ΠΛΗ30

ΕΝΟΤΗΤΑ 2: ΣΧΕΔΙΑΣΗ ΑΛΓΟΡΙΘΜΩΝ

Μάθημα 2.1:
Διαίρει και Βασίλευε

Δημήτρης Ψούνης

www.psounis.gr

ΠΕΡΙΕΧΟΜΕΝΑ

Α. Σκοπός του Μαθήματος

Οι στόχοι του μαθήματος είναι:

Επίπεδο Α
- (-)

Επίπεδο Β
- Η τεχνική σχεδίασης αλγόριθμων Διαίρει και Βασίλευε

Επίπεδο Γ
- Ο αλγόριθμος Γρήγορης Ταξινόμησης (QuickSort)
- Ο αλγόριθμος Επιλογής του k-μικρότερου αριθμού σε έναν πίνακα
- Ο αλγόριθμος Πολλαπλασιασμού Πινάκων Strassen

Β. Θεωρία
Τεχνικές Σχεδίασης Αλγορίθμων

- Στην 2η ενότητα του μαθήματος ασχολούμαστε με τεχνικές που έχουν αναπτυχθεί, ως γενικές μεθοδολογίες για την κατασκευή ενός αλγορίθμου:
  - **Η τεχνική Διαίρει και Βασίλευε (Μάθημα 2.1)**
  - Η τεχνική του Δυναμικού Προγραμματισμού (Μάθημα 2.2)
  - Η κατασκευή των Άπληστων Αλγόριθμων (Μάθημα 2.3)
- Υπάρχουν ακόμη δεκάδες τεχνικές κατασκευής αλγορίθμων που είναι εκτός ύλης.

Β. Θεωρία
1. Διαίρει και Βασίλευε

ΔΙΑΙΡΕΙ ΚΑΙ ΒΑΣΙΛΕΥΕ
Ένας αλγόριθμος διαίρει και βασίλευε συνίσταται από τα εξής βήματα:

1. ΒΗΜΑ ΔΙΑΙΡΕΣΗΣ: Διάσπαση του αρχικού προβλήματος σε μικρότερα επιμέρους υποπροβλήματα.
2. ΒΗΜΑ ΕΠΙΛΥΣΗΣ ΕΠΙΜΕΡΟΥΣ ΣΤΙΓΜΙΟΤΥΠΩΝ: Επίλυση των επιμέρους υποπροβλημάτων (με αναδρομικές κλήσεις του ίδιου αλγόριθμου)
3. ΒΗΜΑ ΣΥΝΘΕΣΗΣ ΛΥΣΕΩΝ: Υπολογισμός της λύσης του αρχικού προβλήματος, από τις επιμέρους λύσεις των υποπροβλημάτων.

Β. Θεωρία

1. Διαίρει και Βασίλευε

1. Ο αλγόριθμος ταξινόμησης MergeSort

- Ο αλγόριθμος ταξινόμησης MergeSort (Ο αλγόριθμος Ταξινόμησης με Συγχώνευση) είναι εφαρμογή του Διαίρει και Βασίλευε:

1. ΒΗΜΑ ΔΙΑΙΡΕΣΗΣ: Ο πίνακας $A=[a_1,\ldots,a_n]$ χωρίζεται σε δύο μέρη: τον πίνακα $A_1=[a_1,\ldots,a_{n/2}]$ και τον πίνακα $A_2=[a_{n/2+1},\ldots,a_n]$
2. ΒΗΜΑ ΕΠΙΛΥΣΗΣ ΕΠΙΜΕΡΟΥΣ ΣΤΙΓΜΙΟΤΥΠΩΝ: Η επίλυση των δύο υποπροβλημάτων που προκύπτουν γίνονται με αναδρομικές κλήσεις του MergeSort.
3. ΒΗΜΑ ΣΥΝΘΕΣΗΣ ΛΥΣΕΩΝ: Η συγχώνευση των δύο πινάκων γίνεται με την διαδικασία Merge (βλέπε Μάθημα 1.4) σε γραμμικό χρόνο

- Η πολυπλοκότητα του αλγορίθμου είναι: $T(n)=2T\left(\frac{n}{2}\right)+\Theta(n)=\ldots=\Theta(n\log n)$

Β. Θεωρία

1. Διαίρει και Βασίλευε

2. Ο αλγόριθμος ταξινόμησης QuickSort

- Ο αλγόριθμος ταξινόμησης QuickSort (Ο αλγόριθμος Γρήγορης Ταξινόμησης) είναι επίσης εφαρμογή του Διαίρει και Βασίλευε:

1. ΒΗΜΑ ΔΙΑΙΡΕΣΗΣ: Ο πίνακας $A=[a_1,...,a_n]$ χωρίζεται σε δύο μέρη:
   τον πίνακα A_1 που περιεχει όλα τα στοιχεία που είναι μικρότερα από το $α_1$ και
   τον A_2 που περιέχει όλα τα στοιχεία που είναι μεγαλύτερα από το $α_1$

2. ΒΗΜΑ ΕΠΙΛΥΣΗΣ ΕΠΙΜΕΡΟΥΣ ΣΤΙΓΜΙΟΤΥΠΩΝ: Η επίλυση των δύο υποπροβλημάτων (ταξινόμηση των υποπινάκων A_1 και A_2) που προκύπτουν γίνονται με αναδρομικές κλήσεις του QuickSort.

3. ΒΗΜΑ ΣΥΝΘΕΣΗΣ ΛΥΣΕΩΝ: Ο πίνακας $A = [A_1]α_1[A_2]$

Β. Θεωρία

1. Διαίρει και Βασίλευε

2. Ο αλγόριθμος ταξινόμησης QuickSort (1.Ψευδοκώδικας)

- Παρακάτω φαίνεται μια υλοποίηση της διαδικασίας QuickSort:

```
procedure QuickSort(A,start,finish)
   if start<finish then
      pos=Partition(A,start,finish)
      QuickSort(A,start,pos-1)
      QuickSort(A,pos+1,finish)
   end if
end procedure
```

```
procedure Partition(A,start,finish)
    odigo=A[start]
    i=start;   j=finish
    for (k=start+1 to finish)
       if (A[k]>odigo)
          B[j]=A[k]; j=j-1
       else
          B[i]=A[k]; i=i+1
    end for
    B[i]=odigo; A=B
    return pos;
end procedure
```

Β. Θεωρία

1. Διαίρει και Βασίλευε

2. Ο αλγόριθμος ταξινόμησης QuickSort (2.Παράδειγμα Εκτέλεσης)

- Ας τρέξουμε ένα παράδειγμα εκτέλεσης της Partition:
- B.1

1	2	3	4	5	6	7	8
9	7 k	4	11	18	20	6	1

1	2	3	4	5	6	7	8
7	i						j

- B.2

1	2	3	4	5	6	7	8
9	7	4 k	11	18	20	6	1

1	2	3	4	5	6	7	8
7	4	i					j

- B.3

1	2	3	4	5	6	7	8
9	7	4	11 k	18	20	6	1

1	2	3	4	5	6	7	8
7	4	i				j	11

- B.4

1	2	3	4	5	6	7	8
9	7	4	11	18 k	20	6	1

1	2	3	4	5	6	7	8
7	4	i			j	18	11

- B.5

1	2	3	4	5	6	7	8
9	7	4	11	18	20 k	6	1

1	2	3	4	5	6	7	8
7	4	i		j	20	18	11

- B.6

1	2	3	4	5	6	7	8
9	7	4	11	18	20	6 k	1

1	2	3	4	5	6	7	8
7	4	6	i	j	20	18	11

- B.7

1	2	3	4	5	6	7	8
9	7	4	11	18	20	6	1 k

1	2	3	4	5	6	7	8
7	4	6	1	i j	20	18	11

1	2	3	4	5	6	7	8
7	4	6	1	9	20	18	11

Ο αλγόριθμος ταξινόμησης QuickSort (2.Παράδειγμα Εκτέλεσης)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	7	4	11	9	20	6	1	22	19	14	5	2	3	10	13
7	4	11	9	6	1	14	5	2	3	10	13	18	19	22	20

1	2	3	4	5	6	7	8	9	10	11	12
7	4	11	9	6	1	14	5	2	3	10	13
4	6	1	5	2	3	7	13	10	14	9	11

14	15	16
19	22	20
19	20	22

1	2	3	4	5	6
4	6	1	5	2	3
1	2	3	4	5	6

8	9	10	11	12
13	10	14	9	11
10	9	11	13	14

15	16
20	22
20	22

1	2	3
1	2	3
1	3	2

5	6
5	6
5	6

8	9	10
10	9	11
9	10	11

12
14
14

16
22
22

2	3
3	2
2	3

6
6
6

8
9
9

10
11
11

2
2
2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	9	10	11	13	14	18	19	20	22

Β. Θεωρία

1. Διαίρει και Βασίλευε

2. Ο αλγόριθμος ταξινόμησης QuickSort (3. Ανάλυση)

Η διαδικασία του Partition έχει πολυπλοκότητα Θ(n)

- Ανάλυση Καλύτερης Περίπτωσης:
  - Όταν το σπάσιμο γίνεται στην μέση του πίνακα. Τότε προκύπτει η αναδρομή:

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n) \overset{\theta.\kappa\upsilon\rho\iota\alpha\rho\chi\acute{\iota}\alpha\varsigma}{=} \dots = \Theta(n \log n)$$

- Ανάλυση Χειρότερης Περίπτωσης:
  - Όταν το σπάσιμο γίνεται στην μέση του πίνακα. Τότε προκύπτει η αναδρομή:

$$T(n) = T(n-1) + \Theta(n) \overset{\mu\varepsilon\theta.\varepsilon\pi\alpha\nu\alpha\lambda\eta\psi\eta\varsigma}{=} \dots = \Theta(n^2)$$

- Στην μέση περίπτωση?

Β. Θεωρία

1. Διαίρει και Βασίλευε

2. Ο αλγόριθμος ταξινόμησης QuickSort (3. Ανάλυση)

- Ανάλυση Μέσης Περίπτωσης:
  - Αν θεωρήσουμε ότι είναι ισοπίθανα τα δυνατά σπασίματα του πίνακα σε δύο κομμάτια τότε κατά μέσο όρο οι πράξεις του αλγορίθμου θα δίνονται από την αναδρομική σχέση:

$$T(n)=\frac{[T(0)+T(n-1)+\Theta(n)]+[T(1)+\mathrm{T}(n-2)+\Theta(n)]+\ldots+[T(n-1)+T(0)+\Theta(n)]}{n}$$

  - Ή πιο απλά:

$$T(n)=\frac{2[T(0)+T(1)+\ldots+T(n)]+n\Theta(n)}{n}$$

  - Και τελικά

$$T(n)=\frac{2[T(0)+T(1)+\ldots+T(n)]}{n}+\Theta(n)$$

  - Για την οποία αποδεικνύεται ότι ισχύει T(n)=Θ(nlogn)

Β. Θεωρία

1. Διαίρει και Βασίλευε

3. Ο αλγόριθμος επιλογης QuickSelect

- ΠΡΟΒΛΗΜΑ: Δίνεται ένας αταξινόμητος πίνακας με n στοιχεία. Ζητείται να βρεθεί το k-μικρότερο στοιχείο
- Παράδειγμα στιγμιοτύπου:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	7	4	11	9	20	6	1	22	19	14	5	2	3	10	13

  - Το 5-μικρότερο στοιχείο είναι το 5
  - Το 10-μικρότερο στοιχείο είναι το 11

- Αλγόριθμοι:
  - Προφανής Αλγόριθμος: Ταξινόμησε τον πίνακα (με MergeSort) και επέλεξε το k-ο μικρότερο στοιχείο. Πολυπλοκότητα: Θ(nlogn)
  - QuickSelect: Εφαρμογή Διαίρει και Βασίλευε: Πολ/τα: O(n)

Β. Θεωρία

1. Διαίρει και Βασίλευε

3. Ο αλγόριθμος επιλογης QuickSelect (Ιδέα)

- Εκμεταλλευόμαστε το σπάσιμο του πίνακα που κάνει η διαδικασία partition. Θα χρειάζεται κάθε φορά να ψάχνουμε έναν από τους δύο υποπίνακες που προκύπτουν με κατάλληλη τροποποίηση του k.

- Έτσι αφού η διαδικασία Partition σπάει τον πίνακα ως εξής $[A_1]\ a_{pos}\ [A_2]$
  - Αν k=pos τότε βρήκαμε το στοιχείο είναι το apos.
  - Αν k<pos τότε ψαχνουμε στον πίνακα A_1 για το k-μικρότερο στοιχείο
  - Αν k>pos τότε ψάχνουμε στον πίνακα A_2 για το $(k-n_1-1)$-μικρότερο στοιχείο (όπου n_1 το πλήθος των στοιχείων του A_1)

Β. Θεωρία

1. Διαίρει και Βασίλευε

3. Ο αλγόριθμος επιλογης QuickSelect (1.Ψευδοκώδικας)

- Η υλοποιηση σε ψευδογλώσσα της παραπάνω διαδικασίας είναι η ακόλουθη:

```
procedure QuickSelect(A,start,finish,k)
   if start>finish then
      return 0
   else
      pos=Partition(A,start,finish)
      if k=pos then
          return A[pos]
      else if k<pos then
          return QuickSelect(A,start,pos-1,k)
      else if k>pos then
          return QuickSelect(A,pos+1,finish,k-pos)
      end if
   end if
end procedure
```

3. Ο αλγόριθμος επιλογης QuickSelect(2.Παράδειγμα Εκτέλεσης για k=12)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	7	4	11	9	20	6	1	22	19	14	5	2	3	10	13
7	4	11	9	6	1	14	5	2	3	10	13	18	19	22	20

1	2	3	4	5	6	7	8	9	10	11	12
7	4	11	9	6	1	14	5	2	3	10	13
4	6	1	5	2	3	7	13	10	14	9	11

8	9	10	11	12
13	10	14	9	11
10	9	11	13	14

12
14
14

Β. Θεωρία

1. Διαίρει και Βασίλευε

3. Ο αλγόριθμος επιλογης QuickSelect (3. Ανάλυση)

Η διαδικασία του Partition έχει πολυπλοκότητα Θ(n)

- Ανάλυση Καλύτερης Περίπτωσης:
  - Όταν το σπάσιμο γίνεται στην μέση του πίνακα. Τότε προκύπτει η αναδρομή:

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(n) = \overset{\theta.\kappa\upsilon\rho\iota\alpha\rho\chi\acute{\iota}\alpha\varsigma}{\ldots} = \Theta(n)$$

- Ανάλυση Χειρότερης Περίπτωσης:
  - Όταν το σπάσιμο γίνεται στην μέση του πίνακα. Τότε προκύπτει η αναδρομή:

$$T(n) = T(n-1) + \Theta(n) = \overset{\mu\varepsilon\theta.\varepsilon\pi\alpha\nu\alpha\lambda\eta\psi\eta\varsigma}{\ldots} = \Theta(n^2)$$

- Χρειαζόμαστε ένα πιο έξυπνο σπάσιμο σπάσιμο του πίνακα ώστε να επιτυγχάνεται ένα σπάσιμο του πίνακα περίπου στην μέση.

Β. Θεωρία

1. Διαίρει και Βασίλευε

3. Ο αλγόριθμος επιλογης QuickSelect (4. Παραλλαγή με τη διαδικασία Πεντάδων)

Προτείνεται η διαδικασία των πεντάδων η οποία:

- Χωρίζει τα στοιχεία σε 5-αδες
- Από κάθε 5-άδα επιλέγουμε το μεσαίο στοιχειο.
- Επαναλαμβάνουμε αναδρομικά για τα μεσαία στοιχεία.
- Επιλέγουμε δηλαδή το μεσαίο των μεσαίων (των μεσαίων…) από κάθε πεντάδα.
- Π.χ.:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
18	7	4	11	9	20	6	1	22	19	14	5	2	3	10	13

- 5-άδες

1	2	3	4	5
18	7	4	11	9

6	7	8	9	10
20	6	1	22	19

11	12	13	14	15
14	5	2	3	10

16
13

- Νεος Πίνακας

1	2	3	4
9	19	5	13

αναδρομική εκτέλεση

- Επιστρέφεται το 9 (επιλογή οδηγού στοιχείου)

Β. Θεωρία

1. Διαίρει και Βασίλευε

3. Ο αλγόριθμος επιλογης QuickSelect (4. Παραλλαγή με τη διαδικασία Πεντάδων)

Η επιλογή του μεσαίου στοιχείου από 5 στοιχεία θέλει σταθερό χρόνο:

- Άρα αν ο πίνακας έχει n στοιχεία και θέλουμε συνολικά χρόνο T(n)
  - Επιλέγουμε n/5 φορές το μεσαίο (με χρόνο Θ(1)). Το βήμα αυτό θέλει συνολικό χρόνο $\frac{n}{5}\Theta(1)=\Theta(n)$
  - Έπειτα κάνουμε μια αναδρομική κλήση για n/5 δεδομένα.
- Συνεπώς η πολυπλοκότητα της επιλογής του στοιχείου με την διαδικασία των 5-άδων προκύπτει από την επίλυση της αναδρομικής σχέσης:

$$T(n)=T\left(\frac{n}{5}\right)+\Theta(n)=\overset{\theta.\kappa\upsilon\rho\iota\alpha\rho\chi\acute{\iota}\alpha\varsigma}{\ldots}=\Theta(n)$$

- Αποδεικνύεται ότι είναι πολύ καλό σπάσιμο, διότι σε κάθε βήμα θα απορρίπτονται τουλάχιστον 3n/10 στοιχεία (Η απόδειξη παραλείπεται). Δηλαδή με άλλα λόγια η αναδρομή της QuickSelect θα γίνει για το πολύ 7n/10 στοιχεία.

Β. Θεωρία

1. Διαίρει και Βασίλευε

3. Ο αλγόριθμος επιλογης QuickSelect (4. Παραλλαγή με τη διαδικασία Πεντάδων)

- Ενσωματώνουμε την διαδικασία επιλογής οδηγού στοιχείου στον προηγούμενο κώδικα:

```
procedure QuickSelect(A,start,finish,k)
   if start=finish then
      return A[start]
   else
      Επιλογή στοιχείου m με την διαδικασία των 5-άδων.
      swap(A[m],A[start])
      pos=Partition(A,start,finish)
      if k=pos then
          return A[pos]
      else if k<pos then
          return QuickSelect(A,start,pos-1,k)
      else if k>pos then
          return QuickSelect(A,pos+1,finish,k-pos)
      end if
   end if
end procedure
```

Β. Θεωρία

1. Διαίρει και Βασίλευε

3. Ο αλγόριθμος επιλογης QuickSelect (4. Παραλλαγή με τη διαδικασία Πεντάδων)

Η αναδρομική σχέση που περιγράφει την τελική πολυπλοκότητα του αλγορίθμου είναι:

- Για να λύσουμε ένα πρόβλημα μεγέθους n:
  - Οι πεντάδες θέλουν χρόνο Θ(n)
  - Η διαδικασία Partition θέλει χρόνο Θ(n)
  - Η αναδρομική κλήση θα γίνει για ένα πρόβλημα μεγέθους το πολύ 7n/10, συνεπώς ο χρόνος που απαιτείται είναι T(7n/10)
- Συνεπώς:

$$T(n) = T\left(\frac{7n}{10}\right) + \Theta(n) \overset{\theta.\kappa\upsilon\rho\iota\alpha\rho\chi\acute{\iota}\alpha\varsigma}{=\ldots=} \Theta(n)$$

- Αυτή είναι και η χειρότερη περίπτωση, άρα τελικά έχουμε T(n)=O(n)

Β. Θεωρία

1. Διαίρει και Βασίλευε

4. Ο αλγόριθμος πολλαπλασιασμού πινάκων Strassen

- ΠΡΟΒΛΗΜΑ: Να υπολογιστεί το γινόμενο δύο nxn πινάκων A και B.

$$C = A \times B =$$

$$C = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

- Αλγόριθμοι:
  - Εφαρμογή του γνωστού αλγορίθμου από ΠΛΗ12. Η πολυπλοκότητα του είναι $\Theta(n^2)$.
  - Αλγόριθμος του Strassen: Εφαρμογή του Διαίρει και Βασίλευε: Πολυπλοκότητα: $\Theta(n^{2,81})$
  - Ακόμη καλύτεροι αλγόριθμοι Διαίρει και Βασίλευε. Η καλύτερη πολυπλοκότητα μέχρι στιγμής: $\Theta(n^{2,38})$

Β. Θεωρία

1. Διαίρει και Βασίλευε

4. Ο αλγόριθμος πολλαπλασιασμού πινάκων Strassen

- Ο γνωστός αλγόριθμος πολλαπλασιασμού διδιάστατων πινάκων μπορεί να υλοποιηθεί σε ψευδογλώσσα ως εξής:

```
procedure MultMatrix(A,B)

    for (i=1 to n)
       for (j=1 to n)
          C[i][j]=0;
          for (k=1 to n)
             C[i][j]=C[i][j]+A[i][k]*B[k][j];
          end for
       end for
    end for

    return C
end procedure
```

- Η πολυπλοκότητα είναι:

$$T(n) = n \cdot n \cdot \left(\Theta(1) + n \cdot \Theta(1)\right) = \ldots = \Theta(n^3)$$

Β. Θεωρία

1. Διαίρει και Βασίλευε

4. Ο αλγόριθμος πολλαπλασιασμού πινάκων Strassen

- Strassen ‘70: Εφαρμογή του διαίρει και βασίλευε (θεωρώ ότι n άρτιος)

- Οι πίνακες A και B σπάνε στους πίνακες:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \qquad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

Όπου κάθε ένας από τους υποπίνακες έχει μέγεθος n/2 x n/2

- Ο Strassen προσπάθησε να υπολογίσει το γινόμενο C=AxB κατασκευάζοντας τον πίνακα:

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

Β. Θεωρία

1. Διαίρει και Βασίλευε

4. Ο αλγόριθμος πολλαπλασιασμού πινάκων Strassen

- 1η προσέγγιση: Ισχύει ότι:

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

- Αντί να κάνουμε τον nxn πολ/μο των πινάκων.
    - ΔΙΑΣΠΑΣΗ: Γίνονται 8 πολ/μοι n/2 x n/2 πινάκων
    - ΕΠΙΛΥΣΗ: Αναδρομικά με τον ίδιο τρόπο.
    - ΣΥΝΘΕΣΗ ΛΥΣΕΩΝ: 4 προσθέσεις nx2 x nx2 πινάκων
- Άρα για να λύσω ένα πρόβλημα μεγέθους n, λύνω 8 υποπροβλήματα μεγέθους n/2 και συνδυάζω τις λύσεις σε χρόνο Θ(n²)
- Η πολυπλοκότητα είναι:

$$T(n) = 8T\left(\frac{n}{2}\right) + \Theta(n^2) = \overset{Master Theorem}{\dots} = \Theta(n^3)$$

Καμία βελτίωση σε σχέση με τον προφανή αλγόριθμο

B. Θεωρία

1. Διαίρει και Βασίλευε

4. Ο αλγόριθμος πολλαπλασιασμού πινάκων Strassen

- 1η προσέγγιση: Ισχύει ότι:

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} = A_{11}B_{12} + A_{12}B_{22}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

- Αντί να κάνουμε τον nxn πολ/μο των πινάκων.
  - ΔΙΑΣΠΑΣΗ: Γίνονται 8 πολ/μοι n/2 x n/2 πινάκων
  - ΕΠΙΛΥΣΗ: Αναδρομικά με τον ίδιο τρόπο.
  - ΣΥΝΘΕΣΗ ΛΥΣΕΩΝ: 4 προσθέσεις nx2 x nx2 πινάκων
- Άρα για να λύσω ένα πρόβλημα μεγέθους n, λύνω 8 υποπροβλήματα μεγέθους n/2 και συνδυάζω τις λύσεις σε χρόνο Θ(n^2)
- Η πολυπλοκότητα είναι:

$$T(n) = 8T\left(\frac{n}{2}\right) + \Theta(n^2) = \overset{MasterTheorem}{\cdots} = \Theta(n^3)$$

Καμία βελτίωση σε σχέση με τον προφανή αλγόριθμο

Β. Θεωρία

1. Διαίρει και Βασίλευε

4. Ο αλγόριθμος πολλαπλασιασμού πινάκων Strassen

- 2η προσέγγιση (Strassen): Βρήκε ότι μπορεί να παρει το ίδιο αποτέλεσμα με τις εξής πράξεις:

$$M_1 = (A_{21} + A_{22} - A_{11})(B_{22} - B_{12} + B_{11})$$
$$M_2 = A_{11}B_{11}$$
$$M_3 = A_{12}B_{21}$$
$$M_4 = (A_{11} - A_{21})(B_{22} - B_{12})$$
$$M_5 = (A_{21} + A_{22})(B_{12} - B_{11})$$
$$M_6 = (A_{12} - A_{21} + A_{11} - A_{22})B_{22}$$
$$M_7 = A_{22}(B_{11} + B_{22} - B_{12} - B_{21})$$

$$C_{11} = M_2 + M_3$$
$$C_{12} = M_1 + M_2 + M_5 + M_6$$
$$C_{21} = M_1 + M_2 + M_4 - M_7$$
$$C_{22} = M_1 + M_2 + M_4 + M_5$$

- Αντί να κάνουμε τον nxn πολ/μο των πινάκων.
    - ΔΙΑΣΠΑΣΗ: Γίνονται 7 πολ/μοι n/2 x n/2 πινάκων
    - ΕΠΙΛΥΣΗ: Αναδρομικά με τον ίδιο τρόπο.
    - ΣΥΝΘΕΣΗ ΛΥΣΕΩΝ: 24 προσθαφαιρέσεις nx2 x nx2 πινάκων
- Άρα για να λύσω ένα πρόβλημα μεγέθους n, λύνω 7 υποπροβλήματα μεγέθους n/2 και συνδυάζω τις λύσεις σε χρόνο $\Theta(n^2)$
- Η πολυπλοκότητα είναι: $T(n) = 7T\left(\frac{n}{2}\right) + \Theta(n^2) = \overset{MasterTheorem}{\ldots} = \Theta(n^{2,81})$
- Υπάρχουν και ακόμη καλύτερες διασπάσεις που φτάνουν την πολυπλοκότητα μέχρι και $\Theta(n^{2,37})$. Ωστόσο ισχύει ότι κάθε αλγόριθμος πολ/μου πινάκων θα έχει πολυπλοκότητα $\Omega(n^2)$

Γ. Ασκήσεις
Εφαρμογή 1

- Ας υποθέσουμε ότι έχουμε να επιλέξουμε ανάμεσα στους ακόλουθους τρεις αλγόριθμους:
  I. Ο αλγόριθμος Α λύνει προβλήματα μεγέθους n με το να επιλύει αναδρομικά οκτώ υποπροβλήματα του μισού μεγέθους το καθένα, και συνδυάζοντας τις λύσεις τους σε γραμμικό χρόνο ως προς n.
  II. Ο αλγόριθμος Β λύνει προβλήματα μεγέθους n με το να επιλύει αναδρομικά ένα υποπρόβλημα μεγέθους n-1 και, στην συνέχεια, συνάγει την τελική λύση σε χρόνο O(n).
  III. Ο αλγόριθμος Γ λύνει προβλήματα μεγέθους n με το να επιλύει αναδρομικά εννιά υποπροβλήματα μεγέθους n/3 και συνδυάζοντας τις λύσεις τους σε $\Omega(n^2)$ χρόνο.
- Ποιοι είναι οι ασυμπτωτικοί χρόνοι εκτέλεσης για καθένα από τους τρεις αλγορίθμους και ποιον από αυτούς θα διαλέγατε με βάση την ασυμπτωτική του πολυπλοκότητα?

Γ. Ασκήσεις
Εφαρμογή 2

- Ας υποθέσουμε ότι έχουμε να επιλέξουμε ανάμεσα στους ακόλουθους τρεις αλγόριθμους:
  I. Ο αλγόριθμος Α λύνει προβλήματα μεγέθους n με το να επιλύει αναδρομικά ένα υποπρόβλημα, μεγέθους 1/4 του αρχικού προβλήματος, ενώ ο χρόνος που απαιτείται για να βρεθεί το πρόβλημα αυτό (μεγέθους n/4) από το αρχικό (μεγέθους n) είναι γραμμικός ως προς n.
  II. Ο αλγόριθμος Β λύνει προβλήματα μεγέθους n με το να επιλύει αναδρομικά ένα υποπρόβλημα μεγέθους n/2 και, στην συνέχεια, συνάγει την τελική λύση σε $\Omega(n^3)$.
  III. Ο αλγόριθμος Γ λύνει προβλήματα μεγέθους n με το να επιλύει αναδρομικά τέσσερα υποπροβλήματα μεγέθους n/2 και συνδυάζοντας τις λύσεις τους σε $O(n^2)$ χρόνο.
- Ποιοι είναι οι ασυμπτωτικοί χρόνοι εκτέλεσης για καθένα από τους τρεις αλγορίθμους και ποιον από αυτούς θα διαλέγατε με βάση την ασυμπτωτική του πολυπλοκότητα?