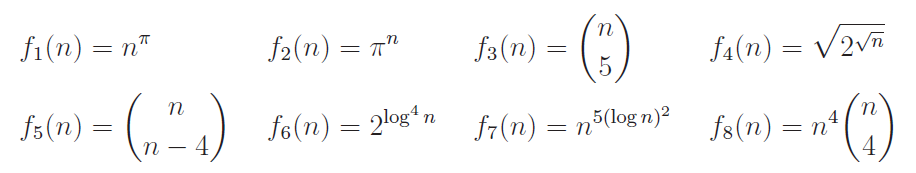
1. 时间复杂度
2. 将以下函数按照增长速度排序：

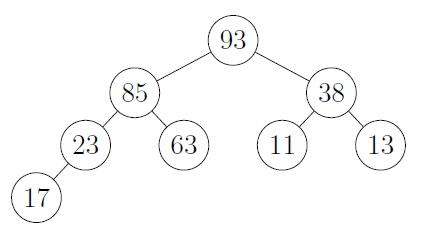


1. 求解以下分治算法分析是产生的递归关系：

C:\Users\bzhan\AppData\Local\Temp\1636119215(1).png

1. 堆算法

考虑以下堆：



说明这个堆如何使用数组表达。如果这个堆里的最大元素被删除，剩下的堆的数组表达是什么？

1. 最长交替子序列

我们称一个子序列为交替序列，如果每个相邻的都满足或。也就是说，如果，则，反之亦然。给定一个序列，我们想要找到这个序列的最长的交替子序列。

例如，如果初始序列是：

这个序列的最长交替子序列的长度为5，由组成。

使用动态规划设计寻找最长交替子序列的算法。我们定义子问题，其中，是一个布尔值。如果为真，则代表最长的终止于的交替子序列，其中最后一步是上升的。如果为假，代表最长的终止于的交替子序列，其中最后一步是下降的。如果子序列的长度为1，则我们定义它既是上升的也是下降的。例如，因为子序列的长度为4，并且最后一步上升。，因为子序列的长度为3，并且最后一步下降。

1. 对于以上序列，计算，对于每个和，写入以下表中：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

1. 写出的递归关系。
2. 写出递归关系的基本情况（）。
3. 如果采用自底向上的计算方式，给出一个合适的计算顺序。
4. 如果计算了所有的，如何计算最长交替子序列的长度？
5. 将以上步骤放到一起，给出计算最长交替子序列长度的算法，使用伪代码表示。分析算法的时间复杂度。
6. 括号小游戏

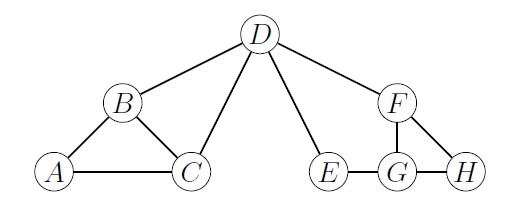
给定一个不带括号的，由加法和乘法组成的表达式，找出如何添加括号，使得表达式的取值最大。

例如，如果提供的表达式是，则应该添加括号为，结果为，而不是，结果为。再比如，如果提供的表达式是，则应该添加括号为，结果为，而不是，结果为。

使用动态规划设计多项式时间的算法，在给定表达式之后，找到最好的添加括号的方法。假设输入的格式为，其中每个是一个数字，每个是加号或乘号。分析算法的时间复杂度。

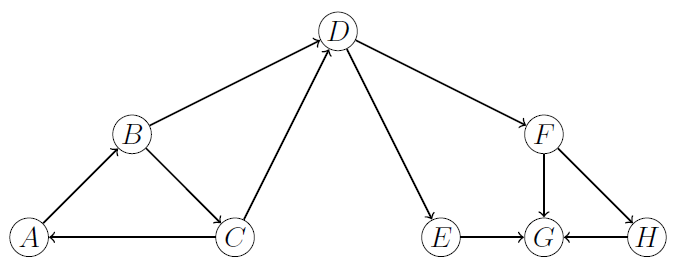
1. 图的遍历

考虑以下图：

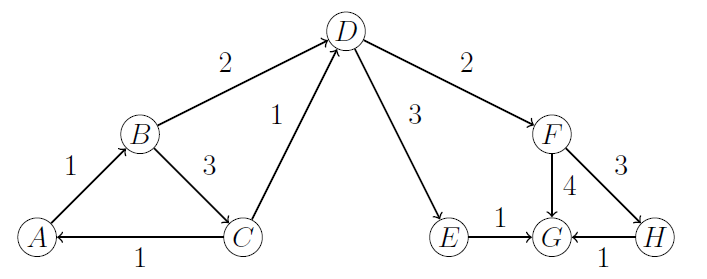


假设该图使用邻接列表(adjacency list)表示，每个节点的相邻节点按照字母顺序排序（例如，的相邻节点表示为）。

1. 假设我们使用广度优先搜索(breadth-first search)寻找从到的路径，写出该算法得到的路径。
2. 假设我们使用深度优先搜索(depth-first search)寻找从到的路径，写出该算法得到的路径。
3. 假设我们从A开始采用深度优先搜索，将每条边标记为树边(tree edge, T)、后向边(back edge, B)、前向边(forward edge, F)、和横向边(cross edge, C)。回顾这些概念的定义：一条边是
   * 树边如果在深度优先搜索产生的树中。
   * 后向边如果在深度优先树中是的祖先。
   * 前向边如果在深度优先数中是的后代。
   * 横向边如果以上都不成立。
4. 重复问题(c)，但使用以下的有向图：



1. 现在假设图的每条边有一个距离如下：



使用Dijkstra算法寻找到的最短路径，给出Dijkstra算法执行中顶点从优先队列中移出的顺序。

1. 婚礼计划

在婚礼上，你需要为到来的客人安排座位。

1. 假设你知道客人之间谁互相认识。如果认识，则必然认识。你需要将客人安排到不同的桌子上，使得每个桌子的客人或者互相认识，或者通过同一个桌子的人认识。例如，如果认识，并且认识，则可以坐在同一个桌子上。设计高效的算法，在给定客人之间谁互相认识的信息后，返回至少需要多少张桌子才能满足这个要求。分析该算法的时间复杂度。
2. 现在考虑另一个场景：假设只有两张桌子，另外你知道有些客人不喜欢对方。如果不喜欢，则必然不喜欢。你的目标是将客人分配到两张桌子上，使得同一桌子上不存在两个客人不喜欢对方。例如，如果客人间不喜欢的关系如图(a)所示，则可以将A,C安排在一个桌子，B,D,E安排在另一个桌子。如果客人间的不喜欢关系如图(b)所示，则这个目标无法达到。设计高效的算法，在给定客人间的不喜欢关系之后，返回目标是否可能达到。分析算法的时间复杂度。

