|  |  |
| --- | --- |
| **自然语言处理及应用**  **实验报告** | |
|  | |
|  |  |
| **名称** | 最小编辑距离 |
| **姓名** | 卢佳源 |
| **班级** | 人工智能91 |
| **学号** | 2191121196 |
| Email | bjlujiayuan@126.com |
| **日期** | 2021-9-19 |

# 实验目的

最小编辑距离是指两个字符串之间进行插入、删除和替换的编辑操作的最小次数。本实验要求编写程序来求出输入的两个字符串的最小编辑距离，同时需要记录两个字符串转化的过程。

# 实验环境

OS：Linux操作系统的ubuntu20.04版本

IDE：Code::Blocks IDE20.03

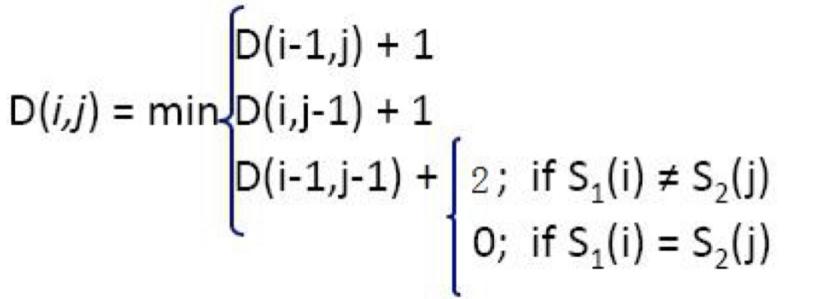
Terminal：gnome-terminal -t $TITLE -x

Language：C++

PATH：/home/lujiayuan/c++/mycpp.cpp

# 实验方法

1. 建立数学模型：（动态规划模型）



1. 实验思路 ：
2. 输入将要转换的两个字符串，假设a要转化成b；初始化字符串数组和最小编辑距离二维数组，用flag矩阵表示从i-1步到i步所做的操作，假设flag==100为插入，flag==20为删除，flag==3为替换；
3. 将mindis矩阵的第一行和第一列初始化为0到a和b的字符串长度（al和bl）的自然数列，作为基本条件；
4. 动态规划过程：遍历每一行和每一列，找到mindis(i-1,j)+1（插入，代价为1），mindis(i,j-1)+1（删除，代价为1），mindis(i-1,j-1)+2（替换，代价为2）的最小值作为mindis(i,j)，最终mindis(al,bl)为求得的两个字符串转换的最小编辑距离；
5. 回退过程：在（3）计算的同时，得到一个flag矩阵，记录了每一步可能进行的操作。因此根据flag(i,j)的数值从mindis(al,bl)开始回退，找到到每个点可能的路径中最小的mindis数值作为回退路径。
6. 代码：

#include<iostream>

#include<cstring>

using namespace std;

int min(int a,int b,int c)

{

if(a>=b)

{

if(c>=b) return b;

else return c;

}

else

{

if(c<=a) return c;

else return a;

}

}

int main()

{

char a[1001],b[1001];

a[0]='#';

b[0]='#';

int mindis[1001][1001];

int flag[1001][1001];

int al=0,bl=0;

int i=0,j=0;

cout<<"Please input the source and target:"<<endl;

while(cin>>a>>b)

{

al=strlen(a+1);

bl=strlen(b+1);

for(i=0;i<=al+1;i++)

{

for(j=0;j<=bl;j++)

{

mindis[i][j]=10000000;

}

}

for(i=0;i<=al+1;i++)

{

for(j=0;j<=bl;j++)

{

flag[i][j]=0;

}

}

for(i=0;i<=al+1;i++)

{

mindis[i][0]=i;

}

for(j=0;j<=bl+1;j++)

{

mindis[0][j]=j;

}

for(i=1;i<=al+1;i++)

{

for(j=1;j<=bl+1;j++)

{

if(a[i-1]==b[j-1])

{

mindis[i][j]=min(mindis[i-1][j]+1,mindis[i][j-1]+1,mindis[i-1][j-1]);

if(mindis[i][j]==mindis[i-1][j]+1) flag[i][j]=100;

if(mindis[i][j]==mindis[i][j-1]+1) flag[i][j]+=20;

if(mindis[i][j]==mindis[i-1][j-1]) flag[i][j]+=3;

}

else

{

mindis[i][j]=min(mindis[i-1][j]+1,mindis[i][j-1]+1,mindis[i-1][j-1]+2);

if(mindis[i][j]==mindis[i-1][j]+1) flag[i][j]=100;

if(mindis[i][j]==mindis[i][j-1]+1) flag[i][j]+=20;

if(mindis[i][j]==mindis[i-1][j-1]+2) flag[i][j]=3;

}

}

}

for (i=0;i<=al+1;i++)

{

for(j=0;j<=bl+1;j++)

{

cout<<mindis[i][j]<<" ";

}

cout<<endl;

}

/\* for (i=0;i<=al+1;i++)

{

for(j=0;j<=bl+1;j++)

{

cout<<flag[i][j]<<" ";

}

cout<<endl;

}\*/

cout<<mindis[al+1][bl+1]<<endl;

// cout<<a<<endl;

// cout<<b<<endl;

int x=0,y=0;

i=al+1;

j=bl+1;

cout<<"The edition path is:"<<endl;

cout<<mindis[al+1][bl+1]<<" ";

while(i>=1&&j>=1)

{

if(flag[i][j]==3)

{

if(mindis[i][j]-mindis[i-1][j-1]==2)

cout<<"substitute: "<<a[i-1]<<" to "<<b[i-1]<<endl;

i--;j--;

}

else if(flag[i][j]==100)

{

if(mindis[i][j]-mindis[i-1][j]==1)

cout<<"delete:"<<a[i-1]<<endl;

i--;

}

else if(flag[i][j]==20)

{

if(mindis[i][j]-mindis[i][j-1]==1)

cout<<"insert:"<<b[j-1]<<endl;

j--;

}

else if(flag[i][j]==23)

{

if(mindis[i][j-1]<=mindis[i-1][j-1])

{

if(mindis[i][j]-mindis[i][j-1]==1)

cout<<"insert:"<<b[j-1]<<endl;

j--;

}

else

{

if(mindis[i][j]-mindis[i-1][j-1]==2)

cout<<"substitute: "<<a[i-1]<<" to "<<b[i-1]<<endl;

i--;j--;

}

}

else if(flag[i][j]==103)

{

if(mindis[i-1][j]<=mindis[i-1][j-1])

{

if(mindis[i][j]-mindis[i-1][j]==1)

cout<<"delete:"<<a[i-1]<<endl;

i--;

}

else

{

if(mindis[i][j]-mindis[i-1][j-1]==2)

cout<<"substitute: "<<a[i-1]<<" to "<<b[i-1]<<endl;

i--;j--;

}

}

else if(flag[i][j]==120)

{

if(mindis[i-1][j]<=mindis[i][j-1])

{

if(mindis[i][j]-mindis[i-1][j]==1)

cout<<"delete:"<<a[i-1]<<endl;

i--;

}

else

{

if(mindis[i][j]-mindis[i][j-1]==1)

cout<<"insert:"<<b[j-1]<<endl;

j--;

}

}

else if(flag[i][j]==123)

{

int k=min(mindis[i-1][j-1],mindis[i-1][j],mindis[i][j-1]);

if(k==mindis[i-1][j-1])

{

if(mindis[i][j]-mindis[i-1][j-1]==2)

cout<<"substitute: "<<a[i-1]<<" to "<<b[i-1]<<endl;

i--;j--;

}

else if(k==mindis[i-1][j])

{

if(mindis[i][j]-mindis[i-1][j]==1)

cout<<"delete:"<<a[i-1]<<endl;

i--;

}

else if(k==mindis[i][j-1])

{

if(mindis[i][j]-mindis[i][j-1]==1)

cout<<"insert:"<<b[j-1]<<endl;

j--;

}

}

cout<<mindis[i][j]<<" ";

}

int m=0;

if(i==0&&j>1)

{

for(m=j;m>=1;m--)

{

if(mindis[i][m]-mindis[i][m-1]==1)

cout<<"insert:"<<b[m-1]<<endl;

cout<<mindis[i][m-1]<<" ";

}

}

else if(j==0&&i>1)

{

for(m=i;m>=1;m--)

{

if(mindis[m][j]-mindis[m-1][j]==1)

cout<<"delete:"<<a[m-1]<<endl;

cout<<mindis[m-1][j]<<" ";

}

}

}

return 0;

}

/\*

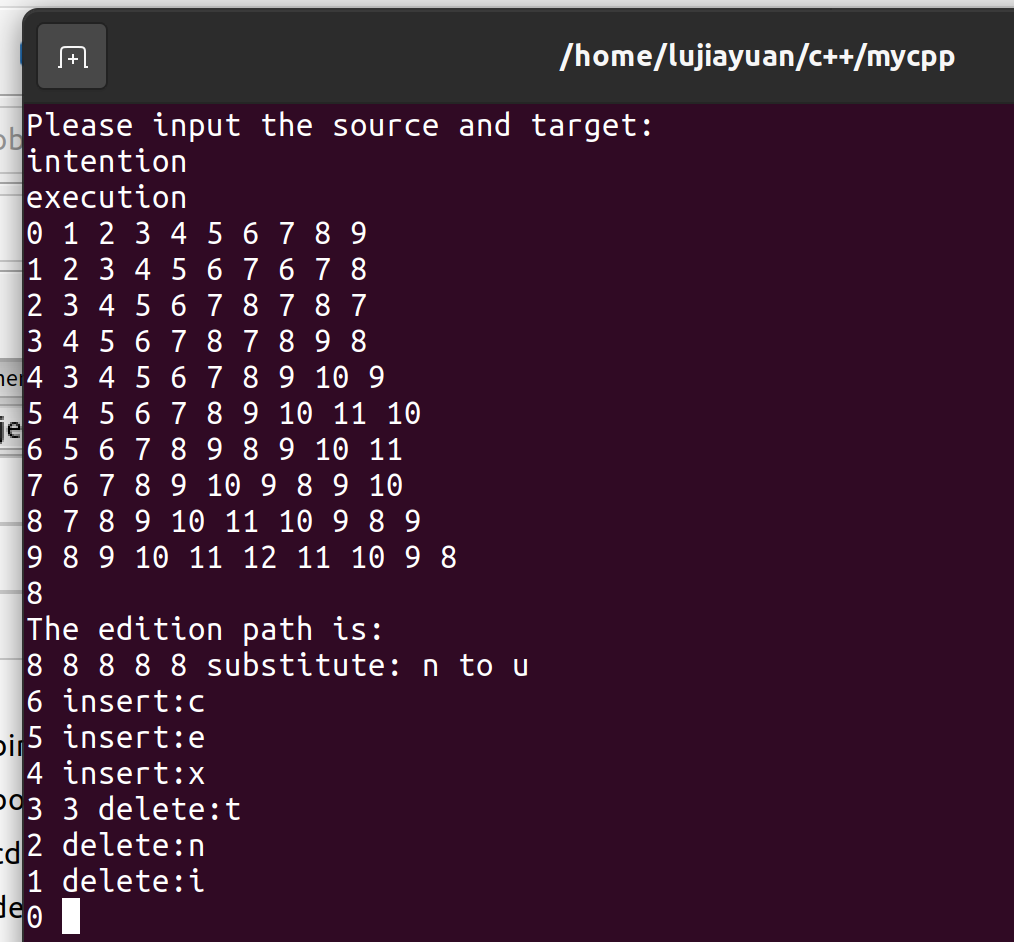
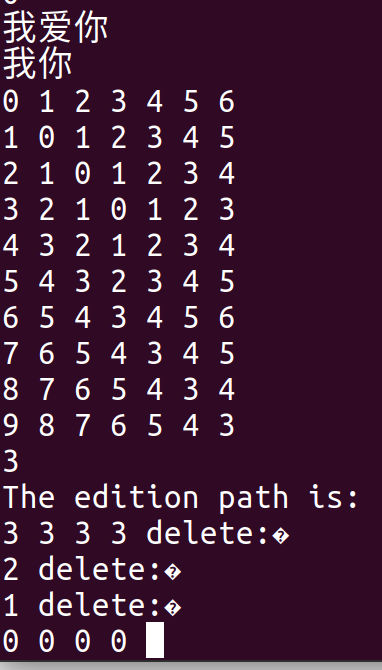
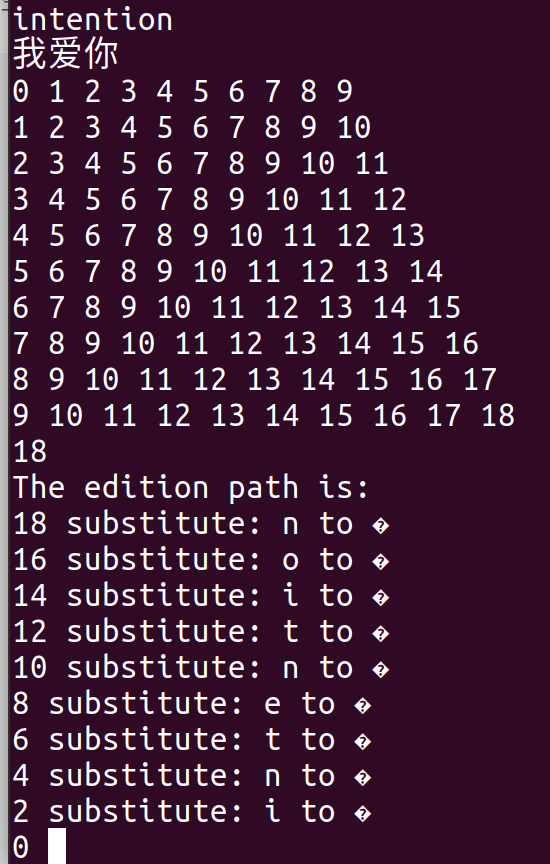
intention

execution

\*/

1. **实验结果**

1、结果截屏：

1. 实验结果说明：
2. 前两行为输入，先输入的要转化为后输入的；
3. 之后打印动态规划计算出的mindis矩阵，第一行为目标字 符串，第一列为源字符串，矩阵右下角为最小编辑距离，本列 例为8；
4. 矩阵下面第一行为两个字符串转换的最小编辑距离结果， 本例为8；
5. 打印回退路径，从截图中的最下面一行开始为由 intention转化成execution的过程，打印形式为 “mindist 操 作：字母”，其中重复的mindist表示没有操作，例如本例中的 3和8，打印了它们的mindist后由于没有操作，所以继续打印下 一个点的mindist而不再回车。

3、实验结果评估：

（1）评估标准：在短时间内能够准确计算出两个字符串转换的最小编辑距离，同时可以得到其中一条转化路径。

（2）度量指标：程序运行时间，是否计算出正确的最小编辑距离，以及是否可以沿得到的路径成功转化。

（3）评估结果：瞬间可得到结果，并且经过手工计算已验证该程序得到的最小编辑距离是正确的，沿转化路径也可以得到目标字符串。

（4）中英文说明：由于UTF-8的一个中文汉字为3字节，若仍然认为一个汉字的删除、插入代价为1，则汉字之间转换的最小编辑距离为所得结果的1/3，汉字与英文之间转换的最小编辑距离需要根据回退路径来重新计算（凡是涉及汉字的操作都把代价除以3，而涉及字母的操作代价不变即可）。

1. **遇到问题及解决思路**
2. 环境问题：

由于我使用的是linux系统（之前已经配置好深度学习的环境了，所以想编程都在这个环境里来实现），ubuntu的vscode有一些功能是被简化的，如对中文的兼容性不好，因此我下载了IDE来编程；另外ubuntu每一次对c++的编译都需要配置脚本，因此我尽量在一个文件里面删改代码；

1. 编程过程中的问题：

刚开始我没有区分源字符串和目标字符串，所以在回退过程中把操作和要操作的字符串弄混了，因此我还是从头梳理代码，弄清楚每一步是对哪个字符串的哪个字母进行何种操作，将回退过程的代码改通过了。