Sample Question Format
(For all courses having end semester Full Mark=50)

KIIT Deemed to be University
Online End Semester Examination(Autumn Semester-2021)

Subject Name & Code: **Design & Analysis of Algorithms (CS-2012)**

Applicable to Courses:CSE, IT, CSCE, CSSE, ECS

Full Marks=50 **Time:2 Hours**

SECTION-A(Answer All Questions. Each question carries 2 Marks)

Time:30 Minutes **(7×2=14 Marks)**

Question No	Question Type(MCQ/SAT)	Question	CO Mapping	Answer Key (For MCQ Questions only)
Q.No :1	**MCQ**	Let B(n), W(n) and A(n) denote the best case, worst case and average case running time of an algorithmrespectively, executed on an input of size n. Which of the following is always TRUE? A. B(n) = O(W(n)) B. W(n) = Θ(A(n)) C. A(n) = O(B(n)) D. A(n) = Ω(W(n)) E. NONE OF THE OPTION	**A**	**CO1**
	MCQ	Let B(n), W(n) and A(n) denote the best case, worst case and average case running time of an algorithm respectively, executed on an input of size n. Which of the following is always TRUE? A. B(n) = Θ(W(n)) B. W(n) = Ω(A(n)) C. A(n) = O(B(n)) D. B(n) = Ω(W(n)) E. NONE OF THE OPTION	**B**	**CO1**
	MCQ	Let B(n), W(n) and A(n) denote the best case, worst case and average case running time of an algorithm respectively, executed on an input of size n. Which of the following is NOT always TRUE? A. B(n) = O(W(n)) B. W(n) = Ω(A(n)) C. A(n) = Ω(B(n)) D. B(n) = Ω(W(n))	**D**	**CO1**

		E. NONE OF THE OPTION		
	MCQ	Let B(n), W(n) and A(n) denote the best case, worst case and average case running time of an algorithm respectively, executed on an input of size n. Which of the following is NOT always TRUE? A. B(n) = O(W(n)) B. W(n) = Ω(A(n)) C. A(n) = Θ(B(n)) D. W(n) = Θ(W(n)) E. NONE OF THE OPTION	**C**	**CO1**

Q.No :2	**MCQ**	Consider the following function.
		int fun (int n)
		{
		if (n == 0 || n==1)
		return n;
		else
		return 1 + fun(n-1) + fun(n-1);
		}
		What is the least upper bound time complexity of the function fun?
		A. O(n)
		B. O(nlog n)
		C. O(n^2)
		D. O(2^n)
		E. NONE OF THE OPTION
			D	**CO3**

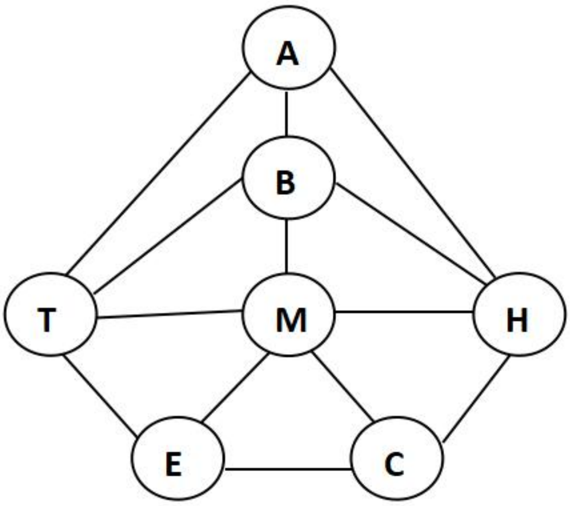
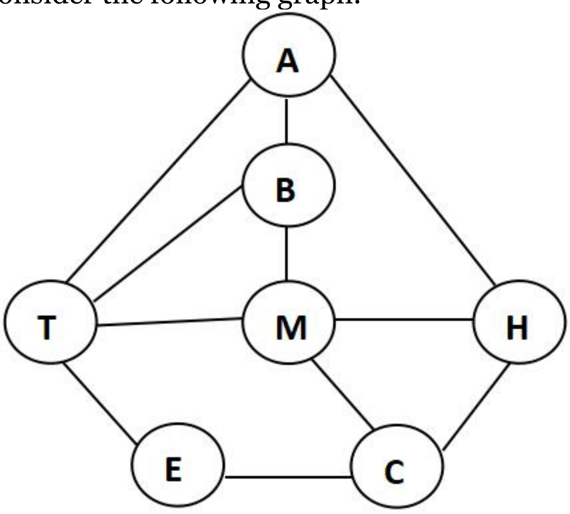
	MCQ	void fun(int n, int A[])
		{
		int i=0, j=0;
		while(i < n){
		while(j < n &&A[i] < A[j])
		{ j++;}
		i++;
		}
		}
		What is the least upper bound time complexity of the function fun?
		A. O(n)
		B. O(nlog n)
		C. O(n^2)
		D. O(2^n)
		E. NONE OF THE OPTION
			A	**CO3**

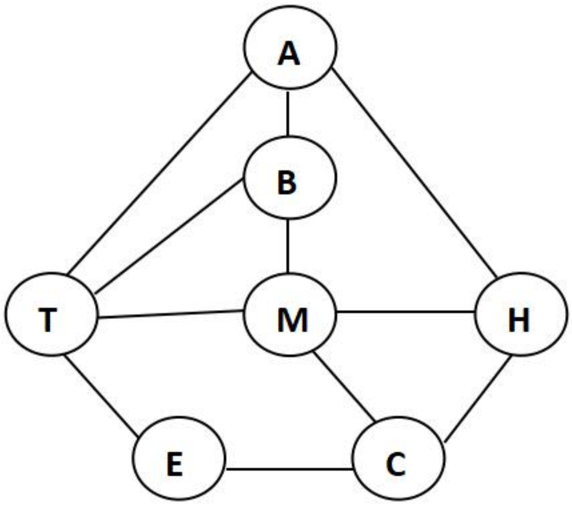
	MCQ	Consider the following C function
		int fun(intn, int A[])
		{
		int i, j, s=0;
		for(i = 1; i<= n; i++)
		{
		for(j=1; j<n; j += i)
		{s = s + A[i];}
		}
		retun s;
		}
		What is the least upper bound time complexity of the function fun?
			B	**CO3**

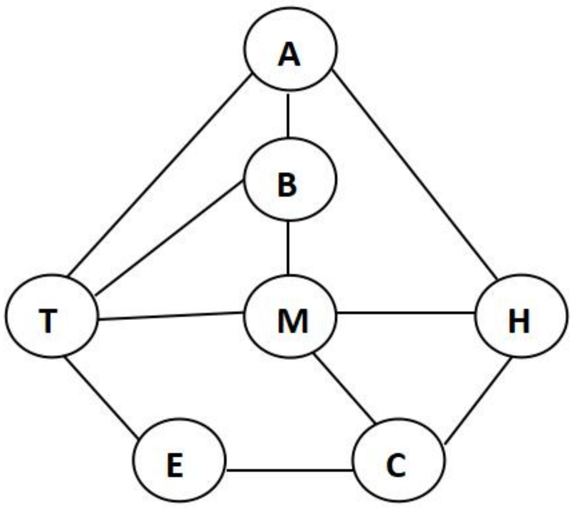
		A. O(n) B. O(nlog n) C. $O(n^2)$ D. $O(2^n)$ E. NONE OF THE OPTION		

	MCQ	Consider the following C function
		int fun(intn, int A[])
		{
		int i, j, s=0;
		for(i = 1; i<= n; i++)
		{
		j = n;
		while (j> 0)
		{
		s= s +j;
		j = j - 2;
		}
		}
		return s;
		}
		
		What is the least upper bound time complexity of the function fun?
		A. O(n)
		B. O(nlog n)
		C. $O(n^2)$
		D. $O(2^n)$
		F. NONE OF THE OPTION
			C	**CO3**

Q.No :3	**MCQ**	Let Array A[1..11]={9, 5, 7, 3, 2, 6, 7, 3, 1, 2, 1} is a max-heap. What will be resultant max-heap, if the value at index 5 is changed to 7. A. {9, 7, 7, 3, 5, 6, 7, 3, 1, 2, 1} B. {9, 5, 7, 3, 3, 6, 7, 3, 1, 2, 1} C. {9, 7, 7, 3, 5, 6, 7, 3, 1, 2, 2} D. {9, 7, 7, 5, 2, 6, 7, 3, 1, 2, 1} E. NONE OF THE OPTION	**A**	**CO4**
	MCQ	Let Array A[1..11]={9, 5, 7, 3, 2, 6, 7, 3, 1, 2, 1} is a max-heap. What will be resultant max-heap, if the value at index 5 is changed to 3. A. {9, 7, 7, 3, 5, 6, 7, 3, 1, 2, 1} B. {9, 5, 7, 3, 3, 6, 7, 3, 1, 2, 1} C. {9, 7, 7, 3, 5, 6, 7, 3, 1, 2, 2} D. {9, 7, 7, 5, 2, 6, 7, 3, 1, 2, 1} E. NONE OF THE OPTION	**B**	**CO4**
	MCQ	Let Array A[1..11]={9, 5, 7, 3, 2, 6, 7, 3, 1, 2, 1} is a max-heap. What will be resultant max-heap, if the value at index 11 is changed to 7. A. {9, 7, 7, 3, 5, 6, 7, 3, 1, 2, 1} B. {9, 5, 7, 3, 3, 6, 7, 3, 1, 2, 1} C. {9, 7, 7, 3, 5, 6, 7, 3, 1, 2, 2} D. {9, 7, 7, 5, 2, 6, 7, 3, 1, 2, 1} E. NONE OF THE OPTION	**C**	**CO4**

	MCQ	Let Array A[1..11]={9, 5, 7, 3, 2, 6, 7, 3, 1, 2, 1} is a max-heap. What will be resultant max-heap, if the value at index 8 is changed to 7. A. {9, 7, 7, 3, 5, 6, 7, 3, 1, 2, 1} B. {9, 5, 7, 3, 3, 6, 7, 3, 1, 2, 1} C. {9, 7, 7, 3, 5, 6, 7, 3, 1, 2, 2} D. {9, 7, 7, 5, 2, 6, 7, 3, 1, 2, 1} E. NONE OF THE OPTION	**D**	**CO4**
Q.No :4	**MCQ**	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) BMATHEC ii) MTECHBA iii) ABTHCME iv)ECTMHBA A. i, ii & iii B. ii & iii C. i, ii & iv D. i & iii E. NONE OF THE OPTION	**C**	**CO4**
	MCQ	Consider the following graph.  Among the following sequences, which are possible breadth first traversals of the above CO4graph if the first	**A**	**CO4**

<table>
<tr><td></td><td></td><td>symbol of each sequence is considered as start vertex.
i) MBCTHAE ii) MTCHBEA
iii) ABTHCME iv) ETCMHBA
A. i &ii
B. ii & iii
C. iii & iv
D. i & iv
E. NONE OF THE OPTION</td><td></td><td></td></tr>
<tr><td></td><td>MCQ</td><td>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) MBCTHEA ii) MTCHBEA
iii) ABTHCME iv) ETCMBAH
A. i&ii
B. ii & iii
C. iii & iv
D. ii & iv
E. NONE OF THE OPTION</td><td>D</td><td>CO4</td></tr>
<tr><td></td><td>MCQ</td><td>Consider the following graph.


Among the following sequences, which are possible</td><td>D</td><td>CO4</td></tr>
</table>

		breadth first traversals of the above graph if the first symbol of each sequence is considered as start vertex. i) MBCTHEA ii) MTCHBEA iii) HCMAETB iv)HCETMBA A. i &ii B. iii & iv C. i & iv D. ii & iii E. NONE OF THE OPTION		
Q.No :5	**MCQ**	Let A1, A2, A3 and A4 be four matrices of dimensions 3x2, 2x4, 4x5, 5x6 respectively. The number of scalar multiplications required to find the product like ((A1A2)A3)A4) is _______ using the basic matrix multiplication method. A. 165 B. 170 C. 174 D. 180 E. NONE OF THE OPTION	**C**	**CO2**
	MCQ	Let A1, A2, A3 and A4 be four matrices of dimensions 6x2, 2x4, 4x5, 5x3 respectively. The number of scalar multiplications required to find the product like ((A1A2)(A3A4)) is _______ using the basic matrix multiplication method. A. 165 B. 170 C. 174 D. 180 E. NONE OF THE OPTION	**D**	**CO2**
	MCQ	Let A1, A2, A3 and A4 be four matrices of dimensions 3x2, 2x4, 4x5, 5x5 respectively. The number of scalar multiplications required to find the product like (A1(A2(A3A4))) is _______ using the basic matrix multiplication method. A. 165 B. 170 C. 174 D. 180 E. NONE OF THE OPTION	**B**	**CO2**
	MCQ	Let A1, A2, A3 and A4 be four matrices of dimensions 5x2, 2x4, 4x5, 5x3 respectively. The number of scalar multiplications required to find the product like ((A1(A2A3))A4) is _______ using the basic matrix multiplication method. A. 165 B. 170 C. 174 D. 180 E. NONE OF THE OPTION	**A**	**CO2**
Q.No :6	**MCQ**	Let the problem X ϵ NP. Which of the following statements are TRUE?	**B**	**CO5**

		i. If X satisfies NP-hard condition, problem X is NP-complete. ii. There is a polynomial time algorithm for X. iii. X can be verified in polynomial time. A) i, ii and iii B) i, iii C) ii, iii D) i, ii E) None of the options		
	MCQ	Let the problem A ϵ NP and B ϵ NP Hard. Which of the following statements are TRUE? i. There is a polynomial time algorithm for A. ii. There is no polynomial time algorithm for B. iii. Problem A may belongs to NPC A) i, ii and iii B) i, iii C) ii, iii D) i, ii E) None of the options	**C**	**CO5**
	MCQ	Let the problem X ϵ NP. Which of the following statements are TRUE? i. If X satisfies NP-hard condition, problem X is NP-complete. ii. There is no polynomial time algorithm for X. iii. X cannot be verified in polynomial time. A) i, ii and iii B) i, iii C) ii, iii D) i, ii E) None of the options	**D**	**CO5**
	MCQ	Let the problem X ϵ NP and Y ϵ NPC. Which of the following statements are TRUE? i. Y ϵ NP ii. There is no polynomial time algorithm for X. iii. X can be verified in polynomial time. A) i, ii and iii B) i, iii C) ii, iii	**A**	**CO5**

		D) i, ii E) None of the options		
Q.No :7		Match the following pairs: P. O(log n) i. Worst case Quick Sort Q. O(n) ii. Binary Search R. O(n log n) iii.Best Case Insertion Sort S. $O(n^2)$ iv. Merge Sort v.Linear Search A. P-ii, Q-iii, Q-v, R-iv, S-i, B. P-iii, Q-ii, R-iv, R-i, S-i C. P-i, Q-ii, R-iv, S-iii, S-i D. P-iv, P-v, Q-ii, R-i, S-iii E. NONE OF THE OPTION	**A**	**CO3**
	MCQ	Match the following pairs: P. O(1) i. Best case Insertion Sort Q. O(n) ii. Best case Linear Search R. O(n log n) iii.Worst case Bubble Sort S. $O(n^2)$ iv. Heap Sort v.Merge Sort A. P-ii, Q-iii, R-iv, S-i, S-iii B. P-iii, Q-ii, Q-v, R-iv, S-i C. P-ii, Q-i, R-iv, R-v, S-iii D. P-iv, Q-ii, Q-v, R-i, S-iii E. NONE OF THE OPTION	**C**	**CO3**
	MCQ	Match the following pairs: P. Floyd-Warshall Algorithm i. Divide-and-Conquer Q. Quick Sort ii. Greedy Approach R. Fractional Knapsack problem iii.Dynamic Programming S. O(n) iv. Linear Search v.Best Case Insertion Sort A. P-ii, Q-iii, R-iv, R-v, S-i B. P-iii, Q-i, R-ii, S-iv, S-v C. P-ii, P-v, Q-i, R-iv, S-iii D. P-iv, Q-ii, R-i, S-iii, S-v E. NONE OF THE OPTION	**B**	**CO3**
	MCQ	Match the following pairs: P. Dijkestrals Algorithm i. Divide-and-Conquer Q. LCS ii. Greedy Approach R. Merge Sort iii.Dynamic Programming S. O(nlog n) iv.Merge Sort v.Heap Sort A. P-ii, Q-iii, R-iv, S-iv, S-v B. P-iii, Q-i, R-ii, R-v, S-iv C. P-ii, Q-i, R-iv, S-iv, S-v	**D**	**CO3**

		D. P-ii, Q-iii, R-i, S-iv, S-v E. NONE OF THE OPTION		

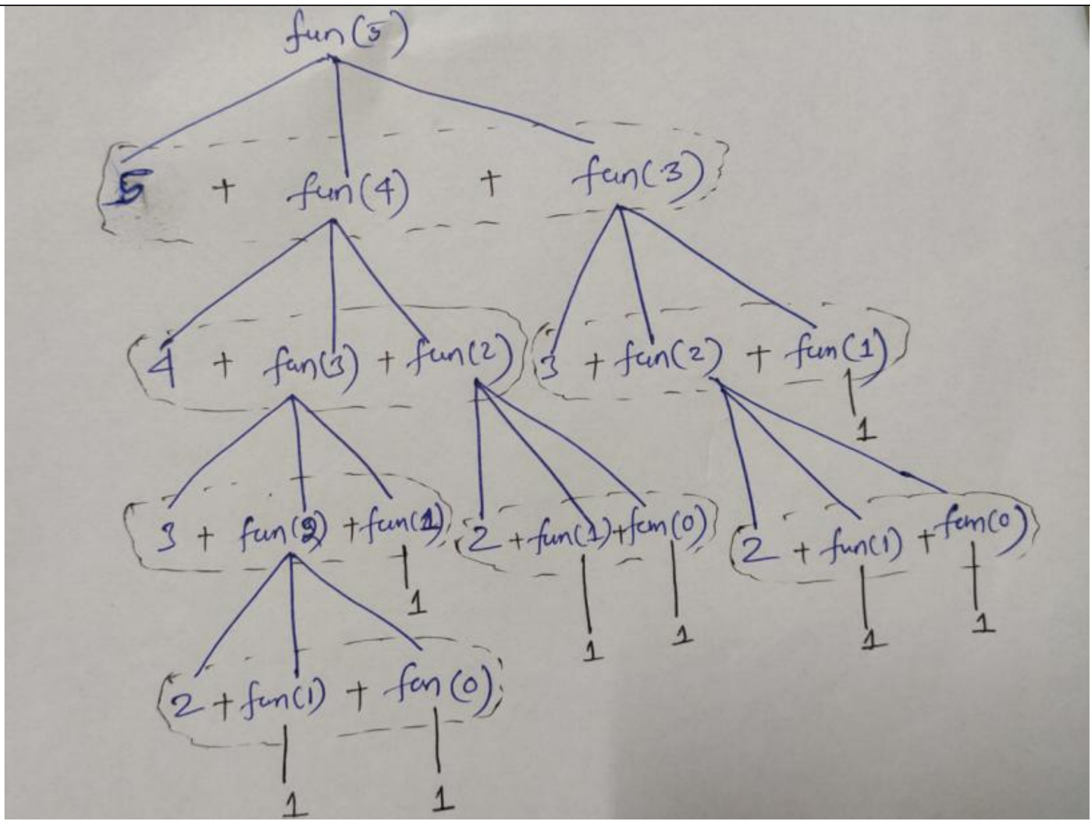
SECTION-B(Answer Any Three Questions. Each Question carries 12 Marks)

Time: 1 Hour and 30 Minutes **(3×12=36 Marks)**

Question No	Question	CO Mapping

Q.No: 8 (a)	Consider the following function
	int fun(int n)
	{
	    if (n<=1)
	      return 1;
	    else
	      return n + fun(n-1) + fun(n-2);
	}
	a) What does the above function compute?
	b) Executing the function for n=5+d%4 (where d is the last digit of your roll number). Draw a recurrence tree to illustrate this fact.
	c) How many additions are performed to compute fun(n), where n=5+d%4 and d represents the last digit of your roll number?
	d) Assuming that each addition/subtraction takes constant time, write a recurrence relation for the running time of fun(n) & solve the recurrence.
		CO1

Q.No: 8 (a) Answer	**Evaluation Scheme** • Ignore bit a. • Drawing of Recurrence Tree: 4 Mark • Number of additions performed by fun(n): 4 Marks • Correct Recurrence relation & its solution: 4 Marks **Solution** n=5+d%4, So possible values for n=5, 6, 7, 8 a) What does the above function compute? **Answer:** No specific Pattern, some modifications in fibonacci sequence. b) Executing the function for n=5+d%4 (where d is the last digit of your roll number). Draw a recurrence tree to illustrate this fact. **Answer:** As, n=5+d%4, So possible values for n=5, 6, 7, 8 **Sample Solution for n=5**	



N.B: Number of function calls in the above where it gets return value after two additions in each function call=7, so total number of addition=7x2=14 for the function call fun(5).

c) How many additions are performed to compute fun(n), where n=5+d%4 and d represents the last digit of your roll number?

Answer: As, n=5+d%4, So possible values for n=5, 6, 7, 8

Function Call	Return Value	Number of Additions performed
fun(5)	29	14
fun(6)	51	24
fun(7)	87	40
fun(8)	146	66

d) Assuming that each addition/subtraction takes constant time, write a recurrence relation for the running time of fun(n) & solve the recurrence.

Answer:

Assuming that each addition taken constant time, the recurrence relation for the running time of fun(n) is as follows:

$T(n) = T(n-1) + T(n-2) + 1$ …………………(I)

Now the recurrence is solved as follows:

Establishing a lower bound by approximating that $T(n-1) \sim T(n-2)$, though $T(n-1) \geq T(n-2)$, hence lower bound, therecurrence eq-1 becomes

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

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

$= 2\{2T(n-4) +1\} + 1$

$= 2^2T(n-4) +3$

$= 2^2\{2T(n-6) +1\} + 3$

$= 2^3T(n-6) + 7$

$= 2^3\{2T(n-8) + 1\} + 7$
$= 2^4T(n-8) + 15$
……
……
……
$= 2^iT(n-2*i) + 2^i-1$

To find out the value of i for which: n - 2*i = 0 => i = n/2
$= 2^{n/2}T(0) + 2^{n/2}-1$
$= 2^{n/2}x1 + 2^{n/2}-1$
$= 2^{n/2} + 2^{n/2}-1 \quad \sim O(2^{n/2})$

Establishing an upper bound by approximating that T(n-2) ~ T(n-1) , though T(n-1) ≥ T(n-2), hence upper bound, the recurrence eq-1 becomes
$T(n) = T(n-1) + T(n-1) + 1$
$= 2T(n-1) + 1$
$= 2\{2T(n-2) +1\} + 1$
$= 2^2T(n-2) +3$
$= 2^2\{2T(n-3) +1\} + 3$
$= 2^3T(n-3) + 7$
$= 2^3\{2T(n-4) + 1\} + 7$
$= 2^4T(n-4) + 15$
……
……
……
$= 2^iT(n-i) + 2^i-1$

To find out the value of i for which: n - i = 0 => i = n
$= 2^nT(0) + 2^n-1$
$= 2^nx1 + 2^n-1$
$= 2^n + 2^n-1 \quad \sim O(2^n)$

Hence, **the time complexity of function fun() in worst case = O(2^n)**

Q.No: 8 (b)

Consider the following function

```
int fun( int n)
{
   if (n<=1)
     return 1;
   else
     return fun(n-1) + fun(n-2) - n;
}
```

a) What does the above function compute?
b) Executing the function for n=5+d%3 (where d is the last digit of your roll number). Draw a recurrence tree to illustrate this fact.
c) How many additions& subtractions are performed to compute fun(n), where n=5+d%4 and d represents the last digit of your roll number?
d) Assuming that each addition/subtraction takes constant time, write a recurrence relation for the running time of fun(n) & solve the recurrence.

Q.No: 8 (b) Answer

Evaluation Scheme

- Ignore bit a.
- Drawing of Recurrence Tree: 4 Mark
- Number of additions performed by fun(n): 4 Marks

- Correct Recurrence relation & its solution: 4 Marks

Solution

n=5+d%3, So possible values for n=5, 6, 7

a) What does the above function compute?

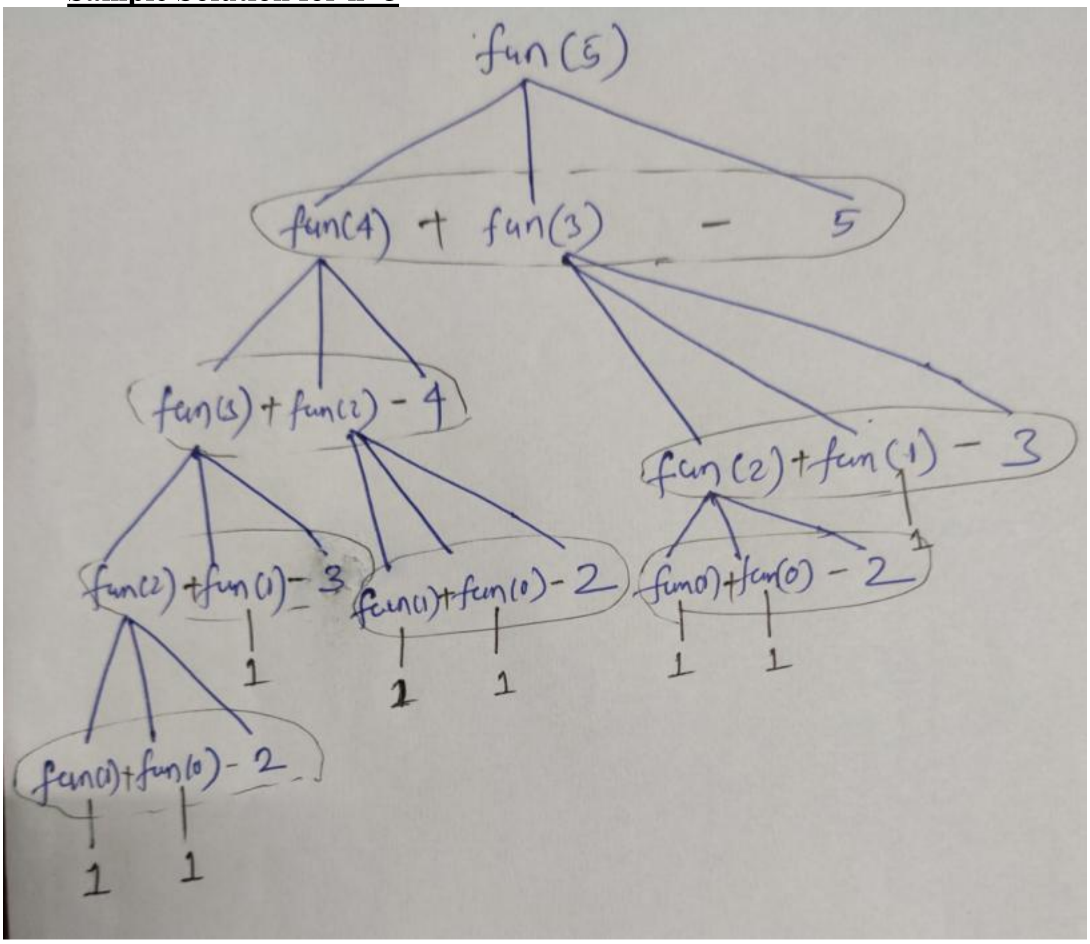
Answer: No specific Pattern, some modifications in fibonacci sequence.

b) Executing the function for n=5+d%3 (where d is the last digit of your roll number). Draw a recurrence tree to illustrate this fact.

Answer:

As, n=5+d%3, So possible values for n=5, 6, 7

Sample Solution for n=5



N.B: Number of function calls in the above where it gets return value after two additions in each function call=7, so total number of addition=7x2=14 for the function call fun(5).

c) How many additions are performed to compute fun(n), where n=5+d%4 and d represents the last digit of your roll number?

Answer: As, n=5+d%4, So possible values for n=5, 6, 7 & 8

Function Call	Return Value	Number of Additions performed
fun(5)	-13	14
fun(6)	-25	24
fun(7)	-45	40
fun(8)	-78	66

d) Assuming that each addition/subtraction takes constant time, write a recurrence relation for the running time of fun(n) & solve the recurrence.

	Answer: Assuming that each addition taken constant time, the recurrence relation for the running time of fun(n) is as follows: $T(n) = T(n-1) + T(n-2) + 1$ …………………(I) Now the recurrence is solved as follows: Establishing a lower bound by approximating that $T(n-1) \sim T(n-2)$, though $T(n-1) \geq T(n-2)$, hence lower bound, the recurrence eq-1 becomes $T(n) = T(n-2) + T(n-2) + 1$ $= 2T(n-2) + 1$ $= 2\{2T(n-4) +1\} + 1$ $= 2^2T(n-4) +3$ $= 2^2\{2T(n-6) +1\} + 3$ $= 2^3T(n-6) + 7$ $= 2^3\{2T(n-8) + 1\} + 7$ $= 2^4T(n-8) + 15$ …... …... …... $= 2^iT(n-2*i) + 2^i-1$ To find out the value of i for which: $n - 2*i = 0 \Rightarrow i = n/2$ $= 2^{n/2}T(0) + 2^{n/2}-1$ $= 2^{n/2}x1 + 2^{n/2}-1$ $= 2^{n/2} + 2^{n/2}-1 \sim O(2^{n/2})$ Establishing an upper bound by approximating that $T(n-2) \sim T(n-1)$, though $T(n-1) \geq T(n-2)$, hence upper bound, the recurrence eq-1 becomes $T(n) = T(n-1) + T(n-1) + 1$ $= 2T(n-1) + 1$ $= 2\{2T(n-2) +1\} + 1$ $= 2^2T(n-2) +3$ $= 2^2\{2T(n-3) +1\} + 3$ $= 2^3T(n-3) + 7$ $= 2^3\{2T(n-4) + 1\} + 7$ $= 2^4T(n-4) + 15$ …... …... …... $= 2^iT(n-i) + 2^i-1$ To find out the value of i for which: $n - i = 0 \Rightarrow i = n$ $= 2^nT(0) + 2^n-1$ $= 2^nx1 + 2^n-1$ $= 2^n + 2^n-1 \sim O(2^n)$ Hence, **the time complexity of function fun() in worst case = $O(2^n)$**	
Q.No: 8 (c)	Consider the following function int fun(int n) {	

	if (n<=1)
	  return 1;
	else
	  return fun(n-1) + 2 + fun(n-2);
	}
	a) What does the above function compute?
	b) Executing the function for n=5+d%4 (where d is the last digit of your roll number). Draw a recurrence tree to illustrate this fact.
	c) How many additions& subtractions are performed to compute fun(n), where n=5+d%5 and d represents the last digit of your roll number?
	d) Assuming that each addition/subtraction takes constant time, write a recurrence relation for the running time of fun(n) & solve the recurrence.	

Q.No: 8 (c)
Answer

Evaluation Scheme

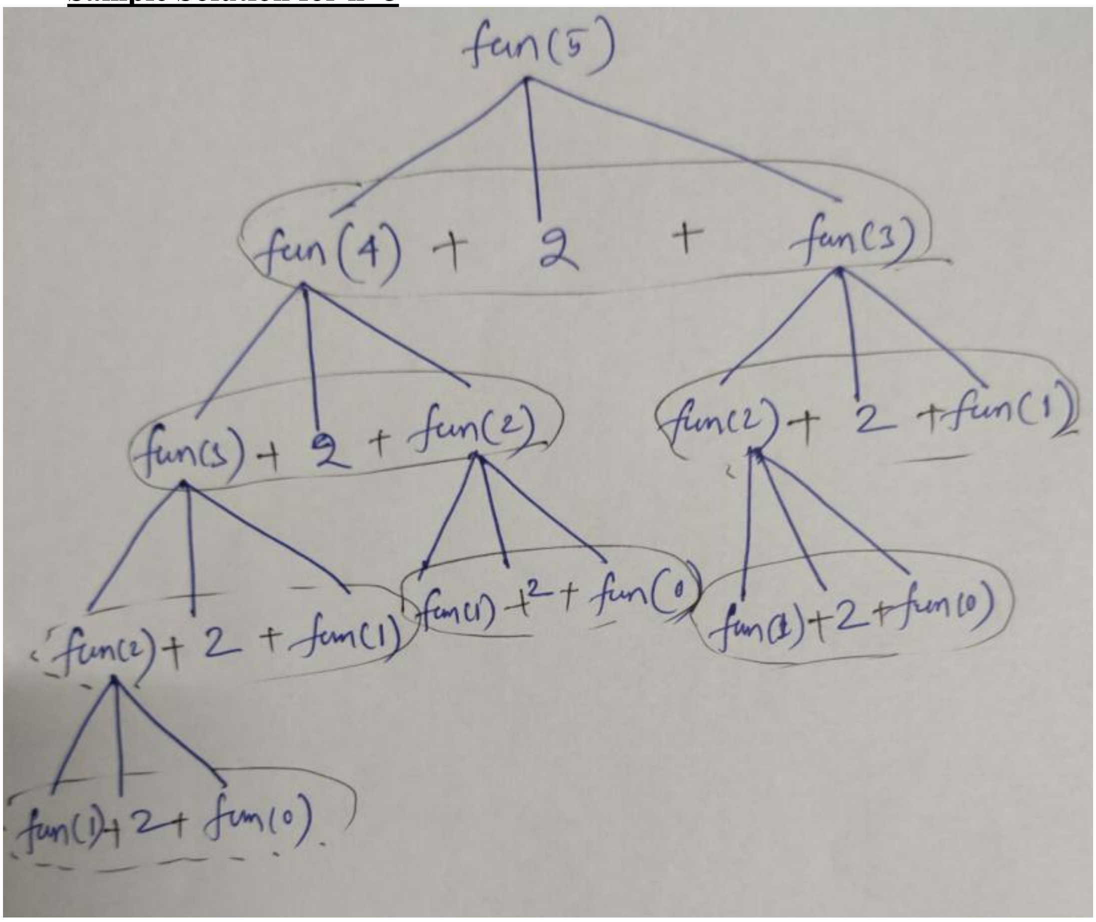
- Ignore bit a.
- Drawing of Recurrence Tree: 4 Mark
- Number of additions performed by fun(n): 4 Marks
- Correct Recurrence relation & its solution: 4 Marks

Solution

n=5+d%4, So possible values for n=5, 6, 7, 8

a) What does the above function compute?
Answer: No specific Pattern, some modifications in fibonacci sequence.

b) Executing the function for n=5+d%4 (where d is the last digit of your roll number). Draw a recurrence tree to illustrate this fact.
Answer:
As, n=5+d%4, So possible values for n=5, 6, 7, 8
Sample Solution for n=5



N.B: Number of function calls in the above where it gets return value after two additions in each function call=7, so total number of addition=7x2=14 for the function call fun(5).

c) How many additions are performed to compute fun(n), where n=5+d%4 and d represents the last digit of your roll number?

Answer: As, n=5+d%5, So possible values for n=5, 6, 7, 8 & 9

Function Call	Return Value	Number of Additions performed
fun(5)	22	14
fun(6)	37	24
fun(7)	61	40
fun(8)	100	66
Fun(9)	163	108

d) Assuming that each addition/subtraction takes constant time, write a recurrence relation for the running time of fun(n) & solve the recurrence.

Answer:

Assuming that each addition taken constant time, the recurrence relation for the running time of fun(n) is as follows:

$T(n) = T(n-1) + T(n-2) +1$ …………………(I)

Now the recurrence is solved as follows:

Establishing a lower bound by approximating that $T(n-1) \sim T(n-2)$, though $T(n-1) \geq T(n-2)$, hence lower bound, therecurrence eq-1 becomes

$T(n) = T(n-2) + T(n-2) + 1$
$= 2T(n-2) + 1$
$= 2\{2T(n-4) +1\} + 1$
$= 2^2T(n-4) +3$
$= 2^2\{2T(n-6) +1\} + 3$
$= 2^3T(n-6) + 7$
$= 2^3\{2T(n-8) + 1\} + 7$
$= 2^4T(n-8) + 15$
…...
…...
…...
$= 2^iT(n-2*i) + 2^i-1$

To find out the value of i for which: $n - 2*i = 0 \Rightarrow i = n/2$
$= 2^{n/2}T(0) + 2^{n/2}-1$
$= 2^{n/2}x1 + 2^{n/2}-1$
$= 2^{n/2} + 2^{n/2}-1 \sim O(2^{n/2})$

Establishing an upper bound by approximating that $T(n-2) \sim T(n-1)$, though $T(n-1) \geq T(n-2)$, hence upper bound, the recurrence eq-1 becomes

$T(n) = T(n-1) + T(n-1) + 1$
$= 2T(n-1) + 1$
$= 2\{2T(n-2) +1\} + 1$
$= 2^2T(n-2) +3$
$= 2^2\{2T(n-3) +1\} + 3$
$= 2^3T(n-3) + 7$

	$= 2^3\{2T(n-4) + 1\} + 7$ $= 2^4T(n-4) + 15$ …… …… …… $= 2^iT(n-i) + 2^i-1$ To find out the value of i for which: $n - i = 0 \Rightarrow i = n$ $= 2^nT(0) + 2^n-1$ $= 2^n x1 + 2^n-1$ $= 2^n + 2^n-1 \quad \sim O(2^n)$ Hence, **the time complexity of function fun() in worst case = $O(2^n)$**	
Q.No: 9 (a)	There is a set of n activities with their start and finish times. Assume that the activities are arranged in non-decreasing order of their finish time. Write an algorithm for Activity Selection. The algorithm must give priority in choosing longest duration activity in case of more than one activity having same finish time.	**CO5**

Q.No: 9 (a) Answer	**Evaluation Scheme**
	● Algorithm for Modified Activity Selection Problem: 12 Marks
	● Function Call used only, no definition for each function call, 2 marks will be deducted.
	
	Solution
	/*Algorithm for Activity Selection Problem: Each activity ai is defined by a pair consisting of a start time si and a finish time fi, where $0 \leq si < fi < \infty$. The activities are arranged in non-decreasing order of their finish time. */
	GREEDY-ACTIVITY-SELECTOR(s, f)
	{
	n ← length[s]
	A ← $\{a_1\}$
	i ← 1
	for m ← 2 to n
	{
	if $s_m \geq$ fi
	{
	A ← A U $\{a_m\}$
	i ← m
	}
	}
	return A
	}
	
	/*Modified Algorithm for Activity Selection Problem:*/
	MODIFIED-GREEDY-ACTIVITY-SELECTOR(s, f)
	{
	/*Sort the activities in non-decreasing order of their start time for the activities that have same finish time*/
	for(i=1; i<n; i=k+1)
	{	

```
            k=i;
            while(f(i)==f(k+1))
            {
                k=k+1;
            }
            if(k!=i)
                SORT-ASCENDING(s,f, i, k);
        }
        //Call the base Activity Selection Algorithm
        GREEDY-ACTIVITY-SELECTOR(s, f)
}

/*Sorting Algorithm to sort data in ascending order*/
INSERTION-SORT-ASCENDING(s,f, lb,ub)
{
        for j←lb+1 to ub
        {
                key←s[j]
                //Insert s[j] into the sorted sequence s[lb..j-1]
                i←j-1
                while(i>=lb and s[i]>key)
                {
                        s[i+1]←s[i]
                        i←i-1
                 }
                 s[i+1]←key
           }
}
```

Q.No:	Question	CO
Q.No: 9 (b)	There is a set of n activities with their start and finish times. Assume that the activities are arranged in non-decreasing order of their finish time. Write an algorithm for Activity Selection. The algorithm must give priority in choosing shortest duration activity in case of more than one activity having same finish time.	**CO5**

Q.No: 9 (b) Answer

Evaluation Scheme

- Algorithm for Modified Activity Selection Problem: 12 Marks
- Function Call used only, no definition for each function call, 2 marks will be deducted.

Solution

/*Algorithm for Activity Selection Problem: Each activity ai is defined by a pair consisting of a start time si and a finish time fi, where 0≤ si <fi<∞. The activities are arranged in non-decreasing order of their finish time. */

```
GREEDY-ACTIVITY-SELECTOR(s, f)
{
        n ← length[s]
        A ← {a1}
        i ← 1
        for m ← 2 to n
        {
            if sm ≥ fi
```

<table>
<tr><td></td><td>

```
        {
          A ← A U {a_m}
          i ← m
        }
    }
    return A
}

/*Modified Algorithm for Activity Selection Problem:*/
MODIFIED-GREEDY-ACTIVITY-SELECTOR(s, f)
{
    /*Sort the activities in non-increasing of their start time for the activities
      that have same finish time*/
    for(i=1; i<n; i=k+1)
    {
        k=i;
        while(f(i)==f(k+1))
        {
            k=k+1;
        }
        if(k!=i)
            SORT-DESCENDING(s,f, i, k);
    }
    //Call the base Activity Selection Algorithm
    GREEDY-ACTIVITY-SELECTOR(s, f)
}

/*Sorting Algorithm to sort data in ascending order*/
INSERTION-SORT-DESCENDING(s,f, lb,ub)
{
    for j←lb+1 to ub
    {
          key←s[j]
          //Insert s[j] into the sorted sequence s[lb..j-1]
          i←j-1
          while(i>=lb and s[i]<key)
          {
                s[i+1]←s[i]
                i←i-1
           }
           s[i+1]←key
        }
}
```

</td><td></td></tr>
<tr><td>Q.No: 9 (c)</td><td>There is a set of n activities with their start and finish times. Assume that the activities are arranged in non-decreasing order of their finish time. Write an algorithm for Activity Selection. The algorithm must give priority in choosing the late start activity in case of more than one activity having same finish time.</td><td>CO5</td></tr>
<tr><td>Q.No: 9 (c)
Answer</td><td>Evaluation Scheme

- Algorithm for Modified Activity Selection Problem: 12 Marks
- Function Call used only, no definition for each function call, 2 marks will be

</td><td></td></tr>
</table>

deducted.

Solution

/*Algorithm for Activity Selection Problem: Each activity ai is defined by a pair consisting of a start time si and a finish time fi, where 0≤ si <fi<∞. The activities are arranged in non-decreasing order of their finish time. */

```
GREEDY-ACTIVITY-SELECTOR(s, f)
{
        n ← length[s]
        A ← {a1}
        i ← 1
        for m ← 2 to n
        {
             if sm ≥ fi
             {
                A ← A U {am}
                i ← m
             }
        }
        return A
}
```

/*Modified Algorithm for Activity Selection Problem:*/

```
MODIFIED-GREEDY-ACTIVITY-SELECTOR(s, f)
{
        /*Sort the activities in non-increasing of their start time for the activities
           that have same finish time*/
        for(i=1; i<n; i=k+1)
        {
             k=i;
             while(f(i)==f(k+1))
             {
                  k=k+1;
             }
             if(k!=i)
                  SORT-DESCENDING(s,f, i, k);
        }
        //Call the base Activity Selection Algorithm
        GREEDY-ACTIVITY-SELECTOR(s, f)
}
```

/*Sorting Algorithm to sort data in ascending order*/

```
INSERTION-SORT-DESCENDING(s,f, lb,ub)
{
        for j←lb+1 to ub
        {
                key←s[j]
                //Insert s[j] into the sorted sequence s[lb..j-1]
                i←j-1
                while(i>=lb and s[i]<key)
                {
```

```
                s[i+1]←s[i]
                i←i-1
            }
            s[i+1]←key
        }
}
```

Q.No: 10 (a) | CO2

Suppose a file to be transferred through the network contains the following characters with their number of occurrences as < a: 10, b: 25, c: 5, d: 30, e: 20 >. Determine an efficient strategy that can minimize the total cost of transferring that file of 1000 characters. Find out the total cost of transfer if transferring cost for 1-bit of data is 4 units.

Q.No: 10 (a) Answer

Evaluation Scheme

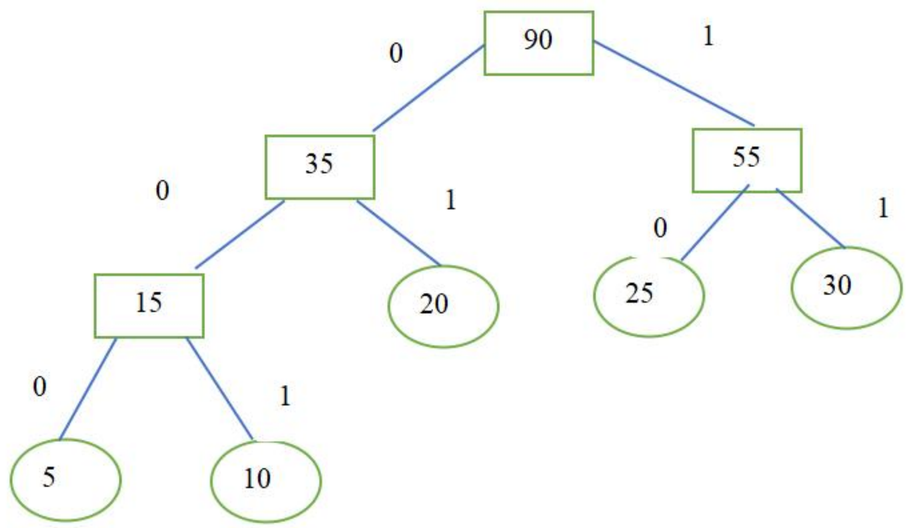
- Huffman tree construction with the given data: 8 Marks
- Comparing Huffman with fixed code: 4 Marks

Solution

The number of occurrence of characters in the file to be transferred is

< a:10, b:25, c:5, d:30, e:20 >.

The constructed Huffman tree is



Huffman code for each character is: <a: 001, b: 10, c:000, d:11, e:01>

Average code length = $\sum$ (frequencyi $*$ code lengthi) /$\sum$ $frequencyi$

= (10*3 + 25*2 + 5*3 + 30*2 + 20*2)/ (10 + 25 + 5 + 30 +20) = 195/90 = 2.166

Size of the message to be transferred is 1000 character.

Total bit in the Huffman encoded message = 2.166 * 1000 = 2166

Cost per bit is 4 unit.

Cost of Huffman coded message is 2166 * 4 = 8664

Q.No: 10 (b) | CO2

State and explain the Longest Common Subsequence problem. Determine an LCS of the given two sequences < a, a, b, a, b, a, b, a > and <b, a, b, a, b, b, a, a, b>.

Q.No: 10 (b)

Evaluation Scheme

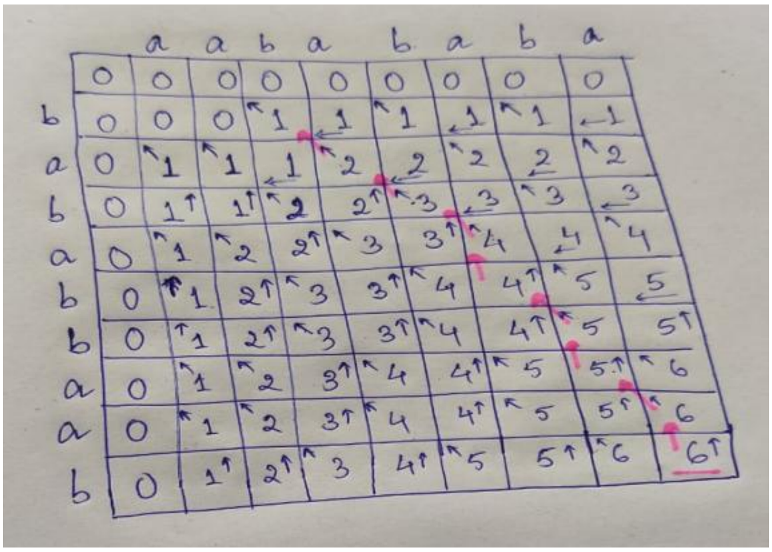
- Explanation of LCS with algorithm: 5 Marks

Answer

- Construction of LCS as per algorithm: 7 Marks

Solution

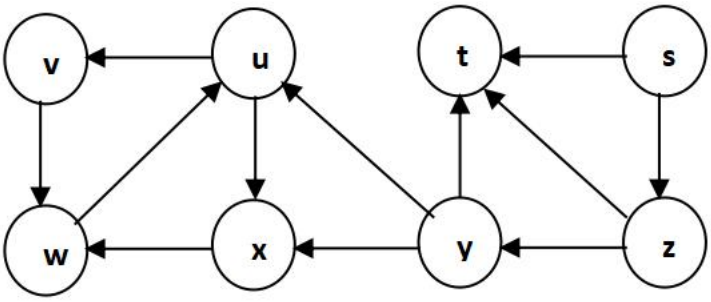
LCS: bababa



Q.No: 10 (c)

Construct the adjacency list of the following directed graph and demonstrate the DFS (depth-first search) algorithm on it. Write the initialization and explain how the relevant parameters and data structures are updated during the execution. In the final step, you should write the DFS tree/trees, and also the forward edges, cross edges, and back edges, if any. Use node 'v' as source node while answering the question.

CO2



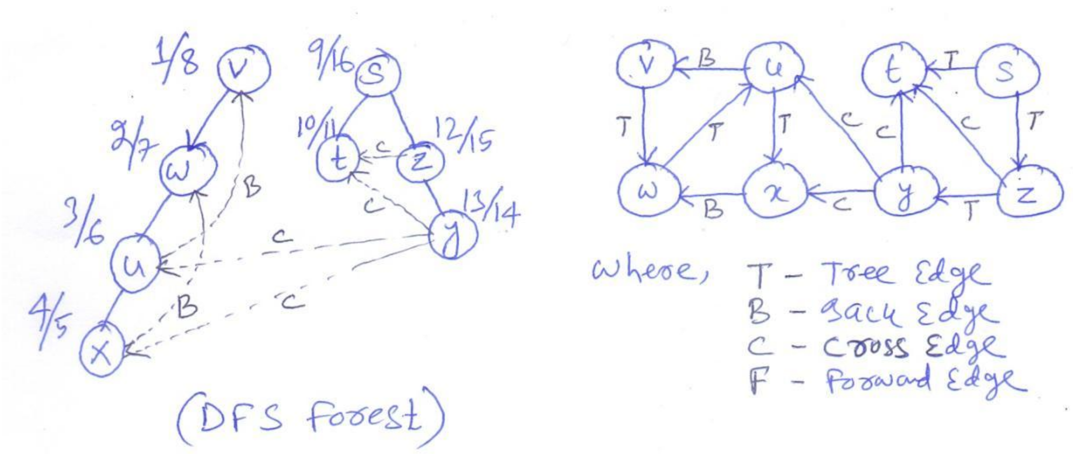
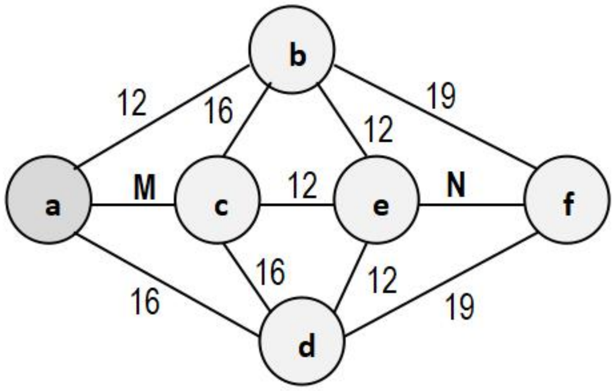
Q.No: 10 (c) Answer

Evaluation Scheme

- Construction of the adjacency list of the given directed graph: 2 Marks
- Explanation of DFS algorithm through DFS tree/forest: 6 Marks
- Explanation/marking of tree edges, forward edges, cross edges, and back edges on final step of the DFS tree/forest or on the given graph: 4 Marks

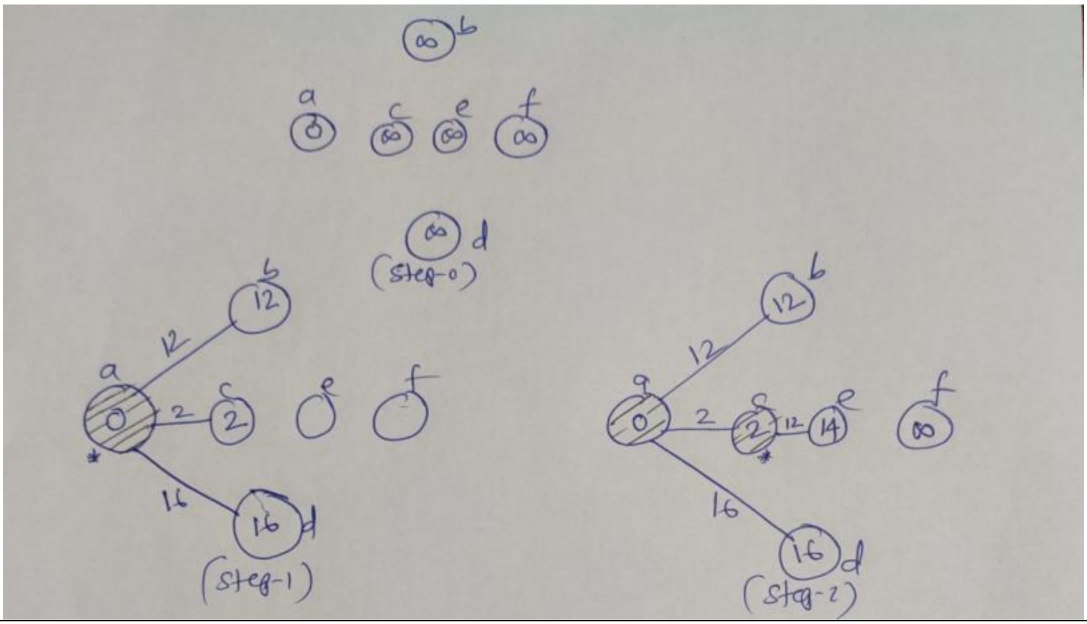
Solution

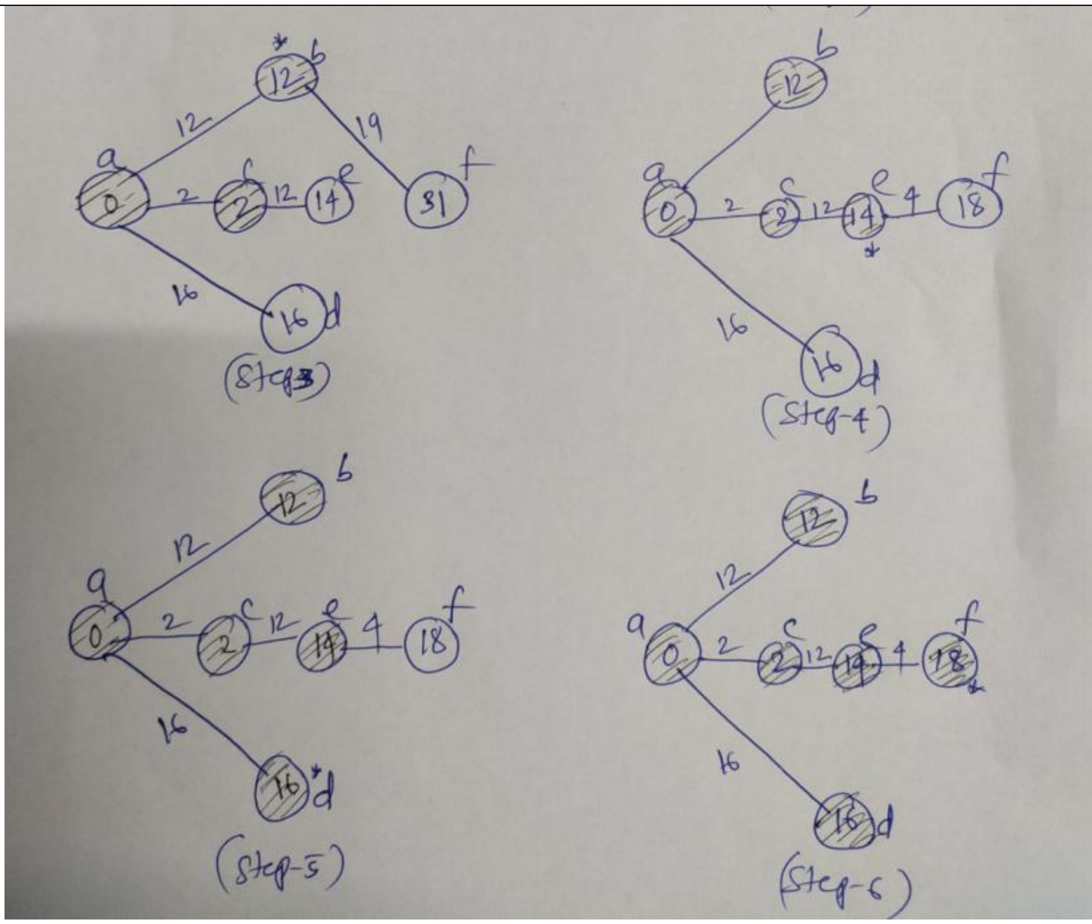
Sl. No.	Vertex	Adjancy list
1	s	t, z
2	t	
3	u	v, x
4	v	w
5	w	u
6	x	w
7	y	u, x
8	z	t, y

<table>
<tr><td></td><td>
</td><td></td></tr>
<tr><td>Q.No: 11 (a)</td><td>Apply Dijkstra’s algorithm to find the shortest path (values) from vertex ‘a’ to rests of the vertices. Write down the contents of the data structure used to solve the above problem after finding shortest distance to a new vertex. Show the shortest path to all destinations.
(Hints M = Your Roll Number % 5 and N = M * 2)

</td><td>CO3</td></tr>
<tr><td>Q. No : 11 (a)
Answer</td><td>Evaluation Scheme
● Execute Dijkstra’s algorithm through step by step diagram: 12 Marks

Solution
M = Your Roll Number % 5 => Possible values of M=0, 1, 2, 3, 4
So, N=0, 2. 6, 8
Sample Solution for M=2, So N=4 with ‘a’ as start vertex</td><td></td></tr>
</table>

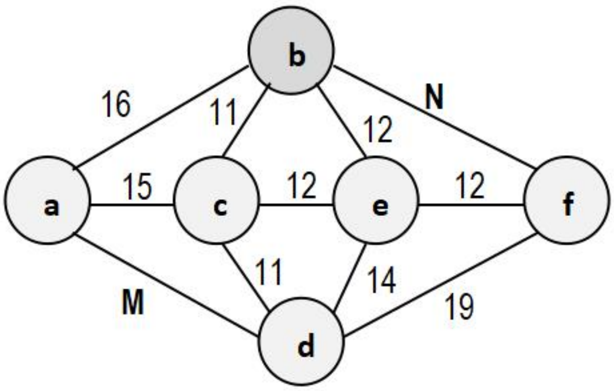
- circle marked with lines is removed from priority queue.
- Number inside circle represents the min. distance so far from start vertex.





Q.No: 11 (b)

Apply Dijkstra's algorithm to find the shortest path (values) from vertex 'b' to rests of the vertices. Write down the contents of the data structure used to solve the above problem after finding shortest distance to a new vertex. Show the shortest path to all destinations.
(Hints M = Your Roll Number % 5 and N = M * 2)



CO3

Q.No: 11 (b) Answer

Evaluation Scheme

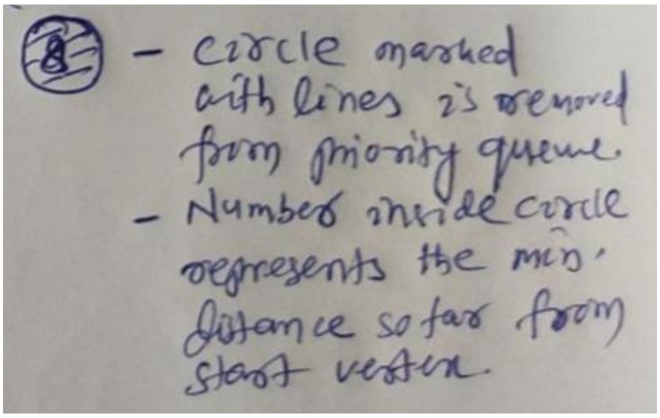
- Execute Dijkstra's algorithm through step by step diagram: 12 Marks

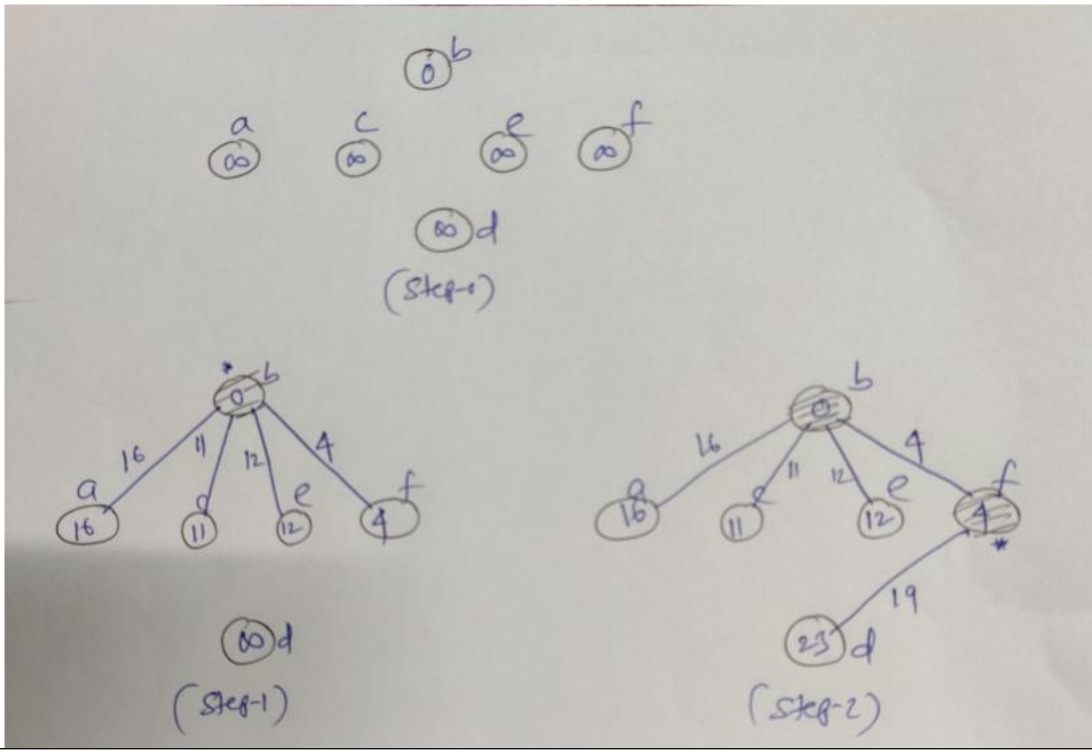
Solution

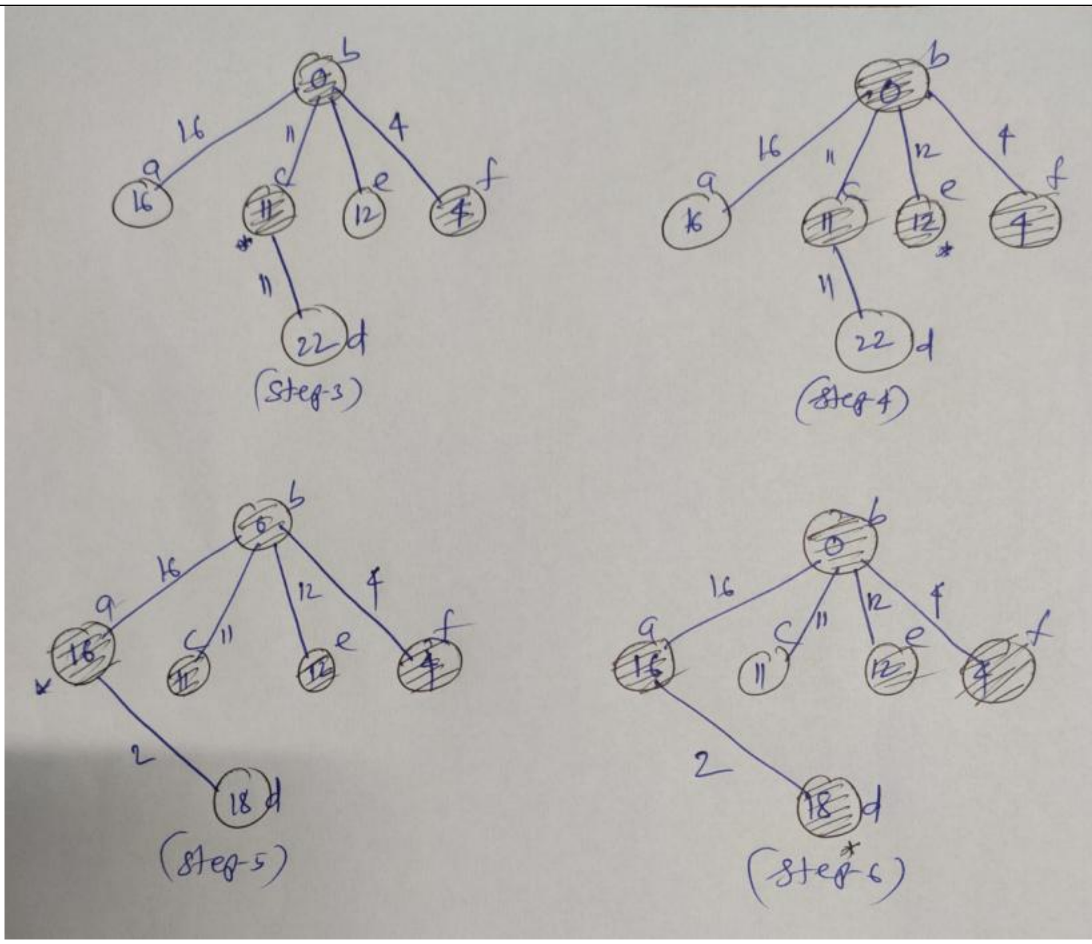
M = Your Roll Number % 5 => Possible values of M=0, 1, 2, 3, 4

So, N=0, 2. 6, 8

Sample Solution for M=2, So N=4 with ‘b’ as start vertex

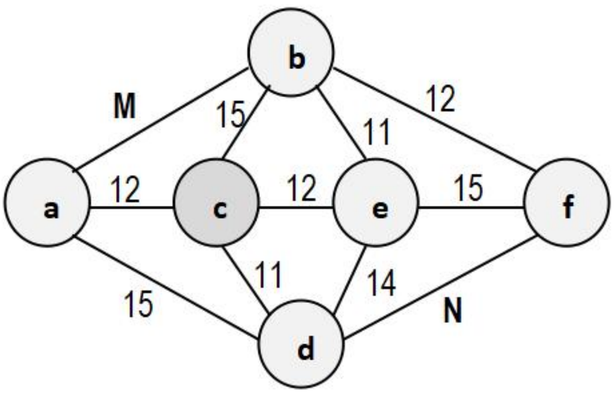

- circle marked with lines is removed from priority queue.
- Number inside circle represents the min. distance so far from start vertex.





Q.No: 11 (c) Answer

Apply Dijkstra's algorithm to find the shortest path (values) from vertex 'c' to rests of the vertices. Write down the contents of the data structure used to solve the above problem after finding shortest distance to a new vertex. Show the shortest path to all destinations.
(Hints M = Your Roll Number % 5 & N = M * 2)



Q.No: 11 (c) Answer

CO3

<u>**Evaluation Scheme**</u>

- Execute Dijkstra's algorithm through step by step diagram: 12 Marks

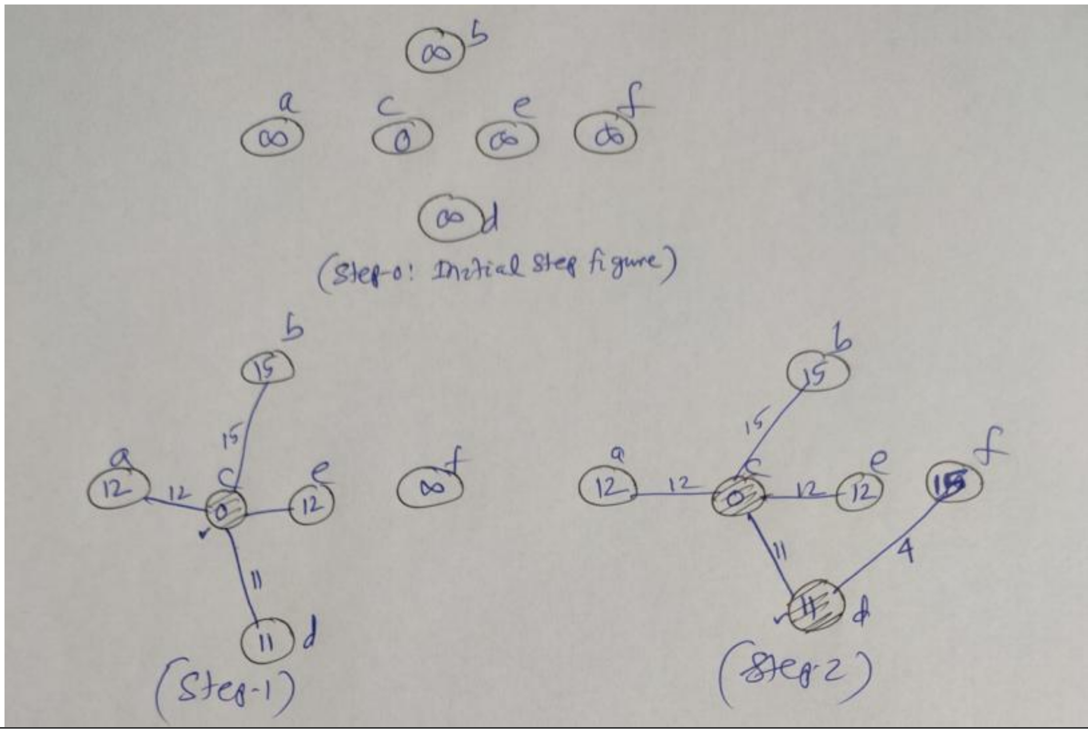
<u>**Solution**</u>

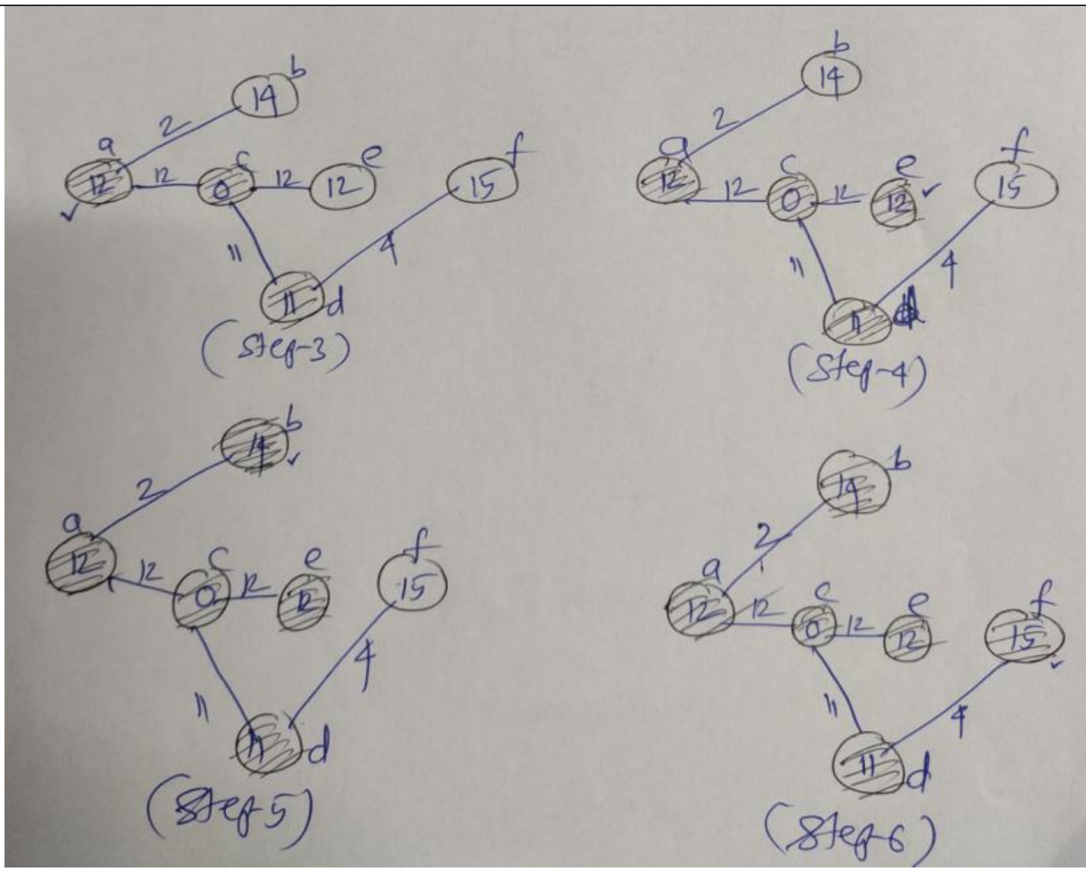
M = Your Roll Number % 5 => Possible values of M=0, 1, 2, 3, 4

So, N=0, 2. 6, 8

Sample Solution for M=2, So N=4 with ‘c’ as start vertex

8 - circle marked with lines is removed from priority queue.
- Number inside circle represents the min. distance so far from start vertex.




(Step-3)
(Step-4)
(Step 5)
(Step 6)