

# Queue (Antrian)

IF2110/IF2111 – Algoritma dan Struktur Data  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung



He thinks he has to wait in line to get a treat.

# Queue



**Queue** adalah sederetan elemen yang:

- dikenali elemen pertama (HEAD) dan elemen terakhirnya (TAIL).
- aturan penambahan dan penghapusan elemennya didefinisikan sebagai berikut:  
**Penambahan** selalu dilakukan setelah **elemen terakhir**,  
**Penghapusan** selalu dilakukan pada **elemen pertama**.

# Queue

Elemen Queue tersusun secara **FIFO** (*First In First Out*)

Contoh pemakaian Queue:

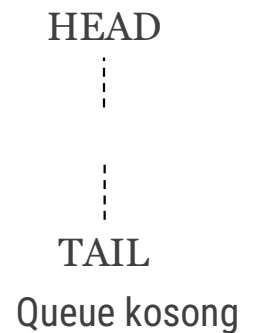
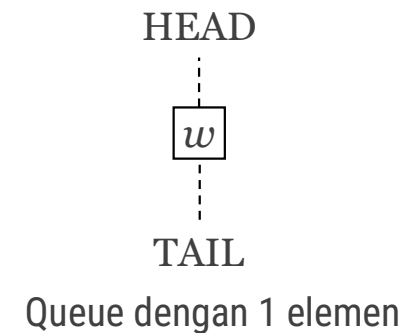
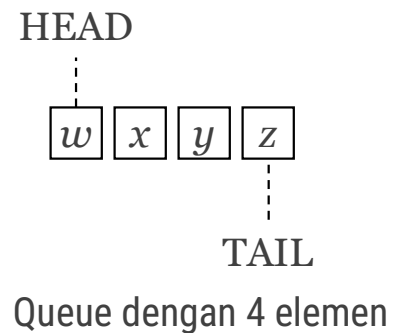
- antrian job yang harus ditangani oleh sistem operasi (*job scheduling*).
- antrian pemrosesan *request* oleh *web server*.
- antrian dalam dunia nyata.

Queue seperti sebuah List dengan batasan lokasi penambahan & penghapusan elemen.

# Queue

Secara logik:

- Elemen
- Head (elemen terdepan)
- Posisi tail (elemen paling belakang)
- Queue kosong



# Definisi operasi

Jika diberikan  $Q$  adalah Queue dengan elemen  $ElmtQ$

$CreateQueue: \rightarrow Q$	{ Membuat sebuah antrian kosong }
$head: Q \rightarrow ElmtQ$	{ Mengirimkan elemen terdepan $Q$ saat ini }
$length: Q \rightarrow \underline{integer}$	{ Mengirimkan banyaknya elemen $Q$ saat ini }
$enqueue: ElmtQ \times Q \rightarrow Q$	{ Menambahkan sebuah elemen setelah elemen paling belakang Queue }
$dequeue: Q \rightarrow Q \times ElmtQ$	{ Menghapus kepala Queue, mungkin $Q$ menjadi kosong }
$isEmpty: Q \rightarrow \underline{boolean}$	{ Tes terhadap $Q$ : true jika $Q$ kosong, false jika $Q$ tidak kosong }

# Axiomatic semantics (fungsional)

- 1)  $\text{new}()$  returns a queue
- 2)  $\text{head}(\text{enqueue}(v, \text{new}())) = v$
- 3)  $\text{dequeue}(\text{enqueue}(v, \text{new}())) = \text{new}()$
- 4)  $\text{head}(\text{enqueue}(v, \text{enqueue}(w, Q))) = \text{head}(\text{enqueue}(w, Q))$
- 5)  $\text{dequeue}(\text{add}(v, \text{enqueue}(w, Q))) = \text{add}(v, \text{dequeue}(\text{enqueue}(w, Q)))$

Di mana  $Q$  adalah Queue dan  $v, w$  adalah value.

# Implementasi Queue dengan array

Memori tempat penyimpan elemen adalah sebuah array dengan indeks  $0..CAPACITY-1$ .

Perlu informasi indeks array yang menyatakan posisi Head dan Tail.



# ADT Queue dengan array

## KAMUS UMUM

constant IDX\_UNDEF: integer = -1

constant CAPACITY: integer = 100

type ElType: integer { *elemen Queue* }

{ *Queue dengan array statik* }

type Queue: < buffer: array [0..CAPACITY-1] of ElType, { *penyimpanan elemen* }  
    idxHead: integer, { *indeks elemen terdepan* }  
    idxTail: integer > { *indeks elemen terakhir* }

# ADT Queue – Konstruktor, akses, & predikat

procedure CreateQueue(output q: Queue)

*{ I.S. Sembarang*

*F.S. Membuat sebuah Queue q yang kosong berkapasitas CAPACITY  
jadi indeksinya antara 0..CAPACITY-1*

*Ciri Queue kosong: idxHead dan idxTail bernilai IDX\_UNDEF }*

function head(q: Queue) → ElType

*{ Prekondisi: q tidak kosong.*

*Mengirim elemen terdepan q, yaitu q.buffer[q.idxHead]. }*

function length(q: Queue) → integer

*{ Mengirim jumlah elemen q saat ini }*

function isEmpty(q: Queue) → boolean

*{ Mengirim true jika q kosong: lihat definisi di atas }*

function isFull(q: Queue) → boolean

*{ Mengirim true jika penyimpanan q penuh }*

# ADT Queue - Operasi

procedure enqueue (input/output q: Queue, input val: ElType)

*{ Menambahkan val sebagai elemen Queue q.*

*I.S. q mungkin kosong, TIDAK penuh*

*F.S. q bertambah elemen val sebagai tail yang baru }*

procedure dequeue (input/output q: Queue, output val: ElType)

*{ Menghapus head dari Queue q.*

*I.S. q tidak kosong*

*F.S. val berisi nilai head yang lama.*

*Jika q tidak menjadi kosong,*

*q.idxHead berpindah ke elemen berikutnya pada q.*

*Jika q menjadi kosong,*

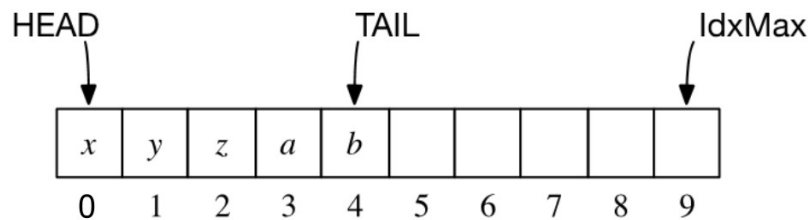
*q.idxHead dan q.idxTail menjadi bernilai IDX\_UNDEF. }*

# Implementasi Queue dengan array – alt-1

Jika Queue tidak kosong: **idxTail** adalah indeks elemen terakhir, **idxHead** selalu diset = 0.

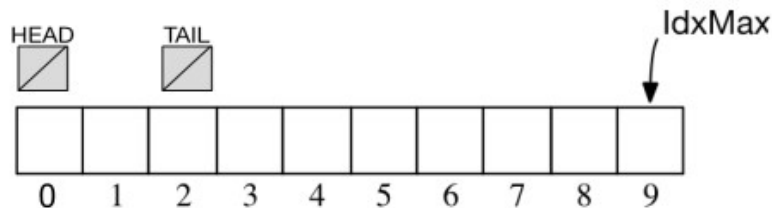
Jika Queue kosong, maka **idxHead** dan **idxTail** diset = **IDX\_UNDEF**.

- Ilustrasi Queue tidak kosong dengan 5 elemen:



\*dengan  $\text{IdxMax} = \text{CAPACITY} - 1$

- Ilustrasi Queue kosong:



# Implementasi Queue dengan array – alt-1

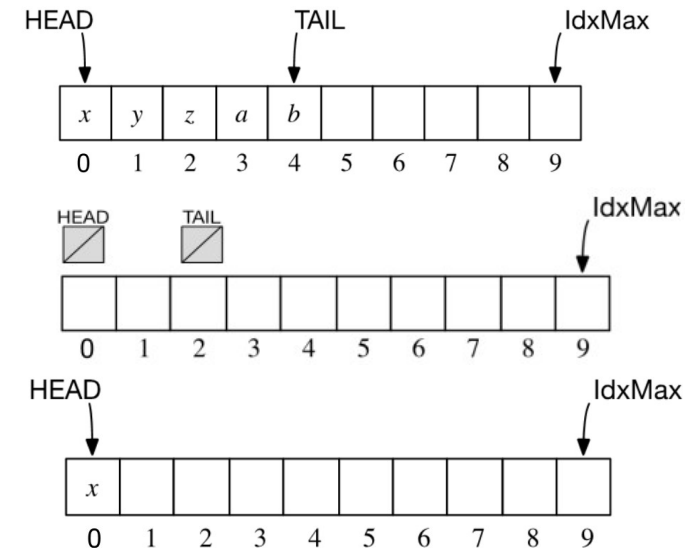
Algoritma **penambahan elemen**:

- **Jika masih ada tempat**: geser TAIL ke kanan.
- **Kasus khusus** (Queue kosong): **idxHead** dan **idxTail** diset = 0.

Algoritma paling sederhana dan “naif” untuk **penghapusan elemen**:

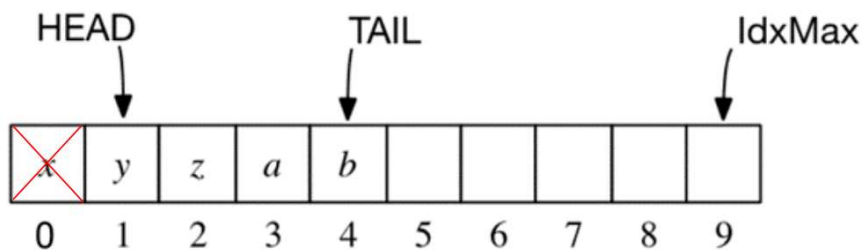
- **Jika Queue tidak kosong**: ambil nilai elemen HEAD, geser semua elemen mulai dari **idxHead+1** s.d. **idxTail**, kemudian geser TAIL ke kiri.
- **Kasus khusus** (Queue berelemen 1): **idxHead** dan **idxTail** diset = **IDX\_UNDEF**.

Algoritma ini mencerminkan pergeseran orang yang sedang mengantri di dunia nyata, tapi tidak efisien.

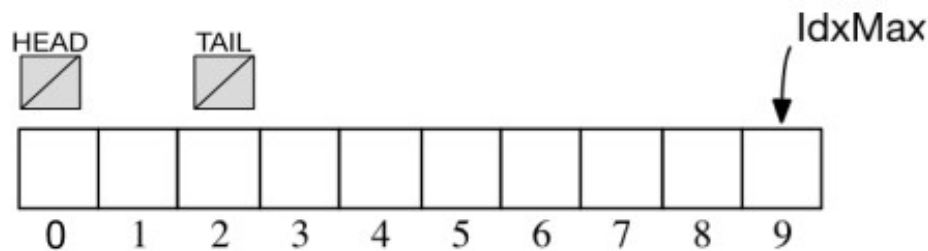


# Implementasi Queue dengan array – alt-2

Tabel dengan representasi HEAD dan TAIL yang mana **HEAD bergeser ke kanan** ketika sebuah elemen dihapus.

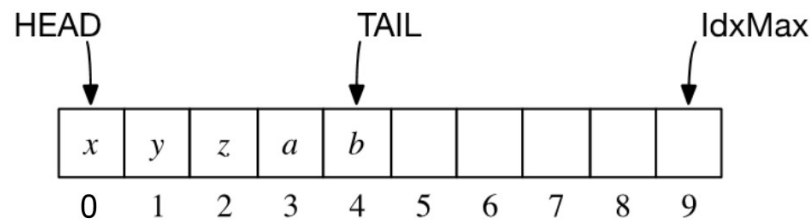


Jika Queue kosong, maka `idxHead` dan `idxTail` diset = `IDX_UNDEF`.

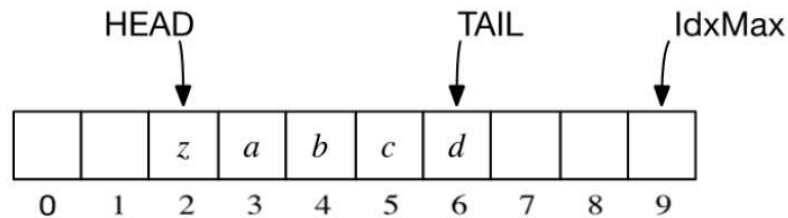


# Implementasi Queue dengan array – alt-2

Ilustrasi Queue tidak kosong, dengan 5 elemen, kemungkinan pertama HEAD sedang berada di indeks 0:



Ilustrasi Queue tidak kosong, dengan 5 elemen, kemungkinan lain HEAD tidak berada di indeks 0 (akibat algoritma penghapusan):



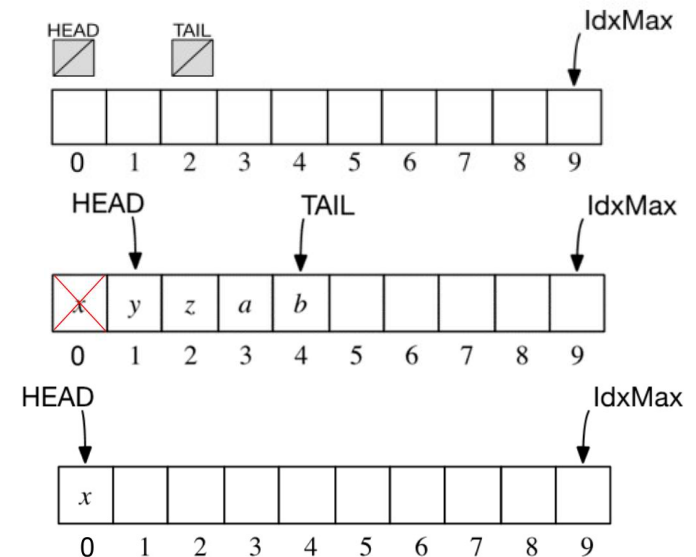
# Implementasi Queue dengan array – alt-2

Algoritma **penambahan elemen** sama dengan alt-1, kecuali pada saat “penuh semu” (lihat slide berikutnya.)

Algoritma **penghapusan elemen**:

- **Jika Queue tidak kosong**: ambil nilai elemen HEAD, kemudian HEAD digeser ke kanan.
- **Kasus khusus (Queue berelemen 1)**: `idxHead` dan `idxTail` diset = `IDX_UNDEF`.

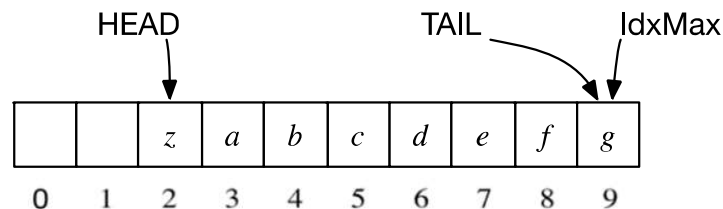
Algoritma ini **TIDAK** mencerminkan pergeseran orang yang sedang mengantri di dunia nyata, tapi **efisien**.





# Implementasi Queue dengan array – alt-2

Keadaan Queue penuh tetapi “semu” sebagai berikut:



Harus dilakukan aksi menggeser elemen untuk menciptakan ruangan kosong.

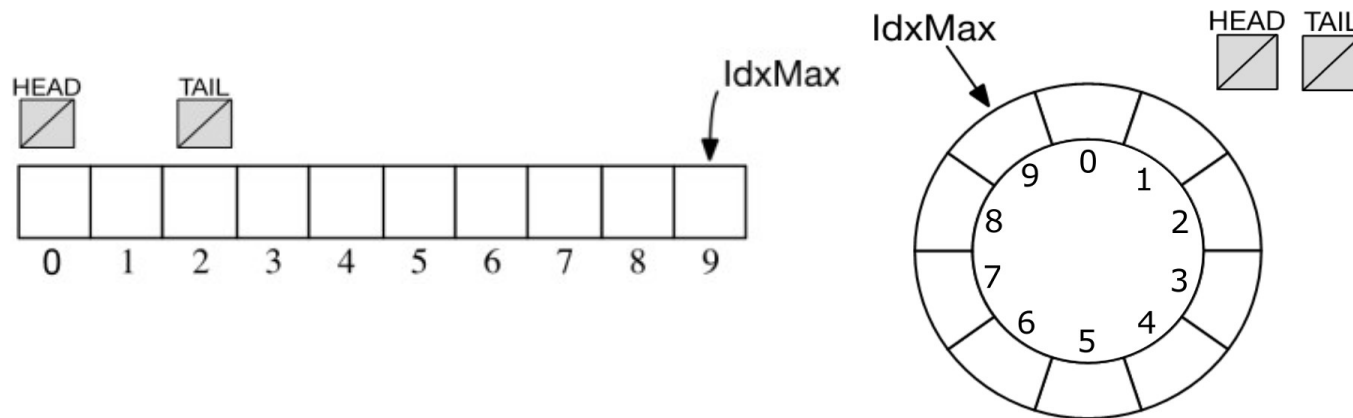
Pergeseran hanya dilakukan jika dan hanya jika  $\text{idxTail} = \text{IdxMax}$ ,  
i.e.,  $\text{idxTail} = \text{CAPACITY} - 1$ .

# Implementasi Queue dengan array – alt-3

Tabel dengan representasi HEAD dan TAIL yang “berputar” mengelilingi indeks tabel dari awal sampai akhir, kemudian kembali ke awal.

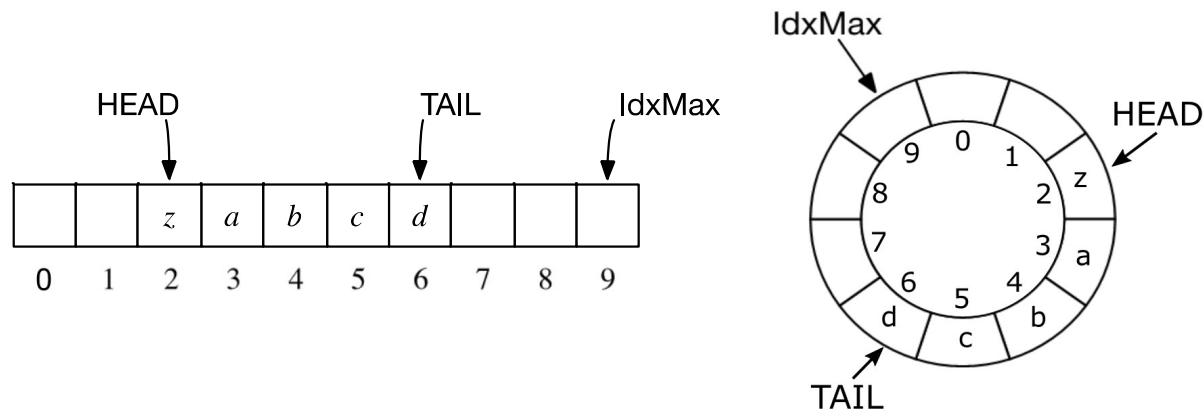
Jika Queue kosong, maka `idxHead` dan `idxTail` = `IDX_UNDEF`.

Representasi ini memungkinkan tidak perlu lagi ada pergeseran yang harus dilakukan seperti pada alt-1 dan alt-2.



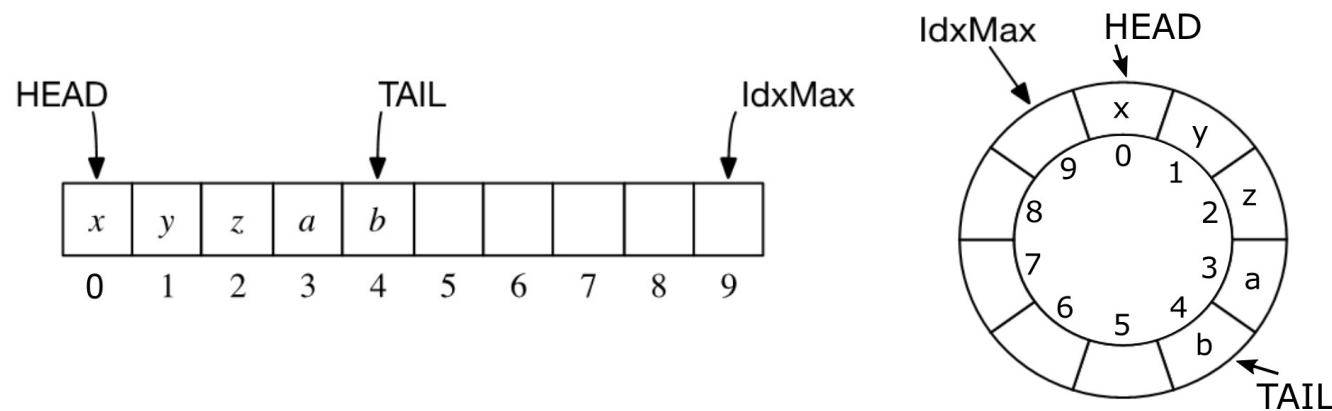
# Implementasi Queue dengan array – alt-3

Ilustrasi Queue tidak kosong, dengan 5 elemen, dengan HEAD tidak berada di indeks 0, tetapi **masih “lebih kecil” atau “sebelum” TAIL** (akibat penghapusan/ penambahan):



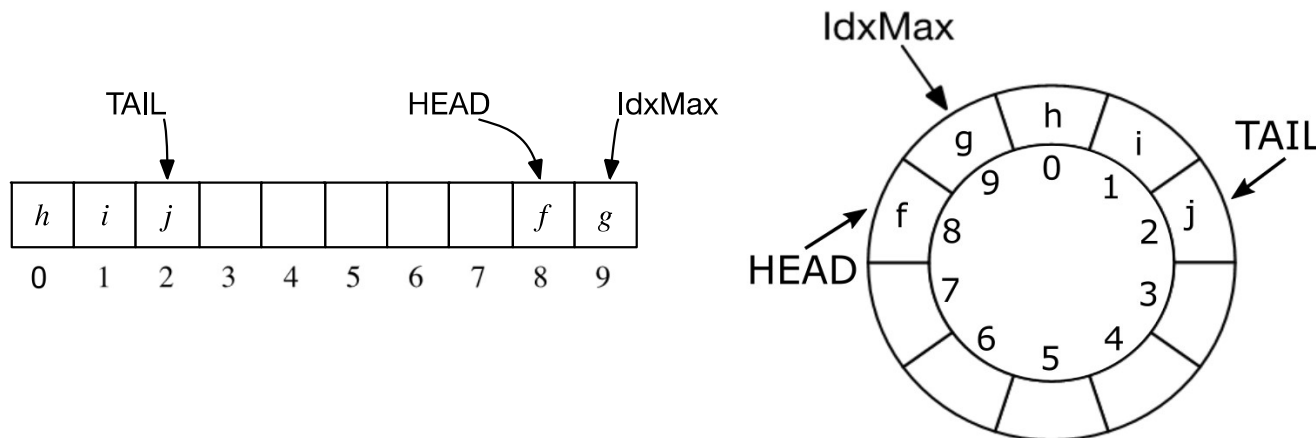
# Implementasi Queue dengan array – alt-3

Ilustrasi Queue tidak kosong, dengan 5 elemen, dengan HEAD sedang berada di indeks 0:



# Implementasi Queue dengan array – alt-3

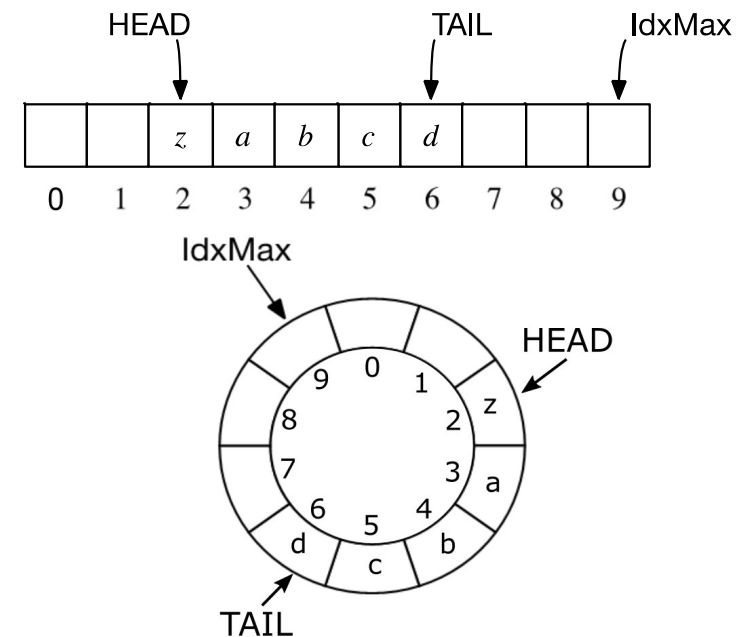
Ilustrasi Queue tidak kosong, dengan 5 elemen, HEAD tidak berada di indeks 0, dan **“lebih besar”** atau **“sesudah”** TAIL (akibat penghapusan/penambahan):



# Implementasi Queue dengan array – alt-3

Algoritma **penambahan elemen**:

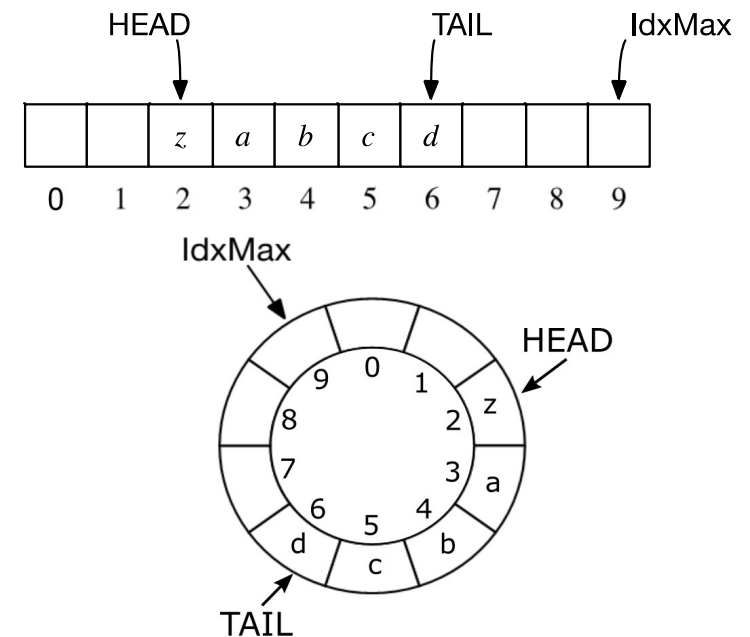
- **Jika masih ada tempat**: geser TAIL
  - **Jika  $\text{idxTail} < \text{IdxMax}$** : algoritma penambahan elemen sama dengan alt-1 dan alt-2.
  - **Jika  $\text{idxTail} = \text{IdxMax}$** : suksesor dari  $\text{IdxMax}$  adalah 0 sehingga  $\text{idxTail}$  yang baru adalah 0.
- **Kasus khusus (Queue kosong)**:  $\text{idxHead}$  dan  $\text{idxTail}$  diset = 0.



# Implementasi Queue dengan array – alt-3

Algoritma **penghapusan elemen**:

- **Jika Queue tidak kosong:**
  - Ambil nilai elemen HEAD, kemudian HEAD digeser ke kanan.
  - **Jika  $\text{idxHead} = \text{IdxMax}$ :**  $\text{idxHead}$  yang baru adalah 0.
- **Kasus khusus (Queue berelemen 1):**  $\text{idxHead}$  dan  $\text{idxTail}$  diset =  $\text{IDX_UNDEF}$ .



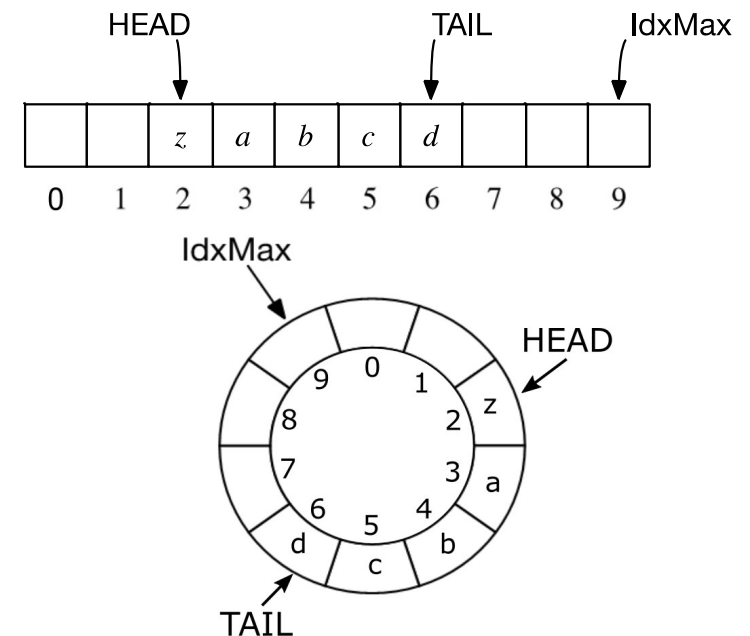
# Implementasi Queue dengan array – alt-3

Algoritma ini **efisien** karena tidak perlu pergeseran.

Seringkali strategi pemakaian tabel semacam ini disebut sebagai **circular buffer**.

Salah satu variasi dari representasi pada alt-3:

Menggantikan representasi TAIL dari “idxTail” menjadi “**count**” (banyaknya elemen Queue).





# Thought exercise

Contoh-contoh sebelumnya menggunakan buffer yang terbatas dan statis. Di sini “isFull” menjadi relevan meskipun tidak ada di bagian “definisi operasi”.

Renungkan apa yang perlu diubah untuk membuat:

- 1) ukuran buffer dapat berbeda untuk setiap queue  
(contoh:  $q1, q2$ : Queue;  $q1$  memiliki kapasitas 100 sedangkan  $q2$  150)
- 2) queue tidak boleh memiliki batas length  
(ukuran buffer bisa  $\infty$  secara teoretis)

Apa konsekuensinya terhadap model alt-1, alt-2, dan alt-3?