Bab Queue

Convert a Queue to a List

**import** java.util.ArrayList;

**import** java.util.LinkedList;

**import** java.util.List;

**import** java.util.Queue;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Queue<String> myQueue = **new** LinkedList<String>();

myQueue.add("A");

myQueue.add("B");

myQueue.add("C");

myQueue.add("D");

List<String> myList = **new** ArrayList<String>(myQueue);

**for** (Object theFruit : myList)

System.out.println(theFruit);

}

}

Use the Stack class in Java

**import** java.util.Stack;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Stack<**Integer**> stack = **new** Stack<**Integer**>();

**for** (**int** i = 0; i < 10; i++) {

stack.push(i);

System.out.print(i + " ");

}

System.out.println("");

**int** position = stack.search(3);

System.out.println("Search result position: " + position);

System.out.println("Stack top: " + stack.peek());

**while** (!stack.empty()) {

System.out.print(stack.pop() + " ");

}

}

}

Create a queue using LinkedList class

**import** java.util.LinkedList;

**import** java.util.Queue;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Queue<String> queue = **new** LinkedList<String>();

queue.offer("First");

queue.offer("Second");

queue.offer("Third");

queue.offer("Fourth");

System.out.println("Size: " + queue.size());

System.out.println("Queue head using peek : " + queue.peek());

System.out.println("Queue head using element: " + queue.element());

Object data;

**while** ((data = queue.poll()) != null) {

System.out.println(data);

}

}

}

Checking what item is first in line without removing it

**import** java.util.LinkedList;

**import** java.util.Queue;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Queue<String> queue = **new** LinkedList<String>();

queue.add("A");

queue.add("B");

queue.offer("C");

queue.offer("D");

System.out.println("remove: " + queue.remove());

System.out.println("element: " + queue.element());

System.out.println("poll: " + queue.poll());

System.out.println("peek: " + queue.peek());

}

}

Checking what item is first in line without removing it

**import** java.util.LinkedList;

**import** java.util.Queue;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Queue<String> queue = **new** LinkedList<String>();

queue.add("A");

queue.add("B");

queue.offer("C");

queue.offer("D");

System.out.println("remove: " + queue.remove());

System.out.println("element: " + queue.element());

System.out.println("poll: " + queue.poll());

System.out.println("peek: " + queue.peek());

}

}

Queue (Antrian)

Definisi:

1. Queue adalah daftar terurut yang dapat dilakukan proses penyisipan di salah satu ujung di sebut REAR, dan proses penghapusan du ujung lainnya di sebut FRONT.
2. Queue juga di sebut First in First Out



Aplikasi queue

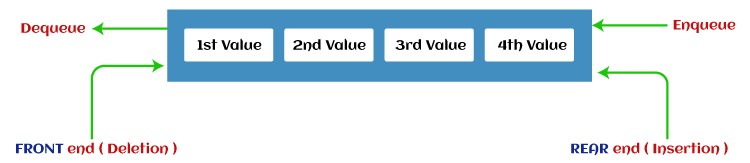
* Antrian banyak digunakan sebagai daftar tunggu untuk satu sumber daya ystem seperti printer, disk, CPU.
* Antrian digunakan dalam transfer data asinkron (di mana data tidak ditransfer pada kecepatan yang sama antara dua proses) misalnya. Pipa, file IO, soket.
* Antrian digunakan sebagai buffer di ystemn besar aplikasi seperti pemutar media MP3, pemutar CD, dll.
* Antrian digunakan untuk mempertahankan daftar putar di pemutar media untuk menambah dan menghapus lagu dari daftar putar.
* Antrian digunakan dalam ystem operasi untuk menangani interupsi.

Apa itu Antrian?

Antrian adalah struktur data yang mirip dengan antrian di dunia nyata. Antrian adalah struktur data di mana apa yang lebih dulu akan keluar lebih dulu, dan mengikuti kebijakan FIFO (First-In-First-Out). Antrian juga dapat didefinisikan sebagai daftar atau kumpulan di mana penyisipan dilakukan dari satu ujung yang dikenal sebagai ujung **belakang** atau **ekor** antrian, sedangkan penghapusan dilakukan dari ujung lain yang dikenal sebagai ujung **depan** atau **kepala** antrian. .

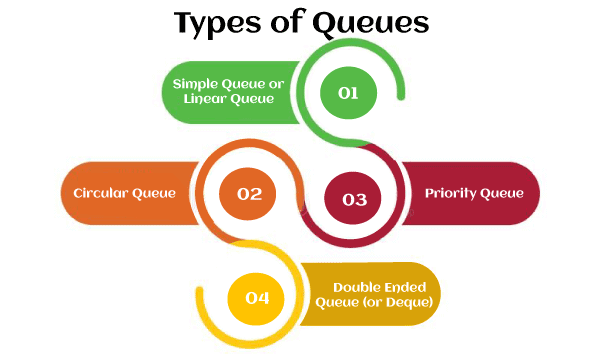
Contoh antrean di dunia nyata adalah antrean tiket di luar gedung bioskop, di mana orang yang masuk terlebih dahulu dalam antrean mendapatkan tiket terlebih dahulu, dan orang terakhir yang masuk dalam antrean akhirnya mendapatkan tiket. Pendekatan serupa diikuti dalam antrian dalam struktur data.

Representasi antrian ditunjukkan pada gambar di bawah ini -



Jenis Antrian

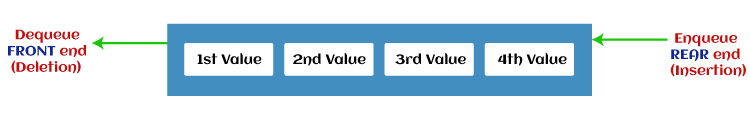
Ada empat jenis antrian yang terdaftar sebagai berikut -



* Antrian Sederhana atau Antrian Linear
* Antrian Melingkar
* Antrian Prioritas
* Antrian Berakhir Ganda (atau Deque)

Antrian Sederhana atau Antrian Linear

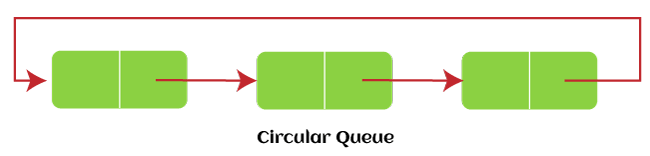
Dalam Linear Queue, penyisipan terjadi dari satu ujung sementara penghapusan terjadi dari ujung yang lain. Ujung di mana penyisipan terjadi dikenal sebagai ujung belakang, dan ujung di mana penghapusan terjadi dikenal sebagai ujung depan. Ini secara ketat mengikuti aturan FIFO.



Kelemahan utama menggunakan Antrian linier adalah bahwa penyisipan dilakukan hanya dari ujung belakang. Jika tiga elemen pertama dihapus dari Antrian, kita tidak dapat menyisipkan lebih banyak elemen meskipun ruang tersedia dalam Antrian Linier. Dalam hal ini, Antrian linier menunjukkan kondisi overflow karena bagian belakang menunjuk ke elemen Antrian terakhir.

Antrian Melingkar

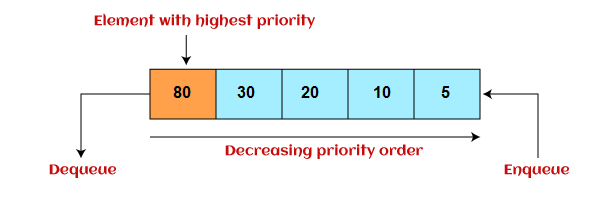
Dalam Circular Queue, semua node direpresentasikan sebagai lingkaran. Ini mirip dengan Antrian linier kecuali bahwa elemen terakhir dari antrian terhubung ke elemen pertama. Ini juga dikenal sebagai Ring Buffer, karena semua ujungnya terhubung ke ujung yang lain. Representasi antrian melingkar ditunjukkan pada gambar di bawah ini -



Kelemahan yang terjadi pada antrian linier diatasi dengan menggunakan antrian melingkar. Jika ruang kosong tersedia dalam antrian melingkar, elemen baru dapat ditambahkan di ruang kosong hanya dengan menambah nilai belakang. Keuntungan utama menggunakan antrian melingkar adalah pemanfaatan memori yang lebih baik.

Antrian Prioritas

Ini adalah jenis antrian khusus di mana elemen-elemennya disusun berdasarkan prioritas. Ini adalah tipe khusus dari struktur data antrian di mana setiap elemen memiliki prioritas yang terkait dengannya. Misalkan beberapa elemen terjadi dengan prioritas yang sama, mereka akan diatur menurut prinsip FIFO. Representasi antrian prioritas ditunjukkan pada gambar di bawah ini -



Penyisipan dalam antrian prioritas terjadi berdasarkan kedatangan, sedangkan penghapusan dalam antrian prioritas terjadi berdasarkan prioritas. Antrian prioritas terutama digunakan untuk mengimplementasikan algoritma penjadwalan CPU.

Ada dua jenis antrian prioritas yang dibahas sebagai berikut -

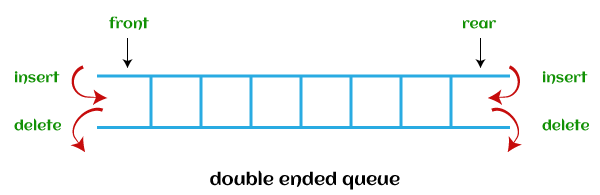
* **Antrian prioritas naik -** Dalam antrian prioritas naik, elemen dapat dimasukkan dalam urutan arbitrer, tetapi hanya yang terkecil yang dapat dihapus terlebih dahulu. Misalkan sebuah array dengan elemen 7, 5, dan 3 dalam urutan yang sama, maka penyisipan dapat dilakukan dengan urutan yang sama, tetapi urutan penghapusan elemen adalah 3, 5, 7.
* **Antrian prioritas menurun -** Dalam antrian prioritas menurun, elemen dapat dimasukkan dalam urutan arbitrer, tetapi hanya elemen terbesar yang dapat dihapus terlebih dahulu. Misalkan sebuah array dengan elemen 7, 3, dan 5 dalam urutan yang sama, maka penyisipan dapat dilakukan dengan urutan yang sama, tetapi urutan penghapusan elemen adalah 7, 5, 3.

Deque (atau, Antrian Berakhir Ganda)

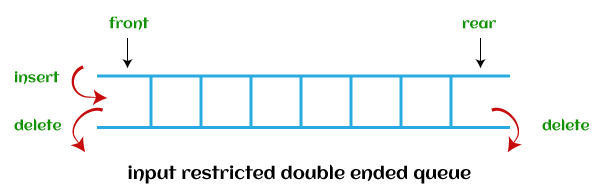
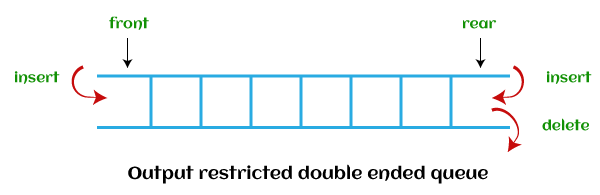
Pada Deque atau Double Ended Queue, penyisipan dan penghapusan dapat dilakukan dari kedua ujung antrian baik dari depan maupun belakang. Ini berarti bahwa kita dapat menyisipkan dan menghapus elemen dari ujung depan dan belakang antrian. Deque dapat digunakan sebagai pemeriksa palindrom artinya jika kita membaca string dari kedua ujungnya, maka string tersebut akan sama.

Deque dapat digunakan baik sebagai tumpukan dan antrian karena memungkinkan operasi penyisipan dan penghapusan di kedua ujungnya. Deque dapat dianggap sebagai stack karena stack mengikuti prinsip LIFO (Last In First Out) dimana penyisipan dan penghapusan keduanya hanya dapat dilakukan dari satu ujung. Dan dalam deque, dimungkinkan untuk melakukan penyisipan dan penghapusan dari satu ujung, dan Deque tidak mengikuti prinsip FIFO.

Representasi deque ditunjukkan pada gambar di bawah ini -



Ada dua jenis deque yang dibahas sebagai berikut -

* **Input dibatasi deque -** Seperti namanya, dalam antrian input terbatas, operasi penyisipan dapat dilakukan hanya pada satu ujung, sedangkan penghapusan dapat dilakukan dari kedua ujungnya.  
  
* **Output dibatasi deque -** Seperti namanya, dalam antrian output terbatas, operasi penghapusan dapat dilakukan hanya pada satu ujung, sedangkan penyisipan dapat dilakukan dari kedua ujungnya.  
  

Sekarang, mari kita lihat operasi yang dilakukan pada antrian.

Operasi dilakukan pada antrian

Operasi dasar yang dapat dilakukan pada antrian terdaftar sebagai berikut -

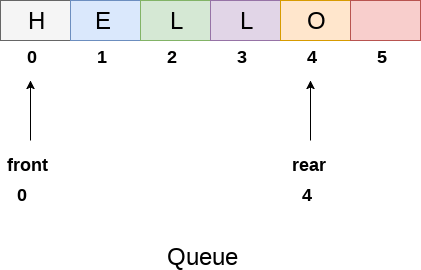
* **Enqueue:** Operasi Enqueue digunakan untuk menyisipkan elemen di bagian belakang antrian. Ini kembali batal.
* **Dequeue:** Ini melakukan penghapusan dari front-end antrian. Itu juga mengembalikan elemen yang telah dihapus dari front-end. Ini mengembalikan nilai integer.
* **Peek:** Ini adalah operasi ketiga yang mengembalikan elemen, yang ditunjuk oleh pointer depan dalam antrian tetapi tidak menghapusnya.
* **Queue overflow (isfull):** Ini menunjukkan kondisi overflow ketika antrian benar-benar penuh.
* **Queue underflow (isempty):** Ini menunjukkan kondisi underflow saat Queue kosong, yaitu, tidak ada elemen di Queue.

Cara menerapkan antrian

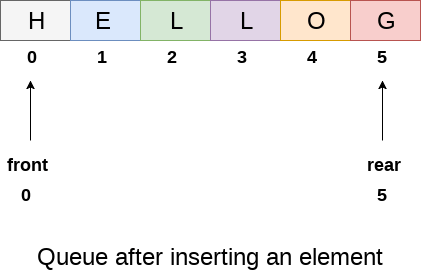
Ada dua cara untuk mengimplementasikan Antrian:

* **Implementasi menggunakan array:** Alokasi berurutan dalam Antrian dapat diimplementasikan menggunakan array.

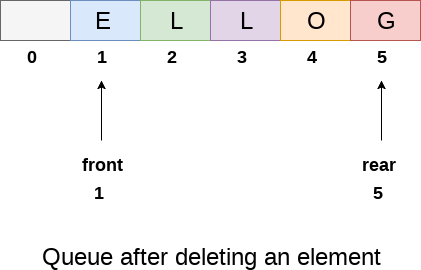
Kita dapat dengan mudah merepresentasikan antrian dengan menggunakan array linier. Ada dua variabel yaitu depan dan belakang, yang diterapkan dalam kasus setiap antrian. Variabel depan dan belakang menunjuk ke posisi dari mana penyisipan dan penghapusan dilakukan dalam antrian. Awalnya, nilai depan dan antrian adalah -1 yang mewakili antrian kosong. Representasi array dari antrian yang berisi 5 elemen beserta nilai masing-masing depan dan belakang, ditunjukkan pada gambar berikut.



Gambar di atas menunjukkan antrian karakter yang membentuk kata bahasa Inggris **"HELLO"** . Karena, Tidak ada penghapusan yang dilakukan dalam antrian sampai sekarang, oleh karena itu nilai front tetap -1 . Namun, nilai belakang meningkat satu setiap kali penyisipan dilakukan dalam antrian. Setelah memasukkan elemen ke dalam antrian yang ditunjukkan pada gambar di atas, antrian akan terlihat seperti berikut. Nilai belakang akan menjadi 5 sedangkan nilai depan tetap sama.



Setelah menghapus elemen, nilai front akan meningkat dari -1 menjadi 0. Namun, antrian akan terlihat seperti berikut.



* **Implementasi menggunakan Daftar tertaut:** Alokasi daftar tertaut dalam Antrian dapat diimplementasikan menggunakan daftar tertaut.

Karena kelemahan yang dibahas di bagian sebelumnya dari tutorial ini, implementasi array tidak dapat digunakan untuk aplikasi skala besar di mana antrian diimplementasikan. Salah satu alternatif implementasi array adalah implementasi linked list dari antrian.

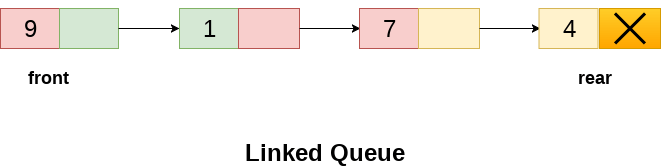
Persyaratan penyimpanan representasi terkait antrian dengan n elemen adalah o(n) sedangkan persyaratan waktu untuk operasi adalah o(1).

Dalam sebuah linked queue, setiap node dari antrian terdiri dari dua bagian yaitu bagian data dan bagian link. Setiap elemen antrian menunjuk ke elemen berikutnya langsung dalam memori.

Dalam antrian yang terhubung, ada dua pointer yang disimpan di memori yaitu pointer depan dan pointer belakang. Pointer depan berisi alamat elemen awal antrian sedangkan pointer belakang berisi alamat elemen terakhir dari antrian.

Penyisipan dan penghapusan masing-masing dilakukan di bagian belakang dan depan. Jika depan dan belakang keduanya NULL, itu menandakan antrian kosong.

Representasi terkait antrian ditunjukkan pada gambar berikut.



**Rinkasan:**

Struktur Data Linier

Mengikuti FIFO: First In First Out

Penyisipan dapat dilakukan dari bagian belakang.

Penghapusan dapat dilakukan dari ujung depan.

Misal: antri di loket tiket, terminal bus

**4 operasi utama:**

enqueue(ele) – digunakan untuk menyisipkan elemen di atas

dequeue() – menghapus elemen teratas dari antrian

peekfirst() – untuk mendapatkan elemen pertama dari antrian

peeklast() – untuk mendapatkan elemen terakhir dari antrian

Semua operasi bekerja dalam waktu yang konstan yaitu, O(1)

**Keuntungan**

Menjaga data dengan cara FIFO

Penyisipan dari awal dan penghapusan dari akhir membutuhkan waktu O(1)

**Aplikasi**

Penjadwalan

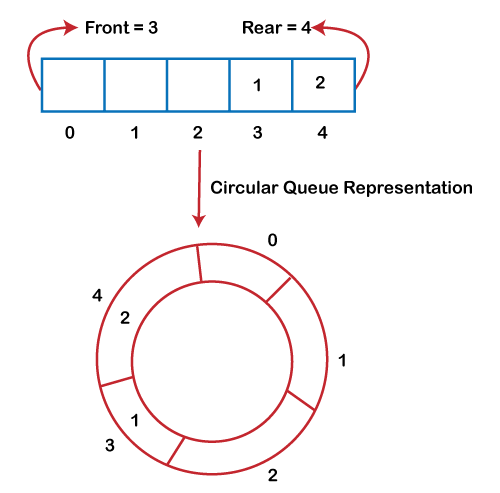
Mempertahankan daftar putar

Penanganan interupsi

Antrian Melingkar

Mengapa konsep antrian melingkar diperkenalkan?

Ada satu batasan dalam implementasi array [Queue](https://www.javatpoint.com/data-structure-queue) . Jika bagian belakang mencapai posisi akhir Antrian maka mungkin ada beberapa ruang kosong yang tersisa di awal yang tidak dapat dimanfaatkan. Maka, untuk mengatasi keterbatasan tersebut, konsep antrian melingkar diperkenalkan.



Seperti yang bisa kita lihat pada gambar di atas, bagian belakang berada di posisi terakhir Antrian dan bagian depan menunjuk ke suatu tempat daripada posisi ke- 0 . Dalam array di atas, hanya ada dua elemen dan tiga posisi lainnya kosong. Bagian belakang berada di posisi terakhir Antrian; jika kita mencoba menyisipkan elemen maka akan terlihat bahwa tidak ada ruang kosong di Queue. Ada satu solusi untuk menghindari pemborosan ruang memori dengan menggeser kedua elemen di kiri dan menyesuaikan bagian depan dan belakang. Ini bukan pendekatan yang praktis karena memindahkan semua elemen akan menghabiskan banyak waktu. Pendekatan yang efisien untuk menghindari pemborosan memori adalah dengan menggunakan struktur data antrian melingkar.

Apa itu Antrian Melingkar?

Antrian melingkar mirip dengan antrian linier karena juga didasarkan pada prinsip FIFO (First In First Out) kecuali bahwa posisi terakhir terhubung ke posisi pertama dalam antrian melingkar yang membentuk lingkaran. Itu juga dikenal sebagai ***Penyangga Cincin*** .

Operasi pada Antrian Melingkar

Berikut ini adalah operasi yang dapat dilakukan pada antrian melingkar:

* **Depan:** Digunakan untuk mendapatkan elemen depan dari Antrian.
* **Belakang:** Digunakan untuk mendapatkan elemen belakang dari Antrian.
* **enQueue(value):** Fungsi ini digunakan untuk memasukkan nilai baru ke dalam Queue. Elemen baru selalu disisipkan dari bagian belakang.
* **deQueue():** Fungsi ini menghapus elemen dari Antrian. Penghapusan dalam Antrian selalu terjadi dari ujung depan.

Aplikasi Antrian Melingkar

**Antrian melingkar dapat digunakan dalam skenario berikut:**

* **Manajemen memori:** Antrian melingkar menyediakan manajemen memori. Seperti yang telah kita lihat bahwa dalam antrian linier, memori tidak dikelola dengan sangat efisien. Tetapi dalam kasus antrian melingkar, memori dikelola secara efisien dengan menempatkan elemen di lokasi yang tidak digunakan.
* **Penjadwalan CPU:** Sistem operasi juga menggunakan antrian melingkar untuk memasukkan proses dan kemudian menjalankannya.
* **Sistem lalu lintas:** Dalam sistem lalu lintas yang dikendalikan komputer, lampu lalu lintas adalah salah satu contoh terbaik dari antrian melingkar. Setiap lampu lalu lintas menyala satu per satu setelah setiap jinterval waktu. Seperti lampu merah menyala selama satu menit kemudian lampu kuning selama satu menit dan kemudian lampu hijau. Setelah lampu hijau, lampu merah menyala.

[IKLAN OLEH ADRECOVER](https://adrecover.com/?utm_campaign=ByAdRecover&utm_source=www.javatpoint.com&utm_medium=ByAdRecover-728)

Operasi antrian

**Langkah-langkah operasi enqueue diberikan di bawah ini:**

* Pertama, kita akan mengecek apakah Antrian sudah penuh atau belum.
* Awalnya bagian depan dan belakang disetel ke -1. Saat kita memasukkan elemen pertama dalam Antrian, depan dan belakang keduanya disetel ke 0.
* Saat kita memasukkan elemen baru, bagian belakang bertambah, yaitu, ***rear=rear+1*** .

Deque (atau antrian berujung ganda)

Pada artikel kali ini, kita akan membahas tentang double-ended queue atau deque. Pertama-tama kita harus melihat deskripsi singkat tentang antrian.

Apa itu antrian?

Antrian adalah struktur data di mana apa yang lebih dulu akan keluar lebih dulu, dan mengikuti kebijakan FIFO (First-In-First-Out). Penyisipan dalam antrian dilakukan dari satu ujung yang dikenal sebagai ujung **belakang** atau **ekor,** sedangkan penghapusan dilakukan dari ujung lain yang dikenal sebagai ujung **depan** atau **kepala** antrian.

Contoh antrian di dunia nyata adalah antrian tiket di luar gedung bioskop, di mana orang yang masuk lebih dulu dalam antrian mendapatkan tiket terlebih dahulu, dan orang yang masuk terakhir dalam antrian mendapatkan tiket terakhir.

Apa itu Deque (atau antrian berujung ganda)

Deque adalah singkatan dari Double Ended Queue. Deque adalah struktur data linier di mana operasi penyisipan dan penghapusan dilakukan dari kedua ujungnya. Kita dapat mengatakan bahwa deque adalah versi umum dari antrian.

Meskipun penyisipan dan penghapusan dalam deque dapat dilakukan di kedua ujungnya, itu tidak mengikuti aturan FIFO. Representasi deque diberikan sebagai berikut -



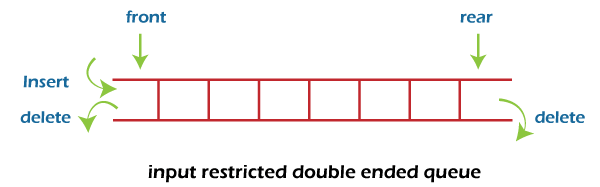
Jenis deque

Ada dua jenis deque -

* Masukan antrian terbatas
* Keluaran antrian terbatas

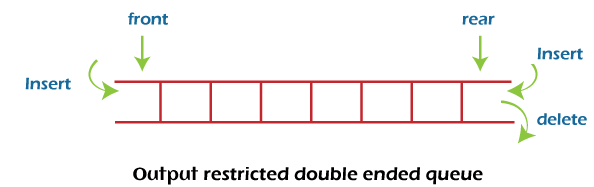
**Masukan Antrian terbatas**

Dalam antrian input terbatas, operasi penyisipan dapat dilakukan hanya pada satu ujung, sedangkan penghapusan dapat dilakukan dari kedua ujung.



**Antrian keluaran terbatas**

Dalam antrian keluaran terbatas, operasi penghapusan hanya dapat dilakukan pada satu ujung, sedangkan penyisipan dapat dilakukan dari kedua ujung.



Operasi dilakukan pada deque

Ada operasi berikut yang dapat diterapkan pada deque -

* Sisipan di depan
* Penyisipan di belakang
* Penghapusan di depan
* Penghapusan di belakang

[IKLAN OLEH ADRECOVER](https://adrecover.com/?utm_campaign=ByAdRecover&utm_source=www.javatpoint.com&utm_medium=ByAdRecover-728)

Kami juga dapat melakukan operasi mengintip di deque bersama dengan operasi yang tercantum di atas. Melalui operasi mengintip, kita bisa mendapatkan elemen depan dan belakang deque dari deque. Jadi, selain operasi di atas, operasi berikut juga didukung dalam deque -

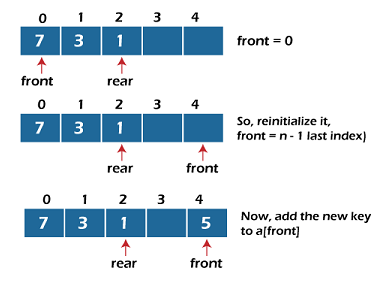
* Dapatkan item depan dari deque
* Dapatkan item belakang dari deque
* Cek apakah deque sudah penuh atau belum
* Cek apakah deque kosong atau tidak

Sekarang, mari kita pahami operasi yang dilakukan pada deque menggunakan sebuah contoh.

**Penyisipan di ujung depan**

Dalam operasi ini, elemen dimasukkan dari ujung depan antrian. Sebelum melakukan operasi, terlebih dahulu kita harus mengecek apakah antrian sudah penuh atau belum. Jika antrian tidak penuh, maka elemen dapat dimasukkan dari ujung depan dengan menggunakan kondisi di bawah ini -

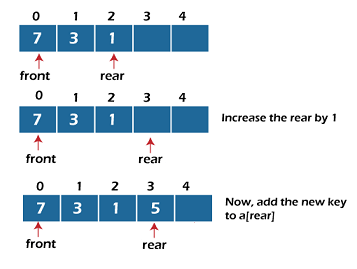
* Jika antrian kosong, baik belakang dan depan diinisialisasi dengan 0. Sekarang, keduanya akan menunjuk ke elemen pertama.
* Jika tidak, periksa posisi depan jika depan kurang dari 1 (depan < 1), lalu inisialisasi ulang dengan **front = n - 1** , yaitu indeks terakhir dari array.



**Penyisipan di bagian belakang**

Dalam operasi ini, elemen dimasukkan dari ujung belakang antrian. Sebelum melakukan operasi, terlebih dahulu kita harus mengecek kembali apakah antrian sudah penuh atau belum. Jika antrian tidak penuh, maka elemen dapat dimasukkan dari ujung belakang dengan menggunakan kondisi di bawah ini -

* Jika antrian kosong, baik belakang dan depan diinisialisasi dengan 0. Sekarang, keduanya akan menunjuk ke elemen pertama.
* Jika tidak, tambah bagian belakang sebesar 1. Jika bagian belakang pada indeks terakhir (atau ukuran - 1), maka alih-alih menambahnya sebesar 1, kita harus membuatnya sama dengan 0.



**Penghapusan di ujung depan**

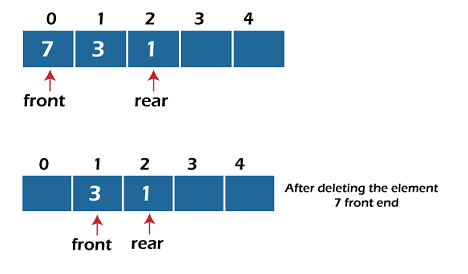
Dalam operasi ini, elemen dihapus dari ujung depan antrian. Sebelum mengimplementasikan operasi tersebut, terlebih dahulu kita harus mengecek apakah antrian tersebut kosong atau tidak.

Jika antrian kosong, yaitu front = -1, itu adalah kondisi underflow, dan kami tidak dapat melakukan penghapusan. Jika antrian tidak penuh, maka elemen dapat dimasukkan dari ujung depan dengan menggunakan kondisi di bawah ini -

Jika deque hanya memiliki satu elemen, setel belakang = -1 dan depan = -1.

Lain jika depan di ujung (itu berarti depan = ukuran - 1), set depan = 0.

Jika tidak, tambah bagian depan dengan 1, (yaitu, depan = depan + 1).



**Penghapusan di bagian belakang**

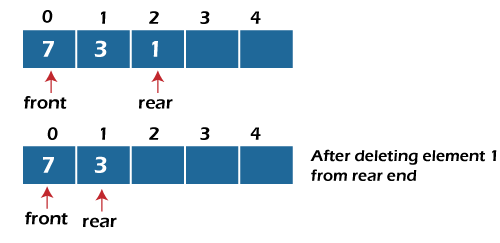
Dalam operasi ini, elemen dihapus dari ujung belakang antrian. Sebelum mengimplementasikan operasi tersebut, terlebih dahulu kita harus mengecek apakah antrian tersebut kosong atau tidak.

Jika antrian kosong, yaitu front = -1, itu adalah kondisi underflow, dan kami tidak dapat melakukan penghapusan.

Jika deque hanya memiliki satu elemen, setel belakang = -1 dan depan = -1.

Jika belakang = 0 (belakang ada di depan), maka atur belakang = n - 1.

Jika tidak, kurangi bagian belakang sebesar 1 (atau, belakang = belakang -1).



**Cek kosong**

Operasi ini dilakukan untuk memeriksa apakah deque kosong atau tidak. Jika front = -1, berarti deque kosong.

**Periksa penuh**

Operasi ini dilakukan untuk mengecek apakah deque sudah penuh atau belum. Jika depan = belakang + 1, atau depan = 0 dan belakang = n - 1 berarti deque sudah penuh.

Kompleksitas waktu dari semua operasi deque di atas adalah O(1), yaitu, konstan.

**Aplikasi deque**

* Deque dapat digunakan sebagai stack dan queue, karena mendukung kedua operasi tersebut.
* Deque dapat digunakan sebagai pemeriksa palindrom artinya jika kita membaca string dari kedua ujungnya, string akan sama.

Apa itu antrian prioritas?

Antrian prioritas adalah tipe data abstrak yang berperilaku mirip dengan antrian normal kecuali bahwa setiap elemen memiliki beberapa prioritas, yaitu elemen dengan prioritas tertinggi akan datang pertama dalam antrian prioritas. Prioritas elemen dalam antrian prioritas akan menentukan urutan elemen yang akan dihapus dari antrian prioritas.

Antrian prioritas hanya mendukung elemen yang sebanding, yang berarti bahwa elemen tersebut disusun dalam urutan menaik atau menurun.

Sebagai contoh, misalkan kita memiliki beberapa nilai seperti 1, 3, 4, 8, 14, 22 yang dimasukkan dalam antrian prioritas dengan urutan yang dikenakan pada nilai-nilai tersebut dari yang terkecil hingga yang terbesar. Oleh karena itu, 1 angka akan memiliki prioritas tertinggi sedangkan 22 akan memiliki prioritas terendah.

Karakteristik antrian Prioritas

Antrian prioritas adalah perpanjangan dari antrian yang memiliki karakteristik sebagai berikut:

* Setiap elemen dalam antrian prioritas memiliki beberapa prioritas yang terkait dengannya.
* Elemen dengan prioritas lebih tinggi akan dihapus sebelum penghapusan dengan prioritas lebih rendah.
* Jika dua elemen dalam antrian prioritas memiliki prioritas yang sama, maka akan diatur menggunakan prinsip FIFO.

**Mari kita pahami antrian prioritas melalui sebuah contoh.**

Kami memiliki antrian prioritas yang berisi nilai-nilai berikut:

**1, 3, 4, 8, 14, 22**

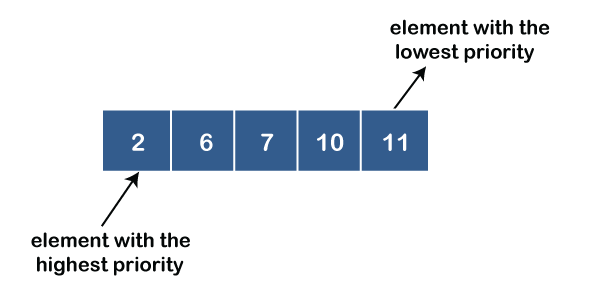
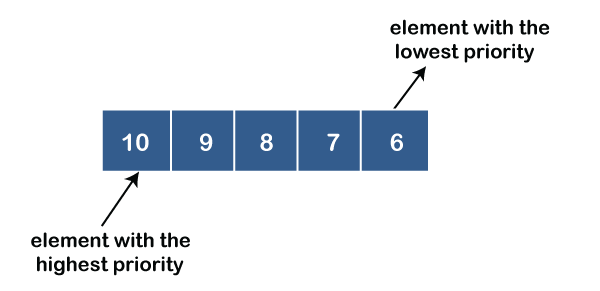
Semua nilai diatur dalam urutan menaik. Sekarang, kita akan mengamati bagaimana antrian prioritas akan terlihat setelah melakukan operasi berikut:

* **poll():** Fungsi ini akan menghapus elemen prioritas tertinggi dari antrian prioritas. Pada antrian prioritas di atas, elemen '1' memiliki prioritas tertinggi, sehingga akan dihapus dari antrian prioritas.
* **add(2):** Fungsi ini akan menyisipkan elemen '2' dalam antrian prioritas. Karena 2 adalah elemen terkecil di antara semua angka sehingga akan mendapatkan prioritas tertinggi.
* **poll():** Ini akan menghapus elemen '2' dari antrian prioritas karena memiliki antrian prioritas tertinggi.
* **add(5):** Ini akan memasukkan 5 elemen setelah 4 karena 5 lebih besar dari 4 dan lebih kecil dari 8, sehingga akan mendapatkan prioritas tertinggi ketiga dalam antrian prioritas.

[IKLAN OLEH ADRECOVER](https://adrecover.com/?utm_campaign=ByAdRecover&utm_source=www.javatpoint.com&utm_medium=ByAdRecover-728)

Jenis Antrian Prioritas

**Ada dua jenis antrian prioritas:**

* **Antrian prioritas urutan naik:** Dalam antrian prioritas urutan menaik, nomor prioritas yang lebih rendah diberikan sebagai prioritas yang lebih tinggi dalam prioritas. Misalnya, kita ambil angka dari 1 sampai 5 yang disusun dalam urutan menaik seperti 1,2,3,4,5; oleh karena itu, angka terkecil, yaitu 1 diberikan sebagai prioritas tertinggi dalam antrian prioritas.  
  
* **Antrian prioritas urutan menurun:** Dalam antrian prioritas urutan menurun, nomor prioritas yang lebih tinggi diberikan sebagai prioritas yang lebih tinggi dalam prioritas. Misalnya, kita mengambil angka dari 1 hingga 5 yang disusun dalam urutan menurun seperti 5, 4, 3, 2, 1; oleh karena itu, angka terbesar, yaitu 5 diberikan sebagai prioritas tertinggi dalam antrian prioritas.  
  

Representasi antrian prioritas

Sekarang, kita akan melihat bagaimana merepresentasikan antrian prioritas melalui daftar satu arah.

Kami akan membuat antrian prioritas dengan menggunakan daftar yang diberikan di bawah ini di mana daftar **INFO** berisi elemen data, daftar **PRN** berisi nomor prioritas dari setiap elemen data yang tersedia dalam daftar **INFO** , dan LINK pada dasarnya berisi alamat node berikutnya.



**Mari kita buat antrian prioritas langkah demi langkah.**

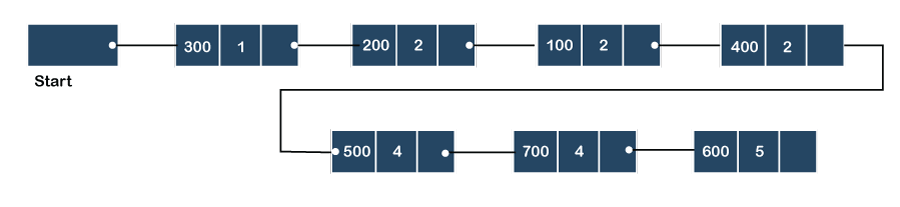
**Dalam kasus antrian prioritas, nomor prioritas yang lebih rendah dianggap sebagai prioritas yang lebih tinggi, yaitu** nomor prioritas yang lebih rendah = prioritas yang lebih tinggi.

**Langkah 1:** Dalam daftar, nomor prioritas lebih rendah adalah 1, yang nilai datanya 333, sehingga akan dimasukkan ke dalam daftar seperti yang ditunjukkan pada diagram di bawah ini:

**Langkah 2:** Setelah memasukkan 333, prioritas nomor 2 memiliki prioritas yang lebih tinggi, dan nilai data yang terkait dengan prioritas ini adalah 222 dan 111. Jadi, data ini akan dimasukkan berdasarkan prinsip FIFO; oleh karena itu 222 akan ditambahkan terlebih dahulu dan kemudian 111.

**Langkah 3:** Setelah memasukkan elemen prioritas 2, angka prioritas berikutnya yang lebih tinggi adalah 4 dan elemen data yang terkait dengan 4 angka prioritas adalah 444, 555, 777. Dalam hal ini, elemen akan dimasukkan berdasarkan prinsip FIFO; oleh karena itu, 444 akan ditambahkan terlebih dahulu, lalu 555, dan kemudian 777.

**Langkah 4:** Setelah memasukkan elemen prioritas 4, nomor prioritas berikutnya yang lebih tinggi adalah 5, dan nilai yang terkait dengan prioritas 5 adalah 666, sehingga akan dimasukkan di akhir antrian.



Implementasi Antrian Prioritas

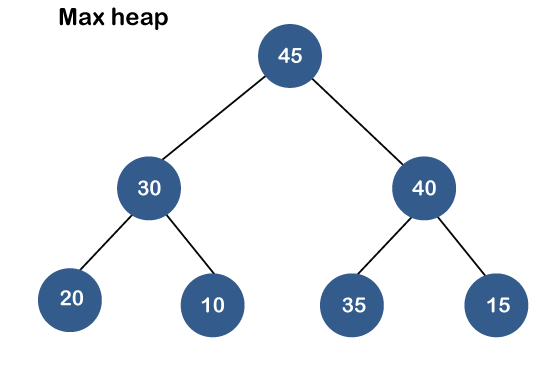
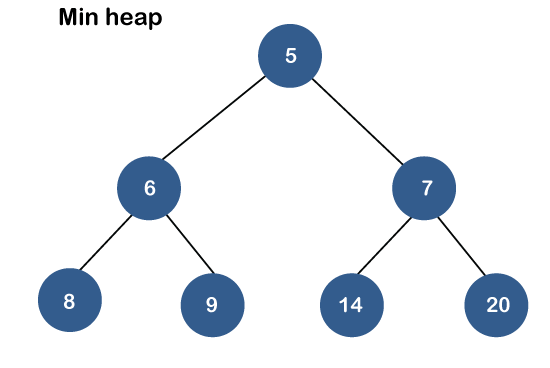
Antrian prioritas dapat diimplementasikan dalam empat cara yang mencakup array, daftar tertaut, struktur data heap, dan pohon pencarian biner. Struktur data heap adalah cara paling efisien untuk mengimplementasikan antrian prioritas, jadi kami akan mengimplementasikan antrian prioritas menggunakan struktur data tumpukan dalam topik ini. Sekarang, pertama-tama kita pahami alasan mengapa heap adalah cara paling efisien di antara semua struktur data lainnya.

**Analisis kompleksitas menggunakan implementasi yang berbeda**

|  |  |  |  |
| --- | --- | --- | --- |
| Penerapan | Menambahkan | Menghapus | mengintip |
| Daftar tertaut | O(1) | Pada) | Pada) |
| tumpukan biner | O (masuk) | O (masuk) | O(1) |
| Pohon pencarian biner | O (masuk) | O (masuk) | O(1) |

Apa itu Heap?

Heap adalah struktur data berbasis pohon yang membentuk pohon biner lengkap, dan memenuhi properti heap. Jika A adalah simpul induk dari B, maka A diurutkan terhadap simpul B untuk semua simpul A dan B dalam suatu heap. Artinya nilai simpul induk bisa lebih dari atau sama dengan nilai simpul anak, atau nilai simpul induk bisa lebih kecil atau sama dengan nilai simpul anak. Oleh karena itu, kita dapat mengatakan bahwa ada dua jenis tumpukan:

* **Max heap:** Max heap adalah tumpukan di mana nilai node induk lebih besar dari nilai node anak.  
  
* **Min heap:** Min heap adalah heap di mana nilai node induk lebih kecil dari nilai node anak.  
  

Kedua tumpukan adalah tumpukan biner, karena masing-masing memiliki tepat dua simpul anak.

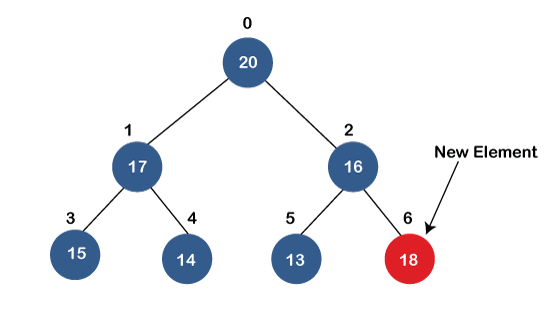
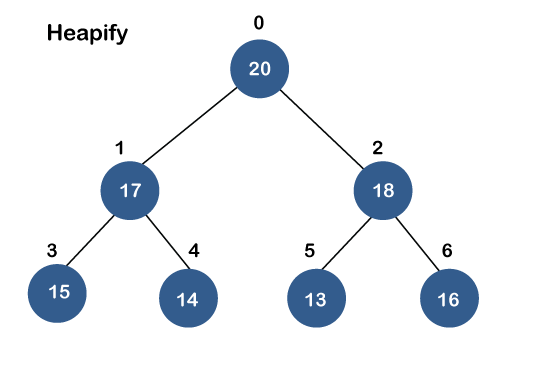
Operasi Antrian Prioritas

Operasi umum yang dapat kita lakukan pada antrian prioritas adalah penyisipan, penghapusan dan mengintip. Mari kita lihat bagaimana kita dapat mempertahankan struktur data heap.

* **Memasukkan elemen dalam antrian prioritas (tumpukan maksimum)**

Jika kita memasukkan elemen dalam antrian prioritas, itu akan pindah ke slot kosong dengan melihat dari atas ke bawah dan kiri ke kanan.

Jika elemen tidak berada di tempat yang benar, maka elemen tersebut dibandingkan dengan simpul induk; jika ditemukan rusak, elemen ditukar. Proses ini berlanjut sampai elemen ditempatkan pada posisi yang benar.

* **Menghapus elemen minimum dari antrian prioritas**

Seperti yang kita ketahui bahwa dalam tumpukan maksimum, elemen maksimum adalah simpul akar. Saat kami menghapus node root, itu membuat slot kosong. Elemen yang terakhir dimasukkan akan ditambahkan di slot kosong ini. Kemudian, elemen ini dibandingkan dengan node anak, yaitu, anak kiri dan anak kanan, dan bertukar dengan yang lebih kecil dari keduanya. Itu terus bergerak ke bawah pohon sampai properti heap dipulihkan.

Aplikasi antrian Prioritas

**Berikut ini adalah aplikasi dari antrian prioritas:**

* Ini digunakan dalam algoritma jalur terpendek Dijkstra.
* Ini digunakan dalam algoritma prim
* Ini digunakan dalam teknik kompresi data seperti kode Huffman.
* Ini digunakan dalam heap sort.
* Ini juga digunakan dalam sistem operasi seperti penjadwalan prioritas, penyeimbangan beban, dan penanganan interupsi.