1652218 段巍

# 3-2 编辑距离问题

## 一、核心算法：

$$d_{i0} = \sum_{k=1}^{i} w_{\text{del}}(b_k), \qquad \text{for } 1 \le i \le m$$

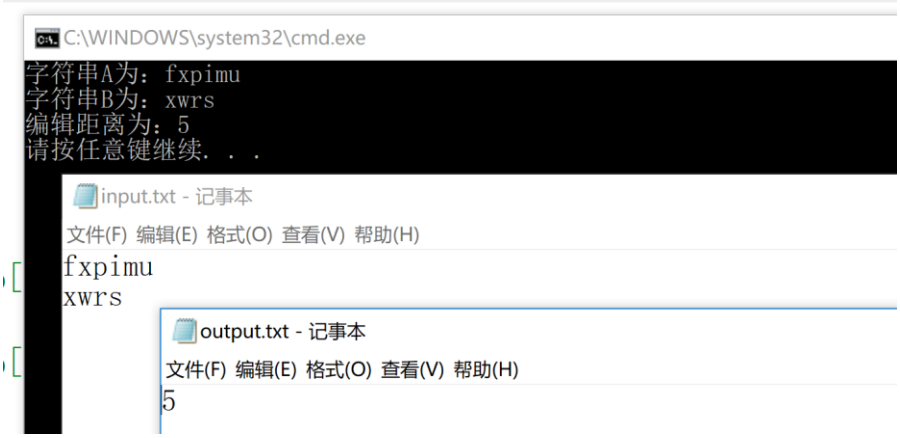
$$d_{0j} = \sum_{k=1}^{j} w_{\text{ins}}(a_k), \qquad \text{for } 1 \le j \le n$$

$$d_{ij} = \begin{cases} d_{i-1,j-1} & \text{for } a_j = b_i \\ \min \begin{cases} d_{i-1,j} + w_{\text{del}}(b_i) \\ d_{i,j-1} + w_{\text{ins}}(a_j) \\ d_{i-1,j-1} + w_{\text{sub}}(a_j, b_i) \end{cases} & \text{for } a_j \ne b_i \end{cases} \qquad \text{for } 1 \le i \le m, 1 \le j \le n.$$

## 二、代码实现：

```
//m:字符串A长度
//n:字符串B长度
int Edit_Dist(string A,string B)
{
    int m = A.length();
    int n = B.length();
    for (int i = 0; i <= m; i++) //p[i][0] = i delete;
        p[i][0] = i;
    for (int j = 0; j <= n; j++) //p[0][j] = j insert;
        p[0][j] = j;
    for(int i = 1;i <= m; i++)
    {
        ca = A[i - 1];
        for (int l = 1; l <= n; l++)
        {
            cb = B[l - 1];
            if (ca == cb)
                p[i][l] = p[i - 1][l - 1];
            else
                p[i][l] = mini(p[i-1][l-1]+1,p[i-1][l]+1,p[i][l-1]+1);
        }//end for
    }//end for
    return p[m][n];
}
```

三、结果演示




## 3-3 石子合并问题

一、核心算法：

$$dp[i][j]=\begin{cases}0 & i=j\\ min(dp[i][k],dp[k+1][j])+sum[i][j] & i\neq j\end{cases}$$

题目要求圆排列，可以将存放石子数的数字 P[]向左轮换，模拟圆排列

二、代码实现：

1、max

```
//当一个单独合并时，m[i][i]设为0，表示没有石子
    for (i = 1; i <= n; i++)
        m[i][i] = 0;
    //相邻的两堆石子合并
    for (i = 1; i < n; i++)
        m[i][i + 1] = p[i] + p[i + 1];
    //n>=3
    for (int r = 3; r <= n; r++)
        for (i = 1; i <= n - r + 1; i++)
        {
            j = i + r - 1; //m[i][j]
            int sum = 0;
            for (int b = i; b <= j; b++)//sum=p[i:j]
                sum += p[b];

            m[i][j] = m[i + 1][j] + sum; //m[i][i]+m[i+1][j] + sum
```

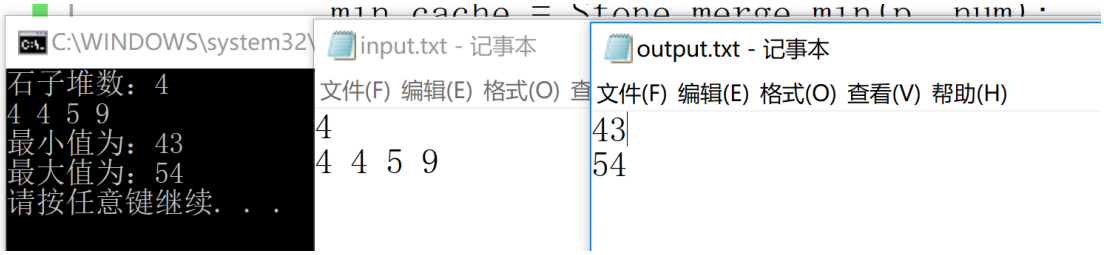
```
            for (k = i + 1; k <= r - 1; k++)
            {
                int t = m[i][k] + m[k + 1][j] + sum;
                if (t > m[i][j])m[i][j] = t;
            }//end for
        }//end for
    max = m[1][n];
```

**2、min 最小值与最大值情况类似**

**3、圆排列**

```
//向左轮换p[] 模拟圆排列，总共有n种情况
     for (int j = 1; j <= num - 1; j++)
        {
            int min_cache = 0;
            int max_cache = 0;
            int cache = p[1];
            for (int k = 2; k <= num; k++) //向左轮换p[] 模拟圆排列
            {
                p[k - 1] = p[k];
            }
            p[num] = cache;
            min_cache = Stone_merge_min(p, num);
            max_cache = Stone_merge_max(p, num);
            if (min_cache<min)
                min = min_cache;
            if (max_cache>max)
                max = max_cache;
        }
```

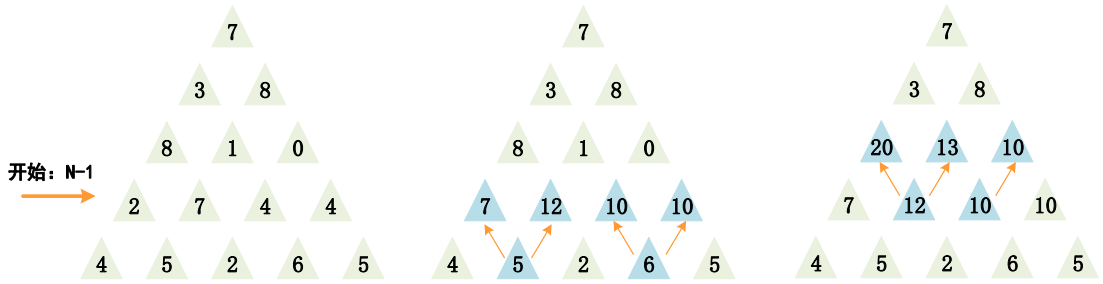
**三、结果演示**



## 3-4 数字三角形问题

### 一、核心算法：

自底向上




### 二、代码实现：

```
void Num_Tri(int **m,int n)
{
    for(int row=n-1;row>=0;row--)
        for (int col = 0; col <= row; col++)
        {
            if (m[row + 1][col] > m[row + 1][col + 1])
                m[row][col] += m[row + 1][col];
            else m[row][col] += m[row + 1][col + 1];
        }
}
```

### 三、结果演示