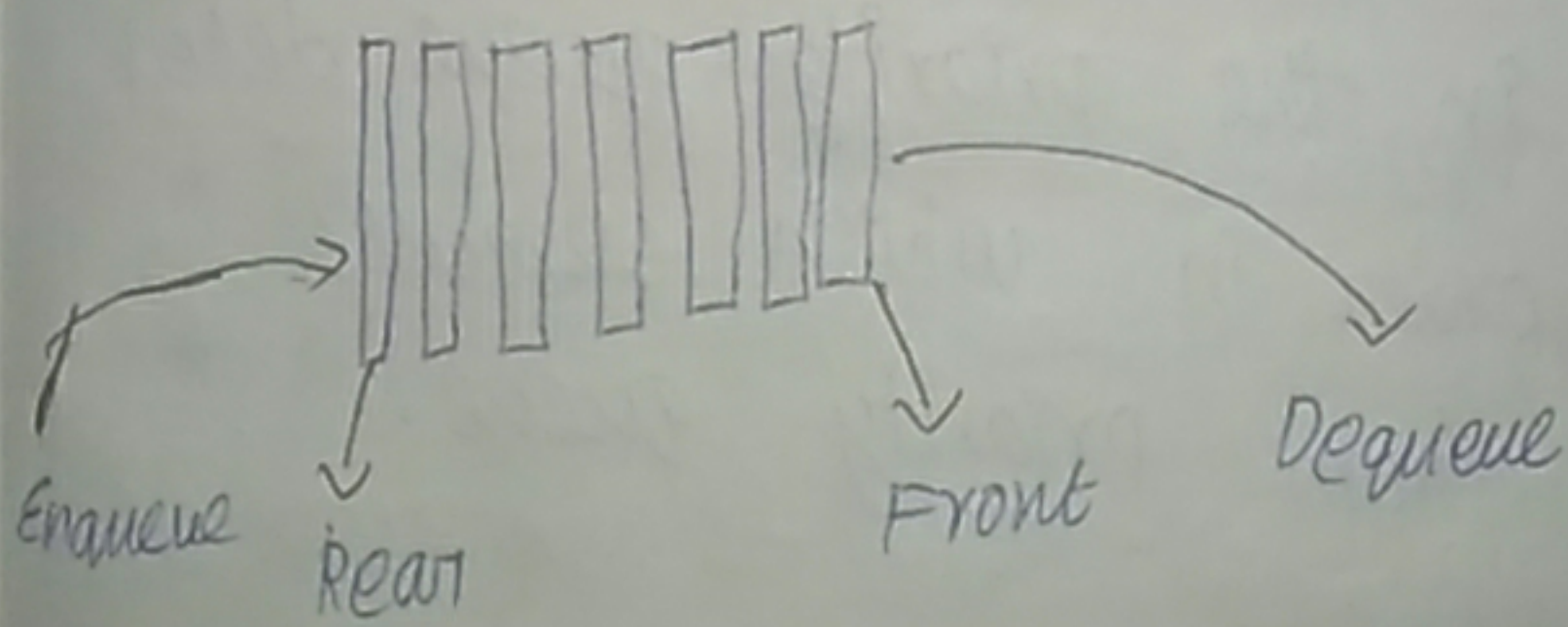
Day 7

Queue

A queue linear data structure which follows a particular order in which the operations are performed. The order is First in first out.

The difference between stacks and queue. In a stacks we remove the item the most recently added. In queue, we remove the item the least recently added.



* Insertion and dellion happen on different ends.

Application Of Queue data -structure:-

* when a resource is shared among multiple customers. Examples including cpu scheduling, Disk scheduling.

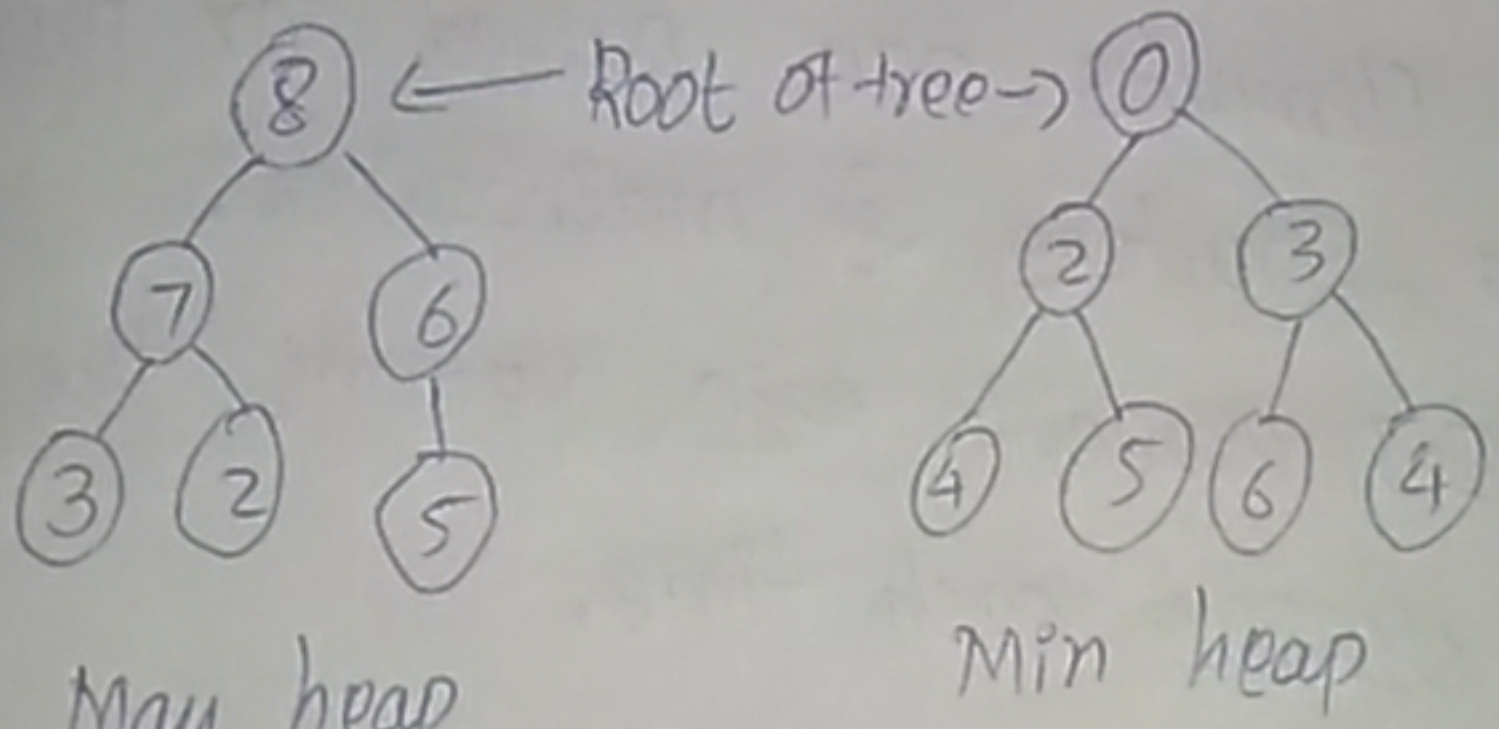
* when data is transfered asynchronsly (data not nessarily received at same rate as sent) between two process.

'what is ~~pirity queue~~? priority queue?

A priority queue is an abstract data type that operate similar to a normal queue except that each element has a certain priority. The priority of the element in the priority queue determine the order in which elements are removed from priority queue.

What is the heap?

A heap is a tree based data structure that satisfies the heap invariant. It A is a parent node of B then A is ordered with respect to B for all nodes A, B in the heap.



Max heap

Min heap

Complexity priority queue with binary heap?

* Binary heap Construction - O(n)
* polling - O(log(n))
* peeking - O(1)
* Adding - O(log(n))

Heap Queue or heapq

In python, it is available using "Heapq" module. The property of this data structure in python is that each time Smallest of heap element is popped. whenever elements are pushed or popped, heap data structure is maintained. The heap[0] element also returns the Smallest element each time.

Various Operations of heap :-

* Heapify (iterable) :- This function used to convert the iterable into a heap data structure. i.e in heap order.

* heappush (heap, ele) : This function used to insert the element mentioned in its arguments into heap. The order is adjusted, so as heap structure is maintained

* heappop(heap):

This function is used to remove and return smallest element from the heap.

* Heappushpop(heap, ele)

This function combines the functioning in both push and pop operations in one statement, increasing efficiency.

* heapreplace(heap, ele):

In this, element is first popped, then the element is pushed, i.e the value larger then the pushed value can be returned. heapreplace() returns the smallest value originally in heap regardless of pushed element as opposed to heappushpop()

* nlargest(k, iterable, key = fun)

This function is used to return the k largest element from the iterable specified and satisfying the key if mentioned.

* nsmallest :-

This function is used to return the k smallest elements from the iterable specified and satisfied key if mentioned.

Queue Implementation using list:

```
queue =[]
queue.append(1)
queue.append(2)          => queue = [1,2,3]
queue.append(3)
# removing elements in queue
queue.pop(0)
queue.pop(0)             => queue = []
queue.pop(0)
```

Queue implementation using collection.deque.-

```
from collections import deque
q = queue()
q.append('a')
q.append('b')            => queue = ['a','b'
q.append('c')
```

q.popleft()
q.popleft() ⟹ q = []
q.popleft()

Queue Implementation using queue.Queue.

From queue import Queue

q = Queue(maxsize=3)

q.put('a')
q.put('b')
q.put('c')

print("\nfull: " q.full()) => Full : true

q.get()
q.get()
q.get()

print("\nEmpty : ", q.Empty())

Empty : True.