|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm and Data Structure Experimental Assignment | | | |
| Class | 17 CS 01-02 | Teacher Name | Mano and Hong Zhang |
| Student Name | 张钧豪 | Submitted Time | 20191108 |
| Assignment Topic | Implement three sorting algorithms | | |
| Assignment Requirement:  Sorting --Practical Exercises  1、 Reading the data from the text file input1-data.txt, and using the quick  sort to sort the data, then write the ordered data in text file output1-  data.txt.  **For example:**  **Input1-data.txt is**  **23, 34, 15, 19, 56, 57, 9**  **Output1-data.txt is:**  **9, 15, 19, 23, 34, 56, 57**  2、 Reading the data from the text file input2-data.txt, and using the merge  sort to sort the data, then write the ordered data in text file output2-  data.txt.  3、 Reading the data from the text file input-data3.txt, and using the heap  sort to sort the data, then write the ordered data in text file outputdata3.  txt. | | | |
| Code of Main Functions:  **Quick sort :**  #include "stdafx.h"  #include<stdio.h>  #include<string>  #define MAX\_LINE 1024  #define MAX\_NUM 1024  int read(int numbers[])  {  int number[MAX\_LINE] = { 0 }, i = 0, j = 1;  for (i = 0; i < MAX\_LINE; i++) { //初?始º?化¡¥  number[i] = MAX\_NUM;  }  FILE \*fp;  if((fp = fopen("G:\\a大ä¨®三¨y上¦?\\算?法¤¡§分¤?析?与®?设¦¨¨计?\\算?法¤¡§实º¦Ì验¨¦\\实º¦Ì验¨¦一°?\\input1-data.txt","r")) == NULL){  printf("Open file failed\n");  }  i = 0;  printf("Before QuickSorting: ");  while (fscanf(fp, "%d", &number[i])!=EOF) { // file writ in arrary number[] first  printf("%d ", number[i]);  i++;  }  for (i = 0; i < MAX\_LINE; i++) {  if (number[i] == MAX\_NUM)  break;  numbers[j] = number[i];  ++j; // The size of numbers[]  }  //for (i = 0; i < j; i++)  //printf("\*%d ", numbers[i]);  //printf("\n");  if (fclose(fp) != 0)  printf("file can't close\n");  return j;  }  void write(int numbers[], int size) // size: the size of numbers[]  {  int i;  FILE \*fp;  if((fp = fopen("G:\\a大ä¨®三¨y上¦?\\算?法¤¡§分¤?析?与®?设¦¨¨计?\\算?法¤¡§实º¦Ì验¨¦\\实º¦Ì验¨¦一°?\\output1-data.txt","w")) == NULL){  printf("Open file failed\n");  }  for(i=1;i<size;i++){  fprintf(fp, "%d ", numbers[i]);  }  if (fclose(fp) != 0)  printf("file can't close\n");  }  int partition(int number[8], int low, int high) {  number[0] = number[low];  int pkey = number[0];  while (low < high) {  while (low < high && number[high] >= pkey) high--;  number[low] = number[high];  while (low < high && number[low] <= pkey) low++;  number[high] = number[low];  }  number[low] = number[0];  //printf("partition done \n");  return low; //返¤¦Ì回?位?置?  }  void qsort(int number[8], int low, int high) { //递ÌY归¨¦进?行D排?序¨°  if (low < high) {  int pkey = partition(number, low, high);  //for (int i = 0; i < 8; i++) {  //printf("d ");  //printf("%d \n", number[i]);  //}  qsort(number, low, pkey - 1);  qsort(number, pkey + 1, high);  }  //printf("qsort done \n");  }  void QuickSort(int number[], int high) {  qsort(number, 1, high);  }  int main(int argc, \_TCHAR\* argv[])  {  int numbers[MAX\_LINE] = { 0 }, high, i;  //FILE\* fp;  //\*fp = read();  high = read(numbers) -1;  printf("high: %d \n\n\n",high);  QuickSort(numbers, high);  write(numbers,high+1);  printf("After QuickSorting: ");  for (i = 1; i < high+1; i++) {  printf("%d ", numbers[i]);  }  return 0;  }  **Merge sort:**  #include<stdio.h>  #define MAX\_LINE 1024  #define MAX\_NUM 1024  int read(int numbers[]) // read data in file, write data in number's' []  {  int number[MAX\_LINE] = { 0 }, i = 0, j = 1;  for (i = 0; i < MAX\_LINE; i++) { //初始化  number[i] = MAX\_NUM;  }  FILE \*fp;  if((fp = fopen("G:\\a大三上\\算法分析与设计\\算法实验\\实验一\\input2-data.txt","r")) == NULL){  printf("Open file failed\n");  }  i = 0;  printf("Before QuickSorting: ");  while (fscanf(fp, "%d", &number[i])!=EOF) { // file writ in the arrary number[] first  printf("%d ", number[i]);  i++; // the location of last  }  for (i = 0; i < MAX\_LINE; i++) {  if (number[i] == MAX\_NUM)  break;  numbers[j] = number[i];  ++j; // j initlize j = 1. So j is The size of numbers[]  //  }  //for (i = 0; i < j; i++)  //printf("\*%d ", numbers[i]);  //printf("\n");  if (fclose(fp) != 0)  printf("file can't close\n");  return j;  }  void write(int numbers[], int size) // size: the size of numbers[]  {  int i;  FILE \*fp;  if((fp = fopen("G:\\a大三上\\算法分析与设计\\算法实验\\实验一\\output2-data.txt","w")) == NULL){  printf("Open file failed\n");  }  for(i=1;i<size;i++){  fprintf(fp, "%d ", numbers[i]);  }  if (fclose(fp) != 0)  printf("file can't close\n");  }  void merge(int SR[], int TR[], int i, int m, int n)  {    int j, k;  for(k=i,j=m+1;i<=m&&j<=n;k++){ // merge SR[i..m] SR[m+1..n] to TR[i..n]  if(SR[i]<=SR[j])  TR[k] = SR[i++];  else  TR[k] = SR[j++];  }  /\* i or j will always small than m or n \*/  for( ;i<=m;i++)  TR[k++] = SR[i];  for( ;j<=n;j++)  TR[k++] = SR[j];  }  void msort(int SR[], int TR1[], int s, int t)  {  int m, TR2[MAX\_LINE];  if(s==t) TR1[s] = SR[s];  else{  m = (s+t)/2;  msort(SR, TR2, s, m); //SR左边有序  msort(SR, TR2, m+1, t); //SR右边有序  merge(TR2, TR1, s, m, t);  }  }  int main()  {  int i, numbers[MAX\_LINE] = {0}, high; //最后一个数组元素下标  high = read(numbers) - 1;  msort(numbers,numbers,1,high);  write(numbers, high);  printf("\n\n");  printf("After MergeSorting :");  for(i=1; i < high+1; i++)  printf("%d ",numbers[i]);  return 0;  }  **Heap sort:**  #include<stdio.h>  #define MAX\_LINE 1024  #define MAX\_NUM 1024  int read(int numbers[]) // read data in file, write data in number's' []  {  int number[MAX\_LINE] = { 0 }, i = 0, j = 1;  for (i = 0; i < MAX\_LINE; i++) { //初始化  number[i] = MAX\_NUM;  }  FILE \*fp;  if((fp = fopen("G:\\a大三上\\算法分析与设计\\算法实验\\实验一\\input3-data.txt","r")) == NULL){  printf("Open file failed\n");  }  i = 0;  printf("Before HeapSorting: ");  while (fscanf(fp, "%d", &number[i])!=EOF) { // file writ in the arrary number[] first  printf("%d ", number[i]);  i++; // the location of last  }  for (i = 0; i < MAX\_LINE; i++) {  if (number[i] == MAX\_NUM)  break;  numbers[j] = number[i];  ++j; // j initlize j = 1. So j is The size of numbers[]  //  }  //for (i = 0; i < j; i++)  //printf("\*%d ", numbers[i]);  //printf("\n");  if (fclose(fp) != 0)  printf("file can't close\n");  return j;  }  void write(int numbers[], int size) // size: the size of numbers[]  {  int i;  FILE \*fp;  if((fp = fopen("G:\\a大三上\\算法分析与设计\\算法实验\\实验一\\output3-data.txt","w")) == NULL){  printf("Open file failed\n");  }  for(i=1;i<size;i++){  fprintf(fp, "%d ", numbers[i]);  }  if (fclose(fp) != 0)  printf("file can't close\n");  }  void swap(int numbers[], int i, int j)  {  int temp;  temp = numbers[i];  numbers[i] = numbers[j];  numbers[j] = temp;  }  void HeapAdjust(int numbers[], int s, int n)  {  int i, temp;  temp = numbers[s];  for(i=2\*s; i<=n; i\*=2){  /\* i永远指向最大的子节点 \*/  if(i<n && numbers[i] < numbers[i+1]) i++; //右子树大于左子节点 i指向右子节点  if(temp >= numbers[i]) break; //双亲已经大于了最大子节点  numbers[s] = numbers[i];  s = i;  }  numbers[s] = temp;  }  void Heapsort(int numbers[], int n)  {  int i;  for(i=n/2; i>0; i--){ //从下至上，从右到左构建大顶堆  HeapAdjust(numbers, i, n);  }    for (i=n; i>1; i--) //调整 第一个元素和最后一个元素互换  {  swap(numbers, 1, i); //第一个元素和最后一个元素互换  HeapAdjust(numbers, 1, i-1); //调整完重新构建  }  }  int main()  {  int i, numbers[MAX\_LINE] = {-1,49,38,65,97,76,13,27}, high; // high is down flag  high = read(numbers) - 1;  Heapsort(numbers,high);  write(numbers, high+1);  printf("\n\nAfter HeapSorting :");  for(i=1;i<high+1;i++)  printf("%d ",numbers[i]);  printf("\n");  /\*int i, numbers[8] = {-1,49,38,65,97,76,13,27};  printf("Before sorting :") ;  for(i=1;i<9;i++)  printf("%d ",numbers[i]);  printf("\n");  Heapsort(numbers,7);  printf("After sorting :") ;  for(i=1;i<9;i++)  printf("%d ",numbers[i]);  printf("\n");\*/  return 0;  } | | | |
| Experimental Results:  **Quick sort :**    **Merge sort:**    **Heap sort:** | | | |