山东大学 软件 学院

数据结构 课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：201400301034 | 姓名： 石兴帮 | | 班级： 软件八班 |
| 实验题目：排序算法 | | | |
| 实验学时：4h | | 实验日期： 2015-12-13 | |
| 实验目的：  掌握各种排序方法的实现思想 | | | |
| 硬件环境：  MacBook Pro  OS X Yosemite 10.10.3 | | | |
| 软件环境：  Xcode 6.4 | | | |
| 实验内容与设计：   1. 实验内容（题目内容，输入要求，输出要求）   1.创建排序类  2.提供操作：选择排序，冒泡排序，插入排序，基数排序，快速排序，归并排序。  3.能够显示各种排序算法的中间过程  2.数据结构与算法描述 （整体思路描述，所需要的数据结构与算法）  1.选择排序：遍历每个元素，找到它后面最小的元素，换位。  2.冒泡排序：遍历每个位置，与它相邻位置比，需要则换位。  3.插入排序：假定要插入的是已经排序好了的数组，寻找位置，找到后，它后面元素后移，然后插入。  4.基数排序：为所有可能出现的情况创建箱子，根据循环次数，每次将元素放入相应箱子数组，再依次提出。  5.快速排序：每次声明一个基准元素，每次操作后，使得基准元素前元素都比他小，基准元素后元素都比它大，然后对由基准元素拆分的两个数组分别继续快排。  6.归并排序：不断二分，将数组拆成若干个排序好了的小数组，然后二路归并。  3.测试结果（测试输入，测试输出）    0 2 3 5 6 7 8 9 11 42  4.实现源代码（程序风格清晰易理解，有充分的注释）  #include<iostream>  #define GET\_ARRAY\_LEN(array,len){len = (sizeof(array) / sizeof(array[0]));}  using namespace std;  int len;  int alen=1; //归并前数组a的大小  int blen=1; //归并前数组b的大小  void xuanze(int shuzu[])  {  int min;  int pos;  for (int i=0; i<len; i++) {  min = shuzu[i];  pos = i;  for (int j=i; j<len; j++) {  if (min > shuzu[j]) {  min = shuzu[j];  pos = j;  }  }  int temp = shuzu[i];  shuzu[i] = min;  shuzu[pos] = temp;  }  }  void bubble(int shuzu[])  {  int temp = 0;  for (int i = len - 1; i > 0; --i)  {  for (int j = 0; j < i; ++j)  {  if (shuzu[j + 1] < shuzu[j])  {  temp = shuzu[j];  shuzu[j] = shuzu[j + 1];  shuzu[j + 1] = temp;  }  }  }  }  void charu(int shuzu[])  {  // for (int i = 1; i < len; i++)  // {  // if (shuzu[i - 1] > shuzu[i])  // {  // int temp = shuzu[i];  // int j = i;  // while (j > 0 && shuzu[j - 1] > temp)  // {  // shuzu[j] = shuzu[j - 1];  // j--;  // }  // shuzu[j] = temp;  // }  // }    for (int i = 1; i<len; i++) {  if (shuzu[i] < shuzu[i-1])  {  int max = shuzu[i-1];  int pos = i-1;  for (int j=0; j<i ; j++) {  if (shuzu[j] > shuzu[i]) {  max = shuzu[j];  pos = j;  break;  }  }  int temp = shuzu[i];  for (int k=i; k>pos ; k--) {  shuzu[k] = shuzu[k-1];  }  shuzu[pos] = temp;  }  }  }  int maxbit(int data[], int n) //辅助函数，求数据的最大位数  {  int d = 1; //保存最大的位数  int p = 10;  for(int i = 0; i < n; ++i)  {  while(data[i] >= p)  {  p \*= 10;  ++d;  }  }  return d;  }  void binsort(int shuzu[])  {  int d = maxbit(shuzu, len);  int \*shuzu2 = shuzu;  int cishu = 1;  for (int i=1; i<=d ; i++) {  int yushu[10];  int temp[10][len];  int count=0;  for (int q=0; q<10; q++) {  yushu[q] = 0;  }  for (int p=0; p<len; p++) {  int yu = shuzu[p]/cishu%10;  temp[yu][yushu[yu]] = shuzu[p];  cout << yu << " " << yushu[yu] << endl;  yushu[yu]++;  }  for (int w=0; w<10; w++) {  cout << yushu[w] << endl;  for (int r=0; r<yushu[w]; r++) {  shuzu2[count++] = temp[w][r];  }  }  cishu \*=10;  shuzu = shuzu2;  }  }  void kuaisu(int shuzu[], int low, int high)  {  if(low >= high)  {  return;  }  int first = low;  int last = high;  int key = shuzu[first];        while(first < last)  {  while(first < last && shuzu[last] >= key)  {  --last;  }    shuzu[first] = shuzu[last];/\*将比第一个小的移到低端\*/    while(first < last && shuzu[first] <= key)  {  ++first;  }    shuzu[last] = shuzu[first];  /\*将比第一个大的移到高端\*/  }  shuzu[first] = key;/\*枢轴记录到位\*/  kuaisu(shuzu, low, first-1);  kuaisu(shuzu, first+1, high);  }  void merge(int a[], int b[],int \*c)  {  int d[alen+blen];  c = d;  int i=0;  int j=0;  for (int count = 0; count < alen+blen; count++) {  if (i==alen) {  cout << "a数组全加进去了" <<endl;  c[count] = b[j];  j++;  }  else if (j==blen) {  cout << "b数组全加进去了" <<endl;    c[count] = a[i];  cout << c[count] << endl;  i++;  }  else if (a[i] <= b[j]) {  c[count] = a[i];  cout << c[count] << endl;  i++;  }  else if (a[i] > b[j]) {  c[count] = b[j];  cout << c[count] << endl;    j++;  }  }  }  void Merge( int\* arr, int low, int mid, int high )  {  int i, j, k;  i = low;  j = mid + 1;  k = 0;  int\* tmp = new int[high - low + 1];    while( i <= mid && j <= high )  {  if ( \*(arr + i) <= \*(arr + j) )  \*(tmp + k++) = \*(arr + i++);  else  \*(tmp + k++) = \*(arr + j++);  }    while( i <= mid )  \*(tmp + k++) = \*(arr + i++);  while( j <= high )  \*(tmp + k++) = \*(arr + j++);    for( i=low, k=0; i<=high; i++, k++ )  {  \*(arr + i) = \*(tmp + k);  }    delete [] tmp;  }  void MergeSort( int\* arr, int low, int high )  {  int mid;  if ( low < high )  {  mid = ( low + high ) / 2;  MergeSort( arr, low, mid );  MergeSort( arr, mid+1, high );  Merge( arr, low, mid, high );  }  }  int main()  {  int a[10] = {7,3,9,6,2,8,11,42,5,0};  // int a[4] = {2,1,3,6};  len = sizeof(a )/sizeof(a[0]);      // xuanze(a);  // bubble(a );  // charu(a );  // binsort(a );  // kuaisu(a , 0, len-1);  MergeSort(a , 0 , len-1);  for (int i=0; i<len; i++) {  cout << a[i] << " ";  }  cout << endl;    return 0;  } | | | |
| 结论分析与体会：  学会了多种排序算法。 | | | |