**实验报告（第 1 次）**

实验名称 二分归并排序

实验时间 2022.3.23 实验地点 实训楼

学 号 姓 名 wby

**一、【实验目的及要求】**

1.掌握分治策略的基本思想。

2.学会运用分治策略的思想解决实际问题（如：二分归并排序）。

3.掌握二分归并排序的思想以及运算步骤。

**二、【实验内容】**

有待排序数组如下：

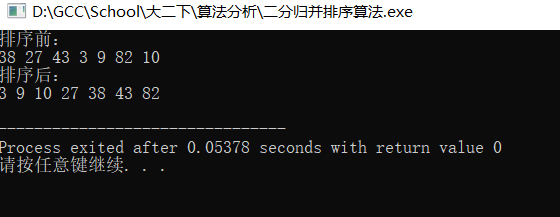
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 38 | 27 | 43 | 3 | 9 | 82 | 10 |

要求使用二分排序的思想将其排好序，排好的顺序如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 3 | 9 | 10 | 27 | 38 | 43 | 82 |

**三、测试数据及结果**

|  |
| --- |
| #include<stdio.h>  #include<stdlib.h>  void print(int arr[], int n)//输出  {  for (int i = 0; i < n; i++)  {  printf("%d ", arr[i]);  }putchar('\n');  }  void getone(int arr[], int tarr[], int left, int mid, int right)//排序且合并  {  int L\_arr = left;//左数组第一位  int R\_arr = mid + 1;//右数组第一位  int i = left;  while (L\_arr <= mid && R\_arr <= right)//左右组对比排序  {  if (arr[L\_arr] < arr[R\_arr])  {  tarr[i++] = arr[L\_arr++];  }  else tarr[i++] = arr[R\_arr++];  }  while (L\_arr <= mid)//剩下左组排序  {  tarr[i++] = arr[L\_arr++];  }  while (R\_arr <= right)//剩下右组排序  {  tarr[i++] = arr[R\_arr++];;  }  while (left <= right)//将临时数组的数据转移  {  arr[left] = tarr[left];  left++;  }  }  void gettwo(int arr[], int tarr[], int left, int right)//递归分组  {  if (left < right)  {  int mid = (left + right) / 2;  gettwo(arr, tarr, left, mid);  gettwo(arr, tarr, mid + 1, right);  getone(arr, tarr, left, mid, right);  }  }  void merge(int arr[], int n)//分配空间  {  int\* tarr = (int\*)malloc(n \* sizeof(int));  if (tarr)//分配成功  {  gettwo(arr, tarr, 0, n - 1);  free(tarr);//释放临时数组空间  }  else printf("error!");//分配失败  }  int main(int argc, char const\* argv[])  {  int arr[] = { 38,27,43,3,9,82,10 };  int n = 7;  printf("排序前：\n");  print(arr, n);  merge(arr, n);  printf("排序后：\n");  print(arr, n);  return 0;  } |

**四、心得体会与自我评价（一定要写）**

算法还是需要再加思考才能完全参悟。

实验报告二

**姓名： wby 学号：**

**实验名称： 最长公共子序列 实验时间： 2022年4月14日星期四**

**实验地点： 实训楼306**

1. **实验目的**

1、熟悉C/C++语言的集成开发环境；

2、通过动态规划算法的示例程序理解动态规划算法的基本思想；

3、运用动态规划算法解决实际问题加深对动态规划算法的理解和运用。

**二、实验内容**

1、动态规划算法思想：

把待求解问题分解成若干个子问题，先求解子问题，然后由这些子问题的解得到原问题的解，但动态规划求解过的子问题的结果会被保留下来，不像递归那样每个子问题的求解都要从头开始返回求解。动态规划求解问题的关键在于获得各个阶段子问题的递推关系式：

（1）分析原问题的最优解性质，刻画其结构特征

（2）递归定义最优值

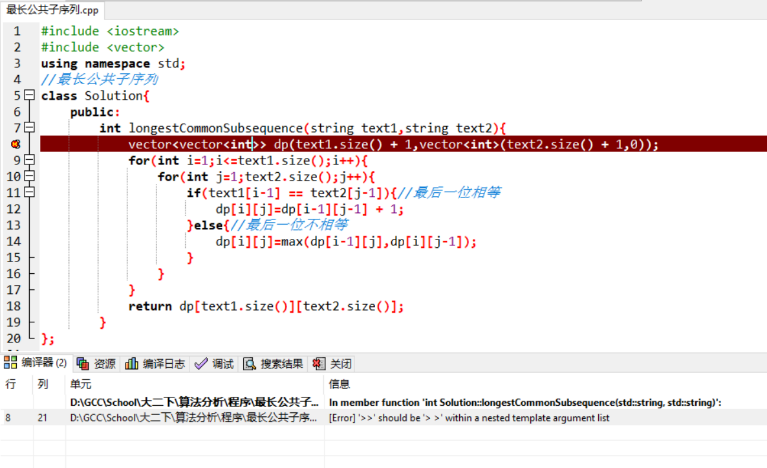
（3）自底向上（由后向前）的方式计算最优值

（4）根据计算最优值时得到的信息，构造一个最优解。

2、最长公共子序列问题：

若给定序列X={x1,x2, ,xm}，则另一序列Z={z1,z2, ,zk}，是X的子序列是指存在一个严格递增下标序列{i1,i2, ,ik}使得对于所有j=1,2, ,k有：zj=xij。例如，序列Z={B，C，D，B}是序列X={A，B，C，B，D，A，B}的子序列，相应的递增下标序列为{2，3，5，7}。

**三、主要问题和解决方案**

**【主要问题】**

【解决方案】

// vector< vector<int> >这里要在<>里空出空格，vector< 定义的类型 >，这样

//【正确代码】vector< vector<int> > dp(text1.size() + 1,vector<int>(text2.size() + 1,0));

**四、测试数据及结果**

|  |
| --- |
| #代码  #include <iostream>  #include <vector>  using namespace std;  //最长公共子序列  class Solution {  public:  int longestCommonSubsequence(string text1,string text2) {  vector< vector< int > > dp(text1.size() + 1,vector<int>(text2.size() + 1,0));  for(int i=1;i<=text1.size();i++){  for(int j=1;j<=text2.size();j++){  if(text1[i-1] == text2[j-1]){//最后一位相等  dp[i][j]=dp[i-1][j-1] + 1;  }else{//最后一位不相等  dp[i][j]=max(dp[i-1][j],dp[i][j-1]);  }  }  }  return dp[text1.size()][text2.size()];  }  };  int main(){  // Solution solution;  // int value = solution.longestCommonSubsequence("abcde","abc");  // cout<<"hello"<<endl;  // cout<<value<<endl;  Solution s1;  int r1 = s1.longestCommonSubsequence("abcde", "abc");  cout << r1 << endl;  // cout << "Hello, world!" << endl;  return 0;  } |
| #测试效果图 |

**五、心得体会与自我评价**

感觉受益匪浅！我很喜欢这个算法。