Αλγόριθμοι και Πολυπλοκότητα
1η σειρά γραπτών ασκήσεων

Νικόλαος Παγώνας, el18175

Άσκηση 1: Αναδρομικές Σχέσεις

1.

Δεν εμπίπτει στο Master Theorem, αφού

$$\frac{n^2 \log n}{n^{\log_2 4}} = \log n$$

Όμως λόγω του παραπάνω, υποπτευόμαστε $\Theta(n^2 \log^2 n)$.
Επειδή $\alpha = 4$ και $b = 2$, έχουμε $\Theta(n^{\log_2 4}) = \Theta(n^2)$ φύλλα. Επίσης το κόστος του i-οστού επιπέδου είναι:

$$4^i \cdot c \cdot \left(\frac{n}{2^i}\right)^2 \cdot \log \frac{n}{2^i} = cn^2 \log \frac{n}{2^i},\ i = 0, ..., \log n$$

Επομένως το συνολικό κόστος όλων των επιπέδων είναι:

$$\begin{aligned}
\sum_{i=0}^{\log n} cn^2 \log \frac{n}{2^i} &= cn^2 \left(\sum_{i=0}^{\log n} \log n - \sum_{i=0}^{\log n} \log 2^i \right) \\
&= cn^2 \left(\log^2 n - \sum_{i=0}^{\log n} i \log 2 \right) \\
&= cn^2 \left(\log^2 n - \log 2 \cdot \frac{\log n \, (\log n + 1)}{2} \right) \\
&= \Theta(n^2 \log^2 n)
\end{aligned}$$

Άρα τελικά έχουμε όντως $T(n) = \Theta(n^2 \log^2 n)$

2.

Εμπίπτει στην περίπτωση του Master Theorem:

$$n^2 \log n = O(n^{\log_2 5 - \varepsilon})$$

άρα

$$T(n) = \Theta(n^{\log_2 5})$$

3.

Εμπίπτει στην ειδική περίπτωση του Master Theorem (”Διαίρει και βασίλευε”, διαφάνεια 15):

$$\mathrm{T}(n) = T(\gamma_1 n) + T(\gamma_2 n) + \Theta(n),\ \gamma_1 + \gamma_2 < 1 - \varepsilon$$

για $\gamma_1 = 1/4,\ \gamma_2 = 1/2,$ και οποιοδήποτε ε < 1/4

$$T(n) = \Theta(n).$$

4.

Επειδή $\frac{2n}{4} + \frac{n}{2} = n$, υποπτευόμαστε $T(n) = \Theta(n \log n)$.
Επειδή δεν ”πετάμε” στοιχεία μεταξύ επιπέδων, το δέντρο θα έχει $\Theta(n)$ φύλλα. Μένει να ελέγξουμε το κόστος που συνεισφέρουν τα ενδιάμεσα επίπεδα. Θα δείξουμε ότι αυτό το κόστος είναι $\Theta(n \log n)$.

Το ύψος του δέντρου θα είναι τουλάχιστον $\log_4 n$, λόγω της μεριάς $T(n) \to 2T(n/4)$, και το πολύ $\log_2 n$, λόγω της μεριάς $T(n) \to T(n/2)$.

Επίσης παρατηρούμε ότι κάθε επίπεδο έχει σταθερό κόστος cn (στην πραγματικότητα, μετά από ένα συγκεκριμένο βάθος θα έχουμε κόστος μικρότερο του cn, γιατί η μεριά του $T(n) \to 2T(n/4)$ θα φτάσει γρηγορότερα στα φύλλα, όμως αυτό δεν αλλάζει το αποτέλεσμα).

Αυτό σημαίνει ότι το κόστος είναι σίγουρα $\Omega(n \log n)$ και (με βάση την αμέσως παραπάνω παρατήρηση) θα είναι και $O(n \log n)$.

Έτσι τελικά έχουμε $T(n) = \Theta(n \log n)$.

5.

Έχουμε μόνο ένα φύλλο αυτή τη φορά (το δέντρο αναδρομής είναι στην ουσία γραμμική λίστα).
Το κόστος ανά επίπεδο είναι

$$c \log n^{1/2^i} = c \frac{1}{2^i} \log n$$

Και άρα το συνολικό κόστος όλων των επιπέδων είναι

$$\sum c \log n \frac{1}{2^i} = \Theta(\log n).$$

Το κόστος των φύλλων είναι $\Theta(1)$ άρα τελικά $T(n) = \Theta(\log n)$.

6.

Εμπίπτει στην περίπτωση του Master Theorem:

$$\sqrt{n} = \Omega(n^{\log_4 1+\varepsilon})$$

άρα

$$T(n) = \Theta(\sqrt{n}).$$

Άσκηση 2: Προθεματική Ταξινόμηση

(α)

Ένας αλγόριθμος είναι ο εξής:

- Βρίσκουμε το max στοιχείο του Α. Έστω ότι βρίσκεται στην θέση m.
- Κάνοντας μία προθεματική περιστροφή από το 1 μέχρι το m, φέρνουμε το max στην αρχή του πίνακα.
- Αν ξανακάνουμε μία προθεματική περιστροφή από το 1 μέχρι το τέλος του πίνακα, καταφέρνουμε να φέρουμε το max στοιχείο στην τελευταία θέση.
- Επαναλαμβάνουμε την διαδικασία, αλλά για τις θέσεις $1, ..., n-1$, μετά για τις θέσεις $1, ..., n-2$ κ.ο.κ. Έτσι φέρνουμε το 2ο μεγαλύτερο στοιχείο στην προτελευταία θέση του πίνακα, το 3ο μεγαλύτερο στην παραπροτελευταία κ.ο.κ.

Στην χειρότερη περίπτωση θα χρειαστεί να κάνουμε 2 προθεματικές περιστροφές για κάθε στοιχείο, άρα συνολικά $2n$ περιστροφές.

(β)

Εφαρμόζουμε τον αλγόριθμο του (α), με την εξής διαφορά:

Έστω ότι το max βρίσκεται στην θέση m. Αφού φέρουμε το max στην αρχή του πίνακα με μία περιστροφή, τα στοιχεία στις θέσεις $1, ..., m$ (ομάδα 1) θα έχουν γίνει αρνητικά, ενώ τα υπόλοιπα (ομάδα 2) θα παραμείνουν θετικά. Στη συνέχεια, όπως και πριν, με μία περιστροφή που περιλαμβάνει ολόκληρο τον πίνακα φέρνουμε το max στο τέλος του πίνακα. Αυτό σημαίνει ότι τώρα τα στοιχεία της ομάδας 1 θα έχουν γίνει θετικά, και θα έχουν πάει στο δεξί μέρος του πίνακα, ενώ τα στοιχεία της ομάδας 2 θα έχουν γίνει αρνητικά και θα έχουν πάει στο αριστερό μέρος του πίνακα. Κάνοντας μία επιπλέον περιστροφή στα στοιχεία της ομάδας 2, αλλάζουμε το πρόσημο τους και πλέον όλα τα στοιχεία είναι θετικά. Επαναλαμβάνουμε την διαδικασία, αλλά για τις θέσεις $1, ..., n-1$, μετά για τις θέσεις $1, ..., n-2$ κ.ο.κ. Έχουμε εισάγει μία επιπλέον προθεματική περιστροφή ανά στοιχείο σε σχέση με πριν, άρα συνολικά έχουμε $3n$ περιστροφές στην χειρότερη περίπτωση.

(γ)

1.

Έχουμε δύο περιπτώσεις:
Περίπτωση 1: Υπάρχει θετικό στοιχείο στον πίνακα. Τότε:

- Βρίσκουμε το μέγιστο στοιχείο του πίνακα που είναι θετικό, έστω m.
- Αν $m = n$, τότε όπως στο ερώτημα (α) μπορούμε να φέρουμε το m στην τελευταία θέση με θετικό πρόσημο και να έχουμε φτιάξει "καταχρηστικά" ένα συμβατό ζεύγος.
- Αν $m < n$, τότε στον πίνακα θα υπάρχει το στοιχείο $-(m+1)$ (που θα έχει αρνητικό πρόσημο). Διακρίνουμε 2 υποπεριπτώσεις:
  - Αν το $-(m+1)$ βρίσκεται αριστερά του m, τότε κάνουμε μία περιστροφή μέχρι και το m (ο πίνακας γίνεται $[-m, ..., m+1, ...]$), και μία περιστροφή μέχρι το $m+1$ αλλά χωρίς το $m+1$ (ο πίνακας γίνεται $[..., m, m+1, ...]$), οπότε φτιάξαμε ένα συμβατό ζεύγος.
  - Αν το $-(m+1)$ βρίσκεται δεξιά του m, τότε κάνουμε μία περιστροφή μέχρι και το $-(m+1)$ (ο πίνακας γίνεται $[m+1, ..., -m, ...]$), και μία περιστροφή μέχρι το $-m$ αλλά χωρίς το $-m$ (ο πίνακας γίνεται $[..., -(m+1), -m, ...]$), οπότε φτιάξαμε πάλι ένα συμβατό ζεύγος.

Περίπτωση 2: Δεν υπάρχει θετικό στοιχείο στον πίνακα. Τότε:

- Επειδή ο πίνακας δεν είναι ο $[-1, -2, ..., -n]$ (από εκφώνηση), σίγουρα θα υπάρχει κάποιο στοιχείο $-d$ που να έχει το $-(d+1)$ στα αριστερά του (δηλαδή ο πίνακας είναι $[..., -(d+1), ..., -d, ...]$).
- Αυτό σημαίνει ότι με μία περιστροφή μέχρι και το $-(d+1)$ (ο πίνακας γίνεται $[d+1, ..., -d, ...]$) και μία περιστροφή μέχρι το $-d$, αλλά χωρίς το $-d$ (ο πίνακας γίνεται $[..., -(d+1), -d, ...]$), φτιάχνουμε ένα συμβατό ζεύγος.

2.

Ο αλγόριθμος έχει ως εξής:

1. Αν ήδη έχουμε συμβατά ζεύγη τότε τα απαλείφουμε αντικαθιστώντας κάθε ζεύγος με ένα στοιχείο (διατηρώντας όμως το πρόσημο του ζεύγους που απαλείψαμε). Έτσι έχουμε το ίδιο πρόβλημα αλλά με λιγότερα στοιχεία πλέον, έστω n'. Όταν δεν έχουμε άλλα στοιχεία, ο πίνακας έχει ταξινομηθεί.
2. Αν συναντήσουμε τον πίνακα $[-1, -2, ..., -n']$, τότε:
   - Κάνουμε μία περιστροφή σε όλα τα στοιχεία
   - Κάνουμε μία περιστροφή σε όλα τα στοιχεία εκτός από το τελευταίο.

   Εύκολα φαίνεται ότι αν επαναλάβουμε τα δύο παραπάνω βήματα i φορές, ο πίνακας που προκύπτει έχει i πρώτα στοιχεία ταξινομημένα στις i τελευταίες θέσεις του πίνακα. Επομένως αν επαναλάβουμε τα δύο παραπάνω βήματα n' φορές, τότε ο πίνακας θα ταξινομηθεί πλήρως.

3. Αν δεν έχουμε κάποια από τις δύο παραπάνω περιπτώσεις, τότε φτιάχνουμε ένα συμβατό ζεύγος όπως έχουμε ήδη δείξει, και πάμε στο βήμα 1.

Έστω ότι ο αλγόριθμός μας χρειάζεται l επαναλήψεις. Σε κάθε επανάληψη κάνουμε το πολύ 2 περιστροφές (άρα $2l$ περιστροφές σύνολο), και κάθε επανάληψη μειώνει τα ζεύγη που μας μένουν κατά 1 (τουλάχιστον). Ίσως στην τελευταία επανάληψη χρειαστεί να τρέξουμε τον αλγόριθμο της περίπτωσης $[-1, -2, ..., -n']$, οπότε θα κάνουμε $2n'$ περιστροφές. Όμως $n' \leq n - l$,αφού θα έχουμε απαλείψει l ζεύγη μέχρι τότε. Τελικά οι περιστροφές που θα χρειαστούν θα είναι το πολύ $2l + 2n' \leq 2l + 2(n - l) = 2n$.

Ασκηση 3: Υπολογισμός Κυρίαρχων Θέσεων

Ο αλγόριθμος χρησιμοποιεί μια στοίβα από ζεύγη της μορφής $(A[i], i)$. Αρχικά κάνουμε push στην στοίβα το ζεύγος $(\mathrm{A}[0] = \infty, 0)$. Ύστερα, για κάθε $i = 1, ..., n$:

- **Περίπτωση 1:** Αν $A[i] \geq A[top]$ (όπου $A[top]$ το στοιχείο που βρίσκεται στην κορυφή της στοίβας), τότε κάνουμε pop το ζεύγος $(\mathrm{A}[top], top)$ από την στοίβα και ξανασυγκρίνουμε το $A[i]$ με το επόμενο στοιχείο της στοίβας.
- **Περίπτωση 2:** Αν $\mathrm{A}[i] < A[top]$, τότε καταγράφουμε ότι η θέση top κυριαρχεί της θέσης i. Κάνουμε push το ζεύγος $(A[i], i)$ στην στοίβα. Συνεχίζουμε τον αλγόριθμο για $i' = i + 1$.

(**Σημείωση:** Η στοίβα δεν θα αδειάσει ποτέ γιατί έχουμε $A[0] = \infty$).

Η βασική ιδέα του αλγορίθμου είναι ότι κάθε στοιχείο που μπαίνει στην στοίβα είναι και ένας "υποψήφιος κυρίαρχος" των στοιχείων που έπονται.

- Αν βρεθεί στοιχείο του πίνακα (έστω $A[i]$) που είναι μεγαλύτερο ή ίσο από το $A[top]$, αυτό σημαίνει ότι το $A[i]$ είναι "καλύτερο" από το $A[top]$ όσον αφορά τις επόμενες θέσεις του πίνακα, αφού έχει: **1.** μεγαλύτερη τιμή **2.** μεγαλύτερο index. Οπότε κάθε φορά πετάμε το κορυφαίο στοιχείο της στοίβας μέχρι αυτό να γίνει μεγαλύτερο από το $A[i]$.
- Αν $A[i] < A[top]$, τότε το $A[i]$ δεν είναι κατ' ανάγκη "καλύτερο", αφού μπορεί το $A[i]$ να μην είναι αρκετά μεγάλο ώστε να κυριαρχήσει σε κάποιο επόμενο στοιχείο, αλλά το $A[top]$ να είναι.

Η εγγύηση που προσφέρει ο αλγόριθμός μας είναι ότι, καθώς διασχίζουμε τον πίνακα, το πρώτο στοιχείο της στοίβας που θα είναι μεγαλύτερο από το $A[i]$ (το στοιχείο που εξετάζουμε) θα είναι και το κοντινότερο με αυτή την ιδιότητα, και άρα θα κυριαρχεί της θέσης i.

Η πολυπλοκότητα του αλγορίθμου καθορίζεται από τις εισαγωγές/εξαγωγές στην στοίβα και τις συγκρίσεις που κάνουμε.

Κάθε στοιχείο θα καταλήξει κάποια στιγμή στην Περίπτωση 2 (αφού $\mathrm{A}[0] = \infty$), οπότε θα μπει ακριβώς μία φορά στην στοίβα, και θα βγει το πολύ μία φορά από την στοίβα (αν πέσει στην

Περίπτωση 1). Επομένως οι εισαγωγές/εξαγωγές της στοίβας έχουν κόστος $\Theta(n)$.

Όσον αφορά τις συγκρίσεις, κάθε στοιχείο θα συγκριθεί ακριβώς μία φορά με ένα μεγαλύτερό του (αυτό θα γίνει την στιγμή που μπαίνει στην στοίβα, Περίπτωση 2), ενώ μπορεί να συγκριθεί πολλές φορές με μικρότερά του (Περίπτωση 1). Συνολικά όμως, οι συγκρίσεις με μικρότερα στοιχεία μπορούν να είναι το πολύ n, αφού συνολικά μπαίνουν n στοιχεία στην στοίβα, και κάθε φορά που έχουμε μια τέτοια σύγκριση βγάζουμε ένα στοιχείο από την στοίβα. Άρα και το κόστος των συγκρίσεων είναι κι αυτό $\Theta(n)$

Επομένως η συνολική πολυπλοκότητα του αλγορίθμου είναι $\Theta(n)$.

Άσκηση 4: Φόρτιση Ηλεκτρικών Αυτοκινήτων

Έστω $A = [a_1, a_2, ..., a_n]$. Στην χειρότερη περίπτωση χρειαζόμαστε n φορτιστές. Για να βρούμε την ελάχιστη τιμή του s^*, θα χρησιμοποιήσουμε δυαδική αναζήτηση στο διάστημα $[1, ..., n]$. Αν το υποψήφιο πλήθος φορτιστών επαρκεί, τότε θα συνεχίσουμε αναδρομικά την αναζήτηση στο αριστερό μισό του πίνακα, για να δούμε μήπως υπάρχει μικρότερη τιμή που επίσης επαρκεί. Αλλιώς, η υποψήφια τιμή δεν επαρκεί και συνεχίζουμε την αναζήτηση στο δεξί μισό του πίνακα, για να δούμε μήπως υπάρχει μεγαλύτερη τιμή που επαρκεί. Στο τέλος της αναζήτησης, κρατάμε το ελάχιστο επαρκές s^*.

Για δεδομένη υποψήφια τιμή του s^*, κάνουμε τα εξής:

- Δημιουργούμε ένα ιστόγραμμα των αφίξεων, δηλαδή έναν πίνακα της μορφής:

  $$[(t_1, k_1), (t_2, k_2), ..., (t_m, k_m)]$$

  όπου k_i είναι ο αριθμός των αμαξιών που ήρθαν την στιγμή t_i. Να σημειωθεί ότι στον πίνακα δεν θα βάζουμε ζεύγη (t_i, k_i) με $k_i = 0$.

- Κρατάμε σε μια μεταβλητή c τον συνολικό αριθμό αμαξιών που βρίσκονται στην ουρά την δεδομένη χρονική στιγμή t που μελετάμε, με αρχική τιμή k_1. Ξεκινάμε από την στιγμή $t = 1$ και για κάθε t κάνουμε $c = c + k_i - s^*$, δηλαδή αυξάνουμε το c ανάλογα με το πόσα αμάξια ήρθαν και το μειώνουμε ανάλογα με το πόσα αμάξια εξυπηρετήθηκαν εκείνη την χρονική στιγμή. Αν προκύψει $c < 0$, τότε θέτουμε $c = 0$ (δεν μπορούμε να έχουμε αρνητικό αριθμό αμαξιών να περιμένουν στην ουρά). Επίσης αν το t δεν αντιστοιχεί σε κάποιο k στο ιστόγραμμα, τότε θεωρούμε $k = 0$. Επαναλαμβάνουμε για $t = t + 1$.

- Κάθε φορά ελέγχουμε αν $c > s^*d$. Αν ισχύει η συνθήκη αυτή, τότε το s^* δεν είναι επαρκές, αφού σε κάθε βήμα εξυπηρετούνται s^* αμάξια, και μετά από d χρονικές μονάδες θα έχει μείνει σίγουρα τουλάχιστον ένα αμάξι να περιμένει παραπάνω από d.

- Αντίθετα αν το t φτάσει μέχρι και τον τελευταίο χρόνο t_m χωρίς ισχύσει ποτέ $c > s^*d$, τότε το s^* είναι επαρκές.

Μία παραλλαγή που μπορούμε να κάνουμε στο παραπάνω αφορά τις περιοχές όπου έχουμε πολλές διαδοχικές στιγμές με 0 αφίξεις. Σε περιπτώσεις όπου τα t διαδοχικών στοιχείων του ιστογράμματος δεν διαφέρουν κατά 1, η πράξη $c = c + k_i - s^*$, αντικαθίσταται από την $c = c - zs^*$, όπου z είναι ο αριθμός των διαδοχικών στιγμών με 0 αφίξεις. Το z βρίσκεται εύκολα κάθε φορά από το ιστόγραμμα ($z = b - a - 1$, όπου a, b οι τιμές του t για διαδοχικά στοιχεία του ιστογράμματος).

Πολυπλοκότητα:

- Η δημιουργία του ιστογράμματος απαιτεί την διάσχιση του Α μία φορά και έχει κόστος $\Theta(n)$.
- Αν δεν ακολουθήσουμε ειδική στρατηγική για τις διαδοχικές στιγμές με 0 αφίξεις, τότε ο αλγόριθμος για μια συγκεκριμένη τιμή του s^* έχει κόστος $\Theta(T)$. Με την βελτιστοποίηση όμως, το κόστος γίνεται $O(n)$ (γιατί το ιστόγραμμα στην χειρότερη περίπτωση μπορεί να έχει n στοιχεία -κάθε αμάξι σε διαφορετική χρονική στιγμή-). Αν $T \gg n$, τότε η παραλλαγή είναι προτιμότερη.
- Το κόστος της δυαδικής αναζήτησης είναι $\mathrm{O}(\log n)$.

Τελικά η πολυπλοκότητα του αλγορίθμου είναι $O(n \log n)$.

Σημείωση: Στην πραγματικότητα, στην χειρότερη περίπτωση χρειαζόμαστε τόσους φορτιστές όσο και το μέγιστο k_i του ιστογράμματος, οπότε η πολυπλοκότητα μπορεί να μειωθεί σε

$$O(n \log \max_{1 \le i \le m} k_i)$$

Άσκηση 5: Επιλογή

Σημείωση: Το k-οστό μικρότερο στοιχείο του S μπορεί να οριστεί ως ο θετικός ακέραιος q για τον οποίο ισχύει $F_S(q-1) < k$ και $F_S(q) \ge k$, δηλαδή ο ελάχιστος θετικός ακέραιος που είναι μεγαλύτερος ή ίσος από τουλάχιστον k στοιχεία.

(α)

Θα χρησιμοποιήσουμε δυαδική αναζήτηση. Συγκεκριμένα, θα ρωτήσουμε την F_S πόσα στοιχεία είναι μικρότερα ή ίσα από $M/2$. Αν $F_S(M/2) > k$, τότε συνεχίζουμε αναδρομικά την αναζήτηση στο διάστημα $[0, \mathrm{M}/2]$, για να δούμε πού θα έχουμε $F_S(x) < k$. Αν $F_S(M/2) < k$, τότε συνεχίζουμε αναδρομικά την αναζήτηση στο διάστημα $[M/2, M]$, για να δούμε πού θα έχουμε $F_S(x) \ge k$. Σταματάμε όταν βρούμε q για το οποίο ισχύει $F_S(q-1) < k$ και $F_S(q) \ge k$, όπως αναφέραμε παραπάνω. Συνολικά θα έχουμε $\Theta(\log M)$ κλήσεις της F_S στην χειρότερη περίπτωση.

(β)

Αρχικά πρέπει να ταξινομήσουμε τον πίνακα Α (σε αύξουσα σειρά). Έστω $(a_1, a_2, ..., a_n)$ η ταξινόμηση αυτή.
Ύστερα, μπορούμε να υλοποιήσουμε την κλήση της $F_S(l)$ αποδοτικά ως εξής:

- Υπολογίζουμε τις διαφορές που περιλαμβάνουν το ελάχιστο στοιχείο a_1, δηλαδή $a_2 - a_1, a_3 - a_1, ...$ και σταματάμε όταν βρούμε μία διαφορά η οποία να είναι μεγαλύτερη του l. Έστω ότι σταματήσαμε στο a_s.
- Παρατηρούμε ότι δεν χρειάζεται να επαναλάβουμε ολόκληρη την διαδικασία για τις διαφορές που περιλαμβάνουν το a_2, αφού όλες οι διαφορές της μορφής $a_i - a_2$, $1 \leq i \leq n$ θα είναι μικρότερες του l, επειδή ο πίνακας είναι σε αύξουσα σειρά, οπότε μπορούμε να τις προσμετρήσουμε κατευθείαν, χωρίς να κάνουμε τους υπολογισμούς. Θα αρχίσουμε να υπολογίζουμε διαφορές από το a_{s+1} και έπειτα.
- Το παραπάνω ισχύει για κάθε στοιχείο του πίνακα, και επομένως χρειάζεται μόνο ένα πέρασμα του πίνακα για την κλήση της $F_S(l)$.

Η συνολική πολυπλοκότητα του αλγορίθμου έχει ως εξής:

- $\Theta(n \log n)$ για την ταξινόμηση του πίνακα A.
- $\Theta(n)$ για κάθε κλήση της F_S.
- $\Theta(\log(a_{max} - a_{min}))$ κλήσεις της F_S.

Επομένως τελικά η πολυπλοκότητα του αλγορίθμου είναι $\Theta\left(n \log\left(\max\left\{n, a_{max} - a_{min}\right\}\right)\right)$.