

# Project 3:动态规划

## 1 简介

动态规划(dynamic programming)是运筹学的一个分支，是求解决策过程最优化的数学方法。动态规划一般可分为线性动规，区域动规，树形动规，背包动规四类。

该项目由三道动态规划编程题目和一份报告文档组成。你除了需要完成对应三道题目的三个函数，还需要编写一份报告文档，来说明你的解题思路和代码实现思路，并回答一些该文档中提出的问题。

在命令行中输入 `make lab3`，编译程序；

在命令行中输入 `make run`，或者在编译后输入 `./lab3`，启动程序，检查正确性。

在命令行中输入 `make clean`，删除当前编译的版本。

## 2 评分标准

该项目满分共 100 分，分为三项：

(1) 完成三道题目，共 75 分。以通过所有测试用例为准。测试用例在 `./lab3/test` 和 `./lab3/answer` 中。

(2) 代码规范，共 5 分。例如分配堆空间的回收、变量命名规范等等。

(3) 报告文档，共 20 分。

## 3 从简单的例子说起

part1 是一道简单的动态规划题目：给定不同面额的硬币 $\{N\}$ 和一个总金额  $M$ 。

写出一个函数来计算可以凑成总金额的硬币组合数。假设每一种面额的硬币有无限个。

样例：总金额=5,硬币面额=[1,2,5]。程序将输出一个正整数 4,代表：

$$5=1+1+1+1+1$$

$$5=1+1+1+2$$

$$5=1+2+2$$

$$5=5$$

注：这部分代码需完成在./lab3/lab3.cpp 的 func1 中。

这是一个完全背包问题。该如何思考这个问题呢？根据动态规划的思想，首先我们应该确定动态规划中需要不断计算的中间变量是什么。很显然，在此题中是从 1 开始直到总金额中每一个金额被硬币组合出的组合数。为此，构建一个  $M \times |N|$  的矩阵，用来存放中间结果，如下图所示。

硬币\金额	0	1	2	3	4	5
0						
1						
2						
5						

接着，考虑初始条件，当没有硬币但总金额为 0 时，我们应该默认其有一种组合（可以理解为集合里的  $\emptyset$ ），即在  $dp[0][0]$ 处填上 1，显然，没有硬币是没法组合出大于零的金额的，在第一行其余位置填 0。

我们试图寻找规律，把只用 1 元硬币的那行填上。很显然，都只有一种，在第二行填上 1。

当可以使用 1 元和 2 元硬币时，情况发生了变化。总金额为 2 时，组合数发生了增加。除了仅用 1 元硬币的一种方法，另一种方法是先用  $\emptyset$  拼出一个 0 元，

再用一个 2 元硬币拼出 2 元的总金额，共两种。是不是有些抽象？那么我们再来分析  $dp[2][4]$  中的结果。当不用 2 元硬币时，4 元的总金额有一种仅用 1 元硬币的组合；当使用 2 元硬币时，可以先组合出 2 元的总金额（共两种，这个数据之前已经被计算出来存在哪里？），然后再添上一枚 2 元硬币来达到 4 元的总金额。那么  $dp[2][4]=1+2=3$ 。

把这个表格填完，找一找规律吧！

问题：

- (1) 设总金额为 13，硬币= $\{1,2,5,10\}$ ，画出动态规划的表格并填满结果。
- (2) 这个动态规划的时间复杂度是多少？
- (3) 请将这个动态规划的空间复杂度优化到  $O(M)$ ，并指出你是如何优化的。

## 4 问题变得复杂起来 ……

一群人落在了绝地岛闵达凰上，它们成一个圈降落在地图的边缘。安全区最终会刷新在这个岛的中心艾西艾斯峰顶，于是所有人都要向这个峰顶前进。一路上，他们会遇到和他们有一样目标的异类，并试图击败对方。他们不断地前进，不断地决斗。最终只有一个人会活着到达艾西艾斯峰顶。到底是苟到最后击杀第二名成功吃鸡，还是选择淘汰二十九名玩家无尽杀戮，这取决于你的选择。

游戏规则：1.所有人围成一个环；2.所有人只可能会碰到他当前身边的两人的其中之一并和他决斗，且必定只存活一人；3.淘汰的人被移出环（即 A-B-C-D-A 成环，A 若击败了 B，A 的身边就变成了 C 或 D）；4.所有决斗的发生顺序都是随机的；5.最终只存活下来一人（俗称吃鸡）。

那么现在我们知道这所有人两两对决的优胜关系表，问决斗顺序随机发生的

情况下，有多少人可能最终吃鸡？

注意，优胜关系是不传递的。比如 A-B-C-A，A 胜 B，B 胜 C，C 胜 A，那么谁吃鸡完全取决于决斗的顺序。

样例：共 4 人，其优胜矩阵如下：

A胜B	A	B	C	D
A		1	0	1
B	0		1	1
C	1	0		1
D	0	0	0	

显然，D 谁都赢不了，D 必不可能吃鸡；而 A,B,C 都有可能吃鸡。所以对于这个样例，程序应该输出 3。

注：这部分代码需完成在 [./lab3/lab3.cpp](#) 的 `func2` 中，其中参数 `conquer` 就是上述矩阵，`conquer[i][j]` 代表 i 能战胜 j。

看上去这个问题似乎有些无从下手。但是我们要知道，**动态规划的核心思想是先求解子问题，然后从这些子问题得到原问题的解**。那么这个问题的原问题是什么？子问题又是什么？试着把原问题（谁能吃鸡？）用其它的方式表达出来，看看能不能把它拆成子问题。

从另一个角度想，既然是动态规划，那么我们就需要像 part1 一样构建一个矩阵。这个矩阵的行和列会代表什么？每个值又代表了什么？这个矩阵的意义和生长方式可能和 part1 的意义不太一样，不要被简单的 part1 禁锢住了思路。

问题：

(1) 请分析./lab3/test/test2.txt 中 case3 (amount=6) 究竟是哪个倒霉蛋不可能吃鸡，并写出其余每个人吃鸡的一种随机决斗过程。

(2) 这个程序的时间复杂度和空间复杂度是多少？能否继续优化？你可以写下你的优化思路或者在自己的代码中实现它，或是说明你的程序时间复杂度已经达到最佳了。

## 5 期望 DP 与解方程组

part3 是一道比较困难的题目。

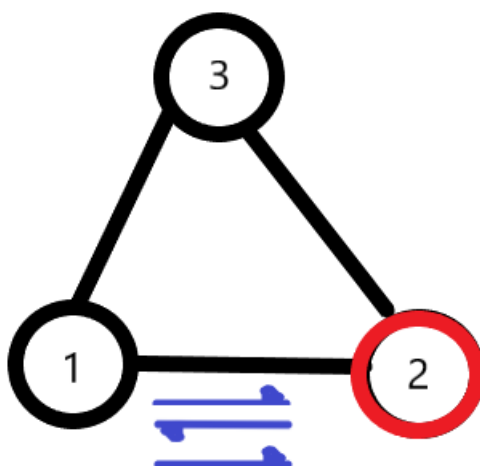
小 Z 来到一个古墓去寻找宝藏。古墓中有非常多的路口和岔路，有些路口有陷阱，小 Z 在每次经过路口  $i$  的陷阱的时候都要掉  $A[i]$  点血，而且陷阱是永久有效的（即小 Z 每到一次路口  $i$  就要掉  $A[i]$  点血）。幸运的是，有一些路口没有陷阱。可不幸的是，小 Z 是个路痴，他完全无法判断他走过哪里，要去哪里；他只能在每一个路口**随机（等概率地）**走向某一条岔路到达下一个路口。小 Z 现在在古墓的入口处（即路口 1），这里没有陷阱；宝藏藏匿在路口  $n$ ，那里也没有陷阱。

而你万万没有想到的是，你是这个古墓的守护者。你知道这个古墓所有的构造（包括它的路口、岔路和陷阱的情况），现在你需要计算出小 Z 能见到宝藏的期望概率。

注：这部分代码需完成在./lab3/lab3.cpp 的 func3 及一些其他函数中，其中参数  $n$  是路口数量； $hp$  是小 Z 的初始血量；数组  $damage$  是  $n$  个数据，代表  $n$  个路口陷阱的伤害（无陷阱处为 0，保证路口 1 和  $n$  处无陷阱）；数据  $edges$  是  $2*$ （岔路条数）个数据，每 2 个数据是一条边，边都是双向的。

举个例子：

有个三角形路口 1, 2, 3。小 Z 有 2 点血。小 Z 在 1，宝藏在 3，陷阱在 2（伤害为 1）。那么它的结果为 0.875。小 Z 在血量降到 0 之前走到 3 就算成功，所以它失败的唯一路径是 1-2-1-2，每次寻路都是随机的，三次都走错的概率是  $0.5^3 = 0.125$ ，那么成功走到 3 的概率就是 0.875。



(图中每个蓝色路径的概率都是 0.5，经过三次“错误”的决策最终失败)

考虑这个问题，想法是比较直接的。以  $hp \times n$  建立动态规划的矩阵是一个不错的想法，把  $f(i,j)$  设置为到达  $j$  点时剩下  $i$  点血的概率( 初始值  $f(hp,1)=1$  )，相信你能很快的写出状态转移方程（这是问题 1）。但事情就这么完了吗？你可以试一试用这种思路填上述例子的矩阵，看看能不能得出正确的结果。

你会发现，因为有一些路口的伤害值为 0，好像在填这些没有陷阱的路口的格子是非常棘手的事情……（可惜！没有陷阱对小 Z 来说真是幸运极了，但是对你来说好像不是这样……）

由于我并不能告诉你问题 1 的答案，可能接下来的讲述会抽象一些：对于一个  $f(i,j)$ ，如果它伤害值  $damage[j]>0$ ，显然是简单的；如果它伤害值=0，它会成为一个方程（因为同  $hp$  层其他的值也有一些未知数），将该  $hp$  层的所有伤害

值为 0 的节点  $j$  写出的方程联立，你会得到一个方程组（这是问题 2）。

线 性 代 数 ！

这是绕不开的问题了。计算机通常用**高斯消元法**（gauss-jordan 消去法）求解大规模的非齐次线性方程组。有关高斯消元法，可以自行使用搜索引擎学习。如果用高斯消元法解决了方程组的问题，你将可以顺利地计算出同  $hp$  层的概率值，继而计算出最终结果（这次是真的）。

试试吧！

问题：

(1) 写出这个算法的状态转移方程。

(2) 对于某个特定的  $hp$ ，所有无陷阱节点构成的方程组，这个方程组的未知数是什么？系数是什么？常数项又是什么？写出它的增广矩阵。PS:如果你觉得一般情况很抽象的话，可以以 [part3-case3](#) 在  $hp=2$  时的方程组为特例考虑。

(3) 这个算法的时间复杂度和空间复杂度是多少？如果按照该文档截至今的思路，算法的时间复杂度还有提升的空间。上一小问给了你什么启发？如果你能回答出来，你将会获得更高一点的分；如果你能在你的程序中实现，你将获得满分。[Hint:你需要从高斯消元法的时间复杂度去考虑这个问题。](#)

## 6 后记

你需要将 lab3.cpp 和报告文档 answer.pdf 打包成一个压缩文件，并命名为[学号]\_[姓名]\_proj3.zip。**本次作业禁止一切形式的抄袭**。助教会代码进行查重，重复率过高的代码编写者将被助教特约答辩，视答辩情况对分数严肃处理。