



算法基础

郭炜 微博 <http://weibo.com/guoweiofpku>

<http://blog.sina.com.cn/u/3266490431>

刘家瑛 微博 <http://weibo.com/pkuliujiaying>



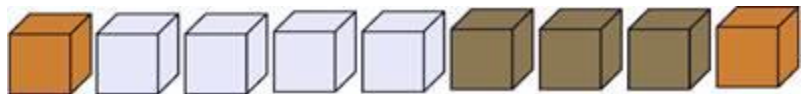
北京大学
PEKING UNIVERSITY

动态规划

方盒游戏

例题：POJ1390 方盒游戏

- N个方盒(box)摆成一排，每个方盒有自己的颜色。连续摆放的同颜色方盒构成一个“大块”(Block)。下图中共有四个大块，每个大块分别有1、4、3、1个方盒



- 玩家每次点击一个方盒，则该方盒所在大块就会消失。若消失的大块中共有k个方盒，则玩家获得 $k*k$ 个积分。
- 请问：给定游戏开始时的状态，玩家可获得的最高积分是多少？

- 输入：第一行是一个整数 $t(1 \leq t \leq 15)$ ，表示共有多少组测试数据。每组测试数据包括两行
 - 第一行是一个整数 $n(1 \leq n \leq 200)$ ，表示共有多少个方盒
 - 第二行包括 n 个整数，表示每个方盒的颜色。这些整数的取值范围是 $[1, n]$
- 输出：对每组测试数据，分别输出该组测试数据的序号、以及玩家可以获得的最高积分

■ 样例输入

2

9

1 2 2 2 2 3 3 3 1

1

1

■ 样例输出

Case 1: 29

Case 2: 1

■ 思路:

开始一共有 n 个“大块”，编号从左到右依次为0到 $n-1$

第 i 个大块的颜色是 $color[i]$, 包含的方块数目, 即长度, 是 $len[i]$

用 $click_box(i)$ 表示从大块0到大块 i 这一段消除后所能得到的最高分

整个问题就是求 $click_box(n-1)$

无法形成递推关系

- 将问题的描述形式细化（复杂化）：

用 $\text{click_box}(i,j)$ 表示从大块 i 到大块 j 这一段消除后所能得到的最高分

则整个问题就是： $\text{click_box}(0,n-1)$

■ 往前做一步：

要求click_box(i,j)时，考虑最右边的大块j，对它有两种处理方式，要取其优者：

- 1) 直接消除它，此时能得到最高分就是：
 $\text{click_box}(i, j-1) + \text{len}[j] * \text{len}[j]$
- 2) 期待以后它能和左边的某个同色大块合并

■ 考虑和左边的某个同色大块合并:

左边的同色大块可能有很多个，到底和哪个合并最好，不知道，只能枚举。假设大块j和左边的大块k($i \leq k < j-1$)合并，此时能得到的最高分是多少呢？

是不是：

$$\text{click_box}(i, k-1) + \text{click_box}(k+1, j-1) + (\text{len}[k] + \text{len}[j])^2$$

■ 不对!

$$\text{click_box}(i, k-1) + \text{click_box}(k+1, j-1) + (\text{len}[k] + \text{len}[j])^2$$

将大块 k 和大块 j 合并后，形成的新大块会在最右边。将该新大块直接将其消去的做法，才符合上述式子，但直接将其消去，未必是最好的，也许它还应该和左边的同色大块合并，才更好

递推关系无法形成，怎么办？

- 将问题描述进一步细化(复杂化)

考虑新的形式:

`click_box(i,j,ex_len)`

表示:

大块j的右边已经有一个长度为ex_len的大块(该大块可能是在合并过程中形成的,不妨就称其为ex_len),且j的颜色和ex_len相同,在此情况下将i到j以及ex_len都消除所能得到的最高分。

于是整个问题就是求: `click_box(0,n-1,0)`

■ 可以形成递推关系

求 $\text{click_box}(i,j,\text{ex_len})$ 时，有两种处理方法，取最优者
假设 j 和 ex_len 合并后的大块称作 Q

1) 将 Q 直接消除，这种做法能得到的最高分就是：

$$\text{click_box}(i,j-1,0) + (\text{len}[j] + \text{ex_len})^2$$

2) 期待 Q 以后能和左边的某个同色大块合并。需要枚举可能和 Q 合并的大块。

假设让大块 k 和 Q 合并，则此时能得到的最大分数是：

$$\text{click_box}(i,k,\text{len}[j] + \text{ex_len}) + \text{click_box}(k+1,j-1,0)$$

递归的终止条件是什么？ $i == j$

```

#include<cstring>
#include<iostream>
using namespace std;
struct Block {
    int color;
    int len;
};
struct Block segment[200];
int score[200][200][200]; //存放计算结果，避免重复计算
int click_box(int start, int end, int extra_len) {
    int i, result, temp;
    if ( score[start][end][extra_len]>0 )
        return score[start][end][extra_len];
    result = segment[end].len + extra_len;
    result = result*result; //end和extra_len一起消去的得分
    if (start==end) {
        score[start][end][extra_len]= result;
        return score[start][end][extra_len];
    }
}

```

```
result += click_box(start, end-1, 0);
i = end - 1;
for ( i = end - 1; i >= start; i-- ) {
    if (segment[i].color!=segment[end].color)
        continue;
    temp = click_box(start,i,segment[end].len + extra_len) +
           click_box(i+1, end-1, 0);
    if ( temp<=result ) continue;
    result = temp;
}
score[start][end][extra_len] = result;
return score[start][end][extra_len];
}
```

```

int main(){
    int t, n, i, j, end, color;
    cin >> t;
    for (i=0;i<t;i++) {
        cin >> n;  end = 0; //大块总数
        cin >> segment[end].color;
        segment[end].len = 1;
        for (j=1;j<n;j++) {
            cin >> color;
            if ( color==segment[end].color )    segment[end].len++;
            else {
                end++;
                segment[end].color = color;  segment[end].len = 1;
            }
        }
        memset(score,0,sizeof(score));
        cout << "Case " << i+1 << ": " << click_box(0,end,0) << endl;
    }
    return 0;
}

```