Name : Mirza Mohammed Juniad

CORE CSE

Q1. List out different OOPS principles and explain with examples?

Ans :

1. Object
→ Object can be defined as an instance of a class. An object contains an address and takes up some space in memory. Objects can communicate without knowing the details of each other's data or code.
Syntax : in python → obj_name = className()

2. Class
→ Collection of object is called class, It is also blueprint from which you can create an individual object. Class doesn't consume any space.
Syntax : in python → class className :

3. Inheritance
→ *When one object acquires all the properties and behaviors of a parent object*, it is known as inheritance. It provides code reusability.
Syntax : in python →

```
class parent:
    def func():
        print("Hello World")

class base(parent):
    a = 10

obj = base()
obj.func()
```

4. Polymorphism
→ Polymorphism is the ability of an object to take on many forms. Example, 2 function can have same name at the same the may have same no of perimeters.

5. Encapsulation
→ *Binding (or wrapping) code and data together into a single unit are known as encapsulation.*

Q.2) Explain data structures that are mutable versus immutable?

Ans :

Mutable Data Structure :
I. Mutable is when something is changeable or has the ability to change.
ii. In Python, 'mutable' is the ability of objects to change their values. These are often the objects that store a collection of data.

Mutable Object are Lists,Sets,Dictionaries,etc..

Immutable Data Structure :
I. Immutable is the when no change is possible over time.

ii. In Python, if the value of an object cannot be changed over time, then it is known as immutable. Once created, the value of these objects is permanent.

Imutable Object are Int,Strings,Tuples,etc..

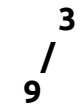
Q3. Construct a binary tree using inorder and post order traversal given below.
Inorder Traversal: 9, 3, 15, 20, 7
Post Order Traversal: 9, 15, 7, 20, 3

Ans :

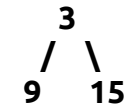
Inorder Traversal :
Order : left → parent → right

Step 1)

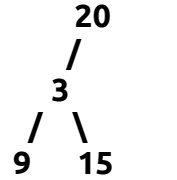
Step 2)



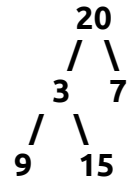
Step 3)



Step 4)



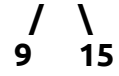
Step 5)



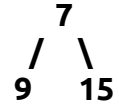
Post Order Traversal :
Order : left→ right → parent

Step 1)

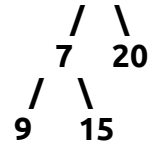
Step 2)



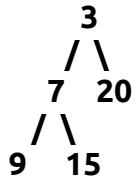
Step 3)



Step 4)



Step5)



Q4. Explain Dijkstra's algorithm.

Ans :

Dijkstra algorithm is a single-source shortest path algorithm. Here, single-source means that only one source is given, and we have to find the shortest path from the source to all the nodes.

Algorithm Steps:

- **Set all vertices distances = infinity except for the source vertex, set the source distance = .**
- **Push the source vertex in a min-priority queue in the form (distance , vertex), as the comparison in the min-priority queue will be according to vertices distances.**
- **Pop the vertex with the minimum distance from the priority queue (at first the popped vertex = source).**
- **Update the distances of the connected vertices to the popped vertex in case of "current vertex distance + edge weight < next vertex distance", then push the vertex with the new distance to the priority queue.**
- **If the popped vertex is visited before, just continue without using it.**
- **Apply the same algorithm again until the priority queue is empty.**

Q5. Differentiate BFS and DFS with their Time complexity.

Ans :

Sr.No	BFS	DFS
1	BFS stands for Breadth First Search.	DFS stands for Depth First Search.
2	BFS(Breadth First Search) uses Queue data structure for finding the shortest path.	DFS(Depth First Search) uses Stack data structure.
3	BFS can be used to find single source shortest path in an unweighted graph, because in BFS, we reach a vertex with minimum number of edges from a source vertex.	In DFS, we might traverse through more edges to reach a destination vertex from a source.
4	BFS is more suitable for searching vertices which are closer to the given source.	DFS is more suitable when there are solutions away from source.
5	BFS considers all neighbors first and therefore not suitable for decision making trees used in games or puzzles.	DFS is more suitable for game or puzzle problems. We make a decision, then explore all paths through this decision. And if this decision leads to win situation, we stop.
6	Here, siblings are visited before the children	Here, children are visited before the siblings
7	The Time complexity of BFS is O(V + E) when Adjacency List is used and O(V^2) when Adjacency Matrix is used, where V stands for vertices and E stands for edges.	The Time complexity of DFS is also O(V + E) when Adjacency List is used and O(V^2) when Adjacency Matrix is used, where V stands for vertices and E stands for edges.

Q6. Explain all disc scheduling algorithms.

Ans :

List of Disk Scheduling Algorithn ;

- **FCFS scheduling algorithm**
- **SSTF (shortest seek time first) algorithm**
- **SCAN scheduling**
- **C-SCAN scheduling**
- **LOOK Scheduling**
- **C-LOOK scheduling**

1) FCFS Scheduling

It is the simplest Disk Scheduling algorithm. It services the IO requests in the order in which they arrive.

The First Process which Arrives get Executes First.

Diadvantage :

- **The scheme does not optimize the seek time.**
- **The request may come from different processes therefore there is the possibility of inappropriate movement of the head.**

2) SSTF (shortest seek time first) algorithm

Shortest seek time first (SSTF) algorithm selects the disk I/O request which requires the least disk arm movement from its current position regardless of the direction. It reduces the total seek time as compared to FCFS.
It allows the head to move to the closest track in the service queue.

Disadvantage :

- **It may cause starvation for some requests.**
- **Switching direction on the frequent basis slows the working of algorithm.**
- **It is not the most optimal algorithm.**

3) SCAN scheduling

It is also called as Elevator Algorithm. In this algorithm, the disk arm moves into a particular direction till the end, satisfying all the requests coming in its path,and then it turns backand moves in the reverse direction satisfying requests coming in its path.

It works in the way an elevator works, elevator moves in a direction completely till the last floor of that direction and then turns back.

4) C-SCAN algorithm

In C-SCAN algorithm, the arm of the disk moves in a particular direction servicing requests until it reaches the last cylinder, then it jumps to the last cylinder of the opposite direction without servicing any request then it turns back and start moving in that direction servicing the remaining requests.

5) Look Scheduling

It is like SCAN scheduling Algorithm to some extant except the difference that, in this scheduling algorithm, the arm of the disk stops moving inwards (or outwards) when no more request in that direction exists. This algorithm tries to overcome the overhead of SCAN algorithm which forces disk arm to move in one direction till the end regardless of knowing if any request exists in the direction or not.

6) C Look Scheduling

C Look Algorithm is similar to C-SCAN algorithm to some extent. In this algorithm, the arm of the disk moves outwards servicing requests until it reaches the highest request cylinder, then it jumps to the lowest request cylinder without servicing any request then it again start moving outwards servicing the remaining requests.

It is different from C SCAN algorithm in the sense that, C SCAN force the disk arm to move till the last cylinder regardless of knowing whether any request is to be serviced on that cylinder or not.