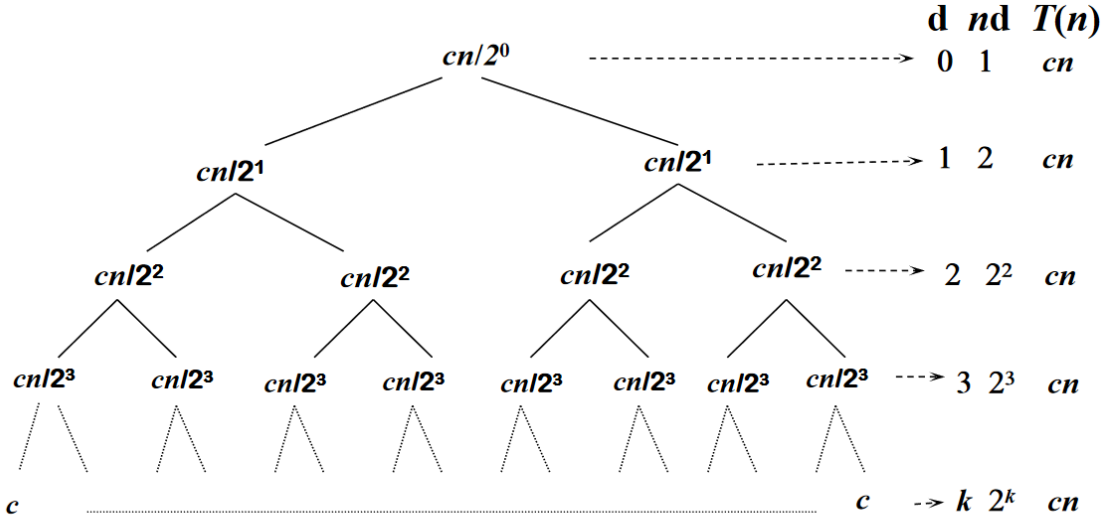
算法设计与分析

一、分治法

- 主方法：包含主方法定义以及例题分析

$$T(n)=\begin{cases}\Theta(n^{\log_b a}) & \text{if } f(n)\in O(n^{\log_b a-\varepsilon}),\varepsilon>0 \quad (1)\\ \Theta(n^{\log_b a}\lg^{k+1} n) & \text{if } f(n)\in \Theta(n^{\log_b a}\lg^{k} n),k\geq 0 \quad (2)\\ \Theta(f(n)) & \text{if } f(n)\in \Omega(n^{\log_b a+\varepsilon}),\varepsilon>0 \text{ and} \quad (3)\\ & af(n/b)\leq cf(n) \text{ for some}\\ & c<1 \text{ and all sufficiently large } n\end{cases}$$

- 代入法：主要靠经验，非重点
- 递归树：要掌握如何画以及如何求解，对于一般的递归树题目，大部分是递归树呈现对称形态的比较容易计算，去年的算法考试考了一道递归树非对称形态的，与这题类似，题目要求画出该递归树并求解算法复杂度，可以自己尝试画一下

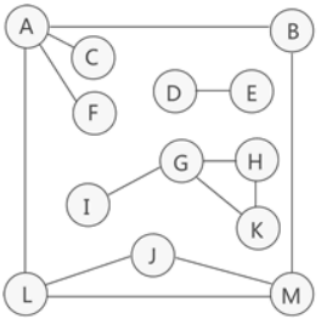


$$T(n)=(k+1)(cn)=(\lg n+1)(cn)=\Theta(n\lg n)$$

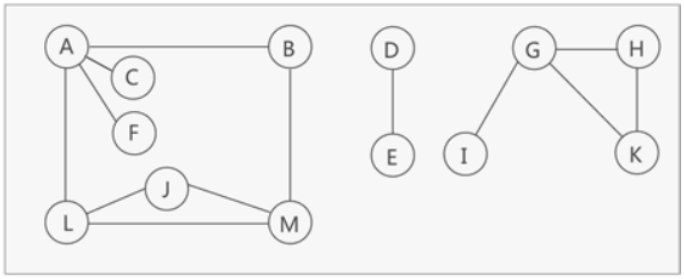
二、图论

- 图的定义：有向图、无向图、节点的度、入度、出度、简单图、简单路径、环、有环图、无环图.....
  - 有向图：边有起点与终点
  - 无向图：边是节点对，不区分起点与终点
  - 节点的度：节点出度与入度之和，即连接该节点边的条数；
  - 入度：有向图中连向该节点边的条数
  - 出度：有向图中从该节点连出的边的条数

- 简单图：没有多重边，没有自环
- 简单路径：对于一条由连续边与节点组成的路径，没有经过重复的节点
- 环：一条路径的起点与终点一致
- 有环图：图中包含环
- 无环图：图中不包含环
- s-t连通，即是否存在以节点s和节点t为起点与终点的路径
- 连通图：对于一个无向图，任意两个节点之间都存在一条路径连接



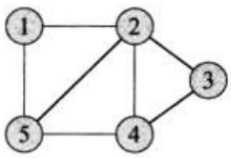
a) 非连通图

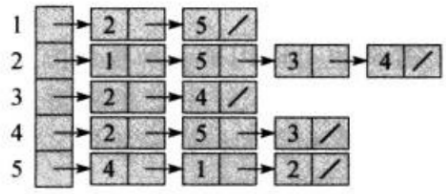


b) 连通分量

- 强连通图：对于一个有向图，任意2个节点之间都存在一条有向路径连接
- 稀疏图：$|E| \approx |V|$
- 稠密图：$|E| \approx |V|^2$
- 完全图：对于一个有向或者无向图，任意两个节点之间都有边邻接（对于有向图需要两个方向的边）

- 图的存储：
  - 图的邻接链表：优点：鲁棒性高，方便修改实现多种图的表示；缺点：无法快速判断一条边是否存在于该图中，只能对一个节点的一条链进行搜索；存储空间：$\Theta(v+2e)$
  - 图的邻接矩阵：优点：方便查找边；缺点：空间开销大；存储空间：$\Theta(v^2)$，其中当图为无向图时，可以只存储邻接矩阵的一半





	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

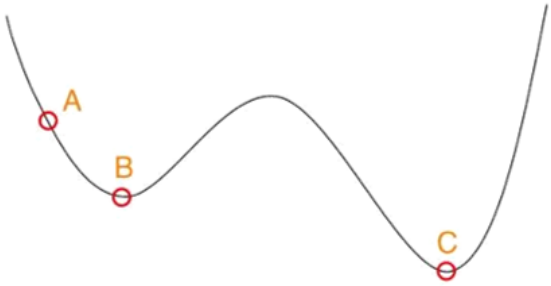
- 图的算法：
  - 广度优先搜索BFS
  - 深度优先搜索DFS
  - 拓扑排序
  - 最小生成树：
    - Kruskal算法
    - Prim算法
    - 广度优先搜索BFS
    - 深度优先搜索DFS
  - 并查集
- 最大流问题：
  - Ford-Fulkerson方法：会让画出最大流网络以及各个节点的变化表
  - Edmonds-Karp算法：Ford-Fulkerson方法的实现
  - Dinic算法及优化：主要在实验中优化，非课程内容
  - 最短增益路径算法

三、动态规划

- 与分治差别：动态规划的子问题只求解一次，而分治法有时子问题会被重复计算多次
- 与分治相同：原问题分解成若干个子问题，先求解子问题，由子问题的解构造原问题的解
- 适用条件：
  - 最优子结构：证明题
  - 重叠子问题
- 实现方式：
  - 自底向上法：子问题按规模从小到大进行求解。当求解子问题时，其更小子问题已求解，并已保存，可直接使用
  - 带备忘录的自顶向下法：按自然递归形式调用过程，保存每个子问题的解。当求解一个子问题时，首先判断是否存在子问题的解，若存在则直接使用；否则，按递归形式求解该子问题
- 特点：以空间换时间。动态规划是较为简单的算法，考试会考察写出某一问题的动态规划表达式，可以将书上的例题搞清楚一点：
  - 钢条切割问题
  - 矩阵乘法问题
  - 最长公共子序列问题
  - 最优二叉搜索树问题
  - PPT上的01背包问题

四、贪心算法

- 贪心算法简单理解：每一步都选择当前最优的方法，易陷入局部最优

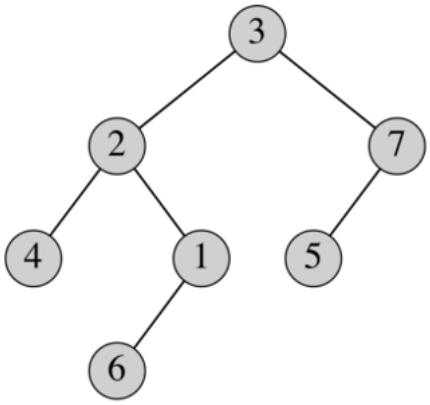


- 特点：
  - 可行的：即必须满足问题的约束
  - 局部最优：前步骤中所有可行选择中局部最佳选择
  - 贪婪不可撤销：选择一旦做出，后面步骤无法更改
  - 全局最优解：需要满足贪心选择性质。简单理解就是每次进行贪心选择皆是全局最优解中的一个局部最优解，举个例子，你需要经过十个地方，每个地方有面额大小不一的钱币，每个地方只能拿一个钱币，那么全局最优解便是拿每个地方面值最大的钱币，局部最优解便是在当前地方拿面值最大的钱币。显然，每次进行贪心选择皆是全局最优解中的一个局部最优解，因此此问题满足贪心选择性质，最终得到的解是全局最优解
- 哈夫曼编码：节省空间的计算

五、堆排序

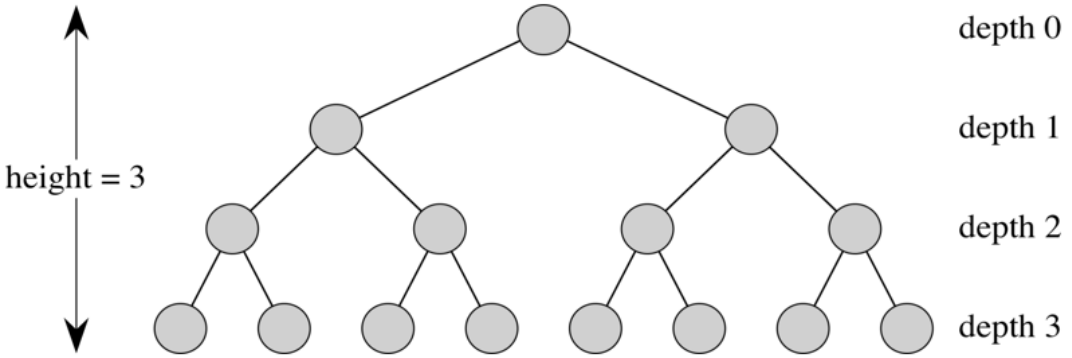
- 树：

- 二叉树：
  - 一个结点的深度 = 从这个结点到根结点的简单路径上边的数目
  - 一颗树T的深度 = 树中所有结点最大的深度
  - 一个结点的高度 = 从该结点到一个叶子结点的最长简单路径的边数
  - 一棵树 T 的高度 = 树的根结点的高度 = 树的深度

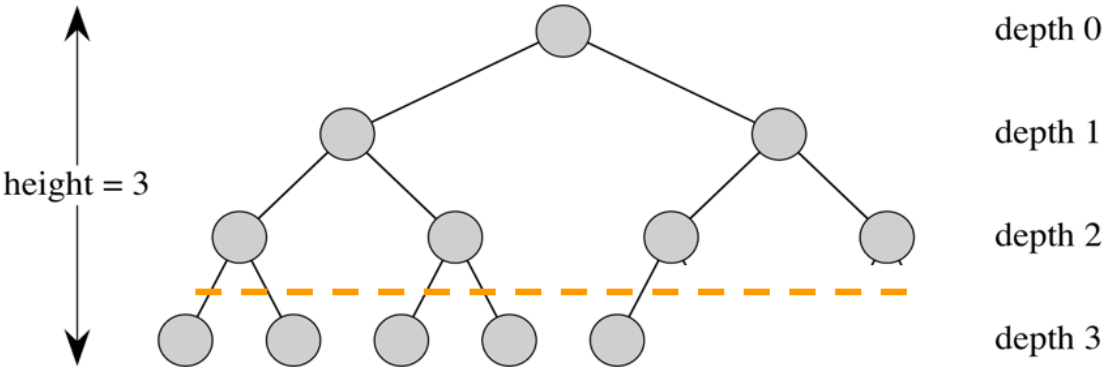


结点2的深度 = 1
树T 的深度 = 3
结点2的高度 = 2
树T 的高度 = 3

- 完全二叉树：所有叶子结点在同一深度，而且每个非叶节点都有两个孩子结点的二叉树

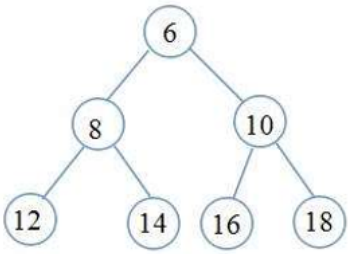


- 近似完全二叉树：深度为 d-1 时是完全二叉树，深度为 d 的结点都在靠左部分

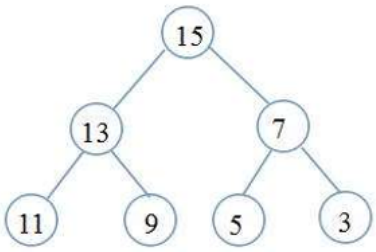


- 二叉树节点与深度转换计算：会考，这个自己整理一下比看网上资源好

- 堆：一棵近似完全二叉树
  - 最小堆性质：每个结点存储的数值小于 ≤ 该结点的孩子节点存储的数值。最小值存储在根结点
  - 最大堆性质：每个结点存储的数值大于 ≥ 该结点的孩子节点存储的数值。最大值存储在根结点

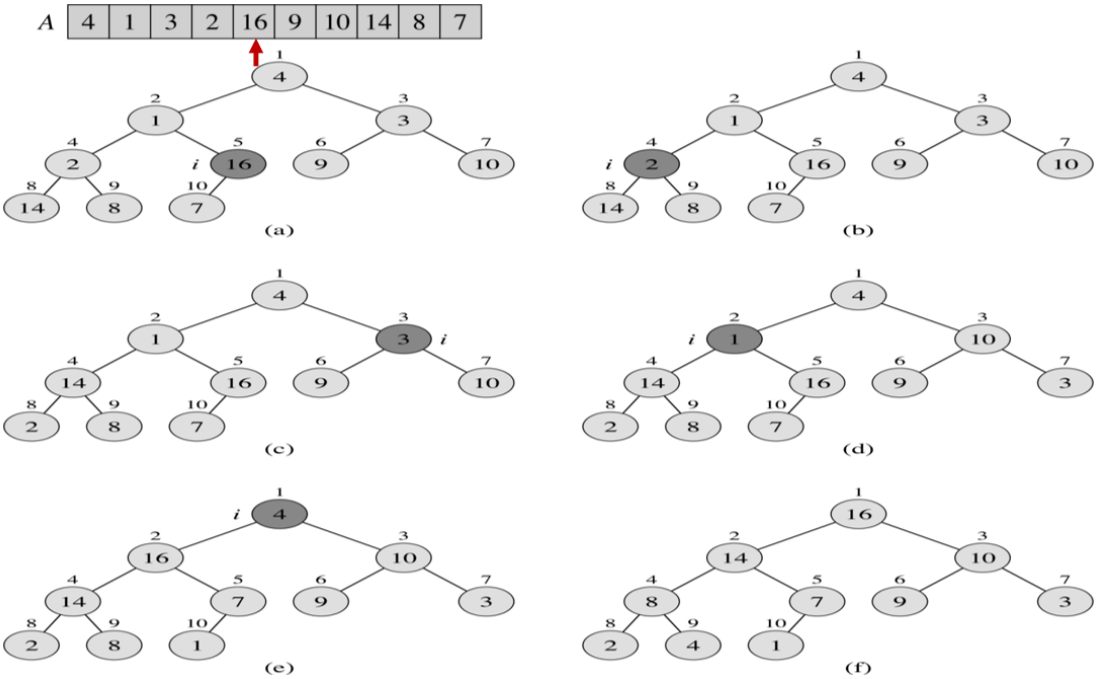


最小堆



最大堆

- 数组实现堆排序：推导时注意下标关系，或是直接画出堆结构进行逐步推导，考过给定一个初始数组，让写出其最大堆或最小堆的最终形式的题



- 优先队列与堆