ΚΣΕ

ΑΛΓΟΡΙΘΜΟΙ – ΑΝΑΔΡΟΜΙΚΕΣ ΣΥΝΑΡΤΗΣΕΙΣ

Ι.Κοτίνη

ΣΤΟΧΟΙ

- Αποτίμηση της επίδοσης ενός Αλγόριθμου
- Πρόβλημα:
  - Σύγκριση δύο αλγόριθμων
  - Ανάπτυξη 'καλού' αλγόριθμου

ΑΛΓΌΡΙΘΜΟΣ

- Η εκτέλεση ενός αλγόριθμου πρέπει να είναι τυφλή εφαρμογή κανόνων
- Η επεξεργαζόμενη πληροφορία πρέπει να είναι πεπερασμένη
- Κωδικοποίηση της πληροφορίας από πεπερασμένο αριθμό συμβόλων
- Πεπερασμένη ακολουθία πράξεων σε πεπερασμένες ακολουθίες συμβόλων

Κάθε αλγόριθμος μπορεί να προγραμματιστεί;

- πρόγραμμα $\rightarrow$ αλγόριθμος
- αλγόριθμος$\rightarrow$πρόγραμμα

- Για κάθε αλγόριθμο πρέπει:

  πρόγραμμα $\Leftrightarrow$αλγόριθμος

Αναδρομικοί Αλγόριθμοι

- Αναδρομικός αλγόριθμος (recursive algorithm)
  - Επιλύει ένα πρόβλημα λύνοντας ένα ή περισσότερα στιγμιότυπα του ίδιου προβλήματος.

Παράδειγμα: Υπολογισμός παραγοντικού

```
int factorial (int N)
{
    if (N==0) return 1;
    return N*factorial(N-1);
}
```

$$N! = 1 \cdot 2 \cdot 3 \ldots (N-1) \cdot N = N \cdot (N-1)!$$

Αναδρομικοί Αλγόριθμοι

Παράδειγμα: Υπολογισμός παραγοντικού

$$N! = 1 \cdot 2 \cdot 3 \ldots (N-1) \cdot N = N \cdot (N-1)!$$

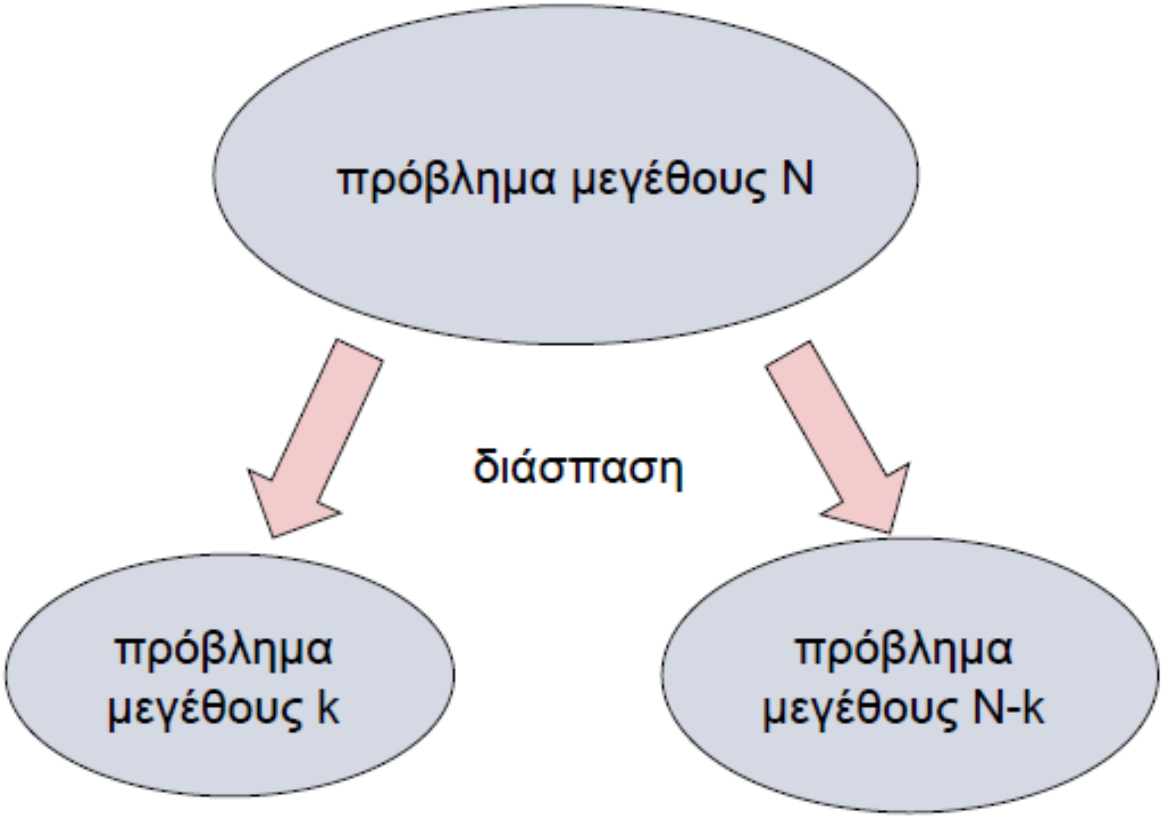
αναδρομικό πρόγραμμα

```
int factorial (int N)
{
    if (N==0) return 1;
    return N*factorial(N-1);
}
```

μη αναδρομικό πρόγραμμα

```
for (t=1, i=1; i<=N; i++){
        t *= i;
}
```

Αναδρομικοί Αλγόριθμοι: Διαίρει και βασίλευε



Διαίρει και Βασίλευε

- Η μέθοδος αυτή:
  - διαιρεί το πρόβλημα σε ανεξάρτητα επιμέρους προβλήματα
  - τα επιλύει
  - συνδυάζει τις λύσεις
  - υπολογίζει μια λύση για το αρχικό πρόβλημα

Πλεονεκτήματα

- οδηγεί σε σε αλγόριθμους που είναι εύκολο να αναλυθούν
  - διατύπωση και επίλυση της αναδρομικής εξίσωσης
  - η αναδρομική εξίσωση προκύπτει απ'ευθείας από την ιδέα στην οποία βασίζεται ο αλγόριθμος
  - υπάρχουν μαθηματικά εργαλεία για την επίλυση της αναδρομικής εξίσωσης
    - θεώρημα Κυριαρχίας
    - Μέθοδος Επανάληψης (Δένδρο αναδρομής)
    - Μέθοδος Αντικατάστασης

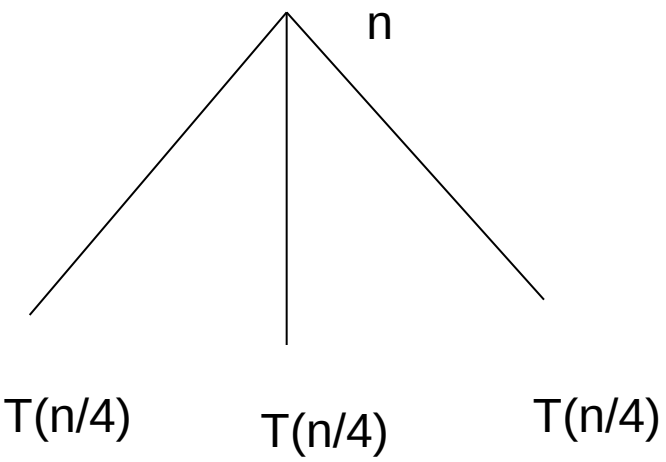
Διαίρει και Βασίλευε

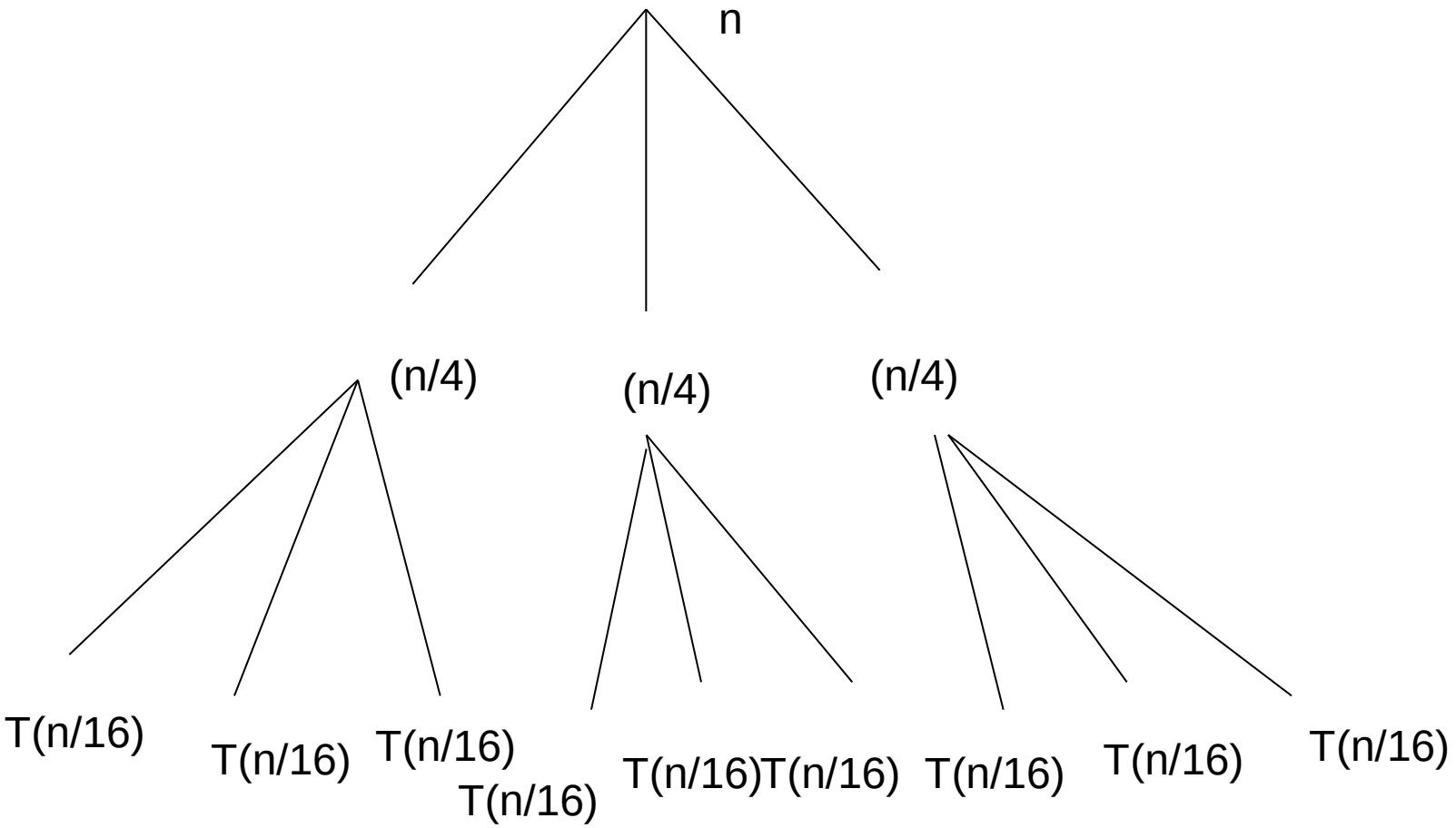
- Μειονεκτήματα
- Στην περίπτωση όπου τα επιμέρους προβλήματα περιέχουν κοινά επιμέρους υποπροβλήματα:
  - επιλύει πολλές φορές το ίδιο επιμέρους πρόβλημα

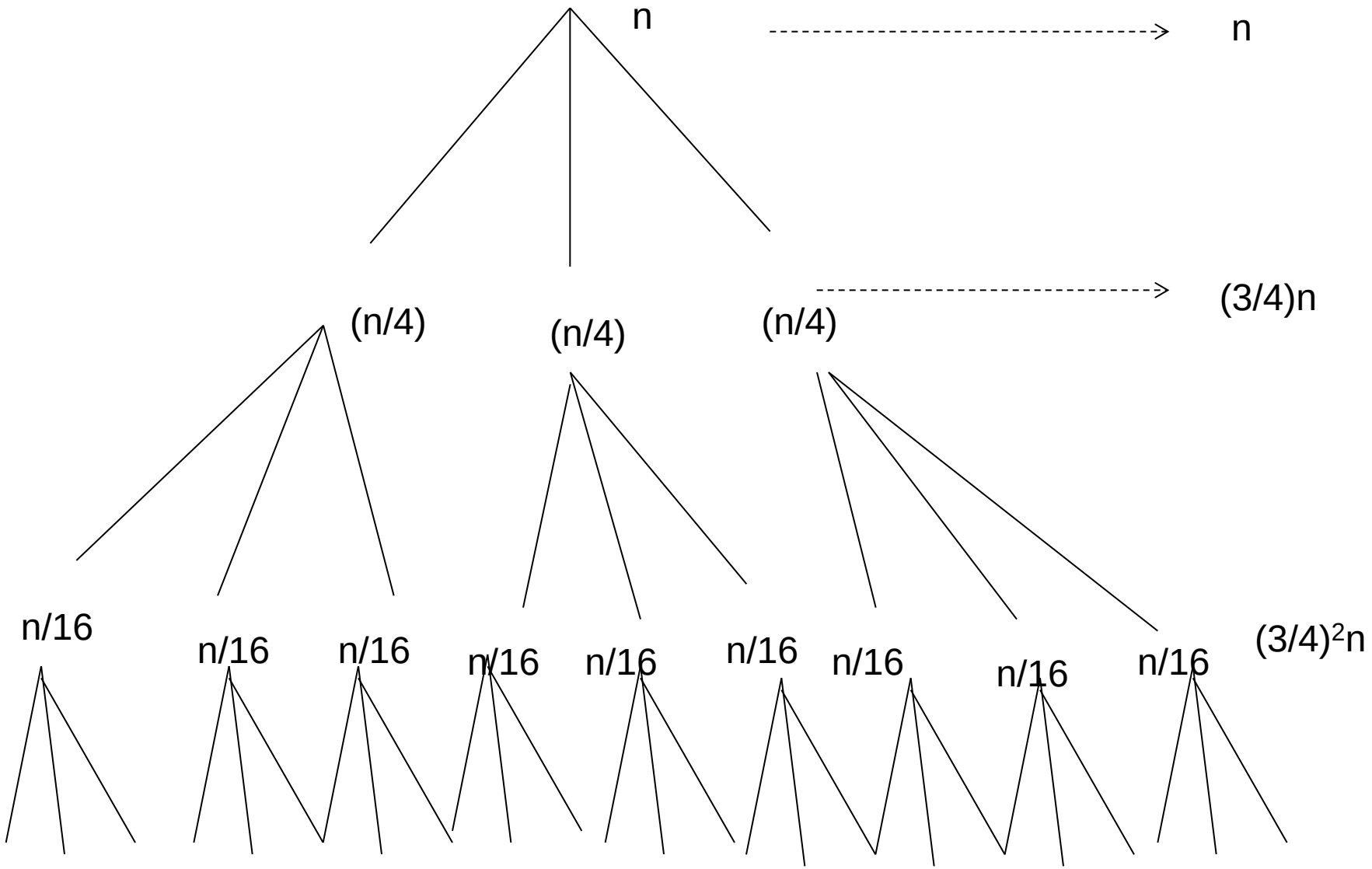
Δένδρα Αναδρομής

- Έστω ότι θέλουμε να επιλύσουμε την αναδρομή

$$T(n) = 3T(n/4) + n$$




n
(n/4)
(n/4)
(n/4)
T(n/16)
T(n/16)
T(n/16)
T(n/16)
T(n/16)
T(n/16)
T(n/16)
T(n/16)
T(n/16)


n
n
(n/4)
(n/4)
(n/4)
(3/4)n
n/16
n/16
n/16
n/16
n/16
n/16
n/16
n/16
n/16
(3/4)²n

Αναδρομή;

- Η αναδρομή χρησιμοποιείται σε αλγορίθμους διαίρει και βασίλευε γιατί:
  - Ένα πρόβλημα εκφράζεται σε υποπροβλήματα που είναι σημαντικά μικρότερα του αρχικού προβλήματος.

Παραδείγματα Αναδρομικών συναρτήσεων με αλγόριθμους Διαίρει και Βασίλευε

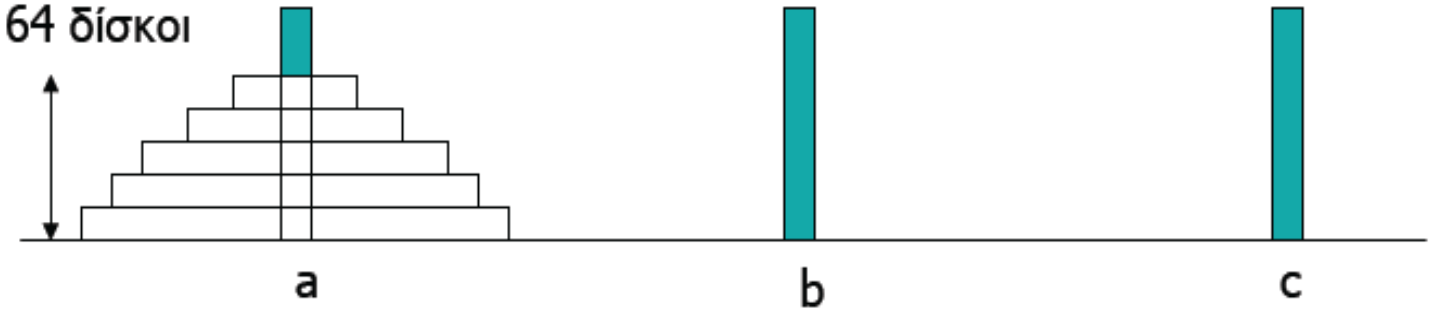
- Αλγόριθμος εύρεσης του μέγιστου και του ελάχιστου στοιχείου ενός πίνακα
- Πύργοι του Hanoi

Αλγόριθμος εύρεσης του μέγιστου και του ελάχιστου στοιχείου ενός πίνακα

```
MinMax (A, i, j, min, max)
1.  if i ≥ j − 1 then
2.          if ai  < aj  then
3.                  max = aj
4.                  min = ai
5.          else
6.                  max = ai
7.                  min = aj
8.          end if
9.  else
10.         m = ⌊(i + j)/2⌋
11.         MinMax(A, i, m, min1, max1 )
12.         MinMax(A, m, j, min2, max2 )
13.         min = f min(min1, min2)
14.         max = f max(max1 , max2)
15.  end if
```

Παράδειγμα: οι πύργοι του Hanoi

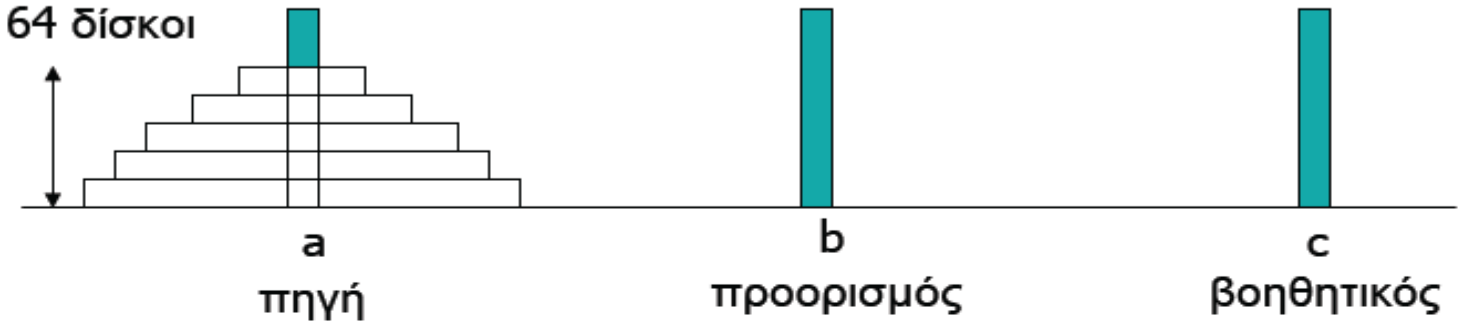
Πρόβλημα: να μεταφέρουμε όλους τους δίσκους σε έναν άλλο πάσσαλο κουνώντας ένα δίσκο τη φορά, χωρίς ποτέ ένας δίσκος να τοποθετηθεί πάνω από μικρότερους δίσκους.



Ψάχνουμε για αναδρομική σχέση:

Πρόβλημα Α: μετάφερε 64 δίσκους από το a στο b (χωρίς να παραβείς τους περιορισμούς)

Παράδειγμα: οι πύργοι του Hanoi (2)



μεταφορά_πύργου (*N, πηγή, προορισμός, βοηθητικός*)

module *μεταφορά_πύργου* (*N, a, b, c*)
if N=1
  then *μετακίνησε 1 δίσκο από a σε b*
  else { *μεταφορά_πύργου* (*N-1, a, c, b*);
    μετακίνησε 1 δίσκο από a σε b;
    μεταφορά_πύργου (*N-1, c, b, a*) }

Παράδειγμα: οι πύργοι του Hanoi (3)

```
module μεταφορά_πύργου (N, a, b, c)
if N=1
   then μετακίνησε τον δίσκο από πηγή σε προορισμό
   else { μεταφορά_πύργου (N-1, a, c, b);
          μετακίνησε 1 δίσκο από πηγή σε προορισμό;
          μεταφορά_πύργου (N-1, c, b, a) }
```