ΚΑΤΑΝΕΜΗΜΈΝΑ ΣΥΣΤΉΜΑΤΑ
Ι

Εργασία Ι

Γιώργος Ντάκος 1059569

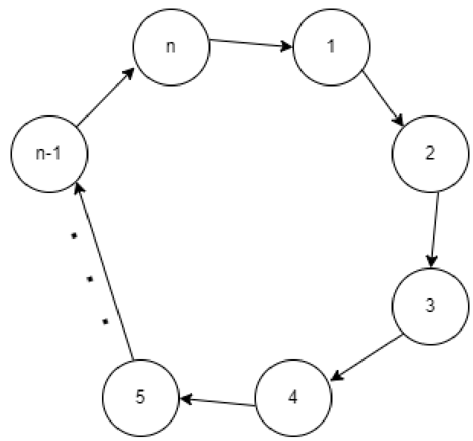
23 Νοεμβρίου 2021

Περιεχόμενα

Κατάλογος Σχημάτων

1 Άσκηση 1

Σύμφωνα με την άσκηση έχουμε ένα πλήθος από διεργασίες(ν ας πύμε) τις πίες για δικιά μας ευκλία θα τις παριστάνυμε με κόμβυς στ σχήμα 1. Κάθε μια από αυτές τις διεργασίες έχει ένα μναδικό ID το οποίο θα το λεμέ UID και ότι είναι συνδεδεμένες σε τοπολογία κατευθυνόμενου δακτυλίου. Επιπλέον κάθε μια γνωρίζει τον εισερχόμενο και εξερχόμενο γείτονα της και τίποτα άλλο παραπάνω. Εμείς θα υποθέσουμε ότι ο κάθε κόμβος έχει επικοινωνία δεξιόστροφης κατεύθυνσης όπως φαίνεται στο σχήμα 1.



Σχήμα 1: Τοπολογία δικτύου.

Σύμφωνα με όλα τα παραπάνω ο αλγόριθμος που θα κατασκευάσουμε θα βασίζεται σε έναν αλγόριθμο εκλογής αρχηγού στην συγκεκριμένη περίπτωση τον αλγόριθμο LCR . Κάθε διεργασία θα έχει αποθηκευμένες 2 μεταβλητές

$$hops = 1, sum = a_i$$

. Σε κάθε γύρο ο κάθε κόμβος i στέλνει στον γειτονικό κόμβο του που είναι εξερχόμενος ένα μήνυμα που θα περιέχει το UID του και το a_i . Κάθε φορά που ένας κόμβος λαμβάνει ένα τέτοιο είδους μήνυμα θα κάνει μια σύγκριση του δικού της UID με αυτό που έλαβε και έχουμε τις εξής περιπτώσεις:

- **Περίπτωση 1:** Αν είναι διαφορετικό αυξάνει την τοπική μεταβλητή hops κατά 1 και την sum κατά a_iκαι προωθεί στον επόμενο του γείτονά το μήνυμα που ελαβε.

- **Περίπτωση 2:** Αν είναι ισούνται τότε έχουμε το συνολικό πλήθος των κόμβων και άθροισμα α οπότε υπολογίζουμε τον ΜΟ και ο αλγόριθμος μας τερματίζει.

Ο ψευδοκώδικας του παραπάνω αλγορίθμου περιγράφεται παρακάτω:

- **Αλφάβητο των μηνυμάτων M:** <u , a >, με u = UID και a = a_i.

- **Το σύνολο καταστάσεων διεργασίας states_i :**
  u όπου είναι ένα UID και αρχικά θα είναι το u_i της i.
  a_i πραγματικός αριθμός όπου είναι η είσοδος του αλγορίθμου.
  sum οπου είναι ένας πραγματικός αριθμός και αρχικά θα είναι a_i .
  hops οπου είναι ένας ακέραιος αρχικά ίσος με 1.
  a ένας πραγματικός αριθμός που θα είναι η έξοδος του αλγορίθμου.
  send είναι ένα μήνυμα <u , a >, ή NULL και αρχικά θα είναι το $<u_i , a_i>$

- **• Η γεννήτρια εξερχομένων μηνυμάτων msgs_i:** Στείλε την τιμή send στον εξερχόμενο γείτονα

- **• Η συνάρτηση αλλαγής κατάστασης της i trans_i:**
  If το εισερχόμενο μήνυμα είναι $<u_i , a_i>$, then
  If u $\neq$ u then
  send = $<u , a_k >$;
  hops = hops + 1;
  sum = sum + a_k;
  else a = sum/hops;
  send = NULL;
  endif
  endif

Όπως βλέπουμε η hops έχει αρχική τιμή 1 και αυξάνεται κατά 1 κάθε φορά που μια διεργασία λαμβάνει μήνυμα με UID διαφορετικό από το δικό της και επειδή κάθε μήνυμα κινείται δεξιόστροφα στο δίκτυο μας και περνάει σιγουρά από κάθε διεργασία ακριβώς μια φορά, οπότε καταλήγει στον κόμβο τον οποίο ξεκίνησε αρά έχουμε το συνολικό πλήθος των διεργασιών. Τώρα για την sum έχει αρχική τιμή a_i και αυξάνεται κατά a_k κάθε φορά που κάποια διεργασία παίρνει μια τιμή a_k από διεργασία με διαφορετικό u. Άρα η sum υπολογίζεται σωστά οπότε η a υπολογίζεται σωστά οπότε και περιέχει τον σωστό ΜΟ. Η χρονική πολυπλοκότητα είναι $\mathbf{O(n)}$ αφού στο τέλος του αλγορίθμου το hops είναι ίσο με n αφού μετράει το συνολικό πλήθος των γυρών που εκτελούνται το οποίο είναι n αρά επαληθεύονται τα παραπάνω. Η πολυπλοκότητα επικοινωνίας είναι $\mathbf{O(n^2)}$ διότι σε κάθε γύρ κάθε κόμβς στέλνει ακριβώς ένα μήνυμα. πότε έχω n μηνύματα σε κάθε γύρο και συνολικά n γύρους αρά επαληθεύονται τα παραπάνω.

2 Άσκηση 2

Σύμφωνα με την άσκηση έχουμε ένα πλήθος από διεργασίες(n ας πούμε). Η κάθε διεργασία γνωρίζει μόνο τους γείτονες του και τίποτα άλλο σχετικά με το δίκτυο. Επίσης θα υποθέσουμε ότι κάθε διεργασία έχει μια μοναδική ταυτότητα uid ώστε να μας βοηθήσει στον αλγόριθμο. Ο διακεκριμένος κόμβος που στο τέλος του αλγορίθμου θα μας υπολογίζει το πλήθος των ακμών του δικτύου θα τον πούμε u_0 και θα επιλέγεται τυχαία στο δίκτυο. Τώρα όπως γνωρίζουμε το πλήθος των ακμών ενός γραφήματος είναι ίσο με το ημιάθροισμα όλων των βαθμών των κόμβων του γραφήματος. Ο βαθμός του κάθε κόμβου είναι ίσος με το πλήθος των ακμών που συνδέονται με αυτόν. Επίσης ξέρουμε ότι το δίκτυο μας είναι ένα μη κατευθυνόμενο συνεκτικό δίκτυο. Οπότε τώρα ο κόμβος u_0 θα τρέξει έναν αλγόριθμο SynchBFS με ριζά τον ίδιο τον κόμβο. Αφού κατασκευαστεί το δένδρο BFS ξεκινάμε από τα φύλλα και στέλνουμε ένα μήνυμα προς τον γονέα του καθενός φύλλου με τον βαθμό του κόμβου που στέλνει το μήνυμα αυτό. Ο κάθε γονέας που λαμβάνει το μήνυμα με του βαθμούς των παιδιών τους προσθέτει στον δικό του βαθμό και έπειτα προωθεί το άθροισμα αυτό στον γονέα του και ούτε κάθε εξής μέχρι να φτάσουμε στην ριζά του δένδρου. Όταν φτάσουμε στην ριζά προσθέτει το βαθμό που έλαβε με τον δικό του και το αποτέλεσμα το διαιρεί δια 2 οπότε και έχουμε το συνολικό πλήθος των ακμών.

Ο ψευδοκώδικας του παραπάνω αλγορίθμου περιγράφεται παρακάτω:

- **Αλφάβητο των μηνυμάτων M:** τα μηνύματα <υ>με u = UID, CHILD και το <d>όπου d = degree.
- **States$_i$:** Parent , ένα UID , αρχικά NULL εκτός αν u = u_0 (ριζά) οπότε parent = u.
  Marked, Boolean ,αρχικά false εκτός αν u = u0 (ριζά) πότε marked = true.
  first_marked, Boolean, αρξικα false.
  send, ένα μήνυμα ¡u¿ ή NULL, αρχικά NULL εκτός αν u = u_0 (ριζά) οπότε send = u.
  children, ένας ακέραιος , αρχικά 0.
  degree, ένας ακέραιος αρχικά 0.
  Confirm,ένας ακέραιος αρχικά 0.
  m, ένας ακέραιος αρχικά 0
  pop ανήκει 0,1, αρχικά 0.
- **msgs$_i$:** στείλε το send σε κάθε γείτονα
  if first_marked = true then στείλε CHILD στον parent endif
  if pop = 1 then στείλε <d>στον parent endif
- **trans$_i$:** if τα εισερχόμενα μηνύματα είναι <u>και στο πλήθος k then degree = k
  endif if marked = false και το σύνολο των εισερχομένων μηνυμάτων είναι

<u_i>για i ανήκει U οπου U ένα υποσύνολο των UIDs, then
επίλεξε τυχαία ένα u ανήκει U και θέσε :
parent = u.
marked = true.
send = <u>
else if first_marked = true
first_marked = false
if έλαβες k εισερχόμενα μηνύματα CHILD then children = k endif
if children = 0 then
pop = 1
endif
else if έλαβες k εισερχόμενα μηνύματα <d>then
confirm = k.
for each <d>
degree = degree + d.
If confirm = children then
pop = 1
endif endif if pop της ρίζας == 1 then
m = degree/2
endif.

Ο αλγόριθμος είναι σωστός γιατί στην αρχή τρέχουμε τον αλγόριθμο synchBFS από έναν διακεκριμένο κόμβο u_0 που επιλέγετε στην τύχη. Αφού κατασκευαστεί το δέντρο χάρης στην μεταβλητή pop ξέρουμε πότε πρέπει να προωθήσουμε τον βαθμό του κόμβου στον γονέα της . Επίσης με την μεταβλητή children μπορούμε να καταλάβουμε ποσά παιδιά έχει ένας κόμβος και ότι για να προωθήσει τον βαθμό στον γονέα του θα πρέπει πρώτα να λάβει τόσα μηνύματα της μορφής <δ>οπου το πλήθος αυτών των εισερχομένων μηνυμάτων το αποθηκεύουμε στην μεταβλητή confirm . Οπότε όταν confirm = children ή children = 0 τότε και μόνο τότε προωθούνται τα μηνύματα <d>στον γονέα των κόμβων. Επίσης με την παραπάνω πρόταση εξασφαλίζουμε ότι ξεκινάμε την προώθηση των μηνυμάτων <d>από τα φύλλα του δέντρου οπότε δεν χάνουμε κάποιο κόμβο. Οπότε κάθε φορά που ένας κόμβος δέχεται ένα μήνυμα <d>προσθέτει στον δικό του βαθμό τον ακέραιο d που είναι ο βαθμός του παιδιού του. Στο τέλος αφού ξέρουμε ποια είναι ριζά, αφού πρώτα προσθέσει το d με τον βαθμό της θα θέσει την μεταβλητή pop = 1 αλλά η ριζά δεν έχει κάποιον γονέα για να προωθήσει κάποιο μήνυμα οπότε και παίρνει το ημιάθροισμα των degrees του γράφου αρά και το πλήθος των ακμών και τερματίζει. Ο αλγόριθμος απαιτεί πρώτα **O(m)** μηνύματα για την κατασκευή του δέντρου οπου m το πλήθος των ακμών του δικτύου και στέλνονται επιπλέον $n - 1$ μηνύματα <d>δηλαδή ένα για κάθε πλευρά του δένδρου, αρά η πολυπλοκότητα επικοινωνίας είναι **O(m+n)** . Ο αλγόριθμος απαιτεί για την κατασκευή του δέντρου diam γύρους και άλλους τόσους γύρους για να προωθήσει τα μηνύματα αρά η χρονική πολυπλοκότητα είναι **O(diam)** .