信息学奥赛笔记14

前缀和测试T3，借教室和Oranżada

[U407641]拥挤的城市 https://www.luogu.com.cn/problem/U407641?contestId=158718

题目描述

有 n 个城市坐落在一条线上，每个城市的位置可表示为 a_i，确保这些城市的位置组成一个递增序列，即 $a_1 < a_2 < a_3 < \ldots < a_n$。
城市 x 和城市 y 之间的距离为 $\left| a_x - a_y \right|$。

你可以在这些城市间穿行，假设你当前在起始城市 x，要想到达终点城市 y，可有两种穿行规则：

规则一：直接从城市 x 到城市 y，花费 $\left| a_x - a_y \right|$ 体力；

规则二：先从城市 x 去到距离**最近的**城市，花费 1 点体力，之后再从距离城市 x 最近的城市到城市 y，此时同样需要符合两种穿行规则。

注：距离城市 1 最近的城市是城市 2，距离城市 n 最近的城市是 $n-1$。

题目中总共会给出 m 对起始城市和终点城市，需要你选取最优路线，求出 m 个最小体力花费结果。

输入格式

第一行两个正整数 n 和 m,代表有 n 个城市和 m 次查询；

第二行有 n 个正整数，代表 n 个城市的位置 a_i；

之后有 m 行，每行两个正整数，表示起始城市的序号 x 和终点城市的序号 y 。

输出格式

共 m 行，每行一个结果，表示从起始城市 a_x 到终点城市 a_y 花费的体力数。

样例 #1

样例输入 #1

```
5 5
0 8 12 15 20
1 4
1 5
3 4
3 2
5 1
```

样例输出 #1

```
3
8
1
4
14
```

提示

【样例解释】

对于第二组城市，从城市 1 到城市 5，可以从城市 1 到它最近的城市也就是城市 2, 花费 1 点体力；再从城市 2 到城市 3，花费 1 点体力；再从城市 3 到城市 4，花费 1 点体力；最后从城市 4 到城市 5，**由于城市 5 不是城市 4 的最近城市**，所以需要花费 $\mid 15-20 \mid=5$ 体力，因此最终需要花费 $1+1+1+5=8$ 点体力。

对于第五组城市，从城市 5 到城市 1，城市 4 是城市 5 距离最近的城市，花费1点体力；从城市 4 到城市 3，城市 3 是城市 4距离最近的城市，花费一点体力；从城市 3 到城市 2，城市 2 不是城市 3 距离最近的城市，需要花费 $\mid 12-8 \mid=4$ 点体力；从城市 2 到城市 1，需要花费 $\mid 8-0 \mid=8$ 点体力。 因此总计需要花费 $1+1+4+8=14$ 点体力。

【数据范围】

对于 100% 的数据，$1 \leq n \leq 10^7$,$1 \leq m \leq 10^6$。

> 对于这道题，我们一定要注意一个细节，那就是距离当前城市较近的城市，移动步数为1，什么叫距离当前城市较近的那个？除了最左边和最右边，每个城市相邻的城市有两个，如果从a[i] 到 a[i + 1]是距离最近的，那么从`i`到`i + 1`的移动消耗为1，那如果a[i + 1]距离最近的是a[i + 2]而不是a[i]从`i + 1`到`i`的移动消耗并不是1，这点需要关注。
>
> 那么同学们可以发现，从左往右走和从右往左走的效果是完全不一样的。
>
> 我们应该使用两个前缀和数组同时控制从左往右的前缀和和从右往左的前缀和。

```cpp
#include<bits/stdc++.h>
using namespace std;
long long n, m, x, y, cnt, ans;
int a[10000010+10], b[10000010+10], c[10000010+10];
int main() {
    cin >> n >> m;
    a[0] = INT_MIN;
    for (int i = 1; i <= n; i++){
        cin >> a[i];
        if (i == 1) continue;
        if (a[i] - a[i - 1] > a[i - 1] - a[i - 2]) {
           b[i] = b[i - 1] + a[i] - a[i - 1];
        } else {
           b[i] = b[i - 1] + 1;
        }
    }
    for (int i = n - 1; i > 0; i--){
        if (a[i + 1] - a[i] < a[i + 2] - a[i + 1]) {
           c[i] = c[i + 1] + 1;
```

```
        } else {
            c[i] = c[i + 1] + a[i + 1] - a[i];
        }
    }
    for (int i = 1; i <= m; i++){
        cin >> x >> y;
        if (x > y) {
            cout << c[y] - c[x] << endl;
        } else{
            cout << b[y] - b[x] << endl;
        }
    }
    return 0;
}
```

[PA2021] Oranżada https://www.luogu.com.cn/problem/P9045

题目描述

有一排共 n 瓶橙汁，其中第 i 瓶的品牌为 a_i。

你可以花费 1 个单位的的代价交换两瓶相邻的橙汁。

求最小代价使得最左边 k 瓶橙汁品牌两两不同。

输入格式

第一行，两个整数 n, k;

第二行，n 个整数 $a_1, a_2, \cdots, a_n$。

输出格式

一行，一个整数，若有解，输出最小代价；否则，输出 -1。

样例 #1

样例输入 #1

```
5 3
3 3 3 1 2
```

样例输出 #1

```
4
```

样例 #2

样例输入 #2

```
3 2
1 1 1
```

样例输出 #2

```
-1
```

提示

样例 #1 解释

最优方案为先交换位置 3 和 4 的瓶子、再交换位置 4 和 5 的瓶子，接着交换位置 2 和 3 的瓶子，最后交换位置 3 和 4 的瓶子，共 4 次操作。

样例 #2 解释

显然无解。

数据范围

对于 100% 的数据，$1 \leq k, a_i \leq n \leq 5 \times 10^5$。

> 这道题是一道思维题，需要用到桶。首先我们先思考什么情况下会导致这道题无解？是不是当饮料种类小于k的时候，无论怎么交换都无法满足，所以我们在这可以想到需要统计饮料出现的种类。那么接下来我们应该考虑一下，该如何使得移动次数更少？
>
> 对于一个位置靠左，并且之前从来没有出现过的饮料，我们把它移动到左边有重复的那个饮料的位置，这样做是最少的。
>
> 因为我们每次移动一定是要让这个橙汁离目标更近。所以一定最起码要保证，没有一瓶饮料在往右移动
>
> 那么移动的距离就是当前这个饮料的位置`i`减去上次出现重复的饮料`pos`。

```cpp
#include<bits/stdc++.h>
using namespace std;
bool a[500005];
int n, k, x, pos = 1, cnt;
long long ans;
int main() {
    cin >> n >> k;
    for(int i = 1; i <= n; i++){
        cin >> x;
        if(!a[x]){
            a[x] = 1;
            cnt++;
            ans += i - pos;
            pos++;
            if (cnt == k){
                cout << ans;
                return 0;
            }
```

```
        }
    }
    cout << -1;
    return 0;
}
```

[P1083][NOIP2012 提高组] 借教室https://www.luogu.com.cn/problem/P1083

题目描述

在大学期间，经常需要租借教室。大到院系举办活动，小到学习小组自习讨论，都需要向学校申请借教室。教室的大小功能不同，借教室人的身份不同，借教室的手续也不一样。

面对海量租借教室的信息，我们自然希望编程解决这个问题。

我们需要处理接下来 n 天的借教室信息，其中第 i 天学校有 r_i 个教室可供租借。共有 m 份订单，每份订单用三个正整数描述，分别为 d_j, s_j, t_j，表示某租借者需要从第 s_j 天到第 t_j 天租借教室（包括第 s_j 天和第 t_j 天），每天需要租借 d_j 个教室。

我们假定，租借者对教室的大小、地点没有要求。即对于每份订单，我们只需要每天提供 d_j 个教室，而它们具体是哪些教室，每天是否是相同的教室则不用考虑。

借教室的原则是先到先得，也就是说我们要按照订单的先后顺序依次为每份订单分配教室。如果在分配的过程中遇到一份订单无法完全满足，则需要停止教室的分配，通知当前申请人修改订单。这里的无法满足指从第 s_j 天到第 t_j 天中有至少一天剩余的教室数量不足 d_j 个。

现在我们需要知道，是否会有订单无法完全满足。如果有，需要通知哪一个申请人修改订单。

输入格式

第一行包含两个正整数 n, m，表示天数和订单的数量。

第二行包含 n 个正整数，其中第 i 个数为 r_i，表示第 i 天可用于租借的教室数量。

接下来有 m 行，每行包含三个正整数 d_j, s_j, t_j，表示租借的数量，租借开始、结束分别在第几天。

每行相邻的两个数之间均用一个空格隔开。天数与订单均用从 1 开始的整数编号。

输出格式

如果所有订单均可满足，则输出只有一行，包含一个整数 0。否则（订单无法完全满足）

输出两行，第一行输出一个负整数 -1，第二行输出需要修改订单的申请人编号。

样例 #1

样例输入 #1

```
4 3
2 5 4 3
2 1 3
3 2 4
4 2 4
```

样例输出 #1

```
-1
2
```

提示

【输入输出样例说明】

第 1份订单满足后，4天剩余的教室数分别为 $0, 3, 2, 3$。第 2 份订单要求第 2天到第 4 天每天提供3个教室，而第 3 天剩余的教室数为2，因此无法满足。分配停止，通知第2 个申请人修改订单。

【数据范围】

对于10%的数据，有$1 \leq n, m \leq 10$;

对于30%的数据，有$1 \leq n, m \leq 1000$;

对于 70%的数据，有$1 \leq n, m \leq 10^5$;

对于 100%的数据，有$1 \leq n, m \leq 10^6, 0 \leq r_i, d_j \leq 10^9, 1 \leq s_j \leq t_j \leq n$。

> 这道题的查询机制是，给定教室的数量，给定在哪些天使用。那么，从含义上来看，我们需要把从开始到结束天，这么写天的教室都减去使用数量个即可。这是暴搜的思想，但是我们发现，假如有极端的数据样例1 1 1000000，难道我们真的要循环这么多遍将教室-1吗?
>
> 这里就能够想到了，对区间要进行同时修改的算法是差分数组，我们用差分数组来记录教室剩余的数量，那么，到底哪个教室为止才是不可借的呢?
>
> 大家可以想想，订单是按顺序来的，那么假如说某个订单无法满足要求的时候，那么这个订单之后的所有订单都无法满足要求。
>
> 如果到达某个订单时要求可以完成，那么该订单前所有订单均可以满足。此时我们能想到，可以使用二分答案来做。

```cpp
#include<bits/stdc++.h>
using namespace std;
const int maxn = 1e6+5
int n, m, r[maxn], b[maxn], d[maxn], s[maxn], t[maxn];
bool check(int x){
    memset(b,0,sizeof(b));
    for(int i = 1; i <= n; i++) b[i] = r[i] - r[i-1];
    for(int i = 1; i <= x; i++){
        b[s[i]] -= d[i];
        b[t[i]+1] += d[i];
    }
    for(int i = 1; i <= n; i++){
        b[i] = b[i] + b[i-1];
        if(b[i] < 0) return false;
    }
    return true;
}
int main(){
    cin >> n >> m;
    for(int i = 1; i <= n; i++) cin >> r[i];
```

```
    for(int i = 1; i <= m; i++) cin >> d[i] >> s[i] >> t[i];
    int l = 0,r = m + 1;
    while(r - l != 1){
        int mid = l + ((r - l) >> 1);
        if(check(mid)) l = mid;
        else r = mid;
    }
    if(l == m) cout << 0;
    else cout << -1 << endl << l+1;
    return 0;
}
```