**1.** 将下列两组函数按如下方式排序：如果，则排在前；当且仅当时，函数和函数之间排序表示为{,}。如无特殊说明，。例：如果函数为

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

则正确的排序应为(, {, })或(, {, })。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A组 |  |  |  |  |  |
| B组 |  |  |  |  |  |

提示：可以通过判断和、或和之间的关系来判断函数和函数间关系。

**2.** 在操作系统Unix中，一个**文件**可以被看作是有序字符串的集合，该文件内的第 个字符串被称为该文件的第 行。对于文件可以进行如下简单的**变换**操作：

* 在文件中插入新的一行
* 从文件中删除已有的一行
* 交换文件中相邻的两行

其中，交换两行的操作比较便宜（因为他们已经存在文件中），而插入和删除相对较贵。

在Unix中，存在一个文件比较功能diff(A,B)，它可以通过将一系列变换操作应用在文件A上使其转换为文件B。假设文件A和B都有且仅有行，**变换中每一行最多仅可以进行一次交换操作**，且每次交换的两行需要在文件A和B中分别相邻。请给出非交换操作次数最少的diff实现算法。

1. 给出求解此问题的动态规划递归关系。
2. 用循环实现基于该递归公式构造的动态规划算法伪代码。
3. 分析该伪代码的时间复杂度。

**# dp[i][j]：将 A[1..i] 转换为 B[1..j] 所需的最少 非交换操作 数**

**Initialize dp array of size (n+1) x (n+1)**

**dp[0][0] = 0**

**for i in [1..n]:**

**dp[i][0] = i # 全部删除**

**for j in [1..n]:**

**dp[0][j] = j # 全部插入**

**for i in [1..n]:**

**for j in [1..n]:**

**if A[i] == B[j]:**

**dp[i][j] = dp[i-1][j-1] # 行内容相同，无需插/删**

**else:**

**dp[i][j] = min(**

**dp[i-1][j] + 1, # 删 A[i]**

**dp[i][j-1] + 1 # 插 B[j]**

**)**

**# 尝试“转置”(相邻交换)**

**if i >= 2 and j >= 2:**

**if (A[i] == B[j-1]) and (A[i-1] == B[j]):**

**# 这对行在 A、B 中都是相邻且完全相反的顺序**

**dp[i][j] = min(**

**dp[i][j],**

**dp[i-2][j-2] # 通过一次“免费交换”就能对齐**

**)**

**3.** 你正在帮助一组民族学家分析他们通过访谈一个村庄的成员收集到的一些口述历史资料，以了解过去两百年中生活在那里的人们的生活情况。通过访谈，您了解到一组人的情况，这些人现在都已去世，我们将他们分别记为 。民族学家收集了关于这些人寿命的若干传言如下：

* 在 出生前就死了
* 和 在某个时期都活着

然而，民族学家们并不确定他们的事实是否正确；他们的记忆力不太好，而且所有这些信息都是口口相传的。因此，他们希望你能确定他们收集到的数据是否至少具有内部一致性，即是否存在一组人，他们所了解到的所有事实都同时成立。描述并分析一个算法来回答民族学家的问题。你的算法要么输出与所有已陈述事实一致的可能的出生和死亡日期，要么正确地报告不存在这样的日期。

(提示：构建一个二元有向图，其中有 个节点，分别代表 个人的出生日期和死亡日期。当我们有一个事实表明前者先于后者时，从一个节点到另一个节点画边。显然，一个人的出生日期先于他/她的死亡日期，所以这就是一条边。如果有两个人声称在同一时间活着，这意味着什么？）

**4.** 给定任意n件物品和一个背包，物品的重量为，价值为，背包的重量限制为B，其中、和B均为正整数。

请给出一个PTAS，其可在不超过重量限制的条件下可装入价值最大。

**5.** 假设你开发了一种能开又能飞的新型航空汽车带人们游览城市。为了优化游览路线，你构建了一个加权有向图 ，表示在不同城市地点（由V中的顶点表示）之间驾车或乘飞机的最佳时间，同时由于A城市具有独特的的历史文化，所以这些地点之间的最佳访问次序为。你的目标是找到G中依次访问的最短路径。（中间可能还要访问其他顶点：即路径必须有作为子序列；路径允许访问一个顶点不止一次，例如，是合法的）