LAB 4 CLO:07

CLO	0	1	2	3
CLO2	0 for no problem solved	3.0 Marks for Question 1	3.0 Marks for Question 1	.0 Marks for Question 1

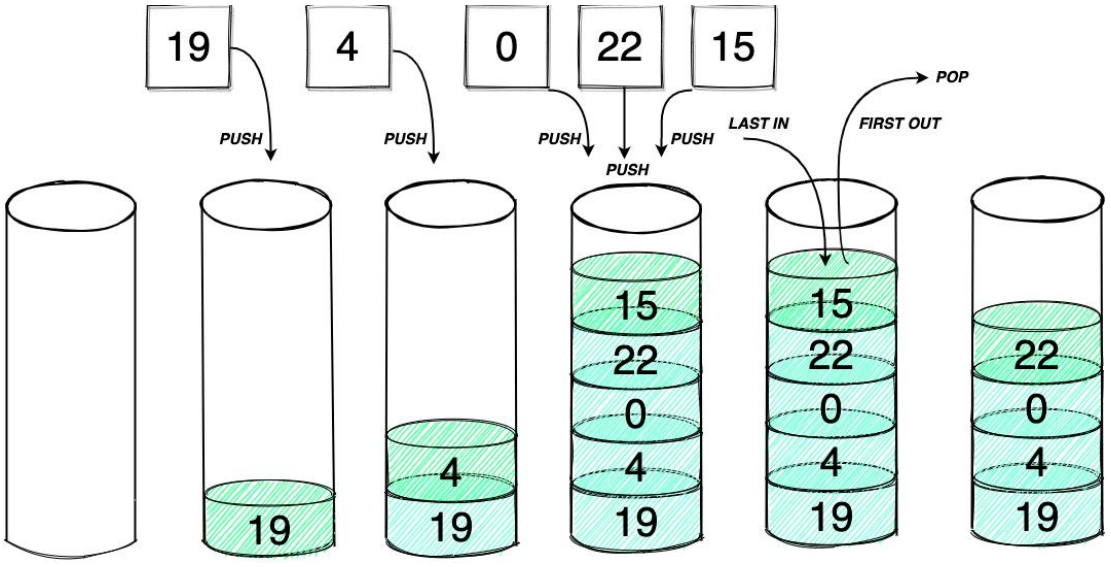
Class Learning Outcomes

Students will learn about the implementation in Python of

- Stack
- Queue
- Binary Search
- Priority Queue

Stack

A **stack** is a linear data structure that stores items in a Last-In/First-Out (LIFO) or First-In/Last-Out (FILO) manner. In stack, a new element is added at one end and an element is removed from that end only. The insert and delete operations are often called push and pop.

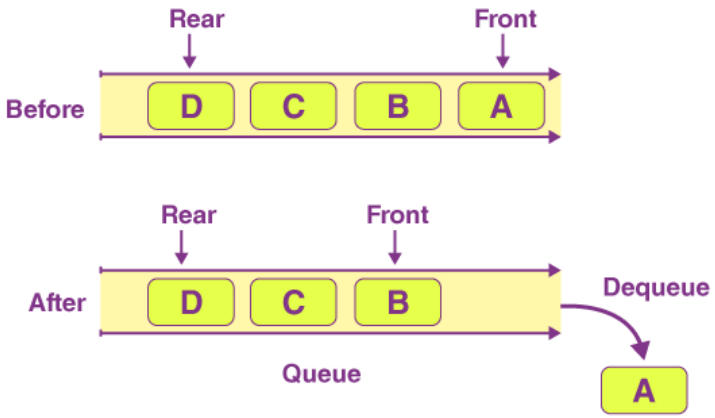


The functions that can be implemented on the stack are as follows:

- **empty()** – Returns whether the stack is empty
- **size()** – Returns the size of the stack
- **top() / peek()** – Returns a reference to the topmost element of the stack
- **push(a)** – Inserts the element 'a' at the top of the stack
- **pop()** – Deletes the topmost element of the stack

Queue

Like a stack, the queue is a linear data structure that stores items in a First In First Out (FIFO) manner. With a queue, the least recently added item is removed first. A good example of a queue is any queue of consumers for a resource where the consumer that came first is served first.



Queue Dequeue

The operations associated with queue are:

- **Enqueue:** Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition
- **Dequeue:** Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow
- **Front:** Get the front item from the queue
- **Rear:** Get the last item from the queue

Priority Queue

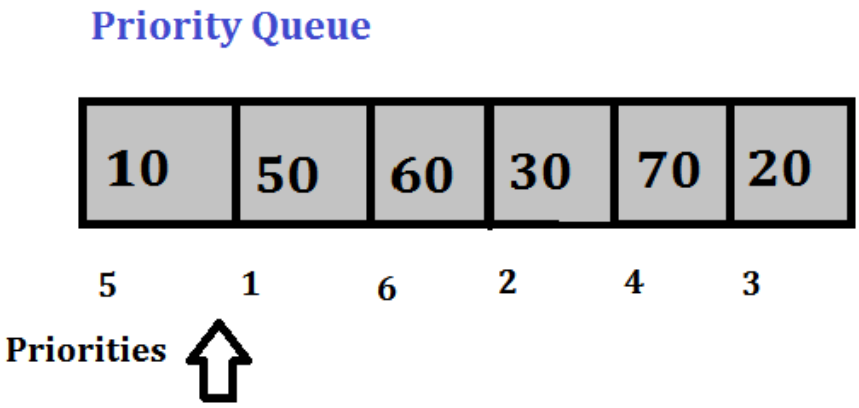
Priority Queues are abstract data structures where each data/value in the queue has a certain priority. For example, airlines' baggage having the title "Business" or "First-class" arrives earlier than the rest.

Priority Queue is an extension of the queue with the following properties.

1. An element with high priority is dequeued before an element with low priority.

2. If two elements have the same priority, they are served according to their order in the queue. Various **applications** of the Priority queue in Computer Science are:

   Job Scheduling algorithms, CPU and Disk Scheduling, managing resources shared among different processes, etc.



Key differences between Priority Queue and Queue:

1. In Queue, the oldest element is dequeued first. While, in the Priority Queue, an element based on the highest priority is dequeued.
2. When elements are popped out of a priority queue the result obtained is either sorted in Increasing order or Decreasing Order. While, when elements are popped from a simple queue, a FIFO order of data is obtained in the result.

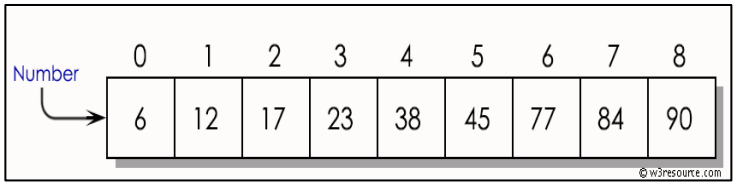
Binary Search

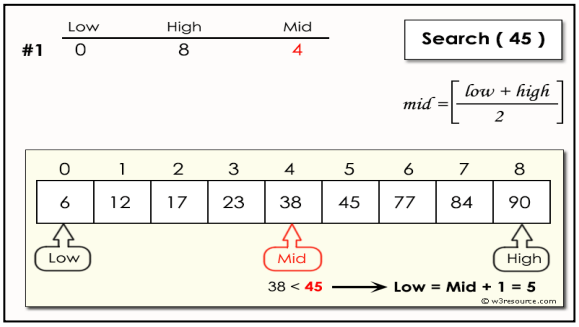
Binary Search is a technique used to search element in a sorted list. In computer science, a binary search or half-interval search algorithm finds the position of a target value within a sorted array. The binary search algorithm can be classified as a dichotomies divide-and-conquer search algorithm and executes in logarithmic time.

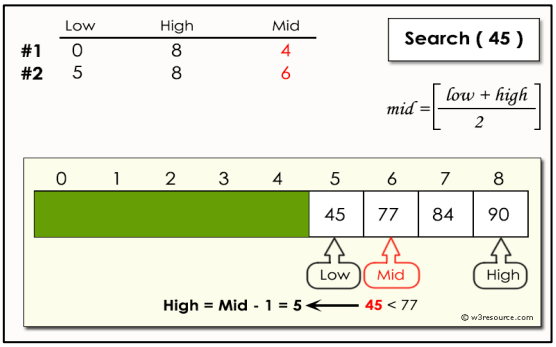
In a nutshell, this search algorithm takes advantage of a collection of elements that are already sorted by ignoring half of the elements after just one comparison.

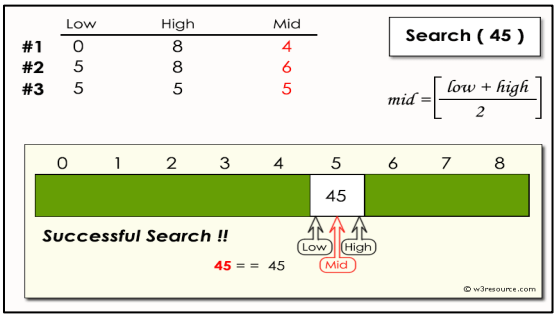
1. Compare x with the middle element.
2. If x matches with the middle element, we return the mid index.
3. Else if x is greater than the mid element, then x can only lie in the right (greater) half subarray after the mid element. Then we apply the algorithm again for the right half.
4. Else if x is smaller, the target x must lie in the left (lower) half. So, we apply the algorithm for the left half.

Below is the step-by-step example:









Classes in Python

- **Class**: A blueprint for creating objects (instances). It defines attributes (data) and methods (functions).
- **Object**: An instance of a class. You can create multiple objects from a single class.
- **Attributes**: Variables that belong to a class. There are two types:
  - Instance attributes: Attributes specific to an object.
  - Class attributes: Shared by all instances of the class.
- **Methods**: Functions that belong to a class. They usually operate on the attributes of the object.
- **__init__() method**: A special method (constructor) that is called when an object is instantiated. It initializes the object's attributes.
- **self**: Refers to the instance of the class. It is used to access variables and methods within the class from methods of that object.

```python
class Car:
    # Constructor method (__init__) to initialize the object's attributes
    def __init__(self, brand, model, year):
        self.brand = brand
        self.model = model
        self.year = year
    # Method to display car details
    def car_details(self):
        return f"Car: {self.year} {self.brand} {self.model}"
    # Method to simulate starting the car
    def start(self):
        return f"{self.brand} {self.model} is starting."

car1 = Car("Toyota", "Corolla", 2021)
car2 = Car("Honda", "Civic", 2020)

print(car1.car_details())  # Output: Car: 2021 Toyota Corolla
print(car2.car_details())  # Output: Car: 2020 Honda Civic
print(car1.start())  # Output: Toyota Corolla is starting.
print(car2.start())  # Output: Honda Civic is starting.
```

Exercise Questions

1. Stack Implementation

Problem: Implement a basic stack with the operations ***push***, ***pop***, ***top***, ***size***, and ***empty***.

2. Queue Implementation

Problem: Implement a basic queue with the operations ***enqueue***, ***dequeue***, ***front***, ***rear***, ***empty***, and ***size***.

3. Priority Queue Implementation

Problem: Implement a priority queue that dequeues the element with the highest priority first.

4. Binary Search Algorithm

Problem: Implement a binary search to find the index of a target element in a sorted list.