

Pertemuan 9

QUEUE (ANTREAN)



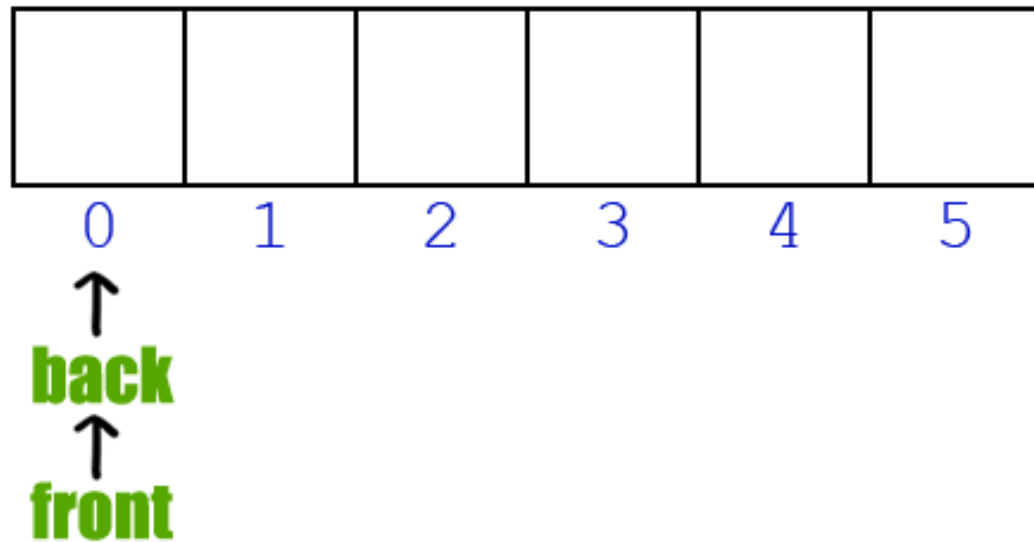
PENGERTIAN QUEUE (ANTREAN)

Struktur Data Antrean (Queue) adalah suatu bentuk khusus dari List Linier dengan operasi pemasukan data hanya diperbolehkan pada salah satu sisi, yang disebut sisi Belakang / ekor (Tail) dan operasi penghapusan hanya diperbolehkan pada sisi lainnya yang disebut sisi Depan / kepala (Head) dari LinkedList.

Prinsip Antrean : FIFO (First In First Out)
FCFS (First Come First Serve)

“Yang Tiba lebih awal Maka akan dilayani Terlebih Dahulu”

Deklarasi Queue



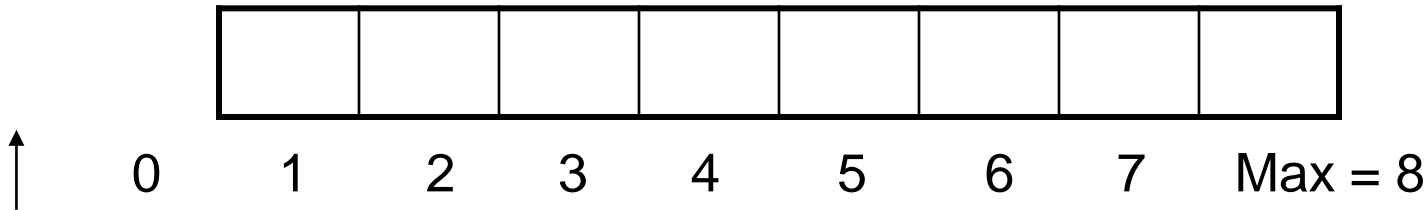
OPERASI QUEUE

- **PEEK ()**
Digunakan untuk mendapatkan elemen yang terdapat pada posisi terakhir tanpa menghapus element tersebut.
- **ISEMPTY()**
Untuk memeriksa apakah queue kosong
- **SIZE()**
mengembalikan jumlah item di dalam list. Tidak memerlukan parameter dan mengembalikan suatu integer.
- **ENQUEUE()**
Untuk menambahkan item pada posisi paling belakang
- **DEQUEUE()**
Untuk menghapus item dari posisi paling depan

Inisialisasi Head

Digunakan untuk membentuk dan menunjukan awal terbentuknya suatu Antrean / Queue

```
def __init__(self):  
    self.head = Node("head")  
    self.size = 0
```



Antrian pertama kali

Fungsi IsEmpty

- Untuk memeriksa apakah Antrian penuh atau kosong
- Head adalah tanda untuk kepala antrian (elemen pertama dalam antrian) yang tidak akan berubah-ubah
- Pergerakan pada Antrian terjadi dengan penambahan elemen Antrian kebelakang,

Fungsi isEmpty (Lanjutan)

```
def isEmpty(self):  
    return self.size == 0
```



↑
0 1 2 3 4 5 6 7 Max = 8

Antrian kosong

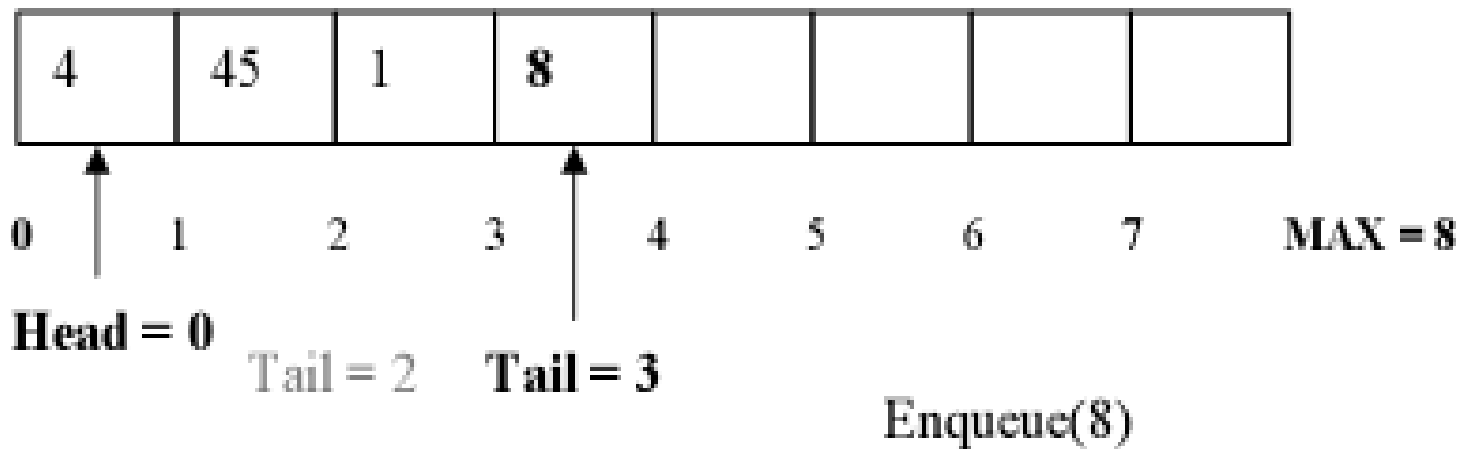
Fungsi Peek

Periksa antrian untuk melihat apakah ada antrian kosong

```
def peek(self):  
  
    if self.isEmpty():  
        raise Exception("Peeking from an empty stack")  
    return self.head.next.value
```


Fungsi Enqueue

- Untuk menambahkan elemen ke dalam Antrian, penambahan elemen selalu dilakukan pada elemen paling **belakang**
- Penambahan elemen selalu menggerakkan variabel Tail dengan cara menambahkan Tail terlebih dahulu



Fungsi Enqueue (Lanjutan)

Menambahkan nilai pada antrian

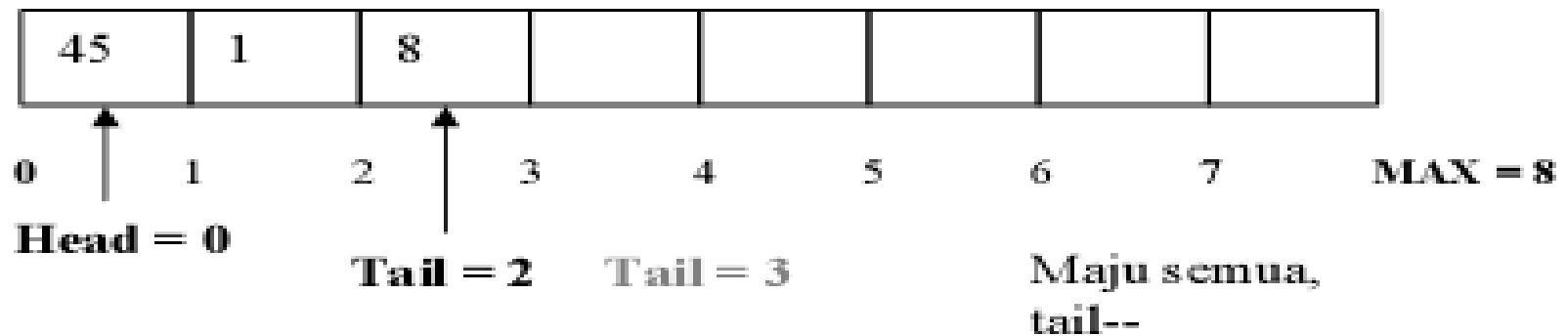
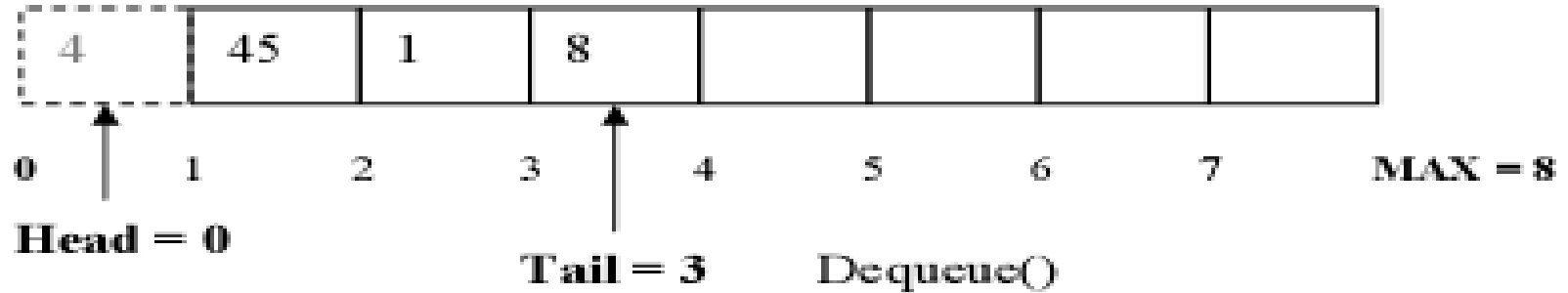
```
def enqueue(self, value):  
    node = Node(value)  
    node.next = self.head.next  
    self.head.next = node  
    self.size += 1
```

Fungsi Dequeue

- Digunakan untuk menghapus elemen terdepan (head) dari Antrian
- Dengan cara : menggeser semua elemen antrian kedepan dan mengurangi Tail dgn 1. Penggeseran dilakukan dengan menggunakan looping

```
def dequeue(self):  
    if self.isEmpty():  
        raise Exception("dequeue from an empty stack")  
    remove = self.head.next  
    self.head.next = self.head.next.next  
    self.size -= 1  
    return remove.value
```

Fungsi Dequeue (Lanjutan)



Fungsi SIZE

digunakan untuk mengetahui banyak elemen atau data yang ada di dalam list.

```
def getSize(self):  
    return self.size
```

Latihan

Berikan gambaran/ilustrasi dari kasus antrian berikut :

- a) Diketahui suatu Antrian/queue dgn $\text{max} = 6$.
- b) Lakukan Enqueue 4 elemen ke dalam antrian, dimanakah posisi Head dan Tail ?
- c) Kemudian lakukan Dequeue 2 elemen dari antrian. Maka dimana posisi Head dan Tail ?
- d) Dari keadaan diatas, bagaimanakah kondisi penuh dan IsEmpty nya ?

```
class Node:
```

```
    def __init__(self, value):  
        self.value = value  
        self.next = None
```

```
class Stack:
```

```
    # Initializing a stack.  
    # Use a dummy node, which is  
    # easier for handling edge cases.  
    def __init__(self):  
        self.head = Node("head")  
        self.size = 0  
  
    # String representation of the stack  
    def __str__(self):  
        cur = self.head.next  
        out = ""
```

```
        while cur:
```

```
            out += str(cur.value) + "->"  
            cur = cur.next  
        return out[:-3]
```

```
# Get the current size of the stack
```

```
def getSize(self):  
    return self.size
```

```
# Check if the stack is empty
```

```
def isEmpty(self):  
    return self.size == 0
```

```
# Get the top item of the stack
```

```
def peek(self):
```

```
    # Sanitary check to see if we  
    # are peeking an empty stack.  
    if self.isEmpty():  
        raise Exception("Peeking from an  
empty stack")
```

```
return self.head.next.value
```

```
# Push a value into the stack.
```

```
def push(self, value):
```

```
    node = Node(value)
```

```
    node.next = self.head.next
```

```
    self.head.next = node
```

```
    self.size += 1
```

```
# Remove a value from the stack and return.
```

```
def pop(self):
```

```
    if self.isEmpty():
```

```
        raise Exception("Popping from an empty  
stack")
```

```
    remove = self.head.next
```

```
    self.head.next = self.head.next.next
```

```
    self.size -= 1
```

```
    return remove.value
```

```
# Driver Code
```

```
if __name__ == "__main__":
```

```
    stack = Stack()
```

```
    for i in range(1, 11):
```

```
        stack.push(i)
```

```
    print(f"Stack: {stack}")
```

```
    for _ in range(1, 6):
```

```
        remove = stack.pop()
```

```
        print(f"Pop: {remove}")
```

```
    print(f"Stack: {stack}")
```


Latihan Soal Struktur Data (Pertemuan 9)

1. Prinsip antrian adalah ?
 - a. First In First Out
 - b. First Out First In
 - c. Never In First Out
 - d. Last Out Last In
 - e. Last In First Out

2. Bentuk deklarasi manipulasi linked list harus menggunakan suatu pointer penunjuk ke node pertama (Head) dalam linked list yaitu?
 - a. `self.head = Node("head")`
 - b. `Head *TNode`
 - c. `Self.print = Node("head")`
 - d. `self.head = node`
 - e. `TNode +head`

2. Bentuk deklarasi manipulasi linked list harus menggunakan suatu pointer penunjuk ke node pertama (Head) dalam linked list yaitu?
 - a. `self.head = Node("head")`
 - b. `Head *TNode`
 - c. `Self.print = Node("head")`
 - d. `self.head = node`
 - e. `TNode +head`

3. Yang tidak termasuk dalam operasi antrian, adalah ...
 - a. Clear
 - b. Enqueue
 - c. Peek
 - d. Size
 - e. Dequeue

3. Yang tidak termasuk dalam operasi antrian, adalah ...
 - a. Clear
 - b. Enqueue
 - c. Peek
 - d. Size
 - e. Dequeue

4. Menghapus elemen dari antrian dilakukan dari posisi :
 - a. Tengah / Middle
 - b. Depan / Head
 - c. Bawah / bottom
 - d. Belakang / Tail
 - e. Atas / Top

4. Menghapus elemen dari antrian dilakukan dari posisi :
 - a. Tengah / Middle
 - b. Depan / Head
 - c. Bawah / bottom
 - d. Belakang / Tail
 - e. Atas / Top

5. Maksud dari perintah program
node.next = self.head.next
self.head.next = node
self.size += 1
 - a. Menambah elemen antrian
 - b. Mengecek kondisi antrian kosong atau tidak
 - c. Mengecek kondisi antrian penuh atau tidak
 - d. Membentuk atau menghapus semua elemen antrian
 - e. Menghapus elemen antrian

5. Maksud dari perintah program

node.next = self.head.next

self.head.next = node

self.size += 1

a. Menambah elemen antrian

b. Mengecek kondisi antrian kosong atau tidak

c. Mengecek kondisi antrian penuh atau tidak

d. Membentuk atau menghapus semua elemen antrian

e. Menghapus elemen antrian

1. Prinsip antrian adalah?

a. First In First Out

b. First Out First In

c. Never In First Out

d. Last Out Last In

e. Last In First Out