, res1, res2);
            if(res1 > ans1 || (res1 == ans1 && res2 < ans2)) ans1 = res1, ans2 = res2;
        }
    }

    cout << ans1 << ' ' << ans2;

    return 0;
}
```

K 重构代码

- 题意：给定n个bug位置，修改每个bug需要i需改 $a[i]-a[i-1]$ 行代码，只修改m个bug，求最少修改的行数
- 将 a_n 从小到大排序后，相邻两项做差，删去前 $m-1$ 大的差值剩下求和就是答案。

- ```cpp
#include<bits/stdc++.h>
using namespace std;
#define LL long long
#define PII pair<int, int>
#define PLL pair<LL, LL>
#define xx first
#define yy second
#define endl '\n'
const int N = 2e5 + 10;
int a[N], b[N];
void solve(){
    int x, n, m;
    cin >> x >> n >> m;
    for(int i = 1; i <= n; i++) cin >> a[i];
    sort(a + 1, a + 1 + n);
    for(int i = 2; i <= n; i++){
        b[i - 1]=a[i] - a[i - 1];
    }
    sort(b + 1, b + n);
    int ans = 0;
    for(int i = n - m; i >= 1; i--){
        ans += b[i];
    }
    cout<<ans<<endl;
}
int main(){
    int T = 1;
//   cin >> T;
    while(T--){
        solve();
    }
}
```

- 另给出使用大根堆的做法

```cpp
#include<bits/stdc++.h>
using namespace std;
#define int long long
typedef long long ll;
typedef pair<int, int> pii;
const int INF = 0x3f3f3f3f;
const int MAXN = 2e5 + 10;

int x, n, m, a[MAXN];

priority_queue<int> q;
int vis[MAXN];
int ans;

signed main()
{
    ios::sync_with_stdio(false);
    cin >> x >> n >> m;
    for(int i = 1; i <= n; ++i)
```
```

```
    {
        cin >> a[i];
    }
    sort(a + 1, a + 1 + n);

    for(int i = 1; i < n; ++i)
    {
        q.push(a[i + 1] - a[i]);
    }
    --m;
    while(!q.empty() && m--)
    {
        q.pop();
    }

    while(!q.empty())
    {
        ans += q.top();
        q.pop();
    }

    cout << ans;
    return 0;
}
```

L 概率机器人

- 每个单元格都可以向下或向右走，dfs 的复杂度为 $O(2^{nm})$，一定会超时。
- 注意到某一单元格的答案一定只能从下方单元格和右方单元格转移过来，所以 dp 倒序遍历，复杂度为 $O(nm)$。
- 这里给出代码，暂不要求掌握

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 1e3 + 10;

int n, m, v[MAXN][MAXN], p[MAXN][MAXN];
double ans;

double dp[MAXN][MAXN];

void fast_io() {ios::sync_with_stdio(false); cin.tie(0);
cout.tie(0);}
signed main()
{
    fast_io();
    cin >> n >> m;
    for(int i = 1; i <= n; ++i)
        for(int j = 1; j <= m; ++j)
            cin >> v[i][j];

    for(int i = 1; i <= n; ++i)
```

```
        for(int j = 1; j <= m; ++j)
            cin >> p[i][j];

    for(int i = n; i >= 1; --i)
    {
        for(int j = m; j >= 1; --j)
        {
            dp[i][j] += v[i][j] + dp[i + 1][j] * p[i][j] * 1.0 / 100
+ dp[i][j + 1] * (100 - p[i][j]) * 1.0 / 100;
        }
    }

    cout << floor(dp[1][1] + 0.5);
    return 0;
}
```

M 养生的大学生

- 所以你想到位运算了吗?
- 样例一：3 -> `11` 加一杯 `100`
- 样例二：`5` -> `101` 刚好