- 数据结构
  - 线性数据结构
    - 数组：存放着一组相同类型且有序的数据，需要预先指定数组的长度，有一维数组、二维数组、多维数组等
    - 链表：它采用动态分配内存的形式实现，用一组任意的存储单元存放数据元素链表的，一般为每个元素增设指针域，用来指向后继元素
    - 栈：先进后出的线性结构。栈底是最高位，操作只能在栈顶进行
    - 队列：允许在序列两端进行操作，一般队列也被称为先进先出的线性结构
    - 线性表：允许在序列任意位置进行操作，线性表的操作位置不受限制，线性表的操作十分灵活，常用操作包括在任意位置插入和删除，以及查询和修改任意位置的元素
  - 树形结构
    - 二叉树：二叉树中的节点最多只能有两个子节点：一个是左侧节点，一个是右侧节点
      - 二叉树可以是空树
      - 二叉树的每个结点都恰好有两棵子树，其中一个或两个可能为空
      - 二叉树中每个结点的左、右子树的位置不能颠倒，若改变两者的位置，就成为另一棵二叉树
    - 完全二叉树：每个结点都与深度为k的满二叉树中编号从1至n的结点一一对应，则称为完全二叉树
      - 堆
        - 无序，顺序随机
        - 由程序员分配释放， 若程序员不释放，程序结束时可能由OS回收
        - 大顶堆
        - 小顶堆
    - 二叉查找树(排序树)
      - 若它的左子树不空，则左子树上所有结点的值均小于根结点的值
      - 若它的右子树不空，则右子树上所有结点的值均大于根结点的值
      - 它的左、右子树也分别是二叉查找树
      - 二叉查找树的遍历
        - 先序遍历：根左右
        - 中序遍历：左根右（要还原一颗树，必须知道中序遍历结果）
        - 后序遍历：左右根
    - 平衡二叉树
      - 左子树和右子树的高度之差的绝对值不超过1
      - 它的左子树和右子树都是平衡二叉树
    - 红黑树
      - 平衡二叉树的一种变种
        - 最上面的节点是黑色
        - 每个结点都只能是红色或者黑色
        - 每片叶子都是黑色的
        - 如果一个结点是红色的，则它的两个子节点都是黑色的
        - 从任意一个结点到其每个叶子的所有路径都包含着相同数目的黑色结点
      - 这些约束强制了红黑树的关键性质：从根到叶子的最长的可能路径不多于最短的可能路径的两倍长。结果就是这棵树大致上是平衡的，因为插入、删除和查找某个值得最坏情况时间都要求与树的高度成比例，这个高度理论上限允许红黑树只在最坏情况下都是高效的。
  - 图形结构
    - 有向图
    - 无向图
  - 散列表
    - 主要用的是hash