一.链表

1.逆序打印单向链表

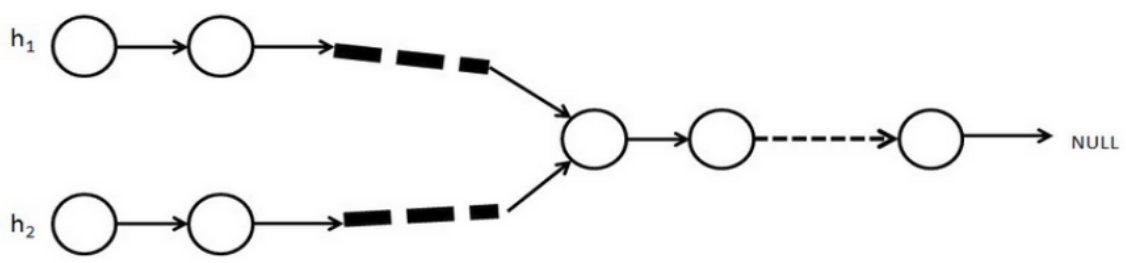
- 利用数组或者栈，保存链表中的元素，再逆序打印数组或者出栈
- 新建一个链表，新建节点，保存链表中的值，头插入新的链表中。
- 递归打印

2.倒置一个单向链表

- 链表中的元素的地址压栈，再出栈。
- 指针数组记录元素的地址，倒序赋值给链表
- 从链表中挨个取出，再头添加
- 三个指针从头到尾遍历整个链表

```
List *p1 = NULL;//被指向
    List *p2 = pHead;//指向
    List *p3 = pHead->pNext;//被断开位置
    while(p3 != NULL)
    {
        //改变指向
        p2->pNext = p1;
        //指针移动
        p1 = p2;
        p2 = p3;
        p3 = p3->pNext;
    }
    //链入最后一个节点
    p2->pNext = p1;
```

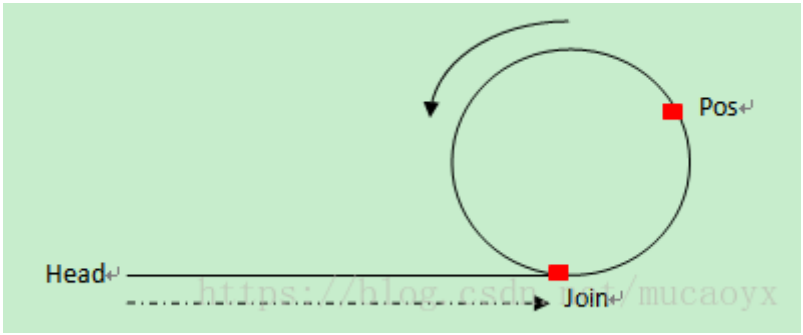
3.如何检测两个普通的单向链表是否有交点?



参考：https://blog.csdn.net/fengxinlinux/article/details/78885764

- 长的链表先走差值，然后再一起走
- 两个链表的地址压栈，再出栈，找到两个出栈的元素不同的前一个，就是相交节点
- 数组倒序比较，原理同上

4.如何检测单向链表是否有环？如果有，找到环的入口点?



参考：https://blog.csdn.net/u010983881/article/details/78896293

https://www.cnblogs.com/dancingrain/p/3405197.html

检测：

- 快慢指针的方法
- 穷举遍历，找到第一个遍历到的重复的地址
- 哈希缓存

找到入环的节点：

- 快慢指针的方法，相遇点断开，变成问题3解决
- 环外和环内遍历比较
- 若环的长度是x，两个指针p1和p2从头开始走，p1先走x，p1和p2再一起走x，p2

的点就是入口点。

二.栈和队列

1.栈(FILO)

(1).栈的操作：

函数	操作	函数	操作
init()	初始化	pop()	出栈
push()	入栈	clear()	清空栈
distory()	销毁栈	GetCount()	获得栈的大小
GetTop()	获取栈顶元素	IsEmpty()	判断栈是否为空

(2).栈的应用

- 递归
- 斐波那契数列(解法：递归，非递归等)
- 后缀表达式(逆波兰表示法)

2.队列(FIFI)

(1).队列的一种特殊形式：循环队列(循环队列可能会造成假溢出)

(2).队列的操作：

init() Pop() Push() IsEmpty()

(3).应用：

- 用两个栈实现队列：先入一个栈 再入另一个栈 再出栈

- 用两个队列实现栈：入栈：入任意一个非空的队列 出栈：将队尾之前元素全部出队 只留下队尾元素让队尾元素出栈

三.树

1.树的种类：满二叉树、完全二叉树(只有最后一层有空缺， 而且空缺位置从左到右)、BST二叉搜索树(左子树的节点值都比父节点小 右子树的节点值都比父节点大)、平衡二叉树(左右子树高度差不能超过1， AVL树)

2.树的性质：

(1).k层二叉树 最多有 2^(k) -1 个节点

(2).k层二叉树 最多有2^(k-1)个叶子节点

(3).度为0的节点比度为2的节点数多1个

(4).完全二叉树中度为1 的个数最多有1个

(5).N个节点的完全二叉树的层数为log2N向下取整再+1层

(6).一个n个节点的完全二叉树从上到下、从左到右从1开始编号 第i个节点的左子树为2*i，右子树为2*i+ 1，父节点的编号为1——n/2；如果从0开始编号，第i个节点的左子树为2*i+1， 右子树为2*i+2，父节点的编号为0——(n/2 - 1).

3.树的遍历

(1).递归的方法

深度遍历(前序遍历、中序遍历、后序遍历)， 广度遍历

https://blog.csdn.net/fansongy/article/details/6798278

(2).非递归遍历

4.排序二叉树

(1).排序二叉树的构建

(2).排序二叉树节点的删除

左的最右或者右的最左

(3).把BST变成有序的双向链表

(4).BST树的旋转

左的左加加点 右旋

右的右加节点 左旋

左的右 双旋

右的左 双旋

红黑树：

(1).红黑树的5个性质

树的每个节点要么是红的，要么是黑的。

树的根节点必须是黑的

树种不允许两个节点互为父子关系

树中的终端节点都是黑的，空节点也是黑的，称为Nil节点

从任意节点出发到所有可能到达的各个终端的各个路径上，黑节点的数目必须完全相同

(2).红黑树的插入删除查找的复杂度都是log2N(N是节点的个数)

(3).红黑树的创建：

https://www.jianshu.com/p/e136ec79235c

(4).红黑树节点的删除

https://www.cnblogs.com/qingergege/p/7351659.html

哈夫曼树：

(1).哈夫曼树的构建过程

四. 图

(1). 图的遍历：

深度优先和广度优先

(2).迪杰斯特拉和贪心

(3).克鲁斯卡尔和普利姆算法生成最小树

五.排序

(1).冒泡排序

(2).选择排序

(3).插入排序

(4).计数排序

(5).快速排序：挖坑填补法和区间分割法

(6).希尔排序

(7).归并排序

(8).堆排序

(9).桶排序

(10).基数排序

六.查找

(1).二分

(2).HASH查找

七.String查找

（1）KMP

（2）Sunday算法.