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++

```cpp
#include<bits/stdc++.h>
using namespace std;
const int N = 1e5 + 10; // 数组最大长度
int a[N]; // 存储数组元素
int n, q; // 数组长度和查询次数

int main() {
    // 输入数组长度和查询次数
    scanf("%d%d", &n, &q);
    for (int i = 1; i <= n; i++) {
        scanf("%d", &a[i]); // 输入数组元素
    }

    // 处理每个查询
    for (int i = 1; i <= q; i++) {
        int l, r;
        scanf("%d%d", &l, &r); // 查询区间 [l, r]
        int sum = 0; // 初始化区间和
        for (int j = l; j <= r; j++) {
            sum = sum + a[j]; // 累加区间内的元素
        }
        printf("%d\n", sum); // 输出区间和
    }
}

```

- Java

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        final int N = 100010; // 数组最大长度
        int[] a = new int[N];
        int n = sc.nextInt(); // 数组长度
        int q = sc.nextInt(); // 查询次数

```

```java
        // 输入数组元素
        for (int i = 1; i <= n; i++) {
            a[i] = sc.nextInt();
        }

        // 处理每个查询
        for (int i = 0; i < q; i++) {
            int l = sc.nextInt(); // 查询左端点
            int r = sc.nextInt(); // 查询右端点
            int sum = 0; // 初始化区间和
            for (int j = l; j <= r; j++) {
                sum += a[j]; // 累加区间内的元素
            }
            System.out.println(sum); // 输出区间和
        }

        sc.close();
    }
}

```

- Python

```python
n, q = map(int, input().split())  # 输入数组长度和查询次数
a = [0] + list(map(int, input().split()))  # 一行读取数组并从索引 1 开始存储

for _ in range(q):
    l, r = map(int, input().split())  # 查询左端点和右端点
    total_sum = 0  # 初始化区间和
    for i in range(l, r + 1):  # 暴力遍历区间 [l, r]
        total_sum += a[i]
    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++

```cpp
#include <iostream>
```

```cpp
using namespace std;
int a[100010];
int s[100010];
int main()
{
  int n,q;
  scanf("%d %d",&n,&q);
  for(int i=1;i<=n;i++){
    scanf("%d",&a[i]);
    s[i]=a[i]+s[i-1];
  }
  for(int i=1;i<=q;i++){
    int l,r;
    scanf("%d%d",&l,&r);
    printf("%d\n",s[r]-s[l-1]);
  }
  return 0;
}
```

- Java

```java
import java.util.Scanner;

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

        int n = sc.nextInt(); // 数组长度
        int q = sc.nextInt(); // 查询次数

        int[] a = new int[n + 1]; // 数组从索引 1 开始
        int[] s = new int[n + 1]; // 前缀和数组

        // 输入数组并计算前缀和
        for (int i = 1; i <= n; i++) {
            a[i] = sc.nextInt();
            s[i] = s[i - 1] + a[i];
        }

        // 处理查询
        for (int i = 0; i < q; i++) {
            int l = sc.nextInt();
            int r = sc.nextInt();
            System.out.println(s[r] - s[l - 1]); // 输出区间和
        }

        sc.close();
    }
}

```

- Python

```python
n, q = map(int, input().split())  # 输入数组长度和查询次数
a = [0] + list(map(int, input().split()))  # 数组从索引 1 开始存储
s = [0] * (n + 1)  # 前缀和数组

# 计算前缀和
for i in range(1, n + 1):
    s[i] = s[i - 1] + a[i]

# 处理查询
for _ in range(q):
    l, r = map(int, input().split())  # 输入查询范围
    print(s[r] - s[l - 1])  # 输出区间和

```

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

- C++

```cpp
#include <iostream>
using namespace std;

const int N = 200010; // 数组最大长度
int a[N];
long long s[N]; // 使用 long long 存储前缀和，避免溢出

typedef long long ll;

// 计算区间和函数
ll calc(int l, int r) {
    return s[r] - s[l - 1]; // 返回区间 [l, r] 的和
}

int main() {
    int n;
    scanf("%d", &n); // 输入数组长度

    // 输入数组并计算前缀和
    for (int i = 1; i <= n; i++) {
        scanf("%d", &a[i]);
        s[i] = a[i] + s[i - 1]; // 计算前缀和
    }

    ll ans = 0;

    // 计算结果 ans
    for (int i = 1; i < n; i++) {
        ans = ans + calc(i + 1, n) * a[i]; // 累加结果，使用 calc 函数计算区间和
    }

    // 输出结果
    printf("%lld\n", ans);
    return 0;
}

```

- Java

```java
import java.util.Scanner;

public class Main {

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

        int n = sc.nextInt(); // 输入数组长度
        int[] a = new int[n + 1];
        long[] s = new long[n + 1]; // 使用 long 存储前缀和，避免溢出

        // 输入数组并计算前缀和
        for (int i = 1; i <= n; i++) {
            a[i] = sc.nextInt();
            s[i] = a[i] + s[i - 1]; // 计算前缀和
        }

        long ans = 0;

        // 计算结果 ans
        for (int i = 1; i < n; i++) {
            ans += calc(i + 1, n, s) * a[i]; // 累加结果，使用 calc 函数计算区间和
        }

        // 输出结果
        System.out.println(ans);
    }

    // 计算区间和函数
    public static long calc(int l, int r, long[] s) {
        return s[r] - s[l - 1]; // 返回区间 [l, r] 的和
    }
}

```

- Python

```python
def calc(l, r, s):
    return s[r] - s[l - 1]  # 返回区间 [l, r] 的和

def main():
    n = int(input())  # 输入数组长度
    a = [0] * (n + 1)  # 数组长度为 n+1
    s = [0] * (n + 1)  # 前缀和数组

    # 输入数组并计算前缀和
    a[1:] = list(map(int, input().split()))  # 一次性读取并转换为整数数组
    for i in range(1, n + 1):
        s[i] = a[i] + s[i - 1]  # 计算前缀和

    ans = 0

    # 计算结果 ans
    for i in range(1, n):
        ans += calc(i + 1, n, s) * a[i]  # 累加结果，使用 calc 函数计算区间和

```

```
    # 输出结果
    print(ans)

if __name__ == "__main__":
    main()

```