Drew Goldman

Drainage - Description

The Drainage problem requires a relatively simple Dynamic algorithm, particularly in the form of memoization. The objective of this problem was to determine the longest sequence of grid locations between which water can flow, given that water flows from higher elevations to lower elevations. I approached this problem through two major steps: iterate through each cell in the grid of input and for each cell, the algorithm called a helper method, drainage(), which determined the current optimal solution. After all the cells have been iterated through, the maximum length path is correctly returned. The drainage() method required four edge cases to check paths throughout the grid. The algorithm accounted for if going left, right, up, or down would cause an out of bounds error. Therefore, it would recursively move in one of those four directions each time, as long as it didn't go out of bounds, until either the remaining solution has previously been memoized and it can look up the remaining solution in constant time, or it will return the maximum of all the different movements. Of all the lengths calculated by the drainage calls from traversing through the matrix, the maximum distance is kept track of and ultimately returned. Because the dynamic memoization makes the drainage() method run in constant time, the **total runtime of this algorithm is Θ(n*m) for each test case** .

No pseudocode required, as the code is available to the grader via Gradescope.