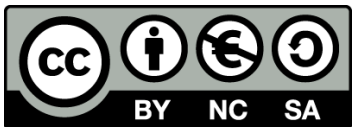
ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΪΚΑ
ΜΑΘΗΜΑΤΑ

ΑΛΓΟΡΙΘΜΟΙ

Ενότητα 2: Ανάλυση Αλγορίθμων

Ιωάννης Μανωλόπουλος, Καθηγητής
Αναστάσιος Γούναρης, Επίκουρος Καθηγητής
Τμήμα Πληροφορικής ΑΠΘ



Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΪΚΑ
ΜΑΘΗΜΑΤΑ

Ανάλυση Αλγορίθμων

Αποδοτικότητα, Συμβολισμοί Ο, Ω, Θ

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

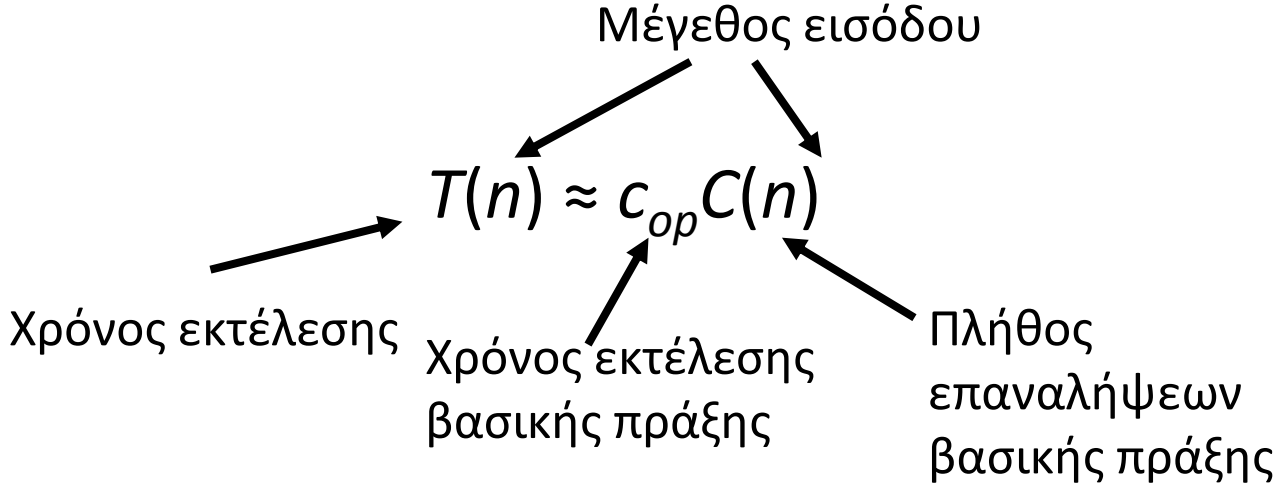
Ανάλυση Αλγορίθμων

- Θέματα:
  - Ορθότητα
  - Χρονική αποδοτικότητα
  - Χωρική αποδοτικότητα
  - Βελτιστότητα
- Προσεγγίσεις:
  - Θεωρητική ανάλυση
  - Εμπειρική ανάλυση

Θεωρητική ανάλυση χρονικής αποδοτικότητας

Η χρονική αποδοτικότητα αναλύεται προσδιορίζοντας τον αριθμό των επαναλήψεων της βασικής πράξης ως συνάρτηση του μεγέθους εισόδου.

Βασική πράξη: η πράξη που συνεισφέρει περισσότερο από τις άλλες στο χρόνο εκτέλεσης του αλγορίθμου.



Μέγεθος εισόδου & βασική πράξη

Πρόβλημα	***Μέγεθος εισόδου***	***Βασική πράξη***
Αναζήτηση κλειδιού σε λίστα με *n* αντικείμενα	Το πλήθος *n των αντικειμένων*	Συγκρίσεις κλειδιών
Πολλαπλασιασμός πινάκων με πραγματικούς αριθμούς	Διαστάσεις των πινάκων	Πολλαπλασιασμός πραγματικών αριθμών
Υπολογισμός a^n	*n*	Πολλαπλασιασμός πραγματικών αριθμών
Προβλήματα με γράφους	Πλήθος κορυφών ή/και ακμών	Η επίσκεψη ενός κόμβου ή η διάσχιση μίας ακμής

Εμπειρική ανάλυση της χρονικής αποδοτικότητας

- Επιλέγουμε ένα συγκεκριμένο δείγμα εισόδου
- Χρησιμοποιούμε τη φυσική μονάδα μέτρησης (π.χ., milliseconds)

  ή

- Μετρούμε τον πραγματικό αριθμό επαναλήψεων της βασικής πράξης
- Αναλύουμε τα εμπειρικά δεδομένα

Καλύτερη, μέση, χειρότερη περίπτωση

Σε μερικούς αλγορίθμους η αποδοτικότητα εξαρτάται από τον τύπο της εισόδου:

- Χειρότερη περίπτωση: W(*n*) – το μέγιστο από όλες τις εισόδους μεγέθους *n*
- Καλύτερη περίπτωση: B(*n*) – το ελάχιστο από όλες τις εισόδους μεγέθους *n*
- Μέση περίπτωση: A(*n*) – ο μέσος όρος από όλες τις εισόδους μεγέθους *n*
  - Είναι το πλήθος εκτελέσεων της βασικής πράξης σε μία τυπική είσοδο
  - Όχι ο μέσος όρος της καλύτερης και της χειρότερης περίπτωσης
  - Η αναμενόμενη τιμή των επαναλήψεων της βασικής πράξης θεωρείται μία τυχαία μεταβλητή υπό την προϋπόθεση της γνώσης της κατανομής πιθανότητας όλων των δυνατών εισόδων μεγέθους *n*

Παράδειγμα: Σειριακή αναζήτηση (1)

- *Πρόβλημα:* Δοθείσης λίστας με *n* στοιχεία και ενός κλειδιού *K*, να βρεθεί ένα στοιχείο ίσο με *K*, αν υπάρχει.
- *Αλγόριθμος:* Σαρώνουμε τη λίστα και συγκρίνουμε διαδοχικά τα στοιχεία με την τιμή *K* μέχρι είτε να βρούμε μία τιμή που να ταιριάζει (*επιτυχής αναζήτηση*) ή να εξαντληθεί η λίστα (*ανεπιτυχής αναζήτηση*)

Παράδειγμα: Σειριακή αναζήτηση (2)

Algorithm Sequential(A[0..n-1])

```
// Input: An array A[0..n-1] and a sought value k
// Output: the position of k if found, -1 otherwise
i ← 0
while i<n and A[i]<>k do
   i ← i+1
If i<n return i
else return -1
```

- Χειρότερη περίπτωση
- Καλύτερη περίπτωση
- Μέση περίπτωση

Εκφράσεις για τη μέτρηση της βασικής πράξης

- Επακριβής τύπος

  π.χ., $C(n) = n(n-1)/2$

- Τύπος ενδεικτικός της τάξης μεγέθους με συγκεκριμένη πολλαπλασιαστική σταθερά

  π.χ., $C(n) \approx 0.5\ n^2$

- Τύπος ενδεικτικός της τάξης μεγέθους με άγνωστη πολλαπλασιαστική σταθερά

  π.χ., $C(n) \approx cn^2$

Τάξη μεγέθους (1)

- Σημείωση: η τάξη μεγέθους θεωρείται με κάποια σταθερά καθώς $n \to \infty$
- Παράδειγμα:
  - Πόσο ταχύτερα θα τρέξει ένας αλγόριθμος σε ένα υπολογιστή δύο φορές ταχύτερο?
  - Πόσο περισσότερο χρόνο θα χρειασθεί η επίλυση προβλήματος με διπλάσιο μέγεθος εισόδου?

Τάξη μεγέθους (2)

n	$\log_2 n$	n	$n\log_2 n$	n^2	n^3	2^n	n!
10	3.3	10^1	$3.3\cdot10^1$	10^2	10^3	10^3	$3.6\cdot10^6$
10^2	6.6	10^2	$6.6\cdot10^2$	10^4	10^6	$1.3\cdot10^{30}$	$9.3\cdot10^{157}$
10^3	10	10^3	$1.0\cdot10^4$	10^6	10^9		
10^4	13	10^4	$1.3\cdot10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7\cdot10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2.0\cdot10^7$	10^{12}	10^{18}		

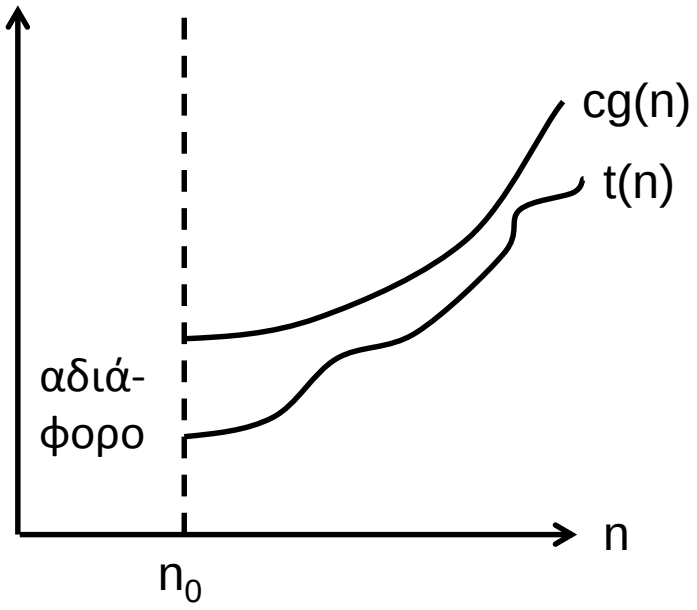
Τιμές μερικών σημαντικών συναρτήσεων για την ανάλυση αλγορίθμων

Ασυμπτωτικός λόγος αύξησης

- Ένας τρόπος σύγκρισης συναρτήσεων που αγνοεί τους σταθερούς παράγοντες και τα μικρά μεγέθη εισόδου
- $O(g(n))$: η κλάση των συναρτήσεων $f(n)$ που μεγαλώνουν *όχι γρηγορότερα* από τη $g(n)$
- $\Theta(g(n))$: η κλάση των συναρτήσεων $f(n)$ που μεγαλώνουν *με τον ίδιο ρυθμό* όπως η $g(n)$
- $\Omega(g(n))$: η κλάση των συναρτήσεων $f(n)$ που μεγαλώνουν *τουλάχιστον τόσο γρήγορα* όπως η $g(n)$

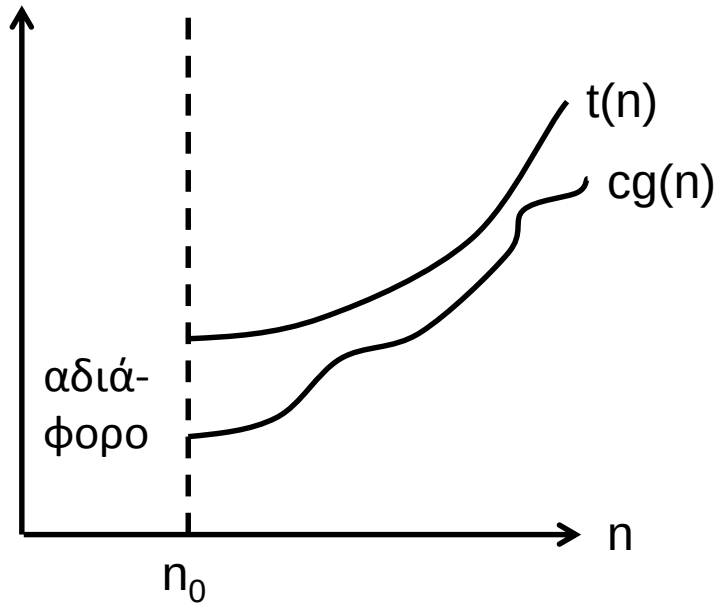
Δες επόμενα σχήματα

Συμβολισμός Ο



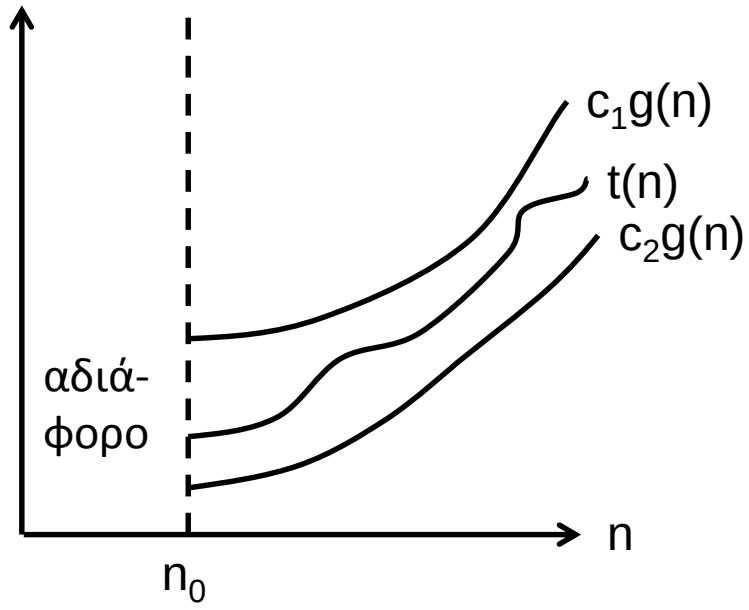
$t(n) \in O(g(n))$

Συμβολισμός Ω



$t(n) \in \Omega(g(n))$

Συμβολισμός Θ



$t(n) \in \Theta(g(n))$

Εύρεση ρυθμού αύξησης: Μέθοδος 1 – με όρια

$\lim_{n\to\infty} T(n)/g(n) =$

- **0** τάξη αύξησης της $T(n)$ < τάξη αύξησης της $g(n)$
- $c>0$ τάξη αύξησης της $T(n)$ = τάξη αύξησης της $g(n)$
- ∞ τάξη αύξησης της $T(n)$ > τάξη αύξησης της $g(n)$

Παραδείγματα:

- $10n$ vs. $2n^2$
- $n(n+1)/2$ vs. n^2
- $\log_b n$ vs. $\log_c n$

Κανόνας του L'Hôpital

- Αν $lim_{n\to\infty} f(n) = lim_{n\to\infty} g(n) = \infty$ και υπάρχουν οι παράγωγοι f' και g', τότε ισχύει

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

- Παράδειγμα: *logn* vs. *n*

Εύρεση ρυθμού αύξησης: Μέθοδος 2 – με τον ορισμό

- *Η f*(*n*) είναι O(*g*(*n*)) αν η τάξη αύξησης της $f(n) \leq$ τάξης αύξησης της $g(n)$ (εντός μίας σταθεράς)
- Υπάρχει μία θετική σταθερά c και ένας μη αρνητικός ακέραιος n_0 τέτοιος ώστε να ισχύει

$$f(n) \leq c\, g(n) \text{ για κάθε } n \geq n_0$$

Παραδείγματα:

- $10n$ είναι $O(2n^2)$
- $5n+20$ είναι $O(10n)$

Βασικές ασυμπτωτικές κλάσεις αποδοτικότητας

1	σταθερή
log *n*	λογαριθμική
n	γραμμική
n log *n*	*n* log *n*
n^2	τετραγωνική
n^3	κυβική
2^n	εκθετική
n!	παραγοντική

Χρονική αποδοτικότητα επαναληπτικών αλγορίθμων

Βήματα της μαθηματικής ανάλυσης επαναληπτικών αλγορίθμων:

- Αποφασίζουμε την παράμετρο *n* που δείχνει το *μέγεθος της εισόδου*
- Προσδιορίζουμε τη *βασική πράξη* του αλγορίθμου
- Προσδιορίζουμε τη *χειρότερη*, τη *μέση*, και την *καλύτερη* περίπτωση για μία είσοδο μεγέθους *n*
- Βρίσκουμε την έκφραση *C(n)* με βάση τη δομή βρόχου του αλγορίθμου
- Απλοποιούμε την έκφραση με τη βοήθεια τυποποιημένων σχέσεων

Ένα απλό παράδειγμα

Algorithm MaxElement(A[0..n-1])

```
// Determines the largest value in a given array
// Input: An array A[0..n-1] of real numbers
// Output: The value of the largest element in A
maxval ← A[0]
for i←1 to n-1 do
    if A[i]>maxval
       maxval ← A[i]
return maxval
```

Ένα άλλο απλό παράδειγμα

Algorithm UniqueElement(A[0..n-1])

```
// Check whether all the elements in an array are
    distinct
// Input: An array A[0..n-1] of real numbers
// Output: Returns "true" if all the elements are
    distinct
for i←0 to n-2 do
    for j←i+1 to n-1 do
       if A[i]=A[j] return false
return true
```

Άλλα παραδείγματα

- Πολλαπλασιασμός πινάκων
- Ταξινόμηση με επιλογή
- Ταξινόμηση με εισαγωγή
- Μυστηριώδης αλγόριθμος

Πολλαπλασιασμός πινάκων

Algorithm MatrixMultiplication(A[0..n-1,0..n-1], B[0..n-1,0..n-1], C[0..n-1,0..n-1])

```
// Input: Two n-by-n matrices A and B
// Output: Matrix C=AB
for i←0 to n-1 do
for j←0 to n-1 do
   C[i,j]← 0.0
   for k ← 0 to n-1 do
     C[i,j] ← C[i,j]+A[i,k]*B[k,j]
return C
```

‘Ενα τρίτο απλό παράδειγμα

Algorithm Binary(n)

```
// Input: A positive decimal integer n
// Output: The number of binary digits in
n's binary representation
count←1
while n>1 do
      count ← count+1
      n ← ⌊n/2⌋
return count
```

Ταξινόμηση με επιλογή

Algorithm SelectionSort(A[0..n-1])

```
// Input: An array A[0..n-1] of unordered elements
// Output: Array A[0..n-1] in ascending 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]
```

Ταξινόμηση με εισαγωγή

Algorithm InsertionSort(A[0..n-1])

```
// Input: An array A[0..n-1] of unordered elements
// Output: Array A[0..n-1] in ascending order
for i←0 to n-1 do
        v ← A[i]
        j ← i-1
        while j≥0 and A[j] > v do
                A[j+1]<- A[j]
                j ← j-1
        A[j+1] ← v
```

Μυστηριώδης αλγόριθμος

```
for i := 1 to n-1 do
   max := i;
   for j := i+1 to n do
      if |A[j,i]|>|A[max,i]| then max := j;
   for k := i to n+1 do
      swap A[i,k] with A[max,k];
   for j := i+1 to n do
      for k := n+1 downto i do
          A[j,k]:=A[j,k]-
           A[i,k]*A[j,i]/A[i,i];
```

Χρονική αποδοτικότητα αναδρομικών αλγορίθμων

Βήματα της μαθηματικής ανάλυσης αναδρομικών αλγορίθμων:

- Αποφασίζουμε την παράμετρο *n* που δείχνει το *μέγεθος της εισόδου*
- Προσδιορίζουμε τη *βασική πράξη* του αλγορίθμου
- Προσδιορίζουμε τη *χειρότερη*, τη *μέση*, και την *καλύτερη* περίπτωση για μία είσοδο μεγέθους *n*
- Σχηματίζουμε για τη $C(n)$ μία αναδρομική εξίσωση με αρχικές συνθήκες – το πλήθος των επαναλήψεων που θα εκτελεσθεί η βασική πράξη για μία είσοδο μεγέθους *n* (εναλλακτικά μπορούμε να μετρήσουμε τις αναδρομικές κλήσεις).
- Επιλύουμε την αναδρομική εξίσωση για να πετύχουμε μία κλειστή έκφραση ή να εκτιμήσουμε το μέγεθος της λύσης

Αναδρομικός υπολογισμός του *n*!

- Ορισμός: *n*! = 1*2*…*(*n*-1)**n*
- Αναδρομικός ορισμός του *n*!: *n*! = (*n*-1)!**n*
- Αλγόριθμος:

```
if n=0 then F(n) := 1
   else F(n) := F(n-1) * n
return F(n)
```

- Αναδρομή για το πλήθος των πολλαπλασιασμών:

  M(n) = M(n-1) + 1 for n>0

  M(0) = 0

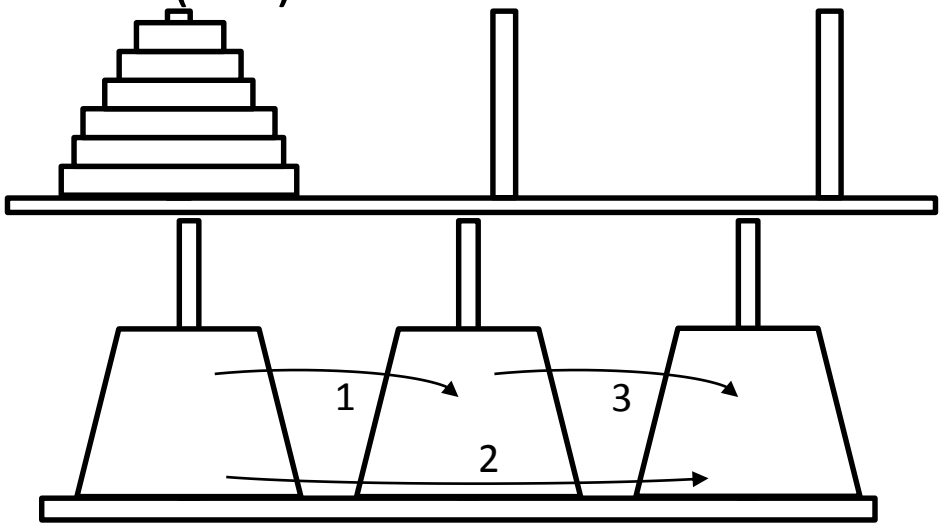
- Μέθοδος: αντικατάσταση προς τα πίσω

Οι πύργοι του Ανόι

- Πρόβλημα: να μετρήσουμε το πλήθος των κινήσεων
- Αναδρομή για το πλήθος των κινήσεων:

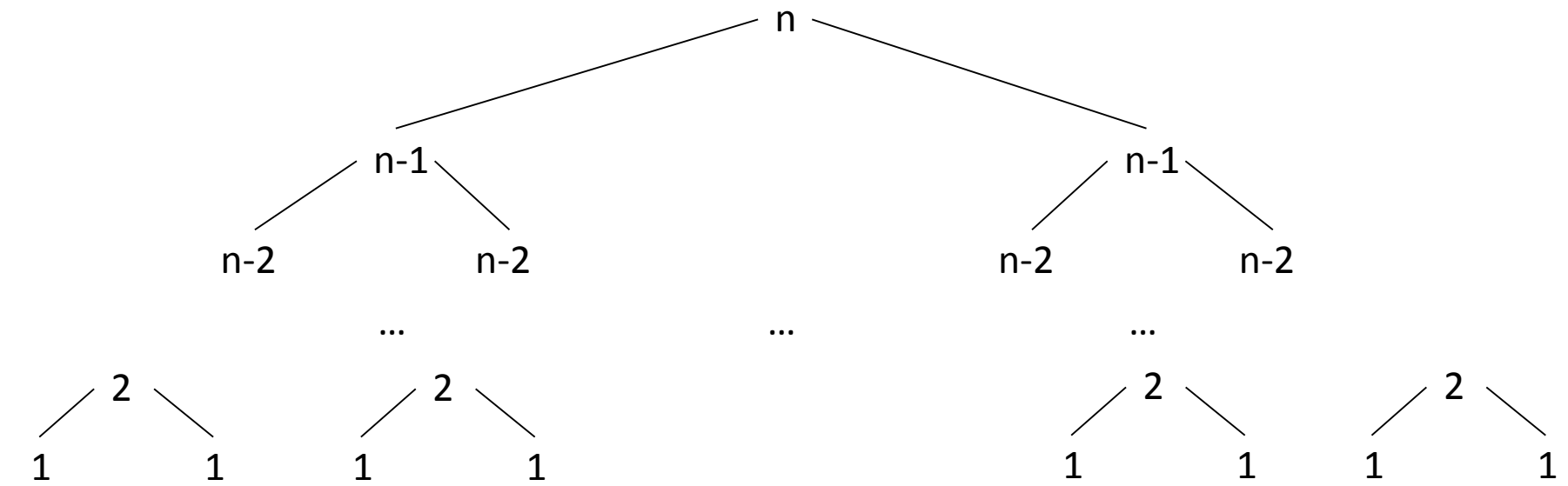
$M(n) = M(n-1) + 1 + M(n-1)$ for $n>1$

$M(1) = 1$



Οι πύργοι του Ανόι (συνέχεια)

- Εναλλακτικά: σχηματίζουμε ένα δένδρο για το πλήθος των αναδρομικών κλήσεων
- Σχόλιο: εγγενής μη αποδοτικότητα



Επανεξέταση του «δυαδικού» αλγόριθμου

- **Algorithm BinRec(n)**

```
// Input: A positive decimal integer n
// Output: The number of binary digits in n's
    binary representation
if n=1 return 1
else return BinRec(⌊n/2⌋)+1
```

- Αναδρομή για το πλήθος των προσθέσεων:

$A(n) = A(\lfloor n/2 \rfloor) + 1$ for $n>1$

$A(1) = 0$

- Εφαρμόζουμε την απλοποίηση: $n=2^k$

Υπολογισμός αριθμών Fibonacci (1)

- Leonardo Fibonacci, 1202.
- Ακολουθία Fibonacci : 0, 1, 1, 2, 3, 5, 8, 13, 21, …
- Αναδρομή Fibonacci :

  $F(n)=F(n-1)+F(n-2),\ F(0)=0,\ F(1)=1$
- **Algorithm F(n):**

```
// Computes the n-th Fibonacci number
   recursively
// Input: A nonnegative integer n
// Output: The n-th Fibonacci number
if n <= 1 return n
else return F(n-1)+F(n-2)
```

Υπολογισμός αριθμών Fibonacci (2)

1. Αναδρομικός αλγόριθμος με βάση τον ορισμό
2. Επαναληπτικός ορισμός με ωμή βία
3. Αλγόριθμος με βοήθεια ρητής έκφρασης
4. Αλγόριθμος βασισμένος σε πολλαπλασιασμό πινάκων

Υπολογισμός αριθμών Fibonacci (3)

Αναδρομικός αλγόριθμος με βάση τον ορισμό

- Γραμμική ομογενής αναδρομική σχέση $2^{ου}$ βαθμού με σταθερούς συντελεστές
- Χαρακτηριστική εξίσωση
- Επίλυση

  $F(n) = \phi^n/\sqrt{5} - \phi'^n/\sqrt{5}$

  $\phi = (1+\sqrt{5})/2$ (golden ratio) $\phi' = (1-\sqrt{5})/2$

  $F(n) \approx \phi^n/\sqrt{5}$
- Κατασκευάζουμε ένα δένδρο για το πλήθος των αναδρομικών κλήσεων
- Οι λύσεις αναδρομικών εξισώσεων δεν είναι πανάκεια

Υπολογισμός αριθμών Fibonacci (4)

Επαναληπτικός ορισμός με ωμή βία

- **Algorithm F(n):**

```
// Computes the n-th Fibonacci number
   iteratively
// Input: A nonnegative integer n
// Output: The n-th Fibonacci number
F[0] ← 0; F[1]← 1;
for i ← 2 to n do
     F[i] ← F[i-1] + F[i-2]
return F(n)
```

- Πολυπλοκότητα ?

Υπολογισμός αριθμών Fibonacci (5)

Αλγόριθμος με βοήθεια ρητής έκφρασης

- Με έξυπνη στρογγυλοποίηση της εξίσωσης

  $F(n) \approx \phi^n/\sqrt{5}$

- Επαφίεται στον εκθετικό αλγόριθμο
- Λύσεις: Θ(n), Θ(logn)

Υπολογισμός αριθμών Fibonacci (6)

Αλγόριθμος με πολλαπλασιασμό πινάκων

- Ισχύει ότι

$$\begin{pmatrix} F(n-1) & F(n) \\ F(n) & F(n+1) \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n$$

για $n \geq 1$,

- Υποθέτοντας έναν αποτελεσματικό τρόπο πολλαπλασιασμού δυνάμεων πινάκων, η λύση είναι Θ(logn)

Σημαντικοί τύποι αναδρομών

- Μία πράξη μειώνει το μέγεθος του προβλήματος κατά 1.

  $T(n) = T(n\text{-}1) + c$ $\quad T(1) = d$

  Λύση: $T(n) = (n\text{-}1)c + d$ $\quad$ *γραμμική*

- Ένα πέρασμα από την είσοδο μειώνει το πρόβλημα κατά ένα.

  $T(n) = T(n\text{-}1) + cn$ $\quad T(1) = d$

  Λύση: $T(n) = [n(n+1)/2 - 1]\, c + d$ $\quad$ *τετραγωνική*

- Μία πράξη μειώνει το μέγεθος του προβλήματος κατά το μισό.

  $T(n) = T(n/2) + c$ $\quad T(1) = d$

  Λύση: $T(n) = c \lg n + d$ $\quad$ *λογαριθμική*

- Ένα πέρασμα από την είσοδο μειώνει το πρόβλημα κατά το μισό.

  $T(n) = 2T(n/2) + cn$ $\quad T(1) = d$

  Λύση: $T(n) = cn \lg n + d\, n$ $\quad n \log n$

Μία γενική αναδρομή διαίρει-και-βασίλευε

$T(n) = aT(n/b) + f(n)$ όπου $f(n) \in \Theta(n^k)$

1. $a < b^k$ $T(n) \in \Theta(n^k)$
2. $a = b^k$ $T(n) \in \Theta(n^k \lg n)$
3. $a > b^k$ $T(n) \in \Theta(n^{\log_b a})$

Σημείωση: το ίδιο ισχύει και για Ο αντί για Θ.

Ποια η κλάση αποδοτικότητας

size	1000	2000	3000	4000	5000	6000	7000	8000	9000	10000
count	11966	24303	39992	53010	67272	78692	91274	113063	129799	140538

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Τέλος ενότητας

Επεξεργασία: Ανδρέας Κοσματόπουλος

Θεσσαλονίκη, Αύγουστος 2015

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημειώματα

Σημείωμα Ιστορικού Εκδόσεων Έργου

Το παρόν έργο αποτελεί την έκδοση 1.00.

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.