Queues

Indah Agustien Siradjuddin

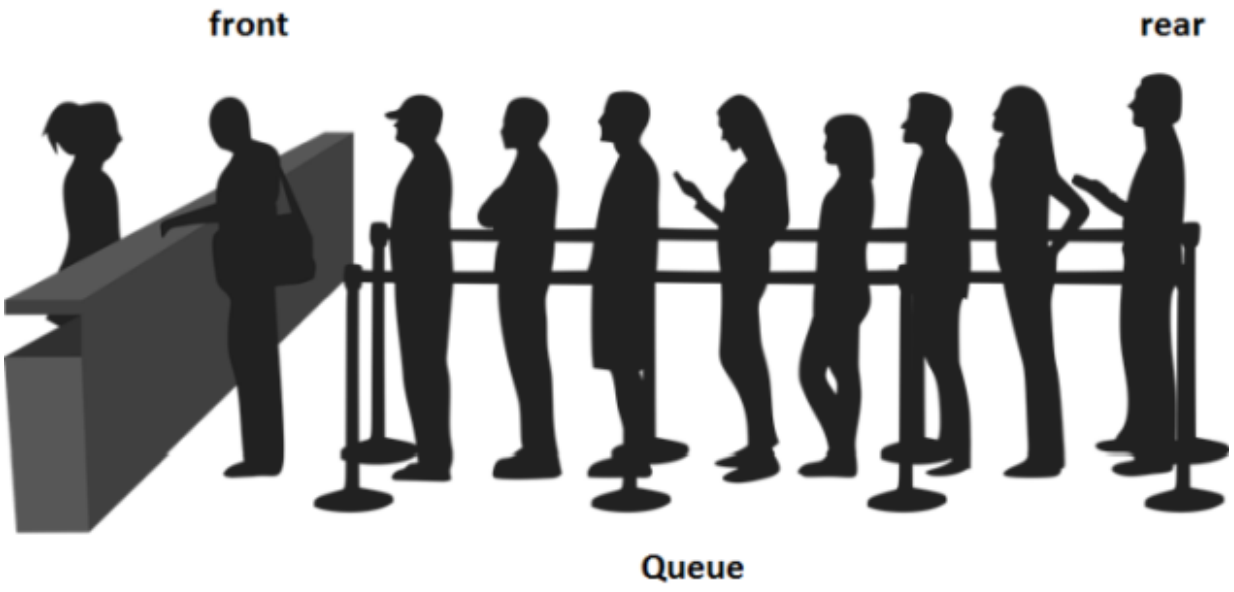
Struktur Data

Struktur data kedua adalah *Queues*, yang terdiri dari :

1. Queues
2. Operasi Queues
3. Contoh Implementasi Queues

Definisi Queues

Queues atau antrian merupakan struktur data dimana penambahan data baru dan penghapusan data berada di ujung yang berbeda. Hal ini berbeda dengan stacks, dimana penambahan data baru dan penghapusan data, dilakukan pada ujung yang sama.
Pada Queues, seperti halnya antrian, penambahan data baru dilakukan di suatu ujung atau yang dikenal dengan nama **rear**, dan penghapusan data dilakukan pada ujung yang dikenal dengan nama **front**, seperti yang terlihat pada Gambar 1 berikut :



Gambar 1. Queues atau Antrian

Jika konsep penambahan atau penghapusan data pada stacks dikenal dengan nama LIFO (Last In First Out), maka pada Queues menggunakan konsep **FIFO (First In First Out)**

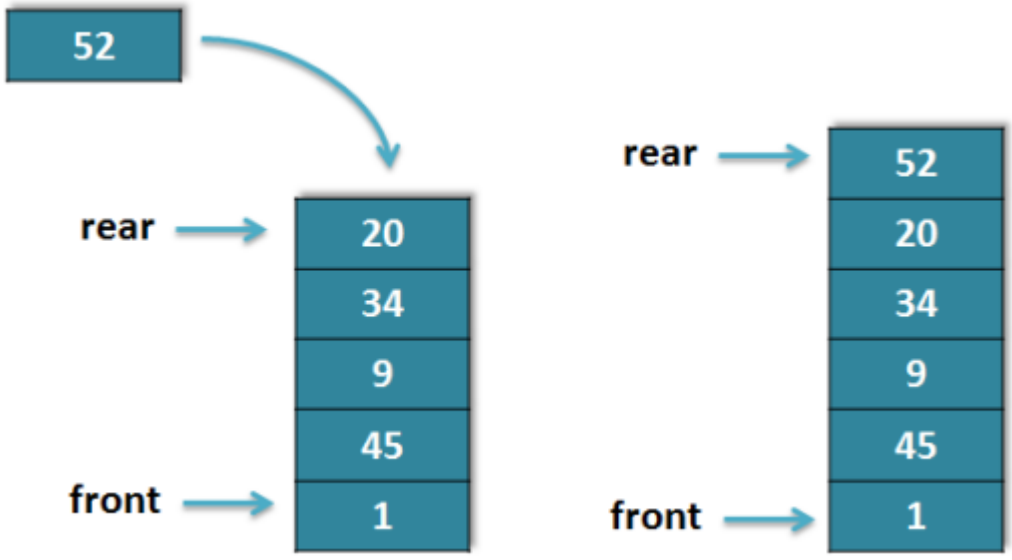
Kembali ke Menu Awal

Operasi pada Queues

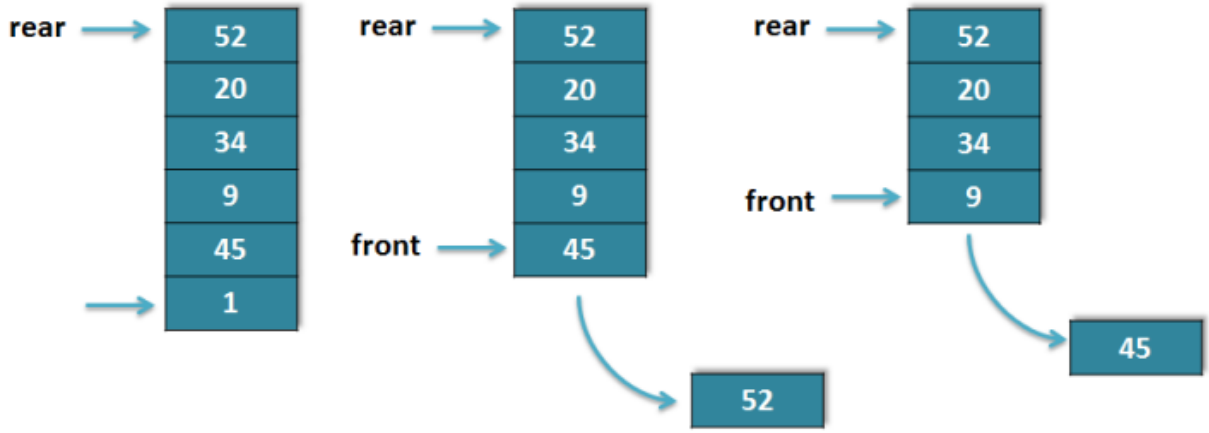
Terdapat beberapa operasi dasar pada struktur data Queues ini, antara lain :

```
- queue (),          inisialisasi struktur data queue kosong
- enqueue (data),    penambahan *data* baru pada queue
- dequeue (),         penghapusan data
- isEmpty(),         pengecekan apakah queue dalam keadaan kosong
- size (),           informasi jumlah data yang terdapat pada queue
```

Ilustrasi penambahan dan penghapusan data dapat dilihat pada Gambar 2 dan Gambar 3 berikut :



Gambar 2. Operasi enqueue pada queues



Gambar 3. Operasi dequeue pada queues

Pada Gambar 2 ditunjukkan bahwa penambahan data baru (enqueue) dilakukan pada ujung data 'rear', sedangkan penghapusan data (dequeque) dilakukan pada ujung data 'front', karena sifat queues yang FIFO.

Implementasi queues dapat dilakukan pada tipe data lists. Data pada **indeks terakhir** pada list adalah merupakan data yang terletak pada ujung **front** pada queues. Sehingga proses dequeue dilakukan dengan menggunakan method pop pada list.
Sedangkan data pada indeks awal, atau indeks ke-0 dari list, menunjukkan data yang terletak pada posisi **rear** dari queues. Oleh karena itu proses enqueue dilakukan dengan method insert pada posisi 0 dari list.

Misalkan terdapat data berbentuk list, sebagai berikut :

```
data=[8, 3, 9, 2]
```

Jika dilakukan `data.insert(0,74)` , atau penambahan data pada posisi **rear**, maka data akan berubah menjadi :

```
data=[74, 8, 3, 9, 2]
```

Jika dilakukan `data.pop()` , atau penghapusan data pada posisi **front**, maka data akan berubah menjadi :

```
data=[74, 8, 3, 9]
```

Code

Berikut adalah code-code untuk pembentukan queues dan operasi-operasi yang terdapat pada queues

In [1]:
```
def createQueue():
    q=[]
    return (q)
def enqueue(q,data):
    q.insert(0,data)
    return(q)
def dequeue(q):
    data=q.pop()
    return(data)
def isEmpty(q):
    return (q==[])
def size(q):
    return (len(q))
```

Berikut adalah contoh penggunan struktur queues yang telah dibuat

In [2]:
```
q=createQueue()
```

In [3]:
```
enqueue(q,'matematika')
enqueue(q,'struktur data')
enqueue(q,'bahasa inggris')
enqueue(q,'pemrograman web')
print(q)
```

```
['pemrograman web', 'bahasa inggris', 'struktur data', 'matematika']
```

In [ ]:
```
temp=dequeue(q)
print(q)
print(temp)
```

In [ ]:
```
enqueue(q,dequeue(q))
print(q)
```

In [ ]:
```
a=dequeue(q)
print(a,q)
```

In [ ]:
```
enqueue(q,'new')
```

In [ ]:
```
isEmpty(q)
```

In [ ]:
```
size(q)
```

[Kembali ke Menu Awal]

Implementasi-Queues

Contoh implementasi queues ini adalah permainan anak-anak, yaitu permainan ular naga. Pada permainan ini, sekelompok anak berbaris dan dan terdapat dua buah anak sebagai penjaga, yang bertugas untuk 'menangkap anak' seiring dengan lagu 'ular naga' berakhir. Permainan ini akan berakhir jika hanya terdapat satu anak yang memenangkan permainan, yaitu anak tersebut lolos dari tangkapan penjaga, seperti yang terlihat pada Gambar 4

Gambar 4. Permainan Ular Naga

Berikut code simulasi dari permainan tersebut, dengan sedikit modifikasi, yaitu fungsi penjaga dan lagu 'ular naga' digantikan dengan bilangan tertentu, dan sekolompok anak disusun dalam suatu antrian. Permainan akan terus berjalan, sampai dengan hitungan bilangan tertentu tersebut. Jika pada hitungan tertentu, terdapat anak yang berada di ujung antrian (*front*), maka anak tersebut harus keluar barisan antrian. Permainan dimenangkan oleh seorang anak yang tersisa di dalam antrian.

Code

In [ ]:
```
def ularNaga(nama, hitungan):
    gameQueue = createQueue()
    for namaAnak in nama:
        enqueue(gameQueue,namaAnak)
    print('Peserta Permainan=',gameQueue)
    while size(gameQueue) > 1:
        for i in range(hitungan):
            enqueue(gameQueue,dequeue(gameQueue))
            print('hitungan ke-',i,'=',gameQueue)
        dequeue(gameQueue)
        print('Peserta Permainan=',gameQueue)
    return dequeue(gameQueue)
```

In [ ]:
```
ularNaga(['andi','rita','sari','anton','rafa','diana','zaki'],2)
```

[Kembali ke Menu Awal]

In [ ]: