

# Τεχνικές Σχεδίασης Αλγορίθμων

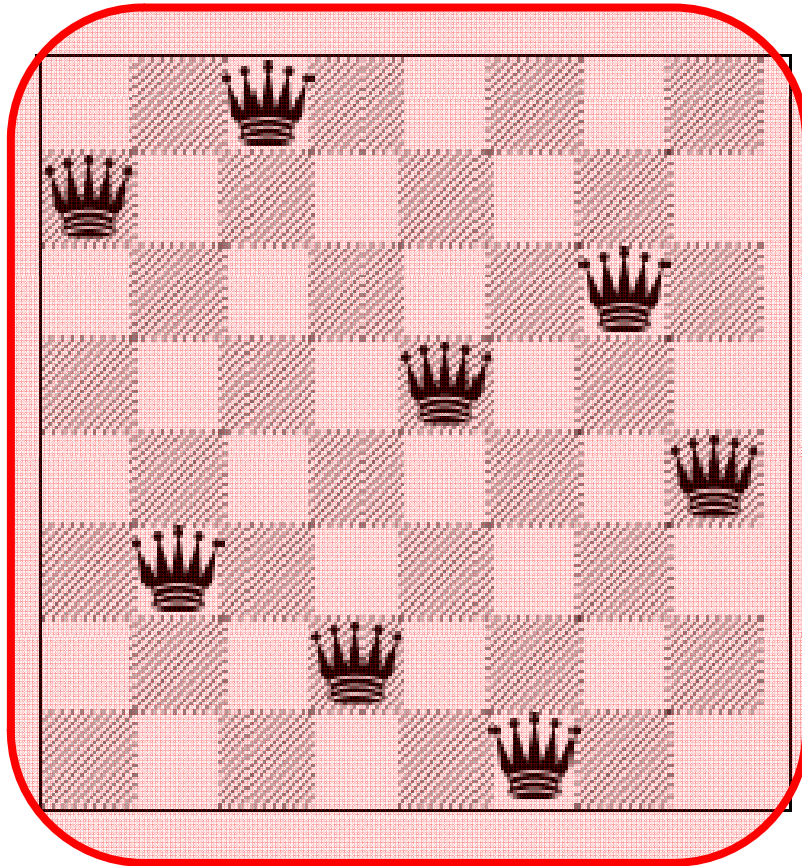
- Divide and Conquer  
(Merge Sort, QSort,...)
- Greedy algorithms  
(Prim, Kruskal, Huffman,...)
- Dynamic Programming  
(Bellman, LCS,...)
- Backtracking  
(κ-Queens, TSP,...)

# Πλήρης Αναζήτηση (Exhaustive search)

Πλήρης αναζήτηση = Απαρίθμηση όλων των λύσεων!

Πλήρης απαρίθμηση:  
 **$n$  Queens**

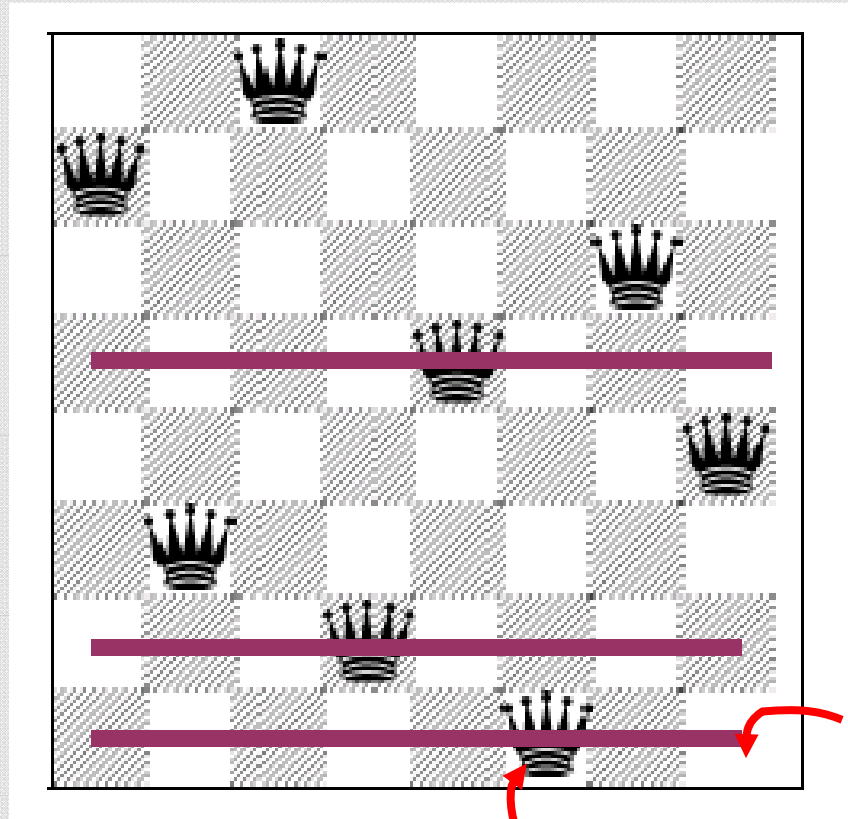
$n$  βασίλισσες σε  $n \times n$  σκακιέρα



instance: 8 βασίλισσες σε 8 x 8  
σκακιέρα

Πλήρης απαρίθμηση:  
**n Queens**

8 βασίλισσες σε 8 x 8 σκακιέρα



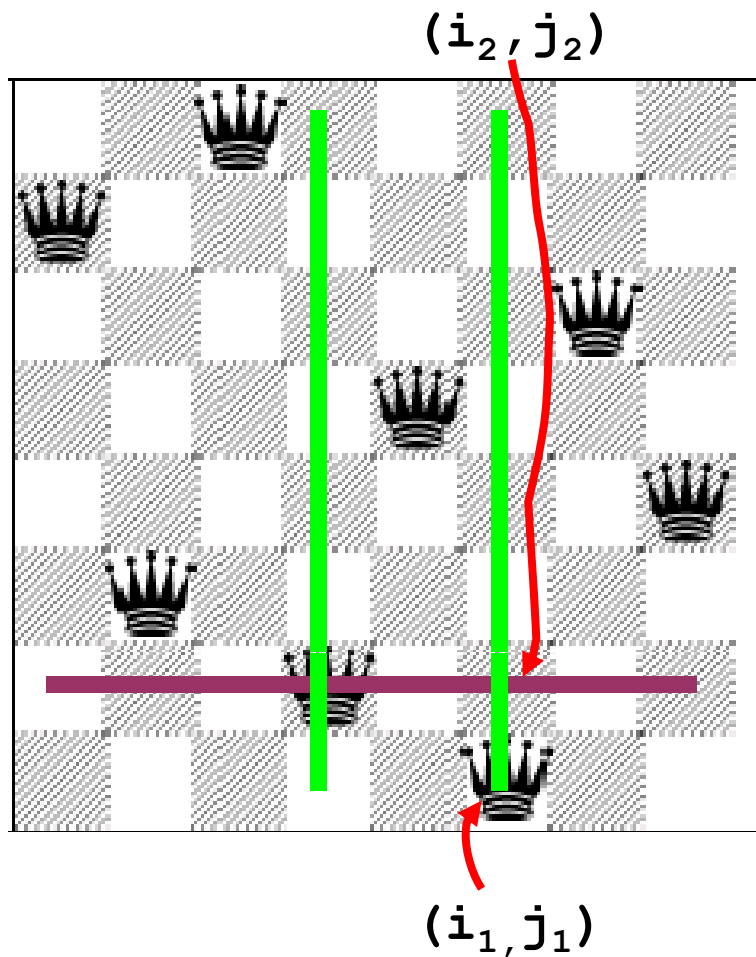
$(i_1, j_1)$

$(i_2, j_2)$

**Conflict**  $(i_1, j_1)$  and  $(i_2, j_2)$  :  
 $(i_1 = i_2)$

Πλήρης απαρίθμηση:  
**n Queens**

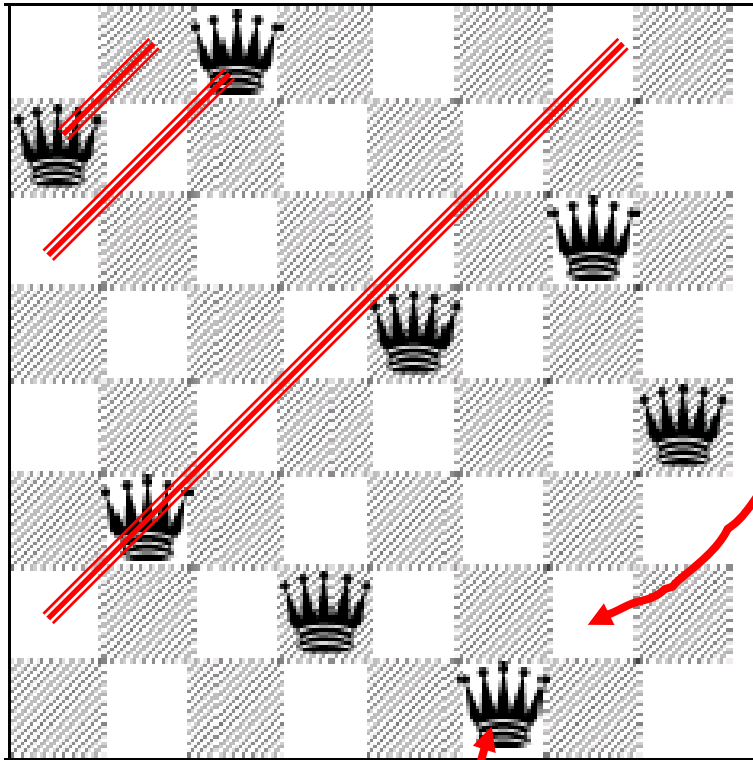
8 βασίλισσες σε 8 x 8 σκακιέρα



Conflict  $(i_1, j_1)$  and  $(i_2, j_2)$  :  
 $(i_1 = i_2)$  OR  
 $(j_1 = j_2)$

Πλήρης απαρίθμηση:  
**n Queens**

8 βασίλισσες σε 8 x 8 σκακιέρα



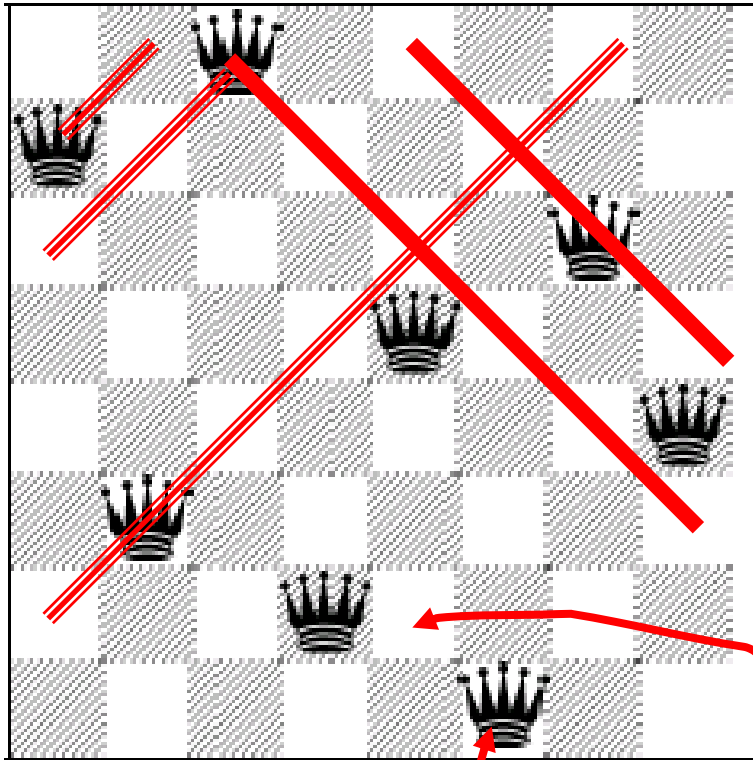
$(i_2, j_2)$

**Conflict**  $(i_1, j_1)$  and  $(i_2, j_2)$  :  
 $(i_1 = i_2)$  OR  
 $(j_1 = j_2)$  OR  
 $(\text{abs}(i_1 - i_2) = \text{abs}(j_1 - j_2))$

$(i_1, j_1)$

Πλήρης απαρίθμηση:  
**n Queens**

8 βασίλισσες σε 8 x 8 σκακιέρα



$(i_1, j_1)$

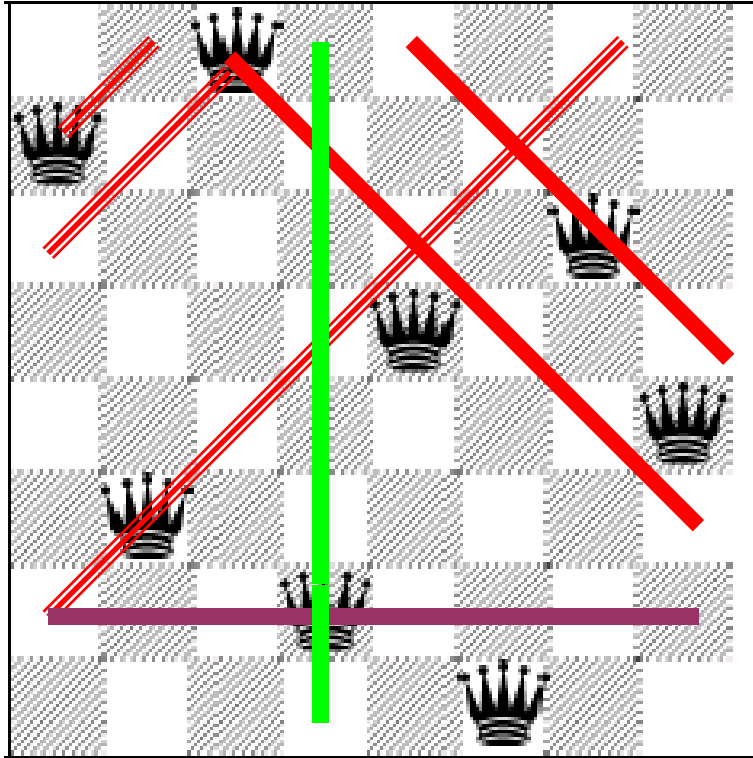
$(i_2, j_2)$

Conflict  $(i_1, j_1)$  and  $(i_2, j_2)$  :  
 $(i_1 = i_2)$  OR  
 $(j_1 = j_2)$  OR  
 $(\text{abs}(i_1 - i_2) = \text{abs}(j_1 - j_2))$

Πλήρης απαρίθμηση:  
**n Queens**

8 βασίλισσες σε 8 x 8 σκακιέρα

n βασίλισσες σε n x n σκακιέρα



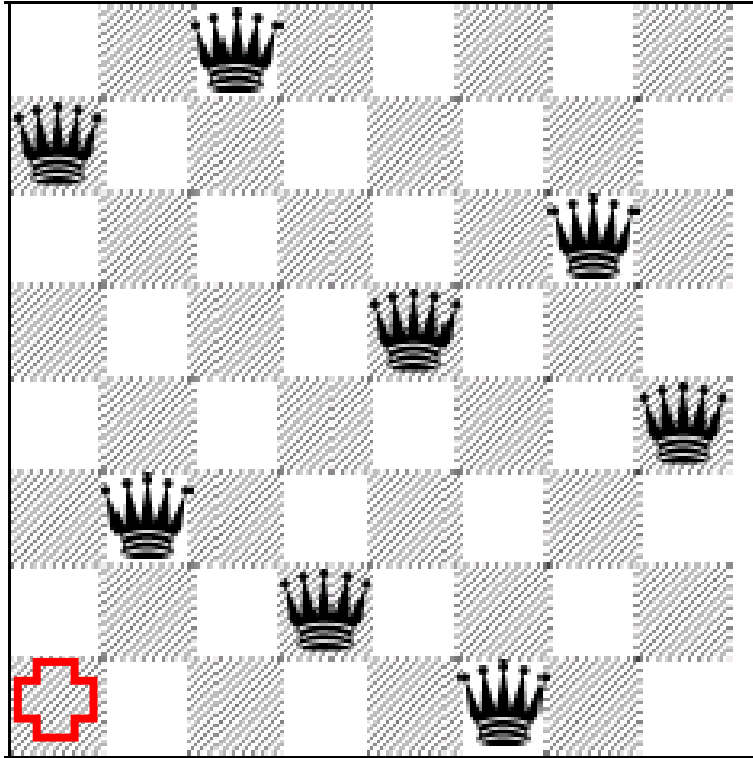
**Conflict**  $(i_1, j_1)$  and  $(i_2, j_2)$  :  
 $(i_1 = i_2)$  OR  
 $(j_1 = j_2)$  OR  
 $(\text{abs}(i_1 - i_2) = \text{abs}(j_1 - j_2))$



Πλήρης απαρίθμηση:  
**n Queens**

n βασίλισσες σε n x n σκακιέρα

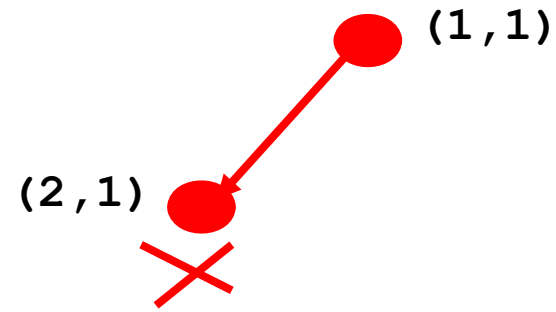
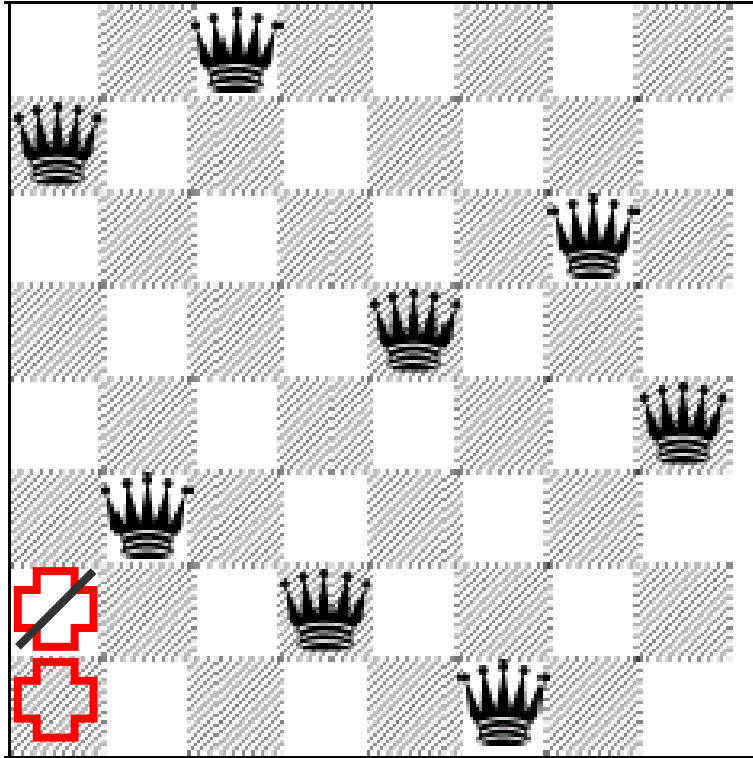
● (1,1)



(1, Pos[1])

Πλήρης απαρίθμηση:  
 **$n$  Queens**

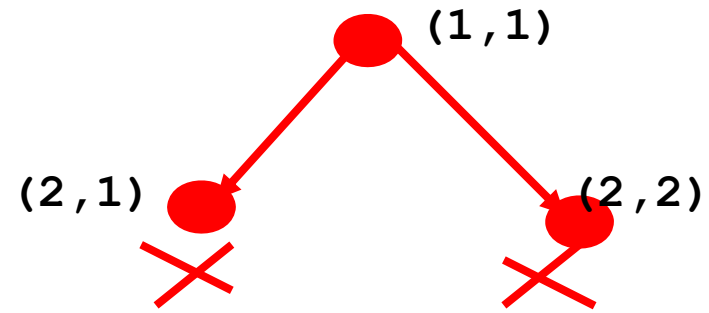
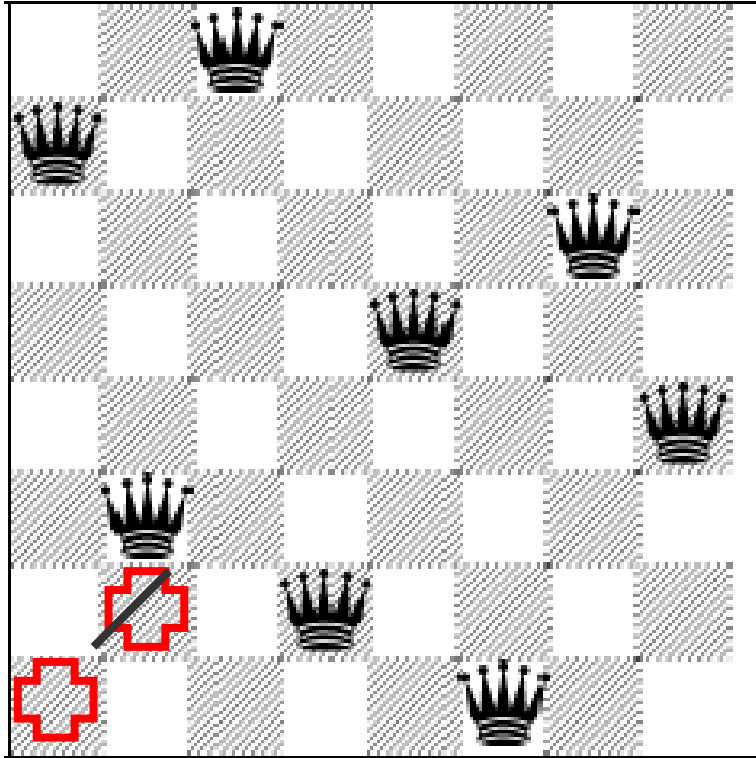
$n$  βασίλισσες σε  $n \times n$  σκακιέρα



$(1, \text{Pos}[1]), (2, \text{Pos}[2])$

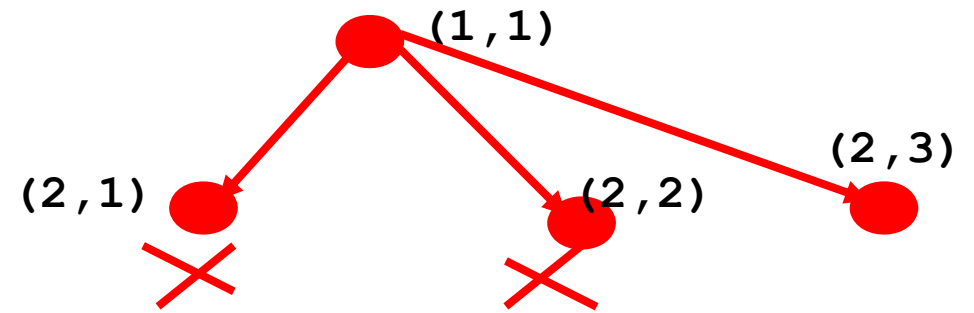
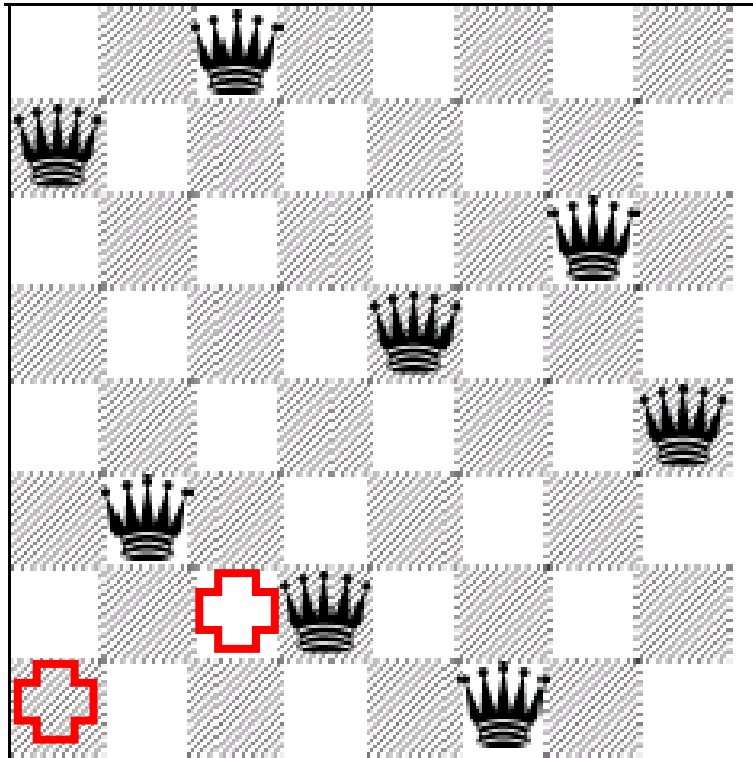
Πλήρης απαρίθμηση:  
 **$n$  Queens**

$n$  βασίλισσες σε  $n \times n$  σκακιέρα



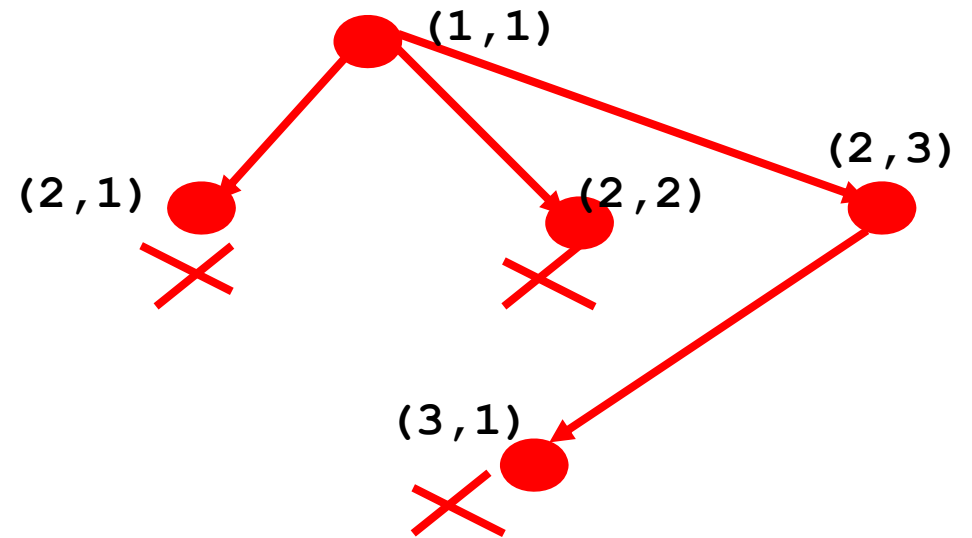
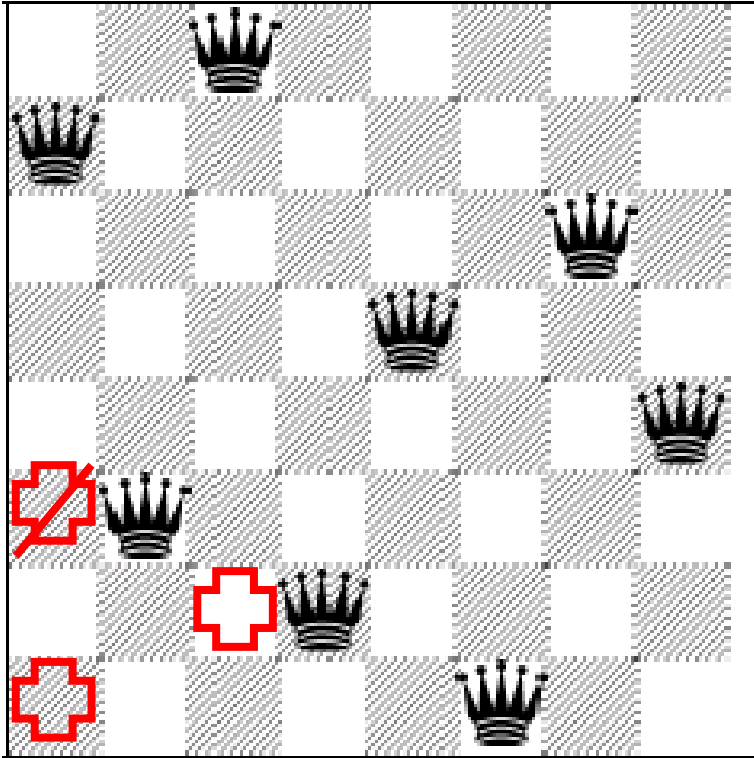
Πλήρης απαρίθμηση:  
 **$n$  Queens**

$n$  βασίλισσες σε  $n \times n$  σκακιέρα



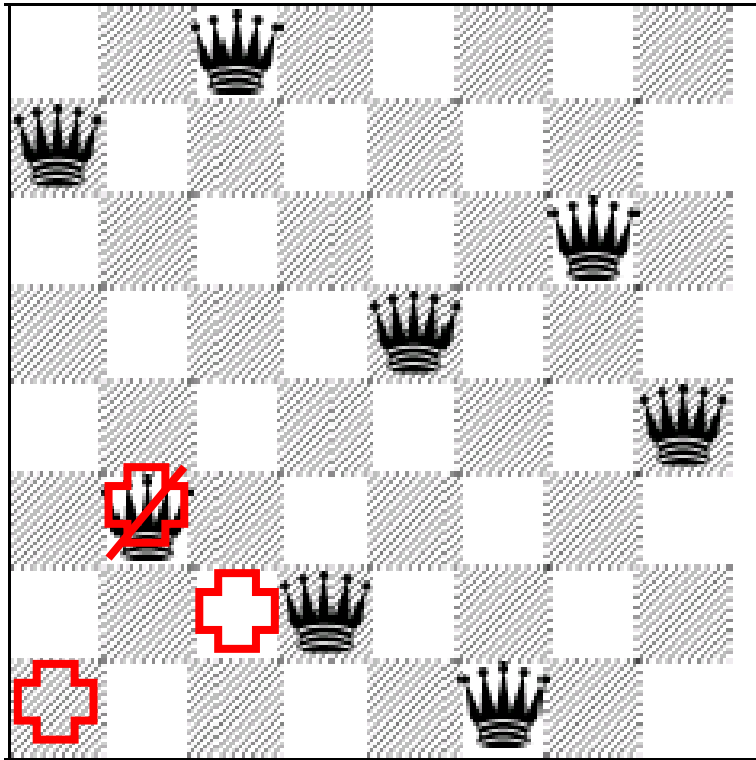
Πλήρης απαρίθμηση:  
 **$n$  Queens**

$n$  βασίλισσες σε  $n \times n$  σκακιέρα



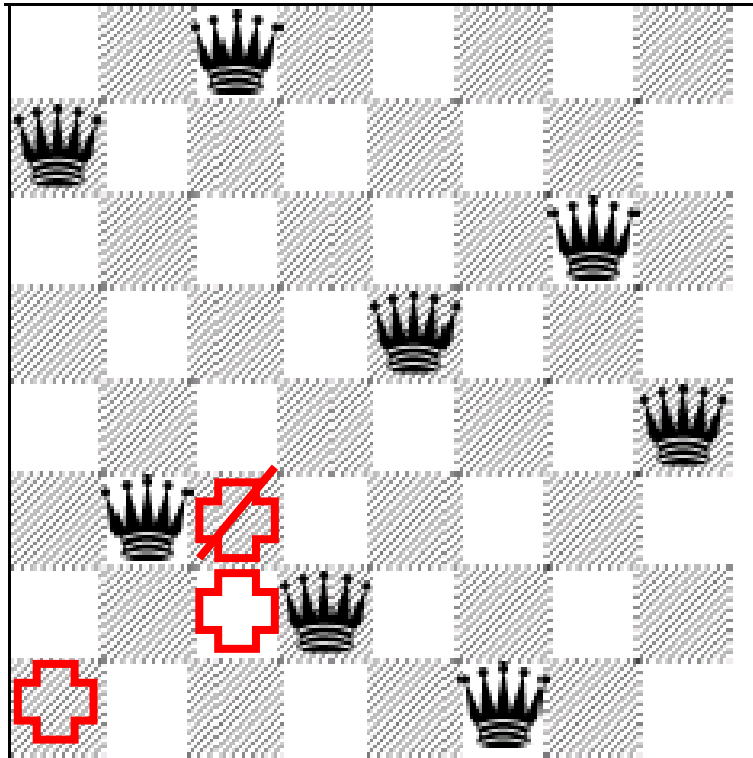
Πλήρης απαρίθμηση:  
 **$n$  Queens**

$n$  βασίλισσες σε  $n \times n$  σκακιέρα



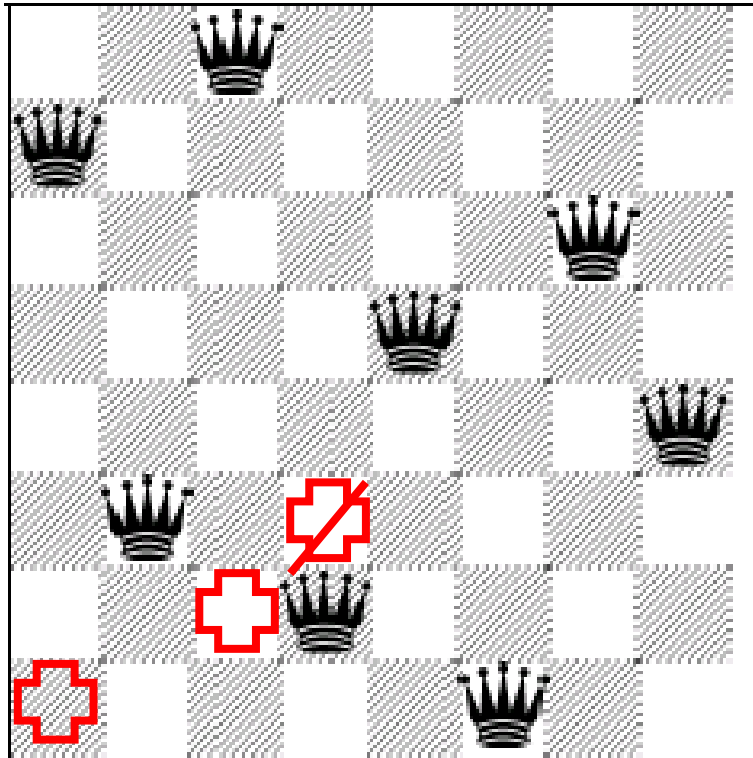
Πλήρης απαρίθμηση:  
 **$n$  Queens**

$n$  βασίλισσες σε  $n \times n$  σκακιέρα



Πλήρης απαρίθμηση:  
 **$n$  Queens**

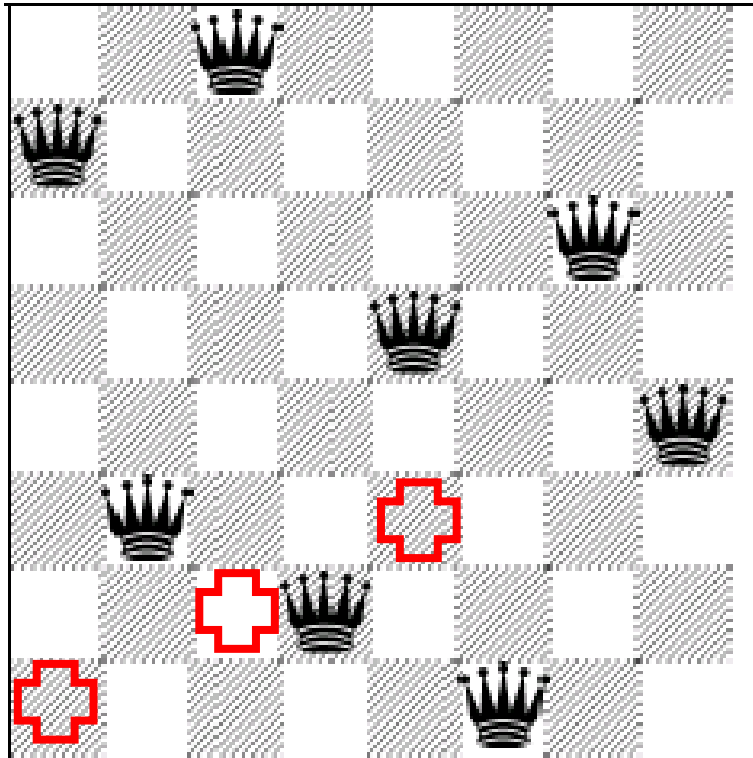
$n$  βασίλισσες σε  $n \times n$  σκακιέρα





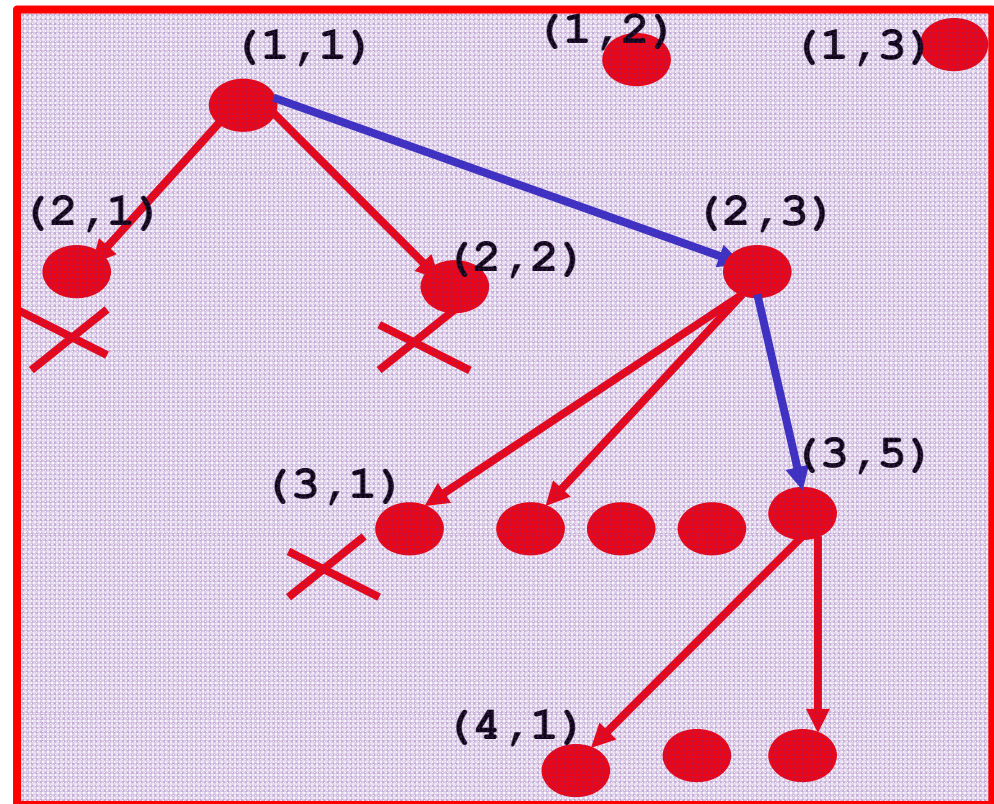
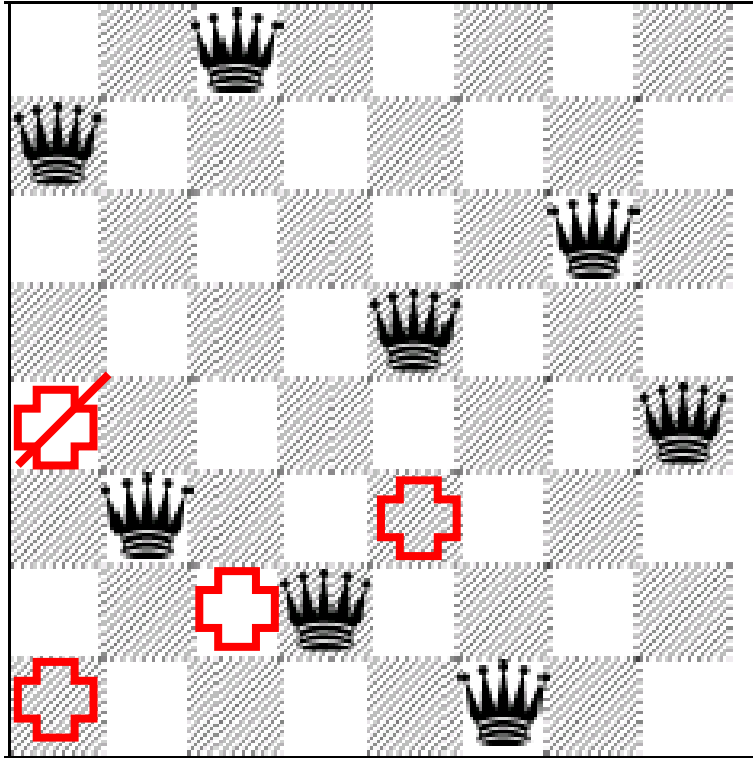
Πλήρης απαρίθμηση:  
 **$n$  Queens**

$n$  βασίλισσες σε  $n \times n$  σκακιέρα



Πλήρης απαρίθμηση:  
 **$n$  Queens**

$n$  βασίλισσες σε  $n \times n$  σκακιέρα



Πλήρης απαρίθμηση/  
n Queens / αλγόριθμος

```
function conflict(i1, j1, i2, j2): boolean
```

```
    conflict := (i1 = i2) OR  
                (j1 = j2) OR  
                (abs(i1 - i2) = abs(j1 - j2))
```

Πλήρης απαρίθμηση/  
n Queens / αλγόριθμος

Βασίλισσα  $i$  στη θέση  $(i, j)$   
Βασίλισσα  $k$  στη θέση  $(k, Pos[k])$

```
function compatible(i, j) : boolean
```

```
  c := TRUE; k := 1;
```

```
  while c and (k < i) do
```

```
    c := not conflict(i, j, k, Pos[k])
```

```
    k := k+1
```

```
  endwhile
```

```
  compatible := c
```

Πλήρης απαρίθμηση/  
n Queens / αλγόριθμος

Queens(i)

```
if i > NQueens then  
    print solution
```

```
else
```

```
    for j := 1 to NQueens do  
        if compatible(i,j) then  
            Pos[i] := j;  
            Queens(i+1);  
        endif  
    endfor
```

```
endif
```

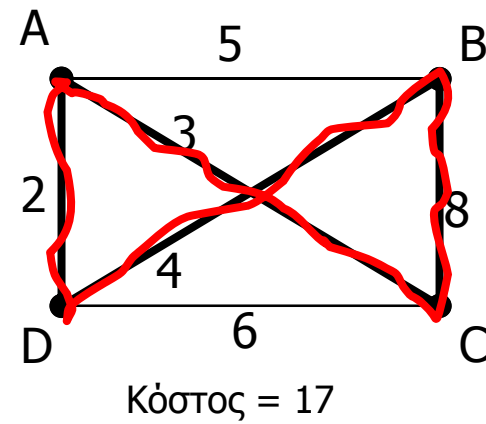
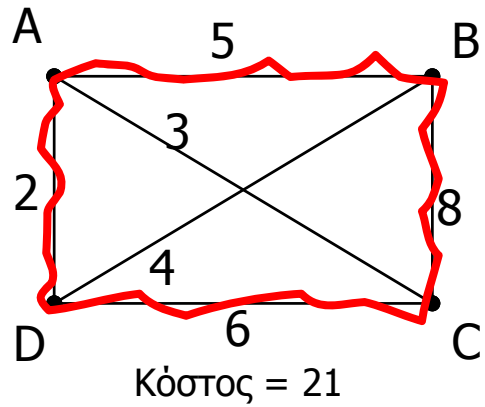
Πλήρης απαρίθμηση:  
**n Queens**

- Πολυπλοκότητα (άσκηση)
- Μοντελοποίηση με γράφο  $G=(V,E)$ 
  - $V=?$ ,  $E=?$
- Μέγιστο Ανεξάρτητο σύνολο (maximum independent set problem)
- Branch and Bound (Heuristics)

## TSP/ Πρόβλημα πλανόδιου πωλητή

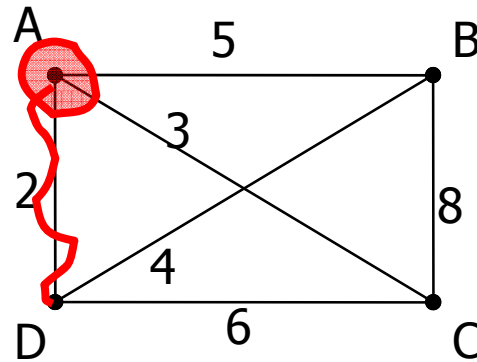
Δίνεται ένας πλήρης γράφος  $G=(V, E, W)$  με βάρη, τάξης  $n$ . Να βρεθεί ένας κύκλος ο οποίος να περνά από όλους τους κόμβους μία και μόνο μία φορά (Hamilton cycle) και ο οποίος να έχει ελάχιστο κόστος.

# Πρόβλημα πλανόδιου πωλητή (TSP)



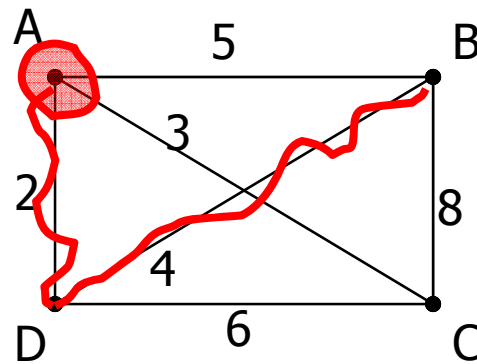


Greedy πλησιέστερου  
γείτονα (nearest neighbour)



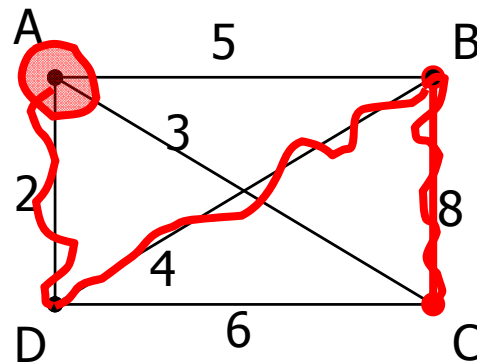
Κόστος = 2

Greedy πλησιέστερου  
γείτονα (nearest neighbour)



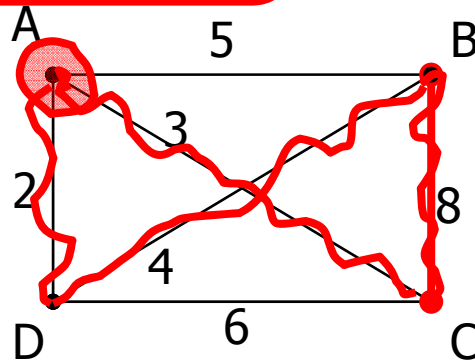
Κόστος = 6

Greedy πλησιέστερου  
γείτονα (nearest neighbour)



Κόστος = 14

Greedy πλησιέστερου  
γείτονα (nearest neighbour)



Κόστος = 17

## TSP/ Πρόβλημα πλανόδιου πωλητή

Greedy: μη βέλτιστη λύση σε πολυωνυμικό χρόνο

Πλήρης απαρίθμηση: βέλτιστη λύση αλλά  
πολυπλοκότητα **εκθετική**

Δυναμικός προγραμματισμός: βέλτιστη λύση αλλά  
πολυπλοκότητα **εκθετική**

# 0-1 Knapsack

$$x_i \in \{1, 0\}$$

$$\text{Max} \sum_{i=1}^n c_i x_i$$

$$\sum_{i=1}^n a_i x_i \leq b$$

**Instance:**

$$\begin{aligned} \max & 10 x_1 + 5 x_2 + 8 x_3 \\ & 3 x_1 + 2 x_2 + 2 x_3 \leq 4 \end{aligned}$$

# 0-1 Knapsack

Greedy: μη βέλτιστη λύση σε **πολυωνυμικό χρόνο**

Πλήρης απαρίθμηση: βέλτιστη λύση αλλά **πολυπλοκότητα εκθετική**

Δυναμικός προγραμματισμός: βέλτιστη λύση αλλά **πολυπλοκότητα ψευδοπολυωνυμική**