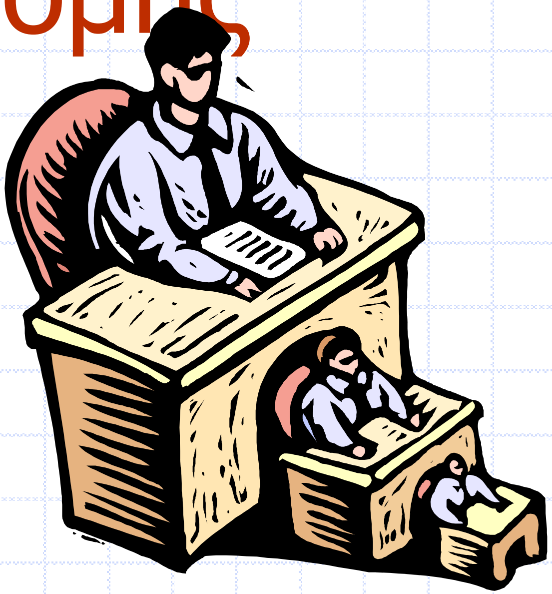
Χρήση της Αναδρομής



Το πρότυπο της αναδρομής

- **Αναδρομή**: όταν μια μέθοδος καλεί τον εαυτό της
- Κλασσικό παράδειγμα –η συνάρτηση παραγοντικό:
  - n! = n · (n-1) · … · 2· 1
- Αναδρομικός ορισμός:

$$f(n)=\begin{cases} 1 & \text{if } n=0 \\ n\cdot f(n-1) & else \end{cases}$$

- Σαν μέθοδος Java :

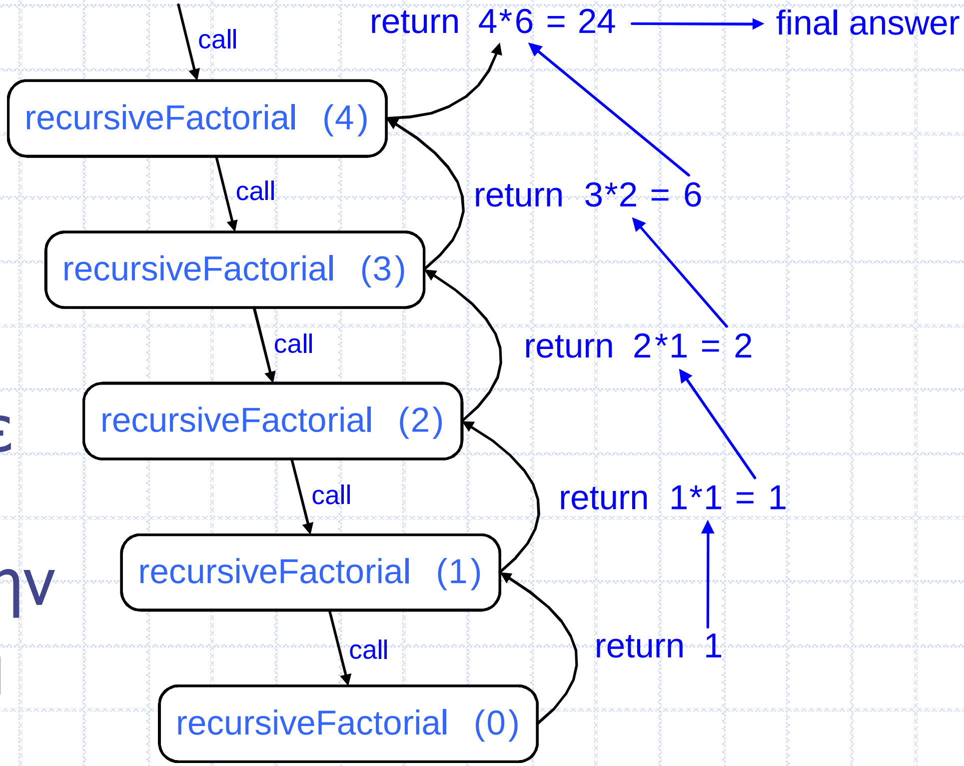
```
// αναδρομική συνάρτηση παραγοντικό
public static int  recursiveFactorial(int n) {
   if  (n  ==  0)  return  1;      // περίπτωση βάσης
   else return  n  *  recursiveFactorial(n- 1);  // αναδρομή
}
```

Περιεχόμενο Αναδρομικής Μεθόδου

- Περίπτωση(εις) Βάσης
  - Οι τιμές των μεταβλητών εισόδου για τις οποίες δεν εκτελούμε αναδρομικές κλήσεις ονομάζονται περιπτώσεις βάσης (πρέπει να υπάρχει τουλάχιστον μια περίπτωση βάσης).
  - Κάθε πιθανή ακολουθία αναδρομικών κλήσεων πρέπει τελικά να φτάνει μια περίπτωση βάσης.
- Αναδρομικές κλήσεις
  - Κλήσεις στην τρέχουσα μέθοδο.
  - Κάθε αναδρομική κλήση πρέπει να ορίζεται έτσι που προοδευτικά να οδηγεί σε περίπτωση βάσης.

Οπτικοποίηση της Αναδρομής

- Ίχνη Αναδρομής
  - Ένα κουτί για κάθε αναδρομική κλήση
  - Ένα βέλος από την κλήση προς το καλούμενο
  - Ένα βέλος από κάθε καλούμενο στην κλήση δείχνοντας την επιστρεφόμενη τιμή
- Παράδειγμα



Παράδειγμα: Αγγλικός Χάρακας

- Εκτύπωση των γραμμών και αριθμών όπως στον Αγγλικο χάρακα :

```
---- 0          ----- 0         --- 0
-               -               -
--              --              --
-               -               -
---             ---             --- 1
-               -               -
--              --              --
-               -               -
---- 1          ----            --- 2
-               -               -
--              --              --
-               -               -
---             ---             --- 3
-               -
--              --
-               -
---- 2          ----- 1
```

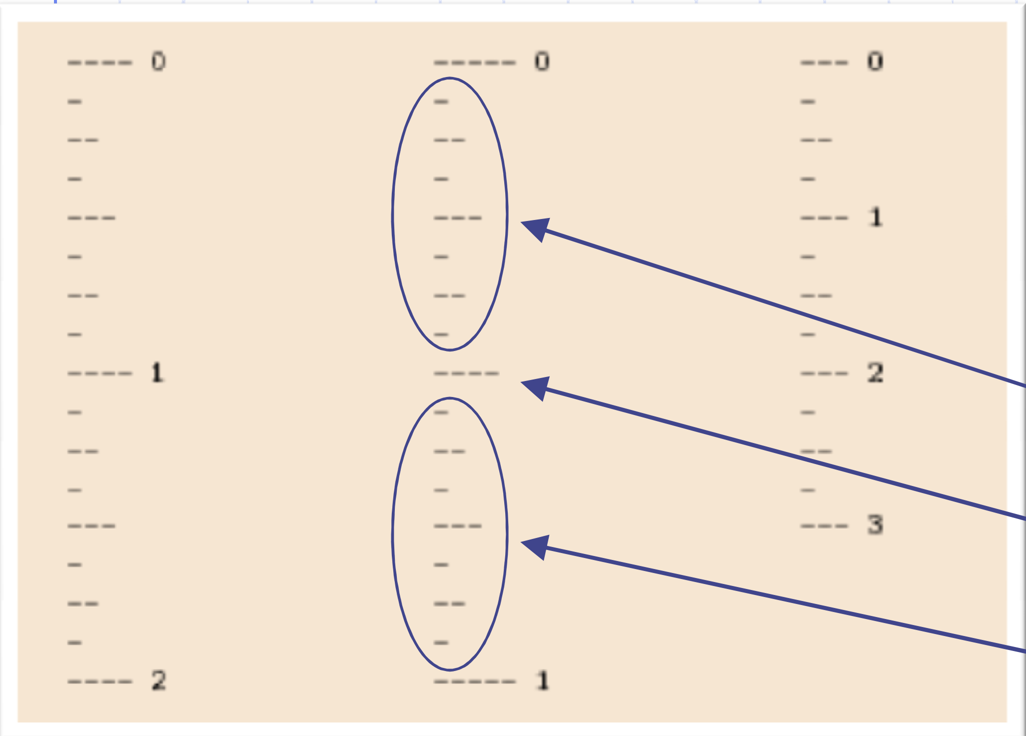
Από τον Matt Stallmann.

Χρήση της Αναδρομής

drawTicks(length)

Είσοδος: μήκος μιας γραμμής ‘tick’

Έξοδος: χάρακας με γραμμή δοθέντος μήκους στο μέσο και μικρότερους χάρακες σε κάθε μεριά



drawTicks(length)

if(length > 0) then

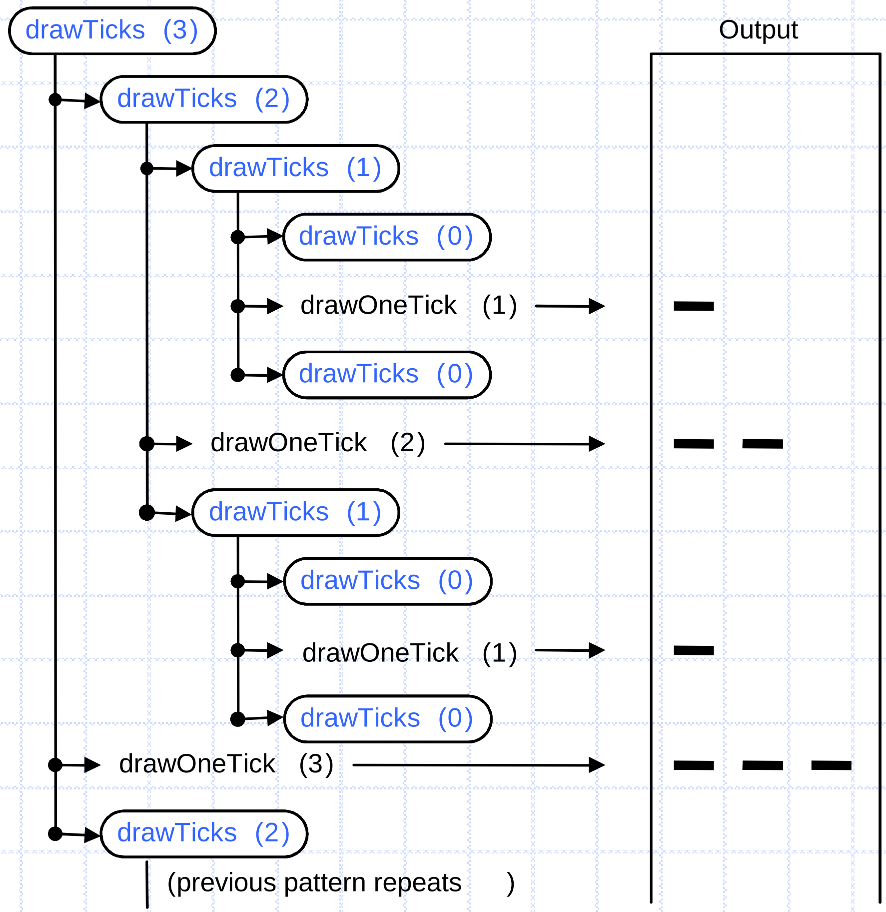
drawTicks(length – 1)

draw tick of the given length

drawTicks(length – 1)

Αναδρομική Μέθοδος Σχεδιασμού

- Η μέθοδος σχεδιασμού βασίζεται στον παρακάτω αναδρομικό ορισμό
- Ένα διάστημα με μια κεντρική γραμμή μήκους L ≥1 αποτελείται από:
  - Ένα διάστημα με κεντρική γραμμή μήκους L–1
  - Μια γραμμή μήκους L
  - Ένα διάστημα με κεντρική γραμμή μήκους L–1



Γραμμική Αναδρομή

- Έλεγχος για περιπτώσεις βάσης
  - Έναρξη με έλεγχο ενός συνόλου περιπτώσεων βάσης (πρέπει να υπάρχει τουλάχιστον μια).
  - Κάθε πιθανή ακολουθία αναδρομικών κλήσεων πρέπει τελικά να φτάνει μια περίπτωση βάσης, και ο χειρισμός κάθε περίπτωσης βάσης δεν πρέπει να είναι αναδρομικός.
- Αναδρομή μια φορά
  - Εκτέλεση μιας αναδρομικής κλήσης
  - Το βήμα αυτό μπορεί να περιλαμβάνει έναν έλεγχο που αποφασιζει ποιά από πιθανές αναδρομικές κλήσεις να εκτελέσει, αλλά τελικά πρέπει να εκτελέσει μια μόνο από αυτές
  - Ορίστε κάθε πιθανή αναδρομική κλήση έτσι που προοδευτικά να οδηγεί σε περίπτωση βάσης.

Παράδειγμα Γραμμικής Αναδρομής

Algorithm LinearSum(*A*, *n*):

Είσοδος:

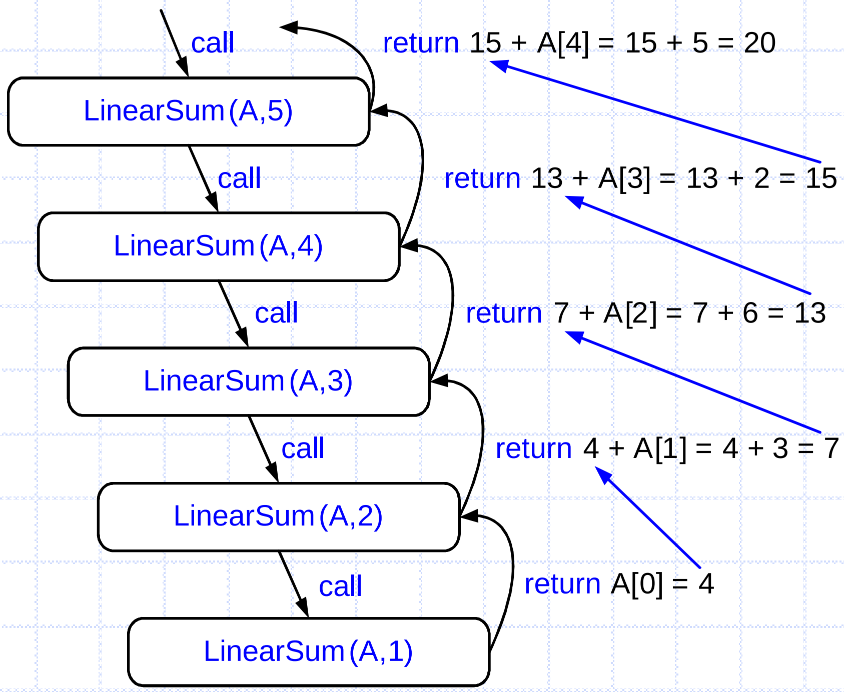
Ένας ακέραιος πίνακας *A και ένας ακέραιος n* >= 1, έτσι που ο *A* έχει τουλάχιστον *n* στοιχεία

Έξοδος:

Το άθροισμα των πρώτων ακεραίων του *A*

if *n* = 1 **then**
  return *A*[0]
else
  return LinearSum(*A*, *n* - 1) + *A*[*n* - 1]

Παράδειγμα ίχνη της αναδρομής:



A={4,3,2,5} και n=5

Αντιστροφή ενός Πίνακα

Algorithm ReverseArray(*A*, *i*, *j*):

Είσοδος: ένας πίνακας *A* και οι μη αρνητικοί δείκτες *i* και *j*

Έξοδος: η αντιστροφή των στοιχείων στον *A* αρχίζοντας από τη θέση *i* και τερματίζοντας στην *j*

```
if i < j then
    Swap A[i] and A[j]
    ReverseArray(A, i + 1, j - 1)
return
```

Ορισμός Παραμέτρων για Αναδρομή

- Για την δημιουργία αναδρομικών μεθόδων, είναι σημαντικό να ορισθούν οι μέθοδοι κατά τρόπους που διευκολύνουν την αναδρομή.
- Αυτό μερικές φορές απαιτεί τον ορισμό επιπλέον παραμέτρων που παιρνούν στη μέθοδο.
- Για παράδειγμα, ορίσαμε την μέθοδο αντιστροφής των στοιχείων του πίνακα σαν ReverseArray(*A*, *i*, *j*), και όχι σαν ReverseArray(*A*).

Υπολογισμός Δυνάμεων

- Η συνάρτηση ύψωση σε δύναμη, $p(x,n)=x^n$, μπορεί να ορισθεί αναδρομικά:

$$p(x,n)=\begin{cases} 1 & \text{if } n=0 \\ x\cdot p(x,n-1) & \text{else} \end{cases}$$

- Αυτό οδηγεί σε μια συνάρτηση δύναμη που τρέχει σε O(n) χρόνο (εκτελούμε n αναδρομικές κλήσεις).
- Ωστόσο μπορούμε να το βελτιώσουμε.

Αναδρομική ύψωση στο τετράγωνο

- Μπορούμε να έχουμε ένα πιο αποδοτικό γραμμικό αλγόριθμο χρησιμοποιώντας επαναλαμβανόμενη ύψωση στο τετράγωνο:

$$p(x,n) = \begin{cases} 1 & \text{if } n = 0 \\ x \cdot p(x,(n-1)/2)^2 & \text{if } n > 0 \text{ is odd} \\ p(x,n/2)^2 & \text{if } n > 0 \text{ is even} \end{cases}$$

- Για παράδειγμα,

$2^4 = 2^{(4/2)2} = (2^{4/2})^2 = (2^2)^2 = 4^2 = 16$

$2^5 = 2^{1+(4/2)2} = 2(2^{4/2})^2 = 2(2^2)^2 = 2(4^2) = 32$

$2^6 = 2^{(6/2)2} = (2^{6/2})^2 = (2^3)^2 = 8^2 = 64$

$2^7 = 2^{1+(6/2)2} = 2(2^{6/2})^2 = 2(2^3)^2 = 2(8^2) = 128.$

Μέθοδος Αναδρομική ύψωση σε δύναμη

```
Algorithm Power(x, n):
    Είσοδος: ένας αριθμός x και ένας ακέραιος n ≥ 0
    Output: η τιμή του x^n
    if n = 0   then
        return 1
    if n is odd then
        y  = Power(x, (n - 1)/ 2)
        return x · y · y
    else
        y = Power(x, n/ 2)
        return y · y
```

Ανάλυση

Algorithm Power(x, n):
    Είσοδος: ένας αριθμός x και ένας ακέραιος $n \geq 0$
    Έξοδος: η τιμή x^n
    if $n = 0$ **then**
        return 1
    if n is odd **then**
        y = Power(x, $(n - 1)/2$)
        return $x \cdot y \cdot y$
    else
        y = Power(x, $n/2$)
        return $y \cdot y$

Κάθε αναδρομική κλήση κόβει στο μισό την τιμή του n· επομένως, εκτελούνται n αναδρομικές κλήσεις. Δηλαδή, αυτή η μέθοδος τρέχει σε O(log n) χρόνο.

είναι σημαντικό να χρησιμοποιήσουμε εδώ μια μεταβλητή δύο φορές αντι για να καλέσουμε τη μέθοδο δύο φορές.

Αναδρομή Ουράς

- Η αναδρομή ουράς συμβαίνει όταν μια γραμμική αναδρομική μέθοδος εκτελεί την αναδρομική κλήση σαν τελευταίο βήμα.
- Ένα παράδειγμα είναι η μέθοδος αντιστροφής ενός πίνακα.
- Τέτοιες μέθοδοι μπορούν εύκολα να μετατραπούν σε μη αναδρομικές (που εξοικονομεί πόρους).
- Παράδειγμα:

Algorithm IterativeReverseArray(*A*, *i*, *j*):
    Είσοδος: Ένας πίνακας *A* και μη αρνητικοί δείκτες *i* και *j*
    Έξοδος: η αντιστροφή των στοιχείων στον *A* αρχίζοντας από τη θέση *i* και τερματίζοντας στην *j*
  while *i* < *j* **do**
      Swap *A*[*i*] and *A*[*j*]
      i = *i* + 1
      j = *j* - 1
  return

Δυαδική Αναδρομή

- Η δυαδική αναδρομή συμβαίνει όταν υπάρχουν **δύο** αναδρομικές κλήσεις για κάθε μη βασική περίπτωση.
- Παράδειγμα: Εκτύπωση των γραμμών και αριθμών όπως στον Αγγλικο χάρακα.

```
---- 0        ----- 0       --- 0
-             -             -
--            --            --
-             -             -
---           ---           --- 1
-             -             -
--            --            --
-             -             -
---- 1        ----          --- 2
-             -             -
--            --            --
-             -             -
---           ---           --- 3
-             -
--            --
-             -
---- 2        ----- 1
```

A Binary Recursive Method for Drawing Ticks

```
    // draw a tick with no label
public static void drawOneTick(int tickLength) { drawOneTick(tickLength, - 1); }
    // draw one tick
public static void drawOneTick(int tickLength, int tickLabel) {
  for (int i = 0; i < tickLength; i++)
     System.out.print("-");
  if (tickLabel >= 0) System.out.print(" " + tickLabel);
  System.out.print("\n");
}
public static void drawTicks(int tickLength) { // draw ticks of given length
  if (tickLength > 0) {                       // stop when length drops to 0
     drawTicks(tickLength- 1);      // recursively draw left ticks
     drawOneTick(tickLength);       // draw center tick
     drawTicks(tickLength- 1);      // recursively draw right ticks
  }
}
public static void drawRuler(int nInches, int majorLength) { // draw ruler
  drawOneTick(majorLength, 0);  // draw tick 0 and its label
  for (int i = 1; i <= nInches; i++){
     drawTicks(majorLength- 1);  // draw ticks for this inch
     drawOneTick(majorLength, i);          // draw tick i and its label
  }
}
```

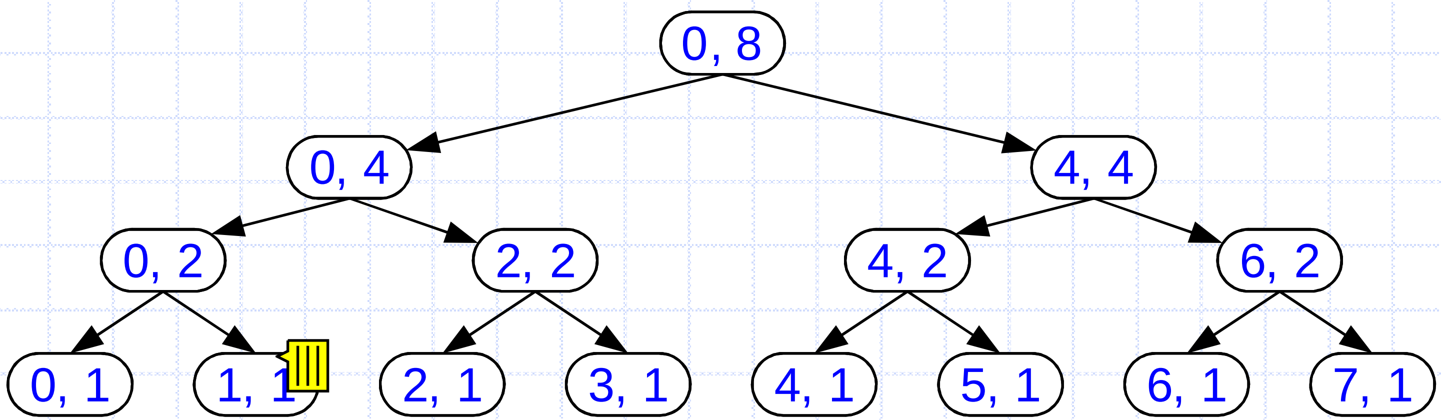
Note the two recursive calls

Μια άλλη δυαδική αναδρομική μέθοδος

- Πρόβλημα: άθροισμα όλων των αριθμών σε ένα ακέραιο πίνακα A:

Algorithm BinarySum(*A, i, n*):
Είσοδος: ένας πίνακας *A* και οι ακέραιοι *i* και *n*
Έξοδος: Tτο άθροισμα των *n* ακεραίων στο *A* αρχίζοντας από το index *i*
if *n* = 1 **then**
return *A*[*i*]
return BinarySum(*A, i, n/* 2) + BinarySum(*A, i* + *n/* 2, *n/* 2)

- **Παράδειγμα:**



Υπολογισμός των Αριθμών Fibonacci

- Οι αριθμοί Fibonacci ορίζονται αναδρομικά:

$$F_0 = 0$$
$$F_1 = 1$$
$$F_i = F_{i-1} + F_{i-2} \quad \text{for } i > 1.$$

- Αναδρομικός αλγόριθμος (πρώτη προσέγγιση):

Algorithm BinaryFib(k):
  Input: Nonnegative integer k
  Output: The kth Fibonacci number F_k
  if $k \leq 1$ **then**
   return k
  else
   return BinaryFib(k - 1) + BinaryFib(k - 2)

Ανάλυση

- Έστω n_k το πλήθος των αναδρομικών κλήσεων της BinaryFib(k)
  - $n_0 = 1$
  - $n_1 = 1$
  - $n_2 = n_1 + n_0 + 1 = 1 + 1 + 1 = 3$
  - $n_3 = n_2 + n_1 + 1 = 3 + 1 + 1 = 5$
  - $n_4 = n_3 + n_2 + 1 = 5 + 3 + 1 = 9$
  - $n_5 = n_4 + n_3 + 1 = 9 + 5 + 1 = 15$
  - $n_6 = n_5 + n_4 + 1 = 15 + 9 + 1 = 25$
  - $n_7 = n_6 + n_5 + 1 = 25 + 15 + 1 = 41$
  - $n_8 = n_7 + n_6 + 1 = 41 + 25 + 1 = 67.$
- Το n_k τουλάχιστον διπλασιάζεται κάθε δεύτερη φορά
- Δηλαδή, $n_k > 2^{k/2}$. Είναι εκθετική!

Ένας καλύτερος αλγόριθμος Fibonacci

- Χρήση γραμμικής αναδρομής

```
Algorithm LinearFibonacci(k):
 Input: A nonnegative integer k
 Output: Pair of Fibonacci numbers (F_k , F_{k−1})
 if k ≤ 1 then
     return (k, 0)
 else
     (i, j) ← LinearFibonacci(k − 1)
     return (i +j, i)
```

- LinearFibonacci εκτελεί k−1 αναδρομικές κλήσεις

Πολλαπλή Αναδρομή

- Παραδειγμα:
  - σπαζοκεφαλιά αθροίσματος
    - *pot* + *pan* = *bib*
    - *dog* + *cat* = *pig*
    - *boy* + *girl* = *baby*
- Πολλαπλή αναδρομή:
  - τελικά απαιτεί πολλές αναδρομικές κλήσεις
  - όχι μια ή δύο

Αλγόριθμος για πολλαπλή αναδρομή

Algorithm PuzzleSolve(k,S,U):
Είσοδος: Ακέραιος k, ακολουθία S, και το σύνολο U (σύνολο όλων των προς εξέταση στοιχείων)
Έξοδος: Απαρίθμηση όλων επιλογών μήκους k στο S με χρήση στοιχείων από το U χωρίς επαναλήψεις επιλογών

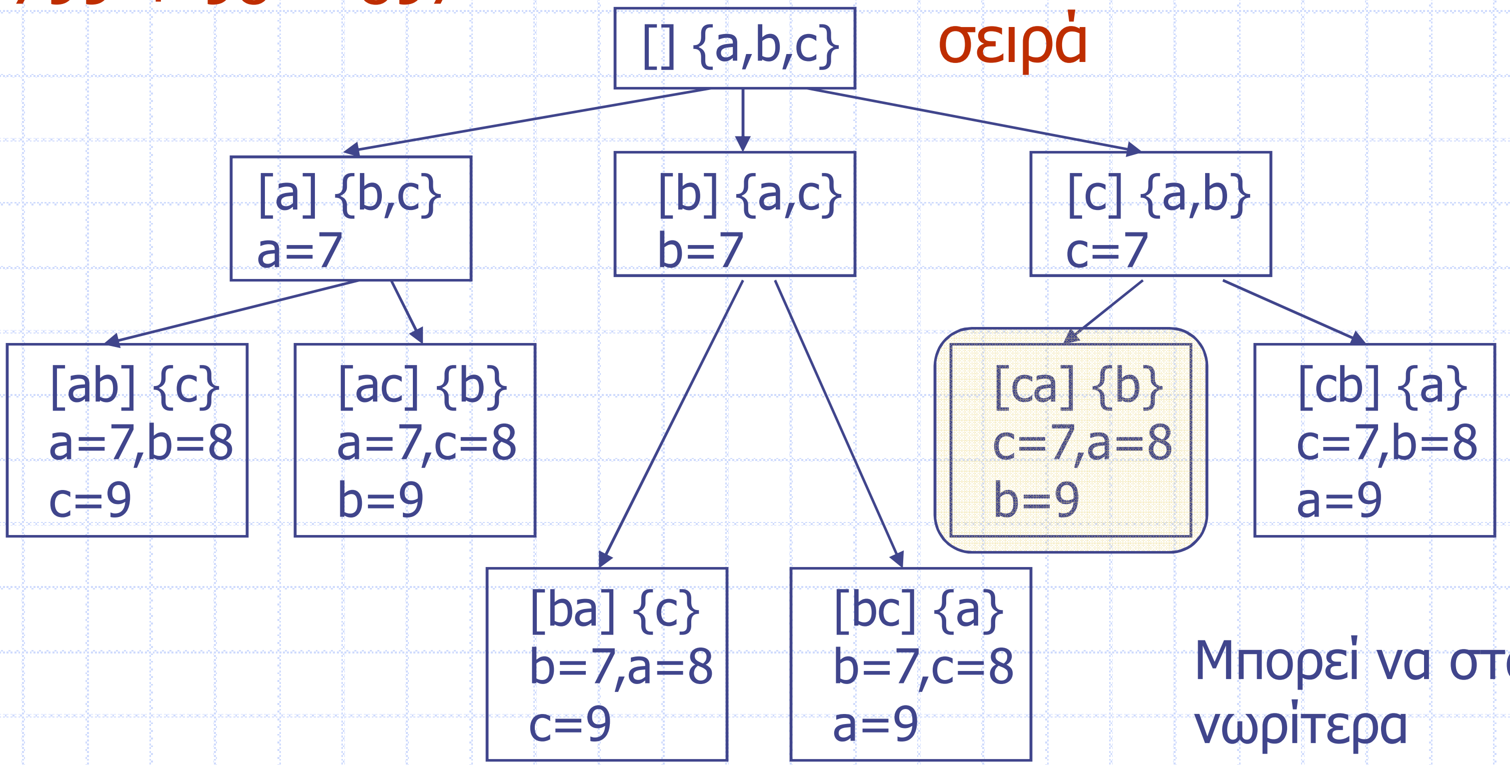
```
for all e in U do
    Remove e from U          {η e χρησιμοποιείται τώρα}
    Add e to the end of S
    if k = 1 then
        Test whether S is a configuration that solves the puzzle
        if S solves the puzzle then
            return "Solution found: " S
    else
        PuzzleSolve(k - 1, S,U)
    Add e back to U          {η e δεν χρησιμοποιείται τώρα}
    Remove e from the end of S
```

Παράδειγμα

cbb + ba = abc

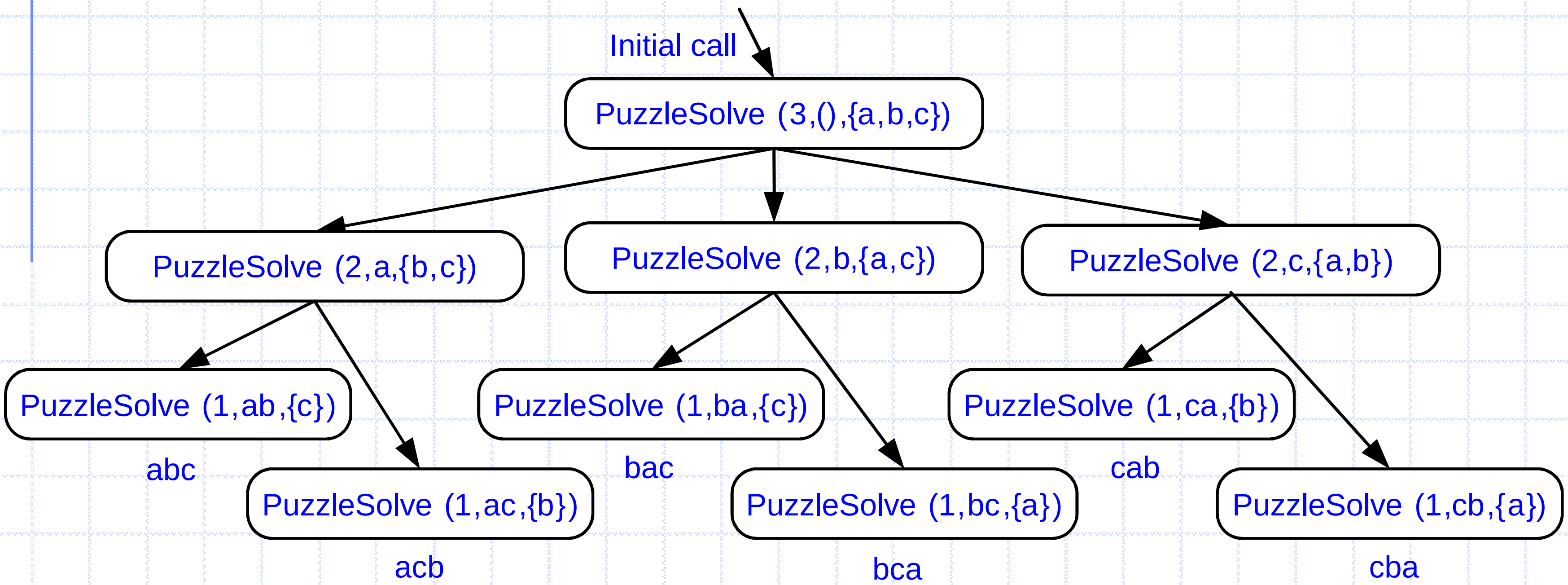
799 + 98 = 897

a,b,c είναι τα 7,8,9; Όχι υποχρεωτικά με αυτή τη σειρά



Μπορεί να σταματάει νωρίτερα

Οπτικοποίηση της Σπαζοκεφαλιάς



Πύργοι του Ανόι

Στο παιγνίδι Πύργοι του Ανόι, μας δίδεται μια πλατφόρμα με τρεις πασσάλους, a, b, και c, που εξέχουν από αυτή. Στον πάσσαλο a υπάρχει μια στοίβα από n δίσκους, που ο καθένας είναι μεγαλύτερος από τον επόμενο, έτσι που ο μικρότερος είναι στην κορυφή και ο μεγαλύτερος στη βάση. Το παιγνίδι έγκειται να μετακινηθούν όλοι οι δίσκοι από τον πάσσαλο a στον πάσσαλο c, μετακινώντας ένα δίσκο κάθε φορά, έτσι που ποτέ δεν τοποθετούμε έναν μεγάλο δίσκο πάνω σε έναν μικρότερο.