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

Αλγόριθμοι και Πολυπλοκότητα

Διδάσκοντες: Δημήτρης Φωτάκης, Δώρα Σούλιου, Θανάσης Λιανέας

2η Σειρά Γραπτών Ασκήσεων - Σχέδιο Λύσεων

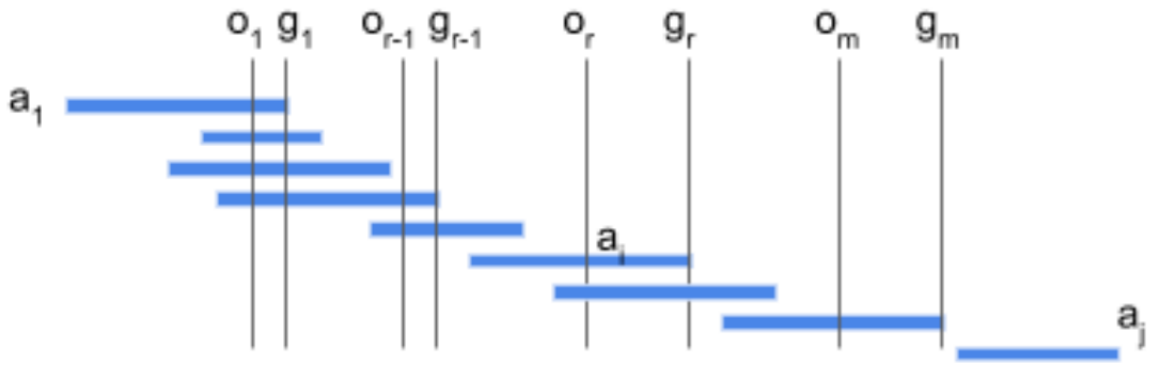
Άσκηση 1: Δίσκοι και Σημεία

Για κάθε δοσμένο σημείο p, μπορούμε να υπολογίσουμε σε σταθερό χρόνο τα δύο σημεία στην ευθεία l, pl και pr, τέτοια ώστε $d(p, pl) = d(p, pr) = r$. Το p καλύπτεται από κάποιον δίσκο, αν και μόνο αν υπάρχει δίσκος με κέντρο ανάμεσα στα pl και pr.

Έτσι το πρόβλημα μας είναι ισοδύναμο με το ακόλουθο πρόβλημα: δοσμένων n διαστημάτων σε μία γραμμή, να υπολογίσετε τον ελάχιστο αριθμό σημείων στην ευθεία τέτοια ώστε κάθε διάστημα να περιέχει τουλάχιστον ένα σημείο. Ταξινομούμε τα διαστήματα ως προς το τέλος τους (δεξί άκρο) και διατηρούμε μια λίστα για τα σημεία την οποία αρχικοποιούμε σαν άδειο σύνολο. Διατρέχουμε τα διαστήματα από τα αριστερά προς τα δεξιά. Αν το αριστερό άκρο του τρέχοντος διαστήματος βρίσκεται δεξιά από το τελευταίο σημείο που προσθέσαμε στη λίστα, το οποίο σημαίνει πως το τελευταίο σημείο στη λίστα βρίσκεται στο τρέχον διάστημα, τότε αγνοούμε αυτό το διάστημα. Εναλλακτικά προσθέτουμε το δεξί άκρο του τρέχοντος διαστήματος στη λίστα ως το τελευταίο σημείο. Η χρονική πολυπλοκότητα του αλγορίθμου ισούται με $O(n \log n)$ εξαιτίας της χρονικής πολυπλοκότητας της ταξινόμησης.

Ας αποδείξουμε ότι αυτός άπληστος αλγόριθμος επιστρέφει βέλτιστη λύση. Δηλώνουμε τα διαστήματα ταξινομημένα από το δεξί άκρο τους (τέλος) ως $a_1, a_2, \ldots, a_n$, τα αριστερά άκρα τους ας ως $s_1, s_2, \ldots, s_n$, και τα δεξιά άκρα τους ως $f_1, f_2, \ldots, f_n$. Οι θέσεις των σημείων που δίνονται από τον άπληστο αλγόριθμο συμβολίζονται ως $g_1, g_2, \ldots, g_k$ (ταξινομημένα από τα αριστερά προς τα δεξιά), και τα σημεία που δίνονται από τον βέλτιστο αλγόριθμο $o_1, o_2, \ldots, o_m$ (ταξινομημένα από τα αριστερά προς τα δεξιά).
Αρχικά, αποδεικνύουμε ότι για κάθε $r \leq m$ ισχύει $g_r \geq o_r$ με επαγωγή.

- Βάση: $r = 1$
- Άπληστη επιλογή: $g_r \geq o_r$ για να καλύψουμε το διάστημα a_1



Σχήμα 1: Απόδειξη ορθότητας του άπληστου αλγόριθμου για την Άσκηση 1.

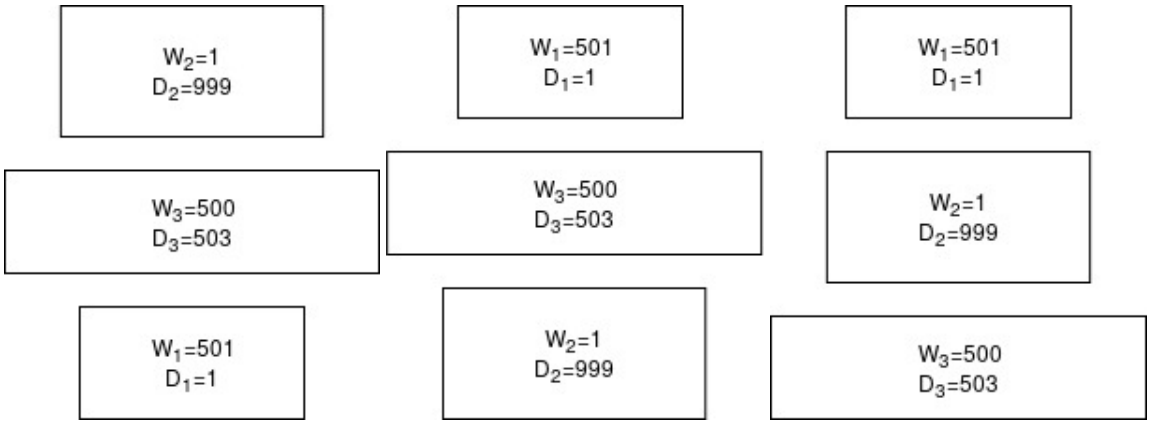
- Επαγωγική υπόθεση: $g_{r-1} \geq o_{r-1}$
- Θα δείξουμε ότι: $g_r \geq o_r$. Έστω a_i το επόμενο διάστημα με $s_i > g_{r-1}$. Ο άπληστος αλγόριθμος δημιουργεί ένα καινούριο σημείο $g_r = f_i$ για να καλύψει το διάστημα a_i. Δεδομένου ότι $o_{r-1} \leq g_{r-1} < s_i$ το o_{r-1} δεν καλύπτει το s_i. Επομένως το o_r πρέπει να καλύψει το a_i που σημαίνει ότι $s_i \leq o_r \leq f_i$. Επομένως $o_r \leq g_r$.

Με παρόμοιο συλλογισμό θα αποδείξουμε ότι $k = m$. Έστω $k > m$. Αν ο άπληστος αλγόριθμος χρειάζεται να προσθέσει κάποιο ακόμη σημείο μετά το g_m, αυτό σημαίνει ότι υπάρχει ένα διάστημα a_j με $s_j > g_m$. Δεδομένου ότι $o_m \leq g_m$, $o_m < s_j$. Έτσι o_m δεν καλύπτει το a_j. Όμως το o_m είναι το πιο δεξί σημείο της βέλτιστης λύσης που αντιτίθεται στην αποδοχή της $o_1, o_2, \ldots, o_m$ ως εφικτής λύσης.

Άσκηση 2: Μεταφορά Δεμάτων

Ερώτημα (α)

Τα κριτήρια (1),(2) δεν δίνουν απαραίτητα σωστή στοίβαξη.
Αντιπαράδειγμα: Για τρία δέματα με $w_1 = 501, d_1 = 1, w_2 = 1, d_2 = 999$ και $w = 500, d_3 = 503$.



Το κριτήριο (3) δίνει σωστή στοίβαξη, εφόσον υπάρχει.
Απόδειξη: Έστω μια σωστή στοίβαξη, όπου το δέμα i είναι ακριβώς πάνω από το δέμα j και $w_i + d_i > w_j + d_j$. Τότε αν εναλλάξουμε τα i,j θα έχουμε και πάλι σωστή στοίβαξη. Πράγματι, αφού στην αρχή η στοίβαξη ήταν σωστή, τότε $W + w_i < d_j \implies W < d_j - w_i < d_i - w_j \implies W + w_j < d_i$. Άρα απο κάθε σωστή στοίβαξη μπορούμε να φτιάξουμε μια σωστή στοίβαξη με τα $w_i + d_i$ διατεταγμένα με τη σωστή σειρά.

Ερώτημα (β)

Παρόμοια με το 0-1 knapsack. Διατάσσουμε τα δέματα σε φθίνουσα σειρά $w_i + d_i$. Για κάθε (i, w), με $d_{i-1} \geq w$, έστω $K(i, w)$ το μέγιστο κέρδος από ασφαλή στοίβαξη δεμάτων από το n μέχρι και το i (από πάνω προς τα κάτω) και με συνολικό βάρος το πολύ w. Η αναδρομική είναι (συμπληρώστε τις αρχικές συνθήκες):

$$K(i, w) = \max(p_i + K(i + 1, \min\{d_i, w - w_i\}), K(i + 1, w))$$

Όπως και στο knapsack η πολυπλοκότητα είναι $O(n \log n + nW)$ δηλαδή ψευδοπολυωνυμική. Η ορθότητα του αλγορίθμου προκύπτει από το ότι όλες οι στοιβάξεις με μέγιστο κέρδος μπορούν να υλοποιηθούν με ασφαλή στοίβαξη. Δε μπορούμε να κάνουμε κάτι καλύτερο από το παραπάνω (μάλλον), αφού μπορούμε να ανάξουμε το πρόβλημα των δεμάτων στο knapsack, όπου M αυθαίρετα μεγάλος αριθμός και p_i, w_i, w οι παραμέτροι του knapsack instance.

Άσκηση 3: Τριγωνοποίηση Πολυγώνου

Αρχικά θα ορίσουμε το $t[i, j]$, για $1 \leq i \leq j \leq n$ ως το βάρος της βέλτιστης τριγωνοποίησης για το πολύγωνο $\langle u_{i-1}, u_i, \ldots, u_j \rangle$. Για την διευκόλυνσής μας θα ορίσουμε για το εκφυλισμένο πολύγωνο $\langle u_{i-1}, u_i \rangle$ βάρος 0. Το βάρος της βέλτιστης τριγωνοποίησης του πολυγώνου P δίνεται από το $t[1, n]$.

Επομένως χρειάζεται να ορίσουμε το $t[i, j]$. αναδρομικά. Η βάση (αρχική συνθήκη) θα είναι η περίπτωση του εκφυλισμένου πολυγώνου δύο κορυφών: $t[i, i] = 0$ για $i = 1, 2, \ldots, n$. Όταν $j - i \geq 1$, έχουμε το πολύγωνο $\langle u_{i-1}, u_i, \ldots, j \rangle$ με τουλάχιστον τρεις κορυφές. Στόχος μας είναι να ελαχιστοποιήσουμε πάνω σε όλες τις κορυφές u_k, για κάθε $k = i, i + 1, \ldots, j - 1$, το βάρος του $\Delta u_{i-1} u_k u_j$ συν το βάρος των βέλτιστων τριγωνοποιήσεων των πολυγώνων $\langle u_{i-1}, u_i, \ldots, k \rangle$ και $\langle u_k, u_{k+1}, \ldots, j \rangle$.

Η αναδρομική σχέση είναι η ακόλουθη:

$$t[i, j] = \left\{ \begin{array}{ll} 0, & \text{if } i = j \\ \min\limits_{i \leq k \leq j-1} \{t[i, k] + t[k + 1, j] + w(\Delta u_{i-1} u_k u_j)\}, & \text{if } i < j \end{array} \right\}$$

Η χρονική πολυπλοκότητα του αλγορίθμου είναι $O(n^3)$ και ο χώρος που χρειάζεται είναι $O(n^2)$

Άσκηση 4: Τοποθέτηση Στεγάστρων (και Κυρτό Κάλυμμα)

Ερώτημα (α)

Για τη βέλτιστη λύση, ξεκινάμε από το σημείο x_f και εξετάζουμε ποιο από τα σημεία x_i θα δώσει το ελάχιστο κόστος όταν χρησιμοποιηθεί ως αρχή της τελευταίας στέγης. Είναι:

$$cost(i) = \min_{0 \leq j \leq i} \{cost(j - 1) + (x_i - x_j)^2 + C\}$$

όπου $cost(i)$ το ελάχιστο κόστος για να καλύψουμε τα σημεία $x_0 = 0$ έως x_i με $cost(-1) = 0$. Χρόνος για την επίλυσή της αναδρομικής με Δυναμικό Προγραμματισμό: $\Theta(n^2)$.

Ερώτημα (β)

Η δομή που κρατάει τη ζητούμενη πληροφορία για τις ευθείες είναι το κυρτό περίβλημα που σχηματίζουν. Αναζητούμε διάταξη των ευθειών $[i(1), i(2), \ldots, i(m)]$ έτσι ώστε η ευθεία $i(1)$ να έχει ελάχιστη τιμή από το $-\infty$ έως το σημείο τομής της με την $i(2)$, η ευθεία $i(2)$ να έχει ελάχιστη τιμή από το σημείο τομής της με την $i(1)$ έως το σημείο τομής της με την $i(3)$ κοκ. Επειδή οι κλίσεις των ευθειών δίνονται ταξινομημένες, κάθε νέα ευθεία θα πρέπει να εισάγεται στο κυρτό περίβλημα, διαγράφοντας πιθανώς κάποιες από τις προηγούμενες. Επειδή όμως κάθε ευθεία εισάγεται και διαγράφεται από το περίβλημα μία το πολύ φορά και έχουμε n ευθείες, μπορούμε να σχηματίσουμε το περίβλημα αυτό σε χρόνο $\Theta(n)$. Έπειτα, για κάθε σημείο x_i που μας δίνεται ξεκινάμε από την πρώτη ευθεία του περιβλήματος και βρίσκουμε πια ευθεία καλύπτει την τιμή x_i την οποία και επιστρέφουμε. Επειδή και τα σημεία είναι ταξινομημένα, ξέρουμε ότι η ευθεία που ελαχιστοποιεί το σημείο x_{i+1} θα είναι πιο μετά στο περίβλημα από την ευθεία που ελαχιστοποιεί το x_i, και συνεπώς δεν χρειάζεται να ξεκινήσουμε από την αρχή αλλά από την τελευταία ευθεία που τυπώσαμε. Άρα, θα διασχίσουμε μία φορά συνολικά το περίβλημα και η συνολική πολυπλοκότητα της λύσης θα είναι $\Theta(n + k)$.

Προσέξτε ότι η αναδρομική σχέση του προηγούμενου ερωτήματος μπορεί να γραφτεί και ως:

$$cost(i) = x_i^2 + C + \min_{0 \le j \le i} \{a[j]x_i + b[j]\}$$

όπου $a[j] = -2x_j$ και $b[j] = cost(j-1) + x_j^2$. Το τετραγωνικό κόστος της αρχικής λύσης προκύπτει από τον υπολογισμό των $\Theta(n)$ ελαχίστων σε γραμμικό χρόνο έκαστο. Ωστόσο, η σχέση αυτή μας δείχνει ότι μπορούμε να αντιμετωπίσουμε το πρόβλημα ως την εύρεση ελάχιστης τιμής από σύνολο ευθειών με κλίσεις $a[j]$ στα σημεία x_i. Επιπλέον, τόσο οι κλίσεις όσο και τα σημεία είναι ταξινομημένα, άρα μπορούμε να εφαρμόσουμε την λύση με το κυρτό περίβλημα και να υπολογίσουμε όλα τα ελάχιστα σε συνολικό χρόνο $\Theta(n)$ αντί για $\Theta(n^2)$. Μία πιο αναλυτική λύση του προβλήματος μπορεί να βρεθεί εδώ.

Άσκηση 5: Καλύπτοντας το Δέντρο

Ερώτημα (α)

Έστω $f(v, d, i)$ το ελάχιστο κόστος κάλυψης από σύνολο μεγέθους i του υποδέντρου με ρίζα το v η οποία έχει απόσταση d. Τότε:

$$f(v, d, i) = \min\{A, B\}\,, \quad \text{όπου:} \tag{1}$$

- A: να ανήκει η κορυφή v στο K
- B: να μην ανήκει η κορυφή v στο K

Το κόστος του υποδέντρου με ρίζα την κορυφή v αν αυτή ανήκει στο K δίνεται από την αναδρομική σχέση:

$$A = \min_{j=0,\ldots,i-1} \{\max\{f(ch_l(v), 1, j), f(ch_r(v), 1, i-1-j)\}\} \tag{2}$$

Η λύση για το δέντρο είναι το μέγιστο κόστος των δύο υποδέντρων (όπου οι ρίζες τους απέχουν 1 από το σύνολο) για την καλύτερη μοιρασιά των $i-1$ εναπομηνάντων στοιχείων του συνόλου κάλυψης.

Το κόστος του υποδέντρου με ρίζα την κορυφή v αν αυτή ΔΕΝ ανήκει στο K δίνεται από την αναδρομική σχέση:

$$B = \min_{j=0,\ldots,i} \{\max\{f(ch_l(v), d+1, j), f(ch_r(v), d+1, i-j), d\}\} \tag{3}$$

Η λύση για το δέντρο είναι το μέγιστο κόστος μεταξύ της ρίζας (δηλ d) και των λύσεων των δύο υποδέντρων (όπου οι ρίζες τους απέχουν $d+1$ από το σύνολο), για την καλύτερη μοιρασιά των i στοιχείων του συνόλου κάλυψης.

Η τελική αναδρομική σχέση δίνεται από τις σχέσεις:1,2,3 με την παρακάτω αρχικοποίηση.

Για v φύλλο:

$$f(v,d,i) = \begin{cases} d, & i = 0 \\ 0, & i > 0 \end{cases}$$

.

Το ζητούμενο κόστος είναι $f(r,0,k)$, όπου στη ρίζα εκτελείται μόνο ο κανόνας . Για να επιστρέψουμε το **σύνολο** κάλυψης αρκεί να αποθηκεύουμε επιπλέον (π.χ. σε έναν δεύτερο πίνακα $g(v,d,i)$) αν η καλύτερη επιλογή ήταν η κορυφή v να ανήκει στο σύνολο κάλυψης ή όχι καθώς και το καλύτερο μοίρασμα των i κορυφών στα υποδέντρα.

Για την πολυπλοκότητα έχουμε τους βρόχους επανάληψης:

- Για κάθε επίπεδο l του δέντρου και για κάθε κορυφή v στο επίπεδο $l \Rightarrow \Theta(n)$
- Για κάθε $d \in \{0,1,\ldots,n-1\} \Rightarrow \Theta(n)$
- Για κάθε $i \in \{0,1,\ldots,k\} \Rightarrow \Theta(k)$
- Υπολόγισε $f(v,d,i) \Rightarrow \Theta(k)$

Άρα εκτελείται σε $O(n^2k^2)$.

Ερώτημα (β)

Greedy αλγόριθμος:

Μέχρι να καλύψουμε το δέντρο:

- Ξεκινώντας από ένα φύλλο στο χαμηλότερο επίπεδο ανεβαίνουμε στο δέντρο μέχρι την κορυφή-πρόγονο σε απόσταση z.
- Προσθέτουμε την κορυφή αυτή στο σύνολο K.
- Διαγράφουμε το υποδέντρο με ρίζα την κορυφή αυτή.

Πολυπλοκότητα:

Διερχόμαστε από κάθε κορυφή σταθερό πλήθος φορών, επομένως έχουμε χρονική πολυπλοκότητα (n).

Απόδειξη Ορθότητας:

Με επαγωγή που χρησιμοποιεί τα παρακάτω.

- Αρχή Βελτιστότητας Υπολύσεων: Αν έχουμε μια βέλτιση λύση K^* για όλο το δέντρο T και ορίσουμε το T' ως το δέντρο που προκύπτει από το T αν αφαιρέσουμε το υποδέντρο που βρίσκεται κάτω από μια κορυφή που ανήκει στο σύνολο K, τότε τα εναπομείναντα στοιχεία του K καλύπτουν βέλτιστα το T'.

- Άπληστη Επιλογή: Έστω v η κορυφή που διαλέγει πρώτη ο αλγόριθμός μας. Θα δείξουμε ότι υπάρχει βέλτιστη λύση που περιέχει την v:
  Έστω ότι η K^* είναι βέλτιστη λύση που δεν περιέχει την v. Τότε αναγκαστικά θα περιέχει κάποια κορυφή-απόγονο της v. Έστω v' αυτή η κορυφή. Τότε, μπορούμε να κατασκευάσουμε λύση με ίσο (ή μικρότερο) πλήθος κορυφών κάλυψης απλά αντικαθιστώντας την v' με την v στο σύνολο κάλυψης.

Θεωρώντας ως μαύρο κουτί τον αλγόριθμο $A(z)$ του δεύτερου ερωτήματος βρίσκουμε λύση για το πρώτο. Κάνουμε δυαδική αναζήτηση ως προς z για να βρούμε το βέλτιστο κόστος z^* για το οποίο ο αλγόριθμος του δεύτερου ερωτήματος επιστρέφει σύνολο μεγέθους το πολύ k. Τυπικά ψάχνουμε το z^* τ.ω.

$$\forall z \geq z^* \quad A(z) \leq k$$
$$\forall z < z^* \quad A(z) > k$$

Αυτό είναι εφικτό γιατί το μέγεθος του βέλτιστου συνόλου είναι φθίνουσα συνάρτηση του z.

Πολυπλοκότητα: $O(n \log n)$ γιατί έχουμε $O(n)$ για τον αλγόριθμο που τρέχει σε κάθε ένα από τα $O(\log n)$ βήματα της δυαδικής αναζήτησης.