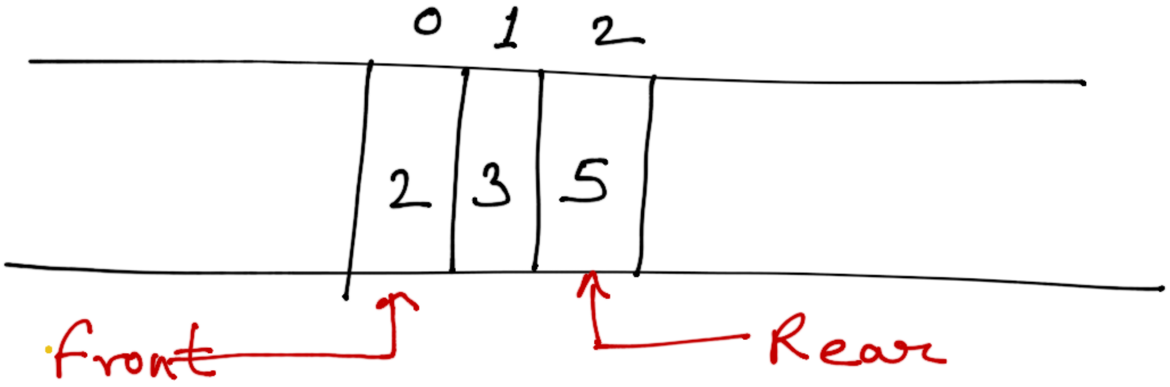
Queue

- Linear data structure
- as ADT
- Based on FIFO

Enqueue → Insertion — Rear/Tail

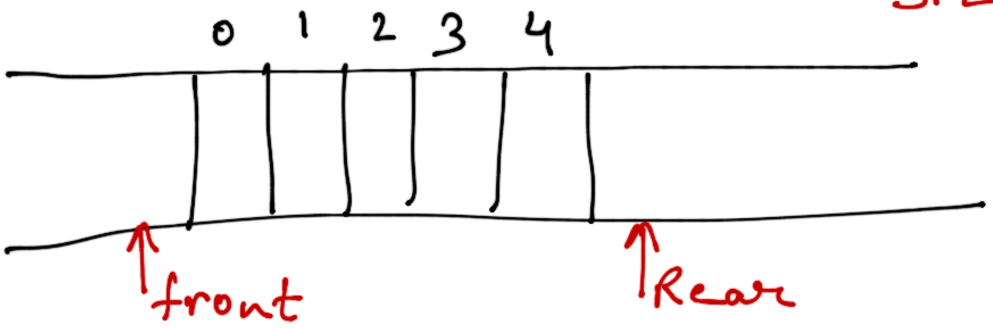
Dequeue → Deletion — Head/front



Operations

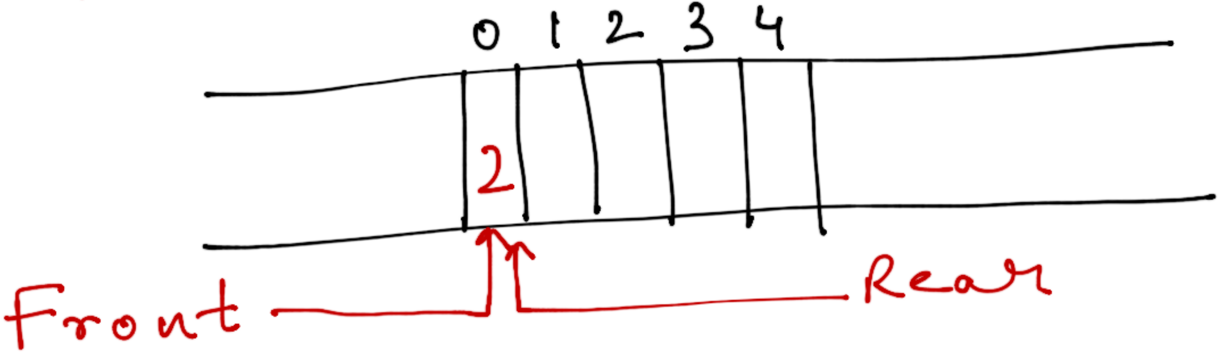
- Enqueue(2)
- Dequeue()
- Peek() / front()
- isFull()
- isEmpty()

Size = 5

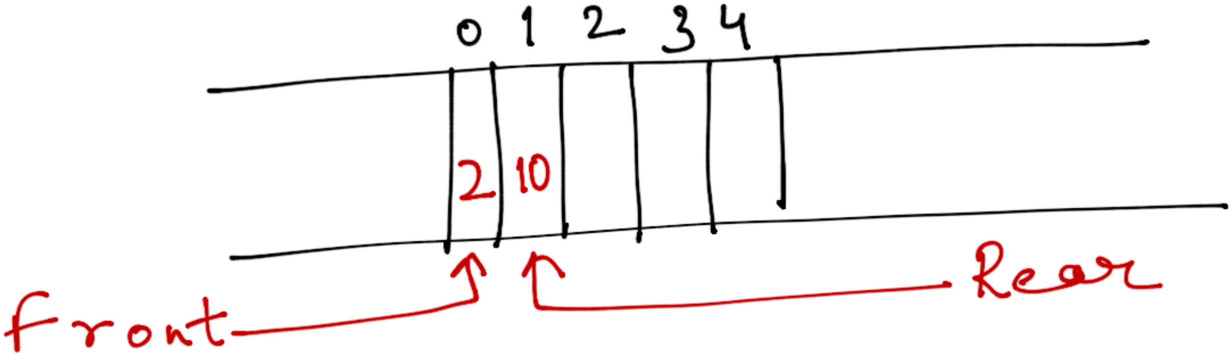


Intial (No Element) front = Rear = -1 Empty

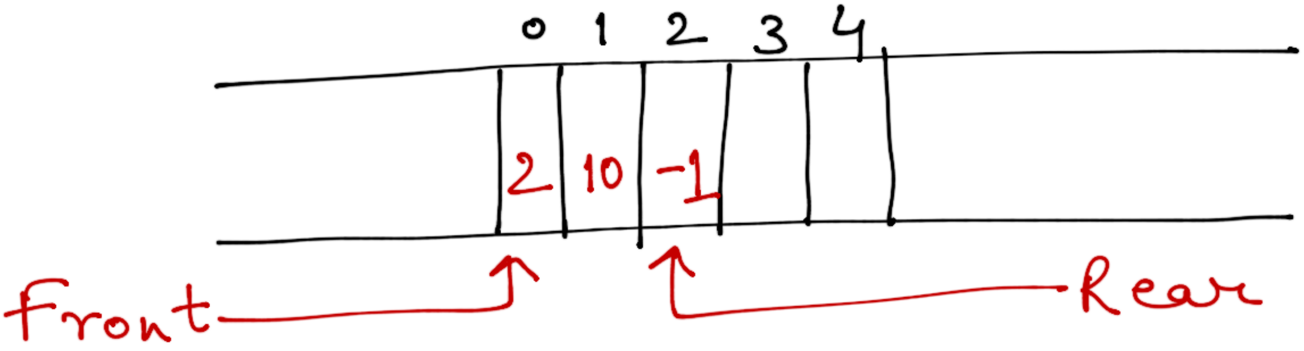
Enqueue(2)



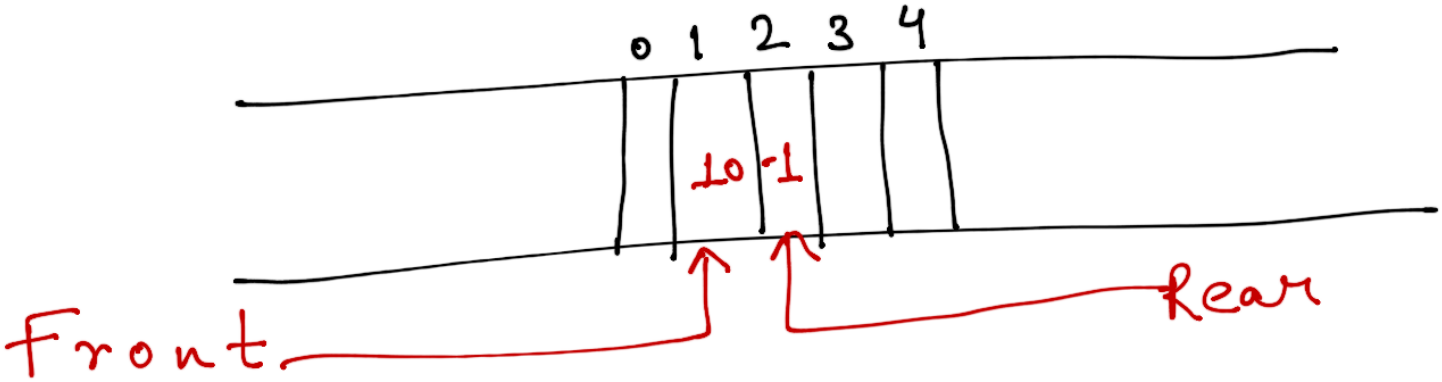
Enqueue(10)



Enqueue(-1)

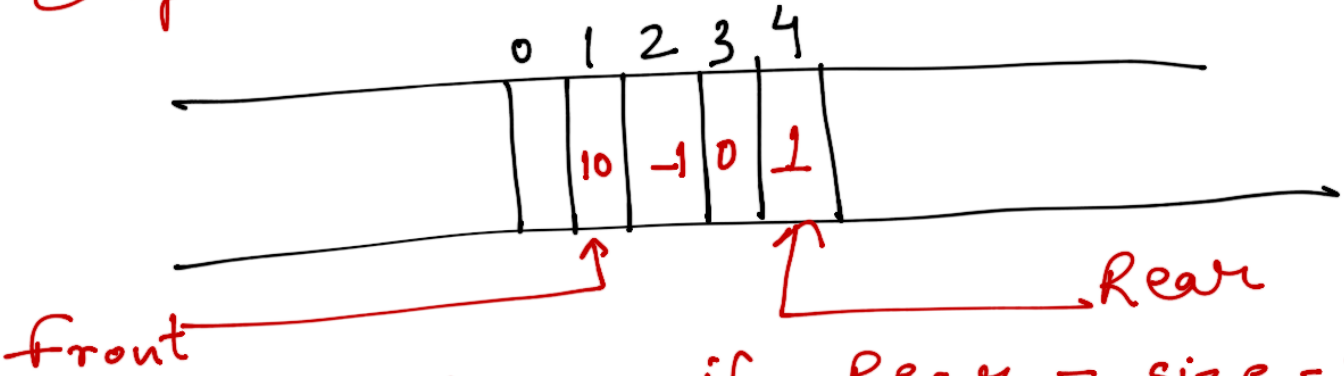


Dequeue()



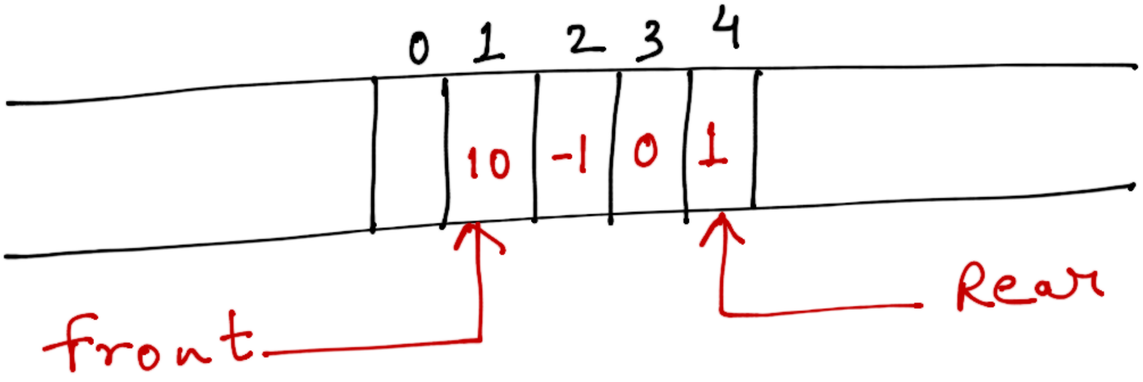
(Increment the front)

Enqueue(0), Enqueue(1)



Enqueue(5)

if Rear = size-1
"overflow"

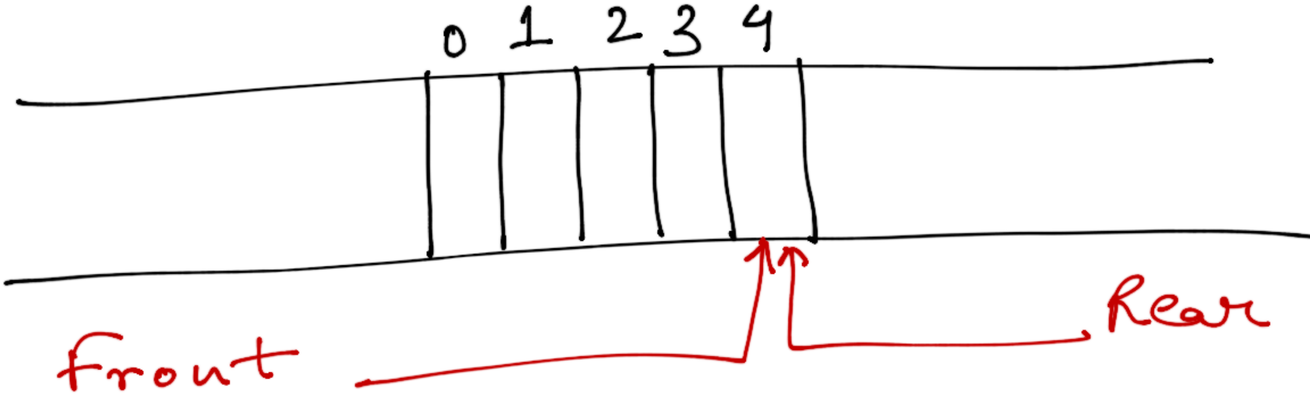


Peek() → 10 (Element present at front position)

Dequeue()

Dequeue()

Dequeue()

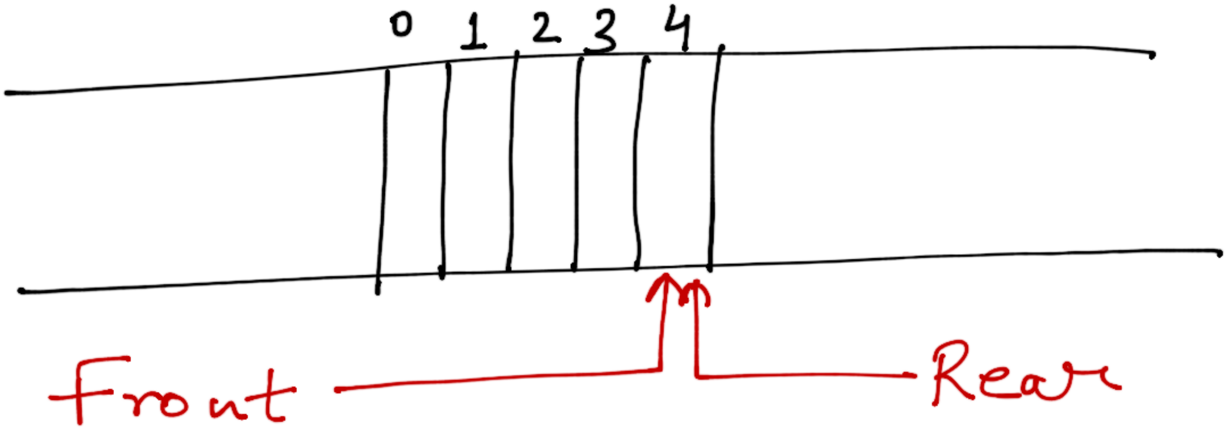


front = = Rear, only 1 element

After removing, front = Rear = -1

OR, front > Rear ⇒ Empty

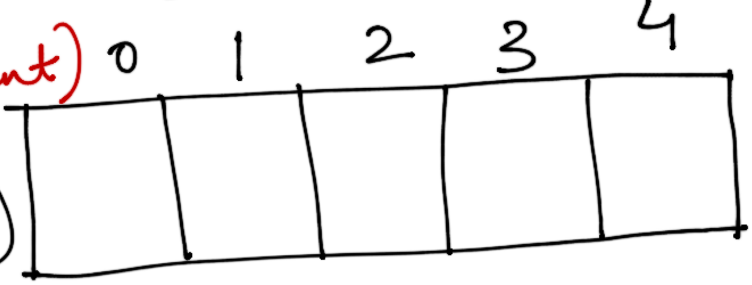
Enqueue(-15) Size = 5



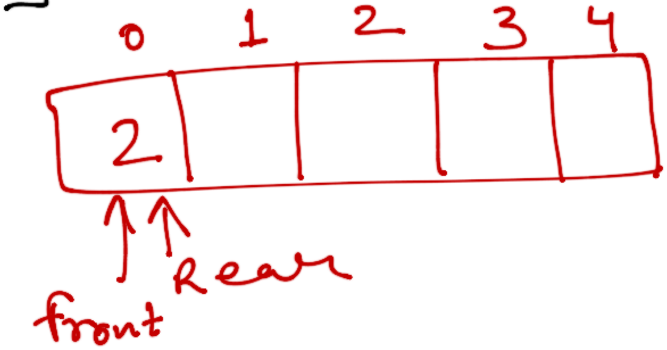
Can we enqueue here ??

Implementation Using Arrays

```
int queue [size]
```



```
Enqueue(DataElement)
{
if (Rear == Size-1)
    { "Overflow" }
elsif(Front == -1 && Rear == -1)
                    ( Empty Queue)
  { front = Rear = 0
       queue[Rear] = Data Element
  }
```



```
else { Rear ++
       queue[Rear] = Data Elemt
     }
}
```

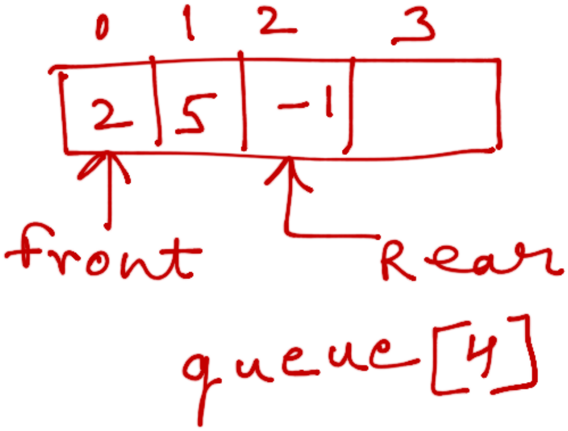
```
Dequeue ( )
{
 if(Front = = -1 && Rear = = -1 )
          { " Underflow" }
elseif(Front = = Rear )
          { Front = Rear = -1 }
else
   { Front++

     }

  }
```

Display ( )

```
{
  if (front = = -1 && Rear = = -1)
        {    Empty        }
  else
    { for (i = front ; i < Rear+1 ; i++)
          {    PRINT queue[i]
             }
      }
}
```


0
1
2
3
2
5
-1
front
Rear
queue[4]