

区间修改问题

一维差分 <https://www.lanqiao.cn/problems/18438/learning/>

小蓝的操作 <https://www.lanqiao.cn/problems/3399/learning/>

问题描述

给定一个长度为 n 的序列 a 。

再给定 m 组操作，每次操作给定 3 个正整数 l, r, d ，表示对 $a_{l \sim r}$ 中的所有数增加 d 。

最终输出操作结束后的序列 a 。

输入格式

第一行输入两个正整数 n, m 。 ($1 \leq n, m \leq 2 \times 10^5$)

第二行输入 n 个正整数 a_i 。 ($1 \leq i \leq n, 1 \leq a_i \leq 10^4$)。

接下来 m 行，每行输入 3 个正整数 l, r, d 。 ($1 \leq l \leq r \leq n, -10^4 \leq d \leq 10^4$)。

- C++

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  const int N=2e5+10;
4  int a[N];
5  int n,m;
6  int main(){
7      scanf("%d%d",&n,&m);
8      for(int i=1;i<=n;i++) scanf("%d",&a[i]);
9      for(int i=1;i<=m;i++){
10         int l,r,d;
11         scanf("%d%d%d",&l,&r,&d);
12         for(int j=l;j<=r;j++){
13             a[j]+=d;
14         }
15     }
16     for(int i=1;i<=n;i++) printf("%d ",a[i]);
17 }
```

- Java

```
1  import java.util.Scanner;
2
3  public class Main {
4      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in);
6          int N = 200010; // 与 C++ 中的数组大小一致
7          int[] a = new int[N];
8          int n = scanner.nextInt();
9          int m = scanner.nextInt();
10
11          // 输入数组
12          for (int i = 1; i <= n; i++) {
```

```

13         a[i] = scanner.nextInt();
14     }
15
16     // 处理每个操作
17     for (int i = 1; i <= m; i++) {
18         int l = scanner.nextInt();
19         int r = scanner.nextInt();
20         int d = scanner.nextInt();
21         for (int j = l; j <= r; j++) {
22             a[j] += d;
23         }
24     }
25
26     // 输出结果
27     for (int i = 1; i <= n; i++) {
28         System.out.print(a[i] + " ");
29     }
30     scanner.close();
31 }
32 }
33

```

- Python

```

1  import sys
2  input = sys.stdin.read
3
4  # 读取所有输入
5  data = input().split()
6  index = 0
7
8  # 读取 n 和 m
9  n = int(data[index])
10 m = int(data[index + 1])
11 index += 2
12
13 # 初始化数组
14 N = 20010
15 a = [0] * N
16
17 # 读取数组元素
18 for i in range(1, n + 1):
19     a[i] = int(data[index])
20     index += 1
21
22 # 处理每个操作
23 for _ in range(m):
24     l = int(data[index])
25     r = int(data[index + 1])
26     d = int(data[index + 2])
27     index += 3
28     for j in range(l, r + 1):
29         a[j] += d
30
31 # 输出结果
32 sys.stdout.write(" ".join(map(str, a[1:n + 1])) + "\n")
33

```

一维差分

差分数组是一种简单而高效的数据结构，常用于快速处理**区间加减问题**。

对于一个长度为 n 的数组 a ，我们构造一个差分数组 b ，其定义为：

$$b[i] = a[i] - a[i - 1], \quad \text{对于 } i \geq 2, \quad b[1] = a[1].$$

- **从差分数组还原原数组：**

通过对差分数组求前缀和可以还原出原数组：

$$a[i] = \sum_{j=1}^i b[j].$$

- **差分数组的意义：**

差分数组中的每个元素 $b[i]$ 表示原数组中 $a[i]$ 相对于 $a[i - 1]$ 的变化量。

差分的性质

1. **原数组求差分：**

从原数组构造差分数组时，每次仅需 $O(n)$ 。

2. **差分数组求前缀和：**

从差分数组恢复原数组，每次查询或还原原数组均为 $O(n)$ 。

3. **快速区间修改：**

通过调整差分数组可以高效完成原数组的区间操作。

- 对区间 $[l, r]$ 的每个元素加上一个值 d ，只需：

$$b[l] += d, \quad b[r + 1] -= d.$$

- 在完成所有修改后，通过对差分数组求前缀和即可得到最终的结果数组。

差分与前缀和的关系

原数组求前缀和 = 前缀和数组

差分数组求前缀和 = 原数组

前缀和数组求差分 = 原数组

原数组求差分 = 差分数组

- C++

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  #define endl '\n'
4  const int N = 1e6 + 10;
5  int a[N];
6  int diff[N];
7
8  void solve() {
9      int n, m;
10     scanf("%d %d", &n, &m);
11     for (int i = 1; i <= n; i++) {
12         scanf("%d", &a[i]);
13         diff[i] = a[i] - a[i - 1];
14     }
15     while (m--) {
```

```

16     int l, r, x;
17     scanf("%d %d %d", &l, &r, &x);
18     diff[l] += x;
19     diff[r + 1] -= x;
20 }
21 for (int i = 1; i <= n; i++) {
22     diff[i] += diff[i - 1];
23     printf("%d ", diff[i]);
24 }
25 printf("\n");
26 }
27
28 int main() {
29     int t = 1;
30     while (t--) {
31         solve();
32     }
33     return 0;
34 }
35

```

- Java

```

1  import java.util.Scanner;
2
3  public class Main {
4      public static void main(String[] args) {
5          Scanner sc = new Scanner(System.in);
6
7          // 输入 n 和 m
8          int n = sc.nextInt();
9          int m = sc.nextInt();
10
11         long[] a = new long[n + 1];
12         long[] diff = new long[n + 2];
13
14         // 读取数组 a 并构造差分数组
15         for (int i = 1; i <= n; i++) {
16             a[i] = sc.nextLong();
17             diff[i] = a[i] - a[i - 1];
18         }
19
20         // 处理 m 次操作
21         while (m-- > 0) {
22             int l = sc.nextInt();
23             int r = sc.nextInt();
24             long x = sc.nextLong();
25
26             diff[l] += x;
27             diff[r + 1] -= x;
28         }
29
30         // 计算最终数组并输出
31         for (int i = 1; i <= n; i++) {
32             diff[i] += diff[i - 1];
33             System.out.print(diff[i] + " ");
34         }
35

```

```

35         System.out.println();
36
37         sc.close();
38     }
39 }

```

- Python

```

1  import sys
2  input = sys.stdin.read
3  output = sys.stdout.write
4
5  def solve():
6      # 读取所有输入数据
7      data = input().splitlines()
8
9      # 解析 n 和 m
10     n, m = map(int, data[0].split())
11
12     # 初始化数组 a 和差分数组 diff
13     a = [0] * (n + 1)
14     diff = [0] * (n + 2)
15
16     # 读取数组 a 并构造差分数组
17     for i in range(1, n + 1):
18         a[i] = int(data[i])
19         diff[i] = a[i] - a[i - 1]
20
21     # 处理 m 次操作
22     for i in range(n + 1, n + 1 + m):
23         l, r, x = map(int, data[i].split())
24         diff[l] += x
25         if r + 1 <= n:
26             diff[r + 1] -= x
27
28     # 计算最终数组并输出
29     result = []
30     for i in range(1, n + 1):
31         diff[i] += diff[i - 1]
32         result.append(str(diff[i]))
33
34     # 使用 ACM 输出
35     output(" ".join(result) + "\n")
36
37 if __name__ == "__main__":
38     solve()
39

```

小蓝的操作

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  const int N = 1e5 + 10;
5  int n;
6  int a[N];

```

```

7   int d[N];
8
9   void solve() {
10      cin >> n;
11      long long ans = 0;
12      for (int i = 1; i <= n; i++) {
13          cin >> a[i];
14          d[i] = a[i] - a[i - 1];
15          if (d[i] > 0) ans += d[i];
16      }
17      cout << ans - 1 << "\n";
18  }
19
20  int main() {
21      ios::sync_with_stdio(false);
22      cin.tie(0);
23      solve();
24      return 0;
25  }
26

```

- Java

```

1   import java.io.*;
2   import java.util.*;
3
4   public class Main {
5       static int N = (int)(1e5+10);
6       static int n;
7       static int[] a = new int[N];
8       static int[] d = new int[N];
9
10      public static void main(String[] args) {
11          solve();
12          out.flush();
13      }
14
15      static void solve() {
16          n = in.nextInt();
17          long ans = 0;
18          for(int i=1;i<=n;i++){
19              a[i] = in.nextInt();
20              d[i] = a[i]-a[i-1];
21              if(d[i]>0) ans+=d[i];
22          }
23          out.println(ans-1);
24      }
25
26      static FastReader in = new FastReader();
27      static PrintWriter out = new PrintWriter(System.out);
28
29      static class FastReader {
30          static BufferedReader br;
31          static StringTokenizer st;
32
33          FastReader() {
34              br = new BufferedReader(new InputStreamReader(System.in));

```

```

35     }
36
37     String next() {
38         String str = "";
39         while(st==null||!st.hasMoreElements()) {
40             try {
41                 str = br.readLine();
42             } catch(IOException e) {
43                 throw new RuntimeException(e);
44             }
45             st = new StringTokenizer(str);
46         }
47         return st.nextToken();
48     }
49
50     int nextInt() {
51         return Integer.parseInt(next());
52     }
53
54     double nextDouble() {
55         return Double.parseDouble(next());
56     }
57
58     long nextLong() {
59         return Long.parseLong(next());
60     }
61
62 }
63 }

```

- Python

```

1  import sys
2  input = sys.stdin.read
3
4  def solve():
5      data = input().splitlines()
6      n = int(data[0])
7      ans = 0
8      a = [0] * (n + 1)
9      d = [0] * (n + 1)
10
11     for i in range(1, n + 1):
12         a[i] = int(data[i])
13         d[i] = a[i] - a[i - 1]
14         if d[i] > 0:
15             ans += d[i]
16
17     print(ans - 1)
18
19 if __name__ == "__main__":
20     solve()
21

```