CSCI-GA.1170-001/002 Fundamental Algorithms November 10, 2015

Solutions to Problem 2 of Homework 7 (8 points)

Name: GOWTHAM GOLI (N17656180) *Due: Tuesday, November 10*

Assume that you have an n by n checkerboard. You must move a checker from the bottom left corner (position $(1,1)$) square the board to the top right corner (position (n,n)) square. In each step you may either

- move the checker up one square, or
- move the checker diagonally one square up and to the right, or
- move the checker right one square.

If you move a checker from square $x = (i,j)$ to square $y = (i',j')$ you get $p(x,y)$ dollars. You are told all of the $p(x,y)$ a priori. The $p(x,y)$ may be negative, zero or positive. You want to get as much money as possible.

(a) (4 pts) Let $M[i,j]$ be the highest profit you can collect from position $(1,1)$ to (i,j). Write a dynamic programming recurrence relation for $M[i,j]$ (do not forget the initial condition). Based on this recurrence relation, analyze the running time of the dynamic programming algorithm to compute $M[n,n]$.

Solution:

The best way to get to any square (i,j) is the best way to get from the square to it's left or the from the square to it's bottom or the from the diagonal square. Therefore,

$$\begin{aligned} M[i,j] = max\{&M[i,j-1] + p((i,j-1),(i,j)),\\ &M[i-1,j-1] + p((i-1,j-1),(i,j)),\\ &M[i-1,j] + p((i-1,j),(i,j))\} \end{aligned}$$

Note that the base case is $M[1,1] = 0$ and in the above equation, there will be no diagonal and bottom square for all the squares in the first row and there will be no diagonal and left squares for all the squares in the first column.

To fill an entry in the matrix M we look the adjacent three entries and appropriately fill the corresponding entry. Therefore, the total time taken to build this matrix is $\leq 3n^2 = O(n^2)$

□

(a) (4 pts) Give the "bottom-up" pseudocode for an efficient procedure CHECKERBOARD(n) for computing $M[n, n]$.

Solution:

```
Algorithm: CheckerBoard($n$)
$M \leftarrow$ NewArray($n \times n$)
$M[1, 1] \leftarrow 0$
for $i \leftarrow 2$ to $n$ do
    $M[i, 1] \leftarrow M[i-1, 1] + p((i-1, 1), (i, 1))$
end
for $j \leftarrow 2$ to $n$ do
    $M[1, j] \leftarrow M[1, j-1] + p((1, j-1), (1, j))$
end
for $i \leftarrow 2$ to $n$ do
    for $j \leftarrow 2$ to $n$ do
        $max \leftarrow M[i, j-1] + p((i, j-1), (i, j))$
        if $M[i-1, j-1] + p((i-1, j-1), (i, j)) > max$ then
            $max \leftarrow M[i-1, j-1] + p((i-1, j-1), (i, j))$
        else if $M[i-1, j] + p((i-1, j), (i, j)) > max$ then
            $max \leftarrow M[i-1, j] + p((i-1, j), (i, j))$
        $max \leftarrow M[i, j]$
    end
end
Return $M[n, n]$
```

Algorithm 4: Dynamic Programming Algorithm to calculate maximum profit in checker-board in $O(n^2)$ time

From the above psuedocode, it is clear that the dynamic programming algorithm runs in $O(n^2)$ time □