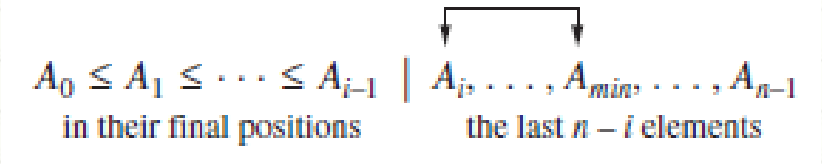
Kaba Kuvvet Algoritmaları

- **Kaba Kuvvet**
  - Bir problemi çözmek için en basit yaklaşım
  - Genellikle problemin tanımına ve konseptine bağımlıdır
  - Genellikle uygulaması en basit çözümdür

- **En temel örnekler**
  - a^n hesaplanması
  - n! Hesaplanması
  - İki matrisin çarpımı
  - Bir dizide bir elemanın aranması

Seçimli Sıralama

- Seçimli Sıralama *(Selection Sort)*
  - Dizinin içerisindeki en küçük eleman bulunur.
  - 1. sıradaki elemanla yer değiştirilir.
  - En küçük bulma işlemi dizinin ikinci elemanından başlanılarak tekrar edilir.
  - Bulunan en küçük değer 2.sıradaki elemanla yer değiştirir.
  - Bu işlem dizinin son elemanına kadar devam eder.



ALGORITHM *SelectionSort*$(A[0..n-1])$

```
//Sorts a given array by selection sort
//Input: An array A[0..n − 1] of orderable elements
//Output: Array A[0..n − 1] sorted in nondecreasing order
for i ← 0 to n − 2 do
    min ← i
    for j ← i + 1 to n − 1 do
        if A[j] < A[min]  min ← j
    swap A[i] and A[min]
```

Seçimli Sıralama

| 89	45	68	90	29	34	**17**
17 |	45	68	90	**29**	34	89
17	29 |	68	90	45	**34**	89
17	29	34 |	90	**45**	68	89
17	29	34	45 |	90	**68**	89
17	29	34	45	68 |	90	**89**
17	29	34	45	68	89 |	90

Seçimli Sıralama

ALGORITHM *SelectionSort*$(A[0..n-1])$

```
//Sorts a given array by selection sort
//Input: An array A[0..n − 1] of orderable elements
//Output: Array A[0..n − 1] sorted in nondecreasing order
for i ← 0 to n − 2 do
    min ← i
    for j ← i + 1 to n − 1 do
        if A[j] < A[min]  min ← j
    swap A[i] and A[min]
```

- Temel işlemin gerçekleşme sayısı formülü:

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} [(n-1) - (i+1) + 1] = \sum_{i=0}^{n-2} (n-1-i).$$

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} (n-1-i) = \frac{(n-1)n}{2}. \quad \Theta(n^2)$$

Kabarcık Sıralama

ALGORITHM *BubbleSort*$(A[0..n-1])$

//Sorts a given array by bubble sort

//Input: An array $A[0..n-1]$ of orderable elements

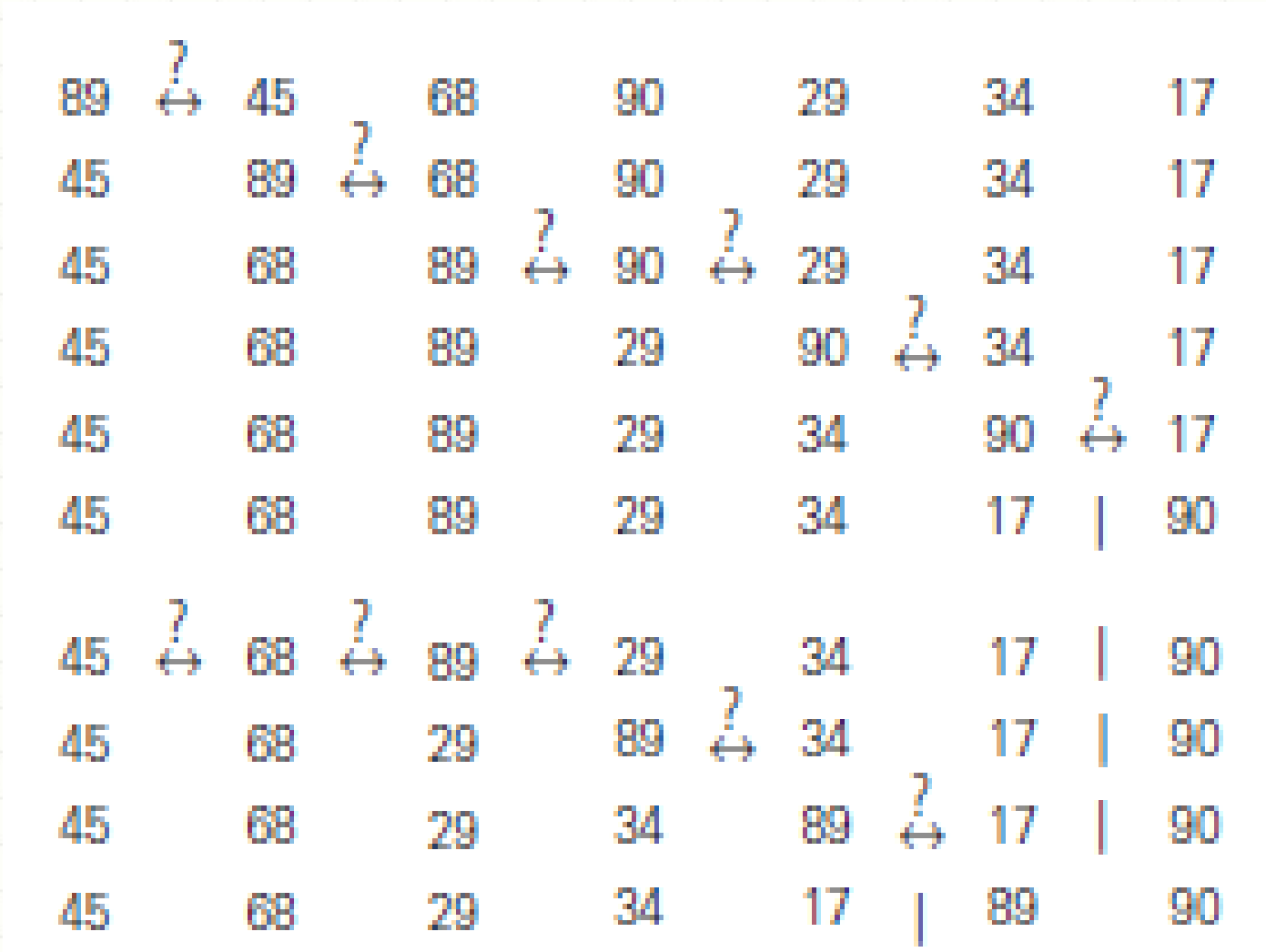
//Output: Array $A[0..n-1]$ sorted in nondecreasing order

for $i \leftarrow 0$ **to** $n-2$ **do**

  for $j \leftarrow 0$ **to** $n-2-i$ **do**

    if $A[j+1] < A[j]$ swap $A[j]$ and $A[j+1]$

Kabarcık Sıralama



Kabarcık Sıralama

ALGORITHM *BubbleSort*$(A[0..n-1])$

//Sorts a given array by bubble sort
//Input: An array $A[0..n-1]$ of orderable elements
//Output: Array $A[0..n-1]$ sorted in nondecreasing order
for $i \leftarrow 0$ **to** $n-2$ **do**
  for $j \leftarrow 0$ **to** $n-2-i$ **do**
    if $A[j+1] < A[j]$ swap $A[j]$ and $A[j+1]$

- Temel İşlem :?
- Gerçekleşme Sayısı :?

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=0}^{n-2-i} 1 := \sum_{i=0}^{n-2} [(n-2-i) - 0 + 1]$$

$$= \sum_{i=0}^{n-2} (n-1-i) = \frac{(n-1)n}{2} \in \Theta(n^2).$$

Kaba Kuvvet Arama Algoritmaları

- Sıralı Arama *(Sequential Search)*
- Kaba Kuvvet String Eşleme *(Bruteforce String Matching)*

Sıralı Arama

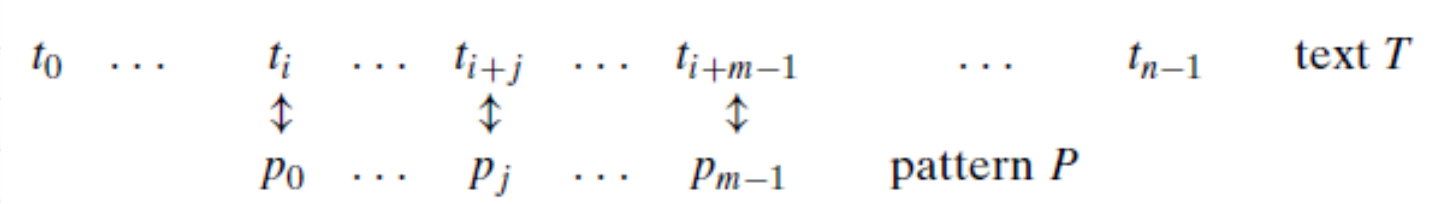
- Aranan elemanı dizi elemanları ile tek tek karşılaştırarak arama yapar
- Aranan elemanın dizi içerisinde (varsa) bulunduğu ilk indisi verir

ALGORITHM *SequentialSearch2*$(A[0..n], K)$

```
//Implements sequential search with a search key as a sentinel
//Input: An array A of n elements and a search key K
//Output: The index of the first element in A[0..n − 1] whose value is
//        equal to K or −1 if no such element is found
A[n] ← K
i ← 0
while A[i] ≠ K do
        i ← i + 1
if i < n return i
else return −1
```

Kaba Kuvvet String Eşleme

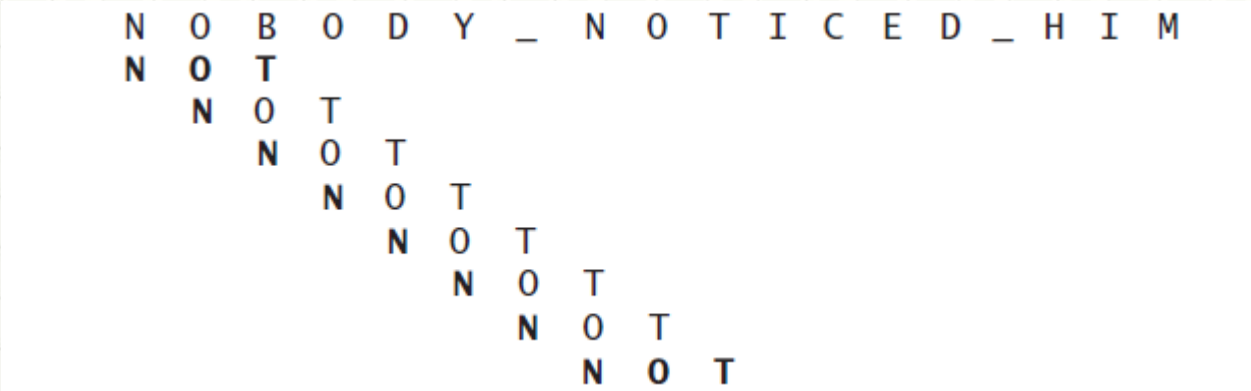
- n uzunluğundaki T metni içerisinde m uzunluğundaki P örüntüsünü arama



ALGORITHM *BruteForceStringMatch*$(T[0..n-1], P[0..m-1])$

```
//Implements brute-force string matching
//Input: An array T[0..n − 1] of n characters representing a text and
//       an array P[0..m − 1] of m characters representing a pattern
//Output: The index of the first character in the text that starts a
//        matching substring or −1 if the search is unsuccessful
for i ← 0 to n − m do
    j ← 0
    while j < m and P[j] = T[i + j] do
        j ← j + 1
    if j = m return i
return −1
```

Kaba Kuvvet String Eşleme



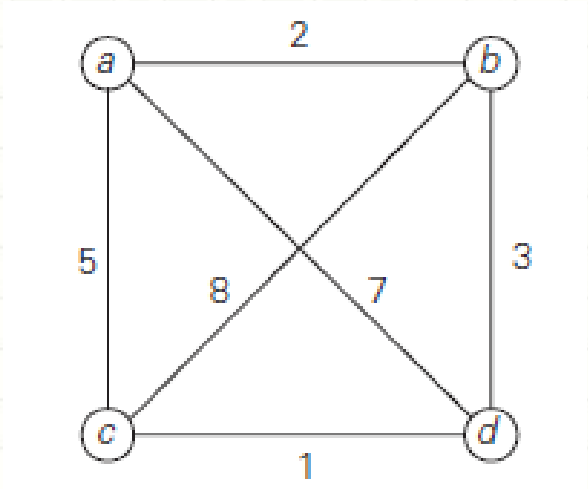
- Kötü durum
  - Aranan örüntünün bulunamaması
    - Kaydırmanın örüntünün son karakteri karşılaştırıldıktan sonra yapılabilmesi
  - Her seferinde (n-m+1 kez) 3. karakter karşılaştırmasında uyumsuzluk çıkması
    - Bu durumda sınıfı : ***O(nm)***
  - Ortalama durum:
    - Bir doğal dil için birkaç karşılaştırmada uyumsuzluk yakalanır

Etraflı Arama

- **Etraflı Arama (Exhaustive Search)**
  - Kombinasyonel çözümler içerisinden belli bir özelliğe sahip olanı arama için kullanılan bir kaba kuvvet çözümü
  - Her ihtimal denenerek arama yapılır
    - Gezgin Satıcı Problemi *(Traveling Salesman)*
    - Sırt Çantası Problemi *(Knapsack Problem)*
    - İşe Alma Problemi *(Assignment Problem)*

Gezgin Satıcı Problemi

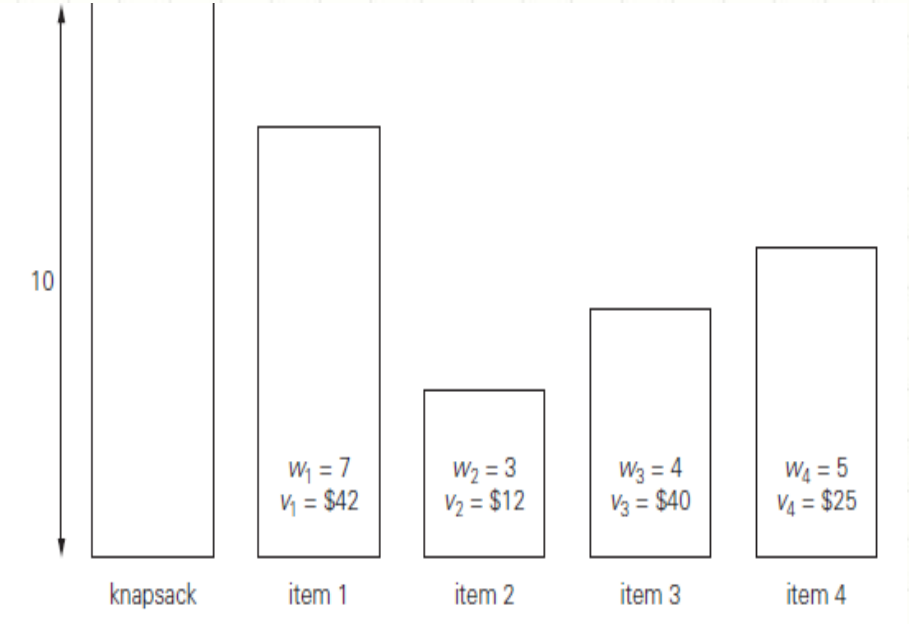
- Bir satıcının
  - Aralarındaki mesafeler bilinen şehirleri
    - Her şehirden bir kez geçerek başladığı şehre en kısa yoldan dönmesi
    - Hamiltonian Circuit
    - (n-1)!/2



Tour	Length	
$a \rightarrow b \rightarrow c \rightarrow d \rightarrow a$	$l = 2 + 8 + 1 + 7 = 18$	
$a \rightarrow b \rightarrow d \rightarrow c \rightarrow a$	$l = 2 + 3 + 1 + 5 = 11$	optimal
$a \rightarrow c \rightarrow b \rightarrow d \rightarrow a$	$l = 5 + 8 + 3 + 7 = 23$	
$a \rightarrow c \rightarrow d \rightarrow b \rightarrow a$	$l = 5 + 1 + 3 + 2 = 11$	optimal
$a \rightarrow d \rightarrow b \rightarrow c \rightarrow a$	$l = 7 + 3 + 8 + 5 = 23$	
$a \rightarrow d \rightarrow c \rightarrow b \rightarrow a$	$l = 7 + 1 + 8 + 2 = 18$	

Sırt Çantası Problemi

- Bir hırsızın çalacağı nesnelere karar verme problemi
  - Çantasına sığmalı
  - Değeri mümkün olduğunca yüksek olmalı



Subset	Total weight	Total value
∅	0	$ 0
{1}	7	$42
{2}	3	$12
{3}	4	$40
{4}	5	$25
{1, 2}	10	$54
{1, 3}	11	not feasible
{1, 4}	12	not feasible
{2, 3}	7	$52
{2, 4}	8	$37
{3, 4}	**9**	**$65**
{1, 2, 3}	14	not feasible
{1, 2, 4}	15	not feasible
{1, 3, 4}	16	not feasible
{2, 3, 4}	12	not feasible
{1, 2, 3, 4}	19	not feasible

İşe Alma Problemi

- n aday n pozisyon için işe başvuruyor
  - Her aday her farklı pozisyon için farklı maaş talep ediyor
  - Her pozisyona 1 kişi alınacak
  - Her aday işe alınacak
  - Toplam maaş maliyeti minimum olacak

	Job 1	Job 2	Job 3	Job 4
Person 1	9	2	7	8
Person 2	6	4	3	7
Person 3	5	8	1	8
Person 4	7	6	9	4

İşe Alma Problemi

	Job 1	Job 2	Job 3	Job 4
Person 1	9	2	7	8
Person 2	6	4	3	7
Person 3	5	8	1	8
Person 4	7	6	9	4

$$C = \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix}$$

<1, 2, 3, 4> cost = 9 + 4 + 1 + 4 = 18
<1, 2, 4, 3> cost = 9 + 4 + 8 + 9 = 30
<1, 3, 2, 4> cost = 9 + 3 + 8 + 4 = 24
<1, 3, 4, 2> cost = 9 + 3 + 8 + 6 = 26
<1, 4, 2, 3> cost = 9 + 7 + 8 + 9 = 33
<1, 4, 3, 2> cost = 9 + 7 + 1 + 6 = 23

etc.