名词翻译

Anagram:异位词
Palindrome:回文序列
Sibling node:兄弟结点
Heuristic:启发式

代码题

栈和队列

- 括号匹配
- 进制转换：分别使用栈和使用递归完成。辗转相除法。
- 中缀表达式转后缀表达式：符号栈的情况(优先级小于等于当前栈顶运算符则出栈)
- 计算后缀表达式的值：栈。
- 传土豆Hot potato：队列。
- 回文序列检测器：deque。
- 无序和有序链表的insert，remove和search：双指针。

哈希表

> 这里没有代码题

递归

- 谢尔平斯基三角形:
  基本条件,递归次数.
- 汉诺塔问题
- 硬币找零问题

排序

- 冒泡排序
- 选择排序
- 插入排序:

> 最优情况复杂度：当**列表已经有序的情况**下实现

- 谢尔排序
- 归并排序：时间复杂度和空间复杂度

> 时间复杂度:**稳定不变**
> 空间复杂度:$O(n)$

- 快速排序:

> 最坏情况:**列表有序**,例如 `[1,2,3,4,5]` 不断选择最左边的元素作为 `pivot`

树

- 解析树构造过程
- 二叉树遍历和层次遍历

二叉排序树

- 二叉搜索树的删除操作

平衡二叉树

- **平衡因子的递归调整**:

> 父结点平衡因子为0不用调整
> 否则不断向上**递归调整**祖先结点的平衡因子.

- **重新平衡的过程**:翻转操作

> left heavy和right heavy情况
> left heavy+right heavy:**先调整子节点的形态,再调节失衡的父结点**.

图

- 邻接表和邻接矩阵的优缺点
- 词梯图问题：为什么BFS优于DFS?

> BFS一层一层逐步向外探索,可以保证在探索到最短路径后及时停止.
> DFS深入探索,回溯(backtracking)次数多.
> **词梯问题本质上是权重为1的最短路径问题**,当权重相等时,dijsktra算法等于BFS

- BFS和DFS遍历的问题:

> 三种颜色,**白色**：未探索;**黑色**:已经探索(已经加入其子结点到队列中),**灰色**:正在探索(被添加到队列中)

- BFS和DFS的时间复杂度:

> **取决于图的结构**,邻接表是$O(n+e)$,邻接矩阵是$O(n^2)$。

开放题和代码设计题

用栈或队列实现网页

> 两个栈：一个前进栈，一个后退栈。
> 前进操作：当前网页加入后退栈，从前进栈里取出前一个网页。
> 后退操作：当前网页加入前进栈，从后退栈里取出后一个网页。
> 新网页：**清空前进栈**，将当前网页压入后退栈。

中缀表达式转前缀表达式：

> **倒序遍历**中缀表达式,
> 执行**中缀表达式转后缀表达式**的操作
> **最后翻转**即可

计算前缀表达式的值：用栈实现。

循环队列：

> 注意栈顶指针指的是栈顶元素还是栈顶元素的下一个位置

判断队空和队满的区别

> 队空： `Q.front=Q.rear?`

队满：两种不同方式

空位法：

```
(Q.rear+1)mod size==Q.front mod size
```

标记位法:

push时设置 `Tag=1` ,Pop时设置 `Tag=0` .根据 `Tag&&Q.front==Q.rear?1:0` 判断队列满或者队空。

链队列：

链队列的优点,普通队列的缺点

> 普通队列的缺点：大小有限
> 链队列的优点：**大小无限**

链队列中加入新元素的位置：头插法还是尾插法?

反转列表

递归次数，基本条件

上台阶问题

> 动态规划: `dp[i]=dp[i-1]+dp[i-2]`
> 递归

硬币找零问题的优化

> **0-1背包问题**

树的遍历

使用栈实现树的前序、中序和后序遍历
栈起到了回溯的作用,并不像BFS中队列记录即将访问的元素.