CSP-S 第九场模拟赛讲评

QSGame-S 模拟赛(9)

难度：橙、黄、绿、蓝。

1. clock：贪心、分类讨论。
2. computer：堆、模拟。
3. section：组合数学、逆元、费马小定理。
4. order：折半搜索、无序哈希。

clock

题意

给出三个数字 a, b, c，给出满足性质下的最小操作次数。

满足以下两个条件之一为满足性质：

1. 如果恰好 b 是第二大；
2. 存在任意两个数字相等；

分析

如何使得操作数最小，先排序满足单调性再分类讨论：

1. 一开始就满足的情况，b 排在第二或者 $a=b|a=c|b=c$ 满足其中一个条件。
2. 其他情况，取俩俩操作数的差值最绝对的最小值（即操作次数）。

主要时间复杂度为排序，但只有 3 个数字，所以为常数 $O(1)$。

参考代码

```
int a, b, c;
int q[4];
int main() {
    cin >> a >> b >> c;
    q[1] = a, q[2] = b, q[3] = c;
    sort(q + 1, q + 4);
    if (a == b || a == c || b == c || q[2] == b) {
        cout << 0 << endl;
    } else
        cout << min({abs(a - b), abs(a - c), abs(b - c)}) << endl;
    return 0;
}
```

computer

题意

有 n 台计算机，第 i 台计算机的运算能力为 v_i 。

第 i 个任务在 a_i 时刻分配，指定计算机编号为 b_i, 耗时为 c_i 且算力消耗为 d_i。如果此任务成功分配，将立刻开始运行, 期间持续占用 b_i 号计算机 d_i 的算力, 持续 c_i 秒。

对于每次任务分配，如果计算机剩余的运算能力不足则输出 -1，并取消这次分配，否则输出分配完这个任务后这台计算机的剩余运算能力。

数据范围： $1 \leq n, m \leq 2 \times 10^5, 1 \leq a_i, b_i, c_i, v_i \leq 10^9, 1 \leq b_i \leq n;$

分析

算法时间复杂度在 $O(nlogn)$ 可以接受。

考虑开 N 个小根堆来维护计算机 i 的任务 `<结束时间，算力>`。

> 题目保证时刻 a_i 递增，不需要排序。

处理 m 个任务时，先处理掉小根堆内已过期的任务，即 $p[b_i].endTime <= a_i$，弹出并归还算力。

处理完过期任务后，判断当然计算机 b_i 是否剩余足够算力处理任务 m_i。

m 个任务，增删为 $logn$，每个任务最多进出一次，总时间复杂度为 $nlogn$。

参考代码

```
/* pii (结束时间, 算力) */
priority_queue<pii, vector<pii>, greater<pii>> q[N];
int n, m, power[N];
int main() {
    cin >> n >> m;
    for (int i = 1; i <= n; i++)
        cin >> power[i];
    for (int i = 1, a, b, c, d; i <= m; i++) {
        /* 开始, 计算机, 时长, 算力 */
        cin >> a >> b >> c >> d;
        /* 当前任务开始(及之前)的第b台计算机任务剔除 */
        while (q[b].size() && q[b].top().ft <= a) {
            power[b] += q[b].top().sd;
            q[b].pop();
        }
        /* 不够算力 */
        if (power[b] < d)
            cout << -1 << endl;
        else {
            power[b] -= d;
            q[b].push({a + c, d});
            cout << power[b] << endl;
        }
    }
}
```

section

题目理解

我们需要计算所有恰好包含 n 个 '0' 和 m 个 '1' 的 01 串 S 的极长颜色段数 $g(S)$ 之和，即 $f(n,m)$。

极长颜色段定义：满足以下条件的区间 $[l,r]$：

- 如果 $l \neq 1$，则 $S_{l-1} \neq S_l$
- 如果 $r \neq |S|$，则 $S_{r+1} \neq S_r$
- $\forall i \in [l,r), S_i = S_{i+1}$

关键观察

极长颜色段数 $g(S)$ 可以表示为：

$$g(S) = (\text{相邻不同字符对的数量}) + 1$$

因此：

$$f(n,m) = \sum_{\text{所有字符串}} g(S) = \sum_{\text{所有字符串}} (\text{相邻不同字符对数量} + 1)$$

公式推导

1. 计算字符串总数

所有恰好包含 n 个 '0' 和 m 个 '1' 的字符串个数为组合数：

$$\text{字符串总数} = C(n+m, n) = C(n+m, m)$$

2. 计算相邻不同字符对总数

对于每个相邻位置对 $(i, i+1)$，计算该位置字符不同的字符串数量：

- **相邻位置对数量**：$n+m-1$
- **每个位置对字符不同的情况**："01" 或 "10" 两种
- **固定一个位置对为不同时**，剩余 $n+m-2$ 个位置需要分配 $n-1$ 个 '0' 和 $m-1$ 个 '1'
- **分配方式数**：$C(n+m-2, n-1)$

因此相邻不同字符对总数为：

$$2 \times (n+m-1) \times C(n+m-2, n-1)$$

3. 综合公式

$$f(n,m) = 2 \times (n+m-1) \times C(n+m-2, n-1) + C(n+m, n)$$

边界情况：

- 当 $n=0$ 且 $m=0$ 时：$f(0,0)=1$
- 当 $n=0$ 或 $m=0$ 时：$f(n,m)=1$

样例验证

样例 1： $n=2, m=2$

- $C(4,2)=6$（字符串总数）
- $2\times 3\times C(2,1)=2\times 3\times 2=12$（相邻不同字符对总数）
- $f(2,2)=6+12=18\checkmark$

样例 2： $n=1, m=46$

- $C(47,1)=47$
- $2\times 46\times C(45,0)=2\times 46\times 1=92$
- $f(1,46)=47+92=139\checkmark$

代码实现

```
ll t, n, m;
ll fac[N], infac[N]; // 阶乘 i! 、阶乘的逆元 i!^{-1}
ll quickpow(ll a, ll b, int p) {
    ll res = 1;
    while (b) {
        if (b & 1)
            res = res * a % p;
        a = a * a % p;
        b >>= 1;
    }
    return res;
}
/* 预处理数据，计算组合数 */
void init() {
    /* 初始化 */
    fac[0] = infac[0] = fac[1] = infac[1] = 1;
    for (int i = 2; i < N; i++)
        fac[i] = fac[i - 1] * i % mod;
    for (int i = 2; i < N; i++)
        infac[i] = quickpow(fac[i], mod - 2, mod);
}
```

```
ll solve(ll n, ll m) { return fac[n] * infac[m] % mod * infac[n - m] % mod; }
int main() {
    /* 预处理 */
    init();
    cin >> t;
    while (t--) {
        cin >> n >> m;
        if (n == 0 && m == 0)
            cout << 1 << endl;
        else if (n == 0 || m == 0)
            cout << 1 << endl;
        else {
            ll p1 = 2 * (n + m - 1) % mod * solve(n + m - 2, n - 1) % mod;
            ll p2 = solve(n + m, n);
            cout << (p1 + p2) % mod << endl;
        }
    }
    return 0;
}
```

order

题意

从 $(0,0)$ 点出发，给定终点 (x_g,y_g) 和 n 条移动指令，每条指令表示在平面上的移动向量 (x_i,y_i)。要求选择恰好 k 条指令（$1 \le k \le n$），使得按照这些指令的顺序移动后能到达终点 (x_g,y_g)。对于每个 k，求有多少种选择指令的方案。

数据范围：$n \le 40$。

分析

由于 n 最大为 40，直接暴力枚举所有 2^n 种选择方案会超时（$2^{40} \approx 10^{12}$）。考虑使用**折半搜索**（Meet in the Middle）算法：

1. 将 n 条指令平均分成前后两半
2. 对前半部分进行深度优先搜索，记录所有可能的 (x, y, cnt) 三元组，其中：
    - (x, y) 表示移动后的坐标位置
    - cnt 表示使用的指令数量
    - 使用哈希表 `mp[x][y][cnt]` 记录方案数

3. 对后半部分进行深度优先搜索，对于每个到达的位置 (x', y') 和使用指令数 cnt'，在哈希表中查找是否存在位置 $(x_g - x', y_g - y')$ 的记录
4. 如果存在，则将对应的方案数累加到答案 $ans[cnt' + cnt]$ 中

时间复杂度：$O(2^{n/2})$，将指数级复杂度降低为平方根级别。

参考代码

```cpp
struct node {
    int x, y;
} a[N];

ll xg, yg; // 终点
ll ans[N]; // 使用 k 个指令可抵达终点的方案数

// 抵达某个点，指令个数，选择了哪几个指令。
unordered_map<ll, unordered_map<ll, unordered_map<int, int>>> mp;
```

核心搜索

```cpp
// _arrpos数组当前位置, _arrend数组最后遍历的位置, 当前所在位置x,y, 第几遍dfs_dfscnt
void dfs(int _arrpos, int _arrend, ll x, ll y, int cnt, int _dfscnt) {
    if (_arrpos > _arrend) {
        if (_dfscnt == 0)
            mp[x][y][cnt]++;
        else {
            if (mp.count(xg - x) && mp[xg - x].count(yg - y))
                for (auto &iter : mp[xg - x][yg - y])
                    ans[cnt + iter.first] += iter.second;
        }
        return;
    }
    // 选/不选
    dfs(_arrpos + 1, _arrend, x + a[_arrpos].x, y + a[_arrpos].y, cnt + 1,
        _dfscnt);
    dfs(_arrpos + 1, _arrend, x, y, cnt, _dfscnt);
}
```

主函数

```
int main() {
    ios::sync_with_stdio(false), cin.tie(nullptr);
    cin >> n >> xg >> yg;
    for (int i = 1; i <= n; i++)
        cin >> a[i].x >> a[i].y;
    dfs(1, n / 2, 0, 0, 0, 0); // 前半段
    dfs(n / 2 + 1, n, 0, 0, 0, 1); // 后半段
    for (int i = 1; i <= n; i++)
        cout << ans[i] << endl;
    return 0;
}
```