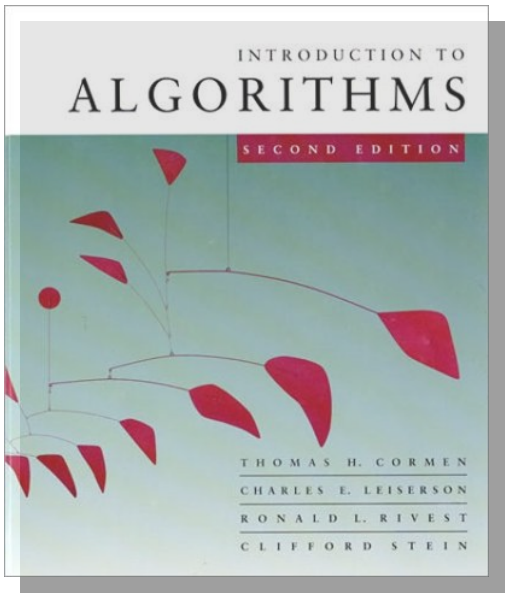
Algoritmalara Giriş
6.046J/18.401J

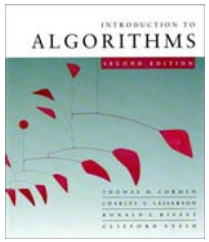


DERS 1

Algoritmaların Çözümlemesi

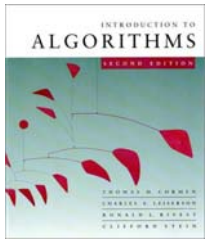
- Araya yerleştirme sıralaması
- Asimptotik çözümleme
- Birleştirme sıralaması
- Yinelemeler

Prof. Charles E. Leiserson



Dersle ilgili bilgiler

1. Öğretim kadrosu
2. Uzaktan eğitim
3. Ön koşullar
4. Dersler
5. Etütler
6. Ders notları
7. Ders kitabı
8. Dersin WEB sitesi
9. Ek destek
10. Kayıt
11. Problem setleri
12. Algoritmaları tanımlamak
13. Not verme politikası
14. Ortak çalışma politikası

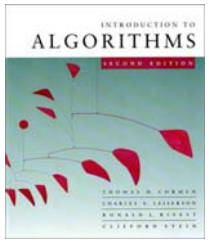


Algoritmaların çözümlemesi

Bilgisayar program başarımı ve kaynak kullanımı konusunda teorik çalışmalar

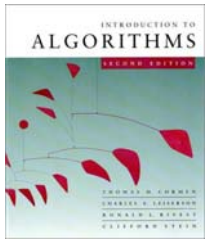
Başarımdan daha önemli ne vardır ?

- modülerlik
- doğruluk
- bakım kolaylığı
- işlevsellik
- sağlamlık
- kullanıcı dostluğu
- programcı zamanı
- basitlik
- genişletilebilirlik
- güvenilirlik



Neden algoritmalar ve başarımla uğraşırız?

- Algoritmalarla ölçeklenebilirlik anlaşılabilir.
- Başarım genelde yapılabilir olanla imkansızın arasındaki çizgiyi tanımlar.
- Algoritmik matematik program davranışlarını açıklamak için ortak **dil** oluşturur.
- Başarım bilgi işleme'nin ***para birimi***dir.
- Program başarımından alınan dersler diğer bilgi işleme kaynaklarına genellenebilir.
- Hız eğlencelidir!



Sıralama (sorting) problemi

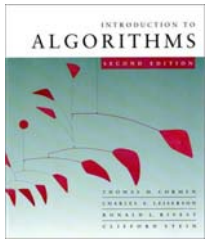
Girdi: dizi $\langle a_1, a_2, \ldots, a_n \rangle$ sayıları.

Çıktı: permütasyon $\langle a'_1, a'_2, \ldots, a'_n \rangle$ öyle ki $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

Örnek:

Girdi: 8 2 4 9 3 6

Çıktı: 2 3 4 6 8 9

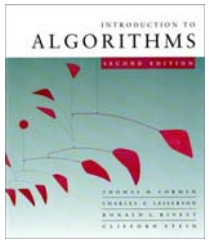


Araya yerleştirme sıralaması (Insertion sort)

"pseudocode"

(sözdekod)

```
INSERTION-SORT (A, n)      ▷ A[1 . . n]
    for j ← 2 to n
        do key ← A[ j]
           i ← j – 1
           while i > 0 and A[i] > key
               do A[i+1] ← A[i]
                  i ← i – 1
           A[i+1] = key (anahtar)
```

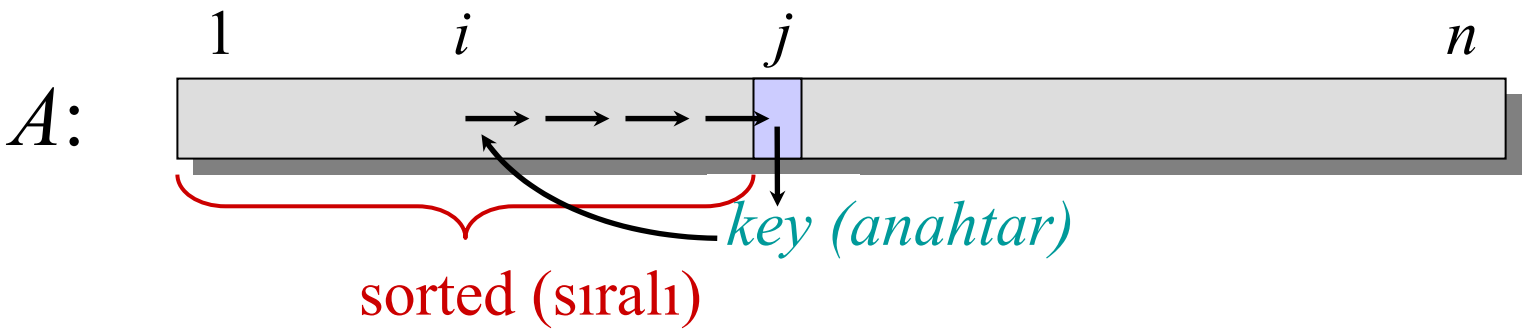


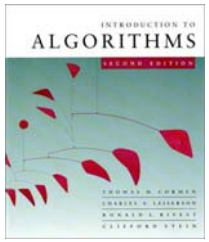
Araya yerleştirme sıralaması (Insertion sort)

"pseudocode"

(sözde kod)

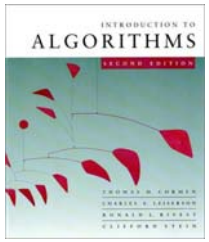
```
INSERTION-SORT (A, n)      ▷ A[1 . . n]
    for j ← 2 to n
        do key ← A[ j]
           i ← j – 1
           while i > 0 and A[i] > key
               do A[i+1] ← A[i]
                  i ← i – 1
           A[i+1] = key
```



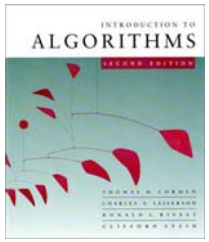


Araya yerleştirme sıralaması örneği

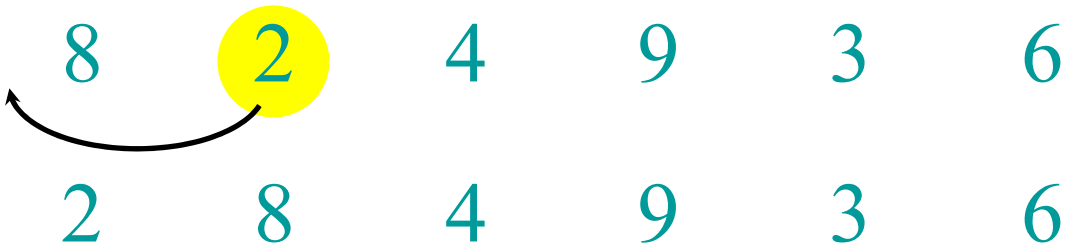
8 2 4 9 3 6

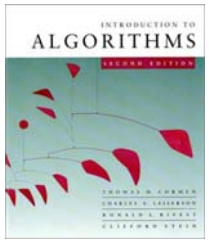


Araya yerleştirme sıralaması örneği

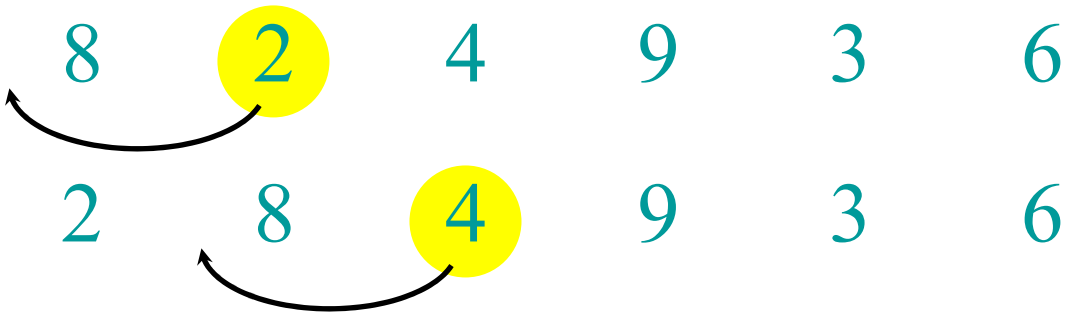


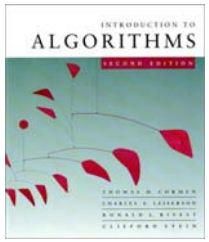
Araya yerleştirme sıralaması örneği



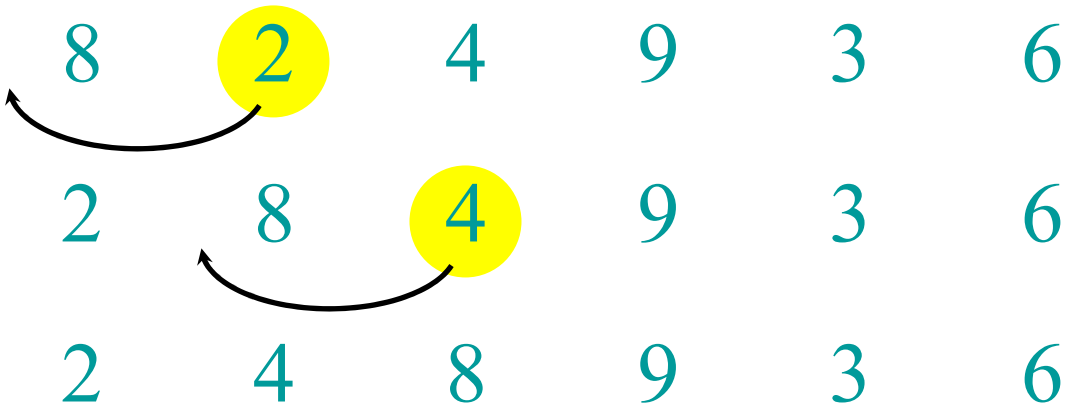


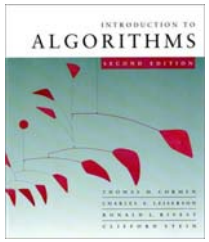
Araya yerleştirme sıralaması örneği



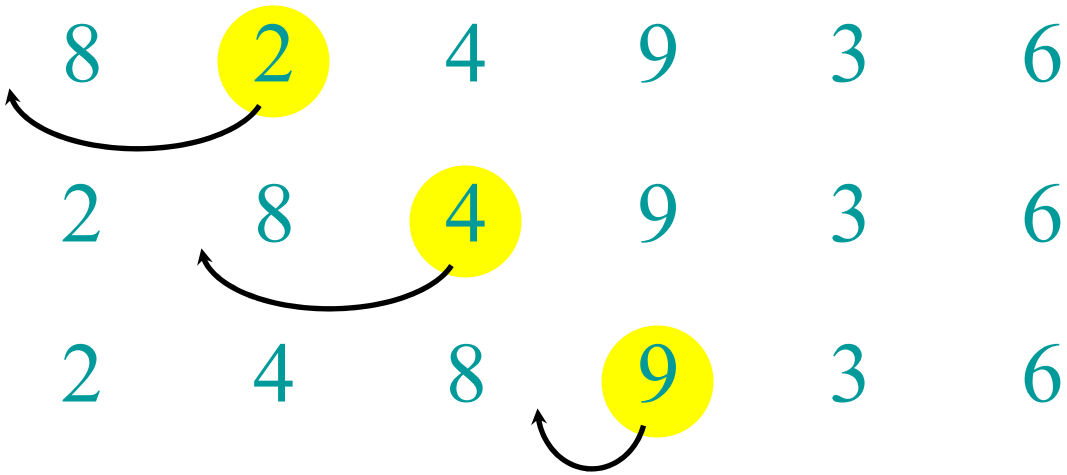


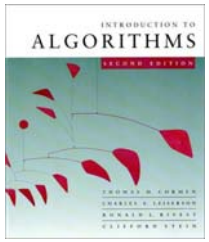
Araya yerleştirme sıralaması örneği



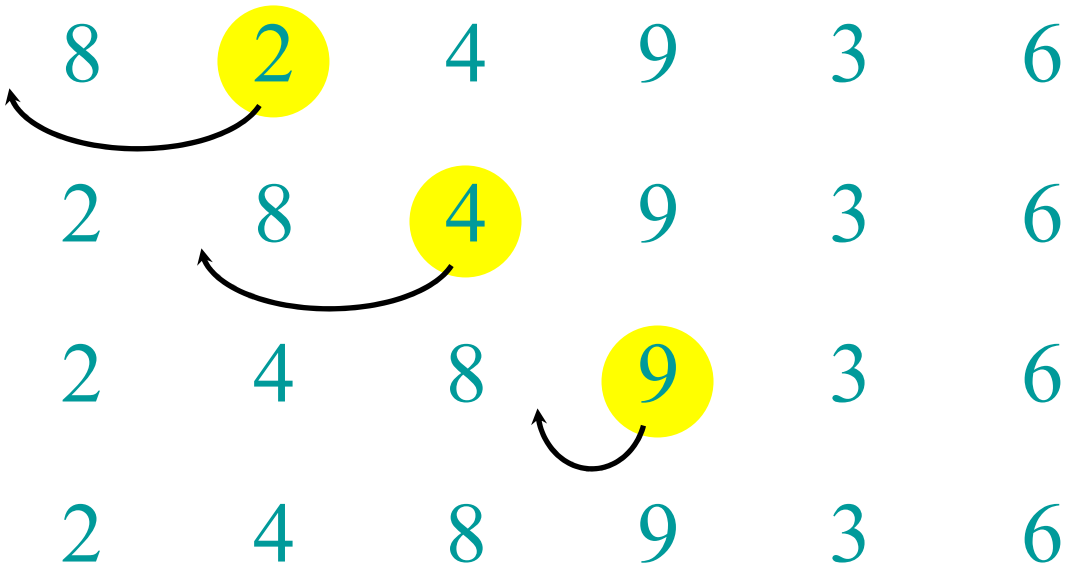


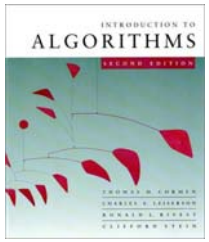
Araya yerleştirme sıralaması örneği



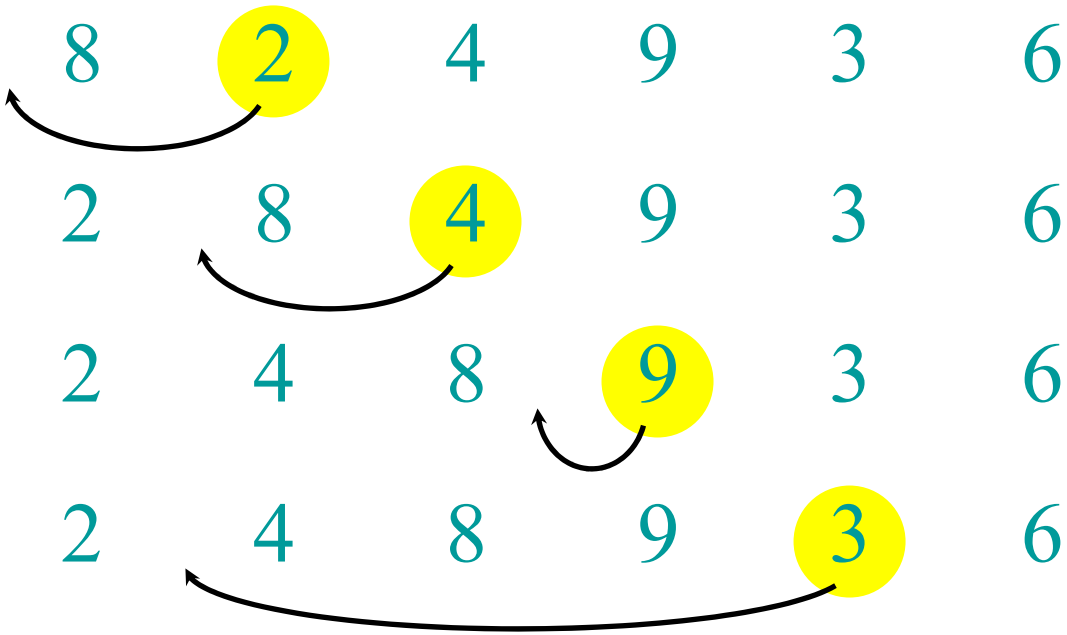


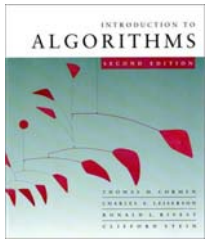
Araya yerleştirme sıralaması örneği



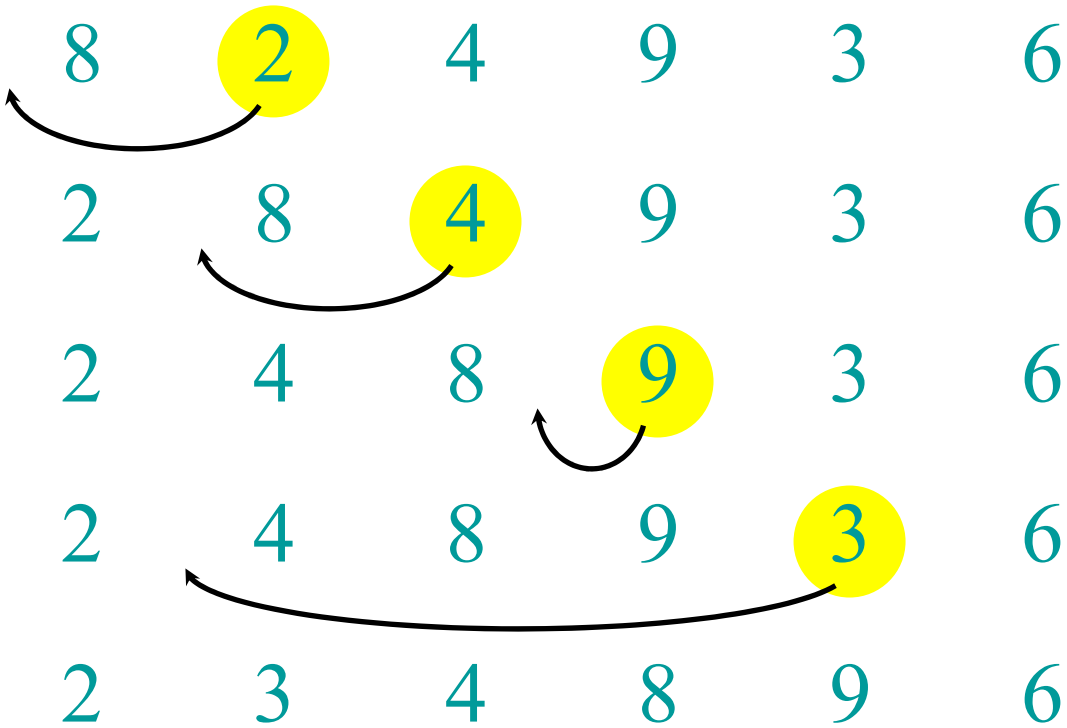


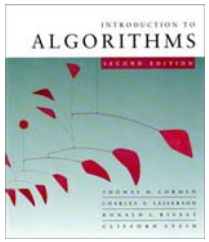
Araya yerleştirme sıralaması örneği



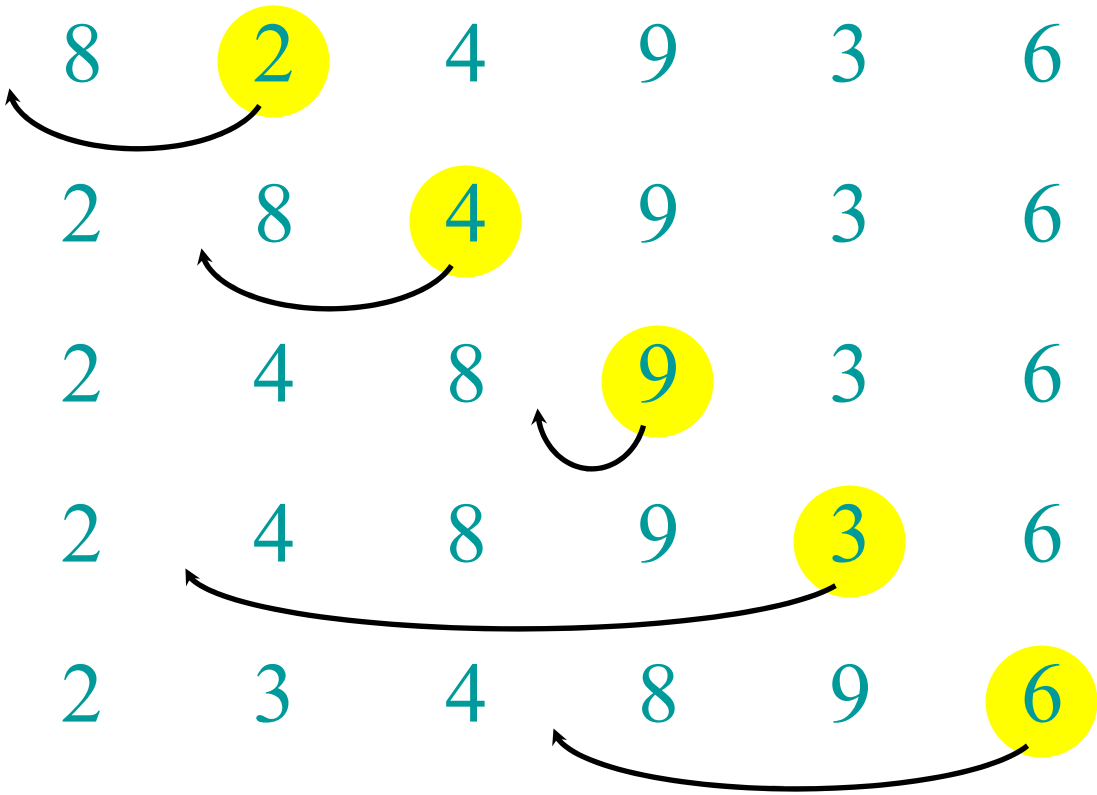


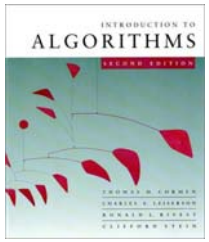
Araya yerleştirme sıralaması örneği



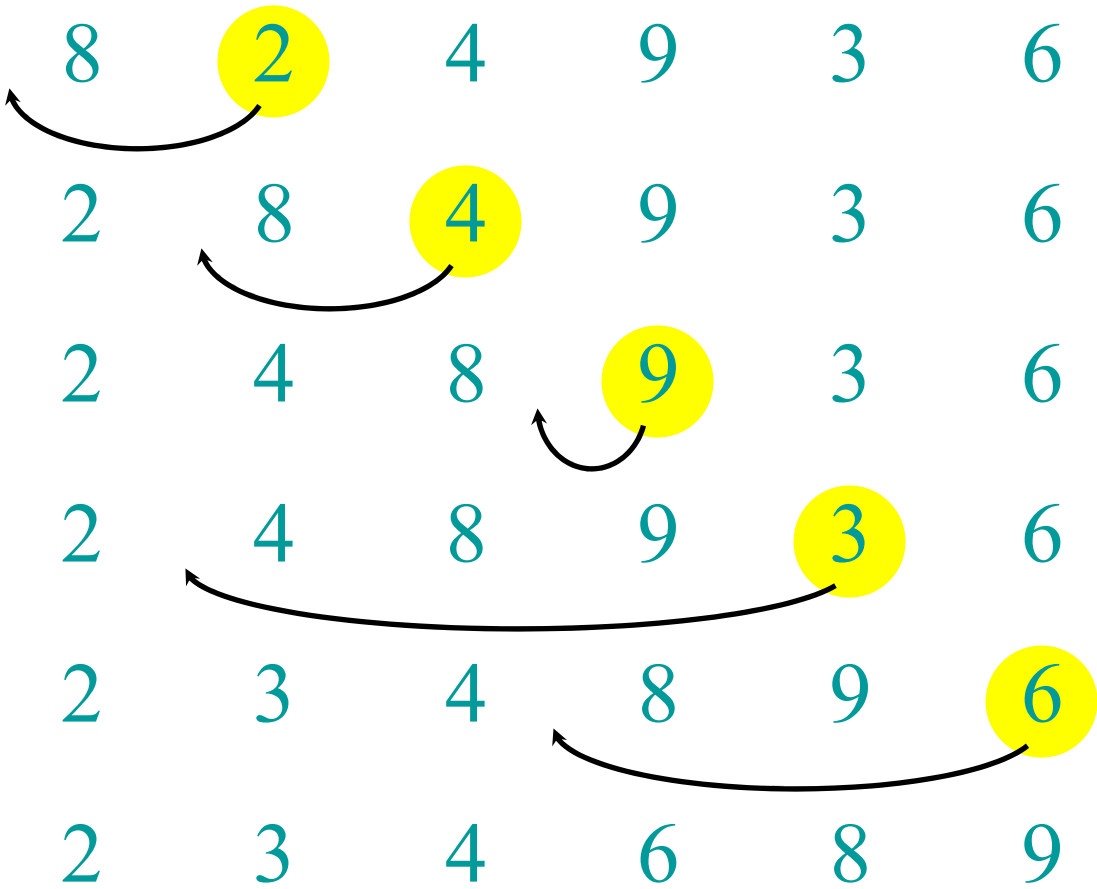


Araya yerleştirme sıralaması örneği

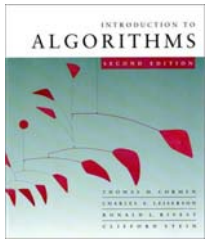




Araya yerleştirme sıralaması örneği

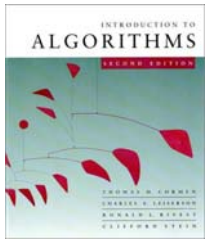


(bitti)



Koşma süresi (Running time)

- Koşma süresi girişe bağımlıdır: Önceden sıralanmış bir diziyi sıralamak daha kolaydır.
- Koşma süresinin girişin boyutuna göre parametrelenmesi yararlıdır, çünkü kısa dizileri sıralamak uzun dizilere oranla daha kolaydır.
- Genellikle, koşma süresinde üst sınırları ararız, çünkü herkes garantiden hoşlanır.



Çözümleme türleri

En kötü durum (Worst-case): (genellikle)

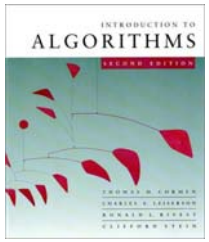
- $T(n) =$ n boyutlu bir girişte algoritmanın maksimum süresi

Ortalama durum: (bazen)

- $T(n) =$ n boyutlu her girişte algoritmanın beklenen süresi.
- Girişlerin istatistiksel dağılımı için varsayım gerekli.

En iyi durum: (gerçek dışı)

- Bir giriş yapısında hızlı çalışan yavaş bir algoritma ile hile yapmak.



Makineden-bağımsız zaman

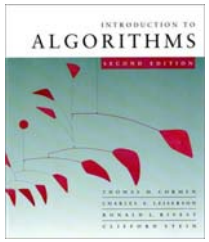
Araya yerleştirme sıralamasının en kötü zamanı nedir?

- Bilgisayarın hızına bağlıdır:
  - bağıl (rölatif) zaman (aynı makinede),
  - mutlak (absolüt) zaman (farklı makinelerde).

BÜYÜK FİKİR:

- Makineye bağımlı sabitleri görmezden gel.
- $n \to \infty$ ' a yaklaştıkça, $T(n)$'nin ***büyümesi***ne bak.

" Asimptotik Çözümleme"



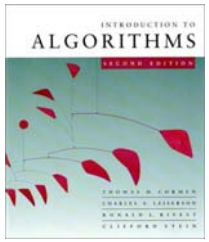
Θ- simgelemi (notation)

Matematik:

$\Theta(g(n)) = \{ f(n)$: Öyle c_1, c_2, n_0 pozitif sabit sayıları vardır ki tüm $n \geq n0\}$ için $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$.

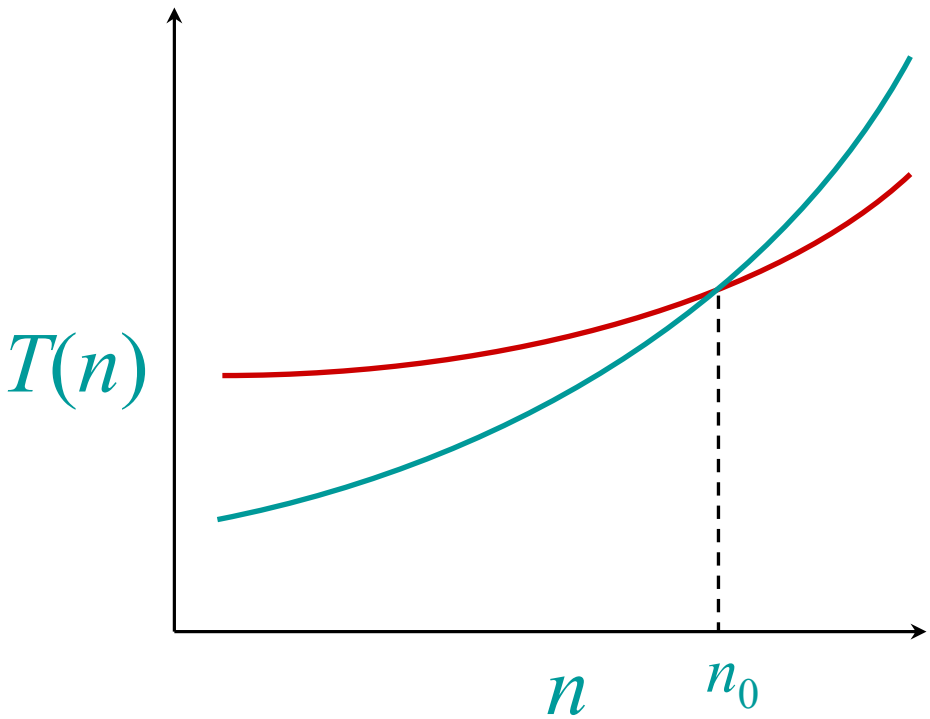
Mühendislik:

- Düşük değerli terimleri at; ön sabitleri ihmal et.
- Örnek: $3n^3 + 90n^2 - 5n + 6046 = \Theta(n^3)$

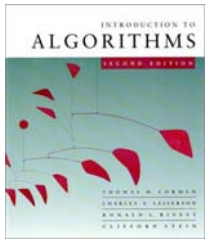


Asimptotik başarım

n yeterince büyürse, $\Theta(n^2)$ algoritması
bir $\Theta(n^3)$ algoritmasından her zaman daha hızlıdır.



- Öte yandan asimptotik açıdan yavaş algoritmaları ihmal etmemeliyiz.
- Gerçek dünyada tasarımın mühendislik hedefleriyle dikkatle dengelenmesi gereklidir.
- Asimptotik çözümleme düşüncemizi yapılandırmada önemli bir araçtır.



Araya yerleştirme sıralaması çözümlemesi

En kötü durum: Giriş tersten sıralıysa.

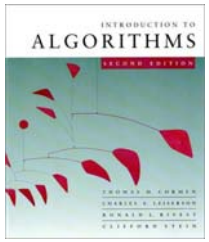
$$T(n) = \sum_{j=2}^{n} \Theta(j) = \Theta(n^2)$$ [aritmetik seri]

Ortalama durum: Tüm permutasyonlar eşit olasılıklı.

$$T(n) = \sum_{j=2}^{n} \Theta(j/2) = \Theta(n^2)$$

Araya yerleştirme sıralaması hızlı bir algoritma mıdır ?

- Küçük n değerleri için olabilir.
- Büyük n değerleri için asla!

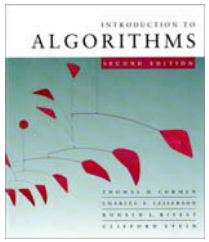


Birleştirme sıralaması

BİRLEŞTİRME-SIRALAMASI $A[1\ldots n]$

1. Eğer $n = 1$ ise, işlem bitti.
2. $A[\,1\ldots\lceil n/2\rceil\,]$ve
   $A[\,\lceil n/2\rceil+1\ldots n\,]$'yi özyinelemeli sırala.
3. 2 sıralanmış listeyi "***Birleştir***".

Anahtar altyordam: **Birleştirme**



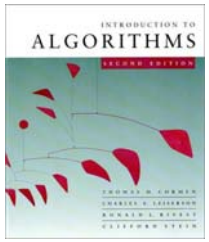
Sıralı iki dizilimi birleştirme

20 12

13 11

7 9

2 1



Sıralı iki dizilimi birleştirme

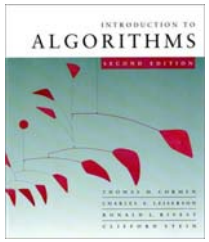
20 12

13 11

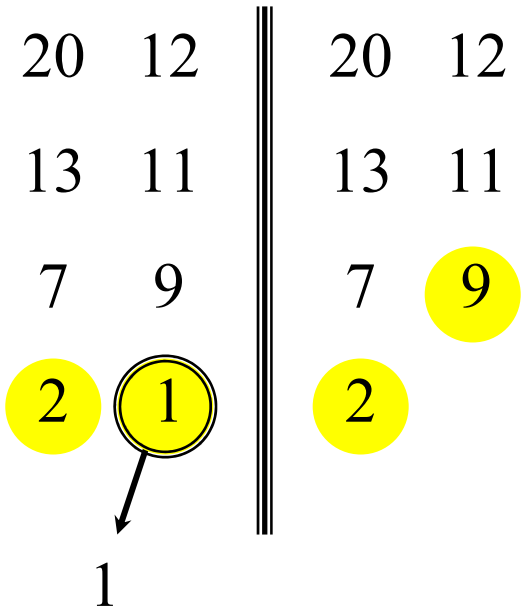
7 9

2 1

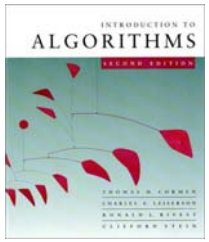
1



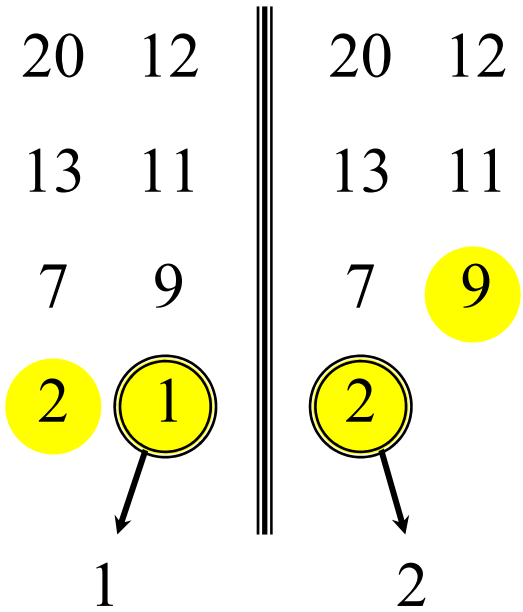
Sıralı iki dizilimi birleştirme

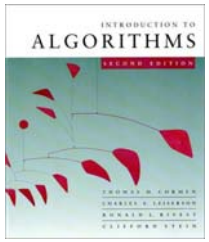


1

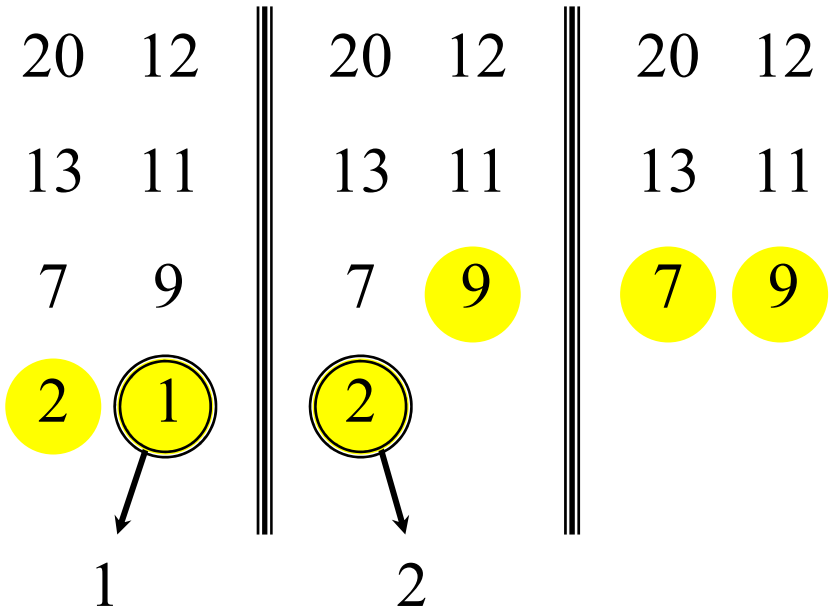


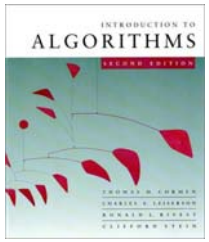
Sıralı iki dizilimi birleştirme



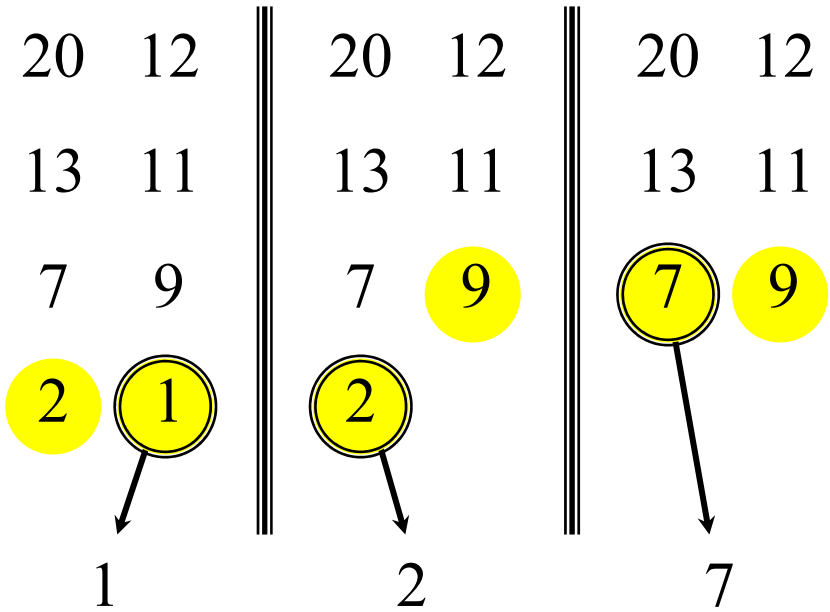


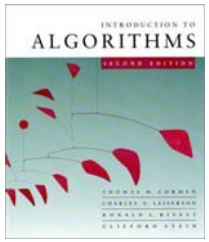
Sıralı iki dizilimi birleştirme



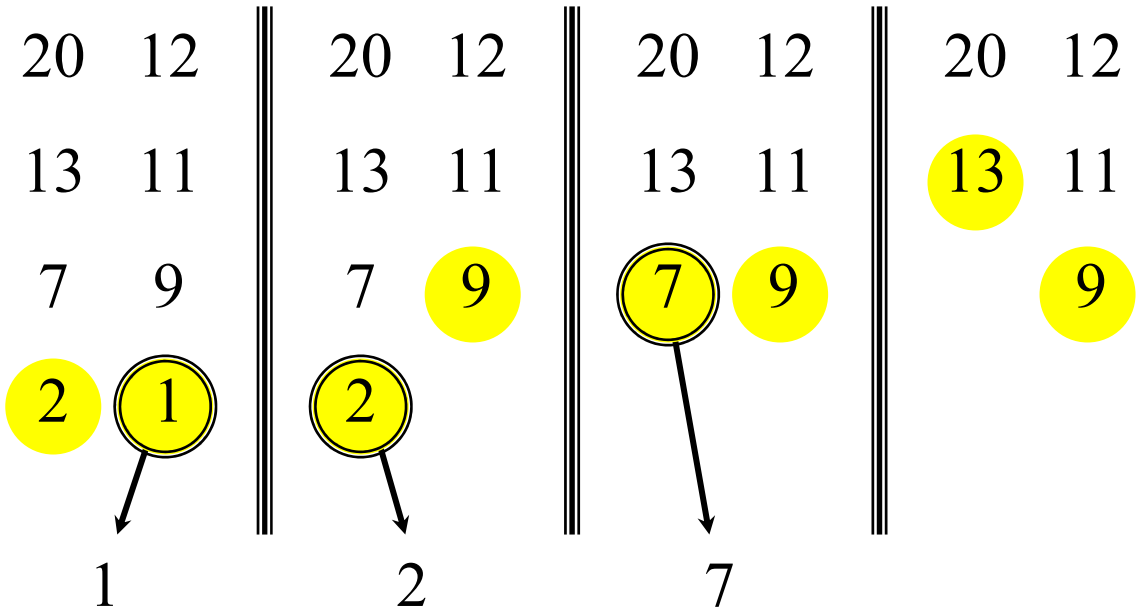


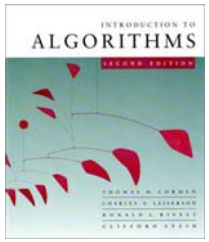
Sıralı iki dizilimi birleştirme



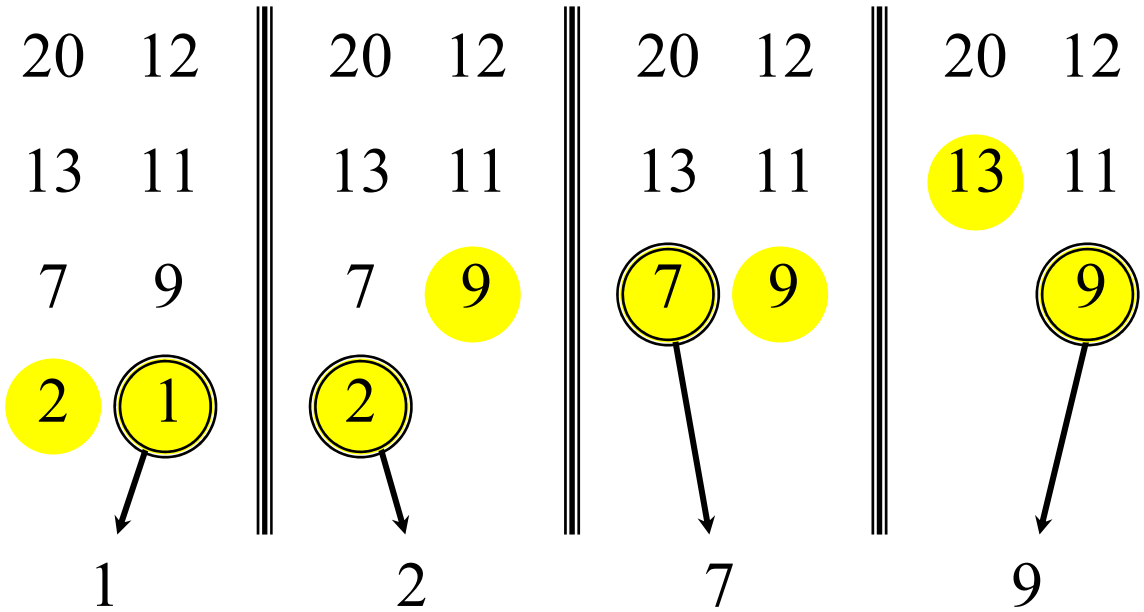


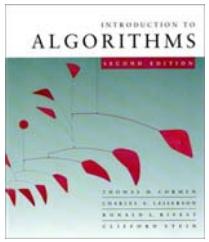
Sıralı iki dizilimi birleştirme



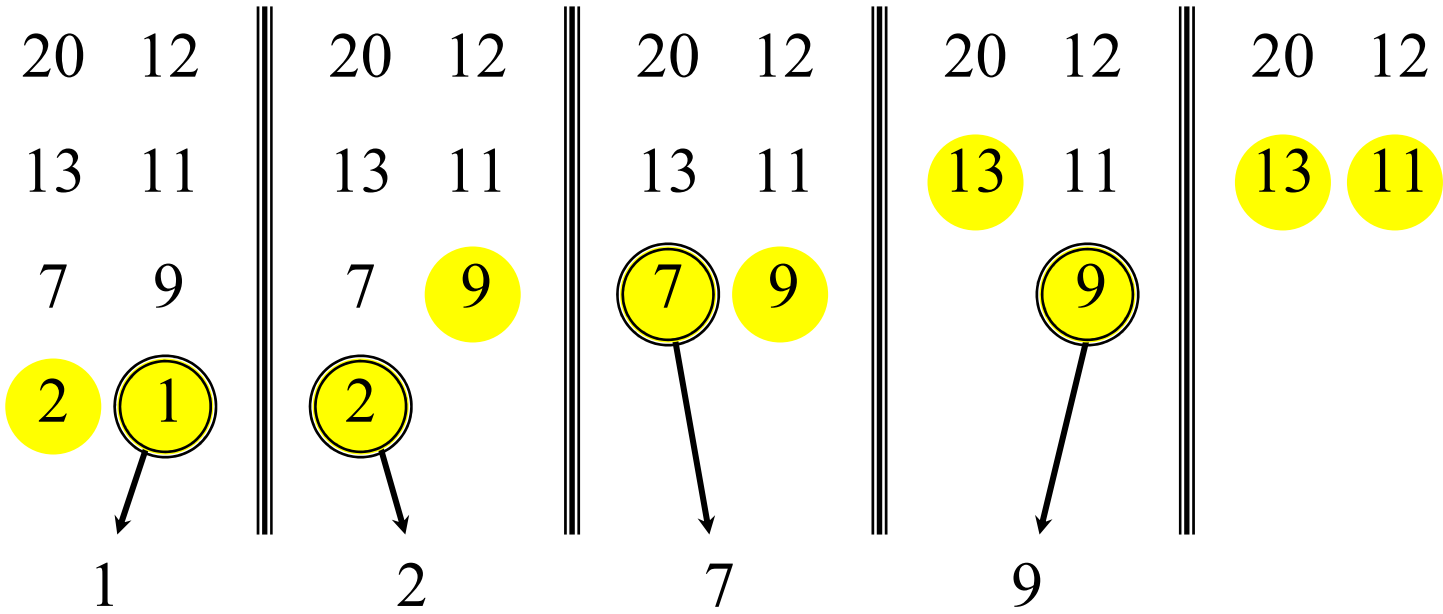


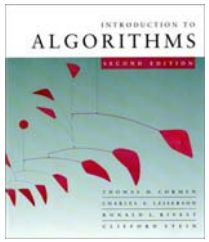
Sıralı iki dizilimi birleştirme



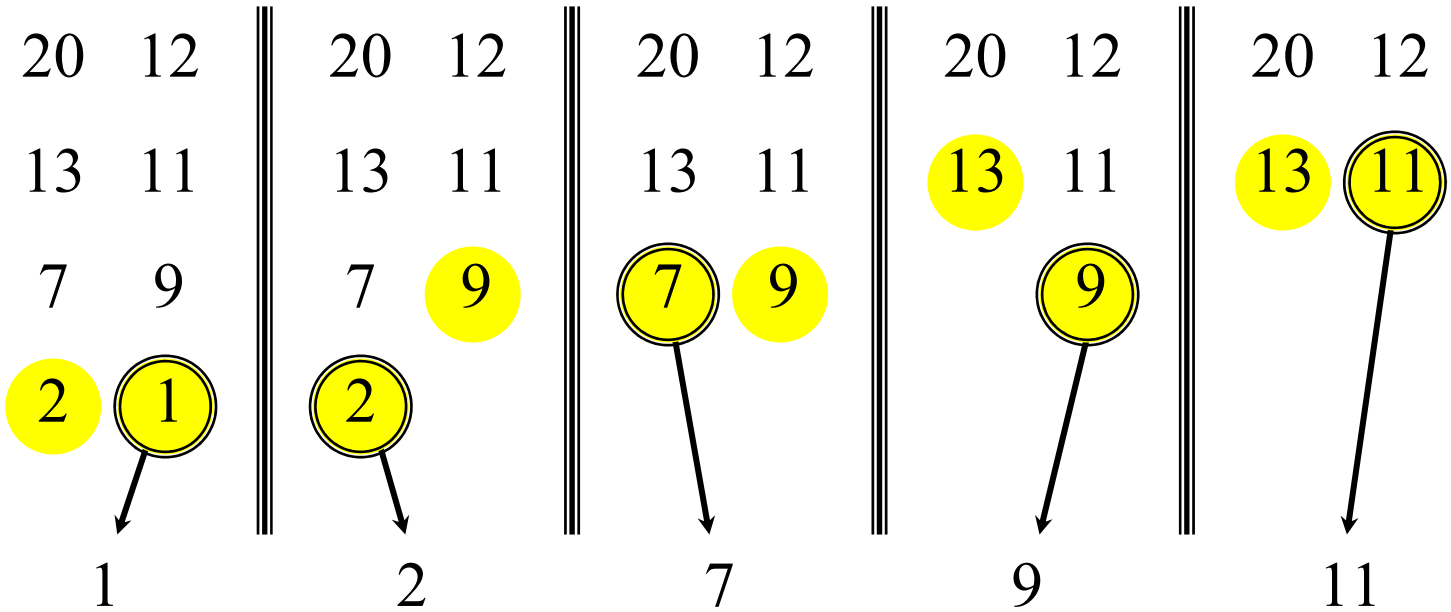


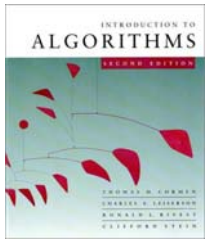
Sıralı iki dizilimi birleştirme



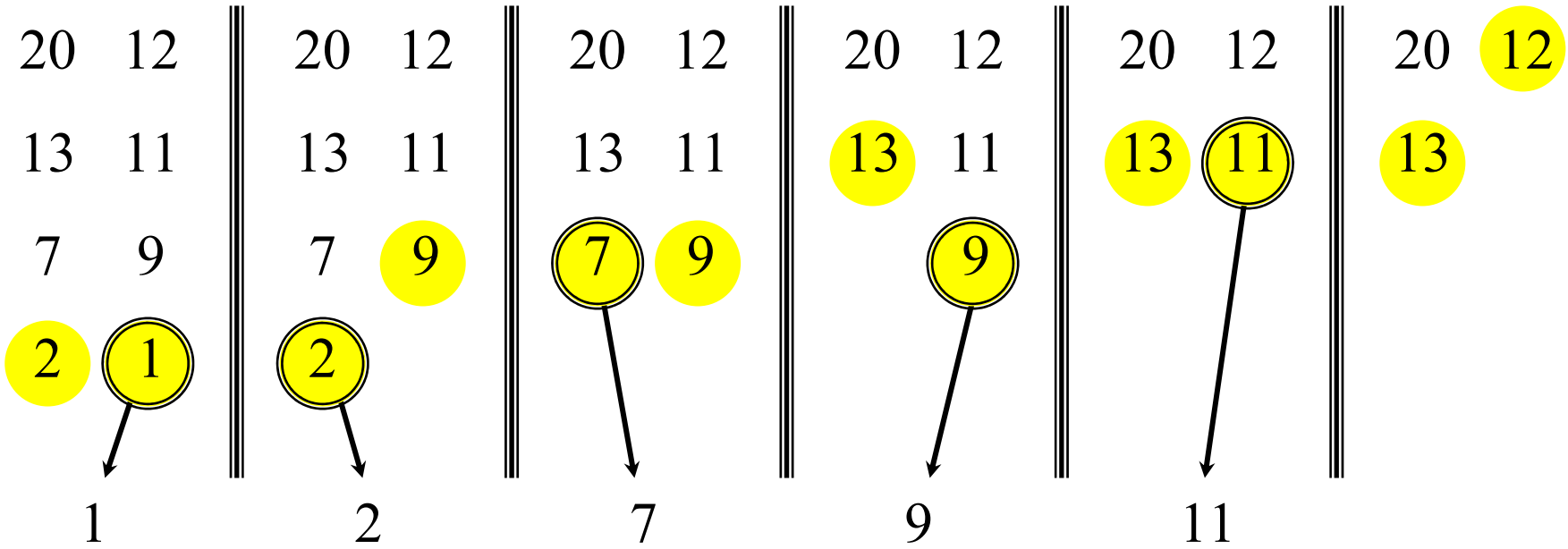


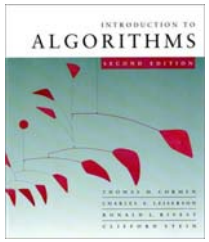
Sıralı iki dizilimi birleştirme



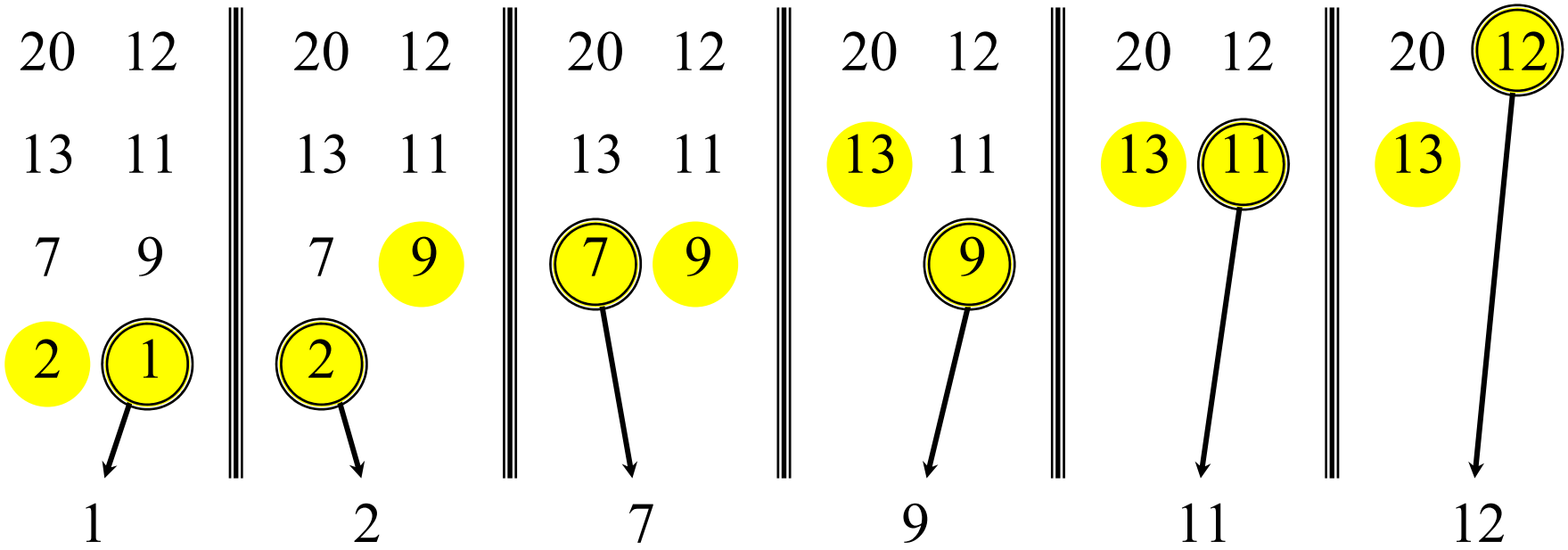


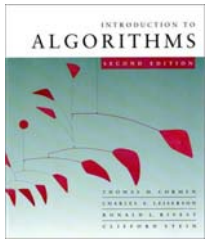
Sıralı iki dizilimi birleştirme



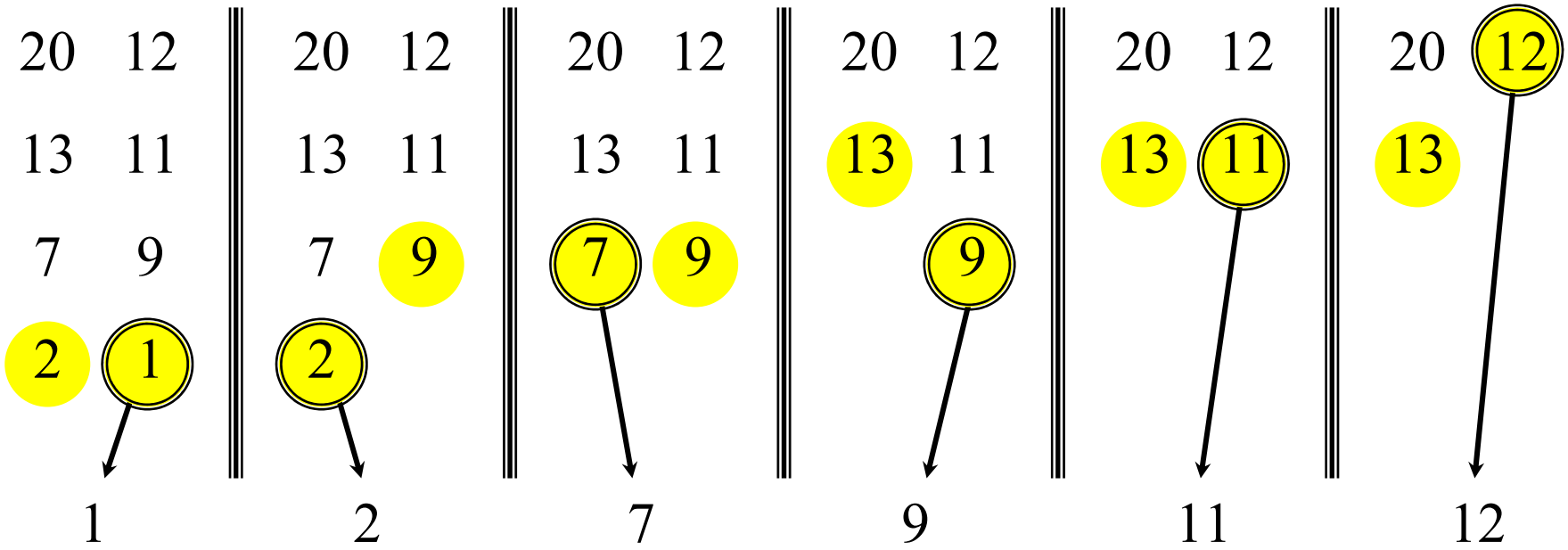


Sıralı iki dizilimi birleştirme

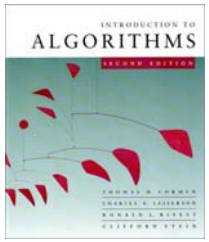




Sıralı iki dizilimi birleştirme



Süre = $\Theta(n)$, toplam n elemanı birleştirmek için (doğrusal zaman).



Birleştirme sıralamasının çözümlenmesi

$T(n)$

$\Theta(1)$

$2T(n/2)$

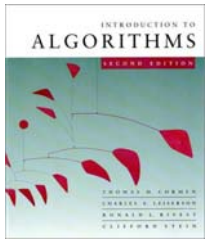
$\Theta(n)$

Suistimal

BİRLEŞTİRME-SIRALAMASI $A[1\ ..\ n]$

1. Eğer $n = 1$'se, bitir.
2. Yinelemeli olarak $A[\ 1\ ..\ \lceil n/2 \rceil\ ]$ ve $A[\ \lceil n/2 \rceil + 1\ ..\ n\ ]$'yi sırala.
3. *2* sıralı listeyi ***"Birleştir"***

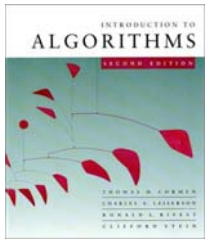
Özensizlik: $T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor)$ olması gerekir, ama asimptotik açıdan bu önemli değildir.



Birleştirme sıralaması için yineleme

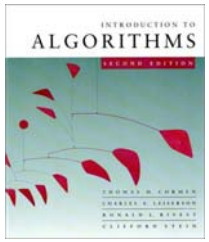
$$T(n) = \begin{cases} \Theta(1) \text{ eğer } n = 1 \text{ise;} \\ 2T(n/2) + \Theta(n) \text{ eğer } n > 1 \text{ise.} \end{cases}$$

- Genellikle ***n***'nin küçük değerleri için taban durumu (base case) olan $T(n) = \Theta(1)$ 'i hesaplara katmayacağız; ama bunu sadece yinelemenin asimptotik çözümünü etkilemiyorsa yapacağız.
- 2. Derste $T(n)$'nin üst sınırını bulmanın birkaç yolunu inceleyeceğiz.



Yineleme ağacı

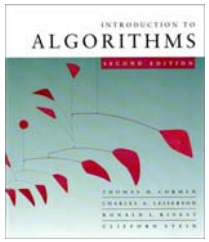
$T(n) = 2T(n/2) + cn$'yi çözün; burada $c > 0$ bir sabittir.



Yineleme ağacı

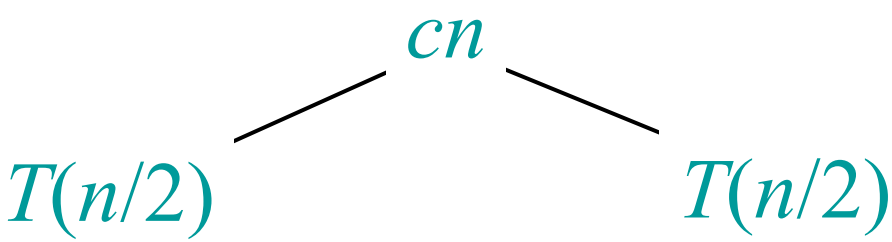
$T(n) = 2T(n/2) + cn$'i çözün; burada $c > 0$ bir sabittir.

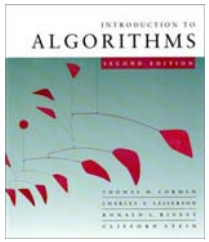
$$T(n)$$



Yineleme ağacı

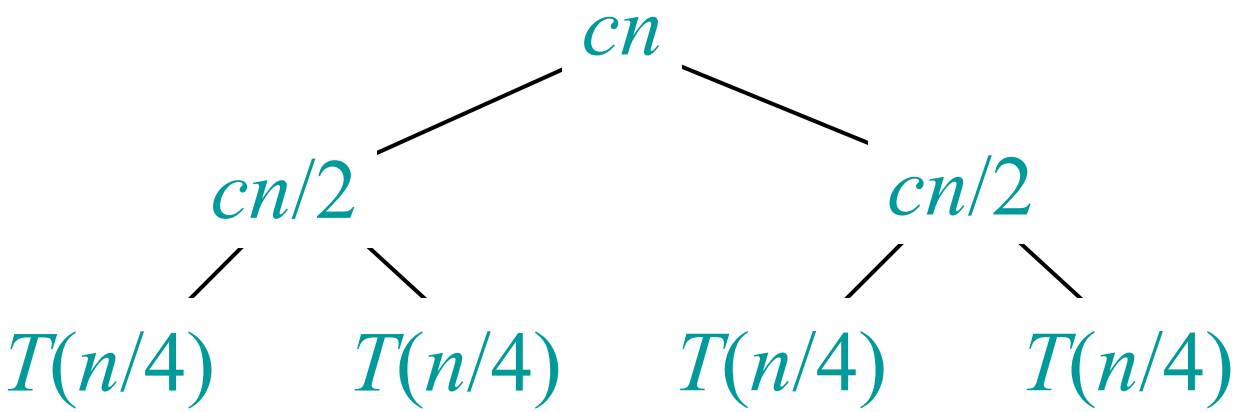
$T(n) = 2T(n/2) + cn$'i çözün; burada $c > 0$ bir sabittir.

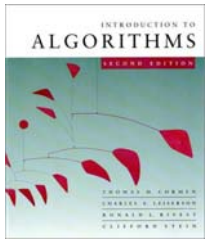




Yineleme ağacı

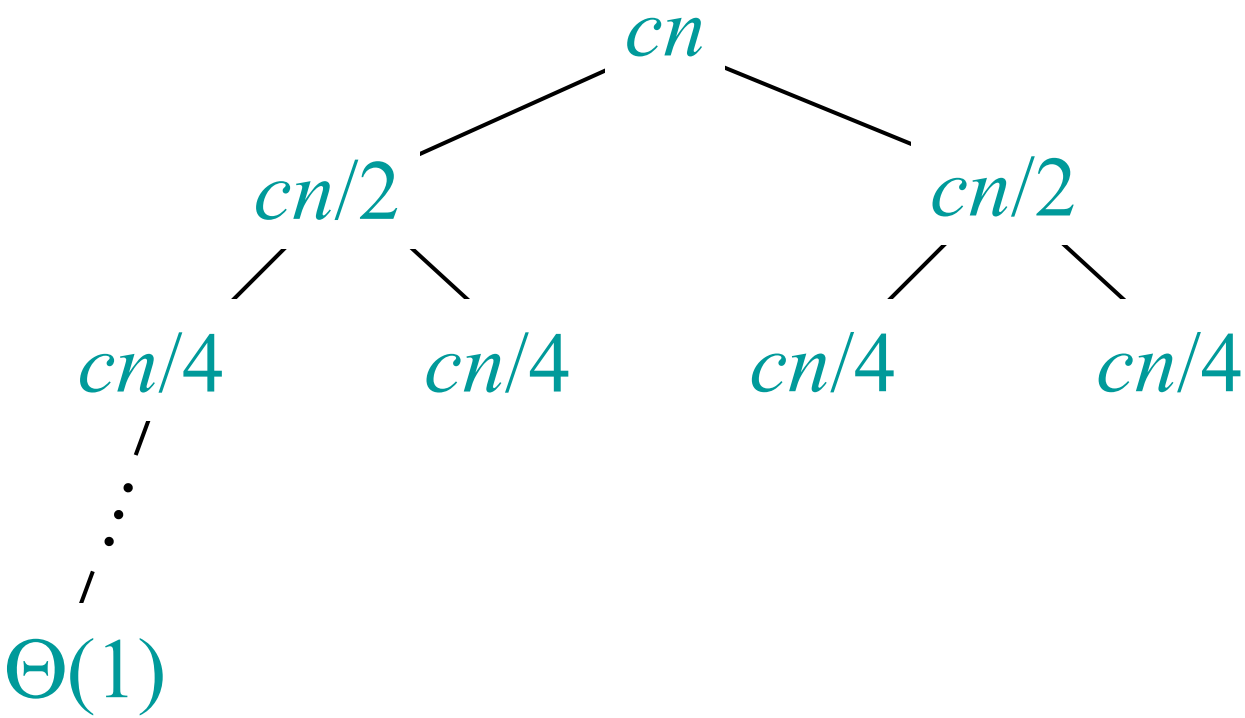
$T(n) = 2T(n/2) + cn$'i çözün; burada $c > 0$ bir sabittir.

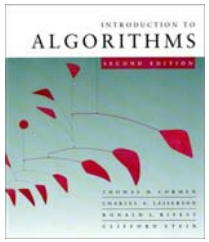




Yineleme ağacı

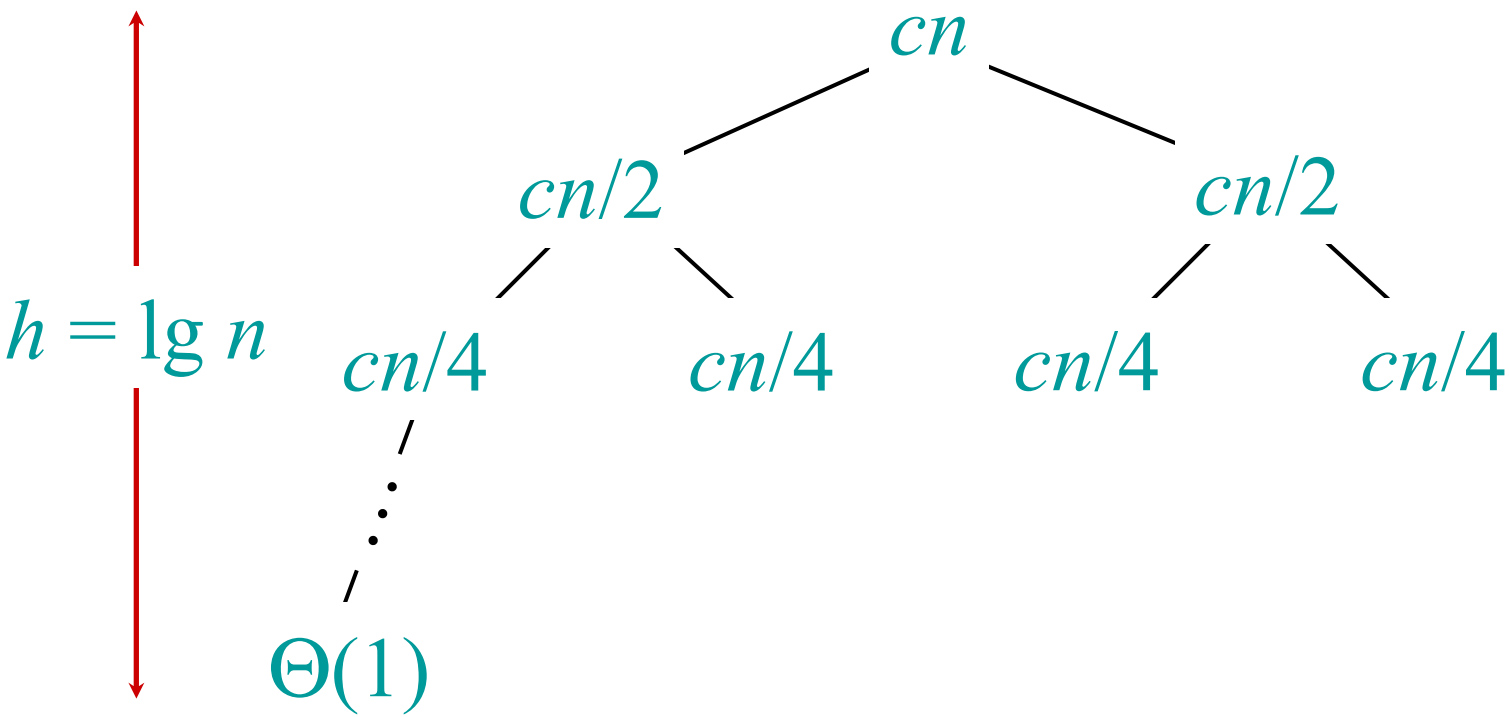
$T(n) = 2T(n/2) + cn$'i çözün; burada $c > 0$ bir sabittir.'

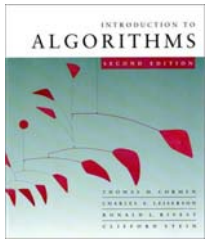




Yineleme ağacı

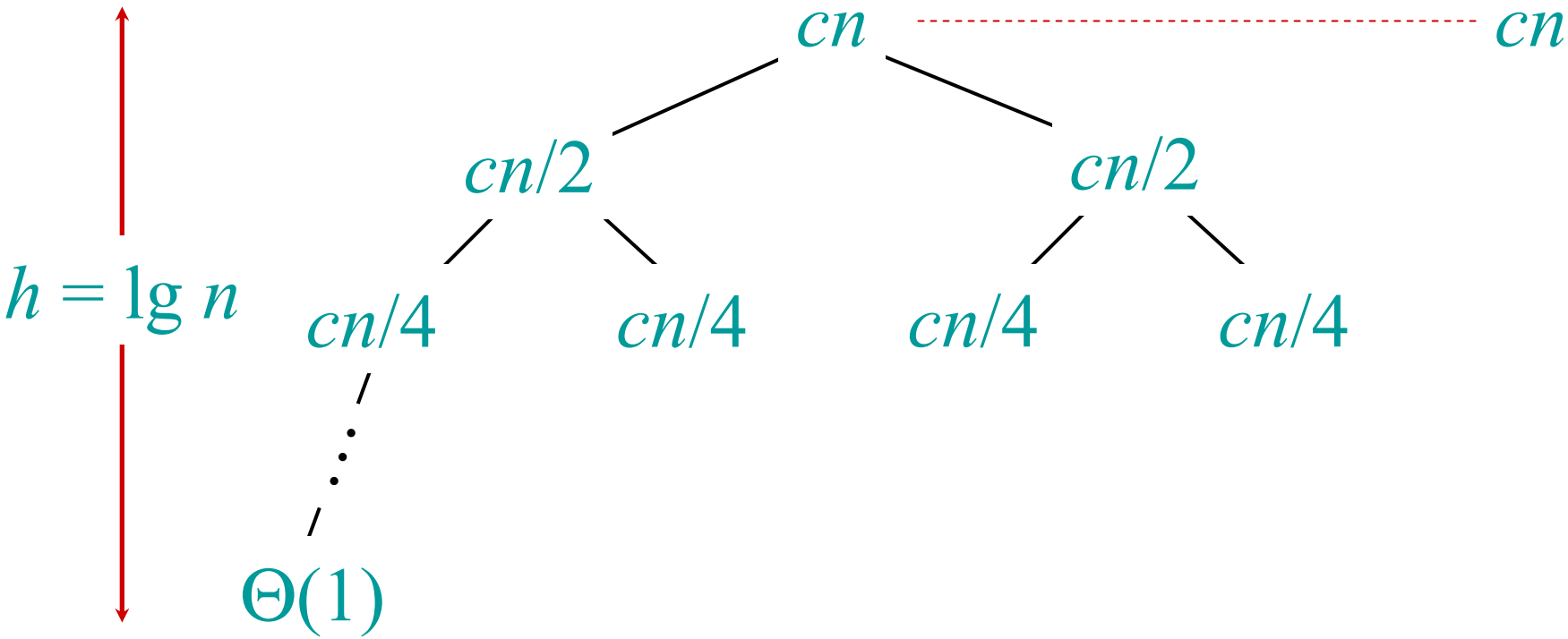
$T(n) = 2T(n/2) + cn$'i çözün; burada $c > 0$ bir sabittir.

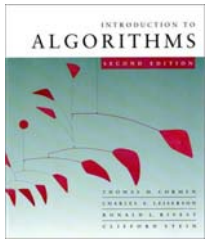




Yineleme ağacı

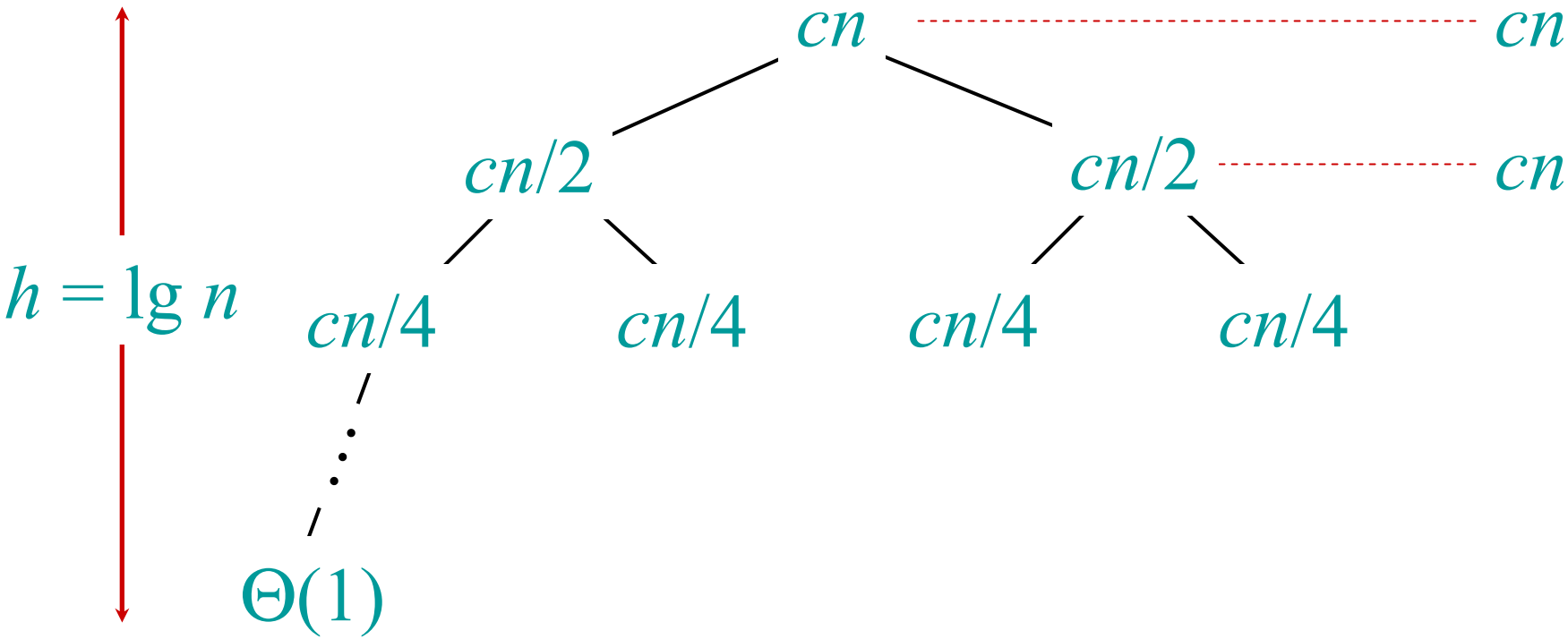
$T(n) = 2T(n/2) + cn$'i çözün; burada $c > 0$ bir sabittir.

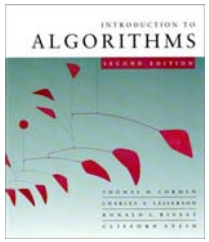




Yineleme ağacı

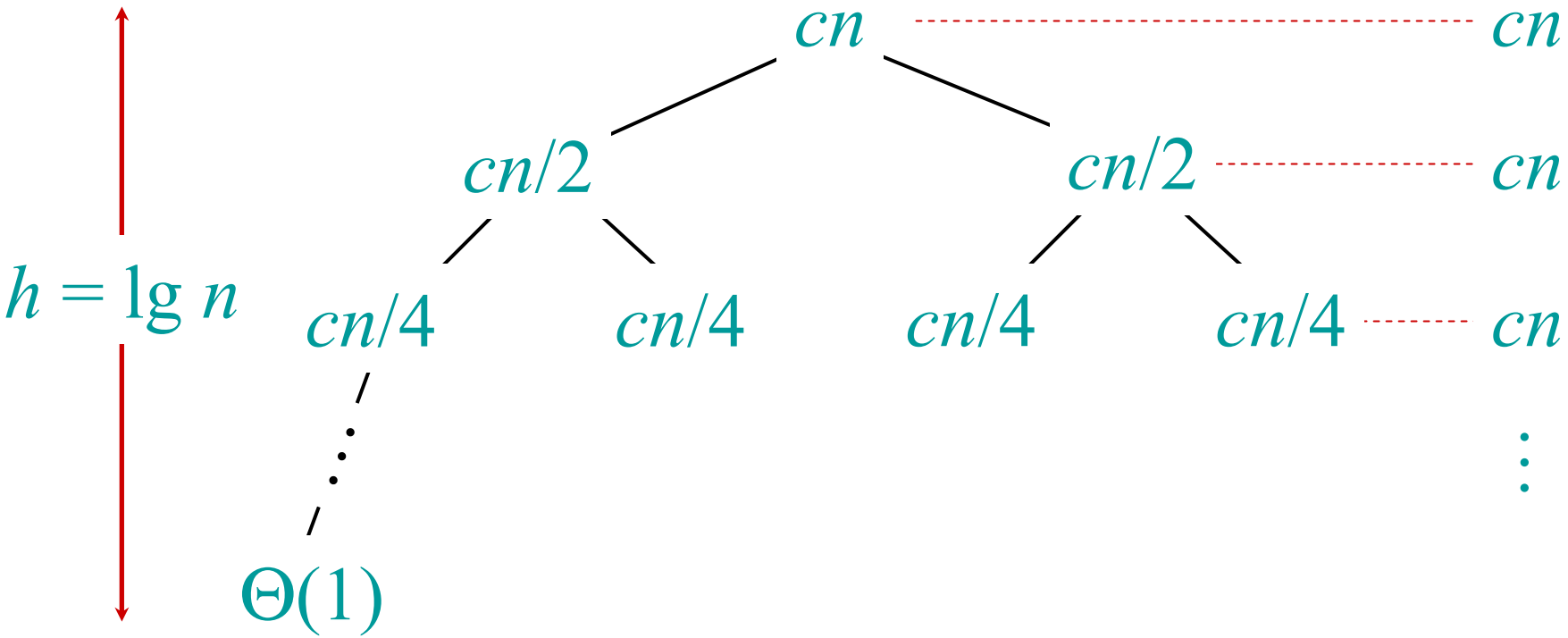
$T(n) = 2T(n/2) + cn$'i çözün; burada $c > 0$ bir sabittir.

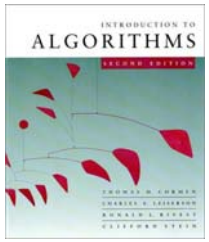




Yineleme ağacı

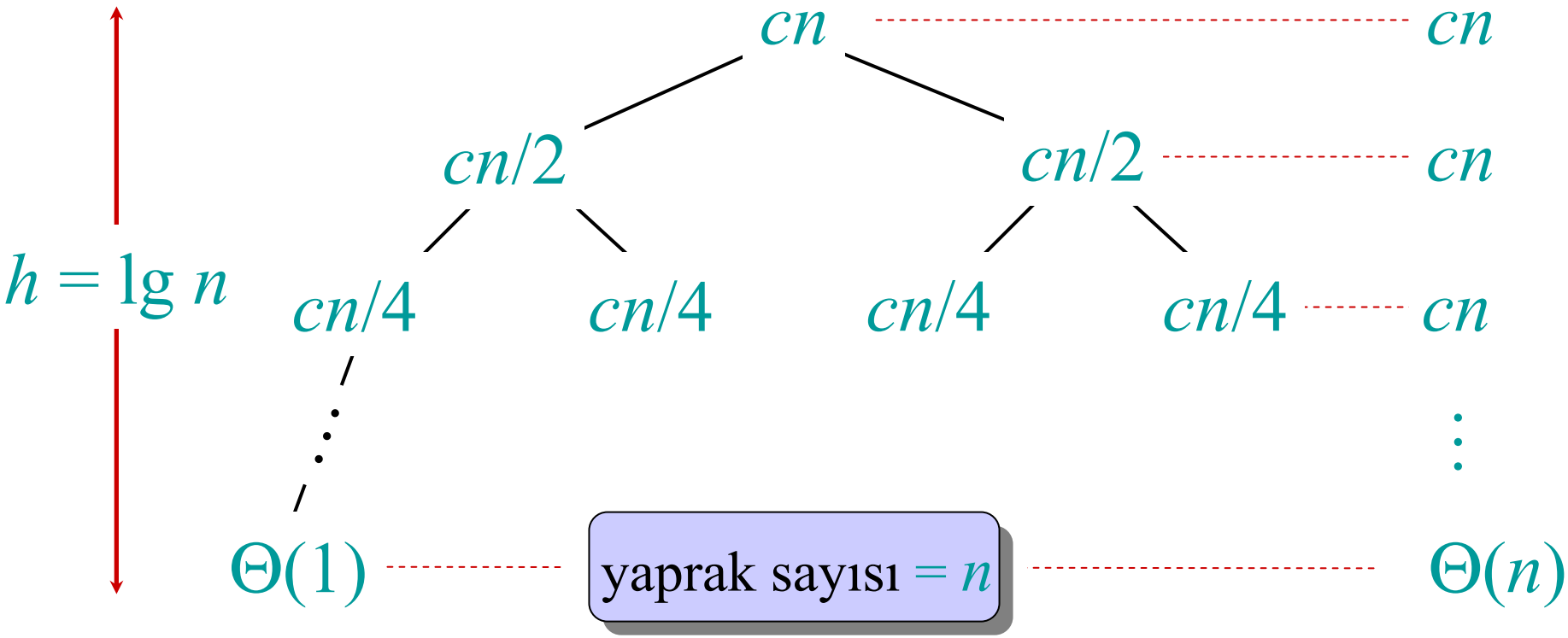
$T(n) = 2T(n/2) + cn$'i çözün; burada $c > 0$ bir sabittir.

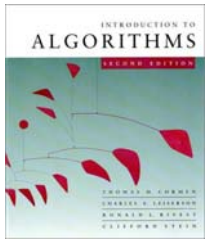




Yineleme ağacı

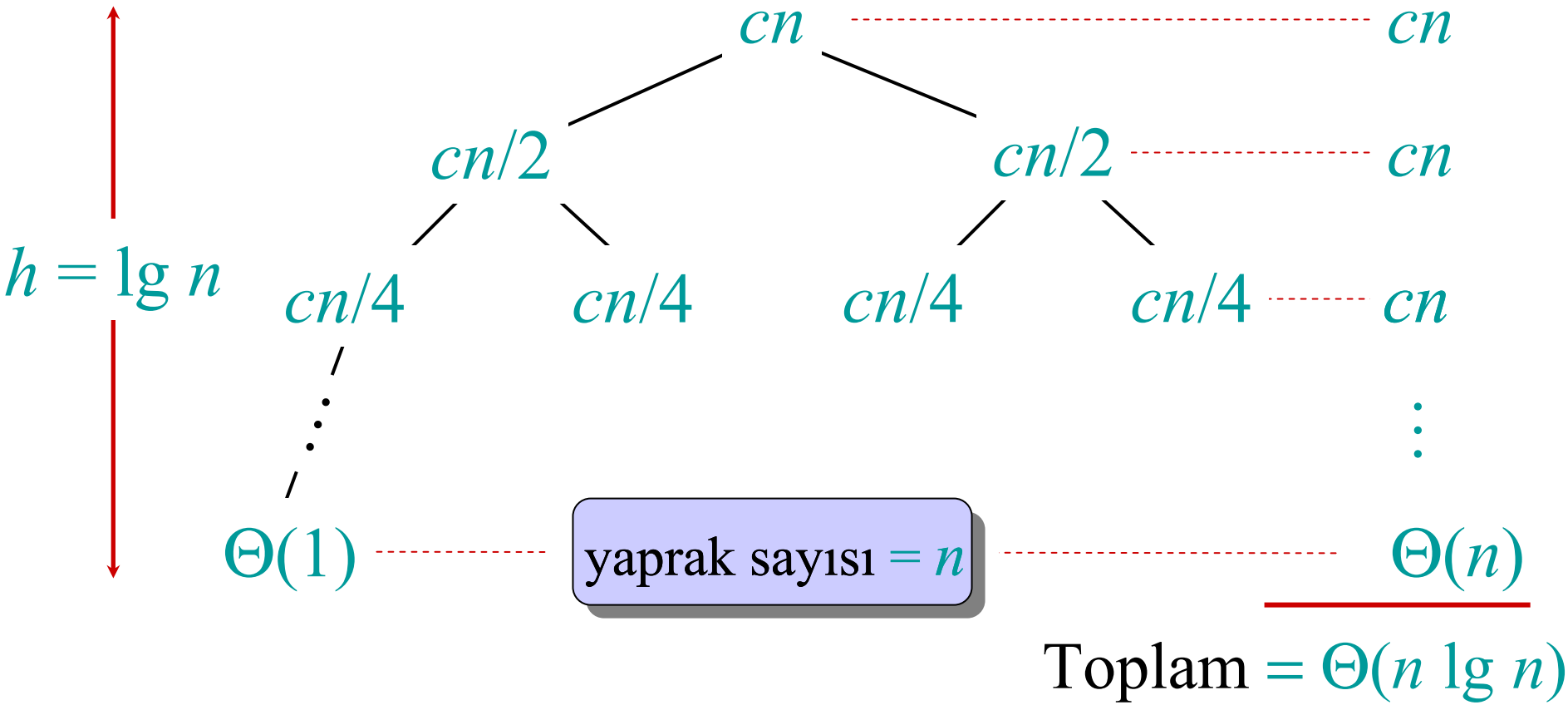
$T(n) = 2T(n/2) + cn$'i çözün; burada $c > 0$ bir sabittir.

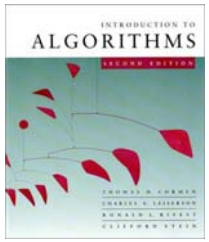




Yineleme ağacı

$T(n) = 2T(n/2) + cn$'i çözün; burada $c > 0$ bir sabittir.





Sonuçlar

- $\Theta(n \lg n)$, $\Theta(n^2)$'dan daha yavaş büyür.

- En kötü durumda, birleştirme sıralaması asimptotik olarak araya yerleştirme sıralamasından daha iyidir.
- Pratikte, birleştirme sıralaması araya yerleştirme sıralamasını $n > 30$ değerlerinde geçer.
- Bunu kendiniz deneyin!