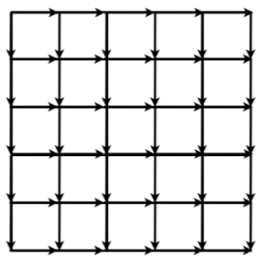
1. Complete the backtracking algorithm discussion on the 3-coloring problem from lecture-21, where we obtain a backtracking algorithm with running time $O(2^n)$ which is much better than the exhaustive search algorithm that considers 3^n possibilities.

2. There is an $n \times n$ grid of one-way street network. At any intersection, you may either travel from west to east or north to south. You want to compute the number of different ways in which you can travel from the north-west corner to the south-east corner. We will develop a backtracking algorithm for this problem.
(*Below is an example of a* 6×6 *(i.e.,* $n = 6$*). We are interested in finding the number of different ways of going from the top-left corner to the bottom-right corner.*)



Consider the following backtracking algorithm for this problem.

```
BacktrackGrid($n, m$)
        - if ($n = 1$ or $m = 1$)return(1)
        - else return(BacktrackGrid($n-1, m$) + BacktrackGrid($n, m-1$))
NumPaths($n$)
        - BacktrackGrid($n, n$)
```

Answer the following questions:

(a) Prove the correctness of the above algorithm.

(b) How many times is the function `BacktrackGrid` called when `NumPaths(`n`)` is executed?

(c) Can you think of a way to reduce the running time of the above program by making a small addition to it? How many times is the function `BacktrackGrid` called in your new algorithm?

3. You are given n types of coin denominations of values $v[1] < v[2] < ... < v[n]$ (all integers). Assume $v[1] = 1$, so you can always make change for any integer amount of money C. You want to make change for C amount of money with as few coins as possible.

Answer the following questions:

(a) Show that the following greedy algorithm for this problem does not work.

```
GreedyCoinSelect($n, v[1...n], C$)
  - For $i = n$ to 1
     - $m \leftarrow \lfloor \frac{V}{v[i]} \rfloor$
```

```
        - $g[i] \leftarrow m$
        - $C \leftarrow C - m \cdot v[i]$
    - return$(g[1], g[2], ..., g[n])$
```

(b) Consider the following backtracking algorithm for this problem. Obtain a lower bound on the running time of your algorithm.

```
BacktrackCoin($c$)
    - if $(c = 0)$return$(0)$
    - $m \leftarrow c$
    - for $i = 1$ to $n$:
        - if $(v[i] \leq c) m \leftarrow \min(m, 1 + \texttt{BacktrackCoin}(\texttt{c} - \texttt{v}[\texttt{i}]))$
    - return$(m)$
MinCoins($n, v[1...n], C$)
        - BacktrackCoin($C$)
```

(c) Can you improve the running time of your algorithm by making a small addition to your backtracking algorithm? What is the running time of the new algorithm?