**常见的数据结构**

1. **排序**
2. **选择排序**

思想：对于给定的一个数组，经过第一轮排序得到了最小的数，然后将其与第一个位置进行交换。接着对剩下的记录按照之前的方式比较。

复杂度：O（n^2）

1. 插入排序

思想：对于给定的一个数组，以第一个数作为有序序列，然后从第二个数开始，按照与前面有序序列的大小关系插入到对应的位置。

复杂度：最好O（n），最坏O（n2）

1. 冒泡排序

思想：对于给定的n个记录，从第一个记录开始依次对相邻的两个记录进行比较，当前面的数大于后面的数时候进行交换，所以第一轮就找到了最大的数或者最小的数。然后在对前面的n-1个数按照相同的方式执行，一直到排序完成。

复杂度：最好O（n），最坏O（n2）

1. 归并排序

思想：堆分为最大堆和最小堆。其实就是完全二叉树。最大堆的要求就是节点不能小于其孩子，最小堆的要求是节点不能小于其左右孩子。

1.首先是将待排序的关键序列构建成大顶堆，此堆为初始化的无序区。

2.将堆顶元素R1和最后一个元素进行交换；此时得到了新的无序区R1和刚交换的得到有序区Rn。

3.由于交换后的元素破坏了堆的性质，现在对当前的无序区进行调整为新堆。然后将R1和最后一个元素进行交换，得到了新的无序区和新的有序区。不断的重复，直到有序区的元素为n -1.

完全二叉树：对于一个树高为h的二叉树，如果其第0层至第h-1层的节点都满。如果最下面一层节点不满，则所有的节点在左边的连续排列，空位都在右边。这样的二叉树就是一棵完全二叉树

现在可能问的就是怎么写代码调整大根堆？

我们首先求得到根据数组得长度一半 – 1；通过for去遍历这些节点的值。替换

等等

复杂度: 假设该操作的时间是T(n)，则最大的T(n) = T(2/3 n) + O(1)，这里的O(1)指的是根节点及其左右子节点的操作.3个节点中取最大值或者最小值只需要2/3次。

1. 快排

思想：首先根据第一个数，安排排序将其分为两份，保证其左边得都比其小，右边的都比其大。最后在对左右两边分别按照这种思路来进行排序。在刚开始的时候i= low，j=high；最后一旦high和low的值相等或者i大于j时候停止递归。

复杂度: O(nlog(n))

优化：如果第一个数是极值的化可能就会出现一边倒的情况；所以在刚开始的时候我们可以随机取3个数；取其中的中间数作为开的选项。

2． 二叉树

1) 前序遍历

先访问根节点，先前序遍历其左子树，再前序遍历右子树。

递归思想：在函数中进行非空判断，如果为空就返回；首先先输出节点，所以第一输出的就是根节点。然后使用递归按照先后顺序指向节点的左节点和右边节点。

非递归的实现思路如下：

对于任一节点P，

1）输出节点P，然后将其入栈，再看P的左孩子是否为空；

2）若P的左孩子不为空，则置P的左孩子为当前节点，重复1）的操作；

3）若P的左孩子为空，则将栈顶节点出栈，但不输出，并将出栈节点的右孩子置为当前节点，看其是否为空；

4）若不为空，则循环至1）操作；

5）如果为空，则继续出栈，但不输出，同时将出栈节点的右孩子置为当前节点，看其是否为空，重复4）和5）操作；

1. 直到当前节点P为NULL并且栈空，遍历结束。

2）中序遍历

思想：先从根节点开始，先遍历根节点的左子树，然后是访问时根节点。最后遍历根节点的右子树。使用递归按照先后顺序指向节点的左节点和右边节点。两者中间穿插节点的输出，

非递归思想：

对于任一节点P，

1）若P的左孩子不为空，则将P入栈并将P的左孩子置为当前节点，然后再对当前节点进行相同的处理；

2）若P的左孩子为空，则输出P节点，而后将P的右孩子置为当前节点，看其是否为空；

3）若不为空，则重复1）和2）的操作；

4）若为空，则执行出栈操作，输出栈顶节点，并将出栈的节点的右孩子置为当前节点，看起是否为空，重复3）和4）的操作；

5）直到当前节点P为NULL并且栈为空，则遍历结束

3）后续遍历

先遍历左子树，然后遍历右子树，最后输出根节点。

非递归思想 ：

对于任一节点P，

1）先将节点P入栈；

2）若P不存在左孩子和右孩子，或者P存在左孩子或右孩子，但左右孩子已经被输出，则可以直接输出节点P，并将其出栈，将出栈节点P标记为上一个输出的节点，再将此时的栈顶结点设为当前节点；

3）若不满足2）中的条件，则将P的右孩子和左孩子依次入栈，当前节点重新置为栈顶结点，之后重复操作2）；

4）直到栈空，遍历结束。

层序遍历

思想：一层一层扫描下来遍历

我们可以使用一个队列来进行操作，先把根节点放入队列中，出队列的时候顺便将其左右孩子入队；循环队列进出对，当队列为空的时候完成层序遍历。

二叉树中两个节点的最大距离？

分别找到左右两边的最深节点，然后加起来 -1 。

1）从根节点开始判断其左右节点是否为空，然后递归。在递归的过程中每次比较节点左右子树的最大值；取最大值作为当前方向的值，然后每次加1.一直到为空为止。最后将左右两边最大值相加即可。

1. 如何判断链表是否为空？

快慢指针即可

1. 如何实现两个栈形成一个队列？

我们来看一下比较重要的push和pop操作，可以这么来想，当有数据要入队的时候，我们就让它压入stack1，要进行pop操作的时候，我们就把stack1里面的数据全部压入stack2中，然后对stack进行一次pop操作就可以了