I – 52492

Register No. | | | | | | | | |

BTech Degree Examination May 2022

Fourth Semester

Information Technology

20ITT43 – DESIGN AND ANALYSIS OF ALGORITHMS

(Regulation 2020)

Time: Three hours | Maximum: 100 marks

Answer all Questions

Part – A (10 × 2 = 20 marks)

1. Define Ω - notation. [CO1,K1]

2. Consider the following algorithm [CO1,K3]

ALGORITHM B(n)

Count ← 1
while n>1 do
  Count ← count +1
  $n \leftarrow \lfloor n/2 \rfloor$
return count

What does the algorithm compute? Find its time efficiency?

3. Find the order of growth for the solution of the following recurrence using master theorem. [CO2,K3]

$T(n) = 4\,T(n/2) + n, \qquad T(1) = .1$

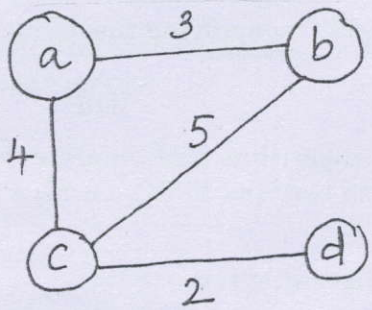
4. Write the time complexity and the number of key swaps that are required in sorting a set of 'n' given numbers using selecting sort algorithm. [CO2,K1]

5. Is bubble sort stable? Justify your answer with an example. [CO3,K2]

6. Compare finding the solution for the closest pair problem using brute force approach and divide-and-conquer approach. [CO3,K2]

7. Differentiate between dynamic programming technique and greedy technique. [CO4,K2]

8. Draw any two possible spanning trees for the given graph G. [CO1,K1]



9. Indicate the solution obtained first for the 5 queen's problem which is solved by backtracking technique. [CO5,K3]

10. Define class P problem. [CO5,K1]

Part – B (5 × 16 = 80 marks)

11. a. i) Using direct method derive the formula to compute the n^{th} Fibonacci number ie F(n), given that F(0)=0 and F(1)=1. (6) [CO1,K2]

ii) Design a recursive algorithm for computing 4^n for any nonnegative integer 'n' that is based on the formula: (10) [CO1,K4]

$$4^n = 4^{n-1} + 4^{n-1} + 4^{n-1} + 4^{n-1}$$

(a) Set up a recurrence relation for the number of additions made by the algorithm and solve it.

(b) Does the efficiency of the recursive algorithm for computing 4^n differ from the efficiency of non-recursive algorithm for computing 4^n. Proof your answer.

(OR)

b. i) Elaborate in detail the sequence of steps one typically goes through in designing and analyzing an algorithm. (6) [CO1,K2]

ii) Consider the following algorithm: (10) [CO1,K4]

ALGORITHM D (A{0..N-1})

```
//Input: An array A{0..N-A} of n real numbers
minval←A[0]: maxval ←A[o]
for i←1 to n-1 do
  if A[i]< minval
     minval←A[i]
  if A[i]> maxval
     maxval←A[i]
return maxval-minval
```

a) What does this algorithm compute?
b) What is its basic operation?
c) How many times is the basic operation executed?
d) What is the efficiency class of this algorithm?
e) Bring out an improvement or a better algorithm altogether, and indicate its efficiency class. If you cannot do it, try to prove that, in fact, it cannot be done.

12. a. i) Write quicksort algorithm and analyze its average-case time complexity. Sort the list A, L, G, O, R, I, T, H, M in alphabetical order using quick sort. (10) [CO2,K2]

ii) Compute 2105 X 1140 by applying the divide- and conquer technique. (6) [CO2,K3]

(OR)

b. i) Write mergesort algorithm and analyze its worst-case time efficiency. Apply mergesort to sort the list C, O, M, P, U, T, I, N, G in alphabetical order. (10) [CO2,K2]

ii) Apply strassen's algorithm to compute (6) [CO2,K3]

$$\begin{pmatrix} 1 & 3 & 2 & 1 \\ 4 & 2 & 1 & 1 \\ 0 & 2 & 3 & 2 \\ 5 & 4 & 2 & 1 \end{pmatrix} X \begin{pmatrix} 0 & 2 & 1 & 3 \\ 2 & 1 & 0 & 4 \\ 2 & 1 & 3 & 1 \\ 1 & 3 & 5 & 2 \end{pmatrix}$$

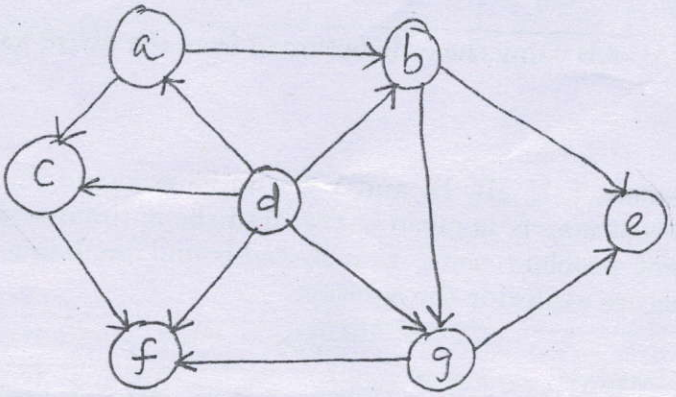
13. a. i) Construct an AVL tree by inserting their elements successively, starting with the empty tree. (6) [CO3,K3]

3, 6, 5, 1, 2, 4, 7.

ii) Consider the problem of finding whether a given array A[0..N-1] contains only unique elements. Write the algorithms for the problem with presorting and without presorting technique. Analyze the time complexity in both the algorithms. (10) [CO3,K3]

(OR)

b. i) Apply the DFs-based algorithm to solve the topological sorting problem for the following digraph. (6) [CO3,K3]



ii) Write insertion sort algorithm and analyze its worst case, best case, and average case time efficiency. Apply insertion sort to sort the set of numbers 89, 45, 68, 90, 29, 34, 17 in ascending order. (10) [CO3,K3]

14. a. i) Illustrate the optimal binary search tree algorithm by applying it to the four-key set and its probability as given below:- (8) [CO4,K3]

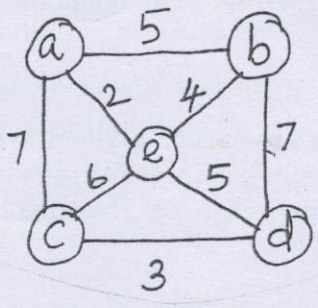
Key	A	B	C	D
Probability	0.1	0.2	0.4	0.3

ii) Solve the all-pairs shortest-path problem using floyd's algorithm for the diagraph with the following weight matrix: (8) [CO4,K3]

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & \infty & 0 & \infty \\ \infty & 2 & 9 & 0 \end{bmatrix} \end{array}$$

(OR)

b. i) Using Kruskal's algorithm find MST for the graph G given below: (8) [CO4,K3]



ii) Construct a Huffman code for the following data: (8) [CO4,K3]

Symbol	A	B	C	D	-
Frequency	0.3	0.2	0.15	0.2	0.15

Encode ABAC-AB using the code obtained from the above data.

15. a. Suppose five persons I, II, III, IV and V are to be assigned the jobs J1, J2, J3 and J4 and the cost matrix is given below. find the optimal assignment for the given assignment problem using branch-and-bound technique and illustrate with a tree structure exploring the solution. (16) [CO5,K3]

Persons	Job			
	J1	J2	J3	J4
I	86	78	62	81
II	55	79	65	60
III	72	65	63	60
IV	86	70	65	71
V	72	70	71	60

(OR)

b. Solve the following instance of the knapsack problem by the branch-and-bound algorithm: Given W=16. (16) [CO5,K3]

Item	Weight	Value
1	9	$100
2	7	$73
3	8	$65
4	4	$23

11.11
10.43
8.13
5.75

Bloom's Taxonomy Level	Remembering (K1)	Understanding (K2)	Applying (K3)	Analysing (K4)	Evaluating (K5)	Creating (K6)
Percentage	3	21	65	11	-	-

KONGU ENGINEERING COLLEGE, PERUNDURAI – 638 060, ERODE
DEPARTMENT OF INFORMATION TECHNOLOGY
APRIL/MAY 2022 END SEMESTER EXAMINATIONS

ANSWER KEY
20ITT43 – DESIGN AND ANALYSIS OF ALGORITHMS
(Regulations 2020)

Part – A (10 × 2 = 20 marks)

1. **Ω**– Notation:
   DEFINITION: A function $t(n)$ is said to be in $\Omega(g(n))$, denoted as $t(n)\in\Omega(g(n))$, if $t(n)$ is bounded below by some constant multiple of $g(n)$ for all large n, i.e., if there exist some positive constant c and some non-negative integer n_0 such that:

$$t(n)\geq cg(n)\ \ foralln\geq n\_o$$

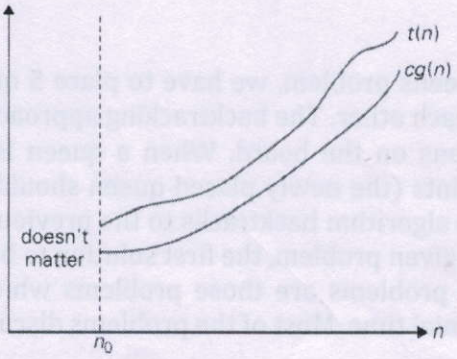


FIGURE Big-omega notation: $t(n) \in \Omega(g(n))$.

2. The algorithm `B(n)` computes the number of bits required for the binary representation of a given decimal number. The algorithms time complexity is: $\log_2 n$
3. Given: $T(n) = 4T\left(\frac{n}{2}\right) + n$, From the recurrence relation we get $a = 4$, $b = 2$ and $d = 1$. Therefore, according to Master theorem, $a > b^d$ which implies that:
$$T(n) \in \Theta\left(n^{\log_b a}\right) = \Theta(n^2).$$
4. The time complexity of Selection Sort is $\Theta(n^2)$and the number of swaps requited to sort a list of n numbers is $(n-1)$.
5. Yes, bubble sort is a stable algorithm: Illustrated as follows:
   Assume the following inputs:

10	**20**	**20**	**30**	**10**
10	**20**	**20**	**10**	**30**
10	**20**	**10**	**20**	**30**
10	**20**	**10**	**20**	**30**
10	**10**	**20**	**20**	**30**

   We see that there are 2 keys with same value, 10 and 10, 20 and 20. From, the algorithms operation we see that the relative ordering between 10s and 20s is preserved in the sorted list. Hence, bubble sort is a stable algorithm.
6. Given a set of n points in a Cartesian plane, the objective of closest pair problem is to find the closest two points (in terms of Euclidean distance) among the n given points. There are 2 approaches – Brute force and Divide & Conquer.
   In Brute force approach, every point (p_i) is compared with the remaining points (p_j) and the distance is computed, keeping track of the minimum distance seen so far. The complexity is $\Theta(n^2)$.

In Divide & Conquer approach, the given set of points is partitioned into two sets and the closest pair is called recursively on both sets to give closest points in both sets - d_l *and* d_r. Then the closest points will have the distance $d = min\{d_l, d_r\}$. As the part of combining the solutions, the closest points, if any, is across the partition - inside the symmetric vertical strip of width $2d$ around the separating line, since the distance between any other pair of points is at least d. This algorithm's time complexity is $T(n) \in \Theta(n \log_2 n)$.

7. Dynamic Programming: An algorithm strategy to solve optimization problems with overlapped sub-solutions spaces.
   Greedy Approach: An algorithm strategy to solve optimization problems with incremental local optimal steps, provided it is feasible. A step once taken cannot be revoked. Does not guarantee optimal solutions for all problems.
8. Two spanning trees of the given graph are any two of the following:

9. In 5-Queens problem, we have to place 5 queens in a 5×5 chessboard, such that no two queens attack each other. The backtracking approach proceeds by DFS of the state-space tree starting with no queens on the board. When a queen is placed in the j^{th} cell of the i^{th} row, the problem's constraints (the newly placed queen should not attack the existing queens) should be satisfied. If not, the algorithm backtracks to the previous state to chose the other alternative in a DFS manner. For the given problem, the first solution to be retrieved would be:
10. Class P problems are those problems which can be solved by some deterministic algorithm in polynomial time. Most of the problems discussed in the course are of P class.

Part – B ($5 \times 16 = 80$ marks)

11. (a)i. Computing the n^{th} Fibonacci number by the direct method.
    First, let us get the explicit formula for the n^{th} Fibonacci number. We may not be able to use backward substitution. Therefore, we make us of use a theorem which finds solution to a *Homogenous Second-order Recurrence relation with constant coefficients namely,*

$$ax(n) + bx(n-1) + c(n-2) = 0$$

where *a*, *b*, and *c* are some fixed real numbers called the coefficients of the recurrence and $x(n)$ is the generic term of an unknown sequence to be found.Applying this theorem to our recurrence with the initial conditions given we obtain:

$$F(n) = \frac{1}{\sqrt{5}}(\phi^n - \hat{\phi}^n)$$

Where, $\phi = {(1+\sqrt{5})}/{2} \approx 1.61803$ and $\hat{\phi} = -\frac{1}{\phi} = -0.61803$.

(a) ii. Recursive algorithm to compute 4^n:

See attachment.

11. (b) i. General sequence of steps in Algorithm design and analysis: The student should have outlined the following steps and briefed on each step.

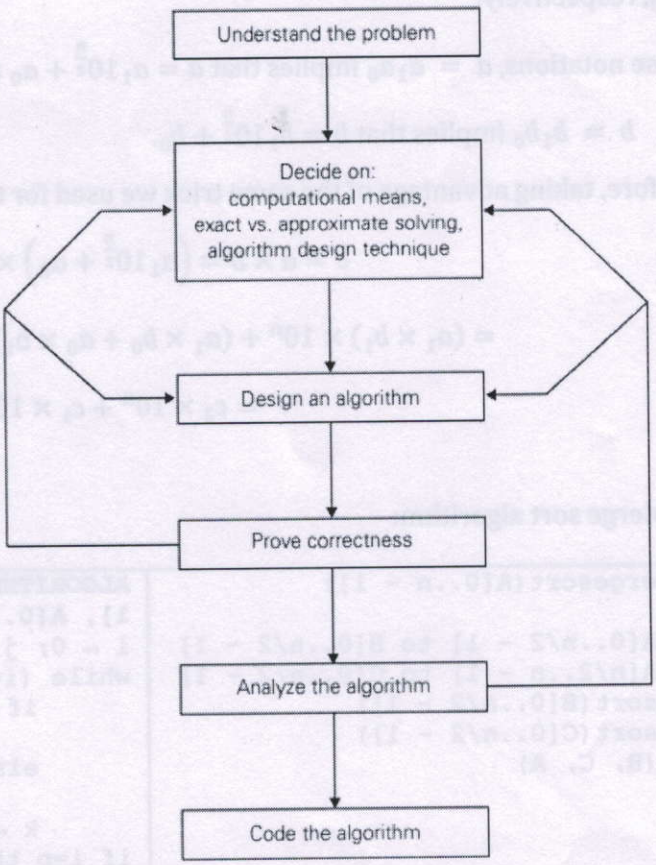


FIGURE 1.2 Algorithm design and analysis process.

(ii). (a) The algorithm computes the max and min values in an array A[n]

(b) The basic operation is comparison.
(c) The basic operation is executed $(n-1)$ times.
(d) $\Theta(n)$
(e) Based on student's justification.

12. a. (i). Quick Sort Algorithm:

```
ALGORITHM Quicksort(A[l..r])

If(l>r)
        s ← HoarePartition(A[l..r])
        Quicksort(A[l..s-1])
        Quicksort(A[s+1..r])
```

```
ALGORITHM HoarePartition(A[l..r])
p ← A[l]; i ← l; j ← r + 1
repeat
        repeat i ← i + 1 until A[i] ≥ p
        repeat j ← j - 1 until A[j] ≤ p
        swap(A[i], A[j])
until (i ≥ j)
//undo last swap when i ≥ j swap(A[l], A[j])
swap(A[i], A[j])
return j
```

Sorting the list:

A L G O R I T H M

See attachment.

(a) ii. Compute 2015 × 1140, Applying divide and conquer.

We denote the first half of the a's digits by a_1 and the second half by a_0; for b, the notations are b_1 and b_0, respectively.

In these notations, $a = a_1 a_0$ implies that $a = a_1 10^{\frac{n}{2}} + a_0$ and

$b = b_1 b_0$ implies that $b = b_1 10^{\frac{n}{2}} + b_0$.

Therefore, taking advantage of the same trick we used for two-digit numbers, we get:

$$c = a \times b = \left(a_1 10^{\frac{n}{2}} + a_0\right) \times \left(b_1 10^{\frac{n}{2}} + b_0\right)$$

$$= (a_1 \times b_1) \times 10^n + (a_1 \times b_0 + a_0 \times b_1) \times 10^{\frac{n}{2}} + (a_0 \times b_0) \times 10^0$$

$$= c_2 \times 10^n + c_1 \times 10^{\frac{n}{2}} + c_0$$

(b) i. Merge sort algorithm:

```
ALGORITHM Mergesort(A[0..n - 1])
if n > 1
      copy A[0..n/2 - 1] to B[0..n/2 - 1]
      copy A[n/2..n - 1] to C[0..n/2 - 1]
      Mergesort(B[0..n/2 - 1])
      Mergesort(C[0..n/2 - 1])
      Merge(B, C, A)
```

```
ALGORITHM Merge(B[0..p - 1], C[0..q - 1], A[0..p + q - 1])
i ← 0; j ← 0; k ← 0
while (i<p) and (j<q)
      if (B[i] <= C[j]) then
            A[k] ← B[i]; i ← i+1;
      else
            A[k] ← C[j]; j ← j+1;
      k ←k+1
if i=p then
      copy C[j..q-1] to A[k..p+q-1]
else
      copy B[i..p-1] to A[k..p+q-1]
```

Sorting the letter in

C O M P U T I N G

See Attachment

(b) ii. Matrix multiplication using Strassen's Approach.

The principal insight of the algorithm lies in the discovery that we can find the product C of two 2×2 matrices A and B with just seven multiplications as opposed to the eight required by the brute-force algorithm.

This is accomplished by using the following formula:

$$\begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \times \begin{bmatrix} b_{00} & b_{01} \\ b_{10} & b_{11} \end{bmatrix} = \begin{bmatrix} m_1 + m_4 - m_5 + m_7 & m_3 + m_5 \\ m_2 + m_4 & m_1 + m_3 - m_2 + m_6 \end{bmatrix}$$

Where,

$m_1 = (a_{00} + a_{11}) \times (b_{00} + b_{11})$

$m_5 = (a_{00} + a_{01}) \times b_{11}$

$$m_2 = (a_{10} + a_{11}) \times b_{00}$$

$$m_3 = a_{00} \times (b_{01} + b_{11})$$

$$m_4 = a_{11} \times (b_{10} + b_{00})$$

$$m_6 = (a_{10} + a_{00}) \times (b_{01} + b_{01})$$

$$m_7 = (a_{01} + a_{11}) \times (b_{10} + b_{11})$$

13. (a) i. AVL Tree Construction for the keys: 3, 6, 5, 1, 2, 4, 7

See Attachment

(a) ii. Finding whether an array contains unique elements – with and without pre-sorting:

Without Pre-Sorting

```
ALGORITHMElementUniqueness(A[0..n - 1])
for i ← 0 to n - 2 do
          for j ← i+1 to n-1 do
      if (A[i] = A[j]) then
           return false
return true
```

Complexity: $\Theta(n^2)$

With Pre-sorting

```
ALGORITHM
PresortElementUniqueness(A[0..n - 1])
for i ← 0 to n - 2 do
      if (A[i] = A[i + 1]) then
           return false
return true
```

Complexity: $\Theta(n)+\Theta(n \log_2 n)=\Theta(n \log_2 n)$

13. (b) i. DFS based Topological Sorting:

- The first algorithm is a simple application of depth-first search: perform a DFS traversal and note the order in which vertices become dead-ends.
- Reversing this order yields a solution to the topological sorting problem, provided, of course, no back edge has been encountered during the traversal.
- If a back edge has been encountered, the digraph is not a *DAG*, and topological sorting of its vertices is impossible.

See attachment

(b) ii. Insertion sort Algorithm:

```
ALGORITHM InsertionSort(A[0..n - 1])
for i ← 1 to (n - 1) do
      v ← A[i]
      j ← i - 1
      while (j ≥ 0) and (A[j] > v) do
            A[j + 1]← A[j]
            j ← j - 1
      A[j + 1] ← v
```

89 | **45**	68	90	29	34	17	
45	89 | **68**	90	29	34	17	
45	68	89 | **90**	29	34	17	
45	68	89	90 | **29**	34	17	
29	45	68	89	90 | **34**	17	
29	34	45	68	89	90 | **17**	
17	29	34	45	68	89	90

FIGURE Example of sorting with insertion sort. A vertical bar separates the sorted part of the array from the remaining elements; the element being inserted is in bold.

$$C_{best}(n) = \sum_{i=1}^{n-1} 1 = (n-1) \in \Theta(n)$$

$$C_{avg}(n) \approx \frac{n^2}{4} \in \Theta(n^2)$$

14. (a) i. Optimal binary search tree:

key	A	B	C	D
probability	0.1	0.2	0.4	0.3

The initial tables look like this:

main table

	0	1	2	3	4
1	0	0.1			
2		0	0.2		
3			0	0.4	
4				0	0.3
5					0

root table

	0	1	2	3	4
1		1			
2			2		
3				3	
4					4
5					

Let us compute $C(1, 2)$:

$$C(1,2) = \min \left\{ \begin{array}{ll} k=1: & C(1,0) + C(2,2) + \sum_{s=1}^{2} p_s = 0 + 0.2 + 0.3 = 0.5 \\ k=2: & C(1,1) + C(3,2) + \sum_{s=1}^{2} p_s = 0.1 + 0 + 0.3 = 0.4 \end{array} \right\}$$
$$= 0.4.$$

main table

	0	1	2	3	4
1	0	0.1	0.4	1.1	1.7
2		0	0.2	0.8	1.4
3			0	0.4	1.0
4				0	0.3
5					0

root table

	0	1	2	3	4
1		1	2	3	3
2			2	3	3
3				3	3
4					4
5					

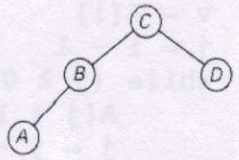
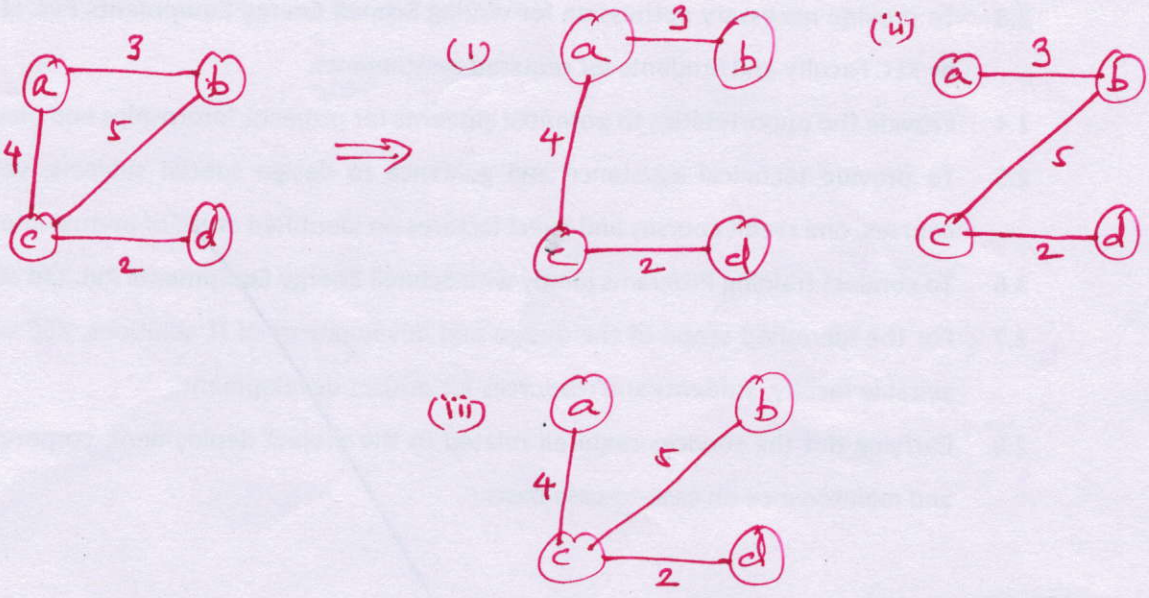


FIGURE Optimal binary search tree for the example.

(a). ii: Floyd's Algorithm:

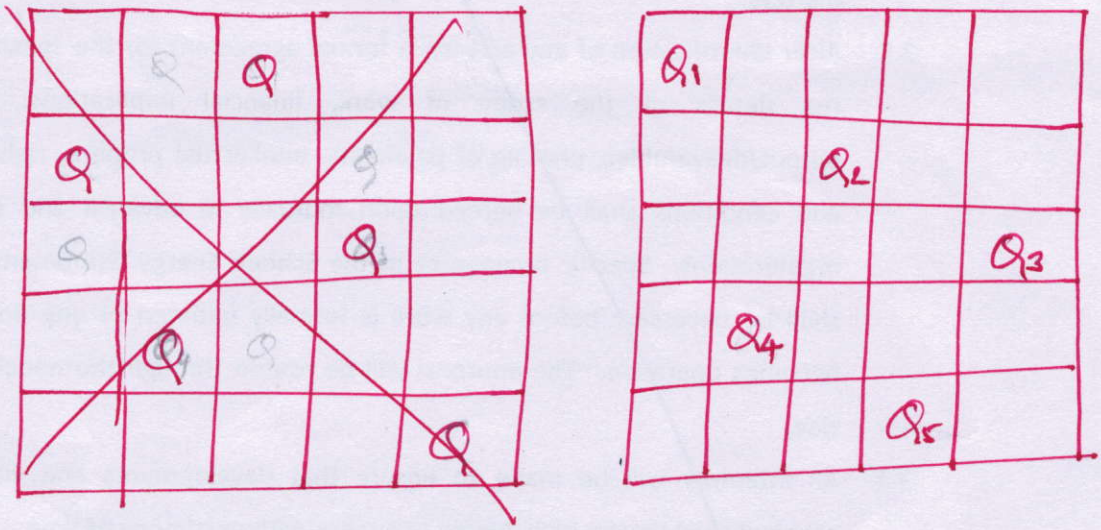
See Attachment.

(8)



---

(9)



---

11. a. (ii) $4^n = 4^{n-1} + 4^{n-1} + 4^{n-1} + 4^{n-1}$ for $n > 0$

Algorithm Power(4, n) ;

if $n = 0$

return (1)

else return Power(4, n-1) + Power(4, n-1) + Power(4, n-1) + Power(4, n-1)

12. a. (1)

l	i							r	
0	1	2	3	4	5	6	7	8	9
A	L	G	O	R	I	T	H	M	
Pivot									
A	L	G	O	R	I	T	H	M	
A	L	G	O	R	I	T	H	M	
A	L	G	O	R	I	T	H	M	
A	L	G	O	R	I	T	H	M	
A	L	G	O	R	I	T	H	M	
A	L	G	O	R	I	T	H	M	
A	L	G	O	R	I	T	H	M	
A	L	G	O	R	I	T	H	M	
A	L	G	O	R	I	T	H	M	

Proceed with quicksort . . .

12| a| (ii)

$$A = (2\ \ 1 \mid 0\ \ 5) \qquad (a_1\ a_0)$$

$$B = (1\ \ 1 \mid 4\ \ 0) \qquad (b_1\ b_0)$$

$$C = AB = C_2 10^4 + C_1 \times 10^2 + C_0 \times 10^0$$

$$C_2 = a_1 \times b_1 = 231$$

$$C_1 = 21 \times 40 + 11 \times 5 = 895$$

$$C_0 = 5 \times 40 = 200$$

$$\therefore C = 231 \times 10^4 + 895 \times 10^2 + 200 \times 1$$

$$C = 2310000 + 89500 + 200$$

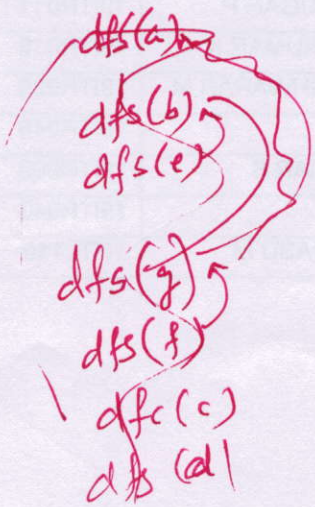
$$C = 2,399,700$$

Suppose we start dfs 'd'

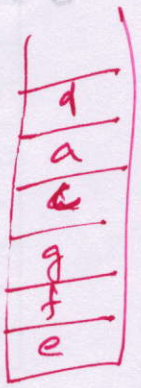
d a b e g f c

dfs(a)
dfs(b)
dfs(e)
dfs(g)
dfs(f)
dfs(c)

d a c b g f e

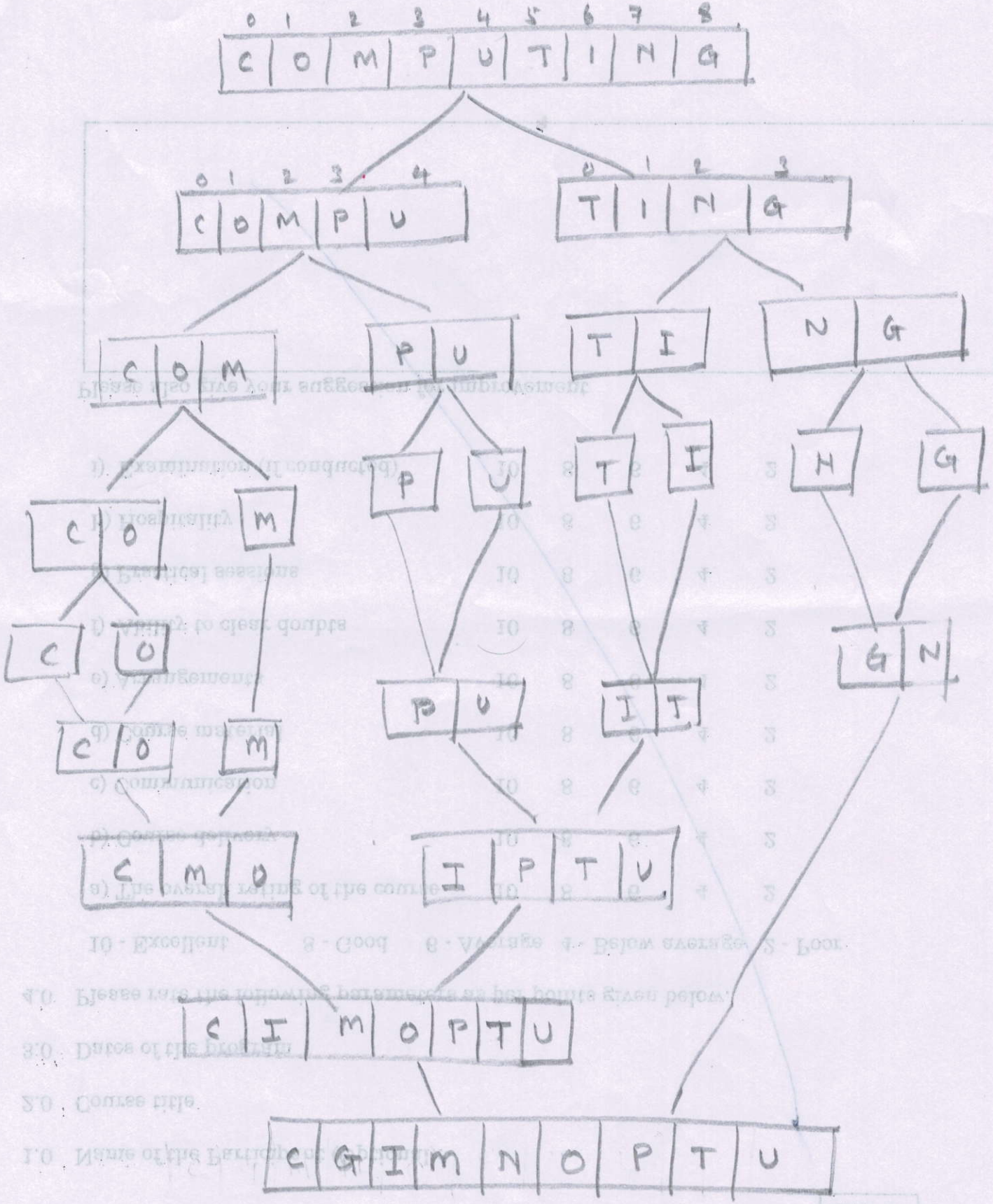


d a b e g f



d a c g f e

12. b. (i)


0 1 2 3 4 5 6 7 8
C O M P U T I N G
0 1 2 3 4
C O M P U
0 1 2 3
T I N G
C O M
P U
T I
N G
C O
M
P U
T I
N G
C O
P U
I I
G N
C O
M
C M O
I P T U
C I M O P T U
C G I M N O P T U

12. b. ii Using Strassen's algorithm, compute

$$C = \left(\begin{array}{cc:cc} 1 & 3 & 2 & 1 \\ 4 & 2 & 1 & 1 \\ \hdashline 0 & 2 & 3 & 2 \\ 5 & 4 & 2 & 1 \end{array}\right) \times \left(\begin{array}{cc:cc} 0 & 2 & 1 & 3 \\ 2 & 1 & 0 & 4 \\ \hdashline 2 & 1 & 3 & 1 \\ 1 & 3 & 5 & 2 \end{array}\right)$$

(A) (B)

$$C = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

where

$$A_{11} = \begin{pmatrix} 1 & 3 \\ 4 & 2 \end{pmatrix} \quad B_{11} = \begin{pmatrix} 0 & 2 \\ 2 & 1 \end{pmatrix}$$

$$A_{12} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} \quad B_{12} = \begin{pmatrix} 1 & 3 \\ 0 & 4 \end{pmatrix}$$

$$A_{21} = \begin{pmatrix} 0 & 2 \\ 5 & 4 \end{pmatrix} \quad B_{21} = \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}$$

$$A_{22} = \begin{pmatrix} 3 & 2 \\ 2 & 1 \end{pmatrix} \quad B_{22} = \begin{pmatrix} 3 & 1 \\ 5 & 2 \end{pmatrix}$$

$C_{11} = A_{11} B_{11} + A_{12} B_{21}$

$C_{12} = A_{11} B_{12} + A_{12} B_{22}$

$C_{21} = A_{21} B_{11} + A_{22} B_{21}$

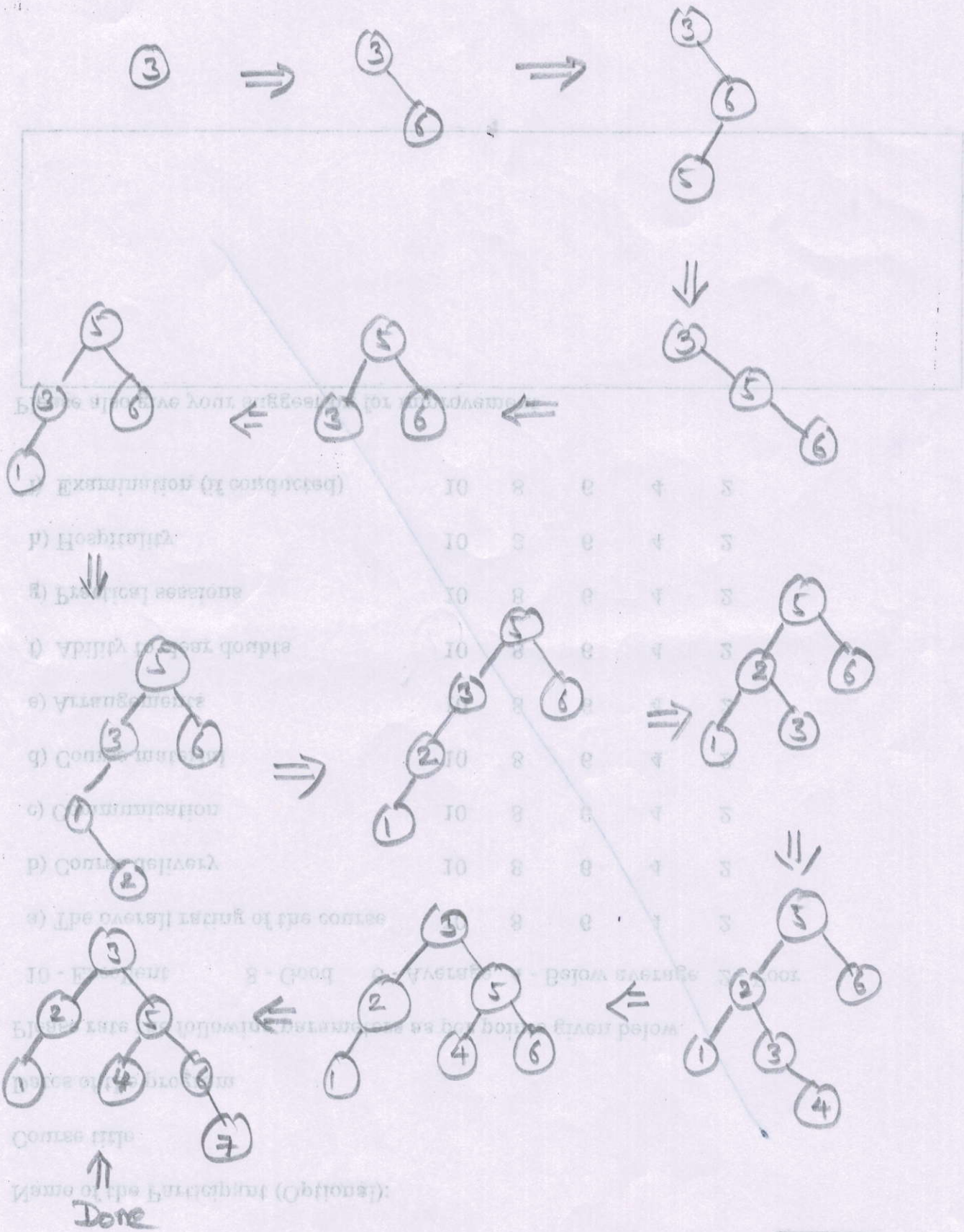
$C_{22} = A_{22} B_{12} + A_{22} B_{22}$

Apply strassens rule to multiply 2x2 matrices.

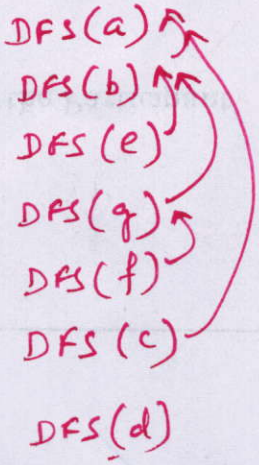
to get- $C = \left[\begin{array}{cc:cc} 11 & 10 & 12 & 19 \\ 7 & 14 & 12 & 23 \\ \hdashline 12 & 11 & 19 & 15 \\ 13 & 19 & 16 & 35 \end{array}\right]$

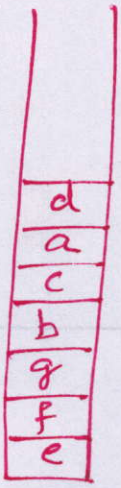
(C_{11} C_{12} / C_{21} C_{22})

13. a. (i)



13 b (i)





stack.

Popping off the stack gives topological ordering

i.e

d a c b g f e

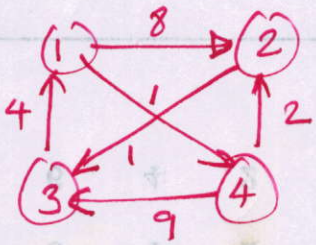
14. a. (i) : Optimal binary search tree

A	B	C	D
0.1	0.2	0.4	0.3

---

14. a. (ii) Solve the all pair shortest path problem for the following weight- matrix

$$D^0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & \infty & 0 & \infty \\ \infty & 2 & 9 & 0 \end{bmatrix} \end{array}$$



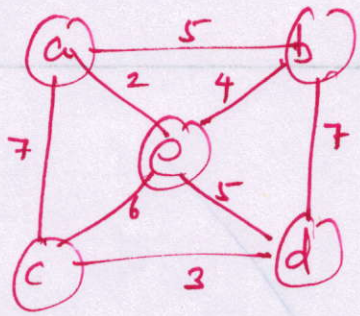
$$D^1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 8 & \infty & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 9 & 0 \end{bmatrix} \end{array} \Rightarrow D^2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 8 & 9 & 1 \\ \infty & 0 & 1 & \infty \\ 4 & 12 & 0 & 5 \\ \infty & 2 & 3 & 0 \end{bmatrix} \end{array}$$

$$D^3 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 8 & 9 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 12 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix} \end{array} \Rightarrow D^4 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 8 & 9 & 1 \\ 5 & 0 & 1 & 6 \\ 4 & 12 & 0 & 5 \\ 7 & 2 & 3 & 0 \end{bmatrix} \end{array}$$

Answer

14. (b) (i)

Find MST using Kruskal's algorithm:

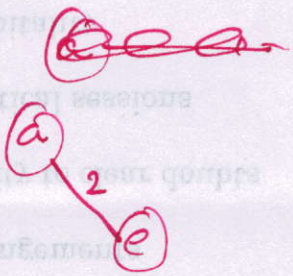


⇒

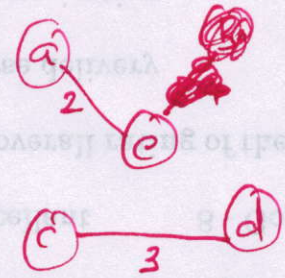
The sorted edges are:

a - e = 2
c - d = 3
b - e = 4
a - b = 5
d - e = 5
c - e = 6
a - c = 7
b - d = 7

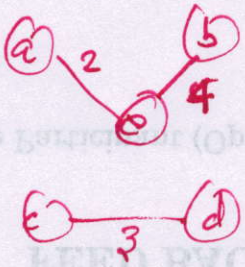
Add : a - e : Detect Cycle : No



Add : c - d : Detect cycle : No

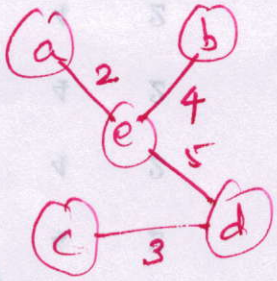


Add b - e : detect cycle : No



Add : a - b : cycle yes skip

add : d - e : cycle No

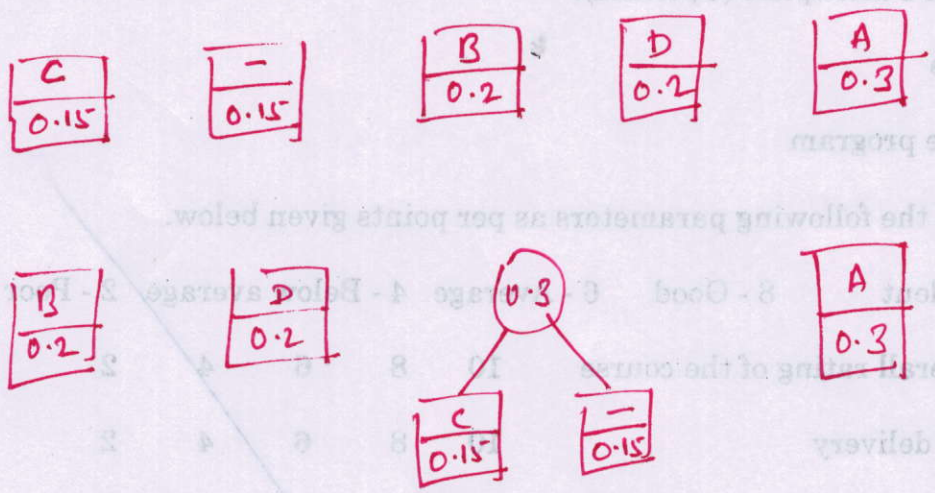


add c - e : cycle yes skip
add a - c : cycle yes skip
add b - d : cycle yes skip

MST :- 14

14 (b) (ii) Huffman code for

A	B	C	D	—
0.3	0.2	0.15	0.2	0.15



Encode ABAC_AB

1400111001011100

A = 11, B = 00

D = 01, C = 100, — = 101

15. a.

Five persons allotted 4 jobs as follows.

Persons	Job			
	J_1	J_2	J_3	J_4
I	86	78	62	81
II	55	79	65	60
III	72	65	63	60
IV	86	70	65	71
V	72	70	71	60

This is a case of unbalanced assignment problem.

Sol.ⁿ Large - Large

Add J_5 with all entries 0.

Small to Small.

Make All enteries of one person as ∞

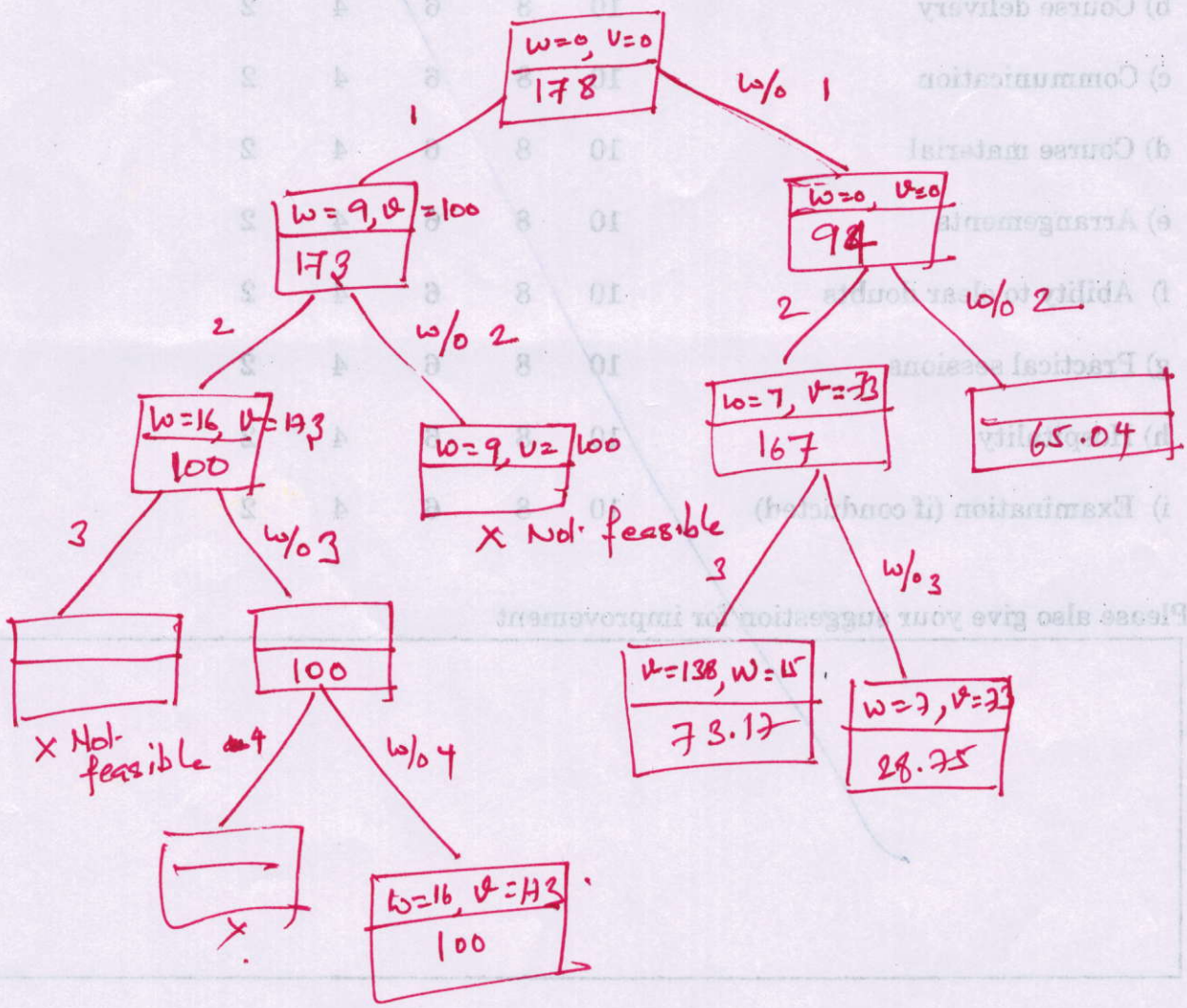
This makes the problem balanced and can be Solved.

15. (b) Knapsack problem using branch & bound.

Item	Weight	Value	V/W
1	9	100	11.11
2	7	73	10.43
3	8	65	8.13
4	4	23	5.75

$W = 16$

Upper bound = $u_b = v + (W - w)\left(\frac{v_{i+1}}{w_{i+1}}\right)$



∴ Value = 173