**基础思维**

1. 局部变量
   1. 如果后续会丢失某个结构数据，可以先存储起来
   2. 如果需要创建数据来补全结构
2. For、While的条件必须清楚，不清楚大概率会失败

**字符串**

1. 英文字符如何设置hash？是否适用字符串？

**数组**

1. 正序遍历的应用
2. 倒序遍历的应用
3. 下标和值，选择哪个更方便解决问题
4. 指针法

**链表**

1. 指针法，通过一些指针辅助自己思考，需要注意以下问题
   1. 指针的移动是否能解决中间链表
   2. 指针的第一次初始化
   3. 指针是否能满足末尾，通常也是跳出循环的部分
   4. 不同指针的重叠是否会有影响
2. 单链表反转

**栈**

栈的实现

1. 数组的实现
2. 链表的实现

后进先出的特性

1. 弹出特性，peek、pop、push

**队列**

队列的实现

1. 数组的实现
2. 链表的实现

先进先出的特性

1. 请看Java Queue API

**双端队列**

双端队列的实现

1. 数组的实现
2. 链表的实现

双端队列的特性

1. 可以从两端增加、或移除数据

**优先队列**

优先队列的实现

1. 数组结构+完全二叉树

完全二叉树的性质

* 1. 数组里的第一个元素 array[0] 拥有最高的优先级别。
  2. 给定一个下标 i，那么对于元素 array[i] 而言：

它的父节点所对应的元素下标是 (i-1)/2

它的左孩子所对应的元素下标是 2×i + 1

它的右孩子所对应的元素下标是 2×i + 2

* 1. 数组里每个元素的优先级别都要高于它两个孩子的优先级别。

**树**

树的遍历

深度优先搜索（DFS）

1. 前序遍历，根左右
2. 中需遍历，左根右
3. 后序遍历，左右根

广度优先搜索（BFS）

**二叉树**

性质

**完全二叉树**

性质

**堆**

性质

**图**

图可以说是所有数据结构里面知识点最丰富的一个，最基本的**知识点**如下。

阶（Order）、度：出度（Out-Degree）、入度（In-Degree）

树（Tree）、森林（Forest）、环（Loop）

有向图（Directed Graph）、无向图（Undirected Graph）、完全有向图、完全无向图

连通图（Connected Graph）、连通分量（Connected Component）

存储和表达方式：邻接矩阵（Adjacency Matrix）、邻接链表（Adjacency List）<https://www.cnblogs.com/chay/articles/10259220.html>

围绕图的**算法**也是五花八门。

图的遍历：深度优先、广度优先

环的检测：有向图、无向图

拓扑排序

最短路径算法：Dijkstra、Bellman-Ford、Floyd Warshall

连通性相关算法：Kosaraju、Tarjan、求解孤岛的数量、判断是否为树

图的着色、旅行商问题等

以上的知识点只是图论里的冰山一角，对于算法面试而言，完全不需要对每个知识点都一一掌握，而应该有的放矢地进行准备。

**必会知识点**

根据长期的经验总结，以下的知识点是必须充分掌握并反复练习的。

图的存储和表达方式：邻接矩阵（Adjacency Matrix）、邻接链表（Adjacency List）

图的遍历：深度优先、广度优先

二部图的检测（Bipartite）、树的检测、环的检测：有向图、无向图

拓扑排序

联合-查找算法（Union-Find）

最短路径：Dijkstra、Bellman-Ford

**递归**

**算法题解题思路**

解题入口

1. 直觉暴力法，一般是大多数人想得到，但效率不怎么好
2. 问题转化法，将现问题转化成已经熟悉的解题模型

解题优化