CSC510 Homework 3

Chris Camano: ccamano@sfsu.edu

October 21, 2022

Question 1

1. Rocky Balboa is running up a staircase with $n \geq 1$ steps and can jump up 1,2 , or 3 steps at a time.

For example, if the staircase had $n = 5$ steps, two possible ways (but not the only two ways) of running up the stairs could be five 1-step jumps, or a 3-step jump followed by a 2-step jump.

Describe a backtracking algorithm for counting the number ways Rocky can run up n stairs.

Algorithm 1 ROCKY(n)

```
if n=1 then
    return 1
end if
if n=2 then
    return 2
end if
if n=3 then
    return 4
else
    return ROCKY(n-1) + ROCKY(n-2) + ROCKY(n-3)
```

Give a verbal justification for why your recurrence works. (Hint: just because the function has to work for $n \geq 1$ doesn't mean it can't work for other values, too)

This recurrence works by first considering the smallest possible base cases. After incorperating these into our backtracking algorithm all sub problems can be recursivley reconsidered through the recursive call after the base cases at the end of the algorithm.

Question 2

2. Now, write down a dynamic programming algorithm for computing the same function as in the previous question.

Algorithm 2 ROCKYDP(n)

```
Initialize S[1,...,n]
S[1]=1
S[2]=2
S[3]=4
for i=4,..,n do
    S[i]=S[i-1]+S[i-2]+S[i-3]
end for
    return S[n] =0
```

NOTE: For some reason my algorithm environment is appending a $= 0$ to the end of my return statement , it is not supposed to be there please ignore this

What's the runtime of your algorithm? The runtime of this algorithm is

$$\mathcal{O}(n-3) = \mathcal{O}(n)$$

since we are iterating over the given value n a single time.

Question 3

3. Suppose you're at the supermarket, looking to buy exactly n cupcakes for an upcoming party, where $n \geq 1$. At the supermarket, the only cupcakes you can buy come in boxes of 4,6 , and 9 .

For example, if $n = 14$, you could buy two boxes of 4 , and one box of 6 cupcakes to get exactly your desired total. However, if $n = 11$, you would have to buy extra cupcakes.

Describe a backtracking algorithm for determining whether you can buy exactly n cupcakes.

Algorithm 3 CUPCAKE (n)

```
1: if n=0 then
       return True
2: end if
3: if n < 0 then
       return False
4: else
       return  CUPCAKE(n-9) or CUPCAKE(n-6) or CUPCAKE(n-4)
```

Give a verbal justification for why your recurrence works.

For this algorithm we really need to just implement modular behavior over each sub problem. The base cases in a boolean context are if we can get the remainder down to zero which means that we can form some linear combination of boxes equating to the desired sum or if we reach below zero indicating that we can not evenly divide the sum into multiples of the given box sizes.

Question 4

4. The SFSU Robotics Club is holding a competition for designing robots that can navigate a maze. Suppose that the maze is laid out on an $m \times n$ grid with m rows and n columns, and you're given a boolean array $W[1,\ldots,m][1,\ldots,n]$, where $W[i][j] =$ TRUE if there is a wall at coordinate (i, j), and FALSE otherwise. The robots can only move down or right one cell at a time and cannot run into any walls. An example path in a grid where $m = 6, n = 7$ is the following:

Describe a backtracking algorithm for finding the number of ways for the robot to navigate from the top-left cell (coordinate $(1,1)$) to some other cell (coordinate (i, j), where $1 \leq i \leq m$ and $1 \leq j \leq n$) of the maze.

Algorithm 4 MAZEROBOT $(W[1,\ldots,m][1,\ldots,n], i, j)$

```
1: if W[1][1]=TRUE then
2:     return 0
3: end if
4: if W[i][j]=TRUE  then
5:     return 0
6: end if
7: if i < 1 or j < 1 then
8:     return 0
9: end if
10: if i=1 and j=1 then
11:     return 1
12: else
        return MAZEROBOT(W,i-1,j)+MAZEROBOT(W,i,j-1)
```

Give a verbal justification for why your recurrence works. (Hint: you can think of row or column 0 as a bunch of walls)

The algorithm above works in the following way: We first consider the base cases for the problem description the first of which is the event in which the starting point is in fact an obstacle. In this event there are 0 ways to move anywhere else. The second base case is the event where the goal in question is in fact an obstacle, in this event there are also 0 ways to reach the goal. Finally the last base case is the event where wer are considering the number of ways to move from the starting point to the starting point which is always 1

We now proceed to traversing the grid. Since the robot has limited movement and can only move down or right one cell at a time this means that for any given row the robot must be moving from the left to get there and for any given column the robot must be moving from above to get there. Since we can only arrive to a specific location on the grid from the left or from above for any given point(i,j) the number of ways to arrive to that location is the sum of the tile above and the tile to the left of it.

This is our recurrence relation