AUTUMN END SEMESTER EXAMINATION-2015

Design & Analysis of Algorithms

[CS-3001/CS-502]

Full Marks: 60 **Time: 3 Hours**

Answer any six questions including question No.1 which is compulsory.
The figures in the margin indicate full marks.
Candidates are required to give their answers in their own words as far as practicable and all parts of a question should be answered at one place only.

DAA SAMPLE ANSWERS & EVALUATION SCHEME

Q1 Answer the following questions: **(2 x 10)**

a) What is the time complexity of the following recursive algorithm?

```
rec (n)
{
   if  n=1   return 1
    else
        return rec(n-1) + n
}
```

Answer:
Time Complexity: O(n)

Explanation
T(n) = T(n-1) + 1 with T(1)=1
Solving this recurrence, finally
$T(n) = 1 + 1 + 1 + \ldots n\ times$
= O(n)

Scheme:
- Only answer : 2 Marks
- Showing correct recurrence, but answer is wrong: 1 Mark

b) Justify the following whether true or false.

I. **$n! = O(n^n)$**

II. **$2^{2n} = O(2^n)$**

Answer:
I. $n! = O(n^n)$ (TRUE)
II. $2^{2n} = O(2^n)$ (FALSE)

Scheme:
- Answer with justification: : 2 Marks
- Only answer: 1 Mark

Explanation
For-I
1 x 2 x 3 x ……x n $\leqslant$ n x n x n x …… x n
⇨ $n! \leqslant n^n$
⇨ $n! \leqslant 1 \times n^n$
⇨ $n! = O(n^n)$ with $c=1, n_0=1$
For-II
Let $2^{2n} = O(2^n)$ TRUE

So, $2^{2n} \leqslant c\ 2^n$

⇨ $2^n \leqslant c$ This is False, So the assumption is FALSE.

OR

Let f(n) and g(n) be two functions such that $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ exists, then a function f(n) = O(g(n)) if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$ where $0 \leq c < \infty$

Here, $\lim_{n\to\infty} (2^{2n}/2^n) = 2^n \neq c$, So FALSE

c) The solution to the recurrence relation: $T(n)=7T(n/2)+n^2$ is

i) $\theta(n^2)$ ii) $\theta(n^{log\ 7})$ iii) $\theta(n \log n)$ iv) $\theta(n^2 \log n)$

Answer:

ii)

Scheme:

- Only answer : 2 Marks
- Explanation with wrong answer: 1 Mark

Explanation

$T(n)=7T(n/2)+n^2$

Here, a=7, b= 2 and $f(n)=n^2$.

Thus, $n^{log_b a}=n^{log_2 7}=n^{2.80}$

Comparing $n^{log_b a}$ with f(n), $n^{log_b a}$ is found asymptotically larger, so case-1 of master theorem will be applied.

We know $2.80 < n^{log_2 7} < 2.81$, therefore $f(n)= O(n^{log_2 7 - \varepsilon})$ for $\varepsilon=0.8$. Using Master Theorem $T(n)=\theta(n^{log_2 7})$

d) Write down the nature of dataset such that insertion sort and quick sort will have same run time behavior.

Answer:

If the data are sorted in reverse (descending or ascending) order, both insertion and quick sort have the same run time behavior that both have the worst case time complexity: $O(n^2)$

Scheme:

- Only answer : 2 Marks

e) Show that an n-element heap has height $\lfloor \log n \rfloor$.

Answer:

- Since a heap is an almost-complete binary tree (complete at all levels except possibly the lowest), it has at most $2^{h+1} - 1$ elements (if it is complete) and at least $2^h - 1 + 1 = 2^h$ elements (if the lowest level has just 1 element and the other levels are complete).
- Given an n-element heap of height h, we know from the above that
  $2^h \leq n \leq 2^{h+1} - 1 < 2^{h+1}$
- Thus, $h \leq \log n < h + 1$. Since h is an integer, $h = \lfloor \log n \rfloor$. (by definition of $\lfloor\ \rfloor$.).

Scheme:

- Correct proof : 2 Marks
- Proving by taking an example: 1.5 Marks

f) What do you mean by principle of optimality? Discuss with a suitable

example.

Answer:

- This principle states that in an optimal sequence of decisions or choices, each subsequence must also be optimal.
- A problem is said to satisfy the Principle of Optimality if the sub-solutions of an optimal solution of the problem are themselves optimal solutions for their sub-problems.
- For example, the shortest path problem satisfies the Principle of Optimality, this is because if $a,x_1,x_2,...,x_n,b$ is a shortest path from node a to node b in a graph, then the portion of x_i to x_j on that path is a shortest path from x_i to x_j.
- Also in matrix chain multiplication problem, not only the value we are interested in is optimal but all the other entries in the table are also represent optimal.

Scheme:

- Defining principle of optimality: 1.5 Marks
- Example: 0.5 Mark

g) Define Big-Oh Notation.

Answer:

Definition-1: For any two functions f(n) and g(n), which are non-negative for all $n \geq 0$, f(n) is said to be g(n), f(n) = O(g(n)), if there exists two positive constants c and n_0 such that

$0 \leq f(n) \leq c\ g(n)$ for all $n \geq n_0$

OR

Definition-2:

Let f(n) and g(n) be two functions such that $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ exists, then a function f(n) = O(g(n))

if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$ where $0 \leq c < \infty$

Scheme:

- Any correct definition: 2 Marks

h) Match the following pairs

A. Quick Sort Algorithm	**P. GREDY**
B. Bellmanford Algorithm	**Q. Divide and Conquer**
C. Floyd'-Warshall Algorithm	**R. Dynamic Programming**
D. N-Queen Problem	**S. Backtracking**

Answer:

A-Q ,B-P ,C-R ,D-S

Scheme:

- Each correct matching : 0.5 Marks

i) What is the difference between fractional knapsack and 0/1 knapsack problem? Which one gives more profit?

Answer:

Fractional knapsack gives more profit.

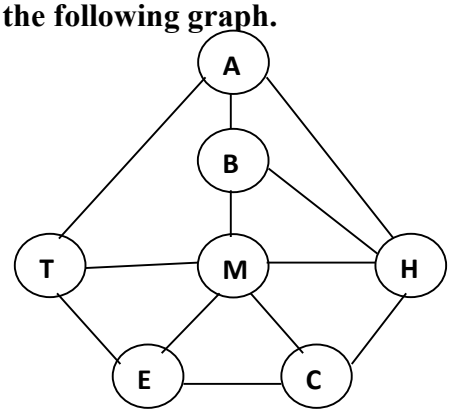
Fractional Knapsack	0/1 Knapsack
• Items are divisible. • Exhibit greedy choice property, So **greedy algorithm exists.**	• Item are indivisible. • Exhibit no greedy choice property, So no greedy algorithm

Scheme:

- Difference : 1.5 Marks
- Choosing correct answer that gives max. profit: 0.5 Mark

• It also exhibit optimal substructure property.	exists. • It exhibit optimal substructure property, so only **dynamic algorithm exists.**

j) Consider the following graph.



Among the following sequences, which are possible breadth first traversals of the above graph if the first symbol of each sequence is considered as start vertex.

i) MBTHAEC ii) HMBACET iii) HABMCET iv) TAMEBHC

Answer:
ii) and iv)

Scheme:
- All possible answer : 2 Marks

Q2 a) Insertion sort can be expressed as a recursive procedure as follows. In order to sort A[1…n], we recursively sort A[1..n-1] and then insert A[n] into the sorted array A[1..n-1]. Write the procedure and a recurrence for the running time of this recursive version of insertion sort. **(4)**

Scheme:
- Recursive insertion sort algorithm: 3 Marks
- Showing correct recurrence: 1 Mark

Answer:

Recursive Insertion Sort Algorithm	Time Complexity

INSERTION-SORT-RECUR(A, p, r)
{
if (p<r)
{
INSERTION-SORT-RECUR(A, p, r-1)
INSERT-INORDER(A, p,r-1, A[r])
}
}
INSERT-INORDER(A,p, r, x)
{
if (x >= A[r])
A[r+1] = x
else if (p <r)
{
A[r+1] = A[r]
INSERT-INORDER (A, p, r-1)
}
	• Since it takes Θ(n) time in the worst case to insert A[n] into the sorted array A[1 . . n-1], we get the recurrence
	$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1, \\ T(n-1) + \Theta(n) & \text{if } n > 1 \end{cases}$
	• The solution to this recurrence is $T(n) = \Theta(n^2)$.

```
else
{
   A[r+1] = A[r]
   A[r] = x
}
}
```

b) Solve the following recurrence **(4)**

$$\mathbf{T(n)} = \begin{cases} \mathbf{2T\left(\frac{n}{4}\right) + \sqrt{n}} & \mathbf{if\ n > 1} \\ \mathbf{1} & \mathbf{if\ n = 1} \end{cases}$$

Scheme:

- Explanation with correct answer: 4 Marks
- Correct Explanation with incorrect answer: 3 Marks

Answer:

$$T(n) = \begin{cases} 2T\left(\frac{n}{4}\right) + \sqrt{n} & \text{if } n > 1 \\ 1 & \text{if } n = 1 \end{cases}$$

By applying master theorem

a=2, b=4 f(n) = $\sqrt{n}$

$n^{\log_b a} = n^{\log_4 2} = \sqrt{n}$

Comparing $n^{\log_b a}$ and f(n) , both are same. So case-2 of master theorem can be applied.

$T(n) = \Theta(n^{\log_b a} \log n) = \Theta(\sqrt{n} \log n)$

Q3 a) Find an optimal parenthesization matrix-chain multiplication whose sequence of dimensions are 2 x 6, 6 x 5, 5 x 4 and 4 x 3. **(4)**

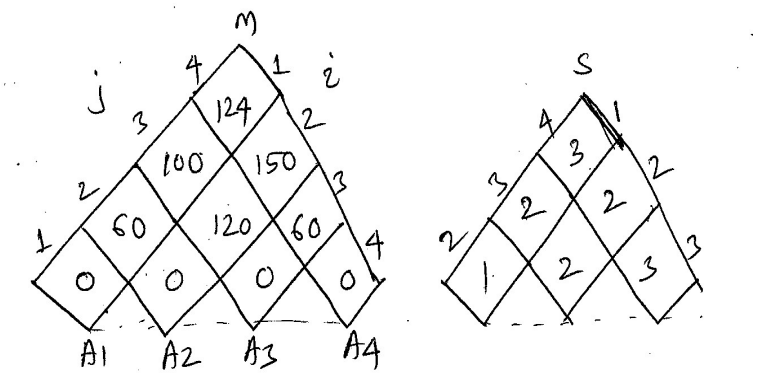
Scheme:

- Drawing m and s table with correct values: 3 Marks
- Finding optimal parenthesization of matrix-chain multiplication: 1 Mark

Answer:

The optimal parenthesization of matrix-chain multiplication: **(((A1A2)A3)A4)**

$p_0=2$, $p_1=6$, $p_2=5$, $p_3=4$, $p_4=3$



Step-1

m[1,2] = m[1,1] + m[2,2] + $p_0p_1p_2$ = 0 + 0 + 60 = 60

m[2,3] = m[2,2] + m[3,3] + $p_1p_2p_3$ = 0 + 0 + 120 = 120

m[3,4] = m[3,3] + m[4,4] + $p_2p_3p_4$ = 0 + 0 + 60 = 60

Step-2

m[1,3] = ?

for k=1 => m[1,1] + m[2,3] + $p_0p_1p_3$ = 0 + 120 + 48 = 168

k=2 => m[1,2] + m[3,3] + $p_0p_2p_3$ = 60 + 0 + 40 = **100 (Min)**

So, m[1,3] = 100 , s[1,3]=2

m[2,4] = ?

for **k=2** => m[2,2] + m[3,4] + $p_1p_2p_4$ = 0 + 60 + 90 = **150 (Min)**

k=3 => m[2,3] + m[4,4] + $p_1p_3p_4$ = 120 + 0 + 72 = 192

So, m[2,4] = 150 , s[2,4]=2

Step-3

m[1,4] = ?

for k=1 => m[1,1] + m[2,4] + $p_0p_1p_4$ = 0 + 150 + 36 = 186

k=2 => m[1,2] + m[3,4] + $p_0p_2p_4$= 60 + 60 + 60 = 180

k=3 => m[1,3] + m[4,4] + $p_0p_3p_4$=100 + 0 + 24 = **124 (Min)**

So, m[1,4] = 124 , s[1,4]=3

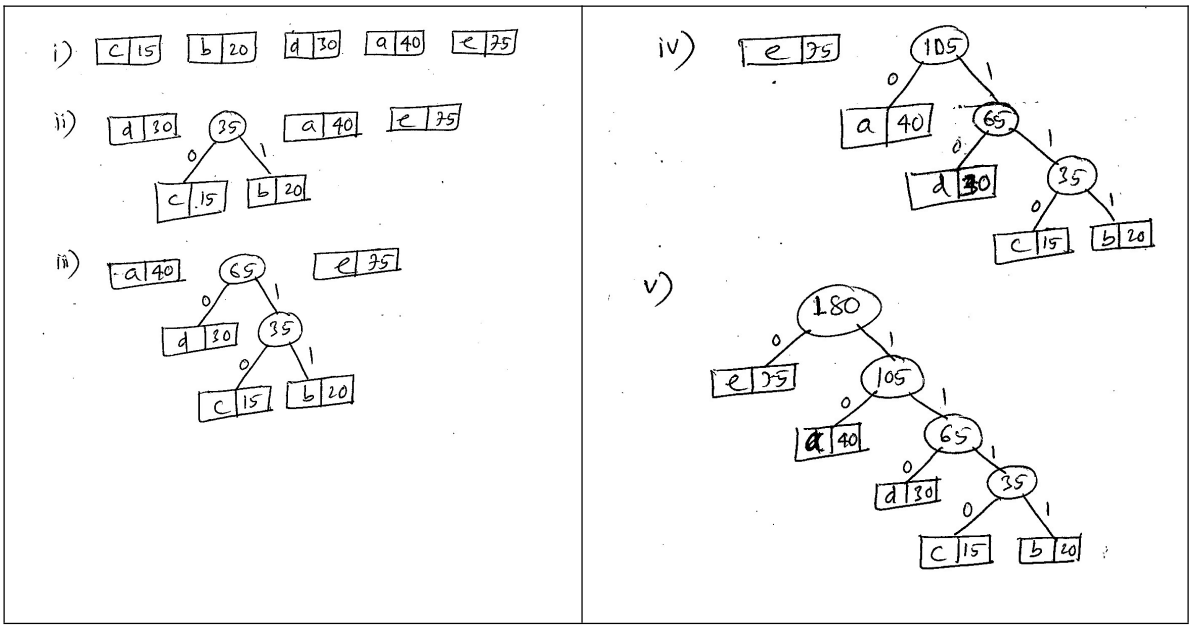
b) Find an optimal Huffman code for the following set of frequencies (4)
a:40 b:20 c:15 d:30 e:75

Scheme:

- Drawing Huffman tree step by step: 3 Marks
- Finding optimal Huffman code: 1 Mark

Answer:

Item	Code	or
a	10	01
b	1111	0000
c	1110	0001
d	110	001
e	0	1



Q4 a) Write master theorem. (4)

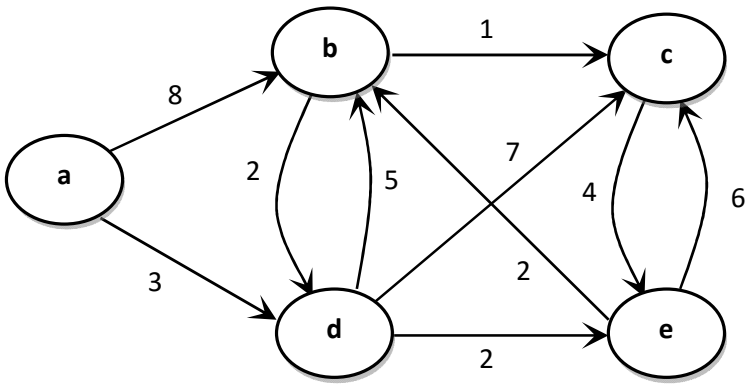
Scheme:

- Illustration of master theorem : 4 Marks

Answer:

- The Master Theorem applies to recurrences of the following form:
- $T(n) = aT(n/b) + f(n)$
- where $a \geq 1$ and $b > 1$ are constants and f(n) is an asymptotically positive function. T(n) is defined on the non-negative integers by the recurrence.
- T(n) can be bounded asymptotically as follows: There are 3 cases:
  a) **Case-1:** If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $\mathbf{T(n) = \Theta(n^{\log_b a})}$
  b) **Case- 2:** If $f(n) = \Theta(n^{\log_b a})$, then $\mathbf{T(n) = \Theta(n^{\log_b a} \log n)}$
  c) **Case-3:** If $f(n) = \Omega(n^{\log_b a + \epsilon})$ with $\epsilon > 0$, and : $af(n/b) \leq cf(n)$, then $T(n) = \Theta(f(n))$, for some constant $c < 1$ and all sufficiently large n, then $\mathbf{T(n) = \Theta(f(n))}$
- **Examples:**
- **Example (Case-1):**
  Solve the the recurrence $T(n)=9T(n/3) +n$
  Sol^n: For this recurrence, we have a=9, b=3, f(n)=n, and thus we have that $n^{\log_3 9} = n^2$ which is asymptotically larger than f(n), we guess case-1. Since $f(n)= O(n^{\log_3 9 - \epsilon})$, where $\epsilon=1$, we can apply case-1 of the master theorem and conclude that the solution is $T(n)= \Theta(n^2)$
- **Example (Case-2)**
  Solve the the recurrence $T(n)=T(2n/3) +1$
  For this recurrence, we have a=1, b=3/2, f(n)=1, and thus we have that $n^{\log_{3/2} 1} = n^0 = 1$ which is same as f(n). So case-2 applies, and thus the solution to the recurrence is $T(n)= \Theta(\log n)$
- **Example(Case-3)**
  Solve the the recurrence $T(n)=3T(n/4) +n \log n$
  For this recurrence, we have a=3, b=4, f(n)=n log n, and thus we have that $n^{\log_4 3} = n^{0.793} = 1$. Comparing $n^{\log_b a}$ with f(n), f(n) is seems to be larger. So case-3 may be applied. Since (n)= $O(n^{\log_4 3+\epsilon})$, where $\epsilon=0.2$, Also $af(n/b)=3(n/4)\log(n/4) \leqslant (3/4)n \log n = cf(n)$ for c=3/4. So we conclude it is case-3 and thus the solution to the recurrence is $T(n)= \Theta(n \log n)$
- **The master theorem does not apply to the recurrence** $T(n)=2T(n/2) + n \log n$ even though it has the proper form : a=2, b=2, f(n)=n log n, and $n^{\log_b a}=n$. It might seem that case-3 should apply. Since f(n)=n log n is asymptotically larger than $n^{\log_b a}=n$. The problem is that it is not polynomially lager. The ratio $f(n)/ n^{\log_b a}=n \log n/n=\log n$ is asymptotically less than n^{ϵ} for any positive constant ϵ.

b) Use suitable shortest path algorithm to find out shortest path between a to c and a to e. (4)



Scheme:

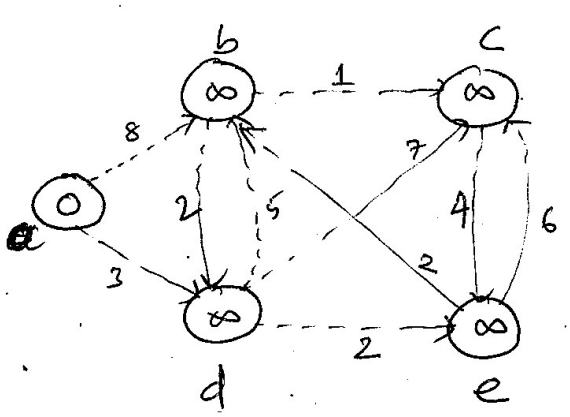
- Step by step description with correct answer by any shortest path algorithm: 4 Marks

Answer:

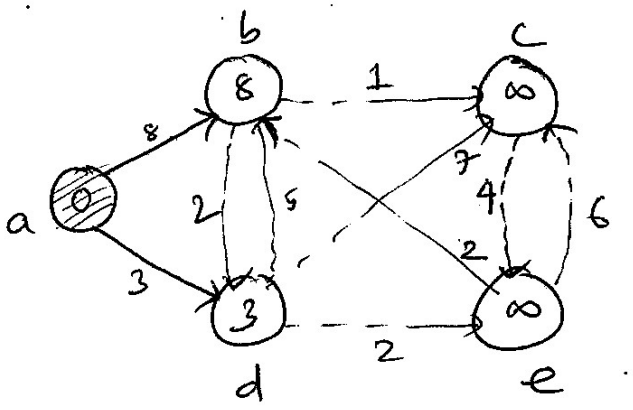
By Dijkstra's Algorithm

shortest path between a to c : 8 (Route: a-d-e-b-c)

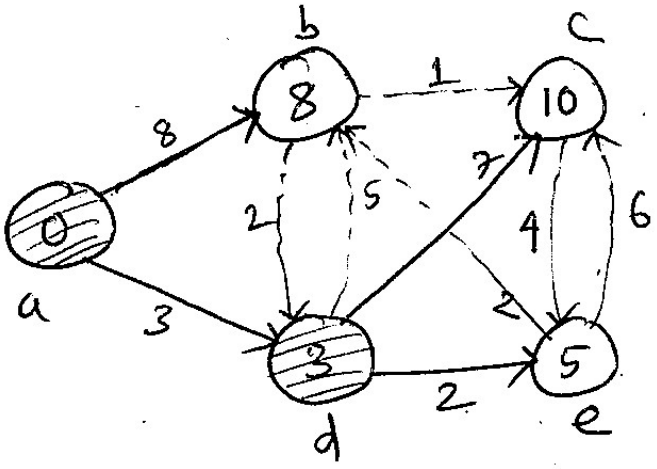
and a to e : 5 (Route: a-d-e)



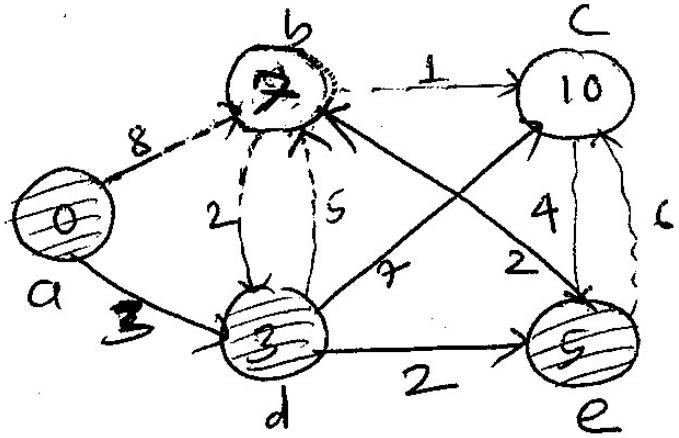
(Figure-0)



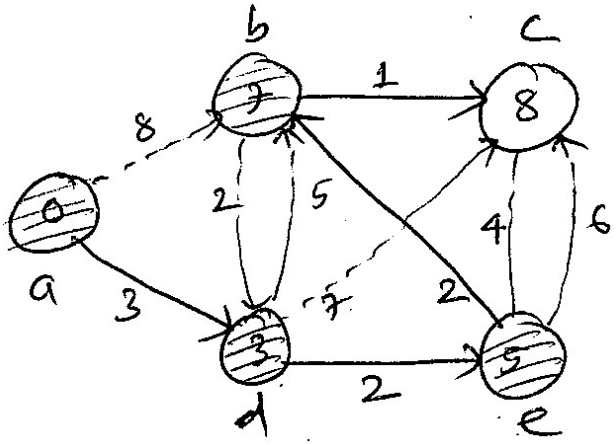
(Figure-1)



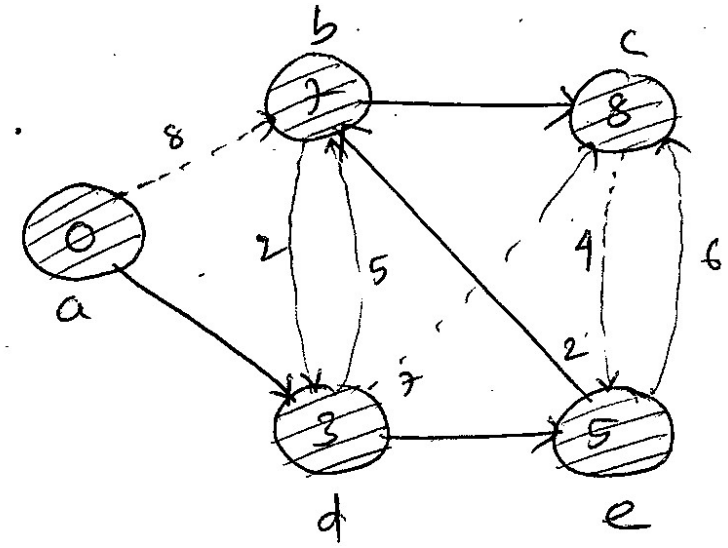
(Figure-2)



(Figure-3)



(Figure-4)



(Figure-5)

Q5 a) Determine an LCS of <a, b, b, a, b, a, b, a> and <b, a, b, a, a, b, a, a, b> (4)

Scheme:

- Correct LCS table construction: 3 Marks
- Finding correct LCS: 1 Mark

Answer:

	y_j	b	a	b	a	a	b	a	a	b
x_i	0	0	0	0	0	0	0	0	0	0
a	0	0 ↑	1 ↖	1 ↖	1 ↖	1 ↖	1 ←	1 ↖	1 ↖	1 ←
b	0	1 ↖	1 ↑	2 ↖	2 ←	2 ←	2 ↖	2 ←	2 ←	2 ↖
b	0	1 ↖	1 ↑	2 ↖	2 ←	2 ←	3 ↖	3 ←	3 ←	3 ↖
a	0	1 ↑	2 ↖	2 ↑	3 ↖	3 ↖	3 ↑	4 ↖	4 ↖	4 ←
b	0	1 ↖	2 ↑	3 ↖	3 ↑	3 ↑	4 ↖	4 ↑	4 ↑	5 ↖
a	0	1 ↑	2 ↖	3 ↑	4 ↖	4 ↖	4	5 ↖	5 ↖	5 ↑
b	0	1 ↖	2 ↑	3 ↖	4 ↑	4 ↑	5 ↖	5 ↑	5 ↑	6 ↖
a	0	1 ↑	2 ↖	3 ↑	4 ↖	5 ↖	5 ↑	6 ↖	6 ↖	6 ↑

LCS-1: a, b, b, a, a, b OR abbaab

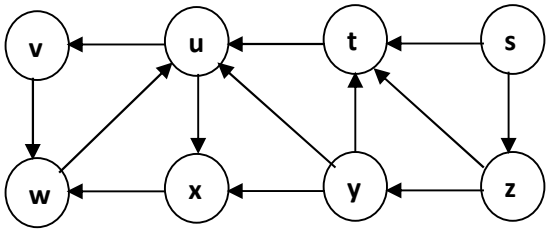
OR

LCS-2: a, b, a, b, a, a OR ababaa

OR

LCS-3: a, b, a, a, b, a OR abaaba

b) Traverse the following graph by DFS technique with 's' as start vertex. (4)



i) Draw the DFS tree/forest.

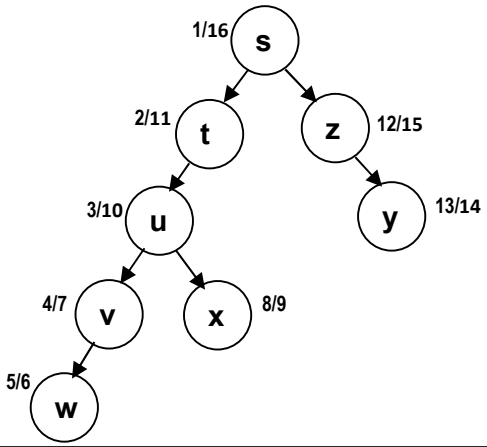
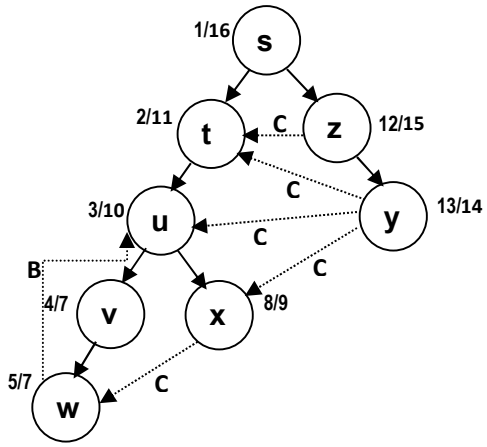
ii) Find out the DFS sequence.

Scheme:

- Drawing DFS DFS tree/forest: 3 Marks
- Finding correct DFS sequence: 1 Mark

Answer:

- In this case the DFS tree/forest is not unique. The sample answer is given as follows:

DFS tree/forest without forward edge, back age, cross edge (only with tree edges)	**DFS tree/forest with tree edge (solid lines) forward edge (labeled as F), back age (labeled as B), cross edge (labeled as C)**
	
DFS Sequence for this DFS tree/forest is **s, t, u, v, w, x, z, y**	**DFS Sequence** for this DFS tree/forest is **s, t, u, v, w, x, z, y**

Q6 a) Schedule the set of jobs in the following to obtain maximum profit and find out the total profit. Assume each job takes 2 Hrs. (4)

Job No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Profit	**15**	**18**	**4**	**25**	**3**	**4**	**10**	**6**
Dead line	**7**	**5**	**6**	**2**	**4**	**2**	**3**	**4**

Scheme:

- Explanation with correct answer: 4 Marks
- Correct Explanation with incorrect answer: 3 Marks

Answer:

Time Slots: 0-2, 2-4, 4-6, 6-8 etc.

Sort the jobs as per profit in descending order.

Job. No.	Profit	Deadline	Assigned to Slot
4	25	2	2
2	18	5	4
1	15	7	6
7	10	3	X
8	6	4	X
3	4	6	X
6	4	2	X
5	3	4	X

Now the optimal ordering of Jobs: 4, 2, 1

Total profit= 25+18+15=58

b) Define dis-joint set data structure. Discuss the tree based algorithms FIND-SET(x) and UNION(x,y). (4)

Scheme:

- Definition: 1Mark
- Each algorithm: 1 Mark

Answer:

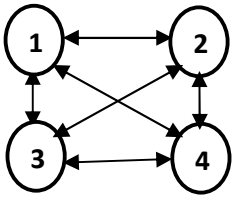
- In mathmatics, two sets are said to be disjoint sets if they have no elements in common. For example, {1, 2, 3,5,8} and {4, 5, 6,9} are disjoint sets. The formal definition of disjoint set is $S=\{S_1,S_2,\ldots,S_K\}$, $S_i \cap S_j = \emptyset$, $1 \leq i < j \leq k$. where S is a collection of sets, and any two sets in S are nonoverlapping.
- **In computer science, a disjoint-set data structure is a data structure used to store and keep track of a set of elements partitioned into disjoint (non-overlapping) subsets.**
- In other words A disjoint-set data structure maintains a collection of disjoint dynamic sets $S=\{S_1,S_2,\ldots,S_k\}$.
- Each set is identified by a representative, which is some member of the set.

FIND-SET(x) Algorithm	UNION(x,y) Algorithm

FIND (x) - Starting at x, traverse parent pointers up to the root. O(n)
 	
Iterative version
FIND-SET(x)
{
while (x ≠ parent[x])
x ← parent[x]
return x
}
 	
Recursive version
FIND-SET(x)
{
if (x == parent[x])
return x
else return FIND-SET(parent[x])
	UNION(x,y) - Set root y's parent pointer to root x. O(1)
	- Here we first find out FIND(x) and FIND(y) which yields the roots of the trees containing x and y respectively. Let it be xRroot and yRroot respectively.
	
	UNION(x,y)
	{
	xRoot=FIND-SET(x)
	yRoot=FIND-SET(y)
	parent[yRoot] ← xRoot
	}
	UNION BY RANK
	• In this heuristic when merging or unioning two sets (trees) together, we would like to attach the root of the smaller tree (tree with smaller depth)

}	to the root of the larger tree (tree with larger depth).
	• This requires us to maintain the depth of each tree, called rank of the tree.
	• By keeping this rank the two roots can be compared and the one with a smaller rank can be attached to the root with a larger rank. After the union if the two roots being joined had equal ranks the new root nodes rank will be increased by 1.
	/*UNION BY RANK ALGORITHM*/
	UNION-RANK(x,y)
	{
	xRoot=FIND-SET(x)
	yRoot=FIND-SET(y)
	if (rank[xRoot] > rank[yRoot])
	then parent[yRoot] ← xRoot
	else
	{
	parent[xRoot] ← yRoot
	if (rank[xRoot] = rank[yRoot])
	then rank[yRoot] ← rank[yRoot] + 1
	}
	}

Q7 a) Consider the following instance of travelling salesperson problem. (4)



	1	2	3	4
1	0	5	10	15
2	4	0	5	6
3	2	8	0	7
4	10	4	5	0

Find the tour of the travelling salesperson and minimum cost of the tour with starting vertex 1.

Scheme:

- Correct steps with correct answer: 4 Marks
- Correct steps with incorrect answer: 3 Marks

Answer:

g(i,S) = length of shortest path starting at vertex i, going through all vertices in S and terminating at vertex 1=$\min_{j \in S}\{c_{ij} + g(j, S-\{j\})\}$

- **Step-1:** with |S|= Ø, calculate g(i, Ø)
  g(2, Ø)=c_{21}=4
  g(3, Ø)=c_{31}=2
  g(4, Ø)=c_{41}=10
- **Step-2:** with |S|=1, calculate g(i,S)
  All possible S are {2}, {3} and {4}
  g(3,{2})=c_{32} + g(2, Ø) =8 + 4=12

$g(4,\{2\})=c_{42} + g(2, Ø) =4 + 4=8$

$g(2,\{3\})=c_{23} + g(3, Ø) =5 + 2=7$
$g(4,\{3\})=c_{43} + g(3, Ø) =5 + 2=7$

$g(2,\{4\})=c_{24} + g(4, Ø) =6 + 10=16$
$g(3,\{4\})=c_{34} + g(4, Ø) =7 + 10=17$

- **Step-3:** with |S|=2, calculate g(i,S)
  All possible S are {2,3}, {2,4} and {3,4}

$$g(4,\{2,3\}) = \begin{cases} c_{42} + g(2, \{3\}) =4 + 7=11 \text{ (Min)} \\ c_{43} + g(3, \{2\}) =5+ 12=17 \end{cases}$$

$$g(3,\{2,4\}) = \begin{cases} c_{32} + g(2, \{4\}) =8 + 16=24 \\ c_{34} + g(4, \{2\}) =7+ 8=15 \text{ (Min)} \end{cases}$$

$$g(2,\{3,4\}) = \begin{cases} c_{23} + g(3, \{4\}) =5 + 17=22 \\ c_{24} + g(4, \{3\}) =6+ 7=13 \text{ (Min)} \end{cases}$$

- **Step-4:**with |S|=3, calculate g(i,S)
  Only possible of S is {2,3,4}

$$g(1,\{2,3,4\}) = \begin{cases} c_{12} + g(2, \{3,4\}) =5 + 13=18 \text{ (Min)} \\ c_{13} + g(3, \{2,4\}) =10 + 15=25 \\ c_{14} + g(4, \{2,3\}) =15+ 17=32 \end{cases}$$

- **So the optimal tour is 1->2->4->3->1 (Total Distance=18)**

b) Write the algorithm for n-queens problem. Explain it for 4 queens. (4)

Scheme:

- Correct algorithm: 4 Marks

Answer:

n-Queen Problem

- This n-queens problem is to place n-queens on an n x n chess board in such a way that no two queens attack; that is, no two queens are on the same row, column or diagonal.
- The solution vector X (X1…Xn) represents a solution in which Xi is the column of the i^{th} row where i^{th} queen is placed.
- First, we have to check no two queens are in same row.
- Second, we have to check no two queens are in same column.
- The function, which is used to check these two conditions, is [I, X (j)], which gives position of the I^{th} queen, where I represents the row and X (j) represents the column position.
- Third, we have to check no two queens are in it diagonal.

- Consider two dimensional array A[1:n,1:n] in which we observe that every element on the same diagonal that runs from upper left to lower right has the same value.
- Also, every element on the same diagonal that runs from lower right to upper left has the same value.
- Suppose two queens are in same position (i,j) and (k,l) then two queens lie on the same diagonal , if and only if |j-l|=|I-k|.

Algorithm NQueen (k,n)

```
/*using backtracking this procedure prints all possible positions of n queens on n x n chessboard. So that they are non-tracking*/
{
    for i=1 to n do
    {
        if place (k,i) then
            {
              X [k]=i;
               if (k=n) then write (X [1:n]);
               else NQeen(k+1,n)    ;
            }
     }
}
```

Algorithm place (k,i)

```
/*Returns true if a queen can be placed in k th row and ith column. otherwise it returns false. x[] is a global array whose first k-1 values have been set. Abs(r) returns the absolute value of r*/
{
  for j=1 to k-1 do
   {
    If ((X [j]=i)                    //two in same column.
     Or (Abs (X [j]-i) = Abs (j-k)))
     Then return false;
    }
    return true;
}
```

4-Queen Problem

Example: 4 queens.

Let's discuss this with example to find solution of 4 queen problem.

Algorithm:

- Start with one queen at the first column first row
- Continue with second queen from the second column first row
- Go up until find a permissible situation
- Continue with next queen

	1	2	3	4
1		Q		
2				Q
3	Q			
4			Q	

Solution 2

(2 4 1 3)

	1	2	3	4
1			Q	
2	Q			
3				Q
4		Q		

Solutin-1

(3 1 4 2)

Q8 a) Write an algorithm to merge two sorted array A (increasing order) and B (decreasing order) to a single sorted array C in decreasing order. **(4)**

Scheme:

- Correct algorithm: 4 Marks

Answer:

```
MERGE(A,lb1, ub2, B, lb2, ub2, C)
{
 i=ub2, j=lb2, k=1;
  while(i>=lb and j<=ub2)
  {
     if(A[i]>B[j))
     {
        C[k]=A[i]
        k=k+1
        i=i-1
     }
     else
     {
       C[k]=B[j]
       k=k+1
       j=j+1
     }
  }
  while(i>=lb)
  {
        C[k]=A[i]
        k=k+1
        i=i-1
 }
  while(j<=ub2)
  {
       C[k]=B[j]
       k=k+1
       j=j+1
 }
}
```

b) Write prim's algorithm to find out minimum cost spanning tree. Explain its time complexity. **(4)**

Scheme:

- Prim's Algorithm: 3Marks
- Time Complexity: 1 Mark

Answer:

Prim's Algorithm

```
MST-PRIM(G, w, r)
{
    for each u ∈ V [G]
    {
        key[u] ← ∞
        π[u] ← NIL
    }
    key[r] ← 0
    Q ← V [G]
    while Q ≠ Ø
    {
        u ← EXTRACT-MIN(Q)
        for each v ∈ Adj[u]
        {
            if v ∈ Q and w(u, v) < key[v]
            {
                π[v] ← u
                key[v] ← w(u, v)
            }
        }
    }
}
```

Time Complexity

Total time=O(V log V+E log V)=O(E log V)

Paper Setter……

Moderator……