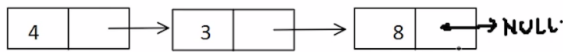


Queue Using Linked Lists



Linked list is like a chain

```

graph LR
    Node1["data: d | next: p"] --> Node2["data: d | next: NULL"]
  
```

data next

45 new Queue Using Linked Lists.mp4 - VLC media player

Media Playback Audio Video Subtitle Tools View Help

Queue Using Linked Lists

Queue Basics

f (Deletion end) r (Insertion end)

Queue:

```

void enqueue(N *f, N *r, int val)
{
    N *n = (N *) malloc(sizeof(N));
    if (n == NULL)
        print("Queue full");
    else {
        n->data = val;
        n->next = NULL;
        if (f == NULL) {
            f = r = n;
        }
        else {
            r->next = n;
            r = n;
        }
    }
}
  
```

Special case!

```

graph TD
    f((f)) --> n["data | next: NULL"]
    r((r)) --> n
  
```

Queue Using Linked Lists

Queue Basics

Diagram illustrating a queue using linked lists. The queue is represented as a sequence of nodes: 4 → 3 → 8 → NULL. The front pointer (f) points to the first node (4), and the rear pointer (r) points to the last node (8). A new node (n) is being added to the end of the queue.

Code:

```

void enqueue(N *f, N *r, int val)
{
    N *n = (N *) malloc(sizeof(N));
    if (n == NULL)
        print("Queue full");
    else {
        n->data = val;
        n->next = NULL;
        if (f == NULL) {
            f = r = n;
        }
        else {
            r->next = n;
            r = n;
        }
    }
}
    
```

Special Case: A diagram showing a node with f and r both pointing to it, labeled "Special Case".

Queue Using Linked Lists

Queue Basics

Diagram illustrating a queue using linked lists. The queue is represented as a sequence of nodes: 4 → 3 → 8 → NULL. The front pointer (f) points to the first node (4), and the rear pointer (r) points to the last node (8). A new node (n) is being added to the end of the queue.

Code:

```

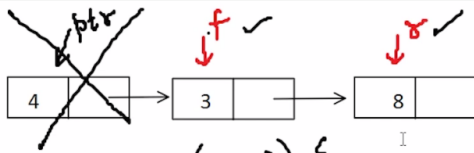
void enqueue(N *f, N *r, int val)
{
    N *n = (N *) malloc(sizeof(N));
    if (n == NULL)
        print("Queue full");
    else {
        n->data = val;
        n->next = NULL;
        if (f == NULL) {
            f = r = n;
        }
        else {
            r->next = n;
            r = n;
        }
    }
}
    
```

Special Case: A diagram showing a node with f and r both pointing to it, labeled "Special Case".

Conditions:

- [A] Condition for Queue Empty: $f == NULL$
- [B] Condition for Queue Full: $n == NULL$

Queue Using Linked Lists



```
int dequeue (N *f) {
```

```
    int val = -1;
```

```
    N *ptr = f;
```

```
    //check for Queue Empty
```

```
    [ f = f->next;
```

```
    val = ptr->data;
```

```
    free(ptr);
```

```
    return val;
```

→ If not Empty

Queue Basics



f
deletion end)

r (Insertion end)

→ Queue: