**Chapter 7 Activity Exercise 16**

**2.**

1. NumPaths(1,1,4) .

* Check if row=4row=4. It's not, so we move.
* Check if col=4col=4. It's not, so we move to condition else.
* Recursively call NumPaths(2,1,4) and NumPaths(1,2,4).

1. NumPaths(2,1,4):

* Check if row=4row=4. It's not, so we move.
* Check if col=4col=4. It's not, so we move to condition else.
* Recursively call NumPaths(3,1,4) and NumPaths(2,2,4).

1. NumPaths(3,1,4)

* Check if row=4row=4. It's not, so we move.
* Check if col=4col=4. It's not, so we move to condition else.
* Recursively call NumPaths(4,1,4) and NumPaths(3,2,4)

1. NumPaths(4,1,4):

* Check if row=4row=4. Yes, it's equal, we return 11.
* The reason this algorithm is inefficient is that it performs calculations over and over. It calculates the number of paths repeated plenty of times for the same positions. What causes an exponential increase in the number of function calls and makes the algorithm slow.

1. In terms of efficiency the dynamic programming version of NumPaths is more time-efficient. This is due to memoization, preventing redundant calculations. But on the other side, the original recursive version has more time complexity. In terms of space efficiency, they both have the same versions of space complexity, but the dynamic programming version uses additional space.