

# Algorithm

Shuwen He

1201220707@pku.edu.cn

Written in Peking University Library

2020 年 12 月 17 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

$\lg n = \log_2 n$  (Base 2 logarithm)

$\ln n = \log_e n$  (Natural logarithm)

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

$\lg \lg n = \lg(\lg n)$  (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 "C"
import "unsafe"

type List struct {
    head *ListNode
    tail *ListNode
    data  *ListData
}

type ListNode struct {
    value int
    next *ListNode
    prev *ListNode
}

type ListData struct {
    value int
}

func NewList() *List {
    list := &List{}
    list.head = nil
    list.tail = nil
    list.data = &ListData{}
    return list
}

func AddTail(list *List, value int) {
    node := &ListNode{
        value: value,
        next: nil,
        prev: list.tail,
    }
    list.tail = node
    list.data.value++
}

func AddHead(list *List, value int) {
    node := &ListNode{
        value: value,
        next: list.head,
        prev: nil,
    }
    list.head = node
    list.data.value++
}

func PrintList(list *List) {
    node := list.head
    for node != nil {
        println(node.value)
        node = node.next
    }
}

func main() {
    list := NewList()
    AddTail(list, 1)
    AddTail(list, 2)
    AddTail(list, 3)
    AddHead(list, 4)
    PrintList(list)
}
```

```

red!20!green!20!blue!20
red!20!green!20!blue!20 if tempCell == nil{
red!20!green!20!blue!20    fmt.Println("out of space")
red!20!green!20!blue!20
red!20!green!20!blue!20
red!20!green!20!blue!20 tempCell.Val = value
red!20!green!20!blue!20 tempCell.Next = position.Next
red!20!green!20!blue!20 position.Next = tempCell
red!20!green!20!blue!20
red!20!green!20!blue!20
red!20!green!20!blue!20 PrintList(list *ListNode) {
red!20!green!20!blue!20 if list.Next != nil{
red!20!green!20!blue!20    fmt.Print(list.Val, "->")
red!20!green!20!blue!20    PrintList(list.Next)
red!20!green!20!blue!20 } else {
red!20!green!20!blue!20    fmt.Println(list.Val)
red!20!green!20!blue!20 }
red!20!green!20!blue!20
red!20!green!20!blue!20
red!20!green!20!blue!20
red!20!green!20!blue!20 // red!50!green!50!blue!50 // red!50!green!50!blue!50
red!20!green!20!blue!50!green!50!blue!50 给 red!50!green!50!blue!50 出
red!20!green!20!blue!50!green!50!blue!50 两 red!50!green!50!blue!50 个
red!20!green!20!blue!50!green!50!blue!50 非 red!50!green!50!blue!50 空
red!20!green!20!blue!50!green!50!blue!50 的 red!50!green!50!blue!50 链
red!20!green!20!blue!50!green!50!blue!50 表 red!50!green!50!blue!50 用
red!20!green!20!blue!50!green!50!blue!50 来 red!50!green!50!blue!50 表
red!20!green!20!blue!50!green!50!blue!50 示 red!50!green!50!blue!50 两
red!20!green!20!blue!50!green!50!blue!50 个 red!50!green!50!blue!50 非
red!20!green!20!blue!50!green!50!blue!50 负 red!50!green!50!blue!50 的
red!20!green!20!blue!50!green!50!blue!50 整 red!50!green!50!blue!50 数。
red!20!green!20!blue!50!green!50!blue!50 其 red!50!green!50!blue!50 中，
red!20!green!20!blue!50!green!50!blue!50 它 red!50!green!50!blue!50 们
red!20!green!20!blue!50!green!50!blue!50 各 red!50!green!50!blue!50 自
red!20!green!20!blue!50!green!50!blue!50 的 red!50!green!50!blue!50 位
red!20!green!20!blue!50!green!50!blue!50 数 red!50!green!50!blue!50 是
red!20!green!20!blue!50!green!50!blue!50 按 red!50!green!50!blue!50 照
red!20!green!20!blue!50!green!50!blue!50 逆 red!50!green!50!blue!50 序
red!20!green!20!blue!50!green!50!blue!50 的 red!50!green!50!blue!50 方

```



red!20!green!20ed!50!20reen!50!blue!50 式 red!50!green!50!blue!50 存  
red!20!green!20ed!50!20reen!50!blue!50 储 red!50!green!50!blue!50 的 ,  
red!20!green!20ed!50!20reen!50!blue!50 // red!50!green!50!blue!50  
red!20!green!20ed!50!20reen!50!blue!50 并 red!50!green!50!blue!50 且  
red!20!green!20ed!50!20reen!50!blue!50 它 red!50!green!50!blue!50 们  
red!20!green!20ed!50!20reen!50!blue!50 的 red!50!green!50!blue!50 每  
red!20!green!20ed!50!20reen!50!blue!50 个 red!50!green!50!blue!50 节  
red!20!green!20ed!50!20reen!50!blue!50 点 red!50!green!50!blue!50 只  
red!20!green!20ed!50!20reen!50!blue!50 能 red!50!green!50!blue!50 存  
red!20!green!20ed!50!20reen!50!blue!50 储 red!50!green!50!blue!50 一  
red!20!green!20ed!50!20reen!50!blue!50 位 red!50!green!50!blue!50 数  
red!20!green!20ed!50!20reen!50!blue!50 字 。  
red!20!green!20ed!50!20reen!50!blue!50 // red!50!green!50!blue!50  
red!20!green!20ed!50!20reen!50!blue!50 如 red!50!green!50!blue!50 果 ,  
red!20!green!20ed!50!20reen!50!blue!50 我 red!50!green!50!blue!50 们  
red!20!green!20ed!50!20reen!50!blue!50 将 red!50!green!50!blue!50 这  
red!20!green!20ed!50!20reen!50!blue!50 两 red!50!green!50!blue!50 个  
red!20!green!20ed!50!20reen!50!blue!50 数 red!50!green!50!blue!50 相  
red!20!green!20ed!50!20reen!50!blue!50 加 red!50!green!50!blue!50 起  
red!20!green!20ed!50!20reen!50!blue!50 来 , red!50!green!50!blue!50 则  
red!20!green!20ed!50!20reen!50!blue!50 会 red!50!green!50!blue!50 返  
red!20!green!20ed!50!20reen!50!blue!50 回 red!50!green!50!blue!50 一  
red!20!green!20ed!50!20reen!50!blue!50 个 red!50!green!50!blue!50 新  
red!20!green!20ed!50!20reen!50!blue!50 的 red!50!green!50!blue!50 链  
red!20!green!20ed!50!20reen!50!blue!50 表 red!50!green!50!blue!50 来  
red!20!green!20ed!50!20reen!50!blue!50 表 red!50!green!50!blue!50 示  
red!20!green!20ed!50!20reen!50!blue!50 它 red!50!green!50!blue!50 们  
red!20!green!20ed!50!20reen!50!blue!50 的 red!50!green!50!blue!50 和 。  
red!20!green!20ed!50!20reen!50!blue!50 // red!50!green!50!blue!50  
red!20!green!20ed!50!20reen!50!blue!50 您 red!50!green!50!blue!50 可  
red!20!green!20ed!50!20reen!50!blue!50 以 red!50!green!50!blue!50 假  
red!20!green!20ed!50!20reen!50!blue!50 设 red!50!green!50!blue!50 除  
red!20!green!20ed!50!20reen!50!blue!50 了 red!50!green!50!blue!50 数  
red!20!green!20ed!50!20reen!50!blue!50 字 red!50!green!50!blue!50 00  
red!20!green!20ed!50!20reen!50!blue!50 之 red!50!green!50!blue!50 外 ,  
red!20!green!20ed!50!20reen!50!blue!50 这 red!50!green!50!blue!50 两  
red!20!green!20ed!50!20reen!50!blue!50 个 red!50!green!50!blue!50 数  
red!20!green!20ed!50!20reen!50!blue!50 都 red!50!green!50!blue!50 不

[illegible]

```

red!20!green!20!blue!20 会 red!50!green!50!blue!50 以
red!20!green!20!blue!20 0 red!50!green!50!blue!50 开
red!20!green!20!blue!20 头。
red!20!green!20!blue!20  red!50!green!50!blue!50 // red!50!green!50!blue!50
red!20!green!20!blue!20 示 red!50!green!50!blue!50 例：
red!20!green!20!blue!20  red!50!green!50!blue!50 // red!50!green!50!blue!50
red!20!green!20!blue!20 输 red!50!green!50!blue!50 入：
red!20!green!20!blue!20 (2 red!50!green!50!blue!50
red!20!green!20!blue!20 -> red!50!green!50!blue!50 red!50!green!50!blue!50 4
red!20!green!20!blue!20 red!50!green!50!blue!50 -> red!50!green!50!blue!50
red!20!green!20!blue!20 3) red!50!green!50!blue!50 red!50!green!50!blue!50 +
red!20!green!20!blue!20 red!50!green!50!blue!50 (5 red!50!green!50!blue!50
red!20!green!20!blue!20 -> red!50!green!50!blue!50 red!50!green!50!blue!50 6
red!20!green!20!blue!20 red!50!green!50!blue!50 -> red!50!green!50!blue!50
red!20!green!20!blue!20 4)
red!20!green!20!blue!20  red!50!green!50!blue!50 // red!50!green!50!blue!50
red!20!green!20!blue!20 输 red!50!green!50!blue!50 出：
red!20!green!20!blue!20 7 red!50!green!50!blue!50 red!50!green!50!blue!50 ->
red!20!green!20!blue!20 red!50!green!50!blue!50 0 red!50!green!50!blue!50
red!20!green!20!blue!20 -> red!50!green!50!blue!50 red!50!green!50!blue!50 8
red!20!green!20!blue!20  red!50!green!50!blue!50 // red!50!green!50!blue!50
red!20!green!20!blue!20 原 red!50!green!50!blue!50 因：
red!20!green!20!blue!20 3 4 2 red!50!green!50!blue!50
red!20!green!20!blue!20 + red!50!green!50!blue!50
red!20!green!20!blue!20 4 6 5 red!50!green!50!blue!50
red!20!green!20!blue!20 = red!50!green!50!blue!50
red!20!green!20!blue!20 8 0 7
red!20!green!20!blue!20
red!20!green!20!blue!20  red!50!green!50!blue!50 // red!50!green!50!blue!50
red!20!green!20!blue!20 typed red!50!green!50!blue!50
red!20!green!20!blue!20 List Node 2 red!50!green!50!blue!50
red!20!green!20!blue!20 s t r u c t red!50!green!50!blue!50
red!20!green!20!blue!20 {
red!20!green!20!blue!20  red!50!green!50!blue!50 // red!50!green!50!blue!50
red!20!green!20!blue!20 red!50!green!50!blue!50 red!50!green!50!blue!50 Val
red!20!green!20!blue!20 red!50!green!50!blue!50
red!20!green!20!blue!20 i n t

```



```

red!20!green!20!blue!20!50!green!50!blue!50 data red!50!green!50!blue!50
red!20!green!20!blue!20!50!green!50!blue!50 to red!50!green!50!blue!50
red!20!green!20!blue!20!50!green!50!blue!50 l1
red!20!green!20!blue!20!50!green!50!blue!50 t2 (9, listDate, 11)
red!20!green!20!blue!20!50!green!50!blue!50 t2 (7, listDate, 11)
red!20!green!20!blue!20!50!green!50!blue!50 t2 (5, listDate, 11)
red!20!green!20!blue!20!50!green!50!blue!50 new (ListNode)
red!20!green!20!blue!20!50!green!50!blue!50 //
red!20!green!20!blue!20!50!green!50!blue!50 := 12
red!20!green!20!blue!20!50!green!50!blue!50 // red!50!green!50!blue!50
red!20!green!20!blue!20!50!green!50!blue!50 insert red!50!green!50!blue!50
red!20!green!20!blue!20!50!green!50!blue!50 data red!50!green!50!blue!50
red!20!green!20!blue!20!50!green!50!blue!50 to red!50!green!50!blue!50
red!20!green!20!blue!20!50!green!50!blue!50 l1
red!20!green!20!blue!20!50!green!50!blue!50 t2 (4, listDate2, 12)
red!20!green!20!blue!20!50!green!50!blue!50 t2 (2, listDate2, 12)
red!20!green!20!blue!20!50!green!50!blue!50 t2 (8, listDate2, 12)
red!20!green!20!blue!20!50!green!50!blue!50 addTwoNumbers (l1, 12)
red!20!green!20!blue!20!50!green!50!blue!50 List (13)
red!20!green!20!blue!20
red!20!green!20!blue!20
red!20!green!20!blue!20 addTwoNumbers (l1 * ListNode, 12 * ListNode) * ListNode
red!20!green!20!blue!20
red!20!green!20!blue!20 position := 0 red!50!green!50!blue!50
red!20!green!20!blue!20!50!green!50!blue!50 // red!50!green!50!blue!50
red!20!green!20!blue!20!50!green!50!blue!50 进 red!50!green!50!blue!50 位
red!20!green!20!blue!20!50!green!50!blue!50 值 red!50!green!50!blue!50,
red!20!green!20!blue!20!50!green!50!blue!50 red!50!green!50!blue!50 只
red!20!green!20!blue!20!50!green!50!blue!50 可 red!50!green!50!blue!50 能
red!20!green!20!blue!20!50!green!50!blue!50 为 red!50!green!50!blue!50 0
red!20!green!20!blue!20!50!green!50!blue!50 或 red!50!green!50!blue!50 1
red!20!green!20!blue!20!50!green!50!blue!50 head * ListNode red!50!green!50!blue!50 //
red!20!green!20!blue!20!50!green!50!blue!50 red!50!green!50!blue!50 结
red!20!green!20!blue!20!50!green!50!blue!50 果 red!50!green!50!blue!50 表
red!20!green!20!blue!20!50!green!50!blue!50 的 red!50!green!50!blue!50 头
red!20!green!20!blue!20!50!green!50!blue!50 结 red!50!green!50!blue!50 点
red!20!green!20!blue!20!50!green!50!blue!50 rear * ListNode red!50!green!50!blue!50 //
red!20!green!20!blue!20!50!green!50!blue!50 red!50!green!50!blue!50 保

```

```

red!20!green!20!blue!20!blue!70for nil != 11 || nil != 12 {
red!20!green!20!blue!20sum := 0
red!20!green!20!blue!20blue!70if nil != 11 {
red!20!green!20!blue!20    sum += 11.Val
red!20!green!20!blue!20    11 = 11.Next
red!20!green!20!blue!20}
red!20!green!20!blue!20blue!70if nil != 12 {
red!20!green!20!blue!20    sum += 12.Val
red!20!green!20!blue!20    12 = 12.Next
red!20!green!20!blue!20}
red!20!green!20!blue!20sum += promotion
red!20!green!20!blue!20promotion = 0
red!20!green!20!blue!20
red!20!green!20!blue!20blue!70if sum >= 10 {
red!20!green!20!blue!20    promotion = 1
red!20!green!20!blue!20    sum = sum % 10
red!20!green!20!blue!20}
red!20!green!20!blue!20
red!20!green!20!blue!20node := &ListNode{
red!20!green!20!blue!20    sum,
red!20!green!20!blue!20    nil,
red!20!green!20!blue!20}
red!20!green!20!blue!20
red!20!green!20!blue!20blue!70if nil == head {
red!20!green!20!blue!20    head = node
red!20!green!20!blue!20    rear = node
red!20!green!20!blue!20} else {
red!20!green!20!blue!20    rear.Next = node
red!20!green!20!blue!20    rear = node
red!20!green!20!blue!20}
red!20!green!20!blue!20
red!20!green!20!blue!20blue!70if promotion > 0 {

```

[illegible]

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

```
red!20!green!20!blue!20
red!20!green!20
red!20!green!20
red!20!green!20
red!20!green!20
red!20!green!20
red!20!green!20
red!20!green!20
```

# Chapter

## hash table

### 6.1 twoSum

给你一个整数数组 `nums` 和一个目标值 `target`，请你在该数组中找出和为目标值的那两个整数，并返回它们的数组下标。

你可以假设每种输入只会对应一个答案。但是，数组中同一个元素在答案里不能重复出现。

你可以按任意顺序返回答案。







# 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

red!20!green!20!blue!20main

red!20!green!20!blue!20

red!20!green!20!blue!20

red!20!green!20!blue!20/ rand ”

red!20!green!20!blue!20

red!20!green!20!blue!20

red!20!green!20!blue!20red!50!green!50!blue!50//red!50!green!50!blue!50

red!20!green!20!blue!20red!50!green!50!blue!50quickred!50!green!50!blue!50

red!20!green!20!blue!20red!50!green!50!blue!50sort

red!20!green!20!blue!20red!50!green!50!blue!50//red!50!green!50!blue!50

red!20!green!20!blue!20red!50!green!50!blue!50分red!50!green!50!blue!50治

red!20!green!20!blue!20red!50!green!50!blue!50排red!50!green!50!blue!50序

red!20!green!20!blue!20( ) {

red!20!green!20!blue!20[] blue!70int

red!20!green!20!blue!20

red!20!green!20!blue!20for i := 0; i < 3; i++{

red!20!green!20!blue!20= append(z, rand.Intn(3))

red!20!green!20!blue!20}

red!20!green!20!blue!20

red!20!green!20!blue!20Sort(z)

red!20!green!20!blue!20

red!20!green!20!blue!20

red!20!green!20!blue!20quickSort(list [] blue!70int) {

red!20!green!20!blue!20if len(list) <= 1{

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

red!20!green!20

```

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

```

### 9.1.2 merge sort

John von Neumann<sup>1945</sup> 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

$O(n \lg n)$  (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  $O(\lg n)$

color, key, left, right, p

**12.1 nature**

**12.2 rotate**