



动态规划

最小编辑距离



目

录

CONTENTS

01 选题背景

02 模型算法分析

03 代码与程序展示

04 总结

第一部分

问题引入

论文查重--最小编辑距离



实际问题引入

选题背景与意义



论文抄袭现象严重



论文查重



通过求解文本编辑距离判断是否抄袭

俄罗斯科学家Vladimir Levenshtein在1965年提出最小编辑距离这个概念
因此最小编辑距离也叫Levenshtein Distance

问题抽象与模型建立

数学模型--最小编辑距离：

论文查重-->比较两段文本的相似程度-->由一段文字转化为另一段文字的操作步数

操作步数-->编辑距离

编辑距离-->抄袭者改动程度

操作步数越少，编辑距离越小，相似度越大



实际问题抽象

查重问题抽象为：

给定两个字符串A和B，求字符串A至少经过多少步操作才能转换成字符串B

操作包括：

- (1) 删除一个字符**
- (2) 插入一个字符**
- (3) 将一个字符改为另一个字符**

通过最小编辑距离判断两个字符串的相似程度进而判断抄袭嫌疑



第二部分

模型与算法分析

动态规划--递归



模型描述

模型分析:

$\text{dis}[i][j]$: 字符串A的前i个字符到字符串B的前j个字符的编辑距离

1.当字符串A和B都为空时

$$\text{dis}[i][j]=0$$

2.当字符串A为空，字符串B不为空时

循环将B的第i个字符插入A中

$$\text{dis}[0][j]=j$$

3.当字符串B为空，字符串A不为空时

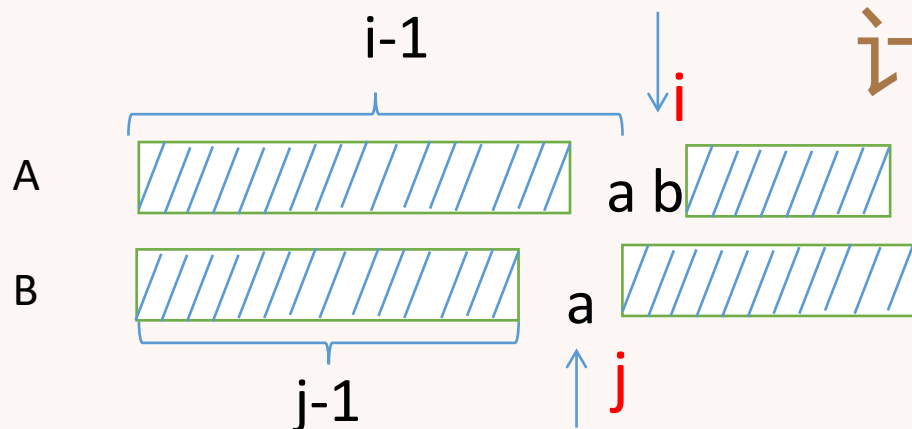
循环删去A中的字符


$$\text{dis}[i][0]=i$$



计算模型

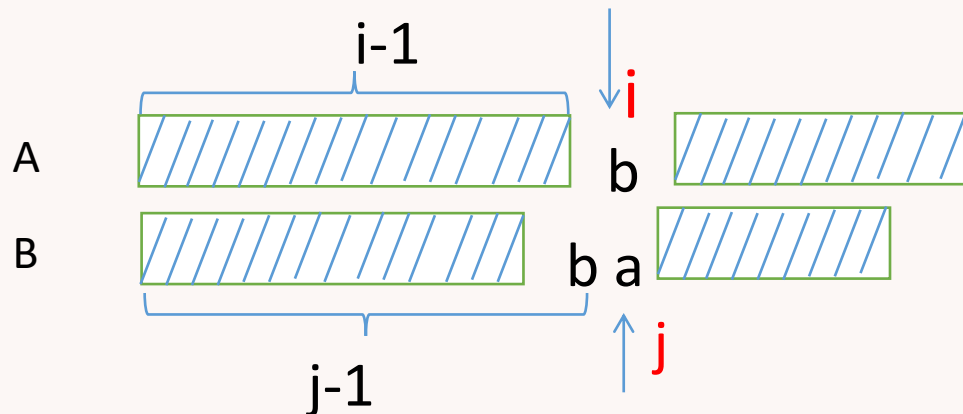
删除



A中 i 个和B中 $j-1$ 个已比较结束, 得出A中 i 个字符到B中 $j-1$ 个字符的编辑距离

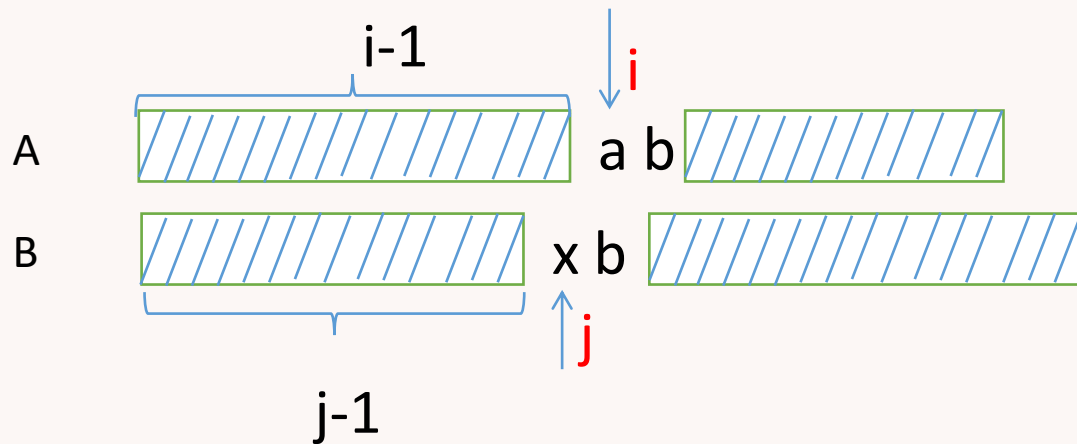
$$\text{dis}[i][j] = \text{dis}[i-1][j] + 1$$

插入



$$\text{dis}[i][j] = \text{dis}[i][j-1] + 1$$

替换



$$\text{dis}[i][j] = \text{dis}[i-1][j-1] + \text{flag}$$

$$A[i-1] = B[j-1] \quad \text{flag} = 0$$

$$A[i-1] \neq B[j-1] \quad \text{flag} = 1$$

计算模型

$$\text{Dis}[i][j] = \begin{cases} 0 & i=0, j=0 \\ i & i>0, j=0 \\ j & i=0, j>0 \\ \text{Min}\{\text{dis}[i][j]=\text{dis}[i-1][j]+1, \text{dis}[i][j]=\text{dis}[i][j-1]+1, \text{dis}[i][j]=\text{dis}[i-1][j-1]+\text{flag}\} & i>0, j>0 \end{cases}$$
$$\text{Flag} = \begin{cases} 0 & A[i-1]=B[j-1] \\ 1 & A[i-1] \neq B[j-1] \end{cases}$$

— 算法策略选择

01 递归算法

02 动态规划算法

递归：插入、删除、替换三种操作每一种都进行递归
计算所有解法，比较后给出最小距离，浪费时间空间
动态规划：每次选择最优步骤

算法策略选择

多阶段逐步解决问题的策略——贪心算法、递推法、递归法和动态规划法

贪心算法：每一步都根据策略得到一个结果，并传递到下一步，自顶向下，一步一步地做出贪心决策。

动态规划算法：每一步决策得到的不是一个唯一结果，而是一组中间结果（且这些结果在以后各步可能得到多次引用），只是每一步都使问题的规模逐步缩小，最终得到问题的一个结果。

递推、递归法：注重每一步之间的关系，决策的因素较少。递推法是根据关系从前向后推导，从小规模问题的结论推解出大问题的解。而递归法是根据关系从后向前使大问题转化为小问题，最后同样由小规模问题的解推解出大问题的解。



动态规划算法的正确性证明

假设有两个数组， $A[], B[]$ 。 $A[i]$ 为 A 的第 i 个元素， $A(i)$ 为由 A 的第一个元素到第 i 个元素所组成的前缀。 $m(i, j)$ 为 $A(i)$ 和 $B(j)$ 的最长公共子序列长度。由于算法本身的递推性质，其实只要证明，对于某个 i 和 j ：

$$m(i, j) = m(i-1, j-1) + 1 \quad (\text{当 } A[i] = B[j] \text{ 时})$$

$$m(i, j) = \max(m(i-1, j), m(i, j-1)) \quad (\text{当 } A[i] \neq B[j] \text{ 时})$$

第一个式子很好证明，即当 $A[i] = B[j]$ 时。可以用反证，假设 $m(i, j) > m(i-1, j-1) + 1$ （ $m(i, j)$ 不可能小于 $m(i-1, j-1) + 1$ ，原因很明显），那么可以推出 $m(i-1, j-1)$ 不是最长的这一矛盾结果。

第二个需要注意。当 $A[i] \neq B[j]$ 时，还是反证，假设 $m(i, j) > \max(m(i-1, j), m(i, j-1))$ 。由反证假设，可得 $m(i, j) > m(i-1, j)$ 。这个可以推出 $A[i]$ 一定在 $m(i, j)$ 对应的LCS序列中（反证可得）。而由于 $A[i] \neq B[j]$ ，故 $B[j]$ 一定不在 $m(i, j)$ 对应的LCS序列中。所以可推出 $m(i, j) = m(i, j-1)$ 。这就推出了与反证假设矛盾的结果。得证。

算法

输入字符串s1, s2 输出最小编辑距离dis[len1][len2]

```
void MinDis()  
{
```

```
    len1←s1.size();  
    len2←s2.size();  
    for i←0 to len1 do  
        dis[i][0]←i;  
    for j←0 to len2 do  
        dis[0][j]←j;  
    for i←1 to len1 do  
    {
```

```
        for j←1 to len2 do  
        {
```

```
            //添加  
            If(s1[i - 1]
```

```
        else  
            add←1 + dis[i][j - 1];  
        //删除  
        del←1 + dis[i - 1][j];
```

```
        //替换  
        If(s1[i - 1] == s2[j - 1])  
            then r←dis[i - 1][j - 1];
```

$$T(m, n) = 8 + \sum_{i=0}^m 3 + \sum_{j=0}^n 3 + \sum_{i=1}^m (4 + \sum_{j=1}^n 8) - 1;$$

$$= 8 + 3(m + 1) + 3(n + 1) + m(4 + 8n)$$

$$= 8mn + 7m + 3n + 14$$

$$= O(mn)$$

算法

```
int MinEditDistance(int i, int j, string s1, string s2)
{
    if(i==0)
        return j;
    if(j==0)
        return i;
    if(s1[i-1]==s2[j-1])
        return MinEditDistance(i-1, j-1, s1, s2);
    else
    {
        int add=MinEditDis
        int del=MinEditDis
        int rep=MinEditDis
        return ((add<del?a
    }
```

$$T(m, n) = T(m-1, n) + T(m-1, n-1) + T(m, n-1) + O(1)$$

$$T(m, 0) = O(1)$$

$$T(0, n) = O(1)$$

指数形式

example

		a	l	m	o	s	t
a							
l							
s							
o							

example

		a	l	m	o	s	t
	0	1	2	3	4	5	6
a	1						
l	2						
s	3						
o	4						

example

example

j

i

		a	l	m	o	s	t
	0	1	2	3	4	5	6
a	1	0					
l	2						
s	3						
o	4						

a→a 0步操作



example

		a	l	m	o	s	t
	0	1	2	3	4	5	6
a	1	0	1				
l	2						
s	3						
o	4						

a→a 0步操作

a→l $\text{dis}[i][j] = \text{dis}[i][j-1] + 1$

example

j
↓

i
→

		a	l	m	o	s	t
	0	1	2	3	4	5	6
a	1	0	1	2			
l	2						
s	3						
o	4						

a→a 0步操作

a→l $\text{dis}[i][j] = \text{dis}[i][j-1] + 1$

a→m $\text{dis}[i][j] = \text{dis}[i][j-1] + 1$



example

		a	l	m	o	s	t
	0	1	2	3	4	5	6
i → a	1	0	1	2	3	4	5
l	2						
s	3						
o	4						

example

		<div>j ↓</div>					
		a	l	m	o	s	t
	0	1	2	3	4	5	6
a	1	0	1	2	3	4	5
l	2	1					
s	3						
o	4						

l -> a $\text{dis}[i][j] = \text{dis}[i-1][j] + 1$

l -> l $\text{dis}[i][j] = \text{dis}[i-1][j-1]$



example

		a	l	m	o	s	t
	0	1	2	3	4	5	6
a	1	0	1	2	3	4	5
l	2	1	0				
s	3						
o	4						

l -> a

l -> l $\text{dis}[i][j] = \text{dis}[i-1][j-1]$



example

		a	l	m	o	s	t
	0	1	2	3	4	5	6
a	1	0	1	2	3	4	5
l	2	1	0	1	2	3	4
s	3	2	1	1	2	2	3
o	4	3	2	2	1	2	3

j ↓

i →

第三部分

代码与程序展示

代码演示



代码及测试数据

```
#include <iostream>
#include <string>
using namespace std;

int Min(int a,int b,int c)
{
    /**返回a b c的最小值**/
    int temp=a<b?a:b;
    return temp<c?temp:c;
}

int MinEditDistance(string s1,string s2)
{
    int add,del,r;//添加,删除,替换 操作数

    int len1=s1.size();//字符串s1长度
    int len2=s2.size();

    int dis[len1][len2];

    for(int i=0;i<len1;i++){
        for(int j=0;j<len2;j++){

            if(i==0 && j==0)
                dis[i][j]=0;
            else if(i==0)
                dis[i][j]=j;
            else if(j==0)
                dis[i][j]=i;
            else{

                //添加
                if(s1[i]==s2[j]){
```

```
                //添加
                if(s1[i]==s2[j]){
                    //若s1[i]与s2[j]相同 则不用添加
                    add=dis[i-1][j-1];
                }
                else{
                    // s1[i]与s2[j]不相同 则在s1[i+1]处添加s2[j]
                    add=1+dis[i][j-1];
                }

                //删除
                del=dis[i][j]=1+dis[i-1][j];

                //替换(replace)
                if(s1[i]==s2[j]){
                    //若s1[i]与s2[j]相同 则不用替换
                    r=dis[i-1][j-1];
                }
                else{
                    // s1[i]与s2[j]不相同 则将s1[i]位置处替换成s2[j]
                    r=1+dis[i-1][j-1];
                }

                dis[i][j]=Min(add,del,r);
                /*最后字符串s1[0..i]与s2[0..j]的最小操作数
                为添加(add),删除(del)和替换(r)的最小值*/
            }
        }
    }
    return dis[len1-1][len2-1].
}

int main()
{
```

```
    string s1,s2;
    cout<<"input two strings"<<endl;
    cout<<"string 1:"<<cin>>s1;
    cout<<"string 2:"<<cin>>s2;
    cout<<"-----"<<endl;
    cout<<"the Minimal Edit Distance is:"<<MinEditDistance(s1,s2)<<endl;
    cout<<"-----"<<endl;
    return 0;
}
```

代码及测试数据

```
int Distance()
{
    int i, j;
    int length1 = strlen(A);
    int length2 = strlen(B);
    int flag, del, ins, change;
    int **dis = new int *[length1 + 1];
    for (i = 0; i <= length1; i++)
        dis[i] = new int[length2 + 1];
    for (i = 0; i <= length1; i++)
        dis[i][0] = i;
    for (j = 1; j <= length2; j++)
        dis[0][j] = j;
    for (i = 1; i <= length1; i++)
    {
        for (j = 1; j <= length2; j++)
        {
            flag = A[i-1] == B[j-1] ? 0 : 1;
            del = dis[i-1][j] + 1; // 删除后的距离
            ins = dis[i][j-1] + 1; // 插入后的距离
            change = dis[i-1][j-1] + flag; // 替换后的距离
            dis[i][j] = Min(del, ins, change);
        }
    }
    return dis[length1][length2];
}
```



程序及测试数据

```
input two strings
string 1:aaaaaaaaaa
string 2:bbbbbbbbbb
-----
the Minimal Edit Distance is:10
-----
耗时: 5601

Process returned 0 (0x0)    execution time : 12.370 s
Press any key to continue.
```

```
input two strings
string 1:aaaaaaaaaa
string 2:bbbbbbbbbb
-----
the Minimal Edit Distance is:10
-----
耗时: 1
```



代码及测试数据

```
7 6 6 7 6 7 6 7 8 7 7 7 7 5 5 7 7 6 5 7 7 6 6 7 7 6 7 7 5 7 6 7 7 7 6 7 6 6 7 7 5 7 7 4 6 6 8 6 6 5 6 6 8 5 4 8 7 7 7 8 5 5 7 5 6 5 6 7 7 5 7 7 6 7 7 6 6 5
6 6 5 6 6 4 7 7 7 7 7 6 7 5 7 7 5 7 7 6 6 5 5 6 7 6 6 6 6 6 7 6 6 6 6 6 7 7 5 6 6 6 6 4 6 6 5 7 7 8
6 6 6 8 8 6 8 6 6 5 6 6 6 7 5 5 7 6 7 7 6 6 7 7 6 8 7 6 6 7 8 7 5 5 7 7 6 7 6 6 7 5 5 8 7 6 6
6 7 5 6 6 7 6 6 8 6 6 7 4 7 6 6 6 7 5 5 6 7 6 5 7 6 5 7 6 5 7 6 6 5 7 6 7 6 7 6 6 8 7 6 6 8 7 6 6 5 5 4 6 6 7 7 7 6 7 7 6 6 7 6 7 7 4 7 7 5 6
6 6 6 4 7 6 7 8 8 7 6 7 5 6 6 6 7 8 6 7 7 7 5 6 5 7 6 7 6 7 6 7 4 7 7 6 5 8 6 8 7 6 5 6 7 6 7 6 6 8 7 6 6 8 7 6 6 5 5 4 6 6 7 7 7 6 7 7 6 6 7 6 7 8 6 7 8 6 7
7 7 6 5 6 6 7 7 7 7 7 6 8 4 6 7 7 5 5 8 7 5 5 5 4 6 6 6 6 6 6 5 6 6 6 5 7 7 7 5 7 5 7 7 4 7 6 6 8 5 7 7 7 5 8 6 6 7 7 7 5 6 6 4 6 6 7 7 7 6 6 5 7 5 7 4 7 6 6
6 7 6 6 7 6 6 6 6 6 6 6 5 6 7 7 6 6 7 8 5 6 7 8 5 6 6 7 7 6 6 7 7 7 6 6 7 7 7 8 6 6 6 7 6 7 7 8 7 6 5 7 5 5 3 5 6 6 7 7 6 7 7 5 7 6 6 7 5 6 8 5 5 7 6 7 7 6 6
7 7 6 6 6 6 5 7 7 7 6 6 4 4 7 7 7 6 7 5 8 7 7 6 7 6 6 6 7 7 5 7 6 6 7 7 6 7 6 8 6 6 8 7 6 5 4 5 7 5 7 6 6 5 7 8 7 7 6 5 6 6 7 8 7 6 6 6 5 5 6 7 5 7 5 7 6
7 7 7 6 7 7 7 7 6 7 6 7 6 6 6 6 6 6 6 6 6 6 7 5 5 7 8 6 6 6 7 7 5 6 7 6 6 6 6 6 5 7 7 5 6 8 6 8 6 5 7 6 7 6 7 6 7 7 5 6 6 6 5 8 6 7 8 6 6 6 7 7 6 6 7 7 5 6
7 5 5 7 5 5 5 7 6 5 7 7 7 6 7 7 6 6 6 6 6 6 6 6 5 6 7 6 7 5 7 6 5 7 6 6 4 5 7 7 5 5 6 6 6 7 6 7 5 5 7 6 7 6 8 7 6 7 5 8 8 6 7 7 7 6 6 6 6 6 8 7 7 7 6 6 6 7 7
7 5 7 7 7 6 5 6 7 8 7 6 7 7 6 7 6 6 7 6 6 7 7 4 7 8 8 8 7 7 7 7 6 6 7 7 7 5 6 8 5 7 5 8 7 7 8 7 6 7 6 5 7 5 6 7 7 7 5 5 6 5 6 6 6 6 5 6 7 7 6 7 7 7 6 7 7 7
7 6 6 5 7 8 6 5 7 6 7 8 8 8 5 6 6 7 7 5 7 6 6 7 6 7 6 7 6 7 6 6 6 6 5 7 6 7 6 4 6 7 5 7 5 7 5 5 7 8 7 7 7 7 6 7 7 6 7 6 4 6 6 5 6 7 6 6 5 4 6 6 7 6 7 7 7 6 6 4
7 6 7 8 4 5 7 6 7 3 7 3 7 6 6 7 6 6 7 6 7 6 7 8 7 7 5 7 7 6 7 8 7 6 7 6 6 5 6 6 6 6 7 5 6 7 7 6 6 7 8 7 6 7 6 7 6 6 5 6 6 6 6 7 5 7 6 6 6 7 7 6 7 8 7 7 7 6
6 5 7 6 7 8 7 7 7 7 6 6 8 6 6 8 6 6 7 8 7 7 7 6 6 7 6 7 7 6 4 5 5 7 6 7 6 6 7 7 8 7 4 7 5 7 7 3 7 6 7 6 6 6 7 5 7 6 5 7 6 7 6 7 6 7 6 7 5 7 6 5 6 6 6 6 6 7 5 7 6
6 6 6 8 5 7 4 6 5 7 6 7 5 7 7 6 6 5 6 6 7 7 6 5 7 7 6 6 5 7 7 6 6 5 7 7 6 7 5 6 6 5 6 5 6 5 7 6 7 7 6 7 7 6 7 6 7 7 7 7 7 5 7 6 7 8 6 7 6 6
5 7 7 6 5 6 6 7 5 6 6 6 5 5 7 6 7 5 7 6 7 7 7 7 5 5 7 7 5 7 6 7 7 7 6 6 5 7 7 7 7 6 7 7 7 5 7 7 7 6 8 8 6 5 6 8 6 6 7 5 6 8 6 7 6 6 7 5 8 6 6 7 3 7 7 6 5 7
8 7 6 6 6 6 6 7 6 6 7 7 6 6 8 6 8 6 8 6 8 6 7 6 7 5 6 7 7 5 4 7 6 7 6 6 7 8 5 7 7 6 6 7 8 6 6 6 6 6 7 5 8 7 7 6 7 6 7 5 7 6 5 6 6 8 7 6 7 8 6 5 6 5 7 6
5 7 4 6 6 7 5 7 6 7 7 6 7 5 7 8 7 7 6 7 6 5 7 7 6 6 6 5 7 6 6 5 7 6 8 5 6 5 4 6 7 7 5 4 8 8 5 7 7 5 6 6 6 8 6 6 6 6 6 7 8 7 6 7 8 6 6 6 6 8 6 6 7 6 7 6 7 5 8 7 7 6 6
7 5 4 7 7 6 6 7 4 6 5 6 8 7 6 6 6 8 6 6 7 7 7 5 5 7 8 7 6 6 6 7 5 7 6 6 7 7 6 6 6 6 5 6 6 7 8 5 8 5 8 7 6 6 7 4 7 6 8 7 6 6 8 7 7 6 7 6 5 6 6 7 7 6 7 6 6 6 8 7 6
7 7 6 7 7 6 7 6 7 7 5 7 6 5 5 5 5 6 5 6 7 6 8 6 6 6 6 7 6 6 6 5 8 7 6 6 7 6 7 7 4 7 6 6 6 5 7 6 7 5 4 7 5 6 7 5 6 7 7 6 6 6 6 5 7 7 7 5 7 6 6 6 7 6 6 6 8 7 6 6 6
5 7 7 5 7 6 7 5 7 5 7 6 6 5 7 4 5 6 6 7 8 6 4 6 7 7 6 7 6 7 8 8 7 7 6 7 6 7 6 7 5 7 7 6 7 6 7 5 6 7 5 6 5 7 5 6 7 7 6 6 6 6 5 7 7 7 5 7 6 6 6 7 6 6 6 8 7 6 6 6
7 7 7 5 6 8 6 8 6 6 7 4 5 6 7 4 7 7 4 7 6 7 6 6 6 6 6 6 7 7 6 7 7 7 5 5 6 5 7 6 6 7 7 7 6 7 7 6 6 8 7 5 6 6 7 6 6 8 7 6 8 6 7 5 7 7 5 6 7 7 6
6 6 7 5 4 6 7 7 6 8 5 6 6 7 5 7 7 6 5 7 6 6 5 7 7 6 7 6 5 8 6 6 7 7 5 8 5 6 6 6 6 7 6 6 6 6 7 5 5 7 5 6 4 6 5 7 7 7 7 6 7 7 8 7 8 6 7 8 8 6 6 5 6 5 7 5 8 7 4 7
8 5 6 7 6 7 6 6 6 7 5 6 7 6 6 6 7 5 6 7 6 6 6 6 6 5 7 7 5 4 7 7 7 6 7 7 5 5 7 8 5 6 8 6 7 8 7 7 6 6 6 7 7 8 7 7 6 6 7 6 5 7 6 6 7 7 8 6 5 5 6 8 5 6 7 7 7 8 6 8
6 5 7 6 7 6 6 6 6 7 5 8 5 7 6 4 7 7 8 6 7 6 7 6 6 6 5 5 7 4 6 7 6 5 6 5 7 7 6 7 7 6 5 6 7 5 7 5 7 7 6 7 7 7 6 6 6 5 7 7 6 6 7 6 6 5 6 6 5 7 6 7 7 5 7 8 6
6 6 8 7 8 7 6 7 6 7 7 6 5 6 7 5 7 7 8 5 7 6 6 6 7 5 6 7 8 7 6 6 3 7 7 4 6 6 6 5 6 6 5 4 6 8 7 6 7 7 7 8 5 6 6 7 8 7 7 7 6 7 6 7 7 6 5 7 7 6 6 7 6 8 7 6 8 7
6 6 6 5 7 6 6 7 7 7 6 7 8 7 7 6 6 7 6 6 7 6 6 7 6 5 7 4 6 6 6 6 6 7 7 8 7 5 7 7 6 7 4 7 7 7 7 6 7 7 6 6 6 6 0
```

耗时: 7287

组数: 62298

第四部分

总结与反思

收获：

学习建模过程

掌握最小编辑距离的动态规划算法

不足：

模型单一

与投入应用有一定差距

