Αλγόριθμοι και Πολυπλοκότητα

1η σειρά γραπτών ασκήσεων

1η σειρά προγραμματιστικών ασκήσεων

Σχέδιο Λύσεων

CoReLab

ΣΗΜΜΥ Ε.Μ.Π.

17 Νοεμβρίου 2019

Άσκηση 1 (α) - Ασυμπτωτικός Συμβολισμός

Η τελική ταξινόμηση έχει ως εξής:

$$\sum_{k=1}^{n} k2^{-k} < \frac{\log(n!)}{(\log n)^3} < \log\binom{2n}{n} \approx n2^{2^{2^{100}}} < 2^{(\log_2 n)^4} <$$

$$< (\sqrt{n})! < n\sum_{k=0}^{n}\binom{n}{k} \approx \sum_{k=1}^{n} k2^k$$

Άσκηση 1 (α) - Ασυμπτωτικός Συμβολισμός

Υπολογισμοί

- *Για τη μικρότερη συνάρτηση ισχύει:*

$$\sum_{k=1}^{n} k2^{-k} < \sum_{k=1}^{\infty} k2^{-k}$$

Αν υπολογίσουμε το δεύτερο άθροισμα θα πάρουμε:

$$\sum_{k=1}^{\infty} k2^{-k} = 2 \text{ Επομένως: } \sum_{k=1}^{n} k2^{-k} \in \Theta(1)$$

- *Για την επόμενη συνάρτηση*

$$\frac{\log(n!)}{(\log n)^3} \text{ ισχύει πως: } n! \leq n^n$$

$$\log(n!) \leq n \log n \Rightarrow \frac{\log(n!)}{(\log n)^3} \leq \frac{n}{(\log n)^2} = o(n)$$

Άσκηση 1 (α) - Ασυμπτωτικός Συμβολισμός

Υπολογισμοί

- *Για τη μικρότερη συνάρτηση ισχύει:*

$$\sum_{k=1}^{n} k2^{-k} < \sum_{k=1}^{\infty} k2^{-k}$$

Αν υπολογίσουμε το δεύτερο άθροισμα θα πάρουμε:

$$\sum_{k=1}^{\infty} k2^{-k} = 2 \text{ Επομένως: } \sum_{k=1}^{n} k2^{-k} \in \Theta(1)$$

- *Για την επόμενη συνάρτηση*

$$\frac{\log(n!)}{(\log n)^3} \text{ ισχύει πως: } n! \leq n^n$$

$$\log(n!) \leq n \log n \Rightarrow \frac{\log(n!)}{(\log n)^3} \leq \frac{n}{(\log n)^2} = o(n)$$

Άσκηση 1 (α) - Ασυμπτωτικός Συμβολισμός

Υπολογισμοί συνέχεια

- *Για την επόμενη συνάρτηση ισχύει η γνωστή ανισότητα:*

$$(\frac{n}{k})^k \leq \binom{n}{k} \leq \frac{n^k}{k!} \leq (\frac{n \cdot e}{k})^k$$

$$(\pi.\chi.\ n \log 2 \leq \log \binom{2n}{n} \leq n \log 2e)$$

Επομένως:

$$\log \binom{2n}{n} \in \Theta(n)$$

- *Για την επόμενη συνάρτηση*

$$n2^{2^{2^{100}}} = cn \in \Theta(n)$$

- *Επόμενη συνάρτηση η:*

$2^{(\log_2 n)^4}$ *για την οποία ισχύει πως:* $2^{(\log_2 n)^4} = \Theta(n^{(\log n)^3})$

Άσκηση 1 (α) - Ασυμπτωτικός Συμβολισμός

Υπολογισμοί συνέχεια

- *Για την επόμενη συνάρτηση ισχύει η γνωστή ανισότητα:*

$$(\frac{n}{k})^k \leq \binom{n}{k} \leq \frac{n^k}{k!} \leq (\frac{n \cdot e}{k})^k$$

$$(\pi.\chi.\ n \log 2 \leq \log \binom{2n}{n} \leq n \log 2e)$$

Επομένως:

$$\log \binom{2n}{n} \in \Theta(n)$$

- *Για την επόμενη συνάρτηση*

$$n2^{2^{2^{100}}} = cn \in \Theta(n)$$

- *Επόμενη συνάρτηση η:*

$2^{(\log_2 n)^4}$ *για την οποία ισχύει πως:* $2^{(\log_2 n)^4} = \Theta(n^{(\log n)^3})$

Άσκηση 1 (α) - Ασυμπτωτικός Συμβολισμός

Υπολογισμοί συνέχεια

- *Για την επόμενη συνάρτηση ισχύει η γνωστή ανισότητα:*

$$(\frac{n}{k})^k \leq \binom{n}{k} \leq \frac{n^k}{k!} \leq (\frac{n \cdot e}{k})^k$$

$$(\pi.\chi.\ n \log 2 \leq \log \binom{2n}{n} \leq n \log 2e)$$

Επομένως:

$$\log \binom{2n}{n} \in \Theta(n)$$

- *Για την επόμενη συνάρτηση*

$$n2^{2^{2^{100}}} = cn \in \Theta(n)$$

- *Επόμενη συνάρτηση η:*

$2^{(\log_2 n)^4}$ *για την οποία ισχύει πως:* $2^{(\log_2 n)^4} = \Theta(n^{(\log n)^3})$

Άσκηση 1 (α) - Ασυμπτωτικός Συμβολισμός

Υπολογισμοί συνέχεια

- *Ακολουθεί η εκθετική συνάρτηση:*

$$(\sqrt{n})! \leq (\sqrt{n})^{\sqrt{n}} \leq 2^{\sqrt{n}\log n} \leq 2^{\sqrt{n}\sqrt{n}} \leq 2^n$$

- *Επόμενη συνάρτηση:*

$$n\sum_{k=0}^{n}\binom{n}{k} = n2^n \in \Theta(n2^n)$$

- *Και τελευταία η συνάρτηση:*

$$\sum_{k=1}^{n} k2^k \in \Theta(n2^n)$$

Άσκηση 1 (α) - Ασυμπτωτικός Συμβολισμός

Υπολογισμοί συνέχεια

- *Ακολουθεί η εκθετική συνάρτηση:*

$$(\sqrt{n})! \leq (\sqrt{n})^{\sqrt{n}} \leq 2^{\sqrt{n}\log n} \leq 2^{\sqrt{n}\sqrt{n}} \leq 2^n$$

- *Επόμενη συνάρτηση:*

$$n\sum_{k=0}^{n}\binom{n}{k} = n2^n \in \Theta(n2^n)$$

- *Και τελευταία η συνάρτηση:*

$$\sum_{k=1}^{n} k2^k \in \Theta(n2^n)$$

Υπολογισμοί συνέχεια

- *Ακολουθεί η εκθετική συνάρτηση:*

$$(\sqrt{n})! \leq (\sqrt{n})^{\sqrt{n}} \leq 2^{\sqrt{n}\log n} \leq 2^{\sqrt{n}\sqrt{n}} \leq 2^n$$

- *Επόμενη συνάρτηση:*

$$n\sum_{k=0}^{n}\binom{n}{k} = n2^n \in \Theta(n2^n)$$

- *Και τελευταία η συνάρτηση:*

$$\sum_{k=1}^{n} k2^k \in \Theta(n2^n)$$

Άσκηση 1 (β) - Αναδρομικές Σχέσεις

- *Για την πρώτη αναδρομική*

$$T(n) = 3T(n/2)+n^2 \log n = aT(n/b)+n^2 \log n \text{ όπου } a = 3, b = 2$$

$$\text{από Master Theorem } T(n) = \Theta(n^2 \log n)$$

- *Για τη δεύτερη αναδρομική*

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

από το δέντρο αναδρομής για το κ-οστό επίπεδο έχουμε

$$n^2 \log n - n^2 \log(2^k)$$

Αθροίζοντας τα επίπεδα παίρνουμε

$$\Theta(n^2 \log^2 n)$$

- *Για την τρίτη ανδρομική*

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

$$n^2 \log n < n^{\log_2 5 - 0.22} \approx n^{2.1} \text{ Master Theorem } T(n) = \Theta(n^{log_2 5})$$

Άσκηση 1 (β) - Αναδρομικές Σχέσεις

- *Για την πρώτη αναδρομική*

  $T(n) = 3T(n/2)+n^2 \log n = aT(n/b)+n^2 \log n$ *όπου* $a = 3, b = 2$

  $$\text{από Master Theorem } T(n) = \Theta(n^2 \log n)$$

- *Για τη δεύτερη αναδρομική*

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

  από το δέντρο αναδρομής για το κ-οστό επίπεδο έχουμε

  $$n^2 \log n - n^2 \log(2^k)$$

  Αθροίζοντας τα επίπεδα παίρνουμε

  $$\Theta(n^2 \log^2 n)$$

- *Για την τρίτη ανδρομική*

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

  $n^2 \log n < n^{\log_2 5 - 0.22} \approx n^{2.1}$ *Master Theorem* $T(n) = \Theta(n^{log_2 5})$

Άσκηση 1 (β) - Αναδρομικές Σχέσεις

- *Για την πρώτη αναδρομική*

  $T(n) = 3T(n/2)+n^2 \log n = aT(n/b)+n^2 \log n$ *όπου* $a = 3, b = 2$

  $$\text{από Master Theorem } T(n) = \Theta(n^2 \log n)$$

- *Για τη δεύτερη αναδρομική*

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

  από το δέντρο αναδρομής για το κ-οστό επίπεδο έχουμε

  $$n^2 \log n - n^2 \log(2^k)$$

  Αθροίζοντας τα επίπεδα παίρνουμε

  $$\Theta(n^2 \log^2 n)$$

- *Για την τρίτη ανδρομική*

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

  $n^2 \log n < n^{\log_2 5 - 0.22} \approx n^{2.1}$ *Master Theorem* $T(n) = \Theta(n^{log_2 5})$

Άσκηση 1 (β) - Αναδρομικές Σχέσεις

- *Για την τέταρτη αναδρομική*

$$T(n) = T(n/2) + T(n/3) + n,\ 1/2 + 1/3 = 5/6 < 1$$

$$\epsilon\pi o\mu\acute{\epsilon}\nu\omega\varsigma\ T(n) = \Theta(n)$$

- *Για την πέμπτη ανδρομική*

$T(n) = T(n/2)+T(n/3)+T(n/6)+n$ *επειδή* $n/2+n/3+n/6 = n$

χρησιμοποιούμε δέντρο αναδρομής για να αποδείξουμε πως

$T(n) = \Theta(n \log n)$(*το ύψος φράσσεται από* $\log_6 n$ *και* $\log_2 n$)

- *Για την έκτη ανδρομική*

$T(n) = T(n/4)+\sqrt{n}$ *για* $n = m^2 T(n) = T(m^2) = T((m/2)^2)+m$

Έστω $S(m) = T(m^2) = T((m/2)^2)+m$ *αλλά* $S(m/2) = T((m/2)^2)$

άρα $S(m) = S(m/2)+m$ *από δέντρο αναδρομής* $S(m) = \Theta(m)$

$$Τελικά\ T(n) = T(m^2) = S(m) = \Theta(m) = \Theta(\sqrt{n})$$

Άσκηση 1 (β) - Αναδρομικές Σχέσεις

- *Για την τέταρτη αναδρομική*

$$T(n) = T(n/2) + T(n/3) + n,\ 1/2 + 1/3 = 5/6 < 1$$

$$\textit{επομένως } T(n) = \Theta(n)$$

- *Για την πέμπτη ανδρομική*

$T(n) = T(n/2)+T(n/3)+T(n/6)+n$ *επειδή* $n/2+n/3+n/6 = n$

χρησιμοποιούμε δέντρο αναδρομής για να αποδείξουμε πως

$T(n) = \Theta(n \log n)$(*το ύψος φράσσεται από* $\log_6 n$ *και* $\log_2 n$)

- *Για την έκτη ανδρομική*

$T(n) = T(n/4)+\sqrt{n}$ *για* $n = m^2 T(n) = T(m^2) = T((m/2)^2)+m$

Έστω $S(m) = T(m^2) = T((m/2)^2)+m$ *αλλά* $S(m/2) = T((m/2)^2)$

άρα $S(m) = S(m/2)+m$ *από δέντρο αναδρομής* $S(m) = \Theta(m)$

Τελικά $T(n) = T(m^2) = S(m) = \Theta(m) = \Theta(\sqrt{n})$

Άσκηση 1 (β) - Αναδρομικές Σχέσεις

- *Για την τέταρτη αναδρομική*

$$T(n) = T(n/2) + T(n/3) + n,\ 1/2 + 1/3 = 5/6 < 1$$

$$\textit{επομένως } T(n) = \Theta(n)$$

- *Για την πέμπτη ανδρομική*

$T(n) = T(n/2)+T(n/3)+T(n/6)+n$ *επειδή* $n/2+n/3+n/6 = n$

χρησιμοποιούμε δέντρο αναδρομής για να αποδείξουμε πως

$T(n) = \Theta(n \log n)$(*το ύψος φράσσεται από* $\log_6 n$ *και* $\log_2 n$)

- *Για την έκτη ανδρομική*

$T(n) = T(n/4)+\sqrt{n}$ *για* $n = m^2 T(n) = T(m^2) = T((m/2)^2)+m$

Έστω $S(m) = T(m^2) = T((m/2)^2)+m$ *αλλά* $S(m/2) = T((m/2)^2)$

άρα $S(m) = S(m/2)+m$ *από δέντρο αναδρομής* $S(m) = \Theta(m)$

$$\textit{Τελικά } T(n) = T(m^2) = S(m) = \Theta(m) = \Theta(\sqrt{n})$$

Άσκηση 2(α) Εύρεση Ελαχίστου Διαστήματος

- **Είσοδος:** Ταξινομημένοι πίνακες $A_1[1...n_1]$, $A_2[1...n_2]$
- **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 $t_1 = 1$, $t_2 = 1$
- Προχωρούμε 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
- Επιστρέφουμε i_1, i_2 που αντιστοιχούν στο ελάχιστο διάστημα

Άσκηση 2(α) Εύρεση Ελαχίστου Διαστήματος

- **Είσοδος:** Ταξινομημένοι πίνακες $A_1[1...n_1]$, $A_2[1...n_2]$
- **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 $t_1 = 1$, $t_2 = 1$
- Προχωρούμε 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
- Επιστρέφουμε i_1, i_2 που αντιστοιχούν στο ελάχιστο διάστημα

Άσκηση 2(α) Εύρεση Ελαχίστου Διαστήματος

- **Είσοδος:** Ταξινομημένοι πίνακες $A_1[1...n_1]$, $A_2[1...n_2]$
- **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 $t_1 = 1$, $t_2 = 1$
- Προχωρούμε 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
- Επιστρέφουμε i_1, i_2 που αντιστοιχούν στο ελάχιστο διάστημα

Άσκηση 2(α) Εύρεση Ελαχίστου Διαστήματος

- **Είσοδος:** Ταξινομημένοι πίνακες $A_1[1...n_1]$, $A_2[1...n_2]$
- **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 $t_1 = 1$, $t_2 = 1$
- Προχωρούμε 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
- Επιστρέφουμε i_1, i_2 που αντιστοιχούν στο ελάχιστο διάστημα

Άσκηση 2(α) Εύρεση Ελαχίστου Διαστήματος

- **Είσοδος:** Ταξινομημένοι πίνακες $A_1[1...n_1]$, $A_2[1...n_2]$
- **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 $t_1 = 1$, $t_2 = 1$
- Προχωρούμε 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
- Επιστρέφουμε i_1, i_2 που αντιστοιχούν στο ελάχιστο διάστημα

Άσκηση 2(α) Εύρεση Ελαχίστου Διαστήματος

- **Είσοδος:** Ταξινομημένοι πίνακες $A_1[1...n_1]$, $A_2[1...n_2]$
- **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 $t_1 = 1$, $t_2 = 1$
- Προχωρούμε 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
- Επιστρέφουμε i_1, i_2 που αντιστοιχούν στο ελάχιστο διάστημα

Άσκηση 2(α) Εύρεση Ελαχίστου Διαστήματος

- **Είσοδος:** Ταξινομημένοι πίνακες $A_1[1...n_1]$, $A_2[1...n_2]$
- **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 $t_1 = 1$, $t_2 = 1$
- Προχωρούμε 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
- Επιστρέφουμε i_1, i_2 που αντιστοιχούν στο ελάχιστο διάστημα

Άσκηση 2(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 στοιχείων δεν έχει νόημα να μειώσουμε το δείκτη του μεγαλύτερου στοιχείου
- Επομένως για να πετύχουμε ενδεχόμενη μείωση του διαστήματος αυξάνουμε τον δείκτη του μικρότερου στοιχείου

Άσκηση 2(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 στοιχείων δεν έχει νόημα να μειώσουμε το δείκτη του μεγαλύτερου στοιχείου
- Επομένως για να πετύχουμε ενδεχόμενη μείωση του διαστήματος αυξάνουμε τον δείκτη του μικρότερου στοιχείου

Άσκηση 2(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 στοιχείων δεν έχει νόημα να μειώσουμε το δείκτη του μεγαλύτερου στοιχείου
- Επομένως για να πετύχουμε ενδεχόμενη μείωση του διαστήματος αυξάνουμε τον δείκτη του μικρότερου στοιχείου

Άσκηση 2(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 δειξουμε ότι ο αλγόριθμός μας περνά 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). $x < i_1 \Rightarrow A_i[x] < A_i[1] \leq A_2[i_2]$ 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 2(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 δειξουμε ότι ο αλγόριθμός μας περνά 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). $x < i_1 \Rightarrow A_i[x] < A_i[1] \leq A_2[i_2]$ 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 2(β) Ελαχιστοποίηση Ποσότητας

Είσοδος: Ταξινομημένοι πίνακες $A_1[1...n_1]$, $A_2[1...n_2]$, ... , $A_m[1...n_m]$

Έξοδος: Θέσεις i_1, i_2, i_m ώστε $max A_1[i_1], ..., A_m[i_m] - min A_1[i_1], ..., A_m[i_m]$ 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 στο προηγούμενο ερώτημα.

Άσκηση 2(β) Ελαχιστοποίηση Ποσότητας

Είσοδος: Ταξινομημένοι πίνακες $A_1[1...n_1]$, $A_2[1...n_2]$, ... , $A_m[1...n_m]$

Έξοδος: Θέσεις i_1, i_2, i_m ώστε $max A_1[i_1], ..., A_m[i_m] - min A_1[i_1], ..., A_m[i_m]$ 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 στο προηγούμενο ερώτημα.

Άσκηση 2(β) Ελαχιστοποίηση Ποσότητας

- Δημιουργούμε ένα πίνακα με m θέσεις, κάθε θέση i την αρχικοποιούμε με το πρώτο στοιχείο του A_i
- Προχωρούμε 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 $i_1, ..., i_m$ που αντιστοιχούν στο ελάχιστο διάστημα

Άσκηση 2(β) Ελαχιστοποίηση Ποσότητας

- Δημιουργούμε ένα πίνακα με m θέσεις, κάθε θέση i την αρχικοποιούμε με το πρώτο στοιχείο του A_i
- Προχωρούμε 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 $i_1, ..., i_m$ που αντιστοιχούν στο ελάχιστο διάστημα

Άσκηση 2(β) Ελαχιστοποίηση Ποσότητας

- Δημιουργούμε ένα πίνακα με m θέσεις, κάθε θέση i την αρχικοποιούμε με το πρώτο στοιχείο του A_i
- Προχωρούμε 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 $i_1, ..., i_m$ που αντιστοιχούν στο ελάχιστο διάστημα

Άσκηση 2(β) Ελαχιστοποίηση Ποσότητας

- Δημιουργούμε ένα πίνακα με m θέσεις, κάθε θέση i την αρχικοποιούμε με το πρώτο στοιχείο του A_i
- Προχωρούμε 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 $i_1, ..., i_m$ που αντιστοιχούν στο ελάχιστο διάστημα

Άσκηση 2(β) Ελαχιστοποίηση Ποσότητας

- Δημιουργούμε ένα πίνακα με m θέσεις, κάθε θέση i την αρχικοποιούμε με το πρώτο στοιχείο του A_i
- Προχωρούμε 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 $i_1, ..., i_m$που αντιστοιχούν στο ελάχιστο διάστημα

Άσκηση 2(β) Ελαχιστοποίηση Ποσότητας

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 2(β) Ελαχιστοποίηση Ποσότητας

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

- Αν $t_1 \leq i_1 \Rightarrow t_j \leq i_j, \forall j \geq 2$, όπου t_j ο δείκτης στο τρέχον στοιχείο του 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 φορά θα φτάσει να είναι το min). Επίσης $\forall t_j, j \geq 2$, ισχύει $A_j[t_j] \leq A_j[i_j] \leq A_m[i_m] \Rightarrow max_j A_j[t_j] \leq A_m[i_m]$
- 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 2(β) Ελαχιστοποίηση Ποσότητας

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

- Αν $t_1 \leq i_1 \Rightarrow t_j \leq i_j, \forall j \geq 2$, όπου t_j ο δείκτης στο τρέχον στοιχείο του 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 φορά θα φτάσει να είναι το min). Επίσης $\forall t_j, j \geq 2$, ισχύει $A_j[t_j] \leq A_j[i_j] \leq A_m[i_m] \Rightarrow max_j A_j[t_j] \leq A_m[i_m]$
- 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 2(β) Ελαχιστοποίηση Ποσότητας

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

- Αν $t_1 \leq i_1 \Rightarrow t_j \leq i_j, \forall j \geq 2$, όπου t_j ο δείκτης στο τρέχον στοιχείο του 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 φορά θα φτάσει να είναι το min). Επίσης $\forall t_j, j \geq 2$, ισχύει $A_j[t_j] \leq A_j[i_j] \leq A_m[i_m] \Rightarrow max_j A_j[t_j] \leq A_m[i_m]$
- 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 2(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 στειχείου 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) δεν χρειάζεται σωρός, μπορούμε να υπολογίσουμε το αρχικό max σε χρόνο $O(m)$ και σε κάθε επανάληψη συγκρίνουμε το νέο στοιχείο με το παλαιό max.
- Επομένως χρειαζόμαστε συνολικό χρόνο $O(N \log m)$

Άσκηση 2(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 στειχείου 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) δεν χρειάζεται σωρός, μπορούμε να υπολογίσουμε το αρχικό max σε χρόνο $O(m)$ και σε κάθε επανάληψη συγκρίνουμε το νέο στοιχείο με το παλαιό max.
- Επομένως χρειαζόμαστε συνολικό χρόνο $O(N \log m)$

Άσκηση 2(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 στειχείου 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) δεν χρειάζεται σωρός, μπορούμε να υπολογίσουμε το αρχικό max σε χρόνο $O(m)$ και σε κάθε επανάληψη συγκρίνουμε το νέο στοιχείο με το παλαιό max.
- Επομένως χρειαζόμαστε συνολικό χρόνο $O(N \log m)$

Άσκηση 2(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 στειχείου 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) δεν χρειάζεται σωρός, μπορούμε να υπολογίσουμε το αρχικό max σε χρόνο $O(m)$ και σε κάθε επανάληψη συγκρίνουμε το νέο στοιχείο με το παλαιό max.
- Επομένως χρειαζόμαστε συνολικό χρόνο $O(N \log m)$

Άσκηση 3 - Παίζοντας Χαρτιά

Είσοδος: 6 3 5 2 4 1 (κορυφή τράπουλας 6)

ROUND 1: Το 6 εκκινεί το παιχνίδι και δημιουργεί μία στοίβα.
ROUND 2: Αφού $3 < 6$, το 3 μπαίνει πάνω από το 6.

6
3

ROUND 3: Τώρα $5 > 3$, άρα αναγκαστικά δημιουργούμε μία νέα στοίβα.

6 5
3

ROUND 4: Τραβάμε 2. Παρατηρείστε ότι η εξέλιξη του παιχνιδιού δεν είναι μοναδική :

6 5 3 2	ή	6 5 3 2

...

Input : $d = c_1 c_2 ... c_N$
(GS) GreedyStrategy $(d = c_1 c_2 ... c_N)$:

Initialization : Δημιούργησε μία στοίβα με την κάρτα c_1.

Step : Τοποθετούμε την k-οστή κάρτα c_k πάνω στην μικρότερης αξίας κάρτα που είναι μεγαλύτερη από την c_k. Διαφορετικά (αν δεν υπάρχει τέτοια), ανοίγουμε νέα στοίβα και τοποθετούμε την c_k την κορυφή της.

Ερώτημα 1

- Έστω ότι η GS δεν είναι η βέλτιστη στρατηγική. Έστω OPT ο βέλτιστος αλγόριθμος που δημιουργεί λιγότερες στοίβες.
- Ας υποθέσουμε, ότι σε κάποιον γύρο, οι δύο στρατηγικές διαφοροποιούνται. Έστω ότι η άπληστη στρατηγική τοποθετεί το τρέχον φύλλο c στην στοίβα p, ενώ ο 'βέλτιστος' αλγόριθμος την τοποθετεί στην στοίβα q. (c_p, c_q top cards).

$$c < c_p < c_q$$

- Μπορείτε να σχεδιάσετε στρατηγική OPT' που να αποδίδει τουλάχιστον όσο καλά αποδίδει η OPT?
- Strategy Stealing Argument
- Time Complexity : $O(NlgN)$.
- *Παρατήρηση :* Σε κάθε βήμα του αλγορίθμου GS, η ακολουθία των top cards $\{c_{i,|p_i|}\}_{i=1}^{m}$ είναι άυξουσα από αριστερά προς τα δεξιά και κάθε στοίβα είναι προφανώς φθίνουσα.

Ερώτημα 1

- Έστω ότι η GS δεν είναι η βέλτιστη στρατηγική. Έστω OPT ο βέλτιστος αλγόριθμος που δημιουργεί λιγότερες στοίβες.
- Ας υποθέσουμε, ότι σε κάποιον γύρο, οι δύο στρατηγικές διαφοροποιούνται. Έστω ότι η άπληστη στρατηγική τοποθετεί το τρέχον φύλλο c στην στοίβα p, ενώ ο 'βέλτιστος' αλγόριθμος την τοποθετεί στην στοίβα q. (c_p, c_q top cards).

$$c < c_p < c_q$$

- Μπορείτε να σχεδιάσετε στρατηγική OPT' που να αποδίδει τουλάχιστον όσο καλά αποδίδει η OPT?
- Strategy Stealing Argument
- Time Complexity : $O(NlgN)$.
- *Παρατήρηση :* Σε κάθε βήμα του αλγορίθμου GS, η ακολουθία των top cards $\{c_{i,|p_i|}\}_{i=1}^{m}$ είναι άυξουσα από αριστερά προς τα δεξιά και κάθε στοίβα είναι προφανώς φθίνουσα.

Ερώτημα 1

- Έστω ότι η GS δεν είναι η βέλτιστη στρατηγική. Έστω OPT ο βέλτιστος αλγόριθμος που δημιουργεί λιγότερες στοίβες.
- Ας υποθέσουμε, ότι σε κάποιον γύρο, οι δύο στρατηγικές διαφοροποιούνται. Έστω ότι η άπληστη στρατηγική τοποθετεί το τρέχον φύλλο c στην στοίβα p, ενώ ο 'βέλτιστος' αλγόριθμος την τοποθετεί στην στοίβα q. (c_p, c_q top cards).

$$c < c_p < c_q$$

- Μπορείτε να σχεδιάσετε στρατηγική OPT' που να αποδίδει τουλάχιστον όσο καλά αποδίδει η OPT?
- Strategy Stealing Argument
- Time Complexity : $O(NlgN)$.
- *Παρατήρηση :* Σε κάθε βήμα του αλγορίθμου GS, η ακολουθία των top cards $\{c_{i,|p_i|}\}_{i=1}^{m}$ είναι άυξουσα από αριστερά προς τα δεξιά και κάθε στοίβα είναι προφανώς φθίνουσα.

Ερώτημα 2

Ιδέα Α : $k-$way merge - Time : $O(Nlgk)$. Κάνε MERGE (η οποία αντιστοιχεί στην 2-way merge) σε ζεύγη πινάκων και άρα, σε κάθε επανάληψη, το πλήθος των πινάκων πέφτει στο μισό.

Ερώτημα 2

Ιδέα Β : Direct k-way merge - Time : $\Theta(Nlgk)$.
Βήματα :

- Έχοντας k στοίβες, δημιουργούμε ένα δέντρο σε $\Theta(k)$.
- Το δέντρο είναι balanced, άρα από τα φύλλα στην ρίζα $\Theta(lgk)$.
- Συνολικά, μεταφέρουμε N στοιχεία (φύλλα) άρα $\Theta(Nlgk)$.

Ερώτημα 2

Παράδειγμα :
Είσοδος : 2 1 10 5 3 4 9 8 7 6.
$p_1 = \{1, 2\}$, $p_2 = \{3, 5, 10\}$, $p_3 = \{4\}$, $p_4 = \{6, 7, 8, 9\}$
Φτιάχνω balanced δέντρο με $k = 4$ φύλλα. Στο φύλλο $leaf_i$, τοποθετώ το $\min p_i$ και το αφαιρώ από την αντίστοιχη λίστα. Τα φύλλα παίζουν ένα τουρνουά και ανεβάζω στην ρίζα το ελάχιστο όλων. Αν το ελάχιστο προήλθε από την λίστα p_j, στην θέση του $leaf_j$ θέτω το νέο ελάχιστο της λίστας j.

Ερώτημα 3

Παράδειγμα :

3	4	7	8
2		5	6
1			

$\triangleright$ Στοίβες $= 4$

$\triangleright$ $len(LIS) = 4 = \text{length}(\ 2 \to 4 \to 5 \to 6) =$
$\text{length}(3 \to 4 \to 5 \to 6) = ...$

Αν παίξουμε σύμφωνα με την GS, τότε $\#\text{piles} = len(LIS)$.

Ερώτημα 3

Παράδειγμα :

3	4	7	8
2		5	6
1			

▷ Στοίβες $= 4$

▷ $len(LIS) = 4 = \text{length}(\ 2 \to 4 \to 5 \to 6) =$
$\text{length}(3 \to 4 \to 5 \to 6) = ...$

Αν παίξουμε σύμφωνα με την GS, τότε $\#\text{piles} = len(LIS)$.

Ερώτημα 4

Αν $c_{i_1}, c_{i_2}, ..., c_{i_L}$ μία μέγιστη αύξουσα υπακολουθία της τράπουλας $c_1, ..., c_N$, τότε σύμφωνα με τους κανόνες του παιχνιδιού και έχοντας τοποθετήσει τις κάρτες $c_{i_1}, c_{i_2}, .., c_{i_k}$, η τοποθέτηση της κάρτας c_{i_k} μας επιβάλλει να ανοίξουμε νέα στοίβα.

Ερώτημα 5

▷ Σε κάθε στάδιο του άπληστου αλγορίθμου, οι top cards αποτελούν μία αύξουσα ακολουθία από αριστερά προς τα δεξιά.

▷ Ισχύει :$\#piles \geq$ length of any increasing subsequence
Γιατί; Σε κάθε στοίβα, οι κάρτες σχηματίζουν μία φθίνουσα ακολουθία. Κάθε πιθανή αύξουσα ακολουθία χρησιμοποιεί το πολύ ένα φύλλο από κάθε στοίβα. Η ιδιότητα αυτή ονομάζεται weak duality.

▷ $min\{\#piles\} = max\{$length of any increasing subsequence$\}$ και η GS τα υπολογίζει. Γιατί; Κάθε φορά που εισάγεται μία κάρτα σε κάποια στοίβα με $id = p$, τοποθετούμε έναν δείκτη προς την κάρτα που βρίσκεται στην κορυφή της στοίβας με $id = p - 1$ (στην αριστερή της). Ακολουθώντας τα μονοπάτια από αριστερά προς τα δεξιά, σχηματίζουμε αύξουσες ακολουθίες. Αν αρχίσουμε από την δεξιότερη στοίβα, θα πάρουμε ακολουθία με μήκος όσο το πλήθος των στοιβών. Από weak duality,οι δύο ποσότητες είναι optimal. (strong duality).

Ερώτημα 5

▷ Σε κάθε στάδιο του άπληστου αλγορίθμου, οι top cards αποτελούν μία αύξουσα ακολουθία από αριστερά προς τα δεξιά.
▷ Ισχύει :$\#piles \geq$ length of any increasing subsequence
Γιατί; Σε κάθε στοίβα, οι κάρτες σχηματίζουν μία φθίνουσα ακολουθία. Κάθε πιθανή αύξουσα ακολουθία χρησιμοποιεί το πολύ ένα φύλλο από κάθε στοίβα. Η ιδιότητα αυτή ονομάζεται weak duality.
▷ $min\{\#piles\} = max\{$length of any increasing subsequence$\}$ και η GS τα υπολογίζει. Γιατί; Κάθε φορά που εισάγεται μία κάρτα σε κάποια στοίβα με $id = p$, τοποθετούμε έναν δείκτη προς την κάρτα που βρίσκεται στην κορυφή της στοίβας με $id = p - 1$ (στην αριστερή της). Ακολουθώντας τα μονοπάτια από αριστερά προς τα δεξιά, σχηματίζουμε αύξουσες ακολουθίες. Αν αρχίσουμε από την δεξιότερη στοίβα, θα πάρουμε ακολουθία με μήκος όσο το πλήθος των στοιβών. Από weak duality,οι δύο ποσότητες είναι optimal. (strong duality).

Ερώτημα 5

▷ Σε κάθε στάδιο του άπληστου αλγορίθμου, οι top cards αποτελούν μία αύξουσα ακολουθία από αριστερά προς τα δεξιά.
▷ Ισχύει :$\#piles \geq$ length of any increasing subsequence
Γιατί; Σε κάθε στοίβα, οι κάρτες σχηματίζουν μία φθίνουσα ακολουθία. Κάθε πιθανή αύξουσα ακολουθία χρησιμοποιεί το πολύ ένα φύλλο από κάθε στοίβα. Η ιδιότητα αυτή ονομάζεται weak duality.
▷ $min\{\#piles\} = max\{$length of any increasing subsequence$\}$ και η GS τα υπολογίζει. Γιατί; Κάθε φορά που εισάγεται μία κάρτα σε κάποια στοίβα με $id = p$, τοποθετούμε έναν δείκτη προς την κάρτα που βρίσκεται στην κορυφή της στοίβας με $id = p - 1$ (στην αριστερή της). Ακολουθώντας τα μονοπάτια από αριστερά προς τα δεξιά, σχηματίζουμε αύξουσες ακολουθίες. Αν αρχίσουμε από την δεξιότερη στοίβα, θα πάρουμε ακολουθία με μήκος όσο το πλήθος των στοιβών. Από weak duality,οι δύο ποσότητες είναι optimal. (strong duality).

Ερώτημα 5

▷ Σε κάθε στάδιο του άπληστου αλγορίθμου, οι top cards αποτελούν μία αύξουσα ακολουθία από αριστερά προς τα δεξιά.
▷ Ισχύει :$\#piles \geq$ length of any increasing subsequence
Γιατί; Σε κάθε στοίβα, οι κάρτες σχηματίζουν μία φθίνουσα ακολουθία. Κάθε πιθανή αύξουσα ακολουθία χρησιμοποιεί το πολύ ένα φύλλο από κάθε στοίβα. Η ιδιότητα αυτή ονομάζεται weak duality.
▷ $min\{\#piles\} = max\{$length of any increasing subsequence$\}$ και η GS τα υπολογίζει. Γιατί; Κάθε φορά που εισάγεται μία κάρτα σε κάποια στοίβα με $id = p$, τοποθετούμε έναν δείκτη προς την κάρτα που βρίσκεται στην κορυφή της στοίβας με $id = p - 1$ (στην αριστερή της). Ακολουθώντας τα μονοπάτια από αριστερά προς τα δεξιά, σχηματίζουμε αύξουσες ακολουθίες. Αν αρχίσουμε από την δεξιότερη στοίβα, θα πάρουμε ακολουθία με μήκος όσο το πλήθος των στοιβών. Από weak duality,οι δύο ποσότητες είναι optimal. (strong duality).

Ερώτημα 5

▷ Σε κάθε στάδιο του άπληστου αλγορίθμου, οι top cards αποτελούν μία αύξουσα ακολουθία από αριστερά προς τα δεξιά.
▷ Ισχύει :$\#piles \geq$ length of any increasing subsequence
Γιατί; Σε κάθε στοίβα, οι κάρτες σχηματίζουν μία φθίνουσα ακολουθία. Κάθε πιθανή αύξουσα ακολουθία χρησιμοποιεί το πολύ ένα φύλλο από κάθε στοίβα. Η ιδιότητα αυτή ονομάζεται weak duality.
▷ $min\{\#piles\} = max\{$length of any increasing subsequence$\}$ και η GS τα υπολογίζει. Γιατί; Κάθε φορά που εισάγεται μία κάρτα σε κάποια στοίβα με $id = p$, τοποθετούμε έναν δείκτη προς την κάρτα που βρίσκεται στην κορυφή της στοίβας με $id = p - 1$ (στην αριστερή της). Ακολουθώντας τα μονοπάτια από αριστερά προς τα δεξιά, σχηματίζουμε αύξουσες ακολουθίες. Αν αρχίσουμε από την δεξιότερη στοίβα, θα πάρουμε ακολουθία με μήκος όσο το πλήθος των στοιβών. Από weak duality,οι δύο ποσότητες είναι optimal. (strong duality).

Ερώτημα 6

▷ $\{1, 2, ..., mn + 1\}$. Εκτελούμε την GS για την τράπουλα με $mn + 1$ φύλλα. Το πλήθος των στοιβών είναι όσο το $len(LIS)$ και η κάθε στοίβα αντιστοιχεί σε μία φθίνουσα ακολουθία.
▷ Erdős–Szekeres Theorem.
▷ ([5,4,3,2,1], [10,9,8,7,6], ...)

Άσκηση 4- Γρήγορη Επιλογή στο Πεδίο της Μάχης

- **Ερώτημα 1:** Μπορούμε να βρούμε με ακρίβεια τη θέση οποιουδήποτε σωματιδίου οποιαδήποτε χρονική στιγμή σε $\mathcal{O}(1)$.
- **Ερώτημα 2:** Αν μας δοθούν 2 σωματίδια διαφορετικού τύπου, μπορούμε με ακρίβεια να βρούμε πότε θα συγκρούονταν (αν δεν γίνονταν οι εξαϋλώσεις) σε $\mathcal{O}(1)$.

Ερώτημα 1

- Παρατηρούμε πως η πρώτη σύγκρουση θα πρέπει να γίνει κάποια στιγμή στο διάστημα $\left[\frac{L}{2V_{max}}, \frac{L}{2V_{min}}\right]$.
- Θεωρώντας κάποια ακρίβεια (πχ $a = 10^{-4}$), έχουμε ένα σύνολο από $\frac{1}{a}\left(\frac{L}{2V_{min}} - \frac{L}{2V_{max}}\right) + 1 = \Theta\left(L\left(\frac{1}{V_{min}} - \frac{1}{V_{max}}\right)\right)$ χρονικές στιγμές.
- Κάνουμε δυαδική αναζήτηση σε αυτό το χρονικό διάστημα.
- Αν κάποια στιγμή όλα τα σωματίδια της 1ης κατηγορίας βρίσκονται πιο αριστερά από τα αντίστοιχα της 2ης, κοιτάμε μεταγενέστερες χρονικές στιγμές. Αλλιώς, κοιτάμε προγενέστερες.
- Σε κάθε βήμα, $\mathcal{O}(n)$ χρόνος για να βρούμε τις θέσεις των στοιχείων $\implies \mathcal{O}\left(n \log\left(L\left(\frac{1}{V_{min}} - \frac{1}{V_{max}}\right)\right)\right)$ συνολικά στη **χειρότερη περίπτωση**.

Ερώτημα 1

- Παρατηρούμε πως η πρώτη σύγκρουση θα πρέπει να γίνει κάποια στιγμή στο διάστημα $\left[\frac{L}{2V_{max}}, \frac{L}{2V_{min}}\right]$.
- Θεωρώντας κάποια ακρίβεια (πχ $a = 10^{-4}$), έχουμε ένα σύνολο από $\frac{1}{a}\left(\frac{L}{2V_{min}} - \frac{L}{2V_{max}}\right) + 1 = \Theta\left(L\left(\frac{1}{V_{min}} - \frac{1}{V_{max}}\right)\right)$ χρονικές στιγμές.
- Κάνουμε δυαδική αναζήτηση σε αυτό το χρονικό διάστημα.
- Αν κάποια στιγμή όλα τα σωματίδια της 1ης κατηγορίας βρίσκονται πιο αριστερά από τα αντίστοιχα της 2ης, κοιτάμε μεταγενέστερες χρονικές στιγμές. Αλλιώς, κοιτάμε προγενέστερες.
- Σε κάθε βήμα, $\mathcal{O}(n)$ χρόνος για να βρούμε τις θέσεις των στοιχείων $\implies \mathcal{O}\left(n \log\left(L\left(\frac{1}{V_{min}} - \frac{1}{V_{max}}\right)\right)\right)$ συνολικά στη **χειρότερη περίπτωση**.

Ερώτημα 1

- Παρατηρούμε πως η πρώτη σύγκρουση θα πρέπει να γίνει κάποια στιγμή στο διάστημα $\left[\frac{L}{2V_{max}}, \frac{L}{2V_{min}}\right]$.
- Θεωρώντας κάποια ακρίβεια (πχ $a = 10^{-4}$), έχουμε ένα σύνολο από $\frac{1}{a}\left(\frac{L}{2V_{min}} - \frac{L}{2V_{max}}\right) + 1 = \Theta\left(L\left(\frac{1}{V_{min}} - \frac{1}{V_{max}}\right)\right)$ χρονικές στιγμές.
- Κάνουμε δυαδική αναζήτηση σε αυτό το χρονικό διάστημα.
- Αν κάποια στιγμή όλα τα σωματίδια της 1ης κατηγορίας βρίσκονται πιο αριστερά από τα αντίστοιχα της 2ης, κοιτάμε μεταγενέστερες χρονικές στιγμές. Αλλιώς, κοιτάμε προγενέστερες.
- Σε κάθε βήμα, $\mathcal{O}(n)$ χρόνος για να βρούμε τις θέσεις των στοιχείων $\implies \mathcal{O}\left(n \log\left(L\left(\frac{1}{V_{min}} - \frac{1}{V_{max}}\right)\right)\right)$ συνολικά στη **χειρότερη περίπτωση**.

Ερώτημα 1

- Παρατηρούμε πως η πρώτη σύγκρουση θα πρέπει να γίνει κάποια στιγμή στο διάστημα $\left[\frac{L}{2V_{max}}, \frac{L}{2V_{min}}\right]$.
- Θεωρώντας κάποια ακρίβεια (πχ $a = 10^{-4}$), έχουμε ένα σύνολο από $\frac{1}{a}\left(\frac{L}{2V_{min}} - \frac{L}{2V_{max}}\right) + 1 = \Theta\left(L\left(\frac{1}{V_{min}} - \frac{1}{V_{max}}\right)\right)$ χρονικές στιγμές.
- Κάνουμε δυαδική αναζήτηση σε αυτό το χρονικό διάστημα.
- Αν κάποια στιγμή όλα τα σωματίδια της 1ης κατηγορίας βρίσκονται πιο αριστερά από τα αντίστοιχα της 2ης, κοιτάμε μεταγενέστερες χρονικές στιγμές. Αλλιώς, κοιτάμε προγενέστερες.
- Σε κάθε βήμα, $\mathcal{O}(n)$ χρόνος για να βρούμε τις θέσεις των στοιχείων $\implies \mathcal{O}\left(n \log\left(L\left(\frac{1}{V_{min}} - \frac{1}{V_{max}}\right)\right)\right)$ συνολικά στη **χειρότερη περίπτωση**.

Ερώτημα 1

- Παρατηρούμε πως η πρώτη σύγκρουση θα πρέπει να γίνει κάποια στιγμή στο διάστημα $\left[\frac{L}{2V_{max}}, \frac{L}{2V_{min}}\right]$.
- Θεωρώντας κάποια ακρίβεια (πχ $a = 10^{-4}$), έχουμε ένα σύνολο από $\frac{1}{a}\left(\frac{L}{2V_{min}} - \frac{L}{2V_{max}}\right) + 1 = \Theta\left(L\left(\frac{1}{V_{min}} - \frac{1}{V_{max}}\right)\right)$ χρονικές στιγμές.
- Κάνουμε δυαδική αναζήτηση σε αυτό το χρονικό διάστημα.
- Αν κάποια στιγμή όλα τα σωματίδια της 1ης κατηγορίας βρίσκονται πιο αριστερά από τα αντίστοιχα της 2ης, κοιτάμε μεταγενέστερες χρονικές στιγμές. Αλλιώς, κοιτάμε προγενέστερες.
- Σε κάθε βήμα, $\mathcal{O}(n)$ χρόνος για να βρούμε τις θέσεις των στοιχείων $\implies \mathcal{O}\left(n \log\left(L\left(\frac{1}{V_{min}} - \frac{1}{V_{max}}\right)\right)\right)$ συνολικά στη **χειρότερη περίπτωση**.

Ερώτημα 2

- **Απλή Λύση:** Brute Force → δοκίμασε όλα τα n^2 ζεύγη (πολυπλοκότητα $\mathcal{O}\left(n^2\right)$).
- **Καλή Λύση:** Quickselect στα σωματίδια της 1ης κατηγορίας (πολυπλοκότητα $\mathcal{O}\left(n \log\left(n\right)\right)$).
- **Βέλτιστη Λύση:** Quickselect στα σωματίδια και των 2 κατηγοριών εναλλάξ (πολυπλοκότητα $\mathcal{O}\left(n\right)$).

Ιδέα Καλής Λύσης (I)

- Επίλεξε τυχαία ένα σωματίδιο της 1ης κατηγορίας.
- Βρες πότε θα συγκρουόταν με καθένα από τα σωματίδια της 2ης κατηγορίας (αν δεν είχαμε τις εξαϋλώσεις) και κράτα εκείνο όπου αντιστοιχεί ο μικρότερος χρόνος σύγκρουσης (έστω j^* το σωματίδιο και t^* ο χρόνος).
- Ισχυρισμός: Δεν υπάρχουν σωματίδια της 2ης κατηγορίας που να βρίσκονται πιο αριστερά από το j^* τη στιγμή t^*.

Ιδέα Καλής Λύσης (I)

- Επίλεξε τυχαία ένα σωματίδιο της 1ης κατηγορίας.
- Βρες πότε θα συγκρουόταν με καθένα από τα σωματίδια της 2ης κατηγορίας (αν δεν είχαμε τις εξαϋλώσεις) και κράτα εκείνο όπου αντιστοιχεί ο μικρότερος χρόνος σύγκρουσης (έστω j^* το σωματίδιο και t^* ο χρόνος).
- Ισχυρισμός: Δεν υπάρχουν σωματίδια της 2ης κατηγορίας που να βρίσκονται πιο αριστερά από το j^* τη στιγμή t^*.

Ιδέα Καλής Λύσης (I)

- Επίλεξε τυχαία ένα σωματίδιο της 1ης κατηγορίας.
- Βρες πότε θα συγκρουόταν με καθένα από τα σωματίδια της 2ης κατηγορίας (αν δεν είχαμε τις εξαϋλώσεις) και κράτα εκείνο όπου αντιστοιχεί ο μικρότερος χρόνος σύγκρουσης (έστω j^* το σωματίδιο και t^* ο χρόνος).
- Ισχυρισμός: Δεν υπάρχουν σωματίδια της 2ης κατηγορίας που να βρίσκονται πιο αριστερά από το j^* τη στιγμή t^*.

Ιδέα Καλής Λύσης (II)

- Πράγματι, αν υπήρχαν τέτοια σωματίδια, τότε το τυχαία επιλεγμένο σωματίδιο θα συγκρουόταν με καθένα από αυτά κάποια στιγμή $t < t^*$ (άτοπο).
- Άρα, τα σωματίδια της 1ης κατηγορίας που βρίσκονται αριστερά του σημείου σύγκρουσης δεν μπορούν να συμμετέχουν στην πρώτη σύγκρουση (αφού κάθε στιγμή $t < t^*$ όλα τα σωματίδια της 2ης κατηγορίας θα βρίσκονται πιο δεξιά).
- Έτσι, μπορούμε τα σωματίδια αυτά να τα αγνοήσουμε.
- Επαναλαμβάνοντας το ίδιο όσες φορές χρειαστεί ώστε να μείνει μόνο ένα σωματίδιο της 1ης κατηγορίας, μπορούμε μετά να βρούμε με ποιο από αυτά της 2ης συγκρούεται πρώτα.

Ιδέα Καλής Λύσης (II)

- Πράγματι, αν υπήρχαν τέτοια σωματίδια, τότε το τυχαία επιλεγμένο σωματίδιο θα συγκρουόταν με καθένα από αυτά κάποια στιγμή $t < t^*$ (άτοπο).
- Άρα, τα σωματίδια της 1ης κατηγορίας που βρίσκονται αριστερά του σημείου σύγκρουσης δεν μπορούν να συμμετέχουν στην πρώτη σύγκρουση (αφού κάθε στιγμή $t < t^*$ όλα τα σωματίδια της 2ης κατηγορίας θα βρίσκονται πιο δεξιά).
- Έτσι, μπορούμε τα σωματίδια αυτά να τα αγνοήσουμε.
- Επαναλαμβάνοντας το ίδιο όσες φορές χρειαστεί ώστε να μείνει μόνο ένα σωματίδιο της 1ης κατηγορίας, μπορούμε μετά να βρούμε με ποιο από αυτά της 2ης συγκρούεται πρώτα.

Ιδέα Καλής Λύσης (II)

- Πράγματι, αν υπήρχαν τέτοια σωματίδια, τότε το τυχαία επιλεγμένο σωματίδιο θα συγκρουόταν με καθένα από αυτά κάποια στιγμή $t < t^*$ (άτοπο).
- Άρα, τα σωματίδια της 1ης κατηγορίας που βρίσκονται αριστερά του σημείου σύγκρουσης δεν μπορούν να συμμετέχουν στην πρώτη σύγκρουση (αφού κάθε στιγμή $t < t^*$ όλα τα σωματίδια της 2ης κατηγορίας θα βρίσκονται πιο δεξιά).
- Έτσι, μπορούμε τα σωματίδια αυτά να τα αγνοήσουμε.
- Επαναλαμβάνοντας το ίδιο όσες φορές χρειαστεί ώστε να μείνει μόνο ένα σωματίδιο της 1ης κατηγορίας, μπορούμε μετά να βρούμε με ποιο από αυτά της 2ης συγκρούεται πρώτα.

Ιδέα Καλής Λύσης (II)

- Πράγματι, αν υπήρχαν τέτοια σωματίδια, τότε το τυχαία επιλεγμένο σωματίδιο θα συγκρουόταν με καθένα από αυτά κάποια στιγμή $t < t^*$ (άτοπο).
- Άρα, τα σωματίδια της 1ης κατηγορίας που βρίσκονται αριστερά του σημείου σύγκρουσης δεν μπορούν να συμμετέχουν στην πρώτη σύγκρουση (αφού κάθε στιγμή $t < t^*$ όλα τα σωματίδια της 2ης κατηγορίας θα βρίσκονται πιο δεξιά).
- Έτσι, μπορούμε τα σωματίδια αυτά να τα αγνοήσουμε.
- Επαναλαμβάνοντας το ίδιο όσες φορές χρειαστεί ώστε να μείνει μόνο ένα σωματίδιο της 1ης κατηγορίας, μπορούμε μετά να βρούμε με ποιο από αυτά της 2ης συγκρούεται πρώτα.

Ανάλυση Πολυπλοκότητας

- Μετά από κάθε βήμα, με μεγάλη πιθανότητα θεωρούμε πως μένουν το πολύ τα $\frac{3}{4}$ των σωματιδίων της 1ης κατηγορίας (βλ. ανάλυση QUICKSELECT).
- Άρα, έχουμε $\mathcal{O}(\log(n))$ επαναλήψεις.
- Σε κάθε επανάληψη, κάνουμε μία γραμμική αναζήτηση στα στοιχεία της 2ης κατηγορίας όπου κάθε σύγκρουση υπολογίζεται σε $\mathcal{O}(1)$ χρόνο (άρα $\mathcal{O}(n)$ συνολικά).
- Για τα σωματίδια της 1ης κατηγορίας (που σε κάθε επανάληψη είναι $\leq n$), υπολογίζουμε τη στιγμή που συγκρούονται με το j^*. Αν είναι $> t^*$ καταλαβαίνουμε πως είναι αριστερά από το σημείο σύγκρουσης, οπότε ξεσκαρτάρονται.
- Συνολικός Χρόνος: $\mathcal{O}(n \log(n))$ στη **μέση περίπτωση**.

Ανάλυση Πολυπλοκότητας

- Μετά από κάθε βήμα, με μεγάλη πιθανότητα θεωρούμε πως μένουν το πολύ τα $\frac{3}{4}$ των σωματιδίων της 1ης κατηγορίας (βλ. ανάλυση QUICKSELECT).
- Άρα, έχουμε $\mathcal{O}(\log(n))$ επαναλήψεις.
- Σε κάθε επανάληψη, κάνουμε μία γραμμική αναζήτηση στα στοιχεία της 2ης κατηγορίας όπου κάθε σύγκρουση υπολογίζεται σε $\mathcal{O}(1)$ χρόνο (άρα $\mathcal{O}(n)$ συνολικά).
- Για τα σωματίδια της 1ης κατηγορίας (που σε κάθε επανάληψη είναι $\leq n$), υπολογίζουμε τη στιγμή που συγκρούονται με το j^*. Αν είναι $> t^*$ καταλαβαίνουμε πως είναι αριστερά από το σημείο σύγκρουσης, οπότε ξεσκαρτάρονται.
- Συνολικός Χρόνος: $\mathcal{O}(n \log(n))$ στη **μέση περίπτωση**.

Ανάλυση Πολυπλοκότητας

- Μετά από κάθε βήμα, με μεγάλη πιθανότητα θεωρούμε πως μένουν το πολύ τα $\frac{3}{4}$ των σωματιδίων της 1ης κατηγορίας (βλ. ανάλυση QUICKSELECT).
- Άρα, έχουμε $\mathcal{O}(\log(n))$ επαναλήψεις.
- Σε κάθε επανάληψη, κάνουμε μία γραμμική αναζήτηση στα στοιχεία της 2ης κατηγορίας όπου κάθε σύγκρουση υπολογίζεται σε $\mathcal{O}(1)$ χρόνο (άρα $\mathcal{O}(n)$ συνολικά).
- Για τα σωματίδια της 1ης κατηγορίας (που σε κάθε επανάληψη είναι $\leq n$), υπολογίζουμε τη στιγμή που συγκρούονται με το j^*. Αν είναι $> t^*$ καταλαβαίνουμε πως είναι αριστερά από το σημείο σύγκρουσης, οπότε ξεσκαρτάρονται.
- Συνολικός Χρόνος: $\mathcal{O}(n \log(n))$ στη **μέση περίπτωση**.

Ανάλυση Πολυπλοκότητας

- Μετά από κάθε βήμα, με μεγάλη πιθανότητα θεωρούμε πως μένουν το πολύ τα $\frac{3}{4}$ των σωματιδίων της 1ης κατηγορίας (βλ. ανάλυση QUICKSELECT).
- Άρα, έχουμε $\mathcal{O}(\log(n))$ επαναλήψεις.
- Σε κάθε επανάληψη, κάνουμε μία γραμμική αναζήτηση στα στοιχεία της 2ης κατηγορίας όπου κάθε σύγκρουση υπολογίζεται σε $\mathcal{O}(1)$ χρόνο (άρα $\mathcal{O}(n)$ συνολικά).
- Για τα σωματίδια της 1ης κατηγορίας (που σε κάθε επανάληψη είναι $\leq n$), υπολογίζουμε τη στιγμή που συγκρούονται με το j^*. Αν είναι $> t^*$ καταλαβαίνουμε πως είναι αριστερά από το σημείο σύγκρουσης, οπότε ξεσκαρτάρονται.
- Συνολικός Χρόνος: $\mathcal{O}(n \log(n))$ στη **μέση περίπτωση**.

Ανάλυση Πολυπλοκότητας

- Μετά από κάθε βήμα, με μεγάλη πιθανότητα θεωρούμε πως μένουν το πολύ τα $\frac{3}{4}$ των σωματιδίων της 1ης κατηγορίας (βλ. ανάλυση QUICKSELECT).
- Άρα, έχουμε $\mathcal{O}(\log(n))$ επαναλήψεις.
- Σε κάθε επανάληψη, κάνουμε μία γραμμική αναζήτηση στα στοιχεία της 2ης κατηγορίας όπου κάθε σύγκρουση υπολογίζεται σε $\mathcal{O}(1)$ χρόνο (άρα $\mathcal{O}(n)$ συνολικά).
- Για τα σωματίδια της 1ης κατηγορίας (που σε κάθε επανάληψη είναι $\leq n$), υπολογίζουμε τη στιγμή που συγκρούονται με το j^*. Αν είναι $> t^*$ καταλαβαίνουμε πως είναι αριστερά από το σημείο σύγκρουσης, οπότε ξεσκαρτάρονται.
- Συνολικός Χρόνος: $\mathcal{O}(n \log(n))$ στη **μέση περίπτωση**.

Ιδέα Βέλτιστης Λύσης

- Ο λόγος που πριν είχαμε πολυπλοκότητα $\mathcal{O}(n \log(n))$ ήταν επειδή έπρεπε να κάνουμε γραμμική αναζήτηση στα σωματίδια της 2ης κατηγορίας (που ήταν πάντα n).
- Χρησιμοποιούμε την ίδια τεχνική με την προηγούμενη λύση, αλλά την εφαρμόζουμε εναλλάξ στα σωματίδια και των 2 κατηγοριών.
- Μειώνεται παράλληλα το πλήθος των υποψήφιων σωματιδίων και από τις 2 κατηγορίες.
- Τελική Πολυπλοκότητα: $\mathcal{O}(n)$ στη **μέση περίπτωση**.

Ιδέα Βέλτιστης Λύσης

- Ο λόγος που πριν είχαμε πολυπλοκότητα $\mathcal{O}(n \log(n))$ ήταν επειδή έπρεπε να κάνουμε γραμμική αναζήτηση στα σωματίδια της 2ης κατηγορίας (που ήταν πάντα n).
- Χρησιμοποιούμε την ίδια τεχνική με την προηγούμενη λύση, αλλά την εφαρμόζουμε εναλλάξ στα σωματίδια και των 2 κατηγοριών.
- Μειώνεται παράλληλα το πλήθος των υποψήφιων σωματιδίων και από τις 2 κατηγορίες.
- Τελική Πολυπλοκότητα: $\mathcal{O}(n)$ στη **μέση περίπτωση**.

Ιδέα Βέλτιστης Λύσης

- Ο λόγος που πριν είχαμε πολυπλοκότητα $\mathcal{O}(n \log(n))$ ήταν επειδή έπρεπε να κάνουμε γραμμική αναζήτηση στα σωματίδια της 2ης κατηγορίας (που ήταν πάντα n).
- Χρησιμοποιούμε την ίδια τεχνική με την προηγούμενη λύση, αλλά την εφαρμόζουμε εναλλάξ στα σωματίδια και των 2 κατηγοριών.
- Μειώνεται παράλληλα το πλήθος των υποψήφιων σωματιδίων και από τις 2 κατηγορίες.
- Τελική Πολυπλοκότητα: $\mathcal{O}(n)$ στη **μέση περίπτωση**.

Ιδέα Βέλτιστης Λύσης

- Ο λόγος που πριν είχαμε πολυπλοκότητα $\mathcal{O}(n \log(n))$ ήταν επειδή έπρεπε να κάνουμε γραμμική αναζήτηση στα σωματίδια της 2ης κατηγορίας (που ήταν πάντα n).
- Χρησιμοποιούμε την ίδια τεχνική με την προηγούμενη λύση, αλλά την εφαρμόζουμε εναλλάξ στα σωματίδια και των 2 κατηγοριών.
- Μειώνεται παράλληλα το πλήθος των υποψήφιων σωματιδίων και από τις 2 κατηγορίες.
- Τελική Πολυπλοκότητα: $\mathcal{O}(n)$ στη **μέση περίπτωση**.

Άσκηση 5 - Κρυμμένος Θησαυρός

Amortized Ανάλυση:

- Παίζουμε σε γύρους.
- Στον γύρο i έχουμε ψάξει τους ακέραιους σε απόσταση 2^i.
- Σε κάθε γύρο i ξεκινάμε από το 0 και πηγαίνουμε μέχρι το -2^i, μετά στο 2^i και μετά στο 0 πάλι $\Rightarrow 4 \cdot 2^i$ βήματα

$$\textit{Συνολικά: } \sum_{i=0}^{n} \Theta(2^i) = \Theta(2^n)$$

- Οπότε για να βρούμε τον θησαυρό σε απόσταση $|x|$, πρέπει να φτάσουμε στον γύρο n, όπου 2^n είναι η επόμενη δύναμη του 2 μετά το $|x|$. Άρα $2^n < 2|x|$.

Άσκηση 5 - Κρυμμένος Θησαυρός

Λύση με καλύτερο παράγοντα.

- Ίδια λογική, αλλά σε κάθε γύρο i ξεκινάμε είτε από το -2^{i-1}, είτε από το 2^i.
- Ας υποθέσουμε ότι ξεκινάμε από το -2^{i-1}, και πάμε στο -2^i και μετά στο 2^i. Έχουμε εξευρενήσει όλα τα σημεία σε απόσταση 2^i και κάνουμε συνολικά $5 \cdot 2^{i-1}$ βήματα.

$$\textit{Συνολικά: } 3 + \sum_{i=1}^{n} 5 \cdot 2^{i-1} = 5 \cdot 2^n - 2$$

- Για τον τελικό γύρο n ισχύει $2^{n-1} < |x| \leq 2^n$.

$$\textit{Συνολικά βήματα: } 5 \cdot 2^n - 2 < 10|x| - 2$$

- Μία πρώτη παρατήρηση-κλειδί που είναι απαραίτητη για την κατανόηση και την επίλυση της άσκησης είναι ότι ο (βέλτιστος) χρόνος στον οποίο ένα όχημα μπορεί να πραγματοποιήσει τη διαδρομή από την πόλη A στην πόλη B εξαρτάται μόνο από την χωρητικότητά του σε καύσιμα. Επιπλέον στις K ενδιάμεσες πόλεις-σταθμούς το όχημα μπορεί να γεμίσει πάλι την χωρητικότητά του και έτσι μας ενδιαφέρουν μόνο οι χρόνοι στους οποίους ένα όχημα μπορεί να διασχίσει τις αποστάσεις από τον ένα σταθμό στον άλλο.

Ξεκινάμε υπολογίζοντας τις αποστάσεις αυτές. Επειδή οι θέσεις των σταθμών **δεν** δίνονται ταξινομημένες, πρέπει πρώτα να ταξινομήσουμε τον πίνακα *position* σε αύξουσα σειρά, με χρήση *mergesort/quicksort* σε $O(KlogK)$ (όπως θα δούμε στη συνέχεια η πολυπλοκότητα υπολογισμού του βέλτιστου οχήματος ξεπερνάει το $O(KlogK)$ και επομένως η ταξινόμηση αυτή δεν επηρεάζει την πολυπλοκότητα της λύσης μας). Έπειτα, ο υπολογισμός των $(K+1)$-αποστάσεων ανάμεσα σε αυτούς τους σταθμούς γίνεται γραμμικά ως προς K από την σχέση:

$$distance[i] = \begin{cases} position[0] & ,i = 0 \\ position[i] - position[i-1] & ,0 < i < K \\ S - position[K-1] & ,i = K \end{cases}$$

όπου στον πίνακα *distance* έχουμε συμπεριλάβει και τις αποστάσεις από την πόλη Α (θέση 0) στον πρώτο σταθμό και από τον τελευταίο σταθμό στην πόλη Β (θέση S).

Στην συνέχεια πρέπει να αναλύσουμε τον τρόπο με τον οποίο υπολογίζεται ο βέλτιστος χρόνος στον οποίο ένα όχημα χωρητικότητας C διανύει μία απόσταση D. Ας συμβολίσουμε με $time(C, D)$ αυτήν την ποσότητα. Κάθε όχημα έχει δύο λειτουργίες και μπορεί να κινείται 'αργά' καλύπτοντας απόσταση 1 σε χρόνο T_s και χρησιμοποιώντας καύσιμα C_s είτε 'γρήγορα' καλύπτοντας απόσταση 1 σε χρόνο $T_f < T_s$ αλλά χρησιμοποιώντας περισσότερα καύσιμα $C_f > C_s$. Επομένως, τα ελάχιστα καύσιμα που μπορεί να χρησιμοποιήσει για να διανύσει απόσταση D είναι $D \cdot C_s$ αν κινείται αργά σε όλη τη διαδρομή (κάνοντας τον χειρότερο χρόνο $D \cdot T_s$) ενώ ο βέλτιστος χρόνος είναι $D \cdot T_f$ δεδομένου ότι έχει τουλάχιστον $D \cdot C_f$ χωρητικότητα σε καύσιμα. Στην ενδιάμεση περίπτωση, επειδή θέλουμε να ελαχιστοποιήσουμε τον χρόνο και δεν μας νοιάζει να εξοικονομήσουμε καύσιμα (αφού στους σταθμούς γεμίζουμε όσο θέλουμε) θα πρέπει να κινηθούμε όσο μπορούμε στη γρήγορη λειτουργία και έπειτα να καλύψουμε την υπόλοιπη απόσταση με την αργή λειτουργία.

Από τα παραπάνω, προκύπτει ότι:

$$time(C,D) = \begin{cases} \infty & ,C < D \cdot C_s \\ D \cdot T_s - (C - D \cdot C_s)\frac{T_s - T_f}{C_f - C_s} & ,D \cdot C_s \leq C < D \cdot C_f \\ D \cdot T_f & ,C \geq D \cdot C_f \end{cases}$$

Η παραπάνω σχέση μας λέει ότι αν ένα όχημα έχει χωρητικότητα $C < D \cdot C_s$ τότε δεν μπορεί να καλύψει απόσταση D ακόμα και να πηγαίνει μόνο αργά, άρα θα χρειαστεί άπειρο χρόνο. Αντιθέτως οχήματα με χωρητικότητα $C \geq D \cdot C_f$ μπορούν να καλύψουν όλη την απόσταση κινούμενα γρήγορα και έτσι πετυχαίνουν τον βέλτιστο εφικτό χρόνο $D \cdot T_f$. Στις ενδιάμεσες περιπτώσεις, ένα όχημα θα κινηθεί γρήγορα για να καλύψει απόσταση x και αργά για να καλύψει την υπόλοιπη απόσταση $D - x$. Σε αυτήν την περίπτωση θα χρειαστεί καύσιμα $f = x \cdot C_f + (D - x) \cdot C_s$. Όπως είπαμε ψάχνουμε τον βέλτιστο χρόνο, άρα θα χρησιμοποιήσουμε όλα τα καύσιμά μας. Άρα, από $f = C$ προσδιορίζουμε το x και έπειτα τον χρόνο $time(C,D) = x \cdot T_f + (D - x) \cdot T_s$.

Για τον συνολικό βέλτιστο χρόνο που χρειάζεται ένα όχημα χωρητικότητας C για να μεταβεί από την πόλη Α στην πόλη Β είναι εύκολο να δούμε ότι απλά πρέπει να αθροίσουμε τους χρόνους που απαιτούνται για όλες τις ενδιάμεσες αποστάσεις, δηλαδή:

$$opt - time(C) = \sum_{i=0}^{K} time(C, distance[i])$$

Παρατηρούμε επίσης ότι ο υπολογισμός του $time(C, D)$ γίνεται σε χρόνο $O(1)$ και επομένως ο συνολικός χρόνος που απαιτείται για να μεταβεί ένα όχημα χωρητικότητας C από την πόλη Α στην πόλη Β υπολογίζεται σε χρόνο $O(K)$. Έχοντας ολοκληρώσει την παραπάνω ανάλυση, μπορούμε πλέον να λύσουμε το πρόβλημα.

Ενοικίαση Αυτοκινήτου Λύση 1: O(KN)

Η πιο απλή λύση για το πρόβλημα είναι να εξετάσουμε για κάθε όχημα i χωρητικότητας $capacity[i]$ αν ισχύει ότι $opt - time(capacity[i]) \leq T$. Σε αυτήν την περίπτωση το όχημά μας είναι $feasible$, δηλαδή μπορεί να πραγματοποιήσει το ταξίδι από την πόλη A στην πόλη B σε χρόνο το πολύ T. Δεν μένει παρά να διαλέξουμε το φθηνότερο από τα $feasible$ οχήματα (-1 αν αυτό δεν υπάρχει). Επειδή έχουμε N οχήματα και ο υπολογισμός του $opt - time$ χρειάζεται χρόνο $O(K)$, η συνολική πολυπλοκότητα της λύσης είναι $O(NK)$ και αρκεί για να περάσει μόνο τα μικρά $testcases$ λόγο του χρονικού περιορισμού.

Ενοικίαση Αυτοκινήτου Λύση 2: $O(KlogN)$

Ο λόγος για τον οποίο μπορούμε να βελτιώσουμε την πολυπλοκότητα της λύσης μας, είναι ότι δεν έχουμε αξιοποιήσει την ακόλουθη ιδιότητα του προβλήματος: η συνάρτηση $opt - time(C)$ είναι φθίνουσα ως προς C. Αυτό σημαίνει ότι αν ένα αμάξι χωρητικότητας C δεν μπορεί να κάνει το ταξίδι σε χρόνο το πολύ T τότε κανένα άλλο όχημα με χωρητικότητα μικρότερη του C δεν θα μπορεί. Αντίστοιχα αν ένα αμάξι χωρητικότητας C μπορεί να πραγματοποιήσει το ταξίδι σε χρόνο το πολύ T τότε και κάθε άλλο όχημα με μεγαλύτερη χωρητικότητα θα μπορεί.

Ενοικίαση Αυτοκινήτου Λύση 2: $O(KlogN)$

Η ιδιότητα αυτή μας επιτρέπει να κάνουμε **δυαδική αναζήτηση** και να βελτιώσουμε την πολυπλοκότητα του αλγορίθμου μας. Ταξινομούμε τα οχήματα (τιμές - χωρητικότητες) σε αύξουσα σειρά χωρητικότητας σε χρόνο $O(NlogN)$ με κάποιον καλό αλγόριθμο ταξινόμησης. Έπειτα κάνουμε δυαδική αναζήτηση πάνω στις χωρητικότητες των οχημάτων μέχρι να βρούμε το όχημα ελάχιστης χωρητικότητας που μπορεί να ολοκληρώσει το ταξίδι από την πόλη A στην πόλη B σε χρόνο το πολύ T - αυτό είναι το πρώτο $feasible$ όχημα. Ξεκινάμε από το όχημα N/2, και αν είναι $feasible$ εξετάζουμε το όχημα N/4, διαφορετικά το όχημα 3N/4 κοκ. Συνολικά θα εξετάσουμε $logN$ οχήματα ο έλεγχος για το αν είναι $feasible$ χρειάζεται χρόνο $O(K)$. Έτσι σε χρόνο $O(KlogN)$ μπορέσαμε να προσδιορίσουμε τα $feasible$ οχήματα και έπειτα διαλέγουμε αυτό με την ελάχιστη τιμή, σε γραμμικό ως προς N χρόνο.

Προγραμματιστική 2 - Πολεοδομικό Τέλος (Λύση $O(N^2)$)

- **Είσοδος:** Πίνακας Υψών $H[1...N]$.
- **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 2 - Πολεοδομικό Τέλος (Λύση $O(N^2)$)

- **Είσοδος:** Πίνακας Υψών $H[1...N]$.
- **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 2 - Πολεοδομικό Τέλος (Λύση $O(N)$)

- Συμβολίζουμε με $CL[i]$ το κόστος που προκύπτει αν επιλέξουμε τον $i-$οστό ουρανοξύστη για όλους τους ουρανοξύστες στα αριστερά. Αντίστοιχα συμβολίθουμε ως $CR[i]$ το κόστος που προκύπτει για τους ουρανοξύστες στα δεξιά.
- Η λύση είναι: $max_i\{CL[i] + CR[i] - H[i]\}$
- Θα δείξουμε πως 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 ακόλουθη σχέση: $CL[i] = CL[k] + (i - k)A[i]$ Όπου k η πρώτη θέση αν ξεκινήσουμε από το ι κατευθυνόμενοι προς τα αριστερά όπου $H[k] > H[i]$.
- Το πρόβλημά μας έχει αναχτεί στο να μπορούμε να βρούμε για κάθε i το αμέσως προηγούμενο σημείο k, όπου $H[k] > H[i]$.

Προγραμματιστική 2 - Πολεοδομικό Τέλος (Λύση $O(N)$)

- Συμβολίζουμε με $CL[i]$ το κόστος που προκύπτει αν επιλέξουμε τον $i-$οστό ουρανοξύστη για όλους τους ουρανοξύστες στα αριστερά. Αντίστοιχα συμβολίθουμε ως $CR[i]$ το κόστος που προκύπτει για τους ουρανοξύστες στα δεξιά.
- Η λύση είναι: $max_i\{CL[i] + CR[i] - H[i]\}$
- Θα δείξουμε πως 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 ακόλουθη σχέση: $CL[i] = CL[k] + (i - k)A[i]$ Όπου k η πρώτη θέση αν ξεκινήσουμε από το ι κατευθυνόμενοι προς τα αριστερά όπου $H[k] > H[i]$.
- Το πρόβλημά μας έχει αναχτεί στο να μπορούμε να βρούμε για κάθε i το αμέσως προηγούμενο σημείο k, όπου $H[k] > H[i]$.

Προγραμματιστική 2 - Πολεοδομικό Τέλος (Λύση $O(N)$)

- Συμβολίζουμε με $CL[i]$ το κόστος που προκύπτει αν επιλέξουμε τον $i-$οστό ουρανοξύστη για όλους τους ουρανοξύστες στα αριστερά. Αντίστοιχα συμβολίθουμε ως $CR[i]$ το κόστος που προκύπτει για τους ουρανοξύστες στα δεξιά.
- Η λύση είναι: $max_i\{CL[i] + CR[i] - H[i]\}$
- Θα δείξουμε πως 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 ακόλουθη σχέση: $CL[i] = CL[k] + (i - k)A[i]$ Όπου k η πρώτη θέση αν ξεκινήσουμε από το ι κατευθυνόμενοι προς τα αριστερά όπου $H[k] > H[i]$.
- Το πρόβλημά μας έχει αναχτεί στο να μπορούμε να βρούμε για κάθε i το αμέσως προηγούμενο σημείο k, όπου $H[k] > H[i]$.

Προγραμματιστική 2 - Πολεοδομικό Τέλος (Λύση $O(N)$)

- Συμβολίζουμε με $CL[i]$ το κόστος που προκύπτει αν επιλέξουμε τον $i-$οστό ουρανοξύστη για όλους τους ουρανοξύστες στα αριστερά. Αντίστοιχα συμβολίθουμε ως $CR[i]$ το κόστος που προκύπτει για τους ουρανοξύστες στα δεξιά.
- Η λύση είναι: $max_i\{CL[i] + CR[i] - H[i]\}$
- Θα δείξουμε πως 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 ακόλουθη σχέση: $CL[i] = CL[k] + (i - k)A[i]$ Όπου k η πρώτη θέση αν ξεκινήσουμε από το ι κατευθυνόμενοι προς τα αριστερά όπου $H[k] > H[i]$.
- Το πρόβλημά μας έχει αναχτεί στο να μπορούμε να βρούμε για κάθε i το αμέσως προηγούμενο σημείο k, όπου $H[k] > H[i]$.

Προγραμματιστική 2 - Πολεοδομικό Τέλος (Λύση $O(N)$)

- Συμβολίζουμε με $CL[i]$ το κόστος που προκύπτει αν επιλέξουμε τον $i-$οστό ουρανοξύστη για όλους τους ουρανοξύστες στα αριστερά. Αντίστοιχα συμβολίθουμε ως $CR[i]$ το κόστος που προκύπτει για τους ουρανοξύστες στα δεξιά.
- Η λύση είναι: $max_i\{CL[i] + CR[i] - H[i]\}$
- Θα δείξουμε πως 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 ακόλουθη σχέση: $CL[i] = CL[k] + (i - k)A[i]$ Όπου k η πρώτη θέση αν ξεκινήσουμε από το ι κατευθυνόμενοι προς τα αριστερά όπου $H[k] > H[i]$.
- Το πρόβλημά μας έχει αναχτεί στο να μπορούμε να βρούμε για κάθε i το αμέσως προηγούμενο σημείο k, όπου $H[k] > H[i]$.

- 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 ακόλουθο 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 αυτό που ψάχνουμε.
- Ο παραπάνω αλγόριθμος έχει πολυπλοκότητα $O(N)$ 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.

- 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 ακόλουθο 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 αυτό που ψάχνουμε.
- Ο παραπάνω αλγόριθμος έχει πολυπλοκότητα $O(N)$ 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.

- 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 ακόλουθο 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 αυτό που ψάχνουμε.
- Ο παραπάνω αλγόριθμος έχει πολυπλοκότητα $O(N)$ 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.

- 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 ακόλουθο 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 αυτό που ψάχνουμε.
- Ο παραπάνω αλγόριθμος έχει πολυπλοκότητα $O(N)$ 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.

- 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 ακόλουθο 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 αυτό που ψάχνουμε.
- Ο παραπάνω αλγόριθμος έχει πολυπλοκότητα $O(N)$ 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.

- 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 ακόλουθο 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 αυτό που ψάχνουμε.
- Ο παραπάνω αλγόριθμος έχει πολυπλοκότητα $O(N)$ 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.