

CS3110: Assignment 7

(Dynamic Programming)

July 12 2017

Q1: [25 marks] You are given a robot with only 3 possible instructions:

A: move forward 1 foot

B: move forward 2 feet

C: move forward 3 feet

Complete the following steps to design an algorithm that given n , prints **the number** of all possible chain of instructions that can move the robot exactly n feet from its current position.

- 1) Formulate the problem in a recurrence form
- 2) Implement (pseudocode) the recurrence directly and calculate its complexity
- 3) Use memoization and reimplement the algorithm
- 4) What is the complexity of the new algorithm

Q2. [25 marks] There is another robot, and this one can move on a gridded surface and has two instructions:

A: Move Forward

B: Move Down

Part 1 [10 marks]: We want to calculate the number of different ways that the robot can get to point (x,y) from $(0,0)$

- 1) Formulate the problem in a recurrence form
- 2) Use Memoization rewrite the algorithm
- 3) Calculate the complexity of the algorithm

Part 2 [15 marks]: Now assume there are cells that are forbidden,

- 1) write an algorithm that prints a chain of instructions that can help the robot navigate to (x,y) , staying away from the forbidden cells
- 2) What is the complexity of the algorithm?

Note that in part A the algorithm needs to return the **number** of the paths, and in part B, should print out **one solution** among possible many of them




Q3. [30 marks+10 Bonus] We are given the description of a game, as follows.




Emoji generation Game:




The players agree on a set of symbols (marked in red) and a set of emojis. Player A designs a set of “**replacement rules**”, and an ordered list of **emojis**. Player B should generate the given list by applying the replacement rules. For example:



The players agree on these symbols: {, , , } ( is mandatory and should always be one of the symbols) and these emojis {, , , }



Player A: Generate this list       using the following rules

R0:  \rightarrow  


R1:  \rightarrow  

R2:  \rightarrow  



R3:  \rightarrow 

R4:  \rightarrow 

R5:  \rightarrow  

R6:  \rightarrow  

R7:  \rightarrow 

R8:  \rightarrow 

R9:  \rightarrow  

R10:  \rightarrow 

Player B:

Start from R0 and replace the *blank* with:

Replace the first symbol using R2 and the second symbol using R5

Replace the first symbol using R6, and last symbol using R2

Now from left to right, apply the following rules on every symbol:
R10, R3, R10, R7, R8, R10



We have the following restrictions on the set of the rules:

- 1) You should Always start from Blank Symbol, ϵ
- 2) The left side should be a symbol (red shapes)
- 3) Right side can be only **2** symbols or **1** emoji
- 4) You can apply any rule any number of times to replace a symbol

Design a program that given a set of rules and the list of the emojis, returns one possible chain of rules that can generate the emojis list. Follow these steps to solve the problem:

- A) Using a tree representation, try to capture and understand the recursive nature of the solution
- B) Explain how it exhibits optimal-substructure
- C) Design the steps of the algorithm, (verbal but step by step and clear explanation is enough)
- D) **[10 Bonus marks]**: Write an iterative solution.

Hint: This problem is more similar to the matrix chain problem, however it is slightly different, which makes it a bit more complicated.

You are allowed to search and properly give reference for the following problem, however do not copy, but give your own interpretation:

Q6 [20 marks] . Just because almost all the problems we solved in last few weeks exhibited optimal substructure, we cannot always assume it! Find at least two problems that optimal substructure does not apply, and clearly explain the reason.