> 注意！！！这部分内容会进行重构，以下内容仅作为参考。

Queue

什么是队列

队列是数据结构中比较重要的一种类型，它支持 FIFO，尾部添加、头部删除（先进队列的元素先出队列），跟我们生活中的排队类似。

队列的种类

- **单队列**（单队列就是常见的队列, 每次添加元素时，都是添加到队尾，存在“假溢出”的问题也就是明明有位置却不能添加的情况）
- **循环队列**（避免了“假溢出”的问题）

Java 集合框架中的队列 Queue

Java 集合中的 Queue 继承自 Collection 接口 ， Deque, LinkedList, PriorityQueue, BlockingQueue 等类都实现了它。 Queue 用来存放 等待处理元素 的集合，这种场景一般用于缓冲、并发访问。 除了继承 Collection 接口的一些方法，Queue 还添加了额外的 添加、删除、查询操作。

Set

什么是 Set

Set 继承于 Collection 接口，是一个不允许出现重复元素，并且无序的集合，主要 HashSet 和 TreeSet 两大实现类。

在判断重复元素的时候，HashSet 集合会调用 hashCode()和 equal()方法来实现；TreeSet 集合会调用 compareTo方法来实现。

补充：有序集合与无序集合说明

- 有序集合：集合里的元素可以根据 key 或 index 访问 (List、Map)
- 无序集合：集合里的元素只能遍历。（Set）

HashSet 和 TreeSet 底层数据结构

HashSet 是哈希表结构，主要利用 HashMap 的 key 来存储元素，计算插入元素的 hashCode 来获取元素在集合中的位置；

TreeSet 是红黑树结构，每一个元素都是树中的一个节点，插入的元素都会进行排序；

List

什么是List

在 List 中，用户可以精确控制列表中每个元素的插入位置，另外用户可以通过整数索引（列表中的位置）访问元素，并搜索列表中的元素。 与 Set 不同，List 通常允许重复的元素。 另外 List 是有序集合而 Set 是无序集合。

List的常见实现类

ArrayList 是一个数组队列，相当于动态数组。它由数组实现，随机访问效率高，随机插入、随机删除效率低。

LinkedList 是一个双向链表。它也可以被当作堆栈、队列或双端队列进行操作。LinkedList随机访问效率低，但随机插入、随机删除效率高。

Vector 是矢量队列，和ArrayList一样，它也是一个动态数组，由数组实现。但是ArrayList是非线程安全的，而Vector是线程安全的。

Stack 是栈，它继承于Vector。它的特性是：先进后出(FILO, First In Last Out)。

树

1 二叉树

二叉树（百度百科）

(1)完全二叉树——若设二叉树的高度为h，除第 h 层外，其它各层 (1～h-1) 的结点数都达到最大个数，第h层有叶子结点，并且叶子结点都是从左到右依次排布，这就是完全二叉树。

(2)满二叉树——除了叶结点外每一个结点都有左右子叶且叶子结点都处在最底层的二叉树。

(3)平衡二叉树——平衡二叉树又被称为AVL树（区别于AVL算法），它是一棵二叉排序树，且具有以下性质：它是一棵空树或它的左右两个子树的高度差的绝对值不超过1，并且左右两个子树都是一棵平衡二叉树。

2 完全二叉树

完全二叉树（百度百科）

完全二叉树：叶节点只能出现在最下层和次下层，并且最下面一层的结点都集中在该层最左边的若干位置的二叉树。

3 满二叉树

满二叉树（百度百科，国内外的定义不同）

国内教程定义：一个二叉树，如果每一个层的结点数都达到最大值，则这个二叉树就是满二叉树。也就是说，如果一个二叉树的层数为K，且结点总数是(2^k) -1 ，则它就是满二叉树。

堆

数据结构之堆的定义

堆是具有以下性质的完全二叉树：每个结点的值都大于或等于其左右孩子结点的值，称为大顶堆；或者每个结点的值都小于或等于其左右孩子结点的值，称为小顶堆。

4 二叉查找树（BST）

浅谈算法和数据结构: 七 二叉查找树

二叉查找树的特点：

1. 若任意节点的左子树不空，则左子树上所有结点的 值均小于它的根结点的值；
2. 若任意节点的右子树不空，则右子树上所有结点的值均大于它的根结点的值；
3. 任意节点的左、右子树也分别为二叉查找树；
4. 没有键值相等的节点（no duplicate nodes）。

5 平衡二叉树（Self-balancing binary search tree）

平衡二叉树（百度百科，平衡二叉树的常用实现方法有红黑树、AVL、替罪羊树、Treap、伸展树等）

6 红黑树

红黑树特点:

1. 每个节点非红即黑；
2. 根节点总是黑色的；
3. 每个叶子节点都是黑色的空节点（NIL节点）；
4. 如果节点是红色的，则它的子节点必须是黑色的（反之不一定）；
5. 从根节点到叶节点或空子节点的每条路径，必须包含相同数目的黑色节点（即相同的黑色高度）。

红黑树的应用：

TreeMap、TreeSet以及JDK1.8的HashMap底层都用到了红黑树。

为什么要用红黑树?

简单来说红黑树就是为了解决二叉查找树的缺陷，因为二叉查找树在某些情况下会退化成一个线性结构。详细了解可以查看 漫画：什么是红黑树？（也介绍到了二叉查找树，非常推荐）

推荐文章：

- 漫画：什么是红黑树？（也介绍到了二叉查找树，非常推荐）
- 寻找红黑树的操作手册（文章排版以及思路真的不错）
- 红黑树深入剖析及Java实现（美团点评技术团队）

7 B-，B+，B*树

二叉树学习笔记之B树、B+树、B*树

《B-树，B+树，B*树详解》

《B-树，B+树与B*树的优缺点比较》

B-树（或B树）是一种平衡的多路查找（又称排序）树，在文件系统中有所应用。主要用作文件的索引。其中的B就表示平衡(Balance)

1. B+ 树的叶子节点链表结构相比于 B- 树便于扫库，和范围检索。
2. B+树支持range-query（区间查询）非常方便，而B树不支持。这是数据库选用B+树的最主要原因。
3. B*树 是B+树的变体，B*树分配新结点的概率比B+树要低，空间使用率更高；

8 LSM 树

[HBase] LSM树 VS B+树

B+树最大的性能问题是会产生大量的随机IO

为了克服B+树的弱点，HBase引入了LSM树的概念，即Log-Structured Merge-Trees。

LSM树由来、设计思想以及应用到HBase的索引

图

BFS及DFS

- 《使用BFS及DFS遍历树和图的思路及实现》