ΕΘΝΙΚΌ ΜΕΤΣΌΒΙΟ ΠΟΛΥΤΕΧΝΕΊΟ

ΣΧΟΛΉ ΗΛΕΚΤΡΟΛΌΓΩΝ ΜΗΧΑΝΙΚΏΝ ΚΑΙ ΜΗΧΑΝΙΚΏΝ ΥΠΟΛΟΓΙΣΤΏΝ

ΑΛΓΌΡΙΘΜΟΙ & ΠΟΛΥΠΛΟΚΟΤΗΤΑ

(2020-2021)

1η Σειρά Γραπτών Ασκήσεων

Ονοματεπώνυμο:

- Χρήστος Τσούφης

Αριθμός Μητρώου:

- 03117176

<u>1η Άσκηση – Ασυμπτωματικός Συμβολισμός, Αναδρομικές Σχέσεις</u>

Α) Οι δοσμένες συναρτήσεις είναι οι εξής:

- ❖ $n * \sum_{k=0}^{n} \binom{n}{k}$ = Θ(n*2^n)
- ▪ <u>Επεξήγηση</u>:
  $n * \sum_{k=0}^{n} \binom{n}{k}$ = Θ(n*2^n) = O(n*2^n)

- ❖ $\sum_{k=1}^{n} k * 2^k$ = Θ(n*2^n)
- ▪ <u>Επεξήγηση</u>:
  $\sum_{k=0}^{n} x^k = \frac{x^{n+1}-1}{x-1} \rightarrow (\sum_{k=0}^{n} 2^k)' = \frac{(n+1)x^n(x-1)-(x^{n+1}-1)}{(x-1)^2} \rightarrow \ldots \rightarrow$
  $\sum_{k=0}^{n} k * x^k$ = (n+1)2^{n+1} – 2(2^{n+1} – 1) = O(n2^{n+1})

- ❖ $\sum_{k=1}^{n} k * 2^{-k}$ = Θ(1)
- ▪ <u>Επεξήγηση</u>:
  $\sum_{k=1}^{n} k * x^k \leq \sum_{k=1}^{\infty} k * x^k = \frac{x}{(1-x)^2}$, |x| <1 $\overset{x=\frac{1}{2}}{\Longrightarrow} \sum_{k=1}^{n} k * \frac{1}{2}^k \leq \frac{\frac{1}{2}}{(\frac{1}{2})^2}$ = O(1)

- ❖ $(\sqrt{n})^{\log_2 \log_2 (n!)}$ = O($n^{\log n}$)
- ▪ <u>Επεξήγηση</u>:
  $(\sqrt{n})^{\log_2 \log_2 (n!)}$ = O($\sqrt{n^{\log(n \log n)}}$) = O($n^{0.5 \log(n \log n)}$) = O($n^{0.5 \log n}$) = O($n^{\log n}$)

- ❖ $2^{(\log_2 \log_2 n)^4}$ = O(n)
- ▪ <u>Επεξήγηση</u>:
  Προκύπτει από πράξεις/ιδιότητες λογαρίθμων.

- ❖ $\log(n!) / (\log \log n)^5$ = Θ($n \log n / (\log \log n)^5$)
- ▪ <u>Επεξήγηση</u>:
  Από τον *τύπο του Stirling*: $\log n! = \Theta(n \log n)$
  Άρα: $\frac{\log n!}{(\log \log n)^5} = \frac{\Theta(n \log n)}{(\log \log n)^5}$

- ❖ $n * 2^{2^{2^{100}}}$ = Θ(n)
- ▪ <u>Επεξήγηση</u>:
  c*n = Θ(n)

- ❖ log($\binom{2n}{n}$) = Θ(n)
- ▪ <u>Επεξήγηση</u>:
  Από τον *τύπο του Stirling*: n! ≈ $\sqrt{2\pi n}\ (\frac{n}{e})^n$
  Οπότε: log($\binom{2n}{n}$) = log($\left(\frac{(2n)!}{2*(n!)}\right)$) = Θ(log($\left(\frac{2\sqrt{\pi n}\ (\frac{2n}{e})^{2n}}{(\sqrt{2\pi n}(\frac{n}{e})^n)^2}\right)$)) = Θ(log($\left(\frac{4^n}{\sqrt{n}}\right)$)) =
  = Θ(2n – $\log_2 \sqrt{n}$) = O(n)

Η **ταξινόμηση** είναι η εξής: $\sum_{k=1}^{n} k*2^{-k} \subset 2^{(\log_2 \log_2 n)^4} \subset \log(\binom{2n}{n}) \subset n*2^{2^{2^{100}}} \subset \log(n!)\big/(\log\log n)^5 \subset (\sqrt{n})^{\log_2 \log_2(n!)} \subset n*\sum_{k=0}^{n}\binom{n}{k} \subset \sum_{k=1}^{n} k*2^{k}$

B) Παρακάτω φαίνεται ο υπολογισμός της τάξης μεγέθους των λύσεων των αναδρομικών σχέσεων.

1. $T(n) = 6T(n/3) + n^2\log n$
   Δουλεύοντας με το Master Theorem, προκύπτει: $f(n) = n^2\log n$
   - $a*f(n/b) < f(n) \rightarrow 6*f(n/3) \leq (2/3)*f(n)$, αληθές
   - $n^{4\log_3 2} < n^2 \rightarrow 1 + \log_3 2 < 2$, αληθές

   Οπότε, $T(n) = \Theta(n^2\log n) = \Theta(f(n))$.

2. $T(n) = 9T(n/3) + n^2\log n$
   Δουλεύοντας με το Master Theorem, προκύπτει: $f(n) = n^2\log n$
   - $n^{\log_b a} = n^{\log_3 9} = n^2$
   - $d = \log_b a = 2$

   Οπότε, $T(n) = \Theta(n^2\log^2 n)$.

3. $T(n) = 11T(n/3) + n^2\log n$
   Δουλεύοντας με το Master Theorem (1η περίπτωση), προκύπτει: $f(n) = n^2\log n$
   - $\log_3 11 > 2$

   Οπότε, $T(n) = \Theta(n^{\log_3 11})$.

4. $T(n) = T(n/4) + T(n/2) + n$
   Δουλεύοντας με το Master Theorem, προκύπτει: $f(n) = \Theta(n)$
   - $1/4 + 1/2 < 1$

   Οπότε, $T(n) = \Theta(n)$.

5. $T(n) = 2T(n/4) + T(n/2) + n$
   Δουλεύοντας με το Master Theorem, προκύπτει: $f(n) = \Theta(n)$
   - $1/4 + 1/4 + 1/2 = 1$

   Οπότε, $T(n) = \Theta(n*\log n)$.

6. $T(n) = T(n^{2/3}) + \Theta(\log n)$
   Θεωρώντας τις σταθερές x, y, προκύπτει:
   $x*\log n + T(n^{2/3}) \leq T(n) \leq T(n^{2/3}) + y*\log n \rightarrow$
   $x*\sum_{i=0}^{k} \log n^{(\frac{2}{3})^i} \leq T(n) \leq y*\sum_{i=0}^{k} \log n^{(\frac{2}{3})^i} \rightarrow$
   $x*\log n*\sum_{i=0}^{k}(\frac{2}{3})^i \leq T(n) \leq y*\log n*\sum_{i=0}^{k}(\frac{2}{3})^i$
   Οπότε, $T(n) = \Theta(\log n)$.

7. $T(n) = T(n/3) + \sqrt{n}$
   Δουλεύοντας με το Master Theorem, προκύπτει: $f(n) = \sqrt{n}$
   - $0 < 1/2$
   - $n^{\log_3 1} = n^0 = 1$

   Οπότε, $T(n) = \Theta(\sqrt{n})$.

$2^η$ Άσκηση – Προθεματική Ταξινόμηση

Έστω ένας μη ταξινομημένος πίνακας A[1..n], αρχικοποιημένος σε μια μετάθεση του συνόλου {1,…,n}.

A) Περιγραφή του Αλγόριθμου:

Πρώτα, ολόκληρος ο πίνακας αποτελείται από προθέματα μη ταξινομημένα. Κάνοντας σειριακή αναζήτηση, μιας και δεν είναι ταξινομημένος, εντοπίζεται το στοιχείο με την μεγαλύτερη τιμή προθέματος, το οποίο βρίσκεται στη k-οστή θέση του πίνακα. Έπειτα, εκτελώντας προθεματική περιστροφή των στοιχείων μέχρι και των k πρώτων στοιχείων, το μεγαλύτερο αυτό στοιχείο καταλήγει στην αρχή του πίνακα. Στην συνέχεια, περιστρέφονται όλα τα στοιχεία που το πρόθεμα του πίνακα δεν έχει ακόμη ταξινομηθεί, έτσι ώστε το στοιχείο με την μεγαλύτερη τιμή να καταλήξει στο τέλος του πίνακα οπότε, μειώνεται το μέγεθος του πίνακα αυτού κατά ένα. Η διαδικασία αυτή επαναλαμβάνεται με όσα στοιχεία υπολείπονται έως ότου να γίνει η ταξινόμηση. Πρέπει να σημειωθούν και οι οριακές περιπτώσεις όπου, αφενός στην περίπτωση που το στοιχείο είναι στην αρχή του πίνακα και αφετέρου, εάν το στοιχείο βρίσκεται στην τελευταία θέση του πίνακα που δεν έχει ταξινομηθεί, δεν χρειάζεται προθεματική περιστροφή.

Αναλυτικότερα, στην χειρότερη περίπτωση ο αλγόριθμος αυτός απαιτεί 2n-3 προθεματικές περιστροφές, διότι κάθε φορά ταξινομείται ένας αριθμός και μπορεί να χρειάζονται μέχρι και δύο προθεματικές περιστροφές, εκτός από τις οριακές περιπτώσεις που αναφέρθηκαν. Επισημαίνεται επίσης ότι, το τελευταίο στοιχείο δεν απαιτεί καμία προθεματική περιστροφή, ενώ το προτελευταίο στοιχείο θα είναι σίγουρα είτε στην αρχή είτε στο τέλος του προθέματος που δεν έχει ακόμη ταξινομηθεί εφόσον αυτό θα περιέχει μόνο δύο στοιχεία.

Ο παραπάνω αλγόριθμος μπορεί να υλοποιηθεί σε ψευδογλώσσα όπως φαίνεται παρακάτω:

```
for (i=n; i>=1){
    if (A[1]<=A[i])
        i--;
    else
        rotate(A, i);
        rotate(A, i-1);
        i=n;
}
```

B) Θεωρώντας πλέον τις επιτρεπτές κινήσεις ως προσημασμένες προθεματικές περιστροφές, ο αλγόριθμος μεταβάλλεται ως εξής:

Όταν το μεγαλύτερο στοιχείο είναι στην αρχή του τμήματος του πίνακα που το πρόθεμα δεν έχει ακόμα ταξινομηθεί, θα γίνεται μια προσημασμένη προθεματική περιστροφή για το πρώτο στοιχείο μόνο στην περίπτωση που είναι θετικά προσημασμένο. Με αυτόν τον τρόπο, το στοιχείο παίρνει το κατάλληλο πρόσημο (αρνητικό) και στη συνέχεια γίνεται η προσημασμένη προθεματική περιστροφή η οποία το τοποθετεί στην τελική θέση και του επαναφέρει το πρόσημο. Πρέπει να σημειωθούν και οι οριακές περιπτώσεις όπου, αφενός στην περίπτωση που το στοιχείο στο τέλος είναι θετικό δεν χρειάζεται προθεματική περιστροφή και αφετέρου, εάν το στοιχείο βρίσκεται στην τελευταία θέση του πίνακα και είναι αρνητικό, θα γίνει η επεξεργασία του όπως και των υπολοίπων.

Αναλυτικότερα, στην χειρότερη περίπτωση ο αλγόριθμος αυτός απαιτεί (2n-3) + (n-1) = 3n-4 προσημασμένες προθεματικές περιστροφές, διότι προστίθεται, πέρα από τις προηγούμενες περιστροφές, και άλλη μία για κάθε στοιχείο λόγω του πρόσημου που παίρνει προτού γίνει η περιστροφή. Από αυτή τη μέτρηση, δεν λαμβάνεται υπόψιν το πρώτο στοιχείο αφού σίγουρα θα έχει αρνητικό πρόσημο πριν την τελική περιστροφή.

Ο παραπάνω αλγόριθμος μπορεί να υλοποιηθεί σε ψευδογλώσσα όπως φαίνεται παρακάτω:

```
for (i=n; i>=1){
    if (abs(A[1])<=abs(A[i]))
        i--;
    else
        rotate(A, 1);
        rotate(A, i);
        rotate(A, i-1);
        i=n;
}
```

Γ.1) Για να δειχθεί ότι για κάθε ενδιάμεσο πίνακα A_t, μπορεί να κατασκευαστεί ένα νέο συμβατό ζεύγος έπειτα από 2 το πολύ προσημασμένες προθεματικές περιστροφές, πρέπει να γίνει αρχικά μια διάκριση των περιπτώσεων όπου δημιουργούνται συμβατά ζεύγη.

Η πρώτη περίπτωση είναι να έχουν όλοι οι αριθμοί στον ενδιάμεσο πίνακα A_t αρνητικό πρόσημο και να υπάρχει έστω και ένας αριθμός A_t με $k_t > k_{t+1}$, οπότε ο πίνακας από $[\ldots, -A_{t+1}, \ldots, -A_t, \ldots]$ γίνεται $[-A_{t+1}, \ldots, -A_t, \ldots]$ και τελικά $[\ldots, -A_{t+1}, -A_t, \ldots]$. Η δεύτερη περίπτωση είναι να υπάρχει ένας αριθμός στον ενδιάμεσο πίνακα A_t, που να έχει θετικό πρόσημο οπότε προκύπτουν οι εξής υποπεριπτώσεις. Έστω ότι A_t ο πιο μεγάλος αριθμός με θετικό πρόσημο στη θέση k_t και A_{t+1} ο αμέσως μεγαλύτερος κατά απόλυτη τιμή στη θέση k_{t+1}. Η 1η υποπερίπτωση είναι να μην υπάρχει A_{t+1} μιας και ο A_t θα είναι ο πιο μεγάλος αριθμός, οπότε ο πίνακας από $[\ldots, A_t, \ldots]$ γίνεται $[-A_t, \ldots]$ και τελικά $[\ldots, A_t]$. Η 2η υποπερίπτωση είναι να υπάρχουν και τα δύο αλλά σε θέση τέτοια ώστε $k_t < k_{t+1}$, οπότε ο πίνακας από $[\ldots, A_t, \ldots, -A_{t+1}, \ldots]$ γίνεται $[A_{t+1}, \ldots, -A_t, \ldots]$ και τελικά $[\ldots, -A_{t+1}, -A_t, \ldots]$. Τέλος, η 3η υποπερίπτωση είναι να υπάρχουν και τα δύο αλλά σε θέση τέτοια ώστε $k_t > k_{t+1}$, οπότε ο πίνακας από $[\ldots, -A_{t+1}, \ldots, A_t, \ldots]$ γίνεται $[-A_t, \ldots, A_{t+1}, \ldots]$ και τελικά $[\ldots, A_t, A_{t+1}, \ldots]$.

Η μόνη περίπτωση που δεν δημιουργείται συμβατό ζεύγος είναι όταν όλοι οι αριθμοί είναι ταξινομημένοι σε απόλυτη τιμή από τον μικρότερο στον μεγαλύτερο.

Γ.2) Περιγραφή του Αλγόριθμου:

Με χρήση του προηγούμενου αλγόριθμου, σχηματίζονται n/2 συμβατά ζεύγη σε $2 \cdot \frac{n}{2}$ προσημασμένες προθεματικές περιστροφές. Έπειτα, αφού θεωρηθεί ότι η κάρτα με την μεγαλύτερη αξία, μπαίνει στο τέλος θετικά προσημασμένη, δημιουργείται ένα συμβατό ζεύγος. Στην συνέχεια, τα νέα αυτά ζεύγη τα μεταχειρίζεται ως μια ενιαία κάρτα, ενώ συνεχόμενες τέτοιες κάρτες δημιουργούν με την σειρά τους συμβατά ζεύγη, οπότε πλέον θα είναι n/4 συμβατά ζεύγη σε $2 \cdot \frac{n}{4}$ προσημασμένες προθεματικές περιστροφές. Αυτή η διαδικασία επαναλαμβάνεται συνεχώς μέχρι την ολοκλήρωση της ταξινόμησης. Εν τέλει, θα υπάρχουν $2 \cdot n$ προσημασμένες προθεματικές περιστροφές.

<u>3^{η} Άσκηση – Παίζοντας Χαρτιά</u>

1) Περιγραφή του Αλγόριθμου:

Αρχικά επιλέγεται ένα φύλλο από την τράπουλα. Κάθε φορά που γίνεται αυτό, εντοπίζεται η πρώτη στοίβα, με binary search, στην οποία χωράει (O(logn)) και διακρίνονται οι εξής περιπτώσεις. Δηλαδή, είτε το φύλλο δεν χωράει σε καμιά στοίβα, οπότε τότε δημιουργείται μια νέα στοίβα δεξιότερα από τις υπόλοιπες, είτε, τοποθετείται σε μια άλλη στοίβα διότι η συγκεκριμένη είχε χώρο για μεγαλύτερους αριθμούς που τώρα έχει πάψει να έχει. Επειδή για την εκτέλεσή του απαιτούνται n binary searches, η πολυπλοκότητα θα είναι O(nlogn).

2) Επεξήγηση:

Με τον παραπάνω αλγόριθμο, επιτυγχάνονται k ταξινομημένες στοίβες. Αυτό, λοιπόν, που απαιτείται ώστε να ταξινομηθεί η τράπουλα, είναι γίνουν merge, όλες οι στοίβες μεταξύ τους. Αυτό μπορεί να γίνει σωστά με χρήση του Αλγόριθμου K-way Merge ο οποίος έχει πολυπλοκότητα O(nlogk).

3) Περιγραφή Λειτουργίας:

Θα προκύψουν οι στοίβες: [3, 2, 1], [7,5], [8,6], [4].

Η μέγιστη αύξουσα υπακολουθία είναι η [2, 4, 7, 8] με μήκος 4.

4) Απόδειξη:

Η εύρεση ενός μεγαλύτερου αριθμού οδηγεί στην τοποθέτησή του σε διαφορετική ή νέα στοίβα. Επομένως, θα χρειαστούν τουλάχιστον k στοίβες διότι η μέγιστη αύξουσα υπακολουθία θα έχει k στοιχεία.

5) Επεξήγηση:

Αυτό που ισχύει είναι ότι τα κορυφαία στοιχεία των αριστερών στοιβών που δεν ανήκουν στην k-οστή στοίβα θα είναι μικρότερα από το κορυφαίο στοιχείο αυτής, οπότε γίνεται λόγος για μια αύξουσα υπακολουθία. Επομένως, η μέγιστη αύξουσα υπακολουθία θα είναι με κορυφές, και τότε θα έχει μήκος ίσο με το ελάχιστο πλήθος στοιβών. Αν πάλι, η μέγιστη αύξουσα υπακολουθία, έχει k στοιχεία, τότε η μέγιστη φθίνουσα θα έχει το πολύ n-k+1 στοιχεία.

6) Επεξήγηση:

Στην περίπτωση που υπάρχουν στοίβες πλήθους ≤ n, και μέγιστου μήκος = m, τότε θα υπάρχουν το πολύ n*m φύλλα, οπότε το μέγιστο μήκος πρέπει να είναι τουλάχιστον m+1.

Στην περίπτωση που υπάρχουν στοίβες μήκος ≤ m, τότε για n στοίβες θα υπάρχουν n*m φύλλα το μέγιστο, οπότε το μέγιστο πλήθος πρέπει να είναι τουλάχιστον n+1.

Το γνωστό θεώρημα από τα Διακριτά Μαθηματικά είναι η Αρχή του Περιστερώνα.

Ένα παράδειγμα, αποτελεί η ακολουθία: 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1. Οι στοίβες που δημιουργούνται είναι: [25, 24, 23, 22, 21], [20, 19, 18, 17, 16], [15, 14, 13, 12, 11], [10, 9, 8, 7, 6], [5, 4, 3, 2, 1].

$4^η$ Άσκηση – Γρήγορη Επιλογή στο Πεδίο της Μάχης

- $1^ο$ Σενάριο:
  Επισημαίνεται ότι, στην χειρότερη περίπτωση, η πρώτη σύγκρουση θα συμβεί όταν όλα τα σωματίδια (a & b) κινηθούν σταθερά με την ελάχιστη ταχύτητα Vmin. Αυτό είναι κάτι που πρέπει να αποφευχθεί καθώς όταν συμβεί κάτι τέτοιο, κανένα δεν θα φθάσει στο κέντρο του πεδίου μάχης στον ίδιο χρόνο. Αυτή η χρονική στιγμή θα προκύψει, όπως είναι γνωστό από την Φυσική:
  dx = u*dt → dt = $\frac{dx}{u}$ = $\frac{\frac{L}{2}}{Vmin}$. Ομοίως, στην καλύτερη περίπτωση, αν ένα σωματίδιο a & ένα σωματίδιο b κινείται σταθερά με Vmax, πάλι από την Φυσική θα προκύψει: dt = $\frac{dx}{u}$ = $\frac{\frac{L}{2}}{Vmax}$.
  Διατύπωση Αλγορίθμου:
  Αρχικά, εφαρμόζεται binary search στον χρόνο, οπότε για συγκεκριμένο t, εντοπίζονται από όλα τα σωματίδια a, b η μέγιστη τιμή και η ελάχιστη τιμή. Εάν μέγιστη τιμή < ελάχιστη τιμή, τότε δεν έχει γίνει σύγκρουση οπότε δεν απαιτείται binary search στο αριστερό μισό του χρόνου, αλλά στο δεξί. Η πολυπλοκότητα προκύπτει από τον υπολογισμό μέγιστηw & ελάχιστης τιμής όπου απαιτούνται 2n επαναλήψεις, και από το binary search, οπότε θα είναι O($n * \log\frac{L}{Vmin}$).

- $2^ο$ Σενάριο:
  Με χρήση του υπερυπολογιστή, εντοπίζεται πότε δύο σωματίδια συγκρούονται. Συνεπώς, αρκεί να επιλεχθεί ένα σωματίδιο και στη συνέχεια να κληθεί η συνάρτηση με κάθε άλλο σωματίδιο ώστε να βρεθεί με ποιο σωματίδιο X έγινε πρώτα η σύγκρουση. Η διαδικασία αυτή επαναλαμβάνεται για το X και για κάθε άλλο σωματίδιο και γίνεται σύγκριση του χρόνου. Εάν ο ελάχιστος χρόνος που υπολογίζεται είναι ίδιος με αυτόν του προηγούμενου σωματιδίου, τότε έχει βρεθεί η ζητούμενη σύγκρουση. Η πολυπλοκότητα εδώ, στην χειρότερη περίπτωση θα είναι O(n^2), και στην καλύτερη O(n).

$5^η$ Άσκηση – Ερωτήματα Ανήκειν

Μια τέτοια δομή δεδομένων ονομάζεται bloom filter και είναι πιθανοτική.

$6^η$ Άσκηση – Προθέματα Συμβολοσειρών

Απαιτείται η αξιοποίηση των δομών δεδομένων tries.