ΠΛΗ 30

8η Version της λύσης

25/11/2021

1η Γραπτή Εργασία

Αλγόριθμοι και Πολυπλοκότητα

Ενδεικτικές Απαντήσεις

ΠΕΡΙΕΧΟΜΕΝΑ

ΑΡΝΟΣ Online Education
Σολωμού 29 Αθήνα ☎ 210.38.22.157 ☎ 210.38.22.495
On-line Φροντιστήριο www.arnos.gr info@arnos.gr

ΕΡΩΤΗΜΑ 1 Α

(Α) Έστω f, g μη αρνητικές πραγματικές συναρτήσεις με πεδίο ορισμού το σύνολο των φυσικών αριθμών $\mathbb{N}$.

1. Να αποδείξετε ότι $f(n) + g(n) = \Theta\left(\max\{f(n), g(n)\}\right)$.
2. Να αποδείξετε ότι $f(n) + g(n) = \Omega\left(\min\{f(n), g(n)\}\right)$. Ποια ασυμπτωτική σχέση αρκεί να ισχύει μεταξύ των f, g για να ισχύει επιπλέον $f(n) + g(n) = O\left(\min\{f(n), g(n)\}\right)$; Αιτιολογήστε την απάντησή σας δίνοντας μια κατάλληλη απόδειξη.
3. Αν $0 < g(n) \leq f(n) \leq g(n) + 1$, για κάθε $n \geq 1$, να αποδείξετε ότι δεν ισχύει αναγκαστικά $f(n) = \Theta(g(n))$, δίνοντας ένα κατάλληλο αντιπαράδειγμα.

1. Θα αποδείξουμε ότι $\exists c_1, c_2, n_0 > 0$ τέτοια ώστε

$$c_1 \max\{f(n), g(n)\} \leq f(n) + g(n) \leq c_2 \max\{f(n), g(n)\}, \forall\, n \geq n_0$$

Συμβολισμός Θ – αντιστοιχεί στο =
$\Theta(g(n)) = \{f(n) | \exists\, c_1,\, c_2, n_0 > 0,$ *τέτοια ώστε* $c_1 g(n) \leq f(n) \leq c_2 g(n),\ \forall\, n \geq n_0\}$
Πρακτικά: Γράφουμε $f(n) = \Theta\big(g(n)\big)$ αν η f φράσσεται <u>ανάμεσα</u> σε δύο πολλαπλάσια της g για μεγάλα n – από κάποιο n_0 και πάνω

Αφού $f(n) \geq 0$ και $g(n) \geq 0$ ισχύει:

Αν $\max\{f(n), g(n)\} = f(n)$ τότε $\max\{f(n), g(n)\} = f(n) \leq f(n) + g(n)$ για κάθε $n \geq 1$

Αν $\max\{f(n), g(n)\} = g(n)$ τότε $\max\{f(n), g(n)\} = g(n) \leq f(n) + g(n)$ για κάθε $n \geq 1$

Άρα ισχύει

$$1 \cdot \max\{f(n), g(n)\} \leq f(n) + g(n)$$

Σε κάθε περίπτωση ισχύει $\begin{cases} f(n) \leq \max\{f(n), g(n)\} \\ g(n) \leq \max\{f(n), g(n)\} \end{cases}$ άρα προσθέτοντας κατά μέλη ισχύει

$$f(n) + g(n) \leq 2 \cdot \max\{f(n), g(n)\}$$

Δηλαδή

$$1 \cdot \max\{f(n), g(n)\} \leq f(n) + g(n) \leq 2 \cdot \max\{f(n), g(n)\} \text{ για κάθε } n \geq 1$$

Χρήσιμα παρεμφερή https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-introduction-to-algorithms-sma-5503-fall-2005/assignments/ps1sol.pdf

2. Θα αποδείξουμε ότι $f(n)+g(n)=\Omega(\min\{f(n),g(n)\})$

> **Συμβολισμός Ω – αντιστοιχεί στο $\geq$**
> $\Omega(g(n))=\{f(n)|\ \exists\ c,n_0>0$, τέτοια ώστε $cg(n)\leq f(n),\ \forall\ n\geq n_0\}$
> Πρακτικά: Γράφουμε $f(n)=\Omega\big(g(n)\big)$ αν κάποιο πολλαπλάσιο της g είναι κάτω φράγμα της f για μεγάλα n – από κάποιο n_0 και πάνω

Θα αποδείξουμε ότι $\exists\ c,n_0>0$, τέτοια ώστε $c\cdot\min\{f(n),g(n)\}\leq f(n)+g(n),\ \forall\ n\geq n_0$

Είναι για κάθε $n\geq 1\ \ \min\{f(n),g(n)\}\leq f(n)$

Είναι για κάθε $n\geq 1\ \ \min\{f(n),g(n)\}\leq g(n)$

Άρα $2\cdot\min\{f(n),g(n)\}\leq f(n)+g(n)$

Εκφώνηση:

Ποια ασυμπτωτική σχέση αρκεί να ισχύει μεταξύ των f,g για να ισχύει επιπλέον

$f(n)+g(n)=O(\min\{f(n),g(n)\})$; Αιτιολογήστε την απάντησή σας δίνοντας μια κατάλληλη απόδειξη.

> **Συμβολισμός O – αντιστοιχεί στο $\leq$**
> $O(g(n))=\{f(n)|\ \exists\ c,n_0>0$, τέτοια ώστε $f(n)\leq cg(n),\ \forall\ n\geq n_0\}$
> Πρακτικά: Γράφουμε $f(n)=O\big(g(n)\big)$ αν κάποιο πολλαπλάσιο της g είναι άνω φράγμα της f για μεγάλα n – από κάποιο n_0 και πάνω

Θα πρέπει να $\exists\ c',n_0'>0$, τέτοια ώστε $f(n)+g(n)\leq c'\cdot\min\{f(n),g(n)\}$

Άρα θα υπάρχουν $c,c',n_1=\max\{n_0,n_0'\}$ έτσι ώστε για κάθε $n\geq n_1$ να είναι

$$c\cdot\min\{f(n),g(n)\}\leq f(n)+g(n)\leq c'\cdot\min\{f(n),g(n)\}$$

Άρα η επιπλέον συνθήκη ισχύει αν και μόνο αν $f(n)+g(n)=\Theta(min\{f(n),g(n)\})$

3. Έστω $g(n)=\frac{1}{n}$ και $f(n)=\frac{1}{n}+1$

Είναι $0<\frac{1}{n}\leq\frac{1}{n}+1\leq\frac{1}{n}+1$ για κάθε $n\geq 1$ όμως **δεν** μπορούμε να ισχυριστούμε ότι $f(n)=\Theta\big(g(n)\big)$

Διότι $f(n)=\Theta(1)$ και $g(n)=o(1)$

Δείτε στο **βίντεο** της άσκησης το πως μπορείτε να κατασκευάσετε το **δικό σας παράδειγμα**

ΕΡΩΤΗΜΑ 1 Β

(Β) Να αποδείξετε ότι η συνάρτηση $f(n) = 1^1 + 2^2 + \cdots + n^n = \sum_{i=1}^{n} i^i$, με πεδίο ορισμού το σύνολο των φυσικών αριθμών $\mathbb{N}$, ικανοποιεί τη σχέση $f(n) = \Theta(n^n)$.

Για το άνω φράγμα:

$$1^1 + 2^2 + \cdots + n^n \leq n^1 + n^2 + \cdots . + n^n \leq n\frac{n^n - 1}{n-1} = \frac{n}{n-1} \cdot (n^n - 1)$$

Ισχύει $\frac{n}{n-1} \leq 2 \Leftrightarrow n \leq 2(n-1) \Leftrightarrow n \leq 2n - 2 \Leftrightarrow 2n - n \geq 2 \Leftrightarrow n \geq 2$

Άρα για $n \geq 2$ έχουμε

$$\frac{n}{n-1} \cdot (n^n - 1) \leq 2 \cdot (n^n - 1) = 2 \cdot n^n - 2 \leq 2 \cdot n^n$$

Για το κάτω φράγμα

Για κάθε $n \geq 1$ είναι:

$$1^1 + 2^2 + \cdots + n^n \geq n^n = 1 \cdot n^n$$

Επομένως για κάθε $n \geq 2$ ισχύει $1 \cdot n^n \leq 1^1 + 2^2 + \cdots + n^n \leq 2 \cdot n^n$

Άρα $1^1 + 2^2 + \cdots + n^n = \Theta(n^n)$

ΕΡΩΤΗΜΑ 1 Γ

(Γ) Διατάξτε τις παρακάτω συναρτήσεις, σε αύξουσα σειρά, ως προς τον ρυθμό αύξησής τους:

$f_1(n) = 3^{1.5n}, f_2(n) = (3^n)^2, f_3(n) = n^{10\log_2 n}, f_4(n) = (\log_2 n)^{(\log_2 n)^2}, f_5(n) = n^{100}.$

$f_1(n) = 3^{1.5n} = (3^{1.5})^n = (5.196)^n$

$f_2(n) = (3^n)^2 = 3^{2n} = (3^2)^n = 9^n$

$f_3(n) = n^{10\log n}$

$f_4(n) = (\log n)^{\log^2 n}$

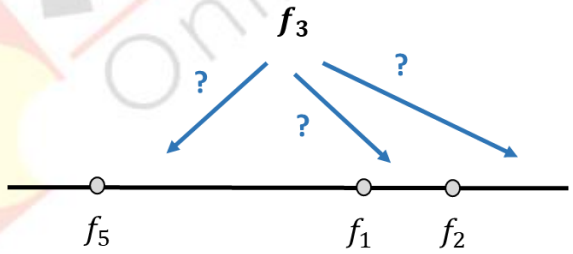
$f_5(n) = n^{100}$

Ως τώρα μπορούμε εύκολα να διαπιστώσουμε ότι $\boldsymbol{f_5 \ll f_1 \ll f_2}$ διότι η f_5 είναι πολυωνυμική, επομένως θα είναι μικρότερη από τις δυο εκθετικές και οι δύο εκθετικές έχουν κοινό εκθέτη και διαφορετική βάση.

Σχέση $\boldsymbol{f_3, f_5}$ → Είναι $\boldsymbol{f_5 \ll f_3}$

Είναι $f_5 = n^{100} \ll n^{10\log n} = f_3$ διότι $100 \ll 10\log n$

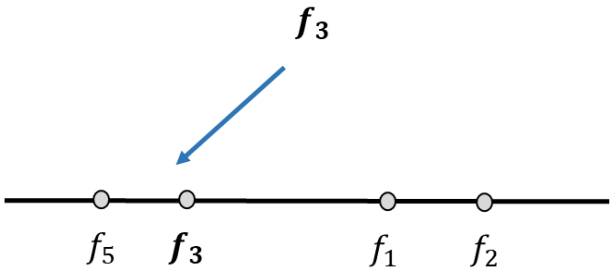
Τώρα θα βρούμε την σχέση της $f_3(n) = n^{10\log n}$ με την $f_1(n) = (5.196)^n$



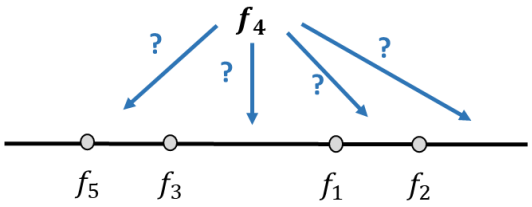
$\log f_3(n) = \log\left(n^{10\log n}\right) = 10\log n \cdot \log n = 10\log^2 n = \Theta(\log^2 n)$

$\log f_1(n) = \log(5.196)^n = n\log 5.196 = n \cdot 2.377 = \Theta(n)$

Άρα $\log f_3(n) \ll \log f_1(n)$ επομένως $\boldsymbol{f_3 \ll f_1}$



Μένει η f_4



Συγκρίνουμε την $\boldsymbol{f_4(n) = (\log n)^{\log^2 n}}$ με την $\boldsymbol{f_2(n) = 9^n}$

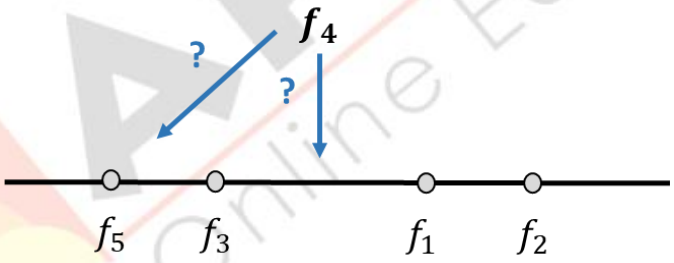
$\log f_4(n) = \log\left((\log n)^{\log^2 n}\right) = \log(\log n)^{\mathbf{\log^2 n}} = \mathbf{\log^2 n} \cdot \log(\log n) = \Theta\left(\mathbf{\log^2 n} \cdot \log(\log n)\right)$

$\log f_2\,(n) = \log 9^n = n \log 9 = n \cdot 3{,}16 = \Theta(n)$

Ισχύει $\log^2 n \cdot \log(\log n) \ll n$ άρα $\log f_4(n) \ll \log f_2\,(n)$ άρα $\boldsymbol{f_4(n) \ll f_2(n)}$

Συγκρίνουμε την $\boldsymbol{f_4(n) = (\log n)^{\log^2 n}}$ με την $\boldsymbol{f_1(n) = (5.196)^n}$

Παρατηρούμε πως λογαριθμίζοντας θα έχουμε παρόμοιο αποτέλεσμα με πριν άρα καταλαβαίνουμε ότι $\boldsymbol{f_4 \ll f_1}$



Συγκρίνουμε την $\boldsymbol{f_4(n) = (\log n)^{\log^2 n}}$ με την $\boldsymbol{f_3(n) = n^{10 \log n}}$

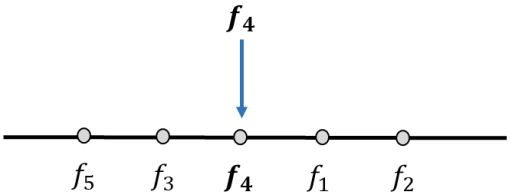
$\log f_4(n) = \log\left((\log n)^{\log^2 n}\right) = \log(\log n)^{\mathbf{\log^2 n}} = \mathbf{\log^2 n} \cdot \log(\log n) = \Theta\left(\mathbf{\log^2 n} \cdot \log(\log n)\right)$

$\log f_3(n) = \log\left(n^{10 \log n}\right) = 10 \log n \cdot \log n = 10 \log^2 n = \Theta\left(\mathbf{\log^2 n}\right)$

Ισχύει $\frac{\mathbf{\log^2 n} \cdot \log(\log n)}{\mathbf{\log^2 n}} = \log(\log n) \to \infty$ άρα $\mathbf{\log^2 n} \ll \mathbf{\log^2 n} \cdot \log(\log n)$

Δηλαδή $\log f_3(n) \ll \log f_4(n)$ άρα $\boldsymbol{f_3 \ll f_4}$

Άρα $\boldsymbol{f_5 \ll f_3 \ll f_4 \ll f_1 \ll f_2}$



ΕΡΩΤΗΜΑ 2

Μια ομάδα μηχανικών υπολογιστών που εργάζονται στην εταιρεία DataBite προσπαθεί να εφεύρει νέους αλγόριθμους ανάλυσης μεγάλου όγκου χρηματιστηριακών και τραπεζικών δεδομένων που περιήλθαν στην κατοχή τους. Συγκεκριμένα, έχουν αναπτύξει πέντε αλγορίθμους «διαίρει και βασίλευε» με τα εξής χαρακτηριστικά:

- Ο αλγόριθμος A_1 διασπά το αρχικό πρόβλημα μεγέθους n σε τρία υποπροβλήματα μεγέθους $n/3$, $n/6$ και $n/9$, τα επιλύει και στη συνέχεια συνθέτει τις λύσεις τους σε χρόνο n.
- Ο αλγόριθμος A_2 διασπά το αρχικό πρόβλημα μεγέθους n σε 6 υποπροβλήματα μεγέθους $n/6$, τα επιλύει και στη συνέχεια συνθέτει τις λύσεις τους σε χρόνο $6n$.
- Ο αλγόριθμος A_3 διασπά το αρχικό πρόβλημα μεγέθους n σε 256 υποπροβλήματα μεγέθους $n/4$, τα επιλύει και στη συνέχεια συνθέτει τις λύσεις τους σε χρόνο $8n^4$.
- Ο αλγόριθμος A_4 διασπά το αρχικό πρόβλημα μεγέθους n σε 49 υποπροβλήματα μεγέθους $n/343$, τα επιλύει και στη συνέχεια συνθέτει τις λύσεις τους σε χρόνο $5n^2$.
- Ο αλγόριθμος A_5 διασπά το αρχικό πρόβλημα μεγέθους n σε 625 υποπροβλήματα μεγέθους $n/125$, τα επιλύει και στη συνέχεια συνθέτει τις λύσεις τους σε χρόνο $10n^{5/4}$.

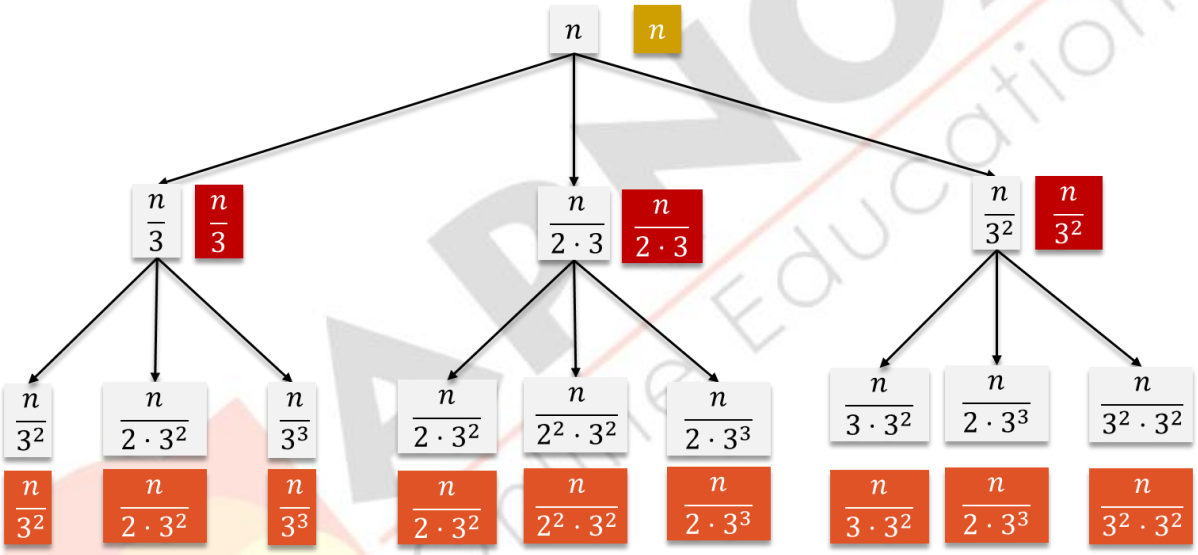
Προκειμένου να μπορέσουν να εκτιμήσουν τον χρόνο επίλυσης κάθε αλγορίθμου, οι μηχανικοί υπολογιστών ενδιαφέρονται για την ασυμπτωτική πολυπλοκότητα καθενός από αυτούς και ζητούν τη βοήθειά σας. Συγκεκριμένα σας ζητούν:

ΕΡΩΤΗΜΑ 2 Α

(Α) Να γράψετε την αναδρομική εξίσωση που δίνει τον χρόνο εκτέλεσης του αλγορίθμου A_1 και στη συνέχεια να την επιλύσετε με το δέντρο της αναδρομής.

Ανακαλύπτουμε το μοτίβο:

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{n}{6}\right) + T\left(\frac{n}{9}\right) + n$$



Στο κόκκινο επίπεδο έχουμε:

$$\underbrace{\frac{n}{3}}_{2\cdot 3^1} + \underbrace{\frac{n}{2\cdot 3}}_{3^1} + \underbrace{\frac{n}{3^2}}_{2} = \frac{(2\cdot 3^1 + 3^1 + 2)\cdot n}{2\cdot 3^2} = \boxed{\left(\frac{11}{2\cdot 3^2}\right)^1 n}$$

Στο πορτοκαλί επίπεδο έχουμε:

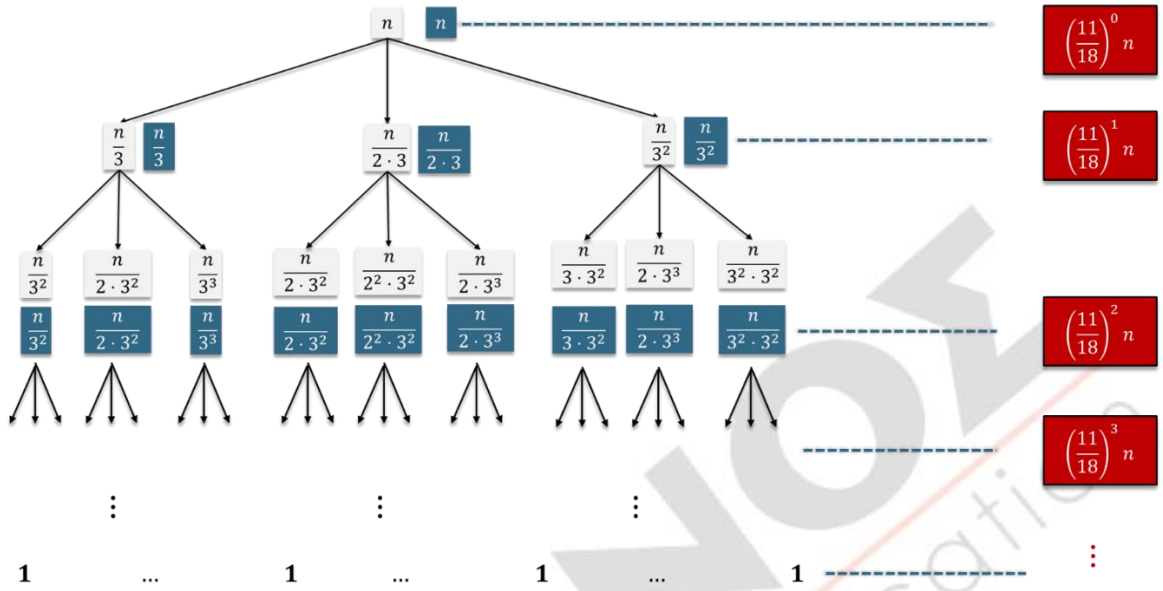
$$\frac{n}{3^2} + \frac{n}{2\cdot 3^2} + \frac{n}{3^3} + \frac{n}{2\cdot 3^2} + \frac{n}{2^2\cdot 3^2} + \frac{n}{2\cdot 3^3} + \frac{n}{3\cdot 3^2} + \frac{n}{2\cdot 3^3} + \frac{n}{3^2\cdot 3^2} =$$

$$\underbrace{\frac{n}{3^2}}_{2^2\cdot 3^2} + \underbrace{\frac{n}{2\cdot 3^2}}_{2^1\cdot 3^2} + \underbrace{\frac{n}{3^3}}_{2^2\cdot 3^1} + \underbrace{\frac{n}{2\cdot 3^2}}_{2^1\cdot 3^2} + \underbrace{\frac{n}{2^2\cdot 3^2}}_{3^2} + \underbrace{\frac{n}{2\cdot 3^3}}_{2^1\cdot 3^1} + \underbrace{\frac{n}{3\cdot 3^2}}_{2^2\cdot 3^1} + \underbrace{\frac{n}{2\cdot 3^3}}_{2^1\cdot 3^1} + \underbrace{\frac{n}{3^2\cdot 3^2}}_{2^2} =$$

$$\frac{(2^2\cdot 3^2 + 2^1\cdot 3^2 + 2^2\cdot 3^1 + 2^1\cdot 3^2 + 3^2 + 2^1\cdot 3^1 + 2^2\cdot 3^1 + 2^1\cdot 3^1 + 2^2)\cdot n}{2^2\cdot 3^4} = \frac{121\cdot n}{2^2\cdot 3^4} = \boxed{\left(\frac{11}{2\cdot 3^2}\right)^2 n}$$

ΑΡΝΟΣ Online Education
Σολωμού 29 Αθήνα ☎ 210.38.22.157 ☎ 210.38.22.495
On-line Φροντιστήριο www.arnos.gr info@arnos.gr

Άρα είναι:



Αρχικά θέτουμε k το ύψος του δέντρου Γνωρίζουμε ότι το ύψος του δέντρο θα είναι της τάξης του logn. Συγκεκριμένα, το ύψος του δέντρου θα καθοριστεί από το πιο αργό μονοπάτι, το οποίο είναι το αριστερότερο μονοπάτι του δέντρου. Άρα θα είναι $k = \log_3 n$

Η λύση της αναδρομικής εξίσωσης προκύπτει αν αθροίσουμε τη συνεισφορά κάθε επιπέδου. Το ύψος του δέντρου θα προσδιοριστεί από τον αριθμό των μειώσεων στο μέγεθος του μεγαλύτερου υποπροβλήματος.

Έχουμε λοιπόν

$$T(n) = \sum_{i=0}^{k} \left(\frac{11}{18}\right)^i n$$

<u>Κάτω φράγμα</u>

Ισχύει

$$T(n) = \sum_{i=0}^{k} \left(\frac{11}{18}\right)^i n = n + \sum_{i=1}^{k} \left(\frac{11}{18}\right)^i n \geq n$$

Άρα $\boxed{1 \cdot n \leq T(n)}$ για $n \geq 1$ δηλαδή $\boxed{\boldsymbol{T(n) = \Omega(n)}}$

<u>Άνω φράγμα</u>

Έχουμε ακόμη

$$\sum_{i=0}^{k} \left(\frac{11}{18}\right)^i n \leq \sum_{i=0}^{\infty} \left(\frac{11}{18}\right)^i n = n \cdot \frac{1}{1 - \frac{11}{18}} = n \cdot \frac{1}{\frac{7}{18}} = \frac{18}{7} n$$

Άρα $\boxed{T(n) \leq \frac{18}{7} n}$ για $n \geq 1$ δηλαδή $\boxed{\boldsymbol{T(n) = O(n)}}$ **<u>Άρα</u>** $\boxed{\boldsymbol{T_1(n) = \Theta(n)}}$

ΑΡΝΟΣ Online Education
Σολωμού 29 Αθήνα ☎ 210.38.22.157 ☎ 210.38.22.495
On-line Φροντιστήριο www.arnos.gr info@arnos.gr

ΕΡΩΤΗΜΑ 2 Β

(Β) Να γράψετε την αναδρομική εξίσωση που δίνει τον χρόνο εκτέλεσης του αλγορίθμου A_2 και στη συνέχεια:

i. Να την επιλύσετε με χρήση του Θεωρήματος της Κυριαρχίας.

ii. Να επαληθεύστε την απάντηση του υποερωτήματος (i) για τον χρόνο εκτέλεσης $T_2(n)$ του αλγορίθμου A_2 με τη μέθοδο της αντικατάστασης, προσδιορίζοντας τις σταθερές n_0, c_1 και c_2 του ορισμού του ασυμπτωτικού συμβολισμού $\Theta(\ )$ (Τόμος Α', ενότητα 2.1.1, σελ. 35). Ως αρχική συνθήκη, ισχύει ότι $T_2(x) = 1, \forall 0 < x < 1$.

Θεώρημα Κυριαρχίας: Έστω η αναδρομική εξίσωση $T(n) = aT(n/b) + f(n)$, όπου $a \geq 1$, $b > 1$ είναι σταθερές, και $f(n)$ είναι μια ασυμπτωτικά θετική συνάρτηση. Τότε διακρίνονται οι ακόλουθες τρεις περιπτώσεις:

(1) αν $f(n) = O(n^{\log_b a - \varepsilon})$, *για κάποια σταθερά* $\varepsilon > 0$, *τότε* $T(n) = \Theta(n^{\log_b a})$,

(2) αν $f(n) = \Theta(n^{\log_b a})$, *τότε* $T(n) = \Theta(n^{\log_b a} \log n)$,

(3) αν $f(n) = \Omega(n^{\log_b a + \varepsilon})$, *για κάποια σταθερά* $\varepsilon > 0$, *και αν υπάρχει σταθερά* n_0, *τέτοια ώστε, για κάθε* $n \geq n_0$, $af(n/b) \leq cf(n)$ *για κάποια σταθερά* $c < 1$, *τότε* $T(n) = \Theta(f(n))$.

i) $T_2(n) = 6\,T_2\left(\frac{n}{6}\right) + 6n$ είναι $\log_6 6 = 1$

Συγκρίνουμε την $f(n) = 6n$ με την n^1

Ισχύει $f(n) = \Theta\left(n^{\log_6 6}\right)$ άρα από την 2η περίπτωση του θεωρήματος κυριαρχίας έχουμε:

$$\boldsymbol{T_2(n) = \Theta(n \log n)}$$

ii) Κατ'αρχας παραλείπουμε τον δείκτη 2 για λόγους ευκολίας. Είναι λοιπόν $T(n) = 6T\left(\frac{n}{6}\right) + 6n$

Με βάση την απάντηση που δώσαμε στο υποερώτημα (Β),προβλέπουμε ότι η λύση της Τ(n) ικανοποιεί τις ανισότητες , $c_1 n \log n \leq T(n) \leq c_2\, n \log n$ για κάποιες σταθερές c_1 και c_2 καθώς και για τιμές του n μεγαλύτερες από κάποιο κατώφλι n_0 που πρέπει να προσδιορίσουμε.

ΑΝΩ ΦΡΑΓΜΑ

Αρχίζουμε με την επαλήθευση του **άνω φράγματος** και τον προσδιορισμό της αντίστοιχης σταθεράς c_2. Καταρχάς, λαμβάνοντας υπόψη και τις αρχικές συνθήκες $T(x) = 1\ \forall x, 0 < x < 1$, παρατηρούμε ότι για n = 1 **δεν** υπάρχει σταθερά c_2 η οποία να ικανοποιεί την ανισότητα

$$\boxed{T(n) = 6 \cdot T\left(\frac{n}{6}\right) + 6 \cdot n \leq c_2\, n \log n}$$

Σχόλιο: Έχουμε $T(x) = 1\ \forall x, 0 < x < 1$ άρα πχ το $T\left(\frac{1}{6}\right)$ ισούται με **1**, το $T\left(\frac{2}{6}\right)$ ισούται με **1**κ.λπ.

Για $\boldsymbol{n = 1}$ → $T(1) = 6 \cdot T\left(\frac{1}{6}\right) + 6 \cdot 1 = 6 \cdot 1 + 6 = 12 \leq c_2 \cdot 1 \cdot \log 1 = 0$

Δεν υπάρχει σταθερά c_2 η οποία να ικανοποιεί την ανισότητα

Για $\boldsymbol{n = 2}$ → $T(2) = 6 \cdot T\left(\frac{2}{6}\right) + 6 \cdot 2 = 6 \cdot 1 + 12 = 18 \leq c_2 \cdot 2 \cdot \log 2 = c_2 \cdot 2$

Άρα είναι $18 \leq c_2 \cdot 2$. Λύνοντας ως προς c_2 έχουμε $c_2 \geq \frac{18}{2} = 9$.

Άρα η τελευταία ισχύει για κάθε $\boldsymbol{c_2 \geq 9}$

Για $\boldsymbol{n = 3}$ → $T(3) = 6 \cdot T\left(\frac{3}{6}\right) + 6 \cdot 3 = 6 \cdot 1 + 18 = 24 \leq c_2 \cdot 3 \log 3 = c_2 \cdot 1{,}58$

Άρα είναι $24 \leq c_2 \cdot 3 \log 3$. Λύνοντας ως προς c_2 έχουμε $c_2 \geq \frac{24}{3 \log 3.} = 5.047$

Άρα η τελευταία ισχύει για κάθε $\boldsymbol{c_2 \geq 5.047}$

Με την παραπάνω διαδικασία προκύπτουν

Για $\boldsymbol{n = 5}$ είναι $\boldsymbol{c_2 \geq 3.10}$.

Για $\boldsymbol{n = 6}$ είναι $\boldsymbol{c_2 \geq 2.708}$.

Άρα μέχρι εδώ, κάθε $\boldsymbol{c_2 \geq 9}$ ικανοποιεί όλες τις παραπάνω ανισώσεις που έχουν προκύψει για το c_2

Πότε παύει το $\frac{n}{6}$ να είναι στο διάστημα $(0, 1)$;

Όταν $n \geq 6$

Για $\boldsymbol{n = 7}$ → $T(7) = 6 \cdot T\left(\frac{7}{6}\right) + 6 \cdot 7$

$T\left(\frac{7}{6}\right) = 6 \cdot T\left(\frac{7}{6^2}\right) + 6 = 6 \cdot 1 + 6 = 12$

Άρα $T(7) = 6 \cdot 12 + 6 \cdot 7 = 114 \leq c_2 \cdot 7 \cdot \log 7$

Άρα είναι $c_2 \geq \frac{114}{7 \log 7} = 5.8$

ΑΡΝΟΣ Online Education
Σολωμού 29 Αθήνα ☎ 210.38.22.157 ☎ 210.38.22.495
On-line Φροντιστήριο www.arnos.gr info@arnos.gr

Ομοίως επαληθεύεται η ανισότητα για κάθε $n \leq 11$ και

Για $n \geq 12$ επαληθεύουμε την ανισότητα εφαρμόζοντας την μέθοδο της αντικατάστασης, όπου μπορούμε επαγωγικά να υποθέσουμε ότι η πρόβλεψή μας ισχύει για κάθε τιμή του n στο διάστημα $[2, n-1]$.

Άρα $\boldsymbol{T\left(\frac{n}{6}\right) \leq c_2 \frac{n}{6} log \frac{n}{6}}$ αφού $n \geq 12 \Leftrightarrow \frac{n}{6} \geq 2$ και $\frac{n}{6} \leq n-1 \Leftrightarrow n \leq 6n-6 \Leftrightarrow n \geq \frac{6}{5}$

Με άλλα λόγια για $n \geq 8$ ισχύει ότι $2 \leq \frac{n}{6} \leq n-1$ άρα μπορούμε να εφαρμόσουμε την επαγωγική υπόθεση.

Αντικαθιστώντας αυτή την τελευταία σχέση στην αναδρομή μας έχουμε:

$$T(n) = 6 \cdot \boldsymbol{T\left(\frac{n}{6}\right)} + 6 \cdot n \leq 6 \cdot \left(c_2 \frac{n}{6} \log \frac{n}{6}\right) + 6 \cdot n = c_2 \cdot 6 \cdot \frac{n}{6} \log \frac{n}{6} + 6 \cdot n = c_2\, n \log \frac{n}{6} + 6 \cdot n =$$

$$c_2\, n \log n - c_2\, n \log 6 + 6n \leq c_2\, n \log n$$

Η τελευταία ανισότητα ισχύει για $-c_2\, n \log 6 + 6n \leq 0 \Leftrightarrow 6n \leq c_2\, n \log 6 \Leftrightarrow c_2 \geq \frac{6}{\log 6} = \frac{6}{2.58} = 2.321$

Άρα πράγματι $\boxed{T(n) \leq c_2\, n \log n \;\; \forall n \geq 2, \forall c_2 \geq 9}$

ΚΑΤΩ ΦΡΑΓΜΑ

Αναζητούμε σταθερά c_1 και n_0 τέτοιο ώστε για κάθε $n \geq n_0$ να ισχύει:

$$\boxed{c_1\, n \log n \leq T(n) = 6 \cdot T\left(\frac{n}{6}\right) + 6 \cdot n}$$

Για $\boldsymbol{n = 1}$ → $c_1 \cdot \mathbf{1} \cdot \log \mathbf{1} \leq T(\mathbf{1})$

Άρα $0 \leq 6 \cdot T\left(\frac{\mathbf{1}}{6}\right) + 6 \cdot \mathbf{1} = 6 \cdot \mathbf{1} + 6 = 12$

Κάθε σταθερά c_1 ικανοποιεί την ανισότητα

Για $\boldsymbol{n = 2}$→ $c_1 \cdot \mathbf{2} \cdot \log \mathbf{2} \leq T(\mathbf{2})$

Άρα $2 \cdot\ c_1 \leq\ T(\mathbf{2}) = 6 \cdot T\left(\frac{\mathbf{2}}{6}\right) + 6 \cdot \mathbf{2} = 6 \cdot \mathbf{1} + 12 = 18$

$2\, c_1 \leq 18 \Leftrightarrow c_1 \leq 9$

Για $\boldsymbol{n = 3}$→ $c_1 \cdot \mathbf{3} \cdot \log \mathbf{3} \leq T(\mathbf{3})$

Άρα $c_1 \cdot \mathbf{3} \cdot \log \mathbf{3} \leq T(\mathbf{3}) = 6 \cdot T\left(\frac{\mathbf{3}}{6}\right) + 6 \cdot \mathbf{3} = 6 \cdot \mathbf{1} + 18 = 24$

$c_1 \cdot \mathbf{3} \cdot \log \mathbf{3} \leq 24 \Leftrightarrow c_1 \cdot 4.75 \leq 24 \Leftrightarrow c_1 \leq 5.04$

Για $\boldsymbol{n = 4}$→ $c_1 \cdot \mathbf{4} \cdot \log \mathbf{4} \leq T(\mathbf{4})$

Άρα $c_1 \cdot \mathbf{4} \cdot \log \mathbf{4} \leq T(\mathbf{4}) = 6 \cdot T\left(\frac{\mathbf{4}}{6}\right) + 6 \cdot \mathbf{4} = 6 \cdot \mathbf{1} + 24 = 30$

$c_1 \cdot \mathbf{4} \cdot \log \mathbf{4} \leq 30 \Leftrightarrow c_1 \cdot 8 \leq 30 \Leftrightarrow c_1 \leq 3.75$

Για $\boldsymbol{n = 5}$ → $c_1 \cdot 5 \cdot \log 5 \leq T(5)$

Άρα $c_1 \cdot 5 \cdot \log 5 \leq T(5) = 6 \cdot T\left(\frac{5}{6}\right) + 6 \cdot 5 = 6 \cdot 1 + 30 = 36$

$$c_1 \cdot 5 \cdot \log 5 \leq 36 \Leftrightarrow c_1 \cdot 11.6 \leq 36 \Leftrightarrow \boldsymbol{c_1 \leq 3.1}$$

Για $\boldsymbol{n \geq 6}$ επαληθεύουμε την ανισότητα εφαρμόζοντας την μέθοδο της αντικατάστασης, όπου μπορούμε επαγωγικά να υποθέσουμε ότι η πρόβλεψή μας ισχύει για κάθε τιμή μικρότερη του n.

Άρα $c_1 \frac{n}{6} \log \frac{n}{6} \leq T\left(\frac{n}{6}\right)$ ($n \geq 6 \Leftrightarrow \frac{n}{6} \geq 1$ και $\frac{n}{6} \leq n - 1$ και άρα μπορούμε να εφαρμόσουμε την επαγωγική υπόθεση).

Αντικαθιστώντας στην τελευταία σχέση από την αναδρομή παίρνουμε:

$$T(n) = 6 \cdot T\left(\frac{n}{6}\right) + 6n \geq 6 \cdot c_1 \frac{n}{6} \log \frac{n}{6} + 6n = n \cdot c_1 \cdot \log \frac{n}{6} + 6n = nc_1 \log n - nc_1 \log 6 + 6n \geq n\, c_1 \log n$$

Η τελευταία ισχύει για $-nc_1 \log 6 + 6n \geq 0 \Leftrightarrow 6n \geq nc_1 \log 6 \Leftrightarrow c_1 \leq \frac{6}{\log 6} = 2.32$

Συνοψίζοντας

Άρα πράγματι $\boxed{c_1\, n \log n \leq T(n)\ \forall n \geq 1, \forall c_1 \leq 2.32}$

ΕΡΩΤΗΜΑ 2 Γ

(Γ) Να γράψετε τις αναδρομικές εξισώσεις που δίνουν τον χρόνο εκτέλεσης των αλγορίθμων A_3, A_4 και A_5 και στη συνέχεια να τις επιλύσετε με χρήση του Θεωρήματος της Κυριαρχίας.

Είναι:

- $T_3(n) = 256\,T_3\left(\frac{n}{4}\right) + 8n^4$
- $T_4(n) = 49\,T_4\left(\frac{n}{343}\right) + 5n^2$
- $T_5(n) = 625\,T_5\left(\frac{n}{125}\right) + 10n^{\frac{5}{4}}$

$\boldsymbol{T_3(n) = 256\,T_3\left(\frac{n}{4}\right) + 8n^4}$

$\log_4 256 = \log_4 4^4 = 4$

Συγκρίνουμε την $f(n) = 8n^4$ με την n^4

Ισχύει $f(n) = \Theta(n^4)$ άρα από την 2η περίπτωση του θεωρήματος κυριαρχίας έχουμε:

$$\boldsymbol{T_3(n) = \Theta\left(n^4 \cdot \log n\right)}$$

$\boldsymbol{T_4(n) = 49\,T_4\left(\frac{n}{343}\right) + 5n^2}$

$\log_{343} 49 = \log_{7^3} 7^2 = \log_{7^3}(7^3)^{\frac{2}{3}} = \frac{2}{3}$

Συγκρίνουμε την $f(n) = 5n^2$ με την $n^{\frac{2}{3}}$

Είναι $f(n) = \Omega\left(n^{\frac{2}{3}+0.00001}\right)$

Μένει να ελέγξουμε την εξτρά συνθήκη της 3ης περίπτωσης του θεωρήματος κυριαρχίας

Πρέπει να υπάρχει $c < 1$ ώστε για τα μεγάλα n να ισχύει

$$a \cdot f\left(\frac{n}{b}\right) \le cf(n)$$

Δηλαδή:

$$49 \cdot 5\left(\frac{n}{343}\right)^2 \le c \cdot 5n^2$$

Που για $n \ge 1$ γίνεται $\frac{49 \cdot 5}{343^2} \le 5c$ δηλαδή $c \ge \frac{49}{343^2} = \frac{7^2}{(7^3)^2} = \frac{1}{7^4}$

Θέτοντας c οποιαδήποτε σταθερά στο $\left[\frac{1}{7^4}, 1\right)$ για $n \ge 1$ έχουμε το ζητούμενο.

Άρα $T_4(n) = \Theta(5n^2)$ δηλαδή $\boldsymbol{T_4(n) = \Theta\left(n^2\right)}$

$\boldsymbol{T_5(n) = 625\, T_5\left(\frac{n}{125}\right) + 10n^{\frac{5}{4}}}$

$\log_{125} 625 = \log_{5^3} 5^4 = \log_{5^3}(5^3)^{\frac{4}{3}} = \frac{4}{3} = 1.33$

Συγκρίνουμε την $f(n) = 10n^{\frac{5}{4}} = 10n^{1.25}$ με την $n^{1.33}$

Ισχύει $f(n) = O(n^{1.33-0.000001})$ άρα από το θεώρημα κυριαρχίας είναι $\boldsymbol{T_5(n) = \Theta(n^{1.33})}$

ΕΡΩΤΗΜΑ 2 Δ

(Δ) Να προσδιορίσετε ποιος είναι ο ασυμπτωτικά αποδοτικότερος αλγόριθμος μεταξύ των A_i (i = 1,2,3,4,5) και να δικαιολογήσετε την απάντησή σας.

Έχουμε:

$\boldsymbol{T_1(n) = \Theta(n)}$

$\boldsymbol{T_2(n) = \Theta(n \log n)}$

$\boldsymbol{T_3(n) = \Theta(n^4 \cdot \log n)}$

$\boldsymbol{T_4(n) = \Theta(n^2)}$

$\boldsymbol{T_5(n) = \Theta(n^{1.33})}$

Αποδοτικότερος είναι ο A_1 με $\boldsymbol{T_1(n) = \Theta(n)}$

ΑΡΝΟΣ Online Education
Σολωμού 29 Αθήνα ☎ 210.38.22.157 ☎ 210.38.22.495
On-line Φροντιστήριο www.arnos.gr info@arnos.gr

ΕΡΩΤΗΜΑ 3

Έστω σύνολο S που περιέχει ακέραιους αριθμούς που είναι όλοι διαφορετικοί και έστω $|S| = n$. Θέλουμε να καταχωρίσουμε τα στοιχεία του συνόλου S σε έναν πίνακα $A[0..n-1]$ ώστε να σχηματισθεί δέντρο-σωρός μεγίστων. Ένα *δέντρο-σωρός μεγίστων* είναι ένα πλήρες δυαδικό δέντρο με διατεταγμένους τους κόμβους του έτσι ώστε η τιμή του στοιχείου κάθε κόμβου να είναι μεγαλύτερη από ή ίση με τις τιμές των στοιχείων των παιδιών του (ΠΛΗ10 Τόμος Γ', Ι. Χατζηλυγερούδης, Δομές Δεδομένων, Β' Έκδοση, 2008, σελ. 188). Υπενθυμίζεται ότι σε ένα δέντρο-σωρό αποθηκευμένο σε πίνακα $A[0..m]$, το αριστερό παιδί του κόμβου $A[j]$ αποθηκεύεται στη θέση $A[2j+1]$ (εφόσον $2j+1 \leq m$) και το δεξί παιδί αποθηκεύεται στη θέση $A[2j+2]$ (εφόσον $2j+2 \leq m$).

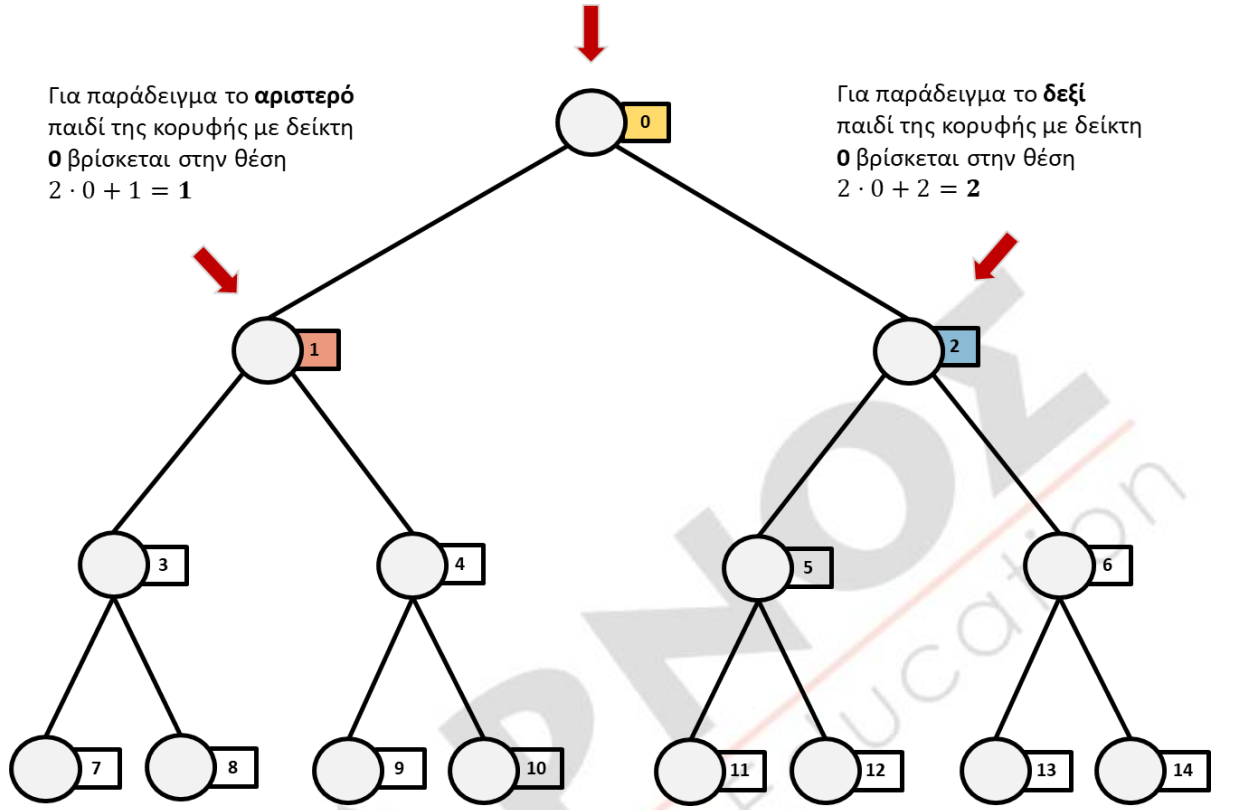
ΕΡΩΤΗΜΑ 3 Α

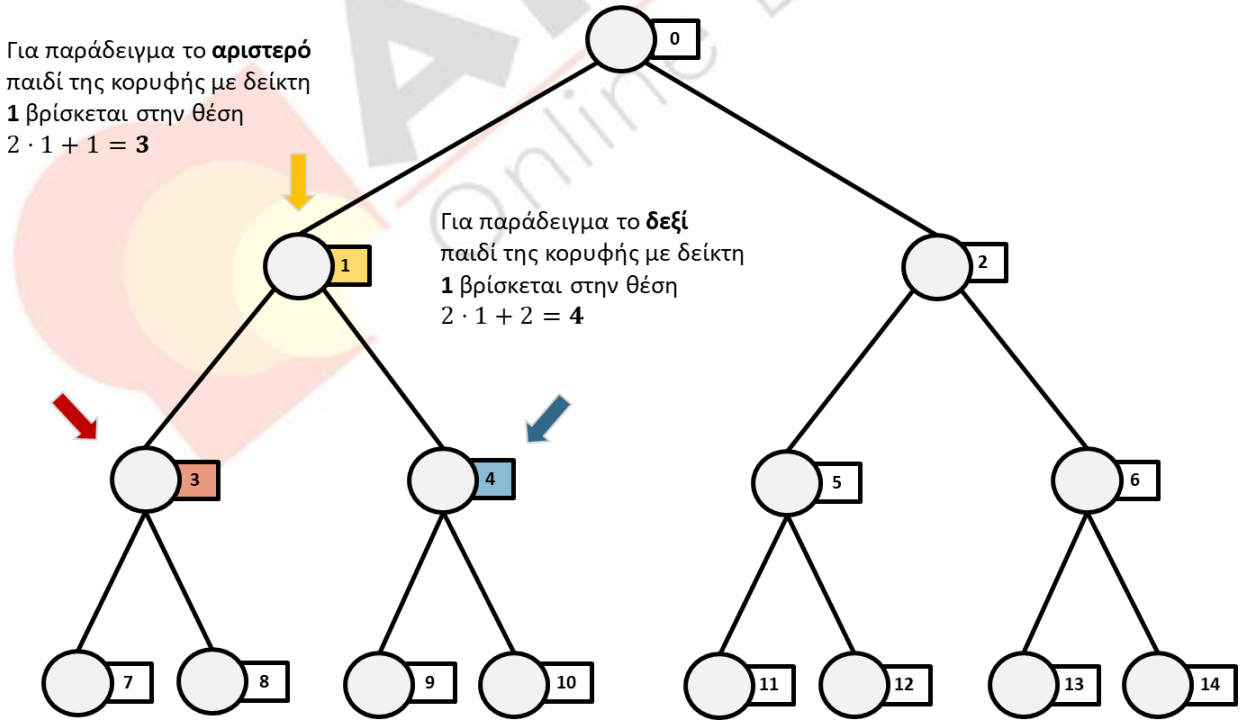
(Α) (i) Λαμβάνοντας υπόψιν ότι ένα δέντρο-σωρός μεγίστων H αποτελείται από τη ρίζα του (που αποθηκεύει το μέγιστο από τα αποθηκευμένα στοιχεία) και δύο υπο-δέντρα-σωρούς μεγίστων με ρίζες τα παιδιά της ρίζας του H, περιγράψτε αλγόριθμο διαίρει-και-βασίλευε με πολυπλοκότητα χρόνου $O(n \log n)$ για την κατασκευή στον πίνακα $A[0..n-1]$ δέντρου-σωρού μεγίστων με όλα τα στοιχεία του συνόλου S όταν ισχύει ότι $|S| = n = 2^k - 1$ (σε αυτήν την περίπτωση όλα τα επίπεδα του δέντρου-σωρού είναι πλήρη) *χωρίς να χρησιμοποιήσετε ταξινόμηση των στοιχείων οποιουδήποτε υποσυνόλου του συνόλου S.*

Σημείωση: Για να μπορέσετε να καταχωρίσετε τα στοιχεία στη σωστή τους θέση, είναι σκόπιμο σε κάθε αναδρομική κλήση να δίνετε ως όρισμα (επιπλέον του πίνακα A και του προς επεξεργασία υποσυνόλου του συνόλου S) τη θέση (στον πίνακα A) της ρίζας του υπο-δέντρου-σωρού στο οποίο θα αποθηκευτούν τα στοιχεία του υποσυνόλου του S.

(ii) Αποδείξτε ότι η πολυπλοκότητα χρόνου του αλγορίθμου σας είναι $O(n \log n)$.

Παρατήρηση 1η: Ο τρόπος με τον οποίο ορίζονται οι θέσεις των παιδιών σημαίνει ότι το δέντρο συμπληρώνεται με τους δείκτες από πάνω προς τα κάτω και από τα αριστερά προς τα δεξιά.





ΑΡΝΟΣ Online Education
Σολωμού 29 Αθήνα ☎ 210.38.22.157 ☎ 210.38.22.495
On-line Φροντιστήριο www.arnos.gr info@arnos.gr

Παρατήρηση 2η: Εδώ έχουμε $n = 15$ κορυφές, όπου $n = 2^4 - 1$, όπου το 4 είναι το πλήθος των επιπέδων.

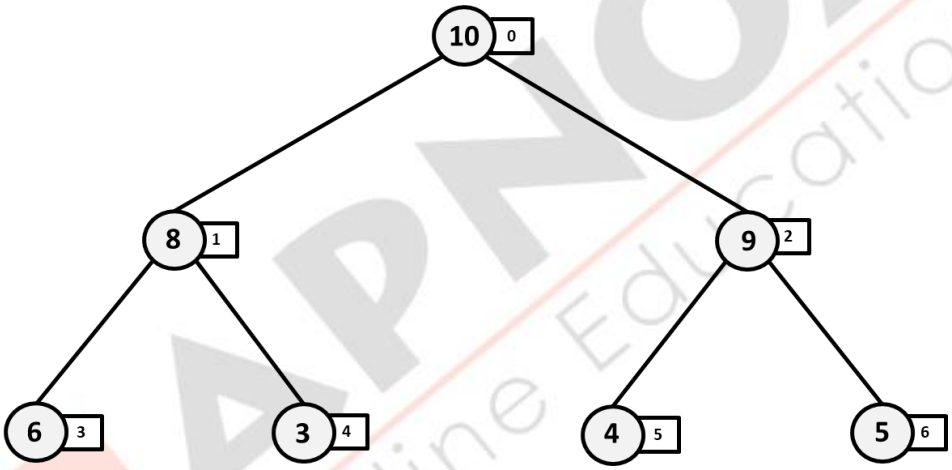
Παρατήρηση 3: Πως μπορούμε να καταλάβουμε αν μια κορυφή είναι φύλλο; Πρώτον τα φύλλα ξεκινάνε από την θέση $\left\lfloor \frac{n}{2} \right\rfloor = \left\lfloor \frac{15}{2} \right\rfloor = \lfloor 7.5 \rfloor = 7$

→Άρα μια κορυφή στην θέση i είναι φύλλο αν και μόνο αν $i \geq \left\lfloor \frac{n}{2} \right\rfloor$

Θα χρειαστεί όμως αυτό; Εύκολα μπορεί κανείς να δει πως στην αναδρομή, ο έλεγχος για τα φύλλα θα είναι λίγο διαφορετικός .

Παράδειγμα δέντρου-σωρού μεγίστων

Παρατηρήστε πως κάθε κορυφή είναι μεγαλύτερη από τα παιδιά της.



Περιγραφή του αλγορίθμου: για δοθέν σύνολο S με $|S| = n = 2^k - 1$ κάνε τα εξής:

Βρες τον **max**, βάλε τον στην θέση της ρίζας

Έχουν μείνει $2^k - 2$ *κορυφές.*

Κόψε το διάνυσμα στην μέση.

Τρέξε το ίδιο στο **πρώτο μισό,** δηλαδή στις πρώτες $\frac{2^k-2}{2} = 2^{k-1} - 1$ κορυφές

Τρέξε το ίδιο στο **δεύτερο μισό**, δηλαδή στις πρώτες $\frac{2^k-2}{2} = 2^{k-1} - 1$ κορυφές

Συνέχισε μέχρι να βρεις φύλλο

ΤΡΕΞΙΜΟ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ ΕΔΩ

(προβάλετέ το σαν slide show – αν κάνετε scroll down θα «χάσετε» το τι γίνεται από το ένα βήμα στο άλλο)

```
procedure construct_heap (S[0,1…,n-1],i)
input: Σύνολο S και δείκτης i
output: Σωρός μεγίστων A
n = |S| #πλήθος κορυφών του S
if n = = 1 #αν η κορυφή i είναι φύλλο
      A[i] = s[0]
else
      max.S = Βρες το max του S
      S'= S-{max.S}
      Θέσε A[i] = max.S
      construct_heap (S'[0,…,floor(n/2)-1], 2i+1)
      construct_heap (S'[floor(n/2),…,n-1] , 2i+2)

construct_heap (S[0,1…,n-1],0)
```

ΧΡΟΝΙΚΗ ΠΟΛΥΠΛΟΚΟΤΗΤΑ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ

ii) Ο χρόνος του αλγορίθμου προκύπτει από την αναδρομική σχέση $\boldsymbol{T(n) = 2T\left(\frac{n}{2}\right) + n}$

Σε κάθε αναδρομική κλήση διατρέχουμε $O(n)$ στοιχεία για να βρούμε το max και έπειτα επιλύουμε 2 υποπροβλήματα μεγέθους $\frac{n}{2}$

Είναι $\log_2 2 = 1$ και $f(n) = n$ άρα αφού $f(n) = \Theta(n^1)$ από την 2η περίπτωση του θεωρήματος κυριαρχίας είναι $\boldsymbol{T(n) = \Theta(nlogn)}$

ΕΡΩΤΗΜΑ 3 Β

(Β) Τροποποιήστε τον αλγόριθμό σας για το υποερώτημα Α(i) ώστε να περιγράψετε αλγόριθμο διαίρει-και-βασίλευε με πολυπλοκότητα χρόνου $O(n \log n)$ για την κατασκευή στον πίνακα $A[0..n-1]$ δέντρου-σωρού μεγίστων με όλα τα στοιχεία του συνόλου S για *οποιαδήποτε* τιμή του $|S| = n \geq 1$ *χωρίς να χρησιμοποιήσετε ταξινόμηση των στοιχείων οποιουδήποτε υποσυνόλου του συνόλου* S.

Η βασική ιδέα είναι η εξής:

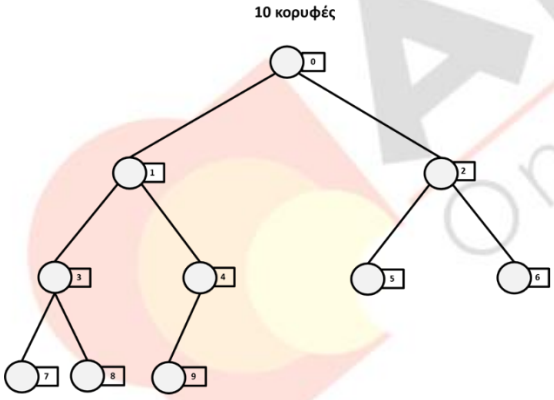
- θα τοποθετήσουμε κάποιες «κορυφές φάντασμα» με τιμή ίση με το **`min.S-1`**
- Θα σορτάρουμε τα φύλλα ώστε οι κορυφές αυτές να βρεθούν όσο δεξιότερα γίνεται στο σωρό
- Θα αφαιρέσουμε τις κορυφές φάντασμα

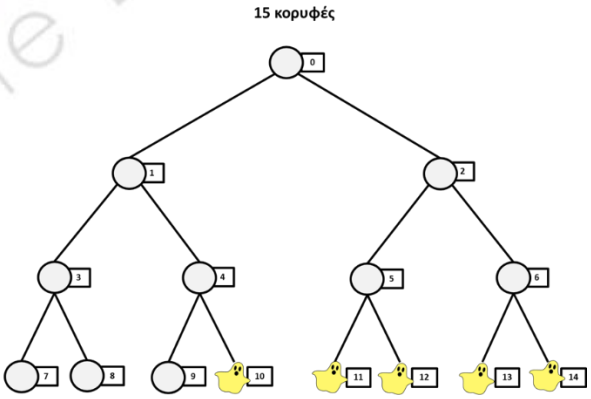
Για δοθέν n έστω ότι λείπουν φ κορυφές φάντασμα.

Βρίσκουμε την κοντινότερη δύναμη του 2 που είναι κοντά στο n

Λύνουμε την εξίσωση $2^k - 1 = n$ ως εξής άρα $k = \log(n+1)$

ΠΑΡΑΔΕΙΓΜΑ για $\boldsymbol{n = 10}$ έχουμε $k = \lceil \log 11 \rceil = \lceil 3.45 \rceil = \mathbf{4}$





Θέλουμε να φτάσουμε στο 15

Στρογγυλοποιούμε προς τα πάνω το $\log 11 = 3.45$ και γίνεται **4**

Άρα μας λείπουν $2^{\mathbf{4}} - 1 - \mathbf{10} = \mathbf{16 - 1 - 10 = 5}$ κορυφές

ΑΡΝΟΣ Online Education
Σολωμού 29 Αθήνα ☎ 210.38.22.157 ☎ 210.38.22.495
On-line Φροντιστήριο www.arnos.gr info@arnos.gr

Input S

Output: Σωρός μεγίστων A

n = |S|

log = log(n+1))

k = ceil(log(n+1))

if k==log *#το n είναι στην μορφή 2^k-1*

    construct_heap (S[0,1…,n-1],0) *# όπως πριν*

else:

    min.S = το ελάχιστο στοιχείο του S

    Βάλε **ΔΙΑΣΠΑΡΤΑ ΜΕΣΑ** στο S, 2^k-1-n κορυφές με τιμή min.S – 1

    construct_heap (S[0,1…,n-1],0)

    Αφαίρεσε τις κορυφές φάντασμα

    Τρέξε μια heapify από το n/2 μέχρι το 0 για να επισκευάσεις τον σωρό.

ΑΡΝΟΣ Online Education
Σολωμού 29 Αθήνα ☎ 210.38.22.157 ☎ 210.38.22.495
On-line Φροντιστήριο www.arnos.gr info@arnos.gr

```
import math
#############################################################################
def log(x):
    return math.log(x,2)
#############################################################################
def construct_heap(S,i):
    n = len(S)
    if n == 1:
        A[i] = S[0]
    else:
        maxS = max(S)
        S.remove(maxS)
        Snew = S[:]
        A[i] = maxS
        construct_heap(Snew[:n//2], 2 * i + 1)
        construct_heap(Snew[n//2:], 2 * i + 2)
#############################################################################

test = [2,3,4,10,7,8,9,12,13]
num = len(test)
height = log(num+1)
k = int(height)+1 #stroggilopoio pros ta pano

if k == height:
    A = [0 for i in range(num)]
    construct_heap(test, 0)
else:
    while len(test)<2**k-1:
        test.append(0)
    A = [0 for i in range(len(test))]
    construct_heap(test, 0)


print(A)

while 0 in A:
    A.remove(0)
print("maybe not cool" + str(A))

def heapify(arr, n, i):
    largest = i #Initialize max as root
    l = 2 * i + 1
    r = 2 * i + 2
    if l < n and arr[i] < arr[l]:
        largest = l
    if r < n and arr[largest] < arr[r]:
        largest = r
    if largest != i:
        arr[i], arr[largest] = arr[largest], arr[i]
        heapify(arr, n, largest)

n = len(A)

for i in range(n // 2, -1, -1):
    heapify(A, len(A), i)

print("probably cool " + str(A))
```


ΑΡΝΟΣ Online Education
Σολωμού 29 Αθήνα ☎ 210.38.22.157 ☎ 210.38.22.495
On-line Φροντιστήριο www.arnos.gr info@arnos.gr

Περί χρόνου έχουμε:

- $O(n)$ χρόνο για την εύρεση του min
- $O(n \log n)$ χρόνο για την αναδρομή
- $O(n \log n)$ χρόνο για το σορτάρισμα των φύλλων και την διαγραφή τους
- $O(\log n)$ για την heapify.

Άρα ο αλγόριθμος είναι $O(n \log n)$

ΑΡΝΟΣ Online Education
Σολωμού 29 Αθήνα ☎ 210.38.22.157 ☎ 210.38.22.495
On-line Φροντιστήριο www.arnos.gr info@arnos.gr

ΕΡΩΤΗΜΑ 4 – ΠΡΟΑΠΑΙΤΟΥΜΕΝΑ

1. Δείτε τα σχετικά βίντεο από το arnos.gr για τον δυναμικό προγραμματισμό. Για την λύση του ερωτήματος και για να μπείτε στο κλίμα του δυναμικού προγραμματισμού, συστήνεται να δώσετε βάση στο βίντεο για την ακολουθία Fibonacci καθώς και στις 3 Ασκήσεις.

2. Δείτε το θέμα δυναμικού από την εξέταση του 16-17 για να καταλάβετε την αναγκαιότητα ύπαρξης του p(j) →**VIDEO**: https://1drv.ms/v/s!Ass38lMvLxD3g4tViRC6acA0Oo_W0Q?e=dV7DJH

3. Δείτε το θέμα δυναμικού από την εξέταση του 18-19 όπου τέθηκε **ακριβώς** το ζήτημα που πραγματεύεται η παράγραφος 6.1 του βιβλίου των Kleinberg Tardos→**VIDEO:**
https://1drv.ms/v/s!Ass38lMvLxD3g4tXbjZNAPB1Ie1w0Q?e=F6lTYJ
Το σχετικό **pdf** βρίσκεται εδώ https://1drv.ms/b/s!Ass38lMvLxD3g4tZdBVITRMkEF4QSQ?e=mHR3fD

4. Διαβάστε την ενότητα 6.1 του βιβλίου

5. Αρχείο με λυμένες ασκήσεις (θέματα) δυναμικού θα βρείτε εδώ
https://1drv.ms/b/s!Ass38lMvLxD3g4tYfnZlTHZSoxRa1w?e=knyemd

ΕΡΩΤΗΜΑ 4 Α

(Α) Μελετήστε την λύση του προβλήματος του «*σταθμισμένου χρονοπρογραμματισμού διαστημάτων*» στο Κεφ. 6.1 του βιβλίου των Kleinberg & Tardos, και βρείτε τον χρόνο υπολογισμού στην χειρότερη περίπτωση της τιμής του κάθε p(j) με σειριακή αναζήτηση. Υπενθυμίζεται ότι για ένα διάστημα j, η τιμή του p(j) είναι ο μεγαλύτερος δείκτης i < j έτσι ώστε τα διαστήματα i και j να είναι ξένα (δείτε και το βιβλίο).

Με δεδομένο ότι τα αιτήματα είναι εξ'αρχής ταξινομημένα κατά αύξουσα σειρά ως προς τον χρόνο λήξης(όπως ακριβώς δίνεται στο βιβλίο), αν εκτελέσουμε σειριακή αναζήτηση ο χρόνος της **χειρότερης** περίπτωσης για το $p(j)$ είναι $\boldsymbol{\Theta(n^2)}$

Υπολογισμός του $p(j)$ - ΣΕΙΡΙΑΚΗ ΑΝΑΖΗΤΗΣΗ

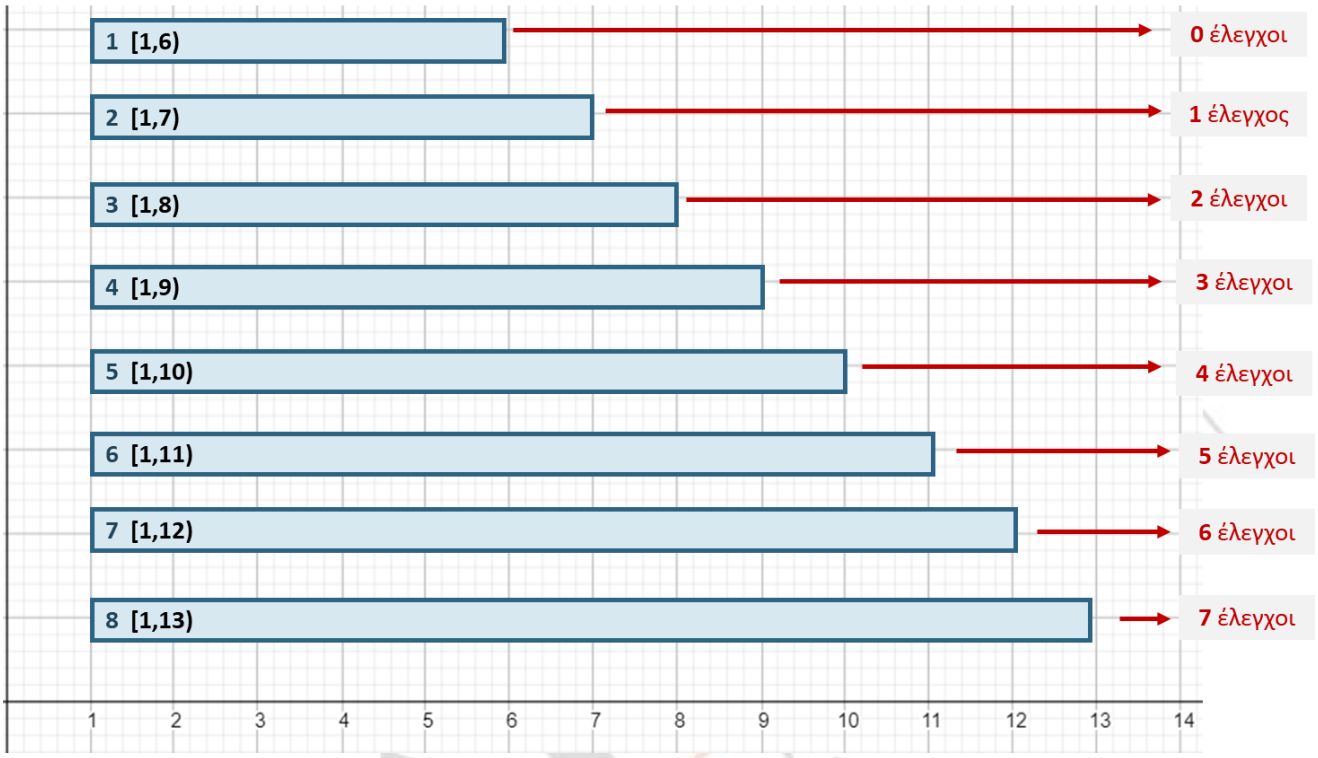
```
if j==1
     return p(1) = 0

else
     for j = 2 to n
          for i=j-1 to 1
               if f(i)<=s(j)
                    return p(j) = i
                    break
               else
                    return p(j) = 0
```

$$1+2+3+\cdots+(n-1)=\frac{(1+n-1)(n-1)}{2}=\boldsymbol{\Theta(n^2)}$$

ΑΡΝΟΣ Online Education
Σολωμού 29 Αθήνα ☎ 210.38.22.157 ☎ 210.38.22.495
On-line Φροντιστήριο www.arnos.gr info@arnos.gr

Τα αιτήματα κατά αύξουσα σειρά των χρόνων λήξης – Η χειρότερη περίπτωση

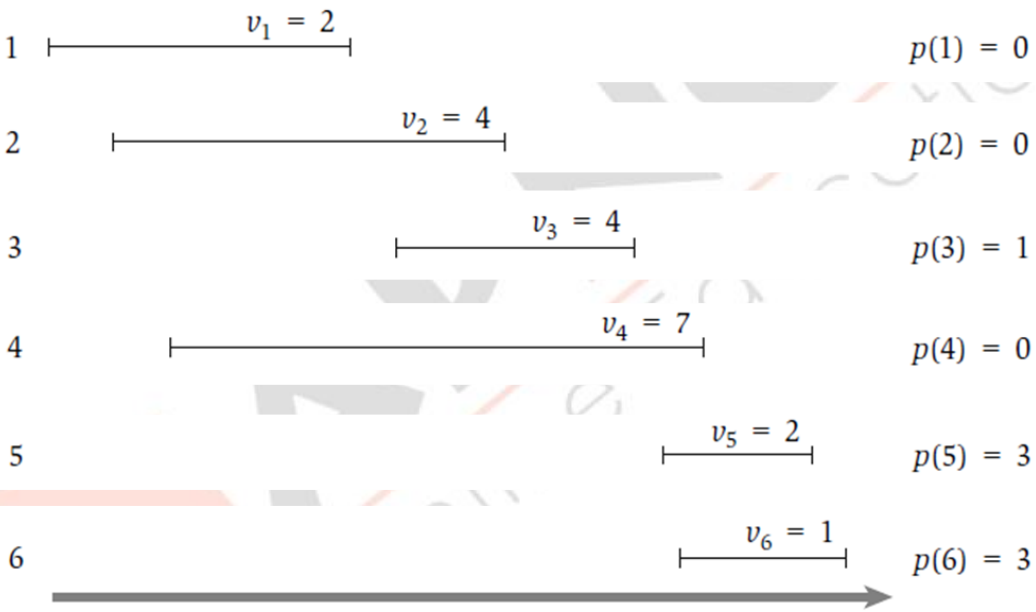


ΑΡΝΟΣ Online Education
Σολωμού 29 Αθήνα ☎ 210.38.22.157 ☎ 210.38.22.495
On-line Φροντιστήριο www.arnos.gr info@arnos.gr

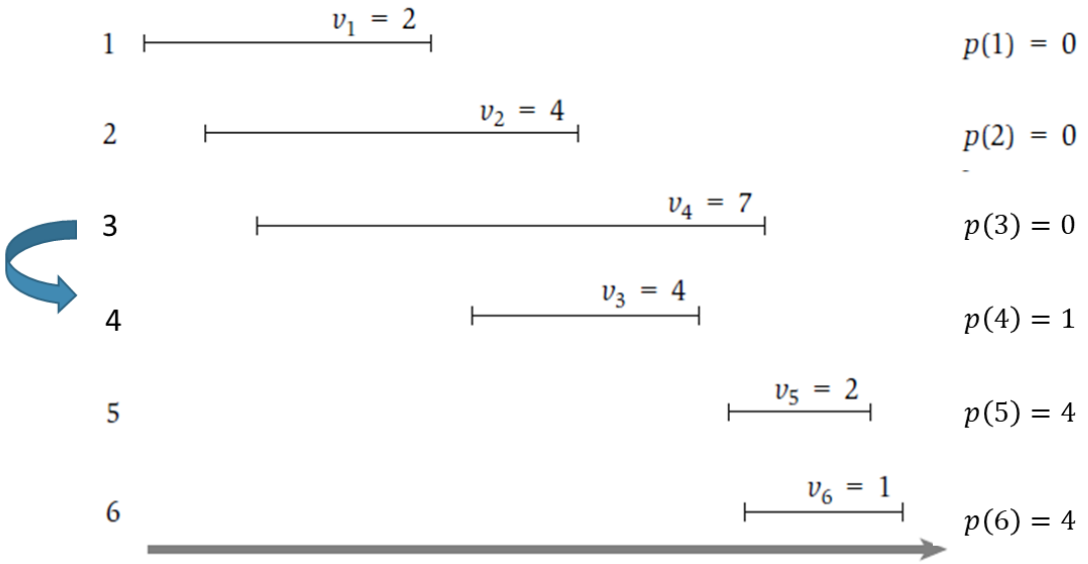
ΕΡΩΤΗΜΑ 4 Β

(Α) Μελετήστε την λύση του προβλήματος του «*σταθμισμένου χρονοπρογραμματισμού διαστημάτων*» στο Κεφ. 6.1 του βιβλίου των Kleinberg & Tardos, και βρείτε τον χρόνο υπολογισμού στην χειρότερη περίπτωση της τιμής του κάθε p(j) με σειριακή αναζήτηση. Υπενθυμίζεται ότι για ένα διάστημα j, η τιμή του p(j) είναι ο μεγαλύτερος δείκτης i < j έτσι ώστε τα διαστήματα i και j να είναι ξένα (δείτε και το βιβλίο).

Τα αιτήματα κατά αύξουσα σειρά των χρόνων λήξης – Η p(j) ΔΕΝ είναι αύξουσα

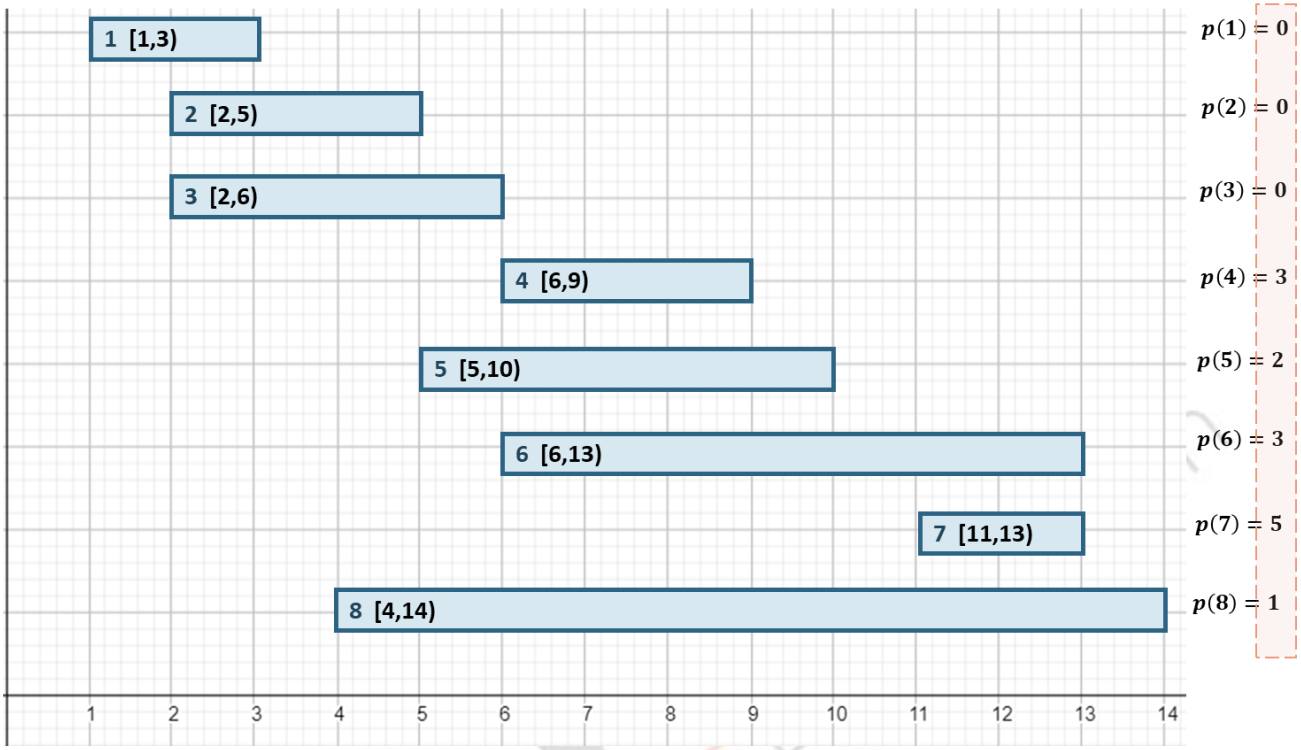


Τα αιτήματα κατά αύξουσα σειρά των χρόνων έναρξης – Η p(j) είναι αύξουσα



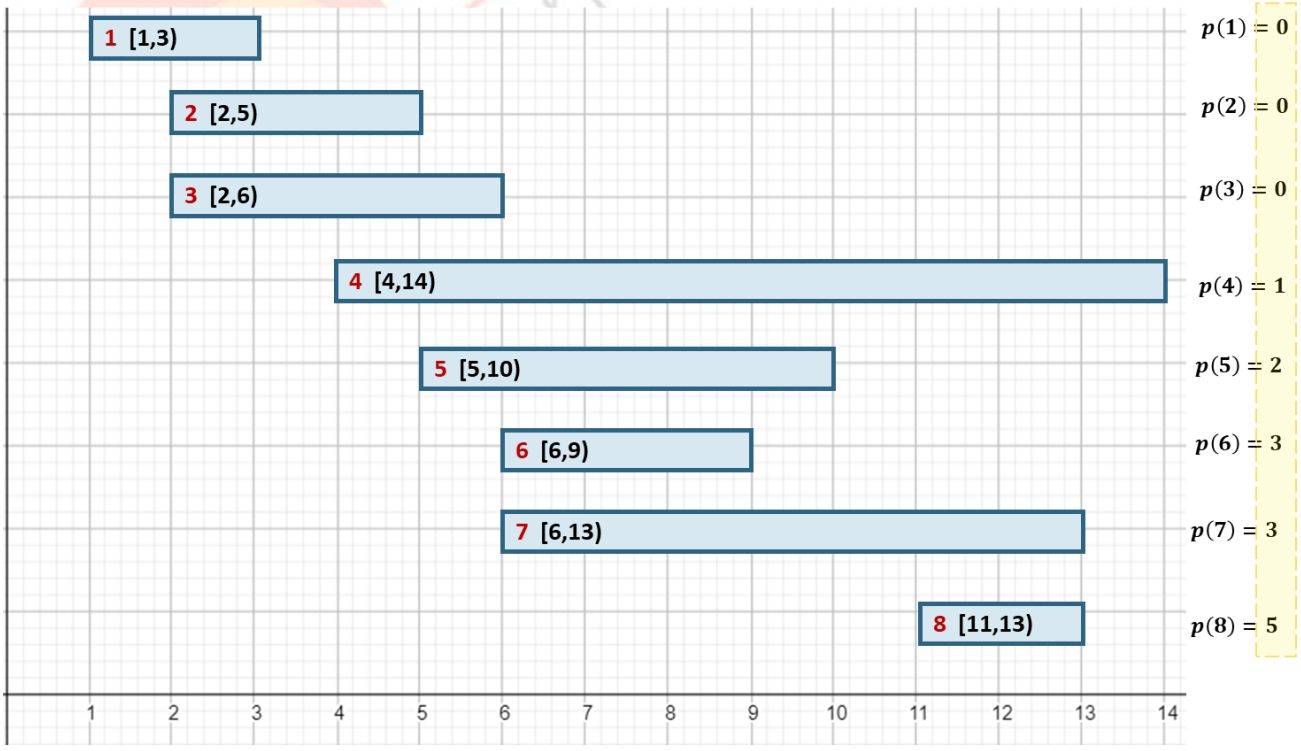
ΑΡΝΟΣ Online Education
Σολωμού 29 Αθήνα ☎ 210.38.22.157 ☎ 210.38.22.495
On-line Φροντιστήριο www.arnos.gr info@arnos.gr

Τα αιτήματα κατά αύξουσα σειρά των χρόνων λήξης – Η p(j) ΔΕΝ είναι αύξουσα



Παράδειγμα 1: για να πειστούμε

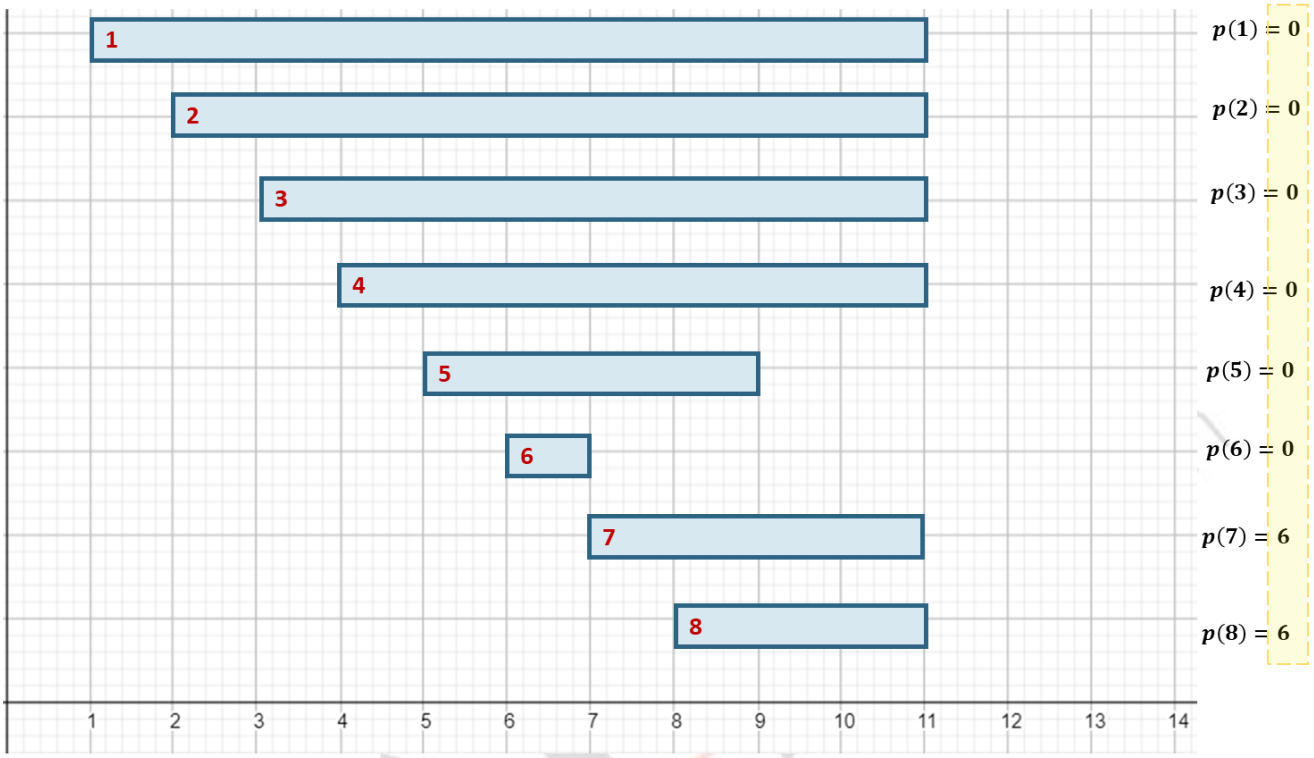
Τα αιτήματα κατά αύξουσα σειρά των χρόνων έναρξης – Η p(j) είναι αύξουσα



Παράδειγμα 1: για να πειστούμε

ΑΡΝΟΣ Online Education
Σολωμού 29 Αθήνα ☎ 210.38.22.157 ☎ 210.38.22.495
On-line Φροντιστήριο www.arnos.gr info@arnos.gr

Τα αιτήματα κατά αύξουσα σειρά των χρόνων έναρξης – Η p(j) είναι αύξουσα



Παράδειγμα 2: για να πειστούμε

ΑΡΝΟΣ Online Education
Σολωμού 29 Αθήνα ☎ 210.38.22.157 ☎ 210.38.22.495
On-line Φροντιστήριο www.arnos.gr info@arnos.gr

ΕΝΔΕΙΚΤΙΚΗ ΑΠΑΝΤΗΣΗ

Ταξινομούμε τα αιτήματα σε αύξουσα σειρά, με βάση τους **χρόνους έναρξης.**

Δηλαδή θεωρούμε το διάνυσμα $\mathrm{A} = [1,2, \dots, \mathrm{n}]$

Με την merge sort ταξινομούμε σε χρόνο $\boldsymbol{\theta(nlogn)}$ τα αιτήματα ως προς τους χρόνους έναρξης s_i

Θεωρούμε διάνυσμα $B = [\beta(1), \beta(2), \dots, \beta(n)]$ το οποίο έχει προέλθει από την ταξινόμηση του A

Θα **αποδείξουμε** ότι η ακολουθία τιμών $p(\beta(1)), p(\beta(2)), \dots, p(\beta(n))$ είναι αύξουσα. **ΔΕΝ** ΤΟ ΖΗΤΑΕΙ Η ΕΚΦΩΝΗΣΗ – ΤΗΝ ΑΠΟΔΕΙΞΗ ΤΗΝ ΚΑΝΟΥΜΕ ΓΙΑ ΝΑ ΠΕΙΣΤΟΥΜΕ

Για λόγους απλότητας θεωρούμε πως μετονομάζουμε τα αιτήματά μας ξανά σε 1,2,..n ώστε $\beta(i) = i$

Έστω ότι μέχρι το αίτημα $k \geq 1$ τα πράγματα έχουν πάει καλά.

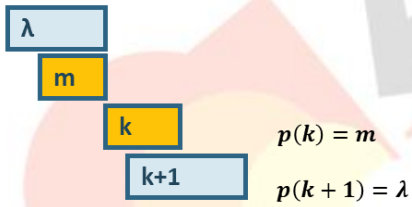
Έστω δηλαδή ότι για κάθε αίτημα μέχρι το k ισχύει πως αν $i \leq j$ τότε $p(i) \leq p(j)$

Θεωρούμε το αίτημα $k+1$. Θα αποδείξουμε ότι $p(k) \leq p(k+1)$

Υπάρχει περίπτωση το $p(k+1)$ να είναι μικρότερο του $p(k)$;

Έστω $\boldsymbol{p(k) = m}$ και $\boldsymbol{p(k+1) = \lambda}$ **ΑΛΛΑ** $\boldsymbol{m > \lambda}$

Έστω $\boldsymbol{\lambda < m}$



Φαίνεται καθαρά στην εικόνα, πως αφού εξ'ορισμού παίρνουμε τον **μεγαλύτερο δείκτη** $\boldsymbol{i}$ για τον οποίο ισχύει ότι το $p(j)$ είναι συμβατό με το i, αφού το $k+1$ ξεκινάει (είτε την ίδια ώρα είτε) μετά το k, τότε ο συμβατό αίτημα μεγαλύτερου δείκτη είναι το m και όχι το λ.
Δηλαδή στην διπλανή εικόνα έχουμε ότι $\boldsymbol{p(k+1) = m}$ **και όχι** $\boldsymbol{\lambda}$

Σχόλιο: τις ισοπαλίες τις λύνουμε κοιτώντας τον χρόνο λήξης. Δηλαδή αν δύο αιτήματα έχουν τον ίδιο χρόνο έναρξης, πρώτο ανάμεσα στα δύο θα μπει εκείνο που έχει μικρότερο χρόνο λήξης.

ΕΡΩΤΗΜΑ 4 Γ

(Γ) Να σχεδιάσετε έναν αλγόριθμο ο οποίος με είσοδο n διαστήματα διατεταγμένα σε αύξουσα τάξη ως προς το χρόνο λήξης, αρχικά θα βρίσκει τη διάταξη της απάντησης του υποερωτήματος (Β) και στη συνέχεια θα υπολογίζει κατά σειρά τις τιμές p(β(1)), p(β(2)), … p(β(n)) χρησιμοποιώντας Δυναμικό Προγραμματισμό. Ο αλγόριθμος θα πρέπει να έχει πολυπλοκότητα ασυμπτωτικά μικρότερη από αυτή που βρήκατε για τον αλγόριθμο σειριακής αναζήτησης στο υποερώτημα (Α). *Υπόδειξη:* Βρείτε μια αναδρομική εξίσωση η οποία να εκφράζει το p(β(j)) ως συνάρτηση ενός κατάλληλου πλήθους τιμών p(β(i)), για i < j.

****Αναδρομική:**

$p(1) = 0$

$p(j) = \max\{p(j-1) + i | i \in \{0, .., j-1-p(j-1)\}, με\ f(i) \leq s(j)\}$

```
p(1)=0
for j=2 to n
     for i=0 to j-1-p(j-1)
         if f(i)>s(j)
             p(j)= p(j-1)+ i-1
             break
```

Θέλουμε το i να τελειώνει πριν αρχίσει το j
Θέλουμε $f(i) \leq s(j)$
Αν βρεις προηγούμενο αίτημα με $f(i) > s(j)$ πάει να πει πως μόλις το περάσαμε. Άρα είναι το αμέσως προηγούμενο αίτημα.

ΕΡΩΤΗΜΑ 4 Γ

(Δ) Υπολογίστε την ασυμπτωτική χρονική πολυπλοκότητα του αλγορίθμου στο (Γ).