Algorithm Analysis And Design

With Infinity Teams

Eng: Abdulrahman Hamdi

INFINITY TEAM
Since 2020

Table of contents

Eng: Abdulrahman Hamdi

01

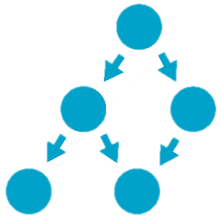
Basic Data Structures

Eng: Abdulrahman Hamdi

Veri Yapılarına Giriş

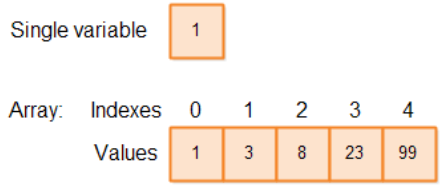
Veri yapıları, verilerin bilgisayarınızın belleğinde (veya
diskinde) nasıl düzenleneceği ile ilgi yöntemlerdir.
•**Algoritmalar**, bir programın bu yapılardaki verileri
işlemek için kullandığı prosedürlerdir.
•Hemen her bilgisayar programı, en basiti dahi olsa,
veri yapılarını kullanır.

•Örnek: Adres etiketlerini yazdıran bir program.
Program, yazılacak adresleri bir dizinin içinde tutabilir
ve basit bir for döngüsü ile dizideki her adresi
yazdırabilirsiniz.



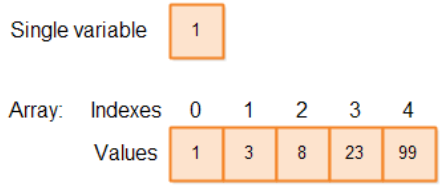
Eng: Abdulrahman Hamdi

Arrays(Dizi)



- En temel veri yapılarından biridir.
- Dizi elemanlarına erişmek üzere genellikle sıfır-tabanlı indisleme
- (zero-based indexing) kullanılır.
- Tek boyutlu (single-dimension) ya da çok-boyutlu (multidimension) olabilir.
- Düzenli (regular) ya da düzensiz (jagged) olarak tanımlanabilir

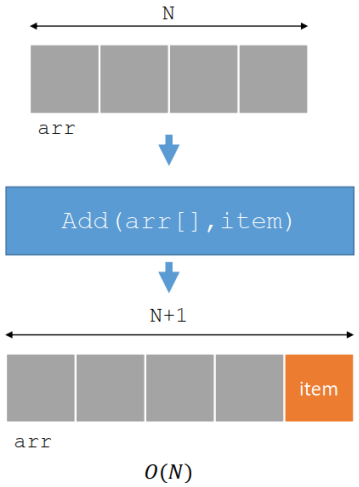
Arrays(Dizi)



- En büyük dezavantajı sabit boyutlu olmasıdır.
- Dizinin sabit boyutlu olmasından dolayı ekleme ve silme gibi
- işlemlerin maliyeti artar.
- En büyük avantajı ise bellek gözlerine doğrudan erişimin
- olmasıdır.

Arrays(Dizi)

Add(arr[],item)



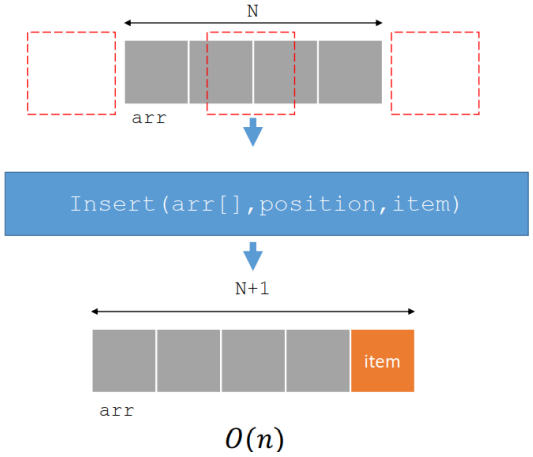

```
input
      arr[] : Array
      item  : To be added item
output
      arr[] : New Array including new item

temp = [];
for i = 1 to N
      temp[i] = arr[i];
end
temp[array.Length] = item;
arr = temp;
```

Eng: Abdulrahman Hamdi

Arrays(Dizi)

Insert(arr[], position, item)



$O(n)$

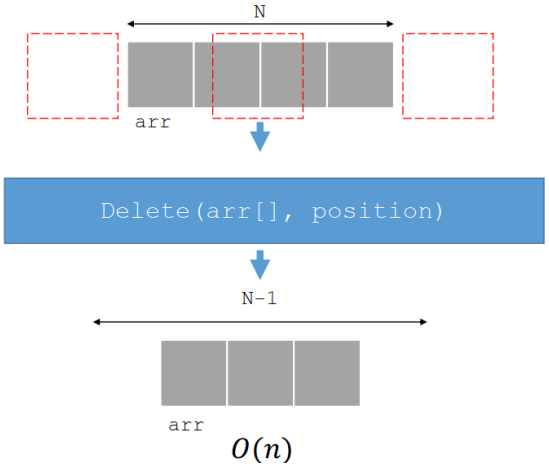
```
input
      arr[]       : Array
      position    : Position on the Array.
      item        : To be added item
output
      arr[]       : New Array including new item

temp = [];
for i = 1 to N
      if (i<position)
            temp[i] = arr[i]
      else
            temp[i+1] = arr[i];
      endif
endfor
temp[position] = item;
arr = temp;
```

Eng: Abdulrahman Hamdi

Arrays(Dizi)

RemoveAt(arr[], position)



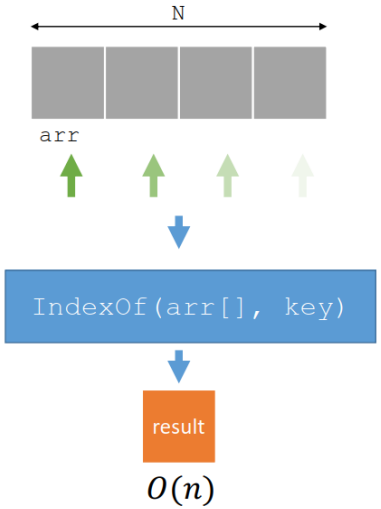

```
input
      arr[]       : Array
      position    : Position on the Array.
output
      arr[]       : New Array

temp = [];
for i = 1 to N
      if (i<position)
            temp[i] = arr[i]
      endif
      if (i>position)
            temp[i-1] = arr[i];
      endif
endfor
arr = temp;
```

Eng: Abdulrahman Hamdi

Arrays(Dizi)

IndexOf(arr[], key)



$O(n)$

```
input
      arr[]       : Array
      key         : The key element.
output
      result      : The position of the key element or -1.

temp = [];
result = -1;
for i = 1 to N
      if (key==arr[i])
            result = i;
            return result;
      endif
endfor
return result;
```

Eng: Abdulrahman Hamdi

Yığın (Stack)

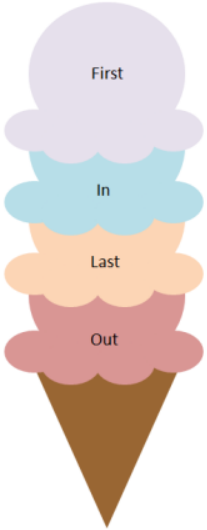
- Veriler doğal olarak listeler halinde düzenlenir.
- Array ve ArrayList yapısı verileri liste düzeninde organize etmek
- üzere kullanılan yapılar arasındadır.
- Anlaşılması kolay soyutlamalar sağlayan liste yapılarından biri
- yığınlardır.

Eng: Abdulrahman Hamdi

Yığın (Stack)

- Yığın yapısında listeye ekleme işlemi sona yapılır ve listeden çıkarma işlemi de yine son eleman dikkate alınarak gerçekleştirilir.

- Yığınlar ifadelerin değerlendirilmelerinden, işlev çağrılarına kadar bilgisayar bilimlerde yaygın bir şekilde kullanılır.



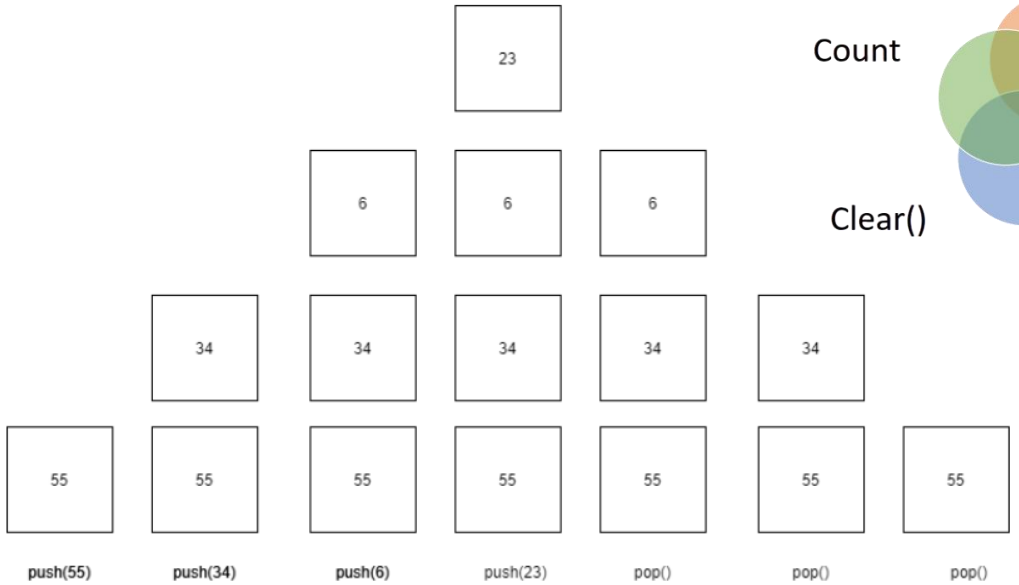
Eng: Abdulrahman Hamdi

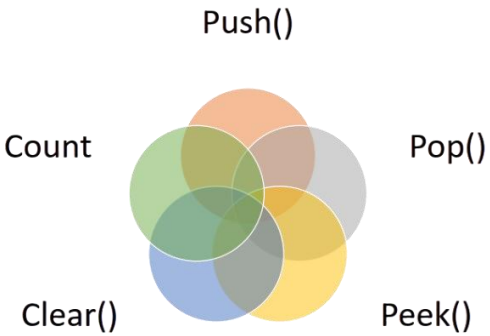
Yığın (Stack)

- Yığınların sadece son elemanlarına erişim sağlanabilir.
- Bu nedenle son-giren ilk-çıkar (last-in first-out, **LIFO**) veri yapısı olarak tanımlanır.
- Java dilinde Stack veri yapısı hem object hem de Generic olarak koleksiyonlar kapsamında sunulmaktadır.

Eng: Abdulrahman Hamdi

Yığın (Stack)





Eng: Abdulrahman Hamdi

Abstract Data Type

Ana işlevler

- `void push()`
- `int pop()`

Yardımcı işlevler

- `int top()`
- `int size()`
- `int isEmpty()`
- `int isFull()`

Yığın (Stack)

- N elemana sahip olan bir yığın için:

Space complexity (for n push operations)	$O(n)$
Time complexity of createStack()	$O(1)$
Time complexity of push()	$O(1)$ (Average)
Time complexity of pop()	$O(1)$
Time complexity of top()	$O(1)$
Time complexity of isEmpty()	$O(1)$
Time complexity of deleteStack()	$O(n)$

Yığın (Stack) Kabakod(Pseudocode)

```
Push(S,x)
   if Stack-Full(S)
   then error "overflow"
   else top(S) = top(S) +
        S[top(S)] = x
```

```
Pop(S)
   if Stack-Empty(S)
   then error "underflow"
   else top(S) = top(S) - 1
        return S[top(S) + 1]
```

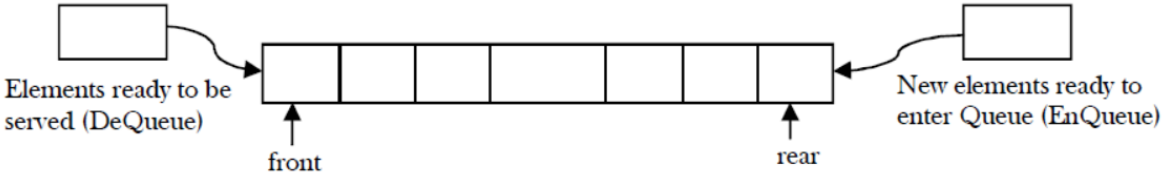
```
Stack-Empty(S)
   if top(S) = 0
   then return True
   else return False
```

```
Stack-Full(S)
   if top(S) = length(S)
   then return True
   else return False
```

Eng: Abdulrahman Hamdi

Kuyruk (Queue)

- Bilgisayar bilimleri açısından liste tabanlı veri yapıları arasında yaygın bir şekilde kullanılan veri yapılarından biri de kuyruklar.
- İlk-giren ilk-çıkar (first-in first-out, FIFO) çalışma ilkesine göre kuyruk işletilir.



Kuyruk Soyut Veri Türü

Ana işlevler

- void enQueue
- T deQueue

Yardımcı işlevler

- front()
- rear()
- size()
- isEmpty()

Eng: Abdulrahman Hamdi

Kuyruk Uygulamaları

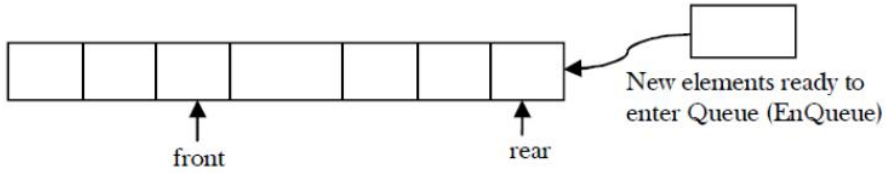
Doğrudan Uygulamalar

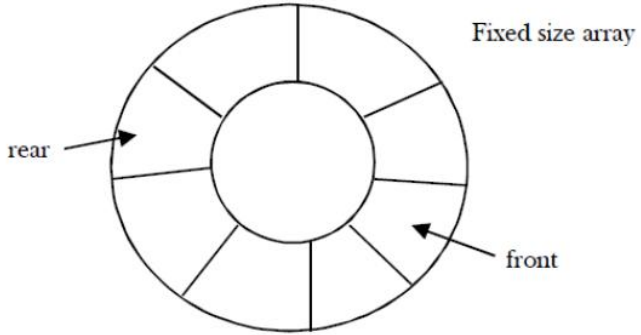
- İşletim sistemler iş planlaması (Yazıcı kuyrukları)
- Gerçek hayattaki kuyruk uygulamalarının modellenmesi
- Çoklu programlama
- Asenkron veri transferi

Yardımcı işlevler

- Algoritmalar için yardımcı veri türü
- Diğer veri yapılarının bileşenleri

Çevrimsel kuyruk





Eng: Abdulrahman Hamdi

Kuyruk (Queue)

- N boyutlu bir kuyruk için:

Space Complexity (for n enQueue operations)	$O(n)$
Time Complexity of enQueue()	$O(1)$
Time Complexity of deQueue()	$O(1)$
Time Complexity of isEmpty()	$O(1)$
Time Complexity of isFull ()	$O(1)$
Time Complexity of size()	$O(1)$

Kuyruk Uygulama(Queue Implementation)

- Basit dairesel dizi tabanlı uygulama
- Dinamik dairesel dizi tabanlı uygulama
- Bağlı liste tabanlı uygulama

Eng: Abdulrahman Hamdi

Kuyruk (Queue)Kabakod(Pseudocode)

Pseudecode - isFull

```
begin procedure isfull

   if rear equals to MAXSIZE
      return true
   else
      return false
   endif

end procedure
```

Pseudecode - Enqueue

```
int enqueue(int data)
   if(isfull())
      return 0;

   rear = rear + 1;
   queue[rear] = data;

   return 1;
end procedure
```

Pseudecode - Dequeue

```
procedure dequeue

   if queue is empty
      return underflow
   end if

   data = queue[front]
   front ← front + 1
   return true

end procedure
```

Eng: Abdulrahman Hamdi

Algorithm Analysis And Design

Thanks for listening!

Eng: Abdulrahman Hamdi