Προγραμματιστική Άσκηση 1 [30.03.2020]

Παράδοση: 28.04.2020, 23:59

Στην παρούσα άσκηση ζητείται η εύρεση και η πιστοποίηση Ελάχιστων Γεννητικών Δέντρων (ΕΓΔ). Βασικός σκοπός είναι η εξοικείωση με τη C++, τη βιβλιοθήκη LEDA και την πειραματική αξιολόγηση.

Συγκεκριμένα, το πρόβλημα που εξετάζεται αφορά στην εύρεση ενός ελάχιστου γεννητικού δέντρου T (Minimum Spanning Tree) από ένα δοθέν συνεκτικό μη-κατευθυνόμενο γράφημα $G = (V, E)$ με συνάρτηση κόστους $wt : E \to \mathbb{N}$. Για την επίλυση του προβλήματος θα χρησιμοποιηθεί ο αλγόριθμος του Kruskal.

Ο αλγόριθμος του Kruskal ταξινομεί τις ακμές του γραφήματος G κατά αύξουσα σειρά κόστους. Έπειτα, με αυτή τη σειρά, προσθέτει μια μια τις ακμές σε ένα υπό ανάπτυξη δένδρο T (αρχικά κενό), αγνοώντας εκείνες που εισάγουν κύκλο στο T. Παρακάτω, παρατίθεται ο ψευδοκώδικας του αλγορίθμου:

```
1  KRUSKAL(G):
2      T = ∅
3      Sort edges of E in increasing order by weight
4      For each edge e = (u, v) ∈ E in order
5          If T ∪ {e} does not create cycle
6              T = T ∪ {e}
```

Η προτεινόμενη, στα πλαίσια της άσκησης, μέθοδος για τον έλεγχο ύπαρξης κύκλων και την οποία καλείστε να υλοποιήσετε βασίζεται στη χρήση λιστών.

Ειδικότερα, θεωρούμε ότι κάθε δενδρική συνεκτική συνιστώσα που δημιουργείται από τον αλγόριθμο αντιστοιχίζεται (ως σύνολο κόμβων) σε μια λίστα L. Για κάθε λίστα L, αποθηκεύουμε: το πλήθος των στοιχείων που περιέχει, $size(L)$, και ένα δείκτη στο τελευταίο στοιχείο της, $last(L)$. Επιπλέον, για κάθε στοιχείο v_i που περιέχει μια λίστα L, διατηρούμε ένα δείκτη $first(v_i)$ που δείχνει στο πρώτο στοιχείο της L. Για παράδειγμα, αν $L = \{v_7, v_4, v_8, v_5, v_9\}$, τότε $size(L) = 5$, $last(L) = v_9$ και $first(v_7) = first(v_4) = first(v_8) = first(v_5) = first(v_9) = v_7$.

Επομένως, εξετάζοντας το δείκτη $first$, μπορούμε να ελέγχουμε αν δυο κορυφές ανήκουν στην ίδια λίστα–συνεκτική συνιστώσα. Αν ο αλγόριθμος πρόκειται να προσθέσει μια ακμή e και τα δυο άκρα της ανήκουν στην ίδια λίστα–συνεκτική συνιστώσα, η e απορρίπτεται (καθώς εισάγει κύκλο). Διαφορετικά, η ακμή e οδηγεί στην ένωση δυο ανεξάρτητων συνεκτικών συνιστωσών. Η ένωση δυο συνεκτικών συνιστωσών θα πρέπει να οδηγεί επιπλέον στη συγχωνεύση των δυο λιστών που αντιστοιχούν σε αυτές. Σε αυτή την περίπτωση, η μικρότερη λίστα πρέπει να προστίθεται στο τέλος της μεγαλύτερης λίστας, ώστε να πραγματοποιηθούν οι λιγότερες ενημερώσεις στους δείκτες $first$. Σε αυτή τη διαδικασία μπορούν να χρησιμοποιηθούν οι μεταβλητές $size$, για την εύρεση της λίστας με το μεγαλύτερο μέγεθος, και ο δείκτης $last$, για την προσθήκη της μικρότερης λίστας αμέσως μετά το τελευταίο στοιχείο της μεγαλύτερης λίστας.

Στα πλαίσια της παρούσας άσκησης, ζητείται:

α) Να κατασκευάσετε έναν αλγόριθμο πιστοποίησης (certifying algorithm) βασισμένο στον αλγόριθμο του Kruskal, χρησιμοποιώντας την παραπάνω προτεινόμενη μέθοδο για τον έλεγχο ύπαρξης κύκλων με χρήση λιστών. Επίσης καλείστε να αξιολογήσετε πειραματικά (ως προς το χρόνο εκτέλεσης) την υλοποίησή σας συγκριτικά με την αντίστοιχη της LEDA, $MIN\_SPANNING\_TREE$.

β) Να σχεδιάσετε και να υλοποιήσετε έναν αλγόριθμο ελεγκτή (checker algorithm) που να ελέγχει αν το ΕΓΔ που επιστρέφει ο αλγόριθμος πιστοποίησης είναι όντως ένα ελάχιστο γεννητικό δένδρο του μη κατευθυνόμενου συνεκτικού γραφήματος $G = (V, E)$. Για τον ελεγκτή, να χρησιμοποιήσετε την ιδιότητα του κύκλου που διέπει τα ΕΓΔ. Δηλαδή, η προσθήκη μιας ***μη δενδρικής*** ακμής $e \in E \setminus T$ δημιουργεί έναν κύκλο C_e στο T, για την οποία θα πρέπει να ισχύει ότι $wt(e) > wt(e')$ για κάθε δενδρική ακμή $e' \in T$ που ανήκει στον C_e. Για την αποδοτική υλοποίηση του ελεγκτή μπορείτε να χρησιμοποιήσετε όποιον τύπο κρίνετε σκόπιμο από τη βιβλιοθήκη LEDA.

Η πειραματική αξιολόγηση πρέπει να γίνει στις παρακάτω κατηγορίες γραφημάτων:

- Τυχαία συνεκτικά γραφήματα με μέγεθος κορυφών $n \in \{4000, 8000, 16000\}$ και μέγεθος ακμών $m = 2n \log n$. Τα κόστη των πλευρών θα παίρνουν τυχαίες ακέραιες τιμές στο διάστημα $[10, 10000]$. Φροντίστε τα παραπάνω γραφήματα να είναι συνεκτικά (δείτε τη $Make\_Connected$ της LEDA) για την ορθή εκτέλεση του αλγόριθμου.

- Γραφήματα τύπου πλέγματος (grid) μεγέθους $r \times c$ (γραμμές $\times$ στήλες), όπου $(r, c) \in \{(200, 200), (300, 300), (400, 400)\}$. Τα κόστη των πλευρών θα παίρνουν τυχαίες ακέραιες τιμές στο διάστημα $[10, 10000]$.

- Συνθετικά γραφήματα τα οποία θα κατασκευάσετε εσείς, με τέτοιο τρόπο ώστε να αναδεικνύουν την πολυπλοκότητα χειρότερης περίπτωσης του αλγορίθμου και τα οποία θα έχουν τουλάχιστον 2000 κορυφές.

Οδηγίες Παράδοσης: Η παράδοση της εργασίας θα πραγματοποιηθεί ηλεκτρονικά μέσω της ιστοσελίδας του μαθήματος στο eclass, υποβάλοντας ένα συμπιεσμένο αρχείο με όνομα hw1_AM_2020, όπου στο AM θα βάλετε τον Αριθμό Μητρώου σας. Το αρχείο θα περιλαμβάνει όλα τα παραδοτέα της εργασίας:

1. **report.pdf**
   Ένα pdf αρχείο που θα περιέχει την αναφορά της εργασίας με τις βασικές αποφάσεις της υλοποίησής σας, τα δεδομένα δοκιμής που χρησιμοποιήσατε και την πειραματική αξιολόγηση σας.

2. **Makefile**
   Ένα αρχείο με τις οδηγίες: α) **compile** για τη μεταγλώττιση του πηγαίου κώδικα και β) **run** για την εκτέλεση των μεταγλωτισμένων αρχείων.

3. **src/**
   Ένα φάκελο στο οποίο θα υπάρχουν όλα τα αρχεία του πηγαίου κώδικά σας.

4. **README**
   Ένα αρχείο που θα περιλαμβάνει τα στοιχεία σας: όνομα, επώνυμο και email.

Παρατήρηση 1: Ο πηγαίος κώδικας που δίνετε για τις υλοποιήσεις και πειραματικές αξιολογήσεις σας πρέπει να είναι σωστά δομημένος, στοιχισμένος και σχολιασμένος. Επίσης ο κώδικάς σας πρέπει να εκτελείται στο σύστημα diogenis.
Παρατήρηση 2: Μπορείτε να χρησιμοποιήσετε όποιους τύπους της LEDA κρίνετε απαραίτητο.
Παρατήρηση 3: Για περαιτέρω διευκρινήσεις σχετικά με την άσκηση επικοινωνήστε με τον Νίκο Ζαχαράτο (zacharato@ceid.upatras.gr).