

Homework 7: Unique Paths LeetCode Challenge

Hayden C. Daly

September 28, 2020

1 Approach

Started with the naive approach of using recursion and backtracking but then realized the problem could be easily done with dynamic programming and memoization. Started with using a grid directly corresponding to the x , y values but realized this was wasted space complexity since it was doing horizontal checks on a row. Using the prior values, I was able to save all of the sums rather than backtracking values reducing the complexity significantly compared to the recursive approach. This change optimized the space complexity from $O(m*n)$ to $O(n)$. In an effort to further optimize the space complexity, I made the row the minimum of m , n to marginally save space on edge cases where one value is significantly larger than the other.

```
1 int uniquePaths(int m, int n) {
2     // Edge case
3     if (m == 0 || n == 0) return 0;
4
5     // Initialize array of all ones in O(n) -> Uses smaller size to save space complexity
6     int row[min(m, n)], x = 1, y;
7     std::fill_n(row, min(m, n), 1);
8
9     // Iterate over grid in O(n*m)
10    for (; x < max(m, n); ++x)
11        for (y = 1; y < min(m, n); ++y)
12            // Only need to do row because horizontal access
13            row[y] += row[y-1];
14
15    // Return last index of array
16    return row[min(m, n)-1];
17 }
```

Listing 1: Submitted Algorithm

The algorithm implementation shown above resulted in a program with a time complexity of $O(m*n)$ and a space complexity of $O(n)$. Given the LeetCode submission percentiles, I believe my implementation is optimal in terms of time complexity but there exists room for improvement in terms of the space. I am not certain but I believe since the problem is not dependent upon data rather just the size of the matrix, there must exist a mathematical representation for the problem which would have a constant space complexity.

2 Proof of Completion

Success Details >

Runtime: 0 ms, faster than 100.00% of C++ online submissions for Unique Paths.

Memory Usage: 6.1 MB, less than 52.13% of C++ online submissions for Unique Paths.

Next challenges:

Unique Paths II

Dungeon Game

Show off your acceptance:



```
1 class Solution {
2 public:
3     int uniquePaths(int m, int n) {
4         // Edge case
5         if (m == 0 || n == 0) return 0;
6         // Initialize array of all ones in O(n)
7         // Uses smaller size to save space complexity
8         int row[min(m, n)], x = 1, y;
9         std::fill_n(row, min(m, n), 1);
10        // Iterate over grid in O(n*m)
11        for (; x < max(m, n); ++x)
12            for (y = 1; y < min(m, n); ++y)
13                // Only need to do row because horizontal access
14                row[y] += row[y-1];
15        // Return last index of array
16        return row[min(m, n)-1];
17    }
18 }
```

Figure 1: LeetCode Submission Proof