Job Interview Questions Series

Data Structures & Algorithms

Interview Questions

You'll Most Likely Be Asked

FREE
2 Aptitude Tests, download available on website

200 Data Structures & Algorithms Questions

51 HR Interview Questions

VP Vibrant Publishers

Data Structures & Algorithms Interview Questions

You'll Most Likely Be Asked

Job Interview Questions Series

www.vibrantpublishers.com

*****

The publisher wishes to thank Dana Mitrea (Romania) for her invaluable inputs to this edition.

Vibrant Publishers books are available at special quantity discount for sales promotions, or for use in corporate training programs. For more information please write to bulkorders@vibrantpublishers.com Please email feedback / corrections (technical, grammatical or spelling) to spellerrors@vibrantpublishers.com To access the complete catalogue of Vibrant Publishers, visit www.vibrantpublishers.com *****

Contents

Queues
Lists
Hash Data Structures
Trees
Sets
Graphs

2. Algorithms

General Concepts Of Algorithms
Sorting Algorithms
Search Algorithms
Huffman Coding

*****

Data Structures & Algorithms Questions

Review these typical interview questions and think about how you would answer them. Read the answers listed; you will find best possible answers along with strategies and suggestions.

*****

Data Structures

Description: Questions in this category are related to Data Structures

Topics:

General Concepts Of Data Structures
Arrays
Stacks
Queues
Lists

Hash Data Structures
Trees
Sets
Graphs

*****

General Concepts Of Data Structures

1: What is a data structure?
Answer:
A data structure is a way to store and organize data in order to facilitate access and modifications.

2: Which are the most important data structures categories?
Answer:
Data structures can be divided into several main categories:
a) data types (primitive, composite, abstract data types)
b) linear data structures (arrays, lists)
c) trees
d) hashes
e) graphs

3: Define the abstraction mechanism. What are its benefits?
Answer:
Abstraction can be thought of as a mechanism for suppressing irrelevant details while at the same time emphasizing relevant ones. An important benefit of abstraction is that it makes it easier for the programmer to think about the problem to be solved.

4: What is encapsulation?
Answer:
Encapsulation is the mechanism of enforcing information hiding. Objects encapsulate data and the procedures for manipulating that data, thus the object hides the details of the implementation from the user of that object.

5: What is a binary numbering system?
Answer:

A binary numbering system consists of two digits called binary digits (bits): zero and one. A switch in the off state represents zero, and a switch in the on state represents one. This means that one transistor can represent one of two digits.

6: What is the binding process?
Answer:
Binding is the process of assigning a value to an attribute. When a value is assigned to an attribute, that attribute is said to be bound to the value. As an example, when defining a variable *int i = 5* , the value of integer variable *i* is bound to *5* .

7: What is the difference between early and late binding?
Answer:
Early binding is the binding done statically by the compiler - it is realized by method overloading. Late binding is done dynamically at run-time and it is realized through inheritance and polymorphism.

8: What is an abstract data type?
Answer:
An abstract data type is a mathematical model for a certain class of data structures that have similar behavior. It is defined by three main characteristics: encapsulation, inheritance and polymorphism.

9: What is a user-defined data type?
Answer:
A user-defined data type is a group of primitive data types defined by the programmer. As an example, in C programming language, a user can define a new data type using *typedef* keyword *(typedef int age)*

10: What is an object container?
Answer:
A container is an object that holds within it other objects. Generally, container classes are expected to implement methods to do the following: a) create a new empty container (constructor)
b) report the number of objects it stores (size)
c) delete all the objects in the container (clear)
d) insert new objects into the container

e) remove objects from it
f) provide access to the stored objects

11: What is the difference between direct and indirect containment?
Answer:
A direct containment contains copies of objects while indirect containments contain pointers to objects. An example of direct container in C++ is an array, which stores the object values. An indirect containment example would be a dynamic linked list (C++ also).

12: Which are the main primitive data type groups?
Answer:
There are four primitive data type groups:
a) Integer - stores whole numbers and signed numbers
b) Floating-point - store real numbers (fractional values)
c) Character - store characters (for example names of things)
d) Boolean - store a true or false value

13: What is a record structure? Give some practical examples.
Answer:
A record structure is a join of elements of arbitrary types, that are possibly themselves structured types. For example, a Person record may contain fields like FirstName, LastName, Birthday, Person (reference to his children) and so on.

14: How can be determined the size of a structure?
Answer:
It can be determined by calculating the sum of the sizes of all the primitive data types within the structure.

15: Which are the factors that need to be taken into account when choosing a specific data type for an algorithm?
Answer:
Relative advantages and disadvantages of different data types include: a) asymptotic operation performance - average lookup and insertion time, worst-case lookup and insertion time b) ordering preservation - allowing one to efficiently iterate over the keys in order or to efficiently locate an association

whose key is nearest to a given value c) allocation behavior - the way data is organized into memory
d) compactness
e) persistence

16: What is a pointer variable?
Answer:
A pointer variable contains a memory address of another memory location, which is usually the memory address of another variable, element of a structure or attribute of a class. In C++ a pointer variable can be declared as: *int * i ;*

17: What is the difference between static and dynamic data types?
Answer:
Static data structures allow fast access to elements but are expensive to insert/remove elements and have fixed, maximum size. On the other side, dynamic data structures offer fast insertion/deletion of element but slower access to elements and have flexible size.

18: Describe the referencing mechanism.
Answer:
Referencing mechanism tells the computer to copy the memory address of the variable instead of copying the value stored in that memory address. A reference is distinct from the data itself. Typically, a reference is the physical address of where the data is stored in memory or in the storage device. For this reason, a reference is often called a pointer or address, and is said to point to the data.

*****

Arrays

19: Describe an array structure.
Answer:
An array consists of components which are all of the same type, called its base type; it is therefore called a homogeneous structure. It is a random-access structure, because all components can be selected at random and are equally quickly accessible. In order to denote an individual component, the name of the entire structure is augmented by the index selecting the component.

20: What is a multidimensional array?

Answer:

A multidimensional array consists of two or more arrays defined by sets of array elements. Each set of array elements is an array. The first set of array elements is considered the primary array, and the second and subsequent sets of array elements are considered subarrays.

21: Give an example of practical use of a multidimensional array.

Answer:

A multidimensional array can be useful to organize subgroups of data within an array. As an example, it can be used to organize a student's grades (if he has three grades, a mid-term grade, a final exam grade, and a final grade). All the three grades can be organized for an endless number of students in a two-dimensional array.

22: What is a parallel array?

Answer:

A parallel array is a data structure which represents arrays of records. It keeps a separate, homogeneous array for each field of the record, each having the same number of elements. Then, objects located at the same index in each array are implicitly the fields of a single record. Pointers from one object to another are replaced by array indices. This contrasts with the normal approach of storing all fields of each record together in memory.

23: Define an associative array.

Answer:

An associative array is an abstract data type composed of a collection of unique keys and a collection of values, where each key is associated with one value (or set of values). The operation of finding the value associated with a key is called a lookup or indexing.

24: Which are the operations usually defined on associative arrays:

Answer:

The operations that are usually defined on associative arrays
are:

a) add - bind a new key to a new value

b) reassign - bind an old key to a new value
c) remove - unbind a key from a value and remove the key from the key set
d) lookup - find the value (if any) that is bound to a key

25: Which the main operations defined on basic abstract arrays?
Answer:
An abstract array structure has two main operations, fetch and store.
a) fetch : obtains the data stored in the element of the array whose state is A and whose index is i b) store : the array state that results by setting the value of that element to V in array state A **26: What is a Judy array?**
Answer:
Judy array is a complex but very fast associative array data structure for storing and looking up values using integer or string keys. Unlike normal arrays, Judy arrays may be sparse; that is, they may have large ranges of unassigned indices. Due to its cache optimizations, Judy arrays are fast, sometimes even faster than a hash table, especially for very big datasets.

27: What is the relation between pointers and arrays? Provide a short example.
Answer:
The array name is like a pointer variable in that the array name by itself references the address of the first element of the array. For example, if there is an array of characters called 'letters' and there is also declared a character pointer called 'pLetters' and the character A is assigned to the first element of the array, the address of the first array element is assigned to the pointer variable.

28: What is the benefit of using arrays of pointers instead of several pointer variables?
Answer:
The benefit of using arrays of pointers is that you can use a for loop to step through each element of the array to access memory addresses that are assigned to the array. This must be done in order to efficiently access and reorder the values stored in memory.

29: How are elements of an array stored in memory?
Answer:
The elements of an array are stored sequentially in memory (one after another).

Each element of the array is identified by a unique index.

*****

Stacks

30: What is a stack? Give a practical example of a stack.

Answer:

A stack is a sequential data organization where data is accessible using LIFO (last in, first out). For example, a stack of dishes.

31: Which are the main operations performed on a stack?

Answer:

The operations that handle data insert/remove from a stack are the following: a) Push( ) - used when placing an item on a stack; the item is placed on top of the stack

b) Pop( ) - the reverse process of pushing; it removes an item from the stack

*****

Queues

32: What is a queue? Give a practical example of a queue.

Answer:

A queue is a sequential data organization where data is accessible using FIFO. That is, the first data in the queue is the first data that is accessible by the program. For example, the checkout line at the supermarket.

33: What is the difference between a simple queue and a priority queue?

Answer:

In a simple queue each item is processed in the order in which it appears in the queue. Processing in a priority queue is based on items priority (items on a priority queue can jump to the front of the line if they have priority) **34: Which are the main operations performed on a queue?**

Answer:

The operations that handle data insert/remove from a queue are the following: a) Enqueue ( ) - inserts a value into the queue

b) Dequeue ( ) - removes a value from the front of the queue

35: What is a deque?
Answer:
The word deque is an acronym derived from double-ended queue; it is an extension of the queue which provides means to insert and remove items at both ends of the queue.

*****

Lists

36: What are the association lists? What are their advantages?
Answer:
An association list is a simple but generally inefficient type of associative array, which simply stores a linked list of key-value pairs. Each lookup does a linear search through the list looking for a key match. Advantages of association lists include: a) they need only to known how to test the keys for equality
b) for small associative arrays, common in some applications, association lists can take less time and space than other data structures c) insertions are done in constant time by adding the new association to the head of the list

37: What is a linked list?
Answer:
A linked list is a dynamic data type consisting of a number of structures (i.e. nodes) which are connected together using pointers.

38: What is a single linked list?
Answer:
It is the most basic of all the pointer-based data structures. A singly-linked list is simply a sequence of dynamically allocated storage elements, each containing a pointer to its successor. Below is described a single linked list node which contains an integer data (C++) *typedef struct node*

{

int data; // will store information
*node *next; // the reference to the next node*

};

39: What is the difference between a single linked list and a double linked list?

Answer:

In a single linked list each node contains reference only to the next node while in a double linked list each node contains reference to both the previous and the next node of the list. Below is described a double linked list node which contains an integer data (C++) *typedef struct node*

{

int data; // will store information
*node *next; // the reference to the next node*
*node * prev // the reference to the previous node*

};

40: What is a circular list?

Answer:

A circular list is a special sort of list in which the last node points to the first node. The list may be either simple or double linked.

41: What condition defines an empty dynamic list?

Answer:

A dynamic list is empty when both the front and back pointers are null.

42: Which condition must be accomplished in order to have only one element in a linked list?

Answer:

A linked list contains only one element if the front and back pointers point both to the same node.

43: What is the maximum dimension a dynamic list can have?

Answer:

The dimension of a dynamic list is limited only by the amount of memory on the machine.

44: Which is the difference between an ordered list and a sorted list?
Answer:
In an ordered list the order of the items is significant. A sorted list is one in which the order of the items is defined by some collating sequence (for example, the indexes of a book is form a sorted list) **45: Which are the most common used operations on ordered lists?**
Answer:
Below are described several operations on ordered list (their name may vary from a programming language to another): a) insert - used to put objects into a the container
b) withdraw - used to remove objects from the container
c) find - used to locate objects in the container
d) isMember - used to test whether a given object instance is in the container
e) findPosition - used to find the position of an object in the ordered list
f) insertAfter - used to insert an object into the ordered list after the object at a given position g) insertBefore - used to insert an object into the ordered list before the object at a given position **46: Which are the most common used operations on sorted lists?**
Answer:
The most common used operations on sorted links are the following:
a) findPosition - used to find the position of an object in the sorted list
b) operator [] - used to access the object at a given position in the sorted list
c) withdraw - used to remove the object at a given position from the sorted list

47: Why insertAfter and insertBefore operations are not provided for sorted lists?
Answer:
Because they allow arbitrary insertions, but arbitrary insertions do not necessarily result in sorted lists.

48: Describe several usage models of circularly-linked lists.
Answer:
A circularly linked list may be a natural option to represent arrays that are naturally circular, *e.g.* the corners of a polygon, a pool of buffers that are used

and released in FIFO order, or a set of processes that should be time-shared in round-robin order. In these applications, a pointer to any node serves as a handle to the whole list. With a circular list, a pointer to the last node gives easy access also to the first node, by following one link. Thus, in applications that require access to both ends of the list (e.g., in the implementation of a queue), a circular structure allows one to handle the structure by a single pointer, instead of two.

49: What is a skip list?
Answer:
A skip list is a data structure for storing a sorted list of items, using a hierarchy of linked lists that connect increasingly sparse subsequences of the items. These auxiliary lists allow item lookup with efficiency comparable to balanced binary search trees.

50: Give example of a way to improve the speed of random access lookups in a skip list.
Answer:
The speed of random access lookups in a skip list can be improved by using an indexable skiplist. To index the skiplist and find the i-th value, the skiplist must be traversed and the widths of each traversed link must be counted down.

51: Give example of several applications and frameworks that use skip lists:
Answer:
Below are described several applications that are built on skip lists: a) QMap - template class of Qt that provides a dictionary implementation
b) Redis - an ANSI-C open-source persistent key/value store for Posix systems
c) skipdb - an open-source database format using ordered key/value pairs
d) ConcurrentSkipListSet and ConcurrentSkipListMap - implemented in the Java 1.6 API

52: What is the purpose of using sentinel nodes?
Answer:
Usage of sentinel nodes may simplify certain list operations, by ensuring that the next and/or previous nodes exist for every element and that even empty lists have at least one node. One may also use a sentinel node at the end of the list, with an appropriate data field, to eliminate some end-of-list tests.

53: What is an unrolled linked list?
Answer:
An unrolled linked list is a linked list in which each node contains an array of data values. This leads to improved cache performance, since more list elements are contiguous in memory, and reduced memory overhead, because less metadata needs to be stored for each element of the list.

*****

Hash Data Structures

54: What is a hash function?
Answer:
A hash function is a well-defined procedure or mathematical function that takes a variable-size input m and returns a fixed-size string, usually called hash value, hash code, hash sum or checksum.

55: What are the characteristics of a good hash function?
Answer:
A good hash function:
a) avoids collisions
b) tends to spread keys evenly in the array
c) is easy to compute

56: What is a collision?
Answer:
A collision is the situation when two different keys x and y have the same hash code - thus, h(x) = h(y).

57: Enumerate several hashing methods.
Answer:
Here are some hashing methods:
a) division method
b) middle square method
c) multiplication method
d) Fibonacci hashing

58: Describe the division hashing method.

Answer:
It is the simplest of all the hashing methods. It consists in dividing an integer x by a natural number M and then to use the remainder modulo M. In this case, the hash function is: $h(x) = x \bmod M$.

59: What is a scatter table?
Answer:
A scatter table is an array-based hash table. The essential idea behind a scatter table is that all of the information is stored within a fixed size array and hashing is used to identify the position where an item should be stored.

60: What is a chained scatter table?
Answer:
It is a scatter table whose elements are ordered pairs. Each array element contains a key and a pointer. All the keys are stored in the table itself. Consequently, there is a fixed limit on the number of items that can be stored in a scatter table.

61: What is open addressing?
Answer:
It is a method of collision resolution in hash tables. With this method a collision is resolved by searching through different locations in the array (the probe sequence) until either the target record is found, or an unused array slot is found, which indicates that there is no such key in the table.

62: Describe several collision resolution strategies in open addressing.
Answer:
There are three resolution strategies in open addressing:
a) linear probing - the interval between probes is fixed - usually at 1
b) quadratic probing - the interval between probes increases linearly. This way, the indices are described by a quadratic function.
c) double hashing - the interval between probes is fixed for each record but is computed by another hash function **63: What is lazy deletion?**
Answer:
It is a method of deleting elements from a hash table which uses open addressing. Deletions are performed by marking an element as deleted, without erasing it entirely. Deleted locations are treated as empty when inserting and as

occupied during a search.

64: Describe the mechanism of inserting a node into a hashtable.
Answer:
Here are the steps of inserting a new node into a hashtable:
a) call the hashing algorithm for the new node key
b) go to the array and see if the value in the array index is null
c) if it is null then change this value to the address of the new node
d) if the array index is not null, then set the next pointer in the new node to the value at the array index and set the array index to the address of the new node
65: Describe the steps that need to be done in order to delete an element from a hashtable.
Answer:
Here are the needed steps in order to delete a hashtable element:
a) call the hashing algorithm for the key of the element that needs to be deleted
b) go to that index in the array
c) traverse the linked list and find the value, and then delete this entry from the linked list. The entry is deleted by setting the next pointer in the previous node to the next pointer of the node being deleted.

66: What condition must be accomplished in order to decide that a hashtable is empty?
Answer:
A hashtable is empty if all the values from the array are null.

67: What is a Bloom filter?
Answer:
The Bloom filter, conceived by Burton Howard Bloom, is a space-efficient probabilistic data structure that is used to test whether an element is a member of a set. False positives are possible, but false negatives are not. It is used in many applications, here are several examples: a) Google BigTable uses Bloom filters to reduce the disk lookups for non-existent rows or columns
b) the Squid Web Proxy Cache uses Bloom filters for cache digests
c) the Venti archival storage system uses Bloom filters to detect previously-stored data
d) the SPIN model checker uses Bloom filters to track the reachable state space for large verification problems **68: What is a hashed array tree?**

Answer:
A hashed array tree (HAT) is a dynamic array algorithm which can perform indexing in constant time. The algorithm has O(1) amortized performance when appending a series of objects to the end of a hashed array tree. Contrary to its name, it does not use hash functions.

*****

Trees

69: What is a tree?
Answer:
A tree is a finite, non-empty set of nodes, with the following properties: a) a designated node of the set, r, is called the root of the tree
b) the remaining nodes are partitioned into subsets, each of which is a tree.

70: Describe several properties of a tree node.
Answer:
a) the depth of a node - represents the length of the path (or the number of edges) from the root to that node b) the height of a node - represents the longest path from that node to its leaves.
c) the height of a tree - represents the height of the root
d) a leaf node - represents a node without children; its only path is up to its parent

71: What is the degree of a tree node?
Answer:
The degree of a tree node is the number of subtrees associated with that node. For example, if a tree node has degree 3, then we can know for sure that there are exactly 3 subtrees of that node.

72: What is an N-ary tree?
Answer:
An N-ary tree is either the empty tree or it is a non-empty set of nodes which consists of a root and exactly N subtrees. It is a generalization of a binary tree.

73: What is a binary tree?
Answer:

A binary tree is a finite set of nodes with one of the following properties: a) the set is empty or
b) the set consists of a root and exactly two distinct binary trees, left and right subtree.

74: Which are the two methods to visit all the nodes of a tree?
Answer:
The two methods of traversing all of the tree nodes are:
a) depth-first traversal
b) breadth-first traversal

75: Which are the depth-first traversal methods?
Answer:
There are three depth-first traversal methods:
a) preorder
b) inorder
c) postorder

76: Describe the preorder traversal algorithm.
Answer:
Preorder traversal gets its name from the fact that it visits the root first. The steps are the following: a) visit the root
b) traverse the left subtree
c) traverse the right subtree

77: Describe the inorder traversal algorithm.
Answer:
Inorder traversal gets its name from the fact that it visits the root in between visiting the left and right subtrees. The steps are the following: a) traverse the left subtree
b) visit the root
c) traverse the right subtree

78: Describe the postorder traversal algorithm.
Answer:
Postorder traversal gets its name from the fact that it visits the root last. The steps are the following: a) traverse the left subtree

b) traverse the right subtree
c) visit the root

79: Describe the breadth-first traversal algorithm.
Answer:
The breadth-first traversal of a tree visits the nodes in the order of their depth in the tree. Breadth-first traversal first visits all the nodes at depth zero (i.e. the root), then all the nodes at depth one, and so on. At each depth the nodes are visited from left to right.

80: Supposing we have an expression tree, which are the three notations associated to preorder, inorder and postorder traversal algorithms?
Answer:
a) for preorder traversal - the prefix notation
b) for inorder traversal - the infix notation
c) for postorder traversal - the postfix notation

81: Enumerate several tree specializations.
Answer:
Heap, B-tree, binary tree, balanced tree, multiway tree, complete tree, search tree.

82: What is a heap?
Answer:
A heap is a specialized tree-based data structure that satisfies the heap property: if B is a child node of A, then key(A) >= key(B). This implies that an element with the greatest key is always in the root node.

83: What is the difference between a max-heap and a min-heap?
Answer:
a) A max-heap respects the property that if B is a child node of A, then key(A) >= key(B) - the greatest key is always in the root node
b) A min-heap respects the property that if B is a child node of A, then key(A) <= key(B) - the smallest key is always in the root node

84: Which are the operations usually performed on a heap?
Answer:

The operations commonly performed on a heap are:
a) find-max or find-min - find the maximum item of a max-heap or a minimum item of a min-heap b) delete-max or delete-min - removing the root node of a max/min-heap c) increase-key or decrease-key - updating a key within a max/min-heap d) insert - adding a new key to the heap
e) merge - joining two heaps to form a valid new heap containing all the elements of both

85: Enumerate several heap specializations.
Answer:
2-3 heap, Beap, Binary heap, Binomial heap, Brodal Queue, D-ary heap, Fibonacci heap, Leftist heap, Pairing heap, Pseudo Queue, Skew heap, Soft heap, Ternary heap, Treap.

86: What is a Fibonacci heap?
Answer:
A Fibonacci heap is a collection of trees satisfying the minimum-heap property, that is, the key of a child is always greater than or equal to the key of the parent. This implies that the minimum key is always at the root of one of the trees.

87: Provide a short description of a B-tree.
Answer:
A B-tree is a generalization of a binary search tree which keeps data sorted and allows searches, sequential access, insertions, and deletions in logarithmic amortized time. It is commonly used in databases and filesystems.

88: What is a search tree?
Answer:
A search tree is a tree which supports efficient search, insertion and withdrawal operations. In this context the tree is used to store a finite set of keys inserted after a data ordering criterion. No duplicate keys are allowed.

89: Give example of some types of search trees.
Answer:
AVL Search Trees, M-Way Search Trees, B-Trees.

90: What is a binary search tree?

Answer:
A binary search tree is a finite set of keys which satisfies the following properties: a) all the keys contained in left subtree are less than the value of the subtree root
b) all the keys contained in the right subtree are greater than the value of the subtree root

91: Describe the algorithm of inserting data into a binary search tree.
Answer:
In order to insert data into a binary search tree you have to perform the following steps: a) pretend that the item is already in the tree and follow the path taken by the Find routine to determine where the item would be.
b) assuming that the item is not already in the tree, the search will be unsuccessful and will terminate with an external, empty node which is precisely where the item to be inserted is placed.

92: Describe the procedure of removing an item from a binary search tree.
Answer:
When removing an item from a search tree, it is imperative that the tree which remains satisfies the data ordering criterion.
a) to remove a non-leaf node, it must be moved down in the tree until it becomes a leaf node since a leaf node is easily deleted b) to move a non-leaf node down in the tree, we either swap it with the smallest key in the right subtree or with the largest one in the left subtree.

93: Define the concept of a treap.
Answer:
A treap is a form of binary search tree data structure that maintains a dynamic set of ordered keys and allows binary searches among the keys. After any sequence of insertions and deletions of keys, the shape of the tree is a random variable with the same probability distribution as a random binary tree; in particular, with high probability its height is proportional to the logarithm of the number of keys, so that each search, insertion, or deletion operation takes logarithmic time to perform.

94: What is an AVL search tree?
Answer:

An AVL tree is a binary search tree such that for every internal node v of T, the heights of the children of v can differ by at most 1. The basic operations of an AVL tree involve carrying out the same actions as would be carried out on an unbalanced binary search tree (search, insertion, deletion, traversal, sort), but modifications are preceded or followed by one or more operations called tree rotations, which help to restore the height balance of the subtrees.

95: Define a self-balancing binary search tree.
Answer:
A self-balancing binary search tree is any node based binary search tree that automatically keeps its height (number of levels below the root) small in the face of arbitrary item insertions and deletions. They provide efficient implementations for mutable ordered lists, and can be used for other abstract data structures such as associative arrays, priority queues and sets.

96: What is a red-black tree?
Answer:
A red-black tree is a binary search tree where each node has a color attribute, the value of which is either red or black. In addition to the ordinary requirements imposed on binary search trees, the following additional requirements apply to red-black trees: a) a node is either red or black
b) the root is black
c) all leaves are black
d) both children of every red node are black
e) every simple path from a given node to any of its descendant leaves contains the same number of black nodes **97: What is the usage of red-black trees?**
Answer:
Red-black trees offer worst-case guarantees for insertion, deletion, and search time. This feature makes them valuable in time-sensitive applications such as real-time applications; for example, many data structures used in computational geometry can be based on red-black trees (e.g. the Completely Fair Scheduler used in current Linux kernels uses red-black trees).

98: Make a short comparison between AVL trees and red-black trees.
Answer:
AVL trees are more rigidly balanced than red-black ones; this leads to slower insertion and removal but faster retrieval. The differences are anyway very small

and notable only when using extremely large data structures that may be built once and loaded without reconstruction (e.g. language dictionaries).

99: Describe a splay tree.
Answer:
A splay tree is a self-adjusting binary search tree with the additional property that recently accessed elements are quick to access again. It performs basic operations such as insertion, look-up and removal in O(log n) amortized time. For many sequences of operations, splay trees perform better than other search trees, even when the specific pattern of the sequence is unknown.

100: What are the characteristics of a good balance condition?
Answer:
A good balance condition:
a) ensures that the height of a tree with n nodes is O(log n)
b) can be maintained efficiently (i.e. the additional work necessary to balance the tree when an item is inserted or deleted is O(1)).

101: Describe the steps that need to be performed while inserting an item into an AVL tree.
Answer:
Inserting an item into an AVL tree is a two-part process. First, the item is inserted into the tree using the usual method for insertion in binary search trees. After the item has been inserted, it is necessary to check that the resulting tree is still AVL balanced and to balance the tree when it is not.

102: What is a M-way search tree?
Answer:
A M-way search tree is a generalization of a binary search tree, which has (M-1) values per node and M subtrees. M is called the degree of the tree. Thus, a binary search tree can be seen as an M-way search tree of degree 2.

103: Which are the algorithms for finding items in an M-way search tree?
Answer:
There are two algorithms for finding items in an M-way search tree: the first is a naive implementation using linear search and the second version improves upon the first by using a binary search.

104: What is a B-Tree?

Answer:

A B-tree is a balanced M-way search tree T. It has the following properties: a) the root of T has at least two subtrees and at most M subtrees

b) all internal nodes of T (other than its root) have between 2 and M subtrees

c) all external nodes of T are at the same level

105: What is a perfect binary tree?

Answer:

A perfect binary tree is a binary tree with all leaf nodes at the same depth. All the internal binary tree nodes have the degree 2. It is also called a complete binary tree.

106: What is a binary heap?

Answer:

A binary heap is a heap data structure created using a binary tree. It can be seen as a binary tree with two additional constraints: a) the shape property: the tree is a complete binary tree; that is, all levels of the tree, except possibly the last one (deepest) are fully filled, and, if the last level of the tree is not complete, the nodes of that level are filled from left to right b) the heap property: each node is greater than or equal to each of its children according to some comparison predicate which is fixed for the entire data structure **107: Define the term of leftist tree.**

Answer:

A leftist tree is a tree which tends to 'lean' to the left. The tendency to lean to the left is defined in terms of the shortest path from the root to an external node. In a leftist tree, the shortest path to an external node is always found on the right.

108: What is a binomial tree?

Answer:

The binomial tree of order k>=0 with root R is the tree Bk defined as follows: a) if k = 0, Bk = B0 = {R} (i.e. the binomial tree of order zero consists of a single node, R) b) if k > 0, Bk = {R, B0, B1, ... Bk-1} (i.e., the binomial tree of order k>0 comprises the root R, and k binomial subtrees B0, B1, ... , Bk-1) **109: Define a T-tree and describe its usage.**

Answer:

A T-tree is a balanced index tree data structure optimized for cases where both the index and the actual data are fully kept in memory. T-trees seek to gain the performance benefits of in-memory tree structures such as AVL trees while avoiding the large storage space overhead which is common to them. They do not keep copies of the indexed data fields within the index tree nodes themselves but instead, they take advantage of the fact that the actual data is always in main memory together with the index so that they just contain pointers to the actual data fields.

110: What is a weight-balanced binary tree?
Answer:
A weight-balanced binary tree is a binary tree which is balanced based on knowledge of the probabilities of searching for each individual node. Within each subtree, the node with the highest weight appears at the root. This can result in more efficient searching performance.

111: Describe a cartesian tree.
Answer:
A Cartesian tree is a binary tree derived from a sequence of numbers; it can be uniquely defined from the properties that it is heap-ordered and that a symmetric (inorder) traversal of the tree returns the original sequence. It was introduced by Vuillemin in the context of geometric range searching data structures.

112: Describe on short the Day/Stout/Warren algorithm.
Answer:
The DSW algorithm is a method for efficiently balancing binary search trees — that is, decreasing their height to O(log n) nodes, where n is the total number of nodes. Unlike a self-balancing binary search tree, it does not do this incrementally during each operation, but periodically, so that's why its cost can be amortized over many operations.

113: Define a BSP tree and describe its usage.
Answer:
A Binary Space Partitioning (BSP) tree represents a recursive,
hierarchical partitioning, or subdivision, of n-dimensional space into convex subspaces. BSP tree construction is a process which takes a subspace and partitions it by any hyperplane that intersects the interior of that subspace. The

result is two new subspaces that can be further partitioned by recursive application of the method.

114: What is a trie?
Answer:
A trie, or prefix tree, is an ordered tree data structure that is used to store an associative array where the keys are usually strings. Unlike a binary search tree, no node in the tree stores the key associated with that node; instead, its position in the tree shows what key it is associated with.

115: Which are the main advantages of tries over binary search trees?
Answer:
Below are described several advantages of tries over binary
search trees:
a) looking up keys is faster. Also, the simple operations tries use during lookup, such as array indexing using a character, are fast on real machines b) tries can require less space when they contain a large number of short strings, because the keys are not stored explicitly and nodes are shared between keys with common initial subsequences c) tries facilitate longest-prefix matching, helping to find the key sharing the longest possible prefix of characters all unique **116: Which are the main advantages of tries over hash tables?**
Answer:
The following are the main advantages of tries over hash tables:
a) tries can perform a 'closest fit' find almost as quickly as an exact find. Hash tables can only perform an exact find as they store no relational state between keys b) tries tend to be faster on average at insertion than hash tables because hash tables must rebuild their index when it becomes full - which is a very expensive operation. Tries therefore have much better bounded worst case time costs which is important for latency sensitive code c) tries can be implemented in a way which avoids the need for additional (dynamic) memory *i.e.* an in-place implementation. Hash tables must always have additional memory in which to store the hash indexation table d) looking up keys can be much faster if a hash function can be avoided
e) tries support longest-prefix matching but hashing does not

117: What are the bitwise tries?
Answer:

Bitwise tries are much the same as a normal character based trie except that individual bits are used to traverse what effectively becomes a form of binary tree. They can handle standard IEEE single and double format floating point numbers.

118: What are ternary search trees?
Answer:
A ternary search tree is a ternary (three-way) tree data structure of strings which combines the speed of a prefix search tree, or trie, with the space efficiency of a binary search tree. They replace each node of the trie with a modified binary search tree. For sparse tries, this binary tree will be smaller than a trie node.

119: What is a binomial queue?
Answer:
A binomial queue is a set of power-of-2 heaps. Its structure is determined by that queue's number of nodes, by correspondence with the binary representation of integers. The remarkable characteristic of binomial queues is that the merge operation is similar in structure to binary addition (e.g. the collection of binomial trees that make up the binomial queue is like the set of bits that make up the binary representation of a non-negative integer). Furthermore, the merging of two binomial queues is done by adding the binomial trees that make up that queue in the same way that the bits are combined when adding two binary numbers.

*****

Sets

120: What is a set?
Answer:
A set is an abstract data structure that can store certain values, without any particular order, and no repeated values. It is a computer implementation of the mathematical concept of a finite set. In C++, the Standard Template Library (STL) provides the set template class, which implements a sorted set using a binary search tree.

121: What are the most common operations on sets? Provide a short description.

Answer:
Below are described the most common operations for combining sets:
a) union - the union of two sets is a superset containing all the elements existing in each of the original sets b) intersection - the intersection (or disjunction) of two sets contains all the common elements
c) difference - the difference (or subtraction) of two sets S and T contains those elements of S which are not also elements of T

122: What is a multiset? Define the cardinality of a multiset.
Answer:
A multiset is a set in which an item may appear more than once (it allows duplicates). They are also known as bags. The cardinality of a multiset defines the total number of elements in a multiset, including repeated memberships.

123: What is a disjoint-set data structure?
Answer:
In computing, given a set of elements, it is often useful to break them up or partition them into a number of separate, non overlapping sets. A disjoint-set data structure is a data structure that keeps track of such a partitioning.

124: What are the disjoint-set forests?
Answer:
Disjoint-set forests are data structures where each set is represented by a tree data structure and each node holds a reference to its parent. In a disjoint-set forest, the representative of each set is the root of that set's tree. One way of implementing these might be: *function MakeSet(x)*

```
  x.parent = x

function Find(x)
  if x.parent == x
  return x
  else
  return Find(x.parent)
function Union(x, y)
  xRoot = Find(x)
  yRoot = Find(y)
  xRoot.parent = yRoot
```

*****

Graphs

125: What is a graph?

Answer:

A graph data structure consists mainly of a finite (and possibly mutable) set of ordered pairs, called edges or arcs, of certain entities called nodes or vertices. As in mathematics, an edge (x,y) is said to point or go from x to y. The nodes may be part of the graph structure, or may be external entities represented by integer indices or references.

126: What is the relationship between trees and graphs?

Answer:

In graph theory, a tree is a connected acyclic graph.

127: What is a hypergraph?

Answer:

A hypergraph is a generalization of a graph, where an edge can connect any number of vertices. It is also called a set system or a family of sets drawn from the universal set X. Hypergraphs can be viewed as incidence structures and vice versa.

128: Give example of data structures used for graphs representation:

Answer:

The data structures used for graph representation are described below: a) adjacency list - it is implemented as an array of lists, with one list of destination nodes for each source node b) incidence list - a variant of the adjacency list that allows for the description of the edges at the cost of additional edges c) adjacency matrix - a two-dimensional Boolean matrix, where the rows and columns represent source and destination vertices and entries in the matrix indicate whether an edge exists between the vertices associated with that row and column d) incidence matrix - a two-dimensional Boolean matrix where the rows represent the vertices and columns represent the edges. The array entries indicate if both are related (incident) **129: What is a dense graph?**

Answer:

A dense graph is a graph in which the number of edges is close to the maximal number of edges.

130: What is a sparse graph?

Answer:
A sparse graph is a graph with only a few edges. It is the opposite of a dense graph.

131: Define a complete graph.
Answer:
A complete graph is a simple graph in which every pair of distinct vertices are connected by a unique edge.

132: What is a circular graph?
Answer:
A circular graph is a graph that consists of a single cycle, or in other words, some number of vertices connected in a closed chain.

133: Define a bipartite graph.
Answer:
A bipartite graph (or bigraph) is a graph whose vertices can be divided into two disjoint sets U and V such that every edge connects a vertex in U to one in V; that is, U and V are independent sets.

134: Describe several types of a cycle graphs.
Answer:
A cycle graph may be:
a) connected - every pair of vertices in the graph is connected
b) 2-regular - each vertex has the same number of neighbors
c) Eulerian - each edge is used only once
d) Hamiltonian - each vertex is visited exactly once
e) 2-vertex colorable - if and only if it has an even number of vertices f) 2-edge colorable - if and only if it has an even number of vertices g) 3-vertex colorable and 3-edge colorable - for any number of vertices h) a unit distance graph - a collection of points where two points are connected by an edge whenever the distance between the two points is exactly one **135: What is the degree of a vertex?**
Answer:
The degree of a vertex is the sum of its out-and indegrees. The outdegree of a node is the number of edges leaving from that node, and the indegree is the number of edges entering that node.

136: What is a dual graph?
Answer:
In the dual graph the edges represent the activities, and the vertices represent the commencement and termination of activities. For this reason, the dual graph is called an event-node graph.

137: What is a Hamiltonian cycle?
Answer:
A Hamiltonian cycle is a cycle in an undirected graph which visits each vertex exactly once and also returns to the starting vertex.

138: What is an Eulerian path?
Answer:
An Eulerian path in an undirected graph is a path that uses each edge exactly once. If such a path exists, the graph is called traversable or semi-eulerian.

139: Define a planar graph.
Answer:
A planar graph is a graph that can be drawn in the plane with no crossing edges. Drawing of a planar graph G is called planar embedding of G.

140: What is a k-connected graph?
Answer:
A k-connected graph is a graph which has the property that when deleting any k-1 vertices (and incident edges) the resulting graph is still connected.

141: Enumerate several graph traversal algorithms:
Answer:
There are three well known graph traversal algorithms:
a) depth-first traversal
b) breadth-first traversal
c) topological sort

142: Describe the depth-first graph traversal algorithm.
Answer:
Depth-first traversal algorithm of a graph starts with a given vertex and then

recursively visits all the vertices adjacent to that node. A depth-first traversal only follows edges that lead to unvisited vertices.

143: Describe the breadth-first graph traversal algorithm.
Answer:
Breadth-first graph traversal algorithm first visits the starting vertex, then all the vertices adjacent to the starting vertex, and the all the vertices adjacent to those, and so on.

144: What is a topological sort?
Answer:
A topological sort is a permutation p of the vertices of a graph such that an edge {i, j} implies that i appears before j in p. Only directed acyclic graphs can be topologically sorted.

145: What is a weighted graph?
Answer:
A weighted graph associates a label (weight) with every edge in the graph. Weights are usually real numbers but they may be restricted to rational numbers or integers, depending on the algorithm they are used for.

146: Describe the shortest path problem in graph theory.
Answer:
In graph theory, the shortest path problem is the problem of finding a path between two vertices (or nodes) such that the sum of the weights of its constituent edges is minimized. It is also known as single-pair shortest path problem.

147: Give examples of generalizations of single-pair shortest path problem.
Answer:
Below are described several generalizations of shortest path problem: a) single-source shortest path problem - in which we have to find shortest paths from a source vertex v to all other vertices in the graph b) single-destination shortest path problem - in which we have to find shortest paths from all vertices in the graph to a single destination vertex v. This can be reduced to the single-source shortest path problem by reversing the edges in the graph c) all-pairs shortest path problem - in which we have to find shortest paths between every pair of

vertices v, v' in the graph **148: Enumerate some algorithms designed to solve the shortest path problem.**

Answer:

There are several algorithms designed to minimize the shortest path cost a) Dijkstra's algorithm - solves the single-pair, single-source, and single-destination shortest path problems b) Bellman-Ford algorithm - solves the single source problem if edge weights may be negative c) A* search algorithm - solves for single pair shortest path using heuristics to try to speed up the search d) Floyd—Warshall algorithm - solves all pairs shortest paths

e) Johnson's algorithm - solves all pairs shortest paths, and may be faster than Floyd— Warshall on sparse graphs

f) Perturbation theory - finds (at worst) the locally shortest path

149: What is Kruskal's algorithm?

Answer:

Kruskal's algorithm is an algorithm in graph theory that finds a minimum spanning tree for a connected weighted graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. If the graph is not connected, then it finds a minimum spanning forest (a minimum spanning tree for each connected component).

150: Describe Dijkstra's algorithm.

Answer:

It is a greedy algorithm for solving the single-source, shortest-path problem on an edge-weighted graph in which all the weights are non-negative. It finds the shortest paths from some initial vertex to all the other vertices one-by-one. The essential feature of Dijkstra's algorithm is the order in which the paths are determined: The paths are discovered in the order of their weighted lengths, starting with the shortest, and proceeding to the longest.

151: Provide a short description of Floyd's algorithm.

Answer:

Floyd's algorithm uses the dynamic programming method to solve the all-pairs shortest-path problem on a dense graph. The method makes efficient use of an adjacency matrix to solve the problem.

152: What is the purpose of critical path analysis?
Answer:
Critical path analysis answers the following questions:
a) what is the minimum amount of time needed to complete all activities?
b) for a given activity v, is it possible to delay the completion of that activity without affecting the overall completion time? If yes, by how much can the completion of activity v be delayed?

153: Describe Christofides algorithm.
Answer:
It is a heuristic algorithm to find a near-optimal solution to the traveling salesman problem. It contains of the following steps: a) find a minimum spanning tree T
b) find a perfect matching M among vertices with odd degree
c) combine the edges of M and T to make a multigraph G
d) find an Euler cycle in G by skipping vertices already seen.

154: What is a knot (in a directed graph)?
Answer:
A knot in a directed graph is a collection of vertices and edges with the property that every vertex in the knot has outgoing edges, and all outgoing edges from vertices in the knot terminate at other vertices in the knot. Thus it is impossible to leave the knot while following the directions of the edges.

155: What is the complement of a graph? Give example of several graph-theoretic concepts which are related to each other via complement graphs:
Answer:
The complement (or inverse) of a graph G is a graph H on the same vertices such that two vertices of H are adjacent if and only if they are not adjacent in G. Below are some concepts related to each other via complement graphs: a) the complement of an edgeless graph is a complete graph and vice versa
b) an independent set in a graph is a clique in the complement graph and vice versa
c) the complement of any triangle-free graph is a claw-free graph
d) a self-complementary graph is a graph that is isomorphic to its own complement
e) cographs are defined as the graphs that can be built up from disjoint union and

complementation operations, and form a self-complementary family of graphs: the complement of any cograph is another (possibly different) cograph

156: Define a dipole graph.
Answer:
A dipole graph is a multigraph consisting of two vertices connected with a number of edges. A dipole graph containing n edges is called the order-n dipole graph, and is denoted by Dn. The order-n dipole graph is dual to the cycle graph Cn.

157: Describe the terms: regular graph, regular directed graph and k-regular graph.
Answer:
A regular graph is a graph without loops and multiple edges where each vertex has the same number of neighbors. A regular directed graph must also satisfy the stronger condition that the indegree and outdegree of each vertex are equal to each other. A regular graph with vertices of degree k is called a k-regular graph or regular graph of degree k.

158: What is a strongly regular graph?
Answer:
A strongly regular graph is a regular graph where every adjacent pair of vertices has the same number of neighbors in common, and every non-adjacent pair of vertices has the same number n of neighbors in common.

159: Which are the smallest graphs that are regular but not strongly regular?
Answer:
The smallest graphs that are regular but not strongly regular are the cycle graph and the circulant graph on 6 vertices.

160: What is a Moore graph?
Answer:
A Moore graph is a connected graph which has the maximal length of the shortest cycle 2d+1 for its diameter d.

*****

Algorithms

Description: Questions in this category are related to Algorithms

Topics:

General Concepts Of Algorithms
Sorting Algorithms
Search Algorithms
Huffman Coding

*****

General Concepts Of Algorithms

161: Define the concept of an algorithm.

Answer:

An algorithm is any well-defined computational procedure that takes some value (or set of values) as input and produces some value (or set of values) as output. In short, it can be seen as a sequence of computational steps that transform the input into the output.

162: Define a brute-force algorithm. Give a short example.

Answer:

A brute force algorithm is a type of algorithm that proceeds in a simple and obvious way, but requires a huge number of steps to complete. As an example, if you want to find out the factors of a given number N, using this sort of algorithm will require to get one by one all the possible number combinations.

163: What is a greedy algorithm? Give examples of problems solved using greedy algorithms.

Answer:

A greedy algorithm is any algorithm that makes the local optimal choice at each stage with the hope of finding the global optimum. A classical problem which can be solved using a greedy strategy is the traveling salesman problem. Another problems that can be solved using greedy algorithms are the graph coloring problem and all the NP-complete problems.

164: Which are the pillars of a greedy algorithm?

Answer:

In general, greedy algorithms have five pillars:
a) a candidate set, from which a solution is created
b) a selection function, which chooses the best candidate to be added to the solution
c) a feasibility function, that is used to determine if a candidate can be used to contribute to a solution d) an objective function, which assigns a value to a solution, or a partial solution
e) a solution function, which will indicate when we have discovered a complete solution

165: What is a backtracking algorithm? Provide several examples.
Answer:
It is an algorithm that considers systematically all possible outcomes for each decision. Examples of backtracking algorithms are the eight queens problem or generating permutations of a given sequence.

166: What is the difference between a backtracking algorithm and a brute-force one?
Answer:
Due to the fact that a backtracking algorithm takes all the possible outcomes for a decision, it is similar from this point of view with the brute force algorithm. The difference consists in the fact that sometimes a backtracking algorithm can detect that an exhaustive search is unnecessary and, therefore, it can perform much better.

167: Define an iterator. Provide usage example of an iterator.
Answer:
An iterator provides a means by which the objects within a container can be accessed one-at-a-time. All the iterators share a common interface, and hide the underlying implementation of the container from the user of that container. The common operations that can be executed using iterators are read, write, be incremented/decremented. Below is a C++ example for using an iterator:
std::vector<int> the_vector; //define a vector
std::vector<int>::iterator the_iterator = the_vector.begin(); //start the iterator
while(the_iterator != the_vector.end())
++the_iterator; //move until the end of the vector **168: Describe the visitor design pattern.**
Answer:
The Visitor design pattern allows behavior addition to a composite structure

without changing the existing class definitions. Visitors reduce the number of operations directly embedded within a class, thus preventing class definitions from becoming cluttered. A C++ code example is described below: *class Equipment{*

```
public:
  virtual Currency NetPrice();
  virtual void Accept (EquipmentVisitor&); };
class EquipmentVisitor{
public:
  virtual void VisitCard(Card*);
  ...
```

}

```
void Card::Accept(EquipmentVisitor& visitor){visitor.VisitCard(this);}
```

The Equipment classes must have an Accept method defined to allow a visitor to be activated for instances of the class. The EquipmentVisitor class must have a visit method defined for every concrete class for which a visit operation may occur. Each concrete subclass of Equipment must define the Accept method. The Accept method for basic equipment subclasses such as Card simply activates the visitor, passing the current equipment object as an argument to the visit method (using the self reference).

169: What is an adapter?

Answer:

An adapter converts the interface of one class into the interface expected by the user of that class. This allows a given class with an incompatible interface to be used in a situation where a different interface is expected. A classical case of using adaptors is when there is a legacy interface which is not compatible with the system but it has to be reused. An abstract base class is created that specifies the desired interface. An adapter class is defined that publicly inherits the interface of the abstract class, and privately inherits the implementation of the legacy component. This adapter class 'maps' the new interface to the old implementation.

170: What is a singleton?

Answer:

A singleton is a class which allows instantiating of only one instance. It also provides a way to access that instance. It can be implemented in C++ in the

following way: *class Singleton{*

```
public:
  static Singleton& Instance(){
  static Singleton singleton;
  return singleton;

                                                    }

private:
  Singleton() {};     // Private constructor
  Singleton(const Singleton&);     // Prevent copy-construction
  Singleton& operator=(const Singleton&);     // Prevent assignment

                                                    };
```

171: Describe on short the branch and bound algorithm.

Answer:

Branch and bound is a general algorithm for finding optimal solutions of various optimization problems, especially in discrete and combinatorial optimization. It consists of a systematic enumeration of all candidate solutions, where large subsets of fruitless candidates are discarded en masse, by using upper and lower estimated bounds of the quantity being optimized.

172: Describe divide and conquer paradigm.

Answer:

When a problem is solved using a divide and conquer algorithm, it is subdivided into one or more subproblems which are all similar to the original problem in such a way that each of the subproblems can be solved independently. In the end, the solutions to the subproblems are combined in order to obtain the solution to the original problem.

173: Describe dynamic programming paradigm.

Answer:

When a specific problem is solved using a dynamic programming algorithm, it solves first a series of subproblems The subproblems are devised carefully in such a way that each subsequent solution is obtained by combining the solutions to one or more of the subproblems that have already been solved. All

intermediate solutions are kept in a table in order to prevent unnecessary duplication of effort.

174: How does a simulated annealing algorithm work?
Answer:
A simulated annealing algorithm moves about randomly in the solution space looking for a solution that minimizes the value of some objective function. Because it is generated randomly, a given move may cause the objective function to increase, to decrease or to remain unchanged.

175: Provide a short description of Euclid's algorithm. Provide its recursive description: Answer:
It is an algorithm which computes the greatest common divisor of two positive integers. It may be implemented recursively in the following way: *Euclid(a,b) =*
if (b=0) then return a;
else return Euclid(b, a mod b); The run time complexity is O((log a)(log b)).

*****

Sorting Algorithms

176: Which are the sorting algorithms categories?
Answer:
Sorting algorithms can be divided into five categories:
a) insertion sorts
b) exchange sorts
c) selection sorts
d) merge sorts
e) distribution sorts

177: Describe on short an insertion sorting algorithm.
Answer:
An algorithm that sorts by insertion takes the initial, unsorted sequence and computes a series of sorted sequences using the following rules: a) the first sequence in the series is the empty sequence
b) given a sequence S(i) in the series, for 0<=i<n, the next sequence in the series, S(i+1), is obtained by inserting the (i+1)th element of the unsorted sequence S(i+1) into the correct position in S(i).

178: Which are the advantages provided by insertion sort?
Answer:
Insertion sort provides several advantages:
a) simple implementation
b) efficient for small data sets
c) adaptive - efficient for data sets that are already substantially sorted: the time complexity is O(n + d), where d is the number of inversions d) more efficient in practice than most other simple quadratic, *i.e.* O(n2) algorithms such as selection sort or bubble sort; the best case (nearly sorted input) is O(n) e) stable - does not change the relative order of elements with equal keys
f) in-place - only requires a constant amount O(1) of additional memory space
g) online - can sort a list as it receives it

179: What search algorithm does straight insertion sorting algorithm use?
Answer:
Straight insertion sorting uses a linear search algorithm to locate the position at which the next element is to be inserted.

180: What is the difference between insertion and exchange sorting?
Answer:
In insertion sorting algorithms, insertion is performed into a sorted list. On the other hand, an exchange sort does not necessarily make use of such a sorted list.

181: Give examples of exchange sorting algorithms:
Answer:
The most well known exchange sorting algorithms are the following:
a) Bubble Sort - simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order b) Quicksort - a divide-and-conquer style algorithm which divides the original list into two sub-lists and sorts recursively each list **182: Shortly describe the quicksort algorithm.**
Answer:
In quicksort, the steps performed are the following:
a) pick an element, called a pivot, from the list
b) reorder the list so that all elements with values less than the pivot come before the pivot, while all elements with values greater than the pivot come after it

(equal values can go either way) c) recursively sort the sub-list of lesser elements and the sub-list of greater elements

183: Describe several variations of quicksort algorithm.
Answer:
Below is a short description of three well known variants of
quicksort:
a) balanced quicksort - choose a pivot likely to represent the middle of the values to be sorted, and then follow the regular quicksort algorithm b) external quicksort - the same as regular quicksort except the pivot is replaced by a buffer
c) three-way radix quicksort (also called multikey quicksort) - a combination of radix sort and quicksort

184: What is the difference between selection and insertion sorting?
Answer:
In insertion sorting elements are added to the sorted sequence in an arbitrary order. In selection sorting, the elements are added to the sorted sequence in order so they are always added at one end.

185: What is merge sorting?
Answer:
Merging is the sorting algorithm which combines two or more sorted sequences into a single sorted sequence. It is a divide and conquer algorithm, an O(n log n) comparison-based sorting algorithm. Most implementations produce a stable sort, meaning that the implementation preserves the input order of equal elements in the sorted output.

186: Which are the main steps of a merge sorting algorithm?
Answer:
Sorting by merging is a recursive, divide-and-conquer strategy. The basic steps to perform are the following: a) divide the sequence into two sequences of length
b) recursively sort each of the two subsequences
c) merge the sorted subsequences to obtain the final result

187: What is distribution sorting? Give example of several distribution sorting algorithms.
Answer:

It is a sorting algorithm which has the unique characteristic that it does not make use of comparisons to do the sorting. Instead, distribution sorting algorithms rely on a priori knowledge about the universal set from which the elements to be sorted are drawn. The most well-known distribution sorting algorithms are bucket sort and radix sort.

188: Describe the adaptive heap sort algorithm.
Answer:
It is a variant of heapsort that uses a randomized binary search tree to structure the input according to any preexisting order. The randomized binary search tree is used to select candidates that are put into the heap so that the heap doesn't need to keep track of all elements.

189: Describe the counting sort algorithm.
Answer:
It is a 2-pass sort algorithm that is efficient when the range of keys is small and there many duplicate keys. The first pass counts the occurrences of each key in an auxiliary array, and then makes a running total so each auxiliary entry is the number of preceding keys. The second pass puts each item in its final place according to the auxiliary entry for that key.

190: Describe Burstsort algorithm.
Answer:
Burstsort algorithm is a cache-efficient algorithms for sorting strings and are faster than quicksort and radix sort for large data sets. It tries to store prefixes of strings, with growable arrays of pointers as end nodes containing sorted, unique, suffixes (referred to as buckets). Some variants copy the string tails into the buckets. As the buckets grow beyond a predetermined threshold, the buckets are "burst", giving the sort its name.

*****

Search Algorithms

191: Describe the interpolation search algorithm.
Answer:
Interpolation search is an algorithm for searching for a given key value in an indexed array that has been ordered by the keys values. In each search step it

calculates where in the remaining search space the sought item might be, based on the key values at the bounds of the search space and the value of the sought key, usually via a linear interpolation. The key value actually found at this estimated position is then compared to the key value being sought. If it is not equal, then depending on the comparison, the remaining search space is reduced to the part before or after the estimated position.

192: Provide a short description of binary search algorithm.
Answer:
Binary search algorithm always chooses the middle of the remaining search space, discarding one half or the other, again depending on the comparison between the key value found at the estimated position and the key value sought. The remaining search space is reduced to the part before or after the estimated position.

193: What is a ternary search algorithm?
Answer:
A ternary search algorithm is a technique used in computer science for finding the minimum or maximum of a unimodal function (function that is either strictly increasing and then strictly decreasing or vice versa). It determines either that the minimum or maximum cannot be in the first third of the domain or that it cannot be in the last third of the domain, then repeats on the remaining two-thirds. It is a good example of a divide and conquer algorithm.

194: Describe golden section search algorithm.
Answer:
The golden section search is a technique for finding the extremum (minimum or maximum) of a unimodal function by successively narrowing the range of values inside which the extremum is known to exist. The technique derives its name from the fact that the algorithm maintains the function values for triples of points whose distances form a golden ratio. The algorithm is closely related to Fibonacci and binary search.

195: Describe Fibonacci search technique.
Answer:
The Fibonacci search technique is a method of searching a sorted array using a divide and conquer algorithm that narrows down possible locations with the aid

of Fibonacci numbers. Compared to binary search, Fibonacci search examines locations whose addresses have lower dispersion. Therefore, when the elements being searched have non-uniform access memory storage (i.e., the time needed to access a storage location varies depending on the location previously accessed), the Fibonacci search has an advantage over binary search in slightly reducing the average time needed to access a storage location.

196: What is the linear search algorithm?
Answer:
Linear search is a method for finding a particular value in a list which consists of checking every one of its elements, one at a time and in sequence, until the desired one is found. It is the simplest search algorithm, a special case of brute-force search. Its worst case cost is proportional to the number of elements in the list; and so is its expected cost, if all list elements are equally likely to be searched for. Therefore, if the list has more than a few elements, other methods (such as binary search or hashing) may be much more efficient.

197: Describe secant search algorithm.
Answer:
It is an algorithm that searches a sorted array by estimating the next position to check based on the values at the two previous positions checked. It is called "secant search" because it uses the secant of the function at two successive points to approximate the derivative in the Newton-Raphson formula.

198: What is best-first search algorithm?
Answer:
It is a search algorithm that considers the estimated best partial solution next. This is typically implemented with priority queues.

*****

Huffman Coding

199: What is Huffman coding?
Answer:
In computer science and information theory, Huffman coding is an entropy encoding algorithm used for lossless data compression. The term refers to the use of a variable-length code table for encoding a source symbol (such as a

character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol.

200: Give several examples of Huffman coding applications.
Answer:
Huffman coding today is often used as a "back-end" to some other compression methods. DEFLATE (PKZIP's algorithm) and multimedia codecs such as JPEG and MP3 have a front-end model and quantization followed by Huffman coding.

*****

HR Questions

Review these typical interview questions and think about how you would answer them. Read the answers listed; you will find best possible answers along with strate gies and suggestions .

*****

1: Tell me about yourself?
Answer:
The most often asked question in interviews. You need to have a short statement prepared in your mind. Keep your answer to one or two minutes. Don’t ramble. Be careful that it does not sound rehearsed. Limit it to work-related items unless instructed otherwise. Talk about things you have done and jobs you have held that relate to the position you are interviewing for. Start with the item farthest back and work up to the present (If you have a profile or personal statement(s) at the top of your CV use this as your starting point).

2: Why did you leave your last job?
Answer:
Stay positive regardless of the circumstances. Never refer to a major problem with management and never speak ill of supervisors, co-workers or the organization. If you do, you will be the one looking bad. Keep smiling and talk about leaving for a positive reason such as an opportunity, a chance to do something special or other forward-looking reasons.

3: What experience do you have in this field?
Answer:
Speak about specifics that relate to the position you are applying for. If you do not have specific experience, get as close as you can.

4: Do you consider yourself successful?
Answer:
You should always answer yes and briefly explain why. A good explanation is that you have set goals, and you have met some and are on track to achieve the others.

5: What do co-workers say about you?
Answer:
Be prepared with a quote or two from co-workers. Either a specific statement or a paraphrase will work. Bill Smith, a co-worker at Clarke Company, always said I was the hardest worker's he had ever known. It should be as powerful as Bill having said it at the interview herself.

6: What do you know about this organization?
Answer:
This question is one reason to do some research on the organization before the interview. Research the company's products, size, reputation, Image, goals, problems, management style, skills, History and philosophy. Be informed and interested. Find out where they have been and where they are going. What are the current issues and who are the major players?

7: What have you done to improve your knowledge in the last year?
Answer:
Try to include improvement activities that relate to the job. A wide variety of activities can be mentioned as positive self-improvement. Have some good ones handy to mention.

8: Are you applying for other jobs?
Answer:
Be honest but do not spend a lot of time in this area. Keep the focus on this job and what you can do for this organization. Anything else is a distraction.

9: Why do you want to work for this organization?
Answer:
This may take some thought and certainly, should be based on the research you have done on the organization. Sincerity is extremely important here and will easily be sensed. Relate it to your long-term career goals. Never talk about what you want; first talk about their Needs. You want to be part of an exciting forward-moving company. You can make a definite contribution to specific company goals.

10: Do you know anyone who works for us?
Answer:
Be aware of the policy on relatives working for the organization. This can affect your answer even though they asked about friends not relatives. Be careful to mention a friend only if they are well thought of.

11: What kind of salary do you need?
Answer:
A loaded question! A nasty little game that you will probably lose if you answer first. So, do not answer it. Instead, say something like, that’s a tough question. Can you tell me the range for this position? In most cases, the interviewer, taken off guard, will tell you. If not, say that it can depend on the details of the job. Then give a wide range.

12: Are you a team player?
Answer:
You are, of course, a team player. Be sure to have examples ready. Specifics that show you often perform for the good of the team rather than for yourself is good evidence of your team attitude. Do not brag; just say it in a matter-of-fact tone. This is a key point.

13: How long would you expect to work for us if hired?
Answer:
Specifics here are not good. Something like this should work: I'd like it to be a long time. Or As long as we both feel I'm doing a good job.

14: Have you ever had to fire anyone? How did you feel about that?
Answer:

This is serious. Do not make light of it or in any way seem like you like to fire people. At the same time, you will do it when it is the right thing to do. When it comes to the organization versus the individual who has created a harmful situation, you will protect the organization. Remember firing is not the same as layoff or reduction in force.

15: What is your philosophy towards work?
Answer:
The interviewer is not looking for a long or flowery dissertation here. Do you have strong feelings that the job gets done? Yes. That's the type of answer that works best here. Keep it short and positive, showing a benefit to the organization.

16: If you had enough money to retire right now, would you?
Answer:
Answer yes if you would. But since you need to work, this is the type of work you prefer. Do not say yes if you do not mean it.

17: Have you ever been asked to leave a position?
Answer:
If you have not, say no. If you have, be honest, brief and avoid saying negative things about the people or organization involved.

18: Explain how you would be an asset to this organization.
Answer:
You should be anxious for this question. It gives you a chance to highlight your best points as they relate to the position being discussed. Give a little advance thought to this relationship.

19: Why should we hire you?
Answer:
Point out how your assets meet what the organization needs. Also mention about your knowledge, experience, abilities, and skills. Never mention any other candidates to make a comparison.

20: Tell me about a suggestion you have made.

Answer:

Have a good one ready. Be sure and use a suggestion that was accepted and was then considered successful. One related to the type of work applied for is a real plus.

21: What irritates you about co-workers?

Answer:

This is a trap question. Think real hard but fail to come up with anything that irritates you. A short statement that you seem to get along with folks is great.

22: What is your greatest strength?

Answer:

Numerous answers are good, just stay positive. A few good examples: Your ability to prioritize, Your problem-solving skills, Your ability to work under pressure, Your ability to focus on projects, Your professional expertise, Your leadership skills, Your positive attitude **23: Tell me about your dream job or what are you looking for in a job?**

Answer:

Stay away from a specific job. You cannot win. If you say the job you are contending for is it, you strain credibility. If you say another job is it, you plant the suspicion that you will be dissatisfied with this position if hired. The best is to stay genetic and say something like: A job where I love the work, like the people, can contribute and can't wait to get to work.

24: Why do you think you would do well at this job?

Answer:

Give several reasons and include skills, experience and interest.

25: What do you find the most attractive about this position? (Least attractive?)

Answer:

a) List a couple of attractive factors such as the responsibility the post offers and the opportunity to work with experienced teams that have a reputation for innovation and creativity.

b) Say you'd need more information and time before being able to make a

judgment on any unattractive aspects.

26: What kind of person would you refuse to work with?
Answer:
Do not be trivial. It would take disloyalty to the organization, violence or lawbreaking to get you to object. Minor objections will label you as a whiner.

27: What is more important to you: the money or the work?
Answer:
Money is always important, but the work is the most important. There is no better answer.

28: What would your previous supervisor say your strongest point is?
Answer:
There are numerous good possibilities: Loyalty, Energy, Positive attitude, Leadership, Team player, Expertise, Initiative, Patience, Hard work, Creativity, Problem solver.

29: Tell me about a problem you had with a supervisor.
Answer:
Biggest trap of all! This is a test to see if you will speak ill of your boss. If you fall for it and tell about a problem with a former boss, you may well below the interview right there. Stay positive and develop a poor memory about any trouble with a supervisor.

30: What has disappointed you about a job?
Answer:
Don't get trivial or negative. Safe areas are few but can include:
Not enough of a challenge. You were laid off in a reduction Company did not win a contract, which would have given you more responsibility.

31: Tell me about your ability to work under pressure.
Answer:
You may say that you thrive under certain types of pressure. Give an example

that relates to the type of position applied for.

32: Do your skills match this job or another job more closely?
Answer:
Probably this one! Do not give fuel to the suspicion that you may want another job more than this one.

33: What motivates you to do your best on the job?
Answer:
This is a personal trait that only you can say, but good examples are: Challenge, Achievement, and Recognition.

34: Are you willing to work overtime? Nights? Weekends?
Answer:
This is up to you. Be totally honest.

35: How would you know you were successful on this job?
Answer:
Several ways are good measures:
You set high standards for yourself and meet them. Your outcomes are a success. Your boss tells you that you are successful and doing a great job.

36: Would you be willing to relocate if required?
Answer:
You should be clear on this with your family prior to the interview if you think there is a chance it may come up. Do not say yes just to get the job if the real answer is no. This can create a lot of problems later on in your career. Be honest at this point. This will save you from future grief.

37: Are you willing to put the interests of the organization ahead of your own?
Answer:
This is a straight loyalty and dedication question. Do not worry about the deep

ethical and philosophical implications. Just say yes.

38: Describe your management style.
Answer:
Try to avoid labels. Some of the more common labels, like progressive, salesman or consensus, can have several meanings or descriptions depending on which management expert you listen to. The situational style is safe, because it says you will manage according to the situation, instead of one size fits all.

39: What have you learned from mistakes on the job?
Answer:
Here you have to come up with something or you strain credibility. Make it small, well intentioned mistake with a positive lesson learned. An example would be, working too far ahead of colleagues on a project and thus throwing coordination off.

40: Do you have any blind spots?
Answer:
Trick question! If you know about blind spots, they are no longer blind spots. Do not reveal any personal areas of concern here. Let them do their own discovery on your bad points. Do not hand it to them.

41: If you were hiring a person for this job, what would you look for?
Answer:
Be careful to mention traits that are needed and that you have.

42: Do you think you are overqualified for this position?
Answer:
Regardless of your qualifications, state that you are very well qualified for the position you've been interviewed for.

43: How do you propose to compensate for your lack of experience?
Answer:
First, if you have experience that the interviewer does not know about, bring that up: Then, point out (if true) that you are a hard working quick learner.

44: What qualities do you look for in a boss?
Answer:
Be generic and positive. Safe qualities are knowledgeable, a sense of humor, fair, loyal to subordinates and holder of high standards. All bosses think they have these traits.

45: Tell me about a time when you helped resolve a dispute between others.
Answer:
Pick a specific incident. Concentrate on your problem solving technique and not the dispute you settled.

46: What position do you prefer on a team working on a project?
Answer:
Be honest. If you are comfortable in different roles, point that out.

47: Describe your work ethic.
Answer:
Emphasize benefits to the organization. Things like, determination to get the job done and work hard but enjoy your work are good.

48: What has been your biggest professional disappointment?
Answer:
Be sure that you refer to something that was beyond your control. Show acceptance and no negative feelings.

49: Tell me about the most fun you have had on the job.
Answer:
Talk about having fun by accomplishing something for the organization.

50: What would you do for us? (What can you do for us that someone else can't?)
a) Relate past experiences that represent success in Working for your previous employer.
b) Talk about your fresh perspective and the relevant experience you can bring to the company.

c) Highlight your track record of providing creative, Workable solutions.

51: Do you have any questions for me?
Answer:
Always have some questions prepared. Questions prepared where you will be an asset to the organization are good. How soon will I be able to be productive? What type of projects will I be able to assist on ?, are few examples.

And Finally Good Luck!

*****

INDEX

Data Structures & Algorithms Questions

Data Structures

22: What is a parallel array?
23: Define an associative array.
24: Which are the operations usually defined on associative arrays:
25: Which the main operations defined on basic abstract arrays?
26: What is a Judy array?
27: What is the relation between pointers and arrays? Provide a short example.
28: What is the benefit of using arrays of pointers instead of several pointer variables?
29: How are elements of an array stored in memory?

Stacks

30: What is a stack? Give a practical example of a stack.
31: Which are the main operations performed on a stack?

Queues

32: What is a queue? Give a practical example of a queue.
33: What is the difference between a simple queue and a priority queue?
34: Which are the main operations performed on a queue?
35: What is a deque ?

Lists

36: What are the association lists? What are their advantages?
37: What is a linked list?
38: What is a single linked list?
39: What is the difference between a single linked list and a double linked list?
40: What is a circular list?
41: What condition defines an empty dynamic list?
42: Which condition must be accomplished in order to have only one element in a linked list?
43: What is the maximum dimension a dynamic list can have?
44: Which is the difference between an ordered list and a sorted list?
45: Which are the most common used operations on ordered lists?
46: Which are the most common used operations on sorted lists?
47: Why insertAfter and insertBefore operations are not provided for sorted lists?
48: Describe several usage models of circularly-linked lists.
49: What is a skip list?
50: Give example of a way to improve the speed of random access lookups in a skip list.
51: Give example of several applications and frameworks that use skip lists: 52: What is the purpose of using sentinel nodes?
53: What is an unrolled linked list?

Hash Data Structures

54: What is a hash function?
55: What are the characteristics of a good hash function?
56: What is a collision?
57: Enumerate several hashing methods.
58: Describe the division hashing method.
59: What is a scatter table?

60: What is a chained scatter table?
61: What is open addressing?
62: Describe several collision resolution strategies in open addressing.
63: What is lazy deletion?
64: Describe the mechanism of inserting a node into a hashtable .
65: Describe the steps that need to be done in order to delete an element from a hashtable .
66: What condition must be accomplished in order to decide that a hashtable is empty?
67: What is a Bloom filter?
68: What is a hashed array tree?

Trees

69: What is a tree?
70: Describe several properties of a tree node.
71: What is the degree of a tree node?
72: What is an Nary tree?
73: What is a binary tree?
74: Which are the two methods to visit all the nodes of a tree?
75: Which are the depth-first traversal methods?
76: Describe the preorder traversal algorithm.
77: Describe the inorder traversal algorithm.
78: Describe the postorder traversal algorithm.
79: Describe the breadth-first traversal algorithm.
80: Supposing we have an expression tree, which are the three notations associated to preorder, inorder and postorder traversal algorithms?
81: Enumerate several tree specializations.
82: What is a heap?
83: What is the difference between a max-heap and a min-heap?
84: Which are the operations usually performed on a heap?
85: Enumerate several heap specializations.
86: What is a Fibonacci heap?
87: Provide a short description of a B-tree.
88: What is a search tree?
89: Give example of some types of search trees.
90: What is a binary search tree?
91: Describe the algorithm of inserting data into a binary search tree.
92: Describe the procedure of removing an item from a binary search tree.
93: Define the concept of a treap .
94: What is an AVL search tree?
95: Define a self-balancing binary search tree.
96: What is a red-black tree?
97: What is the usage of red-black trees?
98: Make a short comparison between AVL trees and red-black trees.
99: Describe a splay tree.
100: What are the characteristics of a good balance condition?
101: Describe the steps that need to be performed while inserting an item into an AVL tree.
102: What is a M-way search tree?
103: Which are the algorithms for finding items in an M-way search tree?

104: What is a B-Tree?
105: What is a perfect binary tree?
106: What is a binary heap?
107: Define the term of leftist tree.
108: What is a binomial tree?
109: Define a T-tree and describe its usage.
110: What is a weight-balanced binary tree?
111: Describe a cartesian tree.
112: Describe on short the Day/Stout/Warren algorithm.
113: Define a BSP tree and describe its usage.
114: What is a trie?
115: Which are the main advantages of tries over binary search trees?
116: Which are the main advantages of tries over hash tables?
117: What are the bitwise tries?
118: What are ternary search trees?
119: What is a binomial queue?

Sets

120: What is a set?
121: What are the most common operations on sets? Provide a short description.
122: What is a multiset? Define the cardinality of a multiset.
123: What is a disjoint-set data structure?
124: What are the disjoint-set forests?

Graphs

125: What is a graph?
126: What is the relationship between trees and graphs?
127: What is a hypergraph?
128: Give example of data structures used for graphs representation: 129: What is a dense graph?
130: What is a sparse graph?
131: Define a complete graph.
132: What is a circular graph?
133: Define a bipartite graph.
134: Describe several types of a cycle graphs.
135: What is the degree of a vertex?
136: What is a dual graph?
137: What is a Hamiltonian cycle?
138: What is an Eulerian path?
139: Define a planar graph.
140: What is a k-connected graph?
141: Enumerate several graph traversal algorithms:
142: Describe the depth-first graph traversal algorithm.
143: Describe the breadth-first graph traversal algorithm.
144: What is a topological sort?
145: What is a weighted graph?
146: Describe the shortest path problem in graph theory.

147: Give examples of generalizations of single-pair shortest path problem.
148: Enumerate some algorithms designed to solve the shortest path problem.
149: What is Kruskal's algorithm?
150: Describe Dijkstra's algorithm.
151: Provide a short description of Floyd's algorithm.
152: What is the purpose of critical path analysis?
153: Describe Christofides algorithm.
154: What is a knot (in a directed graph)?
155: What is the complement of a graph? Give example of several graph-theoretic concepts which are related to each other via complement graphs: 156: Define a dipole graph.
157: Describe the terms: regular graph, regular directed graph and k-regular graph.
158: What is a strongly regular graph?
159: Which are the smallest graphs that are regular but not strongly regular?
160: What is a Moore graph?

Algorithms

General Concepts Of Algorithms

161: Define the concept of an algorithm.
162: Define a brute-force algorithm. Give a short example.
163: What is a greedy algorithm? Give examples of problems solved using greedy algorithms.
164: Which are the pillars of a greedy algorithm?
165: What is a backtracking algorithm? Provide several examples.
166: What is the difference between a backtracking algorithm and a brute-force one?
167: Define an iterator. Provide usage example of an iterator.
168: Describe the visitor design pattern.
169: What is an adapter?
170: What is a singleton?
171: Describe on short the branch and bound algorithm.
172: Describe divide and conquer paradigm.
173: Describe dynamic programming paradigm.
174: How does a simulated annealing algorithm work?
175: Provide a short description of Euclid's algorithm. Provide its recursive description: Sorting Algorithms
176: Which are the sorting algorithms categories?
177: Describe on short an insertion sorting algorithm.
178: Which are the advantages provided by insertion sort?
179: What search algorithm does straight insertion sorting algorithm use?
180: What is the difference between insertion and exchange sorting?
181: Give examples of exchange sorting algorithms:
182: Shortly describe the quicksort algorithm.
183: Describe several variations of quicksort algorithm.
184: What is the difference between selection and insertion sorting?
185: What is merge sorting?
186: Which are the main steps of a merge sorting algorithm?
187: What is distribution sorting? Give example of several distribution sorting algorithms.
188: Describe the adaptive heap sort algorithm.

189: Describe the counting sort algorithm.
190: Describe Burstsort algorithm.

Search Algorithms

191: Describe the interpolation search algorithm.
192: Provide a short description of binary search algorithm.
193: What is a ternary search algorithm?
194: Describe golden section search algorithm.
195: Describe Fibonacci search technique.
196: What is the linear search algorithm?
197: Describe secant search algorithm.
198: What is best-first search algorithm?

Huffman Coding

199: What is Huffman coding?
200: Give several examples of Huffman coding applications.

*****

HR Questions

1: Tell me about yourself?
2: Why did you leave your last job?
3: What experience do you have in this field?
4: Do you consider yourself successful?
5: What do co-workers say about you?
6: What do you know about this organization?
7: What have you done to improve your knowledge in the last year?
8: Are you applying for other jobs?
9: Why do you want to work for this organization?
10: Do you know anyone who works for us?
11: What kind of salary do you need?
12: Are you a team player?
13: How long would you expect to work for us if hired?
14: Have you ever had to fire anyone? How did you feel about that?
15: What is your philosophy towards work?
16: If you had enough money to retire right now, would you?
17: Have you ever been asked to leave a position?
18: Explain how you would be an asset to this organization.
19: Why should we hire you?
20: Tell me about a suggestion you have made.
21: What irritates you about co-workers?
22: What is your greatest strength?
23: Tell me about your dream job or what are you looking for in a job?
24: Why do you think you would do well at this job?
25: What do you find the most attractive about this position? (Least attractive?)

26: What kind of person would you refuse to work with?
27: What is more important to you: the money or the work?
28: What would your previous supervisor say your strongest point is?
29: Tell me about a problem you had with a supervisor.
30: What has disappointed you about a job?
31: Tell me about your ability to work under pressure.
32: Do your skills match this job or another job more closely?
33: What motivates you to do your best on the job?
34: Are you willing to work overtime? Nights? Weekends?
35: How would you know you were successful on this job?
36: Would you be willing to relocate if required?
37: Are you willing to put the interests of the organization ahead of your own?
38: Describe your management style.
39: What have you learned from mistakes on the job?
40: Do you have any blind spots?
41: If you were hiring a person for this job, what would you look for?
42: Do you think you are overqualified for this position?
43: How do you propose to compensate for your lack of experience?
44: What qualities do you look for in a boss?
45: Tell me about a time when you helped resolve a dispute between others.
46: What position do you prefer on a team working on a project?
47: Describe your work ethic.
48: What has been your biggest professional disappointment?
49: Tell me about the most fun you have had on the job.
50: What would you do for us? (What can you do for us that someone else can't?) 51: Do you have any questions for me?

*****

Some of the following titles might also be handy:

1. Oracle / PLSQL Interview Questions
2. ASP.NET Interview Questions
3. VB.NET Interview Questions
4. .NET Framework Interview Questions
5. C#.NET Interview Questions

6. OOPS Interview Questions

7. Core Java Interview Questions

8. JSP-Servlet Interview Questions

9. EJB (J2EE) Interview Questions
10. ADO.NET Interview Questions

11. SQL Server Interview Questions

12. C & C++ Interview Questions
13. 200 (HR) Interview Questions

14. JavaScript Interview Questions

15. JAVA/J2EE Interview Questions
16. Oracle DBA Interview Questions
17. XML Interview Questions
18. Unix Shell Programming Interview Questions
19. PHP Interview Questions
20. J2ME Interview Questions

21. Hardware and Networking Interview Questions

22. Data Structures & Algorithms Interview Questions
23. Oracle E-Business Suite Interview Questions

24. UML Interview Questions

25. HTML, XHTML & CSS Interview Questions

26. JDBC Interview Questions

27. Hibernate, Springs & Struts Interview Questions

28. Linux Interview Questions

For complete list visit **www.vibrantpublishers.com**