Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Θεμελιώδη Θέματα Επιστήμης Υπολογιστών, 2020-21

2η σειρά γραπτών ασκήσεων

(αλγοριθμικές τεχνικές – αριθμητικοί αλγόριθμοι
αλγόριθμοι γράφων – δυναμικός προγραμματισμός)

Άσκηση 1. (Αναδρομή – Επανάληψη – Επαγωγή)

(α) Εκφράστε τον αριθμό κινήσεων δίσκων που κάνει ο αναδρομικός αλγόριθμος για τους πύργους του Hanoi, σαν συνάρτηση του αριθμού των δίσκων n.

(β) Δείξτε ότι ο αριθμός κινήσεων του αναδρομικού ισούται με τον αριθμό μετακινήσεων του επαναληπτικού αλγορίθμου.

(γ) Δείξτε ότι ο αριθμός των κινήσεων των παραπάνω αλγορίθμων είναι ο ελάχιστος μεταξύ όλων των δυνατών αλγορίθμων για το πρόβλημα αυτό.

(δ*) Θεωρήστε το πρόβλημα των πύργων του Hanoi με 4 αντί για 3 πασσάλους. Σχεδιάστε αλγόριθμο μετακίνησης n δίσκων από τον πάσσαλο 1 στον πάσσαλο 4 ώστε το πλήθος των βημάτων να είναι σημαντικά μικρότερο από το πλήθος των βημάτων που απαιτούνται όταν υπάρχουν μόνο 3 πάσσαλοι. Εκφράστε τον αριθμό των απαιτούμενων βημάτων σαν συνάρτηση του n.

Άσκηση 2. (Επαναλαμβανόμενος Τετραγωνισμός – Κρυπτογραφία)

(α) Γράψτε πρόγραμμα σε γλώσσα της επιλογής σας (θα πρέπει να υποστηρίζει πράξεις με αριθμούς 100δων ψηφίων) που να ελέγχει αν ένας αριθμός είναι πρώτος με τον έλεγχο (test) του Fermat:

Αν n πρώτος τότε για κάθε a τ.ώ. $1 < a < n-1$, ισχύει

$$a^{n-1} \bmod n = 1$$

Αν λοιπόν, δεδομένου ενός n βρεθεί a ώστε να μην ισχύει η παραπάνω ισότητα τότε ο αριθμός n είναι οπωσδήποτε σύνθετος. Αν η ισότητα ισχύει, τότε το n είναι πρώτος με αρκετά μεγάλη πιθανότητα (για τους περισσότερους αριθμούς $\geq 1/2$). Για να αυξήσουμε σημαντικά την πιθανότητα μπορούμε να επαναλάβουμε μερικές φορές (τυπικά 30 φορές) με διαφορετικό a. Αν όλες τις φορές βρεθεί να ισχύει η παραπάνω ισότητα τότε λέμε ότι το n "περνάει το test" και ανακηρύσσουμε το n πρώτο αριθμό·αν έστω και μία φορά αποτύχει ο έλεγχος, τότε ο αριθμός είναι σύνθετος.

Η συνάρτησή σας θα πρέπει να δουλεύει σωστά για αριθμούς χιλιάδων ψηφίων. Δοκιμάστε την με τους αριθμούς:

$67280421310721, 170141183460469231731687303715884105721, 2^{2281}-1, 2^{9941}-1, 2^{19939}-1$

Σημείωση 1: το $a^{2^{19939}-2}$ έχει 'αστρονομικά' μεγάλο πλήθος ψηφίων (δεν χωράει να γραφτεί σε ολόκληρο το σύμπαν!), ενώ το $a^{2^{19939}-2} \bmod (2^{19939}-1)$ είναι σχετικά "μικρό" (έχει μερικές χιλιάδες δεκαδικά ψηφία *μόνο* :-)).

Σημείωση 2: Υπάρχουν (λίγοι) σύνθετοι που έχουν την ιδιότητα να περνούν τον έλεγχο Fermat για κάθε a που είναι σχετικά πρώτο με το n, οπότε για αυτούς το test θα αποτύχει όσες δοκιμές και αν γίνουν (εκτός αν πετύχουμε κατά τύχη a που δεν είναι σχετικά πρώτο με το n, πράγμα αρκετά απίθανο). Αυτοί οι αριθμοί λέγονται *Carmichael* – δείτε και `http://en.wikipedia.org/wiki/Carmichael_`

number. Ελέγξτε τη συνάρτησή σας με *αρκετά μεγάλους* αριθμούς Carmichael που θα βρείτε π.χ. στη σελίδα http://de.wikibooks.org/wiki/Pseudoprimzahlen:_Tabelle_Carmichael-Zahlen. Τι παρατηρείτε;

(β) Μελετήστε και υλοποιήστε τον έλεγχο Miller-Rabin που αποτελεί βελτίωση του ελέγχου του Fermat και δίνει σωστή απάντηση με πιθανότητα τουλάχιστον 1/2 για *κάθε* φυσικό αριθμό (οπότε με 30 επαναλήψεις έχουμε αμελητέα πιθανότητα λάθους για κάθε αριθμό εισόδου). Δοκιμάστε τον με διάφορους αριθμούς Carmichael. Τι παρατηρείτε;

(γ*) Γράψτε πρόγραμμα που να βρίσκει όλους τους πρώτους αριθμούς Mersenne, δηλαδή της μορφής $n = 2^x - 1$ με $100 < x < 1000$ (σημειώστε ότι αν το x δεν είναι πρώτος, ούτε το $2^x - 1$ είναι πρώτος – μπορείτε να το αποδείξετε;). Αντιπαραβάλετε με όσα αναφέρονται στην ιστοσελίδα https://www.mersenne.org/primes/.

Άσκηση 3. (Αριθμοί Fibonacci)

(α) Υλοποιήστε και συγκρίνετε τους εξής αλγορίθμους για υπολογισμό του n-οστού αριθμού Fibonacci: αναδρομικό με memoization, επαναληπτικό, και με πίνακα.

Υλοποιήστε τους αλγορίθμους σε γλώσσα που να υποστηρίζει πολύ μεγάλους ακεραίους (100δων ψηφίων), π.χ. σε Python. Χρησιμοποιήστε τον πολλαπλασιασμό ακεραίων που παρέχει η γλώσσα. Τι συμπεραίνετε;

(β*) Δοκιμάστε να λύσετε το παραπάνω πρόβλημα με ύψωση σε δύναμη, χρησιμοποιώντας τη σχέση του F_n με το ϕ (χρυσή τομή). Τι παρατηρείτε;

(γ) Υπολογίστε και συγκρίνετε την πολυπλοκότητα ψηφιοπράξεων (bit complexity) του επαναληπτικού αλγορίθμου για υπολογισμό αριθμών Fibonacci και του αλγορίθμου που χρησιμοποιεί πίνακα. Για τον αλγόριθμο με πίνακα θεωρήστε (α) απλό πολλαπλασιασμό ακεραίων και (β) πολλαπλασιασμό Gauss-Karatsuba. Τι παρατηρείτε σε σχέση και με το ερώτημα (α);

(δ) Υλοποιήστε συνάρτηση που να δέχεται σαν είσοδο δύο θετικούς ακεραίους n, k και να υπολογίζει τα k λιγότερο σημαντικά ψηφία του n-οστού αριθμού Fibonacci.

(ε*) Αναζητήστε και εξετάστε τη μέθοδο Fast Doubling σε σχέση με τα παραπάνω ερωτήματα. Συγκρίνετέ την με τη μέθοδο του πίνακα.

Άσκηση 4. (Λιγότερα Διόδια)

Θεωρήστε το εξής πρόβλημα σε οδικά δίκτυα: κάθε κόμβος έχει διόδια (μια θετική ακέραια τιμή) και θέλουμε να βρεθούν οι διαδρομές με το ελάχιστο κόστος έναν αρχικό κόμβο s προς κάθε άλλο κόμβο. Θεωρήστε ότι στον αρχικό κόμβο δεν πληρώνουμε διόδια (ενώ στον τελικό πληρώνουμε, όπως και σε κάθε ενδιάμεσο) και ότι το δίκτυο παριστάνεται σαν κατευθυνόμενος γράφος. Περιγράψτε όσο το δυνατόν πιο αποδοτικούς αλγόριθμους για το πρόβλημα αυτό στις εξής περιπτώσεις:

(α) Το δίκτυο δεν έχει κύκλους (κατευθυνόμενους).

(β) Το δίκτυο μπορεί να έχει κύκλους.

(γ) Σε κάποιους κόμβους δίνονται προσφορές που μπορεί να είναι μεγαλύτερες από το κόστος διέλευσης (αλλά δεν υπάρχει κύκλος όπου συνολικά οι προσφορές να υπερβαίνουν το κόστος).

Άσκηση 5. (Επιβεβαίωση Αποστάσεων - Δέντρο Συντομότερων Μονοπατιών)

Θεωρούμε ένα κατευθυνόμενο γράφημα $G(V, E, \ell)$ με n κορυφές, m ακμές και (ενδεχομένως αρνητικά) μήκος $\ell(e)$ σε κάθε ακμή του $e \in E$. Συμβολίζουμε με $d(u, v)$ την απόσταση των κορυφών u

και v (δηλ. το $d(u, v)$ είναι ίσο με το μήκος της συντομότερης $u - v$ διαδρομής) στο G. Δίνονται n αριθμοί $\delta_1, \ldots, \delta_n$, όπου κάθε δ_k (υποτίθεται ότι) ισούται με την απόσταση $d(v_1, v_k)$ στο G. Να διατυπώσετε αλγόριθμο που σε χρόνο $\Theta(n + m)$, δηλ. γραμμικό στο μέγεθος του γραφήματος, ελέγχει αν τα $\delta_1, \ldots, \delta_n$ πράγματι ανταποκρίνονται στις αποστάσεις των κορυφών από την v_1, δηλαδή αν για κάθε $v_k \in V$, ισχύει ότι $\delta_k = d(v_1, v_k)$. Αν αυτό αληθεύει, ο αλγόριθμός σας πρέπει να υπολογίζει και να επιστρέφει ένα Δέντρο Συντομότερων Μονοπατιών με ρίζα τη v_1 (χωρίς ο χρόνος εκτέλεσης να ξεπεράσει το $\Theta(n + m)$).

Άσκηση 6. (Προγραμματισμός Διακοπών)

Μπορούμε να αναπαραστήσουμε το οδικό δίκτυο μιας χώρας ως ένα συνεκτικό μη κατευθυνόμενο γράφημα $G(V, E, \ell)$ με n κορυφές και m ακμές. Κάθε πόλη αντιστοιχεί σε μια κορυφή του γραφήματος και κάθε οδική αρτηρία σε μία ακμή. Κάθε οδική αρτηρία $e \in E$ συνδέει δύο πόλεις και έχει μήκος $\ell(e)$ χιλιόμετρα. Η ιδιαιτερότητα της συγκεκριμένης χώρας είναι ότι έχουμε σταθμούς ανεφοδιασμού σε καύσιμα μόνο στις πόλεις / κορυφές του γραφήματος, όχι στις οδικές αρτηρίες / ακμές.

Θέλουμε να ταξιδέψουμε από την πρωτεύουσα s σε ένα ορεινό θέρετρο t για διακοπές, για να αλλάξουμε παραστάσεις, μετά την άρση του lockdown. Θα χρησιμοποιήσουμε το αυτοκίνητό μας που διαθέτει αυτονομία καυσίμου για L χιλιόμετρα.

(α) Να διατυπώσετε έναν αλγόριθμο, με όσο το δυνατόν μικρότερη χρονική πολυπλοκότητα, που υπολογίζει αν κάτι τέτοιο είναι εφικτό. Ποια είναι η χρονική πολυπλοκότητα του αλγορίθμου σας στη χειρότερη περίπτωση;

(β) Να διατυπώσετε αλγόριθμο με χρονική πολυπλοκότητα $(m \log m)$ που υπολογίζει την ελάχιστη αυτονομία καυσίμου (σε χιλιόμετρα) που απαιτείται για το ταξίδι από την πόλη s στην πόλη t.

(γ*) Να διατυπώσετε αλγόριθμο με *γραμμική* χρονική πολυπλοκότητα που υπολογίζει την ελάχιστη αυτονομία καυσίμου για το ταξίδι από την πόλη s στην πόλη t. *Υπόδειξη:* Εδώ μπορεί να σας φανεί χρήσιμο ότι (με λογική αντίστοιχη με αυτή της Quicksort) μπορούμε να υπολογίσουμε τον median ενός μη ταξινομημένου πίνακα σε γραμμικό χρόνο.

Άσκηση 7. (Αγορά Εισιτηρίων)

Εκτός από τις διακοπές σας, έχετε προγραμματίσει προσεκτικά και την παρουσία σας στη Σχολή, μετά την άρση του lockdown. Συγκεκριμένα, έχετε σημειώσει ποιες από τις T ημέρες, που θα ακολουθήσουν την άρση του lockdown, θα έρθετε στο ΕΜΠ για να παρακολουθήσετε μαθήματα και εργαστήρια και να δώσετε εξετάσεις. Το πρόγραμμά σας έχει τη μορφή ενός πίνακα S με T θέσεις, όπου για κάθε ημέρα $t = 1, \ldots, T$, $S[t] = 1$, αν θα έρθετε στο ΕΜΠ, και $S[t] = 0$, διαφορετικά. Το πρόγραμμά σας δεν έχει κανονικότητα και επηρεάζεται από διάφορες υποχρεώσεις και γεγονότα.

Για κάθε μέρα που θα έρθετε στο ΕΜΠ, πρέπει να αγοράσετε εισιτήριο που να επιτρέπει τη μετακίνησή σας με τις αστικές συγκοινωνίες. Υπάρχουν συνολικά k διαφορετικοί τύποι εισιτηρίων (π.χ., ημερήσιο, τριών ημερών, εβδομαδιαίο, μηναίο, εξαμηνιαίο, ετήσιο). Ο τύπος εισιτηρίου i σας επιτρέπει να μετακινηθείτε για c_i διαδοχικές ημέρες (μπορεί βέβαια κάποιες από αυτές να μην χρειάζεται να έρθετε στο ΕΜΠ) και κοστίζει p_i ευρώ. Για τους τύπους των εισιτηρίων, ισχύει ότι $1 = c_1 < c_2 < \cdots < c_k \leq T$, $p_1 < p_2 < \cdots < p_k$, και $p_1/c_1 > p_2/c_2 > \cdots > p_k/c_k$ (δηλαδή, η τιμή αυξάνεται με τη διάρκεια του εισιτηρίου, αλλά η τιμή ανά ημέρα μειώνεται).

Να διατυπώσετε αλγόριθμο που υπολογίζει τον συνδυασμό τύπων εισιτηρίων με ελάχιστο συνολικό κόστος που καλύπτουν όλες τις ημέρες που θα έρθετε στο ΕΜΠ. Ποια είναι η χρονική πολυπλοκότητα του αλγορίθμου σας στη χειρότερη περίπτωση;

Άσκηση 8. (Αντιπροσωπεία Φορητών Υπολογιστών)

Επιθυμούμε να βελτιστοποιήσουμε την λειτουργία μιας νέας αντιπροσωπείας φορητών υπολογιστών για τις επόμενες n ημέρες. Για κάθε ημέρα i, $1 \leq i \leq n$, υπάρχει μια (απόλυτα ασφαλής) πρόβλεψη d_i του αριθμού των πωλήσεων. Όλες οι πωλήσεις λαμβάνουν χώρα το μεσημέρι, και όσοι φορητοί υπολογιστές δεν πωληθούν, αποθηκεύονται. Υπάρχει δυνατότητα αποθήκευσης μέχρι S υπολογιστών, και το κόστος είναι C για κάθε υπολογιστή που αποθηκεύεται και για κάθε ημέρα αποθήκευσης. Το κόστος μεταφοράς για την προμήθεια νέων υπολογιστών είναι K ευρώ, ανεξάρτητα από το πλήθος των υπολογιστών που προμηθευόμαστε, και οι νέοι υπολογιστές φθάνουν λίγο πριν το μεσημέρι (άρα αν πωληθούν αυθημερόν, δεν χρειάζονται αποθήκευση). Αρχικά δεν υπάρχουν καθόλου υπολογιστές στην αντιπροσωπεία.

Το ζητούμενο είναι να προσδιορισθούν οι παραγγελίες (δηλ. πόσους υπολογιστές θα παραγγείλουμε και πότε) ώστε να ικανοποιηθούν οι προβλεπόμενες πωλήσεις με το ελάχιστο δυνατό συνολικό κόστος (αποθήκευσης και μεταφοράς). Να διατυπώσετε αλγόριθμο δυναμικού προγραμματισμού με χρόνο εκτέλεσης πολυωνυμικό στο nS για τη βελτιστοποίηση της λειτουργίας της αντιπροσωπείας.

Προθεσμία υποβολής και οδηγίες. Οι απαντήσεις θα πρέπει να υποβληθούν έως τις 03/12/2020, και ώρα 22:00, σε ηλεκτρονική μορφή, στο mycourses (προσπαθήστε το τελικό αρχείο να είναι μεγέθους <2MB συνολικά).

Τα ερωτήματα με (*) είναι προαιρετικά. Εφ'όσον τα επιλύσετε μπορούν να προσμετρηθούν στη θέση ερωτημάτων που δεν απαντήσατε.

Συνιστάται *θερμά* να αφιερώσετε ικανό χρόνο για να λύσετε τις ασκήσεις μόνοι σας προτού καταφύγετε σε οποιαδήποτε *θεμιτή* βοήθεια (διαδίκτυο, βιβλιογραφία, συζήτηση με συμφοιτητές). Σε κάθε περίπτωση, οι απαντήσεις θα πρέπει να είναι *αυστηρά* ατομικές και να περιλαμβάνουν αναφορές σε κάθε πηγή που χρησιμοποιήσατε.

Για να βαθμολογηθείτε θα πρέπει να παρουσιάσετε σύντομα τις λύσεις σας σε ημέρα και ώρα που θα ανακοινωθεί αργότερα.

Για απορίες / διευκρινίσεις: στείλτε μήνυμα στη διεύθυνση `focs@corelab.ntua.gr`.