由$i+A_i\times x$不难得知我们从一个点出发最坏是每个点都走一次，最好是一步直接走到终点，此时不难根据easyversion的解法想到$o(n^2)$的解法，显然不行，此处我们可以考虑优化成$O(n\sqrt{n})$，此处讲述一个**根号分治**的概念和最基础的应用，更深层次的应用请自行学习，假设一个值$step=\sqrt{n}$，按照如下情况考虑

- $A_i>=step$，此时我们暴力遍历$A_i,2\times A_i,3\times A_i....$，设$dp[i]$为最后踩到第i个点的方案数使得$dp[A_i],dp[2\times A_i],dp[3\times A_i]$加上$dp[A_i]$,由于$A_i>step$，复杂度最多为$o(\sqrt{n})$
- $A_i<step$，此时我们此时暴力遍历会有$O(n)$的复杂度，所以无法暴力，但是由于$A_i<step$，不难得知$A_i^2<N$，此时可以用一个大小为$step\times step$的sum数组。$sum[i][j]$的数组来记录基础值为i，位置为$j\%i$的方案数，那么当前$dp[i]=\sum_{j=1}^{step}sum[j][i\%j]$，同时更新$sum->sum[a[i]][i\%a[i]]=(sum[a[i]][i\%a[i]]+dp[i])\%mod;$,不难看出求dp操作为o(1)，更新操作为$O(\sqrt{n})$，复杂度为$o(\sqrt{n})$
  所以时间复杂度为$o(n\sqrt{n})$，空间复杂度为$o(n)$

```
#define Nmax 200010
const int mod=998244353,S=(int)sqrt(Nmax);
int sum[S+5][S+5],a[Nmax],dp[Nmax];
signed main()
{
    IOS;
    int n,ans=0;
    cin>>n;
    dp[1]=1;
    for(int i=1;i<=n;i++)
    {
        cin>>a[i];
        for(int j=1;j<=S;j++)
            dp[i]=(dp[i]+sum[j][i%j])%mod;
        if(a[i]<=S)
            sum[a[i]][i%a[i]]=(sum[a[i]][i%a[i]]+dp[i])%mod;
        else
        {
            for(int j=i+a[i];j<=n;j+=a[i])
                dp[j]=(dp[j]+dp[i])%mod;
        }
        ans=(ans+dp[i])%mod;
    }
    cout<<ans;
}
```