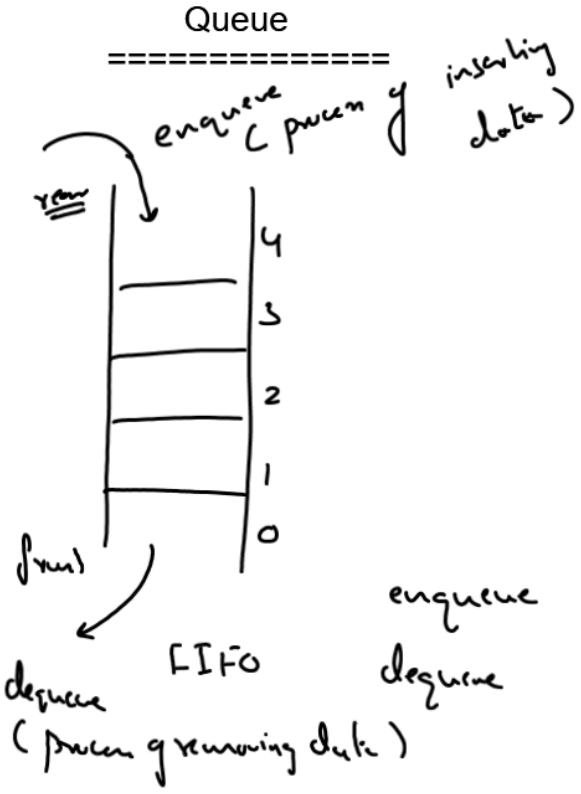


1. A queue , just like a Stack is a linear data structure.

2. But unlike a Stack , a queue is open from both the ends.

3. However its both ends are not used for both the operations.

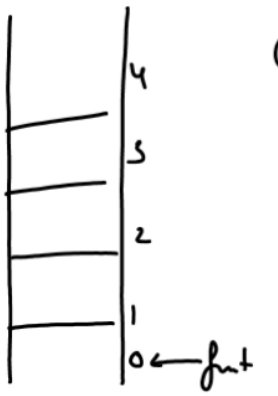
4. Rather one end of the queue is used for insertion of elements and the opposite end is used for deletion of elements.

5. The end used for insertion is called **rear** and the end used for deletion is called **front.**

6. The process of insertion is called ENQUEUE and the process of deletion is called DEQUEUE.

7. Since insertion and deletion take place from opposite ends we say a Queue follows FIFO (First In First Out)

①

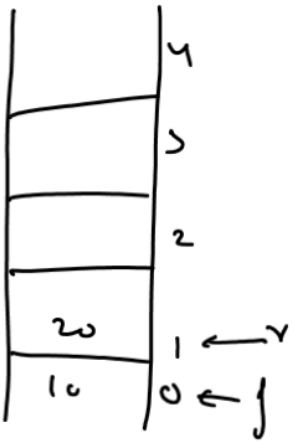


An empty queue

②

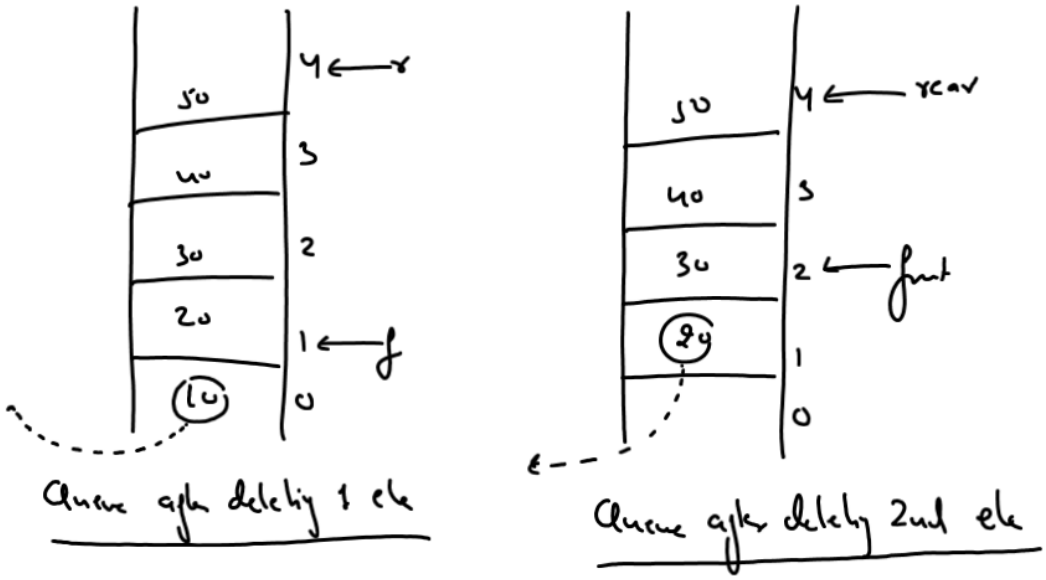
A queue with 1 ele

③



A queue with 2 ele

A full queue



Algorithm For enqueue() Function
==========

1. Check for overflow

2. If Queue is full , then print the message QUEUE OVERFLOW and return.

3. Otherwise , increment REAR by 1.

4. Place the element at the position pointed by REAR in the QUEUE.

5. Finish and return

Algorithm For dequeue() Function
==========

1. Check for underflow

2. If Queue is empty , then print the message QUEUE UNDERFLOW and return.

3. Otherwise , delete the element pointed by FRONT from the QUEUE.

4. Increment FRONT by 1

5. Return the deleted element .

6. Finish

Implementing Queue In C

```
#include<stdio.h>
struct Queue
{
   int arr[5];
   int front,rear;
};
void enqueue(struct Queue *,int);
int dequeue(struct Queue *);
int main()
{
    int choice,x;
    struct Queue Q;
    Q.front=0;
    Q.rear=-1;
    do
    {
      printf("Select an operation:");
      printf("\n1.Enqueue\n2.Dequeue\n3.Quit:");
      printf("\nEnter your choice:");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1:
                  printf("Enter ele:");
                  scanf("%d",&x);
                  enqueue(&Q,x);
                  break;
        case 2:
                 x=dequeue(&Q);
                 if(x!=0)
                    printf("\nDequeued ele: %d",x);
                 break;
```

```
case 3:
            printf("\nThank you for using the app");
            break;
default:
            printf("Wrong choice! Try again");
  }
}while(choice!=3);
return 0;
}
```

```
void enqueue(struct Queue *P,int x)
{
   if(P->rear==4)
     {
          printf("Queue Overflow");
          return;
     }
P->rear++;
P->arr[P->rear]=x;
printf("\nEnqueued %d",x);
}
```