COP3503 Computer Science 2
Dr. Andrew Steinberg
Spring 2025

Programming Assignment 4
Hop or Step Game
Due: 4/6/2025 at 11:59pm

Objective: Students will apply concepts of dynamic programming and implement three solutions using all three techniques including memoization and tabulation.

Assignment Description: We are going to play a fun game in this assignment. You are given a set of adjacent squares each containing a positive value (some can be the same) that represents a cost of stepping on the square. You will start at the beginning of the set of squares before even stepping on the first one. As a player you can only make one of two moves when walking across the set of squares. At each square you can either take a step immediately to the adjacent square or hop over the square to the following square. The goal is utilizing a set of squares that results in the minimum total cost. In other words, what is the minimum cost to reach either the last or second last square. Consider the following two examples.

Example 1:

Let's say we are given three squares with the following cost.

10	15	20
1	2	3

At the beginning we had not stepped on any squares yet, so the total cost is 0. We can either step on square 1 (cost is 10) or hop to square 2 (cost is 15). Our goal again is to reach either the last square (cost is 20) or the second last square (cost is 15). Overall, we can visually see that the total minimum cost of walking across the squares should be 15.

Example 2:

Let's say we are given three squares with the following cost.

25	48	37	66	58	46
1	2	3	4	5	6

At the beginning we had not stepped on any squares yet, so the total cost is 0. We can either step on square 1 (cost is 25) or hop to square 2 (cost is 48). Our goal again is to reach either the last square (cost is 46) or the second last square (cost is 58). Overall, we can visually see that the total minimum cost of walking across the squares should be 120. Here is the derivation of the optimal solution. We would first initially hop to square 1 (cost is 25). Then we would hop to square 3 (cost is 37 which would result in a total cost of 62). Last, we would hop to square 5 (cost is 58 which would result in a total cost of 120).

Your goal is to implement three solutions that will generate the same results, but two of them utilize the concepts of dynamic programming. One of them will utilize a recursive approach.

COP3503 Computer Science 2
Dr. Andrew Steinberg
Spring 2025

For this assignment, you must follow these **requirements**.

1. Create a class called `HopStepGame`. In this class, you will implement the three solutions that should generate the same result.
2. You will not need to actually implement any constructors in this assignment as we are using the default constructor only (the one that is provided for you).
3. You will implement 3 non-static methods that return the same primitive integer value representing the minimum total cost.
    a. The first method will be called `minCost`. This method will take two parameters. The first parameter is a reference to an integer array. This represents the set of squares with their cost. You may assume that the values are proper and abide by the rules of the game. The second parameter is an integer that represents the total number of squares in the array. This method will not use any of the dynamic programming techniques. You must apply a recursive approach. Hint, think about top-down. The running time must be bounded by $O(2^n)$ where n is the number of steps. Please make sure to implement the method based on the description in order to receive full points. Even if the method generates the correct value, the graders will also be looking at the code itself. If the code does not abide by the description, points will be deducted. If the top-down recursive approach is not applied, you will receive a score of 0 in the implementation category.
    b. The second method will extend the previous first method. This method is named `minCostMemoization`. Once you successfully implement the first method, you will apply one of the dynamic programming techniques. In the second method you will utilize **memoization**. The method will take three parameters. The first two parameters are the same values from the first method `minCost`. The additional third parameter is an additional integer array that will store previous results. The array will be declared in the driver and the reference will be passed. The running time must be faster than the general top-down recursive approach. Please make sure to implement the method based on the description in order to receive full points. Even if the method generates the correct value, the graders will also be looking at the code itself. If the code does not abide by the description, points will be deducted. If top-down memoization is not applied, you will receive a score of 0 in the implementation category.
    c. The third method will apply the second dynamic programming technique **tabulation**. This method is called `minCostTabulation`. The method takes only one parameter. The parameter is a reference to an integer array that represets the cost of each square. This is the same array from the first parameter of the previous two methods. This method must apply the bottom-up tabulation approach. The running time must be faster than the general top-down recursive approach. Please make sure to implement the method based on the description in order to receive full points. Even if the method generates the correct value, the

graders will also be looking at the code itself. If the code does not abide by the description, points will be deducted. If tabulation is not applied, you will receive a score of 0 in the implementation category.

4. Make all methods public and class attribute private. It's good practice!
5. You may create additional helper methods and attributes if needed as long as they are implemented and called in your solution file and NOT called from the driver file. Adding extra methods to call in the driver file will not match to what the graders will use evaluate your code. This will result in a low score with no change to be applied!
6. If your code does not compile or is stuck in an infinite loop, you will receive no credit on any of the rubric categories except for code style/comments, assignment directions, and code submission/header.

A driver file (`HopStepGameDriver.java`) has been provided for you to show you how the methods are called along with 5 test cases. Please note that test case 5 will take some chunk of seconds to run due to the running time. You were also provided a test script that the graders will use in evaluating your code. It is your responsibility to run the script with your code before submitting to avoid any issues with the grading process.

What to submit: Submit a file called `HopStepGame.java` to webcourses. You are not required to submit the driver file as that will be provided for the graders to test your code. Please make sure the driver file provided works for your code. Any name changes may cause your program not to work when graded, which will result in a lower score on the assignment and would not be changed.

What happens if my code doesn't compile or is stuck in an infinite loop? How will my code be graded? If you are code falls under not compiling or is stuck in an infinite loop, you will not be eligible to receive any points on the test cases, minCost implementations (all three versions), categories. That means you would receive 0 points in those respective categories. The only full points you would be eligible for are code style, comments, and code submission/header if directions were followed. You will also receive low marks if your code does not compile.

Important Note for running Eustis: Many of you are probably using IDEs like Netbeans and Eclipse to build your Java Solutions. Please note that some of these IDEs will automatically place your Java file in some sort of package. Please make sure your Java file is not defined in some package as this can result package private errors. Any such error that occurs during the grading will not be fixed and points will be deducted as such in accordance with the respective categories in the rubric. Also, DO NOT create a main method in your solution file!! This will result in your code not running properly with the driver file which will result in points being deducted from the respective categories.