4 QUEUE (ANTREAN)

4.1. PENGERTIAN *QUEUE* (ANTREAN)

Setelah pada Bab 3 yang lalu kita bahas tentang salah satu jenis daftar (*list*) *linear*, yakni *stack*, kali ini kita bahas jenis lain dari daftar *linear*, yakni *queue* atau antrean. Struktur data antrean atau *queue* adalah suatu bentuk khusus dari *linear list*, dengan operasi penyisipan (*insertion*) hanya diperbolehkan pada salah satu sisi, yang disebut sisi belakang (REAR), dan operasi penghapusan (*deletion*) hanya diperbolehkan pada sisi lainnya, yang disebut sisi depan (FRONT), dari *list*.

Sebagai contoh dapat kita lihat antrean (Q_1, Q_2,...,Q_N). Kita notasikan bagian depan dari antrean Q sebagai FRONT(Q) dan bagian belakang sebagai REAR(Q).

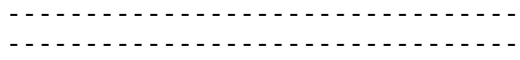
Jadi untuk antrean $Q = [Q_1, Q_2, \ldots, Q_N]$:

$$FRONT(Q) = Q_1 \text{ dan } REAR(Q) = Q_N$$

Kita menggunakan notasi NOEL(Q) untuk menyatakan jumlah elemen di dalam antrean Q. NOEL(Q) mempunyai harga *integer*. Untuk antrean $Q = [Q_1, Q_2, \ldots, Q_N]$, maka NOEL(Q) = N.

Operator penyisipan (*insertion*) disebut INSERT dan operator penghapusan (*deletion*) disebut **REMOVE**.

Sebagai contoh untuk memperjelas bekerjanya antrean, kita perhatikan sederetan operasi berikut ini. Kita mulai dengan antrean hampa Q. Antrean hampa Q, atau Q[ ] dapat disajikan seperti terlihat pada Gambar 4-1



Gambar 4.1. *Antrean hampa*

Di sini :

NOEL(Q) = 0
FRONT(Q) = tidak terdefinisi
REAR(Q) = tidak terdefinisi

Lalu kita INSERT elemen A, diperoleh Q = [A], seperti terlihat di Gambar 4.2.

Gambar 4.2. *Elemen A dimasukkan*

Di sini :

NOEL(Q) = 1
FRONT(Q) = A
REAR(Q) = A

Dilanjutkan dengan INSERT elemen B, sehingga diperoleh Q = [A, B], seperti terlihat di Gambar 4-3.

Gambar 4.3. *Elemen B dimasukkan setelah elemen A*

Di sini :

NOEL(Q) = 2
FRONT(Q) = A
REAR(Q) = B

Dilanjutkan dengan INSERT elemen C, sehingga diperoleh Q = [A, B, C], seperti terlihat di Gambar 4.4.

Gambar 4.4. *Elemen C dimasukkan setelah elemen B*

Di sini :
NOEL(Q) = 3
FRONT(Q) = A
REAR(Q) = C

Dilanjutkan dengan DELETE satu elemen dari Q, sehingga diperoleh Q = [B, C], seperti terlihat di Gambar 4.5.

Gambar 4.5. *Satu elemen dihapus*

Di sini :
NOEL(Q) = 2
FRONT(Q) = B
REAR(Q) = C

Demikian seterusnya, kita dapat melakukan serangkaian INSERT dan DELETE yang lain. Suatu kesalahan *underflow* dapat terjadi, yakni apabila kita melakukan penghapusan pada antrean hampa. Antrean dikatakan beroperasi dalam cara FIRST-IN-FIRST-OUT (FIFO). Disebut demikian karena elemen yang pertama masuk merupakan elemen yang pertama ke luar.

Model antrean, sangat sering ditemukan dalam kejadian sehari-hari, seperti mobil yang menunggu untuk pengisian bahan bakar, mobil pertama dari antrean merupakan mobil pertama yang akan keluar dari antrean. Sebagai contoh lain adalah orang yang menunggu dalam antrean di suatu bank. Orang pertama yang berada di dalam barisan tersebut akan merupakan orang pertama yang akan dilayani.

4.2 OPERASI DASAR PADA ANTREAN

Ada 4 operasi dasar yang dapat dilakukan pada struktur data antrean, yakni :

1. CREATE(antrean)
2. ISEMPTY(antrean)
3. INSERT(elemen,antrean)
4. REMOVE(antrean)

Pandang misalnya antrean Q = [Q_1, Q_2, …, Q_{NOEL}], maka :

CREATE(antrean) :

CREATE(Q) adalah suatu operator untuk membentuk dan menunjukkan suatu antrean hampa Q.

Berarti :

NOEL(CREATE(Q)) = 0
FRONT(CREATE(Q)) = tidak terdefinisi
REAR(CREATE(Q)) = tidak terdefinisi

ISEMPTY(antrean)

ISEMPTY(Q) adalah operator yang menentukan apakah antrean Q hampa atau tidak. *Operand* dari operator ini merupakan antrean, sedangkan hasilnya merupakan tipe data *boolean*.

Di sini :

ISEMPTY(antrean) = *true*, jika Q hampa, yakni jika NOEL(Q)=0
= *false*, dalam hal lain.

Maka, ISEMPTY(CREATE(Q)) = *true.*

INSERT(elemen, antrean)

INSERT(E,Q) adalah operator yang memasukkan elemen E ke dalam antrean Q. Elemen E ditempatkan di bagian belakang dari antrean. Hasil dari operasi ini adalah antrean yang lebih panjang.

REAR(INSERT(E,Q)) = E
Q_{NOEL} adalah E
ISEMPTY(INSERT(E,Q)) = *false*

REMOVE(antrean)

REMOVE(Q) adalah operator yang menghapus elemen bagian depan dari Antrean Q. Hasilnya merupakan antrean yang lebih pendek. Pada setiap operasi ini, harga dari NOEL(Q) berkurang satu, dan elemen kedua dari Q menjadi elemen terdepan.

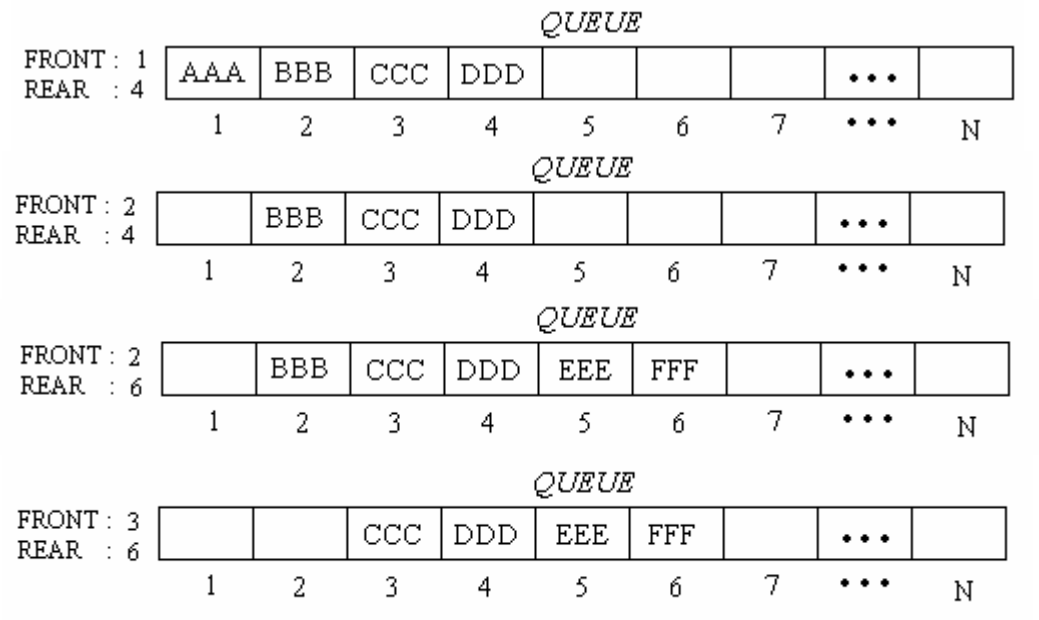
Jika NOEL(Q) = 0, maka REMOVE(Q) memberikan suatu kondisi *error*, yakni suatu *underflow*. Jelas bahwa REMOVE(CREATE(Q)) juga memberikan kondisi *underflow error*.

4.3 PENYAJIAN DARI ANTREAN

Antrean dapat disajikan di dalam komputer dalam berbagai cara. Biasanya dengan menggunakan *one-way-list* (*linear linked list*) ataupun menggunakan *array*. Kalau tidak disebutkan lain, maka antrean kita sajikan dalam *array* QUEUE, dengan dilengkapi dua variabel penunjuk. FRONT, berisi lokasi dari elemen DEPAN antrean dan REAR, berisi lokasi dari elemen BELAKANG antrean. Nilai FRONT = NULL menunjukkan bahwa antrean adalah hampa.

Gambar 4.6 menunjukkan bagaimana menyajikan suatu antrean dalam sebuah *array* QUEUE dengan N elemen. Gambar itu juga menunjukkan bagaimana melakukan pemasukan dan penghapusan elemen antrean.

Pada Gambar 4.6(a) terlihat bahwa antrean mula-mula terdiri atas elemen AAA (sebagai DEPAN), BBB, CCC, dan DDD (sebagai BELAKANG). Gambar 4.6.(b) menunjukkan keadaan setelah penghapusan elemen. Di sini elemen DEPAN yakni AAA dihapus. Gambar 4.6.(c) menggambarkan keadaan setelah penambahan berturut-turut elemen EEE dan FFF. Terakhir sekali, keadaan setelah penghapusan elemen DEPAN, BBB.

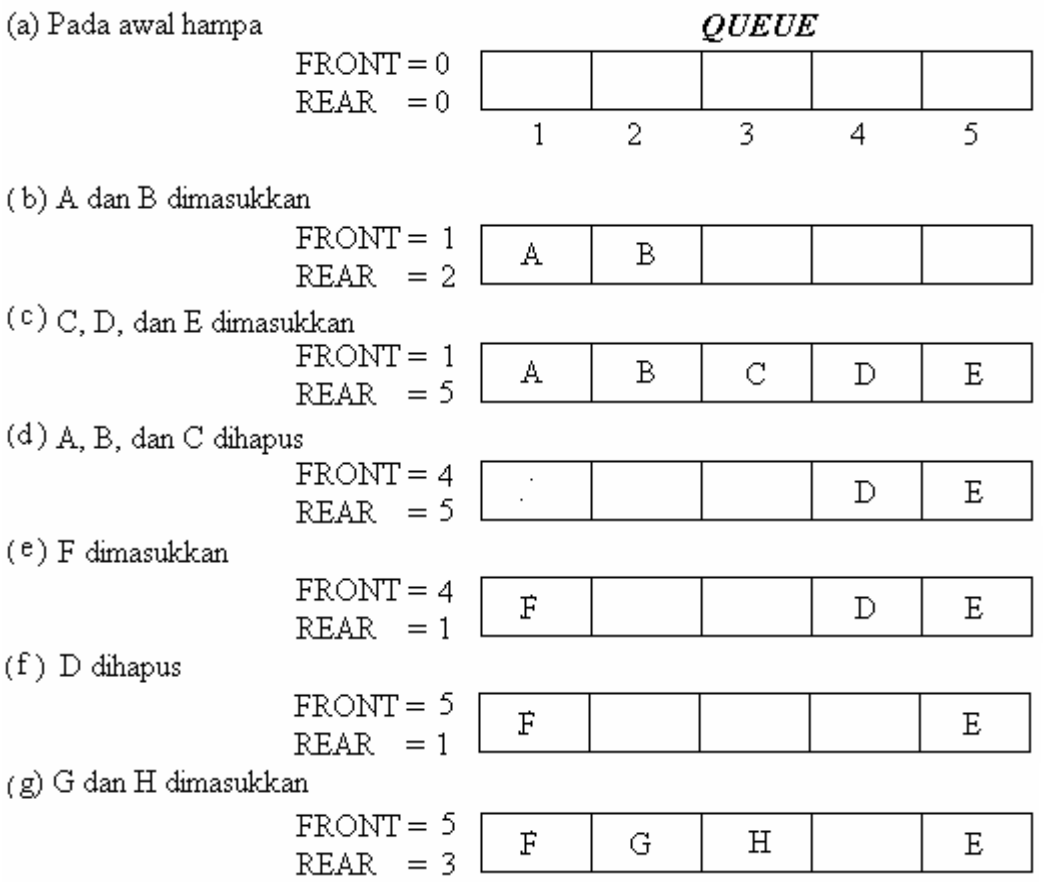


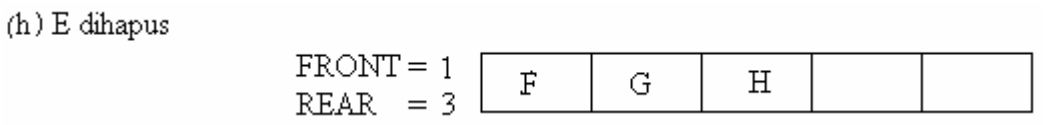
Gambar 4.6. *Cara kerja antrean*

Dapat kita lihat bahwa pada setiap kali penghapusan, nilai lokasi FRONT akan bertambah 1. Untuk setiap kali pemasukan elemen, nilai REAR akan bertambah 1. Hal ini berakibat bahwa setelah pemasukan elemen ke N (berawal dari antrean hampa), maka lokasi QUEUE(N) telah diduduki. Di sini mungkin saja tidak sebanyak N elemen ada dalam antrean (karena sudah dilakukan beberapa penghapusan).

Untuk melakukan pemasukan berikutnya, yakni memasukkan elemen ITEM, kita dapat menggunakan lokasi QUEUE(1). Demikian seterusnya. Dalam hal ini, kita menggunakan *array* sirkular, yakni bahwa QUEUE(1) datang sesudah QUEUE(N) di *array* dalam. Berdasarkan asumsi ini, maka REAR adalah 1.
Secara yang sama, jika FRONT = N dan kita akan melakukan penghapusan, maka sekarang FRONT adalah 1, bukan N+l.

Gambar 4.7 memperlihatkan antrean yang disimpan dalam *array* dengan 5 lokasi memori, sebagai *array* sirkular.





Gambar 4.7. *Circular Array*

Sekarang kita akan menampilkan algoritma QINSERT, yang dimaksudkan untuk memasukkan data ke dalam suatu antrean. Yang mula-mula kita laksanakan dalam algoritma adalah, memeriksa kemungkinan terjadi *overflow error*, yakni dengan melihat apakah antrean tersebut terisi penuh.

Algoritma kedua adalah algoritina QDELETE yang dimaksudkan untuk menghapus elemen DEPAN dari antrean.Yang mula-mula kita laksanakan ialah memeriksa kemungkinan terjadi *underflow error,* yakni dengan melihat apakah antrean tersebut kosong.

Algoritma QINSERT

QINSERT(QUEUE, N, FRONT, DATA)

1. [Apakah antrean penuh]
   Jika FRONT := 1 dan REAR := N, atau jika FRONT := REAR + 1, maka *write* OVERFLOW, return.
2. Jika FRONT := NULL, maka FRONT := 1
   REAR := 1
   dalam hal lain
   jika REAR := N, maka
   REAR := 1
   dalam hal lain
   REAR := REAR + 1
3. QUEUE(REAR) := DATA (masukkan elemen baru)
4. Return

Algoritma QDELETE

QDELETE(QUEUE, N, FRONT, REAR, DATA)

1. [Apakah antrean kosong]
   Jika FRONT := NULL, maka
   Write : UNDERFLOW, return
2. DATA := QUEUE(FRONT)

3. (FRONT mendapat nilai baru). Jika FRONT := REAR, maka (antrean memuat hanya 1 elemen) FRONT := NULL.
   REAR := NULL, dalam hal lain
   Jika FRONT := N, maka FRONT := 1, dalam hal lain :
   FRONT := FRONT + 1
4. Return.

4.4 *DEQUE*

Kali ini akan kita bicarakan beberapa struktur data yang merupakan bentuk variasi dari struktur data antrean atau *queue*, yang telah kita bicarakan terdahulu. Struktur data tersebut adalah *deque* (atau *deck* atau *dequeue*) dan Antrean berprioritas (atau *priority queue*).

DEQUE adalah suatu *linear list* atau daftar *linear*, yang penambahan dan penghapusan elemennya dapat dilakukan pada kedua sisi ujung *list*, tetapi tidak dapat dilakukan di tengah-tengah *list*. Dari sini, kita boleh mengatakan bahwa *deque* adalah suatu *queue* ganda atau *double queue*.

Ada banyak cara penyajian suatu *deque* di dalam komputer. Namun yang biasa digunakan adalah penyajian dengan cara penempatan di dalam sebuah *array* sirkular atau *array* putar DEQUE.

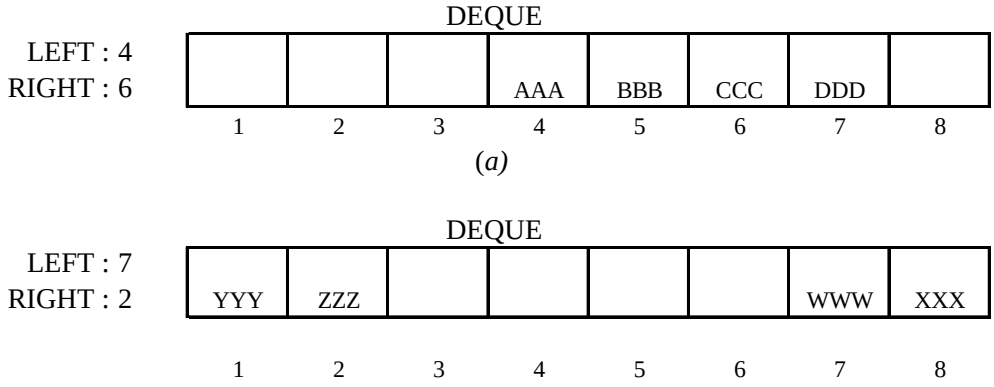
Di sini kita menggunakan dua *pointer* atau penunjuk, LEFT dan RIGHT, yang berturut-turut menunjuk pada sisi kiri dan sisi kanan dari *deque*. Kita senantiasa mengasumsikan bahwa elemen *deque* berurut dari kiri ke kanan. Pengertian sirkular di atas timbul karena elemen DEQUE(l) berada sesudah elemen DEQUE(N) dari *array*.

Gambar 4.8 menggambarkan 2 buah *deque*, masing-masing berisi 4 elemen, yang ditempatkan di dalam sebuah *array* dengan 8 lokasi memori. Kondisi LEFT = NULL dipergunakan untuk menyatakan bahwa suatu *deque* adalah hampa.

Selain *deque* dengan sifat yang telah kita sebutkan di atas, masih ada 2 model variasi *deque*. Kedua variasi tersebut adalah *deque* input terbatas, dan *deque* output terbatas, yang merupakan tengah-tengah antara *deque* dan antrean.

Deque input terbatas adalah suatu *deque* yang membatasi pemasukan elemen hanya pada satu ujung dari *list*, sementara penghapusan elemen boleh dilakukan pada kedua ujung *list*.

Deque output terbatas adalah suatu *deque* yang hanya memperbolehkan penghapusan elemen pada salah satu ujung, tetapi memperbolehkan pemasukan elemen pada kedua ujung *list*.



Gambar 4.8. *Deque*

Sama halnya dengan antrean, komplikasi dapat timbul dalam pemrosesan deque, yakni apabila (a) terjadi *overflow*, yakni pada saat suatu elemen dimasukkan ke dalam *deque* yang sudah berisi penuh, dan (b) terjadi *underflow*, yakni bila suatu elemen harus dihapus dari *deque* yang sudah hampa.

4.5 ANTREAN BERPRIORITAS

Sekarang kita bahas mengenai antrean berprioritas antrean berprioritas adalah himpunan elemen, yang setiap elemennya telah diberikan sebuah prioritas, dan urutan proses penghapusan elemen adalah berdasarkan aturan berikut :

1. Elemen yang prioritasnya lebih tinggi, diproses lebih dahulu dibandingkan dengan elemen yang prioritasnya lebih rendah.
2. Dua elemen dengan prioritas yang sama, diproses sesuai dengan urutan mereka sewaktu dimasukkan ke dalam *priority queue*.

Suatu prototipe dari antrean berprioritas adalah sistem *time sharing*. Di sini pro-gram dengan prioritas yang lebih tinggi diproses terlebih dahulu, dan sejumlah program dengan prioritas yang sama akan membentuk *queue* yang standar.

Ada bermacam-macam cara penyimpanan antrean berprioritas di dalam memori. Kita akan membahas dua di antaranya, yakni yang pertama kita pergunakan cara penyimpanan dalam *one-way list*, dan yang kedua dengan cara penyimpanan dalam *multiple queue*.

Kemudahan dan kesulitan di dalam operasi penambahan ataupun penghapusan elemen, pada antrean berprioritas, akan tergantung pada penyajian mana yang akan dipilih.

4.6 PENYAJIAN *ONE-WAY LIST* DARI ANTREAN BERPRIO-RITAS

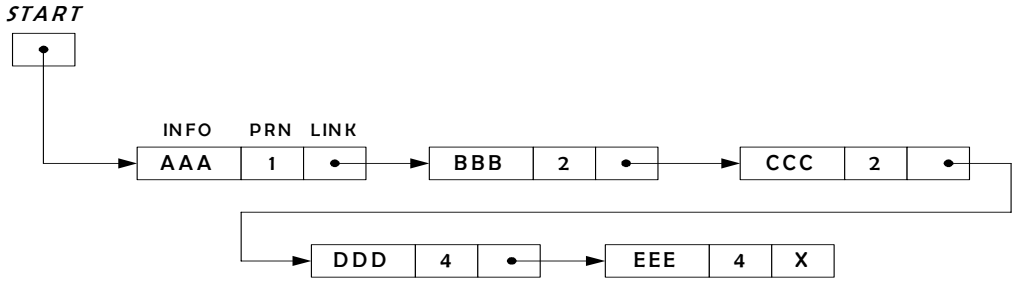
Satu cara untuk menyimpan antrean berprioritas dalam memori adalah dengan menggunakan *one-way list*, dengan ketentuan seperti berikut ini :

1. Setiap simpul dalam *list* akan berisi 3 buah data atau *field*, yakni *field* informasi yang biasa disebut INFO, Nomor prioritas (*priority number*) disebut PRN, dan nomor *link*, disebut LINK.
2. Simpul X mendahului simpul Y di dalam *list* :
   1) bila X mempunyai prioritas yang lebih tinggi dari pada Y
   2) bila keduanya mempunyai prioritas yang sama, tetapi X dimasukkan ke dalam *queue* terlebih dahulu sebelum Y.

Ini berarti bahwa urutan pada *one-way list* adalah berkorespondensi dengan urutan pada antrean berprioritas. *Priority number* akan beroperasi seperti cara yang biasa dipakai, yakni simpul dengan *priority number* terendah, akan mendapat prioritas yang tertinggi.

Sebagai contoh, perhatikan Gambar 4.9 yang memperlihatkan diagram skematik dari antrean berprioritas dengan 7 elemen. Diagram tidak dapat menceritakan kepada kita apakah BBB dimasukkan ke dalam *list* sebelum atau sesudah DDD. Di lain pihak, diagram dapat memperlihatkan kepada kita, bahwa BBB dimasukkan sebelum CCC, karena BBB dan CCC mempunyai *priority number* yang sama dan BBB berada sebelum CCC di dalam *list*. Pada Gambar 4.10 diperlihatkan bagaimana cara antrean berprioritas muncul dalam memori, dengan menggunakan *array* INFO, PRN, dan LINK.

Sifat utama dari penyajian *one-way list* dari sebuah antrean berprioritas adalah bahwa elemen dalam antrean yang seharusnya diproses pertama kali selalu muncul pada bagian permulaan *one-way list*. Oleh karena itu, adalah sangat sederhana untuk menghilangkan dan memproses sebuah elemen antrean prioritas kita tersebut.



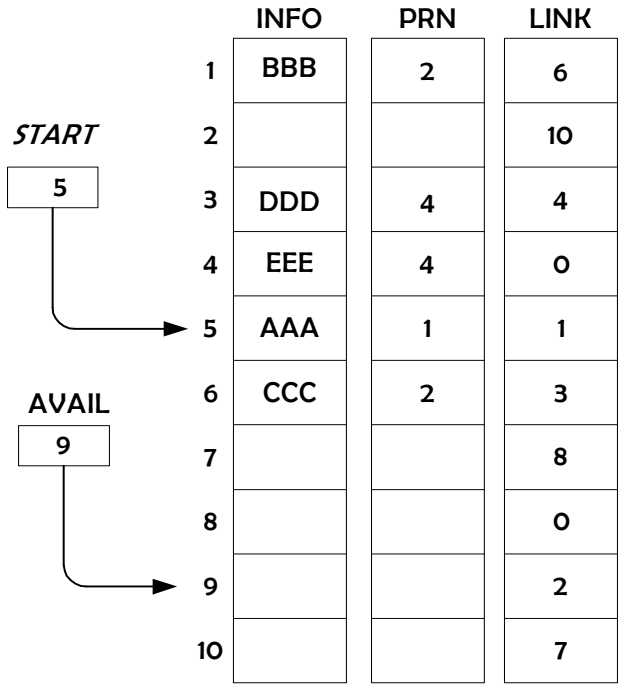
Gambar 4.9. *Antrean berprioritas*

Algoritmanya adalah sebagai berikut :

Algoritma 1

Algoritma ini bekerja untuk menghapus dan memproses elemen pertama dalam sebuah antrean berprioritas yang muncul dalam memori sebagai sebuah *one-way list*.

1. Pasang ITEM := INFO(START). (Langkah ini dimaksudkan untuk menyimpan data dalam simpul pertama).
2. Hapus simpul pertama dari *list*.
3. Proses ITEM
4. Keluar



Gambar 4.10. *Pemetaan antrean berprioritas di memori*

Silakan anda merinci algoritma di atas, lengkap dengan kemungkinan terjadinya *underflow*.

Menambah suatu elemen pada antrean berprioritas kita adalah jauh lebih rumit dibandingkan dengan proses menghapus sebuah elemen dari antrean, karena kita harus menemukan tempat yang benar untuk menyisipkan elemen itu.

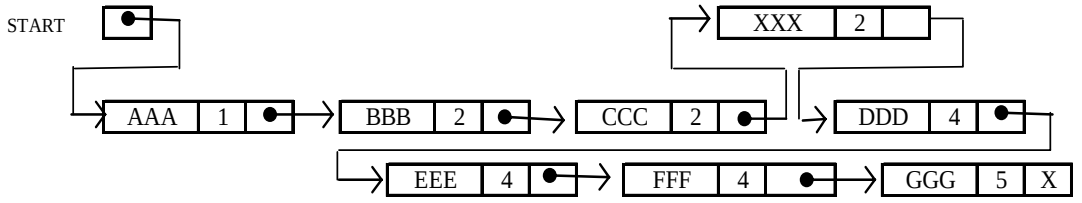
Algoritma 2

Algoritma ini bekerja untuk menambahkan sebuah ITEM dengan nomor prioritas N, pada suatu antrean berprioritas yang disimpan dalam memori sebagai sebuah *one-way list.*

a. Telusuri *one-way list* sampai ditemukan suatu simpul X yang nomor prioritasnya melebihi N. Sisipkan ITEM di depan simpul X

b. Jika tidak ditemukan simpul semacam itu, sisipkan ITEM sebagai elemen terakhir *list*.

Kesulitan utama dalam algoritma muncul dari kenyataan bahwa ITEM disisipkan sebelum simpul X. Hal ini berarti bahwa ketika menelusuri *list* itu, seseorang harus tetap memperhatikan alamat simpul yang mendahului simpul yang sedang diakses.



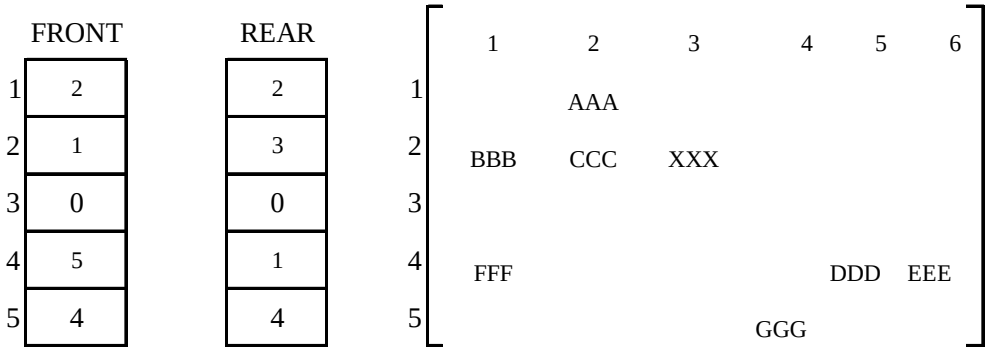
Gambar 4.11. *Penyisipan pada antrean berprioritas*

Sebagai contoh, perhatikan kembali antrean berprioritas pada Gambar 4.9. Misal-kan item XXX dengan nomor prioritas 2, akan dimasukkan ke dalam antrean. Kita telusuri *list*, sambil membandingkan nomor prioritas. Perhatikan bahwa elemen DDD merupakan elemen pertama di dalam *list* yang dijumpai mempunyai nomor prioritas lebih besar dari nomor prioritas XXX Karena itu, XXX dimasukkan ke dalam list di depan DDD, seperti terlihat pada Gambar 4-11.

Dapat dilihat pula bahwa XXX datang sesudah BBB dan CCC, yang mempunyai nomor prioritas sama dengan XXX. Pandang sekarang kita akan menghapus elemen dari *queue*. Dalam hal ini AAA merupakan elemen pertama dari *list* akan terhapus. Pandang bahwa tidak ada lagi penyisipan elemen lain. Elemen berikutnya yang akan dihapus adalah BBB, lalu CCC, kemudian XXX dan seterusnya.

4.7 PENYAJIAN *ARRAY* DARI ANTREAN BERPRIORITAS

Cara lain untuk menyajikan suatu antrean berprioritas dalam memori adalah menggunakan suatu antrean terpisah untuk setiap tingkat prioritas (untuk setiap nomor prioritas). Setiap antrean semacam itu akan muncul dalam *array* sirkularnya sendiri dan harus mempunyai sepasang penunjuk sendiri, FRONT dan REAR. Kenyataannya, jika masing-masing antrean dialokasikan jumlah ruang yang sama, suatu *array* dua dimensi QUEUE dapat digunakan sebagai pengganti *array*. Gambar 4.12 menunjukkan repre-sentasi ini, untuk antrean berprioritas dalam Gambar 4.11, perhatikan bahwa FRONT(K) dan REAR(K) berisi masing-masing elemen depan dan belakang dari baris K *array* QUEUE, baris yang memuat antrean elemen yang bernomor prioritas K.



Gambar 4.12 *Array dua dimensi QUEUE*

Berikut ini garis besar algoritma untuk menghapus dan menyisipkan elemen pada suatu antrean berprioritas yang disimpan dalam memori oleh sebuah *array* dua dimensi QUEUE, seperti tersebut di atas.

Algoritma 3

Algoritma ini bekerja untuk menghapus dan memproses elemen pertama dalam sebuah antrean berprioritas yang disajikan oleh suatu *array* dua dimensi QUEUE.

1. (cari antrean tidak hampa yang pertama). Cari K terkecil, sedemikian sehingga FRONT(K) tidak sama dengan NULL.
2. Hapus dan proses elemen dari baris K QUEUE.
3. Keluar

Algoritma 4

Algoritma ini bekerja untuk menambah sebuah ITEM dengan nomor prioritas M pada suatu antrean berprioritas yang disajikan oleh sebuah *array* dua dimensi QUEUE.

1. Sisipkan ITEM sebagai elemen belakang dari baris M QUEUE.
2. Keluar

Jika kita ambil kesimpulan tentang baik buruknya masing-masing penyajian antrean berprioritas tersebut di atas, maka dapatlah dikatakan bahwa penyajian *array* adalah lebih efisien dari segi waktu dibandingkan dengan penyajian *one-way list*. Namun dari segi ruang yang dibutuhkan, penyajian *one-way list* adalah lebih efisien.

4.8 TAMBAHAN : MESIN ANTREAN

Setelah kita bahas antrean dengan definisi dan teorinya, sekarang akan kita bahas suatu prosedur untuk pelaksanaan beberapa operasi terhadap antrean. Himpunan prosedur tersebut kita namakan "mesin antrean".

Dalam pembahasan mesin antrean, kita menyediakan beberapa perintah dalam bentuk perintah huruf besar tunggal. Input dan output dari nilai yang akan dijalankan harus melalui suatu kode yang berupa perintah huruf besar tunggal. Petunjuk perintah dari mesin antrean adalah sebagai berikut :

I adalah menyisipkan sebuah nilai ke dalam antrean.

D adalah menghapus nilai depan dari sebuah antrean.

F adalah menampilkan nilai depan dari sebuah antrean.

B adalah menentukan maksimum isi antrean.

C adalah mengosongkan antrean.

E adalah keluar.

Deklarasi untuk perintah mesin antrean

```
TYPE TipeNilai = 0.. 99;
     TipeAntrean = ARRAY [1..99] OF
INTEGER;
VAR nilaiMasuk        : TipeNilai;
     Antrean          : TipeAntrean;
     Batasan          : TipeNilai;
     Depan            : TipeNilai;
     Belakang         : TipeNilai;
```

Prosedur pemasukan sesuatu nilai :

```
PROCEDURE    Sekarang    (VAR    Op:CHAR;    VAR
NilaiMasuk:TipeNilai);
    BEGIN
    NilaiMasuk :=0;
    Read(Op);
    WHILE NOT (Op IN ['I','D','F','B','C','E'])
DO
        Read(Op);
    IF (Op :=  'I') THEN Readln(NilaiMasuk);
    IF (Op :='B') THEN Readln(NilaiMasuk)
```

PENYISIPAN SEBUAH NILAI KE DALAM ANTREAN

Maksud dari operasi ini adalah untuk menyisipkan sebuah nilai ke dalam antrean. Sebelum menyisipkan sebuah nilai, maka harus diketahui dahulu batas maksimumnya dengan menggunakan perintah huruf besar tunggal B.

Sebagai contoh dalam suatu antrean akan diisi nilai 1,2,3,4,5 dengan menggunakan perintah huruf besar tunggal I. Sebelum mengisi nilai harus diketahui dahulu batas maksimumnya, misal batas maksimum di sini adalah 5 dengan menggunakan perintah huruf besar tunggal B.

Hasilnya akan terlihat dengan menggunakan perintah huruf tunggal F, maka hasilnya adalah 1,2,3,4,5.

Jika nilai antrean yang disisipkan lebih besar dari batas maksimumnya, maka nilai yang ditentukan tadi tidak akan keluar, di situ akan terlihat bahwa "antrean telah penuh," dengan demikian kita harus mencoba perintah yang lain.

```
PROCEDURE  Penyisipan(NilaiMasuk : TipeNilai);
      PROCEDURE  PerubahanQ(Antrean : TipeAntrean);
           VAR k : INTEGER;
              BEGIN
                   IF (Depan > 0)
                     THEN BEGIN
                         FOR k := 1 TO (Belakang – Depan) DO
                            Antrean[k] := Antrean[Depan+k];
                            Belakang := Belakang;
                            Depan := 0
                          END
              END (PerubahanQ);
BEGIN
   IF (Belakang > Batasan)
      THEN BEGIN Perubahan(Antrean)
      END;
   IF (Belakang >= Batasan)
      THEN
         BEGIN
         Writeln('Antrean penuh, coba perintah yang lain.');
         Writeln(‘Tekan B untuk batas maksimum isi
                  Antrean(0 ... 99)')
         END
      ELSE
        BEGIN
         Belakang := Belakang + 1;
         Antrean[Belakang] := NilaiMasuk
        END
END [Penyisipan];
```

MENGHAPUS NILAI DEPAN DARI SEBUAH ANTREAN

Selain menyisipkan sebuah nilai ke dalam antrean, antrean juga perlu dihapus. Untuk menghapus sebuah nilai antrean yakni dengan menggunakan perintah huruf besar tunggal D.

Pada operasi ini akan ditentukan jika kita ingin menghapus sebuah nilai yang telah disisipkan. Dalam hal ini yang akan dihapus adalah nilai depan dari suatu antrean.

Sebagai contoh dalam suatu antrean yang mempunyai nilai 1,2,3,4,5 dengan menggunakan perintah huruf besar tunggal I, maka nilai yang terhapus adalah nilai I dengan menggunakan perintah huruf besar tunggal D. Akan terlihat hasilnya dengan menggunakan perintah huruf besar tunggal F menjadi 2,3,4,5.

Perlu kita diketahui, jika dalam suatu antrean yang akan dihapus tidak ada nilainya, maka akan terlihat hasilnya "Antrean kosong", seperti terlihat pada *procedure* di bawah ini.

```
PROCEDURE  Penghapusan(Antrean : TipeAntrean);
    BEGIN
     IF Kosong(Antrean)
        THEN Writeln('Antrean kosong, coba perintah lain')
     ELSE  BEGIN
        Depan := Depan + 1;
        IF (Depan := Belakang)
     THEN BEGIN
        Depan := 0;
        Belakang :=0
      END
    END [Penghapusan];
```

MENAMPILKAN NILAI DEPAN DARI SEBUAH ANTREAN

Operasi ini merupakan suatu operasi yang khusus menampilkan sesuatu nilai. Pada dasarnya operasi ini hanya merupakan sebagai tampilan, khususnya menampilkan nilai depan dari sebuah antrean. Nilai-nilai yang telah dihasilkan merupakan nilai-nilai seperti terlihat pada contoh di atas yakni pada penggunaan perintah-perintah huruf besar tunggal I, D, B dan C. Perintah huruf besar tunggal C merupakan perintah yang fungsinya hanya untuk mengosongkan antrean.

Jika antrean yang dijalankan tidak menghasilkan suatu nilai, maka hasilnya akan terlihat menjadi "Antrean kosong"

```
PROCEDURE Tampilan(DepanTipeNilai);
     VAR k TipeNilai;
     BEGIN
          Writeln;
          IF Kosong(Antrean)
     THEN Writeln('Antrean kosong.')
     ELSE Writeln('Nilai depan adalah', Antrean[Depan+l]:3);
     Writeln
END [Tampilan];
```

L A T I H A N 4

1. Apa beda cara kerja *queue* dengan *stack* ?

2. Apa saja operasi yang dapat dilakukan terhadap *queue* ?

3. Operator penghapusan elemen dalam *queue* disebut dengan (a) DELETE, (b) REMOVE, atau (c) ZAP

4. Apa saja yang menyebabkan suatu antrean dalam keadaan hampa ?

5. Apa yang menyebabkan terjadinya kesalahan *overflow* atau *underflow* pada *queue?*

6. Operator apa dari *queue* yang hasilnya berupa data bertipe *boolean* ?

7. Elemen mana yang akan dikeluarkan dari *queue* oleh perintah REMOVE ?

8. Apa yang dimaksud dengan '*deque*' ?

9. Apa yang dimaksud dengan antrean berprioritas ?

10. Buatlah sebuah program dengan bahasa pemrograman apapun untuk membuat 'mesin antrean'.