**算法训练 Tricky and Clever Password**

时间限制：2.0s   内存限制：256.0MB

问题描述

　　在年轻的时候，我们故事中的英雄——国王 Copa——他的私人数据并不是完全安全地隐蔽。对他来说是，这不可接受的。因此，他发明了一种密码，好记又难以破解。后来，他才知道这种密码是一个长度为奇数的回文串。  
  
　　Copa 害怕忘记密码，所以他决定把密码写在一张纸上。他发现这样保存密码不安全，于是他决定按下述方法加密密码：他选定一个整数 X ，保证 X 不小于 0 ，且 2X 严格小于串长度。然后他把密码分成 3 段，最前面的 X 个字符为一段，最后面的 X 个字符为一段，剩余的字符为一段。不妨把这三段依次称之为 prefix, suffix, middle 。显然， middle 的长度为一个大于 0 的奇数，且 prefix 、 suffix 的长度相等。他加密后的密码即为 A + prefix + B + middle + C + suffix ，其中 A 、 B 、 C 是三个由 Copa 选定的字符串，且都有可能为空， + 表示字符串相连。  
  
　　许多年过去了。Copa 昨天找到了当年写下加密后字符串的那张纸。但是，Copa 把原密码、A、B、C 都忘了。现在，他请你找一个尽量长的密码，使得这个密码有可能被当年的 Copa 发明、加密并写下。

输入格式

　　输入包含一个只含有小写拉丁字母的字符串，长度在 1 到 10^5 之内。

输出格式

　　第一行包含一个整数 k ，表示你找到的原密码分成的 3 个部分中有多少个非空字符串。显然 k in {1, 3} 。接下来 k 行，每行 2 个用空格分开的整数 x\_i l\_i ，表示这一部分的起始位置和长度。要求输出的 x\_i 递增。  
  
　　起始位置 x\_i 应该在 1 到加密后的字符串长度之间。 l\_i 必须是正整数，因为你只要输出非空部分的信息。 middle 的长度必须为奇数。  
  
　　如果有多组答案，任意一组即可。提示：你要最大化的是输出的 l\_i 的总和，而不是 k 。

样例输入

abacaba

样例输出

1  
1 7

样例输入

axbya

样例输出

3  
1 1  
2 1  
5 1

样例输入

xabyczba

样例输出

3  
2 2  
4 1  
7 2

数据规模和约定

　　对于 10% 的数据： n <= 10  
  
　　对于 30% 的数据： n <= 100  
  
　　对于 100% 的数据： n <= 100000  
  
　　存在 20% 的数据，输出文件第一行为 1 。

本题的C++参考代码如下：

#include<stdio.h>

#include<string.h>

#define INF 0x3f3f3f3f

char s[100010], s1[200010];

int mana[100010], L, next[200010], match[100010], r[100010];

int min(int x, int y)

{

return x<y?x:y;

}

void getmana()

{

mana[0] = 0;

int id = 0, right = 0;

for(int i = 1; i < L-1; ++i)

{

if(right > i)

mana[i] = min(mana[2\*id - i], right - i);

else

mana[i] = 0;

while(s[i+mana[i]+1] == s[i-mana[i]-1] && (i+mana[i]+1) < L && (i-mana[i]-1) >= 0)

mana[i]++;

if(mana[i]+i > right)

{

id = i;

right = mana[i]+i;

}

}

}

void getnext(char \*temp)

{

memset(next, 0, sizeof(next));

next[0] = -1;

int i = 0, j = -1;

while(temp[i])

{

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

{

i++;

j++;

next[i] = j;

}

else

j = next[j];

}

}

int main()

{

int i, j, k, l;

while(gets(s) != NULL)

{

L = strlen(s);

getmana();

for(i = L-1; i >= 0; --i)

s1[L-1-i] = s[i];

s1[L] = '#';

s1[L+1] = '\0';

strcat(s1,s);

getnext(s1);

for(i = L+1; i <= 2\*L+1; ++i)

match[i-L-1] = next[i];

memset(r, -1, sizeof(r));

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

{

if(match[i] > match[i-1])

r[match[i]-1] = i-1;

else

match[i] = match[i-1];

}

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

{

if(r[i] == -1)

r[i] = r[i-1];

// printf("%d ", r[i]);

}

// printf("\n");

int ans[3][2] = {0}, max = 0;

bool ok = true;

for(i = 1; i < L-1; ++i)//the length must bigger than 3

{

j = i+mana[i]+1;

if(j >= L || i-mana[i] <= 0)

j = L;

else

{

if(r[L-j-1] == -1)

j = L;

else

{

if(r[L-j-1] >= i-mana[i])

j = L - match[i-mana[i]];

else

j = L - match[r[L-j-1]+1];

}

}

if(j < L)

{

if(2\*(L-j) + mana[i]\*2 + 1 > max)

{

max = 2\*(L-j) + mana[i]\*2 + 1;

ans[0][0] = r[L-j-1] - L + j + 1;

ans[0][1] = L-j;

ans[1][0] = i-mana[i];

ans[1][1] = 2\*mana[i]+1;

ans[2][0] = j;

ans[2][1] = L-j;

ok = false;

// printf("1 %d\n", max);

}

}

else

{

if(2\*mana[i]+1 > max)

{

max = 2\*mana[i]+1;

ok = true;

ans[1][0] = i-mana[i];

ans[1][1] = 2\*mana[i]+1;

}

}

}

if(ok)

{

if(ans[1][1] == 0)

ans[1][1] = 1;

printf("1\n%d %d\n", ans[1][0]+1, ans[1][1]);

}

else

{

printf("3\n");

printf("%d %d\n", ans[0][0]+1, ans[0][1]);

printf("%d %d\n", ans[1][0]+1, ans[1][1]);

printf("%d %d\n", ans[2][0]+1, ans[2][1]);

}

}

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define \_min(x,y) ((x)<(y)?(x):(y))

#define \_max(x,y) ((x)>(y)?(x):(y))

void KMP(char \*target\_string,char \*match\_string,int \*match)

{

int target\_size=strlen(target\_string);

int match\_size=strlen(match\_string);

int i,j=-1;

int \*intit\_array=(int \*)malloc(sizeof(int)\*target\_size);

intit\_array[0]=-1;

for(i=1;i<target\_size;++i)

{

while(j>-1&&target\_string[i]!=target\_string[j+1])

j=intit\_array[j];

if(target\_string[i]==target\_string[j+1])

++j;

intit\_array[i]=j;

}

j=-1;

for(i=0;i<match\_size;++i)

{

while(j>-1&&match\_string[i]!=target\_string[j+1])

j=intit\_array[j];

if(match\_string[i]==target\_string[j+1])

++j;

match[i]=j+1;

// if(j==target\_size-1)

// j=intit\_array[j];

}

// free(intit\_array);

}

void manacher(char \*match\_string,int \*match)

{

int size=strlen(match\_string);

int i;

int most\_right\_find=0;

int right\_mid=0;

match[0]=1;

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

{

if(i<most\_right\_find)

match[i]=\_min(match[2\*right\_mid-i],most\_right\_find-i);

else match[i]=1;

while(match\_string[i-match[i]]==match\_string[i+match[i]])

++match[i];

if(match[i]+i>most\_right\_find)

{

most\_right\_find=match[i]+i;

right\_mid=i;

}

}

return;

}

int main()

{

char \*input\_string=(char \*)malloc(100003\*sizeof(char));

int size;

int \*match,\*kmp\_match;

int i,j;

int half;

char \*target\_string;

int target\_size;

int \*dp;

int match\_max=0;

int mid\_len;

int pre\_suf;

int left\_begin=-1;

int \*tail;

int left\_length=0;

int mid\_begin=0;

int mark=0;

input\_string[0]='@';

scanf("%s",input\_string+1);

size=strlen(input\_string);

input\_string[size]='$';

input\_string[++size]=0;

match=(int \*)malloc(sizeof(int)\*size);

manacher(input\_string,match);

i=-1;

j=size-1;

half=(size+1)/2;

target\_size=size-half;

target\_string=(char \*)malloc(sizeof(char)\*target\_size);

while(j>half)

target\_string[++i]=input\_string[--j];

target\_string[++i]=0;

kmp\_match=(int \*)malloc(sizeof(int)\*size);

KMP(target\_string,input\_string,kmp\_match);

dp=(int \*)malloc(sizeof(int)\*size);

tail=(int \*)malloc(sizeof(int)\*size);

dp[0]=0;

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

{

if(kmp\_match[i]>dp[i-1])

dp[i]=kmp\_match[i],tail[i]=i;

else dp[i]=dp[i-1],tail[i]=tail[i-1];

}

for(i=1;i<size-1;++i)

{

mid\_len=2\*match[i]-1;

pre\_suf=\_min(dp[i-match[i]],size-(i+match[i])-1);

if(mid\_len+2\*pre\_suf>match\_max)

{

match\_max=mid\_len+2\*pre\_suf;

left\_begin=tail[i-match[i]]-pre\_suf+1;

if(pre\_suf>0)

{

left\_length=pre\_suf;

mark=1;

}

else mark=0;

mid\_begin=i-match[i]+1;

}

}

if(mark==1)

printf("3\n%d %d\n%d %d\n%d %d",left\_begin,left\_length,mid\_begin,match\_max-2\*left\_length,size-1-left\_length,left\_length);

else printf("1\n%d %d",mid\_begin,match\_max);

return 0;

}

本题的Java参考代码如下：

import java.util.Scanner;

public class Main{

static String str;

static int a[][],b[][],next[];

static int k=1,t=0;

static void manacher(int star,int end){

int i,mid=1,num,max=1,dex=1;

String sub=str.substring(star, end);

char s[]=new char[sub.length()+2];

int p[]=new int[s.length];

p[0]=p[1]=1;

s[0]='$';

s[s.length-1]='!';

for(i=0;i<sub.length();i++)

s[i+1]=sub.charAt(i);

for(i=2;i<s.length-max-1;i++){

num=Math.min(p[2\*mid-i], p[mid]+mid-i);

while(s[i-num]==s[i+num])

num++;

p[i]=num;

if(i+p[i]>mid+p[mid])

mid=i;

if(num>max){

max=num;

dex=i;

}

}

a[1][1]=max\*2-1;

a[1][0]=dex-max+star+1;

}

static void getnext(){

int len=str.length()-1;

while(t>0&&str.charAt(len-k)!=str.charAt(len-t))

t=next[t];

if(str.charAt(len-k)==str.charAt(len-t))

t++;

k++;

next[k]=t;

}

static void search(){

int cur1=1;

manacher(0,str.length());

b[1][0]=a[1][0];

b[1][1]=a[1][1];

int i,j=0,len=str.length()-1,maxlen=0,maxcur=0,cur=0;

for(i=0;i<len-j;i++){

if(i==len-j-1){

if(j>maxcur){

maxcur=j;

a[0][0]=i-maxlen+1;

a[0][1]=maxlen;

a[2][0]=str.length()-maxlen+1;

a[2][1]=a[0][1];

manacher(i,str.length()-j);

if(a[0][1]\*2+a[1][1]>b[0][1]\*2+b[1][1]){

for(int q=0;q<3;q++)

for(int w=0;w<2;w++)

b[q][w]=a[q][w];

cur1=0;

}

}

cur=1;

}

while(j>0&&str.charAt(i)!=str.charAt(len-j)){

if(j>maxcur){

maxcur=j;

a[0][0]=i-maxlen+1;

a[0][1]=maxlen;

a[2][0]=str.length()-maxlen+1;

a[2][1]=a[0][1];

manacher(i,str.length()-j);

if(a[0][1]\*2+a[1][1]>b[0][1]\*2+b[1][1]){

for(int q=0;q<3;q++)

for(int w=0;w<2;w++)

b[q][w]=a[q][w];

cur1=0;

}

}

cur=1;

j=next[j];

}

if(str.charAt(i)==str.charAt(len-j)){

j++;

if(j>maxlen){

maxlen++;

getnext();

}

}

}

if(cur==1&&cur1==0)

System.out.println(3+"\n"+b[0][0]+" "+b[0][1]+"\n"+b[1][0]+" "+b[1][1]+"\n"+b[2][0]+" "+b[0][1]);

else

System.out.println(1+"\n"+b[1][0]+' '+b[1][1]);

}

public static void main(String[]args){

Scanner scan=new Scanner(System.in);

str=scan.nextLine();

a=new int[3][2];

b=new int[3][2];

next=new int[str.length()/2+2];

next[1]=0;

search();

scan.close();

}

}