ΠΑΡΑΛΛΗΛΟΙ ΑΛΓΟΡΙΘΜΟΙ

ΕΔΙΠ Μαρία Λουκά

Τμήμα Πληροφορικής και Τηλεπικοινωνιών

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

28 Απριλίου 2020

Κεφάλαιο 4
Ταξινόμηση (Sorting)

Δεδομένης μιας ακολουθίας $S = \{s_1, s_2, \cdots, s_n\}$ από n στοιχεία στα οποία έχει οριστεί η γραμμική διάταξη $<$, να βρεθεί μια νέα ακολουθία $S' = \{s'_1, s'_2, \cdots, s'_n\}$ τέτοια ώστε $s'_i < s'_{i+1}$ για $i = 1, 2, \cdots, n-1$.

Ένα κάτω όριο

- Η ταξινόμηση με σύγκριση θεωρητικά μπορεί να μελετηθεί με δέντρα απόφασης (decision trees)
- Κάθε μία από τις $n!$ μεταθέσεις των n στοιχείων εμφανίζεται σαν ένα από τα φύλλα του δέντρου απόφασης προκειμένου ο αλγόριθμος ταξινόμησης να ταξινομεί σωστά
- Ο αριθμός των συγκρίσεων στη χειρότερη περίπτωση αντιστοιχεί με το ύψος του δέντρου απόφασης
- Κάτω όριο των υψών των δέντρων απόφασης $\longrightarrow$ κάτω όριο του χρόνου εκτέλεσης οποιουδήποτε αλγόριθμου ταξινόμησης με σύγκριση

Θεώρημα

Ένα δέντρο απόφασης, το οποίο ταξινομεί n στοιχεία έχει ύψος $\Omega(n \log n)$

Ένα κάτω όριο

Απόδειξη.

Επειδή υπάρχουν $n!$ μεταθέσεις των n στοιχείων, το δέντρο πρέπει να έχει τουλάχιστον $n!$ φύλλα. Επειδή ένα δυαδικό δέντρο ύψους h έχει το πολύ 2^h φύλλα, έχουμε

$$n! \leq 2^h \qquad \text{ή} \qquad h \geq \log(n!)$$

επειδή η $\log$ είναι μονότονα αύξουσα.
Αλλά από την προσέγγιση Stirling έχουμε

$$n! \geq \left(\frac{n}{e}\right)^n, e = 2.71828$$

συνεπώς

$$h \geq \log\left(\frac{n}{e}\right)^n = n\log n - n\log e$$

ή

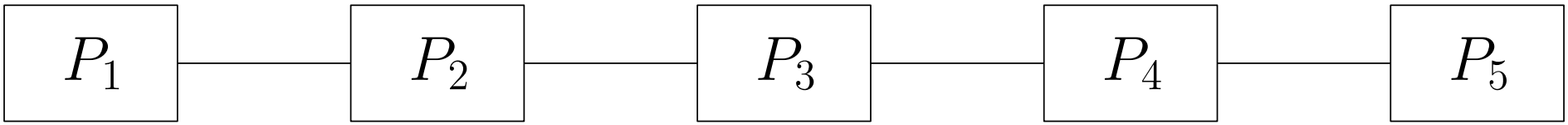
$$h \geq \Omega(n\log n).$$

Άρα ένας αλγόριθμος ταξινόμησης απαιτεί $\Omega(n\log n)$ πολυπλοκότητα στη χειρότερη περίπτωση.

Συνεπώς για ένα παράλληλο αλγόριθμο ταξινόμησης που χρησιμοποιεί N επεξεργαστές έχουμε $\Omega((n\log n)/N), N \leq n\log n$. □

Ταξινόμηση σε ένα μονοδιάστατο πίνακα επεξεργαστών

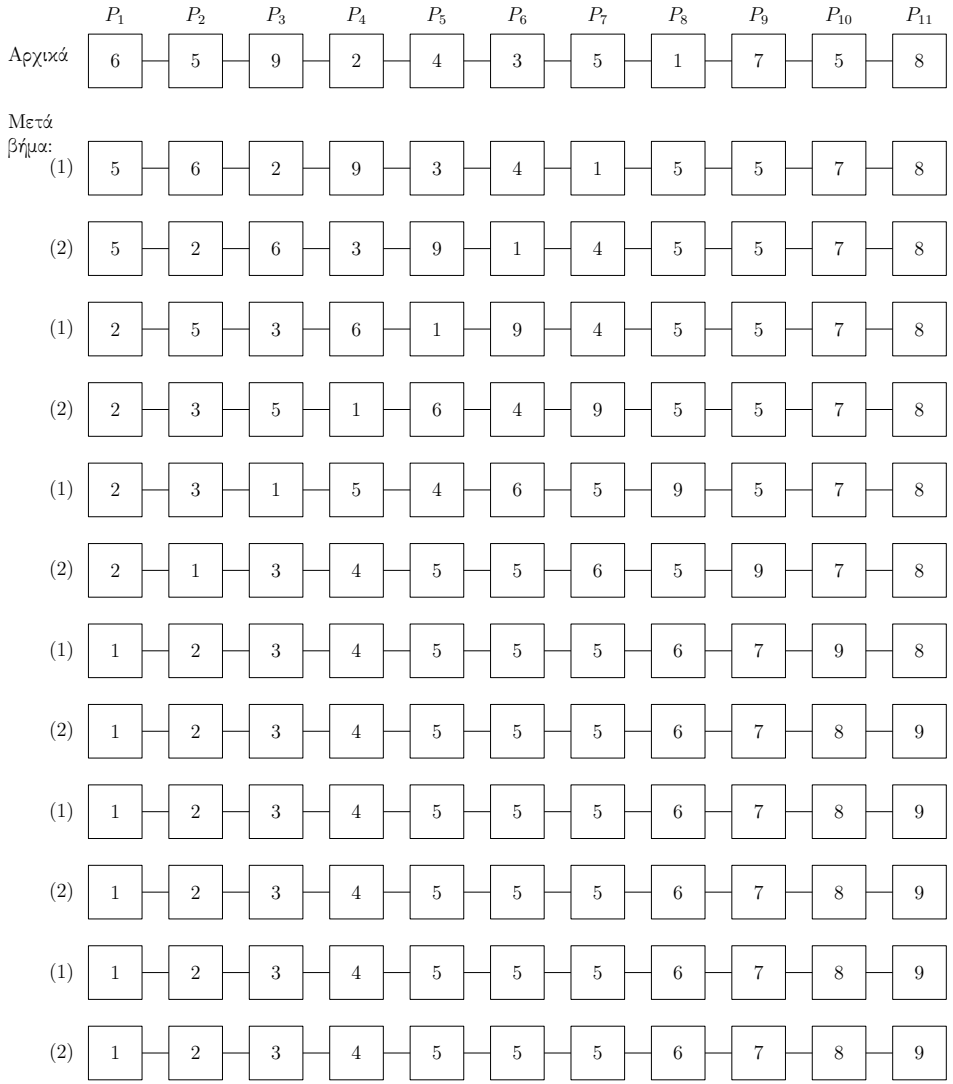
Υποθέτουμε ότι έχουμε ένα SIMD υπολογιστή όπου οι επεξεργαστές είναι συνδεδεμένοι όπως στο σχήμα:



Σχήμα: Σύνδεση επεξεργαστών

Ο αλγόριθμος που θα αναπτυχθεί για την ταξινόμηση της ακολουθίας $S = \{s_1, s_2, \cdots, s_n\}$ χρησιμοποιεί n επεξεργαστές $P_1, P_2, \cdots, P_N$.

Ταξινόμηση σε ένα μονοδιάστατο πίνακα επεξεργαστών



Σχήμα: Ακολουθία ταξινόμησης 11 στοιχείων με τη διαδικασία ODD-EVEN TRANSPOSITION

Ταξινόμηση σε ένα μονοδιάστατο πίνακα επεξεργαστών

Διαδικασία 1: QUICKSORT (S)

αν $|S| = 2$ ***και*** $S_2 < S_1$ **τότε**
  $S_1 \leftrightarrow S_2$
αλλιώς
  αν $|S| > 2$ **τότε**
    (1) {Προσδιορισμός του m, μέσου στοιχείου της S}
    SEQUENTIAL SELECT $(S, \lceil |S|/2 \rceil)$
    (2) {Χωρισμός της S σε δύο υποακολουθίες S_1 και S_2}
      (2.1) $S_1 \leftarrow \{s_i : s_i \leq m\}$ και $|S_1| = \lceil |S|/2 \rceil$
      (2.2) $S_2 \leftarrow \{s_i : s_i \geq m\}$ και $|S_2| = \lceil |S|/2 \rceil$
    (3) QUICKSORT (S_1)
    (4) QUICKSORT (S_2)
  τέλος
τέλος

$$t(n) = cn + 2t(n/2), \quad c = \text{σταθερά}$$

$$t(n) = O(n \log n) \text{βέλτιστος χρόνος}$$

Ταξινόμηση σε ένα μονοδιάστατο πίνακα επεξεργαστών

Διαδικασία 2: ODD-EVEN TRANSPOSITION (S)

για $j = 1$ **έως** $\lceil n/2 \rceil$ **κάνε**

(1) **για** $i = 1, 3, \cdots,$ **έως** $2\lfloor n/2 \rfloor - 1$ **κάνε παράλληλα**

αν $x_i > x_{i+1}$ **τότε**

$x_i \leftrightarrow x_{i+1}$

τέλος

τέλος

(2) **για** $i = 2, 4, \cdots,$ **έως** $2\lfloor (n-1)/2 \rfloor - 1$ **κάνε παράλληλα**

αν $x_i > x_{i+1}$ **τότε**

$x_i \leftrightarrow x_{i+1}$

τέλος

τέλος

τέλος

Ταξινόμηση σε ένα μονοδιάστατο πίνακα επεξεργαστών

- Κατά τη διάρκεια εκτέλεσης του αλγορίθμου ο επεξεργαστής P_i έχει ένα στοιχείο $x_i, i = 1, 2, \cdots, n$
- Αρχικά $x_i = s_i$
- Ο αλγόριθμος επαναλαμβάνει στην ουσία δύο βήματα:
  1. όλοι οι περιττοί επεξεργαστές P_i λαμβάνουν το x_{i+1} από τον P_{i+1}. Αν $x_i > x_{i+1}$ τότε οι P_i και P_{i+1} ανταλλάσουν τα στοιχεία τους
  2. όλοι οι άρτιοι επεξεργαστές εκτελούν την ίδια εργασία όπως οι περιττοί στο πρώτο βήμα
- Μετά από $\lceil n/2 \rceil$ επαναλήψεις αυτών των δύο βημάτων θα έχουν τελειώσει όλες οι ανταλλαγές των στοιχείων

Ταξινόμηση σε ένα μονοδιάστατο πίνακα επεξεργαστών

Ανάλυση

$$t(n) = O(n), \quad p(n) = n, \quad c(n) = O(n^2)$$

άρα δεν είναι βέλτιστος.

Παρατηρήσεις

- Πετυχαίνει σε σχέση με την Quicksort μια ταχύτητα $O(\log n)$
- Χρησιμοποιεί μεγάλο αριθμό επεξεργαστών
- Δεν είναι βέλτιστου κόστους

Ταξινόμηση σε ένα μονοδιάστατο πίνακα επεξεργαστών

Νέος Αλγόριθμος

Υποθέτουμε ότι κάθε ένας P_i από τους $N < n$ επεξεργαστές έχει μία υποακολουθία s_i της S μήκους n/N (προσθέτουμε εικονικά στοιχεία αν το n δεν είναι πολ/σιο του N).

Στον νέο αλγόριθμο οι συγκρίσεις - ανταλλαγές αντικαθίστανται με συγχώνευση - χωρισμό στις υποακολουθίες.

Στο βήμα 1 κάθε επεξεργαστής P_i ταξινομεί την S_i με την QUICKSORT.

Στο βήμα 2.1 κάθε περιττός επεξεργαστής συγχωνεύει τις δύο υποακολουθίες S_i και S_{i+1} σε μία ταξινομημένη ακολουθία $S'_i = \{s'_1, s'_2, \cdots, s'_{2b/N}\}$. Διατηρεί το πρώτο μέρος της S'_i και καταχωρεί στον γειτονικό του επεξεργαστή P_{i+1} το δεύτερο τμήμα.

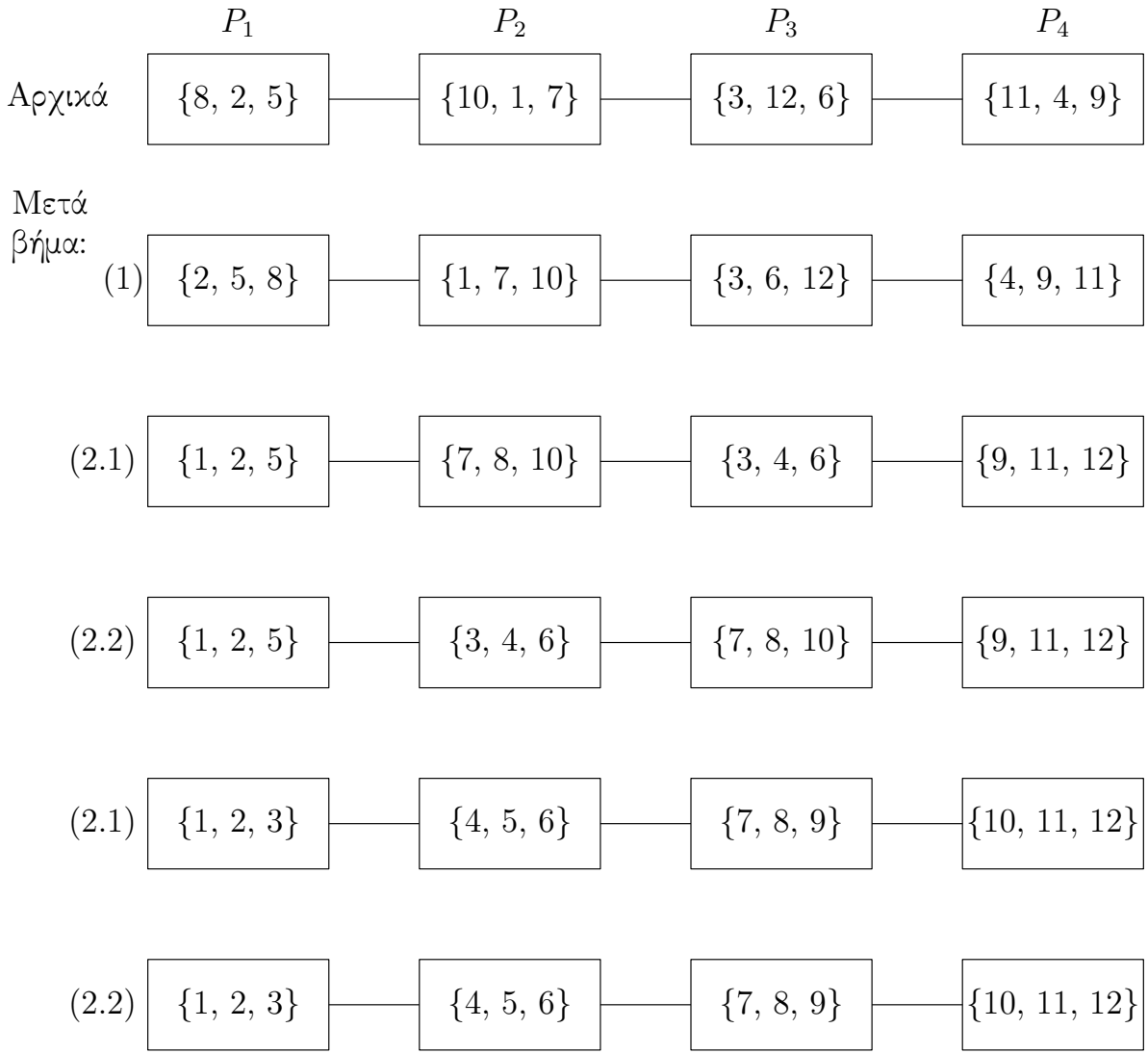
Ταξινόμηση σε ένα μονοδιάστατο πίνακα επεξεργαστών

Το βήμα 2.2 είναι το ίδιο με το 2.1 εκτός ότι αυτή τη φορά χρησιμοποιούνται οι άρτιοι επεξεργαστές.

Μετά από $\lceil N/2 \rceil$ επαναλήψεις έχουν τελειώσει όλες οι ανταλλαγές των στοιχείων μεταξύ δύο επεξεργαστών. Στο τέλος η ακολουθία $S = \{s_1, s_2, \cdots, s_N\}$ είναι ταξινομημένη.

Ταξινόμηση σε ένα μονοδιάστατο πίνακα επεξεργαστών

$S = \{8, 2, 5, 10, 1, 7, 3, 12, 6, 11, 4, 9\}, \quad N = 4, \quad n = 12$



Σχήμα: Ακολουθία ταξινόμησης 12 στοιχείων με τη διαδικασία MERGE SPLIT

Ταξινόμηση σε ένα μονοδιάστατο πίνακα επεξεργαστών

Διαδικασία 3: MERGE SPLIT (S)

- **Βήμα 1**

για $i = 1$ **έως** N **κάνε παράλληλα**
  QUICKSORT (S_i)
τέλος

- **Βήμα 2**

για $j = 1$ **έως** $\lceil N/2 \rceil$ **κάνε**
  (2.1) **για** $i = 1, 3, \cdots,$ **έως** $2\lceil N/2 \rceil - 1$ **κάνε παράλληλα**
    (i) SEQUENTIAL MERGE (S_i, S_{i+1}, S'_i)
    (ii) $S_i \leftarrow \{s'_1, s'_2, \cdots, s'_{n/N}\}$
    (iii) $S_{i+1} \leftarrow \{S'_{(n/N)+1}, S'_{(n/N)+2}, \cdots, S'_{2n/N}\}$
  τέλος
  (2.2) **για** $i = 2, 4, \cdots,$ **έως** $2\lfloor (N-1)/2 \rfloor$ **κάνε παράλληλα**
    (i) SEQUENTIAL MERGE (S_i, S_{i+1}, S'_i)
    (ii) $S_{i+1} \leftarrow \{s'_1, s'_2, \cdots, s'_{n/N}\}$
    (iii) $S_{i+1} \leftarrow \{S'_{(n/N)+1}, S'_{(n/N)+2}, \cdots, S'_{2n/N}\}$
  τέλος
τέλος

Ταξινόμηση σε ένα μονοδιάστατο πίνακα επεξεργαστών

Ανάλυση

Το Βήμα 1 απαιτεί $O((n/N)\log(n/N))$ βήματα. Μεταφορά της S_{i+1} στον P_i, συγχώνευση με SEQUENTIAL MERGE, και επιστροφή της S_{i+1} στον P_{i+1} απαιτούν $O(n/N)$ χρόνο. Συνεπώς

$$t(n) = O((n/N)\log(n/N)) + \lceil N/2 \rceil \cdot O(n/N)$$

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

$$c(n) = O(n\log n) + O(nN)$$

το οποίο είναι βέλτιστο όταν $N \leq \log n$.

Ταξινόμηση στο CRCW Μοντέλο

Υποθέτουμε ότι έχουμε το SM SIMD CRCW μοντέλο, όπου οι συγκρούσεις γραφής (προσπάθεια γραφής διαφορετικών δεδομένων στην ίδια θέση μνήμης) επιλύονται με την αποθήκευση του αθροίσματος των ακεραίων στη διεύθυνση αυτή.

Υποθέτουμε επίσης ότι n^2 επεξεργαστές είναι διαθέσιμοι στο $CRCW$ μοντέλο για την ταξινόμηση της ακολουθίας $S = \{s_1, s_2, ..., s_n\}$.

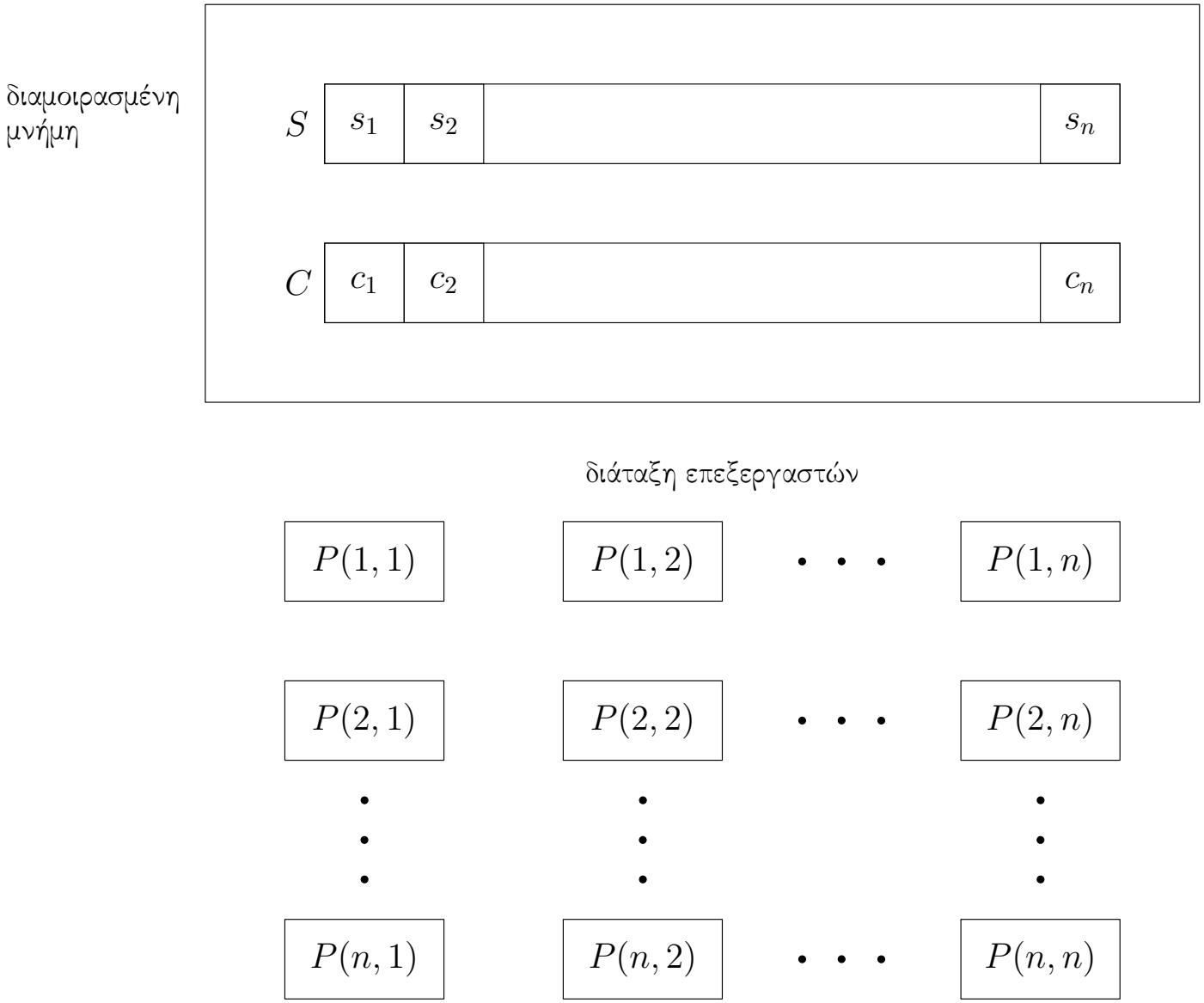
Ταξινόμηση στο CRCW Μοντέλο

Ο αλγόριθμος της ταξινόμησης βασίζεται στην ιδέα της ταξινόμησης με απαρίθμηση. Η θέση του κάθε στοιχείου s_i της S στην ταξινομημένη ακολουθία προσδιορίζεται με τον υπολογισμό του c_i που συμβολίζει το πλήθος των στοιχείων μικρότερα από αυτό. Αν $s_i = s_j$ τότε σαν μεγαλύτερο λαμβάνεται το s_i αν $i > j$, διαφορετικά λαμβάνεται το s_j.

Από τη στιγμή που έχουν υπολογιστεί όλα τα c_i, το s_i τοποθετείται στη θέση $1 + c_i$ της ταξινομημένης ακολουθίας.

Για ευκολία υποθέτουμε ότι οι επεξεργαστές αποτελούν ένα διδιάστατο πίνακα.

Ταξινόμηση στο CRCW Μοντέλο



Σχήμα: Διαμοιραζόμενη μνήμη και διάταξη επεξεργαστών στο CRCW SM SIMD μοντέλο

Ταξινόμηση στο CRCW Μοντέλο

	S	$P(1,1)$	$P(1,2)$	$P(1,3)$	$P(1,4)$	C	S
s_1	5	5,5	5,**2**	5,4	5,5	2	2

	$P(2,1)$	$P(2,2)$	$P(2,3)$	$P(2,4)$		
s_1	**2**	**2**,5	**2**, **2**	**2**,5	2,5	0	4

	$P(3,1)$	$P(3,2)$	$P(3,3)$	$P(3,4)$		
s_1	4	4,5	4,**2**	4,4	4,5	1	5

	$P(4,1)$	$P(4,2)$	$P(4,3)$	$P(4,4)$		
s_1	5	5,5	5,**2**	5,4	5,5	3	5

Σχήμα: Παράδειγμα για $S = \{5, 1, 4, 5\}$

Ταξινόμηση στο CRCW Μοντέλο

Παράδειγμα

$$S = \{5, 1, 4, 5\}$$

- κάθε στοιχείο s_i διαβάζεται ταυτόχρονα από όλους τους επεξεργαστές στην i γραμμή και i στήλη
- όλοι οι επεξεργαστές σε μια δεδομένη γραμμή γράφουν στην ίδια θέση μνήμης.

Ταξινόμηση στο CRCW Μοντέλο

Διαδικασία 4: CRCW SORT (S)

- **Βήμα 1**

για $i = 1$ **έως** n **κάνε παράλληλα**
  για $j = 1$ **έως** n **κάνε παράλληλα**
    αν $(s_i > s_j)$ ***ή*** $(s_i = s_j$ ***και*** $i > j)$ **τότε**
      ο $P(i, j)$ γράφει 1 στην c_i
    αλλιώς
      ο $P(i, j)$ γράφει 0 στην c_i
    τέλος
  τέλος
τέλος

- **Βήμα 2**

για $i = 1$ **έως** n **κάνε παράλληλα**
  ο $P(i, 1)$ αποθηκεύει το s_i στη θέση $1 + c_i$ του S
τέλος

Ταξινόμηση στο CRCW Μοντέλο

Η διαμοιραζόμενη μνήμη περιέχει δύο πίνακες:

- πίνακας $S \leftarrow$ αρχική ακολουθία
- πίνακας $C \leftarrow$ πλήθος των στοιχείων c_i που είναι μικρότερα του s_i

Η τελική ταξινομημένη ακολουθία καταχωρείται στον πίνακα S.

Η $i-$οστή σειρά των επεξεργαστών "είναι υπεύθυνη" για το στοιχείο s_i: οι επεξεργαστές $P(i,1), P(i,2), ..., P(i,n)$ υπολογίζουν το c_i και αποθηκεύουν το s_i στη θέση $c_i + 1$.

$$t(n) = O(1)!$$

$$p(n) = n^2$$

$$c(n) = c(n^2): \text{ όχι βέλτιστο κόστος}$$

Ταξινόμηση στο CREW Μοντέλο

Σκοπός μας είναι να σχεδιάσουμε έναν αλγόριθμο, ο οποίος δεν απαιτεί ταυτόχρονη γραφή και χρησιμοποιεί ένα μικρό αριθμό αριθμό επεξεργαστών.

Στη συνέχεια θα χρησιμοποιηθεί πάλι ένας αλγόριθμος συγχώνευσης για την ταξινόμηση (CREW MERGE).

Ταξινόμηση στο CREW Μοντέλο

Υποθέτουμε ότι έχουμε ένα (CREW SM SIMD) υπολογιστή με N επεξεργαστές $P_1, P_2, ..., P_N$ για την ταξινόμηση της ακολουθίας $S = s_1, s_2, ..., s_N$, όπου $N \leq n$. Τα στοιχεία της S κατανέμονται ομοιόμορφα μεταξύ των N επεξεργαστών $\left(\lceil \frac{n}{N} \rceil\right)$ στοιχεία ανά επεξεργαστή).

- Κάθε επεξεργαστής ταξινομεί την ακολουθία του με τη χρήση της QUICKSORT.
- Οι N ταξινομημένες υποακολουθίες συγχωνεύονται ανά δύο, ταυτόχρονα, με τη χρήση της CREW MERGE για κάθε ζευγάρι.
- Οι προκύπτουσες ακολουθίες συγχωνεύονται ξανά κατά ζεύγη μέχρις ότου ληφθεί μια ακολουθία με n στοιχεία.

Στη συνέχεια δίνεται ο αλγόριθμος, όπου S_j^k συμβολίζει τη συγχωνευμένη ακολουθία και P_j^k τους επεξεργαστές που έκαναν τη συγχώνευση.

Ταξινόμηση στο CREW Μοντέλο

Διαδικασία 5: CREW SORT (S)

- **Βήμα 1**

για $i = 1$ **έως** N **κάνε παράλληλα**

Ο επεξεργαστής P_i

(1.1) διαβάζει μια διακεκριμένη υποακολουθία S_i του S μεγέθους n/N

(1.2) $QUICKSORT(S_i)$

(1.3) $S_i^1 \leftarrow S_i$

(1.4) $P_i^1 \leftarrow \{P_i\}$

τέλος

Ταξινόμηση στο CREW Μοντέλο

- **Βήμα 2**

(2.1) $u \leftarrow 1$

(2.2) $v \leftarrow N$

(2.3) **όσο** $v > 1$ **κάνε**

(2.3.1) **για** $m = 1$ **έως** $\lfloor v/2 \rfloor$ **κάνε παράλληλα**

(i) $P_m^{u+1} \leftarrow P_{2m-1}^u \cup P_{2m}^u$

(ii) Οι επεξεργαστές του συνόλου P_m^{u+1} εκτελούν την CREW MERGE $(S_{2m-1}^u, S_{2m}^u, S_m^{u+1})$

τέλος

(2.3.2) **αν** v *είναι περιττό* **τότε**

(i) $P_{\lceil v/2 \rceil}^{u+1} \leftarrow P_v^u$

(ii) $S_{\lceil v/2 \rceil}^{u+1} \leftarrow S_v^u$

τέλος

(2.3.3) $u \leftarrow u + 1$

(2.3.4) $v \leftarrow \lceil v/2 \rceil$

τέλος

Ταξινόμηση στο CREW Μοντέλο

Ανάλυση

Η QUICKSORT απαιτεί $O((\frac{n}{N})\log(\frac{n}{N}))$ χρόνο.

Σε κάθε επανάληψη του βήματος 2.3 συγχωνεύονται $\lfloor \frac{v}{2} \rfloor$ ζεύγη υπακολουθιών με $\frac{n}{\lfloor \frac{v}{2} \rfloor}$ στοιχεία για κάθε ζεύγος με τη χρήση $\frac{N}{\lfloor \frac{v}{2} \rfloor}$ επεξεργαστών για κάθε ζεύγος.

Η CREW MERGE απαιτεί $O\left([(n/\lfloor v/2 \rfloor) \,/\, (N/\lfloor v/2 \rfloor)] + \log(n/\lfloor v/2 \rfloor)\right)$, δηλαδή $O(n/N + \log n)$ χρόνο.

Ταξινόμηση στο CREW Μοντέλο

Επειδή το βήμα 2.3 επαναλαμβάνεται $\lfloor \log N \rfloor$ φορές, ο συνολικός χρόνος της CREW SORT είναι

$$t(n) = \underbrace{O\left(\left(\frac{n}{N}\right)\log\left(\frac{n}{N}\right)\right)}_{QUICKSORT} + O\left(\left(\frac{n}{N}\right)\log N + \log n \log N\right) =$$

$$= O\left(\left(\frac{n}{N}\right)\log n + \log^2 n\right)$$

Επειδή $p(n) = N$, το κόστος είναι $c(n) = O(n \log n + N \log^2 n)$ το οποίο είναι βέλτιστο για $N \leq n/\log n$.

Ταξινόμηση στο CREW Μοντέλο

Παράδειγμα

Έστω $S = \{2, 8, 5, 10, 15, 1, 12, 6, 14, 3, 11, 7, 9, 4, 13, 16\}$ και $N = 4$.
Έχουμε $n = 16$ και $\frac{n}{N} = 4$, άρα στο Βήμα 1 οι επεξεργαστές P_1, P_2, P_3 και P_4 λαμβάνουν τις υπακολουθίες:

$$S_1 = \{2, 8, 5, 10\}, \quad S_2 = \{15, 1, 12, 6\}, \quad S_3 = \{14, 3, 11, 7\} \quad \text{και}$$
$$S_4 = \{9, 4, 13, 16\}$$

αντίστοιχα, τις οποίες ταξινομούν τοπικά. Στο τέλος του βήματος 1, θα έχουμε:

$$S_1^1 = \{2, 5, 8, 10\},\ S_2^1 = \{1, 6, 12, 15\},\ S_3^1 = \{3, 7, 11, 14\},\ S_4^1 = \{4, 9, 13, 16\}$$

$$P_1^1 = \{P_1\}, \quad P_2^1 = \{P_2\}, \quad P_3^1 = \{P_3\}, \quad P_4^1 = \{P_4\}$$

Ταξινόμηση στο CREW Μοντέλο

Στην πρώτη επανάληψη του βήματος 2.3 οι επεξεργαστές στο $P_1^2 = P_1^1 \cup P_2^1 = \{P_1, P_2\}$ συγχωνεύουν τα στοιχεία των S_1^1 και S_2^1 για το σχηματισμό της

$$S_1^2 = \{1, 2, 5, 6, 8, 10, 12, 15\}.$$

Ταυτόχρονα οι επεξεργαστές του $P_2^2 = P_3^1 \cup P_4^1 = \{P_3, P_4\}$ συγχωνεύουν τις S_3^1 και S_4^1 στην

$$S_2^2 = \{3, 4, 7, 9, 11, 13, 14, 16\}.$$

Στη δεύτερη επανάληψη οι επεξεργαστές στο $P_1^3 = P_1^2 \cup P_2^2 = \{P_1, P_2, P_3, P_4\}$ συγχωνεύουν τις S_1^2 και S_2^2 στην

$$S_1^3 = \{1, 2, ..., 16\}.$$

Ταξινόμηση στο EREW Μοντέλο

Υπόθεση

Έχουμε στη διάθεσή μας Ν επεξεργαστές $P_1, P_2, ..., P_N$ σε ένα EREW SM SIMD υπολογιστή για την ταξινόμηση της ακολουθίας $S = \{s_1, s_2, ..., s_n\}$, όπου $N < n$.

Προσομοίωση της CREW SORT

Ο απλούστερος τρόπος προκειμένου να αποφευχθεί το ταυτόχρονο διάβασμα στην CREW SORT είναι να χρησιμοποιηθεί η MULTIPLE BROADCAST. Η προσομοίωση αυτή της CREW SORT στο EREW μοντέλο απαιτεί

$$t(n) = O((n/N)\log n + \log n \log N) \times \underbrace{\log N}_{BROADCAST} =$$

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

με κόστος

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

το οποίο δεν είναι βέλτιστο.

Ταξινόμηση χωρίς συγκρούσεις διαβάσματος

Η αντικατάσταση της CREW MERGE με την EREW MERGE εξαλείφει το πρόβλημα της σύγκρουσης του διαβάσματος. Το βήμα αυτό απαιτεί $O((n/N) + \log n \log N)$ και επειδή υπάρχουν $\log N$ επαναλήψεις, ο συνολικός χρόνος θα είναι:

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

με κόστος

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

το οποίο είναι βέλτιστο όταν $N \leq n\log^2 n$.

Ταξινόμηση με επιλογή

Στον αλγόριθμο αυτό χρησιμοποιείται πάλι η QUICKSORT. Σημειώνουμε ότι $N < n$ και μπορούμε να γράψουμε $N = n^{1-x}$, όπου $0 < x < 1$.

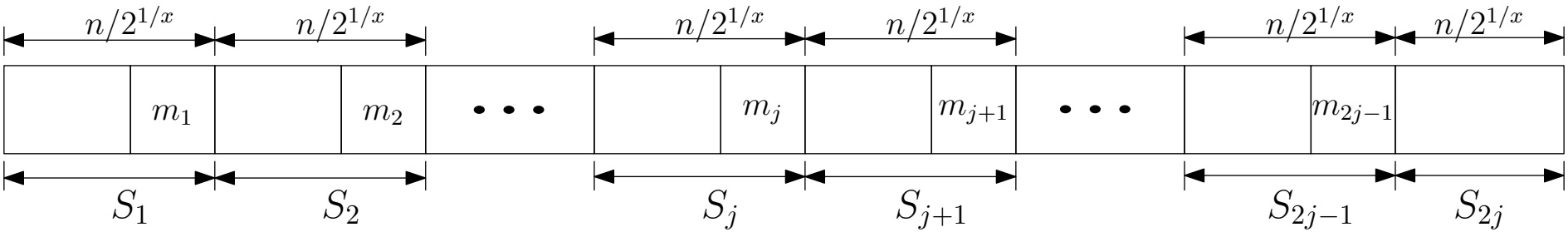
Έστω m_i συμβολίζει το $\lceil i(n/2^{1/x}) \rceil$-οστό μικρότερο στοιχείο της S για $1 \leq i \leq 2^{\frac{1}{x}} - 1$. Τα m_i διαιρούν την S σε $2^{\frac{1}{x}}$ υποακολουθίες μήκους $\frac{n}{2^{\frac{1}{x}}}$ η κάθε μία. Αυτές οι υποακολουθίες συμβολίζονται με

$$S_1, S_2, ..., S_j, S_{j+1}, S_{j+2}, ..., S_{2j}$$

όπου $j = 2^{(\frac{1}{x})-1}$ και ικανοποιούν την ιδιότητα: Κάθε στοιχείο της S_i είναι μικρότερο ή ίσο με κάθε στοιχείο της S_{i+1} για $1 \leq i \leq 2j - 1$ (βλ. σχήμα).

Η εργασία της υποδιαίρεσης μπορεί τώρα να εκτελεστεί αναδρομικά σε κάθε μία από τις υποακολουθίες S_i μέχρις ότου όλη η υποακολουθία S ταξινομηθεί.

Ταξινόμηση με επιλογή



Σχήμα: Διαίρεση ακολουθίας για ταξινόμηση με επιλογή

$2j = 2^{\frac{1}{x}}, j = 2^{\frac{1}{x}-1}$ και κάθε στοιχείο της S_i είναι μικρότερο ή ίσο από κάθε στοιχείο της S_{i+1} για $1 \leq i \leq 2j-1$.

Επίσης $N < n, N = n^{1-x}, 0 < x < 1, m_i \left\lceil i \left(\frac{n}{2^{\frac{1}{x}}} \right) \right\rceil$-οστό μικρότερο στοιχείο της $S, 1 \leq i \leq 2^{\frac{1}{x}} - 1$. Τα m_i διαιρούν την S σε $2^{\frac{1}{x}}$ υποακολουθίες μεγέθους $\frac{n}{2^{\frac{1}{x}}}$ η κάθε μία.

Ταξινόμηση με επιλογή

Ο αλγόριθμος εκτελείται παράλληλα καλώντας αρχικά την PARALLEL SELECT για τον προσδιορισμό των στοιχείων m_i και στη συνέχεια της δημιουργίας των υποακολουθιών S_i. Ο αλγόριθμος εφαρμόζεται παράλληλα στις υποακολουθίες

$$S_1, S_2, ..., S_j$$

χρησιμοποιώντας N/j επεξεργαστές για κάθε υποακολουθία. Στη συνέχεια γίνεται το ίδιο για κάθε υποακολουθία από τις

$$S_{j+1}, S_{j+2}, ..., S_{2j}$$

Ας σημειωθεί ότι ο αριθμός των επεξεργαστών για την ταξινόμηση κάθε υποακολουθίας μεγέθους $n/2^{1/x}$, είναι $n^{1-x}/2^{1/x-1}$, δηλαδή ακριβώς ίσος με όσο απαιτεί μια κανονική αναδρομή $(n/2^{1-x})^{1-x}$.

Ταξινόμηση με επιλογή

Το $2^{1/x}$ θα είναι ένας ακέραιος πεπερασμένου μεγέθους, πράγμα που εξασφαλίζει την ύπαρξη ενός ορίου στον εκτελέσιμο χρόνο καθώς και ότι όλα τα m_i υπάρχουν. Αρχικά οι N διαθέσιμοι επεξεργαστές υπολογίζουν το x από την $N = n^{1-x}$. Αν το x δεν ικανοποιεί τις συνθήκες:

$$\text{(i)} \left\lceil \frac{1}{x} \right\rceil \leq 10^{(1)} \quad \text{(π.χ.) και} \quad \text{(ii)} \; n \geq 2^{\lceil \frac{1}{x} \rceil (2)}$$

τότε ο μικρότερος πραγματικός αριθμός μεγαλύτερος του x που ικανοποιεί τα (1) και (2) λαμβάνεται ο x. Έστω $k = 2^{\lceil \frac{1}{x} \rceil}$, τότε ο αλγόριθμος δίνεται από την EREW SORT.

---

(1) Εξασφαλίζει ότι ο $2^{\frac{1}{x}}$ είναι ακέραιος πεπερασμένου μεγέθους.

(2) Εξασφαλίζει ότι τα m_i θα βρεθούν.

Ταξινόμηση με επιλογή

Διαδικασία 6: EREW SORT (S)

αν $|S| \leq k$ **τότε**
| QUICKSORT(S)
αλλιώς
| (1) **για** $i = 1$ **έως** $k - 1$ **κάνε**
| | PARALLEL SELECT($S, \lceil \frac{i|S|}{k} \rceil$) {βρίσκει το m_i}
| **τέλος**
| (2) $S_1 \leftarrow \{s \in S : s \leq m_1\}$
| (3) **για** $i = 2$ **έως** $k - 1$ **κάνε**
| | $S_i \leftarrow \{s \in S : m_{i-1} \leq s \leq m_i\}$
| **τέλος**
| (4) $S_k \leftarrow \{s \in S : s \geq m_{k-1}\}$
| (5) **για** $i = 1$ **έως** $\frac{k}{2}$ **κάνε παράλληλα**
| | EREW SORT(S_i)
| **τέλος**
| (6) **για** $i = (\frac{k}{2} + 1)$ **έως** k **κάνε παράλληλα**
| | EREW SORT(S_i)
| **τέλος**
τέλος

Ταξινόμηση με επιλογή

Σημειώνεται ότι η ακολουθία S_i (βλ.βήματα 2-4) δημιουργείται με τη μέθοδο που σχετίζεται με την PARALLEL SELECT. Στο βήμα 3 τα στοιχεία της S μικρότερα από

το m_i και μεγαλύτερα ή ίσα του m_{i-1} τοποθετουνται πρώτα στην S_i. Αν $|S_i| < \lceil \frac{|S|}{k} \rceil$, τότε στοιχεία ίσα με το m_i προστίθενται στο S_i έτσι ώστε είτε $|S_i| = \lceil \frac{|S|}{k} \rceil$ ή δεν υπάρχουν (έχουν απομείνει) για να προστεθούν στην S_i.

Τα βήματα 2 και 4 εκτελούνται ανάλογα.

Ταξινόμηση με επιλογή

Παράδειγμα

Έστω $S = \{5, 9, 12, 16, 18, 2, 10, 13, 17, 4, 7, 18, 18, 11, 3, 17, 20, 19, 14, 8, 5, 17, 1, 11, 15, 10, 6\}$.

Είναι $n = 27$ και έχουμε 5 επεξεργαστές P_1, P_2, P_3, P_4 και P_5, δηλαδή $N = 5$.

Συνεπώς $5 = 27^{1-x} \Rightarrow x \simeq 0.5, k = 2^{\lceil \frac{1}{x} \rceil} = 4$, και
$m_1 = 6, m_2 = 11$ και $m_3 = 17$.

Ταξινόμηση με επιλογή

Στο βήμα 5 εφαρμόζεται αναδρομικά ο αλγόριθμος και ταυτόχρονα στις S_1 και S_2. Επειδή $|S_1| = |S_2| = 7$ και $\lfloor 7^{1-x} \rfloor = 2$ άρα 2 επεξεργαστές χρησιμοποιούνται για την ταξινόμηση κάθε υποακολουθίας S_1 και S_2 ο πέμπτος παραμένει ανενεργός.

Για την S_1 οι επεξεργαστές P_1 και P_2 υπολογίζουν τα $m_1 = 2, m_2 = 4$ και $m_3 = 5$ και δημιουργούνται οι τέσσερεις υποακολουθίες $\{1, 2\}, \{3, 4\}, \{5, 5\}$ και $\{6\}$ κάθε μία από τις οποίες είναι ήδη ταξινομημένη.

Για την S_2, οι επεξεργαστές P_3 και P_4 υπολογίζουν τα $m_1 = 8, m_2 = 10$ και $m_3 = 11$ και τις υποακολουθίες $\{7, 8\}, \{9, 10\}, \{10, 11\}$ και $\{11\}$ που ήδη είναι ταξινομημένες.

Ταξινόμηση με επιλογή

Στο βήμα 6, ο αλγόριθμος εφαρμόζεται αναδρομικά και ταυτόχρονα στις S_3 και S_4. Επειδή $|S_3| = 7$ και $|S_4| = 6$, όλα τα m_i υπάρχουν. Αρχικά οι N διαθέσιμοι επεξεργαστές υπολογίζουν το x από την $N = n^{1-x}$. Αν το x δεν ικανοποιεί τις συνθήκες:

$$\text{(i)} \quad \left\lceil \frac{1}{x} \right\rceil \leq 10^{(3)} \quad \text{(π.χ.) και} \quad \text{(ii)} \; n \geq 2^{\lceil \frac{1}{x} \rceil (4)}$$

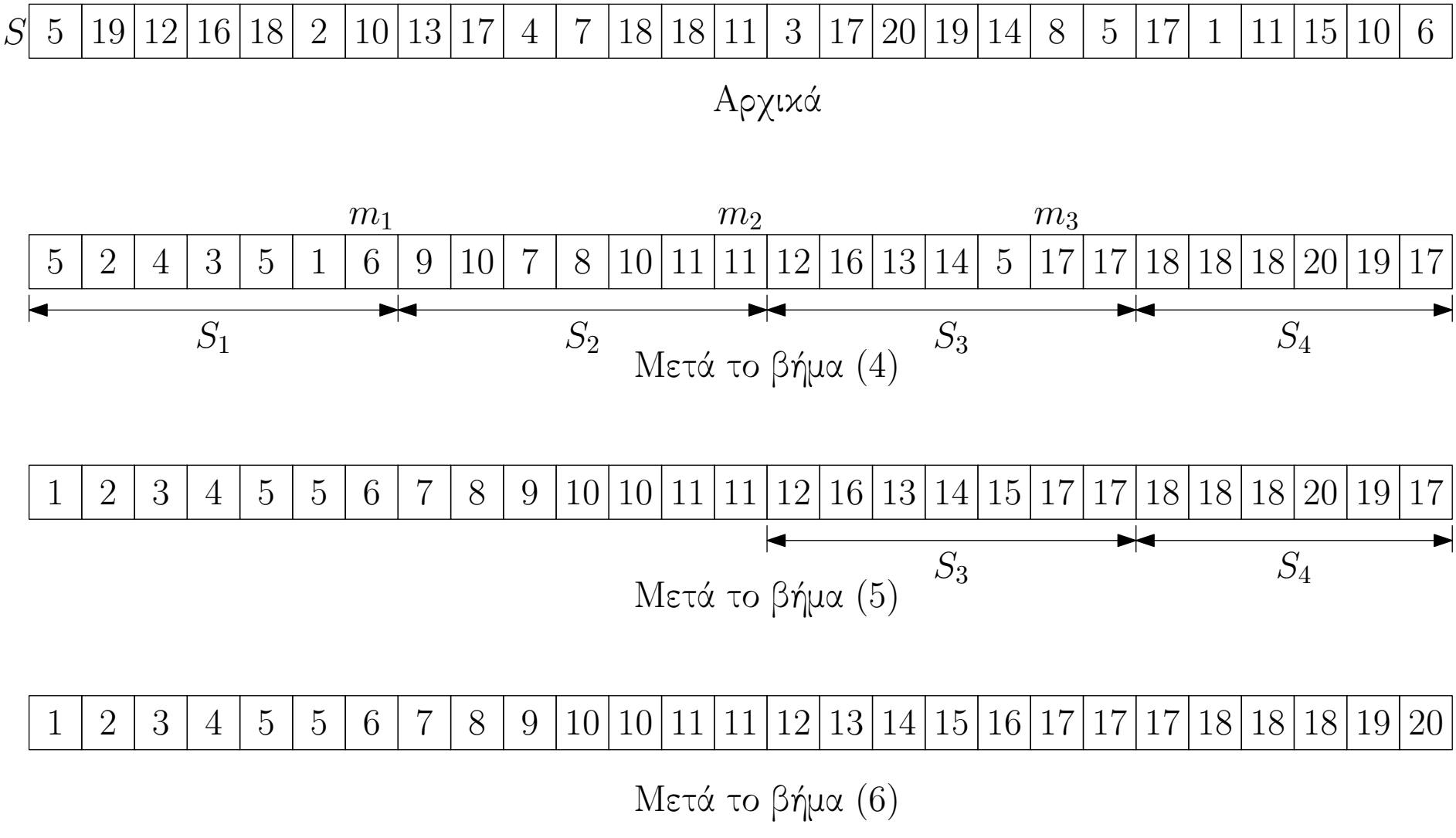
τότε ο μικρότερος πραγματικός αριθμός μεγαλύτερος του x που ικανοποιεί την (1) και (2) λαμβάνεται ο x. Έστω $k = 2^{\lceil \frac{1}{x} \rceil}$, τότε ο αλγόριθμος δίνεται από την EREW SORT.

---

(3) Εξασφαλίζει ότι ο $2^{\frac{1}{x}}$ είναι ακέραιος πεπερασμένου μεγέθους.

(4) Εξασφαλίζει ότι τα m_i θα βρεθούν.

Ταξινόμηση με επιλογή



Σχήμα: Ακολουθία S μετά το βήμα 5

Ταξινόμηση με επιλογή

Τα 7^{1-x} και 6^{1-x} στρογγυλεύονται στο 2 και έτσι δύο επεξεργαστές χρησιμοποιούνται για την ταξινόμηση κάθε μιας από τις δύο υποακολουθίες S_3 και S_4.

Για την $S_3, m_1 = 13, m_2 = 15$ και $m_3 = 17$ και δημιουργούνται οι τέσσερεις υποακολουθίες $\{12, 13\}, \{14, 15\}, \{16, 17\}$ και $\{17\}$ κάθε μία από τις οποίες είναι ήδη ταξινομημένη.

Για την S_4, έχουμε $m_1 = 18, m_2 = 18$ και $m_3 = 20$, οπότε προκύπτουν οι τέσσερεις υποακολουθίες $\{17, 18\}, \{18, 18\}, \{19, 20\}$ και μια κενή.

Η ακολουθία S μετά το βήμα 5 παρουσιάζεται στο Σχήμα 14.

Ταξινόμηση με επιλογή

Ανάλυση

Ο χρόνος για την EREW SORT είναι:

$$t(n) = \underbrace{cn^x}_{\substack{\text{PARALLEL} \\ \text{SELECT}}} + 2t(\frac{n}{k}) = O(n^x \log n)$$

Επειδή $p(n) = n^{1-x}$,

$$c(n) = p(n)t(n) = O(n \log n)$$

το οποίο είναι βέλτιστο.