ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΛΓΟΡΙΘΜΟΙ

Υπολογισμός Παραγοντικού (1/3)

Μαθηματικός τύπος:

$$n! = 1 \cdot \ldots \cdot (n-1) \cdot n = (n-1)! \cdot n \quad \text{for } n \geq 1$$

Βασική λειτουργία: πολ/μος

Το πλήθος των πολ/μων έχει ως εξής:

$$M(n) = \underbrace{M(n-1)}_{\substack{\text{to compute} \\ F(n-1)}} + \underbrace{1}_{\substack{\text{to multiply} \\ F(n-1) \text{ by } n}} \quad \text{for } n > 0$$

Κάθε φορά το πλήθος εξαρτάται από το πλήθος στο n-1

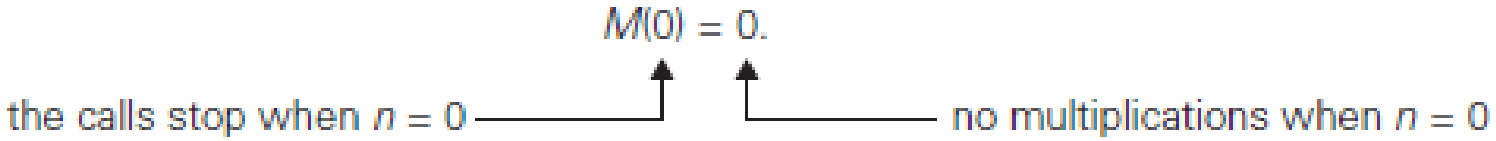
ALGORITHM $F(n)$

//Computes $n!$ recursively
//Input: A nonnegative integer n
//Output: The value of $n!$
if $n = 0$ **return** 1
else return $F(n-1) * n$

Υπολογισμός Παραγοντικού (2/3)

Η αρχική συνθήκη βρίσκεται στο if

Όταν ν=0 ο αλγόριθμος δεν κάνει πολ/μους



Εφαρμόζουμε τη μέθοδο των ***backward substitutions***

$$M(n) = M(n-1) + 1 \qquad \text{substitute } M(n-1) = M(n-2) + 1$$
$$= [M(n-2) + 1] + 1 = M(n-2) + 2 \qquad \text{substitute } M(n-2) = M(n-3) + 1$$
$$= [M(n-3) + 1] + 2 = M(n-3) + 3.$$

Υπολογισμός Παραγοντικού (3/3)

Γενικός τύπος

$$M(n) = M(n-i) + i$$

Αφού η αρχική συνθήκη είναι n=0, αντικαθιστούμε με i=n και έχουμε:

$$M(n) = M(n-1) + 1 = \cdots = M(n-i) + i = \cdots = M(n-n) + n = n$$

Complexity → O(n)

Μεθοδολογία

Η γενική μεθοδολογία για την ανάλυση αναδρομικών αλγορίθμων έχει ως εξής:

- **Αποφασίζουμε** την παράμετρο του μεγέθους της εισόδου
- **Αναγνωρίζουμε** τη βασική λειτουργία του αλγορίθμου (π.χ. σύγκριση, ανάθεση τιμής)
- **Ελέγχουμε** αν το πλήθος εκτέλεσης της βασικής λειτουργίας εξαρτάται μόνο από το μέγεθος εισόδου. Αν εξαρτάται από κάποια επιπλέον παράμετρο τότε μελετούμε ξεχωριστά την καλύτερη, τη χειρότερη και τη μέση περίπτωση.
- **Δημιουργούμε** μια αναδρομική σχέση με μια κατάλληλη αρχική συνθήκη που απεικονίζει το πλήθος των εκτελέσεων της βασικής λειτουργίας.
- **Επιλύουμε** την αναδρομική σχέση.

Backward Substitution

Δύο βασικά βήματα:

- Περιγράφουμε τη μορφή της λύσης
- Χρησιμοποιούμε επαγωγή για να βρούμε σταθερές που αποδεικνύουν ότι η λύση ισχύει

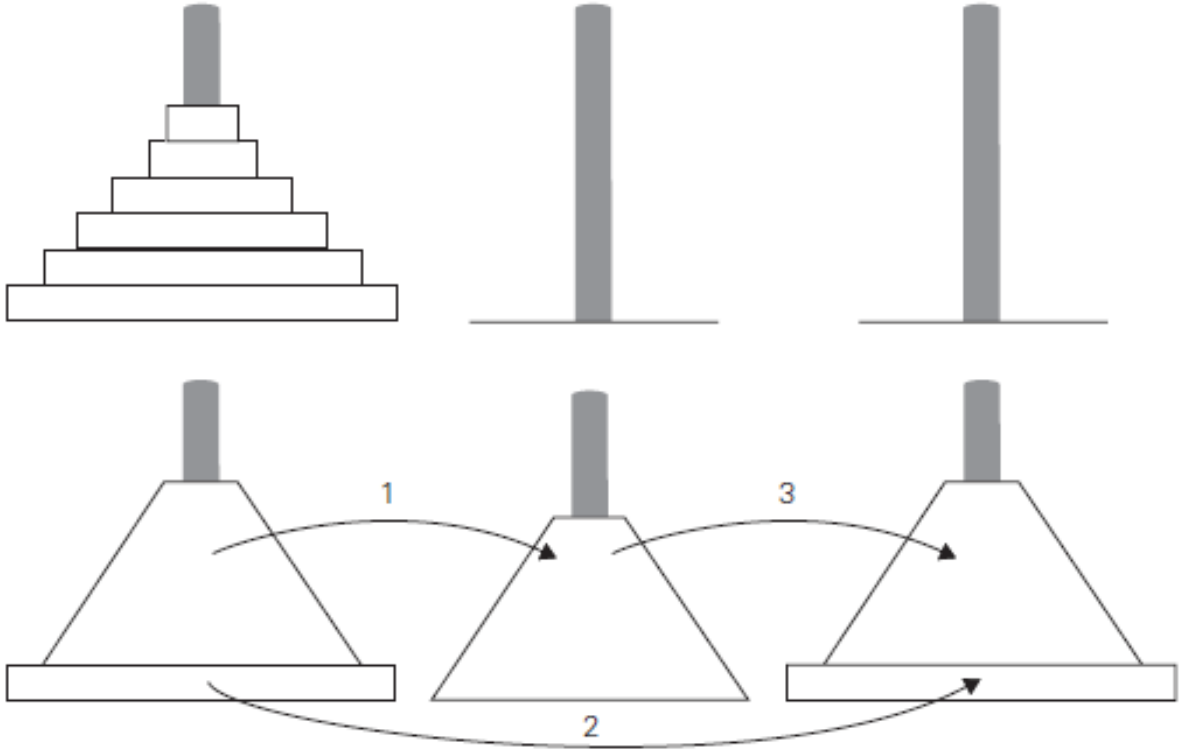
Η λύση που προτείνουμε είναι μια συνάρτηση όπου εφαρμόζουμε την επαγωγή

Οι Πύργοι του Ανόϊ (1/5)

Έχουμε n δίσκους διαφορετικού μεγέθους

Οι δίσκοι πρέπει να μεταφερθούν από τον 1ο στύλο στον 3ο αλλά:

- Ένας δίσκος κάθε φορά μπορεί να μεταφερθεί
- Μεγαλύτερος δίσκος δεν πρέπει να τοποθετηθεί πάνω από μικρότερο



Οι Πύργοι του Ανόϊ (2/5)

Για την μετακίνηση n δίσκων πρώτα μεταφέρουμε αναδρομικά n-1 δίσκους από τον 1ο στύλο στον 2ο (με τον 3ο στύλο βοηθητικό) και έπειτα μεταφέρουμε το μεγαλύτερο δίσκο στον 3ο στύλο

Στη συνέχεια μεταφέρουμε αναδρομικά n-1 δίσκους από τον 2ο δίσκο στον 3ο (με τον 1Ο δίσκο βοηθητικό)

Όταν n=1, απλά μετακινούμε το δίσκο από τον στύλο ‘πηγή’ στο στύλο ‘προορισμό’

Το μέγεθος εισόδου είναι n

Βασική λειτουργία: μετακίνηση / μεταφορά δίσκου

Οι Πύργοι του Ανόϊ (3/5)

Η αναδρομική σχέση είναι

$$M(n) = M(n-1) + 1 + M(n-1) \quad \text{for } n > 1.$$

Έχουμε:

$$M(n) = 2M(n-1) + 1 \quad \text{for } n > 1$$

$$M(1) = 1$$

Εφαρμόζοντας backward substitution παίρνουμε:

$$M(n) = 2M(n-1) + 1 \qquad \text{sub. } M(n-1) = 2M(n-2) + 1$$

$$= 2[2M(n-2) + 1] + 1 = 2^2 M(n-2) + 2 + 1 \quad \text{sub. } M(n-2) = 2M(n-3) + 1$$

$$= 2^2[2M(n-3) + 1] + 2 + 1 = 2^3 M(n-3) + 2^2 + 2 + 1$$

Οι Πύργοι του Ανόϊ (4/5)

Η σχέση που φαίνεται μέσα από την αναδρομική σχέση είναι:

$$M(n) = 2^i M(n-i) + 2^{i-1} + 2^{i-2} + \cdots + 2 + 1 = 2^i M(n-i) + 2^i - 1$$

n-i = 1 → i=n-1

Η αρχική συνθήκη ισχύει για n=1 και την έχουμε για i=n-1 έχουμε:

$$M(n) = 2^{n-1} M(n-(n-1)) + 2^{n-1} - 1$$

$$= 2^{n-1} M(1) + 2^{n-1} - 1 = 2^{n-1} + 2^{n-1} - 1 = 2^n - 1$$

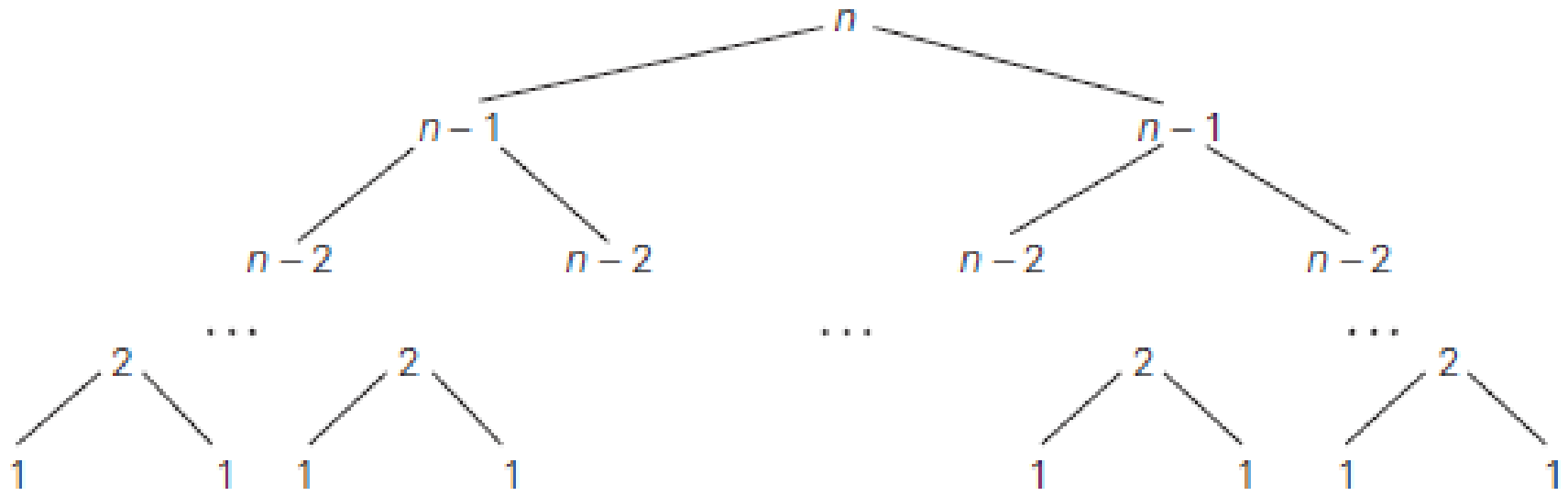
Ένα δένδρο που δείχνει τα βήματα της αναδρομής μπορεί να βοηθήσει στην επίλυση

Complexity $O(2^n)$

Οι Πύργοι του Ανόϊ (5/5)

Πλήθος κόμβων στο δένδρο: $$C(n) = \sum_{l=0}^{n-1} 2^l$$

Όπου l είναι τα επίπεδα του δένδρου



Εύρεση Ψηφίων Αναδρομικά (1/3)

Στόχος: εύρεση αναδρομικής σχέσης

Πλήθος προσθέσεων: $A(\lfloor n/2 \rfloor) + 1$

Αναδρομική σχέση:

$$A(n) = A(\lfloor n/2 \rfloor) + 1 \quad \text{for } n > 1.$$

Πρόβλημα:

- Η μέθοδος backward substitution είναι δύσκολο να εφαρμοστεί για άλλους αριθμούς εκτός από τα πολλαπλάσια του 2

ALGORITHM $BinRec(n)$

//Input: A positive decimal integer n
//Output: The number of binary digits in n's
if $n = 1$ **return** 1
else return $BinRec(\lfloor n/2 \rfloor) + 1$

Εύρεση Ψηφίων Αναδρομικά (2/3)

Επιλύουμε το πρόβλημα για τα πολλαπλάσια του 2 και εφαρμόζουμε τον **κανόνα εξομάλυνσης (smoothness rule)**

Έχουμε:

$$A(2^k) = A(2^{k-1}) + 1 \quad \text{for } k > 0$$
$$A(2^0) = 0.$$

και με backward substitution παίρνουμε:

$$\begin{aligned}
A(2^k) &= A(2^{k-1}) + 1 && \text{substitute } A(2^{k-1}) = A(2^{k-2}) + 1 \\
&= [A(2^{k-2}) + 1] + 1 = A(2^{k-2}) + 2 && \text{substitute } A(2^{k-2}) = A(2^{k-3}) + 1 \\
&= [A(2^{k-3}) + 1] + 2 = A(2^{k-3}) + 3 && \ldots \\
&\ldots \\
&= A(2^{k-i}) + i \\
&\ldots \\
&= A(2^{k-k}) + k
\end{aligned}$$

Εύρεση Ψηφίων Αναδρομικά (3/3)

Τελικά καταλήγουμε στο:

$$A(2^k) = A(1) + k = k$$

και επιστρέφοντας στη συλλογιστική μας για οποιοδήποτε n έχουμε:

$$A(n) = \log_2 n \in \Theta(\log n)$$

αφού $2^k = n$

Ουσιαστικά

$$A(n) = \lfloor \log_2 n \rfloor$$

The Master Theorem (1/7)

Μας προσφέρει μια μεθοδολογία για να επιλύουμε αναδρομικές εξισώσεις της μορφής:

$$T(n) = aT(n/b) + f(n)$$

με α>=1 και b>1 να είναι σταθερές και f(n) είναι μια ασυμπτωτικά θετική συνάρτηση

Η παραπάνω συνάρτηση χωρίζει το πρόβλημα μεγέθους n σε α υπο-προβλήματα μεγέθους n/b

Τα α υπο-προβλήματα λύνονται πάλι αναδρομικά σε χρόνο n/b

Η f(n) απεικονίζει το κόστος της διαίρεσης του προβλήματος και της συγχώνευσης των λύσεων

The Master Theorem (2/7)

Το n/b μπορεί να μην είναι ακέραιος

Παίρνουμε το floor ή το ceiling του n/b χωρίς να επηρεάζεται η ασυμπτωτική συμπεριφορά

Θα πρέπει να θυμόμαστε τρεις περιπτώσεις

Υπάρχουν κενά στη μέθοδο αφού η f(n) πρέπει να είναι πολυωνυμικά μικρότερη / μεγαλύτερη από το $n^{\log_b a}$

Αν είναι τότε δεν μπορούμε να χρησιμοποιήσουμε το master theorem για την επίλυση αναδρομικών σχέσεων

The Master Theorem (3/7)

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n) ,$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$.

The Master Theorem (4/7)

Ας δούμε ένα παράδειγμα:

Για την $T(n) = 9T(n/3) + n$

έχουμε $a = 9,\ b = 3,\ f(n) = n,$

και έτσι $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$

Αφού $f(n) = O(n^{\log_3 9 - \epsilon})$

με $\epsilon = 1$

Εφαρμόζουμε την 1η περίπτωση και καταλήγουμε στο

$$T(n) = \Theta(n^2)$$

The Master Theorem (5/7)

Παράδειγμα

Για την $T(n) = T(2n/3) + 1$

έχουμε $a = 1, b = 3/2, f(n) = 1$

και έτσι $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$

Αφού $f(n) = \Theta(n^{\log_b a}) = \Theta(1)$

Εφαρμόζουμε τη 2η περίπτωση και καταλήγουμε στο

$$T(n) = \Theta(\lg n)$$

The Master Theorem (6/7)

Παράδειγμα

Για την $T(n) = 3T(n/4) + n \lg n$

έχουμε $a = 3,\ b = 4,\ f(n) = n \lg n$

και έτσι $n^{\log_b a} = n^{\log_4 3} = O(n^{0.793})$

Αφού $f(n) = \Omega(n^{\log_4 3 + \epsilon})$

με $\epsilon \approx 0.2$

για μεγάλο n έχουμε $af(n/b) = 3(n/4)\lg(n/4) \leq (3/4)n \lg n = cf(n)$ for $c = 3/4$

Εφαρμόζουμε τη 3η περίπτωση και καταλήγουμε στο $T(n) = \Theta(n \lg n)$

The Master Theorem (7/7)

Παράδειγμα

Για την $T(n) = 2T(n/2) + n \lg n$

η master μέθοδος δεν μπορεί να εφαρμοστεί αφού για

$a = 2,\ b = 2,\ f(n) = n \lg n,$ και $n^{\log_b a} = n$

Δεν μπορεί να εφαρμοστεί η 3η περίπτωση αφού η f(n) είναι ασυμπτωτικά μεγαλύτερη από την $n^{\log_b a} = n$ αλλά όχι πολυωνυμικά μεγαλύτερη. Ο λόγος $f(n)/n^{\log_b a} = (n \lg n)/n = \lg n$

Είναι ασυμπτωτικά μικρότερος από το n^{ϵ}

για οποιοδήποτε ε

Ασκήσεις (1/2)

1. $T(n) = 3T(n/2) + n^2$
2. $T(n) = 4T(n/2) + n^2$
3. $T(n) = T(n/2) + 2^n$
4. $T(n) = 2^n T(n/2) + n^n$
5. $T(n) = 16T(n/4) + n$
6. $T(n) = 2T(n/2) + n \log n$

---

1. $T(n) = 3T(n/2) + n^2 \Longrightarrow T(n) = \Theta(n^2)$ (Case 3)
2. $T(n) = 4T(n/2) + n^2 \Longrightarrow T(n) = \Theta(n^2 \log n)$ (Case 2)
3. $T(n) = T(n/2) + 2^n \Longrightarrow \Theta(2^n)$ (Case 3)
4. $T(n) = 2^n T(n/2) + n^n \Longrightarrow$ Does not apply (a is not constant)
5. $T(n) = 16T(n/4) + n \Longrightarrow T(n) = \Theta(n^2)$ (Case 1)
6. $T(n) = 2T(n/2) + n \log n \Longrightarrow T(n) = n \log^2 n$ (Case 2)

Ασκήσεις (2/2)

7. $T(n) = 2T(n/2) + n/\log n$

8. $T(n) = 2T(n/4) + n^{0.51}$

9. $T(n) = 0.5T(n/2) + 1/n$

10. $T(n) = 16T(n/4) + n!$

11. $T(n) = \sqrt{2}T(n/2) + \log n$

12. $T(n) = 3T(n/2) + n$

7. $T(n) = 2T(n/2) + n/\log n \implies$ Does not apply (non-polynomial difference between $f(n)$ and $n^{\log_b a}$)

8. $T(n) = 2T(n/4) + n^{0.51} \implies T(n) = \Theta(n^{0.51})$ (Case 3)

9. $T(n) = 0.5T(n/2) + 1/n \implies$ Does not apply ($a < 1$)

10. $T(n) = 16T(n/4) + n! \implies T(n) = \Theta(n!)$ (Case 3)

11. $T(n) = \sqrt{2}T(n/2) + \log n \implies T(n) = \Theta(\sqrt{n})$ (Case 1)

12. $T(n) = 3T(n/2) + n \implies T(n) = \Theta(n^{\lg 3})$ (Case 1)