Blocks

一. 考察内容：

动态规划 区间DP

二. 题目分析：

[题目大意]

给出一个方块序列，包含金块、银块和铜块，一次可以消除一些连续的块，获得块个数^2的权值，求如何消除所有块使得获得权值最大。

[写题思路]

首先进行预处理，将相邻的同色块和为1个，因为他们不可能被拆开，a[i]为合并后的第i个色块在合并之间表示几个色块，map[i]为合并后第i个色块的颜色。

考虑区间DP，由于本题的一个特性，消除一些块后，会使本身不连续的块变得连续，所以需要在区间DP的基础上加一维状态，设f[i][j][k]为区间[i,j]，j右边有k个与j颜色相同的色块，很显然，我们想获得更大的权值，有两种方案，可以让j与j右边的k个色块一起消掉，也可以把k和j保留，找到[i,j)中与j颜色一样的色块再合并，所以f[i][j][k]=max{f[i][j-1][0]+(a[j]+k)^2,f[i][x-1][0]+f[x+1][r-1][0]+(a[x]+a[j]+k)^2}。

由于转移不是很方便实现，可以考虑用记忆化搜索来写。

三. 代码实现：

```
#define _CRT_SECURE_NO_DEPRECATE
/************************
*创建时间：2018 09 25
*文件类型：源代码文件
*题目来源：POJ
*当前状态：已通过
*备忘录：动态规划 区间DP 记忆化DFS
*作者：HtBest
************************/
#include <stdio.h>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <string.h>
#include <algorithm>
#include <queue>
#include <bitset>
// #include <sys/wait.h>
// #include <sys/types.h>
// #include <unistd.h>
using namespace std;
#define MAXN 201
int n,a[MAXN],map[MAXN],f[MAXN][MAXN][MAXN];
/* Variable explain:
n:分块后个数
a[i]:第i块的元素个数
map[i]:第i块的颜色
*/
void read()
{
    scanf("%d",&n);
    int s[MAXN],_a=0;
    s[0]=-1;
    for(int i=1;i<=n;++i)
    {
        scanf("%d",s+i);
        if(s[i]==s[i-1])
        {
            a[_a]++;
        }
        else a[++_a]=1,map[_a]=s[i];
```

```
    }
    n=_a;
    return;
}
int dp(int l,int r,int len)
{
    if(f[l][r][len])return f[l][r][len];
    if(l==r)return f[l][r][len]=(a[r]+len)*(a[r]+len);
    f[l][r][len]=(a[r]+len)*(a[r]+len)+dp(l,r-1,0);
    for(int i=l;i<r-1;++i)if(map[i]==map[r])f[l][r][len]=max(f[l][r][len],dp(l,i,a[r]
+len)+dp(i+1,r-1,0));
    return f[l][r][len];
}
int main()
{
    // freopen(".in","r",stdin);
    // freopen(".out","w",stdout);
    int T;
    scanf("%d",&T);
    for(int i=1;i<=T;++i)
    {
        read();
        for(int i=1;i<=n;++i)for(int j=1;j<=n;++j)for(int k=0;k<=n;++k)f[i][j][k]=0;
        printf("Case %d: %d\n",i,dp(1,n,0));
    }

    return 0;
}
```

<题目跳转> <查看代码>