Αλγόριθμοι και Πολυπλοκότητα

Πρώτη Σειρά Ασκήσεων

Εργαστήριο Λογικής και Επιστήμης Υπολογιστών

ΣΗΜΜΥ - ΕΜΠ

14 Νοέμβρη, 2016

Περίληψη

Άσκηση 1α: Ασυμπτωτικός Συμβολισμός

1. $\sum_{k=1}^{n} k2^{-k}$
2. $\log^2 n / \log\log n$
3. $\log(\frac{n}{\log n})$
4. $\log^4 n$
5. $\log(\binom{2n}{n}), n2^{2^{2^{100}}} = \Theta(n)$
6. $\log(n!)/\log^3 n$
7. $n^{3/2}$
8. $n^3/\log^8 n$
9. $\binom{n}{6} = \Theta(n^6)$
10. $(\log n)^{\log n} = \Theta(n^{\log\log n})$
11. $3^{(\log n)^3}$
12. $\sum_{k=1}^{n} k2^k, n\sum_{k=1}^{n}\binom{n}{k} = \Theta(n2^n)$
13. $\sqrt{n!}$
14. $n^{\log n!}$

Άσκηση 1α: Ασυμπτωτικός Συμβολισμός

Χρήσιμες σχέσεις

- $\forall \epsilon > 0, \log^d n = o(n^\epsilon)$ (π.χ. $n^2 / \log^{10} n = \omega(n)$)
- Προσοχή όταν λογαριθμίζουμε, οι μικρότεροι όροι έχουν πλέον σημασία!
  - π.χ. $\log((\log n)^{\log n}) = \Theta(\log((\log n)^{\log(16n)}))$ αλλά $(\log n)^{\log n} = o((\log n)^{\log(16n)})$
  - π.χ. $n! = \omega\left(\frac{n}{2}\right)^{n/2}$ και $n! = o(n^n)$ αλλά $\log n! = \Theta(\log n^n)$
- $\forall \epsilon > 0, n^\epsilon c^n = o(d^n)$ αν $c < d$ (π.χ. $n(2.5)^n = o(e^n)$)

Άσκηση 1α: Ασυμπτωτικός Συμβολισμός

Χρήσιμες γνωστές ανισότητες

- $\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \frac{n^k}{k!} \leq \left(\frac{n \cdot e}{k}\right)^k$ $(\pi.\chi.\ n \log 2 \leq \log \binom{2n}{n} \leq n \log(2e))$
- $e\left(\frac{n}{e}\right)^n \leq n! \leq e\left(\frac{n+1}{e}\right)^{n+1}$
- *Stirling's approximation*: $\log(n!) \simeq n \log n - n + O(\log n)$

Άσκηση 1α: Ασυμπτωτικός Συμβολισμός

Χρήσιμο 'τρικ' για σειρές

Διαισθητικά, $s_n = \sum_{i=1}^{n} i2^{-i} = \Theta(1)$ γιατί ο αριθμητής αυξάνει γραμμικά και ο παρανομαστής εκθετικά. Μία τυπική απόδειξη είναι η εξής:

$$s_n = \frac{1}{2^1} + \frac{2}{2^2} + \frac{3}{2^3} + \ldots + \frac{n}{2^n} \Leftrightarrow 2s_n = \frac{1}{2^0} + \frac{2}{2^1} + \frac{3}{2^2} + \ldots + \frac{n}{2^{n-1}}$$

$$\Leftrightarrow 2s_n = 2 + \sum_{i=1}^{n-2} \frac{i+2}{2^{i+1}} \Leftrightarrow 2s_n = 2 + \frac{1}{2}\left[\sum_{i=1}^{n} \frac{i}{2^i} - \frac{n}{2^n} - \frac{n-1}{2^{n-1}}\right] + \sum_{i=1}^{n-2} \frac{1}{2^i}$$

$$\Leftrightarrow 2s_n = 2 + s_n/2 - \frac{1}{2}\left[\frac{n}{2^n} + \frac{n-1}{2^{n-1}}\right] + \left(1 - \frac{1}{2^{n-2}}\right)$$

$$\Leftrightarrow \frac{3}{2}s_n = 3 - \frac{3n+6}{2^{n+1}} \Leftrightarrow s_n = 2 - \frac{n+2}{2^n}$$

Άσκηση 1β: Αναδρομικές Σχέσεις

1. $T(n) = 2T(n/3) + n\log n = \Theta(n\log n)$: ΜΤ περ.3
2. $T(n) = 3T(n/3) + n\log n = \Theta(n\log^2 n)$: Όχι ΜΤ γιατί δεν είναι πολυωνυμικά διαχωρίσιμες. Επειδή $3n/3 = n$ και $f(n) = n\log n$, υποπτευόμαστε $T(n) = \Theta(n\log^2 n)$ και το αποδεικνύουμε με επαγωγή ή χρησιμοποιώντας το δέντρο αναδρομής.
3. $T(n) = 4T(n/3) + n\log n = \Theta(n^{\log_3 4})$: ΜΤ περ.1
4. $T(n) = T(n/2) + T(n/3) + n = \Theta(n)$: Επειδή $n/2 + n/3 < n$, υποπτευόμαστε $T(n) = \Theta(n)$ και το αποδεικνύουμε με επαγωγή ή δέντρο αναδρομής.
5. $T(n) = T(n/6) + T(n/2) + T(n/3) + n = \Theta(n\log n)$: Επειδή $n/6 + n/2 + n/3 = n$ και $f(n) = n$, υποπτευόμαστε $T(n) = \Theta(n\log n)$ και το αποδεικνύουμε με επαγωγή ή χρησιμοποιώντας το δέντρο αναδρομής. Στο τελευταίο, μπορούμε να φράξουμε το ύψος του από το κοντύτερο κλαδί ($\log_6 n$) και το μακρύτερο ($\log_2 n$).

Άσκηση 1β: Αναδρομικές Σχέσεις

6. $T(n) = T(n-1) + \log n = \Theta(n \log n)$: αφού $T(n) = T(n-1) + \log(n) = T(1) + \log(2) + \ldots + \log(n) = \Theta(1) + \log(n!) = \Theta(n \log n)$

7. $T(n) = T(n^{5/6}) + \Theta(\log n) = \Theta(\log n)$: αφού $T(n) \geq \log n$ και $T(n) \leq \Theta(\log n)$. Το δεύτερο προκύπτει αν αναπτύξουμε την αναδρομική σχέση.

8. $T(n) = T(n/4) + \sqrt{n} = \Theta(\sqrt{n})$:
Θέτουμε $n = k^2$ και εργαζόμαστε με τη σχέση
$S(k) = S(k/2) + \Theta(k), S(k) = T(k^2)$ η οποία καταλήγει ως γνωστόν
$S(k) = \Theta(k)$.

Άσκηση 2α: Ταξινόμηση

Ορισμός

Έστω πίνακας $A[n]$ και οι υποπίνακες $A_1[1...\frac{n}{k}], A_2[\frac{n}{k}+1...\frac{2n}{k}], \ldots, A_k[\frac{(k-1)n}{k}+1...n]$ προκύπτουν από την διαμέριση του $A[n]$ σε k μέρη μεγέθους $\frac{n}{k}$. Χ.β.τ.γ n και k δυνάμεις του 2 με το k να διαιρεί το n. A καλείται ταξινομημένος κατά k μέρη αν $\forall i, j : 1 \leq i < j \leq k$ κάθε στοιχείο του A_i είναι $\leq$ από κάθε στοιχείο του A_j.

Άσκηση 2α: Ταξινόμηση

Ορισμός

Έστω πίνακας $A[n]$ και οι υποπίνακες $A_1[1...\frac{n}{k}], A_2[\frac{n}{k}+1...\frac{2n}{k}], \ldots, A_k[\frac{(k-1)n}{k}+1...n]$ προκύπτουν από την διαμέριση του $A[n]$ σε k μέρη μεγέθους $\frac{n}{k}$. Χ.β.τ.γ n και k δυνάμεις του 2 με το k να διαιρεί το n. A καλείται ταξινομημένος κατά k μέρη αν $\forall i, j : 1 \leq i < j \leq k$ κάθε στοιχείο του A_i είναι $\leq$ από κάθε στοιχείο του A_j.

Πρόβλημα (1)

Άσκηση 2α: Ταξινόμηση

Ορισμός

Έστω πίνακας $A[n]$ και οι υποπίνακες $A_1[1...\frac{n}{k}], A_2[\frac{n}{k}+1...\frac{2n}{k}], \ldots, A_k[\frac{(k-1)n}{k}+1...n]$ προκύπτουν από την διαμέριση του $A[n]$ σε k μέρη μεγέθους $\frac{n}{k}$. Χ.β.τ.γ n και k δυνάμεις του 2 με το k να διαιρεί το n. A καλείται ταξινομημένος κατά k μέρη αν $\forall i, j : 1 \leq i < j \leq k$ κάθε στοιχείο του A_i είναι $\leq$ από κάθε στοιχείο του A_j.

Πρόβλημα (1)

- **Είσοδος:** Πίνακας $A[n]$.

Άσκηση 2α: Ταξινόμηση

Ορισμός

Έστω πίνακας $A[n]$ και οι υποπίνακες $A_1[1...\frac{n}{k}], A_2[\frac{n}{k}+1...\frac{2n}{k}], \ldots, A_k[\frac{(k-1)n}{k}+1...n]$ προκύπτουν από την διαμέριση του $A[n]$ σε k μέρη μεγέθους $\frac{n}{k}$. Χ.β.τ.γ n και k δυνάμεις του 2 με το k να διαιρεί το n. A καλείται ταξινομημένος κατά k μέρη αν $\forall i, j : 1 \leq i < j \leq k$ κάθε στοιχείο του A_i είναι $\leq$ από κάθε στοιχείο του A_j.

Πρόβλημα (1)

- **Είσοδος**: Πίνακας $A[n]$.
- **Ζητούμενο**: Αλγόριθμος ταξινόμησης σε k μέρη του A με χρόνο $\mathcal{O}(nlogk)$.

Άσκηση 2α: Αλγόριθμος Ταξινόμησης (1)

Άσκηση 2α: Αλγόριθμος Ταξινόμησης (1)

- Κάνε *QuickSort* μέχρι το επίπεδο k με ντετερμινιστική *Partition*, βάσει του *median*.

Άσκηση 2α: Αλγόριθμος Ταξινόμησης (1)

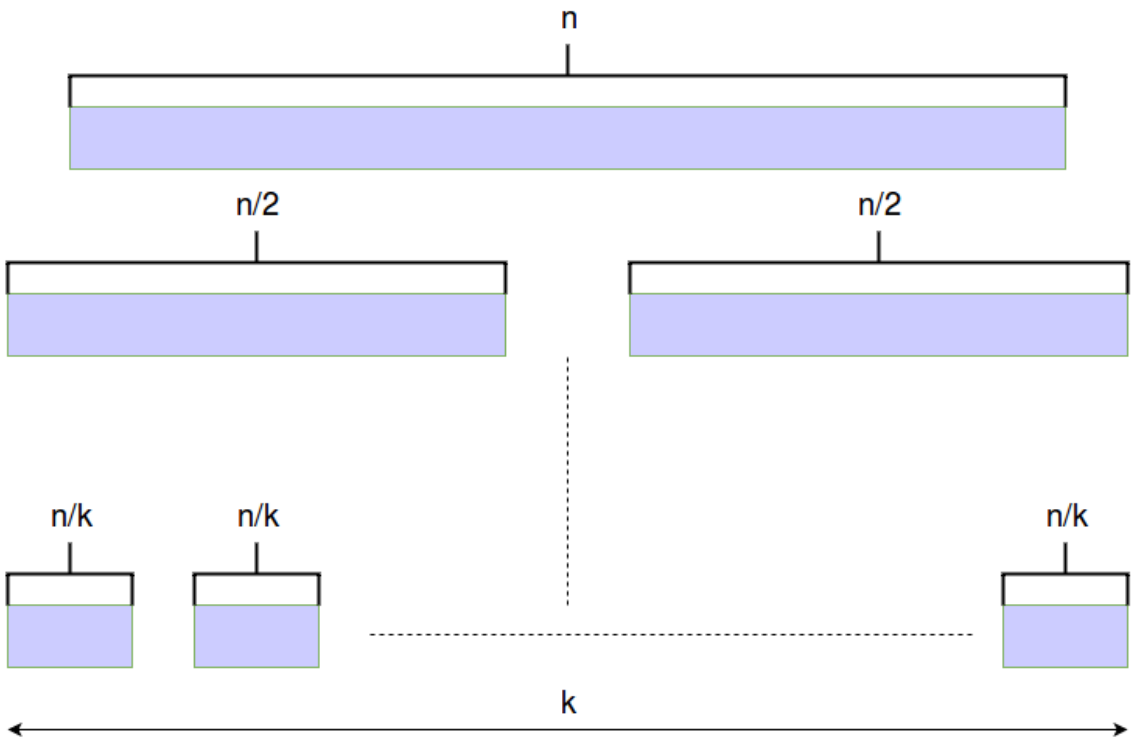
- Κάνε *QuickSort* μέχρι το επίπεδο k με ντετερμινιστική *Partition*, βάσει του *median*.
- Αφού το k είναι πολλαπλάσιο του 2 κάποια στιγμή θα καταλήξουμε σε κομμάτια πίνακα μεγέθους $\frac{n}{2^z}$, όπου $k = 2^z$.

Άσκηση 2α: Αλγόριθμος Ταξινόμησης (1)

- Κάνε *QuickSort* μέχρι το επίπεδο k με ντετερμινιστική *Partition*, βάσει του *median*.
- Αφού το k είναι πολλαπλάσιο του 2 κάποια στιγμή θα καταλήξουμε σε κομμάτια πίνακα μεγέθους $\frac{n}{2^z}$, όπου $k = 2^z$.
- Η πολυπλοκότητα είναι $\mathcal{O}(n)$ (από την *Partition* και *Median*) $\cdot$ $\mathcal{O}(logk)$ (γιατί το ύψος του δέντρου είναι k). Άρα $\mathcal{O}(nlogk)$.

Άσκηση 2α: Αλγόριθμος Ταξινόμησης (1)

- Κάνε *QuickSort* μέχρι το επίπεδο k με ντετερμινιστική *Partition*, βάσει του *median*.
- Αφού το k είναι πολλαπλάσιο του 2 κάποια στιγμή θα καταλήξουμε σε κομμάτια πίνακα μεγέθους $\frac{n}{2^z}$, όπου $k = 2^z$.
- Η πολυπλοκότητα είναι $\mathcal{O}(n)$ (από την *Partition* και *Median*) $\cdot$ $\mathcal{O}(logk)$ (γιατί το ύψος του δέντρου είναι k). Άρα $\mathcal{O}(nlogk)$.



Άσκηση 2α: Απόδειξη Κάτω Φράγματος (1)

Άσκηση 2α: Απόδειξη Κάτω Φράγματος (1)

- Θέλουμε ουσιαστικά να ταξινομήσουμε έναν πίνακα, χώρις όμως να μας νοιάζει η διάταξη των στοιχείων των k υποπινάκων που δημιουργούνται.

Άσκηση 2α: Απόδειξη Κάτω Φράγματος (1)

- Θέλουμε ουσιαστικά να ταξινομήσουμε έναν πίνακα, χώρις όμως να μας νοιάζει η διάταξη των στοιχείων των k υποπινάκων που δημιουργούνται.
- Επομένως το συνολικό πλήθος των φύλλων ενός τέτοιου δέντρου ταξινόμησης είναι: $\frac{n!}{(\frac{n}{k})!^k}$

Άσκηση 2α: Απόδειξη Κάτω Φράγματος (1)

- Θέλουμε ουσιαστικά να ταξινομήσουμε έναν πίνακα, χώρις όμως να μας νοιάζει η διάταξη των στοιχείων των k υποπινάκων που δημιουργούνται.
- Επομένως το συνολικό πλήθος των φύλλων ενός τέτοιου δέντρου ταξινόμησης είναι: $\frac{n!}{(\frac{n}{k})!^k}$
- Άρα το ζητούμενο κάτω φράγμα είναι:

$$\begin{aligned} log(\frac{n!}{(\frac{n}{k})!^k}) &= logn! - klog(\frac{n}{k})! \\ &= \Theta(nlogn) - k\Theta(\frac{n}{k}log(\frac{n}{k})) \\ &= \Theta(nlogn) - \Theta(nlog(\frac{n}{k})) = \Theta(nlogk) \end{aligned}$$

Άσκηση 2α: Ερώτημα 2

Άσκηση 2α: Ερώτημα 2

Πρόβλημα (1)

Άσκηση 2α: Ερώτημα 2

Πρόβλημα (1)

- **Είσοδος:** Δίνεται πίνακας $A[n]$ ταξινομημένος κατά k μέρη.

Άσκηση 2α: Ερώτημα 2

Πρόβλημα (1)

- **Είσοδος**: Δίνεται πίνακας $A[n]$ ταξινομημένος κατά k μέρη.
- **Ζητούμενο**: Ταξινομήστε τον πλήρως σε χρόνο $\mathcal{O}(nlog\frac{n}{k})$.

Άσκηση 2α: Ερώτημα 2

Πρόβλημα (1)

- **Είσοδος**: Δίνεται πίνακας $A[n]$ ταξινομημένος κατά k μέρη.
- **Ζητούμενο**: Ταξινομήστε τον πλήρως σε χρόνο $\mathcal{O}(nlog\frac{n}{k})$.

k ανεξάρτητες ταξινομήσεις άρα, $k\Theta(\frac{n}{k}log(\frac{n}{k})) = \Theta(nlog\frac{n}{k})$.

Απόδειξη Κάτω Φράγματος

Άσκηση 2α: Ερώτημα 2

Πρόβλημα (1)

- **Είσοδος:** Δίνεται πίνακας $A[n]$ ταξινομημένος κατά k μέρη.
- **Ζητούμενο:** Ταξινομήστε τον πλήρως σε χρόνο $\mathcal{O}(nlog\frac{n}{k})$.

k ανεξάρτητες ταξινομήσεις άρα, $k\Theta(\frac{n}{k}log(\frac{n}{k})) = \Theta(nlog\frac{n}{k})$.

Απόδειξη Κάτω Φράγματος

- Έχουμε σαν είσοδο k ανεξάρτητες ομαδες των $\frac{n}{k}$ στοιχείων.

Άσκηση 2α: Ερώτημα 2

Πρόβλημα (1)

- **Είσοδος:** Δίνεται πίνακας $A[n]$ ταξινομημένος κατά k μέρη.
- **Ζητούμενο:** Ταξινομήστε τον πλήρως σε χρόνο $\mathcal{O}(nlog\frac{n}{k})$.

k ανεξάρτητες ταξινομήσεις άρα, $k\Theta(\frac{n}{k}log(\frac{n}{k})) = \Theta(nlog\frac{n}{k})$.

Απόδειξη Κάτω Φράγματος

- Έχουμε σαν είσοδο k ανεξάρτητες ομαδες των $\frac{n}{k}$ στοιχείων.
- Συνεπώς όλες οι δυνατές διατάξεις στα φύλλα είναι $(\frac{n}{k})!^k$.

Άσκηση 2α: Ερώτημα 2

Πρόβλημα (1)

- **Είσοδος:** Δίνεται πίνακας $A[n]$ ταξινομημένος κατά k μέρη.
- **Ζητούμενο:** Ταξινομήστε τον πλήρως σε χρόνο $\mathcal{O}(nlog\frac{n}{k})$.

k ανεξάρτητες ταξινομήσεις άρα, $k\Theta(\frac{n}{k}log(\frac{n}{k})) = \Theta(nlog\frac{n}{k})$.

Απόδειξη Κάτω Φράγματος

- Έχουμε σαν είσοδο k ανεξάρτητες ομαδες των $\frac{n}{k}$ στοιχείων.
- Συνεπώς όλες οι δυνατές διατάξεις στα φύλλα είναι $(\frac{n}{k})!^k$.
- Άρα το κάτω φράγμα του προβλήματων είναι $log(\frac{n}{k})!^k = \Theta(nlog\frac{n}{k})$.

Άσκηση 2β: k-Προταξινομημένος Πίνακας

Ορισμός

Ένας πίνακας $A[n]$ καλείται k-προταξινομημένος αν κάθε στοιχείο του απέχει το πολύ k θέσεις από τη θέση που θα κατείχε αν ταξινομούσαμε τον πίνακα.

Άσκηση 2β: k-Προταξινομημένος Πίνακας

Ορισμός

Ένας πίνακας $A[n]$ καλείται k-προταξινομημένος αν κάθε στοιχείο του απέχει το πολύ k θέσεις από τη θέση που θα κατείχε αν ταξινομούσαμε τον πίνακα.

Πρόβλημα (2)

Άσκηση 2β: k-Προταξινομημένος Πίνακας

Ορισμός

Ένας πίνακας $A[n]$ καλείται k-προταξινομημένος αν κάθε στοιχείο του απέχει το πολύ k θέσεις από τη θέση που θα κατείχε αν ταξινομούσαμε τον πίνακα.

Πρόβλημα (2)

- **Είσοδος:** k-Προταξινομημένος Πίνακας $A[n]$.

Άσκηση 2β: k-Προταξινομημένος Πίνακας

Ορισμός

Ένας πίνακας $A[n]$ καλείται k-προταξινομημένος αν κάθε στοιχείο του απέχει το πολύ k θέσεις από τη θέση που θα κατείχε αν ταξινομούσαμε τον πίνακα.

Πρόβλημα (2)

- **Είσοδος**: k-Προταξινομημένος Πίνακας $A[n]$.
- **Ζητούμενο**: Αλγόριθμος ταξινόμησης του A με χρόνο $\mathcal{O}(nlogk)$.

Άσκηση 2β: k-Προταξινομημένος Πίνακας

Ορισμός

Ένας πίνακας $A[n]$ καλείται k-προταξινομημένος αν κάθε στοιχείο του απέχει το πολύ k θέσεις από τη θέση που θα κατείχε αν ταξινομούσαμε τον πίνακα.

Πρόβλημα (2)

- **Είσοδος:** k-Προταξινομημένος Πίνακας $A[n]$.
- **Ζητούμενο:** Αλγόριθμος ταξινόμησης του A με χρόνο $\mathcal{O}(nlogk)$.

Θα παρουσιάσουμε δυο εναλλακτικούς αλγορίθμους.

Άσκηση 2β: Πρώτη Λύση

Αλγόριθμος 1

Άσκηση 2β: Πρώτη Λύση

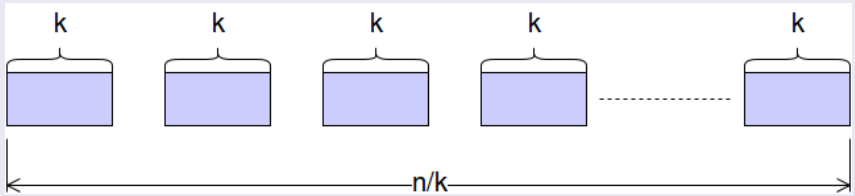
Αλγόριθμος 1

- Χώρισε τον A σε $\frac{n}{k}$ μέρη μεγέθους k.

Άσκηση 2β: Πρώτη Λύση

Αλγόριθμος 1

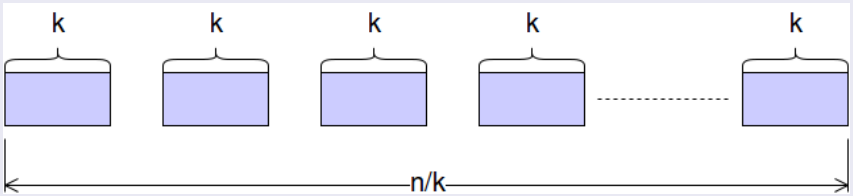
- Χώρισε τον A σε $\frac{n}{k}$ μέρη μεγέθους k.



Άσκηση 2β: Πρώτη Λύση

Αλγόριθμος 1

- Χώρισε τον A σε $\frac{n}{k}$ μέρη μεγέθους k.

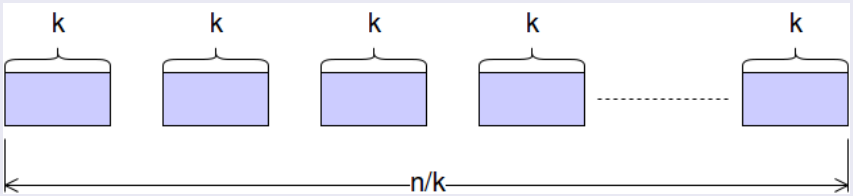


- Ταξινόμησε κάθε τέτοιος μέρος ανεξάρτητα και μάζεψε τα κομμάτια σε άρτιες θέσεις σε μία στοίβα X, και αυτά περιττών θέσεων σε μία Y.

Άσκηση 2β: Πρώτη Λύση

Αλγόριθμος 1

- Χώρισε τον A σε $\frac{n}{k}$ μέρη μεγέθους k.

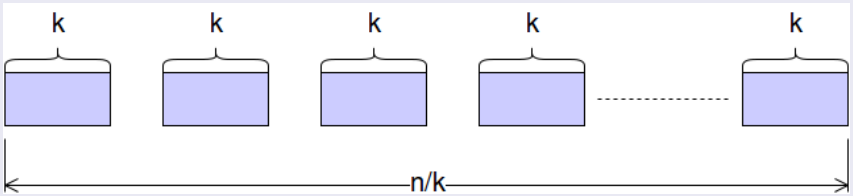


- Ταξινόμησε κάθε τέτοιος μέρος ανεξάρτητα και μάζεψε τα κομμάτια σε άρτιες θέσεις σε μία στοίβα X, και αυτά περιττών θέσεων σε μία Y.
- Χρησιμοποίησε την *Merge* για τις X, Y.

Άσκηση 2β: Πρώτη Λύση

Αλγόριθμος 1

- Χώρισε τον A σε $\frac{n}{k}$ μέρη μεγέθους k.



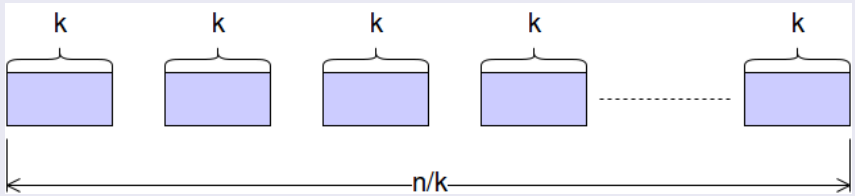
- Ταξινόμησε κάθε τέτοιος μέρος ανεξάρτητα και μάζεψε τα κομμάτια σε άρτιες θέσεις σε μία στοίβα X, και αυτά περιττών θέσεων σε μία Y.
- Χρησιμοποίησε την *Merge* για τις X, Y.

- Πολυπλοκότητα: $\frac{n}{k} \cdot klogk + n = \mathcal{O}(nlogk)$

Άσκηση 2β: Πρώτη Λύση

Αλγόριθμος 1

- Χώρισε τον A σε $\frac{n}{k}$ μέρη μεγέθους k.



- Ταξινόμησε κάθε τέτοιος μέρος ανεξάρτητα και μάζεψε τα κομμάτια σε άρτιες θέσεις σε μία στοίβα X, και αυτά περιττών θέσεων σε μία Y.
- Χρησιμοποίησε την *Merge* για τις X, Y.

- Πολυπλοκότητα: $\frac{n}{k} \cdot klogk + n = \mathcal{O}(nlogk)$
- Ορθότητα: Λόγω της k-προταξινόμησης, οι X, Y, είναι ταξινομημένες.

Άσκηση 2β: Δεύτερη Λύση

Αλγόριθμος 2

Άσκηση 2β: Δεύτερη Λύση

Αλγόριθμος 2

- Φτιάξε ένα σωρό μεγέθους k και βάλε εκεί τα πρώτα k στοιχεία του A.

Άσκηση 2β: Δεύτερη Λύση

Αλγόριθμος 2

- Φτιάξε ένα σωρό μεγέθους k και βάλε εκεί τα πρώτα k στοιχεία του A.
- Τράβα από το σωρό το μικρότερο στοιχείο και τοποθέτησέ το στον πίνακα.

Άσκηση 2β: Δεύτερη Λύση

Αλγόριθμος 2

- Φτιάξε ένα σωρό μεγέθους k και βάλε εκεί τα πρώτα k στοιχεία του A.
- Τράβα από το σωρό το μικρότερο στοιχείο και τοποθέτησέ το στον πίνακα.
- Βάλε στο σωρό το αμέσως επόμενο στοιχείο του πίνακα και συνέχισε αυτή τη διαδικασία μέχρι να εξαντληθεί όλος ο A.

Άσκηση 2β: Δεύτερη Λύση

Αλγόριθμος 2

- Φτιάξε ένα σωρό μεγέθους k και βάλε εκεί τα πρώτα k στοιχεία του A.
- Τράβα από το σωρό το μικρότερο στοιχείο και τοποθέτησέ το στον πίνακα.
- Βάλε στο σωρό το αμέσως επόμενο στοιχείο του πίνακα και συνέχισε αυτή τη διαδικασία μέχρι να εξαντληθεί όλος ο A.

- Πολυπλοκότητα: $k + n(2logk) = \mathcal{O}(nlogk)$

Άσκηση 2β: Δεύτερη Λύση

Αλγόριθμος 2

- Φτιάξε ένα σωρό μεγέθους k και βάλε εκεί τα πρώτα k στοιχεία του A.
- Τράβα από το σωρό το μικρότερο στοιχείο και τοποθέτησέ το στον πίνακα.
- Βάλε στο σωρό το αμέσως επόμενο στοιχείο του πίνακα και συνέχισε αυτή τη διαδικασία μέχρι να εξαντληθεί όλος ο A.

- Πολυπλοκότητα: $k + n(2logk) = \mathcal{O}(nlogk)$
- Ορθότητα: Λόγω της k-προταξινόμησης, το μικρότερο στοιχείο που τραβάμε κάθε φορά από το σωρό θα τοποθετηθεί στη σωστή θέση στον ταξινομημένο πίνακα.

Άσκηση 2β: Απόδειξη Κάτω Φράγματος

Άσκηση 2β: Απόδειξη Κάτω Φράγματος

- Με βάση των διαχωρισμό σε k-άδες της πρώτης λύσης, στη χειρότερη περίπτωση τα μισά στοιχεία της k-άδας βρίσκονται μέσα της και τα άλλα μισά σε κάποια διπλανή.

Άσκηση 2β: Απόδειξη Κάτω Φράγματος

- Με βάση των διαχωρισμό σε k-άδες της πρώτης λύσης, στη χειρότερη περίπτωση τα μισά στοιχεία της k-άδας βρίσκονται μέσα της και τα άλλα μισά σε κάποια διπλανή.
- Επομένως τα φύλλα του δέντρου ταξινόμησης είναι $(\frac{k}{2}!)^{\frac{n}{k}} \cdot c$, όπου c το πλήθος των επιλογών για στοιχεία που πρέπει να αλλάξουν k-άδα.

Άσκηση 2β: Απόδειξη Κάτω Φράγματος

- Με βάση των διαχωρισμό σε k-άδες της πρώτης λύσης, στη χειρότερη περίπτωση τα μισά στοιχεία της k-άδας βρίσκονται μέσα της και τα άλλα μισά σε κάποια διπλανή.
- Επομένως τα φύλλα του δέντρου ταξινόμησης είναι $(\frac{k}{2}!)^{\frac{n}{k}} \cdot c$, όπου c το πλήθος των επιλογών για στοιχεία που πρέπει να αλλάξουν k-άδα.
- Άρα: $log((\frac{k}{2}!)^{\frac{n}{k}} \cdot c) = \Theta(nlogk) + logc$.

Άσκηση 2β: Απόδειξη Κάτω Φράγματος

- Με βάση των διαχωρισμό σε k-άδες της πρώτης λύσης, στη χειρότερη περίπτωση τα μισά στοιχεία της k-άδας βρίσκονται μέσα της και τα άλλα μισά σε κάποια διπλανή.
- Επομένως τα φύλλα του δέντρου ταξινόμησης είναι $(\frac{k}{2}!)^{\frac{n}{k}} \cdot c$, όπου c το πλήθος των επιλογών για στοιχεία που πρέπει να αλλάξουν k-άδα.
- Άρα: $log((\frac{k}{2}!)^{\frac{n}{k}} \cdot c) = \Theta(nlogk) + logc$.
- Εμείς όμως βρήκαμε ήδη αλγόριθμο χρόνου $\mathcal{O}(nlogk)$ πράγμα που σημαίνει ότι το $logc$ είναι αμελητέο και το φράγμα είναι $\Theta(nlogk)$.

Άσκηση 3: Περιστραμμένη Ταξινομημένη Στοίβα

Πρόβλημα

Άσκηση 3: Περιστραμμένη Ταξινομημένη Στοίβα

Πρόβλημα

- **Είσοδος**: Αρχικά έχουμε ταξινομημένο πίνακα $a_1 < a_2 < \ldots < a_n$ που περιστρέφεται στο σημείο k, $a_{k+1}, a_{k+2}, \ldots, a_n, a_1, a_2, \ldots, a_k$.

Άσκηση 3: Περιστραμμένη Ταξινομημένη Στοίβα

Πρόβλημα

- **Είσοδος**: Αρχικά έχουμε ταξινομημένο πίνακα $a_1 < a_2 < \ldots < a_n$ που περιστρέφεται στο σημείο k, $a_{k+1}, a_{k+2}, \ldots, a_n, a_1, a_2, \ldots, a_k$.
- **Ζητούμενο**: Πρέπει να βρούμε αποδοτικά το a_n.

Άσκηση 3: Περιστραμμένη Ταξινομημένη Στοίβα

Πρόβλημα

- **Είσοδος**: Αρχικά έχουμε ταξινομημένο πίνακα $a_1 < a_2 < \ldots < a_n$ που περιστρέφεται στο σημείο k, $a_{k+1}, a_{k+2}, \ldots, a_n, a_1, a_2, \ldots, a_k$.
- **Ζητούμενο**: Πρέπει να βρούμε αποδοτικά το a_n.

Ψάχνουμε αλγόριθμο πολυπλοκότητας $\mathcal{O}(logn)$ (δυαδική αναζήτηση).

Άσκηση 3: Περιστραμμένη Ταξινομημένη Στοίβα

Πρόβλημα

- **Είσοδος**: Αρχικά έχουμε ταξινομημένο πίνακα $a_1 < a_2 < \ldots < a_n$ που περιστρέφεται στο σημείο k, $a_{k+1}, a_{k+2}, \ldots, a_n, a_1, a_2, \ldots, a_k$.
- **Ζητούμενο**: Πρέπει να βρούμε αποδοτικά το a_n.

Ψάχνουμε αλγόριθμο πολυπλοκότητας $\mathcal{O}(logn)$ (δυαδική αναζήτηση).

Αλγόριθμος

Άσκηση 3: Περιστραμμένη Ταξινομημένη Στοίβα

Πρόβλημα

- **Είσοδος**: Αρχικά έχουμε ταξινομημένο πίνακα $a_1 < a_2 < \ldots < a_n$ που περιστρέφεται στο σημείο k, $a_{k+1}, a_{k+2}, \ldots, a_n, a_1, a_2, \ldots, a_k$.
- **Ζητούμενο**: Πρέπει να βρούμε αποδοτικά το a_n.

Ψάχνουμε αλγόριθμο πολυπλοκότητας $\mathcal{O}(logn)$ (δυαδική αναζήτηση).

Αλγόριθμος

- Έλεγχε ζευγάρια διαδοχικών αριθμών a_i, a_{i+1}, ξεκινώντας από το μεσαίο ζευγάρι του πίνακα.

Άσκηση 3: Περιστραμμένη Ταξινομημένη Στοίβα

Πρόβλημα

- **Είσοδος**: Αρχικά έχουμε ταξινομημένο πίνακα $a_1 < a_2 < \ldots < a_n$ που περιστρέφεται στο σημείο k, $a_{k+1}, a_{k+2}, \ldots, a_n, a_1, a_2, \ldots, a_k$.
- **Ζητούμενο**: Πρέπει να βρούμε αποδοτικά το a_n.

Ψάχνουμε αλγόριθμο πολυπλοκότητας $\mathcal{O}(logn)$ (δυαδική αναζήτηση).

Αλγόριθμος

- Έλεγχε ζευγάρια διαδοχικών αριθμών a_i, a_{i+1}, ξεκινώντας από το μεσαίο ζευγάρι του πίνακα.
- Αν $a_i < a_{i+1} < a_k$ το ζητούμενο στοιχείο βρίσκεται αριστερά αυτού του ζευγαριού. Συνέχισε την αναζήτηση σου σε αυτό το κομμάτι.

Άσκηση 3: Περιστραμμένη Ταξινομημένη Στοίβα

Πρόβλημα

- **Είσοδος**: Αρχικά έχουμε ταξινομημένο πίνακα $a_1 < a_2 < \ldots < a_n$ που περιστρέφεται στο σημείο k, $a_{k+1}, a_{k+2}, \ldots, a_n, a_1, a_2, \ldots, a_k$.
- **Ζητούμενο**: Πρέπει να βρούμε αποδοτικά το a_n.

Ψάχνουμε αλγόριθμο πολυπλοκότητας $\mathcal{O}(logn)$ (δυαδική αναζήτηση).

Αλγόριθμος

- Έλεγχε ζευγάρια διαδοχικών αριθμών a_i, a_{i+1}, ξεκινώντας από το μεσαίο ζευγάρι του πίνακα.
- Αν $a_i < a_{i+1} < a_k$ το ζητούμενο στοιχείο βρίσκεται αριστερά αυτού του ζευγαριού. Συνέχισε την αναζήτηση σου σε αυτό το κομμάτι.
- Αν $a_k < a_i < a_{i+1}$ το ζητούμενο στοιχείο βρίσκεται δεξιά αυτού του ζευγαριού. Συνέχισε την αναζήτηση σου σε αυτό το κομμάτι.

Άσκηση 3: Περιστραμμένη Ταξινομημένη Στοίβα

Πρόβλημα

- **Είσοδος**: Αρχικά έχουμε ταξινομημένο πίνακα $a_1 < a_2 < \ldots < a_n$ που περιστρέφεται στο σημείο k, $a_{k+1}, a_{k+2}, \ldots, a_n, a_1, a_2, \ldots, a_k$.
- **Ζητούμενο**: Πρέπει να βρούμε αποδοτικά το a_n.

Ψάχνουμε αλγόριθμο πολυπλοκότητας $\mathcal{O}(logn)$ (δυαδική αναζήτηση).

Αλγόριθμος

- Έλεγχε ζευγάρια διαδοχικών αριθμών a_i, a_{i+1}, ξεκινώντας από το μεσαίο ζευγάρι του πίνακα.
- Αν $a_i < a_{i+1} < a_k$ το ζητούμενο στοιχείο βρίσκεται αριστερά αυτού του ζευγαριού. Συνέχισε την αναζήτηση σου σε αυτό το κομμάτι.
- Αν $a_k < a_i < a_{i+1}$ το ζητούμενο στοιχείο βρίσκεται δεξιά αυτού του ζευγαριού. Συνέχισε την αναζήτηση σου σε αυτό το κομμάτι.
- Αν $a_i < a_k < a_{i+1}$, το μεγαλύτερο είναι και το ζητούμενο.

Άσκηση 3α: Πολυπλοκότητα και Ορθότητα

Άσκηση 3α: Πολυπλοκότητα και Ορθότητα

- Η χρονική πολυπλοκότητα του αλγορίθμου είναι $\mathcal{O}(logn)$, επειδή σε κάθε στάδιο της αναζήτησης το μέγεθος του πίνακα που εξετάζεται υποδιπλασιάζεται.

Άσκηση 3α: Πολυπλοκότητα και Ορθότητα

- Η χρονική πολυπλοκότητα του αλγορίθμου είναι $\mathcal{O}(logn)$, επειδή σε κάθε στάδιο της αναζήτησης το μέγεθος του πίνακα που εξετάζεται υποδιπλασιάζεται.
- Ορθότητα:

Άσκηση 3α: Πολυπλοκότητα και Ορθότητα

- Η χρονική πολυπλοκότητα του αλγορίθμου είναι $\mathcal{O}(logn)$, επειδή σε κάθε στάδιο της αναζήτησης το μέγεθος του πίνακα που εξετάζεται υποδιπλασιάζεται.
- Ορθότητα:
  1. Αν $a_i < a_{i+1} < a_k$, λόγω της αρχικής ταξινόμησης και των διαδοχικών εξεταζόμενων στοιχείων το a_n δεν μπορεί παρά να είναι στο αριστερό μέρος.

Άσκηση 3α: Πολυπλοκότητα και Ορθότητα

- Η χρονική πολυπλοκότητα του αλγορίθμου είναι $\mathcal{O}(logn)$, επειδή σε κάθε στάδιο της αναζήτησης το μέγεθος του πίνακα που εξετάζεται υποδιπλασιάζεται.
- Ορθότητα:
  1. Αν $a_i < a_{i+1} < a_k$, λόγω της αρχικής ταξινόμησης και των διαδοχικών εξεταζόμενων στοιχείων το a_n δεν μπορεί παρά να είναι στο αριστερό μέρος.
  2. Ομοίως αν $a_k < a_i < a_{i+1}$.

Άσκηση 3α: Πολυπλοκότητα και Ορθότητα

- Η χρονική πολυπλοκότητα του αλγορίθμου είναι $\mathcal{O}(logn)$, επειδή σε κάθε στάδιο της αναζήτησης το μέγεθος του πίνακα που εξετάζεται υποδιπλασιάζεται.
- Ορθότητα:
  1. Αν $a_i < a_{i+1} < a_k$, λόγω της αρχικής ταξινόμησης και των διαδοχικών εξεταζόμενων στοιχείων το a_n δεν μπορεί παρά να είναι στο αριστερό μέρος.
  2. Ομοίως αν $a_k < a_i < a_{i+1}$.
  3. Αν βρούμε ζεύγος a_i, a_{i+1} τέτοιο ώστε $a_i < a_k < a_{i+1}$ τότε $a_{i+1} = a_n$ και πάλι λόγω της αρχικής ταξινόμησης.

Άσκηση 3β: Σταθμοί Ταξιδιού

Πρόβλημα

Άσκηση 3β: Σταθμοί Ταξιδιού

Πρόβλημα

- **Είσοδος**: Έχουμε $k \geq 2$ μέρες ταξιδιού και $n > k$ ενδιάμεσους σταθμούς με μεταξύ τους αποστάσεις $d_1, d_2, \ldots, d_n$.

Άσκηση 3β: Σταθμοί Ταξιδιού

Πρόβλημα

- **Είσοδος**: Έχουμε $k \geq 2$ μέρες ταξιδιού και $n > k$ ενδιάμεσους σταθμούς με μεταξύ τους αποστάσεις $d_1, d_2, \ldots, d_n$.
- **Ζητούμενο**: Να βρούμε πρόγραμμα ταξιδιού όπου διανύουμε την ελάχιστη μέγιστη ημερήσια απόσταση.

Άσκηση 3β: Σταθμοί Ταξιδιού

Πρόβλημα

- **Είσοδος**: Έχουμε $k \geq 2$ μέρες ταξιδιού και $n > k$ ενδιάμεσους σταθμούς με μεταξύ τους αποστάσεις $d_1, d_2, \ldots, d_n$.
- **Ζητούμενο**: Να βρούμε πρόγραμμα ταξιδιού όπου διανύουμε την ελάχιστη μέγιστη ημερήσια απόσταση.

Το πρόβλημα λύνεται με δυαδική αναζήτηση στο $\sum_{i=1}^{n} d_i$ με χρήση μαντείου.

Άσκηση 3β: Σταθμοί Ταξιδιού

Πρόβλημα

- **Είσοδος**: Έχουμε $k \geq 2$ μέρες ταξιδιού και $n > k$ ενδιάμεσους σταθμούς με μεταξύ τους αποστάσεις $d_1, d_2, \ldots, d_n$.
- **Ζητούμενο**: Να βρούμε πρόγραμμα ταξιδιού όπου διανύουμε την ελάχιστη μέγιστη ημερήσια απόσταση.

Το πρόβλημα λύνεται με δυαδική αναζήτηση στο $\sum_{i=1}^{n} d_i$ με χρήση μαντείου.

Αλγόριθμος

Άσκηση 3β: Σταθμοί Ταξιδιού

Πρόβλημα

- **Είσοδος**: Έχουμε $k \geq 2$ μέρες ταξιδιού και $n > k$ ενδιάμεσους σταθμούς με μεταξύ τους αποστάσεις $d_1, d_2, \ldots, d_n$.
- **Ζητούμενο**: Να βρούμε πρόγραμμα ταξιδιού όπου διανύουμε την ελάχιστη μέγιστη ημερήσια απόσταση.

Το πρόβλημα λύνεται με δυαδική αναζήτηση στο $\sum_{i=1}^{n} d_i$ με χρήση μαντείου.

Αλγόριθμος

- Έστω διάστημα εξεταζόμενων λύσεων $[x, y]$ όπου αρχικά $x = 0$ και $y = \sum_{i=1}^{n} d_i$.

Άσκηση 3β: Σταθμοί Ταξιδιού

Πρόβλημα

- **Είσοδος**: Έχουμε $k \geq 2$ μέρες ταξιδιού και $n > k$ ενδιάμεσους σταθμούς με μεταξύ τους αποστάσεις $d_1, d_2, \ldots, d_n$.
- **Ζητούμενο**: Να βρούμε πρόγραμμα ταξιδιού όπου διανύουμε την ελάχιστη μέγιστη ημερήσια απόσταση.

Το πρόβλημα λύνεται με δυαδική αναζήτηση στο $\sum_{i=1}^{n} d_i$ με χρήση μαντείου.

Αλγόριθμος

- Έστω διάστημα εξεταζόμενων λύσεων $[x, y]$ όπου αρχικά $x = 0$ και $y = \sum_{i=1}^{n} d_i$.
- Έστω απόσταση στόχος $d = \left\lfloor \frac{x+y}{2} \right\rfloor$.

Άσκηση 3β: Σταθμοί Ταξιδιού

Πρόβλημα

- **Είσοδος**: Έχουμε $k \geq 2$ μέρες ταξιδιού και $n > k$ ενδιάμεσους σταθμούς με μεταξύ τους αποστάσεις $d_1, d_2, \ldots, d_n$.
- **Ζητούμενο**: Να βρούμε πρόγραμμα ταξιδιού όπου διανύουμε την ελάχιστη μέγιστη ημερήσια απόσταση.

Το πρόβλημα λύνεται με δυαδική αναζήτηση στο $\sum_{i=1}^{n} d_i$ με χρήση μαντείου.

Αλγόριθμος

- Έστω διάστημα εξεταζόμενων λύσεων $[x, y]$ όπου αρχικά $x = 0$ και $y = \sum_{i=1}^{n} d_i$.
- Έστω απόσταση στόχος $d = \left\lfloor \frac{x+y}{2} \right\rfloor$.
- Αν υπάρχει εφικτή λύση για d και $d - 1$ τότε $y \leftarrow d$.

Άσκηση 3β: Σταθμοί Ταξιδιού

Πρόβλημα

- **Είσοδος**: Έχουμε $k \geq 2$ μέρες ταξιδιού και $n > k$ ενδιάμεσους σταθμούς με μεταξύ τους αποστάσεις $d_1, d_2, \ldots, d_n$.
- **Ζητούμενο**: Να βρούμε πρόγραμμα ταξιδιού όπου διανύουμε την ελάχιστη μέγιστη ημερήσια απόσταση.

Το πρόβλημα λύνεται με δυαδική αναζήτηση στο $\sum_{i=1}^{n} d_i$ με χρήση μαντείου.

Αλγόριθμος

- Έστω διάστημα εξεταζόμενων λύσεων $[x, y]$ όπου αρχικά $x = 0$ και $y = \sum_{i=1}^{n} d_i$.
- Έστω απόσταση στόχος $d = \left\lfloor \frac{x+y}{2} \right\rfloor$.
- Αν υπάρχει εφικτή λύση για d και $d-1$ τότε $y \leftarrow d$.
- Αν δεν υπάρχει εφικτή λύση για d τότε $x \leftarrow d+1$.

Άσκηση 3β: Σταθμοί Ταξιδιού

Πρόβλημα

- **Είσοδος**: Έχουμε $k \geq 2$ μέρες ταξιδιού και $n > k$ ενδιάμεσους σταθμούς με μεταξύ τους αποστάσεις $d_1, d_2, \ldots, d_n$.
- **Ζητούμενο**: Να βρούμε πρόγραμμα ταξιδιού όπου διανύουμε την ελάχιστη μέγιστη ημερήσια απόσταση.

Το πρόβλημα λύνεται με δυαδική αναζήτηση στο $\sum_{i=1}^{n} d_i$ με χρήση μαντείου.

Αλγόριθμος

- Έστω διάστημα εξεταζόμενων λύσεων $[x, y]$ όπου αρχικά $x = 0$ και $y = \sum_{i=1}^{n} d_i$.
- Έστω απόσταση στόχος $d = \left\lfloor \frac{x+y}{2} \right\rfloor$.
- Αν υπάρχει εφικτή λύση για d και $d-1$ τότε $y \leftarrow d$.
- Αν δεν υπάρχει εφικτή λύση για d τότε $x \leftarrow d+1$.
- Αν υπάρχει εφικτή λύση για d αλλά όχι για το $d-1$, επέστρεψε d.

Άσκηση 3β: Μαντείο, Ορθότητα, Πολυπλοκότητα

Άσκηση 3β: Μαντείο, Ορθότητα, Πολυπλοκότητα

Μαντείο

Άσκηση 3β: Μαντείο, Ορθότητα, Πολυπλοκότητα

Μαντείο

- Δεδομένου d υπάρχει εφικτή λύση;

Άσκηση 3β: Μαντείο, Ορθότητα, Πολυπλοκότητα

Μαντείο

- Δεδομένου d υπάρχει εφικτή λύση;
- Με βάση την απόσταση d προσπάθησε να χωρέσεις όσους περισσότερους σταθμούς γίνεται σε μια μέρα.

Άσκηση 3β: Μαντείο, Ορθότητα, Πολυπλοκότητα

Μαντείο

- Δεδομένου d υπάρχει εφικτή λύση;
- Με βάση την απόσταση d προσπάθησε να χωρέσεις όσους περισσότερους σταθμούς γίνεται σε μια μέρα.
- Αν δεν σου φτάσουν οι k μέρες δεν υπάρχει εφικτή λύση για το d.

Άσκηση 3β: Μαντείο, Ορθότητα, Πολυπλοκότητα

Μαντείο

- Δεδομένου d υπάρχει εφικτή λύση;
- Με βάση την απόσταση d προσπάθησε να χωρέσεις όσους περισσότερους σταθμούς γίνεται σε μια μέρα.
- Αν δεν σου φτάσουν οι k μέρες δεν υπάρχει εφικτή λύση για το d.

- Η χρονική πολυπλοκότητα του αλγορίθμου είναι $\mathcal{O}(nlog\sum_{i=1}^{n} d_i)$.

Άσκηση 3β: Μαντείο, Ορθότητα, Πολυπλοκότητα

Μαντείο

- Δεδομένου d υπάρχει εφικτή λύση;
- Με βάση την απόσταση d προσπάθησε να χωρέσεις όσους περισσότερους σταθμούς γίνεται σε μια μέρα.
- Αν δεν σου φτάσουν οι k μέρες δεν υπάρχει εφικτή λύση για το d.

- Η χρονική πολυπλοκότητα του αλγορίθμου είναι $\mathcal{O}(nlog\sum_{i=1}^{n} d_i)$.
- Ορθότητα:

Άσκηση 3β: Μαντείο, Ορθότητα, Πολυπλοκότητα

Μαντείο

- Δεδομένου d υπάρχει εφικτή λύση;
- Με βάση την απόσταση d προσπάθησε να χωρέσεις όσους περισσότερους σταθμούς γίνεται σε μια μέρα.
- Αν δεν σου φτάσουν οι k μέρες δεν υπάρχει εφικτή λύση για το d.

- Η χρονική πολυπλοκότητα του αλγορίθμου είναι $\mathcal{O}(nlog\sum_{i=1}^{n} d_i)$.
- Ορθότητα:
  1. Η δυαδική αναζήτηση ψάχνει πλήρως τον χώρο των λύσεων.

Άσκηση 3β: Μαντείο, Ορθότητα, Πολυπλοκότητα

Μαντείο

- Δεδομένου d υπάρχει εφικτή λύση;
- Με βάση την απόσταση d προσπάθησε να χωρέσεις όσους περισσότερους σταθμούς γίνεται σε μια μέρα.
- Αν δεν σου φτάσουν οι k μέρες δεν υπάρχει εφικτή λύση για το d.

- Η χρονική πολυπλοκότητα του αλγορίθμου είναι $\mathcal{O}(nlog\sum_{i=1}^{n} d_i)$.
- Ορθότητα:
  1. Η δυαδική αναζήτηση ψάχνει πλήρως τον χώρο των λύσεων.
  2. Η ορθότητα του μαντείου έγκειται στο ότι για μια μέρα γεμίζεις όσο περισσότερο γίνεται το πρόγραμμα.

Άσκηση 4: Εντοπισμός Επαναλαμβανόμενου Στοιχείου

Πρόβλημα

Άσκηση 4: Εντοπισμός Επαναλαμβανόμενου Στοιχείου

Πρόβλημα

- **Είσοδος:** Έχουμε πίνακα $A[n]$ με αριθμούς από το $[n] = \{1, 2, \ldots, n\}$. Ο $k \in [n]$ εμφανίζεται τουλάχιστον 2 φορές, ενώ κάθε άλλος το πολύ μια.

Άσκηση 4: Εντοπισμός Επαναλαμβανόμενου Στοιχείου

Πρόβλημα

- **Είσοδος**: Έχουμε πίνακα $A[n]$ με αριθμούς από το $[n] = \{1, 2, \ldots, n\}$. Ο $k \in [n]$ εμφανίζεται τουλάχιστον 2 φορές, ενώ κάθε άλλος το πολύ μια.
- **Ζητούμενο**: Να βρούμε τον k χρησιμοποιώντας επιπλέον χώρο $\mathcal{O}(1)$.

Αλγόριθμος 1

Άσκηση 4: Εντοπισμός Επαναλαμβανόμενου Στοιχείου

Πρόβλημα

- **Είσοδος**: Έχουμε πίνακα $A[n]$ με αριθμούς από το $[n] = \{1, 2, \ldots, n\}$. Ο $k \in [n]$ εμφανίζεται τουλάχιστον 2 φορές, ενώ κάθε άλλος το πολύ μια.
- **Ζητούμενο**: Να βρούμε τον k χρησιμοποιώντας επιπλέον χώρο $\mathcal{O}(1)$.

Αλγόριθμος 1

- Βρες το ενδιάμεσο στοιχείο του πίνακα, έστω m και χρησιμοποίησε την *Partition* της *Quicksort* με βάση αυτό το m.

Άσκηση 4: Εντοπισμός Επαναλαμβανόμενου Στοιχείου

Πρόβλημα

- **Είσοδος:** Έχουμε πίνακα $A[n]$ με αριθμούς από το $[n] = \{1, 2, \ldots, n\}$. Ο $k \in [n]$ εμφανίζεται τουλάχιστον 2 φορές, ενώ κάθε άλλος το πολύ μια.
- **Ζητούμενο:** Να βρούμε τον k χρησιμοποιώντας επιπλέον χώρο $\mathcal{O}(1)$.

Αλγόριθμος 1

- Βρες το ενδιάμεσο στοιχείο του πίνακα, έστω m και χρησιμοποίησε την *Partition* της *Quicksort* με βάση αυτό το m.
- Αν το m είναι μικρότερο από το ενδιάμεσο στοιχείο του $[1, 2, \ldots, n]$, συνέχισε την αναζητήσή σου αριστερά.

Άσκηση 4: Εντοπισμός Επαναλαμβανόμενου Στοιχείου

Πρόβλημα

- **Είσοδος:** Έχουμε πίνακα $A[n]$ με αριθμούς από το $[n] = \{1, 2, \ldots, n\}$. Ο $k \in [n]$ εμφανίζεται τουλάχιστον 2 φορές, ενώ κάθε άλλος το πολύ μια.
- **Ζητούμενο:** Να βρούμε τον k χρησιμοποιώντας επιπλέον χώρο $\mathcal{O}(1)$.

Αλγόριθμος 1

- Βρες το ενδιάμεσο στοιχείο του πίνακα, έστω m και χρησιμοποίησε την *Partition* της *Quicksort* με βάση αυτό το m.
- Αν το m είναι μικρότερο από το ενδιάμεσο στοιχείο του $[1, 2, \ldots, n]$, συνέχισε την αναζητήσή σου αριστερά.
- Αν το m είναι μεγαλύτερο από το ενδιάμεσο στοιχείο του $[1, 2, \ldots, n]$, συνέχισε την αναζητήσή σου δεξιά.

Άσκηση 4: Εντοπισμός Επαναλαμβανόμενου Στοιχείου

Πρόβλημα

- **Είσοδος:** Έχουμε πίνακα $A[n]$ με αριθμούς από το $[n] = \{1, 2, \ldots, n\}$. Ο $k \in [n]$ εμφανίζεται τουλάχιστον 2 φορές, ενώ κάθε άλλος το πολύ μια.
- **Ζητούμενο:** Να βρούμε τον k χρησιμοποιώντας επιπλέον χώρο $\mathcal{O}(1)$.

Αλγόριθμος 1

- Βρες το ενδιάμεσο στοιχείο του πίνακα, έστω m και χρησιμοποίησε την *Partition* της *Quicksort* με βάση αυτό το m.
- Αν το m είναι μικρότερο από το ενδιάμεσο στοιχείο του $[1, 2, \ldots, n]$, συνέχισε την αναζητήσή σου αριστερά.
- Αν το m είναι μεγαλύτερο από το ενδιάμεσο στοιχείο του $[1, 2, \ldots, n]$, συνέχισε την αναζητήσή σου δεξιά.
- Αν δεν υπάρχει διαφοροποίηση σε σχέση με τον $[1, 2, \ldots, n]$ υπολόγισε το άθροισμα των στοιχείων αριστερά και δεξιά του m και συνεκρινέ τα με αυτά του $[1, 2, \ldots, n]$. Συνέχισε την αναζήτηση εκεί που υπάρχει διαφοροποίηση.

Άσκηση 4: Πολυπλοκότητα και Ορθότητα Πρώτης Λύσης

Πολυπλοκότητα

Άσκηση 4: Πολυπλοκότητα και Ορθότητα Πρώτης Λύσης

Πολυπλοκότητα

- Σε κάθε επανάληψη μειώνουμε το μέγεθος του πίνακα στο μισό.

Άσκηση 4: Πολυπλοκότητα και Ορθότητα Πρώτης Λύσης

Πολυπλοκότητα

- Σε κάθε επανάληψη μειώνουμε το μέγεθος του πίνακα στο μισό.
- Η πολυπλοκότητα της *Partition*, του υπολογισμού του ενδιάμεσου στοιχείου και του υπολογισμού των αθροισμάτων είναι γραμμική στο μέγεθος του μελετούμενου πίνακα.

Άσκηση 4: Πολυπλοκότητα και Ορθότητα Πρώτης Λύσης

Πολυπλοκότητα

- Σε κάθε επανάληψη μειώνουμε το μέγεθος του πίνακα στο μισό.
- Η πολυπλοκότητα της *Partition*, του υπολογισμού του ενδιάμεσου στοιχείου και του υπολογισμού των αθροισμάτων είναι γραμμική στο μέγεθος του μελετούμενου πίνακα.
- Τα αθροίσματα στον $[1, 2, \ldots, n]$ μπορούμε να τα βρούμε από τον τύπο του *Euler* σε $\mathcal{O}(1)$, το ίδιο και τα ενδιάμεσα στοιχεία.

Άσκηση 4: Πολυπλοκότητα και Ορθότητα Πρώτης Λύσης

Πολυπλοκότητα

- Σε κάθε επανάληψη μειώνουμε το μέγεθος του πίνακα στο μισό.
- Η πολυπλοκότητα της *Partition*, του υπολογισμού του ενδιάμεσου στοιχείου και του υπολογισμού των αθροισμάτων είναι γραμμική στο μέγεθος του μελετούμενου πίνακα.
- Τα αθροίσματα στον $[1, 2, \ldots, n]$ μπορούμε να τα βρούμε από τον τύπο του *Euler* σε $\mathcal{O}(1)$, το ίδιο και τα ενδιάμεσα στοιχεία.
- Άρα έχουμε: $n + \frac{n}{2} + \frac{n}{4} + \ldots + 1 = \mathcal{O}(n)$

Ορθότητα

Άσκηση 4: Πολυπλοκότητα και Ορθότητα Πρώτης Λύσης

Πολυπλοκότητα

- Σε κάθε επανάληψη μειώνουμε το μέγεθος του πίνακα στο μισό.
- Η πολυπλοκότητα της *Partition*, του υπολογισμού του ενδιάμεσου στοιχείου και του υπολογισμού των αθροισμάτων είναι γραμμική στο μέγεθος του μελετούμενου πίνακα.
- Τα αθροίσματα στον $[1, 2, \ldots, n]$ μπορούμε να τα βρούμε από τον τύπο του *Euler* σε $\mathcal{O}(1)$, το ίδιο και τα ενδιάμεσα στοιχεία.
- Άρα έχουμε: $n + \frac{n}{2} + \frac{n}{4} + \ldots + 1 = \mathcal{O}(n)$

Ορθότητα

- Απόδειξη στον πίνακα.

Άσκηση 4: Δεύτερη Λύση

Άσκηση 4: Δεύτερη Λύση

Υπάρχει πιο καθαρή και προγραμματιστικά απλή λύση σε χρόνο $\mathcal{O}(n)$.

Αλγόριθμος 2

Άσκηση 4: Δεύτερη Λύση

Υπάρχει πιο καθαρή και προγραμματιστικά απλή λύση σε χρόνο $\mathcal{O}(n)$.

Αλγόριθμος 2

- Διέσχισε γραμμικά τον πίνακα.

Άσκηση 4: Δεύτερη Λύση

Υπάρχει πιο καθαρή και προγραμματιστικά απλή λύση σε χρόνο $\mathcal{O}(n)$.

Αλγόριθμος 2

- Διέσχισε γραμμικά τον πίνακα.
- Όταν βρεθείς στο στοιχείο $A[i]$ κάνε $A[|A[i]|] \leftarrow -A[|A[i]|]$.

Άσκηση 4: Δεύτερη Λύση

Υπάρχει πιο καθαρή και προγραμματιστικά απλή λύση σε χρόνο $\mathcal{O}(n)$.

Αλγόριθμος 2

- Διέσχισε γραμμικά τον πίνακα.
- Όταν βρεθείς στο στοιχείο $A[i]$ κάνε $A[|A[i]|] \leftarrow -A[|A[i]|]$.
- Όταν βρεις $A[|A[i]|] < 0$ τότε ο ζητούμενος αριθμός είναι ο $|A[i]|$.

Άσκηση 5α: Έγκυρα Ζευγάρια Ταξινομημένων Πινάκων

Πρόβλημα

Άσκηση 5α: Έγκυρα Ζευγάρια Ταξινομημένων Πινάκων

Πρόβλημα

- **Είσοδος**: Δίνονται ταξινομημένοι πίνακες $A[n], B[m]$ και αριθμοί L, M με $L < M$.

Άσκηση 5α: Έγκυρα Ζευγάρια Ταξινομημένων Πινάκων

Πρόβλημα

- **Είσοδος**: Δίνονται ταξινομημένοι πίνακες $A[n], B[m]$ και αριθμοί L, M με $L < M$.
- **Ζητούμενο**: Γραμμικός αλγόριθμος που να υπολογίζει το πλήθος των ζευγαριών (i, j), τέτοιων ώστε $L < B[j] - A[i] < M$.

Άσκηση 5α: Έγκυρα Ζευγάρια Ταξινομημένων Πινάκων

Πρόβλημα

- **Είσοδος**: Δίνονται ταξινομημένοι πίνακες $A[n], B[m]$ και αριθμοί L, M με $L < M$.
- **Ζητούμενο**: Γραμμικός αλγόριθμος που να υπολογίζει το πλήθος των ζευγαριών (i, j), τέτοιων ώστε $L < B[j] - A[i] < M$.

- Ο αλγόριθμος μας αφαιρεί από το συνολικό πλήθος ζευγαριών, nm, αυτά για τα οποία ισχύει ότι $L \geq B[j] - A[i]$, έστω n_1 το πλήθος, και αυτά για τα οποία ισχύει $B[j] - A[i] \geq M$, έστω n_2 το πλήθος.

Άσκηση 5α: Έγκυρα Ζευγάρια Ταξινομημένων Πινάκων

Πρόβλημα

- **Είσοδος**: Δίνονται ταξινομημένοι πίνακες $A[n], B[m]$ και αριθμοί L, M με $L < M$.
- **Ζητούμενο**: Γραμμικός αλγόριθμος που να υπολογίζει το πλήθος των ζευγαριών (i, j), τέτοιων ώστε $L < B[j] - A[i] < M$.

- Ο αλγόριθμος μας αφαιρεί από το συνολικό πλήθος ζευγαριών, nm, αυτά για τα οποία ισχύει ότι $L \geq B[j] - A[i]$, έστω n_1 το πλήθος, και αυτά για τα οποία ισχύει $B[j] - A[i] \geq M$, έστω n_2 το πλήθος.
- Επειδή τα τρία σύνολα ζευγαριών που προκύπτουν είναι ξεχωριστά, η αφαίρεση θα δώσει σωστό αποτέλεσμα.

Άσκηση 5α: Έγκυρα Ζευγάρια Ταξινομημένων Πινάκων

Πρόβλημα

- **Είσοδος**: Δίνονται ταξινομημένοι πίνακες $A[n], B[m]$ και αριθμοί L, M με $L < M$.
- **Ζητούμενο**: Γραμμικός αλγόριθμος που να υπολογίζει το πλήθος των ζευγαριών (i, j), τέτοιων ώστε $L < B[j] - A[i] < M$.

- Ο αλγόριθμος μας αφαιρεί από το συνολικό πλήθος ζευγαριών, nm, αυτά για τα οποία ισχύει ότι $L \geq B[j] - A[i]$, έστω n_1 το πλήθος, και αυτά για τα οποία ισχύει $B[j] - A[i] \geq M$, έστω n_2 το πλήθος.
- Επειδή τα τρία σύνολα ζευγαριών που προκύπτουν είναι ξεχωριστά, η αφαίρεση θα δώσει σωστό αποτέλεσμα.
- Θα δείξουμε πως να υπολογίσουμε γραμμικά τα ζευγάρια με $L \geq B[j] - A[i]$. Η άλλη περίπτωση προκύπτει ανάλογα.

Άσκηση 5α: Αλγόριθμος

Άσκηση 5α: Αλγόριθμος

- Έστω ότι και οι δύο πίνακες είναι ταξινομημένοι σε αύξουσα διάταξη.

Άσκηση 5α: Αλγόριθμος

- Έστω ότι και οι δύο πίνακες είναι ταξινομημένοι σε αύξουσα διάταξη.
- Έχουμε δυο δείκτες i, j αρχικοποιημένους στο 1.

Άσκηση 5α: Αλγόριθμος

- Έστω ότι και οι δύο πίνακες είναι ταξινομημένοι σε αύξουσα διάταξη.
- Έχουμε δυο δείκτες i, j αρχικοποιημένους στο 1.
- Αυξάνουμε τον i όσο ισχύει $B[j] - A[i] \geq L$.

Άσκηση 5α: Αλγόριθμος

- Έστω ότι και οι δύο πίνακες είναι ταξινομημένοι σε αύξουσα διάταξη.
- Έχουμε δυο δείκτες i, j αρχικοποιημένους στο 1.
- Αυξάνουμε τον i όσο ισχύει $B[j] - A[i] \geq L$.
- Για όλα αυτά ζευγάρια (i_{first}, j), (i_{final}, j) ισχύει η ζητούμενη σχέση.

Άσκηση 5α: Αλγόριθμος

- Έστω ότι και οι δύο πίνακες είναι ταξινομημένοι σε αύξουσα διάταξη.
- Έχουμε δυο δείκτες i, j αρχικοποιημένους στο 1.
- Αυξάνουμε τον i όσο ισχύει $B[j] - A[i] \geq L$.
- Για όλα αυτά ζευγάρια (i_{first}, j), (i_{final}, j) ισχύει η ζητούμενη σχέση.
- Έπειτα αυξάνω κατά 1 το j. Για το $j+1$ θα έχουμε την σχέση να ισχύει για όλα τα $[i_{first}, i_{final}]$ καθώς $B[j+1] \geq B[j]$.

Άσκηση 5α: Αλγόριθμος

- Έστω ότι και οι δύο πίνακες είναι ταξινομημένοι σε αύξουσα διάταξη.
- Έχουμε δυο δείκτες i, j αρχικοποιημένους στο 1.
- Αυξάνουμε τον i όσο ισχύει $B[j] - A[i] \geq L$.
- Για όλα αυτά ζευγάρια (i_{first}, j), (i_{final}, j) ισχύει η ζητούμενη σχέση.
- Έπειτα αυξάνω κατά 1 το j. Για το $j+1$ θα έχουμε την σχέση να ισχύει για όλα τα $[i_{first}, i_{final}]$ καθώς $B[j+1] \geq B[j]$.
- Μετά συνεχίζουμε να αυξάνουμε το i με την ίδια λογική με *pivot* το $j+1$ για να βρούμε νέο διάστημα $[i'_{first}, i'_{final}]$, όπου $i'_{first} = i_{final} + 1$.

Άσκηση 5α: Αλγόριθμος

- Έστω ότι και οι δύο πίνακες είναι ταξινομημένοι σε αύξουσα διάταξη.
- Έχουμε δυο δείκτες i, j αρχικοποιημένους στο 1.
- Αυξάνουμε τον i όσο ισχύει $B[j] - A[i] \geq L$.
- Για όλα αυτά ζευγάρια (i_{first}, j), (i_{final}, j) ισχύει η ζητούμενη σχέση.
- Έπειτα αυξάνω κατά 1 το j. Για το $j+1$ θα έχουμε την σχέση να ισχύει για όλα τα $[i_{first}, i_{final}]$ καθώς $B[j+1] \geq B[j]$.
- Μετά συνεχίζουμε να αυξάνουμε το i με την ίδια λογική με *pivot* το $j+1$ για να βρούμε νέο διάστημα $[i'_{first}, i'_{final}]$, όπου $i'_{first} = i_{final} + 1$.
- Η παραπάνω διαδικασία κοστίζει $\mathcal{O}(n+m)$, γιατί κάθε δείκτης διασχίζει μια μόνο φορά τον αντίστοιχο πίνακα.

Άσκηση 5β: Έγκυρα Ζευγάρια σε Μη-Ταξινομημένο Πίνακα

Πρόβλημα

Άσκηση 5β: Έγκυρα Ζευγάρια σε Μη-Ταξινομημένο Πίνακα

Πρόβλημα

- **Είσοδος:** Δίνεται μη ταξινομημένος πίνακας $A[n]$ και αριθμοί L, M με $L < M$.

Άσκηση 5β: Έγκυρα Ζευγάρια σε Μη-Ταξινομημένο Πίνακα

Πρόβλημα

- **Είσοδος**: Δίνεται μη ταξινομημένος πίνακας $A[n]$ και αριθμοί L, M με $L < M$.
- **Ζητούμενο**: Αλγόριθμος χρόνου $\mathcal{O}(nlogn)$ που να υπολογίζει το πλήθος των ζευγαριών (i, j), τέτοιων ώστε $L < A[j] - A[i] < M$, $i < j$.

Αλγόριθμος

Άσκηση 5β: Έγκυρα Ζευγάρια σε Μη-Ταξινομημένο Πίνακα

Πρόβλημα

- **Είσοδος**: Δίνεται μη ταξινομημένος πίνακας $A[n]$ και αριθμοί L, M με $L < M$.
- **Ζητούμενο**: Αλγόριθμος χρόνου $\mathcal{O}(nlogn)$ που να υπολογίζει το πλήθος των ζευγαριών (i, j), τέτοιων ώστε $L < A[j] - A[i] < M$, $i < j$.

Αλγόριθμος

- Χρησιμοποιούμε αυτούσια την *MergeSort* με μια διαφορά.

Άσκηση 5β: Έγκυρα Ζευγάρια σε Μη-Ταξινομημένο Πίνακα

Πρόβλημα

- **Είσοδος:** Δίνεται μη ταξινομημένος πίνακας $A[n]$ και αριθμοί L, M με $L < M$.
- **Ζητούμενο:** Αλγόριθμος χρόνου $\mathcal{O}(nlogn)$ που να υπολογίζει το πλήθος των ζευγαριών (i, j), τέτοιων ώστε $L < A[j] - A[i] < M$, $i < j$.

Αλγόριθμος

- Χρησιμοποιούμε αυτούσια την *MergeSort* με μια διαφορά.
- Ακριβώς πριν κάνουμε το *Merge* χρησιμοποιούμε τον αλγόριθμο του 5α με είσοδο τα δυο επιμέρους κομμάτια του πίνακα. Αυτό που παίρνουμε σαν αποτέλεσμα το προσθέτουμε σε ένα μετρητή.

Άσκηση 5β: Πολυπλοκότητα και Ορθότητα Αλγορίθμου

Άσκηση 5β: Πολυπλοκότητα και Ορθότητα Αλγορίθμου

- Η πολυπλοκότητα του αλγορίθμου είναι $\mathcal{O}(nlogn)$ και προκύπτει ακριβώς όπως και αυτή της *MergeSort*. Βέβαια τώρα έχουμε άλλο ένα γραμμικό κόστος πέρα από αυτό της *Merge*, το οποίο προέρχεται από την εκτέλεση του αλγορίθμου από το 5α.

Άσκηση 5β: Πολυπλοκότητα και Ορθότητα Αλγορίθμου

- Η πολυπλοκότητα του αλγορίθμου είναι $\mathcal{O}(nlogn)$ και προκύπτει ακριβώς όπως και αυτή της *MergeSort*. Βέβαια τώρα έχουμε άλλο ένα γραμμικό κόστος πέρα από αυτό της *Merge*, το οποίο προέρχεται από την εκτέλεση του αλγορίθμου από το 5α.
- Ακριβώς πριν την *Merge* μέσα στη αναδρομική κλήση, τα δύο επιμέρους κομμάτια είναι ταξινομημένα και άρα μπορεί να εφαρμοστεί ο αλγόριθμος του προηγούμενου ερωτήματος.

Άσκηση 5β: Πολυπλοκότητα και Ορθότητα Αλγορίθμου

- Η πολυπλοκότητα του αλγορίθμου είναι $\mathcal{O}(nlogn)$ και προκύπτει ακριβώς όπως και αυτή της *MergeSort*. Βέβαια τώρα έχουμε άλλο ένα γραμμικό κόστος πέρα από αυτό της *Merge*, το οποίο προέρχεται από την εκτέλεση του αλγορίθμου από το 5α.
- Ακριβώς πριν την *Merge* μέσα στη αναδρομική κλήση, τα δύο επιμέρους κομμάτια είναι ταξινομημένα και άρα μπορεί να εφαρμοστεί ο αλγόριθμος του προηγούμενου ερωτήματος.
- Επιπλέον, όλα τα στοιχεία του αριστερού κομματιού έχουν δείκτη μικρότερο από κάθε ένα του δεξιού κομματιού, κάτι που προκύπτει από τον τρόπο λειτουργίας της *MergeSort*.

Άσκηση 5γ: Άθροισμα Στοιχείων σε Συγκεκριμένο Διάστημα

Πρόβλημα

Άσκηση 5γ: Άθροισμα Στοιχείων σε Συγκεκριμένο Διάστημα

Πρόβλημα

- **Είσοδος:** Δίνεται μη ταξινομημένος πίνακας $A[n]$ και αριθμοί L, M με $L < M$.

Άσκηση 5γ: Άθροισμα Στοιχείων σε Συγκεκριμένο Διάστημα

Πρόβλημα

- **Είσοδος**: Δίνεται μη ταξινομημένος πίνακας $A[n]$ και αριθμοί L, M με $L < M$.
- **Ζητούμενο**: Αλγόριθμος χρόνου $\mathcal{O}(nlogn)$ που να υπολογίζει το πλήθος των ζευγαριών (i, j), τέτοιων ώστε $L < \sum_{k=i}^{j} A[k] < M$, $i \leq j$.

Άσκηση 5γ: Άθροισμα Στοιχείων σε Συγκεκριμένο Διάστημα

Πρόβλημα

- **Είσοδος**: Δίνεται μη ταξινομημένος πίνακας $A[n]$ και αριθμοί L, M με $L < M$.
- **Ζητούμενο**: Αλγόριθμος χρόνου $\mathcal{O}(nlogn)$ που να υπολογίζει το πλήθος των ζευγαριών (i, j), τέτοιων ώστε $L < \sum_{k=i}^{j} A[k] < M$, $i \leq j$.

- Φτιάχνουμε πίνακα $B[n]$ όπου $B[i] = \sum_{k=1}^{i} A[k]$ σε χρόνο $\mathcal{O}(n)$.

Άσκηση 5γ: Άθροισμα Στοιχείων σε Συγκεκριμένο Διάστημα

Πρόβλημα

- **Είσοδος**: Δίνεται μη ταξινομημένος πίνακας $A[n]$ και αριθμοί L, M με $L < M$.
- **Ζητούμενο**: Αλγόριθμος χρόνου $\mathcal{O}(nlogn)$ που να υπολογίζει το πλήθος των ζευγαριών (i, j), τέτοιων ώστε $L < \sum_{k=i}^{j} A[k] < M$, $i \leq j$.

- Φτιάχνουμε πίνακα $B[n]$ όπου $B[i] = \sum_{k=1}^{i} A[k]$ σε χρόνο $\mathcal{O}(n)$.
- Βάση του B το ζητούμενο μπορεί να εκφραστεί ως το πλήθος των ζευγαριών (i, j), τέτοιων ώστε $L < B[j] - B[i] < M$, $i \leq j$.

Άσκηση 5γ: Άθροισμα Στοιχείων σε Συγκεκριμένο Διάστημα

Πρόβλημα

- **Είσοδος**: Δίνεται μη ταξινομημένος πίνακας $A[n]$ και αριθμοί L, M με $L < M$.
- **Ζητούμενο**: Αλγόριθμος χρόνου $\mathcal{O}(nlogn)$ που να υπολογίζει το πλήθος των ζευγαριών (i, j), τέτοιων ώστε $L < \sum_{k=i}^{j} A[k] < M$, $i \leq j$.

- Φτιάχνουμε πίνακα $B[n]$ όπου $B[i] = \sum_{k=1}^{i} A[k]$ σε χρόνο $\mathcal{O}(n)$.
- Βάση του B το ζητούμενο μπορεί να εκφραστεί ως το πλήθος των ζευγαριών (i, j), τέτοιων ώστε $L < B[j] - B[i] < M$, $i \leq j$.
- Επομένως χρησιμοποιούμε τον αλγόριθμο του ερωτήματος β.

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

Είσοδος: Πίνακας A μεγέθους $N \times N$ με 0 και 1.
Έξοδος: Πλήθος υποπινάκων με άθροισμα στοιχείων k.

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

Είσοδος: Πίνακας A μεγέθους $N \times N$ με 0 και 1.
Έξοδος: Πλήθος υποπινάκων με άθροισμα στοιχείων k.

Naive υλοποίηση

Για κάθε υποπίνακα υπολόγισε το άθροισμα των στοιχείων του. Χρόνος $O(N^4)$.

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

Είσοδος: Πίνακας A μεγέθους $N \times N$ με 0 και 1.
Έξοδος: Πλήθος υποπινάκων με άθροισμα στοιχείων k.

Naive υλοποίηση

Για κάθε υποπίνακα υπολόγισε το άθροισμα των στοιχείων του. Χρόνος $O(N^4)$.

Ιδέα

Αν γνωρίζω το άθροισμα των στοιχείων όλων των πινάκων που ξεκινούν από το $(1, 1)$, μπορώ να υπολογίσω το άθροισμα των στοιχείων οποιουδήποτε υποπίνακα σε $O(1)$.

Για να ορίσουμε έναν υποπίνακα αρκούν 4 συντεταγμένες, έστω x_1, x_2, y_1, y_2 όπου $x_1 \leq y_1$ και $x_2 \leq y_2$.
Αν $s(y_1, y_2) = \sum_{\substack{1 \leq y_1' \leq y_1 \\ 1 \leq y_2' \leq y_2}} A(y_1', y_2')$ τότε το άθροισμα ενός υποπίνακα (x_1, x_2, y_1, y_2) υπολογίζεται από τον τύπο:

$$S(x_1, x_2, y_1, y_2) = s(y_1, y_2) - s(x_1, y_2) + s(y_1, x_2) - s(x_1, x_2)$$

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

Για να ορίσουμε έναν υποπίνακα αρκούν 4 συντεταγμένες, έστω x_1, x_2, y_1, y_2 όπου $x_1 \leq y_1$ και $x_2 \leq y_2$.
Αν $s(y_1, y_2) = \sum_{\substack{1 \leq y_1' \leq y_1 \\ 1 \leq y_2' \leq y_2}} A(y_1', y_2')$ τότε το άθροισμα ενός υποπίνακα (x_1, x_2, y_1, y_2) υπολογίζεται από τον τύπο:

$$S(x_1, x_2, y_1, y_2) = s(y_1, y_2) - s(x_1, y_2) + s(y_1, x_2) - s(x_1, x_2)$$

- Για να υπολογίσουμε τον πίνακα s χρειαζόμαστε $O(N^2)$.
- Για κάθε ένα στοιχείο από τα $O(N^4)$ του πίνακα S χρειαζόμαστε $O(1)$ χρόνο.

Πολυπλοκότητα

Συνολικά, $O(N^4)$.

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

Λύση

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

Λύση

1. Υπολογισμός αθροίσματος στήλης j μέχρι και την i-οστή γραμμή $C(j,i) = \sum_{1 \le i' \le i} A(i',j)$.

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

Λύση

1. Υπολογισμός αθροίσματος στήλης j μέχρι και την i-οστή γραμμή $C(j,i) = \sum_{1 \le i' \le i} A(i',j)$.
2. Σταθεροποιούμε δύο γραμμές x, y με $x < y$ και αναζητούμε το πλήθος των υποπινάκων με άθροισμα στοιχείων k σε αυτή τη λωρίδα.

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

- Κρατάμε δύο δείκτες *front*, *rear* που δείχνουν σε στήλες και ξεκινούν από τη στήλη 1.

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

- Κρατάμε δύο δείκτες *front*, *rear* που δείχνουν σε στήλες και ξεκινούν από τη στήλη 1.
- Για κάθε στήλη j, το άθροισμα των στοιχείων της ανάμεσα στις γραμμές x, y δίνεται από το $F(j) = C(j, y) - C(j, x - 1)$.

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

- Κρατάμε δύο δείκτες *front*, *rear* που δείχνουν σε στήλες και ξεκινούν από τη στήλη 1.
- Για κάθε στήλη j, το άθροισμα των στοιχείων της ανάμεσα στις γραμμές x, y δίνεται από το $F(j) = C(j, y) - C(j, x - 1)$.
- Αν το τρέχον άθροισμα είναι μικρότερο από k, προχωράμε το δείκτη *rear* κατά 1. Αν είναι μεγαλύτερο, το δείκτη *front*.

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

- Κρατάμε δύο δείκτες *front*, *rear* που δείχνουν σε στήλες και ξεκινούν από τη στήλη 1.
- Για κάθε στήλη j, το άθροισμα των στοιχείων της ανάμεσα στις γραμμές x, y δίνεται από το $F(j) = C(j, y) - C(j, x - 1)$.
- Αν το τρέχον άθροισμα είναι μικρότερο από k, προχωράμε το δείκτη *rear* κατά 1. Αν είναι μεγαλύτερο, το δείκτη *front*.
- Αν είναι ίσο με k τότε έχουμε βρει έναν από τους επιθυμητούς υποπίνακες. Προχωράμε τους δείκτες *front*, *rear* μέχρι να βρούμε μη μηδενική στήλη. Αν ο *front* έχει προχωρήσει df και ο *rear* dr, προσθέτουμε στο άθροισμα $(df + 1)(dr + 1)$.

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

Χρονική Πολυπλοκότητα

- Υπολογισμός C σε $O(N^2)$.

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

Χρονική Πολυπλοκότητα

- Υπολογισμός C σε $O(N^2)$.
- Υπολογισμός πλήθους υποπινάκων στη λωρίδα x, y με άθροισμα στοιχείων k σε χρόνο $O(N)$ (οι δείκτες κινούνται μόνο δεξιά).

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

Χρονική Πολυπλοκότητα

- Υπολογισμός C σε $O(N^2)$.
- Υπολογισμός πλήθους υποπινάκων στη λωρίδα x, y με άθροισμα στοιχείων k σε χρόνο $O(N)$ (οι δείκτες κινούνται μόνο δεξιά).
- $O(N^2)$ ζεύγη γραμμών x, y.

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

Χρονική Πολυπλοκότητα

- Υπολογισμός C σε $O(N^2)$.
- Υπολογισμός πλήθους υποπινάκων στη λωρίδα x, y με άθροισμα στοιχείων k σε χρόνο $O(N)$ (οι δείκτες κινούνται μόνο δεξιά).
- $O(N^2)$ ζεύγη γραμμών x, y.
- $O(N^3)$ συνολικά.

Καλύτερη Λύση για μικρό k

- Βρίσκουμε τον αριθμό των υποπινάκων με άθροισμα στοιχείων k που τέμνουν τη γραμμή $N/2$.

Καλύτερη Λύση για μικρό k

- Βρίσκουμε τον αριθμό των υποπινάκων με άθροισμα στοιχείων k που τέμνουν τη γραμμή $N/2$.
- Για να το κάνουμε αυτό βρίσκουμε τις θέσεις των k κοντινότερων άσσων από πάνω και από κάτω από τη γραμμή $N/2$ και τα βάζουμε σε λίστες. Αριθμούμε όλα τα διαστήματα $[\ell, r]$ με $1 \leq \ell \leq r \leq N$ και ενώνουμε τις λίστες που αντιστοιχούν σε διαφορετικές γραμμές. Σε $O(k)$ βρίσκουμε τους υποπίνακες που ανήκουν στο διάστημα $[\ell, r]$ και έχουν άθροισμα στοιχείων k και τέμνουν τη γραμμή $N/2$.

Καλύτερη Λύση για μικρό k

- Βρίσκουμε τον αριθμό των υποπινάκων με άθροισμα στοιχείων k που τέμνουν τη γραμμή $N/2$.
- Για να το κάνουμε αυτό βρίσκουμε τις θέσεις των k κοντινότερων άσσων από πάνω και από κάτω από τη γραμμή $N/2$ και τα βάζουμε σε λίστες. Αριθμούμε όλα τα διαστήματα $[\ell, r]$ με $1 \leq \ell \leq r \leq N$ και ενώνουμε τις λίστες που αντιστοιχούν σε διαφορετικές γραμμές. Σε $O(k)$ βρίσκουμε τους υποπίνακες που ανήκουν στο διάστημα $[\ell, r]$ και έχουν άθροισμα στοιχείων k και τέμνουν τη γραμμή $N/2$.
- Άρα συνολικά $N^2 k$ για όλα τα διαστήματα $[\ell, r]$.

Καλύτερη Λύση για μικρό k

- Βρίσκουμε τον αριθμό των υποπινάκων με άθροισμα στοιχείων k που τέμνουν τη γραμμή $N/2$.
- Για να το κάνουμε αυτό βρίσκουμε τις θέσεις των k κοντινότερων άσσων από πάνω και από κάτω από τη γραμμή $N/2$ και τα βάζουμε σε λίστες. Αριθμούμε όλα τα διαστήματα $[\ell, r]$ με $1 \leq \ell \leq r \leq N$ και ενώνουμε τις λίστες που αντιστοιχούν σε διαφορετικές γραμμές. Σε $O(k)$ βρίσκουμε τους υποπίνακες που ανήκουν στο διάστημα $[\ell, r]$ και έχουν άθροισμα στοιχείων k και τέμνουν τη γραμμή $N/2$.
- Άρα συνολικά $N^2 k$ για όλα τα διαστήματα $[\ell, r]$.
- Στη συνέχεια πρέπει να βρούμε τον αριθμό των υποπινάκων με άθροισμα στοιχείων k που δεν τέμνουν τη $N/2$ γραμμή. Χωρίζουμε τον πίνακα κάθετα στη στήλη $N/2$ και επαναλαμβάνουμε.

1η Προγραμματιστική: Παιδική Χαρά στην Τετραγωνούπολη

Πολυπλοκότητα

- Η αναδρομική σχέση είναι $T(N) = 4T(N/2) + N^2k$.
  Άρα $\Theta(N^2 k \log N)$.

2η προγραμματιστική: Προφορική Εξέταση

Πρόβλημα

2η προγραμματιστική: Προφορική Εξέταση

Πρόβλημα

- **Είσοδος:** N, K, Πίνακας D μεγέθους N με στοιχεία d_i.

2η προγραμματιστική: Προφορική Εξέταση

Πρόβλημα

- **Είσοδος**: N, K, Πίνακας D μεγέθους N με στοιχεία d_i.
- **Έξοδος**: Θέση εξεταστή, Χρόνος ολοκλήρωσης.

2η προγραμματιστική: Προφορική Εξέταση

Πρόβλημα

- **Είσοδος:** N, K, Πίνακας D μεγέθους N με στοιχεία d_i.
- **Έξοδος:** Θέση εξεταστή, Χρόνος ολοκλήρωσης.

- Παρατήρηση: Σύνθετος ο υπολογισμός του χρόνου ολοκλήρωσης. Πιο εύκολος ο υπολογισμός του χρόνου αρχής της εξέτασης.

2η προγραμματιστική: Προφορική Εξέταση

Πρόβλημα

- **Είσοδος:** N, K, Πίνακας D μεγέθους N με στοιχεία d_i.
- **Έξοδος:** Θέση εξεταστή, Χρόνος ολοκλήρωσης.

- Παρατήρηση: Σύνθετος ο υπολογισμός του χρόνου ολοκλήρωσης. Πιο εύκολος ο υπολογισμός του χρόνου αρχής της εξέτασης.
- *Naive* υλοποίηση: Υπολογισμός ΕΚΠ των d_i, έστω t. Την χρονική στιγμή t ολοκληρώνουν την εξέταση όλοι μαζί.

2η προγραμματιστική: Προφορική Εξέταση

Πρόβλημα

- **Είσοδος:** N, K, Πίνακας D μεγέθους N με στοιχεία d_i.
- **Έξοδος:** Θέση εξεταστή, Χρόνος ολοκλήρωσης.

- Παρατήρηση: Σύνθετος ο υπολογισμός του χρόνου ολοκλήρωσης. Πιο εύκολος ο υπολογισμός του χρόνου αρχής της εξέτασης.
- *Naive* υλοποίηση: Υπολογισμός ΕΚΠ των d_i, έστω t. Την χρονική στιγμή t ολοκληρώνουν την εξέταση όλοι μαζί.
- Αν και ο υπολογισμός του ΕΚΠ είναι $\mathcal{O}(N)$, οι αριθμοί ενδέχεται να είναι αρκετά μεγάλοι. Επομένως μπορεί να έχουμε *overflow* ακόμα και με *long long* μεταβλητές.

2η προγραμματιστική: Προφορική Εξέταση

Πρόβλημα

- **Είσοδος:** N, K, Πίνακας D μεγέθους N με στοιχεία d_i.
- **Έξοδος:** Θέση εξεταστή, Χρόνος ολοκλήρωσης.

- Παρατήρηση: Σύνθετος ο υπολογισμός του χρόνου ολοκλήρωσης. Πιο εύκολος ο υπολογισμός του χρόνου αρχής της εξέτασης.
- *Naive* υλοποίηση: Υπολογισμός ΕΚΠ των d_i, έστω t. Την χρονική στιγμή t ολοκληρώνουν την εξέταση όλοι μαζί.
- Αν και ο υπολογισμός του ΕΚΠ είναι $\mathcal{O}(N)$, οι αριθμοί ενδέχεται να είναι αρκετά μεγάλοι. Επομένως μπορεί να έχουμε *overflow* ακόμα και με *long long* μεταβλητές.
- Χειρότερη περίπτωση όλοι οι αριθμοί είναι σχετικά πρώτοι μεταξύ τους.

2η προγραμματιστική: Προφορική Εξέταση

2η προγραμματιστική: Προφορική Εξέταση

- **Ιδέα**: Έστω ότι γνωρίζω την χρονική στιγμή που θα ξεκινήσει η εξέταση μου, t_0. Μπορώ να επαληθεύσω σε $\mathcal{O}(N)$ ότι πράγματι έχει ξεκινήσει η εξέταση την χρονική στιγμή t_0;

2η προγραμματιστική: Προφορική Εξέταση

- **Ιδέα**: Έστω ότι γνωρίζω την χρονική στιγμή που θα ξεκινήσει η εξέταση μου, t_0. Μπορώ να επαληθεύσω σε $\mathcal{O}(N)$ ότι πράγματι έχει ξεκινήσει η εξέταση την χρονική στιγμή t_0;
- Αν $\sum_{i=1}^{N}(1 + \left\lfloor \frac{t_0}{d_i} \right\rfloor) > K$, τότε έχει αρχίσει η εξέτασή μου, αλλιώς όχι.

2η προγραμματιστική: Προφορική Εξέταση

- **Ιδέα**: Έστω ότι γνωρίζω την χρονική στιγμή που θα ξεκινήσει η εξέταση μου, t_0. Μπορώ να επαληθεύσω σε $\mathcal{O}(N)$ ότι πράγματι έχει ξεκινήσει η εξέταση την χρονική στιγμή t_0;
- Αν $\sum_{i=1}^{N}(1 + \left\lfloor \frac{t_0}{d_i} \right\rfloor) > K$, τότε έχει αρχίσει η εξέτασή μου, αλλιώς όχι.
- **Λύση**: Δυαδική αναζήτηση στο $[1, K \cdot d_{max}]$.

2η προγραμματιστική: Προφορική Εξέταση

- **Ιδέα**: Έστω ότι γνωρίζω την χρονική στιγμή που θα ξεκινήσει η εξέταση μου, t_0. Μπορώ να επαληθεύσω σε $\mathcal{O}(N)$ ότι πράγματι έχει ξεκινήσει η εξέταση την χρονική στιγμή t_0;
- Αν $\sum_{i=1}^{N}(1 + \left\lfloor \frac{t_0}{d_i} \right\rfloor) > K$, τότε έχει αρχίσει η εξέτασή μου, αλλιώς όχι.
- **Λύση**: Δυαδική αναζήτηση στο $[1, K \cdot d_{max}]$.
- Κάθε χρονική στιγμή στην δυαδική αναζήτηση ελέγχουμε εάν έχει ξεκινήσει η εξέτασή μας. Εάν ναι, συνεχίζουμε στο μικρότερο μισό. Εάν όχι, στο μεγαλύτερο μισό. Τερματίζουμε όταν η εξέταση έχει ξεκινήσει μία χρονική στιγμή t αλλά όχι την $t - 1$.

2η προγραμματιστική: Προφορική Εξέταση

- **Ιδέα**: Έστω ότι γνωρίζω την χρονική στιγμή που θα ξεκινήσει η εξέταση μου, t_0. Μπορώ να επαληθεύσω σε $\mathcal{O}(N)$ ότι πράγματι έχει ξεκινήσει η εξέταση την χρονική στιγμή t_0;
- Αν $\sum_{i=1}^{N}(1+\left\lfloor \frac{t_0}{d_i} \right\rfloor) > K$, τότε έχει αρχίσει η εξέτασή μου, αλλιώς όχι.
- **Λύση:** Δυαδική αναζήτηση στο $[1, K \cdot d_{max}]$.
- Κάθε χρονική στιγμή στην δυαδική αναζήτηση ελέγχουμε εάν έχει ξεκινήσει η εξέτασή μας. Εάν ναι, συνεχίζουμε στο μικρότερο μισό. Εάν όχι, στο μεγαλύτερο μισό. Τερματίζουμε όταν η εξέταση έχει ξεκινήσει μία χρονική στιγμή t αλλά όχι την $t-1$.
- Όταν γνωρίζουμε την t_0, υπολογίζουμε σε τι σειρά μπαίνουμε: $j = \sum_{i=1}^{N}(1+\left\lfloor \frac{t_0}{d_i} \right\rfloor) - \sum_{i=1}^{N}(1+\left\lfloor \frac{t_0-1}{d_i} \right\rfloor)$. Έπειτα, κάνουμε ένα πέρασμα στον D, μετρώντας πόσα $t_0 \ mod \ d_i = 0$. Στο j-ιοστό που βρίσκουμε, αυτός θα μας εξετάσει.