Array

1. Which of these best describes an array?
a) A data structure that shows a hierarchical behavior
b) Container of objects of similar types
c) Arrays are immutable once initialised
d) Array is not a data structure

Answer: b
Explanation: Array contains elements only of the same type.

2. How do you initialize an array in C?
a) int arr[3] = (1,2,3);
b) int arr(3) = {1,2,3};
c) int arr[3] = {1,2,3};
d) int arr(3) = (1,2,3);

Answer: c
Explanation: This is the syntax to initialize an array in C.

7. When does the ArrayIndexOutOfBoundsException occur?
a) Compile-time
b) Run-time
c) Not an error
d) Not an exception at all

Answer: b
Explanation: ArrayIndexOutOfBoundsException is a run-time exception and the compilation is error-free.

8. Which of the following concepts make extensive use of arrays?
a) Binary trees
b) Scheduling of processes
c) Caching
d) Spatial locality

Answer: d
Explanation: Whenever a particular memory location is referred to, it is likely that the locations nearby are also referred, arrays are stored as contiguous blocks in memory, so if you want to access array elements, spatial locality makes it easy to access quickly.

9. What are the advantages of arrays?
a) Objects of mixed data types can be stored
b) Elements in an array cannot be sorted

c) Index of first element of an array is 1
d) Easier to store elements of same data type

Answer: d
Explanation: Arrays store elements of the same data type and present in continuous memory locations.

10. What are the disadvantages of arrays?
a) Data structure like queue or stack cannot be implemented
b) There are chances of wastage of memory space if elements inserted in an array are lesser than the allocated size
c) Index value of an array can be negative
d) Elements are sequentially accessed

Answer: b
Explanation: Arrays are of fixed size. If we insert elements less than the allocated size, unoccupied positions can't be used again. Wastage will occur in memory.

11. Assuming int is of 4bytes, what is the size of int arr[15];?
a) 15
b) 19
c) 11
d) 60

Answer: d
Explanation: Since there are 15 int elements and each int is of 4 bytes, we get 15*4 = 60 bytes

12. In general, the index of the first element in an array is _____
a) 0
b) -1
c) 2
d) 1

Answer: a
Explanation: In general, Array Indexing starts from 0. Thus, the index of the first element in an array is 0.

13. Elements in an array are accessed _____
a) randomly
b) sequentially
c) exponentially
d) logarithmically

Answer: a
Explanation: Elements in an array are accessed randomly. In Linked lists, elements are accessed sequentially.

Stack

3. In a stack, if a user tries to remove an element from an empty stack it is called:
a) Underflow
b) Empty collection
c) Overflow
d) Garbage Collection

Answer: a
Explanation: Underflow occurs when the user performs a pop operation on an empty stack. Overflow occurs when the stack is full and the user performs a push operation. Garbage Collection is used to recover the memory occupied by objects that are no longer used.

4. Pushing an element into stack already having five elements and stack size of 5, then stack becomes _____
a) Overflow
b) Crash
c) Underflow
d) User flow

Answer: a
Explanation: The stack is filled with 5 elements and pushing one more element causes a stack overflow. This results in overwriting memory, code and loss of unsaved work on the computer.

5. Entries in a stack are “ordered”. What is the meaning of this statement?
a) A collection of stacks is sortable
b) Stack entries may be compared with the ‘<‘ operation
c) The entries are stored in a linked list
d) There is a Sequential entry that is one by one

Answer: d
Explanation: In stack data structure, elements are added one by one using push operation. Stack follows LIFO Principle i.e. Last In First Out(LIFO).

6. Which of the following is not the application of stack?
a) A parentheses balancing program
b) Tracking of local variables at run time
c) Compiler Syntax Analyzer
d) Data Transfer between two asynchronous process

Answer: d
Explanation: Data transfer between the two asynchronous process uses the queue data structure for synchronisation. The rest are all stack applications.

7. Consider the usual algorithm for determining whether a sequence of parentheses is balanced. The maximum number of parentheses that appear on the stack AT ANY ONE TIME when the algorithm analyzes: (()(())(())))?
a) 1
b) 2
c) 3
d) 4 or more

Answer: c
Explanation: In the entire parenthesis balancing method when the incoming token is a left parenthesis it is pushed into stack. A right parenthesis makes pop operation to delete the elements in stack till we get left parenthesis as top most element. 3 elements are there in stack before right parentheses comes. Therefore, maximum number of elements in stack at run time is 3.

8. Consider the usual algorithm for determining whether a sequence of parentheses is balanced. Suppose that you run the algorithm on a sequence that contains 2 left parentheses and 3 right parentheses (in some order). The maximum number of parentheses that appear on the stack AT ANY ONE TIME during the computation?
a) 1
b) 2
c) 3
d) 4 or more

Answer: b
Explanation: In the entire parenthesis balancing method when the incoming token is a left parenthesis it is pushed into stack. A right parenthesis makes pop operation to delete the elements in stack till we get left parenthesis as top most element. 2 left parenthesis are pushed whereas one right parenthesis removes one of left parenthesis. 2 elements are there before right parenthesis which is the maximum number of elements in stack at run time.

9. What is the value of the postfix expression 6 3 2 4 + – *?
a) 1
b) 40
c) 74
d) -18

Answer: d
Explanation: Postfix Expression is (6*(3-(2+4))) which results -18 as output.

10. Here is an infix expression: 4 + 3*(6*3-12). Suppose that we are using the usual stack algorithm to convert the expression from infix to postfix notation. The maximum number of symbols that will appear on the stack AT ONE TIME during the conversion of this expression?
a) 1
b) 2
c) 3
d) 4

Answer: d
Explanation: When we perform the conversion from infix to postfix expression +, *, (, * symbols are placed inside the stack. A maximum of 4 symbols are identified during the entire conversion.

1. The postfix form of the expression (A+ B)*(C*D- E)*F / G is?
a) AB+ CD*E – FG /**
b) AB + CD* E – F **G /
c) AB + CD* E – *F *G /
d) AB + CDE * – * F *G /

Answer: c
Explanation: (((A+ B)*(C*D- E)*F) / G) is converted to postfix expression as

(AB+(*(C*D- E)*F)/ G)

(AB+CD*E-*F) / G

(AB+CD*E-*F * G/). Thus Postfix expression is AB+CD*E-*F*G/

2. The data structure required to check whether an expression contains a balanced parenthesis is?
a) Stack
b) Queue
c) Array
d) Tree

Answer: a
Explanation: The stack is a simple data structure in which elements are added and removed based on the LIFO principle. Open parenthesis is pushed into the stack and a closed parenthesis pops out elements till the top element of the stack is its corresponding open parenthesis. If the stack is empty, parenthesis is balanced otherwise it is unbalanced.

3. What data structure would you most likely see in non recursive implementation of a recursive algorithm?
a) Linked List

b) Stack
c) Queue
d) Tree

Answer: b
Explanation: In recursive algorithms, the order in which the recursive process comes back is the reverse of the order in which it goes forward during execution. The compiler uses the stack data structure to implement recursion. In the forwarding phase, the values of local variables, parameters and the return address are pushed into the stack at each recursion level. In the backing-out phase, the stacked address is popped and used to execute the rest of the code.

4. The process of accessing data stored in a serial access memory is similar to manipulating data on a _____
a) Heap
b) Binary Tree
c) Array
d) Stack

Answer: d
Explanation: In serial access memory data records are stored one after the other in which they are created and are accessed sequentially. In stack data structure, elements are accessed sequentially. Stack data structure resembles the serial access memory.

5. The postfix form of A*B+C/D is?
a) *AB/CD+
b) AB*CD/+
c) A*BC+/D
d) ABCD+/*

Answer: b
Explanation: Infix expression is (A*B)+(C/D)

AB*+(C/D)

AB*CD/+. Thus postfix expression is AB*CD/+.

6. Which data structure is needed to convert infix notation to postfix notation?
a) Branch
b) Tree
c) Queue
d) Stack

Answer: d
Explanation: The Stack data structure is used to convert infix expression to postfix expression. The purpose of stack is to reverse the order of the operators in the expression. It also serves as a storage structure, as no operator can be printed until both of its operands have appeared.

7. The prefix form of A-B/ (C * D ^ E) is?
a) -/*^ACBDE
b) -ABCD*^DE
c) -A/B*C^DE
d) -A/BC*^DE

Answer: c
Explanation: Infix Expression is (A-B)/(C*D^E)

(-A/B)(C*D^E)

-A/B*C^DE. Thus prefix expression is -A/B*C^DE.

8. What is the result of the following operation?
Top (Push (S, X))
a) X
b) X+S
c) S
d) XS

Answer: a
Explanation: The function Push(S,X) pushes the value X in the stack S. Top() function gives the value which entered last. X entered into stack S at last.

9. The prefix form of an infix expression (p + q) – (r * t) is?
a) + pq – *rt
b) – +pqr * t
c) – +pq * rt
d) – + * pqrt

Answer: c
Explanation: Given Infix Expression is ((p+q)-(r*t))

(+pq)-(r*t)

(-+pq)(r*t)

-+pq*rt. Thus prefix expression is -+pq*rt.

10. Which data structure is used for implementing recursion?
a) Queue
b) Stack
c) Array
d) List

Answer: b
Explanation: Stacks are used for the implementation of Recursion.

1. The result of evaluating the postfix expression 5, 4, 6, +, *, 4, 9, 3, /, +, * is?
a) 600
b) 350
c) 650
d) 588

Answer: b
Explanation: The postfix expression is evaluated using stack. We will get the infix expression as

(5*(4+6))*(4+9/3). On solving the Infix Expression, we get

(5*(10))*(4+3)

= 50*7

= 350.

2. Convert the following infix expressions into its equivalent postfix expressions.
(A + B ∧D)/(E – F)+G
a) (A B D ∧ + E F – / G +)
b) (A B D +∧ E F – / G +)
c) (A B D ∧ + E F/- G +)
d) (A B D E F + ∧ / – G +)

Answer: a
Explanation: The given infix expression is (A + B ∧D)/(E – F)+G.

(A B D ^ +) / (E – F) +G

(A B D ^ + E F –) + G. ‘/’ is present in stack.

A B D ^ + E F – / G +. Thus Postfix Expression is A B D ^ + E F – / G +.

3. Convert the following Infix expression to Postfix form using a stack.
x + y * z + (p * q + r) * s, Follow the usual precedence rule and assume that the expression is legal.
a) xyz*+pq*r+s*+
b) xyz*+pq*r+s+*
c) xyz+*pq*r+s*+
d) xyzp+**qr+s*+

Answer: a
Explanation: The Infix Expression is x + y * z + (p * q + r) * s.

(x y z) + (p * q + r) * s. ‘+’, ‘*’ are present in stack.

(x y z * + p q * r) * s. ‘+’ is present in stack.

x y z * + p q * r + s * +. Thus Postfix Expression is x y z * + p q * r + s * +.

4. Which of the following statement(s) about stack data structure is/are NOT correct?
a) Linked List are used for implementing Stacks
b) Top of the Stack always contain the new node
c) Stack is the FIFO data structure
d) Null link is present in the last node at the bottom of the stack

Answer: c
Explanation: Stack follows LIFO.

5. Consider the following operation performed on a stack of size 5.

Push(1);
Pop();
Push(2);
Push(3);
Pop();
Push(4);
Pop();
Pop();
Push(5);

After the completion of all operation, the number of elements present in stack is?
a) 1
b) 2
c) 3
d) 4

Answer: a
Explanation: Number of elements present in stack is equal to the difference between number of push operations and number of pop operations. Number of elements is 5-4=1.

6. Which of the following is not an inherent application of stack?
a) Reversing a string
b) Evaluation of postfix expression
c) Implementation of recursion
d) Job scheduling

Answer: d
Explanation: Job Scheduling is not performed using stacks.

7. The type of expression in which an operator succeeds its operands is?
a) Infix Expression
b) Prefix Expression
c) Postfix Expression
d) Both Prefix and Postfix Expressions

Answer: c
Explanation: The expression in which operator succeeds its operands is called postfix expression. The expression in which operator precedes the operands is called prefix expression. If an operator is present between two operands, then it is called infix expressions.

8. Assume that the operators +,-, x are left associative and ^ is right associative. The order of precedence (from highest to lowest) is ^, x, +, -. The postfix expression for the infix expression a + b x c – d ^ e ^ f is?
a) a b c x + d e f ^ ^ –
b) a b c x + d e ^ f ^ –
c) a b + c x d – e ^ f ^
d) – + a x b c ^ ^ d e f

Answer: a
Explanation: Given Infix Expression is a + b x c – d ^ e ^ f. And ^ is right associative. Thus, the final postfix expression is a b c x + d e f ^ ^ –

9. If the elements “A”, “B”, “C” and “D” are placed in a stack and are deleted one at a time, what is the order of removal?
a) ABCD
b) DCBA
c) DCAB
d) ABDC

Answer: b
Explanation: Stack follows LIFO(Last In First Out). So the removal order of elements are DCBA.

Queue

1. A linear list of elements in which deletion can be done from one end (front) and insertion can take place only at the other end (rear) is known as _____
a) Queue
b) Stack
c) Tree
d) Linked list

Answer: a
Explanation: Linear list of elements in which deletion is done at front side and insertion at rear side is called Queue. In stack we will delete the last entered element first.

2. The data structure required for Breadth First Traversal on a graph is?
a) Stack
b) Array
c) Queue
d) Tree

Answer: c
Explanation: In Breadth First Search Traversal, BFS, starting vertex is first taken and adjacent vertices which are unvisited are also taken. Again, the first vertex which was added as an unvisited adjacent vertex list will be considered to add further unvisited vertices of the graph. To get the first unvisited vertex we need to follow the First In First Out principle. Queue uses FIFO principle.

3. A queue follows _____
a) FIFO (First In First Out) principle
b) LIFO (Last In First Out) principle
c) Ordered array
d) Linear tree

Answer: a
Explanation: Element first added in queue will be deleted first which is FIFO principle.

4. Circular Queue is also known as _____
a) Ring Buffer
b) Square Buffer
c) Rectangle Buffer
d) Curve Buffer

Answer: a
Explanation: Circular Queue is also called as Ring Buffer. Circular Queue is a linear data structure in which last position is connected back to the first position to make a circle. It forms a ring structure.

5. If the elements “A”, “B”, “C” and “D” are placed in a queue and are deleted one at a time, in what order will they be removed?
a) ABCD
b) DCBA
c) DCAB
d) ABDC

Answer: a
Explanation: Queue follows FIFO approach. i.e. First in First Out Approach. So, the order of removal elements are ABCD.

6. A data structure in which elements can be inserted or deleted at/from both ends but not in the middle is?
a) Queue
b) Circular queue
c) Dequeue
d) Priority queue

Answer: c
Explanation: In dequeuer, we can insert or delete elements from both the ends. In queue, we will follow first in first out principle for insertion and deletion of elements. Element with least priority will be deleted in a priority queue.

7. A normal queue, if implemented using an array of size MAX_SIZE, gets full when?
a) Rear = MAX_SIZE – 1
b) Front = (rear + 1)mod MAX_SIZE
c) Front = rear + 1
d) Rear = front

Answer: a
Explanation: When Rear = MAX_SIZE – 1, there will be no space left for the elements to be added in queue. Thus queue becomes full.

8. Queues serve major role in _____
a) Simulation of recursion
b) Simulation of arbitrary linked list
c) Simulation of limited resource allocation
d) Simulation of heap sort

Answer: c
Explanation: Simulation of recursion uses stack data structure. Simulation of arbitrary linked lists uses linked lists. Simulation of resource allocation uses queue as first entered data needs to be given first priority during resource allocation. Simulation of heap sort uses heap data structure.

9. Which of the following is not the type of queue?
a) Ordinary queue
b) Single ended queue
c) Circular queue
d) Priority queue

Answer: b
Explanation: Queue always has two ends. So, a single ended queue is not the type of queue.

1. A linear collection of data elements where the linear node is given by means of pointer is called?
a) Linked list
b) Node list
c) Primitive list
d) Unordered list

Answer: a
Explanation: In Linked list each node has its own data and the address of next node. These nodes are linked by using pointers. Node list is an object that consists of a list of all nodes in a document with in a particular selected set of nodes.

2. Consider an implementation of unsorted singly linked list. Suppose it has its representation with a head pointer only. Given the representation, which of the following operation can be implemented in O(1) time?

```
i) Insertion at the front of the linked list
ii) Insertion at the end of the linked list
iii) Deletion of the front node of the linked list
iv) Deletion of the last node of the linked list
```

a) I and II
b) I and III

c) I, II and III
d) I, II and IV

Answer: b
Explanation: We know the head node in the given linked list. Insertion and deletion of elements at the front of the linked list completes in O (1) time whereas for insertion and deletion at the last node requires to traverse through every node in the linked list. Suppose there are n elements in a linked list, we need to traverse through each node. Hence time complexity becomes O(n).

3. In linked list each node contains a minimum of two fields. One field is data field to store the data second field is?
a) Pointer to character
b) Pointer to integer
c) Pointer to node
d) Node

Answer: c
Explanation: Each node in a linked list contains data and a pointer (reference) to the next node. Second field contains pointer to node.

4. What would be the asymptotic time complexity to add a node at the end of singly linked list, if the pointer is initially pointing to the head of the list?
a) O(1)
b) O(n)
c) θ(n)
d) Both O(n) and θ(n)

Answer: d
Explanation: In case of a linked list having n elements, we need to travel through every node of the list to add the element at the end of the list. Thus asymptotic time complexity is both θ(n) and O(n). Θ(n) represents the **tight bound** of the algorithm's time complexity, meaning it captures the best, average, and worst-case scenarios that are all linear in this case. O(n) signifies the **upper bound**, indicating the worst-case scenario is no worse than linear.

5. What would be the asymptotic time complexity to insert an element at the front of the linked list (head is known)?
a) O(1)
b) O(n)
c) O(n2)
d) O(n3)

Answer: a
Explanation: To add an element at the front of the linked list, we will create a new node which holds

the data to be added to the linked list and pointer which points to head position in the linked list. The entire thing happens within O (1) time. Thus the asymptotic time complexity is O (1).

6. What would be the asymptotic time complexity to find an element in the linked list?
a) O(1)
b) O(n)
c) O(n2)
d) O(n4)

Answer: b
Explanation: If the required element is in the last position, we need to traverse the entire linked list. This will take O (n) time to search the element.

7. What would be the asymptotic time complexity to insert an element at the second position in the linked list?
a) O(1)
b) O(n)
c) O(n2)
d) O(n3)

Answer: a
Explanation: A new node is created with the required element. The pointer of the new node points the node to which the head node of the linked list is also pointing. The head node pointer is changed and it points to the new node which we created earlier. The entire process completes in O (1) time. Thus the asymptotic time complexity to insert an element in the second position of the linked list is O (1).

8. The concatenation of two lists can be performed in O(1) time. Which of the following variation of the linked list can be used?
a) Singly linked list
b) Doubly linked list
c) Circular doubly linked list
d) Array implementation of list

Answer: c
Explanation: We can easily concatenate two lists in O (1) time using singly or doubly linked list, provided that we have a pointer to the last node at least one of the lists. But in case of circular doubly linked lists, we will break the link in both the lists and hook them together. Thus circular doubly linked list concatenates two lists in O (1) time.

9. Consider the following definition in the C programming language.

```
struct node
```

```
{
    int data;
    struct node * next;
}
typedef struct node NODE;
```

```
NODE *ptr;
```

Which of the following c code is used to create a new node?
a) ptr = (NODE*)malloc(sizeof(NODE));
b) ptr = (NODE*)malloc(NODE);
c) ptr = (NODE*)malloc(sizeof(NODE*));
d) ptr = (NODE)malloc(sizeof(NODE));

Answer: a

LinkedList

1. What kind of linked list is best to answer questions like “What is the item at position n?”
a) Singly linked list
b) Doubly linked list
c) Circular linked list
d) Array implementation of linked list

Answer: d
Explanation: Arrays provide random access to elements by providing the index value within square brackets. In the linked list, we need to traverse through each element until we reach the nth position. Time taken to access an element represented in arrays is less than the singly, doubly and circular linked lists. Thus, array implementation is used to access the item at the position n.

2. Linked lists are not suitable for the implementation of _____
a) Insertion sort
b) Radix sort
c) Polynomial manipulation
d) Binary search

Answer: d
Explanation: It cannot be implemented using linked lists.

3. Linked list is considered as an example of _____ type of memory allocation.
a) Dynamic
b) Static

c) Compile time
d) Heap

Answer: a
Explanation: As memory is allocated at the run time.

4. In Linked List implementation, a node carries information regarding _____
a) Data
b) Link
c) Data and Link
d) Node

Answer: c
Explanation: A linked list is a collection of objects linked together by references from an object to another object. By convention these objects are names as nodes. Linked list consists of nodes where each node contains one or more data fields and a reference(link) to the next node.

5. Linked list data structure offers considerable saving in _____
a) Computational Time
b) Space Utilization
c) Space Utilization and Computational Time
d) Speed Utilization

Answer: c
Explanation: Linked lists saves both space and time.

6. Which of the following points is/are not true about Linked List data structure when it is compared with an array?
a) Arrays have better cache locality that can make them better in terms of performance
b) It is easy to insert and delete elements in Linked List
c) Random access is not allowed in a typical implementation of Linked Lists
d) Access of elements in linked list takes less time than compared to arrays

Answer: d
Explanation: To access an element in a linked list, we need to traverse every element until we reach the desired element. This will take more time than arrays as arrays provide random access to its elements.

7. What does the following function do for a given Linked List with first node as head?

```
void fun1(struct node* head)
```

```
{

    if(head == NULL)

    return;

    fun1(head->next);

    printf("%d  ", head->data);

}
```

a) Prints all nodes of linked lists
b) Prints all nodes of linked list in reverse order
c) Prints alternate nodes of Linked List
d) Prints alternate nodes in reverse order

Answer: b
Explanation: fun1() prints the given Linked List in reverse manner.

For Linked List 1->2->3->4->5, fun1() prints 5->4->3->2->1.

8. Which of the following sorting algorithms can be used to sort a random linked list with minimum time complexity?
a) Insertion Sort
b) Quick Sort
c) Heap Sort
d) Merge Sort

Answer: d
Explanation: Both Merge sort and Insertion sort can be used for linked lists. The slow random-access performance of a linked list makes other algorithms (such as quicksort) perform poorly, and others (such as heapsort) completely impossible. Since worst case time complexity of Merge Sort is O(nLogn) and Insertion sort is O(n2), merge sort is preferred.

5. In the worst case, the number of comparisons needed to search a singly linked list of length n for a given element is?
a) log 2 n
b) n⁄2

c) log 2 n – 1
d) n

Answer: d
Explanation: In the worst case, the element to be searched has to be compared with all elements of the linked list.

6. Given pointer to a node X in a singly linked list. Only one pointer is given, pointer to head node is not given, can we delete the node X from given linked list?
a) Possible if X is not last node
b) Possible if size of linked list is even
c) Possible if size of linked list is odd
d) Possible if X is not first node

Answer: a

7. You are given pointers to first and last nodes of a singly linked list, which of the following operations are dependent on the length of the linked list?
a) Delete the first element
b) Insert a new element as a first element
c) Delete the last element of the list
d) Add a new element at the end of the list

Answer: c
Explanation: Deletion of the first element of the list is done in O (1) time by deleting memory and changing the first pointer.

Insertion of an element as a first element can be done in O (1) time. We will create a node that holds data and points to the head of the given linked list. The head pointer was changed to a newly created node.

Deletion of the last element requires a pointer to the previous node of last, which can only be obtained by traversing the list. This requires the length of the linked list.

Adding a new element at the end of the list can be done in O (1) by changing the pointer of the last node to the newly created node and last is changed to a newly created node.

8. In the worst case, the number of comparisons needed to search a singly linked list of length n for a given element is?
a) log2 n
b) n⁄2

c) log2 n – 1
d) n

Answer: d
Explanation: The worst-case happens if the required element is at last or the element is absent in the list. For this, we need to compare every element in the linked list. If n elements are there, n comparisons will happen in the worst case.

Trees

1. The number of edges from the root to the node is called _____ of the tree.
a) Height
b) Depth
c) Length
d) Width

Answer: b
Explanation: The number of edges from the root to the node is called depth of the tree.

2. The number of edges from the node to the deepest leaf is called _____ of the tree.
a) Height
b) Depth
c) Length
d) Width

Answer: a
Explanation: The number of edges from the node to the deepest leaf is called height of the tree.

3. What is a full binary tree?
a) Each node has exactly zero or two children
b) Each node has exactly two children
c) All the leaves are at the same level
d) Each node has exactly one or two children

Answer: a
Explanation: A full binary tree is a tree in which each node has exactly 0 or 2 children.

4. What is a complete binary tree?
a) Each node has exactly zero or two children
b) A binary tree, which is completely filled, with the possible exception of the bottom level, which is filled from right to left
c) A binary tree, which is completely filled, with the possible exception of the bottom level, which is

filled from left to right
d) A tree In which all nodes have degree 2

Answer: c
Explanation: A binary tree, which is completely filled, with the possible exception of the bottom level, which is filled from left to right is called complete binary tree. A Tree in which each node has exactly zero or two children is called full binary tree. A Tree in which the degree of each node is 2 except leaf nodes is called perfect binary tree.

5. What is the average case time complexity for finding the height of the binary tree?
a) h = O(loglogn)
b) h = O(nlogn)
c) h = O(n)
d) h = O(log n)

Answer: d
Explanation: The nodes are either a part of left sub tree or the right sub tree, so we don't have to traverse all the nodes, this means the complexity is lesser than n, in the average case, assuming the nodes are spread evenly, the time complexity becomes O(logn).

6. Which of the following is not an advantage of trees?
a) Hierarchical structure
b) Faster search
c) Router algorithms
d) Undo/Redo operations in a notepad

Answer: d
Explanation: Undo/Redo operations in a notepad is an application of stack. Hierarchical structure, Faster search, Router algorithms are advantages of trees.

7. In a full binary tree if number of internal nodes is I, then number of leaves L are?
a) L = 2*I
b) L = I + 1
c) L = I – 1
d) L = 2*I – 1

Answer: b
Explanation: Number of Leaf nodes in full binary tree is equal to 1 + Number of Internal Nodes i.e L = I + 1

8. In a full binary tree if number of internal nodes is I, then number of nodes N are?
a) N = 2*I
b) N = I + 1

c) N = I – 1
d) N = 2*I + 1

Answer: d
Explanation: Relation between number of internal nodes(I) and nodes(N) is N = 2*I+1.

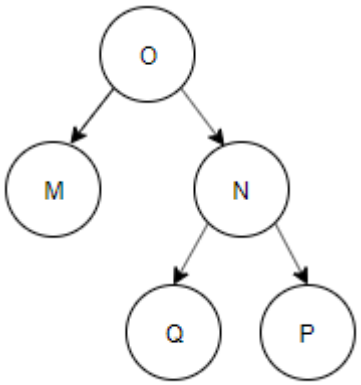
9. In a full binary tree if there are L leaves, then total number of nodes N are?
a) N = 2*L
b) N = L + 1
c) N = L – 1
d) N = 2*L – 1

Answer: d
Explanation: The relation between number of nodes(N) and leaves(L) is N=2*L-1.

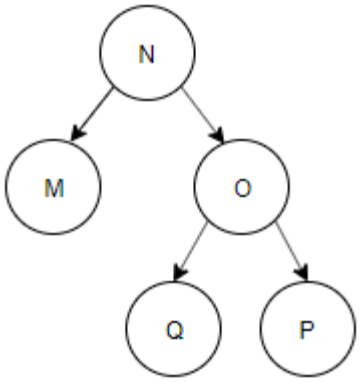
10. Which of the following is incorrect with respect to binary trees?
a) Let T be a binary tree. For every k ≥ 0, there are no more than 2k nodes in level k
b) Let T be a binary tree with λ levels. Then T has no more than 2λ – 1 nodes
c) Let T be a binary tree with N nodes. Then the number of levels is at least ceil(log (N + 1))
d) Let T be a binary tree with N nodes. Then the number of levels is at least floor(log (N + 1))

Answer: d
Explanation: In a binary tree, there are atmost 2k nodes in level k and 2k-1 total number of nodes. Number of levels is at least ceil(log(N+1)).

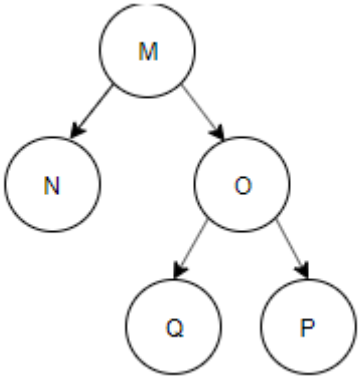
11. Construct a binary tree by using postorder and inorder sequences given below.
Inorder: N, M, P, O, Q
Postorder: N, P, Q, O, M



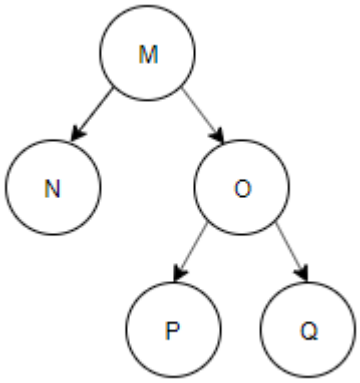
a)



b)



c)



d)

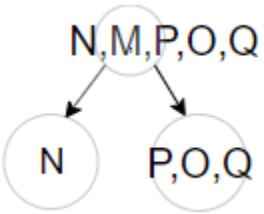
Answer: d
Explanation: Here,

Postorder Traversal: N, P, Q, O, M

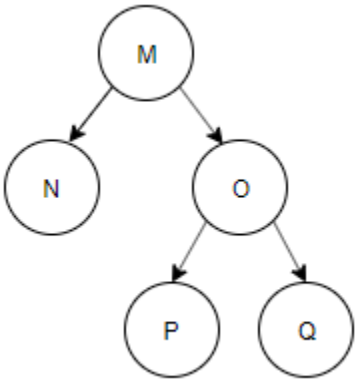
Inorder Traversal: N, M, P, O, Q

Root node of tree is the last visiting node in Postorder traversal. Thus, Root Node = ‘M’.

The partial tree constructed is:

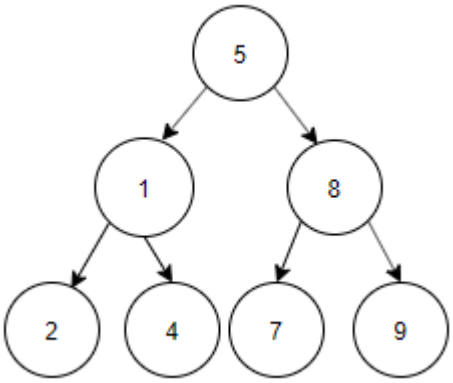


The second last node in postorder traversal is O. Thus, node P becomes left child of node O and node Q becomes right child of node Q. Thus, the final tree is:

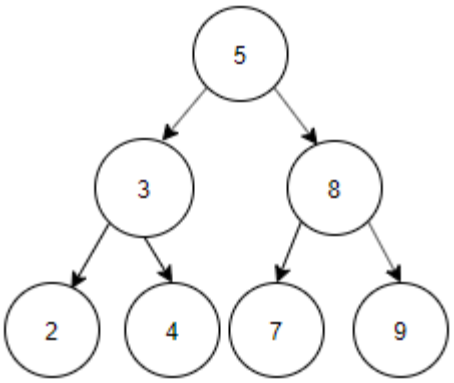


12. Construct a binary search tree by using postorder sequence given below.
Postorder: 2, 4, 3, 7, 9, 8, 5.

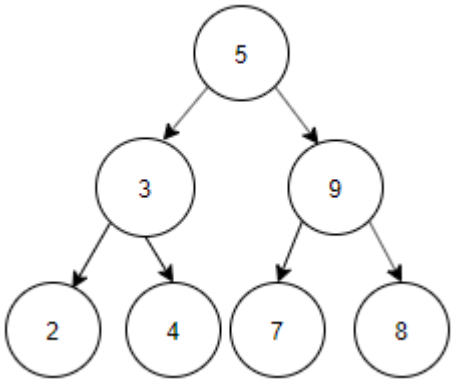
a)



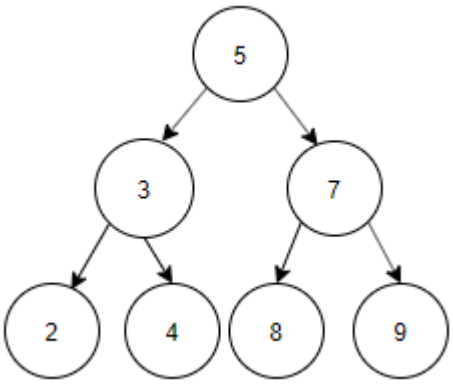
b)



c)



d)

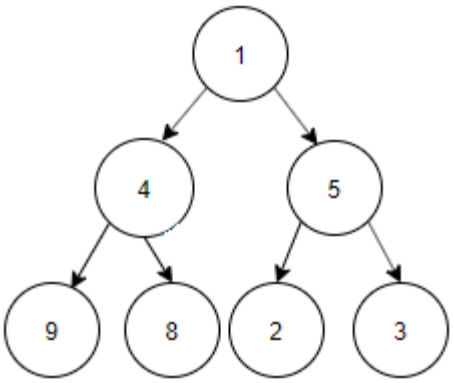


Answer: b
Explanation: Postorder sequence is 2, 4, 3, 7, 9, 8, 5.

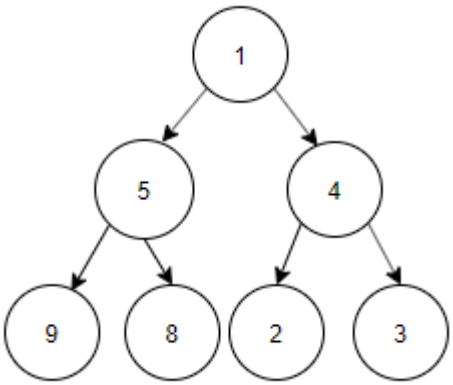
Inorder sequence is the ascending order of nodes in Binary search tree. Thus, Inorder sequence is 2, 3, 4, 5, 7, 8, 9. The tree constructed using Postorder and Inorder sequence is

13. Construct a binary tree using inorder and level order traversal given below.
Inorder Traversal: 3, 4, 2, 1, 5, 8, 9
Level Order Traversal: 1, 4, 5, 9, 8, 2, 3

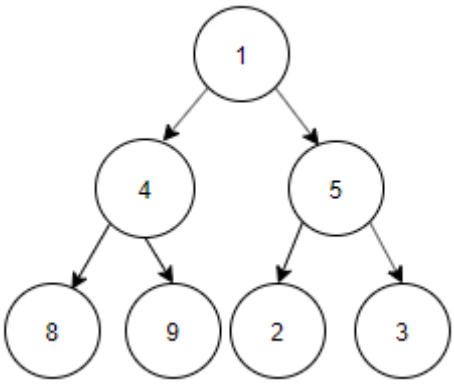
a)



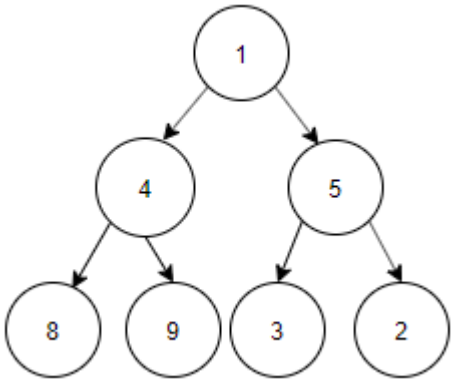
b)



c)



d)

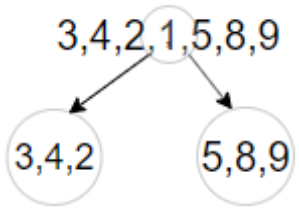


Answer: a
Explanation: Inorder Traversal: 3, 4, 2, 1, 5, 8, 9

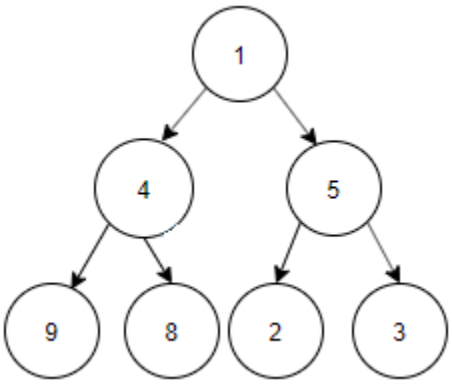
Level Order Traversal: 1, 4, 5, 9, 8, 2, 3

In level order traversal first node is the root node of the binary tree.

Thus the partially formed tree is:



In level order traversal, the second node is 4. Then, node 3 becomes left child of node 4 and node 2 becomes right child of node 4. Third node of level order traversal is 8. Then, node 5 becomes left child of node 8 and node 9 becomes right child of node 8. Thus, the final tree is:



Binary Search Tree

1. Which of the following is false about a binary search tree?
a) The left child is always lesser than its parent
b) The right child is always greater than its parent
c) The left and right sub-trees should also be binary search trees
d) In order sequence gives decreasing order of elements

Answer: d
Explanation: In order sequence of binary search trees will always give ascending order of elements. Remaining all are true regarding binary search trees.

3. What is the speciality about the inorder traversal of a binary search tree?
a) It traverses in a non increasing order
b) It traverses in an increasing order
c) It traverses in a random fashion
d) It traverses based on priority of the node

Answer: b
Explanation: As a binary search tree consists of elements lesser than the node to the left and the ones greater than the node to the right, an inorder traversal will give the elements in an increasing order.

5. What does the following piece of code do?

```
public void func(Tree root)

{

	System.out.println(root.data());

	func(root.left());

	func(root.right());

}
```

a) Preorder traversal
b) Inorder traversal
c) Postorder traversal
d) Level order traversal

Answer: a
Explanation: In a preorder traversal, first the parent is visited, then the left child and finally the right child.

8. What are the worst case and average case complexities of a binary search tree?
a) O(n), O(n)

b) O(logn), O(logn)
c) O(logn), O(n)
d) O(n), O(logn)

Answer: d
Explanation: Worst case arises when the tree is skewed(either to the left or right) in which case you have to process all the nodes of the tree giving O(n) complexity, otherwise O(logn) as you process only the left half or the right half of the tree.

10. What are the conditions for an optimal binary search tree and what is its advantage?
a) The tree should not be modified and you should know how often the keys are accessed, it improves the lookup cost
b) You should know the frequency of access of the keys, improves the lookup time
c) The tree can be modified and you should know the number of elements in the tree before hand, it improves the deletion time
d) The tree should be just modified and improves the lookup time

Answer: a
Explanation: For an optimal binary search The tree should not be modified and we need to find how often keys are accessed. Optimal binary search improves the lookup cost.

1. What will be the height of a balanced full binary tree with 8 leaves?
a) 8
b) 5
c) 6
d) 4

Answer: d
Explanation: A balanced full binary tree with l leaves has height h, where h = log2l + 1.

So, the height of a balanced full binary tree with 8 leaves = log28 + 1 = 3 + 1 = 4.

2. The balance factor of a node in a binary tree is defined as _____
a) addition of heights of left and right subtrees
b) height of right subtree minus height of left subtree
c) height of left subtree minus height of right subtree
d) height of right subtree minus one

Answer: c
Explanation: For a node in a binary tree, the difference between the heights of its left subtree and right subtree is known as the balance factor of the node.

4. A binary tree is balanced if the difference between left and right subtree of every node is not more than ____
a) 1
b) 3
c) 2
d) 0

Answer: a
Explanation: In a balanced binary tree the heights of two subtrees of every node never differ by more than 1.

5. Which of the following tree data structures is not a balanced binary tree?
a) AVL tree
b) Red-black tree
c) Splay tree
d) B-tree

Answer: d
Explanation: All the tree data structures given in options are balanced, but B-tree can have more than two children.

7. Balanced binary tree with n items allows the lookup of an item in ____ worst-case time.
a) O(log n)
b) O(nlog 2)
c) O(n)
d) O(1)

Answer: a
Explanation: Searching an item in balanced binary is fast and worst-case time complexity of the search is O(log n).

8. Which of the following data structures can be efficiently implemented using height balanced binary search tree?
a) sets
b) priority queue
c) heap
d) both sets and priority queue

Answer: d
Explanation: Height-Balanced binary search tree can provide an efficient implementation of sets, priority queues.

11. AVL trees are more balanced than Red-black trees.
a) True
b) False

Answer: a

10. Which of the following is an advantage of balanced binary search tree, like AVL tree, compared to binary heap?
a) insertion takes less time
b) deletion takes less time
c) searching takes less time
d) construction of the tree takes less time than binary heap

Answer: a
Explanation: Insertion and deletion, in both the binary heap and balanced binary search tree takes O(log n). But searching in balanced binary search tree requires O(log n) while binary heap takes O(n). Construction of balanced binary search tree takes O(nlog n) time while binary heap takes O(n).

9. Two balanced binary trees are given with m and n elements respectively. They can be merged into a balanced binary search tree in ____ time.
a) O(m+n)
b) O(mn)
c) O(m)
d) O(mlog n)

Answer: a
Explanation: First we store the in-order traversals of both the trees in two separate arrays and then we can merge these sorted sequences in O(m+n) time. And then we construct the balanced tree from this final sorted array.

Binary Heap

1. What is the space complexity of searching in a heap?
a) O(logn)
b) O(n)
c) O(1)
d) O(nlogn)

Answer: b
Explanation: The space complexity of searching an element in heap is O (n). Heap consists of n elements and we need to compare every element. Here no addition or deletion of elements takes place. Hence space complexity is O (n).

2. What is the best case complexity in building a heap?
a) O(nlogn)
b) O(n2)
c) O(n*longn *logn)
d) O(n)

Answer: d
Explanation: The best case complexity occurs in bottom-up construction when we have a sortes array given.

6. Given an array of element 5, 7, 9, 1, 3, 10, 8, 4. Which of the following are the correct sequences of elements after inserting all the elements in a min-heap?
a) 1,3,4,5,7,8,9,10
b) 1,4,3,9,8,5,7,10
c) 1,3,4,5,8,7,9,10
d) 1,3,7,4,8,5,9,10

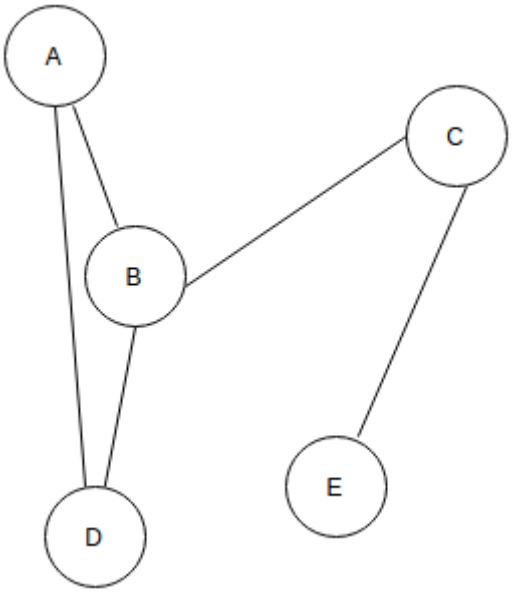
Answer: a
Explanation: Building a min-heap the result will a sorted array so the 1, 3, 4, 5, 7, 8, 9, 10 is correct. If we change the implementation strategy 1, 4, 3, 8, 9, 5, 7, 10 is also correct. (First filling the right child rather than left child first).

Graphs

1. Which of the following statements for a simple graph is correct?
a) Every path is a trail
b) Every trail is a path
c) Every trail is a path as well as every path is a trail
d) Path and trail have no relation

Answer: a
Explanation: In a walk if the vertices are distinct it is called a path, whereas if the edges are distinct it is called a trail.

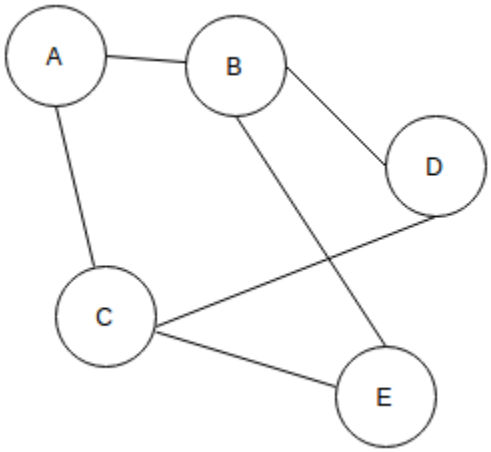
2. In the given graph identify the cut vertices.



a) B and E
b) C and D
c) A and E
d) C and B

Answer: d
Explanation: After removing either B or C, the graph becomes disconnected.

3. For the given graph(G), which of the following statements is true?



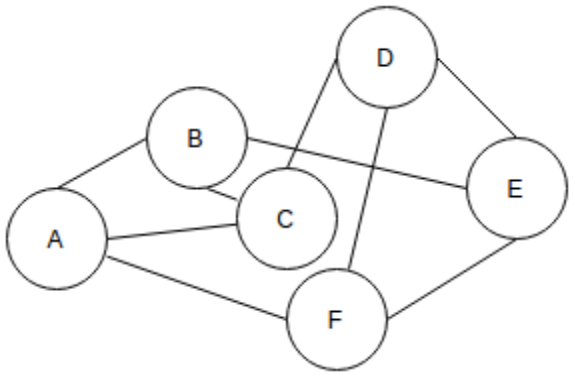
a) G is a complete graph
b) G is not a connected graph
c) The vertex connectivity of the graph is 2
d) The edge connectivity of the graph is 1

Answer: c
Explanation: After removing vertices B and C, the graph becomes disconnected.

4. What is the number of edges present in a complete graph having n vertices?
a) (n*(n+1))/2
b) (n*(n-1))/2
c) n
d) Information given is insufficient

Answer: b
Explanation: Number of ways in which every vertex can be connected to each other is nC2.

5. The given Graph is regular.



a) True
b) False

Answer: a
Explanation: In a regular graph, degrees of all the vertices are equal. In the given graph the degree of every vertex is 3.

6. In a simple graph, the number of edges is equal to twice the sum of the degrees of the vertices.
a) True
b) False

Answer: b
Explanation: The sum of the degrees of the vertices is equal to twice the number of edges.

7. A connected planar graph having 6 vertices, 7 edges contains _____ regions.
a) 15
b) 3
c) 1
d) 11

Answer: b
Explanation: By euler's formula the relation between vertices(n), edges(q) and regions(r) is given by n-q+r=2.

8. If a simple graph G, contains n vertices and m edges, the number of edges in the Graph G'(Complement of G) is _____
a) (n*n-n-2*m)/2
b) (n*n+n+2*m)/2
c) (n*n-n-2*m)/2
d) (n*n-n+2*m)/2

Answer: a
Explanation: The union of G and G' would be a complete graph so, the number of edges in G'= number of edges in the complete form of G(nC2)-edges in G(m).

9. Which of the following properties does a simple graph not hold?
a) Must be connected
b) Must be unweighted
c) Must have no loops or multiple edges
d) Must have no multiple edges

Answer: a
Explanation: A simple graph maybe connected or disconnected.

10. What is the maximum number of edges in a bipartite graph having 10 vertices?
a) 24
b) 21
c) 25
d) 16

Answer: c
Explanation: Let one set have n vertices another set would contain 10-n vertices.

Total number of edges would be n*(10-n), differentiating with respect to n, would yield the answer.

11. Which of the following is true?
a) A graph may contain no edges and many vertices
b) A graph may contain many edges and no vertices
c) A graph may contain no edges and no vertices
d) A graph may contain no vertices and many edges

Answer: b
Explanation: A graph must contain at least one vertex.

12. For a given graph G having v vertices and e edges which is connected and has no cycles, which of the following statements is true?
a) v=e
b) v = e+1
c) v + 1 = e
d) v = e-1

Answer: b
Explanation: For any connected graph with no cycles the equation holds true.

13. For which of the following combinations of the degrees of vertices would the connected graph be eulerian?
a) 1,2,3
b) 2,3,4
c) 2,4,5
d) 1,3,5

Answer: a
Explanation: A graph is eulerian if either all of its vertices are even or if only two of its vertices are odd.

14. A graph with all vertices having equal degree is known as a _____
a) Multi Graph
b) Regular Graph
c) Simple Graph
d) Complete Graph

Answer: b
Explanation: The given statement is the definition of regular graphs.

15. Which of the following ways can be used to represent a graph?
a) Adjacency List and Adjacency Matrix
b) Incidence Matrix
c) Adjacency List, Adjacency Matrix as well as Incidence Matrix
d) No way to represent

Answer: c
Explanation: Adjacency Matrix, Adjacency List and Incidence Matrix are used to represent a graph.

Recursion

1. Recursion is a method in which the solution of a problem depends on _____
a) Larger instances of different problems
b) Larger instances of the same problem
c) Smaller instances of the same problem
d) Smaller instances of different problems

Answer: c
Explanation: In recursion, the solution of a problem depends on the solution of smaller instances of the same problem.

2. Which of the following problems can't be solved using recursion?
a) Factorial of a number
b) Nth fibonacci number
c) Length of a string
d) Problems without base case

Answer: d
Explanation: Problems without base case leads to infinite recursion call. In general, we will assume a base case to avoid infinite recursion call. Problems like finding Factorial of a number, Nth Fibonacci number and Length of a string can be solved using recursion.

3. Recursion is similar to which of the following?
a) Switch Case
b) Loop
c) If-else
d) if elif else

Answer: b
Explanation: Recursion is similar to a loop.

4. In recursion, the condition for which the function will stop calling itself is _____
a) Best case
b) Worst case
c) Base case
d) There is no such condition

Answer: c
Explanation: For recursion to end at some point, there always has to be a condition for which the function will not call itself. This condition is known as base case.

5. What will happen when the below code snippet is executed?

```
void my_recursive_function()
```

```
{
   my_recursive_function();
}
int main()
{
   my_recursive_function();
   return 0;
}
```

a) The code will be executed successfully and no output will be generated
b) The code will be executed successfully and random output will be generated
c) The code will show a compile time error
d) The code will run for some time and stop when the stack overflows

Answer: d
Explanation: Every function call is stored in the stack memory. In this case, there is no terminating condition(base case). So, my_recursive_function() will be called continuously till the stack overflows and there is no more space to store the function calls. At this point of time, the program will stop abruptly.

6. What is the output of the following code?

```
void my_recursive_function(int n)
{
    if(n == 0)
    return;
    printf("%d ",n);
```

```
    my_recursive_function(n-1);

}

int main()

{

    my_recursive_function(10);

    return 0;

}
```

a) 10
b) 1
c) 10 9 8 … 1 0
d) 10 9 8 … 1

Answer: d
Explanation: The program prints the numbers from 10 to 1.

7. What is the base case for the following code?

```
void my_recursive_function(int n)

{

     if(n == 0)

     return;

     printf("%d ",n);

     my_recursive_function(n-1);

}
```

```
int main()

{

     my_recursive_function(10);

     return 0;

}
```

a) return
b) printf(“%d “, n)
c) if(n == 0)
d) my_recursive_function(n-1)

Answer: c
Explanation: For the base case, the recursive function is not called. So, “if(n == 0)” is the base case.

8. How many times is the recursive function called, when the following code is executed?

```
void my_recursive_function(int n)

{

     if(n == 0)

     return;

     printf("%d ",n);

     my_recursive_function(n-1);

}

int main()

{
```

```
    my_recursive_function(10);

    return 0;

}
```

a) 9
b) 10
c) 11
d) 12

Answer: c
Explanation: The recursive function is called 11 times.

9. What does the following recursive code do?

```
void my_recursive_function(int n)

{

    if(n == 0)

    return;

    my_recursive_function(n-1);

    printf("%d ",n);

}

int main()

{

    my_recursive_function(10);

    return 0;
```

```
}
```

a) Prints the numbers from 10 to 1
b) Prints the numbers from 10 to 0
c) Prints the numbers from 1 to 10
d) Prints the numbers from 0 to 10

Answer: c
Explanation: The above code prints the numbers from 1 to 10.

10. Which of the following statements is true?
a) Recursion is always better than iteration
b) Recursion uses more memory compared to iteration
c) Recursion uses less memory compared to iteration
d) Iteration is always better and simpler than recursion

Answer: b
Explanation: Recursion uses more memory compared to iteration because every time the recursive function is called, the function call is stored in stack.

11. What will be the output of the following code?

```
int cnt=0;

void my_recursive_function(int n)

{

     if(n == 0)

     return;

     cnt++;

     my_recursive_function(n/10);

}

int main()
```

```
{
    my_recursive_function(123456789);
    printf("%d",cnt);
    return 0;
}
```

a) 123456789
b) 10
c) 0
d) 9

Answer: d
Explanation: The program prints the number of digits in the number 123456789, which is 9.

12. What will be the output of the following code?

```
void my_recursive_function(int n)
{
    if(n == 0)
    {
        printf("False");
        return;
    }
    if(n == 1)
    {
```

```
            printf("True");
            return;
        }
        if(n%2==0)
        my_recursive_function(n/2);
        else
        {
            printf("False");
            return;
        }

}
int main()
{
     my_recursive_function(100);
     return 0;
}
```

a) True
b) False

Answer: b
Explanation: The function checks if a number is a power of 2. Since 100 is not a power of 2, it prints false.

13. What is the output of the following code?

```
int cnt = 0;

void my_recursive_function(char *s, int i)

{

     if(s[i] == '\0')

        return;

     if(s[i] == 'a' || s[i] == 'e' || s[i] == 'i' || s[i] == 'o' || s[i] == 'u')

     cnt++;

     my_recursive_function(s,i+1);

}

int main()

{

     my_recursive_function("thisisrecursion",0);

     printf("%d",cnt);

     return 0;
```

```
}
```

a) 6
b) 9
c) 5
d) 10

Answer: a
Explanation: The function counts the number of vowels in a string. In this case the number is vowels is 6.

14. What is the output of the following code?

```
void my_recursive_function(int *arr, int val, int idx, int len)

{

    if(idx == len)

    {

        printf("-1");

        return ;

    }

    if(arr[idx] == val)

    {

        printf("%d",idx);

        return;

    }
```

```
    my_recursive_function(arr,val,idx+1,len);

}

int main()

{

     int array[10] = {7, 6, 4, 3, 2, 1, 9, 5, 0, 8};

     int value = 2;

     int len = 10;

     my_recursive_function(array, value, 0, len);

     return 0;

}
```

a) 3
b) 4
c) 5
d) 6

Answer: b
Explanation: The program searches for a value in the given array and prints the index at which the value is found. In this case, the program searches for value = 2. Since, the index of 2 is 4(0 based indexing), the program prints 4.

Searching

1. Which of the following searching algorithm is fastest?
a) binary search
b) linear search
c) jump search
d) all are equally fast

Answer: a
Explanation: Binary search has the least time complexity (equal to log n) out of the given searching algorithms. This makes binary search preferable in most cases.

2. Where is linear searching used?
a) Used all the time
b) When the list has only a few elements
c) When performing a single search in an unordered list
d) When the list has only a few elements and When performing a single search in an unordered list

Answer: d
Explanation: It is practical to implement linear search in the situations mentioned in When the list has only a few elements and When performing a single search in an unordered list, but for larger elements the complexity becomes larger and it makes sense to sort the list and employ binary search or hashing.

3. How can Jump Search be improved?
a) Step size should be other than sqrt(n)
b) Cannot be improved
c) Begin from the kth item, where k is the step size
d) Start searching from the end

Answer: c
Explanation: This gives a very slight improvement as you are skipping the first k elements.

4. Which of the following searching algorithm is used with exponential sort after finding the appropriate range?
a) Jump search
b) Fibonacci Search
c) Linear search
d) Binary search

Answer: d
Explanation: In exponential search, we first find a range where the required elements should be present in the array. Then we apply binary search in this range.

5. Which of the following searching algorithm is fastest when the input array is not sorted but has uniformly distributed values?
a) linear search
b) jump search
c) interpolation search
d) binary search

Answer: a
Explanation: Out of the given options linear search is the only searching algorithm which can be applied to arrays which are not sorted. It has a time complexity of O(n) in the worst case.

6. What is the time complexity of Z algorithm for pattern searching (m = length of text, n = length of pattern)?
a) O(n)
b) O(m)
c) O(n + m)
d) O(m * n)

Answer: c
Explanation: Z algorithm is an efficient pattern searching algorithm as it searches the pattern in linear time. It has a time complexity of O(m + n) where m is the length of text and n is the length of the pattern.

7. In which of the cases uniform binary search fails compared to binary search?
a) Complexity of code
b) Many searches will be performed on several arrays of the same length
c) Many searches will be performed on the same array
d) A table lookup is generally faster than an addition and a shift

Answer: a
Explanation: Uniform binary search code is more complex to implement than binary search as it involves mid points to be computed in hand before search.

8. Interpolation search is a variation of?
a) Exponential search
b) Linear search
c) Binary search
d) Jump search

Answer: c
Explanation: Interpolation search is a variation of binary search which gives the best result when the array has uniformly distributed values. Interpolation search goes to different positions depending on the value being searched whereas binary search always goes to the middle element.

9. Which of the following is not an application of binary search?
a) To search in unordered list
b) Debugging
c) Union of intervals
d) To find the lower/upper bound in an ordered sequence

Answer: a
Explanation: In Binary search, the elements in the list should be sorted. It is applicable only for ordered list. Hence Binary search in unordered list is not an application.

10. Which of the following step is taken after finding an element having value greater than the element being searched?
a) binary search takes place in the forward direction
b) binary search takes place in a backward direction
c) linear search takes place in the forward direction
d) linear search takes place in the backward direction

Answer: d
Explanation: First an element having value greater than the element being searched is found. After this linear search is performed in a backward direction.

11. Which of the following is not an advantage of Fibonacci Search?
a) When the element being searched for has a non uniform access storage
b) It can be applied efficiently on unsorted arrays
c) Can be used for large arrays which do not fit in the CPU cache or in the RAM
d) Can be used in magnetic tapes

Answer: b
Explanation: When the speed of access depends on the location previously accessed, Fibonacci search is better compared to binary search as it performs well on those locations which have lower dispersion. Fibonacci search won't work on unsorted arrays. The input should be a sorted array or array should be sorted before Fibonacci search.

12. In which of the following case jump search will be preferred over exponential search?
a) when the given array is very small in size
b) when the given array is very large in size
c) jumping backwards takes significantly more time than jumping forward
d) jumping forward takes significantly more time than jumping backwards

Answer: d
Explanation: Jump search only needs to jump backwards once, while an exponential search can jump backwards up to log n times. Thus jump search will be preferred if jumping backwards is expensive.

1. Where is linear searching used?
a) When the list has only a few elements
b) When performing a single search in an unordered list
c) Used all the time
d) When the list has only a few elements and When performing a single search in an unordered list

Answer: d
Explanation: It is practical to implement linear search in the situations mentioned in When the list has only a few elements and When performing a single search in an unordered list, but for larger elements the complexity becomes larger and it makes sense to sort the list and employ binary search or hashing.

3. What is the best case for linear search?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(1)

Answer: d
Explanation: The element is at the head of the array, hence O(1).

4. What is the worst case for linear search?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(1)

Answer: c
Explanation: Worst case is when the desired element is at the tail of the array or not present at all, in this case you have to traverse till the end of the array, hence the complexity is O(n).

6. What is the best case and worst case complexity of ordered linear search?
a) O(nlogn), O(logn)
b) O(logn), O(nlogn)
c) O(n), O(1)
d) O(1), O(n)

Answer: d
Explanation: Although ordered linear search is better than unordered when the element is not present in the array, the best and worst cases still remain the same, with the key element being found at first position or at last position.

10. Which of the following is a disadvantage of linear search?
a) Requires more space
b) Greater time complexities compared to other searching algorithms
c) Not easy to understand
d) Not easy to implement

Answer: b
Explanation: The complexity of linear search as the name suggests is O(n) which is much greater than other searching techniques like binary search(O(logn)). Linear search is easy to implement and understand than other searching techniques.

1. What is the advantage of recursive approach than an iterative approach?
a) Consumes less memory
b) Less code and easy to implement
c) Consumes more memory
d) More code has to be written

Answer: b
Explanation: A recursive approach is easier to understand and contains fewer lines of code.

5. What is the worst case complexity of binary search using recursion?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n2)

Answer: b
Explanation: Using the divide and conquer master theorem.

6. What is the average case time complexity of binary search using recursion?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n2)

Answer: b

Explanation: T(n) = T(n/2) + 1, Using the divide and conquer master theorem.

7. Which of the following is not an application of binary search?
a) To find the lower/upper bound in an ordered sequence
b) Union of intervals
c) Debugging
d) To search in unordered list

Answer: d
Explanation: In Binary search, the elements in the list should be sorted. It is applicable only for ordered list. Hence Binary search in unordered list is not an application.

9. Binary Search can be categorized into which of the following?
a) Brute Force technique
b) Divide and conquer
c) Greedy algorithm
d) Dynamic programming

Answer: b
Explanation: Since ‘mid’ is calculated for every iteration or recursion, we are diving the array into half and then try to solve the problem.

12. What is the time complexity of binary search with iteration?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n2)

Answer: b

Sorting

1. Which of the following sorting algorithms is the fastest for sorting small arrays?
a) Quick sort
b) Shell sort
c) Insertion sort
d) Heap sort

Answer: c
Explanation: For sorting small arrays, insertion sort runs even faster than quick sort. But, it is impractical to sort large arrays.

2. What is the advantage of selection sort over other sorting techniques?
a) It is faster than any other sorting technique
b) It is scalable
c) It works best for inputs which are already sorted
d) It requires no additional storage space

Answer: d
Explanation: Since selection sort is an in-place sorting algorithm, it does not require additional storage.

3. Which of the following method is used for sorting in merge sort?
a) partitioning

b) merging
c) exchanging
d) selection

Answer: b
Explanation: Merge sort algorithm divides the array into two halves and applies merge sort algorithm to each half individually after which the two sorted halves are merged together. Thus its method of sorting is called merging.

4. Which of the following sorting algorithm does not use recursion?
a) bottom up merge sort
b) merge sort
c) heap sort
d) quick sort

Answer: a
Explanation: Bottom up merge sort uses the iterative method in order to implement sorting. It begins by merging a pair of adjacent array of size 1 each and then merge arrays of size 2 each in the next step and so on.

5. Merge sort uses which of the following method to implement sorting?
a) selection
b) exchanging
c) merging
d) partitioning

Answer: c
Explanation: Merge sort implements sorting by merging the sorted versions of smaller parts of the array. Thus its method of sorting is called merging.

6. Which of the following sorting algorithms is the fastest?
a) Merge sort
b) Shell sort
c) Insertion sort
d) Quick sort

Answer: d
Explanation: Quick sort is the fastest known sorting algorithm because of its highly optimized inner loop.

7. Shell sort algorithm is an example of?
a) Bottom-up sorting
b) In-place sorting

c) Internal sorting
d) External sorting

Answer: c
Explanation: Shell sort is an example of internal sorting because sorting of elements is done internally using an array.

8. Quick sort uses which of the following method to implement sorting?
a) partitioning
b) selection
c) exchanging
d) merging

Answer: a
Explanation: Quick sort makes partitions of the input array about the pivot in order to implement sorting. Thus its method of sorting is called partitioning.

9. In heap sort, after deleting the last minimum element, the array will contain elements in?
a) increasing sorting order
b) tree preorder
c) tree inorder
d) decreasing sorting order

Answer: d
Explanation: By logic, after deleting minimum element, the heap will contain elements in decreasing sorting order. We can change this by altering the ordering property.

10. Which of the following sorting algorithm is used by C++ internally?
a) quicksort
b) merge sort
c) introsort
d) heap sort

Answer: c
Explanation: Introsort is the in built sorting algorithm used by C++. It is an example of a hybrid sorting algorithm which means it uses more than one sorting algorithm as a routine.

11. Which of the following sorting algorithm is stable?
a) Introsort
b) Tim sort
c) Heap sort
d) Quick sort

Answer: b
Explanation: Out of the given options Tim sort is the only algorithm which is stable. As both constituents of Tim sort (I.e insertion sort and merge sort) are stable so Tim sort also becomes stable.

12. Which of the following sorting algorithm uses the method of insertion?
a) selection sort
b) quick sort
c) bubble sort
d) cycle sort

Answer: d
Explanation: Cycle sort is the only algorithm from the given ones that uses the method of insertion. Other than this, insertion sort also uses the method of insertion for sorting.

13. Which of the following pair of sorting algorithms are stable?
a) gnome sort and merge sort
b) heap sort and merge sort
c) gnome sort and quick sort
d) merge sort and selection sort

Answer: a
Explanation: Gnome sort and merge sort are stable sorting algorithms as the elements with identical values appear in the same order in the output array as they were in the input array when any of these sorting algorithms are implemented.

1. How many passes does an insertion sort algorithm consist of?
a) N
b) N-1
c) N+1
d) N2

Answer: b
Explanation: An insertion algorithm consists of N-1 passes when an array of N elements is given.

2. Which of the following algorithm implementations is similar to that of an insertion sort?
a) Binary heap
b) Quick sort
c) Merge sort
d) Radix sort

Answer: a
Explanation: Insertion sort is similar to that of a binary heap algorithm because of the use of temporary variable to swap.

3. What is the average case running time of an insertion sort algorithm?
a) O(N)
b) O(N log N)
c) O(log N)
d) O(N2)

Answer: d
Explanation: The average case analysis of a tight bound algorithm is mathematically achieved to be O(N2).

4. Any algorithm that sorts by exchanging adjacent elements require O(N2) on average.
a) True
b) False

Answer: a
Explanation: Each swap removes only one inversion, so O(N2) swaps are required.

5. What is the average number of inversions in an array of N distinct numbers?
a) N(N-1)/4
b) N(N+1)/2
c) N(N-1)/2
d) N(N-1)/3

Answer: a
Explanation: The total number of pairs in a list L is N(N-1)/2. Thus, an average list has half this amount, or N(N-1)/4 inversions.

6. What is the running time of an insertion sort algorithm if the input is pre-sorted?
a) O(N2)
b) O(N log N)
c) O(N)
d) O(M log N)

Answer: c
Explanation: If the input is pre-sorted, the running time is O(N), because the test in the inner for loop always fails immediately and the algorithm will run quickly.

7. What will be the number of passes to sort the elements using insertion sort?
14, 12,16, 6, 3, 10

a) 6
b) 5
c) 7
d) 1

Answer: b
Explanation: The number of passes is given by N-1. Here, N=6. Therefore,

6-1=5 passes.

8. For the following question, how will the array elements look like after second pass?
34, 8, 64, 51, 32, 21
a) 8, 21, 32, 34, 51, 64
b) 8, 32, 34, 51, 64, 21
c) 8, 34, 51, 64, 32, 21
d) 8, 34, 64, 51, 32, 21

Answer: d
Explanation: After swapping elements in the second pass, the array will look like, 8, 34, 64, 51, 32, 21.

9. Which of the following real time examples is based on insertion sort?
a) arranging a pack of playing cards
b) database scenarios and distributes scenarios
c) arranging books on a library shelf
d) real-time systems

Answer: a
Explanation: Arranging a pack of cards mimics an insertion sort. Database scenario is an example for merge sort, arranging books is a stack and real-time systems uses quick sort.

10. In C, what are the basic loops required to perform an insertion sort?
a) do- while
b) if else
c) for and while
d) for and if

Answer: c

Explanation: To perform an insertion sort, we use two basic loops- an outer for loop and an inner while loop.

11. Binary search can be used in an insertion sort algorithm to reduce the number of comparisons.
a) True
b) False

Answer: a

12. Which of the following options contain the correct feature of an insertion sort algorithm?
a) anti-adaptive
b) dependable
c) stable, not in-place
d) stable, adaptive

Answer: d
Explanation: An insertion sort is stable, adaptive, in-place and incremental in nature.

13. Which of the following sorting algorithms is the fastest for sorting small arrays?
a) Quick sort
b) Insertion sort
c) Shell sort
d) Heap sort

Answer: b
Explanation: For sorting small arrays, insertion sort runs even faster than quick sort. But, it is impractical to sort large arrays.

14. For the best case input, the running time of an insertion sort algorithm is?
a) Linear
b) Binary
c) Quadratic
d) Depends on the input

Answer: a
Explanation: The best case input for an insertion sort algorithm runs in linear time and is given by O(N).

15. Which of the following examples represent the worst case input for an insertion sort?
a) array in sorted order
b) array sorted in reverse order
c) normal unsorted array
d) large array

Answer: b
Explanation: The worst case input for an insertion sort algorithm will be an array sorted in reverse order and its running time is quadratic.

1. Which of the following is correct with regard to insertion sort?
a) insertion sort is stable and it sorts In-place
b) insertion sort is unstable and it sorts In-place
c) insertion sort is stable and it does not sort In-place
d) insertion sort is unstable and it does not sort In-place

Answer: a
Explanation: During insertion sort, the relative order of elements is not changed. Therefore, it is a stable sorting algorithm. And insertion sort requires only O(1) of additional memory space. Therefore, it sorts In-place.

2. Which of the following sorting algorithm is best suited if the elements are already sorted?
a) Heap Sort
b) Quick Sort
c) Insertion Sort
d) Merge Sort

Answer: c
Explanation: The best case running time of the insertion sort is O(n). The best case occurs when the input array is already sorted. As the elements are already sorted, only one comparison is made on each pass, so that the time required is O(n).

3. The worst case time complexity of insertion sort is O(n2). What will be the worst case time complexity of insertion sort if the correct position for inserting element is calculated using binary search?
a) O(nlogn)
b) O(n2)
c) O(n)
d) O(logn)

Answer: b
Explanation: The use of binary search reduces the time of finding the correct position from O(n) to O(logn). But the worst case of insertion sort remains O(n2) because of the series of swapping operations required for each insertion.

4. Insertion sort is an example of an incremental algorithm.
a) True
b) False

Answer: a
Explanation: In the incremental algorithms, the complicated structure on n items is built by first building it on n − 1 items. And then we make the necessary changes to fix things in adding the last item. Insertion sort builds the sorted sequence one element at a time. Therefore, it is an example of an incremental algorithm.

6. Which of the following is good for sorting arrays having less than 100 elements?
a) Quick Sort
b) Selection Sort
c) Merge Sort
d) Insertion Sort

Answer: d
Explanation: The insertion sort is good for sorting small arrays. It sorts smaller arrays faster than any other sorting algorithm.

9. In insertion sort, the average number of comparisons required to place the 7th element into its correct position is ____
a) 9
b) 4
c) 7
d) 14

Answer: b
Explanation: On average (k + 1) / 2 comparisons are required to place the kth element into its correct position. Therefore, average number of comparisons required for 7th element = (7 + 1)/2 = 4.

10. Which of the following is not an exchange sort?
a) Bubble Sort
b) Quick Sort
c) Partition-exchange Sort
d) Insertion Sort

Answer: d
Explanation: In Exchange sorts, we compare each element of an array and swap those elements that are not in their proper position. Bubble Sort and Quick Sort are exchange sorts. Quick Sort is also called as Partition-exchange Sort. Insertion sort is not an exchange sort

1. What is an in-place sorting algorithm?
a) It needs O(1) or O(logn) memory to create auxiliary locations
b) The input is already sorted and in-place
c) It requires additional storage
d) It requires additional space

Answer: a
Explanation: Auxiliary memory is required for storing the data temporarily.

2. In the following scenarios, when will you use selection sort?
a) The input is already sorted
b) A large file has to be sorted
c) Large values need to be sorted with small keys
d) Small values need to be sorted with large keys

Answer: c
Explanation: Selection is based on keys, hence a file with large values and small keys can be efficiently sorted with selection sort.

3. What is the worst case complexity of selection sort?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n2)

Answer: d
Explanation: Selection sort creates a sub-list, LHS of the ‘min’ element is already sorted and RHS is yet to be sorted. Starting with the first element the ‘min’ element moves towards the final element.

5. What is the advantage of selection sort over other sorting techniques?
a) It requires no additional storage space
b) It is scalable
c) It works best for inputs which are already sorted
d) It is faster than any other sorting technique

Answer: a
Explanation: Since selection sort is an in-place sorting algorithm, it does not require additional storage.

6. What is the average case complexity of selection sort?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n2)

Answer: d
Explanation: In the average case, even if the input is partially sorted, selection sort behaves as if the entire array is not sorted. Selection sort is insensitive to input.

7. What is the disadvantage of selection sort?
a) It requires auxiliary memory
b) It is not scalable
c) It can be used for small keys
d) It takes linear time to sort the elements

Answer: b
Explanation: As the input size increases, the performance of selection sort decreases.

8. The given array is arr = {3,4,5,2,1}. The number of iterations in bubble sort and selection sort respectively are _____
a) 5 and 4
b) 4 and 5
c) 2 and 4
d) 2 and 5

Answer: a
Explanation: Since the input array is not sorted, bubble sort takes 5 iterations and selection sort takes 4(n-1) iterations.

9. The given array is arr = {1,2,3,4,5}. (bubble sort is implemented with a flag variable)The number of iterations in selection sort and bubble sort respectively are _____
a) 5 and 4
b) 1 and 4
c) 0 and 4
d) 4 and 1

Answer: d
Explanation: Selection sort is insensitive to input, hence 4(n-1) iterations. Whereas bubble sort iterates only once to set the flag to 0 as the input is already sorted.

10. What is the best case complexity of selection sort?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n2)

Answer: d

Explanation: The best, average and worst case complexities of selection sort is O(n2).

(n-1) + (n-2) + (n-3) + …. + 1 = (n(n-1))/2 ~ (n2)/2.

1. What is an external sorting algorithm?
a) Algorithm that uses tape or disk during the sort
b) Algorithm that uses main memory during the sort
c) Algorithm that involves swapping
d) Algorithm that are considered ‘in place’

Answer: a
Explanation: As the name suggests, external sorting algorithm uses external memory like tape or disk.

2. What is an internal sorting algorithm?
a) Algorithm that uses tape or disk during the sort
b) Algorithm that uses main memory during the sort
c) Algorithm that involves swapping
d) Algorithm that are considered ‘in place’

Answer: b
Explanation: As the name suggests, internal sorting algorithm uses internal main memory.

3. What is the worst case complexity of bubble sort?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n2)

Answer: d
Explanation: Bubble sort works by starting from the first element and swapping the elements if required in each iteration.

5. What is the average case complexity of bubble sort?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n2)

Answer: d
Explanation: Bubble sort works by starting from the first element and swapping the elements if required in each iteration even in the average case.

6. Which of the following is not an advantage of optimised bubble sort over other sorting techniques in case of sorted elements?
a) It is faster
b) Consumes less memory

c) Detects whether the input is already sorted
d) Consumes less time

Answer: c
Explanation: Optimised Bubble sort is one of the simplest sorting techniques and perhaps the only advantage it has over other techniques is that it can detect whether the input is already sorted. It is faster than other in case of sorted array and consumes less time to describe whether the input array is sorted or not. It consumes same memory than other sorting techniques. Hence it is not an advantage.

7. The given array is arr = {1, 2, 4, 3}. Bubble sort is used to sort the array elements. How many iterations will be done to sort the array?
a) 4
b) 2
c) 1
d) 0

Answer: a
Explanation: Even though the first two elements are already sorted, bubble sort needs 4 iterations to sort the given array.

9. What is the best case efficiency of bubble sort in the improvised version?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n2)

Answer: c
Explanation: Some iterations can be skipped if the list is sorted, hence efficiency improves to O(n).

10. The given array is arr = {1,2,4,3}. Bubble sort is used to sort the array elements. How many iterations will be done to sort the array with improvised version?
a) 4
b) 2
c) 1
d) 0

Answer: b
Explanation: Only 2 elements in the given array are not sorted, hence only 2 iterations are required to sort them.

1. Merge sort uses which of the following technique to implement sorting?
a) backtracking

b) greedy algorithm
c) divide and conquer
d) dynamic programming

Answer: c
Explanation: Merge sort uses divide and conquer in order to sort a given array. This is because it divides the array into two halves and applies merge sort algorithm to each half individually after which the two sorted halves are merged together.

2. What is the average case time complexity of merge sort?
a) O(n log n)
b) O(n2)
c) O(n2 log n)
d) O(n log n2)

Answer: a
Explanation: The recurrence relation for merge sort is given by T(n) = 2T(n/2) + n. It is found to be equal to O(n log n) using the master theorem.

3. What is the auxiliary space complexity of merge sort?
a) O(1)
b) O(log n)
c) O(n)
d) O(n log n)

Answer: c
Explanation: An additional space of O(n) is required in order to merge two sorted arrays. Thus merge sort is not an in place sorting algorithm.

4. Merge sort can be implemented using O(1) auxiliary space.
a) true
b) false

Answer: a
Explanation: Standard merge sort requires O(n) space to merge two sorted arrays. We can optimize this merging process so that it takes only constant space. This version is known as in place merge sort.

5. What is the worst case time complexity of merge sort?
a) O(n log n)
b) O(n2)
c) O(n2 log n)
d) O(n log n2)

Answer: a
Explanation: The time complexity of merge sort is not affected by worst case as its algorithm has to implement the same number of steps in any case. So its time complexity remains to be O(n log n).

6. Which of the following method is used for sorting in merge sort?
a) merging
b) partitioning
c) selection
d) exchanging

Answer: a
Explanation: Merge sort algorithm divides the array into two halves and applies merge sort algorithm to each half individually after which the two sorted halves are merged together. Thus its method of sorting is called merging.

7. What will be the best case time complexity of merge sort?
a) O(n log n)
b) O(n2)
c) O(n2 log n)
d) O(n log n2)

Answer: a
Explanation: The time complexity of merge sort is not affected in any case as its algorithm has to implement the same number of steps. So its time complexity remains to be O(n log n) even in the best case.

8. Which of the following is not a variant of merge sort?
a) in-place merge sort
b) bottom up merge sort
c) top down merge sort
d) linear merge sort

Answer: d
Explanation: In-place, top down and bottom up merge sort are different variants of merge sort. Whereas linear merge sort is not a possible variant as it is a comparison based sort and the minimum time complexity of any comparison based sort is O(n log n).

9. Choose the incorrect statement about merge sort from the following?
a) it is a comparison based sort
b) it is an adaptive algorithm
c) it is not an in place algorithm
d) it is stable algorithm

Answer: b
Explanation: Merge sort is not an adaptive sorting algorithm. This is because it takes O(n log n) time complexity irrespective of any case.

10. Which of the following is not in place sorting algorithm by default?
a) merge sort
b) quick sort
c) heap sort
d) insertion sort

Answer: a
Explanation: Quick sort, heap sort, and insertion sort are in-place sorting algorithms, whereas an additional space of O(n) is required in order to merge two sorted arrays. Even though we have a variation of merge sort (to do in-place sorting), it is not the default option. So, among the given choices, merge sort is the most appropriate answer.

11. Which of the following is not a stable sorting algorithm?
a) Quick sort
b) Cocktail sort
c) Bubble sort
d) Merge sort

Answer: a
Explanation: Out of the given options quick sort is the only algorithm which is not stable. Merge sort is a stable sorting algorithm.

12. Which of the following stable sorting algorithm takes the least time when applied to an almost sorted array?
a) Quick sort
b) Insertion sort
c) Selection sort
d) Merge sort

Answer: d
Explanation: Insertion sort takes linear time to sort a partially sorted array. Though merge and quick sort takes O(n*logn) complexity to sort, merge sort is stable. Hence, Merge sort takes less time to sort partially sorted array.

13. Merge sort is preferred for arrays over linked lists.
a) true
b) false

Answer: b
Explanation: Merge sort is preferred for linked list over arrays. It is because in a linked list the insert operation takes only O(1) time and space which implies that we can implement merge operation in constant time.

14. Which of the following sorting algorithm makes use of merge sort?
a) tim sort
b) intro sort
c) bogo sort
d) quick sort

Answer: a
Explanation: Tim sort is a hybrid sorting algorithm as it uses more than one sorting algorithm internally. It makes use of merge sort and insertion sort.

16. Which of the following sorting algorithm does not use recursion?
a) quick sort
b) merge sort
c) heap sort
d) bottom up merge sort

Answer: d
Explanation: Bottom up merge sort uses the iterative method in order to implement sorting. It begins by merging a pair of adjacent array of size 1 each and then merge arrays of size 2 each in the next step and so on.

1. Which of the following sorting algorithms is the fastest?
a) Merge sort
b) Quick sort
c) Insertion sort
d) Shell sort

Answer: b
Explanation: Quick sort is the fastest known sorting algorithm because of its highly optimized inner loop.

2. Quick sort follows Divide-and-Conquer strategy.
a) True
b) False

Answer: a
Explanation: In quick sort, the array is divided into sub-arrays and then it is sorted (divide-and-conquer strategy).

3. What is the worst case time complexity of a quick sort algorithm?
a) O(N)
b) O(N log N)
c) O(N2)
d) O(log N)

Answer: c
Explanation: The worst case performance of a quick sort algorithm is mathematically found to be O(N2).

4. Which of the following methods is the most effective for picking the pivot element?
a) first element
b) last element
c) median-of-three partitioning
d) random element

Answer: c
Explanation: Median-of-three partitioning is the best method for choosing an appropriate pivot element. Picking a first, last or random element as a pivot is not much effective.

5. Find the pivot element from the given input using median-of-three partitioning method.
8, 1, 4, 9, 6, 3, 5, 2, 7, 0.
a) 8
b) 7
c) 9
d) 6

Answer: d

Explanation: Left element=8, right element=0,

Centre=[position(left+right)/2]=6.

6. Which is the safest method to choose a pivot element?
a) choosing a random element as pivot
b) choosing the first element as pivot
c) choosing the last element as pivot
d) median-of-three partitioning method

Answer: a
Explanation: This is the safest method to choose the pivot element since it is very unlikely that a random pivot would consistently provide a poor partition.

7. What is the average running time of a quick sort algorithm?
a) O(N2)
b) O(N)
c) O(N log N)
d) O(log N)

Answer: c
Explanation: The best case and average case analysis of a quick sort algorithm are mathematically found to be O(N log N).

8. Which of the following sorting algorithms is used along with quick sort to sort the sub arrays?
a) Merge sort
b) Shell sort
c) Insertion sort
d) Bubble sort

Answer: c

Explanation: Insertion sort is used along with quick sort to sort the sub arrays. It is used only at the end.

9. Quick sort uses join operation rather than merge operation.
a) true
b) false

Answer: a
Explanation: Quick sort uses join operation since join is a faster operation than merge.

10. How many sub arrays does the quick sort algorithm divide the entire array into?
a) one
b) two
c) three
d) four

Answer: b
Explanation: The entire array is divided into two partitions, 1st sub array containing elements less than the pivot element and 2nd sub array containing elements greater than the pivot element.

11. Which is the worst method of choosing a pivot element?
a) first element as pivot
b) last element as pivot

c) median-of-three partitioning
d) random element as pivot

Answer: a
Explanation: Choosing the first element as pivot is the worst method because if the input is pre-sorted or in reverse order, then the pivot provides a poor partition.

12. Which among the following is the best cut-off range to perform insertion sort within a quick sort?
a) N=0-5
b) N=5-20
c) N=20-30
d) N>30

Answer: b
Explanation: A good cut-off range is anywhere between N=5 and N=20 to avoid nasty degenerate cases.

10. Quick sort is a space-optimised version of ____
a) Bubble sort
b) Selection sort
c) Insertion sort
d) Binary tree sort

Answer: d
Explanation: Quick sort is a space-optimised version of the binary tree sort. In binary sort tree, the elements are inserted sequentially into the binary search tree and Quick sort organises elements into a tree that is implied by the recursive calls.

9. Which one of the following sorting algorithm is best suited to sort an array of 1 million elements?
a) Bubble sort
b) Insertion sort
c) Merge sort
d) Quick sort

Answer: d
Explanation: The Quick sort is best suited to sort the array of 1 million elements. The practical implementations of Quick sort use randomised version. In practice randomised Quick sort algorithms rarely shows worst case behaviour and is almost always O(nlogn). And Quick sort requires little additional space and exhibits good cache locality.

5. Quick sort is a stable sorting algorithm.
a) True
b) False

Answer: b

4. What is a randomized QuickSort?
a) The leftmost element is chosen as the pivot
b) The rightmost element is chosen as the pivot
c) Any element in the array is chosen as the pivot
d) A random number is generated which is used as the pivot

Answer: c
Explanation: QuickSort is randomized by placing the input data in the randomized fashion in the array or by choosing a random element in the array as a pivot.

7. The given array is arr = {2,3,4,1,6}. What are the pivots that are returned as a result of subsequent partitioning?
a) 1 and 3
b) 3 and 1
c) 2 and 6
d) 6 and 2

Answer: a
Explanation: The call to partition returns 1 and 3 as the pivot elements.

10. Which of the following is not true about QuickSort?
a) in-place algorithm
b) pivot position can be changed
c) adaptive sorting algorithm
d) can be implemented as a stable sort

Answer: b
Explanation: Once a pivot is chosen, its position is finalized in the sorted array, it cannot be modified

8. The complexity of Fibonacci series is _____
a) O(2n)
b) O(log n)
c) O(n2)
d) O(n log n)

Answer: a

2. If for an algorithm time complexity is given by O(1) then the complexity of it is:
a) constant
b) polynomial

c) exponential
d) none of the mentioned

Answer: a

5. If for an algorithm time complexity is given by O(n2) then complexity will _____
a) constant
b) quadratic
c) exponential
d) none of the mentioned

Answer: b
Explanation: The growth rate of that function will be quadratic therefore complexity will be quadratic.

Asymptotic notations are used in computer science and mathematics to describe the rate of growth of functions. There are three commonly used asymptotic notations:

1. **Big O notation -** This notation describes an upper bound on the rate of growth of a function. It is used to describe the worst-case scenario for an algorithm. For example, if a function is **O(n)**, it means that the rate of growth of the function is no more than linear with respect to the input size.
2. **Omega notation -** This notation describes a lower bound on the rate of growth of a function. It is used to describe the best-case scenario for an algorithm. For example, if a function is **Ω(n)**, it means that the rate of growth of the function is at least linear with respect to the input size.
3. **Theta notation -** This notation describes an upper and lower bound on the rate of growth of a function. It is used to describe the average-case scenario for an algorithm. For example, if a function is **Θ(n)**, it means that the rate of growth of the function is linear with respect to the input size.

1. Which of the following algorithms are used to find the shortest path from a source node to all other nodes in a weighted graph?

BFS

Djikstra's Algorithm

Prims Algorithm

Kruskal's Algorithm

Answer - B)

9. What should be considered when designing an algorithm?

If this software is used correctly

In the hardware is used correctly

If there is more than one way to solve the problem

All of the above are correct

Answer - C)

10. What will be the best sorting algorithm, given that the array elements are small (<= 1e6)?

Bubble Sort

Merge Sort

Counting Sort

Heap Sort

Answer - C)

7. Which of the following algorithms are used for string and pattern matching problems??

Z Algorithm

Rabin Karp Algorithm

KMP Algorithm

All of the above

Answer - D)

19. Which of the following is known to be not an NP-Hard Problem?

Vertex Cover Problem

0/1 Knapsack Problem

Maximal Independent Set Problem

Travelling Salesman Problem

Answer - B)

20. Which of the following is used for solving the N Queens Problem?

Greedy algorithm

Dynamic programming

Backtracking

Sorting

Answer - C)

22. Which of the following statements is true about AVL Trees?

The difference between the heights of left and right nodes cannot be more than 1.

The height of an AVL Tree always remains of the order of O(logn)

AVL Trees are a type of self-balancing Binary Search Trees.

All of the above.

Answer - D)

26. Another name of the fractional knapsack is?

Non-continuous knapsack problem

Divisible knapsack problem

0/1 knapsack problem

Continuous Knapsack Problem

Answer - D)

27. Dijkstra's algorithm is used to solve _____ problems?

Network lock

Single source shortest path

All pair shortest path

Sorting

Answer - B)

28. Hamiltonian path problem is _____?

NP problem

P class problem

NP-complete problem

N class problem

Answer - C)

29. Heap is a _____?

Tree structure

Complete binary tree

Binary tree

None of the above

Answer - B)

30. Identify the approach followed in Floyd Warshall's algorithm?

Linear programming

Dynamic Programming

Greedy Technique

Backtracking

Answer - B)

32. Identify the function of the stack that returns the top data element of the stack?

pop()

peek()

push()

findTop()

Answer - B)

33. Identify the slowest sorting technique among the following?

Merge Sort

Quick Sort

Bubble Sort

Selection Sort

Answer - C)

34. Identify the sorting technique which compares adjacent elements in a list and switches whenever necessary?

Merge Sort
Quick Sort
Bubble Sort
Selection Sort
Answer - C)

35. In a graph of n nodes and n edges, how many cycles will be present?

Exactly 1
At most 1
At most 2
Depending on the graph
Answer - A) A tree contains by definition n nodes and n - 1 edge, and it is an acyclic graph. When we add 1 edge to the tree, we can form exactly one cycle by adding this edge.

6. Among the following options which is the best sorting algorithm when the list is already sorted?

Merge Sort
Insertion Sort
Bubble Sort
Selection Sort
Answer - B) Insertion Sort has a time complexity of O(N) when the array is always sorted.

37. Kruskal's Algorithm for finding the Minimum Spanning Tree of a graph is a kind of a?
DP Problem
Greedy Algorithm
Adhoc Problem
None of the above
Answer - B)

38. Representation of data structure in memory is known as?

Storage structure

File structure

Recursive

Abstract Data Type

Answer - D)

39. Select the correct recurrence relation for Tower of Hanoi?

T(N) = 2T(N-1)+1

T(N) = 2T(N/2)+1

T(N) = 2T(N-1)+N

T(N) = 2T(N-2)+2

Answer - A)

40. The Bellmann Ford Algorithm returns _____ value?

String

Boolean

Double

Integer

Answer - B)

42. The time complexity to find the longest common subsequence of two strings of length M and N is?

O(N)

O(M * N)

O(M)

O(log N)

Answer - B)

45. To main measures of the efficiency of an algorithm are?

Time and space complexity

Data and space

Processor and memory

Complexity and capacity

Answer - A)

11. What is the time complexity of Floyd–Warshall algorithm to calculate all pair shortest path in a graph with n vertices?

A) O(n2log(n))
B) Theta(n2log(n))
C) Theta(n4)
D) Theta(n3)

Answer: D

9. Which of the following is not a backtracking algorithm?

A. Knight tour problem
B. N queen problem
C. Tower of hanoi
D. M coloring problem

Answer: C
Knight tour problem, N Queen problem and M coloring problem involve backtracking. Tower of hanoi uses simple recursion.

12. Consider the following table

Algorithms	Design Paradigms
(P) Dijkastra's Algorithm	(i) Divide and Conquer
(Q) Strassen's Matrix Multiplication	(ii) Greedy
(R) Fibonacci numbers	(iii) Dynamic Programming

Match the algorithm to design paradigms they are based on:
A. P-(ii), Q-(iii), R-(i)
B. P-(iii), Q-(i), R-(ii)
C. P-(ii), Q-(i), R-(iii)
D. P-(i), Q-(ii), R-(iii)

Dijkstra's algorithm is Greedy technique to find the shortest path from a single source vertex to all other vertices in the given graph. Strassen's Matrix Multiplication is Divide and conquer technique to multiply matrices in efficient way. Fibonacci numbers uses Dynamic programming. Therefore, option (C) is true.

13. A problem in NP is NP-complete if

A. It can be reduced to the 3-SAT problem in polynomial time

B. The 3-SAT problem can be reduced to it in polynomial time

C. It can be reduced to any other problem in NP in polynomial time

D. Some problem in NP can be reduced to it in polynomial time

Answer: B

14. The time taken by binary search algorithm to search a key in a sorted array of n elements is

A .O (log2n)

B. O (n)

C. O (n log2n)

D. O (n^2)

Answer: A

15. Match the following:

List - I	List - II
(a) Sequential Search	(i) Dynamic programming principle
(b)Branch - and - bound	(ii) repeatedly double index
(c) Exponential Search	(iii) O(LogN)
(d) Binary Search	(iv)O(N)

codes:

	(a)	(b)	(c)	(d)
(1)	(i)	(iv)	(iii)	(ii)
(2)	(iv)	(i)	(ii)	(iii)
(3)	(i)	(iv)	(ii)	(iii)
(4)	(iv)	(ii)	(i)	(iii)

A (1)

B (2)

C (3)

D (4)

Answer: 2

9. In BFS, how many times a node is visited?
a) Once
b) Twice
c) Equivalent to number of indegree of the node
d) Thrice

Answer: c

3. The Data structure used in standard implementation of Breadth First Search is?
a) Stack
b) Queue
c) Linked List
d) Tree

Answer: b

1. Which of the following is false in the case of a spanning tree of a graph G?
a) It is tree that spans G
b) It is a subgraph of the G
c) It includes every vertex of the G
d) It can be either cyclic or acyclic

Answer: d
Explanation: A graph can have many spanning trees. Each spanning tree of a graph G is a subgraph of the graph G, and spanning trees include every vertex of the gram. Spanning trees are always acyclic.

2. Every graph has only one minimum spanning tree.
a) True
b) False

Answer: b

Explanation: Minimum spanning tree is a spanning tree with the lowest cost among all the spacing trees. Sum of all of the edges in the spanning tree is the cost of the spanning tree. There can be many minimum spanning trees for a given graph.

3. Consider a complete graph G with 4 vertices. The graph G has ____ spanning trees.

a) 15

b) 8

c) 16

d) 13

Answer: c

Explanation: A graph can have many spanning trees. And a complete graph with n vertices has n(n-2) spanning trees. So, the complete graph with 4 vertices has 4(4-2) = 16 spanning trees.

4. The travelling salesman problem can be solved using _____

a) A spanning tree

b) A minimum spanning tree

c) Bellman – Ford algorithm

d) DFS traversal

Answer: b

Explanation: In the travelling salesman problem we have to find the shortest possible route that visits every city exactly once and returns to the starting point for the given a set of cities. So, travelling salesman problem can be solved by contracting the minimum spanning tree.

9. Which of the following is not the algorithm to find the minimum spanning tree of the given graph?

a) Boruvka's algorithm

b) Prim's algorithm

c) Kruskal's algorithm

d) Bellman–Ford algorithm

Answer: d

Explanation: The Boruvka's algorithm, Prim's algorithm and Kruskal's algorithm are the algorithms

that can be used to find the minimum spanning tree of the given graph.The Bellman-Ford algorithm is used to find the shortest path from the single source to all other vertices.

1. Kruskal's algorithm is used to _____
a) find minimum spanning tree
b) find single source shortest path
c) find all pair shortest path algorithm
d) traverse the graph

Answer: a
Explanation: The Kruskal's algorithm is used to find the minimum spanning tree of the connected graph. It construct the MST by finding the edge having the least possible weight that connects two trees in the forest.

2. Kruskal's algorithm is a _____
a) divide and conquer algorithm
b) dynamic programming algorithm
c) greedy algorithm
d) approximation algorithm

Answer: c
Explanation: Kruskal's algorithm uses a greedy algorithm approach to find the MST of the connected weighted graph. In the greedy method, we attempt to find an optimal solution in stages.

4. What is the time complexity of Kruskal's algorithm?
a) O(log V)
b) O(E log V)
c) O(E2)
d) O(V log E)

Answer: b

7. Which of the following is true?
a) Prim's algorithm can also be used for disconnected graphs
b) Kruskal's algorithm can also run on the disconnected graphs

c) Prim’s algorithm is simpler than Kruskal’s algorithm
d) In Kruskal’s sort edges are added to MST in decreasing order of their weights

Answer: b
Explanation: Prim’s algorithm iterates from one node to another, so it can not be applied for disconnected graph. Kruskal’s algorithm can be applied to the disconnected graphs to construct the minimum cost forest. Kruskal’s algorithm is comparatively easier and simpler than prim’s algorithm.

8. Which of the following is false about the Kruskal’s algorithm?
a) It is a greedy algorithm
b) It constructs MST by selecting edges in increasing order of their weights
c) It can accept cycles in the MST
d) It uses union-find data structure

Answer: c
Explanation: Kruskal’s algorithm is a greedy algorithm to construct the MST of the given graph. It constructs the MST by selecting edges in increasing order of their weights and rejects an edge if it may form the cycle. So, using Kruskal’s algorithm is never formed.

9. Kruskal’s algorithm is best suited for the dense graphs than the prim’s algorithm.
a) True
b) False
Answer: b
Explanation: Prim’s algorithm outperforms the Kruskal’s algorithm in case of the dense graphs. It is significantly faster if graph has more edges than the Kruskal’s algorithm.

10. Consider the following statements.
S1. Kruskal’s algorithm might produce a non-minimal spanning tree.
S2. Kruskal’s algorithm can efficiently implemented using the disjoint-set data structure.
a) S1 is true but S2 is false
b) Both S1 and S2 are false
c) Both S1 and S2 are true
d) S2 is true but S1 is false

Answer: d

Explanation: In Kruskal's algorithm, the disjoint-set data structure efficiently identifies the components containing a vertex and adds the new edges. And Kruskal's algorithm always finds the MST for the connected graph.

13. Prim's algorithm is also known as _____

a) Dijkstra–Scholten algorithm

b) Borůvka's algorithm

c) Floyd–Warshall algorithm

d) DJP Algorithm

Answer: d

Explanation: The Prim's algorithm was developed by Vojtěch Jarník and it was latter discovered by the duo Prim and Dijkstra. Therefore, Prim's algorithm is also known as DJP Algorithm.

9. Prim's algorithm can be efficiently implemented using _____ for graphs with greater density.

a) d-ary heap

b) linear search

c) fibonacci heap

d) binary search

Answer: a

Explanation: In Prim's algorithm, we add the minimum weight edge for the chosen vertex which requires searching on the array of weights. This searching can be efficiently implemented using binary heap for dense graphs. And for graphs with greater density, Prim's algorithm can be made to run in linear time using d-ary heap(generalization of binary heap).

10. Which of the following is false about Prim's algorithm?

a) It is a greedy algorithm

b) It constructs MST by selecting edges in increasing order of their weights

c) It never accepts cycles in the MST

d) It can be implemented using the Fibonacci heap

Answer: b

Explanation: Prim's algorithm can be implemented using Fibonacci heap and it never accepts cycles. And Prim's algorithm follows greedy approach. Prim's algorithms span from one vertex to another.

1. Dijkstra's Algorithm is used to solve _____ problems.

a) All pair shortest path

b) Single source shortest path

c) Network flow

d) Sorting

Answer: b

Explanation: Dijkstra's Algorithm is used for solving single source shortest path problems. In this algorithm, a single node is fixed as a source node and shortest paths from this node to all other nodes in graph is found.

2. Which of the following is the most commonly used data structure for implementing Dijkstra's Algorithm?

a) Max priority queue

b) Stack

c) Circular queue

d) Min priority queue

Answer: d

Explanation: Minimum priority queue is the most commonly used data structure for implementing Dijkstra's Algorithm because the required operations to be performed in Dijkstra's Algorithm match with specialty of a minimum priority queue.

3. What is the time complexity of Dijikstra's algorithm?

a) O(N)

b) O(N3)

c) O(N2)

d) O(logN)

Answer: c
Explanation: Time complexity of Dijkstra's algorithm is O(N2) because of the use of doubly nested for loops. It depends on how the table is manipulated.

4. Dijkstra's Algorithm cannot be applied on _____
a) Directed and weighted graphs
b) Graphs having negative weight function
c) Unweighted graphs
d) Undirected and unweighted graphs

Answer: b
Explanation: Dijkstra's Algorithm cannot be applied on graphs having negative weight function because calculation of cost to reach a destination node from the source node becomes complex.

6. How many priority queue operations are involved in Dijkstra's Algorithm?
a) 1
b) 3
c) 2
d) 4

Answer: b
Explanation: The number of priority queue operations involved is 3. They are insert, extract-min and decrease key.

8. The maximum number of times the decrease key operation performed in Dijkstra's algorithm will be equal to _____
a) Total number of vertices
b) Total number of edges
c) Number of vertices – 1
d) Number of edges – 1

Answer: B

1. The Bellmann Ford algorithm returns _____ value.
a) Boolean

b) Integer
c) String
d) Double

Answer: a
Explanation: The Bellmann Ford algorithm returns Boolean value whether there is a negative weight cycle that is reachable from the source.

2. Bellmann ford algorithm provides solution for _____ problems.
a) All pair shortest path
b) Sorting
c) Network flow
d) Single source shortest path

Answer: d
Explanation: Bellmann ford algorithm is used for finding solutions for single source shortest path problems. If the graph has no negative cycles that are reachable from the source then the algorithm produces the shortest paths and their weights.

5. What is the running time of Bellmann Ford Algorithm?
a) O(V)
b) O(V2)
c) O(ElogV)
d) O(VE)

Answer: d

6. How many times the for loop in the Bellmann Ford Algorithm gets executed?
a) V times
b) V-1
c) E
d) E-1

Answer: b
Explanation: The for loop in the Bellmann Ford Algorithm gets executed for V-1 times. After making V-1 passes, the algorithm checks for a negative weight cycle and returns appropriate boolean value.

7. Dijikstra's Algorithm is more efficient than Bellmann Ford Algorithm.
a) True
b) False

Answer: a
Explanation: The running time of Bellmann Ford Algorithm is O(VE) whereas Dijikstra's Algorithm has running time of only O(V2).

9. What is the basic principle behind Bellmann Ford Algorithm?
a) Interpolation
b) Extrapolation
c) Regression
d) Relaxation

Answer: d
Explanation: Relaxation methods which are also called as iterative methods in which an approximation to the correct distance is replaced progressively by more accurate values till an optimum solution is found.

10. Bellmann Ford Algorithm can be applied for _____
a) Undirected and weighted graphs
b) Undirected and unweighted graphs
c) Directed and weighted graphs
d) All directed graphs

Answer: c
Explanation: Bellmann Ford Algorithm can be applied for all directed and weighted graphs. The weight function in the graph may either be positive or negative.

11. Bellmann Ford algorithm was first proposed by _____
a) Richard Bellmann

b) Alfonso Shimbe

c) Lester Ford Jr

d) Edward F. Moore

Answer: b

14. Bellmann Ford Algorithm is an example for _____

a) Dynamic Programming

b) Greedy Algorithms

c) Linear Programming

d) Branch and Bound

Answer: a

3. Floyd Warshall Algorithm can be used for finding _____

a) Transitive closure

b) Minimum spanning tree

c) Topological sort

d) Single source shortest path

Answer: a