Topic 1. Array and Array Operations

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Array and Array Operations”.

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
View Answer

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);
View Answer

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

3. How do you instantiate an array in Java?
a) int arr[] = new int(3);
b) int arr[];
c) int arr[] = new int[3];
d) int arr() = new int(3);
View Answer

Answer: c
Explanation: Note that int arr[]; is declaration whereas int arr[] = new int[3]; is to instantiate an array.

4. Which of the following is the correct way to declare a multidimensional array in Java?
a) int[] arr;
b) int arr[[]];
c) int[][]arr;
d) int[[]] arr;
View Answer

Answer: c
Explanation: The syntax to declare multidimensional array in java is either int[][] arr; or int arr[][];

5. What is the output of the following Java code?

```
public class array
{
 public static void main(String args[])
 {
  int []arr = {1,2,3,4,5};
  System.out.println(arr[2]);
  System.out.println(arr[4]);
 }
}
```

a) 3 and 5
b) 5 and 3
c) 2 and 4
d) 4 and 2
View Answer

Answer: a
Explanation: Array indexing starts from 0.

6. What is the output of the following Java code?

```
public class array
{
 public static void main(String args[])
 {
  int []arr = {1,2,3,4,5};
  System.out.println(arr[5]);
 }
}
```

a) 4
b) 5
c) ArrayIndexOutOfBoundsException
d) InavlidInputException
View Answer

Answer: c
Explanation: Trying to access an element beyond the limits of an array gives ArrayIndexOutOfBoundsException.

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

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
View Answer

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 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
View Answer

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
View Answer

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
View Answer

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

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

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
View Answer

Answer: a
Explanation: Elements in an array are accessed randomly. In Linked lists, elements are accessed sequentially.
Topic 2. Stack Operations – 1

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Stack Operations – 1”.

1. Process of inserting an element in stack is called ____________
a) Create
b) Push
c) Evaluation
d) Pop
View Answer

Answer: b
Explanation: Push operation allows users to insert elements in the stack. If the stack is filled completely and trying to perform push operation stack – overflow can happen.

2. Process of removing an element from stack is called __________
a) Create
b) Push
c) Evaluation
d) Pop
View Answer

Answer: d
Explanation: Elements in the stack are removed using pop operation. Pop operation removes the top most element in the stack i.e. last entered element.

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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.
Topic 3. Stack Operations – 2

This set of Data Structure Interview Questions and Answers focuses on “Stack Operations – 2”.

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 /
View Answer

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
View Answer

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
View Answer

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
View Answer

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+/*
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

Answer: b
Explanation: Stacks are used for the implementation of Recursion.
Topic 4. Stack Operations – 3

This set of Data Structure Questions and Answers for Freshers focuses on “Stack Operations – 3”.

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

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 +)
View Answer

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 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*+
View Answer

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
View Answer

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
View Answer

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
View Answer

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

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

View Answer

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) abc X+ def ^^ –
b) abc X+ de^f^ –
c) ab+c Xd – e ^f^
d) -+aXbc^ ^def
View Answer

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

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
View Answer

Answer: b
Explanation: Stack follows LIFO(Last In First Out). So the removal order of elements are DCBA.
Topic 5. Queue Operations

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Queue Operations”.

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
View Answer

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
View Answer

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 follows 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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

Answer: b
Explanation: Queue always has two ends. So, single ended queue is not the type of queue.
Topic 6. Singly Linked Lists Operations – 1

This set of Data Structure Interview Questions & Answers focuses on “Singly Linked List Operations – 1”.

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
View Answer

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
View Answer

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
View Answer

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) θ(1)
View Answer

Answer: c
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 θ(n).

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(n^2)
d) O(n^3)
View Answer

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(n^2)
d) O(n^4)
View Answer

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(n^2)
d) O(n^3)
View Answer

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
View Answer

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 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 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));
View Answer

Answer: a
Explanation: As it represents the right way to create a node.
Topic 7. Singly Linked Lists Operations – 2

This set of Data Structure Interview Questions and Answers for freshers focuses on “Singly Linked Lists Operations – 2”.

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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(n^2), merge sort is preferred.
Topic 8. Singly Linked Lists Operations – 3

This set of Data Structure Questions and Answers for Experienced people focuses on “Singly Linked Lists Operations – 3”.

1. The following function reverse() is supposed to reverse a singly linked list. There is one line missing at the end of the function.

```
/* Link list node */
struct node
{
    int data;
    struct node* next;
};

/* head_ref is a double pointer which points to head (or start) pointer
  of linked list */
static void reverse(struct node** head_ref)
{
    struct node* prev   = NULL;
    struct node* current = *head_ref;
    struct node* next;
    while (current != NULL)
    {
        next  = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    /*ADD A STATEMENT HERE*/
}
```

What should be added in place of “/*ADD A STATEMENT HERE*/”, so that the function correctly reverses a linked list.
a) *head_ref = prev;
b) *head_ref = current;
c) *head_ref = next;
d) *head_ref = NULL;
View Answer

Answer: a
Explanation: *head_ref = prev; At the end of while loop, the prev pointer points to the last node of original linked list.
We need to change *head_ref so that the head pointer now starts pointing to the last node.

2. What is the output of following function for start pointing to first node of following linked list?

```
1->2->3->4->5->6
void fun(struct node* start)
{
    if(start == NULL)
    return;
    printf("%d  ", start->data);
    if(start->next != NULL )
    fun(start->next->next);
    printf("%d  ", start->data);
}
```

a) 1 4 6 6 4 1
b) 1 3 5 1 3 5
c) 1 2 3 5
d) 1 3 5 5 3 1
View Answer

Answer: d
Explanation: fun() prints alternate nodes of the given Linked List, first from head to end, and then from end to head.
If Linked List has even number of nodes, then skips the last node.

3. The following C function takes a simply-linked list as an input argument. It modifies the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank. Choose the correct alternative to replace the blank line.

```
typedef struct node
{
    int value;
    struct node *next;
}Node;
```

```
Node *move_to_front(Node *head)
{
    Node *p, *q;
    if ((head == NULL: || (head->next == NULL))
    return head;
    q = NULL; p = head;
    while (p-> next !=NULL)
    {
        q = p;
        p = p->next;
    }
   _______________________________
  return head;
}
```

a) q = NULL; p->next = head; head = p;
b) q->next = NULL; head = p; p->next = head;
c) head = p; p->next = q; q->next = NULL;
d) q->next = NULL; p->next = head; head = p;
View Answer

Answer: d
Explanation: When while loop completes its execution, node ‘p’ refers to the last node whereas the ‘q’ node refers to the node before ‘p’ in the linked list. q->next=NULL makes q as the last node. p->next=head places p as the first node. the head must be modified to ‘p’ as ‘p’ is the starting node of the list (head=p). Thus the sequence of steps are q->next=NULL, p->next=head, head=p.

4. The following C function takes a single-linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node
{
    int value;
    struct node *next;
};
void rearrange(struct node *list)
{
    struct node *p, * q;
    int temp;
    if ((!list) || !list->next)
      return;
    p = list;
    q = list->next;
    while(q)
    {
         temp = p->value;
         p->value = q->value;
         q->value = temp;
         p = q->next;
         q = p?p->next:0;
    }
}
```

a) 1, 2, 3, 4, 5, 6, 7
b) 2, 1, 4, 3, 6, 5, 7
c) 1, 3, 2, 5, 4, 7, 6
d) 2, 3, 4, 5, 6, 7, 1
View Answer

Answer: b
Explanation: The function rearrange() exchanges data of every node with its next node. It starts exchanging data from the first node itself.

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
View Answer

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
View Answer

Answer: a
Explanation: Following are simple steps.

```
    struct node *temp  = X->next;
    X->data  = temp->data;
    X->next  = temp->next;
    free(temp);
```

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
View Answer

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
View Answer

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.
Topic 9. Singly Linked Lists

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Singly Linked List”.

1. Which of the following is not a disadvantage to the usage of array?
a) Fixed size
b) There are chances of wastage of memory space if elements inserted in an array are lesser than the allocated size
c) Insertion based on position
d) Accessing elements at specified positions
View Answer

Answer: d
Explanation: Array elements can be accessed in two steps. First, multiply the size of the data type with the specified position, second, add this value to the base address. Both of these operations can be done in constant time, hence accessing elements at a given index/position is faster.

2. What is the time complexity of inserting at the end in dynamic arrays?
a) O(1)
b) O(n)
c) O(logn)
d) Either O(1) or O(n)
View Answer

Answer: d
Explanation: Depending on whether the array is full or not, the complexity in dynamic array varies. If you try to insert into an array that is not full, then the element is simply stored at the end, this takes O(1) time. If you try to insert into an array which is full, first you will have to allocate an array with double the size of the current array and then copy all the elements into it and finally insert the new element, this takes O(n) time.

3. What is the time complexity to count the number of elements in the linked list?

a) O(1)
b) O(n)
c) O(logn)
d) O(n^2)
View Answer

Answer: b
Explanation: To count the number of elements, you have to traverse through the entire list, hence complexity is O(n).

4. Which of the following performs deletion of the last element in the list? Given below is the Node class.

```
class Node
{
 protected Node next;
 protected Object ele;
 Node(Object e,Node n)
 {
  ele = e;
  next = n;
 }
 public void setNext(Node n)
 {
  next = n;
 }
 public void setEle(Object e)
 {
  ele = e;
 }
 public Node getNext()
 {
  return next;
 }
 public Object getEle()
 {
  return ele;
 }
}
class SLL
{
 Node head;
 int size;
 SLL()
 {
  size = 0;
 }
}
```

a)

```
public Node removeLast()
{
 if(size == 0)
 return null;
 Node cur;
 Node temp;
 cur = head;
 while(cur.getNext() != null)
 {
   temp = cur;
   cur = cur.getNext();
        }
 temp.setNext(null);
 size--;
 return cur;
}
```

b)

```
public void removeLast()
{
 if(size == 0)
 return null;
 Node cur;
 Node temp;
 cur = head;
 while(cur != null)
 {
  temp = cur;
  cur = cur.getNext();
        }
 temp.setNext(null);
 return cur;
}
```

c)

```
public void removeLast()
{
 if(size == 0)
     return null;
 Node cur;
 Node temp;
 cur = head;
 while(cur != null)
 {
  cur = cur.getNext();
  temp = cur;
  }
 temp.setNext(null);
 return cur;
}
```

d)

```
public void removeLast()
{
 if(size == 0)
  return null;
 Node cur;
 Node temp;
 cur = head;
 while(cur.getNext() != null)
 {
  cur = cur.getNext();
  temp = cur;
 }
 temp.setNext(null);
 return cur;
}
```

View Answer
Answer: a
Explanation: Since you have to traverse to the end of the list and delete the last node, you need two reference pointers. ‘cur’ to traverse all the way and find the last node, and ‘temp’ is a trailing pointer to ‘cur’. Once you reach the end of the list, setNext of ‘temp’ to null, ‘cur’ is not being pointed to by any node, and hence it is available for garbage collection.

5. What is the functionality of the following code?

```
public void function(Node node)
{
 if(size == 0)
  head = node;
 else
 {
  Node temp,cur;
  for(cur = head; (temp = cur.getNext())!=null; cur = temp);
  cur.setNext(node);
 }
 size++;
}
```

a) Inserting a node at the beginning of the list
b) Deleting a node at the beginning of the list
c) Inserting a node at the end of the list
d) Deleting a node at the end of the list
View Answer

Answer: c
Explanation: The for loop traverses through the list and then inserts a new node as cur.setNext(node);

6. What is the space complexity for deleting a linked list?
a) O(1)
b) O(n)
c) Either O(1) or O(n)
d) O(logn)
View Answer

Answer: a
Explanation: You need a temp variable to keep track of current node, hence the space complexity is O(1).

7. How would you delete a node in the singly linked list? The position to be deleted is given.

a)

```
public void delete(int pos)
{
 if(pos < 0)
 pos = 0;
 if(pos > size)
 pos = size;
 if( size == 0)
 return;
 if(pos == 0)
 head = head.getNext();
 else
 {
     Node temp = head;
     for(int i=1; i<pos; i++)
            {
  temp = temp.getNext();
            }
     temp.setNext(temp.getNext().getNext());
 }
     size--;
}
```

b)

```
public void delete(int pos)
{
 if(pos < 0)
 pos = 0;
 if(pos > size)
 pos = size;
 if( size == 0)
 return;
 if(pos == 0)
 head = head.getNext();
 else
 {
     Node temp = head;
     for(int i=1; i<pos; i++)
     {
  temp = temp.getNext();
     }
     temp.setNext(temp.getNext());
 }
     size--;
}
```

c)

```
public void delete(int pos)
{
        if(pos < 0)
 pos = 0;
 if(pos > size)
 pos = size;
 if( size == 0)
 return;
 if(pos == 0)
 head = head.getNext();
 else
 {
     Node temp = head;
     for(int i=1; i<pos; i++)
     {
  temp = temp.getNext().getNext();
            }
     temp.setNext(temp.getNext().getNext());
 }
     size--;
}
```

d)

```
public void delete(int pos)
{
        if(pos < 0)
        pos = 0;
        if(pos > size)
        pos = size;
        if( size == 0)
 return;
 if(pos == 0)
 head = head.getNext();
 else
 {
     Node temp = head;
     for(int i=0; i<pos; i++)
     {
  temp = temp.getNext();
     }
     temp.setNext(temp.getNext().getNext());
 }
 size--;
}
```

View Answer
Answer: a
Explanation: Loop through the list to get into position one behind the actual position given. temp.setNext(temp.getNext().getNext()) will delete the specified node.

8. Which of these is not an application of a linked list?
a) To implement file systems
b) For separate chaining in hash-tables
c) To implement non-binary trees
d) Random Access of elements
View Answer

Answer: d
Explanation: To implement file system, for separate chaining in hash-tables and to implement non-binary trees linked lists are used. Elements are accessed sequentially in linked list. Random access of elements is not an applications of linked list.

9. Which of the following piece of code has the functionality of counting the number of elements in the list?
a)

```
public int length(Node head)
{
 int size = 0;
 Node cur = head;
 while(cur!=null)
 {
     size++;
     cur = cur.getNext();
 }
 return size;
}
```

b)

```
public int length(Node head)
{
        int size = 0;
 Node cur = head;
 while(cur!=null)
 {
     cur = cur.getNext();
     size++;
 }
 return size;
}
```

c)

```
public int length(Node head)
{
 int size = 0;
 Node cur = head;
 while(cur!=null)
 {
     size++;
     cur = cur.getNext();
 }
}
```

d)

```
public int length(Node head)
{
 int size = 0;
 Node cur = head;
 while(cur!=null)
 {
     size++;
     cur = cur.getNext().getNext();
 }
 return size;
}
```

View Answer
Answer: a
Explanation: ‘cur’ pointer traverses through list and increments the size variable until the end of list is reached.

10. How do you insert an element at the beginning of the list?
a)

```
public void insertBegin(Node node)
{
 node.setNext(head);
 head = node;
 size++;
}
```

b)

```
public void insertBegin(Node node)
{
 head = node;
 node.setNext(head);
 size++;
}
```

c)

```
public void insertBegin(Node node)
{
 Node temp = head.getNext()
 node.setNext(temp);
 head = node;
 size++;
}
```

d)

```
public void insertBegin(Node node)
{
 Node temp = head.getNext()
 node.setNext(temp);
 node = head;
 size++;
}
```

View Answer
Answer: a
Explanation: Set the ‘next’ pointer point to the head of the list and then make this new node as the head.

11. What is the functionality of the following piece of code?

```
public int function(int data)
{
 Node temp = head;
 int var = 0;
 while(temp != null)
 {
  if(temp.getData() == data)
  {
   return var;
  }
  var = var+1;
  temp = temp.getNext();
 }
 return Integer.MIN_VALUE;
}
```

a) Find and delete a given element in the list
b) Find and return the given element in the list
c) Find and return the position of the given element in the list
d) Find and insert a new element in the list
View Answer

Answer: c

Explanation: When temp is equal to data, the position of data is returned.
Topic 10. Doubly Linked Lists

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Doubly Linked List”.

1. Which of the following is false about a doubly linked list?
a) We can navigate in both the directions
b) It requires more space than a singly linked list
c) The insertion and deletion of a node take a bit longer
d) Implementing a doubly linked list is easier than singly linked list
View Answer

Answer: d
Explanation: A doubly linked list has two pointers ‘left’ and ‘right’ which enable it to traverse in either direction. Compared to singly liked list which has only a ‘next’ pointer, doubly linked list requires extra space to store this extra pointer. Every insertion and deletion requires manipulation of two pointers, hence it takes a bit longer time. Implementing doubly linked list involves setting both left and right pointers to correct nodes and takes more time than singly linked list.

2. Given the Node class implementation, select one of the following that correctly inserts a node at the tail of the list.

```
public class Node
{
 protected int data;
 protected Node prev;
 protected Node next;
 public Node(int data)
 {
  this.data = data;
  prev = null;
  next = null;
 }
 public Node(int data, Node prev, Node next)
 {
  this.data = data;
  this.prev = prev;
  this.next = next;
 }
 public int getData()
 {
  return data;
 }
 public void setData(int data)
 {
  this.data = data;
 }
 public Node getPrev()
 {
  return prev;
 }
 public void setPrev(Node prev)
 {
  this.prev = prev;
 }
 public Node getNext
 {
  return next;
 }
 public void setNext(Node next)
 {
  this.next = next;
 }
}
public class DLL
{
 protected Node head;
 protected Node tail;
 int length;
 public DLL()
 {
  head = new Node(Integer.MIN_VALUE,null,null);
  tail = new Node(Integer.MIN_VALUE,null,null);
  head.setNext(tail);
  length = 0;
 }
}
```

a)

```
public void insertRear(int data)
{
 Node node = new Node(data,tail.getPrev(),tail);
 node.getPrev().setNext(node);
 tail.setPrev(node);
 length++;
}
```

b)

```
public void insertRear(int data)
{
 Node node = new Node(data,tail.getPrev(),tail);
 node.getPrev().getPrev().setNext(node);
 tail.setPrev(node);
 length++;
}
```

c)

```
public void insertRear(int data)
{
 Node node = new Node(data,tail.getPrev(),tail);
 node.getPrev().setNext(tail);
 tail.setPrev(node);
 length++;
}
```

d)

```
public void insertRear(int data)
{
 Node node = new Node(data,head,tail);
 node.getPrev().setNext(node);
 tail.setPrev(node);
 length++;
}
```

View Answer
Answer: a
Explanation: First create a new node whose 'prev' points to the node pointed to by the 'prev' of tail. The 'next' of the new node should point to tail. Set the 'prev' of tail to point to new node and the 'prev' of new node to point to the new node.

3. What is a memory efficient double linked list?
a) Each node has only one pointer to traverse the list back and forth
b) The list has breakpoints for faster traversal
c) An auxiliary singly linked list acts as a helper list to traverse through the doubly linked list
d) A doubly linked list that uses bitwise AND operator for storing addresses
View Answer

Answer: a
Explanation: Memory efficient doubly linked list has only one pointer to traverse the list back and forth. The implementation is based on pointer difference. It uses bitwise XOR operator to store the front and rear pointer addresses. Instead of storing actual memory address, every node store the XOR address of previous and next nodes.

4. Which of the following piece of code removes the node from a given position?
a)

```
public void remove(int pos)
{
 if(pos<0 || pos>=size)
 {
  System.out.println("Invalid position");
  return;
 }
 else
 {
  if(head == null)
   return;
  if(pos == 0)
  {
   head = head.getNext();
   if(head == null)
   tail = null;
  }
        else
        {
   Node temp = head;
   for(int i=1; i<position; i++)
   temp = temp.getNext();
  }
  temp.getNext().setPrev(temp.getPrev());
  temp.getPrev().setNext(temp.getNext());
 }
 size--;
}
```

b)

```
public void remove(int pos)
{
 if(pos<0 || pos>=size)
 {
  System.out.println("Invalid position");
  return;
 }
 else
 {
  if(head == null)
  return;
  if(pos == 0)
  {
   head = head.getNext();
   if(head == null)
   tail = null;
  }
  else
  {
   Node temp = head;
   for(int i=1; i<position; i++)
   temp = temp.getNext();
  }
  temp.getNext().setPrev(temp.getNext());
  temp.getPrev().setNext(temp.getPrev());
 }
 size--;
}
```

c)

```
public void remove(int pos)
{
 if(pos<0 || pos>=size)
 {
  System.out.println("Invalid position");
  return;
 }
 else
 {
  if(head == null)
   return;
  if(pos == 0)
  {
   head = head.getNext();
   if(head == null)
   tail = null;
  }
  else
  {
   Node temp = head;
   for(int i=1; i<position; i++)
   temp = temp.getNext().getNext();
  }
  temp.getNext().setPrev(temp.getPrev());
  temp.getPrev().setNext(temp.getNext());
 }
 size--;
}
```

d)

```
public void remove(int pos)
{
 if(pos<0 || pos>=size)
 {
  System.out.println("Invalid position");
  return;
 }
 else
 {
  if(head == null)
   return;
  if(pos == 0)
  {
   head = head.getNext();
   if(head == null)
   tail = null;
  }
  else
  {
   Node temp = head;
   for(int i=1; i<position; i++)
   temp = temp.getNext().getNext();
  }
  temp.getNext().setPrev(temp.getNext());
  temp.getPrev().setNext(temp.getPrev());
 }
 size--;
}
```

View Answer
Answer: a
Explanation: If the position to be deleted is not the head, advance to the given position and manipulate the previous and next pointers of next and previous nodes respectively.

5. How do you calculate the pointer difference in a memory efficient double linked list?
a) head xor tail
b) pointer to previous node xor pointer to next node
c) pointer to previous node – pointer to next node
d) pointer to next node – pointer to previous node
View Answer

Answer: b
Explanation: The pointer difference is calculated by taking XOR of pointer to previous node and pointer to the next node.

6. What is the worst case time complexity of inserting a node in a doubly linked list?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(1)
View Answer

Answer: c
Explanation: In the worst case, the position to be inserted maybe at the end of the list, hence you have to traverse through the entire list to get to the correct position, hence O(n).

7. How do you insert a node at the beginning of the list?
a)

```
public class insertFront(int data)
{
 Node node = new Node(data, head, head.getNext());
 node.getNext().setPrev(node);
 head.setNext(node);
 size++;
}
```

b)

```
public class insertFront(int data)
{
 Node node = new Node(data, head, head);
 node.getNext().setPrev(node);
 head.setNext(node);
 size++;
}
```

c)

```
public class insertFront(int data)
{
 Node node = new Node(data, head, head.getNext());
 node.getNext().setPrev(head);
 head.setNext(node);
 size++;
}
```

d)

```
public class insertFront(int data)
{
 Node node = new Node(data, head, head.getNext());
 node.getNext().setPrev(node);
 head.setNext(node.getNext());
 size++;
}
```

View Answer
Answer: a
Explanation: The new node’s previous pointer will point to head and next pointer will point to the current next of head.

8. Consider the following doubly linked list: head-1-2-3-4-5-tail. What will be the list after performing the given sequence of operations?

```
 Node temp = new Node(6,head,head.getNext());
 Node temp1 = new Node(0,tail.getPrev(),tail);
 head.setNext(temp);
 temp.getNext().setPrev(temp);
 tail.setPrev(temp1);
 temp1.getPrev().setNext(temp1);
```

a) head-0-1-2-3-4-5-6-tail
b) head-1-2-3-4-5-6-tail
c) head-6-1-2-3-4-5-0-tail
d) head-0-1-2-3-4-5-tail
View Answer

Answer: c
Explanation: The given sequence of operations performs addition of nodes at the head and tail of the list.

9. What is the functionality of the following piece of code?

```
public int function()
{
 Node temp = tail.getPrev();
 tail.setPrev(temp.getPrev());
 temp.getPrev().setNext(tail);
 size--;
 return temp.getItem();
}
```

a) Return the element at the tail of the list but do not remove it
b) Return the element at the tail of the list and remove it from the list
c) Return the last but one element from the list but do not remove it
d) Return the last but one element at the tail of the list and remove it from the list
View Answer

Answer: b
Explanation: The previous and next pointers of the tail and the last but one element are manipulated, this suggests that the last node is being removed from the list.

10. Consider the following doubly linked list: head-1-2-3-4-5-tail. What will be the list after performing the given sequence of operations?

```
 Node temp = new Node(6,head,head.getNext());
 head.setNext(temp);
 temp.getNext().setPrev(temp);
 Node temp1 = tail.getPrev();
 tail.setPrev(temp1.getPrev());
 temp1.getPrev().setNext(tail);
```

a) head-6-1-2-3-4-5-tail
b) head-6-1-2-3-4-tail
c) head-1-2-3-4-5-6-tail
d) head-1-2-3-4-5-tail
View Answer

Answer: b
Explanation: A new node is added to the head of the list and a node is deleted from the tail end of the list.
Topic 11. Circular Linked Lists

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Circular Linked List”.

1. What differentiates a circular linked list from a normal linked list?
a) You cannot have the ‘next’ pointer point to null in a circular linked list
b) It is faster to traverse the circular linked list
c) You may or may not have the ‘next’ pointer point to null in a circular linked list
d) Head node is known in circular linked list
View Answer

Answer: c
Explanation: The ‘next’ pointer points to null only when the list is empty, otherwise it points to the head of the list. Every node in a circular linked list can be a starting point(head).

2. How do you count the number of elements in the circular linked list?
a)

```
public int length(Node head)
{
 int length = 0;
 if( head == null)
  return 0;
 Node temp = head.getNext();
 while(temp != head)
 {
  temp = temp.getNext();
  length++;
 }
 return length;
}
```

b)

```
public int length(Node head)
{
 int length = 0;
 if( head == null)
  return 0;
 Node temp = head.getNext();
 while(temp != null)
 {
  temp = temp.getNext();
  length++;
 }
 return length;
}
```

c)

```
public int length(Node head)
{
 int length = 0;
 if( head == null)
  return 0;
 Node temp = head.getNext();
 while(temp != head && temp != null)
 {
  temp = head.getNext();
  length++;
 }
 return length;
}
```

d)

```
public int length(Node head)
{
 int length = 0;
 if( head == null)
  return 0;
 Node temp = head.getNext();
 while(temp != head && temp == null)
 {
  temp = head.getNext();
  length++;
 }
 return length;
}
```

View Answer
Answer: a
Explanation: If the head is null, it means that the list is empty. Otherwise, traverse the list until the head of the list is reached.

3. What is the functionality of the following piece of code? Select the most appropriate.

```
public void function(int data)
{
 int flag = 0;
 if( head != null)
 {
  Node temp = head.getNext();
  while((temp != head) && (!(temp.getItem() == data)))
  {
   temp = temp.getNext();
   flag = 1;
   break;
  }
 }
 if(flag)
  System.out.println("success");
 else
  System.out.println("fail");
}
```

a) Print success if a particular element is not found
b) Print fail if a particular element is not found
c) Print success if a particular element is equal to 1
d) Print fail if the list is empty
View Answer

Answer: b
Explanation: The function prints fail if the given element is not found. Note that this option is inclusive of option “Print fail if the list is empty”, the list being empty is one of the cases covered.

4. What is the time complexity of searching for an element in a circular linked list?
a) O(n)
b) O(nlogn)
c) O(1)
d) O(n^2)
View Answer

Answer: a
Explanation: In the worst case, you have to traverse through the entire list of n elements.

5. Which of the following application makes use of a circular linked list?
a) Undo operation in a text editor
b) Recursive function calls
c) Allocating CPU to resources
d) Implement Hash Tables
View Answer

Answer: c
Explanation: Generally, round robin fashion is employed to allocate CPU time to resources which makes use of the circular linked list data structure. Recursive function calls use stack data structure. Undo Operation in text editor uses doubly linked lists. Hash tables uses singly linked lists.

6. Choose the code snippet which inserts a node to the head of the list?
a)

```
public void insertHead(int data)
{
 Node temp = new Node(data);
 Node cur = head;
 while(cur.getNext() != head)
  cur = cur.getNext()
 if(head == null)
 {
  head = temp;
  head.setNext(head);
 }
 else
 {
  temp.setNext(head);
  head = temp;
  cur.setNext(temp);
 }
 size++;
}
```

b)

```
public void insertHead(int data)
{
 Node temp = new Node(data);
 while(cur != head)
  cur = cur.getNext()
 if(head == null)
 {
  head = temp;
  head.setNext(head);
 }
 else
 {
  temp.setNext(head.getNext());
  cur.setNext(temp);
 }
 size++;
}
```

c)

```
public void insertHead(int data)
{
 Node temp = new Node(data);
 if(head == null)
 {
  head = temp;
  head.setNext(head);
 }
 else
 {
  temp.setNext(head.getNext());
  head = temp;
 }
 size++;
}
```

d)

```
public void insertHead(int data)
{
 Node temp = new Node(data);
 if(head == null)
 {
  head = temp;
  head.setNext(head.getNext());
 }
 else
 {
  temp.setNext(head.getNext());
  head = temp;
 }
 size++;
}
```

View Answer
Answer: a
Explanation: If the list is empty make the new node as 'head', otherwise traverse the list to the end and make its 'next' pointer point to the new node, set the new node's next point to the current head and make the new node as the head.

7. What is the functionality of the following code? Choose the most appropriate answer.

```
public int function()
{
 if(head == null)
  return Integer.MIN_VALUE;
 int var;
 Node temp = head;
 while(temp.getNext() != head)
  temp = temp.getNext();
 if(temp == head)
 {
  var = head.getItem();
  head = null;
  return var;
 }
 temp.setNext(head.getNext());
 var = head.getItem();
 head = head.getNext();
 return var;
}
```

a) Return data from the end of the list
b) Returns the data and deletes the node at the end of the list
c) Returns the data from the beginning of the list
d) Returns the data and deletes the node from the beginning of the list
View Answer

Answer: d
Explanation: First traverse through the list to find the end node, then manipulate the ‘next’ pointer such that it points to the current head’s next node, return the data stored in head and make this next node as the head.

8. What is the functionality of the following code? Choose the most appropriate answer.

```
public int function()
{
 if(head == null)
  return Integer.MIN_VALUE;
 int var;
 Node temp = head;
 Node cur;
 while(temp.getNext() != head)
 {
  cur = temp;
  temp = temp.getNext();
 }
 if(temp == head)
 {
  var = head.getItem();
  head = null;
  return var;
 }
 var = temp.getItem();
 cur.setNext(head);
 return var;
}
```

a) Return data from the end of the list
b) Returns the data and deletes the node at the end of the list
c) Returns the data from the beginning of the list
d) Returns the data and deletes the node from the beginning of the list
View Answer

Answer: b
Explanation: First traverse through the list to find the end node, also have a trailing pointer to find the penultimate node, make this trailing pointer’s ‘next’ point to the head and return the data stored in the ‘temp’ node.

9. Which of the following is false about a circular linked list?
a) Every node has a successor
b) Time complexity of inserting a new node at the head of the list is O(1)
c) Time complexity for deleting the last node is O(n)
d) We can traverse the whole circular linked list by starting from any point
View Answer

Answer: b
Explanation: Time complexity of inserting a new node at the head of the list is O(n) because you have to traverse through the list to find the tail node.

10. Consider a small circular linked list. How to detect the presence of cycles in this list effectively?
a) Keep one node as head and traverse another temp node till the end to check if its ‘next points to head
b) Have fast and slow pointers with the fast pointer advancing two nodes at a time and slow pointer advancing by one node at a time
c) Cannot determine, you have to pre-define if the list contains cycles
d) Circular linked list itself represents a cycle. So no new cycles cannot be generated
View Answer

Answer: b
Explanation: Advance the pointers in such a way that the fast pointer advances two nodes at a time and slow pointer advances one node at a time and check to see if at any given instant of time if the fast pointer points to slow pointer or if the fast pointer’s ‘next’ points to the slow pointer. This is applicable for smaller lists.
Topic 12. Stack using Array

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Stack using Array”.

1. Which of the following real world scenarios would you associate with a stack data structure?
a) piling up of chairs one above the other
b) people standing in a line to be serviced at a counter
c) offer services based on the priority of the customer
d) tatkal Ticket Booking in IRCTC
View Answer

Answer: a
Explanation: Stack follows Last In First Out (LIFO) policy. Piling up of chairs one above the other is based on LIFO, people

standing in a line is a queue and if the service is based on priority, then it can be associated with a priority queue. Tatkal Ticket Booking Follows First in First Out Policy. People who click the book now first will enter the booking page first.

2. What does the following function check for? (all necessary headers to be included and function is called from main)

```
#define MAX 10

typedef struct stack
{
    int top;
    int item[MAX];
}stack;

int function(stack *s)
{
    if(s->top == -1)
        return 1;
    else return 0;
}
```

a) full stack
b) invalid index
c) empty stack
d) infinite stack
View Answer

Answer: c
Explanation: An empty stack is represented with the top-of-the-stack(‘top’ in this case) to be equal to -1.

3. What does ‘stack underflow’ refer to?
a) accessing item from an undefined stack
b) adding items to a full stack
c) removing items from an empty stack
d) index out of bounds exception
View Answer

Answer: c
Explanation: Removing items from an empty stack is termed as stack underflow.

4. What is the output of the following program?

```
public class Stack
{
 protected static final int CAPACITY = 100;
 protected int size,top = -1;
 protected Object stk[];

 public Stack()
 {
  stk = new Object[CAPACITY];
 }

 public void push(Object item)
 {
  if(size_of_stack==size)
  {
   System.out.println("Stack overflow");
    return;
  }
  else
  {
   top++;
   stk[top]=item;
  }
 }
 public Object pop()
 {
  if(top<0)
  {
   return -999;
  }
  else
  {
   Object ele=stk[top];
   top--;
   size_of_stack--;
   return ele;
  }
 }
}
```

```
public class StackDemo
{
 public static void main(String args[])
 {
  Stack myStack = new Stack();
  myStack.push(10);
  Object element1 = myStack.pop();
  Object element2 = myStack.pop();
  System.out.println(element2);
 }
}
```

a) stack is full
b) 20
c) 0
d) -999
View Answer

Answer: d
Explanation: The first call to pop() returns 10, whereas the second call to pop() would result in stack underflow and the program returns -999.

5. What is the time complexity of pop() operation when the stack is implemented using an array?
a) O(1)
b) O(n)
c) O(logn)
d) O(nlogn)
View Answer

Answer: a
Explanation: pop() accesses only one end of the structure, and hence constant time.

6. Which of the following array position will be occupied by a new element being pushed for a stack of size N elements(capacity of stack > N)?
a) S[N-1]
b) S[N]
c) S[1]
d) S[0]
View Answer

Answer: b
Explanation: Elements are pushed at the end, hence N.

7. What happens when you pop from an empty stack while implementing using the Stack ADT in Java?
a) Undefined error
b) Compiler displays a warning
c) EmptyStackException is thrown
d) NoStackException is thrown
View Answer

Answer: c
Explanation: The Stack ADT throws an EmptyStackException if the stack is empty and a pop() operation is tried on it.

8. What is the functionality of the following piece of Java code?
Assume: ‘a’ is a non empty array of integers, the Stack class creates an array of specified size and provides a top pointer indicating TOS(top of stack), push and pop have normal meaning.

```
public void some_function(int[] a)
{
 Stack S=new Stack(a.length);
 int[] b=new int[a.length];
 for(int i=0;i<a.length;i++)
 {
  S.push(a[i]);
 }
 for(int i=0;i<a.length;i++)
 {
  b[i]=(int)(S.pop());
 }
 System.out.println("output :");
 for(int i=0;i<b.length;i++)
 {
  System.out.println(b[i]);
 }
}
```

a) print alternate elements of array
b) duplicate the given array
c) parentheses matching
d) reverse the array
View Answer

Answer: d
Explanation: Every element from the given array ‘a’ is pushed into the stack, and then the elements are popped out into the array ‘b’. Stack is a LIFO structure, this results in reversing the given array.

9. Array implementation of Stack is not dynamic, which of the following statements supports this argument?
a) space allocation for array is fixed and cannot be changed during run-time
b) user unable to give the input for stack operations
c) a runtime exception halts execution
d) improper program compilation
View Answer

Answer: a
Explanation: You cannot modify the size of an array once the memory has been allocated, adding fewer elements than the array size would cause wastage of space, and adding more elements than the array size at run time would cause Stack Overflow.

10. Which of the following array element will return the top-of-the-stack-element for a stack of size N elements(capacity of stack > N)?
a) S[N-1]
b) S[N]
c) S[N-2]
d) S[N+1]
View Answer

Answer: a
Explanation: Array indexing start from 0, hence N-1 is the last index.
Topic 13. Stack using Linked List

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Stack using Linked List”.

1. What is the best case time complexity of deleting a node in a Singly Linked list?
a) O (n)
b) O (n^2)
c) O (nlogn)
d) O (1)
View Answer

Answer: d
Explanation: Deletion of the head node in the linked list is taken as the best case. The successor of the head node is changed to head and deletes the predecessor of the newly assigned head node. This process completes in O(1) time.

2. Which of the following statements are not correct with respect to Singly Linked List(SLL) and Doubly Linked List(DLL)?
a) Complexity of Insertion and Deletion at known position is O(n) in SLL and O(1) in DLL
b) SLL uses lesser memory per node than DLL
c) DLL has more searching power than SLL
d) Number of node fields in SLL is more than DLL
View Answer

Answer: d
Explanation: To insert and delete at known positions requires complete traversal of the list in worst case in SLL, SLL consists of an item and a node field, while DLL has an item and two node fields, hence SLL occupies lesser memory, DLL can be traversed both ways(left and right), while SLL can traverse in only one direction, hence more searching power of DLL. Node fields in SLL is 2 (data and address of next node) whereas in DLL is 3(data, address to next node, address to previous node).

3. Given below is the Node class to perform basic list operations and a Stack class with a no arg constructor.
Select from the options the appropriate pop() operation that can be included in the Stack class. Also ‘first’ is the top-of-the-stack.

```
class Node
{
 protected Node next;
 protected Object ele;
 Node()
 {
  this(null,null);
 }
 Node(Object e,Node n)
 {
  ele=e;
  next=n;
 }
 public void setNext(Node n)
 {
  next=n;
 }
 public void setEle(Object e)
 {
  ele=e;
 }
 public Node getNext()
 {
  return next;
 }
 public Object getEle()
 {
  return ele;
 }
}

class Stack
{
 Node first;
 int size=0;
 Stack()
 {
  first=null;
 }
}
```

a)

```
public Object pop()
{
 if(size == 0)
 System.out.println("underflow");
 else
 {
  Object o = first.getEle();
  first = first.getNext();
  size--;
  return o;
 }
}
```

b)

```
public Object pop()
{
 if(size == 0)
 System.out.println("underflow");
 else
 {
  Object o = first.getEle();
  first = first.getNext().getNext();
  size--;
  return o;
 }
}
```

c)

```
public Object pop()
{
 if(size == 0)
 System.out.println("underflow");
 else
 {
  first = first.getNext();
  Object o = first.getEle();
  size--;
  return o;
 }
}
```

d)

```
public Object pop()
{
 if(size == 0)
 System.out.println("underflow");
 else
 {
  first = first.getNext().getNext();
  Object o = first.getEle();
  size--;
  return o;
 }
}
```

View Answer
Answer: a
Explanation: pop() should return the Object pointed to by the node ‘first’. The sequence of operations is, first, get the element stored at node ‘first’ using getEle(), and second, make the node point to the next node using getNext().

4. What does the following function do?

```
public Object some_func()throws emptyStackException
{
 if(isEmpty())
  throw new emptyStackException("underflow");
 return first.getEle();
}
```

a) pop
b) delete the top-of-the-stack element
c) retrieve the top-of-the-stack element
d) push operation
View Answer

Answer: c
Explanation: This code is only retrieving the top element, note that it is not equivalent to pop operation as you are not setting the ‘next’ pointer point to the next node in sequence.

5. What is the functionality of the following piece of code?

```
public void display()
{
 if(size == 0)
  System.out.println("underflow");
 else
 {
  Node current = first;
  while(current != null)
  {
   System.out.println(current.getEle());
   current = current.getNext();
  }
 }
}
```

a) reverse the list
b) display the list
c) display the list excluding top-of-the-stack-element
d) reverse the list excluding top-of-the-stack-element
View Answer

Answer: b
Explanation: An alias of the node ‘first’ is created which traverses through the list and displays the elements.

6. What does ‘stack overflow’ refer to?
a) accessing item from an undefined stack
b) adding items to a full stack
c) removing items from an empty stack
d) index out of bounds exception
View Answer

Answer: b
Explanation: Adding items to a full stack is termed as stack underflow.

7. Given below is the Node class to perform basic list operations and a Stack class with a no arg constructor. Select from the options the appropriate push() operation that can be included in the Stack class. Also ‘first’ is the top-of-the-stack.

```
class Node
{
 protected Node next;
 protected Object ele;
 Node()
 {
  this(null,null);
 }
 Node(Object e,Node n)
 {
  ele=e;
  next=n;
 }
 public void setNext(Node n)
 {
  next=n;
 }
 public void setEle(Object e)
 {
  ele=e;
 }
 public Node getNext()
 {
  return next;
 }
 public Object getEle()
 {
  return ele;
 }
}

class Stack
{
 Node first;
 int size=0;
 Stack()
 {
  first=null;
 }
}
```

a)

```
public void push(Object item)
{
 Node temp = new Node(item,first);
 first = temp;
 size++;
}
```

b)

```
public void push(Object item)
{
 Node temp = new Node(item,first);
 first = temp.getNext();
 size++;
}
```

c)

```
public void push(Object item)
{
 Node temp = new Node();
 first = temp.getNext();
 first.setItem(item);
 size++;
}
```

d)

```
public void push(Object item)
{
 Node temp = new Node();
 first = temp.getNext.getNext();
 first.setItem(item);
 size++;
}
```

View Answer
Answer: a
Explanation: To push an element into the stack, first create a new node with the next pointer point to the current top-of-the-stack node, then make this node as top-of-the-stack by assigning it to ‘first’.

8. Consider these functions:
push() : push an element into the stack
pop() : pop the top-of-the-stack element
top() : returns the item stored in top-of-the-stack-node
What will be the output after performing these sequence of operations

```
push(20);
push(4);
top();
pop();
pop();
push(5);
top();
```

a) 20
b) 4
c) stack underflow
d) 5
View Answer

Answer: d
Explanation: 20 and 4 which were pushed are popped by the two pop() statements, the recent push() is 5, hence top() returns 5.

9. Which of the following data structures can be used for parentheses matching?
a) n-ary tree
b) queue
c) priority queue
d) stack
View Answer

Answer: d
Explanation: For every opening brace, push it into the stack, and for every closing brace, pop it off the stack. Do not take action for any other character. In the end, if the stack is empty, then the input has balanced parentheses.

10. Minimum number of queues to implement stack is ___________
a) 3
b) 4
c) 1
d) 2
View Answer

Answer: c
Explanation: Use one queue and one counter to count the number of elements in the queue.
Topic 14. Queue using Array

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Queue using Array”.

1. Which of the following properties is associated with a queue?
a) First In Last Out
b) First In First Out
c) Last In First Out
d) Last In Last Out
View Answer

Answer: b
Explanation: Queue follows First In First Out structure.

2. In a circular queue, how do you increment the rear end of the queue?
a) rear++
b) (rear+1) % CAPACITY
c) (rear % CAPACITY)+1
d) rear–
View Answer

Answer: b
Explanation: Ensures rear takes the values from 0 to (CAPACITY-1).

3. What is the term for inserting into a full queue known as?
a) overflow
b) underflow
c) null pointer exception

d) program won’t be compiled
View Answer

Answer: a
Explanation: Just as stack, inserting into a full queue is termed overflow.

4. What is the time complexity of enqueue operation?
a) O(logn)
b) O(nlogn)
c) O(n)
d) O(1)
View Answer

Answer: d
Explanation: Enqueue operation is at the rear end, it takes O(1) time to insert a new item into the queue.

5. What does the following Java code do?

```
public Object function()
{
 if(isEmpty())
 return -999;
 else
 {
  Object high;
  high = q[front];
  return high;
 }
}
```

a) Dequeue
b) Enqueue
c) Return the front element
d) Return the last element
View Answer

Answer: c
Explanation: q[front] gives the element at the front of the queue, since we are not moving the ‘front’ to the next element, it is not a dequeue operation.

6. What is the need for a circular queue?
a) effective usage of memory
b) easier computations
c) to delete elements based on priority
d) implement LIFO principle in queues
View Answer

Answer: a
Explanation: In a linear queue, dequeue operation causes the starting elements of the array to be empty, and there is no way you can use that space, while in a circular queue, you can effectively use that space. Priority queue is used to delete the elements based on their priority. Higher priority elements will be deleted first whereas lower priority elements will be deleted next. Queue data structure always follows FIFO principle.

7. Which of the following represents a dequeue operation? (count is the number of elements in the queue)
a)

```
public Object dequeue()
{
 if(count == 0)
 {
  System.out.println("Queue underflow");
  return 0;
 }
 else
 {
  Object ele = q[front];
  q[front] = null;
  front = (front+1)%CAPACITY;
  count--;
  return ele;
 }
}
```

b)

```
public Object dequeue()
{
 if(count == 0)
 {
  System.out.println("Queue underflow");
  return 0;
 }
 else
 {
  Object ele = q[front];
  front = (front+1)%CAPACITY;
  q[front] = null;
  count--;
  return ele;
 }
}
```

c)

```
public Object dequeue()
{
 if(count == 0)
 {
  System.out.println("Queue underflow");
  return 0;
 }
 else
 {
  front = (front+1)%CAPACITY;
  Object ele = q[front];
  q[front] = null;
  count--;
  return ele;
 }
}
```

d)

```
public Object dequeue()
{
 if(count == 0)
 {
  System.out.println("Queue underflow");
  return 0;
 }
 else
 {
  Object ele = q[front];
  q[front] = null;
  front = (front+1)%CAPACITY;
  return ele;
  count--;
 }
}
```

View Answer
Answer: a
Explanation: Dequeue removes the first element from the queue, ‘front’ points to the front end of the queue and returns the first element.

8. Which of the following best describes the growth of a linear queue at runtime? (Q is the original queue, size() returns the number of elements in the queue)
a)

```
private void expand()
{
 int length = size();
 int[] newQ = new int[length<<1];
 for(int i=front; i<=rear; i++)
 {
  newQ[i-front] = Q[i%CAPACITY];
 }
 Q = newQ;
 front = 0;
 rear = size()-1;
}
```

b)

```
private void expand()
{
 int length = size();
 int[] newQ = new int[length<<1];
 for(int i=front; i<=rear; i++)
 {
  newQ[i-front] = Q[i%CAPACITY];
 }
 Q = newQ;
}
```

c)

```
private void expand()
{
 int length = size();
 int[] newQ = new int[length<<1];
 for(int i=front; i<=rear; i++)
 {
  newQ[i-front] = Q[i];
 }
 Q = newQ;
 front = 0;
 rear = size()-1;
}
```

d)

```
private void expand()
{
 int length = size();
 int[] newQ = new int[length*2];
 for(int i=front; i<=rear; i++)
 {
  newQ[i-front] = Q[i%CAPACITY];
 }
 Q = newQ;
}
```

View Answer
Answer: a
Explanation: A common technique to expand the size of array at run time is simply to double the size. Create a new array of double the previous size and copy all the elements, after copying do not forget to assign front = 0 and rear = size()-1, as these are necessary to maintain the decorum of the queue operations.

9. What is the space complexity of a linear queue having n elements?
a) O(n)
b) O(nlogn)
c) O(logn)
d) O(1)
View Answer

Answer: a
Explanation: The space complexity of an algorithm is the total space taken by the algorithm with respect to the input size. Space complexity includes both Auxiliary space and space used by input. Because there are n elements the space complexity of a linear queue having n elements is O(n).

10. What is the output of the following Java code?

```
public class CircularQueue
{
 protected static final int CAPACITY = 100;
 protected int size,front,rear;
 protected Object q[];
 int count = 0;

 public CircularQueue()
 {
  this(CAPACITY);
 }
 public CircularQueue (int n)
 {
  size = n;
  front = 0;
  rear = 0;
  q = new Object[size];
 }


 public void enqueue(Object item)
 {
  if(count == size)
  {
   System.out.println("Queue overflow");
    return;
  }
  else
  {
   q[rear] = item;
   rear = (rear+1)%size;
   count++;
  }
 }
 public Object dequeue()
 {
  if(count == 0)
  {
   System.out.println("Queue underflow");
   return 0;
  }
  else
  {
   Object ele = q[front];
   q[front] = null;
   front = (front+1)%size;
   count--;
   return ele;
  }
 }
 public Object frontElement()
 {
  if(count == 0)
  return -999;
  else
  {
   Object high;
   high = q[front];
   return high;
  }
 }
 public Object rearElement()
 {
  if(count == 0)
  return -999;
  else
  {
   Object low;
   rear = (rear-1)%size;
   low = q[rear];
   rear = (rear+1)%size;
   return low;
  }
 }
}
public class CircularQueueDemo
{
 public static void main(String args[])
 {
  Object var;
  CircularQueue myQ = new CircularQueue();
  myQ.enqueue(10);
  myQ.enqueue(3);
  var = myQ.rearElement();
  myQ.dequeue();
  myQ.enqueue(6);
  var = mQ.frontElement();
  System.out.println(var+" "+var);
 }
}
```

a) 3 3
b) 3 6
c) 6 6
d) 10 6
View Answer

Answer: a
Explanation: First enqueue 10 and 3 into the queue, followed by a dequeue(removes 10), followed by an enqueue(6), At this point, 3 is at the front end of the queue and 6 at the rear end, hence a call to frontElement() will return 3 which is displayed twice.
Topic 15. Queue using Linked List

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Queue using Linked List”.

1. In linked list implementation of queue, if only front pointer is maintained, which of the following operation take worst case linear time?

a) Insertion
b) Deletion
c) To empty a queue
d) Both Insertion and To empty a queue
View Answer

Answer: d
Explanation: Since front pointer is used for deletion, so worst time for the other two cases.

2. In linked list implementation of a queue, where does a new element be inserted?
a) At the head of link list
b) At the centre position in the link list
c) At the tail of the link list
d) At any position in the linked list
View Answer

Answer: c
Explanation: Since queue follows FIFO so new element inserted at last.

3. In linked list implementation of a queue, front and rear pointers are tracked. Which of these pointers will change during an insertion into a NONEMPTY queue?
a) Only front pointer
b) Only rear pointer
c) Both front and rear pointer
d) No pointer will be changed
View Answer

Answer: b
Explanation: Since queue follows FIFO so new element inserted at last.

4. In linked list implementation of a queue, front and rear pointers are tracked. Which of these pointers will change during an insertion into EMPTY queue?
a) Only front pointer
b) Only rear pointer
c) Both front and rear pointer
d) No pointer will be changed
View Answer

Answer: c
Explanation: Since its the starting of queue, so both values are changed.

5. In case of insertion into a linked queue, a node borrowed from the __________ list is inserted in the queue.
a) AVAIL
b) FRONT
c) REAR
d) NULL
View Answer

Answer: a
Explanation: All the nodes are collected in AVAIL list.

6. In linked list implementation of a queue, from where is the item deleted?
a) At the head of link list
b) At the centre position in the link list
c) At the tail of the link list
d) Node before the tail
View Answer

Answer: a
Explanation: Since queue follows FIFO so new element deleted from first.

7. In linked list implementation of a queue, the important condition for a queue to be empty is?
a) FRONT is null
b) REAR is null
c) LINK is empty
d) FRONT==REAR-1
View Answer

Answer: a
Explanation: Because front represents the deleted nodes.

8. The essential condition which is checked before insertion in a linked queue is?
a) Underflow
b) Overflow
c) Front value
d) Rear value
View Answer

Answer: b
Explanation: To check whether there is space in the queue or not.

9. The essential condition which is checked before deletion in a linked queue is?
a) Underflow
b) Overflow
c) Front value
d) Rear value
View Answer

Answer: a
Explanation: To check whether there is element in the list or not.

10. Which of the following is true about linked list implementation of queue?
a) In push operation, if new nodes are inserted at the beginning of linked list, then in pop operation, nodes must be removed from end
b) In push operation, if new nodes are inserted at the beginning, then in pop operation, nodes must be removed from the beginning
c) In push operation, if new nodes are inserted at the end, then in pop operation, nodes must be removed from end
d) In push operation, if new nodes are inserted at the end, then in pop operation, nodes must be removed from beginning
View Answer

Answer: a
Explanation: It can be done by both the methods.
Topic 16. Priority Queue

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Priority Queue”.

1. With what data structure can a priority queue be implemented?
a) Array
b) List
c) Heap
d) Tree
View Answer

Answer: c
Explanation: Priority queue can be implemented using an array, a list, a binary search tree or a heap, although the most efficient one being the heap.

2. Which of the following is not an application of priority queue?
a) Huffman codes
b) Interrupt handling in operating system
c) Undo operation in text editors
d) Bayesian spam filter
View Answer

Answer: c
Explanation: Undo operation is achieved using a stack.

3. Select the appropriate code that inserts elements into the list based on the given key value.
(head and trail are dummy nodes to mark the end and beginning of the list, they do not contain any priority or element)
a)

```
public void insert_key(int key,Object item)
{
 if(key<0)
 {
  Systerm.our.println("invalid");
  System.exit(0);
 }
 else
 {
  Node temp = new Node(key,item,null);
  if(count == 0)
  {
   head.setNext(temp);
   temp.setNext(trail);
  }
  else
  {
   Node dup = head.getNext();
   Node cur = head;
   while((key>dup.getKey()) && (dup!=trail))
   {
    dup = dup.getNext();
    cur = cur.getNext();
   }
   cur.setNext(temp);
   temp.setNext(dup);
  }
  count++;
 }
}
```

b)

```
public void insert_key(int key,Object item)
{
 if(key<0)
 {
  Systerm.our.println("invalid");
  System.exit(0);
 }
 else
 {
  Node temp = new Node(key,item,null);
  if(count == 0)
  {
   head.setNext(temp);
   temp.setNext(trail);
  }
  else
  {
   Node dup = head.getNext();
   Node cur = dup;
   while((key>dup.getKey()) && (dup!=trail))
   {
    dup = dup.getNext();
    cur = cur.getNext();
   }
   cur.setNext(temp);
   temp.setNext(dup);
  }
  count++;
 }
}
```

c)

```
public void insert_key(int key,Object item)
{
 if(key<0)
 {
  Systerm.our.println("invalid");
  System.exit(0);
 }
 else
 {
  Node temp = new Node(key,item,null);
  if(count == 0)
  {
   head.setNext(temp);
   temp.setNext(trail);
  }
  else
  {
   Node dup = head.getNext();
   Node cur = head;
   while((key>dup.getKey()) && (dup!=trail))
   {
    dup = dup.getNext();
    cur = cur.getNext();
   }
   cur.setNext(dup);
   temp.setNext(cur);
  }
  count++;
 }
}
```

d)

```
public void insert_key(int key,Object item)
{
 if(key<0)
 {
  Systerm.our.println("invalid");
  System.exit(0);
 }
 else
 {
  Node temp = new Node(key,item,null);
  if(count == 0)
  {
   head.setNext(temp);
   temp.setNext(trail);
  }
  else
  {
   Node dup = head.getNext();
   Node cur = head;
   while((key>dup.getKey()) && (dup!=trail))
   {
    dup = cur
    cur = cur.getNext();
   }
   cur.setNext(dup);
   temp.setNext(cur);
  }
  count++;
 }
}
```

View Answer
Answer: a
Explanation: Have two temporary pointers 'dup' and 'cur' with 'cur' trailing behind 'dup'. Traverse through the list until the given key is greater than some element with a lesser key, insert the new node 'temp' in that position.

4. What is the time complexity to insert a node based on key in a priority queue?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n^2)
View Answer

Answer: c
Explanation: In the worst case, you might have to traverse the entire list.

5. What is the functionality of the following piece of code?

```
public Object delete_key()
{
 if(count == 0)
 {
  System.out.println("Q is empty");
  System.exit(0);
 }
 else
 {
  Node cur = head.getNext();
  Node dup = cur.getNext();
  Object e = cur.getEle();
  head.setNext(dup);
  count--;
  return e;
 }
}
```

a) Delete the second element in the list
b) Return but not delete the second element in the list
c) Delete the first element in the list
d) Return but not delete the first element in the list
View Answer

Answer: c
Explanation: A pointer is made to point at the first element in the list and one more to point to the second element, pointer manipulations are done such that the first element is no longer being pointed by any other pointer, its value is returned.

6. What is not a disadvantage of priority scheduling in operating systems?
a) A low priority process might have to wait indefinitely for the CPU

b) If the system crashes, the low priority systems may be lost permanently
c) Interrupt handling
d) Indefinite blocking
View Answer

Answer: c
Explanation: The lower priority process should wait until the CPU completes the processing higher priority process. Interrupt handling is an advantage as interrupts should be given more priority than tasks at hand so that interrupt can be serviced to produce desired results.

7. Which of the following is not an advantage of a priority queue?
a) Easy to implement
b) Processes with different priority can be efficiently handled
c) Applications with differing requirements
d) Easy to delete elements in any case
View Answer

Answer: d
Explanation: In worst case, the entire queue has to be searched for the element having the highest priority. This will take more time than usual. So deletion of elements is not an advantage.

8. What is the time complexity to insert a node based on position in a priority queue?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n^2)
View Answer

Answer: c
Explanation: In the worst case, you might have to traverse the entire list.
Topic 17. Double Ended Queue (Dequeue)

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Double Ended Queue (Dequeue)”.

1. What is a dequeue?
a) A queue with insert/delete defined for both front and rear ends of the queue
b) A queue implemented with a doubly linked list
c) A queue implemented with both singly and doubly linked lists
d) A queue with insert/delete defined for front side of the queue
View Answer

Answer: a
Explanation: A dequeue or a double ended queue is a queue with insert/delete defined for both front and rear ends of the queue.

2. Select the function which performs insertion at the front end of the dequeue?
a)

```
public void function(Object item)
{
 Node temp = new Node(item,null);
 if(isEmpty())
 {
  temp.setNext(trail);
  head.setNext(temp);
 }
 else
 {
  Node cur = head.getNext();
  temp.setNext(cur);
  head.setNext(temp);
 }
 size++;
}
```

b)

```
public void function(Object item)
{
 Node temp = new Node(item,null);
 if(isEmpty())
 {
  temp.setNext(trail);
  head.setNext(trail);
 }
 else
 {
  Node cur = head.getNext();
  temp.setNext(cur);
  head.setNext(temp);
 }
 size++;
}
```

c)

```
public void function(Object item)
{
 Node temp = new Node(item,null);
 if(isEmpty())
 {
  Node cur = head.getNext();
  temp.setNext(cur);
  head.setNext(temp);
 }
 else
 {
  temp.setNext(trail);
  head.setNext(temp);
 }
 size++;
}
```

d)

```
public void function(Object item)
{
 Node temp = new Node(item,null);
 if(isEmpty())
 {
  Node cur = head.getNext();
  temp.setNext(cur);
  cur.setNext(temp);
 }
 else
 {
  head.setNext(trail);
  trail.setNext(temp);
 }
 size++;
}
```

View Answer
Answer: a
Explanation: Create a new node, if the current list is empty, the 'head' points to this node and this new node points to 'trail'. Otherwise, 'head' points to the new node and this in turn points to the current first element(head.getNext()).

3. What is the functionality of the following piece of code?

```
public void function(Object item)
{
 Node temp=new Node(item,trail);
 if(isEmpty())
 {
  head.setNext(temp);
  temp.setNext(trail);
 }
 else
 {
  Node cur=head.getNext();
  while(cur.getNext()!=trail)
  {
   cur=cur.getNext();
  }
  cur.setNext(temp);
 }
 size++;
}
```

a) Insert at the front end of the dequeue
b) Insert at the rear end of the dequeue
c) Fetch the element at the rear end of the dequeue
d) Fetch the element at the front end of the dequeue
View Answer

Answer: b

Explanation: If the list is empty, this new node will point to ‘trail’ and will be pointed at by ‘head’. Otherwise, traverse till the end of the list and insert the new node there.

4. What are the applications of dequeue?
a) A-Steal job scheduling algorithm
b) Can be used as both stack and queue
c) To find the maximum of all sub arrays of size k
d) All of the mentioned
View Answer

Answer: d
Explanation: All of the mentioned can be implemented with a dequeue.

5. Which of the following can be used to delete an element from the front end of the queue?
a)

```
public Object deleteFront() throws emptyDEQException
{
 if(isEmpty())
  throw new emptyDEQException("Empty");
 else
 {
  Node temp = head.getNext();
  Node cur = temp;
  Object e = temp.getEle();
  head.setNext(cur);
  size--;
  return e;
 }
}
```

b)

```
public Object deleteFront() throws emptyDEQException
{
 if(isEmpty())
  throw new emptyDEQException("Empty");
 else
 {
  Node temp = head.getNext();
  Node cur = temp.getNext();
  Object e = temp.getEle();
  head.setNext(cur);
  size--;
  return e;
 }
}
```

c)

```
public Object deleteFront() throws emptyDEQException
{
 if(isEmpty())
  throw new emptyDEQException("Empty");
 else
 {
  Node temp = head.getNext();
  Node cur = temp.getNext();
  Object e = temp.getEle();
  head.setNext(temp);
  size--;
  return e;
 }
}
```

d)

```
public Object deleteFront() throws emptyDEQException
{
 if(isEmpty())
  throw new emptyDEQException("Empty");
 else
 {
  Node temp = head.getNext();
  Node cur = temp.getNext();
  Object e = temp.getEle();
  temp.setNext(cur);
  size--;
  return e;
 }
}
```

View Answer
Answer: b
Explanation: Have two pointers, one(temp) pointing to the first element and the other(cur) pointing to the second element. Make the ‘head’ point to the second element, this removes all reference for ‘temp’.

6. Which of the following can be used to delete an element from the rear end of the queue?
a)

```
public Object deleteRear() throws emptyDEQException
{
 if(isEmpty())
  throw new emptyDEQException("Empty");
 else
 {
  Node temp = head.getNext();
  Node cur = temp;
  while(temp.getNext() != trail)
  {
   temp = temp.getNext();
   cur = cur.getNext();
  }
  Object e = temp.getEle();
  cur.setNext(trail);
  size--;
  return e;
 }
}
```

b)

```
public Object deleteRear() throws emptyDEQException
{
 if(isEmpty())
  throw new emptyDEQException("Empty");
 else
 {
  Node temp = head.getNext();
  Node cur = head;
  while(temp != trail)
  {
   temp = temp.getNext();
   cur = cur.getNext();
  }
  Object e = temp.getEle();
  cur.setNext(trail);
  size--;
  return e;
 }
}
```

c)

```
public Object deleteRear() throws emptyDEQException
{
 if(isEmpty())
 throw new emptyDEQException("Empty");
 else
 {
  Node temp = head.getNext();
  Node cur = head;
  while(temp.getNext()!=trail)
  {
   temp = temp.getNext();
   cur = cur.getNext();
  }
  Object e = temp.getEle();
  cur.setNext(trail);
  size--;
  return e;
 }
}
```

d)

```
public Object deleteRear() throws emptyDEQException
{
 if(isEmpty())
 throw new emptyDEQException("Empty");
 else
 {
  Node temp = head.getNext();
  Node cur = head;
  while(temp.getNext()!=trail)
  {
   temp = temp.getNext();
   cur = cur.getNext();
  }
  Object e = temp.getEle();
  temp.setNext(trail);
  size--;
  return e;
 }
}
```

View Answer
Answer: c
Explanation: Traverse till the end of the list with a pointer ‘temp’ and another ‘cur’ which is trailing behind temp, make ‘cur’ point to trail, this removes all reference for ‘temp’.

7. What is the time complexity of deleting from the rear end of the dequeue implemented with a singly linked list?
a) O(nlogn)
b) O(logn)
c) O(n)
d) O(n^2)
View Answer
Answer: c
Explanation: Since a singly linked list is used, first you have to traverse till the end, so the complexity is O(n).

8. After performing these set of operations, what does the final list look contain?

```
InsertFront(10);
InsertFront(20);
InsertRear(30);
DeleteFront();
InsertRear(40);
InsertRear(10);
DeleteRear();
InsertRear(15);
display();
```

a) 10 30 10 15
b) 20 30 40 15
c) 20 30 40 10
d) 10 30 40 15
View Answer

Answer: d
Explanation: A careful tracing of the given operation yields the result.
10
20 10
20 10 30
10 30
10 30 40
10 30 40 10
10 30 40
10 30 40 15
Topic 18. Queue using Stacks

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Queue using Stacks”.

1. A Double-ended queue supports operations such as adding and removing items from both the sides of the queue. They support four operations like addFront(adding item to top of the queue), addRear(adding item to the bottom of the queue), removeFront(removing item from the top of the queue) and removeRear(removing item from the bottom of the queue). You are given only stacks to implement this data structure. You can implement only push and pop operations. What are the total number of stacks required for this operation?(you can reuse the stack)
a) 1
b) 2
c) 3
d) 4
View Answer

Answer: b
Explanation: The addFront and removeFront operations can be performed using one stack itself as push and pop are supported (adding and removing element from top of the stack) but to perform addRear and removeRear you need to pop each element from the current stack and push it into another stack, push or pop the element as per the asked operation from this stack and in the end pop elements from this stack to the first stack.

2. You are asked to perform a queue operation using a stack. Assume the size of the stack is some value ‘n’ and there are ‘m’ number of variables in this stack. The time complexity of performing deQueue operation is (Using only stack operations like push and pop)(Tightly bound).
a) O(m)
b) O(n)
c) O(m*n)
d) Data is insufficient
View Answer

Answer: a
Explanation: To perform deQueue operation you need to pop each element from the first stack and push it into the second stack. In this case you need to pop ‘m’ times and need to perform push operations also ‘m’ times. Then you pop the first element from this second stack (constant time) and pass all the elements to the first stack (as done in the beginning)(‘m-1’ times). Therfore the time complexity is O(m).

3. Consider you have an array of some random size. You need to perform dequeue operation. You can perform it using stack operation (push and pop) or using queue operations itself (enQueue and Dequeue). The output is guaranteed to be same. Find some differences?
a) They will have different time complexities
b) The memory used will not be different
c) There are chances that output might be different
d) No differences
View Answer

Answer: a
Explanation: To perform operations such as Dequeue using stack operation you need to empty all the elements from the current stack and push it into the next stack, resulting in a O(number of elements) complexity whereas the time complexity of dequeue operation itself is O(1). And there is a need of a extra stack. Therefore more memory is needed.

4. Consider you have a stack whose elements in it are as follows.
5 4 3 2 << top
Where the top element is 2.
You need to get the following stack
6 5 4 3 2 << top
The operations that needed to be performed are (You can perform only push and pop):
a) Push(pop()), push(6), push(pop())
b) Push(pop()), push(6)
c) Push(pop()), push(pop()), push(6)
d) Push(6)
View Answer

Answer: a
Explanation: By performing push(pop()) on all elements on the current stack to the next stack you get 2 3 4 5 << top.Push(6) and perform push(pop()) you’ll get back 6 5 4 3 2 << top. You have actually performed enQueue operation using push and pop.

5. A double-ended queue supports operations like adding and removing items from both the sides of the queue. They support four operations like addFront(adding item to top of the queue), addRear(adding item to the bottom of the queue), removeFront(removing item from the top of the queue) and removeRear(removing item from the bottom of the queue). You are given only stacks to implement this data structure. You can implement only push and pop operations. What’s the time complexity of performing addFront and addRear? (Assume ‘m’ to be the size of the stack and ‘n’ to be the number of elements)
a) O(m) and O(n)
b) O(1) and O(n)
c) O(n) and O(1)
d) O(n) and O(m)
View Answer

Answer: b
Explanation: addFront is just a normal push operation. Push operation is of O(1). Whereas addRear is of O(n) as it requires two push(pop()) operations of all elements of a stack.

6. Why is implementation of stack operations on queues not feasible for a large dataset (Asssume the number of elements in the stack to be n)?
a) Because of its time complexity O(n)
b) Because of its time complexity O(log(n))
c) Extra memory is not required
d) There are no problems
View Answer

Answer: a
Explanation: To perform Queue operations such as enQueue and deQueue there is a need of emptying all the elements of a current stack and pushing elements into the next stack and vice versa. Therfore it has a time complexity of O(n) and the need of extra stack as well, may not be feasible for a large dataset.

7. Consider yourself to be in a planet where the computational power of chips to be slow. You have an array of size 10.You want to perform enqueue some element into this array. But you can perform only push and pop operations .Push and pop operation both take 1 sec respectively. The total time required to perform enQueue operation is?
a) 20
b) 40
c) 42
d) 43
View Answer

Answer: d
Explanation: First you have to empty all the elements of the current stack into the temporary stack, push the required element and empty the elements of the temporary stack into the original stack. Therfore taking 10+10+1+11+11= 43 seconds.

8. You have two jars, one jar which has 10 rings and the other has none. They are placed one above the other. You want to remove the last ring in the jar. And the second jar is weak and cannot be used to store rings for a long time.
a) Empty the first jar by removing it one by one from the first jar and placing it into the second jar
b) Empty the first jar by removing it one by one from the first jar and placing it into the second jar and empty the second jar by placing all the rings into the first jar one by one
c) There exists no possible way to do this
d) Break the jar and remove the last one
View Answer

Answer: b
Explanation: This is similar to performing dequeue operation using push and pop only. Elements in the first jar are taken out and placed in the second jar. After removing the last element from the first jar, remove all the elements in the second jar and place them in the first jar.

9. Given only a single array of size 10 and no other memory is available. Which of the following operation is not feasible to implement (Given only push and pop operation)?
a) Push
b) Pop
c) Enqueue
d) Returntop
View Answer

Answer: c
Explanation: To perform Enqueue using just push and pop operations, there is a need of another array of same size. But as there is no extra available memeory, the given operation is not feasible.

10. Given an array of size n, let’s assume an element is ‘touched’ if and only if some operation is performed on it(for example, for performing a pop operation the top element is ‘touched’). Now you need to perform Dequeue operation. Each element in the array is touched atleast?
a) Once
b) Twice
c) Thrice
d) Four times
View Answer

Answer: d
Explanation: First each element from the first stack is popped, then pushed into the second stack, dequeue operation is done on the top of the stack and later the each element of second stack is popped then pushed into the first stack. Therfore each element is touched four times.
Topic 19. Stack using Queues

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Stack using Queues”.

1. To implement a stack using queue(with only enqueue and dequeue operations), how many queues will you need?
a) 1
b) 2
c) 3
d) 4
View Answer

Answer: b

Explanation: Either the push or the pop has to be a costly operation, and the costlier operation requires two queues.

2. Making the push operation costly, select the code snippet which implements the same.(let q1 and q2 be two queues)
a)

```
public void push(int x)
{
        if(empty())
        {
            q1.offer(x);
        }
        else{
                if(q1.size()>0)
                {
                    q2.offer(x);
                    int size = q1.size();
                    while(size>0)
                    {
                        q2.offer(q1.poll());
                        size--;
                    }
                }
            else if(q2.size()>0)
            {
                q1.offer(x);
                int size = q2.size();
                while(size>0)
                {
                    q1.offer(q2.poll());
                    size--;
                }
            }
        }
    }
```

b)

```
public void push(int x)
{
        if(empty())
        {
            q1.offer(x);
        }
        else
        {
            if(q1.size()>0)
            {
                q1.offer(x);
                int size = q1.size();
                while(size>0)
                {
                    q2.offer(q1.poll());
                    size--;
                }
            }
            else if(q2.size()>0)
            {
                q2.offer(x);
                int size = q2.size();
                while(size>0)
                {
                    q1.offer(q2.poll());
                    size--;
                }
            }
        }
}
```

c)

```
public void push(int x)
{
        if(empty())
        {
            q1.offer(x);
        }
        else
        {
            if(q1.size()>0)
            {
                q2.offer(x);
                int size = q1.size();
                while(size>0)
                {
                    q1.offer(q2.poll());
                    size--;
                }
            }
            else if(q2.size()>0)
            {
                q1.offer(x);
                int size = q2.size();
                while(size>0)
                {
                    q2.offer(q1.poll());
                    size--;
                }
            }
        }
}
```

d)

```
public void push(int x)
{
    if(empty())
    {
        q1.offer(x);
    }
    else
    {
        if(q1.size()>0)
        {
            q2.offer(x);
            int size = q1.size();
            while(size>0)
            {
                q2.offer(q2.poll());
                size--;
            }
        }
        else if(q2.size()>0)
        {
            q1.offer(x);
            int size = q2.size();
            while(size>0)
            {
                q2.offer(q1.poll());
                size--;
            }
        }
    }
}
```

View Answer
Answer: a
Explanation: Stack follows LIFO principle, hence a new item added must be the first one to exit, but queue follows FIFO principle, so when a new item is entered into the queue, it will be at the rear end of the queue. If the queue is initially empty, then just add the new element, otherwise add the new element to the second queue and dequeue all the elements from the second queue and enqueue it to the first one, in this way, the new element added will be always in front of the queue. Since two queues are needed to realize this push operation, it is considered to be costlier.

3. Making the push operation costly, select the code snippet which implements the pop operation.
a)

```
public void pop()
{
    if(q1.size()>0)
    {
        q2.poll();
    }
    else if(q2.size()>0)
    {
        q1.poll();
    }
}
```

b)

```
public void pop()
{
    if(q1.size()>0)
    {
        q1.poll();
    }
    else if(q2.size()>0)
    {
        q2.poll();
    }
}
```

c)

```
public void pop()
{
        q1.poll();
 q2.poll();
}
```

d)

```
public void pop()
{
        if(q2.size()>0)
        {
            q1.poll();
        }
        else
        {
            q2.poll();
        }
}
```

View Answer
Answer: b
Explanation: As the push operation is costly, it is evident that the required item is in the front of the queue, so just dequeue the element from the queue.

4. Select the code snippet which returns the top of the stack.
a)

```
public int top()
{
       if(q1.size()>0)
       {
            return q1.poll();
       }
       else if(q2.size()>0)
       {
            return q2.poll();
       }
       return 0;
}
```

b)

```
public int top()
{
       if(q1.size()==0)
       {
            return q1.peek();
       }
       else if(q2.size()==0)
       {
            return q2.peek();
       }
        return 0;
    }
```

c)

```
public int top()
{
       if(q1.size()>0)
       {
            return q1.peek();
       }
       else if(q2.size()>0)
       {
            return q2.peek();
       }
       return 0;
}
```

d)

```
public int top()
{
       if(q1.size()>0)
```

```
      {
           return q2.peek();
      }
      else if(q2.size()>0)
      {
           return q1.peek();
      }
      return 0;
}
```

View Answer
Answer: c
Explanation: Assuming its a push costly implementation, the top of the stack will be in the front end of the queue, note that peek() just returns the front element, while poll() removes the front element from the queue.

5. Select the code snippet which return true if the stack is empty, false otherwise.
a)

```
public boolean empty()
{
     return q2.isEmpty();
}
```

b)

```
public boolean empty()
{
     return q1.isEmpty() || q2.isEmpty();
}
```

c)

```
public boolean empty()
{
     return q1.isEmpty();
}
```

d)

```
public boolean empty()
{
     return q1.isEmpty() & q2.isEmpty();
}
```

View Answer
Answer: b
Explanation: If both the queues are empty, then the stack also is empty.

6. Making the pop operation costly, select the code snippet which implements the same.
a)

```
public int pop()
{
 int res=-999,count=0;
 if(q1.size()>0)
        {
  count = q1.size();
  while(count>0)
   q2.offer(q1.poll());
  res = q1.poll();
 }
 if(q2.size()>0)
        {
  count = q2.size();
  while(count>0)
   q1.offer(q2.poll());
  res = q2.poll();
 }
 return res;
}
```

b)

```
public int pop()
{
 int res=-999,count=0;
 if(q1.size()>0)
        {
  count = q1.size();
  while(count>1)
```

```
   q2.offer(q1.poll());
  res = q2.poll();
 }
 if(q2.size()>0)
        {
  count = q2.size();
  while(count>1)
   q1.offer(q2.poll());
  res = q1.poll();
 }
 return res;
}
```

c)

```
public int pop()
{
 int res=-999,count=0;
 if(q1.size()>0)
        {
  count = q1.size();
  while(count>1)
   q2.offer(q1.poll());
  res = q1.poll();
 }
 if(q2.size()>0)
        {
  count = q2.size();
  while(count>1)
   q1.offer(q2.poll());
  res = q2.poll();
 }
 return res;
}
```

d)

```
public int pop()
{
 int res=-999,count=0;
 if(q1.size()>0)
        {
  count = q2.size();
  while(count>1)
   q2.offer(q1.poll());
  res = q1.poll();
 }
 if(q2.size()>0)
        {
  count = q1.size();
  while(count>1)
   q1.offer(q2.poll());
  res = q2.poll();
 }
 return res;
}
```

View Answer
Answer: c
Explanation: Here the pop operation is costly, hence we need two queues, other than the first element, all the the elements from one queue are dequeued and enqueued to the second queue, hence only one element remains in the first queue which is the item we want, so dequeue it and return the result.

7. What is the functionality of the following piece of code?

```
public void fun(int x)
{
 q1.offer(x);
}
```

a) Perform push() with push as the costlier operation
b) Perform push() with pop as the costlier operation
c) Perform pop() with push as the costlier operation
d) Perform pop() with pop as the costlier operation
View Answer

Answer: b
Explanation: offer() suggests that it is a push operation, but we see that it is performed with only one queue, hence the pop operation is costlier.
Topic 20. Decimal to Binary using Stacks

This set of Data Structure Interview Questions and Answers for Experienced people focuses on “Decimal to Binary using Stacks”.

1. Express -15 as a 6-bit signed binary number.
a) 001111
b) 101111
c) 101110
d) 001110
View Answer

Answer: b
Explanation: The first 4 1s from the right represent the number 15, 2 more bits are padded to make it 6 digits and the leftmost bit is a 1 to represent that it is -15.

2. Which of the following code snippet is used to convert decimal to binary numbers?
a)

```
public void convertBinary(int num)
{
    int bin[] = new int[50];
    int index = 0;
    while(num > 0)
    {
      bin[index++] = num%2;
      num = num/2;
    }
    for(int i = index-1;i >= 0;i--)
    {
      System.out.print(bin[i]);
    }
}
```

b)

```
public void convertBinary(int num)
{
    int bin[] = new int[50];
    int index = 0;
    while(num > 0)
    {
      bin[++index] = num%2;
      num = num/2;
    }
    for(int i = index-1;i >= 0;i--)
    {
      System.out.print(bin[i]);
    }
}
```

c)

```
public void convertBinary(int num)
{
    int bin[] = new int[50];
    int index = 0;
    while(num > 0)
    {
       bin[index++] = num/2;
       num = num%2;
    }
    for(int i = index-1;i >= 0;i--)
    {
       System.out.print(bin[i]);
    }
}
```

d)

```
public void convertBinary(int num)
 {
    int bin[] = new int[50];
    int index = 0;
    while(num > 0)
    {
       bin[++index] = num/2;
       num = num%2;
    }
    for(int i = index-1;i >= 0;i--)
    {
       System.out.print(bin[i]);
    }
  }
```

View Answer
Answer: a
Explanation: Take the modulus by 2 of the number and store in an array while halving the number during each iteration and then display the contents of the array.

3. Which is the predefined method available in Java to convert decimal to binary numbers?
a) toBinaryInteger(int)
b) toBinaryValue(int)
c) toBinaryNumber(int)
d) toBinaryString(int)
View Answer
Answer: d
Explanation: The method toBinaryString() takes an integer argument and is defined in java.lang package. Usage is java.lang.Integer.toBinaryString(int) this returns the string representation of the unsigned integer value.

4. Using stacks, how to obtain the binary representation of the number?
a)

```
public void convertBinary(int num)
{
    Stack<Integer> stack = new Stack<Integer>();
    while (num != 0)
    {
        int digit = num / 2;
        stack.push(digit);
        num = num % 2;
    }
    System.out.print("\nBinary representation is:");
    while (!(stack.isEmpty() ))
    {
        System.out.print(stack.pop());
    }
 }
```

b)

```
public void convertBinary(int num)
{
    Stack<Integer> stack = new Stack<Integer>();
    while (num != 0)
    {
        int digit = num % 2;
        stack.push(digit);
    }
    System.out.print("\nBinary representation is:");
    while (!(stack.isEmpty() ))
    {
        System.out.print(stack.pop());
    }
 }
```

c)

```
public void convertBinary(int num)
{
    Stack<Integer> stack = new Stack<Integer>();
    while (num != 0)
    {
        int digit = num % 2;
        stack.push(digit);
        num = num / 2;
    }
    System.out.print("\nBinary representation is:");
    while (!(stack.isEmpty() ))
    {
        System.out.print(stack.pop());
    }
 }
```

d)

```
public void convertBinary(int num)
{
    Stack<Integer> stack = new Stack<Integer>();
    while (num != 0)
    {
        int digit = num % 2;
        stack.push(digit%2);
        num = num / 2;
    }
    System.out.print("\nBinary representation is:");
    while (!(stack.isEmpty() ))
    {
        System.out.print(stack.pop());
    }
 }
```

View Answer
Answer: c
Explanation: Here instead of adding the digits to an array, you push it into a stack and while printing, pop it from the stack.

5. What is the time complexity for converting decimal to binary numbers?
a) O(1)
b) O(n)
c) O(logn)
d) O(nlogn)
View Answer
Answer: c
Explanation: Since each time you are halving the number, it can be related to that of a binary search algorithm, hence the complexity is O(logn).

6. Write a piece of code which returns true if the string contains balanced parenthesis, false otherwise.
a)

```
public boolean isBalanced(String exp)
{
 int len = exp.length();
 Stack<Integer> stk = new Stack<Integer>();
 for(int i = 0; i < len; i++)
        {
  char ch = exp.charAt(i);
                if (ch == '(')
                stk.push(i);
                else if (ch == ')')
                {
   if(stk.peek() == null)
                        {
    return false;
   }
   stk.pop();
  }
 }
 return true;
}
```

b)

```
public boolean isBalanced(String exp)
{
 int len = exp.length();
 Stack<Integer> stk = new Stack<Integer>();
 for(int i = 0; i < len; i++)
            {
  char ch = exp.charAt(i);
                if (ch == '(')
                stk.push(i);
                else if (ch == ')')
                {
   if(stk.peek() != null)
                        {
    return true;
   }
   stk.pop();
  }
 }
 return false;
  }
```

c)

```
public boolean isBalanced(String exp)
{
 int len = exp.length();
 Stack<Integer> stk = new Stack<Integer>();
 for(int i = 0; i < len; i++)
        {
        char ch = exp.charAt(i);
               if (ch == ')')
               stk.push(i);
               else if (ch == '(')
               {
```

```
   if(stk.peek() == null)
                        {
    return false;
   }
   stk.pop();
  }
 }
 return true;
}
```

d)

```
public boolean isBalanced(String exp)
{
 int len = exp.length();
 Stack<Integer> stk = new Stack<Integer>();
 for(int i = 0; i < len; i++)
        {
  char ch = exp.charAt(i);
                if (ch == '(')
                stk.push(i);
                else if (ch == ')')
                {
   if(stk.peek() != null)
                        {
    return false;
   }
   stk.pop();
  }
 }
 return true;
  }
```

View Answer
Answer: a
Explanation: Whenever a '(' is encountered, push it into the stack, and when a ')' is encountered check the top of the stack to see if there is a matching '(', if not return false, continue this till the entire string is processed and then return true.

7. What is the time complexity of the following code?

```
public boolean isBalanced(String exp)
{
 int len = exp.length();
 Stack<Integer> stk = new Stack<Integer>();
 for(int i = 0; i < len; i++)
        {
  char ch = exp.charAt(i);
                if (ch == '(')
                stk.push(i);
                else if (ch == ')')
                {
   if(stk.peek() == null)
                        {
    return false;
   }
   stk.pop();
  }
 }
 return true;
}
```

a) O(logn)
b) O(n)
c) O(1)
d) O(nlogn)
View Answer

Answer: b
Explanation: All the characters in the string have to be processed, hence the complexity is O(n).

8. Which of the following program prints the index of every matching parenthesis?
a)

```
public void dispIndex(String exp)
{
    Stack<Integer> stk = new Stack<Integer>();
    for (int i = 0; i < len; i++)
    {
        char ch = exp.charAt(i);
        if (ch == '(')
        stk.push(i);
        else if (ch == ')')
        {
          try
          {
            int p = stk.pop() + 1;
            System.out.println("')' at index "+(i+1)+" matched with ')' at index "+p);
          }
          catch(Exception e)
          {
             System.out.println("')' at index "+(i+1)+" is unmatched");
          }
        }
    }
    while (!stk.isEmpty() )
    System.out.println("'(' at index "+(stk.pop() +1)+" is unmatched");
}
```

b)

```
public void dispIndex(String exp)
{
    Stack<Integer> stk = new Stack<Integer>();
    for (int i = 0; i < len; i++)
    {
        char ch = exp.charAt(i);
        if (ch == '(')
        stk.push(i);
        else if (ch == ')')
        {
           try
           {
             int p = stk.pop() + 1;
             System.out.println("')' at index "+(i)+" matched with ')' at index "+p);
           }
           catch(Exception e)
           {
              System.out.println("')' at index "+(i)+" is unmatched");
           }
        }
    }
    while (!stk.isEmpty() )
    System.out.println("'(' at index "+(stk.pop() +1)+" is unmatched");
}
```

c)

```
public void dispIndex(String exp)
{
    Stack<Integer> stk = new Stack<Integer>();
    for (int i = 0; i < len; i++)
    {
        char ch = exp.charAt(i);
        if (ch == ')')
        stk.push(i);
        else if (ch == '(')
        {
          try
          {
            int p = stk.pop() +1;
            System.out.println("')' at index "+(i+1)+" matched with ')' at index "+p);
          }
          catch(Exception e)
          {
             System.out.println("')' at index "+(i+1)+" is unmatched");
          }
        }
    }
    while (!stk.isEmpty() )
    System.out.println("'(' at index "+(stk.pop() +1)+" is unmatched");
}
```

d)

```
public void dispIndex(String exp)
{
    Stack<Integer> stk = new Stack<Integer>();
    for (int i = 0; i < len; i++)
    {
        char ch = exp.charAt(i);
        if (ch == ')')
        stk.push(i);
        else if (ch == '(')
        {
          try
          {
            int p = stk.pop();
            System.out.println("')' at index "+(i+1)+" matched with ')' at index "+p);
          }
          catch(Exception e)
          {
            System.out.println("')' at index "+(i+1)+" is unmatched");
          }
        }
    }
    while (!stk.isEmpty() )
    System.out.println("'(' at index "+(stk.pop() +1)+" is unmatched");
}
```

View Answer
Answer: a
Explanation: Whenever a '(' is encountered, push the index of that character into the stack, so that whenever a corresponding ')' is encountered, you can pop and print it.
Topic 21. Evaluation of an Infix Expression (Not Parenthesized)

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on "Evaluation of an Infix Expression (Not Parenthesized)".

1. How many stacks are required for applying evaluation of infix expression algorithm?
a) one
b) two
c) three
d) four
View Answer

Answer: b
Explanation: Two stacks are required for evaluation of infix expression – one for operands and one for operators.

2. How many passes does the evaluation of infix expression algorithm makes through the input?
a) One
b) Two
c) Three
d) Four
View Answer

Answer: a
Explanation: Evaluation of infix expression algorithm is linear and makes only one pass through the input.

3. Identify the infix expression from the list of options given below.
a) a/b+(c-d)
b) abc*+d+ab+cd+*ce-f-
c) ab-c-
d) +ab
View Answer

Answer: a
Explanation: a/b+(c-d) is an infix expression since the operators are placed in between the operands.

4. Which of the following statement is incorrect with respect to evaluation of infix expression algorithm?
a) Operand is pushed on to the stack
b) If the precedence of operator is higher, pop two operands and evaluate
c) If the precedence of operator is lower, pop two operands and evaluate
d) The result is pushed on to the operand stack
View Answer

Answer: b
Explanation: If the precedence of the operator is higher than the stack operator, then it is pushed on to the stack operator.

5. Evaluate the following statement using infix evaluation algorithm and choose the correct answer. 1+2*3-2
a) 3
b) 6
c) 5
d) 4
View Answer

Answer: c
Explanation: According to precedence of operators, * is evaluated first. + and – have equal priorities. Hence, 1+6-2= 5.

6. Evaluation of infix expression is done based on precedence of operators.
a) True

b) False
View Answer

Answer: a
Explanation: During evaluation of infix expression, the operators with higher precedence are evaluated first, followed by operators with lower precedence.

7. Of the following choices, which operator has the lowest precedence?
a) ^
b) +
c) /
d) #
View Answer

Answer: d
Explanation: The operator with the lowest precedence is #, preceded by +, / and then ^.

8. The system throws an error if parentheses are encountered in an infix expression evaluation algorithm.
a) True
b) False
View Answer

Answer: b
Explanation: The algorithm holds good for infix expression with parentheses. The system does not throw error.

9. Evaluate the following and choose the correct answer.
a/b+c*d where a=4, b=2, c=2, d=1.
a) 1
b) 4
c) 5
d) 2
View Answer

Answer: b
Explanation: * and / have higher priority. Hence, they are evaluated first. Then, + is evaluated. Hence, 2+2=4.

10. Evaluate the following statement using infix evaluation algorithm and choose the correct answer. 4*2+3-5/5
a) 10
b) 11
c) 16
d) 12
View Answer

Answer: a
Explanation: 4*2 and 5/5 are evaluated first and then, 8+3-1 is evaluated and the result is obtained as 10.

11. Using the evaluation of infix expression, evaluate a^b+c and choose the correct answer. (a=2, b=2, c=2)
a) 12
b) 8
c) 10
d) 6
View Answer

Answer: d
Explanation: ^ has the highest precedence. Hence, 2^2 is evaluated and then 4+2 gives 6.

12. Evaluate the following infix expression using algorithm and choose the correct answer. a+b*c-d/e^f where a=1, b=2, c=3, d=4, e=2, f=2.
a) 6
b) 8
c) 9
d) 7
View Answer

Answer: a
Explanation: ^ has the highest order of precedence. Hence, 2^2 is evaluated first, and then, 2*3 and 4/4 are evaluated. Therefore, 1+6-1=6.

13. From the given expression tree, identify the infix expression, evaluate it and choose the correct result.

a) 5

b) 10
c) 12
d) 16
View Answer

Answer: c
Explanation: From the given expression tree, the result of the infix expression is evaluated to be 12.
Topic 22. Evaluation of a Prefix Expression

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Evaluation of a Prefix Expression”.

1. How many stacks are required for evaluation of prefix expression?
a) one
b) two
c) three
d) four
View Answer

Answer: b
Explanation: 2 stacks are required for evaluation of prefix expression, one for integers and one for characters.

2. While evaluating a prefix expression, the string is read from?
a) left to right
b) right to left
c) center to right
d) center to left to right
View Answer

Answer: b
Explanation: The string is read from right to left because a prefix string has operands to its right side.

3. The associativity of an exponentiation operator ^ is right side.
a) True
b) False
View Answer

Answer: a
Explanation: The associativity of ^ is right side while the rest of the operators like +,-,*,/ has its associativity to its left.

4. How many types of input characters are accepted by this algorithm?
a) one
b) two
c) three
d) four
View Answer

Answer: c
Explanation: Three kinds of input are accepted by this algorithm- numbers, operators and new line characters.

5. What determines the order of evaluation of a prefix expression?
a) precedence and associativity
b) precedence only
c) associativity only
d) depends on the parser
View Answer

Answer: a
Explanation: Precedence is a very important factor in determining the order of evaluation. If two operators have the same precedence, associativity comes into action.

6. Find the output of the following prefix expression.

```
*+2-2 1/-4 2+-5 3 1
```

a) 2
b) 12
c) 10
d) 4
View Answer

Answer: a
Explanation: The given prefix expression is evaluated using two stacks and the value is given by (2+2-1)*(4-2)/(5-3+1)= 2.

7. An error is thrown if the character ‘\n’ is pushed in to the character stack.
a) true
b) false
View Answer

Answer: b
Explanation: The input character ‘\n’ is accepted as a character by the evaluation of prefix expression algorithm.

8. Using the evaluation of prefix algorithm, evaluate +-9 2 7.
a) 10
b) 4
c) 17
d) 14
View Answer

Answer: d
Explanation: Using the evaluation of prefix algorithm, +-9 2 7 is evaluated as 9-2+7=14.

9. If -*+abcd = 11, find a, b, c, d using evaluation of prefix algorithm.
a) a=2, b=3, c=5, d=4
b) a=1, b=2, c=5, d=4
c) a=5, b=4, c=7,d=5
d) a=1, b=2, c=3, d=4
View Answer

Answer: b
Explanation: The given prefix expression is evaluated as ((1+2)*5)-4 = 11 while a=1, b=2, c=5, d=4.

10. In the given C snippet, find the statement number that has error.

```
//C code to push an element into a stack
1. void push( struct stack *s, int x)
2. {
3.     if(s->top==MAX-1)
4.     {
5.         printf(“stack overflow”);
6.     }
7.     else
8.     {
9.         s->items[++s->top]=x;
10.         s++;
11.    }
12. }
```

a) 1
b) 9
c) 10
d) 11
View Answer

Answer: c
Explanation: If the stack is not full then we are correctly incrementing the top of the stack by doing “++s->top” and storing the value of x in it. However, in the next statement “s++”, we are un-necessarily incrementing the stack base pointer which will lead to memory corruption during the next push() operation.

Topic 23. Evaluation of a Postfix Expression

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Evaluation of a Postfix Expression”.

1. What is the other name for a postfix expression?
a) Normal polish Notation
b) Reverse polish Notation
c) Warsaw notation
d) Infix notation
View Answer

Answer: b
Explanation: Reverse polish Notation is the other name for a postfix expression whereas Polish Notation, Warsaw notation are the other names for a prefix expression.

2. Which of the following is an example for a postfix expression?
a) a*b(c+d)
b) abc*+de-+
c) +ab
d) a+b-c
View Answer

Answer: b
Explanation: abc*+de-+ is a postfix expression. +ab is a prefix expression and others are infix expressions.

3. Reverse Polish Notation is the reverse of a Polish Notation.
a) True
b) False
View Answer

Answer: b
Explanation: Reverse Polish Notation is not the reverse of a polish notation. Though both NPN and RPN read the expression from left to right, they follow different strategies.

4. What is the time complexity of evaluation of postfix expression algorithm?
a) O (N)
b) O (N log N)
c) O (N^2)
d) O (M log N)
View Answer

Answer: a
Explanation: The time complexity of evaluation of infix, prefix and postfix expressions is O (N).

5. In Postfix expressions, the operators come after the operands.
a) True
b) False
View Answer

Answer: a
Explanation: In postfix expressions, the operators follow operands. In prefix expressions, the operands follow operators.

6. Which of these operators have the highest order of precedence?
a) '(' and ')'
b) '*' and '/'
c) '~' and '^'
d) '+' and '-'
View Answer

Answer: c
Explanation: The highest order of precedence is ~ and ^ followed by '*' ,' /', '+' ,'-' and then braces '(' ')'.

7. Which of the following is not an application of stack?
a) evaluation of postfix expression
b) conversion of infix to postfix expression
c) balancing symbols
d) line at ticket counter
View Answer

Answer: d
Explanation: Line at ticket counter is an application of queue whereas conversion of infix to postfix expression, balancing symbols, line at ticket counter are stack applications.

8. While evaluating a postfix expression, when an operator is encountered, what is the correct operation to be performed?
a) push it directly on to the stack
b) pop 2 operands, evaluate them and push the result on to the stack
c) pop the entire stack
d) ignore the operator
View Answer

Answer: b
Explanation: When an operator is encountered, the first two operands are popped from the stack, they are evaluated and the result is pushed into the stack.

9. Which of the following statement is incorrect?

a) Postfix operators use value to their right
b) Postfix operators use value to their left
c) Prefix operators use value to their right
d) In postfix expression, operands are followed by operators
View Answer

Answer: a
Explanation: All prefix operators use values to their right and all postfix operators use values to their left.

10. What is the result of the given postfix expression? abc*+ where a=1, b=2, c=3.
a) 4
b) 5
c) 6
d) 7
View Answer

Answer: d
Explanation: The infix expression is a+b*c. Evaluating it, we get 1+2*3=7.

11. What is the result of the following postfix expression?
ab*cd*+ where a=2,b=2,c=3,d=4.
a) 16
b) 12
c) 14
d) 10
View Answer

Answer: a
Explanation: The infix expression is a*b+c*d. Evaluating it, we get, 2*2+3*4=16.

12. Consider the stack
| 5 |
| 4 |
| 3 |
| 2 |.
At this point, '*' is encountered. What has to be done?
a) 5*4=20 is pushed into the stack
b) * is pushed into the stack
c) 2*3=6 is pushed into the stack
d) * is ignored
View Answer

Answer: a
Explanation: When an operator is encountered, the first two operands of the stack are popped, evaluated and the result is pushed into the stack.

13. Evaluate the postfix expression ab + cd/- where a=5, b=4, c=9, d=3.
a) 23
b) 15
c) 6
d) 10
View Answer

Answer: c
Explanation: The infix expression is (a+b)-c/d. Evaluating it, (5+4)-9/3 gives 6.

14. Evaluate and write the result for the following postfix expression
abc*+de*f+g*+ where a=1, b=2, c=3, d=4, e=5, f=6, g=2.
a) 61
b) 59
c) 60
d) 55
View Answer

Answer: b
Explanation: The infix expression is a+b*c+(d*e+f)*g. Evaluating it, 1+2*3+(4*5+6)*2 gives 59.

15. For the given expression tree, write the correct postfix expression.

a) abc*+

b) abc+*
c) ab+c*
d) a+bc*
View Answer

Answer: a
Explanation: Evaluating the given expression tree gives the infix expression a+b*c. Converting it to postfix, we get, abc*+.
Topic 24. Infix to Prefix Conversion

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Infix to Prefix Conversion”.

1. What data structure is used when converting an infix notation to prefix notation?
a) Stack
b) Queue
c) B-Trees
d) Linked-list
View Answer

Answer: a
Explanation: First you reverse the given equation and carry out the algorithm of infix to postfix expression. Here, the data structure used is stacks.

2. What would be the Prefix notation for the given equation?

```
A+(B*C)
```

a) +A*CB
b) *B+AC
c) +A*BC
d) *A+CB
View Answer

Answer: c
Explanation: Reverse the equation or scan the equation from right to left. Apply the infix-postfix algorithm. The equation inside the bracket evaluates to CB* and outside the bracket evaluates to A+ therefore getting CB*A+. Reversing this and we get +A*BC.

3. What would be the Prefix notation for the given equation?

```
(A*B)+(C*D)
```

a) +*AB*CD
b) *+AB*CD
c) **AB+CD
d) +*BA*CD
View Answer

Answer: a
Explanation: Reverse the equation or scan the equation from right to left. Apply the infix-postfix algorithm. The equation inside the brackets evaluate to DC* and BA* respectively giving us DC*BA*+ in the end. Reversing this we get the +*AB*CD.

4. What would be the Prefix notation for the given equation?

```
A+B*C^D
```

a) +A*B^CD
b) +A^B*CD
c) *A+B^CD
d) ^A*B+CD
View Answer

Answer: a
Explanation: Reverse the equation or scan the equation from right to left. Apply the infix-prefix algorithm. The preference order in ascending order are as follows +*^. Operators are pushed into the stack and popped if its preference is greater than the one which is getting pushed. In the end all operators are popped. The equation evaluates to DC^B*A+. Reversing this we get our following answer.

5. Out of the following operators (^, *, +, &, $), the one having highest priority is _________
a) +
b) $
c) ^

d) &
View Answer

Answer: c
Explanation: According to the algorithm (infix-prefix), it follows that the exponentiation will have the highest priority.

6. Out of the following operators (|, *, +, &, $), the one having lowest priority is ________
a) +
b) $
c) |
d) &
View Answer

Answer: c
Explanation: According to the algorithm (infix-prefix), it follows that the logical OR will have the lowest priority.

7. What would be the Prefix notation for the given equation?

```
A^B^C^D
```

a) ^^^ABCD
b) ^A^B^CD
c) ABCD^^^
d) AB^C^D
View Answer

Answer: a
Explanation: Reverse the equation or scan the equation from right to left. Apply the infix-prefix algorithm. Here we have to remember that the exponentiation has order of associativity from right to left. Therefore the stack goes on pushing ^. Therefore resulting in ^^^ABCD.

8. What would be the Prefix notation for the given equation?

```
a+b-c/d&e|f
```

a) |&-+ab/cdef
b) &|-+ab/cdef
c) |&-ab+/cdef
d) |&-+/abcdef
View Answer

Answer: a
Explanation: Reverse the equation or scan the equation from right to left. Apply the infix-prefix algorithm. The preference order in ascending order are as follows |&+*/.

9. What would be the Prefix notation for the given equation?

```
(a+(b/c)*(d^e)-f)
```

a) -+a*/^bcdef
b) -+a*/bc^def
c) -+a*b/c^def
d) -a+*/bc^def
View Answer

Answer: b
Explanation: Reverse the equation or scan the equation from right to left. Apply the infix-prefix algorithm. The preference order in ascending order are as follows +*/^. Brackets have the highest priority. The equations inside the brackets are solved first.

10. What would be the Prefix notation and Postfix notation for the given equation?

```
A+B+C
```

a) ++ABC and AB+C+
b) AB+C+ and ++ABC
c) ABC++ and AB+C+
d) ABC+ and ABC+
View Answer

Answer: a
Explanation: For prefix notation there is a need of reversing the giving equation and solving it as a normal infix-postfix

question. We see that it doesn't result as same as normal infix-postfix conversion.

11. What would be the Prefix notation for the given equation?

```
a|b&c
```

a) a|&bc
b) &|abc
c) |a&bc
d) ab&|c
View Answer

Answer: c
Explanation: The order of preference of operators is as follows (descending): & |.
The equation **a|b&c** will be parenthesized as **(a|(b&c))** for evaluation.
Therefore the equation for prefix notation evaluates to **|a&bc**.
Topic 25. Infix to Postfix Conversion

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on "Infix to Postfix Conversion".

1. When an operand is read, which of the following is done?
a) It is placed on to the output
b) It is placed in operator stack
c) It is ignored
d) Operator stack is emptied
View Answer

Answer: a
Explanation: While converting an infix expression to a postfix expression, when an operand is read, it is placed on to the output. When an operator is read, it is placed in the operator stack.

2. What should be done when a left parenthesis '(' is encountered?
a) It is ignored
b) It is placed in the output
c) It is placed in the operator stack
d) The contents of the operator stack is emptied
View Answer

Answer: c
Explanation: When a left parenthesis is encountered, it is placed on to the operator stack. When the corresponding right parenthesis is encountered, the stack is popped until the left parenthesis and remove both the parenthesis.

3. Which of the following is an infix expression?
a) (a+b)*(c+d)
b) ab+c*
c) +ab
d) abc+*
View Answer

Answer: a
Explanation: (a+b)*(c+d) is an infix expression. +ab is a prefix expression and ab+c* is a postfix expression.

4. What is the time complexity of an infix to postfix conversion algorithm?
a) O(N log N)
b) O(N)
c) O(N^2)
d) O(M log N)
View Answer

Answer: b
Explanation: The time complexity of an infix to postfix expression conversion algorithm is mathematically found to be O(N).

5.What is the postfix expression for the corresponding infix expression?

```
a+b*c+(d*e)
```

a) abc*+de*+
b) abc+*de*+
c) a+bc*de+*
d) abc*+(de)*+

View Answer

Answer: a
Explanation: Using the infix to postfix expression conversion algorithm, the corresponding postfix expression is found to be abc*+de*+.

6. Parentheses are simply ignored in the conversion of infix to postfix expression.
a) True
b) False
View Answer

Answer: b
Explanation: When a parenthesis is encountered, it is placed on the operator stack. When the corresponding parenthesis is encountered, the stack is popped until the other parenthesis is reached and they are discarded.

7. It is easier for a computer to process a postfix expression than an infix expression.
a) True
b) False
View Answer

Answer: a
Explanation: Computers can easily process a postfix expression because a postfix expression keeps track of precedence of operators.

8. What is the postfix expression for the infix expression?

```
a-b-c
```

a) -ab-c
b) ab – c –
c) – -abc
d) -ab-c
View Answer

Answer: b
Explanation: The corresponding postfix expression for the given infix expression is found to be ab-c- and not abc- -.

9. What is the postfix expression for the following infix expression?

```
a/b^c-d
```

a) abc^/d-
b) ab/cd^-
c) ab/^cd-
d) abcd^/-
View Answer

Answer: a
Explanation: Using the infix to postfix conversion algorithm, the corresponding postfix expression for the infix expression is found to be abc^/d-.

10. Which of the following statement is incorrect with respect to infix to postfix conversion algorithm?
a) operand is always placed in the output
b) operator is placed in the stack when the stack operator has lower precedence
c) parenthesis are included in the output
d) higher and equal priority operators follow the same condition
View Answer

Answer: c
Explanation: Parentheses are not included in the output. They are placed in the operator stack and then discarded.

11. In infix to postfix conversion algorithm, the operators are associated from?
a) right to left
b) left to right
c) centre to left
d) centre to right
View Answer

Answer: b
Explanation: In infix, prefix and postfix expressions, the operators are associated from left to right and not right to left.

12. What is the corresponding postfix expression for the given infix expression?

```
a*(b+c)/d
```

a) ab*+cd/
b) ab+*cd/
c) abc*+/d
d) abc+*d/
View Answer

Answer: d
Explanation: Using the infix to postfix conversion algorithm, the corresponding postfix expression is obtained as abc+*d/.

13. What is the corresponding postfix expression for the given infix expression?

```
a+(b*c(d/e^f)*g)*h)
```

a) ab*cdef/^*g-h+
b) abcdef^/*g*h*+
c) abcd*^ed/g*-h*+
d) abc*de^fg/*-*h+
View Answer

Answer: b
Explanation: Using the infix to postfix expression conversion algorithm using stack, the corresponding postfix expression is found to be abcdef^/*g*h*+.

14. What is the correct postfix expression for the following expression?

```
a+b*(c^d-e)^(f+g*h)-i
```

a) abc^de-fg+*^*+i-
b) abcde^-fg*+*^h*+i-
c) abcd^e-fgh*+^*+i-
d) ab^-dc*+ef^gh*+i-
View Answer

Answer: c
Explanation: The postfix expression for the given infix expression is found to be abcd^e-fgh*+^*+i- when we use infix to postfix conversion algorithm.

15. From the given Expression tree, identify the correct postfix expression from the list of options.

a) ab*cd*+
b) ab*cd-+
c) abcd-*+
d) ab*+cd-
View Answer

Answer: b
Explanation: From the given expression tree, the infix expression is found to be (a*b)+(c-d). Converting it to postfix, we get, ab*cd-+.
Topic 26. Prefix to Infix Conversion

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Prefix to Infix Conversion”.

1. What would be the solution to the given prefix notation?

```
- + 5 / 10 5 5
```

a) 2
b) 5
c) 10
d) 7
View Answer

Answer: a
Explanation: The infix notation of the given prefix notation is 5+10/5-5 which gives us 2 as our answer.

2. What would be the solution to the given prefix notation?

```
/ / / 16 4 2 1
```

a) 1

b) 4
c) 2
d) 8
View Answer

Answer: c
Explanation: The infix notation to the given prefix notation is 16/4/2/1 which gives us 2 as our answer. The infix notation is got from the prefix notation by traversing the equation from the right.
= 16/4/2/1
= 4/2/1 (16/4=4)
= 2/1 (4/2=2)
= 2 (2/1=2).

3. What would be the solution to the given prefix notation?

```
+ 9 * 3 / 8 4
```

a) 14
b) 15
c) 18
d) 12
View Answer

Answer: b
Explanation: The infix notation for the given prefix notation is (9+(3*(8/4))) which solves to 15. So 15 is correct answer.

4. What would be the solution to the given prefix notation?

```
- + 1 2 * 3 / 6 2
```

a) 6
b) -6
c) 3
d) -3
View Answer

Answer: b
Explanation: The infix notation for the given prefix notation is (1+2)-3*(6/2). The result of the given equation is -6.

5. What would be the solution to the given prefix notation?

```
- * 1 5 / * / 6 3 6 2
```

a) 1
b) 0
c) -1
d) -2
View Answer

Answer: c
Explanation: The infix notation for the given prefix notation is (1*5)-(6/3)*6/2. The result of the equation is -1.

6. What would be the solution to the given prefix notation?

```
* / + 1 2 / 4 2 + 3 5
```

a) 12
b) 7.5
c) 9
d) 13.5
View Answer

Answer: a
Explanation: The infix notation of the given prefix notation is ((1+2)/(4/2))*(3+5) which solves to (3/2)*8 which by solving gives us 12.

7. Given a prefix and a postfix notation what are the difference between them?
a) The postfix equation is solved starting from the left whereas the prefix notation is solved from the right
b) The postfix equation is solved starting from the right whereas the prefix notation is solved from the left
c) Both equations are solved starting from the same side(right)
d) Both equations are solved starting from the same side(left)
View Answer

Answer: a
Explanation: The postfix notation is solved starting from left but whereas the prefix notation is reversed after creating them, therefore it’s solved starting from right.

8. When converting the prefix notation into an infix notation, the first step to be followed is ________
a) Reverse the equation
b) Push the equation to the stack
c) Push the equation onto the queue
d) Push the equation to the stack or queue
View Answer

Answer: a
Explanation: The steps that are followed are: the equation is reversed, pushed onto a stack, popped one by one and solved. Therefore the first step is reversing the equation.

9. The time complexity of converting a prefix notation to infix notation is _________
a) O(n) where n is the length of the equation
b) O(n) where n is number of operands
c) O(1)
d) O(logn) where n is length of the equation
View Answer

Answer: a
Explanation: The processes that are involved are reversing the equation (O(n)), pushing them all onto the stack(O(n)), and popping them one by one and solving them (O(n)). Hence the answer is O(n) where n is the length of the equation.

10. Given two processes (conversion of postfix equation to infix notation and conversion of prefix notation to infix notation), which of the following is easier to implement?
a) Both are easy to implement
b) Conversion of postfix equation to infix equation is harder than converting a prefix notation to infix notation
c) Conversion of postfix equation to infix equation is easier than converting a prefix notation to infix notation
d) Insufficient data
View Answer

Answer: c
Explanation: As the conversion of prefix notation to infix notation involves reversing the equation, the latter is harder to implement than postfix to infix process.
Topic 27. Postfix to Infix Conversion

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Postfix to Infix Conversion”.

1. Which of the following data structure is used to convert postfix expression to infix expression?
a) Stack
b) Queue
c) Linked List
d) Heap
View Answer

Answer: a
Explanation: To convert the postfix expression into infix expression we need stack. We need stack to maintain the intermediate infix expressions. We use stack to hold operands.

2. The postfix expression abc+de/*- is equivalent to which of the following infix expression?
a) abc+-de*/
b) (a+b)-d/e*c
c) a-(b+c)*(d/e)
d) abc+*-(d/e)
View Answer

Answer: c
Explanation: Given postfix expression : abc+de/*-
infix a(b+c)(d/e)*-
 a(b+c)*(d/e)-
 a-(b+c)*(d/e)
Hence, correct choice is a-(b+c)*(d/e).

3. The equivalent infix expression and value for the postfix form 1 2 + 3 * 4 5 * – will be ___________
a) 1 + 2 * 3 – 4 * 5 and -13
b) (2 + 1) * (3 – 4) * 5 and 13

c) 1 + 2 * (3 – 4) * 5 and -11
d) (1 + 2) * 3 – (4 * 5) and -11
View Answer

Answer: d
Explanation: Given postfix expression : 1 2 + 3 * 4 5 * –
 (1 + 2) 3 * 4 5 * –
 ((1 + 2) * 3) 4 5 * –
 ((1 + 2) * 3) (4 * 5) –
 ((1 + 2) * 3) – (4 * 5)
So, the equivalent infix expression is (1 + 2) * 3 – (4 * 5) and it's value is -11.

4. What is the value of the postfix expression 2 3 + 4 5 6 – – *
a) 19
b) 21
c) -4
d) 25
View Answer

Answer: d
Explanation: Given postfix expression : 2 3 + 4 5 6 – – *
infix (2 + 3)4 (5 – 6) – *
 (2 + 3)*4 – (5 – 6)
Hence, value = (2 + 3) * (4 – (5 – 6)) = 5 *(4 – (-1)) = 5*5 = 25.

5. The prefix expression of the postfix expression AB+CD-* is __________
a) (A+B)*(C-D)
b) +AB*-CD
c) A+*BCD-
d) *+AB-CD
View Answer

Answer: d
Explanation: To convert from postfix to prefix, we first convert it to infix and then to prefix.
postfix : AB+CD-*
infix (A+B) * (C-D)
So, prefix +AB*-CD,
 *+AB-CD.
Therefore, correct choice is *+AB-CD.

6. Consider the postfix expression 4 5 6 a b 7 8 a c, where a, b, c are operators. Operator a has higher precedence over operators b and c. Operators b and c are right associative. Then, equivalent infix expression is
a) 4 a 5 6 b 7 8 a c
b) 4 a 5 c 6 b 7 a 8
c) 4 b 5 a 6 c 7 a 8
d) 4 a 5 b 6 c 7 a 8
View Answer

Answer: c
Explanation: Given postfix expression: 4 5 6 a b 7 8 a c
infix 4 (5 a 6) b (7 a 8) c
 (4 b (5 a 6)) (7 a 8) c
 (4 b (5 a 6)) c (7 a 8)
So, the required infix expression is 4 b 5 a 6 c 7 a 8.

7. To convert the postfix expression into the infix expression we use stack and scan the postfix expression from left to right.
a) True
b) False
View Answer

Answer: a
Explanation: Stack is used to postfix expression to infix expression. And to convert we follow the following steps: (i) Scan the expression from left to right. (ii) If operand is found, push it on stack.(iii) If operator is found, the two operands are popped and the combined infix expression is formed and pushed onto the stack.

8. Which of the following is valid reverse polish expression?
a) a op b
b) op a b
c) a b op
d) both op a b and a b op

View Answer

Answer: c
Explanation: The postfix expression is also known as the reverse polish expression. In postfix expressions, the operators come after the operands. So, the correct expression is a b op and hence a b op is correct.

9. The result of the postfix expression 5 3 * 9 + 6 / 8 4 / + is _____________
a) 8
b) 6
c) 10
d) 9
View Answer

Answer: b
Explanation: Given postfix expression: 5 3 * 9 + 6 / 8 4 / +
Result = 5 3 * 9 + 6 / 8 4 / +
= (5 * 3) 9 + 6 / (8 / 4) +
= ((5 * 3) + 9) / 6 + (8 / 4) = (24 / 6) + 2 = 4 + 2 = 6.
Topic 28. Towers of Hanoi

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Towers of Hanoi”.

1. The optimal data structure used to solve Tower of Hanoi is _________
a) Tree
b) Heap
c) Priority queue
d) Stack
View Answer

Answer: d
Explanation: The Tower of Hanoi involves moving of disks ‘stacked’ at one peg to another peg with respect to the size constraint. It is conveniently done using stacks and priority queues. Stack approach is widely used to solve Tower of Hanoi.

2. Select the appropriate code for the recursive Tower of Hanoi problem.(n is the number of disks)
a)

```
public void solve(int n, String start, String auxiliary, String end)
{
      if (n == 1)
      {
          System.out.println(start + " -> " + end);
      }
      else
      {
          solve(n - 1, start, end, auxiliary);
          System.out.println(start + " -> " + end);
          solve(n - 1, auxiliary, start, end);
      }
}
```

b)

```
public void solve(int n, String start, String auxiliary, String end)
{
      if (n == 1)
      {
          System.out.println(start + " -> " + end);
      }
      else
      {
          solve(n - 1, auxiliary, start, end);
          System.out.println(start + " -> " + end);
      }
}
```

c)

```
public void solve(int n, String start, String auxiliary, String end)
{
      if (n == 1)
      {
          System.out.println(start + " -> " + end);
      }
      else
      {
          System.out.println(start + " -> " + end);
    solve(n - 1, auxiliary, start, end);
      }
}
```

d)

```
public void solve(int n, String start, String auxiliary, String end)
{
      if (n == 1)
      {
          System.out.println(start + " -> " + end);
      }
      else
      {
          solve(n - 1, start, end, auxiliary);
          System.out.println(start + " -> " + end);
      }
}
```

View Answer
Answer: a
Explanation: First transfer all the diska to the auxiliary and then to the end peg, this is achieved by making auxiliary peg as the end peg in the first recursive call, in the second recursive call, the auxiliary becomes the start peg from where the disks are transferred to the end peg.

3. Which among the following is not a palindrome?
a) Madam
b) Dad
c) Malayalam
d) Maadam
View Answer

Answer: d
Explanation: A palindrome is a string that reads the same forward and backward, Madam, Dad and Malayalam are palindromes where as Maadam is not a palindrome.

4. Which data structure can be used to test a palindrome?
a) Tree
b) Heap
c) Stack
d) Priority queue
View Answer

Answer: c
Explanation: Stack is a convenient option as it involves pushing and popping of characters.

5. Select the appropriate code which tests for a palindrome.
a)

```
public static void main(String[] args)
{
 System.out.print("Enter any string:");
       Scanner in=new Scanner(System.in);
       String input = in.nextLine();
       Stack<Character> stk = new Stack<Character>();
 for (int i = 0; i < input.length(); i++)
 {
           stk.push(input.charAt(i));
       }
 String reverse = "";
 while (!stk.isEmpty())
 {
           reverse = reverse + stk.pop();
       }
 if (input.equals(reverse))
       System.out.println("palindrome");
       else
       System.out.println("not a palindrome");
}
```

b)

```
public static void main(String[] args)
{
 System.out.print("Enter any string:");
       Scanner in=new Scanner(System.in);
       String input = in.nextLine();
       Stack<Character> stk = new Stack<Character>();
 for (int i = 0; i < input.length(); i++)
 {
           stk.push(input.charAt(i));
       }
 String reverse = "";
 while (!stk.isEmpty())
 {
           reverse = reverse + stk.peek();
       }
 if (input.equals(reverse))
       System.out.println("palindrome");
       else
           System.out.println("not a palindrome");
}
```

c)

```
public static void main(String[] args)
{
 System.out.print("Enter any string:");
       Scanner in=new Scanner(System.in);
       String input = in.nextLine();
       Stack<Character> stk = new Stack<Character>();
 for (int i = 0; i < input.length(); i++)
 {
           stk.push(input.charAt(i));
       }
 String reverse = "";
 while (!stk.isEmpty())
 {
           reverse = reverse + stk.pop();
   stk.pop();
       }
 if (input.equals(reverse))
       System.out.println("palindrome");
       else
           System.out.println("not a palindrome");
}
```

d)

```
public static void main(String[] args)
{
 System.out.print("Enter any string:");
       Scanner in=new Scanner(System.in);
       String input = in.nextLine();
       Stack<Character> stk = new Stack<Character>();
 for (int i = 0; i < input.length(); i++)
 {
           stk.push(input.charAt(i));
       }
 String reverse = "";
 while (!stk.isEmpty())
 {
           reverse = reverse + stk.pop();
   stk.pop();
       }
 if (!input.equals(reverse))
       System.out.println("palindrome");
       else
           System.out.println("not a palindrome");
}
```

View Answer
Answer: a
Explanation: Push all the characters in the input string to a stack, now pop them and append to a new string which is checked for equality with the original string.

6. What is the number of moves required to solve Tower of Hanoi problem for k disks?
a) 2k – 1
b) 2k + 1
c) $2^k + 1$
d) $2^k – 1$
View Answer

Answer: d
Explanation: Tracing of the moves in the above ToH problem will prove this result, instead you can simply add a count for each recursive call to check the number of moves.

7. Select the appropriate code which reverses a word.

a)

```
public String reverse(String input)
{
 for (int i = 0; i < input.length(); i++)
 {
           stk.push(input.charAt(i));
       }
 String rev = "";
 while (!stk.isEmpty())
 {
           rev = rev + stk.peek();
       }
 return rev;
}
```

b)

```
public String reverse(String input)
{
 for (int i = 0; i < input.length(); i++)
 {
           stk.push(input.charAt(i));
       }
 String rev = "";
 while (!stk.isEmpty())
 {
           rev = rev + stk.pop();
       }
 return rev;
}
```

c)

```
public String reverse(String input)
{
 for (int i = 0; i < input.length(); i++)
 {
           stk.push(input.charAt(i));
       }
 String rev = "";
 while (!stk.isEmpty())
 {
           rev = rev + stk.pop();
       }
}
```

d)

```
public String reverse(String input)
{
 for (int i = 0; i < input.length(); i++)
 {
           stk.push(input.charAt(i));
       }
 String rev = "";
 while (!stk.isEmpty())
 {
           rev = rev + stk.pop();
     stk.pop();
       }
 return rev;
}
```

View Answer
Answer: b
Explanation: Although, it is possible to reverse the string without using stack, it is done by looping through the string from the end character by character.
In Java, it is also possible to use the StringBuilder and StringBuffer classes which have a built-in method ‘reverse’.
Note its similarity to PalindromeTest.
Topic 29. Reverse a Word using Stack

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Reverse a Word using Stack”.

1. Reversing a word using stack can be used to find if the given word is a palindrome or not.
a) True
b) False
View Answer

Answer: a

Explanation: This application of stack can also be used to find if the given word is a palindrome because, if the reversed is same as that of the original word, the given word is a palindrome.

2. Which is the most appropriate data structure for reversing a word?
a) queue
b) stack
c) tree
d) graph
View Answer

Answer: b
Explanation: Stack is the most appropriate data structure for reversing a word because stack follows LIFO principle.

3. Operations required for reversing a word or a string using stack are push() and pop().
a) True
b) False
View Answer

Answer: a
Explanation: Push operation inserts a character into the stack and pop operation pops the top of the stack.

4. What is the time complexity of reversing a word using stack algorithm?
a) O (N log N)
b) O (N^2)
c) O (N)
d) O (M log N)
View Answer

Answer: c
Explanation: The time complexity of reversing a stack is mathematically found to be O (N) where N is the input.

5. What will be the word obtained if the word “abbcabb” is reversed using a stack?
a) bbabbca
b) abbcabb
c) bbacbba
d) bbacabb
View Answer

Answer: c
Explanation: The string “abbcabb” is pushed on to the stack. If the characters are popped one by one, the word obtained will be bbacbba.

6. How many stacks are required for reversing a word algorithm?
a) one
b) two
c) three
d) four
View Answer

Answer: a
Explanation: Only 1 stack is required for reversing a word using stack. In that stack, push and pop operations are carried out.

7. What will be result if the given stack is popped?

a) pat
b) tap
c) atp
d) apt
View Answer

Answer: b
Explanation: The word ‘pat’ is pushed on to the stack. When the characters of the stack are popped one by one, the word ‘tap’ is obtained.

8. What will be output if the following sequence of operations are executed?

```
Push(a,s);
Push(b,s);
Pop(b);
Push(c,s);
```

a) abc
b) b
c) ac
d) acb
View Answer

Answer: b
Explanation: The element ‘b’ is popped out of the stack. Hence the output of the following sequence of operations will be ‘b’.

9. What are the set of functions that are to be executed to get the following output?
cat
a) push(c, s); push(a, s); push(t, s);
pop(s); pop(s); pop(s);
b) push(c,s); pop(s); push(a,s); pop(s);push(t,s);pop(s);
c) pop(c); pop(a); pop(t);
d) push(c,s); push(a,s); pop(t);
View Answer

Answer: b
Explanation: During push operation, the characters ‘c’, ’a’, ’t’ are inserted into the stack and popped immediately after push.

10. How will your stack look like if the word ‘java’ is pushed?
a)
b)
c)
d)
View Answer

Answer: a
Explanation: When a character is pushed, it stays on the top of the stack. While popping, the word occurs in reverse order since stack follows LIFO principle.

11. Find the error (if any) in the following code snippet for pop operation.

```
void pop() //removing an element from a stack
{
    printf(“%s”, stack[top++]);
}
```

a) run time error
b) compile time error
c) pop operation is performed, but top moved in wrong direction
d) pop operation is performed properly
View Answer

Answer: c
Explanation: The statement printf(“%s”, stack[top++]) does a pop, but top gets incremented which is not correct. The statement stack[top++] should be replaced with stack[top–] in order to pop an operand and maintain stack properly.

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

```
main()
{
   char str[]="san foundry";
   int len = strlen(str);
   int i;

   for(i=0;i<len;i++)
        push(str[i]);  // pushes an element into stack

   for(i=0;i<len;i++)
      pop();  //pops an element from the stack
}
```

a) sanfoundry
b) san foundry
c) yrdnuof nas
d) foundry nas
View Answer

Answer: c
Explanation: First, the string ‘san foundry’ is pushed one by one into the stack.
When it is popped, the output will be as ‘yrdnuof nas’.
Topic 30. Balanced Parenthesis

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Balanced Parenthesis”.

1. What is the time complexity of balancing parentheses algorithm?
a) O (N)
b) O (N log N)
c) O (M log N)
d) O (N^2)
View Answer

Answer: a
Explanation: The time complexity of balancing parentheses algorithm is mathematically found to be O (N).

2. Which application of stack is used to ensure that the pair of parentheses is properly nested?
a) Balancing symbols
b) Reversing a stack
c) Conversion of an infix to postfix expression
d) Conversion of an infix to prefix expression
View Answer

Answer: a
Explanation: Balancing symbols application ensures that the pair of parentheses are properly nested while reversing stack reverses a stack.

3. In balancing parentheses algorithm, the string is read from?
a) right to left
b) left to right
c) center to right
d) center to left
View Answer

Answer: b
Explanation: Any string is read by the compiler from left to right and not from right to left.

4. Which is the most appropriate data structure for applying balancing of symbols algorithm?
a) stack
b) queue
c) tree
d) graph
View Answer

Answer: a
Explanation: Stack is the most appropriate data structure for balancing symbols algorithm because stack follows LIFO principle (Last In First Out).

5. Which of the following does the balancing symbols algorithm include?
a) balancing double quotes
b) balancing single quotes
c) balancing operators and brackets
d) balancing parentheses, brackets and braces
View Answer

Answer: d
Explanation: The balancing symbols algorithm using stack only includes balancing parentheses, brackets and braces and not any other symbols.

6. Which of the following statement is incorrect with respect to balancing symbols algorithm?
a) {[()]}
b) ([ )]
c) {( )}
d) { [ ] }
View Answer

Answer: b
Explanation: ([ )] is incorrect because’)’ occurs before the corresponding ‘]’ is encountered.

7. What should be done when an opening parentheses is read in a balancing symbols algorithm?
a) push it on to the stack
b) throw an error
c) ignore the parentheses
d) pop the stack

View Answer

Answer: a
Explanation: When an opening bracket/braces/parentheses is encountered, it is pushed on to the stack. When the corresponding end bracket/braces/parentheses is not found, throw an error.

8. When the corresponding end bracket/braces/parentheses is not found, what happens?
a) The stack is popped
b) Ignore the parentheses
c) An error is reported
d) It is treated as an exception
View Answer

Answer: c
Explanation: When the corresponding end bracket/braces/parentheses is not found, throw an error since they don't match.

9. If the corresponding end bracket/braces/parentheses is encountered, which of the following is done?
a) push it on to the stack
b) pop the stack
c) throw an error
d) treated as an exception
View Answer

Answer: b
Explanation: When the corresponding end bracket/braces/parentheses is encountered, the stack is popped. When an opening bracket/braces/parentheses is encountered, it is pushed on to the stack.

10. An error is reported when the stack is not empty at the end.
a) True
b) False
View Answer

Answer: a
Explanation: When the stack contains elements at the end, it means that the given string of parentheses is not balanced.

11. Is the given statement ((A+B) + [C-D]] valid with respect to balancing of symbols?
a) True
b) False
View Answer

Answer: b
Explanation: The given statement is invalid with respect to balancing of symbols because the last parentheses do not correspond to the opening braces.

12. How many passes does the balancing symbols algorithm makes through the input?
a) one
b) two
c) three
d) four
View Answer

Answer: a
Explanation: The balancing symbols algorithm makes only one pass through the input since it is linear.

13. Which of the following statement is invalid with respect to balancing symbols?
a) [(A+B) + (C-D)]
b) [{A+B}-{C-[D+E]}]
c) ((A+B) + (C+D)
d) {(A+B) + [C+D]}
View Answer

Answer: c
Explanation: ((A+B) + (C+D) is invalid because the last close parentheses is not found in the statement.
Topic 31. Bit Array

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on "Bit Array".

1. What is a bit array?
a) Data structure for representing arrays of records
b) Data structure that compactly stores bits
c) An array in which most of the elements have the same value

d) Array in which elements are not present in continuous locations
View Answer

Answer: b
Explanation: It compactly stores bits and exploits bit-level parallelism.

2. Which of the following bitwise operations will you use to set a particular bit to 1?
a) OR
b) AND
c) XOR
d) NOR
View Answer

Answer: a
Explanation: 1 OR 1 = 1, 0 OR 1 = 1, any bit OR'ed with 1 gives 1.

3. Which of the following bitwise operations will you use to set a particular bit to 0?
a) OR
b) AND
c) XOR
d) NAND
View Answer

Answer: b
Explanation: 1 AND 0 = 0, 0 AND 0 = 0, any bit AND with 0 gives 0.

4. Which of the following bitwise operations will you use to toggle a particular bit?
a) OR
b) AND
c) XOR
d) NOT
View Answer

Answer: c
Explanation: 1 XOR 1 = 0, 0 XOR 1 = 1, note that NOT inverts all the bits, while XOR toggles only a specified bit.

5. Which of the following is not an advantage of bit array?
a) Exploit bit level parallelism
b) Maximal use of data cache
c) Can be stored and manipulated in the register set for long periods of time
d) Accessing Individual Elements is easy
View Answer

Answer: d
Explanation: Individual Elements are difficult to access and can't be accessed in some programming languages. If random access is more common than sequential access, they have to be compressed to byte/word array. Exploit Bit parallelism, Maximal use of data cache and storage and manipulation for longer time in register set are all advantages of bit array.

6. Which of the following is not a disadvantage of bit array?
a) Without compression, they might become sparse
b) Accessing individual bits is expensive
c) Compressing bit array to byte/word array, the machine also has to support byte/word addressing
d) Storing and Manipulating in the register set for long periods of time
View Answer

Answer: d
Explanation: Bit arrays allow small arrays of bits to be stored and manipulated in the register set for long periods of time with no memory accesses because of their ability to exploit bit-level parallelism, limit memory access, and maximally use the data cache, they often outperform many other data structures on practical data sets. This is an advantage of bit array. The rest are all disadvantages of bit array.

7. Which of the following is/are not applications of bit arrays?
a) Used by the Linux kernel
b) For the allocation of memory pages
c) Bloom filter
d) Implementation of Vectors and Matrices
View Answer

Answer: d
Explanation: Normal Arrays are used to implement vectors and matrices. Bit arrays have no prominent role. Remaining all

are applications of Bit Arrays.

8. Which class in Java can be used to represent bit array?
a) BitSet
b) BitVector
c) BitArray
d) BitStream
View Answer

Answer: a
Explanation: The BitSet class creates a special type of array that can hold bit values.

9. Which of the following bitwise operator will you use to invert all the bits in a bit array?
a) OR
b) NOT
c) XOR
d) NAND
View Answer

Answer: b
Explanation: NOT operation is used to invert all the bits stored in a bit array.
Eg: NOT (10110010) = 01001101.

10. Run-Length encoding is used to compress data in bit arrays.
a) True
b) False
View Answer

Answer: a
Explanation: A bit array stores the combinations of bit 0 and bit 1. Each bit in the bit array is independent. Run Length encoding is a data compression technique in which data are stored as single value and number of times that value repeated in the data. This compression reduces the space complexity in arrays. Bit arrays without compression require more space. Thus, we will use Run-Length encoding in most of the cases to compress data in bit arrays.

11. What does Hamming weight/population count mean in Bit arrays?
a) Finding the number of 1 bit in a bit array
b) Finding the number of 0 bit in a bit array
c) Finding the sum of bits in a bit array
d) Finding the average number of 1’s and 0’s in bit arrays
View Answer

Answer: a
Explanation: Hamming/ population count involves finding the number of 1’s in the bit array. Population count is used in data compression.

12. Bit fields and Bit arrays are same.
a) True
b) False
View Answer

Answer: b
Explanation: Bit field contains the number of adjacent computer locations used to store the sequence of bits to address a bit or groups of bits. Bit array is an array that stores combinations of bit 0 and bit 1. Thus, bit fields and Bit arrays are different.

13. Which one of the following operations returns the first occurrence of bit 1 in bit arrays?
a) Find First Zero
b) Find First One
c) Counting lead Zeroes
d) Counting lead One
View Answer

Answer: b
Explanation: Find First One operation returns the first occurrence of bit 1 in the bit array. Find First Zero operation returns the first occurrence of bit 0 in the bit array. If the most significant bit in bit array is 1, then count lead zeroes operation returns the number of zeroes present before the most significant bit. If the most significant bit in bit array is 0, then the count lead one returns the number of ones present before the most significant bit.
Topic 32. Dynamic Array

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Dynamic Array”.

1. What is a dynamic array?

a) A variable size data structure
b) An array which is created at runtime
c) The memory to the array is allocated at runtime
d) An array which is reallocated everytime whenever new elements have to be added
View Answer

Answer: a
Explanation: It is a varying-size list data structure that allows items to be added or removed, it may use a fixed sized array at the back end.

2. What is meant by physical size in a dynamic array?
a) The size allocated to elements
b) The size extended to add new elements
c) The size of the underlying array at the back-end
d) The size visible to users
View Answer

Answer: c
Explanation: Physical size, also called array capacity is the size of the underlying array, which is the maximum size without relocation of data.

3. The number of items used by the dynamic array contents is its __________
a) Physical size
b) Capacity
c) Logical size
d) Random size
View Answer

Answer: c
Explanation: The number of items used by the dynamic array contents is called logical size. Physical size is the size of the underlying array, which is the maximum size without reallocation of data.

4. How will you implement dynamic arrays in Java?
a) Set
b) Map
c) HashMap
d) List
View Answer

Answer: d
Explanation: ArrayList is used to implement dynamic arrays in Java.

5. Which of the following is the correct syntax to declare an ArrayList in Java?
a) ArrayList al = new ArrayList();
b) ArrayList al = new ArrayList[];
c) ArrayList al() = new ArrayList();
d) ArrayList al[] = new ArrayList[];
View Answer

Answer: a
Explanation: This is a non-generic way of creating an ArrayList.

6. Array is divided into two parts in ____________
a) Hashed Array Tree
b) Geometric Array
c) Bounded-size dynamic array
d) Sparse Array
View Answer

Answer: c
Explanation: The first part stores the items of the dynamic array and the second part is reserved for new allocations.

7. Which of the following is a disadvantage of dynamic arrays?
a) Locality of reference
b) Data cache utilization
c) Random access
d) Memory leak
View Answer

Answer: d

Explanation: Dynamic arrays share the advantage of arrays, added to it is the dynamic addition of elements to the array. Memory can be leaked if it is not handled properly during allocation and deallocation. It is a disadvantage.

8. What is the time complexity for inserting/deleting at the beginning of the array?
a) O(1)
b) O(n)
c) O(logn)
d) O(nlogn)
View Answer

Answer: b
Explanation: All the other elements will have to be moved, hence O(n).

9. Dynamic arrays overcome the limit of static arrays.
a) True
b) False
View Answer

Answer: a
Explanation: Static arrays have fixed capacity. The capacity must be specified during memory allocation. Dynamic arrays don’t require to specify their capacity during memory allocation. Dynamic arrays have fixed physical size at backend and its capacity increases if required. Thus, Dynamic arrays overcome the limit of static arrays.

10. The size of the dynamic array is deallocated if the array size is less than _________% of the backend physical size.
a) 30
b) 40
c) 10
d) 20
View Answer

Answer: a
Explanation: The size of the dynamic array is decreased/deallocated if the actual size of the array is less than 30% of the backend physical size. This is used to avoid memory wastage.

11. Both Dynamic array and Dynamically memory allocated array are same.
a) True
b) False
View Answer

Answer: b
Explanation: Physical size of a Dynamic array is fixed with a larger value. Dynamically memory allocated arrays are arrays whose memory is allocated at run time rather than at compile time. Dynamically memory allocated arrays don’t have physical size at the backend. Thus, Dynamic arrays and Dynamically memory allocated arrays are different.

12. In which of the following cases dynamic arrays are not preferred?
a) If the size of the array is unknown
b) If the size of the array changes after few iterations
c) If the memory reallocation takes more time i.e. expensive
d) If the array holds less number of elements
View Answer

Answer: d
Explanation: Dynamic arrays are preferred when the size of the array is unknown during memory allocation or the size changes after few iterations or the memory reallocation is expensive. If array holds less number of elements, the physical size is reduced and reduction takes more time. In that case, we can use normal arrays instead of dynamic arrays.

13. The growth factor of ArrayList in Java is _______
a) 1
b) 1.5
c) 2
d) 0
View Answer

Answer: b
Explanation: The growth factor of dynamic arrays (Array List) in Java is 3/2.
The new array capacity is calculated as new_array_size = (old_array_size*3)/2+1.

14. In special case, the time complexity of inserting/deleting elements at the end of dynamic array is __________
a) O (n)
b) O ($n^{1/2}$)

c) O (log n)
d) O (1)
View Answer

Answer: a
Explanation: In general, the time complexity of inserting or deleting elements at the end of dynamic array is O (1). Elements are added at reserved space of dynamic array. If this reserved space is exceeded, then the physical size of the dynamic array is reallocated and every element is copied from original array. This will take O(n) time to add new element at the end of the array.

15. Which of the following arrays are used in the implementation of list data type in python?
a) Bit array
b) Dynamic arrays
c) Sparse arrays
d) Parallel arrays
View Answer

Answer: b
Explanation: Dynamic arrays are used in the implementation of list data type in python. Sparse arrays are used in the implementation of sparse matrix in Numpy module. All bit array operations are implemented in bitarray module.
Topic 33. Parallel Array

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Parallel Array”.

1. What are parallel arrays?
a) Arrays of the same size
b) Arrays allocated one after the other
c) Arrays of the same number of elements
d) Arrays allocated dynamically
View Answer

Answer: c
Explanation: Different arrays can be of different data types but should contain same number of elements. Elements at corresponding index belong to a record.

2. Which of the following can be called a parallel array implementation?
a)

```
firstName  = ['Joe','Bob','Frank','Hans']
lastName   = ['Smith','Seger','Sinatra','Schultze']
heightInCM = [169,158,201,199]

for i in xrange(len(firstName)):
    print "Name:",firstName[i], lastName[i]
    print "Height in CM:,",heightInCM[i]
```

b)

```
firstName  = ['Joe','Bob','Frank','Hans']
lastName   = ['Smith','Seger']
heightInCM = [169,158]

for i in xrange(len(firstName)):
    print "Name:",firstName[i], lastName[i]
    print "Height in CM:,",heightInCM[i]
```

c)

```
firstName  = ['Joe','Bob']
lastName   = ['Smith','Seger','Sinatra','Schultze']
heightInCM = [169,158]

for i in xrange(len(firstName)):
    print "Name:",firstName[i], lastName[i]
    print "Height in CM:,",heightInCM[i]
```

d)

```
firstName  = ['Joe','Bob']
lastName   = ['Smith','Seger' ,'Schultze']
heightInCM = [169,158]

for i in xrange(len(firstName)):
    print "Name:",firstName[i], lastName[i]
    print "Height in CM:,",heightInCM[i]
```

View Answer

Answer: a
Explanation: All the arrays must have equal length, that is, contain same number of elements.

3. Which of the following is a disadvantage of parallel array over the traditional arrays?
a) When a language does not support records, parallel arrays can be used
b) Increased locality of reference
c) Ideal cache behaviour
d) Insertion and Deletion becomes tedious
View Answer

Answer: d
Explanation: Insertion and deletion of elements require to move every element from their initial positions. This will become tedious. For Record collection, locality of reference and Ideal Cache behaviour we can use parallel arrays.

4. Which of the following is an advantage of parallel arrays?
a) Poor locality of reference for non-sequential access
b) Very little direct language support
c) Expensive to shrink or grow
d) Increased Locality of Reference
View Answer

Answer: d
Explanation: Elements in the parallel array are accessed sequentially as one arrays holds the keys whereas other holds the values. This sequential access generally improves Locality of Reference. It is an advantage.

5. What is a sorted array?
a) Arrays sorted in numerical order
b) Arrays sorted in alphabetical order
c) Elements of the array are placed at equally spaced addresses in the memory
d) All of the mentioned
View Answer

Answer: d
Explanation: The array can be sorted in any way, numerical, alphabetical or any other way but the elements are placed at equally spaced addresses.

6. To search for an element in a sorted array, which searching technique can be used?
a) Linear Search
b) Jump Search
c) Binary Search
d) Fibonacci Search
View Answer

Answer: c
Explanation: Since the array is sorted, binary search is preferred as its time complexity is O(logn).

7. Which of the following is not an application of sorted array?
a) Commercial computing
b) Priority Scheduling
c) Discrete Mathematics
d) Hash Tables
View Answer

Answer: d
Explanation: Sorted arrays have widespread applications as all commercial computing involves large data which is very useful if it is sorted. It makes best use of locality of reference and data cache. Linked lists are used in Hash Tables not arrays.

8. What is the worst case time complexity of inserting an element into the sorted array?
a) O(nlogn)
b) O(logn)
c) O(n)
d) $O(n^2)$
View Answer

Answer: c
Explanation: In the worst case, an element must added to the front of the array, which means that rest of the elements have to be shifted, hence the worst case time complexity becomes O(n).

Topic 34. Sparse Array

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Sparse Array”.

1. What is a sparse array?
a) Data structure for representing arrays of records
b) Data structure that compactly stores bits
c) An array in which most of the elements have the same value
d) An array in which memory is allocated in run time
View Answer

Answer: c
Explanation: They are set to a default value, usually 0 or null.

2. When do you use a sparse array?
a) When there are unique elements in the array
b) When the array has more occurrence of zero elements
c) When the data type of elements differ
d) When elements are sorted
View Answer

Answer: b
Explanation: It need not necessarily be zero, it could be any default value, usually zero or null.

3. What is the difference between a normal(naive) array and a sparse array?
a) Sparse array can hold more elements than a normal array
b) Sparse array is memory efficient
c) Sparse array is dynamic
d) A naive array is more efficient
View Answer

Answer: b
Explanation: A naive implementation allocates space for the entire size of the array, whereas a sparse array(linked list implementation) allocates space only for the non-default values.

4. Choose the code which performs the store operation in a sparse array.(Linked list implementation)
a)

```
public void store(int index, Object val)
{
      List cur = this;
      List prev = null;

      List node = new List(index);
      node.val = val;

      while (cur != null && cur.index < index)
      {
          prev = cur;
          cur = cur.next;
      }

      if (cur == null)
      {
          prev.next = node;
      } else
      {
          if (cur.index == index)
          {
              System.out.println("DUPLICATE");
              return;
          }
          prev.next = node;
          node.next = cur;
      }
      return;
}
```

b)

```
public void store(int index, Object val)
{
       List cur = this;
       List prev = null;

       List node = new List(index);
       node.val = val;
```

```
        while (prev != null && prev.index < index)
        {
            prev = cur;
            cur = cur.next;
        }

        if (cur == null)
        {
            prev.next = node;
        } else
        {
            if (cur.index == index)
            {
                System.out.println("DUPLICATE");
                return;
            }
            prev.next = node;
            node.next = cur;
        }
        return;
}
```

c)

```
public void store(int index, Object val)
{
        List cur = this;
        List prev = null;

        List node = new List(index);
        node.val = val;

        while (cur != null && cur.index < index)
        {
   cur = cur.next;
            prev = cur;
        }

        if (cur == null)
        {
            prev.next = node;
        } else
        {
            if (cur.index == index)
            {
                System.out.println("DUPLICATE");
                return;
            }
            prev.next = node;
            node.next = cur;
        }
        return;
}
```

d)

```
public void store(int index, Object val)
{
    List cur = this;
    List prev = null;

    List node = new List(index);
    node.val = val;

    while (cur != null && prev.index < index)
    {
        cur = cur.next;
        prev = cur;
    }

    if (cur == null)
    {
        prev.next = node;
    }
    else
    {
        if (cur.index == index)
    {
        System.out.println("DUPLICATE");
        return;
    }
    prev.next = cur;
    node.next = node;
    }
    return;
}
```

View Answer
Answer: a
Explanation: Create a new node and traverse through the list until you reach the correct position, check for duplicate and nullity of the list and then insert the node.

5. Which of the following performs the fetch operation?
a)

```
public Object fetch(int index)
{
        List cur = this.next;
        Object val = null;
        while (cur != null && cur.index != index)
        {
            cur = cur.next;
        }
        if (cur != null)
        {
            val = cur.val;
        } else
        {
            val = null;
        }
        return val;
}
```

b)

```
public Object fetch(int index)
{
        List cur = this;
        Object val = null;
        while (cur != null && cur.index != index)
        {
            cur = cur.next;
        }
        if (cur != null)
        {
            val = cur.val;
        } else
        {
            val = null;
        }
        return val;
}
```

c)

```
public Object fetch(int index)
{
        List cur = this;
        Object val = null;
        while (cur != null && cur.index != index)
        {
            cur = cur.index;
        }
        if (cur != null)
        {
            val = cur.val;
        } else
        {
            val = null;
        }
        return val;
}
```

d)

```
public Object fetch(int index)
{
    List cur = this.next;
    Object val = null;
    while (cur != null && cur.index != index)
    {
        cur = cur.index;
    }
    if (cur != null)
    {
        val = cur.val;
    }
    else
    {
        val = null;
    }
    return val;
}
```

View Answer
Answer: b
Explanation: You traverse through the array until the end is reached or the index is found and return the element at that index, else null is returned.

6. Choose the appropriate code that counts the number of non-zero(non-null) elements in the sparse array.
a)

```
public int count()
{
        int count = 0;
        for (List cur = this.next; (cur != null); cur = cur.next)
        {
            count++;
        }
        return count;
}
```

b)

```
public int count()
{
        int count = 0;
        for (List cur = this; (cur != null); cur = cur.next)
        {
            count++;
        }
        return count;
}
```

c)

```
public int count()
{
        int count = 1;
        for (List cur = this.next; (cur != null); cur = cur.next)
        {
            count++;
        }
        return count;
}
```

d)

```
public int count()
{
    int count = 1;
    for (List cur = this.next; (cur != null); cur = cur.next.next)
    {
        count++;
    }
    return count;
}
```

View Answer
Answer: a
Explanation: A simple 'for loop' to count the non-null elements.

7. Suppose the contents of an array A are, A = {1, null, null, null, null, 10};
What would be the size of the array considering it as a normal array and a sparse array?
a) 6 and 6
b) 6 and 2
c) 2 and 6
d) 2 and 2
View Answer

Answer: b
Explanation: A normal array considers null also as an element, but in the sparse array only a non-zero or a non-null element is considered.

8. What is sparsity of a matrix?
a) The fraction of zero elements over the total number of elements
b) The fraction of non-zero elements over the total number of elements
c) The fraction of total number of elements over the zero elements
d) The fraction of total number of elements over the non-zero elements
View Answer

Answer: a
Explanation: Sparsity of a matrix is the fraction of number of zero elements over the total number of zero elements.

9. How would you store an element in a sparse matrix?
a)

```
public void store(int row_index, int col_index, Object val)
{
        if (row_index < 0 || row_index > N)
 {
            System.out.println("row index out of bounds");
   return;
 }
        if (col_index < 0 || col+index > N)
 {
            System.out.println("column index out of bounds");
   return;
 }
        sparse_array[row_index].store(col_index, val);
}
```

b)

```
public void store(int row_index, int col_index, Object val)
{
        if (row_index < 0 || row_index > N)
 {
            System.out.println("column index out of bounds");
   return;
 }
        if (col_index < 0 || col+index > N)
 {
            System.out.println("row index out of bounds");
   return;
 }
        sparse_array[row_index].store(col_index, val);
}
```

c)

```
public void store(int row_index, int col_index, Object val)
{
        if (row_index < 0 && row_index > N)
 {
            System.out.println("row index out of bounds");
   return;
 }
        if (col_index < 0 && col+index > N)
 {
            System.out.println("column index out of bounds");
   return;
 }
        sparse_array[row_index].store(col_index, val);
    }
```

d)

```
public void store(int row_index, int col_index, Object val)
{
        if (row_index < 0 && row_index > N)
 {
            System.out.println("column index out of bounds");
   return;
 }
        if (col_index < 0 && col+index > N)
 {
            System.out.println("row index out of bounds");
   return;
 }
        sparse_array[row_index].store(col_index, val);
}
```

View Answer
Answer: a
Explanation: Each row in a sparse matrix acts as a sparse array, hence this row with the specified col_index is the array and

the specified position where the element is stored.

10. What is the functionality of the following piece of code?

```
public Object function(int row_index, int col_index)
{
      if (row_index < 0 || col_index > N)
 {
         System.out.println("column index out of bounds");
   return;
 }
      return (sparse_array[row_index].fetch(col_index));
}
```

a) Store the element in the specified position
b) Get the element from the specified position
c) Alter the element in the specified position
d) Removes the element from the specified position
View Answer

Answer: b
Explanation: The fetch method of SparseArray class is called , the row specified by row_index makes it an array and the col_index denotes the specified position.

11. Which of the following is the disadvantage of sparse matrices over normal matrices?
a) Size
b) Speed
c) Easily compressible
d) Algorithm complexity
View Answer

Answer: d
Explanation: As the sparse matrix contains zeroes we will compute operations only on non zero values. This increases the complexity of algorithm as we need to identify index of zero elements first and during computation we should not take those index. It is a disadvantage. Sparse matrix is easily compressible by not storing the zero/null elements, they require less memory space, also only the non zero elements have to be computed, hence computational speed increases.
Topic 35. Suffix Array

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Suffix Array”.

1. Which of the following is false?
a) Suffix array is always sorted
b) Suffix array is used in string matching problems
c) Suffix array is always unsorted
d) Suffix array contains all the suffixes of the given string
View Answer

Answer: c
Explanation: Suffix array is always sorted as it contains all the suffixes of a string in sorted order. Suffix arrays are used to solve problems related to string, like string matching problems.

2. Suffix array of the string “statistics” is ____________
a) 2 8 7 4 9 0 5 1 6 3
b) 2 7 4 9 8 0 5 1 6 3
c) 2 4 9 0 5 7 8 1 6 3
d) 2 8 7 0 5 1 6 9 4 3
View Answer

Answer: a
Explanation: The suffix array of the string statistics will be:
2 atistics
8 cs
7 ics
4 istics
9 s
0 statistics
5 stics
1 tatistics
6 tics
3 tistics

In Suffix array, we only store the indices of suffixes. So, correct option is 2 8 7 4 9 0 5 1 6 3.

3. Suffix array can be created by performing __________ traversal of a suffix tree.
a) breadth-first
b) level order
c) depth-first
d) either breadth-first or level order
View Answer

Answer: c
Explanation: A suffix tree is a trie, which contains all the suffixes of the given string as their keys and positions in the string as their values. So, we can construct a suffix array by performing the depth-first traversal of a suffix tree.

4. Suffix array is space efficient than the suffix tree.
a) True
b) Fasle
View Answer

Answer: b
Explanation: Suffix arrays are more space efficient than the suffix trees as they just store the original string and an array of integer. But working with suffix tree is faster than that of the suffix array.

5. If comparison based sorting algorithm is used construct the suffix array, then what will be time required to construct the suffix array?
a) O(nlogn)
b) O(n^2)
c) O(n^2logn)
d) O(n^2) + O(logn)
View Answer

Answer: c
Explanation: On average comparison based sorting algorithms require O(nlogn) comparisons. But comparing a suffix takes O(n). So, overall time to construct the suffix array will be O(nlogn) * O(n) = O(n^2logn).

6. What will be the suffix array of the string “engineering”?
a) 2 3 8 4 9 1 7 5 0 6 10
b) 5 0 6 1 4 9 1 7 0 2 3 8
c) 5 0 6 10 2 4 9 1 7 3 8
d) 5 0 6 10 2 3 8 4 9 1 7
View Answer

Answer: d
Explanation: Correct choice is : 5 0 6 10 2 3 8 4 9 1 7.
Because the suffix array formed will be: 5 eering 0 engineering 6 ering 10 g 2 gineering 3 ineering 8 ing 4 neering 9 ng 1 ngineering 7 ring.

7. LCP array and ______ is used to construct suffix tree.
a) Hash tree
b) Hash trie
c) Suffix array
d) Balanced tree
View Answer

Answer: c
Explanation: Suffix tree can be created using an LCP array and a suffix array. If we are given a string of length (n + 1) and its suffix array and LCP array, we can construct the suffix tree in linear time i.e in O(n) time.

8. What is the time required to locate the occurrences of a pattern P of length m in a string of length n using suffix array?
a) O(nm)
b) O(n^2)
c) O(mnlogn)
d) O(mlogn)
View Answer

Answer: d
Explanation: Suffix arrays are used to find the occurrences of a pattern in a string. Pattern of length m will require m characters to compare, so using suffix array we can find occurrences of a pattern in the string of length n in O(mlogn) time.

9. Suffix array can be created in O(nlogn) time.
a) True

b) False
View Answer

Answer: a
Explanation: Suffix array can be constructed in O(n^2logn) time using sorting algorithms but it is possible to build the suffix array in O(nlogn) time using prefix doubling.

10. Which of the following is/are advantages suffix array one suffix tree?
I. Lesser space requirement
II. Improved cache locality
III. Easy construction in linear time
a) Only I
b) All I, II and III
c) Only I and III
d) Only II and III
View Answer

Answer: b
Explanation: Advantages of the suffix array over suffix tree are : (i) Lesser space requirement (ii) Improved cache locality and (iii) Simple algorithms to construct suffix arrays in linear time.
Topic 36. Matrix

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on "Matrix".

1. What is the order of a matrix?
a) number of rows X number of columns
b) number of columns X number of rows
c) number of rows X number of rows
d) number of columns X number of columns
View Answer

Answer: a
Explanation: The order of the matrix is the number of rows X number of columns.

2. Which of the following property does not hold for matrix multiplication?
a) Associative
b) Distributive
c) Commutative
d) Additive Inverse
View Answer

Answer: c
Explanation: In matrix multiplication, AB != BA

3. How do you allocate a matrix using a single pointer in C?(r and c are the number of rows and columns respectively)
a) int *arr = malloc(r * c * sizeof(int));
b) int *arr = (int *)malloc(r * c * sizeof(int));
c) int *arr = (int *)malloc(r + c * sizeof(int));
d) int *arr = (int *)malloc(r * c * sizeof(arr));
View Answer

Answer: b
Explanation: Total number of elements in the matrix will be r*c

4. Select the code snippet which performs matrix multiplication.(a and b are the two given matrices, resultant marix is c)
a)

```
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        for (int k = 0; k < n; k++)
        {
            c[i][j] = c[i][j] + a[i][k] * b[k][j];
        }
    }
}
```

b)

```
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        for (int k = 0; k < n; k++)
        {
            c[i][j] = c[i][j] * a[i][k] * b[k][j];
        }
    }
}
```

c)

```
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        for (int k = 0; k < n; k++)
        {
            c[i][j] = c[i][j] + a[i][k] + b[k][j];
        }
    }
}
```

d)

```
for (int i = 0; i < n; i++)
{
    for (int j = 0; j < n; j++)
    {
        for (int k = 0; k < n; k++)
        {
            c[i][j] = c[i][j] + a[i][j] + b[k][j];
        }
    }
}
```

View Answer
Answer: a
Explanation: The corresponding elements from the row and column are multiplied and a cumulative sum is formed.

5. What does the following piece of code do?

```
for(int i = 0; i < row; i++)
{
    for(int j = 0; j < column; j++)
    {
        if(i == j)
            sum = sum + (array[i][j]);
    }
}
System.out.println(sum);
```

a) Normal of a matrix
b) Trace of a matrix
c) Square of a matrix
d) Transpose of a matrix
View Answer

Answer: b
Explanation: Trace of a matrix is the sum of the principal diagonal elements.

6. If row-major order is used, how is the following matrix stored in memory?
a b c
d e f
g h i
a) ihgfedcba
b) abcdefghi
c) cfibehadg
d) adgbehcfi
View Answer

Answer: b
Explanation: It starts with the first element and continues in the same row until the end of row is reached and then proceeds with the next row. C follows row-major order.

7. If column-major order is used, how is the following matrix stored in memory?
a b c
d e f

g h i
a) ihgfedcba
b) abcdefghi
c) cfibehadg
d) adgbehcfi
View Answer

Answer: d
Explanation: It starts with the first element and continues in the same column until the end of column is reached and then proceeds with the next column. Fortran follows column-major order.

8. Which of the following don't use matrices?
a) In solving linear equations
b) Image processing
c) Graph theory
d) Sorting numbers
View Answer

Answer: d
Explanation: Numbers uses arrays(1-D) for sorting not matrices(2-D arrays). Solving linear equations is a separate field in Mathematics involving matrices, Image processing stores the pixels in the form of matrices, and the graphs are represented with the help of matrices to indicate the nodes and edges.

9. Which of the following is an advantage of matrices?
a) Internal complexity
b) Searching through a matrix is complex
c) Not space efficient
d) Graph Plotting
View Answer

Answer: d
Explanation: Adjacency and Incidence Matrices are used to store vertices and edges of a graph. It is an advantage to plot graphs easily using matrices. But Time complexity of a matrix is $O(n^2)$ and sometimes the internal organization becomes tedious. They are all disadvantages of matrices.

10. Matrix A when multiplied with Matrix C gives the Identity matrix I, what is C?
a) Identity matrix
b) Inverse of A
c) Square of A
d) Transpose of A
View Answer

Answer: b
Explanation: Any square matrix when multiplied with its inverse gives the identity matrix. Note that non square matrices are not invertible.
Topic 37. Sparse Matrix

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on "Sparse Matrix".

1. Which matrix has most of the elements (not all) as Zero?
a) Identity Matrix
b) Unit Matrix
c) Sparse Matrix
d) Zero Matrix
View Answer

Answer: c
Explanation: Sparse Matrix is a matrix in which most of the elements are Zero. Identity Matrix is a matrix in which all principle diagonal elements are 1 and rest of the elements are Zero. Unit Matrix is also called Identity Matrix. Zero Matrix is a matrix in which all the elements are Zero.

2. What is the relation between Sparsity and Density of a matrix?
a) Sparsity = 1 – Density
b) Sparsity = 1 + Density
c) Sparsity = Density*Total number of elements
d) Sparsity = Density/Total number of elements
View Answer

Answer: a
Explanation: Sparsity of a matrix is equal to 1 minus Density of the matrix. The Sparsity of matrix is defined as the total

number of Zero Valued elements divided total number of elements.

3. Who coined the term Sparse Matrix?
a) Harry Markowitz
b) James Sylvester
c) Chris Messina
d) Arthur Cayley
View Answer

Answer: a
Explanation: Harry Markowitz coined the term Sparse Matrix. James Sylvester coined the term Matrix. Chris Messina coined the term Hashtag and Arthur Cayley developed the algebraic aspects of a matrix.

4. Is O(n) the Worst case Time Complexity for addition of two Sparse Matrix?
a) True
b) False
View Answer

Answer: a
Explanation: In Addition, the matrix is traversed linearly, hence it has the time complexity of O(n) where n is the number of non-zero elements in the largest matrix amongst two.

5. The matrix contains m rows and n columns. The matrix is called Sparse Matrix if ________
a) Total number of Zero elements > (m*n)/2
b) Total number of Zero elements = m + n
c) Total number of Zero elements = m/n
d) Total number of Zero elements = m-n
View Answer

Answer: a
Explanation: For matrix to be Sparse Matrix, it should contain Zero elements more than the non-zero elements. Total elements of the given matrix is m*n. So if Total number of Zero elements > (m*n)/2, then the matrix is called Sparse Matrix.

6. Which of the following is not the method to represent Sparse Matrix?
a) Dictionary of Keys
b) Linked List
c) Array
d) Heap
View Answer

Answer: d
Explanation: Heap is not used to represent Sparse Matrix while in Dictionary, rows and column numbers are used as Keys and values as Matrix entries, Linked List is used with each node of Four fields (Row, Column, Value, Next Node) (2D array is used to represent the Sparse Matrix with three fields (Row, Column, Value).

7. Is Sparse Matrix also known as Dense Matrix?
a) True
b) False
View Answer

Answer: b
Explanation: Sparse Matrix is a matrix with most of the elements as Zero elements while Dense Matrix is a matrix with most of the elements as Non-Zero element.

8. Which one of the following is a Special Sparse Matrix?
a) Band Matrix
b) Skew Matrix
c) Null matrix
d) Unit matrix
View Answer

Answer: a
Explanation: A band matrix is a sparse matrix whose non zero elements are bounded to a diagonal band, comprising the main diagonal and zero or more diagonals on either side.

9. In what way the Symmetry Sparse Matrix can be stored efficiently?
a) Heap
b) Binary tree
c) Hash table
d) Adjacency List

View Answer

Answer: b
Explanation: Since Symmetry Sparse Matrix arises as the adjacency matrix of the undirected graph. Hence it can be stored efficiently as an adjacency list.
Topic 38. Count Inversion

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Count Inversion”.

1. What does the number of inversions in an array indicate?
a) mean value of the elements of array
b) measure of how close or far the array is from being sorted
c) the distribution of values in the array
d) median value of the elements of array
View Answer

Answer: b
Explanation: The number of inversions in an array indicates how close or far the array is from being completely sorted. The array is sorted if the number of inversions are 0.

2. How many inversions does a sorted array have?
a) 0
b) 1
c) 2
d) cannot be determined
View Answer

Answer: a
Explanation: When an array is sorted then there cannot be any inversion in the array. As the necessary condition for an inversion is arr[i]>arr[j] and i<j.

3. What is the condition for two elements arr[i] and arr[j] to form an inversion?
a) arr[i]<arr[j]
b) i < j
c) arr[i] < arr[j] and i < j
d) arr[i] > arr[j] and i < j
View Answer

Answer: d
Explanation: For two elements to form an inversion the necessary condition is arr[i] > arr[j] and i < j. The number of inversions in an array indicate how close or far the array is from being completely sorted.

4. Under what condition the number of inversions in an array are maximum?
a) when the array is sorted
b) when the array is reverse sorted
c) when the array is half sorted
d) depends on the given array
View Answer

Answer: b
Explanation: Number of inversions in an array are maximum when the given array is reverse sorted. As the necessary condition for an inversion is arr[i]>arr[j] and i<j.

5. Under what condition the number of inversions in an array are minimum?
a) when the array is sorted
b) when the array is reverse sorted
c) when the array is half sorted
d) depends on the given array
View Answer

Answer: a
Explanation: Number of inversions in an array are minimum when the given array is sorted. As the necessary condition for an inversion is arr[i]>arr[j] and i<j.

6. How many inversions are there in the array arr = {1,5,4,2,3}?
a) 0
b) 3
c) 4
d) 5
View Answer

Answer: d
Explanation: The necessary condition for an inversion is arr[i]>arr[j] and i<j. So there are 5 inversions in the array.

7. Which of the following form inversion in the array arr = {1,5,4,2}?
a) (5,4), (5,2)
b) (5,4), (5,2), (4,2)
c) (1,5), (1,4), (1,2)
d) (1,5)
View Answer

Answer: b
Explanation: The necessary condition for an inversion is arr[i]>arr[j] and i<j. So there are 3 inversions in the array. These are (5,4), (5,2), (4,2).

8. Choose the correct function from the following which determines the number of inversions in an array?
a)

```
int InvCount(int arr[], int n)
{
 int count = 0;
 for (int i = 0; i < n - 1; i++)
  for (int j = i ; j < n; j++)
   if (arr[i] >= arr[j])
    count++;

 return count;
}
```

b)

```
int InvCount(int arr[], int n)
{
 int count = 0;
 for (int i = 0; i < n - 1; i++)
  for (int j = i + 1; j < n; j++)
   if (arr[i] > arr[j])
    count++;

 return count;
}
```

c)

```
int InvCount(int arr[], int n)
{
 int count = 0;
 for (int i = 0; i < n - 1; i++)
  for (int j = i + 1; j < n; j++)
   if (arr[i] > arr[j])
    count++;

 return count + 1;
}
```

d)

```
int InvCount(int arr[], int n)
{
 int count = 0;
 for (int i = 0; i < n - 1; i++)
  for (int j = i + 1; j < n; j++)
   if (arr[i] < arr[j])
    count++;

 return count + 1;
}
```

View Answer
Answer: b
Explanation: To determine the number of inversions we apply a nested loop and compare the value of each element with all the elements present after it. Then the count of number of inversions is counted and returned to the main function.

9. What is the time complexity of the following code that determines the number of inversions in an array?

```
int InvCount(int arr[], int n)
{
 int count = 0;
 for (int i = 0; i < n - 1; i++)
```

```
  for (int j = i + 1; j < n; j++)
   if (arr[i] > arr[j])
    count++;

 return count;
}
```

a) O(n)
b) O(n log n)
c) O(n^2)
d) O(log n)
View Answer

Answer: c
Explanation: The time complexity of the given code is O(n2). It is due to the presence of nested loop.

10. The time complexity of the code that determines the number of inversions in an array using merge sort is lesser than that of the code that uses loops for the same purpose.
a) true
b) false
View Answer

Answer: a
Explanation: The time complexity of the code that determines the number of inversions in an array using merge sort is O(n log n) which is lesser than the time complexity taken by the code that uses loops.

11. What is the time complexity of the code that uses merge sort for determining the number of inversions in an array?
a) O(n^2)
b) O(n)
c) O(log n)
d) O(n log n)
View Answer

Answer: d
Explanation: The code of merge sort is slightly modified in order to calculate the number of inversions in an array. So the time complexity of merge sort remains unaffected and hence the time complexity is O(n log n).

12. What is the time complexity of the code that uses self balancing BST for determining the number of inversions in an array?
a) O(n^2)
b) O(n)
c) O(log n)
d) O(n log n)
View Answer

Answer: d
Explanation: When a self balancing BST like an AVL tree is used to calculate the number of inversions in an array then the time complexity is O(n log n) as AVL insert takes O(log n) time.

13. The time complexity of the code that determines the number of inversions in an array using self balancing BST is lesser than that of the code that uses loops for the same purpose.
a) true
b) false
View Answer

Answer: a
Explanation: The time complexity of the code that determines the number of inversions in an array using self balancing BST is O(n log n) which is lesser than the time complexity taken by the code that uses loops.

14. What is the space complexity of the code that uses merge sort for determining the number of inversions in an array?
a) O(n)
b) O(log n)
c) O(1)
d) O(n log n)
View Answer

Answer: a
Explanation: The space complexity required by the code will be O(n). It is the same as the space complexity of the code of standard merge sort.
Topic 39. Rotation Array Operation

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Rotation Array Operation”.

1. What will be the resulting array after rotating arr[]={1, 2, 3, 4, 5} by 2?
a) 2, 1, 3, 4, 5
b) 3, 4, 5, 1, 2
c) 4, 5, 1, 2, 3
d) 1, 2, 3, 5, 4
View Answer

Answer: b
Explanation: When the given array is rotated by 2 then the resulting array will be
Rotation 1: {2,3,4,5,1}
Rotation 2: {3,4,5,1,2}.
Thus, the final array is {3,4,5,1,2}.

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

```
#include <iostream>
using namespace std;
int main()
{
    int arr[] = {1,2,3,4,5,6};
    int n = sizeof(arr)/sizeof(arr[0]);
    int d=4;
    int temp[10];

    for(int i=0;i<d;i++)
    temp[i]=arr[i];

    int j=0;
    for(int i=d;i<n;i++,j++)
    arr[j]=arr[i];

    int k=0;
    for(int i=n-d;i<n;i++,k++)
    arr[i]=temp[k];

    for(int i=0;i<n;i++)
    cout<<arr[i]<<" ";
    return 0;
}
```

a) 5 6 1 2 3 4
b) 6 5 4 3 1 2
c) 3 4 5 6 1 2
d) error
View Answer

Answer: a
Explanation: The given code rotates the given array by 4. It does so by using an array temp[] which stores the first d elements and then shift them to the end of the array. So the output will be 5 6 1 2 3 4.

3. What will be the time complexity of the following code?

```
#include <iostream>
using namespace std;
int main()
{
    int arr[] = {1,2,3,4,5,6};
    int n = sizeof(arr)/sizeof(arr[0]);
    int d=4;
    int temp[10];

    for(int i=0;i<d;i++)
    temp[i]=arr[i];

    int j=0;
    for(int i=d;i<n;i++,j++)
    arr[j]=arr[i];

    int k=0;
    for(int i=n-d;i<n;i++,k++)
    arr[i]=temp[k];

    for(int i=0;i<n;i++)
    cout<<arr[i]<<" ";
    return 0;
}
```

a) O(d)
b) O(n)
c) O(n^2)
d) O(n*d)
View Answer

Answer: b
Explanation: The given code rotates an input array by d. The longest loop in the code takes n iterations so the time complexity will be O(n).

4. What will be the auxiliary space complexity of the following code?

```
#include <iostream>
using namespace std;
int main()
{
    int arr[] = {1,2,3,4,5,6};
    int n = sizeof(arr)/sizeof(arr[0]);
    int d=4;
    int temp[10];

    for(int i=0;i<d;i++)
    temp[i]=arr[i];

    int j=0;
    for(int i=d;i<n;i++,j++)
    arr[j]=arr[i];

    int k=0;
    for(int i=n-d;i<n;i++,k++)
    arr[i]=temp[k];

    for(int i=0;i<n;i++)
    cout<<arr[i]<<" ";
    return 0;
}
```

a) O(1)
b) O(n)
c) O(d)
d) O(n*d)
View Answer

Answer: c
Explanation: The given code rotates an input array by d. It does so by using an auxiliary array temp[] which stores first d elements of the original array. So the auxiliary space complexity will be O(d).

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

```
#include <bits/stdc++.h>
using namespace std;

void func1(int arr[], int n)
{
 int k = arr[0], i;
 for (i = 0; i < n - 1; i++)
  arr[i] = arr[i + 1];

 arr[i] = k;
}

void func(int arr[], int d, int n)
{
 for (int i = 0; i < d; i++)
  func1(arr, n);
}

void printArray(int arr[], int n)
{
 for (int i = 0; i < n; i++)
  cout << arr[i] << " ";
}

int main()
{
 int arr[] = { 1, 2, 3, 4, 5};
 int n = sizeof(arr) / sizeof(arr[0]);


 func(arr, 3, n);
 printArray(arr, n);
 return 0;
}
```

a) 4 5 1 2 3
b) 3 4 5 1 2
c) 5 4 3 1 2
d) error
View Answer

Answer: a
Explanation: The given code rotates the input array by 3. It does so by rotating the elements one by one until the desired rotation is achieved. So the output will be 4 5 1 2 3.

6. What will be the time complexity of the following code?

```
#include <bits/stdc++.h>
using namespace std;
void func1(int arr[], int n)
{
 int k = arr[0], i;
 for (i = 0; i < n - 1; i++)
  arr[i] = arr[i + 1];

 arr[i] = k;
}

void func(int arr[], int d, int n)
{
 for (int i = 0; i < d; i++)
  func1(arr, n);
}

void printArray(int arr[], int n)
{
 for (int i = 0; i < n; i++)
  cout << arr[i] << " ";
}

int main()
{
 int arr[] = { 1, 2, 3, 4, 5};
 int n = sizeof(arr) / sizeof(arr[0]);

    int d = 3;
 func(arr, d, n);
 printArray(arr, n);

 return 0;
}
```

a) O(n*d)
b) O(n)
c) O(d)
d) O(n^2)
View Answer

Answer: a
Explanation: The given code rotates the input array by d. It does so by rotating the elements one by one until the desired rotation is achieved. Each element takes O(n) time for rotation and there are d such elements in the array. So the time complexity would be O(n*d).

7. What will be the auxiliary space complexity of the following code?

```
#include <bits/stdc++.h>
using namespace std;
void func1(int arr[], int n)
{
 int k = arr[0], i;
 for (i = 0; i < n - 1; i++)
  arr[i] = arr[i + 1];

 arr[i] = k;
}

void func(int arr[], int d, int n)
{
 for (int i = 0; i < d; i++)
  func1(arr, n);
}
```

```
void printArray(int arr[], int n)
{
 for (int i = 0; i < n; i++)
  cout << arr[i] << " ";
}

int main()
{
 int arr[] = { 1, 2, 3, 4, 5};
 int n = sizeof(arr) / sizeof(arr[0]);

    int d = 3;
 func(arr, d, n);
 printArray(arr, n);

 return 0;
}
```

a) O(1)
b) O(n)
c) O(d)
d) O(n*d)
View Answer

Answer: a
Explanation: The given code rotates the input array by d. It does so by rotating the elements one by one until the desired rotation is achieved. It does not require any auxiliary array for this purpose. So the auxiliary space complexity will be O(1).

8. To rotate an array by using the algorithm of rotating its elements one by one is an in place algorithm.
a) true
b) false
View Answer

Answer: a
Explanation: The auxiliary space requirement of the mentioned algorithm is O(1). So it qualifies to be an in place algorithm.

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

```
#include <bits/stdc++.h>
using namespace std;
void func1(int arr[], int left, int right)
{
 while (left < right)
 {
  int temp = arr[left];
  arr[left] = arr[right];
  arr[right] = temp;
  left++;
  right--;
 }
}

void func(int arr[], int d, int n)
{
 func1(arr, 0, d-1);
 func1(arr, d, n-1);
 func1(arr, 0, n-1);
}

void printArray(int arr[], int size)
{
 for (int i = 0; i < size; i++)
 cout << arr[i] << " ";
}

int main()
{
 int arr[] = {1, 2, 3, 4, 5};
 int n = sizeof(arr)/sizeof(arr[0]);
 int d = 2;
 func(arr, d, n);
 printArray(arr, n);

 return 0;
}
```

a) 3 2 1 4 5
b) 3 4 5 1 2
c) 5 4 3 2 1
d) error
View Answer

Answer: b
Explanation: The given code rotates the input array by 2. It does so by applying a reversal algorithm to different segments of the array. First d elements and the rest of the array is reversed individually. Then the whole array is reversed which gives us the desired rotated array. So the output will be 3 4 5 1 2.

10. What will be the auxiliary space complexity of the code to rotate an array by using the reversal algorithm (d = number of rotations)?
a) O(1)
b) O(n)
c) O(d)
d) O(n*d)
View Answer

Answer: a
Explanation: The reversal algorithm for rotating an array does not require any auxiliary space. So the auxiliary space complexity will be O(1).

11. Which of the following is the predefined function for array reversal in C++?
a) rotate()
b) arr_rotate()
c) array_rotate()
d) rot()
View Answer

Answer: a
Explanation: The predefined function for rotating an array is rotate() in C++. It is defined under the library algorithm and requires 3 arguments.

12. How many arguments are required by the predefined function rotate() in C++?
a) 1
b) 2
c) 3
d) 4
View Answer

Answer: c
Explanation: The predefined function for rotating an array is rotate() in C++ which comes under the library called an algorithm. It requires 3 arguments.

13. Predefined function rotate() in C++ is available under which header file?
a) math
b) stdio
c) stdlib
d) algorithm
View Answer

Answer: d
Explanation: The predefined function for rotating an array is rotate() in C++ which comes under the library called algorithm. It requires 3 arguments the first being the pointer to the starting index of the array and the last being the pointer to the last index of the array. The middle argument is the pointer to the element that becomes the first element in the rotated array.

14. Which of the following algorithm to rotate an array has the maximum time complexity?
a) rotate elements one by one
b) juggling algorithm
c) reversal algorithm
d) using a temporary array
View Answer

Answer: a
Explanation: The maximum time complexity is required by the algorithm that rotates elements one by one. It requires O(n*d) time.

15. What is the time complexity of the juggling algorithm to rotate an array?
a) O(1)
b) O(n)
c) O(d)
d) O(n*d)
View Answer

Answer: b

Explanation: Time complexity of juggling algorithm is O(n). Its auxiliary space complexity is O(1).

16. Reversal algorithm and juggling algorithm for array rotation have the same time complexity.
a) True
b) False
View Answer

Answer: a
Explanation: Time complexity of juggling algorithm is O(n) which like that of reversal algorithm. They also have the same space complexity.
Topic 40. Reversal Array Operation

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Reversal Array Operation”.

1. What will be the resulting array after reversing arr[]={3,5,4,2}?
a) 2,3,5,4
b) 4,2,3,5
c) 5,4,2,3
d) 2,4,5,3
View Answer

Answer: d
Explanation: The resulting array upon reversing after reversal is arr[]={2,4,5,3}. We can implement an algorithm for this purpose in various possible ways.

2. How many swaps are required for reversing an array having n elements where n is an odd number?
a) (n-1) / 2
b) n/2
c) (n/2) – 1
d) (n+1)/2
View Answer

Answer: a
Explanation: The number of swaps required for an odd element and an even element array is different because in an odd element array the position of the middle element does not need to be changed. So the number of swaps will be (n-1) / 2.

3. How many swaps are required for reversing an array having n elements where n is an even number?
a) (n-1) / 2
b) n/2
c) (n/2) – 1
d) (n+1)/2
View Answer

Answer: b
Explanation: The number of swaps required for an odd element and an even element array is different because in an odd element array the position of the middle element does not need to be changed. So number of swaps required will be n/2.

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

```
#include <bits/stdc++.h>
using namespace std;

void func(int arr[], int left, int right)
{
 while (left < right)
 {
  int temp = arr[left];
  arr[left] = arr[right];
  arr[right] = temp;
  left++;
  right--;
 }

}

void printArray(int arr[], int size)
{
   for (int i = 0; i < size; i++)
   cout << arr[i] << " ";
}

int main()
{
 int arr[] = {1,4,3,5};
 int n = sizeof(arr) / sizeof(arr[0]);
 func(arr, 0, n-1);
 printArray(arr, n);
 return 0;
}
```

a) 5 1 4 3
b) 3 5 1 4
c) 5 3 4 1
d) error
View Answer

Answer: c
Explanation: The given code reverses the input array and then prints the resulting array. So the output of the given code will be 5 3 4 1.

5. What will be the time complexity of the following code?

```
#include <bits/stdc++.h>
using namespace std;
void func(int arr[], int left, int right)
{
    while (left < right)
 {
  int temp = arr[left];
  arr[left] = arr[right];
  arr[right] = temp;
  left++;
  right--;
 }

}

void printArray(int arr[], int size)
{
   for (int i = 0; i < size; i++)
   cout << arr[i] << " ";
}

int main()
{
 int arr[] = {1,4,3,5};
 int n = sizeof(arr) / sizeof(arr[0]);
 func(arr, 0, n-1);
 printArray(arr, n);
 return 0;
}
```

a) O(n)
b) O(log n)
c) O(1)
d) O(n log n)
View Answer

Answer: a
Explanation: The given code reverses the input array and then prints the resulting array. So the time complexity of the given code will linearly vary with the number of elements in the array and thus the time complexity will be O(n).

6. What will be the auxiliary space requirement of the following code?

```
#include <bits/stdc++.h>
using namespace std;
void func(int arr[], int left, int right)
{
 while (left < right)
 {
  int temp = arr[left];
  arr[left] = arr[right];
  arr[right] = temp;
  left++;
  right--;
 }

}

void printArray(int arr[], int size)
{
   for (int i = 0; i < size; i++)
   cout << arr[i] << " ";
}

int main()
```

```
{
 int arr[] = {1,4,3,5};
 int n = sizeof(arr) / sizeof(arr[0]);
 func(arr, 0, n-1);
 printArray(arr, n);
 return 0;
}
```

a) O(1)
b) O(n)
c) O(log n)
d) O(n log n)
View Answer

Answer: a
Explanation: The given code reverses the input array and then prints the resulting array. The given code does not use any extra array to complete this task thus the auxiliary space requirement is O(1).

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

```
#include <bits/stdc++.h>
using namespace std;

void func(int arr[], int left, int right)
{
   if (left >= right)
   return;

   int temp = arr[left];
   arr[left] = arr[right];
   arr[right] = temp;

   func(arr, left + 1, right - 1);
}

void printArray(int arr[], int size)
{
   for (int i = 0; i < size; i++)
   cout << arr[i] << " ";
}

int main()
{
 int arr[] = {1,2,3,4};
 int n = sizeof(arr) / sizeof(arr[0]);
 func(arr, 0, n-1);
 printArray(arr, n);
 return 0;
}
```

a) 1 2 3 4
b) 4 3 2 1
c) 1 4 2 3
d) 4 1 2 3
View Answer

Answer: b
Explanation: The given code reverses the original array and prints the resulting array. Recursive function is used to reverse the array.

8. What will be the time complexity of the following code?

```
#include <bits/stdc++.h>
using namespace std;
void func(int arr[], int left, int right)
{
   if (left >= right)
   return;

   int temp = arr[left];
   arr[left] = arr[right];
   arr[right] = temp;

   func(arr, left + 1, right - 1);
}

void printArray(int arr[], int size)
{
   for (int i = 0; i < size; i++)
   cout << arr[i] << " ";
}
```

```
int main()
{
 int arr[] = {1,2,3,4};
 int n = sizeof(arr) / sizeof(arr[0]);
 func(arr, 0, n-1);
 printArray(arr, n);
 return 0;
}
```

a) O(1)
b) O(n)
c) O(log n)
d) O(n log n)
View Answer

Answer: b
Explanation: The given code reverses the original array and prints the resulting array. The number of swaps is proportional to the number of elements in the array so it requires a time complexity of O(n).

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

```
#include <bits/stdc++.h>
using namespace std;
void func(int a[], int n, int k)
{
 if (k <= n)
 {
  for (int i = 0; i < k/2; i++)
  swap(a[i], a[k-i-1]);
 }

}
int main()
{
 int a[] = {1, 2, 3, 4, 5};
 int n = sizeof(a) / sizeof(int), k = 3;
 func(a, n, k);
 for (int i = 0; i < n; ++i)
  cout << a[i]<<" ";
 return 0;
}
```

a) 3 2 1 4 5
b) 5 4 3 2 1
c) 1 2 5 4 3
d) error
View Answer

Answer: a
Explanation: The given code reverses only a specified segment of the input array. As the value of k is given to be 3 in the code thus only the first three elements of the array will be reversed.

10. What will be the time complexity of the following code?

```
#include <bits/stdc++.h>
using namespace std;

void func(int a[], int n, int k)
{
 if (k <= n)
 {
  for (int i = 0; i < k/2; i++)
  swap(a[i], a[k-i-1]);
 }

}
int main()
{
 int a[] = {1, 2, 3, 4, 5};
 int n = sizeof(a) / sizeof(int), k = 3;
 func(a, n, k);
 for (int i = 0; i < n; ++i)
  cout << a[i]<<" ";
 return 0;
}
```

a) O(k)
b) O(n)
c) O(k log k)

d) O(n log n)
View Answer

Answer: a
Explanation: The given code reverses only a specified segment of the input array. This segment is decided by the value of k so the time complexity of the code will be O(k).

11. When array reversal and rotation is applied to the same array then the output produced will also be the same every time.
a) true
b) false
View Answer

Answer: b
Explanation: Array rotation and array reversal are different operations and thus they give different outputs when applied to the same array.

12. Which of the following is the predefined function for array reversal in C++ ?
a) reverse()
b) arr_reverse()
c) array_reverse()
d) rev()
View Answer

Answer: a
Explanation: The predefined function for reversing an array is reverse() in C++. It is defined under the library algorithm and requires 2 arguments.

13. Which of the following is the predefined function for array reversal in javascript?
a) reverse()
b) arr_reverse()
c) array_reverse()
d) rev()
View Answer

Answer: a
Explanation: The predefined function for reversing an array is reverse() in javascript. It does not requires any argument.

14. Predefined function reverse() in C++ is available under which header file?
a) math
b) stdio
c) stdlib
d) algorithm
View Answer

Answer: d
Explanation: The predefined function for reversing an array is reverse() in C++ which comes under the library called an algorithm. It requires 2 arguments the first being the pointer to the starting index of the array and the second being the pointer to the last index of the array.
Topic 41. Number of Jumps to Reach End-array Operation

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Number of Jumps to Reach End-array Operation”.

1. What will be the minimum number of jumps required to reach the end of the array arr[] = {1,3,6,3,6,8,5}?
a) 1
b) 2
c) 3
d) not possible to reach the end
View Answer

Answer: c
Explanation: Each element of the array represents the maximum number of steps that can be taken forward from that element. If the first element is 0 then it is not possible to reach the end.

2. What will be the minimum number of jumps required to reach the end of the array arr[] ={0,1,3,6,3,6,8,5}?
a) 1
b) 2
c) 3
d) not possible to reach the end

View Answer

Answer: d
Explanation: Each element of the array represents the maximum number of steps that can be taken forward from that element. So as the first element here is 0 so we cannot move any further from the first element. Thus, it is not possible to reach the end of the array.

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

```
#include <bits/stdc++.h>
using namespace std;

int func(int arr[], int s, int e)
{
   if (s == e)
 return 0;
   if (arr[s] == 0)
 return INT_MAX;

int min = INT_MAX;
for (int i = s + 1; i <= e && i <= s + arr[s]; i++)
{
 int jumps = func(arr, i, e);
 if(jumps != INT_MAX && jumps + 1 < min)
  min = jumps + 1;
}
return min;
}

int main()
{
 int arr[] = {1, 3, 6, 3, 8, 5};
 int n = sizeof(arr)/sizeof(arr[0]);
 cout << func(arr, 0, n-1);
 return 0;
}
```

a) 1
b) 2
c) 3
d) error
View Answer

Answer: c
Explanation: The given code finds the minimum number of steps required to reach the end of the array by using recursion. So the output will be 3.

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

```
#include <bits/stdc++.h>
using namespace std;

int min(int x, int y)
{ return (x < y)? x: y; }

int func(int arr[], int n)
{

 int *jump = new int[n];
 int i, j;

 if (n == 0 || arr[0] == 0)
  return INT_MAX;

 jump[0] = 0;

 for (i = 1; i < n; i++)
 {
  jump[i] = INT_MAX;
  for (j = 0; j < i; j++)
  {
   if (i <= j + arr[j] && jump[j] != INT_MAX)
   {
    jump[i] = min(jump[i], jump[j] + 1);
    break;
   }
  }
 }
 return jump[n-1];
}

int main()
```

```
{
 int arr[] = {1, 3, 6, 1, 9,7};
 int size = sizeof(arr)/sizeof(int);
 cout<< func(arr,size);
 return 0;
}
```

a) 1
b) 2
c) 3
d) error
View Answer

Answer: c
Explanation: The given code finds the minimum number of steps required to reach the end of the array by using dynamic programming. So the output will be 3.

5. What will be the time complexity of the following code?

```
#include <bits/stdc++.h>
using namespace std;

int min(int x, int y)
{ return (x < y)? x: y; }

int func(int arr[], int n)
{

 int *jump = new int[n];
 int i, j;

 if (n == 0 || arr[0] == 0)
  return INT_MAX;

 jump[0] = 0;

 for (i = 1; i < n; i++)
 {
  jump[i] = INT_MAX;
  for (j = 0; j < i; j++)
  {
   if (i <= j + arr[j] && jumps[j] != INT_MAX)
   {
    jump[i] = min(jump[i], jump[j] + 1);
    break;
   }
  }
 }
 return jump[n-1];
}

int main()
{
 int arr[] = {1, 3, 6, 1, 9,7};
 int size = sizeof(arr)/sizeof(int);
 cout<< func(arr,size);
 return 0;
}
```

a) O(n log n)
b) O(n)
c) $O(n^{1/2})$
d) $O(n^2)$
View Answer

Answer: d
Explanation: The given code finds the minimum number of steps required to reach the end of an array by using dynamic programming. As there is a nested loop in the code so the time complexity will be $O(n^2)$.

6. What will be the minimum number of jumps required to reach the end of the array arr[] = {1,2,0,0,3,6,8,5}?
a) 1
b) 2
c) 3
d) not possible to reach the end
View Answer

Answer: d
Explanation: Each element of the array represents the maximum number of steps that can be taken forward from that element. So we cannot move any further after reaching the second element hence it is impossible to reach the end of the

array.

7. It is not possible to find the minimum number of steps to reach the end of an array in linear time.
a) true
b) false
View Answer

Answer: b
Explanation: It is possible to find the minimum number of steps to reach the end of an array in O(n) time complexity. So it is the fastest possible method of finding the minimum number of steps to reach the end of an array.

8. In how many different ways we can reach the end of the array arr[]={1,3,5,8,9}?
a) 1
b) 2
c) 3
d) 4
View Answer

Answer: d
Explanation: There are 4 possible ways in which we can reach the end of the array. The possible paths are – 1->3->5->8->9, 1->3->5->9, 1->3->8->9, 1->3->9.

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

```
#include <bits/stdc++.h>
using namespace std;

void func(int arr[], int n)
{
 int count[n];
 memset(count, 0, sizeof(count));

 for (int i=n-2; i>=0; i--)
 {
  if (arr[i] >= n - i - 1)
   count[i]++;

  for (int j=i+1; j < n-1 && j <= arr[i] + i; j++)

   if (count[j] != -1)
    count[i] += count[j];

  if (count[i] == 0)
   count[i] = -1;
 }

 for (int i=0; i<n; i++)
  cout << count[i] << " ";
}

int main()
{
 int arr[] = {1, 3, 5, 8, 9};
 int n = sizeof(arr) / sizeof(arr[0]);
 func(arr, n);
 return 0;
}
```

a) 3
b) 4
c) 4 4 2 1 0
d) 4 2 2 0 1
View Answer

Answer: c
Explanation: The given code finds the number of possible ways to reach the end of an array from each element. So the output will be 4 4 2 1 0.

10. What will be the worst case time complexity of the following code?

```
#include <bits/stdc++.h>
using namespace std;

void func(int arr[], int n)
{
 int count[n];
 memset(count, 0, sizeof(count));

 for (int i=n-2; i>=0; i--)
 {
  if (arr[i] >= n - i - 1)
   count[i]++;

  for (int j=i+1; j < n-1 && j <= arr[i] + i; j++)

   if (count[j] != -1)
    count[i] += count[j];

  if (count[i] == 0)
   count[i] = -1;
 }

 for (int i=0; i<n; i++)
  cout << count[i] << " ";
}


int main()
{
 int arr[] = {1, 3, 5, 8, 9};
 int n = sizeof(arr) / sizeof(arr[0]);
 func(arr, n);
 return 0;
}
```

a) O($n^{1/2}$)
b) O(n)
c) O($n^{3/2}$)
d) O(n^2)
View Answer

Answer: d
Explanation: The given code finds the number of possible ways to reach the end of an array from each element. By observing the nested loop in the code we can say that the worst case time complexity will be O(n^2).

11. : It is not possible to reach the end of an array if starting element of the array is 0.
a) true
b) false
View Answer

Answer: a
Explanation: If the first element of an array is 0 then it is not possible to reach the end. However, if 0 is present at other positions then we may/may not be able to reach the end.

12. What is the minimum possible time complexity to find the number of steps to reach the end of an array?
a) O(n)
b) O(n^2)
c) O($n^{3/2}$)
d) O(1)
View Answer

Answer: a
Explanation: The minimum possible time complexity to reach the end of an array is O(n). So a linear time complexity is possible.
Topic 42. Skip List

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Skip List”.

1. What is a skip list?
a) a linkedlist with size value in nodes
b) a linkedlist that allows faster search within an ordered sequence
c) a linkedlist that allows slower search within an ordered sequence
d) a tree which is in the form of linked list
View Answer

Answer: b
Explanation: It is a datastructure, which can make search in sorted linked list faster in the same way as binary search tree and sorted array (using binary search) are faster.

2. Consider the 2-level skip list

How to access 38?
a) travel 20-30-35-38

b) travel 20-30-40-38
c) travel 20-38
d) travel 20-40-38
View Answer

Answer: a
Explanation: Let us call the nodes 20, 30, 40 as top lines and the nodes between them as normal lines. the advantage of skip lists is we can skip all the elements between the top line elements as required.

3. Skip lists are similar to which of the following datastructure?
a) stack
b) heap
c) binary search tree
d) balanced binary search tree
View Answer

Answer: d
Explanation: Skip lists have the same asymptotic time complexity as balanced binary search tree. For a Balanced Binary Search Tree, we skip almost half of the nodes after one comparison with root element. The same thing done in the skip lists. Hence skip lists are similar to balanced Binary search trees.

4. What is the time complexity improvement of skip lists from linked lists in insertion and deletion?
a) O(n) to O(logn) where n is number of elements
b) O(n) to O(1) where n is number of elements
c) no change
d) O(n) to O(n^2) where n is number of elements
View Answer

Answer: a
Explanation: In Skip list we skip some of the elements by adding more layers. In this the skip list resembles balanced binary search trees. Thus we can change the time complexity from O (n) to O (logn)

5. To which datastructure are skip lists similar to in terms of time complexities in worst and best cases?
a) balanced binary search trees
b) binary search trees
c) binary trees
d) linked lists
View Answer

Answer: a
Explanation: Skip lists are similar to any randomly built binary search tree. a BST is balanced because to avoid skew tree formations in case of sequential input and hence achieve O(logn) in all 3 cases. now skip lists can gurantee that O(logn) complexity for any input.

6. The nodes in a skip list may have many forward references. their number is determined
a) probabilistically
b) randomly
c) sequentially
d) orthogonally
View Answer

Answer: a
Explanation: The number of forward references are determined probabilistically, that is why skip list is a probabilistic algorithm.

7. Are the below statements true about skiplists?
In a sorted set of elements skip lists can implement the below operations
i.given a element find closest element to the given value in the sorted set in O(logn)
ii.find the number of elements in the set whose values fall a given range in O(logn)
a) true
b) false
View Answer

Answer: a
Explanation: To achieve above operations augment with few additional stuff like partial counts.

8. How to maintain multi-level skip list properties when insertions and deletions are done?
a) design each level of a multi-level skip list with varied probabilities
b) that cannot be maintained
c) rebalancing of lists

d) reconstruction
View Answer

Answer: a
Explanation: For example consider a 2 level skip list. the level-2 skip list can skip one node on a average and at some places may skip 2 nodes, depending on probabilities. this ensures O(logn).

9. Is a skip list like balanced tree?
a) true
b) false
View Answer

Answer: a
Explanation: Skip list behaves as a balanced tree with high probability and can be commented as such because nodes with different heights are mixed up evenly.

10. What is indexed skip list?
a) it stores width of link in place of element
b) it stores index values
c) array based linked list
d) indexed tree
View Answer

Answer: a
Explanation: The width is defined as number of bottom layer links that are being traversed by each of higher layer elements. e.g: for a level-2 skip lists, all level-1 nodes have 1 as width, for level-2 width will be 2.
Topic 43. Self Organizing List

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Self Organizing List”.

1. The self organizing list improves the efficiency of _______
a) binary search
b) jump search
c) sublist search
d) linear search
View Answer

Answer: d
Explanation: Linear search in a linked list has time complexity O(n). To improve the efficiency of the linear search the self organizing list is used. A self-organizing list improves the efficiency of linear search by moving more frequently accessed elements towards the head of the list.

2. Which of the following is true about the Move-To-Front Method for rearranging nodes?
a) node with highest access count is moved to head of the list
b) requires extra storage
c) may over-reward infrequently accessed nodes
d) requires a counter for each node
View Answer

Answer: c
Explanation: In Move-To-front Method the element which is searched is moved to the head of the list. And if a node is searched even once, it is moved to the head of the list and given maximum priority even if it is not going to be accessed frequently in the future. Such a situation is referred to as over-rewarding.

3. What technique is used in Transpose method?
a) searched node is swapped with its predecessor
b) node with highest access count is moved to head of the list
c) searched node is swapped with the head of list
d) searched nodes are rearranged based on their proximity to the head node
View Answer

Answer: a
Explanation: In Transpose method, if any node is searched, it is swapped with the node in front unless it is the head of the list. So, in Transpose method searched node is swapped with its predecessor.

4. The worst case running time of a linear search on the self organizing list is ____
a) O(1)
b) O(logn)
c) O(n)
d) O(n^2)

View Answer

Answer: c
Explanation: Worst case occurs when the element is located at the very end of list. So n comparisons must be made to the locate element. So the worst case running time of linear search on self organizing list is O(n).

5. Which of the following data structure is preferred to have lesser search time when the list size is small?
a) search tree
b) sorted list
c) self organizing list
d) linked list
View Answer

Answer: c
Explanation: Self-organizing list is easy and simple to implement than search tree and it requires no additional space. So using self organizing list is preferred when list size is small.

6. In _____________ method, whenever a node is accessed, it might move to the head of the list if its number of accesses becomes greater than the records preceding it.
a) least recently used
b) count
c) traspose
d) exchange
View Answer

Answer: b
Explanation: In the count method, the number of times a node was accessed is counted and is stored in a counter variable associated with each node. Then the nodes are arranged in descending order based on their access counts. And the node with highest access count is head of the list.

7. Symbol tables during compilation of program is efficiently implemented using __________
a) a singly linked list
b) a doubly linked list
c) a self organizing list
d) an array
View Answer

Answer: c
Explanation: Self organizing list allows fast sequential search and it is simple to implement and requires no extra storage. Self-organizing list is used to implement the symbol table.

8. Which of the following method performs poorly when elements are accessed in sequential order?
a) count method
b) move to front method
c) transpose meth
d) ordering method
View Answer

Answer: b
Explanation: Move-to-front method performs poorly when the elements are accessed in sequential order, especially if that sequential order is then repeated multiple times.

9. The self organizing list improves _____
a) average access time
b) insertion
c) deletion
d) binary search
View Answer

Answer: a
Explanation: The self-organizing list rearranges the nodes based on the access probabilities of the nodes. So the required elements can be located efficiently. Therefore, self-organizing list is mainly used to improve the average access time.

10. Which of the following is not the rearranging method used to implement self-organizing lists?
a) count method
b) move to front method
c) ordering method
d) least frequently used
View Answer

Answer: d
Explanation: Least frequently used is a buffer replacement policy, while other three are methods to reorder the nodes in the self-organizing lists based on their access probability.
Topic 44. Xor Linked List

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Xor Linked List”.

1. What is xor linked list?
a) uses of bitwise XOR operation to decrease storage requirements for doubly linked lists
b) uses of bitwise XOR operation to decrease storage requirements for linked lists
c) uses of bitwise operations to decrease storage requirements for doubly linked lists
d) just another form of linked list
View Answer

Answer: a
Explanation: Why we use bitwise XOR operation is to decrease storage requirements for doubly linked lists.

2. What does a xor linked list have?
a) every node stores the XOR of addresses of previous and next nodes
b) actuall memory address of next node
c) every node stores the XOR of addresses of previous and next two nodes
d) every node stores xor 0 and the current node address
View Answer

Answer: a
Explanation: Every node stores the XOR of addresses.

3. What does first and last nodes of a xor linked lists contain ? (let address of first and last be A and B)
a) NULL xor A and B xor NULL
b) NULL and NULL
c) A and B
d) NULL xor A and B
View Answer

Answer: a
Explanation: NULL xor A and B xor NULL.

4. Which of the following is an advantage of XOR list?
a) Almost of debugging tools cannot follow the XOR chain, making debugging difficult
b) You need to remember the address of the previously accessed node in order to calculate the next node’s address
c) In some contexts XOR of pointers is not defined
d) XOR list decreases the space requirement in doubly linked list
View Answer

Answer: d
Explanation: XOR linked list stores the address of previous and next nodes by performing XOR operations. It requires single pointer to store both XOR address of next and previous nodes. Thus it reduces space. It is an advantage. But the main disadvantages are debugging tools cannot follow XOR chain, previous node address must be remembered to get next nodes and pointers are not defined accurately.

5. Which of the following is not the properties of XOR lists?
a) XX = 0
b) X0 = X
c) (XY)Z = X(YZ)
d) X0 = 1
View Answer

Answer: d
Explanation: The important properties of XOR lists are XX=0, X0=X and (XY)Z = X(YZ).

6. Which of the following statements are true?
i) practical application of XOR linked lists are in environments with limited space requirements, such as embedded devices.
ii)xor lists are not suitable because most garbage collectors will fail to work properly with classes or structures that don’t contain literal pointers
iii)in order to calculate the address of the next node you need to remember the address of the previous node
iv)xor lists are much efficient than single, doubly linked lists and arrays
a) i, ii, iii, iv
b) i, ii, iii
c) i, ii
d) i

View Answer

Answer: b
Explanation: Xor lists requires same time for most of the operations as arrays would require.

7. What's wrong with this code which returns xor of two nodes address ?

```
//struct is common userdefined datatype in c/c++ and class is it's alternative

struct node* XOR (struct node *a, struct node *b)
{
    //this logic is used to fill the nodes with address of a xor linked list
    return  ((int) (a) ^ (int) (b));
}
```

a) nothing wrong. everything is fine
b) type casting at return is missing
c) parameters are wrong
d) total logic is wrong
View Answer

Answer: b
Explanation: It must be typecasted– return (struct node*)((int) (a) (int) (b));

8. Given 10,8,6,7,9
swap the above numbers such that finally you got 6,7,8,9,10
so now reverse 10
9,7,6,8,10
now reverse 9
8,6,7,9,10
7,6,8,9,10
6,7,8,9,10
at this point 6 is ahead so no more reversing can be done so stop.
To implement above algorithm which datastructure is better and why ?
a) linked list. because we can swap elements easily
b) arrays. because we can swap elements easily
c) xor linked list. because there is no overhead of pointers and so memory is saved
d) doubly linked list. because you can traverse back and forth
View Answer

Answer: c
Explanation: XOR linked lists are used to reduce the memory by storing the XOR values of address instead of actual address in pointers.

9. Consider the following pseudocode of insertion in XOR list and write the approximate code snippet of it.

```
void xor-linked-list insert(struct node **head_ref, int value)
{
    node *new_node  = new (struct node);
    new_node->value = value;
    new_node->nodepointerxored = xor (*head_ref, NULL);
    if (*head_pointer == NULL)
    {
        printf("invalid");
    }
    else
    {
        let b,c,d are nodes and a is to be inserted at beginning,
        a address field must contain NULL xor b and b
        address filed must be a xor c.
    }
    *head_pointer = new_node;
}
```

a)

```
node* next = XOR ((*head_ref)->npx,  NULL);
  (*head_ref)->npx = XOR (new_node, next);
```

b)

```
node* next = XOR ((*head_ref)->npx,  NULL);
  (*head_ref) = XOR (new_node, next);
```

c)

```
node* next = XOR ((*head_ref)->npx,  NULL);
```

```
  (*head_ref)->npx->npx = XOR (new_node, next);
```

d)

```
node* next = XOR ((*head_ref),  NULL);
  (*head_ref)->npx = XOR (new_node, next);
```

View Answer
Answer: a
Explanation: They code for the english is

```
node* next = XOR ((*head_ref)->npx,  NULL);
  (*head_ref)->npx = XOR (new_node, next);
```

.

10. In the above question would using arrays and swaping of elements in place of xor linked list would have been more efficient?
a) no not all
b) yes arrays would have been better than xor lists
c) both would be same in efficiency
d) can’t say
View Answer

Answer: b
Explanation: The locality of a normal array is faster in memory and moreover one has to traverse n-nodes to reach the target to reverse in case of xor linked list.
Topic 45. Free List

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Free List”.

1. Free lists are used in
a) static memory allocation
b) dynamic memory allocation
c) contagious allocations
d) are used for speeding up linked list operations
View Answer

Answer: b
Explanation: Their property is meant for dynamic allocations.

2. What are implicit and explicit implementations of freelists?
a) garbage collection and new or malloc operators respectively
b) new or malloc and garbage collection respectively
c) implicit implementation is not favored
d) explicit implementation is not favored
View Answer

Answer: a
Explanation: Gc and new most widely known.

3. What datastructures can be used in implementing a free list?
a) only linked list
b) linked list or sort trees
c) arrays
d) trees
View Answer

Answer: b
Explanation: Sort trees can also be used in impelementing free lists which remaincomplex.

4. What are different ways of implementing free lists and which is simple among them?
a) best fit, first fit, worst fit, simple-first fit
b) best fit, first fit, worst fit, simple-best fit
c) best fit, first fit, worst fit, simple-worst fit
d) best fit simple-best fit
View Answer

Answer: a
Explanation: The simplest form of memory management system can be called as first-fit. a device or system maintains a single list of free memory locations. When request to memory is sent, the list is searched and the first block that is large enough is returned.

5. What is buddy memory management of free lists ?
a) modified version of first fit
b) buddy allocation keeps several free lists, each one holds blocks which are of one particular size
c) modified version of best fit
d) a tree representation of free lists
View Answer

Answer: b
Explanation: When an allocation request is received, the list that holds blocks that are just large enough to satisfy the request are considered, and an open location is returned. If no free blocks that are smaller than two times the size that are requested are available, a larger block is split in two to satisfy the requirements.

6. How does implicit free lists(garbage collection) works in adding memory to free list ?
a) whichever comes last will be added to free list
b) whichever comes first will be added to free list
c) certain blocks cannot be used if there are no pointers to them and hence they can be freed
d) makes a probabilistic guess
View Answer

Answer: c
Explanation: When no pointers pointing a block that means it is useless to be in memory.

7. What are the disadvantages in implementing buddy system algorithm for free lists?
a) internal fragmentation
b) it takes so much space
c) we no more have the hole lists in order of memory address, so it is difficult to detect if 2 holes remain adjacent in memory and shall be merged into one hole
d) both a and c are correct
View Answer

Answer: d
Explanation: Internal fragmentation is an issue to be dealt and it takes so much space.

8. Assume there is a free list which contains nodes and is filled with a value if it is already assigned and the value will be the size of requested block else will be 0.

```
 z = startpoint;
 while ((z < end) &&    \\ didn't reach end
   (*z <= len))          \\ too small to satisfy request
{
   assign this block
 }
```

The above code represents what?
a) code for first fit
b) code for best fit
c) code for worst fit
d) none of the mentioned
View Answer

Answer: a
Explanation: As z is start point and now from beginning we are moving and checking if we reached end and then checking size naively assigning the first block which is bigger than required size hence it is first fit.

9. How are free blocks linked together mostly and in what addressing order?
a) circular linked list and increasing addressing order
b) linked list and decreasing addressing order
c) linked list and in no addressing order
d) none of the mentioned
View Answer

Answer: a
Explanation: A common way is circular linked list and address are arranged in increasing order because merging would be easier which is actually a problem in buddy memory allocation.

10. Accessing free list very frequently for wide range of addresses can lead to
a) paging
b) segmentation fault
c) memory errors
d) cache problems
View Answer

Answer: a
Explanation: Paging in/out of disk will be caused.
Topic 46. Binary Trees using Array

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Binary Trees using Array”.

1. How many children does a binary tree have?
a) 2
b) any number of children
c) 0 or 1 or 2
d) 0 or 1
View Answer

Answer: c
Explanation: Can have atmost 2 nodes.

2. What is/are the disadvantages of implementing tree using normal arrays?
a) difficulty in knowing children nodes of a node
b) difficult in finding the parent of a node
c) have to know the maximum number of nodes possible before creation of trees
d) difficult to implement
View Answer

Answer: c
Explanation: The size of array is fixed in normal arrays. We need to know the number of nodes in the tree before array declaration. It is the main disadvantage of using arrays to represent binary trees.

3. What must be the ideal size of array if the height of tree is ‘l’?
a) 2^l-1
b) l-1
c) l
d) 2l
View Answer

Answer: a
Explanation: Maximum elements in a tree (complete binary tree in worst case) of height ‘L’ is 2^L-1. Hence size of array is taken as 2^L-1.

4. What are the children for node ‘w’ of a complete-binary tree in an array representation?
a) 2w and 2w+1
b) 2+w and 2-w
c) w+1/2 and w/2
d) w-1/2 and w+1/2
View Answer

Answer: a
Explanation: The left child is generally taken as 2*w whereas the right child will be taken as 2*w+1 because root node is present at index 0 in the array and to access every index position in the array.

5. What is the parent for a node ‘w’ of a complete binary tree in an array representation when w is not 0?
a) floor(w-1/2)
b) ceil(w-1/2)
c) w-1/2
d) w/2
View Answer

Answer: a
Explanation: Floor of w-1/2 because we can’t miss a node.

6. If the tree is not a complete binary tree then what changes can be made for easy access of children of a node in the array?
a) every node stores data saying which of its children exist in the array
b) no need of any changes continue with 2w and 2w+1, if node is at i
c) keep a seperate table telling children of a node
d) use another array parallel to the array with tree
View Answer

Answer: a
Explanation: Array cannot represent arbitrary shaped trees. It can only be used in case of complete trees. If every node stores data saying that which of its children exists in the array then elements can be accessed easily.

7. What must be the missing logic in place of missing lines for finding sum of nodes of binary tree in alternate levels?

```
//e.g:-consider -complete binary tree:-height-3, [1,2,3,4,5,6,7]-answer must be 23
n=power(2,height)-1; //assume input is height and a[i] contains tree elements
for(i=1;i<=n;)
{
      //present level is initialized to 1 and sum is initialized to  0
      for(j=1;j<=pow(2,currentlevel-1);j++)
      {
         sum=sum+a[i];
         i=i+1;
      }
   //missing logic
}
```

a)

```
i=i+pow(2,currentlevel);
currentlevel=currentlevel+2;
j=1;
```

b)

```
i=i+pow(2,currentlevel);
currentlevel=currentlevel+2;
j=0;
```

c)

```
i=i-pow(2,currentlevel);
currentlevel=currentlevel+2;
j=1;
```

d)

```
i=i+pow(2,currentlevel);
currentlevel=currentlevel+1;
j=1;
```

View Answer
Answer: a
Explanation: The i value must skip through all nodes in the next level and current level must be one+next level.

8. Consider a situation of writing a binary tree into a file with memory storage efficiency in mind, is array representation of tree is good?
a) yes because we are overcoming the need of pointers and so space efficiency
b) yes because array values are indexable
c) No it is not efficient in case of sparse trees and remaning cases it is fine
d) No linked list representation of tree is only fine
View Answer

Answer: c
Explanation: In case of sparse trees (where one node per level in worst cases), the array size (2^h)-1 where h is height but only h indexes will be filled and (2^h)-1-h nodes will be left unused leading to space wastage.

9. Why is heap implemented using array representations than tree(linked list) representations though both tree representations and heaps have same complexities?

```
for binary heap
-insert: O(log n)
-delete min: O(log n)

for a tree
-insert: O(log n)
-delete: O(log n)
```

Then why go with array representation when both are having same values ?
a) arrays can store trees which are complete and heaps are not complete
b) lists representation takes more memory hence memory efficiency is less and go with arrays and arrays have better caching
c) lists have better caching
d) In lists insertion and deletion is difficult
View Answer

Answer: b

Explanation: In memory the pointer address for next node may not be adjacent or nearer to each other and also array have wonderful caching power from os and manipulating pointers is a overhead. Heap data structure is always a complete binary tree.

10. Can a tree stored in an array using either one of inorder or post order or pre order traversals be again reformed?
a) Yes just traverse through the array and form the tree
b) No we need one more traversal to form a tree
c) No in case of sparse trees
d) Yes by using both inorder and array elements
View Answer

Answer: b
Explanation: We need any two traversals for tree formation but if some additional stuff or techniques are used while storing a tree in an array then one traversal can facilitate like also storing null values of a node in array.
Topic 47. Binary Trees using Linked Lists

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Binary Trees using Linked Lists”.

1. Advantages of linked list representation of binary trees over arrays?
a) dynamic size
b) ease of insertion/deletion
c) ease in randomly accessing a node
d) both dynamic size and ease in insertion/deletion
View Answer

Answer: d
Explanation: It has both dynamic size and ease in insertion and deletion as advantages.

2. Disadvantages of linked list representation of binary trees over arrays?
a) Randomly accessing is not possible
b) Extra memory for a pointer is needed with every element in the list
c) Difficulty in deletion
d) Random access is not possible and extra memory with every element
View Answer

Answer: d
Explanation: Random access is not possible with linked lists.Also it includes extra memory space for each element thereby increasing its space complexity.

3. Which of the following traversing algorithm is not used to traverse in a tree?
a) Post order
b) Pre order
c) Post order
d) Randomized
View Answer

Answer: d
Explanation: Generally, all nodes in a tree are visited by using preorder, inorder and postorder traversing algorithms.

4. Level order traversal of a tree is formed with the help of
a) breadth first search
b) depth first search
c) dijkstra’s algorithm
d) prims algorithm
View Answer

Answer: a
Explanation: Level order is similar to bfs.

5. Identify the reason which doesn’t play a key role to use threaded binary trees?
a) The storage required by stack and queue is more
b) The pointers in most of nodes of a binary tree are NULL
c) It is Difficult to find a successor node
d) They occupy less size
View Answer

Answer: d
Explanation: Threaded binary trees are introduced to make the Inorder traversal faster without using any stack or recursion. Stack and Queue require more space and pointers in the majority of binary trees are null and difficulties are raised while finding successor nodes. Size constraints are not taken on threaded binary trees, but they occupy less space than a

stack/queue.

6. The following lines talks about deleting a node in a binary tree.(the tree property must not be violated after deletion)
i) from root search for the node to be deleted
ii)
iii) delete the node at
what must be statement ii) and fill up statement iii)
a) ii)-find random node,replace with node to be deleted. iii)- delete the node
b) ii)-find node to be deleted. iii)- delete the node at found location
c) ii)-find deepest node,replace with node to be deleted. iii)- delete a node
d) ii)-find deepest node,replace with node to be deleted. iii)- delete the deepest node
View Answer

Answer: d
Explanation: We just replace a to be deleted node with last leaf node of a tree. this must not be done in case of BST or heaps.

7. What may be the psuedo code for finding the size of a tree?
a) find_size(root_node–>left_node) + 1 + find_size(root_node–>right_node)
b) find_size(root_node–>left_node) + find_size(root_node–>right_node)
c) find_size(root_node–>right_node) – 1
d) find_size(root_node–>left_node + 1
View Answer

Answer: a
Explanation: Draw a tree and analyze the expression. we are always taking size of left subtree and right subtree and adding root value(1) to it and finally printing size.

8. What is missing in this logic of finding a path in the tree for a given sum (i.e checking whether there will be a path from roots to leaf nodes with given sum)?

```
checkSum(struct bin-treenode *root , int sum) :
  if(root==null)
    return sum as 0
  else :
     leftover_sum=sum-root_node-->value
     //missing
```

a) code for having recursive calls to either only left tree or right trees or to both subtrees depending on their existence
b) code for having recursive calls to either only left tree or right trees
c) code for having recursive calls to either only left tree
d) code for having recursive calls to either only right trees
View Answer

Answer: a
Explanation: if(left subtree and right subtree) then move to both subtrees
else if only left subtree then move to left subtree carrying leftover_sum parameter
else if only right subtree then move to right subtree carrying leftover_sum parameter.

9. What must be the missing logic below so as to print mirror of a tree as below as an example?

```
if(rootnode):
  mirror(rootnode-->left)
  mirror(rootnode-->right)

  //missing

end
```

a) swapping of left and right nodes is missing
b) swapping of left with root nodes is missing
c) swapping of right with root nodes is missing
d) nothing is missing
View Answer

Answer: a
Explanation: Mirror is another tree with left and right children of nodes are interchanged as shown in the figure.

10. What is the code below trying to print?

```
void print(tree *root,tree *node)
{
  if(root ==null) return 0
  if(root-->left==node || root-->right==node) || print(root->left,node)
  ||printf(root->right,node)
  {
    print(root->data)
  }
}
```

a) just printing all nodes
b) not a valid logic to do any task
c) printing ancestors of a node passed as argument
d) printing nodes from leaf node to a node passed as argument
View Answer

Answer: c
Explanation: We are checking if left or right node is what the argument sent or else if not the case then move to left node or right node and print all nodes while searching for the argument node.
Topic 48. Binary Tree Operations

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Binary Tree Operations”.

1. What is the maximum number of children that a binary tree node can have?
a) 0
b) 1
c) 2
d) 3
View Answer

Answer: c
Explanation: In a binary tree, a node can have atmost 2 nodes (i.e.) 0,1 or 2 left and right child.

2. The following given tree is an example for?

a) Binary tree
b) Binary search tree
c) Fibonacci tree
d) AVL tree
View Answer

Answer: a
Explanation: The given tree is an example for binary tree since has got two children and the left and right children do not satisfy binary search tree’s property, Fibonacci and AVL tree.

3. A binary tree is a rooted tree but not an ordered tree.
a) true
b) false
View Answer

Answer: b
Explanation: A binary tree is a rooted tree and also an ordered tree (i.e) every node in a binary tree has at most two children.

4. How many common operations are performed in a binary tree?
a) 1
b) 2
c) 3
d) 4
View Answer

Answer: c
Explanation: Three common operations are performed in a binary tree- they are insertion, deletion and traversal.

5. What is the traversal strategy used in the binary tree?
a) depth-first traversal
b) breadth-first traversal
c) random traversal
d) Priority traversal
View Answer

Answer: b
Explanation: Breadth first traversal, also known as level order traversal is the traversal strategy used in a binary tree. It involves visiting all the nodes at a given level.

6. How many types of insertion are performed in a binary tree?

a) 1
b) 2
c) 3
d) 4
View Answer

Answer: b
Explanation: Two kinds of insertion operation is performed in a binary tree- inserting a leaf node and inserting an internal node.

7. What operation does the following diagram depict?

a) inserting a leaf node
b) inserting an internal node
c) deleting a node with 0 or 1 child
d) deleting a node with 2 children
View Answer

Answer: c
Explanation: The above diagram is a depiction of deleting a node with 0 or 1 child since the node D which has 1 child is deleted.

8. General ordered tree can be encoded into binary trees.
a) true
b) false
View Answer

Answer: a
Explanation: General ordered tree can be mapped into binary tree by representing them in a left-child-right-sibling way.

9. How many bits would a succinct binary tree occupy?
a) n+O(n)
b) 2n+O(n)
c) n/2
d) n
View Answer

Answer: b
Explanation: A succinct binary tree occupies close to minimum possible space established by lower bounds. A succinct binary tree would occupy 2n+O(n) bits.

10. The average depth of a binary tree is given as?
a) O(N)
b) O(√N)
c) O(N^2)
d) O(log N)
View Answer

Answer: b
Explanation: The average depth of a binary tree is given as O(√N). In case of a binary search tree, it is O(log N).

11. How many orders of traversal are applicable to a binary tree (In General)?
a) 1
b) 4
c) 2
d) 3
View Answer

Answer: d
Explanation: The three orders of traversal that can be applied to a binary tree are in-order, pre-order and post order traversal.

12. If binary trees are represented in arrays, what formula can be used to locate a left child, if the node has an index i?
a) 2i+1
b) 2i+2
c) 2i
d) 4i
View Answer

Answer: a

Explanation: If binary trees are represented in arrays, left children are located at indices 2i+1 and right children at 2i+2.

13. Using what formula can a parent node be located in an array?
a) (i+1)/2
b) (i-1)/2
c) i/2
d) 2i/2
View Answer

Answer: b
Explanation: If a binary tree is represented in an array, parent nodes are found at indices (i-1)/2.

14. Which of the following properties are obeyed by all three tree – traversals?
a) Left subtrees are visited before right subtrees
b) Right subtrees are visited before left subtrees
c) Root node is visited before left subtree
d) Root node is visited before right subtree
View Answer

Answer: a
Explanation: In preorder, inorder and postorder traversal the left subtrees are visited before the right subtrees. In Inorder traversal, the Left subtree is visited first then the Root node then the Right subtree. In postorder traversal, the Left subtree is visited first, then Right subtree and then the Root node is visited.

15. Construct a binary tree using the following data.
The preorder traversal of a binary tree is 1, 2, 5, 3, 4. The inorder traversal of the same binary tree is 2, 5, 1, 4, 3.
a)
b)
c)
d)
View Answer

Answer: d
Explanation: Here,
Preorder Traversal is 1, 2, 5, 3, 4
Inorder Traversal is 2, 5, 1, 4, 3
Root node of binary tree is the first node in Preorder traversal.
The rough sketch of tree is:

Second node in preorder traversal is 2. This makes 5 as right child to node 2. The fourth node in preorder traversal is 3. This makes 4 as right child to node 3. Thus the final tree is:
Topic 49. Preorder Traversal

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Preorder Traversal”.

1. For the tree below, write the pre-order traversal.

a) 2, 7, 2, 6, 5, 11, 5, 9, 4
b) 2, 7, 5, 2, 6, 9, 5, 11, 4
c) 2, 5, 11, 6, 7, 4, 9, 5, 2
d) 2, 7, 5, 6, 11, 2, 5, 4, 9
View Answer

Answer: a
Explanation: Pre order traversal follows NLR(Node-Left-Right).

2. For the tree below, write the post-order traversal.

a) 6, 2, 7, 2, 5, 11, 9, 5, 4
b) 6, 5, 11, 2, 7, 5, 9, 4, 2
c) 6, 5, 2, 11, 7, 4, 9, 5, 2
d) 6, 2, 7, 2, 11, 5, 5, 9, 4
View Answer

Answer: c
Explanation: Post order traversal follows LRN(Left-Right-Node).

3. Select the code snippet which performs pre-order traversal.
a)

```
public void preorder(Tree root)
```

```
{
 System.out.println(root.data);
 preorder(root.left);
 preorder(root.right);
}
```

b)

```
public void preorder(Tree root)
{
 preorder(root.left);
 System.out.println(root.data);
 preorder(root.right);
}
```

c)

```
public void preorder(Tree root)
{
 System.out.println(root.data);
 preorder(root.right);
 preorder(root.left);
}
```

d)

```
public void preorder(Tree root)
{
 preorder(root.right);
 preorder(root.left);
        System.out.println(root.data);
}
```

View Answer
Answer: a
Explanation: Pre-order traversal follows NLR(Node-Left-Right).

4. Select the code snippet which performs post-order traversal.
a)

```
public void postorder(Tree root)
{
 System.out.println(root.data);
 postorder(root.left);
 postorder(root.right);
}
```

b)

```
public void postorder(Tree root)
{
 postorder(root.left);
 postorder(root.right);
 System.out.println(root.data);
}
```

c)

```
public void postorder(Tree root)
{
 System.out.println(root.data);
 postorder(root.right);
 postorder(root.left);
}
```

d)

```
public void postorder(Tree root)
{
 postorder(root.right);
        System.out.println(root.data);
 postorder(root.left);
}
```

View Answer
Answer: b
Explanation: Post order traversal follows NLR(Left-Right-Node).

5. Select the code snippet that performs pre-order traversal iteratively.

a)

```
public void preOrder(Tree root)
{
        if (root == null) return;
 Stack<Tree> stk = new Stack<Tree>();
        st.add(root);
 while (!stk.empty())
        {
            Tree node = stk.pop();
            System.out.print(node.data + " ");
   if (node.left != null) stk.push(node.left);
            if (node.right != null) stk.push(node.right);
        }
}
```

b)

```
public void preOrder(Tree root)
{
        if (root == null) return;
  Stack<Tree> stk = new Stack<Tree>();
 while (!stk.empty())
        {
            Tree node = stk.pop();
            System.out.print(node.data + " ");
            if (node.right != null) stk.push(node.right);
            if (node.left != null) stk.push(node.left);
        }
}
```

c)

```
public void preOrder(Tree root)
{
        if (root == null) return;
 Stack<Tree> stk = new Stack<Tree>();
        st.add(root);
 while (!stk.empty())
        {
            Tree node = stk.pop();
            System.out.print(node.data + " ");
            if (node.right != null) stk.push(node.right);
            if (node.left != null) stk.push(node.left);
        }
}
```

d)

```
public void preOrder(Tree root)
{
        if (root == null) return;
 Stack<Tree> stk = new Stack<Tree>();
        st.add(root);
 while (!stk.empty())
        {
            Tree node = stk.pop();
            System.out.print(node.data + " ");
            if (node.right != null) stk.push(node.left);
            if (node.left != null) stk.push(node.right);
        }
}
```

View Answer
Answer: c
Explanation: Add the root to the stack first, then continously check for the right and left children of the top-of-the-stack.

6. Select the code snippet that performs post-order traversal iteratively.
a)

```
public void postorder(Tree root)
{
        if (root == null)
        return;
 Stack<Tree> stk = new Stack<Tree>();
        Tree node = root;
 while (!stk.isEmpty() || node != null)
        {
            while (node != null)
            {
                if (node.right != null)
                    stk.add(node.left);
                stk.add(node);
                node = node.right;
            }
     node = stk.pop();
     if (node.right != null && !stk.isEmpty() && node.right == stk.peek())
            {
                Trees nodeRight = stk.pop();
                stk.push(node);
                node = nodeRight;
            } else
            {
                System.out.print(node.data + " ");
                node = null;
            }
        }
}
```

b)

```
public void postorder(Tree root)
{
        if (root == null)
        return;
 Stack<Tree> stk = new Stack<Tree>();
        Tree node = root;
 while (!stk.isEmpty() || node != null)
        {
            while (node != null)
            {
                if (node.right != null)
                    stk.add(node.right);
                stk.add(node);
                node = node.left;
            }
            node = stk.pop();
     if (node.right != null && !stk.isEmpty() && node.right == stk.peek())
            {
                Trees nodeRight = stk.pop();
                stk.push(node);
                node = nodeRight;
            } else
            {
                System.out.print(node.data + " ");
                node = null;
            }
        }
}
```

c)

```
public void postorder(Tree root)
{
        if (root == null)
            return;
 Stack<Tree> stk = new Stack<Tree>();
        Tree node = root;
 while (!stk.isEmpty() || node != null)
        {
            while (node != null)
            {
                if (node.right != null)
                    stk.add(node.right);
                stk.add(node);
                node = node.left;
            }
     node = stk.pop();
            if (node.right != null)
            {
                Trees nodeRight = stk.pop();
                stk.push(node);
                node = nodeRight;
            } else
            {
                System.out.print(node.data + " ");
                node = null;
            }
        }
}
```

d)

```
public void postorder(Tree root)
{
        if (root == null)
            return;
 Stack<Tree> stk = new Stack<Tree>();
        Tree node = root;
 while (!stk.isEmpty() || node != null)
       {
           while (node != null)
           {
               if (node.right != null)
                   stk.add(node.left);
               stk.add(node);
               node = node.left;
           }
    node = stk.pop();
           if (node.right != null)
           {
               Trees nodeRight = stk.pop();
               stk.push(node);
               node = nodeLeft;
           } else
           {
               System.out.print(node.data + " ");
               node = null;
           }
       }
}
```

View Answer
Answer: b
Explanation: The left and right children are added first to the stack, followed by the node, which is then popped. Post-order follows LRN policy.

7. What is the time complexity of pre-order traversal in the iterative fashion?
a) O(1)
b) O(n)
c) O(logn)
d) O(nlogn)
View Answer

Answer: b
Explanation: Since you have to go through all the nodes, the complexity becomes O(n).

8. What is the space complexity of the post-order traversal in the recursive fashion? (d is the tree depth and n is the number of nodes)
a) O(1)
b) O(nlogd)
c) O(logd)
d) O(d)
View Answer

Answer: d
Explanation: In the worst case we have d stack frames in the recursive call, hence the complexity is O(d).

9. To obtain a prefix expression, which of the tree traversals is used?
a) Level-order traversal
b) Pre-order traversal
c) Post-order traversal
d) In-order traversal
View Answer

Answer: b
Explanation: As the name itself suggests, pre-order traversal can be used.

10. Consider the following data. The pre order traversal of a binary tree is A, B, E, C, D. The in order traversal of the same binary tree is B, E, A, D, C. The level order sequence for the binary tree is _________
a) A, C, D, B, E
b) A, B, C, D, E
c) A, B, C, E, D
d) D, B, E, A, C
View Answer

Answer: c
Explanation: The inorder sequence is B, E, A, D, C and Preorder sequence is A, B, E, C, D. The tree constructed with the inorder and preorder sequence is

The levelorder traversal (BFS traversal) is A, B, C, E, D.

11. Consider the following data and specify which one is Preorder Traversal Sequence, Inorder and Postorder sequences.
S1: N, M, P, O, Q
S2: N, P, Q, O, M
S3: M, N, O, P, Q
a) S1 is preorder, S2 is inorder and S3 is postorder
b) S1 is inorder, S2 is preorder and S3 is postorder
c) S1 is inorder, S2 is postorder and S3 is preorder
d) S1 is postorder, S2 is inorder and S3 is preorder
View Answer

Answer: c
Explanation: Preorder traversal starts from the root node and postorder and inorder starts from the left child node of the left subtree. The first node of S3 is different and for S1 and S2 it's the same. Thus, S3 is preorder traversal and the root node is M. Postorder traversal visits the root node at last. S2 has the root node(M) at last that implies S2 is postorder traversal. S1 is inorder traversal as S2 is postorder traversal and S3 is preorder traversal. Therefore, S1 is inorder traversal, S2 is postorder traversal and S3 is preorder traversal.
Topic 50. Postorder Traversal

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on "Postorder Traversal".

1. In postorder traversal of binary tree right subtree is traversed before visiting root.
a) True
b) False
View Answer

Answer: a
Explanation: Post-order method of traversing involves – i) Traverse left subtree in post-order, ii) Traverse right subtree in post-order, iii) visit the root.

2. What is the possible number of binary trees that can be created with 3 nodes, giving the sequence N, M, L when traversed in post-order.
a) 15
b) 3
c) 5
d) 8
View Answer

Answer: c
Explanation: 5 binary trees are possible and they are,

3. The post-order traversal of a binary tree is O P Q R S T. Then possible pre-order traversal will be ________
a) T Q R S O P
b) T O Q R P S
c) T Q O P S R
d) T Q O S P R
View Answer

Answer: c
Explanation: The last, second last nodes visited in post-order traversal are root and it's right child respectively. Option T Q R S O P can't be a pre-order traversal, because nodes O, P are visited after the nodes Q, R, S. Option T O Q R P S, can't be valid, because the pre-order sequence given in option T O Q R P S and given post-order traversal creates a tree with node T as root and node O as left subtree. Option T Q O P S R is valid. Option T Q O S P R is not valid as node P is visited after visiting node S.

4. A binary search tree contains values 7, 8, 13, 26, 35, 40, 70, 75. Which one of the following is a valid post-order sequence of the tree provided the pre-order sequence as 35, 13, 7, 8, 26, 70, 40 and 75?
a) 7, 8, 26, 13, 75, 40, 70, 35
b) 26, 13, 7, 8, 70, 75, 40, 35
c) 7, 8, 13, 26, 35, 40, 70, 75
d) 8, 7, 26, 13, 40, 75, 70, 35
View Answer

Answer: d
Explanation: The binary tree contains values 7, 8, 13, 26, 35, 40, 70, 75. The given pre-order sequence is 35, 13, 7, 8, 26, 70, 40 and 75. So, the binary search tree formed is

Thus post-order sequence for the tree is 8, 7, 26, 13, 40, 75, 70 and 35.

5. Which of the following pair's traversals on a binary tree can build the tree uniquely?

a) post-order and pre-order
b) post-order and in-order
c) post-order and level order
d) level order and preorder
View Answer

Answer: b
Explanation: A binary tree can uniquely be created by post-order and in-order traversals.

6. A full binary tree can be generated using ______
a) post-order and pre-order traversal
b) pre-order traversal
c) post-order traversal
d) in-order traversal
View Answer

Answer: a
Explanation: Every node in a full binary tree has either 0 or 2 children. A binary tree can be generated by two traversals if one of them is in-order. But, we can generate a full binary tree using post-order and pre-order traversals.

7. The maximum number of nodes in a tree for which post-order and pre-order traversals may be equal is ______
a) 3
b) 1
c) 2
d) any number
View Answer

Answer: b
Explanation: The tree with only one node has post-order and pre-order traversals equal.

8. The steps for finding post-order traversal are traverse the right subtree, traverse the left subtree or visit the current node.
a) True
b) False
View Answer

Answer: b
Explanation: Left subtree is traversed first in post-order traversal, then the right subtree is traversed and then the output current node.

9. The pre-order and in-order are traversals of a binary tree are T M L N P O Q and L M N T O P Q. Which of following is post-order traversal of the tree?
a) L N M O Q P T
b) N M O P O L T
c) L M N O P Q T
d) O P L M N Q T
View Answer

Answer: a
Explanation: The tree generated by using given pre-order and in-order traversal is

Thus, L N M O Q P T will be the post-order traversal.

10. For a binary tree the first node visited in in-order and post-order traversal is same.
a) True
b) False
View Answer

Answer: b
Explanation: Consider a binary tree,

Its in-order traversal – 13 14 16 19
Its post-order traversal- 14 13 19 16. Here the first node visited is not same.

11. Find the postorder traversal of the binary tree shown below.

a) P Q R S T U V W X
b) W R S Q P V T U X
c) S W T Q X U V R P
d) S T W U X V Q R P
View Answer

Answer: c
Explanation: In postorder traversal the left subtree is traversed first and then the right subtree and then the current node. So, the posturer traversal of the tree is, S W T Q X U V R P.
Topic 51. Inorder Traversal

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Inorder Traversal”.

1. For the tree below, write the in-order traversal.

a) 6, 2, 5, 7, 11, 2, 5, 9, 4
b) 6, 5, 2, 11, 7, 4, 9, 5, 2
c) 2, 7, 2, 6, 5, 11, 5, 9, 4
d) 2, 7, 6, 5, 11, 2, 9, 5, 4
View Answer

Answer: a
Explanation: In-order traversal follows LNR(Left-Node-Right).

2. For the tree below, write the level-order traversal.

a) 2, 7, 2, 6, 5, 11, 5, 9, 4
b) 2, 7, 5, 2, 11, 9, 6, 5, 4
c) 2, 5, 11, 6, 7, 4, 9, 5, 2
d) 2, 7, 5, 6, 11, 2, 5, 4, 9
View Answer

Answer: b
Explanation: Level order traversal follows a breadth first search approach.

3. Select the code snippet which performs in-order traversal.
a)

```
public void inorder(Tree root)
{
 System.out.println(root.data);
 inorder(root.left);
 inorder(root.right);
}
```

b)

```
public void inorder(Tree root)
{
 inorder(root.left);
 System.out.println(root.data);
 inorder(root.right);
}
```

c)

```
public void inorder(Tree root)
{
 System.out.println(root.data);
 inorder(root.right);
 inorder(root.left);
}
```

d)

```
public void inorder(Tree root)
{
 inorder(root.right);
 inorder(root.left);
 System.out.println(root.data);
}
```

View Answer
Answer: b
Explanation: In-order traversal follows LNR(Left-Node-Right).

4. Select the code snippet which performs level-order traversal.
a)

```
public static void levelOrder(Tree root)
{
    Queue<Node> queue=new LinkedList<Node>();
    queue.add(root);
    while(!queue.isEmpty())
    {
        Node tempNode=queue.poll();
        System.out.println("%d ",tempNode.data);
        if(tempNode.left!=null)
        queue.add(tempNode.left);
        if(tempNode.right!=null)
        queue.add(tempNode.right);
    }
}
```

b)

```
public static void levelOrder(Tree root)
{
    Queue<Node> queue=new LinkedList<Node>();
    queue.add(root);
    while(!queue.isEmpty())
    {
        Node tempNode=queue.poll();
        System.out.println("%d ",tempNode.data);
        if(tempNode.left!=null)
        queue.add(tempNode.right);
        if(tempNode.right!=null)
        queue.add(tempNode.left);
    }
}
```

c)

```
public static void levelOrder(Tree root)
{
    Queue<Node> queue=new LinkedList<Node>();
    queue.add(root);
    while(!queue.isEmpty())
    {
        Node tempNode=queue.poll();
        System.out.println("%d ",tempNode.data);
        if(tempNode.right!=null)
        queue.add(tempNode.left);
        if(tempNode.left!=null)
        queue.add(tempNode.right);
    }
}
```

d)

```
public static void levelOrder(Tree root)
{
    Queue<Node> queue=new LinkedList<Node>();
    queue.add(root);
    while(!queue.isEmpty())
    {
        Node tempNode=queue.poll();
        System.out.println("%d ",tempNode.data);
        if(tempNode.right!=null)
        queue.add(tempNode.left.left);
        if(tempNode.left!=null)
        queue.add(tempNode.right.right);
    }
}
```

View Answer
Answer: a
Explanation: Firstly add the root node to the queue. Then for all the remaining nodes, pop the front end of the queue and print it, add the left and right children of the popped node to the queue.

5. What is the space complexity of the in-order traversal in the recursive fashion? (d is the tree depth and n is the number of nodes)
a) O(1)
b) O(nlogd)
c) O(logd)
d) O(d)
View Answer

Answer: d
Explanation: In the worst case we have d stack frames in the recursive call, hence the complexity is O(d).

6. What is the time complexity of level order traversal?
a) O(1)
b) O(n)
c) O(logn)
d) O(nlogn)
View Answer

Answer: b
Explanation: Since you have to go through all the nodes, the complexity becomes O(n).

7. Which of the following graph traversals closely imitates level order traversal of a binary tree?
a) Depth First Search
b) Breadth First Search
c) Depth & Breadth First Search
d) Binary Search
View Answer

Answer: b
Explanation: Both level order tree traversal and breadth first graph traversal follow the principle that visit your neighbors first and then move on to further nodes.

8. In a binary search tree, which of the following traversals would print the numbers in the ascending order?
a) Level-order traversal
b) Pre-order traversal
c) Post-order traversal
d) In-order traversal
View Answer

Answer: d
Explanation: In a binary search tree, a node's left child is always lesser than the node and right child is greater than the node, hence an in-order traversal would print them in a non decreasing fashion.
Topic 52. Binary Tree Properties

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on "Binary Tree Properties".

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

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
View Answer

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
View Answer

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

View Answer

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)
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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 $2^{λ – 1}$ 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))
View Answer

Answer: d
Explanation: In a binary tree, there are atmost 2k nodes in level k and 2^{k-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)
View Answer

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)
View Answer

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)
View Answer

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:

Topic 53. Binary Search Tree

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on "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
View Answer

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.

2. How to search for a key in a binary search tree?
a)

```
public Tree search(Tree root, int key)
{
 if( root == null || root.key == key )
        {
  return root;
 }
 if( root.key < key )
        {
  return search(root.right,key);
 }
 else
 return search(root.left,key);
}
```

b)

```
public Tree search(Tree root, int key)
{
 if( root == null || root.key == key )
        {
  return root;
 }
 if( root.key < key )
        {
  return search(root.left,key);
 }
 else
 return search(root.right,key);
}
```

c)

```
public Tree search(Tree root, int key)
{
 if( root == null)
        {
  return root;
 }
 if( root.key < key )
        {
  return search(root.right,key);
 }
 else
  return search(root.left,key);
}
```

d)

```
public Tree search(Tree root, int key)
{
 if( root == null)
        {
  return root;
 }
 if( root.key < key )
        {
  return search(root.right.right,key);
 }
 else
  return search(root.left.left,key);
}
```

View Answer
Answer: a
Explanation: As we know that the left child is lesser than the parent, if the root’s key is greater than the given key, we look only into the left sub-tree, similarly for right sub-tree.

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
View Answer

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.

4. What does the following piece of code do?

```
public void func(Tree root)
{
 func(root.left());
 func(root.right());
 System.out.println(root.data());
}
```

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

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

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
View Answer

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

6. How will you find the minimum element in a binary search tree?
a)

```
public void min(Tree root)
{
 while(root.left() != null)
 {
  root = root.left();
 }
 System.out.println(root.data());
}
```

b)

```
public void min(Tree root)
{
 while(root != null)
 {
  root = root.left();
 }
 System.out.println(root.data());
}
```

c)

```
public void min(Tree root)
{
 while(root.right() != null)
 {
  root = root.right();
 }
 System.out.println(root.data());
}
```

d)

```
public void min(Tree root)
{
 while(root != null)
 {
  root = root.right();
 }
 System.out.println(root.data());
}
```

View Answer

Answer: a
Explanation: Since all the elements lesser than a given node will be towards the left, iterating to the leftmost leaf of the root will give the minimum element in a binary search tree.

7. How will you find the maximum element in a binary search tree?
a)

```
public void max(Tree root)
{
 while(root.left() != null)
 {
  root = root.left();
 }
 System.out.println(root.data());
}
```

b)

```
public void max(Tree root)
{
 while(root != null)
 {
  root = root.left();
 }
 System.out.println(root.data());
}
```

c)

```
public void max(Tree root)
{
 while(root.right() != null)
 {
  root = root.right();
 }
 System.out.println(root.data());
}
```

d)

```
public void max(Tree root)
{
 while(root != null)
 {
  root = root.right();
 }
 System.out.println(root.data());
}
```

View Answer
Answer: c
Explanation: Since all the elements greater than a given node are towards the right, iterating through the tree to the rightmost leaf of the root will give the maximum element in a binary search tree.

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)
View Answer

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.

9. Given that 2 elements are present in the tree, write a function to find the LCA(Least Common Ancestor) of the 2 elements.
a)

```
public void lca(Tree root,int n1, int n2)
{
 while (root != NULL)
       {
            if (root.data() > n1 && root.data() > n2)
            root = root.right();
            else if (root.data() < n1 && root.data() < n2)
            root = root.left();
     else break;
        }
        System.out.println(root.data());
}
```

b)

```
public void lca(Tree root,int n1, int n2)
{
    while (root != NULL)
    {
        if (root.data() > n1 && root.data() < n2)
        root = root.left();
        else if (root.data() < n1 && root.data() > n2)
        root = root.right();
 else break;
    }
    System.out.println(root.data());
}
```

c)

```
public void lca(Tree root,int n1, int n2)
{
    while (root != NULL)
    {
        if (root.data() > n1 && root.data() > n2)
        root = root.left();
        else if (root.data() < n1 && root.data() < n2)
        root = root.right();
 else break;
    }
    System.out.println(root.data());
}
```

d)

```
public void lca(Tree root,int n1, int n2)
{
    while (root != NULL)
    {
        if (root.data() > n1 && root.data() > n2)
        root = root.left.left();
        else if (root.data() < n1 && root.data() < n2)
        root = root.right.right();
 else break;
    }
    System.out.println(root.data());
}
```

View Answer
Answer: c
Explanation: The property of a binary search tree is that the lesser elements are to the left and greater elements are to the right, we use this property here and iterate through the tree such that we reach a point where the 2 elements are on 2 different sides of the node, this becomes the least common ancestor of the 2 given elements.

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
View Answer

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.

11. Construct a binary search tree with the below information.
The preorder traversal of a binary search tree 10, 4, 3, 5, 11, 12.
a)
b)
c)
d)
View Answer

Answer: c
Explanation: Preorder Traversal is 10, 4, 3, 5, 11, 12. Inorder Traversal of Binary search tree is equal to ascending order of the nodes of the Tree. Inorder Traversal is 3, 4, 5, 10, 11, 12. The tree constructed using Preorder and Inorder traversal is

Topic 54. Balanced Binary Tree

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on "Balanced Binary Tree".

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

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
View Answer

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 balance factor of the node.

3. Figure below is a balanced binary tree. If a node inserted as child of the node R, how many nodes will become unbalanced?

a) 2
b) 1
c) 3
d) 0
View Answer

Answer: b
Explanation: Only the node P will become unbalanced, with balance factor +2.

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
View Answer

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
View Answer

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

6. Which of following figures is a balanced binary tree?
a)
b)
c)
d)
View Answer

Answer: b

Explanation: In Some tree diagrams, the root of tree has balance factor +2, so the tree is not balanced. If every node in the tree is balanced, then it’s a balanced tree.

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)
View Answer

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
View Answer

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

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)
View Answer

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.

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
View Answer

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).

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

Answer: a
Explanation: AVL tree is more balanced than a Red-black tree because AVL tree has less height than Red-black tree given that both trees have the same number of elements.

12. The figure shown below is a balanced binary tree. If node P is deleted, which of the following nodes will get unbalanced?

a) U
b) M
c) H
d) A
View Answer

Answer: a
Explanation: Node U will get unbalanced if node P is deleted, because it’s balance factor will become -2.
Topic 55. Self Balancing Binary Search Tree

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Self Balancing Binary Search Tree”.

1. Which of the following is not the self balancing binary search tree?
a) AVL Tree
b) 2-3-4 Tree
c) Red – Black Tree
d) Splay Tree
View Answer

Answer: b
Explanation: 2-3-4 Tree is balanced search trees. But it is not a binary tree. So, it is not a self balancing binary tree. AVL tree, Red-Black Tree and Splay tree are self balancing binary search tree.

2. The binary tree sort implemented using a self – balancing binary search tree takes ______ time is worst case.
a) O(n log n)
b) O(n)
c) O(n^2)
d) O(log n)
View Answer

Answer: a
Explanation: The worst case running time of the binary tree sort is O(n^2). But, the worst case running time can be improved to the O(n log n) using a self – balancing binary search tree.

3. An AVL tree is a self – balancing binary search tree, in which the heights of the two child sub trees of any node differ by _________
a) At least one
b) At most one
c) Two
d) At most two
View Answer

Answer: b
Explanation: In an AVL tree, the difference between heights of the two child sub trees of any node is at most one. If the height differs by more than one, AVL tree performs rotations to balance the tree.

4. Associative arrays can be implemented using __________
a) B-tree
b) A doubly linked list
c) A single linked list
d) A self balancing binary search tree
View Answer

Answer: d
Explanation: Associative arrays can be implemented using a self balancing binary search tree as the worst-case time performance of self – balancing binary search trees is O(log n).

5. Self – balancing binary search trees have a much better average-case time complexity than hash tables.
a) True
b) False
View Answer

Answer: b
Explanation: For lookup, insertion and deletion hash table take O(1) time in average-case while self – balancing binary search trees takes O(log n). Therefore, hash tables perform better in average-case.

6. Which of the following is a self – balancing binary search tree?
a) 2-3 tree
b) Threaded binary tree
c) AA tree
d) Treap
View Answer

Answer: c
Explanation: An AA tree, which is a variation of red-black tree, is a self – balancing binary search tree. 2-3 is B-tree of order 3 and Treat is a randomized binary search tree. A threaded binary tree is not a balanced tree.

7. A self – balancing binary search tree can be used to implement ________
a) Priority queue
b) Hash table
c) Heap sort
d) Priority queue and Heap sort

View Answer

Answer: a
Explanation: Self-balancing binary search trees can be used to construct and maintain ordered lists, to achieve the optimal worst case performance. So, self – balancing binary search tree can be used to implement a priority queue, which is ordered list.

8. In which of the following self – balancing binary search tree the recently accessed element can be accessed quickly?
a) AVL tree
b) AA tree
c) Splay tree
d) Red – Black tree
View Answer

Answer: c
Explanation: In a Splay tree, the recently accessed element can be accessed quickly. In Splay tree, the frequently accessed nodes are moved towards the root so they are quick to access again.

9. The minimum height of self balancing binary search tree with n nodes is _____
a) $\log_2(n)$
b) n
c) 2n + 1
d) 2n – 1
View Answer

Answer: a
Explanation: Self – balancing binary trees adjust the height by performing transformations on the tree at key insertion times, in order to keep the height proportional to $\log_2(n)$.

10. Binary tree sort implemented using a self balancing binary search tree takes O(n log n) time in the worst case but still it is slower than merge sort.
a) True
b) False
View Answer

Answer: a
Explanation: The worst case performance of binary tree sort is O(n log n) when it is implemented using a self balancing binary search tree. Self balancing binary search trees perform transformations to balance the tree, which caused balancing overhead. Due to this overhead, binary tree sort is slower than merger sort.
Topic 56. Randomized Binary Search Tree

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Randomized Binary Search Tree”.

1. Which of the following is not a random tree?
a) Treap
b) Random Binary Tree
c) Uniform Spanning Tree
d) AVL Tree
View Answer

Answer: d
Explanation: Treap, also known as random binary search tree, Random binary tree and Uniform spanning tree are all random tree. Random tree is a tree formed by a random process of addition and deletion of nodes. AVL tree is a self – balanced binary search tree.

2. Which process forms the randomized binary search tree?
a) Stochastic Process
b) Branching Process
c) Diffusion Process
d) Aggregation Process
View Answer

Answer: a
Explanation: The randomized binary search tree is formed by the stochastic process. The stochastic process or also called random process is a mathematical tool or object including random variables.

3. How many randomized binary search trees can be formed by the numbers (1, 3, 2)?
a) 2
b) 3

c) 6
d) 5
View Answer

Answer: d
Explanation: As there are 3 numbers (1, 3, 2) so total of 6 combinations can be formed using three numbers but Since (2, 1, 3) and (2, 3, 1) are same so in total there are 5 randomized binary search tree that can be formed.

4. What is the expected depth of a node in a randomized binary search tree?
a) log n
b) n!
c) n^2
d) 2 log n + O(1)
View Answer

Answer: d
Explanation: The expected value of depth of a node that is for a node a, the expected value of length of path from root to node a is found to be at most 2 log n + O(1).

5. What is the longest length path for a node x in random binary search tree for the insertion process?
a) log x
b) x^2
c) x!
d) 4.311 log x
View Answer

Answer: d
Explanation: Although it is difficult to find the length of the longest path in randomized binary search tree, but it has been found that the longest length is around 4.311 log x.

6. What is the range of β in finding the length of the longest path in a randomized binary search tree?
a) (-1, 0)
b) (1, 0)
c) (0, 5)
d) (0, 1)
View Answer

Answer: d
Explanation: The longest path in a randomized binary search tree, but it has been found that the longest length is around 4.311 log x for node x. This is also equal to 1/β log x where β lies in the range (0, 1).

7. What is the expected number of leaves in a randomized binary search tree?
a) n + 1
b) (n + 1)/3
c) (n + 1)/2
d) n + 3
View Answer

Answer: b
Explanation: In a random mathematical model, the expected value of number of leaves in a randomized binary search tree is found to be exactly (n + 1)/3 using probability.

8. Is Treap a randomized tree.
a) True
b) False
View Answer

Answer: a
Explanation: Treap is a type of data structure which is a combination of binary tree and heap. It is an example of a randomized binary search tree. It stores value in pairs.

9. What is the probability of selecting a tree uniformly at random?
a) Equal to Catalan Number
b) Less Than Catalan Number
c) Greater than Catalan Number
d) Reciprocal of Catalan Number
View Answer

Answer: d
Explanation: Catalan number is a sequence of natural number that is used in counting problem. Hence it is found that the

selecting off a tree uniformly at random is reciprocal of Catalan number.

10. Is mathematical randomized tree can be generated using beta distribution.
a) True
b) False
View Answer

Answer: a
Explanation: Beta distribution can be used using a different shape to generate a randomized binary search tree to create a special type of tree known as a botanical tree.
Topic 57. AA Tree

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “AA Tree”.

1. AA Trees are implemented using?
a) Colors
b) Levels
c) Node size
d) Heaps
View Answer

Answer: b
Explanation: AA Trees are implemented using levels instead of colors to overcome the disadvantages of Red-Black trees.

2. Which of the following is the correct definition for a horizontal link?
a) connection between node and a child of equal levels
b) connection between two nodes
c) connection between two child nodes
d) connection between root node and leaf node
View Answer

Answer: a
Explanation: A horizontal link is a connection between a node and a child of equal levels.

3. How will you remove a left horizontal link in an AA-tree?
a) by performing right rotation
b) by performing left rotation
c) by deleting both the elements
d) by inserting a new element
View Answer

Answer: a
Explanation: A left horizontal link is removed by right rotation. A right horizontal link is removed by left rotation.

4. What are the two different operations done in an AA-Tree?
a) shift and color
b) skew and split
c) zig and zag
d) enqueue and dequeue
View Answer

Answer: b
Explanation: A skew removes a left horizontal link by right rotation and a split removes a right horizontal link by left rotation.

5. In an AA-tree, we process split first, followed by a skew.
a) True
b) False
View Answer

Answer: b
Explanation: In an AA-tree, skew is processed first followed by a split.

6. How many different shapes does maintenance of AA-Tree need to consider?
a) 7
b) 5
c) 2
d) 3
View Answer

Answer: c
Explanation: An AA-Tree needs to consider only two shapes unlike a red-black tree which needs to consider seven shapes

of transformation.

7. What is the prime condition of AA-tree which makes it simpler than a red-black tree?
a) Only right children can be red
b) Only left children can be red
c) Right children should strictly be black
d) There should be no left children
View Answer

Answer: a
Explanation: The prime condition of AA-Tree is that only the right children can be red to eliminate possible restructuring cases.

8. Which of the following trees is similar to that of an AA-Tree?
a) Splay Tree
b) B+ Tree
c) AVL Tree
d) Red-Black Tree
View Answer

Answer: d
Explanation: AA- Tree is a small variation of Red-Black tree. AA-Trees overcome the complexity faced in performing insertion and deletion in Red-Black Trees.

9. What is the worst case analysis of an AA-Tree?
a) O(N)
b) O(log N)
c) O(N log N)
d) O(N^2)
View Answer

Answer: b
Explanation: The worst case analysis of an AA-Tree is mathematically found to be O(log N).

10. AA-Trees makes more rotations than a red-black tree.
a) True
b) False
View Answer

Answer: a
Explanation: AA- trees make more rotations than a red-black tree since only two shapes are considered for an AA-Tree whereas seven shapes are considered in Red-Black trees.

11. Who is the inventor of AA-Tree?
a) Arne Anderson
b) Daniel Sleator
c) Rudolf Bayer
d) Jon Louis Bentley
View Answer

Answer: a
Explanation: AA-tree is invented by Arne Anderson. Daniel Sleator invented Splay Tree. Rudolf Bayer invented a Red-Black tree. Jon Louis Bentley invented K-d tree.

12. What should be the condition for the level of a left node?
a) It should be less than or equal to that of its parent
b) It should be greater than that of its parent
c) It should be strictly less than that of its parent
d) The level should be equal to one
View Answer

Answer: c
Explanation: The level of a left node should be strictly less than that of its parent. The level of a right node is less than or equal to that of its parent.

13. Of the following rules that are followed by an AA-tree, which of the following is incorrect?
1- Only right children can be red
2- Procedures are coded recursively
3- Instead of storing colors, the level of a node is stored
4- There should not be any left children

a) 1
b) 2
c) 3
d) 4
View Answer

Answer: d
Explanation: In an AA-Tree, both left and right children can be present. The only condition is that only right children can be red.

14. In the given figure, find ‘?’.

a) left rotation
b) right rotation
c) insertion
d) deletion
View Answer

Answer: b
Explanation: B is initially the right child of X. It is then rotated right side and now, B is the left child of P.

15. Comparing the speed of execution of Red-Black trees and AA-trees, which one has the faster search time?
a) AA-tree
b) Red-Black tree
c) Both have an equal search time
d) It depends
View Answer

Answer: a
Explanation: Since an AA-tree tends to be flatter, AA-tree has a faster search time than a Red-Black tree.
Topic 58. AVL Tree

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “AVL Tree”.

1. What is an AVL tree?
a) a tree which is balanced and is a height balanced tree
b) a tree which is unbalanced and is a height balanced tree
c) a tree with three children
d) a tree with atmost 3 children
View Answer

Answer: a
Explanation: It is a self balancing tree with height difference atmost 1.

2. Why we need to a binary tree which is height balanced?
a) to avoid formation of skew trees
b) to save memory
c) to attain faster memory access
d) to simplify storing
View Answer

Answer: a
Explanation: In real world dealing with random values is often not possible, the probability that u are dealing with non random values(like sequential) leads to mostly skew trees, which leads to worst case. hence we make height balance by rotations.

3. Which of the below diagram is following AVL tree property?
i.
ii.
a) only i
b) only i and ii
c) only ii
d) i is not a binary search tree
View Answer

Answer: b
Explanation: The property of AVL tree is it is height balanced tree with difference of atmost 1 between left and right subtrees. All AVL trees are binary search tree.

4. What is the maximum height of an AVL tree with p nodes?

a) p
b) log(p)
c) log(p)/2
d) $^{p}/_{2}$
View Answer

Answer: b
Explanation: Consider height of tree to be ‘he’, then number of nodes which totals to p can be written in terms of height as N(he)=N(he-1)+1+N(he-2). since N(he) which is p can be written in terms of height as the beside recurrence relation which on solving gives N(he)= O(logp) as worst case height.

5. To restore the AVL property after inserting a element, we start at the insertion point and move towards root of that tree. is this statement true?
a) true
b) false
View Answer

Answer: a
Explanation: It is interesting to note that after insertion, only the path from that point to node or only that subtrees are imbalanced interms of height.

6. Given an empty AVL tree, how would you construct AVL tree when a set of numbers are given without performing any rotations?
a) just build the tree with the given input
b) find the median of the set of elements given, make it as root and construct the tree
c) use trial and error
d) use dynamic programming to build the tree
View Answer

Answer: b
Explanation: Sort the given input, find the median element among them, make it as root and construct left and right subtrees with elements lesser and greater than the median element recursively. this ensures the subtrees differ only by height 1.

7. What maximum difference in heights between the leafs of a AVL tree is possible?
a) log(n) where n is the number of nodes
b) n where n is the number of nodes
c) 0 or 1
d) atmost 1
View Answer

Answer: a
Explanation: At every level we can form a tree with difference in height between subtrees to be atmost 1 and so there can be log(n) such levels since height of AVL tree is log(n).

8. Consider the pseudo code:

```
int avl(binarysearchtree root):
   if(not root)
     return 0
   left_tree_height = avl(left_of_root)

   if(left_tree_height== -1)
     return left_tree_height

   right_tree_height= avl(right_of_root)

   if(right_tree_height==-1)
     return right_tree_height
```

Does the above code can check if a binary search tree is an AVL tree?
a) yes
b) no
View Answer

Answer: a
Explanation: The condition to check the height difference between left and right subtrees is missing. if (absolute(left_tree_height – right_tree_height)>1) must be added.

9. Consider the below left-left rotation pseudo code where the node contains value pointers to left, right child nodes and a height value and Height() function returns height value stored at a particular node.

```
avltree leftrotation(avltreenode z):
  avltreenode w =x-left
  x-left=w-right
  w-right=x
  x-height=max(Height(x-left),Height(x-right))+1
  w-height=max(missing)+1
 return w
```

What is missing?
a) Height(w-left), x-height
b) Height(w-right), x-height
c) Height(w-left), x
d) Height(w-left)
View Answer

Answer: a
Explanation: In the code we are trying to make the left rotation and so we need to find maximum of those two values.

10. Why to prefer red-black trees over AVL trees?
a) Because red-black is more rigidly balanced
b) AVL tree store balance factor in every node which costs space
c) AVL tree fails at scale
d) Red black is more efficient
View Answer

Answer: b
Explanation: Every node in an AVL tree need to store the balance factor (-1, 0, 1) hence space costs to O(n), n being number of nodes. but in red-black we can use the sign of number (if numbers being stored are only positive) and hence save space for storing balancing information. there are even other reasons where redblack is mostly prefered.
Topic 59. Cartesian Tree

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Cartesian Tree”.

1. What is a Cartesian tree?
a) a skip list in the form of tree
b) a tree which obeys cartesian product
c) a tree which obeys heap property and whose inorder traversal yields the given sequence
d) a tree which obeys heap property only
View Answer

Answer: c
Explanation: A tree with heap property (parent is either small or big than children) and when traversed in inorder yields the given input sequence. refer below diagram question for clarity.

2. Is the below tree representation of 50,100,400,300,280 correct way to represent cartesian tree?

a) true
b) false
View Answer

Answer: a
Explanation: A tree with heap property (parent is either small or big than children) and when traversed in inorder yields the given input sequence is called as a cartesian tree. as the above figure satisies both the properties. note that even min heap tree can be generated. the above is a max heap tree.

3. Which of the below statements are true?
i. Cartesian tree is not a height balanced tree
ii. Cartesian tree of a sequence of unique numbers can be unique generated
a) both statements are true
b) only i. is true
c) only ii. is true
d) both are false
View Answer

Answer: a
Explanation: A height balanced cartesian tree is not possible as seen in above question. also any time a unique sequnce possess a unique cartesian tree, this can be proven through induction.

4. What is the speciality of cartesian sorting?
a) it sorts partially sorted set of data quickly
b) it considers cartesian product of elements
c) it sorts elements in less than O(logn)
d) it is a self balancing tree

View Answer

Answer: a
Explanation: It can sort a set which requires only some sorting or displacements. for example consider 78, 79, 80, 82, 81, 83, In this only 81 and 82 must be swaped to make it a complete sorted set, in this case cartesian sort comes to the rescue.

5. Consider a sequence of numbers to have repetitions, how a cartesian tree can be constructed in such situations without violating any rules?
a) use any tie-breaking rule between repeated elements
b) cartesian tree is impossible when repetitions are present
c) construct a max heap in such cases
d) construct a min heap in such cases
View Answer

Answer: a
Explanation: Consider any of the tie breaking rules, for example the element which appears first can be taken as small among the same elements and then apply cartesian tree rules.

6. What happens if we apply the below operations on an input sequence?
i. construct a cartesian tree for input sequence
ii. put the root element of above tree in a priority queue
iii. if(priority queue is not empty) then
iv. search and delete minimum value in priority queue
v. add that to output
vi. add cartesian tree children of above node to priority queue
a) constructs a cartesian tree
b) sorts the input sequence
c) does nothing
d) produces some random output
View Answer

Answer: b
Explanation: The above given steps are for sorting a cartesian tree. cartesian sort is benificial in case of partially sorted set of elements. a cartesian sort can be considered as a selection or heap sort maintaing a priority queue.

7. Cartesian trees are most suitable for?
a) searching
b) finding nth element
c) minimum range query and lowest common ancestors
d) self balancing a tree
View Answer

Answer: c
Explanation: In a cartesian tree minimum value can be found by finding lowest common ancestor for the extreme elements. consider 11,9,19,16 the lowest element is 9 and is a lowest common ancestor for 11 and 16. and by applying few techniques cartesian tree can be used to even find lowest common ancestors efficiently.
these can be done in constant time. tree can be constructed in linear time (this is the most efficient time for any tree construction) and takes space as many elements are there.

8. A treap is a cartesian tree with ___________
a) additional value, which is a priority value to the key generated randomly
b) additional value, which is a priority value to the key generated sequentially
c) additional heap rule
d) additional operations like remove a range of elements
View Answer

Answer: a
Explanation: A cartesian tree, if feeded with a sorted sequence will generate a straight path (or in tree terminology a skew tree). moreover a cartesian tree basing on same values from the search keys doesnot work well. so a cartesian tree with priority value in addition to search key is called treap.

9. Cartesian trees solve range minimum query problem in constant time.
a) true
b) false
View Answer

Answer: a
Explanation: Range minmum query is finding the minimum element in a given subarray of an array. Constant time is achieved by storing the Cartesian trees for all the blocks in the array. Rmq's are used in string matchings, computing lowest common ancestor and longest common prefix of a sring.

10. Consider below sequences.

```
array=60 90 10 100 40 150 90
reverse 2 to 3
array=60 10 90 100 40 150 90
reverse 3 to 6
array= 60 100 150 40 100 90 90
  now printout from 1 to 6 :-- 60 100 150 40 100 90
```

How to achieve the above operation efficiently?
a) use linked lists
b) use avl trees
c) use red-black trees
d) use treaps (cartesian trees)
View Answer

Answer: d
Explanation: This can be solved efficiently using treap which is a modification of cartesian tree. an attribute like “boolean reverse” can be maintained with every node representing whether to reverse or not.
Topic 60. Weight Balanced Tree

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Weight Balanced Tree”.

1. What is a weight balanced tree?
a) A binary tree that stores the sizes of subtrees in nodes
b) A binary tree with an additional attribute of weight
c) A height balanced binary tree
d) A normal binary tree
View Answer

Answer: a
Explanation: Unlike AVL and redblack trees which uses height and color as book keeping information, weight balanced trees use the size of subtrees.

2. What are the applications of weight balanced tree?
a) dynamic sets, dictionaries, sequences, maps
b) heaps
c) sorting
d) storing strings
View Answer

Answer: a
Explanation: They are a type of self balancing trees which are mostly used in storing key-value pairs, which is mostly used in functional programming languages. they are very useful to maintain big set of ordered objects.

3. A node of the weight balanced tree has
a) key, left and right pointers, size
b) key, value
c) key, size
d) key
View Answer

Answer: a
Explanation: As a weight balanced tree stores height of the subtrees, we need to use size as an additional attribute to every node. also value(for mappings) may be an optional attribute.

4. The size value of various nodes in a weight balanced tree are
leaf – zero
internal node – size of it’s two children
is this true?
a) true
b) false
View Answer

Answer: a
Explanation: Size of a node k is size[k] = size[k.left] + 1 + size[k.right] and based on this the weight will be given as weight[k] = size[k] + 1.

5. What is the condition for a tree to be weight balanced. where a is factor and n is a node?
a) weight[n.left] >= a*weight[n] and weight[n.right] >= a*weight[n].
b) weight[n.left] >= a*weight[n.right] and weight[n.right] >= a*weight[n].
c) weight[n.left] >= a*weight[n.left] and weight[n.right] >= a*weight[n].

d) weight[n] is a non zero
View Answer

Answer: a
Explanation: The tree is said to be a-balanced if the condition is satisfied. and ‘a’ value will be determined during tree formation. large value of ‘a’ is more effective.

6. What are the operations that can be performed on weight balanced tree?
a) all basic operations and set intersection, set union and subset test
b) all basic operations
c) set intersection, set union and subset test
d) only insertion and deletion
View Answer

Answer: a
Explanation: The speciality of a weight balanced tree is a part from basic operations we can perform collective operations like set intersection, which helps in rapid prototyping in functional programming languages.

7. Consider a weight balanced tree such that, the number of nodes in the left sub tree is at least half and at most twice the number of nodes in the right sub tree. The maximum possible height (number of nodes on the path from the root to the farthest leaf) of such a tree on k nodes can be described as
a) log2 n
b) log4/3 n
c) log3 n
d) log3/2 n
View Answer

Answer: d
Explanation: Total number of nodes can be described by the recurrence T(n) = T((n-1)/3)) + T(2(n-1)/3) + 1 T(1) = 1. height of the tree will be H(n) = H(2/3(n-1)) + 1, H(1). drawing a recurrence tree and the cost at each level is 1 and the height will be log(3/2)n.

8. Why the below pseudo code where x is a value, wt is weight factor and t is root node can’t insert?

```
WeightBalanceTreeNode insert(int x, int wt, WeightBalanceTreeNode k) :

        if (k == null)
            k = new WeightBalanceTreeNode(x, wt, null, null)
        else if (x < t.element) :

            k.left = insert (x, wt, k.left)
            if (k.left.weight < k.weight)
                k = rotateWithRightChild (k)

         else if (x > t.element) :

            k.right = insert (x, wt, k.right)
            if (k.right.weight < k.weight)
                k = rotateWithLeftChild (k)
```

a) when x>t. element Rotate-with-left-child should take place and vice versa
b) the logic is incorrect
c) the condition for rotating children is wrong
d) insertion cannot be performed in weight balanced trees
View Answer

Answer: a
Explanation: The rotations of children must be interchanged in the code.

9. What does the below definations convey?
i. A binary tree is balanced if for every node it is gonna hold that the number of inner nodes in the left subtree and the number of inner nodes in the right subtree differ by at most 1.
ii. A binary tree is balanced if for any two leaves the difference of the depth is at most 1.
a) weight balanced and height balanced tree definations
b) height balanced and weight balanced tree definations
c) definations of weight balanced tree
d) definations of height balanced tree
View Answer

Answer: a
Explanation: They are the definations of weight and height balanceness. height balanced trees wont convey weight balanceness but opposite can be true.

10. Elements in a tree can be indexed by its position under the ordering of the keys and the ordinal position of an element can be determined, both with good efficiency.
a) true
b) false
View Answer

Answer: a
Explanation: In a weight balanced tree we can even store the key information so as to use as a key value pair.
Topic 61. Red Black Tree

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Red Black Tree”.

1. What is the special property of red-black trees and what root should always be?
a) a color which is either red or black and root should always be black color only
b) height of the tree
c) pointer to next node
d) a color which is either green or black
View Answer

Answer: a
Explanation: An extra attribute which is a color red or black is used. root is black because if it is red then one of red-black tree property which states that number of black nodes from root to null nodes must be same, will be violated.

2. Why do we impose restrictions like
. root property is black
. every leaf is black
. children of red node are black
. all leaves have same black
a) to get logarithm time complexity
b) to get linear time complexity
c) to get exponential time complexity
d) to get constant time complexity
View Answer

Answer: a
Explanation: We impose such restrictions to achieve self balancing trees with logarithmic complexities for insertions, deletions, search.

3. Cosider the below formations of red-black tree.

All the above formations are incorrect for it to be a redblack tree. then what may be the correct order?
a) 50-black root, 18-red left subtree, 100-red right subtree
b) 50-red root, 18-red left subtree, 100-red right subtree
c) 50-black root, 18-black left subtree, 100-red right subtree
d) 50-black root, 18-red left subtree, 100-black right subtree
View Answer

Answer: a
Explanation: Considering all the properties of red-black tree, 50 must be the black root and there are two possibilities for subtrees. one is option “50-black root, 18-red left subtree, 100-red right subtree” and other is making all nodes of the tree to be black.

4. What are the operations that could be performed in O(logn) time complexity by red-black tree?
a) insertion, deletion, finding predecessor, successor
b) only insertion
c) only finding predecessor, successor
d) for sorting
View Answer

Answer: a
Explanation: We impose restrictions to achieve logarithm time complexities.
impose restrictions are:
. root property is black
. every leaf is black
. children of red node are black
. all leaves have same black.

5. Which of the following is an application of Red-black trees and why?
a) used to store strings efficiently
b) used to store integers efficiently

c) can be used in process schedulers, maps, sets
d) for efficient sorting
View Answer

Answer: c
Explanation: RB tree is used for Linux kernel in the form of completely fair scheduler process scheduling algorithm. It is used for faster insertions, retrievals.

6. When it would be optimal to prefer Red-black trees over AVL trees?
a) when there are more insertions or deletions
b) when more search is needed
c) when tree must be balanced
d) when log(nodes) time complexity is needed
View Answer

Answer: a
Explanation: Though both trees are balanced, when there are more insertions and deletions to make the tree balanced, AVL trees should have more rotations, it would be better to use red-black. but if more search is required AVL trees should be used.

7. Why Red-black trees are preferred over hash tables though hash tables have constant time complexity?
a) no they are not preferred
b) because of resizing issues of hash table and better ordering in redblack trees
c) because they can be implemented using trees
d) because they are balanced
View Answer

Answer: b
Explanation: Redblack trees have O(logn) for ordering elements in terms of finding first and next elements. also whenever table size increases or decreases in hash table you need to perform rehashing which can be very expensive in real time. also red black stores elements in sorted order rather than input order.

8. How can you save memory when storing color information in Red-Black tree?
a) using least significant bit of one of the pointers in the node for color information
b) using another array with colors of each node
c) storing color information in the node structure
d) using negative and positive numbering
View Answer

Answer: a
Explanation: The node pointers can be used to store color with the help of significant bits. the exceptions of this method are in languages like java where pointers are not used this may not work.

9. When to choose Red-Black tree, AVL tree and B-trees?
a) many inserts, many searches and when managing more items respectively
b) many searches, when managing more items respectively and many inserts respectively
c) sorting, sorting and retrieval respectively
d) retrieval, sorting and retrieval respectively
View Answer

Answer: a
Explanation: Red black when frequent inserts and deletes, AVL when less frequent inserts and deletes, B-tree when using paging from a slow storage device.

10. What is the below pseudo code trying to do, where pt is a node pointer and root pointer?

```
redblack(Node root, Node pt) :
  if (root == NULL)
     return pt

  if (pt.data < root.data)
  {
      root.left =   redblack(root.left, pt);
      root.left.parent = root
  }
  else if (pt.data > root.data)
  {
      root.right = redblackt(root.right, pt)
      root.right.parent = root
  }
 return root
```

a) insert a new node

b) delete a node
c) search a node
d) count the number of nodes
View Answer

Answer: a
Explanation: The code is taking the root node and to be inserted node and is performing insertion operation.
Topic 62. Top Tree

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Top Tree”.

1. Which algorithm is used in the top tree data structure?
a) Divide and Conquer
b) Greedy
c) Backtracking
d) Branch
View Answer

Answer: a
Explanation: Top tree is a type of data structure which is based on unrooted dynamic binary tree and is used to solve path related problems. It allows an algorithm called divide and conquer.

2. For how many vertices in a set, is top tree defined for underlying tree?
a) 3
b) 4
c) 5
d) 2
View Answer

Answer: d
Explanation: Top tree is defined for a set having a maximum of 2 vertices for its underlying tree. Those sets having at maximum 2 vertices is called External Boundary Vertices.

3. How many edges are present in path cluster?
a) 2
b) 3
c) 6
d) 1
View Answer

Answer: a
Explanation: There are at least 2 edges present in path cluster. Cluster in data structure is defined as the subtree that is connect having maximum of 2 vertices known as Boundary Vertices.

4. How many edges does a leaf cluster contain?
a) 0
b) 1
c) 2
d) 3
View Answer

Answer: a
Explanation: If a cluster has no edges and contains only one vertex known as boundary vertex then, it is known as leaf cluster. So a leaf cluster doesn’t contain any edges. It is also known as Point cluster.

5. How many edges are present in Edge cluster?
a) 0
b) 1
c) 2
d) 4
View Answer

Answer: b
Explanation: A cluster containing only single edge is known as Edge cluster. So there are in total 1 edge present in edge cluster. Cluster in data structure is defined as the subtree that is connect having maximum of 2 vertices known as Boundary Vertices.

6. Which data structure is used to maintain a dynamic forest using a link or cut operation?
a) Top Tree
b) Array

c) Linked List
d) Stack
View Answer

Answer: a
Explanation: Top tree data structure is used to maintain a dynamic forest using link or cut operations. Top tree is a type of data structure which is based on unrooted dynamic binary tree and is used to solve path related problems.

7. If A ▯ B (A and B are two clusters) is a singleton set then it is a Merge able cluster.
a) True
b) False
View Answer

Answer: a
Explanation: If A ▯ B is a singleton set where A and B are two clusters, that is there are only one node that is common between the clusters then they are known as Merge able cluster.

8. Is Top tree used for maintaining Dynamic set of trees called forest.
a) True
b) False
View Answer

Answer: a
Explanation: Top tree data structure is used to maintain a dynamic forest using link or cut operations. Top tree is a type of data structure which is based on unrooted dynamic binary tree and is used to solve path related problems.

9. What is the time complexity for the initialization of top tree?
a) O (n)
b) O (n^2)
c) O (log n)
d) O (n!)
View Answer

Answer: a
Explanation: Generally, trees have weight on its edges. Also there is one to one correspondence of the edges with the top trees. Therefore, top trees can be initialized in O (n) time.

10. How many top trees are there in a tree with single vertex?
a) 0
b) 1
c) 2
d) 3
View Answer

Answer: a
Explanation: Tree having a single vertex has no clusters of tree present in the structure. Therefore, there are empty top trees in a tree having a single vertex. Trees with one node are single node.

11. Which property makes top tree a binary tree?
a) Nodes as Cluster
b) Leaves as Edges
c) Root is Tree Itself
d) All of the mentioned
View Answer

Answer: d
Explanation: Top tree can be considered as a binary tree if the nodes form a cluster, leaves act as an edge and the root of the top tree acts as a tree itself. Then the top tree is called binary tree.

12. Which of the dynamic operations are used in Top Tree data structure implementation?
a) Link
b) Cut
c) Expose
d) All of the mentioned
View Answer

Answer: d
Explanation: Link returns a single tree having different vertices from top trees. Cut removes the edge from the top tree. Expose is used to implement queries on top trees. Hence all of the options are used as dynamic operations.

13. Which of the following are used as an internal operation in Top tree?
a) Merge
b) Cut
c) Expose
d) Link
View Answer

Answer: a
Explanation: Link returns a single tree having different vertices from top trees. Cut removes the edge from the top tree. Expose is used to implement queries on top trees. While merge is an internal operation used to merge two clusters and return as a parent cluster.

14. What is the time complexity for maintaining a dynamic set of weighted trees?
a) O (n)
b) O (n^2)
c) O (log n)
d) O (n!)
View Answer

Answer: c
Explanation: A lot of applications have been implemented using Top tree interface. Maintaining a dynamic set of weighted trees is one such application which can be implemented with O (log n) time complexity.
Topic 63. Splay Tree

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Splay Tree”.

1. What are splay trees?
a) self adjusting binary search trees
b) self adjusting binary trees
c) a tree with strings
d) a tree with probability distributions
View Answer

Answer: a
Explanation: Splay trees are height balanced, self adjusting BST’s.

2. Which of the following property of splay tree is correct?
a) it holds probability usage of the respective sub trees
b) any sequence of j operations starting from an empty tree with h nodes atmost, takes O(jlogh) time complexity
c) sequence of operations with h nodes can take O(logh) time complexity
d) splay trees are unstable trees
View Answer

Answer: b
Explanation: This is a property of splay tree that ensures faster access. we push the most recently used nodes to top which leads to faster access to recently used values.

3. Why to prefer splay trees?
a) easier to program
b) space efficiency
c) easier to program and faster access to recently accessed items
d) quick searching
View Answer

Answer: c
Explanation: Whenever you insert an element or remove or read an element that will be pushed or stored at the top which facilitates easier access or recently used stuff.

4. Is it true that splay trees have O(logn) amortized complexity ?
a) true
b) false
View Answer

Answer: a
Explanation: We go with amortized time complexity when we feel that not all operations are worst and some can be efficiently done. in splay trees not all splay operations will lead to O(logn) worst case complexity.

5. What is a splay operation?
a) moving parent node to down of child
b) moving a node to root

c) moving root to leaf
d) removing leaf node
View Answer

Answer: b
Explanation: Splay trees mainly work using splay operations. wheneve we insert, delete and search for a node we splay the respective nodes to root. we have zig-zag and zig-zig operations.

6. Which of the following options is an application of splay trees?
a) cache Implementation
b) networks
c) send values
d) receive values
View Answer

Answer: a
Explanation: Splay trees can be used for faster access to recently accessed items and hence used for cache implementations.

7. When we have red-black trees and AVL trees that can perform most of operations in logarithmic times, then what is the need for splay trees?
a) no there is no special usage
b) In real time it is estimated that 80% access is only to 20% data, hence most used ones must be easily available
c) redblack and avl are not upto mark
d) they are just another type of self balancing binary search trees
View Answer

Answer: b
Explanation: May be the stats showing 80-20% may be not accurate, but in real time that is the widely spread scenario seen. If you are into this type of situation, you must choose implementing splay trees.

8. After the insertion operation, is the resultant tree a splay tee?

a) true
b) false
View Answer

Answer: a
Explanation: There is a zig-zag and right operation(zig) which gives the right hand side tree. refer splay operations for insertion in splay tree.

9. What output does the below pseudo code produces?

```
Tree_node function(Tree_node x)
{
    Tree_node y = x.left;
    x.left = y.right;
    y.right = x;
    return y;
}
```

a) right rotation of subtree
b) left rotation of subtree
c) zig-zag operation
d) zig-zig operation
View Answer

Answer: a
Explanation: When a right rotation is done the parent of the rotating node becomes it's right node and it's child becomes it's left child.

10. What is the disadvantage of using splay trees?
a) height of a splay tree can be linear when accessing elements in non decreasing order.
b) splay operations are difficult
c) no significant disadvantage
d) splay tree performs unnecessary splay when a node is only being read
View Answer

Answer: a
Explanation: This will be the case after accessing all n elements in non-decreasing order. Since the height of a tree corresponds to the worst-case access time, this means that the actual cost of an operation can be high. However the amortized access cost of this worst case is logarithmic O(log n).

Topic 64. Treap

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Treap”.

1. What is the space complexity of a treap algorithm?
a) O(N)
b) O(log N)
c) O(N log N)
d) O(N^2)
View Answer

Answer: a
Explanation: The average case and worst case space complexity of a treap is mathematically found to be O(N).

2. A treap is a combination of a tree and a heap.
a) false
b) true
View Answer

Answer: b
Explanation: A treap is a combination of a tree and a heap. The structure of a treap is determined by the fact that it is heap-ordered.

3. Which is the simplest of all binary search trees?
a) AVL tree
b) Treap
c) Splay tree
d) Binary heap
View Answer

Answer: b
Explanation: A treap is the simplest of all binary search trees. Each node is given a numeric priority and implementation is non recursive.

4. What is the reason behind the simplicity of a treap?
a) Each node has data and a pointer
b) Each node is colored accordingly
c) It is a binary search tree following heap principles
d) Each node has a fixed priority field
View Answer

Answer: d
Explanation: A treap is the simplest of all because we don’t have to worry about adjusting the priority of a node.

5. What is the condition for priority of a node in a treap?
a) a node’s priority should be greater than its parent
b) a node’s priority should be at least as large as its parent
c) the priority is randomly assigned and can have any value
d) a node’s priority is always given in decreasing order
View Answer

Answer: b
Explanation: A node’s priority should satisfy heap order. That is, any node’s priority should be at least as large as its parent.

6. Several other operations like union set difference and intersection can be done in treaps.
a) True
b) False
View Answer

Answer: a
Explanation: Other than insertion, deletion and search operations, several operations like union, intersection and set difference can be done in treaps.

7. What is the average running time of a treap?
a) O(N)
b) O(N log N)
c) O(log N)
d) O(M log N)
View Answer

Answer: c

Explanation: The average case and worst case analysis of a treap are mathematically found to be O(log N).

8. Which node has the lowest priority in a treap?
a) root node
b) leaf node
c) null node
d) centre node
View Answer

Answer: a
Explanation: A root node has the lowest priority in a treap since the node’s priority is based on heap order.

9. What is the priority of a null node?
a) 1
b) 0
c) random number
d) infinity
View Answer

Answer: d
Explanation: The priority of a null node is set to be infinity in a treap so that during deletion, priority of that particular node is set to infinity, rotated and freed.

10. Who invented treaps?
a) Cecilia and Raimund
b) Arne Andersson
c) Donald Shell
d) Harris and Ross
View Answer

Answer: a
Explanation: Cecilia and Raimund invented Treaps. Arne Andersson invented AA – Trees. Donald Shell invented shell sort and Harris and Ross formulated maximum flow problem.
Topic 65. Threaded Binary Tree

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Threaded Binary Tree”.

1. What is a threaded binary tree traversal?
a) a binary tree traversal using stacks
b) a binary tree traversal using queues
c) a binary tree traversal using stacks and queues
d) a binary tree traversal without using stacks and queues
View Answer

Answer: d
Explanation: This type of tree traversal will not use stack or queue.

2. What are the disadvantages of normal binary tree traversals?
a) there are many pointers which are null and thus useless
b) there is no traversal which is efficient
c) complexity in implementing
d) improper traversals
View Answer

Answer: a
Explanation: As there are majority of pointers with null value going wasted we use threaded binary trees.

3. In general, the node content in a threaded binary tree is ________
a) leftchild_pointer, left_tag, data, right_tag, rightchild_pointer
b) leftchild_pointer, left_tag
c) leftchild_pointer, left_tag, right_tag, rightchild_pointer
d) leftchild_pointer, left_tag, data
View Answer

Answer: a
Explanation: It contains additional 2 pointers over normal binary tree node structure.

4. What are null nodes filled with in a threaded binary tree?
a) inorder predecessor for left node and inorder successor for right node information
b) right node with inorder predecessor and left node with inorder successor information
c) they remain null

d) some other values randomly
View Answer

Answer: a
Explanation: If preorder or postorder is used then the respective predecessor and successor info is stored.

5. Which of the following tree traversals work if the null left pointer pointing to the predecessor and null right pointer pointing to the successor in a binary tree?
a) inorder, postorder, preorder traversals
b) inorder
c) postorder
d) preorder
View Answer

Answer: a
Explanation: In threaded binary trees, the null left pointer points to the predecessor and the right null pointer point to the successor. In threaded binary trees, we can use in-order, preorder and postorder traversals to visit every node in the tree.

6. What are double and single threaded trees?
a) when both left, right nodes are having null pointers and only right node is null pointer respectively
b) having 2 and 1 node
c) using single and double linked lists
d) using heaps and priority queues
View Answer

Answer: a
Explanation: They are properties of double and single threaded binary trees respectively.

7. What is wrong with below code for inorder traversal of inorder threaded binary tree:

```
inordertraversal(threadedtreenode root):
threadedtreenode q = inorderpredecessor(root)
while(q!=root):
q=inorderpredecessor(q)
print q.data
```

a) inordersuccessor instead of inorderpredecessor must be done
b) code is correct
c) it is code for post order
d) it is code for pre order
View Answer

Answer: a
Explanation: Property of inorder threaded binary tree is left node with inorder predecessor and right node with inorder successor information are stored.

8. What is inefficient with the below threaded binary tree picture?

a) it has dangling pointers
b) nothing inefficient
c) incorrect threaded tree
d) space is being used more
View Answer

Answer: a
Explanation: The nodes extreme left and right are pointing to nothing which could be also used efficiently.
Topic 66. Tango Tree

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Tango Tree”.

1. Who developed the concept of tango tree?
a) Erik Demaine
b) Mihai Patrascu
c) John Lacono
d) All of the mentioned
View Answer

Answer: d
Explanation: Erik Demaine is a well-known professor of Computer Science at MIT. John Lacono is an American computer scientist specialized in data structure and algorithm while Mihai Patrascu was a Romanian- American computer scientist. All of them together developed the concept of Tango tree.

2. Which type of tree is tango tree?
a) Ternary Tree
b) AVL Tree
c) Binary Search Tree
d) K-ary Tree
View Answer

Answer: c
Explanation: Tango tree is an example of binary search tree which was developed by four famous scientists Erik Demaine, Mihai Patrascu, John Lacono and Harmon in the year 2004.

3. After which city is tango tree named?
a) Vatican City
b) Buenos Aires
c) New York
d) California
View Answer

Answer: b
Explanation: Tango is a popular couple dance or partner dance that was originated in the 1880s somewhere between Argentina and Uruguay. Buenos Aires is a capital city off Argentina. Hence they named after Buenos Aires.

4. Which type of binary search tree or algorithm does tango tree use?
a) Online
b) Offline
c) Static
d) Dynamic
View Answer

Answer: a
Explanation: Tango tree is an online binary search tree whose time complexity is O (log (log n)) when compared to the time complexity of offline binary search tree model. Online algorithm processes input or data provided piece by piece.

5. What is the time complexity of for achieving competitive ratio by tango tree?
a) O (log n)
b) O (n^2)
c) O (n!)
d) O (log (log n))
View Answer

Answer: d
Explanation: Tango tree is an online binary search tree whose time complexity is O (log (log n)) when compared to the time complexity of offline binary search tree model. Online algorithm processes input or data provided piece by piece.

6. Which type of binary search tree is imitated for construction of tango tree?
a) Complete Binary Search Tree
b) Perfect Binary Search Tree
c) Balanced Binary Search Tree
d) Degenerate Binary Search Tree
View Answer

Answer: a
Explanation: Tango tree is constructed by simulating a complete binary search tree. This tree is also known as Reference tree, that contains all the elements of the tree. Also, the reference tree is never showed in actual implementation.

7. Which special balanced binary search tree is used to store the nodes of auxiliary tree?
a) Red – Black Tree
b) Red – Brown Tree
c) Red – Yellow Tree
d) Red – Tango Tree
View Answer

Answer: a
Explanation: The path starting from the root and following the path of preferred child node till the end of leaf node is known as preferred path. Nodes are stored in Red – Black tree for the representation of the preferred path.

8. Is tango tree represented as a tree of trees.
a) True
b) False
View Answer

Answer: a
Explanation: Partitioning method is used by tango tree which partitions a binary search tree into small sets of paths and then storing them to auxiliary trees. Hence tango tree is represented as a tree of trees.

9. Which operation is used to combine two auxiliary trees?
a) Join
b) Combinatorial
c) Add
d) Concatenation
View Answer

Answer: a
Explanation: If the top node of one of the reference tree amongst the two, is the is the child of the bottom node of the other reference tree, then the join operation can be carried out to join the two auxiliary trees.

10. Is partitioning method used by Tango Tree.
a) True
b) False
View Answer

Answer: a
Explanation: Partitioning method is used by tango tree which partitions a binary search tree into small sets of paths and then storing them to auxiliary trees. Hence tango tree is represented as a tree of trees.

11. Which operation is used to break a preferred path into two sets of parts at a particular node?
a) Differentiate
b) Cut
c) Integrate
d) Join
View Answer

Answer: b
Explanation: A preferred path is broken into two parts. One of them is known as top part while other is known as bottom part. To break a preferred path into two sets, cut operation is used at a particular node.

12. What is the upper bound for a tango tree if k is a number of interleaves?
a) k+2 O (log (log n))
b) k O (log n)
c) K^2 O (log n)
d) k+1 O (log (log n))
View Answer

Answer: d
Explanation: Upper bound is found to analyze the work done by a tango tree on a given set of sequences. In order to connect to the tango tree, the upper bound is found to be k+1 O (log (log n)).

13. What is the time complexity for searching k+1 auxiliary trees?
a) k+2 O (log (log n))
b) k+1 O (log n)
c) K+2 O (log n)
d) k+1 O (log (log n))
View Answer

Answer: d
Explanation: Since each search operation in the auxiliary tree takes O (log (log n)) time as auxiliary tree size is bounded by the height of the reference tree that is log n. So for k+1 auxiliary trees, total search time is k+1 O (log (log n)).

14. What is the time complexity for the update cost on auxiliary trees?
a) O (log (log n))
b) k-1 O (log n)
c) K^2 O (log n)
d) k+1 O (log (log n))
View Answer

Answer: d
Explanation: The update cost also is bounded by the upper bound. We perform one cut as well as one join operation for the auxiliary tree, so the total update cost for the auxiliary tree is found to be k+1 O (log (log n)).

15. Which of the following is the self-adjusting binary search tree?
a) AVL Tree

b) Splay Tree
c) Top Tree
d) Ternary Tree
View Answer

Answer: b
Explanation: Splay tree is a self – adjusting binary search tree. It performs basic operations on the tree like insertion, deletion, loop up performing all these operations in O (log n) time.
Topic 67. Rope

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Rope”.

1. Which of the following is also known as Rope data structure?
a) Cord
b) String
c) Array
d) Linked List
View Answer

Answer: a
Explanation: Array is a linear data structure. Strings are a collection and sequence of codes, alphabets or characters. Linked List is a linear data structure having a node containing data input and the address of the next node. The cord is also known as the rope data structure.

2. Which type of data structure does rope represent?
a) Array
b) Linked List
c) Queue
d) Binary Tree
View Answer

Answer: d
Explanation: Rope is a special binary tree in which the end nodes contain the string and its length. The array is a linear data structure. Linked List is a linear data structure having a node containing data input and the address of the next node. The queue is a data structure working on the principle of FIFO.

3. What is the time complexity for finding the node at x position where n is the length of the rope?
a) O (log n)
b) O (n!)
c) O (n^2)
d) O (1)
View Answer

Answer: a
Explanation: In order to find the node at x position in a rope data structure where N is the length of the rope, we start a recursive search from the root node. So the time complexity for worst case is found to be O (log N).

4. What is the time complexity for creating a new node and then performing concatenation in the rope data structure?
a) O (log n)
b) O (n!)
c) O (n^2)
d) O (1)
View Answer

Answer: d
Explanation: In order to perform the concatenation on the rope data structure, one can create two nodes S1 and S2 and then performing the operation in constant time that is the time complexity is O (1).

5. What is the time complexity for splitting the string into two new string in the rope data structure?
a) O (n^2)
b) O (n!)
c) O (log n)
d) O (1)
View Answer

Answer: c
Explanation: In order to perform the splitting on the rope data structure, one can split the given string into two new string S1 and S2 in O (log n) time. So, the time complexity for worst case is O (log n).

6. Which type of binary tree does rope require to perform basic operations?

a) Unbalanced
b) Balanced
c) Complete
d) Full
View Answer

Answer: b
Explanation: To perform the basic operations on a rope data structure like insertion, deletion, concatenation and splitting, the rope should be a balanced tree. After performing the operations one should again re-balance the tree.

7. What is the time complexity for inserting the string and forming a new string in the rope data structure?
a) O (log n)
b) O (n!)
c) O (n^2)
d) O (1)
View Answer

Answer: a
Explanation: In order to perform the insertion on the rope data structure, one can insert the given string at any position x to form a new string in O (log n) time. So, the time complexity for worst case is O (log n). This can be done by one split operation and two concatenation operations.

8. Is insertion and deletion operation faster in rope than an array?
a) True
b) False
View Answer

Answer: a
Explanation: In order to perform the insertion on the rope data structure, the time complexity is O (log n). In order to perform the deletion on the rope data structure, the time complexity for worst case is O (log n). While for arrays the time complexity is O (n).

9. What is the time complexity for deleting the string to form a new string in the rope data structure?
a) O (n^2)
b) O (n!)
c) O (log n)
d) O (1)
View Answer

Answer: c
Explanation: In order to perform the deletion on the rope data structure, one can delete the given string at any position x to form a new string in O (log n) time. So, the time complexity for worst case is O (log n). This can be done by two split operations and one concatenation operation.

10. Is it possible to perform a split operation on a string in the rope if the split point is in the middle of the string.
a) True
b) False
View Answer

Answer: a
Explanation: In order to perform the splitting on the rope data structure, one can split the given string into two new string S1 and S2 in O (log n) time. So, the time complexity for worst case is O (log n). The split operation can be performed if the split point is either at the end of the string or in the middle of the string.
Topic 68. B-Tree

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “B-Tree”.

1. Which of the following is the most widely used external memory data structure?
a) AVL tree
b) B-tree
c) Red-black tree
d) Both AVL tree and Red-black tree
View Answer

Answer: b
Explanation: In external memory, the data is transferred in form of blocks. These blocks have data valued and pointers. And B-tree can hold both the data values and pointers. So B-tree is used as an external memory data structure.

2. B-tree of order n is a order-n multiway tree in which each non-root node contains __________
a) at most (n – 1)/2 keys

b) exact (n – 1)/2 keys
c) at least 2n keys
d) at least (n – 1)/2 keys
View Answer

Answer: d
Explanation: A non-root node in a B-tree of order n contains at least (n – 1)/2 keys. And contains a maximum of (n – 1) keys and n sons.

3. A B-tree of order 4 and of height 3 will have a maximum of _______ keys.
a) 255
b) 63
c) 127
d) 188
View Answer

Answer: a
Explanation: A B-tree of order m of height h will have the maximum number of keys when all nodes are completely filled. So, the B-tree will have n = (m^{h+1} – 1) keys in this situation. So, required number of maximum keys = 4^{3+1} – 1 = 256 – 1 = 255.

4. Five node splitting operations occurred when an entry is inserted into a B-tree. Then how many nodes are written?
a) 14
b) 7
c) 11
d) 5
View Answer

Answer: c
Explanation: If s splits occur in a B-tree, 2s + 1 nodes are written (2 halves of each split and the parent of the last node split). So, if 5 splits occurred, then 2 * 5 + 1, i.e. 11 nodes are written.

5. B-tree and AVL tree have the same worst case time complexity for insertion and deletion.
a) True
b) False
View Answer

Answer: a
Explanation: Both the B-tree and the AVL tree have O(log n) as worst case time complexity for insertion and deletion.

6. 2-3-4 trees are B-trees of order 4. They are an isometric of _____ trees.
a) AVL
b) AA
c) 2-3
d) Red-Black
View Answer

Answer: d
Explanation: 2-3-4 trees are isometric of Red-Black trees. It means that, for every 2-3-4 tree, there exists a Red-Black tree with data elements in the same order.

7. Figure shown below is B-tree of order 5. What is the result of deleting 130 from the tree?

a)
b)
c)
d)
View Answer

Answer: c
Explanation: Each non-root in a B-tree of order 5 must contain at least 2 keys. Here, when the key 130 is deleted the node gets underflowed i.e. number of keys in the node drops below 2. So we combine the node with key 110 with it's brother node having keys 144 and 156. And this combined node will also contain the separator key from parent i.e. key 140, leaving the root with two keys 110 and 160.

8. What is the best case height of a B-tree of order n and which has k keys?
a) logn (k+1) – 1
b) nk
c) $\log_k$ (n+1) – 1
d) $k\log_n$
View Answer

Answer: a
Explanation: B-tree of order n and with height k has best case height h, where h = logn (k+1) – 1. The best case occurs when all the nodes are completely filled with keys.

9. Compression techniques can be used on the keys to reduce both space and time requirements in a B-tree.
a) True
b) False
View Answer

Answer: a
Explanation: The front compression and the rear compression are techniques used to reduce space and time requirements in B-tree. The compression enables to retain more keys in a node so that the number of nodes needed can be reduced.

10. Which of the following is true?
a) larger the order of B-tree, less frequently the split occurs
b) larger the order of B-tree, more frequently the split occurs
c) smaller the order of B-tree, more frequently the split occurs
d) smaller the order of B-tree, less frequently the split occurs
View Answer

Answer: a
Explanation: The average probability of the split is 1/(⌈m / 2⌉ – 1), where m is the order of B-tree. So, if m larger, the probability of split will be less.
Topic 69. B+ Tree

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “B+ Tree”.

1. In a B+ tree, both the internal nodes and the leaves have keys.
a) True
b) False
View Answer

Answer: b
Explanation: In a B+ -tree, only the leaves have keys, and these keys are replicated in non-leaf nodes for defining the path for locating individual records.

2. Which of the following is true?
a) B + tree allows only the rapid random access
b) B + tree allows only the rapid sequential access
c) B + tree allows rapid random access as well as rapid sequential access
d) B + tree allows rapid random access and slower sequential access
View Answer

Answer: c
Explanation: The B+ -tree being a variation of B-tree allows rapid random access. In a B+ -tree the leaves are linked together, so it also provides rapid sequential access.

3. A B+ tree can contain a maximum of 7 pointers in a node. What is the minimum number of keys in leaves?
a) 6
b) 3
c) 4
d) 7
View Answer

Answer: b
Explanation: Maximum number of pointers in a node is 7, i.e. the order of the B+ -tree is 7. In a B+ tree of order n each leaf node contains at most n – 1 key and at least ⌈(n − 1)/2⌉ keys. Therefore, a minimum number of keys each leaf can have = ⌈(7 – 1)/2⌉ = 3.

4. Which of the following is false?
a) A B+ -tree grows downwards
b) A B+ -tree is balanced
c) In a B+ -tree, the sibling pointers allow sequential searching
d) B+ -tree is shallower than B-tree
View Answer

Answer: a
Explanation: A B+ -tree always grows upwards. And In a B+tree – i)The path from the root to every leaf node is of the same length, so the tree is balanced. ii) Leaves are linked, so allow sequential searching. iii) An index is built with a single key per block of data rather than with one key per data record, so it is shallower than B-tree.

5. A B+ -tree of order 3 is generated by inserting 89, 9 and 8. The generated B+ -tree is __________
a)
b)
c)
d)
View Answer

Answer: b
Explanation:

6. Statement 1: When a node is split during insertion, the middle key is promoted to the parent as well as retained in right half-node.
Statement 2: When a key is deleted from the leaf, it is also deleted from the non-leaf nodes of the tree.
a) Statement 1 is true but statement 2 is false
b) Statement 2 is true but statement 1 is false
c) Both the statements are true
d) Both the statements are false
View Answer

Answer: a
Explanation: During the split, the middle key is retained in the right half node and also promoted to parent node. When a key is deleted from the leaf, it is retained in non-leaves, because it can be still a valid separator between keys in nodes below.

7. Efficiency of finding the next record in B+ tree is ____
a) O(n)
b) O(log n)
c) O(nlog n)
d) O(1)
View Answer

Answer: d
Explanation: In a B+ -tree finding the next recored (successor) involves accessing an additional leaf at most. So, the efficiency of finding the next record is O(1).

8. What is the maximum number of keys that a B+ -tree of order 3 and of height 3 have?
a) 3
b) 80
c) 27
d) 26
View Answer

Answer: d
Explanation: A B+ tree of order n and height h can have at most n^h – 1 keys. Therefore maximum number of keys = 3^3 -1 = 27 -1 = 26.

9. Which of the following is false?
a) Compared to B-tree, B+ -tree has larger fanout
b) Deletion in B-tree is more complicated than in B+ -tree
c) B+ -tree has greater depth than corresponding B-tree
d) Both B-tree and B+ -tree have same search and insertion efficiencies
View Answer

Answer: c
Explanation: A B+ -tree has larger fanout and therefore have a depth smaller than that of corresponding B-tree.

10. Which one of the following data structures are preferred in database-system implementation?
a) AVL tree
b) B-tree
c) B+ -tree
d) Splay tree
View Answer

Answer: c
Explanation: The database-system implementations use B+ -tree data structure because they can be used for multilevel indexing.
Topic 70. 2-3 Tree

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “2-3 Tree”.

1. 2-3 tree is a specific form of _________
a) B – tree
b) B+ – tree
c) AVL tree
d) Heap
View Answer

Answer: a
Explanation: The 2-3 trees is a balanced tree. It is a specific form the B – tree. It is B – tree of order 3, where every node can have two child subtrees and one key or 3 child subtrees and two keys.

2. Which of the following is the 2-3 tree?
a)
b)
c)
d)
View Answer

Answer: c
Explanation: Tree should have two subtrees at node2, but it should not have three elements. The node with elements 11 and 15 should have three child subtrees.

3. The height of 2-3 tree with n elements is ______
a) between (n/2) and (n/3)
b) (n/6)
c) between (n) and log2(n + 1)
d) between log3(n + 1) and log2(n + 1)
View Answer

Answer: d
Explanation: The number of elements in a 2-3 tree with height h is between 2h – 1 and 3h – 1. Therefore, the 2-3 tree with n elements will have the height between log3(n + 1) and log2(n + 1).

4. Which of the following the BST is isometric with the 2-3 tree?
a) Splay tree
b) AA tree
c) Heap
d) Red – Black tree
View Answer

Answer: b
Explanation: AA tree is isometric of the 2-3 trees. In an AA tree, we store each node a level, which is the height of the corresponding 2-3 tree node. So, we can convert a 2-3 tree to an AA tree.

5. The figure shown below is a 2-3 tree. What is the result of deleting 110 from the tree?

a)
b)
c)
d)
View Answer

Answer: c
Explanation: When 110 is deleted the respective node becomes empty, so the 2-3 tree properties get violated. Hence, the element from its sibling node, 93 is moved to the root and root node element 100 is fed to the empty node. So, the resultant 2-3 tree will be,

6. Which of the following data structure can provide efficient searching of the elements?
a) unordered lists
b) binary search tree
c) treap
d) 2-3 tree
View Answer

Answer: d
Explanation: The average case time for lookup in a binary search tree, treap and 2-3 tree is O(log n) and in unordered lists it is O(n). But in the worst case, only the 2-3 trees perform lookup efficiently as it takes O(log n), while others take O(n).

7. LLRB maintains 1-1 correspondence with 2–3 trees.
a) True

b) False
View Answer

Answer: a
Explanation: LLRB (Left Leaning Red Black tree)is the data structure which is used to implement the 2-3 tree with very basic code. The LLRB is like the 2-3 tree where each node has one key and two links. In LLRB the 3-node is implemented as two 2-nodes connected by the red link that leans left. Thus, LLRB maintains 1-1 correspondence with 2–3 tree.

8. Which of the following is not true about the 2-3 tree?
a) all leaves are at the same level
b) it is perfectly balanced
c) postorder traversal yields elements in sorted order
d) it is B-tree of order 3
View Answer

Answer: c
Explanation: In a 2-3 tree, leaves are at the same level. And 2-3 trees are perfectly balanced as every path from root node to the null link is of equal length. In 2-3 tree in-order traversal yields elements in sorted order.

9. AVL trees provide better insertion the 2-3 trees.
a) True
b) False
View Answer

Answer: b
Explanation: Insertion in AVL tree and 2-3 tree requires searching for proper position for insertion and transformations for balancing the tree. In both, the trees searching takes O(log n) time, but rebalancing in AVL tree takes O(log n), while the 2-3 tree takes O(1). So, 2-3 tree provides better insertions.

10. Which of the following is false?
a) 2-3 tree requires less storage than the BST
b) lookup in 2-3 tree is more efficient than in BST
c) 2-3 tree is shallower than BST
d) 2-3 tree is a balanced tree
View Answer

Answer: a
Explanation: Search is more efficient in the 2-3 tree than in BST. 2-3 tree is a balanced tree and performs efficient insertion and deletion and it is shallower than BST. But, 2-3 tree requires more storage than the BST.
Topic 71. Ternary Tree – 1

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Ternary Tree – 1”.

1. How many child nodes does each node of Ternary Tree contain?
a) 4
b) 6
c) 5
d) 3
View Answer

Answer: d
Explanation: Each node of Ternary tree contains at most 3 nodes. So Ternary tree can have 1, 2 or 3 child nodes but not more than that.

2. Which of the following is the name of the node having child nodes?
a) Brother
b) Sister
c) Mother
d) Parent
View Answer

Answer: d
Explanation: Parent node is the node having child nodes and child nodes may contain references to their parents. Parent node is a node connected by a directed edge to its child.

3. What is the depth of the root node of the ternary tree?
a) 2
b) 1
c) 0
d) 3

View Answer

Answer: c
Explanation: Depth is defined as the length of the path from root to the node. So the depth of root node in ternary tree is 0.

4. What is the Height of the root node of ternary tree?
a) 1
b) 2
c) 3
d) 0
View Answer

Answer: d
Explanation: Height of ternary tree is defined as the length of path from root to deepest node in tree. Therefore, height off root node in ternary tree is 0.

5. Which node is the root node of the following ternary tree?

a) A
b) B
c) C
d) D
View Answer

Answer: a
Explanation: Node A is called the root node of the above ternary tree while the Node B, Node C, Node D are called Leaf node.

6. Which node is the Leaf node in the following ternary tree?

a) A
b) B
c) D
d) G
View Answer

Answer: d
Explanation: Leaf node is any node that does not contain any children. Since Node G is the node without any children, So G is called Leaf Node. While Node A is root node and Node B, Node C, Node D is parent node of their children.

7. Which node is the parent node of Node 6?

a) 1
b) 5
c) 2
d) 3
View Answer

Answer: c
Explanation: Since Node 2 has two children Node 5 and Node 6, So Node 2 is the parent node of Node 6. While Node 1 is root node and Node 3 and Node 5 are Leaf node.

8. Is parent node of Node 3 and root node of the given ternary tree same?

a) True
b) False
View Answer

Answer: a
Explanation: Since Root node of the ternary tree is Node 1 and also Node 1 has three children that is Node 2, Node 3, Node 4. So parent node of Node 3 and the root node of the ternary tree are same.

9. Which node is the child node of Node D in the following ternary tree?

a) A
b) C
c) G
d) H
View Answer

Answer: d
Explanation: The Child node is the node that has a directed path from its parent node. Since Node D has a direct path to Node H, So Node H is the Child node.

10. Which node is the child node of the Node D in the following ternary tree?

a) A
b) C
c) B
d) No child node
View Answer

Answer: d
Explanation: Since Node D is the Leaf node of the above ternary tree and leaf node has no child node. So there is no child node for Node D in the above ternary tree.

11. What is the depth of Node G in the given ternary tree?

a) 0
b) 1
c) 2
d) 3
View Answer

Answer: c
Explanation: Depth of the node is the length of the path from root to the node. Here, length of path from root to Node G is 2. So depth of Node G is 2.

12. What is the Height of the given ternary tree?

a) 0
b) 1
c) 2
d) 3
View Answer

Answer: c
Explanation: Height of the tree is defined as the length of the path from root node to the deepest node of the tree. Here deepest nodes are 5,6,7 which are at length 2. So the height of the ternary tree is 2.

13. Which nodes are the siblings of Node B of given ternary tree?

a) E
b) C
c) F
d) Both E and F
View Answer

Answer: d
Explanation: Siblings are the nodes that share same parent. Since both the Node E and Node F have same parent Node B, So the sibling of Node B is Node E and Node F.
Topic 72. Ternary Tree – 2

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Ternary Tree – 2”.

1. How many extra nodes are there in Full ternary tree than a complete ternary tree?
a) 1
b) 2
c) 3
d) Both have same number of nodes
View Answer

Answer: d
Explanation: Every Full ternary tree is also a complete ternary tree. Therefore, both have same number of nodes.

2. Is Node A sibling of Node D in the given ternary tree?

a) True
b) False
View Answer

Answer: b
Explanation: Node B, Node C are the siblings of Node D while Node A is the parent node of Node D.

3. What is the size of the given ternary tree?

a) 3
b) 2
c) 6
d) 4
View Answer

Answer: c
Explanation: Size of the ternary tree is defined as the total number of nodes present in the tree. Since there are total of 6 nodes in the ternary tree. So the size of the ternary tree is 6.

4. Who is the ancestor of Node G?

a) C
b) F
c) H
d) A
View Answer

Answer: a
Explanation: Ancestor node is a node that comes in between the path from the node to the root. Since Node C comes between node G and root, so Node C is called the ancestor node.

5. Who is descendant of Node B?

a) A
b) E
c) F
d) Both E and F
View Answer

Answer: d
Explanation: Descendant node is a node which has a direct path from ancestor node. Since both E and F are connected to B, so Node E and Node F are the descendants of Node B.

6. What is the size of Node A?

a) 1
b) 2
c) 3
d) 4
View Answer

Answer: d
Explanation: Size of node is defined as the total number of descendants of that node including itself. So, size of Node A is 4.

7. Can leaf node be called child node in a ternary tree?
a) True
b) False
View Answer

Answer: a
Explanation: Leaf node is a node that has no child. Since Leaf node will always be the node on the last level of ternary tree, so it can be called child node of given parent node in ternary tree.

8. Can child node be always called Leaf node in the ternary tree?
a) True
b) False
View Answer

Answer: b
Explanation: Leaf node is any node that does not contain any children. Child node may or may not contain more nodes. Child node will only be called leaf Node if the node has no child node.

9. Which of the following is the implementation of the ternary tree?
a) AVL Tree

b) Ternary Heap
c) Hash Table
d) Dictionary
View Answer

Answer: b
Explanation: Ternary tree is used to implement ternary search tree and ternary heap. While AVL Tree, hash Table, dictionary are different types of Data Structures.
Topic 73. K-ary Tree – 1

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “K-ary Tree – 1”.

1. How many child nodes does each node of K-ary Tree contain?
a) 2
b) 3
c) more than k
d) at most k
View Answer

Answer: d
Explanation: Each node of K-ary tree contains at most k nodes. While tree with 2 nodes is called Binary tree and tree with 3 nodes is called Ternary tree.

2. Which of the following is the name of the node having child nodes?
a) Brother
b) Sister
c) Mother
d) Parent
View Answer

Answer: d
Explanation: Parent node is the node having child nodes and child nodes may contain references to their parents. Parent node is a node connected by a directed edge to its child.

3. What is the depth of the root node of K-ary tree?
a) 2
b) 1
c) 0
d) 3
View Answer

Answer: c
Explanation: Depth is defined as the length of the path from root to the node. So the depth of root node in K-ary tree is 0.

4. What is the Height of the root node of K-ary tree?
a) 1
b) 2
c) 3
d) 0
View Answer

Answer: d
Explanation: Height of K-ary tree is defined as the length of path from root to deepest node in tree. Therefore, height of root node in K-ary tree is 0.

5. Which node is the root node of the following K-ary tree?

a) A
b) B
c) C
d) D
View Answer

Answer: a
Explanation: Node A is called the root node of the above K-ary tree while the Node B, Node C, Node D are called Leaf node.

6. Which node is the Leaf node in the following K-ary tree?

a) A

b) B
c) D
d) F
View Answer

Answer: d
Explanation: Leaf node is any node that does not contain any children. Since Node F is the node without any children, So F is called Leaf Node. While Node A is root node and Node B, Node C, Node D is parent node of their children.

7. Which node is the parent node of Node 5?

a) 1
b) 5
c) 2
d) 3
View Answer

Answer: c
Explanation: Since Node 2 has two children Node 5 and Node 6, So Node 2 is the parent node of Node 5. While Node 1 is root node and Node 3 and Node 5 are Leaf node.

8. Is parent node of Node 4 and root node of the given K-ary tree same?

a) True
b) False
View Answer

Answer: a
Explanation: Since Root node of the ternary tree is Node 1 and also Node 1 has three children that is Node 2, Node 3, Node 4. So parent node of Node 4 and the root node of the ternary tree are same.

9. Which node is the child node of Node C in the following K-ary tree?

a) A
b) C
c) G
d) H
View Answer

Answer: c
Explanation: The Child node is the node that has a directed path from its parent node. Since Node C has a direct path to Node G, So Node G is the Child node.

10. Which node is the child node of the Node B in the following K-ary tree?

a) A
b) C
c) B
d) No child node
View Answer

Answer: d
Explanation: Since Node B is the Leaf node of the above ternary tree and leaf node has no child node. So there is no child node for Node B in the above K-ary tree.

11. What is the depth of Node F in the given K-ary tree?

a) 0
b) 1
c) 2
d) 3
View Answer

Answer: c
Explanation: Depth of the node is the length of the path from root to the node. Here, length of path from root to Node F is 2. So depth of Node F is 2.

12. What is the Height of the given K-ary tree?

a) 0

b) 1
c) 2
d) 3
View Answer

Answer: c
Explanation: Height of the tree is defined as the length of the path from root node to the deepest node of the tree. Here deepest nodes are 5,6,7 which are at length 2. So the height of the K-ary tree is 2.
Topic 74. K-ary Tree – 2

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “K-ary Tree – 2”.

1. What is the size of the given K-ary tree?

a) 3
b) 2
c) 6
d) 4
View Answer

Answer: c
Explanation: Size of the K-ary tree is defined as the total number of nodes present in the tree. Since there are total of 6 nodes in the K-ary tree. So the size of the K-ary tree is 6.

2. Who is the ancestor of Node H?

a) D
b) F
c) H
d) A
View Answer

Answer: a
Explanation: Ancestor node is a node that comes in between the path from the node to the root. Since Node D comes between node H and root, so Node D is called the ancestor node.

3. Who is descendant of Node C?

a) A
b) E
c) D
d) G
View Answer

Answer: d
Explanation: Descendant node is a node which has a direct path from ancestor node. Since Node G is connected to C, so Node G is the descendant of Node C.

4. What is the size of Node B in given K-ary tree?

a) 1
b) 2
c) 3
d) 4
View Answer

Answer: a
Explanation: Size of node is defined as the total number of descendants of that node including itself. So, size of Node B is 1.

5. Can leaf node be called child node in a K-ary tree?
a) True
b) false
View Answer

Answer: a
Explanation: Leaf node is a node that has no child. Since Leaf node will always be the node on the last level of k-ary tree, so it can be called child node of given parent node in K-ary tree.

6. Can child node be always called Leaf node in the K-ary tree?
a) True

b) False
View Answer

Answer: b
Explanation: Leaf node is any node that does not contain any children. Child node may or may not contain more nodes. Child node will only be called leaf Node if the node has no child node.

7. What is the upper bound for maximum leaves in K-ary tree with height h?
a) K*h
b) K^h
c) K+h
d) K-h
View Answer

Answer: b
Explanation: In the K-ary tree having height h, the upper bound for having maximum number of leaves is k^h.

8. What is the height of a K-ary tree having only root node?
a) 1
b) 0
c) 2
d) 3
View Answer

Answer: b
Explanation: Height of a K-ary tree does not include the root node. So the height of the K-ary tree is without root node is 0.

9. Which one of the following is the correct formulae to find the parent node at index I?
a) (I-1)/K
b) (I+1)/K
c) (I*1)/K
d) (I-2)/K
View Answer

Answer: a
Explanation: The parent node for the node of index I in a K-ary tree is given by (I-1)/K.

10. Which nodes are the siblings of Node D of given ternary tree?

a) E
b) C
c) F
d) H
View Answer

Answer: d
Explanation: Siblings are the nodes that share same parent. Since both the Node H is parent Node D, So the sibling of Node D is Node H.

11. How many extra nodes are there in Full K-ary tree than complete K-ary tree?
a) 1
b) 2
c) 3
d) Both have same number of nodes
View Answer

Answer: d
Explanation: Every Full K-ary tree is also a complete K-ary tree. Therefore, both have same number of nodes.

12. Is Node A sibling of Node B in the given K-ary tree?

a) True
b) False
View Answer

Answer: b
Explanation: Node D, Node C are the siblings of Node B while Node A is the parent node of Node B.
Topic 75. Van Emde Boas Tree

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Van Emde Boas Tree”.

1. What is the other name or Van Emde Boas Tree data structure?
a) Van Emde Boas Array
b) Van Emde Boas Stack
c) Van Emde Boas Priority Queue
d) Van Emde Boas Heap
View Answer

Answer: c
Explanation: The Van Emde Boas Tree data structure is also popularly known as Van Emde Boas Priority Queue. This data structure implements the array associatively for the given integer keys. It was formulated by Peter Van Emde Boas.

2. Who Invented The vEB also known as Van Emde Boas Tree?
a) Peter Van Emde Boas
b) Samuel F. B. Morse
c) Friedrich Clemens Gerke
d) Alexander Morse
View Answer

Answer: a
Explanation: The Van Emde Boas Tree data structure is also popularly known as Van Emde Boas Priority Queue. This data structure implements the array associatively for the given integer keys. It was formulated by Peter Van Emde Boas.

3. What is the time complexity for storing the maximum number of elements in Van Emde Boas tree if M is the maximum number of elements?
a) O (log M)
b) O (M!)
c) O (M)
d) O (1)
View Answer

Answer: c
Explanation: In order to store the maximum number of elements in Van Emde Boas data structure where M is the maximum number of elements, the tree has great efficiency for storing them. So the time complexity for worst case is found to be O (M).

4. Does Van Emde Boas data structure perform all operation in O (log (log M)) time where M = 2^m.
a) True
b) False
View Answer

Answer: a
Explanation: All the operations performed on the Van Emde Boas tree with an associative array like Insertion, Deletion, Searching and many more can be performed in O (log (log M)) time where M = 2^m.

5. What is the time complexity for searching a key or integer in Van Emde Boas data structure?
a) O (log M!)
b) O (M!)
c) O (M^2)
d) O (log (log M))
View Answer

Answer: d
Explanation: In order to search a key or integer in the Van Emde Boas data structure, the operation can be performed on an associative array. Hence, the time complexity for searching a key or integer in Van Emde Boas data structure is O (log (log M)).

6. Which type of tree does Van Emde Boas require to perform basic operations?
a) Unbalanced
b) Balanced
c) Complete
d) Non – Binary
View Answer

Answer: d
Explanation: The Van Emde Boas Tree data structure is also popularly known as Van Emde Boas Priority Queue. This data structure implements the array associatively for the given integer keys. It was formulated by Peter Van Emde Boas. It is a non – binary type of tree.

7. What is the time complexity for inserting a key or integer in Van Emde Boas data structure?
a) O (log M!)

b) O (M!)
c) O (M^2)
d) O (log (log M))
View Answer

Answer: d
Explanation: In order to insert a key or integer in the Van Emde Boas data structure, the operation can be performed on an associative array. Hence, the time complexity for inserting a key or integer in Van Emde Boas data structure is O (log (log M)).

8. In which year was Van Emde Boas tree invented?
a) 1972
b) 1973
c) 1974
d) 1975
View Answer

Answer: d
Explanation: The Van Emde Boas Tree data structure is also popularly known as Van Emde Boas Priority Queue. This data structure implements the array associatively for the given integer keys. It was formulated by Peter Van Emde Boas in 1975.

9. What is the time complexity for deleting a key or integer in Van Emde Boas data structure?
a) O (log M!)
b) O (log (log M))
c) O (M!)
d) O (M^2)
View Answer

Answer: b
Explanation: In order to delete a key or integer in the Van Emde Boas data structure, the operation can be performed on an associative array. Hence, the time complexity for deleting a key or integer in Van Emde Boas data structure is O (log (log M)).

10. Can operation like Find Next and Find Previous be implemented.
a) True
b) False
View Answer

Answer: a
Explanation: Since the Van Emde Boas data structure follows an associative array abstract data type to perform different operations. Hence, an operation like Find Next and Find Previous be implemented.

11. What is the time complexity for finding a maximum and minimum integer in Van Emde Boas data structure?
a) O (log M!)
b) O (M!)
c) O (1)
d) O (log (log M))
View Answer

Answer: c
Explanation: In order to find a maximum or minimum integer in the Van Emde Boas data structure, the operation can be performed on an associative array. Hence, the time complexity for finding a maximum or minimum integer in Van Emde Boas data structure is O (1).

12. On which abstract data type does van Emde Boas tree performs the operation?
a) Tree
b) Linked List
c) Heap
d) Associative Array
View Answer

Answer: d
Explanation: The Van Emde Boas Tree data structure is also popularly known as Van Emde Boas Priority Queue. This data structure implements an abstract data type called associative array for the given integer keys.

13. Which operation find the value associated with a given key?
a) Insert
b) Find Next
c) Look up
d) Delete

View Answer

Answer: c
Explanation: This data structure implements an abstract data type called associative array for the given integer keys. Hence, to find the value associated with a given key, Look Up operation is performed.
Topic 76. Disjoint-Set Data Structure

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Disjoint-Set Data Structure”.

1. How many properties will an equivalent relationship satisfy?
a) 1
b) 2
c) 3
d) 4
View Answer

Answer: c
Explanation: An equivalent relationship will satisfy three properties – reflexive, symmetric and transitive.

2. A relation R on a set S, defined as x R y if and only if y R x. This is an example of?
a) reflexive relation
b) symmetric relation
c) transitive relation
d) invalid relation
View Answer

Answer: b
Explanation: A symmetric property in an equivalence relation is defined as x R y if and only y R x.

3. Electrical connectivity is an example of equivalence relation.
a) true
b) false
View Answer

Answer: a
Explanation: Electrical connectivity is reflexive, symmetric and also transitive. Hence, electrical connectivity is an equivalence relation.

4. What is the worst case efficiency for a path compression algorithm?
a) O(N)
b) O(log N)
c) O(N log N)
d) O(M log N)
View Answer

Answer: d
Explanation: The worst case efficiency for a path compression algorithm is mathematically found to be O(M log N).

5. Path Compression algorithm performs in which of the following operations?
a) Create operation
b) Insert operation
c) Find operation
d) Delete operation
View Answer

Answer: c
Explanation: Path compression algorithm is performed during find operation and is independent of the strategy used to perform unions.

6. What is the definition for Ackermann’s function?
a) A(1,i) = i+1 for i>=1
b) A(i,j) = i+j for i>=j
c) A(i,j) = i+j for i = j
d) A(1,i) = i+1 for i<1
View Answer

Answer: a
Explanation: The Ackermann’s function is defined as A(1,i) = i+1 for i>=1. This form in text grows faster and the inverse is slower.

7. ___________ is one of the earliest forms of a self-adjustment strategy used in splay trees, skew heaps.
a) Union by rank
b) Equivalence function
c) Dynamic function
d) Path compression
View Answer

Answer: d
Explanation: Path compression is one of the earliest forms of self-adjustment used in extremely important strategies using theoretical explanations.

8. What is the depth of any tree if the union operation is performed by height?
a) O(N)
b) O(log N)
c) O(N log N)
d) O(M log N)
View Answer

Answer: b
Explanation: If the Unions are performed by height, the depth of any tree is calculated to be O(log N).

9. When executing a sequence of Unions, a node of rank r must have at least 2^r descendants.
a) true
b) false
View Answer

Answer: a
Explanation: By the induction hypothesis, each tree has at least 2^{r-1} descendants, giving a total of 2^r and establishing the lemma.

10. What is the value for the number of nodes of rank r?
a) N
b) N/2
c) $N/2^r$
d) N^r
View Answer

Answer: c
Explanation: Each node of a rank r is the root of a subtree of at least 2^r. Therefore, there are at most $N/2^r$ disjoint subtrees.

11. What is the worst-case running time of unions done by size and path compression?
a) O(N)
b) O(logN)
c) O(N logN)
d) O(M logN)
View Answer

Answer: d
Explanation: The worst case running time of a union operation done by size and path compression is mathematically found to be O(M logN).

12. In the Union/Find algorithm, the ranks of the nodes on a path will increase monotonically from?
a) leaf to root
b) root to node
c) root to leaf
d) left subtree to right subtree
View Answer

Answer: a
Explanation: One of the lemmas state that, in the Union/Find algorithm, the ranks of the nodes on a path will increase monotonically from leaf to root.

13. How many strategies are followed to solve a dynamic equivalence problem?
a) 1
b) 2
c) 3
d) 4
View Answer

Answer: b

Explanation: There are two strategies involved to solve a dynamic equivalence problem- executing find instruction in worst-case time and executing union instruction in worst-case time.

14. What is the total time spent for N-1 merges in a dynamic equivalence problem?
a) O(N)
b) O(log N)
c) O(N log N)
d) O(M log N)
View Answer

Answer: c
Explanation: The total time spent for N-1 merges in a dynamic equivalence problem is mathematically found to be O(N log N).

15. What is the condition for an equivalence relation if two cities are related within a country?
a) the two cities should have a one-way connection
b) the two cities should have a two-way connection
c) the two cities should be in different countries
d) no equivalence relation will exist between two cities
View Answer

Answer: b
Explanation: If the two towns are in the same country and have a two-way road connection between them, it satisfies equivalence property.
Topic 77. Bin

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Bin”.

1. What is the use of the bin data structure?
a) to have efficient insertion
b) to have efficient deletion
c) to have efficient region query
d) to have efficient traversal
View Answer

Answer: c
Explanation: Bin data structure allows us to have efficient region queries. A frequency of bin is increased by one each time a data point falls into a bin.

2. Bin is an example of a range query data structure.
a) true
b) false
View Answer

Answer: a
Explanation: Bin is an example of a range query data structure. It is because it efficiently answers any number of queries on any subset of the input.

3. What is the worst case time complexity of query operation(n is the no. of candidates)?
a) O(1)
b) O(n)
c) O(log n)
d) O(n log n)
View Answer

Answer: b
Explanation: The worst case in a bin query occurs when all the candidates are concentrated in one bin. So in this case the time complexity is O(n).

4. What is the worst case time complexity of delete operation(n is the no. of candidates)?
a) O(1)
b) O(n)
c) O(log n)
d) O(n log n)
View Answer

Answer: b
Explanation: The worst case in a bin delete operation occurs when all the candidates are concentrated in one bin. So in this case the time complexity is O(n).

5. What is the worst case time complexity of insertion operation(n =no. of candidates)?

a) O(1)
b) O(n)
c) O(log n)
d) O(n log n)
View Answer

Answer: a
Explanation: The worst case in a bin insert operation occurs when all the candidates are concentrated in one bin. So in this case the time complexity is O(1).

6. What is computational geometry?
a) study of geometry using a computer
b) study of geometry
c) study of algorithms
d) study of algorithms related to geometry
View Answer

Answer: d
Explanation: Computational geometry deals with the study of algorithms which can be expressed in terms of geometry. Bin data structure is an example of it.

7. What will be the time complexity of query operation if all the candidates are evenly spaced so that each bin has constant no. of candidates? (k = number of bins query rectangle intersects)
a) O(1)
b) O(k)
c) O(k^2)
d) O(log k)
View Answer

Answer: b
Explanation: The process of query becomes faster in a case when the number of candidates are equally distributed among the bins. In such a case the query operation becomes O(k).

8. What will be the time complexity of delete operation if all the candidates are evenly spaced so that each bin has constant no. of candidates? (m = number of bins intersecting candidate intersects)
a) O(1)
b) O(m)
c) O(m^2)
d) O(log m)
View Answer

Answer: b
Explanation: The process of deletion becomes faster in a case when the number of candidates are equally distributed among the bins. In such a case the query operation becomes O(m). It is practically slower than insertion in this case.

9. What will be the time complexity of insertion operation if all the candidates are evenly spaced so that each bin has constant no. of candidates? (m = number of bins intersecting candidate intersects)
a) O(1)
b) O(m)
c) O(m^2)
d) O(log m)
View Answer

Answer: b
Explanation: The process of insertion becomes faster in the case when the number of candidates are equally distributed among the bins. In such a case the query operation becomes O(m). It is practically faster than deletion in this case.

10. Efficiency of bin depends upon ___________
a) size of query and candidates
b) location of query and candidates
c) location and size of query and candidates
d) depends on the input
View Answer

Answer: c
Explanation: Efficiency of bin depends upon the location and size of query and candidates. It is similar to that of a hash table.

11. Bigger the query rectangle the better is the query efficiency.
a) true

b) false
View Answer

Answer: b
Explanation: Efficiency of bin depends upon the location and size of query and candidates. Also, the smaller is the query rectangle the better is the query efficiency.
Topic 78. KD Tree

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “KD Tree”.

1. In what time can a 2-d tree be constructed?
a) O(N)
b) O(N log N)
c) O(N^2)
d) O(M log N)
View Answer

Answer: b
Explanation: A perfectly balanced 2-d tree can be constructed in O(N log N) time. This value is computed mathematically.

2. Insertion into a 2-d tree is a trivial extension of insertion into a binary search tree.
a) true
b) false
View Answer

Answer: a
Explanation: Insertion of elements in a 2-d tree is similar to that of a binary search tree. Hence, it is a trivial extension of the binary search tree.

3. In a two-dimensional search tree, the root is arbitrarily chosen to be?
a) even
b) odd
c) depends on subtrees
d) 1
View Answer

Answer: b
Explanation: In a two- dimensional k-d tree (i.e.) 2-d tree, the root is arbitrarily chosen to be an odd level and it applies to all 2-d trees.

4. Which of the following is the simplest data structure that supports range searching?
a) Heaps
b) binary search trees
c) AA-trees
d) K-d trees
View Answer

Answer: d
Explanation: K-d trees are the simplest data structure that supports range searching and also it achieves the respectable running time.

5. In a k-d tree, k originally meant?
a) number of dimensions
b) size of tree
c) length of node
d) weight of node
View Answer

Answer: a
Explanation: Initially, 2-d trees were created. Then, 3-d trees, 4-trees etc., where k meant the number of dimensions.

6. What will be the correct sequence of insertion for the following k-d tree?

a) (30,40),(5,25),(70,70),(10,12),(50,30),(35,45)
b) (40,30),(5,25),(12,10),(70,70),(30,50),(45,35)
c) (30,40),(5,25),(10,12),(70,70),(50,30),(35,45)
d) (40,30),(25,5),(12,10),(70,70),(50,30),(45,35)
View Answer

Answer: c

Explanation: The correct sequence of insertion of the above given tree is (30,40),(5,25),(10,12),(70,70),(50,30),(35,45). The insertion is given by, first left, then right.

7. Each level in a k-d tree is made of?
a) dimension only
b) cutting and dimension
c) color code of node
d) size of the level
View Answer

Answer: b
Explanation: Each level in a k-d tree is made of dimensions and cutting. Cutting and dimensions are used for insertion, deletion and searching purposes.

8. What is the worst case of finding the nearest neighbour?
a) O(N)
b) O(N log N)
c) O(log N)
d) O(N^3)
View Answer

Answer: a
Explanation: The worst case analysis of finding the nearest neighbour in a k-d tree is mathematically found to be O(N).

9. What is the run time of finding the nearest neighbour in a k-d tree?
a) O(2+ log N)
b) O(log N)
c) O(2^d log N)
d) O(N log N)
View Answer

Answer: c
Explanation: The run time of finding the nearest neighbour in a kd tree is given as O(2^d log N) where 2^d is the time taken to search the neighbourhood.

10. How many prime concepts are available in nearest neighbour search in a kd tree?
a) 1
b) 2
c) 3
d) 4
View Answer

Answer: c
Explanation: Three important concepts are available in finding the nearest neighbour. They are partial results, pruning, traversal order.

11. Reducing search space by eliminating irrelevant trees is known as?
a) pruning
b) partial results
c) freeing space
d) traversing
View Answer

Answer: a
Explanation: Pruning is eliminating irrelevant trees. Partial results are keeping best results and updating. Traversal is visiting all the nodes of a tree.

12. Several kinds of queries are possible on a k-d called as?
a) partial queries
b) range queries
c) neighbour queries
d) search queries
View Answer

Answer: b
Explanation: Several range queries are possible on a k-d tree. One of the range queries is known as a partial match query.

13. What is the time taken for a range query for a perfectly balanced tree?
a) O(N)
b) O(log N)

c) O(√N+M)
d) O(√N)
View Answer

Answer: c
Explanation: For a perfectly balanced k-d tree, the range query could take O(√N+M) in the worst case to report M matches.

14. The 2d search tree has the simple property that branching on odd levels is done with respect to the first key.
a) True
b) False
View Answer

Answer: a
Explanation: Branching on odd levels is done with respect to the first key and branching on even levels is done with respect to the second key in a 2-d tree.

15. Who invented k-d trees?
a) Arne Andersson
b) Jon Bentley
c) Jon Von Newmann
d) Rudolf Bayer
View Answer

Answer: b
Explanation: Jon Bentley found k-d trees. Rudolf Bayer found red black trees. Arne Andersson found AA- trees.
Topic 79. Expression Tree

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on "Expression Tree".

1. The leaves of an expression tree always contain?
a) operators
b) operands
c) null
d) expression
View Answer

Answer: b
Explanation: The leaves of an expression tree always contain the result of a given expression (i.e.) operands.

2. A node can have a minimum of one child.
a) true
b) false
View Answer

Answer: a
Explanation: It is possible for a node to have at least one child, as is the case with the unary minus operator.

3. What does the other nodes of an expression tree(except leaves) contain?
a) only operands
b) only operators
c) both operands and operators
d) expression
View Answer

Answer: b
Explanation: The nodes other than leaves always contain only operators. There cannot be any operand in those nodes.

4. An expression tree is a kind of?
a) Binary search tree
b) Fibonacci tree
c) Binary tree
d) Treap
View Answer

Answer: c
Explanation: The expression tree is a binary tree. It contains operands at leaf nodes and remaining nodes are filled with operators. The operands and the operators can be arranged in any order (ascending, descending).

5. The expression obtained by recursively producing a left expression, followed by an operator, followed by recursively producing a right expression is called?
a) prefix expression

b) infix expression
c) postfix expression
d) paranthesized expression
View Answer

Answer: b
Explanation: It is an infix expression because the format of an infix expression is given by operand-operator-operand.

6. The average depth of a binary tree is given as?
a) O(N)
b) O(log N)
c) O(M log N)
d) O(√N)
View Answer

Answer: d
Explanation: The average depth of a binary expression tree is mathematically found to be O(√N).

7. Only infix expression can be made into an expression tree.
a) true
b) false
View Answer

Answer: b
Explanation: All infix, prefix and postfix expressions can be made into an expression tree using appropriate algorithms.

8. An expression tree is created using?
a) postfix expression
b) prefix expression
c) infix expression
d) paranthesized expression
View Answer

Answer: a
Explanation: A postfix expression is converted into an expression tree by reading one symbol at a time and constructing a tree respectively.

9. ++a*bc*+defg is an?
a) postfix expression
b) infix expression
c) prefix expression
d) invalid expression
View Answer

Answer: c
Explanation: It is a prefix expression obtained from a preorder traversal since it is of the form operator-operand-operand.

10. An expression tree's nodes can be deleted by calling?
a) malloc
b) calloc
c) delete
d) free
View Answer

Answer: d
Explanation: In Binary trees, nodes are created by calling malloc and they are deleted by calling free.

11. What is the postfix expression for the following expression tree?

a) abcde++**
b) ab+cde+**
c) abc+de+**
d) abcd+*e+*
View Answer

Answer: b
Explanation: If the given expression tree is evaluated, the postfix expression ab+cde+** is obtained.

12. In an expression tree algorithm, what happens when an operand is encountered?
a) create one node pointing to a stack
b) pop the nodes from the stack

c) clear stack
d) merge all the nodes
View Answer

Answer: a
Explanation: When an operand is encountered, create one node trees and push it on to the stack. When an operator is encountered, pop the pointers from last two trees from the stack.
Topic 80. Heap

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Heap”.

1. In a max-heap, element with the greatest key is always in the which node?
a) Leaf node
b) First node of left sub tree
c) root node
d) First node of right sub tree
View Answer

Answer: c
Explanation: This is one of the property of max-heap that root node must have key greater than its children.

2. Heap exhibits the property of a binary tree?
a) True
b) False
View Answer

Answer: a
Explanation: Yes, because the leaf nodes are present at height h or h-1, which is a property of complete binary tree.

3. What is the complexity of adding an element to the heap.
a) O(log n)
b) O(h)
c) O(log n) & O(h)
d) O(n)
View Answer

Answer: c
Explanation: The total possible operation in re locating the new location to a new element will be equal to height of the heap.

4. The worst case complexity of deleting any arbitrary node value element from heap is __________
a) O(logn)
b) O(n)
c) O(nlogn)
d) O(n^2)
View Answer

Answer: a
Explanation: The total possible operation in deleting the existing node and re locating new position to all its connected nodes will be equal to height of the heap.

5. Heap can be used as ________________
a) Priority queue
b) Stack
c) A decreasing order array
d) Normal Array
View Answer

Answer: a
Explanation: The property of heap that the value of root must be either greater or less than both of its children makes it work like a priority queue.

6. If we implement heap as min-heap, deleting root node (value 1)from the heap. What would be the value of root node after second iteration if leaf node (value 100) is chosen to replace the root at start.

a) 2
b) 100
c) 17
d) 3
View Answer

Answer: a
Explanation: As the root is deleted and node with value 100 is used as replaced one. There is a violation of property that root node must have value less than its children. So there will be shuffling of node with value 100 with node of value 2. And thus 2 becomes root. And it will remain to be at root after all the possible operations. So the value at root will be 2.

7. If we implement heap as maximum heap , adding a new node of value 15 to the left most node of right subtree. What value will be at leaf nodes of the right subtree of the heap.

a) 15 and 1
b) 25 and 1
c) 3 and 1
d) 2 and 3
View Answer

Answer: a
Explanation: As 15 is less than 25 so there would be no violation and the node will remain at that position. So leaf nodes with value 15 and 1 will be at the position in right sub tree.

8. An array consists of n elements. We want to create a heap using the elements. The time complexity of building a heap will be in order of
a) O(n*n*logn)
b) O(n*logn)
c) O(n*n)
d) O(n *logn *logn)
View Answer

Answer: b
Explanation: The total time taken will be N times the complexity of adding a single element to the heap. And adding a single element takes logN time, so That is equal to N*logN.
Topic 81. Binary Heap

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “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)
View Answer

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(n^2)
c) O(n*longn *logn)
d) O(n)
View Answer

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

3. Given the code, choose the correct option that is consistent with the code. (Here A is the heap)

```
build(A,i)
left-> 2*i
right->2*i +1
temp- > i
if(left<= heap_length[A] ans A[left] >A[temp])
temp -> left
if (right = heap_length[A] and A[right] > A[temp])
temp->right
if temp!= i
swap(A[i],A[temp])
build(A,temp)
```

a) It is the build function of max heap
b) It is the build function of min heap
c) It is general build function of any heap
d) It is used to search element in any heap

View Answer

Answer: a
Explanation: Since in every condition we are comparing the current value is less than the parent of that node. So this is build function of Max heap.

4. What is the location of a parent node for any arbitary node i?
a) (i/2) position
b) (i+1)/ position
c) floor(i/2) position
d) ceil(i/2) position
View Answer

Answer: c
Explanation: For any node child nodes are located at either 2*i, 2*i +1 So the parent node could be found by taking the floor of the half of child node.

5. State the complexity of algorithm given below.

```
int function(vector<int> arr)
int len=arr.length();
if(len==0)
return;
temp=arr[len-1];
arr.pop_back();
return temp;
```

a) o(n)
b) O(logn)
c) O(1)
d) O(n logn)
View Answer

Answer: c
Explanation: Deletion in a min-heap is in O(1) time.

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
View Answer

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).

7. For construction of a binary heap with property that parent node has value less than child node. In reference to that which line is incorrect. Line indexed from 1.

```
1. add(int k)
2. {
3.     heap_size++;
4.     int i = heap_size - 1;
5.     harr[i] = k;
6.     while (i != 0 && harr[parent(i)] < harr[i])
7.     {
8.             swap(&harr[i], &harr[parent(i)]);
9.             i = parent(i);
10.    }
11. }
```

a) Line – 3
b) Line – 5
c) Line – 6
d) Line – 7
View Answer

Answer: c
Explanation: The condition under while condition is wrong for a (min) binary heap The correct condition should be while(i!=0 && harr[parent(i)] > harr[i]). Otherwise the constructed heap will be a max-binary heap.
Topic 82. Weak Heap

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Weak Heap”.

1. Choose the correct properties of weak-heap.
a) Every node has value greater than the value of child node
b) Every right child of node has greater value than parent node
c) Every left child of node has greater value than parent node
d) Every left and right child of node has same value as parent node
View Answer

Answer: b
Explanation: This is the property of a weak heap.

2. Left child of parent node has value lesser than the parent node.
a) True
b) False
View Answer

Answer: b
Explanation: Weak heap has no left child.

3. What is the other name of weak heap?
a) Min-heap
b) Max-heap
c) Relaxed -heap
d) Leonardo heap
View Answer

Answer: c
Explanation: Relaxed heap is just another name of weak heap.

4. What is the worst case time in searching minimum value in weak -heap?
a) O(log n)
b) O(n)
c) O(n logn)
d) O(1)
View Answer

Answer: d
Explanation: Weak heap is an array based form that supports the operation of finding a minimum in O(1).

5. The total comparisons in finding both smallest and largest elements are
a) 2*n +2
b) n + ((n+1)/2) -2
c) n+logn
d) n^2
View Answer

Answer: b
Explanation: The total comparisons in finding smallest and largest elements is n + ((n+1)/2) – 2.

6. What is the complexity of given function of insertion.

```
insert(int n)
{
 if(buffer_size()< maxi_biffer_size())
 buffer_aar[ind]==n;
 else
 move_to_heap(buffer,buffer+maxi_buffer_size())
}
```

a) O(logn)
b) amortized O(1)
c) O(n)
d) O (n*logn)
View Answer

Answer: b
Explanation: Use a buffer array to store a fixed number of elements when the buffer is full the content of buffer is moved to heap.As a result the complexity
is amotized O(1).

7. Does there exist a heap with seven distinct elements so that the Inorder traversal gives the element in sorted order.

a) Yes
b) No
View Answer

Answer: b
Explanation: No, The inorder traversal will not give elements in sorted order. As heap is implemented as either min-heap or max-heap, the root will be have highest or lowest value than remaining values of the nodes. So this traversal will not give a sorted list.

8. The leaf node for a heap of height h will be at which position.
a) h
b) h-1
c) h or h-1
d) h-2
View Answer

Answer: c
Explanation: A complete binary tree is also a heap so by the property of binary tree the leaf nodes will be must at height h or h-1.
Topic 83. Binomial and Fibonacci Heap

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Binomial and Fibonacci Heap”.

1. The main distinguishable characterstic of a binomial heap from a binary heap is that
a) it allows union operations very efficiently
b) it does not allow union operations that could easily be implemented in binary heap
c) the heap structure is not similar to complete binary tree
d) the location of child node is not fixed i.e child nodes could be at level (h-2) or (h-3), where h is height of heap and h>4
View Answer

Answer: a
Explanation: The main use of binomial heap is to unify two different heap efficiently.

2. The number of trees in a binomial heap with n nodes is
a) logn
b) n
c) nlogn
d) n/2
View Answer

Answer: a
Explanation: At each depth there is a binomial tree in a binomial heap.

3. In a binomial heap the root value is greater than left child and less than right child.
a) True
b) False
View Answer

Answer: b
Explanation: Binomial tree used in making binomial heap follows min heap property.

4. Given the pseudo code, state whether the function for merging of two heap is correct or not?

```
mergeTree(p,q)
    if p.root.value <= q.root.value
       return p.addTree(q)
    else
       return q.addTree(p)
```

a) True
b) False
View Answer

Answer: a
Explanation: Binomial heap has a property that root value is less than both the child node’s value. So the given function of merging two different heap is correct.

5. What is order of resultant heap after merging two tree of order k?
a) 2*k
b) k+1
c) k*k
d) k+logk

View Answer

Answer: b
Explanation: This could be easily verified by looking at the structure of a binomial heap.

6. Time taken in decreasing the node value in a binomial heap is
a) O(n)
b) O(1)
c) O(logn)
d) O(nlogn)
View Answer

Answer: c
Explanation: Decreasing a node value may result in violating the min property. As a result be there would be exchange in the value of parent and child which at max goes up to height of the heap.

7. What does this pseudo_code return ?

```
int myfun(heap_arr[])
{
 int mini=INF;
 for(int i=0;i<tot_node;i++)
 mini=min(mini,heap_arr)
 return mini;
}
```

a) Last added element to heap
b) First element added to heap
c) Root of the heap
d) Leftmost node of the heap
View Answer

Answer: c
Explanation: The function return minimum value in the heap_Array which is equal to the root value of the heap.

8. Which of these operations have same complexities?
a) Insertion, find_min
b) Find_min, union
c) Union, Insertion
d) Deletion, Find _max
View Answer

Answer: c
Explanation: With proper implementation using link list find_min and find_max operation can be done in O(1), while the remaining takes O(logn) time.

9. The Statement “Fibonacci heap has better amortized running time in compare to a binomial heap”.
a) True
b) False
View Answer

Answer: a
Explanation: Overall complexity of insertion, merging, deleting is in order of O((a+b)logn) For Fibonacci the complexity reduces to O(a+ blogn).

10. Given a heap of n nodes.The maximum number of tree for building the heap is.
a) n
b) n-1
c) n/2
d) logn
View Answer

Answer: a
Explanation: Each node could be seen as a tree with only one node and as a result maximum subtree in the heap is equal to number of nodes in the heap.

11. Choose the option with function having same complexity for a fibonacci heap.
a) Insertion, Union
b) Insertion, Deletion
c) extract_min, insertion
d) Union, delete
View Answer

Answer: a
Explanation: For a fibonacci heap insertion, union take O(1) while remaining take O(logn) time.

12. What is wrong with the following code of insertion in fibonacci heap.
Choose the correct option

```
FIB-INSERT(H, x)
degree[x]= 0
p[x]=  NIL
child[x] =NIL
left[x] =x
right[x] =x
mark[x] =FALSE
concatenate the root list containing x with root list H
if min[H] = NIL or key[x] > key[min[H]]
then min[H]= x
n[H]= n[H] + 1
```

a) Line -11
b) Line -3
c) Line 9
d) Line 7
View Answer

Answer: c
Explanation: The main characterstics of a fibonacci heap is violated since min[H] must conatin one with smallest value.

13. What will be the order of new heap created after union of heap H1 and H2 when created by the following code.Initially both are of the order n.

```
FIB_UNION(H1,H2)
{
 H =MAKE_HEAP()
 min[H]= min[H1]
 concatenate the root list of H2 with the root list of H
 if (min[H1] = NIL) or (min[H2]!= NIL and min[H2] < min[H1])
 then min[H] = min[H2]
 n[H]=  n[H1] + n[H2]
 free the objects H1 and H2
 return H
}
```

a) n+1
b) n+n/2
c) nlogn
d) 2*n
View Answer

Answer: a
Explanation: Union of two trees increase the order of the resultant by atmost value 1.
Topic 84. D-ary Heap

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “D-ary Heap”.

1. d-heap is similar to that of a?
a) binary heap
b) fibonacci heap
c) leftist heap
d) treap
View Answer

Answer: a
Explanation: A d-heap is similar to that of a binary heap except that binary heaps have two children and d-heaps have d children.

2. d-heap is shallower than a binary heap.
a) true
b) false
View Answer

Answer: a
Explanation: d-heap is much shallower than a binary heap with respect to performance efficiency of insert and delete operations.

3. Which operation cannot be directly performed in a d-heap?

a) insert
b) delete
c) find
d) create
View Answer

Answer: c
Explanation: Find operation in a d-heap cannot be performed as in other heaps. This is the main weakness of d-heap.

4. Which operation is not efficiently performed in a d-heap?
a) insert
b) delete
c) find
d) merge
View Answer

Answer: d
Explanation: Unlike find operation, which cannot be performed in a d-heap, the task of merging two d-heaps is very difficult.

5. What is the run time efficiency of an insertion algorithm in d-heap?
a) O(N)
b) O(log N)
c) O($\log_d$ N)
d) O(N^d)
View Answer

Answer: c
Explanation: The run time efficiency of an insertion algorithm in a d-heap is found to be O($\log_d$ N) where d is the number of children.

6. How many comparisons will occur while performing a delete-min operation?
a) d
b) d-1
c) d+1
d) 1
View Answer

Answer: b
Explanation: Since, the delete-min operation is more expensive and the heap is shallow, the minimum of d elements can be found using d-1 comparisons.

7. How many basic operations can be performed in a d-heap?
a) 1
b) 2
c) 3
d) 4
View Answer

Answer: b
Explanation: The two basic operations performed in a d-heap are insert and delete-min operations.

8. What is the run time efficiency of delete-min operation?
a) O(log N)
b) O($\log_d$ N)
c) O(d $\log_d$ N)
d) O(d)
View Answer

Answer: c
Explanation: The run time efficiency of a delete-min algorithm using d-1 comparisons is mathematically found to be O(d $\log_d$ N).

9. The following figure is an example for

a) d-heap
b) binary heap
c) leftist heap
d) skew heap
View Answer

Answer: a
Explanation: The given heap is a d-heap since it looks like a binary heap with d- children. Here, d=3.

10. Multiplication and division to find children and parents cannot be implemented in a d-heap.
a) true
b) false
View Answer

Answer: b
Explanation: Multiplication and division for finding children and parents can be implemented in a d-heap but d should be a power of 2.

11. How many secondary operations are performed in a d-heap?
a) 1
b) 2
c) 3
d) 4
View Answer

Answer: d
Explanation: The other operations that can be performed in a d-heap are increasekey, decreasekey, buildheap and delete.

12. On which data structure is a d-ary heap based?
a) stack
b) queue
c) linked list
d) priority queue
View Answer

Answer: d
Explanation: d-ary heap is a priority queue based data structure that is a generalization of binary heaps.
Topic 85. Ternary Heap – 1

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Ternary Heap – 1”.

1. What is the smallest element of the given minimum ternary heap?

a) 1
b) 10
c) 18
d) 20
View Answer

Answer: a
Explanation: Ternary heap is a type of data structure in the field of computer science. It is a part of the Heap data structure family. Minimum ternary heap has the smallest element as its root node. The parent node is all either equal or less than children node in a minimum ternary heap.

2. What is the highest element of the given maximum ternary heap?

a) 31
b) 10
c) 18
d) 20
View Answer

Answer: a
Explanation: Ternary heap is a type of data structure in the field of computer science. It is a part of the Heap data structure family. Maximum ternary heap has the highest element as its root node. The parent node is all either equal or greater than children node in a maximum ternary heap.

3. What is the child of smallest element of the given minimum ternary heap?

a) 1
b) 10
c) 22
d) 24
View Answer

Answer: b

Explanation: Minimum ternary heap has the smallest element as its root node. The parent node is all either equal or less than children node in a minimum ternary heap. In the above minimum ternary heap, the smallest element is 1 and its children are 10, 18, 20.

4. What are the siblings of smallest element of the given maximum ternary heap?

a) 31
b) 12
c) 18
d) 22
View Answer

Answer: c
Explanation: Maximum ternary heap has the highest element as its root node. The parent node is all either equal or greater than children node in a maximum ternary heap. The smallest element in the maximum ternary heap is 10 and its siblings are 18, 20.

5. What is the height of a given minimum ternary heap?

a) 1
b) 10
c) 2
d) 24
View Answer

Answer: a
Explanation: Minimum ternary heap has the smallest element as its root node. The parent node is all either equal or less than children node in a minimum ternary heap. Height is the total length from the root node to the leaf node. So the height of the minimum ternary heap is 1.

6. What is the ancestor of the leaf node in a given minimum ternary heap?

a) 1
b) 10
c) 18
d) 20
View Answer

Answer: a
Explanation: Minimum ternary heap has the smallest element as its root node. The parent node is all either equal or less than children node in a minimum ternary heap. Ancestor is the node falling on the path from that node to the root node. So here ancestor of all leaf nodes is 1.

7. Which property should ternary heap hold for execution?
a) Associative
b) Commutative
c) Tree
d) Heap
View Answer

Answer: d
Explanation: Ternary heap is a type of data structure in the field of computer science. It is a part of the Heap data structure family. So, it should hold all the properties of Heap that is all the levels of the heap has to be filled from left to right.

8. Should leaves in ternary heap be distributed from left to right.
a) True
b) False
View Answer

Answer: a
Explanation: Ternary heap is a type of data structure in the field of computer science. It is a part of the Heap data structure family. So, it should hold all the properties of Heap that is all the levels of the heap has to be filled from left to right.

9. What is the process of building a ternary heap called?
a) Heapify
b) Hashing
c) Linking
d) Merging
View Answer

Answer: a
Explanation: Ternary heap is a type of data structure in the field of computer science. It is a part of the Heap data structure family. So, the process of building a ternary heap is known as Heapify.

10. Which type of data structure is a ternary heap?
a) Array
b) Hash
c) Priority Queue
d) Priority Stack
View Answer

Answer: c
Explanation: Ternary heap is a type of data structure in the field of computer science. It is a part of the Heap data structure family. It is a priority queue type of data structure that follows all the property of heap.

11. Is the priority queue abstract data type.
a) True
b) False
View Answer

Answer: a
Explanation: Priority queue is an abstract data type. It is also the extension of the Queue data structure where all the elements have been assigned some priority and on the basis of this priority, the elements are dequeued from the structure.

12. What is a ternary heap?
a) An array with three elements
b) Linked list with three elements
c) Tree with three children
d) Heap with all nodes having three children
View Answer

Answer: d
Explanation: Ternary heap is a type of data structure in the field of computer science. It is a part of the Heap data structure family. So, it follows all the property of heap. Therefore, all the nodes in the ternary heap have 3 nodes.

13. Who invented d-ary heap?
a) Carl Rick
b) Alan Turing
c) Donald Johnson
d) Euclid
View Answer

Answer: c
Explanation: Ternary heap is a type of data structure in the field of computer science. It is a part of the Heap data structure family. The d-ary heap was invented by Donald Johnson in the year 1975.
Topic 86. Ternary Heap – 2

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Ternary Heap – 2”.

1. What is the time complexity for inserting a new item in a ternary heap of n elements?
a) O (log n/ log 3)
b) O (n!)
c) O (n)
d) O (1)
View Answer

Answer: a
Explanation: In order to insert a new item in a ternary heap data structure having n elements, the heap has great efficiency for inserting them. So the time complexity for worst case is found to be O (log n/ log 3).

2. Is decrease priority operation performed more quickly in a ternary heap with respect to the binary heap.
a) True
b) False
View Answer

Answer: a
Explanation: Ternary heap is a type of data structure in the field of computer science. It is a part of the Heap data structure family. Due to the swapping process, the decrease priority operation performs more quickly in a ternary heap.

3. What is the time complexity for decreasing priority of key in a minimum ternary heap of n elements?
a) O (log n/ log 3)

b) O (n!)
c) O (n)
d) O (1)
View Answer

Answer: a
Explanation: In order to decrease the priority of an item in a ternary heap data structure having n elements, the heap has great efficiency for decreasing them. So the time complexity for worst case is found to be O (log n/ log 3). This is due to the upwards swapping process.

4. What is the time complexity for increasing priority of key in a maximum ternary heap of n elements?
a) O (log n/ log 3)
b) O (n!)
c) O (n)
d) O (1)
View Answer

Answer: a
Explanation: In order to increase the priority of an item in a ternary heap data structure having n elements, it performs upwards swapping. So the time complexity for worst case is found to be O (log n/ log 3).

5. What is the time complexity for deleting root key in a ternary heap of n elements?
a) O (log n/ log 3)
b) O (3log n/ log 3)
c) O (n)
d) O (1)
View Answer

Answer: b
Explanation: In order to delete a root key in a ternary heap data structure having n elements, it performs downward swapping. So the time complexity for worst case is found to be O (3log n/ log 3).

6. What is the time complexity for increasing priority of key in a minimum ternary heap of n elements?
a) O (log n/ log 3)
b) O (3log n/ log 3)
c) O (n)
d) O (1)
View Answer

Answer: b
Explanation: In order to the increasing the priority of key in a minimum ternary heap data structure having n elements, it performs downward swapping. So the time complexity for worst case is found to be O (3log n/ log 3).

7. What is the time complexity for decreasing priority of key in a maximum ternary heap of n elements?
a) O (log n/ log 3)
b) O (3log n/ log 3)
c) O (n)
d) O (1)
View Answer

Answer: b
Explanation: In order to decrease the priority of key in a maximum ternary heap data structure having n elements, it performs downward swapping. So the time complexity for worst case is found to be O (3log n/ log 3).

8. Do ternary heap have better memory cache behavior than binary heap.
a) True
b) False
View Answer

Answer: a
Explanation: Ternary heap is a type of data structure in the field of computer science. It is a part of the Heap data structure family. Due to the swapping process, they have better memory cache behavior.

9. What is the time complexity for creating a ternary heap using swapping?
a) O (log n/ log 3)
b) O (n!)
c) O (n)
d) O (1)
View Answer

Answer: c
Explanation: Ternary Heap can be formed by two swapping operations. Therefore, the time complexity for creating a ternary heap using two swapping operation is found to be O (n).

10. Which of the following is the application of minimum ternary heap?
a) Prim's Algorithm
b) Euclid's Algorithm
c) Eight Queen Puzzle
d) Tree
View Answer

Answer: a
Explanation: When working on the graph in the computer science field, the Prim's Algorithm for spanning trees uses a minimum ternary heap as there are delete operation equal to a number of edges and decrease priority operation equal to the number of vertices associated with the graph.
Topic 87. Pairing Heap

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on "Pairing Heap".

1. What is the reason for the efficiency of a pairing heap?
a) simplicity
b) time-efficient
c) space-efficient
d) advanced
View Answer

Answer: a
Explanation: The reason for the simplicity of a pairing heap is its simplicity as it is simpler and outperform other heap structures.

2. How is a pairing heap represented?
a) binary tree
b) fibonacci tree
c) heap ordered tree
d) treap
View Answer

Answer: c
Explanation: A pairing heap is represented as a heap-ordered tree and the analysis of pairing heap is open.

3. The actual pairing heap implementation uses the right child and left child representation.
a) true
b) false
View Answer

Answer: b
Explanation: The actual pairing heap implementation uses a left child and right sibling representation since it follows heap order property.

4. Which node contains a pointer to its parent?
a) root node
b) right most child
c) left most child
d) left sibling
View Answer

Answer: c
Explanation: A node that is a leftmost node contains a pointer to its parent, otherwise, the node is a right sibling.

5. Which of the heaps is implemented by the following figure?

a) fibonacci heaps
b) pairing heap
c) skew heap
d) leftist heap
View Answer

Answer: b
Explanation: The above figure is a representation of a pairing heap because it has left children and right siblings.

6. What is the basic operation performed in a pairing heap?

a) merge
b) deletion
c) insertion
d) swapping
View Answer

Answer: a
Explanation: The basic operation performed in a pairing heap is merging. Insertion is also done by merging.

7. If there are c children of the root, how many calls to the merge procedure is required to reassemble the heap?
a) c
b) c+1
c) c-1
d) 1
View Answer

Answer: c
Explanation: If there are c children of the root, then c-1 merges are required to reassemble the pairing heap.

8. Which of the following methods is the best choice for complex applications?
a) binary heap
b) d-heap
c) treap
d) pairing heap
View Answer

Answer: d
Explanation: Pairing heap is the best choice for complex applications because it is simple and better than the others.

9. Pairing heaps time complexity was inspired by that of?
a) splay tree
b) treap
c) red-black tree
d) avl tree
View Answer

Answer: a
Explanation: The pairing heaps insertion, deletion and search time complexity was initially inspired by that of splay trees.

10. The roots of the elements of the subtrees are smaller than the root of the heap.
a) True
b) False
View Answer

Answer: b
Explanation: The heap ordering property requires that all the root elements of the subtrees in the list are not smaller than the root element of the heap.

11. The amortized time efficiency for performing deletion of a minimum element is?
a) O(N)
b) O(log N)
c) $O(N^2)$
d) O(M log N)
View Answer

Answer: b
Explanation: The amortized time efficiency for performing deletion of a minimum element is mathematically found to be O(log N).

12. Out of the following given options, which is the fastest algorithm?
a) fibonacci heap
b) pairing heap
c) d-ary heap
d) binary heap
View Answer

Answer: a
Explanation: Although pairing heap is an efficient algorithm, it is worse than the Fibonacci heap. Also, pairing heap is faster than d-ary heap and binary heap.

13. What is the run time efficiency of an insertion algorithm?
a) O(N)
b) O(log N)
c) O(N^2)
d) O(M log N)
View Answer

Answer: a
Explanation: The run time efficiency of an insertion algorithm in a pairing heap is mathematically found to be O(N).
Topic 88. Leftlist Heap

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on "Leftlist Heap".

1. Pointer manipulation is generally more time-consuming than multiplication and division.
a) true
b) false
View Answer

Answer: a
Explanation: Use of pointers for merging reduces the speed of other operations. This is the main drawback of all advanced data structures.

2. How many properties does a leftist heap support?
a) 1
b) 2
c) 3
d) 4
View Answer

Answer: c
Explanation: A leftist heap supports two properties- structural property, ordering property and a heap order property.

3. In a leftist heap, the null path length of a null node is defined as?
a) 0
b) 1
c) null
d) -1
View Answer

Answer: d
Explanation: In a leftist heap tree, the null path length of a null node with no children is defined as -1.

4. How many nodes does a leftist tree with r nodes must have?
a) 2^r
b) 2^r-1
c) 2r
d) 2r-1
View Answer

Answer: b
Explanation: A leftist tree with r nodes on the right path is proved to have at least 2^r-1 nodes. This theorem is proved by induction.

5. Which of the following operations does not destroy the leftist heap property?
a) insert
b) merge
c) delete
d) swap
View Answer

Answer: c
Explanation: Performing insert and merge operations on the right path could destroy the leftist heap property. It is extremely easy to restore that property.

6. What is the fundamental operation on leftist heap?
a) insertion
b) merging
c) deletion
d) swapping
View Answer

Answer: b
Explanation: The fundamental operations on leftist heaps is merge. Insertion operation is a merge of a one-node heap with a larger heap.

7. A leftist heap is also said to be a binary heap.
a) true
b) false
View Answer

Answer: a
Explanation: A leftist heap has a structural property and an ordering property which is similar to that of a binary heap. Hence, leftist heap is also said to be binary heap.

8. What is the efficiency of merge used in leftist heaps?
a) O(N)
b) O(N log N)
c) O(M log N)
d) O(log N)
View Answer

Answer: d
Explanation: The efficiency of merge operations in leftist heap is mathematically found to be O(log N) which is the same in binary heaps.

9. What is the node path length of a node with 0 or 1 child?
a) 1
b) -1
c) 0
d) null
View Answer

Answer: c
Explanation: The length of the shortest path from a node to a node without two children is defined as 0.

10. Why is this heap named leftist heap?
a) only left subtrees exist
b) the tree is biased to get deep down the left
c) it is balanced
d) right trees are unbalanced
View Answer

Answer: b
Explanation: The heap is named as leftist heap because it tends to have deep left paths. It follows that the right path ought to be short.

11. In a leftist heap, all the operations should be performed on?
a) left path
b) centre path
c) right path
d) root
View Answer

Answer: c
Explanation: All the operations are performed on the right path because right paths are short. However, insertion and merges cannot be performed on the right path.

12. What would be the result if the left subtree of the root has a null path length of 1 and the right subtree has a null path length of 2?
a) merge occurs without violation
b) violation at left subtree
c) violation at right subtree
d) violation at the root
View Answer

Answer: d
Explanation: When two leftist heaps are merged, if the left subtree of the root has a null path length of 1 and the right subtree has a null path length of 2, leftist property is violated at the root.

13. What happens if the null path length is not updated?
a) error occurs

b) all null path lengths will be 0
c) all null path lengths will be -1
d) all null path lengths will be 1
View Answer

Answer: b
Explanation: While performing insertion via merge operation in a leftist heap, if the null path length is not updated, all null path lengths will be 0.

14. What is the time taken to delete a minimum element in a leftist heap?
a) O(N)
b) O(N log N)
c) O(log N)
d) O(M log N)
View Answer

Answer: c
Explanation: The time taken to delete a minimum element in a leftist heap is mathematically found to be O(log N).

15. In what time can a leftist heap be built?
a) O(N)
b) O(N log N)
c) O(log N)
d) O(M log N)
View Answer

Answer: a
Explanation: The mathematical calculation yields a result that, a leftist heap can be built in O(N) time by building a binary heap.
Topic 89. Skew Heap

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Skew Heap”.

1. ___________ is a self-adjusting version of a leftist heap.
a) Rightist heap
b) Skew heap
c) d-heap
d) Binary heap
View Answer

Answer: b
Explanation: A skew heap is a self-adjusting version of a leftist heap and it is simpler to implement.

2. The worst case running time of all operations in a skew heap is given as?
a) O(N)
b) O(N log N)
c) $O(N^2)$
d) O(M log N)
View Answer

Answer: a
Explanation: The worst case running time of all operations in a skew heap is mathematically found to be O(N).

3. What is the amortized cost per operation of a skew heap?
a) O(N)
b) O(N log N)
c) $O(N^2)$
d) O(log N)
View Answer

Answer: d
Explanation: The amortized cost per operation of a skew heap is O(log N) since the worst case analysis of skew heap is O(N) and splay tree is O(M log N).

4. The relationship of skew heaps to leftist heaps is analogous to that of?
a) Splay tree and AVL tree
b) Red black tree and AVL tree
c) Binary tree and Splay tree
d) Binary tree and Red black tree
View Answer

Answer: a
Explanation: Splay tree is a self -adjusting version of AVL tree. Similarly, skew heap is a self-adjusting version of leftist heap.

5. What is the fundamental operation performed in skew heaps?
a) intersection
b) difference
c) merging
d) sorting
View Answer

Answer: c
Explanation: The fundamental operation of skew heaps is merging. Hence, it is similar to that of a leftist heap.

6. What is the time per operation of merging, insertion and deletion operations in a skew heap?
a) O(N)
b) O(log N)
c) O(N log N)
d) O(N^2)
View Answer

Answer: b
Explanation: Skew heaps support merging, insertion and deletion all effectively in O(log N) time per operation.

7. Why would a recursive implementation fail in skew heaps?
a) skew heaps are self adjusting
b) efficiency gets reduced
c) lack of stack space
d) time complexity
View Answer

Answer: c
Explanation: In skew heaps, a recursive implementation could fail because of lack of stack space even though performance is acceptable.

8. Which of the following is difficult to determine the right path length?
a) Skew heaps
b) Binomial tree
c) Leftist heap
d) d-heap
View Answer

Answer: a
Explanation: It is an open problem to determine precisely the expected right path length of both leftist and skew heaps and comparatively, the latter is difficult.

9. The worst case analysis for a naïve merge is given as?
a) O(N)
b) O(log N)
c) O(N log N)
d) O(N^2)
View Answer

Answer: a
Explanation: The worst-case analysis for the naïve merge is an insertion in a right heavy tree. So, insertion takes O(N).

10. How many types of the merge are available in skew heaps?
a) 1
b) 2
c) 3
d) 4
View Answer

Answer: b
Explanation: Two kinds of the merge are available in skew heaps- naïve merge and skew merge.

11. Naïve merge cannot be done in a skew merge.
a) true
b) false
View Answer

Answer: b
Explanation: One way of doing skew merge is to begin with naïve merge and then swapping the left and right children of the tree.

12. What is the amortized efficiency of skew merge?
a) O(N)
b) O(log N)
c) O(N log N)
d) $O(N^2)$
View Answer

Answer: b
Explanation: The amortized efficiency of a skew heap is mathematically found to be O(log N).

13. In skew heaps, certain constraints are to be met in order to perform swapping.
a) true
b) false
View Answer

Answer: b
Explanation: In skew heaps, swaps are unconditional. It is done with the exception that the largest of all nodes does not have its children swapped.
Topic 90. Min/Max Heap

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Min/Max Heap”.

1. Descending priority queue can be implemented using ______
a) max heap
b) min heap
c) min-max heap
d) trie
View Answer

Answer: a
Explanation: Descending priority queue arranges the elements based on their priority or value and allows removing the elements in descending order. So, it can be efficiently implemented using max heap.

2. Min heap can be used to implement selection sort.
a) True
b) False
View Answer

Answer: a
Explanation: In min heap, the insertion and deletion operation takes O(logn) time. Therefore, a selection sort with n insertions and n deletions can be implemented using a min heap in O(nlogn) operations.

3. Which of the following is the valid min heap?
a)
b)
c)
d)
View Answer

Answer: d
Explanation: In min heap the smallest is located at the root and the largest elements are located at the leaf nodes. So, all leaf nodes need to be checked to find the largest element.

4. The procedure given below is used to maintain min-order in the min heap. Find out the missing statements, represented as X.

```
procedure TrickleDownMin(i)
  if A[i] has children then
  m := index of smallest of the children
          or grandchildren (if any) of A[i]
  if A[m] is a grandchild of A[i] then
    if A[m] < A[i] then
    swap A[i] and A[m]
    X: _____________________
     ___________________
    endif
   TrickleDownMin(m)
   endif
  else //{A[m] is a child of A[i]}
```

```
   if A[m] < A[i] then
    swap A[i] and A[m]
  endif
  endif
```

a) if A[m] > A[parent(m)] then
swap A[m] and A[parent(m)]
b) if A[m] > A[parent(m)] then
swap A[i] and A[parent(m)]
c) if A[m] < A[parent(m)] then
swap A[m] and A[parent(m)]
d) if A[m] > A[parent(m)] then
swap A[i] and A[parent(m)]
View Answer

Answer: a
Explanation: In TrickleDownMin() procedure, we maintain the min-ordering of the min heap. In this procedure, we locate the lowest child or grandchild of the element at positions i. If the lowest element is grandchild then we check that it is smaller than both, its parent and A[i].

5. The ascending heap property is ___________
a) A[Parent(i)] =A[i]
b) A[Parent(i)] <= A[i]
c) A[Parent(i)] >= A[i]
d) A[Parent(i)] > 2 * A[i]
View Answer

Answer: b
Explanation: The min heap is also known as ascending heap. Min heap of size n is an almost complete binary tree of n nodes such that the element at each node is greater than or equal to the element at its parent node.

6. The procedure FindMin() to find the minimum element and the procedure DeleteMin() to delete the minimum element in min heap take _________
a) logarithmic and linear time constant respectively
b) constant and linear time respectively
c) constant and quadratic time respectively
d) constant and logarithmic time respectively
View Answer

Answer: d
Explanation: In the min heap, the root is the maximum element in the tree. So, locating it takes constant time, but deleting it takes logarithmic time. Because after deleting it, the root is replaced with last element and then the procedure to maintain the min ordering is invoked.

7. Which one of the following array elements represents a binary min heap?
a) 12 10 8 25 14 17
b) 8 10 12 25 14 17
c) 25 17 14 12 10 8
d) 14 17 25 10 12 8
View Answer

Answer: b
Explanation: A tree is min heap when data at every node in the tree is smaller than or equal to it's children' s data. So, only 8 10 12 25 14 17 generates required tree.

8. In a binary min heap containing n elements, the largest element can be found in __________ time.
a) O(n)
b) O(nlogn)
c) O(logn)
d) O(1)
View Answer

Answer: a
Explanation: In min heap the smallest is located at the root and the largest elements are located at the leaf nodes. So, all leaf nodes need to be checked to find the largest element. Thus, worst case time will be O (n).

9. Min heap is a complete binary tree.
a) True
b) False
View Answer

Answer: a
Explanation: A tree, in which all levels are fully filled, except possibly the last level, is called as the complete binary tree. And min heap maintains shape property, so it is a complete binary tree. The shape property ensures that all levels in the min heap are fully filled, except the last one, and, if the last level is not filled completely, then fill the elements from left to right.

10. What will be the position of 5, when a max heap is constructed on the input elements 5, 70, 45, 7, 12, 15, 13, 65, 30, 25?
a) 5 will be at root
b) 5 will be at last level
c) 5 will be at second level
d) 5 can be anywhere in heap
View Answer

Answer: b
Explanation: In max heap the greatest element is at the root and the smallest elements are at the last level. As 5 is the smallest input element, it will be at the last level.
Topic 91. Trie

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Trie”.

1. Trie is also known as _________
a) Digital Tree
b) Treap
c) Binomial Tree
d) 2-3 Tree
View Answer

Answer: a
Explanation: Trie is a very useful data structure which is based on the prefix of a string. Trie is used to represent the “Retrieval” of data and thus the name Trie. And it is also known as a digital tree.

2. What traversal over trie gives the lexicographical sorting of the set of the strings?
a) postorder
b) preorders
c) inorder
d) level order
View Answer

Answer: c
Explanation: In trie, we store the strings in such a way that there is one node for every common prefix. Therefore the inorder traversal over trie gives the lexicographically sorted set of strings.

3. Which of the following is the efficient data structure for searching words in dictionaries?
a) BST
b) Linked List
c) Balancded BST
d) Trie
View Answer

Answer: d
Explanation: In a BST, as well as in a balanced BST searching takes time in order of mlogn, where m is the maximum string length and n is the number of strings in tree. But searching in trie will take O(m) time to search the key.

4. Which of the following special type of trie is used for fast searching of the full texts?
a) Ctrie
b) Hash tree
c) Suffix tree
d) T tree
View Answer

Answer: c
Explanation: Suffix tree, a special type of trie, contains all the suffixes of the given text at the key and their position in the text as their values. So, suffix trees are used for fast searching of the full texts.

5. Following code snippet is the function to insert a string in a trie. Find the missing line.

```
private void insert(String str)
   {
      TrieNode node = root;
      for (int i = 0; i < length; i++)
      {
         int index = key.charAt(i) - 'a';
```

```
            if (node.children[index] == null)
                node.children[index] = new TrieNode();

            ______________________
        }

        node.isEndOfWord = true;
    }
```

a) node = node.children[index];
b) node = node.children[str.charAt(i + 1)];
c) node = node.children[index++];
d) node = node.children[index++];
View Answer

Answer: a
Explanation: In the insert() method we search if the string is present or not. If the string is not present, then we insert the string into the trie. If it is present as the prefix, we mark the leaf node. So, correct option is node = node.children[index];.

6. Which of the following is not true?
a) Trie requires less storage space than hashing
b) Trie allows listing of all the words with same prefix
c) Tries are collision free
d) Trie is also known as prefix tree
View Answer

Answer: a
Explanation: Both the hashing and the trie provides searching in the linear time. But trie requires extra space for storage and it is collision free. And trie allows finding all the strings with same prefix, so it is also called prefix tree.

7. A program to search a contact from phone directory can be implemented efficiently using ______
a) a BST
b) a trie
c) a balanced BST
d) a binary tree
View Answer

Answer: b
Explanation: Dictionaries, phone directories can be implemented efficiently using the trie. Because it trie provides the efficient linear time searching over the entries.

8. What can be the maximum depth of the trie with n strings and m as the maximum sting the length?
a) log2n
b) log2m
c) n
d) m
View Answer

Answer: d
Explanation: In the trie, the strings are stored efficiently based on the common prefixes. And trie has maximum fan-out 26 if english alphabets are considered. Owing to this, the maximum depth is equal to the maximum string length.

9. Which of the following is true about the trie?
a) root is letter a
b) path from root to the leat yields the string
c) children of nodes are randomly ordered
d) each node stores the associated keys
View Answer

Answer: b
Explanation: A trie is an ordered tree where (i) the root represents an empty string(“”) (ii) each node other than root is labeled with a character (iii) the children of a nodes are lexicographically ordered (iv) the paths from the leaves to the root yields the strings.

10. Auto complete and spell checkers can be implemented efficiently using the trie.
a) True
b) False
View Answer

Answer: a
Explanation: Trie provides fast searching and storing of the words. And tries stores words in lexicographical order so, trie is the efficient data structure for implementation of spell checkers and auto complete.

Topic 92. Suffix Tree – 1

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Suffix Tree – 1”.

1. What is the other name for Suffix Tree?
a) Array
b) Stack
c) Priority Queue
d) PAT Tree
View Answer

Answer: d
Explanation: In computer science, a suffix tree is also known as PAT tree or position tree. It is a compressed search tree or prefix tree in which keys contain the suffix of text values as the text position.

2. Which tree allows fast implementation of string operation?
a) Rope Tree
b) Suffix Tree
c) Tango Tree
d) Top Tree
View Answer

Answer: b
Explanation: In computer science, a suffix tree is also known as PAT tree or position tree. It is a compressed search tree or prefix tree in which keys contain the suffix of text values as the text position. It allows fast string operation to be carried out by the user.

3. How much time does construction of suffix tree take?
a) O (log M)
b) O (M!)
c) Exponential to Length of Tree
d) Linear to Length of Tree
View Answer

Answer: d
Explanation: Suffix tree is also known as PAT tree or position tree. It is a compressed search tree or prefix tree in which keys contain the suffix of text values as the text position. It allows fast string operation. Total time taken for construction of suffix tree is linear to the length of the tree.

4. How much space does construction of suffix tree takes?
a) O (log M)
b) Exponential to Length of Tree
c) O (M!)
d) Linear to Length of Tree
View Answer

Answer: d
Explanation: Suffix tree is also known as PAT tree or position tree. It is a compressed search tree or prefix tree in which keys contain the suffix of text values as the text position. It allows fast string operation. Total space taken for construction of a suffix tree is linear to the length of the tree.

5. Which tree provides a linear time solution for substring operation?
a) Rope Tree
b) Suffix Tree
c) Tango Tree
d) Top Tree
View Answer

Answer: b
Explanation: It is a compressed search tree or prefix tree in which keys contain the suffix of text values as the text position. It allows fast string operation to be carried out by the user. The substring operation can be performed by suffix tree in linear time.

6. Who proposed the concept of Suffix Tree?
a) Weiner
b) Samuel F. B. Morse
c) Friedrich Clemens Gerke
d) Alexander Morse
View Answer

Answer: a
Explanation: In computer science, a suffix tree is also known as PAT tree or position tree. It is a compressed search tree or prefix tree in which keys contain the suffix of text values as the text position. The concept of Suffix Tree was introduced by Weiner in 1973.

7. Who among the following provided the first online contribution of Suffix Tree?
a) Weiner
b) Samuel F. B. Morse
c) Ukkonen
d) Alexander Morse
View Answer

Answer: c
Explanation: In computer science, a suffix tree is also known as PAT tree or position tree. The concept of Suffix Tree was introduced by Weiner in 1973. Ukkonen provided the first online contribution of Suffix tree which had the time complexity of the fastest algorithm of that period.

8. What is the time complexity of Uttkonen's algorithm?
a) O (log n!)
b) O (n!)
c) O (n^2)
d) O (n log n)
View Answer

Answer: d
Explanation: The concept of Suffix Tree was introduced by Weiner in 1973. Ukkonen provided the first online contribution of Suffix tree which had the time complexity of the fastest algorithm of that period. Ukkonen's algorithm had a time complexity of n log n.

9. Who among the following provided the first suffix tree contribution for all alphabet?
a) Weiner
b) Farach
c) Ukkonen
d) Alexander Morse
View Answer

Answer: b
Explanation: The concept of Suffix Tree was introduced by Weiner in 1973. Ukkonen provided the first online contribution of Suffix tree which had the time complexity of the fastest algorithm of that period. Farach gave the first suffix tree contribution for all alphabets in 1997.

10. Who among the following algorithm is used in external memory and compression of the suffix tree?
a) Weiner's algorithm
b) Farach's algorithm
c) Ukkonen's algorithm
d) Alexander Morse
View Answer

Answer: b
Explanation: The concept of Suffix Tree was introduced by Weiner in 1973. Ukkonen provided the first online contribution of the Suffix tree. Farach gave the first suffix tree contribution for all alphabets in 1997. Farach's algorithm is used in external memory and compression.

11. Which statement is correct of suffix tree with a string of length n?
a) The tree has n leaves.
b) The tree has n roots
c) Height of Tree is n
d) Depth of tree is n
View Answer

Answer: a
Explanation: In computer science, a suffix tree is also known as PAT tree or position tree. It is a compressed search tree or prefix tree in which keys contain the suffix of text values as the text position. For a string of length n, the suffix tree has leaves equal to n.

12. Do all the nodes have at least two children in suffix tree.
a) True
b) False
View Answer

Answer: b
Explanation: It is a compressed search tree or prefix tree in which keys contain the suffix of text values as the text position. All the nodes (internal) except for the root nodes have at least two children.

13. Can the two edges that are coming out of a node have labels of string beginning with the same character?
a) True
b) False
View Answer

Answer: b
Explanation: It is a compressed search tree or prefix tree in which keys contain the suffix of text values as the text position. All the nodes (internal) except for the root nodes have at least two children. No two edges that are coming out of a node have labels of string beginning with the same character.

14. Which tree allows fast implementation of a set of string operation?
a) Rope Tree
b) Tango Tree
c) Generalized Suffix Tree
d) Top Tree
View Answer

Answer: c
Explanation: In computer science, the generalized suffix is a special suffix tree which contains a set of strings or set of words instead of a single string like suffix tree. Hence Different operation can be performed on a set of strings using a generalized suffix tree.

15. What is a time complexity for checking a string of length n is substring or not?
a) O (log n!)
b) O (n!)
c) O (n^2)
d) O (n)
View Answer

Answer: d
Explanation: Suffix tree is also known as PAT tree or position tree. It allows fast string operation. Total time taken for construction of suffix tree is linear to the length of the tree. To check if a substring is present in a string of a length of n, the time complexity for such operation is found to be O (n).
Topic 93. Suffix Tree – 2

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Suffix Tree – 2”.

1. What is a time complexity for x pattern occurrence of length n?
a) O (log n!)
b) (n!)
c) O (n^2)
d) (n + x)
View Answer

Answer: d
Explanation: Suffix tree is also known as PAT tree or position tree. It allows fast string operation. To check if a substring is present in a string of a length of n, the time complexity for such operation is found to be O (n). The time complexity for x pattern occurrence of length n is (n + x).

2. What is a time complexity for finding the longest substring that is common in string S1 and S2 (n1 and n2 are the string lengths of strings s1, s2 respectively)?
a) O (log n!)
b) (n!)
c) O (n^2+ n1)
d) (n1 + n2)
View Answer

Answer: d
Explanation: Suffix Tree allows fast string operation. To check if a substring is present in a string of a length of n, the time complexity for such operation is found to be O (n). The time complexity for finding the longest substring that is common in string S1 and S2 is (n1 + n2).

3. What is a time complexity for finding the longest substring that is repeated in a string?
a) O (log n!)
b) (n!)
c) O (n^2+ n1)

d) (n)
View Answer

Answer: d
Explanation: Suffix Tree allows fast string operation. To check if a substring is present in a string of a length of n, the time complexity for such operation is found to be O (n). The time complexity for finding the longest substring that is repeated in a string is (n).

4. What is a time complexity for finding frequently occurring of a substring of minimum length in a string?
a) (n)
b) (n!)
c) O (n^2+ n1)
d) O (log n!)
View Answer

Answer: a
Explanation: Suffix Tree allows fast string operation. To check if a substring is present in a string of a length of n, the time complexity for such operation is found to be O (n). The time complexity for finding frequently occurring of a substring of minimum length in a string is (n).

5. What is a time complexity for finding the longest prefix that is common between suffix in a string?
a) (n)
b) (n!)
c) (1)
d) O (log n!)
View Answer

Answer: c
Explanation: Suffix Tree allows fast string operation. To check if a substring is present in a string of a length of n, the time complexity for such operation is found to be O (n). The time complexity for finding the longest prefix that is common between suffix in a string is (1).

6. What is a time complexity for finding all the maximal palindrome in a string?
a) (n)
b) (n!)
c) (1)
d) O (log n!)
View Answer

Answer: a
Explanation: Palindrome is a string that is the same when reading forward as well as backward. To check if a substring is present in a string of a length of n, the time complexity for such operation is found to be O (n). The time complexity for finding all the maximal palindrome in a string is (n).

7. What is a time complexity for finding all the tandem repeats?
a) (n)
b) (n!)
c) (1)
d) O (n log n + z)
View Answer

Answer: d
Explanation: Tandem Repeats are formed in DNA when the nucleotides pattern repeats more than once. To check if a substring is present in a string of a length of n, the time complexity for such operation is found to be O (n). The time complexity for finding all the tandem repeats in a string is O (n log n + z).

8. What is a time complexity for finding the longest palindromic substring in a string by using the generalized suffix tree?
a) Linear Time
b) Exponential Time
c) Logarithmic Time
d) Cubic Time
View Answer

Answer: a
Explanation: Palindrome is a string that is same when reading forward as well as backward. The time complexity for finding the longest palindromic substring in a string by using generalized suffix tree is linear time.

9. Which of the following algorithm of data compression uses a suffix tree?
a) Weiner’s algorithm
b) Farach’s algorithm

c) Lempel – Ziv – Welch’s algorithm
d) Alexander Morse’s algorithm
View Answer

Answer: c
Explanation: The concept of Suffix Tree was introduced by Weiner in 1973. Ukkonen provided the first online contribution of the Suffix tree. Farach gave the first suffix tree contribution for all alphabets in 1997. Lempel – Ziv – Welch’s algorithm of data compression uses a suffix tree.

10. Which of the following data clustering algorithm uses suffix tree in search engines?
a) Weiner’s algorithm
b) Farach’s algorithm
c) Lempel – Ziv – Welch’s algorithm
d) Suffix Tree Clustering
View Answer

Answer: d
Explanation: The concept of Suffix Tree was introduced by Weiner in 1973. Ukkonen provided the first online contribution of Suffix. Farach gave the first suffix tree contribution for all alphabets in 1997. Suffix Tree Clustering is a data clustering algorithm that uses suffix tree in search engines.

11. What is a time complexity for finding the total length of all string on all edges of a tree?
a) □ (n)
b) □ (n!)
c) □ (1)
d) O (n^2)
View Answer

Answer: d
Explanation: To check if a substring is present in a string of a length of n, the time complexity for such operation is found to be O (n). The time complexity for finding the total length of all string on all edges of a tree is O (n^2).

12. Can suffix tree be used in string problems occurring in a text editor.
a) True
b) False
View Answer

Answer: a
Explanation: It is a compressed search tree or prefix tree in which keys contain the suffix of text values as the text position. So, the suffix tree can be used in string problems occurring in a text editor. The time taken to solve the problem is linear to the length of the string.

13. Can suffix tree be used in bioinformatics problems and solutions.
a) True
b) False
View Answer

Answer: a
Explanation: It is a compressed search tree or prefix tree in which keys contain the suffix of text values as the text position. So, a suffix tree is used in bioinformatics problems and solutions like pattern searching in DNA and protein sequences.

14. For what size of nodes, the worst case of usage of space in suffix tree seen?
a) n Nodes
b) 2^n Nodes
c) 2n nodes
d) n! nodes
View Answer

Answer: c
Explanation: In computer science, the worst case of usage of space in a suffix tree is found to be for a Fibonacci word for a full 2n nodes. The time complexity for usage of space is found to be O (n).

15. What is a time complexity for inserting an alphabet in the tree using hash maps?
a) O (log n!)
b) O (n!)
c) O (n^2)
d) O (1)
View Answer

Answer: d

Explanation: Suffix tree is also known as PAT tree or position tree. It allows fast string operation. Total time taken for construction of suffix tree is linear to the length of the tree. The time complexity for inserting an alphabet in the tree using hash maps is constant, O (1).
Topic 94. Hash Tables

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Hash Tables”.

1. What is a hash table?
a) A structure that maps values to keys
b) A structure that maps keys to values
c) A structure used for storage
d) A structure used to implement stack and queue
View Answer

Answer: b
Explanation: A hash table is used to implement associative arrays which has a key-value pair, so the has table maps keys to values.

2. If several elements are competing for the same bucket in the hash table, what is it called?
a) Diffusion
b) Replication
c) Collision
d) Duplication
View Answer

Answer: c
Explanation: In a hash table, if sevaral elements are computing for the same bucket then there will be a clash among elements. This condition is called Collision. The Collision is reduced by adding elements to a linked list and head address of linked list is placed in hash table.

3. What is direct addressing?
a) Distinct array position for every possible key
b) Fewer array positions than keys
c) Fewer keys than array positions
d) Same array position for all keys
View Answer

Answer: a
Explanation: Direct addressing is possible only when we can afford to allocate an array that has one position for every possible key.

4. What is the search complexity in direct addressing?
a) O(n)
b) O(logn)
c) O(nlogn)
d) O(1)
View Answer

Answer: d
Explanation: Since every key has a unique array position, searching takes a constant time.

5. What is a hash function?
a) A function has allocated memory to keys
b) A function that computes the location of the key in the array
c) A function that creates an array
d) A function that computes the location of the values in the array
View Answer

Answer: b
Explanation: In a hash table, there are fewer array positions than the keys, so the position of the key in the array has to be computed, this is done using the hash function.

6. Which of the following is not a technique to avoid a collision?
a) Make the hash function appear random
b) Use the chaining method
c) Use uniform hashing
d) Increasing hash table size
View Answer

Answer: d

Explanation: On increasing hash table size, space complexity will increase as we need to reallocate the memory size of hash table for every collision. It is not the best technique to avoid a collision. We can avoid collision by making hash function random, chaining method and uniform hashing.

7. What is the load factor?
a) Average array size
b) Average key size
c) Average chain length
d) Average hash table length
View Answer

Answer: c
Explanation: In simple chaining, load factor is the average number of elements stored in a chain, and is given by the ratio of number of elements stored to the number of slots in the array.

8. What is simple uniform hashing?
a) Every element has equal probability of hashing into any of the slots
b) A weighted probabilistic method is used to hash elements into the slots
c) Elements has Random probability of hashing into array slots
d) Elements are hashed based on priority
View Answer

Answer: a
Explanation: In simple uniform hashing, any given element is equally likely to hash into any of the slots available in the array.

9. In simple uniform hashing, what is the search complexity?
a) O(n)
b) O(logn)
c) O(nlogn)
d) O(1)
View Answer

Answer: d
Explanation: There are two cases, once when the search is successful and when it is unsuccessful, but in both the cases, the complexity is O(1+alpha) where 1 is to compute the hash function and alpha is the load factor.

10. In simple chaining, what data structure is appropriate?
a) Singly linked list
b) Doubly linked list
c) Circular linked list
d) Binary trees
View Answer

Answer: b
Explanation: Deletion becomes easier with doubly linked list, hence it is appropriate.
Topic 95. Hash Tables Chaining using Linked Lists

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Hash Tables Chaining using Linked Lists”.

1. The case in which a key other than the desired one is kept at the identified location is called?
a) Hashing
b) Collision
c) Chaining
d) Open addressing
View Answer

Answer: b
Explanation: When some other value is placed at a specified location other than the desired key, it is said to be a collision.

2. What data organization method is used in hash tables?
a) Stack
b) Array
c) Linked list
d) Queue
View Answer

Answer: c
Explanation: The data structure used to organize data for hash tables is linked list. It contains a data field and a pointer field.

3. The task of generating alternative indices for a node is called?
a) Collision handling
b) Collision detection
c) Collision recovery
d) Closed hashing
View Answer

Answer: a
Explanation: Collision handling involves the process of formulating alternative indices for a key.

4. Which of the following is not a collision resolution technique?
a) Separate chaining
b) Linear probing
c) Quadratic probing
d) Hashing
View Answer

Answer: d
Explanation: Hashing is a technique of placing data items in specific locations. Collision may occur in hashing but hashing is not a collision resolution technique.

5. Hashing is the problem of finding an appropriate mapping of keys into addresses.
a) True
b) False
View Answer

Answer: a
Explanation: Hashing is a data structure which is used to locate data in a table based on a key value.

6. In a hash table of size 10, where is element 7 placed?
a) 6
b) 7
c) 17
d) 16
View Answer

Answer: b
Explanation: The hash location is defined as hash(f)= key mod table_size.
7 mod 10 gives 7. It is placed in 7th position.

7. What should be the load factor for separate chaining hashing?
a) 0.5
b) 1
c) 1.5
d) 2
View Answer

Answer: b
Explanation: For hashing using separate chaining method, the load factor should be maintained as 1. For open addressing method, it should not exceed 0.5.

8. Which of the following operations are done in a hash table?
a) Insert only
b) Search only
c) Insert and search
d) Replace
View Answer

Answer: c
Explanation: Hash tables are used to implement Insert and Find operations in constant average time. This is the prime purpose of hashing.

9. Which of the following is identical to that of a separate chaining hash node?
a) Linked list
b) Array
c) Stack
d) Queue
View Answer

Answer: a

Explanation: The hash node in separate chaining is similar to that of a linked list. The separate chaining hash table is an array of linked lists.

10. Which of the following is the hashing function for separate chaining?
a) H(x)=(hash(x)+f(i)) mod table size
b) H(x)=hash(x)+i^2 mod table size
c) H(x)=x mod table size
d) H(x)=x mod (table size * 2)
View Answer

Answer: c
Explanation: The hashing function for separate chaining is given by H(x)= key mod table size. H(x)=hash(x)+i2 mod table size defines quadratic probing.

11. What is the correct notation for a load factor?
a) Ω
b) ∞
c) ∑
d) □
View Answer

Answer: d
Explanation: In general, load factor is denoted as □. In separate chaining method, load factor is maintained as 1.0.

12. In hash tables, how many traversal of links does a successful search require?
a) 1+□
b) 1+□^{2}
c) 1+ (□/2)
d) □^{3}
View Answer

Answer: c
Explanation: A successful search requires about 1+ (□/2) links to be traversed. There is a guarantee that at least one link must be traversed.

13. Which of the following is a disadvantage of using separate chaining using linked lists?
a) It requires many pointers
b) It requires linked lists
c) It uses array
d) It does not resolve collision
View Answer

Answer: a
Explanation: One of the major disadvantages of using separate chaining is the requirement of pointers. If the number of elements are more, it requires more pointers.

14. What is the worst case search time of a hashing using separate chaining algorithm?
a) O(N log N)
b) O(N)
c) O(N^2)
d) O(N^3)
View Answer

Answer: b
Explanation: The worst case search time of separate chaining algorithm using linked lists is mathematically found to be O(N).

15. From the given table, find '?'.
Given: hash(x)= x mod 10

a) 13
b) 16
c) 12
d) 14
View Answer

Answer: c
Explanation: From the given options, 12 mod 10 hashes to 2 and hence '?' = 12.
Topic 96. Hash Tables Chaining using Doubly Linked Lists

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Hash Tables Chaining using Doubly Linked Lists”.

1. Which of the following is used in hash tables to determine the index of any input record?
a) hash function
b) hash linked list
c) hash tree
d) hash chaining
View Answer

Answer: a
Explanation: Hash table is an example of a data structure that is built for fast access of elements. Hash functions are used to determine the index of any input record in a hash table.

2. What is the advantage of a hash table as a data structure?
a) faster access of data
b) easy to implement
c) very efficient for less number of entries
d) exhibit good locality of reference
View Answer

Answer: a
Explanation: Hash table is a data structure that has an advantage that it allows fast access of elements. Hash functions are used to determine the index of any input record in a hash table.

3. What is the use of a hash function?
a) to calculate and return the index of corresponding data
b) to store data
c) to erase data
d) to change data
View Answer

Answer: a
Explanation: Hash function calculates and returns the index for corresponding data. This data is then mapped in a hash table.

4. What is the time complexity of insert function in a hash table using a doubly linked list?
a) O(1)
b) O(n)
c) O(log n)
d) O(n log n)
View Answer

Answer: a
Explanation: Time complexity of insert function in a hash table is O(1). Condition is that the number of collisions should be low.

5. What is the time complexity of search function in a hash table using a doubly linked list?
a) O(1)
b) O(n)
c) O(log n)
d) O(n log n)
View Answer

Answer: a
Explanation: Time complexity of search function in a hash table is O(1). Condition is that the number of collisions should be low.

6. What is the time complexity of delete function in the hash table using a doubly linked list?
a) O(1)
b) O(n)
c) O(log n)
d) O(n log n)
View Answer

Answer: a
Explanation: Time complexity of delete function in a hash table is O(1). Condition is that the hash function should be such that the number of collisions should be low.

7. Hashing can be used to encrypt and decrypt digital signatures.

a) true
b) false
View Answer

Answer: a
Explanation: Hashing is also used in encryption algorithms. It is used for encryption and decryption of digital signatures.

8. What is the advantage of using a doubly linked list for chaining over singly linked list?
a) it takes less memory
b) it is easy to implement
c) it makes the process of insertion and deletion faster
d) it causes less collisions
View Answer

Answer: c
Explanation: Using a doubly linked list reduces time complexity significantly. Though it uses more memory to store the extra pointer.

9. Which of the following technique stores data in the hash table itself in case of a collision?
a) Open addressing
b) Chaining using linked list
c) Chaining using doubly linked list
d) Chaining using binary tree
View Answer

Answer: a
Explanation: Open addressing is used to store data in the table itself in case of a collision. Whereas chaining stores data in a separate entity.

10. Which of the following technique stores data in a separate entity in case of a collision?
a) Open addressing
b) Chaining using doubly linked list
c) Linear probing
d) Double hashing
View Answer

Answer: b
Explanation: Chaining using doubly linked list is used to store data in a separate entity (doubly linked list in this case) in case of a collision. Whereas open addressing stores it in the table itself.

11. Collision is caused due to the presence of two keys having the same value.
a) True
b) False
View Answer

Answer: a
Explanation: A collision is caused due to the presence of two keys having the same value. It is handled by using any one of the two methods namely:- Chaining and Open addressing.
Topic 97. Hash Tables Chaining with Binary Trees

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Hash Tables Chaining with Binary Trees”.

1. Which of the following variant of a hash table has the best cache performance?
a) hash table using a linked list for separate chaining
b) hash table using binary search tree for separate chaining
c) hash table using open addressing
d) hash table using a doubly linked list for separate chaining
View Answer

Answer: c
Explanation: Implementation of the hash table using open addressing has a better cache performance as compared to separate chaining. It is because open addressing stores data in the same table without using any extra space.

2. What is the advantage of hashing with chaining?
a) cache performance is good
b) uses less space
c) less sensitive to hash function
d) has a time complexity of O(n) in the worst case
View Answer

Answer: c
Explanation: Hashing with separate chaining has an advantage that it is less sensitive to a hash function. It is also easy to implement.

3. What is the disadvantage of hashing with chaining?
a) not easy to implement
b) takes more space
c) quite sensitive to hash function
d) table gets filled up easily
View Answer

Answer: b
Explanation: Hashing with separate chaining has a disadvantage that it takes more space. This space is used for storing elements in case of a collision.

4. What is the time complexity of insert function in a hash table using a binary tree?
a) O(1)
b) O(n)
c) O(log n)
d) O(n log n)
View Answer

Answer: a
Explanation: Time complexity of insert function in a hash table is O(1) on an average. Condition is that the number of collisions should be low.

5. What is the time complexity of the search function in a hash table using a binary tree?
a) O(1)
b) O(n)
c) O(log n)
d) O(n log n)
View Answer

Answer: a
Explanation: Time complexity of search function in a hash table is O(1). Condition is that the number of collisions should be low.

6. What is the time complexity of the delete function in the hash table using a binary tree?
a) O(1)
b) O(n)
c) O(log n)
d) O(n log n)
View Answer

Answer: a
Explanation: Time complexity of delete function in a hash table is O(1). Condition is that the hash function should be such that the number of collisions should be low.

7. What is the advantage of a hash table over BST?
a) hash table has a better average time complexity for performing insert, delete and search operations
b) hash table requires less space
c) range query is easy with hash table
d) easier to implement
View Answer

Answer: a
Explanation: Hash table and BST both are examples of data structures. Hash table has an advantage that it has a better time complexity for performing insert, delete and search operations.

8. What is the disadvantage of BST over the hash table?
a) BST is easier to implement
b) BST can get the keys sorted by just performing inorder traversal
c) BST can perform range query easily
d) Time complexity of hash table in inserting, searching and deleting is less than that of BST
View Answer

Answer: d
Explanation: BST has an advantage that it is easier to implement, can get the keys sorted by just performing in-order traversal and can perform range query easily. Hash table has lesser time complexity for performing insert, delete and search operations.

9. Which of the following technique stores data separately in case of a collision?
a) Open addressing
b) Double hashing
c) Quadratic probing
d) Chaining using a binary tree
View Answer

Answer: d
Explanation: Open addressing is used to store data in the table itself in case of a collision. Whereas chaining stores data in a separate entity.

10. Separate chaining is easier to implement as compared to open addressing.
a) true
b) false
View Answer

Answer: a
Explanation: There are two methods of handling collisions in a hash table:- open addressing and separate chaining. Open addressing requires more computation as compared to separate chaining.

11. In open addressing the hash table can never become full.
a) True
b) False
View Answer

Answer: b
Explanation: There are two methods of handling collisions in a hash table:- open addressing and separate chaining. In open addressing the hash table can become full.
Topic 98. Hash Tables Chaining with List Heads

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Hash Tables Chaining with List Heads”.

1. Which of the following helps keys to be mapped into addresses?
a) hash function
b) separate chaining
c) open addressing
d) chaining using a linked list
View Answer

Answer: a
Explanation: Hash table is an example of a data structure that is built for fast access of elements. Hash functions are used to determine the index of any input record in a hash table.

2. What is the advantage of the hash table over a linked list?
a) faster access of data
b) easy to implement
c) very efficient for less number of entries
d) exhibit good locality of reference
View Answer

Answer: a
Explanation: Hash table is a data structure that has an advantage that it allows fast access of elements. But linked list is easier to implement as compared to the hash table.

3. Which of the following trait of a hash function is most desirable?
a) it should cause less collisions
b) it should cause more collisions
c) it should occupy less space
d) it should be easy to implement
View Answer

Answer: a
Explanation: Hash function calculates and returns the index for corresponding data. So the most important trait of a hash function is that it should cause a minimum number of collisions.

4. What is the time complexity of insert function in a hash table using list head?
a) O(1)
b) O(n)
c) O(log n)

d) O(n log n)
View Answer

Answer: a
Explanation: Time complexity of insert function in a hash table is O(1). Condition is that the number of collisions should be low.

5. What is the time complexity of search function in a hash table using list head?
a) O(1)
b) O(n)
c) O(log n)
d) O(n log n)
View Answer

Answer: a
Explanation: Time complexity of search function in a hash table is O(1). Condition is that the number of collisions should be low.

6. What is the time complexity of delete function in the hash table using list head?
a) O(1)
b) O(n)
c) O(log n)
d) O(n log n)
View Answer

Answer: a
Explanation: Time complexity of delete function in a hash table is O(1). Condition is that the hash function should be such that the number of collisions should be low.

7. A hash table may become full in the case when we use open addressing.
a) true
b) false
View Answer

Answer: a
Explanation: A hash table may become full in the case when we use open addressing. But when we use separate chaining it does not happen.

8. What is the advantage of using linked list over the doubly linked list for chaining?
a) it takes less memory
b) it causes more collisions
c) it makes the process of insertion and deletion faster
d) it causes less collisions
View Answer

Answer: a
Explanation: Singly linked list takes lesser space as compared to doubly linked list. But the time complexity of the singly linked list is more than a doubly linked list.

9. What is the worst case time complexity of insert function in the hash table when the list head is used for chaining?
a) O(1)
b) O(n log n)
c) O(log n)
d) O(n)
View Answer

Answer: d
Explanation: Worst case time complexity of insert function in the hash table when the list head is used for chaining is O(n). It is caused when a number of collisions are very high.

10. Which of the following technique is used for handling collisions in a hash table?
a) Open addressing
b) Hashing
c) Searching
d) Hash function
View Answer

Answer: a
Explanation: Open addressing is the technique which is used for handling collisions in a hash table. Separate chaining is another technique which is used for the same purpose.

11. By implementing separate chaining using list head we can reduce the number of collisions drastically.
a) True
b) False
View Answer

Answer: b
Explanation: Collision is caused when a hash function returns repeated values. So collisions can be reduced by developing a better hash function. Whereas separate chaining using list head is a collision handling technique so it has no relation with a number of collisions taking place.

12. Which of the following is an advantage of open addressing over separate chaining?
a) it is simpler to implement
b) table never gets full
c) it is less sensitive to hash function
d) it has better cache performance
View Answer

Answer: a
Explanation: Open addressing is the technique which is used for handling collisions in a hash table. It has a better cache performance as everything is stored in the same table.
Topic 99. Hash Tables with Linear Probing

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Hash Tables with Linear Probing”.

1. Which of the following problems occur due to linear probing?
a) Primary collision
b) Secondary collision
c) Separate chaining
d) Extendible hashing
View Answer

Answer: a
Explanation: Primary collision occurs due to linear probing technique. It is overcome using a quadratic probing technique.

2. How many probes are required on average for insertion and successful search?
a) 4 and 10
b) 2 and 6
c) 2.5 and 1.5
d) 3.5 and 1.5
View Answer

Answer: c
Explanation: Using formula, the average number of probes required for insertion is 2.5 and for a successful search, it is 1.5.

3. What is the load factor for an open addressing technique?
a) 1
b) 0.5
c) 1.5
d) 0
View Answer

Answer: b
Explanation: The load factor for an open addressing technique should be 0.5. For separate chaining technique, the load factor is 1.

4. Which of the following is not a collision resolution strategy for open addressing?
a) Linear probing
b) Quadratic probing
c) Double hashing
d) Rehashing
View Answer

Answer: d
Explanation: Linear probing, quadratic probing and double hashing are all collision resolution strategies for open addressing whereas rehashing is a different technique.

5. In linear probing, the cost of an unsuccessful search can be used to compute the average cost of a successful search.
a) True
b) False

View Answer

Answer: a
Explanation: Using random collision resolution algorithm, the cost of an unsuccessful search can be used to compute the average cost of a successful search.

6. Which of the following is the correct function definition for linear probing?
a) F(i)= 1
b) F(i)=i
c) F(i)=i^2
d) F(i)=i+1
View Answer

Answer: b
Explanation: The function used in linear probing is defined as, F(i)=I where i=0,1,2,3….,n.

7. ___________ is not a theoretical problem but actually occurs in real implementations of probing.
a) Hashing
b) Clustering
c) Rehashing
d) Collision
View Answer

Answer: b
Explanation: Clustering is not a theoretical problem but it occurs in implementations of hashing. Rehashing is a kind of hashing.

8. What is the hash function used in linear probing?
a) H(x)= key mod table size
b) H(x)= (key+ F(i^2)) mod table size
c) H(x)= (key+ F(i)) mod table size
d) H(x)= X mod 17
View Answer

Answer: c
Explanation: The hash function used in linear probing is defined to be H(x)= (key+ F(i)) mod table size where i=0,1,2,3,…,n.

9. Hashing can be used in online spelling checkers.
a) True
b) False
View Answer

Answer: a
Explanation: If misspelling detection is important, an entire dictionary can be pre-hashed and words can be checked in constant time.

10. In the following given hash table, use linear probing to find the location of 49.

0	
1	
2	
3	
4	
5	
6	
7	
8	18
9	89

a) 7
b) 6
c) 2
d) 0
View Answer

Answer: d
Explanation: Initially, collision occurs while hashing 49 for the first time.
Hence, after setting f(i)=1, the hashed location is found to be 0.

11. What is the formula to find the expected number of probes for an unsuccessful search in linear probing?
a) \(\frac{1}{2} \frac{1+1}{(1-□)}\)
b) \(\frac{1}{2}\frac{1+1}{(1-□)^2}\)
c) \(\frac{1}{2}\frac{1+1}{(1+□)}\)
d) \(\frac{1}{2}\frac{1+1}{(1+□)(1-□)}\)
View Answer

Answer: b
Explanation: The mathematical formula for calculating the number of probes for an unsuccessful search is \(\frac{1}{2}\frac{1+1}{(1-□)^2}\). For insertion, it is \(\frac{1}{2} \frac{1+1}{(1-□)}\).
Topic 100. Hash Tables with Quadratic Probing

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Hash Tables with Quadratic Probing”.

1. Which of the following schemes does quadratic probing come under?
a) rehashing
b) extended hashing
c) separate chaining
d) open addressing
View Answer

Answer: d
Explanation: Quadratic probing comes under open addressing scheme to resolve collisions in hash tables.

2. Quadratic probing overcomes primary collision.
a) True
b) False
View Answer

Answer: a
Explanation: Quadratic probing can overcome primary collision that occurs in linear probing but a secondary collision occurs in quadratic probing.

3. What kind of deletion is implemented by hashing using open addressing?
a) active deletion
b) standard deletion
c) lazy deletion
d) no deletion
View Answer

Answer: c
Explanation: Standard deletion cannot be performed in an open addressing hash table, because the cells might have caused collision. Hence, the hash tables implement lazy deletion.

4. In quadratic probing, if the table size is prime, a new element cannot be inserted if the table is half full.
a) True
b) False
View Answer

Answer: b
Explanation: In quadratic probing, if the table size is prime, we can insert a new element even though table is exactly half filled. We can’t insert element if table size is more than half filled.

5. Which of the following is the correct function definition for quadratic probing?
a) $F(i)=i^2$
b) $F(i)=i$
c) $F(i)=i+1$
d) $F(i)=i^2+1$
View Answer

Answer: a
Explanation: The function of quadratic probing is defined as $F(i)=i^2$. The function of linear probing is defined as $F(i)=i$.

6. How many constraints are to be met to successfully implement quadratic probing?
a) 1
b) 2
c) 3
d) 4
View Answer

Answer: b
Explanation: 2 requirements are to be met with respect to table size. The table size should be a prime number and the table size should not be more than half full.

7. Which among the following is the best technique to handle collision?
a) Quadratic probing
b) Linear probing
c) Double hashing
d) Separate chaining
View Answer

Answer: a
Explanation: Quadratic probing handles primary collision occurring in the linear probing method. Although secondary collision occurs in quadratic probing, it can be removed by extra multiplications and divisions.

8. Which of the following techniques offer better cache performance?
a) Quadratic probing
b) Linear probing
c) Double hashing
d) Rehashing
View Answer

Answer: b
Explanation: Linear probing offers better cache performance than quadratic probing and also it preserves locality of reference.

9. What is the formula used in quadratic probing?
a) Hash key = key mod table size
b) Hash key=(hash(x)+F(i)) mod table size
c) Hash key=(hash(x)+F(i^2)) mod table size
d) H(x) = x mod 17
View Answer

Answer: c
Explanation: Hash key=(hash(x)+F(i^2)) mod table size is the formula for quadratic probing. Hash key = (hash(x)+F(i)) mod table size is the formula for linear probing.

10. For the given hash table, in what location will the element 58 be hashed using quadratic probing?

0	49
1	
2	
3	
4	
5	
6	
7	
8	18
9	89

a) 1
b) 2
c) 7
d) 6
View Answer

Answer: b
Explanation: Initially, 58 collides at position 8. Another collision occurs one cell away. Hence, F(i2)=4. Using quadratic probing formula, the location is obtained as 2.
Topic 101. Hashing Functions

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Hashing Functions”.

1. Which scheme uses a randomization approach?
a) hashing by division
b) hashing by multiplication
c) universal hashing
d) open addressing

View Answer

Answer: c
Explanation: Universal hashing scheme uses a randomization approach whereas hashing by division and hashing by multiplication are heuristic in nature.

2. Which hash function satisfies the condition of simple uniform hashing?
a) h(k) = lowerbound(km)
b) h(k)= upperbound(mk)
c) h(k)= lowerbound(k)
d) h(k)= upperbound(k)
View Answer

Answer: a
Explanation: If the keys are known to be random real numbers k independently and uniformly distributed in the range 0<=k<=1, the hash function which satisfies the condition of simple uniform hashing is
h(k)= lowerbound(km).

3. A good hash approach is to derive the hash value that is expected to be dependent of any patterns that might exist in the data.
a) True
b) False
View Answer

Answer: b
Explanation: A hash value is expected to be unrelated or independent of any patterns in the distribution of keys.

4. Interpret the given character string as an integer expressed in suitable radix notation.

```
Character string = pt
```

a) 14963
b) 14392
c) 12784
d) 14452
View Answer

Answer: d
Explanation: The given character string can be interpreted as (112,116) (Ascii values) then expressed as a radix-128 integer, hence the value is 112*128 + 116 = 14452.

5. What is the hash function used in the division method?
a) h(k) = k/m
b) h(k) = k mod m
c) h(k) = m/k
d) h(k) = m mod k
View Answer

Answer: b
Explanation: In division method for creating hash functions, k keys are mapped into one of m slots by taking the reminder of k divided by m.

6. What can be the value of m in the division method?
a) Any prime number
b) Any even number
c) $2^p – 1$
d) 2^p
View Answer

Answer: a
Explanation: A prime number not too close to an exact power of 2 is often a good choice for m since it reduces the number of collisions which are likely to occur.

7. Which scheme provides good performance?
a) open addressing
b) universal hashing
c) hashing by division
d) hashing by multiplication
View Answer

Answer: b

Explanation: Universal hashing scheme provides better performance than other schemes because it uses a unique randomisation approach.

8. Using division method, in a given hash table of size 157, the key of value 172 be placed at position ____
a) 19
b) 72
c) 15
d) 17
View Answer

Answer: c
Explanation: The key 172 can be placed at position 15 by using the formula
H(k) = k mod m
H(k) = 172 mod 157
H(k) = 15.

9. How many steps are involved in creating a hash function using a multiplication method?
a) 1
b) 4
c) 3
d) 2
View Answer

Answer: d
Explanation: In multiplication method 2 steps are involved. First multiplying the key value by a constant. Then multiplying this value by m.

10. What is the hash function used in multiplication method?
a) h(k) = floor(m(kA mod 1))
b) h(k) = ceil(m(kA mod 1))
c) h(k) = floor(kA mod m)
d) h(k) = ceil(kA mod m)
View Answer

Answer: a
Explanation: The hash function can be computed by multiplying m with the fractional part of kA (kA mod 1) and then computing the floor value of the result.

11. What is the advantage of the multiplication method?
a) only 2 steps are involved
b) using constant
c) value of m not critical
d) simple multiplication
View Answer

Answer: c
Explanation: The value of m can be simply in powers of 2 since we can easily implement the function in most computers.
$m=2^p$ where p is an integer.

12. What is the table size when the value of p is 7 in multiplication method of creating hash functions?
a) 14
b) 128
c) 49
d) 127
View Answer

Answer: b
Explanation: In multiplication method of creating hash functions the table size can be taken in integral powers of 2.
$m = 2^p$
$m= 2^7$
m = 128.

13. What is the value of h(k) for the key 123456?
Given: p=14, s=2654435769, w=32
a) 123
b) 456
c) 70
d) 67
View Answer

Answer: d
Explanation: $A = s/2^w$
$A = 2654435769/ 2^{32}$
$k.A = 123456 * (2654435769/ 2^{32})$
$= (76300 * 2^{32}) + 17612864$
Hence r1= 76300; r0=17612864
Since w=14 the 14 most significant bits of r0 yield the value of h(k) as 67.

14. What is the average retrieval time when n keys hash to the same slot?
a) Theta(n)
b) Theta(n^2)
c) Theta(nlog n)
d) Big-Oh(n^2)
View Answer

Answer: a
Explanation: The average retrieval time when n keys hash to the same slot is given by Theta(n) as the collision occurs in the hash table.

15. Collisions can be reduced by choosing a hash function randomly in a way that is independent of the keys that are actually to be stored.
a) True
b) False
View Answer

Answer: a
Explanation: Because of randomization, the algorithm can behave differently on each execution, providing good average case performance for any input.
Topic 102. Double Hashing

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Double Hashing”.

1. Double hashing is one of the best methods available for open addressing.
a) True
b) False
View Answer

Answer: a
Explanation: Double hashing is one of the best methods for open addressing because the permutations produced have many characteristics of randomly chosen permutations.

2. What is the hash function used in Double Hashing?
a) (h1(k) – i*h2(k))mod m
b) h1(k) + h2(k)
c) (h1(k) + i*h2(k))mod m
d) (h1(k) + h2(k))mod m
View Answer

Answer: c
Explanation: Double hashing uses a hash function of the form (h1(k) + i*h2(k))mod m where h1 and h2 are auxiliary hash functions and m is the size of the hash table.

3. On what value does the probe sequence depend on?
a) c1
b) k
c) c2
d) m
View Answer

Answer: b
Explanation: The probe sequence depends in upon the key k since the initial probe position, the offset or both may vary.

4. The value of h2(k) can be composite relatively to the hash table size m.
a) True
b) False
View Answer

Answer: b
Explanation: The value h2(k) must be relatively prime to the hash table size m for the entire hash table to be searched. It

can be ensured by having m in powers of 2 and designing h2 so that it produces an odd number.

5. What are the values of h1(k) and h2(k) in the hash function?
a)

h1(k) = m mod k
h2(k) = 1+ (m' mod k)

b)

h1(k) = 1 + (m mod k)
h2(k) = m' mod k

c)

h1(k) = 1+ (k mod m)
h2(k) = k mod m

d)

h1(k) = k mod m
h2(k) = 1+ (k mod m')

View Answer
Answer: d
Explanation: The values h1(k) and h2(k) are k mod m and 1+(k mod m') respectively where m is a prime number and m' is chosen slightly less than m. (m'=m-1).

6. What is the running time of double hashing?
a) Theta(m)
b) Theta(m^2)
c) Theta(m log k)
d) Theta(m^3)
View Answer
Answer: a
Explanation: The running time of double hashing is Theta(m) since each possible (h1(k), h2(k)) pair yields a distinct probe sequence. Hence the performance of double hashing is better.

7. Which technique has the greatest number of probe sequences?
a) Linear probing
b) Quadratic probing
c) Double hashing
d) Closed hashing
View Answer

Answer: c
Explanation: Double hashing has the greatest number of probe sequences thereby efficiently resolves hash collision problems.

8. At what position the number 72 gets inserted in the following table?

Index	Key
0	22
1	34
2	
3	
4	
5	56
6	
7	18
8	41
9	
10	

a) 3
b) 10
c) 4
d) 6

View Answer

Answer: d
Explanation: The number 72 must be inserted at index 6.
H(k)=k mod m
H(72)= 72 mod 11
H(72)= 6.

9. Where does the number 14 get inserted in the following table?

Index	Key
0	
1	79
2	
3	
4	69
5	98
6	
7	72
8	
9	
10	
11	50
12	

a) 2
b) 9
c) 4
d) 8
View Answer

Answer: b
Explanation: Here the hash table is of size 13 with h1(k) = k mod 13 and h2(k) = 1 + k mod 11. Since 14 = 1 (mod 13) and 14 = 3 (mod 11), the key 14 is inserted into empty slot 9.

10. What is the value of m' if the value of m is 19?
a) 11
b) 18
c) 17
d) 15
View Answer

Answer: c
Explanation: The value of m' is chosen as a prime number slightly less than the value of m. Here the value of m is 19, hence the value of m' can be chosen as 17.
Topic 103. Hash Tree

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on "Hash Tree".

1. Hash tree is generalization of ______
a) Heap
b) Hash list
c) BST
d) B – tree
View Answer

Answer: b
Explanation: Hash list is the list of hashes of the blocks in a set file. Hash tree is a generalization of the hash list in which leaves are labeled with the hash of a data block and every non-leaf node is hash of the labels of its children.

2. Hash tree is used in effective data verification in distributed systems.
a) True
b) False
View Answer

Answer: a
Explanation: Hash trees are used in distributed systems for efficient data verification. Hash tree used hashes instead of the

full files, hence they are efficient. Because Hashes are ways of encoding files that are much smaller than the actual file itself.

3. Which of the following is a widely used form of the hash tree?
a) B+ – tree
b) T tree
c) Tiger tree hash
d) Htree
View Answer

Answer: c
Explanation: The general form the hash tree which is used widely is the Tiger tree hash. It uses a binary hash tree, usually has a data block size of 1024 bytes and uses the Tiger hash.

4. Which of the following is true for a Hash tree?
a) Hashing is used for sequential access
b) Indexing is used for direct access
c) Hash tree allows only sequential access
d) Hashing is used for direct access
View Answer

Answer: d
Explanation: Hash tree allows direct as well as sequential access of the records. Hashing is used for direct access and indexing is generally used for the sequential access.

5. Hash tree is also known as _____
a) Merkle tree
b) T -tree
c) Hash table
d) B^x-tree
View Answer

Answer: a
Explanation: Hash tree is generally known as Merkle tree after Ralph Merkle who patented it in 1979. Typically Merkle trees have a branching factor of 2, meaning that each node has up to 2 children.

6. What will be the height of the hash tree with branching factor 2 and with 8 records?
a) 3
b) 5
c) 4
d) 6
View Answer

Answer: c
Explanation: Consider 8 records A B C D E F G H. These records are stored in Hash tree in as shown in figure below.

7. Where is the hash tree used?
a) in digital currency
b) in sorting of large data
c) for indexing in databases
d) in encryption of data
View Answer

Answer: a
Explanation: Using Hash tree the data verification, data synchronisation and the consistency verification can be done efficiently. So, the hash tree are digital currencies to organise the transactions.

8. What is the worst case time complexity of the insertion in the hash tree?
a) $O(\log_k(n))$
b) $O(n^2)$
c) $O(n\log_k(n))$
d) $O(kn)$
View Answer

Answer: a
Explanation: To insert a record in the hash tree the key is compressed and hashed to get the slot for the entry. So, a hash tree with branching factor k takes $O(\log_k(n))$ for insertion in worst case.

9. Sequential access in a Hash tree is faster than in B-trees.
a) True

b) False
View Answer

Answer: a
Explanation: The sequential access in the hash tree is more efficient and faster than in B-tree. Because while constructing the hash tree in the expansions and contractions of the file is an estimated.

10. Hash tree is used in data synchronisation. In the worst case the data synchronisation takes ______ time.
a) O(logn)
b) O(n^2)
c) O(nlogn)
d) O(n)
View Answer

Answer: d
Explanation: In average scenarios, the synchronisation takes O(logn) because it is based on the traversal and searching. The worst case occurs when there are no nodes in common, so the synchronisation takes O(n) time.
Topic 104. Min Hash

This set of Data Structures & Algorithms Multiple Choice Questions & Answers (MCQs) focuses on “Min Hash”.

1. Which technique is used for finding similarity between two sets?
a) MinHash
b) Stack
c) Priority Queue
d) PAT Tree
View Answer

Answer: a
Explanation: In computer science as well as data mining, to find the similarity between two given sets, a technique called MinHash or min-wise independent permutation scheme is used. It helps in the quick estimation of the similarity between two sets.

2. Who invented the MinHash technique?
a) Weiner
b) Samuel F. B. Morse
c) Friedrich Clemens Gerke
d) Andrei Broder
View Answer

Answer: d
Explanation: In computer science as well as data mining, to find the similarity between two given sets, a technique called MinHash or min-wise independent permutation scheme is used. It helps in the quick estimation of the similarity between two sets. It was invented by Andrei Broder in 1997.

3. Which technique was firstly used to remove duplicate web pages from search results in AltaVista search engine?
a) MinHash
b) Stack
c) Priority Queue
d) PAT Tree
View Answer

Answer: a
Explanation: In computer science as well as data mining, to find the similarity between two given sets, a technique called MinHash or min-wise independent permutation scheme is used. It helps in the quick estimation of the similarity between two sets. It is used in removing duplicate web pages from search results in AltaVista search engine.

4. Which technique was firstly used clustering documents using the similarity of two words or strings?
a) MinHash
b) Stack
c) Priority Queue
d) PAT Tree
View Answer

Answer: a
Explanation: In computer science as well as data mining, to find the similarity between two given sets, a technique called MinHash or min-wise independent permutation scheme is used. It helps in the quick estimation of similarity between two sets. It is used in clustering documents using the similarity of two words or strings.

5. Which indicator is used for similarity between two sets?

a) Rope Tree
b) Jaccard Coefficient
c) Tango Tree
d) MinHash Coefficient
View Answer

Answer: b
Explanation: In computer science as well as data mining, to find the similarity between two given sets, a technique called MinHash or min-wise independent permutation scheme is used. It helps in the quick estimation of similarity between two sets. Jaccard Coefficient is used for similarity between two sets.

6. Which of the following is defined as the ratio of total elements of intersection and union of two sets?
a) Rope Tree
b) Jaccard Coefficient Index
c) Tango Tree
d) MinHash Coefficient
View Answer

Answer: b
Explanation: MinHash helps in the quick estimation of similarity between two sets. Jaccard Coefficient is used for similarity between two sets. Jaccard Coefficient Index is defined as the ratio of total elements of intersection and union of two sets.

7. What is the value of the Jaccard index when the two sets are disjoint?
a) 1
b) 2
c) 3
d) 0
View Answer

Answer: d
Explanation: MinHash helps in the quick estimation of similarity between two sets. Jaccard Coefficient is used for the similarity between two sets. Jaccard Coefficient Index is defined as the ratio of total elements of intersection and union of two sets. For two disjoint sets, the value of the Jaccard index is zero.

8. When are the members of two sets more common relatively?
a) Jaccard Index is Closer to 1
b) Jaccard Index is Closer to 0
c) Jaccard Index is Closer to -1
d) Jaccard Index is Farther to 1
View Answer

Answer: a
Explanation: Jaccard Coefficient Index is defined as the ratio of total elements of intersection and union of two sets. For two disjoint sets, the value of the Jaccard index is zero. The members of two set more common relatively when the Jaccard Index is Closer to 1.

9. What is the expected error for estimating the Jaccard index using MinHash scheme for k different hash functions?
a) O (log k!)
b) O (k!)
c) O (k^2)
d) O (1/k½)
View Answer

Answer: d
Explanation: Jaccard Coefficient Index is defined as the ratio of total elements of intersection and union of two sets. For two disjoint sets, the value of the Jaccard index is zero. The expected error for estimating the Jaccard index using MinHash scheme for k different hash functions is O (1/k½).

10. How many hashes will be needed for calculating Jaccard index with an expected error less than or equal to 0.05?
a) 100
b) 200
c) 300
d) 400
View Answer

Answer: d
Explanation: The expected error for estimating the Jaccard index using MinHash scheme for k different hash functions is O (1/k½). 400 hashes will be needed for calculating Jaccard index with an expected error less than or equal to 0.05.

11. What is the expected error by the estimator Chernoff bound on the samples performed without replacement?

a) O (log k!)
b) O (k!)
c) O (k^2)
d) O (1/k½)
View Answer

Answer: d
Explanation: The expected error for estimating the Jaccard index using MinHash scheme for k different hash functions is O (1/k½). The expected error by the estimator Chernoff bound on the samples performed without replacement is O (1/k½).

12. What is the time required for single variant hashing to maintain the minimum hash queue?
a) O (log n!)
b) O (n!)
c) O (n^2)
d) O (n)
View Answer

Answer: d
Explanation: The expected error for estimating the Jaccard index using MinHash scheme for k different hash functions is O (1/k½). The time required for single variant hashing to maintain the minimum hash queue is O (n).

13. How many bits are needed to specify the single permutation by min-wise independent family?
a) O (log n!)
b) O (n!)
c) Ω (n^2)
d) Ω (n)
View Answer

Answer: d
Explanation: The time required for single variant hashing to maintain the minimum hash queue is O (n). Ω (n) bits are needed to specify the single permutation by min-wise independent family.

14. Is MinHash used as a tool for association rule learning.
a) True
b) False
View Answer

Answer: a
Explanation: MinHash was originally used to remove the duplicate webpages from a search engine. But in data mining, MinHash used as a tool for association rule learning by Cohen at 2001.

15. Did Google conduct a large evaluation for comparing the performance by two technique MinHash and SimHash.
a) True
b) False
View Answer

Answer: a
Explanation: MinHash was originally used to remove the duplicate webpages from a search engine. But in data mining, MinHash used as a tool for association rule learning by Cohen at 2001. Google conducted a survey to compare the performance by two technique MinHash and SimHash.
Topic 105. Direct Addressing Tables

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Direct Addressing Tables”.

1. What is direct addressing?
a) Distinct array position for every possible key
b) Fewer array positions than keys
c) Fewer keys than array positions
d) Distinct array positions for keys based on priority
View Answer

Answer: a
Explanation: Direct addressing is possible only when we can afford to allocate an array that has one position for every possible key.

2. When is it appropriate to use direct addressing?
a) When the array is comparatively large
b) When the universe U of keys is reasonably small
c) When the universe U of keys is reasonably large
d) When the array is comparatively small

View Answer

Answer: b
Explanation: Since each key is associated with a slot in the array, it is better to use direct addressing when the universe of keys is small as the array size grows with the increase in number of keys.

3. What is the search complexity in direct addressing?
a) O(n)
b) O(logn)
c) O(nlogn)
d) O(1)
View Answer

Answer: d
Explanation: Since every key has a unique array position, searching takes a constant time.

4. What is the time complexity to insert an element into the direct address table?
a) O(n)
b) O(logn)
c) O(nlogn)
d) O(1)
View Answer

Answer: d
Explanation: As every key has a unique array position, it takes constant time to insert an element.

5. What is the advantage of using a dynamic set in direct addressing?
a) It saves time
b) It saves space
c) It saves both time and space
d) It reduces code complexity
View Answer

Answer: b
Explanation: Using a dynamic set, the size of the array is restricted to the number of keys, hence saves space. The complexity to implement dynamic array is larger than in normal case.

6. What is the time complexity to delete an element from the direct address table?
a) O(n)
b) O(logn)
c) O(nlogn)
d) O(1)
View Answer

Answer: d
Explanation: As every key has a unique array position, it takes constant time to delete an element, although the deleted position must be specified by nil.

7. How is a bit vector better compared to a normal array for implementing the hash table?
a) It saves time
b) It saves space
c) It saves both time and space
d) It reduces code complexity
View Answer

Answer: b
Explanation: A bit vector is an array of bits of only 0s and 1s, a bit vector of length m takes much less space than an array of m pointers. The complexity to implement bit vector is larger than in normal case.
Topic 106. Graph

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Graph”.

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

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
View Answer

Answer: c

Explanation: Adjacency Matrix, Adjacency List and Incidence Matrix are used to represent a graph.
Topic 107. Adjacency Matrix

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Adjacency Matrix”.

1. The number of elements in the adjacency matrix of a graph having 7 vertices is __________
a) 7
b) 14
c) 36
d) 49
View Answer

Answer: d
Explanation: There are n*n elements in the adjacency matrix of a graph with n vertices.

2. What would be the number of zeros in the adjacency matrix of the given graph?

a) 10
b) 6
c) 16
d) 0
View Answer

Answer: b
Explanation: Total number of values in the matrix is 4*4=16, out of which 6 entries are non zero.

3. Adjacency matrix of all graphs are symmetric.
a) False
b) True
View Answer

Answer: a
Explanation: Only undirected graphs produce symmetric adjacency matrices.

4. The time complexity to calculate the number of edges in a graph whose information in stored in form of an adjacency matrix is ____________
a) O(V)
b) O(E^2)
c) O(E)
d) O(V^2)
View Answer

Answer: d
Explanation: As V entries are 0, a total of V^2-V entries are to be examined.

5. For the adjacency matrix of a directed graph the row sum is the _________ degree and the column sum is the ________ degree.
a) in, out
b) out, in
c) in, total
d) total, out
View Answer

Answer: b
Explanation: Row number of the matrix represents the tail, while Column number represents the head of the edge.

6. What is the maximum number of possible non zero values in an adjacency matrix of a simple graph with n vertices?
a) (n*(n-1))/2
b) (n*(n+1))/2
c) n*(n-1)
d) n*(n+1)
View Answer

Answer: c
Explanation: Out of n*n possible values for a simple graph the diagonal values will always be zero.

7. On which of the following statements does the time complexity of checking if an edge exists between two particular vertices is not, depends?
a) Depends on the number of edges
b) Depends on the number of vertices

c) Is independent of both the number of edges and vertices
d) It depends on both the number of edges and vertices
View Answer

Answer: c
Explanation: To check if there is an edge between to vertices i and j, it is enough to see if the value of A[i][j] is 1 or 0, here A is the adjacency matrix.

8. In the given connected graph G, what is the value of rad(G) and diam(G)?
a) 2, 3
b) 3, 2
c) 2, 2
d) 3, 3
View Answer

Answer: a
Explanation: Value of eccentricity for vertices A, C is 2 whereas for F, B, D, E it is 3.

9. Which of these adjacency matrices represents a simple graph?
a) [[1, 0, 0], [0, 1, 0], [0, 1, 1]]
b) [[1, 1, 1], [1, 1, 1], [1, 1, 1]]
c) [[0, 0, 1], [0, 0, 0], [0, 0, 1]]
d) [[0, 0, 1], [1, 0, 1], [1, 0, 0]]
View Answer

Answer: d
Explanation: A simple graph must have no-self loops, should be undirected.

10. Given an adjacency matrix A = [[0, 1, 1], [1, 0, 1], [1, 1, 0]], The total no. of ways in which every vertex can walk to itself using 2 edges is ________
a) 2
b) 4
c) 6
d) 8
View Answer

Answer: c
Explanation: A^2 = [[2, 1, 1], [1, 2, 1], [1, 1, 2]], all the 3 vertices can reach to themselves in 2 ways, hence a total of 3*2, 6 ways.

11. If A[x+3][y+5] represents an adjacency matrix, which of these could be the value of x and y.
a) x=5, y=3
b) x=3, y=5
c) x=3, y=3
d) x=5, y=5
View Answer

Answer: a
Explanation: All adjacency matrices are square matrices.

12. Two directed graphs(G and H) are isomorphic if and only if A=PBP-1, where P and A are adjacency matrices of G and H respectively.
a) True
b) False
View Answer

Answer: a
Explanation: This is a property of isomorphic graphs.

13. Given the following program, what will be the 3rd number that'd get printed in the output sequence for the given input?

```
#include <bits/stdc++.h>
using namespace std;
int cur=0;
int G[10][10];
bool visited[10];
deque <int> q;

void fun(int n);

int main()
{
 int num=0;
 int n;
 cin>>n;

 for(int i=0;i<n;i++)
       for(int j=0;j<n;j++)
         cin>>G[i][j];

 for(int i=0;i<n;i++)
        visited[i]=false;

        fun(n);
 return 0;
}

void fun(int n)
{
 cout<<cur<<" ";
 visited[cur]=true;
 q.push_back(cur);

 do
        {
  for(int j=0;j<n;j++)
                {
      if(G[cur][j]==1 && !visited[j])
                    {
          q.push_back(j);
          cout<<j<<" ";
          visited[j]=true;
             }

                }

  q.pop_front();
  if(!q.empty())
  cur=q.front();
  }while(!q.empty());
}
```

Input Sequence:-

```
9
0 1 0 0 0 0 0 0 1
1 0 0 0 0 0 0 0 0
0 0 0 1 1 1 0 0 1
0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 1 0
0 0 1 0 0 0 1 0 0
0 0 0 0 0 1 0 1 1
0 0 0 0 1 0 1 0 0
1 0 1 0 0 0 1 0 0
```

a) 2
b) 6
c) 8
d) 4
View Answer

Answer: c
Explanation: The given code performs the breadth first search routine on the Graph.
The sequence obtained would be 0 1 8 2 6 3 4 5 7.

14. For which type of graph, the given program won’t run infinitely? The Input would be in the form of an adjacency Matrix and n is its dimension (1<n<10).

```
#include <bits/stdc++.h>
using namespace std;
int G[10][10];
void fun(int n);

int main()
{
 int num=0;
 int n;
 cin>>n;
 for(int i=0;i<n;i++)
   for(int j=0;j<n;j++)
         cin>>G[i][j];
     fun(n);
 return 0;
}

void fun(int n)
```

```
{
 for(int i=0;i<n;i++)
 for(int j=0;j<n;j++)
 if(G[i][j]==1)
 j--;
}
```

a) All Fully Connected Graphs
b) All Empty Graphs
c) All Bipartite Graphs
d) All simple graphs
View Answer

Answer: b
Explanation: For any graph (except empty graph) having edges, the condition G[i][j]==1 would hold true, which would result in an infinite loop.

15. Given the following adjacency matrix of a graph(G) determine the number of components in the G.

```
[0 1 1 0 0 0],
[1 0 1 0 0 0],
[1 1 0 0 0 0],
[0 0 0 0 1 0],
[0 0 0 1 0 0],
[0 0 0 0 0 0].
```

a) 1
b) 2
c) 3
d) 4
View Answer

Answer: c
Explanation: 0th 1st and 2nd vertices form a component, 3rd and 4th forms another and 5th vertex forms a component of a single vertex.
Topic 108. Incidence Matrix and Graph Structured Stack

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on "Incidence Matrix and Graph Structured Stack".

1. Incidence matrix and Adjacency matrix of a graph will always have same dimensions?
a) True
b) False
View Answer

Answer: b
Explanation: For a graph having V vertices and E edges, Adjacency matrix have V*V elements while Incidence matrix have V*E elements.

2. The column sum in an incidence matrix for a simple graph is __________
a) depends on number of edges
b) always greater than 2
c) equal to 2
d) equal to the number of edges
View Answer

Answer: c
Explanation: For every edge only the vertices with which it is connected would have the value 1 in the matrix, as an edge connects two vertices sum will always be 2.

3. What are the dimensions of an incidence matrix?
a) Number of edges*number of edges
b) Number of edges*number of vertices
c) Number of vertices*number of vertices
d) Number of edges * ($\frac{1}{2}$ * number of vertices)
View Answer

Answer: b
Explanation: Columns may represent edges and vertices may be represented by the rows.

4. The column sum in an incidence matrix for a directed graph having no self loop is __________
a) 0
b) 1

c) 2
d) equal to the number of edges
View Answer

Answer: a
Explanation: Under every edge column there would be either all 0 values or a pair of -1 and +1 value exists.

5. Time complexity to check if an edge exists between two vertices would be ___________
a) O(V*V)
b) O(V+E)
c) O(1)
d) O(E)
View Answer

Answer: d
Explanation: We have to check for all edges, in the worst case the vertices will have no common edge.

6. The graphs G1 and G2 with their incidences matrices given are Isomorphic.

$$\begin{array}{ccccccc} & e1 & e2 & e3 & e4 & e5 & e6 \\ v1 & 1 & 0 & 0 & 0 & 0 & 0 \\ v2 & 1 & 1 & 0 & 0 & 0 & 1 \\ v3 & 0 & 1 & 1 & 0 & 1 & 0 \\ v4 & 0 & 0 & 1 & 1 & 0 & 0 \\ v5 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}$$

$$\begin{array}{ccccccc} & e1 & e2 & e3 & e4 & e5 & e6 \\ v1 & 0 & 0 & 1 & 0 & 0 & 0 \\ v2 & 1 & 0 & 1 & 0 & 1 & 0 \\ v3 & 1 & 1 & 0 & 1 & 0 & 0 \\ v4 & 0 & 1 & 0 & 0 & 0 & 1 \\ v5 & 0 & 0 & 0 & 1 & 1 & 1 \end{array}$$

a) True
b) False
View Answer

Answer: a
Explanation: Two graphs are isomorphic if their Incidence Matrices differ only by permutation of columns and rows.

7. If a connected Graph (G) contains n vertices what would be the rank of its incidence matrix?
a) n-1
b) values greater than n are possible
c) values less than n-1 are possible
d) insufficient Information is given
View Answer

Answer: a
Explanation: Every column of the incidence matrix may contain only +1 and -1 as non zero entries rank would be less than n.

8. In the following DAG find out the number of required Stacks in order to represent it in a Graph Structured Stack.

a) 1
b) 2
c) 3
d) 4
View Answer

Answer: c
Explanation: Path ADE, BDE and BCE are possible.

9. A Graph Structured Stack is a _____________
a) Undirected Graph
b) Directed Graph
c) Directed Acyclic Graph
d) Regular Graph
View Answer

Answer: c
Explanation: A Graph Structured Stack is a Directed Acyclic Graph with each path representing a stack.

10. If a Graph Structured Stack contains {1,2,3,4} {1,5,3,4} {1,6,7,4} and {8,9,7,4}, what would be the source and sink vertices of the DAC?
a) Source – 1, 8 Sink – 7,4
b) Source – 1 Sink – 8,4
c) Source – 1, 8 Sink – 4
d) Source – 4, Sink – 1,8
View Answer

Answer: c
Explanation: Every Stack of the Graph Structured Stack represents a path, each path starts with the source vertex and ends with the sink vertex.

11. Graph Structured Stack finds its application in _____________
a) Bogo Sort
b) Tomita's Algorithm
c) Todd–Coxeter algorithm
d) Heap Sort
View Answer

Answer: b
Explanation: Tomita's is a parsing algorithm which uses Graph Structured Stack in its implementation.

12. If in a DAG N sink vertices and M source vertices exists, then the number of possible stacks in the Graph Structured Stack representation would come out to be N*M.
a) True
b) False
View Answer

Answer: b
Explanation: The answer would depend on the intermediate vertices also.
Topic 109. Adjacency List

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on "Adjacency List".

1. Space complexity for an adjacency list of an undirected graph having large values of V (vertices) and E (edges) is ___________
a) O(E)
b) O(V*V)
c) O(E+V)
d) O(V)
View Answer

Answer: c
Explanation: In an adjacency list for every vertex there is a linked list which have the values of the edges to which it is connected.

2. For some sparse graph an adjacency list is more space efficient against an adjacency matrix.
a) True
b) False
View Answer

Answer: a
Explanation: Space complexity for adjacency matrix is always O(V*V) while space complexity for adjacency list in this case would be O(V).

3. Time complexity to find if there is an edge between 2 particular vertices is _________
a) O(V)
b) O(E)
c) O(1)
d) O(V+E)
View Answer

Answer: a
Explanation: The maximum edges a vertex can have is V-1.

4. For the given conditions, which of the following is in the correct order of increasing space requirement?
i) Undirected, no weight
ii) Directed, no weight
iii) Directed, weighted
iv) Undirected, weighted

a) ii iii i iv
b) i iii ii iv
c) iv iii i ii
d) i ii iii iv
View Answer

Answer: a
Explanation: i) takes v+4e, ii) takes v+2e, iii) takes v+3e, iv) takes v +6e space.

5. Space complexity for an adjacency list of an undirected graph having large values of V (vertices) and E (edges) is __________
a) O(V)
b) O(E*E)
c) O(E)
d) O(E+V)
View Answer

Answer: c
Explanation: In an adjacency list for every vertex there is a linked list which have the values of the edges to which it is connected.

6. Complete the given snippet of code for the adjacency list representation of a weighted directed graph.

```
class neighbor
      {
 int vertex, weight;
 ____ next;
}

class vertex
      {
 string name;
 _____ adjlist;
}

vertex adjlists[101];
```

a) vertex, vertex
b) neighbor, vertex
c) neighbor, neighbor
d) vertex, neighbor
View Answer

Answer: c
Explanation: Vertex would have a name and a linked list attached to it.

7. In which case adjacency list is preferred in front of an adjacency matrix?
a) Dense graph
b) Sparse graph
c) Adjacency list is always preferred
d) Complete graph
View Answer

Answer: b
Explanation: In case of sparse graph most of the entries in the adjacency matrix would be 0, hence adjacency list would be preferred.

8. To create an adjacency list C++'s map container can be used.
a) True
b) False
View Answer

Answer: a
Explanation: We can create a mapping from string to a vector, where string would be the name of the vertex and vector would contains the name of the vertices to which it is connected.

9. What would be the time complexity of the following function which adds an edge between two vertices i and j, with some weight 'weigh' to the graph having V vertices?

```
vector<int> adjacent[15] ;
vector<int> weight[15];

void addEdge(int i,int j,int weigh)
{
 adjacent[a].push_back(i);
 adjacent[b].push_back(j);
 weight[a].push_back(weigh);
 weight[b].push_back(weigh);
}
```

a) O(1)
b) O(V)
c) O(V*V)
d) O(log V)
View Answer

Answer: a
Explanation: The function win in the constant time as all the four step takes constant time.

10. What would be the time complexity of the BFS traversal of a graph with n vertices and $n^{1.25}$ edges?
a) O(n)
b) O($n^{1.25}$)
c) O($n^{2.25}$)
d) O(n*n)
View Answer

Answer: b
Explanation: The time complexity for BFS is O(|V| + |E|) = O(n + $n^{1.25}$) = O($n^{1.25}$).
Topic 110. Undirected Graph

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on "Undirected Graph".

1. The number of possible undirected graphs which may have self loops but no multiple edges and have n vertices is ________
a) $2^{((n*(n-1))/2)}$
b) $2^{((n*(n+1))/2)}$
c) $2^{((n-1)*(n-1))/2)}$
d) $2^{((n*n)/2)}$
View Answer

Answer: d
Explanation: There can be at most, n*n edges in an undirected graph.

2. Given a plane graph, G having 2 connected component, having 6 vertices, 7 edges and 4 regions. What will be the number of connected components?
a) 1
b) 2
c) 3
d) 4
View Answer

Answer: b
Explanation: Euler's Identity says V – E + R = 1+ number of connected components.

3. Number of vertices with odd degrees in a graph having a eulerian walk is ________
a) 0
b) Can't be predicted
c) 2
d) either 0 or 2
View Answer

Answer: d
Explanation: If the start and end vertices for the path are same the answer would be 0 otherwise 2.

4. How many of the following statements are correct?
i) All cyclic graphs are complete graphs.
ii) All complete graphs are cyclic graphs.
iii) All paths are bipartite.
iv) All cyclic graphs are bipartite.
v) There are cyclic graphs which are complete.
a) 1
b) 2
c) 3
d) 4

View Answer

Answer: b
Explanation: Statements iii) and v) are correct.

5. All paths and cyclic graphs are bipartite graphs.
a) True
b) False
View Answer

Answer: b
Explanation: Only paths and even cycles are bipartite graphs.

6. What is the number of vertices of degree 2 in a path graph having n vertices,here n>2.
a) n-2
b) n
c) 2
d) 0
View Answer

Answer: a
Explanation: Only the first and the last vertex would have degree 1, others would be of degree 2.

7. All trees with n vertices consists of n-1 edges.
a) True
b) False
View Answer

Answer: a
Explanation: A trees is acyclic in nature.

8. Which of the following graphs are isomorphic to each other?

a) fig 1 and fig 2
b) fig 2 and fig 3
c) fig 1 and fig 3
d) fig 1, fig 2 and fig 3
View Answer

Answer: d
Explanation: All three graphs are Complete graphs with 4 vertices.

9. In the given graph which edge should be removed to make it a Bipartite Graph?

a) A-C
b) B-E
c) C-D
d) D-E
View Answer

Answer: a
Explanation: The resultant graph would be a Bipartite Graph having {A,C,E} and {D, B} as its subgroups.

10. What would the time complexity to check if an undirected graph with V vertices and E edges is Bipartite or not given its adjacency matrix?
a) O(E*E)
b) O(V*V)
c) O(E)
d) O(V)
View Answer

Answer: b
Explanation: A graph can be checked for being Bipartite by seeing if it is 2-colorable or not, which can be obtained with the help of BFS.
Topic 111. Directed Graph

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Directed Graph”.

1. Dijkstra’s Algorithm will work for both negative and positive weights?
a) True
b) False

View Answer

Answer: b
Explanation: Dijkstra’s Algorithm assumes all weights to be non-negative.

2. A graph having an edge from each vertex to every other vertex is called a ___________
a) Tightly Connected
b) Strongly Connected
c) Weakly Connected
d) Loosely Connected
View Answer

Answer: a
Explanation: This is a part of the nomenclature followed in Graph Theory.

3. What is the number of unlabeled simple directed graph that can be made with 1 or 2 vertices?
a) 2
b) 4
c) 5
d) 9
View Answer

Answer: b
Explanation:

4. Floyd Warshall Algorithm used to solve the shortest path problem has a time complexity of __________
a) O(V*V)
b) O(V*V*V)
c) O(E*V)
d) O(E*E)
View Answer

Answer: b
Explanation: The Algorithm uses Dynamic Programming and checks for every possible path.

5. All Graphs have unique representation on paper.
a) True
b) False
View Answer

Answer: b
Explanation: Same Graph may be drawn in different ways on paper.

6. Assuming value of every weight to be greater than 10, in which of the following cases the shortest path of a directed weighted graph from 2 vertices u and v will never change?
a) add all values by 10
b) subtract 10 from all the values
c) multiply all values by 10
d) in both the cases of multiplying and adding by 10
View Answer

Answer: c
Explanation: In case of addition or subtraction the shortest path may change because the number of edges between different paths may be different, while in case of multiplication path wont change.

7. What is the maximum possible number of edges in a directed graph with no self loops having 8 vertices?
a) 28
b) 64
c) 256
d) 56
View Answer

Answer: d
Explanation: If a graph has V vertices than every vertex can be connected to a possible of V-1 vertices.

8. What would be the DFS traversal of the given Graph?

a) ABCED
b) AEDCB
c) EDCBA
d) ADECB

View Answer

Answer: a
Explanation: In this case two answers are possible including ADEBC.

9. What would be the value of the distance matrix, after the execution of the given code?

```
#include <bits/stdc++.h>
#define INF 1000000
int graph[V][V] = {   {0,   7,  INF, 4},
                      {INF, 0,   13, INF},
                      {INF, INF, 0,   12},
                      {INF, INF, INF, 0}
                  };

int distance[V][V], i, j, k;

for (i = 0; i < V; i++)
        for (j = 0; j < V; j++)
     distance[i][j] = graph[i][j];

for (k = 0; k < V; k++)
 for (i = 0; i < V; i++)
         for (j = 0; j < V; j++)
                {
              if (distance[i][k] + distance[k][j] < distance[i][j])
                  distance[i][j] = distance[i][k] + distance[k][j];

                         return 0;
                }
```

a)

```
{
   {0,   7,  INF, 4},
   {INF, 0,   13, INF},
   {INF, INF, 0,   12},
   {INF, INF, INF, 0}
};
```

b)

```
{
   {0,   7,  20, 24},
   {INF, 0,   13, 25},
   {INF, INF, 0,   12},
   {INF, INF, INF, 0}
};
```

c)

```
{
   {0,   INF,  20, 24},
   {INF, INF,   13, 25},
   {INF, INF, 0,   12},
   {INF, INF, INF, 0}
   {INF, 0,   13, 25},
   {INF, INF, 0,   12},
   {24, INF, INF, 0}
};
```

d) None of the mentioned
View Answer

Answer: b
Explanation: The program computes the shortest sub distances.

10. What is the maximum number of edges present in a simple directed graph with 7 vertices if there exists no cycles in the graph?
a) 21
b) 7
c) 6
d) 49
View Answer

Answer: c
Explanation: If the no cycles exists then the difference between the number of vertices and edges is 1.
Topic 112. Directed Acyclic Graph

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Directed Acyclic Graph”.

1. Every Directed Acyclic Graph has at least one sink vertex.
a) True
b) False
View Answer

Answer: a
Explanation: A sink vertex is a vertex which has an outgoing degree of zero.

2. Which of the following is not a topological sorting of the given graph?

a) A B C D E F
b) A B F E D C
c) A B E C F D
d) A B C D F E
View Answer

Answer: d
Explanation: Topological sorting is a linear arrangement of vertices such that for every directed edge uv from vertex u to vertex v, u comes before v in the ordering. In A B C D F E, F comes before E in ordering.

3. With V(greater than 1) vertices, how many edges at most can a Directed Acyclic Graph possess?
a) (V*(V-1))/2
b) (V*(V+1))/2
c) (V+1)C2
d) (V-1)C2
View Answer

Answer: a
Explanation: The first edge would have an outgoing degree of atmost V-1, the next edge would have V-2 and so on, hence V-1 + V-2…. +1 equals (V*(V-1))/2.

4. The topological sorting of any DAG can be done in ________ time.
a) cubic
b) quadratic
c) linear
d) logarithmic
View Answer

Answer: c
Explanation: Topological sorting can be done in O(V+E), here V and E represents number of vertices and number of edges respectively.

5. If there are more than 1 topological sorting of a DAG is possible, which of the following is true.
a) Many Hamiltonian paths are possible
b) No Hamiltonian path is possible
c) Exactly 1 Hamiltonian path is possible
d) Given information is insufficient to comment anything
View Answer

Answer: b
Explanation: For a Hamiltonian path to exist all the vertices must be connected with a path, had that happened there would have been a unique topological sort.

6. What sequence would the BFS traversal of the given graph yield?

a) A F D B C E
b) C B A F E D
c) A B D C E F
d) E F D C B A
View Answer

Answer: c
Explanation: In BFS nodes gets explored and then the neighbors of the current node gets explored, before moving on to the next levels.

7. What would be the output of the following C++ program if the given input is

```
0 0 0 1 1
0 0 0 0 1
0 0 0 1 0
1 0 1 0 0
```

```
1 1 0 0 0

#include <bits/stdc++.h>
using namespace std;
bool visited[5];
int G[5][5];

void fun(int i)
{
 cout<<i<<" ";
 visited[i]=true;
 for(int j=0;j<5;j++)
  if(!visited[j]&&G[i][j]==1)
   fun(j);
}

int main()
{
 for(int i=0;i<5;i++)
  for(int j=0;j<5;j++)
   cin>>G[i][j];

 for(int i=0;i<5;i++)
  visited[i]=0;

 fun(0);
  return 0;
}
```

a) 0 2 3 1 4
b) 0 3 2 4 1
c) 0 2 3 4 1
d) 0 3 2 1 4
View Answer

Answer: b
Explanation: Given Input is the adjacency matrix of a graph G, whereas the function ‘fun’ prints the DFS traversal.

8. Which of the given statement is true?
a) All the Cyclic Directed Graphs have topological sortings
b) All the Acyclic Directed Graphs have topological sortings
c) All Directed Graphs have topological sortings
d) All the cyclic directed graphs hace non topological sortings
View Answer

Answer: d
Explanation: Cyclic Directed Graphs cannot be sorted topologically.

9. For any two different vertices u and v of an Acyclic Directed Graph if v is reachable from u, u is also reachable from v?
a) True
b) False
View Answer

Answer: b
Explanation: If such vertices exists it means that the graph contains a cycle which contradicts the first part of the statement.

10. What is the value of the sum of the minimum in-degree and maximum out-degree of an Directed Acyclic Graph?
a) Depends on a Graph
b) Will always be zero
c) Will always be greater than zero
d) May be zero or greater than zero
View Answer

Answer: b
Explanation: Every Directed Acyclic Graph has a source and a sink vertex.
Topic 113. Propositional and Directed Acyclic Word Graph

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Propositional and Directed Acyclic Word Graph”.

1. In which of the following does a Directed Acyclic Word Graph finds its application in?
a) String Matching
b) Number Sorting
c) Manipulations on numbers
d) Pattern Printing
View Answer

Answer: a
Explanation: A Directed Acyclic Word Graph is similar to suffix tree, it can be viewed as a Deterministic Finite Automata.

2. What is the number of words that can be formed from the given Directed Acyclic Word Graph?

a) 2
b) 4
c) 12
d) 7
View Answer

Answer: b
Explanation: Words namely BATS, BOTS, BAT and BOT can be formed.

3. Determine the longest string which is described by the given Directed Acyclic Word Graph.

a) BATS
b) BOATS
c) BOT
d) BAT
View Answer

Answer: a
Explanation: Starting from the initial state and choosing B, A, T, S respectively.

4. What is time complexity to check if a string(length S1) is a substring of another string(length S2) stored in a Directed Acyclic Word Graph, given S2 is greater than S1?
a) O(S1)
b) O(S2)
c) O(S1+S2)
d) O(1)
View Answer

Answer: a
Explanation: For each check of a word of length S1, we need to follow at most S1 edges.

5. In which of the following case does a Propositional Directed Acyclic Graph is used for?
a) Representation of Boolean Functions
b) String Matching
c) Searching
d) Sorting of number
View Answer

Answer: a
Explanation: A Propositional Directed Acyclic Graph is used to represent a boolean function.

6. Consider the following symbols and choose which of the symbols represent nodes having atleast one child?

i) Δ ii) ◊ iii) ▯ iv) T v) ▯

a) iv) and v)
b) iii) iv) and v)
c) i) and ii)
d) i) and iii)
View Answer

Answer: c
Explanation: The symbols Δ and ◊ represents logical AND and OR gates.

7. Which of the following symbols represent nodes having exactly one child?

i) Δ ii) ◊ iii) ▯ iv) T v) ▯

a) iv) and v)
b) v)
c) i) and iii)
d) iii)
View Answer

Answer: d
Explanation: ▯ symbol represents the logical NOT gate.

8. Which of the following symbols represent leaf nodes?

i) Δ ii) ◊ iii) ▯ iv) T v) ▯

a) iv) and v)
b) v)
c) i) and iii)
d) ii)
View Answer

Answer: a
Explanation: The two symbols T, ▯ represent the Boolean values.

9. Every Binary Decision Diagram is also a Propositional Directed Acyclic Graph.
a) True
b) False
View Answer

Answer: a
Explanation: Both Binary Decision Diagram and Propositional Directed Acyclic Graph may be used to represent the same Boolean function.

10. In a Propositional Directed Acyclic Graph Leaves maybe labelled with a boolean variable.
a) True
b) False
View Answer

Answer: a
Explanation: In a Propositional Directed Acyclic Graph leaves maybe labelled with a boolean variable, T or ▯.
Topic 114. Multigraph and Hypergraph

This set of Data Structure Multiple Choice Questions & Answers (MCQs) focuses on “Multigraph and Hypergraph”.

1. Given Adjacency matrices determine which of them are PseudoGraphs?
i) {{1,0} {0,1}}
ii) {{0,1}{1,0}}
iii) {{0,0,1}{0,1,0}{1,0,0}}
a) only i)
b) ii) and iii)
c) i) and iii)
d) i) ii) and iii)
View Answer

Answer: c
Explanation: In i) self loops exist for both the vertices, in iii) self loop exists in the second vertex.

2. All undirected Multigraphs contain eulerian cycles.
a) True
b) False
View Answer

Answer: a
Explanation: Only graphs with every vertex having even degree have eulerian circuits or cycles.

3. Determine the number of vertices for the given Graph or Multigraph?
G is a 4-regular Graph having 12 edges.
a) 3
b) 6
c) 4
d) Information given is insufficient
View Answer

Answer: b
Explanation: Sum of degrees of all the edges equal to 2 times the number of edges. 2*12=4*n, n=>6.

4. Which of the following statement is true.
a) There exists a Simple Graph having 10 vertices such that minimum degree of the graph is 0 and maximum degree is 9
b) There exists a MultiGraph having 10 vertices such that minimum degree of the graph is 0 and maximum degree is 9
c) There exists a MultiGraph as well as a Simple Graph having 10 vertices such that minimum degree of the graph is 0 and maximum degree is 9
d) None of the mentioned

View Answer

Answer: b
Explanation: If a vertex has a degree 9 that means it is connected to all the other vertices, in case of Multigraphs for an isolate vertex, and a multiple edge may compensate.

5. Given Adjacency matrices determine which of them are PseudoGraphs?
i) {{1,0} {0,1}}
ii) {{0,1}{1,0}}
iii) {{0,0,1}{0,1,0}{1,0,0}}
a) only i)
b) ii) and iii)
c) i) and iii)
d) i) ii) and iii)
View Answer

Answer: c
Explanation: In i) self loops exist for both the vertices, in iii) self loop exists in the second vertex.

6. Possible number of labelled simple Directed, Pseudo and Multigarphs exist having 2 vertices?
a) 3, Infinite, 4
b) 4, 3, Infinite
c) 4, Infinite, infinite
d) 4, Infinite, Infinite
View Answer

Answer: d
Explanation: MultiGraphs and PseudoGraphs may have infinite number of edges, while 4 possible simple graphs exist.

7. Which of the following is a HyperGraph, where V is the set of vertices, E is the set of edges?
a) V = {v1, v2, v3} E = {e1, e2} = {{v2, v3} {v1, v3}}
b) V = {v1, v2} E = {e1} = {{v1, v2}}
c) V = {v1, v2, v3} E = {e1, e2, e3} = {{v2, v3}{v3, v1}{v2, v1}}
d) All of the mentioned
View Answer

Answer: d
Explanation: In a uniform Graph all the hyper-edges have the same cardinality.

8. What would be the Incidence Matrix of the given HyperGraph?
V = {x,y,z} E = {{x,y}{y}{x,z}{z,y}}
a) {{1,0,1,0},
{1,1,0,1},
{0,0,1,1}}
b) {{1,1,0,0},
{0,1,0,0},
{1,1,1,0}}
c) {{0,1,0,1},
{0,0,1,0},
{1,1,0,0}}
d) None of the Mentioned
View Answer

Answer: a
Explanation: The columns represent edges while rows represent vertices.

9. What is the degree sequence of the given HyperGraph, in non-increasing order.
V = {v1,v2,v3,v4,v5,v6} E = {{v1,v4,v5} {v2,v3,v4,v5} {v2} {v1} {v1,v6}}
a) 3,2,1,1,1,1
b) 3,2,2,2,1,1
c) 3,2,2,2,2,1
d) 3,2,2,1,1,1
View Answer

Answer: b
Explanation: The degree of v1,v2,v3,v4,v5,v6 is 3,2,1,2,2,1 respectively.

10. MultiGraphs having self-loops are called PseudoGraphs?
a) True
b) False

View Answer

Answer: a
Explanation: All PsuedoGraphs are MultiGraphs, but all MultiGraphs are not PseudoGraphs as all PseudoGraphs have self loop, but all MultiGraphs do not have self loops.
Topic 115. Binary Decision Diagrams & and Inverter Graph

This set of Data Structure test focuses on “Binary Decision Diagrams & And Inverter Graph”.

1. Binary Decision Diagram is a type of __________
a) Multigraph
b) Cyclic Graph
c) Directed Acyclic Graph
d) Directed Acyclic Word Graph
View Answer

Answer: c
Explanation: An Inverter is a directed graph which is used to solve Boolean expressions, hence have no loops.

2. In which of the following case does a Binary Decision Diagram is used for?
a) Representation of Boolean Functions
b) String Matching
c) Searching
d) Sorting of number
View Answer

Answer: a
Explanation: A Binary Decision Diagram is used to represent a Boolean function.

3. In a Binary Decision Diagram, how many types of terminal exists?
a) 1
b) 2
c) 3
d) 4
View Answer

Answer: b
Explanation: In a BDD, 2 terminals namely terminal-0 and terminal-1 exists.

4. In a Binary Decision Diagrams 0 values by a _________ line and the 1 values are represented by a _________ line.
a) dashed, bold
b) bold, dashed
c) dotted, bold
d) dotted, dashed
View Answer

Answer: c
Explanation: It is used to distinguish between the 2 values without explicitly writing.

5. How many nodes are required to create a Binary Decision Tree having 4 variables?
a) 2^4
b) 2^{4-1}
c) 2^5
d) 2^{5-1}
View Answer

Answer: d
Explanation: Binary Decision Trees are complete Binary Trees of level V + 1, here V is the number of variables.

6. Two or more And Inverter Graphs can represent same function.
a) True
b) False
View Answer

Answer: a
Explanation: And Inverter Graphs are not canonical in nature.

7. Size of an And Inverter Graph is the number of _______ gates and the number of logic levels is number of ________ gates on the __________ path from a primary input to a primary output.
a) AND, AND, average

b) AND, OR, longest
c) OR, OR, shortest
d) AND, AND, longest
View Answer

Answer: d
Explanation: The given statement forms the attributes of the And Inverter Graph.

8. And Inverter Graph is a type of __________
a) Multigraph
b) Cyclic Graph
c) Directed Acyclic Graph
d) Directed Acyclic Word Graph
View Answer

Answer: c
Explanation: And Inverter is a directed graph which is used to solve boolean expressions, hence have no loops.

9. The And Inverter Graph representation of a Boolean function is more efficient than the Binary Decision Diagram.
a) True
b) False
View Answer

Answer: a
Explanation: The conversion from the network logic is faster and more scalable than in the case of the Binary Decision Diagram.

10. Which of the following logical operation can’t be implemented by polynomial time graph manipulation algorithms using Binary Decision Diagrams?
a) Conjunction
b) Disjunction
c) Negation
d) Tautology Checking
View Answer

Answer: d
Explanation: In Binary Decision Diagram, Conjunction, Disjunction, Negotiation can be implemented in polynomial time whereas tautology checking can be implemented in linear time.
Total Topics Found: 115 Total Q/A Found: 1271