Veri Yapıları ve Algoritmalar

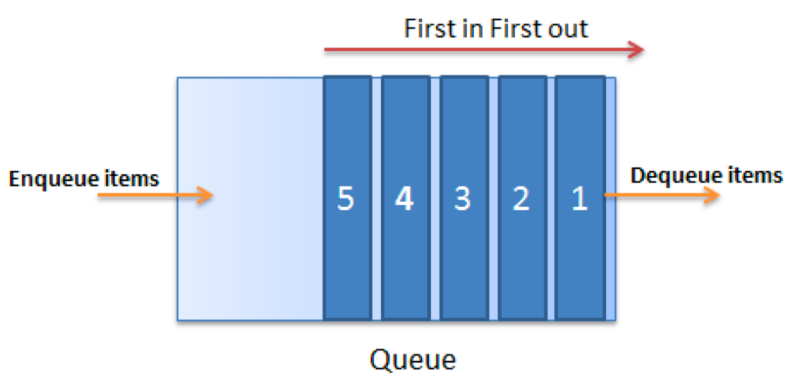
Kuyruk (Queue)

Yunus Santur , ysantur@gmail.com

Kuyruk

- FIFO (First In First Out): İlk giren ilk çıkar, veya
- LILO (Last In Last Out):Son giren son çıkar

- İlk giren ilk çıkar (günlük hayatta ki kuyruklara benzer)



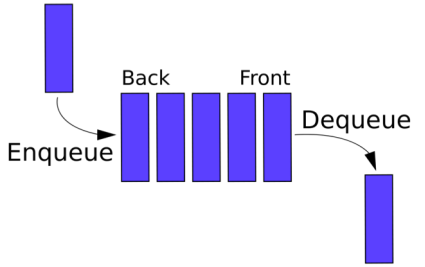
http://www.tutorialsteacher.com/Content/images/csharp/csharp-queue.png

Kuyruk

- İki temel işlev vardır
  - enqueue/insert :Elemanı kuyruğun en sonuna ekler.
  - dequeue/remove :Kuyruğun en başındaki elemanı çeker (***silerek***).

- Diğer İşlevler
  - size() Kuyrukta ki eleman sayısı
  - front() Kuyruğun başındaki elemanı *silmeden* çeker.
  - isempty() Kuyruk boşmu?

- İstisna durumları
  - Boş kuyruktan eleman çekmeye çalışmak



http://www.tutorialsteacher.com/Content/images/csharp/csharp-queue.png

Kuyruk

- Kuyrukları gerçekleştirmek için, diziler ve bağlı listeler kullanılabilir.

Öncelikli Kuyruklar

- Geleneksel olarak her eleman en sona eklenir, eleman çekileceği zaman en baştan seçilirdi (FIFO).
- İşletim sistemleri birden fazla process'i kuyruk prensibi ile ele alır.
- Sırası gelen process çalışır ve tekrar en sona eklenir.
- Problem: Bazı processler öncelikli olabilir?

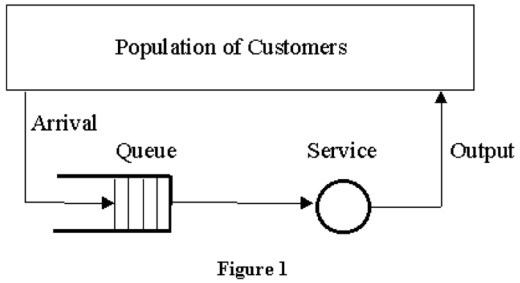


Figure 1

http://staff.um.edu.mt/jskl1/simweb/fig1.gif

Liste Üzerinde Kuyruk Gerçekleştirimi

- İşlemler

- x=Queue()
- x.insert(1)
- x.remove()
- x.front()
- x.size()
- x.show()

- Python kodu

```
class Queue():
    def __init__(self):
        self.entries = []
        self.length = 0
        self.front=0


    def insert(self, item):
        self.entries.append(item)
        self.length = self.length + 1


    def remove(self):
        if self.length>0:
            item=self.entries[0]
            self.length = self.length - 1
            del self.entries[0]
            return item
        else:
            print("Kuyruk boş")


    def front(self):
        return self.entries[0]

    def size(self):
        return self.length

    def show(self):
        print(self.entries)
```

Kuyruk Gerçekleştirimi

- Sınıf oluşturma ve yapılandırıcı metot

```
class Queue():
    def __init__(self):
        self.entries = []
        self.length = 0
        self.front=0
```

Kuyruk Gerçekleştirimi

- Kuyruğa eleman ekleme

```
def insert(self, item):
    self.entries.append(item)
    self.length = self.length + 1
```

Kuyruk Gerçekleştirimi

- Kuyruğun başındaki elemanı seçme

```
def remove(self):
    if self.length>0:
        item=self.entries[0]
        self.length = self.length - 1
        del self.entries[0]
        return item
    else:
        print("Kuyruk boş")
```

Kuyruk Gerçekleştirimi

- Kuyruğun başındaki elemanı göster
- Kuyruğun eleman sayısı
- Kuyruktaki tüm elemanlar

```
def front(self):
    return self.entries[0]

def size(self):
    return self.length

def show(self):
    print(self.entries)
```

Ödev

- İşletim sisteminde çoklu process çalıştırma yapısını kuyruk ile simüle etmek istiyoruz.
- N adet process ve bu processlerin toplam çalışacağı süreyi gireceğiz.
- Her process x süre çalışacak, eğer işi bittiyse kuyruktan çıkacak, bitmediyse tekrar en sona eklenecek.
- Gerçekleştireceğiniz program adım adım hangi process ne kadar süre çalıştı, kuyruğun o andaki durumunu göstermeli.
- İşlemcide geçen toplam süre=processlerin toplam süresi olmalı
- Program web tabanlı/görsel olarak ta yazılabilir.