Αλγόριθμοι και Πολυπλοκότητα

Γαβαλάς Νίκος, AM 03113121

Νοέμβριος 2018

2η Γραπτή Σειρά Ασκήσεων

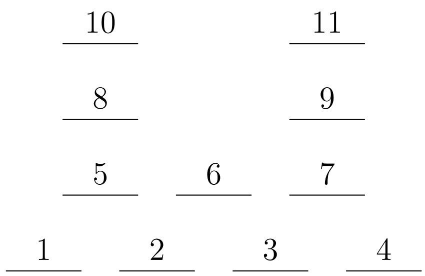
Άσκηση 1

α

α.1

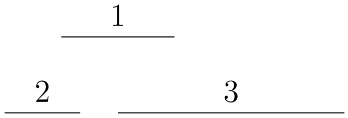
Κανένα από τα τρία άπληστα κριτήρια δεν οδηγούν σε βέλτιστη λύση. Παρακάτω δίδονται ένα αντιπαράδειγμα για κάθε κριτήριο:

<u>Λιγότερες Επικαλύψεις:</u>



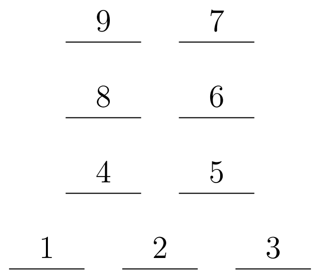
Εδώ βέλτιστη επιλογή είναι $|\{1, 2, 3, 4\}| = 4$ αλλά επιλέγεται η $|\{6, 1, 4\}| = 3$

<u>Μεγαλύτερη Διάρκεια:</u>



Εδώ βέλτιστη είναι η $|\{2, 3\}| = 2$ αλλά επιλέγεται η $|\{2\}| = 1$

<u>Περισσότερες Επικαλύψεις:</u>



Εδώ βέλτιστη είναι η $|\{1,2,3\}| = 3$ αλλά επιλέγεται κάποια σαν την $|\{1,3\}| = 2$

a.2

Αρχικά ταξινομούμε τα χρονικά διαστήματα ως προς τα f_i (χρόνοι ολοκλήρωσης) σε $O(n \log n)$.

Ύστερα κατασκευάζουμε έναν βοηθητικό πίνακα $prev$, κάθε στοιχείο του οποίου είναι ο δείκτης i του χρονικού διαστήματος που η διάρκεια του ολοκληρώνεται αμέσως πριν την έναρξη του i-οστού και δεν έχει επικάλυψη με αυτό. Ο πίνακας αυτός δημιουργείται σε χρόνο $O(n \log n)$, αφού για κάθε μάθημα i από τα n κάνουμε μια δυαδική αναζήτηση σε $O(\log n)$ για να βρούμε τη θέση του προηγούμενου.

Έχοντας τώρα τον $prev$, κατασκευάζουμε την αναδρομική σχέση:

$$creds[i] = \max\{creds[prev[i]] + w[i], creds[i-1]\}$$

, όπου $creds[1] = w[1]$.

Με την σχέση αυτή βρίσκουμε το ζητούμενο, $creds[n]$, σε γραμμικό χρόνο.

Επομένως συνολικά κάνουμε χρόνο $O(n \log n)$.

β

Ταξινομούμε αρχικά τα μαθήματα ως προς τους χρόνους ολοκλήρωσης f_i. Θεωρώντας ότι ανακοίνωση τη στιγμή f_i περιλαμβάνει και τους φοιτητές του i-οστού τμήματος, επιλέγουμε χρονική στιγμή ανακοίνωσης τον χρόνο ολοκλήρωσης του πρώτου «ακάλυπτου» διαστήματος που βρίσκουμε, αφαιρούμε τα διαστήματα που επικαλύπτονται με αυτή την επιλογή και συνεχίζουμε στο επόμενο ακάλυπτο.

Η εύρεση (και διαγραφή - μαρκάρισμα) των επικαλυπτόμενων μπορεί να γίνει σε λογαριθμικό χρόνο αν ταξινομήσουμε μια δεύτερη φορά ως προς τους χρόνους έναρξης.

Ορθότητα: Έστω O^* μια βέλτιστη λύση και O η δικη μας. Έστω ότι οι δύο αλγόριθμοι που αντιστοιχούν σε αυτές διαφέρουν για πρώτη φορά κατά την επιλογή της χρονικής στιγμής i. Υποθέτουμε επίσης ότι ο βέλτιστος αλγόριθμος, στο σημείο που οι δύο λύσεις διαφέρουν για πρώτη φορά επιλέγει τη χρονική στιγμή o^* ενώ ο δικός μας την $o = f_k$.

- Αν $o^* < o$, ο βέλτιστος αλγόριθμος έχει επιλέξει χρονική στιγμή που καλύπτει τουλάχιστον όσα ο δικός μας, αφού μετά τη χρ. στιγμή που οι αλγόριθμοι δίνουν ίδια λύση και πρίν την f_k δεν τελειώνει άλλο διάστημα ενδιάμεσα.
- Αν $o^* > o$, ο βέλτιστος αλγόριθμος χάνει το διάστημα $[s_k, f_k)$, άρα δεν ορθός.

Επομένως $o = o^*$.

Άσκηση 2

Έχουμε f_i, p_i, u_i να είναι αντίστοιχα η τιμή της δύναμης του i-οστού σφυριού, οι τιμές των δυνάμεων που χρειάζεται το i-οστό κουτί για να σπάσει, και τα ευρώ που κερδίζονται από το i-οστό κουτί.

Αρχικά ταξινομούμε ως προς u_i τα κουτιά, και ύστερα ξεκινώντας από τα κουτιά με τη μεγαλύτερη αμοιβή, βλέπουμε για κάθε ένα πόση δύναμη χρειάζεται για να σπάσει, οπότε έπειτα βρίσκουμε από τα διαθέσιμα σφυριά ποιό είναι εκείνο με την ελάχιστη f_i που μπορεί να το σπάσει ($f_i \geq p_i$) .

Το lookup για το σφυρί με αυτό το χαρακτηριστικό μπορεί να γίνει σε χρόνο $O(\log n)$ με μια δυαδική αναζήτηση, αφού πρώτα «πληρώσουμε» χρόνο $O(n \log n)$ για να τα ταξινομήσουμε.

Αφού βρούμε ποιο σφυρί χρειάζεται, το αφαιρούμε από τη λίστα με τα σφυριά. Αυτό μπορεί να γίνει αποδοτικά (μαζί με τη διαδικασία της αναζήτησης) με χρήση ενός Binary Search Tree (ή ακόμα καλύτερα με χρήση ενός AVL Tree για να έχουμε διαγραφή σε λογαριθμικό χρόνο σίγουρα).

Η πολυπλοκότητα του αλγορίθμου αυτού είναι $O(n \log n + n \log n + n(\log n + \log n)) = O(n \log n)$.

Άσκηση 3

Έστω n οι χώρες, με k_i αναμνηστικά η καθεμία, και c_{ij}, p_{ij} το κόστος και η «συναισθηματική αξία» του καθενός από αυτά. Θέλουμε να επιλέξουμε το πολύ ένα από κάθε χώρα, (για την ακρίβεια θέλουμε ακριβώς ένα, αλλά αφού κάθε χώρα μπορεί να έχει αντικείμενο με συναισθηματική αξία 0 και κόστος 0, το πρόβλημα είναι ισοδύναμο με το «το πολύ ένα») ώστε το $\sum p_{ij}$ τους να είναι μέγιστο με τον περιορισμό $\sum c_{ij} \leq C$.

Η αναδρομική σχέση που επιλύει το πρόβλημα είναι η εξής:

$$opt(i, c) = \max_{0<j<k_i} \{opt(i-1, c-c_{ij}) + p_{ij}\}$$

, όπου i η χώρα, και c το τρέχον ποσό.

Η σχέση αποτελεί παραλλαγή του knapsack και υλοποιείται όπως αυτό με τη χρήση πίνακα, και έχει χρονική πολυπλοκότητα $O(C \sum_{i=0}^{n} k_i)$. Η τιμή που αποτελεί τη λύση και βρίσκουμε βάσει της αναδρομικής είναι η $opt(n, C)$.

Άσκηση 4

Αρχικά, εφόσον πρέπει $q_i \leq q_j, i < j$, ταξινομούμε ως προς τα q_i για να μπορούμε να «κινούμαστε» ως προς μία κατεύθυνση (αντί για δύο) - δεξιά. Κατά την ταξινόμηση αυτή φροντίζουμε να διατηρήσουμε τα αρχικά indexes για να μπορέσουμε αργότερα να υπολογίσουμε την απόσταση-χρόνο.

Για την εύρεση του ελάχιστου χρόνου χρησιμοποιούμε την εξής αναδρομική σχέση:

$$opt(i, q) = \min_{0<j<i} \{opt(j, q-q_i) + d(i, j)\}, t_i \neq t_j$$

, όπου q ο αριθμός των σοκολατών ως τώρα, q_i ο αριθμός των σοκολατών του i-οστού κουτιού, $d(i, j)$ η απόσταση-χρόνος μεταξύ των δύο κουτιών (όπως υπολογίζεται από την διαφορά των δεικτών τους που έχουμε κρατήσει), και φυσικά κάθε φορά ισχύει η προϋπόθεση $t_i \neq t_j$, δηλαδή να είναι διαφορετικού τύπου δύο διαδοχικά κουτιά.

Η απάντηση δίνεται από τον υπολογισμό της τιμής $\min_{0<i<n}\{opt(i, Q)\}$. Η πολυπλοκότητα του αλγορίθμου αυτού είναι $O(n^2 Q)$.

Ασκηση 5

Έχουμε n κεραίες που μπορούν να λειτουργούν ως πομποί ή δέκτες, καταναλώνοντας αντίστοιχα ενέργεια (T_i, R_i), με $R_i \leq T_i$, και θέλουμε να τις χωρίσουμε όλες σε ζευγάρια με τρόπο τέτοιο ώστε η συνολική κατανάλωση του δικτύου να είναι η ελάχιστη δυνατή.

Το ότι ισχύει $R_i \leq T_i$ σημαίνει ότι η διαφορά $T_i - R_i$ είναι πάντα μη αρνητική. Το γεγονός αυτό μας επιτρέπει να ακολουθήσουμε την εξής άπληστη προσέγγιση: Έστω ότι μέχρι την i-οστή κεραία έχουμε βέλτιστη λύση και θέλουμε να αποφασίσουμε ως τι θα λειτουργεί η επόμενη κεραία.

- Αν είναι το i περιττός, τότε κάνουμε την κεραία $i + 1$ δέκτη, και προσθέτουμε το R_i στην συνολική κατανάλωση.
- Αν είναι το i άρτιος, τότε κάνουμε την $i+1$ κεραία δέκτη (και προσθέτουμε R_i στη συνολική κατανάλωση), αλλά αυτή τη φορά πρέπει κάποιος δέκτης από τα προηγούμενα να γίνει πομπός, γιατί πρέπει συνολικά τα ζεύγη να είναι $n/2$.

Σχετικά με την επιλογή του πομπού, γίνεται πομπός αυτός που τελικά θα μας εξασφαλίσει ελάχιστη συνολική ενέργεια (και αυτό είναι που εξασφαλίζει και τη βελτιστότητα), δηλαδή ο πομπός j στον οποίο αντιστοιχεί η μικρότερη διαφορά $T_j - R_j$. Τη διαφορά αυτή την «παρακολουθούμε» αποδοτικά με χρήση ενός min-heap με στοιχεία τα ζεύγη $(T_i - R_i, i)$.

Ο αλγόριθμος αυτός κάνει $O(n \log n)$ για την κατασκευή του heap, και ύστερα για κάθε στοιχείο n κάνει $O(1)$ για την εύρεση της ελάχιστης διαφοράς και $O(\log n)$ για να αναδιατάξει το heap, οπότε έχει συνολικά χρονική πολυπλοκότητα $O(n \log n)$.