Αλγόριθμοι και Πολυπλοκότητα

1η Σειρά Γραπτών Ασκήσεων

7ο Εξάμηνο, Ακαδημαϊκό Έτος 2021 – 2022

Ονοματεπώνυμο	Αριθμός Μητρώου
Στεφανάκης Γεώργιος	el18436

Άσκηση 1η – Αναδρομικές Σχέσεις

1.

$$T(n) = 4\,T(n/2) + \Theta(n^2 \log n)$$

Είναι της μορφής $T(n) = a\,T(n/b) + f(n)$ με παραμέτρους $a = 4, b = 2, f(n) = \Theta(n^2 \log n)$

$$n^{\log_b a} = n^{\log_2 4} = n^2$$
$$\frac{n^2 \log n}{n^{\log_b a}} = \log n \neq n^{\varepsilon}, \varepsilon > 0$$

Άρα πρέπει να κάνουμε το δέντρο αναδρομής. Σε κάθε επόμενο επίπεδο του δέντρου το πρόβλημα χωρίζεται σε 4 υποπροβλήματα με μέγεθος όσο το $\frac{1}{2}$ του προηγούμενου. Το βάθος του δένδρου είναι $\log_b n = \log n$ και τα φύλλα του δέντρου είναι $n^{\log_b a} = n^2$. Σε κάθε επίπεδο ο χρόνος εκτέλεσης είναι $4^i\, f\left(n/2^i\right) = n^2 \log \frac{n}{2^i}$. Συνεπώς για το $T(n)$ προκύπτει το άθροισμα:

$$T(n) = \sum_{i=0}^{\log n - 1} n^2 \log \frac{n}{2^i} + \Theta(n^2) = \Theta(n^2) + n^2 \sum_{i=0}^{\log n - 1} (\log n - i \log 2) = \Theta(n^2) + n^2 \log^2 n - n^2 \frac{(\log n - 1)(\log n)}{2}$$
$$= \Theta(n^2 \log^2 n)$$

2.

$$T(n) = 5\,T(n/2) + \Theta(n^2 \log n)$$

Είναι της μορφής $T(n) = a\,T(n/b) + f(n)$ με παραμέτρους $a = 5, b = 2, f(n) = \Theta(n^2 \log n)$

$$n^{\log_b a} = n^{\log_2 5} = n^{\log 5}$$
$$\frac{n^2 \log n}{n^{\log_b a}} = \frac{n^2 \log n}{n^{\log 5}}$$

Οι δύο όροι είναι πολυωνυμικά διαχωρίσιμοι και μάλιστα επικρατούν τα φύλλα. Συνεπώς $T(n) = \Theta(n^{\log 5})$.

3.

$$T(n) = T(n/4) + T(n/2) + \Theta(n)$$

Είναι της μορφής $T(n) = T(\gamma_1 n) + T(\gamma_2 n) + \Theta(n)$ με $\gamma_1 = \frac{1}{4}, \gamma_2 = \frac{1}{2}$ και άρα $\gamma_1 + \gamma_2 < 1$. Αυτό σημαίνει ότι όσο κατεβαίνουμε στο δέντρο αναδρομής οι κόμβοι για τους οποίους έχουμε να επιλύσουμε το πρόβλημα μειώνονται με αποτέλεσμα το δέντρο να «στενεύει». Το υψηλότερο επίπεδο της αναδρομής συνεισφέρει κόστος $\Theta(n)$, συνεπώς γνωρίζουμε για το κάτω φράγμα ότι $T(n) = \Omega(n)$. Εάν τώρα ακολουθήσουμε την μακρύτερη διαδρομή του δέντρου

λαμβάνοντας υπόψιν μόνο τις ακμές που κατεβαίνουμε κατά $n/2$, τότε θα φτάσουμε στα φύλλα όταν $\frac{n}{2^k} = 1 \Rightarrow k = \log n$ (ύψος του δέντρου). Κάθε ένα από τα επίπεδα $0, 1, \dots, k$ συνεισφέρει κόστος το πολύ $n\ (3/4)^i$ και μάλιστα, όταν τερματίσουν οι κόμβοι από την πλευρά του $T(n/4)$, ο χρόνος εκτέλεσης κάθε επιπέδου θα είναι σημαντικά μικρότερος. Συνεπώς:

$$T(n) = \sum_{i=0}^{k} n \left(\frac{3}{4}\right)^i < \sum_{i=0}^{\infty} n \left(\frac{3}{4}\right)^i = n\frac{1}{1-\frac{3}{4}} = 4n = \Theta(n)$$

4.

$$T(n) = 2\,T(n/4) + T(n/2) + \Theta(n)$$

Παρατηρούμε ότι $\frac{2n}{4} + \frac{n}{2} = 1$. Συνεπώς, υποπτευόμαστε ότι $T(n) = \Theta(n \log n)$. Κάθε μία από τις διαδρομές διαδοχικών κινήσεων $n/4$ και $n/2$ θα μας δώσει τα δύο φράγματα στον χρόνο εκτέλεσης της $T(n)$. Συγκεκριμένα, η διαδρομή με τα διαδοχικά $n/2$ θα μας δώσει ύψος υπόδεντρου ίσο με $\frac{n}{2^{h_1}} = 1 \Rightarrow h_1 = \log n$ και η διαδρομή με τα διαδοχικά $n/4$ θα μας δώσει ύψος υπόδεντρου ίσο με $\frac{n}{4^{h_2}} = 1 \Rightarrow h_2 = \log_4 n$. Συνεπώς:

$$\sum_{i=0}^{h_2} n \le T(n) \le \sum_{i=0}^{h_1} n \ \Rightarrow n\,(\log_4 n + 1) \le T(n) \le n\,(\log n + 1) \Rightarrow T(n) = \Theta(n)$$

5.

$$T(n) = T\left(\sqrt{n}\right) + \Theta(\log n)$$

Γνωρίζουμε ότι το υψηλότερο επίπεδο του δέντρου αναδρομής συνεισφέρει $\log n$ στον χρόνο εκτέλεσης, συνεπώς $T(n) = \Omega(\log n)$. Κάθε επίπεδο του δέντρου έχει χρόνο εκτέλεσης $\Theta\left(\log n^{(1/2)^i}\right) = \frac{c}{2^i}\log n$ για $i = 1, \dots, h$. Συνεπώς, για το πάνω φράγμα έχουμε ότι:

$$T(n) < \sum_{i=1}^{\infty} \frac{c}{2^i}\log n = \mathrm{c}\log n \sum_{i=1}^{\infty} \frac{1}{2^i} = c\log n \Rightarrow T(n) = \mathrm{O}(\log n) \Rightarrow T(n) = \Theta(\log n)$$

6.

$$T(n) = T(n/4) + \Theta(\sqrt{n})$$

Γνωρίζουμε ότι το υψηλότερο επίπεδο του δέντρου αναδρομής συνεισφέρει $\Theta\left(\sqrt{n}\right)$ στον χρόνο εκτέλεσης, συνεπώς $T(n) = \Omega\left(\sqrt{n}\right)$. Κάθε επίπεδο του δέντρου συνεισφέρει στον χρόνο εκτέλεσης $\Theta\left(\frac{\sqrt{n}}{2^i}\right)$ για $i = 1, \dots, \log_4 n$. Συνεπώς, για το πάνω φράγμα έχουμε ότι:

$$T(n) < \sum_{i=1}^{\infty} \frac{c}{2^i}\sqrt{n} = c\sqrt{n} \sum_{i=1}^{\infty} \frac{1}{2^i} = c\sqrt{n} \Rightarrow T(n) = \mathrm{O}\left(\sqrt{n}\right) \Rightarrow T(n) = \Theta\left(\sqrt{n}\right)$$

Άσκηση 2η – Προθεματική Ταξινόμηση

(a)

Ένας αλγόριθμος που ταξινομεί τον πίνακα είναι ο εξής:

1. Βρες τη θέση του μεγαλύτερου στοιχείου του πίνακα, έστω k.
2. Εφάρμοσε προθεματική περιστροφή γύρω από το k. Τώρα το $A[k]$ βρίσκεται στην πρώτη θέση.
3. Εφάρμοσε προθεματική περιστροφή γύρω από το n. Τώρα το στοιχείο βρίσκεται στην τελευταία θέση.
4. Εκτέλεσε την διαδικασία αναδρομικά για το $A[1 \dots (n-1)]$ μέχρις ότου ο A να έχει μόνο ένα στοιχείο.

Σε κάθε βήμα ο αλγόριθμος εκτελεί 2 προθεματικές περιστροφές, ενώ τα βήματα είναι n. Συνεπώς έχουμε το πολύ $2n$ περιστροφές. Αυτό μπορούμε να το βελτιώσουμε κατά ένα μικρό πλήθος κινήσεων σταματώντας την αναδρομή π.χ. όταν ο A θα έχει 3 στοιχεία. Τότε μπορούμε να πετύχουμε πλήθος περιστροφών ίσο με $2n - 3$.

(b)

Ακολουθούμε παρόμοιο αλγόριθμο με εκείνο του πρώτου ερωτήματος με τη διαφορά ότι φροντίζουμε να επιδιορθώνουμε τα πρόσημα των στοιχείων που αντιστρέφουμε σε κάθε βήμα της αναδρομής. Για να το κάνουμε αυτό εκτελούμε μία επιπλέον περιστροφή. Συγκεκριμένα:

1. Βρες τη θέση του μεγαλύτερου στοιχείου του πίνακα, έστω k.
2. Εφάρμοσε προσημασμένη προθεματική περιστροφή γύρω από το k. Τώρα το $A[k]$ βρίσκεται στην πρώτη θέση. Επιπλέον όλα τα στοιχεία με δείκτη $1 \dots k$ έχουν ανάποδο πρόσημο.
3. Εφάρμοσε προσημασμένη προθεματική περιστροφή γύρω από το n. Τώρα το στοιχείο βρίσκεται στην τελευταία θέση με σωστό πρόσημο ενώ οι μέχρι πρότινος $n - k$ τελευταίοι όροι έχουν μεταφερθεί στην αρχή του πίνακα με ανάποδο πρόσημο.
4. Εφάρμοσε προσημασμένη προθεματική περιστροφή γύρω από το $n - k$. Τώρα, τα πρώτα $n - k$ στοιχεία έχουν περιστραφεί αλλά έχουν σωστό πρόσημο, ενώ το μεγαλύτερο στοιχείο βρίσκεται στο τέλος του πίνακα.
5. Εκτέλεσε την διαδικασία αναδρομικά για το $A[1 \dots (n-1)]$ μέχρις ότου ο A να έχει μόνο ένα στοιχείο.

Το πλήθος των προσημασμένων προθεματικών περιστροφών στην χειρότερη περίπτωση είναι $3n$. Παρατηρούμε επίσης ότι η υποπερίπτωση κατά την οποία το μεγαλύτερο στοιχείο βρίσκεται στην πρώτη θέση του πίνακα συμπεριλαμβάνεται. Τότε κάνουμε προθεματική περιστροφή γύρω από το στοιχείο με δείκτη 1, πράγμα που έχει ως αποτέλεσμα απλά την αλλαγή του προσήμου του πρώτου στοιχείου. Έπειτα το στέλνουμε στην τελευταία θέση με περιστροφή γύρω από το n και για την επιδιόρθωση των προσήμων των $n - 1$ προηγούμενων στοιχείων κάνουμε περιστροφή γύρω από το $n - 1$.

(c)

1.

Εάν στον τρέχοντα πίνακα A_t υπάρχει στοιχείο με θετικό πρόσημο, μπορούμε να βρούμε το μεγαλύτερό του στοιχείο, έστω x (αναγκαστικά θετικό), και να προσπαθήσουμε να φτιάξουμε κάποιο συμβατικό ζεύγος που το περιέχει. Εάν $x = n$, τότε μπορούμε να το μεταφέρουμε απλά στο τέλος του πίνακα με δύο περιστροφές, πράγμα που θεωρείται καταχρηστικά συμβατό ζεύγος στην εκφώνηση. Σε κάθε άλλη περίπτωση πρέπει να ενώσουμε τα στοιχεία x και $x + 1$, τα οποία μπορεί να είναι διατεταγμένα ως εξής:

i. $[\dots, -(x+1), \dots, x, \dots] \rightarrow [-x, \dots, (x+1), \dots] \rightarrow [\dots, x, (x+1), \dots]$
ii. $[\dots, x, \dots, -(x+1), \dots] \rightarrow [(x+1), \dots, -x, \dots] \rightarrow [\dots, -(x+1), -x, \dots]$

Καθώς το $x + 1$ έχει αρνητικό πρόσημο αφού το x είναι ο μεγαλύτερος θετικός.

Εάν όλα τα στοιχεία στον πίνακα είναι αρνητικά και δεν βρισκόμαστε στην κατάσταση $[-1, -2, \dots, -n]$, τότε υπάρχει στοιχείο $-x$ για το οποίο υπάρχει στοιχείο $-(x+1)$ αριστερότερά του. Συνεπώς, μπορούμε να κάνουμε την εξής αλληλουχία περιστροφών:

$$[\dots, -(x+1), \dots, -x, \dots] \rightarrow [(x+1), \dots, -x, \dots] \rightarrow [\dots, -(x+1), -x, \dots]$$

Άρα σε οποιαδήποτε κατάσταση A_t βρισκόμαστε, μπορούμε με το πολύ δύο προσημασμένες προθεματικές περιστροφές να κατασκευάσουμε νέο συμβατό ζεύγος.

2.

Ένας αλγόριθμος ταξινόμησης που χρησιμοποιεί τη λογική των συμβατών ζευγών που περιγράψαμε παραπάνω είναι ο εξής:

1. Απάλειψε όλα τα συμβατά ζεύγη αναδρομικά. Κάθε φορά που συναντάμε ένα συμβατό ζεύγος μπορούμε να το αντικαταστήσουμε με την τιμή του ενός από τα δύο στοιχεία. Έτσι θα δημιουργηθεί νέος πίνακας με στοιχεία πλήθους m. Αν $m = 0$, ο πίνακας είναι ταξινομημένος.

2. Αν ο πίνακας έχει τη μορφή $[-1, -2, \dots, -m]$
   Επαναλαμβάνουμε m φορές:
   i. Εκτέλεσε προσημασμένη προθεματική περιστροφή όλων των στοιχείων.
   ii. Εκτέλεσε προσημασμένη προθεματική περιστροφή των $m-1$ αριστερότερων στοιχείων.
   Ο πίνακας έχει ταξινομηθεί.
3. Αλλιώς, κατασκεύασε ένα συμβατό ζεύγος (βλ. προηγούμενο ερώτημα) και επανάλαβε το βήμα 1.

Ο αρχικός πίνακας n στοιχείων για να ταξινομηθεί απαιτεί την κατασκευή n συμβατών ζευγών. Συνεπώς, αν ο αλγόριθμος εκτελεστεί x φορές τότε σε κάθε μία από αυτές θα γίνουν το πολύ 2 κινήσεις για την κατασκευή ενός συμβατού ζεύγους και έτσι θα μειώνουμε το πλήθος των ζευγών κατά 1 κάθε φορά. Άρα συνολικά απαιτούνται $2x + 2(n-x) = 2n$ προσημασμένες προθεματικές περιστροφές.

Άσκηση 3η – Υπολογισμός Κυρίαρχων Θέσεων

Θα κατασκευάσουμε πίνακα $K[1, \dots, n]$ που στη θέση i θα περιέχει τη θέση του κυρίαρχου στοιχείου του $A[i]$ για $i = 1, \dots, n$. Τα βήματα του αλγόριθμου είναι τα εξής:

1. Έστω στοίβα S και τελικός πίνακας K.
2. Κάνε push στην S τον αριθμό 1 και append στον K τον αριθμό 0, καθώς πρέπει $K[0] = 0$.
3. Για i από 2 μέχρι n:
   i. Όσο η S δεν είναι άδεια και $A[top] \leq A[i]$, κάνε pop στην S.
   ii. Εάν η S είναι άδεια, κάνε append στον K τον αριθμό 0, αφού το κυρίαρχο στοιχείο του $A[i]$ είναι το $A[0] = \infty$. Αλλιώς κάνε append στον K το top.
   iii. Κάνε push στην S το i.

Η πολυπλοκότητα του αλγορίθμου που περιγράφηκε είναι $O(n)$ καθώς διατρέχουμε μία φορά τον αρχικό πίνακα. Σε κάθε βήμα κάνουμε pop από τη στοίβα θέσεις στοιχείων του πίνακα A στις οποίες γνωρίζουμε ότι η τιμή τους είναι μικρότερη ή ίση από την τιμή της τρέχουσας θέσης. Αφού λοιπόν βρούμε τον κυρίαρχο του i-οστού στοιχείου και προσθέσουμε το i στη στοίβα, για τον κυρίαρχο του $i+1$-οστού όρου δεν χρειάζεται να ελέγξουμε τα στοιχεία τα οποία βρίσκονται αριστερότερα του $A[i]$ και είναι μικρότερα από αυτό καθώς θέλουμε είτε να σταματήσουμε στο i (περίπτωση όπου $A[i] > A[i+1]$) είτε να βρούμε κάποιο j αριστερότερα του i για το οποίο ισχύει $A[j] > A[i+1]$ και συνεπώς $A[j] > A[i]$. Με τη χρήση της στοίβας επιπλέον εξασφαλίζουμε το ότι θα βρούμε πάντα το πλησιέστερο j καθώς εισάγονται σε αυτήν δείκτες με αύξουσα σειρά.

Άσκηση 4η – Φόρτιση Ηλεκτρικών Αυτοκινήτων

Η βασική ιδέα είναι να βρω το ελάχιστο δυνατό s^* για το οποίο ικανοποιείται η συνθήκη όπου κανένα όχημα δεν παραμένει πάνω από $d+1$ χρόνο σε σταθμό φόρτισης (d χρόνος αναμονής και μία μονάδα που απαιτείται για την φόρτιση). Για δεδομένο s^* είναι προφανές ότι μπορούν μία στιγμή να εξυπηρετηθούν ταυτόχρονα ακριβώς s^* οχήματα, ενώ τα υπόλοιπα μπαίνουν σε αναμονή. Συνεπώς ένα όχημα που βρίσκεται πίσω από $d-1$ s^*-άδες αυτοκινήτων θα εξυπηρετηθεί σε χρόνο ακριβώς d. Αυτό σημαίνει ότι κάθε χρονική στιγμή ο αριθμός των οχημάτων που βρίσκονται σε αναμονή δεν πρέπει να ξεπερνά το s^*d. Προτού ξεκινήσουμε την αναζήτηση για το βέλτιστο s^* θα κάνουμε μια προεπεξεργασία του πίνακα αφίξεων. Θα κατασκευάσουμε έναν πίνακα B με τούπλες της μορφής (χρόνος, αυτοκίνητα που καταφθάνουν σε αυτό τον χρόνο), αποκλείοντας τις στιγμές όπου δεν καταφθάνει κανένα αυτοκίνητο στον σταθμό. Αυτό θα έχει ως αποτέλεσμα ο B να έχει το πολύ n στοιχεία. Με δυαδική αναζήτηση για το s^* στο διάστημα $[1, n]$ με αρχικούς δείκτες $left = 1$, $right = n$ και $mid = \left\lfloor \frac{right+left}{2} \right\rfloor$. Όταν για κάποιο s^* δεν θα επαρκούν οι φορτιστές θα πηγαίνουμε στο δεξί υποδιάστημα ανανεώνοντας τους δείκτες σε $left = mid + 1$, $mid = \left\lfloor \frac{right+left}{2} \right\rfloor$ ενώ όταν βρίσκουμε έγκυρο s^* θα πηγαίνουμε στο αριστερό υποδιάστημα μήπως εντοπίσουμε μικρότερη έγκυρη τιμή για το s^*

με νέους δείκτες $right = mid$ και $mid = \left\lfloor \frac{right+left}{2} \right\rfloor$. Έτσι, θα βρούμε την ελάχιστη τιμή για το s^* που ικανοποιεί τον εξής αλγόριθμο.

Σε κάθε βήμα κοιτάμε τον αριθμό των αυτοκινήτων που έρχονται μία δεδομένη χρονική στιγμή. Το πλήθος αυτό προστίθεται σε μία βοηθητική μεταβλητή $cars_{waiting}$. Ύστερα αφαιρούμε από αυτή τη μεταβλητή s^* οχήματα που μπορούν να εξυπηρετηθούν αυτή τη στιγμή. Εάν παρατηρήσουμε ότι έχουν απομείνει αυτοκίνητα στην $cars_{waiting}$ και η επόμενη χρονική στιγμή στον πίνακα B δεν διαφέρει κατά 1 από την τωρινή, θα προχωρήσουμε στην αμέσως επόμενη και θα προσπαθήσουμε να εξυπηρετήσουμε s^* από αυτά τα αυτοκίνητα που βρίσκονται σε αναμονή. Αυτή τη διαδικασία αποσυμφόρησης της ουράς αναμονής την επαναλαμβάνουμε έως ότου είτε να μηδενιστούν τα αυτοκίνητα στην $cars_{waiting}$ είτε να φτάσουμε σε χρονική στιγμή που καταφθάνουν νέα αυτοκίνητα, δηλαδή στο επόμενο στοιχείο που εμφανίζεται στον B. Οπότε τότε προσθέτουμε τα νέα οχήματα στην $cars_{waiting}$ κ.ο.κ. Εάν σε κάποιο βήμα η μεταβλητή $cars_{waiting}$ ξεπεράσει την τιμή s^*d τότε δηλώνουμε αποτυχία, ενώ αν τερματίσει ο αλγόριθμος χωρίς αποτυχία τότε ανακοινώνουμε ότι το δεδομένο s^* είναι έγκυρο.

Η πολυπλοκότητα της κατασκευής του B είναι $\Theta(n)$ χρονικά καθώς διατρέχουμε μία φορά τον πίνακα αφίξεων και $\Theta(n)$ χωρικά καθώς περιέχει το πολύ n στοιχεία. Για κάθε υποψήφιο s^* ο χρόνος εκτέλεσης του αλγορίθμου που περιγράφηκε είναι $\Theta(n)$. Συνεπώς, με τη δυαδική αναζήτηση η συνολική πολυπλοκότητα είναι $\Theta(n \log n)$.

Άσκηση 5η – Επιλογή

(a) Για την εύρεση του k-οστού μικρότερου στοιχείου του multiset, θα εφαρμόσουμε δυαδική αναζήτηση στο διάστημα των δυνατών τιμών που μπορεί να έχει ο k-οστός όρος. Συγκεκριμένα, αρχικοποιώντας μία μεταβλητή mid στην τιμή $mid = \left\lfloor \frac{start+end}{2} \right\rfloor$, όπου $start = 0$ και $end = M$, θα ελέγξουμε πόσα στοιχεία στο σύνολο δεν ξεπερνούν την τιμή mid καλώντας τη συνάρτηση κατανομής $F_S(mid)$. Εάν $F_S(mid) \leq k$ μεταφερόμαστε στο διάστημα $[start, mid]$ με νέες τιμές για τα $end = mid$ και $mid = \left\lfloor \frac{start+mid}{2} \right\rfloor$, ενώ αν προκύψει $F_S(mid) > k$ μεταφερόμαστε στο διάστημα $[mid + 1, end]$ με $start = mid + 1$ και $mid = \left\lfloor \frac{mid+1+end}{2} \right\rfloor$. Συνεχίζουμε αναδρομικά τη διαδικασία μέχρις ότου οι δείκτες $start$ και end να ταυτιστούν (μία πιθανή τιμή για το mid). Αυτή η τιμή είναι και το στοιχείο που ψάχνουμε.

Ουσιαστικά ψάχνουμε τον μικρότερο αριθμό x του συνόλου για τον οποίο ισχύει ότι το πλήθος των στοιχείων του S που δεν ξεπερνούν το x είναι ακριβώς ίσο με k και άρα αυτό το x είναι το μεγαλύτερο από τα k μικρότερα στοιχεία του S. Έτσι είναι λογικό όταν συναντάμε mid με $F_S(mid) = k$ να ψάχνουμε στο αριστερό ημιδιάστημα ώστε να εντοπίσουμε το μικρότερο από τα mid για τα οποία ισχύει η ισότητα. Σε κάθε άλλη περίπτωση μετακινούμαστε ανάλογα ώστε να διορθώσουμε το mid. Ο αλγόριθμος προφανώς λειτουργεί και για τις υποπεριπτώσεις όπου το $F_S(mid)$ δεν γίνεται k ή υπάρχουν διπλότυπα στο πολυσύνολο, όπου πάντα στο προτελευταίο βήμα της αναδρομής θα επιλεχθεί ο δεξής όρος (αφού $F_S(mid) < k$) και θα λάβουμε ως αποτέλεσμα τον αμέσως επόμενο όρο του mid.

Ο παραπάνω αλγόριθμος, στη χειρότερη περίπτωση, θα εκτελέσει τη συνάρτηση F_S το πολύ $\log M$ φορές καθώς τότε θα κάνουμε δυαδική αναζήτηση στο διάστημα $[1, M]$.

(b) Χρησιμοποιώντας τον αλγόριθμο που περιγράφηκε στο προηγούμενο ερώτημα, θα βρούμε το k-οστό μικρότερο στοιχείο στο σύνολο διαφορών S. Για να μη χρειαστεί να κατασκευάσουμε αυτές τις διαφορές και να πληρώσουμε χωρική και χρονική πολυπλοκότητα $\Theta(n^2)$, θα ταξινομήσουμε τον πίνακα A σε χρόνο $O(n \log n)$. Με αυτό τον τρόπο, γνωρίζουμε ότι η μεγαλύτερη διαφορά έχει τιμή $max = A[n] - A[1]$ ενώ για να βρούμε τη μικρότερη μπορούμε να διατρέξουμε μία φορά τον πίνακα και να ελέγχουμε διαφορές διαδοχικών στοιχείων, κρατώντας πάντα την μικρότερη. Πλέον, μπορούμε να εφαρμόσουμε τον αλγόριθμο του πρώτου ερωτήματος για να βρούμε τη ζητούμενη διαφορά, αφού αναφέρουμε μία

αποδοτική υλοποίηση της F_S για το παρόν πρόβλημα. Μπορούμε να υλοποιήσουμε αποδοτικά σε χρόνο $O(n)$ την κατανομή των διαφορών F_S ως εξής:

1. Αρχικοποίηση μεταβλητών $result = 0, j = 1$
2. Για i από 0 μέχρι n:
   Όσο $j < n$ και $A[j] - A[i] \leq k$:
   $j = j + 1$
   $result = result + (j - 1) - i$
3. Επίστρεψε το $result$

Η παραπάνω διαδικασία είναι σωστή καθώς ο A είναι ταξινομημένος. Σε κάθε βήμα προχωρώντας τον δείκτη j και αποτιμώντας τη διαφορά $A[j] - A[i]$ βρίσκουμε το μήκος ενός διαστήματος $[i, j-1]$ με έγκυρες διαφορές. Προσθέτουμε το μήκος του διαστήματος στο τελικό αποτέλεσμα και προχωράμε τον δείκτη i ώστε να βρούμε κι άλλες έγκυρες διαφορές που εντοπίζονται στον πίνακα. Ο αλγόριθμος κάνει n επαναλήψεις και συνεπώς έχει πολυπλοκότητα $O(n)$. Τελικά ο αλγόριθμος θέλει $O(n \log n)$ για την ταξινόμηση και $O(n \log M)$ για την εύρεση της k-οστή μικρότερη διαφορά. Άρα η πολυπλοκότητα είναι $O\big(n\ (\log n + \log M)\big) = O(n \log(n\ M)) = O(n \min(\log n, \log M))$.

Άσκηση 6η – Ερωτήματα Ανήκειν

(a) Υποθέτουμε πίνακα A, m θέσεων, όπου αποθηκεύουμε τη σύνοψη του S, αρχικοποιημένο με μηδενικά. Για την αρχικοποίηση της δομής θα προβούμε στα εξής βήματα. Για κάθε $x \in S$, υπολογίζουμε τη συνάρτηση κατακερματισμού $h(x)$ η οποία επιστρέφει κάποιον δείκτη j με $\mathbb{P}[h(x) = j] = 1/m$. Μεταβάλλουμε το $A[j] = 1$. Για να ελέγξουμε για την ύπαρξη ενός στοιχείου x στο σύνολο S, υπολογίζουμε την τιμή $h(x)$ και εάν $A[h(x)] = 1$ τότε απαντάμε θετικά. Σε αυτή την περίπτωση η δομή μας απαντάει σωστά. Εάν θεωρήσουμε ότι στο S υπάρχουν κλειδιά πλήθους n, η πιθανότητα κάποιο bit να μην έχει πάρει την τιμή 1 είναι $\mathbb{P}\big[A[h(x) \neq 1\ ]\big] = \left(1 - \frac{1}{m}\right)^n = e^{-n/m}$. Συνεπώς, η πιθανότητα ενός False Positive ερωτήματος είναι ίση με $\mathbb{P}[False\ Positive] = 1 - e^{-n/m}$. Για $m = 8n$ αυτή είναι ίση με $1 - e^{-1/8} \approx 0{,}1175$.

(b) Για k συναρτήσεις κατακερματισμού, η αρχικοποίηση του πίνακα A για κάθε στοιχείο $x \in S$ υπολογίζουμε τις τιμές των συναρτήσεων κατακερματισμού $h_i(x)$ για $i = 1, \dots, k$ και θέτουμε $A[h_i(x)] = 1$. Για να ελέγξουμε εάν κάποιο στοιχείο x ανήκει στο σύνολο, κοιτάμε εάν όλες οι θέσεις $A[h_1(x)], A[h_2(x)], \dots, A[h_k(x)]$ έχουν τιμή ίση με 1. Αν αυτό ισχύει απαντάμε θετικά. Η πιθανότητα κάποια θέση του A να μην έχει γίνει 1, πράγμα που σημαίνει ότι κανένα από τα n στοιχεία δεν έχει γίνει map στο i-οστό bit για καμία από τις k συναρτήσεις κατακερματισμού, είναι ίση με $\mathbb{P}\big[A[h_i(x) \neq 1\ ]\big] = \left(1 - \frac{1}{m}\right)^{kn} = e^{-kn/m}$. Συνεπώς ένα οποιοδήποτε bit i θα τεθεί ίσο με 1 με πιθανότητα $\mathbb{P}\big[A[h_i(x) = 1\ ]\big] = 1 - e^{-kn/m}$. Επειδή όμως για κάθε στοιχείο που κατακερματίζουμε θέτουμε k θέσεις του A ίσες με 1, για να έχουμε περίπτωση False Positive, πρέπει η προηγούμενη πιθανότητα να πολλαπλασιαστεί k φορές με τον εαυτό της, καθώς ισχύει για μόνο μία από τις k αναθέσεις που γίνονται. Άρα η πιθανότητα False Positive είναι $\mathbb{P}[False\ Positive] = \left(1 - e^{-kn/m}\right)^k$. Για $m = 8n$ έχουμε ότι $\mathbb{P}[False\ Positive] = \left(1 - e^{-k/8}\right)^k$ και για να βρούμε το βέλτιστο k που δίνει την ελάχιστη πιθανότητα σφάλματος έχουμε:

$$\frac{d}{dk}\left(1 - e^{-k/8}\right)^k = 0 \Rightarrow k \approx 5{,}54$$

Δηλαδή $k = 6$.