ΣΥΓΧΩΝΕΥΣΗ (merging)

Έστω

$$A=\{a_1, a_2, \dots, a_r\} \qquad B=\{b_1, b_2, \dots, b_s\}$$

ταξινομημένες ακολουθίες αριθμών σε μη φθίνουσα σειρά. Να συνδυαστεί η ακολουθία

$$C=\{c_1, c_2, \dots, c_{r+s}\}$$

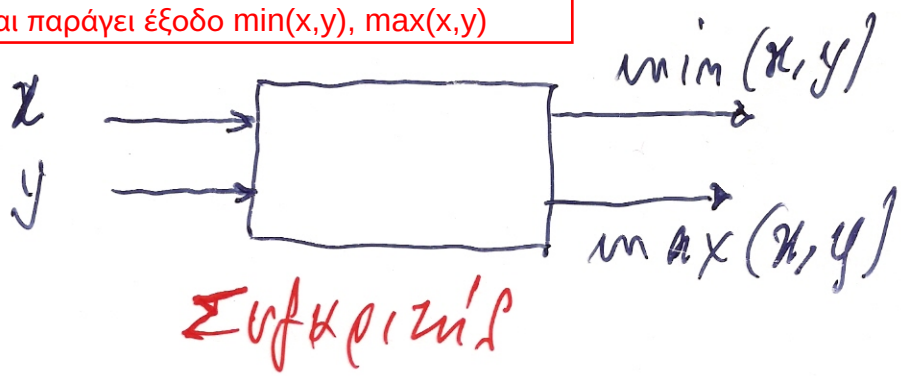
ταξινομημένη σε μη φθίνουσα σειρά. Αν $r=s=n$ (χειρότερη περίπτωση), τότε ο ακολουθιακός αλγόριθμος έχει πολυπλοκότητα $O(n)$, η οποία είναι βέλτιστη ($\Omega(n)$). Ο λόγος μας είναι να μελετηθεί το πρόβλημα της συγχώνευσης σε μια κλίμακα παραλληλων υπολογιστικών μοντέλων. Με βάση το κάτω όριο θα επιδιωχθεί όση λιγότερη χρόνος $\underline{\Omega(n/N)}$ χρόνος για οποιοδήποτε παράλληλο αλγόριθμο συγχώνευσης που χρησιμοποιεί N επεξεργαστές.

O(n) βέλτιστο γιατί κάθε στοιχείο των A,B πρέπει να εξεταστεί τουλάχιστον μια φορά, δηλ. έχουμε Ω(n) βήματα αναγκαία για τη συγχώνευση.

ΕΝΑ ΔΙΚΤΥΟ ΓΙΑ ΣΥΓΧΩΝΕΥΣΗ

(r,s) δίκτυο συγχώνευσης: ειδικού σκοπού παράλληλη αρχιτεκτονική. ίδια,
- όλοι οι επεξεργαστές είναι ταξινομημένοι και επικοινωνούν με ένα ειδικού σκοπού δίκτυο (συγκριτές (comparators))

Λειτουργία συγκριτή: Σύγκριση των 2 τιμών της εισόδου δηλ. των x,y και παράγει έξοδο min(x,y), max(x,y)



Συγκριτής

Χρησιμοποιώντας τους συγκριτές μπορούμε να φτιάξουμε ένα δίκτυο που να έχει σαν είσοδο 2 ταξινομημένες ακολουθίες $A=\{a_1,a_2,\ldots,a_r\}$ και $B=\{b_1,b_2,\ldots,b_s\}$ και να παράγει σαν έξοδο μια απλή ταξινομημένη ακολουθία $C=\{c_1,c_2,\ldots,c_{r+s}\}$

Υποθέσεις

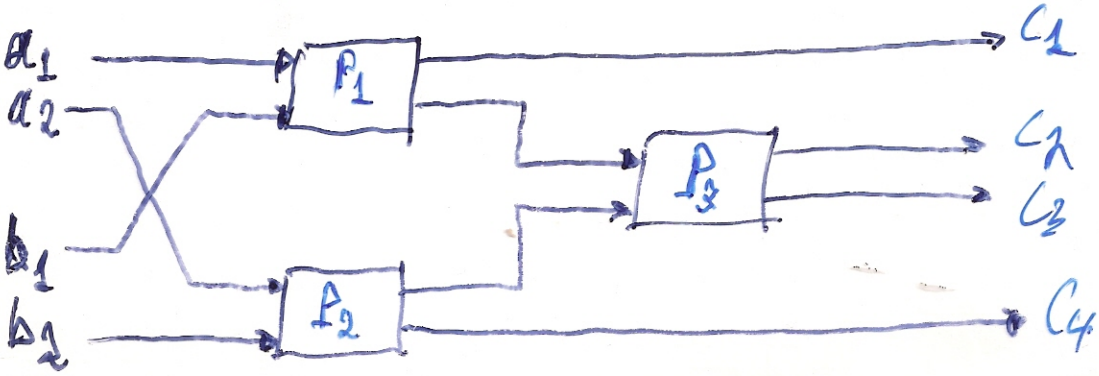
1. $r=s=n\geq 1$ πλήθος στοιχείων των ακολουθιών A,B
2. $n=2^k$

Με τη χρήση των συγκριτών θα κατασκευάσει ένα δίκτυο.

$n=1$ Αρκεί ένας συγκριτής

$n=2$ $A=\{a_1,a_2\}$ $B=\{b_1,b_2\}$

γιατί παράγει σαν έξοδο τις δύο εισόδους τους ταξινομημένες ανάλογα



Ο Επεξ. P1 συγκρίνει minA με minB. Η πάνω έξοδος του P1 είναι το minC=c1. Όμοια η κάτω έξοδος του P2 είναι maxC=c4. Μια απλή επιπλέον σύγκριση επιτρέπει στον P3 να παράγει δύο ενδιάμεσα στοιχεία της C

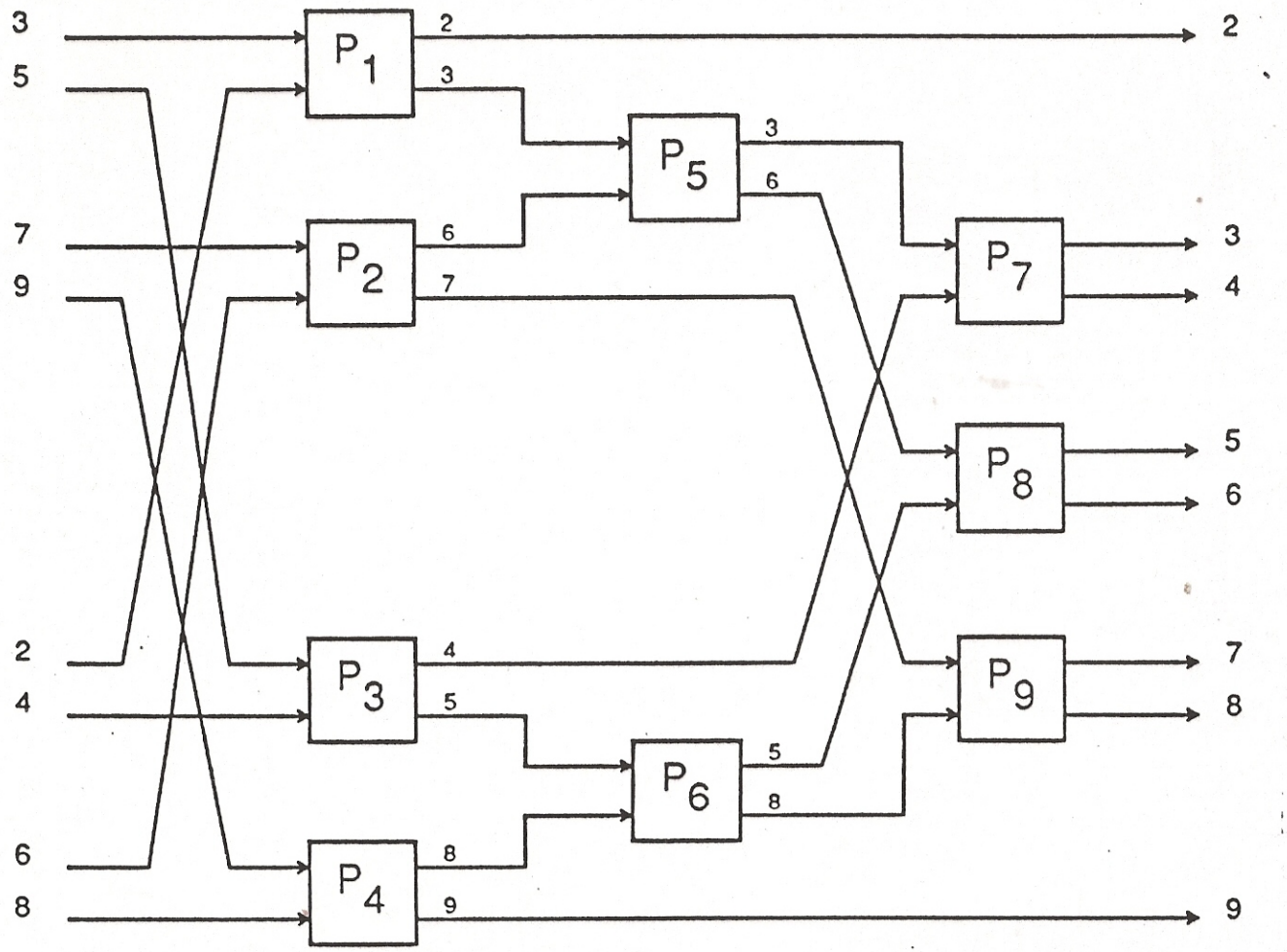
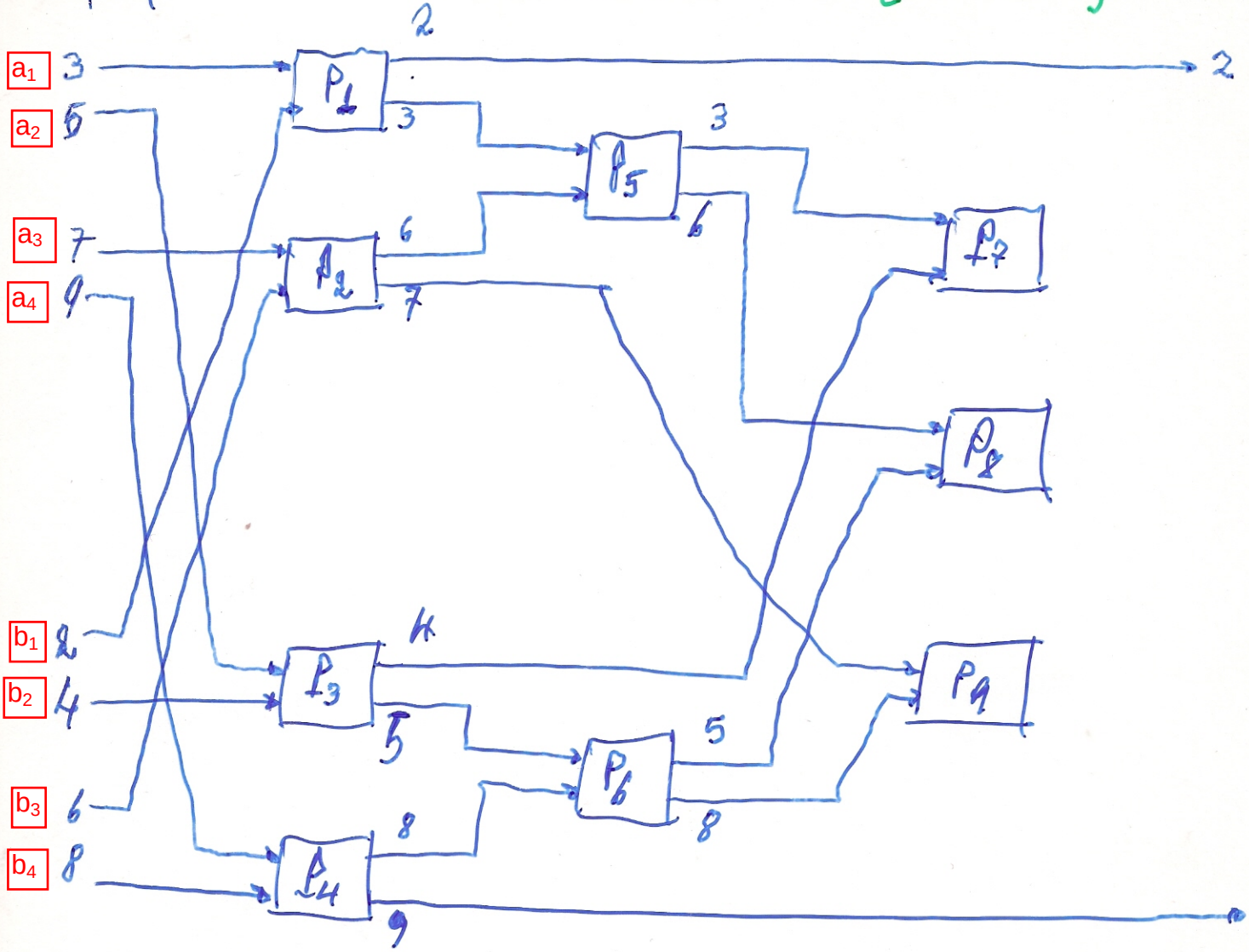


Figure 3.3 Merging two sequences of four elements each.

n = 4

A = {3, 5, 7, 9} B = {2, 4, 6, 8}



Γενικά ένα (n, n) δίκτυο ταξινόμησης διαχωρίζεται σε δύο (n/2, n/2) δίκτυα.

Τα στοιχεία των πρώτων θέσεων των A και B δηλ. $\{a_1, a_3, a_5, \ldots, a_{n-1}\}$ και $\{b_1, b_3, b_5, \ldots, b_{n-1}\}$ διαχωρίζονται στο ένα (n/2, n/2) δίκτυο ταξινόμησης, διαμορφώνεται των

$$\{d_1, d_2, d_3, \ldots, d_n\}$$

Ταυτόχρονα για τα στοιχεία στις άλλες

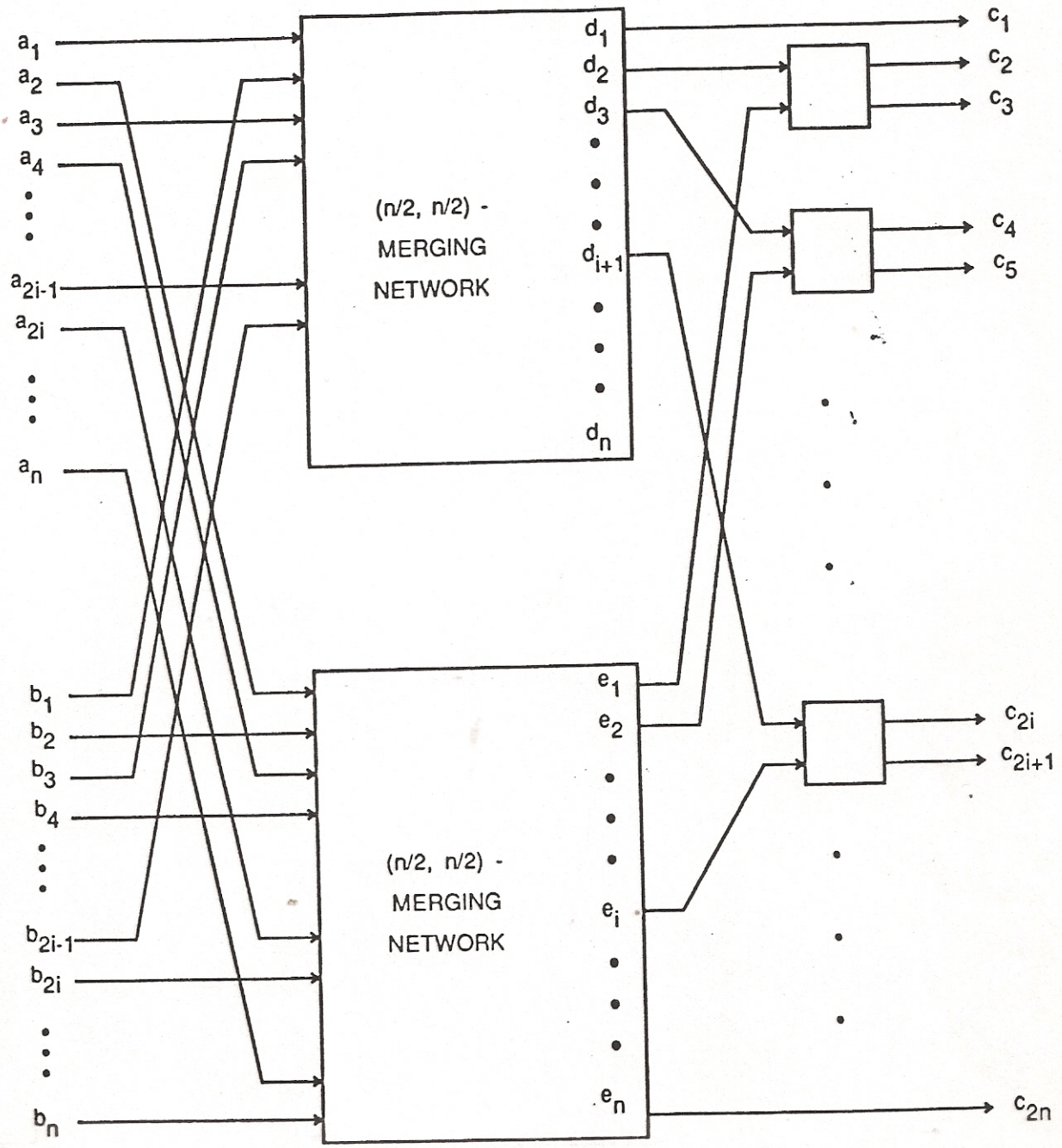


Figure 3.4 Odd–even merging.

Θέλεις συγχωνεύσεις τις

$\{e_1, e_2, \dots, e_n\}$

με τη χρήση του λήμματος (n/2, n/2) δικτύου.

Η τελική ακολουθία $\{c_1, c_2, \dots, c_n\}$ προκύπτει από odd-even merging

$c_1 = d_1$, $c_{2n} = e_n$, $c_{2i} = \min(d_{i+1}, e_i)$ και

$c_{2i+1} = \max(d_{i+1}, e_i)$, $i = 1, 2, \dots, n-1$.

Τρόπος λειτουργίας Odd-Even Merging: Να σημειωθεί αρχικά ότι d_1=min(a1,b1) και e_n=max(an,bn)-> υπολογισμός των c1,cn. Παρατηρούμε ότι:

Στην $\{d_1, d_2, \dots, d_n\}$

υπάρχουν i στοιχεία $\leq d_{i+1}$ → όπου το καθένα από αυτό είναι ένα μονό στοιχείο είτε της Α είτε της Β. ¨Αρα,

$2i$ στοιχεία των Α και Β $\leq d_{i+1}$

ή

$c_{2i} \leq d_{i+1}$

όμοια

$c_{2i} \leq e_i$

Επίσης στην ακολουθία {c1, c2,...,cn}, υπάρχουν 2i στοιχεία των Α και Β <=c_{2i+1}, δηλ.

$d_{i+1} \leq c_{2i+1}$

και όμοια

$e_i \leq c_{2i+1}$

Επειδή

$c_{2i} \leq c_{2i+1}$

συνεπάγεται

$c_{2i} = \min(d_{i+1}, e_i)$, $c_{2i+1} = \max(d_{i+1}, e_i)$

Ανάλυση χρόνος

t(n) είναι ο χρόνος για συγχώνευση 2 ακολουθιών μήκους n η καθεμια σε (n,n) δίκτυο συγχώνευσης.

$$t(1) = 1, \quad n = 1$$
$$t(2n) = t(n) + 1, \quad n > 1$$

άρα

$$t(2n) = 1 + \log n \quad O(n) \text{ ακολουθιακός}$$

Επεξεργασίες

p(2n) = αριθμός συγκριτών σε (n,n) δίκτυο συγχώνευσης

$$p(1) = 1, \quad n = 1$$
$$p(2n) = 2p(n) + n - 1 \quad n \geq 1$$

άρα

$$p(2n) = 1 + n\log n$$

Κόστος

$$c(2n) = p(2n) \cdot t(2n)$$
$$= O(n\log^2 n)$$

Το δίκτυο δεν είναι βέλτιστη κάτω κλιμακωτός

$O(n) < O(n\log^2 n)$

→ θεωρητικός κάτος

Συγχώνευση CREW SM SIMD

Procedure BINARY SEARCH (S, x, k)

Βήμα 1: (1.1) $i \leftarrow 1$ το 1ο στοιχείο της ακολουθίας

(1.2) $h \leftarrow n$ το τελευταίο στοιχείο της ακολουθίας

(1.3) $k \leftarrow 0$ βοηθητική μεταβλητή

Βήμα 2: **όσο** $i \leq h$ **κάνε**

(2.1) $m \leftarrow \lfloor (i+h)/2 \rfloor$

(2.2) **αν** $x = s_m$ **τότε** (i) $k \leftarrow m$

(ii) $i \leftarrow h+1$

αλλιώς αν $x < s_m$ **τότε** $h \leftarrow m-1$

αλλιώς $i \leftarrow m+1$

τέλος αν

τέλος αν

τέλος όσο.

Πολυπλοκότητα: $O(\log n)$

Procedure SEQUENTIAL MERGE (A, B, C)

Βήμα 1: (1.1) $i \leftarrow 1$

(1.2) $j \leftarrow 1$.

$a_{r+1} \leftarrow \infty$, $b_{s+1} \leftarrow \infty$

Βήμα 2: **για** $k = 1$ **ως** $r + s$ **κάνε**

αν $a_i < b_j$ **τότε** (i) $c_k \leftarrow a_i$

(ii) $i \leftarrow i+1$

αλλιώς (i) $c_k \leftarrow b_j$

(ii) $j \leftarrow j+1$

τέλος αν

τέλος για.

$O(n) = \Omega(n) =$ Κάτω όριο συμπλοκότητας

Η SEQUENTIAL MERGE είναι βέλτιστη!

Παραλληλη Συγχώνευση

Υποθέτουμε ότι έχουμε στη διάθεσή μας $P_1, P_2, \ldots, P_N$ επεξεργαστές, $N \le r \le s$ και στη χειρότερη περίπτωση $r = s = n$. Επίσης κάθε επεξεργαστής είναι σε θέση να εκτελεί τις εξής ακολουθιακές διαδικασίες:

1. Τη διαδικασία SEQUENTIAL MERGE
2. Τη διαδικασία BINARY SEARCH

Η εφαρμογή της παραλληλίας στο πρόβλημα της συγχώνευσης βασίζεται στη μέθοδο του χωρισμού (partitioning). Δίνονται οι δύο ταξινομημένες ακολουθίες:

$$A = \{a_1, a_2, \ldots, a_r\} \text{ και } B = \{b_1, b_2, \ldots, b_s\}$$

οι οποίες χωρίζονται σε ένα πλήθος ζευγών υποακολουθιών έτσι ώστε να είναι δυνατός ο σχηματισμός της τελικής ταξινομημένης ακολουθίας από την ταυτόχρονη συγχώνευση των ζευγών υποακολουθιών.

Χωρισμός

$a_1 \ldots a_{\lceil r/N \rceil} \mid a_{\lceil r/N \rceil + 1} \ldots a_{2\lceil r/N \rceil} \mid a_{2\lceil r/N \rceil + 1} \ldots \mid \ldots \mid a_{i\lceil r/N \rceil + 1} \ldots a_{(i+1)\lceil r/N \rceil} \mid \ldots$

$b_1 \ldots b_{\lceil s/N \rceil} \mid b_{\lceil s/N \rceil + 1} \ldots b_{2\lceil s/N \rceil} \mid \ldots \mid b_{i\lceil s/N \rceil + 1} \ldots b_{(i+1)\lceil s/N \rceil} \mid \ldots$

a_1' a_2' b_1' b_2'

Εφαρμογή της BINARY SEARCH για την εύρεση του βαθμού του b_j στην A.

Συγκρίνεται το b_j με το μεσαίο στοιχείο της A. Ανάλογα με το αποτέλεσμα της σύγκρισης, η αναζήτηση περιορίζεται στο πρώτο ή δεύτερο μισό τμήμα της A. Η εφαρμογή αυτή συνεχίζεται (επαναλαμβάνεται) μέχρις ότου το b_j αποκτήσει μεταξύ δύο διαδοχικών στοιχείων της A, δηλαδή όταν

$$a_i < b_j < a_{i+1}$$

οπότε βαθμός $(b_j : A) = i$. Στο σημείο αυτό κάνω με χρήση της υπόθεσης ότι τα στοιχεία των A και B είναι διάφορα μεταξύ τους.

Το πρόβλημα της συγχώνευσης μπορεί να θεωρηθεί σαν το πρόβλημα του προσδιορισμού του βαθμού κάθε στοιχείου x από την A ή την B στην ακολουθία AUB. Αν βαθμός $(x : A \cup B) = i$, τότε $c_i = x$, όπου c_i είναι το i-οστό στοιχείο της τελικής ταξινομημένης ακολουθίας. Επειδή βαθμός $(x : A \cup B) =$ βαθμός $(x : A) +$ βαθμός $(x : B)$, το πρόβλημα της συγχώνευσης επιλύεται με τον προσδιορισμό των

βαθμός $(A:B]$ και βαθμός $(B:A)$

όπου

βαθμός $(Y:X) = (r_1, r_2, \ldots, r_s)$

με

$r_i =$ βαθμός $(y_i : X)$, $Y = \{y_1, y_2, \ldots, y_s\}$

Παρατηρήσεις

1. Αν $a_i = b_j$ τότε αυθαίρετα αποφασίζεται ότι το a_i είναι μικρότερο
2. Η πράξη του ταυτόχρονου διαβάσματος εκτελείται κατά τη διάρκεια της BINARY SEARCH. Σε κάθε τέτοια χρονική στιγμή ορισμένοι επεξεργαστές εκτελούν μία δυαδική αναζήτηση στην ίδια ακολουθία

Procedure CREW MERGE (A, B, C)

Βήμα 1: {Επέλεξε N-1 στοιχεία της A που υποδιαιρούν την ακολουθία αυτή σε N υποακολουθίες του ιδίου περίπου μεγέθους. Ονόμασε την υποακολουθία που δημιουργήθηκε από αυτά τα N-1 στοιχεία A΄. Δημιούργησε μια υποακολουθία B΄ από N-1 στοιχεία της B με τον ίδιο τρόπο. Η εκτέλεση του βήματος αυτού είναι η ακόλουθη:}

για i=1 **μέχρι** N-1 **κάνε παράλληλα**
Ο Επεξεργαστής P_i υπολογίζει το a'_i και b'_i από
(1.1) $a'_i \leftarrow a_{i\lceil r/N \rceil}$
(1.2) $b'_i \leftarrow b_{i\lceil s/N \rceil}$
τέλος για.

$a_i'=a_{i*r/N}$
$b_i'=b_{i*s/N}$

Βήμα 2: {Συγχώνευσε τις A΄ και B΄ σε μία ακολουθία τριάδων $V=\{v_1,v_2,\ldots,v_{2N-2}\}$, όπου κάθε τριάδα αποτελείται από ένα στοιχείο της A΄ ή της B΄, ακολουθούμενο από τη θέση του στην A΄ ή B΄, ακολουθούμενο από το όνομα της αρχικής ακολουθίας, δηλαδή, της A ή B. Αυτό γίνεται με τον ακόλουθο τρόπο:}

Για το παράδειγμα:
i=1 ?j: $a_i'<b_j'$
Είναι $a_1'<b_1'$-> j=1
άρα $v_{i+j-1}=v_{1+1-1}=v_1=$
$=(a_1',1,A)=(4,1,A)$

i=2 ?j: $a_i'<b_j'$
Είναι $a_2'<b_3'$-> j=3
άρα $v_{i+j-1}=v_{2+3-1}=v_4=$
$=(a_2',2,A)=(12,2,A)$

i=3 ?j: $a_i'<b_j'$
Είναι $a_3'<b_3'$-> j=3
άρα $v_{i+j-1}=v_{3+3-1}=v_5=$
$=(a_3',3,A)=(16,3,A)$

Όμοια Βήμα 2.2

(2.1) **για** i=1 **μέχρι** N-1 **κάνε παράλληλα**
(i) Ο Επεξεργαστής P_i χρησιμοποιεί την BINARY SEARCH στη B΄για να βρεί το μικρότερο j έτσι ώστε $a'_i < b'_j$
(ii) **αν** υπάρχει j **τότε** $v_{i+j-1} \leftarrow (a'_i, i, A)$
αλλιώς $v_{i+N-1} \leftarrow (a'_i, i, A)$
τέλος αν
τέλος για
(2.2) **για** i=1 **μέχρι** N-1 **κάνε παράλληλα**
(i) Ο Επεξεργαστής P_i χρησιμοποιεί την BINARY SEARCH στην A΄για να βρεί το μικρότερο j έτσι ώστε $b'_i < a'_j$
(ii) **αν** υπάρχει j **τότε** $v_{i+j-1} \leftarrow (b'_i, i, B)$
αλλιώς $v_{i+N-1} \leftarrow (b'_i, i, B)$
τέλος αν
τέλος για.

Βήμα 3: {Κάθε επεξεργαστής συγχωνεύει και εισάγει στην C τα στοιχεία δύο υποακολουθιών, ένα από την A και ένα από την B. Οι δείκτες των δύο στοιχείων (ένα από την A και ένα από την B) από όπου κάθε επεξεργαστής πρόκειται να αρχίσει τη συγχώνευση, υπολογίζονται πρώτα και αποθηκεύονται σε έναν πίνακα Q από ταξινομημένα ζεύγη. Το βήμα αυτό εκτελείται ως εξής:}

(3.1) $Q(1) \leftarrow (1, 1)$

(3.2) **για** i=2 **μέχρι** N **κάνε παράλληλα**

αν $v_{2i-2} = (a'_k, k, A)$ **τότε** ο επεξεργαστής P_i

(i) χρησιμοποιεί την BINARY SEARCH στο B για να βρεί το μικρότερο j τέτοιο ώστε $b_j > a'_k$

(ii) $Q(i) \leftarrow (k\lceil r/N \rceil, j)$

αλλιώς ο επεξεργαστής P_i

(i) χρησιμοποιεί την BINARY SEARCH στο A για να βρεί το μικρότερο j τέτοιο ώστε $a_j > b'_k$

(ii) $Q(i) \leftarrow (j, k\lceil s/N \rceil)$

τέλος αν

τέλος για

(3.3) **για** i=1 **μέχρι** N **κάνε παράλληλα**

Ο Επεξεργαστής P_i χρησιμοποιεί την SEQUENTIAL MERGE και τον Q(i)=(x,y) για να συγχωνεύσει δύο υποακολουθίες, η μία ξεκινώντας από το a_x και η άλλη από το b_y και τοποθετεί τα αποτελέσματα της συγχώνευσης στον πίνακα C αρχίζοντας στη θέση x+y-1. Η συγχώνευση συνεχίζεται μέχρις ότου

(i) βρεθεί ένα στοιχείο μεγαλύτερο ή ίσο με το πρώτο στοιχείο του v_{2i} σε κάθε μία από τις A και B (όταν $i \leq N-1$)

(ii) δεν υπάρχει κανένα στοιχείο στην A ή B (όταν i=N)

τέλος για.

Για τα Q: Από το V χρησιμοποιούμε μόνο τις ζυγές θέσεις (παρανθέσεις). Όταν έχουμε B στο V, ψάχνουμε να βρούμε το μεγαλύτερο στοιχείο από αυτό που έχουμε στο A, οπότε Q(θέση, k*s/N)

Όταν έχουμε A στο v, ψάχνουμε να βρούμε το μεγαλύτερο στοιχείο από αυτό που έχουμε στο B, οπότε Q(k*r/N,θέση), όπου k είναι το i στοιχείο στο v_i δηλ. στο (7,1,B)->k=1, στο (12,2,A) είναι k=2, στο (12,3,8) είναι k=3.

Ανάλυση

Βήμα 1. Κάθε επεξεργαστής υπολογίζει δύο δείκτες. Συνεπώς το βήμα αυτό απαιτεί σταθερό χρόνο. $O(1)$

Βήμα 2. Το βήμα αυτό αποτελείται από δύο εφαρμογές της BINARY SEARCH σε μία ακολουθία με $N-1$ στοιχεία, ακολουθούμενη η κάθε μία από μία εντολή κατανομής. Συνεπώς απαιτείται $O(\log N)$ χρόνος.

Βήμα 3. Το βήμα 3.1 απαιτεί από μία καταχώρηση σταθερού χρόνου. και το βήμα 3.2 απαιτεί το πολύ $O(\log s)$ $(r \leq s)$ χρόνο. Για την ανάλυση του βήματος 3.3, παρατηρούμε ότι η V περιέχει $2N-2$ στοιχεία τα οποία διαιρούν τη C σε $2N-1$ υποακολουθίες με μέγιστο μέγεθος ίσο με $\lceil r/N \rceil + \lceil s/N \rceil$. Το μέγιστο μέγεθος αυτό προκύπτει, αν, για παράδειγμα, ένα στοιχείο α' της A' είναι ίσο με ένα στοιχείο β' της B'. Τότε τα $\lceil r/N \rceil$ στοιχεία μικρότερα ή ίσα του α'; (και μεγαλύτερα ή ίσα

η ίδια του a'_{j-1}) είναι επίσης μικρότερα η ίδια του b'_j και όρισα τα $\lceil s/N \rceil$ στοιχεία μικρότερα η ίδια του b'_j (και μεγαλύτερα του b'_{j-1}) είναι επίσης μικρότερα η ίδια του a'_i. Στο βήμα αυτό κάθε επεξεργασία δημιουργεί δύο τέτοιες υποακολουθίες της C των οποίων το αρχικό μέγεθος δεν είναι μεγαλύτερο από $2(\lceil r/N \rceil + \lceil s/N \rceil)$, εκτός του P_N, ο οποίος δημιουργεί μόνο μία υποακολουθία της C. Συνεπώς η SEQUENTIAL MERGE διαρκεί το πολύ $O((r+s)/N)$ χρόνο.

Στην χειρότερη περίπτωση $r = s = n$ και επειδή $n \geq N$, ο χρόνος εκτέλεσης του αλγορίθμου ουσιαστικά είναι ο χρόνος εκτέλεσης του βήματος 3. Συνεπώς

$$t(n) = O((n/N) + \log n)$$

Επειδή $p(2n) = N$, $c(2n) = p(2n) \cdot t(2n) = O(n + N \log n)$

Ο αλγόριθμος έχει βέλτιστο κόστος αν $N \leq n/\log n$.

Παράδειγμα

a_1 a_2 a_3 a_4 a_5 a_6

Έστω $N=4$, $A=\{2,3,4,6,11,12,13,15,16,20,22,24\}$

b_1 b_2 b_3

$B=\{1,5,7,8,9,10,14,17,18,19,21,23\}$. δηλαδή

$r=s=12$

for i=1 to N-1 δηλ. για i=1,2,3
$a_1'=a_{1*3}=a_3$, $a_2'=a_{2*3}=a_6$

b1'=b3

Βήμα 1: $A'=\{4,12,16\}$, $B'=\{7,10,18\}$

$\lceil r/N \rceil = \lceil s/N \rceil = 3$ =12/4

Βήμα 2: Συγχώνευση των A', B' και δημιουργία της

Το 4 είναι το 1ο στοιχείο που πήρα από το A

Το 7 είναι το 1ο στοιχείο που πήρα από το B

Το 10 είναι το 2ο στοιχείο που πήρα από το B

Το 12 είναι το 2ο στοιχείο που πήρα από το A

$V=\{(4,1,A),(7,1,B),(10,2,B),(12,2,A),(16,3,A),(18,3,B)\}$

1 2 3 4 5 6

Βήμα 3: $Q(1)=(1,1)$, $Q(2)=(5,3)$, $Q(3)=(6,7)$

$Q(4)=(10,9)$ για τα Q βλ. επόμ. διαφάνεια

Βήμα 3.3: Ο επεξεργαστής P_1 δεχίζει από τα στοιχεία $a_1=2$ και $b_1=1$ και συγχωνεύει όλα τα στοιχεία των A και B μικρότερα του 7, δημιουργώντας την υποακολουθία $\{1,2,3,4,5,6\}$ της C. Ομοίως ο επεξεργαστής P_2 δεχίζει από τα στοιχεία $a_5=11$ και $b_3=7$ και συγχωνεύει όλα τα στοιχεία μικρότερα του 12, δημιουργώντας την

{7,8,9,10,11}. Ο επεξεργαστής P_3 δεχίζει από τα στοιχεία $a_6 = 12$ και $b_7 = 14$ και δημιουργεί την {12, 13, 14, 15, 16, 17}. Τέλος, ο P_4 αρχίζει από τα $a_{10} = 20$ και $b_9 = 18$ και δημιουργεί την {18, 19, 20, 21, 22, 23, 24}. Η τελική ακολουθία C είναι συνεπώς η

V_1 V_2 V_3 V_4 V_5 V_6

C = {1, 2, 3, (4), 5, 6, (7), 8, 9, (10), 11, (12), 13, 14, 15, (16), 17, (18), 19, 20, 21, 22, 23, 24}

στους κύκλους βρίσκονται τα στοιχεία των A' και B'

Συγχώνευση στο μοντέλο EREW

για τα Q:
Από το V χρησιμοποιούμε μόνο τις ζυγές θέσεις (παρενθέσεις)
Όταν έχουμε B στο V ψάχνουμε να βρούμε μεγαλύτερο στοιχείο από αυτό που έχουμε στο A, οπότε Q(θέση, k*s/N)
Όταν έχουμε A στο V ψάχνουμε να βρούμε μεγαλύτερο στοιχείο από αυτό που έχουμε στο B, οπότε Q(k*r/N, θέση)
όπου k είναι τα i στα v_i δηλαδή στο (7,1,B)->k=1, (12,2,A)->k=2, (18,3,B)->k=3

$N = 2^q$, $q \geq 1$ M = τοποθεσίες μνήμης → M(2N-1)

- Κάθε τοποθεσία μνήμης θεωρείται η ρίζα ενός δυαδικού δέντρου με N φύλλα. Ένα τέτοιο δέντρο έχει q+1 επίπεδα και 2N-1 κόμβους.

Οι κόμβοι του δέντρου αναπαριστούν τις αντίστοιχες τοποθεσίες μνήμης. Έτσι αν D είναι η ρίζα, το αριστερό και το δεξί παιδί είναι αντίστοιχα D+1, D+2. Γενικά, το αριστερο και δεξί παιδί του D+X είναι τα D+2X+1, D+2X+2, αντίστοιχα.
Έστω ότι ο επεξεργαστής P_i επιθυμεί να διαβαστεί κάποιο σημίο από μια τοποθεσία μνήμης d(i). Η αίτηση γίνεται στη θέση d(i)+(N-1)+(i-1)<- ένα φύλλο του δέντρου με ρίζα στην d(i)

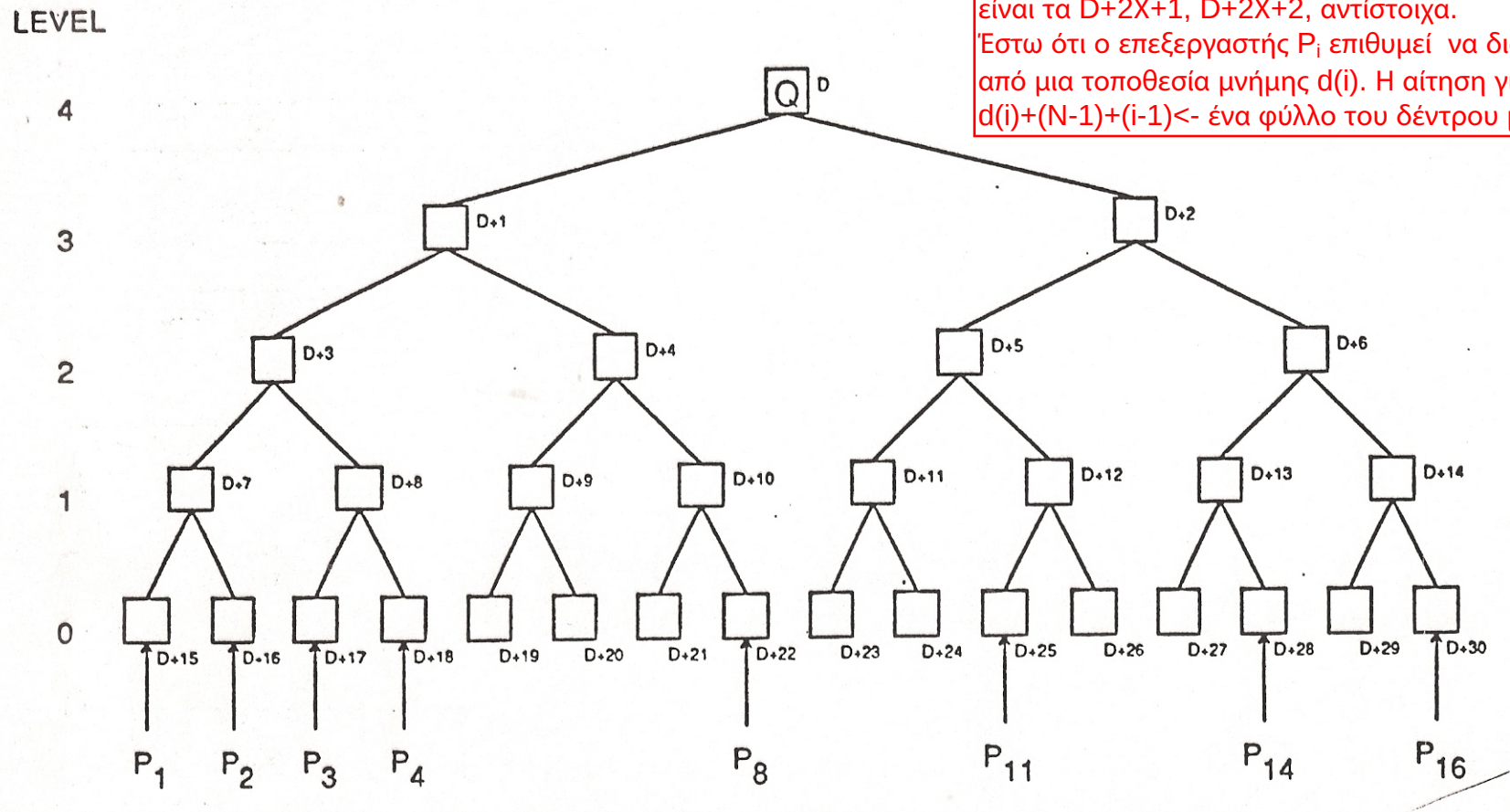
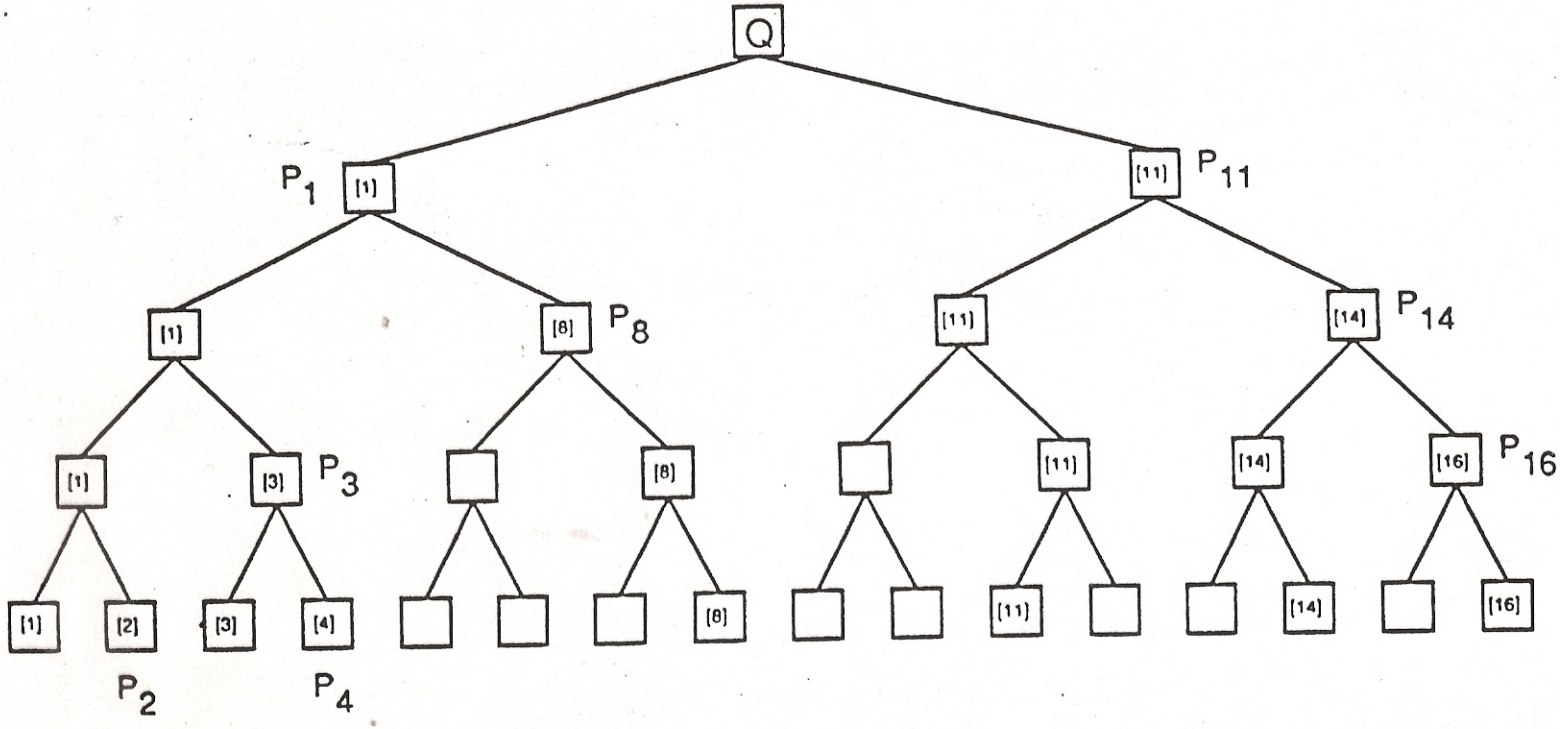
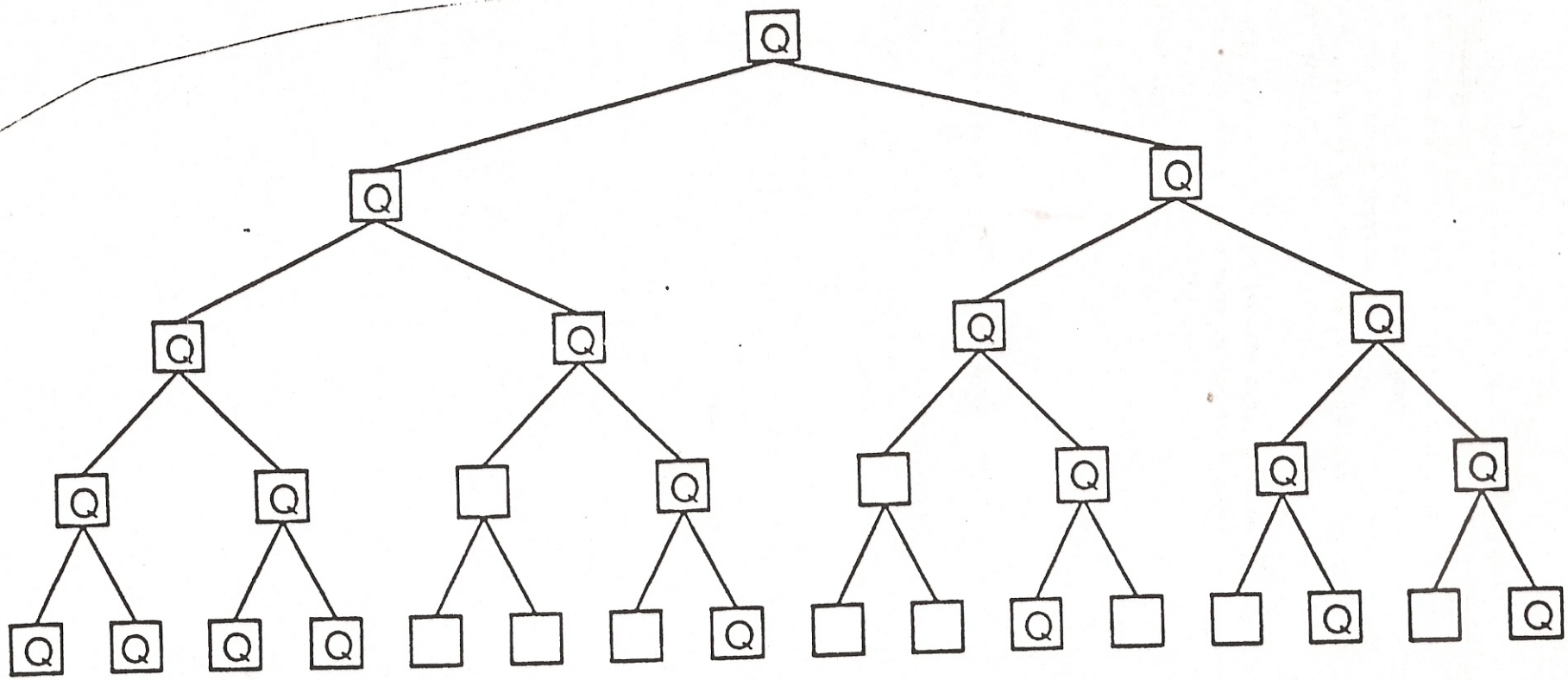


Figure 3.5 Memory organization for multiple broadcasting.

d(i)+(N-1)+(i-1) = φύλλο με ρίζα στην d(i)



Περιεχόμενα μνήματος μετά το βήμα 2
της MULTIPLE BROADCAST



Περιεχόμενα μνήμης στο τέλος της MULTIPLE BROADCAST

Procedure MULTIPLE BROADCAST (d(1), d(2), ..., d(N))

Βήμα 1: για i=1 **μέχρι** N **κάνε παράλληλα**
{Ο P_i δίνει αρχικές τιμές στο επίπεδο(i) και loc(i)}
(1.1) επίπεδο(i) ← 0
(1.2) loc(i) ← N+i-2
(1.3) αποθήκευσε [i] στην τοποθεσία d(i)+loc(i)
τέλος για.

O(1)

{Ανοδική πορεία στο δέντρο}

Βήμα 2: για v=0 **μέχρι** (log N)-2 **κάνε**
(2.1) **για** i=1 **μέχρι** N **κάνε παράλληλα** (προς τα
{ο P_i σε ένα αριστερό παιδί προχωράει πάνω στο δέντρο του}
(2.1.1) x ← ⌊(loc(i)-1)/2⌋
(2.1.2) **αν** loc(i) είναι περιττό και επίπεδο(i)=v
τότε (i) loc(i) ← x
(ii) αποθήκευσε [i] στη θέση d(i)+loc(i)
(iii) επίπεδο(i) ← επίπεδο(i)+1
τέλος αν
τέλος για
(2.2) **για** i=1 **μέχρι** N **κάνε παράλληλα**
{ο P_i σε ένα δεξί παιδί προχωράει πάνω στο δέντρο του, αν είναι δυνατόν}
αν d(i)+x δεν περιέχει ήδη ένα σημάδι[j] για κάποιο $1 \le j \le N$
τότε (i) loc(i) ← x
(ii) αποθήκευσε [i] στη θέση d(i)+loc(i)
(iii) επίπεδο(i) ← επίπεδο(i)+1
τέλος αν
τέλος για
τέλος για.

O(logN)

{Καθοδική πορεία στο δέντρο}

Βήμα 3: **για** v=(log N)-1 **μέχρι** 0 **κάνε**
(3.1) **για** i=1 **μέχρι** N **κάνε παράλληλα**
{ο P_i σε ένα αριστερό παιδί διαβάζει από τον πατέρα του και κατεβαίνει το δέντρο}
(3.1.1) x ← ⌊(loc(i)-1)/2⌋
(3.1.2) y ← (2 × loc(i))+1
(3.1.3) **αν** loc(i) είναι περιττό και επίπεδο(i)=v
τότε (i) διάβασε τα περιεχόμενα του d(i)+x
(ii) γράψε τα περιεχόμενα του d(i)+x στη θέση d(i)+loc(i)
(iii) επίπεδο(i) ← επίπεδο(i)-1
(iv) **αν** η θέση d(i)+y περιέχει [i]
τότε loc(i) ← y
αλλιώς loc(i) ← y+1
τέλος αν
τέλος αν
τέλος για
(3.2) **για** i=1 **μέχρι** N **κάνε παράλληλα**
{ο P_i σε ένα δεξί παιδί διαβάζει από τον πατέρα του και κατεβαίνει το δέντρο}
αν loc(i) είναι άρτιο και επίπεδο(i)=v
τότε (i) διάβασε το περιεχόμενο του d(i)+x
(ii) γράψε τα περιεχόμενα του d(i)+x στη θέση d(i)+loc(i)
(iii) επίπεδο(i) ← επίπεδο(i)-1
(iv) **αν** η θέση d(i)+y περιέχει [i]
τότε loc(i) ← y
αλλιώς loc(i) ← y+1
τέλος αν
τέλος αν
τέλος για
τέλος για.

O(logN)

Συνεπώς η MULTIPLE BROADCAST απαιτεί γραμμικό χρόνο O(logN)

Πολυπλοκότητα προσαρμογής της CREW MERGE για το EREW μοντέλο

$$t(2n) = \underbrace{O(\log N)}_{\text{MULTIPLE BROADCAST}} \cdot \underbrace{O(n/N + \log n)}_{\text{CREW MERGE}}$$

$$t(2n) = O\left((n/N)\log n + \log^2 n\right)$$

$(\log N \approx \log n)$

$$c(2n) = t(2n) \cdot p(2n) = O(n\log n + N\log^2 n)$$

το κόστος δεν είναι βέλτιστο. Επίσης η CREW MERGE χρησιμοποιεί $O(n)$ τοποθεσίες της κοινής μνήμης και οι λειτουργίες σε μνήμη της προσαρμογής της στο EREW είναι $O(Nn)$.

Ένας καλύτερος αλγόριθμος για το EREW μοντέλο

$A = \{a_1, a_2, \ldots, a_r\}$, $B = \{b_1, b_2, \ldots, b_s\}$

$C = \{c_1, c_2, \ldots, c_{r+s}\}$

$P_1, P_2, \ldots, P_N$, $1 \le N \le r+s$

Εύρεση του μέσου δύο ταξινομημένων ακολουθιών

Δεδομένων δύο ταξινομημένων ακολουθιών $A=\{a_1, a_2, \ldots, a_r\}$ και $B=\{b_1, b_2, \ldots, b_s\}$ όπου $r, s \geq 1$ συμβολίζουμε με $A.B$ την ακολουθία μήκους $m=r+s$ που προκύπτει από τη συγχώνευση των A και B. Ζητείται να βρεθεί το μέσο, δηλαδή το $\lceil m/2 \rceil$ στοιχείο της $A.B$ χωρίς να δημιουργηθεί (φυσικά) η $A.B$. Ο αλγόριθμος ελαττώνει ένα μέγεθος (a_x, b_y) που ικανοποιεί τις ιδιότητες:

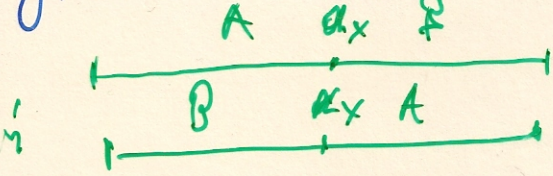
1. Εάν από τα a_x ή b_y είναι το μέσον της $A.B$ δηλαδή το a_x ή το b_y είναι μεγαλύτερο από $\lceil m/2 \rceil - 1$ στοιχεία και μικρότερο από $\lfloor m/2 \rfloor$ στοιχεία
2. Αν το a_x είναι το μέσο τότε το b_y θέτει είναι
   (i) το μεγαλύτερο στοιχείο του B μικρότερο ή ίσο του a_x ή
   (ii) το μικρότερο στοιχείο του B μεγαλύτερο ή ίσο του a_x

Διαφορετικά αν το b_y είναι το μέσον, τότε a_x είναι ένα από:

(i) το μεγαλύτερο στοιχείο της A και μικρότερο ή ίσο του b_y ή

(ii) το μικρότερο στοιχείο της A μεγαλύτερο ή ίσο του b_y.

3. Αν οποιοδήποτε από τα δύο ζεύγη ικανοποιεί τη 1 και 2, τότε ο παρατηρητής επιλέγει το ζεύγος για το οποίο το x+y είναι το μικρότερο.



Το (a_x, b_y) δείχνει μεσαίο ζεύγος της A,B δηλαδή x και y είναι οι δείκτες του μεσαίου ζεύγους. Ας υποθέσουμε ότι a_x είναι το μέσον της A,B αν

(i) $a_x > b_y$ και $x+y-1 = \lceil m/2 \rceil - 1$

ή

(ii) $a_x < b_y$ και $m-(x+y-1) = \lfloor m/2 \rfloor$

Διαφορετικά το b_y είναι το μέσον της A,B.

Εύρεση μέσου δύο ταξινομημένων ακολουθιών

Procedure TWO-SEQUENCE MEDIAN (A, B, x, y)

Βήμα 1:
(1.1) $\chi\alpha\mu\eta\lambda\acute{o}_A \leftarrow 1$ — αριστερός δείκτης ακολουθίας A
(1.2) $\chi\alpha\mu\eta\lambda\acute{o}_B \leftarrow 1$ — αριστερός δείκτης ακολουθίας B
(1.3) $\psi\eta\lambda\acute{o}_A \leftarrow r$ — δεξί δείκτης ακολουθίας A
(1.4) $\psi\eta\lambda\acute{o}_B \leftarrow s$ — δεξί δείκτης ακολουθίας B
(1.5) $n_A \leftarrow r$ — πλήθος στοιχείων της A
(1.6) $n_B \leftarrow s$. — πλήθος στοιχείων της B

Βήμα 2: **όσο** $n_A > 1$ **και** $n_B > 1$ **κάνε**
(2.1), (2.2)-> μεσαίες θέσεις των A, B
(2.1) $u \leftarrow \chi\alpha\mu\eta\lambda\acute{o}_A + \lceil(\psi\eta\lambda\acute{o}_A - \chi\alpha\mu\eta\lambda\acute{o}_A - 1)/2\rceil$
(2.2) $v \leftarrow \chi\alpha\mu\eta\lambda\acute{o}_B + \lceil(\psi\eta\lambda\acute{o}_B - \chi\alpha\mu\eta\lambda\acute{o}_B - 1)/2\rceil$
(2.3) $w \leftarrow \min(\lfloor n_A/2\rfloor, \lfloor n_B/2\rfloor)$
(2.4), (2.5)-> ενημέρωση των A,B αφού αφαιρεθούν τα τμήματα
(2.4) $n_A \leftarrow n_A - w$
(2.5) $n_B \leftarrow n_B - w$
(2.6) **αν** $a_u \geq b_v$
τότε (i) $\psi\eta\lambda\acute{o}_A \leftarrow \psi\eta\lambda\acute{o}_A - w$
(ii) $\chi\alpha\mu\eta\lambda\acute{o}_B \leftarrow \chi\alpha\mu\eta\lambda\acute{o}_B + w$
αλλιώς (i) $\chi\alpha\mu\eta\lambda\acute{o}_A \leftarrow \chi\alpha\mu\eta\lambda\acute{o}_A + w$
(ii) $\psi\eta\lambda\acute{o}_B \leftarrow \psi\eta\lambda\acute{o}_B - w$
τέλος αν
τέλος όσο.

$O(\log n)$

$O(\log(\min\{r, s\}))$

ΠΡΟΣΟΧΗ! Η συνάρτηση αυτή επιστρέφει τους δείκτες του μεσαίου ζεύγους. Όχι το ίδιο το ζεύγος.

Βήμα 3: Επέστρεψε σαν x και y τους δείκτες του ζεύγους από $\{\alpha_{u-1}, \alpha_u, \alpha_{u+1}\} \times \{b_{v-1}, b_v, b_{v+1}\}$ ικανοποιώντας τις ιδιότητες 1-3 ενός μεσαίου ζεύγους.

Παράδειγμα

$A = \{10, 11, 12, 13, 14, 15, 16, 17, 18\}$

$B = \{3, 4, 5, 6, 7, 8, 14, 20, 21, 22\}$

$low_A = low_B = 1 \quad high_A = n_A = 9 \quad high_B = n_B = 10$

Β.2

1η επανάληψη

$u = v = 5 \quad w = \min(4, 5) = 4, \; n_A = 5, \; n_B = 6$

Επειδή $a_5 > b_5 \quad high_A = low_B = 5$

1η επανάληψη

$u = 3, \; v = 7, \; w = \min(2, 3) = 2, \; n_A = 3$

$n_B = 4$. Επειδή $a_2 < b_7 \quad low_A = 3$ και

$high_B = 8$

3η επανάληψη

$u = 4, \; v = 6, \; w = \min(1, 2) = 1 \quad n_A = 2,$

$n_B = 3$. Επειδή $a_4 > b_6 \quad high_A = 4$

$low_B = 6$

4η επανάληψη

$u = 3, \; v = 7 \quad w = \min(1, 1) = 1 \quad n_A = 1$ και

$n_B = 1$. Επειδή $a_3 < b_7 \quad low_A = 4$ και

$high_B = 7$

Βήμα 3

Αυτά που ικανοποιούν τις δύο πρώτες ιδιότητες είναι δύο τα: Έχουμε το ζεύγμα $\{11,12,13\}\times\{8,19,20\}$

$(a_4, b_6) = (13, 8)$ και $(a_4, b_7) = (13, 14)$

όπου επιβεβαιώνεται το $(4,6)$.

Συγχώνευση στο μοντέλο EREW

Δεδομένων δύο ταξινομημένων ακολουθιών $A = \{a_1, a_2, \ldots, a_r\}$ και $B = \{b_1, b_2, \ldots, b_s\}$ υποθέτουμε την ύπαρξη N επεξεργαστών $P_1, P_2, \ldots, P_N$, όπου N είναι μια δύναμη του 2 και $1 \le N \le r+s$. Ο αλγόριθμος που θα αναπτυχθεί στη συνέχεια συγχωνεύει τις A και B σε μία ταξινομημένη ακολουθία $C = \{c_1, c_2, \ldots, c_{r+s}\}$ σε δύο φάσεις ως εξής:

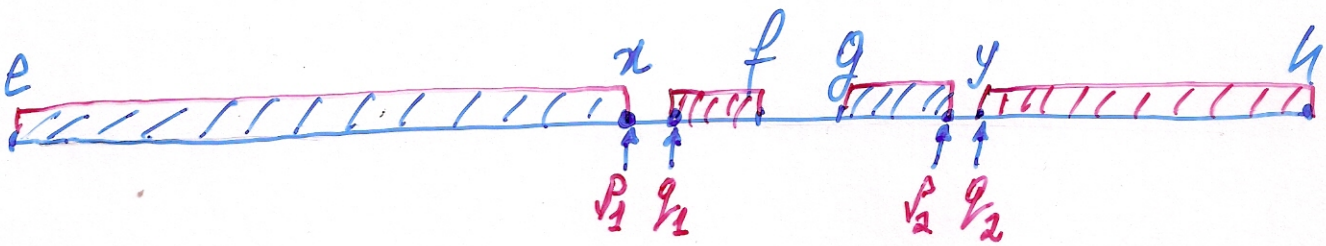
Φάση 1: Κάθε μία από τις δύο ακολουθίες A και B χωρίζονται σε N (πιθανά κενές) υποακολουθίες $A_1, A_2, \ldots, A_N$ και $B_1, B_2, \ldots, B_N$ τέτοια ώστε

(i) $|A_i| + |B_i| = (r+s)/N$ για $1 \le i \le N$

(ii) όλα τα στοιχεία στην $A_i \cdot B_i$ είναι μικρότερα ή ίσα από όλα τα στοιχεία στην $A_{i+1} \cdot B_{i+1}$ για $1 \le i \le N$

Περίπτωση όπου

- a_x είναι το μεσαίο στοιχείο
- $a_x \leq b_y$

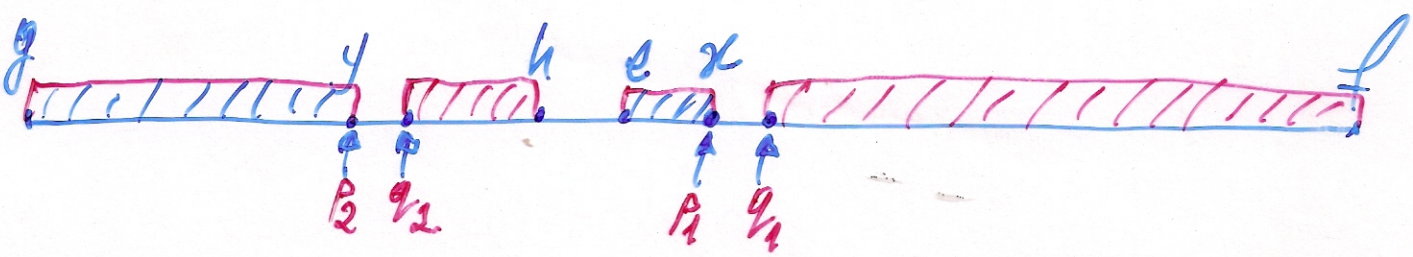


Παρατήρηση

- Οι υπακολουθίες με τα μικρότερα στοιχεία στέλνονται σε έναν επεξεργαστή P_{2i-1}, ενώ οι άλλες με τα μεγαλύτερα στοιχεία στέλνονται στον επεξεργαστή P_{2i}
- Το b_y συμπεριλαμβάνεται στην υπακολουθία με τα μεγαλύτερα στοιχεία

Περίπτωση όπου

- a_x είναι το μεσαίο στοιχείο
- $b_y \leq a_x$



Procedure EREW MERGE (A, B, C)

Βήμα 1: (1.1) Ο Επεξεργαστής P_1 λαμβάνει την τετράδα (1, r, 1, s)

(1.2) **Για** j=1 **ως** log N **κάνε**

Για i=1 **ως** 2^{j-1} **κάνε** παράλληλα

Ο Επεξεργαστής P_i έχοντας λάβει την τετράδα (e, f, g, h)

(1.2.1) {Βρίσκει το μεσαίο ζεύγος των δύο ακολουθιών

TWO-SEQUENCE MEDIAN (A[e, f], B[g, h], x, y)

(1.2.2) {Υπολογίζει τέσσερις δείκτες p_1, p_2, q_1 και q_2 ως εξής}

αν α_x είναι το μεσαίο

τότε (i) $p_1 \leftarrow x$

(ii) $q_1 \leftarrow x+1$

(iii) **αν** $b_y \leq \alpha_x$ **τότε** (a) $p_2 \leftarrow y$

(b) $q_2 \leftarrow y+1$

αλλιώς (a) $p_2 \leftarrow y-1$

(b) $q_2 \leftarrow y$

τέλος αν

αλλιώς(i) $p_2 \leftarrow y$

(ii) $q_2 \leftarrow y+1$

(iii) **αν** $\alpha_x \leq b_y$ **τότε** (a) $p_1 \leftarrow x$

(b) $q_1 \leftarrow x+1$

αλλιώς (a) $p_1 \leftarrow x-1$

(b) $q_1 \leftarrow x$

τέλος αν

τέλος αν

(1.2.3) Μεταδίδει τις τετράδες (e, p_1, g, p_2) στον P_{2i-1}

(1.2.4) Μεταδίδει τις τετράδες (q_1, f, q_2, h) στον P_{2i}

τέλος για

τέλος για.

Βήμα 2: για i=1 **ως** N **κάνε** παράλληλα

Ο Επεξεργαστής P_i έχοντας λάβει την τετράδα (a, b, c, d)

(2.1) $w \leftarrow 1+((i-1)(r+s))/N$

(2.2) $z \leftarrow \min \{i(r+s)/N, (r+s)\}$

(2.3) SEQUENTIAL MERGE (A[a, b], B[c, d], C[w, z])

τέλος για.

e-> κάτω όριο της A
f ->άνω όριο της A
g->κάτω όριο της B
h->άνω όριο της B

Το πλήθος των στοιχείων που θα πάρει ο κάθε επεξεργαστής είναι το ίδιο και ίσο με (r+s)/n. Άρα ο P_1 θα πάρει από 0 έως (r+s)/n στοιχεία, ο P_2 από (r+s)/n έως 2(r+s)/n κλπ. Έτσι εξασφαλίζεται το EREW.

x,y einai to mesaio zeugari twn A,B

Θεωρούμε σταθερό χρόνο μετάδοσης

w,z -> θέσεις που θα γράψει ο κάθε επεξεργαστής

(ii) Όλα τα ζεύγη A_i και B_i, $1 \leq i \leq N$ συγχωνεύονται ταυτόχρονα και τοποθετούνται στην C.

Παράδειγμα

$A=\{10,11,12,13,14,15,16,17,18\}$,

$B=\{3,4,5,6,7,8,19,20,21,22\}$, $N=4$

$r=9$, $s=10$

Βήμα 1.1 : Ο P_1 παρατάσσει $(1,9,1,10)$

Β. 1.2 Ο P_1 προσδιορίζει το μεσαίο στοιχείο των A και B που είναι το $(4,6)$. Διατηρεί το $(1,4,1,6)$ και μεταδίδει το $(5,9,7,10)$ στον P_2. Στη δεύτερη επανάληψη, ο P_1 υπολογίζει τους δείκτες του μεσαίου στοιχείου των $A[1,4]=\{10,11,12,13\}$ και $B[1,6]=\{3,4,5,6,7,8\}$ που είναι το $(1,5)$. Ταυτόχρονα ο P_2 κάνει το ίδιο με τις $A[5,9]=\{14,15,16,17,18\}$ και $B[7,10]=\{19,20,21,22\}$ και παρατάσσει το στοιχείο $(9,7)$. Ο P_1 κρατά το $(1,0,1,5)$ και μεταδίδει το $(1,4,6,6)$ στον P_2. Όμοια ο P_2 μεταδίδει το $(5,9,7,6)$ στον P_3 και το $(10,9,7,10)$ στον P_4

Στο βήμα 2 οι P_1, P_2, P_3 και P_4 διαβιβάζουν ταυτόχρονα την $C[1,19]$ ως εξής. Η τελευταία τετράδα που έλαβε ο P_1 είναι η (1,0,1,5) και ο P_1 υπολογίζει $W=1, Z=5$ και ανιχνεύει την $B[1,5]=\{3,4,5,6,7\}$ στην $C[1,5]$. Όμοια ο P_2, έχοντας λάβει τελευταία την (1,4,6,6), υπολογίζει $W=6$ και $Z=10$ και ανιχνεύει τις $A[1,4]$ και $B[6,6]$ για το ενδιαφέρον της $C[6,10]=\{8,10,11,12,13\}$. Ο P_3 έχοντας λάβει την τελευταία (5,9,7,6) υπολογίζει $W=11$ και $Z=15$ και ανιχνεύει την $A[5,9]=\{14,15,16,17,18\}$ στην $C[11,15]$. Τέλος ο P_4 έχοντας λάβει στο τέλος την (10,9,7,10) υπολογίζει $W=16$, $Z=19$ και ανιχνεύει την $B[7,10]=\{19,20,21,22\}$ στην $C[16,19]$.

Ανάλυση

Βήμα 1.1. Ο P_1 διαβάζει από την μνήμη σε σταθερό χρόνο

Βήμα 1.2. Κάθε μία από τις j-τάξεις επαναλήψεις κάθε επεξεργαστής υπολογίζει το μέσο, έστω m, από $(r+s)/2^{j-1}$ στοιχεία. Η εργασία αυτή γίνεται με την κλήση της TWO-SEQUENCE MEDIAN σε $O(\log[(r+s)/2^{j-1}])$ χρόνο, ή $O(\log(r+s))$. Τα άλλα δύο βήματα στο βήμα 1.2 χρειάζονται σταθερό χρόνο.

Επειδή εκτελούνται $\log N$ επαναλήψεις του βήματος 1.2 το Βήμα 1 χρειάζεται

Βήμα 1: $O(\log N \cdot \log(r+s))$ χρόνο

Στο Βήμα 2 κάθε επεξεργαστής συγχωνεύει το πολύ $(r+s)/N$ στοιχεία. Η εργασία αυτή γίνεται με την SEQUENTIAL MERGE και χρειάζεται $O((r+s)/N)$ χρόνο.

τα βήματα 1 και 2 θέλουν χρόνο

Συνολικά

$$O((r+s)/N + \log N \cdot \log(r+s))$$

Στην χειρότερη περίπτωση $r = s = n$ και

$$t(2n) = O(n/N + \log^2 n)$$

με κόστος

$$c(2n) = O(n + N\log^2 n)$$

Επειδή $\Omega(n)$ είναι το κάτω φράγμα του αριθμού των πράξεων για συγχώνευση το δυνατόν κόστος είναι βέλτιστο όταν $N \leq n/\log^2 n$.

$\log^2 n$ αύξουσα συνάρτηση σχεδόν flat δηλ. έχω βέλτιστο κόστος για μεγάλο πλήθος επεξεργαστών.
Ο αλγόριθμος αυτός αποτελεί σπάνια περίπτωση βέλτιστου κόστους με μεγάλο πλήθος επεξεργαστών.