GATE PREP SERIES

GATE

Graduate Aptitude Test in Engineering

2025

Topic-wise
Previous Solved Papers

25 Years'
Solved Papers

Computer Science & Information Technology

G. K. PUBLICATIONS (P) LTD.

Title : **GATE 2025 :** Computer Science and Information Technology - 25 Years' Topic wise Previous Solved Papers

Language : English

Editor's Name **:** GKP Editorial Team

Typeset & Published by :

Career Launcher Infrastructure (P) Ltd.

A-45, Mohan Cooperative Industrial Area, Near Mohan Estate Metro Station, New Delhi - 110044

Marketed by :

G.K. Publications (P) Ltd.

Plot No. 63, Sector-27A, Near Sector - 28 Metro Station, Faridabad, Haryana-121003

ISBN **: 978-93-5681-989-4**

Printer's Details : Printed in India, New Delhi.

For product information :

Visit ***www.gkpublications.com*** or email to ***gkp@gkpublications.com***

GATE
Computer Science &
Information Technology

Contents

Solved Papers (Topic-Wise)

Unit IX Databases

Unit X Computer Networks

Unit XI General Aptitude

7 CHAPTER

Searching & Hashing

MEMORY

1. The most appropriate matching for the following pairs

List-I	List-II
X. Indirect addressing	1. Loops
Y. Immediate addressing	2. Pointers
Z. Auto decrement addressing	3. Constants

(a) X– 3 Y– 2 Z – 1 (b) X – 1 Y – 3 Z – 2
(c) X – 2 Y – 3 Z – 1 (d) X – 3 Y – 1 Z – 2

[2000 :1 Mark]

2. The most appropriate matching for the following pairs

List-I	List-II
X. m=malloc(5); m = NULL;	1. using dangling pointers
Y. free(n); n->value=5;	2. using uninitialized pointers
Z. char *p; *p='a';	3. lost memory

Codes:

(a) X – 1 Y – 3 Z – 2 (b) X – 2 Y – 1 Z – 3
(c) X – 3 Y – 2 Z – 1 (d) X – 3 Y – 1 Z – 2

[2000 :1 Mark]

3. The value of j at the end of the execution of the following C program is_____

```
int incr (int i) {
static int count = 0;
count = count + i;
return (count);
}
main ( ) {
int i,j;
for (i = 0; i <=4; i++)
j = incr(i);
}
```

(a) 10 (b) 4
(c) 6 (d) 7 **[2000 : 2 Marks]**

4. In the C language

(a) at most one activation record exists between the current activation record and the activation record for the main

(b) the number of activation records between the current activation record and the activation record for the main depends on the actual function calling sequence

(c) The visibility of global variables depends on the actual function calling sequence

(d) Recursion requires the activation record for the recursive function to be saved on a different stack before the recursive function can be called. **[2002 : 1 Mark]**

5. Consider the following declaration of a two-dimensional array in C

char a[100] [100];

Assuming that the main memory is byte-addressable and that the array is stored starting from memory address 0, the address of a [40] [50] is

(a) 4040 (b) 4050
(c) 5040 (d) 5050 **[2002 : 2 Marks]**

6. Consider the following C function

```
void swap (int a, int b)
{ int temp ;
temp = a ;
a = b ;
b = temp ;
}
```

In order to exchange the values of two variables x and y.

(a) call swap (x, y)

(b) call swap (&x, &y)

(c) swap (x, y) cannot be used as it does not return any value

(d) swap (x, y) cannot be used as the parameters are passed by value

[2004 :1 Mark]

7. An Abstract Data Type (ADT) is

(a) same as an abstract class

(b) a data type that cannot be instantiated

(c) a data type for which only the operations defined on it can be used, but none else

(d) all of the above

[2005 : 1 Mark]

8. Match the following:

List-I

(P) static char var;

(Q) m = malloc(10); m = NULL;

(R) char *ptr[10];

(S) register int varl;

List-II

(i) Sequence of memory locations to store addresses

(ii) A variable located in data section of memory

(iii) Request to allocate a CPU register to store data

(iv) A lost memory which cannot be freed

(a) P → (ii), Q → (iv), R → (i), S → (iii)

(b) P → (ii), Q → (i), R → (iv), S → (iii)

(c) P → (ii), Q → (iv), R → (iii), S → (i)

(d) P → (iii), Q → (iv), R → (i), S → (ii)

[2017 (Set-2) : 1 Mark]

9. Consider the following C code:

```
#include <stdio.h>
int *assignval (int *x, int val)
{ *x = val;
return x;
}
void main()
{ int *x = malloc (size of (int));
if (NULL == x) return;
x = assignval(x, 0);
if(x)
{ x = (int*) malloc (size of (int));
if (NULL = = x) return;
x = assignval (x, 10);
}
printf ("%d"\n", *x);
free (x);
}
```

The code suffers from which one of the following problems:

(a) compiler error as the return of malloc is not typecast appropriately

(b) compiler error because the comparison should be made as x == NULL and not as shown

(c) compiles successfully but execution may result in dangling pointer

(d) compiles successfully but execution may result in memory leak

[2017 (Set-1) : 1 Mark]

HEAP

10. In a heap with n elements with the smallest element at the root, the 7th smallest element can be found in time

(a) Θ(n log n) (b) Θ(n)

(c) Θ(log n) (d) Θ(1)

[2003 : 1 Mark]

11. A data structure is required for storing a set of integers such that each of the following operations can be done in O(log n) time, where n is the number of elements in the set.

1. Deletion of the smallest element.
2. Insertion of an element if it is not already present in the set.

Which of the following data structures can be used for this purpose?

(a) A heap can be used but not a balanced binary search tree

(b) A balanced binary search tree can be used but not a heap

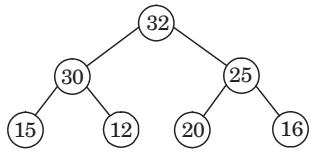
(c) Both balanced binary search tree and heap can be used

(d) Neither balanced binary search tree nor heap can be used

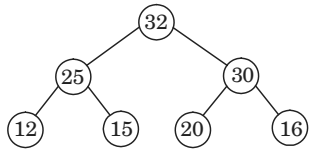
[2003 : 2 Marks]

12. The elements 32, 15, 20, 30, 12, 25, 16 are inserted one by one in the given order into a maxHeap. The resultant maxHeap is

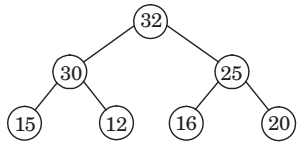
(a)



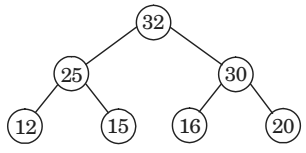
(b)



(c)



(d)



[2004 : 2 Marks]

13. In a binary max heap containing n numbers, the smallest element can be found in time

(a) θ (n) (b) θ (log n)

(c) θ (log log n) (d) θ (1)

[2006 : 1 Mark]

14. Which of the following sequences of array elements forms a heap?

(a) {23, 17, 14, 6, 13, 10, 1, 12, 7, 5}

(b) {23, 17, 14, 6, 13, 10, 1, 5, 7, 12}

(c) {23, 17, 14, 7, 13, 10, 1, 5, 6, 12}

(d) {23, 17, 14, 7, 13, 10, 1, 12, 5, 7}

[2006 : 2 Marks]

Common Data for Q. 15 & Q. 16:

A 3-ary max heap is like a binary max heap, but instead of 2 children, nodes have 3 children. A 3-ary heap can be represented by an array as follows: The root is stored in the first location, a [0], nodes in the next level, from left to right, is stored from a [1] to a [3]. The nodes from the second level of the tree from left to right are stored from a [4] location onward. An item x can be inserted into a 3-ary heap containing n items by placing x in the location a [n] and pushing it up the tree to satisfy the heap property.

15. Which one of the following is a valid sequence of elements in an array representing 3-ary max heap?

(a) 1, 3, 5, 6, 8, 9 (b) 9, 6, 3, 1, 8, 5

(c) 9, 3, 6, 8, 5, 1 (d) 9, 5, 6, 8, 3, 1

[2006 : 2 Marks]

16. Suppose the elements 7, 2, 10, and 4 are inserted, in that order, into the valid 3-ary max heap found in the above question. Which one of the following is the sequence of items in the array representing the resultant heap?

(a) 10, 7, 9, 8, 3, 1, 5, 2, 6, 4

(b) 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

(c) 10, 9, 4, 5, 7, 6, 8, 2, 1, 3

(d) 10, 8, 6, 9, 7, 2, 3, 4, 1, 5 **[2006 : 2 Marks]**

17. Consider the process of inserting an element into a Max Heap, where the Max Heap is represented by an array. Suppose we perform a binary search on the path from the new leaf to the root to find the position for the newly inserted element, the number of comparisons performed is

(a) $\theta(\log_2 n)$ (b) $\theta(\log_2 \log_2 n)$

(c) $\theta(n)$ (d) $\theta(n \log_2 n)$

[2007 : 2 Marks]

18. We have a binary heap on n elements and wish to insert n more elements (not necessarily one after another) into this heap. The total time required for this is

(a) $\Theta(\log n)$ (b) $\Theta(n)$

(c) $\Theta(n \log n)$ (d) $\Theta(n^2)$

[2008 : 2 Marks]

Linked Answer Question 19 and 20:

Consider a binary max-heap implemented using an array

19. Which one of the following array represents a binary max-heap?

(a) {25, 12, 16, 13, 10, 8, 14}

(b) {25, 14, 13, 16, 10, 8, 12}

(c) {25, 14, 16, 13, 10, 8, 12}

(d) {25, 14, 12, 13, 10, 8, 16}

[2009 : 2 Marks]

20. What is the content of the array after two delete operations on the correct answer to the previous question?

(a) {14, 13, 12, 10, 8}

(b) {14, 12, 13, 8, 10}

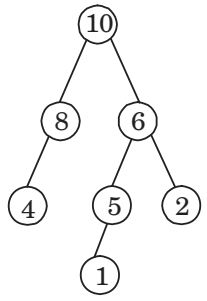
(c) {14, 13, 8, 12, 10}

(d) {14, 13, 12, 8, 10}

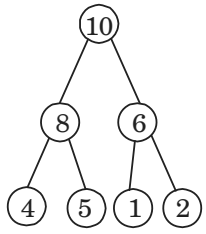
[2009: 2 Marks]

21. A max-heap is a heap where the value of each parent is greater than or equal to the value of its children. Which of the following is a max-heap?

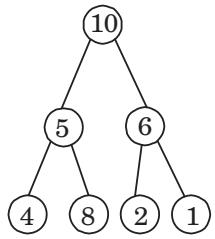
(a)



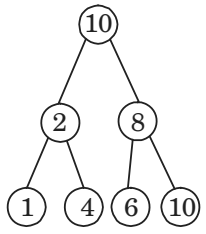
(b)



(c)



(d)



[2011: 1 Mark]

22. A priority queue is implemented as a Max-Heap. Initially, it has 5 elements. The level-order traversal of the heap is: 10, 8, 5, 3, 2. Two new elements 1 and 7 are inserted into the heap in that order. The level-order traversal of the heap after the insertion of the elements is:

(a) 10, 8, 7, 3, 2, 1, 5

(b) 10, 8, 7, 2, 3, 1, 5

(c) 10, 8, 7, 1, 2, 3, 5

(d) 10, 8, 7, 5, 3, 2, 1

[2014 (Set-2) : 1 Mark]

23. Consider a max heap, represented by the array: 40, 30, 20, 10, 15, 16, 17, 8, 4.

Array Index	1	2	3	4	5	6	7	8	9
Value	40	30	20	10	15	16	17	8	4

Now consider that a value 35 is inserted into this heap. After insertion, the new heap is

(a) 40, 30, 20, 10, 15, 16, 17, 8, 4, 35

(b) 40, 35, 20, 10, 30, 16, 17, 8, 4, 15

(c) 40, 30, 20, 10, 35, 16, 17, 8, 4, 15

(d) 40, 35, 20, 10, 15, 16, 17, 8, 4, 30

[2015 (Set-1) : 2 Marks]

24. Consider the following array of elements ('89, 19, 50, 17, 12, 15, 2, 5, 7, 11, 6, 9, 100). The minimum number of interchanges needed to convert it into a max-heap is

(a) 4 (b) 5

(c) 2 (d) 3

[2015 (Set-3) :1 Mark]

25. An operator delete (i) for a binary heap data structure is to be designed to delete the item in the i-th node. Assume that the heap is implemented in an array and i refers to the i-th index of the array. If the heap tree has depth d (number of edges on the path from the root to the farthest leaf), then what is the time complexity to re-fix the heap efficiently after the removal of the element?

(a) O(1)

(b) O(d) but not O(1)

(c) $O(2^d)$ but not O(d)

(d) $O(d2^d)$ but not O(2d)

[2016 (Set-1) : 2 Marks]

26. A complete binary min-heap is made by including each integer in [1, 1023] exactly once. The depth of a node in the heap is the length of the path from the root of the heap to that node. Thus, the root is at depth 0. The maximum depth at which integer 9 can appear is

[2016 (Set-2) : 2 Marks]

27. The number of possible min-heaps containing each value from {1, 2, 3, 4, 5, 6, 7} exactly once is _____

[2018: 2 Marks]

28. An array [82, 101, 90, 11, 111, 75, 33, 131, 44, 93] is heapified. Which one of the following options represents the first three elements in the heapified array?

(a) 131, 111, 90 (b) 131, 11, 93

(c) 82, 11, 93 (d) 82, 90, 101

[2024 (Set-1) : 2 Marks]

29. Consider a binary min - heap containing 105 distinct elements. Let k be the index (in the underlying array) of the maximum element stored in the heap. The number of possible values of k is

(a) 52 (b) 1

(c) 53 (d) 27

[2024 (Set-1) : 2 Marks]

AVL

30. Which of the following is TRUE?

(a) The cost of searching an AVL tree is $\Theta(\log n)$ but that of a binary search tree is O(n)

(b) The cost of searching an AVL tree is $\Theta(\log n)$ but that of a complete binary tree is $\Theta(n \log n)$

(c) The cost of searching a binary search tree is O (log n) but that of an AVL tree is $\Theta(n)$

(d) The cost of searching an AVL tree is $\Theta(n \log n)$ but that of a binaiy search tree is O(n)

[2008: 1 Mark]

31. What is the maximum height of any AVL-tree with 7 nodes? Assume that the height of a tree with a single node is 0.

(a) 2

(b) 3

(c) 4

(d) 5

[2009 : 2 Marks]

LINEAR PROBING

32. A hash table contains 10 buckets and uses linear probing to resolve collisions. The key values are integers and the hash function used is key % 10. If the values 43, 165, 62, 123, 142 are inserted in the table, in what location would the key value 142 be inserted?

(a) 2 (b) 3

(c) 4 (d) 6 **[2005 : 1 Mark]**

33. Consider a hash function that distributes keys uniformly. The hash table size is 20. After hashing of how many keys will the probability that any new key hashed collides with an existing one exceed 0.5.

(a) 5 (b) 6

(c) 7 (d) 10

[2007 : 2 Marks]

34. Consider a hash table of size 11 that uses open addressing with linear probing. Let h(k) = k mod 11 be the hash function used. A sequence of records with keys 43 36 92 87 11 4 71 13 14 is inserted into an initially empty hash table, the bins of which are indexed from zero to ten. What is the index of the bin into which the last record is inserted?

(a) 3 (b) 4

(c) 6 (d) 7

[2008 : 2 Marks]

35. The keys 12, 18, 13, 2, 3, 23, 5 and 15 are inserted into an initially empty hash table of length 10 using open addressing with hash function h(k) = k mod 10 and linear probing. What is the resultant hash table?

(a)

0	
1	
2	2
3	23
4	
5	15
6	
7	
8	18
9	

(b)

0	
1	
2	2
3	13
4	
5	5
6	
7	
8	18
9	

(c)

0	
1	
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

(d)

0	
1	
2	12, 2
3	13, 3, 23
4	
5	5, 15
6	
7	
8	18
9	

[2009 : 2 Marks]

Linked Answer Questions 36 and 37

A has table of length 10 uses open addressing with hash function h(k) = k mod 10, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below.

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

36. Which one of the following choices gives a possible order in which the key values could have been inserted in the table?

(a) 46, 42, 34, 52, 23, 33

(b) 34, 42, 23, 52, 33, 46

(c) 46, 34, 42, 23, 52, 33

(d) 42, 46, 33, 23, 34, 52 **[2010 : 2 Marks]**

37. How many different insertion sequences of the key values using the same hash function and linear probing will result in the hash table shown above?

(a) 10 (b) 20

(c) 30 (d) 40

[2010 : 2 Marks]

38. Consider a hash table with 9 slots. The hash function is h(k) = k mod 9. The collisions are resolved by chaining. The following 9 keys are inserted in the order: 5, 28, 19, 15, 20, 33, 12, 17, 10. The maximum, minimum, and average chain lengths in the hash table, respectively, are

(a) 3, 0, and 1 (b) 3, 3, and 3

(c) 4, 0, and 1 (d) 3, 0, and 2

[2014 (Set-1) : 2 Marks]

39. Consider a hash table with 100 slots. Collisions are resolved using chaining. Assuming simple uniform hashing, what is the probability that the first 3 slots are unfilled after the first 3 insertions?

(a) $(97 \times 97 \times 97)/100^3$

(b) $(99 \times 98 \times 97)/100^3$

(c) $(97 \times 96 \times 95)/100^3$

(d) $(97 \times 96 \times 95)1(3! \times 100^3)$

[2014 (Set-3) : 2 Marks]

HASHING CONCEPT

40. Given the following input (4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199) and the hash function x mod 10, which of the following statements are true?

1. 9679, 1989, 4199 hash to the same value
2. 1471, 6171 hash to the same value
3. All elements hash to the same value
4. Each element hashes to a different value

(a) 1 only (b) 2 only
(c) 1 and 2 only (d) 3 and 4 only

[2004 : 1 Mark]

41. Which of the following statement(s) is TRUE?

I. A hash function takes a message of arbitrary length and generates a fixed length code.
II. A hash function takes a message of fixed length and generates a code of variable length.
III. A hash function may give the same hash value for distinct messages.

(a) I only (b) II and III only
(c) I and III only (d) II only

[2006 : 1 Mark]

42. Consider the hash table of size seven, with starting index zero, and a hash function (3x + 4) mod 7. Assuming the has table is initially empty, which of the following is the contents of the table when the sequence 1, 3, 8,10 is inserted into the table using closed hashing? Note that -denotes an empty location in the table

(a) 8, -, -,-, 10 (b) 1, 8, 10, -, -, 3
(c) 1, -, 3 (d) 1, 10, 8, -, -, 3

[2007 : 2 Marks]

43. Which one of the following hash functions on integers will distribute keys most uniformly over 10 buckets numbered 0 to 9 for i ranging from 0 to 2020?

(a) $h(i) = i^2 \bmod 10$
(b) $h(i) = i^3 \bmod 10$
(c) $h(i) = (11 * i^2) \bmod 10$
(d) $h(i) = (12 * i) \bmod 10$

[2015 (Set-2) : 2 Marks]

44. Given a hash table T with 25 slots that stores 2000 elements, the load factor α for T is ____.

[2015 (Set-3) : 1 Mark]

45. An algorithm has to store several keys generated by an adversary in a hash table. The adversary is malicious who tries to maximize the number of collisions. Let k be the number of keys, m be the number of slots in the hash table, and k > m.

Which one of the following is the best hashing strategy to counteract the adversary?

(a) Division method, i.e., use the hash function h(k) = k mod m.
(b) Multiplication method, i.e., use the hash function $h(k) = \lfloor m(kA - \lfloor kA \rfloor) \rfloor$, where A is a carefully chosen constant.
(c) Universal hashing method.
(d) If k is a prime number, use Division method. Otherwise, use Multiplication method.

[2023 : 1 Mark]

ANSWERS

1. (c)	**2.** (d)	**3.** (a)	**4.** (b)	**5.** (b)	**6.** (d)	**7.** (c)	**8.** (a)	**9.** (d)	**10.** (c)
11. (b)	**12.** (a)	**13.** (a)	**14.** (c)	**15.** (d)	**16.** (a)	**17.** (b)	**18.** (b)	**19.** (c)	**20.** (d)
21. (b)	**22.** (a)	**23.** (b)	**24.** (d)	**25.** (b)	**26.** (8)	**27.** (80)	**28.** (a)	**29.** (c)	**30.** (a)
31. (b)	**32.** (d)	**33.** (d)	**34.** (d)	**35.** (c)	**36.** (c)	**37.** (c)	**38.** (a)	**39.** (a)	**40.** (c)
41. (c)	**42.** (b)	**43.** (b)	**44.** (80)	**45.** (c)					

EXPLANATIONS

1. X-2, Y-3, Z-1

2. X – 3, A pointer is assigned to NULL without freeing memory so it is example of memory leak

Y – 1, Trying to retrieve value after freeing it so dangling pointer.

Z – 2, Using uninitialized pointers

3. At i = 0, j = 0

At i = 1, j = 1

At i = 2, j = 3

At i = 3, j = 6

At i = 4, j = 10

4. (a) False.There is no such restriction in C language

(b) True.

(c) False. In C, variables are statically scoped, not dynamically.

(d) False. The activation records are stored on the same stack.

5. By defalut we are assuming that array store in row major order.

Address of a[40][50]

= Base address + 40 * 100 * element_size
+ 50 * element_size

= 0 + 4000*1 + 50*1

= 4050

6. Since no print & instruction inside the body of the function and function will return nothing in the calling environment since (x, y) cannot be used as, the parameters are passed by value.

7. The abstract data type (ADT) refers to a programmer defined data type together with a set of operations that can be performed on that data so the choice (c) is correct.

8. P → (ii), Q → (iv), R → (i), S → (iii)

9. The output of code is 10.

int * x = malloc (sizeof(int)); //assigns a memory to x.

Now,

(int*) malloc(sizeof(int)); // again assigns a memory to x. previous memory location is lost because now we have no reference to that location resulting in memory leak.

10. To find Kth smallest element the time Requierd is klogn.

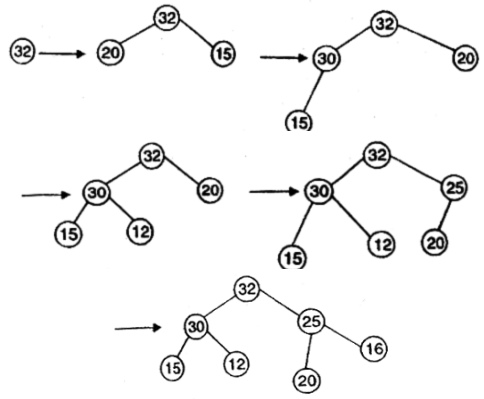
Here k is constant value so time complexity is O(logn).

11. Deletion of the smallest element or randomly any element is insertion or deletion.

Complexity is O $(\log_2 n)$.

12. Max Heap are those tree in which root node has larger value than that of the child and left sibling has larger value than right.

Given numbers are 32, 15, 20, 30, 12, 25, 16 then Max Heap tree will be

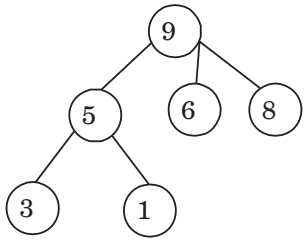


13. MAX Heap used to identity max element in O(1) time for to identity min element require O(n).

14. Option (c) is level order traversal so max heap.

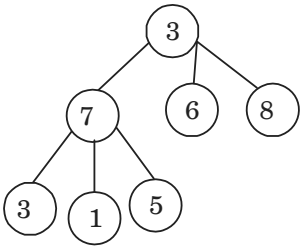
15. For keys 9, 5, 6, 8, 3, 1

3 - ary Max Heap is

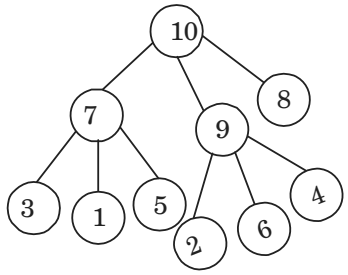


16. After insertion of 7.

Heap is :-



After inserting 2 &1024 resultant Head is
{10, 7, 9, 8, 3, 1, 5 2, 6, 4 } is level order traversal.

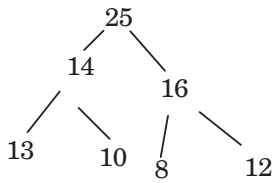


17. The height of a Max Heap is Θ(logn).
If we perform binary search for finding the correct position then we need to do Θ(Log Logn) comparisons.

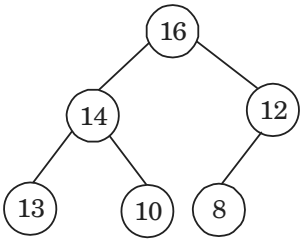
18. Time complexity for build heap operation is O(n) for n elements. If there are 2n elements then still complexity is O(n).

19. Option (c) is level order traversal so max heap.

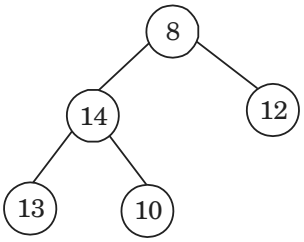
20.

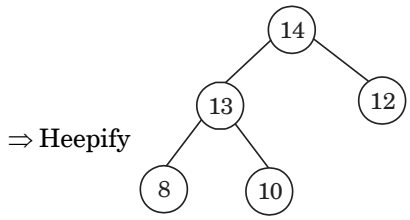


After deleting 25 the Resultant Heap is



Now delete 16 then resultant Heap is

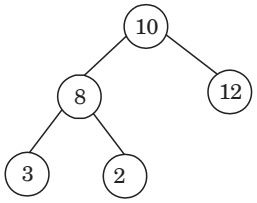




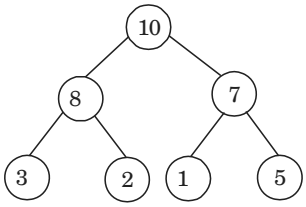
level order traversal of Max Heap is 14, 13, 12, 18, 10.

21. Heap is a complete binary tree.

22. Given level order traversal is 10, 8, 5, 3, 2
So Heap is

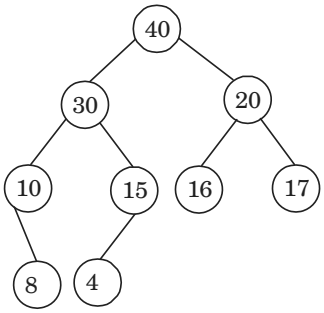


Now insert 1 & 7 after than the resultant Heap is

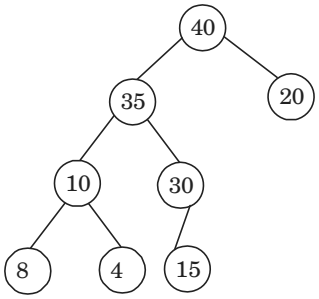


Now, Level order traversal is = 10, 8, 7, 3, 2, 1, 5

23. Given Heap is

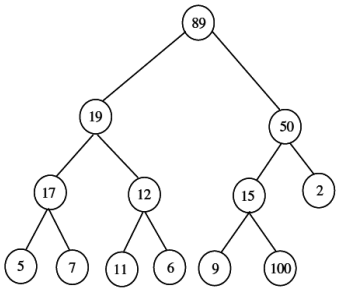


After inserting 35, find result is



So Level order traversal is 40 35 20 10 30 8 4 15

24. 1st swap is : 100 and 15
2nd swap is : 100 and 50
3rd swap is : 100 and 89



25. After deletion to perform rehepify operation the required time depends on the depth of the tree. So required time for rehepification is O(d).

26. To find maximum depth of the tree we can use the following concept.

Make integer 1 as a root node and take root as level 1.

Make node 2 at level 2 as a child node of node 1.

Make node 3 at level 3 as the child node of node 2.

Similarly and so on for nodes 4,5,6,7

Make node 8 at level 8 as the child node of node 7.

Make node 9 at level 9 as the child node of node 8.

Putting other nodes properly, this arrangement of the the complete binary tree will follow the property of min heap.

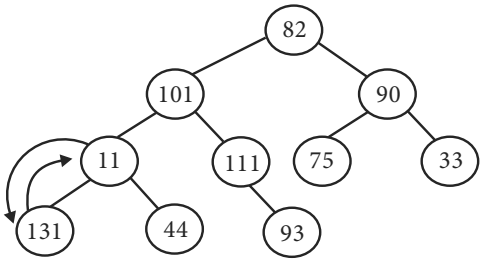
So total levels are 9. node 9 is at level 9 and depth of node 9 is 8 from the root.

27. Total number of ways to design min-heap with 7-elements

$= C(6, 3) * 2! * 2! = 80.$

28. **Heapify :** bottom-up construction :

(i) Heapify node with value 111. Nothing changes as max-heap property is already satisfied.

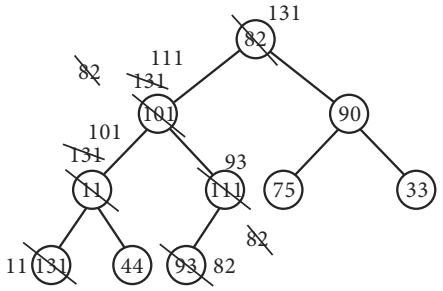


(ii) Heapify node with value 11. Now, 11 is swapped with 131 so that max-heap property is satisfied. (blue colored text in image).

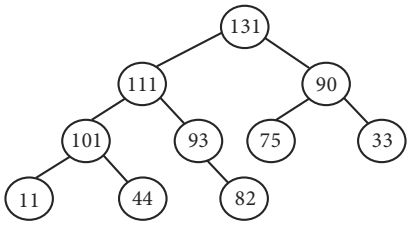
(iii) Heapify node with value 90. Nothing changes as max-heap property is already satisfied.

(iv) Heapify node with value 101. Now, 101 is swapped with 131 so that max-heap property is satisfied. (red colored text in image)

(v) Heapify node with value 82. Now 82 is swapped with 131, then 82 is swapped with 111, then 82 is swapped with 93 so that max-heap property is satisfied. (green colored text in image).



After heapify tree will look like



Final array –

[131, 111, 90, 101, 93, 75, 33, 11, 44, 82]

Hence, option (a) is the correct answer.

29. As given that;

The distinct elements (n) = 105

As we know that,

In binary min heap of n elements exactly $\left\lfloor \frac{n}{2} \right\rfloor$ internal nodes and $\left\lceil \frac{n}{2} \right\rceil$ leaf nodes.

Here, min heap is given, there is k index which is of maximum elements.

So, the max element in min heap of n distinct elements must be one of the leaf node is $\left\lceil \frac{n}{2} \right\rceil$

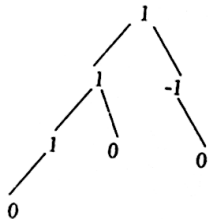
So, $\left\lceil \frac{105}{2} \right\rceil$ are number of leaf nodes = 53

Hence, option (c) is the correct answer.

30. AVL tree is a balanced tree. So, time complexity of searching = $\theta(\log n)$

But a BST, may be skewed tree, so in worst case BST searching time = $\theta(n)$

31.



Maximum height = 3

32. 43 store at location 3

165 store at a location 5

62 store at a location 2

123 → 3(collide) So according to linear probing 3 + 1 = 4

142 → 2(collide), 3(collide), 4(collide), 5(collide) → = 6

33. For each entry probability of collision is 1/20.

According to question after inserting x values probability becomes ½

(1/20) * x = 1/2

so x = 10

34. After insetting key into empty hash table using hash function :-

0	11
1	
2	13
3	36
4	92
5	4
6	71
7	14
8	
9	43
10	87

So last value (i. e 14) is store at location 7.

35. In linear probing, if there is already an element then the new element is fed into the next blank value field in the table.

36. Sequencde in option (c) creates the hash table as 42, 23 and 34 appear before 52 and 33, and 46 appears before 33.

37. In a valid insertion sequence, the elements 42, 23 and 34 must appear before 52 and 33, and 46 must appear before 33.

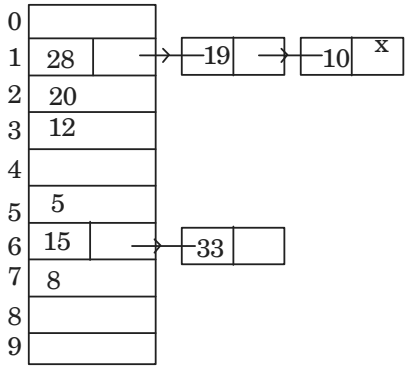
So, 3! is for elements 42, 23 and 34 because they can appear in any order, and 5 is for element 46 because 46 can appear at 5 different places (i.e. any location before 33).

So total ways are 3! * 5 = 30 ways.

38. Keys :- 5, 28, 19, 15, 20, 33, 12 17, 10

Hash function h (k) = k mod 9.

After inserting keys into hash table the Status of table is



The Maximum chain length is 3.

The Minimum chain length is 0.

The average chain length is

$$\frac{(0+3+1+1+0+1+2+0+1)}{9} = 1$$

39. **A**

P(First insertion in such a way that first 3 slots are unfilled) $= \frac{97C_1}{100C_1} = \frac{97}{100}$

B

P(second insertion in such a way that first 3 slots are unfilled) $= \frac{97C_1}{100C_1} = \frac{97}{100}$

(∵ chaining is used to resolve collision, so second insertion can be done at same index as first index]

C

P (Third insertion in such a way that first 3 slots are unfilled) $= \frac{97C_1}{100C_1} = \frac{97}{100}$

(∵ Third insertion can be done at same index as first or second index]

So total prob. P (A) × P (B) ×P(C)

$$= \frac{97}{100} \times \frac{97}{100} \times \frac{97}{100} = \frac{(97 \times 97 \times 97)}{100^3}$$

40. The given Hash function is h(key) = key mod(10).

(*) hash value for the keys 9679, 1989 and 4199 are same, i.e. 9.

(*) Hash value for the keys 1471 and 6171 are same i.e. 1

41. 1. True.

2. False.

3. True.

42. 1 will occupy location 0.

3 will occupy location 6.

8 hashed to location 0 which is already occupied, so it will be hashed to one location next to it (0) i.e. to location 1.

Since 10 also clashes, so it will be hashed to location 2.

43. $h(i) = l^3 \bmod 10$

$i = 1 = 1^3 \bmod 10 = 1$

$2 = 2^3 \bmod 10 = 8$

$3 = 3^3 \bmod 10 = 7$

$4 = 4^3 \bmod 10 = 4$

$5 = 5^3 \bmod 10 = 5$

$6 = 6^3 \bmod 10 = 6$

$7 = 7^3 \bmod 10 = 3$

$8 = 8^3 \bmod 10 = 2$

$9 = 9^3 \bmod 10 = 9$

$10 = 10^3 \bmod 10 = 0$

Hence the hash function will be $h(i) = l^3 \bmod 10$

44. The load factor = (no. of elements)/(no. of table slots)

$= 2000/25 = 80$

45. The adversary is malicious who tries to maximize the number of collisions by choosing keys that all hash to the same slot. In such case, consider a finite collection H of hash functions that maps universe U of keys into {0, 1, 2, 3, ... m – 1}.

Here m is number of slots.

H is a function called universal, if for each pair of keys k, I, $\in$ U where $k \neq 1$, the number of hash functions h $\in$ H for which h(k) = h(l) is less than or equal to $\frac{|H|}{m}$.

In otherwords, with a hash function 'h' chosen randomly from H, the probability of collision between two different keys is no more than $\frac{1}{m}$.

This is the chance of collision when choosing two slots randomly and independently.

Unit - VI

Algorithms

1 CHAPTER

Algorithm Analysis & Sorting

ASYMTOTIC NOTATION

1. Let $f(n) = n^2 \log n$ and $g(n) = n(\log n)^{10}$ be two positive functions of n. Which of the followving statements is correct?

(a) $f(n) = O(g(n))$ and $g(n) \neq O(f(n))$

(b) $g(n) = O(f(n))$ and $f(n) \neq O(g(n))$

(c) $f(n) \neq O(g(n))$ and $g(n) \neq O(f(n))$

(d) $f(n) = O(g(n))$ and $g(n) = O(f(n))$

[2001 : 1 Mark]

2. In the worst case, the number of comparisons needed to search singly linked list of length n for a given element is

(a) $\log_2 n$ (b) n/2

(c) $\log_2 n$–l (d) n **[2002 : 1 Mark]**

3. Consider the followving functions

$f(n) = 3n^{\sqrt{n}}$

$g(n) = 2^{\sqrt{n}\log_2 n}$

$h(n) = n!$

Which of the followving is true?

(a) h(n) is O(f(n)) (b) h(n) is O(g(n))

(c) g(n)is not O(f(n)) (d) f(n) is O(g(n))

[2002 : 2 Marks]

4. Consider the foliowing three claims

l. $(n + k)m = \Theta(n^m)$, where k and m are constants

2. $2^{n+1} = O(2^n)$

3. $2^{2n+1} = O(2^n)$

Which of these claims are correct?

(a) 1 and 2

(b) 1 and 3

(c) 2 and 3

(d) 1, 2 and 3 **[2003 : 1 Mark]**

5. Consider the following C function.

```
float f (float x, int y)
{
    float p, s; int i;
    for (s = 1, p = 1, i = 1; i < y; i ++)
    {
        p* = x/i;
        s + = p;
    }
    return s;
}
```

For large values of y, the return value of the function f best approximates

(a) x^y (b) e^x

(c) $\ln(1 + x)$ (d) x^x

[2003 : 1 Mark]

6. The cube root of a natural number n is defined as the largest natural number m such that $m^3 \leq n$. The complexity of computing the cube root of n (n is represented in binary notation) is

(a) $O(n)$ but not $O(n^{0.5})$

(b) $O(n^{0.5})$ but not $O((\log n)^k)$ for any constant $k > 0$.

(c) $O((\log n)^k)$ for some constant $k > 0$, but not $O((\log \log n)^m)$ for any constant $m > 0$

(d) $O((\log \log n)^k)$ for some constant $k > 0.5$, but not$O((\log \log n)^{0.5})$

[2003 : 2 Marks]

7. The tightest lower bound on the number of comparisons, in the worst case, for comparison-based sorting is of the order of

(a) n (b) n^2

(c) n log n (d) $n \log^2 n$

[2004 : 1 Mark]

8. Let A [1,..., n] be an array storing a bit (1 or 0) at each location, and f(m) is a function whose time complexity is $\Theta(m)$. Consider the following program fragment written in a C like language:

```
counter = 0;
for(i = 1; i < n; i++)
{
    if (A[i] = = 1) counter ++;
    else
    {
    f(counter);
    counter = 0;
    }
}
```

The complexity of trris program fragment is

(a) $\Omega(n^2)$ (b) $\Omega(n \log n)$ and $O(n^2)$

(c) $\Theta(n)$ (d) $O(n \log n)$

[2004 : 2 Marks]

9. The time complexity of the following C function is (assume n > 0)

```
int recursive (int n)
{
    if (n = = 1)
        return (1);
    else
        return(recursive(n-l)+ recursive(n-l));
}
```

(a) O(n) (b) O (n log n)
(c) $O(n^2)$ (d) O (2^n)

[2004 : 2 Marks]

10. The recurrence equation:

T(1) = 1
T(n) = 2T(n–l) + n, n ≥ 2
evaluates to

(a) $2^{n+1} - n - 2$ (b) $2^n - n$
(c) $2^{n+1} - 2n - 2$ (d) $2^n + n$

[2004 : 2 Marks]

11. Let f (n), g(n) and h(n) be functions defined for positive integers such that f (n) = O(g(n), g(n)) ≠ O(f(n)), g(n) = O(h(n)), and h(n) = O(g(n)). Which one of the foliowing statements is FALSE?

(a) f(n) + g(n) = O(h(n)) + h(n))
(b) f(n) = O(h(n))
(c) h(n) ≠ O(f(n))
(d) f(n)h(n) ≠ O(g(n)h(n))

[2004 : 2 Marks]

12. The time complexity of computing the transitive closure of a binary relation on a set of n elements is known to be

(a) O(n) (b) O(nlogn)
(c) $O(n^{3/2})$ (d) $O(n^3)$

[2005 : 1 Mark]

13. Let T(n) be a function defined by the recurrence T(n) = 2T(n/2) + $\sqrt{n}$ for n ≥ 2 and T(1) = 1. Which of the following statements is TRUE?

(a) T(n) = Θ(log n) (b) T(n) = Θ($\sqrt{n}$)
(c) T(n) = Θ(n) (d) T(n) = Θ(n log n)

[2005 : 2 Marks]

14. Suppose T(n) = 2T (n/2) + n, T(0) = T(l) = 1 Which one of the following is FALSE?

(a) T(n) = O (n^2) (b) T(n) = Θ (n log n)
(c) T(n) = Ω(n^2) (d) T(n) = O (n log n)

[2005 : 2 Marks]

Common Data for Q. 24 & Q. 25

Consider the following C function:

```
double foo(int n)
{
int i;
double sum;
if (n = = 0) return 1.0;
else
{
sum = 0.0;
for (i = 0; i < n ; i ++)
sum + = foo(i);
return sum;
}
}
```

[2005 : 2 Marks]

15. The space complexity of the above function is

(a) O(1) (b) O(n)
(c) O(n!) (d) $O(n^n)$

[2005 : 2 Marks]

16. The space complexity of the above function is foo() and store the values of foo(i), 0 < = i < n, as and when they are computed. With this modification, the time complexity for function foo() is significantly reduced. The space complexity of the modified function would be

(a) O(1) (b) O(n)
(c) $O(n^2)$ (d) O(n!)

[2005 : 2 Marks]

17. Consider the following C-program fragment in which i, j, and n are integer variables.

for(i = n, j = 0; i > 0; i/ = 2, j + = i);

Let Val (j) = denote the value stored in the variable j after termination of the for loop. Which one of the followving is true?

(a) val (j) = Θ (log n) (b) val (j) = Θ ($\sqrt{n}$)
(c) val (j) = Θ (n) (d) val (j) = Θ (n log n)

[2006 : 1 Mark]

18. Consider the following is true?

$T(n) = 2T\left(\left\lceil \sqrt{n} \right\rceil\right) + 1, T(1) = 1$

Which one of the following is true?

(a) T (n) = Θ (log log n)
(b) T (n) = Θ (log n)
(c) T (n) = $\Theta\left(\sqrt{n}\right)$
(d) T (n) = Θ (n) **[2006 : 2 Marks]**

19. Consider the following segment of C code

```
int j, n;
j = i;
while (j < = n)
j = j*2;
```

The number of comparisons made in the execution of the loop for any n > 0 is

(a) $\lceil \log_2 n \rceil + 1$ (b) n

(c) $\lceil \log_2 n \rceil$ (d) $\lfloor \log_2 n \rfloor + 1$

[2007 : 1 Mark]

20. What is the time complexity of the following recursive function:

```
int DoSomething (int n)
{
    if (n < = 2)
        return 1;
    else
        return DoSomething (floor (sqrt (n)))+n;
}
```

(a) $\Theta(n^2)$

(b) $\Theta(n \log_2 n)$

(c) $\Theta(\log_2 n)$

(d) $\Theta(\log_2 \log_2 n)$

[2007 : 2 Marks]

21. An array of n numbers is given, where n is an even number. The maximum as well as the minimum of these n numbers needs to be determined. Which of the following is true about the number of comparisons needed?

(a) At least 2n – c comparisons, for some constant c, are needed.

(b) At most 1.5n – 2 comparisons are needed.

(c) At least n $\log_2$ n comparisons are needed.

(d) Noneoftheabove

[2007 : 2 Marks]

22. Consider the following C code segment:

```
int IsPrime(n)
{
    int i, n;
    for(i = 2; i < = sqrt(n); i ++)
  {
    if(n% i = = 0)
        printf ("Not Prime\n");
    return 0;
  }
return 1;
}
```

Let T(n) denote the number of times the for loop is executed by the program on input n. Which of the following is TRUE?

(a) $T(n) = O\left(\sqrt{n}\right)$ and $T(n) = \Omega\left(\sqrt{n}\right)$

(b) $T(n) = O\left(\sqrt{n}\right)$ and $T(n) = \Omega(1)$

(c) $T(n) = O(n)$ and $T(n) = \Omega\left(\sqrt{n}\right)$

(d) None of these

[2007 : 2 Marks]

23. Arrange the following functions in increasing asymptotic order:

A. $n^{1/3}$ B. e^n

C. $n^{7/4}$ D. $n \log^9 n$

E. 1.0000001^n

(a) A, D, C, E, B (b) D, A, C, E, B

(c) A, C, D, E, B (d) A, C, D, B, E

[2008 : 1 Mark]

24. When $n = 2^{2k}$ for some $k \geq 0$, the recurrence relation $T(n) = \sqrt{2}\, T(n/2) + \sqrt{n}$, $T(1) = 1$ evaluates to:

(a) $\sqrt{n}(\log n + 1)$ (b) $\sqrt{n} \log n$

(c) $\sqrt{n} \log \sqrt{n}$ (d) $n \log \sqrt{n}$

[2008 : 2 Marks]

25. Consider the followving functions:

$f(n) = 2^n$

$g(n) = n!$

$h(n) = n^{\log n}$

which of the followving statements about the asymptotic behaviour of f (n), g(n), and h (n) is true?

(a) $f(n) = O(g(n)); g(n) = O(h(n))$

(b) $f(n) = \Omega(g(n)); g(n) = O(h(n))$

(c) $g(n) = O(f(n)); h(n) = O(f(n))$

(d) $h(n) = O(f(n)); g(n) = \Omega(f(n))$

[2008 : 2 Marks]

26. The minimum number of comparison required to determine if an integer appears more than n/2 times in a sorted array of n integers is

(a) $\Theta(n)$

(b) $\Theta(\log n)$

(c) $\Theta(\log^2 n)$

(d) $\Theta(1)$

[2008 : 2 Marks]

Common Data Questions Q. 27 and Q. 28

Consider the following C functions:

```
int f l(int n)
{
if(n = = 0 | | n = = 1) return n;
else return (2* f1( n –1) + 3 * f1(n – 2));
}
intf2 (int n)
{ int i;
int X[N], Y[N], Z[N];
X[1] = 1; Y[1] = 2;Z[1] = 3;
for( i = 2; i < = n ; i ++ )
 {
  X[i]=Y[i-l] + Z[i-2];
  Y[i]=2*X[i];
  Z[i] = 3*X[i];
 }
return X[n] ;
 }
```

27. The running time of f 1 (n) and f2(n) are

(a) $\Theta(n)$ and $\Theta(n)$ (b) $\Theta(2^n)$ and $\Theta(n)$

(c) $\Theta(n)$ and $\Theta(2^n)$ (d) $\Theta(2^n)$ and $\Theta(2^n)$

[2008 : 2 Marks]

28. f1(8) and f2(8) return the values

(a) 1661 and 1640 (b) 59 and 59

(c) 1640 and 1640 (d) 1640 and 1661

[2008 : 2 Marks]

29. The running time of an algorithm is represented by the following recurrence relation:

$$T(n) = \begin{cases} n & n \le 3 \\ T\left(\frac{n}{3}\right) + cn & \text{otherwise} \end{cases}$$

Which one of the following represents the time complexity of the algorithm?

(a) $\Theta(n)$ (b) $\Theta(n \log n)$

(c) $\Theta(n^2)$ (d) $\Theta(n^2 \log n)$

[2009 : 2 Marks]

30. Two alternative packages A and B are available for processing a database having 10^k records. Package A requires $0.0001\ n^2$ time units and package B requires $10n\log_{10}n$ time units to process n records. What is the smallest value of k for which package B will be preferred over A?

(a) 12 (b) 10

(c) 6 (d) 5 **[2010 : 1 Mark]**

31. Let W(n) and A(n) denote respectively, the worst case and average case running time of an algorithm executed on an input of size n. Which of the following is ALWAYS TRUE?

(a) $A(n) = \Omega(W(n))$ (b) $A(n) = \Theta(W(n))$

(c) $A(n) = O(W(n))$ (d) $A(n) = o(W(n))$

[2012 : 1 Mark]

32. The recurrence relation capturing the optimal execution time of the Towers of Hanoi problem with n discs is

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

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

(c) $T(n) = 2T(n/2) + 1$

(d) $T(n) = 2T(n - 1) + 1$ **[2012 : 1 Mark]**

33. Consider the following function:

```
int unknown (int n)
{
    int i, j, k = 0;
    for (i = n/2; i < = n; i++)
       for(j = 2; j< = n; j = j*2)
          k = k + n/2;
return (k);
}
```

The return value of the function is

(a) $\Theta(n^2)$ (b) $\Theta(n^2 \log n)$

(c) $\Theta(n^3)$ (d) $\Theta(n^3 \log n)$

[2013 : 2 Marks]

34. Which one of the following correctly determines the solution of the recurrence relation with T(1)= 1?

$$T(n) = 2T\left(\frac{n}{2}\right) + \log n$$

(a) $\Theta(n)$ (b) $\Theta(n \log n)$

(c) $\Theta(n^2)$ (d) $\Theta(\log n)$

[2014 (Set-2) : 1 Mark]

35. Suppose we have a balanced binary search tree Tholding n-numbers. We are given two numbers L and H and wish to sum up all the numbers in Tthat lie between L and H. Suppose there are m such numbers in T.

If the tightest upper bound on the time to compute the sum is $O(n^a \log^b n + m^c \log^d n)$, the value of $\alpha + 10b + 100c + 1000d$ is ____.

[2014 (Set-3): 2 Marks]

36. Consider the following C function.

```
int fun 1 (int n)
{   int i, j, k, p, q = 0;
    for (i = 1; i<n; ++i)
    {     P = 0;
```

```
        for(j = n; j > 1; j = j/2)
        ++p;
        for (k=l; k<p; k=k*2)
        ++q;
    } return q;
}
```

Which one of the following most closely approximates the return value of the function fun1?

(a) n^3 (b) $n(\log n)^2$

(c) n log n (d) n log (log n)

[2015 (Set-1) : 2 Marks]

37. An algorithm performs $(\log N)^{1/2}$ find operations, N insert operations, $(\log N)^{1/2}$ delete operations, and $(\log N)^{1/2}$ decrease-key operations on a set of data items with keys drawn from a linearly ordered set. For a delete operation, a pointer is provided to the record that must be deleted. For the decrease-key operation, a pointer is provided to the record that has its key decreased. Which one of the foliowing data structures is the most suited for the algorithm to use, if the goal is to achieve the best total asymptotic complexity considering ali the operations?

(a) Unsorted array

(b) Min-heap

(c) Sorted array

(d) Sorted doubly linked list

[2015 (Set-1) : 2 Marks]

38. Consider a complete binary tree where the left and the right subtrees of the root are max-heaps. The lower bound for the number of operations to convert the tree to a heap is

(a) $\Omega(\log n)$ (b) $\Omega(n)$

(c) $\Omega(n \log n)$ (d) $\Omega(n^2)$

[2015 (Set-2) : 1 Mark]

39. An unordered list contains n distinct elements. The number of comparisons to find an element in this list that is neither maximum nor minimum is

(a) $\Theta(n \log n)$ (b) $\Theta(n)$

(c) $\Theta(\log n)$ (d) $\Theta(1)$

[2015 (Set-2) : 1 Mark]

40. Consider the equality $\sum_{i=0}^{n} i^3 = X$ and the following choices for X

I. $\Theta(n^4)$ II. $\Theta(n^5)$

III. $O(n^5)$ IV. $\Omega(n^3)$

The equality above remains correct if X is replace by

(a) Onlyl

(b) Onlyll

(c) I or III or IV but not II

(d) II or III or IV but not I

[2015 (Set-3) : 1 Mark]

41. Let $f(n) = n$ and $g(n) = n^{(1+\sin n)}$, where n is a positive integer. Which of the following statements is/are correct?

I. $f(n) = O(g(n))$

II. $f(n) = \Omega(g(n))$

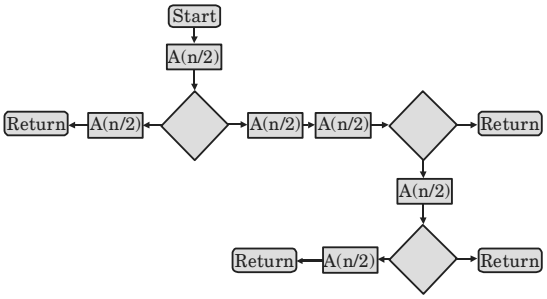
(a) Only I (b) Only ll

(c) Both I and II (d) Neither I nor II

[2015 (Set-3): 2 Marks]

42. The given diagram shows the flow chart for a recursive function A(n). Assume that all statements, except for the recursive calls, have O(1) time complexity. If the worst case time complexity of this function is $O(n^{\alpha})$, then the least possible value (accurate up to two decimal positions) of a is _____.

Flow chart for Recursive Function A(n)



[2016 (Set-2) : 2 Marks]

43. N items are stored in a sorted doubly linked list. For a delete operation, apointer is provided to the record to be deleted. For a decrease-key operation, a pointer is provided to the record on which the operation is to be performed. An algorithm performs the following operations on the list in this order: $\Theta(N)$ delete, $O(\log N)$ insert, $O(\log N)$ find, and $\Theta(N)$ decrease-key. What is the time complexity of all these operations put together?

(a) $O(\log^2 N)$ (b) $O(N)$

(c) $O(N^2)$ (d) $\Theta(N^2 \log N)$

[2016 (Set-2) : 1 Mark]

44. Match the algorithms with their time complexities:

List-I (Algorithm)

(P) Towers of Hanoi with n disks

(Q) Binary search given n sorted numbers

(R) Heap sort given n numbers at the worst case

(S) Addition of two n × n matrices

List-II (Time complexity)

(i) $\Theta(n^2)$

(ii) $\Theta(n \log n)$

(iii) $\Theta(2^n)$

(iv) $\Theta(\log n)$

(a) P – (iii), Q – (iv), R – (i), S – (ii)
(b) P – (iv), Q – (iii), R – (i), S – (ii)
(c) P – (iii), Q – (iv), R – (ii), S – (i)
(d) P – (iv), Q – (iii), R – (ii), S – (i)

[2017 (Set-2) : 1 Mark]

45. Consider the recurrence function

$$T(n) = \begin{cases} 2T\left(\sqrt{n}\right)+1, & n > 2 \\ 2, & 0 < n \le 2 \end{cases}$$

Then T(n) in terms of Θ notation is

(a) Θ(loglogn) (b) Θ(log n)

(c) $\Theta\left(\sqrt{n}\right)$ (d) Θ(n)

[2017 (Set-2) : 2 Marks]

46. Consider the following C function.

```
int fun (int n)
{   int i, j;
    for (i = 1; i <= n; i++)
    {   for (j = 1; j < n; j + = i)
        {
            printf{"%d %d", i, j);
        }
    }
}
```

Time complexity of fun in terms of Θ notation is

(a) $\Theta\left(n\sqrt{n}\right)$ (b) $\Theta(n^2)$

(c) Θ(n log n) (d) $\Theta(n^2 \log n)$

[2017 (Set-2) : 2 Marks]

47. Consider the following functions from positive integers to real numbers:

$$10,\ \sqrt{n},\ n,\ \log_2 n,\ \frac{100}{n}$$

Tire CORRECT arrangement of the above functions in increasing order of asymptotic complexity is:

(a) $\log_2 n,\ \frac{100}{n}, 10,\ \sqrt{n}, n$

(b) $\frac{100}{n},\ 10,\ \log_2 n,\ \sqrt{n},\ n$

(c) $10,\ \frac{100}{n},\ \sqrt{n},\ \log_2 n,\ n$

(d) $\frac{100}{n},\ \log_2 n,\ 10,\ \sqrt{n},\ n$

[2017 (Set-l):l Mark]

48. Let A be an array of 31 numbers consisting of a sequence of 0's followed by a sequence of l's. The problem is to find the smallest index i such that A[i] is 1 by probing the minimum number of locations in A. The worst case number of probes performed by an optimal algorithm is _____.

[2017 (Set-1): 2 Marks]

49. For parameters a and b, both of which are ω(1), $T(n) = T(n^{1/a}) + 1$, and T(b) = 1. Then T (n) is

(a) $\Theta(\log_a \log_b n)$ (b) $\Theta(\log_{ab} n)$

(c) $\Theta(\log_b \log_a n)$ (d) $\Theta(\log_2 \log_2 n)$

[2020 : 1 Mark]

50. Let P be an array containing n integers. Let t be the lowest upper bound on the number of comparisons of the array elements, required to find the minimum and maximum values in an arbitrary array of n elements. Which one of the following choices is correct ?

(a) $t > 2n - 2$

(b) $t > 3\left\lceil \frac{n}{2} \right\rceil$ and $t \le 2n - 2$

(c) $t > n$ and $t \le 3\left\lceil \frac{n}{2} \right\rceil$

(d) $t > \left\lceil \log_2 (n) \right\rceil$ and $t \le n$

[2021 (Set-1) : 1 Mark]

51. Consider the following three functions.

$$f_1 = 10^n \quad f_2 = n^{\log n} \quad f_3 = n^{\sqrt{n}}$$

Which one of the following options arranges the functions in the increasing order of asymptotic growth rate ?

(a) f_3, f_2, f_1 (b) f_2, f_1, f_3
(c) f_1, f_2, f_3 (d) f_2, f_3, f_1

[2021 (Set-1) : 1 Mark]

52. Consider the following recurrence relation.

$$T(n) = \begin{cases} T\left(\frac{n}{2}\right) + T\left(\frac{2n}{5}\right) + 7n & \text{if } n > 0 \\ 1 & \text{if } n = 0 \end{cases}$$

Which one of the following options is correct ?

(a) $T(n) = \Theta(n^{5/2})$
(b) $T(n) = \Theta(n \log n)$
(c) $T(n) = \Theta(n)$
(d) $T(n) = \Theta\left((\log n)^{\frac{5}{2}}\right)$ **[2021 (Set-1) : 2 Marks]**

53. For constants $a \geq 1$ and $b > 1$, consider the following recurrence defined on the non-negative integers :

$$T(n) = a.T\left(\frac{n}{b}\right) + f(n)$$

Which one of the following options is correct about the recurrence T(n) ?

(a) If f(n) is $n \log_2(n)$, then T(n) is $\Theta(n \log_2(n))$.

(b) If f(n) is $\frac{n}{\log_2(n)}$, then T(n) is $\Theta(\log_2(n))$.

(c) If f(n) is $O\left(n^{\log_b(a)-\epsilon}\right)$ for some $\epsilon > 0$, then T(n) is $\Theta\left(n^{\log_b(a)}\right)$.

(d) If f(n) is $\Theta\left(n^{\log_b(a)}\right)$, then T(n) is $\Theta\left(n^{\log_b(a)}\right)$.

[2021 (Set-2) : 2 Marks]

54. Which one of the following statements is TRUE for all positive functions f(n)?

(a) $f(n^2) = \theta(f(n)^2)$, when $f(n)$ is a polynomial

(b) $f(n^2) = o(f(n)^2)$

(c) $f(n^2) = O(f(n)^2)$, when $f(n)$ is an exponential function

(d) $f(n^2) = \Omega(f(n)^2)$

[2022 : 1 Mark]

55. Let f and g be functions of natural numbers given by $f(n) = n$ and $g(n) = n^2$.

Which of the following statements is/are TRUE?

(a) $f \in O(g)$ (b) $f \in \Omega(g)$

(c) $f \in o(g)$ (d) $f \in \theta(g)$

[2023 : 1 Mark]

56. Consider functions Function 1 and Function 2 expressed in pseudocode as follows:

Function 1	**Function 2**
while n > 1 do	for i = 1 to 100 * n do
for i = 1 to n do	x = x+ 1;
x = x+ 1;	end for
end for	
$n = \lfloor n / 2 \rfloor$;	
end while	

Let $f_1(n)$ and $f_2(n)$ denote the number of times the statement "x = x + 1" is executed in **Function_ 1** and **Function_2**, respectively.

Which of the following statements is/are TRUE?

(a) $f_1(n) \in \theta(f_2(n))$ (b) $f_1(n) \in o(f_2(n))$

(c) $f_1(n) \in \omega(f_2(n))$ (d) $f_1(n) \in O(n)$

[2023 : 2 Marks]

57. Given an integer array of size N, we want to check if the array is sorted (in either ascending or descending order). An algorithm solves this problem by making a single pass through the array and comparing each element of the array only with its adjacent elements. The worst - case time complexity of this algorithm is

(a) Ω (N) but not O(N)

(b) O(N) but not Ω(N)

(c) Both O (N) and Ω(N)

(d) Neither O (N) nor Ω(N)

[2024 (Set-1) : 1 Mark]

58. Let T(n) be the recurrence relation defined as follows:

T (0) = 1,

T (1) = 2, and

T (n) = 5T(n – 1) – 6T(n – 2) for $n \geq 2$

Which one of the following statements is TRUE?

(a) $T(n) = \Theta(n3^n)$ (b) $T(n) = \Theta(n2^n)$

(c) $T(n) = \Theta(3^n)$ (d) $T(n) = \Theta(2^n)$

[2024 (Set-2) : 1 Mark]

59. Consider the following recurrence relation:

$$T(n) = \begin{cases} \sqrt{n}T\left(\sqrt{n}\right) + n & \text{for } n \geq 1 \\ 1 & \text{for } n = 1 \end{cases}$$

Which one of the following options is CORRECT?

(a) $T(n) = \Theta(n \log \log n)$

(b) $T(n) = \Theta(n^2 \log n)$

(c) $T(n) = \Theta(n \log n)$

(d) $T(n) = \Theta(n^2 \log \log n)$

[2024 (Set-1) : 2 Marks]

SORTING

60. Suppose there are log n sorted lists of n/log n elements each. The time complexity of producing a sorted list of all these elements is:

(**Hint:** Use a heap data structure)

(a) $O(n \log \log n)$ (b) $\Theta(n \log n)$

(c) $\Omega(n \log n)$ (d) $\Omega(n^{3/2})$

[2005 : 2 Marks]

61. Which one the foll owing inplace sorting algorithms needs the minimum number of swaps?

(a) Quicksort (b) Insertion sort

(c) Selection sort (d) Heap sort

[2006 : 1 Mark]

62. Which of the following sorting algorithms has the lowest worst-case complexity?

(a) Mergesort (b) Bubblesort

(c) Quicksort (d) Selectionsort

[2007 : 1 Mark]

63. If we use Radix Sort to sort n integers in the range $(n^{k/12}, n^k]$, for some $k > 0$ which is independent of n, the time taken would be

(a) $\Theta(n)$ (b) $\Theta(kn)$

(c) $\Theta(n \log n)$ (d) $\Theta(n^2)$

[2008 : 2 Marks]

64. What is the number of swaps required to sort n elements using selection sort, in the worst case?

(a) $\Theta(n)$ (b) $\Theta(n \log n)$

(c) $\Theta(n^2)$ (d) $\Theta(n^2 \log n)$

[2009 : 1 Mark]

65. A list of n strings, each of length n, is sorted into lexicographic order using the merge sort algorithm. The worst case running time of this computation is

(a) $O(n \log n)$ (b) $O(n^2 \log n)$

(c) $O(n^2 + \log n)$ (d) $O(n^2)$

[2012 : 2 Marks]

66. Which one of the foliowing is the tightest upper bound that represents the time complexity of inserting an object into a binary search tree of n nodes?

(a) O(1) (b) O(log n)

(c) O(n) (d) O(n log n)

[2013 : 1 Mark]

67. Which one of the following is the tightest upper bound that represents the number of swaps required to sort n numbers using selection sort?

(a) O(log n) (b) O(n)

(c) O(n log n) (d) $O(n^2)$

[2013 : 1 Mark]

68. The number of elements that can be stored in $\Theta(\log n)$ time using heap sort is

(a) $\Theta(1)$ (b) $\Theta\left(\sqrt{\log n}\right)$

(c) $\Theta\left(\frac{\sqrt{\log n}}{\log \log n}\right)$ (d) $\Theta(\log n)$

[2013 : 2 Marks]

69. The minimum number of comparisons required to find the minimum and the maximum of 100 numbers is _____.

[2014 (Set-1) : 2 Marks]

70. An array of 25 distinct elements is to be sorted using quicksort. Assume that the pivot element is chosen uniformly at random. The probability that the pivot element gets placed in the worst possible location in the first round of partitioning (rounded off to 2 decimal places) is _____.

[2019 : 1 Mark]

QUICK SORT

71. Randomized quicksort is an extension of quicksort where the pivot is chosen randomly. What is the worst case complexity of sorting n numbers using randomized quicksort?

(a) O(n) (b) (n log n)

(c) $O(n^2)$ (d) O(n!)

[2001 : 1 Mark]

72. The median of n elements can be found in O(n) time. Which one of the following is correct about the complexity of quick sort, in which remains is selected as pivot?

(a) $\Theta(n)$ (b) $\Theta(n \log n)$

(c) $\Theta(n^2)$ (d) $\Theta(n^3)$

[2006 : 2 Marks]

73. Consider the Quicksort algorithm. Suppose there is a procedure for finding a pivot element which splits the list into sub-lists each of which contains at least one-fifth of the elements. Let T(n) be the number of comparisons required to sort n elements. Then

(a) $T(n) \leq 2T(n/5) + n$

(b) $T(n) \leq T(n/5) + T(4n/5) + n$

(c) $T(n) \leq 2T(4n/5) + n$

(d) $T(n) \leq 2T(n/2) + n$ **[2008 : 2 Marks]**

74. In quick sort, for sorting n elements, the (n/4)th smallest element is selected as pivot using an O(n) time algorithm. What is the worst case time complexity of the quick sort?

(a) $\Theta(n)$ (b) $\Theta(n \log n)$

(c) $\Theta(n^2)$ (d) $\Theta(n^2 \log n)$

[2009 : 2 Marks]

75. Let P be a quicksort program to sort numbers in ascending order using the first element as the pivot. Let t_1 and t_2 be the number of comparisons made by Pfor the inputs [1 2 3 4 5] and [4 1 5 3 2] respectively. Which one of the following holds?

(a) $t_1 = 5$ (b) $t_1 < t_2$

(c) $t_1 > t_2$ (d) $t_1 = t_2$

[2014 (Set-1) : 1 Mark]

76. You have an array of n elements. Suppose you implement quicksort by always choosing the central element of the array as the pivot. Then the tightest upper bound for the worst case performance is

(a) $O(n^2)$ (b) $O(n\log n)$

(c) $\Theta(n\log n)$ (d) $O(n^3)$

[2014 (Set-3) : 1 Mark]

77. Which one of the following is the recurrence equation for the worst case time complexity of the Quicksort algorithm for sorting $n(\geq 2)$ numbers? In the recurrence equations given in the options below, c is a constant.

(a) $T(n) = 2T(n/2) + cn$

(b) $T(n) = T(n-1) + T(1) + cn$

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

(d) $T(n) = T(n/2) + cn$

[2015 (Set-1): 1 Mark]

78. Suppose you are provided with the following function declaration in the C programming language.

```
int partition (int a[ ], int n);
```

The function treats the first element of a[ ] as a pivot, and rearranges the array so that ali elements less than or equal to the pivot is in the left part of the array, and all elements greater than the pivot is in the right part. In addition, it moves the pivot so that the pivot is the last element of the left part. The return value is the number of elements in the left part.

The following partially given function in the C programming language is used to find the kth smallest element in an array a[ ] of size n using the partition function. We assume $k \leq n$.

```
int kth_smallest (int a[ ], int n, int k)
{
    int left_end = partition (a, n);
    if(left_end+ 1==k)
    {
        return a [left_end];
    }
    if(left end+l>k)
    {
        return kth_smallest (_______);
    }
    else
    {
        return kth_smallest (_______);
    }
}
```

The missing argument lists are respectively

(a) (a, left_end, k) and (a+left_end+l, n-left_end-1, k-left_end-l)

(b) (a, left_end, k) and (a, n-left_end-l, k-left_end-l)

(c) (a+left_end+l, n-left_end-l, k-left_end-l) and(a, left_end, k)

(d) (a, n-left_end-l, k-left_end-l) and (a, left_end, k)

[2015 (Set-2) : 2 Marks]

79. The worst case running times of Insertion sort, Merge sort and Quick sort, respectively, are:

(a) $\Theta(n\log n)$, $\Theta(n\log n)$, and $\Theta(n^2)$

(b) $\Theta(n^2)$, $\Theta(n^2)$, and $\Theta(n\log n)$

(c) $\Theta(n^2)$, $\Theta(n\log n)$, and $\Theta(n\log n)$

(d) $\Theta(n^2)$, $\Theta(n\log n)$, and $\Theta(n^2)$

[2016 (Set-1): 1 Mark]

INSERTION SORT

80. The usual $\Theta(n^2)$ implementation of Insertion Sort to sort an array uses linear search to identify the position where an element is to be inserted into the already sorted part of the array. If instead, we use binary search to identify the position, the worst case running time will

(a) remain $\Theta(n^2)$ (b) become $\Theta(n(\log n)^2)$

(c) become $\Theta(n\log n)$ (d) become $\Theta(n)$

[2003 : 1 Mark]

81. What would be the worst case time complexity of the insertion sort algorithm, if the inputs are restricted to permutations of 1...n with at most n inversions?

(a) $\Theta(n^2)$ (b) $\Theta(n\log n)$

(c) $\Theta(n^{1.5})$ (d) $\Theta(n)$ **[2003 : 2 Marks]**

MERGE SORT

82. Consider a list of recursive algorithms and a list of recurrence relations as shown below. Each recurrence relation corresponds to exactly one algorithm and is used to derive the time complexity of the algorithm.

List-I (Recursive Algorithm)

P. Binary search

Q. Merge sort

R. Quick sort

S. Tower of Hanoi

List-II (Recurrence Relation)

I. $T(n) = T(n-k) + T(k) + cn$

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

III. $T(n) = 2T(n/2) + cn$

IV. $T(n) = T(n/2) + 1$

Which of the following is the correct match between the algorithms and their recurrence relations?

Codes:

	P	Q	R	S
(a)	II	III	IV	I
(b)	IV	III	I	II
(c)	III	II	IV	I
(d)	IV	II	I	III

[2004 : 2 Marks]

83. Assume that a mergesort algorithm in the worst case takes 30 seconds for an input of size 64. Which of the following most closely approximates the maximum input size of a problem that can be solved in 6 minutes?

(a) 256 (b) 512

(c) 1024 (d) 2048

[2015 (Set-3): 2 Marks]

84. Assume that the algorithms considered here sort the input sequences in ascending order. If the input is already in ascending order, which of the followingareTRUE?

I. Quicksort runs in Q(n^2) time

II. Bubblesort runs in Q(n^2) time

III. Mergesort runs in Q(n) time

IV. Insertion sort runs in Q(n) time

(a) I and II only (b) I and III only

(c) II and lV only (d) I and lV only

[2016 (Set-2): 1 Mark]

ALGO CONCEPT

85. Consider the following algorithm for searching for a given number x in an unsortedarray A[1 .....n] having n distinct values:

1. Choose an i uniformly at random from 1.. .n;
2. If A[i] = x then Stop else Goto 1;

Assuming that x is present on A, vvhat is the expected number of comparisons made by the algorithm before it terminates?

(a) n (b) n – 1

(c) 2n (d) n/2 **[2002 : 2 Marks]**

86. The running time of the follovving algorithm Procedure A(n)

If n < = 2 return (1) else return $\left(A\left(\lceil\sqrt{n}\rceil\right)\right)$; is best described by

(a) O(n) (b) O(logn)

(c) O(log logn) (d) O(1) **[2002:2 Marks]**

87. If all permutations are equally likely, what is the expected number of inversions in a randomly chosen permutation of l...n?

(a) n(n – 1)/2 (b) n(n – 1)/4

(c) n(n + 1)/4 (d) 2n[$\log_2$n]

[2003 : 2 Marks]

88. What does the following algorithm approximate? (Assume m > 1, $\in$ > 0).

```
x = m ;
y = 1 ;
while (x – y > ∈)
{   x = (x + y) / 2 ;
    y = m / m ;
}
print (x) ;
```

(a) log m

(b) m^2

(c) $m^{1/2}$

(d) $m^{1/3}$ **[2004 : 2 Marks]**

89. Suppose each set is represented as a linked list with elements in arbitrary order. Which of the operations among union, intersection, membership, cardinality will be the slowest?

(a) union only

(b) intersection, membership

(c) membership, cardinality

(d) union, intersection

[2004 : 2 Marks]

90. In the followving table, the left column contains the names of standard graph algorithms and the right column contains the time complexities of the algorithms. Match each algorithm with its time complexity.

List-I

1. Bellman-Ford algorithm
2. Kruskal's algorithm
3. Floyd-Warshall algorithm
4. Topological sorting

List-II

A. O(mlog n)

B. O(n^3)

C. O(nm)

D. O(n + m)

(a) 1-C, 2-A, 3-B, 4-D

(b) 1-B, 2-D, 3-C, 4-A

(c) 1-C, 2-D, 3-A, 4-B

(d) 1-B, 2-A, 3-C, 4-D **[2005 : 1 Mark]**

91. An element in an array X is called a leader if it is greater than all elements to the right of it in X. The best algorithm to find ali leaders in an array.

(a) Solves it in linear time using a left to right pass of the array

(b) Solves in linear time using a right to left pass

(c) Solves it is using divide and conquer in time $\Theta(n \log n)$

(d) Solves it in time $\Theta(n^2)$ **[2006 : 1 Mark]**

92. Given two arrays of numbers $a_1,..., a_n$ and $b_1,..., b_n$ where each number is 0 or 1, the fastest algorithm to find the largest span (i, j) such that $a_i + a_{i+i} +....+ a_j = b_i + b_{i+i} +....+ b_j$"or report that there is no such span,

(a) Takes $O(3^n)$ and $\Omega(2^n)$ time if hashing is permitted

(b) Takes $O(n^3)$ and $\Omega(n^{2.5})$ time in the key comparison model

(c) Takes $\Theta(n)$ time and space

(d) Takes $O(\sqrt{n})$ time only if the sum of the 2n **[2006 : 2 Marks]**

93. A set X can be represented by an array x[n] as follows

$$x[i] = \begin{cases} 1; & \text{if } i \in X \\ 0; & \text{otherwise} \end{cases}$$

Consider the following algorithm in which x, y, and z are boolean arrays of size n: algorithm zzz (x[ ], y[ ], z[ ])

```
{
    int i;
    for (i = 0; i < n; ++ i)
        z[i] = (x [i] ∧ ~ y[i]) ∨ (~x[i] ∧ y[i]);
}
```

The set Z computed by the algorithm is

(a) $(X \cup Y)$ (b) $(X \cap Y)$

(c) $(X - Y) \cap (Y - X)$ (d) $(X - Y) \cup (Y - X)$

[2006 : 2 Marks]

94. 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

(a) Dijkstra's algorithm starting from S.

(b) Warshall's algorithm

(c) performing a DFS starting from S

(d) performing a BFS starting from S

[2007 : 2 Marks]

95. Consider n jobs $J_1, J_2,..., J_n$ such that job J_i has execution time t_i and a non-negative integer weight w_i. The weighted mean completion time of the jobs is defined to be $\sum_{i=1}^{n} w_i T_i \Big/ \sum_{i=1}^{n} w_i$.

where T_i is the completion time of job J_i. Assuming that there is only one processor available, in what order must the jobs be executed in order to minimize the weighted mean completion time of thejobs?

(a) Non-decreasing order of t_i

(b) Non-increasing order of w_i

(c) Non-increasing order of $w_i t_i$

(d) Non-increasing order of w_i/t_i

[2007 : 2 Marks]

96. Suppose P, Q, R, S, T are sorted sequences havinglengths 20,24,30,35,50 respectively. They are to be merged into a single sequence by merging together two sequences at a time. The number of comparisons that will be needed in the worst case by the optimal algorithm for doing this is _____.

[2014 (Set-2) : 2 Marks]

97. Suppose you want to move from 0 to 100 on the number line. In each step, you either move right by a unit distance or you take a shortcut. A shortcut is simply a pre-specified pair of integers i, j with i < j. Given a shortcut i, j if you are at position i on the number line, you may directly move to j. Suppose T(k) denotes the smallest number of steps needed to move from k to 100. Suppose further that there is at most 1 shortcut involving any number, and in particular from 9 there is a shortcut to 15.

Let y and z be such that T(9) = 1 + min (T(y), T(z)). Then the value of the product yz is_____.

[2014 (Set-3) : 2 Marks]

98. Match the followving:

List-I

A. Prim's algorithm for minimum spanning tree

B. Floyd-Warshall algorithm for ali pairs shortest paths

C. Mergesort

D. Hamiltonian circuit

List-II

1. Backtracking
2. Greedy method
3. Dynamic programming
4. Divide and conquer

Codes:

	A	B	C	D
(a)	3	2	4	1
(b)	1	2	4	3
(c)	2	3	4	1
(d)	2	1	3	4

[2015 (Set-1) : 1 Mark]

99. Given below are some algorithms, and some algorithm design paradigms.

List-I

A. Dijkstra's Shortest Path
B. Floyd-Warshall algorithm to compute all pairs shortest path
C. Binary search on a sorted array
D. Backtracking search on a graph

List-II

1. Divide and Conquer
2. Dynamic Programming
3. Greedy design
4. Depth-first search
5. Breadth-first search

Match the above algorithms (List-I) to the corresponding design paradigm (List-II) they follow.

Codes:

	A	**B**	**C**	**D**
(a)	1	3	1	5
(b)	3	3	1	5
(c)	3	2	1	4
(d)	3	2	1	5

[2015 (Set-2): 2 Marks]

100. The Floyd-Warshall algorithm for all-pair shortest paths computation is based on

(a) Greedy paradigm
(b) Divide-and-conquer paradigm
(c) Dynamic Programming paradigm
(d) Neither Greedy nor Divide-and-Conquer nor Dynamic Programming paradigm.

[2016 (Set-2): 1 Mark]

101. Consider the following table:

Algorithms	**Design Paradigms**
(P) Kruskal	(i) Divide and Conquer
(Q) Quicksort	(ii) Greedy
(R) Floyd-Warshall	(iii) Dynamic Programming

Match the algorithms to the design paradigms they are based on.

(a) (P) $\leftrightarrow$ (ii), (Q) $\leftrightarrow$ (iii), (R) $\leftrightarrow$ (i)
(b) (P) $\leftrightarrow$ (iii), (Q) $\leftrightarrow$ (i), (R) $\leftrightarrow$ (ii)
(c) (P) $\leftrightarrow$ (ii), (Q) $\leftrightarrow$ (i), (R) $\leftrightarrow$ (iii)
(d) (P) $\leftrightarrow$ (i), (Q) $\leftrightarrow$ (ii), (R) $\leftrightarrow$ (iii)

[2017 (Set-1) : 1 Mark]

102. Consider the following array.

23	32	45	69	72	73	89	97

Which algorithm out of the following options uses the least number of comparisons (among the array elements) to sort the above array in ascending order ?

(a) Selection sort
(b) Mergesort
(c) Insertion sort
(d) Quicksort using the last element as pivot

[2021 (Set-1): 1 Mark]

ANSWERS

1. (*b*)	**2.** (*d*)	**3.** (*d*)	**4.** (*a*)	**5.** (*b*)	**6.** (*c*)	**7.** (*c*)	**8.** (*c*)	**9.** (*d*)	**10.** (*a*)
11. (*d*)	**12.** (*d*)	**13.** (*c*)	**14.** (*c*)	**15.** (*b*)	**16.** (*b*)	**17.** (*c*)	**18.** (*b*)	**19.** (*d*)	**20.** (*d*)
21. (*b*)	**22.** (*b*)	**23.** (*a*)	**24.** (*a*)	**25.** (*d*)	**26.** (*b*)	**27.** (*b*)	**28.** (*c*)	**29.** (*a*)	**30.** (*c*)
31. (*c*)	**32.** (*d*)	**33.** (*b*)	**34.** (*a*)	**35.** (110)	**36.** (*d*)	**37.** (*a*)	**38.** (*a*)	**39.** (*d*)	**40.** (*c*)
41. (*d*)	**42.** (2.32)	**43.** (*c*)	**44.** (*c*)	**45.** (*b*)	**46.** (*c*)	**47.** (*b*)	**48.** (5)	**49.** (a)	**50.** (c)
51. (d)	**52.** (c)	**53.** (c)	**54.** (a)	**55.** (a, c)	**56.** (a, d)	**57.** (c)	**58.** (b)	**59.** (a)	**60.** (*a*)
61. (*c*)	**62.** (*a*)	**63.** (*c*)	**64.** (*a*)	**65.** (*b*)	**66.** (*b*)	**67.** (*b*)	**68.** (*c*)	**69.** (148)	**70.** (0.08)
71. (*c*)	**72.** (*d*)	**73.** (*b*)	**74.** (*b*)	**75.** (*c*)	**76.** (*a*)	**77.** (*b*)	**78.** (*a*)	**79.** (*d*)	**80.** (*a*)
81. (*d*)	**82.** (*b*)	**83.** (*b*)	**84.** (*d*)	**85.** (*a*)	**86.** (*c*)	**87.** (*b*)	**88.** (*c*)	**89.** (*d*)	**90.** (*a*)
91. (*b*)	**92.** (*c*)	**93.** (*d*)	**94.** (*d*)	**95.** (*d*)	**96.** (358 to 358)		**97.** (150)	**98.** (*c*)	**99.** (*c*)
101. (*c*)	**101.** (*c*)	**102.** (c)							

EXPLANATIONS

1. $f(n) = n^2 \log n$

$g(n) = n (\log n)^{10}$

Here we can remove common factor from both the function to simply the value of f (n) & g (n)

So $f(n) = n$

$g(n) = (\log n)^9$

Now choose the largest value of n to compare both the function.

if $n = 2^{100}$ then $f(n) = 2^{100}$

$g(n) = (100)^9$

$f(n) > g(n)$.

2. The worst case number of comparison is n. to search any element from singly link list.

3. $f(n) = 3n^{\sqrt{n}}$

$g(n) = 2^{\sqrt{n}\log_2 n} \Rightarrow 2^{\log 2n\sqrt{n}} \Rightarrow n^{\sqrt{n}}$

$h(n) = n! \cong h^n$

So, $f(n) \cong g(n) < h(n)$ (in terms of growth rate)

so $f(n) = O(g(n))$

4. **Note:-** In devotion place change first point by

1. $(n+k)^m = O(n^m)$, where k and m are constants.
1. $(n+k)m \cong O(nm)$ be cause k & m are constant then growth rate of both the function is same.
2. $2^{n+1} \Rightarrow 2.2^n$, Here 2 is constant
   So function is 2^n. Statement is true.
3. $2^{2n+1} \Rightarrow 2.2^{2n} \Rightarrow 2.4^n$
   So $4^n > 2^n$ then $2^{2n+1} \neq O(2^n)$.

5. For large value of y

$$P = P * \frac{x}{i}$$

$$S = S + P$$

When i < y and i + +

$$S = 1 + \frac{x}{1} + \frac{x^2}{21} + \frac{x^3}{31} + \ldots\ldots$$

Hence $S = e^x$

6. Use binary search in the array of number from 1 ..... n to check cube of the number matches n (i.e. a[i]* a [i]* a [i] = = n). option (c) is true.

7. The best complexity in worst case for any comparision based sorting technical cannot be less then nlogn.

8. **Case I :**– if A = | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... |

then always else part is execute so f (wanter) where counter = 0

is executed n times. As given in clvestion complexity of f (o) is O (1) So O (1) + O (1)..... b times $\Rightarrow$ O (n)

Case II : if A = [1, 1,.....1]

the loop execute n time and only if statment is executed hence complexity is O (n).

Case III : if A = [1,0,1,0,1,0.....]

or A = [0,1,0,1,0,1.....]

or any other case.

Both if and else is executed but every time when else part is executed counter is again set to O. Hence complexity is O (n).

9. Recurrence relation for function is

$$\begin{aligned} T(n) &= 2\,T(n-1) + 1 \\ &= 2\,[2T(n-2) + 1] + 1 \\ &= 4\,T(n-2) + 2 + 1 \\ &= 4\,[2T(n-3)+1] + 2 + 1 \\ &= 2^3\,T(n-3) + 2^2 + 2 + 1 \end{aligned}$$

$2k\,T(n-k) + 2^{K-1} + \text{---}2+1$

put $n-k = 1$

So $2^{n-1}\,T(1) + 2^{h-2} + \text{-----}\ 2 + 1$

$\Rightarrow\ 2^{n-1} + 2^{n-2} + \text{---}\ 2 + 1$

$\Rightarrow\ 2^{n-1} \Rightarrow O(2^n)$.

10.

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2\,T(n-1) + n\ ,\ n \geq 2 \end{aligned}$$

if $n = 2$ then

$$\begin{aligned} T(2) &= 2\,T(1) + 2 \\ &= 2 \times 1 + 2 = 4 \end{aligned}$$

if $n = 3$ then $T(3) = 2\,T(2) + 3$

$$= 2^* 4 + 3 = 11$$

Put n = 3 in option (a)

$$2^4 - 3 - 2 = 16 - 3 - 2 = 11$$

so option (a) is true.

11. Given,

$f(n) = O(g(n))$

$g(n) \neq O(f(n))$

$g(n) = O(h(n))$

$h(n) = O(g(n))$

So conclusion is

$f(n) < g(n) = h(n)$ { in terms of growth rate}

So

(a) $f(n) + g(n) = O(h(n) + h(n))$ is true.

(b) $f(n) = O(h(n))$ is true.

(c) $h(n) \neq O(f(n))$ is true.

(d) $f(n) * h(n) \neq O(g(n) * h(n)$ is false.

12. The time complexity of computing the transitive closure of binary relation on a set of n elements is $O(n^3)$.

13. $T(n) = 2T\left(\frac{n}{2}\right) + \sqrt{n}$

Apply master's method

$n^{\log_2^2} \Rightarrow n > \sqrt{n}$

So $\boxed{T(n) = Q(n)}$

14. $T(n) = 2\,T(n/2) + n$

Apply masters method.

$n^{\log_2^2} \Rightarrow n \cong n$

So $T(n) = O(n\log n)$

Now There growth rate of n^2 and hlogn is greater then and equal to nlogn theme (c) is flase $T(n) = 52/$ represent lower growth (n^2)

15. Here foo(1) is recursive function. Spare complexity is O (n) as there can be at most O (n) active function at a time.

16. Space complexity is O (n). The space we need array of size O (n). The space required for recursive call would be O (1) as the value would be taken from array again & again.

17. If n is a power of 2, then

$$j = n + \frac{n}{2} + \frac{n}{2} + \text{.....} + \log_2 n \text{ term}$$

If n is not a power of 2, then there will be minor differences of 1 at $\frac{n}{2^i}$ wherever $\frac{n}{2^{i-1}}$ is odd.

Hence val (j) computed on the basis of $n = 2^n$ will give a fair answer

$$\begin{aligned} j &= n + \frac{n}{2} + \text{.....1 G.P.} \\ &= \frac{n\left(\left(\frac{1}{2}\right)^{\log n} - 1\right)}{\frac{1}{2} - 1} \\ &= 2n\left(1 - \frac{1}{2^{\log n}}\right) = 2n\left(1 - \frac{1}{n}\right) \\ &= 2\,(n-1) = \theta\,(n) \end{aligned}$$

18. $T(n) = 2\,T(\sqrt{n}) + 1$

Put $n = 2^K$

$T(2^K) = 2T(n^{k/2}) + 1$

replace $T(2^k)$ by $S(k)$

$S(k) = 2\,S(k/2) + 1$

Apply masters

$K^{\log_2^2} \Rightarrow K > 1$

So $Q(k)$

Now we know $n=2^k$, $k = \log_2(n)$

Then $Q(\log n)$

19. Since, j increases in power of 2's.

if statement $j = j * 2$ executes k times, then

$$2^k \leq n$$

$$\Rightarrow \quad k \leq \log_2 n$$

Since k will be integer,

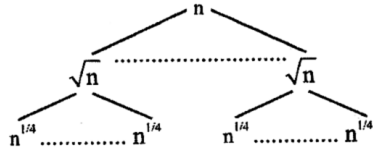
total number of comparison

$$= \lfloor \log_2 n \rfloor + 1 \text{ (when loop exits)}$$

20. The given function is recursive so the equivalent recursion equation is

$$T(n) = 1 \quad n \le 2$$

$$T(n) = \lfloor \sqrt{n} \rfloor + n \quad n > 2$$



All the level sums are equal to n. The problem size at level k of the recursion tree is n^{2-k} and we stop recursing when this value is a constent. Setting $n^{2-k} = 2$ and solving for k gives us

$$2^{-k} \log_2 n = 1$$

$$\Rightarrow \quad 2^k = \log_2 n$$

$$\Rightarrow \quad k = \log_2 \log_2 n$$

So $\quad T(n) = \theta(\log_2 \log_2 n)$

21. Since,

$2n - c$ = average number of comparison needed.

$1.5\,n - 2$ = number of comparison in case.

$n \log_2 n$ = also doesn't conform with number of comparison needed.

22. Upper bound may be $\sqrt{n}$ times it gets executed. Lower bound may be that if gets executed only once or twice.

23. B & E are exponential functional

so {B, E} > {A, C, D}

* for larger value of n E > B.
* A < C (clearly power is less)
* D < C for larger value of n

So A < D < C < E < B.

24. Given if n = 1 then T (1) = 1

So put n = 1 in all options only option (a) gives 1.

25. $f(n) = 2^n$

$g(n) = n! \cong n^n$

$h(n) = n^{\log n}$

for larger value of n

g (n) > f (n) > h (n)

so we can conclude that

f (n) = O (g (n)) or g(n) = w(f(n))

and h (n)= 0f(n).

26. The minimum number of comparison is logn.

27. Recurrence relation for $f_1()$is

T (n) = 2T (n–1) + 3 T (n–2)

After solving it would be O $((1.6)^n)$ so the largest value among the options is $O(2^n)$.

$f_2()$ is simple loop executed n times.

So O (n).

28. Both function preform same operation so result is same 1640 & 1640

29. Applying Master's Theorem

$cn > n^{\ln_3 1}$

$cn > n^0$

Now checking af(n/b) ≤ kf(n) for some k < 1

1*cm/3 ≤ kcn

This is true for k > 1/3

Hence solution is θ(n).

30. B must be preferred on A as

$$0.0001\, n^2 < 10\, n \log_{10} n$$

$$\Rightarrow \quad 10^{-5}\, n < \log_{10} n$$

$$\Rightarrow \quad 10^{-5} < \log_{10} n^{1/n}$$

$$\Rightarrow \quad n\, 10^{-5} < \log_{10} n$$

We know $\quad \log_{10} n = k$

$$\Rightarrow \quad 10^k = t$$

$$\therefore \quad k > 10^{-5}\, 10k$$

$$\Rightarrow \quad k > 10^{k-5}$$

$$\Rightarrow \quad k - 5 > 0$$

$$\Rightarrow \quad k > 5$$

Then $\quad \min k = 6$

31. If B(n), XXA(n) and W(n) denote best case , average case and worst case time complexities of an algorithm P respectively then B(n) = O(A(n)), A(n) = O(W(n))

32. Let the three pegs be A,B and C, the goal is to move n pegs from A to C usingpeg B

The following sequence of steps are executed recursively

1. move $n - 1$ discs from A to B. This leaves disc n alone on peg A ..... $T(n-1)$
2. move disc n from A to C 1
3. move $n - 1$ discs from B to C so they sit on disc n $T(n-1)$

So, $\quad T(n) = 2T(n-1) + 1$

33. So, this is in $\theta(\log n)$

Hence Answer is (*c*)

The outer for-loop goes for $\frac{n}{2}+1$ iterations. The inner for-loop runs independent of the otuer loop.

And for each inner iteration. $\frac{n}{2}$ gets added to k.

$\therefore$ Our answer = $\frac{n}{2} \times$ # outer loops × #Inner loops per outer loop

Inner loops = $\theta(\log n)$ $\quad [\because 2^{\theta(\log n)} = \theta(n)]$

$$\therefore \quad \text{Our answer} = \frac{n}{2} + \left[\frac{n}{2}+1\right] \cdot \theta(\log n) = \theta(n^2 \log n)$$

$L_1 = \{0^p\, 1^q\, 0^r \mid p, q, r \ge 0\}$

$L_2 = \{0^p\, 1^q\, 0^r \mid p, q, r \ge 0, p \ne r\}$

34. Since $f(n) = \log n$

$a = 2, b = 2$

finding $\log_b a$ using master's method = $\log_2 2 = 1$

Hence, $f(n) = n'$

So, $T(n) = O(n)$

35. The time complexity is O (m+ logn)

So C = 1, d = 0, a = 0, b = 1

$0 + 10 \times 1 + 100 \times 1 + 1000 \times 0 = [110]$

36. * Outer loop (for i) executed n times.

* Loop (for j) executed logn times so value of p = logn.

loop (for k) executed log p times.

So value of q = log p = log logn.

for every value of i loop (k) is executed so return value is n * log logn

37. Unsorted array

The algorithm perform

find operation $(\log N)^{1/2}$

Insert operation N

delete operation $(\log N)^{1/2}$

decrease key operations $(\log N)^{1/2}$

Hence unsorted array is best data structure for all above operations.

38. The subtrees are already Max-heap, so to make it half, we have to heapify the root, which takes $\Omega(\log n)$ time.

39. Suppose a list contains n elements, consider first three element and find middle element which will be neither maximum nor minimum.

Hence it is $\Theta(1)$.

40. I or III or IV but not II

as $\sum_{i=0}^{n} i^3 \Rightarrow \frac{n^2(n+1)^2}{4}$,

this can be represent by $Q(n^4)$, $O(n^5)$ 4 $\Omega(n^3)$ but not $Q(n^5)$.

41. As $-1 \le \sin x \le 1$, neither of them is true

42. The worst case recurrence reaction for flow chart is

$T(n) = 5\,T(n/2) + 1$

Apply master's method

$n^{\log_2^5} \Rightarrow n^{2.32} > 1$

$T(n) = O(n^{2.32})$

$\boxed{\text{So} \propto = 2.32}$

43. Delete operation require O (1) time total O (N) Delete so time is = O (n).

Insert require O (n) time in worst case total O (logn) insert so time is = O (Nlogn) to Search (logn) key time is = nlogn.

To perform decrease key we need O (n) time (because after decrease we need to arrange elements in sorted sequerce) so total O (N) decrease key So total time is O (N^2) All operations put together than worst time is O (N^2)

44. As per the given algorithms and time complexities we derive:

Towers of Hanoi with n disks

$= 2T(n-1) + 1 = \Theta(2^n)$

Binary search given n sorted numbers

$= T(n/2) + 1 = \Theta(\log n)$

Heap sort given n numbers at the worst case

$= 2T(n/2) + n = \Theta(n \log n)$

Addition of two $n \times n$ matrices

$= 4T(n/2) + 1 = \Theta(n^2)$

45. $T(n) = 2T\left(\sqrt{n}\right) + 1 \quad n > 2$ ...(*i*)

$T(n) = 2T\left(\sqrt[2]{n}\right) + 1$ ...(*ii*)

Put (*ii*) in (*i*)

$T(n) = 2 * 2T\left(\sqrt[2]{n}\right) + 2$

$T(n) = 2^2 T\left(\sqrt[2]{n}\right) + 2$ ...(*iii*)

$T\left(\sqrt[2]{n}\right) = 2T\left(\sqrt[3]{n}\right) + 1$ ...(*iv*)

Substituting (*iv*) in (*iii*)

$T(n) = 2^3 T\left(\sqrt[3]{n}\right) + 3$

Running the same till K times,

$T(n) = 2^K T\left(\sqrt[K]{n}\right) + K$

$\sqrt[K]{n} = 2$

$K = \log_2 n$

Solving this will give

$T(n) = \Theta(\log n)$

46. By the given C function

First loop will execute 'n' times and the inner loop will execute $\Theta(n \log n)$ times.

Hence the complexity will be $\Theta(n \log n)$

47. 10 = growth is o because constant function

$\sqrt{n}$ = growth is slower but faster than logn.

n = growth is linear.

log n = logarithmic growth.

$\frac{100}{n}$ = growth rate decrease with n.

48. The worst number of probes perform by an best algorithm is $\log_2(31) \cong 5$.

49. (a)

50. To find the Maximum and Minimum Elements in an array simultaneously, Ceil(n/2) - 2 comparisons are necessary and sufficient in the worst case to find both the maximum and minimum of n numbers.

So, $t = \text{Ceil}\left(\frac{n}{2}\right) - 2$

Now by applying divide & conquer method, so the number of comparision element are $\left\lceil \frac{3n}{2} - 2 \right\rceil$

Hence, option (c) is correct.

51. As given that $f_1 = 10^n$, $f_2 = n^{\log n}$

$f_3 = n^{\sqrt{n}}$

By taking log of all

$f_1 = n\log 10 = n.c$

$f_2 = \log n \log n = (\log n)^2$

$f_3 = \sqrt{n}\log n \left(\because \sqrt{n} > \log n\right)$

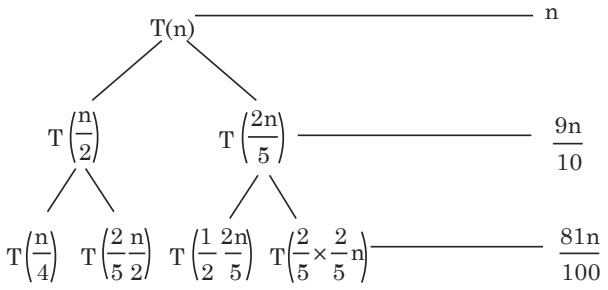
f_1 is exponential function, so grows fastest. And between f_2, f_3 ; f_3 grows faster than f_2.

Hence, the increasing order is $f_1 < f_2 < f_3$.

52. As given recurrence relation:

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{2n}{5}\right) + 7n$$

With recursion tree approach the cost at every level is proportional to n.



$$n + \frac{9}{10}n + \left(\frac{9}{10}\right)^2 n + \ldots \log n \text{ times}$$

$$n\left[1 + \frac{9}{10} + \left(\frac{9}{10}\right)^2 + \ldots \log n \text{ times}\right]$$

$= \Theta(n)$

Hence, option (c) is the correct ansser.

53. Master Theorem

The Master Theorem applies to recurrences of the following form:

$$T(m) = aT\left(\frac{n}{b}\right) + f(n)$$

Where $a \geq 1$ and $b > 1$ are constant and f (n) is an asymptotically positive function.

There are 3 cases:

(i) If $f(n) = O\left(n^{\log_b a - \epsilon}\right)$ for some constant $\epsilon > 0$, then $T(n) = \Theta\left(n^{\log_b a}\right)$.

(ii) If $f(n) = \Theta\left(n^{\log_b a} \log^k n\right)$ with $1k \geq 0$, then $T(n) = \Theta\left(n^{\log_b a} \log^{k+1} n\right)$

(iii) If $f(n) = \Omega\left(n^{\log_b a + \epsilon}\right)$ with $\epsilon > 0$, f(n) satisfier the regularity condition, then $T(n) = \Theta(f(n))$.

Regularity condition: $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n.

Hence, option (c) is the correct answer.

54. Option (a) will always true for polynomial function.

$f(n^2) = \theta(f(n)^2)$, when, f(n) is a polynomial

$$\left.\begin{aligned} f(n) = n^3 \Rightarrow f(n)^2 = (n^2)^3 = n^6 \\ f(n)^2 = (n^3)^2 = n^6 \end{aligned}\right\} \text{equal}$$

Option (b) is not true for the polynomial function.

$f(n^2) = o(f(n)^2)$

$f(n) = n^3 \Rightarrow f(n)^2 = (n^2)^3 = n^6$

$f(n)^2 = (n^3)^2 = n^6$

Option (c) is not true for any exponential function.

$f(n^2) = \Omega(f(n)^2)$ when, f(n) is a exponential

$f(n) = 2^n \Rightarrow f(n)^2 = 2^{(n^2)} \qquad 2^{(n^2)} > 2^{2n}$

$f(n)^2 = (2^n)^2 = 2^{2n}$

Option (d) is not true if the f(n) is log(n), then

$f(n) = n^2 \Rightarrow f(n^2) = f(n^2)$

$$\left.\begin{aligned} f(n) = \log n \Rightarrow f(n)^2 = \log n^2 = 2\log n \\ f(n)^2 = (\log n)^2 \end{aligned}\right\} 2\log n < (\log n)^2$$

So $f(n^2) = \Omega(f(n)^2)$

55. $f(n) \in O(g(n))$ iff $f(n)$ asymptotically smaller or equal to $g(n)$ ....(A)

$f(n) \in o(g(n))$ iff $f(n)$ asymptotically smaller than $g(n)$ ....(B)

As Given function, $f(n) = n$, $g(n) = n^2$

By taking each options:

(a) $f \in O(g)$

$n \in O(n^2)$($\therefore$ True, according to (A))

(b) $f \in \Omega(g)$

$n \in \Omega(n^2)$ ($\therefore$ False)

(c) $f \in o(g)$

$n \in o(n^2)$($\therefore$ True, According to (B))

(d) $f \in \theta(g)$

$n \in \theta(n^2)$ ($\therefore$ False)

56. According to question

Function 1

```
while n > 1 do
  for i = 1 to n do
    x = x + 1;
  end for
  n = ⌊n / 2⌋;
end while
```

$f_1(n)$: Number of times x = x + 1 executes,

For Ist while loop- for loop will execute n times.

For IInd while loop- for loop will execute $\frac{n}{2}$ times

For IIIrd while loop- for loop will execute $\frac{n}{2^2}$ times

$\vdots$

For $(k + 1)^{th}$ while loop- for loop will execute $\frac{n}{2^k}$ times

So, total no. of time: $f_1(n) = n + \frac{n}{2} + \frac{n}{2^2} + ... + \frac{n}{2^k}$

where $\frac{n}{2^k} > 1$, $n > 2^k$

Then $k \approx \log_2 n$

$f_1(n) = n\left[1 + \frac{1}{2} + \frac{1}{2^2} + ... + \frac{1}{2^k}\right] = \theta(n)$

Function 2

```
for i = 1 to 100 * n do
  x = x + 1;
end for
```

$f_2(n)$: Number of times x = x + 1 executes, for loop will execute $100 \times n$ times.

So, $f_2(n) = 100 * n = \theta(n)$

$\therefore$ $f_1(n) = \theta(f_2(n))$

$f_1(n) = O(n)$ are correct

57. The objective of the algorithm is to check whether the array is sorted or not, and it does so by making a sinlge pass through the array.

For this the worst case will be when the array is already sorted.

So, for this worst case, the algorithm has to traverse the whole array and in doing so it will take $O(n)$ time when specified in $O(\cdot)$ notation of $\Omega(n)$ time when specified in $\Omega(\cdot)$ notation.

The above array can be sorted using bubble sort, which will take one iteration is it will take $O(n)$, $\Omega(n)$ which is $\theta(n)$.

58. As given that,

$T(0) = 1$

$T(1) = 2$

$T(n) = 5T(n-1) - 6T(n-2)$; $n \geq 2$

So, $T(2) = 5T(1) - 6T(0) = 5 * 2 - 6 * 1 = 4 = 2^2$

$T(3) = 5T(2) - 6T(1) = 5 * 4 - 6 * 2 = 8 = 2^3$

$T(4) = 5T(3) - 6T(2) = 5 * 8 - 6 * 4 = 16 = 2^4$

$T(5) = 5T(4) - 6T(3) = 5 * 16 - 6 * 8 = 32 = 2^5$ ... so on

$\therefore$ $T(n) = \theta(2^n)$,

Alternative :

Characteristic equation = $t^2 - 5t + 6 = 0$

Characteristic root = $t = 3, 2$

Complimentary function

$$= C_1(t_1)^n + C_2(t_2)^n + (n)$$

$$T(n) = C_1 2^n + C_3 3^n$$

$1 = C_1 + C_2$ {T(0) = 1}

$2 = 2C_1 + 3C_2$ {T(1) = 2}

$C_1 = 1, C_2 = 0$

$T(n) = 1 \times 2^n + 0 \times 3^n$

$T(n) = 2^n$

Hence, $T(n) = \theta(2^n)$

59. As given recurrence relation

$$T(n) = \begin{cases} \sqrt{n}.T(\sqrt{n}+n) & \text{for } n \geq 1 \\ 1 & \text{for } n = 1 \end{cases}$$

$T(n) = n^{1/2} . T(n^{1/2}) + n$

$$T(n) = n^{1/2}\left[n^{\frac{1}{2^2}} T\left(n^{\frac{1}{2^2}} \right) + n^{1/2} \right] + n$$

$$T_{(n)} = n^{\frac{3}{2^2}} .T\left(n^{\frac{1}{2^2}} \right) + n + n$$

$$T(n) = n^{\frac{3}{2^2}}\left[n^{\frac{1}{2^3}} + \left(n^{\frac{1}{2^3}} \right) + n^{\frac{1}{2^2}} \right] + 2n$$

$$T(n) = n^{\frac{7}{2^3}} .T\left(n^{\frac{1}{2^3}} \right) + 3n \text{ : k times}$$

$\{\therefore K = 1, 2, 3...\}$

$$T(n) = n^{\frac{2^{k-1}}{2^k}} .T\left(n\frac{1}{2^k} \right) + k.n \qquad ...(i)$$

Let us consider;

$$\left\{ \begin{array}{l} \therefore n^{\frac{1}{2^k}} = 2 \\ 2^k = \log_2 n \\ k = \log_2 \log_2 n \end{array} \right\}$$

Now, putting all values in equation (i);

$$T(n) = n\left(1 - \frac{1}{2}k \right)\frac{T(2)}{2} + \bar{n}\log_2 \log_2 n$$

$$T(n) = \left[\frac{n}{n^{\frac{1}{2k}}} .T(2) + n.\log_2 \log_2 n \right]$$

$$T(n) = \left[\frac{n}{2} T(1) + n.\log_2 \log_2 n \right] = \theta(n \log_2 \log_2 n)$$

Hence, option (a) is the correct answer.

60. $x = \dfrac{n}{\log n}$

$y = \log n$

$$\text{we get } O\left(\frac{n}{109n} * \log n * \log\log n \right)$$

So O (n log logn)

61. Minimum no of swap for selection sort is O when array is already sorted and maximum swap is Q(n).

62. Merge sort has lowest worst – case complexity, i.e. O(n log n), whereas all remaining three has O(n^2).

63. Time complexity of radix sort is = O (nd)

Where n = keys, d = maximum digit in keys.

$$= d = \log (n^k)$$
$$O(nd) = O(n * k \log n)$$
$$= k \times O(n \log n)$$
$$= O(n \log n)$$

64. If there are n elements, then in worst case, total swaps will be (n-1) in selection sort. So the number of swaps is θ(n).

Alternately

The selection sort is similar to bubble sort except it does not swap elements with every move. The sorting algorithm first finds the smallest element in the list and then puts it in to place in single swap. So n swap required for n elements.

65. Recurrence relation for merge sort

T (n) = 2 T (n/2) + n

So T (n) = Q (n logn)

but instead of integers whose comparison take O (1) time, we are given n strings so to compare strings we need Q (n) time hence complexity is Q (n^2 logn)

66. **Given:** n numbers

To find: Tighest upper bound on number of swaps required to sort n numbers using selection sort.

Nalysis: In selection sort, in the unsorted part of the array, we find the minimum element and swap it with the value placed at the index where the unsorted array starts.

Hence, for each element to put it in its sorted position, we will do some swaps. In each itenation, when we find the minimum and place it in its sorted position, we do only one swap.

There are n such iterations, since maximum number of positions to sort is n.

Hence, there are n. 0(1) swaps

$\Rightarrow$ 0(n) swaps.

$\therefore$ The solution is (b)

67. **Given:** n numbers

To find: Tighest upper bound on number of swaps required to sort n numbers using selection sort.

Nalysis: In selection sort, in the unsorted part of the array, we find the minimum element and swap it with the value placed at the index where the unsorted array starts.

Hence, for each element to put it in its sorted position, we will do some swaps. In each

itenation, when we find the minimum and place it in its sorted position, we do only one swap.

There are n such iterations, since maximum number of positions to sort is n.

Hence, there are n. 0(1) swaps

$\Rightarrow$ 0(n) swaps.

$\therefore$ The solution is (b)

68. Let the number of element is "K", they can be sorted in θ(k log k) time.

We try the options in decreasing order of complexity since, we need a tight band i.e. θ

i.e. $\theta(\log n) \cdot \theta\left(\frac{(\log n)}{\log\log n}\right), \theta\left(\sqrt{\log n}\right), \theta(1)$

So if K $\in \theta(\log n)$ time required for loop sort is. θ(k log k) i.e.

$\theta(\log n \times \log\log n)$, But this is not in $\theta(\log n)$

if k $\in \theta\left(\frac{\log n}{\log\log n}\right)$ time required for loop sort:

$$\theta\left(\frac{\log n}{\log\log n} \times \log\left(\frac{\log n}{\log\log n}\right)\right),$$

i.e. $\theta\left(\log n \times \underbrace{\left[\frac{\log\left(\frac{\log n}{\log 1 > n}\right)}{\log\log n}\right]}_{\leq 1}\right)$

69. From the list of given n numbers [say n is even],

Pick up first two elements, compare them

assign Current – min = min of two numbers

Current – max = max of two numbers

From the remaining n – 2 numbers, take pairs wise and follow this process given below.

1. Compare two elements
   Assign min = min of two numbers
   max = max of two numbers
2. Compare min and current – min
   Assign current – min
   = min{current–min,min}
3. Compare max and current – max
   Assign current – max
   = max{current – max, max}

Repeat above procedure for all the remaining pairs of numbers. We can observe that each of pair requires 3 comparisons

1. for finding min and max
2. For updating current – min
3. for updating current – max

But for initial pair we need only one comparison not 3.

$\therefore$ Total number of comparisons = $\frac{3(n-2)}{2} + 1$

$$= \frac{3n}{2} - 3 + 1 = \frac{3n}{2} - 2$$

Here n =100, so number of comparisons = 148.

70. (0.08)

71. The worst case time complexity is O (n^2).

72. T (n) = T (n/2) + T (n/2) + O (n) (for partition)
+ O (n) (for finding median)

On solving above using

Masters Method $\boxed{T(n) = O(n \log n)}$

73. In quicksort a set of number is reduced to sorting two smaller set. We take first element as key value and combine all other with this. A pivot element which splits list into two sublists each of which at least one fifth of element only (B), i.e.

$T(n) \leq T(n/5) + T(4n/5) + n$ the problem.

Alternately

If one sublist contains 1/5 elements other contains 4/5 elements.

If T(n) number of comparisons for sorting n elements.

So, for 1/5 elements = T(1/5n)

and for 4/5 elements = T(4n/5)

So, $T(n) \leq T(n/5) + T(4n/5) + n$.

Here, n = time to spilt

74. Recurrance relation is

T (n) = T (n/4) + T (3n/4) + n

On solving using tree's method.

$\boxed{T(n) = O(n \log n)}$

75. Partition algorithm for quick sort

Partition (A, P, q)// A [P, ...q]

$x \leftarrow$ A [P] // pivot = A [P]

$i \leftarrow$ P

for j = P + 1 to q

do if A [j] ≤ x

then $i \leftarrow i + 1$

exchange A [i] ↔ A [j]

exchange A [P] ↔ A [i]

return i [returning where pivot element is there after partitioning]

Recursively call the above algorithm for the two sub arrays [elements before and after pivot element] to complete the sorting.

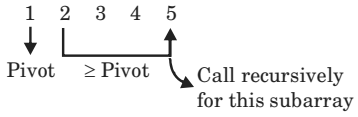
x = pivot

1 2 3 4 5 i j	2 ≤ 1 ? NO	
1 2 3 4 5 i j	3 ≤ 1 ? NO	
1 2 3 4 5 i j	4 ≤ 1 ? NO	
1 2 3 4 5 i j	5 ≤ 1 ? NO	

↗ Pivot =x = A [B]

1 2 3 4 5 i j	3 ≤ 2 ? NO
1 2 3 4 5 i j	4 ≤ 2 ? NO
1 2 3 4 5 i j	5 ≤ 2 ? NO

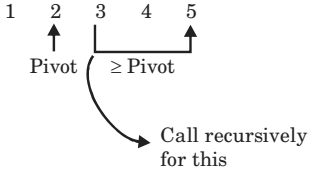
exchange A [P] & A [i]



↗ x = Pivot = A [P]

1 2 3 4 5 i j	4 ≤ 3 ? NO
1 2 3 4 5 i j	5 ≤ 3 ? NO

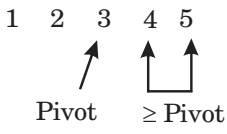
exchange A [P] & A [J]



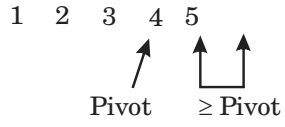
↗ x = Pivot

1 2 3 4 5 i j	5 ≤ 4 ? NO

exchange A [P] & A [i]



exchange A [P] & A[i]



∴ total 10 conparisons

↗ x = Pivot = A [P]

4 1 5 3 2

i j $1 \le 4$? Yes

$i \leftarrow i + 1$ exchange A[i] & A[j] & increment j

4 1 5 3 2

i j $5 \le 4$? NO

4 1 5 3 2

i j $3 \le 4$? Yes

$i \leftarrow i + 1$ exchange A[i] & A[j] & increment j

4 1 3 5 2

i j $2 \le 4$? Yes

$i \leftarrow i + 1$

4 1 3 2 5

i j

exchange A[P] & A[i]

x = pivot = A (P)

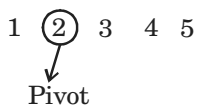
2 1 3 |4| 5

i j $1 \le 2$? Yes

2 1 3 |4| 5

i j $3 \le 2$? NO

exchange A [P] & A[i]



∴ 6 conparisons

76. The Worst case time complexity of quick sort is O (n^2). This will happen when the elements of the input array are already in order (ascending or descending), irrespective of position of pivot element in array.

77. The recurrence equation is

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

In quick sort, worst case divide the list into two list with 1 and $(n-1)$ elements each time.

78. When we put (a, left–End, K) and (a+ left – end + 1, n– left – end –1, k– left–end –1)at the place of fill in the blank alqorithm work for partition.

79. Worst Case

Insertion sort → $O(n^2)$

Merge sort → $O(n \log n)$

Quick sort → $O(n^2)$

80. Complexity is remain same because we had swap after finding to position of element so in worst case swapirg for each element requires Q (n) time hence complexihy is O (n^2).

81. If array of size n with n inversion then complexity of insation sort is Q (n)

Let us take an example.

(6 1 2 3 5 6 4)

total no. of comparision require to

Sort above array is 2 (n–1) = 2* 5 = 10

so complexity is Q (n).

82.

Algorithm	Recurrence Relation
P. Binary Search	IV. T (n) = T (n/2) + 1
Q. Merge Sort	III. T (n) = 2T (n/2) + cn
R. Quick Sort	I. T (n) = T (n-k) + T (K)
S. Tower of Hanoi	II T (n) = 2T (n-1)+1

83. Time complexty is O (nlogn)

C * 64 log 64 = 30

(C = 5/14)

for 6 minuts

$$\frac{5}{64} * n \log n = 6 * 60 \Rightarrow [n = 512]$$

84. 1. Quicksort will take worst case, if the input is in ascending order i.e $\Theta(n^2)$

2. Insertion sort runs in $\Theta(n)$ time.

85. The expected number of comparision is n.

86. Recurrance relation for procedure A (n) is

$T(n) = T\sqrt{n} + 1$ if $n > 2$

$T(n) = 1$ if $n \le 2$

$T(n) = 1$

Now, $T(n) = T\sqrt{n} + 1$

Put $n = 2^k$, $T(2^k) = T(2^{k/2}) + 1$

Use S (K) for T (2^K) then

$S(K) = S(k/2) + 1$

Apply masters method.

$K^{\log 2 1} \cong 1$

So θ (log k)
Now we know that
$n = 2^k$ so, k= $\log_2 n$
So, O (log logn).

87. There are $\frac{n(n-1)}{2}$ Pairs so average inversions are

$$\frac{n(n-1)}{2} * \frac{1}{2} = \frac{n(n-1)}{4}$$

88. Given program fragment in Pseudo language is

$x = m$;
$y = 1$;
while $(x - y > e)$
$x = (x + y)/2$;
$y = m/x$;
}
Print (x)
This program will find out the square root of m, suppose that m = 2

X –	Y	X	Y
Ist looping	2	3/2 = 1.5	2/1.5 = 1.33
2nd looping	.16	$\frac{1.5+1.3}{2} = 1.415$	$\frac{2.0}{1.415} = 1.413$
3rd looping	0.02	1.414	1.414
4th looping	0		

as $x - y = 0$ exit from loop hence print 1.414 which is root of 2.

89. Union & intersection is Slowest opeeration among all.

90. Bellman ford = O (n*m)
Kruskal's Alqo = O (m logn)
Floyd war shall = O (n^3)
Topological sort = O (n+m)

91. The best algorithm is we can solve in linear time using right to left pass.

92. a_1, a_2 .....a_n
b_1, b_2 .....b_n
In order to find out the largest span, check the sums of
a_i +.....a_j and b_i +.....b_j at each step.
If $a_1 + a_2 = b_1 + b_2$ go on, check $a_1 + a_2 + a_3$ and $b_1 + b_2 + b_3$.
If not, then check $a_2 + a_3$ and $b_2 + b_3$
Similarly a check is done at each of the n places during traversal. A separate variable has to be kept that contains the maximum span observed hitherto.
Hence fastest algorithm computes with (~) (n) time and space.

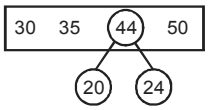
93. The statement inside the for loop is similar to X–OR operation such that the set obtained is $(X \cap Y') \cup (X' \cap Y)$.

It is equivalent to $(X - Y) \cup (Y - X)$.

94. In case of unweighted, undirected graphs, BFS gives the most time efficient computation for shortest path. It is guaranteed to find first shortest path.

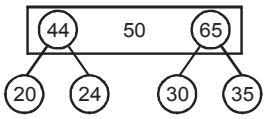
95. Non-increasing order of $\frac{w_i}{t_i}$

96. 20, 24, 30, 35, 50
Step 1 : We take



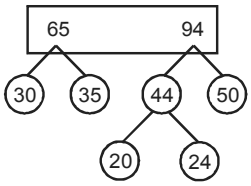
Comparisons = (20 + 24 – 1) Max = 43

Step 2 :



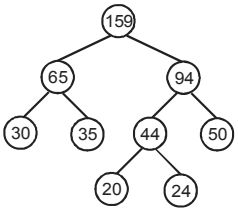
Total comparison = 43 + (65 – 44) = 64

Step 3 :



Total comparison = 64 + 19 = 83

Step 4 :



Total comparison = 93 + 65 = 158
= 158 + 93 + 64 + 43 = 358

97. By definition, T(9) = Dist. From 9 to 100

As given, T(9) = 1+min (T(y), T()(z) = 1 + min (Dist. from y to 100, Dist. From z to 100)

$\Rightarrow$ 1 = Dist. from 9 to y/Dist. From 9 to z

$\Rightarrow$ There are only two such values-one is the simple one step on number line i.e. 10, and the other is the shortcut associated with 9 i.e. 15.

$\Rightarrow$ Therefore, y and z are 10 and 15 (in any order)

$\Rightarrow$ Product yz = 150.

98. Prims algorithm is greedy method because it choose minimum distance node.

Floyd warshal's is dynamic because it changes the distance on each iteration.

Merge sort is divide and conquer because it divide the hole list is sublist and combine after sorting each sublist Hamiltonian cycle is based on backtracking.

99. Dijkstra use gready approach, warshals use dynamic programming approach, Binary search is based on divide and conquer and Back tracking is depth first search.

100. Dynamic programming

101. P – (ii)
Cl – (i)
R – (iii)

102. The given array is already sorted in ascending order.

So, for already sorted array we can used these sort techniques:

(i) Selection sort :

No matter how the data is arranged there would always be comparisons and swaps made and so the time complexity for best, average and worst case is : O(n^2).

In first pass, we need n–1 comparisons (Or n comparisons, depending on the implementation)

In second pass, we need n–2 comparisons (Or n–1 comparisons, depending on the implementation) and so on.

So, The number of comparisons required by a selection sort of n items can be computed by the formula :

$$(n-1) + (n-2) + \ldots + 1 = (n)\frac{(n-1)}{2}$$

or

Number of selection sort comparisons

$$= (n+1)\frac{(n)}{2}$$

Basically, number of comparisons are Θ(n^2) in all cases.

(ii) Insertions Sort :

When elements are sorted, there are no swaps and the correct position of the element in the sorted list is the current index itself. The time complexity is : O(n). Insertion sort takes least Number of comparisons = n – 1

Comparisons in total : 1 + 1 + ... + 1
= n – 1∈ Θ(n).

(iii) Merge Sort :

We are dividing the list into two no matter if the list is sorted or no. But if the array is sorted, while merging the list there are no swaps merging results into an array itself. Thus, the best, average and worst case time complexity is : O(nlogn).

Number of comparisons, in all cases, will be O(nlogn).

(iv) Quick Sort :

The best case is when the elements are in a sorted manner. The best and average case time complexity is : O(nlogn).

Number of comparisons, in best case, will be O(nlogn).

Since, for a number which is to be inserted in the already sorted array, only one comparison will be required.

So, answer will be insertion sort.

Hence, option (d) is correct answer.