13 双序列动态规划

笔记本: DP Note
创建时间: 11/4/2019 12:27 AM **更新时间:** 11/4/2019 12:46 AM
作者: tanziqi1756@outlook.com
URL: https://www.jiuzhang.com/course/36/dialog/#chapter-116

我们先来复习一下子串和子序列的概念: 对于一个字符串而言, 子串和子序列都是原字符串的一部分, 区别在于子串必须是连续的一段子区间的字符, 而子序列不要求连续. 可以认为, 子序列是原串删除一部分(可以是0个)字符之后得到的字符串. (相对顺序不能改变)

正确答案是 B C D E， 有60%的同学答对了，加油赶上他们!

正确答案是 A B C E F，有51%的同学超过了你，但是千万不要气馁。

解析: 根据定义, 所有的子串一定是子序列, 但子序列不一定是子串. 并且子序列不能改变字符的相对顺序, 所以 D 选项是不对的.

必做

多选题 **假定空串不算子序列也不算子串, 那么一个长度为 N 的字符串最多有多少个不同的子序列和子串呢?**

A 2^N个子序列

B 2^N-1个子序列

C (N+1)×N/2个子串

D (N+1)×N/2-1个子串

提交 我不会

正确答案是 B C，有29%的同学做对了这道题目哦，继续努力!

解析：子序列，就像数学里面的子集一样，有n个元素的集合，那么他的子集个数就是2^n的，但是空串不是子序列。所以这里的答案就是2^n - 1.而子串我们可以发现以第一个字符开头的子串个数有n个，第二个字符有n-1个，.....第n个字符有1个，所以总数就是n+(n-1)+...+1 = (n+1)×n/2,而空串是一个字符串的子串，所以不需要减一。

必做

单选题 **"jiuzhang" 和 "lijiang" 这两个字符串的的所有子串/子序列中, 最长的相同子串/子序列的长度是多少?**

A 最长相同的子串长度为 2, 最长相同的子序列长度为 3

B 最长相同的子串长度为 3, 最长相同的子序列长度为 3

C 最长相同的子串长度为 3, 最长相同的子序列长度为 5

D 最长相同的子串长度为 5, 最长相同的子序列长度为 3

提交 我不会

正确答案是 C，有55%的同学答对了，要加油了。

解析: 最长的相同子串是 "ang", 最长的相同子序列是 "jiang"

必做

单选题 **求两个字符串的最长的相同的子串或子序列，你认为哪个更适合使用动态规划的方法来分析求解?**

A 子串

B 子序列

C 都适合

提交 我不会

正确答案是 B ，有24%的同学答对了，加油赶上他们!

解析: 最长公共子序列 (Longest Common Subsequence) 是一个很经典的问题, 可以归属为双序列型动态规划问题, 马上侯老师就会为你讲解如何用动态规划的方法来分析求解.

必做

单选题 **在编辑距离这道题目中, 状态转移方程**: f[i][j] = min{f[i][j-1]+1, f[i-1][j-1]+1, f[i-1][j]+1, f[i-1][j-1]|A[i-1]=B[j-1]} **如果** A[i - 1] = B[j - 1] **那么** f[i][j] **必然可以取最后这一种决策. 也就是说, 我们的状态转移方程可以写成:**

```
if (A[i - 1] == B[j - 1]) {
        f[i][j] = f[i - 1][j - 1];
}
else {
        f[i][j] = min{f[i][j - 1] + 1,
                f[i - 1][j - 1] + 1,
                f[i - 1][j] + 1}
}
```

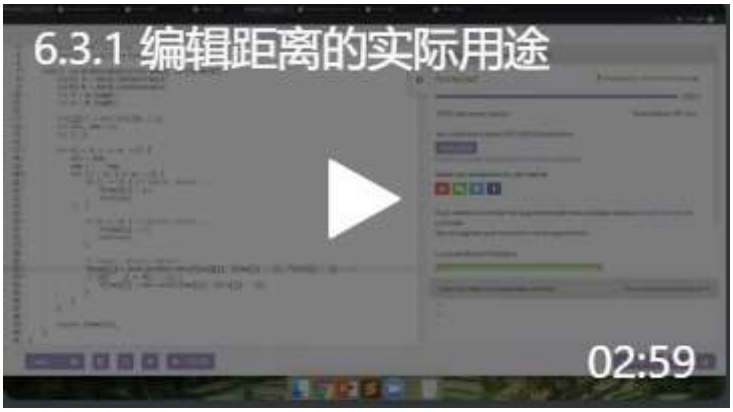
A 正确

B 错误

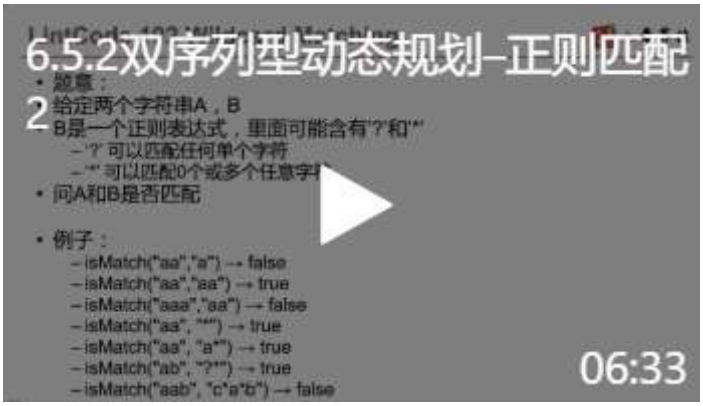
提交 我不会

正确答案是 A，有37%的同学做对了这道题目哦，继续努力!

解析: 这时前三个决策或许会与第四个决策一样, 但是绝对不会比第四个决策更优. 与 LCS 的情况类似, 同样是贪心.


6.3.1 编辑距离的实际用途
02:59


6.5.1 双序列型动态规划—正则匹配
• 题意：
• 给定两个字符串A，B
• B是一个正则表达式，里面可能含有'.'和'*'
– '.' 可以匹配任何单个字符
– '*' 可以匹配0个或多个前一个字符
• 问A和B是否匹配
• 例子：
– isMatch("aa","a") → false
– isMatch("aa","aa") → true
– isMatch("aaa","aa") → false
– isMatch("aa", "a*") → true
– isMatch("aa", ".*") → true
– isMatch("ab", ".*") → true
– isMatch("aab", "c*a*b") → true
13:52


6.5.2双序列型动态规划-正则匹配2
• 题意：
• 给定两个字符串A，B
• B是一个正则表达式，里面可能含有'?'和'*'
– '?' 可以匹配任何单个字符
– '*' 可以匹配0个或多个任意字
• 问A和B是否匹配
• 例子：
– isMatch("aa","a") → false
– isMatch("aa","aa") → true
– isMatch("aaa","aa") → false
– isMatch("aa", "*") → true
– isMatch("aa", "a*") → true
– isMatch("ab", "?*") → true
– isMatch("aab", "c*a*b") → false
06:33


6.6 双序列型动态规划—01串
• 题意：
• 给定T个01串S0, S1, …, ST-1
• 现有m个0，n个1
• 问最多能组成多少个给定01串
• 每个串最多组成一次
10:35

必做

单选题 Ones And Zeros 这道题中的状态 f[i][j][k] 中, 表示 0 和 1 的数量的 j 和 k 之间也有一定关系, 所以最终可以使用滚动数组优化至一维.

A 正确

B 错误

提交 我不会

正确答案是 B，有52%的同学答对了，加油赶上他们!

解析: 我们无法在进行决策前就得知前 i 个字符串中的最优决策是选择哪些, 所以不能确定 j 和 k 的关系.

必做

单选题 "rabbbit" 的所有子序列中, "rabbit" 出现了多少次? (假定删除不同下标的字符得到的子序列算为不同的子序列)

A 2

B 3

C 4

D 5

提交 我不会

正确答案是 B，有69%的同学超过了你，但是千万不要气馁。

解析: 我们可以删除三个 b 中的任意一个 b, 得到 "rabbit", 也就是说有三个子序列.

必做

多选题 给定两个字符串 A 和 B, 问 A 的所有子序列中 B 出现了多少次. 你准备使用什么算法来解决这个问题?

A 枚举, 枚举 A 的所有的子序列

B 数学, 这其中存在一定的数学规律, 可以直接计算出结果

C 动态规划, 尝试定义状态, 思考状态转移方程

D 肉眼判断法

提交 我不会

正确答案是 A C，有38%的同学做对了这道题目哦，继续努力!

解析:
A 是可行的方案, 但是枚举 A 的所有子序列是 O(2^N) 复杂度的算法, 效率很低
B 到目前为止, 人们还没能找到这个问题中的数学规律, 你想尝试一下?
C 是正确的做法，具体的方案可以参照接下来几题。
D 显然不可取, 肉眼观察容易疏漏而且效率极低

必做

单选题 **很多人选择了动态规划，那么我们尝试定义状态，你认为这个问题的状态应该是一维的还是二维的(不考虑滚动数组优化的情况下)？**

A 一维的

B 二维的

提交 我不会

正确答案是 B，有69%的同学答对了，要加油了。

必做

单选题 **下面哪个是合理的状态定义?**

A f[i] 表示 A 的前 i 个字符的子序列中 B 出现的次数

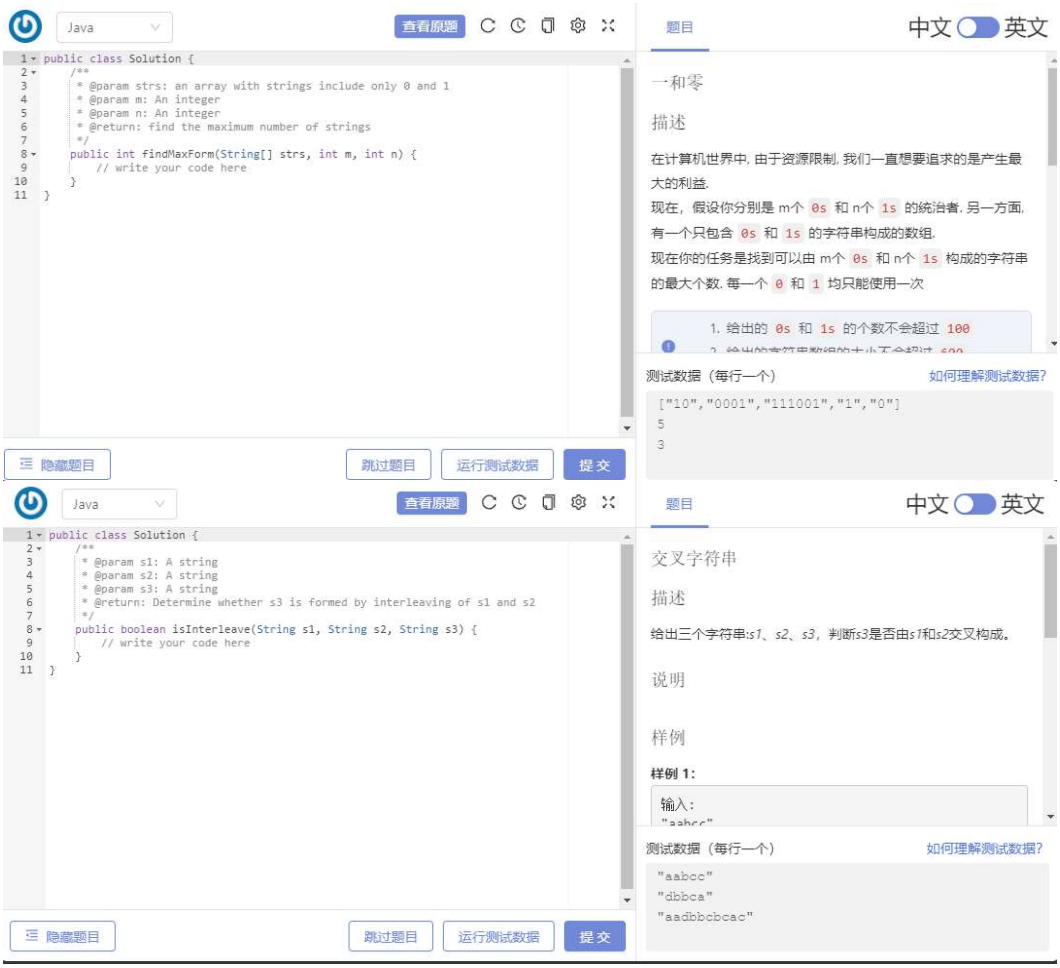
B f[i][j] 表示 A 的前 i 个字符的子序列中 B 的前 j 个字符的子串出现的次数

C f[i][j] 表示 A 的以 i 结尾的子序列中 B 的前 j 个字符的子串出现的次数

我不会

正确答案是 B，有50%的同学答对了，加油赶上他们!

解析: 怎么判断一个状态好不好? 只要尝试推导一下状态转移方程就可以了. A 选项的状态无法推导出一个有效的状态转移方程. 而 C 选项推导出的状态转移方程比 B 要更复杂.


Java
查看原题
public class Solution {
/**
* @param strs: an array with strings include only 0 and 1
* @param m: An integer
* @param n: An integer
* @return: find the maximum number of strings
*/
public int findMaxForm(String[] strs, int m, int n) {
// write your code here
}
}
隐藏题目
跳过题目
运行测试数据
提交
题目
中文 英文
一和零
描述
在计算机世界中, 由于资源限制, 我们一直想要追求的是产生最大的利益.
现在，假设你分别是 m个 0s 和 n个 1s 的统治者, 另一方面, 有一个只包含 0s 和 1s 的字符串构成的数组.
现在你的任务是找到可以由 m个 0s 和 n个 1s 构成的字符串的最大个数, 每一个 0 和 1 均只能使用一次
1. 给出的 0s 和 1s 的个数不会超过 100
测试数据（每行一个）
如何理解测试数据?
["10","0001","111001","1","0"]
5
3
Java
查看原题
public class Solution {
/**
* @param s1: A string
* @param s2: A string
* @param s3: A string
* @return: Determine whether s3 is formed by interleaving of s1 and s2
*/
public boolean isInterleave(String s1, String s2, String s3) {
// write your code here
}
}
隐藏题目
跳过题目
运行测试数据
提交
题目
中文 英文
交叉字符串
描述
给出三个字符串:s1、s2、s3，判断s3是否由s1和s2交叉构成。
说明
样例
样例 1:
输入:
测试数据（每行一个）
如何理解测试数据?
"aabcc"
"dbbca"
"aadbbcbcac"