BITTIGER

CS102 Top100高频算法设计课

第五、六节 动态规划和经典算法

左程云

BITTIGER

版权声明

BITTIGER

Copyright Policy

All content included on the Site or third-party platforms as part of the class, such as text, graphics, logos, button icons, images, audio clips, video clips, live streams, digital downloads, data compilations, and software, is the property of BitTiger or its content suppliers and protected by copyright laws.

Any attempt to redistribute or resell BitTiger content will result in the appropriate legal action being taken.

We thank you in advance for respecting our copyrighted content.For more info see https://www.bittiger.io/termsofuse and https://www.bittiger.io/termsofservice

斐波那契系列问题的递归和动态规划

【题目一】
给定整数*N*，返回斐波那契数列的第*N*项。
【补充题目一】
给定整数*N*，代表台阶数，一次可以跨2个或者1个台阶，返回有多少种走法。
【举例】
N=3，可以三次都跨1个台阶；也可以先跨2个台阶，再跨1个台阶；还可以先跨1个台阶，再跨2个台阶。所以有三种走法，返回3。

斐波那契系列问题的递归和动态规划

【补充题目二】
假设农场中成熟的母牛每年只会生1头小母牛，并且永远不会死。第一年农场有1只成熟的母牛，从第二年开始，母牛开始生小母牛。每只小母牛3年之后成熟又可以生小母牛。给定整数*N*，求出*N*年后牛的数量。
【举例】
N=6，第1年1头成熟母牛记为a；第2年a生了新的小母牛，记为b，总牛数为2；第3年a生了新的小母牛，记为c，总牛数为3；第4年a生了新的小母牛，记为d，总牛数为4。第5年b成熟了，a和b分别生了新的小母牛，总牛数为6；第6年c也成熟了，a、b和c分别生了新的小母牛，总牛数为9，返回9。
【要求】
对以上所有的问题，请实现时间复杂度*O*(log*N*)的解法。

斐波那契系列问题的递归和动态规划

【解题点】

1，如果递归式形如：f(n) = a*f(n-1)+b*f(n-2)+c*f(n-3)+…+k*f(n-i)，其中a、b、c……k都是整数。注意，递归式除了初始条件之外，都严格满足这种形式。

2，那么，都一定可以用i阶矩阵的方法求解：

{f(n)…f(n-i+1)} = {f(n-1)…f(n-i)} * (一个i*i矩阵的n-i次方)

3，因为有足够多的初始值，所以一定可以把i*i矩阵中的每个值求出来

4，关键在于理解一个整数的n次方怎么求更快，矩阵也是一样的

矩阵的最小路径和

【题目二】
给定一个矩阵*m*，从左上角开始每次只能向右或者向下走，最后到达右下角的位置，路径上所有的数字累加起来就是路径和，返回所有的路径中最小的路径和。
【举例】
如果给定的*m*如下：
1　3　5　9
8　1　3　4
5　0　6　1
8　8　4　0
路径1，3，1，0，6，1，0是所有路径中路径和最小的，所以返回12。

矩阵的最小路径和

【解题点】
1，题是简单题，动态规划的思路没什么好说的
2，注意动态规划空间压缩的技巧！可以用在很多题上！
3，如果要求得到“轨迹”，往往不能使用空间压缩，关键看这种“轨迹”是否需要整张动态表的信息。很多题都是这样！

换钱的方法数

【题目三】
给定数组arr，arr中所有的值都为正数且不重复。每个值代表一种面值的货币，每种面值的货币可以使用任意张，再给定一个整数aim代表要找的钱数，求换钱有多少种方法。
【举例】
arr=[5,10,25,1]，aim=0。
组成0元的方法有1种，就是所有面值的货币都不用。所以返回1。
arr=[5,10,25,1]，aim=15。
组成15元的方法有6种，分别为3张5元、1张10元+1张5元、1张10元+5张1元、10张1元+1张5元、2张5元+5张1元和15张1元。所以返回6。
arr=[3,5]，aim=2。
任何方法都无法组成2元。所以返回0。

换钱的方法数

【解题点】
1，这道题非常重要，他解释了所有动态规划面试题的普遍规律
2，暴力递归最难写，但只要写出来，剩下的优化水到渠成
3，分析自己的暴力递归，根据你的参数分析出哪些参数可以代表一个递归过程，把那些代表一个递归过程的参数联合起来做为一个key，返回值做为value，存到一个全局结构中（比如哈希表），在要计算一个递归过程之前，先查一下之前计算过没有，如果有，直接拿出值用；如果没有，去计算然后把返回值存进入。说白了就是当缓存使用。就是记忆化搜索的方法。

换钱的方法数

【解题点】
4，记忆化搜索的方法就是“粗糙版”的动态规划，此时根据你的缓存结构，分析出你的动态规划表的实际结构，弄清这个结构中的依赖关系。写出状态表达式。
5，根据状态表达式，看看这个状态表达式能否化简，得到“精致版”的动态规划。
6，最初始的位置先计算，然后根据依赖线索，逐渐算上去！
7，这道题是这个章节的题眼，你想想不到这个方法论有多么重要！好多题都是这样！我会让你看到的！
8，学会优化是一方面，但在面试场上，有很多约定俗成的经典题目，经典题目你看到了，没什么好说的，你需要直接把动态规划的“最终结论”说给面试官，而不要和面试官掰扯方法论。

换钱的最少货币数

【题目四】
给定数组arr，arr中所有的值都为正数且不重复。每个值代表一种面值的货币，每种面值的货币可以使用任意张，再给定一个整数aim代表要找的钱数，求组成aim的最少货币数。
【举例】
arr=[5,2,3]，aim=20。
4张5元可以组成20元，其他的找钱方案都要使用更多张的货币，所以返回4。
arr=[5,2,3]，aim=0。
不用任何货币就可以组成0元，返回0。
arr=[3,5]，aim=2。
根本无法组成2元，钱不能找开的情况下默认返回-1。

换钱的最少货币数

【补充题目】
给定数组arr，arr中所有的值都为正数。每个值仅代表一张钱的面值，再给定一个整数aim代表要找的钱数，求组成aim的最少货币数。
【举例】
arr=[5,2,3]，aim=20。
5元、2元和3元的钱各有1张，所以无法组成20元，默认返回-1。
arr=[5,2,5,3]，aim=10。
5元的货币有2张，可以组成10元，且该方案所需张数最少，返回2。
arr=[5,2,5,3]，aim=15。
所有的钱加起来才能组成15元，返回4。
arr=[5,2,5,3]，aim=0。
不用任何货币就可以组成0元，返回0。

换钱的最少货币数

【原问题解题点】

1，如果arr的长度为*N*，生成行数为*N*、列数为aim+1的动态规划表的dp。dp[i][j]的含义是，在可以任意使用arr[0..i]货币的情况下，组成*j*所需的最小张数。

2，

1) dp[0..N-1][0]的值（即dp矩阵中第一列的值）表示找的钱数为0时需要的最少张数，钱数为0时，完全不需要任何货币，所以全设为0即可。

2)dp[0][0..aim]的值（即dp矩阵中第一行的值）表示只能使用arr[0]货币的情况下，找某个钱数的最小张数。比如，arr[0]=2，那么能找开的钱数为2，4，6，8，...所以令dp[0][2]=1，dp[0][4]=2，dp[0][6]=3，...第一行其他位置所代表的钱数一律找不开，所以一律设为32位整数的最大值，我们把这个值记为max。

换钱的最少货币数

【原问题解题点】

3．剩下的位置依次从左到右，再从上到下计算。假设计算到位置(*i*,*j*)，dp[i][j]的值可能来自下面的情况。

完全不使用当前货币arr[i]情况下的最少张数，即dp[i-1][j]的值。

只使用1张当前货币arr[i]情况下的最少张数，即dp[i-1][j-arr[i]]+1。

只使用2张当前货币arr[i]情况下的最少张数，即dp[i-1][j-2*arr[i]]+2。

只使用3张当前货币arr[i]情况下的最少张数，即dp[i-1][j-3*arr[i]]+3。

所有的情况中，最终取张数最小的。所以

dp[i][j]=min{dp[i-1][j-k*arr[i]]+k(0<=k)}

换钱的最少货币数

【原问题解题点】

dp[i][j]=min{dp[i-1][j-k*arr[i]]+k(0<=k)}

=>dp[i][j]=min{dp[i-1][j], min{dp[i-1][j-x*arr[i]]+x(1<=x)}}

=>dp[i][j]=min{dp[i-1][j], min{dp[i-1][j-arr[i]-y*arr[i]]+y+1(0<=y)}}

又有 min{dp[i-1][j-arr[i]-y*arr[i]]+y(0<=y)} => dp[i][j-arr[i]]，

所以，最终有：dp[i][j]=min{dp[i-1][j],dp[i][j-arr[i]]+1}

这是啥？这就是化简状态表达式！变成精致版动态规划版的过程！熟悉吗？

换钱的最少货币数

【补充解题点】
没难度，比原问题差远了。就是一个煎蛋背包问题而已

【关键】
可以进行空间压缩哦！~

最长递增子序列

【题目五】
给定数组arr，返回arr的最长递增子序列。
【举例】
arr=[2,1,5,3,6,4,8,9,7]，返回的最长递增子序列为[1,3,4,8,9]。
【要求】
如果arr长度为*N*，请实现时间复杂度为*O*(*N*log*N*)的方法。

最长递增子序列

【解题点】

1，先理解O(N^2)的方法：生成长度为*N*的数组dp，dp[i]表示在以arr[i]这个数结尾的情况下，arr[0..i]中的最大递增子序列长度。 那么在计算位置i的时候，就遍历之前的(0..i-1)位置，看看接在谁后面最大。

2， O(N^2)的方法之所以慢，是因为有遍历的过程！所以生成ends数组，ends[0..right]为有效区，ends[right+1..N-1]为无效区。对有效区上的位置*b*，如果有ends[b]==c，则表示遍历到目前为止，在所有长度为*b*+1的递增序列中，最小的结尾数是c。无效区的位置则没有意义。

求解的过程中，用二分的觉悟逐渐扩充+更新ends数组的有效区！

汉诺塔问题

【题目六】
给定一个整数*n*，代表汉诺塔游戏中从小到大放置的*n*个圆盘，假设开始时所有的圆盘都放在左边的柱子上，想按照汉诺塔游戏的要求把所有的圆盘都移到右边的柱子上。实现函数打印最优移动轨迹。
【进阶题目】
给定一个整型数组arr，其中只含有1、2和3，代表所有圆盘目前的状态，1代表左柱，2代表中柱，3代表右柱，arr[i]的值代表第*i*+1个圆盘的位置。比如，arr=[3,3,2,1]，代表第1个圆盘在右柱上、第2个圆盘在右柱上、第3个圆盘在中柱上、第4个圆盘在左柱上。如果arr代表的状态是最优移动轨迹过程中出现的状态，返回arr这种状态是最优移动轨迹中的第几个状态。如果arr代表的状态不是最优移动轨迹过程中出现的状态，则返回-1。

汉诺塔问题

【原问题解题点】
假设有from、mid和to三个柱子，都在from的圆盘1~i完全移动到to，最优过程为：
步骤1为圆盘1~*i*-1从from移动到mid。
步骤2为单独把圆盘*i*从from移动到to。
步骤3为把圆盘1~*i*-1从mid移动到to。
如果圆盘只有1个，直接把这个圆盘从from移动到to即可。

汉诺塔问题

【进阶解题点】

首先求都在from柱子上的圆盘1~i，如果都移动到to上的最少步骤数，假设为$S(i)$。

根据上面的步骤，$S(i)$=步骤1的步骤总数+1+步骤3的步骤总数=$S(i-1)+1+S(i-1)$，$S(1)=1$。所以$S(i)+1=2(S(i-1)+1)$，$S(1)+1==2$。

根据等比数列求和公式得到$S(i)+1=2^i$，所以$S(i)=2^i-1$。

汉诺塔问题

【进阶解题点】
对于数组arr来说，arr[*N*-1]表示最大圆盘*N*在哪个柱子上，情况有以下三种：
圆盘*N*在左柱上，说明步骤1或者没有完成，或者已经完成，需要考查圆盘1~*N*-1的状况。
圆盘*N*在右柱上，说明步骤1已经完成，起码走完了2^{N-1}-1步。步骤2也已经完成，起码又走完了1步，所以当前状况起码是最优步骤的2^{N-1}步，剩下的步骤怎么确定还得继续考查圆盘1~*N*-1的状况。
圆盘*N*在中柱上，这是不可能的，最优步骤中不可能让圆盘*N*处在中柱上，直接返回-1。

汉诺塔问题

【进阶解题点】
整个过程可以总结为：
对圆盘1~*i*来说，如果目标为从from到to，那么情况有三种：
1，圆盘*i*在from，需要继续考查圆盘1~*i*-1的状况，圆盘1~*i*-1的目标为从from到mid。
2，圆盘*i*在to上，说明起码走完了2^{i-1}步，剩下的步骤怎么确定还得继续考查圆盘1~*i*-1的状况，圆盘1~*i*-1的目标为从mid到to。
3，圆盘*i*在mid上，直接返回-1。

最长公共子序列问题

【题目七】
给定两个字符串str1和str2，返回两个字符串的最长公共子序列。
【举例】
str1="1A2C3D4B56"，str2="B1D23CA45B6A"。
"123456"或者"12C4B6"都是最长公共子序列，返回哪一个都行。

最长公共子序列问题

【解题点】

如果str1的长度为*M*，str2的长度为*N*，生成大小为*M*×*N*的矩阵dp，行数为*M*，列数为*N*。dp[i][j]的含义是str1[0..i]与str2[0..j]的最长公共子序列的长度。从左到右，再从上到下计算矩阵dp

最长公共子序列问题

【解题点】

3．对其他位置(i,j)，dp[i][j]的值只可能来自以下三种情况：

可能是dp[i-1][j]，代表str1[0..i-1]与str2[0..j]的最长公共子序列长度。

可能是dp[i][j-1]，代表str1[0..i]与str2[0..j-1]的最长公共子序列长度。

如果str1[i]==str2[j]，还可能是dp[i-1][j-1]+1。

这三个可能的值中，选最大的作为dp[i][j]的值

最长公共子序列问题

【解题点】

3．对其他位置(i,j)，dp[i][j]的值只可能来自以下三种情况：

可能是dp[i-1][j]，代表str1[0..i-1]与str2[0..j]的最长公共子序列长度。

可能是dp[i][j-1]，代表str1[0..i]与str2[0..j-1]的最长公共子序列长度。

如果str1[i]==str2[j]，还可能是dp[i-1][j-1]+1。

这三个可能的值中，选最大的作为dp[i][j]的值

最长公共子序列问题

【解题点】
1，得到动态规划表，找出选择路径是很容易的，某个位置怎么得到的决策值，就怎么回去
2，注意如果是求长度还可以空间压缩哦！

最长公共子串问题

【题目八】
给定两个字符串str1和str2，返回两个字符串的最长公共子串。
【举例】
str1="1AB2345CD"，str2="12345EF"，返回"2345"。
【要求】
如果str1长度为M，str2长度为N，实现时间复杂度为$O(M\times N)$，额外空间复杂度为$O(1)$的方法。

最长公共子串问题

【解题点】
碉堡的地方完全在于怎么省空间！额外空间复杂度O(1)！

最小编辑代价

【题目九】
给定两个字符串str1和str2，再给定三个整数ic、dc和rc，分别代表插入、删除和替换一个字符的代价，返回将str1编辑成str2的最小代价。
【举例】
str1="abc"，str2="adc"，ic=5，dc=3，rc=2。
从"abc"编辑成"adc"，把'b'替换成'd'是代价最小的，所以返回2。
str1="abc"，str2="adc"，ic=5，dc=3，rc=100。
从"abc"编辑成"adc"，先删除'b'，然后插入'd'是代价最小的，所以返回8。
str1="abc"，str2="abc"，ic=5，dc=3，rc=2。
不用编辑了，本来就是一样的字符串，所以返回0。

最小编辑代价

【解题点】

生成大小为(*M*+1)×(*N*+1)的矩阵dp，dp[i][j]的值代表str1[0..i-1]编辑成str2[0..j-1]的最小代价。比如str1="ab12cd3"，str2="abcdf"，ic=5，dc=3，rc=2。

dp是一个8×6的矩阵，最终计算结果如下。

	‘ ’	‘a’	‘b’	‘c’	‘d’	‘f’
‘ ’	0	5	10	15	20	25
‘a’	3	0	5	10	15	20
‘b’	6	3	0	5	10	15
‘1’	9	6	3	2	7	12
‘2’	12	9	6	5	4	9
‘c’	15	12	9	6	7	6
‘d’	18	15	12	9	6	9
‘3’	21	18	15	12	9	8

最小编辑代价

【解题点】

1，设置第一行和第一列

2，一个位置dp[i][j]来自四种决策：

1）可能来自dc+dp[i-1][j]

2）可能来自dp[i][j-1]+ic

3）如果str1[i-1]!=str2[j-1]，可能来自dp[i-1][j-1]+rc

4）如果str1[i-1]!=str2[j-1]，可能来自dp[i-1][j-1]

3，空间压缩哦～

字符串的交错组成

【题目十】
给定三个字符串str1、str2和aim，如果aim包含且仅包含来自str1和str2的所有字符，而且在aim中属于str1的字符之间保持原来在str1中的顺序，属于str2的字符之间保持原来在str2中的顺序，那么称aim是str1和str2的交错组成。实现一个函数，判断aim是否是str1和str2交错组成。
【举例】
str1="AB"，str2="12"。那么"AB12"、"A1B2"、"A12B"、"1A2B"和"1AB2"等都是str1和str2的交错组成。

字符串的交错组成

【解题点】
aim如果是str1和str2的交错组成，aim的长度一定是$M+N$，否则直接返回false。然后生成大小为$(M+1)\times(N+1)$布尔类型的矩阵dp，dp[i][j]的值代表aim[0..i+j-1]能否被str1[0..i-1]和str2[0..j-1]交错组成。

计算dp矩阵的时候，是从左到右，再从上到下计算的，dp[M][N]也就是dp矩阵中最右下角的值，表示aim整体能否被str1整体和str2整体交错组成，也就是最终结果。

字符串的交错组成

【解题点】

1．dp[0][0]=true。aim为空串时，当然可以被str1为空串和str2为空串交错组成。

2．矩阵dp第一列即dp[0..M-1][0]。dp[i][0]表示aim[0..i-1]能否只被str1[0..i-1]交错组成。如果aim[0..i-1]等于str1[0..i-1]，则令dp[i][0]=true，否则令dp[i][0]=false。

3．矩阵dp第一行即dp[0][0..N-1]。dp[0][j]表示aim[0..j-1]能否只被str2[0..j-1]交错组成。如果aim[0..j-1]等于str1[0..j-1]，则令dp[i][0]=true，否则令dp[i][0]=false。

4．对其他位置(*i*,*j*)，dp[i][j]的值由下面的情况决定。

dp[i-1][j]代表aim[0..i+j-2]能否被str1[0..i-2]和str2[0..j-1]交错组成，如果可以，那么如果再有str1[i-1]等于aim[i+j-1]，说明str1[i-1]又可以作为交错组成aim[0..i+j-1]的最后一个字符。令dp[i][j]=true。

dp[i][j-1]代表aim[0..i+j-2]能否被str1[0..i-1]和str2[0..j-2]交错组成，如果可以，那么如果再有str2[j-1]等于aim[i+j-1]，说明str1[j-1]又可以作为交错组成aim[0..i+j-1]的最后一个字符。令dp[i][j]=true。

如果第1种情况和第2种情况都不满足，令dp[i][j]=false。

字符串的交错组成

【解题点】
空间压缩哦～

数字字符串转换为字母组合的种数

【题目十一】
给定一个字符串str，str全部由数字字符组成，如果str中某一个或某相邻两个字符组成的子串值在1~26之间，则这个子串可以转换为一个字母。规定"1"转换为"A"，"2"转换为"B"，"3"转换为"C"..."26"转换为"Z"。写一个函数，求str有多少种不同的转换结果，并返回种数。
【举例】
str="1111"。
能转换出的结果有"AAAA"、"LAA"、"ALA"、"AAL"和"LL"，返回5。
str="01"。
"0"没有对应的字母，而"01"根据规定不可转换，返回0。
str="10"。
能转换出的结果是"J"，返回1。

数字字符串转换为字母组合的种数

【解题点 - 暴力递归方法】
假设str的长度为N，先定义递归函数$p(i)(0\le i\le N)$。$p(i)$的含义是str[0..i-1]已经转换完毕，而str[i..N-1]还没转换的情况下，最终合法的转换种数有多少并返回。特别指出，$p(N)$表示str[0..N-1]（也就是str的整体）都已经转换完，没有后续的字符了，那么合法的转换种数为1，即$p(N)=1$。

数字字符串转换为字母组合的种数

【解题点 - 暴力递归方法】

情况1：如果i==N，$p(N)=1$，直接返回1

情况2：如果不满足情况1，又有str[i]==‘0’，str[0..i-1]已经转换完毕，而str[i..N-1]此时又以'0'开头，str[i..N-1]无论怎样都不可能合法转换，所以直接返回0。

情况3：

1）如果不满足情况1和情况2，说明str[i]属于'1'~'9'，str[i]可以转换为'A'~'I'，那么$p(i)$的值一定包含$p(i+1)$的值，即$p(i)=p(i+1)$。

2）如果不满足情况1和情况2，说明str[i]属于'1'~'9'，如果又有str[i..i+1]在"10”"~"26"之间，str[i..i+1]可以转换为'J'~'Z'，那么$p(i)$的值一定也包含$p(i+2)$的值，即$p(i)+=p(i+2)$。

数字字符串转换为字母组合的种数

【解题点 - 动态规划方法】

1，$p(i)$最多依赖$p(i+1)$和$p(i+2)$的值，这是可以从后往前进行顺序计算的。

2，这也是可以空间压缩的，只用有限几个变量滚动更新即可。

3，这是不能用矩阵乘法的，因为状态表达式有条件转移。

表达式得到期望结果的组成种数

【题目十二】
给定一个只由0（假）、1(真)、&(逻辑与)、|（逻辑或）和^(异或)五种字符组成的字符串express，再给定一个布尔值desired。返回express能有多少种组合方式，可以达到desired的结果。
【举例】
express="1^0|0|1"，desired=false。
只有1^((0|0)|1)和1^(0|(0|1))的组合可以得到false，返回2。
express="1"，desired=false。
无组合则可以得到false，返回0。

表达式得到期望结果的组成种数

【解题点】
应该首先判断express是否合乎题目要求，比如"1^"和"10"，都不是有效的表达式。
总结起来有以下三个判断标准：
表达式的长度必须是奇数。
表达式下标为偶数位置的字符一定是'0'或者'1'。
表达式下标为奇数位置的字符一定是'&'或'|'或'^'。

表达式得到期望结果的组成种数

【解题点】
你只要想出的暴力递归方法，根据之前讲过的方法论，我们可以顺畅的得到动态规划的方法。
【关键点】
总结出合理的状态依赖路径！

排成一条线的纸牌博弈问题

【题目十三】
给定一个整型数组arr，代表数值不同的纸牌排成一条线。玩家A和玩家B依次拿走每张纸牌，规定玩家A先拿，玩家B后拿，但是每个玩家每次只能拿走最左或最右的纸牌，玩家A和玩家B都绝顶聪明。请返回最后获胜者的分数。

排成一条线的纸牌博弈问题

【举例】
arr=[1,2,100,4]。
开始时玩家A只能拿走1或4。如果玩家A拿走1，则排列变为[2,100,4]，接下来玩家B可以拿走2或4，然后继续轮到玩家A。如果开始时玩家A拿走4，则排列变为[1,2,100]，接下来玩家B可以拿走1或100，然后继续轮到玩家A。玩家A作为绝顶聪明的人不会先拿4，因为拿4之后，玩家B将拿走100。所以玩家A会先拿1，让排列变为[2,100,4]，接下来玩家B不管怎么选，100都会被玩家A拿走。玩家A会获胜，分数为101。所以返回101。
arr=[1,100,2]。
开始时玩家A不管拿1还是2，玩家B作为绝顶聪明的人，都会把100拿走。玩家B会获胜，分数为100。所以返回100。

排成一条线的纸牌博弈问题

【解题点】
你只要想出的暴力递归方法，根据之前讲过的方法论，我们可以顺畅的得到动态规划的方法。
【关键点】
总结出合理的状态依赖路径！

N皇后问题

【题目十四】
*N*皇后问题是指在*N*×*N*的棋盘上要摆*N*个皇后，要求任何两个皇后不同行、不同列，也不在同一条斜线上。给定一个整数*n*，返回*n*皇后的摆法有多少种。
【举例】
n=1，返回1。
n=2或3，2皇后和3皇后问题无论怎么摆都不行，返回0。
n=8，返回92。

N皇后问题

【解题点】

如果在(*i*,*j*)位置(第*i*行第*j*列)放置了一个皇后，接下来在哪些位置不能放置皇后呢？

1．整个第*i*行的位置都不能放置。

2．整个第*j*列的位置都不能放置。

3．如果位置(*a*,*b*)满足|a-i|==|b-j|，说明(*a*,*b*)与(*i*,*j*)处在同一条斜线上，也不能放置。

N皇后问题

【解题点】
1，标准有了，程序不难写，最重要的是加速过程
2，加速过程用位运算来加速，这个不是特例，很多题目优化基于此
3，这个话题比较高阶，只是提醒同学们注意，位运算的执行速度很快，而位信息本身就可以表达一种状态，那么基于位信息的状态检查和更新，可以极大加速处理过程。如果遇到类似的实现，别慌，就是加速过程用位运算了而已。

添加最少字符使字符串整体都是回文字符串

【题目十五】
给定一个字符串str，如果可以在str的任意位置添加字符，请返回在添加字符最少的情况下，让str整体都是回文字符串的一种结果。
【举例】
str="ABA"。str本身就是回文串，不需要添加字符，所以返回"ABA"。
str="AB"。可以在'A'之前添加'B'，使str整体都是回文串，故可以返回"BAB"。也可以在'B'之后添加'A'，使str整体都是回文串，故也可以返回"ABA"。总之，只要添加的字符数最少，只返回其中一种结果即可。

添加最少字符使字符串整体都是回文字符串

【解题点】

如果str的长度为*N*，动态规划表是一个*N*×*N*的矩阵记为dp[][]。dp[i][j]值的含义代表子串str[i..j]最少添加几个字符可以使str[i..j]整体都是回文串。

如果字符串str[i..j]只有一个字符，此时dp[i][j]=0

如果字符串str[i..j]只有两个字符。

1）如果两个字符相等，那么dp[i][j]=0

2）如果两个字符不相等，那么dp[i][j]=1

如果字符串str[i..j]多于两个字符。

如果str[i]==str[j]，那么dp[i][j]=dp[i+1][j-1]

如果str[i]!=str[j]，要让str[i..j]整体变为回文串有两种方法：

1）一种方法是让str[i..j-1]先变成回文串，然后在左边加上字符str[j]，

2）另一种方法是让str[i+1..j]先变成回文串，然后在右边加上字符str[i]，就是str[i..j]整体变成回文串的结果。

两种方法中哪个代价最小就选择哪个，即dp[i][j] = min { dp[i][j-1] , dp[i+1][j] }+1。

添加最少字符使字符串整体都是回文字符串

【解题点】
根据位置依赖，设计计算顺序。
1，对本题来说，如果开始位置是0，结束位置是j，想计算dp[0][j]，依次计算(j，j)、(j-1，j)、(j-2, j)… (1，j)、(0,j)
2，完成步骤1，开始位置是0，结束位置变成j+1，然后重复步骤1
3，直到j扩到最右的位置。

括号字符串的有效性和最长有效长度

【题目十六】
给定一个字符串str，判断是不是整体有效的括号字符串。
【举例】
str="()"，返回true；str="(()())"，返回true；str="(())"，返回true。
str="())"。返回false；str="()("，返回false；str="()a()"，返回false。
【补充题目】
给定一个括号字符串str，返回最长的有效括号子串。
【举例】
str="(()())"，返回6；str="())"，返回2；str="()(()()("，返回4。

括号字符串的有效性和最长有效长度

【原问题解题点】
1．从左到右遍历字符串str，判断每一个字符是不是'('或')'，如果不是，就直接返回false。
2．遍历到每一个字符时，都检查到目前为止'('和')'的数量，如果')'更多，则直接返回false。
3．遍历后检查'('和')'的数量，如果一样多，则返回true，否则返回false。

括号字符串的有效性和最长有效长度

【进阶问题解题点】
首先生成长度和str字符串一样的数组dp[]，dp[i]值的含义为str[0..i]中必须以字符str[i]结尾的最长的有效括号子串长度。那么dp[i]值可以按如下方式求解：
1．dp[0]=0。只含有一个字符肯定不是有效括号字符串，长度自然是0。
2．从左到右依次遍历str[1..N-1]的每个字符，假设遍历到str[i]。
3．如果str[i]=='('，有效括号字符串必然是以')'结尾，而不是以'('结尾，dp[i] = 0。

括号字符串的有效性和最长有效长度

【进阶问题解题点】
4．如果str[i]==')'，那么以str[i]结尾的最长有效括号子串可能存在。dp[i-1]的值代表必须以str[i-1]结尾的最长有效括号子串的长度，所以如果i-dp[i-1]-1位置上的字符是'('，就能与当前位置的str[i]字符再配出一对有效括号。比如"(()())"，假设遍历到最后一个字符')'，必须以倒数第二个字符结尾的最长有效括号子串是"()()"，找到这个子串之前的字符，即i-dp[i-1]-1位置的字符，发现是'('，所以它可以和最后一个字符再配出一对有效括号。如果该情况发生，dp[i]的值起码是dp[i-1]+2

括号字符串的有效性和最长有效长度

【进阶问题解题点】
5，同时还有一部分长度容易被人忽略。比如，"()(())"，假设遍历到最后一个字符')'，通过上面的过程找到的必须以最后字符结尾的最长有效括号子串起码是"(())"，但是前面还有一段"()"，可以和"(())"结合在一起构成更大的有效括号子串。也就是说，str[i-dp[i-1]-1]和str[i]配成了一对，这时还应该把dp[i-dp[i-1]-2]的值加到dp[i]中，这么做表示把str[i-dp[i-1]-2]结尾的最长有效括号子串接到前面，才能得到以当前字符结尾的最长有效括号子串。
6，dp[0..N-1]中的最大值就是最终的结果。

0左边必有1的二进制字符串数量

【题目十七】
给定一个整数*N*，求由"0"字符与"1"字符组成的长度为*N*的所有字符串中，满足"0"字符的左边必有"1"字符的字符串数量。
【举例】
N=1。只由"0"与"1"组成，长度为1的所有字符串："0"、"1"。只有字符串"1"满足要求，所以返回1。
N=2。只由"0"与"1"组成，长度为2的所有字符串为："00"、"01"、"10"、"11"。只有字符串"10"和"11"满足要求，所以返回2。
N=3。只由"0"与"1"组成，长度为3的所有字符串为："000"、"001"、"010"、"011"、"100"、"101"、"110"、"111"。字符串"101"、"110"、"111"满足要求，所以返回3。

0左边必有1的二进制字符串数量

【解题点】
化简之后是斐波那列问题，矩阵乘法的解法

回文最少分割数

【题目十八】
给定一个字符串str，返回把str全部切成回文子串的最小分割数。
【举例】
str="ABA"。
不需要切割，str本身就是回文串，所以返回0。
str="ACDCDCDAD"。
最少需要切2次变成3个回文子串，比如"A"、"CDCDC"和"DAD"，所以返回2。

回文最少分割数

【解题点】
dp[i]的含义是子串str[i..len-1]至少需要切割几次，才能把str[i..len-1]全部切成回文子串。那么，dp[0]就是最后的结果。
从右往左依次计算dp[i]的值，i初始为len-1，具体计算过程如下：
1．假设*j*位置处在*i*与len-1位置之间(i<=j<len)，如果str[i..j]是回文串，那么dp[i]的值可能是dp[j+1]+1，其含义是在str[i..len-1]上，既然str[i..j]是一个回文串，那么它可以自己作为一个分割的部分，剩下的部分（即str[j+1..len-1]）继续做最经济的切割，而dp[j+1]值的含义正好是str[j+1..len-1]的最少回文分割数。
2．根据步骤2的方式，让*j*在*i*到len-1位置上枚举，那么所有可能情况中的最小值就是dp[i]的值，即dp[i] = Min { dp[j+1]+1 (i<=j<len，且str[i..j]必须是回文串) }。

回文最少分割数

【解题点】

3．如何方便快速地判断str[i..j]是否是回文串呢？具体过程如下。

1）定义一个二维数组boolean[][] p，如果p[i][j]值为true，说明字符串str[i..j]是回文串，否则不是。在计算dp数组的过程中，希望能够同步、快速地计算出矩阵p。

2）p[i][j]如果为true，一定是以下三种情况，str[i..j]由1个字符组成；str[i..j]由2个字符组成且2个字符相等；str[i+1..j-1]是回文串，即p[i+1][j-1]为true，且str[i]==str[j]，即str[i..j]上首尾两个字符相等

3）在计算dp数组的过程中，位置*i*是从右向左依次计算的。而对每一个*i*来说，又依次从*i*位置向右枚举所有的位置*j*(i<=j<len)，以此来决策出dp[i]的值。所以对p[i][j]来说，p[i+1][j-1]值一定已经计算过。这就使判断一个子串是否为回文串变得极为方便。

4．最终返回dp[0]的值，过程结束。全部过程请参看如下代码中的minCut方法。

字符串匹配问题

【题目十九】
给定字符串str，其中绝对不含有字符'.'和'*'。再给定字符串exp，其中可以含有'.'或'*'，'*'字符不能是exp的首字符，并且任意两个'*'字符不相邻。exp中的'.'代表任何一个字符，exp中的'*'表示'*'的前一个字符可以有0个或者多个。请写一个函数，判断str是否能被exp匹配。
【举例】
str="abc"，exp="abc"，返回true。
str="abc"，exp="a.c"，exp中单个'.'可以代表任意字符，所以返回true。
str="abcd"，exp=".*"。exp中'*'的前一个字符是'.'，所以可表示任意数量的'.'字符，当exp是"...."时与"abcd"匹配，返回true。
str=""，exp="..*"。exp中'*'的前一个字符是'.'，可表示任意数量的'.'字符，但是".*"之前还有一个'.'字符，该字符不受'*'的影响，所以str起码有一个字符才能被exp匹配。所以返回false。

字符串匹配问题

【暴力递归方法解题点】
1，首先解决str和exp有效性的问题。根据描述，str中不能含有‘.’和‘*’，exp中‘*’字符不能是首字符，并且任意两个‘*’字符不相邻 。
2，下面解释一下递归过程， process(char[] s, char[] e, int si, int ei)函数的意义是，从str的si位置开始，一直到str结束位置的子串，即str[si...slen]，是否能被从exp的ei位置开始一直到exp结束位置的子串（即exp[ei...elen]）匹配，所以process(s,e,0,0)就是最终返回的结果。
那么在递归过程中如何判断str[si...slen]是否能被exp[ei...elen]匹配呢？

字符串匹配问题

【暴力递归方法解题点】
假设当前判断到str的si位置和exp的ei位置，即process(s,e,si,ei)。
1．如果ei为exp的结束位置(ei==elen)，si也是str的结束位置，返回true，因为“”可以匹配“”。如果si不是str的结束位置，返回false，这是显而易见的。
2．如果ei位置的下一个字符(e[ei+1])不为'*'。那么就必须关注str[si]字符能否和exp[ei]字符匹配。如果str[si]与exp[ei]能匹配(e[ei] == s[si] || e[ei] == '.')，还要关注str后续的部分能否被exp后续的部分匹配，即process(s,e,si+1,ei+1)的返回值。如果str[si]与exp[ei]不能匹配，当前字符都匹配，当然不用计算后续的，直接返回false。

字符串匹配问题

【暴力递归方法解题点】

3．如果当前ei位置的下一个字符(e[ei+1])为'*'字符。

1）如果str[si]与exp[ei]不能匹配，那么只能让exp[ei..ei+1]这个部分为""，也就是exp[ei+1]=='*'字符的前一个字符exp[ei]的数量为0才行，然后考查process(s,e,si,ei+2)的返回值。举个例子，str[si..slen]为"bXXX"，"XXX"代指字符'b'之后的字符串。exp[ei..elen]为"a*YYY"，"YYY"代指字符'*'之后的字符串。当前无法匹配('a'!='b')，所以让"a*"为""，然后考查str[si..slen]（即"bXXX"）能否被exp[ei+2..elen]（即"YYY"）匹配。

字符串匹配问题

【暴力递归方法解题点】

2）如果str[si]与exp[ei]能匹配，这种情况下举例说明。

str[si...slen]为"aaaaaXXX"，"XXX"指不再连续出现'a'字符的后续字符串。exp[ei...elen])为"a*YYY"，"YYY"指字符'*'之后的后续字符串。

如果令"a"和"a*"匹配，且有"aaaaXXX"和"YYY"匹配，可以返回true。

如果令"aa"和"a*"匹配，且有"aaaXXX"和"YYY"匹配，可以返回true。

如果令"aaa"和"a*"匹配，且有"aaXXX"和"YYY"匹配，可以返回true。

如果令"aaaa"和"a*"匹配，且有"aXXX"和"YYY"匹配，可以返回true。

如果令"aaaaa"和"a*"匹配，且有"XXX"和"YYY"匹配，可以返回true。

也就是说，exp[ei..ei+1]（即"a*"）的部分如果能匹配str后续很多位置的时候，只要有一个返回true，就可以直接返回true。

4，整体递归过程结束。

字符串匹配问题

【动态规划方法解题点】
在分析完如上递归过程之后，来看递归函数的结构。我们很容易发现递归函数process(s,e,si,ei)在每次调用的时候，有两个参数是始终不变的(s和e)，所以代表process函数状态的就是si和ei值的组合。所以，如果把递归函数*p*在所有不同参数(si和ei)的情况下的所有返回值看作一个范围，这个范围就是一个(slen+1)*(elen+1)的二维数组，并且p(si,ei)在整个递归过程中，依赖的总是p(si+1,ei+1)或者p(si+k(k>=0),ei+2)，
假设二维数组dp[i][j]代表*p*(*i*,*j*)的返回值，dp[i][j]就只是依赖dp[i+1][j+1]或者dp[i+k(k>=0)][j+2]的值。进一步可以看出，想要求dp[i][j]的值，只需要(*i*,*j*)位置右下方的某些值。所以只要从二维数组的右下角开始，从右到左、再从下到上地计算出二维数组dp中每个位置的值就可以，dp[0][0]就是最终的结果。*p*(*i*,*j*)的递归过程如何，dp[i][j]的值就怎样去计算。这种方法实际上就是动态规划的方法，省去了递归过程中很多重复计算的过程。

字符串匹配问题

【动态规划方法解题点】
先从右到左计算dp[slen][...]，也就是二维数组dp中的最后一行，dp[slen][elen]值的含义是str已经结束，剩下的字符串为""，exp也已经结束，剩下的字符串为""，所以此时exp可以匹配str，dp[slen][elen]=true。对于dp[slen][0..elen-1]的部分，dp[slen][i]的含义是str已经结束，剩下的字符串为""，exp却没有结束，剩下的字符串为exp[i..elen-1]，什么情况下exp[i..elen-1]可以匹配""？只能是不停地重复出现"X*"这种方式。比如，exp[i..elen-1]为"*"，这种情况下，exp[i+1..elen-1]根本不合法，匹配不了""。如果exp[i..elen-1]="A*"，可以匹配""。如果exp[i..elen-1]="A*B*"，也能匹配""。也就是说，在从右向左计算dp[slen][0..elen-1]的过程中，看exp是不是从右往左重复出现"X*"，如果是重复出现，那么如果exp[i]='X'，exp[i+1]='*'，令dp[slen][i]=true，如果exp[i]='*'，exp[i+1]='X'，令dp[slen][i]=false。如果不是重复出现，最后一行后面的部分（即dp[slen][0..i]），全都是false。这样就搞定了dp[][]最后一行的值。

字符串匹配问题

【动态规划方法解题点】
再看看dp[][]除右下角的值之外，最后一列其他位置的值，即dp[0..slen-1][elen]。这表示如果exp已经结束，而str还没结束，显然，exp为""匹配不了任何非空字符串，所以dp[0..slen-1][elen]都为false。
接着看dp[][]倒数第二列的值，即dp[0..slen-1][elen-1]。这表示如果exp还剩一个字符即（exp[elen-1]），而str还剩1个字符或多个字符。很明显，str还剩多个字符的情况下，exp匹配不了。str还剩1个字符的情况下（即str[slen-1]），如果和exp[elen-1]相等，则可以匹配，或者exp[elen-1]=='.'的情况下可以匹配。

字符串匹配问题

【动态规划方法解题点】
因为dp[i][j]只依赖dp[i+1][j+1]或者dp[i+k][j+2](k>=0)的值，所以在单独计算完最后一行、最后一列与倒数第二列之后，剩下的位置在从右到左、再从下到上计算dp值的时候，所有依赖的值都被计算出来，直接拿过来用即可。如果str的长度为N，exp的长度为M，因为有枚举的过程，所以时间复杂度为$O(N^2 \times M)$，额外空间复杂度为$O(N \times M)$。

经典算法串讲

Manacher算法、 KMP算法、 BFPRT算法、 前缀树、 蓄水池算法、 完美洗牌算法

我要打10个！

课程项目负责人：Catherine

邮件：weiyi@bittiger.io

左程云答疑邮箱：chengyunzuo@gmail.com

微信二维码：

关注微信，获得太阁最新信息

微信: bit_tiger

官网: BitTiger.io