

计蒜客信息学 2023 年 2 月普及组模拟赛题解

题目	知识点
外卖	分类讨论、全排列
区间问题 I	前缀和
涂色	森林、连通块、快速幂
区间问题 II	数学、前缀和、预处理

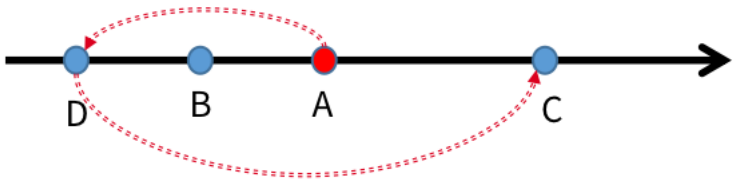
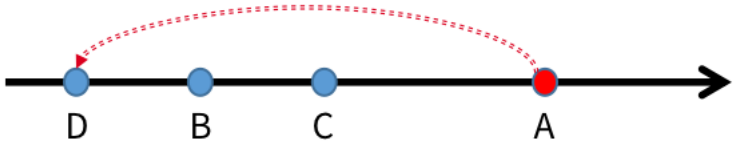
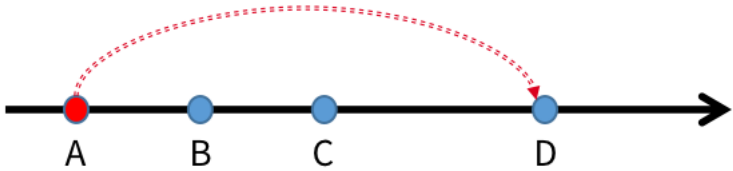
外卖

题解

你当然可以通过直接全排列枚举走的顺序通过本题，但是这样太麻烦了。

其实本题可以分为三种情况：

- 1. A 点在最左侧（最小值），那么最短距离 = 最大值 - 最小值；
- 2. A 点在最右侧（最大值），那么最短距离 = 最大值 - 最小值；
- 3. A 点在中间，那么最短距离 = 最大值 - 最小值 + min(最大值 - A, A - 最小值)；



标程

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    freopen("take_out.in", "r", stdin);
    freopen("take_out.out", "w", stdout);
    int a, b, c, d;
    cin >> a >> b >> c >> d;
    int mi = min(min(a, b), min(c, d));
    int mx = max(max(a, b), max(c, d));
    long long ans = 0;
    if (a == mi) {
        ans = mx - a;
    } else if (a == mx) {
        ans = a - mi;
    } else {
        ans = (long long)(mx - mi) + min(a - mi, mx - a);
    }
    cout << ans << endl;
    return 0;
}
```

区间问题 I

题解

首先，给定 l, r ，如何快速计算 $\sum_{i=l}^r a_i^2 - \sum_{i=l}^r a_i - k \times (r - l + 1)$ ？

令 $sum(x) = \sum_{i=1}^x a_i^2 - \sum_{i=1}^x a_i - k \times x = \sum_{i=1}^x (a_i^2 - a_i - k)$ ，那么可以发现， $sum(x)$ 是 $a_i^2 - a_i - k$ 的前缀和，所以原式可以写作 $sum(\textcircled{R}) - sum(l - 1)$ 。

想要让 $sum(\textcircled{R}) - sum(l - 1)$ 尽可能大，应该最小化 $sum(l - 1)$ ，因此在计算 $sum(i)$ 时，维护 $minn = \min_{0 \leq j \leq i-1} \{sum(j)\}$ ，因此答案为： $\max\{sum(i) - minn\}$ 。

注意使用 `long long`，时间复杂度： $\Theta(n)$ 。

标程

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1000010;
long long sum[maxn];
int main() {
    freopen("interval.in", "r", stdin);
    freopen("interval.out", "w", stdout);
    int n, k;
    scanf("%d%d", &n, &k);
    long long ans = -1e18, minn = 0;
    for (int i = 1; i <= n; i++) {
        long long a;
        scanf("%lld", &a);
        a = a * a - a - k;
        sum[i] = sum[i - 1] + a;
        ans = max(ans, sum[i] - minn);
        minn = min(minn, sum[i]);
    }
    printf("%lld\n", ans);
    return 0;
}
```

涂色

题解

首先，如果这棵树的形态已经确定，那么只需要将与 1 相连的边全部染为红色，其他边可以为任意颜色，因此答案为 $g = 2^{n-1-k}$ ，其中 k 为树上和 1 相邻的节点个数。

原问题给定的是一个森林，不妨设该森林有 m 个连通块，且与 1 相邻的节点有 k 个，则最后可能的树中， g 的值为 2^{n-1-x} ，其中 $x \in [k, k + m - 1]$ 。

那么枚举所有 x 即可，使用快速幂加速计算 $2^{n-1-x} \% p$ 的结果。

时间复杂度 $\Theta(n \log n)$ 。

标程

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1000010;
int cnt, vis[maxn];
vector<int> G[maxn];
void dfs(int u) {
    if (vis[u]) {
        return;
    }
    vis[u] = 1;
    for (int i = 0; i < G[u].size(); i++) {
        int v = G[u][i];
        dfs(v);
    }
}
long long qpow(int n, int p) {
    long long res = 1, x = 2;
    while (n) {
        if (n & 1) {
            res = res * x % p;
        }
        x = x * x % p;
        n >>= 1;
    }
    return res;
}
int main() {
    freopen("paint.in", "r", stdin);
    freopen("paint.out", "w", stdout);
    int n, m, p;
    cin >> n >> m >> p;
    for (int i = 1; i <= m; i++) {
        int u, v;
        cin >> u >> v;
        G[u].push_back(v);
        G[v].push_back(u);
    }
    for (int i = 1; i <= n; i++) {
        if (!vis[i]) {
            cnt++;
            dfs(i);
        }
    }
    long long ans = 0;
    for (int i = 0; i <= cnt - 1; i++) {
        int x = G[1].size() + i;
        ans = max(ans, qpow(n - 1 - x, p));
    }
    cout << ans << endl;
    return 0;
}
```

区间问题 II

题解

结论：最终选择的区间中必然包含原序列的最大值。

证明：考虑调整。假设当前选择的区间 $[l, r]$ 中不包含原序列的最大值 mx ，如果 l 左边存在最大值，且最靠近 l 的位置在 l' ，将区间拓展为 $[l', r]$ （如果 l 左边不存在最大值，则最大值一定在 r 右边，证明方法类似。）

设 $[l, r]$ 的最大值为 mx' ，则拓展后总分值的变化量为：

$$mx \times (r - l' + 1) - mx' \times (r - l + 1) - \sum_{i=l'}^{l-1} a_i - (l' - l) = (mx - mx') \times (r - l + 1) + \sum_{i=l'+1}^{l-1} (mx - a_i - 1) + mx - a_{l'} - 1 = (mx$$

因为 $mx > mx'$, $l \leq r$, $mx > a_i$ ，因此前两项必然大于 0，这样调整是不劣的。

那么原式可以改写为最大化 $\sum_{l \leq i \leq r} mx - a_i - 1$ ，直接求最大子段和即可。

但是不要忙着开香槟，因为还应该保证选出的区间包含 mx ，即包含至少一个 -1，因此常用的分治法在此题中并不能用。

可以写出 $mx - a_i - 1$ 的前缀和 $sum[i]$ ，因为要包含最大值，所以可以通过预处理：

- 前缀： $pre[i] = \min_{0 \leq j < i} \{sum[j]\}$;
- 后缀： $suf[i] = \max_{i \leq j \leq n} \{sum[j]\}$;

那么最终答案为： $\max\{suf[i] - pre[i]\}$ ，其中 $1 \leq i \leq n$ ，并且 $a[i] = mx$ 。

时间复杂度： $\mathcal{O}(n)$

标程

```
#include <bits/stdc++.h>
using namespace std;
const long long inf = 4e18;
const int maxn = 1000010;
long long a[maxn], sum[maxn], pre[maxn], suf[maxn];
signed main() {
    freopen("interval.in", "r", stdin);
    freopen("interval.out", "w", stdout);
    int n;
    scanf("%d", &n);
    long long mx = 0;
    for (int i = 1; i <= n; i++) {
        scanf("%lld", &a[i]);
        mx = max(mx, (long long)a[i]);
    }
    for (int i = 1; i <= n; i++) {
        sum[i] = sum[i - 1] + (mx - a[i] - 1);
    }
    pre[1] = 0;
    for (int i = 2; i <= n; i++) {
        pre[i] = min(pre[i - 1], sum[i - 1]);
    }
    suf[n + 1] = -inf;
    for (int i = n; i >= 1; i--) {
        suf[i] = max(suf[i + 1], sum[i]);
    }
    long long ans = -inf;
    for (int i = 1; i <= n; i++) {
        if (a[i] == mx) {
            ans = max(ans, suf[i] - pre[i]);
        }
    }
    printf("%lld\n", ans);
    return 0;
}
```