

# 递归 — 棋盘分割

郭 炜 刘家瑛

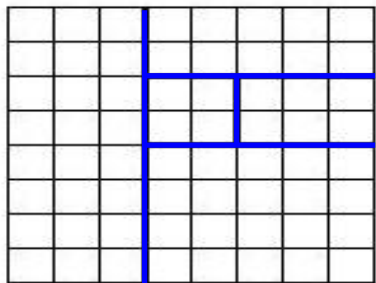


北京大學

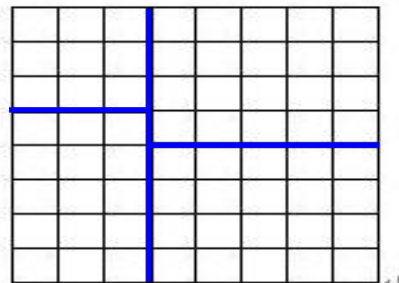


# 棋盘分割

- 将一个 $8*8$ 的棋盘进行如下分割:
  - 将原棋盘割下一块矩形棋盘并使剩下部分也是矩形,
  - 再将剩下的部分继续如此分割, 这样割了 $(n-1)$ 次后,
  - 连同最后剩下的矩形棋盘共有 $n$ 块矩形棋盘.
- (每次切割都只能沿着棋盘格子的边进行)



允许的分割方案



不允许的分割方案



- 原棋盘上每一格有一个分值，  
一块矩形棋盘的总分为其所含各格分值之和
- 现在需要把棋盘按上述规则分割成  $n$  块矩形棋盘，  
并使各矩形棋盘总分的均方差最小

均方差  $\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}}$ ，其中平均值  $\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$ ，

$x_i$  为第  $i$  块矩形棋盘的总分

请编程对给出的棋盘及  $n$ ，求出  $\sigma$  的最小值



## 输入

第1行为一个整数 $n$  ( $1 < n < 15$ )

第2行至第9行每行为8个小于100的非负整数, 表示棋盘上相应格子的分值

每行相邻两数之间用一个空格分隔

## 输出

仅一个数, 为 $\sigma$  (四舍五入精确到小数点后三位)



## 样例输入

3

1	1	1	1	1	1	1	3
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	0
1	1	1	1	1	1	0	3

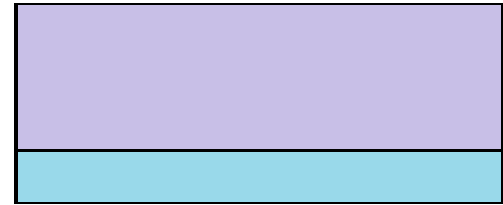
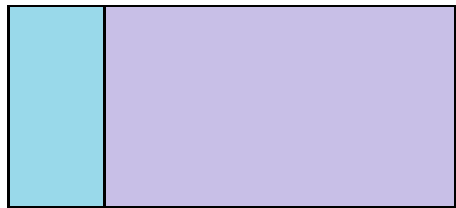
## 样例输出

1.633



# 问题分析 (1)

- 每一次分割有以下4种方法:



$$f(k, \text{棋盘}) = \{ f(1, \text{割下的棋盘}) + f(k-1, \text{待割的棋盘}) \} \quad (k \geq 2)$$



## 问题分析 (2)

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n}}$$

如右式, 若要求出最小方差, 只需要求出最小的  $\sum x_i^2$

$$\begin{aligned} & \sum (x_i - \bar{x})^2 \\ &= \sum (x_i^2 - 2x_i\bar{x} + \bar{x}^2) \\ &= \sum x_i^2 - \sum 2x_i\bar{x} + n\bar{x}^2 \\ &= \sum x_i^2 - 2n\bar{x}^2 + n\bar{x}^2 \\ &= \sum x_i^2 - n\bar{x}^2 \end{aligned}$$



# 问题分析 (3)

- 设 $\text{fun}(n, x1, y1, x2, y2)$ 为以 $(x1, y1)$ 为左上角,  $(x2, y2)$ 为右下角的棋盘分割成 $n$ 份后的最小平方和
- 那么 $\text{fun}(n, x1, y1, x2, y2) =$

$$\begin{aligned} & \min \left\{ \min_{i=x1}^{x2-1} \{ \text{fun}(n-1, x1, y1, i, y2) + \text{fun}(1, i+1, y1, x2, y2) \}, \right. \\ & \min_{i=x1}^{x2-1} \{ \text{fun}(1, x1, y1, i, y2) + \text{fun}(n-1, i+1, y1, x2, y2) \}, \\ & \min_{i=y1}^{y2-1} \{ \text{fun}(n-1, x1, y1, x2, i) + \text{fun}(1, x1, i+1, x2, y2) \}, \\ & \left. \min_{i=y1}^{y2-1} \{ \text{fun}(1, x1, y1, x2, i) + \text{fun}(n-1, x1, i+1, x2, y2) \} \right\} \end{aligned}$$

其中 $\text{fun}(1, x1, y1, x2, y2)$ 等于该棋盘内分数和的平方





# 问题分析 (3)

- 只想到这个还不够, TLE!
- 对于某个 $\text{fun}(n, x1, y1, x2, y2)$ 来说, 可能使用多次这个值, 所以每次都计算太消耗时间
- 解决办法: **记录表**
  - 用 $\text{res}[n][x1][y1][x2][y2]$ 来记录 $\text{fun}(n, x1, y1, x2, y2)$
  - res初始值统一为-1
  - 当需要使用 $\text{fun}(n, x1, y1, x2, y2)$ 时, 查看 $\text{res}[n][x1][y1][x2][y2]$ 
    - 如果为-1, 那么计算 $\text{fun}(n, x1, y1, x2, y2)$ , 并保存于 $\text{res}[n][x1][y1][x2][y2]$
    - 如果不为-1, 直接返回 $\text{res}[n][x1][y1][x2][y2]$



# 参考程序

```
int s[9][9];    //每个格子的分数
int sum[9][9];  //(1,1)到(i,j)的矩形的分数之和
int res[15][9][9][9][9]; //fun的记录表

int calSum(int x1,int y1,int x2,int y2)//(x1,y1)到(x2,y2)的矩形的分数之和
{
    return sum[x2][y2]-sum[x2][y1-1]-sum[x1-1][y2]+sum[x1-1][y1-1];
}
```



```
int fun(int n,int x1,int y1,int x2,int y2)
{
    int t, a, b, c, e, MIN=100000000;
    if(res[n][x1][y1][x2][y2] != -1)
        return res[n][x1][y1][x2][y2];
    if(n==1) {
        t=calSum(x1,y1,x2,y2);
        res[n][x1][y1][x2][y2]=t*t;
        return t*t;
    }
}
```



```
for(a=x1;a<x2;a++) {  
    c=calSum(a+1,y1,x2,y2);  
    e=calSum(x1,y1,a,y2);  
    t=min(fun(n-1,x1,y1,a,y2)+c*c, fun(n-1,a+1,y1,x2,y2)+e*e);  
    if(MIN>t) MIN=t;  
}  
for(b=y1;b<y2;b++) {  
    c=calSum(x1,b+1,x2,y2);  
    e=calSum(x1,y1,x2,b);  
    t=min(fun(n-1,x1,y1,x2,b)+c*c, fun(n-1,x1,b+1,x2,y2)+e*e);  
    if(MIN>t) MIN=t;  
}  
res[n][x1][y1][x2][y2]=MIN;  
return MIN;  
}
```



```
int main() {  
    memset(sum, 0, sizeof(sum));  
    memset(res, -1, sizeof(res)); //初始化记录表  
    int n;  
    cin>>n;  
    for (int i=1; i<9; i++)  
        for (int j=1, rowsum=0; j<9; j++) {  
            cin>>s[i][j];  
            rowsum +=s[i][j];  
            sum[i][j] += sum[i-1][j] + rowsum;  
        }  
    double result = n*fun(n,1,1,8,8)-sum[8][8]*sum[8][8];  
    cout<<setiosflags(ios::fixed)<<setprecision(3)<<sqrt(result/(n*n))<<endl;  
    return 0;  
}
```