Algorithm

Shuwen He
1201220707@pku.edu.cn
Written in Peking University Library

2020 年 12 月 20 日

目录

Chapter
Algorithm

1.1 Book

discrete mathematics

Abstract algebra

Ordinary differential equation

Mathematical analysis

Probability statistics

Computer software and theory

English

1.2 Overview

In this book I define algorithms as a branch of mathematics

Algorithm must be designed to solve practical problems. An algorithm that cannot solve practical problems is a vase.

Divide the algorithm in a way that solves the problem

Please believe that learning algorithms have methods

Fresh and refined, immediate, don't be sloppy

I proved the algorithmic mathematical kilometers system

Chapter
Logarithm

2.1 lgn

$lgn = log_2 n$ (Base 2 logarithm)

$lnn = log_e n$ (Natural logarithm)

$lg^k n = (lgn)^k$ (Exponentiation)

$lglgn = lg(lgn)$ (complex)

Chapter
Stack

stack : push + pop

Chapter
Queue

queue : enqueue + dequeue

Chapter
list

double linked list: value + next and prev

5.1 list

```
package main
import (
)
type ListNode struct{
    Val int
    Next *ListNode
}
func main() {
    headNode := &ListNode{}
    listData := headNode
    insertTail(1,listData,headNode)
    PrintList(listData)
    insertTail(2,listData,headNode)
    PrintList(listData)
    insertTail(3,listData,headNode)
    PrintList(listData)
}
func insertTail(value
        int,list,position *ListNode) {
    newCell := new(ListNode)
```

```
        if tempCell == nil{
            fmt.Println("out of space")
        }

        tempCell.Val = value
        tempCell.Next = position.Next
        position.Next = tempCell
    }

    func PrintList(list *ListNode) {
        if list.Next != nil{
            fmt.Print(list.Val,"->")
            PrintList(list.Next)
        }else{
            fmt.Println(list.Val)
        }
    }
```

//red!50!green!50!blue!50 red!50!green!50!blue!50给red!50!green!50!blue!50出red!50!green!50!blue!50两red!50!green!50!blue!50个red!50!green!50!blue!50非red!50!green!50!blue!50空red!50!green!50!blue!50的red!50!green!50!blue!50链red!50!green!50!blue!50表red!50!green!50!blue!50用red!50!green!50!blue!50来red!50!green!50!blue!50表red!50!green!50!blue!50示red!50!green!50!blue!50两red!50!green!50!blue!50个red!50!green!50!blue!50非red!50!green!50!blue!50负red!50!green!50!blue!50的red!50!green!50!blue!50整red!50!green!50!blue!50数。red!50!green!50!blue!50其red!50!green!50!blue!50中，red!50!green!50!blue!50它red!50!green!50!blue!50们red!50!green!50!blue!50各red!50!green!50!blue!50自red!50!green!50!blue!50的red!50!green!50!blue!50位red!50!green!50!blue!50数red!50!green!50!blue!50是red!50!green!50!blue!50按red!50!green!50!blue!50照red!50!green!50!blue!50逆red!50!green!50!blue!50序red!50!green!50!blue!50的red!50!green!50!blue!50方

red!20!green!20!blue!50!red!50!green!50!blue!50式red!50!green!50!blue!50存red!20!green!20!blue!50!red!50!green!50!blue!50储red!50!green!50!blue!50的，red!20!green!20!blue!50!red!50!green!50!blue!50//red!50!green!50!blue!50并red!50!green!50!blue!50且red!20!green!20!blue!50!red!50!green!50!blue!50它red!50!green!50!blue!50们red!20!green!20!blue!50!red!50!green!50!blue!50的red!50!green!50!blue!50每red!20!green!20!blue!50!red!50!green!50!blue!50个red!50!green!50!blue!50节red!20!green!20!blue!50!red!50!green!50!blue!50点red!50!green!50!blue!50只red!20!green!20!blue!50!red!50!green!50!blue!50能red!50!green!50!blue!50存red!20!green!20!blue!50!red!50!green!50!blue!50储red!50!green!50!blue!50一red!20!green!20!blue!50!red!50!green!50!blue!50位red!50!green!50!blue!50数red!20!green!20!blue!50!red!50!green!50!blue!50字。

red!20!green!20!blue!50!red!50!green!50!blue!50//red!50!green!50!blue!50如red!50!green!50!blue!50果，red!20!green!20!blue!50!red!50!green!50!blue!50我red!50!green!50!blue!50们red!20!green!20!blue!50!red!50!green!50!blue!50将red!50!green!50!blue!50这red!20!green!20!blue!50!red!50!green!50!blue!50两red!50!green!50!blue!50个red!20!green!20!blue!50!red!50!green!50!blue!50数red!50!green!50!blue!50相red!20!green!20!blue!50!red!50!green!50!blue!50加red!50!green!50!blue!50起red!20!green!20!blue!50!red!50!green!50!blue!50来，red!50!green!50!blue!50则red!20!green!20!blue!50!red!50!green!50!blue!50会red!50!green!50!blue!50返red!20!green!20!blue!50!red!50!green!50!blue!50回red!50!green!50!blue!50一red!20!green!20!blue!50!red!50!green!50!blue!50个red!50!green!50!blue!50新red!20!green!20!blue!50!red!50!green!50!blue!50的red!50!green!50!blue!50链red!20!green!20!blue!50!red!50!green!50!blue!50表red!50!green!50!blue!50来red!20!green!20!blue!50!red!50!green!50!blue!50表red!50!green!50!blue!50示red!20!green!20!blue!50!red!50!green!50!blue!50它red!50!green!50!blue!50们red!20!green!20!blue!50!red!50!green!50!blue!50的red!50!green!50!blue!50和。

red!20!green!20!blue!50!red!50!green!50!blue!50//red!50!green!50!blue!50您red!50!green!50!blue!50可red!20!green!20!blue!50!red!50!green!50!blue!50以red!50!green!50!blue!50假red!20!green!20!blue!50!red!50!green!50!blue!50设red!50!green!50!blue!50除red!20!green!20!blue!50!red!50!green!50!blue!50了red!50!green!50!blue!50数red!20!green!20!blue!50!red!50!green!50!blue!50字red!50!green!50!blue!500red!20!green!20!blue!50!red!50!green!50!blue!50之red!50!green!50!blue!50外，red!20!green!20!blue!50!red!50!green!50!blue!50这red!50!green!50!blue!50两red!20!green!20!blue!50!red!50!green!50!blue!50个red!50!green!50!blue!50数red!20!green!20!blue!50!red!50!green!50!blue!50都red!50!green!50!blue!50不red!20!green!20

```
red!20!green!20!blue!20red!50!green!50!blue!50会red!50!green!50!blue!50以                                    red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!500red!50!green!50!blue!50开                                    red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50头。                                                          red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50        red!50!green!50!blue!50//red!50!green!50!blue!50      red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50示red!50!green!50!blue!50例：                                  red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50        red!50!green!50!blue!50//red!50!green!50!blue!50      red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50输red!50!green!50!blue!50入：                                  red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50(2red!50!green!50!blue!50                                     red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50–>red!50!green!50!blue!50  red!50!green!50!blue!504          red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50  red!50!green!50!blue!50–>red!50!green!50!blue!50          red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!503)red!50!green!50!blue!50  red!50!green!50!blue!50+         red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50  red!50!green!50!blue!50(5red!50!green!50!blue!50          red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50–>red!50!green!50!blue!50  red!50!green!50!blue!506          red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50  red!50!green!50!blue!50–>red!50!green!50!blue!50          red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!504)                                                            red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50        red!50!green!50!blue!50//red!50!green!50!blue!50      red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50输red!50!green!50!blue!50出：                                  red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!507red!50!green!50!blue!50  red!50!green!50!blue!50–>         red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50  red!50!green!50!blue!500red!50!green!50!blue!50          red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50–>red!50!green!50!blue!50  red!50!green!50!blue!508          red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50        red!50!green!50!blue!50//red!50!green!50!blue!50      red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50原red!50!green!50!blue!50因：                                  red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50342red!50!green!50!blue!50                                    red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50+red!50!green!50!blue!50                                      red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50465red!50!green!50!blue!50                                    red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50=red!50!green!50!blue!50                                      red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50807                                                           red!20!green!20
red!20!green!20!blue!20                                                                                     red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50        red!50!green!50!blue!50//red!50!green!50!blue!50      red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50type red!50!green!50!blue!50                                  red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50ListNode2red!50!green!50!blue!50                              red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50struct red!50!green!50!blue!50                                red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50{                                                             red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50        red!50!green!50!blue!50//red!50!green!50!blue!50      red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50 red!50!green!50!blue!50 red!50!green!50!blue!50Val           red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50  red!50!green!50!blue!50                                      red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50int                                                           red!20!green!20
```

```
type List2 struct {
    head *ListNode
}
type ListNode2 struct {
    val int
    next *ListNode
}
func Insert2(value int, list *ListNode, position *ListNode) {
    tempCell := new(ListNode)
    if tempCell == nil {
        fmt.Println("out of space")
    }
    tempCell.Val = value
    tempCell.Next = position.Next
    position.Next = tempCell
}
func PrintList2(list *ListNode) {
    if list.Next != nil {
        fmt.Println(list.Val)
        PrintList2(list.Next)
    } else {
        fmt.Println(list.Val)
    }
}
func main() {
    l1 := new(ListNode)
    l1.Val = 11
```

```
red!20!green!20!blue!20red!50!green!50!blue!50datared!50!green!50!blue!50 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50tored!50!green!50!blue!50 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50l1 red!20!green!20
red!20!green!20!blue!20 Insert2(9, listDate, l1) red!20!green!20
red!20!green!20!blue!20 Insert2(7, listDate, l1) red!20!green!20
red!20!green!20!blue!20 Insert2(5, listDate, l1) red!20!green!20
red!20!green!20!blue!20 l2 := blue!70new(ListNode) red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50 red!50!green!50!blue!50// red!20!green!20
red!20!green!20!blue!20 listDate2 := l2 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50 red!50!green!50!blue!50//red!50!green!50!blue!50 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50insertred!50!green!50!blue!50 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50datared!50!green!50!blue!50 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50tored!50!green!50!blue!50 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50l1 red!20!green!20
red!20!green!20!blue!20 Insert2(4, listDate2, l2) red!20!green!20
red!20!green!20!blue!20 Insert2(2, listDate2, l2) red!20!green!20
red!20!green!20!blue!20 Insert2(8, listDate2, l2) red!20!green!20
red!20!green!20!blue!20 l3 := addTwoNumbers(l1, l2) red!20!green!20
red!20!green!20!blue!20 PrintList(l3) red!20!green!20
red!20!green!20!blue!20 } red!20!green!20
red!20!green!20!blue!20 red!20!green!20
red!20!green!20!blue!20 func addTwoNumbers(l1 *ListNode, l2 *ListNode) *ListNode red!20!green!20
red!20!green!20!blue!20 { red!20!green!20
red!20!green!20!blue!20 promotion := 0 red!50!green!50!blue!50 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50//red!50!green!50!blue!50 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50进red!50!green!50!blue!50位 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50值red!50!green!50!blue!50, red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50 red!50!green!50!blue!50只 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50可red!50!green!50!blue!50能 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50为red!50!green!50!blue!500 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50或red!50!green!50!blue!501 red!20!green!20
red!20!green!20!blue!20 var head *ListNode red!50!green!50!blue!50// red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50 red!50!green!50!blue!50结 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50果red!50!green!50!blue!50表 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50的red!50!green!50!blue!50头 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50结red!50!green!50!blue!50点 red!20!green!20
red!20!green!20!blue!20 var rear *ListNode red!50!green!50!blue!50// red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50 red!50!green!50!blue!50保 red!20!green!20
```

```
for nil != l1 || nil != l2 {
    sum := 0
    if nil != l1 {
        sum += l1.Val
        l1 = l1.Next
    }
    if nil != l2 {
        sum += l2.Val
        l2 = l2.Next
    }

    sum += promotion
    promotion = 0

    if sum >= 10 {
        promotion = 1
        sum = sum % 10
    }

    node := &ListNode{
        sum,
        nil,
    }

    if nil == head {
        head = node
        rear = node
    } else {
        rear.Next = node
        rear = node
    }
}

if promotion > 0 {
```

```
        rear.Next = &ListNode{
            promotion,
            nil,
        }
    }
    return head
}
```

Chapter
hash table

6.1 twoSum

red!20!green!20!blue!20 //red!50!green!50!blue!50 red!50!green!50!blue!50给 red!20!green!20
red!20!green!20!blue!20 定red!50!green!50!blue!50一red!50!green!50!blue!50个 red!20!green!20
red!20!green!20!blue!20 整red!50!green!50!blue!50数red!50!green!50!blue!50数 red!20!green!20
red!20!green!20!blue!20 组red!50!green!50!blue!50numsred!50!green!50!blue!50和 red!20!green!20
red!20!green!20!blue!20 一red!50!green!50!blue!50个red!50!green!50!blue!50目 red!20!green!20
red!20!green!20!blue!20 标red!50!green!50!blue!50值 red!20!green!20
red!20!green!20!blue!20 t a r g e t red!50!green!50!blue!50 , red!20!green!20
red!20!green!20!blue!20 //red!50!green!50!blue!50 red!50!green!50!blue!50请 red!20!green!20
red!20!green!20!blue!20 你red!50!green!50!blue!50在red!50!green!50!blue!50该 red!20!green!20
red!20!green!20!blue!20 数red!50!green!50!blue!50组red!50!green!50!blue!50中 red!20!green!20
red!20!green!20!blue!20 找red!50!green!50!blue!50出red!50!green!50!blue!50和 red!20!green!20
red!20!green!20!blue!20 为red!50!green!50!blue!50目red!50!green!50!blue!50标 red!20!green!20
red!20!green!20!blue!20 值red!50!green!50!blue!50的red!50!green!50!blue!50那 red!20!green!20
red!20!green!20!blue!20 两red!50!green!50!blue!50个red!50!green!50!blue!50整 red!20!green!20
red!20!green!20!blue!20 数， red!50!green!50!blue!50并red!50!green!50!blue!50返 red!20!green!20
red!20!green!20!blue!20 回red!50!green!50!blue!50他red!50!green!50!blue!50们 red!20!green!20
red!20!green!20!blue!20 的red!50!green!50!blue!50数red!50!green!50!blue!50组 red!20!green!20
red!20!green!20!blue!20 下red!50!green!50!blue!50标。 red!20!green!20
red!20!green!20!blue!20 //red!50!green!50!blue!50 red!50!green!50!blue!50你 red!20!green!20
red!20!green!20!blue!20 可red!50!green!50!blue!50以red!50!green!50!blue!50假 red!20!green!20
red!20!green!20!blue!20 设red!50!green!50!blue!50每red!50!green!50!blue!50种 red!20!green!20
red!20!green!20!blue!20 输red!50!green!50!blue!50入red!50!green!50!blue!50只 red!20!green!20
red!20!green!20!blue!20 会red!50!green!50!blue!50对red!50!green!50!blue!50应 red!20!green!20
red!20!green!20!blue!20 一red!50!green!50!blue!50个red!50!green!50!blue!50答 red!20!green!20
red!20!green!20!blue!20 案。 red!50!green!50!blue!50但 red!20!green!20
red!20!green!20!blue!20 是， red!50!green!50!blue!50你red!50!green!50!blue!50不 red!20!green!20
red!20!green!20!blue!20 能red!50!green!50!blue!50重red!50!green!50!blue!50复 red!20!green!20
red!20!green!20!blue!20 利red!50!green!50!blue!50用red!50!green!50!blue!50这 red!20!green!20
red!20!green!20!blue!20 个red!50!green!50!blue!50数red!50!green!50!blue!50组 red!20!green!20

```
// 中同样的元素。
// 示例：
// 给定 nums = [2, 7, 11, 15], target = 9
// 因为 nums[0] + nums[1] = 2 + 7 = 9
// 所以返回 [0, 1]

package main

import (
    "fmt"
)

func main() {
    nums := []int{2, 7, 11, 15}
    target := 9
    arr := twoSum(nums, target)
    fmt.Println("arr = ", arr)
}
```

```
func twoSum(nums []int, target int) []int {
    h := make(map[int]int)
    for i, value := range nums {
        if wanted, ok := h[value]; ok {
            return []int{wanted, i}
        } else {
            h[target-value] = i
        }
    }
    return nil
}
```

Chapter
divide and conquer

Chapter
Search

8.1 Binary Search

8.2 Binary Search Tree

Chapter

Sort

9.1 comparison sort

9.1.1 quick sort

quick sort (C. A. R. Hoare 1960)

Divide and Conquer Algorithm for Quick Sort

Quick sort is better than heap sort

```
package main

import (
    "math/rand"
)

red!50!green!50!blue!50    red!50!green!50!blue!50//red!50!green!50!blue!50
red!50!green!50!blue!50quickred!50!green!50!blue!50
red!50!green!50!blue!50sort
red!50!green!50!blue!50    red!50!green!50!blue!50//red!50!green!50!blue!50
red!50!green!50!blue!50分red!50!green!50!blue!50治
red!50!green!50!blue!50排red!50!green!50!blue!50序
func main() {
    var z [] blue!70int

    blue!70for i := 0; i < 3; i++{
        z = append(z, rand.Intn(3))
    }

    quickSort(z)
}

func quickSort(list [] blue!70int) {
    blue!70if len(list) <= 1{
```

```
red!20!green!20!blue!20blue!70return red!20!green!20
red!20!green!20!}blue!20 red!20!green!20
red!20!green!20!blue!20 red!20!green!20
red!20!green!20!i, j blue!20:= 0, len(list) − 1 red!20!green!20
red!20!green!20!index blue!20 := 1 red!50!green!50!blue!50//red!50!green!50!blue!50 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50第red!50!green!50!blue!50一 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50次red!50!green!50!blue!50比 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50较red!50!green!50!blue!50索 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50引red!50!green!50!blue!50位 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50置 red!20!green!20
red!20!green!20!key blue!20 := list[0] red!50!green!50!blue!50//red!50!green!50!blue!50 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50第red!50!green!50!blue!50一 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50次red!50!green!50!blue!50比 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50较red!50!green!50!blue!50参 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50考red!50!green!50!blue!50值, red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50选red!50!green!50!blue!50择 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50第red!50!green!50!blue!50一 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50个 red!20!green!20
red!20!green!20!blue!20 red!20!green!20
red!20!green!20!blue!20blue!70if list[index] > key{ red!20!green!20
red!20!green!20!blue!20list[i], list[j] = list[j], list[i] red!20!green!20
red!20!green!20!blue!20j−− red!20!green!20
red!20!green!20!}blue!20blue!70else{ red!20!green!20
red!20!green!20!blue!20list[i], list[index] = list[index],list[i] red!20!green!20
red!20!green!20!blue!20i++ red!20!green!20
red!20!green!20!blue!20index++ red!20!green!20
red!20!green!20!}blue!20 red!20!green!20
red!20!green!20!blue!20 red!20!green!20
red!20!green!20!quickSort blue!20(list[:i]) red!50!green!50!blue!50// red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50 red!50!green!50!blue!50处 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50理red!50!green!50!blue!50参 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50考red!50!green!50!blue!50值 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50前red!50!green!50!blue!50面 red!20!green!20
red!20!green!20!blue!20red!50!green!50!blue!50值 red!20!green!20
red!20!green!20!quickSort blue!20(list[i+1:]) red!20!green!20
red!20!gre}n!20!blue!20 red!20!green!20
```

9.1.2 merge sort

John von Neumann1945 Merge Sort: The linear table to be sorted is continuously divided into several subtables until each subtable contains only one element. At this time, the subtable containing only one element can be considered as an ordered list. Merge the subtables in pairs. Each time a new table is generated, a new and longer ordered list is generated. Repeat this step until there is only one subtable left. This subtable is an ordered linear table.

9.1.3 heap sort

$O(nlgn)$ (heap sort run time)

9.2 bubble sort

Repeated swapping of two adjacent elements in reverse order

Chapter
Dynamic programming

10.1 调度问题

10.2 矩阵链乘法

10.3 公共子序列

10.4 Binary search tree

Chapter
greedy algorithm

11.1 Minimum spanning tree

11.2 Huffman coding

Data compression technology (David A. Huffman1952MIT)

A Method for the Construction of Minimum-Redundancy Codes

Chapter
RBTree

RBTree height is $O(lgn)$

color,key,left,right,p

12.1 nature

12.2 rotate