单调栈，单调队列，st 表

邓明

2023 年 4 月 25 日

1 单调栈

模板题https://www.luogu.com.cn/problem/P5788

【模板】单调栈

题目背景

模板题，无背景。
2019.12.12 更新数据，放宽时限，现在不再卡常了。

题目描述

给出项数为 n 的整数数列 $a_{1\ldots n}$。

定义函数 $f(i)$ 代表数列中第 i 个元素之后第一个大于 a_i 的元素的**下标**，即 $f(i)=\min_{i<j\le n,a_j>a_i}\{j\}$。若不存在，则 $f(i)=0$。

试求出 $f(1\ldots n)$。

输入格式

第一行一个正整数 n。

第二行 n 个正整数 $a_{1...n}$。

输出格式

一行 n 个整数 $f(1...n)$ 的值。

样例 1

样例输入 1

5
1 4 2 3 5

样例输出 1

2 5 4 5 0

提示【数据规模与约定】

对于 30% 的数据，$n \leq 100$；
对于 60% 的数据，$n \leq 5 \times 10^3$ ；
对于 100% 的数据，$1 \leq n \leq 3 \times 10^6$，$1 \leq a_i \leq 10^9$。

这道题概括一下就是求每个数向右最近的大于这个数的下标。朴素的想法是对于每一个数向右遍历找到大于它的数就存储下标，最坏的情况是 $O(n^2)$ 的。

我们需要想一个办法，在遍历到每一个元素的时候就已经可以找到最近的大于它的元素的下标，这里我们可以使用栈来进行维护。

这个栈里面只存在单调的数的下标，满足：

栈里的元素从上到下的值为 x_i，则 $x_i > x_{i+1}$ 且 $a_{x_i} > a_{x_{i+1}}$

这样就保证了我们每次遍历到一个数的时候栈顶就已经是比它大的值的下标，并且离它最近。

我的答案：

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
ll n;
ll a[3000010];
ll f[3000010];
int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>a[i];
    }
    stack<ll> s;
    s.push(0);
    for(ll i=n;i>=1;i--){
        while(a[s.top()]<=a[i]&&s.top()!=0)s.pop();
        f[i]=s.top();
        s.push(i);
    }
    for(int i=1;i<=n;i++){
        cout<<f[i]<<" ";
    }
```

```
    cout<<endl;
    return 0;
}
```

例如 1 4 2 3 5 我们反着题目来，从左边找到最近大于的数的下标

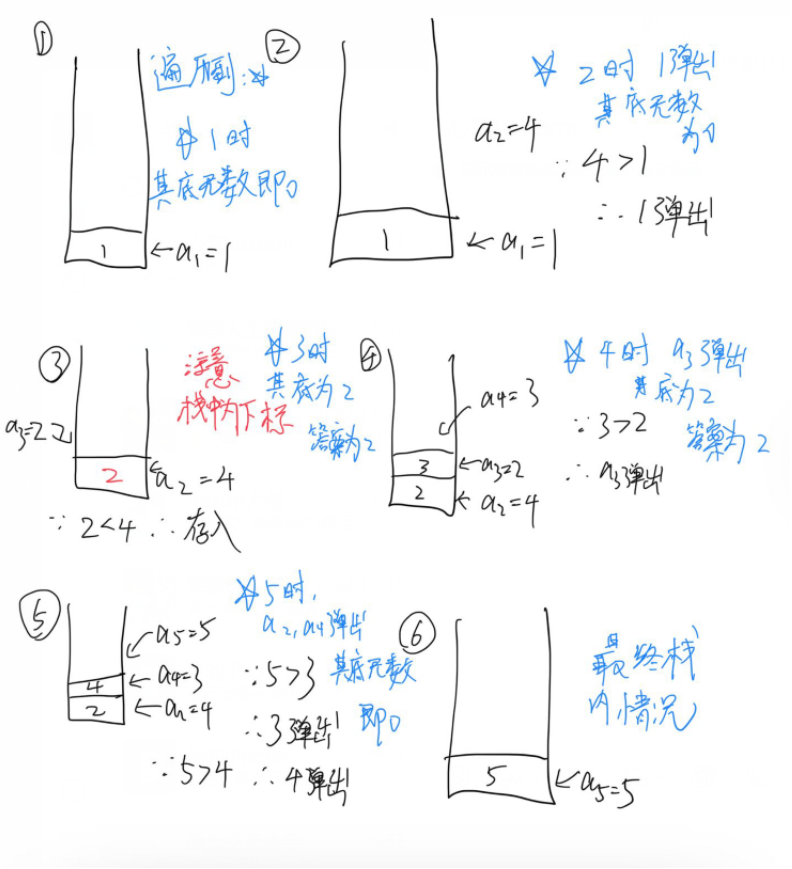


图 1: 栈内存放示意图

练习题https://www.luogu.com.cn/problem/P1901

~~实际上单调栈的题一般不难~~

发射站

题目描述

某地有 N 个能量发射站排成一行，每个发射站 i 都有不相同的高度 H_i，并能向两边（两端的发射站只能向一边）同时发射能量值为 V_i 的能量，发出的能量只被两边**最近的且比它高**的发射站接收。显然，每个发射站发来的能量有可能被 0 或 1 或 2 个其他发射站所接受。

请计算出接收最多能量的发射站接收的能量是多少。

输入格式

第 1 行一个整数 N。

第 2 到 $N+1$ 行，第 $i+1$ 行有两个整数 H_i 和 V_i，表示第 i 个发射站的高度和发射的能量值。

输出格式

输出仅一行，表示接收最多能量的发射站接收到的能量值。答案不超过 32 位带符号整数的表示范围。

样例 1

样例输入 1

3

4 2

3 5

6 10

样例输出 1

7

提示

对于 40% 的数据，$1 \leq N \leq 5000, 1 \leq H_i \leq 10^5, 1 \leq V_i \leq 10^4$。
对于 70% 的数据，$1 \leq N \leq 10^5, 1 \leq H_i \leq 2 \times 10^9, 1 \leq V_i \leq 10^4$。
对于 100% 的数据，$1 \leq N \leq 10^6, 1 \leq H_i \leq 2 \times 10^9, 1 \leq V_i \leq 10^4$。

2 单调队列

模板题https://www.luogu.com.cn/problem/P1886

滑动窗口 /【模板】单调队列

题目描述

有一个长为 n 的序列 a，以及一个大小为 k 的窗口。现在这个从左边开始向右滑动，每次滑动一个单位，求出每次滑动后窗口

中的最大值和最小值。

输入格式

输入一共有两行，第一行有两个正整数 n,k。第二行 n 个整数，表示序列 a

输出格式

输出共两行，第一行为每次窗口滑动的最小值
第二行为每次窗口滑动的最大值

样例 1

样例输入 1

8 3

1 3 -1 -3 5 3 6 7

样例输出 1

-1 -3 -3 -3 3 3

3 3 5 5 6 7

提示

【数据范围】
对于 50% 的数据，$1 \leq n \leq 10^5$；对于 100% 的数据，$1 \leq k \leq n \leq 10^6$，$a_i \in [-2^{31}, 2^{31})$。

单调队列和单调栈很像，区别在于单调队列从前端取值。一样的想法，我们想要在遍历到 i 时就已经知道该区间最大最小的信息

这个想法可以用队列维护，我们维护一个单调的队列，每次队首就是最大最小值，为了保证最大最小值，我们像单调栈一样，大于（小于）的就从队首弹出去，由于弹出去的一定比加进来的小（大），因此最大（最小）值一定不是他可以忽略。

等到 $k-1$ 都加入队列了之后查看头部值，如果头部值的下标已经在我们区间外了，那么后面的区间用不上，直接舍去即可

具体的，以区间最小值为例设：

设单调队列从头部开始的元素为 x_i, 则有 $x_i < x_{i+1}$，且 $a_{x_i} < a_{x_{i+1}}$

我的答案:

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
ll n,k;
ll a[2000010];
ll demx[2000010];
ll demi[2000010];
ll ansmx[2000010];
ll ansmi[2000010];
int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    cin>>n>>k;
    for(int i=1;i<=n;i++){
        cin>>a[i];
    }
    ll s = 0;
    ll t = 0;
    for(int i=1;i<=n;i++){
```

```
        while(s<t&&a[demx[t-1]]<=a[i])t--;
        demx[t++]=i;
        if(i>=k) ansmx[i-k]=a[demx[s]];
        if(i-demx[s]==k-1){
            s++;
        }
    }
    s=0;
    t=0;
    for(int i=1;i<=n;i++){
        while(s<t&&a[demi[t-1]]>=a[i])t--;
        demi[t++]=i;
        if(i>=k) ansmi[i-k]=a[demi[s]];
        if(i-demi[s]==k-1){
            s++;
        }
    }
    for(int i=0;i<=n-k;i++){
        cout<<ansmi[i]<<" ";
    }
    cout<<endl;
    for(int i=0;i<=n-k;i++){
        cout<<ansmx[i]<<" ";
    }
    cout<<endl;
    return 0;
}
```

2.1 多重背包单调队列优化

我们先来看看多重背包的题

https://www.luogu.com.cn/problem/P1776

宝物筛选

题目描述

终于，破解了千年的难题。小 FF 找到了王室的宝物室，里面堆满了无数价值连城的宝物。

这下小 FF 可发财了，嘎嘎。但是这里的宝物实在是太多了，小 FF 的采集车似乎装不下那么多宝物。看来小 FF 只能含泪舍弃其中的一部分宝物了。

小 FF 对洞穴里的宝物进行了整理，他发现每样宝物都有一件或者多件。他粗略估算了下每样宝物的价值，之后开始了宝物筛选工作：小 FF 有一个最大载重为 W 的采集车，洞穴里总共有 n 种宝物，每种宝物的价值为 v_i，重量为 w_i，每种宝物有 m_i 件。小 FF 希望在采集车不超载的前提下，选择一些宝物装进采集车，使得它们的价值和最大。

输入格式

第一行为一个整数 n 和 W，分别表示宝物种数和采集车的最大载重。

接下来 n 行每行三个整数 v_i, w_i, m_i。

输出格式

输出仅一个整数，表示在采集车不超载的情况下收集的宝物

的最大价值。

样例 1

样例输入 1

4 20

3 9 3

5 9 1

9 4 2

8 1 3

样例输出 1

47

提示

对于 30% 的数据，$n \leq \sum m_i \leq 10^4$，$0 \leq W \leq 10^3$。

对于 100% 的数据，$n \leq \sum m_i \leq 10^5$，$0 \leq W \leq 4 \times 10^4$，$1 \leq n \leq 100$。

经典多重背包问题，如果我们用朴素的方法:

$dp[i][j]$:= 到第 i 个物品为止总重量不超过 j 的所有选法中最大可能的价值

那么有:

$$dp[i+1][j] = max\{dp[i][j - k \times w[i]] + k \times v[i]\} \qquad (1)$$

且

$$0 \le k \le m_i \quad \&\& \quad j - k \times w[i] \ge 0 \tag{2}$$

这个式子显然超时，我们要想办法优化，有人就通过推柿子发现他能用单调队列优化...

观察这个式子，我们先想办法吧 $k \times w[i]$ 消掉。

由于

$$j \to j - k \times w[i] \tag{3}$$

那么有

$$(j + k) \times w[i] \to j \times w[i] \tag{4}$$

这样我们消掉了 $k \times w[i]$, 化简一下，设 $a[j] = dp[i][j \times w[i]]$，我们得到

$$dp[i+1][(j+k) \times w[i]] = \max\{a[j] + k \times v[i], a[j+1] + (k-1) \times v[i], \cdots, a[j+k]\} \tag{5}$$

这个时候我们已经可以隐约地看见区间的影子了，但是还是有一个 $k \times v[i]$ 干扰，我们可以像之前一样把它消了:$b[j] = a[j] - j \times v[i]$

这样我们得到：

$$dp[i+1][(j+k) \times w[i]] = \max\{b[j], b[j+1], \cdots, b[j+k]\} + (j+k) \times v[i] \tag{6}$$

可以发现这就是一个 b 数组的滑动最大值，这个地方就可以用单调队列优化。

但是这种做法存在一个小问题，我们这样只给 $w[i]$ 的倍数转移了，但是问题不大，我们可以遍历 $w[i]$ 的余数来将所有情况考虑进去。

我的答案:

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
```

```
ll dp[2000010];
ll n,W;
ll m[200010];
ll v[200010];
ll w[200010];
ll de[200010];
ll dev[200010];
int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);
    cin>>n>>W;
    for(int i=1;i<=n;i++){
        cin>>v[i]>>w[i]>>m[i];
    }
    //dp[]
    for(int i=1;i<=n;i++){
        for(int mod = 0;mod<w[i];mod++){
            ll s=0;
            ll t =0;
            for(int j=0;j*w[i]+mod<=W;j++){
                //这里的j实际上是j+k
                ll val = dp[j*w[i]+mod] - j*v[i];
                while(s<t&&dev[t-1]<=val)t--;
                de[t]=j;
                dev[t++]=val;
                dp[j*w[i]+mod] = dev[s]+j*v[i];
                if(j-de[s]==m[i]) {
                    s++;
                }
```

```
            }
        }
    }
    cout<<dp[W]<<endl;
    return 0;
}
```

练习题https://www.luogu.com.cn/problem/P2254

[NOI2005] 瑰丽华尔兹

题目背景

你跳过华尔兹吗？当音乐响起，当你随着旋律滑动舞步，是不是有一种漫步仙境的惬意？

众所周知，跳华尔兹时，最重要的是有好的音乐。但是很少有几个人知道，世界上最伟大的钢琴家一生都漂泊在大海上，他的名字叫丹尼　布德曼　T.D.　柠檬　1900，朋友们都叫他 1900。

1900 在 20 世纪的第一年出生在往返于欧美的邮轮弗吉尼亚号上。很不幸，他刚出生就被抛弃，成了孤儿。1900 孤独的成长在弗吉尼亚号上，从未离开过这个摇晃的世界。也许是对他命运的补偿，上帝派可爱的小天使艾米丽照顾他。可能是天使的点化，1900 拥有不可思议的钢琴天赋：从未有人教，从没看过乐谱，但他却能凭着自己的感觉弹出最沁人心脾的旋律。当 1900 的音乐获得邮轮上所有人的欢迎时，他才 8 岁，而此时，他已经乘着海轮往返欧美大陆 50 余次了。

虽说是钢琴奇才，但 1900 还是个孩子，他有着和一般男孩一样的好奇和调皮，只不过更多一层浪漫的色彩罢了：这是一个风雨交加的夜晚，海风卷起层层巨浪拍打着弗吉尼亚号，邮轮随着巨浪剧烈的摇摆。船上的新萨克斯手迈克斯　托尼晕船了，1900 招呼

托尼和他一起坐到舞厅里的钢琴上，然后松开了固定钢琴的闸，于是，钢琴随着海轮的倾斜滑动起来。准确的说，我们的主角 1900…

题目描述

不妨认为舞厅是一个 N 行 M 列的矩阵，矩阵中的某些方格上堆放了一些家具，其他的则是空地。钢琴可以在空地上滑动，但不能撞上家具或滑出舞厅，否则会损坏钢琴和家具，引来难缠的船长。每个时刻，钢琴都会随着船体倾斜的方向向相邻的方格滑动一格，相邻的方格可以是向东、向西、向南或向北的。而艾米丽可以选择施魔法或不施魔法：如果不施魔法，则钢琴会滑动；如果施魔法，则钢琴会原地不动。

艾米丽是个天使，她知道每段时间的船体的倾斜情况。她想使钢琴在舞厅里滑行的路程尽量长，这样 1900 会非常高兴，同时也有利于治疗托尼的晕船。但艾米丽还太小，不会算，所以希望你能帮助她。

输入格式

输入文件的第一行包含 5 个数 N, M, x, y 和 K。N 和 M 描述舞厅的大小，x 和 y 为钢琴的初始位置；我们对船体倾斜情况是按时间的区间来描述的，且从 1 开始计算时间，比如“在 $[1, 3]$ 时间里向东倾斜，$[4, 5]$ 时间里向北倾斜”，因此这里的 K 表示区间的数目。

以下 N 行，每行 M 个字符，描述舞厅里的家具。第 i 行第 j 列的字符若为‘.’，则表示该位置是空地；若为‘x‘，则表示有家具。

以下 K 行，顺序描述 K 个时间区间，格式为：$s_i t_i d_i (1 \le i \le K)$。表示在时间区间 $[s_i, t_i]$ 内，船体都是向 d_i 方向倾斜的。d_i 为 1, 2, 3, 4 中的一个，依次表示北、南、西、东（分别对应矩阵中的上、下、左、右）。输入保证区间是连续的，即

$s_1 = 1$

$s_i = t_{i-1} + 1(1 < i \leq K)$
$t_K = T$

输出格式

输出文件仅有 1 行，包含一个整数，表示钢琴滑行的最长距离 (即格子数)。

样例 1

样例输入 1

4 5 4 1 3
..xx.
.....
...x.
.....
1 3 4
4 5 1
6 7 3

样例输出 1

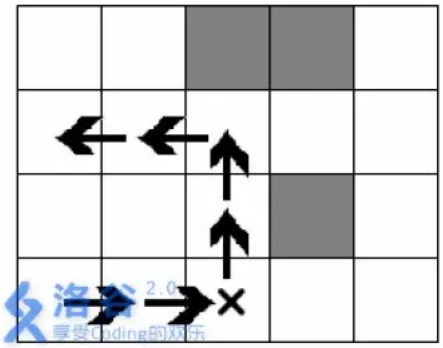
6

提示

钢琴的滑行路线：
钢琴在“×”位置上时天使使用一次魔法，因此滑动总长度为 6。

【数据范围】
50% 的数据中，$1 \leq N, M \leq 200$ $T \leq 200$；

100% 的数据中，$1 \leq N, M \leq 200$，$K \leq 200$，$T \leq 40000$。



3 ST 表

ST 表是用于解决可重复贡献问题的数据结构，所谓可重复贡献，例如 $\max(x, x) = x$, 以及 $\gcd(x, x) = x$ 这一类问题，和线段树以及树状数组类似，其必须满足结合律。

模板题https://www.luogu.com.cn/problem/P3865

【模板】ST 表

题目背景

这是一道 ST 表经典题——静态区间最大值

请注意最大数据时限只有 0.8s，数据强度不低，请务必保证你的每次查询复杂度为 $O(1)$。若使用更高时间复杂度算法不保证能通过。

如果您认为您的代码时间复杂度正确但是 TLE，可以尝试使用快速读入：

```
inline int read()
```

```
{
int x=0,f=1;char ch=getchar();
while (ch<'0'||ch>'9'){if (ch=='-') f=-1;ch=getchar();}
while (ch>='0'&&ch<='9'){x=x*10+ch-48;ch=getchar();}
return x*f;
}
```

函数返回值为读入的第一个整数。

快速读入作用仅为加快读入，并非强制使用。

题目描述

给定一个长度为 N 的数列，和 M 次询问，求出每一次询问的区间内数字的最大值。

输入格式

第一行包含两个整数 N, M，分别表示数列的长度和询问的个数。

第二行包含 N 个整数（记为 a_i），依次表示数列的第 i 项。

接下来 M 行，每行包含两个整数 l_i, r_i，表示查询的区间为 $[l_i, r_i]$。

输出格式

输出包含 M 行，每行一个整数，依次表示每一次询问的结果。

样例 1

样例输入 1

8 8
9 3 1 7 5 6 0 8

1 6
1 5
2 7
2 6
1 8
4 8
3 7
1 8

样例输出 1

9
9
7
7
9
8
7
9

提示

对于 30% 的数据，满足 $1 \leq N, M \leq 10$。

对于 70% 的数据，满足 $1 \leq N, M \leq 10^5$。

对于 100% 的数据，满足 $1 \leq N \leq 10^5$，$1 \leq M \leq 2 \times 10^6$，$a_i \in [0, 10^9]$，$1 \leq l_i \leq r_i \leq N$。

ST 表基于倍增的思想，这里先简单介绍一下倍增，其主要思想是考虑预处理本体倍增，以倍增的 $\log(n)$ 时间来优化 n 的时间。

抽象的说例如 1 2 3 4 5 6 7 8 我们遍历 n 的即为从 1 遍历到 8，而使用倍增思想则是遍历 1 2 4 8，这样是 $\log(n)$ 的遍历。

具体地说，例如 LCA（最近公共祖先），我们也是用倍增法的思想，不是一个一个向上遍历。

首先对于任意顶点 v,利用其父亲节点信息,可以通过 $parent2[v] = parent[parent[v]]$ 向上走到其两步地顶点，再利用 $parent4[v] = parent2[parent2[v]]$ 得到其向上走 4 步的顶点，以此类推，可以得到其向上走 2^k 步的顶点 $parent[k][v]$, 然后二分找到公共祖先。

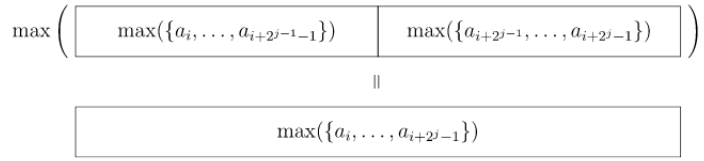
ST 表能够实现除了倍增节省时间以外，还因为其的可重复贡献性，也就是说我把区间分为两倍，前半倍和后半倍合并并不会影响最终结果，并且节省了时间。

具体实现如下（这里抄一下 oiwiki）

令 $f(i,j)$ 表示区间 $[i, i+2^j-1]$ 的最大值。

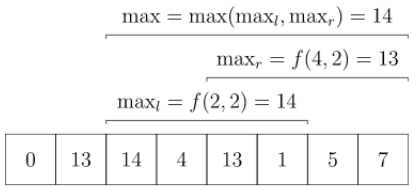
显然 $f(i,0) = a_i$。

根据定义式，第二维就相当于倍增的时候「跳了 2^j-1 步」，依据倍增的思路，写出状态转移方程：$f(i,j) = \max(f(i,j-1), f(i+2^{j-1}, j-1))$。



以上就是预处理部分。而对于查询，可以简单实现如下：

对于每个询问 $[l,r]$，我们把它分成两部分：$f[l, l+2^s-1]$ 与 $f[r-2^s+1, r]$，其中 $s = \lfloor \log_2(r-l+1) \rfloor$。两部分的结果的最大值就是回答。



根据上面对于「可重复贡献问题」的论证，由于最大值是「可重复贡献问题」，重叠并不会对区间最大值产生影响。又因为这两个区间完全覆盖了 $[l,r]$，可以保证答案的正确性。

注意点

1. 最好使用快读，取消标准输入输出不太好使，（洛谷过不了）
2. 最好是自行预处理 $\log 2$ 的值，stl 的库比较慢。

我的答案：

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
ll logn=21;
ll Logn[2000010];
ll f[2000010][25];
ll dp[2000010][25];
ll n,m;
inline int read() {  // 快读
    char c = getchar();
    int x = 0, f = 1;
    while (c < '0' || c > '9') {
        if (c == '-') f = -1;
        c = getchar();
    }
    while (c >= '0' && c <= '9') {
        x = x * 10 + c - '0';
        c = getchar();
    }
    return x * f;
}
```

```
void init(){
    Logn[1]=0;
    Logn[2]=1;
    for(int i=3;i<=2000000;i++){
        Logn[i]=Logn[i/2]+1;
    }
}

int main() {

    n=read(),m=read();
    for(int i=1;i<=n;i++){f[i][0]=read();}
    init();
    for(int j=1;j<=logn;j++){
        for(int i=1;i<=n;i++){
            //f[i][j] 表示i 到 i+(1<<j)-1  所以这里不用+1
            f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
        }
    }
    for(int i=1;i<=m;i++){
        ll l,r;
        l=read(),r=read();
        ll s = Logn[r-l+1];
        //由于这里要+1 才能到s所以 后半区间第一值下标要加一
        ll ans = max(f[l][s],f[r-(1<<(s))+1][s]);
        printf("%d\n",ans);
    }
    return 0;
```

```
}
```

练习题：https://www.luogu.com.cn/problem/P2880

[USACO07JAN] Balanced Lineup G

题目描述

For the daily milking, Farmer John's N cows (1 N 50,000) always line up in the same order. One day Farmer John decides to organize a game of Ultimate Frisbee with some of the cows. To keep things simple, he will take a contiguous range of cows from the milking lineup to play the game. However, for all the cows to have fun they should not differ too much in height.

Farmer John has made a list of Q (1 Q 180,000) potential groups of cows and their heights (1 height 1,000,000). For each group, he wants your help to determine the difference in height between the shortest and the tallest cow in the group.

每天, 农夫 John 的 $n(1 \leq n \leq 5 \times 10^4)$ 头牛总是按同一序列排队。

有一天, John 决定让一些牛们玩一场飞盘比赛。他准备找一群在队列中位置连续的牛来进行比赛。但是为了避免水平悬殊，牛的身高不应该相差太大。John 准备了 $q(1 \leq q \leq 1.8 \times 10^5)$ 个可能的牛的选择和所有牛的身高 $h_i(1 \leq h_i \leq 10^6, 1 \leq i \leq n)$。他想知道每一组里面最高和最低的牛的身高差。

输入格式

Line 1: Two space-separated integers, N and Q.

Lines 2..N+1: Line i+1 contains a single integer that is the

height of cow i

Lines N+2..N+Q+1: Two integers A and B (1 A B N), representing the range of cows from A to B inclusive.

第一行两个数 n, q。

接下来 n 行，每行一个数 h_i。

再接下来 q 行，每行两个整数 a 和 b，表示询问第 a 头牛到第 b 头牛里的最高和最低的牛的身高差。

输出格式

Lines 1..Q: Each line contains a single integer that is a response to a reply and indicates the difference in height between the tallest and shortest cow in the range.

输出共 q 行，对于每一组询问，输出每一组中最高和最低的牛的身高差。

样例 1

样例输入 1

```
6 3
1
7
3
4
2
5
1 5
4 6
```

2 2

样例输出 1

6
3
0