Κ08 Δομές Δεδομένων και Τεχνικές Προγραμματισμού

Διδάσκων: Μανόλης Κουμπαράκης

Εαρινό Εξάμηνο 2020-2021

Εργασία 3

Ανακοινώθηκε στις 12 Μαΐου 2021

Προθεσμία: Παραμονή της ημερομηνίας που θα αρχίσει η εξεταστική, στις 23:59 το βράδυ

20% του συνολικού βαθμού στο μάθημα (Άριστα=295 μονάδες, υπάρχουν επίσης 50 μονάδες bonus)

Προσοχή: Πριν διαβάσετε παρακάτω, διαβάστε παρακαλώ προσεκτικά τις οδηγίες υποβολής των ασκήσεων που βρίσκονται στην ιστοσελίδα http://cgi.di.uoa.gr/~k08/homework.html, ειδικά ότι αναφέρεται στο github και τα σχετικά αρχεία.

Απορίες: Αν έχετε απορίες σχετικά με την εργασία, ρωτήστε στο piazza και όχι στέλνοντας e-mail στον διδάσκοντα. Τέτοια e-mails ΔΕΝ θα λαμβάνουν απάντηση.

Κώδικας: Ο κώδικας που παρουσιάζουμε στην Ενότητα 15 των διαλέξεων του μαθήματος και θα χρειαστείτε για την εργασία αυτή βρίσκεται στο παρακάτω private repository του classroom του μαθήματος: https://github.com/artioi-k08/2021-ergasia-3. Για να συνδεθείτε στο repository αυτό, θα πρέπει να χρησιμοποιήσετε το λινκ https://classroom.github.com/a/bHqxMIvd .

Όταν συνδεθείτε, θα μπορείτε να δείτε το προσωπικό σας repository https://github.com/artioi-k08/2021-ergasia-3-<github-your-username> στο οποίο θα δουλέψετε για την Εργασία 3.

Μετά τη σύνδεση σας, στο προσωπικό σας repository, θα βρείτε τον κώδικα που καλύπτει την Ενότητα 15 στο φάκελο enotita15. Θα βρείτε επίσης ένα φάκελο solutions-ergasia3 με υπο-φακέλους question1, question2, …, question9 στους

οποίους θα πρέπει να γράψετε τον κώδικα ή τις απαντήσεις σας για τα 9 ερωτήματα της εργασίας που θα βρείτε παρακάτω. Παρακαλώ τηρείστε ευλαβικά αυτήν την οργάνωση αλλιώς θα χάσετε 20% του συνολικού βαθμού κατά την βαθμολόγηση. Για τα θεωρητικά ερωτήματα, οι απαντήσεις πρέπει να είναι σε ένα αρχείο τύπου pdf.

Κύριο πρόγραμμα: Σε όσες από τις παρακάτω ασκήσεις είναι προγραμματιστικές, θα πρέπει να υλοποιήσετε και ένα κύριο πρόγραμμα (συνάρτηση `main`) το οποίο θα διαβάζει τα δεδομένα εισόδου, θα επιδεικνύει τη λειτουργικότητα της συνάρτησης σας για κατάλληλα επιλεγμένες εισόδους, και θα πείθει τον βαθμολογητή ώστε να σας βαθμολογήσει με τον υψηλότερο δυνατό βαθμό.

1. Σχεδιάστε το B-δένδρο που προκύπτει αν εισάγουμε τα κλειδιά 88, 2, 86, 6, 4, 82, 70, 20, 66, 30, 90, 59, 25, 72, 64, 77, 39, 12 με αυτή τη σειρά σε ένα αρχικά κενό B-δένδρο τάξεως 7. Μετά σχεδιάστε το B-δένδρο που θα προκύψει αν διαγράψουμε όλα τα στοιχεία της ρίζας από το μικρότερο προς το μεγαλύτερο. Όλα τα βήματα που θα κάνετε πρέπει να δειχθούν αναλυτικά.

   (10+5=15 μονάδες)

2. Δώστε ένα παράδειγμα κατευθυνόμενου γράφου για τον οποίο ο αριθμός των δένδρων DFS τα οποία μπορούμε να έχουμε είναι διαφορετικός ανάλογα με το ποια είναι η αρχική κορυφή της DFS αναζήτησης που κάνουμε. Προσπαθήστε ο γράφος που θα δώσετε να έχει όσο το δυνατόν λιγότερες ακμές.

   (5 μονάδες)

3. Θεωρήστε ένα μη κατευθυνόμενο γράφο $G = (V, E)$ με σύνολο κορυφών $V = \{1, 2, \dots, 7\}$ και σύνολο ακμών $E = \{(1,2), (1,6), (6,5), (2,7), (2,3), (7,5), (7,4), (5,4), (3,4)\}$. Να σχεδιάσετε τον παραπάνω γράφο. Να δώσετε την ακολουθία κορυφών που προκύπτει αν διασχίσουμε το γράφο χρησιμοποιώντας τους αλγόριθμους DFS (με αναδρομική κλήση) και BFS από τις διαφάνειες της Ενότητας 15. Να υποθέσετε ότι ο αλγόριθμος ξεκινάει από τον κόμβο 1 και οι κορυφές

αποθηκεύονται στον σχετικό πίνακα και στις λίστες γειτνίασης με αριθμητική σειρά. Να σχεδιάσετε το πρώτα κατά βάθος δένδρο επικάλυψης που αντιστοιχεί στην παραπάνω DFS διάσχιση του γράφου. Να δώσετε τις ακμές δένδρου και τις ακμές οπισθοχώρησης για την παραπάνω DFS διάσχιση.

(5+5+5+5+5=25 μονάδες)

4. Θεωρήστε τον κατευθυνόμενο γράφο $G = (V, E)$ με σύνολο κορυφών $V = \{0, 1, 2, \dots, 9\}$ και σύνολο ακμών $E = \{ (1,2), (1,4), (2,3), (4,3), (4,6), (6,5), (5,7), (6,8), (5,8), (8,7), (9,8), (0,5)\}$.
   Να σχεδιάσετε τον δοσμένο γράφο και να δώσετε μια τοπολογική ταξινόμηση του (topological sort).

   (5+10=15 μονάδες)
5. Θεωρήστε τον κατευθυνόμενο γράφο $G = (V, E)$ με σύνολο κορυφών $V = \{1, 2, \dots, 12\}$ και σύνολο ακμών $E = \{ (1,2), (2,3), (2,4), (4,5), (2,5), (5,2), (3,6), (6,3), (5,6), (6,8), (5,7), (7,8), (8,7), (9,7), (7,10), (10,9), (10,11), (11,12), (12,10) \}$
   Να σχεδιάσετε τον δοσμένο γράφο και να δώσετε τις ισχυρά συνεκτικές συνιστώσες του (strongly connected components).

   (5+10=15 μονάδες)

6. Θεωρήστε τον κατευθυνόμενο γράφο με βάρη στις ακμές $G = (V, E)$ με σύνολο κορυφών $V = \{a, b, c, d, e, f, g, h\}$ και σύνολο ακμών $E = \{ (a, b, 1), (a, e, 10), (b, e, 8), (e, d, 15), (e, f, 5), (e, c, 2), (c, f, 2), (e, g, 3), (g, f, 4), (f, h, 5)\}$.
   Στο σύνολο ακμών συμβολίζουμε μια ακμή από την κορυφή x στην κορυφή y με βάρος w με την τριάδα (x, y, w). Να σχεδιάσετε τον δοσμένο γράφο. Μετά να εκτελέσετε τον αλγόριθμο του Dijkstra στον γράφο αυτό για να λύσετε το πρόβλημα του συντομότερου μονοπατιού κοινής αφετηρίας ξεκινώντας από την κορυφή a.

   (5+10=15 μονάδες)

7. Θεωρήστε τον κώδικα της Ενότητας 15 για μη κατευθυνόμενους γράφους με χρήση λιστών γειτνίασης που σας δόθηκε στο repository της εργασίας. Στην

άσκηση αυτή θα οργανώσουμε και θα εμπλουτίσουμε τον κώδικα αυτό ώστε να ορίζει και να υλοποιεί ένα αφαιρετικό τύπο δεδομένων `UndirectedGraph` με τις εξής λειτουργίες που υλοποιούνται από κατάλληλες συναρτήσεις της C:

- `Initialize`. Η συνάρτηση αυτή αρχικοποιεί το γράφο που παίρνει σαν όρισμα.
- `InsertEdge`. Η συνάρτηση αυτή εισάγει μια ακμή σε ένα γράφο. Η ακμή και ο γράφος είναι ορίσματα της συνάρτησης.
- `ShowGraph`. Η συνάρτηση αυτή εκτυπώνει ένα γράφο που δίνεται σαν όρισμα χρησιμοποιώντας την αναπαράσταση των λιστών γειτνίασης όπως κάνουμε στις διαφάνειες της Ενότητας 15 (π.χ., διαφάνειες 64 και 65).
- `BreadthFirstSearch`. Η συνάρτηση αυτή παίρνει σαν όρισμα ένα γράφο και μια κορυφή του, κάνει μια πρώτα κατά πλάτος διάσχιση του γράφου ξεκινώντας από την κορυφή που δόθηκε σαν όρισμα, και εκτυπώνει τις κορυφές που επισκέφθηκε με τη σειρά που τις επισκέφθηκε. Επιπλέον, εκτυπώνει όλες τις ακμές που διασχίζει και τις ταξινομεί σε ακμές δένδρου και εγκάρσιες ακμές. Η υλοποίηση αυτής της συνάρτησης θα πρέπει να επεκτείνει τον σχετικό κώδικα που σας δόθηκε στο αρχείο `bfs.c`.
- `IsConnected`. Η συνάρτηση αυτή παίρνει σαν είσοδο ένα γράφο και ελέγχει αν είναι συνεκτικός χρησιμοποιώντας πρώτα κατά πλάτος διάσχιση.
- `ShortestPaths`. Η συνάρτηση αυτή παίρνει σαν όρισμα ένα γράφο και μια κορυφή του και επιστρέφει τα συντομότερα μονοπάτια (αυτά με τις λιγότερες ακμές) σε όλες τις άλλες κορυφές ή μια ένδειξη ότι τέτοιο μονοπάτι δεν υπάρχει χρησιμοποιώντας πρώτα κατά πλάτος διάσχιση.
- `ConnectedComponents`. Η συνάρτηση αυτή παίρνει σαν όρισμα ένα γράφο και υπολογίζει τις συνεκτικές συνιστώσες του γράφου χρησιμοποιώντας πρώτα κατά πλάτος διάσχιση.

Εκτός από τις παραπάνω συναρτήσεις, θα πρέπει να γράψετε και μια συνάρτηση `main` η οποία θα διαβάζει ένα γράφο από την είσοδο και θα εκτελεί τις διάφορες λειτουργίες που περιγράψαμε. Μπορείτε να υποθέσετε ότι ο γράφος δίνεται σε ένα αρχείο εισόδου το οποίο περιέχει στην πρώτη γραμμή του τον αριθμό κόμβων του γράφου, και σε κάθε επόμενη γραμμή την αναπαράσταση μιας ακμής (x, y) με την απλούστερη μορφή `x-y`.

Το πρόγραμμα που θα παραδώσετε θα πρέπει να έχει οργανωθεί σαν ένα module της C που υλοποιεί τον αφηρημένο τύπο δεδομένων και κάνει καλή απόκρυψη πληροφορίας. Θα πρέπει να υπάρχει και το αντίστοιχο makefile.

(10+10+10+30+10+30+30=130 μονάδες)

8. Στην άσκηση αυτή θα ορίσουμε τον αφαιρετικό τύπο δεδομένων `WeightedUndirectedGraph` με τις εξής λειτουργίες που υλοποιούνται από κατάλληλες συναρτήσεις:
    - `Initialize`, `InsertEdge` και `ShowGraph` με αντίστοιχη λειτουργικότητα με τις συναρτήσεις του προηγούμενου ερωτήματος.
    - `MinimumSpanningTree`. Η συνάρτηση αυτή υλοποιεί τον αλγόριθμο του Kruskal για τον υπολογισμό του ελάχιστου δέντρου επικάλυψης (minimum spanning tree) ενός δοσμένου μη κατευθυνόμενου γράφου.

    Εκτός από τις παραπάνω συναρτήσεις, θα πρέπει να γράψετε και μια συνάρτηση `main` η οποία θα διαβάζει ένα μη κατευθυνόμενο γράφο με βάρη από την είσοδο και θα εκτελεί τις διάφορες λειτουργίες που περιγράψαμε. Μπορείτε να υποθέσετε ότι ο γράφος δίνεται σε ένα αρχείο εισόδου το οποίο περιέχει στην πρώτη γραμμή του τον αριθμό κόμβων του γράφου, και σε κάθε επόμενη γραμμή την αναπαράσταση μιας ακμής (x, y, w) (όπου x και y είναι κορυφές και w είναι το βάρος της ακμής (x, y)) με την απλούστερη μορφή `x-y-w`.

Το πρόγραμμα που θα παραδώσετε θα πρέπει να έχει οργανωθεί σαν ένα module της C που υλοποιεί τον αφηρημένο τύπο δεδομένων. Θα πρέπει να υπάρχει και το αντίστοιχο makefile.

(10+50=60 μονάδες)

9. Να υπολογίσετε την υπολογιστική πολυπλοκότητα χειρίστης περίπτωσης του αλγόριθμου του Kruskal που υλοποιήσατε στο προηγούμενο ερώτημα (δηλ. θα υπολογίσετε $O(\dots)$ με παραμέτρους e τον αριθμό των ακμών και n τον αριθμό των κορυφών του γράφου).

(15 μονάδες)

Καλή Επιτυχία!