

5.Hafta

Alt Sınırları Sıralama

Doğrusal-Zaman (linear time)

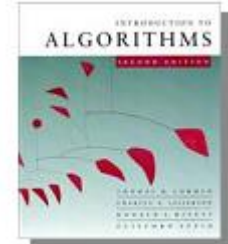
Sıralaması (devam)

Alt Sınırları Sıralama

- Karar ağaçları

Doğrusal-Zaman Sıralaması

- Sayma sıralaması
- **Taban sıralaması**
- Kova sıralaması



Sayma Sıralaması

$\Theta(k)$ { **for** $i \leftarrow 1$ **to** k
 do $C[i] \leftarrow 0$

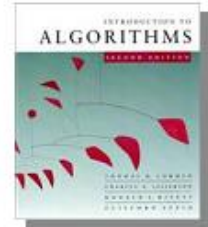
$\Theta(n)$ { **for** $j \leftarrow 1$ **to** n
 do $C[A[j]] \leftarrow C[A[j]] + 1$

$\Theta(k)$ { **for** $i \leftarrow 2$ **to** k
 do $C[i] \leftarrow C[i] + C[i-1]$

$\Theta(n)$ { **for** $j \leftarrow n$ **down to** 1
 do $B[C[A[j]]] \leftarrow A[j]$
 $C[A[j]] \leftarrow C[A[j]] - 1$

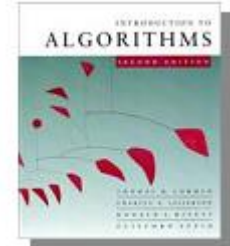
$\Theta(n + k)$

Taban (Radix) sıralaması



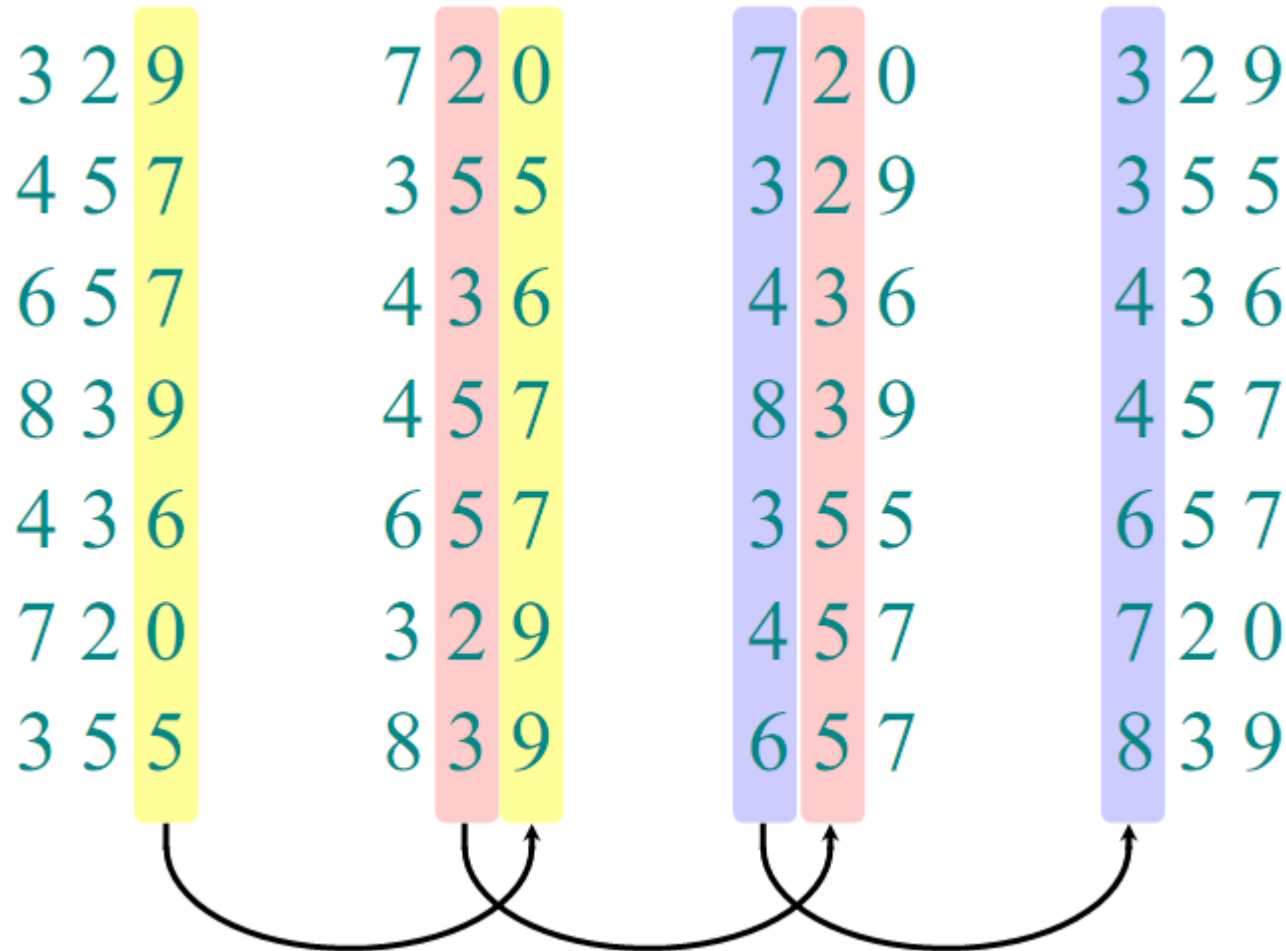
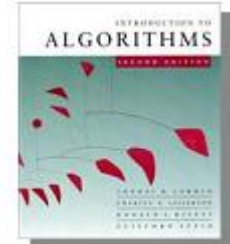
- En az öneme sahip basamaktan başlayarak sıralama yapar.
- Birden fazla anahtara göre sıralama gerektiğinde kolaylıkla kullanılabilir.
- Ör: tarihe göre sıralamada yıl, ay, gün'e göre sıralama yapılır. Tarih sıralamasında önce gün, sonra ay ve yıl' a göre kolayca sıralanır.

Taban (Radix) sıralaması

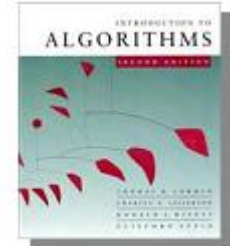


- Basamak basamak sıralama.
- **Kötü fikir:** sıralamaya önceli en önemli basamaktan başlamak.
- **İyi fikir:** Sıralamaya **en önemsiz basamaktan** başlamak ve **ek kararlı** sıralama uygulamak.

Taban sıralaması uygulaması

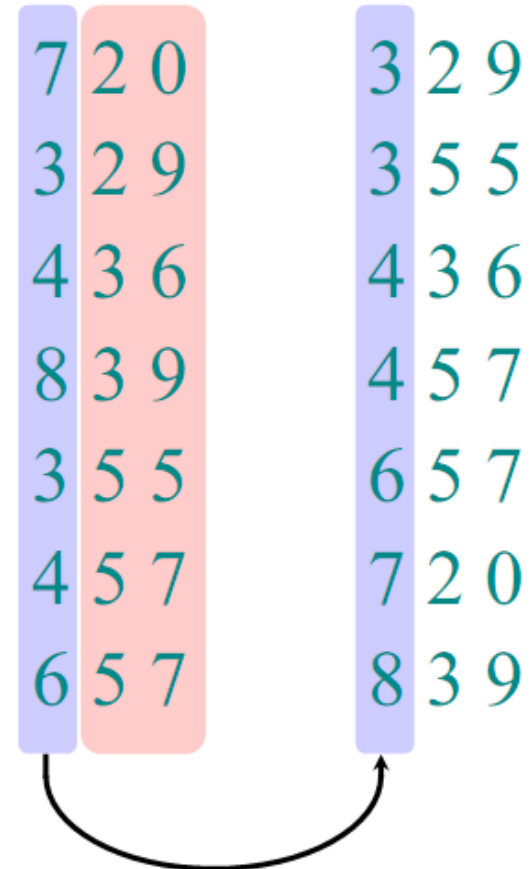


Taban sıralaması uygulaması

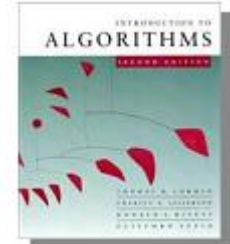


Basamak konumunda tümevarım

- Sayıların düşük düzeyli $t - 1$ basamaklarına göre sıralandığını varsayın.
- t basamağında sıralama yapın.

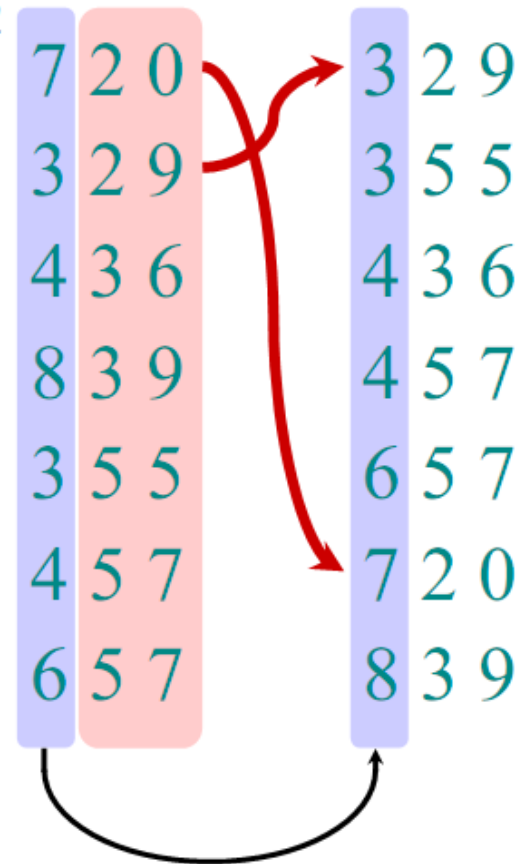


Taban sıralaması uygulaması

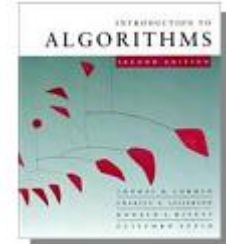


Basamak konumunda tümevarım

- Sayıların düşük düzeyli $t-1$ basamaklarına göre sıralandığını varsayın.
- t basamağında sıralama yapın.
 - t basamağında farklı olan iki sayı doğru sıralanmış.

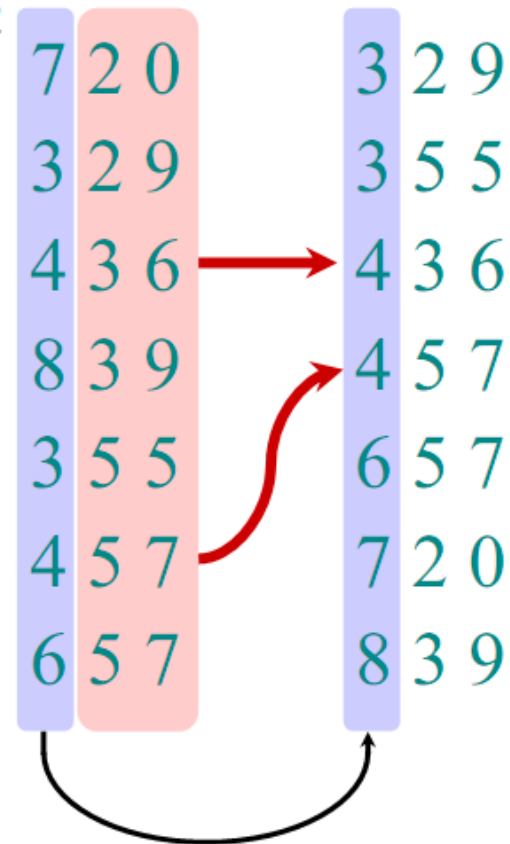


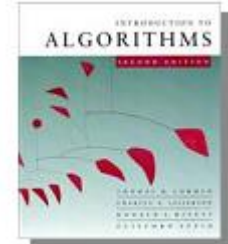
Taban sıralaması uygulaması



Basamak konumunda tümevarım

- Sayıların düşük düzeyli $t-1$ basamaklarına göre sıralandığını varsayın.
- t basamağında sıralama yapın.
 - t basamağında farklı olan iki sayı doğru sıralanmış.
 - t basamağındaki iki eşit sayının girişteki sıraları muhafaza edilmiş \Rightarrow doğru sıra.



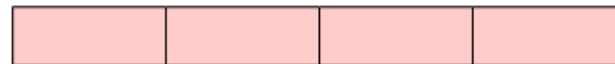


Taban sıralamasının çözümlemesi

- Sayma sıralamasını ek kararlı sıralama varsayın.
- Herbiri b bit olan n bilgiişlem sözcüğünü sıralayın.
- Her sözcüğün basamak yapısı b/r taban- 2^r olarak görülebilir.

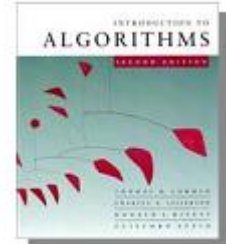
8 8 8 8

Örnek: 32-bit sözcük



$r = 8 \Rightarrow b/r = 4$ ise, taban- 2^8 basamak durumunda sıralama 4 geçiş yapar; veya $r = 16 \Rightarrow b/r = 2$ ise, taban- 2^{16} basamakta 2 geçiş yapar.

Kaç geçiş yapmalıyız?



Taban sıralamasının çözümlemesi

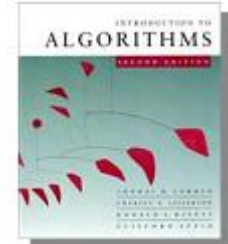
Hatırla: Sayma sıralaması $\Theta(n + k)$ süresini alır;
(0 ile $k - 1$ aralığında n sayıyı sıralamak için).
Her b -bitlik sözcük r -bitlik parçalara ayrılırsa,
sayma sıralamasının her geçişi $\Theta(n + 2^r)$ süre alır.
Bu durumda b/r geçiş olduğundan, elimizde:

$$T(n, b) = \Theta\left(\frac{b}{r}(n + 2^r)\right) \text{ olur.}$$

r' yi, $T(n, b)$ ' yi en aza düşürecek gibi seçin:

- r' yi arttırmak daha az geçiş demektir, ama $r \gg \lg n$ olduğundan, süre üstel olarak artar.

r' yi seçmek



(r' 'ye göre)

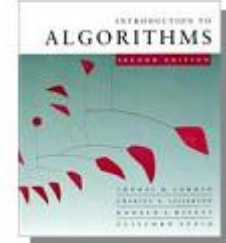
$T(n, b)$ 'yi türevini alıp 0'a eşitleyerek en aza düşürün.

Veyahut da , istemediğimiz değer $2^{r'} \gg n$ olduğundan, bu sınırlamaya bağlı kalarak r' 'yi olabildiğince büyük seçmenin asimptotik bir sakıncası olmadığını gözleyin.

$r = \lg n$ seçimi $T(n, b) = \Theta(bn/\lg n)$ anlamına gelir.

- 0 ile $n^d - 1$ aralığındaki sayılarla $b = d \lg n$ 'yi elde ederiz. \Rightarrow taban sıralaması $\Theta(dn)$ süresini alır.

Not: Değer aralığı $0 \dots 2^b - 1 = 0 \dots n^d$ kabul edip her iki tarasın logaritması alınır. $b = d \lg n$ olur. Burada d basamak sayısıdır.



Sonuçlar

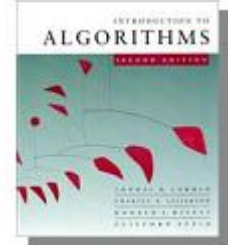
Pratikte taban sıralaması büyük girişler için hızlıdır; aynı zamanda kod yazması ve bakımı kolaydır.

Örnek (32-bitlik sayılar için):

- En çok 3 geçiş (≥ 2000 sayının sıralanmasında).
- Birleştirme sıralaması /çabuk sıralama $\lceil \lg 2000 \rceil$ en az 11 geçiş yaparlar.

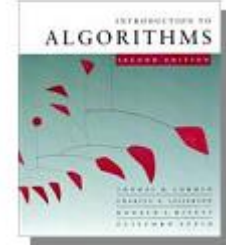
Dezavantajı: Çabuk sıralamanın aksine, taban sıralamasının yer referansları zayıftır ve bu nedenle ince ayarlı bir çabuk sıralama, dik bellek sıradüzeni olan günümüz işlemcilerinde daha iyi çalışır.

Pratikte radix sort yani taban sıralaması, sayılarınız gerçekten küçük değilse çok hızlı bir algoritma değildir.



Radix Sort

```
○ public void RadixSort(int[] Dizi)    {  
○     // Yardımcı dizimiz  
○     int[] t = new int[Dizi.Length];  
○     // her defasında kaç bit işleme alınacak  
○     int r = 4; // 2, 8 veya 16 bit ile kaç geçiş yapılacağı denenebilir  
○     // int 4 byte yani 32 bit  
○     int b = 32;  
○     // counting ve prefix dizileri  
○     // (unutmayın bu dizlerin boyutu 2^r düzeyinde olacaktır.  
○     // Bu değerin uygun seçilmesi gerekir r=4 için dizi boyutu 16  
○     // r=16 için dizi boyutu 65535)  
○     int[] count = new int[1 << r];  
○     int[] pref = new int[1 << r];  
○     // grupların sayısı yani geçiş bulunuyor  
○     int groups = (int)Math.Ceiling((double)b / (double)r);  
○     // gruplara uygulanacak maske  
○     int mask = (1 << r) - 1; //r=4 için 15
```



Radix Sort

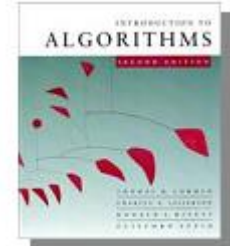
```

○ for (int c = 0, shift = 0; c < groups; c++, shift += r)
○ { // count dizisini resetleme
○     for (int j = 0; j < count.Length; j++) count[j] = 0;
○         // c inci grubun elamanlarının sayılması
○     for (int i = 0; i < Dizi.Length; i++)
○         count[(Dizi[i] >> shift) & mask]++;
○     // prefix dizisinin hesaplanması
○     pref[0] = 0;
○     for (int i = 1; i < count.Length; i++)
○         pref[i] = pref[i - 1] + count[i - 1];
○     // t[] dizisine c. inci grupta sıralanmış elamanların indisine
○     //uygun değerin Dizi den atanması
○     for (int i = 0; i < Dizi.Length; i++)
○         t[pref[(Dizi[i] >> shift) & mask]++] = Dizi[i];
○     // Dizi[]=t[] c inci guruba göre sıralanmış değerlerin diziyeye aktarılması
○     t.CopyTo(Dizi, 0);
○ }
○     // Dizi sıralı
○ }

```

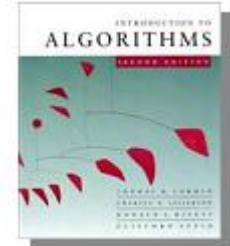
Çıktı: 32/4= 8 geçiş
 256, 1, 120, 10, 235, 987
 256, 1, 10, 120, 987, 235
 1, 10, 120, 235, 256, 987
 1, 10, 120, 235, 256, 987
 1, 10, 120, 235, 256, 987
 1, 10, 120, 235, 256, 987
 1, 10, 120, 235, 256, 987
 1, 10, 120, 235, 256, 987

Kova Sıralama (Bucket Sort)



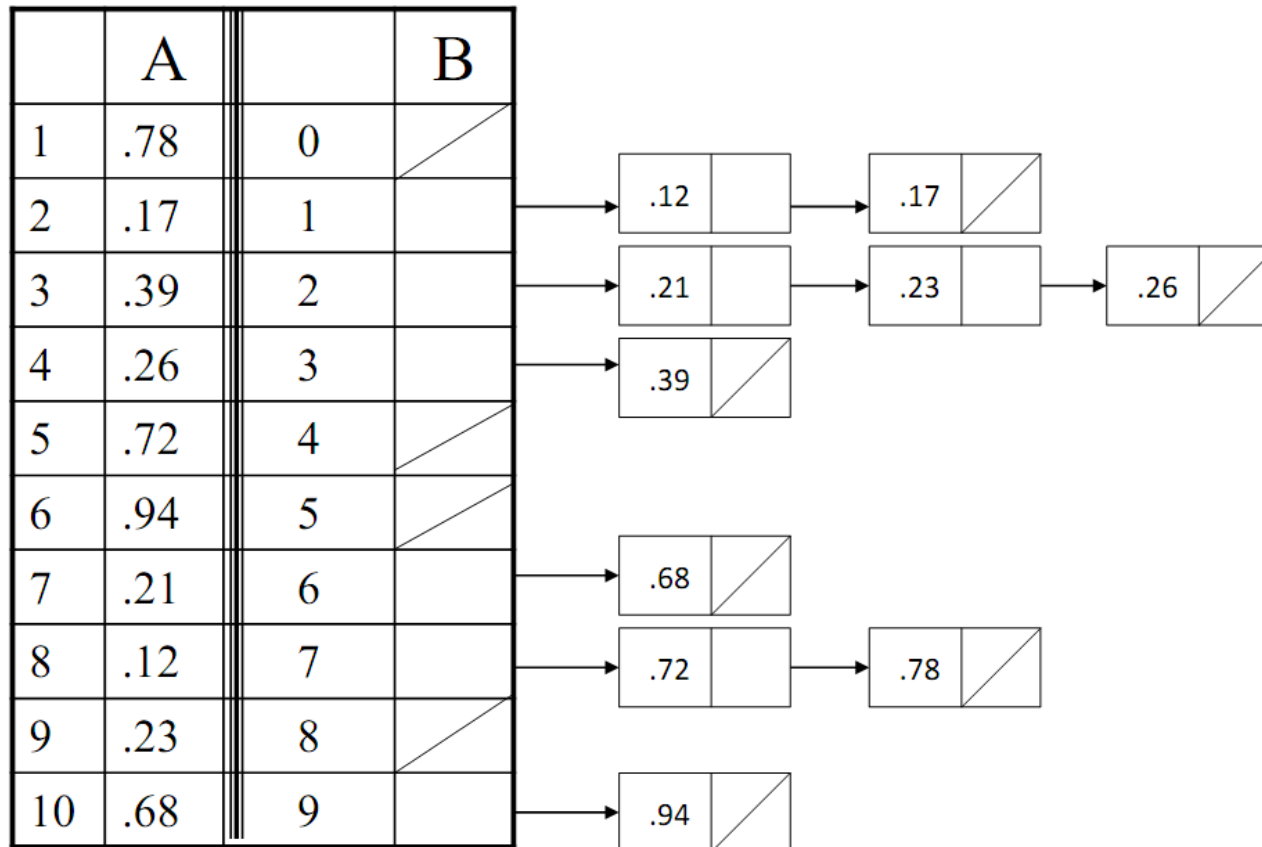
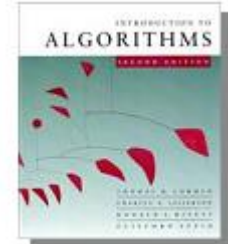
- **Kova Sıralaması** (ya da **sepet sıralaması**), sıralanacak bir diziyi parçalara ayırarak sınırlı sayıdaki *kovalara* (ya da *sepetlere*) atan bir sıralama algoritmasıdır.
- Ayrışma işleminin ardından her kova kendi içinde ya farklı bir algoritma kullanılarak ya da kova sıralamasını özyinelemeli olarak çağırarak sıralanır.
- Kova sıralaması aşağıdaki biçimde çalışır:
- Başlangıçta boş olan bir "kovalar" dizisi oluştur.
- Asıl dizinin üzerinden geçerek her öğeyi ilgili aralığa denk gelen kovaya at.
- Boş olmayan bütün kovaları sırala.
- Boş olmayan kovalardaki bütün öğeleri yeniden diziye al.

Kova Sıralama (Bucket Sort)

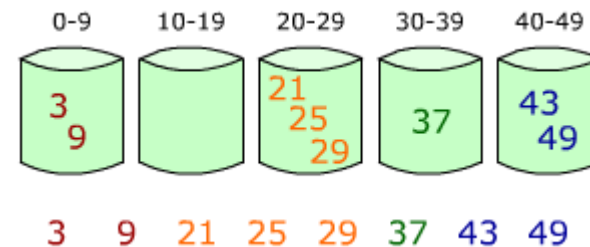
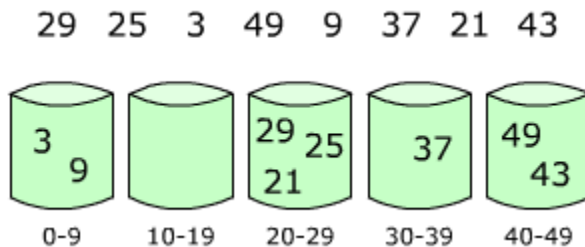


- Kova sıralaması doğrusal zamanda çalışır.
- Girişin düzgün dağılımlı olduğu kabul edilir.
- Random olarak $[0,1)$ aralığında oluşturulmuş giriş bilgileri olduğu kabul edilir.
- Temel olarak $[0, 1)$ aralığını n eşit alt aralığa böler ve girişi bu aralıklara dağıtır.
- Aralıklardaki değerleri insert sort ile sıralar.
- Aralıkları bir biri ardına ekleyerek sıralanmış diziyi elde eder.

Kova Sıralama:



Kova Sıralama (Bucket Sort)



BUCKET-SORT (A)

- $\Theta(n)$ 1. $n \leftarrow \text{length}(A)$
- $\Theta(n)$ 2. for $i \leftarrow 0$ to n
- $\Theta(n)$ 3. do insert $A[i]$ into list $B[\lfloor n \times A[i] \rfloor]$
- $\Theta(n)$ 4. for $i \leftarrow 0$ to n
- $\Theta(n_i^2)$ 5. do sort $B[i]$ with insertion sort
- $\Theta(n)$ 6. concatenate the lists $B[0], B[1], \dots, B[n-1]$ together in order

Çalışma zamanı= $\Theta(n)$

$\Theta(n_i^2)$ $B[i]$. Yerdeki eleman sayısını ifade eder. Beklenen çalışma süresi doğrusaldır

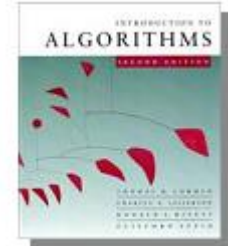
Worst case performance

$O(n^2)$

Average case performance

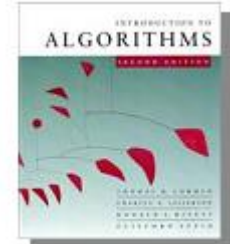
$O(n + k)$

Kova Sıralama (Bucket Sort)



- Değer aralıkları çok büyük seçilirse veya girişler düzgün dağılımlı değil ise sıralama $O(n^2)$ olur. Değer aralığını yani kova sayısını bulmak için
- Dizi boyutu * i. Dizi elamanı / (Girişlerin maksimumu + x_sayı)
- ile bulunabilir.
- Aralık-kova sayısı $m \rightarrow n * \max(A[i]) / (\max(A[i]) + x)$
- 29 25 3 49 9 37 21 43
- 0 1 2 3 4 5 6 7
- $n=8$, giriş maksimum değeri 49, $k > \text{maks} \rightarrow k=50$ üst sınır
- $29 * 8 / 50 = 4,64 \rightarrow \text{Kova}[4] = 29$
- $25 * 8 / 50 = 4 \rightarrow \text{Kova}[4] = 25 \rightarrow 29$
- $3 * 8 / 50 = 0,48 \rightarrow \text{Kova}[0] = 3$
- $49 * 8 / 50 = 7,84 \rightarrow \text{Kova}[7] = 49$

Kova Sıralama:



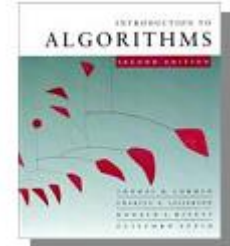
Algorithm	Worst-case running time	Average-case/ expected running time
Insert Sort	$\Theta(n^2)$	$\Theta(n^2)$
Merge Sort	$\Theta(n \log n)$	$\Theta(n \log n)$
Heap Sort	$\Theta(n \log n)$	$\Theta(n \log n)$
QuickSort	$\Theta(n^2)$	$\Theta(n \log n)$
Counting Sort	$\Theta(n+k)$	$\Theta(n+k)$
Radix Sort	$\Theta(d(n+k))$	$\Theta(d(n+k))$
Bucket Sort	$\Theta(n^2)$	$\Theta(n)$

Sıra İstatistikleri

Order Statistics

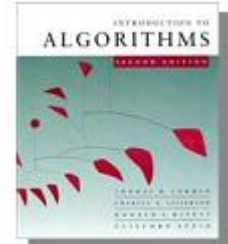
Rastgele böl ve fethet

- Beklenen sürenin çözümlemesi
 - En kötü durum doğrusal-süre
- sıra istatistikleri
- Çözümleme



Sıra İstatistikleri

- Doğrusal zaman çözümüne gereksinim duyulur.
- n elamanlı bir dizide i ' inci sıra istatistiği, i ' inci en küçük elemanı bulmak
- $i=1$ ise *minimum*
- $i=n$ ise *maximum*
- $i=n/2$ orta değeri (medyan)
 - Eğer n tek ise, 2 medyan vardır.
- *Sıra istatistiğini nasıl hesaplayabiliriz?*
- *Çalışma zamanı nedir?*



Sıra istatistikleri (doğrusal zamanda)

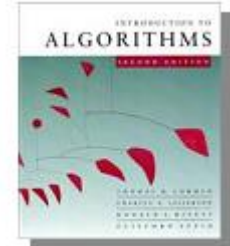
n elemanın i 'ninci küçük değerini seçin
(i *ranklı* eleman).

- $i = 1$: *minimum*; (en az)
- $i = n$: *maximum*; (en çok)
- $i = \lfloor (n+1)/2 \rfloor$ veya $\lceil (n+1)/2 \rceil$: *median*. (ortanca)

Saf algoritma: i 'ninci elemanı sırala ve dizinle.

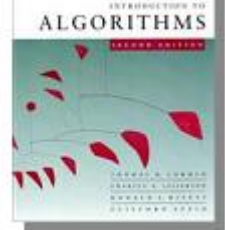
$$\begin{aligned}\text{En kötü koşma süresi} &= \Theta(n \lg n) + \Theta(1) \\ &= \Theta(n \lg n),\end{aligned}$$

birleştirme veya yığın sıralaması kullan (*çabuk sıralamayı değil*).



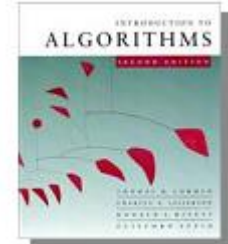
Sıra İstatistikleri

- $O(n \lg n)$ daha iyisi olabilir mi?
- Bir dizideki minimum elemanı bulmak için kaç karşılaştırma gereklidir?
- Minimum ve Maksimumu, 2 kez daha az maliyetli bulabilir miyiz?
- Evet:
 - Çiftler şeklinde ilerleyerek
 - Diğer çiftteki her bir eleman ile karşılaştır.
 - Maksimum için en büyük, minimum için en küçük elemanı karşılaştır.
 - Toplam maliyet: 2 eleman başına 3 karşılaştırma = $O(3n/2)$



Sıra istatistiklerinin Bulunması: Seçim Problemi

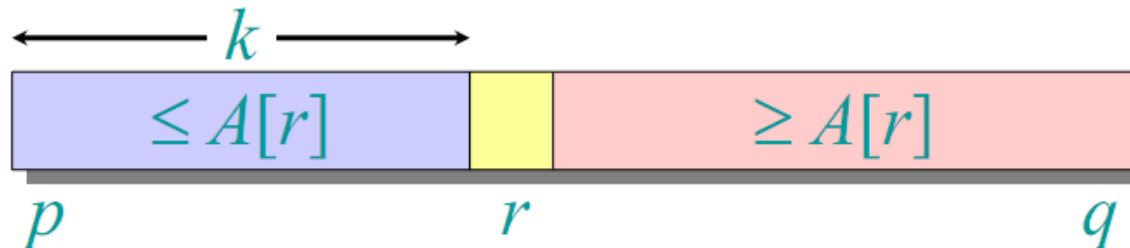
- Seçme daha ilginç problemidir: Bir dizideki i ' inci en küçük elemanı bulma. Bunun için iki algoritma;
 - Beklenen çalışma zamanı $O(n)$ olan bir pratik rastgele algoritması
 - En kötü çalışma zamanı sadece $O(n)$ ile ilgili teorik algoritma
- Anahtar Fikir: Quicksort algoritmasındaki rastgele bölüntüyü kullanmak.
 - Fakat, sadece bir altdizi incelememiz gerekir
 - Bu işlem çalışma zamanında tasarruf sağlar: $O(n)$



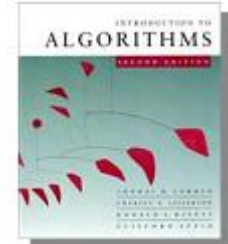
Rastgele böl-ve-fethet algoritması

```

RAND-SELECT( $A, p, q, i$ )  ▷  $A[p..q]$ 'nin  $i$ 'ninci en küçüğü
  if  $p = q$  then return  $A[p]$ 
   $r \leftarrow$  RAND-PARTITION( $A, p, q$ ) (Rastgele bölüntü)
   $k \leftarrow r - p + 1$   ▷  $k = \text{rank}(A[r])$  (rütbeli)
  if  $i = k$  then return  $A[r]$ 
  if  $i < k$ 
    then return RAND-SELECT( $A, p, r - 1, i$ )
  else return RAND-SELECT( $A, r + 1, q, i - k$ )
  
```



Örnek



$i = 7$ ' nci en küçük olarak seçin:

6	10	13	5	8	3	2	11
---	----	----	---	---	---	---	----

$i = 7$

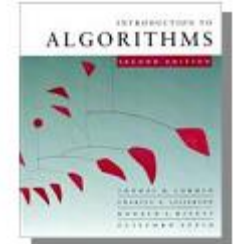
pivot (*esas eleman*)

Partition (Bölüntü):

2	5	3	6	8	13	10	11
---	---	---	---	---	----	----	----

$k = 4$

$7 - 4 = 3$ 'üncü küçüğü özyinelemeyle seçin.



Rastgele Seçme Analizi: Çözümlemede sezgi (öngörü)

- (Çözümlemelerin hepsinde tüm elemanların farklı olduğu varsayılıyor.)

Şanslı durum:

$$\begin{aligned} T(n) &= T(9n/10) + \Theta(n) \\ &= \Theta(n) \end{aligned}$$

En iyi durum (Best case):
9:1 bölüntü (partition)
olduğunu farz edin

$$n^{\log_{10/9} 1} = n^0 = 1$$

DURUM 3

Şanssız durum:

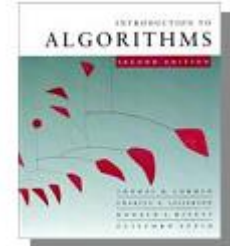
$$\begin{aligned} T(n) &= T(n-1) + \Theta(n) \\ &= \Theta(n^2) \end{aligned}$$

aritmetik seri

Sıralamadan daha kötü!

En kötü durum (Worst case):
Bölüntü daima 0:n-1

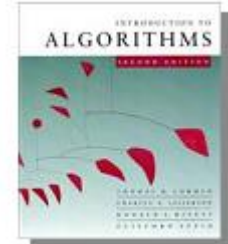
Beklenen süre çözümlemesi: (Average Case)



- Çözümleme rastgele çabuk sıralamanın benzeri ama bazı farkları var.
- $T(n)$, Rastgele-seçim çalışma zamanının rastgele değişkeni olsun (n boyutlu bir girişte), ve rastgele sayılar birbirinden bağımsız olsun.
- $k = 0, 1, \dots, n-1$ için **göstergesel rastgele değişkeni** tanımlayın.

$$X_k = \begin{cases} 1 & \text{eğer BÖLÜNTÜ } k:n-k-1 \text{ bölmeli ise,} \\ 0 & \text{diğer durumlarda.} \end{cases}$$

Beklenen süre çözümlemesi

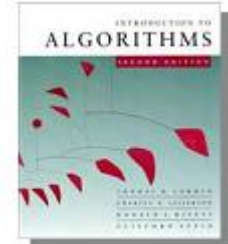


- Bir üst sınır elde etmek için, **i**' ninci elemanın her zaman bölüntünün büyük bölgesinde olduğunu varsayın:

$$T(n) = \begin{cases} T(\max\{0, n-1\}) + \Theta(n), & 0 : n-1 \text{ bölünmesi,} \\ T(\max\{1, n-2\}) + \Theta(n), & 1 : n-2 \text{ bölünmesi,} \\ \vdots \\ T(\max\{n-1, 0\}) + \Theta(n), & n-1 : 0 \text{ bölünmesi,} \end{cases}$$

$$= \sum_{k=0}^{n-1} X_k (T(\max\{k, n-k-1\}) + \Theta(n)).$$

Beklenenin hesaplanması



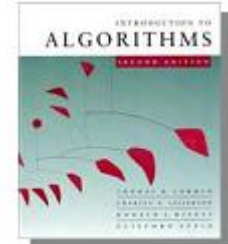
$$E[T(n)] = E\left[\sum_{k=0}^{n-1} X_k (T(\max\{k, n-k-1\}) + \Theta(n))\right]$$

Her iki taraftaki beklenenleri bulun.

$$\begin{aligned} E[T(n)] &= E\left[\sum_{k=0}^{n-1} X_k (T(\max\{k, n-k-1\}) + \Theta(n))\right] \\ &= \sum_{k=0}^{n-1} E[X_k (T(\max\{k, n-k-1\}) + \Theta(n))] \end{aligned}$$

Beklenenin doğrusallığı.

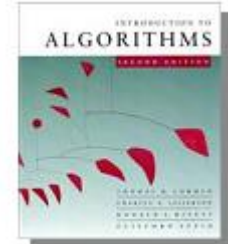
Beklenenin hesaplanması



$$\begin{aligned} E[T(n)] &= E\left[\sum_{k=0}^{n-1} X_k (T(\max\{k, n-k-1\}) + \Theta(n))\right] \\ &= \sum_{k=0}^{n-1} E[X_k (T(\max\{k, n-k-1\}) + \Theta(n))] \\ &= \sum_{k=0}^{n-1} E[X_k] \cdot E[T(\max\{k, n-k-1\}) + \Theta(n)] \end{aligned}$$

X_k 'nin diğer rastgele seçimlerden bağımsızlığı.

Beklenenin hesaplanması



$$\begin{aligned}
 E[T(n)] &= E\left[\sum_{k=0}^{n-1} X_k (T(\max\{k, n-k-1\}) + \Theta(n))\right] \\
 &= \sum_{k=0}^{n-1} E[X_k (T(\max\{k, n-k-1\}) + \Theta(n))] \\
 &= \sum_{k=0}^{n-1} E[X_k] \cdot E[T(\max\{k, n-k-1\}) + \Theta(n)] \\
 &= \frac{1}{n} \sum_{k=0}^{n-1} E[T(\max\{k, n-k-1\})] + \frac{1}{n} \sum_{k=0}^{n-1} \Theta(n)
 \end{aligned}$$

Beklenenin doğrusallığı; $E[X_k] = 1/n$.

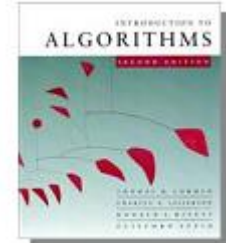
Beklenenin hesaplanması



$$\begin{aligned}
 E[T(n)] &= E\left[\sum_{k=0}^{n-1} X_k (T(\max\{k, n-k-1\}) + \Theta(n))\right] \\
 &= \sum_{k=0}^{n-1} E[X_k (T(\max\{k, n-k-1\}) + \Theta(n))] \\
 &= \sum_{k=0}^{n-1} E[X_k] \cdot E[T(\max\{k, n-k-1\}) + \Theta(n)] \\
 &= \frac{1}{n} \sum_{k=0}^{n-1} E[T(\max\{k, n-k-1\})] + \frac{1}{n} \sum_{k=0}^{n-1} \Theta(n) \\
 &\leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \Theta(n)
 \end{aligned}$$

Üstteki terimler
iki kez görünüyor.

Karmaşık yineleme



(Ama çabuk sıralamanınki kadar karmaşık değil.)

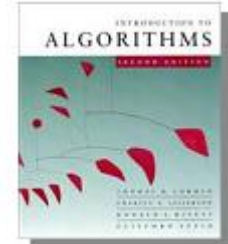
$$E[T(n)] = \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} E[T(k)] + \Theta(n)$$

Kanıtla: $E[T(n)] \leq cn$ sabiti için $c > 0$.

- c sabiti öyle büyük seçilebilir ki,
 $E[T(n)] \leq cn$ tüm taban durumlarında geçerli olur.

Veri: $\sum_{k=\lfloor n/2 \rfloor}^{n-1} k \leq \frac{3}{8}n^2$ (alıştırma).

Yerine koyma metodu



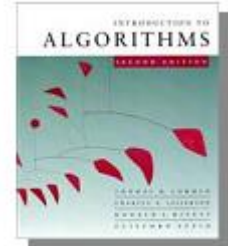
$$E[T(n)] \leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + \Theta(n)$$

Tümevarım hipotezini yerleştirin.

$$\begin{aligned} E[T(n)] &\leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + \Theta(n) \\ &\leq \frac{2c}{n} \left(\frac{3}{8} n^2 \right) + \Theta(n) \end{aligned}$$

Veriyi kullanın.

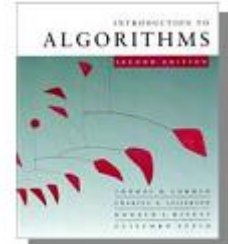
Yerine koyma metodu



$$\begin{aligned} E[T(n)] &\leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + \Theta(n) \\ &\leq \frac{2c}{n} \left(\frac{3}{8} n^2 \right) + \Theta(n) \\ &= cn - \left(\frac{cn}{4} - \Theta(n) \right) \end{aligned}$$

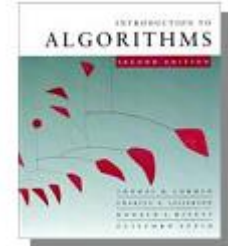
istenen – kalan şeklinde gösterin.

Yerine koyma metodu



$$\begin{aligned} E[T(n)] &\leq \frac{2}{n} \sum_{k=\lfloor n/2 \rfloor}^{n-1} ck + \Theta(n) \\ &\leq \frac{2c}{n} \left(\frac{3}{8} n^2 \right) + \Theta(n) \\ &= cn - \left(\frac{cn}{4} - \Theta(n) \right) \\ &\leq cn, \end{aligned}$$

c yeterince büyük seçilirse
 $cn/4$, $\Theta(n)$ 'nin üstünde olur.



İspat, Yerine koyma 2. yöntem

- $T(n) \leq cn$, c sabitini çok büyük seç:

$$T(n) \leq \frac{2}{n} \sum_{k=n/2}^{n-1} T(k) + \Theta(n)$$

Rekürans ile başlandı

$$\leq \frac{2}{n} \sum_{k=n/2}^{n-1} ck + \Theta(n)$$

T(k) için alt durum $T(n) \leq cn$

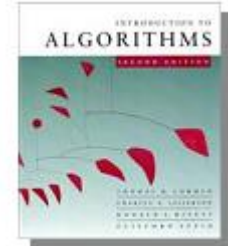
$$= \frac{2c}{n} \left(\sum_{k=1}^{n-1} k - \sum_{k=1}^{n/2-1} k \right) + \Theta(n)$$

Reküransı böl

$$= \frac{2c}{n} \left(\frac{1}{2}(n-1)n - \frac{1}{2} \left(\frac{n}{2} - 1 \right) \frac{n}{2} \right) + \Theta(n)$$

Aritmetik seriyi genişlet

$$= c(n-1) - \frac{c}{2} \left(\frac{n}{2} - 1 \right) + \Theta(n)$$



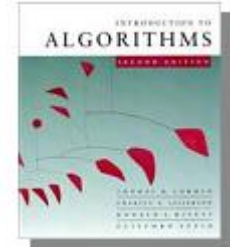
İspat, Yerine koyma 2. yöntem

- $T(n) \leq cn$, c sabitini çok büyük seç:

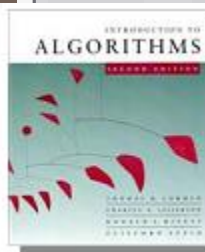
$$\begin{aligned}
 T(n) &\leq c(n-1) - \frac{c}{2} \left(\frac{n}{2} - 1 \right) + \Theta(n) \\
 &= cn - c - \frac{cn}{4} + \frac{c}{2} + \Theta(n) \\
 &= cn - \frac{cn}{4} - \frac{c}{2} + \Theta(n) \\
 &= cn - \left(\frac{cn}{4} + \frac{c}{2} - \Theta(n) \right) \\
 &\leq cn \quad (c, \text{ yeterince büyük ise})
 \end{aligned}$$

İspat

Rastgele sıra istatistik seçiminin özeti



- Hızlı çalışır: doğrusal beklenen süre.
- Pratikte mükemmel bir algoritma.
- Ama, en kötü durumu **çok** kötü: $\Theta(n^2)$.
 - **Q.** En kötü durumda doğrusal zamanda çalışan bir algoritma var mıdır?
 - **A.** Evet, Blum, Floyd, Pratt, Rivest ve Tarjan [1973] sayesinde vardır. Çok karmaşık bir algoritmadır.
 - **FİKİR:** İyi bir pivotu yinelemeye üretmek.
 - $n=100$ elaman olduğunu düşünün



En kötü durum doğrusal-zaman sıra istatistikleri

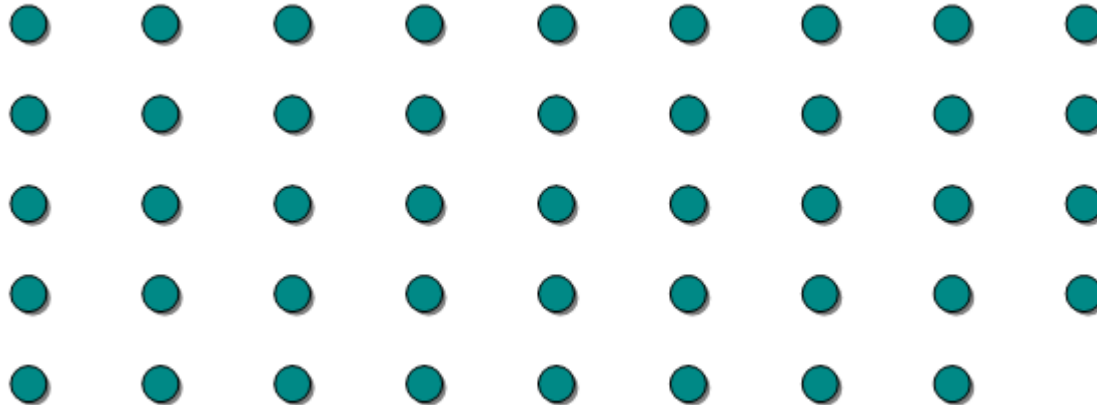
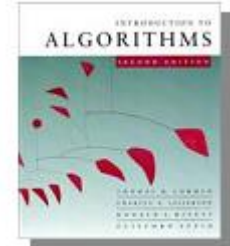
SEÇ (i, n)

1. n elemanı 5' li gruplara bölün. Her 5' li grubun ortancasını ezbere bulun.
2. $\lfloor n/5 \rfloor$ gruplarının ortancası olacak x ' i yinelemeli SEÇME ile pivot olarak belirleyin.
3. Pivot x etrafında bölüntü yapın. $k = \text{rank}(x)$.
 if $i = k$ then return x (eğer / öyleyse çıkar)
 elseif $i < k$ (diğer durumlarda)
 then i ' ninci en küçük elemanı alt bölgede yinelemeyle SEÇİN.
 else $(i-k)$ ' ninci en küçük elemanı üst bölgede yinelemeyle SEÇİN.

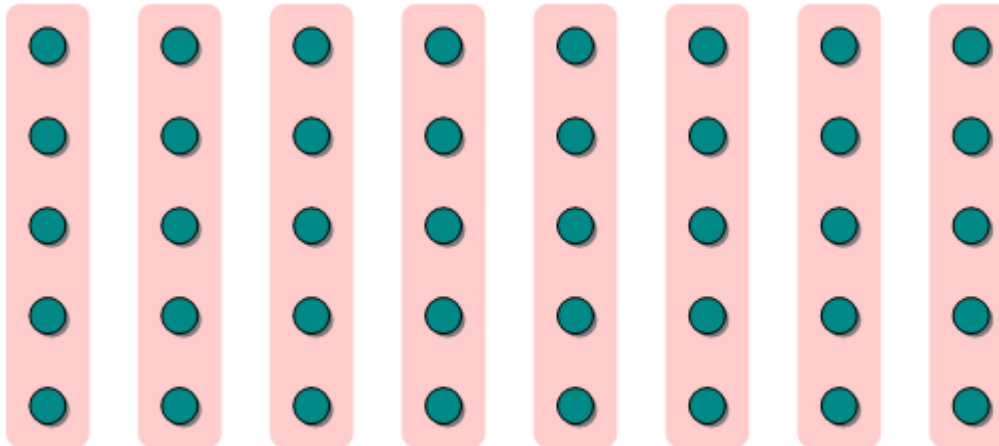
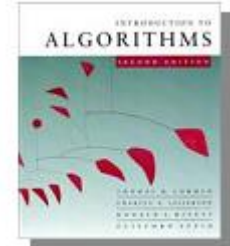
RASTGELE-
SEÇİMİN

aynısı

Pivot seçimi



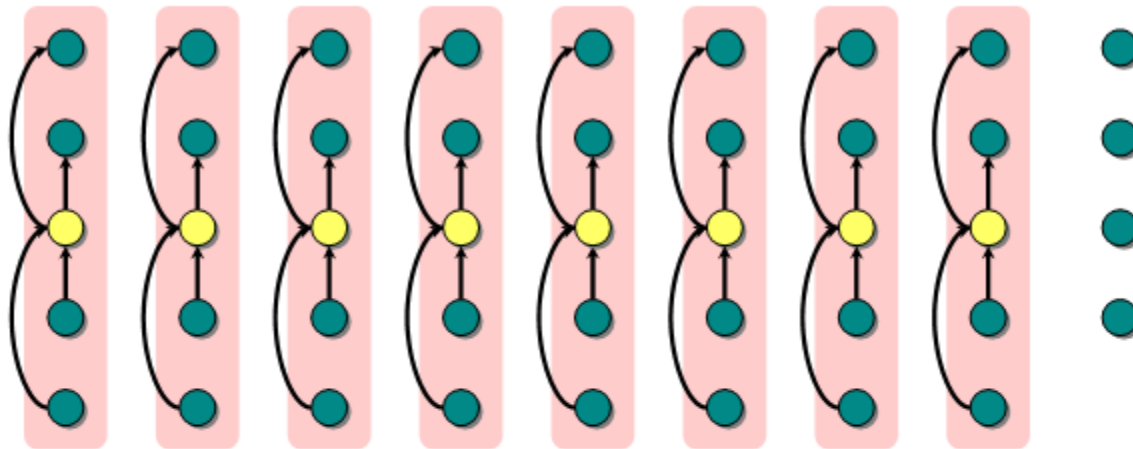
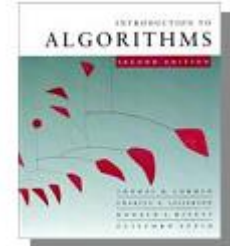
Pivot seçimi



- n her zaman tam
- bölünmeyebilir son grupta
- elemanlar eksik kalabilir bu
- durumda o sütun dikkate
- alınmaz.

1. n elemanı 5' li gruplara bölün.

Pivot seçimi

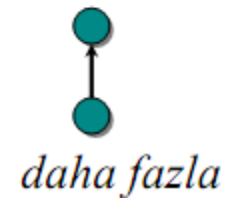


1. n elemanlarını 5'li gruplara bölün. 5-elemanlı *daha az* grupların ortancasını ezbere bulun.

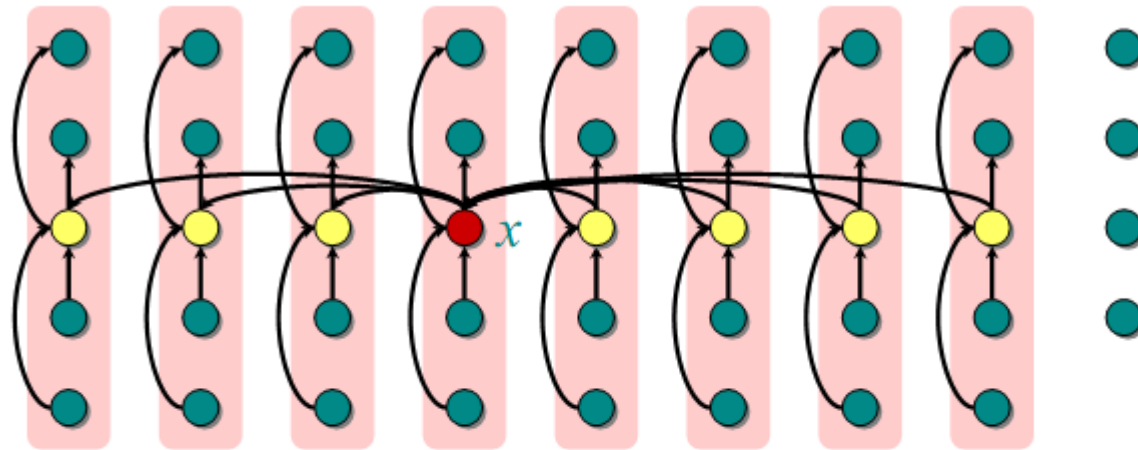
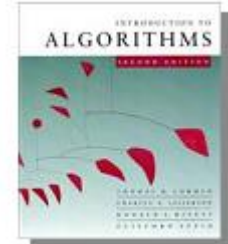
$n/5$ öbek, her birinde beş eleman var; her birinin ortancasını hesaplamak ne kadar zaman alır?

2 kere $n/5$. Yani, karşılaştırmaları sayıyorsunuz ve bu $\Theta(n)$ 'dir.

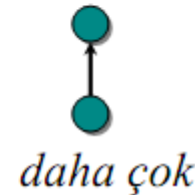
Sonuçta her grupta 5 sayı var ve sabit sayıda karşılaştırma yapılır.



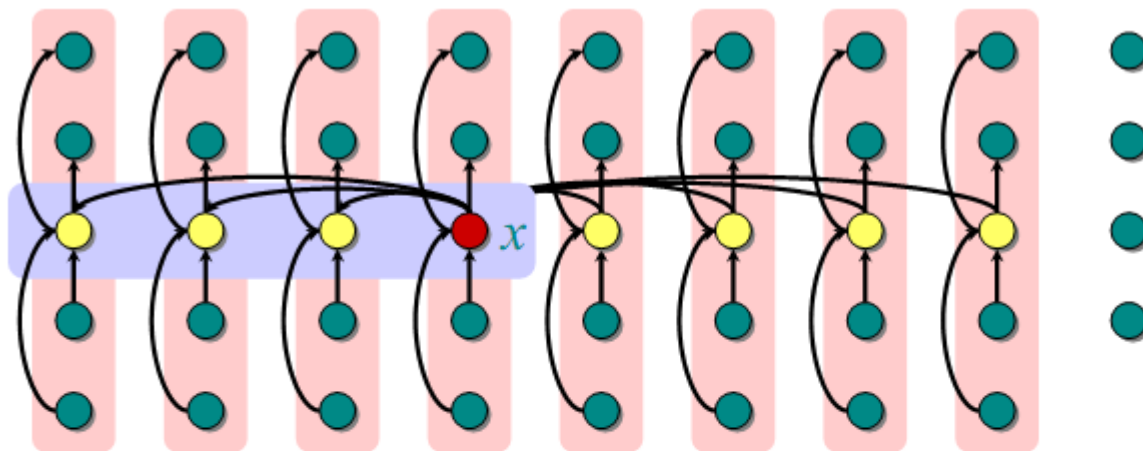
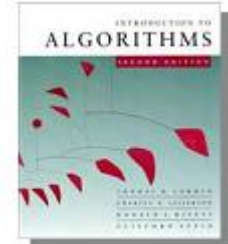
Pivot seçimi



1. n elemanlarını 5' li gruplara bölün. 5 elemanlı grupların ortancalarını ezbere bulun. *daha az*
2. $\lfloor n/5 \rfloor$ gruplarının ortancası olacak x' i, yinelemeli SEÇME ile pivot olarak belirleyin. *daha çok*



Çözümleme



Grup ortancalarının en az yarısı $\leq x$, bu da en az $\lfloor \lfloor n/5 \rfloor / 2 \rfloor = \lfloor n/10 \rfloor$ grup ortancası eder.

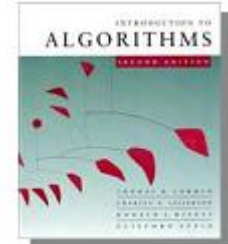
$n=100$, $n/5=20$, $20/2=10$ ortanca değer

daha az

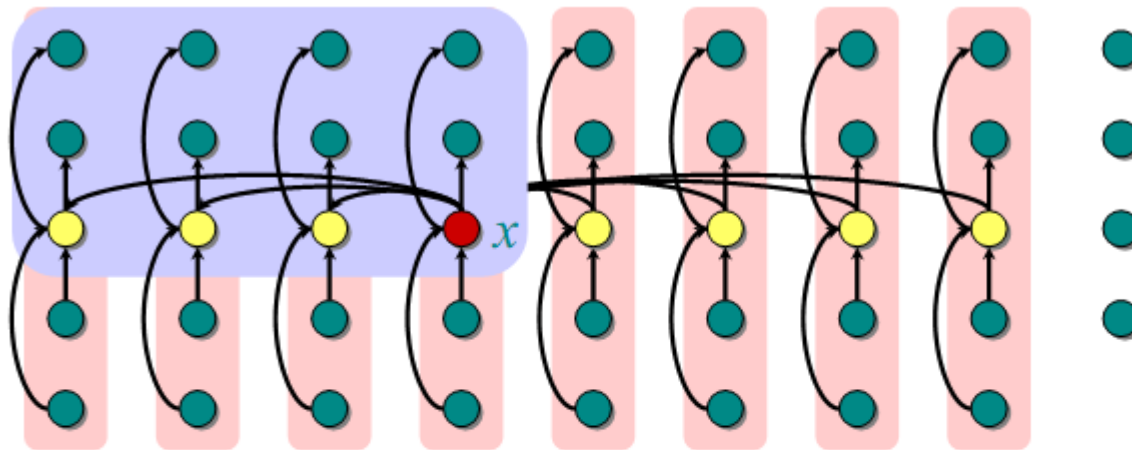


daha çok

Çözümleme (Tüm elemanları farklı varsay.)



Her grubun içinde 3 eleman var



Grup ortancalarının en az yarısı $\leq x$, bu da en az $\lfloor \lfloor n/5 \rfloor / 2 \rfloor = \lfloor n/10 \rfloor$ grup ortancası eder.

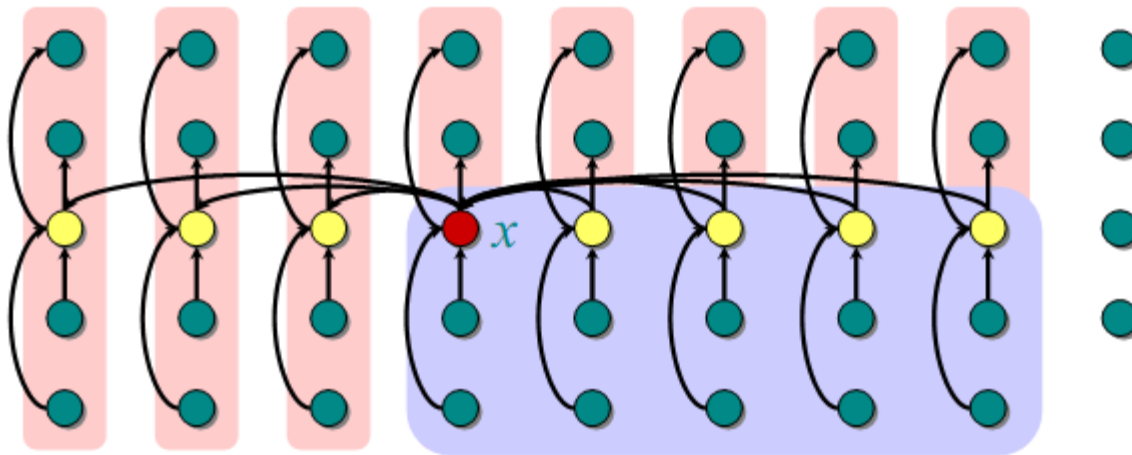
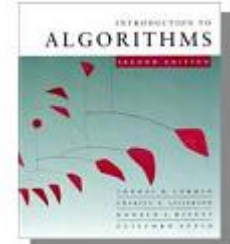
- Bu nedenle, en az $3 \lfloor n/10 \rfloor$ eleman $\leq x$.

daha az



daha çok

Çözümleme (Tüm elemanları farklı varsay.)



Grup ortancalarının en az yarısı $\leq x$, bu da en az $\lfloor \lfloor n/5 \rfloor / 2 \rfloor = \lfloor n/10 \rfloor$ grup ortancası eder.

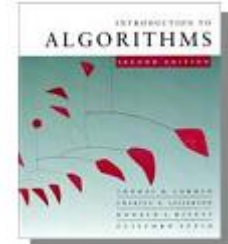
- Bu nedenle, en az $3 \lfloor n/10 \rfloor$ eleman $\leq x$.
- Benzer şekilde, en az $3 \lfloor n/10 \rfloor$ eleman $\geq x$.

daha az



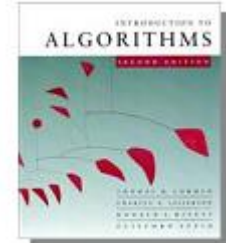
daha çok

Önemsiz basitleştirme

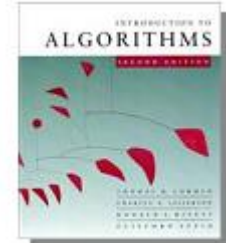


- $n \geq 50$ için, $3 \lfloor n/10 \rfloor \geq n/4$ olur.
- Bu nedenle, $n \geq 50$ için Adım 4'teki SEÇİM özyinelemeli olarak $\leq 3n/4$ eleman kapsamında yapılır. Kalan $7n/10$ alınsa da çözüm değişmez
- Böylece, koşma süresinin yinelemesinde Adım 4'ün en kötü durumda $T(3n/4)$ zamanı alacağı farz edilebilir.
- $n < 50$ için en kötü sürenin $T(n) = \Theta(1)$ olduğunu biliyoruz.

Yinelemeyi geliřtirmek



$T(n)$	<u> </u>	SELECT (i, n) (SEÇİN)
$\Theta(n)$	{	1. n elemanı 5' li gruplara ayırın. 5-elemanlı grupların ortancasını ezberden bulun.
$(n/5)$	{	2. $\lfloor n/5 \rfloor$ gruplarının ortancası olacak x ' i, yinelemeli SEÇME ile pivot olarak belirleyin.
$\Theta(n)$	{	3. Pivot x etrafında bölüntü yapın. $k = \text{rank}(x)$ olsun.
$T(3n/4)$	{	4. if $i = k$ then return x elseif $i < k$ then i ' ninci en küçük elemanı alt bölgede yinelemeli olarak SEÇİN. else $(i-k)$ ' ninci en küçük elemanı üst bölgede yinelemeli olarak SEÇİN.



Yinelemeyi çözmek

$$T(n) = T\left(\frac{1}{5}n\right) + T\left(\frac{3}{4}n\right) + \Theta(n)$$

Yerine koyma:

$$T(n) \leq cn$$

$$T(n) \leq \frac{1}{5}cn + \frac{3}{4}cn + \Theta(n)$$

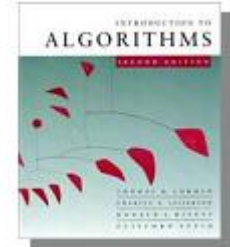
$$= \frac{19}{20}cn + \Theta(n)$$

$$= cn - \left(\frac{1}{20}cn - \Theta(n)\right)$$


$$\leq cn ,$$

c , hem $\Theta(n)$ ' i hem de başlangıç koşullarını gözeterek yeterince büyük seçilirse...

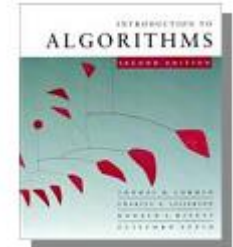
Sonuçlar



- Yinelemenin her düzeyindeki iş sabit bir kesir (**19/20**) oranında küçüldüğünden, düzeylerdeki iş bir geometrik seri gibidir ve kökteki doğrusal iş ön plana çıkar.
- Pratikte bu algoritma yavaş çalışır, çünkü **n**'nin önündeki sabit büyüktür (1 yakın bir değer, Eğer 1 olsaydı $T(n) \leq cn$ olmazdı)
- Rastgele algoritma çok daha pratiktir.
- **Alıştırma:** Neden **3**'lü gruplara bölmüyoruz?

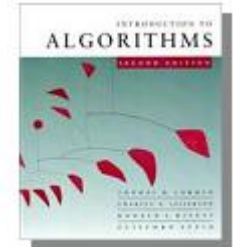


Bilinen Probleme İndirgeme Tasarım Yöntemi



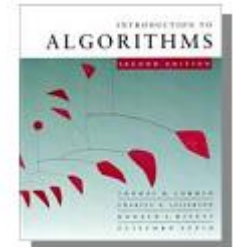
Bilinen Probleme İndirgeme

- Bu yöntemde, karmaşık olan problem çözümü yapılmadan önce problem bilinen problemlerden birine dönüştürülür ve ondan sonra bilinen problemin çözümü nasıl yapılıyorsa, bu problemin de çözümü benzer şekilde yapılır.
- Problemi bilinen bir probleme dönüştürme işlemi sofistike(yapmacık) bir işlemdir, bundan dolayı çok karmaşık problemlerde her zaman başarılı olmak mümkün olmayabilir. Belki de problem alt problemlere bölündükten sonra her alt problemin bilinen probleme dönüşümü yapılacaktır.
- Çözümü yapılmış problemlerin algoritmalarının daha etkili hale getirilmesi için de bu tasarım yöntemine başvurulabilir.



Bilinen Probleme İndirgeme

- **Örnek 1:** Verilen bir dizi ya da liste içerisinde tekrar eden sayılar var mıdır? Tekrar varsa, tekrar eden sayıdan kaç tane vardır? Birbirinden farklı kaç tane tekrar eden sayı vardır ve her birinden kaç tane vardır?
- Bu sorulara cevap vermenin farklı yolları olabilir. Bunlar içinde en etkili algoritma hangi yöntemle elde edilmişse, o çözüm en iyi çözüm olarak kabul edilir.
- **I. YOL**
- Birinci yol olarak bütün ikililer birbiri ile karşılaştırılırlar. Bu işlemi yapan algoritma;



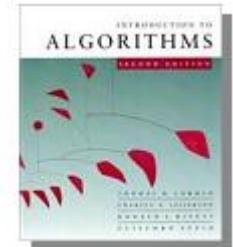
Bilinen Probleme İndirgeme

○ I. YOL

□ A parametresi içinde tekrar eden sayıların olup olmadığının kontrol edileceği dizidir ve B parametresinin i. elemanı A dizisinin i. elemanından kaç tane olduğunu tutan bir dizidir.

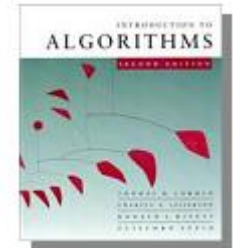
TekrarBul(A,B)

1. for $i \leftarrow 1 \dots n$
2. $B[i] \leftarrow -1$
3. for $i \leftarrow 1 \dots (n-1)$
4. if $B[i] = -1$ then
5. $B[i] \leftarrow 1$
6. for $j \leftarrow (i+1) \dots n$
7. if $A[i] = A[j]$ then
8. $B[i] \leftarrow B[i] + 1$
9. $B[j] \leftarrow 0$



Bilinen Probleme İndirgeme

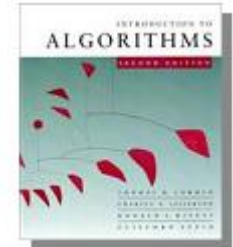
- I. YOL: En kötü çalışma zamanı
- Bütün sayılar ikili olarak karşılaştırılırlar.
- 1. sayı için $n-1$ karşılaştırma
- 2. sayı için $n-2$ karşılaştırma
-
- $(n-1)$. sayı için 1 karşılaştırma yapılır.
- Bunun sonucunda elde edilen karşılaştırma toplamaları
- $(n-1)+(n-2)+\dots+1 = \frac{n(n-1)}{2}$
- $T(n)=O(n^2)$ olur.



Bilinen Probleme İndirgeme

- I. YOL En iyi çalışma zamanı
- Bu mertebe bu algoritmanın en kötü durum analizidir ve en kötü durum A dizisi içindeki bütün sayıların farklı çıkması durumudur. Eğer A dizisi içindeki bütün sayılar aynı iseler, dış döngünün değişkeninin ilk değeri için iç döngü baştan sona kadar çalışır ve ondan sonraki değerler için iç döngü hiç çalışmaz. Bunun sonucunda en iyi durum elde edilir ve en iyi durumun mertebesi $\Theta(n)$ olur.
- $T(n) = \Theta(n)$
- Bu algoritmadan daha iyisi var mı?

Bilinen Probleme İndirgeme

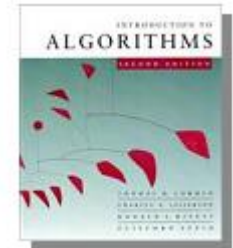


- **II. YOL: Bilinen probleme dönüştürme**
- Aynı problemin bilinen bir yöntemle çözülmesi daha iyi sonuç verebilir. Bilinen problem sıralama işlemidir. A dizisi içindeki bütün sayılar sıralanır ve ondan sonra birinci elemandan başlanarak sona doğru ardışıl olan elemanlar karşılaştırılır. Bu şekilde kaç tane tekrar olduğu bulunur.

TekrarBul(A,B)

1. **HeapSort(A)**
2. **for $j \leftarrow 1 \dots n$**
3. **$B[j] \leftarrow 1$**
4. **$i \leftarrow 1$**
5. **for $j \leftarrow 1 \dots (n-1)$**
6. **if $A[j]=A[j+1]$ then**
7. **$B[i] \leftarrow B[i]+1$**
8. **$B[j+1] \leftarrow 0$**
9. **else**
10. **$i \leftarrow j+1$**

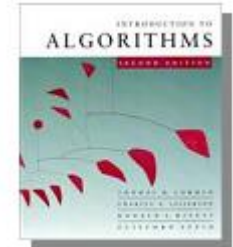
Bilinen Probleme İndirgeme



○ II. YOL

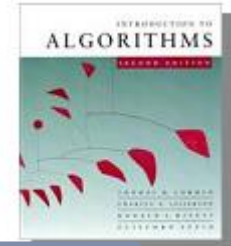
- Bu algoritma iki kısımdan oluşmaktadır. Birinci kısmı A dizisinin sıralanması ve ikinci kısımda ise bir döngü ile tekrar sayısının bulunması işlemidir. YığınSıralama algoritmasının mertebesinin $T_1(n) = \Theta(n \lg n)$ olduğu daha önceden bilinmektedir ve ikinci kısımda ise bir tane döngü olduğundan, bu kısmın mertebesi $T_2(n) = \Theta(n)$ olur. Bunun sonucunda algoritmanın zaman bağıntısı $T(n)$
- $T(n) = T_1(n) + T_2(n)$
- $= \Theta(n \lg n) + \Theta(n)$
- $= \Theta(n \lg n)$
- sınıfına ait olur. Dikkat edilirse, ikinci yol ile elde edilen çözüm birinci yol ile elde edilen çözümden daha iyidir. Bilinen probleme indirgeme yapılarak elde edilen algoritma birinci algoritmaya göre daha etkili bir algoritmadır.
- Daha iyisi var mı? Araştırma

Bilinen Probleme İndirgeme



- **Örnek 2:**
- İki boyutlu bir uzayda n tane noktadan hangi üç noktanın aynı doğru üzerinde olup olmadığı kontrolü yapılmak isteniyor. Bu problemin çözümü için en etkili algoritma nedir?
- **I. YOL**
- İlk olarak tasarlanacak olan algoritma klasik mantık olarak bütün nokta ikilileri arasındaki eğimler hesaplanır ve bu eğimler birbiri ile karşılaştırılarak hangi üç noktanın aynı doğru üzerinde olduğu belirlenir. Bu işlemi yapan algoritma;

Bilinen Probleme İndirgeme

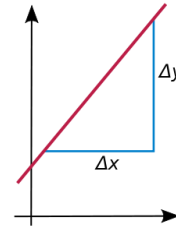


○ I. YOL

□ **A parametresi, her elemanı iki tane gerçel sayıdan oluşan bir iki boyutlu uzay noktaları kümesidir.**

Dogru_Uz_Noktalar(A)

1. for $k \leftarrow 1 \dots n$
2. for $j \leftarrow 1 \dots n$
3. for $i \leftarrow 1 \dots n$
4. if $k \neq j \neq i$ then
5. $m_1 = \text{eğim}(A[k], A[j])$
6. $m_2 = \text{eğim}(A[k], A[i])$
7. if $m_1 = m_2$ then
8. $(A[k], A[j])$ ve $(A[k], A[i])$ noktaları aynı doğru üzerindedir.

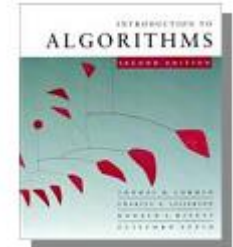


$$m = \frac{\Delta y}{\Delta x}$$

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

- Bu algoritma, iç içe üç tane döngüden oluşmaktadır ve her döngü n kez çalışmaktadır. En kötü durumda çalışma zamanı
- $T(n) = \Theta(n^3)$ olur.

Bilinen Probleme İndirgeme



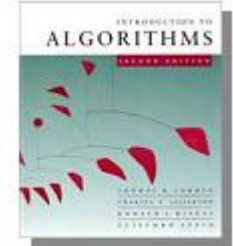
○ II. YOL

- İkinci çözüm şeklinde ise, ilk önce oluşabilecek iki nokta arasındaki eğimlerin hepsi hesaplanır. Meydana gelebilecek eğim sayısı n tane noktanın 2' li kombinasyonu olur ve eğim sayısı M olmak üzere

$$M = \binom{n}{2} = \frac{n(n-1)}{2}$$

- olur. Bundan sonraki işlem M tane eğimi sıralamaktır ve ondan sonra M tane elemanlı dizide tekrar eden elemanın olup olmadığı kontrol edilir. Bu işlemleri yapan algoritma ;

Bilinen Probleme İndirgeme



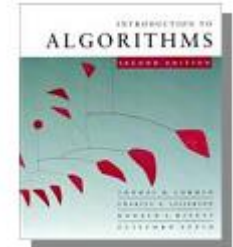
○ II. YOL :Bilinen probleme indirgeme

□ A parametresi, her elemanı iki tane gerçel sayıdan oluşan bir iki boyutlu uzay noktaları kümesidir. Bu dizideki her eleman çifti arasındaki eğim hesaplanır ve bu eğim B dizisine atılır. Ondan sonra B dizisi sıralanır ve bu dizinin tekrar eden elemanı olup olmadığı kontrol edilir.

Dogru_Uz_Noktalar(A)

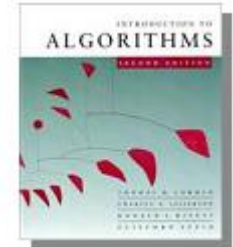
1. for $k \leftarrow 1 \dots n$
2. for $j \leftarrow (k+1) \dots n$
3. $m = \text{eğim}(A[k], A[j])$
4. $B[i] = m$
5. $i \leftarrow i+1$
6. HeapSort(B, M)
7. for $j \leftarrow 1 \dots M$
8. $C[j] \leftarrow 1$
9. $i \leftarrow 1$
10. for $j \leftarrow 1 \dots M-1$
11. if $B[j] = B[j+1]$
12. $C[i] \leftarrow C[i] + 1$
13. $C[j+1] \leftarrow 0$
14. else
15. $i \leftarrow j+1$

Bilinen Probleme İndirgeme



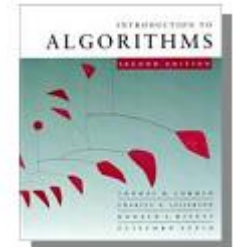
II. YOL

- C dizisindeki elemanlar kendisi ile aynı endekse sahip B dizisinin o elemanından kaç tane olduğunu tutmaktadır.
- Bu algoritmanın mertebesi hesaplanacak olursa, algoritmada üç parçadan oluşan bir zaman bağıntısı elde edilir.
- İlk parça eğimleri hesaplama zamanı ve bu zaman $T_1(n)$ olsun.
- İkinci parça B dizisini sıralama zamanı ve bu zaman $T_2(n)$ olsun.
- Son parçada ise sıralı B dizisi içinde tekrar eden eleman olup olmadığını kontrol etme zamanıdır ve bu zaman $T_3(n)$ olsun.
- Bu zamanlar
 - $T_1(n) = \Theta(n^2)$
 - $T_2(n) = \Theta(M \lg M) = \Theta(n^2 \lg n)$
 - $T_3(n) = \Theta(M) = \Theta(n^2)$
- Algoritmanın mertebesi $T(n) = T_1(n) + T_2(n) + T_3(n) = \Theta(n^2 \lg n)$ olur.



Bilinen Probleme İndirgeme

- **Örnek 3:** $n \times n$ boyutlarında kare matrislerin çarpımı. Klasik yöntemle $O(n^3)$ çarpma ve $O(n^3)$ toplama vardır.
- **Çözüm:** Strassen'in fikri daha önce değinilmişti.
- **Örnek 4:** Bir kümenin maksimum ve minimum elemanlarının belirlenmesi için gerekli algoritmanın kaba kodunu yazınız.



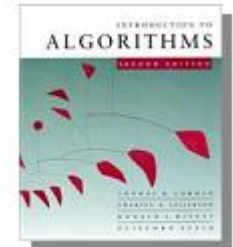
Bilinen Probleme İndirgeme

- Uygulama çözüm:
- I. Yol: İlk olarak $n-1$ karşılaştırma yapılarak maksimum bulunur ve $n-2$ karşılaştırma yapılarak minimum bulunur. Buradan $T(n)=2n-3$ olur ve Çalışma zamanı $T(n)=O(n)$ olur.

MAXIMUM-MINIMUM(A)

```
MAXIMUM-MINIMUM(A)
1  max ← min ← A[1]
2  for i ← 2 to length[A]
3      do if A[i] > max
4          then max ← A[i]
5          else if A[i] < min
6              then min ← A[i]
7  return min & max
```

- Daha iyisi olan bir algoritma tasarlayıp çalışma zamanını bulunuz?



Bilinen Probleme İndirgeme

○ II.Yol Çözüm:

- Eğer n sayısı çift ise($\lg n$ sayısının katı): a) İlk olarak $n/2$ çift elamanlar bulunur.

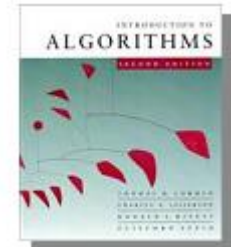


- Daha sonra her bir çift karşılaştırılır. $\lfloor n/2 \rfloor$, $\lceil n/2 \rceil$, çiftler arasında en fazla 3 karşılaştırma yapılır.



● = larger

● = smaller



Bilinen Probleme İndirgeme

- n çift ise $T(n) = (3/2)n - 2$ olur.

$$T(n) = \begin{cases} 0 & n=1 \\ 1 & n=2 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 2 & \text{olur} \end{cases}$$

(Eğer n sayısı tek ise: $3(n-1)/2$ olur.)

Algoritma(S)

if $|S|=1$ or $|S|=2$ then bir karşılaştırma yapılır

elseif $|S|>2$ then

$S = S1 \cup S2$

$(\min1, \max1) \leftarrow \text{MaxMin}(S1)$

$(\min2, \max2) \leftarrow \text{MaxMin}(S2)$

if $\min1 \leq \min2$ then sonuç($\min = \min1$)

else sonuç($\min = \min2$)

if $\max \geq \max2$ then sonuç ($\max = \max1$)

else sonuç ($\max = \max2$) $T(n) = O(n)$

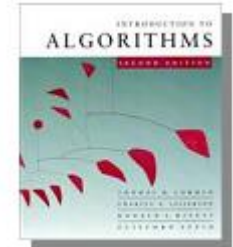
n için Sıkı sınır

$$T(n) = \lceil 3n/2 \rceil - 2$$

```

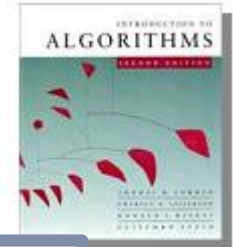
MAXMIN(i, j, fmax, fmin)
1 if (i=j)
2   then fmax ← fmin ← a[i]
3 if (i=(j-1)) then
4   if a[i]<a[j]
5     then fmax ← a[j]
6         fmin ← a[i]
7   else fmax ← a[i]
8         fmin ← a[j]
9 else
10   mid ← ⌊(i+j)/2⌋
11   MAXMIN(i, mid, gmax, gmin)
12   MAXMIN(mid+1, j, hmax, hmin)
13   fmax ← max{gmax, hmax}
14   fmin ← min{gmin, hmin}
  
```

Bilinen Probleme İndirgeme



- **Örnek 4:**
- Bir binanın güvenlik işlemleri kamera tertibatı ile yapılmak isteniyor ve kurulacak olan kamera sistemi, en az sayıda kamera içerecek ve binada görüş alanı dışında da yer kalmayacak şekil olacaktır. Bu problem nasıl çözülür?
- **Çözüm**
- İlk olarak problemin bilinen bir probleme dönüştürülmesi gerekir. Binada kirişler ve kolonlar ayırıt olarak düşünüldüğünde, kiriş ve kolonların birleştiği noktalar da düğüm olarak düşünülebilir. Bu şekilde binanın çizgesi çıkarılmış olur. Binaya yerleştirilecek kameraların görmediği kiriş veya kolon kalmamalı. Kiriş ve kolonlar ayırıt olduklarına göre çözüm minimum-düğüm kapsama probleminin çözümü olur. Binayı modelleyen çizge $G=(V,E)$ olmak üzere problemin çözümü aşağıdaki algoritma ile yapılır.
- (Graflara sonra değinilecektir)

Bilinen Probleme İndirgeme



○ Çözüm

□ **C kümesi hangi köşelere kamera konulacaksa, o köşeleri temsil eden düğümleri içerir.**

Düğüm_Kapsama(G)

1. $C \leftarrow \emptyset$
2. $E' \leftarrow E$
3. $E' \neq \emptyset$ olduğu sürece devam et
4. $(u,v) \in E'$ olan bir ayrit seç ve
5. $C \leftarrow C \cup \{u,v\}$
6. E' kümesinde u veya v düğümüne çakışık olan ayritların hepsini sil.

6.Hafta

Kıyım Fonksiyonu (Hashing),İkili Arama Ağaçları (BST)

Rastgele yapılanmış ikili
arama ağaçları

- Beklenen düğüm derinliği
- Yüksekliği çözümlmek