

CSCE 411: Mastery Quiz 5

Name: _____

UIN: _____

You are attempting to get from the northwest corner to the southeast corner of a city arranged in a rectangular grid. The citizens of this city are very absent-minded, and have a tendency to drop coins on the ground. You are in a hurry, so you are only willing to move south and east, but whenever you reach an intersection, you will pick up any coins on the ground.

Give an efficient algorithm that takes as input an array $A[1, \dots, m][1, \dots, n]$ of positive integers, where $A[i, j]$ stores the total value (in cents) of the coins lying at the intersection $i - 1$ blocks south and $j - 1$ blocks east of your starting location (your destination is $m - 1$ blocks south and $n - 1$ blocks east of your starting location, and you will always pick up the money there when you reach it). Your algorithm should return the maximum amount of money (in cents) that you can pick up on your walk. Briefly analyze the correctness and runtime of your algorithm.

For example, if $A = \begin{bmatrix} 1 & 2 & 12 & 9 \\ 5 & 4 & 8 & 3 \\ 6 & 10 & 11 & 7 \end{bmatrix}$, then your algorithm should return 41, corresponding to the path $1 \rightarrow 2 \rightarrow 12 \rightarrow 8 \rightarrow 11 \rightarrow 7$.

Solution: Let $Coins(i, j)$ be the maximum number of amount of money that you could earn if you started $i - 1$ blocks south and $j - 1$ blocks east of the city's northwest corner. We note that $Coins(i, j)$ satisfies the following recurrence:

$$Coins(i, j) = A[i, j] + \begin{cases} 0 & \text{if } i = m \wedge j = n \\ Coins(i + 1, j) & \text{if } i < m \wedge j = n \\ Coins(i, j + 1) & \text{if } i = m \wedge j < n \\ \max(Coins(i + 1, j), Coins(i, j + 1)) & \text{otherwise} \end{cases}$$

We note that the first case is when we start at our destination, the second case is when we start on the east edge of the city and can only move south, the third case is when we start on the south edge of the city and can only move east, and in the last case we have a choice between moving south and moving east, and choose whichever option yields the most money. The answer to our original problem is $Coins(1, 1)$

We can memoize $Coins$ using a 2-D array. We fill in the array in order of decreasing i and for each i will fill in the row in order of decreasing j (i.e. bottom to top, right to left).

We have $O(nm)$ subproblems, and we can solve each one in constant time (since $Coins(i, j)$ depends on at most two other subproblems). Thus, our algorithm runs in $O(mn)$ time (and $O(mn)$ space).