Κ08 Δομές Δεδομένων και Τεχνικές Προγραμματισμού (Τμήμα Αρτίων)

Διδάσκων: Μανόλης Κουμπαράκης

Εαρινό Εξάμηνο 2019-2020.

Εργασία 3 (ανακοινώθηκε στις 4 Μαΐου 2020, προθεσμία παράδοσης: 7 Ιουνίου 2020, ώρα 23:59).

15% του συνολικού βαθμού στο μάθημα.

1. Σχεδιάστε το δένδρο (2,4) που προκύπτει όταν η ακολουθία κλειδιών (χαρακτήρων του Αγγλικού αλφάβητου) E A S Y Q U T I O N B L K εισάγεται σε ένα αρχικά κενό δένδρο. Μετά σχεδιάστε το δένδρο (2,4) που προκύπτει αν αφαιρεθούν όλα τα κλειδιά που βρίσκονται στη ρίζα από το μικρότερο προς το μεγαλύτερο.

   (10+5=15 μονάδες)
2. Είναι αληθής ο παρακάτω ισχυρισμός; Αν ναι, αποδείξτε τον. Αν όχι, δώστε ένα μικρό αντιπαράδειγμα.

   Ισχυρισμός: Ένα δένδρο (2,4) που αποθηκεύει ένα σύνολο κλειδιών θα έχει πάντα την ίδια δομή ανεξάρτητα από τη σειρά εισαγωγής των κλειδιών.

   (5 μονάδες)
3. 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)
4. Σχεδιάστε το κόκκινο-μαύρο δένδρο που προκύπτει αν εισάγουμε τα κλειδιά E A S Y Q U T I O N B L K σε ένα αρχικά κενό δένδρο. Μετά σχεδιάστε το κόκκινο-μαύρο δένδρο που θα προκύψει αν διαγράψουμε τη ρίζα.

   (10+5=15 μονάδες)
5. Σχεδιάστε το B-δένδρο που προκύπτει αν εισάγουμε τα κλειδιά 4, 40, 23, 50, 11, 34, 62, 78, 66, 22, 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.

(10+5=15 μονάδες)

6. Θεωρήστε ένα μη κατευθυνόμενο γράφο $G = (V, E)$ με σύνολο κορυφών $V = \{0,1,2,\dots,9\}$ και σύνολο ακμών $E = \{(3,7),(1,4),(7,8),(0,5),(5,2),(3,8),(2,9),(0,6),(4,9),(2,6),(6,4)\}$.
   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 0 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)

7. Θεωρήστε τον κατευθυνόμενο γράφο $G = (V, E)$ με σύνολο κορυφών $V = \{0,1,2,\dots,12\}$ και σύνολο ακμών $E = \{(2,3),(0,6),(0,1),(2,0),(11,12),(9,12),(9,10),(9,11),(3,5),(8,7),(5,4),(0,5),(6,4),(6,9),(7,6)\}$.
   Να σχεδιάσετε τον δοσμένο γράφο και να δώσετε μια τοπολογική ταξινόμηση του.

   (5+10=15 μονάδες)

8. Θεωρήστε τον κατευθυνόμενο γράφο $G = (V, E)$ με σύνολο κορυφών $V = \{0,1,2,\dots,12\}$ και σύνολο ακμών $E = \{(0,1),(0,5),(0,6),(2,0),(2,3),(3,5),(3,2),(4,2),(4,3),(4,11),(5,4),(6,4),(6,9),(7,6),(7,8),(8,7),(8,9),(9,10),(9,11),(10,12),(11,12),(12,9)\}$
   Να σχεδιάσετε τον δοσμένο γράφο και να δώσετε τις ισχυρά συνεκτικές συνιστώσες του.

   (5+10=15 μονάδες)

9. Θεωρήστε τον κώδικα της Ενότητας 15 για κατευθυνόμενους γράφους με χρήση λιστών γειτνίασης. Στην άσκηση αυτή θα οργανώσουμε και θα εμπλουτίσουμε τον κώδικα αυτό ώστε να ορίζει και να υλοποιεί ένα

αφαιρετικό τύπο δεδομένων `DirectedGraph` 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., στη διαφάνεια 63).
- `DepthFirst`. Η συνάρτηση αυτή κάνει μια πρώτα κατά βάθος διάσχιση του γράφου και εκτυπώνει τις κορυφές που επισκέφθηκε. Επιπλέον εκτυπώνει όλες τις ακμές που διασχίζει και τις ταξινομεί σε ακμές δένδρου, ακμές οπισθοχώρησης, ακμές προώθησης και εγκάρσιες ακμές. Η υλοποίηση αυτής της συνάρτησης θα πρέπει να επεκτείνει τον σχετικό κώδικα από τις διαφάνειες της Ενότητας 15.
- `BreadthTopSort`. Η συνάρτηση αυτή υπολογίζει μια τοπολογική ταξινόμηση του γράφου και την εκτυπώνει. Η υλοποίηση της συνάρτησης αυτής θα χρησιμοποιεί τον σχετικό κώδικα από τις διαφάνειες της Ενότητας 15.
- `GraphReverse`. Η συνάρτηση αυτή παίρνει σαν όρισμα ένα γράφο που παριστάνεται με λίστες γειτνίασης και επιστρέφει ένα νέο γράφο που έχει τις ίδιες κορυφές με τον πρώτο αλλά η κατεύθυνση κάθε ακμής στο νέο γράφο έχει αντιστραφεί.
- `StrongComponents`. Η λειτουργία αυτή υπολογίζει τις ισχυρά συνεκτικές συνιστώσες του γράφου χρησιμοποιώντας τον αλγόριθμο του Kosaraju. Προτείνουμε να χρησιμοποιήσετε ένα πίνακα `sc` για να κωδικοποιείτε σε ποια συνεκτική συνιστώσα ανήκει κάθε κορυφή ως εξής: αν `sc[v]=i` τότε η κορυφή `v` ανήκει στην συνιστώσα `i`.

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. Θα πρέπει να υπάρχει και το αντίστοιχο makefile.

(10+10+10+20+5+10+50+20=135 μονάδες)

10. Θεωρήστε τον κώδικα της Ενότητας 13 για μη κατευθυνόμενους γράφους με χρήση λιστών γειτνίασης. 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:
    - `Initialize`, `InsertEdge` και `ShowGraph` με την ίδια λειτουργικότητα της προηγούμενης άσκησης.
    - `SimplePathCheck`. Η συνάρτηση αυτή παίρνει σαν είσοδο ένα μη κατευθυνόμενο γράφο και δύο κορυφές διαφορετικές μεταξύ τους και επιστρέφει ένα απλό μονοπάτι που συνδέει αυτές τις κορυφές ή μας πληροφορεί ότι τέτοιο μονοπάτι δεν υπάρχει. Ένα μονοπάτι λέγεται **απλό** εάν όλες οι κορυφές από τις οποίες αποτελείται είναι διαφορετικές μεταξύ τους.

Γράψτε μια κατάλληλη `main` για την επίδειξη των δυνατοτήτων του προγράμματος σας. Ο γράφος θα δίνεται σε ένα αρχείο χρησιμοποιώντας τις συμβάσεις του προηγούμενου ερωτήματος. Το πρόγραμμα που θα παραδώσετε θα πρέπει να είναι οργανωμένο όπως και το πρόγραμμα του προηγούμενου ερωτήματος.

(10+20=30 μονάδες)

11. 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 των Prim-Jarnik για τον υπολογισμό του ελάχιστου δέντρου επικάλυψης 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)

12. Να υπολογίσετε την υπολογιστική πολυπλοκότητα χειρίστης περίπτωσης του αλγόριθμου Prim-Jarnik 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)

Παρατηρήσεις:

- Τα προγράμματα σας πρέπει να «τρέχουν» στους υπολογιστές του Τμήματος με το λειτουργικό linux αφού μεταφραστούν με τον μεταγλωττιστή gcc.

Πως να παραδώσετε τις λύσεις σας: Οι λύσεις θα πρέπει να σταλούν στο e-mail ddproj@di.uoa.gr μέχρι την προθεσμία παράδοσης και να είναι οργανωμένες ως εξής. Θα στείλετε ακριβώς ένα συμπιεσμένο αρχείο. Η λύση κάθε προβλήματος που απαιτεί υλοποίηση θα είναι σε χωριστό directory το οποίο θα περιλαμβάνει τα source files που χρειάζονται, και ένα Makefile το οποίο θα μπορεί να χρησιμοποιηθεί για να μεταγλωττιστούν τα αρχεία σας και να παραχθεί το αντίστοιχο executable. Θα πρέπει να υποβάλλετε και ένα αρχείο pdf στο οποίο θα περιέχονται οι απαντήσεις των θεωρητικών ερωτημάτων και το οποίο, για τα προγραμματιστικά ερωτήματα, θα περιέχει όσο documentation χρειάζεται ώστε οι βαθμολογητές να κατανοήσουν πλήρως τη λύση σας και να τη βαθμολογήσουν ανάλογα. Αυτό θα πρέπει να γίνει ανεξάρτητα με το αν είναι τα προγράμματα σας καλά σχολιασμένα, πράγμα που συνιστάται. Τέλος, το αρχείο pdf θα πρέπει να ξεκινά με το όνομα σας γραμμένο με Ελληνικούς χαρακτήρες και τον αριθμό μητρώου σας.

Καλή Επιτυχία!