前缀和、差分

选自东方博宜和《信息学竞赛金牌导航》

2060 - 计算能力（区间求和）

题目描述

小 A 同学有着很强的计算能力，张老师为了检验小 A 同学的计算能力，写了一个 n 个数的数列。

张老师问了小 A 同学 m 个问题，每个问题都是请小 A 同学计算这个数列从第 x 个数开始到第 y 个数结束这个区间的所有数的和。

请你编程帮助张老师计算出结果。

输入

第一行包含两个整数 n 和 m 。

第二行包含 n 个整数。

接下来 m 行，每行包含两个整数 x 和 y 表示。

数据范围

$1 \leqslant x \leqslant y \leqslant n$。

$1 \leqslant n,m \leqslant 100000$。

数列中元素的值，在 [-1000,1000] 的范围内。

输出

共 m 行，每行输出一个询问的结果。

样例

输入

6 3

1 3 6 5 4 2

1 3

2 5

3 6

输出

10

18

17

```cpp
#include<bits/stdc++.h>//2060:【入门】计算能力（区间求和）
using namespace std;
const int N =100010;
int a[N],b[N];//a 数组表示读入的值，b 数组代表前缀和
int n,m;
int main()
{
	scanf("%d%d",&n,&m);// 读入 n 个数
	for(int i=1;i<=n;i++)
	{
		scanf("%d",&a[i]);
		b[i]= b[i-1] + a[i];// 求前缀和
	}

	int x,y;// 读入 m 个问题
	for(int i=1;i <= m;i++)
	{
		scanf("%d%d",&x,&y);
		printf("%d\n",b[y]-b[x-1]);
	}
	return 0;
}
```

2061 - 子矩阵求和

小 A 同学有着很强的计算能力，张老师为了检验小 A 同学的计算能力，写了一个 n 行 m 列的矩阵数列。

张老师问了小 A 同学 k 个问题，每个问题会先告知小 A 同学 4 个数 x_1,y_1,x_2,y_2 表示这是矩阵中 2 个点的行列的值，以这两个点为一个矩形的左上角和右下角，可以从矩阵中画出一个子矩阵，张老师请小 A 同学计算出这个子矩阵中所有数的和。

请你编程帮助张老师计算出结果。

输入：

第一行包含三个整数 n，m，k 。

接下来 n 行，每行包含 m 个整数。

接下来 k 行，每行包含四个整数 x_1, y_1, x_2, y_2，表示一组询问。

数据范围

$1 \leqslant n,m \leqslant 1000$。

$1 \leqslant k \leqslant 200000$。

$1 \leqslant x\_1 \leqslant x\_2 \leqslant n, 1 \leqslant y\_1 \leqslant y\_2 \leqslant m$。

矩阵内元素的值均在 [-1000,1000] 的范围内。

输出 ： 共 k 行，每行输出一个询问的结果。

样例

输入

3 5 4

1 1 6 7 4

6 10 4 9 9

2 6 7 3 7

1 2 2 4

2 4 3 5

2 2 3 5

1 3 2 4

输出

37

28

55

26

```cpp
#include <bits/stdc++.h>//2061:【入门】子矩阵求和（二维数组前缀和）（区间求和）
using namespace std;
const int N=1010;//a 代表读入的数

//b 代表前缀和
int a[N][N],b[N][N];
int n,m,k;

int main()
{
	scanf("%d%d%d" ,&n, &m,&k);
	for(int i=1;i <= n;i++)// 读入二维数组
	{
		for(int j= 1;j<= m;j++)
		{
			scanf("%d",&a[i][j]);
			b[i][j]=b[i-1][j]+ b[i][j-1]-b[i-1][j-1] + a[i][j];// 求前缀和
		}
	}

	int x1,y1,x2,y2;
	for(int i=1;i <= k;i++)// 读入 k 次询问
	{
		scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
		printf("%d\n",b[x2][y2]-b[x1-1][y2]-b[x2][y1-1]+b[x1-1][y1-1]);// 输出区间和
	}
	return 0;
}
```

2119 - 任务的最少完成时间

题目描述

小 A 同学接到了 n 个需要完成的任务，这 n 个任务必须按照接到的顺序完成，每个任务的完成时间为 a_i。

由于任务非常艰巨，小 A 同学从老师那里领到了一张减负卡，用这张卡，小 A 可以从 n 个任务中任意的删除 k 个连续的任务，只需要完成剩余的任务。

请问，小 A 完成所有任务的总时间最少是多少？

输入

第 1 行，有两个整数 n 和 k（1 ≤ n ≤ 10^6，0 ≤ k ≤ 10^6）。（10 的 6 次方）

接下来有 n 个整数，每个整数 a_i 表示每个任务的完成时间。（1 ≤ a_i ≤ 10^{12}）（10 的 12 次方）

输出

一个整数，表示小 A 任务完成的最少时间。

样例

输入

5 2

1 3 2 5 4

输出

6

```cpp
#include <bits/stdc++.h>//2119    xuegirl     任务的最少完成时间（区间求和）
using namespace std;
const int N = 1e6 + 100;
long long a[N], b[N];

int main()
{
	long long n, k, i,max=0,sum=0;
	scanf("%lld%lld", &n, &k);
	for (i = 1 ; i <= n ; i ++)
	{
		scanf("%lld", &a[i]);
		sum+=a[i];
				//计算项与项之间的差值
		b[i] = a[i] + b[i - 1];
				//cout<<b[i]<<endl;
	}

	for(i=1;i<=n-k+1;i++)
	{
		if(max<b[i+k-1]-b[i-1])
		max=b[i+k-1]-b[i-1];
	}

	printf("%lld", sum-max);
	return 0;
}
```

2105 - 不太甜的糖果

小 Y 走啊走啊，翻山越岭、跋山涉水，终于，小 Y 累了。虽然，糖果的诱惑强大，但他的两条腿已经不听使唤，只能坐在地上叹气，内心无比焦急……

突然眼前一黑，小 Y 没有昏过去，但是眼前出现了一个糖人。在这无人之地，小 Y 没有别的办法，只得求助糖人。

善良的糖人没法拒绝小 Y 的请求，但还要遵守这个世界的规则，所以，小 Y 不能”不劳而获”，但小 Y 现在已经没有力气。糖人只让他玩一个小小的游戏，完成这个游戏，小 Y 才能获得补充能量，继续前进。

但是，小 Y 满脑子都是糖果，他没有心思玩游戏，只想着吃糖。所以，他向你求助。

游戏的规则是这样的： 给定一排长度为 n 的糖果串，每个糖果有一个甜度；在不改变糖果顺序的前提下，求出一个最短的糖果串使得它的甜度之和大于等于 m 。

输入

第一行包含两个数 n 和 m，第二行有 n 个数。

输出

输出一行，包含一个数，即最短的糖果串的长度；如果找不到这样的糖果串，输出 0。

样例

输入

10 15

5 1 3 5 10 7 4 9 2 8

输出

2

【样例说明】

糖果串为连续的。对于样例数据，选第四五个可以达到 15 或者第五六个能达到 17 ，所以最短糖果串为 2 。

【数据范围】

对于 20\% 的数据，n ≤ 200

对于 50\% 的数据，n ≤ 2000

对于 80\% 的数据，n ≤ 100000

对于 90\% 的数据，n ≤ 200000

对于 100\% 的数据，n ≤ 230000

1 ≤ M ≤ 1000，糖果的甜度为正整数，且本题数据保证连续若干糖果的甜度和在 int 范围内。

本题的核心问题：在长度为 n 的数组中，如果能求到连续 k 个数的和 s，使得总和 s>=m，问：k 的值最小是多少。

大家不难想到穷举 k 的范围，如果穷举 k 的范围从每个数开始求连续 k 个数的和，采用三重循环求解的话，本题的数据量，1 秒是计算不完的。

```
// 这个思路时间不够
for(穷举 k 的范围)
{
    for(穷举从每个数开始的位置)
    {
        for(循环连续 k 个数，求和){
        }
    }
}
```

优化思路：

1. k 的范围可以二分；

2. 从每个数开始求连续 k 个数的和，可以用前缀和求区间和，也可以简单一点，递推动归求解。

不难想到：从第 i 个数开始连续 k 个数的和 = 从第 i-1 个数开始连续 k 个数的和 - a[i-1] + a[i+k-1]。

因此，如果 f[i] 表示求从第 i 个数开始连续 k 个数的和，那么 f[i] = f[i-1] - a[i-1] + a[i + k - 1];

通过上述优化，使得三重循环优化到二分 + 嵌套一个 for 循环，最多循环次数大概是：2 * 106，这样就能解决问题了。

参考代码如下：

```cpp
#include<bits/stdc++.h>//2105-1      javacn  不太甜的糖果 （区间求和）
using namespace std;

const int N = 230010;
int a[N];
int f[N];// 从每个数开始求连续 mid 个数的和
int n,m;

// 检验：连续 mid 个数的和是否 >=m
bool check(int k)
{
	//f[i] 代表从第 i 个数开始连续 k 个数的和
	// 先求出连续 k 个数的和
	memset(f,0,sizeof(f));
	for(int i = 1;i <= k;i++)
	{
		f[1] = f[1] + a[i];
	}

	if(f[1] >= m) return true;

	// 求从第 2 个数开始连续 k 个数的和
	for(int i = 2;i <= n - k + 1;i++)
	{
		f[i] = f[i-1] - a[i-1] + a[i + k - 1];
		if(f[i] >= m) return true;
	}

	return false;
}
```

```
int main()
{
	cin>>n>>m;
	for(int i = 1;i <= n;i++)
	{
		cin>>a[i];
	}

	// 二分可能长度
	int l = 1,r = n,mid;
	while(l <= r)
	{
		mid = l + r >> 1;// 二进制位移运算，本语句等于 mid = （l + r ）/2
		// 如果连续 mid 个数的和 >=m，缩小 mid
		if(check(mid))	r = mid - 1;
		else		l = mid + 1;
	}

	if(l == n + 1)	cout<<0;
	else		cout<<l;
}
```

```
#include<bits/stdc++.h>//2105-2    javacn   二分+前缀和求解      不太甜的糖果
using namespace std;//（区间求和）

/*
有 n 个数，求其中最短的区间，使其区间和 >=m。
答案范围：[1,n]。
*/
const int N = 2.3e5 + 10;
int s[N];// 所有数的前缀和
int n,m,l,r,mid,x;

// 检验如果区间长度为 mid，是否存在一个区间和 >=m
bool check(int mid)
{
    // 枚举区间的起点
    // 从第 1 个数枚举到倒数第 mid 个数
      for(int i = 1;i <= n - mid + 1;i++)
      {
            //[i,i+mid-1]
            if(s[i+mid-1] - s[i-1] >= m)        return true;
      }

      return            false;
}
```

```
int main()
{
	scanf("%d%d",&n,&m);

	for(int i = 1;i <= n;i++)
	{
		scanf("%d",&x);
		s[i] = s[i-1] + x;// 前缀和
	}

	l = 1,r = n;
	while(l <= r)
	{
		mid = l + r >> 1;// 二进制位移运算，本语句等于 mid = （l + r ）/2
		if(check(mid)) r = mid - 1;
		else l = mid + 1;
	}

	if(l == n + 1) cout<<0;
	else cout<<l;
	return 0;
}
```

```cpp
#include<bits/stdc++.h>//2105-3    javacn  双指针求解      不太甜的糖果（区间求和）
using namespace std;
const int N = 2.3e5 + 10;
int s[N];// 所有数的前缀和
int n,m,l,r,x,ans;

int main()
{
	scanf("%d%d",&n,&m);

	for(int i = 1;i <= n;i++)
	{
		scanf("%d",&x);
		s[i] = s[i-1] + x;// 前缀和
	}

	//i 表示区间起点，r 停留在使得 [i,r] 区间和 >=m 的第 1 个位置
	r = 1;
	ans = n + 1;
	for(int i = 1;i <= n;i++)
	{
		while(r <= n && s[r] - s[i-1] < m)		r++;

		if(s[r] - s[i-1] >= m)		ans = min(ans,r-i+1);

		if(r >= n) break;
	}

	if(ans == n + 1)  printf("%d",-1);
	else printf("%d",ans);
	return 0;
}
```

2120 - 连续自然数和

题目描述

对一个给定的自然数 M ，求出所有的连续的自然数段，这些连续的自然数段中的全部数之和为 M 。

例子：1998+1999+2000+2001+2002 = 10000，所以从 1998 到 2002 的一个自然数段为 M=10000 的一个解。

输入

包含一个整数的单独一行给出 M 的值（10 \le M \le 2,000,000 ）。

输出

每行两个自然数，给出一个满足条件的连续自然数段中的第一个数和最后一个数，两数之间用一个空格隔开；

所有输出行的第一个按从小到大的升序排列，对于给定的输入数据，保证至少有一个解。

样例

输入

10000

输出

18 142

297 328

388 412

1998 2002

```cpp
#include <iostream>//2120-1  kevinh    连续自然数和  （区间求和）
using namespace std;
const int N=2e6+10;
typedef long long LL;
LL a[N],n;

LL check(LL e)
{
	LL l=1,r=n,mid;
	while(l<=r)
	{
		mid = (l+r)/2;
		if(a[mid]==e)
		{
			return mid;
		}
		else if(e<a[mid])
		{
			r = mid-1;
		}
			else
			{
				l = mid+1;
			}
	}
	return 0;
}
```

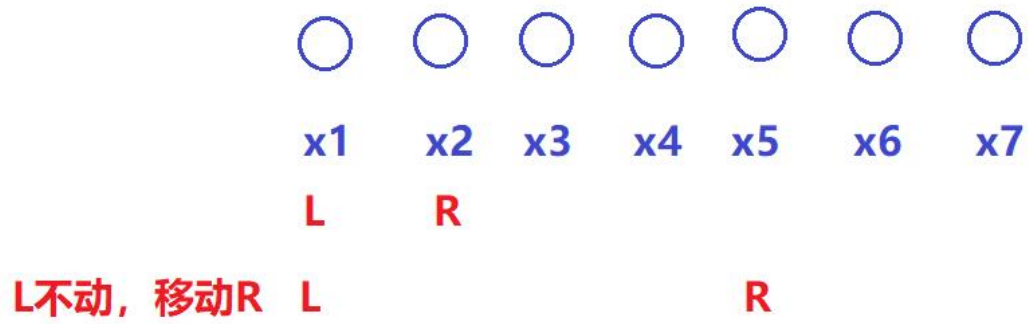
```
int main()
{
	cin>>n;

	for(int i=1;i<=n;i++)
	{
		a[i] = a[i-1]+i;
	}

	for(int i=1;i<=n;i++)
	{
		LL b = a[i];
		LL e = b+n;

		// 查的 e 在 a 数据中是否存在
		LL c = check(e);
		if(c!=0 && i+1!=c)
		{
			cout<<i+1<<”   “<<c<<endl;
		}
	}

	return 0;
}
```



假设有数列：$x1x1x1 \ldots xnxnxn$，是连续的递增的数字。

暴力版的解法：设LLL从$x1x1x1$开始求和，由于要数段的和为MMM，区间的右端点，要从$x2x2x2$开始，假设区间的右端点移动到$x5x5x5$发现总和$> m > m > m$，暴力的做法是$L++L++L++$，RRR重新从$L+1L+1L+1$的位置求和。不过根据本题的数据量，这种解法是不可行的。

分析一下，RRR是否有回头的可能？假设RRR回头了，也就是在$x1-x5x1-x5x1-x5$之间存在连续数的和为mmm，我们假设$x3-x4x3-x4x3-x4$的和是mmm，那么，当RRR移动到$x4x4x4$时，$L\ RL\ RL\ R$的和已经$> m > m > m$了，RRR是不可能移动到$x5x5x5$的位置的，因此RRR没有回头的可能。

经过这样的分析我们发现LLL和RRR都只需要右移，因此双指针的做法即可，时间复杂度为：$O(n)O(n)O(n)$。

```
#include<bits/stdc++.h>//2120-2  javacn  双指针解法  连续自然数和（区间求和）
using namespace std;
long long l,r,m,s;
int main()
{
	cin>>m;
	l = 1,r = 2;
	//l 不可能超过 m/2，因为如果 l,r 都 >m/2，则总和一定 >m
	while(l <= m / 2 && r <= m)
	{
		s = (l+r)*(r-l+1)/2;// 等差数列的区间和
		if(s == m)
		{
			cout<<l<<" "<<r<<endl;
			l++,r++;
		}
		else if(s < m) r++;// 区间和太小，扩大
		else l++;// 区间和太大，缩小
	}
	return 0;
}
```

本解法由用户：AACC 提供。

先求出所有可能的前缀和，再利用前缀和求区间和为 m 的区间。

```
#include <bits/stdc++.h>//2120-3     ojcpp       使用前缀和求解       连续自然数和
using namespace std;//（区间求和）
long long m,dp[1000010];// 准备前缀和求 1+2+...+n 的和，可能超 int
int main()
{
	scanf("%d",&m);
	dp[1] = 1;//dp 数组第一个的值为 1
	for(int i = 2; i <= m / 2 + 1; i++)
	{
		dp[i] = dp[i-1]+i;// 状态转移方程求 1+2+...+n 的值
	}

	for(int i = 1; i <= m/2; i++)
	{
		for(int j = i+1; j <= m/2+1; j++) // 双重循环求 i+i+1 + i+2 + j 的值
		{
			if(dp[j] - dp[i - 1] == m) // 如果 i+ i+1 + i+2 == m，输出
			{
				cout<<i<<" "<<j<<endl;
				break;
			}
			else if(dp[j]-dp[i-1]>m)	break;
		}
	}
	return 0;
}
```

差分

2062 - 倒水

在一个桌子上摆放了 n 个杯子，每个杯子中有一定量的水。小 A 同学负责向杯子中倒水，他总共倒了 k 次，每次会向从第 L 个杯子到第 R 个杯子中添加 P 毫升的水（注意：水只可能增加，不可能减少）。

请问小 A 同学倒了 k 次水之后， n 个杯子每个杯子有多少毫升的水。

输入

第一行包含两个整数 n 和 k。

第二行包含 n 个整数，表示一开始每个杯子中水的毫升数。

接下来 k 行，每行包含三个整数 L,R,P，表示一次操作。

数据范围

$1 \leqslant n,k \leqslant 100000$。

$1 \leqslant L \leqslant R \leqslant n,0 \leqslant P \leqslant 1000$。

杯子中水的初始量在 [0,1000] 的范围内。

本题数据上保证所有的杯子在加水之后，水量值仍然在 int 范围内。

输出

共一行，包含 n 个整数，表示最终 n 个杯子每个杯子有多少毫升的水。

样例

输入

```
8 3
1 2 10 8 1 5 1 1
7 8 12
1 8 4
2 3 12
```

输出

```
5 18 26 12 5 9 17 17
```

```cpp
#include <bits/stdc++.h>//2062-1  javacn    解法一：利用原数组求差分数组    倒水
using namespace std;//（区间修改）

const int N = 100010;
int a[N],b[N];//a 代表读入的原数组，b 代表是差分数组
int n,k,l,r,p;

int main()
{
	cin>>n>>k;
	for(int i = 1;i <= n;i++)
	{
		cin>>a[i];
		// 求差分数组
		b[i] = a[i] - a[i-1];
	}

	//k 次操作
	for(int i = 1;i <= k;i++)
	{
		cin>>l>>r>>p;
		b[l] = b[l] + p;
		b[r+1] = b[r+1] - p;
	}

	// 求 b 数组的前缀和，就是 a 数组做了 k 次操作的结果
	for(int i = 1;i <= n;i++)
	{
		b[i] = b[i-1] + b[i];
		cout<<b[i]<<" ";
	}
	return 0;
}
```

```cpp
#include <bits/stdc++.h>//2062-2    javacn    解法二：读入时直接求差分数组    倒水
using namespace std;//（区间修改）
const int N = 100010;
int a[N],b[N];//a 代表读入的原数组，b 代表是差分数组
int n,k,l,r,p;
int main()
{
	cin>>n>>k;
	for(int i = 1;i <= n;i++)
	{
		//cin>>a[i];
		// 每读入一个数，直接求出对应的差分数组
		cin>>p;
		// 求差分数组
		//b[i] = a[i] - a[i-1];
		b[i] = b[i] + p;
		b[i+1] = b[i+1] - p;
	}
	//k 次操作
	for(int i = 1;i <= k;i++)
	{
		cin>>l>>r>>p;
		b[l] = b[l] + p;
		b[r+1] = b[r+1] - p;
	}
	// 求 b 数组的前缀和，就是 a 数组做了 k 次操作的结果
	for(int i = 1;i <= n;i++)
	{
		b[i] = b[i-1] + b[i];
		cout<<b[i]<<" ";
	}
	return 0;
}
```

```cpp
#include<bits/stdc++.h>//2062-3    w2016010182    倒水
using namespace std;//（区间修改）
int a[100005];
int b[100005];
int main()
{
	int n,m;
	scanf("%d%d",&n,&m);

	for(int i=1;i<=n;++i)
	{
		scanf("%d",&a[i]);
		b[i]=a[i]-a[i-1];
	}

	for(int i=1;i<=m;++i)
	{
		int l,r,p;
		scanf("%d%d%d",&l,&r,&p);
		b[l]+=p;
		b[r+1]-=p;
	}

	for(int i=1;i<=n;++i)
	{
		a[i]=b[i]+a[i-1];//b[i]=a[i]-a[i-1];
	}

	for(int i=1;i<=n;++i)
	{
		printf("%d",a[i]);
	}
	return 0;
}
```

sin1538 - 小 X 与煎饼达人(flip)

玩着玩着小 X 觉得有点饿了， 他想出门买些吃的。 刚刚走出大门，小 X 就看到有位大叔在做煎饼，而且做法十分有趣。 只见此人将 n 块煎饼排成一排，手持一把大铲，将煎饼铲得上下翻飞， 煞是好看。 小 X 顿时食指大动， 赶紧走上前去细细打量， 发现此人做煎饼还十分的讲究，在做的过程中，大叔每次会将从第 x 块煎饼开始到第 y 块煎饼结束的这 y-x+1 块煎饼全部翻个个儿(正面翻到反面，反面翻到正面)。而他每次会选择不同的区间(区间是指连续的一段煎饼，如 3, 4, 5, 6 四块煎饼用区间 [3, 6] 表示）来翻这些煎饼。每块煎饼都有正反两面， 开始时这些煎饼都是反面朝上。

此人一共翻了 m 次煎饼，看得小 X 眼花缭乱。 但是小 X 很想知道这 n 块煎饼到最后一共有多少块是正面朝上的， 于是他只好求助于你了。

输入

输入数据第一行包含两个用空格隔开的正整数表示 n 和 m 。

接下来 m 行每行两个用空格隔开的正整数 x 和 y， 表示每次将区间 [x， y] 中的 y-x+1 块煎饼翻个个儿。

开始时这 n 块煎饼都是反面朝上（提示： 可以用 0 表示煎饼的反面，1 表示煎饼的正面）

输出

输出仅有一行包含一个整数 ans ， 表示最后有 ans 块煎饼是正面朝上的。

样例

输入

10 5

1 8

5 6

1 9

3 8

2 7

输出

5

【样例解释】

共有 10 块煎饼， 开始时状态为“反反反反反反反反反反”， 第一次操作将区间［1，8］的煎饼翻个身，状态变成“ 正正正正正正正正反反”， 红色表示翻的区间。 第二次操作将区间［5，6］的煎饼翻个身， 状态变成“正正正正反反正正反反”。 第三次操作将区间［1， 9］的煎饼翻个身， 状态变成“ 反反反反正正反反正反”。 第四次操作将区间［3，8］的煎饼翻个身， 状态变成“反反正正反反正正正反”。 第五次操作将区间［2，7］的煎饼翻个身， 状态变成“反正反反正正反正正反”。最后共有 5 块煎饼正面朝上。

【数据范围】

对于 30\% 的数据， 1 ≤ n,m ≤ 100，1 ≤ x ≤ y ≤ n;

对于另外 30\% 的数据， 1 ≤ n ≤ 1000000 ，1 ≤ m ≤ 100000， x = 1，1 ≤ y ≤ n;

对于另外 40\% 的数据， 1 ≤ n ≤ 1000000，1 ≤ m ≤ 100000，1 ≤ x ≤ y ≤ n;

```
#include<bits/stdc++.h>//1538      wangpeng  小 X 与煎饼达人（flip）
using namespace std;//（区间修改）
const int N = 1000000+ 100 ;
int n , m, l , r  , c = 0, b[N] ;
int main()
{
      scanf("%d %d" , &n , & m ) ;
      for( int i = 1; i <= m ; i++)
      {
            cin >> l >> r ;
            b[l]++;
            b[r+1]--;
      }
      for( int i = 1; i <= n ; i++)
      {
            b[i] = b[i] + b[i-1];
            if( b[i] % 2 != 0 )
            {
                  c++;
            }
      }
      cout<< c ;
      return 0 ;
}
```

2121 - 修正成绩

某校期中考试结束，学校采用一台阅卷机阅卷。老师在检查大家的成绩时，发现阅卷机阅卷有误，因此不得不手动调整大家的成绩。

现已知有 n 个同学成绩，需要做 p 次调整，每次调整操作都是将第 x 个同学到第 y 个同学每位同学成绩都加上 z 分。

请问：经过调整后，全班同学成绩的最低分是多少分？

输入

第一行有两个整数 n,p，代表学生数与增加分数的次数。

第二行有 n 个数，a_1 \sim a_n，代表各个学生的初始成绩。

接下来 p 行，每行有三个数，x,y,z，代表给第 x 个到第 y 个学生每人增加 z 分。

数据范围

$n \leqslant 100000$，$p \leqslant n$，$1 \leqslant x,y \leqslant n$，学生初始成绩$\leqslant 100$，$z \leqslant 100$。

输出

输出仅一行，代表更改分数后，全班的最低分。

样例

输入

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

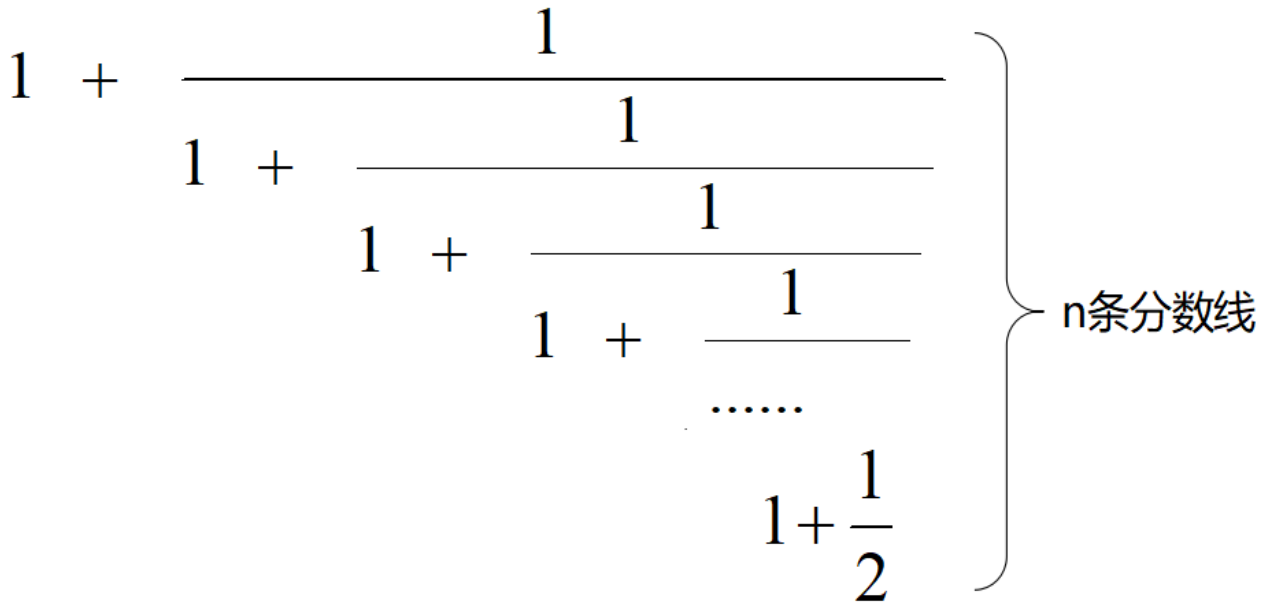
输出

```
2
```

```cpp
#include<bits/stdc++.h>//2121   w2016010182  修正成绩
using namespace std;//（区间修改）
int a[100005];
int sum[100005];
int main()
{
	int n,m;
	scanf("%d%d",&n,&m);
	for(int i=1;i<=n;++i)
	{
		scanf("%d",&sum[i]);
		a[i]=sum[i]-sum[i-1];
	}

	for(int i=1;i<=m;++i)
	{
		int l,r,p;
		scanf("%d%d%d",&l,&r,&p);
		a[l]=a[l]+p;
		a[r+1]=a[r+1]-p;
	}

	for(int i=1;i<=n;++i)
	{
		sum[i]=a[i]+sum[i-1];
	}
	sort(sum+1,sum+n+1);
	printf("%d",sum[1]);
	return 0;
}
```



有 n 条分数线的繁分式如上，请把它化简为一般的分式

输入：一个正整数 n。

输出：两行，第一行为分子，第二行为分母。[样]

输入：1

输出：:3　　2

[数据范围]

对于 100% 的数据 :1<n<40。

8.3

```
#include<bits/stdc++.h>// 分式
using namespace std;
int n,a=2,b=1;
int main(){
	scanf("%d",&n);
	for(int i=1;i<=n;i++)
	{
		swap(a,b);
		a+=b;
	}
	printf("%d\n%d",a,b);

	return 0;
}
```

习题 18-3 国庆阅兵

【题目描述】

十年一度的国庆阅兵仪式将要开始了：编号 1~n 的 n 个士兵整齐地排成一排，等待检阅。每位士兵胸口上都有他的兵种徽章 :1 表示海军，2 表示陆军，3 表示空军。

主席同志会进行 Q 次检阅，每次检阅区间 [a,b 中的士兵，请你帮忙统计每次检阅中海陆空三军的士兵数量分别是多少。

【输入格式】

第一行两个正整数 n 和 Q，表示士兵人数和检阅次数。

第二行 n 个数字 (1 或 2 或 3)，表示士兵的兵种 ;

接下来的 0 行，每行两个整数 a 和 b，表示每次检阅的士兵编号区间。

【输出格式】

共 Q 行，每行三个整数，依次表示海军、陆军、空军的士兵人数。

【数据范围】

对于 100% 的数据 :1 ≤ n ≤ 10 的 5 次方，1 ≤ Q ≤ 10 的 5 次方，1 ≤ a ≤ b ≤ n。

【解法分析】

本题考查前缀和数组，分别定义海军、陆军、空军人数的前缀和数组 s1、s2、s3;，再根据每次询问的区间 [l,r]，计算区间内海军、陆军、空军人数 s1[r]-s1[l-1]、s2[r]-s2[l-1]、s3[r]-s3[l-1]

【样例】

输入：

6 3

2 1 1 3 2 1

1 6

3 3

2 4

输出：

3 2 1

1 0 0

2 0 1

```
#include<iostream>//18.3 差分，国庆阅兵
using namespace std;
int main()
{
	int s[100],s1[100],s2[100],s3[100];
	int i,n,q,a,b;
	for(i=0;i<=99;i++)
	{
		s[i]=0;
		s1[i]=0;
		s2[i]=0;
		s3[i]=0;
	}
	cin>>n>>q;
	for(i=1;i<=n;i++)// 输入海陆空士兵，并且放入三个数组
	{
		cin>>s[i];
		if(s[i]==1)		s1[i]=s[i];
		if(s[i]==2)		s2[i]=s[i];
		if(s[i]==3)		s3[i]=s[i];
	}
	for(i=1;i<=n;i++)// 海陆空数组前缀和
	{
		s1[i]+=s1[i-1];
		s2[i]+=s2[i-1];
		s3[i]+=s3[i-1];
	}
	for(i=1;i<=q;i++)
	{
		cin>>a>>b;
		cout<<s1[b]-s1[a-1]<<" "<<(s2[b]-s2[a-1])/2<<" "<<(s3[b]-s3[a-
1])/3<<endl;
	}
	return 0;
}
```

```
#include<iostream>// 差分，国庆阅兵
using namespace std;
int main()
{
	int s[100],s1[100],s2[100],s3[100];
	int i,n,q,a,b;
	for(i=0;i<=99;i++)// 数组初始化为 0
	{
		s[i]=0;
		s1[i]=0;
		s2[i]=0;
		s3[i]=0;
	}

	cin>>n>>q;
	for(i=1;i<=n;i++)// 输入海陆空士兵，并且放入三个数组
	{
		cin>>s[i];
		if(s[i]==1)	s1[i]=s1[i-1]+s[i];
		else  s1[i]=s1[i-1]+0;

		if(s[i]==2)	s2[i]=s2[i-1]+s[i];
		else  s2[i]=s2[i-1]+0;

		if(s[i]==3)	s3[i]=s3[i-1]+s[i];
		else  s3[i]=s3[i-1]+0;
	}
	for(i=1;i<=q;i++)
	{
		cin>>a>>b;
		cout<<s1[b]-s1[a-1]<<" "<<(s2[b]-s2[a-1])/2<<" "<<(s3[b]-s3[a-
1])/3<<endl;
	}
	return 0;
}
```

小羊们上完课后，纷纷到草场上吃草。而羊村现在正在进行特色示范羊村检查，领导们想看看羊村的草场。羊村的草场是连续分布的，每块草场上都有数量不等的羊在吃草。领导们想要查看连续若干个草场，但是又不想看到超过 T 只羊。而村长希望领导们多看看羊村的风貌，尽可能多参观几个草场。现在，请你帮村长决定，带领导们去参观哪一段草场，满足领导和村长的要求。

输入

第一行一个整数N和T,表示羊村共有多少个连续草场,以及领导们希望看到羊数量的最大值。

第二行 N 个整数，两个整数间用一个空格分开，第 i 个数 ai 表示第 i 个草场上有 ai 只羊在吃草。编号从 1 到 N。

输出

输出一行，共两个数，表示参观的起点编号和终点编号，中间用空格分开。走的方向总是从编号小的到编号大的。另外，若有长度相同的可能性，输出起点编号较小的答案。数据保证至少有答案存在。

8.4-a

```
#include<iostream>
#include<cstdio>
using namespace std;
int n,m,j,x,y,sum=0,a[100100];
int main()
{
	scanf("%d%d",&n,&m);
	int i,j=1;
	for(i=1;i<=n;i++)
	{
		scanf("%d",&a[i]);
		sum+=a[i];
		while(sum>m)// 保证当前草场可以参观
			sum-=a[j++];
		if(i-j+1>y-x+1)
			x=j,y=i;
	}
	printf("%d %d",x,y);
}
```

样例输入

5 10

6 3 2 1 7

样例输出

2 4

数据范围限制

30% 的数据，1<=N<=100;60% 的数据，1<=N<=1000;

100% 的数据，1<=N<=100000,

0<=ai<=109， 0<=T<=231-1。

提示要满足连续个总和不超过10，有 3+2+1 和 2+1+7 两种可能性，优先输出 2 到 4 个草场。

解题思路，枚举草场，保证当前的草场可以参观

8.4-b

```
#include<iostream>
using namespace std;
int i,j,n,a[100010],t,mx,mi,mj;
int main()
{
	scanf("%d%d",&n,&t);//n 块草场 t 只羊
	for(i=1;i<=n;i++)
	{
		scanf("%d",&a[i]);
		a[i]=a[i-1]+a[i];
	}
	for(i=1;i<=n-1;i++)// 从第一块草场开始走
	{
		for(j=n;j>=i+1;j--)// 从最后一块草场
		{
			if(a[j]-a[i-1]<=t)
			{
				if(j-i+1>mx)
				{
					mx=j-i+1;
					mi=i;
				mj=j;
					break;
				}
			}
		}
		if(n-i<=mj-mi) break;
	}
	printf("%d %d",mi,mj);

    return 0;
}
```

Bessie 对她自己最近在农场周围的恶作剧感到抱歉，于是她同意帮助 Farmer John 堆叠新送达的一批干草捆。

开始时有 N(N 是奇数）个空草堆，标记为 1…N。FJ 将会给 Bessie 包含 K 条指令的序列，每条指令的格式为 "A B"，表示 Bessie 应该在草堆 A…B 中每一个草堆顶部新放一捆干草。例如，如果 Bessie 听到指令 "10 13"，那么她应该在草堆 10,11,12 和 13 上各新放一捆干草。在 Bessie 完成了 FJ 的所有指令后，FJ 想要知道 N 个草堆高度的中位数——也就是说，所有草堆排序后中央草堆（由于 N 是奇数，这个草堆是唯一的）的高度。请帮助 Bessie 确定 FJ 问题的答案。

8.5

```
#include<iostream>// 堆叠草堆
#include<algorithm>
using namespace std;
const int N=1e6+100;
int a[N],f[N];
int main(){
int n,k,b,c;
cin>>n>>k;//7 4
for(int i=1;i<=k;i++)
{
      cin>>b>>c;
      f[b]++;
      f[c+1]=f[c+1]-1;
}
for(int i=1;i<=n;i++)
{
      a[i]=a[i-1]+f[i];
}
sort(a+1,a+n+1);
cout<<a[(n+1)/2]<<endl;
return 0;
}
```

输入

第 1 行：两个被空格分隔的整数 N 和 K。

第 2…1+K 行：每行包含一条 FJ 的指令，其格式为两个被空格分隔的整数 A 和 B。

输出

一个整数， Bessie 完成所有指令后草堆高度的中位数。

输入样例

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

输出样例

```
1
```