Veri Yapıları ve Algoritmalar

Yığıt (Stack)

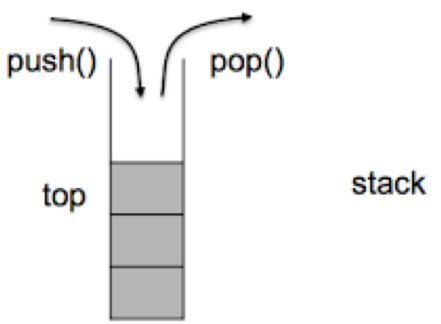
Yunus Santur , ysantur@gmail.com

Kuyruklar

- FIFO (First In First Out)
  - İlk giren ilk çıkar
- LIFO (Last In First Out)
  - Son giren ilk çıkar

Yığıt (Stack)

- LIFO (Son giren ilk çıkar)
- Eleman ekleneceği zaman en üste eklenir (top)
- Eleman çıkarılacağı zaman en üstten çekilir (top)



https://www.algoritmauzmani.com/wp-content/uploads/2017/08/images.png

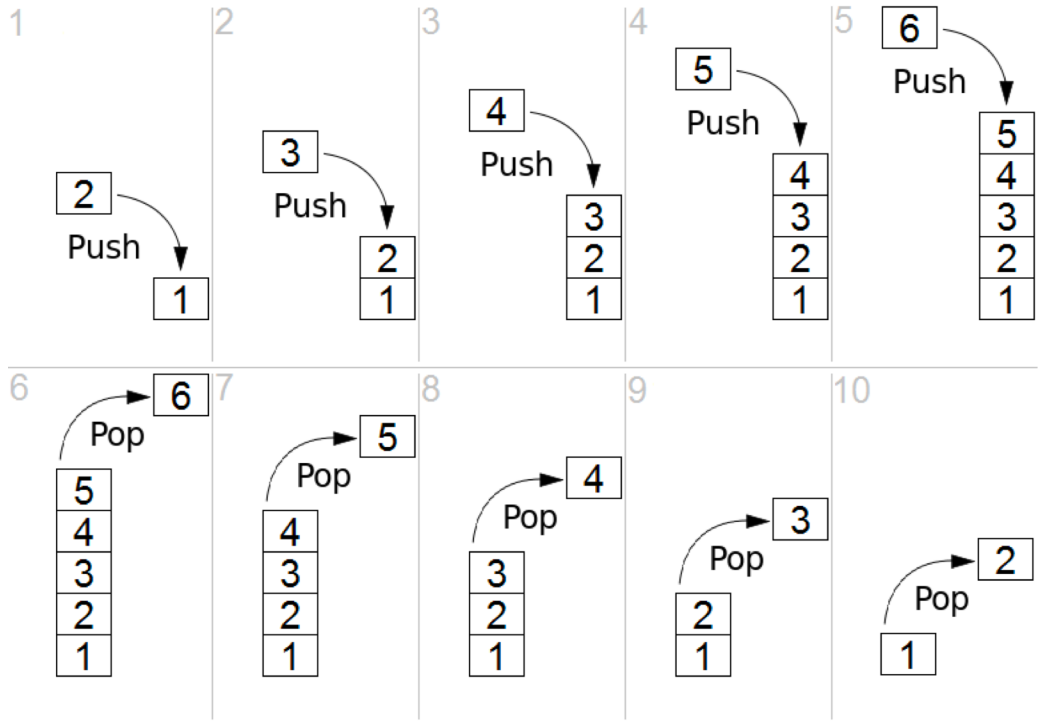
Yığıt İşlemleri

- Temel olarak iki işlem vardır
  - push (veri): Yığının en üstüne yeni veri ekler.
  - pop(): Yığının en üstünden veri çeker (Aynı zamanda siler).

- Diğer İşlemler
  - top() : Yığının en üstündeki elemanı çeker (Silmez!).
  - size(): Yığının eleman sayısı.
  - isempty?: Yığın dolumu? true/false

Stack İşlemleri

- Ekleme/çıkarma



https://upload.wikimedia.org/wikipedia/commons/b/b4/Lifo_stack.png

Stack

- Kullanım Alanları
  - Paket programlarda undo/redo işlemleri
  - Tarayıcılarda back işlemi

Stack İşlemleri

- N eleman için, Her bir işlem O(1)
- Yığınların eleman sayısı baştan tanımlanmalıdır.
- Dolu yığına eleman eklemeye çalışmak, veya
- Boş yığında eleman çekmeye çalışmak *istisna* ya neden olabilir.

Stack

- Bazı İşlemler

```
x = Stack()
x.push(5)
print(x.pop())
x.push(7)
print(x.size())
print(x.is_empty())
```

- Python kodu

```
class Stack(object):

    def __init__(self, limit=10):
        self.stack = []
        self.limit = limit


    def push(self, data):
        if len(self.stack) >= self.limit:
            raise StackOverflowError
        self.stack.append(data)

    def pop(self):
        if self.stack:
            return self.stack.pop()
        else:
            print('Stack boş')

    def is_empty(self):
        return not bool(self.stack)

    def size(self):
        return len(self.stack)
```

Stack

- Stack sınıfı tanımlama ve yapılandırıcı metot

```
class Stack(object):

    def __init__(self, limit=10):
        self.stack = []
        self.limit = limit
```

Stack

- Eleman ekleme

```
def push(self, data):
    if len(self.stack) >= self.limit:
        raise StackOverflowError
    self.stack.append(data)
```

Stack

- Eleman çekme

```
def pop(self):
    if self.stack:
        return self.stack.pop()
    else:
        print('Stack boş')
```

Stack

- Eleman çekme (top işlemi)

```
def top(self):
    if self.stack:
        return self.stack[-1]
    else:
        print('Stack boş')
```

Stack

- Stack boşmu?

```
def is_empty(self):
    return not bool(self.stack)
```

Stack

- Eleman sayısı

```
def size(self):
    return len(self.stack)
```

Python dilinde kullanılabilecek hazır yapılar

- Python dilinde *list()* veri tipi, boyutu dinamik dizi ve/veya yığın olarak kullanılabilir

```
x=list()            #Boş liste
x.append(1)         #Sonuna eleman ekle
x.append(3)
x.append(5)

print(len(x))       #Listenin uzunluğu

print(x.index(3))   #Değeri 3 olan elemanın yerini dönder

x.insert(1,2)       #Araya eleman ekle
x.insert(3,4)

print(x[-1])        #En üstteki veriyi seç, silmeden!

x.pop()             #En üstteki veriyi seç, silerek!

x.sort()            #Sırala
x.reverse()         #Ters sırala
```