

一维序列区间求和问题

给定一个长度为 n 的序列 a 。

再给定 q 组查询，对于每次查询：

给定一对 l, r ，你需要输出 $\sum_{i=l}^r a_i$ 的结果。

输入格式

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

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 q 行，每行输入 2 个正整数 l, r 。 ($1 \leq l \leq r \leq n$)。

- C++

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  const int N = 1e5 + 10; // 数组最大长度
4  int a[N]; // 存储数组元素
5  int n, q; // 数组长度和查询次数
6
7  int main() {
8      // 输入数组长度和查询次数
9      scanf("%d%d", &n, &q);
10     for (int i = 1; i <= n; i++) {
11         scanf("%d", &a[i]); // 输入数组元素
12     }
13
14     // 处理每个查询
15     for (int i = 1; i <= q; i++) {
16         int l, r;
17         scanf("%d%d", &l, &r); // 查询区间 [l, r]
18         int sum = 0; // 初始化区间和
19         for (int j = l; j <= r; j++) {
20             sum = sum + a[j]; // 累加区间内的元素
21         }
22         printf("%d\n", sum); // 输出区间和
23     }
24 }
25
```

- 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          final int N = 100010; // 数组最大长度
7          int[] a = new int[N];
8          int n = sc.nextInt(); // 数组长度
9          int q = sc.nextInt(); // 查询次数
10
```

```

11 // 输入数组元素
12 for (int i = 1; i <= n; i++) {
13     a[i] = sc.nextInt();
14 }
15
16 // 处理每个查询
17 for (int i = 0; i < q; i++) {
18     int l = sc.nextInt(); // 查询左端点
19     int r = sc.nextInt(); // 查询右端点
20     int sum = 0; // 初始化区间和
21     for (int j = l; j <= r; j++) {
22         sum += a[j]; // 累加区间内的元素
23     }
24     System.out.println(sum); // 输出区间和
25 }
26
27 sc.close();
28 }
29 }
30

```

- Python

```

1 n, q = map(int, input().split()) # 输入数组长度和查询次数
2 a = [0] + list(map(int, input().split())) # 一行读取数组并从索引 1 开始存储
3
4 for _ in range(q):
5     l, r = map(int, input().split()) # 查询左端点和右端点
6     total_sum = 0 # 初始化区间和
7     for i in range(l, r + 1): # 暴力遍历区间 [l, r]
8         total_sum += a[i]
9     print(total_sum) # 输出区间和

```

一维前缀和

一维前缀和便是能高效解决该问题的一种算法，其原理是通过预处理出 a 的前 n 项和数组，使得在每次查询区间和时，可以通过 **常数时间** 的计算快速得出结果。以下是实现的具体步骤：

1. 构建前缀和数组：

- 定义一个数组 `prefix`，其中 `prefix[i]` 表示数组 a 的前 i 项元素的和。
- 初始化 `prefix[0] = 0`，然后从 1 到 n 依次累加计算前缀和。

数学公式表示为：

$$\text{prefix}[i] = \text{prefix}[i - 1] + a[i]$$

2. 区间和查询：

- 对于任意的查询区间 (l, r) ，其和可以通过以下公式计算：

$$\text{sum}[l, r] = \text{prefix}[r] - \text{prefix}[l - 1]$$

这样，我们在查询时只需进行两次数组访问和一次减法操作，时间复杂度为 $O(1)$ 。

一维前缀和 <https://www.lanqiao.cn/problems/18437/learning/>

- C++

```

1 #include <iostream>

```

```

2  using namespace std;
3  int a[100010];
4  int s[100010];
5  int main()
6  {
7      int n,q;
8      scanf("%d %d",&n,&q);
9      for(int i=1;i<=n;i++){
10         scanf("%d",&a[i]);
11         s[i]=a[i]+s[i-1];
12     }
13     for(int i=1;i<=q;i++){
14         int l,r;
15         scanf("%d%d",&l,&r);
16         printf("%d\n",s[r]-s[l-1]);
17     }
18     return 0;
19 }

```

- 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          int n = sc.nextInt(); // 数组长度
8          int q = sc.nextInt(); // 查询次数
9
10         int[] a = new int[n + 1]; // 数组从索引 1 开始
11         int[] s = new int[n + 1]; // 前缀和数组
12
13         // 输入数组并计算前缀和
14         for (int i = 1; i <= n; i++) {
15             a[i] = sc.nextInt();
16             s[i] = s[i - 1] + a[i];
17         }
18
19         // 处理查询
20         for (int i = 0; i < q; i++) {
21             int l = sc.nextInt();
22             int r = sc.nextInt();
23             System.out.println(s[r] - s[l - 1]); // 输出区间和
24         }
25
26         sc.close();
27     }
28 }
29

```

- Python

```

1  n, q = map(int, input().split()) # 输入数组长度和查询次数
2  a = [0] + list(map(int, input().split())) # 数组从索引 1 开始存储
3  s = [0] * (n + 1) # 前缀和数组
4
5  # 计算前缀和
6  for i in range(1, n + 1):
7      s[i] = s[i - 1] + a[i]
8
9  # 处理查询
10 for _ in range(q):
11     l, r = map(int, input().split()) # 输入查询范围
12     print(s[r] - s[l - 1]) # 输出区间和
13

```

求和 <https://www.lanqiao.cn/problems/2080/learning/>

- C++

```

1  #include <iostream>
2  using namespace std;
3
4  const int N = 200010; // 数组最大长度
5  int a[N];
6  long long s[N]; // 使用 long long 存储前缀和, 避免溢出
7
8  typedef long long ll;
9
10 // 计算区间和函数
11 ll calc(int l, int r) {
12     return s[r] - s[l - 1]; // 返回区间 [l, r] 的和
13 }
14
15 int main() {
16     int n;
17     scanf("%d", &n); // 输入数组长度
18
19     // 输入数组并计算前缀和
20     for (int i = 1; i <= n; i++) {
21         scanf("%d", &a[i]);
22         s[i] = a[i] + s[i - 1]; // 计算前缀和
23     }
24
25     ll ans = 0;
26
27     // 计算结果 ans
28     for (int i = 1; i < n; i++) {
29         ans = ans + calc(i + 1, n) * a[i]; // 累加结果, 使用 calc 函数计算区间和
30     }
31
32     // 输出结果
33     printf("%lld\n", ans);
34     return 0;
35 }
36

```

- Java

```

1  import java.util.Scanner;
2
3  public class Main {
4
5      public static void main(String[] args) {
6          Scanner sc = new Scanner(System.in);
7
8          int n = sc.nextInt(); // 输入数组长度
9          int[] a = new int[n + 1];
10         long[] s = new long[n + 1]; // 使用 long 存储前缀和，避免溢出
11
12         // 输入数组并计算前缀和
13         for (int i = 1; i <= n; i++) {
14             a[i] = sc.nextInt();
15             s[i] = a[i] + s[i - 1]; // 计算前缀和
16         }
17
18         long ans = 0;
19
20         // 计算结果 ans
21         for (int i = 1; i < n; i++) {
22             ans += calc(i + 1, n, s) * a[i]; // 累加结果，使用 calc 函数计算区间
和
23         }
24
25         // 输出结果
26         System.out.println(ans);
27     }
28
29     // 计算区间和函数
30     public static long calc(int l, int r, long[] s) {
31         return s[r] - s[l - 1]; // 返回区间 [l, r] 的和
32     }
33 }
34

```

- Python

```

1  def calc(l, r, s):
2      return s[r] - s[l - 1] # 返回区间 [l, r] 的和
3
4  def main():
5      n = int(input()) # 输入数组长度
6      a = [0] * (n + 1) # 数组长度为 n+1
7      s = [0] * (n + 1) # 前缀和数组
8
9      # 输入数组并计算前缀和
10     a[1:] = list(map(int, input().split())) # 一次性读取并转换为整数数组
11     for i in range(1, n + 1):
12         s[i] = a[i] + s[i - 1] # 计算前缀和
13
14     ans = 0
15
16     # 计算结果 ans
17     for i in range(1, n):
18         ans += calc(i + 1, n, s) * a[i] # 累加结果，使用 calc 函数计算区间和
19

```

```
20     # 输出结果
21     print(ans)
22
23 if __name__ == "__main__":
24     main()
25
```