**问题描述：**

给定数组a1, a2,  a3, ... an,要求编写函数f输出改数组的最大值和最小值。假设数组中的值两两不相同。

**思路：**

朴素的算法可以通过遍历的算法，通过2n次的比较得到数组中的最大值和最小值。实现代码如下:

复制代码

  public class Pair  
    {  
        // 构造函数  
        public Pair(int max, int min)  
        {  
            this.\_max = max;  
            this.\_min = min;  
        }  
  
        // 属性  
        private int \_max;  
        public int Max  
        {  
            get   
            {  
                return this.\_max;  
            }  
            set  
            {  
                this.\_max = value;  
            }  
        }  
  
        private int \_min;  
        public int Min  
        {  
            get  
            {  
                return this.\_min;  
            }  
            set  
            {  
                this.\_min = value;  
            }  
        }  
    }  
  
    public class SearchMinAndMax  
    {  
        private int[] arr =  {2, 3, 1, 5, 6,  9, 7, 8};  
  
        public Pair function1()   
        {  
            // 初始化  
            int max, min;  
            if (this.arr[0] < this.arr[1])  
            {  
                max = this.arr[1];  
                min = this.arr[0];  
            }  
            else  
            {  
                max = this.arr[0];  
                min = this.arr[1];  
            }  
  
            // 开始循环查找  
            for (int i = 2; i < this.arr.Length; ++i )  
            {  
                // 比min还小  
                if(this.arr[i] < min)  
                {  
                    min = this.arr[i];  
                }  
  
                 // 比max还大  
                if(this.arr[i] > max)  
                {  
                    max = this.arr[i];  
                }  
  
            }  
  
            // 循环结束，返回结果  
            return new Pair(max, min);  
        }  
    }

复制代码

下面的过程将是不断修正上面算法的过程。考虑这个算法：将数组逻辑上分为两个两个为一组，然后比较每组中两个值的大小，如果比max还大，修改max的值，同理对min，通过这种算法的修正的话，需要的比较次数如下：1.5n。改算法实现起来主要是需要判定数组长度的奇偶性。代码如下：

复制代码

public Pair function2()  
        {  
            // 这里使用的分组的方式   
  
            // 计算分组数   
            int n = this.arr.Length / 2;    // 取整运算  
  
            // 使用第一个分组初始化max和min  
            int max, min;  
            if (this.arr[0] < this.arr[1])  
            {  
                max = this.arr[1];  
                min = this.arr[0];  
            }  
            else  
            {  
                max = this.arr[0];  
                min = this.arr[1];  
            }  
  
            // 遍历分组数  
            int tmpMax = int.MaxValue, tmpMin = int.MinValue;  
            for (int i = 2; i <= n; ++i)  
            {  
                // 得到每个分组中的最大值和最小值，一次比较  
                if(this.arr[2 \* i - 2] > this.arr[2 \* i - 1])  
                {  
                    tmpMax = this.arr[2 \* i - 2];  
                    tmpMin = this.arr[2 \* i - 1];  
                }  
                else  
                 {  
                     tmpMin = this.arr[2 \* i - 2];  
                     tmpMax = this.arr[2 \* i - 1];  
                 }  
  
                // 更新整个数组中的最大值和最小值  
  
                // 第二次比较  
                if (tmpMax > max)  
                {  
                    max = tmpMax;  
                }  
  
                // 第三次比较  
                if(tmpMin < min)  
                {  
                    min = tmpMin;  
                }  
  
            }  
  
            // 是否存在剩下的元素  
            if (2 \* n < this.arr.Length)  
            {  
                // 还剩下一个元素  
                if (this.arr[2\* n] > max)  
                {  
                    max = this.arr[2 \* n];  
                }  
  
                if(this.arr[2 \* n] < min)  
                {  
                    min = this.arr[2 \* n];  
                }  
            }  
  
            // 返回值  
            return new Pair(max, min);  
        }

复制代码

下面继续优化上面的算法。使用分治思想。实现代码：

复制代码

 // begin, end实现闭区间  
        public Pair function3(int begin, int end)  
        {  
            // 递归的出口条件  
            if(begin + 1  >= end)  
            {  
                // 如果是奇数个元素，这个可能存在是同一个元素  
                if (this.arr[begin] >= this.arr[end])  
                    return new Pair(this.arr[begin], this.arr[end]);  
                else  
                    return new Pair(this.arr[end], this.arr[begin]);  
            }  
  
            // 没有达到出口条件，开始递归  
            int middle = (end + begin + 1) / 2;   
            Pair left = this.function3(begin, middle - 1);  
            Pair right = this.function3(middle, end);  
  
            // 返回值  
            int max = int.MinValue, min = int.MinValue;  
            if(left.Max >right.Max)  
            {  
                max = left.Max;  
            }  
            else  
            {  
                max = right.Max;  
            }  
  
            if(left.Min < right.Min)  
            {  
                min = left.Min;  
            }  
            else  
            {  
                min = right.Min;  
            }  
  
            return new Pair(max, min);  
        }  
  
        public Pair function3Wrapper()  
        {  
            return this.function3(0, this.arr.Length - 1);  
        }

复制代码