# 数据结构复习要点

关于教学内容，总结一下（以下内容不需要掌握）：

2.3.3 单向循环链表：不要求

2.3.4 双向链表：不要求

3.2.2 栈的链式存储结构及操作实现：不要求

3.4.2 队列的链式存储结构及操作实现：不要求

3.5 递归中的 2. 递归数据结构不要求

4.3. 串的链式存储结构及其实现：不要求

6.4 线索二叉树：不介绍

7.4.3 最小代价生成树: 只介绍概念和过程，无需详细算法。

有时考题中有这方面的内容，一般能得分。

7.5 最短路径：只介绍概念，无需算法。

8.2.3 分块查找：只介绍概念，无需算法。

9.2.2 希尔排序：不介绍

9.4.2 堆排序：不介绍

-----------------------------------------------

前半段20-30%的题目，其他则来源后半段6-9章。

第5章point：

**1. 稀疏矩阵的三元组表示。**

**2. 稀疏矩阵的行/列的单链表示**。

3. 根据广义表表示画图。

**4. 已知矩阵的行/列的单链表示，写出矩阵，并把矩阵转置后，用行/列的单链表示。**

第6章point：

1. 给出树的广义表表示法，画出这棵树，并分析这棵树的度和高度。

2. 画出有4个结点的树的所有形态。

**3. 【编程】二叉树的先根、中根、后根遍历算法。**

4. 写出给定二叉树的先根、中根、后根序列。

5. 给定先根/后根+中根表示的二叉树遍历，画出这棵树。

6. 给定先根+后根表示的二叉树遍历，画出所有形态。

**7. 画出用栈实现二叉树的非递归中根遍历过程。**

8. 画出用队列实现按层次遍历二叉树时队列内容的变化过程。

**9. 编程实现完全二叉树的创建函数。**

10. 给出二叉树的广义表表示法，画出这棵树。

11. 二叉树与森林的相互转化。

第7章point：

1. 写出无向图/有向图（带权/不带权）的邻接矩阵。

2. 根据邻接矩画图。

3. 写出无向图/有向图的邻接表。

4. 根据邻接表画图。

**5. 【编程】已知图的结点类定义为：**

**public class GraphNode<T> {**

**private T data;**

**private bool visited;**

**private List<GraphNode<T>> neighbors;**

**private List<int> costs; // 边的权值**

**……**

**}**

**图类定义为：**

**public class Graph<T> {**

**private List<GraphNode<T>> nodes;//结点表**

**……**

**}**

**编写在无向图/有向图类中增加边的函数。该函数定义为：**

**void AddUndirectedEdge(T from, T to, int cost)**

6. 从一个结点出发，写出图的深度优先遍历序列，并说明算法过程。

7. 从一个结点出发，写出图的广度优先遍历序列，并说明算法过程。

**8. 【编程】实现以邻接矩阵存储的图的深度优先遍历算法。**

**9. 【编程】实现以邻接表存储的图的深度优先遍历算法。**

10. 画出给定图的深度优先遍历生成树。

11. 画出给定图的广度优先遍历生成树。

12. 用Kruskal算法构造给定带权图的最小代价生成树，并简述算法过程。

13. 用Prim算法构造给定带权图的最小代价生成树，并简述算法过程。

14. 写出从给定结点开始，到某个结点的最短路径，并计算路径长度。

第8章point：

1： 算法的文字描述

2： 算法的过程作图

3： 算法的编程实现

各种查找算法的时间复杂度，最好情况，最坏情况，空间复杂度，算法的稳定性。

**1：【编程】定义一个顺序查找表，并实现顺序查找，要在主函数中测试。**

**2：【编程】定义一个链式查找表，并实现顺序查找，要在主函数中测试。**

**3：【编程】用C#语言和递归法实现二分查找算法，要在主函数中测试。**

**4：【编程】用C#语言和非递归法实现二分查找算法，要在主函数中测试。**

5：画出n=9时的二分查找算法的二叉判定树。

6：数据序列： {10, 6, 23, ...}，

已知数据小于100，当块大小为10时，画出动态分块查找表结构。

9：给出一个数据序列，要求画出二叉查找树。

**10：【编程】实现建立二叉查找树的代码。**

**11：【编程】实现在二叉查找树中查找的函数。**

12：哈希表探测地址法。（作图）

13：哈希链法。（作图）

第9章point：

1： 算法的文字描述

2： 算法的过程作图

3： 算法的编程实现

各种排序的时间复杂度，最好情况，最坏情况，空间复杂度，算法的稳定性。

1： 插入

直接插入

二分法查找+直接插入

\*数组的直接插入排序算法实现，并编写主函数测试。

\*二分查找+直接插入排序算法实现，并编写主函数测试。

2： 交换

冒泡法

快速排序

\*冒泡排序，并编写主函数测试。

\*快速排序的算法，并编写主函数测试。

3： 选择

直接选择排序

\*选择排序算法，并编写主函数测试。

4： 归并

作图，对给定序列进行归并排序，画出每趟循环后的执行结果。