餐厅清扫

一. 考察内容：

动态规划

二. 题目分析：

[题目大意]

有一个长度为n的序列，包含正整数1-m，连续选择其中一些数，花费是这些数中不同的个数的平方，求出最小花费。

[写题思路]

考虑设状态f[i]为选择前i个数的最小花费，显然有一种转移方式，f[i]=max{f[j]+num[j+1~i]}，j∈[1,i-1]，但是这样显然会超时。考虑对这个转移进行优化。我们会发现，对于任意一个长度为n的序列，最小花费一定不会大于n，而如果一个序列包含m个不同的数，那么这个序列一次选完的花费就是m^2，当$m > \sqrt{n}$时，我们一定不会连续选择这写数，所以，上述转移方程可以被优化为f[i]=max{f[j]+num[j+1~i]}，其中j+1~i中不同的数字个数不能超过$\sqrt{n}$个，这样通常可以过随机数据。

但是对于一些特殊数据，还是有可能被卡成n^2。若f[j]这个状态无法更新f[i]，那么f[j]也无法更新所有的f[i+k]。所以我们可以用链表进行加速。

注：下面的代码没有用双向链表加速。

三. 代码实现：

```
#define _CRT_SECURE_NO_DEPRECATE
/*************************
*创建时间：2018 08 19
*文件类型：源代码文件
*题目来源：COGS
*当前状态：已通过
*备忘录：动态规划
*作者：HtBest
*************************/
#include <stdio.h>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <string.h>
#include <algorithm>
#include <queue>
#include <math.h>
using namespace std;
namespace IO
{
    #define SIZE 10000
    int T=0,end=0;
    char s[SIZE+1];
    int in()
    {
        T=0;
        return end=fread(s,1,SIZE,stdin);
    }
    int read(int &a)
    {
        int i=a=0,flag=1;
        while(1)
```

```
        {
            if(T==end&&!in())return i;
            if(s[T]=='\n'||s[T]=='\r'||s[T]==' ')
                {if(i)return i;}
            else if(s[T]=='-')flag=-1;
            else {++i;a=flag*(flag*a*10+s[T]-'0');}
            ++T;
        }
    }
}
#define MAXN 400001
int n,m,a[MAXN],vis[MAXN],f[MAXN],num;
/* Variable explain:
n:牛个数
m:食品个数
a:牛爱吃的食品
vis:是否访问过
f:状态数组
num:当前食品种类数
*/
void read()
{
    using IO::read;
    freopen("cleanup.in","r",stdin);
    freopen("cleanup.out","w",stdout);
    read(n);
    read(m);
    for(int i=1;i<=n;++i)read(a[i]);
    return;
}
void dp()
{
    //设状态f[i]为前i头牛吃饭的最小花费
    int ls1=sqrt(n);
    for(int i=1;i<=n;++i)f[i]=1e9;
    for(int i=1;i<=n;++i)
    {
        num=0;
        for(int j=1;j<=i&&num<=ls1;++j)//ls1=√n
        {
            if(vis[a[i-j+1]]!=i)vis[a[i-j+1]]=i,++num;
            f[i]=min(f[i],f[i-j]+num*num);
        }
    }
    printf("%d\n",f[n]);
}
int main()
{
    read();
    dp();
    return 0;
}
```

<题目跳转> <查看代码>