

# “数据结构基础”复习

**主要内容：**三大类数据结构+两大类问题（排序与查找）

**掌握层次：**概念、方法、编程（下划线部分的程序需要熟练掌握）

**题型（机考）：**判断(2\*10)，选择（3\*20），填空（3\*4），函数（1\*8）

## 一 基本概念

1. 时、空复杂性（ $\Omega$   $\theta$   $O$ ）及等级、RUN TIME CALCULATION
2. 数据结构基本概念：
  - 数据类型、对象、操作、数据结构
  - 三大类数据结构：线性（堆栈、队列）、树、图
  - 数据结构的物理表示方式：数组、链表

## 二、堆栈和队列（STACK AND QUEUE）

1. 堆栈（STACK）：
  - 概念：在同一端插入和删除，FILO
  - 表示：数组、链表
  - 操作：入/出栈，空/满判断
2. 队列（Queue）
  - 概念：在一端插入而在另一端删除，FIFO
  - 表示：数组、链表
  - 操作：入/出队列，空/满判断
3. 应用----表达式求值，EVAL，POSTFIX

## 三、树（TREE）

- 1、二叉树的若干性质：

树节点数与层次的关系、完全二叉树数组表示中结点与父子节点的关系、 $n_0/n_1/n_2$  之间的关系。

- 2、二叉树的表示：数组、链表方式

- 3、树的基本操作：遍历（traversal）

二叉数的先/中/后序/层次 (preorder/inorder/postorder/level) 以及非递归方法  
森林的遍历 (forest travel)  
森林与树的二叉树表示

#### 4. 堆 (HEAP):

- 概念
- 表示: 数组 (完全二叉树方式) (complete binary tree)
- 操作: 插入 (any)、删除 (min/max),  $O(n)$  方式建堆
- d-heap

#### 5. 二分查找树(binary search tree)

- 概念
- 表示: 链表
- 操作: 查找(findMax/findMin)、插入(Insert)、删除(Delete)

### 四、图(graph)

1. 基本概念: component, degree, connected, path,...
2. 表示: 邻接矩阵(adjacency matrix)、邻接表 adjacency list (逆邻接表)、  
邻接多重表(adjacency multi-list)、十字链表(orthotropic list)
3. 基本操作: DFS (depth-first search)、BFS(bread-first search)
4. 图上的典型算法:
  - 最小生成树(minimum spanning tree): Prim's algorithm and Kruskal's Algorithm
  - 最短路径(shortest-path): Dijkstra Algorithm, unweighted shortest path<sup>2</sup>
  - 拓扑排序 (Topological Sort)<sup>2</sup>
  - 网络流量问题 (Network Flow Problems),
  - 关键路径 (Critical Path)
  - 双连图/关节点(bi-connectivity/articulation points)  
(DFS, LOW), bi-connected components

### 五、排序(sorting) (算法、时间复杂性、空间复杂性、稳定性)

1. 插入排序: 基本插入排序方法(insertion sort)、希尔排序(shell sort)
2. 交换排序: 基本交换排序 (冒泡 bubble sort)、快速排序 (quick sort)
3. 归并排序(merge sort):
4. 选择排序: 基本选择排序(selection sort)、堆排序 (heap sort)
5. 基数排序 (radix sorting)
6. 排序算法的时间、空间复杂性比较, 稳定性

## 六 HASH

1. 基本思想: 解决动态查找问题
2. HASH 函数的构造方法(hash function construction):
3. 冲突处理方法(collision resolution):  
    开放地址(open addressing) (linear probing, quadratic probing)、  
    链表(chaining)、double hashing, 再 HASH(rehashing)

## 七、Disjoint Set

1. Disjoint Set → Find、union on sets.
2. Representation: tree, node with parent link
3. Find improvement: path compression
4. Union improvement: union-by-size, union-by-height