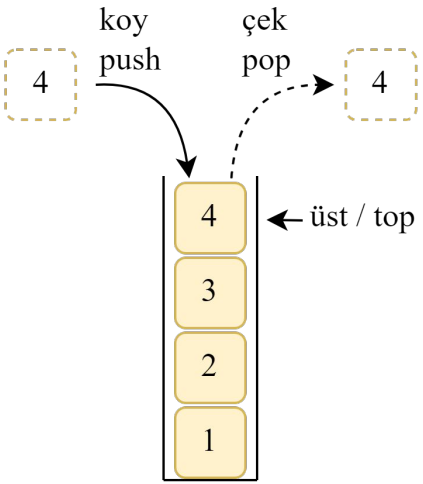
Yığın (Stack) Veri Yapısı

Dr. Hakan Temiz

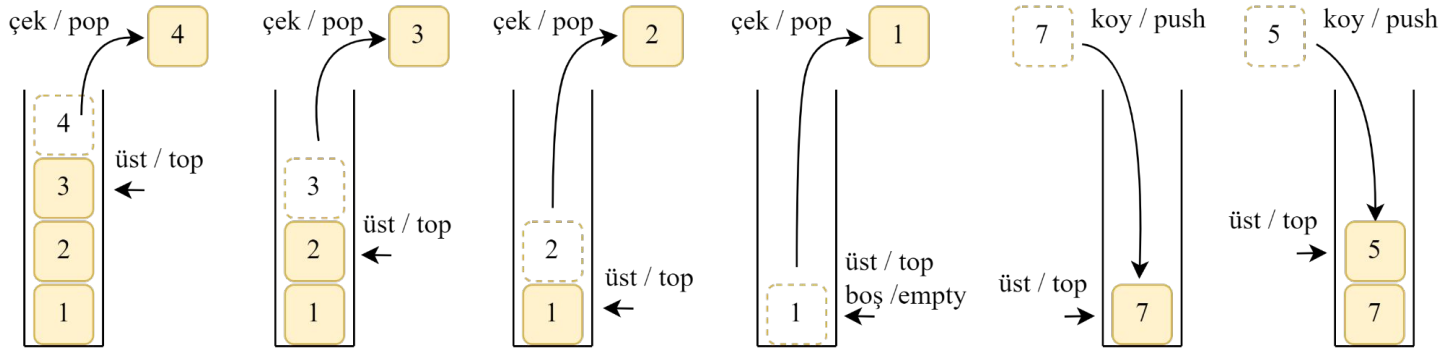
Yığın

- Yığın veri yapısı, veriyi, üst üste konan tabaklara benzer bir yapıda saklar.
- Tabak örneği ele alınırsa, bir tabak gerektiğinde, en üstteki tabak alınır. Tabaklar yığınına yeni bir tabak konduğunda en üste konur. Bir sonraki de, son konan tabağın üzerine konur. Tabak alınmak istendiğinde de en üstteki alınır.
- Buna örneğe benzer şekilde, yığın veri yapısında yeni bir eleman daima koleksiyonun en sonuna eklenir. Çıkarma işleminde de, yine, en sondaki (üstteki) eleman çıkarılır.
- Bu şekilde **Son Giren İlk Çıkar (Last in First out - LIFO)** prensibiyle çalışan doğrusal veri yapısına **Yığın (Stack)** denir.



Yığın

- Yığında, genellikle, son elemanın konum bilgisi tutulur (örnekte, üst/top).
- Ekleme işlemine **koy/it (push)** ve çıkarma işlemine **çek (pop)** adı da verilir. Ekleme ve çıkarma işlemlerine uygun olarak konum bilgisi de her seferinde güncellenir.
- **pop** işleminde, eleman yapıdan çıkarılır ve aynı zamanda değeri döndürülür/verilir. pop işlemine ilave olarak, **peek (gözet)** şeklinde bir işlev de bulunabilir. peek'in pop 'dan farkı, peek'in üstteki (**top**) elemanı yapıdan çıkarmadan göstermesi; pop'ın ise, elemanı göstermekle (döndürmekle) beraber, yapıdan çıkarmasıdır.



Yığın - Kullanım Örnekleri

Yığın kullanımına örnek olarak şunlar gösterilebilir:

- Programlama dillerinde, fonksiyon çağrıları sırasında dönüş adresinin takibi
- Programlama dillerinde açılan parantezlerin kapanışlarının takibinde.
- İnternet Web tarayıcılarında son ziyaret edilen sitelerin adreslerinin depolanmasında. Ziyaret edilen sitelerin adresleri adres yığınına "itilir". Kullanıcı "geri" düğmesine tıkladığında en sondan geriye doğru gezilen sayfalar getirilir.
- Çoğu uygulamada (programda) son düzenleme işlemini iptal edip, işlemden önceki duruma geri dönmeye yarayan, "geri alma" mekanizması değişikliklerin bir yığında tutulmasıyla gerçekleştirilir.

Yığın - Gerçekleştirim Biçimleri

Yığın veri yapısı iki türlü gerçekleştirilebilir:

- **Sabit Boyutlu Yığın :** Adından da anlaşılacağı gibi, sabit boyutludur ve dinamik olarak büyüyüp küçülemez. Yığın doluysa ve yığına bir öğe eklenmeye çalışılırsa, taşma hatası (**overflow**); yığın boşken bir öğe kaldırılmaya çalışılırsa, alttaşma (**underflow**) hatası oluşur. Sabit boyutlu yığın, diziler üzerinde gerçekleştirilebilir.

- **Dinamik Boyutlu Yığın :** Dinamik yığın, dinamik olarak büyüyebilir veya küçülebilir. Yığın dolduğunda, yeni öğeyi için otomatik olarak boyutu artırılır ve yığın boş olduğunda boyutu azaltılır. Dinamik yığın, kolayca yeniden boyutlandırılmaya izin verdiği için bağlı liste kullanılarak uygulanır.

Yığın - İşlemler

Tipik bir yığında şu işlemler bulunur:

- **push()** yığına öğe ekler. Türkçe, koy() veya ekle() isimleriyle anılır.
- **pop()** yığından son öğeyi çıkarır. Türkçe, çek() adı verilebilir.
- **top()** en üstteki öğeyi verir. Kimi yerde peek() adına da rastlanır. Türkçe, üst() veya üstteki() adları verilebilir.
- **isEmpty()** yığın boş ise True, aksi durumda False döndürür. Türkçe, bosMu() adı verilebilir.
- **isFull()** yığın dolu ise True, aksi durumda False döndürür. Türkçe, doluMu() adı verilebilir.
- **len()** yığının eleman sayısını verir. Türkçe, uzunluk() adı verilebilir.

Yığın - İşlemler - push()

push() işlevi, yığına öğe eklemeyi sağlar. Yığın dolu ise **Yığın Taşması (stack overflow)** hatası alınır.

Algoritma

- Yığına eleman itmeden (koymadan) önce yığının dolu olup olmadığı kontrol edilir.
- Yığın doluysa (yani, top = = kapasite - 1) , Yığın Taşması (**overflow**) meydana gelir; eleman eklenemez.
- Aksi durumda, top += 1 olur ve yeni değer en üst konuma (top) eklenir.
- Kapasite dolana kadar yığına eleman eklenebilir.

Yığın - İşlemler - pop()

pop() işlemi, yığından bir öğe çıkarır. Öğeler, kondukları (itildikleri) sıranın tersine doğru çıkarılır. Yığın boşsa, buna **Alt Taşma (Overflow)** durumu denir.

Algoritma

- Elemanı yığından çıkarmadan önce yığının boş olup olmadığı kontrol edilir.
- Yığın boşsa (top = = -1) **Yığın Alt Taşması (Underflow)** meydana gelir ve yığından hiçbir eleman kaldırılamaz.
- Aksi takdirde, en üstteki öğe yığından alınır; top - = 1 olur ve en üst öğe döndürülür (return).

Yığın - İşlemler - peek() veya top()

top() veya **peek()** işlevi, yığının en üstündeki öğeyi verir.

Algoritma

- Yığının boş olup olmadığı kontrol edilir.
- Yığın boşsa (top = = -1), işlev “Yığın boş” yazar ve/veya None döndürür.
- Aksi takdirde, işlev, top 'ın işaret ettiği konumundaki öğeyi döndürür.

Yığın - İşlemler - isEmpty()

Yığının boş olup olmadığını kontrol etmeyi sağlayan işlevdir. Yığın boş ise **True**; değilse **False** değeri üretir.

Algoritma

- Yığının en üst konumunu veren top değeri incelenir.
- top = = -1 ise yığın boştur, dolayısıyla **True** değeri döndürülür.
- Aksi halde yığın boş değildir ve dolayısıyla **False** değeri döndürülür.

Yığın - İşlemler - isFull()

Yığın doluysa **True**; aksi durumda **False** değeri üretir.

Algoritma

- Yığının top değeri kontrol edilir.
- top = = kapasite -1 ise, yığın doludur ve bu nedenle **True** üretilir.
- Aksi halde, yığın dolu değildir ve **False** üretilir.

Yığın - İşlemler - len()

len() işlevi, yığının boyutunu veya uzunluğunu verir

Algoritma

- Dizi indeksi 0 'dan başladığı için, (top + 1) değeri öğe sayısını verir. Bu nedenle, (top + 1) değeri döndürülür.

Yığın - Karmaşıklık

İşlem	Zaman Karmaşıklığı	Alan Karmaşıklığı
push()	O(1)	O(1)
pop()	O(1)	O(1)
top() veya peek()	O(1)	O(1)
isEmpty()	O(1)	O(1)
isFull()	O(1)	O(1)
len()	O(1)	O(1)

Yığın - Avantajlar

- **Basitlik**: Yığınlar, basit ve anlaşılması kolay bir veri yapısıdır.
- **Verimlilik**: Yığına ekleme ve çıkarma işlemleri O(1) karmaşıklığa sahip olduğundan, verimlidir.
- **Düşük Bellek Kullanımı**: Genellikle, sadece içerdikleri öğeleri depolamak için bellek kullanmaları, yığınları, bellek açısından verimli kılar.
- **Son Giren İlk Çıkar (LIFO)**: LIFO davranışı, fonksiyon çağrıları, işlem veya düzenlemeleri ileri-geri alma gerektiren durumlar, programlama dili grameri ve ifade değerlendirmesi gibi birçok senaryoda faydalıdır.

Yığın - Dezavantajlar

- **Sınırlı erişim**: Yığındaki daima en üstteki öğelere erişilebilir; üsttekiler işlenmeden (kaldırılmadan) alttaki elemanlara erişilemez. Dizilerdeki gibi rastgele erişim mümkün değildir.
- **Taşma olasılığı**: Sınırlı bir bellek alanında gerçekleştirilen bir yığına, kapasitesinden fazla öğe yüklenmesi taşma hatasına ve veri kaybına neden olur.

Yığın - Dizi Üzerinde Yığın Gerçekleştirimi

```
class DizideYigin:
  """Dizi üzerinde Yığın yapısını uygular"""

  def __init__(self, kapasite):
    self.kapasite = kapasite
    self.top = -1 # yığında başlangıçta eleman yok.
    self.yigin = [None] * self.kapasite # yığının tutulduğu dizi


# ...

# ?
```

Yığın - Dizi Üzerinde Yığın Gerçekleştirimi

```
class DizideYigin:# Dizi üzerinde Yığın yapısı

  def __init__(self, kapasite):
    self.kapasite = kapasite
    self.top = -1
    self.yigin = [None] * self.kapasite # dizi

  def len(self, ):
    return self.top + 1

  def isEmpty(self, ):
    return self.top < 0

  def isFull(self, ):
    return self.top >= self.kapasite -1

  def peek(self, ):
    if not self.isEmpty():
      return self.yigin[self.top]
    else:
      return None
```

```
  def push(self, veri):
    if self.isFull():
      print("Yığın Taşma Hatası. Yığın dolu. Eleman eklenemez!")
      return False # ekleme başarısız manasında
    else:
      self.top += 1
      self.yigin[self.top] = veri
      print(f"{veri} yığına eklendi.")

      return True # eklemenin sorunsuz yapıldığı anlamında

  def pop(self, ):
    if self.isEmpty():
      print("Yığın Alt Taşma Hatası. Yığın boş. Eleman çıkarılamaz!")
      return None #
    else:
      oge = self.yigin[self.top]
      self.yigin[self.top] = None # None: boş demek
      self.top -= 1
      print(f"{oge} yığından çıkarıldı.")
      return oge  # öğeyi döndür
```

Yığın - Dizi Üzerinde Yığın Gerçekleştirimi

```
yigin = DizideYigin(3) # Uzunluğu 3 olan bir dizide yığın
yigin.push("A")
yigin.push("B")
yigin.push("C")

yigin.pop()
print (f"En üstteki öğe: {yigin.peek()}")
```

```
A yığına eklendi.

B yığına eklendi.

C yığına eklendi.

C yığından çıkarıldı.

En üstteki öğe: B
```

Yığın - Dizi Üzerinde Yığın Gerçekleştirimi

```
yigin.kapasite # Yığının toplam boyu nedir?
3


yigin.len() # Yığının uzunluğu (öğe sayısı)

2

yigin.push("C")
C yığına eklendi.


yigin.push("D") # yığın dolu olduğundan taşma hatası (Overflow) meydana gelir.
Yığın Taşma Hatası! Yığın dolu! Eleman eklenemez!


bos_yigin = DizideYigin(100) # Uzunluğu 100 olan bir dizide yığın
bos_yigin.pop()
Yığın AltTaşma Hatası! Yığın boş! Eleman çıkarılamaz!
```

Yığın - Bağlı Liste ile Yığın Gerçekleştirimi

```
# Yığın (Stack) Veri Yapısı Sınıfı
class Yigin:
  """Yığın "Stack" Veri Yapısını Uygular"""

  class YiginDugumu:
    """Yığın için bağlı liste düğümü"""
    # Sınıfın inşaa metodu
    def __init__(self, veri):
        self.veri = veri
        self.sonraki = None



# ...

# ?
```

Yığın - Bağlı Liste ile Yığın Gerçekleştirimi

```
class Yigin:
  """Yığın "Stack" Veri Yapısını Uygular"""

  class YiginDugumu:
    """Yığın için bağlı liste düğümü"""
    # Sınıfın inşaa metodu
    def __init__(self, veri):
        self.veri = veri
        self.sonraki = None

  # Bağlı liste inşaası
  def __init__(self):
      self.kok = None

  def isEmpty(self): # Stack boş ise True,
değilse False
      return True if self.kok is None else False
```

```
  def push(self, veri): # yığına öğe ekle (it)
      oge = self.YiginDugumu(veri)
      oge.sonraki = self.kok
      self.kok = oge
      print(f"{veri} yığına itildi")

  def pop(self): # yığından bir öğe çek
      if (self.isEmpty()):
          return None # öğe yok
      oge = self.kok
      self.kok = self.kok.sonraki
      cekilen = oge.veri
      return cekilen

  def peek(self): # üstteki öğeyi göster (çıkarmadan)
      if self.isEmpty():
          return None # öğe yok
      return self.kok.veri
```

Yığın - Bağlı Liste ile Yığın Gerçekleştirimi

```
# Yığını oluştur ve "A", "B", "C" ile doldur.
yigin = Yigin()
yigin.push("A")
yigin.push("B")
yigin.push("C")

print (f"{yigin.pop()} yığından çekildi")
print (f"En üstteki öğe: {yigin.pop()}")
```

```
A yığına itildi

B yığına itildi

C yığına itildi

C yığından çekildi

En üstteki öğe: B
```