201533661 이승수’s Algorithm homework#4 date: 2016.09.22

Find Subsequence

<code>

#include <stdio.h>

#include <string.h>

FILE \*inF, \*outF;

char stringX[10000] = { '0' };

char stringZ[100] = { '0' };

char frontZ, rearZ;

int lengthZ = 0;

int lengthX = 0;

int FindSubSequence(int frontIndexX, int rearIndexX);

int factorial(int n);

void main()

{

/\*int i = 0;

gets(stringX);

printf("%s\n",stringX);

while (stringX[i] != '\n')

{

printf("%c",stringX[i]);

i++;

}\*/

inF = fopen("input#2.txt", "r");

fscanf(inF, "%s", stringX);

fscanf(inF, "%s", stringZ);

fclose(inF);

printf("%s(%s)\n", stringX, stringZ);

lengthZ = strlen(stringZ);

lengthX = strlen(stringX);

frontZ = stringZ[0];

rearZ = stringZ[lengthZ - 1];

//find duplicated index

//find index of character same as frontZ,rearZ. store them in array.

int indexFront[10] = { 0 };

int numF = 0;

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

{

if (stringX[i] == frontZ)

indexFront[numF++] = i;

}

int indexRear[10] = { 0 };

int numR = 0;

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

{

if (stringX[i] == rearZ)

indexRear[numR++] = i;

}

//find subsequence

int occurence = 0;

for (int front = 0; front < numF; front++)

{

for (int rear = 0; rear < numR; rear++)

{

occurence += FindSubSequence(indexFront[front], indexRear[rear]);

}

}

printf("\noccurence: %d", occurence);

outF = fopen("output#2.txt", "w");

fprintf(outF, "%d", occurence);

fclose(outF);

}

int FindSubSequence(int frontIndexX, int rearIndexX)///////0,length

{

if (frontIndexX >= rearIndexX)

return 0;

int subSequence = 0;

int duplicated = 1;

//char[lengthZ + 1][lengthX + 1] = {'0'};

int currentX = frontIndexX, currentZ = 0;

int cursorX = currentX, cursorZ = 0;

while (cursorX <= rearIndexX&&cursorZ<lengthZ)////////

{

int checkX = cursorX;

int countStraightinX = 1;//count how many duplicated sequence in straight

if (cursorX < (lengthX - 2))

{

while (stringX[checkX + 1] == stringX[checkX + 2])

{

countStraightinX++;

checkX++;

}

}

int checkZ = cursorZ;

int countStraightinZ = 1;

if (cursorZ < (lengthZ - 2))

{

while (stringZ[checkZ + 1] == stringZ[checkZ + 2])

{

countStraightinZ++;

checkZ++;

}

}

duplicated \*= (factorial(countStraightinX) / (factorial(countStraightinZ)\*factorial(countStraightinX - countStraightinZ)));

if (stringZ[cursorZ + 1] == stringX[cursorX + 1])

{

cursorZ += countStraightinZ;

cursorX += countStraightinX;

}

else

{

cursorX += countStraightinX;

}

//if ()

}

if ((cursorZ - 1) == (lengthZ - 1) && (cursorX - 1) == rearIndexX)

subSequence++;

return subSequence\*duplicated;

}

int factorial(int n)

{

if (n < 0)

printf("factorial error:%d\n", n);

if (n == 0)

return 1;

else

return n\*factorial(n - 1);

} x