****

**算法分析与设计实验报告**

****

**学 院： 电子信息与人工智能学院**

**专业名称： 计算机科学与技术**

**班 级： 计算机203**

**学 号： 202007020625**

**姓 名： 於俊涛**

**任课老师： 齐 勇**

**排序算法性能分析实验报告**

班级 计算机203 学号 202007020625 姓名 於俊涛 角色 coder

**一、实验目的**

1、掌握8大排序算法的原理。

2、分析不同算法的性能及适用环境。

**二、实验环境**

线上平台sustoj.com

**三、实验内容**

选择排序、插入排序、冒泡排序、快速排序、堆排序、基数排序、桶排序，归并排序。  
 1.选择排序

#include <stdio.h>  
void mysort(int array[],int n)  
{int i,j,k,t;  
for(i=0;i<n-1;i++)  
{k=i;  
for(j=i+1;j<n;j++)  
if(array[j]<array[k])  
k=j;  
t=array[k];array[k]=array[i];array[i]=t;  
}}     //设计排序函数  
int main()  
{  
int x[1000];  
int i,n,j,m,t;  
scanf("%d",&n);  
for(i=0;i<n;i++)  
scanf("%d",&x[i]);  
mysort(x,n);   //调用排序函数  
for(i=0;i<n-1;i++)  
printf("%d ",x[i]);  
printf("%d\n",x[n-1]);  
    return 0;  
}

2.插入排序

#include <stdio.h>

 void mysort(int\* a, int n)   
{  
for (int i = 0; i < n - 1; i++)  
{  
int end = i;  
int tmp = a[end + 1];  
while (end >= 0)  
{  
if (a[end] > tmp)  
{  
a[end + 1] = a[end];  
end--;  
}  
else  
{  
break;  
}  
}  
a[end + 1] = tmp;  
}  
} //设计排序函数  
int main()  
{  
int x[1000];  
int i,n,j,m,t;  
scanf("%d",&n);  
for(i=0;i<n;i++)  
scanf("%d",&x[i]);  
mysort(x,n);   //调用排序函数  
for(i=0;i<n-1;i++)  
printf("%d ",x[i]);  
printf("%d\n",x[n-1]);  
    return 0;  
}

3.冒泡排序

#include <stdio.h>

void mysort(int\* a, int n)  
{  
for (int j = 1; j <= n; j++)  
{  
int flag = 1;  
for (int i = 0; i < n - j; i++)  
{  
if (a[i + 1] < a[i])  
{  
Swap(&a[i + 1], &a[i]);  
flag = 0;  
}}  
if (flag)  
break;  
}  
} //设计排序函数  
int main()  
{  
int x[1000];  
int i,n,j,m,t;  
scanf("%d",&n);  
for(i=0;i<n;i++)  
scanf("%d",&x[i]);  
mysort(x,n);   //调用排序函数  
for(i=0;i<n-1;i++)  
printf("%d ",x[i]);  
printf("%d\n",x[n-1]);  
    return 0;  
}

4.快速排序

#include <stdio.h>

 void mysort(int\* a, int left, int right)  
{  
if (left >= right)  
return;  
int key = PartSort1(a, left, right);  
QuickSort(a, left, key - 1);  
QuickSort(a, key + 1, right);  
}

int PartSort2(int\* a, int left, int right)  
{  
int mid = GetMidIndex(a, left, right);  
Swap(&a[left], &a[mid]);  
int key = a[left];  
int hole = left;  
while (left < right)  
{  
while (left < right && a[right] >= key)  
{  
right--;  
}  
a[hole] = a[right];  
hole = right;  
while (left < right && a[left] <= key)  
{  
left++;  
}  
a[hole] = a[left];  
hole = left;  
}  
a[hole] = key;  
return hole;  
} //设计排序函数  
int main()  
{  
int x[1000];  
int i,n,j,m,t;  
scanf("%d",&n);  
for(i=0;i<n;i++)  
scanf("%d",&x[i]);  
mysort(x,0,n-1);   //调用排序函数  
for(i=0;i<n-1;i++)  
printf("%d ",x[i]);  
printf("%d\n",x[n-1]);  
    return 0;  
}

5.堆排序

import java.util.Scanner;  
public class Main {  
    public static void adjustHeap (int[] arr,int i,int length){  
        /\*取出当前非叶子结点的值保到临时变量中\*/  
        int temp = arr[i];  
        /\*j=i\*2+1表示的是i结点的左子结点\*/  
        for (int j = i \* 2 + 1; j < length ; j = j \* 2 + 1) {  
            if (j+1 < length && arr[j] < arr[j+1]){ //左子结点小于右子结点  
                j++; //j指向右子结点  
            }  
            if (arr[j] > temp){ //子节点大于父节点  
                arr[i] = arr[j]; //把较大的值赋值给父节点  
                //arr[j] = temp; 这里没必要换  
                i = j; //让i指向与其换位的子结点 因为  
            }else{  
                /\*子树已经是大顶堆了\*/  
                break;  
            }  
        }  
        arr[i] = temp;  
    }  
    public static void heapSort(int[] arr){  
        //为什么从arr.length/2-1开始？  
        for (int i = arr.length/2-1; i >= 0 ; i--) {  
            adjustHeap(arr,i,arr.length);  
        }  
        for (int j = arr.length-1; j > 0; j--) {  
            int temp = arr[j];  
            arr[j] = arr[0];  
            arr[0] = temp;  
            adjustHeap(arr,0,j);  
        }  
    }  
    public static void main(String[] args) {  
        Scanner sc=new Scanner(System.in);  
        int n=sc.nextInt();  
        int[] arr=new int[n];  
        for(int i=0;i<arr.length;i++){  
            arr[i]=sc.nextInt();  
        }  
        heapSort(arr);  
        for(int i=0;i< arr.length;i++){  
            System.out.print(arr[i]+" ");  
        }  
    }  
}

6.基数排序

import java.util.Scanner;  
public class Main {  
    public static void radixSort(int[] arr) {  
        //得到数组中的最大数是几位数。  
        int max = arr[0];//假设第一位是最大数  
        for (int i = 0; i < arr.length; i++) {  
            if (arr[i] > max) {  
                max = arr[i];  
            }  
        }  
        //最大数的长度  
        int maxLength = (max + "").length();  
        //首先定义一个二维数组，10行，arr.length列，10行表示的是有10个桶，arr.length表示的是桶中的数据  
        //考虑到一个桶中可能存放的全部是一个数组中的值。  
        int[][] bucket = new int[10][arr.length];  
        //定义一个辅助数组，用于计算各个桶中的数据的个数  
        int[] bucketEleCount = new int[10];  
        for (int k = 0, n = 1; k < maxLength; k++, n \*= 10) {  
            //1，第一轮排序，将数组中的个位数进行处理  
            for (int i = 0; i < arr.length; i++) {  
                int rest = arr[i] / n % 10;//计算出各个数据的个位数  
                //将数组中数据放入个位数对应的桶中  
                bucket[rest][bucketEleCount[rest]] = arr[i];  
                bucketEleCount[rest]++;  
            }  
            int index = 0;//定义一个辅助索引值，数组的索引  
            //将放入二维数组中的数据依次取出，并放入原数组  
            for (int i = 0; i < bucketEleCount.length; i++) {  
                //遍历用于记录放入桶中数据数组，只要数组中的值不为0，就将桶中数据取出  
                if (bucketEleCount[i] != 0) {  
                    //取出数据  
                    for (int j = 0; j < bucketEleCount[i]; j++) {  
                        //将桶中的数据放入原数组  
                        arr[index++] = bucket[i][j];  
                    }  
                }  
                bucketEleCount[i] = 0;  
            }  
        }  
    }  
    public static void main(String[] args) {  
        Scanner sc=new Scanner(System.in);  
        int n=sc.nextInt();  
        int[] arr=new int[n];  
        for(int i=0;i<arr.length;i++){  
            arr[i]=sc.nextInt();  
        }  
        radixSort(arr);  
        for(int i=0;i< arr.length;i++){  
            System.out.print(arr[i]+" ");  
        }  
    }  
}

7.桶排序

import java.util.\*;  
public class Main{  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int len = sc.nextInt();  
        int[] nums = new int[len];  
        //数据输入  
        for (int i = 0; i < len; i++) {  
            nums[i] = sc.nextInt();  
        }  
        bucketSort(nums);  
        //输出  
        for (int n : nums) {  
            System.out.print(n + " ");  
        }  
    }  
    private static void bucketSort(int[] array){  
        int min = Integer.MAX\_VALUE;  
        int max = Integer.MIN\_VALUE;  
        //找出数组中的最大值和最小值并记录下来  
        for (int i:array) {  
            min = Math.min(min,i);  
            max = Math.max(max,i);  
        }  
//辅助数组的大小为max-min+1  
        int[] bucket = new int[max - min+1];  
        for (int i=0;i<array.length;i++){  
         //减去min是为了防止array[i]小于0，数组索引可是不能小于0的，相当于做了个偏移。++是为了记录次数  
            bucket[array[i] - min] ++;  
        }  
        int j = 0;  
        //遍历辅助数组，取出结果  
        for (int i=0;i<bucket.length;i++){  
            while (bucket[i]-- > 0){  
                array[j++] = i+min;  
            }  
        }  
    }  
}

8.归并排序

    #include<stdio.h>  
      void merge(int a[], int low, int mid, int high){  
          int i, j, k, s[high];  
          for(i = low;i <= high;i++)  
              s[i] = a[i];//将数组a复制到数组s  
          for(i = low, j = mid + 1, k = i;i <= mid && j <= high;k++){//判定条件为，数组结束。  
              if(s[i] <= s[j]) a[k] = s[i++];//比较两数组中的大小，小的进a中  
              else a[k] = s[j++];  
          }  
          while(i <= mid) a[k++] = s[i++];  
          while(j <= high) a[k++] = s[j++];  
      }  
      void mergesort(int a[], int low, int high){  
          if(low < high){//只要能继续分组，那么就继续分  
              int mid = (low + high) / 2;  
              mergesort(a, low, mid);//分为左组，并进行归并排序  
              mergesort(a, mid + 1, high);//分为右组，并进行归并排序  
              merge(a, low, mid, high);  
          }  
      }  
      int main(){  
          int n;  
          scanf("%d", &n);  
          int a[n];  
          for(int i = 0;i < n;i++)  
              scanf("%d", &a[i]);  
          mergesort(a, 0, n - 1);  
          for(int i = 0;i < n;i++){   
            printf("%d%c", a[i], i == n - 1?&apos;\n&apos;:&apos; &apos;);  
        }   
    }

**四、实验总结**

通过本次实验，我对八大排序有了更加深刻的认识。通过基础知识的学习及与小组成员的沟通中，我学习了不同排序算法的原理及各自的时间复杂度，了解了不同算法的使用场景。作为coder，本次实验对我编程能力的提高有很大帮助。