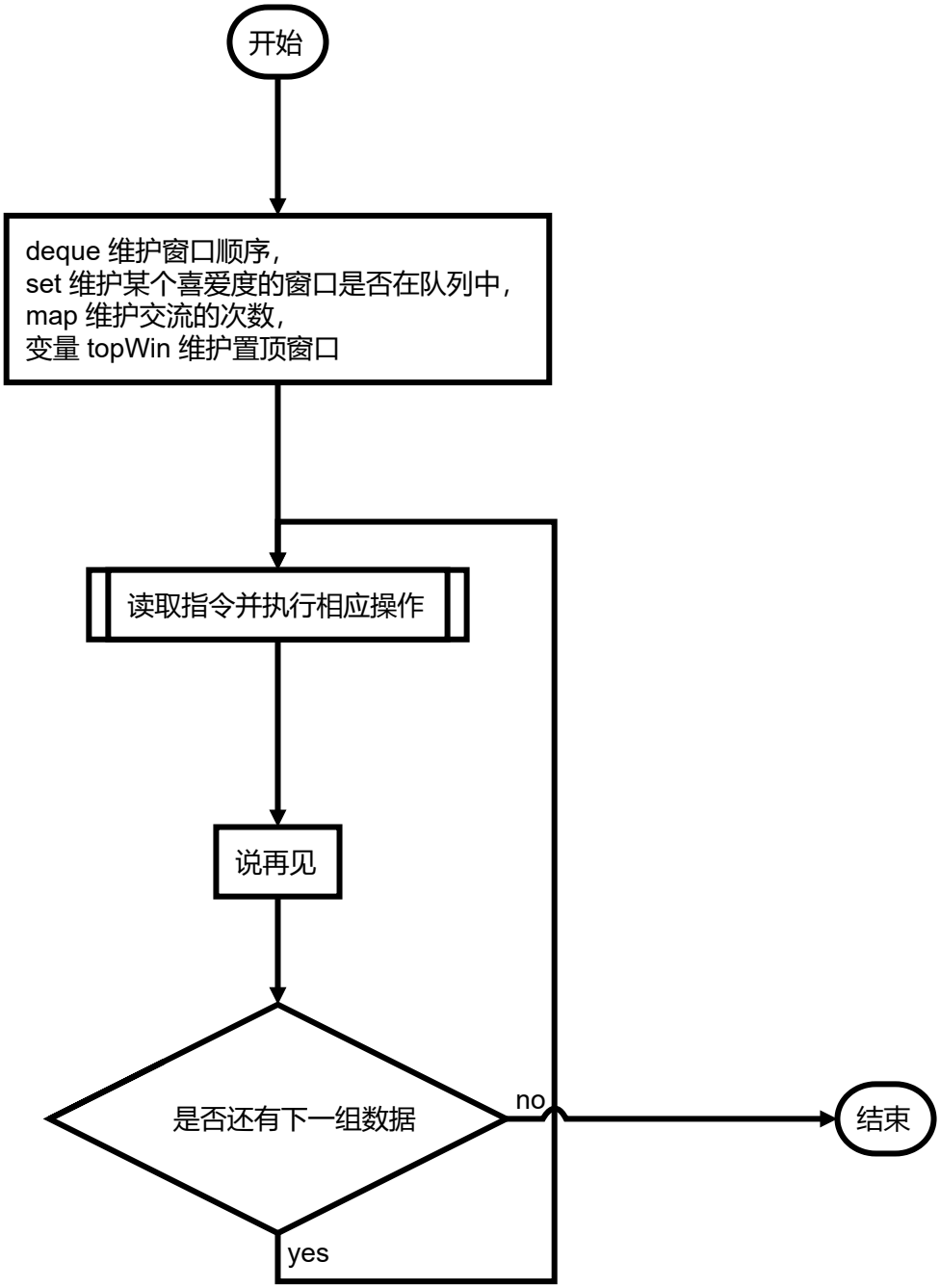
一、复杂模拟题

1. 代码流程



具体操作：

Add： 添加窗口，通过 set 判断是否存在，若无则加至 deque 队尾；

Close： set 判断存在，若存在则从 deque 中 erase， 并输出 map 中对应值；

chat： 交流， map 计数；

rotate： 将队列中第 x 个窗口移至队首，遍历 deque 找到指定窗口， push_front 至队首；

prior： 将 TopWin 移至队首，通过 TopWin == INT32_MAX 判断是否有窗口置顶,操作类同 rotate；
choose： 类同 rotate；
top： 用 topWin 存储对应窗口；
untop： 将 topWin 赋值为 INT32_MAX ， 表示当前没有指定窗口；

2. 提交代码

```cpp
#include<bits/stdc++.h>
using namespace std;
using ll = long long;
const string op = "OpId #";
int Cnt = 1;

// 置顶、喜爱度
// 窗口次序

set<int> s;
int topWin = INT32_MAX;
map<int, ll> mp;
deque<int> Q;

// O(5 * n * n)

void solve() {
    int n;
    cin >> n;

    string str;
    ll u;

    // simulation operation
    for (int i = 1; i <= n; i++) {
        cin >> str;
        if (str == "Add") {
            cin >> u;
            if (s.count(u)) {
                cout << op << Cnt++ << ": same likeness.\n";
            }
            else {
                s.insert(u);
                mp[u] = 0;
                Q.push_back(u);
                cout << op << Cnt++ << ": success.\n";
            }
        }
        else if (str == "Close") {
            cin >> u;
            if (s.count(u)) {
                s.erase(u);
                if (topWin == u) topWin = INT32_MAX;
                for (auto j = Q.begin(); j != Q.end(); j++) {
                    if (*j == u) {
                        Q.erase(j);
                        break;
                    }
                }
                cout << op << Cnt++ << ": close " << u << " with " << mp[u] << ".\n";
```

```
    }
    else {
        cout << op << Cnt++ << ": invalid likeness.\n";
    }
}
else if (str == "Chat") {
    cin >> u;
    if (s.size() > 0) {
        if (topWin != INT32_MAX)
            mp[topWin] += u;
        else {
            int f = Q.front();
            mp[f] += u;
        }
        cout << op << Cnt++ << ": success.\n";
    }
    else {
        cout << op << Cnt++ << ": empty.\n";
    }
}
else if (str == "Rotate") {
    cin >> u;

    if (u >= 1 && u <= s.size()) {
        int r = 0, tmp;
        for (auto j = Q.begin(); j != Q.end(); j++) {
            ++r;
            if (r == u) {
                tmp = *j;
                Q.erase(j);
                Q.push_front(tmp);
                break;
            }
        }
        cout << op << Cnt++ << ": success.\n";
    }
    else {
        cout << op << Cnt++ << ": out of range.\n";
    }

}
else if (str == "Prior") {
    if (s.size() > 0) {
        int Max = INT32_MIN;
        auto tmp = Q.begin();
        for (auto j = Q.begin(); j != Q.end(); j++) {
            if (*j > Max) {
                Max = *j;
                tmp = j;
            }
        }
```

```
            Q.erase(tmp);
            Q.push_front(Max);
            cout << op << Cnt++ << ": success.\n";
        }
        else {
            cout << op << Cnt++ << ": empty.\n";
        }

    }
    else if (str == "Choose") {
        cin >> u;
        if (s.count(u)) {
            for (auto j = Q.begin(); j != Q.end(); j++) {
                if (*j == u) {
                    Q.erase(j);
                    break;
                }
            }
            Q.push_front(u);
            cout << op << Cnt++ << ": success.\n";
        }
        else {
            cout << op << Cnt++ << ": invalid likeness.\n";
        }
    }
    else if (str == "Top") {
        cin >> u;
        if (s.count(u)) {
            topWin = u;
            cout << op << Cnt++ << ": success.\n";
        }
        else {
            cout << op << Cnt++ << ": invalid likeness.\n";
        }
    }
    else if (str == "Untop") {
        if (topWin == INT32_MAX) {
            cout << op << Cnt++ << ": no such person.\n";
        }
        else {
            topWin = INT32_MAX;
            cout << op << Cnt++ << ": success.\n";
        }
    }

    // for (auto j = Q.begin(); j != Q.end(); j++) {
    //     cout << *j << " ";
    // }
    // cout << '\n';
}
```

```
    // say "Bye"
    if (topWin != INT32_MAX) {
        if (mp[topWin] > 0)
            cout << op << Cnt++ << ": Bye " << topWin << ": " << mp[topWin] << ".\n";
    }

    while (!Q.empty()) {
        int f = Q.front();
        Q.pop_front();
        if (f == topWin) continue;
        if (mp[f] > 0) cout << op << Cnt++ << ": Bye " << f << ": " << mp[f] << ".\n";
    }
}

int main() {

    // freopen("out", "w", stdout);
    ios::sync_with_stdio(false);
    // cin.tie(nullptr);

    int t;
    cin >> t;
    while (t--) {
        Cnt = 1;
        topWin = INT32_MAX;
        s.clear();
        mp.clear();
        solve();
    }
    return 0;
}
```

3. 解法的时间复杂度

操作 rotate,prior,choose 的时间复杂度达到 $O(n)$ ， 总模拟的复杂度为 $O(t*n^2)$.

B. 晨晨的苹果Plus

题目大意

给定一个序列 $\{a_n\}$, 求满足 $a_1+a_2+...+a_i=a_{i+1}+a_{i+2}+...+a_j=a_{j+1}+a_{j+2}+..+a_n$ 的数对 (i,j) 的个数。

解法

扫描数组，变量 c 记录前缀和为 $sum/3$ 的节点，前缀和等于 $2*sum/3$ 时, $ans=ans+c$.

时间复杂度

$O(n)$

代码

```cpp
#include<iostream>
#include<vector>
using namespace std;
using ll = long long;

// 1e10
vector<int> a(1000010);

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n; cin >> n;
    ll sum = 0;
    ll ans = 0;
    for (int i = 1; i <= n; i++) cin >> a[i], sum += a[i];

    if (sum % 3) cout << "0\n";
    else {
        ll b = 0, c = 0;
        for (int i = 1; i < n; i++) {
            b += a[i];
            if (b == sum * 2 / 3) {
                ans += c;
            }
            if (b == sum / 3) c++;
        }
        cout << ans << '\n';
    }

    return 0;
}
```

C.

题目大意

栈中维护 n 个元素， m 次操作，每次将某个元素取出，它上面的元素按原顺序放回栈中，之后将该元素放在栈顶。每次操作的代价为放在元素上方的元素权重之和，求 m 次操作的最小代价。

解法

给定操作序列 1 3 2 3 1，我们知道操作元素 2 时代价至少为元素 1 和 3 的权重之和，即操作某个元素 a 的代价至少为上次操作 a 之后到此次操作 a 之间的元素序列之和。
据此，我们可以推断出初始状态为元素在操作序列中的出现次序，出现越早的放在栈顶。

时间复杂度

$O(m^2)$

代码

```cpp
#include<iostream>
#include<stack>
#include<cstring>
using namespace std;
using ll = long long;

int a[10010], b[10010];
bool vis[10010];
bool vis2[10010];

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n, m;
    cin >> n >> m;

    for (int i = 1; i <= n; i++) cin >> a[i];

    for (int i = 1; i <= m; i++) {
        cin >> b[i];
    }

    int cnt;
    ll ans = 0;
    // 计算代价时反向考虑
    // 在操作哪些元素时会拿出该元素
    for (int i = 1; i <= m; i++) {
        if(vis2[b[i]]) continue;
        vis2[b[i]] = 1;
        memset(vis, 0, sizeof vis);
        cnt = 0;
        for (int j = i + 1; j <= m; j++) {
            if (b[j] == b[i]) {
                ans += a[b[i]] * cnt;
                memset(vis, 0, sizeof vis);
                cnt = 0;
                continue;
            }
            if (vis[b[j]]) continue;
            vis[b[j]] = true;
            cnt++;
        }
        ans += a[b[i]] * cnt;
        // cout << "c: " << ans << ' ' << cnt << '\n';
    }
    cout << ans << '\n';
    return 0;
}
```