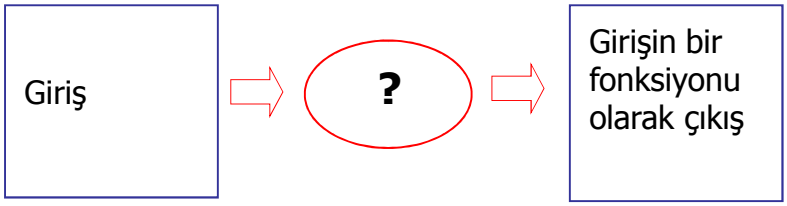
BM 210 Algoritma Analizi (Analysis of Algorithms)

Hazırlayan: M.Ali Akcayol
Gazi Üniversitesi
Bilgisayar Mühendisliği Bölümü

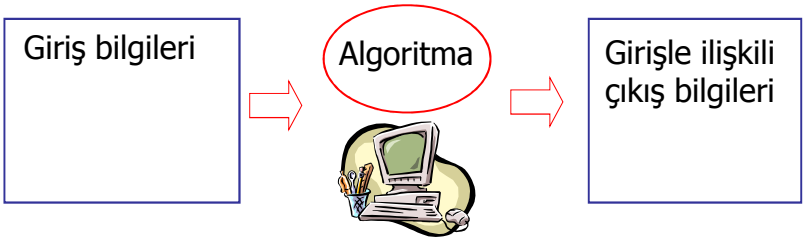
Algoritma

- Algoritma
  - Bir problemin çözümünü belirli bir zamanda çözmek için sonlu sayıdaki adım-adım birbirini takip eden işlemlerdir.
- Program
  - Bir programlama dilinde bir algoritmanın gerçekleştirilmesidir.
- Veri Yapısı
  - Problemin çözümü için gereken data'nın organize edilmesidir.

Algoritmik problem



Algoritmik çözüm



- Algoritma giriş örnekleri üzerindeki işlemleri tanımlar
- Aynı algoritmik problem için farklı algoritmalar olabilir

Algoritmalar

- Algoritma bir problemin nasıl çözüleceğinin kesin tanımlamasıdır
- Algoritma her adımı komple tanımlamalıdır
- Bir algoritma problemin olası tüm girişleri için çalışmak zorundadır
- Algoritmalarda şu özellikler olmalıdır;
  - Correct: Her giriş için uygun bir çıkış üretmelidir
  - Efficient: Olabildiğince hızlı çalışmalıdır, olabildiğince az hafıza kullanmalıdır

Örnek 1: Arama (Searching)

INPUT

• n (n >0) adet azalmayan sırada sayı
• bir sayı (sorgu-query)

$a_1, a_2, a_3, \ldots, a_n;\ q$

2 5 4 10 11; **5**

2 5 4 10 11; **9**

OUTPUT

• bulunan sayının sıra numarası veya yoksa NIL değeri

j

2

NIL

Arama (2)

INPUT: A[1..n] – an array of integers, q – an integer.
OUTPUT: an index j such that A[j] = q. *NIL*, if $\forall j$ ($1 \le j \le n$): A[j] $\neq q$

```
j←1
while j ≤ n and A[j] ≠ q
   do j++
if j ≤ n then return j
else return NIL
```

- Bu algoritma *brute-force* algoritma tasarım tekniğiyle yazılmıştır – (Girişleri sırayla tarar).
- Kod pseudocode olarak yazılmıştır ve algoritmanın giriş ve çıkışı açıkca belirtilmiştir.

Pseudo-code

- Pascal, C, Java veya diğer diller:
  - Kontrol yapıları (**if then else**, **while** ve **for** döngüleri)
  - Atama işlemi (←)
  - A dizisinde erişilen eleman: A[i]
  - Composite tip (record veya object) elemanı: A.b (veya b[A])

Örnek 2: Sıralama (Sorting)

INPUT
n adet sayı

$a_1, a_2, a_3, \ldots, a_n$

2 5 4 10 7

OUTPUT
Girişteki sayıların bir permütasyonu

$b_1, b_2, b_3, \ldots, b_n$

2 4 5 7 10

Doğruluk (Correctness)

Giriş kümesi için aşağıdaki çıkış durumunda algoritma sonlandırılır:

- $b_1 < b_2 < b_3 < \ldots < b_n$
- $b_1, b_2, b_3, \ldots, b_n$ elamanları $a_1, a_2, a_3, \ldots, a_n$ elamanlarının bir permütasyonudur

Insertion Sort

A

3	4	6	8	9	7	2	5	1

1 j→ n

← i

Strateji

- Boş bir diziyle başla
- Bir elemanı ait olduğu sıraya ekle
- Bütün elemanları sıralayana kadar devam et

INPUT: A[1..n] – an array of integers
OUTPUT: a permutation of A such that A[1]≤ A[2]≤ …≤A[n]

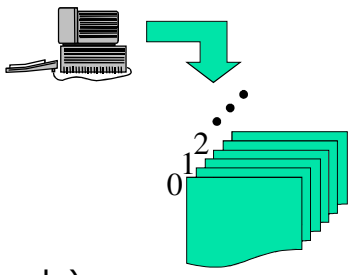
```
for j←2 to n
   do key←A[j]
      Insert A[j] into the sorted
sequence A[1..j-1]
      i←j-1
      while i>0 and A[i]>key
         do A[i+1]←A[i]
            i--
      A[i+1]←key
```

Algoritma Analizi

- Efficiency:
  - Çalışma süresi (Running time)
  - Kullanılan alan (Used Space)
- Efficiency giriş boyutunun bir fonksiyonudur:
  - Eleman sayısı (sayılar, noktalar, v.b.)
  - Bir sayıdaki bit sayısı
  - .....

RAM (Random Access Machine) modeli

- Bir CPU
- Herbiri belirli sayıda karakteri kaydedebilen sınırsız sayıda hafıza hücresi



- Instructions (herbiri sabit zaman alır)
  - Aritmetik (add, subtract, multiply, etc.)
  - Data movement (assign)
  - Control (branch, subroutine call, return)
  - Comparison
- Data types – integers, characters, and floats

Insertion Sort Analizi

- Giriş boyutunun bir fonksiyonu olarak çalışma süresinin hesabı

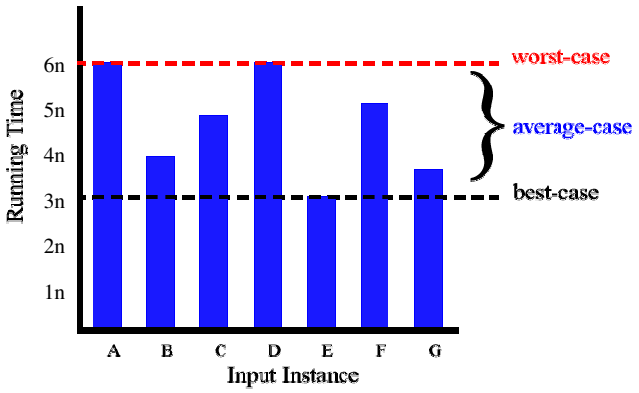
	cost	times
for j←2 **to** *n*	c_1	n
do key←A[j]	c_2	n−1
Insert A[j] into the sorted sequence A[1..j-1]	0	n−1
i←j−1	c_3	n−1
while i>0 **and** A[i]>key	c_4	$\sum_{j=2}^{n} t_j$
do A[i+1]←A[i]	c_5	$\sum_{j=2}^{n} (t_j - 1)$
i−−	c_6	$\sum_{j=2}^{n} (t_j - 1)$
A[i+1]:=key	c_7	n−1

Best/Worst/Average Case

- **Best case**: elemanlar zaten sıralıdır
- **Worst case**: elemanlar ters sıradadır
- **Average case**: elemanların yaklaşık yarısı kendi sırasındadır

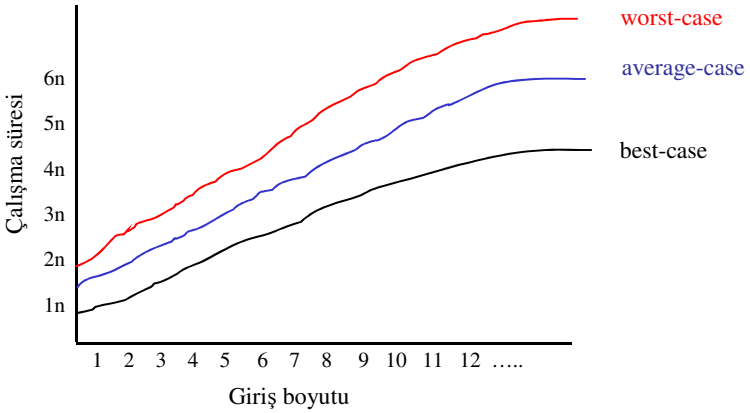
Best/Worst/Average Case (2)

- Belirli boyuttaki farklı girişler için örnek çalışma süreleri



Best/Worst/Average Case (3)

- Farklı boyuttaki girişler için:



Best/Worst/Average Case (4)

- Genellikle **Worst case** kullanılır:
  - Algoritma için garanti çalışma durumunu verir ve asla daha kötü bir durum olamaz.
  - Bazı algoritmalar için **worst case** nadiren gerçekleşir. (Örn.: Arama algoritmasında istenen elemanın veritabanında olmaması veya en sonda olması)

Ödev - Arama (Searching)

- Aşağıdaki arama algoritmasının analizini yapınız

INPUT: A[1..n] – an array of integers, q – an integer.
OUTPUT: an index j such that A[j] = q. *NIL*, if $\forall j$ ($1 \le j \le n$): A[j] $\neq q$

```
j←1
while j ≤ n and A[j] ≠ q
   do j++
if j ≤ n then return j
else return NIL
```