《数据结构与算法》实验报告3

——排序算法

吉林大学数学学院2020级计算1班 姓名 陈文宇 学号10200115

日期 2023/5/28

一、实验题目

分别编写选择排序（SelectSort）、插入排序（InsertSort）、冒泡排序（BubbleSort）、快速排序（QuickSort）、归并排序（MergeSort）、快速排序（QuickSort）的程序，并验证结果。

二、算法简述（）

记待排序顺序表长度为n。

选择排序：顺序遍历前n-1个元素。将每个元素与其后比它小的最小元素交换（如果有的话）。

插入排序：顺序遍历第2~n个元素。此时每个元素前应是已经排好序的子序列，将该元素插入前面序列合适的位置。

冒泡排序：逆序遍历前n-1个元素。将每个元素“赶”到后面，直到遇到比它还大的元素为止。

快速排序：找到一个基准点（如中间点）。用“将把比基准点小的元素赶到左边，比基准点大的元素赶到右边”的方式，找到基准点排好序以后的位置。然后以此为边界，对基准点左右两个子序列递归应用该方法。

归并排序：递归分割子序列，将两个相邻的子序列按排好序的要求归并得到一个稍大的序列，递归应用该方法。

三、程序代码

|  |
| --- |
| InternalSorting.cpp |
| //陈文宇  //10200115  #include<iostream>  using namespace std;  const int MAXSIZE=20;  const bool TRUE=1;  const bool FALSE=0;  //定义变量类型  typedef int KeyType;  typedef char InfoType;  typedef struct{  KeyType key;  InfoType val;  }RcdType;  typedef struct {  RcdType r[MAXSIZE+1]; //R[0]闲置或作为判别标志的“哨兵单元”  int length;  }SqList;  //函数声明  void SelectPass(SqList &L,int i); //选择排序  void SelectSort(SqList &L);  void InsertPass(SqList &L,int i); //插入排序  void InsertSort(SqList &L);  void BubbleSort(SqList &L); //冒泡排序  int Partition(RcdType R[], int low, int high);  void Qsort(RcdType R[], int s,int t);  void QuickSort(SqList &L); //快速排序  void Merge(RcdType SR[], RcdType TR[], int i, int m, int n); //归并排序  void Msort(RcdType SR[], RcdType TR1[], int s,int t,int n);  void MergeSort(SqList &L);  int main(){  int n=12,m=20;  SqList\* A;  A=(SqList\*)malloc(sizeof(SqList)\*n);  for(int i=0;i<n;i++){  A[i].length=m;  A[i].r[1 ].key=35;  A[i].r[2 ].key=35;  A[i].r[3 ].key=37;  A[i].r[4 ].key=49;  A[i].r[5 ].key=63;  A[i].r[6 ].key=30;  A[i].r[7 ].key=81;  A[i].r[8 ].key=70;  A[i].r[9 ].key=54;  A[i].r[10].key=46;  A[i].r[11].key=15;  A[i].r[12].key=70;  A[i].r[13].key=71;  A[i].r[14].key=12;  A[i].r[15].key=65;  A[i].r[16].key=20;  A[i].r[17].key=27;  A[i].r[18].key=86;  A[i].r[19].key=44;  A[i].r[20].key=22;  }  //49 38 65 49 76 13 27 52  //35 35 37 49 63 30 81 70 54 46 15 70 71 12 65 20 27 86 44 22  //25 32 98 50 35 86 61 27 92 28 67 75 96 11 19 85 94 85 92 76  //-------------------------------------------------------  printf("选择排序\n");  cout<<"排序前:";  for(int i=1;i<=m; i++){  cout<<A[0].r[i].key<<" ";  }  SelectSort(A[0]);  cout<<endl;  cout<<"排序后:";  for(int i=1;i<=m; i++){  cout<<A[0].r[i].key<<" ";  }  cout<<endl;    //--------------------------------------------------------  printf("\n\n\n插入排序\n");  cout<<"排序前:";  for(int i=1;i<=m; i++){  cout<<A[1].r[i].key<<" ";  }  InsertSort(A[1]);  cout<<endl;  cout<<"排序后:";  for(int i=1;i<=m; i++){  cout<<A[1].r[i].key<<" ";  }  cout<<endl;    //-------------------------------------------------------  printf("\n\n\n冒泡排序\n");  cout<<"排序前:";  for(int i=1;i<=m; i++){  cout<<A[2].r[i].key<<" ";  }  BubbleSort(A[2]);  cout<<endl;  cout<<"排序后:";  for(int i=1;i<=m; i++){  cout<<A[2].r[i].key<<" ";  }  cout<<endl;    //-------------------------------------------------------  printf("\n\n\n快速排序\n");  cout<<"排序前:";  for(int i=1;i<=m; i++){  cout<<A[3].r[i].key<<" ";  }  QuickSort(A[3]);  cout<<endl;  cout<<"排序后:";  for(int i=1;i<=m; i++){  cout<<A[3].r[i].key<<" ";  }  cout<<endl;    //------------------------------------------  printf("\n\n\n归并排序\n");  cout<<"排序前:";  for(int i=1;i<=m; i++){  cout<<A[4].r[i].key<<" ";  }  MergeSort(A[4]);  cout<<endl;  cout<<"排序后:";  for(int i=1;i<=m; i++){  cout<<A[4].r[i].key<<" ";  }  cout<<endl;    free(A);  }  //选择排序  void SelectPass(SqList &L,int i){  //已知L.r[1:1:i-1]中关键字非递减排序，本算法实现第i躺选择排序  //即在L.r[i:1:n]的记录中选出关键字 最小的记录L.r[j]和r[i]进行交换  int j=i;  RcdType W;    for(int k=i+1; k<=L.length; k++)  if(L.r[k].key<L.r[j].key) j=k;  if(i!=j){  W=L.r[j];  L.r[j]=L.r[i];  L.r[i]=W;  }    }// SelectPass  //顺序表的选择排序  void SelectSort(SqList &L){  RcdType W;  int j;  int k;  for(int i=1; i<L.length; i++){  j=i;  for(k=i+1; k<=L.length; k++){  if(L.r[k].key<L.r[j].key) j=k;  }  if(i!=j){  W=L.r[j]; L.r[j]=L.r[i]; L.r[i]=W;  }  }  }//SelectSort  //插入排序  void InsertPass(SqList &L,int i){  int j=i-1;  L.r[0]=L.r[i];    for(; L.r[0].key < L.r[j].key; j--)  L.r[j+1]=L.r[j];  L.r[j+1]=L.r[j];  }//InsertPass  //顺序表的插入排序  void InsertSort(SqList &L){  int j;  for(int i=2; i<=L.length; i++){  if(L.r[i].key < L.r[i-1].key){  L.r[0]=L.r[i];  for(j=i-1; L.r[0].key<L.r[j].key; j--)  L.r[j+1]=L.r[j];  L.r[j+1]=L.r[0];  }//if  }//for  }//InsertSort  // 顺序表的起泡排序  void BubbleSort(SqList &L){  int i=L.length,LastExchangeIndex;  RcdType W;  int j;  while(i>1){  LastExchangeIndex=1;  for(j=1; j<i; j++){  if(L.r[j+1].key<L.r[j].key){  W=L.r[j]; L.r[j]=L.r[j+1]; L.r[j+1]=W;  LastExchangeIndex=j;  }//if  }//for  i=LastExchangeIndex;  }//while  } //BubbleSort  //快速排序算法  int Partition(RcdType R[], int low, int high){  R[0]=R[low];  KeyType pivotkey=R[low].key;  while(low<high){  while(low<high && R[high].key>=pivotkey) --high;  if(low<high) R[low++]=R[high];  while(low<high && R[low].key<=pivotkey) ++low;  if(low<high) R[high--]=R[low];  }//while  R[low]=R[0];  //printf("陈文宇");  return low;  } //Partition  void Qsort(RcdType R[], int s,int t){  int pivotloc;  if(s<t){  pivotloc=Partition(R,s,t);  Qsort(R,s,pivotloc-1);  Qsort(R,pivotloc+1,t);  }//if  }//Qsort  void QuickSort(SqList &L){  Qsort(L.r,1,L.length);  }//QuickSort    //归并排序  void Merge(RcdType SR[], RcdType TR[], int i, int m, int n){  int j=m+1;  int k;  for(k=i; i<=m && j<=n; k++){  if(SR[i].key<=SR[j].key) TR[k]=SR[i++];  else TR[k]=SR[j++];  }//for  while(i<=m) TR[k++]=SR[i++];  while(j<=n) TR[k++]=SR[j++];    } //Merge  void Msort(RcdType SR[], RcdType TR1[], int s,int t,int n){  RcdType TR2[n];  int m;  if(s==t) TR1[s]=SR[s];  else{  m=(s+t)/2;  Msort(SR,TR2,s,m,n);  Msort(SR,TR2,m+1,t,n);  Merge(TR2,TR1,s,m,t);  }//else  }//Msort  void MergeSort(SqList &L){  Msort(L.r, L.r, 1, L.length,L.length+1);  }//MergeSort |
|  |

四、实验结果

|  |
| --- |
| 第一次 |
| 选择排序：  排序前：49 38 65 49 76 13 27 52  排序后：13 27 38 49 49 52 65 76  插入排序：  排序前：49 38 65 49 76 13 27 52  排序后：13 27 38 49 49 52 65 76  冒泡排序：  排序前：49 38 65 49 76 13 27 52  排序后：13 27 38 49 49 52 65 76  快速排序：  排序前：49 38 65 49 76 13 27 52  排序后：13 27 38 49 49 52 65 76  归并排序：  排序前：49 38 65 49 76 13 27 52  排序后：13 27 38 49 49 52 65 76  --------------------------------  Process exited after 5.418 seconds with return value 0  请按任意键继续. . . |
| 第二次 |
| 选择排序：  排序前:35 35 37 49 63 30 81 70 54 46 15 70 71 12 65 20 27 86 44 22  排序后:12 15 20 22 27 30 35 35 37 44 46 49 54 63 65 70 70 71 81 86  插入排序：  排序前:35 35 37 49 63 30 81 70 54 46 15 70 71 12 65 20 27 86 44 22  排序后:12 15 20 22 27 30 35 35 37 44 46 49 54 63 65 70 70 71 81 86  冒泡排序：  排序前:35 35 37 49 63 30 81 70 54 46 15 70 71 12 65 20 27 86 44 22  排序后:12 15 20 22 27 30 35 35 37 44 46 49 54 63 65 70 70 71 81 86  快速排序：  排序前:35 35 37 49 63 30 81 70 54 46 15 70 71 12 65 20 27 86 44 22  排序后:12 15 20 22 27 30 35 35 37 44 46 49 54 63 65 70 70 71 81 86  归并排序：  排序前:35 35 37 49 63 30 81 70 54 46 15 70 71 12 65 20 27 86 44 22  排序后:12 15 20 22 27 30 35 35 37 44 46 49 54 63 65 70 70 71 81 86  --------------------------------  Process exited after 5.501 seconds with return value 0  请按任意键继续. . . |
| 第三次 |
| 选择排序：  排序前：25 32 98 50 35 86 61 27 92 28 67 75 96 11 19 85 94 85 92 76  排序后：11 19 25 27 28 32 35 50 61 67 75 76 85 85 86 92 92 94 96 98  插入排序：  排序前：25 32 98 50 35 86 61 27 92 28 67 75 96 11 19 85 94 85 92 76  排序后：11 19 25 27 28 32 35 50 61 67 75 76 85 85 86 92 92 94 96 98  冒泡排序：  排序前：25 32 98 50 35 86 61 27 92 28 67 75 96 11 19 85 94 85 92 76  排序后：11 19 25 27 28 32 35 50 61 67 75 76 85 85 86 92 92 94 96 98  快速排序：  排序前：25 32 98 50 35 86 61 27 92 28 67 75 96 11 19 85 94 85 92 76  排序后：11 19 25 27 28 32 35 50 61 67 75 76 85 85 86 92 92 94 96 98  归并排序：  排序前：25 32 98 50 35 86 61 27 92 28 67 75 96 11 19 85 94 85 92 76  排序后：11 19 25 27 28 32 35 50 61 67 75 76 85 85 86 92 92 94 96 98  --------------------------------  Process exited after 5.441 seconds with return value 0  请按任意键继续. . . |