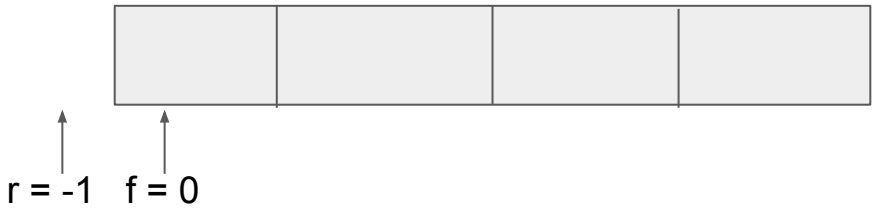
Linear Queue

By Er. Kushal Ghimire

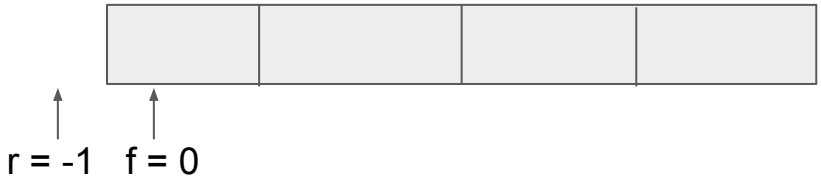
Linear Queue ADT

- A linear data structure that follows FIFO principle.
- Two pointers rear and front keeps track of the start and end of the queue.
- Time Complexity:
  - Search = O(n)
  - Insert = O(1)
  - Delete = O(1)

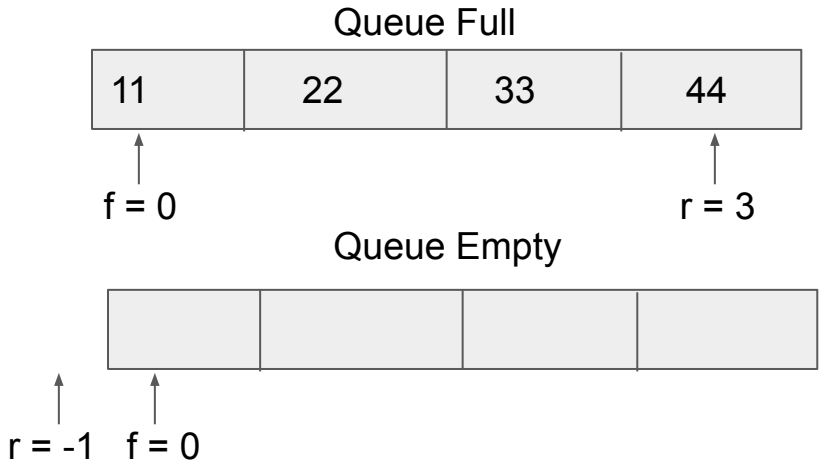


Initialization

- There are various ways to initialize a queue.
- We are considering the following condition to initialize a queue:
  - rear = -1
  - front = 0

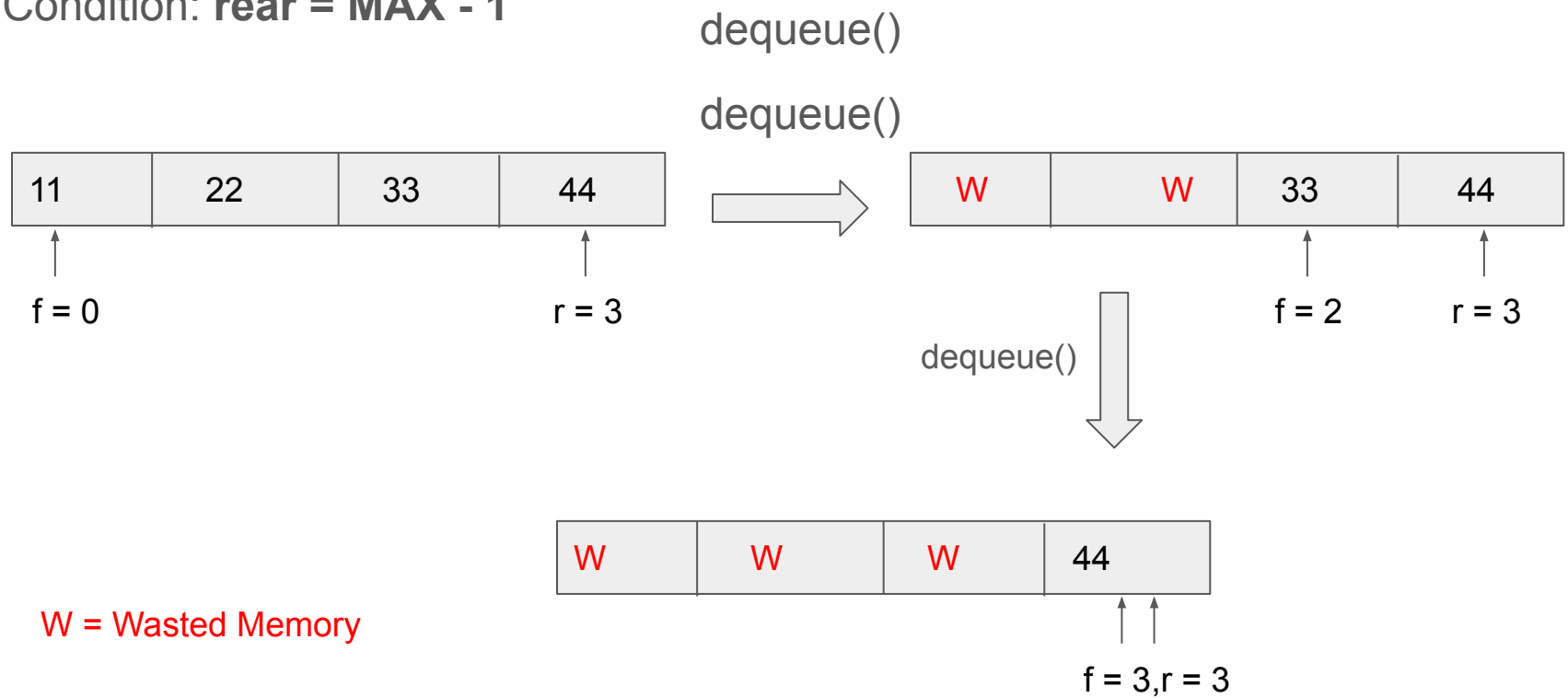


- For above initial condition:
  - Queue Full : **rear = MAX - 1**
  - Queue Empty : **front > rear**



Queue Full

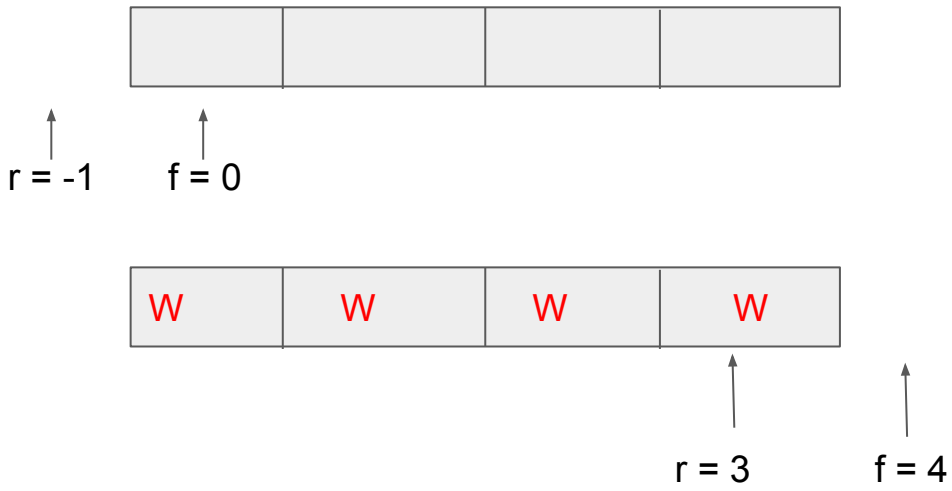
Condition: **rear = MAX - 1**



W = Wasted Memory

Queue Empty

Condition: **front > rear**



W = Wasted Memory

Enqueue

Step 1: Start

Step 2: Check if the queue is full: rear = MAX - 1

Step 3: If the queue is full then, display an appropriate message such as queue overflow.

Else Increase the rear and insert the data in the rear position.

Step 4: Stop

Dequeue

Step 1: Start

Step 2: Check if the queue is empty: front > rear

Step 3: If the queue is empty then, display an appropriate message such as queue underflow.

Else delete the element and Increase the front.

Step 4: Stop

Implementation of linear queue in C

```c
#include <stdio.h>
#include<stdlib.h>
#define MAX 5       // queue size

// initialize an empty queue
int linear_queue[MAX];
int rear = -1, front = 0;

//function declaration/prototype
void enqueue();
void dequeue();
void display();

```

Implementation of linear queue in C

```c
//driver function
int main()
{
    int choice;
    do
    {
        printf("\nEnter \n1 for Enqueue \n2 for Dequeue \n3 for Display \n4 for Exit: ");
        scanf("%d", &choice);

        switch(choice)
        {
            case 1:
                enqueue();
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                exit(0);
                break;
            default:
                printf("\nInvalid Choice ...");
        }
    }while(1);

    return 0;
}
```

Implementation of linear queue in C

```c
void enqueue()
{
    int data;
    // queue full condition
    if(rear == MAX - 1)
    {
        printf("\nQueue is full. Queue Overflow");
    }
    else
    {
        printf("\nEnter a data to enqueue: ");
        scanf("%d", &data);
        rear++;
        linear_queue[rear] = data;
    }

}

void dequeue()
{
    //queue empty condition
    if(front > rear)
    {
        printf("\nQueue is empty. Queue Underflow");
    }
    else
    {
        printf("\n%d is popped from the queue.", linear_queue[front]);
        front++;
    }
}

```

Implementation of linear queue in C

```
76
77  void display()
78  {
79      int i;
80      //queue empty condition
81      if(front > rear)
82      {
83          printf("\nQueue is empty");
84      }
85      else
86      {
87          printf("\nThe elements in the queue are : ");
88          for(i=front; i<=rear; i++)
89          {
90              printf("%d\t", linear_queue[i]);
91          }
92      }
93  }
```

Implementation of linear queue in C