Chapter 3

Divide-and-conquer

LEARNING OBJECTIVES

- *Divide-and-conquer*
- *Divide-and-conquer examples*
- *Divide-and-conquer technique*
- *Merge sort*
- *Quick sort*
- *Performance of quick sort*
- *Recurrence relation*
- *Searching*
- *Linear search*
- *Binary search*

Divide-and-Conquer

Divide-and-conquer is a top down technique for designing algorithms that consists of dividing the problem into smaller sub problems hoping that the solutions of the sub problems are easier to find and then composing the partial solutions into the solution of the original problem.

Divide-and-conquer paradigm consists of following major phases:

- Breaking the problem into several sub-problems that are similar to the original problem but smaller in size.
- Solve the sub-problem recursively (successively and independently)
- Finally, combine these solutions to sub-problems to create a solution to the original problem.

Divide-and-Conquer Examples

- Sorting: Merge sort and quick sort
- Binary tree traversals
- Binary Search
- Multiplication of large integers
- Matrix multiplication: Strassen's algorithm
- Closest-pair and Convex-hull algorithm

Merge Sort

Merge sort is a sorting algorithm for rearranging lists (or any other data structure that can only be accessed sequentially, e.g., file streams) into a specified order.

Merge sort works as follows:

1. Divide the unsorted list into two sub lists of about half the size.
2. Sort each of the two sub lists.

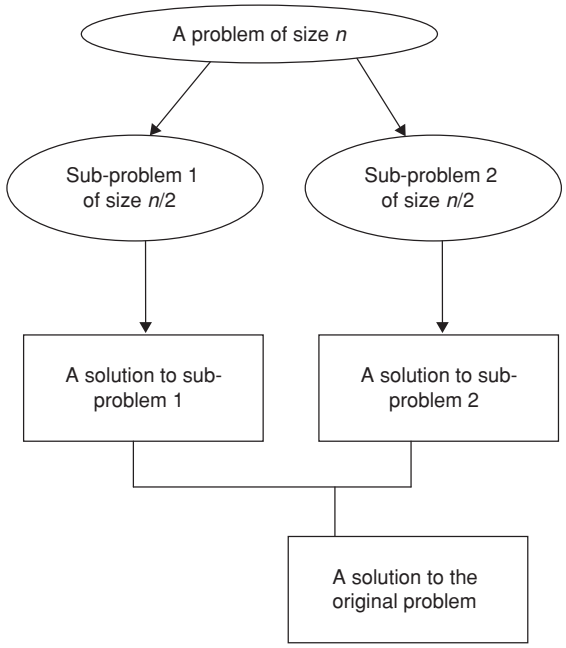


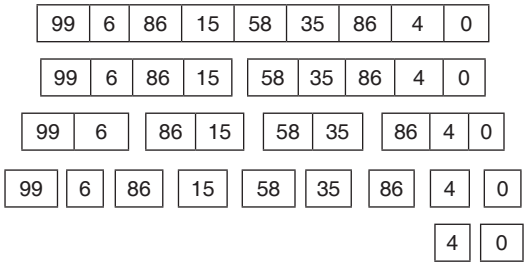
Figure 1 Divide-and-conquer technique.

3. Merge the two sorted sub lists back into one sorted list
4. The key of merge sort is merging two sorted lists into one, such that if we have 2 lists
   $X(x_1 \le x_2 \le x_3 \cdots \le x_m)$ and
   $Y(y_1 \le y_2 \le y_3 \cdots \le y_n)$ the resulting list is $z(z_1 \le z_2 \le \cdots \le z_{m+n})$

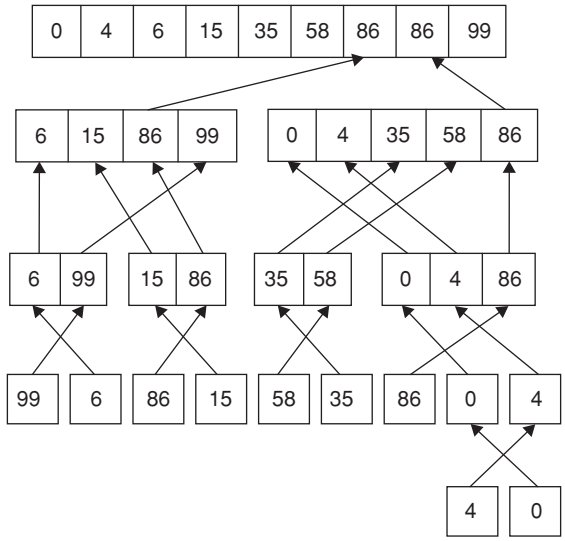
Example 1: $L_1 = \{3, 8, 9\}$, $L_2 = \{1, 5, 7\}$

Merge $(L_1, L_2) = \{1, 3, 5, 7, 8, 9\}$

Example 2:



Merge:



Implementing Merge Sort

Merging is done with a temporary array of the same size as the input array.

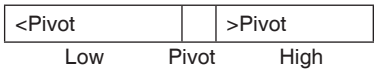
Pro: Faster than in-place since the temp array holds the resulting array until both left and right sides are merged into the temp array then the temp array is appended over the input array.

Con: The memory required is doubled. The double memory merge sort runs $O(N \log N)$ for all cases, because of its Divide-and-conquer approach.

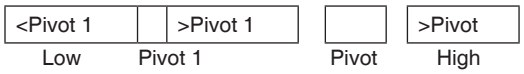
$$T(N) = 2T(N/2) + N$$
$$= O(N \log N)$$

Quick Sort

Quick sort is an example of Divide-and-conquer strategy. In Quick sort we divide the array of items to be sorted into two partitions and then call the quick sort procedure recursively to sort the two partitions, i.e., we divide the problem into two smaller ones and conquer by solving the smaller ones. The conquer part of the quick sort routine looks like this



Make bold



Divide: Partition the array A $[p - r]$ into 2 sub arrays A $[p - q - 1]$ and A $[q + 1 - r]$ such that each element of A $[p - q - 1]$ is less than or equal to A$[q]$, which is, in turn, less than or equal to each element of A $[q + 1 - r]$

Conquer: Sort the 2 sub arrays A $[p - q - 1]$ and A $[q + 1 - r]$ by recursive calls to quick sort.

Combine: Since the sub arrays are sorted inplace, no work is needed to combine them.

Sort left partition in the same way. For this strategy to be effective, the partition phase must ensure that the pivot, is greater than all the items in one part (the lower part) and less than all those in the other (upper) part. To do this, we choose a pivot element and arrange that all the items in the lower part are less than the pivot and all those in the upper part are greater than it. In the general case, the choice of pivot element is first element.

(Here $\lfloor$number of elements/2$\rfloor$ is pivot)

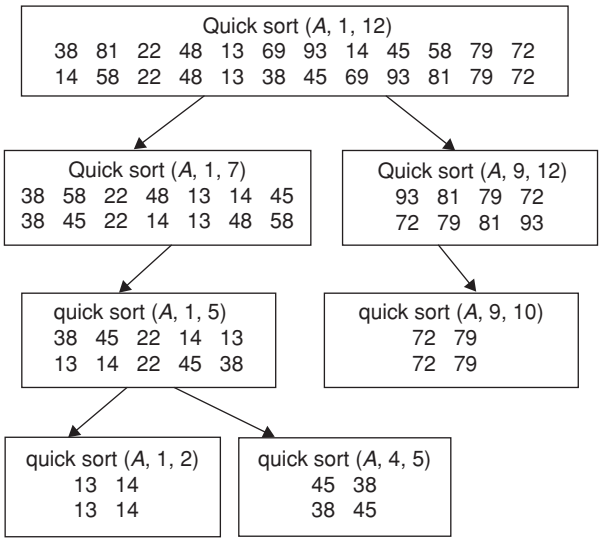


Figure 2 Tree of recursive calls to quick sort.

- Quick sort is a sorting algorithm with worst case running time $O(n^2)$ on an input array of n numbers. Inspite of this slow worst case running time, quick sort is often the best practical choice for sorting because it is efficient on the average: its expected running time is $O(n \log n)$ and the constants hidden in the O-notation are quite small
- Quick sort algorithm is fastest when the median of the array is chosen as the pivot element. This is because the resulting partitions are of very similar size. Each partition splits itself in two and thus the base case is reached very quickly.

Example: Underlined element is pivot.

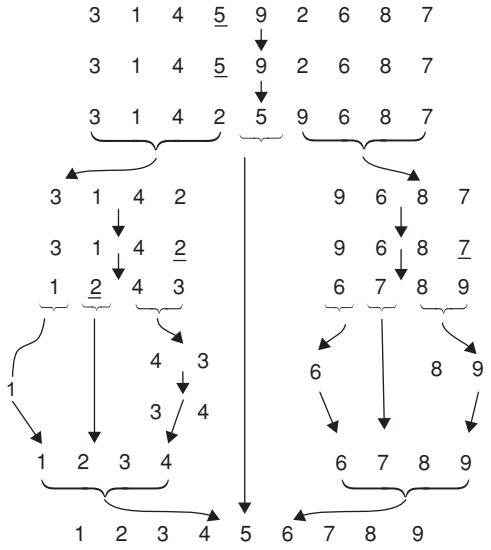


Figure 3 The ideal quick sort on a random array

Performance of Quick Sort

- Running time of quick sort depends on whether the partitioning is balanced or unbalanced, it depends on which elements are used for partitioning. If the partitioning is balanced, the algorithm runs asymptotically as fast as merge sort. If the partitioning is unbalanced, it runs as slowly as insertion sort.
- The worst case of quick sort occurs when the partitioning routine produces one sub-problem with $n-1$ elements and one with '1' element. If this unbalanced partitioning arises in each recursive call, the partitioning costs $\theta(n)$ time.

Recurrence Relation

$$T(n) = T(n-1) + T(1) + \theta(n)$$
$$(\therefore\ T(0) = \theta(1))$$
$$= T(n-1) + \theta(n)$$

If we sum the costs incurred at each level of the recursion we get an arithmetic series, which evaluates to $\theta(n^2)$.

- Best case partitioning–PARTITION produces 2 sub problems, each of size no more than $n/2$, since one is of size $\lfloor n/2 \rfloor$ and one of size $\lceil n/2 \rceil - 1$

The recurrence for the running time is then

$$T(n) \le 2T(n/2) + \theta(n)$$

The above Recurrence relation has the solution $T(n) = O(n \log n)$ by case 2 of the master theorem.

- The average–case time of quick sort is much closer to the best than to the worst case

For example, that the partitioning algorithm always produces a 8-to-2 proportional split, which at first seems unbalanced. The Recurrence relation will be

$$T(n) \le T(8n/10) + T(2n/10) + cn$$

The recursion tree for this recurrence has cost 'cn' at every level, until a boundary condition is reached at depth $\log_{10} n = \theta(\log n)$. The recursion terminates at depth $\log_{10/8} n = \theta(\log n)$. The total cost of quick sort is $O(n \log n)$

Searching

Two searching techniques are:

- Linear search
- Binary search

Linear Search

Linear search (or) sequential search is a method for finding a particular value in list that consists of checking every one of its elements, one at a time and in sequence, until the desired one is found. Linear search is a special case of brute force search. Its worst case cost is proportional to the number of elements in the list.

Implementation

boolean linear search `(int [ ] arr, int target)`

```
{
int i = 0;
while (i < arr. length) {
if (arr [i] = = target){
return true;
}
+ + i;
}
return false;
}
```

Example:

Consider the array

10	7	1	3	–4	2	20

Search for 3

10	7	1	3	–4	2	20

3?

Move to next element

10	7	1	3	–4	2	20

3?

Move to next element

10	7	1	3	–4	2	20

3?

Move to next element

10	7	1	3	–4	2	20

3?

Element found; stop the search.

Binary Search

A binary search algorithm is a technique for finding a particular value in a linear array, by ruling out half of the data at each

step; a binary search finds the median, makes comparison, to determine whether the desired value comes before or after it, and then searches the remaining half in the same manner. A binary search is an example of Divide-and-conquer algorithm.

Implementation

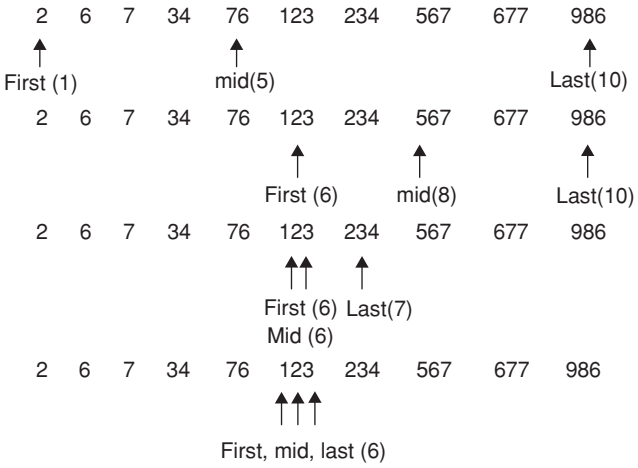
function binary search (a, value, left, right)

```
{
if right < left
       return not found
      mid: = floor ((right -left)/2) + left
      if a [mid] = value
      return mid
      if value < a[mid]
```

return binary search (a, value, left, mid –1) else return binary search (a, value, mid + 1, right)

}

Example: Value being searched 123



Exercises

Practice Problems I

Directions for questions 1 to 15: Select the correct alternative from the given choices.

1. How many comparisons are required to search an item 89 in a given list, using Binary search?

4	8	19	25	34	39	45	48	66	75	89	95
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]

(A) 3 (B) 4
(C) 5 (D) 6

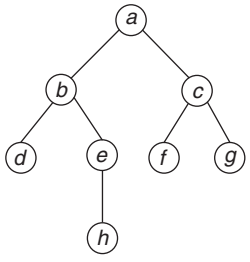
2. Construct a Binary search tree with the given list of elements:

300, 210, 400, 150, 220, 370, 450, 100, 175, 215, 250

Which of the following is a parent node of element 250?

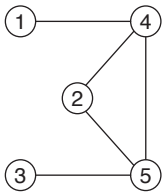
(A) 220
(B) 150
(C) 370
(D) 215

3. What is the breadth first search order of the given tree?



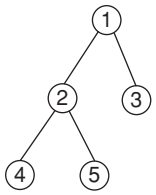
(A) acbhdefg (B) abcdefgh
(C) adbcefgh (D) aebcdfgh

4. What is the depth first search order of the given graph?



(A) 14325
(B) 12435
(C) 14253
(D) 12354

5. When pre-order traversal is applied on a given tree, what is the order of elements?

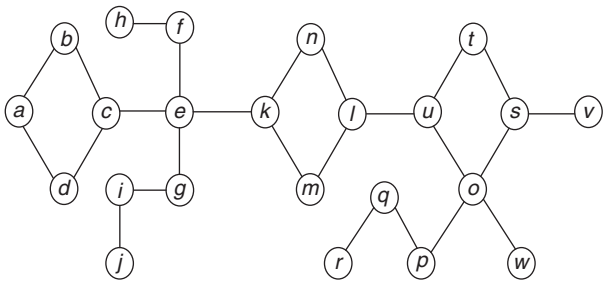


(A) 1 – 2 – 4 – 5 – 3
(B) 1 – 4 – 2 – 5 – 3
(C) 1 – 2 – 4 – 3 – 5
(D) 1 – 2 – 3 – 4 – 5

6. What is the order of post-order traversal and in-order traversals of graph given in the above question?

(A) 4 – 2 – 5 – 1 – 3 and 4 – 5 – 2 – 3 – 1
(B) 4 – 5 – 2 – 3 – 1 and 4 – 2 – 5 – 1 – 3
(C) 4 – 5 – 2 – 1 – 3 and 4 – 2 – 5 – 1 – 3
(D) 4 – 5 – 2 – 3 – 1 and 4 – 2 – 5 – 3 – 1

7. Find the number of bridges in the given graph

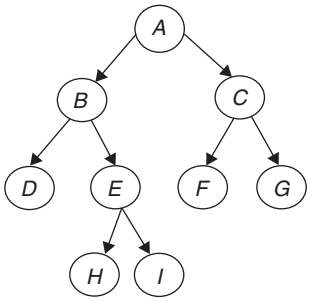


(A) 12 (B) 13
(C) 11 (D) 10

8. Match the following:

I.	In-order	1.	*ABCDEFGHI*
II.	Pre-order	2.	*DBHEIAFCG*
III.	Post-order	3.	*ABDEHICFG*
IV.	Level-order	4.	*DHIEBFGCA*

For the tree



(A) I – 2, II – 3, III – 4, IV – 1
(B) I – 3, II – 1, III – 4, IV – 2
(C) I – 1, II – 2, III – 3, IV – 4
(D) I – 4, II – 3, III – 2, IV – 1

9. A complete *n*-array tree in which each node has '*n*' children (or) no children.

Let '*I*' be the number of internal nodes and '*L*' be the number of leaves in a complete *n*-ary tree.

If $L = 51$ and $I = 10$ what is the value of '*n*'?

(A) 4 (B) 5
(C) 6 (D) Both (A) and (B)

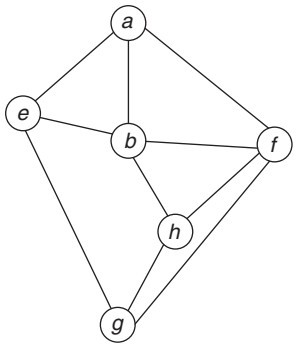
10. A complete *n*-ary tree is one in which every node has 0 (or) *n* children. If '*X*' is the number of internal nodes of a complete *n*-ary tree, the number of leaves in it is given by

(A) $X(n-1)+1$ (B) $Xn-1$
(C) $Xn+1$ (D) $X(n+1)+1$

11. The numbers 7, 5, 1, 8, 3, 6, 0, 9, 4, 2 are inserted in the given order into an initially empty binary search tree. The binary search tree uses the usual ordering on natural numbers. What is the in-order traversal sequence of the resultant tree?

(A) 7 5 1 0 3 2 4 6 8 9
(B) 0 2 4 3 1 6 5 9 8 7
(C) 0 1 2 3 4 5 6 7 8 9
(D) 9 8 6 4 2 3 0 1 5 7

12. Consider the following graph:



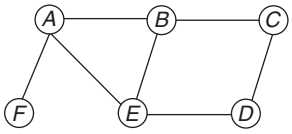
Among the following sequences

I. *a b e g h f* II. *a b f e h g*
III. *a b f h g e* IV. *a f g h b e*

Which are depth first traversals of the above graph?

(A) I, II and IV only (B) I and IV only
(C) I, III only (D) I, III and IV only

13. The breadth first search algorithm has been implemented using the queue data structure. One possible order of visiting the nodes is



(A) *A B C D E F* (B) *B E A D C F*
(C) *E A B D F C* (D) Both (A) and (B)

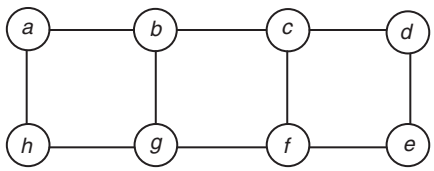
14. An undirected graph *G* has '*n*' nodes. Its adjacency matrix is given by an $n \times n$ square matrix.

(i) Diagonal elements are 0's
(ii) Non-diagonal elements are 1's

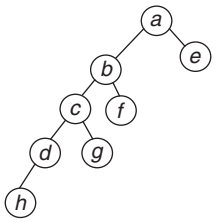
Which of the following is true?

(A) Graph *G* has no minimum spanning tree
(B) Graph *G* has a unique minimum spanning tree of cost $(n-1)$
(C) Graph *G* has multiple distinct minimum spanning trees, each of cost $(n-1)$
(D) Graph *G* has multiple spanning trees of different cost.

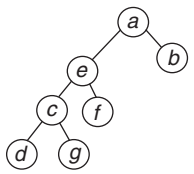
15. Which of the following is the breadth first search tree for the given graph?



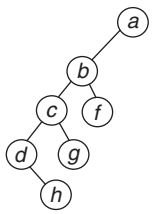
(A)



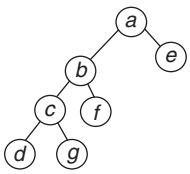
(B)



(C)



(D)



Practice Problems 2

Directions for questions 1 to 15: Select the correct alternative from the given choices.

1. Which of the following algorithm design technique is used in finding all pairs of shortest distances in a graph?
(A) Divide-and-conquer
(B) Greedy method
(C) Back tracking
(D) Dynamic programming

2. Let LASTPOST, LASTIN and LASTPRE denote the last vertex visited in a post-order, in-order and pre-order traversals respectively of a complete binary tree. Which of the following is always true?
(A) LASTIN = LASTPOST
(B) LASTIN = LASTPRE
(C) LASTPRE = LASTPOST
(D) LASTIN = LASTPOST = LASTPRE

3. Match the following:

X : Depth first search

Y : Breadth first search

Z : Sorting

a : Heap

b : Queue

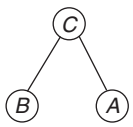
c : Stack

(A) X – a, Y – b, Z – c
(B) X – c, Y – a, Z – b
(C) X – c, Y – b, Z – a
(D) X – a, Y – c, Z – b

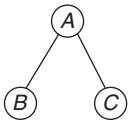
4. Let G be an undirected graph, consider a depth first traversal of G, and let T be the resulting DFS Tree. Let 'U' be a vertex in 'G' and let 'V' be the first new (unvisited) vertex visited after visiting 'U' in the traversal. Which of the following is true?
(A) $\{U, V\}$ must be an edge in G and 'U' is a descendant of V in T.
(B) $\{U, V\}$ must be an edge in 'G' and V is a descendant of 'U' in T.
(C) If $\{U, V\}$ is not an edge in 'G' then 'U' is a leaf in T.
(D) if $\{U, V\}$ is not an edge in G then U and V must have the same parent in T.

5. Identify the binary tree with 3 nodes labeled A, B and C on which preorder traversal gives the sequence C, B, A.

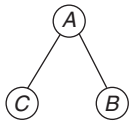
(A)



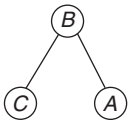
(B)



(C)



(D)

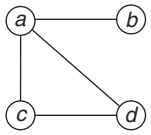


6. Consider an undirected unweighted graph G. Let a breadth first traversal of G be done starting from a node r. Let $d(r, u)$ and $d(r, v)$ be the lengths of the shortest paths from r to u and v respectively in 'G'. If u is visited before v during the breadth first travel, which of the following is correct?
(A) $d(r, u) < d(r, v)$ (B) $d(r, u) > d(r, v)$
(C) $d(r, u) \leq d(r, v)$ (D) None of these

7. In a complete 5-ary tree, every internal node has exactly 5 children. The number of leaves in such a tree with '3' internal nodes are:
(A) 15 (B) 20
(C) 13 (D) Can't predicted

8. Which of the following algorithm is single pass that is they do not traverse back up the tree for search, create, insert etc.
(A) Depth first search (B) Pre-order traversal
(C) B-tree traversal (D) Post-order traversal

9. Which of the following is the adjacency matrix of the given graph?



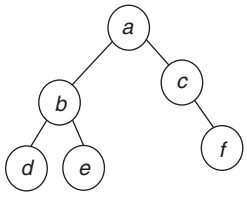
(A) $\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$

(B) $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

(C) $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}$

(D) $\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$

10. Which one of the following is the post-order traversal of the given tree?

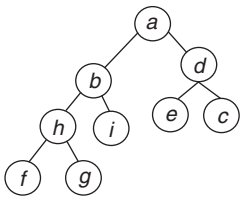


(A) *d e a f c b a* (B) *d e b f c a*
(C) *e b d f c a* (D) *a b c d e f*

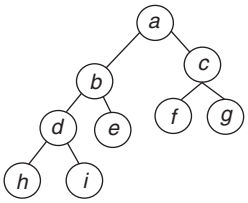
Common data for questions 11 and 12:

11. The pre-order traversal of a tree is *a b d h i e c f g*. Which of the following is the correct tree?

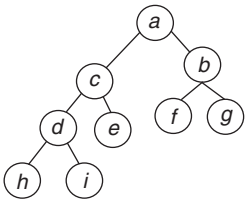
(A)



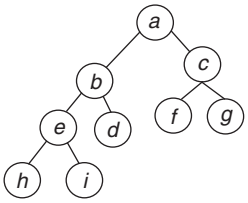
(B)



(C)



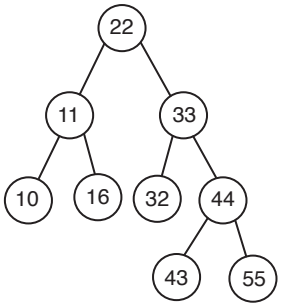
(D)



12. Which of the following is in-order traversal of the above tree?

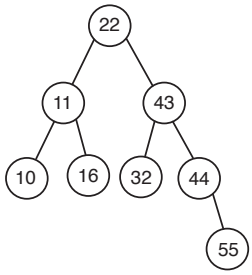
(A) *a b h d e i f g c* (B) *a b d h e i f g c*
(C) *h d i b e a f c g* (D) *i d h b e a f c g*

13. Consider the below binary search tree

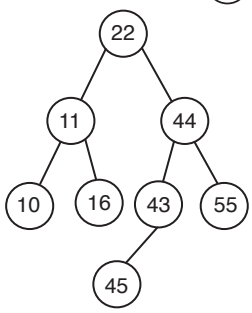


Which of the following is the resultant binary search tree after deletion of 33?

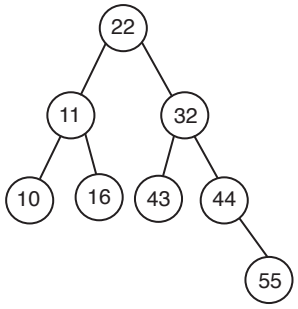
(A)



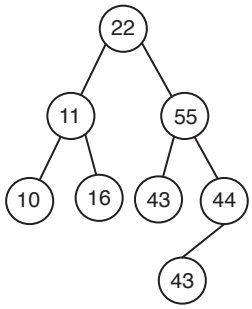
(B)



(C)



(D)



14. Match the following:

I.	Articulation Point	1.	An edge whose removal disconnects graph
II.	Bridge	2.	A vertex whose removal disconnects graph
III.	Bi connected component	3.	Maximal set of edges such that any two edges in the set lie on a common simple cycle

(A) I – 1, II – 2, III – 3 (B) I – 2, II – 1, III – 3
(C) I – 2, II – 3, III – 1 (D) I – 1, II – 2, III – 3

15. If x is the root of an n-node subtree, then the inorder-tree-walk takes

(A) $\theta(n)$ (B) $\theta(n^2)$
(C) $\theta(n^3)$ (D) $\theta(n \log n)$

Previous Years' Questions

1. Which one of the following is the tightest upper bound that represents the time complexity of inserting an object into a binary search tree of n nodes? **[2013]**

(A) $O(1)$ (B) $O(\log n)$
(C) $O(n)$ (D) $O(n \log n)$

2. Consider a rooted n node binary tree represented using pointers. The best upper bound on the time required to determine the number of sub trees having exactly 4 nodes is $O(n^a \log^b n)$. The value of $a + 10b$ is _____ **[2014]**

3. 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. **[2015]**

(A) $T(n) = 2T(n/2) + cn$
(B) $T(n) = T(n-1) + T(1) + cn$
(C) $T(n) = 2T(n-1) + cn$
(D) $T(n) = T(n/2) + cn$

4. 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 all 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 $k'' n$.

int $k$th_smallest (int a [ ], int n, int k) **[2015]**

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

The missing argument lists are respectively

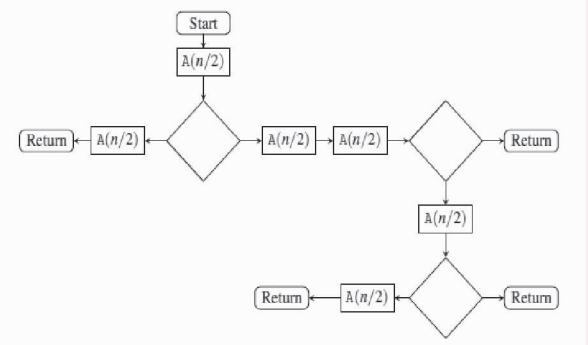
(A) (a, left_end, k) and (a+left_end+1, n-left_end-1, k-left_end-1)
(B) (a, left_end, k) and (a, n-left_end-1, k-left_end-1)
(C) (a+left_end+1, n-left_end-1, k-left_end-1) and (a, left_end, k)
(D) (a, n-left_end-1, k-left_end-1) and (a, left_end, k)

5. 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? **[2015]**

(A) 256 (B) 512
(C) 1024 (D) 2048

6. The given diagram shows the flowchart 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^α), then the least possible value (accurate up to two decimal positions) of α is ____. **[2016]**

Flowchart for Recursive Function A(n)



7. Let A be an array of 31 numbers consisting of a sequence of 0's followed by a sequence of 1'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]**

8. Match the algorithms with their time complexities:

Algorithm	Time complexity
(P) Towers of Hanoi with n disks	(i) $\Theta(n^2)$
(Q) Binary search given n sorted numbers	(ii) $\Theta(n \log n)$
(R) Heap sort given n numbers at the worst case	(iii) $\Theta(2^n)$
(S) Addition of two $n \times n$ matrices	(iv) $\Theta(\log n)$

[2017]

(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)

Answer Keys

Exercises

Practice Problems 1

1. A	**2.** A	**3.** B	**4.** C	**5.** A	**6.** B	**7.** B	**8.** A	**9.** C	**10.** A
11. C	**12.** D	**13.** A	**14.** C	**15.** A					

Practice Problems 2

1. B	**2.** B	**3.** C	**4.** B	**5.** A	**6.** D	**7.** C	**8.** C	**9.** A	**10.** B
11. B	**12.** C	**13.** A	**14.** B	**15.** A					

Previous Years' Questions

1. C	**2.** 1	**3.** B	**4.** A	**5.** B	**6.** 2.2 to 2.4	**7.** 5	**8.** C