# Διαίρει και Βασίλευε

Στην ενότητα αυτή θα μελετηθούν τα εξής θέματα:

*Η Μέθοδος Σχεδιασμού Αλγορίθμων 'Διαίρει και Βασίλευε'*

*Επίλυση Αναδρομικών Εξισώσεων*

# Διαίρει και Βασίλευε

- Μέθοδος σχεδιασμού αλγορίθμων.
- Βασίζεται στην ιδιότητα αναδρομικότητας που έχουν οι λύσεις πολλών προβλημάτων
- Πρόσφορη για υλοποίηση αλγορίθμων, σε προγραμματιστικές γλώσσες που υποστηρίζουν την αναδρομή.
- Κάθε αλγόριθμος του τύπου "διαίρει και βασίλευε" περιλαμβάνει τρεις φάσεις:
  1. *Διαιρούμε το πρόβλημα σε ένα αριθμό από υποπροβλήματα.*
  2. *Κατακτούμε τα υποπροβλήματα λύνοντας τα αναδρομικά.*
  3. *Συνδυάζουμε τις λύσεις των υποπροβλημάτων για να βρούμε τη λύση του αρχικού προβλήματος.*
- Πρέπει να υπάρχει βασική περίπτωση.
- Παραδείγματα: Quicksort, Mergesort.

# Πολλαπλασιασμός Πινάκων

## Ο Αλγόριθμος του Strassen

- Πρόβλημα: Έστω δύο n×n πίνακες

$$A = [a_{ij}] \quad \kappa\alpha\iota \quad B = [b_{ij}], \qquad 1 \le i, j \le n$$

- Θέλουμε να υπολογίσουμε το γινόμενο C=A·B.

- Ο πίνακας C είναι n×n και τέτοιος ώστε

$$C = [c_{ij}] \quad o\pi o\upsilon \quad c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

- Πόσος χρόνος χρειάζεται για να υπολογίσουμε τον πίνακα C σύμφωνα με τον πιο πάνω ορισμό;

  $\Theta(n^3)$, n πολλαπλασιασμοί και n-1 προσθέσεις για κάθε ένα από τα $n^2$ στοιχεία.

# Πολλαπλασιασμός Πινάκων

- Μπορούμε να βελτιώσουμε το $\Theta(n^3)$;
- Βασική ιδέα: πολλαπλασίασε $2\times 2$ πίνακες χρησιμοποιώντας μόνο 7 πολλαπλασιασμούς (αντί για 8).

Έστω

$$\begin{pmatrix} r & s \\ t & u \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} e & g \\ f & h \end{pmatrix}$$

- Ορίζουμε

$$p_1 = a \cdot (g - h) \qquad p_5 = (a + d) \cdot (e + h)$$

$$p_2 = (a + b) \cdot h \qquad p_6 = (b - d) \cdot (f + h)$$

$$p_3 = (c + d) \cdot e \qquad p_7 = (a - c) \cdot (e + g)$$

$$p_4 = d \cdot (f - e)$$

- Υπολογισμός των πιο πάνω τιμών απαιτεί 7 πολλαπλασιασμούς και 10 προσθέσεις/ αφαιρέσεις.

# Πολλαπλασιασμός Πινάκων

- Μπορούμε να αποδείξουμε ότι:

$$r = p_5 + p_4 - p_2 + p_6 \qquad t = p_3 + p_4$$

$$s = p_1 + p_2 \qquad u = p_5 + p_1 - p_3 - p_7$$

- (Σημείωση: άλλες 8 προσθέσεις/αφαιρέσεις)

***Επαλήθευση της ορθότητας της πρώτης πρότασης.***

- Έχουμε

$$\begin{aligned} r &= p_5 + p_4 - p_2 + p_6 \\ &= (a+d)\cdot(e+h) + d\cdot(f-e) - (a+b)\cdot h + (b-d)\cdot(f+h) \\ &= ae + ah + de + dh + df - de - ah - bh + bf + bh - df - dh \\ &= ae + bf \end{aligned}$$

όπως χρειάζεται.

- (Όμοια επαληθεύεται η ορθότητα των υπόλοιπων προτάσεων.)

# Πολλαπλασιασμός Πινάκων

- Άρα η μέθοδος αυτή μας στοιχίζει
  - 7 πολλαπλασιασμούς και
  - 18, προσθέσεις/αφαιρέσεις
- Πόσο καλή είναι η μέθοδος;
- Μας γλιτώνει ένα πολλαπλασιασμό, όμως μας στοιχίζει 14 επιπρόσθετες προσθέσεις/ αφαιρέσεις.
- Δεν αξίζει τον κόπο για πολλαπλασιασμό 2×2 πινάκων… όμως έχει γενικότερη χρησιμότητα.
- Παρατήρηση: δεν στηρίζεται στην αντιμεταθετικότητα της πράξης του πολλαπλασιασμού. Συνεπώς μπορεί να χρησιμοποιηθεί στην περίπτωση που τα a, b, c, d, e, f, g, και h, είναι πίνακες (ο πολλαπλασιασμός πινάκων δεν είναι αντιμεταθετική πράξη).

# Ο Αλγόριθμος του Strassen

- Αυτή η ιδέα μας δίνει τον αλγόριθμο του Strassen για πολλαπλασιασμό n×n πινάκων:

1. *(Διαίρεσε)* Μοίρασε τους πίνακες Α και Β σε 4 υποπίνακες n/2×n/2 ο καθένας.
2. *(Κατάκτησε)* Κάνε 7 πολλαπλασιασμούς πινάκων αναδρομικά.
3. *(Συνδύασε)* Υπολόγισε τον C=A·B χρησιμοποιώντας προσθέσεις και αφαιρέσεις. (πως;)

# Χρόνος Εκτέλεσης

- Έστω T(n) ο χρόνος εκτέλεσης του αλγόριθμου του Strassen πάνω σε πίνακες $n \times n$.

- Προφανώς

$$T(n) = 7\, T(n/2) + 18\, (n/2)^2$$

- Ποια είναι η βασική περίπτωση;

- Με βάση μεθόδους λύσης αναδρομικών εξισώσεων μπορούμε να αποδείξουμε ότι

$$\begin{aligned} T(n) &\in \Theta(n^{\lg 7}) \\ &\in O(n^3) \end{aligned}$$

- Άρα ο διαίρει και βασίλευε αλγόριθμος του Strassen βελτιώνει τον προφανή αλγόριθμο που βασίζεται στον ορισμό πολλαπλασιασμού πινάκων.

# Αναδρομικές Εξισώσεις

- Μας ενδιαφέρουν κυρίως αναδρομικές εξισώσεις του τύπου `διαίρει και βασίλευε`:

$$T(n) = \begin{cases} c, & \text{if } n = 1 \\ a \cdot T\left(\dfrac{n}{b}\right) + f(n), & \text{if } n > 1 \end{cases}$$

Όπου

| | |
|---|---|
| **T(n/b)** | *το κόστος για αναδρομική επίλυση υποπροβλημάτων μεγέθους n/b.* |
| **f(n)** | *το κόστος για συνδυασμό των λύσεων των υποπροβλημάτων* |
| **a≥1** | *ο αριθμός των υποπροβλημάτων* |
| **b>1** | *ο συντελεστής μείωσης του προβλήματος* |

- Πιο κάτω παρουσιάζονται γενικές τεχνικές για την επίλυση αναδρομικών εξισώσεων.

# Μέθοδος της επαγωγής

1. Προβλέπουμε μια συνάρτηση f(n) ως λύση της εξίσωσης, και
2. Επαληθεύουμε τη λύση με επαγωγή προσδιορίζοντας κατάλληλα τις σταθερές.

*<u>Παράδειγμα</u>*

Έχουμε την αναδρομική εξίσωση

$T(n) = 2 \cdot T(n/2) + n, \qquad n \geq 2$

$T(1) = 1$

*Προβλέπουμε ότι $T(n) \in O(n^2)$. Δηλαδή, θα πρέπει να υπάρχουν σταθερά c και τιμή m τέτοιες ώστε για κάθε $n>m$*

$T(n) \leq cn^2$

Θα αποδείξουμε το πιο πάνω επαγωγικά.

# Μέθοδος της επαγωγής: Παράδειγμα

*Υπόθεση της επαγωγής*: Έστω ότι $T(k) \le ck^2$ για κάθε $m<k<n$.

Θα αποδείξουμε ότι $T(n) \le cn^2$ προσδιορίζοντας κατάλληλα τη σταθερά c.

Εχουμε:

$$
\begin{aligned}
T(n) &= 2\cdot T(n/2) + n \\
&\le 2\cdot c(n/2)^2 + n \\
&= (c/2)\cdot n^2 + n \\
&= c\, n^2 - ((c/2)\cdot n^2 - n)
\end{aligned}
$$

Άρα $T(n) \le cn^2$ αν $(c/2)\cdot n^2 - n \ge 0$.

Δηλαδή, αν $c \ge 2/n$ $(n>0)$.

Για να ισχύει η ανισότητα αρκεί $c \ge 2$.

# Μέθοδος της επαγωγής: Παράδειγμα

Αποδείξαμε ότι $T(n) \in O(n^2)$. Μπορούμε όμως να βρούμε ακριβέστερη λύση…

*Προβλέπουμε ότι $T(n) \in O(n \lg n)$. Δηλαδή, θα αποδείξουμε την ύπαρξη σταθεράς c και τιμής m τέτοιες ώστε για κάθε $n>m$*

$$T(n) \leq c \cdot n \lg n$$

Θα αποδείξουμε το πιο πάνω επαγωγικά.

*Βάση της επαγωγής*: για n=1, το ζητούμενο ισχύει για οποιαδήποτε τιμή $c \geq 2$.

*Υπόθεση της επαγωγής*: Έστω ότι $T(k) \leq c \cdot k \lg k$ για κάθε $k<n$.

Θα αποδείξουμε ότι $T(n) \leq c \cdot n \lg n$ προσδιορίζοντας κατάλληλα τη σταθερά c.

Εχουμε:

$$\begin{aligned} T(n) &= 2 \cdot T(n/2) + n \\ &\leq 2c \cdot n/2 \lg (n/2) + n \\ &= c \cdot n (\lg n - 1) + n \\ &= c \cdot n \lg n - (cn - n) \end{aligned}$$

Άρα $T(n) \leq c \cdot n \lg n$ αν $c \cdot n - n \geq 0$. Δηλαδή, αν $c \geq 2$ $(n>0)$.

Επομένως $T(n) \in O(n \lg n)$.

# Μέθοδος της αντικατάστασης

Χρησιμοποιούμε το βήμα της αναδρομής επανελημένα, ώστε να εκφράσουμε το T(n) ως συνάρτηση που περιέχει μόνο τη βασική περίπτωση, δυνάμεις του n και σταθερές τιμές.

*Παράδειγμα*

Έχουμε την αναδρομική εξίσωση

$T(n) = 4 \cdot T(n/2) + n,$ για κάθε $n \geq 2$

$T(1) = 1$

Τότε, αντικαθιστώντας το T(n/2) με την τιμή του παίρνουμε

$$
\begin{aligned}
T(n) &= 4 \cdot T(n/2) + n \\
&= 4(4 \cdot T(n/4) + n/2) + n \\
&= 4^2 \cdot T(n/4) + 2n + n \\
&= 4^3 \cdot T(n/8) + 2^2\, n + 2n + n \\
&= \ \ldots
\end{aligned}
$$

# Μέθοδος της αντικατάστασης - Παράδειγμα

Διακρίνουμε τη γενική μορφή

$$T(n) = 4^i \cdot T\left(\frac{n}{2^i}\right) + 2^{i-1} n + \ldots + 2n + n$$

Υποθέτουμε ότι το n είναι δύναμη του 2 και k=lg n. Τότε

$$\begin{aligned} T(n) &= 4^k \cdot T\left(\frac{n}{2^k}\right) + 2^{k-1} n + \ldots + 2n + n \\ &= 4^k + n \cdot \sum_{i=0}^{k-1} 2^i = (2^k)^2 + n \cdot \frac{2^k - 1}{2 - 1} \\ &= n^2 + n(n-1) \\ &= 2n^2 - n \end{aligned}$$

# Χρήση γενικών λύσεων

**ΘΕΩΡΗΜΑ**

*Έστω a≥1 και b>1 σταθερές, συνάρτηση f(n) και συνάρτηση T(n) που ορίζεται στους μη αρνητικούς ακέραιους μέσω της αναδρομικής εξίσωσης*

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

Τότε η T(n) φράσσεται ως εξής:

1. Αν $f(n) \in O(n^{\log_b a - \varepsilon})$, για κάποια σταθερά ε>0,
   τότε $\mathrm{T}(n) \in \Theta(n^{\log_b a})$

2. Αν $f(n) \in \Theta(n^{\log_b a})$,
   τότε $\mathrm{T}(n) \in \Theta(n^{\log_b a} \cdot \lg n)$

3. Αν $f(n) \in \Omega(n^{\log_b a + \varepsilon})$, για κάποια σταθερά ε>0, και
   a·f(n/b)≤c·f(n), τότε $\mathrm{T}(n) \in \Theta(f(n))$.

# Παραδείγματα

1. $T(n) = 2 \cdot T(n/2) + 1000n$

2. $T(n) = 7 \cdot T(n/2) + 18n^2$

# Πολλαπλασιασμός Πινάκων

- Ο αλγόριθμος που μελετήσαμε ανακαλύφθηκε από τον Strassen το τέλος της δεκαετίας του 60.
- Ακολούθησαν διάφορες προσπάθειες για βελτίωση του αλγόριθμου (εύρεση ω τέτοιου ώστε ο πολλαπλασιασμός $n \times n$ πινάκων να γίνεται σε χρόνο $O(n^{\omega})$ όπου $\omega \in O(\lg 7)$):
  - Οι Hopcroft και Kerr απέδειξαν πως είναι αδύνατος ο πολλαπλασιασμός δύο $2\times2$ πολλαπλασιασμούς με 6 πολλαπλασιασμούς.
  - Επίσης αδύνατος είναι ο πολλαπλασιασμός $3\times3$ πινάκων με 21 πολλαπλασιασμούς.
  - Ο Pan έδειξε πως δύο $70\times70$ πίνακες μπορούν να πολλαπλασιαστούν με 143640 πολλαπλασιασμούς ακεραίων.
  - Ακολούθησαν αλγόριθμοι πολυπλοκότητας $O(n^{2.521813})$,..., $O(n^{2.52180})$, και σήμερα ο πιο "γρήγορος" γνωστός αλγόριθμος είναι αυτός των Coppersmith και Winograd (1986) πολυπλοκότητας $O(n^{2.376})$.
- Στην πράξη ο αλγόριθμος του Strassen είναι ο πιο αποδοτικός.