

# 《大话数据结构》

以下是本人（黄国洪Samwee）学习《大话数据结构》这本书的学习笔记。

封面：



## 1 数据机构绪论

## 2 算法

## 3 线性表

### 3.1 顺序存储结构

### 3.2 链式存储结构

单链表

静态链表

循环链表

双向链表

## 4 栈与队列

### 4.1 栈 stack 只能一头插入删除，后进先出

4.2 队列 queue 一端插入一端删除，先进先出

## 5 串 string

KMP模式匹配算法

## 6 树 tree

6.1 度 degree 生出几个，度就是几个。

树的度是树内各结点的度的最大值

6.5 二叉树 binary tree

6.8 遍历二叉树

1 前序

2 中序 从左往右，从下往上

3 后序 先叶子后结点

## 7 图 graph

7.2 一些定义

图 $G=(V, \{E\})$

无向图undirected graphs，无向边edge，用  $()$  表示

有向图directed graphs，有向边arc，用  $\langle \rangle$  表示

7.4 图的存储结构

邻接矩阵

邻接表

十字链表

邻接多重表

边集数组

7.6 最小生成树minimum cost spanning tree

Prim算法，适合稠密图

Kruskal算法，针对边展开，适合稀疏图

7.7 最短路径

Dijkstra算法

Floyd算法

## 7.8 拓扑排序

AOV网 activity on vertex network

顶点全被输出，说明是AOV网，不存在环

顶点数少了，说明不是AOV网，存在环

## 7.9 关键路径

AOE网 activity on edge network

从源点到汇点具有最大长度的路径叫关键路径

# 8 查找 searching

## 8.2 概论

关键字key

主关键字primary key，理解为股票代码一样唯一

次关键字secondary key，理解为涨跌幅一样不唯一

## 8.3 顺序查找 sequential search

## 8.4 有序表查找

1 折半查找

2 插值查找interpolation search

3 斐波那契查找 fibonacci search

## 8.5 线性索引查找

1 稠密索引：每个记录对应一个索引项，就像一个小本本一样

2 分块索引

3 倒排索引

## 8.6 二叉排序树 binary sort tree

8.7 平衡二叉树 AVL树 self-balancing binary search tree 或 height-balanced binary search tree

深度差不超过1，也就是平衡因子BF (balance factor) 只能是-1, 0, 1

## 8.8 多路查找树 multi-way search tree

2-3树：一个2结点包括1个元素2个孩子，一个3结点包括2个元素3个孩子

2-3-4树：一个4结点包括3个元素4个孩子

B树：2-3树是3阶B树，2-3-4树是4阶B树

B+树：

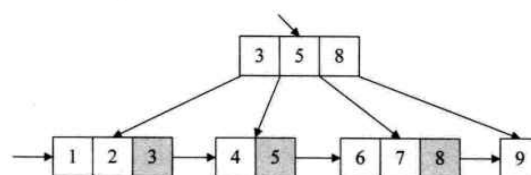


图 8-8-19

## 8.9 散列表查找，哈希表

散列技术，每个关键字key对应一个存储位置 $f(key)$ 。

$f$  称为散列函数或哈希hash函数。采用散列技术将记录存储在一块连续的存储空间中，这块连续存储空间称为散列表或哈希表hash table

散列技术既是一种存储方法，也是一种查找方法。

散列技术的记录之间不存在什么逻辑关系，它只与关键字有关联。因此，散列主要是面向查找的存储结构。

## 8.11 处理散列冲突的方法

- 1 开放定址法：房子被买走就找别的房子
- 2 再散列函数法：事先准备多个散列函数
- 3 链地址法：有冲突的话就在单链表中增加结点
- 4 公共溢出区法：为所有冲突的关键字建立一个公共溢出区来存放

# 9 排序

## 简单算法

9.3 冒泡排序bubble sort，复杂度 $O(n^2)$

9.4 简单选择排序simple selection sort，比较次数还是那么多，但是少操作。复杂度 $O(n^2)$

9.5 直接插入排序straight insertion sort，像扑克牌。复杂度 $O(n^2)$ ，是简单排序中性能最好的。

## 改进算法

9.6 希尔排序shell sort，复杂度 $O(n^2)$ ，基本有序时，对全体记录进行一次直接插入排序。

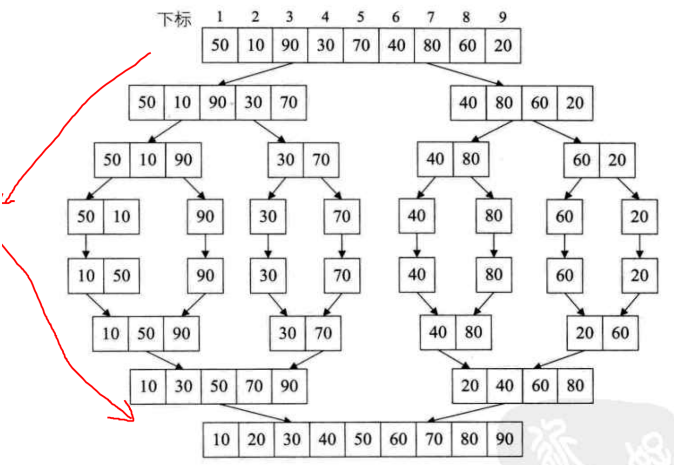
9.7 堆排序heap sort，复杂度 $O(n\log n)$

堆是具有下列性质的完全二叉树：每个结点的值都大于等于其左右孩子结点的值，叫大顶堆。

每个结点的值都小于等于其左右孩子结点的值，叫小顶堆。

堆排序就是最大的移走，剩下的再做成一个堆。如此反复。

9.8 归并排序 merging sort，时间复杂度 $O(n\log n)$ ，空间复杂度 $O(n+\log n)$



9.9 快速排序 quick sort

这一段代码的核心是“`pivot=Partition(L,low,high);`”在执行它之前，L.r 的数组值为{50,10,90,30,70,40,80,60,20}。**Partition** 函数要做的，就是先选取其中的一个关键字，比如选择第一个关键字 50，然后想尽办法将它放到一个位置，使得它左边的值都比它小，右边的值比它大，我们将这样的关键字称为枢轴（**pivot**）。

平均的时间复杂度 $O(n\log n)$ ，空间复杂度 $O(\log n)$

表 9-10-1

排序方法	平均情况	最好情况	最坏情况	辅助空间	稳定性
冒泡排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定
简单选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	稳定
直接插入排序	$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$	稳定
希尔排序	$O(n\log n) \sim O(n^2)$	$O(n^{1.3})$	$O(n^2)$	$O(1)$	不稳定
堆排序	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(1)$	不稳定
归并排序	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n)$	稳定
快速排序	$O(n\log n)$	$O(n\log n)$	$O(n^2)$	$O(\log n) \sim O(n)$	不稳定

