记忆化搜索：参考lintcode 109，利用Hash表来存储搜索的中间结果，避免重复计算。

将函数的计算结果保存下来，下次以同样的参数访问时，直接返回保存下来的结果。

应用记忆化搜索时，函数必须有返回值，并且返回值只跟输入的参数有关，跟全局变量无关。和系统设计中的cache比较像。

记忆化搜索可以将指数级别的时间复杂度变成多项式级别。

记忆化搜索是利用搜索的方式来实现了动态规划，它是动态规划的一种实现方式。

lintcode 1300，如果直接用记忆化搜索，会导致stackoverflow，直接用记忆化搜索，会出现栈溢出的问题，因为递归深度太深，时间复杂度为O(n)，递归栈的深度也是O(n)，正因为时间复杂度为O(n)，所以n可以取得比较大导致递归栈深度也太深；如果时间复杂度是O(n^2)，递归栈深度为O(n)，反而溢出的可能性会变小，因为如果n较大，在递归栈溢出前，已经超时了。

记忆化搜索基于递归实现，所以它不同适合解决O(n)时间复杂度的动态规划。

动态规划(Dynamic Programming)的四要素：是一种算法的思想，而不是具体实现。

动态规划的核心思想是：由大化小，大规模问题的依赖小规模问题的计算思想。

分治法和动态规划的区别：分治法进行分治时，子问题没有重复问题，例如二叉树。而动态规划的子问题会有重复的情况。

动态规划DP和贪心法Greedy的区别：

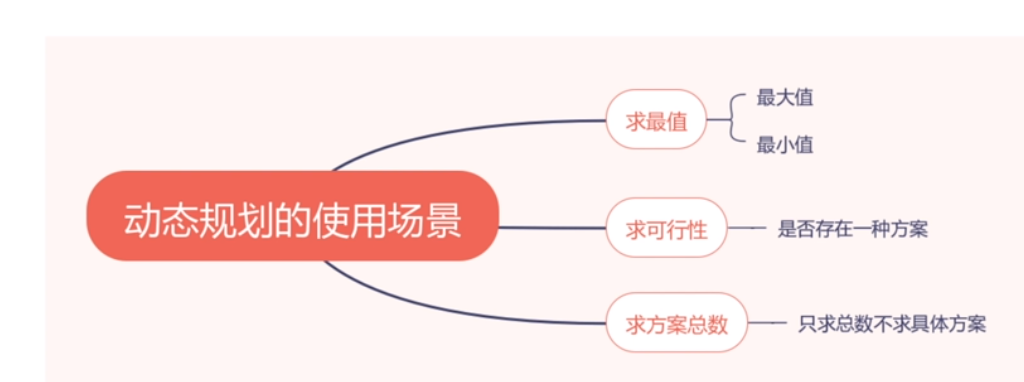
动态规划为了长远的利益损失当前利益，而贪心法只追求当前利益最大化。

动态规划的两种常用实现方法：记忆化搜索（递归）和for循环实现（递推）。

动态规划的四要素：

1. 状态(state)：用dp[i]或dp[i][j]描述小规模子问题的答案，实际就是将问题抽象出来，相当于递归中的定义。
2. 状态转移方程：大问题如何拆解为小问题，相当于递归的拆解。
3. 初始化：设定无法拆解的极小状态，也就是对无法在状态转移方程中进行计算的某些状态值进行初始化，相当于递归的出口。
4. 答案：最后要求的答案是什么。

动态规划的使用场景：



动态规划的题目类型：



动态规划最难得的是其状态，其次是状态转移方程。

坐标型动态规划：

dp[i]表示从起点到坐标i的最优值/方案数/可行性

dp[i][j]表示从起点到坐标i,j的最优值/方案数/可行性

前缀型之划分型：开空间时，一定要多开1行1列的空间，因为有前0个的概念。

dp[i]表示前i个字符的最优值/方案数/可行性

dp[i][j]表示前i个字符划分为j个部分的最值/方案数/可行性

前缀型之匹配型：

dp[i][j]表示第一个字符串的前i个字符匹配第二个字符串的前j个字符的最优值/方案数/可行性

区间型：

dp[i][j]表示区间i~j的最优值/可行性/方案数

背包型：

dp[i][j]表示前i个物品里选出一些物品组成和为j的大小的最优值/方案数/可行性

动态规划时，如果通过递推的方式(for循环)来实现，要注意状态之间的依赖关系，不要形成循环依赖，尤其对于dp[i][j]这样的情况，i和j对应的循环顺序要正确。

背包问题：动态规划与深度优先搜索的时间复杂度比较。

如果使用组合型DFS的话，时间复杂度为。n个数，每个数都有选或不选两种情况

如果使用动态规划，时间复杂度为。

但动态规划不一定比DFS要好，考虑下列的极端情况：

[1, 1000, 1000000]，m=1001001，这种情况下，DFS的效率要远高于动态规划。