Temel Kavramlar

- **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

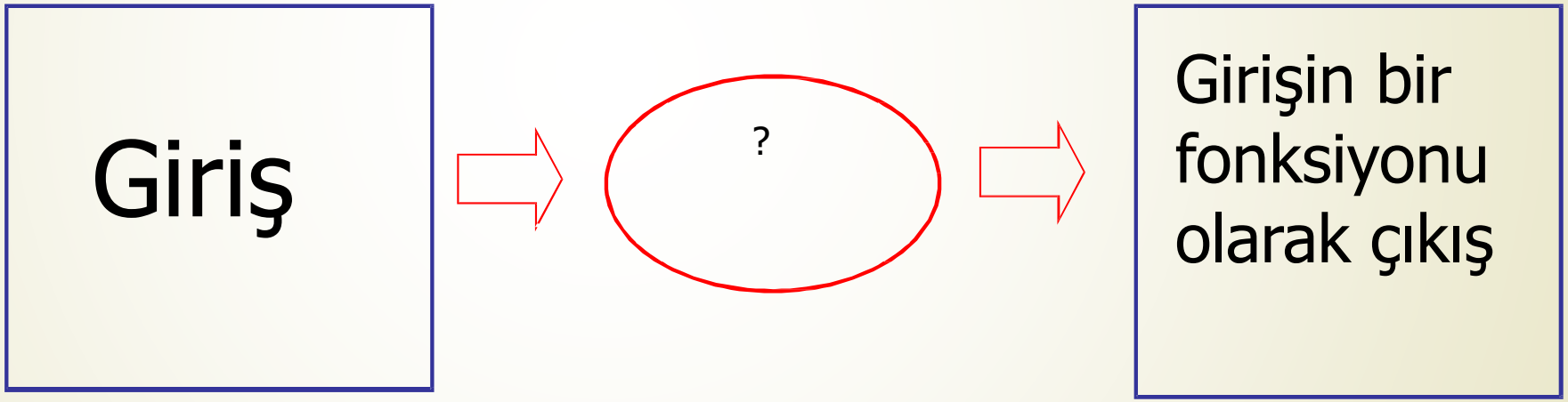
Algoritmalar

- Sıralama
- En-kısa-yol problemi
- Süreç planlama
- Benzerlik
- Paketleme vs.

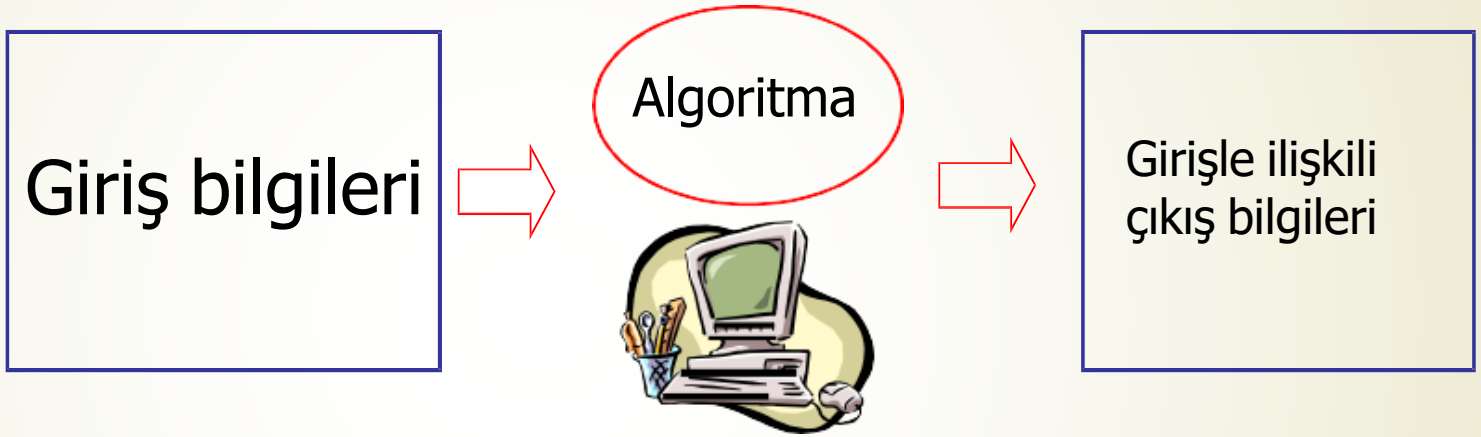
Algoritmalar

- Network alanında,
- Kriptografi alanında,
- Bilgisayar Grafiği alanında,
- Veritabanı alanında,
- Hesapsal biyoloji/geometri/işlem. Vs.

Algoritmik Problem



Algoritmik Çözüm



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

Algoritmadan Beklenenler

- 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: Olabildigince hızlı çalışmalıdır, olabildigince az hafıza kullanmalıdır.

Pseudo-code

- Pascal, C, Java veya diger 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: Arama Algoritması

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

Örnek: Arama Algoritması

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
```

Algoritma Analizi

- Algoritma Analizi bilgisayar programlarının performans ve kaynak kullanımı üzerine teorik çalışmadır.
- Algoritma analizinde öncelikle ve özellikle performans üzerinde durulur.
- Bilgisayar Programlarında işlerin nasıl daha hızlı gerçekleştirilebileceği ele alınır.

Neden Performans?

- **Bir bilgisayar programında performanstan daha önemli neler vardır?**
- Doğruluk,
- Basitlik,
- Bakım kolaylığı,
- Kaynak maliyeti,
- Sağlamlık,
- İşlevsellik/modüler yapı,
- Güvenlik,
- Kullanıcı dostluğu/kullanım kolaylığı

Neden Performans?

- Performansı iyi olan bir program kullanıcıyı mutlu eder.
- Bir işin yapılabilir olması veya olmaması konusunda performans önemli bir kriterdir.
- Örneğin, zaman kısıtlaması olursa, işiniz başarılı görülmeyebilir. Bellek kullanımı yüksekse işe yaramayabilir.

Neden Performans?

- Daha önce yapılagelen bir işi yapıyorsanız burada performansın çok da bir önemi olmayabilir.
- Ancak amacınız, daha önce yapılamayan bir iş ise, o zaman performans sizin için öncelikli maddedir.

Algoritma Analizinde Temel Kavramlar

- Yürütme zamanı,
- Zaman karmaşıklığı,
- Alan maliyeti,
- Alan karmaşıklığı,
- En iyi, ortalama, en kötü durumlar.

Yürütme Zamanı (Running Time)

- Bir programın veya algoritmanın işlevini yerine getirebilmesi için, temel kabul edilen işlemlerden kaç adet yürütülmesi gerektiğini veren bir bağıntıdır.
- Bağıntının bağımsız değişkeni genel olarak eleman sayısıdır.
  - Örneğin ***n*** elemanlı bir küme için yürütme zamanı ***T(n)*** şeklinde gösterilir.
- **Temel İşlemler:**
  - Karşılaştırma, döngü çevrimi, aritmetik işlem vs.

Zaman Karmaşıklığı (Time Complexity)

- Bir algoritmanın asimptotik notasyona göre karmaşıklık mertebesini gösteren ifadedir. Genellikle algoritmanın çok fazla eleman sayısı olması durumunda gerekli işlemlerin düzeyini gösterir.
- Zaman karmaşıklığı yürütme zamanı bağıntısı ***T(n)*** 'de olduğu gibi açık bir bağıntı olmayıp asimptotik ifadelerle gösterilir.

Alan Maliyeti (Space cost)

- Bir programın veya algoritmanın işlevini yerine getirmesi için gerekli bellek alanını veren bir bağıntı veya değerdir.
- Alan maliyeti program kodu,yığın ve veri için gerekli tüm alanları kapsar.
- Program kodu için gerekli belek alanı veri iiçn gerekli bellek alanının yanında ihmal edilecek düzeyde ise yalnızca veri alanı hesaplaması yeterlidir.
- Ör: 2 bytelık tamsayı olan ***n*** elemanlı bir küme için alan maliyeti bağıntısı ***S(n)=2n*** şeklinde verilir.

Alan Karmaşıklığı (Space Complexity)

- Alan karmaşıklığı eleman sayısı $\boldsymbol{n}$' nin çok büyük değerleri için bellek alanı gereksiniminin artış mertebesini gösteren asimptotik ifadelerdir.
- Bu amaçla zaman karmaşıklığında kullanılan asimptotik ifadelerle gösterilir.
- Ör: Alan karnaşıklığı olarak $\boldsymbol{O(n^2)}$ verilirse bellek gereksiniminin $\boldsymbol{n}$'nin büyük değerleri için en kötü durumda karesel arttığı ifade edilmektedir.

En İyi – Ortalama – En Kötü Durumlar (Best-Average-Worst Cases)

- Yürütme zamanı, maliyet ve karmaşıklık hesaplanırken en iyi sonucun elde edildiği durum **en iyi durum (best case)** dur.
- En Kötü durum (worst case) ise en olumsuz koşulların gerçekleşmesi durumunda algoritmanın çözüm üretmesi için gerekli hesaplama sonucudur.
  - Ör: Arama algoritması için en iyi durum aranan değerin dizinin ilk elemanı olması, en kötü durum ise anahtar değerin dizide bulunmamasıdır.
- Ortalama durum (Average case) ise giriş parametrelerinin en iyi durum ile en kötü durum arasında gelmesi, ortalama bir sonuç oluşturmasıdır.