Programming 2 - Assignment 6
Dynamic Programming

December, 2022

Learning Objectives

- Improve the speed of complex algorithms by using dynamic programming techniques.

Part 1: A Game of Chance (Revisited)

In `DiceSolver.java` you can find an implementation of a recursive algorithm that calculates the probability of winning in a game of chance (for details on what it does, see the bonus of assignment 5 as well as the documentation). Create a new function `getWinChanceDynamic` (choose the function parameters based on your implementation) that uses dynamic programming techniques to improve the speed of the program.

In `DiceTester.java` (found in the helperFiles) you will find a script that automatically compares your implementation with the one already provided in `DiceSolver.java`. You are free to use this main (or not), but this script will automatically call the (empty) function `getWinChanceDynamic(int maximumValue, int sides)`. Treat this function as a helper function for your own algorithm (call your own algorithm from this function with the extra variables that you need initialised).

Part 2: Pathfinder (Revisited)

In `PathFinder.java` you can find an implementation of a recursive search algorithm that finds the length of the shortest path from a starting location to the a goal (for details on what it does, see the bonus of assignment 5 as well as the documentation). Create a new function

`getShortestPathDynamic` (choose the function parameters based on your implementation) that uses dynamic programming techniques to improve the speed of the program.

In `DiceTester.java` (found in the helperFiles) you can find a script that automatically compares your implementation with the one provided in `PathFinder.java`. You are free to use this main (or not), but this script will automatically call the (empty) function `getQuickestPathDynamic(int maximumValue, int sides)`. Treat this function as a helper function for your own algorithm (call your own algorithm from this function with the extra variables that you need initialised).

Part 3: Complexity Interaction (Bonus)

Write an analysis about the trade-off between time and space-complexity that dynamic programming exploits. What are the time and space-complexities of the algorithms without dynamic programming? How does that compare to the complexities of the dynamic algorithms?

You can use the console output from `PathTester.java` and `DiceTester.java` to help you get some indication of the time complexities.

Handing in the Assignment

Make sure that you hand in your exercise in the following format.

- Make sure your group name and students names and numbers are in each .java file in comments.
- Zip your project folder.
- Include a PDF with the analysis from part 3 if you have done the bonus this week.
- Rename your zip folder to `Assignment_6_group_x.zip`, where `X` is your group's number.