

第 5 章 前缀和、差分

一、前缀和

1、前缀和

对于一维数组，前缀和是一个数组的某项下标之前（包括此项元素）的所有数组元素的和。

比如：定义一个数组 $a[]$ 的前缀和数组 b ， $b[i]=a[1]+a[2]+\dots+a[i]$ 。

二维前缀和与一维前缀和类似，设 $b[i][j]$ 表示所有 $b[i'][j']$ 的和。（ $1 \leq i' \leq i, 1 \leq j' \leq j$ ）

有一点点像“矩形的面积”那样，把一整块区域的值都加起来。

类型	定义	递推式
一维数组求前缀和	$b[i] = \sum_{j=0}^i a[j]$	$b[i] = b[i-1] + a[i]$
二维数组求前缀和	$b[x][y] = \sum_{i=0}^x \sum_{j=0}^y a[i][j]$	$b[x][y] = b[x-1][y] + b[x][y-1] - b[x-1][y-1] + a[x][y]$

2、前缀和的作用

利用前缀和求区间和。

举例：

（1）一维数组：

a 数组：存储元素

1	3	6	5	4	2
1	2	3	4	5	6

b 数组：存储前缀和

1	4	10	15	19	21
1	2	3	4	5	6

问：从第 1 个数到第 r 个数的和是多少？

答： $\text{sum}[1, r] = b[r] - b[1-1]$

如： $\text{sum}[3, 5] = b[5] - b[2] = 15$

(2) 二维数组:

	1	2	3	4	5	6
1						
2						
3						
4			i, j			
5						
6						

	1	2	3	4	5	6
1						
2						
3			x1y1			
4						
5						
6				x2y2		

结论:

(1) $b[i, j]$ 即为图 1 红框中所有数的和 (前缀和) 为:

$$b[i][j] = b[i-1][j] + b[i][j-1] - b[i-1][j-1] + a[i][j]$$

(2) 根据前缀和求图 2 红框中的区间和为:

$$b[x2y2] - b[x1-1][y2] - b[x2][y1-1] + b[x1-1][y1-1]$$

3、课堂练习

2060: 【入门】计算能力

```
#include<bits/stdc++.h>
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",&n,&m);
    //读入 n 个数
    for(int i = 1;i <= n;i++){
        scanf("%d",&a[i]);
        //求前缀和
        b[i] = b[i-1] + a[i];
    }

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

    return 0;
}
```

2061: 【入门】子矩阵求和

```
#include <bits/stdc++.h>
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);
    //读入二维数组
    for(int i = 1;i <= n;i++){
        for(int j = 1;j <= m;j++){
            scanf("%d",&a[i][j]);
            //求前缀和
            b[i][j] = b[i-1][j] + b[i][j-1] - b[i-1][j-1] + a[i][j];
        }
    }

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

2119: 【入门】任务的最少完成时间

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

/*
有 n 个数，可以删除连续的 k 个数
求剩余数的和最小是多少？
相当于求：连续 k 个数的和最大是多少？
*/
int n,k;
const int N = 1000010;
//为 long long 类型起别名
typedef long long LL;
LL a[N],b[N],ma;//b 代表前缀和

int main(){
    scanf("%d%d",&n,&k);
    //读入 n 个整数
    for(int i = 1;i <= n;i++){
        scanf("%d",&a[i]);
        //求前缀和
        b[i] = b[i-1] + a[i];
    }

    //求从每个数开始连续 k 个数的和，在所有的和中求最大
    for(int i = 1;i <= n - k + 1;i++){
        //区间范围：i~i+k-1
        ma = max(ma,b[i+k-1]-b[i-1]);
    }

    //求剩余任务时间和的最小值
    cout<<b[n] - ma;
    return 0;
}
```

2105: 【入门】不太甜的糖果

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

/*
```

给定一排长度为 n 的糖果串，每个糖果有一个甜度，
求出一个最短的糖果串使得它的甜度之和大于等于 m 。

给定一个数组，求最短的子串，使得元素之和 $\geq m$ 。

解题关键：

1. 二分可能的长度
2. 用前缀和求区间和

```
*/
const int N = 230010;
int a[N], f[N]; // f: 代表前缀和
int n, m;

// 检验子串长度为 mid 的情况下，子串和是否可能  $\geq m$ 
bool check(int mid){
    // 从每个可能的开头开始求连续 mid 个数的和
    for(int i = 1; i <= n - mid + 1; i++){
        // 区间范围: i ~ i+mid-1
        if(f[i+mid-1] - f[i-1] >= m) return true;
    }
    return false;
}

int main(){
    cin >> n >> m;
    for(int i = 1; i <= n; i++){
        cin >> a[i];
        // 求前缀和
        f[i] = f[i-1] + a[i];
    }

    // 如果所有数的和都不满足  $\geq m$ ，则无解
    if(f[n] < m){
        cout << 0;
        return 0;
    }

    // 二分可能的长度
    int l = 1, r = n, mid;
    while(l <= r){
        mid = l + r >> 1;
        // 如果长度为 mid 的子串和满足  $\geq m$ 
        if(check(mid)) r = mid - 1;
        else l = mid + 1;
    }

    // 如果找不到这样的串，输出 0
    // if(l == n+1) cout << 0;
    // else cout << l; // 输出左边界

    cout << l; // 输出左边界

    return 0;
}
```

4、课后练习

2120: 【基础】连续自然数和

1560: 【基础】连续数的和

二、差分

1、什么是差分

假设 a 数组是 b 数组的前缀和，那么 b 数组就是 a 数组的差分数组。差分与前缀和是逆运算。（之所以叫做差分，是因为我们需要维护的数据是“相邻两个数之差”。）

比如：

a 数组：相当于存储的 b 数组的前缀和

1	4	10	15	19	21
1	2	3	4	5	6

b 数组：由于 a 数组相当于 b 数组的前缀和，因此 b 数组就是 a 数组的差分

1	3	6	5	4	2
1	2	3	4	5	6

由于： $a[i] = a[i-1] + b[i]$ ，因此 $b[i] = a[i] - a[i-1]$ 。

2、差分的作用

差分数组的功能是修改区间，查询点。

比如：假设有 k 次操作，每次都需要将 a 数组中下标为 [l, r] 之间的数都加上元素 x，那么如果要在 O(1) 的时间完成上述操作，可以直接对差分数组 b 进行如下操作。

$b[l] = b[l] + x;$

$b[r+1] = b[r+1] - x;$

实际上，用上述思想就可以直接构建差分数组，把 a 数组的每个值想象为要在 b 数组对应位置要加的值，那么，可以这样构建差分数组：

$b[i+1] = a[i];$

$b[i] = a[i] - b[i+1];$

3、课堂练习

2062：【基础】倒水

解法一：利用原数组求差分数组

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

```
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;
}
```

解法二：读入时直接求差分数组

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

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;
}
```

1538: 【提高】小 X 与煎饼达人(flip)

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

/*
开始时这些煎饼都是反面朝上(用 0 表示)
m 次操作之后，问：有多少正面朝上的？
求：每个位置翻的次数，如果翻了奇数次，就是正面朝上
*/
const int N = 1000010;
int a[N];//差分数组
int n,m,c = 0;

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

```
//读入操作
int x,y;
for(int i = 1;i <= m;i++){
    scanf("%d%d",&x,&y);
    a[x]++;
    a[y+1]--;
}

//求 a 数组的前缀和，统计奇数出现的次数
for(int i = 1;i <= n;i++){
    a[i] = a[i-1] + a[i];
    if(a[i] % 2 != 0) c++;
}

cout<<c;
return 0;
}
```

4、课后练习

2121: 【入门】修正成绩

