

记忆化

Memoization and Dynamic Programming

在做树递归时经常遇見重复用同样的参数调用递归函数.

为了简化流程、提高效率, 可以使用记忆化的方法减少递归函数的调用.

具体写法略.

我们可以用字典存储输入数据(以传入参数构成的元组作为键)

亦可在数据格式明确的情况下(如 count-change)用列表(多维列表)维护记忆化内容.

用列表的效率比字典高. 字典尚需查找(主要是用特定算法将键转化为索引这一步较为困难)

列表在索引一定的情况下还是比较快的.

那么, 某种程度上, 用列表完成的记忆化搜索与动态规划本质上是一致的.

我们仍以 count-change 为例. 假设需要找零的钱为 57. memo[i][j] 存储用 j 种零钱找 i 元的方案数.

首先, memo[0][0] 肯定是 0. 对 count-change() 的跟踪可以发现一开始全是这类值最先加入字典.

然后, 计算所有的 memo[i][1]. 它们要调用的无非是 memo[i-1][1] 和 memo[i][0]. 从 memo[0][1] 开始即可.

再是计算所有的 memo[i][2], ..., memo[i][6]. 即可得到答案.

上面这个过程就是动态规划 (dynamic programming, dp)

如果说搜索是“分塔成沙”, 那么动规就是“聚沙成塔”, 从基本情况开始推及复杂情况.