Koishi 与数列划分

题意简述

把一段长度为 n 的数列划分为两段，使得这两段的和相等，求划分方案数。

题目解析

第一个想法肯定是枚举所有可能的划分点，然后暴力求和，判断左右两边的和是否相等。

然而在 $1 \le n \le 10^5$ 的数据范围下，这样是会超时的。

我们想想怎么优化，发现枚举划分点的时间已经不能省了，得从求和这里入手。

能不能通过一些预处理，加速求和的过程呢？

这里我们引入 **前缀和** 的概念。

我们记整个数列是 $a_1, a_2, a_3, \dots, a_n$，那么前缀和可以被表述为：

$$s_i = \sum_{j=1}^{i} a_j$$

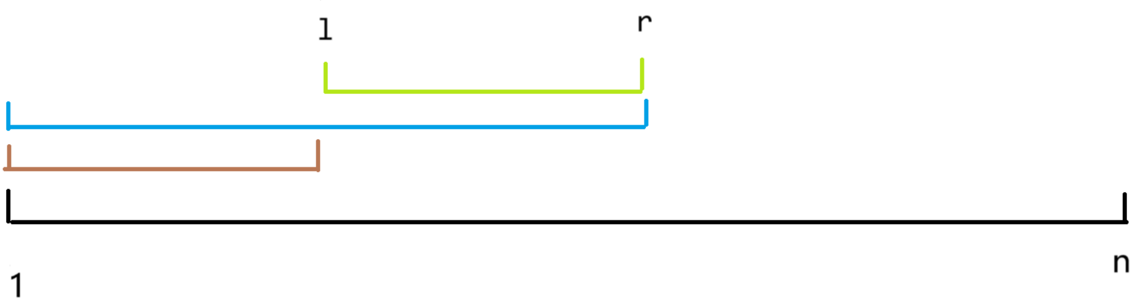
于是我们如果要求区间 $[l, r]$ 上数的和，可以进行如下转化：

$$\sum_{i=l}^{r} a_i = \sum_{i=1}^{l-1} a_i - \sum_{i=1}^{r} a_i = s_{l-1} - s_r$$

我相信有人肯定看不来上面这段，所以我们使用形象化的图形语言来表述一下这件事。

前缀和，顾名思义，就是把从开头到位置 i 上的数全求和。

如下图所示，当你要求 $[l, r]$ 这段区间上的数的和（绿色段）的时候，你可以通过把 $[1, r]$ 上的数的和（蓝色段）减去 $[1, l-1]$ 这一段上的数的和（红色段）来完成这件事。



从此你就剩下了一个循环的时间。你的代码自然也就能通过这题了。

具体的实现就是，对于每个位置，先求出它的前缀和 s_i（可以通过一个循环，用 $s_i = s_{i-1} + a_i$ 来求解）。

然后枚举每个位置 i 作为划分点，前半段的和就是 s_i，后半段的和就是 $s_n - s_i$。判断出现了几次 $s_n - s_i = s_i$ 的情况即可。

代码

```
#include<stdio.h>
#include<string.h>

const int mxn = 1e5+5;

int a[mxn],s[mxn];

int main()
{
    int n,ans=0;
    scanf("%d",&n);
    memset(s,0,sizeof(s));  //用来给数组全赋值为 0
    for(int i=1;i<=n;i++)
        scanf("%d",&a[i]);
    for(int i=1;i<=n;i++){
        s[i]=s[i-1]+a[i];   //对前缀和的求解
    }
    for(int f=1;f<n;f++)
        if(s[f]==s[n]-s[f]) //判断并统计划分方案数
            ans++;
    printf("%d",ans);
    return 0;
}
```

补充

没别的，出这题时就是想考一下前缀和，毕竟这算一个很基础且常见的优化，了解一下挺好的。

同时也表现出 **对数据的预处理** 操作是非常重要的，有时候能很大程度优化代码的运行效率。