题目限制

1000ms 256M

题目描述

一开始，你有一个长为n的序列$a_1, a_2, a_3 \dots a_n$。

你觉得它有些长了，所以决定将它按原来的顺序分割成若干**连续**的段。但分割时要保证第i段中的数字之和是i的倍数。

两个分割方案被视为不同当且仅当存在一个数字，在两个方案中属于不同的段。请你输出满足要求的分割方案数 $mod\ 10^9 + 7$的值。

输入格式

第一行一个正整数n。

第二行n个正整数表示序列的每一项。

输出格式

输出一个非负整数表示你的答案。

输入输出样例

样例输入 #1

```
4
1 2 3 4
```

样例输出 #1

```
3
```

样例解释 #1

合法的三种方案为：

1 | 2 | 3 | 4（第一段为1，第二段为2，第三段为3，第四段为4）

1 2 3 | 4（第一段为1 2 3 ，第二段为4）

1 2 3 4 （不做分割）

以下是不合法的分割方案

1 2 | 3 |4（第二段和为3，不是2的倍数）

1 3 |2 4 （调换了位置，不合法）

样例 #2

样例输入 #2

```
5
8 6 3 3 3
```

样例输出 #2

```
5
```

数据范围与约定

- 对于 20% 的数据，保证$1 \leq n \leq 20$
- 对于 50% 的数据，保证$1 \leq n \leq 500$
- 对于 100% 的数据，保证$1 \leq n \leq 3000$，$1 \leq a_i \leq 10^{15}$。

题意解释

你需要将一个数组分割成若干个连续的段，使得第i段内元素之和是i的倍数。询问方案数。

知识点提炼

动态规划、前缀和

核心解题思路

思路1：枚举划分方案(10pts)

考虑使用dfs枚举第i个位置与第$i+1$个位置之间是否进行划分并判断是否合法，复杂度为2^n。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
ll n,a[3010],v[3010],ans;
void dfs(int d,int i,ll sum)//当前深度、当前第几段、当前和
{
    if(d==n)//到了dfs终点
    {
        if(sum%i==0)//如果最后一段满足要求则答案赠一
            ans++;
        return ;//必须返回
    }
    if(sum%i==0)//允许分段
        dfs(d+1,i+1,a[d+1]);//新的一段是第i+1段，新的和为a[d]+1
    dfs(d+1,i,sum+a[d+1]);//不分段，仍然是第i段，新的和为sum+a[d+1]
}
int main()
{
    scanf("%lld",&n);//读入n
    for(int i=1;i<=n;i++)
        scanf("%lld",&a[i]);//读入数组
    dfs(1,1,a[1]);//开始dfs
```

```
    cout<<ans;
}
```

时间复杂度$O(2^n)$，期望得分20分。

思路2：动态规划(20pts)

考虑动态规划，令$f[i][x]$表示以$a[x]$作为第i段的结尾的方案数，则初始状态为$f[0][0]=1$。

$f[i][x]=\sum f[i-1][y]$，其中y满足$y+1$到x元素之和是i的倍数。答案为$\sum_{i=1}^{n} f[i][n]$。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
ll n,a[3010],sum[3010],f[3010][3010],ans,mod=1e9+7;
//元素数量、原数组、前缀和数组、dp数组、统计答案用、取模用
int main()
{
    scanf("%lld",&n);//读入n
    for(int i=1;i<=n;i++)
    {
        scanf("%lld",&a[i]);//读入数列
        sum[i]=sum[i-1]+a[i];//预处理前缀和数组(可以省略)
    }
    f[0][0]=1;//初始状态
    for(int i=1;i<=n;i++)
    {
        for(int x=1;x<=n;x++)
        {
            for(int y=0;y<x;y++)
                if((sum[x]-sum[y])%i==0)//如果区间之和满足是i的倍数
                    f[i][x]=(f[i][x]+f[i-1][y])%mod;//则将f[i-1][y]更新至f[i][x]中
        }
        ans=(ans+f[i][n])%mod;//统计答案
    }
    cout<<ans;//输出答案
    return 0;
}
```

时间复杂度$O(n^3)$，期望得分50分。

思路3：优化的动态规划

注意到对于$f[i][x]$，需要满足要求的$f[i-1][y]$。不妨设前缀和数组$sum[i]=sum[i-1]+a[i]$这个要求就可以等价于要求$sum[y-1]=sum[x]$。于是我们边转移边处理$g[v]=\sum f[i-1][y]$其中$sum[y]\%i==v$，将满足要求的$f[i-1][y]$之和预先存入数组中，状态直接转移即可。

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
ll n,a[3010],sum[3010],f[3010][3010],ans,mod=1e9+7,g[3010];
int main()
{
    scanf("%lld",&n);//读入n
```

```
    for(int i=1;i<=n;i++)
    {
        scanf("%lld",&a[i]);//读入数列
        sum[i]=sum[i-1]+a[i];//预处理前缀和数组
    }
    f[0][0]=1;//初始状态为第0段，前缀和为0的方案数为1
    for(int i=1;i<=n;i++)
    {
        for(int j=0;j<i;j++)
            g[j]=0;//清空g数组
        for(int x=1;x<=n;x++)
        {
            g[sum[x-1]%i]=(g[sum[x-1]%i]+f[i-1][x-1])%mod;//将满足f[i-1][x-1]放入
g[sum[x-1]%i]中
            f[i][x]=g[sum[x]%i];//状态更新时直接拿取所需的方案数
        }
        ans=(ans+f[i][n])%mod;//更新答案
    }
    cout<<ans;//输出答案
}
```

时间复杂度$O(n^2)$，期望得分100分。

本题易错点

容易忘记取模导致溢出。

容易在读入a_i或进行前缀和时忘记开long long。

利用前缀和进行优化时，容易误判什么时候更新g数组，什么时候转移状态。

标准代码

C++

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
ll n,a[3010],sum[3010],f[3010][3010],ans,mod=1e9+7,g[3010];
int main()
{
    scanf("%lld",&n);
    for(int i=1;i<=n;i++)
    {
        scanf("%lld",&a[i]);
        sum[i]=sum[i-1]+a[i];
    }
    f[0][0]=1;
    for(int i=1;i<=n;i++)
    {
        for(int j=0;j<i;j++)
            g[j]=0;
        for(int x=1;x<=n;x++)
        {
            g[sum[x-1]%i]=(g[sum[x-1]%i]+f[i-1][x-1])%mod;
```

```
            f[i][x]=g[sum[x]%i];
        }
        ans=(ans+f[i][n])%mod;
    }
    cout<<ans;
}
```