Problem 3(a)

Give dynamic Programming algorithm expressed recursively without memoization to determine the path the robot should follow to maximize the total value of coins collected as robot wanders on the board from cell (1,1) to cell (n,m). Analyze time required and gine corresponding pseudocode.

Solution.→ Let $F(i,j)$, be the largest value of coins the robot can collect and bring to cell (i,j)

→ This cell can be reached either from adjacent cell above it i.e. ~~Eei~~ cell $(i-1, j)$ or adjacent cell left of it i.e. cell $(i, j-1)$.

→ The largest value of coins that can be brought to this cell are either

$F(i-1, j)$

or

$F(i, j-1)$.

→ If there are no adjacent cells to left of first column or above first row, than we assume there are no neighbours.

→ The largest values of coins the robot can bring to cell (i, j) is the maximum of two cell values cell (i-1, j) or cell (i, j-1) i.e. maximum of F(i-1, j) or F(i, j-1)

→ The above maximum will be added with the coin values in cell (i, j).

→ Hence the maximum value of coins in cell (i, j) is

$$\max [F(i, j-1), F(i-1, j)] + c[i, j].$$

→ Recurrence Algorithm.

$$\begin{cases} F(i,j) = \max [F(i-1,j), F(i,j-1)] + c[i,j] & \text{for } 1 \le i \le n, \ 1 \le j \le m. \\ F(0,j) = 0 & \text{for } 1 \le j \le m \\ F(i,0) = 0 & \text{for } 1 \le i \le n. \end{cases}$$

→ c[i, j] is the value of coins in cell (i, j)

Pseudocode.

```
Max-Coins (c, i, j)
    if i=0 and j>1
        return 0
    if j=0 and i>1
        return 0
    else
    return c[i,j] + max(Max-coins (c, i-1, j),
                        Max-coins (c, i, j-1)).
```

Time - complexity.

In order to compare the values adjacent of cell (left or above), we need to make two recursive calls everytime to the algorithm. Hence, the time complexity will be polynomial function with the maximum power of n or m

Time Complexity = $\Theta(2^{\max(n,m)})$

Problem 3(b)

Give the algorithm iteratively with memorization. Analyze the time required.

Solution We will take a two-dimensional array F(i,j). It will be filled along the path in such a way, that the robot can collect maximum number for coins.

The last cell of F(i,j) will have the maximum number of coins.

As per the problem, the robot can move only in two direction.

→ Right

→ Bottom.

Suppose if Robot is at cell (i,j), then the possible moves which Robot will take in order to reach cell (i,j) is either from left or previous column, or from above ~~column~~. row.

previous column → cell (i, j-1)

above row → cell (i-1, j)

→ This means the largest number of coins that can be brought to these cells are maximum of F(i-1, j) and F(i, j-1) plus one possible coin at cell (i,j) itself.

→ Pseudocode.

→ Input : C[1...n, 1.....m] , n , m

cells of matrix C contains value of coin.

→ Max-Coins (C, n, m)

```
F[1,1] ← C[1,1].
for j = 2 to m do
        F[1, j] ← F[1, j-1] + C[1, j]
    end for.
for i = 2 to n do
        F[i,1] ← F[i-1, j] + C[i,1]
        for j = 2 to m do
                if F[i-1, j] > F[i, j-1] then
                        F[i, j] ← F[i-1, j] + C[i, j].
                else
                        F[i, j] ← F[i, j-1] + c[i, j].
                end if
        end for
    end for
return F[n, m].
```

→ Output : Maximum total value of coins collected

Time complexity:

→ if $F(i-1, j) > F(i, j-1)$, than the cell (i, j) will be reached from cell above cell (i, j) i.e $F(i-1, j)$

→ If $F(i, j-1) > F(i-1, j)$, than the cell (i, j) will be reached from cell left of cell (i, j) i.e. $F(i, j-1)$

→ If $F(i-1, j) = F(i, j-1)$, than the cell (i, j) will be reached from either direction i.e from above or left of cell (i, j)

→ If only one choice, either $F(i-1, j)$ or ~~e.g.~~ $F(i, j-1)$, use the available choice.

→ Hence, the time complexity will be product of number of cells ~~from~~ in the board i.e. $\Theta(n \times m)$.

∴ It always takes a constant time $\Theta(n \times m)$.