Κ08 Δομές Δεδομένων και Τεχνικές Προγραμματισμού

Διδάσκων: Μανόλης Κουμπαράκης

Εαρινό Εξάμηνο 2024-2025

Εργασία 3

Ανακοινώθηκε στις 5 Μαΐου 2025

Προθεσμία: 8 Ιουνίου στις 23:59 το βράδυ

1.6 μονάδες στις 10 του συνολικού βαθμού στο μάθημα (Άριστα=285 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 στις Ενότητες 14 και 16 των διαλέξεων του μαθήματος και θα χρειαστείτε για την εργασία αυτή βρίσκεται στο παρακάτω private repository του classroom του μαθήματος: https://github.com/artioi-k08/2025-ergasia3. Για να συνδεθείτε στο repository αυτό, θα πρέπει να χρησιμοποιήσετε το link https://classroom.github.com/a/IT5cPrR- .

Όταν συνδεθείτε, θα μπορείτε να δείτε το προσωπικό σας repository https://github.com/artioi-k08/2025-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 τις Ενότητες 14 και 16 στο φάκελο code-from-lectures. Θα βρείτε επίσης ένα φάκελο solutions-ergasia3 με υπο-φακέλους question1, question2, …,

question8 στους οποίους θα πρέπει να γράψετε τον κώδικα ή τις απαντήσεις σας για τα 8 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. 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)

2. Διαβάστε προσεκτικά τις σελίδες 81-84 των διαφανειών της Ενότητας 16 (Γράφοι). Μετά να κάνετε την άσκηση που προτείνουμε στην σελίδα 84 για την περίπτωση που το container του αλγόριθμου `GraphSearch` της σελίδας 81 είναι queue (δηλαδή, ο `GraphSearch` γίνεται BFS). Η βασική ιδέα της άσκησης αυτής είναι να εξηγήσετε με ένα παράδειγμα γιατί αυτά που λέμε στις διαφάνειες 82 και 83 ισχύουν. Προσπαθήστε ο γράφος που θα δώσετε σαν παράδειγμα να είναι όσο το δυνατόν πιο μικρός.

   (10 μονάδες)

3. Θεωρήστε τον κώδικα της Ενότητας 16 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 16 (διαφάνειες 67 έως 72).
- `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. Επιπλέον, εκτυπώνει όλες τις ακμές που διασχίζει και τις ταξινομεί σε ακμές δένδρου και εγκάρσιες ακμές. Η διαδικασία θα πρέπει να γίνει για όλες τις ακμές του γράφου. 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)

4. Δώστε ένα παράδειγμα μη κατευθυνόμενου γράφου και 2 διαφορετικά δένδρα επικάλυψης του, που προκύπτουν από 2 διαφορετικές εκτελέσεις του αλγόριθμου BFS. Εξηγήστε ποιες είναι οι ακμές δένδρου και ποιες οι εγκάρσιες ακμές στα δένδρα αυτά. Το παράδειγμα σας πρέπει να είναι το μικρότερο δυνατό.

(5 μονάδες)

5. 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)

6. 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)

7. Να οργανώσετε τον κώδικα της Ενότητας 14 (λίστες παράλειψης) σε μια ενότητα (module) της C που να υλοποιεί τον αφαιρετικό τύπο δεδομένων Λίστα Παράλειψης (skip list). Μετά να γράψετε μια συνάρτηση η οποία να συγχωνεύει δύο λίστες παράλειψης σε μια. Ο αλγόριθμος που θα χρειαστεί να υλοποιήσετε δίνεται στο σχήμα 7, σελίδα 13 του άρθρου «A skip list cookbook» που σας δίνουμε μαζί με την εργασία. Τέλος, να γράψετε μια main που να επιδεικνύει τη συμπεριφορά των διαφόρων συναρτήσεων της ενότητας.

(10+50=60 μονάδες)

8. Στην άσκηση αυτή θα υλοποιήσουμε μια απλή έκδοση του Chord, του πιο γνωστού κατανεμημένου πίνακα κατακερματισμού (distributed hash table, DHT). Αν και πρόκειται για ένα κατανεμημένο δίκτυο, εμείς θα το υλοποιήσουμε σε ένα υπολογιστή. Το άρθρο που παρουσιάζει το Chord μπορεί να βρεθεί στον σύνδεσμο https://pdos.csail.mit.edu/papers/ton:chord/paper-ton.pdf. Χρειάζεται να διαβάσετε μέχρι τη σελίδα 6 ώστε να καταλάβετε τι είναι το Chord, πως οργανώνονται οι κόμβοι του, πως εισάγουμε ένα ζευγάρι `(key,value)` στο δίκτυο και πως βρίσκουμε την τιμή που αντιστοιχεί σε ένα δοσμένο κλειδί (lookup function). Επίσης υπάρχουν πολλές λεπτομερείς παρουσιάσεις και βίντεο για το Chord στον Παγκόσμιο Ιστό που μπορείτε να δείτε. Μετά πρέπει να ορίσετε ένα αφαιρετικό τύπο δεδομένων `DHT` που υλοποιεί το Chord και έχει την παρακάτω διεπαφή:

- `void initialize(void)`. Η συνάρτηση αυτή δημιουργεί ένα δίκτυο Chord με τόσους κόμβους όση η τιμή μια σταθεράς `MAXNODENUMBER`.
- `void insert(nodeType, keyType, valueType)`. Η συνάρτηση αυτή ζητεί από ένα κόμβο με τύπο `nodeType` να εισάγει ένα ζευγάρι `(key, value)` με τύπους `(keyType, valueType)` στο δίκτυο.
- `valueType lookup(nodeType, keyType)`. Η συνάρτηση αυτή ζητεί από ένα κόμβο με τύπο `nodeType` να βρει την τιμή που είναι αποθηκευμένη στο δίκτυο για το κλειδί `key` που είναι τύπου `keyType`.
- `valueType smartLookup(nodeType, keyType)`. Η συνάρτηση αυτή είναι μια βελτιστοποίηση της προηγούμενης στην οποία οι κόμβοι του δικτύου χρησιμοποιούν ένα finger table.

Θα πρέπει να γράψετε και μια συνάρτηση `main` η οποία θα επιδεικνύει τη λειτουργία του Chord.

(50 μονάδες)

Καλή Επιτυχία!