Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Θεμελιώδη Θέματα Επιστήμης Υπολογιστών, 2016-17

Επιμέλεια ασκήσεων: Ε. Ζάχος, Α. Παγουρτζής, Π. Ποτίκας, Α. Χαλκή

2η σειρά γραπτών ασκήσεων - Υποδείξεις λύσεων
(αλγοριθμικές τεχνικές – αλγόριθμοι γράφων και δικτύων – εφαρμογές)

Άσκηση 1. (Αναδρομή – Επανάληψη – Επαγωγή)

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

Υπόδειξη: Τρέχουμε τον αλγόριθμο για τιμές του $n = 1, 2, 3$ και παρατηρούμε ότι χρειάζεται 1, 3, 7 κινήσεις αντίστοιχα, οπότε υποψιαζόμαστε ότι ο αριθμός των κινήσεων είναι $2^n - 1$. Για να το επαληθεύσουμε χρησιμοποιούμε επαγωγή στο πλήθος των δίσκων n.

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

Διευκρίνιση: Αν ο n είναι περιττός τότε κινούμαστε κατά τη φορά των δεικτών του ρολογιού, ενώ αν είναι άρτιος αντίθετα (υποθέτουμε ότι η σειρά των πασσάλων είναι αντίθετη από τη φορά του ρολογιού, δηλαδή όπως στη διαφάνεια 12). Υπόδειξη: τρέξτε τον επαναληπτικό αλγόριθμο για $n = 1, 2, 3$ για να δείτε πώς ακριβώς λειτουργεί και χρησιμοποιήστε επαγωγή. Προσοχή στη φορά κίνησης.

(γ)* Δείξτε ότι ο αριθμός των μετακινήσεων του αναδρομικού αλγορίθμου είναι ο ελάχιστος μεταξύ όλων των δυνατών αλγορίθμων για το πρόβλημα αυτό.

Υπόδειξη: Με επαγωγή. Για να μετακινήσουμε τον πιο μεγάλο δίσκο, πρέπει να μετακινηθούν οι $n - 1$ πιο μικροί που βρίσκονται από πάνω του και αυτό να γίνει με τον βέλτιστο τρόπο.

(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.

Υπόδειξη: χωρίστε τους πασσάλους σε δύο ομάδες και προσπαθήστε να μετακινήσετε κατάλληλα τις ομάδες αυτές.

Άσκηση 2. (Λογική και Αλγόριθμοι)

Διατυπώστε αποδοτικό αλγόριθμο που να δέχεται σαν είσοδο οποιονδήποτε τύπο της προτασιακής λογικής σε διαζευκτική κανονική μορφή (DNF) και να αποφαίνεται αν είναι ικανοποιήσιμος. Σε περίπτωση που είναι θα πρέπει να επιστρέφει μία ανάθεση αληθοτιμών που ικανοποιεί τον τύπο.

Π.χ. με είσοδο $(x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_2 \wedge x_3) \vee (x_4 \wedge x_5 \wedge \neg x_6)$ θα πρέπει να επιστρέφει 'Ναι' και μία από τις αναθέσεις αληθοτιμών στις $(x_1, \ldots, x_6)$ που ικανοποιούν τον τύπο, π.χ. την ανάθεση $(\text{True}, \text{False}, \text{True}, \text{False}, \text{False}, \text{False})$ ενώ με είσοδο $(x_1 \wedge \neg x_1 \wedge x_2) \vee (\neg x_3 \wedge x_3)$ θα πρέπει να επιστρέφει 'Όχι'.

Θεωρήστε ότι όλες οι μεταβλητές ενός τύπου δίνονται στη μορφή x_n, όπου n ένας φυσικός αριθμός.

Υπόδειξη: Η είσοδος $(x_1 \wedge \neg x_2 \wedge x_3) \vee (\neg x_2 \wedge x_3) \vee (x_4 \wedge x_5 \wedge \neg x_6)$ περιέχει 3 φράσεις (clauses): $C_1 = \{x_1, \neg x_2, x_3\}$, $C_2 = \{\neg x_2, x_3\}$, $C_3 = \{x_4, x_5, \neg x_6\}$. Στη γενική περίπτωση θεωρήστε ότι η

είσοδος περιέχει τις clauses C_i, $1 \leq i \leq m$, και τις μεταβλητές x_j, $1 \leq j \leq n$. Μπορείτε να γράφετε εκφράσεις όπως "Αν κάποια clause περιέχει μια μεταβλητή και την άρνησή της, τότε..." ως εξής:

for all i
  for all j
    if $x_j \in C_i$ **and** $\neg x_j \in C_i$ **then** ...

Γίνονται δεκτές και λύσεις που περιγράφουν τον αλγόριθμο σε φυσική γλώσσα αρκεί να είναι αρκετά σαφής η διαδικασία.

Άσκηση 3. (Αλγόριθμοι γράφων / δικτύων)

Περιγράψτε σε ψευδοκώδικα όσο το δυνατόν πιο αποδοτικό αλγόριθμο που να δέχεται σαν είσοδο έναν γράφο (δοσμένο με μορφή πίνακα γειτνίασης) και να επιστρέφει έναν κύκλο Euler ή μια διαδρομή Euler (αν υπάρχει κάτι από τα δύο). Ποια είναι η πολυπλοκότητα του αλγορίθμου σας και γιατί;

Υπόδειξη: Ελέγχουμε αν έχει κύκλο (όλοι οι κόμβοι έχουν άρτιο βαθμό). Αν ναι, ξεκινάμε από έναν οποιονδήποτε κόμβο v και ακολουθώντας ακμές που δεν έχουμε ακόμη επισκεφθεί καταλήγουμε κάποια στιγμή στον v (δεν εγκλωβιζόμαστε ποτέ σε κάποιο κόμβο, γιατί ο βαθμός κάθε κόμβου είναι άρτιος). Άρα έχουμε φτιάξει έναν κύκλο. Αν έχουμε επισκεφθεί όλες τις ακμές τελειώσαμε. Διαφορετικά, πρέπει να βρούμε νέο κύκλο που ξεκινάει από κόμβο που έχει προσπίπτουσες ακμές που δεν έχουμε επισκεφθεί ακόμα και να ενώσουμε τους δύο κύκλους, κ.ο.κ.. Σκεφτείτε τρόπους ώστε η εύρεση των επόμενων κύκλων να γίνεται αποδοτικά.
Περιγράψτε τον αντίστοιχο αλγόρθμο για την περίπτωση που υπάρχει διαδρομή Euler.

Άσκηση 4.* (Αλγόριθμοι γράφων / δικτύων)

Θεωρήστε το ακόλουθο "αντίστροφο" πρόβλημα ροής: σε έναν κατευθυνόμενο *ακυκλικό* γράφο (DAG) με τιμές στις ακμές του που εκφράζουν ζήτηση κάποιου αγαθού (π.χ. φυσικού αερίου), και δύο διακεκριμένους κόμβους s και t αναζητούμε την ανάθεση ροής σε κάθε ακμή, ώστε να διατηρείται η ροή σε κάθε κόμβο, εκτός των s και t (από τον s πηγάζει η ροή και καταλήγει στον t). Η ροή θα πρέπει να είναι η ελάχιστη δυνατή που να εξασφαλίζει ότι σε κάθε ακμή υπερκαλύπτεται η ζήτηση.

Προσαρμόστε κατάλληλα τον αλγόριθμο Ford-Fulkerson ώστε να λύνει το παραπάνω πρόβλημα. Διατυπώστε και αποδείξτε το αντίστοιχο θεώρημα ελάχιστης ροής - μέγιστης τομής.

Υπόδειξη: δείτε το πρόβλημα σαν πρόβλημα εύρεσης της μέγιστης ροής που πρέπει να αφαιρεθεί *από κάποια 'μεγάλη' ροή που υπερκαλύπτει τη ζήτηση (και που μπορούμε να υπολογίσουμε εύκολα).*

Άσκηση 5. (Αλγόριθμοι γράφων / δικτύων)

(α) Αποδείξτε την ορθότητα του αλγορίθμου Bellman-Ford όταν δεν υπάρχουν αρνητικοί κύκλοι στον γράφο.

(β) Εξηγήστε τι ακριβώς πρέπει να κάνει ο αλγόριθμος Bellman-Ford για να εντοπίζει αρνητικούς κύκλους; Αποδείξτε την ορθότητα της προτεινόμενης τροποποίησης.

Υπόδειξη: (α) Δείτε προσεκτικά τη λειτουργία του αλγόριθμου Bellman-Ford. Παρατηρήστε ότι μετά το 1ο βήμα, αν μία κορυφή u συνδέεται με ακμή με την αρχική κορυφή s, δηλαδή αν υπάρχει διαδρομή μήκους 1 από τον s στον u, τότε το $dist[u]$ είναι ίσο με το βάρος της ελαφρύτερης διαδρομής απο την s στην u, η οποία περιέχει ακριβώς μία ακμή.

Παρατηρήστε ακόμη ότι μετά το 2ο βήμα το $dist[u]$ είναι ίσο με το βάρος της ελαφρύτερης διαδρομής από την s στη u που περιέχει το πολύ 2 ακμές (αν υπάρχει τέτοια διαδρομή).

Χρησιμοποιήστε τις παραπάνω παρατηρήσεις και επαγωγή για να δείτε τί συμβαίνει στο 3ο, 4ο, ... , κ-οστό βήμα....

(β) Την τροποποίηση θα την βρείτε εύκολα στις σημειώσεις. Για να δείτε πώς δουλεύει κάνετε μερικά παραδείγματα. Προσπαθήστε να εκφράσετε αυτό που συμβαίνει με έναν γενικό τρόπο.

Άσκηση 6. (Δυναμικός προγραμματισμός)

Σε ένα εργοστάσιο υαλικών θέλουν να μετρήσουν την αντοχή των κρυστάλλινων ποτηριών που κατασκευάζουν. Συγκεκριμένα, θέλουν να βρουν ποιο ακριβώς (με ακρίβεια εκατοστού) είναι το μέγιστο ύψος από το οποίο μπορούν να πέσουν τα ποτήρια τους χωρίς να σπάσουν. Η διεύθυνση του εργοστασίου μπορεί να διαθέσει έως ένα πλήθος ποτηριών k για τις δοκιμές (δηλ. δέχεται να καταστραφούν k ποτήρια) και θέλει το μέγιστο πλήθος δοκιμών που θα χρειαστούν να είναι όσο το δυνατόν μικρότερο. Επιπλέον, είναι γνωστό ότι τα ποτήρια σίγουρα σπάνε από ένα δεδομένο ύψος n εκατοστών και πάνω και ότι όλα τα ποτήρια είναι της ίδιας ακριβώς αντοχής.

Με βάση τα παραπάνω δεδομένα σχεδιάστε αλγόριθμο που να βρίσκει την βέλτιστη σειρά δοκιμών, με είσοδο τα n και k έτσι ώστε να ελαχιστοποιείται το πλήθος των δοκιμών στη χειρότερη περίπτωση. Ειδικότερα:

(α) Βρείτε την βέλτιστη σειρά δοκιμών για $n = 100$ και $k = 1$, καθώς και για $n = 100$ και $k = 2$. Ποιό είναι το μέγιστο πλήθος δοκιμών που μπορεί να χρειαστεί η λύση σας;

Υπόδειξη: Για $k = 2$, μια πρώτη σκέψη είναι να δοκιμάσουμε με το πρώτο ποτήρι σε ύψος 50 και αν σπάσει να χρησιμοποίησουμε το δεύτερο ποτήρι από το 1 μέχρι το 49. Αν δεν σπάσει, να χρησιμοποίησουμε το ίδιο σε ύψος 75 κ.λπ. Αυτό δίνει 50 δοκιμές στη χειρότερη περίπτωση.
Μια άλλη ιδέα είναι να δοκιμάζουμε κάθε 10 μονάδες ύψους όσο δεν σπάει το πρώτο ποτήρι. Μόλις σπάσει, έχουμε να κάνουμε το πολύ άλλες 9 δοκιμές. Αυτό δίνει 19 δοκιμές στη χειρότερη περίπτωση.
Μήπως υπάρχει και καλύτερη λύση;
Μια ιδέα είναι να δοκιμάσουμε το πρώτο ποτήρι σε ένα ύψος h_1 (άγνωστο προς το παρόν) και αν σπάσει, με το δεύτερο θα χρειαστεί να κάνουμε $h_1 - 1$ το πολύ ακόμα δοκιμές, συνολικά h_1 δοκιμές το πολύ. Αν δεν σπάσει, θα βρούμε ένα δεύτερο ύψος h_2 (που να σχετίζεται κάπως με το h_1) και να δοκιμάσουμε από εκεί να ρίξουμε το πρώτο ποτήρι, ώστε να εξασφαλίσουμε ότι και πάλι αν σπάσει θα γίνουν h_1 δοκιμές το πολύ. Ποιο είναι αυτό το ύψος; Συνεχίζουμε από εκεί και πέρα με αντίστοιχο τρόπο την ανάλυσή μας. Το ερώτημα είναι, ποιό είναι το h_1 (και ακολούθως το $h_2, h_3, \ldots$) για το οποίο το πλήθος των δοκιμών στη χειρότερη περίπτωση ελαχιστοποιείται; Το πρόβλημα μπορεί να λυθεί, με λίγη σκέψη, για την περίπτωση $k = 2$, και να βρεθεί μάλιστα και μαθηματικός τύπος. Παρακάτω θα επιχειρήσουμε μια μάλλον απλούστερη λύση με δυναμικό προγραμματισμό που μπορεί να μας δώσει ιδέες και για τη γενική περίπτωση.

(β) Προσπαθήστε να γενικεύσετε για οποιαδήποτε n και k (*υπόδειξη*: προσπαθήστε αρχικά να εκφράσετε τη βέλτιστη λύση της περίπτωσης οποιουδήποτε n και $k = 2$ αναδρομικά, χρησιμοποιώντας λύσεις της περίπτωσης $k = 1$ και $k = 2$ για μικρότερα n).

Υπόδειξη: Όπως είναι φανερό, έχουμε μπροστά μας ένα πρόβλημα βελτιστοποίησης, του οποίου η λύση μπορεί να εκφραστεί μέσω της λύσης υποπροβλημάτων (εισόδων μικρότερου μεγέθους). Αν για παράδειγμα ορίσουμε ως $F(n, 1)$ το ελάχιστο πλήθος δοκιμών για ύψος n και ένα αντικείμενο, ισχύει

$F(n,1) = n$ για οποιαδήποτε τιμή του n (γιατί;). Πώς μπορώ με βάση τις τιμές $F(n,1)$ να υπολογίσω το $F(n,2)$, δηλαδή το ελάχιστο πλήθος δοκιμών για ύψος n και δύο αντικείμενα, για οποιοδήποτε n; Μήπως θα βοηθήσει αν ξέρω το $F(i,2)$ για κάθε $i < n$; Σκεφτείτε ότι ουσιαστικά χρειάζεται να συμπληρώσουμε έναν πίνακα $n \times 2$ όπου ξέρουμε όλες τις τιμές της πρώτης γραμμής (τα $F(i,1)$) και την τιμή $F(1,2) = 1$ (για ύψος 1 αρκεί πάντα μια δοκιμή). Χρειαζόμαστε απλώς μια κατάλληλη αναδρομική σχέση ώστε να γεμίσουμε τον πίνακα με συστηματικό τρόπο. Η σχέση προκύπτει αν σκεφτούμε τι θα συμβεί αν κάνουμε την 1η δοκιμή σε ύψος h_1 και μελετήσουμε τις εξής δύο περιπτώσεις: είτε σπάει είτε δεν σπάει το ποτήρι στην 1η δοκιμή. Πώς συνεχίζω; η χειρότερη από τις δύο περιπτώσεις μου δίνει την λύση που προκύπτει αν κάνω την 1η δοκιμή στο ύψος h_1. Για να βρω την καλύτερη λύση συνολικά, δεν έχω παρά να διαλέξω την λύση με την μικρότερη τιμή ανάμεσα στις λύσεις που θα πάρω εξετάζοντας όλες τις πιθανές τιμές του h_1.
Η γενίκευση σε οποιαδήποτε n και k προκύπτει άμεσα αν καταφέρουμε να εκφράσουμε αντίστοιχα το $F(n,k)$ συναρτήσει μικρότερων υποπροβλημάτων. Μπορούμε να το πετύχουμε αν σκεφτούμε με εντελώς ανάλογο τρόπο.
Ας σημειωθεί ότι υπάρχουν και άλλοι τρόποι επίλυσης του προβλήματος με δυναμικό προγραμματισμό.
(γ)* Υλοποιήστε τον αλγόριθμό σας σε γλώσσα προγραμματισμού της επιλογής σας.

Άσκηση 7. (Αναγωγές προς επίλυση προβλημάτων)

Αντί για το πρόβλημα του Λαβυρίνθου που δόθηκε στη σειρά ασκήσεων, θα λύσουμε ένα παρόμοιο:

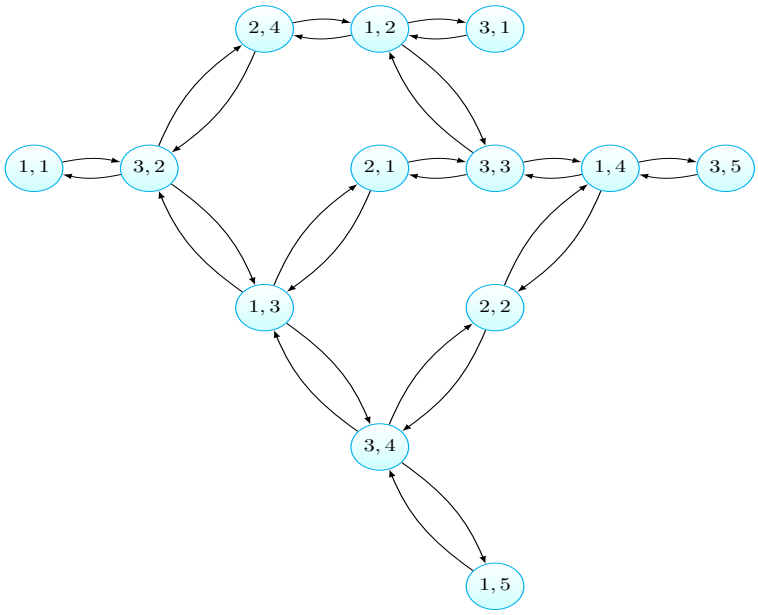
Ορίζουμε το πρόβλημα της Διαδρομής Ίππου ως εξής: είσοδος είναι οι διαστάσεις ορθογώνιας σκακιέρας n, m, καθώς και κάποια απαγορευμένα τετράγωνα που δίνονται σαν ζεύγη (i, j). Σκοπός είναι να βρούμε αν ένα άλογο (ίππος) που ξεκινά από το αρχικό τετράγωνο $(1, 1)$ μπορεί να φτάσει στο τελικό τετράγωνο (n, m) χωρίς να χρησιμοποιήσει απαγορευμένα τετράγωνα, και αν ναι με ποιον τρόπο. (Κίνηση αλόγου: 2 τετράγωνα προς μία κατεύθυνση, οριζόντια ή κάθετα, και μετά 1 τετράγωνο αριστερά ή δεξιά, κάθετα στην αρχική κατεύθυνση).

(α) Έχει λύση το πρόβλημα για τη σκακιέρα 3×5 με απαγορευμένα τετράγωνα τα $(2, 3)$ και $(2, 5)$; Αν ναι, ποιά;

Λύση: Η σκακιέρα 3×5 με απαγορευμένα τετράγωνα τα $(2, 3)$ και $(2, 5)$:

$1,1$	$1,2$	$1,3$	$1,4$	$1,5$
$2,1$	$2,2$	$\times$	$2,4$	$\times$
$3,1$	$3,2$	$3,3$	$3,4$	$3,5$

Κατασκευάζουμε τον κατευθυνόμενο γράφο $G_{3,5}$, έτσι ώστε κάθε κορυφή του να αντιστοιχεί σε κάποιο τετράγωνο της σκακιέρας και κάθε ακμή του σε μία επιτρεπόμενη κίνηση του αλόγου.



Στον παραπάνω γράφο περιέχονται μόνο οι κορυφές που αντιστοιχούν σε τετράγωνα προσβάσιμα από το τετράγωνο $(1, 1)$. Παρατηρήστε ότι τα απαγορευμένα τετράγωνα δε θεωρούνται προσβάσιμα.

Οποιοδήποτε μονοπάτι του $G_{3,5}$ ξεκινάει από την κορυφή $\boxed{1,1}$ και καταλήγει στην κορυφή $\boxed{3,5}$ αντιστοιχεί σε λύση του προβλήματος Διαδρομής Ίππου για τη σκακιέρα 3×5.
Για παράδειγμα, το μονοπάτι:

$$\boxed{1,1} \longrightarrow \boxed{3,2} \longrightarrow \boxed{2,4} \longrightarrow \boxed{1,2} \longrightarrow \boxed{3,3} \longrightarrow \boxed{1,4} \longrightarrow \boxed{3,5}$$

αντιστοιχεί σε μια ακολουθία έξι διαδοχικών κινήσεων του αλόγου από το τετράγωνο $(1, 1)$ στο τετράγωνο $(3, 5)$.

(β) Περιγράψτε με σαφήνεια έναν όσο το δυνατόν πιο αποδοτικό αλγόριθμο που να επιλύει το πρόβλημα της Διαδρομής Ίππου στη γενική περίπτωση. Ποια είναι η πολυπλοκότητα του αλγορίθμου σας σε σχέση με τα n, m και το πλήθος k των απαγορευμένων τετραγώνων;

Υπόδειξη: προσπαθήστε να μετατρέψετε το πρόβλημα σε κάποιο γνωστό σας πρόβλημα για το οποίο υπάρχει αποδοτικός αλγόριθμος. Μπορείτε να χρησιμοποιήσετε γνωστούς αλγορίθμους σαν υπορουτίνες, εάν όμως χρειάζονται τροποποιήσεις θα πρέπει να τις αναφέρετε συνοπτικά.

Λύση: Ανάγουμε το πρόβλημα Διαδρομής Ίππου στο Πρόβλημα Προσβασιμότητας σε γράφο.
Δεδομένης σκακιέρας $n \times m$ με k απαγορευμένα τετράγωνα κατασκευάζουμε τον κατευθυνόμενο γράφο $G_{n,m,k} = (V, E)$. Στον $G_{n,m,k}$ το σύνολο των κορυφών V αντιστοιχεί στα $n \cdot m - k$ μη απαγορευμένα τετράγωνα, δηλ.

$$V = \{u_{i,j} : (i, j) \textit{ μη απαγορευμένο τετράγωνο}\}.$$

Άρα $|V| = n \cdot m - k$. Το σύνολο των ακμών αντιστοιχεί στις κινήσεις του αλόγου, δηλ.

$$E = \{(u_{i,j}, u_{i',j'}) : \textit{ ο ίππος επιτρέπεται να κινηθεί από το τετράγωνο } (i, j) \textit{ στο } (i', j')\}.$$

Επειδή από κάθε τετράγωνο της σκακιέρας το άλογο μπορεί να κινηθεί το πολύ σε 8 διαφορετικά τετράγωνα, $|E| \leq 8 \cdot (n \cdot m - k)$.

Το πρόβλημα Διαδρομής Ίππου έχει λύση για τη δεδομένη σκακιέρα αν και μόνο αν στο γράφο $G_{n,m,k}$ η κορυφή $u_{n,m}$ είναι προσβάσιμη από την κορυφή $u_{1,1}$.

Επιλύουμε το Πρόβλημα Προσβασιμότητας για τον $G_{n,m,k}$ με εφαρμογή του αλγόριθμου BFS με αρχική κορυφή την $u_{1,1}$: η κορυφή $u_{n,m}$ είναι προσβάσιμη από την $u_{1,1}$ αν και μόνο αν έχει διερευνηθεί κατά την εφαρμογή του BFS. Η πολυπλοκότητα του BFS είναι $\mathcal{O}(|V| + |E|) = \mathcal{O}(n \cdot m - k)$.

Εναλλακτικά, μπορούμε να εφαρμόσουμε DFS με την ίδια πολυπλοκότητα.

(γ) Πώς πρέπει να τροποποιήσετε / συμπληρώσετε τον αλγόριθμό σας ώστε να επιστρέφει τη διαδρομή με το ελάχιστο πλήθος κινήσεων του ίππου (αν βέβαια υπάρχει διαδρομή); Αλλάζει η πολυπλοκότητα του αλγορίθμου σας; Εξηγήστε.

Υπόδειξη: Τροποποιήστε κατάλληλα τον αλγόριθμο BFS έτσι ώστε να επιστρέφει ένα μονοπάτι από την αρχική κορυφή $u_{1,1}$ στην κορυφή $u_{n,m}$ με το ελάχιστο πλήθος ακμών, αν υπάρχει τέτοιο μονοπάτι.

Σκεφτείτε γιατί σε αντίθεση με το ερώτημα (β), ο αλγόριθμος DFS δεν αποτελεί εναλλακτική επιλογή εδώ.

(δ) Τροποποιήστε (συμπληρώνοντας ή επανασχεδιάζοντας) τη διαδικασία bfs ώστε να δέχεται επιπλέον μία παράμετρο k και να δίνει στην έξοδο το πλήθος των κόμβων που έχουν απόσταση από τον αρχικό κόμβο μικρότερη ή ίση του k.

Υπόδειξη: Τροποποιήστε κατάλληλα τον αλγόριθμο BFS έτσι ώστε να αυξάνει ένα μετρητή κάθε φορά που διερευνά κορυφή που βρίσκεται σε απόσταση μικρότερη ή ίση με k από την αρχική κορυφή.

Άσκηση 8. (Αναγωγές προς απόδειξη δυσκολίας)

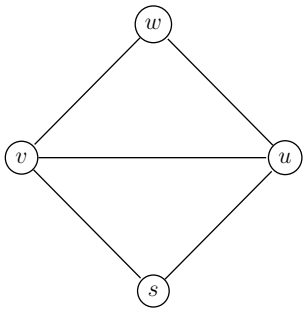
Το πρόβλημα **Hamilton Path** ορίζεται παρόμοια με το **Hamilton Cycle** με τη διαφορά ότι δίνονται επιπλέον δύο κόμβοι s, t και ζητείται να βρεθεί *μονοπάτι* από τον κόμβο s στον κόμβο t που να περνάει από κάθε κόμβο του γράφου ακριβώς μία φορά.

Θεωρώντας γνωστό ότι το πρόβλημα **Hamilton Cycle** είναι NP-πλήρες αποδείξτε ότι και το **Hamilton Path** είναι NP-πλήρες.

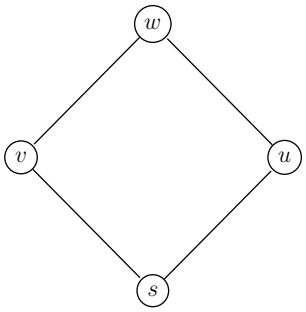
Υπόδειξη: βρείτε έναν τρόπο μετατροπής κάθε *στιγμιοτύπου του **Hamilton Cycle** σε* κάποιο *στιγμιότυπο του **Hamilton Path** έτσι ώστε να υπάρχει κύκλος Hamilton στο αρχικό στιγμιότυπο αν και μόνο αν υπάρχει μονοπάτι Hamilton στο τελικό στιγμιότυπο.*

*Ένα στιγμιότυπο του **Hamilton Cycle** είναι ένας γράφος $G = (V, E)$. Έστω ότι επιλέγουμε τυχαία μία ακμή (u, v) του G και την αφαιρούμε από το G. Παρατηρήστε ότι αν ο γράφος $G \backslash (u, v)$ έχει μονοπάτι Hamilton από την κορυφή u στην κορυφή v, τότε ο γράφος G έχει κύκλο Hamilton. Το αντίστροφο όμως δεν ισχύει πάντα.*

Για παράδειγμα, ενώ ο παρακάτω γράφος G έχει κύκλο Hamilton, ο γράφος $G \backslash (u, v)$ δεν έχει μονοπάτι Hamilton από την κορυφή u στην κορυφή v.



Σχήμα 1: Ο γράφος G, στιγμιότυπο του προβλήματος **Hamilton Cycle**



Σχήμα 2: Ο γράφος $G\backslash(u,v)$

*Ενώ με αυτή την απλή μετατροπή δεν μπορούμε να ανάγουμε το **Hamilton Cycle** στο **Hamilton Path**, μπορούμε με κατάλληλη τροποποίησή της να αποκτήσουμε τη ζητούμενη αναγωγή.*

*Εναλλακτική αναγωγή από το πρόβλημα **Hamilton Cycle** στο **Hamilton Path** μπορεί να γίνει και με διαφορετική ιδέα από την παραπάνω! Ας σημειωθεί ότι έχετε "δικαίωμα" να προσθέσετε κόμβους ή/και ακμές, αρκεί η κατασκευή να γίνεται σχετικά γρήγορα (τυπικά: να είναι πολυωνυμικού χρόνου σε σχέση με το αρχικό στιγμιότυπο).*

Άσκηση 9. (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 μεγάλη πιθανότητα (για τους περισσότερους αριθμούς $\geq 1/2$). Για να αυξήσουμε σημαντικά την πιθανότητα μπορούμε να επαναλάβουμε μερικές φορές (τυπικά 40 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 έως και 1000 ψηφίων, π.χ. να μπορεί να ελέγξει αν ο αριθμός 2^x-1 είναι πρώτος για κάθε $x \leq 1000$. Δοκιμάστε την για $x = 100i, 1 \leq i \leq 10$.

Δοκιμάστε τη συνάρτησή σας με διάφορους γνωστούς πρώτους αριθμούς, π.χ. δείτε: http://en.wikipedia.org/wiki/List_of_prime_numbers, καθώς και με σύνθετους αριθμούς (μπορείτε να κατασκευάσετε μεγάλους σύνθετους ως γινόμενα 2 πρώτων).

Σημείωση 1: θυμηθείτε ότι το $a^{2^{1000}-2}$ έχει αστρονομικά μεγάλο πλήθος ψηφίων (δεν χωράει να γραφτεί σε ολόκληρο το σύμπαν!), ενώ το $a^{2^{1000}-2} \bmod (2^{1000}-1)$ είναι σχετικά "μικρό" (έχει μερικές εκατοντάδες 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 αριθμούς Carmichael που θα βρείτε π.χ. στη σελίδα http://de.

wikibooks.org/wiki/Pseudoprimzahlen:_Tabelle_Carmichael-Zahlen.

(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 40 επαναλήψεις έχουμε αμελητέα πιθανότητα λάθους για κάθε αριθμό εισόδου). Δοκιμάστε τον με αριθμούς Carmichael.

Υπόδειξη: χρησιμοποιήστε κάποια γλώσσα που να υποστηρίζει απευθείας πράξεις με αριθμούς 100δων ψηφίων, όπως η Python ή η Haskell. Εναλλακτικά, χρησιμοποιήστε κατάλληλες βιβλιοθήκες. Προσέξτε ότι η πράξη mod *θα πρέπει να γίνεται σε κάθε βήμα ώστε να μην μεγαλώνουν πολύ οι αριθμοί. Για το (β): έχετε υπόψη ότι και στους αριθμούς Carmichael, αν επιλεγεί αριθμός* a *που δεν είναι σχετικά πρώτος με τον* n *τότε ο έλεγχος Fermat θα δώσει τη σωστή απάντηση επειδή για τέτοιους αριθμούς δεν ισχύει* $a^{n-1} \bmod n = 1$. *Για πολύ μεγάλους όμως αριθμούς Carmichael η πιθανότητα αυτή είναι πάρα πολύ μικρή.*