算法基础补充(四)--暴力递归到动态规划

2021-01-21 0条评论 176浏览

算法 基础

暴力递归

什么叫尝试?

1. 打印n层汉诺塔从最左边移动到最右边的全部过程
2. 打印一个字符串的全部子序列
3. 打印一个字符串的全部子序列，要求不要出现重复字面值的子序列
4. 打印一个字符串的全部排列
5. 打印一个字符串的全部排列，要求不要出现重复的排列

模型一：从左往右的尝试模型

这种模型，我们都是从左往右去尝试的。并且在每一步的尝试都会分成几种情况，形如一个n叉树的形式。

题目一:

规定1和A对应、2和B对应、3和C对应...

那么一个数字字符串比如"111"就可以转化为:

"AAA"、"KA"和"AK"

给定一个只有数字字符组成的字符串str，返回有多少种转化结果?

【解法】

从左往右，两种情况:

1. 第一个1转成A，然后剩下的元素继续递归
2. 第一个1不转成A，和后面的元素相结合，然后剩下的元素继续递归
3. 当该数不是1时，如果是2，也可以进行第1、2步的操作，但是在进行第2步的操作时，需要考虑后面一个元素大于6与否
4. 如果是其他数，则只有第1步这个操作。

题目二：

给定两个长度都为N的数组weights和values， weights[i]和values[i]分别代表i号物品的重量和价值。 给定一个正数bag，表示一个载重bag的袋子， 你装的物品不能超过这个重量。 返回你能装下最多的价值是多少?

模型二：范围上尝试的模型

给定一个整形数组arr，代表树枝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。

「绝顶聪明在博弈论中的学术定义」

双方玩家都会使得对方在单独改变策略的时候不会获得更大收益

【解法】

设有一个先手函数

```
int f(arr, L, R)
        if (L == R) return arr[L];
        return max(arr[L] + s(arr, L + 1, R), arr[R] + s(arr, L, R - 1));
```

设有一个后手函数

```
int s(arr, L, R)
        if (L == R) return 0;
        return min(f(arr, L + 1, R), f(arr, L, R - 1));
```

什么是绝顶聪明

这种题严格来讲没什么技巧，就是看脑子，从小到大，从里往外，从小样本到大样本来抽丝剥茧地分析，看看你想的全面不全面。

例子一：

有一个人，要经过一条小河，河中有很多鳄鱼，鳄鱼可以吃人，如果这个鳄鱼吃了人，那么它自己就会进入虚弱状态，被别的鳄鱼吃掉。假设，所有的鳄鱼都是绝顶聪明的，那么这个人怎么过河？

【题解】

那么这种情况下，当河中鳄鱼是偶数个的时候，这个人可以过河；当河中的鳄鱼是奇数个的时候，这个人过不了河。

因为如果是奇数个，那么随便一条鳄鱼上来就可以把这个人吃了，而且，别的鳄鱼还不敢吃这条鳄鱼，因为如果它自己把这条虚弱的鳄鱼吃了，那么它自己就会被别的鳄鱼吃掉。

如果是偶数个，那么就可以过河。因为所有的鳄鱼都害怕自己吃了这个人，然后被别的鳄鱼吃掉，所以就不敢吃这个人了。

例子二：

有5个海盗，要分100个金币，ABCDE这五个人，从前往后，依次提分配方案，如果同意这个方案的人超过了一半，那么就可以这么分，如果没有超过一半，那么提方案的这个人就要被打死。假设所有的海盗都绝顶聪明，并信守诺言，那么A应该怎么分配方案？

【题解】

可以先从一个人的情况看起：

1. 如果只剩下了E一个人，那么他可以随便提方案，100个金币都拿到手中。
2. 如果剩下了DE两个人，那么无论怎样，E都不会同意D的方案，因为E不同意，所以就没有办法超过一半的同意人数，E想的是，我不同意你的方案，然后你死，然后我自己拿这100个金币。
3. 如果剩下了CDE三个人，那么无论怎样，D都会同意C的方案，因为如果他不同意C的方案，那么D自己就会死了，所以C可以随便提方案。
4. 如果剩下了BCDE四个人，那么，因为在3这种情况下D和E都是不会拿到金币的，而C是能拿到100个金币的，所以B不会去拉拢C，B拉拢到D和E就可以了。所以，给他两个一人分一个金币就可以，B自己拿98个。
5. 如果是ABCDE五个人，那么，因为在4这种情况下，B是拿到98个金币的，C是拿不到金币的，D和E只能拿到一个金币，因此，A不会去拉拢B，A只需要拉拢两个人就可以了，很显然，C最容易拉拢，因为C在B统治的情况下是拿不到任何好处的，所以A只需要给C一个金币就可以，而D和E中只需要任选一个就够了，给2金币就可以了。A自己拿97个金币就够了。

例子三：

「欧拉信封问题」

我规定一个村里，每一个人都必须往外寄出一封信(不能寄给自己)，每个人都只能收到一封信。如果你收到了一封信，那其他人就不能向我寄信了。

请问，村里有多少种寄信的方式?

1. 如果村里只有一个人，那么就是0种方法
2. 如果是两个人一个村，那么就是1种方法
3. 如果是三个人一个村，那么就是ABC，然后再一条条地分析这种情况下多少种方法。2种
4. 。。。

再往后我们就会发现，有个递推公式：

f(n) = (n−1) [f(n−2) + f(n−1)]

N皇后问题

这个问题是指，在一个N * N的棋盘上要摆放 N 个皇后，要求任何两个皇后不同行、不同列，也不在同一个斜线上。请告诉我，该N皇后的摆放有多少种?

n = 1；1 n = 2或3；0 n = 8；92

【题解】

实际上就是按行，从上往下，依次由前一行推出下一行哪些地方不能放皇后，然后依次往下找，依次找到所有情况即可。

【时间复杂度】

O(n^n)

暴力递归到动态规划的套路

从最经典的说起

斐波那契数列

[1, 1, 2, 3, 5, 8, ...]

【暴力过程】

```
public static int f(int N) {
        if (N == 1) {
                return 1;
        }
        if (N == 1) {
                return 1;
        }
        return f(N-1) + f(N-2);
}
```

【如何优化】

你把计算过程写一下就会发现：

f(7) = f(6) + f(5)

f(6) = f(5) + f(4)

f(5) = f(4) + f(3)

在这里面你会发现，f(5)重复计算了，f(4)也重复计算了，等等等等。

我没必要重复计算这些值。

暴力过程一定是因为有重复计算才暴力的。

来一道阿里原题

假设有排成一行的N个位置，记为1～N，N一定大于或等于2. 开始时机器人在其中的M位置上（M一定是1～N中的一个）。 如果机器人来到1位置，那么下一步只能往右来到2位置； 如果机器人来到N位置，那么下一步只能往左来到N-1位置； 如果机器人来到中间位置，那么下一步可以往左走或者往右走； 规定机器人必须走K步，最终能来到P位置（P也是1～N中的一个）的方法有多少种?

给定四个参数N、M、K、 P，请返回方法数。

【题解】

同样也是，通过从简单到复杂，找到其规律，找出通项公式，然后就可以递归了。

递归中，和上面斐波那契数列一样，也会有重复的，那我可以整一个缓存，来保存数据，这样免重复计算，从而进行优化。

暴力递归到动态规划

不是所有的暴力递归都可以转成动态规划

但是，所有的动态规划，都来自于暴力递归

【常见的四种尝试模型】

1. 从左往右的尝试模型
2. 范围上的尝试模型
3. 多样本位置全对应的尝试模型
4. 寻找业务限制的尝试模型

前面所述的题目二的那个背包问题：

给定两个长度都为N的数组weights和values， weights[i]和values[i]分别代表i号物品的重量和价值。 给定一个正数bag，表示一个载重bag的袋子， 你装的物品不能超过这个重量。 返回你能装下最多的价值是多少?

【递归解】

递归解就是从第一个物品开始，尝试两种情况，一种是把自己装进去，另一种是不把自己装进去，把自己装进去的情况就是把自己的value也加进去，并把自己的weight也加进去。后续的工作就继续去递归调用即可。

然后选取两种情况下的最大值。

【优化解一】

把重复的解放到一个缓存中，发现重复的解，就从缓存中拿数据即可。

比如说，我有一个数组[2,1,3,5,4]，我没要2，没要1，要了3，等同于我要了2，要了1，没要3，后续的情况的解。这就是重复了。我们把重复解不重复计算即可。

【优化动态规划解】

动态规划解就是，生成一个dp表，将所计算过的结果全部记录下来，最后结果需要哪个结果就从dp表中找相应的结果。

什么是动态规划?

就是我通过暴力递归，找到了递归的一种方法。然后我把所有的递归过程，换成一个动态规划表
在我从头往后计算的过程中，我把所有的计算结果都记录到动态规划表中，当遇到重复值时，
接从动态规划表中获取该元素即可。

最后，我需要哪个结果，就从动态规划表中找哪个结果即可。

通过这样的方式，可以减少递归的重复计算，从而提高效率。

【从暴力递归到动态规划】

给了我一个题目 -> 通过暴力递归等方式去尝试 -> 发现了重复解 -> 方法中有可变参数(暴力递归的方式，不讲究组织) -> 记忆化搜索 -> 精细化组织 -> 得到了经典的动态规划

经典动态规划

上面的动态规划只是一种方式，通过记忆化搜索的方式。

但是，某些问题需要精细化组织。

题目一：

通过这道题来揭示一个动态规划题目的完整优化路径。

【题目】

我有一个数组arr，数组中每一个位置的值是一种货币的面值。假设每一种面值都可以使用任意张。为了简单起见，我们认为arr中都是正数且无重复值。假定给我一个目标值 aim，请问有多少种方式可以组合出这个目标值?

【题解过程】

首先我可以分析出，要想得到这个目标值，我可以从头到尾找这些元素，我可以拿出一个函数int f(0, aim)，意思就是，从arr[0]开始，往后添加那些所有元素，最终得到aim值的所有方法数目。

这里面就涉及到一个循环。

有好多好多种情况:

1. 我要0张arr[0]的元素，后面f(1, aim)的方法数目
2. 我要1张arr[0]的元素，后面f(1, aim-arr[0] * 1) 的方法数目
3. 。。。
4. 直到，我要n张arr[0]的元素，后面aim-n==0或刚刚过了0，后面f(1, aim-arr[0] * n)的方法数目。
5. 那么我用一个dp表来存储 arr.length + 1, aim + 1 个元素即可当作这个缓存。

一定一定要找对可变参数，找对了可变参数，才会是无后效性的

【精细化组织】

画出这张dp表，来分析这些值之间都是如何互相依赖的？

本题，通过分析，我们会发现，对于任何一个普通元素，它都会依赖下一行的元素。

而我们已有的是f(N, 0) = 1，f(N, 其他) = 0; 我们要求的是f(0, aim)，正好是符合这样的依赖关系查找顺序的。

然后，我在这张表中，我会发现，在每次计算当前值的时候，还会依赖当前行的前面的元素，这就会使得我们又有了重复计算。

我们又发现了重复元素，这里又可以再动态规划。

题目二：

给定一个字符串str，给定一个字符串类型的数组arr。 arr里的每一个字符串，代表一张贴纸，你可以把单个字符剪开使用，目的是拼出str来。 返回需要多少张贴纸可以完成这个任务。

> 例子：str = "babac", arr = {"ba", "c", "abcd"} 至少需要两张贴纸"ba"和"abcd"，因为使用这两张贴纸，把每一个字符单独剪开，含有2个a、2个b、1个c。是可以拼出str的。所以返回2。

总结：

如何找到某个问题的动态规划方式？

1. 设计暴力递归：重要原则 + 4种常见尝试模型！重点！
2. 分析有没有重复解：套路解决
3. 用记忆化搜索 -> 用严格表结构实现动态规划：套路解决
4. 看看能否继续优化：套路解决

面试中涉及暴力递归过程的原则：

1. 每一个可变参数的类型，请一定不要比int类型更加复杂
2. 原则1可以违反，让类型突破到一维线性结构，那必须是唯一可变参数
3. 如果发现原则1被违反，但不违反2原则，只需要做到记忆化搜索即可
4. 可变参数的个数，能少则少

常见的四种常见模型（再说一遍）

1. 从左往右的尝试模型

   如背包问题（重量和价值）、数字字符串转化成字母的形式

2. 范围上的尝试模型

   纸牌博弈问题

3. 多样本位置全对应的尝试模型

   后面来讲

4. 寻找业务限制的尝试模型

   后面来讲

暴力递归到动态规划的套路

1. 已经有了一个不违反原则的暴力递归，而且的确存在解的重复调用
2. 找到哪些参数的变化会影响返回值，对每一个列出变化范围
3. 参数间的所有的组合数量，意味着表大小
4. 记忆化搜索的方法就是傻缓存，非常容易得到
5. 规定好严格表的大小，分析位置的依赖顺序，然后从基础填写到最终解
6. 对于有枚举行为的决策过程，进一步优化

动态规划的进一步优化

1. 空间压缩
2. 状态化简
3. 四边形不等式
4. 等等。。。

尝试模型三：多样本位置全对应的尝试模型

题目：

两个字符串的最长公共子序列问题

[例] str1 = "a123bc"; str2 = "12de3fz";

如何求他俩的最长公共子序列?

我可以建立一张dp表，这张表是个二维表，每个维度长度分别为这两个str的长度。

在这张表中存储的是，到了当前 i, j 位置时，从 0-i 的 str1 和从 0-j 的 str2 的最长公共子序列的长度。

「步骤」

1. 然后我从(0,0)点开始往后填这张表，我就会发现，我每次只增加一个元素的话，很容易得到有下面几种可能：
    1. 这个元素能和另一个str里的某个元素相匹配，那么这个地方就可以加一
    2. 如果不能和另一个str里的任意一个元素相匹配，那么就保持原样

「通过上面的分析」

有以下几种递归的情况：

假设str1从0到i，str2从0到j

1. 相匹配的最长公共子序列不以str1的i结尾，也不以str2的j结尾，那么这种情况下，dp[i, j] = dp[i-1,j-1]
2. 相匹配的最长公共子序列以str1的i结尾，而不以str2的j结尾，那么这种情况下，dp[i, j] = dp[i, j-1]
3. 相匹配的最长公共子序列不以str1的i结尾，而以str2的j结尾，那么这种情况下，dp[i, j] = dp[i-1, j]
4. 相匹配的最长公共子序列既以str1的i结尾，又以str2的j结尾，那么这种情况下，dp[i, j] = dp[i-1, j-1] + 1

尝试模型四：寻找业务限制的尝试模型

题目：

给定一个数组，代表每个人喝完咖啡准备刷杯子的时间 只有一台咖啡机，依次只能洗一个杯子，时间耗费a，洗完才能洗下一杯 每个咖啡杯也可以自己挥发干净，时间耗费b，咖啡杯可以并行挥发 返回让所有咖啡杯变干净的最早完成时间 三个参数：int[] arr、int a、int b

「模型」

这也是个从左往右的尝试模型，但是需要看该业务所需的时间，最长的时间是什么，所以是个业务限制的尝试模型。

先遍历该数组一遍，然后把最长的耗费时间求出来，这就是其中的一个限制参数的值。所以叫「业务限制」的尝试模型。

【两个可变参数】

遍历到了的当前杯子index、洗碗机排队等待队列中当前已到达的时间 这两个参数

每次递归的意思就是，从当前杯子开始，到最后一个杯子处理好。然后返回一个从当前杯子开始到最后一个杯子处理好，最小的花费时间。

【然后改动态规划】

改的过程，就是很显然，两个参数。其中后面那个参数不确定。所以我们需要通过找出最长耗费时间来确定其最大长度，然后放到dp表中。

添加评论

(*必填)怎么称呼你?

(*必填)你的联系邮箱

你的网站地址(可不填)

(*必填)请输入验证码

(*必填)请输入你的评论

提交