Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αλγόριθμοι και Πολυπλοκότητα

Διδάσκοντες: Σ. Ζάχος, Δ. Φωτάκης

Συμπληρωματικές Ασκήσεις - Σχέδιο Λύσεων

Άσκηση 1: Φωλιασμένα Διαστήματα

Αφού τα s_i είναι ταξινομημένα, αρκεί να υπολογίσουμε για πόσα ζεύγη t_i, t_j, με $i < j$, ισχύει ότι $t_j \leq t_i$. Για κάθε τέτοιο ζεύγος t_i, t_j, έχουμε ότι $s_i \leq s_j$ και $t_j \leq t_i$, άρα $[s_j, t_j] \subseteq [s_i, t_i]$. Το πλήθος αυτών των ζευγών μπορεί να υπολογιστεί σε χρόνο $O(n \log n)$ όπως στην Άσκηση 4 της 2ης Γραπτής Εργασίας.

Άσκηση 2: Ενδιάμεσος Δύο Ταξινομημένων Ακολουθιών

Για απλότητα θέτουμε $p = n/2$, και συμβολίζουμε με m το ζητούμενο ενδιάμεσο στοιχείο της ένωσης $A \cup B$ των δύο ακολουθιών. Έστω ότι για τα ενδιάμεσα στοιχεία a_p και b_p των ακολουθιών A και B ισχύει ότι $a_p < b_p$. Τότε παρατηρούμε ότι το a_p θα βρίσκεται μεταξύ των θέσεων $p = n/2$ και $2p - 1 = n - 1$ στην ταξινομημένη ακολουθία $A \cup B$, και ότι το b_p θα βρίσκεται μεταξύ των θέσεων $p + 1 = 1 + n/2$ και $n + p = 3n/2$ στην ταξινομημένη ακολουθία $A \cup B$. Συνεπώς για το ενδιάμεσο στοιχείο m της $A \cup B$ ισχύει ότι $a_p < m \leq b_p$. Έτσι αν $a_p < b_p$, το ενδιάμεσο στοιχείο της $A \cup B$ είναι το ενδιάμεσο στοιχείο της $(a_{p+1}, \ldots, a_n) \cup (b_1, \ldots, b_p)$, όπου οι υπακολουθίες $(a_{p+1}, \ldots, a_n)$ και $(b_1, \ldots, b_p)$ είναι ταξινομημένες. Αντίστοιχα, αν $a_p > b_p$, το ενδιάμεσο στοιχείο της $A \cup B$ είναι το ενδιάμεσο στοιχείο της $(a_1, \ldots, a_p) \cup (b_{p+1}, \ldots, b_n)$, όπου οι υπακολουθίες $(a_1, \ldots, a_p)$ και $(b_{p+1}, \ldots, b_n)$ είναι ταξινομημένες.

Παρατηρούμε λοιπόν ότι συγκρίνοντας τα στοιχεία a_p και b_p, μπορούμε να μειώσουμε το μέγεθος του προβλήματος στο μισό, και εν συνεχεία, να λύσουμε το πρόβλημα αναδρομικά, με τον ίδιο αλγόριθμο. Η ορθότητα μπορεί να αποδειχθεί με επαγωγή, χρησιμοποιώντας την παραπάνω παρατήρηση για την απόδειξη του επαγωγικού βήματος. Η χρονική πολυπλοκότητα δίνεται από την αναδρομική σχέση $T(2n) = T(n) + \Theta(1)$, με αρχική συνθήκη $T(1) = \Theta(1)$, η οποία έχει λύση $T(n) = \Theta(\log n)$.

Άσκηση 3: Το Πρόβλημα της Βιβλιοθήκης

Πρόκειται για το πρόβλημα επιλογής των σελίδων που διατηρούνται στην "γρήγορη" μνήμη σε υπολογιστές με ιεραρχία μνήμης (το πρόβλημα είναι γνωστό ως *paging* ή *cache maintenance*). Η βέλτιστη πολιτική είναι είναι να επιστρέφουμε το βιβλίο που θα χρειαστούμε τελευταίο (από αυτά που έχουμε). Δείτε την Ενότητα 4.3, σελ. 164-171, στο βιβλίο των Kleinberg και Tardos.

Άσκηση 4: Δρομολόγηση Εργασιών

(α) Το ζητούμενο αποδεικνύεται με ένα επιχείρημα ανταλλαγής. Έστω F ένα εφικτό σύνολο που περιλαμβάνει k εργασίες. Αγνοούμε τις άλλες εργασίες του A, και αριθμούμε τις εργασίες του F σε αύξουσα σειρά προθεσμιών $d_1 \leq d_2 \leq \cdots \leq d_k$.

Έστω $\sigma' = (i_1, \ldots, i_k)$ μια δρομολόγηση των εργασιών του F όπου όλες οι εργασίες ολοκληρώνονται εμπρόθεσμα. Ισχύει δηλαδή ότι για κάθε $i_j \in F$, $\sum_{\ell=1}^{j} t_{i_\ell} \leq d_{i_j}$. Μια τέτοια δρομολόγηση υπάρχει, γιατί το F είναι εφικτό, αλλά δεν συμφωνεί κατ' ανάγκη με την δρομολόγηση των εργασιών του F σε αύξουσα σειρά προθεσμιών. Τροποποιώντας την $(i_1, \ldots, i_k)$ όπου είναι απαραίτητο, θα δείξουμε ότι και στην δρομολόγηση $\sigma = (1, \ldots, k)$ σε αύξουσα σειρά προθεσμιών, όλες οι εργασίες ολοκληρώνονται εμπρόθεσμα.

Έστω p, $1 \leq p < k$, η πρώτη θέση στην οποία η δρομολόγηση $\sigma = (1, \ldots, k)$ διαφέρει από την $\sigma' = (i_1, \ldots, i_k)$. Έχουμε λοιπόν ότι για $\ell = 1, \ldots, p-1$, $\ell = i_\ell$. Στην θέση p όμως, η σ δρομολογεί την εργασία p, ενώ η σ' δρομολογεί την εργασία $i_p > p$, και δρομολογεί την εργασία p στην θέση $q > p$ (δηλ. $i_q = p$). Επειδή οι εργασίες δρομολογούνται σε αύξουσα σειρά προθεσμιών στην σ, και επειδή οι δρομολογήσεις σ και σ' ταυτίζονται στις πρώτες $p-1$ θέσεις τους, οι εργασίες στις θέσεις $p, \ldots, q$ της σ' έχουν προθεσμία μεγαλύτερη ή ίση του d_p. Δηλαδή για $\ell = p, \ldots, q$, $d_p \leq d_{i_\ell}$.

Θα τροποποιήσουμε την σ', και θα φτιάξουμε μια εφικτή δρομολόγηση που ταυτίζεται με την σ στις πρώτες p θέσεις. Αφού η δρομολόγηση σ' είναι εφικτή, η εργασία p ολοκληρώνεται εμπρόθεσμα, δηλ. ισχύει ότι $\sum_{\ell=1}^{q} t_{i_\ell} \leq d_p$. Ανταλλάσσουμε τις p και i_p στην δρομολόγηση σ'. Δρομολογούμε δηλαδή την εργασία p στην θέση p (όπως στην σ) και την εργασία i_p στην θέση q. Και στη νέα δρομολόγηση, η εργασία p και οι εργασίες $i_p, \ldots, i_{q-1}$ ολοκληρώνονται όχι αργότερα από τη χρονική στιγμή $\sum_{\ell=1}^{q} t_{i_\ell} \leq d_p$. Αφού για κάθε $\ell = p, \ldots, q$, $d_p \leq d_{i_\ell}$, η εργασία p και οι εργασίες $i_p, \ldots, i_{q-1}$ ολοκληρώνονται εμπρόθεσμα. Παρατηρούμε ακόμη ότι ο χρόνος ολοκλήρωσης των υπόλοιπων εργασιών (δηλ. των εργασιών $1, \ldots, p-1$ και $i_{q+1}, \ldots, i_k$) δεν επηρεάζεται από αυτή την ανταλλαγή. Άρα και στη νέα δρομολόγηση, όλες οι εργασίες ολοκληρώνονται εμπρόθεσμα.

(β) Για τη διατύπωση της λύσης, (ταξινομούμε και) αριθμούμε τις εργασίες σε αύξουσα σειρά με βάση τις προθεσμίες τους, δηλ. $d_1 \leq d_2 \leq \cdots \leq d_n$. Θα χρησιμοποιήσουμε δυναμικό προγραμματισμό. Αρχικά διατυπώνουμε την αρχή της βελτιστότητας: αν η ακολουθία $(i_1, \ldots, i_k)$, με $i_1 < \cdots < i_k$ λόγω του (α), αποτελεί βέλτιστη λύση για το σύνολο εργασιών $A = \{1, \ldots, n\}$ και χρόνο ολοκλήρωσης της τελευταίας εργασίας D, η υπακολουθία $(i_1, \ldots, i_{k-1})$ αποτελεί βέλτιστη λύση για το υποσύνολο εργασιών $A' = \{1, \ldots, i_{k-1}\}$ και χρόνο ολοκλήρωσης της τελευταίας εργασίας $\min\{d_{i_{k-1}}, D - t_k\}$.

Θεωρούμε λοιπόν τα υποσύνολα $A_i = \{1, \ldots, i\}$ που αποτελούνται από τις πρώτες i εργασίες στο σύνολο A, για κάθε $i = 0, \ldots, n$ (προφανώς είναι $A_0 = \emptyset$ και $A_n = A$). Έστω $P(A_i, D)$, $0 \leq D \leq d_i$, το βέλτιστο κέρδος από το υποσύνολο εργασιών A_i με χρόνο ολοκλήρωσης D για την τελευταία εργασία. Υποθέτουμε ότι γνωρίζουμε τα $P(A_{n-1}, d_{n-1})$ και $P(A_{n-1}, \min\{d_n - t_n, d_{n-1}\})$, και εστιάζουμε στην εργασία n. Λόγω του (α), αν η εργασία n επιλεγεί, θα εκτελεστεί τελευταία. Μπορούμε είτε να επιλέξουμε την εργασία n, και να έχουμε κέρδος $p_n + P(A_{n-1}, \min\{d_n - t_n, d_{n-1}\})$, είτε να μην επιλέξουμε την εργασία n, και να έχουμε κέρδος $P(A_{n-1}, d_{n-1})$. Προφανώς επιλέγουμε το καλύτερο από τα δύο. Με άλλα λόγια:

$$P(A_n, d_n) = \max\{p_n + P(A_{n-1}, \min\{d_n - t_n, d_{n-1}\}), P(A_{n-1}, d_{n-1})\}$$

Γενικεύοντας, καταλήγουμε ότι το βέλτιστο κέρδος $P(A_n, d_n)$ υπολογίζεται από την αναδρομική σχέση:

$$P(A_i, D) = \begin{cases} \max\{p_i + P(A_{i-1}, D - t_i), P(A_{i-1}, D)\} & \text{όταν } i \geq 1 \text{ και } 1 \leq D \leq d_i \\ P(A_i, d_i) & \text{όταν } i \geq 1 \text{ και } d_i < D \\ 0 & \text{όταν } i = 0 \text{ ή } D \leq 0 \end{cases}$$

Ο χρόνος για τον υπολογισμό του βέλτιστου κέρδους $P(A_n, d_n)$ είναι $\Theta(n \log n)$ για την ταξινόμηση των εργασιών σε αύξουσα σειρά με βάση τις προθεσμίες τους, και $\Theta(nd_n)$ για τον υπολογισμό

(με δυναμικό προγραμματισμό) των τιμών της παραπάνω αναδρομικής σχέσης και των εργασιών που εκτελούνται, δηλαδή $\Theta(n \log n + nd_n)$ συνολικά.

Άσκηση 5: Κοπή Υφασμάτων

Θα χρησιμοποιήσουμε δυναμικό προγραμματισμό. Για την αρχή της βελτιστότητας, θεωρούμε μια βέλτιστη λύση που κόβει το ύφασμα π.χ. στην θέση k, $1 \leq k < X$, στην οριζόντια διάσταση. Τότε η βέλτιστη εκμετάλλευση του υφάσματος προκύπτει αν θεωρήσουμε την βέλτιστη εκμετάλλευση των δύο τμημάτων υφάσματος που προκύπτουν (ένα με διαστάσεις $k \times Y$ και ένα με διαστάσεις $(X-k) \times Y$). Μάλιστα η βέλτιστη εκμετάλλευση για τα δύο τμήματα υπολογίζεται ανεξάρτητα.

Έτσι το βέλτιστο κέρδος $G(x, y)$, $1 \leq x \leq X$, $1 \leq y \leq Y$, που μπορούμε να έχουμε από ένα κομμάτι υφάσματος με διαστάσεις $x \times y$ δίνεται από την αναδρομική σχέση:

$$G(x,y) = \begin{cases} 0 & \text{αν για κάθε } i \in [n],\ a_i > x \vee b_i > y \\ \max\{\ \max_{1 \leq k < x}\{G(k,y) + G(x-k,y)\}, & \\ \quad \max_{1 \leq k < y}\{G(x,k) + G(x,k-y)\}, & \text{διαφορετικά} \\ \quad \max_{i:a_i \leq x \wedge b_i \leq y}\{c_i\}\ \} & \end{cases}$$

Το βέλτιστο κέρδος $G(X, Y)$ υπολογίζεται από την παραπάνω αναδρομική εξίσωση σε χρόνο $O(X \cdot Y \cdot (n + X + Y))$ με εφαρμογή δυναμικού προγραμματισμού.

Άσκηση 6: Αντιπροσωπεία Φορητών Υπολογιστών

Θα χρησιμοποιήσουμε δυναμικό προγραμματισμό. Για την αρχή της βελτιστότητας, θεωρούμε μια βέλτιστη ακολουθία παραγγελιών $(p_1, \ldots, p_{n-1}, p_n)$ για n ημέρες, όπου μετά το τέλος της n-οστής ημέρας δεν υπάρχουν αδιάθετοι υπολογιστές. Αν $d_n > S$, τότε η μόνη τιμή που μπορεί να πάρει το p_n σε μια βέλτιστη λύση είναι d_n (όλοι οι υπολογιστές που πωλούνται την n-οστή ημέρα παραγγέλνονται την ίδια ημέρα με κόστος K). Σε αυτή την περίπτωση, η υπακολουθία $(p_1, \ldots, p_{n-1})$ είναι μια βέλτιστη πολιτική παραγγελιών για τις $n-1$ πρώτες ημέρες, όπου στο τέλος της $(n-1)$-οστής ημέρας δεν υπάρχουν αδιάθετοι υπολογιστές. Αν $d_n \leq S$, το p_n μπορεί να είναι είτε d_n (όταν $K \leq d_n C$), οπότε ισχύει ότι και παραπάνω, είτε 0 (όταν $K > d_n C$), οπότε η υπακολουθία $(p_1, \ldots, p_{n-1})$ είναι μια βέλτιστη πολιτική παραγγελιών για τις $n-1$ πρώτες ημέρες, υπό την προϋπόθεση ότι στο τέλος της ημέρας $n-1$ υπάρχουν d_n αδιάθετοι υπολογιστές (που θα αποθηκευθούν με κόστος $d_n C$).

Έστω λοιπόν $\text{cost}[i, s]$, $1 \leq i \leq n$, $0 \leq s \leq S$, το ελάχιστο κόστος παραγγελιών για τις πρώτες i ημέρες, υπό την προϋπόθεση ότι στο τέλος της ημέρας i υπάρχουν s διαθέσιμοι υπολογιστές (το κόστος αποθήκευσής τους δεν συμπεριλαμβάνεται στο $\text{cost}[i, s]$). Σύμφωνα με τα παραπάνω, το $\text{cost}[i, s]$ δίνεται από την αναδρομική σχέση:

$$\text{cost}[i,s] = \begin{cases} \min\{\text{cost}[i-1, s+d_i] + (s+d_i)C, \text{cost}[i-1,0] + K\} & \text{αν } i \geq 2 \text{ και } s + d_i \leq S \\ \text{cost}[i-1,0] + K & \text{αν } i \geq 2 \text{ και } s + d_i > S \\ K & \text{αν } i = 1 \end{cases}$$

Το βέλτιστο κόστος δίνεται από το $\text{cost}[n, 0]$, το οποίο υπολογίζεται από την παραπάνω εξίσωση σε χρόνο $O(nS)$. Η ακολουθία των παραγγελιών μπορεί να υπολογισθεί στον ίδιο χρόνο ανατρέχοντας στις επιλογές που οδήγησαν στον υπολογισμό του βέλτιστου κόστους $\text{cost}[n, 0]$ (οι λεπτομέρειες αφήνονται ως άσκηση).