

4.3 课堂笔记 5

本周作业：

- 1、 P6568
- 2、 P3406

上周作业答案：

P2678

```
1  #include<iostream>
2  #include<algorithm>
3  using namespace std;
4  int a[50001]={0},l,n,m,i;
5
6  bool Check(int mid)
7  {
8      int start=0,x=0;//用start表示每次落脚点的坐标，每落一次地更新一次start
9      for(i=1;i<=n;i++)
10     {
11         if(a[i]-start<mid)
12             x++;//x表示去掉的石头数，如果mid大于要跳的距离，就跳过当前这个石头，此时x++，并且不落地
13         else
14             start=a[i];//此时落地
15     }
16     if(l-start<mid)//判断最后一跳跳的距离要是小于mid的话那是不可以的
17         return false;
18     if(x>m)//要是x>m就说明最小距离mid太大
19         return false;
20     return true;
21 }
22
23
24 int Erfen(){
25     int left=0,right=l,mid,ans;
26     while(right>=left)
27     {
28         mid=(left+right)/2;//mid表示最小的距离
29         if(Check(mid))
30         {
31             left=mid+1;
32             ans=mid;
33         }
34         else{
35             right=mid-1;
36         }
37     }
38     return ans;
39 }
40
41 int main(){
42     cin >> l >> n >> m;
43     for(i=1;i<=n;i++){
44         cin >> a[i];
45     }
46     sort(a,a+n+1);
47     cout << Erfen();
48     return 0;
49 }
50
```

P3743

```
1  #include<iostream>
2  using namespace std;
3
4  const int N = 100010;
5
6  double a[N], b[N], p;
7  int n;
8
9  bool check(double x){
10     double sum = 0;
11     for(int i = 0; i < n; i++){
12         if(a[i] * x > b[i]) sum += a[i] * x - b[i];
13     }
14     if(sum > x * p) return false; // 如果总耗电数大于这段时间内充电宝能提供的最大电量，说明不能一起使用x秒
15     return true;
16 }
17
```

```

16 }
17
18 int main(){
19     cin >> n >> p;
20
21     for(int i = 0; i < n; i++) cin >> a[i] >> b[i];
22
23     double sum = 0;
24     for(int i = 0; i < n; i++) sum += b[i];
25     if(sum <= p){
26         cout << -1;
27         return 0;
28     }
29
30     double l = 0, r = 1e10;
31
32     while(r - l >= 1e-4){
33         double mid = (l + r) / 2;
34         if(check(mid)) l = mid;
35         else r = mid;
36     }
37
38     cout << l << endl;
39
40     return 0;
41 }
42

```

前缀和与差分

一、一维前缀和

如果给你一串长度为 n 的数列 $a_1, a_2, a_3, \dots, a_n$, 再给出 m 个询问, 每次询问给出 L, R 两个数, 要求给出区间 $[L, R]$ 里的数的和, 你会怎么做? 对于 m 次询问, 每次都遍历一遍它给的区间, 计算出答案, 但是其时间复杂度达到了 $O(n*m)$, 如果数据量稍微大一点就有可能超时。故用一维前缀和做预处理, 降低时间复杂度。

$a[1], a[2] \dots a[i] \dots a[n]$

$sum[i] = a[1] + a[2] + \dots + a[i]$

样例:

样例输入1:

```

5
5 7 -1 3 0
2
1 3
2 5

```

样例输出1:

```

11
9

```

算法实现:

```

for(int i=1; i<=n; i++) {
    cin >> a[i];
    sum[i] = sum[i-1] + a[i];
}
for(int i=1; i<=m; i++) {
    cin >> start >> end;
    cout << sum[end] - sum[start - 1] << " ";
}

```

练习：P5638

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  long long sum[100000],a[100000];
4  long long n,k,x,maxx;
5  int main()
6  {
7      cin>>n>>k;
8      if(k>=n-1)
9      {
10         cout<<0<<endl; //半径大于等于n-1, 则直接在起点使用传送门, 不耗费时间到达终点
11         return 0;
12     }
13
14     for(int i=1;i<=n-1;i++)
15     {
16         cin>>a[i];
17         sum[i]=sum[i-1]+a[i]; //耗时前缀和
18     }
19
20     for(int i=0; i<=n-1-k; i++) //注意边界, 要包含从起点就使用传送门的情况
21     {
22         maxx = max(maxx, sum[i+k]-sum[i]); //求使用传送门能减少的最大耗时
23     }
24
25     cout<<sum[n-1]-maxx<<endl;
26     return 0;
27 }

```

二、二维前缀和

左上坐标 (1,1) 右下坐标 (4,4)	9	1	5	2
	3	5	7	9
	2	1	1	2
	5	9	10	3

整体求和：

```

for(int i = 1; i <= _n; i++) {
    for(int j = 1; j <= _n; j++) {
        cin >> _a[i][j];
        _sum[i][j] = _sum[i][j - 1] + _sum[i - 1][j] - _sum[i - 1][j - 1] + _a[i][j];
    }
}

```

部分求和：

公式：

$$\text{sum}[x2][y2] - \text{sum}[x2][y1-1] - \text{sum}[x1-1][y2] + \text{sum}[x1-1][y1-1]$$

灰色求和： $\text{sum}(2,2)$
黄色求和： $\text{sum}(2,4) - \text{sum}(2,2)$
绿色求和： $\text{sum}(4,2) - \text{sum}(2,2)$
紫色求和： $\text{sum}(4,4) - \text{sum}(4,2) - \text{sum}(2,4) + \text{sum}(2,2)$

练习 1：P1719

```

3
4 int s[121][121], a[121][121];
5 int ans = -1e9;
6
7 int main(){
8     int n;
9     cin >> n;
10    for(int i = 1; i <= n; i++){
11        for(int j = 1; j <= n; j++){
12            cin >> a[i][j];
13            s[i][j] = s[i-1][j] + s[i][j-1] - s[i-1][j-1] + a[i][j]; //二维前缀和
14        }
15    }
16    //遍历矩阵，求出所有形状矩阵的加权和
17    for(int x1 = 1; x1 <= n; x1++){
18        for(int y1 = 1; y1 <= n; y1++){
19            for(int x2 = x1; x2 <= n; x2++){
20                for(int y2 = y1; y2 <= n; y2++){
21                    ans = max(ans, s[x2][y2] - s[x2][y1-1] - s[x1-1][y2] + s[x1-1][y1-1]);
22                }
23            }
24        }
25    }
26    cout << ans;
27    return 0;
28 }
29

```

练习 2: P2004

```

2 using namespace std;
3
4 int s[1010][1010], a[1010][1010];
5 int maxx=-1e9;
6 int x,y;
7
8 int main(){
9     int n, m, c;
10    cin >> n >> m >> c;
11    for(int i = 1; i <= n; i++){
12        for(int j = 1; j <= m; j++){
13            cin >> a[i][j];
14            s[i][j] = s[i-1][j] + s[i][j-1] - s[i-1][j-1] + a[i][j]; //二维前缀和
15        }
16    }
17    for(int x1 = 1; x1 <= n-c+1; x1++){ //注意边界
18        for(int y1 = 1; y1 <= m-c+1; y1++){
19            if((s[x1+c-1][y1+c-1] - s[x1+c-1][y1-1] - s[x1-1][y1+c-1] + s[x1-1][y1-1]) > maxx){
20                maxx = s[x1+c-1][y1+c-1] - s[x1+c-1][y1-1] - s[x1-1][y1+c-1] + s[x1-1][y1-1]; //更新最大值
21                x = x1;
22                y = y1;
23            }
24        }
25    }
26    cout << x << " " << y << endl;
27    return 0;
28 }
29

```

三、一维差分

首先给定一个原数组 a : $a[1], a[2], a[3], \dots, a[n]$;

然后我们构造一个数组 b : $b[1], b[2], b[3], \dots, b[i]$;

使得 $a[i] = b[1] + b[2] + b[3] + \dots + b[i]$

a 数组是 b 数组的前缀和数组，反过来我们把 b 数组叫做 a 数组的差分数组。

$$a[i] = b[1] + b[2] + b[3] + \dots + b[i]$$

$$a[i-1] = b[1] + b[2] + b[3] + \dots + b[i-1]$$

上述两个式子做差得到:

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

$b[i]$ 的值可看作是 a 数组每个元素与前一个元素的差值

前缀和与差分的区别:

前缀和: 快速求某个区间的和

差分: 快速表示某个区间的变化趋势

对差分数组求前缀和可求得原数组

给定区间 $[l, r]$ ，让我们把 a 数组中的 $[l, r]$ 区间中的每一个数都加上 c ，即 $a[l] + c, a[l+1] + c, a[l+2] + c, \dots, a[r] + c$ ；暴力做法是 for 循环 l 到 r 区间，时间复杂度 $O(n)$ ，如果我们需要对原数组执行 m 次这样的操作，时间复杂度就会变成 $O(n*m)$ 。有没有更高效的做法？同样，我们用一维差分预处理，降低时间复杂度。

差分数组操作步骤：

$b[l] + c$

$b[r+1] - c$

对 b 求前缀和即可得到 a 数组

练习 1:

有 N 头牛站成一行，被编队为 $1, 2, 3, \dots, N$ ，每头牛的身高都为整数。当且仅当两头牛中间的牛身高都比它们矮时，两头牛方可看到对方。现在，我们只知道其中最高的牛是第 P 头，它的身高是 H ，剩余牛的身高未知。但是，我们还知道这群牛之中存在着 M 对关系，每对关系都指明了某两头牛 A 和 B 可以相互看见。求每头牛的身高的最大可能值是多少。

输入格式

第一行输入整数 N, P, H, M ，数据用空格隔开。

接下来 M 行，每行输出两个整数 A 和 B ，代表牛 A 和牛 B 可以相互看见，数据用空格隔开。

输出格式

一共输出 N 行数据，每行输出一个整数。

第 i 行输出的整数代表第 i 头牛可能的最大身高。

数据范围

$1 \leq N \leq 10000$,

$1 \leq H \leq 1000000$,

$1 \leq A, B \leq 10000$,

$0 \leq M \leq 10000$

注意：

此题中给出的关系对可能存在重复

输入样例：

9 3 5 5

1 3

5 3

4 3

3 7

9 8

输出样例：

5

4

5

3

4

4

5

5

5

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 int n, p, h, m, a, b, cow[10005], c[10005], cnt = 0;
4 struct stu{
5     int a, b;
6 }s[10005];
7
8 void ParseInCore(){
9     cin >> n >> p >> h >> m;
10    for(int i = 1; i <= m; i++){
11        bool check = true;
12        cin >> a >> b;
13        if(a > b){
14            swap(a, b); // 确定左右区间
15        }
16        for(int j = 1; j <= cnt; j++){
17            if(a == s[j].a && b == s[j].b){
18                check = false; // 如果关系对有重复的情况，则不需要再减
19            }
20        }
21        if(check){
22            c[a + 1]--; // 求差分数组，c[i]表示相邻两头牛高度差，c[i]=cow[i]-cow[i-1]
23            c[b]++;
24            s[i].a = a;
25            s[i].b = b;
26            cnt++; // cnt表示关系对数
27        }
28    }
29 }
```

```

27 |     }
28 | }
29 |
30 |
31 | void WriteOut (){
32 |     cow[1] = c[1] = h; //初始化, 第一头牛一定可以是最高高度
33 |     for(int i = 1; i <= n; i++){
34 |         cow[i] = cow[i-1] + c[i];
35 |         cout << cow[i] << endl;
36 |     }
37 | }
38 |
39 | int main (){
40 |     ParseInCore ();
41 |     WriteOut ();
42 |     return 0;
43 | }

```

练习 2:

给定一个长度为 n 的数列 a_1, a_2, \dots, a_n , 每次可以选择一个区间 $[l, r]$, 使下标在这个区间内的数都加一或者都减一。

求至少需要多少次操作才能使数列中的所有数都一样, 并求出在保证最少次数的前提下, 最终得到的数列可能有多少种。

输入格式

第一行输入正整数 n 。

接下来 n 行, 每行输入一个整数, 第 $i+1$ 行的整数代表 a_i 。

输出格式

第一行输出最少操作次数。

第二行输出最终能得到多少种结果。

数据范围

$0 < n \leq 10^5$,

$0 \leq a_i < 2147483648$

输入样例:

```

4
1
1
2
2

```

输出样例:

```

1
2

```

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  int n,a[10001],c[10001];
4  int pos,neg;
5  int main(){
6      cin >> n;
7      for(int i=1; i<=n; i++){
8          cin >> a[i];
9      }
10     for(int i=1; i<=n; i++){
11         c[i] = a[i]-a[i-1]; //差分数组
12     }
13     for(int i=2; i<=n; i++){ //c[1]不用统计
14         if(c[i]>0){
15             pos += c[i]; //差分大于0的累计和
16         }
17         else if(c[i]<0){
18             neg -= c[i]; //差分小于0的累计和
19         }
20     }
21     cout << max(pos,neg) << endl; //最少变换次数
22     cout << abs(pos-neg) + 1 << endl; //数列种类
23     return 0;
24 }

```