Αλγόριθμοι και Πολυπλοκότητα
1η Σειρά Γραπτών Ασκήσεων

CoReLab

ΣΗΜΜΥ - Ε.Μ.Π.

12 Νοεμβρίου 2018

Outline

Άσκηση 1 (α): Ασυμπτωτικός Συμβολισμός

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

Άσκηση 1 (α): 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 1 (α): Ασυμπτωτικός Συμβολισμός

Χρήσιμες γνωστές ανισότητες

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

Άσκηση 1 (α): 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 1 (β): Αναδρομικές Σχέσεις

Λύση

1. $T(n) = 2T(n/3) + n\log n = \Theta(n\log n)$: ΜΤ περ.3

Άσκηση 1 (β): Αναδρομικές Σχέσεις

Λύση

1. $T(n) = 2T(n/3) + n \log n = \Theta(n \log n)$: MT περ.3
2. $T(n) = 3T(n/3) + n \log n = \Theta(n \log^2 n)$: Όχι MT 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.

Άσκηση 1 (β): Αναδρομικές Σχέσεις

Λύση

1. $T(n) = 2T(n/3) + n\log n = \Theta(n\log n)$: MT περ.3
2. $T(n) = 3T(n/3) + n\log n = \Theta(n\log^2 n)$: Όχι MT 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. $T(n) = 4T(n/3) + n\log n = \Theta(n^{\log_3 4})$: MT περ.1

Άσκηση 1 (β): Αναδρομικές Σχέσεις

Λύση

1. $T(n) = 2T(n/3) + n\log n = \Theta(n\log n)$: MT περ.3
2. $T(n) = 3T(n/3) + n\log n = \Theta(n\log^2 n)$: Όχι MT 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. $T(n) = 4T(n/3) + n\log n = \Theta(n^{\log_3 4})$: MT 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.

Άσκηση 1 (β): Αναδρομικές Σχέσεις

5. $T(n) = T(n/2) + T(n/3) + T(n/6) + n = \Theta(n \log n)$: Επειδή $n/2 + n/3 + n/6 = n$, υποπτευόμαστε $T(n) = \Theta(n \log n)$ και το αποδεικνύουμε με επαγωγή ή δέντρο αναδρομής. Στο τελευταίο, μπορούμε να φράξουμε το ύψος από το κοντύτερο κλαδί ($\log_6 n$) και το μακρύτερο ($\log_2 n$).

Άσκηση 1 (β): Αναδρομικές Σχέσεις

5. $T(n) = T(n/2) + T(n/3) + T(n/6) + n = \Theta(n \log n)$: Επειδή $n/2 + n/3 + n/6 = n$, υποπτευόμαστε $T(n) = \Theta(n \log n)$ και το αποδεικνύουμε με επαγωγή ή δέντρο αναδρομής. Στο τελευταίο, μπορούμε να φράξουμε το ύψος από το κοντύτερο κλαδί ($\log_6 n$) και το μακρύτερο ($\log_2 n$).

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

Άσκηση 1 (β): Αναδρομικές Σχέσεις

5. $T(n) = T(n/2) + T(n/3) + T(n/6) + n = \Theta(n \log n)$: Επειδή $n/2 + n/3 + n/6 = n$, υποπτευόμαστε $T(n) = \Theta(n \log n)$ και το αποδεικνύουμε με επαγωγή ή δέντρο αναδρομής. Στο τελευταίο, μπορούμε να φράξουμε το ύψος από το κοντύτερο κλαδί ($\log_6 n$) και το μακρύτερο ($\log_2 n$).

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

7. $T(n) = T(n/4) + \sqrt{n} = \Theta(\sqrt{n})$:
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 $S(k) = \Theta(k)$.

Outline

Άσκηση 2 (α.1): Μερική ταξινόμηση πίνακα σε k–ομάδες

Είσοδος: Πίνακας $A[1...n]$.
Πρόβλημα: Καταμερισμός του πίνακα σε k ομάδες, εξωτερικά ταξινομημένες, σε χρόνο $O(n \log k)$.

Άσκηση 2 (α.1): Μερική ταξινόμηση πίνακα σε k– ομάδες

Είσοδος: Πίνακας $A[1...n]$.
Πρόβλημα: Καταμερισμός του πίνακα σε k ομάδες, εξωτερικά ταξινομημένες, σε χρόνο $O(n \log k)$.
Στόχος:

Άσκηση 2 (α.1): Μερική ταξινόμηση πίνακα σε k–ομάδες

Είσοδος: Πίνακας $A[1...n]$.
Πρόβλημα: Καταμερισμός του πίνακα σε k ομάδες, εξωτερικά ταξινομημένες, σε χρόνο $O(n \log k)$.
Στόχος:

- Τμηματοποίηση του πίνακα A σε k ομάδες των $\frac{n}{k}$ στοιχείων.

Άσκηση 2 (α.1): Μερική ταξινόμηση πίνακα σε k– ομάδες

Είσοδος: Πίνακας $A[1...n]$.
Πρόβλημα: Καταμερισμός του πίνακα σε k ομάδες, εξωτερικά ταξινομημένες, σε χρόνο $O(n \log k)$.
Στόχος:

- Τμηματοποίηση του πίνακα A σε k ομάδες των $\frac{n}{k}$ στοιχείων.
- Κάθε ομάδα δεν χρειάζεται να είναι εσωτερικά ταξινομημένη

Άσκηση 2 (α.1): Μερική ταξινόμηση πίνακα σε k–ομάδες

Είσοδος: Πίνακας $A[1...n]$.
Πρόβλημα: Καταμερισμός του πίνακα σε k ομάδες, εξωτερικά ταξινομημένες, σε χρόνο $O(n \log k)$.
Στόχος:

- Τμηματοποίηση του πίνακα A σε k ομάδες των $\frac{n}{k}$ στοιχείων.
- Κάθε ομάδα δεν χρειάζεται να είναι εσωτερικά ταξινομημένη
- Το μικρότερο στοιχείο κάθε ομάδας να είναι μεγαλύτερο από τα μεγαλύτερα στοιχεία των προηγούμενων ομάδων.

Άσκηση 2 (α.1): Μερική ταξινόμηση πίνακα σε k– ομάδες

Είσοδος: Πίνακας $A[1...n]$.
Πρόβλημα: Καταμερισμός του πίνακα σε k ομάδες, εξωτερικά ταξινομημένες, σε χρόνο $O(n \log k)$.
Στόχος:

- Τμηματοποίηση του πίνακα A σε k ομάδες των $\frac{n}{k}$ στοιχείων.
- Κάθε ομάδα δεν χρειάζεται να είναι εσωτερικά ταξινομημένη
- Το μικρότερο στοιχείο κάθε ομάδας να είναι μεγαλύτερο από τα μεγαλύτερα στοιχεία των προηγούμενων ομάδων.
- Το μεγαλύτερο στοιχείο κάθε ομάδας να είναι μικρότερο από τα μικρότερα στοιχεία των επόμενων ομάδων.

Άσκηση 2 (α.1): Παράδειγμα

2	1	4	3	6	5	7	8	10	12	11	9	16	15	13	14

Άσκηση 2 (α.1): Παράδειγμα

2	1	4	3	6	5	7	8	10	12	11	9	16	15	13	14

Ιδέα

Αρκεί να μπορούσαμε να βρούμε τα στοιχεία που έχουν την μέγιστη τιμή σε κάθε block.
Δηλαδή: το $(\frac{n}{k})$-στό, $(\frac{2n}{k})$-στό, $\cdots$, $(\frac{(k-1)n}{k})$-στό, n-στό μικρότερο στοιχείο του πίνακα

Άσκηση 2 (α.1): Παράδειγμα

2	1	4	3	6	5	7	8	10	12	11	9	16	15	13	14

Ιδέα

Αρκεί να μπορούσαμε να βρούμε τα στοιχεία που έχουν την μέγιστη τιμή σε κάθε block.
Δηλαδή: το $(\frac{n}{k})$-στό, $(\frac{2n}{k})$-στό, $\cdots$, $(\frac{(k-1)n}{k})$-στό, n-στό μικρότερο στοιχείο του πίνακα

Γιατί?

Άσκηση 2 (α.1): Παράδειγμα

2	1	4	3	6	5	7	8	10	12	11	9	16	15	13	14

Ιδέα

Αρκεί να μπορούσαμε να βρούμε τα στοιχεία που έχουν την μέγιστη τιμή σε κάθε block.
Δηλαδή: το $(\frac{n}{k})$-στό, $(\frac{2n}{k})$-στό, $\cdots$, $(\frac{(k-1)n}{k})$-στό, n-στό μικρότερο στοιχείο του πίνακα

Γιατί? Κάθε άλλο στοιχείο, μπορούμε να το τοποθετήσουμε με δυαδική αναζήτηση $O(n) \times O(\log k)$

Άσκηση 2 (α.1): Πρώτη προσέγγιση (Διαδοχική)

- Μπορούμε να χρησιμοποιήσουμε τη ντετερμινιστική μέθοδο που είδαμε στο μάθημα για την εύρεση του k-στού μικρότερου του πίνακα σε $O(n)$.
- Να βρούμε διαδοχικά το $(\frac{n}{k})$-στό, $(\frac{2n}{k})$-στό, $\cdots$, $(\frac{(k-1)n}{k})$-στό, n-στό μικρότερο στοιχείο του πίνακα σε χρόνο $k \times O(n) = O(kn)$.

Άσκηση 2 (α.1): Πρώτη προσέγγιση (Διαδοχική)

- Μπορούμε να χρησιμοποιήσουμε τη ντετερμινιστική μέθοδο που είδαμε στο μάθημα για την εύρεση του k-στού μικρότερου του πίνακα σε $O(n)$.
- Να βρούμε διαδοχικά το $(\frac{n}{k})$-στό, $(\frac{2n}{k})$-στό, $\cdots$, $(\frac{(k-1)n}{k})$-στό, n-στό μικρότερο στοιχείο του πίνακα σε χρόνο $k \times O(n) = O(kn)$.

Μπορούμε καλύτερα?

Άσκηση 2 (α.1): Δεύτερη Προσέγγιση (Διαίρει & Βασίλευε)

Παρατήρηση

Θέλουμε να βρούμε το $(\frac{n}{k})$-στό, $(\frac{2n}{k})$-στό, $\cdots$, $(\frac{(k-1)n}{k})$-στό, n-στό μικρότερο στοιχείο. Δεν είναι αναγκαίο να τα βρούμε με αυτή την σειρά

Άσκηση 2 (α.1): Δεύτερη Προσέγγιση (Διαίρει & Βασίλευε)

Παρατήρηση

Θέλουμε να βρούμε το $(\frac{n}{k})$-στό, $(\frac{2n}{k})$-στό, $\cdots$, $(\frac{(k-1)n}{k})$-στό, n-στό μικρότερο στοιχείο. Δεν είναι αναγκαίο να τα βρούμε με αυτή την σειρά

- Μπορούμε να βρούμε το μεσαίο στοιχείο από τα ζητούμενα, δηλαδή το $\frac{n}{k}(k/2)$-στό ή αλλιώς $\mathcal{M}edian$.

Άσκηση 2 (α.1): Δεύτερη Προσέγγιση (Διαίρει & Βασίλευε)

Παρατήρηση

Θέλουμε να βρούμε το $(\frac{n}{k})$-στό, $(\frac{2n}{k})$-στό, $\cdots$, $(\frac{(k-1)n}{k})$-στό, n-στό μικρότερο στοιχείο. Δεν είναι αναγκαίο να τα βρούμε με αυτή την σειρά

- Μπορούμε να βρούμε το μεσαίο στοιχείο από τα ζητούμενα, δηλαδή το $\frac{n}{k}(k/2)$-στό ή αλλιώς $\mathcal{M}edian$.
- Κάνουμε Partition τον πίνακα A γύρω από αυτό το στοιχείο.

Άσκηση 2 (α.1): Δεύτερη Προσέγγιση (Διαίρει & Βασίλευε)

Παρατήρηση

Θέλουμε να βρούμε το $(\frac{n}{k})$-στό, $(\frac{2n}{k})$-στό, $\cdots$, $(\frac{(k-1)n}{k})$-στό, n-στό μικρότερο στοιχείο. <u>Δεν είναι αναγκαίο να τα βρούμε με αυτή την σειρά</u>

- Μπορούμε να βρούμε το μεσαίο στοιχείο από τα ζητούμενα, δηλαδή το $\frac{n}{k}(k/2)$-στό ή αλλιώς $\mathcal{M}edian$.
- Κάνουμε Partition τον πίνακα A γύρω από αυτό το στοιχείο.
- Εφαρμόζουμε σε κάθε υποπίνακα την ίδια τακτική.

Άσκηση 2 (α.1): Δεύτερη Προσέγγιση (Διαίρει & Βασίλευε)

Παρατήρηση

Θέλουμε να βρούμε το $(\frac{n}{k})$-στό, $(\frac{2n}{k})$-στό, $\cdots$, $(\frac{(k-1)n}{k})$-στό, n-στό μικρότερο στοιχείο. Δεν είναι αναγκαίο να τα βρούμε με αυτή την σειρά

- Μπορούμε να βρούμε το μεσαίο στοιχείο από τα ζητούμενα, δηλαδή το $\frac{n}{k}(k/2)$-στό ή αλλιώς $\mathcal{M}edian$.
- Κάνουμε Partition τον πίνακα A γύρω από αυτό το στοιχείο.
- Εφαρμόζουμε σε κάθε υποπίνακα την ίδια τακτική.

Πολυπλοκότητα

$$\begin{cases} T(n) = 2T(n/2) + \Theta(n) \\ T(n/k) = 1 \end{cases} \Rightarrow T(n) = \Theta(n \log k)$$

Άσκηση 2 (α.1): Δεύτερη Προσέγγιση: Παράδειγμα

2	11	4	3	15	16	8	7	9	12	10	14	1	5	13	6
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

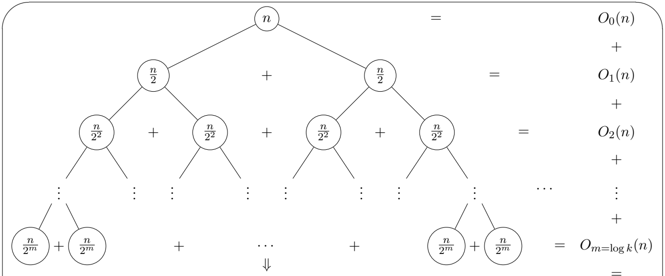
Άσκηση 2 (α.1): Δεύτερη Προσέγγιση: Παράδειγμα

2	1	4	3	5	6	7	8	11	9	12	10	14	15	13	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Άσκηση 2 (α.1): Δεύτερη Προσέγγιση: Παράδειγμα

2	1	3	4	5	6	7	8	11	9	10	12	14	15	13	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Άσκηση 2 (α.1): Δέντρο αναδρομής



$$O\left(\sum_{i=0}^{m} 2^i \cdot \frac{n}{2^i}\right) = O\left(\sum_{i=0}^{m} n\right) = O(m \cdot n) \quad \Leftrightarrow \quad O(n \cdot \log k)$$

Άσκηση 2 (α.1): Φράγμα Μερικής Ταξινόμησης σε k-ομάδες

Ζητούμενο

Κάτω φράγμα μερικής ταξινόμησης πίνακα σε $k-$ ομάδες. Προφανώς αναζητούμε φράγμα καλύτερο από το $n \log n$.

Άσκηση 2 (α.1): Φράγμα Μερικής Ταξινόμησης σε k-ομάδες

Ζητούμενο

Κάτω φράγμα μερικής ταξινόμησης πίνακα σε $k-$ ομάδες. Προφανώς αναζητούμε φράγμα καλύτερο από το $n \log n$.

Παρατήρηση

Ένας βέλτιστος αλγόριθμος δεν πρέπει να ενδιαφέρεται για το πως θα απεικονίσει εσωτερικά κάθε μία από τις k ομάδες. Συνεπώς πρέπει να περιοριστεί το σύνολο των πιθανών output του αλγορίθμου.

Άσκηση 2 (α.1): Φράγμα Μερικής Ταξινόμησης σε k-ομάδες

Ζητούμενο

Κάτω φράγμα μερικής ταξινόμησης πίνακα σε $k-$ ομάδες. Προφανώς αναζητούμε φράγμα καλύτερο από το $n \log n$.

Παρατήρηση

Ένας βέλτιστος αλγόριθμος δεν πρέπει να ενδιαφέρεται για το πως θα απεικονίσει εσωτερικά κάθε μία από τις k ομάδες. Συνεπώς πρέπει να περιοριστεί το σύνολο των πιθανών output του αλγορίθμου.

Κάθε συγκριτικός αλγόριθμος πρέπει να αντιμετωπίζει την οποιαδήποτε εσωτερική αναδιάταξη μιας ομάδας ως μία περίπτωση.

Άσκηση 2 (α.1): Φράγμα ταξινόμησης

Στην περίπτωση αυτή, κάθε συγκριτικός αλγόριθμος θα κατασκευάζει ένα δυαδικό δέντρο με τουλάχιστον

$$L = \frac{n!}{q_1!q_2!\dots q_k!} = \frac{n!}{((\frac{n}{k})!)^k}$$

φύλλα και συνεπώς ύψος τουλάχιστον $\log L$.

Άσκηση 2 (α.1): Φράγμα ταξινόμησης

Στην περίπτωση αυτή, κάθε συγκριτικός αλγόριθμος θα κατασκευάζει ένα δυαδικό δέντρο με τουλάχιστον

$$L = \frac{n!}{q_1!q_2!\dots q_k!} = \frac{n!}{((\frac{n}{k})!)^k}$$

φύλλα και συνεπώς ύψος τουλάχιστον $\log L$.

Stirling's Approximation

$\log(n!) = n\log n - n + O(\log n)$

Άσκηση 2 (α.1): Φράγμα ταξινόμησης

Χρόνος Εκτέλεσης $\geq$

$$\log \frac{n!}{((\frac{n}{k})!)^k} = \log n! - k \log \left(\frac{n}{k}\right)! \overset{\text{Stirling's Apx}}{=}$$

$$= n \log n - n + O(\log n) - n \log \left(\frac{n}{k}\right) + n - kO(\log \left(\frac{n}{k}\right))$$

$$= n \log k + O(\log n) - kO(\log(n/k))$$

$$= \Theta(n \log k)$$

Άσκηση 2 (α.2): Πλήρης ταξινόμηση Πίνακα μερικώς ταξινομημένο σε k-ομάδες

Είσοδος: Πίνακας $A[1...n]$ που είναι ταξινομημένος κατά k μέρη.
Πρόβλημα: Ταξινόμηση σε χρόνο $O(n \log(n/k))$.

Άσκηση 2 (α.2): Πλήρης ταξινόμηση Πίνακα μερικώς ταξινομημένο σε k-ομάδες

Είσοδος: Πίνακας $A[1...n]$ που είναι ταξινομημένος κατά k μέρη.
Πρόβλημα: Ταξινόμηση σε χρόνο $O(n \log(n/k))$.
Λύση:

- Ταξινομούμε ξεχωριστά κάθε ομάδα των n/k στοιχείων.

Άσκηση 2 (α.2): Πλήρης ταξινόμηση Πίνακα μερικώς ταξινομημένο σε k-ομάδες

Είσοδος: Πίνακας $A[1...n]$ που είναι ταξινομημένος κατά k μέρη.
Πρόβλημα: Ταξινόμηση σε χρόνο $O(n \log(n/k))$.
Λύση:

- Ταξινομούμε ξεχωριστά κάθε ομάδα των n/k στοιχείων.
- Για κάθε ομάδα, αυτό κοστίζει $O(\frac{n}{k} \log \frac{n}{k})$

Άσκηση 2 (α.2): Πλήρης ταξινόμηση Πίνακα μερικώς ταξινομημένο σε k-ομάδες

Είσοδος: Πίνακας $A[1...n]$ που είναι ταξινομημένος κατά k μέρη.
Πρόβλημα: Ταξινόμηση σε χρόνο $O(n \log(n/k))$.
Λύση:

- Ταξινομούμε ξεχωριστά κάθε ομάδα των n/k στοιχείων.
- Για κάθε ομάδα, αυτό κοστίζει $O(\frac{n}{k} \log \frac{n}{k})$

Συνεπώς:

$$k \times O(\frac{n}{k} \log \frac{n}{k}) = O(n \log(n/k))$$

Άσκηση 2 (α.2): Φράγμα ταξινόμησης

Παρατήρηση: Αυτό το φράγμα ταξινόμησης μπορεί να προκύψει και χωρίς συνδυαστική.

Άσκηση 2 (α.2): Φράγμα ταξινόμησης

Παρατήρηση: Αυτό το φράγμα ταξινόμησης μπορεί να προκύψει και χωρίς συνδυαστική.

Λύση:

- Έστω ότι υπάρχει συγκριτικός αλγόριθμος για το ζητούμενο task με χρόνο $o(n \log(n/k))$.

Άσκηση 2 (α.2): Φράγμα ταξινόμησης

Παρατήρηση: Αυτό το φράγμα ταξινόμησης μπορεί να προκύψει και χωρίς συνδυαστική.

Λύση:

- Έστω ότι υπάρχει συγκριτικός αλγόριθμος για το ζητούμενο task με χρόνο $o(n \log(n/k))$.
- Εξαιτίας του προηγούμενου κάτω φράγματος, ξέρουμε ότι το πρόβλημα δημιουργίας ενός μερικώς ταξινομημένου πίνακα κοστίζει $\Theta(n \log k)$

Άσκηση 2 (α.2): Φράγμα ταξινόμησης

Παρατήρηση: Αυτό το φράγμα ταξινόμησης μπορεί να προκύψει και χωρίς συνδυαστική.

Λύση:

- Έστω ότι υπάρχει συγκριτικός αλγόριθμος για το ζητούμενο task με χρόνο $o(n \log(n/k))$.
- Εξαιτίας του προηγούμενου κάτω φράγματος, ξέρουμε ότι το πρόβλημα δημιουργίας ενός μερικώς ταξινομημένου πίνακα κοστίζει $\Theta(n \log k)$
- Συνεπώς θα μπορούσαμε να κατασκευάσουμε έναν πλήρη συγκριτικό αλγόριθμο ταξινόμησης με χρόνο:
  $\Theta(n \log k) + o(n \log(n/k)) = o(n \log(k \times n/k)) = o(n \log n)$

Άσκηση 2 (α.2): Φράγμα ταξινόμησης

Παρατήρηση: Αυτό το φράγμα ταξινόμησης μπορεί να προκύψει και χωρίς συνδυαστική.

Λύση:

- Έστω ότι υπάρχει συγκριτικός αλγόριθμος για το ζητούμενο task με χρόνο $o(n\log(n/k))$.
- Εξαιτίας του προηγούμενου κάτω φράγματος, ξέρουμε ότι το πρόβλημα δημιουργίας ενός μερικώς ταξινομημένου πίνακα κοστίζει $\Theta(n\log k)$
- Συνεπώς θα μπορούσαμε να κατασκευάσουμε έναν πλήρη συγκριτικό αλγόριθμο ταξινόμησης με χρόνο:
  $\Theta(n\log k) + o(n\log(n/k)) = o(n\log(k \times n/k)) = o(n\log n)$

Άτοπο $\Rightarrow$ Κάθε συγκριτικός αλγόριθμος για το ζητούμενο task χρειάζεται τουλάχιστον $\Omega(n\log(n/k))$.

Άσκηση 2 (α.2): Φράγμα ταξινόμησης

Με συνδυαστική: Αρκεί να μετρήσουμε τα διαφορετικά output που μπορεί να δημιουργήσει ο αλγόριθμος

- Για κάθε ομάδα των $\frac{n}{k}$ στοιχείων μπορεί να προκύψει οποιαδήποτε αναδιάταξη από τις $\frac{n}{k}!$

Άσκηση 2 (α.2): Φράγμα ταξινόμησης

Με συνδυαστική: Αρκεί να μετρήσουμε τα διαφορετικά output που μπορεί να δημιουργήσει ο αλγόριθμος

- Για κάθε ομάδα των $\frac{n}{k}$ στοιχείων μπορεί να προκύψει οποιαδήποτε αναδιάταξη από τις $\frac{n}{k}!$
- Εφαρμόζουμε πολλαπλασιαστική αρχή για να συνδιάσουμε ανεξάρτητες ομάδες

$$L = \frac{n}{k}! \times \frac{n}{k}! \times \frac{n}{k}! \times \cdots \frac{n}{k}! = (\frac{n}{k}!)^k$$

Άσκηση 2 (α.2): Φράγμα ταξινόμησης

<u>Με συνδυαστική:</u> Αρκεί να μετρήσουμε τα διαφορετικά output που μπορεί να δημιουργήσει ο αλγόριθμος

- Για κάθε ομάδα των $\frac{n}{k}$ στοιχείων μπορεί να προκύψει οποιαδήποτε αναδιάταξη από τις $\frac{n}{k}!$
- Εφαρμόζουμε πολλαπλασιαστική αρχή για να συνδιάσουμε ανεξάρτητες ομάδες

$$L = \frac{n}{k}! \times \frac{n}{k}! \times \frac{n}{k}! \times \cdots \frac{n}{k}! = (\frac{n}{k}!)^k$$

- Και ομοίως με πριν $\Theta(\log L) = \Theta(n \log \frac{n}{k})$

Άσκηση 2(α): Γιατί να ταξινομήσεις με αυτό τον τρόπο?

Parallel Programming

Άσκηση 2(α): Γιατί να ταξινομήσεις με αυτό τον τρόπο?

Parallel Programming

Ας υποθέσουμε ότι έχεις στην διάθεση σου ένα πολυ-επεξεργαστικό σύστημα και θέλεις να φτιάξεις έναν εύκολο αλγόριθμο ταξινόμισης γιγαντιαίων υπερπινάκων.

Άσκηση 2(α): Γιατί να ταξινομήσεις με αυτό τον τρόπο?

Parallel Programming

Ας υποθέσουμε ότι έχεις στην διάθεση σου ένα πολυ-επεξεργαστικό σύστημα και θέλεις να φτιάξεις έναν εύκολο αλγόριθμο ταξινόμισης γιγαντιαίων υπερπινάκων. Βασικές αρχές παράλληλου προγραμματισμού:

1. Πρέπει να αποφασείς ποια κομμάτια θα κάνεις Broadcast σε ποιους πυρήνες.
2. Πρέπει η ολοκλήρωση κάθε task να μην απαιτεί επικοινωνία με άλλους πυρήνες.

Άσκηση 2(α): Γιατί να ταξινομήσεις με αυτό τον τρόπο?

Parallel Programming

Ας υποθέσουμε ότι έχεις στην διάθεση σου ένα πολυ-επεξεργαστικό σύστημα και θέλεις να φτιάξεις έναν εύκολο αλγόριθμο ταξινόμισης γιγαντιαίων υπερπινάκων. Βασικές αρχές παράλληλου προγραμματισμού:

1. Πρέπει να αποφασείς ποια κομμάτια θα κάνεις Broadcast σε ποιους πυρήνες.
2. Πρέπει η ολοκλήρωση κάθε task να μην απαιτεί επικοινωνία με άλλους πυρήνες.

Παρατήρηση: Ακόμη και ο διαχωρισμός σε k μη εσωτερικά ταξινομημένες ομάδες, πριν το Broadcast, λόγω της Divide & Conquer προσέγγισης μπορεί να παραλληλοποιηθεί.

Άσκηση 2 (β): Ταξινόμηση σε Πίνακα με Πολλά Ίδια Στοιχεία

<u>Είσοδος:</u> Πίνακας $A[1...n]$ με $O \log^d n$ διαφορετικά στοιχεία (d σταθερά).

Άσκηση 2 (β): Ταξινόμηση σε Πίνακα με Πολλά Ίδια Στοιχεία

Είσοδος: Πίνακας $A[1...n]$ με $O\log^d n$ διαφορετικά στοιχεία (d σταθερά).
Πρόβλημα: Ταξινόμηση σε χρόνο $O(n \log\log n)$.

Άσκηση 2 (β): Ταξινόμηση σε Πίνακα με Πολλά Ίδια Στοιχεία

Είσοδος: Πίνακας $A[1...n]$ με $O\log^d n$ διαφορετικά στοιχεία (d σταθερά).
Πρόβλημα: Ταξινόμηση σε χρόνο $O(n\log\log n)$.
Παρατηρήσεις

- Θέλουμε να εκμεταλλευτούμε το μικρό πλήθος διαφορετικών στοιχείων, ταξινομώντας στην ουσία μόνο αυτά, και στη συνέχεια "γεμίζοντας" τον πίνακα με τα αντίγραφα τους.

Άσκηση 2 (β): Ταξινόμηση σε Πίνακα με Πολλά Ίδια Στοιχεία

Είσοδος: Πίνακας $A[1...n]$ με $O\log^d n$ διαφορετικά στοιχεία (d σταθερά).
Πρόβλημα: Ταξινόμηση σε χρόνο $O(n\log\log n)$.
Παρατηρήσεις

- Θέλουμε να εκμεταλλευτούμε το μικρό πλήθος διαφορετικών στοιχείων, ταξινομώντας στην ουσία μόνο αυτά, και στη συνέχεια "γεμίζοντας" τον πίνακα με τα αντίγραφα τους.
- Αφού δεν ξέρουμε τις διακριτές τιμές όμως, πρέπει να τις εντοπίσουμε πρώτα.

Άσκηση 2 (β): Ταξινόμηση σε Πίνακα με Πολλά Ίδια Στοιχεία

Είσοδος: Πίνακας $A[1...n]$ με $O\log^d n$ διαφορετικά στοιχεία (d σταθερά).
Πρόβλημα: Ταξινόμηση σε χρόνο $O(n\log\log n)$.
Παρατηρήσεις

- Θέλουμε να εκμεταλλευτούμε το μικρό πλήθος διαφορετικών στοιχείων, ταξινομώντας στην ουσία μόνο αυτά, και στη συνέχεια "γεμίζοντας" τον πίνακα με τα αντίγραφα τους.
- Αφού δεν ξέρουμε τις διακριτές τιμές όμως, πρέπει να τις εντοπίσουμε πρώτα.
- Εδώ μπορούμε να χρησιμοποιήσουμε μια ουρά προτεραιότητας, έτσι ώστε να ελέγχουμε γρήγορα αν μια τιμή έχει ήδη βρεθεί ή όχι, και στη συνέχεια να ταξινομήσουμε, ή να τις ταξινομούμε "online", χρησιμοποιώντας insertion sort.

Άσκηση 2 (β): Ταξινόμηση σε Πίνακα με Πολλά Ίδια Στοιχεία

Λύση με insertion sort

- Χρησιμοποιούμε έναν απλό πίνακα μεθέδους $O(\log^d n)$, στον οποίο θα βάλουμε τις διακριτές τιμές, μαζί με τις συχνότητες εμφάνισης τους.

Άσκηση 2 (β): Ταξινόμηση σε Πίνακα με Πολλά Ίδια Στοιχεία

Λύση με insertion sort

- Χρησιμοποιούμε έναν απλό πίνακα μεθέδους $O(\log^d n)$, στον οποίο θα βάλουμε τις διακριτές τιμές, μαζί με τις συχνότητες εμφάνισης τους.
- Σε κάθε βήμα, με binary search (κόστους $(\log\log n)$) ελέγχουμε αν υπάρρχει ήδη η προς εξέταση τιμή, και αν ναι αυξάνουμε τον αντίστοιχο counter, διαφορετικά με κόστος $(\log^d n)$ την τοποθετούμε, κρατώντας sorted τον πίνακα.

Άσκηση 2 (β): Ταξινόμηση σε Πίνακα με Πολλά Ίδια Στοιχεία

Λύση με insertion sort

- Χρησιμοποιούμε έναν απλό πίνακα μεθέδους $O(\log^d n)$, στον οποίο θα βάλουμε τις διακριτές τιμές, μαζί με τις συχνότητες εμφάνισης τους.
- Σε κάθε βήμα, με binary search (κόστους $(\log\log n)$) ελέγχουμε αν υπάρρχει ήδη η προς εξέταση τιμή, και αν ναι αυξάνουμε τον αντίστοιχο counter, διαφορετικά με κόστος $(\log^d n)$ την τοποθετούμε, κρατώντας sorted τον πίνακα.
- Αφού όμως θα γίνουν $O(\log^d n)$ insertions, έχουμε συνολικό κόστος $O(n\log\log n) + O(\log^{2d} n) = O(n\log\log n)$.

Άσκηση 2 (β): Ταξινόμηση σε Πίνακα με Πολλά Ίδια Στοιχεία

Λύση με insertion sort

- Χρησιμοποιούμε έναν απλό πίνακα μεθέδους $O(\log^d n)$, στον οποίο θα βάλουμε τις διακριτές τιμές, μαζί με τις συχνότητες εμφάνισης τους.
- Σε κάθε βήμα, με binary search (κόστους $(\log \log n)$) ελέγχουμε αν υπάρρχει ήδη η προς εξέταση τιμή, και αν ναι αυξάνουμε τον αντίστοιχο counter, διαφορετικά με κόστος $(\log^d n)$ την τοποθετούμε, κρατώντας sorted τον πίνακα.
- Αφού όμως θα γίνουν $O(\log^d n)$ insertions, έχουμε συνολικό κόστος $O(n \log \log n) + O(\log^{2d} n) = O(n \log \log n)$.
- Η αντιγραφή στον αρχικό πίνακα θέλει χρόνο $\Theta(n)$ και άρα συνολικά έχουμε κόστος $O(n \log \log n)$.

Άσκηση 2 (β): Ταξινόμηση σε Πίνακα με Πολλά Ίδια Στοιχεία

Λύση με ουρά προτεραιότητας

- Καθώς έχουμε λίγα στοιχεία προς ταξινόμηση, η ταξινόμηση τους σε κάθε περίπτωση είναι γρήγορη. Οπότε η χρονική επιβάρυνση προκύπτει καθώς αναζητούμε τις ήδη υπάρχουσες διακριτές τιμές, για τον εντοπισμό των duplicates στον αρχικό πίνακα.

Άσκηση 2 (β): Ταξινόμηση σε Πίνακα με Πολλά Ίδια Στοιχεία

Λύση με ουρά προτεραιότητας

- Καθώς έχουμε λίγα στοιχεία προς ταξινόμηση, η ταξινόμηση τους σε κάθε περίπτωση είναι γρήγορη. Οπότε η χρονική επιβάρυνση προκύπτει καθώς αναζητούμε τις ήδη υπάρχουσες διακριτές τιμές, για τον εντοπισμό των duplicates στον αρχικό πίνακα.
- Για το λόγο αυτό, προσφέρεται να χρησιμοποιήσουμε μια ουρά προτεραιότητας που δίνει καλούς χρόνους αναζήτησης. Για παράδειγμα, μπορούμε να χρησιμοποιήσουμε κάποια δομή well-balanced binary search tree (AVL trees, Red-black trees, κ.ά.)

Άσκηση 2 (β): Ταξινόμηση σε Πίνακα με Πολλά Ίδια Στοιχεία

Ερώτηση: Γιατί δεν ισχύει το κάτω φράγμα $\Omega(n \log n)$;

Άσκηση 2 (β): Ταξινόμηση σε Πίνακα με Πολλά Ίδια Στοιχεία

Ερώτηση: Γιατί δεν ισχύει το κάτω φράγμα $\Omega(n \log n)$;
Απάντηση: Καθώς έχουμε πολλαπλές εμφανίσεις ίδιων στοιχείων, δε χρειάζεται να διακρίνουμε αυτές τις εμφανίσεις. Ουσιαστικά λοιπόν από τις $n!$ μεταθέσεις, θέλουμε να αγνοήσουμε τις μεταθέσεις των στοιχείων που έχουν ίδια τιμή.

Άσκηση 2 (β): Ταξινόμηση σε Πίνακα με Πολλά Ίδια Στοιχεία

Ερώτηση: Γιατί δεν ισχύει το κάτω φράγμα $\Omega(n \log n)$;
Απάντηση: Καθώς έχουμε πολλαπλές εμφανίσεις ίδιων στοιχείων, δε χρειάζεται να διακρίνουμε αυτές τις εμφανίσεις. Ουσιαστικά λοιπόν από τις $n!$ μεταθέσεις, θέλουμε να αγνοήσουμε τις μεταθέσεις των στοιχείων που έχουν ίδια τιμή.
Εξετάζοντας την περίπτωση που τα στοιχεία έχουν ισοκατανεμηθεί, έχουμε ότι το πλήθος των μεταθέσεων είναι

$$\frac{n!}{(\frac{n}{\log^d n}!)^{\log^d n}}$$

που δίνει ύψος του δέντρου συγκρίσεων $\Theta(n \log \log n)$). Άρα, ο αλγόριθμος μας είναι και βέλτιστος!

Outline

Άσκηση 3 (α): Εύρεση ελαχίστου διαστήματος

Είσοδος: Ταξινομημένοι πίνακες $A_1[1 \dots n_1], A_2[1 \dots n_2]$
Έξοδος: Θέσεις i_1, i_2 ώστε $|A_1[i_1] - A_2[i_2]|$ το ελάχιστο δυνατό

Άσκηση 3 (α): Εύρεση ελαχίστου διαστήματος

Είσοδος: Ταξινομημένοι πίνακες $A_1[1 \dots n_1], A_2[1 \dots n_2]$
Έξοδος: Θέσεις i_1, i_2 ώστε $|A_1[i_1] - A_2[i_2]|$ το ελάχιστο δυνατό
Λύση: Μετακίνηση Δεικτών

Άσκηση 3 (α): Εύρεση ελαχίστου διαστήματος

Είσοδος: Ταξινομημένοι πίνακες $A_1[1 \dots n_1], A_2[1 \dots n_2]$
Έξοδος: Θέσεις i_1, i_2 ώστε $|A_1[i_1] - A_2[i_2]|$ το ελάχιστο δυνατό
Λύση: Μετακίνηση Δεικτών

- Κρατάμε 2 δείκτες t_1, t_2 και αρχικοποιούμε $t_1 = 1, t_2 = 1$

Άσκηση 3 (α): Εύρεση ελαχίστου διαστήματος

Είσοδος: Ταξινομημένοι πίνακες $A_1[1 \dots n_1], A_2[1 \dots n_2]$
Έξοδος: Θέσεις i_1, i_2 ώστε $|A_1[i_1] - A_2[i_2]|$ το ελάχιστο δυνατό
Λύση: Μετακίνηση Δεικτών

- Κρατάμε 2 δείκτες t_1, t_2 και αρχικοποιούμε $t_1 = 1, t_2 = 1$
- Προχωράμε κατά 1 τον δείκτη που αντιστοιχεί στο μικρότερο στοιχείο

Άσκηση 3 (α): Εύρεση ελαχίστου διαστήματος

Είσοδος: Ταξινομημένοι πίνακες $A_1[1 \dots n_1], A_2[1 \dots n_2]$

Έξοδος: Θέσεις i_1, i_2 ώστε $|A_1[i_1] - A_2[i_2]|$ το ελάχιστο δυνατό

Λύση: Μετακίνηση Δεικτών

- Κρατάμε 2 δείκτες t_1, t_2 και αρχικοποιούμε $t_1 = 1, t_2 = 1$
- Προχωράμε κατά 1 τον δείκτη που αντιστοιχεί στο μικρότερο στοιχείο
- Επαναλαμβάνουμε μέχρι να εξαντληθούν τα στοιχεία των δύο πινάκων

Άσκηση 3 (α): Εύρεση ελαχίστου διαστήματος

Είσοδος: Ταξινομημένοι πίνακες $A_1[1 \dots n_1], A_2[1 \dots n_2]$
Έξοδος: Θέσεις i_1, i_2 ώστε $|A_1[i_1] - A_2[i_2]|$ το ελάχιστο δυνατό
Λύση: Μετακίνηση Δεικτών

- Κρατάμε 2 δείκτες t_1, t_2 και αρχικοποιούμε $t_1 = 1, t_2 = 1$
- Προχωράμε κατά 1 τον δείκτη που αντιστοιχεί στο μικρότερο στοιχείο
- Επαναλαμβάνουμε μέχρι να εξαντληθούν τα στοιχεία των δύο πινάκων
- Κατά τη διάσχιση κρατάμε τους δείκτες που αντιστοιχούν στο μικρότερο διάστημα

Άσκηση 3 (α): Εύρεση ελαχίστου διαστήματος

<u>Είσοδος</u>: Ταξινομημένοι πίνακες $A_1[1 \ldots n_1], A_2[1 \ldots n_2]$
<u>Έξοδος</u>: Θέσεις i_1, i_2 ώστε $|A_1[i_1] - A_2[i_2]|$ το ελάχιστο δυνατό
<u>Λύση</u>: Μετακίνηση Δεικτών

- Κρατάμε 2 δείκτες t_1, t_2 και αρχικοποιούμε $t_1 = 1, t_2 = 1$
- Προχωράμε κατά 1 τον δείκτη που αντιστοιχεί στο μικρότερο στοιχείο
- Επαναλαμβάνουμε μέχρι να εξαντληθούν τα στοιχεία των δύο πινάκων
- Κατά τη διάσχιση κρατάμε τους δείκτες που αντιστοιχούν στο μικρότερο διάστημα
- Επίστρέφουμε i_1, i_2 που αντιστοιχούν στο ελάχιστο διάστημα

Άσκηση 3 (α): Εύρεση ελαχίστου διαστήματος

Ορθότητα (ιδέα)

- Δεν έχει νόημα να αυξήσουμε το δείκτη του μεγαλύτερου στοιχείου ή να μειώσουμε το δείκτη του μικρότερου στοιχείου, αφού το εύρος του διαστήματος θα αυξηθεί

Άσκηση 3 (α): Εύρεση ελαχίστου διαστήματος

Ορθότητα (ιδέα)

- Δεν έχει νόημα να αυξήσουμε το δείκτη του μεγαλύτερου στοιχείου ή να μειώσουμε το δείκτη του μικρότερου στοιχείου, αφού το εύρος του διαστήματος θα αυξηθεί
- Επίσης από τον τρόπο διάσχισης και ταξινόμησης των στοιχειών δεν έχει νόημα να μειώσουμε τον δείκτη του μεγαλύτερου στοιχείου

Άσκηση 3 (α): Εύρεση ελαχίστου διαστήματος

Ορθότητα (ιδέα)

- Δεν έχει νόημα να αυξήσουμε το δείκτη του μεγαλύτερου στοιχείου ή να μειώσουμε το δείκτη του μικρότερου στοιχείου, αφού το εύρος του διαστήματος θα αυξηθεί
- Επίσης από τον τρόπο διάσχισης και ταξινόμησης των στοιχειών δεν έχει νόημα να μειώσουμε τον δείκτη του μεγαλύτερου στοιχείου
- Επομένως για να πετύχουμε ενδεχόμενη μείωση του διαστήματος αυξάνουμε τον δείκτη του μικρότερου στοιχείου

Άσκηση 3 (α): Εύρεση ελαχίστου διαστήματος

Ορθότητα Έστω i_1, i_2 οι θέσεις που ελαχιστοποιούν την ποσότητα με $A_1[i_1] \leq A_2[i_2]$. Θα δείξουμε ότι ο αλγόριθμός μας περνάει από αυτή την κατάσταση.

Άσκηση 3 (α): Εύρεση ελαχίστου διαστήματος

<u>Ορθότητα</u> Έστω i_1, i_2 οι θέσεις που ελαχιστοποιούν την ποσότητα με $A_1[i_1] \leq A_2[i_2]$. Θα δείξουμε ότι ο αλγόριθμός μας περνάει από αυτή την κατάσταση.

- Έστω ότι φτάνουμε πρώτα στο στοιχείο i_2. Τότε $t_1 = x < i_1$ (αφού φτάνουμε πρώτα στο i_2), όταν $t_2 = i_2$ (για πρώτη φορά).
  $x < i_1 \Rightarrow A_1[x] < A_1[i_1] \leq A_2[i_2]$ και άρα αυξάνοντας το μικρότερο, θα φτάσουμε στο i_1

Άσκηση 3 (α): Εύρεση ελαχίστου διαστήματος

Ορθότητα Έστω i_1, i_2 οι θέσεις που ελαχιστοποιούν την ποσότητα με $A_1[i_1] \leq A_2[i_2]$. Θα δείξουμε ότι ο αλγόριθμός μας περνάει από αυτή την κατάσταση.

- Έστω ότι φτάνουμε πρώτα στο στοιχείο i_2. Τότε $t_1 = x < i_1$ (αφού φτάνουμε πρώτα στο i_2), όταν $t_2 = i_2$ (για πρώτη φορά).
  $x < i_1 \Rightarrow A_1[x] < A_1[i_1] \leq A_2[i_2]$ και άρα αυξάνοντας το μικρότερο, θα φτάσουμε στο i_1
- Έστω ότι φτάνουμε πρώτα στο στοιχείο i_1. Τότε $t_2 = x < i_2$, όταν $t_1 = i_1$ (για πρώτη φορά). Αν κουνήσουμε τον t_1 πριν ο t_2 φτάσει στο i_2 (έστω $t_2 = y < i_2$ όταν συμβαίνει αυτό) $\Rightarrow A_1[i_1] < A_2[y] < A_2[i_2]$, άτοπο αφού θα είχαμε $A_2[y] - A_1[i_1] < A_2[i_2] - A_1[i_1]$

Άσκηση 3 (β): Ελαχιστοποίηση Ποσότητας

Είσοδος: Ταξινομημένοι πίνακες $A_1[1\dots n_1], A_2[1\dots n_2], \dots, A_m[1\dots m]$
Έξοδος: Θέσεις $i_1, i_2, \dots, i_m$ ώστε
$\max\{A_1[i_1], \dots, A_m[i_m]\} - \min\{A_1[i_1], \dots, A_m[i_m]\}$ το ελάχιστο δυνατό

Άσκηση 3 (β): Ελαχιστοποίηση Ποσότητας

Είσοδος: Ταξινομημένοι πίνακες $A_1[1 \dots n_1], A_2[1 \dots n_2], \dots, A_m[1 \dots m]$
Έξοδος: Θέσεις $i_1, i_2, \dots, i_m$ ώστε
$\max\{A_1[i_1], \dots, A_m[i_m]\} - \min\{A_1[i_1], \dots, A_m[i_m]\}$ το ελάχιστο δυνατό
Λύση: Μετακίνηση Δεικτών

Άσκηση 3 (β): Ελαχιστοποίηση Ποσότητας

Είσοδος: Ταξινομημένοι πίνακες $A_1[1 \dots n_1], A_2[1 \dots n_2], \dots, A_m[1 \dots m]$
Έξοδος: Θέσεις $i_1, i_2, \dots, i_m$ ώστε
$\max\{A_1[i_1], \dots, A_m[i_m]\} - \min\{A_1[i_1], \dots, A_m[i_m]\}$ το ελάχιστο δυνατό
Λύση: Μετακίνηση Δεικτών

- Παρατηρούμε ότι το να ελαχιστοποιήσουμε τη ζητούμενη ποσότητα είναι ισοδύναμο με το να βρούμε ένα διάστημα ελάχιστου μήκους που να περιέχει τουλάχιστον ένα στοιχείο από κάθε έναν από τους m πίνακες

Άσκηση 3 (β): Ελαχιστοποίηση Ποσότητας

Είσοδος: Ταξινομημένοι πίνακες $A_1[1 \dots n_1], A_2[1 \dots n_2], \dots, A_m[1 \dots m]$
Έξοδος: Θέσεις $i_1, i_2, \dots, i_m$ ώστε
$\max\{A_1[i_1], \dots, A_m[i_m]\} - \min\{A_1[i_1], \dots, A_m[i_m]\}$ το ελάχιστο δυνατό
Λύση: Μετακίνηση Δεικτών

- Παρατηρούμε ότι το να ελαχιστοποιήσουμε τη ζητούμενη ποσότητα είναι ισοδύναμο με το να βρούμε ένα διάστημα ελάχιστου μήκους που να περιέχει τουλάχιστον ένα στοιχείο από κάθε έναν από τους m πίνακες
- Επομένως μπορούμε να χρησιμοποιήσουμε την ίδια ιδέα που είχαμε και στο προηγούμενο ερώτημα

Άσκηση 3 (β): Ελαχιστοποίηση Ποσότητας

- Δημιουργούμε έναν πίνακα με m θέσεις, κάθε θέση i την αρχικοποιούμε με το πρώτο στοιχείο του A_i

Άσκηση 3 (β): Ελαχιστοποίηση Ποσότητας

- Δημιουργούμε έναν πίνακα με m θέσεις, κάθε θέση i την αρχικοποιούμε με το πρώτο στοιχείο του A_i
- Προχωράμε κατά 1 τον δείκτη που αντιστοιχεί στο μικρότερο στοιχείο

Άσκηση 3 (β): Ελαχιστοποίηση Ποσότητας

- Δημιουργούμε έναν πίνακα με m θέσεις, κάθε θέση i την αρχικοποιούμε με το πρώτο στοιχείο του A_i
- Προχωράμε κατά 1 τον δείκτη που αντιστοιχεί στο μικρότερο στοιχείο
- Επαναλαμβάνουμε μέχρι να εξαντληθούν τα στοιχεία όλων των πινάκων

Άσκηση 3 (β): Ελαχιστοποίηση Ποσότητας

- Δημιουργούμε έναν πίνακα με m θέσεις, κάθε θέση i την αρχικοποιούμε με το πρώτο στοιχείο του A_i
- Προχωράμε κατά 1 τον δείκτη που αντιστοιχεί στο μικρότερο στοιχείο
- Επαναλαμβάνουμε μέχρι να εξαντληθούν τα στοιχεία όλων των πινάκων
- Κατά τη διάσχιση κρατάμε τους δείκτες που αντιστοιχούν στο μικρότερο διάστημα

Άσκηση 3 (β): Ελαχιστοποίηση Ποσότητας

- Δημιουργούμε έναν πίνακα με m θέσεις, κάθε θέση i την αρχικοποιούμε με το πρώτο στοιχείο του A_i
- Προχωράμε κατά 1 τον δείκτη που αντιστοιχεί στο μικρότερο στοιχείο
- Επαναλαμβάνουμε μέχρι να εξαντληθούν τα στοιχεία όλων των πινάκων
- Κατά τη διάσχιση κρατάμε τους δείκτες που αντιστοιχούν στο μικρότερο διάστημα
- Επιστρέφουμε $i_1, \ldots, i_m$ που αντιστοιχούν στο ελάχιστο διάστημα

Άσκηση 3 (β): Ελαχιστοποίηση Ποσότητας

- Δημιουργούμε έναν πίνακα με m θέσεις, κάθε θέση i την αρχικοποιούμε με το πρώτο στοιχείο του A_i
- Προχωράμε κατά 1 τον δείκτη που αντιστοιχεί στο μικρότερο στοιχείο
- Επαναλαμβάνουμε μέχρι να εξαντληθούν τα στοιχεία όλων των πινάκων
- Κατά τη διάσχιση κρατάμε τους δείκτες που αντιστοιχούν στο μικρότερο διάστημα
- Επιστρέφουμε $i_1, \ldots, i_m$ που αντιστοιχούν στο ελάχιστο διάστημα

Πολυπλοκότητα: Σε κάθε επανάληψη θέλουμε χρόνο $O(m)$ για να βρούμε ελάχιστο/μέγιστο στοιχείο από τον πίνακα με τις m θέσεις, συνολικά κάνουμε $O(N)$ επαναλήψεις, άρα θέλουμε χρόνο (mN)

Άσκηση 3 (β): Ελαχιστοποίηση Ποσότητας

Ορθότητα

Έστω $i_1, \ldots, i_m$ οι ζητούμενοι δείκτες των πινάκων $A_1, \ldots, A_m$ αντίστοιχα, με $A_1[i_1] \leq A_2[i_2] \leq \ldots \leq A_m[i_m]$

Άσκηση 3 (β): Ελαχιστοποίηση Ποσότητας

Ορθότητα

Έστω $i_1, \ldots, i_m$ οι ζητούμενοι δείκτες των πινάκων $A_1, \ldots, A_m$ αντίστοιχα, με $A_1[i_1] \leq A_2[i_2] \leq \ldots \leq A_m[i_m]$

- Αν $t_1 \leq i_1 \implies t_j \leq i_j, \forall j \geq 2$, όπου t_j ο δείκτης στο τρέχον στοιχείο του πίνακα A_j (λόγω του τρόπου αύξησης των δεικτών και της διάταξης που θεωρήσαμε)

Άσκηση 3 (β): Ελαχιστοποίηση Ποσότητας

Ορθότητα

Έστω $i_1, \ldots, i_m$ οι ζητούμενοι δείκτες των πινάκων $A_1, \ldots, A_m$ αντίστοιχα, με $A_1[i_1] \leq A_2[i_2] \leq \ldots \leq A_m[i_m]$

- Αν $t_1 \leq i_1 \implies t_j \leq i_j, \forall j \geq 2$, όπου t_j ο δείκτης στο τρέχον στοιχείο του πίνακα A_j (λόγω του τρόπου αύξησης των δεικτών και της διάταξης που θεωρήσαμε)
- Όταν $t_1 = i_1$ θεωρούμε, χωρίς βλάβη της γενικότητας, ότι $A_1[i_1] = \min_j\{A_j[t_j]\}$ (αφού προχωράμε όλους τους δείκτες κάποια στιγμή θα φτάσει να είναι το min). Επίσης $\forall t_j, j \geq 2$, ισχύει $A_j[t_j] \leq A_j[i_j] \leq A_m[i_m] \implies \max_j\{A_j[t_j]\} \leq A_m[i_m]$
- Έτσι, το τρέχον διάστημα $(A_1[i_1], \max_j\{A_j[t_j]\})$ έχει εύρος $\leq OPT$. Επειδή δε γίνεται να είναι αυστηρά μικρότερο θα ισχύει η ισότητα

Άσκηση 3 (γ): Βελτίωση Επίδοσης

Πρόβλημα: Εύρεση ελάχιστου/μέγιστου στοιχείου του πίνακα δεικτών σε $O(m)$

Άσκηση 3 (γ): Βελτίωση Επίδοσης

Πρόβλημα: Εύρεση ελάχιστου/μέγιστου στοιχείου του πίνακα δεικτών σε $O(m)$

Λύση: Χρήση σωρού

Άσκηση 3 (γ): Βελτίωση Επίδοσης

Πρόβλημα: Εύρεση ελάχιστου/μέγιστου στοιχείου του πίνακα δεικτών σε $O(m)$

Λύση: Χρήση σωρού

- Θέλουμε χρόνο $O(m)$ για την αρχικοποίηση του σωρού

Άσκηση 3 (γ): Βελτίωση Επίδοσης

Πρόβλημα: Εύρεση ελάχιστου/μέγιστου στοιχείου του πίνακα δεικτών σε $O(m)$

Λύση: Χρήση σωρού

- Θέλουμε χρόνο $O(m)$ για την αρχικοποίηση του σωρού
- Για κάθε εύρεση ελαχίστου και για κάθε εισαγωγή νέου στοιχείου στο σωρό (από τη μετακίνηση του δείκτη) θέλουμε χρόνο $O(\log m)$

Άσκηση 3 (γ): Βελτίωση Επίδοσης

Πρόβλημα: Εύρεση ελάχιστου/μέγιστου στοιχείου του πίνακα δεικτών σε $O(m)$

Λύση: Χρήση σωρού

- Θέλουμε χρόνο $O(m)$ για την αρχικοποίηση του σωρού
- Για κάθε εύρεση ελαχίστου και για κάθε εισαγωγή νέου στοιχείου στο σωρό (από τη μετακίνηση του δείκτη) θέλουμε χρόνο $O(\log m)$
- Για την εύρεση του μέγιστου στοιχείου (ώστε να ελέγχουμε το τρέχον διάστημα σε κάθε επανάληψη) δε χρειάζεται σωρός, μπορούμε να υπολογίσουμε το αρχικό max σε χρόνο $O(m)$ και σε κάθε επανάληψη συγκρίνουμε το νέο στοιχείο με το παλιό max

Άσκηση 3 (γ): Βελτίωση Επίδοσης

Πρόβλημα: Εύρεση ελάχιστου/μέγιστου στοιχείου του πίνακα δεικτών σε $O(m)$

Λύση: Χρήση σωρού

- Θέλουμε χρόνο $O(m)$ για την αρχικοποίηση του σωρού
- Για κάθε εύρεση ελαχίστου και για κάθε εισαγωγή νέου στοιχείου στο σωρό (από τη μετακίνηση του δείκτη) θέλουμε χρόνο $O(\log m)$
- Για την εύρεση του μέγιστου στοιχείου (ώστε να ελέγχουμε το τρέχον διάστημα σε κάθε επανάληψη) δε χρειάζεται σωρός, μπορούμε να υπολογίσουμε το αρχικό max σε χρόνο $O(m)$ και σε κάθε επανάληψη συγκρίνουμε το νέο στοιχείο με το παλιό max
- Επομένως χρειαζόμαστε συνολικό χρόνο $O(N \log m)$

Outline

Άσκηση 4 (α): Μαγικό Φίλτρο

Είσοδος:
1000000 το πλήθος φιάλες με κωδικούς από το 1 ως το 1000000.

Άσκηση 4 (α): Μαγικό Φίλτρο

Είσοδος:
1000000 το πλήθος φιάλες με κωδικούς από το 1 ως το 1000000.
Έξοδος:
Εντοπισμός της μαγικής φιάλης με τους λιγότερους δυνατούς εθελοντές.

Άσκηση 4 (α): Μαγικό Φίλτρο

Λύση: Δυαδική Αναζήτηση. Μετατρέπουμε τους κωδικούς στην δυαδική αναπαράσταση των $\log_2 1000000 < 20$ bits

Άσκηση 4 (α): Μαγικό Φίλτρο

Λύση: Δυαδική Αναζήτηση. Μετατρέπουμε τους κωδικούς στην δυαδική αναπαράσταση των $\log_2 1000000 < 20$ bits

- Ένας εθελοντής θα δοκιμάσει από όλες τις φιάλες που έχουν το MSB 1.

Άσκηση 4 (α): Μαγικό Φίλτρο

Λύση: Δυαδική Αναζήτηση. Μετατρέπουμε τους κωδικούς στην δυαδική αναπαράσταση των $\log_2 1000000 < 20$ bits

- Ένας εθελοντής θα δοκιμάσει από όλες τις φιάλες που έχουν το MSB 1.
- Ένας εθελοντής θα δοκιμάσει από όλες τις φιάλες που έχουν το 2ο MSB 1.

Άσκηση 4 (α): Μαγικό Φίλτρο

Λύση: Δυαδική Αναζήτηση. Μετατρέπουμε τους κωδικούς στην δυαδική αναπαράσταση των $\log_2 1000000 < 20$ bits

- Ένας εθελοντής θα δοκιμάσει από όλες τις φιάλες που έχουν το MSB 1.
- Ένας εθελοντής θα δοκιμάσει από όλες τις φιάλες που έχουν το 2ο MSB 1.
- ...

Άσκηση 4 (α): Μαγικό Φίλτρο

Λύση: Δυαδική Αναζήτηση. Μετατρέπουμε τους κωδικούς στην δυαδική αναπαράσταση των $\log_2 1000000 < 20$ bits

- Ένας εθελοντής θα δοκιμάσει από όλες τις φιάλες που έχουν το MSB 1.
- Ένας εθελοντής θα δοκιμάσει από όλες τις φιάλες που έχουν το 2ο MSB 1.
- ...
- Ένας εθελοντής θα δοκιμάσει από όλες τις φιάλες που έχουν το LSB 1.

Άσκηση 4 (α): Μαγικό Φίλτρο

Λύση: Δυαδική Αναζήτηση. Μετατρέπουμε τους κωδικούς στην δυαδική αναπαράσταση των $\log_2 1000000 < 20$ bits

- Ένας εθελοντής θα δοκιμάσει από όλες τις φιάλες που έχουν το MSB 1.
- Ένας εθελοντής θα δοκιμάσει από όλες τις φιάλες που έχουν το 2ο MSB 1.
- ...
- Ένας εθελοντής θα δοκιμάσει από όλες τις φιάλες που έχουν το LSB 1.

Αν κάποιος εθελοντής έχει επηρεαστεί, τότε ξέρουμε ότι το bit που ανέλαβε στο κωδικό της μαγικής φιάλης ήταν 1.Διαφορετικά, ήταν 0.

Άσκηση 4 (α): Μαγικό Φίλτρο

Λύση: Δυαδική Αναζήτηση. Μετατρέπουμε τους κωδικούς στην δυαδική αναπαράσταση των $\log_2 1000000 < 20$ bits

- Ένας εθελοντής θα δοκιμάσει από όλες τις φιάλες που έχουν το MSB 1.
- Ένας εθελοντής θα δοκιμάσει από όλες τις φιάλες που έχουν το 2ο MSB 1.
- ...
- Ένας εθελοντής θα δοκιμάσει από όλες τις φιάλες που έχουν το LSB 1.

Αν κάποιος εθελοντής έχει επηρεαστεί, τότε ξέρουμε ότι το bit που ανέλαβε στο κωδικό της μαγικής φιάλης ήταν 1.Διαφορετικά, ήταν 0. Άρα με 20 εθελοντές έχουμε προσδιορίσει το κωδικό της μαγικής φιάλης.

Άσκηση 4 (β): Βελτιστοποίηση Ταξιδιού

Είσοδος: n το πλήθος πόλεις με διαδοχικές αποστάσεις $d_1, \ldots, d_n$ και το πλήθος ημερών k.

Άσκηση 4 (β): Βελτιστοποίηση Ταξιδιού

Είσοδος: n το πλήθος πόλεις με διαδοχικές αποστάσεις $d_1, \ldots, d_n$ και το πλήθος ημερών k.
Έξοδος: Χρονοπρογραμματισμός του ταξιδιού σε ημέρες & πόλεις ώστε να ελαχιστοποιήσουμε την μέγιστη ημερήσια απόσταση σταματώντας στο τέλος της κάθε ημέρας πάντα σε πόλη και ολοκληρώνοντας το ταξίδι σε k μέρες.

Άσκηση 4 (β): Βελτιστοποίηση Ταξιδιού

Είσοδος: n το πλήθος πόλεις με διαδοχικές αποστάσεις $d_1, \ldots, d_n$ και το πλήθος ημερών k.

Έξοδος: Χρονοπρογραμματισμός του ταξιδιού σε ημέρες & πόλεις ώστε να ελαχιστοποιήσουμε την μέγιστη ημερήσια απόσταση σταματώντας στο τέλος της κάθε ημέρας πάντα σε πόλη και ολοκληρώνοντας το ταξίδι σε k μέρες.

Λύση: Δυαδική Αναζήτηση για την εύρεση της ζητούμενης μέγιστης ημερήσιας απόστασης (Μ.Η.Α)

Άσκηση 4 (β): Βελτιστοποίηση Ταξιδιού

Λύση: Δυαδική Αναζήτηση για την εύρεση της ζητούμενης Μ.Η.Δ

Σε κάθε επανάληψη i ελέγχουμε αν μπορούμε να χωρίσουμε σε k μέρες το ταξίδι, διανύοντας συνολική απόσταση κάθε μέρα το πολύ $L_i \in \{\ell, \dots, r\}$ καλύπτοντας όλες τις πόλεις και σταματώντας πάντα στο τέλος της ημέρας σε πόλη, όπου $\{\ell, \dots, r\}$ είναι το διάστημα της αναζήτησης.

Λύση: Δυαδική Αναζήτηση για την εύρεση της ζητούμενης Μ.Η.Α

- Για κάθε πόλη j, ελέγχουμε αν αυτή χωράει στην τρέχουσα ημέρα, δηλαδή αν $sum + d_j \leq L_i$, όπου sum είναι ο τρέχον αριθμός χιλιομέτρων της ημέρας που εξετάζουμε.

Λύση: Δυαδική Αναζήτηση για την εύρεση της ζητούμενης Μ.Η.Α

- Για κάθε πόλη j, ελέγχουμε αν αυτή χωράει στην τρέχουσα ημέρα, δηλαδή αν $sum + d_j \leq L_i$, όπου sum είναι ο τρέχον αριθμός χιλιομέτρων της ημέρας που εξετάζουμε.
  - Αν πράγματι χωράει, την τοποθετούμε σε αυτή την ημέρα. Δηλαδή αυξάνουμε το sum κατά d_j και συνεχίζουμε με την πόλη $j + 1$.

Άσκηση 4 (β): Βελτιστοποίηση Ταξιδιού

Λύση: Δυαδική Αναζήτηση για την εύρεση της ζητούμενης Μ.Η.Α

- Για κάθε πόλη j, ελέγχουμε αν αυτή χωράει στην τρέχουσα ημέρα, δηλαδή αν $sum + d_j \leq L_i$, όπου sum είναι ο τρέχον αριθμός χιλιομέτρων της ημέρας που εξετάζουμε.
  - Αν πράγματι χωράει, την τοποθετούμε σε αυτή την ημέρα. Δηλαδή αυξάνουμε το sum κατά d_j και συνεχίζουμε με την πόλη $j + 1$.
  - Αν δε χωράει, τότε αλλάζουμε ημέρα. Δηλαδή αυξάνουμε κατά 1 το μετρητή των ημερών c, επαναρχικοποιούμε το $sum = 0$ και συνεχίζουμε ελέγχοντας αν η j χωράει στη νέα ημέρα.

Άσκηση 4 (β): Βελτιστοποίηση Ταξιδιού

Λύση: Δυαδική Αναζήτηση για την εύρεση της ζητούμενης Μ.Η.Α

- Για κάθε πόλη j, ελέγχουμε αν αυτή χωράει στην τρέχουσα ημέρα, δηλαδή αν $sum + d_j \leq L_i$, όπου sum είναι ο τρέχον αριθμός χιλιομέτρων της ημέρας που εξετάζουμε.
  - Αν πράγματι χωράει, την τοποθετούμε σε αυτή την ημέρα. Δηλαδή αυξάνουμε το sum κατά d_j και συνεχίζουμε με την πόλη $j + 1$.
  - Αν δε χωράει, τότε αλλάζουμε ημέρα. Δηλαδή αυξάνουμε κατά 1 το μετρητή των ημερών c, επαναρχικοποιούμε το $sum = 0$ και συνεχίζουμε ελέγχοντας αν η j χωράει στη νέα ημέρα.
- Αν όταν τελειώσουν οι πόλεις, έχουμε επιτύχει πλήθος ημερών $c \leq k$, τότε αναζητούμε μικρότερη μέγιστη ημερήσια απόσταση, θέτοντας $r = L_i$. Διαφορετικά, οι πόλεις δε χωράνε στις k ημέρες ταξιδιού με μέγιστη δυνατή ημερήσια απόσταση L_i και άρα αναζητούμε μεγαλύτερη, θέτοντας $\ell = L_i + 1$.

Άσκηση 4 (β): Βελτιστοποίηση Ταξιδιού

Λύση: Δυαδική Αναζήτηση για την εύρεση της ζητούμενης Μ.Η.Δ

Σε ποιό διάστημα $\{\ell, \ldots, r\}$ θα κάνουμε την αναζήτηση?

Άσκηση 4 (β): Βελτιστοποίηση Ταξιδιού

Λύση: Δυαδική Αναζήτηση για την εύρεση της ζητούμενης Μ.Η.Δ

Σε ποιό διάστημα $\{\ell, \ldots, r\}$ θα κάνουμε την αναζήτηση?

Κάποια πάνω όρια:

- Αν κάναμε όλο το ταξίδι σε μία μέρα, θα είχαμε διανύσει $r_1 = \sum_{i=1}^{n} d_i$, άρα $r \leq r_1$.

Άσκηση 4 (β): Βελτιστοποίηση Ταξιδιού

Λύση: Δυαδική Αναζήτηση για την εύρεση της ζητούμενης Μ.Η.Δ

Σε ποιό διάστημα $\{\ell, \ldots, r\}$ θα κάνουμε την αναζήτηση?

Κάποια πάνω όρια:

- Αν κάναμε όλο το ταξίδι σε μία μέρα, θα είχαμε διανύσει $r_1 = \sum_{i=1}^{n} d_i$, άρα $r \leq r_1$.
- Για να χρησιμοποιήσουμε όλες τις ημέρες, το λιγότερο που θα μπορούσαμε να κάνουμε είναι να βάλουμε από μία πόλη στις πρώτες $k-1$ μέρες και τις υπόλοιπες στην τελευταία, το οποίο θα μας έδινε μέγιστη διαδρομή ημέρας $r_2 = \max\{d_1, \ldots, d_{k-1}, \sum_{i=k}^{n} d_i\}$, άρα $r \leq r_2$.

Άσκηση 4 (β): Βελτιστοποίηση Ταξιδιού

Λύση: Δυαδική Αναζήτηση για την εύρεση της ζητούμενης Μ.Η.Δ

Σε ποιό διάστημα $\{\ell, \ldots, r\}$ θα κάνουμε την αναζήτηση?

Κάποια πάνω όρια:

- Αν κάναμε όλο το ταξίδι σε μία μέρα, θα είχαμε διανύσει $r_1 = \sum_{i=1}^{n} d_i$, άρα $r \leq r_1$.
- Για να χρησιμοποιήσουμε όλες τις ημέρες, το λιγότερο που θα μπορούσαμε να κάνουμε είναι να βάλουμε από μία πόλη στις πρώτες $k-1$ μέρες και τις υπόλοιπες στην τελευταία, το οποίο θα μας έδινε μέγιστη διαδρομή ημέρας $r_2 = \max\{d_1, \ldots, d_{k-1}, \sum_{i=k}^{n} d_i\}$, άρα $r \leq r_2$.

Διαλέγουμε ως r το ελάχιστο από τα δυνατά πάνω όρια.

Λύση: Δυαδική Αναζήτηση για την εύρεση της ζητούμενης Μ.Η.Δ

Κάποια κάτω όρια:

- Προφανώς, $\ell \geq 0$. Επειδή κάποια ημέρα σίγουρα θα περιέχει τη μακρύτερη διαδρομή μεταξύ πόλεων, $\ell \geq d_{\max}$.

Λύση: Δυαδική Αναζήτηση για την εύρεση της ζητούμενης Μ.Η.Δ

Κάποια κάτω όρια:

- Προφανώς, $\ell \geq 0$. Επειδή κάποια ημέρα σίγουρα θα περιέχει τη μακρύτερη διαδρομή μεταξύ πόλεων, $\ell \geq d_{\max}$.
- $\ell \geq \ell_1$, $\ell_1 = \frac{1}{k}(\sum_{i=1}^{n} d_i)$
  Διαφορετικά, το άθροισμα των χιλιομέτρων όλων των ημερών θα ήταν $< k\ell_1 = \sum_{i=1}^{n} d_i$, δηλαδή λιγότερο από το άθροισμα των χιλιομέτρων που όντως πρέπει να διανύσουμε στο ταξίδι.

Άσκηση 4 (β): Βελτιστοποίηση Ταξιδιού

Λύση: Δυαδική Αναζήτηση για την εύρεση της ζητούμενης Μ.Η.Δ

Κάποια κάτω όρια:

- Προφανώς, $\ell \geq 0$. Επειδή κάποια ημέρα σίγουρα θα περιέχει τη μακρύτερη διαδρομή μεταξύ πόλεων, $\ell \geq d_{\max}$.
- $\ell \geq \ell_1$, $\ell_1 = \frac{1}{k}(\sum_{i=1}^{n} d_i)$
  Διαφορετικά, το άθροισμα των χιλιομέτρων όλων των ημερών θα ήταν $< k\ell_1 = \sum_{i=1}^{n} d_i$, δηλαδή λιγότερο από το άθροισμα των χιλιομέτρων που όντως πρέπει να διανύσουμε στο ταξίδι.

Διαλέγουμε ως ℓ το μέγιστο από τα δυνατά κάτω όρια.

Λύση: Δυαδική Αναζήτηση για την εύρεση της ζητούμενης Μ.Η.Δ

Ο χρόνος εκτέλεσης είναι ο αριθμός των επαναλήψεων ($\log(\ell - r)$) επί το χρόνο που καταναλώνεται σε κάθε επανάληψη ($\Theta(n)$). Όποια όρια κι αν διαλέξουμε, $(\ell - r) = O(\sum_{i=1}^{n} d_i)$, αφού $\forall i, d_i \geq 1$ κι επομένως $\sum_{i=1}^{n} d_i \geq n$.

'Αρα η χρονική πολυπλοκότητα του αλγορίθμου μας είναι:

$$O(n \log(\sum_{i=1}^{n} d_i))$$

Outline

Άσκηση 5 (α): Επιλογή k-στού μικρότερου

Είσοδος: Αριθμός k, φράγμα M, πολυσύνολο S και "μαντείο" πλήθους μικρότερων στοιχείων, F_S.
Έξοδος: k-στό μικρότερο στοιχείο στο S

Άσκηση 5 (α): Επιλογή k-στού μικρότερου

Είσοδος: Αριθμός k, φράγμα M, πολυσύνολο S και "μαντείο" πλήθους μικρότερων στοιχείων, F_S.
Έξοδος: k-στό μικρότερο στοιχείο στο S
Λύση: Δυαδική Αναζήτηση

Άσκηση 5 (α): Επιλογή k-στού μικρότερου

Είσοδος: Αριθμός k, φράγμα M, πολυσύνολο S και "μαντείο" πλήθους μικρότερων στοιχείων, F_S.
Έξοδος: k-στό μικρότερο στοιχείο στο S
Λύση: Δυαδική Αναζήτηση

- Ρωτάμε πλήθος στοιχείων μικρότερα από $(M/2)$, $F_S(M/2)$

Άσκηση 5 (α): Επιλογή k-στού μικρότερου

Είσοδος: Αριθμός k, φράγμα M, πολυσύνολο S και "μαντείο" πλήθους μικρότερων στοιχείων, F_S.
Έξοδος: k-στό μικρότερο στοιχείο στο S
Λύση: Δυαδική Αναζήτηση

- Ρωτάμε πλήθος στοιχείων μικρότερα από $(M/2)$, $F_S(M/2)$
- Αν $F_S(M/2) < k$ συνεχίζουμε αναδρομικά στο $[M/2, M]$
- Αν $F_S(M/2) > k$ συνεχίζουμε αναδρομικά στο $[0, M/2]$

Άσκηση 5 (α): Επιλογή k-στού μικρότερου

<u>Είσοδος</u>: Αριθμός k, φράγμα M, πολυσύνολο S και "μαντείο" πλήθους μικρότερων στοιχείων, F_S.
<u>Έξοδος</u>: k-στό μικρότερο στοιχείο στο S
<u>Λύση: Δυαδική Αναζήτηση</u>

- Ρωτάμε πλήθος στοιχείων μικρότερα από $(M/2)$, $F_S(M/2)$
- Αν $F_S(M/2) < k$ συνεχίζουμε αναδρομικά στο $[M/2, M]$
- Αν $F_S(M/2) > k$ συνεχίζουμε αναδρομικά στο $[0, M/2]$
- Τελειώνουμε όταν βρούμε t τέτοιο ώστε $F_S(t-1) < k$ και $F_S(t) \geq k$

Άσκηση 5 (α): Επιλογή k-στού μικρότερου

Είσοδος: Αριθμός k, φράγμα M, πολυσύνολο S και "μαντείο" πλήθους μικρότερων στοιχείων, F_S.
Έξοδος: k-στό μικρότερο στοιχείο στο S
Λύση: Δυαδική Αναζήτηση

- Ρωτάμε πλήθος στοιχείων μικρότερα από $(M/2)$, $F_S(M/2)$
- Αν $F_S(M/2) < k$ συνεχίζουμε αναδρομικά στο $[M/2, M]$
- Αν $F_S(M/2) > k$ συνεχίζουμε αναδρομικά στο $[0, M/2]$
- Τελειώνουμε όταν βρούμε t τέτοιο ώστε $F_S(t-1) < k$ και $F_S(t) \geq k$
- Επιστρέφουμε το t σε χρόνο $O(\log M)$

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Είσοδος: Πίνακας διακεκριμένων ακεραίων $A[1 \ldots n]$, το μέγιστο στοιχείο του M και αριθμός k.
Έξοδος: k-στό μικρότερο στοιχείο στον πίνακα θετικών διαφορών του A

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Είσοδος: Πίνακας διακεκριμένων ακεραίων $A[1 \dots n]$, το μέγιστο στοιχείο του M και αριθμός k.
Έξοδος: k-στό μικρότερο στοιχείο στον πίνακα θετικών διαφορών του A
Λύση: Θα υλοποιήσουμε το μαντείο και θα εφαρμόσουμε το (α)

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Ιδέα

$$A = \begin{pmatrix} a_1 & a_2 & \ldots & a_i & \ldots & a_j & \ldots & a_n \end{pmatrix}$$

- Αν A ταξινομημένος και (π.χ.) $a_i - a_1 \leq \delta < a_{i+1} - a_1$ τότε

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Ιδέα

$$A = \begin{pmatrix} a_1 & a_2 & \ldots & a_i & \ldots & a_j & \ldots & a_n \end{pmatrix}$$

- Αν A ταξινομημένος και (π.χ.) $a_i - a_1 \leq \delta < a_{i+1} - a_1$ τότε
  - $a_k - a_1 \leq \delta, \quad \forall i,\ 1 < k \leq i$

Ιδέα

$$A = \begin{pmatrix} a_1 & a_2 & \dots & a_i & \dots & a_j & \dots & a_n \end{pmatrix}$$

- Αν A ταξινομημένος και (π.χ.) $a_i - a_1 \leq \delta < a_{i+1} - a_1$ τότε
  - $a_k - a_1 \leq \delta, \quad \forall i,\ 1 < k \leq i$
  - $a_k - a_1 > \delta, \quad \forall i,\ i < k \leq n$

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Ιδέα

$$A = \begin{pmatrix} a_1 & a_2 & \dots & a_i & \dots & a_j & \dots & a_n \end{pmatrix}$$

- Αν Α ταξινομημένος και (π.χ.) $a_i - a_1 \leq \delta < a_{i+1} - a_1$ τότε
  - $a_k - a_1 \leq \delta, \quad \forall i,\ 1 < k \leq i$
  - $a_k - a_1 > \delta, \quad \forall i,\ i < k \leq n$
- Άρα υπάρχουν $i - 1$ διαφορές $\leq \delta$ που προκύπτουν από το a_1.

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Ιδέα

$$A = \begin{pmatrix} a_1 & a_2 & \dots & a_i & \dots & a_j & \dots & a_n \end{pmatrix}$$

- Αν A ταξινομημένος και (π.χ.) $a_i - a_1 \leq \delta < a_{i+1} - a_1$ τότε
  - $a_k - a_1 \leq \delta, \quad \forall i,\ 1 < k \leq i$
  - $a_k - a_1 > \delta, \quad \forall i,\ i < k \leq n$
- Άρα υπάρχουν $i - 1$ διαφορές $\leq \delta$ που προκύπτουν από το a_1.
- Αν αντί για a_1, ψάχνω τις διαφορές που προκύπτουν από το a_2 πως προκύπτει το αντίστοιχο j ώστε $a_j - a_2 \leq \delta < a_{j+1} - a_2$?

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Ιδέα

$$A = \left(\begin{array}{ccccccc} a_1 & \color{blue}{a_2} & \dots & a_i & \dots & \color{blue}{a_j} & \dots & a_n \end{array} \right)$$

- Αν A ταξινομημένος και (π.χ.) $a_i - a_1 \leq \delta < a_{i+1} - a_1$ τότε
  - $a_k - a_1 \leq \delta, \quad \forall i,\ 1 < k \leq i$
  - $a_k - a_1 > \delta, \quad \forall i,\ i < k \leq n$
- Άρα υπάρχουν $i - 1$ διαφορές $\leq \delta$ που προκύπτουν από το a_1.
- Αν αντί για a_1, ψάχνω τις διαφορές που προκύπτουν από το a_2 πως προκύπτει το αντίστοιχο j ώστε $a_j - a_2 \leq \delta < a_{j+1} - a_2$?
- Προφανώς $j \geq i$ αφού αν $a_j > a_i \Rightarrow a_j - a_2 > a_i - a_2$

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Ιδέα

$$A = \begin{pmatrix} a_1 & \color{blue}{a_2} & \dots & a_i & \dots & \color{blue}{a_j} & \dots & a_n \end{pmatrix}$$

- Αν A ταξινομημένος και (π.χ.) $a_i - a_1 \leq \delta < a_{i+1} - a_1$ τότε
  - $a_k - a_1 \leq \delta, \quad \forall i,\ 1 < k \leq i$
  - $a_k - a_1 > \delta, \quad \forall i,\ i < k \leq n$
- Άρα υπάρχουν $i - 1$ διαφορές $\leq \delta$ που προκύπτουν από το a_1.
- Αν αντί για a_1, ψάχνω τις διαφορές που προκύπτουν από το a_2 πως προκύπτει το αντίστοιχο j ώστε $a_j - a_2 \leq \delta < a_{j+1} - a_2$?
- Προφανώς $j \geq i$ αφού αν $a_j > a_i \Rightarrow a_j - a_2 > a_i - a_2$
- Άρα αρκεί να ψάξω τα επόμενα στοιχεία, το οποίο μου επιτρέπει να διατρέξω τον πίνακα μόνο μία φορά.

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Υλοποίηση Oracle

function ORACLE(int a[1..n], int k) ▷ array a has to be sorted

$i \leftarrow 1$, $j \leftarrow 1$, $sum \leftarrow 0$

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Υλοποίηση Oracle

function ORACLE(int a[1..n], int k) ▷ array a has to be sorted

$i \leftarrow 1$, $j \leftarrow 1$, $sum \leftarrow 0$

while $j \leq n$ & $a_j - a_i \leq k$ **do**

$j \leftarrow j+1$

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Υλοποίηση Oracle

function ORACLE(int a[1..n], int k) $\triangleright$ array a has to be sorted

$i \leftarrow 1,\ j \leftarrow 1,\ sum \leftarrow 0$

while $j \leq n\ \ \&\ \ a_j - a_i \leq k$ **do**

$j \leftarrow j + 1$

$sum \leftarrow sum + (j-1) - i$

Υλοποίηση Oracle

```
function ORACLE(int a[1..n], int k)        ▷ array a has to be sorted
    i ← 1, j ← 1, sum ← 0
    while i < n do
        while j ≤ n & a_j − a_i ≤ k do
            j ← j + 1
        sum ← sum + (j − 1) − i
        i ← i + 1
```

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Υλοποίηση Oracle

function ORACLE(int a[1..n], int k) ▷ array a has to be sorted

- $i \leftarrow 1$, $j \leftarrow 1$, $sum \leftarrow 0$
- **while** $i < n$ **do**
  - **while** $j \leq n$ & $a_j - a_i \leq k$ **do**
    - $j \leftarrow j + 1$
  - $sum \leftarrow sum + (j - 1) - i$
  - $i \leftarrow i + 1$
- **return** sum

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Λύση

- Ταξινόμηση πίνακα
- Δυαδική αναζήτηση με χρήση Oracle όπως 5(α)

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Λύση

- Ταξινόμηση πίνακα
- Δυαδική αναζήτηση με χρήση Oracle όπως 5(α)

Χρονική Πολυπλοκότητα

- Αρχικό sorting του πίνακα: $(n \log n)$

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Λύση

- Ταξινόμηση πίνακα
- Δυαδική αναζήτηση με χρήση Oracle όπως 5(α)

Χρονική Πολυπλοκότητα

- Αρχικό sorting του πίνακα: $(n \log n)$
- Κλήση oracle: (n)

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Λύση

- Ταξινόμηση πίνακα
- Δυαδική αναζήτηση με χρήση Oracle όπως 5(α)

Χρονική Πολυπλοκότητα

- Αρχικό sorting του πίνακα: $(n \log n)$
- Κλήση oracle: (n)
- Πλήθος κλήσεων oracle: $(\log(a_{max} - a_{min}))$

Άσκηση 5 (β): Επιλογή k-στης διαφοράς

Λύση

- Ταξινόμηση πίνακα
- Δυαδική αναζήτηση με χρήση Oracle όπως 5(α)

Χρονική Πολυπλοκότητα

- Αρχικό sorting του πίνακα: $(n \log n)$
- Κλήση oracle: (n)
- Πλήθος κλήσεων oracle: $(\log(a_{max} - a_{min}))$

Συνολικά:

$$(n \cdot \log(max\{n, a_{max} - a_{min}\}))$$

Outline

Προγραμματιστική 1 - Λύση $O(N^2)$

Είσοδος: Πίνακας υψών $H[1 \dots N]$
Έξοδος: Ελάχιστο κόστος για όλους τους ουρανοξύστες

Προγραμματιστική 1 - Λύση $O(N^2)$

Είσοδος: Πίνακας υψών $H[1 \dots N]$
Έξοδος: Ελάχιστο κόστος για όλους τους ουρανοξύστες
Λύση: Δοκιμάζουμε σαν υποψήφια λύση κάθε ουρανοξύστη i, με ύψος $H[i]$. Κάνοντας ένα γραμμικό πέρασμα ξεκινώντας απ' την θέση i και κατευθυνόμενοι προς τα πίσω, κρατώντας κάθε φορά το μέγιστο, υπολογίζουμε τα κόστη όλων των πολυκατοικιών σε θέση $j < i$ και αντίστοιχα για την άλλη κατεύθυνση.

Προγραμματιστική 1 - Λύση $O(N)$

- Συμβολίζουμε $CL[i]$ το κόστος που προκύπτει αν επιλέξουμε τον $i-$οστό ουρανοξύστη για όλους τους ουρανοξύστες στα αριστερά. Αντίστοιχα συμβολίζουμε $CR[i]$ το κόστος που προκύπτει για τους ουρανοξύστες στα δεξιά

Προγραμματιστική 1 - Λύση $O(N)$

- Συμβολίζουμε $CL[i]$ το κόστος που προκύπτει αν επιλέξουμε τον $i-$οστό ουρανοξύστη για όλους τους ουρανοξύστες στα αριστερά. Αντίστοιχα συμβολίζουμε $CR[i]$ το κόστος που προκύπτει για τους ουρανοξύστες στα δεξιά
- Η λύση είναι $max_i\{CL[i] + CR[i] - H[i]\}$

Προγραμματιστική 1 - Λύση $O(N)$

- Συμβολίζουμε $CL[i]$ το κόστος που προκύπτει αν επιλέξουμε τον i–οστό ουρανοξύστη για όλους τους ουρανοξύστες στα αριστερά. Αντίστοιχα συμβολίζουμε $CR[i]$ το κόστος που προκύπτει για τους ουρανοξύστες στα δεξιά
- Η λύση είναι $max_i\{CL[i] + CR[i] - H[i]\}$
- Θα δείξουμε πώς υπολογίζουμε το $CL[i]$ για κάθε i σε γραμμικό χρόνο και συνεπώς και το $CR[i]$ με τον ίδιο ακριβώς τρόπο

Προγραμματιστική 1 - Λύση $O(N)$

- Συμβολίζουμε $CL[i]$ το κόστος που προκύπτει αν επιλέξουμε τον i–οστό ουρανοξύστη για όλους τους ουρανοξύστες στα αριστερά. Αντίστοιχα συμβολίζουμε $CR[i]$ το κόστος που προκύπτει για τους ουρανοξύστες στα δεξιά
- Η λύση είναι $max_i\{CL[i] + CR[i] - H[i]\}$
- Θα δείξουμε πώς υπολογίζουμε το $CL[i]$ για κάθε i σε γραμμικό χρόνο και συνεπώς και το $CR[i]$ με τον ίδιο ακριβώς τρόπο
- Παρατηρούμε ότι αν βρισκόμαστε στην θέση i και έχουμε υπολογίσει το $CL[j]$ για κάθε $j < i$ ισχύει η σχέση $CL[i] = CL[k] + (i - k)A[i]$, όπου k η πρώτη θέση αν ξεκινήσουμε από το i κατευθυνόμενοι προς τα αριστερά όπου $H[k] > H[i]$

Προγραμματιστική 1 - Λύση $O(N)$

- Συμβολίζουμε $CL[i]$ το κόστος που προκύπτει αν επιλέξουμε τον i−οστό ουρανοξύστη για όλους τους ουρανοξύστες στα αριστερά. Αντίστοιχα συμβολίζουμε $CR[i]$ το κόστος που προκύπτει για τους ουρανοξύστες στα δεξιά
- Η λύση είναι $max_i\{CL[i] + CR[i] - H[i]\}$
- Θα δείξουμε πώς υπολογίζουμε το $CL[i]$ για κάθε i σε γραμμικό χρόνο και συνεπώς και το $CR[i]$ με τον ίδιο ακριβώς τρόπο
- Παρατηρούμε ότι αν βρισκόμαστε στην θέση i και έχουμε υπολογίσει το $CL[j]$ για κάθε $j < i$ ισχύει η σχέση $CL[i] = CL[k] + (i - k)A[i]$, όπου k η πρώτη θέση αν ξεκινήσουμε από το i κατευθυνόμενοι προς τα αριστερά όπου $H[k] > H[i]$
- Το πρόβλημά μας έχει αναχθεί στο να μπορούμε να βρούμε για κάθε i το αμέσως προηγούμενο σημείο k, όπου $H[k] > H[i]$

Προγραμματιστική 1 - Λύση $O(N)$

- Διατρέχουμε τον πίνακα από αριστερά προς τα αριστερά διατηρώντας μια στοίβα την οποία κατασκευάζουμε ως εξής: Όταν βρισκόμαστε στην θέση $i = 1$ η στοίβα περιλαμβάνει μόνο το ύψος του πρώτου ουρανοξύστη. Για κάθε $i > 1$ η στοίβα κατασκευάζεται βάσει αυτής που έχουμε υπολογίσει στην προηγούμενη επανάληψη με τον παρακάτω αλγόριθμο

Προγραμματιστική 1 - Λύση $O(N)$

- Διατρέχουμε τον πίνακα από αριστερά προς τα αριστερά διατηρώντας μια στοίβα την οποία κατασκευάζουμε ως εξής: Όταν βρισκόμαστε στην θέση $i = 1$ η στοίβα περιλαμβάνει μόνο το ύψος του πρώτου ουρανοξύστη. Για κάθε $i > 1$ η στοίβα κατασκευάζεται βάσει αυτής που έχουμε υπολογίσει στην προηγούμενη επανάληψη με τον παρακάτω αλγόριθμο
- 1. Όσο το στοιχείο που βρίσκεται στην κορυφή της λίστας είναι μικρότερο ή ίσο σε τιμή από το $H[i]$ αφαίρεσέ το απ' την στοίβα.

Προγραμματιστική 1 - Λύση $O(N)$

- Διατρέχουμε τον πίνακα από αριστερά προς τα αριστερά διατηρώντας μια στοίβα την οποία κατασκευάζουμε ως εξής: Όταν βρισκόμαστε στην θέση $i = 1$ η στοίβα περιλαμβάνει μόνο το ύψος του πρώτου ουρανοξύστη. Για κάθε $i > 1$ η στοίβα κατασκευάζεται βάσει αυτής που έχουμε υπολογίσει στην προηγούμενη επανάληψη με τον παρακάτω αλγόριθμο
- 1. Όσο το στοιχείο που βρίσκεται στην κορυφή της λίστας είναι μικρότερο ή ίσο σε τιμή από το $H[i]$ αφαίρεσέ το απ' την στοίβα.
- 2. Βάλε το $H[i]$ στην κορυφή της στοίβας.

Προγραμματιστική 1 - Λύση $O(N)$

- Διατρέχουμε τον πίνακα από αριστερά προς τα αριστερά διατηρώντας μια στοίβα την οποία κατασκευάζουμε ως εξής: Όταν βρισκόμαστε στην θέση $i = 1$ η στοίβα περιλαμβάνει μόνο το ύψος του πρώτου ουρανοξύστη. Για κάθε $i > 1$ η στοίβα κατασκευάζεται βάσει αυτής που έχουμε υπολογίσει στην προηγούμενη επανάληψη με τον παρακάτω αλγόριθμο
- 1. Όσο το στοιχείο που βρίσκεται στην κορυφή της λίστας είναι μικρότερο ή ίσο σε τιμή από το $H[i]$ αφαίρεσέ το απ' την στοίβα.
- 2. Βάλε το $H[i]$ στην κορυφή της στοίβας.
- Η στοίβα σε κάθε σημείο βρίσκεται ταξινομημένη σε φθίνουσα σειρά, το στοιχείο που βρίσκεται κάτω απ' το $H[i]$ είναι το ζητούμενο.

Προγραμματιστική 1 - Λύση $O(N)$

- Διατρέχουμε τον πίνακα από αριστερά προς τα αριστερά διατηρώντας μια στοίβα την οποία κατασκευάζουμε ως εξής: Όταν βρισκόμαστε στην θέση $i = 1$ η στοίβα περιλαμβάνει μόνο το ύψος του πρώτου ουρανοξύστη. Για κάθε $i > 1$ η στοίβα κατασκευάζεται βάσει αυτής που έχουμε υπολογίσει στην προηγούμενη επανάληψη με τον παρακάτω αλγόριθμο
- 1. Όσο το στοιχείο που βρίσκεται στην κορυφή της λίστας είναι μικρότερο ή ίσο σε τιμή από το $H[i]$ αφαίρεσέ το απ' την στοίβα.
- 2. Βάλε το $H[i]$ στην κορυφή της στοίβας.
- Η στοίβα σε κάθε σημείο βρίσκεται ταξινομημένη σε φθίνουσα σειρά, το στοιχείο που βρίσκεται κάτω απ' το $H[i]$ είναι το ζητούμενο.
- Ο παραπάνω αλγόριθμος έχει γραμμική πολυπλοκότητα αφού κάθε στοιχείο μπαίνει στην στοίβα μια φορά και βγαίνει το πολύ μια φορά. Συνεπώς τα συνολικά operations που θα κάνουμε είναι το πολύ 2N.

Προγραμματιστική 1 - Λύση $O(N)$

- Διατρέχουμε τον πίνακα από αριστερά προς τα αριστερά διατηρώντας μια στοίβα την οποία κατασκευάζουμε ως εξής: Όταν βρισκόμαστε στην θέση $i = 1$ η στοίβα περιλαμβάνει μόνο το ύψος του πρώτου ουρανοξύστη. Για κάθε $i > 1$ η στοίβα κατασκευάζεται βάσει αυτής που έχουμε υπολογίσει στην προηγούμενη επανάληψη με τον παρακάτω αλγόριθμο
- 1. Όσο το στοιχείο που βρίσκεται στην κορυφή της λίστας είναι μικρότερο ή ίσο σε τιμή από το $H[i]$ αφαίρεσέ το απ' την στοίβα.
- 2. Βάλε το $H[i]$ στην κορυφή της στοίβας.
- Η στοίβα σε κάθε σημείο βρίσκεται ταξινομημένη σε φθίνουσα σειρά, το στοιχείο που βρίσκεται κάτω απ' το $H[i]$ είναι το ζητούμενο.
- Ο παραπάνω αλγόριθμος έχει γραμμική πολυπλοκότητα αφού κάθε στοιχείο μπαίνει στην στοίβα μια φορά και βγαίνει το πολύ μια φορά. Συνεπώς τα συνολικά operations που θα κάνουμε είναι το πολύ 2N.
- Όμοια για το $CR[i]$, για κάθε i εκτελούμε τον ίδιο αλγόριθμο ξεκινώντας απλά από το τέλος του πίνακα προς την αρχή.

Outline

Προγραμματιστική 2

Είσοδος: N ζεύγη (t_{a_i}, v_{a_i}), N ζεύγη (t_{b_i}, v_{b_i}) για τα σωματίδια a, b
Έξοδος: K ζεύγη συγκρούσεων μεταξύ σωματιδίων a, b

Προγραμματιστική 2

Είσοδος: N ζεύγη (t_{a_i}, v_{a_i}), N ζεύγη (t_{b_i}, v_{b_i}) για τα σωματίδια a, b

Έξοδος: K ζεύγη συγκρούσεων μεταξύ σωματιδίων a, b

Λύση:

- Είναι εύκολο να βρούμε λύση $O(N^2)$ υπολογίζοντας όλες τις συγκρούσεις των σωματιδίων και επιλέγοντας τις πρώτες K

Προγραμματιστική 2

<u>Είσοδος</u>: N ζεύγη (t_{a_i}, v_{a_i}), N ζεύγη (t_{b_i}, v_{b_i}) για τα σωματίδια a, b
<u>Έξοδος</u>: K ζεύγη συγκρούσεων μεταξύ σωματιδίων a, b
<u>Λύση</u>:

- Είναι εύκολο να βρούμε λύση $O(N^2)$ υπολογίζοντας όλες τις συγκρούσεις των σωματιδίων και επιλέγοντας τις πρώτες K
- Χωρίζουμε το πρόβλημα σε K γύρους και σε κάθε γύρο ψάχνουμε την πρώτη σύγκρουση, έπειτα αφαιρούμε τα σωματίδια που συμμετέχουν

Προγραμματιστική 2

Είσοδος: N ζεύγη (t_{a_i}, v_{a_i}), N ζεύγη (t_{b_i}, v_{b_i}) για τα σωματίδια a, b
Έξοδος: K ζεύγη συγκρούσεων μεταξύ σωματιδίων a, b
Λύση:

- Είναι εύκολο να βρούμε λύση $O(N^2)$ υπολογίζοντας όλες τις συγκρούσεις των σωματιδίων και επιλέγοντας τις πρώτες K
- Χωρίζουμε το πρόβλημα σε K γύρους και σε κάθε γύρο ψάχνουμε την πρώτη σύγκρουση, έπειτα αφαιρούμε τα σωματίδια που συμμετέχουν
- Αν κάποια χρονική στιγμή σωματίδιο-a δεξιά σωματιδίου-b ξέρουμε πώς κάποια στιγμή πριν έγινε σύγκρουση. Αν όλα τα a βρίσκονται αριστερά των b τότε δεν έγινε

Προγραμματιστική 2

Είσοδος: N ζεύγη (t_{a_i}, v_{a_i}), N ζεύγη (t_{b_i}, v_{b_i}) για τα σωματίδια a, b
Έξοδος: K ζεύγη συγκρούσεων μεταξύ σωματιδίων a, b
Λύση:

- Είναι εύκολο να βρούμε λύση $O(N^2)$ υπολογίζοντας όλες τις συγκρούσεις των σωματιδίων και επιλέγοντας τις πρώτες K
- Χωρίζουμε το πρόβλημα σε K γύρους και σε κάθε γύρο ψάχνουμε την πρώτη σύγκρουση, έπειτα αφαιρούμε τα σωματίδια που συμμετέχουν
- Αν κάποια χρονική στιγμή σωματίδιο-a δεξιά σωματιδίου-b ξέρουμε πώς κάποια στιγμή πριν έγινε σύγκρουση. Αν όλα τα a βρίσκονται αριστερά των b τότε δεν έγινε
- Όταν θέλουμε να ελέγξουμε αν έχει γίνει σύγκρουση στο διάστημα $(0, t]$ χρησιμοποιούμε την παραπάνω συνθήκη ελέγχου και κάνουμε δυαδική αναζήτηση

Προγραμματιστική 2

Είσοδος: N ζεύγη (t_{a_i}, v_{a_i}), N ζεύγη (t_{b_i}, v_{b_i}) για τα σωματίδια a, b
Έξοδος: K ζεύγη συγκρούσεων μεταξύ σωματιδίων a, b
Λύση:

- Είναι εύκολο να βρούμε λύση $O(N^2)$ υπολογίζοντας όλες τις συγκρούσεις των σωματιδίων και επιλέγοντας τις πρώτες K
- Χωρίζουμε το πρόβλημα σε K γύρους και σε κάθε γύρο ψάχνουμε την πρώτη σύγκρουση, έπειτα αφαιρούμε τα σωματίδια που συμμετέχουν
- Αν κάποια χρονική στιγμή σωματίδιο-a δεξιά σωματιδίου-b ξέρουμε πώς κάποια στιγμή πριν έγινε σύγκρουση. Αν όλα τα a βρίσκονται αριστερά των b τότε δεν έγινε
- Όταν θέλουμε να ελέγξουμε αν έχει γίνει σύγκρουση στο διάστημα $(0, t]$ χρησιμοποιούμε την παραπάνω συνθήκη ελέγχου και κάνουμε δυαδική αναζήτηση
- Κάθε γύρο θέλουμε χρόνο $O(N \log L)$ (ο μέγιστος χρόνος δε μπορεί να ξεπερνάει το L), άρα συνολικά $O(KN \log L)$

Προγραμματιστική 2 - Bonus

<u>Είσοδος</u>: N ζεύγη (t_{a_i}, v_{a_i}), N ζεύγη (t_{b_i}, v_{b_i}) για τα σωματίδια a, b
<u>Έξοδος</u>: K ζεύγη συγκρούσεων μεταξύ σωματιδίων a, b

Προγραμματιστική 2 - Bonus

Είσοδος: N ζεύγη (t_{a_i}, v_{a_i}), N ζεύγη (t_{b_i}, v_{b_i}) για τα σωματίδια a, b
Έξοδος: K ζεύγη συγκρούσεων μεταξύ σωματιδίων a, b
Λύση:

- Αναπαριστούμε τις κινήσεις των σωματιδίων σαν ευθείες στο επίπεδο, $a : x = v_i t - v_i t_i, b : x = -v_i t + L + v_i t_i$

Προγραμματιστική 2 - Bonus

Είσοδος: N ζεύγη (t_{a_i}, v_{a_i}), N ζεύγη (t_{b_i}, v_{b_i}) για τα σωματίδια a, b
Έξοδος: K ζεύγη συγκρούσεων μεταξύ σωματιδίων a, b
Λύση:

- Αναπαριστούμε τις κινήσεις των σωματιδίων σαν ευθείες στο επίπεδο, $a : x = v_i t - v_i t_i$, $b : x = -v_i t + L + v_i t_i$
- Απ' αυτές κρατάμε το convex hull σε αύξουσα σειρά κλίσης. Το πρόβλημα πλέον είναι, δεδομένων 2 συνόλων ευθειών, να βρούμε τις 2 που δε συμμετέχουν στο ίδιο σύνολο και έχουν τη μικρότερη τεταγμένη τομής

Προγραμματιστική 2 - Bonus

Είσοδος: N ζεύγη (t_{a_i}, v_{a_i}), N ζεύγη (t_{b_i}, v_{b_i}) για τα σωματίδια a, b
Έξοδος: K ζεύγη συγκρούσεων μεταξύ σωματιδίων a, b
Λύση:

- Αναπαριστούμε τις κινήσεις των σωματιδίων σαν ευθείες στο επίπεδο, $a : x = v_i t - v_i t_i$, $b : x = -v_i t + L + v_i t_i$
- Απ' αυτές κρατάμε το convex hull σε αύξουσα σειρά κλίσης. Το πρόβλημα πλέον είναι, δεδομένων 2 συνόλων ευθειών, να βρούμε τις 2 που δε συμμετέχουν στο ίδιο σύνολο και έχουν τη μικρότερη τετμημένη τομής
- Το βρίσκουμε σε γραμμικό χρόνο με 2 δείκτες. Ξεκινάμε τους δείκτες στην πρώτη ευθεία του κάθε συνόλου, αυξάνουμε τον πρώτο όσο το σημείο τομής μετατοπίζεται αριστερά. Αν το σημείο τομής για την επόμενη θέση του πρώτου δείκτη έχει μεγαλύτερη τετμημένη από την τωρινή, προχωράμε τον άλλο δείκτη. Ο αλγόριθμος είναι γραμμικός και τρέχει K φορές, άρα θέλουμε χρόνο $O(NK)$