**算法训练 统计单词个数**

时间限制：1.0s   内存限制：256.0MB

问题描述

　　给出一个长度不超过200的由小写英文字母组成的字母串(约定;该字串以每行20个字母的方式输入，且保证每行一定为20个)。要求将此字母串分成k份 (1<k<=40)，且每份中包含的单词个数加起来总数最大(每份中包含的单词可以部分重叠。当选用一个单词之后，其第一个字母不能再用。例 如字符串this中可包含this和is，选用this之后就不能包含th)。  
　　单词在给出的一个不超过6个单词的字典中。  
　　要求输出最大的个数。

输入格式

　　第一行有二个正整数(p，k)  
　　p表示字串的行数;  
　　k表示分为k个部分。  
　　接下来的p行，每行均有20个字符。  
　　再接下来有一个正整数s，表示字典中单词个数。(1<=s<=6)  
　　接下来的s行，每行均有一个单词。

输出格式

　　每行一个整数，分别对应每组测试数据的相应结果。

样例输入

1 3  
thisisabookyouareaoh  
4  
is  
a  
ok  
sab

样例输出

7

数据规模和约定

　　长度不超过200，1<k<=40，字典中的单词数不超过6。

锦囊1

动态规划。

锦囊2

每计算每个区间的单词数。 用F[i,j]表示前i个字符分成k份的最大单词数，则F[i,j]=F[u,j-1]+Cnt(u+1,i)，其中u

本题的C++参考代码如下：

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

using namespace std;

//ifstream fin("input.txt");

//#define cin fin

int p, k;

int s;

string inputString;

vector<string> words;

void getInput();

int count();

int main()

{

getInput();

int n = count();

cout << n;

return 0;

}

void getInput()

{

// 处理输入

cin >> p >> k;

string temp;

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

{

cin >> temp;

inputString += temp;

}

cin >> s;

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

{

cin >> temp;

words.push\_back(temp);

}

}

int count()

{

vector<int> Sets;

int dividedNum = 0;

int counts = 0;

int begin = 0;

int end = -1;

for (int i = 0; i < inputString.size(); i++)

{

begin = i;

bool isMatching = false;

for (int j = 0; j < words.size(); j++)

{

size\_t wordSize = words[j].size();

if (inputString.substr(i, wordSize) == words[j])

{

isMatching = true;

counts++;

if (begin <= end)

{

\*Sets.rbegin() = \*Sets.rbegin() + 1;

if (begin + wordSize - 1 > end)

end = begin + wordSize - 1;

}

else

{

Sets.push\_back(0);

end = begin + wordSize - 1;

}

}

if (isMatching == true)

break;

}

if (isMatching == false && i >= end)

dividedNum++;

}

if (k < dividedNum)

return counts;

else

return counts - (k - dividedNum);

}

本题的C参考代码如下：

#include <stdio.h>

#define INF\_MIN 1<<31

#define MAXK 41

#define STR\_SIZE 201

#define WORD\_SIZE 16

int p, k, s;

char str[STR\_SIZE];

char word[6][WORD\_SIZE];

int Cnt\_val[STR\_SIZE][STR\_SIZE]={0};

int Cnt(int from, int end)

{

int i, j, k;

int count = 0;

if(Cnt\_val[from][end] <= 0){

for(i=from; i<=end; i++)

for(j=0; j<s; j++){

if(str[i] == word[j][0]){

for(k=1; word[j][k] != '\0' && i+k<=end; k++){

if(word[j][k] != str[i+k]){

break;

}

}

if(word[j][k] == '\0'){

count ++;

break;

}

}

}

Cnt\_val[from][end] = count;

}

return Cnt\_val[from][end];

}

int main()

{

int i, j, u;

int count\_p;

char ch;

int f[STR\_SIZE][MAXK];

scanf("%d%d", &p, &k);

getchar();

count\_p = 0;

for(i=0; i<STR\_SIZE; i++){

ch = getchar();

if(ch == '\n'){

count\_p ++;

if(count\_p == p){

str[i] = '\0';

break;

}

i --;

}else{

str[i] = ch;

}

}

scanf("%d", &s);

for(i=0; i<s; i++){

scanf("%s", word[i]);

}

int temp, max;

int strlen = 20\*p;

for(i=0; i<strlen; i++){

f[i][1] = Cnt(0, i);

}

for(j=2; j<=k; j++)

{

max = INF\_MIN;

for(i=k-1; i<strlen; i++){

for(u=j-2; u<i; u++){

temp = f[u][j-1] + Cnt(u+1, i);

max = temp>max?temp:max;

}

f[i][j] = max;

}

}

printf("%d", f[strlen-1][k]);

return 0;

}

本题的Java参考代码如下：

import java.util.Scanner;

public class Main{

static int p;

static int k;

static String S;

static int wordsize;

static String[] Word;

static int num = 0;

static int maxnum = 0;

static int kefenge = 0;

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

p = sc.nextInt();

k = sc.nextInt();

S = "";

sc.nextLine();

for (int i = 0; i < p; i++) {

S = S + sc.nextLine();

}

wordsize = sc.nextInt();

Word = new String[wordsize];

for (int i = 0; i < wordsize; i++) {

Word[i] = sc.next();

}

zhaodanci(S);

if(kefenge<k){

num = num-(k-1);

}

System.out.println(num);

// System.out.println(kefenge);

}

public static void zhaodanci(String s){

boolean meizhaodao = true;

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

for(int j = 0;j<wordsize;j++){

if(s.substring(i,s.length()).length()<Word[j].length()){

continue;

}

if(s.substring(i,i+Word[j].length()).length()<Word[j].length()){

continue;

}

if(s.substring(i,i+Word[j].length()).equals(Word[j])){

// System.out.println(s+":"+Word[j]);

num++;

if(i != 0){

kefenge++;

}

zhaodanci(s.substring(i+1,s.length()));

return;

}

}

}

}

}