**《程序设计课程实践》设计文档**

# 作业题目：排序算法

**学号：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_19151633\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_\_\_\_\_\_应宇杰\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

完成时间 2021 年 4 月 14 日

第\_\_4\_\_\_次作业 (写上第几次作业)

题目\_\_\_排序算法\_\_\_\_\_（写上题目号）

提交代码：

//golang语言

package main

import (

"fmt"

"math/rand"

"time"

)

func main() {

Begin()

}

func Begin() {

var n int

var nums []int

fmt.Printf("要输入多少个随机数: ")

fmt.Scan(&n)

for i := 0; i < n; i++ {

nums = append(nums, rand.Intn(10000))

}

nums1 := nums

nums2 := nums

nums3 := nums

start := time.Now()

QuickSort(nums1, 0, len(nums1)-1)

fmt.Println("the time of quickSort", time.Now().Sub(start))

start = time.Now()

BigTopReactor(nums2)

fmt.Println("the time of heapSort", time.Now().Sub(start))

start = time.Now()

SelectionSort(nums3)

fmt.Println("the time of selectionSort", time.Now().Sub(start))

fmt.Println("check quicksort: ", check(nums1))

fmt.Println("check heapSort: ", check(nums1))

fmt.Println("check selectionSort: ", check(nums1))

}

//check the array is increase progressively

func check(nums []int) bool {

for i := 1; i < len(nums)-1; i++ {

if nums[i] < nums[i-1] {

return false

}

}

return true

}

func QuickSort(nums []int, a, b int) {

if a >= b {

return

}

i, j := a, b

flag := i

for i < j {

for ; j >= i; j-- {

if nums[j] < nums[flag] {

nums[j], nums[flag] = nums[flag], nums[j]

flag = j

break

}

}

for ; i < j; i++ {

if nums[i] > nums[flag] {

nums[i], nums[flag] = nums[flag], nums[i]

flag = i

break

}

}

}

QuickSort(nums, a, flag-1)

QuickSort(nums, flag+1, b)

}

func BigTopReactor(nums []int) {

heapSize := len(nums)

start := heapSize / 2

for i := start; i >= 0; i-- {

MaxHeap(nums, i, heapSize)

}

for heapSize > 0 {

nums[heapSize-1], nums[0] = nums[0], nums[heapSize-1]

heapSize--

MaxHeap(nums, 0, heapSize)

}

}

func MaxHeap(nums []int, start, size int) {

l, r, largest := start\*2+1, start\*2+2, start

if l < size && nums[l] > nums[largest] {

largest = l

}

if r < size && nums[r] > nums[largest] {

largest = r

}

if start != largest {

nums[start], nums[largest] = nums[largest], nums[start]

MaxHeap(nums, largest, size)

}

}

//选择排序

func SelectionSort(nums []int) {

var min int

for i := 0; i < len(nums)-2; i++ {

min = i

for j := i + 1; j < len(nums)-1; j++ {

if nums[min] > nums[j] {

min = j

}

}

nums[i], nums[min] = nums[min], nums[i]

}

}

运行结果：

（可以截图）



