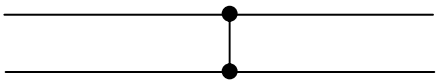
ΕΠΛ 232: Αλγόριθμοι και Πολυπλοκότητα

Κατ'οίκον Εργασία 3 – Σκελετοί Λύσεων

1. (α) Θεωρούμε ένα δίκτυο σύγκρισης με n γραμμές εισόδου το οποίο δεν περιέχει κανένα συγκριτή ανάμεσα στις εισόδους i και $i + 1$, για κάποιο $1 \leq i \leq n$. Έστω η ακολουθία $A = \langle 1, 2, 3, \ldots, i - 1, i + 1, i, i + 2, \ldots, n \rangle$ (ταξινομημένη εκτός από τις θέσεις i και $i + 1$). Μπορούμε να δούμε (απόδειξη με επαγωγή στο πλήθος των συγκριτών του δικτύου) ότι, με δεδομένο εισόδου την ακολουθία Α, το δίκτυο αποτυγχάνει να ταξινομήσει την Α.

   (β) Το ζητούμενο μπορεί να επιτευχθεί μέσω ενός τροποποιημένου ημικαθαριστή. Απόδειξη με επαγωγή στο n.

   Βασική περίπτωση - $n = 1$

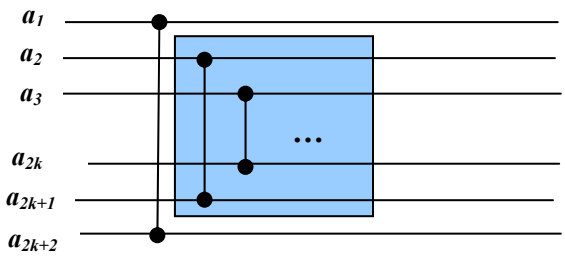
   Προφανώς το δίκτυο διαχωρίζει τους 1 μικρότερους από τους 1 μεγαλύτερους αριθμούς.

   Υπόθεση της επαγωγής: Ένας τροποποιημένος ημι-καθαριστής μεγέθους $2k$ ικανοποιεί τις προδιαγραφές του προβλήματος.

   Βήμα της επαγωγής:
   Μας δίνονται $2(k+1)$ ακέραιοι, $a_1, a_2, \ldots, a_{2k+2}$, και θέλουμε να ξεχωρίσουμε τους $k+1$ μικρότερους από τους $k+1$ μεγαλύτερους. Υπάρχουν δύο περιπτώσεις:
   - Αν $a_1 < a_{2k+2}$ τότε οι $k+1$ μικρότεροι είναι ο a_1 και οι k μικρότεροι από τους $a_2, \ldots, a_{2k+1}$.
   - Αν $a_1 > a_{2k+2}$ τότε οι $k+1$ μικρότεροι είναι ο a_{2k+2} και οι k μικρότεροι από τους $a_2, \ldots, a_{2k+1}$.

   Επομένως οι $k+1$ μικρότεροι αριθμοί είναι ο $\min(a_1, a_{2k+2})$και οι k μικρότεροι από τους $a_2, \ldots, a_{2k+1}$.

   Θεωρήστε ένα τροποποιημένο ημι-καθαριστή μεγέθους $(2k + 2)$.

Από την υπόθεση της επαγωγής ο τροποποιημένος καθαριστής μεγέθους $2k$ που φαίνεται σκιασμένος στο σχήμα διαχωρίζει το δεδομένο εισόδου a_2, a_2, …, a_{2k+1} στους k μικρότερους και στους k μεγαλύτερους αριθμούς και ο αρχικός συγκριτής ανεβάζει στην πρώτη γραμμή τον ελάχιστο ανάμεσα στους a_1 και a_{2k+2}. Κατά συνέπεια το δίκτυο πετυχαίνει το στόχο του.

2. Ο τυχαιοποιημένος Las Vegas αλγόριθμος έχει ως εξής:
   - Διάλεξε τυχαία και ομοιόμορφα δύο θέσεις του πίνακα X, έστω i και j.
   - Αν j≠i και X[i] = X[j] τότε έχεις εντοπίσει το ζητούμενο στοιχείο και μπορείς να το επιστρέψεις και να τερματίσεις.
   - Διαφορετικά επανάλαβε από την αρχή.

   Ο αλγόριθμος τερματίζει μετά από μία μόνο επανάληψη με πιθανότητα

$$\frac{1}{2}\cdot\frac{n/2-1}{n}=\frac{n-2}{4n}=\frac{1}{4}-\frac{1}{2n}>\frac{1}{5},\qquad n\ge 10$$

Πιθανότητα εύρεσης του στοιχείου με τις n/2 επαναλήψεις κατά την πρώτη τυχαιοποιημένη επιλογή στοιχείου.

Πιθανότητα εύρεσης διαφορετικής εμφάνισης του στοιχείου με τις n/2 επαναλήψεις κατά τη δεύτερη τυχαιοποιημένη επιλογή.

Κατά συνέπεια, ο αλγόριθμος τερματίζει μετά από μία μόνο επανάληψη με πιθανότητα > 1/5 για κάθε n ≥ 10, ή, ο αλγόριθμος δεν τερματίζει μετά από μία επανάληψη με πιθανότητα ≤ 4/5.

Επομένως, ο αλγόριθμος δεν τερματίζει μετά από 10 επαναλήψεις με πιθανότητα $<\left(\frac{4}{5}\right)^{10}<0.1074$, και τερματίζει μέσα σε 10 επαναλήψεις με πιθανότητα ≥ 0.8926.
Παρόμοια, ο αλγόριθμος δεν τερματίζει μετά από 100 επαναλήψεις με πιθανότητα $<\left(\frac{4}{5}\right)^{100}<2.04\cdot 10^{-10}$. Δηλαδή, σχεδόν σίγουρα ο αλγόριθμος θα τερματίσει μετά από 100 επαναλήψεις, άσχετα με το μέγεθος του στιγμιότυπου εισόδου n.

Γενικά ισχύει ότι η πιθανότητα να μην τερματίσει ο αλγόριθμος μέσα σε $c\cdot\alpha\cdot\log n$ βήματα, είναι

$$p<\left(\frac{4}{5}\right)^{c\alpha\lg n}=n^{-c\alpha\lg(5/4)}$$

Διαλέγοντας $c \geq \frac{1}{\lg(5/4)}$, έχουμε ότι $p < n^{-\alpha}$. Έτσι φτάνουμε στο συμπέρασμα ότι ο αλγόριθμος τερματίζει μέσα $c \cdot \alpha \cdot \log n$ βήματα με πιθανότητα $\geq 1 - \frac{1}{n^{\alpha}}$.

3. (α) Ο πιο κάτω παράλληλος αλγόριθμος υποθέτει μοντέλο παράλληλου υπολογισμού EREW.
Θεωρούμε την ύπαρξη n καταχωρητών M_i και θέλουμε να αντιγράψουμε την τιμή του πρώτου στους n – 1 υπόλοιπους. Υποθέτουμε ότι το n είναι δύναμη του 2.

Φάση 1: Ο επεξεργαστής 1 διαβάζει την τιμή του M_1 και την αντιγράφει στον M_2.
Φάση 2: Οι επεξεργαστές 1, 2 ταυτόχρονα διαβάζουν τις τιμές των M_1, M_2 και τις αντιγράφουν στους M_3, M_4.
Φάση 3: Οι επεξεργαστές 1 - 4 ταυτόχρονα διαβάζουν τις τιμές των M_1 - M_4 και τις αντιγράφουν στους M_5- M_8.
Φάση 4: Οι επεξεργαστές 1 - 8 ταυτόχρονα διαβάζουν τις τιμές των M_1 - M_8 και τις αντιγράφουν στους M_9 - M_{16}.
…
Φάση lg n: Οι επεξεργαστές 1 – n/2 ταυτόχρονα διαβάζουν τις τιμές των M_1 - $M_{n/2}$ και τις αντιγράφουν στους $M_{n/2+1}$ - M_n.

Σε ψευδοκώδικα:

```
for (i = 0 ; i ≤ lg n - 1; i++)
   k = 2^i;
   Processor j ∈ {1, …, 2^i} in parallel do
        M_{j+k}  := M_j
```

Ο αλγόριθμος πετυχαίνει το ζητούμενο σε χρόνο O(lg n) χρησιμοποιώντας n/2 επεξεργαστές.

(β) Αρχικά, παρατηρούμε ότι το δεδομένο εισόδου μας αποτελείται από lg n ομάδες με n/lg n καταχωρητές η κάθε μια. Ο αλγόριθμος έχει ως εξής:

Βήμα 1: Χρησιμοποιώντας τον αλγόριθμο από το μέρος (α) μπορούμε να γράψουμε την τιμή του M_1 στην πρώτη ομάδα από n/lg n καταχωρητές σε χρόνο O(lg n).

Παραμένουν lg n -1 ομάδες με n/lg n στοιχεία η κάθε μια.

Βήμα 2: Οι n/lg n επεξεργαστές συνεργάζονται για να αντιγράψουν τα στοιχεία των πρώτων n/lg n επεξεργαστών στη δεύτερη ομάδα, σε χρόνο 1.

Βήμα 3: Οι n/lg n επεξεργαστές συνεργάζονται για να αντιγράψουν τα στοιχεία των πρώτων n/lg n επεξεργαστών στην τρίτη ομάδα, σε χρόνο 1.

Βήμα lg n: Οι n/lg n επεξεργαστές συνεργάζονται για να αντιγράψουν τα στοιχεία των πρώτων n/lg n επεξεργαστών στην n/lg n ομάδα, σε χρόνο 1.

Ο χρόνος εκτέλεσης του βήματος 1, είναι της τάξης O(lg n) και κάθε ένα από τα επόμενα lg n - 1 βήματα εκτελούνται σε O(1) χρόνο. Επομένως αλγόριθμος πετυχαίνει το ζητούμενο σε χρόνο O(lg n) χρησιμοποιώντας n/ lg n επεξεργαστές.

4. Ο ζητούμενος παράλληλος αλγόριθμος αφορά παραλληλοποίηση του αλγορίθμου του Strassen για πολλαπλασιασμό πινάκων. Υποθέτουμε πως ο αλγόριθμος υπολογίζει το γινόμενο των πινάκων A και B.

   i. Μοίρασε τους πίνακες A και B σε 4 υποπίνακες.

   ii. Κάνε 7 πολλαπλασιασμούς πινάκων διαστάσεων $n/2 \times n/2$ ο καθένας (δες ανάλυση αλγορίθμου του Strassen) αναδρομικά **και παράλληλα** χρησιμοποιώντας μια ομάδα επεξεργαστών για κάθε πολλαπλασιασμό.

   iii. Υπολόγισε τον $C=A \cdot B$ χρησιμοποιώντας προσθέσεις και αφαιρέσεις διαστάσεων $n/2 \times n/2$ ο καθένας (δες ανάλυση αλγορίθμου του Strassen). Κάθε μια από τις προσθαφαιρέσεις αυτές μπορούν να εκτελεστεί σε χρόνο O(1) χρησιμοποιώντας n^2 επεξεργαστές.

   Ο χρόνος εκτέλεσης του αλγορίθμου δίνεται από την πιο κάτω αναδρομική εξίσωση:

   $T(n) = T(n/2) + 1 = T(n/4) + 2$
   $= T(n/8) + 3 = T(n/2^i) + i$
   $= T(n/2^{\log n}) + \log n = \log n + 1$
   $\in O(\log n)$

   Ο αριθμός επεξεργαστών που χρησιμοποιεί ο αλγόριθμος δίνεται από την πιο κάτω αναδρομική εξίσωση:

   $P(n) = 7P(n/2) + n^2 \in O(n^{\lg 7})$

5. Ο αλγόριθμος εντοπίζει τη ζητούμενη θέση χρησιμοποιώντας ιδέες παρόμοιες με αυτές του αλγορίθμου Δυαδικής Διερεύνησης.

```
int findturn( int X[],int low, int high){
if low == high
     return low;
int mid = (high + low)/2;
if (X[mid] < X[mid + 1])
     findturn(X, mid, h);
else
     findturn(X, low, mid);
}
```