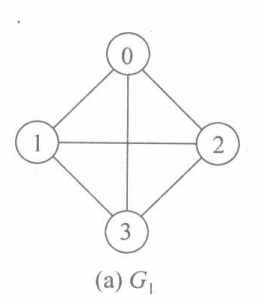
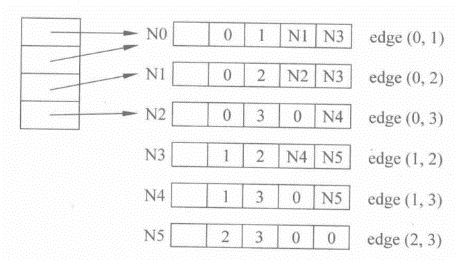
1. 分析一段代码的时间复杂度（log n、n^3、n^2、n）
2. 一个序列入住栈、队列的顺序
3. 链表的遍历、插入、删除
4. 完全二叉树（下标为0不用，从1开始存）
5. 二叉树和森林的相互转化
6. 二叉搜索树
7. 哈夫曼树、哈夫曼编码（Optimal Search Tree最优搜索树）概率越小的越往下
8. AOV（顶点表示活动）选出所有没有前置边的点，再删除连接的边
9. hashing（冲突处理、开放地址）【装填因子（loading factor） = 记录存储数/总容量】相除后溢出则下移

10、binary search（给出一个有序数组和一个数据，得出比较次数）

1. 树、森林的遍历：前序（中左右），中序（左中右），后序（左右中），层次遍历

2、堆的初始化（下标为0不用，从下到上，从小树到大树）

3、败者树（败者在节点，仅最后一个不是）

4、邻接表、邻接多表（操作、遍历）

5、一个无向图通过Prim（选点构建）或Kruskal（选边）方法构建最小代价生成树

DFS（深度优先，一直搜索直到没有后退stack）、BFS（广度优先，一个点所有边遍历queue）

6、快速排序（平均计算时间为n \* log n 使用前中后三值比较取中间值进行的排序【median-of-three】）、

7、KMP失败函数（求一个字符串各个位置f的值）

8、AVL树（Adelson-Velsky-Landis Tree），平衡查找树（定义及构造【标注平衡因子（题目标注是左减右或右减左）、旋转类型（LL型、RL型）】）

9、m阶B树（定义、构造、删除）关键字个数最大为（m-1）个

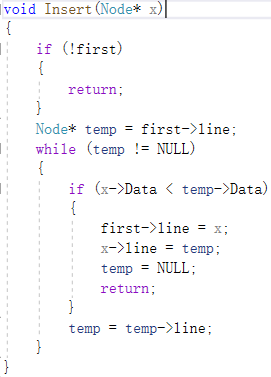
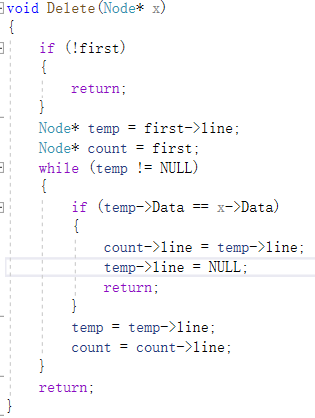
所有限制都为向上取整，排满后向父亲节点移动（原则尽量动右边即向上取整）

删除：借 > 合并，左 > 右

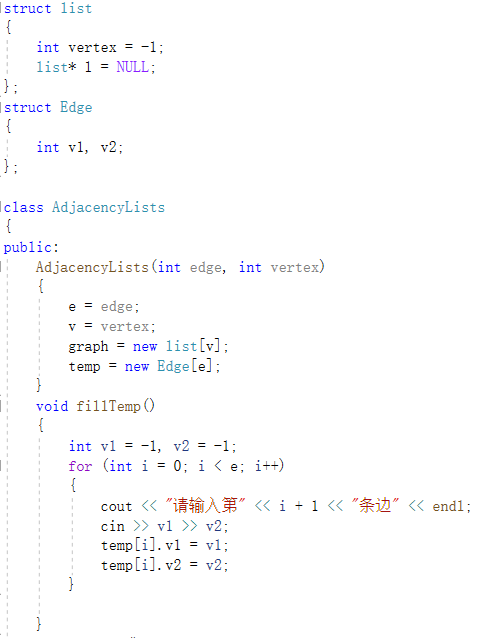
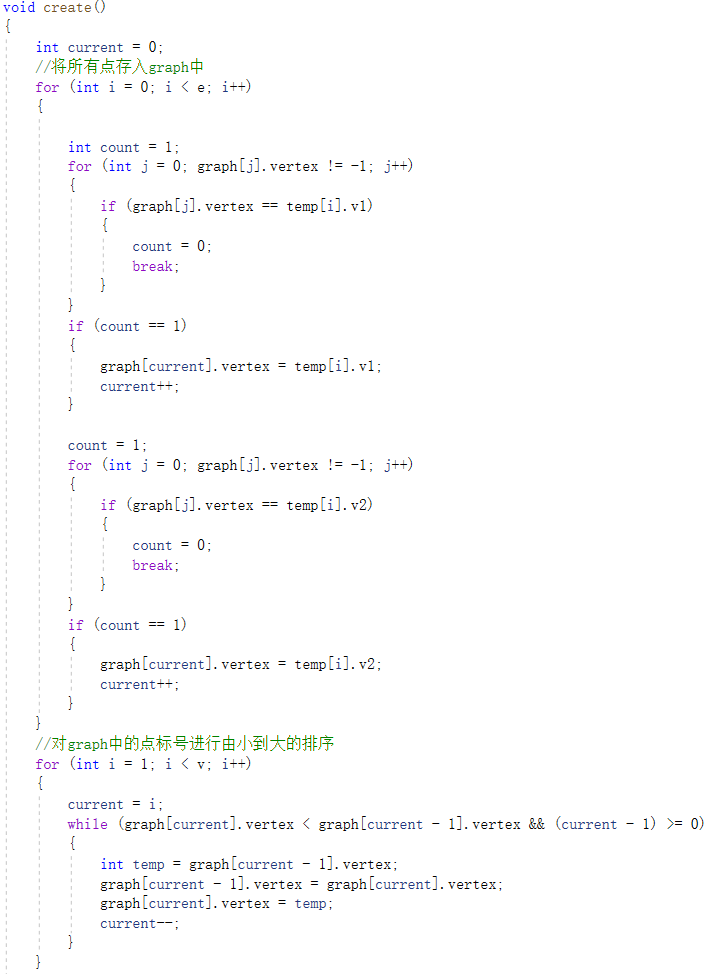
10、归并排序（迭代两两一组，递归一半一半奇数则中间归左）

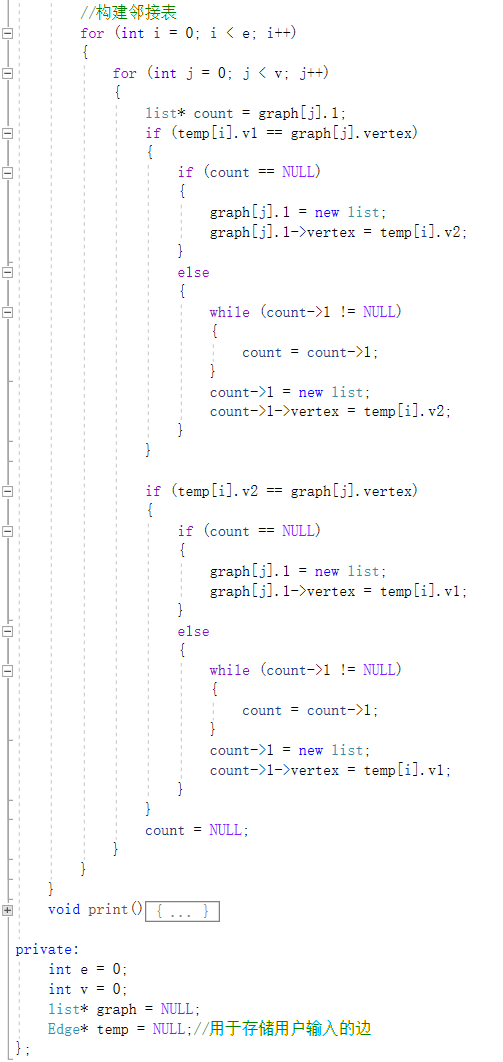
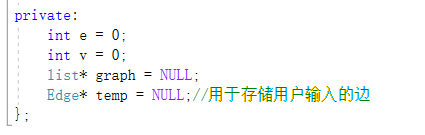
11、堆排序（写出每一步的数组状态）先初始化再逐一调出（升序用最大堆，降序用最小堆）

编程题：

1、单链表（插入、删除）

2、二叉树与森林的相互转化

3、图（存储及遍历）



**注意：1、写出基本思路（中文描述）**

**2、编写具体代码**

**3、分析程序时间复杂度、空间复杂度**

英文：merge（归并）、iteration（迭代）、recursive（递归）、preorder（前序）、inorder（后序）、postorder（后序）