2025 SYNU 4月周赛 Round I

愚人节快乐xD

- B 卡 long long
- C 大模拟
- D 序列有负数

做题要细心哦！

by Orange

题解报告

A. Orange与括号序列

首先，一个合法的括号序列长度一定是偶数，因为一定有 `(` 和 `)` 一一对应，因此如果长度不是偶数则直接输出 `-1` 。

其次，观察到操作可以任意进行，因此问题本质上就是，对于一个指定长度的字符串，构造的括号序列是否唯一，而**只有长度为2的合法括号序列是唯一的**，超过2的括号序列都可以嵌套或者追加。因此再判断字符串长度是否为2即可判断 `yes` 或者 `no` 。

```
void solve()
{
    string s;
    cin >> s;
    if(s.size() == 2) cout << "Yes\n";
    else if(s.size() % 2 == 0) cout << "No\n";
    else cout << "-1\n";
}
```

B. 虚数选择

本题看上去是一个简单题，实际上暗藏玄机。

首先你注意到 $n \leq 500$，因此可能你会考虑暴力枚举 i 和 j 的所有组合，实际上这个做法并没有错，但是你需要观察 a_i, b_i 的数据范围。

假设你直接计算 z_i 和 z_j 的乘积的模：

$$z_i z_j = (a_i + b_i\mathrm{i})(a_j + b_j\mathrm{i}) = a_i a_j + a_i b_j\mathrm{i} + b_i a_j\mathrm{i} - b_i b_j = a_i a_j - b_i b_j + (a_i b_j + b_i a_j)\mathrm{i}$$

$$|z_i z_j| = \sqrt{(a_i a_j - b_i b_j)^2 + (a_i b_j + b_i a_j)^2}$$

此时 a_i, b_i, a_j, b_j 均为 10^6 ，这意味着计算其中两数乘积时答案会来到 10^{12}，再次平方则会来到 10^{24}，这已经远超已知整数类型能表达的最大极限了。

考虑高中的虚数公式：**乘积的模长等于模长的乘积**

$$|z_i z_j| = |z_i||z_j|$$

我们先算出每个虚数模长，再找出最大的两个相乘，这样就避免了溢出。

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long i64;
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    int n;
    cin >> n;
    i64 mx1 = 0, mx2 = 0;
    for (int i = 0; i < n; i++)
    {
        i64 a, b;
        cin >> a >> b;
        i64 w = a * a + b * b;
        if (w > mx1)
        {
            mx2 = mx1;
            mx1 = w;
        }
        else if (w > mx2)
        {
            mx2 = w;
        }
    }
    printf("%.6lf\n", sqrt(mx1) * sqrt(mx2));
}
```

C. 图像压缩

愚人节限定美味大模拟~

把所有灰阶转换成16进制整数用map存其出现次数，然后丢进vector按出现次数排序即可，取前16大即可。

然后对于每个灰阶，暴力枚举剩下的16种灰阶，找到符合条件的进行替换即可。

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

typedef pair<int,int> pii;

int main()
{
    int n;
    cin >> n;
    vector<string> s(n);
    map<int,char> dict = {{0,'0'},{1,'1'},{2,'2'},{3,'3'},
    {4,'4'},{5,'5'},{6,'6'},{7,'7'},{8,'8'},{9,'9'},{10,'A'},
    {11,'B'},{12,'C'},{13,'D'},{14,'E'},{15,'F'}};
    map<char,int> rdict = {{'0',0},{'1',1},{'2',2},{'3',3},
    {'4',4},{'5',5},{'6',6},{'7',7},{'8',8},{'9',9},{'A',10},
    {'B',11},{'C',12},{'D',13},{'E',14},{'F',15}};
    map<int,int> mp;
    for(int i = 0; i < n; i ++)
    {
        cin >> s[i];
        for(int j = 0; j < s[i].size(); j+=2)
        {
            int t = rdict[s[i][j]]*16 +rdict[s[i][j + 1]];
            mp[t]++;
        }
    }
    int cnt = 0;
    vector<pii> base;
    for(auto [x,y] : mp)
    {
        base.push_back({x,y});
    }
    sort(base.begin(),base.end(),[&](pii p1,pii p2){
        if(p1.second == p2.second)
        {
            return p1.first < p2.first;
        }
        else return p1.second > p2.second;
    });
    vector<int> id;
    for(auto [x,y] : base)
    {
        cout << dict[x/16] << dict[x%16];
        id.push_back(x);
        cnt ++;
        if(cnt >= 16) break;
    }
    cout << endl;
    for(int i = 0; i < s.size(); i++)
    {
        for(int j = 0; j < s[i].size(); j+=2)
        {
            int t = rdict[s[i][j]]*16 + rdict[s[i][j+1]];
            int res = 0;
            for(int k = 0; k < id.size(); k++)
            {
                if(abs(t - id[k]) < abs(t - id[res])) res = k;
                else if(abs(t - id[k]) == abs(t - id[res])
                && k < res) res = k;
            }
            cout << dict[res];
        }
        cout << endl;
    }
}
```

D. 互斥子串

注意到，原序列本质上是若干个极长合法子串的叠加。

我们钦定 $st\_pos[i]$ 表示 a_i 所在的极长合法子串的起点，$len[i]$ 表示 a_i 所在的极长合法子串**结束于** i 的**部分长度**。

首先，我们可以利用双指针之类的算法将原序列中所有的**极长合法子串**预处理出来，顺便预处理出上面的 $st\_pos[i]$ 和 $len[i]$。

不难发现，最后的序列一定会被分成若极长合法子串的叠加，每个 a_i 所在的极长合法子序列的左端点肯定要么与前一个位置相同，要么在前一个位置后面，因此 $st\_pos[i]$ 的单调的，具有二分性。

对于每次查询 (l,r) ，我们可以通过 l 将 $st\_pos[i]$ 分成两类：

1. 若 $st\_pos[i]$ 大于 l，则这部分的答案就是 $\max_{st\_pos[i]>l}^{r} len[i]$，那么这就是一个RMQ问题，直接用st表解决。
2. 若 $st\_pos[i]$ 小于等于 l，则说明 l 刚好截断了这些 a_i 构成的一个极长合法子串，那么这部分的答案即为这个极长合法子串在区间内的部分，即 l 与其右端点构成的区间。那么其右端点等价于第一个大于 l 的极长合法子串左端点的前一个位置。因此我们只需要在 st_pos 中二分查找第一个大于 l 的值，不妨记为 p，则 $p-1$ 就是我们要找的右端点，其长度则为 $p-l$。
   - 注意！假设我们二分找到的第一个大于 l 的 st_pos 大于 r，则说明 l,r 完全包含在一个极长合法子串内，答案就是区间长度 $r-l+1$

最终答案来自于上述情况的最大值。

tips: **题中的 a_i 存在负数**，可以通过加上一个偏移使其变成正数，或者做一次离散化也可以

```cpp
#include <bits/stdc++.h>
using namespace std;
#define IOS ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
// #define int long long
#define endl '\n'
#define all(_x) _x.begin(), _x.end()
#define range(_x, _st, _ed) (_x.begin() + _st), (_x.begin() + _ed)
#define rep(_x, _y, _z) for (int _x = _y; _x <= _z; _x++)
#define matrix(_x, _y, _z) vector<vector<int>>(_x, vector<int>(_y, _z))
#define debug(_x) cout << #_x << '=' << _x << endl
using i64 = long long;
using i128 = __int128;
using pii = pair<int, int>;
using mat = vector<vector<int>>;
using u64 = unsigned long long;
constexpr int N = 2e5 + 10;
// dont use umap!!!

struct ST {
    const int n, k;
    vector<int> in1, in2;
    vector<vector<int>> Max, Min;
    ST(int n) : n(n), in1(n + 1), in2(n + 1), k(__lg(n)) {
        Max.resize(k + 1, vector<int>(n + 1));
        Min.resize(k + 1, vector<int>(n + 1));
    }
    void init() {
        for (int i = 1; i <= n; i++) {
            Max[0][i] = in1[i];
            Min[0][i] = in2[i];
        }
        for (int i = 0, t = 1; i < k; i++, t <<= 1) {
            const int T = n - (t << 1) + 1;
            for (int j = 1; j <= T; j++) {
                Max[i + 1][j] = max(Max[i][j], Max[i][j + t]);
                Min[i + 1][j] = min(Min[i][j], Min[i][j + t]);
            }
        }
    }
    int getMax(int l, int r) {
        if (l > r) {
            swap(l, r);
        }
        int k = __lg(r - l + 1);
        return max(Max[k][l], Max[k][r - (1 << k) + 1]);
    }
    int getMin(int l, int r) {
        if (l > r) {
            swap(l, r);
        }
        int k = __lg(r - l + 1);
        return min(Min[k][l], Min[k][r - (1 << k) + 1]);
    }
};

signed main()
{
    IOS;
    int n, m;
    cin >> n >> m;
    vector<int> a(n + 1);
    for(int i = 1; i <= n; i ++)
    {
        cin >> a[i];
        a[i] += 1e6;
    }
    ST ST(n);
    set<int> st;
    vector<int> st_pos(n + 1), Len(n + 1);
```

```
    // 预处理
    for(int i = n, j = n; i >= 1; i --)
    {
        // 找到 i 能到的最左端
        while(st.count(a[j]) == 0 and j >= 1) st.insert(a[j--]);
        st_pos[i] = j + 1;
        Len[i] = i - j;
        ST.in1[i] = i - j;
        st.erase(a[i]);
    }
    // 建立st表
    ST.init();
    while(m--)
    {
        int l, r;
        cin >> l >> r;
        l ++, r ++;
        // 二分第一个大于l的极长子串的左端点
        int p = lower_bound(st_pos.begin() + 1, st_pos.end(), l) - st_pos.begin();
        // 如果在区间内，则答案为两部分的max
        if(p <= r) cout << max(ST.getMax(p, r), p - l) << '\n';
        // 否则，说明l和r位于同一个极长子串内，答案就是区间长度
        else cout << r - l + 1 << '\n';
    }
    return 0;
}
```