

Κατανεμημένα Συστήματα I

Μάθημα Βασικής Επιλογής, Χειμερινού Εξαμήνου

Τομέας Εφαρμογών και Θεμελιώσεων

Ιωάννης Χατζηγιαννάκης

Δευτέρα, 7 Νοεμβρίου, 2011
Αίθουσα Β3



Προηγούμενο Μάθημα

- ▶ Σύγχρονα Κατανεμημένα Συστήματα
- ▶ Μοντελοποίηση Συστήματος
- ▶ Πρόβλημα Εκλογής Αρχηγού
- ▶ Μελέτη κατανεμημένων αλγόριθμων για Δίκτυα Δακτυλίου
- ▶ Μελέτη ενός κατανεμημένου αλγόριθμου για Γενικά Δίκτυα



Μοντέλο Σύγχρονου Δικτύου

- ▶ Μία συλλογή υπολογιστικών μονάδων ή `επεξεργαστές`
 - ▶ κάθε επεξεργαστής εκτελεί μόνο μία διεργασία
- ▶ Οι μονάδες του συστήματος είναι συνδεδεμένες με ένα σύγχρονο δίκτυο
 - ▶ Ορίζουμε το σύγχρονο δίκτυο ως ένα **κατευθυνόμενο γράφημα** $G = (V, E)$
 - ▶ αποτελείται από $n = |V|$ **κορυφές** και $m = |E|$ **ακμές**
- ▶ Υποθέτουμε ότι κάθε κανάλι επικοινωνίας μπορεί να δεχτεί μόνο ένα μήνυμα τη φορά
 - ▶ τα κανάλια είναι οι ακμές του γραφήματος
- ▶ Θεωρούμε ότι υπάρχει ένα δεδομένο αλφάβητο M μηνυμάτων



Οι Καταστάσεις των Διεργασιών

- ▶ Κάθε διεργασία $u \in V$ χαρακτηρίζεται από ένα σύνολο καταστάσεων $states_u$
 - ▶ Ορισμένες τις ονομάζουμε **αρχικές καταστάσεις** $start_u$
 - ▶ Ορισμένες τις ονομάζουμε **καταστάσεις τερματισμού** $halt_u$
- ▶ Διαθέτει μια γεννήτρια εξερχόμενων μηνυμάτων $msgs_u : states_u \times nbrs_u^{out} \rightarrow M \cup \{null\}$
 - ▶ δεδομένης της τρέχουσας κατάστασης
 - ▶ δημιουργεί κάποια μηνύματα για τις γειτονικές διεργασίες
- ▶ Διαθέτει μία συνάρτηση αλλαγής κατάστασης $trans_u : states_u \times (M \cup \{null\})^{nbrs_u^{in}} \rightarrow states_u$
 - ▶ δεδομένης της τρέχουσας κατάστασης
 - ▶ τα μηνύματα που παραλήφθηκαν
 - ▶ υπολογίζει την επόμενη κατάσταση της διεργασίας



Έναρξη εκτέλεσης, Βήματα και Γύροι

- ▶ Αρχικά
 - ▶ όλες οι διεργασίες βρίσκονται σε κάποια αρχική κατάσταση
 - ▶ όλα τα κανάλια είναι άδεια
- ▶ Όλες οι διεργασίες, επαναλαμβάνουν `συντονισμένα` τα ακόλουθα δύο βήματα:

1^ο Βήμα

1. Εφαρμογή της γεννήτριας μηνυμάτων
2. Παραγωγή μηνυμάτων για τους εξερχόμενους γείτονες
3. Αποστολή μηνυμάτων μέσω των αντίστοιχων καναλιών

2^ο Βήμα

1. Εφαρμογή της συνάρτησης αλλαγής κατάστασης
 2. Διαγραφή όλων των μηνυμάτων από τα κανάλια.
- ▶ Ο συνδυασμός των δύο βημάτων ονομάζεται **γύρος**



Μέτρηση πολυπλοκότητας

- ▶ Σχεδιασμός Συστήματος
 - ▶ Ορισμός Ελάχιστων Απαιτήσεων
 - ▶ Επιλογή κατάλληλου κατανεμημένου αλγόριθμου
 - ▶ Πως μπορούμε να μετρήσουμε την απόδοση;

Χρονική πολυπλοκότητα

Το πλήθος των γύρων που απαιτούνται για να παραχθούν όλες οι ζητούμενες έξοδοι, ή μέχρι να τερματιστούν όλες οι διεργασίες (δηλ. να βρεθούν σε μια τερματική κατάσταση).

Πολυπλοκότητα επικοινωνίας

Ο συνολικός αριθμός μη μηδενικών μηνυμάτων (δηλ. δεν προσμετρούνται τα null μηνύματα) που αποστέλλονται.



Πρόβλημα Εκλογής Αρχηγού

- ▶ Ορισμός Προβλήματος
- ▶ Αδυναμία αντιμετώπισης του προβλήματος όταν οι διεργασίες δεν έχουν μοναδικές ταυτότητες
- ▶ Δίκτυα Δακτυλίου
 1. Αλγόριθμος των LeLann, Chang και Roberts
 - ▶ Η χρονική πολυπλοκότητα είναι $\mathcal{O}(n)$
 - ▶ Η πολυπλοκότητα επικοινωνίας είναι $\mathcal{O}(n^2)$
 2. Αλγόριθμος των Hirschberg και Sinclair
 - ▶ Η χρονική πολυπλοκότητα είναι $\mathcal{O}(n)$
 - ▶ Η πολυπλοκότητα επικοινωνίας είναι $\mathcal{O}(n \log n)$
- ▶ Γενικά Δίκτυα
 - ▶ Αλγόριθμος FloodMax
 - ▶ Η χρονική πολυπλοκότητα είναι $\mathcal{O}(\text{diam}(G))$
 - ▶ Η πολυπλοκότητα επικοινωνίας είναι $\mathcal{O}(\text{diam}(G) \cdot m)$



Σύνοψη 4^{ης} Διάλεξης

Προηγούμενο Μάθημα

Προηγούμενο Μάθημα
Σύγχρονα Κατανεμημένα Συστήματα
Εκλογή Αρχηγού

Κατανεμημένες Δομές

Αναζήτηση κατά Εύρος
Συνοτότερα Μονοπάτια
Ασκήσεις

Σύνοψη Μαθήματος

Σύνοψη Μαθήματος
Βιβλιογραφία
Επόμενο Μάθημα

- ▶ <http://techcrunch.com>
- ▶ <http://www.crunchbase.com/>
- ▶ <http://www.seedcamp.com/>
- ▶ <http://ycombinator.com/>



Γενικά

Αναζήτηση κατά Εύρος

Σε ένα σύγχρονο δίκτυο G , η αναζήτηση κατά εύρος απαιτεί την κατασκευή ενός επικαλυπτικού δέντρου $T(G)$, με ρίζα την διεργασία u_0 όπου οι κορυφές που είναι σε απόσταση d από την u_0 στο G , βρίσκονται στο επίπεδο d στο δέντρο $T(G)$.

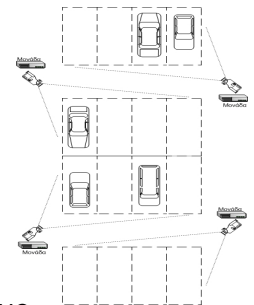
- ▶ Η κατασκευή αυτής της δομής είναι χρήσιμη σε πολλές περιπτώσεις
 - ▶ Γρήγορη μετάδοση πληροφορίας
 - ▶ Πρόβλημα Εκλογής Αρχηγού
 - ▶ Πρόβλημα Καταμέτρησης



Παράδειγμα (1)

Έλεγχος Χώρου Στάθμευσης Αυτοκινήτων

Για λόγους αυτόματου ελέγχου, η εταιρεία τοποθετεί ένα πλήθος από μονάδες εφοδιασμένες με αισθητήρες έτσι ώστε να καλύπτουν όλο τον χώρο στάθμευσης. Οι μονάδες συνδέονται μέσω ενός ασύρματου δικτύου (IEEE 802.11b). Το σύστημα υπολογίζει αυτόματα το σύνολο των σταθμευμένων αυτοκινήτων.



- ▶ Οι μονάδες έχουν ξεχωριστή διεύθυνση στο δίκτυο
- ▶ Οι μονάδες χρησιμοποιούν φωτογραφικές μηχανές
- ▶ Ο χειριστής του συστήματος χρησιμοποιεί έναν τερματικό σταθμό που είναι συνδεδεμένος στο ασύρματο δίκτυο



Παράδειγμα (2)

- ▶ Ο τερματικός σταθμός δημιουργεί ένα δέντρο αναζήτησης κατά εύρος
- ▶ Κάθε μονάδα μετράει το πλήθος των αυτοκινήτων που 'βλέπει' και το στέλνει στον γονέα της στο δέντρο
- ▶ Ο γονέας συγκεντρώνει τις τιμές που έλαβε από τα παιδιά της στο δέντρο, τις προσθέτει με την δικιά της και στέλνει το σύνολο στον δικό της γονέα
- ▶ Το άθροισμα που υπολογίζει η ρίζα του δέντρου είναι το πλήθος των σταθμευμένων αυτοκινήτων



Άλλες Εφαρμογές

Πρόβλημα Εκλογής Αρχηγού:

- ▶ Ο αλγόριθμος κατασκευάζει ένα δέντρο αναζήτησης κατά εύρος για κάθε διεργασία
- ▶ Η διεργασία με την 'μέγιστη ταυτότητα' εκλέγεται αρχηγός και όλες οι υπόλοιπες μεταβαίνουν στην κατάσταση 'μη-εκλεγμένη'
- ▶ Δεν χρειάζεται να γνωρίζουν το πλήθος n ή την διάμετρο του δικτύου $diam(G)$

Υπολογισμός Διαμέτρου Δικτύου:

- ▶ Ο αλγόριθμος κατασκευάζει ένα δέντρο αναζήτησης κατά εύρος για κάθε διεργασία
- ▶ Κάθε διεργασία υπολογίζει το 'ύψος' του δέντρου της
- ▶ Κάθε διεργασία χρησιμοποιεί το δέντρο της για να ανακαλύψει το μέγιστο ύψος, δηλ. την διάμετρο του δικτύου



Ο Αλγόριθμος SynchBFS

Αλγόριθμος SynchBFS

Οι διεργασίες διατηρούν μια μεταβλητή **μαρκαρισμένη** η οποία αρχικά είναι `false` και μια μεταβλητή **γονέας** με αρχική τιμή `0`. Αρχικά, η διεργασία u_0 θέτει την μεταβλητή **μαρκαρισμένη** ως `true`, την μεταβλητή **γονέας** με την ταυτότητα της, και στέλνει ένα μήνυμα 'αναζήτησης' σε όλους τους γείτονες της. Σε κάθε γύρο, εάν μια διεργασία λάβει ένα μήνυμα 'αναζήτησης' και η τιμή της μεταβλητής **μαρκαρισμένη** είναι `false`, τότε θέτει την μεταβλητή σε `true`, θέτει την μεταβλητή **γονέας** με την ταυτότητα της διεργασίας από όπου έλαβε το μήνυμα, και στον επόμενο γύρο στέλνει ένα μήνυμα 'αναζήτησης' σε όλους τους γείτονες της.

- ▶ Δεν γνωρίζουν το πλήθος των διεργασιών (n)
- ▶ Έχουν μοναδικές ταυτότητες

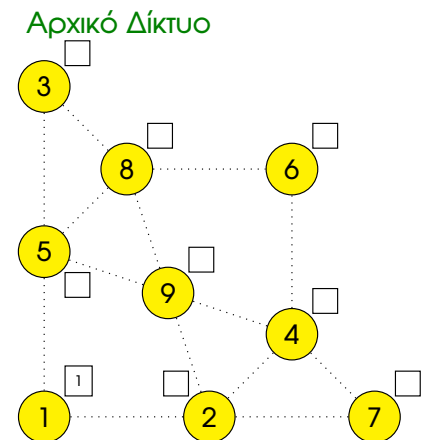


Παράδειγμα Εκτέλεσης Αλγόριθμου SynchBFS

Αρχικό Δίκτυο

Το δίκτυο έχει 9 μονάδες, 14 κανάλια
Η διεργασία 1 ξεκινά την εκτέλεση του αλγορίθμου.

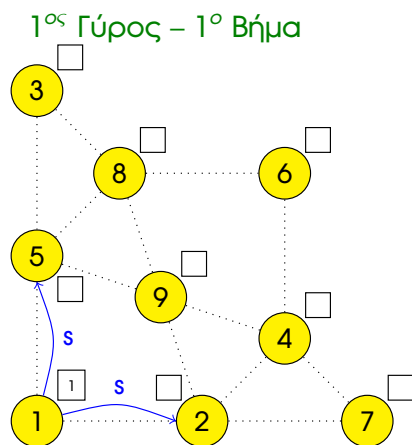
Η διεργασία 1 θεωρείται μαρκαρισμένη.
Όλες οι άλλες διεργασίες δεν είναι μαρκαρισμένες.



Παράδειγμα Εκτέλεσης Αλγόριθμου ΣηχBFS

1^{ος} Γύρος – 1^ο Βήμα

Η διεργασία **1** στέλνει μήνυμα 'αναζήτησης' σε όλους του γείτονες της.



Παράδειγμα Εκτέλεσης Αλγόριθμου ΣηχBFS

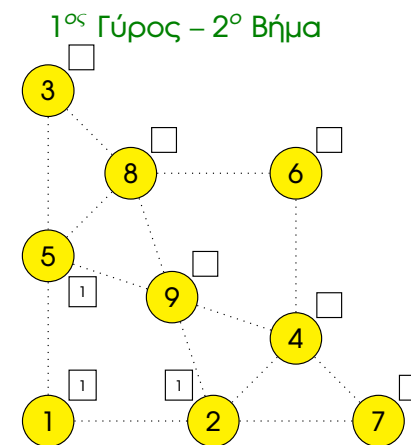
1^{ος} Γύρος – 1^ο Βήμα

Η διεργασία **1** στέλνει μήνυμα 'αναζήτησης' σε όλους του γείτονες της.

1^{ος} Γύρος – 2^ο Βήμα

Οι διεργασίες **2, 5** μαρκάρονται.

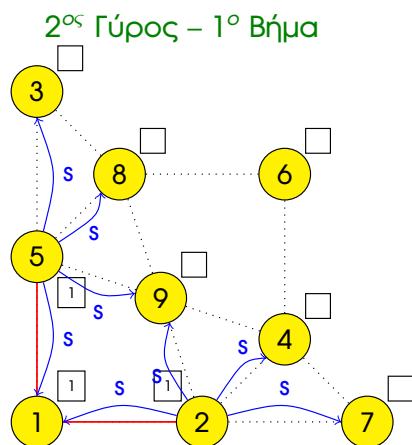
Οι διεργασίες **2, 5** θέτουν την **1** ως γονέα στο δέντρο.



Παράδειγμα Εκτέλεσης Αλγόριθμου ΣηχBFS

2^{ος} Γύρος – 1^ο Βήμα

Οι διεργασίες **2, 5** στέλνουν μήνυμα 'αναζήτησης' σε όλους τους γείτονες τους.



Παράδειγμα Εκτέλεσης Αλγόριθμου ΣηχBFS

2^{ος} Γύρος – 1^ο Βήμα

Οι διεργασίες **2, 5** στέλνουν μήνυμα 'αναζήτησης' σε όλους τους γείτονες τους.

2^{ος} Γύρος – 2^ο Βήμα

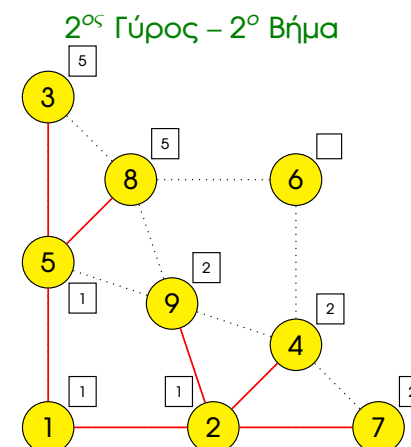
Η διεργασία **1** αγνοεί τα μηνύματα 'αναζήτησης' που έλαβε.

Οι διεργασίες **3, 4, 7, 8, 9** μαρκάρονται.

Οι διεργασίες **3, 8** θέτουν την **5** ως γονέα στο δέντρο.

Οι διεργασίες **4, 7** θέτουν την **2** ως γονέα στο δέντρο.

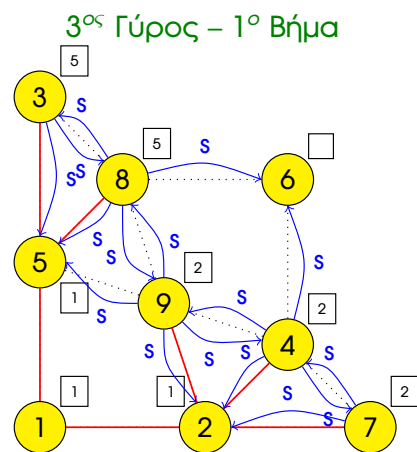
Η διεργασία **9** διαλέγει τυχαία την **2** ως γονέα στο δέντρο.



Παράδειγμα Εκτέλεσης Αλγόριθμου ΣηχBFS

3^{ος} Γύρος – 1^ο Βήμα

Οι διεργασίες **3, 4, 7, 8, 9** στέλνουν μήνυμα `αναζήτησης` σε όλους τους γείτονες τους.



Παράδειγμα Εκτέλεσης Αλγόριθμου ΣηχBFS

3^{ος} Γύρος – 1^ο Βήμα

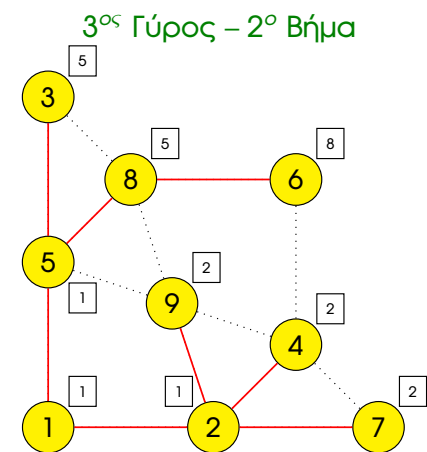
Οι διεργασίες **3, 4, 7, 8, 9** στέλνουν μήνυμα `αναζήτησης` σε όλους τους γείτονες τους.

3^{ος} Γύρος – 2^ο Βήμα

Οι διεργασίες **2, 3, 4, 5, 7, 8, 9** αγνοούν τα μηνύματα `αναζήτησης` που έλαβαν.

Η διεργασία **6** μαρκάρεται.

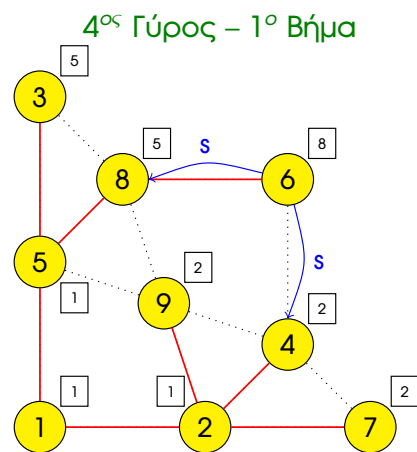
Η διεργασία **6** διαλέγει τυχαία την **8** ως γονέα στο δέντρο.



Παράδειγμα Εκτέλεσης Αλγόριθμου ΣηχBFS

4^{ος} Γύρος – 1^ο Βήμα

Η διεργασία **6** στέλνει μήνυμα `αναζήτησης` σε όλους του γείτονες της.



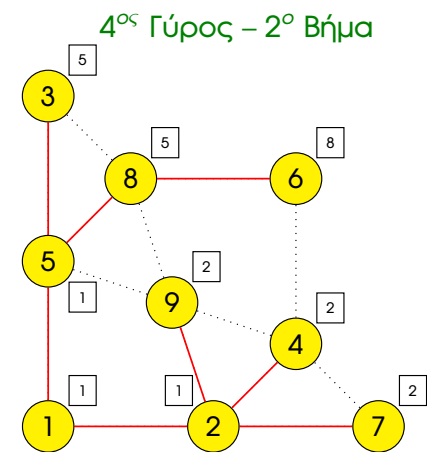
Παράδειγμα Εκτέλεσης Αλγόριθμου ΣηχBFS

4^{ος} Γύρος – 1^ο Βήμα

Η διεργασία **6** στέλνει μήνυμα `αναζήτησης` σε όλους του γείτονες της.

4^{ος} Γύρος – 2^ο Βήμα

Οι διεργασίες **4, 8** αγνοούν τα μηνύματα `αναζήτησης` που έλαβαν.



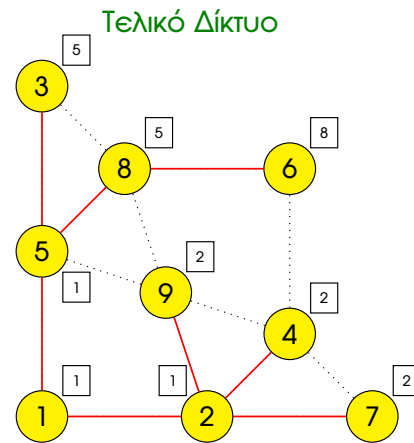
Παράδειγμα Εκτέλεσης Αλγόριθμου SychBFS

Τελικό Δίκτυο

Το δέντρο αναζήτησης κατά εύρος έχει κατασκευαστεί.

Ο αλγόριθμος εκτελέστηκε σε 4 γύρους.

Συνολικά μεταδόθηκαν 28 μηνύματα.



Χαρακτηριστικά του Αλγόριθμου SychBFS

- ▶ Ο αλγόριθμος SychBFS κατασκευάζει ένα δέντρο αναζήτησης κατά εύρος.
- ▶ Η δομή του δέντρου δεν είναι αποθηκευμένη σε κάποια "κεντρική" διεργασία
- ▶ Η χρονική πολυπλοκότητα είναι $\mathcal{O}(\text{diam}(G))$
 - ▶ Στην πραγματικότητα είναι η μέγιστη απόσταση από την u_0
 - ▶ Στο παράδειγμα, η διάμετρος είναι 4 – η μέγιστη απόσταση από την u_0 είναι 3
- ▶ Η πολυπλοκότητα επικοινωνίας είναι $\mathcal{O}(m)$

Βελτίωση Πολυπλοκότητας Μηνυμάτων

Μπορούμε να μειώσουμε τον αριθμό των μηνυμάτων που χρησιμοποιεί ο αλγόριθμος ως εξής

- ▶ Οι διεργασίες μπορούν να αναγνωρίσουν το κανάλι από το οποίο έλαβαν ένα μήνυμα
- ▶ Οι διεργασίες δεν στέλνουν μηνύματα "αναζήτησης" στα κανάλια από τα οποία έλαβαν ένα μήνυμα "αναζήτησης"

Στο παράδειγμα εκτέλεσης του SychBFS τα συνολικά μηνύματα μειώνονται κατά 10 – δηλ. 18

Μετάδοση Μηνύματος

Μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο για την μετάδοση ενός μηνύματος σε όλο το δίκτυο

- ▶ Η διεργασία u_0 θέλει να στείλει το μήνυμα M σε όλες τις διεργασίες του δικτύου
- ▶ Η διεργασία u_0 ξεκινάει την κατασκευή του δέντρου στέλνοντας το μήνυμα "αναζήτησης" το οποίο περιέχει και το μήνυμα M
- ▶ Οι άλλες διεργασίες επισυνάπτουν και αυτές με την σειρά τους το μήνυμα M με τα μηνύματα "αναζήτησης" που στέλνουν
- ▶ Εφόσον το δέντρο περιέχει όλες τις διεργασίες, το μήνυμα M τελικά παραλαμβάνετε από όλες τις διεργασίες του δικτύου

Πλήρη Γνώση (1)

Είναι απαραίτητο κάθε διεργασία να γνωρίζει και τα 'παιδιά' της
Τροποποιούμε τον αλγόριθμο SynchronBFS ως εξής:

- ▶ Κάθε διεργασία που λαμβάνει ένα μήνυμα 'αναζήτησης' επιστρέφει στον αποστολέα ένα μήνυμα 'γονέας' ή 'μη-γονέας' ανάλογα του αν ο αποστολέας είναι ο γονέας της διεργασίας

Κατά την ολοκλήρωση της εκτέλεσης του SynchronBFS, όλες οι διεργασίες γνωρίζουν τα 'παιδιά' τους

Ο αλγόριθμος SynchronBFS απαιτεί το πολύ $diam(G) + 2$ γύρους και χρησιμοποιεί $2 \cdot m$ μηνύματα

Η χρονική πολυπλοκότητα και η πολυπλοκότητα μηνυμάτων παραμένουν σταθερές



Πλήρη Γνώση (2)

Μπορούμε να μειώσουμε τον αριθμό των μηνυμάτων που χρησιμοποιεί ο αλγόριθμος

Τροποποιούμε τον αλγόριθμο SynchronBFS ως εξής:

- ▶ Κάθε διεργασία που λαμβάνει μήνυμα 'αναζήτησης' απαντά στον αποστολέα με ένα μήνυμα 'γονέας' εφόσον ο αποστολέας είναι ο γονέας της διεργασίας
- ▶ Εάν η διεργασία που έστειλε το μήνυμα 'αναζήτησης' σε μία γειτονική διεργασία, δεν λάβει μήνυμα 'γονέας' στον επόμενο γύρο, θεωρεί ότι δεν επιλέχτηκε ως γονέας της γειτονικής διεργασίας

Ο αλγόριθμος SynchronBFS'' χρησιμοποιεί $m + n - 1$ μηνύματα



Τερματισμός (1)

Πως μπορεί η διεργασία u_0 να μάθει πότε ολοκληρώθηκε η κατασκευή του δέντρου;

- ▶ Δεν είναι γνωστή η διάμετρος του δικτύου και το πλήθος των διεργασιών n

Βασίζομαστε την παραλλαγή SynchronBFS όπου οι διεργασίες απαντάνε στον αποστολέα ενός μηνύματος 'αναζήτησης' με ένα μήνυμα 'γονέας' ή 'μη-γονέας' ανάλογα του αν ο αποστολέας είναι ο γονέας της διεργασίας.

Εφόσον οι διεργασίες μπορούν να αναγνωρίσουν το κανάλι από το οποίο έλαβαν ένα μήνυμα, δεν στέλνουν μηνύματα 'αναζήτησης' στα κανάλια από τα οποία έλαβαν ένα μήνυμα 'αναζήτησης'



Τερματισμός (2)

Επεκτείνουμε τον SynchronBFS ως εξής:

- ▶ Κάθε διεργασία που λαμβάνει ένα μήνυμα 'αναζήτησης' επιστρέφει στον αποστολέα στον επόμενο γύρο ένα μήνυμα 'μη-γονέας' αν ο αποστολέας δεν είναι ο γονέας της διεργασίας
- ▶ Η διεργασία καθυστερεί την αποστολή του μηνύματος 'γονέας' έως ότου λάβει κάποιο μήνυμα 'γονέας' ή 'μη-γονέας' από όλες τις διεργασίες στις οποίες έστειλε μηνύματα 'αναζήτησης'



Τερματισμός (3)

Επομένως τα μηνύματα 'γονέας' ή 'μη-γονέας' έχουν δύο χρήσεις

1. **Παροχή πλήρους γνώσης** – κάθε διεργασία γνωρίζει και τα 'παιδιά' της
2. **Ενημέρωση τερματισμού** – κάθε διεργασία γνωρίζει πότε τα υποδέντρα των 'παιδιών' της έχουν κατασκευαστεί

Ο αλγόριθμος $SyncBFS_T$ απαιτεί το πολύ $2 \cdot diam(G) + 2$ γύρους και χρησιμοποιεί $2 \cdot m$ μηνύματα

Παράδειγμα Εκτέλεσης Αλγόριθμου $SyncBFS_T$

Αρχικό Δίκτυο

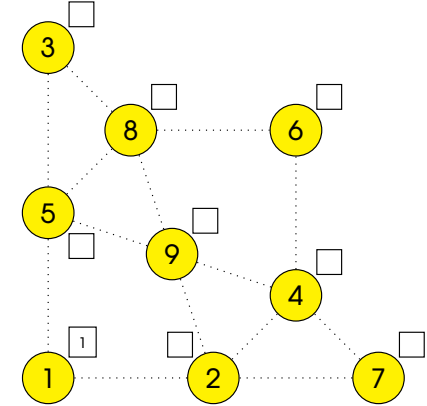
Το δίκτυο έχει 9 μονάδες, 14 κανάλια

Η διεργασία 1 ξεκινά την εκτέλεση του αλγορίθμου.

Η διεργασία 1 θεωρείται μαρκαρισμένη.

Όλες οι άλλες διεργασίες δεν είναι μαρκαρισμένες.

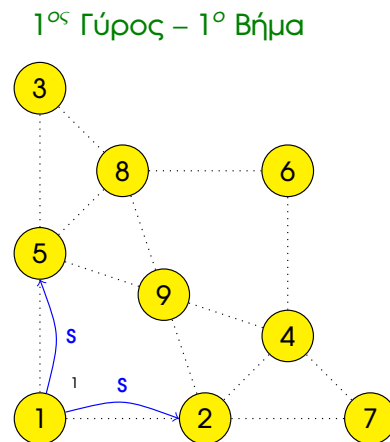
Αρχικό Δίκτυο



Παράδειγμα Εκτέλεσης Αλγόριθμου $SyncBFS_T$

1ος Γύρος – 1ο Βήμα

Η διεργασία 1 στέλνει μήνυμα 'αναζήτησης' σε όλους του γείτονες της.



Παράδειγμα Εκτέλεσης Αλγόριθμου $SyncBFS_T$

1ος Γύρος – 1ο Βήμα

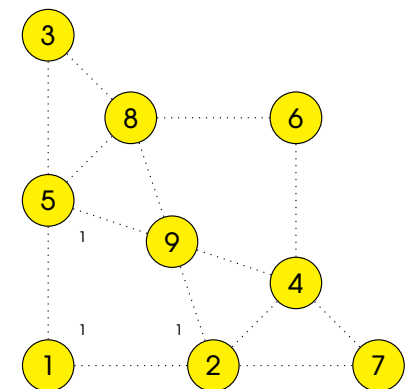
Η διεργασία 1 στέλνει μήνυμα 'αναζήτησης' σε όλους του γείτονες της.

1ος Γύρος – 2ο Βήμα

Οι διεργασίες 2, 5 μαρκάρονται.

Οι διεργασίες 2, 5 θέτουν την 1 ως γονέα στο δέντρο.

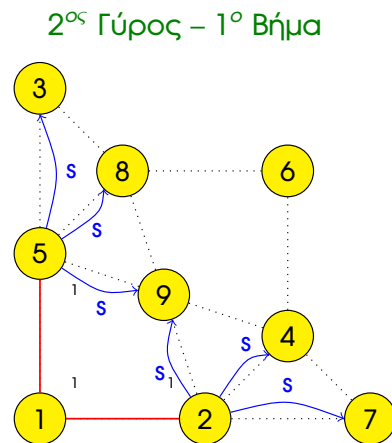
1ος Γύρος – 2ο Βήμα



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

2^{ος} Γύρος – 1^ο Βήμα

Οι διεργασίες **2, 5** στέλνουν μήνυμα 'αναζήτησης' σε όλους τους γείτονες τους.



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

2^{ος} Γύρος – 1^ο Βήμα

Οι διεργασίες **2, 5** στέλνουν μήνυμα 'αναζήτησης' σε όλους τους γείτονες τους.

2^{ος} Γύρος – 2^ο Βήμα

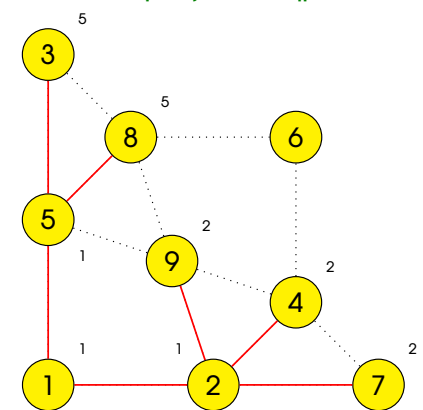
Οι διεργασίες **3, 4, 7, 8, 9** μαρκάρονται.

Οι διεργασίες **3, 8** θέτουν την **5** ως γονέα στο δέντρο.

Οι διεργασίες **4, 7** θέτουν την **2** ως γονέα στο δέντρο.

Η διεργασία **9** διαλέγει τυχαία την **2** ως γονέα στο δέντρο.

2^{ος} Γύρος – 2^ο Βήμα

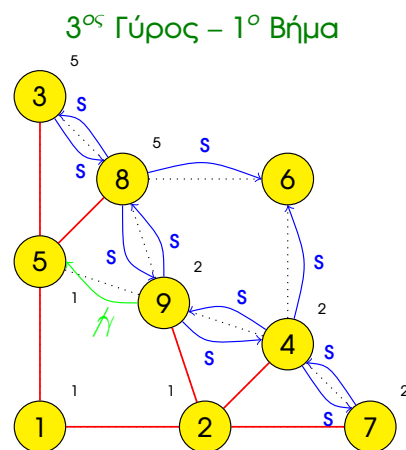


Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

3^{ος} Γύρος – 1^ο Βήμα

Οι διεργασίες **3, 4, 7, 8, 9** στέλνουν μήνυμα 'αναζήτησης' σε όλους τους γείτονες τους.

Η διεργασία **9** στέλνει μήνυμα 'μη-γονέας' στην **5**



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

3^{ος} Γύρος – 1^ο Βήμα

Οι διεργασίες **3, 4, 7, 8, 9** στέλνουν μήνυμα 'αναζήτησης' σε όλους τους γείτονες τους.

Η διεργασία **9** στέλνει μήνυμα 'μη-γονέας' στην **5**

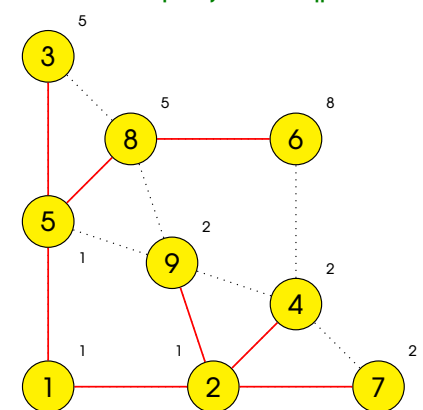
3^{ος} Γύρος – 2^ο Βήμα

Οι διεργασίες **2, 3, 4, 5, 7, 8, 9** αγνοούν τα μηνύματα 'αναζήτησης' που έλαβαν.

Η διεργασία **6** μαρκάρεται.

Η διεργασία **6** διαλέγει τυχαία την **8** ως γονέα στο δέντρο.

3^{ος} Γύρος – 2^ο Βήμα



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

4^{ος} Γύρος – 1^ο Βήμα

Η διεργασία 3 στέλνει μήνυμα

‘μη-γονέας’ στην 8

Η διεργασία 8 στέλνει μήνυμα ‘μη-γονέας’ στην 3

Η διεργασία 8 στέλνει μήνυμα ‘μη-γονέας’ στην 9

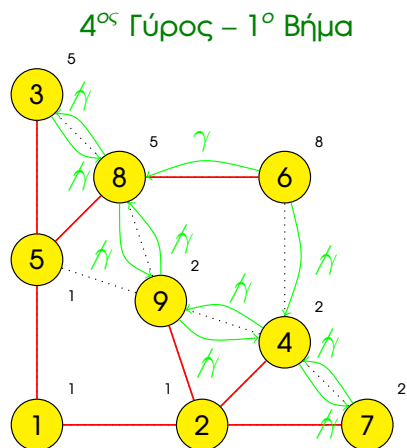
Η διεργασία 9 στέλνει μήνυμα ‘μη-γονέας’ στην 4

Η διεργασία 4 στέλνει μήνυμα ‘μη-γονέας’ στην 7

Η διεργασία 7 στέλνει μήνυμα ‘μη-γονέας’ στην 4

Η διεργασία 6 στέλνει μήνυμα ‘μη-γονέας’ στην 4

Η διεργασία 6 στέλνει μήνυμα ‘γονέας’ στην 8



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

4^{ος} Γύρος – 1^ο Βήμα

Η διεργασία 3 στέλνει μήνυμα

‘μη-γονέας’ στην 8

Η διεργασία 8 στέλνει μήνυμα ‘μη-γονέας’ στην 3

Η διεργασία 8 στέλνει μήνυμα ‘μη-γονέας’ στην 9

Η διεργασία 9 στέλνει μήνυμα ‘μη-γονέας’ στην 4

Η διεργασία 4 στέλνει μήνυμα ‘μη-γονέας’ στην 7

Η διεργασία 7 στέλνει μήνυμα ‘μη-γονέας’ στην 4

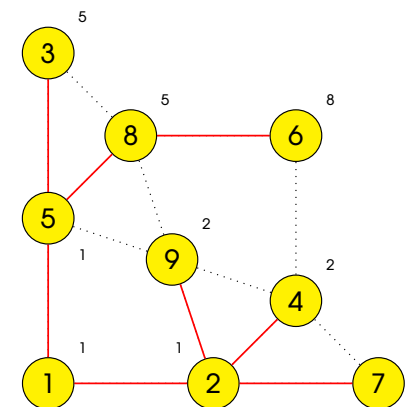
Η διεργασία 6 στέλνει μήνυμα ‘μη-γονέας’ στην 4

Η διεργασία 6 στέλνει μήνυμα ‘γονέας’ στην 8

4^{ος} Γύρος – 2^ο Βήμα

Η διεργασία 8 διαπιστώνει ότι η κατασκευή του υποδέντρου της 6 ολοκληρώθηκε

4^{ος} Γύρος – 2^ο Βήμα



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

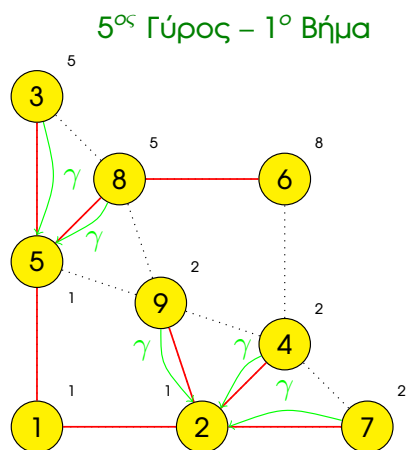
5^{ος} Γύρος – 1^ο Βήμα

Οι διεργασίες 3, 8

στέλνουν μήνυμα ‘γονέας’ στην 5

Οι διεργασίες 4, 7, 9 στέλνουν μήνυμα

‘γονέας’ στην 2



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

5^{ος} Γύρος – 1^ο Βήμα

Οι διεργασίες 3, 8

στέλνουν μήνυμα ‘γονέας’ στην 5

Οι διεργασίες 4, 7, 9 στέλνουν μήνυμα

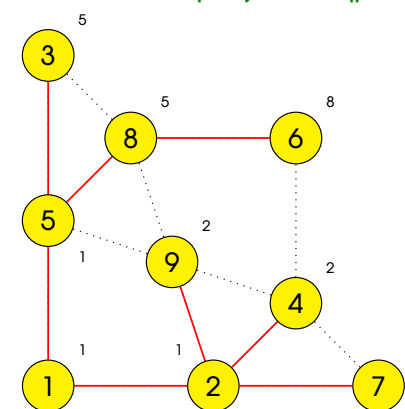
‘γονέας’ στην 2

5^{ος} Γύρος – 2^ο Βήμα

Η διεργασία 5 διαπιστώνει ότι η κατασκευή των υποδέντρων των 3, 8 ολοκληρώθηκε

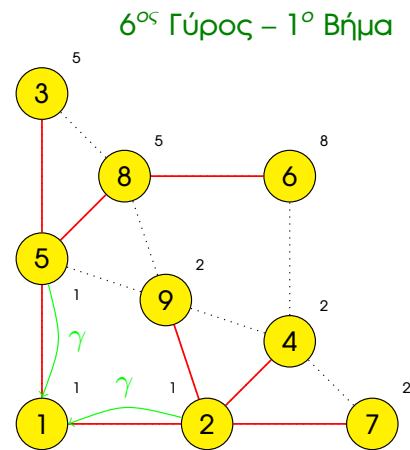
Η διεργασία 2 διαπιστώνει ότι η κατασκευή των υποδέντρων των 4, 7, 9 ολοκληρώθηκε

5^{ος} Γύρος – 2^ο Βήμα



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

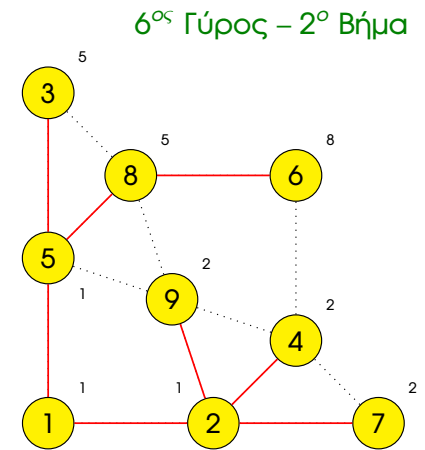
6^{ος} Γύρος – 1^ο Βήμα Οι διεργασίες **2, 5** στέλνουν μήνυμα 'γονέας' στην **1**



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

6^{ος} Γύρος – 1^ο Βήμα Οι διεργασίες **2, 5** στέλνουν μήνυμα 'γονέας' στην **1**
6^{ος} Γύρος – 2^ο Βήμα

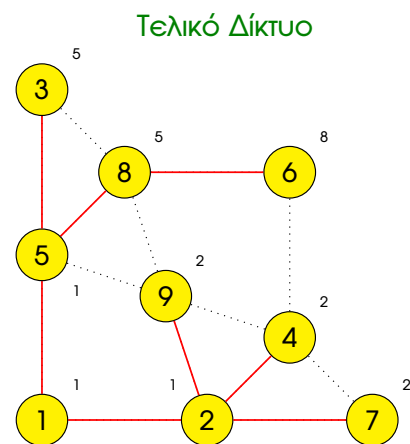
Η διεργασία **1** διαπιστώνει ότι η κατασκευή των υποδέντρων των **2, 5** ολοκληρώθηκε
 Η διεργασία **1** τερματίζει



Παράδειγμα Εκτέλεσης Αλγόριθμου SyncBFS_T

Τελικό Δίκτυο

Το δέντρο αναζήτησης κατά εύρος έχει κατασκευαστεί.
 Ο αλγόριθμος εκτελέστηκε σε 6 γύρους.
 Συνολικά μεταδόθηκαν 36 μηνύματα.



Γενικά (1)

Συνοτότερα Μονοπάτια

Σε ένα σύγχρονο δίκτυο G , όπου οι ακμές έχουν βάρη, η εύρεση όλων των συνοτότερων μονοπατιών από μια διεργασία u_0 προς όλες τις υπόλοιπες διεργασίες, απαιτεί την κατασκευή ενός επικαλυπτικού δέντρου $T(G)$, με ρίζα την διεργασία u_0 όπου οι κορυφές που είναι σε απόσταση d από την u_0 στο G , είναι σε απόσταση d στο δέντρο $T(G)$.

- ▶ Η κατασκευή αυτής της δομής είναι χρήσιμη σε πολλές περιπτώσεις
 - ▶ Ελαχιστοποιεί το μέγιστο κόστος μετάδοσης πληροφορίας
- ▶ Τα βάρη στις ακμές μπορεί να απεικονίζουν
 - ▶ Ταχύτητα μετάδοσης
 - ▶ Κόστος μετάδοσης



Γενικά (2)

- ▶ Υποθέτουμε ότι κάθε διεργασία γνωρίζει τα κόστη των γειτνιάζουσων ακμών
 - ▶ Είναι μια εσωτερική μεταβλητή **βάρος**
- ▶ Κάθε διεργασία γνωρίζει το πλήθος των διεργασιών n
- ▶ Εάν όλα τα βάρη των ακμών είναι ίσα τότε ένα δέντρο αναζήτησης κατά εύρος είναι επίσης και ένα δέντρο συντομότερων μονοπατιών
 - ▶ μια απλή τροποποίηση του SyncBFS_r μπορεί να υπολογίσει μέσω των μηνυμάτων 'γονέας' την απόσταση από την διεργασία u_0



Ο Αλγόριθμος BellmanFord

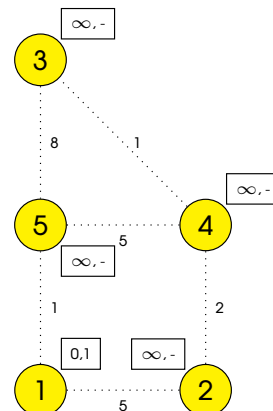
Αλγόριθμος BellmanFord

Οι διεργασίες διατηρούν μια μεταβλητή **απόσταση** η οποία αρχικά είναι ∞ και μια μεταβλητή **γονέας** με αρχική τιμή 0. Σε κάθε γύρο, οι διεργασίες στέλνουν την τιμή της μεταβλητής **απόσταση** σε όλους τους γείτονες τους. Κάθε διεργασία u , επεξεργάζεται τα μηνύματα που έλαβε, και εντοπίζει το μήνυμα της v το οποίο έχει την ελάχιστη τιμή **απόσταση_v + βάρος_{vu}**. Εάν ένα τέτοιο μήνυμα βρεθεί, θέτει την μεταβλητή **απόσταση_u** στην τιμή **απόσταση_v + βάρος_{vu}** και την μεταβλητή **γονέας_u** = v . Μετά από $n - 1$ γύρους, η τιμή της μεταβλητή **απόσταση** είναι η ελάχιστη απόσταση της διεργασίας με την u_0 και η μεταβλητή **γονέας** δείχνει τον γονέα της διεργασίας στο δέντρο των συντομότερων μονοπατιών.



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

Αρχικό Δίκτυο



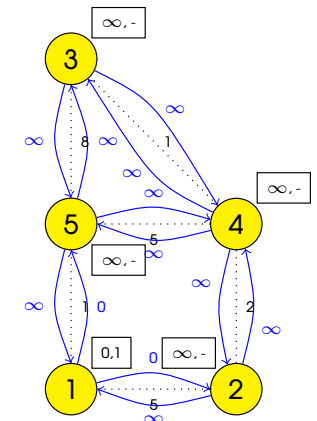
Αρχικό Δίκτυο

Το δίκτυο έχει 5 μονάδες, 6 κανάλια
Η διεργασία 1 ξεκινά την εκτέλεση του αλγορίθμου.
Η διεργασία 1 είναι η ρίζα του δέντρου.



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

1^{ος} Γύρος – 1^ο Βήμα



1^{ος} Γύρος – 1^ο Βήμα

Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.
Οι διεργασίες 2, 3, 4, 5 στέλνουν μήνυμα '∞' σε όλους τους γείτονες τους.



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

1^{ος} Γύρος – 2^ο Βήμα

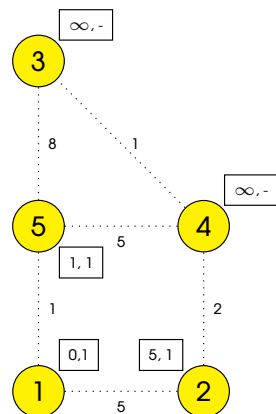
1^{ος} Γύρος – 1^ο Βήμα

Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.

Οι διεργασίες 2, 3, 4, 5 στέλνουν μήνυμα '∞' σε όλους τους γείτονες τους.

1^{ος} Γύρος – 2^ο Βήμα

Οι διεργασίες 2, 5 θέτουν την 1 ως γονέα με απόσταση 5 και 1 αντίστοιχα.



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

2^{ος} Γύρος – 1^ο Βήμα

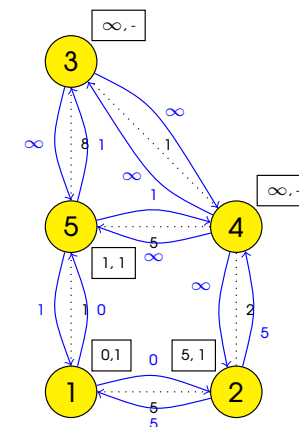
2^{ος} Γύρος – 1^ο Βήμα

Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.

Η διεργασία 2 στέλνει μήνυμα '5' σε όλους του γείτονες της.

Η διεργασία 5 στέλνει μήνυμα '1' σε όλους του γείτονες της.

Οι διεργασίες 3, 4 στέλνουν μήνυμα '∞' σε όλους τους γείτονες τους.



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

2^{ος} Γύρος – 1^ο Βήμα

Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.

Η διεργασία 2 στέλνει μήνυμα '5' σε όλους του γείτονες της.

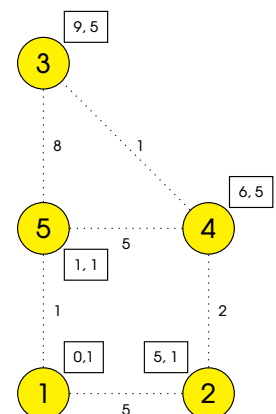
Η διεργασία 5 στέλνει μήνυμα '1' σε όλους του γείτονες της.

Οι διεργασίες 3, 4 στέλνουν μήνυμα '∞' σε όλους τους γείτονες τους.

2^{ος} Γύρος – 2^ο Βήμα

Οι διεργασίες 3, 4 θέτουν την 5 ως γονέα με απόσταση 9 και 6 αντίστοιχα.

2^{ος} Γύρος – 2^ο Βήμα



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

3^{ος} Γύρος – 1^ο Βήμα

3^{ος} Γύρος – 1^ο Βήμα

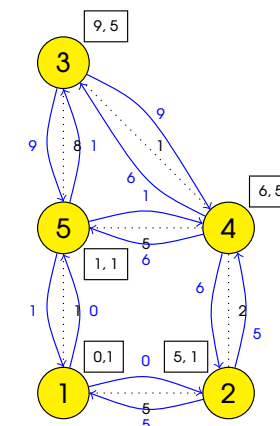
Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.

Η διεργασία 2 στέλνει μήνυμα '5' σε όλους του γείτονες της.

Η διεργασία 3 στέλνει μήνυμα '9' σε όλους του γείτονες της.

Η διεργασία 4 στέλνει μήνυμα '6' σε όλους του γείτονες της.

Η διεργασία 5 στέλνει μήνυμα '1' σε όλους του γείτονες της.



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

3^{ος} Γύρος – 1^ο Βήμα

Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.

Η διεργασία 2 στέλνει μήνυμα '5' σε όλους του γείτονες της.

Η διεργασία 3 στέλνει μήνυμα '9' σε όλους του γείτονες της.

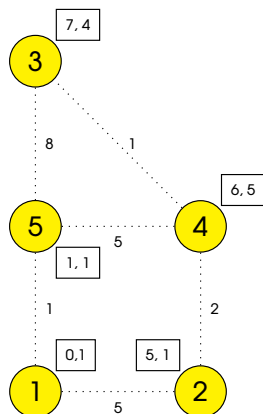
Η διεργασία 4 στέλνει μήνυμα '6' σε όλους του γείτονες της.

Η διεργασία 5 στέλνει μήνυμα '1' σε όλους του γείτονες της.

3^{ος} Γύρος – 2^ο Βήμα

Η διεργασία 3 θέτει την 4 ως γονέα με απόσταση 7.

3^{ος} Γύρος – 2^ο Βήμα



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

4^{ος} Γύρος – 1^ο Βήμα

Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.

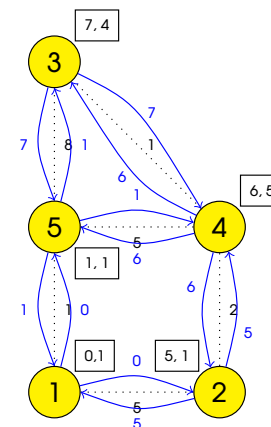
Η διεργασία 2 στέλνει μήνυμα '5' σε όλους του γείτονες της.

Η διεργασία 3 στέλνει μήνυμα '7' σε όλους του γείτονες της.

Η διεργασία 4 στέλνει μήνυμα '6' σε όλους του γείτονες της.

Η διεργασία 5 στέλνει μήνυμα '1' σε όλους του γείτονες της.

4^{ος} Γύρος – 1^ο Βήμα



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

4^{ος} Γύρος – 1^ο Βήμα

Η διεργασία 1 στέλνει μήνυμα '0' σε όλους του γείτονες της.

Η διεργασία 2 στέλνει μήνυμα '5' σε όλους του γείτονες της.

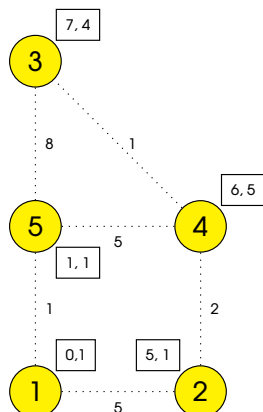
Η διεργασία 3 στέλνει μήνυμα '7' σε όλους του γείτονες της.

Η διεργασία 4 στέλνει μήνυμα '6' σε όλους του γείτονες της.

Η διεργασία 5 στέλνει μήνυμα '1' σε όλους του γείτονες της.

4^{ος} Γύρος – 2^ο Βήμα

4^{ος} Γύρος – 2^ο Βήμα



Παράδειγμα Εκτέλεσης Αλγόριθμου BellmanFord

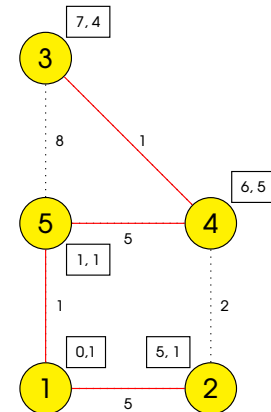
Τελικό Δίκτυο

Το δέντρο ελάχιστων μονοπατιών έχει κατασκευαστεί.

Ο αλγόριθμος εκτελέστηκε σε 4 γύρους.

Συνολικά μεταδόθηκαν 48 μηνύματα.

Τελικό Δίκτυο



Χαρακτηριστικά του Αλγόριθμου BellmanFord

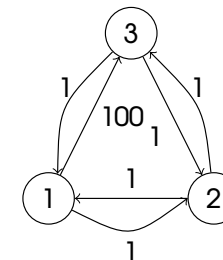
- ▶ Ο αλγόριθμος BellmanFord κατασκευάζει ένα δέντρο ελάχιστων μονοπατιών.
 - ▶ Με επαγωγή στον αριθμό των γύρων
 - ▶ Μετά από γ γύρους, οι μεταβλητές **απόσταση** και **γονέας** για τη διεργασία v αντιστοιχούν στο συντομότερο μονοπάτι που την ενώνει με την u_0 μήκους το πολύ γ
 - ▶ Εάν δεν υπάρχει κάποιο μονοπάτι μήκους το πολύ γ η μεταβλητή **απόσταση** = ∞ και η **γονέας** = 0
 - ▶ Επομένως μετά απο $n - 1$ γύρους, όλες οι διεργασίες θα έχουν εντοπίσει το συντομότερο μονοπάτι που τις ενώνει με την u_0
- ▶ Η χρονική πολυπλοκότητα είναι $\mathcal{O}(n)$
- ▶ Η πολυπλοκότητα επικοινωνίας είναι $\mathcal{O}(n \cdot m)$



Χρονική Πολυπλοκότητα του Αλγόριθμου BellmanFord

- ▶ Αναλογικά με τον αλγόριθμο SynchBFS η χρονική πολυπλοκότητα του BellmanFord περιμένουμε να είναι $diam(G)$
- ▶ Για το παράδειγμα:
 - ▶ η ελάχιστη απόσταση από την διεργασία 1 στην 3 είναι 2
 - ▶ Χρειάζονται 2 γύροι για να βρεθεί
 - ▶ Η διάμετρος του δικτύου είναι 1
- ▶ Η χρονική πολυπλοκότητα είναι $\mathcal{O}(n)$

Παράδειγμα



Παραλλαγές του Αλγόριθμου BellmanFord

- ▶ Μπορούμε να εκτελέσουμε τον αλγόριθμο BellmanFord για β γύρους για να βρούμε όλα τα ελάχιστα μονοπάτια σε απόσταση β
- ▶ Η εύρεση του πλήθους των διεργασιών n μπορεί να γίνει σε συνδυασμό με τον αλγόριθμο SynchBFS



Αρχικές Υποθέσεις

- ▶ Οι ερωτήσεις αντιστοιχούν σε προβλήματα των ΚΣΙ
- ▶ Προσπαθούμε να περιγράψουμε το ΚΣ (όπου εμφανίζεται το πρόβλημα)
- ▶ Η περιγραφή γίνεται με αφαιρετικό τρόπο – οι αρχικές υποθέσεις
- ▶ Ένα πρόβλημα μπορεί να διαφοροποιηθεί αν αλλάξουν οι αρχικές υποθέσεις
 - ▶ είτε να γίνει ποιο εύκολο, είτε ποιο δύσκολο
 - ▶ ακόμα και τελείως διαφορετικό
- ▶ Προτού αρχίσουμε να εξετάζουμε το πρόβλημα πρέπει να καταγράψουμε όλες τις αρχικές υποθέσεις που γίνονται
- ▶ Πρέπει να κατανοήσουμε ακριβώς γιατί δίνεται μια αρχική υπόθεση



Μοντέλο Επικοινωνίας

- ▶ **Σύγχρονο** δίκτυο επικοινωνίας ή **Ασύγχρονο** δίκτυο επικοινωνίας ;
 - ▶ Η πρώτη άσκηση αφορά το σύγχρονο μοντέλο
 - ▶ Η δεύτερη άσκηση αφορά το ασύγχρονο μοντέλο
- ▶ Η πρώτη διαφορά είναι το θέμα 'συγχρονισμού' στην εκτέλεση των βημάτων
- ▶ Η ουσιαστική διαφορά είναι η χρονική απροσδιοριστία
 - ▶ ως προς τους χρόνους εκτέλεσης των βημάτων
 - ▶ ως προς τους χρόνους ανταλλαγής μηνυμάτων
- ▶ Για την μελέτη της απόδοσης στα ασύγχρονα δίκτυα πρέπει να υποθέσουμε κάποια άνω όρια
 - ▶ Η πολυπλοκότητα (χρονική ή μηνυμάτων) εκφράζεται πάντα με βάση αυτών των ορίων



Τοπολογία Δικτύου

- ▶ Το δίκτυο μοντελοποιείται σύμφωνα με ένα γράφημα επικοινωνίας
 - ▶ Κορυφές αντιστοιχούν στις διεργασίες
 - ▶ Ακμές αντιστοιχούν στα κανάλια επικοινωνίας
- ▶ Ποιες είναι οι υποθέσεις για το γράφημα επικοινωνίας ;
- ▶ Είναι ιδιαίτερης κατηγορίας (π.χ. δακτύλιος) ή είναι γενικό ;
- ▶ Κατευθυνόμενο ή μη-κατευθυνόμενο ;
- ▶ Είναι πλήρως συνδεδεμένο ;
- ▶ Γνωρίζουμε το πλήθος των κορυφών και των ακμών ;
- ▶ Πρέπει να κατανοήσουμε ακριβώς το δίκτυο – η λύση μας μπορεί να είναι εντελώς λάθος
- ▶ Συνήθως τα προβλήματα υποθέτουν ένα γενικό, μη-κατευθυνόμενο δίκτυο επικοινωνίας με n διεργασίες και m κανάλια επικοινωνίας



Μοντέλο Σφαλμάτων

- ▶ Υπάρχουν σφάλματα στο σύστημα ;
 - ▶ Σφάλματα στις διεργασίες
 - ▶ Σφάλματα στα κανάλια επικοινωνίας
- ▶ Τα σφάλματα είναι παροδικά ;
- ▶ Γνωρίζουμε το πλήθος των σφαλμάτων που μπορούν να συμβούν κατά την εκτέλεση του αλγορίθμου ;
- ▶ Πρέπει να κατανοήσουμε ακριβώς το μοντέλο σφαλμάτων – η λύση μας μπορεί να είναι εντελώς λάθος
- ▶ Η μελέτη της απόδοσης ίσως να πρέπει να γίνει σε σχέση με το πλήθος σφαλμάτων
- ▶ Συνήθως τα προβλήματα δεν κάνουν υποθέσεις σφαλμάτων – τα μελετήσαμε στην 4η διάλεξη



Αρχικές Γνώσεις Διεργασιών

- ▶ Τι γνωρίζουν οι διεργασίες για το σύστημα ;
 - ▶ Την τοπολογία
 - ▶ Την διάμετρο
 - ▶ Το πλήθος των διεργασιών
- ▶ Οι διεργασίες έχουν (ή δεν έχουν) μοναδικές ταυτότητες
- ▶ Υπάρχει μια διεργασία που την γνωρίζουν όλες οι άλλες (π.χ., u_0)
- ▶ Δίνεται είσοδος κάποια τιμή
 - ▶ ακέραιος αριθμός i_u
 - ▶ σύμφωνα κάποιο σύνολο S
- ▶ Πρέπει να κατανοήσουμε ακριβώς γιατί δίνεται (ή δεν δίνεται) ένα κομμάτι γνώσης



Ζητούμενο – Πρόβλημα

- ▶ Συνήθως είναι προβλήματα σχεδιασμού ενός νέου αλγόριθμου
 - ▶ Υπάρχουν προβλήματα εντοπισμού καλύτερων / χειρότερων διατάξεων διεργασιών
 - ▶ Υπάρχουν προβλήματα απόδειξης συγκεκριμένων συνθηκών
- ▶ Ποιο είναι το ζητούμενο ;
 - ▶ Ποιο πρόβλημα θέλουμε να λύσουμε ;
 - ▶ Τι θέλουμε να 'μάθουν' οι διεργασίες ;
- ▶ Ταιριάζει σε κάποιο από τα προβλήματα που έχουμε μελετήσει ;
 - ▶ Πως διαφοροποιούνται οι αρχικές υποθέσεις ;
 - ▶ Μήπως στις σημειώσεις υποθέτουμε περισσότερη αρχική γνώση ;
 - ▶ Απαιτείται ιδιαίτερη τοπολογία που στην συγκεκριμένη ερώτηση δεν δίνεται ;
- ▶ Πρέπει ο αλγόριθμος να τερματίζει ;



Μεθοδολογία: Καταγραφή Υποθέσεων / Μοντέλου

- ▶ Εντοπισμός αρχικών υποθέσεων
- ▶ Μοντέλο Επικοινωνίας
- ▶ Τοπολογία Δικτύου
- ▶ Σφάλματα
- ▶ Γνώσεις διεργασιών
- ▶ Κατανόηση Προβλήματος
- ▶ Χαρακτηρισμός Προβλήματος



Μεθοδολογία: Αρχική Προσέγγιση

- ▶ Έχουμε κάποια διαίσθηση για την λύση ;
- ▶ Έχουμε πάρει κάποια πρώτη απόφαση για την λύση που θα δώσουμε ;
 - ▶ σκεφτείτε το ξανά !
 - ▶ μήπως μας έχει ξεφύγει κάτι ;
- ▶ Σκιαγραφήστε την λύση
 - ▶ ελέγχουμε αν συμφωνεί με τις αρχικές υποθέσεις
 - ▶ έχουμε χρησιμοποιήσει όλες τις πληροφορίες ;
- ▶ Είναι η λύση σωστή ; μπορούμε να επιχειρηματολογήσουμε ;
- ▶ Ποια είναι η πολυπλοκότητα (χρονική, επικοινωνίας) ;
- ▶ Ποια είναι η ανεκτικότητα σε σφάλματα ;
- ▶ Μήπως υπάρχει καλύτερη λύση ;



Μεθοδολογία: Σχεδιασμός Λύσης

- ▶ Χρειάζεται να τρέξουμε κάποιο αλγόριθμο για να αποκτήσουμε κάποια πληροφορία ;
 - ▶ εκλογή αρχηγού για να προκύψει μια μοναδική διεργασία (π.χ., u_0)
 - ▶ καταμέτρηση για να βρεθεί το πλήθος των διεργασιών (n)
 - ▶ υπολογισμός διαμέτρου
- ▶ Αν το δίκτυο είναι γενικό – απαιτείτε κάποια συγκεκριμένη τοπολογία ;
 - ▶ χρειαζόμαστε ένα δένδρο / δακτύλιο ;
 - ▶ συνήθως κάνει το πρόβλημα 'πιο εύκολο' – ή την λύση 'πιο αποδοτική'
- ▶ Αποτύπωση γενικής ιδέας – 1 παράγραφος
 - ▶ ποια είναι η βασική ιδέα ;
 - ▶ τι κάνει ο αλγόριθμος ;
 - ▶ γιατί είναι σωστός ;



Αποτύπωση Λύσης

- ▶ Αναλυτική καταγραφή μεταβλητών
 - ▶ σκοπός – χρήση
 - ▶ τύπος μεταβλητής
 - ▶ αρχική τιμή
- ▶ Αναλυτική καταγραφή μηνυμάτων
 - ▶ σκοπός – χρήση
 - ▶ περιεχόμενα μηνυμάτων
- ▶ Αρχικοποίηση συστήματος
 - ▶ δημιουργία κάποιας συγκεκριμένης `εικονικής` τοπολογίας
 - ▶ εκτέλεση αλγόριθμου (απόκτηση γνώσης)
 - ▶ αρχικοποίηση μεταβλητών
- ▶ Βασικός γύρος εκτέλεσης
- ▶ Ειδικές περιπτώσεις
- ▶ Τερματισμός
- ▶ Άλλες πληροφορίες



Τελική Απάντηση

1. Σύνομη Περιγραφή
2. Περιγραφή Διεργασιών
 - ▶ μεταβλητές
 - ▶ τύποι μηνυμάτων
 - ▶ αρχικοποίηση
3. Βοηθητικοί Αλγόριθμοι
4. Βασικός Αλγόριθμος – περιγραφή εκτέλεσης
 - ▶ `απλός` / `τυπικός` γύρος εκτέλεσης
 - ▶ ειδικές περιπτώσεις
5. Ψευδοκώδικα (αν υπάρχει χρόνος)
6. Ορθότητα – Συζήτηση . . . Απόδειξη
7. Χρονική Πολυπλοκότητα – Συζήτηση . . . Απόδειξη
8. Πολυπλοκότητα Μηνυμάτων – Συζήτηση . . . Απόδειξη



Εκφώνηση Προβλήματος

Θεωρείστε ένα σύγχρονο καταναμημένο σύστημα με n διεργασίες συνδεδεμένες μέσω ενός γενικού, μη-κατευθυνόμενου δικτύου με m κανάλια επικοινωνίας. Κάθε διεργασία έχει μια μοναδική ταυτότητα και δεν γνωρίζει το σύνολο των διεργασιών, ούτε την τοπολογία του δικτύου. Σχεδιάστε έναν καταναμημένο αλγόριθμο καταμέτρησης που επιπρέπει στη διεργασία u_0 να υπολογίσει το πλήθος των διεργασιών. Αναλύστε την χρονική πολυπλοκότητα και πολυπλοκότητα μηνυμάτων. Αποδείξτε τους ισχυρισμούς σας.



Καταγραφή Υποθέσεων / Μοντέλου

- ▶ Μοντέλο Επικοινωνίας
 - ▶ Σύγχρονο καταναμημένο σύστημα
- ▶ Τοπολογία Δικτύου
 - ▶ Γενικό, μη-κατευθυνόμενο δίκτυο
 - ▶ n διεργασίες, m κανάλια επικοινωνίας
- ▶ Σφάλματα
 - ▶ Κανένα
- ▶ Γνώσεις διεργασιών
 - ▶ Μοναδική ταυτότητα
 - ▶ Υπάρχει μια μοναδική διεργασία u_0



- ▶ Κατανόηση Προβλήματος
 - ▶ Η διεργασία u_0 πρέπει να υπολογίσει τη διάμετρο του δικτύου
 - ▶ Να μετρήσει τις διεργασίες
- ▶ Χαρακτηρισμός Προβλήματος
 - ▶ Κατασκευή ενός δένδρου αναζήτησης κατά εύρος με ρίζα την u_0
 - ▶ Τα φύλλα ξεκινούν τους μετρίες.
 - ▶ Οι γονείς αθροίζουν τους μετρίες των παιδιών και στέλνουν το αποτέλεσμα στους γονείς τους.
- ▶ Αρχική Προσέγγιση
 - ▶ Υπάρχει κάποιος αντίστοιχος αλγόριθμος που μελετήσαμε στο μάθημα ;
 - ▶ Τι τροποποιήσεις πρέπει να κάνουμε ;



Αλγόριθμος SynchBFS_μ (1)

Οι διεργασίες διατηρούν μια μεταβλητή **μαρκαρισμένη** η οποία αρχικά είναι false και μια μεταβλητή **γονέας** με αρχική τιμή 0. Αρχικά, η διεργασία u_0 θέτει την μεταβλητή **μαρκαρισμένη** ως true, την μεταβλητή **γονέας** με την ταυτότητα της, και στέλνει ένα μήνυμα 'αναζήτησης' σε όλους τους γείτονες της. Σε κάθε γύρο, εάν μια διεργασία λάβει ένα μήνυμα 'αναζήτησης' και η τιμή της μεταβλητής **μαρκαρισμένη** είναι

- false** -- θέτει την μεταβλητή **μαρκαρισμένη** true, θέτει την μεταβλητή **γονέας** με την ταυτότητα της διεργασίας από όπου έλαβε το μήνυμα, και στον επόμενο γύρο στέλνει ένα μήνυμα 'αναζήτησης' σε όλους τους γείτονες της.
- true** -- στον επόμενο γύρο, επιστρέφει στον αποστολέα ένα μήνυμα 'μη-γονέας'



Αλγόριθμος SynchBFS_μ (2)

Όταν η διεργασία λάβει ένα μήνυμα 'μη-γονέας' από όλες τις γειτονικές διεργασίες, ή λάβει μήνυμα 'αναζήτησης' από την μοναδική γειτονική διεργασία, στον επόμενο γύρο στέλνει το μήνυμα (γονέας, 1) στον γονέα της.

Όταν η διεργασία λάβει ένα μήνυμα 'μη-γονέας' ή (γονέας, c) από όλες τις γειτονικές διεργασίες, αθροίζει τις μεταβλητές c όλων των μηνυμάτων 'γονέας' που έλαβε, αυξάνει το αποτέλεσμα κατά 1 και στον επόμενο γύρο στέλνει το μήνυμα (γονέας, $\sum(c) + 1$) στον γονέα της.



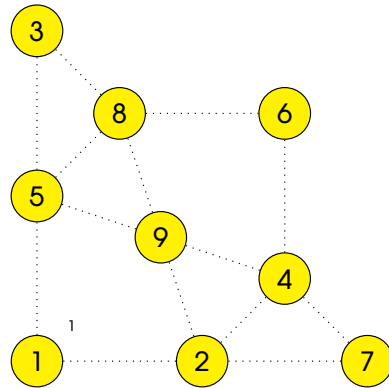
- ▶ Ο αλγόριθμος SynchBFS_τ κατασκευάζει ένα δέντρο αναζήτησης κατά εύρος όπου οι διεργασίες γνωρίζουν τον 'γονέα' τους στο δέντρο και τα 'παιδιά' τους.
- ▶ Ο αλγόριθμος SynchBFS_μ κατασκευάζει ένα δέντρο αναζήτησης κατά εύρος όπου οι διεργασίες γνωρίζουν τον 'γονέα' τους στο δέντρο, τα 'παιδιά' τους και το πλήθος των διεργασιών που ανήκουν στα υποδέντρα των διεργασιών.
- ▶ Η χρονική πολυπλοκότητα είναι $\mathcal{O}(\text{diam}(G))$
- ▶ Η πολυπλοκότητα επικοινωνίας είναι $\mathcal{O}(m)$



Αρχικό Δίκτυο

Το δίκτυο έχει 9 μονάδες, 14 κανάλια
Η διεργασία 1 ξεκινά την εκτέλεση του αλγορίθμου.
Η διεργασία 1 θεωρείται μαρκιαρισμένη.
Όλες οι άλλες διεργασίες δεν είναι μαρκιαρισμένες.

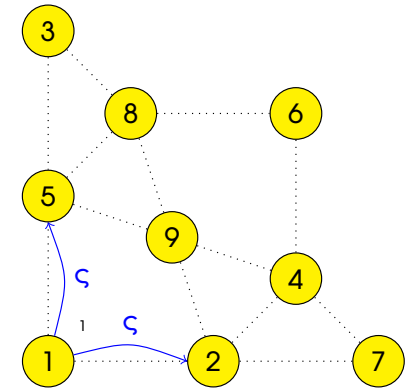
Αρχικό Δίκτυο



1^{ος} Γύρος - 1^ο Βήμα

1^{ος} Γύρος - 1^ο Βήμα

Η διεργασία 1 στέλνει μήνυμα 'αναζήτησης' σε όλους του γείτονες της.



1^{ος} Γύρος - 1^ο Βήμα

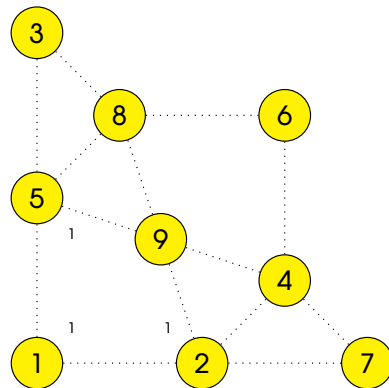
Η διεργασία 1 στέλνει μήνυμα 'αναζήτησης' σε όλους του γείτονες της.

1^{ος} Γύρος - 2^ο Βήμα

Οι διεργασίες 2, 5 μαρκάρονται.

Οι διεργασίες 2, 5 θέτουν την 1 ως γονέα στο δέντρο.

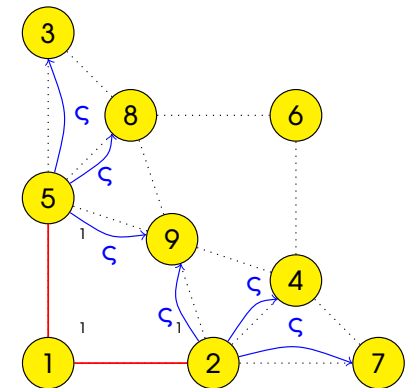
1^{ος} Γύρος - 2^ο Βήμα



2^{ος} Γύρος - 1^ο Βήμα

2^{ος} Γύρος - 1^ο Βήμα

Οι διεργασίες 2, 5 στέλνουν μήνυμα 'αναζήτησης' σε όλους τους γείτονες τους.



2^{ος} Γύρος – 1^ο Βήμα

Οι διεργασίες **2, 5** στέλνουν μήνυμα 'αναζήτησης' σε όλους τους γείτονες τους.

2^{ος} Γύρος – 2^ο Βήμα

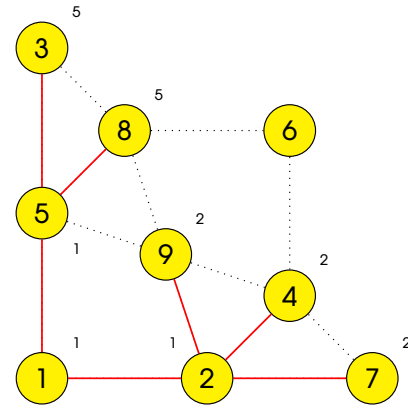
Οι διεργασίες **3, 4, 7, 8, 9** μαρκάρονται.

Οι διεργασίες **3, 8** θέτουν την **5** ως γονέα στο δέντρο.

Οι διεργασίες **4, 7** θέτουν την **2** ως γονέα στο δέντρο.

Η διεργασία **9** διαλέγει τυχαία την **2** ως γονέα στο δέντρο.

2^{ος} Γύρος – 2^ο Βήμα

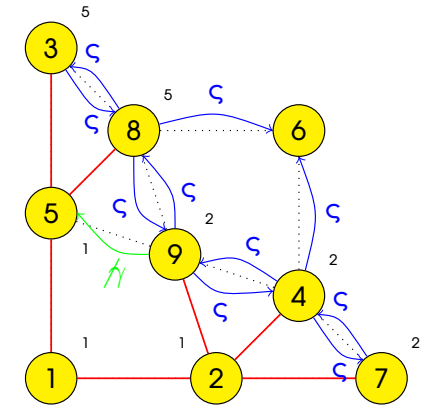


3^{ος} Γύρος – 1^ο Βήμα

3^{ος} Γύρος – 1^ο Βήμα

Οι διεργασίες **3, 4, 7, 8, 9** στέλνουν μήνυμα 'αναζήτησης' σε όλους τους γείτονες τους.

Η διεργασία **9** στέλνει μήνυμα 'μη-γονέας' στην **5**



3^{ος} Γύρος – 1^ο Βήμα

Οι διεργασίες **3, 4, 7, 8, 9** στέλνουν μήνυμα 'αναζήτησης' σε όλους τους γείτονες τους.

Η διεργασία **9** στέλνει μήνυμα 'μη-γονέας' στην **5**

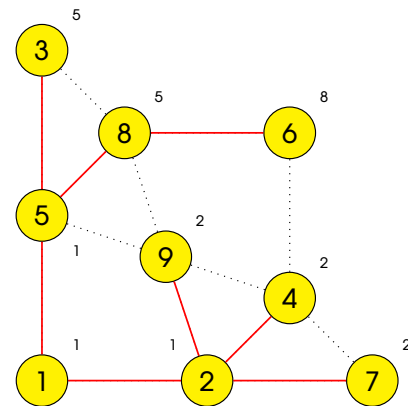
3^{ος} Γύρος – 2^ο Βήμα

Οι διεργασίες **2, 3, 4, 5, 7, 8, 9** αγνοούν τα μηνύματα 'αναζήτησης' που έλαβαν.

Η διεργασία **6** μαρκάρεται.

Η διεργασία **6** διαλέγει τυχαία την **8** ως γονέα στο δέντρο.

3^{ος} Γύρος – 2^ο Βήμα



4^{ος} Γύρος – 1^ο Βήμα

4^{ος} Γύρος – 1^ο Βήμα

Η διεργασία **3** στέλνει μήνυμα 'μη-γονέας' στην **8**

Η διεργασία **8** στέλνει μήνυμα 'μη-γονέας' στην **3**

Η διεργασία **8** στέλνει μήνυμα 'μη-γονέας' στην **9**

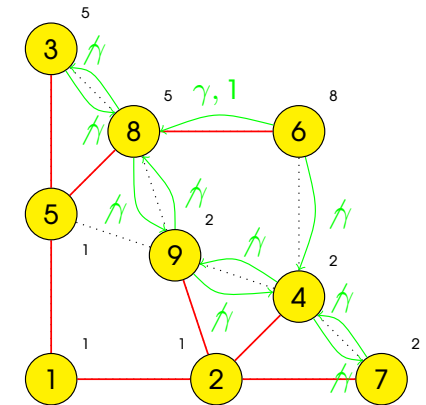
Η διεργασία **9** στέλνει μήνυμα 'μη-γονέας' στην **4**

Η διεργασία **4** στέλνει μήνυμα 'μη-γονέας' στην **7**

Η διεργασία **7** στέλνει μήνυμα 'μη-γονέας' στην **4**

Η διεργασία **6** στέλνει μήνυμα 'μη-γονέας' στην **4**

Η διεργασία **6** στέλνει μήνυμα 'γονέας, 1' στην **8**



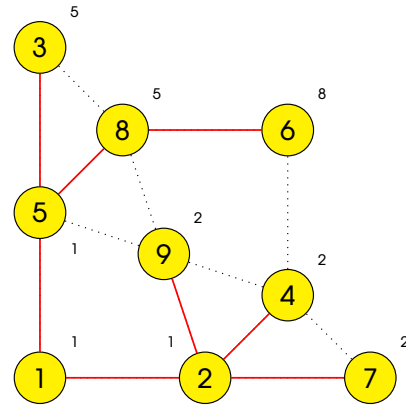
4^{ος} Γύρος – 1^ο Βήμα

- Η διεργασία 3 στέλνει μήνυμα "μη-γονέας" στην 8
- Η διεργασία 8 στέλνει μήνυμα "μη-γονέας" στην 3
- Η διεργασία 8 στέλνει μήνυμα "μη-γονέας" στην 9
- Η διεργασία 9 στέλνει μήνυμα "μη-γονέας" στην 4
- Η διεργασία 4 στέλνει μήνυμα "μη-γονέας" στην 7
- Η διεργασία 7 στέλνει μήνυμα "μη-γονέας" στην 4
- Η διεργασία 6 στέλνει μήνυμα "μη-γονέας" στην 4
- Η διεργασία 6 στέλνει μήνυμα "γονέας,1" στην 8

4^{ος} Γύρος – 2^ο Βήμα

Η διεργασία 8 διαπιστώνει ότι η κατασκευή του υποδέντρου της 6 ολοκληρώθηκε

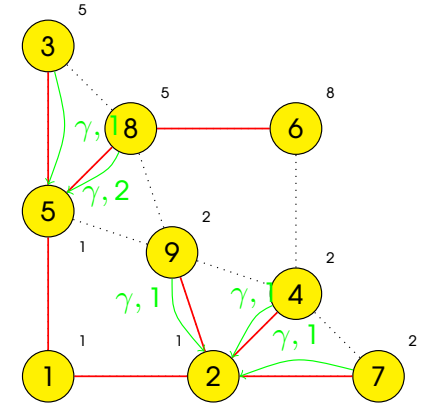
4^{ος} Γύρος – 2^ο Βήμα



5^{ος} Γύρος – 1^ο Βήμα

- Η διεργασία 8 στέλνει μήνυμα "γονέας,2" στην 5
- Η διεργασία 3 στέλνει μήνυμα "γονέας,1" στην 5
- Οι διεργασίες 4, 7, 9 στέλνουν μήνυμα "γονέας,1" στην 2

5^{ος} Γύρος – 1^ο Βήμα



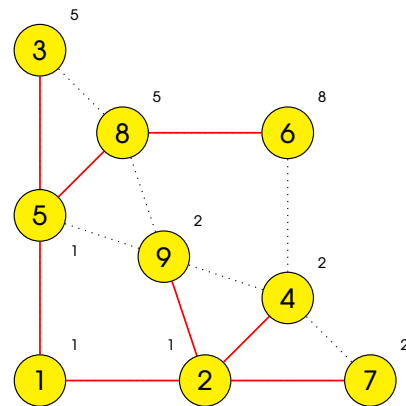
5^{ος} Γύρος – 1^ο Βήμα

- Η διεργασία 8 στέλνει μήνυμα "γονέας,2" στην 5
- Η διεργασία 3 στέλνει μήνυμα "γονέας,1" στην 5
- Οι διεργασίες 4, 7, 9 στέλνουν μήνυμα "γονέας,1" στην 2

5^{ος} Γύρος – 2^ο Βήμα

- Η διεργασία 5 διαπιστώνει ότι η κατασκευή των υποδέντρων των 3, 8 ολοκληρώθηκε
- Η διεργασία 2 διαπιστώνει ότι η κατασκευή των υποδέντρων των 4, 7, 9 ολοκληρώθηκε

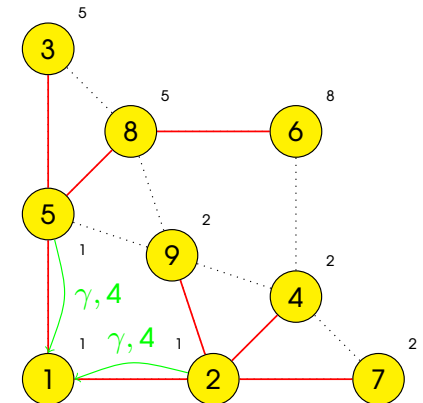
5^{ος} Γύρος – 2^ο Βήμα



6^{ος} Γύρος – 1^ο Βήμα

- Η διεργασία 2 στέλνει μήνυμα "γονέας,4" στην 1
- Η διεργασία 5 στέλνει μήνυμα "γονέας,4" στην 1

6^{ος} Γύρος – 1^ο Βήμα



Βιβλιογραφία

- ▶ Σημειώσεις του μαθήματος
- ▶ Βιβλίο "Distributed Algorithms" (N.Lynch)
 1. Κεφάλαιο 4: Algorithms in General Synchronous Networks
- ▶ Βιβλίο "Distributed Computing Fundamentals, Simulations, and Advanced Topics" (H.Attyia, J.Welch)
 1. Κεφάλαιο 2: Basic Algorithms in Message Passing Systems
- ▶ Βιβλίο "Introduction to Distributed Algorithms" (G.Tel)
 1. Κεφάλαιο 6: Wave and Traversal Algorithms
- ▶ Βιβλίο "Distributed Systems, Concepts and Design" (G.Coulouris, J.Dollimore, T.Kindberg)
 1. Κεφάλαιο 11: Coordination and Agreement



Επόμενο Μάθημα

- ▶ Σύγχρονα Κατανεμημένα Συστήματα
- ▶ Αναζήτηση κατά Βάθος
- ▶ Δέντρα Ελάχιστου Ύψους

