Λύσεις Τελικής Εξέτασης

1.

Έχουμε ένα υπολογιστικό αρχείο και n διακομιστές $S_1, \ldots, S_n$. Θέλουμε να αποφασίσουμε ποιοι διακομιστές θα έχουν τελικά αντίγραφο του αρχείου. Οι αποφάσεις μας πρέπει να λάβουν υπόψη τα κόστη τοποθέτησης και τα κόστη προσπέλασης.

- Η τοποθέτηση ενός αντιγράφου του αρχείου στον διακομιστή S_i, όπου $1 \leq i \leq n$, έχει ένα (ακέραιο) κόστος τοποθέτησης $c_i > 0$.
- Το κόστος προσπέλασης στον διακομιστή S_i ισούται με 0 αν ο διακομιστής S_i έχει αντίγραφο του αρχείου, ή με $j - i$ αν ο διακομιστής S_i δεν έχει αντίγραφο του αρχείου και $j > i$ είναι ο ελάχιστος δείκτης για τον οποίο ο διακομιστής S_j έχει αντίγραφο του αρχείου.

  Για να ορίζεται το κόστος προσπέλασης σε οποιονδήποτε διακομιστή, υποθέτουμε ότι ο διακομιστής S_n θα έχει οπωσδήποτε ένα αντίγραφο του αρχείου.

Μία διευθέτηση είναι μία συλλογή αποφάσεων για κάθε διακομιστή S_i, όπου $1 \leq i \leq n$, αν θα βάλουμε ή όχι αντίγραφο του αρχείου στον διακομιστή S_i. Το συνολικό κόστος μίας διευθέτησης ισούται με το άθροισμα του κόστους τοποθέτησης για όλους τους διακομιστές που έχουν αντίγραφο του αρχείου, και του κόστους προσπέλασης για όλους τους n διακομιστές. Το υπολογιστικό πρόβλημα που έχουμε να επιλύσουμε είναι να βρούμε μία διευθέτηση με το ελάχιστο δυνατό συνολικό κόστος. Μια τέτοια διευθέτηση θα λέγεται βέλτιστη λύση, και η τιμή της θα είναι το συνολικό κόστος της διευθέτησης. Θα επιχειρήσουμε να σχεδιάσουμε και να αναλύσουμε ένα αλγόριθμο δυναμικού προγραμματισμού για το υπολογιστικό μας πρόβλημα.

(α) Προσδιορίστε προσεκτικά τα υποπροβλήματα που θα χρησιμοποιήσετε.

Λύση: Για κάθε j, όπου $1 \leq j \leq n$, ορίζουμε ως υποπρόβλημα $\Pi(j)$ το εξής υπολογιστικό πρόβλημα:

> Προσδιόρισε μία διευθέτηση με ελάχιστο δυνατό συνολικό κόστος για τους διακομιστές $S_1, \ldots, S_j$, στην οποία ο διακομιστής S_j έχει οπωσδήποτε ένα αντίγραφο του αρχείου.

Προφανώς, υπάρχουν n τέτοια υποπροβλήματα και το υπολογιστικό πρόβλημα που έχουμε να επιλύσουμε είναι το υποπρόβλημα $\Pi(n)$.

(β) Διατυπώστε και αποδείξτε (απλά) μία ιδιότητα που χαρακτηρίζει τη δομή της βέλτιστης λύσης.

Λύση: Έστω μία βέλτιστη λύση $d_1, \ldots, d_j$ για το υποπρόβλημα $\Pi(j)$, όπου $1 \leq j \leq n$. στην οποία για κάθε δείκτη r με $1 \leq r \leq j$, $d_r = 1$ αν βάζουμε αντίγραφο του αρχείου στον διακομιστή S_r, και $d_r = 0$ αν δεν βάζουμε αντίγραφο του αρχείου στον διακομιστή S_r. Τότε, η ακολουθία $d_1, \ldots, d_{j-1}$ είναι μία βέλτιστη λύση για το υποπρόβλημα $\Pi(j-1)$.

Πράγματι, ας υποθέσουμε ότι υπήρχε καλύτερη λύση $d'_1, \ldots, d'_{j-1}$ για το υποπρόβλημα $\Pi(j-1)$. Τότε, η ακολουθία $d'_1, \ldots, d'_{j-1}, d_j$ θα ήταν μία καλύτερη λύση για το υποπρόβλημα $\Pi(j)$ (από την βέλτιστη λύση $d_1, \ldots, d_j$). Αντίφαση.

(γ) Διατυπώστε μία αναδρομική εξίσωση για την τιμή της βέλτιστης λύσης.

Λύση: Έστω $\mathsf{V}[j]$ η τιμή της βέλτιστης λύσης $d_1, \ldots, d_j$ για το υποπρόβλημα $\Pi(j)$. Σε μία λύση για το υποπρόβλημα $\Pi(j)$. έστω $i < j$ ο μέγιστος δείκτης για τον οποίο βάζουμε αντίγραφο του αρχείου στον διακομιστή S_i. Τότε, για τους υπόλοιπους διακομιστές $S_{i+1}, \ldots, S_j$, θα έχουμε μόνο κόστος προσπέλασης, και το συνολικό κόστος (προσπέλασης) για αυτούς τους διακομιστές θα είναι $(j-(i+1))+(j-(i+2))+\ldots+0 = \binom{j-i}{2}$. Από τον χαρακτηρισμό της δομής της βέλτιστης λύσης, έχουμε ότι

$$\mathsf{V}[j] \;=\; c_j + \min_{1 \le i < j} \left(\mathsf{V}[i] + \binom{j-i}{2} \right) .$$

(δ) Παρουσιάστε ένα αλγόριθμο δυναμικού προγραμματισμού για τον υπολογισμό της τιμής της βέλτιστης λύσης. Ο αλγόριθμός σας θα συμπληρώνει κατάλληλα τις θέσεις ενός καταλλήλου πίνακα.

Λύση: Για κάθε δείκτη j, κατά αύξουσα τάξη, όπου $1 \le j \le n$, χρησιμοποιούμε την τιμή της βέλτιστης λύσης για να υπολογίσουμε την τιμή $\mathsf{V}[j]$. Καταχωρούμε κάθε τιμή $\mathsf{V}[j]$ που υπολογίζουμε στην αντίστοιχη θέση j ενός πίνακα $\{1, \ldots, n\}$. Η τιμή $\mathsf{V}[n]$ είναι η τιμή της βέλτιστης λύσης.

(ε) Επεκτείνετε τον αλγόριθμό σας ώστε αυτός να επιστρέφει και την ίδια τη βέλτιστη λύση (μαζί με την τιμή της).

Λύση: Στον πίνακα $\{1, \ldots, n\}$, καταχωρούμε επίσης για κάθε δείκτη j, όπου $1 \le j \le n$, τον δείκτη $i = i(j) < j$ για τον οποίο ισχύει ότι

$$\mathsf{V}[j] \;=\; c_j + \mathsf{V}[i] + \binom{j-i}{2} .$$

Τότε, η βέλτιστη λύση ανακατασκευάζεται αναδρομικά από τον πίνακα $\{1, \ldots, n\}$ ως εξής:

- Τοποθέτησε αντίγραφο του αρχείου στον διακομιστή n.
- Χρησιμοποίησε τον υποπίνακα $\{1, \ldots, i(n)\}$ για να ανακατασκευάσεις αναδρομικά την βέλτιστη λύση για το υποπρόβλημα $\Pi(i(n))$.

(στ) Ποιά είναι η πολυπλοκότητα του αλγορίθμου σας (ως συνάρτηση του n);

Λύση: Χρειάζονται $O(n)$ βήματα για τον υπολογισμό οποιασδήποτε τιμής $\mathsf{V}[j]$, όπου $1 \le j \le n$. Έτσι, χρειάζονται συνολικά $O(n^2)$ βήματα για τη συμπλήρωση του πίνακα με τις λύσεις στα υποπροβλήματα. Η (αναδρομική) ανακατασκευή της βέλτιστης λύσης από τον ίδιο πίνακα απαιτεί επιπρόσθετα $O(n)$ βήματα. Έτσι, ο συνολικός αριθμός βημάτων είναι $O(n^2)$.

2.

Σε μία φοιτητική εστία, διαμένουν n φοιτητές $s_1, \ldots, s_n$. Για καθένα από τα επόμενα n βράδυα, κάποιος από τους φοιτητές θα μαγειρέψει δείπνο για όλους τους n φοιτητές έτσι ώστε να ικανοποιούνται οι εξής περιορισμοί:

- Σε κάθε βράδυ i, όπου $1 \leq i \leq n$, ακριβώς ένας φοιτητής μαγειρεύει.
- Κάθε φοιτητής s_i, όπου $1 \leq i \leq n$, μαγειρεύει ακριβώς μία φορά.
- Κάθε φοιτητής s_i, όπου $1 \leq i \leq n$, μαγειρεύει σε κάποιο βράδυ $j \in D_i$, όπου $D_i \subseteq \{1, 2, \ldots, n\}$ είναι το σύνολο από τα βράδυα στα οποία ο φοιτητής s_i μπορεί να μαγειρέψει.

Ένα εφικτό πρόγραμμα δείπνων είναι μία αντιστοίχιση μεταξύ των n φοιτητών και των n βραδυών έτσι ώστε να ικανοποιούνται οι τρεις παραπάνω περιορισμοί.

(α) Κατασκευάστε κατάλληλα ένα διμερή γράφο G έτσι ώστε ο G να έχει ένα τέλειο ταίριασμα αν και μόνο αν υπάρχει εφικτό πρόγραμμα δείπνων.

Λύση: Από τα δεδομένα του προβλήματος, κατασκευάζουμε τον διμερή γράφο G ως εξής:

- Τα δύο σύνολα κορυφών του είναι X $= \{s_1, \ldots, s_n\}$ (οι φοιτητές) και Y $= \{1, \ldots, n\}$ (τα βράδυα).
- Για κάθε ζεύγος s_i και j, όπου $1 \leq i, j \leq n$, έχουμε μία ακμή μεταξύ των κορυφών s_i και j αν και μόνο αν $j \in D_i$.

(β) Για τον διμερή γράφο G που έχετε κατασκευάσει, αποδείξτε ότι ο G έχει ένα τέλειο ταίριασμα αν και μόνο αν υπάρχει εφικτό πρόγραμμα δείπνων.

Λύση: Υποθέτουμε καταρχήν ότι ο διμερής γράφος G έχει ένα τέλειο ταίριασμα. Έτσι, για κάθε κορυφή s_i, όπου $1 \leq i \leq n$, $j(i)$ είναι η κορυφή με την οποία αυτή ταιριάζεται. Θα κατασκευάσουμε ένα εφικτό πρόγραμμα δείπνων:

> Ο φοιτητής s_i θα μαγειρέψει το βράδυ $j(i)$.

Από τον ορισμό του τελείου ταιριάσματος, οι πρώτοι δύο περιορισμοί στον ορισμό ενός εφικτού προγράμματος δείπνων ικανοποιούνται. Ο τρίτος περιορισμός ικανοποιείται από την κατασκευή του συνόλου των ακμών στο διμερή γράφο G.

Υποθέτουμε τώρα ότι υπάρχει ένα εφικτό πρόγραμμα δείπνων. Κατασκευάζουμε ένα τέλειο ταίριασμα στον διμερή γράφο G:

> Αν ο φοιτητής s_i μαγειρεύει το βράδυ j, όπου $1 \leq i, j \leq n$, τότε επιλέγουμε το ζεύγος (s_i, j).

Από τον τρίτο περιορισμό στον ορισμό ενός εφικτού προγράμματος δείπνων, έπεται ότι κάθε ζεύγος (s_i, j) που επιλέγουμε, όπου $1 \leq i, j \leq n$, είναι ακμή. Οι δύο πρώτοι περιορισμοί στον ορισμό ενός εφικτού προγράμματος δείπνων συνεπάγονται ότι το σύνολο των ακμών που έχουμε επιλέξει είναι ένα τέλειο ταίριασμα.

(γ) Παρουσιάστε πολυωνυμικό αλγόριθμο ο οποίος αποφασίζει αν υπάρχει εφικτό πρόγραμμα δείπνων.

Λύση: Το ερώτημα (β) συνεπάγεται τον εξής αλγόριθμο:

- Από τα δεδομένα του προβλήματος, κατασκευάζουμε τον διμερή γράφο G (όπως στο ερώτημα (α)).
- Έλέγχουμε αν ο διμερής γράφος G έχει ένα τέλειο ταίριασμα.
  - Ο έλεγχος γίνεται με αναγωγή στο πρόβλημα μέγιστης ροής. (Δηλαδή, από τον διμερή γράφο G, κατασκευάζουμε ένα δίκτυο ροής, για το οποίο υπολογίζουμε την μέγιστη ροή με τον αλγόριθμο Ford-Fulkerson. Το μέγεθος του μεγίστου ταιριάσματος λαμβάνεται ίσο με την τιμή της μέγιστης ροής που υπολογίσαμε. Αν το μέγεθος του μεγίστου ταιριάσματος ισούται με n, τότε (και μόνο τότε) αποφασίζουμε ότι ο διμερής γράφος G έχει ένα τέλειο ταίριασμα.

Το πρόβλημα αυτό εξετάζει ένα αλγόριθμο πολλαπλασιασμού δύο πολυωνύμων βαθμού n (σε παράσταση συντελεστών), των

$$\mathsf{Q}(x) \;=\; q_0 + q_1 x + \ldots + q_n x^n$$

και

$$\mathsf{S}(x) \;=\; s_0 + s_1 x + \ldots + s_n x^n \,.$$

3. Αυτός αποτελεί έναν εναλλακτικό αλγόριθμο πρός τον Ταχύ Μετασχηματισμό Fourier, που έχουμε παρουσιάσει στο μάθημα. Ο αλγόριθμος θα είναι απαραίτητα αναδρομικός, και θα στηρίζεται (στη βάση του) σε ένα αλγόριθμο πολαπλασιασμού δύο πολυωνύμων δευτέρου βαθμού

$$\mathsf{A}(x) \;=\; a_0 + a_1 x + a_2 x^2$$

και

$$\mathsf{B}(x) \;=\; b_0 + b_1 x + b_2 x^2$$

(σε παράσταση συντελεστών) με μειωμένο αριθμό βαθμωτών πολλαπλασιασμών.

(α) Το τεταρτοβάθμιο πολυώνυμο $\mathsf{C}(x) = \mathsf{A}(x) \cdot \mathsf{B}(x)$ μπορεί να υπολογισθεί (σε παράσταση συντελεστών) κατά τον "αφελή" αλγόριθμο ο οποίος στηρίζεται στον ορισμό του γινομένου δύο πολυωνύμων. Ποίος είναι ο αριθμός των βαθμωτών πολλαπλασιασμών και ο αριθμός των βαθμωτών προσθέσεων που απαιτεί ο "αφελής" αλγόριθμος;

Λύση: Από τον ορισμό του γινομένου δύο πολυωνύμων, έχουμε ότι

$$\mathsf{C}(x) \;=\; a_0 b_0 + (a_0 b_1 + a_1 b_0) x + (a_0 b_2 + a_1 b_1 + a_2 b_0) x^2 + (a_1 b_2 + a_2 b_1) x^3 + a_2 b_2 x^4 \,.$$

Έτσι, ο "αφελής" αλγόριθμος απαιτεί 9 πολλαπλασιασμούς και 4 προσθέσεις.

(β) Θεωρείστε τώρα τα έξη γινόμενα:

i. $\mathsf{P}_1 = a_0 \cdot b_0$

ii. $\mathsf{P}_2 = a_1 \cdot b_1$

iii. $\mathsf{P}_3 = a_2 \cdot b_2$

iv. $\mathsf{P}_4 = (a_0 + a_1) \cdot (b_0 + b_1)$

v. $\mathsf{P}_5 = (a_1 + a_2) \cdot (b_1 + b_2)$

vi. $\mathsf{P}_6 = (a_0 + a_2) \cdot (b_0 + b_2)$

Παρουσιάστε αλγόριθμο για τον υπολογισμό των συντελεστών του πολυωνύμου $\mathsf{C}(x)$ από τα έξι γινόμενα P_1 έως και P_6 ο οποίος χρησιμοποιεί μόνο βαθμωτές προσθέσεις/αφαιρέσεις (και καθόλου βαθμωτούς πολλαπλασιασμούς). Πόσες βαθμωτές προσθέσεις/αφαιρέσεις χρησιμοποιεί ο αλγόριθμός σας;

Λύση: Έστω

$$\mathsf{C}(x) \;=\; c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4 \,.$$

Οι συντελεστές του γινομένου $\mathsf{C}(x)$ θα υπολογιστούν από τα έξι γινόμενα P_1 έως και P_6 ως εξής:

$$\begin{aligned} c_0 &:= \mathsf{P}_1\,; \\ c_1 &:= \mathsf{P}_4 - \mathsf{P}_1 - \mathsf{P}_2\,; \\ c_2 &:= \mathsf{P}_6 - \mathsf{P}_1 + \mathsf{P}_2 - \mathsf{P}_3\,; \\ c_3 &:= \mathsf{P}_5 - \mathsf{P}_2 - \mathsf{P}_3\,; \\ c_4 &:= \mathsf{P}_3\,. \end{aligned}$$

Ο υπολογισμός των έξι γινομένων P_1 έως και P_6 απαιτεί έξι βαθμωτές προσθέσεις. Ο υπολογισμός των συντελεστών του γινομένου $\mathsf{C}(x)$ από τα έξι γινόμενα P_1 έως και P_6 απαιτεί επτά βαθμωτές προσθέσεις/αφαιρέσεις. Έτσι, ο συνολικός αριθμός των βαθμωτών προσθέσεων/αφαιρέσεων που απαιτούνται για τον υπολογισμό του γινομένου $\mathsf{C}(x)$ είναι δεκατρείς.

(γ)

Θα προχωρήσουμε τώρα στον γενικό αλγόριθμο για τον πολλαπλασιασμό των πολυωνύμων $\mathsf{Q}(x)$ και $\mathsf{S}(x)$. Θέτουμε

$$y \;=\; x^{\frac{n+1}{3}+1}\,.$$

Θα θεωρήσουμε τις παραστάσεις

$$\mathsf{Q}(x) \;=\; \mathsf{Q}_0 + \mathsf{Q}_1 y + \mathsf{Q}_2 y^2$$

και

$$\mathsf{S}(x) \;=\; \mathsf{S}_0 + \mathsf{S}_1 y + \mathsf{S}_2 y^2\,,$$

όπου

$$\begin{aligned} \mathsf{Q}_0 &= q_0 + q_1 x + \ldots + q_{\frac{n+1}{3}-1}\, x^{\frac{n+1}{3}-1}\,, \\ \mathsf{Q}_1 &= q_{\frac{n+1}{3}} + q_{\frac{n+1}{3}+1} x + \ldots + q_{2\cdot\frac{n+1}{3}-1}\, x^{\frac{n+1}{3}-1}\,, \\ \mathsf{Q}_2 &= q_{2\cdot\frac{n+1}{3}} + q_{2\frac{n+1}{3}+1} x + \ldots + q_n x^{\frac{n+1}{3}-1}\,, \end{aligned}$$

ενώ αντίστοιχα ορίζονται τα S_0, S_1 και S_2.
Χρησιμοποιείστε τις παραπάνω παραστάσεις για να διατυπώσετε έναν αναδρομικό αλγόριθμο για τον υπολογισμό του γινομένου $\mathsf{Q}(x)\cdot\mathsf{S}(x)$. Η βάση του αναδρομικού αλγορίθμου σας πρέπει να προέρχεται από την απάντησή σας στο ερώτημα (β).

Λύση: Παρουσιάζουμε τον εξής αναδρομικό αλγόριθμο:

- Χρησιμοποιούμε τις προτεινόμενες παραστάσεις για τα πολυώνυμα $\mathsf{Q}(x)$ και $\mathsf{S}(x)$ ώστε να υπολογίσουμε αναδρομικά τα έξι γινόμενα:

$$\begin{aligned}
\mathsf{P}_1 &= \mathsf{Q}_0 \cdot \mathsf{S}_0 \,; \\
\mathsf{P}_2 &= \mathsf{Q}_1 \cdot \mathsf{S}_1 \,; \\
\mathsf{P}_3 &= \mathsf{Q}_2 \cdot \mathsf{S}_2 \,; \\
\mathsf{P}_4 &= (\mathsf{Q}_0 + \mathsf{Q}_1) \cdot (\mathsf{S}_0 + \mathsf{S}_1) \,; \\
\mathsf{P}_5 &= (\mathsf{Q}_1 + \mathsf{Q}_2) \cdot (\mathsf{S}_1 + \mathsf{S}_2) \,; \\
\mathsf{P}_6 &= (\mathsf{Q}_0 + \mathsf{Q}_2) \cdot (\mathsf{S}_0 + \mathsf{S}_2) \,.
\end{aligned}$$

- Από τα έξι γινόμενα P_1 έως και P_6, υπολογίζουμε τους συντελεστές c_0 έως και c_4 ως εξής:

$$\begin{aligned}
c_0 &:= \mathsf{P}_1 \,; \\
c_1 &:= \mathsf{P}_4 - \mathsf{P}_1 - \mathsf{P}_2 \,; \\
c_2 &:= \mathsf{P}_6 - \mathsf{P}_1 + \mathsf{P}_2 - \mathsf{P}_3 \,; \\
c_3 &:= \mathsf{P}_5 - \mathsf{P}_2 - \mathsf{P}_3 \,; \\
c_4 &:= \mathsf{P}_3 \,.
\end{aligned}$$

  Επιστρέφουμε σαν έξοδο το πολυώνυμο

$$\mathsf{C}(x) = c_0 + c_1\, y + c_2\, y^2 + c_3\, y^3 + c_4\, y^4 \,,$$

  όπου $y = x^{\frac{n+1}{3}+1}$.

(δ) Γράψτε μία αναδρομική σχέση για την πολυπλοκότητα $\mathsf{T}(n)$ του αλγορίθμου σας (ως συνάρτηση του n).

Λύση: Προφανώς, καθένα από τα πολυώνυμα Q_0, Q_1, Q_2 και S_0, S_1, S_2 έχει βαθμό $\frac{n+1}{3} - 1 \leq \frac{n}{3}$. Επομένως, ο αναδρομικός υπολογισμός των γινομένων P_1 έως και P_6 περιλαμβάνει έξι αναδρομικούς πολλαπλασιασμούς πολυωνύμων βαθμού το πολύ $\frac{n}{3}$. (Προσέξτε ότι καθένα από τα γινόμενα P_1 έως και P_6 είναι ένα πολυώνυμο βαθμού $2 \cdot \left(\frac{n+1}{3} - 1\right)$.) Στη συνέχεια, ο υπολογισμός των συντελεστών c_0 έως και c_4 περιλαμβάνει επτά προσθέσεις πολυωνύμων βαθμού $2 \cdot \left(\frac{n+1}{3} - 1\right)$. Κάθε τέτοια πρόσθεση απαιτεί $2 \cdot \left(\frac{n+1}{3} - 1\right) + 1 = \Theta(n)$ βαθμωτές προσθέσεις/αφαιρέσεις. Έπεται ότι η πολυπλοκότητα $\mathsf{T}(n)$ ικανοποιεί την αναδρομική σχέση

$$\mathsf{T}(n) \leq 6\, \mathsf{T}\left(\frac{n}{3}\right) + \Theta(n) \,.$$

(ε) Χρησιμοποιείστε την αναδρομική σχέση που έχετε γράψει για να προσδιορίσετε την πολυπλοκότητα $\mathsf{T}(n)$. Πως συγκρίνεται ο αλγόριθμός σας με τον Ταχύ Μετασχηματισμό Fourier?

Λύση: Από το Θ. Γενικής Χρήσης, έχουμε ότι $\mathrm{T}(n) = \Theta(n^{\lg_3 6})$.

Για τον Ταχύ Μετασχηματισμό Fourier, η πολυπλοκότητα είναι $\Theta(n \lg n)$. Αφού $\lg_3 6 > 1$, ο Ταχύς Μετασχηματισμός Fourier είναι πλεονεκτικότερος (ως προς την πολυπλοκότητα).