CSC 323 Algorithm Design and Analysis
Instructor: Dr. Natarajan Meghanathan
Module 4 – Dynamic Programming, Question Bank

1) There are a row of 6 coins of values {5, 1, 2, 10, 6, 2}; the objective is to pick up the maximum amount of money subject to the constraint that no two coins adjacent in the above list can be picked up. Develop a dynamic programming solution for this optimization problem – You need to write the recurrence relation and provide step-by-step details of the solution.

$F(n) = Max\{ c_n + F(n-2), F(n-1)\}$ for $n > 1$
$F(0) = 0$; $F(1) = c_1$.

index	0	1	2	3	4	5	6
C		5	1	2	10	6	2
F	0	5	5	7	15	15	**17**
History		**C1**	-	C3	**C4**	-	**C6**
		-	**F[1]**	F[1]	**F[2]**	F[4]	**F[4]**

F[0]	0
F[1]	C1
F[2]	Max (F[1], F[0]+C2) = Max(5, 0+1) = 5 = F[1]
F[3]	Max (F[2], F[1]+C3) = Max(5, 5+2) = 7 = F[1]+C3
F[4]	Max (F[3], F[2]+C4) = Max(7, 5+10) = 15 = F[2]+C4
F[5]	Max (F[4], F[3]+C5) = Max(15, 7+6) = 15 = F[4]
F[6]	Max (F[5], F[4]+C6) = Max(15, 15+2) = 17 = F[4]+C6

The coins that need to be picked up to obtain the maximum sum of **17** are {C1 = 5, C4 = 10, C6 = 2}

2) There are a row of 8 coins of values {3, 1, 5, 2, 5, 4, 2, 3}; the objective is to pick up the maximum amount of money subject to the constraint that no two coins adjacent in the above list can be picked up. Develop a dynamic programming solution for this optimization problem – You need to write the recurrence relation and provide step-by-step details of the solution.

index	0	1	2	3	4	5	6	7	8
C		3	1	5	2	5	4	2	3
F	0	3	3	8	8	13	13	15	**16**
History		**C1**	-	**C3**	-	**C5**	-	C7	**C8**
			F[1]	**F[1]**	F[3]	**F[3]**	**F[5]**	F[5]	**F[6]**

F[0]	0
F[1]	C1
F[2]	Max (F[1], F[0]+C2) = Max(3, 0+1) = 3 = F[1]
F[3]	Max (F[2], F[1]+C3) = Max(3, 3+5) = 8 = F[1]+C3
F[4]	Max (F[3], F[2]+C4) = Max(8, 3+2) = 8 = F[3]
F[5]	Max (F[4], F[3]+C5) = Max(8, 8+5) = 13 = F[3]+C5
F[6]	Max (F[5], F[4]+C6) = Max(13, 8+4) = 13 = F[5]
F[7]	Max (F[6], F[5]+C7) = Max(13, 13+2) = 15 = F[5]+C7
F[8]	Max (F[7], F[6]+C8) = Max(15, 13+3) = 16 = F[6]+C8

The coins that need to be picked up to obtain the maximum sum of **16** are {C1 = 3, C3 = 5, C5 = 5, and C3 = 3}

3) Compute the binomial coefficient C(10, 7) using dynamic programming. Write the recurrence relation.

Recurrence: $C(n,k) = C(n\text{-}1,k) + C(n\text{-}1,k\text{-}1)$ **for** $n > k > 0$

$C(n,0) = 1, \quad C(n,n) = 1$ **for** $n \geq 0$

n \ k	0	1	2	3	4	5	6	7
0	1							
1	1	1						
2	1	2	1					
3	1	3	3	1				
4	1	4	6	4	1			
5	1	5	10	10	5	1		
6	1	6	15	20	15	6	1	
7	1	7	21	35	35	21	7	1
8	1	8	28	56	70	56	28	8
9	1	9	36	84	126	126	84	36
10	1	10	45	120	210	252	210	**120**

4) Write the recurrence relations (including the initial conditions) and the time/space complexity for the following dynamic programming algorithms/problems discussed in class:

(a) Binomial Coefficients Problem:

$C(n,k) = C(n\text{-}1,k) + C(n\text{-}1,k\text{-}1)$ for $n > k > 0$

$C(n,0) = 1, \quad C(n,n) = 1$ for $n \geq 0$

Both Time and Space Complexity are: $\Theta(nk)$.

(b) Coin Row Problem:

Maximal value of coins selected from the list {c1, c2, ..., cn} is

$F(n) = \text{Max}\{ c_n + F(n\text{-}2), F(n\text{-}1)\}$ for $n > 1$

$F(0) = 0$; $F(1) = c_1$.

Both Time and Space Complexity are: $\Theta(n)$.

(c) Coin Selection Problem

Let F(i, j) be the largest number of coins the robot can collect and bring to the cell (i, j) in the ith row and jth column of the board.

$$F(i, j) = \max\{F(i-1, j), F(i, j-1)\} + c_{ij} \quad \text{for } 1 \leq i \leq n, \quad 1 \leq j \leq m$$

$$F(0, j) = 0 \text{ for } 1 \leq j \leq m \quad \text{and} \quad F(i, 0) = 0 \text{ for } 1 \leq i \leq n,$$

where $c_{ij} = 1$ if there is a coin in cell (i, j) and $c_{ij} = 0$ otherwise

Both Time and Space Complexity are: $\Theta(nm)$.

(d) Longest Common Subsequence (LCS) Problem

Let LCS(i, j) be the length of the longest common subsequence between two sequences X[1...i] and Y[1...j].

LCS(i, j) = LCS(i-1, j-1) + 1, if X[i] = Y[j]

Max{ LCS(i, j-1), LCS(i-1, j) } if X[i] != Y[j]

where LCS(i, 0) = 0 for all i = 1 to m (the original length of sequence X) and
LCS(0, j) = 0 for all j = 1 to n (the original length of sequence Y).
Both the time and space complexity is: Θ(nm)

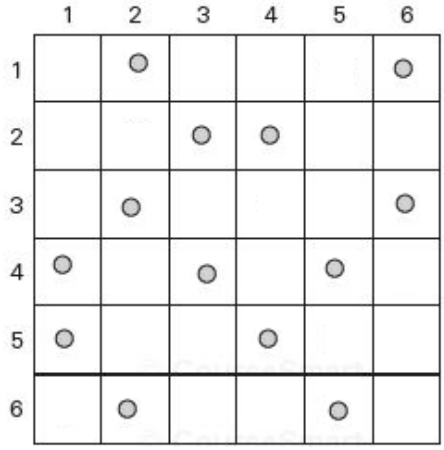
(e) Integer Knapsack Problem
Let F(i, j) be the value of the most valuable subset of the first i items ($1 \le i \le n$) that fit into the knapsack of capacity j ($1 \le j \le W$).

$$F(i, j) = \begin{cases} \max\{F(i-1, j), v_i + F(i-1, j-w_i)\} & \text{if } j - w_i \ge 0, \\ F(i-1, j) & \text{if } j - w_i < 0. \end{cases}$$

Initial Condition: F(0, j) = 0 for $1 \le j \le W$ F(i, 0) = 0 for $1 \le i \le n$
Both Time and Space Complexity are: Θ(nW), where n is the number of items in the list and W is the integer capacity of the knapsack.

5) Several coins are placed in cells of a 6 x 6 board (n x m board) shown below, with no more than one coin per cell. A robot, located in the upper left cell of the board, needs to collect as many of the coins as possible and bring them to the bottom right cell. On each step, the robot can move either one cell to the right or one cell down from its current location. When the robot visits a cell with a coin, it always picks up that coin.

Design a Dynamic Programming algorithm to find the maximum number of coins the robot can collect and a path it needs to follow to do this. You need to write the recurrence relation and clearly define all the terms/variables involved in it. Briefly explain how would you trace back your computation and determine the optimal path for the robot. Execute the above on the board.



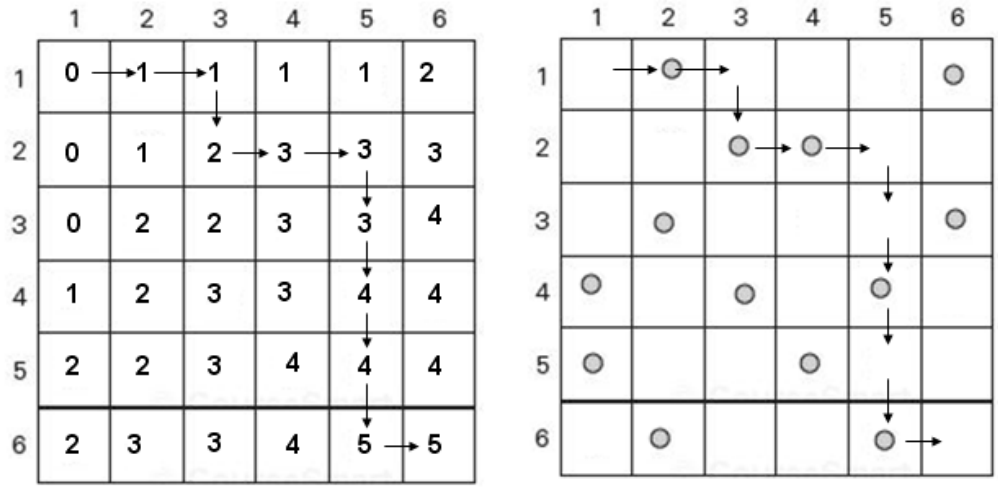
Solution:

Recurrence

$$F(i, j) = \max\{F(i-1, j), F(i, j-1)\} + c_{ij} \quad \text{for } 1 \le i \le n, \ 1 \le j \le m$$

$$F(0, j) = 0 \text{ for } 1 \le j \le m \quad \text{and} \quad F(i, 0) = 0 \text{ for } 1 \le i \le n,$$

where c_{ij} = 1 if there is a coin in cell (i, j) and c_{ij} = 0 otherwise.



6) Find the longest common subsequence among the pairs of sequences given below:

(a) **X = ATGACTATAA** and **Y = GACTAATA**

		G	A	C	T	A	A	T	A
	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
T	0	0	0	1	2	2	2	2	2
G	0	1	1	1	2	2	2	2	2
A	0	1	2	2	2	3	3	3	3
C	0	1	2	3	3	3	3	3	3
T	0	1	2	3	4	4	4	4	4
A	0	1	2	3	4	5	5	5	5
T	0	1	2	3	4	5	5	6	6
A	0	1	2	3	4	5	6	6	7
A	0	1	2	3	4	5	6	6	7

		G	A	C	T	A	A	T	A
	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
T	0	0	0	1	2	2	2	2	2
G	0	1	1	1	2	2	2	2	2
A	0	1	2	2	2	3	3	3	3
C	0	1	2	3	3	3	3	3	3
T	0	1	2	3	4	4	4	4	4
A	0	1	2	3	4	5	5	5	5
T	0	1	2	3	4	5	5	6	6
A	0	1	2	3	4	5	6	6	7
A	0	1	2	3	4	5	6	6	7

A T G A C T - A T A A
- - G A C T A A T - A

Longest Common Sub sequence
G A C T A T A

(b) **X = TGACTAC and Y = ACTGATGC**

		T	G	A	C	T	A	C
	0	0	0	0	0	0	0	0
A	0	0	0	1	1	1	1	1
C	0	0	0	1	2	2	2	2
T	0	1	1	1	2	3	3	3
G	0	1	2	2	2	3	3	3
A	0	1	2	3	3	3	4	4
T	0	1	2	3	3	4	4	4
G	0	1	2	3	3	4	4	4
C	0	1	2	3	4	4	4	5

		T	G	A	C	T	A	C
	0	0	0	0	0	0	0	0
A	0	0	0	1	1	1	1	1
C	0	0	0	1	2	2	2	2
T	0	1	1	1	2	3	3	3
G	0	1	2	2	2	3	3	3
A	0	1	2	3	3	3	4	4
T	0	1	2	3	3	4	4	4
G	0	1	2	3	3	4	4	4
C	0	1	2	3	4	4	4	5

T G A C T G A T G C
- - A C T - A - - C

Longest Common Sub sequence
ACTAC

(c) **X = CAAGTACG and Y = ACTGGAGCAT**

		C	A	A	G	T	A	C	G
	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
C	0	1	1	1	1	1	1	2	2
T	0	1	1	1	1	2	2	2	2
G	0	1	1	1	2	2	2	2	3
G	0	1	1	1	2	2	2	2	3
A	0	1	2	2	2	2	3	3	3
G	0	1	2	2	3	3	3	3	4
C	0	1	2	2	3	3	3	4	4
A	0	1	2	3	3	3	4	4	4
T	0	1	2	3	3	4	4	4	4

		C	A	A	G	T	A	C	G
	0	0	0	0	0	0	0	0	0
A	0	0	1	1	1	1	1	1	1
C	0	1	1	1	1	1	1	2	2
T	0	1	1	1	1	2	2	2	2
G	0	1	1	1	2	2	2	2	3
G	0	1	1	1	2	2	2	2	3
A	0	1	2	2	2	2	3	3	3
G	0	1	2	2	3	3	3	3	4
C	0	1	2	2	3	3	3	4	4
A	0	1	2	3	3	3	4	4	4
T	0	1	2	3	3	4	4	4	4

Longest Common Sub sequence
ATAG

```
- - A - C T G G A - G C A T
C A A G - T - - A C G - - -
```