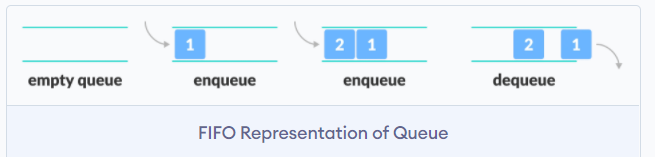
**Queue**

Eleman ekleme işleminin sondan, eleman çıkarma işleminin baştan yapıldığı veri yapısıdır.

* Dizi veya liste üzerinde gerçekleştirilebilir
* Doğrusal veri yapısıdır
* İlk gelen ilk çıkar (FI



**Temel Kuyruk İşlemleri**

Enqueue: eleman ekler (sondan)

Dequeue: eleman çıkartır (baştan)

isEmpty: kuyruk boş mu dolu mu

cout: toplam eleman (stack aksine kuyruktaki eleman sayısı sorulabilir)

peek: çıkmak üzere elemanı döndürür (en öndeki)

clear: kuyruk temizleme

create: kuyruk oluşturma

Kuyruk oluşumu 3 farklı şekilde yapılabilir:

1. Dizi üzerinde kaydırmalı
2. Dizi üzerinde çevrimsel
3. Linked list ile

**Dizi Üzerinde**

**Simple Queue**

Dairesellik yok. Önceden boyut belirlenir. Elemanlar tek tek eklenir kuyruk dolduğunda kapasite iki katına çıkartılır

**Kuyruğun Çalışması**

-iki deger tutulur front ve rear

-front kuyruktaki ilk eleman

-rear kuyruktaki son eleman

-başlangıçta front ve rear -1 ayarlanır

**Ekleme İşlemi**

1-kuyruk boş mu dolu mu kontrol edilir

2- ilk eleman için front değeri 0 ayarlanır

3-rear değeri 1 arttırılır

4-yeni eleman rear ile gösterilen konuma eklenir

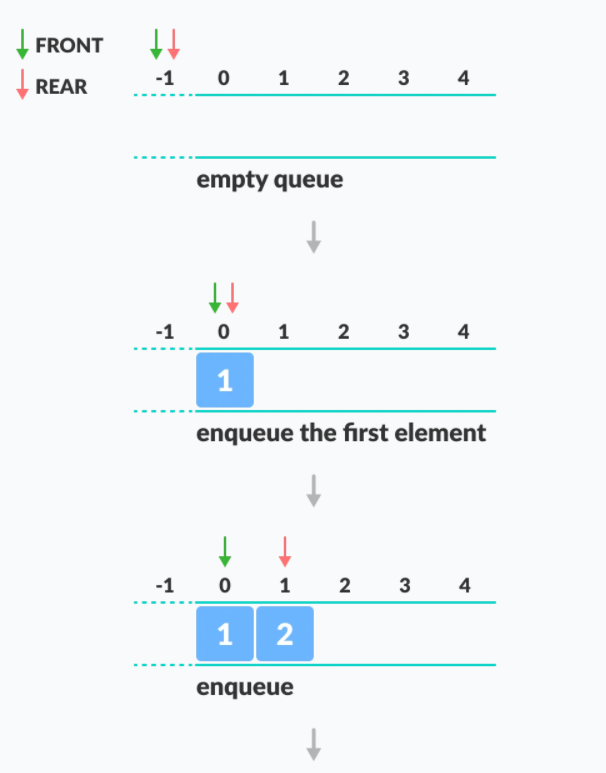
**Çıkarma İşlemi**

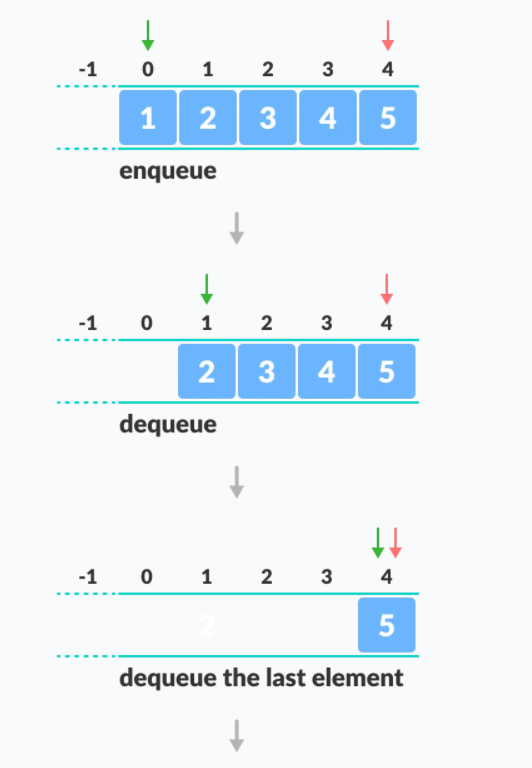
1-kuyruk boş mu kontrol et

2-front’u döndür

3-front ile index 1 arttırılır

4-son elemın ise front ve rear değerleri -1 olarak al







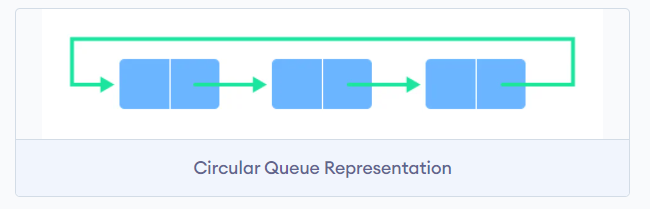
**Circular Queue // dizi üzerinde yapacaksan bunu yap bu daha verimli**

Daha iyi bellek kullanımı sağlar. Boş alanları da kullanabiliriz.

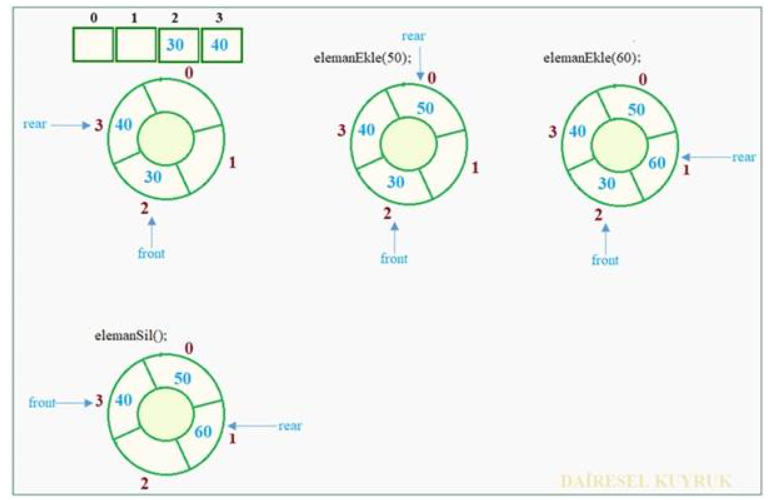
Dört kapasiteli bir dizimiz olsun. 0. ve 1. index elemanları kuyruktan çıkmış

Eleman eklemek istediğimizde kuyruk dolu hatası ile karşılaşmamak için ilk yöntem elemanları öne kaydırmaktır. Her silmeden sonra kaydırma yapılır ve çok yavaştır

Bu yöntem ise dairesel kuyruk oluşturmaktır. Boşalan kısma elemanlar eklenir. Kuyruktaki veriler bir tekerlek üzerinde dönüyormuş gibi davranır.



Dairesel Kuyruk, işlemlerin FIFO (İlk Giren İlk Çıkar) prensibine göre gerçekleştirildiği ve son konumun bir daire oluşturmak için ilk konuma bağlandığı doğrusal bir veri yapısıdır. Bellek yönetimi ve CPU işlemlerini planlamak gibi önemli kullanım alanları vardır. Aşağıda şekilde dairesel kuyruk yapısı gösterilmiştir.



metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Sırasıyla a b c d çıkar. Yeni diziye kopyalarken düzgün bir şekilde kopyalamalıyız.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Kuyruğun Çalışması**

-iki deger tutulur front ve rear

-front kuyruktaki ilk eleman

-rear kuyruktaki son eleman

-başlangıçta front ve rear -1 ayarlanır

**Ekleme İşlemi**

1-kuyruk boş mu dolu mu kontrol edilir

2- ilk eleman için front değeri 0 ayarlanır

3-rear değeri dairesel olarak 1 arttırılır. Rear, sona ulaşırsa sonraki sıranın başında olur

4-yeni eleman rear ile gösterilen konuma eklenir

**Çıkarma İşlemi**

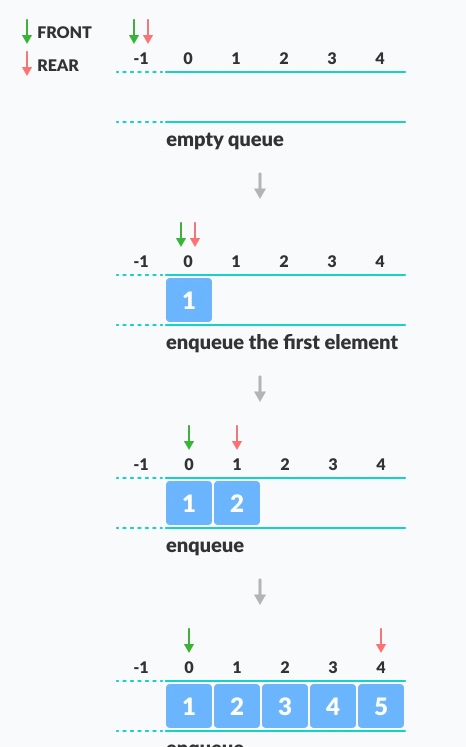
1-kuyruk boş mu kontrol et

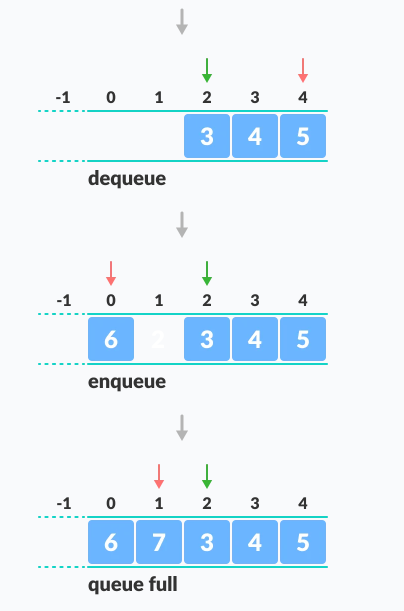
2-front’u döndür

3-front ile indexi dairesel 1 arttırılır

4-son elemın ise front ve rear değerleri -1 olarak al

Kuyruk full dolduğunda yeni bir durum vardır: front = rear + 1 dairesellikten dolayı bu oluşur





**Öncelikli Kuyruk**

Her zaman ilk giren işlem görmez. Bankaları düşündüğümüz zaman bu böyledir. Bazen aradan da elemanın silindiği yapılar vardır. Bu yapı öncelikli kuyruktur. Ekleme yine aynı, çıkartma farklı. Öncelik sırasına sokulur ve ona göre sırasıyla çıkartma yapılır.

**Öncelikli Kuyruk Dizi Üzerinde mi Liste Üzerinde mi Gerçekleştirilmeli?**

Liste üzerinde. Çünkü aradan eleman çıkıyor. Eğer dizi olsaydı yine sağa sola kayma olayı olacaktı ve performans kaybı olurdu.

**Liste Üzerinde**

Tek yönlü bağlı liste yapısı ile hemen hemen aynıdır. Ama bu sefer tail de tutulur.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Front, çıkacak elemanı gösterir.

Eğer back tutmasaydın her seferinde ekleme için kuyruk sonuna gitmemiz gerekirdi

Back yazığımız için ilk elemanı silem ve ilk elemanı ekleme dizi ve liste için aynı performanslıdır. O(1)

**Nerelerde Kullanılır?**

* İşletim sisteminin proseslerinin bilgisayarın kaynaklarını paylaştırılması (CPU scheduling, Disk scheduling)
* giriş çıkış ünitelerinin kullandırılması queue ile yapılır. (IO buffers, pipes, dile IO senkronizasyon kullanmak için)
* gerçek zamanlı sistemlerde interrupt ları elimizde tutmak için
* Bundan başka bilgisayarda yazdırma işlemlerinde de kuyruk yapısı kullanılır. Çünkü yazıcılar görece bilgisayarlara göre daha yavaştır. Dolayısıyla bilgisayarda bir darboğaz oluşturmamak için yazdırma işleri kuyruğa alınır ve sırayla yazdırılır. Bu tam bir kuyruk uygulamasıdır.
* Yine bilgisayarlarda kuyruk uygulamasına başka bir örnek, klavye ve fare işlemlerinin ardışık olarak okunmasını gösterebiliriz.
* Gerçek hayatta ise bir fırında kuyruğa girmek verilebilir. İlk gelen ilk hizmet görür. Ya da çağrı merkezleri arayan kişileri sırayla tutmak için queue yapısını kullanır

**Kuyruk Dizi Üzerinde mi Yapılmalı Liste Üzerinde mi?**

Create: aynı O(1)

Enqueue: dolma ihtimali olduğu için liste avantajlı. (Dizide kapasite iki katına çıkardıktan sonra) ikisi de O(1)

Dequeue: aynı (Dizide kullanmadığımız yeri bellekten silemeyiz ikisi de aynı ama liste daha avantajlı) ikisi de O(1)

isEmpty: aynı O(1)

count: aynı

peek: aynı

clear: dizi avantajlı. Listede tüm düğümler tek tek silinmesi gerektiği için avantajlı değil

**Radix Sort**

Radix sort algoritmasında öncelikli kuyruk kullanılabilir.

Basamak sayıları fazla olan sayıları karşılaştırmada daha mantıklı.

301, 420, 340, 621, 200 sayılarını karşılaştıralım

* İlk önce birler basamağını karşılaştırmak için 10 tane kuryruk oluşturulur. (bu kuyruklar dizide tutulabilir)

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Sonra onlar basamağına göre bu sayıları güncelliyoruz

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Yüzler basamağına göre okumayalım

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

* Sonra dequeue işlemi ile sıralanmış sayıları yazdırıyoruz

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Sıralama algoritması O(n) dir

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

Clear aşırı kullanılmıyorsa liste tercih edilmeli. Dizinin dolma ihtimaline karşı

NOT: O(1) demek milyonlarda veri de olsa çalışma süresi hep aynı kalacak ve hiç değişmeyecek hepsi için aynı olacak demek