INF218
Struktur Data & Algoritma

Queue

Alim Misbullah, S.Si., M.S.

Definisi Queue

Definisi Queue

- **Queue** juga merupakan sebuah Abstract Data Type (ADT) yang digunakan pada bahasa pemrograman, seperti halnya stack
- Dinamai dengan **Queue** karena sesuai dengan aplikasi di dunia nyata seperti antrian pada pembelian tiket, bus stop di halte, dan lain-lain

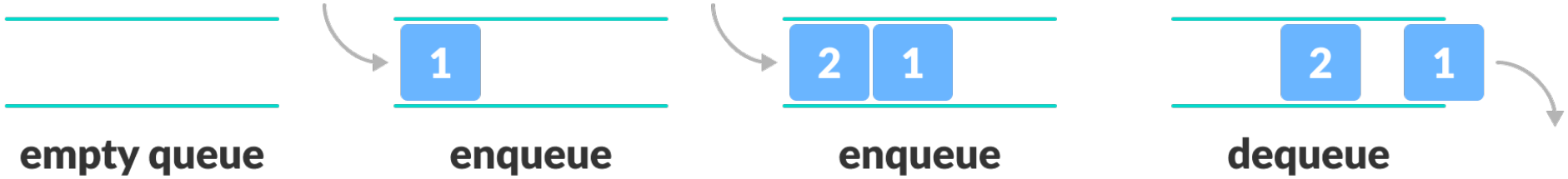
- Pada kehidupan nyata, operasi queue berlaku pada kedua sisi yaitu satu bagian untuk menambahkan data **(enqueue)** dan satu bagian lainnya untuk mengeluarkan data **(dequeue)**

Definisi Queue

- Queue disebut sebagai **FIFO (First In First Out)** data structure yang berarti bahwa elemen yang ditambahkan pertama akan diakses pertama juga

- Operasi penambahan elemen disebut dengan **enqueue()** dan operasi penghapusan elemen disebut dengan **dequeue()**

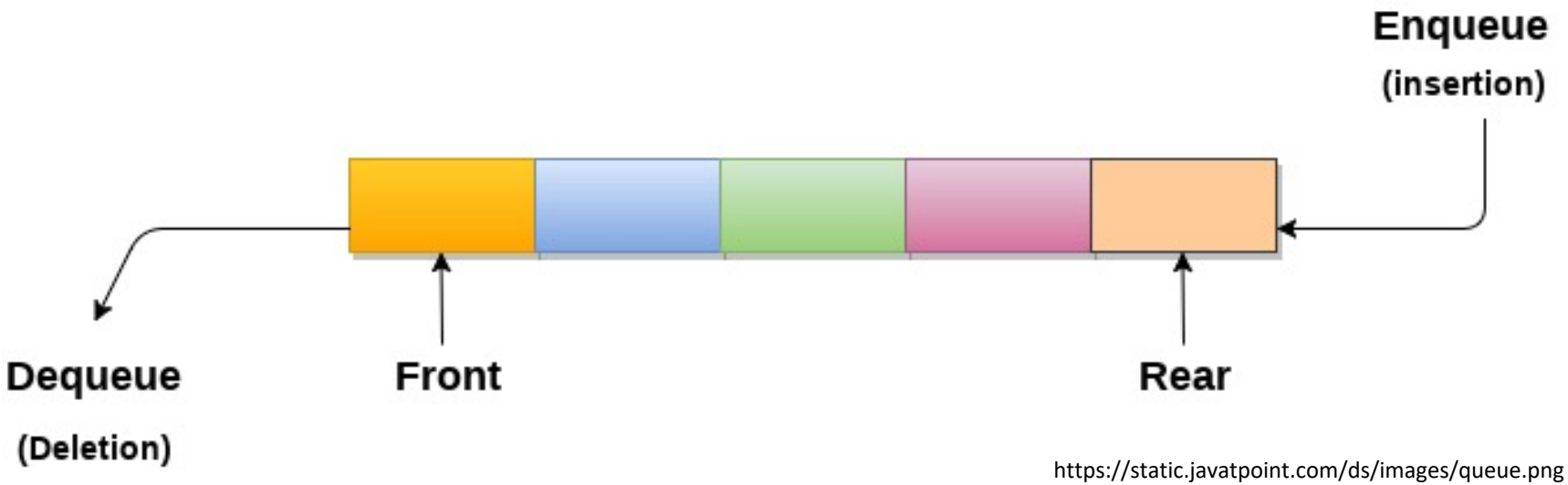
Definisi Queue



- Pada gambar diatas, nilai 1 berada diantrian pertama sebelum nilai 2 maka nilai 1 akan diakses lebih dulu dari sebuah queue, sesuai dengan aturan **FIFO**

Proses pada Queue

- Proses pada queue dilakukan dengan:
  - Menggunakan 2 pointer yaitu **FRONT** dan **REAR**
  - **FRONT** akan menunjuk data elemen pertama dari queue
  - **REAR** akan menunjuk data elemen terakhir dari queue
  - Nilai awal untuk **FRONT** dan **REAR** adalah **-1**



https://static.javatpoint.com/ds/images/queue.png

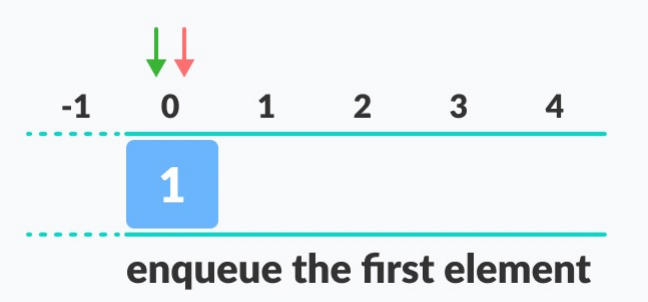
Operasi Pada Queue

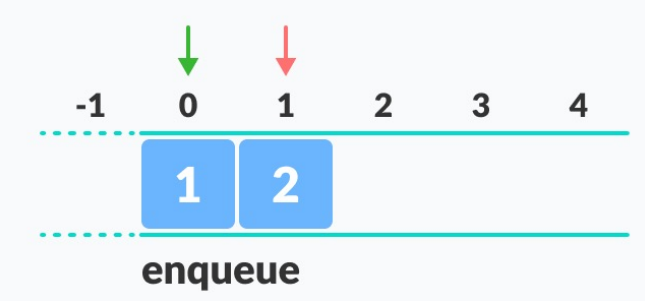
Operasi Enqueue() dan Dequeue()

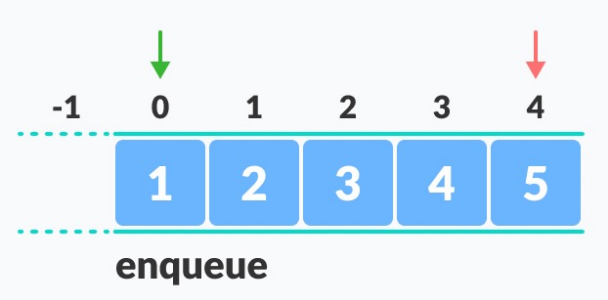
- Operasi **Enqueue** → proses untuk menambahkan data elemen pada bagian akhir (rear) dari sebuah queue. Langkah-langkahnya:
  - Langkah 1: Cek apakah queue sudah penuh
  - Langkah 2: Untuk data elemen pertama, ubah nilai index dari **FRONT** dan **REAR** ke 0
  - Langkah 3: Untuk data elemen selanjutnya, naikkan nilai index **REAR** menjadi 1
  - Langkah 4: Tambahkan data elemen pada posisi yang ditunjuk oleh **REAR**

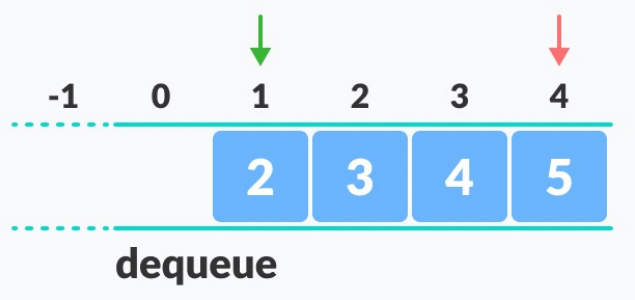
- Operasi **Dequeue** → proses untuk menghapus data elemen pada bagian depan (front) dari sebuah queue. Langkah-langkahnya:
  - Langkah 1: Cek apakah queue kosong
  - Langkah 2: Kembalikan nilai (elemen) yang ditunjuk oleh **FRONT**
  - Langkah 3: Naikkan nilai index **FRONT** sebanyak 1
  - Langkah 4: Untuk data element terakhir, reset nilai index **FRONT** dan **REAR** menjadi -1

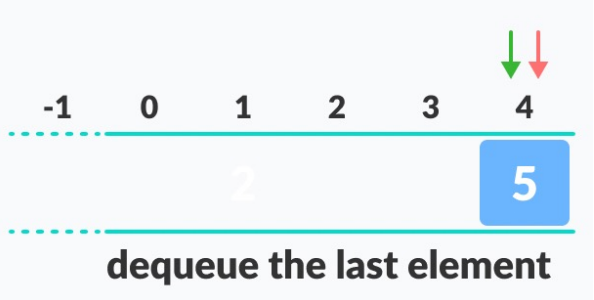
Operasi Enqueue() dan Dequeue

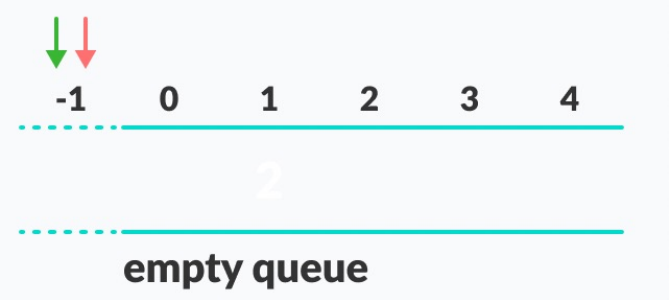












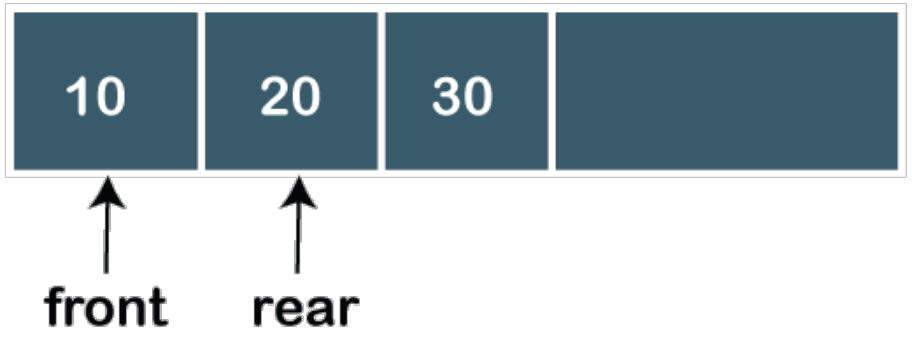
Demo Simple Queue

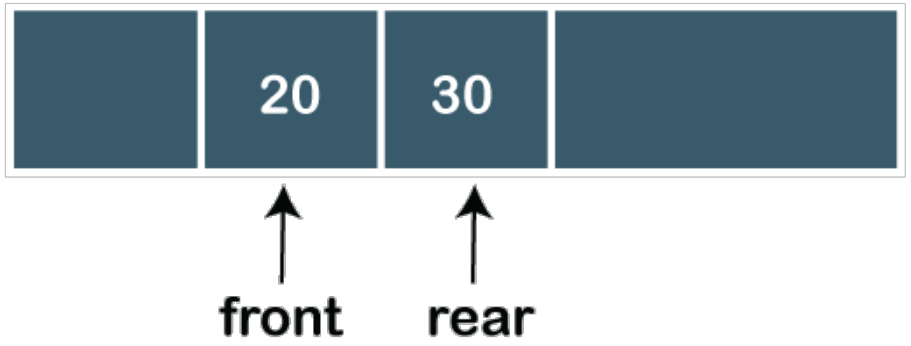
- C
  - https://www.programiz.com/dsa/queue#c-code
  - https://www.tutorialspoint.com/data_structures_algorithms/queue_program_in_c.htm
- Java
  - https://www.programiz.com/dsa/queue#java-code
- Python
  - https://www.programiz.com/dsa/queue#python-code
- C++
  - https://www.programiz.com/dsa/queue#cpp-code
- Kompleksitas operasi enqueue() dan dequeue() pada sebuah queue menggunakan array adalah **O(1)**

Jenis-Jenis Queue

Linear Queue (Simple Queue)

- Pada Linear Queue, penambahan data elemen dilakukan dari salah satu bagian queue yaitu **rear** dan penghapusan data elemen dilakukan dari bagian queue lainnya yaitu **front**
- Linear queue sangat mengikuti aturan **FIFO** sebagaimana aturan dari sebuah queue



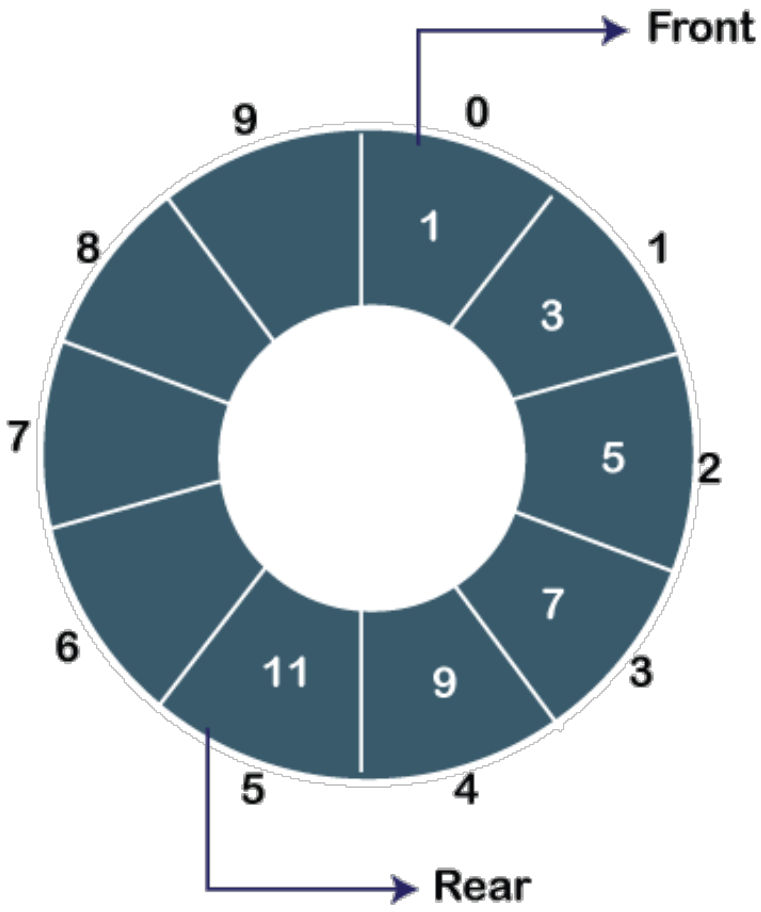


Linear Queue (Simple Queue)

- Pada Linear Queue, jika sebuah data elemen ditambahkan maka nilai indeks dari pointer **REAR** akan dinaikkan sebesar 1
- Sedangkan, jika sebuah data elemen dihapus, maka nilai indeks dari pointer **FRONT** akan dinaikkan sebesar 1
- **Kekurangan dari Linear Queue** adalah proses penambahan data elemen hanya berlaku pada sisi **REAR** saja sehingga jika 3 data elemen dihapus dari queue maka data elemen baru tidak dapat ditambahkan meskipun space masih tersedia di dalam linear queue. Dalam kasus ini, linear queue akan mengalami **overflow** karena pointer **REAR** selalu menunjuk data elemen terakhir dari queue

Circular Queue

- Pada Circular Queue, semua node data elemen direpresentasikan seperti lingkaran, sehingga data elemen terakhir dari queue akan terhubung ke data elemen pertama dari queue tersebut.
- Keuntungan dari Circular Queue adalah **memory utilization** yaitu jika posisi terakhir **(rear)** dari queue penuh dan posisi pertama **(front)** dari queue kosong maka data elemen dapat ditambahkan pada posisi pertama



Circular Queue

Algorithm to insert an element in a circular queue

Step 1: IF (REAR+1)%MAX = FRONT
Write " OVERFLOW "
Goto step 4
[End OF IF]

Step 2: IF FRONT = -1 and REAR = -1
SET FRONT = REAR = 0
ELSE IF REAR = MAX - 1 and FRONT ! = 0
SET REAR = 0
ELSE
SET REAR = (REAR + 1) % MAX
[END OF IF]

Step 3: SET QUEUE[REAR] = VAL

Step 4: EXIT

Algorithm to delete an element from the circular queue

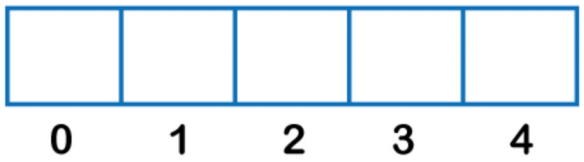
Step 1: IF FRONT = -1
Write " UNDERFLOW "
Goto Step 4
[END of IF]

Step 2: SET VAL = QUEUE[FRONT]

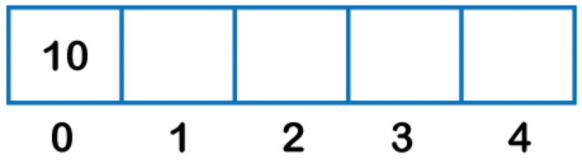
Step 3: IF FRONT = REAR
SET FRONT = REAR = -1
ELSE
IF FRONT = MAX -1
SET FRONT = 0
ELSE
SET FRONT = FRONT + 1
[END of IF]
[END OF IF]

Step 4: EXIT

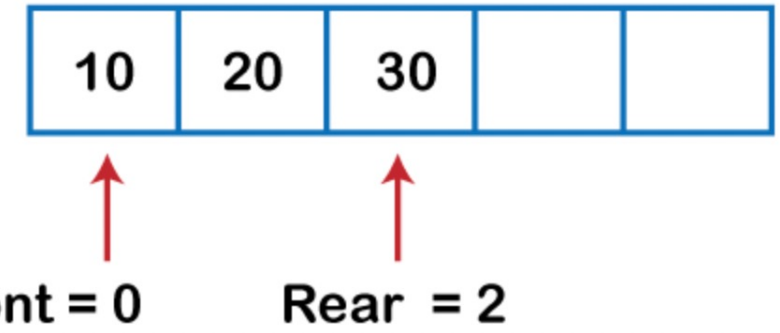
Circular Queue

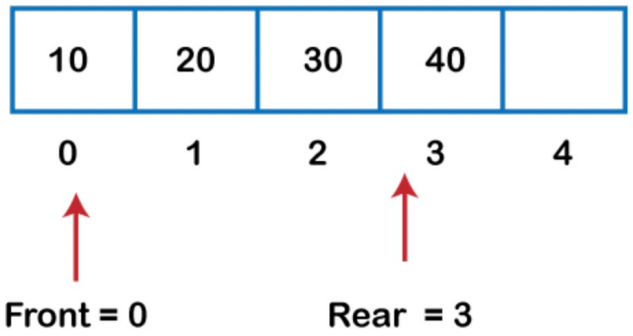


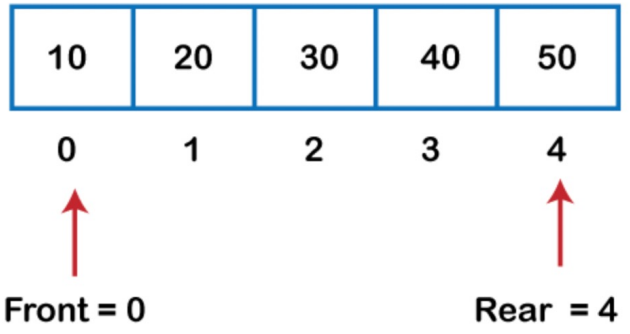
Front = -1
Rear = -1

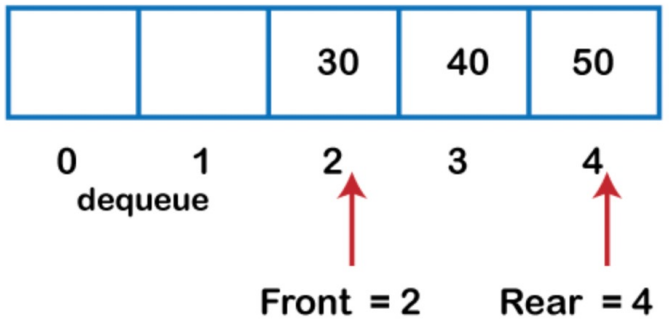


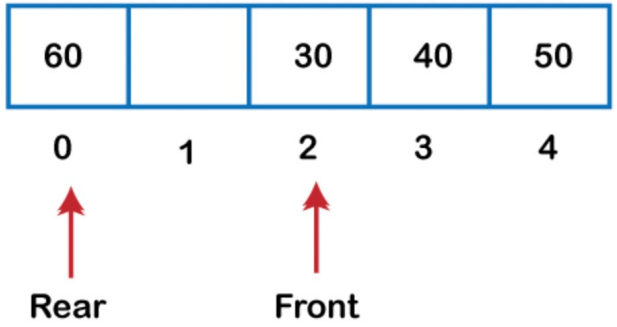
Front = 0
Rear = 0

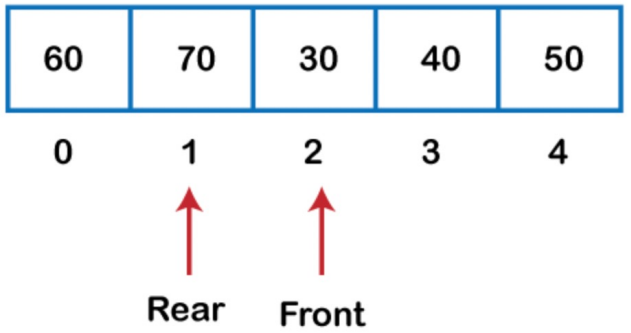










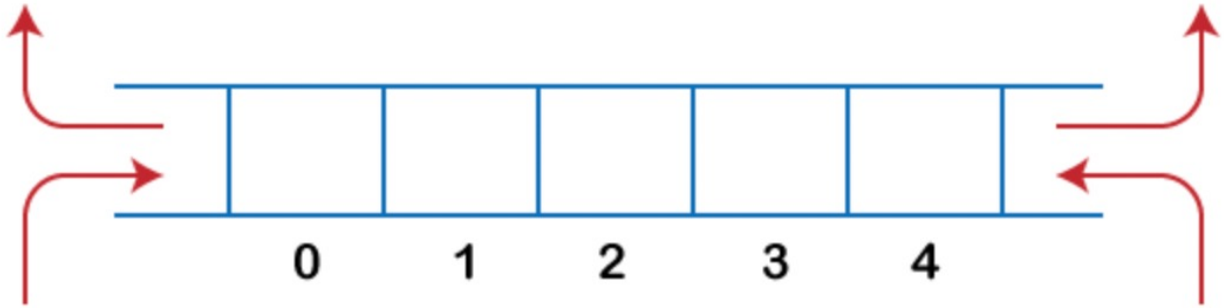


Demo Circular Queue

https://www.javatpoint.com/circular-queue

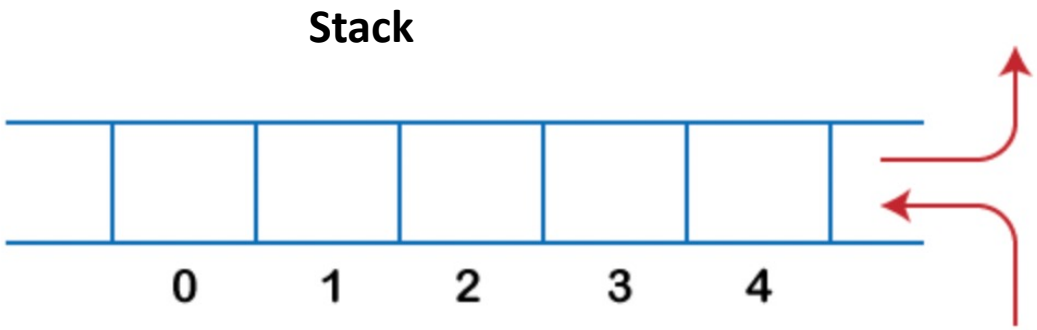
Deque (Double Ended) Queue

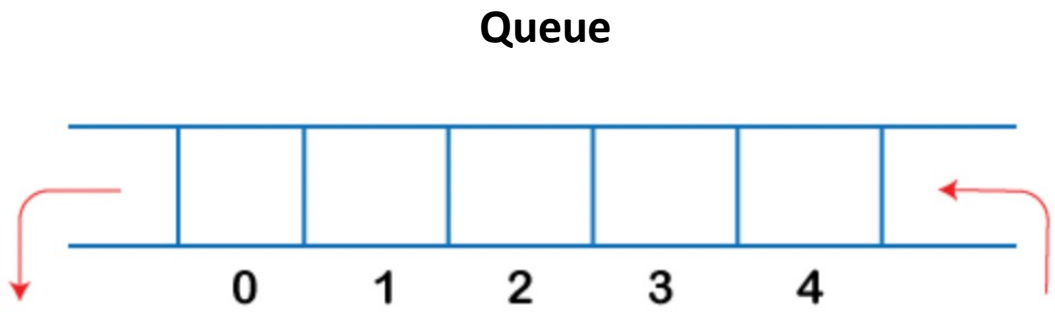
- Pada simple queue, penambahan data elemen dilakukan pada bagian **rear**, sedangkan penghapusan data elemen dilakukan pada bagian **front**
- Pada Deque (Double Ended) Queue, penambahan dan penghapusan data elemen dilakukan pada kedua sisi queue. Dengan kata lain, deque merupakan generalisasi dari sebuah queue
- Deque dapat digunakan sebagai **stack** atau **queue** karena penambahan dan penghapusan data dapat dilakukan pada kedua sisi



Deque (Double Ended) Queue

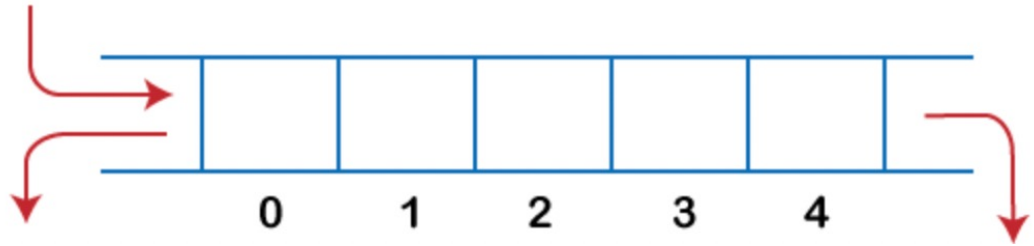
- Pada deque, penambahan dan penghapusan data elemen dapat dilakukan pada satu sisi saja seperti pada **stack**. Sama halnya, penambahan data elemen juga dapat dilakukan pada satu sisi dan penghapusan data elemen pada sisi lainnya seperti pada **queue**



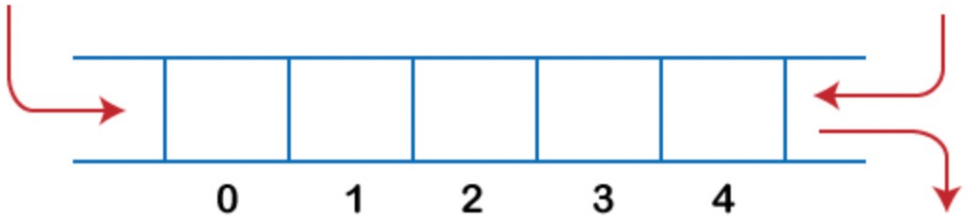


Deque (Double Ended) Queue

- **Input-restricted queue** → penambahan data elemen hanya dilakukan pada salah satu sisi saja, sedangkan penghapusan data elemen dapat dilakukan pada kedua sisi



- **Output-restricted queue** → penghapusan data elemen hanya dilakukan pada salah satu sisi saja, sedangkan penambahan data elemen dapat dilakukan pada kedua sisi

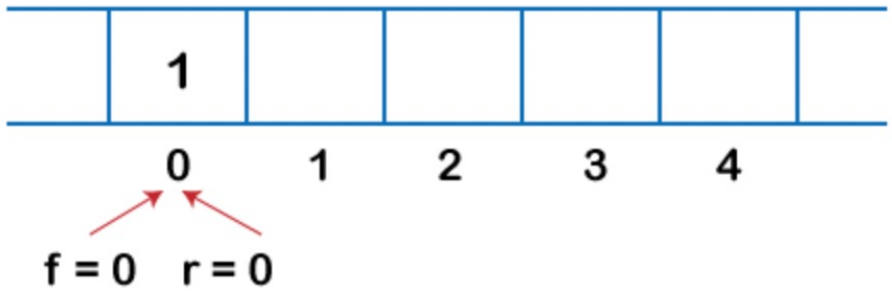


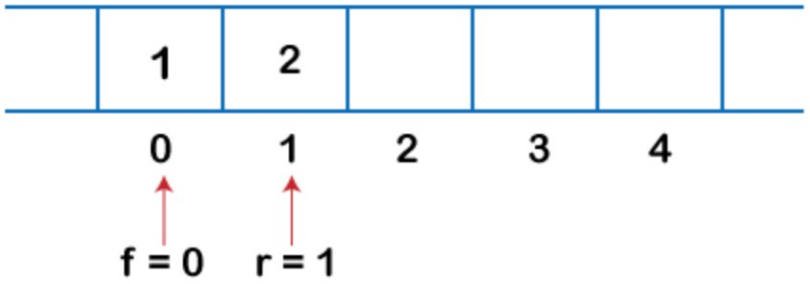
Deque (Double Ended) Queue

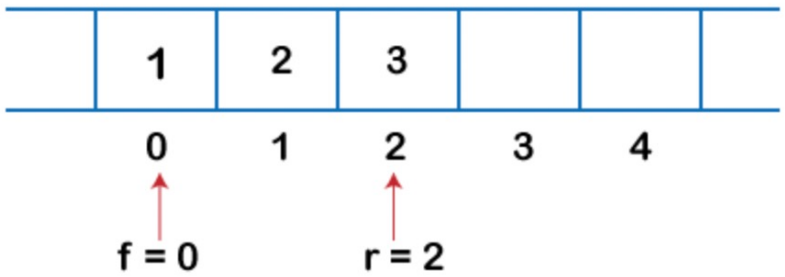
- Deque dapat diimplementasikan menggunakan **circular array** dan **doubly linkedlist**
- **Circular array** adalah sebuah array yang bagian akhir data elemen akan terhubung ke bagian pertama data elemen sehingga tidak akan terjadi **overflow**
- **Operasi pada deque**
  - Insert at front
  - Delete from front
  - Insert at rear
  - Delete from rear

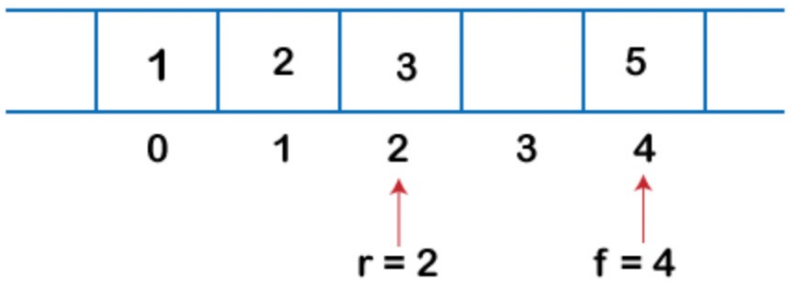
Deque (Double Ended) Queue

- Operasi Enqueue



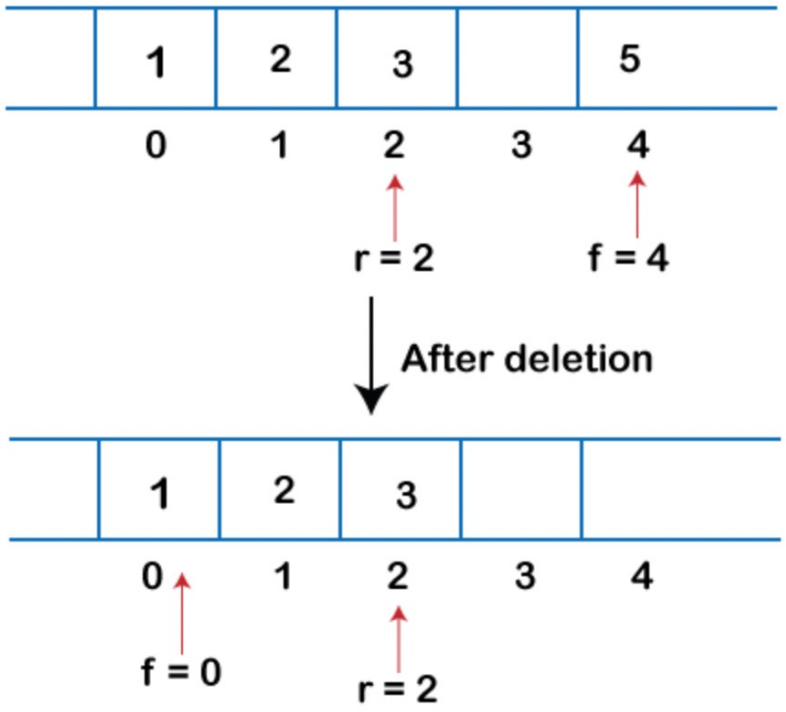


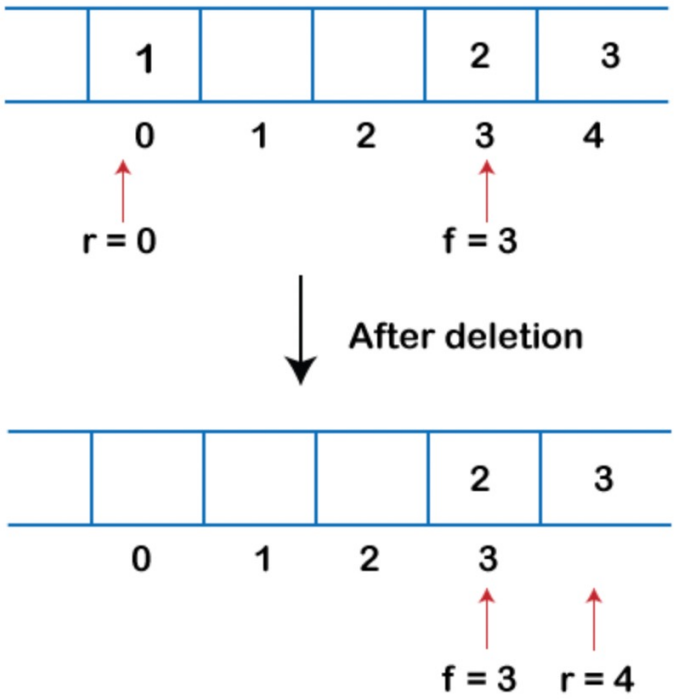


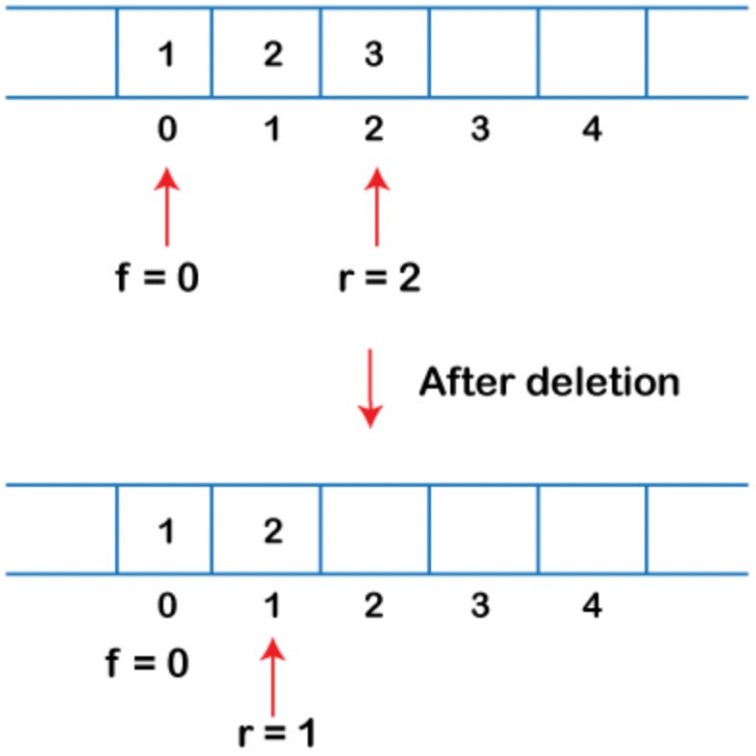


Deque (Double Ended) Queue

- Operasi Dequeue







Demo Deque Queue

https://www.javatpoint.com/ds-deque

Q & A

Referensi

- https://www.tutorialspoint.com/data_structures_algorithms/dsa_queue.htm
- https://www.javatpoint.com/data-structure-queue
- https://www.programiz.com/dsa/queue