Erick Vargas

Interview notes

Contents

1 Theory

1.1 Data structures

1.1.1 ¿What is a data structure?

A simple form to describe this is: a data structure is a container to store data in a specific layout. This layout allow us to make efficient our data structure in some operations but inefficient in others.

1.1.2 ¿Why do we need data strucures?

Depending of different scenarios, data needs to be stored in a specific format, to solve this we need to know the different types of data structures.

1.1.3 Basic data structures

Nowadays there are a lot of types of data structures but here we have a little list of the commonly used data structures.

- Arrays
- Stacks
- Queues
- Linked list
- Trees
- Graphs
- Tries
- Hash tables

Arrays

An array is the simplest used data structure and other data structures can be made using arrays, like stacks and queues. Most of the programming languages index the first element in zero.

We have two types of arrays:

- One-dimensional arrays
- Multi-dimensional arrays

Basic operations on arrays

1. Insert: Inserts an element at given index
2. Get: returns the element at given index
3. Delete: Deletes an element at given index
4. Size: Get the total number of elements in an array

Commonly asked array interview questions

1. Find the second minimum element of an array
2. First non-repeating integers in the array
3. Merge two sorted arrays
4. Rearrage positive and negative values in an array

Stacks

This data structure has a lot of applications, for example when we use the undo command. A stack works like a pile of something for example books. In this example each book are placed in a vertical order. If we have a stack of books we can not get a book placed in the middle, first ypu need to remove all the books on the top to get it. This behaviour is know as LIFO (Last In First Out)

Basic operations

1. Push: inserts an element of the top
2. Pop: returns the top element, after removing from the stack
3. isEmpty: returns true if the stack is is empty
4. Top: returns the top element without removing from the stack

Commonly asked stack interview questions

1. Evalue postfix expression using a stack
2. Sort values in a stack
3. Check balanced parentheses in an expression

Queues

Similar to stack, queue is another data structure that stores the element in a sequential manner. The difference is that instead of use LIFO method, queue implements FIFO method (First In First Out)

An example of a queue in real life is a line of people waiting to buy a ticket. If a new person comes, he or she will join the line from the end, not from the start and the person first person (start) will be the first to buy the ticket and then he or she leaves the line.

Basic operations

1. Enqueue: inserts an element at the end of the queue
2. Dequeue: removes an element from the start of the queue
3. isEmpty: returns true if the queue is empty
4. top: returns the first element of the queue

Commonly asked queue interview questions

1. Implement a stack using a queue
2. Reverse first k elements of a queue
3. Generate binary numbers from 1 to n using a queue

Linked list

A linked list is another important data structure that is very similar to an array but differs in memory allocation, internal structure and how basic operations of insertion and deletion are carried out.

A linked list is like a chain of nodes, where each node contains information like data and a ponter to the succeding node in the chain. There's a head pointer, which points to the first element of the linked list, and if the list is empty then it simply points to null or nothing.

Linked lists are used to implement file systems, hash tables, and adjacency lists.

Also exists different types of linked lists:

- Singly linked lists
- Doubly linked lists

Basic operations

1. InsertAtEnd: inserts given element at the end of the linked list
2. InsertAtHead: inserts given element at the start/head of the linked list
3. Delete: deletes given element from the linked list
4. DeleteAtHead: deletes first element of the linked list
5. Search: returns the given element from a linked list
6. isEmpty: returns true if the linked list is empty

Commonly asked linked list interview questions

1. Reverse a linked list
2. Detect loop in a linked list
3. Return N-th node from the end in a linked list
4. Remove duplicates from a linked list

Graphs

A graph is a set of nodes that are connected to each other in the form of a network. Nodes are also called vertices. A pair(x, y) is called an edge, which indicates that vertex x is conected to vertex y. An edge may contain weight/cost, showing how much cost is required to traverse from vertex x to y.

Also we have different types of graphs:

- Undirected graphs $\longrightarrow$
- Directed graphs $\longleftrightarrow$

We can represent graphs in two forms:

- Adjacency matrix
- Adjacency list

Finally we have two common graph traversing algorithms:

- Breadth First Search (BFS)
- Depth First Search (DFS)

Commonly asked linked list interview questions

1. Implement Breath and Depth First Search
2. Check if a graph is tree or not
3. Count number of edges in a graph
4. Find the shortest path between two vertices