Matthew Agboglo
Professor Moradi
COMP 285
11/30/2023

Staircase Program Analysis

Prompt: You will explore and analyze different algorithms to solve the classic problem of climbing a staircase with n steps. Each time, you can either climb 1 or 2 steps. The goal is to find the number of distinct ways to reach the top of the staircase.

Test Case: The number of steps used for each approach was 35. I will be rounding the runtimes to the third decimal place for simplicity. I will also list the results for each case here.

Naive:
- # of ways: 14930352
- Execution time: 1.778 Seconds

Bottom:
- # of ways: 1493052
- Execution time: 0.000 Seconds

Top:
- # of ways: 1493052
- Execution time: 3.043 Seconds

First Case: Naive/Base Approach

The runtime for the base approach is O(2^n). This is because the function is being called repeatedly without having any values stored aside from the base case. Causing the code to double its workload. The naive case uses only recursive programming which is not optimal for large values.

Second Case: Bottom-up Approach

The runtime of this case is O(n). This is because at each step the subproblems are solved and stored. The values update as the process continues until it reaches the desired value. The program only has to go through the loop once. The bottom-up uses dynamic programming and can handle large values more efficiently than the naive case.

Third Case: Top-Down Approach

The runtime of this case is O(n). This is because values are stored but it starts from the original problem and uses subproblems to reach the base case. This approach uses memoization to help it be efficient.

Analysis:

Out of all three approaches used the Bottom up approach seems to be the most efficient. It can handle large values and it's execution time is faster than the other approaches. The Naive case has a runtime that should be longer than top down but it's execution time is still faster. It may be an error in my programs but I am still not sure why the naive approach is faster. I assume it could be due to the top down approach having to work backwards and may take more time to come to the answer. So I'd say the top down approach is the slowest according the the test provided. Maybe at larger values the Top-down approach would be faster then the naive approach. While the execution times and approaches for the problem varied, the answer for the

prompt was the same across them all. I believe that each approach is dependent on the situation.