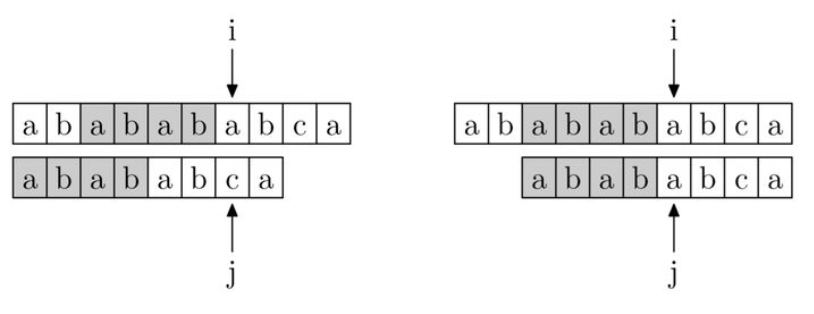
实验六：

2021/4/26

一．

1. 实现字符串匹配KMP算法。

（2）分析：在遇到有不同的字符时，不必从母串的下一个字符开始再进行判断。我们看到如果是在 j 位 失配，那么影响 j 指针回溯的位置的其实是第 j −1 位的 最大真相等真前后缀的长度（PMT）值，所以为了编程的方便， 我们不直接使用PMT数组，而是将PMT数组向后偏移一位。我们把新得到的这个数组称为next数组。下面给出根据next数组进行字符串匹配加速的字符串匹配程序。其中要注意的一个技巧是，在把PMT进行向右偏移时，第0位的值，我们将其设成了-1，这只是为了编程的方便，并没有其他的意义。



（3）代码实现：

#include <stdio.h>

#include <string.h>

#define NUM 80

void PMT(char T[],int next[]){

int i ,j ;

memset(next,0,NUM\*sizeof(int));

next[0] = 0;

if(T[0] == T[1]){

next[1] = 1;

}else{

next[1] = 0;

}

for( i = 2 ; T[i] != '\0' ; i++ ){ //计算【0，i】的最大真前后缀的长度

j = i;

while( j != 0 ){

if( T[i] == T[next[j-1]] ){

next[i] = next[j-1] + 1;

break;

}else{

j = next[j-1];

}

}

}

}

int KMP(char T[],char S[],int next[]){

int i = 0;

int j = 0;

while (i < strlen(T) && j < strlen(S))

{

if (T[i] == S[j])

{

i++;

j++;

}

else{

i = next[j];

}

if (j == strlen(S))

return i - j;

else

return -1;

}

}

void main(){

char S[19] = "ababcaababbabccbab";

char T[13] = "ababbabc";

int next[NUM] = {0};

PMT(T,next);

int a = KMP(S,T,next);

printf("%d",a);

}