回响旋律 题解

Subtask 1&2

时间复杂度 $O(nm)$。

模拟即可。

本 Subtask 的标程为 `melody_sol1.cpp`。

Subtask 3

时间复杂度 $O(1)$。

c_i 相等，那么我们可以认为是 c_1 重复演奏了 nm 次。

我们考虑结束时的演奏回响度 a 由谁的贡献构成。

- 读入的初始 a，贡献了一次。
- 读入的初始 b，每演奏一个音符就会产生一次贡献，所以贡献了 nm 次。
- 读入的c。

  可以发现，由于执行顺序，演奏第一个音符，不会立刻对 a 产生贡献。但在之后的所有 $nm-1$ 次演奏中，会对 a 产生贡献。

  同理，演奏第二个音符，不会立刻对 a 产生贡献。但在此后的 $nm-2$ 次演奏中会产生贡献。

  由于 c_i 固定，因此可以合并。通过等差数列求和公式可知，这部分的贡献为 $\frac{(nm-1)\times(nm)}{2}\times c_i$。

本 Subtask 的标程为 `melody_sol2.cpp`。

Subtask 4

时间复杂度 $O(n+m)$。

一首曲子会重复演奏 m 次，但是每一次演奏的是同一首曲子。

我们就看演奏一次曲子时，a 和 b 会怎么变。

b 很容易发现，一次演奏结束时，b 会加上 c_i 的总和。

而联想到上一个 Subtask 的结论，a 会加上：

- 演奏前的 b，贡献 n 次。
- c_i，贡献 $n-i$ 次。

预处理好这些贡献，模拟 m 次即可。

本 Subtask 的标程为 `melody_sol3.cpp`。

Subtask 5-1

时间复杂度 $O(3^3\times(n+\log m))$，已足以通过本题。

矩阵乘法快速幂。

起始矩阵为 $\begin{bmatrix}a & b & 1\end{bmatrix}$，每演奏一个回响度为 c_i 的音符，就乘上 $\begin{bmatrix}1 & 0 & 0\\1 & 1 & 0\\0 & c_i & 1\end{bmatrix}$。

根据矩阵乘法的结合律，这可以用矩阵乘法快速幂预处理出来后乘到起始矩阵上，结果矩阵的第一个数即为答案。

本 Subtask 的标程为 `melody_sol4.cpp`。

```
#include<cstdio>
#include<algorithm>
using namespace std;
const int MAXN=4e5;
const long long MOD=998244353;
int n;
long long m;
long long c[MAXN+5];
long long a,b;
struct Matrix
{
    int r,c;
    long long data[3][3];
    Matrix()
    {
        r=c=0;
    }
    Matrix(int ir,int ic)
    {
        r=ir;
        c=ic;
        for(int i=0;i<r;i++)
        {
            for(int j=0;j<c;j++)
            {
                data[i][j]=0;
            }
        }
    }
}st,mc;//st是起始矩阵，mc是中间矩阵。
Matrix operator *(Matrix a,Matrix b)//矩阵乘法
{
    Matrix res(a.r,b.c);
    for(int i=0;i<a.r;i++)
    {
        for(int j=0;j<b.c;j++)
        {
            for(int k=0;k<a.c;k++)
            {
                res.data[i][j]=(res.data[i][j]+(a.data[i][k]*b.data[k]
[j]%MOD))%MOD;
            }
        }
    }
    return res;
}
```

```
Matrix qpow(Matrix a,long long b)
{
    Matrix res=Matrix(3,3);
    for(int i=0;i<3;i++)//这个矩阵无论乘上哪个矩阵都等于原矩阵，相当于实数乘法中的1.
    {
        res.data[i][i]=1;
    }
    while(b)
    {
        if(b&1)
        {
            res=res*a;
        }
        a=a*a;
        b>>=1;
    }
    return res;
}
int main()
{
    freopen("melody.in","r",stdin);
    freopen("melody.out","w",stdout);
    st=Matrix(1,3);
    st.data[0][2]=1;
    mc=Matrix(3,3);
    for(int i=0;i<3;i++)
    {
        mc.data[i][i]=1;
    }
    scanf("%d%lld%lld%lld",&n,&m,&st.data[0][0],&st.data[0][1]);
    for(int i=1;i<=n;i++)
    {
        scanf("%lld",&c[i]);
        Matrix nowmc=Matrix(3,3);
        nowmc.data[0][0]=nowmc.data[1][0]=nowmc.data[1][1]=nowmc.data[2][2]=1;
        nowmc.data[2][1]=c[i];
        mc=mc*nowmc;
    }
    printf("%lld",(st*qpow(mc,m)).data[0][0]);
    return 0;
}
```

Subtask 5-2

时间复杂度 $O(n)$，比上一个做法有更优的**理论**复杂度。

结合之前拆贡献的想法，我们可以发现，音符 c_i 的贡献次数为，首项 $n-i$，末项 $nm-i$，项数 m 的等差数列求和。

本 Subtask 的标程为 `melody_sol5.cpp`。

```
#include<cstdio>
#include<algorithm>
using namespace std;
const int MAXN=4e5;
```

```cpp
const long long MOD=998244353;
int n;
long long m;
long long c[MAXN+5];
long long a,b;
long long qpow(long long a,long long b)//快速幂，用于求 2 的逆元
{
    long long res=1;
    while(b)
    {
        if(b&1)
        {
            res=res*a%MOD;
        }
        a=a*a%MOD;
        b>>=1;
    }
    return res;
}
const long long INV2=qpow(2,MOD-2);
int main()
{
    freopen("melody.in","r",stdin);
    freopen("melody.out","w",stdout);
    scanf("%d%lld%lld%lld",&n,&m,&a,&b);
    for(int i=1;i<=n;i++)
    {
        scanf("%lld",&c[i]);
    }
    m%=MOD;//加法和乘法中随便取模
    a=(a+(b*n%MOD*m%MOD))%MOD;
    for(int i=1;i<=n;i++)
    {
        long long st=n-i,ed=n*m-i;//等差数列首项与末项
        st%=MOD;
        ed%=MOD;
        a=(a+(c[i]*(((st+ed)%MOD*((long long)m)%MOD*INV2%MOD))%MOD))%MOD;
    }
    printf("%lld",a);
    return 0;
}
```

版权信息

题解：邓子君

本题改编自广州市赛 2024 D1T1，是原题的加强版。