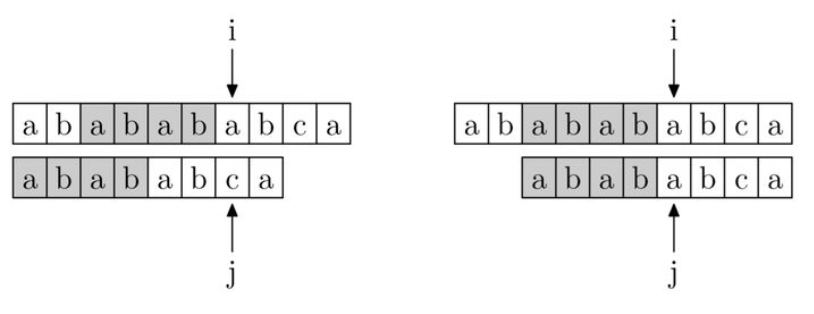
实验八：

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);

}

二．

（1）题目描述：二路归并排序

（2）思路：归并排序是利用递归不断将数组分成两份直到分成每个数组一个元素。然后将分好的有序小数组两两比较进行合并。

（3）代码实现

#include <stdio.h>

#include <stdlib.h>

void merge(int arr[],int temp[],int left,int mid,int right){

int l\_pos = left; //用来存储归并的两个数组中左半部分的首元素

int r\_pos = mid+1;//用来存储归并的两个数组中右半部分的首元素

int pos = left;//用来存储已经排序好的元素位置

while(l\_pos <= mid && r\_pos <= right){//当两个数组中有剩余元素时

if(arr[l\_pos] < arr[r\_pos]){//将两个有序数组合并

temp[pos++] = arr[l\_pos++];

}else{

temp[pos++] = arr[r\_pos++];

}

}

while(l\_pos <= mid){//如果只有左数组有剩余元素则直接填充

temp[pos++] = arr[l\_pos++];

}

while(r\_pos <= right){//如果只有右数组有剩余元素则直接填充

temp[pos++] = arr[r\_pos++];

}

while(left <= right){//将归并的这一部分复制到元素组中

arr[left] = temp[left];

left++;

}

}

void mergeSort(int arr[],int temp[],int left,int right){

if(left < right){//如果没有划分成只剩一个元素的数组，就继续进行递归划分

int mid = (left+right) / 2;//设置中间值

mergeSort(arr,temp,left,mid);//分别划分左半部分

mergeSort(arr,temp,mid+1,right);//和右半部分

merge(arr,temp,left,mid,right); //对划分好的数组进行合并

}

}

void showArr(int arr[],int n){//打印数组

int i;

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

printf("%d",arr[i]);

}

printf("\n");

}

int main(){

int arr[] = {5,7,4,1,6,9,8,2,3};

int n = 9;

int \*temp = (int\*)malloc(sizeof(int));//分配暂时存储的数组

showArr(arr,n);

if(temp){

mergeSort(arr,temp,0,n-1);//如果成功分配则进行排序

}else{

printf("ERROR!");

}

showArr(arr,n);

}