// KMP BM 算法性能实验报告 代码

#include<iostream>

#include<string>

#include<vector>

#include<ctime>

#include<iomanip>

using namespace std;

clock\_t startime,midtime, endtime;

int KMP\_count = 0;

int BM\_count = 0;

void getnext(string t, int\* nextval)

{

int i = 0, j = -1;

nextval[0] = -1;

while (i < t.length())

{

if (j == -1 || t[i] == t[j])

{

i++; j++;

nextval[i] = j;

}

else

j = nextval[j];

}

}

int KMP(string p, string t)

{

int i = 0, j = 0;

int\* nextval = new int[t.length()];

getnext(t, nextval);

while (i < p.length() && j < t.length())

{

KMP\_count++;

if (j == -1 || p[i] == t[j])

{

i++; j++;

}

else

j = nextval[j];

}

if (j >= t.length())

return i - t.length();

else

return -1;

}

void getbc(string p, vector<int> bc)

{

for (int i = 0; i < 256; i++)

bc[i] = -1;

for (int i = 0; i < p.length(); i++)

{

bc[p[i]] = i;

}

}

void getgs(string t, vector<int> suffix, vector<bool> prefix)

{

int n = t.length();

for (int i = 0; i < n - 1; i++)

{

suffix[i] = -1;

prefix[i] = false;

}

for (int i = 0; i < n - 1; i++)

{

int j = i;//从第一个字符开始遍历，t[j]

int k = 0;//最后一个字符的变化，对应下面的t[n - 1 - k]

while (j >= 0 && t[j] == t[n - 1 - k])//和最后一个字符对比，相等则倒数第二个

{

j--;

k++;

suffix[k] = j + 1;//如果k=1，则是一个字符长度的后缀对应匹配位置的值

}

if (j == -1)//说明有前缀字符对应

prefix[k] = true;

}

}

//返回好后缀移动的次数,index为坏字符位置-其后面就是好后缀，size为str大小

int getGsMove(vector<int> suffix, vector<bool> prefix, int index, int size)

{

int len = size - index - 1;//好字符的长度，因为index为坏字符位置，所以要多减1

if (suffix[len] != -1)//当前len长度的后缀坏字符串前边有匹配的字符

{

return index + 1 - suffix[len];//后移位数 = 好后缀的位置(index + 1) - 搜索词中的上一次出现位置

}

//index为坏字符，index+1为好后缀，index+2为子好后缀

for (int i = index + 2; i < size; i++)

{

if (prefix[size - i])//因为prefix从1开始

return i;//移动当前位置离前缀位置，acba-对应a移动3

}

return 0;

}

//返回找到匹配字符串的头，否则返回-1

int BM(string str, string pattern)

{

int n = str.size();

int m = pattern.size();

vector<int> bc(256);//坏字符数组

vector<int> suffix(m);

vector<bool> prefix(m);

getbc(pattern, bc);

getgs(pattern, suffix, prefix);

int i = 0;

while (i <= n - m)

{

int j = 0;

BM\_count++;

for (j = m - 1; j >= 0; j--)

{

if (pattern[j] != str[i + j])//从后往前匹配str和pattern

{

BM\_count++;

break;

}

}

if (j < 0)//匹配结束

{

return i;

}

else

{

int numBc = j - bc[str[i + j]];//bc移动的位数

int numGs = 0;

if (j < m - 1)//最后一个字符就是坏字符，无需判断好字符

{

numGs = getGsMove(suffix, prefix, j, m);//Gs移动的位数

}

i += numBc > numGs ? numBc : numGs;

}

}

return -1;

}

int main()

{

startime = clock();

string s1, s2;

cin >> s1 >> s2;

midtime = clock();

cout << KMP(s1, s2);

cout << endl;

endtime = clock();

double time = (double)(endtime - startime) / CLOCKS\_PER\_SEC;

cout << "KMP 算法耗时：" << time << endl;

cout << "KMP算法比较次数：" << KMP\_count << endl;

cout << BM(s1, s2);

cout << endl;

endtime = clock();

time = (double)(endtime - time + midtime - startime) / CLOCKS\_PER\_SEC;

cout << "BM 算法耗时：" << time << endl;

cout << "BM算法比较次数：" << BM\_count << endl;

return 0;

}