Q1. [10+10 pts]

(a) Write a recurrence to describe the running time T(n) of the following function. Solve the recurrence and given a big Oh bound on the running time.

```
MadSummation (A, left, right)
    //Comment: A in an array of integers, of size n [Assume that n is a power of 7]
     size← right-left+1
     IF size ≥ 7
            seventh_part ← FLOOR(size / 7)
            sum←0

           For i←1 to 7
                  sum← sum+ MadSummation(A, left, left + seventh_part)
                  left ← left + seventh_part
    ELSE
                  sum←0
                For i←left to right
                        sum← sum + A[i]
    return sum
```

(b) Recall that the Partition (A, p, r) procedure of quick sort returns an index q such that each element of the sub-array A[p… q-1] is less than or equal to A[q] and each element of A[q+1… r] is greater than A[q]. Modify the partition procedure such that it produces two indices q and t, where p<= q<= t<=r, such that

- all elements of A[q… t] are equal to A[q],
- each element of A[p…q-1] is less than A[q], and
- each element of A[t+1…r] is greater than A[q].

First give a brief explanation of your method in English, then give C++ code.

For an n sized array, A, the running time of your method should be O(n).

Also note that p are r are the starting and ending indices of the array.

You cannot call any ready-made function inside your function.

Q2. [5+10+5] The Catalan numbers are defined as follows:

$$C_0 = 1 \quad \text{and} \quad C_{n+1} = \sum_{i=0}^{n} C_i \, C_{n-i} \quad \text{for } n \geq 0;$$

The 0th Catalan number is 1, and the rest are obtained by the recursive formula given above.

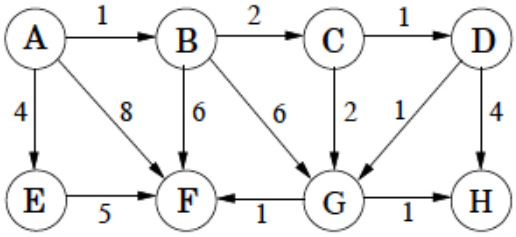
(a) Write a top-down recursive C++ function to compute the nth Catalan number.
(b) Write a bottom-up Dynamic Programming C++ function to compute the nth Catalan number. This function should also allocate the necessary memory required to store sub-problems.
(c) What are the running times in terms of big-Oh of the functions in (a) and (b)

Q3. [6*4] Answer the following questions briefly and top the point, do not write long notes.

(a) Name one feature of quick sort which makes it faster than merge sort in practice. Explain, in at most two lines, why it does so.

(b) If $f(n)=n^{1/2}$ and $g(n)=n^{2/3}$, then which of the following statements is true:
(i) $f(n)=O(g(n))$ (ii) $f(n)=\Omega(g(n))$ (iii) both (i) and (ii)

(c) Suppose Dijkstra's algorithm is run on the following graph. Show the final shortest path tree (take A as the source).



(d) The graph in (c) is also a DAG (directed acyclic graph). Show a linearization of this graph after performing topological sort.

(e) You pay someone k rupees today, and then every day till the end of their life they will keep paying you back half the amount of previous day starting with k/2. (So they will pay you k/2, k/4,…). How many days will it take them to pay back k rupees?

(f) A positive integer is called a perfect number if it is equal to the sum of all of its positive divisors, excluding itself. For example, 6 is the first perfect number, because 6 = 1+2+3. The next is 28 = 14+7+4+2+1. Write a program to find if a number is a perfect numbers. IsPerfect(int num){}

Q4. [5+5+5] Assume you are working in a software house and you are given a new project to work on. Following are the tasks to be done with their duration and dependency.

Activity	Name	Duration (days)	Depends on
A	Account framework	6	
B	Admin login	4	
C	User login	3	A
D	Add course	4	B
E	Remove course	3	B
F	System configuration	10	
G	Logout	3	E,F
H	View course	2	C,D

Starting date of project is 14th Dec. Answer the following questions. Your solution should be optimal.

a) Suppose you want to find the earliest possible project end date. Given what we have studied in class, how will you model this problem? Show it.

b) How will you find the project end date? Explain.

c) Run your algorithm for part (b), show each step in a table.

Note: Rather than re-inventing your algorithm, reuse or transform the algorithms we have studied in class wherever possible.

Design and Analysis of Algorithms Final Exam Solution

1.

Let the transmitted temp value be x.
Actual Temperature = y = x - 5
Now we have to search this y value in two arrays, which are ordered. We can first check which array
should we search by comparing y with maximum of both arrays. Then we can use binary search.
PS. We may have to search in both arrays, so it'll take 2lgn or O(lgn)

2.

a. The solution is just like applying the Shortest Job Next algorithm. By selecting the vehicle for unloading having the LOWEST unloading time. Therefore, if we sort all vehicle's unloading time in ascending order and select vehicles one by one in sequence, it will minimize the total time in the system for all vehicles.
b. Greedy algorithm
c. O(n log n) as the sorting time will dominate other operations.

3a.

i. Solution

	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5
H_i	7	2	5	20	10
O_i	10	10	50	30	25

Given algorithm: H1+O2+H4+O5 = 7+10+20+25 = 62
Optimal solution: O1+O3+O5 = 10+50+25 = 85

ii. Solution:
Let Opt[i] denote the maximum obtainable marks for problems ito n.
Opt[n] = Max (H[n] , O[n]) = O[n] // initialization
Opt[n+1] = 0
For (i = n-1 down to 1)
Opt[i] = Max (H[i] +Opt[i+1] , O[i] + Opt[i+2])
The final answer will be in Opt[1]
Time Complexity = O(n)

3b.

i. Solution:
Total time = 180 minutes

	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5
$time_i$	50	70	60	100	30
$marks_i$	100	110	50	130	30
marks/ time	2	1.6	0.8	1.3	1

Given algorithm solution: 100 + 110 + 30 = = 240 marks
Optimal solution = 100 + 110 + 50 = 260 marks
ii. Solution:
This problem can be mapped to binary knapsack problem as follows:
Items = Problems
Knapsack capacity = Total Time
Value of Item = marks of problem
Weight of Item = time required for problem
The time complexity is same as binary knapsack O(nT) where n is number of problems and T is total Time.

4.

Solution: Find the strongly connected components of the directed graph and compute the component graph G_c. If there is only one node in G_c then no new road is required. Otherwise,find the source and sink nodes of G_c. It is guaranteed that G_c will have only one source node (all places are reachable from Town Hall). Add a directed road from any vertex in sink to any vertex in the source node of G_c. Do this for each sink node. The minimum number of roads is same as the number of sink nodes.

5a.

a. In the following, let $d_A(x)$ be the shortest distance from A to a node x, and $d_B(x)$ the shortest distance from B to x.

Scheme:
Call Dijkstra on G with A as the source, mark all $d_A(.)$ values.
Call Dijkstra on G with B as the source, mark all $d_B(.)$ values.
Find the node h that minimizes $d_A(h) + d_B(h)$.

Pseudocode:

deliverToBob(G(V, E), A, B)

Create Arrays d_A and d_B
DijkstrawrtA(G, A) //this fills up d_A
DijkstrawrtB(G, B) //this fills up d_B

minSum ← INFINITY, minh ← 0
For each h in V (h!=A and h!=B)
IF(d_B[h]+ d_A[h] < minSum)
minSum ←d_A[h]+ d_B[h]
minh ← h

return h

Total time: O((|V|+|E|)lg|V|)

(b) In this case, let d(x) be the shortest distance from A to x. Let d_R(x) be the shortest distance from x to A (note this is the same as the shortest distance from A to x in the reverse graph G_R).

Scheme:

Call Dijkstra on G with A as the source, mark all d(.) values.
Compute G_R, the reverse graph of G.
Call Dijkstra on G_R with A as the source, mark all d_R(.) values.
Find the node h that minimizes d(h) + d_R(h).

Pseudocode:

dropAtHotel(G(V, E), A, B)

Create Arrays d and d_R to store distances and reverse distances with A as source
DijkstrawrtA(G, A) //this fills up d
G_R← ReverseGraph(G)
DijkstrawrtARev(G_R, A) //this fills up d_R
minSum ← INFINITY, minh ← 0
For each h in V (h!=A and h!=B)
IF(d[h]+ d_R[h] < minSum)
minSum ← d[h]+ d_R[h]
minh ← h
return h
Total time: O((|V|+|E|)lg|V|)

5b.

a. G= (V, E) is an undirected weighted graph. The destinations are the vertices in V. The edges are the connections, and their weights are the pair-wise cabling costs.
b. The most cost effective network must minimize the global cost of joining all the destinations. Hence, the network will be a minimum spanning tree, T. We can find it using Kruskal's algorithm in O(|E|lg|V|).
c. Removing an edge, e, from an MST, T, splits in into two sub-MST, T' and T'' respectively. From cut property, we know that e must be the lightest edge between T' and T''(although this makes intuitive sense, the proof needs a bit more detail). Now we will replace it with the second best option. While making sure that we do not create a cycle and only do linear work.

```
updateNetwork(T=(V, E), e=(a, b))
        //Separate T' and T'' by removing e=(a, b) from E
          //… and by marking the component numbers on the vertices in T' and
T''

Remove e from E

          markComponents (G)
//simply uses a single DFS call to mark the component numbers 1 and 2 on the
nodes of T' and T''

bestSoFar INF
bestEdge nil
        For each edge e'=(a', b') ∈ E
                IF(compnum(a')!= compnum(b'))
                        IF(w(a, b) <bestSoFar)
bestSoFarw(a, b)
bestEdge (a, b)

        Add bestEdge to E
```

Time taken: O(|V|+|E|)

d. We can adapt Prim's MST Algorithm to accomplish this task. All we need to do is start from the governor's office, g. After that we stop after Prim has added k-1 more vertices to the MST. This takes O(|V|+ (k+|E|)lg|V|), i.e. O(|E|)lg|V|) work. Below is the pseudo-code:

```
kMST(G=V, E), g)//where g is the starting point: the governor's office
  For each x in V
        cost(x)  INF
        par(x)   nil

 cost(g)  0
```

```
 H ← BuildMinHeap(V, cost)
  T ← {}
  Repeat k times
x ← H. removeMin()
        T.add(x)
For each {x, y} ∈E such that T.contains(y)==false
IF(w(x, y) < cost(y))
                        H.updateKey(y, w(x, y))
                        cost(y) ←w(x, y)
par(y) ← x
```

Note: Kruskal will not work in this case, as it does not grow a single MST. Prim will work because, inductively: for k=2, naturally it will give the correct MST by picking the node with the lightest edge from g.

From the lightest m-size tree that includes g, it will extend it to the lightest m+1 size tree (including g) by picking the next lightest connection.

In the best case scenario, all of the k lightest edges (cost $5, 000) are selected in the MST. In total an MST has |V| - 1 edges. k of these have cost 5000 and |V|-1-k have 1000 cost.
Total cost: (|V|-1-k)*10,000 + k*5000 = 10, 000|V| - 5000k - 10000

National University of Computer and Emerging Sciences, Lahore Campus

Course:	**Design and Analysis of Algorithms**	**Course Code:**	**CS302**
Program:	**BS(Computer Science)**	**Semester:**	**Spring 2020**
Duration:	**4 Hours + 30 minutes for uploading exam**	**Total Marks:**	**100**
Paper Date:	**09-July-2020**	**Weight**	**50**
Section:	**ALL**	**Page(s):**	**5**
Exam:	**Final Exam**		

Instruction/Notes: Attempt the examination on white paper and submit handwritten neat and concise answers. Submit paper as one PDF file by combining images as one PDF file. Name of file should be your roll number and name.
Your explanation is very important and will be used to assess your own contribution in this exam.
You have 4 hours to solve this exam and extra 30 min to scan and upload your solution. The paper should be uploaded no later than 1:30 pm.
You have choice in solving any one of the Problems 3a or 3b.
Similarly you have choice to solve any one of the Problems 5a or 5b

Problem 1: [20 Marks]

Matt Damon: Another Rescue: NASA sent you and Matt Damon on a space mission and the inevitable happened. Matt went missing! AGAIN!

Now, you are on a star in a far off galaxy. There are n planets to your left and n planets to your right. Matt is on one of these planets. You are low on fuel so you can't go after him. So you decide to use your knowledge of Computer Science to find his current location, using the available data. His space suit has a built in thermal sensor that is transmitting the current temperature of his location. You have lost all other communication with him.

NASA has sent you two reports from previous expeditions of this galaxy. One of the reports contains temperatures of all the planets on your left, in decreasing order. The second report has temperatures of all the planets on your right, in increasing order. Each planet has a distinct value of temperature. You also know that the presence of a human body raises the temperature of that planet by 5 degrees. Using this information, devise an algorithm to detect Matt's location that takes no more than O(log n) time.

First give a 3-4 line explanation of your idea and then give detailed pseudo code of your algorithm. Also show that how your designed algorithm is O(log n)

Problem 2: [20 Marks]

When XYZ Sugar Mill opens each morning at 8:00 AM, there is queue of ***n*** vehicles waiting to unload that have travelled from different farms. The XYZ sugar mill owns these vehicles they just pay the drivers by the hours of unloading, and thus would like to find an efficient order in which to unload the vehicles. Let u_i be the time required to unload vehicle i. Keep in mind that the drivers are indifferent to the order in which vehicles are unloaded since they are paid by the hour so if they have to wait more than will get paid more as well.

So, the XYZ Sugar Mill wants to minimize

$$T_n = \sum_{i=1}^{n} t_i$$

Where T_n is the total time to unload n vehicles.

$t_i = T_{i-1} + u_i$

The time required to unload a vehicle depends on its capacity and the type and size of unloading hatches. For example, some vehicles have large, automatically operated hatches; others have small, manually operated hatches. The waiting time of a truck while other trucks are being unloaded (T_{i-1}) is also counted in its time (t_i).

Your task is to design and algorithm to let the XYZ Sugar Mill optimize its chosen objective function given u_i, where $1 \leq i \leq n$.

i. Write pseudo code (or steps) that defines how the algorithm works
ii. What kind of algorithm is it: Divide and Conquer, Greedy, dynamic programming or Brute Force etc?
iii. What is the running time of the algorithm? Explain in few sentences.

Solve any of the following two Problems (3(a) OR 3(b)) [20 Marks]

Problem 3(a):

Suppose your Design and Analysis of Algorithms professor has given you a really tough take home exam having total n challenging problems. You have to solve the problems in the given sequence. Each problem carries different marks. You have limited time and you cannot attempt all problems.

Your professor has offered to give hints for each problem. The marks for correct solution of each problem are reduced if you ask for hint for any problem. Each problem's hint is different according to the problem so the marks are also reduced differently for each problem.

If you decide to solve the problem without any hint then due to stress of solving the problem you will have to skip the next problem.

Your goal is to find which problems should be attempted and for which problem hint should be used in order to maximize total marks.

Suppose the exam has 5 problems. Following table gives total marks of original problem O_i and marks if hint is used H_i for each problem.

	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5
H_i	7	2	5	15	10
O_i	10	10	15	50	25

The optimal solution for this exam is to attempt first problem using hint, attempt second problem without hint and attempt fourth problem without hint.

Optimal = H1+O2+O4 = 7+10+50 = 67

Note that since we have decided to solve second problem without hint so we have to skip the third problem. Similarly we also skip fifth problem.

i. Show that following algorithm does not give optimal answer by giving a counter example. You should show the optimal solution and the solution found by this algorithm.

```
i =0, totalMarks = 0
while ( i <= n ) {
   if  ( O[i] > H[i] + O[i+1]) {
            Print "Attempt Problem i without hint"
            totalMarks += O[i]
        }
    else {
            totalMarks += H[i] + O[i+1]
        }
   i += 2
  }
```

ii. Give an efficient algorithm using dynamic programming to solve the given problem of selecting questions in order to maximize total marks. Also write its time complexity.

OR

Problem 3(b):

Suppose your Design and Analysis of Algorithms professor has given you a really tough take home exam having total n challenging questions. Each question has different marks and different amount of time to solve the question. The time required to solve the question is also given with each question. It is not necessary that the question requiring more time has more marks. There is no relation between the time required to solve the question and its total marks. The professor has announced that she will not give partial credit to incorrect or incomplete solutions and there will be binary marking. The paper is so lengthy that you cannot attempt all questions in given time so you have to choose the questions wisely.

Devise an efficient algorithm to select which questions should be solved in order to maximize total marks. You are given n questions, with marks and time (minutes) for each question. Total time (T minutes) to solve the paper is also given.

For example, suppose the following table gives the 4 questions of the exam. The total time to solve the paper is 180 minutes.

	Problem 1	**Problem 2**	**Problem 3**	**Problem 4**
time$_i$	50	70	60	100
marks$_i$	100	200	50	400

Optimal solution = 200 + 400 = 600 marks

i. Show that following algorithm does not give optimal answer by giving a counter example. You should show the optimal solution and the solution found by this algorithm.

```
Sort all questions by marks/time ratio in descending order
and number them from 1 to n such that the problem numbered
1 has highest marks/time ratio.
remianingTime = totalTime
totalMarks = 0
i = 1
While (i <= n AND reminingTime >= 0) {
   If (time[i] <= remianingTime) {
        totalMarks += marks[i]
        remianingTime = remianingTime  - time[i]
    }

i = i+1
}
Return totalMarks
```

ii. Give an efficient algorithm using dynamic programming to solve the given problem of selecting questions in order to maximize total marks. Also write its time complexity.

Problem 4: [20 Marks]

[Town Planning] The government of Pakistan is planning to build a new town near Lahore to provide houses to homeless people. Initially, it was planned that all the roads will be two way. However, the newly elected government changed the plan and mark all the roads as one way roads. The directions of the roads are marked in a way that that all places are reachable from the Town Hall. Due to this change, not all the places in the town are reachable from each other. In other words there are some pair of places that has no directed path between them. This means some new roads must be constructed to provide path between all those pair of places that are not connected in this one way road network. The government wants to spend minimum amount of money on this project. You are asked to analyze the current road network and add minimum possible new roads such that all pair of places has a path between them. Formulate this problem as a graph problem and give a linear time algorithm that can compute the new edges (roads) given the existing road network.

First give a 3-4 line explanation of your idea and then give detailed pseudo code of your algorithm. Also show that how your designed algorithm is linear.

Solve any of the following two Problems (5(a) OR 5(b)) [20 Marks]

Problem 5(a):

Agents Alice and Bob have been stationed in the enemy territory at houses A and B respectively. At one point during their mission, Alice needs to hand over a bag containing high-valued secret items to Bob. The contents of this bag are top secret and a third party cannot be trusted to deliver it. It is decided that Alice and Bob will meet each other at a third location, a hotel, where the bag will be handed over. The agency has already prepared a map of the enemy territory. It contains the homes A, B, and the n hotels in the area:, h_1, h_2, …, h_n. Like most road maps, this map is a directed graph, but it also contains edge weights. The weight of an edge represents the risk – a positive number with the higher values indicating higher risk – of

being intercepted by enemy agents on that particular road. The total risk of a path is simply the sum of the edge-weights on that path.

i. The agency wishes to tell Alice and Bob which hotel, h, to meet at so that the total risk, for both of them combined, is minimized. Such a hotel would be the *safest hotel*. Note that both Alice and Bob would need to travel to h to make the delivery possible. The problem is that the map is too large for manual processing. Therefore, you have been tasked to write an algorithm that can find the safest hotel. Your goal is to write an algorithm that takes no more than $O((|V|+|E|)\lg|V|)$ to accomplish this task.

ii. Unfortunately, before the delivery could take place the enemy spies have grown suspicious of Bob. His house at B is constantly monitored. He simply cannot step out. The job for Alice has therefore changed. She must now drop the bag at hotel, h, and come back to A, so that Bob could collect it at a later time when the situation has improved. The definition of the safest hotel has changed too. It is now the hotel, h, that minimizes the total risk for Alice to make the drop. Note that Alice must travel to h and come back to her house at A: the total risk will be the risk of the round-trip. Your goal now is to write an algorithm that takes no more than $O((|V|+|E|)\lg|V|)$ to accomplish this task.

OR

Problem 5(b):

The city of Mountana is planning a chair-lift network between N destination points in the city. Since this is a mountainous city, chair-lift promises to be a good alternative to traditional modes of transportation. The city's governor has obtained a map of the N destinations along with the direct aerial distance between every pair of them. This distance is the length of the cable that would be needed to connect two destinations and hence represents the cabling cost of the connection. This large map has been handed over to you to suggest a most cost-effective way of creating the chair-lift network that makes it possible for the citizens to travel from any one destination to any other. All chair-lift cables are bidirectional.

i. Model this problem as a graph problem. Define a graph G=(V, E). What type of a graph is this? What do the vertices, V, and the edges E and their weights represent in this graph?

ii. Which algorithm will you use to find the most cost-effective chair-lift network? What will be the running time, in terms of |V| and |E|, of this algorithm?

iii. The ideal network may not be possible due to various issues. For example, the wind speed between two destinations or the elevation difference between them might make it too difficult to connect them directly even if the cost is low. After providing the program that produces the best chair-lift network, you are now asked to add the functionality that will help the governor's office to modify the network by choosing to disallow a certain connection in the existing network. When a connection is disallowed, the chair-lift network cannot use that connection anymore. Your algorithm should update the best network accordingly. It should take linear time to do so, i.e. $O(|V|+|E|)$.

iv. According to a new survey conducted by the governor's office, it has been estimated that a direct connection between any two destinations in the city can in fact be cabled in $10, 000, regardless of the distance between them. This means that the cost of each connection becomes a fixed amount of $10, 000. Additionally, k of these connections have been sponsored by a beverage company and can each be cabled in only $5, 000. Without even computing the ideal network, can you tell what could be the minimum possible cost of the network under the new scenario?

National University of Computer and Emerging Sciences, Lahore Campus

	Course:	**Design and Analysis of Algorithms**	**Course Code:**	**CS302**
	Program:	**BS(Computer Science)**	**Semester:**	**Spring 2020**
	Duration:	**4 Hours + 30 minutes for uploading exam**	**Total Marks:**	**100**
	Paper Date:	**09-July-2020**	**Weight**	**50**
	Section:	**ALL**	**Page(s):**	**5**
	Exam:	**Final Exam**		

Instruction/Notes: Attempt the examination on white paper and submit handwritten neat and concise answers. Submit paper as one PDF file by combining images as one PDF file. Name of file should be your roll number and name.
Your explanation is very important and will be used to assess your own contribution in this exam.
You have 4 hours to solve this exam and extra 30 min to scan and upload your solution. The paper should be uploaded no later than 1:30 pm.
You have choice in solving any one of the Problems 3a or 3b.
Similarly you have choice to solve any one of the Problems 5a or 5b

Problem 1: [20 Marks]
Matt Damon: Another Rescue: NASA sent you and Matt Damon on a space mission and the inevitable happened. Matt went missing! AGAIN!

Now, you are on a star in a far off galaxy. There are n planets to your left and n planets to your right. Matt is on one of these planets. You are low on fuel so you can't go after him. So you decide to use your knowledge of Computer Science to find his current location, using the available data. His space suit has a built in thermal sensor that is transmitting the current temperature of his location. You have lost all other communication with him.

NASA has sent you two reports from previous expeditions of this galaxy. One of the reports contains temperatures of all the planets on your left, in decreasing order. The second report has temperatures of all the planets on your right, in increasing order. Each planet has a distinct value of temperature. You also know that the presence of a human body raises the temperature of that planet by 5 degrees. Using this information, devise an algorithm to detect Matt's location that takes no more than O(log n) time.

First give a 3-4 line explanation of your idea and then give detailed pseudo code of your algorithm. Also show that how your designed algorithm is O(log n)

Problem 2: [20 Marks]
When XYZ Sugar Mill opens each morning at 8:00 AM, there is queue of $\boldsymbol{n}$ vehicles waiting to unload that have travelled from different farms. The XYZ sugar mill owns these vehicles they just pay the drivers by the hours of unloading, and thus would like to find an efficient order in which to unload the vehicles. Let $\boldsymbol{u_i}$ be the time required to unload vehicle i. Keep in mind that the drivers are indifferent to the order in which vehicles are unloaded since they are paid by the hour so if they have to wait more than will get paid more as well.

So, the XYZ Sugar Mill wants to minimize

$$T_n = \sum_{i=1}^{n} t_i$$

Where T_n *is the total time ¿unload n vehicles*.

$t_i = T_{i-1} + u_i$

The time required to unload a vehicle depends on its capacity and the type and size of unloading hatches. For example, some vehicles have large, automatically operated hatches; others have small, manually operated hatches. The waiting time of a truck while other trucks are being unloaded (T_{i-1}) is also counted in its time (t_i).

Your task is to design and algorithm to let the XYZ Sugar Mill optimize its chosen objective function given $\boldsymbol{u_i}$, where $\boldsymbol{1 \leq i \leq n}$.

i. Write pseudo code (or steps) that defines how the algorithm works
ii. What kind of algorithm is it: Divide and Conquer, Greedy, dynamic programming or Brute Force etc?
iii. What is the running time of the algorithm? Explain in few sentences.

Solve any of the following two Problems (3(a) OR 3(b)) [20 Marks]

Problem 3(a):

Suppose your Design and Analysis of Algorithms professor has given you a really tough take home exam having total n challenging problems. You have to solve the problems in the given sequence. Each problem carries different marks. You have limited time and you cannot attempt all problems.

Your professor has offered to give hints for each problem. The marks for correct solution of each problem are reduced if you ask for hint for any problem. Each problem's hint is different according to the problem so the marks are also reduced differently for each problem.

If you decide to solve the problem without any hint then due to stress of solving the problem you will have to skip the next problem.

Your goal is to find which problems should be attempted and for which problem hint should be used in order to maximize total marks.

Suppose the exam has 5 problems. Following table gives total marks of original problem O_i and marks if hint is used H_i for each problem.

	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5
H_i	7	2	5	15	10
O_i	10	10	15	50	25

The optimal solution for this exam is to attempt first problem using hint, attempt second problem without hint and attempt fourth problem without hint.

Optimal = H1+O2+O4 = 7+10+50 = 67

Note that since we have decided to solve second problem without hint so we have to skip the third problem. Similarly we also skip fifth problem.

i. Show that following algorithm does not give optimal answer by giving a counter example. You should show the optimal solution and the solution found by this algorithm.

```
i =0, totalMarks = 0
while ( i <= n ) {
   if  ( O[i] > H[i] + O[i+1]) {
            Print "Attempt Problem i without hint"
            totalMarks += O[i]
         }
    else {
            totalMarks += H[i] + O[i+1]
         }
   i += 2
 }
```

ii. Give an efficient algorithm using dynamic programming to solve the given problem of selecting questions in order to maximize total marks. Also write its time complexity.

OR

Problem 3(b):

Suppose your Design and Analysis of Algorithms professor has given you a really tough take home exam having total n challenging questions. Each question has different marks and different amount of time to solve the question. The time required to solve the question is also given with each question. It is not necessary that the question requiring more time has more marks. There is no relation between the time required to solve the question and its total marks. The professor has announced that she will not give partial credit to incorrect or incomplete solutions and there will be binary marking. The paper is so lengthy that you cannot attempt all questions in given time so you have to choose the questions wisely.

Devise an efficient algorithm to select which questions should be solved in order to maximize total marks. You are given n questions, with marks and time (minutes) for each question. Total time (T minutes) to solve the paper is also given.

For example, suppose the following table gives the 4 questions of the exam. The total time to solve the paper is 180 minutes.

	Problem 1	**Problem 2**	**Problem 3**	**Problem 4**
time$_i$	50	70	60	100
marks$_i$	100	200	50	400

Optimal solution = 200 + 400 = 600 marks

i. Show that following algorithm does not give optimal answer by giving a counter example. You should show the optimal solution and the solution found by this algorithm.

```
Sort all questions by marks/time ratio in descending order
and number them from 1 to n such that the problem numbered
```

```
1 has highest marks/time ratio.
remianingTime = totalTime
totalMarks = 0
i = 1
While (i <= n AND reminingTime >= 0) {
   If (time[i] <= remianingTime) {
        totalMarks += marks[i]
        remianingTime = remianingTime  - time[i]
    }

i = i+1
}
Return totalMarks
```

ii. Give an efficient algorithm using dynamic programming to solve the given problem of selecting questions in order to maximize total marks. Also write its time complexity.

Problem 4: [20 Marks]

[Town Planning] The government of Pakistan is planning to build a new town near Lahore to provide houses to homeless people. Initially, it was planned that all the roads will be two way. However, the newly elected government changed the plan and mark all the roads as one way roads. The directions of the roads are marked in a way that that all places are reachable from the Town Hall. Due to this change, not all the places in the town are reachable from each other. In other words there are some pair of places that has no directed path between them. This means some new roads must be constructed to provide path between all those pair of places that are not connected in this one way road network. The government wants to spend minimum amount of money on this project. You are asked to analyze the current road network and add minimum possible new roads such that all pair of places has a path between them. Formulate this problem as a graph problem and give a linear time algorithm that can compute the new edges (roads) given the existing road network.

First give a 3-4 line explanation of your idea and then give detailed pseudo code of your algorithm. Also show that how your designed algorithm is linear.

Solve any of the following two Problems (5(a) OR 5(b)) [20 Marks]

Problem 5(a):

Agents Alice and Bob have been stationed in the enemy territory at houses A and B respectively. At one point during their mission, Alice needs to hand over a bag containing high-valued secret items to Bob. The contents of this bag are top secret and a third party cannot be trusted to deliver it. It is decided that Alice and Bob will meet each other at a third location, a hotel, where the bag will be handed over. The agency has already prepared a map of the enemy territory. It contains the homes A, B, and the n hotels in the area:, h_1, h_2, …, h_n. Like most road maps, this map is a directed graph, but it also contains edge weights. The weight of an edge represents the risk – a positive number with the higher values indicating higher risk – of being intercepted by enemy agents on that particular road. The total risk of a path is simply the sum of the edge-weights on that path.

i. The agency wishes to tell Alice and Bob which hotel, h, to meet at so that the total risk, for both of them combined, is minimized. Such a hotel would be the *safest hotel*. Note that both Alice and Bob would need to travel to h to make the delivery possible. The problem is that the map is too large for manual processing. Therefore, you have been tasked to write an algorithm that can find the safest hotel. Your goal is to write an algorithm that takes no more than $O((|V|+|E|)\lg|V|)$ to accomplish this task.

ii. Unfortunately, before the delivery could take place the enemy spies have grown suspicious of Bob. His house at B is constantly monitored. He simply cannot step out. The job for Alice has therefore changed. She must now drop the bag at hotel, h, and come back to A, so that Bob could collect it at a later time when the situation has improved. The definition of the safest hotel has changed too. It is now the hotel, h, that minimizes the total risk for Alice to make the drop. Note that Alice must travel to h and come back to her house at A: the total risk will be the risk of the round-trip. Your goal now is to write an algorithm that takes no more than $O((|V|+|E|)\lg|V|)$ to accomplish this task.

OR

Problem 5(b):

The city of Mountana is planning a chair-lift network between N destination points in the city. Since this is a mountainous city, chair-lift promises to be a good alternative to traditional modes of transportation. The city's governor has obtained a map of the N destinations along with the direct aerial distance between every pair of them. This distance is the length of the cable that would be needed to connect two destinations and hence represents the cabling cost of the connection. This large map has been handed over to you to suggest a most cost-effective way of creating the chair-lift network that makes it possible for the citizens to travel from any one destination to any other. All chair-lift cables are bidirectional.

i. Model this problem as a graph problem. Define a graph G=(V, E). What type of a graph is this? What do the vertices, V, and the edges E and their weights represent in this graph?
ii. Which algorithm will you use to find the most cost-effective chair-lift network? What will be the running time, in terms of |V| and |E|, of this algorithm?
iii. The ideal network may not be possible due to various issues. For example, the wind speed between two destinations or the elevation difference between them might make it too difficult to connect them directly even if the cost is low. After providing the program that produces the best chair-lift network, you are now asked to add the functionality that will help the governor's office to modify the network by choosing to disallow a certain connection in the existing network. When a connection is disallowed, the chair-lift network cannot use that connection anymore. Your algorithm should update the best network accordingly. It should take linear time to do so, i.e. $O(|V|+|E|)$.
iv. According to a new survey conducted by the governor's office, it has been estimated that a direct connection between any two destinations in the city can in fact be cabled in $10, 000, regardless of the distance between them. This means that the cost of each connection becomes a fixed amount of $10, 000. Additionally, k of these connections have been sponsored by a beverage company and can each be cabled in only $5, 000. Without even computing the ideal network, can you tell what could be the minimum possible cost of the network under the new scenario?

CLO #4: Implement the algorithms, compare the implementations empirically, and apply fundamental algorithmic knowledge to solve practical problems related to the program.

Q2) It is observed in NUCES that our LAN is not properly designed and is not using the available resources to its fullest. As you people are familiar with the fact that the NUCES Lahore campus consists of various building blocks so the LAN needs to connect the classrooms, labs, faculty offices in these buildings efficiently.

We have modeled this problem to a graphs, where computers in classrooms, labs, offices are represented by nodes and the networking wires that connect them are represented by edges. The edges have weight that indicates the length of the wire needed to connect two nodes. The management wants to design a LAN network, FLAN that covers all the nodes with minimum wire length and do so efficiently.

a. **a. Which graph algorithm (that you have already studied in class you think is most suitable to compute the FLAN and what is time complexity in term of |V| and |E|?**

b. **b. The ideal FLAN may not be possible due to various issues. For example, the optical fiber might need underground wiring which means additional cost even if the length of the required wire is short. After providing the program that produces the best FLAN, you are now asked to add the functionality that will help the director's office to modify the networking by choosing to disallow a certain connection in the existing network. When a connection is disallowed, the FLAN cannot use that connection anymore. Your algorithm should update the best network accordingly. It should take linear time to do so i.e O(|V|+|E|).**

c. **c. A recent technology has been introduced and it has been estimated that a direct connection between any two nodes in the network can in fact be cabled in $5000, regardless of the distance between them. This means that the cost of each connection becomes a fixed amount of $1000. Additionally, k of these connections have been sponsored by a software house. Without even computing the ideal network, can you tell what could be the minimum possible cost of the network under the new scenario that FAST needs to pay?**

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:

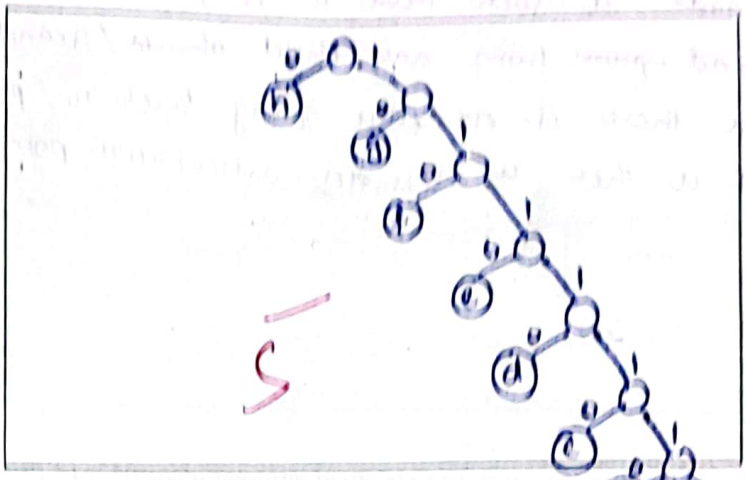
a. **Given a binary string 5 of type(1^m0^n), devise an algorithm that finds the number of zeroes in 0(log k) time. Here m+n=k. Write 3-4 lines to explain your idea.**

Binary String = {1111000000}
we use divide and conquer approach to divide the string. Here upon dividing we get a zero so we have atleast $\frac{k}{2}+1$ zeroes. we can have more so we divide the right string
if we get 0 again we repeat and add the half of the string we, in our answer. we stop when we get 1, then we divide the right substring and repeat the recursive process

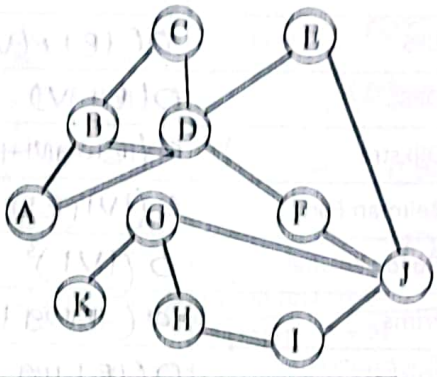
```
zeroes(s, l, r){
  m = l+r/2
  if s[m]='0'
    return 1+zeroes(s, l, m);
  else
    return zeroes(s, m, r);
}
```

b. Few days back I generated these codes using Huffman coding. Unfortunately, I lost the paper where I drew that tree. Please help me in re-generating the tree using given codes

Character	Code
a	1111110
b	1111111
c	111110
d	11110
e	1110
f	110
g	10
h	0



c. Suppose you are a medical practitioner and it is your job to vaccinate people in each village of Pakistan to limit the spread of adenovirus in the country. The adenovirus can be transmitted between two people if they handshake. For each village, you are given a handshaking graph, G, whose vertices are the people in that village and whose edges are pairs of people who regularly handshake. You have limited supply of the vaccine, and you must have to vaccinate the central hand shakers only, where in the graph a central hand shaker is a person D, such that there are no two people C and E, who are hand shake by D such that there is a sequence of handshaking pairs of people that starts with C and leads to E while avoiding D.



I. Which vertices are the central hand shakers in this graph?

II. Describe an efficient algorithm for identifying all the central hand shakers in G, and provide its running time. Explain your idea in 3-4 lines.

(i) J, G, D are central hand shakers

(ii) We need to find articulation points. We can use modified DFS to compute the low value for every vertex low[v] = min(d[v], lowest low[w] from tree edges, lowest d[w] in back edges)
if there exists a node u such that for a child w low[w] ≥ d[u] then u is articulation point / central hand shaker.

Running time = O(|E| + |V|)

In DFS tree, for root if root has two or more children it is central handshaker

Q) How can we say in DFS tree if there exist a node u such that it is not root of the tree and you can't any child of u / grandchild of u there is no edge going back to proper ancestor of u then u is an articulation point / cut vertex.

TO #2: Analyze the time and space complexity of different algorithms by using standard asymptotic notations for recursive and non-recursive algorithms.

[Marks: 5 + 5]

a. Fill the Table. Also mention the choice of data structure for graph algorithms.

Algorithm Name	Worst Case Time Complexity	
BFS	O(|E| + |V|) ✓	using Queue DS ✓
DFS	O(|E|+|V|) ✓	adjacency list for graph
Dijkstra	O(|E|log|V|+|V|log|V|) ✓	Priority Queue
Bellman Ford	O(|V||E|) ✓	adjacency list for Graph
Floyd Warshal	O(|V|)³ ✓	adjacency list for Graph
Prims	O(|E|log|V|) ✓	Priority Queue
Kruskal	O(|E| log |V|) ✓	Priority Queue ✓
Activity Selection Problem	O(n²) ✗	✓
Huffman Coding	O(|V| log|V|) nlogn ✓	using MinHeap
0-1 knapsack Problem	O(n²) / O(W*I) ✓	W = total weight, I = no. of items
Fractional Knapsack Problem	if sorted already then O(n) otherwise, n log n ✓	
Edit Distance	O(n²) / O(n1×n2)	n1 = length of string1, n2 = length of string 2
Rod Cutting	O(n²) / O(L * pieces)	L = length of rod, pieces = possible pieces we can cut in

National University of Computer and Emerging Sciences
Lahore Campus

b. *Derive the recurrence of the given recursive function and solve the recurrence.*

```
tribonacci ( n )
{
  If n == 0
     return 0
  else if ( n == 1  ||  n == 2 )
     return 1
  else
     return  tribonacci ( n/2 )  +  tribonacci ( n/4 ) + tribonacci ( n/4 )
}
```

$T(n) = T(\frac{n}{2}) + 2T(\frac{n}{4}) + c$

height of tree will be decided by left most subtree $(\frac{n}{2})$.

level	p.b size	no. of nodes	cost. 1 node	cost level
0	n	1	c	$1c$
1	$n/2$	3	c	$3c$
2	$n/4$	9	c	$9c$
i	$n/2^i$	3^i	c	$3^i.c$
k	$\frac{n}{2^k} = 1$	3^k	c	$3^k.c$

$\frac{n}{2^k} = 1$

$n = 2^k$

$k = \log_2 n$

$T.C = 1c + 3c + 9c + \cdots 3^k c$

$= c[1 + 3 + 9 \cdots 3^k]$

Number of levels $= \log_2 n + 1$

cost at one level $= c$

Time complexity $= (\log_2 n + 1)^* c$

$= c\log_2 n + c$

$= O(\log_2 n)$

CLO #3: Evaluate the correctness of algorithms by using theorem proving or executing test cases

[Marks: 5 + 5]

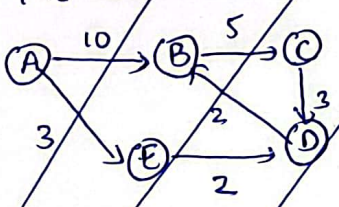
Q5:

a. Following algorithm takes a connected graph and a weight function as input and returns a set of edges T. Prove that T is a minimum spanning tree or prove that T is not a minimum spanning tree.

```
MAYBE-MST (G,w)
1    T ← Nil
2    for each edge e, taken in arbitrary order
3    T ← T U { e }
4    if T has a cycle c
5          let e' be a maximum-weight edge on c
6          T ← T − { e' }
7  return T
```

Answer on Answer Sheet.

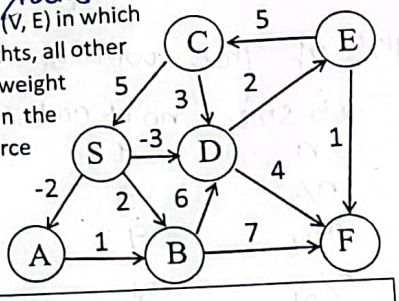
T is not MST. Consider counter example



we will randomly select edge A,B then B,C, C,D and D,B cycle exist so we remove B,C then we take E,D and A,E no cycle exist but the M.ST we get is wrong.

Also by removing the edge B,C which was the max weight edge in cycle we lose a path to node C, so this algorithm fails..

b. Suppose that you are given a weighted, directed graph G = (V, E) in which edges that leave the source vertex may have negative weights, all other edge weights are nonnegative, and there are no negative-weight cycles an example of such graph is provided below. Can the Dijkstra's algorithm correctly find shortest paths from source S? Provide the valid justification.



Yes, such algorithm will provide shortest path.

Problem occured when we took Djikstra and a node could not be relaxed because we already relaxed it and the negative edge did not contribute to shortest path in final answer. Here Djikstra will first account for negative weight edge and relax the nodes connected to source since they get relaxed then there is no chance we will miss out on any relaxation.

FINAL EXAM ANSWER SHEET

National University
of Computer & Emerging Sciences

FAST

16/5/24

SEMESTER : ☑ SPRING ☐ SUMMER ☐ FALL Year 2024

Please Tick Appropriate Boxes

Course Design & Analysis of Algorithms Serial No./ 05352

Roll No. 221-0504 Section BBS-4A Date

Serial No. of continuation sheet(s) attached

INSTRUCTIONS FOR CANDIDATES

- Write Question No. In the middle of the line using thick tipped pen.
- Use only blue or black pen to write your answers.
- Answers written using pencil will not be checked.
- Pencil is only allowed to draw diagrams or write program code.

(THIS ANSWER BOOK CONTAINS NOS. 1-18)

Q./Part No.	Marks
Q. - 1	
Q. - 2	
Q. - 3	
Q. - 4	
Q. - 5	
Q. - 6	
Q. - 7	
Q. - 8	
Q. - 9	
Q. - 10	

Q./Part No.	Marks
Q. - 11	
Q. - 12	
Q. - 13	
Q. - 14	
Q. - 15	
Q. - 16	
Q. - 17	
Q. - 18	
Q. - 19	
Q. - 20	

Marks Obtained

Total Marks

Examiner's Signature

Date

art No.

Rough Work

Question 01

Recursive

MaxProfit [t, b, n] = max [maxProfit(t, b, n-1), } if Tn ≤ t
maxProfit(t-tn, b, n-2) + bn]

= maxProfit(t, b, n-2) if Tn > t

We make two calls for the MaxProfit
for one call we pass
MaxProfit(t, b, T, N)
for other we pass
MaxProfit(t, b, T, N-1)

The max of these two results will be our final answer.

Bottom-up solution

```
MaxProfit(t, b, T, N) {
    init dp[0:T][0:N]
    for (i=0 to T)
        dp[i][0] = 0
    for (i=0 to N)
        dp[0][i] = 0

    for (i=0 1 to T)
        for (j=1 to N) {
            if (T < t[j]) {
                dp[i][j] = dp[i-1][j];
            } else {
                dp[i][j] = max( dp[i-t[i]][j-1] + b
                                dp[i-1][j] );
```

Q / Part No.

```
if (i==1){
    if(T<t[1])
        dp[i][j]=0;
    else dp[i][j]= b[i];
}

MaxProfit (b,t,T,N){

int dp[0:N][0:T];

for (i=0 to N)
    dp[i][0]=0
for (i=0 to T)
    dp[0][i]=0

for (i=1 to N)  <- increment i by 2
    for (j=1 to T){

else{   if(T<t[i])
        if (T< t[i])
            dp[i][j]= dp[i-1][j];
        else{

            dp[i][j]= max( dp[i-2][j],
dp[i-2][j-t[i]]+ b[i] )
        }
}
    }

for (i=2 to N) (increment i by 2)
    for (j=2 to N){

        if (T< t[i])
            dp[i][j]= dp[i-2][j];
        else{
            dp[i][j]= max( dp[i-2][j],
dp[i-2][j-t[i]]+ b[i] )
        }
}
```

12

Rough Work

return max(dp[N][T], dp[N-1][T/2]);

}

Time complexity = ~~O(T*N)~~ 2 O(T*N)

first nested loops fills half of dp array the other nested loop fills other half.

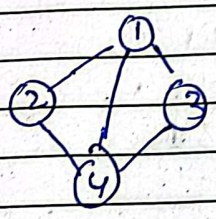
Question. 02

(a) ~~Prims Algorithm for MST~~.

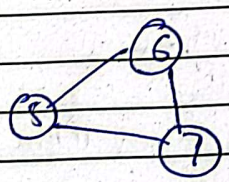
Kruskals Algorithm for MST

Time complexity = O(|E| log |V|) 2

(b) By removing one node we get two separate connected components say A and B.



A



B

we can run DFS on A and color all the nodes black.

Now we run DFS on any node of B. in G

In DFS-visit, if we encounter any node with color black we store that edge in minEdge var we will update minEdge variable if we encounter any other edge such that one node is white and other is black is its weight < minEdge.

Finally we get the minimum Edge after DFS is complete. this minimum edge ≠ the one rem

Q / Part No.

and joins two components back together. This takes O(|V|+|E|) time. The new MST = oldMST ∪ minEdge 10

(c)

If there are |V| nodes then FAST will need to pay

(|V|-1-K) * 1000 $ for the connection 3

Question. 05

(a)

The tree will be a Minimum Spanning Tree.

This method is similar to Kruskal's Algorithm where we pick edges in increasing order of their weight and dont allow any cycle to occur. Here instead of picking edges in increasing order we arbitrarily chose edges and if cycle occur we remove the edge in cycle with max weight. 5

This approach works because we are always removing the edge of cycle with maximum weight

Proof by counter example

Say that T we get is not M.S.T if T is not M.S.T then there exists an M.S.T T' such that T' contains an edge with less weight than one in T. This means T contains an edge with

Rough Work

Part No.

weight greater but we already removed all cycle forming edges in T with max weight. we made optimal choice at occurence of every cycle.
Hence T is a MST.

DFS_visit
time++
d[v] = time
π[v] = 0

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:	Saturday, June 10, 2023	Total Marks:	55
	Section:	ALL	Page(s):	10
	Exam Type:	Final (Solution)	Duration	2 hours 30 minutes

Roll No._____ **Section:**_____

Instruction/Notes: Attempt all questions in the answer sheet in the space provided. You can use extra sheet or rough work, but it should not be submitted. Extra sheet will not be graded.

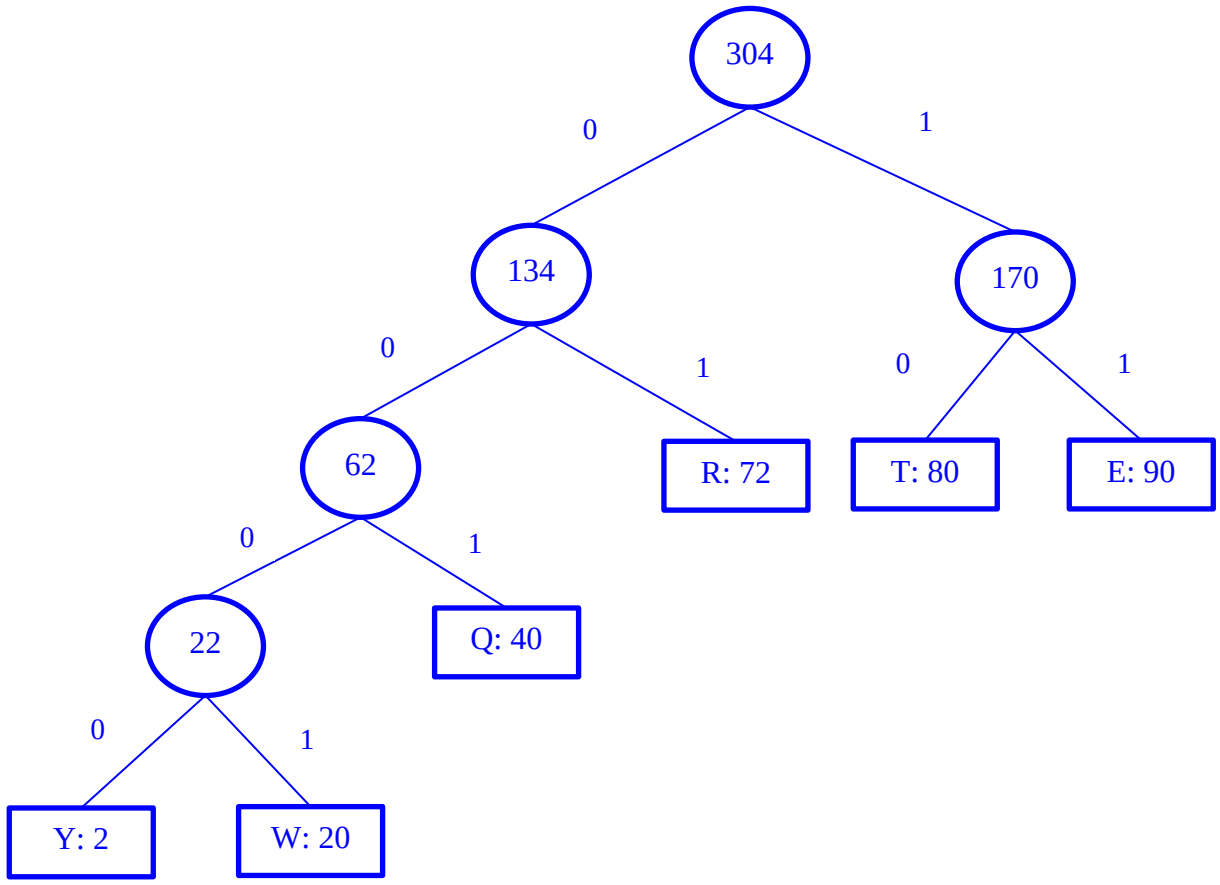
Question #	**Total Marks**	**Marks Obtained**		
		CLO 1	CLO 2	CLO 3
Q1:	10			
Q2(a):	3			
Q2(b),(c):	12			
Q3:	10			
Q4:	10			
Q5(a):	4			
Q5(b),(c):	6			
Total Marks Obtained (**CLO wise**)				
Total Marks Obtained				

Question 1: **[5+5] Marks**

a) A long string consists of five characters Q, W, E, R, T, Y. They appear with the frequency 40, 20, 90, 72, 80 and 2 respectively. Draw the Huffman Tree and write down the Huffman encoding for these six characters. (**No partial marks for this part**)

Alphabet	Huffman Encoding
Q	001
W	0001
E	11
R	01
T	10
Y	0000

The Tree:



b) In the activity selection problem, suppose that instead of always selecting the first activity to finish, we select the last activity to start that is compatible with all previously selected activities. Describe how this approach is a greedy algorithm, and prove that it yields an optimal solution.

As we choose an activity to add to our optimal solution without having to first solve all the subproblems, the approach mentioned above is a greedy one. Now we show that it yields an optimal solution.

Let X be a set of activities and let x be an activity with latest start time in X. Let Y be a maximum-size subset of mutually compatible activities in X; and let y be an activity with latest start time in Y. If y is the same activity x, we are done. Otherwise, let $Z = Y - \{y\} \cup \{x\}$ be a new set of activities.

[**Rubric**] **3 Marks** up to this point. One must have

- isolated an activity with latest start time in the problem
- isolated an activity with latest start time in a solution
- made a new solution by exchanging the two activities

Note: It would be **all three points or none.**

The activities in Z are disjoint, which follows because the activities in Y are disjoint, y is the last activity in Y to start, and $s_y \leq s_x$ (i.e. start time of y is no later than the start time of x).

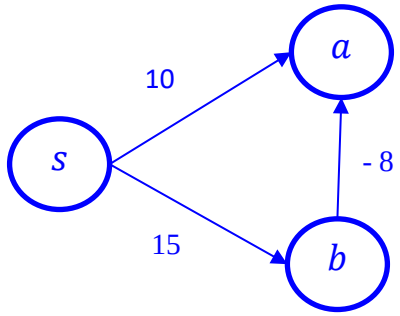
[**Rubric**] **1 Mark** to show that new solution comprises of disjoint activities.

Since $|Z| = |Y| - 1 + 1 = |Y|$, we conclude that Z is a maximum-size subset of mutually compatible activities of X, and it includes x.

[**Rubric**] **1 Mark** to show that new solution comprises of maximum number of disjoint activities; and it includes the already isolated activity with latest start time in the problem.

Question 2: **[3+5+7] Marks**

a) Give an example of a weighted directed graph, G, with negative-weight edges such that Dijkstra's algorithm fails to find the correct shortest path. You must mention the start vertex and the vertex for which Dijkstra's algorithm will compute incorrect shortest path. (**No partial marks for this part**)



start vertex: s

$a.d = 10$

$\delta(s, a) = 7$

b) Suppose that we are given a weighted, directed graph G = (V , E) in which edges that leave the source vertex **s** may have negative weights, all other edge weights are non-negative, and there are no edges going to the source vertex **s**. Describe an algorithm **(basic idea only)** to find the shortest path from **s** to all other vertices.

- Your algorithm must be asymptotically better than the Bellman Ford algorithm.
- Justify (informally, *i.e without using any formal techniques like loop invariant etc.*) the correctness of your algorithm.

Dijkstra's algorithm will work fine in this situation.

[Rubric] 3 Mark

Justification: As there are no negative weight edges starting from an *intermediate* vertex on any path (including the shortest paths), as well as there are no negative weight cycles (as no edges are going to the source vertex); once we have extracted a vertex v, it is not possible that $v.d$ could have been decreased later. Hence, for each vertex $v \in G.V$, $v.d = \delta(s, v)$ at the time v was extracted.

[Rubric] 2 Mark

A more formal argument (optional): We claim that in the given scenario, for each vertex $v \in G.V$, $v.d = \delta(s, v)$ at the time v was extracted. For the sake of contradiction, let's assume

that this is not the case. *For the sake of simplicity, let's assume that there are no zero weight edges.* Let y be the *first* vertex, such that $y.d > \delta(s,y)$ at the time y was extracted. Let x be the last vertex on *a shortest path* from s to y that was extracted before y. At that time, $Relax(x,v,w)$ was called for all $v \in G.Adj[x]$.

- If there were some intermediate vertex $z \in G.Adj[x]$ between x and y on *that* path, then after $Relax(x,z,w)$, $z.d < y.d$ (as $z.d = \delta(s,z) < \delta(s,y) < y.d$), hence, z must have been extracted before y. But that leads to a contradiction (as x is the last vertex on *that* shortest path from s to y that was extracted before y).
- If there were no intermediate vertex $z \in G.Adj[x]$ between x and y on that path, then after $Relax(x,y,w)$, $y.d = \delta(s,y)$, once again leading to a contradiction (as $y.d = \delta(s,y) \rightarrow y.d \not> \delta(s,y)$).

Hence, no such vertex y exists such that $y.d > \delta(s,y)$ at the time y was extracted. In other words, for each vertex $v \in G.V$, $v.d = \delta(s,v)$ at the time v was extracted.

c) Given a graph G and a source vertex s, you have already constructed a minimum spanning tree T using Prim's algorithm (hence, the information $v.\pi$ is available with you for each $v \in G.V$). Suppose that now the weight of one of the edges (u, v) not in T is decreased (moreover, u is also an ancestor of v in T).

Describe an $O(V)$ algorithm **(basic idea only)** for finding the minimum spanning tree in the modified graph (assume that $w(x, y)$ returns the weight of the edge (x, y) in $\Theta(1)$ time).

As u is an ancestor of v in T, we have to see the edges on the path from u to v in T and find the edge with maximum weight on that path.

> [**Rubric**] **1 Mark** to show that we need to find an edge with maximum weight on the path from u to v in T

As we have only π information of the vertices available at hand, we will traverse this path in reverse order; i.e. we start at $(v, v.\pi)$ then gor to $(v.\pi, v.\pi.\pi)$ and so on till we reach some vertex v' such that $v'.\pi = u$, the last edge in this sequence would be (v', u).

This whole process can be done in $O(V)$ time as there are at most $|V| - 1$ edges on this path and the weight of each edge can be obtained in $\Theta(1)$ time.

> [**Rubric**] **1 Mark** to show that this can be done in $O(V)$ time

Let (x, y) be an edge with maximum weight on that path.

If weight of (x, y) is not greater than the new (reduced) weight of (u, v), then we do not have to do anything; T will remain an MST for the modified graph.

> [**Rubric**] **1 Mark** to show that MST will remain unchanged if $w(u, v) \geq w(x, y)$

Otherwise, (i.e. $(u, v) < w(x, y)$), we remove (x, y) from T, this will break T into two components. Adding (u, v) reconnects them and gives us a new MST for the modified graph.

> [**Rubric**] **2 Mark** to show how to modify the MST if $w(u, v) < w(x, y)$

This can be done by setting the π value of each vertex in the path form $v.\pi$ to y to its child vertex, e.g. let $a.\pi = b$, then we set $b.\pi = a$. As there are at most $|V|$ vertices on the path from $v.\pi$ to y, this whole process takes $O(V)$ time. At the end, we make v child of u (i.e we set $v.\pi = u$) which can be done in $\Theta(1)$ time.

> [**Rubric**] **2 Mark** to show that this can be done in $O(V)$ time

Question 3: **[3+7] Marks**

In a certain course there is an MCQ exam with help sheet. This help sheet contains **L** lines. There are *n* topics included in the exam. For each topic $\boldsymbol{i}$, it is known that there will be **Q[$\boldsymbol{i}$]** questions in the exam and the marks for these questions will be guaranteed if the information about the topic $\boldsymbol{i}$ is included in the help sheet. For any topic $\boldsymbol{i}$, there are **H[$\boldsymbol{i}$]** lines available and the student can either write all of these **H[$\boldsymbol{i}$]** lines in their help sheet or do not write anything about that topic. The task at hand is to select the topics whose lines you want to include in the help sheet such that the marks guaranteed is maximized.

For this question you are required to design a bottom-up dynamic programming algorithm (iterative algorithm) that calculates the maximum marks guaranteed for the above scenario.

For each topic $\boldsymbol{i}$ ($1 \leq \boldsymbol{i} \leq n$), the number of questions and the number of lines for the help sheet are available at the $\boldsymbol{i^{th}}$ index of arrays **Q** and **H** respectively. Also note that each question carries equal marks.

Basic Idea (in Plain English):

There are two dimensions (parameters) of this problems, namely, # of lines ($\boldsymbol{L}$), and # of topics ($\boldsymbol{n}$). The question is about maximum marks using (up to) n topics and (at most) L lines. So, we would solve this problem in a bottom-up fashion along these two dimensions.

[Rubric] 3 Marks

[Optional] Following recurrence would be used to solve the problem:

$$Marks[i,j] = \max(Marks[i-1,j], Marks[i-1, j-H[i]] + Q[i])$$

Pseudocode:

```
MaxMarks(L,H,Q)
1    n = H.length
2    let Marks[0…n,0…L] be a new array
3    for i = 0 to n
4         Marks[i,0] = 0
5    for j = 1 to L
6         Marks[0,j] = 0
7    for i = 1 to n
8         for j = 1 to L
9              if H[i]>j
10                  Marks[i,j]=Marks[i-1,j]
11             else Marks[i,j] = max(Marks[i-1,j],
                                     Marks[i-1,j-H[i]]+Q[i])
12   return Marks[n,L]
```

[Rubric] 0.5 Marks each

- Line 2
- Line 3-4
- Line 5-6
- Line 7-8

[Rubric] 1 Mark

- Line 9-10

[Rubric] 3 Mark

- Line 11

[Rubric] 1 Mark

- Line 12

Question 4: **[3+7] Marks**

Mathew is a passionate mountaineer. This season, he climbed $\mathbf{n}$ mountains labelled as mount_1, mount_2, …, mount_n. He started his climbing adventure at mount_start ($\mathbf{1} \leq \mathbf{start} \leq \mathbf{n}$) and finished it at mount_end ($\mathbf{1} \leq \mathbf{end} \leq \mathbf{n}$).

Write an algorithm that prints the mountain labels in the order in which the mountains were climbed by Mathew. Following information would be available to your algorithm as input:

The variables **start** (the mountain from where he started his adventure) and **end** (the mountain where he finished his adventure), and the 2-D array $\mathbf{Highest}[\mathbf{1} \ldots \mathbf{n}, \mathbf{1} \ldots \mathbf{n}]$.

- $\mathbf{Highest}[\boldsymbol{i}, \boldsymbol{j}] = \boldsymbol{k}$ if mount_k is the highest mountain climbed by Mathew while going from mount_i to mount_j.
- $\mathbf{Highest}[\boldsymbol{i}, \boldsymbol{j}] = \mathbf{NIL}$ if
  - ❖ Either, he climbed mount_j immediately after mount_i (hence, no intermediate mountain to climb)
  - ❖ Or, he climbed mount_j before mount_i (hence going from mount_i to mount_j is not a possibility)

Your algorithm MUST finish in $O(n)$ time.

Basic Idea in Plain English:

We first print the starting mountain, then we **recursively** print all the intermediate mountains, finally we print the ending mountain.

[**Rubric**] **3 Marks** if it is mentioned that the intermediate mountains are printed *recursively***.**
[**Rubric**] **1 Mark** otherwise (provided there is a mention of printing the starting mountain and ending mountain)

Pseudocode:

```
PrintTour(start, end, Highest)
     print “mount_”start
     PrintIntermediate(start, end, Highest)
     print “mount_”end
```

[Rubric] 2 Marks

```
PrintIntermediate(i, j, Highest)
     if Highest[i,j]≠NIL
          PrintIntermediate(i, Highest[i,j], Highest)
          print “mount_”Highest[i,j]
          PrintIntermediate(Highest[i,j], j, Highest)
```

[Rubric] 5 Marks

Question 5: **[4+3+3] Marks**

a) In the radix sort algorithm, what possibly may go wrong if the sorting algorithm used by the radix sort as subroutine is not a stable sorting algorithm? Explain with the help of an example. (**No partial marks for this part**)

It may not correctly sort the input array. Let $A = \langle 55,53 \rangle$, then after the first pass, we have $A[1] = 53$ and $A[2] = 55$ (as $3 < 5$). However, in the second pass there would be a tie between two occurrences of 5. If the subroutine is not a stable sorting algorithm, it may pick 5 corresponding to 55 first and the final output would be $A = \langle 55,53 \rangle$, which is not correct.

b) For the following algorithm, write down the recursive expression for $T(n)$. The first call is ALGO(A, 1, n), where A is an array of size n.

$ALGO(A, i, j)$

```
1  ret = i
2  if (i ≤ j) then
3     ret = ALGO(A, i, (i + j)/2)
4     k = (i + j)/2
5     while (k ≤ j)
6        ret = ALGO(A, k, j) + ret − k
7        k = k + 1
8  return ret
```

2 Marks

$$T(n) = \underbrace{T\left(\frac{n}{2}\right)}_{\text{1 Mark}} + \sum_{m=1}^{\frac{n}{2}} T(m) + \Theta(n)$$

c)

i. If $T(n) = 8\,T(n/3) + \Theta(n^2)$, then $T(n) = \Theta(n^2)$.

ii. If $T(n) = 9\,T(n/3) + \Theta(n^2)$, then $T(n) = \Theta(n^2 \lg n)$.

iii. If $T(n) = 10\,T(n/3) + \Theta(n^2)$, then $T(n) = \Theta(n^{\log_3 10})$.

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:	180 Minutes	Total Marks:	61
Paper Date:	24-June-22	Section:	ALL
Exam:	Final Exam	Page(s):	10
Name		Roll Number	

Instruction/Notes: Ample space is provided for rough work; NO EXTRA sheets will be provided.

Question	1-5	6-8	9	10	11	Total
Marks	/16	/15	/10	/10	/10	/61

Q1) Devise the recurrence relation for the following recursive function and then solve it for calculating time complexity in Big O or Theta notation of this algorithm. Show complete working. **[5 Marks]**

```
Mystery(n){
        if(n==0)
                return n;
        else
                int x=0;
                for(i=1;i<n-1;i++)
                        x=x+i;
                return x+Mystery(n-1);
}
```

Solution:

Recurrence: $T(n) = T(n-1) + O(n)$

Time complexity = $O(n^2)$

Q2) Which of the following sorting algorithms in its typical implementation gives best performance (in terms of time complexity) when applied on an array which is sorted or almost sorted (maximum 1 or two elements are misplaced). **[2 Marks]**

a) QuickSort
b) MergeSort
c) Insertion sort

Solution: Insertions sort O(n)

Q3) In an unweighted, undirected connected graph, the shortest path from a node S to every other node is computed most efficiently, in terms of time complexity by: **[2 Marks]**

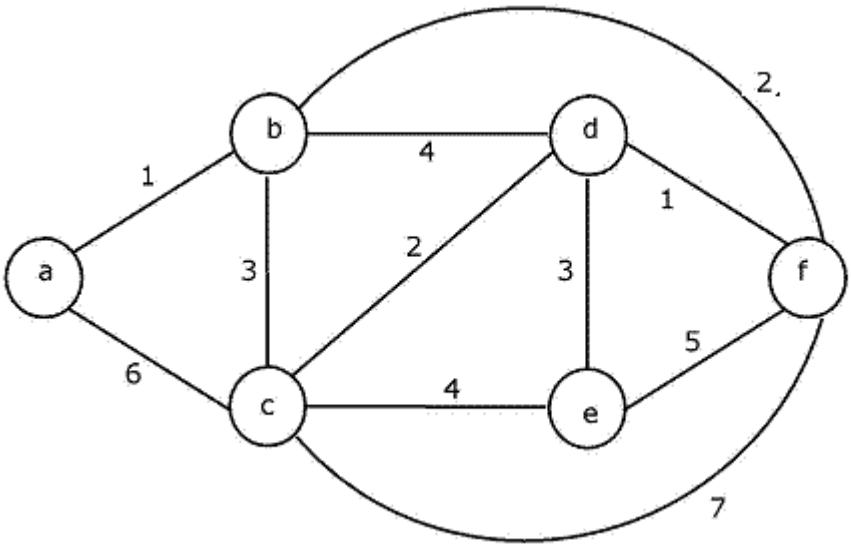
I. Dijkstra's algorithm starting from S
II. Floyd Warshall's algorithm

III. Performing a DFS starting from S
IV. Performing a BFS starting from S

Solution

Performing a BFS starting from S

Q4) Consider the following graph: [2 Marks]



Which one of the following cannot be the sequence of edges added, in that order, to a minimum spanning tree using Kruskal's algorithm?

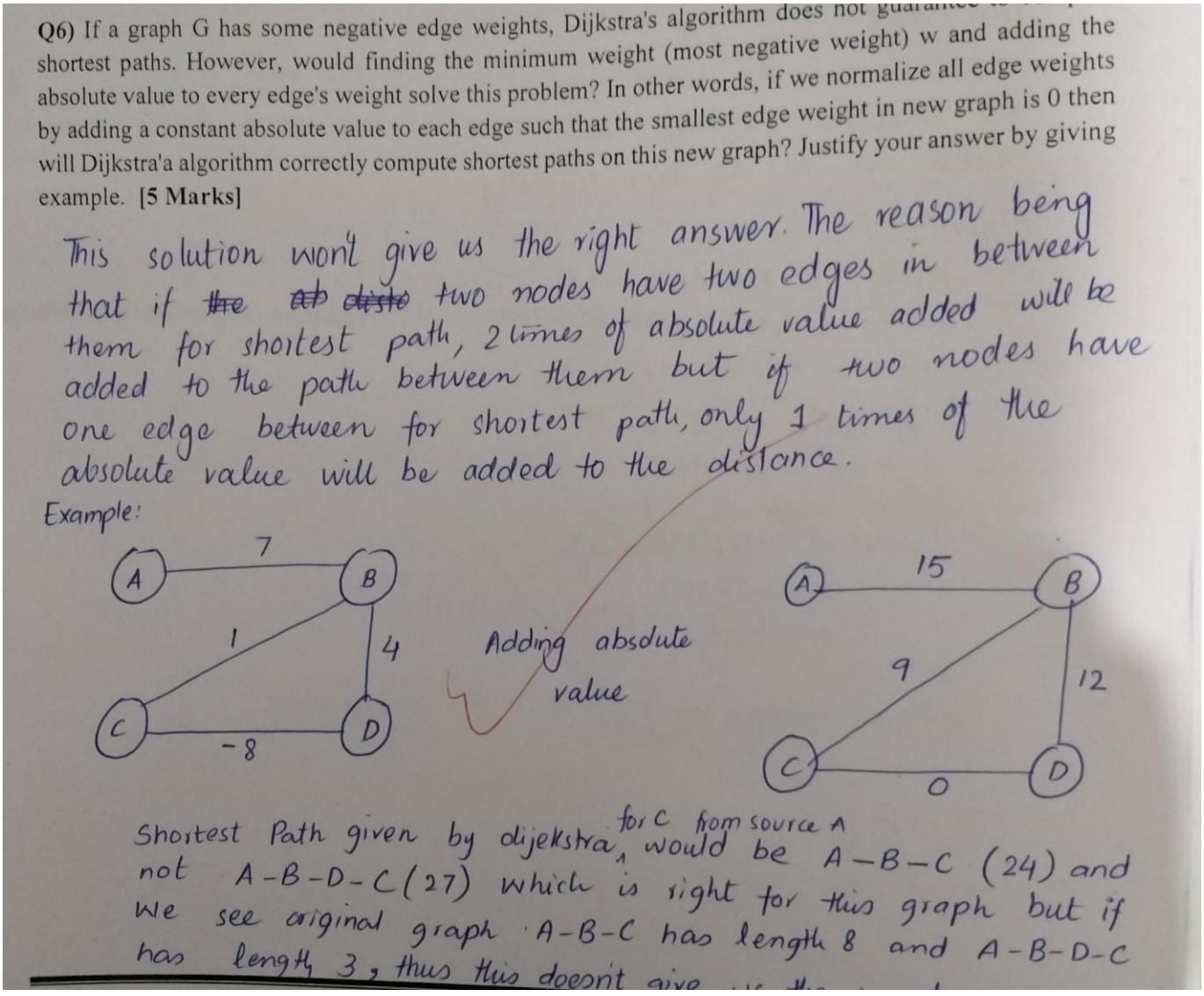
I. (a—b),(d—f),(b—f),(d—c),(d—e)
II. (a—b),(d—f),(d—c),(b—f),(d—e)
III. (d—f),(a—b),(d—c),(b—f),(d—e)
IV. (d—f),(a—b),(b—f),(d—e),(d—c)

Q5) Show how to sort n integers in the range 0 to $n^4 - 1$ in O(n) time. **[5 Marks]**
Solution: Use radix sort and use n as base of the numbers. Radix sort takes $O((n+b) * \log_b(k))$. Here $b = n$, $k = n^4$

Q6) If a graph G has some negative edge weights, Dijkstra's algorithm does not guarantee to compute shortest paths. However, would finding the minimum weight (most negative weight) w and adding the absolute value to every edge's weight solve this problem? In other words, if we normalize all edge weights by adding a constant absolute value to each edge such that the smallest edge weight in new graph is 0 then will Dijkstra'a algorithm correctly compute shortest paths on this new graph? Justify your answer by giving example. **[5 Marks]**

Solution:

No



Q7) You have an unsorted array, A, of unique numbers. You know its median number in advance. Now you wish to arrange the elements of A in such a way that the numbers $[i]$ and $A[i + 1]$ are successively in the increasing and decreasing order. That is, A[1]< A[2], A[2]> A[3], A[3]< A[4]… and so on. A friend of

yours claims that this is possible in O(n). Is her claim True / False. Justify your answer. If you are using an algorithm discussed in class then just name the algorithm and write the additional pseudocode required for to solve this problem. **[5 Marks]**

Solution 1:

The claim is **True**; following is linear time algorithm to solve the given problem:

```
Let B[1..n] be a new array
i = 1
j=2
for k = 1 to n
        if A[k] ≤ median
                B[i]=A[k]
                i=i+2
        else
                B[j]=A[k]
                j=j+2
return B
```

Solution 2:

Use partition function of Quick sort to partition the array around median in O(n) time.

```
x = 1
For (i = 1 to n/2)
   B[x] = A[i]
   B[x+1] = A[n/2+i]
   x= x+2
```

Q8) Given the alphabets from a text and their frequencies, we want to compress the text using Huffman's code. **[5 Marks]**

Alphabet	a	b	c	d	e	f
Frequency	40	10	8	15	25	2

Draw the binary tree for Huffman Code, and using that tree, write down the binary code of each alphabet.

Alphabet	**a**	**b**	**c**	**d**	**e**	**f**
Code						

Solution

Alphabet	**a**	**b**	**c**	**d**	**e**	**f**
Code	**0**	**1110**	**11111**	**110**	**10**	**11110**

Q9) 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 first cell (0, 0) of the matrix from its last cell (M-1, N-1). We can only move one unit left (Column No - 1), one unit up (Row No - 1), and one unit in top diagonal (Row No - 1, Column No - 1). **[10 Marks]**

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 = 3 -> 3 -> -2 -> 1 -> 7-> -6 -> 4 = 10

Provide a Dynamic Programming solution for it.

a) Provide recurrence for sub-problem

Solution 1

(a) `C[i,j] = D[i,j] + min (C[i+1,j], C[i+1,j+1], C[i,j+1])`

(b)

Let C[0..M-1, 0..N-1] be a new matrix

C[M-1,N-1] = A[M-1,N-1]

for i=M-2 **dowto** 0

C[i,N-1] = A[i,N-1] + C[i+1,N-1]

for j=N-2 **dowto** 0

C[M-1,j] = A[M-1,j] + C[M-1,j+1]

for(i: m-1 to 0)

for (j: n-1 to 0)

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

The final answer will be in C[0][0]

(c) Time Complexity = O(m*n)

Solution 2

for(0 to m)

for (0 to n)

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

The final answer will be in C[M-1][N-1]

b) Provide pseudo code for DP solution

c) Provide time complexity of DP solution

Q10) Suppose you are given a weighted undirected graph G, and its subgraph T. The subgraph T and the graph G are represented in the form of adjacency lists. [10 Marks]

(a) Design an efficient algorithm that decides whether T is **spanning tree** of the given graph or not. Hint: Recall the definition of a spanning tree [7 Marks]

```
if |G.V| ≠ |T.V|
    return FALSE
if |T.E| ≠ |T.V| − 1
    return FALSE
//Run DFS (T) to check connectivit
for each vertex u ∈ T.V
        u.color = WHITE
        u.π = NIL
time = 0
visits = 0
for each vertex u ∈ T.V
        if u.color == WHITE
                DFS-VISIT (T, u)
                visits = visits + 1
        if visits > 1
                return FALSE
return TRUE
```

(b) Design an efficient algorithm that decides whether T is **minimum spanning tree (MST)** of the given graph G or not. You are given the total weight of MST of graph G. [3 Marks]

```
//Let M be the weight of MST
X = 0
for each vertex u ∈ T.V
        for each v ∈ T.Adj[u]

                X = X + w(u, v)
if X/2 == M
        return TRUE
else
        return FALSE
```

Q11) "$x - y$ **Logistics**" is a carrier company which has its own transportation network spread across major cities of the country. Most (i.e. around 95%) of the business of the company is related to the transportation of goods **from** city x **to** city y (hence, the name $x - y$ **Logistics**). Every now and then, the company also receives offers from other transportation companies to use their (i.e. other companies') transportation services between different cities on discounted rates. Sometimes these offers are so lucrative (i.e. the discounted price are very low) that $x - y$ **Logistics** prefer to avail some other company's transportation service instead of using their own.

For any such offers, the company has to respond affirmative *in a very short time* (otherwise, that offer may be taken by some other competitors). However, a hasty decision may also has its consequences. Hence, the company has decided to build a system which tells *in real time* whether a transportation offer from city p to city q is in the benefit of the company or not. For the time being, the company is **only** interested to get this real time information for transportation of goods **from city x to city y**. (x and y are names of actual cities not generic notations)

The above mentioned scenario can be modelled using a **weighted directed graph** $G = (V, E)$; the vertices in this graph represent cities where $x - y$ **Logistics** has its offices and an edge (u, v) indicates that the company has its own transportation available from city u to city v, whereas $w(u, v)$ represents the cost of using company's own transport to carry goods from city u to city v.

Any new transportation offer at discounted rate from city p to city q can be modelled as insertion of a new *temporary* edge from vertex p to vertex q, with $w(p, q)$ being the price of the offer.

(Recall, 95% of the company's business is related to the transportation of goods **from** city x **to** city y. Therefore, for the time being, the company is **only** interested to get this real time information for transportation of goods **from city x to city y**.)

Note1: All the edge weights are positive.

Note2: You must determine whether or not this new offer will reduce the cost of transportation of goods from city x to city y.

Hint: Recall that the shortest path problem exhibits optimal substructure.

(a) Suppose you have already computed the weights of all-pairs-shortest distances and have the information in array D. Using this information, how would you decide in $\Theta(1)$ time whether or not a ***new*** transportation offer from city p to city q will lower the cost of the transportation of goods **from** city x **to** city y. [3 Marks]

(b) Write an algorithm that takes $O((V+E)\lg V)$ preprocessing time and then decides in O(1) whether or not an ***new*** transportation offer from city p to city q will lower the cost of the transportation of goods **from** city x **to** city y. [7 Marks]

[10 Marks]

Solution

(a) Suppose the newly added edge is from u to v with weight w(u,v), then

If(D[x,u] + w(u,v) + D[v,y]) < D[x,y]) then use the new edge.

(b)

1) Run Dijkstra by keep x as source. Save distances from x to any other vertex v in d1[v] $O((V+E)\lg V)$
2) Calculate reverse of the graph G^R in linear time. O(V+E)
3) Run Dijkstra by keeping y as source in G^R. Save distances from y to any other vertex v for G^R in d2[v] $O((V+E)\lg V)$

Suppose the newly added edge is from u to v with weight w(u,v), then

If(d1[u] + w(u,v) + d2[v]) < d1[y]) then use the new edge.

Design & Analysis of Algorithms - Spring 2013
Final Exam

May 22, 2013 **Time: 3 hrs.** **Marks: 70**

Q1.(**5+ 5**)The following function recursively adds the elements of an array.

i. Write down the recurrence for the running time of this function on an array of size n. Also give the base case.

ii. Solve the recurrence in (i) to find a tight bound on the running time.

```
AddArrayElements(A, left, right)
 //A[1…n] is an array of integers, where n is a power of 7
size    right – left + 1
IF size > = 7
   seventh_part    FLOOR (size / 7)
   sum    0
   FOR i   1 to 7
      sum   sum + AddArrayElements(A, left, left + seventh_part – 1)
      left    left + seventh_part
   END-FOR
ELSE
   sum    0
   FOR i    left to right
      sum    sum + A[i]
   END-FOR
END-IF
return sum
```

Q2. (**5 + 10**)Following questions are related to the sorting algorithms covered in class.

i. A *stable* sorting algorithm leaves equal-key items in the same relative order as in the original permutation. Explain, *by providing code*, what must be done to **ensure** that mergesort is a stable sorting algorithm.

ii. Give the most efficient algorithm to find the union of sets A and B. The sizes of both A and B are the same, say n. The numbers in A and B are from the range [-k, +k] where k < n. The algorithm returns the result in a set C. Please provide a brief explanation in English, then a neat and clear pseudo code, and then an argument for the running time.

Q3.(**15**) Given a simple undirected graph G=(V, E) and an edge *e* in E, design a linear time algorithm to detect whether there is a cycle in G which contains *e* as one of its edges. The algorithm should report if no such cycle exists, otherwise it should print the cycle found (i.e. print the sequence of vertices in that cycle). Please provide a brief explanation in English, then a neat and clear pseudo code, and an argument for the running time.i

Q4.(**5 * 3**) Answer the following questions, and in each case provide a brief explanation for your answer. No marks shall be awarded without a correct explanation.

i. A k-ary tree is a tree in which every node has at most k children. In a k-ary tree with n nodes and height$h$, what is the maximum possible number of leaves?

ii. You are given an array of size n, containing integers in the range $0 \ldots k$; and you are allowed to preprocess this array in whatever manner you choose to in any amount of time, but using only O(n) space. Then, queries of the following type are made: How many integers fall in the range$[a, b]$(both a&b inclusive)? What is the fastest possible time in which you can answer such a query? What preprocessing will be needed in this case. Clearly explain in English.

iii. Write the equation describing the average case running time of the ***quicksort*** algorithm. State the meaning of each term in the equation and why it has been included.

iv. The shortest path tree computed by Dijkstra's algorithm is not *necessarily* an MST. Show an example graph with 4 vertices where the shortest path tree is different from the MST.

v. If we wanted to find the Maximum spanning tree of a graph, rather than the Minimum spanning tree, we could simple negate the weight on each edge and run Kruskal's algorithm, and the output would be a Maximum spanning tree. Yes or No? If yes, argue why your claim is correct. If no, give an example graph (with no more than four vertices) where this strategy will not work.

Q5. [FOR SECTIONS B, C, ONLY](5 + 5 + 5) You are going on a long trip. You start on the road at milepost $a_0 = 0$. Along the way there are *n* hotels, at mileposts $a_1 < a_2 < a_3 \ldots < a_n$, where each a_i is measured from the starting point. The only places where you are allowed to stop are these hotels, but you can choose which of these hotels to stop at. You must stop at the final hotel at a_n, which is your destination.

Ideally, you will like to travel 200 miles per day, but this may not be possible depending upon the spacing between the hotels. If you travel *x* miles a day then the penalty for that day is $(200 - x)^2$. You want to plan the trip so as to minimize the total penalty, that is, the sum of penalties over all travelling days. Your goal is to design a Dynamic Programming algorithm that gives you the optimal hotels to stop at as well as the penalty incurred with those stops.

i. The optimal structure for this problem is defined as below:
P[i] is the minimum possible penalty when the hotel at a_i is taken as destination (where you must stop).
Obviously, P[0] = 0
Give the recurrence which relates P[i] to smaller sub-problems.

ii. Let S be an array of integersof size n which stores the actual sequence of "stops" required for the optimal (minimum) penalties stored in P. Write a small piece of pseudo-code that will compute S[i] (should be on the basis of your recurrence in (i)).

iii. What is the running time of this solution in Big-O notation?

Note: In this problem you must answer part (i) correctly to get credit for parts (ii) and (iii).

Q5. [FOR SECTIONS A, D, ONLY] (15) Devise an algorithm to delete all occurrences of a pattern P (|P|=n) from Text T (|Y|=m) in O(m+n). Please provide a brief explanation in English, then a neat and clear pseudo code, and an argument for the running time.

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:**	**Spring 2019**
Duration:	**180 Minutes**	**Total Marks:**	**90**
Paper Date:	**27-May-19**	**Weight**	**45%**
Section:	**ALL**	**Page(s):**	**10**
Exam:	**Final Exam**		

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 – 7	8	9	10	Total
Marks	/45	/15	/15	/15	/90

Q1) For each of the following statements, state true or false. **[5 Marks]**

a. If G is a simple graph with n vertices and n−1 or more edges, then G is connected. T/**F**

b. If a simple graph G has n vertices and n or more edges, then G contains a cycle. **T**/F

c. Dijkstra's algorithm can find shortest paths in a directed graph with negative weights, but no negative cycles. T/**F**

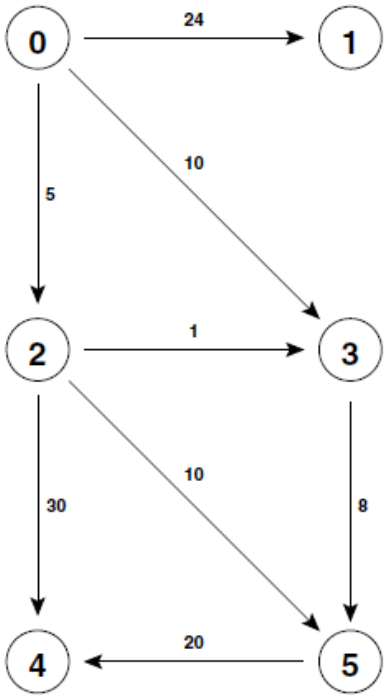
d. Prim's algorithm for computing the MST only work if the weights are positive. T/**F**

e. A graph where every edge weight is unique (there are no two edges with the same weight) has a unique MST. **T**/F

Q2) Simulate Dijkstra's algorithm on the edge-weighted graph below, starting from vertex 0. **[5 Marks]**

Fill in the following table:

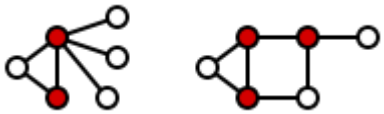
Vertex	Shortest Path from vertex 0	Predecessor (parent) vertex in shortest path
1	24	0
2	5	0
3	6	2
4	34	5
5	14	3



Q3) How many times the partition function will be called in the worst case, during the recursive execution of quick sort if array to be sorted has n elements? Justify your answer. **[5 Marks]**

O(n) times because each element can be a pivot only once.

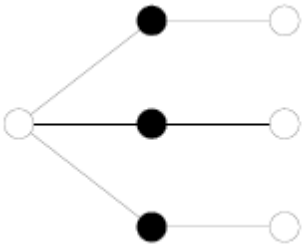
Q4) Let $G = (V, E)$ be an undirected graph. A node cover of G is a subset U of the vertex set V such that every edge in E is incident to at least one vertex in U. A minimum node cover is one with the fewest number of vertices.



shaded vertices shows minimum node cover

Consider the following greedy algorithm for this problem: Select the vertex v of maximum degree (ties may be broken arbitrarily) and include it in the cover. Remove vertex v and all the edges that are incident on v from G repeat the same until G has 0 number of edges. Either prove that this greedy strategy always gives the optimal answer or disproof it using counter example. **[5 Marks]**

In the example below greedy will pick white vertices but optimal solution is black



Q5) You have been given an integer array A of size N. Each element of the array ranges between 1 and 10^5. You need to find the frequency of each distinct element of the array. The elements need to be present in the output in ascending order (sorted). You need to print the value and then frequency of each distinct element. Give high level description of algorithm to solve this problem in O(n) time. **[5 Marks]**

Use count sort

Name: _____ Roll #: _____ Section: _____

Q6) What is the best bound on the running time of the algorithm in the box below? **[5 Marks]**

```
count ← 0
For i ← 1 to n
    For j ← 1 to i
        For k ← 1 to i*i
            count ← count + 1
```

a) $O(nlgn)$
b) $O(n\sqrt{n})$
c) $O(n^3)$
d) $\mathbf{O(n^4)}$
e) $O(n^5)$

Q7) Exams are over! You've rented a car and are ready to set out on a long drive on the Strange Highway. There are n tourist stores on the Strange Highway, numbered 1, 2, …, n and you would want to stop at these and buy some souvenirs (only one souvenir may be bought from a store and all souvenirs everywhere have the same price). You are a greedy shopper and are most interested in maximizing the total discount you get on your shoppings. You know that each store i offers a discount d_i on a souvenir. But it's a strange highway after all. It turns out that you can only buy a souvenir from a store i if you have not bought anything from the previous f_i stores. For example, if $f_6 = 3$ then you can only buy a souvenir from store 6, and get the discount d_6, if you haven't bought anything from stores 3, 4, and 5. All the d_i and f_i are known to you in advance. You have recently learnt the DP technique in your algorithms course and wish to apply it here in order to maximize your total discount under the given constraints. After some brain-storming, you think you can define the optimal sub-problem in one of the following two ways:

A. $D[i]$ = max total discount when store i is the last store where you buy a souvenir.
B. $D[i]$ = max total discount for the trip till store i whether buying at store i or not.

Which of the following recurrences correctly computes $D[i]$ for option (A) above and which one does it for (B). Write the correct option number in the table below. **[10 Marks]**

(A)	(v)
(B)	(iii)

(i) $D[i] = D[i - f_i - 1] + d_i$
(ii) $D[i] = \max(D[i-1] + d_i, D[i-1])$
(iii) $D[i] = \max(D[i - f_i - 1] + d_i, D[i-1])$
(iv) $D[i] = \max\limits_{1<k<(i-f_i)} \{D[k] + d_i, D[i - f_i - 1]\}$
(v) $D[i] = \max\limits_{1<k<(i-f_i)} \{D[k] + d_i\}$
(vi) $D[i] = \text{sum}\limits_{1<k<(i-f_i)} \{d_k\}$

Having identified the recurrences, which one do you think results in a better DP algorithm in terms of time complexity? **(A)** or **(B)**, Justify your answer. **[5 Marks]**

Answer: (B)

Q8) You are given a weighted undirected graph G(V, E) and its MST T(V, E'). Now suppose an edge (a, b) ϵ E' has been deleted from the graph. You need to devise an algorithm to update the MST after deletion of (a, b).

Describe in words an algorithm for updating the MST of a graph when an edge (a, b) is deleted from the MST (and the underlying graph). It's time complexity must be better than running an MST algorithm from scratch. State and explain the time complexity of your algorithm.

You can assume that all edge weights are distinct, that the graph has E edges and V vertices after the deletion, that the graph is still connected after the deletion of the edge, and that your graph and your MST are represented using adjacency lists. [**6 Marks**]

The following algorithm has time complexity O(E). Run DFS twice on what remains of the MST, once starting on a and once starting on b to determine the two connected components A and B, which you store in two HashSets. This takes O(E). Then iterate through all edges going out from the elements of A. If an edge leads to an element of B (can be check in O(1) using HashSet), check whether it is the cheapest edge so far. If it is cheaper than any other edge so far, store it. So, finding the cheapest edge can also be done in O(E). In the end, add the cheapest edge to the MST.

Name: _____ Roll #: _____ Section: _____

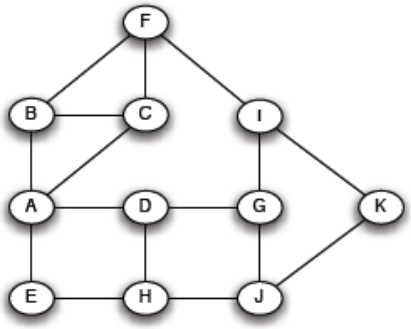
Give complete pseudo code of the algorithm **[7 Marks]**

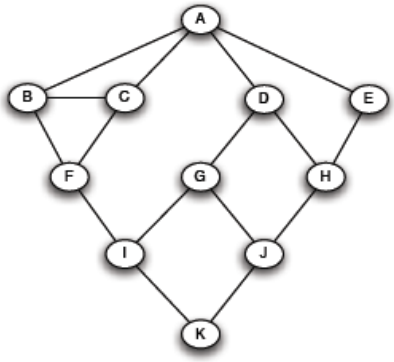
What is the running time of your algorithms? **[2 Marks]**

Name: _____ Roll #: _____ Section: _____

Q9) Recall that BFS can be used to solve the problem of single source shortest path for unweighted graph G(V, E). Also BFS only finds one shortest paths between s and x, where s is the source vertex and x is any other vertex of the graph. But shortest path may not be unique. Suppose we want to count the number of distinct shortest paths between s and x for each x, where xϵV. For example in the graph given below if A is the source vertex then the number of distinct shortest paths for each pair are:

From A to A: 1
From A to B: 1
From A to C: 1
From A to D: 1
From A to E: 1
From A to F: 2
From A to G: 1
From A to H: 2
From A to I: 3
From A to J: 3
From A to K: 6





You are required to design a linear time algorithm that counts the number of distinct shortest paths between a given source vertex s and all the other vertices in the graph. You can assume that graph is represented using adjacency list representation.

Hint: you can use the tree structure of BFS to count the distinct shortest paths.

Give detailed explanation of your algorithm **[6 Marks]**

Use the following idea: Start BFS with the start node but instead of saving only one vertex as parent, store list of parents if more than one nodes will give the same distance. Number of unique shortest paths from s to s is 1. For all other vertices, number of distinct shortest paths are sum of distinct shortest paths of their parents.

Name: _____ Roll #: _____ Section: _____

Give pseudo code of your algorithm **[7 Marks]**

```
Modified_BFS(G(V,E), s)

        distinct_paths[1..v]
        color[1..v]
        pi[1..v] //each element is a list
        dist[1..v]
        for all v in V
                distinct_paths[v]=0
                color[v]=white
                dist[v]=inf
                pi[v]=null

        color[s]=grey
        dist[s]=0
        distinct_path[v]=1
        make FIFO queue Q
        Q.insert(s)
        while(Q is not empty)
                u = Q.remove()
                for each v adj to u
                        if(color[v]==white)
                                color[v]=grey
                                dist[v] = dist[u]+1
                                pi[v].insert(u)
                                distict_path[v] += distint_path[u]
                                Q.insert(v)
                        else
                                if(dist[v] == dist[u+1])
                                        pi[v].insert(u)
                                        distict_path[v] += distint_path[u]

                color[u]=black
```

Argue why running time is linear **[2 Marks]**

Time complexity is same as of BFS, so Linear time

Q10) You're given an n x n matrix, M, of integers. The integers in each row and each column are unique and sorted. Following is an example of a 4x4 matrix with this property:

-1	6	8	11
9	13	17	18
11	16	19	21
15	17	20	23

Sample 4x4 matrix, M

Your goal is to write a search algorithm to find a key x, if it exists, in the matrix M. You need to write two versions of the algorithm.

(i) [5 points] Write a search algorithm with running time $O(n\lg n)$. Write your algorithm as a list of clearly specified pseudo-code steps.

//simple row by row binary search…

Name: _____ Roll #: _____ Section: _____

Write a search algorithm with running time O(n). Write your algorithm as a list of clearly specified pseudo-code steps. **[7 Marks]**
Hint: Use divide and conquer strategy

```
bool findKey(int M[][NUM_DIMS], int rtl, int ctl, int rbr, int cbr, int x, int & r, int & c){
    //performs binary search on rows
    if(rtl<=rbr){
        int rmid=(rtl+rbr)/2;

        //scan middle row, this is O(n)
        bool found=scanrowforkey(M, rmid, ctl, cbr, x, c);

        if(found){
            r=rmid;//(r, c) contain the key coordinates
            return true;
        }else{
            //split in O(1), students can give simple pseudo-code for this function
            int rtl1, ctl1, rbr1,cbr1, rtl2, ctl2, rbr2,cbr2;
            split(rtl,ctl,rbr,cbr,rtl1,ctl1,rbr1,cbr1,rtl2,ctl2,rbr2,cbr2,rmid,c);

            return (findKey(M,rtl2,ctl2,rbr2,cbr2,x,r,c) || findKey(M,rtl1,ctl1,rbr1,cbr1,x,r,c));
        }
    }else return false;
}
```

//basic idea

--- scan the middle row for x
--- if x is found return that (r, c)
--- if c is the column of the first element in the middle row bigger than x, split the matrix into two sub-matrices, (matrices to the north-east and south-west of c) and make two recursive calls
> The total size of these two recursive calls would be half the size of the original call

For an $N \times N$ matrix,
$T(N) = T\left(\frac{N}{a}\right) + T\left(\frac{N}{b}\right) + O(N), where, \frac{1}{a} + \frac{1}{b} = \frac{1}{2}$, this is $O(N)$

Argue why the running time is $O(n)$ **[3 Marks]**

National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	**Design and Analysis of Algorithms**	**Course Code:**	**CS302**
	Program:	**BS Computer Science**	**Semester:**	**Spring 2018**
	Duration:	**180 Minutes**	**Total Marks:**	**80**
	Paper Date:	**25 May 2018**	**Weight**	**50**
	Section:	**ALL**	**Page(s):**	**10**
	Exam Type:	**Final**		

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

Instruction/ Notes: **Attempt the examination on this question paper**. *You can use extra sheets for rough work, do not attach extra sheets with this paper. Do not fill the table titled Question/marks.*

Question	1 to 9	10	11	12	13/14	Total
Marks	/45	/10	/10	10/	/5	/80

Q1) Write the tightest asymptotic upper bound for worst-case running time of following algorithms by selecting running times from following list. For sorting algorithms, n is the number of input elements. For graph algorithms, the number of vertices is n, and the number of edges is Θ(n). You need not justify your answers. Some running times may be used multiple times or not at all. **[5 Marks]**

O(lg n) O(n) $O(n^2)$ $O(n^3)$ O(n lg n) $O(n^2 \lg n)$ $O(n^3 \lg n)$

a. Insertion sort $O(n^2)$.

b. Heapsort O(n lgn) .

c. BUILD-HEAP O(n) .

d. Topological Sort O(n) .

e. Bellman-Ford $O(n^2)$

f. Depth-first search O(n)

g. Floyd-Warshall $O(n^3)$

h. Prim's O(nlgn)

Q2) Give the order of growth of the running time for the following function. **[5 Marks]**

```
public static int f6(int N) {
   if (N == 0) return 1;
   return f6(N-1) + f6(N-1);
}
```

$T(n) = 2T(n-1)+O(1)=O(2^n)$

Q3) You are given a function KSamall that returns the kth smallest element (location of the kth smallest element) of array A. Suppose that KSmall works in O(n) in the worst case where n is the size of array A. Can you write Quick Sort Algorithm using KSmall such that it runs in worst case $O(n \log n)$ time on an n-element sequence. Briefly describe your algorithm. **[5 Marks]**

Use Ksmall to find the median of the array in linear time and use median as pivot. This way quick sort will run in O(nlgn) in the worst case.

Q4) Suppose we are given a set of activities/tasks specified by pairs of the start times and finish times as *T* = *{*(1, 2), (1, 3), (1, 4), (2, 5), (3, 7), (4, 9), (5, 6), (6, 8), (7, 9)*}*. Solve the activity selection problem for this set of tasks. Select maximum set of activities that can be scheduled without overlap. **[5 Marks]**

a) How many activities are selected?

4

b) Which activities are selected?

(1,2),(2,5),(5,6),(6,8)

Q5) Consider two strings: *X* = "adafd", *Y* = "adabd".

a) Show the longest common subsequence **table**, *L*, for strings X and Y. **[3 Marks]**

X/Y		a	d	a	b	D
	0	0	0	0	0	0
A	0	1	1	1	1	1
D	0	1	2	2	2	2
A	0	1	2	3	3	3
F	0	1	2	3	3	3
D	0	1	2	3	3	4

b) What is a longest common subsequence between these strings? **[2 Marks]**

adad

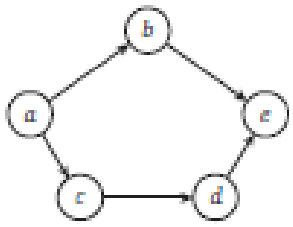
Q6) How can we modify the dynamic programming algorithm from simply computing the maximum benefit/value for the 0-1 knapsack problem to selecting the items that gives specified benefit (if possible)? Write down the changed recurrence relation for this problem. **[5 Marks]**

Let S be the specified benefit

KS(n,W) = KS(n-1,W)==S or KS(n-1,W-w_n)+v_n ==S

by anonymous koala

Q7) Consider the directed acyclic graph *G* in the following Figure. How many topological sorted orderings does it have? Write all different topological sorted orderings. **[5 Marks]**

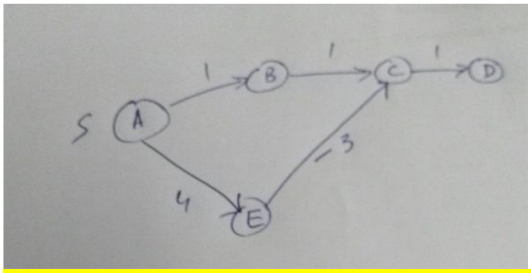


A,b,c,d,e

A,c,b,d,e

A,c,d,b,e

Q8) Give an example of a weighted directed graph, *G*, with negative-weight edges, but no negative-weight cycle, such that Dijkstra's algorithm incorrectly computes the shortest-path distances from some start vertex *v*. Mention the vertex for which Dijkstra will compute incorrect distance. **[5 Marks]**



For vertex D Dijkstra will compute incorrect distance i.e. 3 but shortest distance is 2

Q9) Suppose *G* is an undirected, connected, weighted graph such that the edges in *G* have distinct edge weights, which may be positive or negative. Will minimum spanning tree for *G* be changed if we square all the edge weights in *G*, that is, *G* may have a different set of edges in its minimum spanning tree if we replace the weight, *w*, of each edge *e* in *G*, with w^2. Justify your answer. **[5 Marks]**

No it will be changed as –ve weight become large positive weights. So -9 would become 81 and that may not be selected as a light edge.

Q10) Suppose you are given a connected weighted undirected graph, *G*, with |*V*| vertices and |E| edges, such that the weight of each edge in *G* is an integer in the interval [1, *c*], for a fixed constant $c > 0$. Show how to solve the single-source shortest paths problem, for any given vertex *v*, in *G*, in time $O(|V| + |E|)$.

Hint: Think about how to exploit the fact that the distance from *v* to any other vertex in *G* can be at most $O(cV) = O(V)$. Lesser credit will be awarded for less efficient solutions. **[10 Marks]?**

a) Briefly describe your algorithm for solving this problem. [3 Marks]

Convert each edges (u,v) with weight w to w edges of weight 1 and w-1 additional vertices between u and v. This way the graph will be changed to an undirected graph and the set of vertices and set of edges in the changed graph will be O(V) and O(E) respectively. Now by applying BFS we can compute the shortest paths in linear time.

b) Write pseudocode of your algorithm ? [6 Marks]

1. For each edge (u,v) of weight w add w-1 dummy vertices v1,v2,v3,.., $v_{w-1}$and add eges as follows (u,v1),(v1,v2),…,(v_{w-1},v)
2. Now apply BFS to solve single source shortest path on changed graph.
3. The shortest distance between original nodes of the graph will be same as the one in changed graph

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

O(V+E)

Q11) Let A and B be two sequences of n integers each, in the range $[1, n^4]$. Given an integer x, describe an $O(n)$-time algorithm for determining if there is an integer a in A and an integer b in B such that $x = a + b$. Lesser credit will be awarded for less efficient solutions. **[10 Marks]**

a) Briefly describe your algorithm for solving this problem. [3 Marks]

First sort both A and B using radix sort. Then compare the sum of first element of A and last element of B with x. If sum is greater than x then check compare sum of first element of A and second last element of B with x. If sum is smaller than x then compare sum of second element of A and last element of B with x. If sum is equal to x return the first and last element. And so on

b) Write pseudocode of your algorithm [6 Marks]

```
1. Sort the array A and B using radix sort with b= 4lgn
2. i=1, j=n
3. While(i<j)
4.         If(A[i]+B[j] == x)
5.                 Return true
6.         If(A[i]+B[j] > x)
7.                 j—
8.         If(A[i]+B[j] < x)
9.                 i++
```

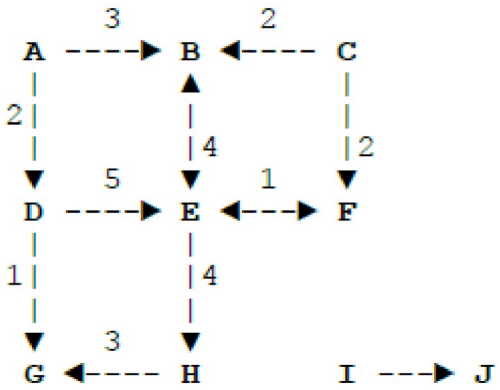
c) What is time complexity of your algorithm [1 Mark]

O(n)

Q12) Write a method popular that accepts a Graph G (G is represented as adjacency list) as a parameter and returns a Set of vertices in that graph that are “popular”. Assume that the graph represents users on a social network such as Facebook. A vertex represents a user, and a directed edge from A to B represents the fact that user A “likes” user B. The weight of the edge represents how much user A “likes” user B. A user v is “popular” if all of the following conditions are met:

- At least two other users “like” v
- More users “like” v than v “likes” other users. *(More arrows coming in than going out)*
- The combined weight of all "likes" toward v is more than the combined outbound weight of all the edges to other users that v "likes". (More total edge weight coming in than going out.)

For example, in the graph below, vertex B is “popular” because vertices A, C and E “like” him with a combined weight of 3+2+4 =9, while he “likes” only vertex E with a weight of 4. For this particular example graph, your method would return the set [B, F, G]. You may assume that the graph and its vertices are not null. **[10 Marks]**



a) Briefly describe your algorithm for solving this problem. [3 Marks]

Compute the indegree and out degree of each vertex. Also compute the sum of incoming edge weights and out going edge weight of each vertex and check condition for each vertex.

b) Write pseudocode of your algorithm [6 Marks]

```
1.  Make four arrays Outdegree, Indegree, Outweight and Inweight of size V and initialize them
    to 0
2.  S = null
3.  For each vertex v in G
4.      For each neighbor u of v
5.              Outdegree[v]++
6.              Indegree[u]++
7.              Outweight[v] += w(v,u)
8.              Inweight[u]+= w(v,u)
9.  For each vertex v
10.     If(Indegree[v]>2 and  Indegree[v]>Outdegree[v] and Inweight[v]>Outweight[v])
11.         Add v to S
12. Return S
```

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

O(V+E)

Q13 is Only for Section A and B

Q13) Let G =(V,E) be a connected, undirected graph and assume all edge weights are distinct. Consider a cycle v1,v2,...,vk,vk+1≥ in G, where vk+1 = v1, and let (vi,vi+1) be the edge in the cycle with the largest edge weight. Prove that (vi,vi+1) does *not* belong to the minimum spanning tree T of G.

Hint (Use Lemmas used in proof of Prim's algorithm) **[5 Marks]**

Q14 is only for Section C, D and E

Q14) a) If the set of stack operations included a MULTIPUSH operation, which pushes k items onto the stack, along with push, pop and multipop. Would the O(n) bound on the amortized cost of stack operations continue to hold? Justify your answer. **[3 Marks]**

No it will not be same because the worst sequence of operations would be multipush, multipop,multipush,multipop … In this case amortized cost would be O(kn) for n operations.

b) In B-tree, t represents its minimum degree and we say that t must be greater than 1. Why don't we allow a minimum degree of t = 1? **[2 Marks]**

Because if t-1 then it will be linked list and minimum number of nodes will be equal to zero