National University of Computer and Emerging Sciences, Lahore Campus

Course:	Design & Analysis of Algorithms	Course Code:	CS-2009
Program:	BS (Computer/Data Science)	Semester:	Fall 2023
Duration:	60 Minutes	Total Marks:	18
Paper Date:	10-Nov-23	Section:	ALL
Exam:	Midterm 2	Page(s):	6
Name		Roll Number	

Instruction/Notes: Solve it on question paper

Question	1	2	3	4
Marks	/5	/5	/5	/3

Q1) Consider the problem of rod cutting discussed in class. Dry run dynamic programming algorithm on following input for rod length 7. Show all computations and write values in the given array C[n] for memoization. [5 Marks]

Length (i)	1	2	3	4	5	6	7
Price P[i]	2	5	7	12	13	16	17

C[0] = 0

$$C[n] = \max_{1 \le i \le n}(P[i] + C[n-i])$$

Solution:

n	1	2	3	4	5	6	7
C[n]							

Q2) Consider the following recursive algorithm. [1+4 = 5 Marks]

```
/* m and n are lengths of char arrays X and Y respectively */
int Function( char *X, char *Y, int m, int n )
{
   if (m == 0 || n == 0)
      return 0;
   if (X[m-1] == Y[n-1])
      return 1 + Function (X, Y, m-1, n-1);
   else
      return max (Function (X, Y, m, n-1) +1,  Function (X, Y, m-1, n) +1);
}
```

(a) What is time complexity of above algorithm? Show all working

(b) Convert the recursive code given above into bottom up iterative dynamic programming algorithm. Write its time complexity.

Q3) You are given n integers in the range [1, m] and a target T. Find a subset of the n integers whose sum is as close to T as possible without exceeding it. For example,
T= 10,
Set of n integers = {3, 5, 4}
All possible subsets = {3}, {5}, {4}, {3, 5}, {3, 4}, {5, 4}, {3, 5, 4}
Optimal Solution = subset = {5, 4} // The sum is 9, any other subset sum will either exceed 10 or will be less than 9.

Consider following greedy approach for solving this problem. Sort number in descending order. Start adding numbers in subset starting from largest number. Terminate at the number for which the sum exceeds T. Prove this greedy strategy is not optimal by giving counter example. [5 Marks]

Solution:

T =

Set =

Subset given by the Greedy algorithm =

Subset given by the optimal solution =

Q4) This question is about maximum subarray sum problem. We are given an integer array A[0..n) and we have to find the subarray A[i..j) such that the sum of the integers A[i] + A[i + 1] + · · · + A[j − 1] is maximal. If M[j] holds the maximum sum for any segment A[i..j) ending in j − 1 how can we efficiently compute the maximum sum for any segment A[i…..j + 1) ending in j? In other words, write the expression (recursive equation) for defining the value of M[j+1].
[Hint: you do not need a loop or pseudocode.] [3 Marks]

National University of Computer and Emerging Sciences

Lahore Campus

Design and Analysis of Algorithms (CS2009)

Date: April 6, 2024

Sessional-II Exam

Total Time (Hrs): 1
Total Marks: 20
Total Questions: 3

Course Instructor(s)

Dr. Saira Karim
Dr. Aasim Qureshi
Ms. Abeeda Akram
Mr. Usama Hassan Alvi
Mr. Sajid Ali Kazmi
Mr. Iteza Muzaffar

20L-0933
Roll No

4G
Section

Student Signature

Do not write below this line

Attempt all the questions.

CLO #1: Design algorithms using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms and apply them to solve problems in the domain of the program

Q1: Given an array A of n numbers in the range of [-n, n], devise an algorithm that finds the number of distinct pairs (i, j) such that j>i and A[i] = A[j]. Your algorithm must work in O(n) time.

[marks 10]

Sample Input:
[1, 2, 1, 2, 3, 1, 4]
Sample Output: 4
Sample Input:
[-1, -1, -1, -1, 4]
Sample Output: 6

CLO #1: Design algorithms using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms and apply them to solve problems in the domain of the program

Q2: Catalan numbers form a fascinating sequence of natural numbers that occur in various counting problems and have many applications in combinatorial mathematics. The first few Catalan numbers for n = 0, 1, 2, 3, 4, 5, 6, 7 … are 1, 1, 2, 5, 14, 42, 132, 429, …
Following recurrence is used for calculation of the nth Catalan number.

$$C_0 = 1, C_1 = 1 \text{ and } C_n = \sum_{i=0}^{n-1} C_i C_{n-1-i} \quad for\ n > 2$$

0271

Consider the following recursive solution for calculation of the nth Catalan.

```
int catalan(n)
{
  if (n <= 1)
    return 1
  num = 0
  for i = 0 to n
    num = num + (catalan(i) * catalan(n - i - 1))
  return num
}
```

Write the pseudocode for the bottom-up DP solution of this problem. Also provide the time complexity of your solution. **[marks 5]**

CLO #1: Design algorithms using different algorithms design techniques i.e. Brute Force, Divide and Conquer, Dynamic Programming, Greedy Algorithms and apply them to solve problems in the domain of the program

Q3: Given N algorithm design problems in an exam with a total exam time of T minutes, each problem has a certain number of checkpoints and marks of a problem are equally distributed for each checkpoint. You are not sure whether you will be able to attempt all the questions or even all the checkpoints of a certain problem. The goal is to maximize the score within the given time constraint. You are given the information of total exam time (T), total number of questions (N), marks and expected time for each problem in arrays M[1...N] and E_Time[1...N] respectively. A student devised an algorithm to maximize the score by prioritizing the problems with maximum marks using a greedy approach. Do you think this greedy approach will always provide an optimal solution. If not, then provide a counter example where this approach will fail. In case of a counter example, you are supposed to follow the same pattern of sample input i.e., clearly mention all the information and a proper justification is required which shows that why this greedy approach will fail to provide an optimal solution. **[marks 5]**

Sample Input:

T = 60 minutes
N = 5
Marks: {30, 25, 20, 15, 10}
Expected Time: {25, 20, 15, 30, 20}
Max Achievable Score = 75

National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Design and Analysis of Algorithms	Course Code:	CS2009
	Degree Program:	BSCS, BSSE	Semester:	SPRING 2023
	Exam Date:	Tuesday, April 11, 2023	Total Marks:	11 + 19 = 30
	Section:	ALL	Page(s):	3
	Exam Type:	Mid-Term - II		

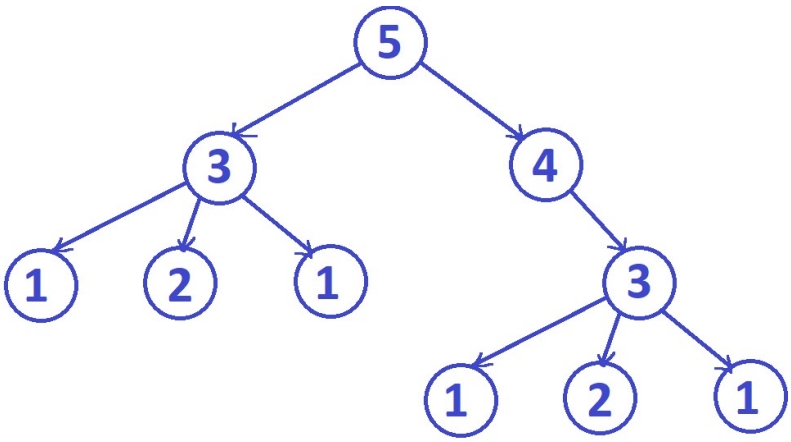
Student : Name:_____ **Roll No.**_____ **Section:**_____

Instruction/Notes: Attempt all questions. There are two questions, don't forget to check the back side as well.

Question 1: [3 + 8 = 11 Marks]

a) Draw **recursion tree** of **MyAlgo** using n = 5.
Only the recursion tree is required. No calculations are needed. **There will be no partial credit for this part.**

```
MyAlgo(n)
{
   IF(n == 2)
      return 3;
   ELSE IF (n == 1)
      return 1;
   ELSE IF (n%3 == 0)          // n is a multiple of 3
      return  MyAlgo(⌊n/2⌋) - MyAlgo(n-1) + 2 x MyAlgo(n-2);
   ELSE IF (n%2 == 0)          // n is even
      return MyAlgo(n-1);
   ELSE
      return MyAlgo( (n+1) / 2 ) - MyAlgo(n-1);
}
```



b) Convert the pseudocode given above (in part **a**) into bottom-up dynamic programming (iterative code). For this part only **pseudocode** is required.

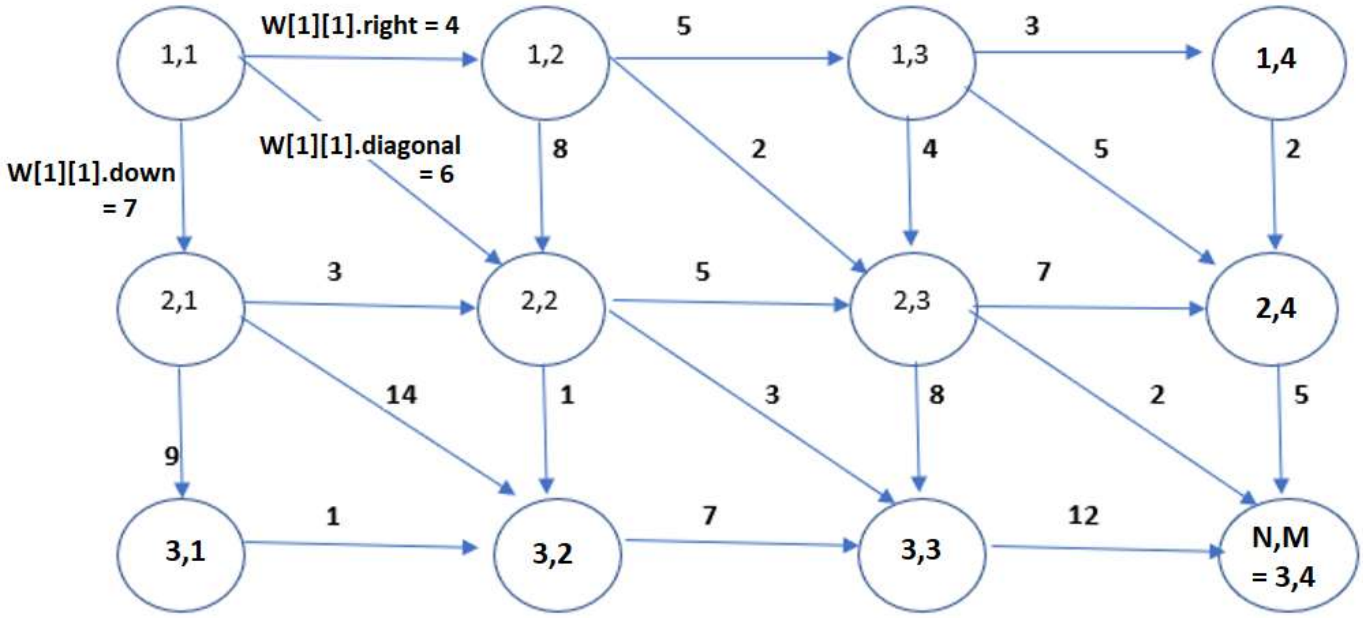
One marks for array of correct size
One marks for base cases
One marks for the loop (3 to n)
One marks for look-up from the array
One marks for storing each value in the array
One mark each for all the cases inside the loop
Total 8 marks

```
A[1..n]
A[2] = 3
A[1] = 1
FOR (i = 3 to n)
        IF (i%3 == 0)
                A[i] = A[ i/2 ] - A[i-1] + 2 x A[i-2]
        ELSE IF (i%2 == 0)
                A[i] = A[i-1]
        ELSE
                A[i] = A[ (i+1)/2] - A[i-1]
RETURN A[n]
```

Question 2:

Assume that there is a moving object of pacman in a weighted grid of (N rows and M columns). Pacman starts from the cell (1,1) and can move a maximum distance of 1 unit cell at a time either in rightwards, downwards or in the diagonal cell, as shown in the figure below. Note that the object can only move forward and there is no way back. At each move pacman can score some points. These points for each move is stored in the array W[1…N][1…M].

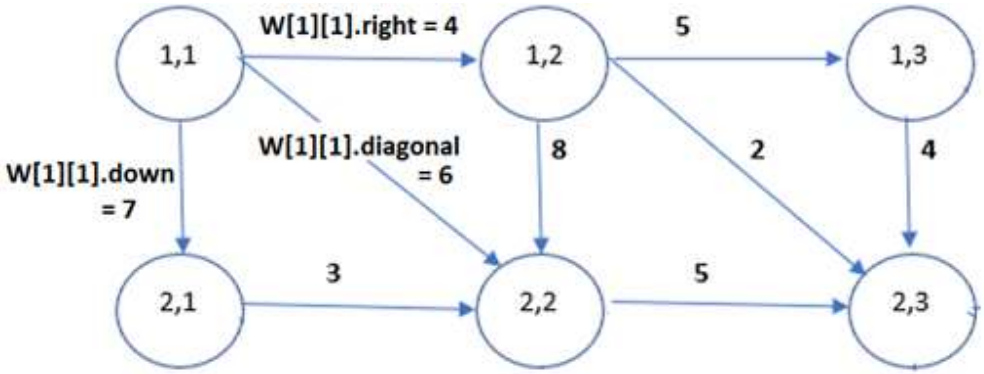
If pacman is at cell (1, 1), it can move towards rightwards to cell (1, 2) and earn 4 points which is stored in W[1][1].**right**, or it can move downwards to cell (2, 1) and earn 7 points which is stored in W[1][1].**down** or it can move diagonally to cell (2, 2) and earn 6 points which is stored in W[1][1].**diagonal**. Pacman wants to earn maximum points while going from cell (1,1) to cell (N,M).



a) COUNTER-EXAMPLE [CLO-3] [7 marks]

Consider the greedy algorithm where pacman selects the maximum value among right, down and diagonal from each cell and move in that direction until it reach the cell (N, M), for example at cell (1, 1), pacman will move down because W[1][1].**down** is greater than W[1][1].**right** and W[1][1].**diagonal**.

Prove that the greedy algorithm is not correct by providing a counter example. Only a counter example is required. **There will be no partial credit for this part**. The size of your counter-example should not be larger than 3 x 3.



b) Dynamic Programming [CLO-1] [12 marks]

Design and bottom-up dynamic programming algorithm (iterative algorithm) that calculates the maximum score pacman can achieve starting from the source cell (1,1) to destination cell (N,M). Weights of edges are given in a 2-D array W[1..N][1…M]. Each entry of this array holds three values i.e. W[i][j].right, W[i][j].down and W[i][j].diagonal. For this part only **pseudocode** is required.

One marks for array of correct size
Each marks each for filling the base cases (first row and first column)
Two marks for using nested loops to fill the remaining entries
One marks for the look-up
One marks for storing results in array
One marks for storing in correct position
One marks each for selecting the corresponding right, down and diagonal correctly
One marks for taking the maximum

```
A[1..N][1..M]
A[1][1] = 0
FOR (i = 2 to N)
    A[i][1] = A[i-1][1] + W[i-1][1].down
FOR (j = 2 to M)
    A[1][j] = A[1][j-1] + W[1][j-1].right
FOR(i = 2 to N)
    FOR (j = 2 to M)
        A[i][j] = max( A[i-1][j] + W[i-1][j].down,
                       A[i][j-1] + W[i][j-1].right,
                       A[i-1][j-1] + W[i-1][j-1].diagonal

Final answer is stored in A[N][M]
```

National University of Computer and Emerging Sciences, Lahore Campus

Course:	Design & Analysis of Algorithms	Course Code:	CS-2009
Program:	BS (Computer Science)	Semester:	Spring 2022
Duration:	60 Minutes	Total Marks:	17
Paper Date:	6-May-22	Section:	ALL
Exam:	Midterm 2	Page(s):	4
Name	*Tahir Ejaz*	~~Roll Number~~	*Solution*

Instruction/Notes: Weightage of the exam is Section Specific (i.e., for each section the weightage would be as per the announcement done in that regard).
Do NOT un-staple your exam, otherwise it might be cancelled.
Ample space is provided for rough work; NO EXTRA sheets will be provided.

Question	1	2	3	Total
Marks	NA/5	NA /5	NA /7	NA /17

Question 1: **[5] Marks**

Consider the problem of rod cutting discussed in class. Dry run dynamic programming algorithm on following input for rod length 7. Show all computations and write values in the given array C[i] $r[n]$ for memoization.

Length	1	2	3	4	5	6	7
Price	2	5	6	11	12	14	17

$$r_n = \max_{1 \le i \le n} (p_i + r_{n-i}).$$

Solution:

i n	1	2	3	4	5	6	7
C[i] $\boldsymbol{r[n]}$	**2**	**5**	**7**	**11**	**13**	**16**	**18**

$r[1] = \max_{1 \le i \le 1}(p_i + r[1-i]) = p_1 + 0 = \mathbf{2}$

$r[2] = \max_{1 \le i \le 2}(p_i + r[2-i]) = \max(p_1 + 2, p_2 + 0) = \max(2+2, 5+0) = \max(4,5) = \mathbf{5}$

$r[3] = \max_{1 \le i \le 3}(p_i + r[3-i]) = \max(p_1 + 5, p_2 + 2, p_3 + 0) = \max(7,7,6) = \mathbf{7}$

$r[4] = \max_{1 \le i \le 4}(p_i + r[4-i]) = \max(p_1 + 7, p_2 + 5, p_3 + 2, p_4 + 0) = \max(9,10,8,11) = \mathbf{11}$

$r[5] = \max_{1 \le i \le 5}(p_i + r[5-i]) = \max(p_1 + 11, p_2 + 7, p_3 + 5, p_4 + 2, p_5 + 0) = \max(13,12,11,13,12) = \mathbf{13}$

$r[6] = \max_{1 \le i \le 6}(p_i + r[6-i]) = \max(p_1 + 13, p_2 + 11, p_3 + 7, p_4 + 5, p_5 + 2, p_6 + 0) =$
$\max(15,16,13,16,14,14) = \mathbf{16}$

$r[7] = \max_{1 \le i \le 7}(p_i + r[7-i]) = \max(p_1 + 16, p_2 + 13, p_3 + 11, p_4 + 7, p_5 + 5, p_6 + 2, p_7 + 0) =$
$\max(18,18,17,18,17,16,17) = \mathbf{18}$

Question 2: **[5] Marks**

Consider the weighted interval scheduling problem. Here the input is a set of n jobs, each with a start time finish time, and reward. Our task is to schedule some of these jobs on a single machine. The output is a non-overlapping subset of the jobs, and the goal is to maximize the total reward from the jobs in the set. Consider following greedy strategy:

Sort the jobs by reward and schedule them one by one starting with the highest reward, rejecting any that overlap with jobs already scheduled.

Give counter example to prove that it does not guarantee optimal solution.

Solution:

Consider following three activities:

- Activity a, with start time 2, finish time 4 and reward 3.
- Activity b, with start time 5, finish time 8 and reward 3.
- Activity c, with start time 3, finish time 6 and reward 4.

Now if we select activity c on the basis of the given strategy (reward 4 being the highest), then we cannot select activities a and b, and the total reward would be $\boldsymbol{4}$.

However, in an optimal solution, both activity a and activity b would be selected making a total reward of $3 + 3 = \mathbf{6}$.

Question 3: **[7] Marks**

Consider the following recursive algorithm

```
P(n){
        IF (n = 1)
                RETURN 1
        ELSE
                SUM = 0
                FOR( i = 1 to n − 1 )
                        SUM = SUM + P(i) x P(n − i)
        RETURN SUM
}
```

Convert the recursive code given above into bottom up iterative dynamic programming algorithm.

Following is an example output of program for input i

i	1	2	3	4	5	6
P(i)	1	1	2	5	14	42

Solution:

```
let P[1...n] be a new array
P[1]=1
for j=2 to n
      sum=0
      for i=1 to j-1
            sum = sum + P[i]*P[j-i]
      A[j]=sum
return A[n]
```

Name: _____ Roll #: _____ Section: _____

National University of Computer and Emerging Sciences, Lahore Campus

	Course:	**Design and Analysis of Algorithms**	**Course Code:**	**CS302**
	Program:	**BS(Computer Science)**	**Semester:**	**Fall 2020**
	Duration:	**90 Minutes**	**Total Marks:**	**18**
	Paper Date:	**3-Feb-21**	**Weight**	**12.5%**
	Section:	**ALL**	**Page(s):**	**5**
	Exam:	**Midterm 2**		

Instruction/Notes: Attempt the examination on the question paper and write concise answers. You can use extra sheet for rough work. Do not attach extra sheets used for rough with the question paper. Don't fill the table titled Questions/Marks.

Question	1	2	3	Total
Marks	/ 5	/5	/8	/18

Q1) In a twist to the fractional Knapsack problem, we allow the repetition of i.e. we can use any number of copies of a certain element if we wish. What will be the new greedy algorithm for this modified fractional knapsack? Give a pseudo-code. [5 Marks]

Name: _____ Roll #: _____ Section: _____

Q2) Dry run the dynamic programming algorithm to find the Max Sub-array Sum of the following input array, A:

-10	50	60	-150	20	80	-10	-5	100	-5

In array S, fill into S[i] : the optimal sum of a subarray ending at A[i]. In array P, fill into P[i]: the starting index of the optimal array ending at A[i] [5 Marks]

Solution:

S[]

P[]

Q3) Given a directed graph G (V,E), we might wish to determine whether G contains a path from i to j for all vertex pairs (i, j) ∈ V. We define the **transitive closure** of G as the graph G*=(V,E*),

where E* = {(i, j) | there is a path from vertex i to vertex j in G}.

In other words you want to find out for each vertex u, which vertices are reachable from vertex u. Give an efficient algorithm that computes the transitive closure of a given graph G(V,E). [8 Marks]

a) Briefly explain your algorithm in English.

b) Write pseudo code of your algorithm

c) Analyze its worst case time complexity.

National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	**Design and Analysis of Algorithms**	**Course Code:**	**CS302**
	Degree Program:	**BSCS**	**Semester:**	**Fall 2021**
	Exam Duration:	**60 Minutes**	**Total Marks:**	**26**
	Paper Date:	**2 Dec 2021**	**Weight**	**13**
	Section:	**ALL**	**Page(s):**	**4**
	Exam Type:	**Midterm-II**		

Student : Name:_____ **Roll No.**_____ **Section:**_____

Instruction/Notes: Attempt the examination on the question paper and write concise answers. You can use extra sheet for rough work. Do not attach extra sheets used for rough with the question paper. Do not fill the table titled Questions/Marks.

Question	1	2	3	Total
Marks	/ 6	/10	/10	/26

Q1) [6 Marks]Dry run the dynamic programming algorithm to find the Max Sub-array Sum of the following input array, A:

-9	40	50	-160	25	90	-10	-8	90	-8

You have to create two arrays for the solution. In array S, fill into S[i] : the optimal sum of a subarray ending at A[i]. In array P, fill into P[i]: the starting index of the optimal array ending at A[i]

Solution:

S[]

-9	40	90	-70	25	115	105	97	187	179

P[]

1	2	2	2	5	5	5	5	5	5

Q2) [10 Marks] Given two arrays A and B of equal size n, you have to design an efficient algorithm that minimizes the sum A[1] x B[1] + A[2] x B[2] + … + A[n] x B[n]. You are allowed to shuffle the elements of each array, A and B.

Example:
N = 3
A = {3, 1, 1}, B = (6, 5, 4)
Minimum sum = 1 x 6 + 3 x 4 + 1 x 5 = 23
There are other possible ways of taking the sum like 3 x 6+ 1 x4+1 x5 = 27, but the minimum sum is 23.
Hint: Use greedy algorithm

Solution

Sort one array in increasing order and other array in decreasing order.

For (i = 1 to n)

MinSum += A[i]*B[i]

Q3) [10 Marks] Given a matrix M * N of integers where each cell has a cost associated with it, the cost can also be negative. Find the minimum cost to reach the last cell (M-1, N-1) of the matrix from its first cell (0,0). We can only move one unit right (Column No + 1), one unit down (Row No + 1), and one unit in bottom diagonal (Row No + 1, Column No + 1). For example from index (i , j) you can move to (i , j+1), (i+1 , j), and (i+1, j+1) where i = row no and j = column no.

Example

4	7	8	6	4
-6	7	3	9	2
3	8	1	-2	4
7	1	7	3	7
2	9	8	9	3

4	7	8	6	4
-6	**7**	3	9	2
3	8	**1**	**-2**	4
7	1	7	**3**	7
2	9	8	9	**3**

Path with minimum cost = 4 -> -6 -> 7 -> 1 -> -2-> 3 -> 3 = 10

Provide a Dynamic Programming solution for it.

a) Provide recurrence for sub-problem

Solution

C (i , j) = ***Min*** (**C**[i][j - 1] , **C**[i - 1][j] , C[i-1][j-1]) + **A**[i][j]

b) Provide pseudo code for DP solution

c) Provide time complexity of DP solution

National University of Computer and Emerging Sciences, Lahore Campus

Course:	Design and Analysis of Algorithms	Course Code:	CS 302
Program:	BS (Computer Science)	Semester:	Fall 2017
Duration:	60 Minutes	Total Marks:	40
Paper Date:	04-Nov-2017	Page(s):	2
Section:	N/A	Section:	
Exam:	Midterm Exam 1	Roll No:	

Instruction/Notes: Solve this exam on the answer sheet.

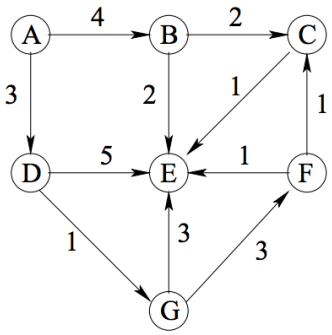
Question 1 (15 points)

A robot, initially standing at point 0, needs to travel from point n on a straight line. In order to do that it can take steps of either size 1, size 2 or size 3. For example, if it is at point k, then it can go next to point k+1, point k+2 or point k+3. This means that there are many different ways in which it can get from point 0 to point n. For example, if n=3, there are 4 ways to get from 0 to 3: either take three steps of 1, or take a step of 2 followed by step of 1, or take a step of 1 followed by a step of 2, or take a single step of 3.
We wish to write a dynamic programming algorithm to compute the total number of ways in which the robot can get from point 0 to point n.

(a) How many ways are there for the robot to get from 0 to 4?
(b) Let V[k] be defined as the number of ways for the robot to get from point 0 to point k. Write a recurrence for V[k] in terms of smaller sub-problems. Also give the base case(s).
(c) Make the array V from v[0] to v[8] and fill in the values using the recurrence in (b).

Question 2 (15 points)

Perform Dijkstra's shortest path algorithm on the following directed graph, taking A as source. List the vertices in the order in which they are deleted from the min-heap and their shortest path lengths from A and their parent node according to the algorithm. For example, the first of be deleted is [A, 0, -], i.e. vertex A with path length 0 and no parent. There is no need to show the contents of the entire heap or any other information.



P.T.O.

Question 3 (10 points)

Given an undirected graph $G = (V, E)$, and an edge $e \in E$ write an algorithm which determines whether G contains a cycle containing the edge e. The algorithm should take no more than $O(|V| + |E|)$ time. Describe your algorithm in English in a few lines, then give pseudo code. *There is no partial credit in this problem.*

THE END

National University of Computer and Emerging Sciences, Lahore Campus

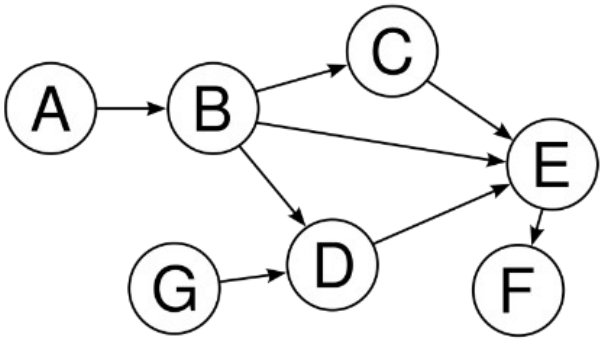
Course:	**Design and Analysis of Algorithms**	**Course Code:**	**CS302**
Program:	**BS(Computer Science)**	**Semester:**	**Spring 2018**
Duration:	**60 Minutes**	**Total Marks:**	**23**
Paper Date:	**13-April-18**	**Weight**	**15%**
Section:	**ALL**	**Page(s):**	**4**
Exam:	**Midterm 2 Solution**		

Instruction/Notes: Attempt the examination on the question paper and write concise answers. You can use extra sheet for rough work. Do not attach extra sheets used for rough with the question paper. Don't fill the table titled Questions/Marks.

Question	1-4	5	6	Total
Marks	/ 10	/ 8	/ 5	/ 23

Question#1

What would be the output (ordering of vertices) of topological sort on the following graph, starting with vertex A. [4 Marks]



Answer: G A B D C E F

Question#2

What would be the time complexity of Breadth First Search in the worst case, if graph is represented as Adjacency Matrix instead of Adjacency List? [2 Marks]

$O(n^2)$

Question#3

Given an adjacency list of a directed graph, how long will it take to compute the outdegree of each vertex? Also how long would it take to compute the indegree of each vertex? [2 Marks]

Answer: O (m + n)

Question#4

Six files F1, F2, F3, F4, F5 and F6 have 100, 200, 50, 80, 120, 150 records respectively. In what order should they be stored so as to optimize file access (Access time of file depends on number of records stored before this file). Assume each file is accessed with the same frequency. [2 Marks]

a) F3, F4, F1, F5, F6, F2
b) F2, F6, F5, F1, F4, F3
c) F1, F2, F3, F4, F5, F6

Answer: option a

Question#5

Below is the recurrence relation for edit distance of strings X_n and Y_m where $X_n = (x_1, x_2, ..., x_n)$ is a string of n characters and $Y_m = (y_1, y_2, .., y_m)$ is a sting of m character:

$$ED(X_n, Y_m) = \begin{cases} m \; if \; n = 0 \\ n \; if \; m = 0 \\ ED(X_n - 1, Y_m - 1) \; if \; x_n = y_m \\ Min\left(ED(X_{n-1}, Y_{m-1}), ED(X_{n-1}, Y_m), ED(X_n, Y_{m-1})\right) + 1 \; if \; x_n \neq y_m \end{cases}$$

Typists often make transposition errors exchanging neighboring characters, such as typing "setve" when you mean "steve." This requires two replace operations (edit distance=2) to convert setve to steve. If we incorporate a swap(swaps two adjacent characters) operation into our edit distance functions along with insert, delete and replace, then such neighboring transposition errors can be fixed at the cost of one operation.

a) What would be the recurrence relation for this modified edit distance (that includes swap operation)? [4 Marks]

$$ED(X_n, Y_m) = \begin{cases} m \; if \; n = 0 \\ n \; if \; m = 0 \\ ED(X_n - 1, Y_m - 1) \; if \; x_n = y_m \; ED(X_n - 2, Y_m - 2) + 1 \; if \; x_n \neq y_m \wedge x_{n-1} = y_m \wedge x_n = y_{m-1} \\ Min\left(ED(X_{n-1}, Y_{m-1}), ED(X_{n-1}, Y_m), ED(X_n, Y_{m-1})\right) + 1 \; if \; x_n \neq y_m \end{cases}$$

b) Write down the pseudo code of a function that computed the edit distance of X_n and Y_m [3 Marks]

Answer:

```
ED (X, Y)
{
    for (i=0   to n)
    {
          For ( j=0 to m )
          {
               If ( i =0)  EditDistance[i][j]  = j
              Else If ( j =0)  EditDistance[i][j]  = i
               Else if ( Xn = Ym )    EditDistance[i][j]  = EditDistance[i-1][j-1]
               Else if ( Xn-1 = Ym  and Xn = Ym-1)    EditDistance[i][j]  = EditDistance[i-2][j-
2]  + 1
               Else EditDistance[i][j]  = Min (EditDistance[i-1][j-1] ,    EditDistance[i-1][j],
EditDistance[i][j-1] ) +1

           }
    }
}
```

c) What is the time complexity of your algorithm? [1 Mark]

O (m n)

Question#6

Consider the weighted interval scheduling problem. Here the input is a set of n jobs, each with a start time finish time, and reward. Our task is to schedule some of these jobs on a single machine. The output is a non-overlapping subset of the jobs, and the goal is to maximize the total reward from the jobs in the set.

Consider following greedy strategy:

Sort the jobs by reward and schedule them one by one starting with the highest reward, rejecting any that overlap with jobs already scheduled.

Either prove that this greedy strategy always gives optimal solution or give counter example to prove that it does not guarantee optimal solution. [5 Marks]

Solution:

Counter Example:

Job	1	2	3
Start	1	2	4
Finish	7	3	5
Reward	50	30	30

Q1) Consider the following recursive algorithm

```
K(n){
        IF (n = 1)
                RETURN 1
        ELSE
                SUM = 0
                FOR( i = 1 to n − 1 )
                        SUM = SUM + (K(i) + K(n − i))/3 + n/2
        RETURN SUM
}
```

Convert the recursive code given above into bottom up iterative dynamic programming algorithm.

Solution:

Let's dry run the above code for better understanding.

n= 1, K(1) = 1

n=2, K(2) = (K(1) +K(1))/3 + 2/2 = 2/3+1 = 1.66

n=3, K(3) = [(K(1) +K(2))/3 + 2/2] + [(K(2) +K(1))/3 + 3/2] = [(1+1.66)/3+1] + [(1.66+1)/3+1.5] = [1.88] + [2.38] = 3.26

We can see that we need value of K(2) multiple times. If we can use memoization to store K[2] in an array then we can save repeated computations. The above recursive code is calculating K(2) again and again.

We can create an array DP[i] which saves solution for i. We will need two for loops, one for calculating smaller subproblems and other for calculating answer of one subproblem.

```
Iterative(n){
                DP[1] = 1
                FOR( j = 2 to n )   // j represents size of subproblems
                        SUM = 0
                        FOR( i = 1 to j − 1 )
                                SUM = SUM + (DP[i] + DP[j − i])/3 + j/2
                        DP[j] = SUM
                RETURN DP[n]
}
```

Q2) Consider the following recursive algorithm

```
P(n, k){
        IF (k > n)
                RETURN 0
        IF (k == n || k == 0)
                RETURN 1

        RETURN   P(n-1, k-1) + P(n-1,k)
}
```

Convert the recursive code given above into bottom up iterative dynamic programming algorithm.

Solution

```
P(n, k){
      C[0] = 1

      FOR( i  =  1 to n )
          FOR( j =  0 to min(i, k) )
              IF(j == 0 || j == i)
                    C[i][j] = 1
              ELSE
                    C[i][j] = C[-1][j] + C[i-1][j-1]

       RETURN   C[n][k]
}
```

Q3) Given a matrix M * N of integers where each cell has a cost associated with it, the cost can also be negative. Find the minimum cost to reach the last cell (M-1, N-1) of the matrix from its first cell (0,0). We can only move one unit right (Column No + 1), one unit down (Row No + 1), and one unit in bottom diagonal (Row No + 1, Column No + 1). For example from index (i , j) you can move to (i , j+1), (i+1 , j), and (i+1, j+1) where i = row no and j = column no.

Example

4	7	8	6	4
-6	7	3	9	2
3	8	1	-2	4
7	1	7	3	7
2	9	8	9	3

4	7	8	6	4
-6	**7**	3	9	2
3	8	**1**	**-2**	4
7	1	7	**3**	7
2	9	8	9	**3**

Path with minimum cost = 4 -> -6 -> 7 -> 1 -> -2-> 3 -> 3 = 10

Provide a Dynamic Programming solution for it.

a) Provide recurrence for sub-problem

Solution

C (i , j) = ***Min*** (**C**[i][j - 1] , **C**[i - 1][j] , C[i-1][j-1]) + **A**[i][j]

a) Provide pseudo code for DP solution

```
// Initialization
for(i: 1 to m)
   C[ i ][ 0] = C[i-1][0] + C[i][0]
for(i: 1 to n)
   C[ 0][ i] = C[0][i-1] + C[0][i]


for(i: 1 to m)
        for (j: 1 to n)
            C [i][j] = Min ( C[ i ][ j - 1] , C[ i - 1][ j ] , C[i-1][j-1]) + A[ i ][ j ]
```

Q 1: GreenAir is a new airline company which wants to start operations in a country. The existing airline of that country, BlackJet goes to the routes as defined in the table below. For example BlackJet goes from City A to B and B to A, as all routes are bidirectional.

GreenAir wants to keep their running cost to a minimum in the starting year and cannot offer all the routes as BlackJet provides. They need to select some routes. They ask you which routes to select such that their fuel expenses are minimum. Remember that fuel expense is directly proportional to distance travelled by aircraft.

a) Which routes they should choose such that the distance travelled by their aircrafts is minimum and all cities are still reachable from each other.
b) After 1 year of operation they are in a good position to expand. Petrol prices also went down last year so they are no more concerned about minimizing distance. Now they want to select routes which maximize their profit, as each route has a different ticket price. Which routes should they choose to maximize profit? (They would be interested in choosing route which have high ticket cost. Also they still want that all cities are reachable from all other cities (no city where GreenAir does not provide service)

For both a) and b) you may show your routes by a table or graph and tell how you came up to the solution. Which algorithm you use? Tell if the routes for a) and b) are same or different.

City	City	Distance (km)	Ticket price (PKR)
A	B	400	5k
A	C	500	10k
B	D	100	3K
B	C	200	3K
D	E	700	8K
C	E	300	2K

Q 2: Consider a maze represented as a two dimensional array of numbers, as shown below. The maze can be traversed following any orthogonal direction (i.e., north, south, east and west). Considering that each cell represents a cost, how will you find the minimum cost to travel the maze from the top-left corner to the bottom-right corner of a given maze of size *NxM*

Example matrix

0	3	1	2	9
7	3	4	9	9
1	7	5	5	3
2	3	4	2	5

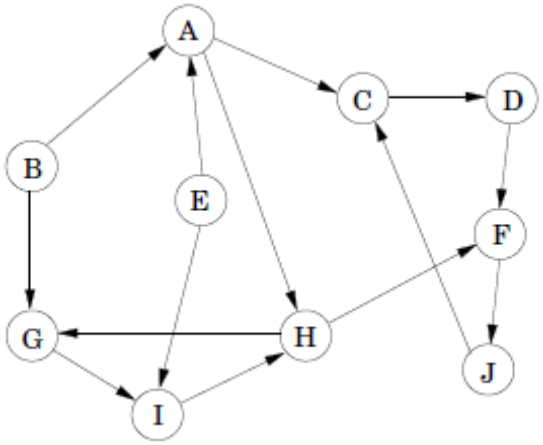
a) How will you solve this problem? Explain in 3 lines in plain English.
b) To find a solution, which known algorithm will you use?
c) Run your algorithm on the above matrix and show how it finds out the minimum cost
   Show your working for each step, in each step you need to choose where to go. Show it like:
   Step 1: A[0,0] start from top left
   Step 2: A[x,y]
   .
   .
   Step Z: A[3,4] which the bottom right corner
d) What is the minimum cost in the above matrix (by running the steps you showed in part c)?

Design & Analysis of Algorithms - Spring 2012
Mid Term 2

April 14, 2012 **Time: 90 min**

Q1. (15)

Run the strongly connected components algorithm on the following directed graphs G. During DFS on G^R: whenever there is a choice of vertices to explore, always pick the one that is alphabetically first



In each case answer the following questions.

a. In what order are the strongly connected components (SCCs) found?
b. Which are source SCCs and which are sink SCCs?
c. Draw the component graph.
d. What is the minimum number of edges you must add to this graph to make it strongly connected?
e. During the final phase of the algorithm, how it is determined, in linear time, without sorting, that the vertices are selected in reverse order of their finishing time.

Q2. (5)

The BFS algorithm finds the shortest distance in terms of number of edges from source node to every other node in the give graph. Can the algorithm, with minor changes, be used to calculate shortest distances for a weighted graph with positive weights? Justify your answer.

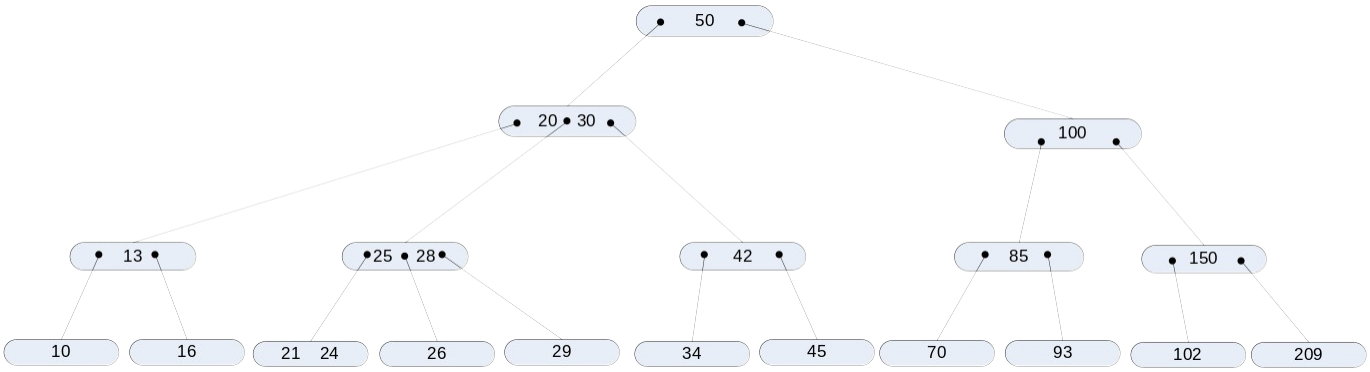
Q3. (10)

Design a linear-time algorithm which, given an undirected graph G and a particular edge **e** in it , determines whether G has a cycle containing **e**. If cycle containing **e** exists then it prints the cycle.

(Note: You have to give detailed pseudo-code. Precisely state your assumptions and don't make any unrealistic assumption for Q1 & Q3)

Q4. (for section C only) (10)

i) Convert the following 2-3 B-tree to 1-2 skip list



ii) Delete 100 from your skip list

iii) After 100 Delete 16 from the resulting skip list

Q4. (for section A &B only) (10)

i. Delete 150 from the following B-Tree of degree t=3

ii. Insert 148 in the resulting B-Tree of (i) above.

iii. Delete 43 from the following 1-2-3 skip list. (You should use "<=" comparison in the algorithm)
(Note: Please show all steps)