Queue (Antrian)

Tim Pengajar IF2110/IF2111
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung


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

Queue

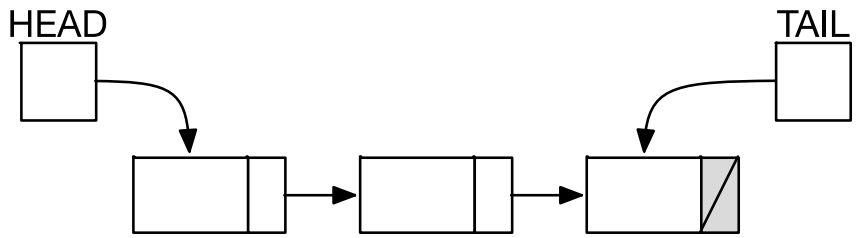
- **QUEUE** adalah list linier yang:
  - dikenali elemen pertama (HEAD) dan elemen terakhirnya (TAIL).
  - aturan penyisipan dan penghapusan elemennya didefinisikan sebagai berikut:
    - Penyisipan selalu dilakukan setelah elemen terakhir,
    - Penghapusan selalu dilakukan pada elemen pertama.
  - satu elemen dengan yang lain dapat diakses melalui informasi NEXT.

Queue

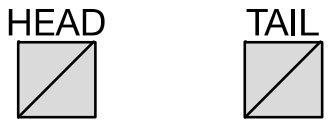
- Elemen Queue tersusun secara FIFO (*First In First Out*)
- Contoh pemakaian queue:
  - antrian job yang harus ditangani oleh sistem operasi (*job scheduling*).
  - antrian dalam dunia nyata.

Queue

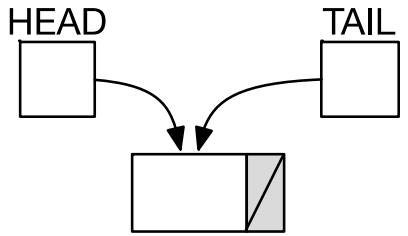
- Secara lojik:
  - Head
  - Tail
  - Elemen
  - Queue kosong



QUEUE dengan tiga elemen



QUEUE kosong



QUEUE dengan satu elemen

Definisi Fungsional

- Jika diberikan Q adalah QUEUE dengan elemen *ElmtQ*

IsEmpty: $Q \rightarrow$ *boolean* { Tes terhadap Q: true jika Q kosong,
false jika Q tidak kosong }

IsFull: $Q \rightarrow$ *boolean* { Tes terhadap Q: true jika memori Q sudah penuh,
false jika memori Q tidak penuh }

NbElmt: $Q \rightarrow$ *integer* { Mengirimkan banyaknya elemen Q }

CreateEmpty: $\rightarrow Q$ { Membuat sebuah antrian kosong }

Enqueue: $ElmtQ \times Q \rightarrow Q$ { Menambahkankan sebuah elemen
setelah elemen ekor QUEUE }

Dequeue: $Q \rightarrow Q \times ElmtQ$ { Menghapus kepala QUEUE,
mungkin Q menjadi kosong }

Implementasi Queue dengan Tabel

- Memori tempat penyimpan elemen adalah sebuah tabel dengan indeks 0..IdxMax.
- IdxMax dapat juga “dipetakan” ke kapasitas Queue (IdxMax+1).
- Representasi field Next: Jika *i* adalah “address” sebuah elemen, maka suksesor *i* adalah Next dari elemen Queue.

ADT Queue

```
Kamus Umum
constant Nil: integer = -1    { Nil = queue dengan elemen kosong }
constant MaxEl: integer = 10 { Banyaknya elemen maksimum }

type infotype: integer    { elemen queue }
type address: integer

{ QUEUE dengan Array statik }
type Queue: ⟨ T: array [0..MaxEl-1] of infotype,
                                { tabel penyimpan elemen queue }
              HEAD: address,  { alamat HEAD: elemen terdepan }
              TAIL: address   { alamat TAIL: elemen terakhir }
            ⟩

{ Deklarasi: Q: Queue
  Definisi akses (diimplementasikan sebagai selektor):
  Tab(Q) adalah Q.T,
  Head(Q) adalah Q.HEAD
  InfoHead(Q) adalah Q.T[Q.HEAD]
  Tail(Q) adalah Q.TAIL
  InfoTail(Q) adalah Q.T[Q.TAIL]}
```

ADT Queue - Konstruktor & Predikat

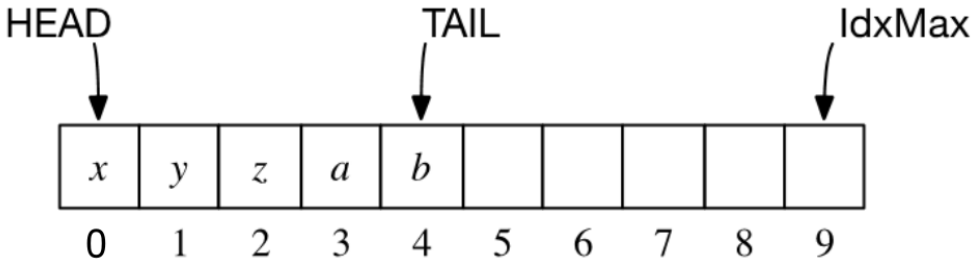
```
{ *** Konstruktor/Kreator *** }
procedure CreateEmpty (output Q: Queue)
{ I.S. Sembarang
  F.S. Membuat sebuah Queue Q yang kosong berkapasitas MaxEl
       jadi indeksnya antara 0..MaxEl-1
       Ciri Queue kosong: HEAD dan TAIL bernilai Nil }

{ *** Predikat Untuk test keadaan Queue *** }
function IsEmpty (Q: Queue) → boolean
{ Mengirim true jika Queue kosong: lihat definisi di atas }

function IsFull(Q: Queue) → boolean
{ Mengirim true jika tabel penampung nilai elemen Queue penuh }
```

ADT Queue - Operasi

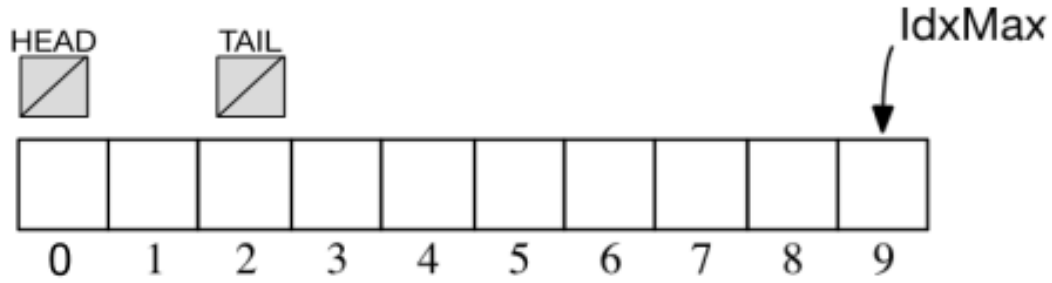
```
{ *** Menambahkan sebuah elemen ke Queue *** }
procedure Enqueue (input/output Q: Queue, input X: infotype)
{ Menambahkan X sebagai elemen Queue Q.
  I.S. Q mungkin kosong, tabel penampung elemen Queue TIDAK penuh
  F.S. Q bertambah elemen X sebagai TAIL yang baru }

{ *** Menghapus sebuah elemen Queue *** }
procedure Dequeue (input/output Q: Queue, output X: infotype)
{ Menghapus X dari Queue Q.
  I.S. Q tidak mungkin kosong
  F.S. X adalah nilai elemen HEAD yang lama.
       Jika Q tidak menjadi kosong,
           HEAD berpindah ke elemen berikutnya pada Q.
       Jika Q menjadi kosong,
           HEAD dan TAIL menjadi bernilai Nil.}
```

Implementasi Queue dengan Tabel - Alternatif I

- Jika Queue tidak kosong: TAIL adalah indeks elemen terakhir, HEAD selalu diset = 0.
- Jika Queue kosong, maka HEAD dan TAIL diset = Nil (-1).
- Ilustrasi Queue tidak kosong dengan 5 elemen:

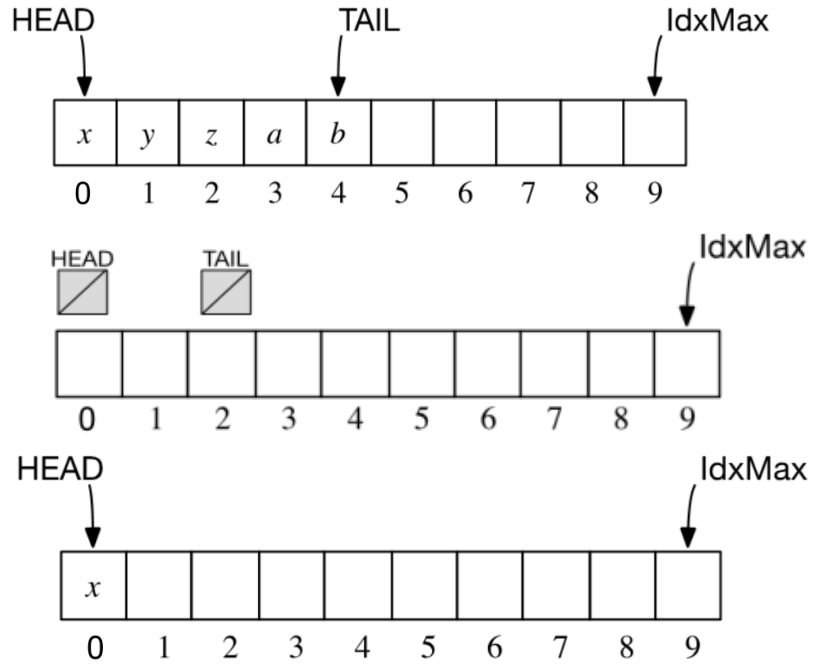


- Ilustrasi Queue kosong:



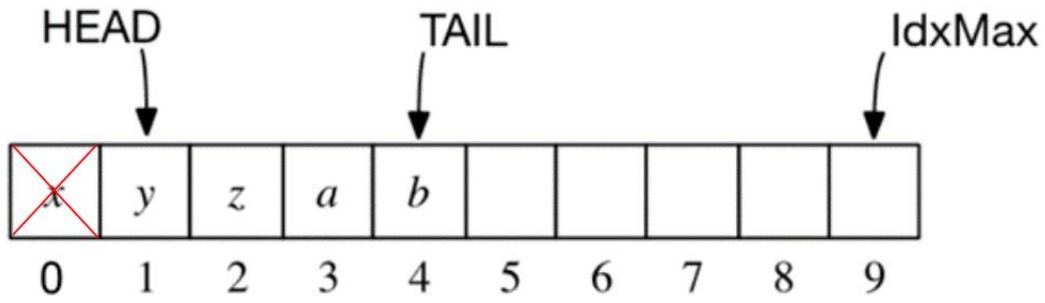
Implementasi Queue dengan Tabel - Alternatif I

- Algoritma **penambahan elemen**:
  - Jika masih ada tempat → geser TAIL ke kanan.
  - Kasus khusus (Queue kosong) → HEAD dan TAIL diset = 0.
- Algoritma paling sederhana dan "naif" untuk **penghapusan elemen**:
  - Jika Queue tidak kosong: ambil nilai elemen HEAD, geser semua elemen mulai dari HEAD+1 s.d. TAIL (jika ada), kemudian geser TAIL ke kiri.
  - Kasus khusus (Queue berelemen 1) → HEAD dan TAIL diset = Nil (-1).
- Algoritma ini mencerminkan pergeseran orang yang sedang mengantri di dunia nyata, tapi tidak efisien.

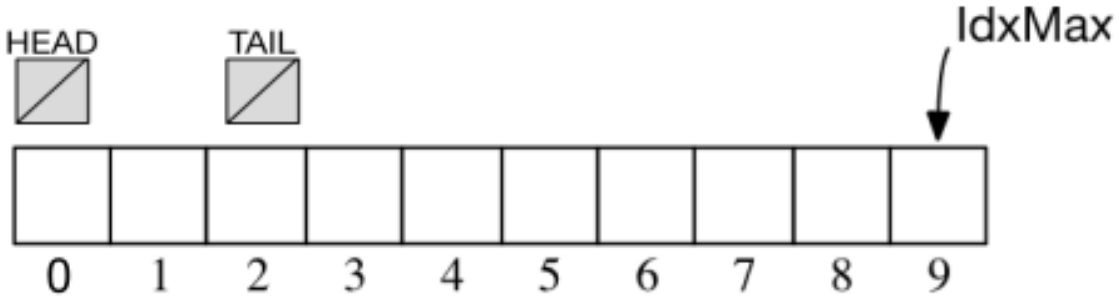


Implementasi Queue dengan Tabel - Alternatif II

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

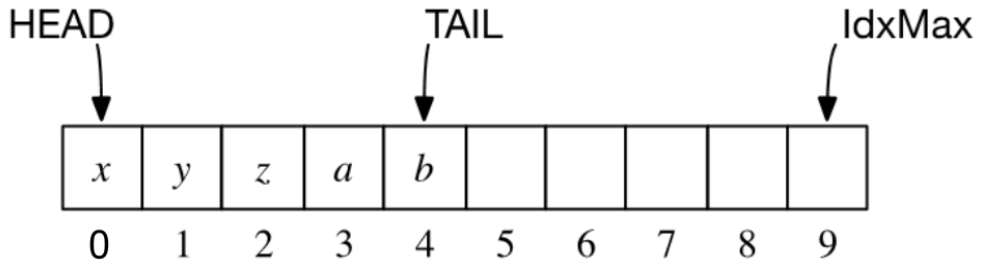


- Jika Queue kosong, maka HEAD dan TAIL diset = Nil (-1).

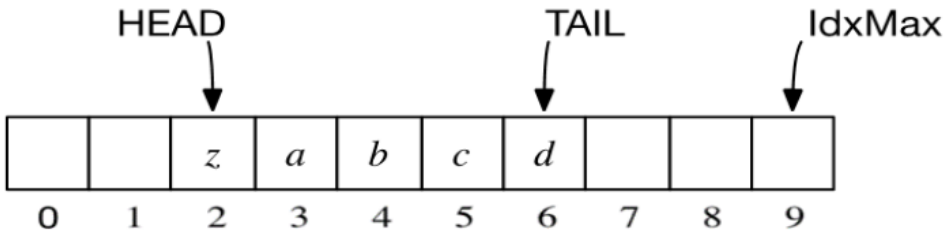


Implementasi Queue dengan Tabel - Alternatif II

- Ilustrasi Queue tidak kosong, dengan 5 elemen, kemungkinan pertama HEAD sedang berada di posisi awal:

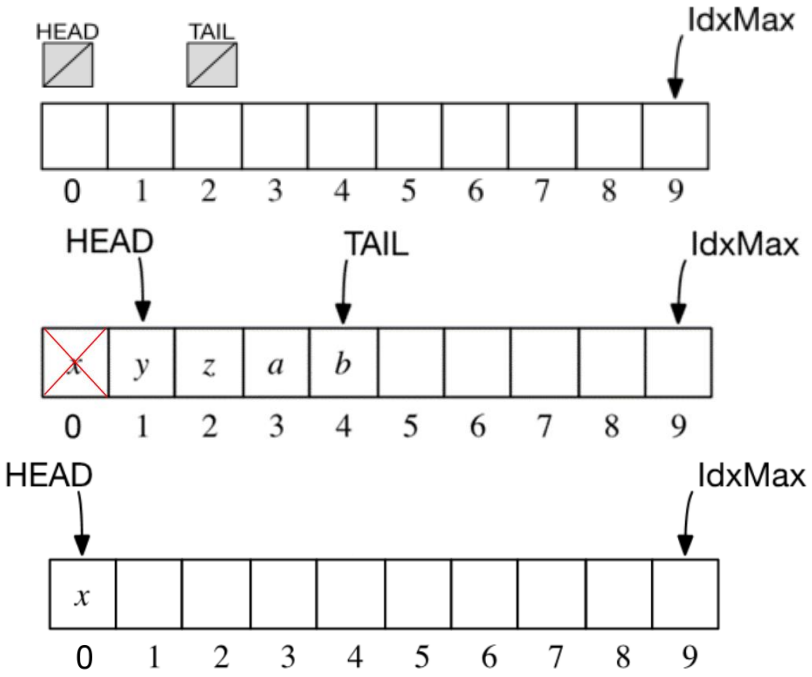


- Ilustrasi Queue tidak kosong, dengan 5 elemen, kemungkinan lain HEAD tidak berada di posisi awal (akibat algoritma penghapusan):



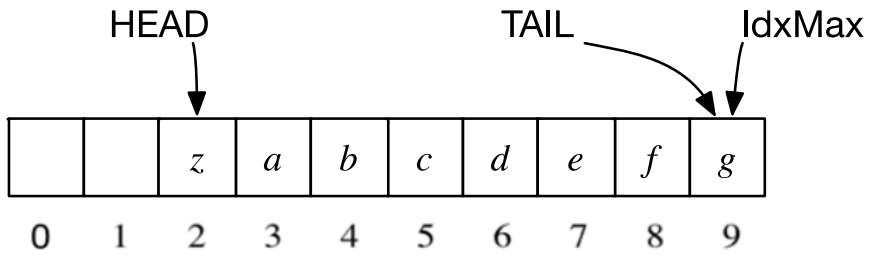
Implementasi Queue dengan Tabel - Alternatif II

- Algoritma **penambahan elemen** sama dengan alternatif I, 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) → HEAD dan TAIL diset = Nil (-1).
- Algoritma ini **TIDAK** mencerminkan pergeseran orang yang sedang mengantri di dunia nyata, tapi **efisien**.



Implementasi Queue dengan Tabel - Alternatif II

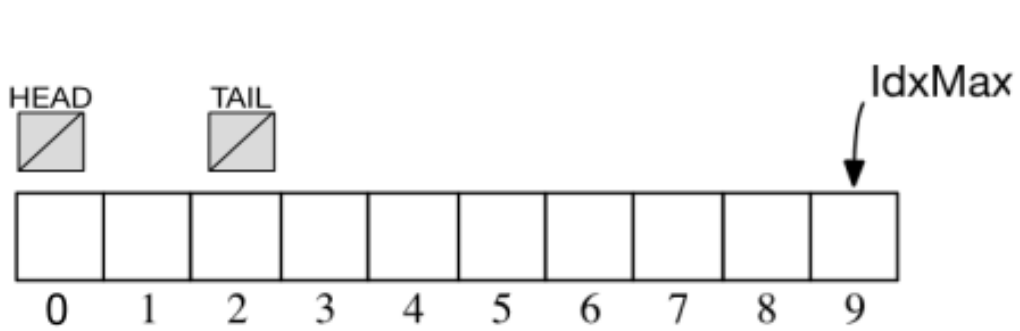
- Keadaan Queue penuh tetapi “semu” sebagai berikut:

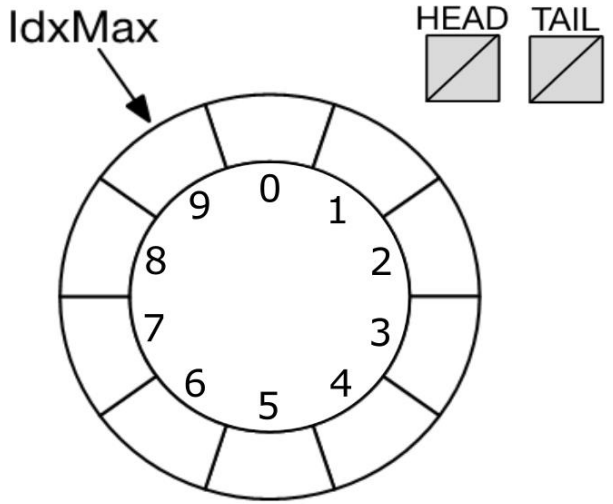


- Harus dilakukan aksi menggeser elemen untuk menciptakan ruangan kosong.
- Pergeseran hanya dilakukan jika dan hanya jika TAIL sudah mencapai IdxMax.

Implementasi Queue dengan Tabel - Alternatif III

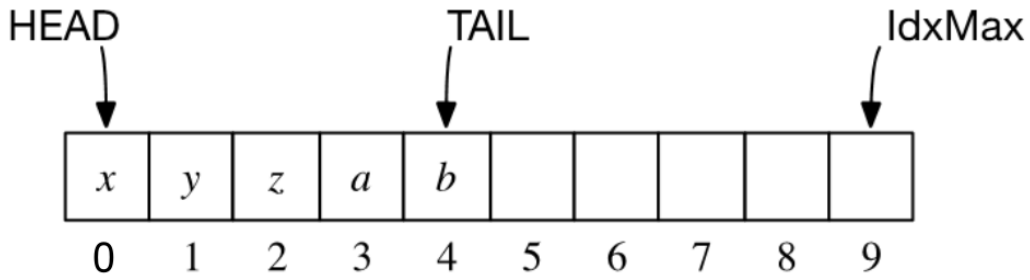
- Tabel dengan representasi HEAD dan TAIL yang “berputar” mengelilingi indeks tabel dari awal sampai akhir, kemudian kembali ke awal.
- Jika Queue kosong, maka HEAD dan TAIL = Nil (-1).
- Representasi ini memungkinkan tidak perlu lagi ada pergeseran yang harus dilakukan seperti pada alternatif I dan II.

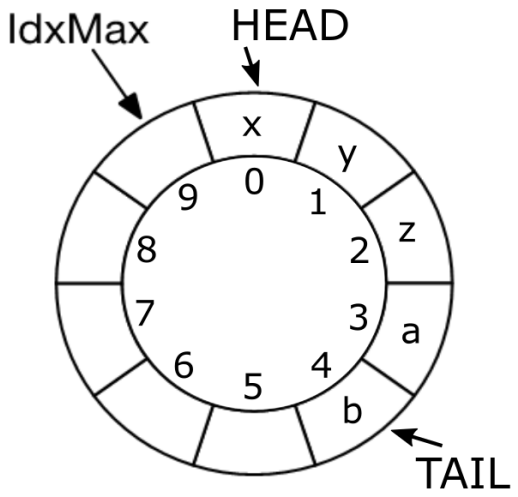




Implementasi Queue dengan Tabel - Alternatif III

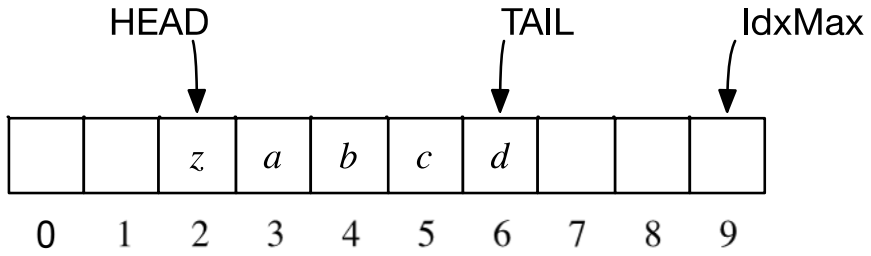
- Ilustrasi Queue tidak kosong, dengan 5 elemen, dengan HEAD “sedang” berada di posisi awal:

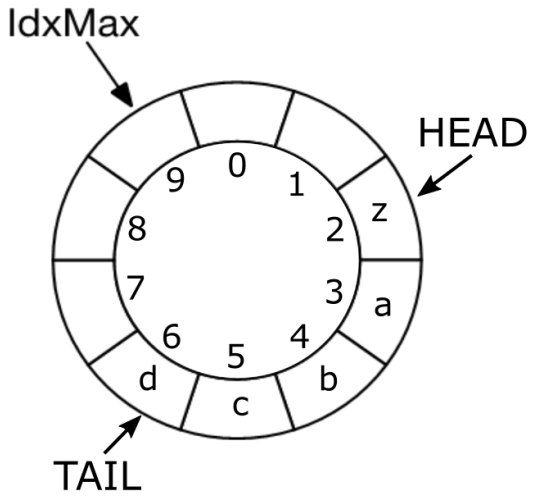




Implementasi Queue dengan Tabel - Alternatif III

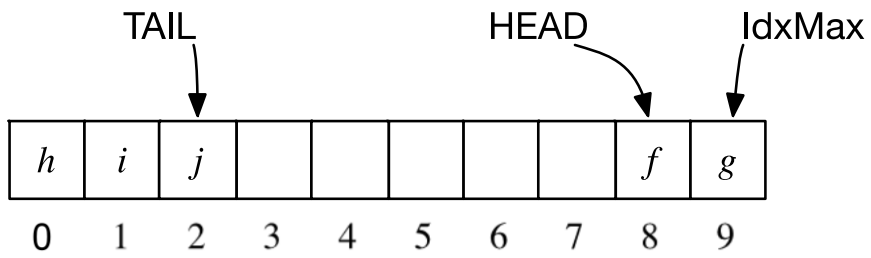
- Ilustrasi Queue tidak kosong, dengan 5 elemen, dengan HEAD tidak berada di posisi awal, tetapi masih “lebih kecil” atau “sebelum” TAIL (akibat penghapusan/ penambahan):

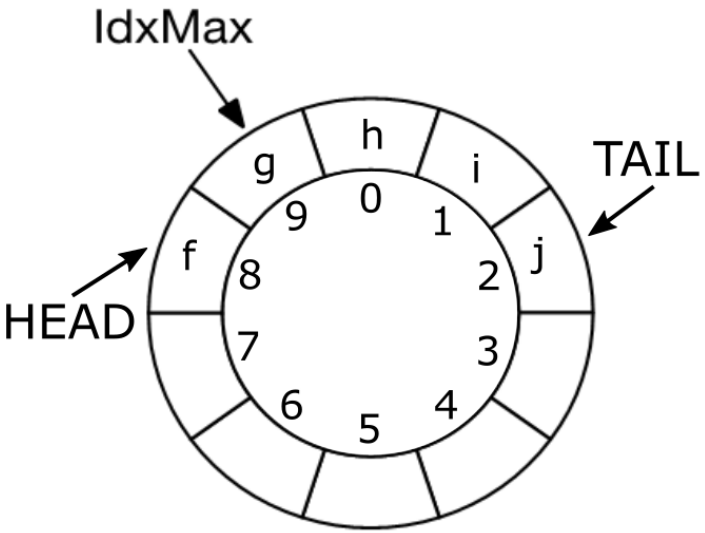




Implementasi Queue dengan Tabel - Alternatif III

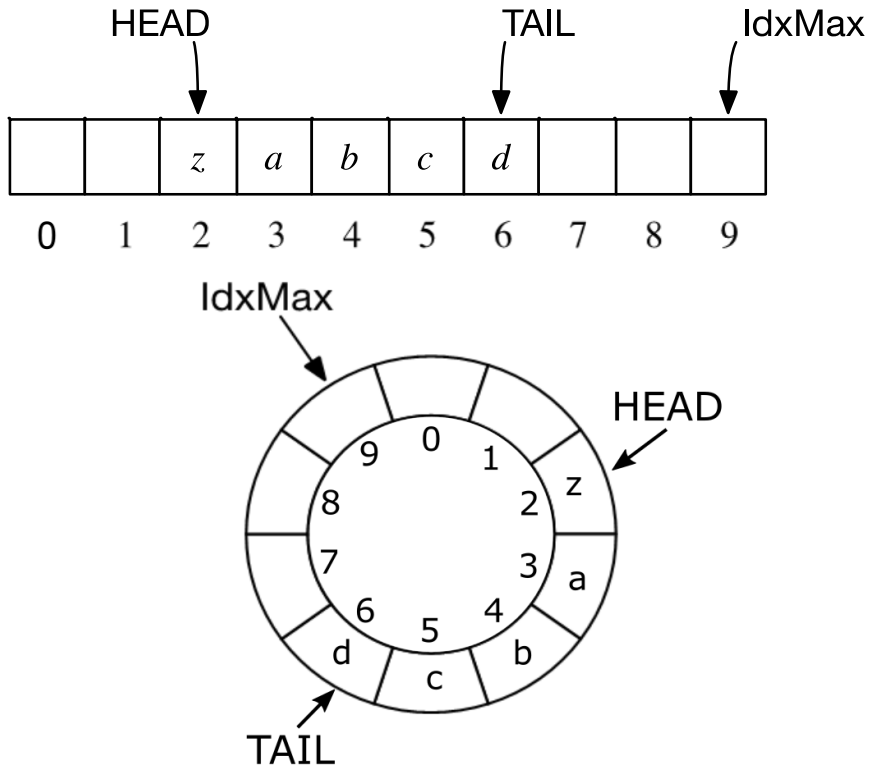
- Ilustrasi Queue tidak kosong, dengan 5 elemen, HEAD tidak berada di posisi awal, tetapi "lebih besar" atau "sesudah" TAIL (akibat penghapusan/penambahan):





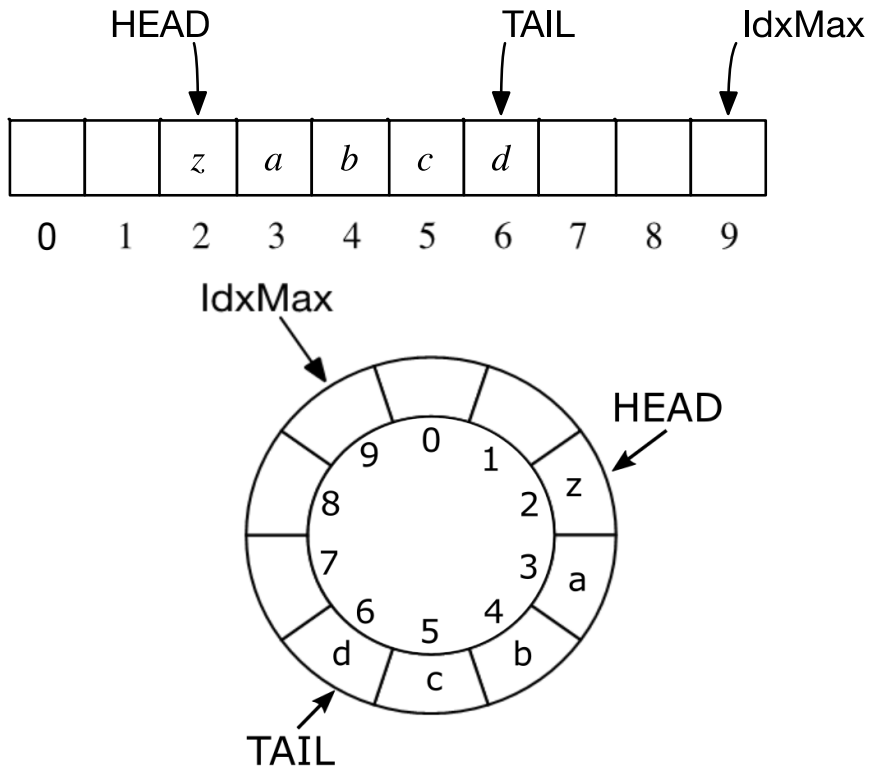
Implementasi Queue dengan Tabel - Alternatif III

- Algoritma penambahan elemen:
  - Jika masih ada tempat, geser TAIL:
    - Jika TAIL belum mencapai IdxMax, maka algoritma penambahan elemen sama dengan alternatif I dan II.
    - Jika TAIL sudah mencapai IdxMax, maka suksesor dari IdxMax adalah 0 sehingga TAIL yang baru adalah 0.
  - Kasus khusus (Queue kosong) → HEAD dan TAIL diset = 0.



Implementasi Queue dengan Tabel - Alternatif III

- Algoritma penghapusan elemen:
  - Jika Queue tidak kosong:
    - Ambil nilai elemen HEAD, kemudian HEAD digeser ke kanan.
    - Jika HEAD sudah mencapai IdxMax, maka suksesor dari HEAD adalah 0.
  - Kasus khusus (Queue berelemen 1) → HEAD dan TAIL diset = Nil (-1).



Implementasi Queue dengan Tabel - Alternatif III

- Algoritma ini **efisien** karena tidak perlu pergeseran.
- Seringkali strategi pemakaian tabel semacam ini disebut sebagai ***circular buffer***, di mana tabel penyimpan elemen dianggap sebagai *buffer*.
- Salah satu variasi dari representasi pada alternatif III:
  - Menggantikan representasi TAIL dengan COUNT (banyaknya elemen Queue).