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++

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N=2e5+10;
int a[N];
int n,m;
int main(){
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
    for(int i=1;i<=m;i++){
        int l,r,d;
        scanf("%d%d%d",&l,&r,&d);
        for(int j=l;j<=r;j++){
            a[j]+=d;
        }
    }
    for(int i=1;i<=n;i++) printf("%d ",a[i]);
}
```

- Java

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int N = 200010;  // 与 C++ 中的数组大小一致
        int[] a = new int[N];
        int n = scanner.nextInt();
        int m = scanner.nextInt();

        // 输入数组
        for (int i = 1; i <= n; i++) {
```

```
            a[i] = scanner.nextInt();
        }

        // 处理每个操作
        for (int i = 1; i <= m; i++) {
            int l = scanner.nextInt();
            int r = scanner.nextInt();
            int d = scanner.nextInt();
            for (int j = l; j <= r; j++) {
                a[j] += d;
            }
        }

        // 输出结果
        for (int i = 1; i <= n; i++) {
            System.out.print(a[i] + " ");
        }
        scanner.close();
    }
}

```

- Python

```
import sys
input = sys.stdin.read

# 读取所有输入
data = input().split()
index = 0

# 读取 n 和 m
n = int(data[index])
m = int(data[index + 1])
index += 2

# 初始化数组
N = 200010
a = [0] * N

# 读取数组元素
for i in range(1, n + 1):
    a[i] = int(data[index])
    index += 1

# 处理每个操作
for _ in range(m):
    l = int(data[index])
    r = int(data[index + 1])
    d = int(data[index + 2])
    index += 3
    for j in range(l, r + 1):
        a[j] += d

# 输出结果
sys.stdout.write(" ".join(map(str, a[1:n + 1])) + "\n")

```

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

- 原数组的差分数组，等于前缀和数组的相邻元素的差。
- 差分数组的前缀和，等于原数组。

差分数组的核心优势在于将**区间操作**降维到**单点操作**，使复杂度从 $O(n \cdot m)$ 降低到 $O(n+m)$，适合处理大量区间操作的问题。
其与一维前缀和关系为：

原数组求前缀和 = 差分数组

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++

```cpp
#include <bits/stdc++.h>
using namespace std;
#define endl '\n'
const int N = 1e6 + 10;
int a[N];
int diff[N];

void solve() {
```

```
    int n, m;
    scanf("%d %d", &n, &m);
    for (int i = 1; i <= n; i++) {
        scanf("%d", &a[i]);
        diff[i] = a[i] - a[i - 1];
    }
    while (m--) {
        int l, r, x;
        scanf("%d %d %d", &l, &r, &x);
        diff[l] += x;
        diff[r + 1] -= x;
    }
    for (int i = 1; i <= n; i++) {
        diff[i] += diff[i - 1];
        printf("%d ", diff[i]);
    }
    printf("\n");
}

int main() {
    int t = 1;
    while (t--) {
        solve();
    }
    return 0;
}

```

- Java

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // 输入 n 和 m
        int n = sc.nextInt();
        int m = sc.nextInt();

        long[] a = new long[n + 1];
        long[] diff = new long[n + 2];

        // 读取数组 a 并构造差分数组
        for (int i = 1; i <= n; i++) {
            a[i] = sc.nextLong();
            diff[i] = a[i] - a[i - 1];
        }

        // 处理 m 次操作
        while (m-- > 0) {
            int l = sc.nextInt();
            int r = sc.nextInt();
            long x = sc.nextLong();

            diff[l] += x;
            diff[r + 1] -= x;
```

```
        }

        // 计算最终数组并输出
        for (int i = 1; i <= n; i++) {
            diff[i] += diff[i - 1];
            System.out.print(diff[i] + " ");
        }
        System.out.println();

        sc.close();
    }
}
```

- Python

```python
import sys
input = sys.stdin.read
output = sys.stdout.write

def solve():
    # 读取所有输入数据
    data = input().splitlines()

    # 解析 n 和 m
    n, m = map(int, data[0].split())

    # 初始化数组 a 和差分数组 diff
    a = [0] * (n + 1)
    diff = [0] * (n + 2)

    # 读取数组 a 并构造差分数组
    for i in range(1, n + 1):
        a[i] = int(data[i])
        diff[i] = a[i] - a[i - 1]

    # 处理 m 次操作
    for i in range(n + 1, n + 1 + m):
        l, r, x = map(int, data[i].split())
        diff[l] += x
        if r + 1 <= n:
            diff[r + 1] -= x

    # 计算最终数组并输出
    result = []
    for i in range(1, n + 1):
        diff[i] += diff[i - 1]
        result.append(str(diff[i]))

    # 使用 ACM 输出
    output(" ".join(result) + "\n")

if __name__ == "__main__":
    solve()

```

小蓝的操作

```cpp
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int N = 1e5 + 10;
int n;
int a[N];
int d[N];

void solve() {
    cin >> n;
    long long ans = 0;
    for (int i = 1; i <= n; i++) {
        cin >> a[i];
        d[i] = a[i] - a[i - 1];
        if (d[i] > 0) ans += d[i];
    }
    cout << ans - 1 << "\n";
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    solve();
    return 0;
}

```

- Java

```java
import java.io.*;
import java.util.*;

public class Main {
    static int N = (int)(1e5+10);
    static int n;
    static int[] a = new int[N];
    static int[] d = new int[N];

    public static void main(String[] args) {
        solve();
        out.flush();
    }

    static void solve() {
        n = in.nextInt();
        long ans = 0;
        for(int i=1;i<=n;i++){
            a[i] = in.nextInt();
            d[i] = a[i]-a[i-1];
            if(d[i]>0) ans+=d[i];
        }
        out.println(ans-1);
    }

    static FastReader in = new FastReader();
    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
```

```python
if __name__ == "__main__":
    solve()

```