Ενότητα 11
Γράφοι (ή γραφήματα)

Γράφοι

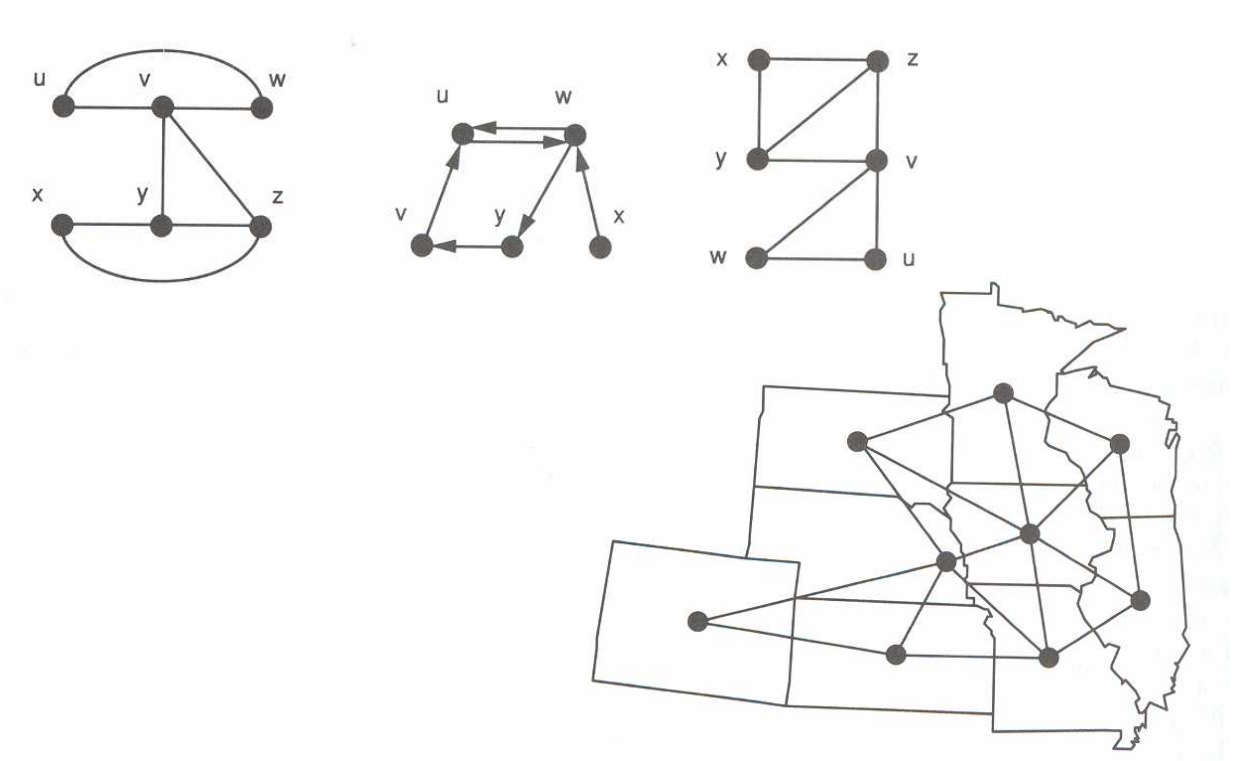
- Ένας γράφος αποτελείται από ένα σύνολο από σημεία (που λέγονται **κόμβοι**) και ένα σύνολο από γραμμές (που λέγονται **ακμές**) οι οποίες συνδέουν ζεύγη κόμβων.

Εφαρμογές

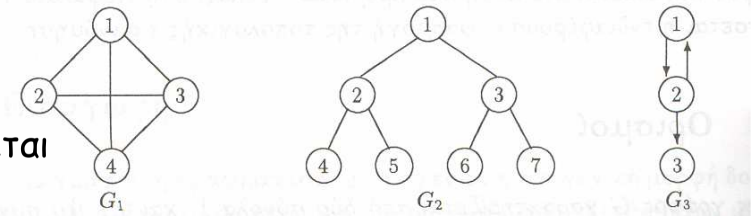
Μοντελοποίηση πολλών προβλημάτων:

- Αεροπορικές πτήσεις μεταξύ κάποιων πόλεων.
- Συνηθίζεται στη δημιουργία χαρτών να ζωγραφίζονται γειτονικές χώρες (νομοί) με διαφορετικό χρώμα. Το πρόβλημα αυτό μπορεί να μοντελοποιηθεί σαν ένα πρόβλημα γράφων.
- Πολλά παιχνίδια μπορούν να μοντελοποιηθούν με χρήση γράφων.
- Traveling Salesman Problem (Πρόβλημα περιπλανώμενου πωλητή): Δεδομένου ενός συνόλου από πόλεις και της απόστασης μεταξύ κάθε ζεύγους πόλεων, βρείτε τη συντομότερη διαδρομή που επισκέπτεται κάθε πόλη (δηλαδή εκείνη στην οποία διανύεται η μικρότερη απόσταση).

Παραδείγματα Γράφων



Χρήσιμη Ορολογία



- Ένας **γράφος** *G* χαρακτηρίζεται από δύο σύνολα V και E.
  Το σύνολο V είναι ένα πεπερασμένο σύνολο, που περιέχει ως στοιχεία τις **κορυφές** (vertices) ή **κόμβους** (nodes) ή **σημεία** (points) του γράφου. Το σύνολο E περιέχει τα ζεύγη κορυφών του γράφου, τα οποία ορίζουν τις **ακμές** (edges) ή **τόξα** (arcs) ή **συνδέσμους** (links) του.
- Οι κορυφές ή οι ακμές ενός γράφου χαρακτηρίζονται από ένα μοναδικό όνομα που ονομάζεται **ετικέτα** (label).

- **V(G) ή V:** το σύνολο των κόμβων ενός γράφου *G*
- **E(G) ή E:** το σύνολο των ακμών ενός γράφου *G*
- **G(V,E):** γράφος με σύνολο κόμβων V και σύνολο ακμών E

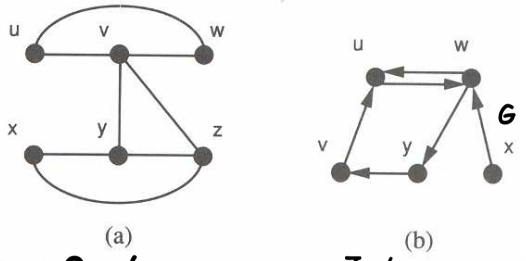
Παραδείγματα

$V(G_1) = \{1,2,3,4\}$, $E(G_1) = \{(1,2),(1,3),(1,4),(2,3),(2,4),(3,4)\}$

$V(G_2) = \{1,2,3,4,5,6,7\}$, $E(G_2) = \{(1,2), (1,3), (2,4), (2,5), (3,6), (3,7)\}$

- **Γράφος με βάρη στις ακμές** (weighted graph) λέγεται ένας γράφος, όπου με κάθε ακμή του έχει συσχετισθεί ένας αριθμός που ονομάζεται **βάρος** (weight).

Χρήσιμη Ορολογία

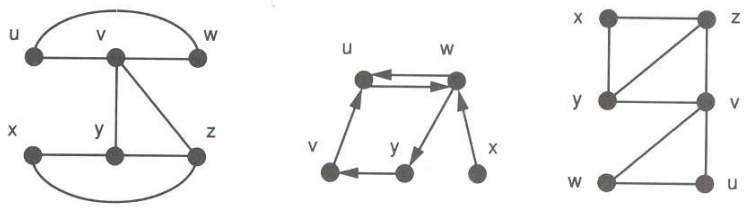


- Ένας γράφος είναι **μη κατευθυνόμενος** αν τα ζεύγη των κορυφών που ορίζουν τις ακμές του στερούνται διάταξης, π.χ., τα ζεύγη ($υ_1$, $υ_2$) και ($υ_2$, $υ_1$) αναφέρονται στην ίδια ακμή.
- Στους **κατευθυνόμενους γράφους** κάθε ακμή συμβολίζεται με το κατευθυνόμενο ζεύγος <$υ_1$, $υ_2$>, όπου $υ_1$ είναι η **ουρά** (tail) και $υ_2$ είναι η **κεφαλή** (head) της ακμής (έτσι, οι ακμές <$υ_1$, $υ_2$> και <$υ_2$, $υ_1$> είναι δυο διαφορετικές ακμές).
- Ένας μη κατευθυνόμενος γράφος μπορεί να θεωρηθεί ως ένας συμμετρικός κατευθυνόμενος γράφος.

Παράδειγμα

- Οι γράφοι *G* και *G*3 είναι κατευθυνόμενοι ενώ ο γράφος του σχήματος (α) είναι μη-κατευθυνόμενος.
- $V(G_3) = \{1,2,3\}$, $E(G_3) = \{<1,2>,<2,1>,<2,3>\}$
- $V(G) = \{u,v,w,x,y\}$, $E(G) = \{<u,w>, <w,u>, <w,y>, <x,w>, <y,v>,<v,u>\}$.

Χρήσιμη Ορολογία



- Αν ($υ_1$, $υ_2$) $\in$ E(*G*), τότε οι κορυφές $υ_1$ και $υ_2$ λέγονται **διπλανές** (adjacent) ή **γειτονικές** (neighboring) και η ακμή ($υ_1$, $υ_2$) ονομάζεται **προσκείμενη** στις κορυφές $υ_1$ και $υ_2$.
- Αν δύο κορυφές $υ_1$ και $υ_2$ δεν συνδέονται μεταξύ τους με ακμή λέγονται **ανεξάρτητες** (independent).
- Αν ($υ_1$, $υ_2$) είναι μια ακμή τότε η κορυφή $υ_1$ λέγεται **διπλανή** (adjacent) της $υ_2$. Επίσης, οι κορυφές $υ_1$, $υ_2$ λέγονται και **γειτονικές**.
- Αν <$υ_1$,$υ_2$> είναι μια ακμή ενός κατευθυνόμενου γράφου, τότε ο κόμβος $υ_1$ είναι **γειτονικός** του κόμβου $υ_2$, αλλά το αντίστροφο ισχύει μόνο αν και η ακμή <$υ_2$, $υ_1$> υπάρχει επίσης στον κατευθυνόμενο γράφο.

- Ένας γράφος με πολλές ακμές λέγεται **πυκνός** (συνήθως με Θ(nlogn) ή περισσότερες ακμές).
- Ένας γράφος με λίγες ακμές (συνήθως λιγότερες από O(n)) λέγεται **αραιός**.

Θεμιτές Λειτουργίες σε Γράφους

- **MakeGraph(V):** επιστρέφει έναν γράφο με σύνολο κορυφών V και καμία ακμή.
- **Vertices(G):** επιστρέφει V(G), το σύνολο των κορυφών του G.
- **Edges(G):** επιστρέφει E(G), το σύνολο των ακμών του G.
- **Neighbors(v,G):** επιστρέφει το σύνολο των κορυφών που είναι γειτονικές του κόμβου v στον G.
- **AddVertex(v,G):** Προσθέτει ένα νέο κόμβο με ετικέτα v στον G.
- **AddDirectedEdge(u,v,G):** προσθέτει μια νέα (κατευθυνόμενη) ακμή <u,v> που συνδέει τους κόμβους u και v στον G.
- **AddUndirectedEdge(u,v,G):** προσθέτει τη νέα ακμή (u,v) που συνδέει τους κόμβους u και v στον G.
- **DeleteVertex(v,G):** διαγράφει τον κόμβο v από τον G, μαζί με όλες τις ακμές που πρόσκεινται σε αυτόν.
- **DeleteEdge(u,v,G):** διαγράφει την ακμή που συνδέει τους κόμβους u και v στον G.

Αναπαράσταση Γράφων

Έστω ένας γράφος G του οποίου οι κόμβοι έχουν αριθμηθεί από το 1 ως το |V| κατά αυθαίρετο τρόπο.

Ο πίνακας γειτνίασης A του G είναι ένας |V|x|V| πίνακας με στοιχεία:

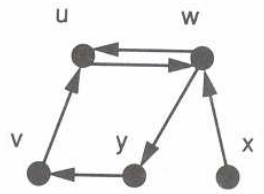
$$A[i,j] = \begin{cases} 1 & \text{αν } (i,j) \in E \\ 0 & \text{διαφορετικά} \end{cases}$$

Ορίζουμε ως ανάστροφο ενός πίνακα γειτνίασης A τον πίνακα $A^T = \{a_{ij}^T\}$, όπου $a_{ij}^T = a_{ji}$, για κάθε $i,j \in \{1,..,|V|\}$.

➡ Ο πίνακας γειτνίασης ενός μη-κατευθυνόμενου γράφου ισούται με τον ανάστροφό του.

- Για κάθε εγγραφή A[i,j] του πίνακα που είναι ίση με 1, η εγγραφή A[j,i] ισούται επίσης με 1.

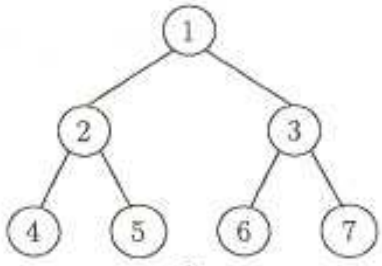
Παραδείγματα



Κατευθυνόμενος γράφος G_1

$$\begin{array}{c|ccccc} & u & v & w & x & y \\ \hline u & 0 & 0 & 1 & 0 & 0 \\ v & 1 & 0 & 0 & 0 & 0 \\ w & 1 & 0 & 0 & 0 & 1 \\ x & 0 & 0 & 1 & 0 & 0 \\ y & 0 & 1 & 0 & 0 & 0 \end{array}$$

Πίνακας γειτνίασης του G_1



G_2

Θετικά & Αρνητικά Πινάκων Γειτνίασης

Θετικά

❑ Αν δεν χρειάζεται να αποθηκευτούν επιπρόσθετες πληροφορίες για κάθε κόμβο του γράφου, η μέθοδος είναι πολύ ελκυστική. Κάθε κόμβος χαρακτηρίζεται από έναν ακέραιο και οι ακέραιοι αυτοί χρησιμοποιούνται για διευθυνσιοδότηση του πίνακα γειτνίασης.

❑ Πολλές λειτουργίες υλοποιούνται απλά και αποτελεσματικά.

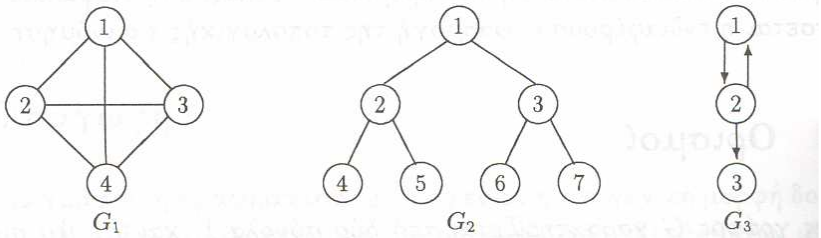
Αρνητικά

❑ Κάθε διαγραφή ή εισαγωγή κόμβου στο γράφο προκαλεί αλλαγή στο μέγεθος του πίνακα.

❑ Η μέθοδος πίνακα γειτνίασης είναι απλή, αλλά δεν υποστηρίζει αποτελεσματικά όλες τις λειτουργίες.

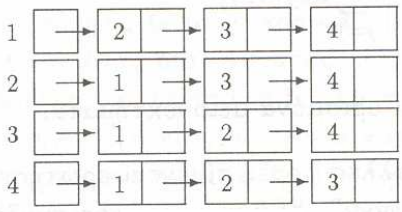
- Ποια η πολυπλοκότητα της λειτουργίας εύρεσης των γειτονικών κόμβων ενός κόμβου υ?
- Θα μπορούσαμε να υλοποιήσουμε τη λειτουργία αυτή πιο αποτελεσματικά αν γνωρίζαμε πως ο κόμβος δεν έχει καθόλου γειτονικούς κόμβους?

Λίστες Γειτνίασης

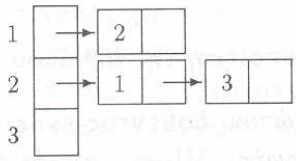


❑ Οι κόμβοι αποθηκεύονται σε μια λίστα. Σε κάθε κόμβο ν της λίστας αποθηκεύεται δείκτης στο πρώτο στοιχείο μιας λίστα που περιέχει τους γειτονικούς κόμβους του κόμβου του γράφου στον οποίο αντιστοιχεί ο ν.

➡ Η λίστα γειτνίασης είναι μια λίστα από λίστες!



Λίστα Γειτνίασης G_1



Λίστα Γειτνίασης G_3

Θετικά

Ένας κόμβος μπορεί να εισαχθεί ή να διαγραφεί με την ίδια ευκολία όπως μια ακμή.
Οδηγεί σε καλή χωρική πολυπλοκότητα για την αναπαράσταση αραιών γράφων.

Αρνητικά

Η χρονική πολυπλοκότητα της λειτουργίας «Εύρεση αν δύο κόμβοι είναι γειτονικοί» είναι μεγαλύτερη από όταν χρησιμοποιείται πίνακας γειτνίασης.
Ακριβή μέθοδος ως προς τη χωρική πολυπλοκότητα για πυκνούς γράφους.

Πολυπλοκότητα Αλγορίθμων Γράφων

Πρόταση: Το μέγιστο πλήθος ακμών για κάθε μη κατευθυνόμενο γράφο με n κορυφές είναι $E_{max} = n(n-1)/2$. Γιατί;

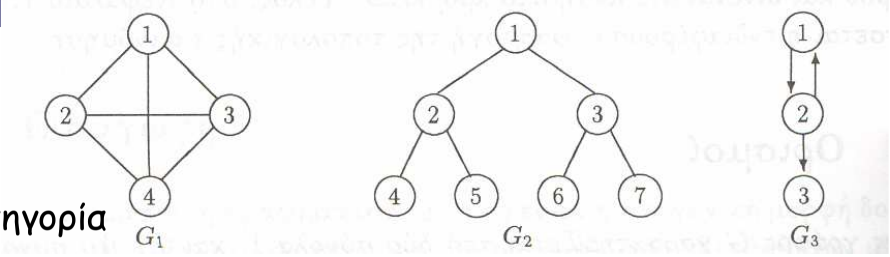
Τι ισχύει αν ένας γράφος *G* είναι κατευθυνόμενος?
Ο G μπορεί να έχει το πολύ διπλάσιο πλήθος ακμών από τον αντίστοιχο μη κατευθυνόμενο γράφο.

Τι θα ήταν καλύτερο, ένας αλγόριθμος που τρέχει σε χρόνο $\Theta(n^2)$ ή σε $\Theta(m)$?

- Αλγόριθμοι με πολυπλοκότητα $\Theta(m)$ συνήθως δεν μπορούν να σχεδιαστούν, αφού αν το $|E|$ είναι μικρό, δεν αρκεί ο χρόνος ούτε για να εξεταστεί κάθε κόμβος.
- Πολλές φορές η γνώση του αν ο γράφος είναι πυκνός ή αραιός βοηθάει στο σχεδιασμό αποτελεσματικών αλγόριθμων.
- Η χρονική πολυπλοκότητα γράφων είναι συνήθως συνάρτηση τόσο του αριθμού των κόμβων, όσο και του αριθμού των ακμών, π.χ., $\Theta(n+m)$.
- Άλλοι παράγοντες που επηρεάζουν σημαντικά την πολυπλοκότητα είναι η υλοποίηση (δηλαδή η μέθοδος αναπαράστασης που χρησιμοποιείται).
  - Πως θα μπορούσε να υλοποιηθεί μια εντολή ανακύκλωσης του τύπου "για κάθε ακμή e στον G" δεδομένου ότι ο G υλοποιείται με:
    - ✓ Πίνακα γειτνίασης;
    - ✓ Λίστες γειτνίασης;

Δένδρα

Τα δένδρα (έχουμε ήδη μελετήσει αρκετά) είναι μια σημαντική κατηγορία γράφων.



Ενδιαφέρουσες Παρατηρήσεις

- ❑ Η έννοια της ρίζας δεν σχετίζεται άμεσα με το αν ένας γράφος είναι δένδρο ή όχι.
- ❑ Σε ένα δένδρο οποιοσδήποτε κόμβος θα μπορούσε να παίξει το ρόλο της ρίζας.

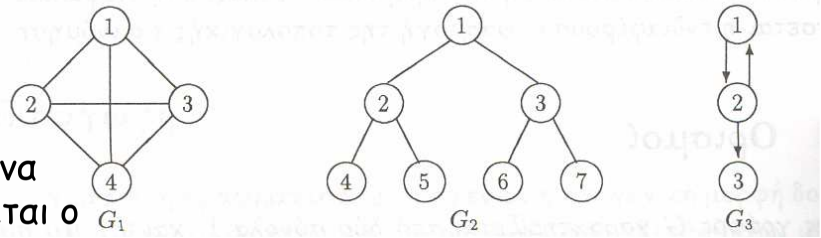
Χρήσιμοι Ορισμοί

Μονοπάτι (path) ή **διαδρομή** (route) από τον κόμβο $υ_p$ στον κόμβο $υ_q$ του γράφου *G* ορίζεται η ακολουθία των κόμβων $υ_p, υ_{i1}, υ_{i2}, \ldots, υ_{in}, υ_q$ που έχουν την ιδιότητα ότι οι ακμές $(υ_p, υ_{i1}), (υ_{i1}, υ_{i2}), \ldots, (υ_{in}, υ_q)$ ανήκουν στο $E(G)$.

Μήκος μονοπατιού (length) είναι το πλήθος των ακμών του μονοπατιού.

Απλό μονοπάτι (simple path) λέγεται το μονοπάτι εκείνο όπου όλοι οι κόμβοι – εκτός ίσως από τον πρώτο και τον τελευταίο - είναι διαφορετικοί.

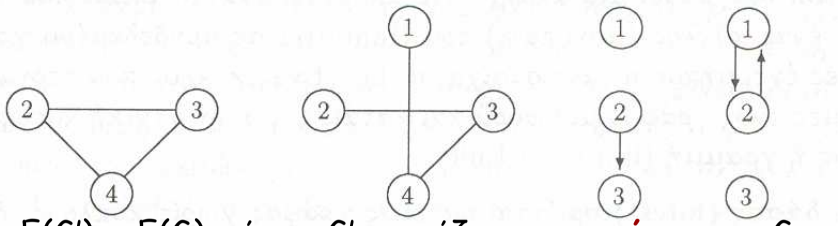
Δένδρα



- **Κύκλος** (cycle) είναι ένα κλειστό μονοπάτι, δηλαδή ένα μονοπάτι στο οποίο ταυτίζεται ο πρώτος και ο τελευταίος κόμβος.
- Ένας **απλός κύκλος** είναι ένας κύκλος που ταυτόχρονα είναι ένα απλό μονοπάτι.
- Ένα δένδρο είναι ένας μη κατευθυνόμενος γράφος T για τον οποίο ισχύει η ακόλουθη ιδιότητα:

  «για κάθε ζεύγος κόμβων u, v του T, υπάρχει ένα και μόνο μονοπάτι που είναι απλό μεταξύ των u και v»
- Σε ένα μη κατευθυνόμενο γράφο *G* δύο κόμβοι λέγονται **συνδεδεμένοι** (connected) όταν υπάρχει ένα μονοπάτι από τον έναν κόμβο στον άλλο.
- Ένας γράφος ονομάζεται **συνδεδεμένος ή συνεκτικός** αν για κάθε ζεύγος κόμβων του γράφου υπάρχει τουλάχιστον ένα μονοπάτι που να τους συνδέει.
- Ένας γράφος που δεν είναι συνεκτικός ονομάζεται **μη-συνεκτικός** (disconnected).
- Ένας γράφος που δεν περιέχει κύκλους ονομάζεται **ακυκλικός** (acyclic).

Δένδρα



Έστω *G* και *G'* δυο οποιοιδήποτε γράφοι.

Αν ισχύει ότι $V(G') \subseteq V(G)$ και $E(G') \subseteq E(G)$, τότε ο *G'* ονομάζεται **υπογράφος** του *G*.

Ένας συνεκτικός υπογράφος *G'* του *G* ονομάζεται **συνεκτική συνιστώσα** (connected component) του *G* αν δεν υπάρχει συνεκτικός υπογράφος *G''* του *G* τ.ω ο *G'* να είναι συνεκτικός υπογράφος του *G''* (*G'* is a maximal connected subgraph of *G*).

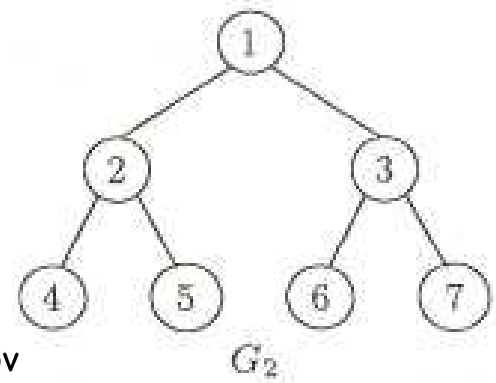
Λήμμα 11.1

1. Ένας συνεκτικός γράφος *G* με n κόμβους έχει τουλάχιστον n-1 ακμές.
2. Ένας ακυκλικός γράφος *G* με n κόμβους έχει το πολύ n-1 ακμές.

Απόδειξη

1. α. Θεωρούμε έναν γράφο *G* με n κόμβους και καμία ακμή.
   β. Προσθέτουμε ακμές μια προς μια στον *G*. Κάθε ακμή είτε συνενώνει δυο κόμβους της ίδιας συνεκτικής συνιστώσας ή συνενώνει δυο κόμβους διαφορετικών συνεκτικών συνιστωσών.
   γ. Άρα, το πλήθος των ακμών που πρέπει να εισαχθούν πριν να προκύψει μόνο μια συνεκτική συνιστώσα είναι τουλάχιστον n-1.
2. α. Έστω ότι ο *G* έχει $k \geq 1$ συνεκτικές συνιστώσες με μεγέθη $n_1, \ldots, n_k$, αντίστοιχα.
   β. Κάθε συνιστώσα C_i είναι ακυκλική και συνεκτική -> η C_i είναι δένδρο -> Η C_i έχει n_i-1 ακμές.
   γ. Συνολικό πλήθος ακμών = $(n_1-1)+(n_2-1)+\ldots+(n_k-1) = n-k \leq n-1$, όπως απαιτείται!

Δένδρα



Θεώρημα 11.2
Έστω ότι T = <V,E> είναι ένα δένδρο.
Τότε το T έχει τις ακόλουθες ιδιότητες:

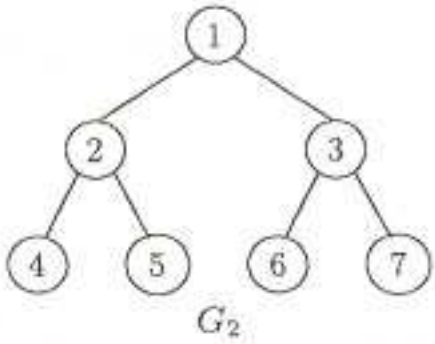
1. ο T είναι ένας συνδεδεμένος γράφος,
2. ο T είναι ένας ακυκλικός γράφος
3. η διαγραφή μιας ακμής από το T καθιστά τον γράφο μη συνεκτικό,
4. αν u,v ∈ V και e = (u,v) δεν είναι ακμή του T, τότε η προσθήκη της e στο T δημιουργεί ένα κύκλο στο γράφο, ο οποίος περιέχει την e,
5. ο T έχει ακριβώς n-1 ακμές, όπου n είναι το πλήθος των κόμβων του T.

Απόδειξη

3. Έστω ότι e = (u,v) είναι οποιαδήποτε ακμή του T. Τότε, (u -> v) είναι το μοναδικό μονοπάτι στο δένδρο από τον κόμβο u στον κόμβο v. Αν η (u,v) αφαιρεθεί από το T, ο προκύπτον γράφος είναι μη συνεκτικός αφού δεν υπάρχει μονοπάτι από τον κόμβο u στον κόμβο v.

Η απόδειξη των υπόλοιπων ιδιοτήτων αφήνεται ως άσκηση για το σπίτι!

Δένδρα



Θεώρημα 11.3
Έστω G ένας μη κατευθυνόμενος γράφος με n κόμβους. Αν ο G ικανοποιεί οποιαδήποτε από τις ακόλουθες συνθήκες, τότε ο G είναι δένδρο:

1. ο G είναι συνεκτικός και ακυκλικός,
2. ο G είναι συνεκτικός, αλλά η διαγραφή μιας οποιασδήποτε ακμής τον μετατρέπει σε μη-συνεκτικό,
3. ο G δεν είναι πλήρης, και η προσθήκη οποιασδήποτε ακμής δημιουργεί ένα και μοναδικό κύκλο στο γράφο,
4. ο G είναι συνεκτικός και έχει ακριβώς n-1 ακμές
5. ο G είναι ακυκλικός και έχει ακριβώς n-1 ακμές.

Απόδειξη
Η απόδειξη των 4 και 5 προκύπτει χρησιμοποιώντας το Λήμμα 1.

Η απόδειξη των 1,2 και 3 αφήνεται ως άσκηση για το σπίτι!

Διάσχιση «Κατά πλάτος» (Breath First Search ή BFS)

Δεδομένων ενός γραφήματος $G=(V,E)$ και ενός κόμβου $s \in V$ (**κόμβος ρίζα** ή **κόμβος εκκίνησης**), η διάσχιση κατά πλάτος συνίσταται στον εντοπισμό (δηλαδή τη διάσχιση) όλων των κόμβων που είναι προσπελάσιμοι από τον s.

Κατά την εκτέλεση μιας διάσχισης κατά πλάτος στον G:

- δημιουργείται ένα δένδρο με ρίζα τον κόμβο s που περιέχει όλους τους κόμβους του G που είναι προσπελάσιμοι από τον s (το δένδρο αυτό ονομάζεται δένδρο «κατά πλάτος»),
- υπολογίζεται η απόσταση, δηλαδή το μήκος του συντομότερου μονοπατιού, από τον s προς οποιονδήποτε κόμβο v (την οποία θα συμβολίζουμε με $\delta(s,v)$).

Η διάσχιση των κόμβων γίνεται «κατά πλάτος»:

«Η διάσχιση οποιουδήποτε κόμβου σε απόσταση $k+1$ από τον s πραγματοποιείται μόνο όταν όλοι οι κόμβοι σε απόσταση k από τον s έχουν διασχισθεί».

Διάσχιση «Κατά πλάτος»

Σημαντικότερα σημεία του αλγορίθμου

- Ο αλγόριθμος χρωματίζει κάθε κόμβο του γράφου λευκό, γκρίζο ή μαύρο.
  - **Λευκοί κόμβοι:** είναι αυτοί που δεν έχουν ακόμη εξερευνηθεί. Ένας κόμβος θεωρείται πως έχει εξερευνηθεί την πρώτη φορά που συναντάται στη διάσχιση, οπότε και καθίσταται μη λευκός.
  - **Γκρίζοι κόμβοι:** είναι κόμβοι που έχουν εξερευνηθεί αλλά ίσως έχουν γείτονες που δεν έχουν ακόμη εξερευνηθεί (δηλαδή που είναι ακόμη λευκοί). Αποτελούν το σύνορο μεταξύ μαύρων και λευκών κόμβων.
  - **Μαύροι κόμβοι:** είναι κόμβοι των οποίων όλοι οι γείτονες έχουν εξερευνηθεί (δηλαδή οι γειτονικοί κόμβοι αυτών είναι είτε γκρίζοι ή μαύροι).
- Ο διαχωρισμός μεταξύ γκρίζων και μαύρων κόμβων γίνεται για να εξασφαλισθεί ότι η διάσχιση των κόμβων θα γίνει «κατά πλάτος».

Διάσχιση «Κατά πλάτος»

Βασικές Ιδέες Αλγορίθμου

- Αρχικά, ο μοναδικός γκρίζος κόμβος είναι ο s.
- Κάθε χρονική στιγμή, οι γκρίζοι κόμβοι είναι αποθηκευμένοι σε μια ουρά Q (που αρχικά περιέχει μόνο τον s).
- Επαναληπτικά, εκτελούνται τα εξής:
  1. Εξαγωγή ενός γκρίζου κόμβου u από την Q;
  2. Διερεύνηση των γειτονικών κόμβων του u και εισαγωγή στην ουρά όσων εξ αυτών είναι λευκοί.
  3. Ακριβώς πριν την εισαγωγή στην Q ενός λευκού κόμβου v εκτελούνται τα εξής:
     - το χρώμα του v αλλάζει σε γκρίζο;
     - ο u ορίζεται να είναι ο πατρικός κόμβος του v στο «κατά πλάτος» δένδρο που δημιουργείται και η ακμή (u,v) εισάγεται στο δένδρο (ο u ονομάζεται **προκάτοχος** του v στο δένδρο);
     - η απόσταση του v από τον κόμβο εκκίνησης s υπολογίζεται και καταχωρείται σε ένα κατάλληλο πεδίο του struct του κόμβου v.
  4. Όταν όλοι οι γειτονικοί κόμβοι του u έχουν εξετασθεί (και οι ενέργειες του βήματος 3 έχουν εκτελεστεί για όσους εξ αυτών είναι λευκοί), το χρώμα του u αλλάζει σε μαύρο.

Διάσχιση «Κατά πλάτος»

Η διάσχιση «κατά πλάτος» σας θυμίζει κάποια από τις διασχίσεις που έχουν μελετηθεί σε δένδρα;

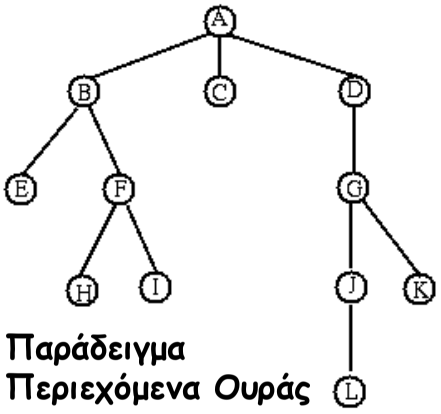
Διάσχιση Δένδρου κατά Επίπεδα (κατά πλάτος)

Επισκέπτεται τους κόμβους κατά αύξον βάθος και τους κόμβους του ίδιου επιπέδου από τα αριστερά προς τα δεξιά.

Χρήση Ουράς

- Αρχικά η ουρά περιέχει μόνο τη ρίζα.
- Επαναληπτικά: κάνουμε Deque ένα στοιχείο της ουράς και προσθέτουμε τα παιδιά από αριστερά προς τα δεξιά του στοιχείου αυτού.

```
Procedure LevelOrder(pointer r) {
  Queue Q;  pointer P;
  MakeEmptyQueue(Q); Enqueue(Q,r);
  while (! IsEmptyQueue(Q)) {
      P = Dequeue(Q);
      Visit(P);
      foreach child c of P, in order, do
            Enqueue(c);
  } }
```



Παράδειγμα

Περιεχόμενα Ουράς

A
B, C, D
C, D, E, F
D, E, F
E, F, G
F, G
G, H, I
H, I, J, K
I, J, K
J, K
K, L
L
<empty>

Διάσχιση «κατά πλάτος»

Ο κώδικας υποθέτει ότι ο γράφος G αναπαρίσταται μέσω λίστας γειτνίασης.

Σε κάθε κόμβο v του γράφου διατηρούνται διάφορες πληροφορίες:

- v->c: το χρώμα του κόμβου v
- v->d: η απόσταση του κόμβου v από τον s
- v->p: ο πατρικός κόμβος του v στο κατά πλάτος δένδρο

Ο αλγόριθμος χρησιμοποιεί μια ουρά Q για τη διαχείριση του συνόλου των γκρίζων κόμβων.

Αναλλοίωτη Συνθήκη

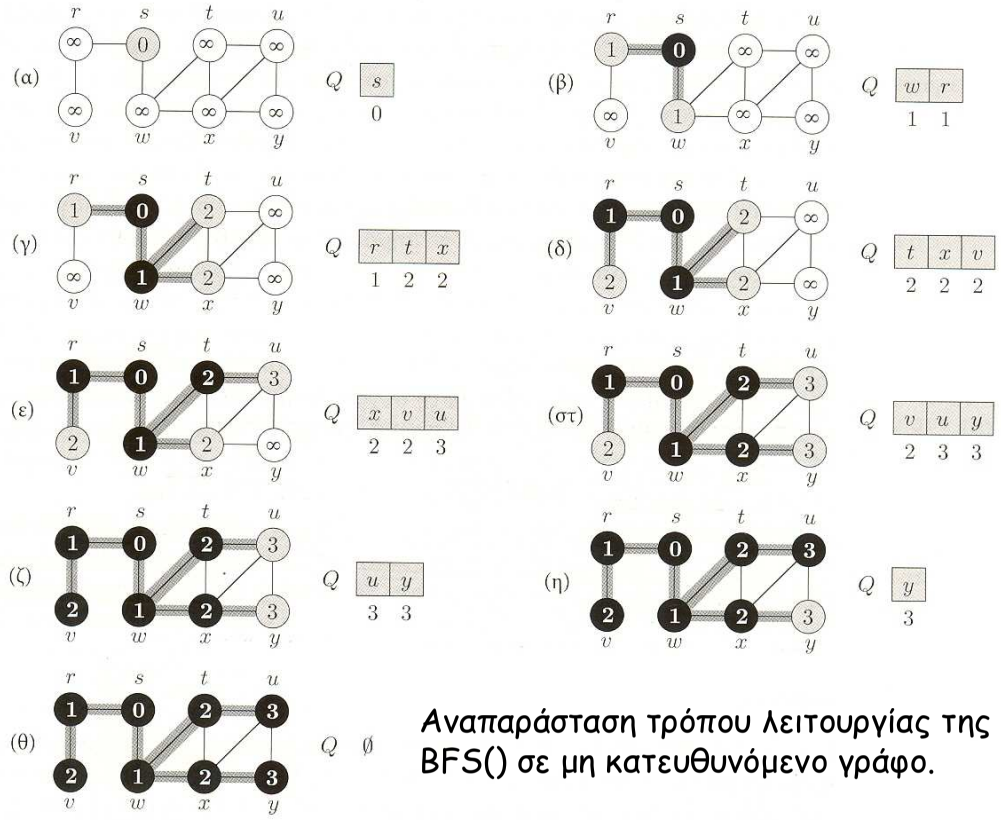
Κάθε φορά που εκτελείται ο έλεγχος της while, η ουρά Q περιέχει το τρέχον σύνολο των γκρίζων κόμβων.

Είναι εύκολο να επιβεβαιωθεί ότι η συνθήκη ισχύει πριν από την πρώτη επανάληψη και διατηρείται σε κάθε επανάληψη!

```
procedure BFS(graph G, node s) {
   Queue Q;
   node u, v;
   foreach node u ∈ V(G) – {s} {        // αρχικοποίηση
      u->c = WHITE;                     // των κόμβων
      u->d = ∞;                         // του V(G)- {s}
      u->p = nill;
   }
   s->c = GRAY;
   s->d = 0;                            // αρχικοποίηση κόμβου
   s->p = nill;                         // s και ουράς Q
   MakeEmptyQueue(Q);
   Enqueue(Q, s);
   while (!IsEmptyQueue(Q)) {
      u = Dequeue(Q);

      // διάσχιση λίστας γειτονικών κόμβων του u
      foreach v ∈ u->Adj {
         if (v->c = WHITE) {
            v->c = GRAY;
            v->d = v->d+1;
            v->p = u;
            Enqueue(Q,v);
         }
      }
      u->c = BLACK;
   }
```


Διάσχιση «Κατά πλάτος»



Αναπαράσταση τρόπου λειτουργίας της BFS() σε μη κατευθυνόμενο γράφο.

Χρονική Πολυπλοκότητα Διάσχισης «Κατά πλάτος»

- Η χρονική πολυπλοκότητα για την αρχικοποίηση των κόμβων του γράφου είναι $O(|V|)$.
- Μετά την εκτέλεση της αρχικοποίησης των κόμβων, κανένας κόμβος δεν χρωματίζεται ποτέ λευκός ⇒ κάθε κόμβος εισέρχεται στην ουρά το πολύ μια φορά!
- Αφού η χρονική πολυπλοκότητα των Enqueue() και Dequeue() είναι $O(1)$, ο συνολικός χρόνος που αναλώνεται στην εκτέλεση των λειτουργιών της ουράς είναι $O(|V|)$.
- Η λίστα γειτνίασης ενός κόμβου διατρέχεται μόνο όταν αυτός διαγράφεται από την ουρά ⇒ η λίστα γειτνίασης κάθε κόμβου διατρέχεται μόνο μια φορά.
- Τα άθροισμα του πλήθους των στοιχείων όλων των λιστών γειτνίασης είναι $O(|E|)$ ⇒ ο συνολικός χρόνος διάσχισης όλων αυτών των λιστών είναι $O(|E|)$.

➡ Επομένως, η χρονική πολυπλοκότητα της BFS() είναι $O(|V| + |E|)$.

```
procedure BFS(graph G, node s) {
   Queue Q;
   node u, v;
   foreach node u ∈ V(G) – {s} {
      u->c = WHITE;
      u->d = ∞;
      u->p = nill;
   }
   s->c = GRAY;
   s->d = 0;
   s->p = nill;
   MakeEmptyQueue(Q);
   Enqueue(Q, s);
   while (!IsEmptyQueue(Q)) {
      u = Dequeue(Q);
      foreach v ∈ u->Adj {
         if (v->c = WHITE) {
            v->c = GRAY;
            v->d = v->d+1;
            v->p = u;
            Enqueue(Q,v);
         }
      }
      u->c = BLACK;
   }
```


Διάσχιση «Κατά πλάτος»

Ένα μονοπάτι μήκους $\delta(s,v)$ από τον κόμβο s σε κάποιο κόμβο v ονομάζεται **συντομότερο μονοπάτι** από τον s στον v. Αν δεν υπάρχει μονοπάτι από τον s στον v, τότε $\delta(s,v) = \infty$.

Θα αποδειχθεί ότι η BFS() υπολογίζει τα μήκη των συντομότερων μονοπατιών από τον s προς οποιονδήποτε κόμβο v.

Έστω $G=(V,E)$ ένας γράφος (κατευθυνόμενος ή μη) και $s \in V$ ένας αυθαίρετος κόμβος.

Λήμμα 11.4: Για κάθε ακμή $(u,v) \in E$, ισχύει ότι $\delta(s,v) \leq \delta(s,u) + 1$.

Απόδειξη: Διακρίνουμε περιπτώσεις:

1. Ο u είναι προσπελάσιμος από τον s. Τότε, ένα από τα μονοπάτια από τον s στον v είναι το μονοπάτι P που αποτελείται από το μονοπάτι από τον s στον u ακολουθούμενο από την ακμή (u,v). Προφανώς, το συντομότερο μονοπάτι από τον s στον v έχει μήκος μικρότερο ή ίσο του μήκους του P.

2. Ο u δεν είναι προσπελάσιμος από τον s. Τότε, $\delta(s,u) = \infty$. Τότε, ούτε ο v είναι προσπελάσιμος από τον s (αφού αν ήταν θα ήταν και ο v). Έτσι, ισχύει ότι $\delta(s,v) = \infty$ και επομένως ο ισχυρισμός ισχύει.

Διάσχιση «Κατά πλάτος»

Λήμμα 11.5: Κατά τον τερματισμό της BFS() για κάθε κόμβο $v \in V$, η τιμή v->d που υπολογίζεται από την BFS() ικανοποιεί τη σχέση $v\text{->}d \geq \delta(s,v)$ (δηλαδή η v->d αποτελεί άνω φράγμα της $\delta(s,v)$).

Απόδειξη: Με επαγωγή ως προς το πλήθος των λειτουργιών Enqueue() που εκτελούνται στην Q.

Βάση Επαγωγής (αμέσως μετά την εισαγωγή του κόμβου s στην ουρά): Από αρχικοποίηση, $s\text{->}d = 0 = \delta(s,s)$ και $\delta(s,v) = \infty$, για κάθε κόμβο $v \in V - \{s\}$. Άρα, ο ισχυρισμός ισχύει.

Επαγωγική Υπόθεση: Μετά από την εκτέλεση $k \geq 1$ λειτουργιών Enqueue() στην Q, ισχύει ότι, για κάθε κόμβο $v \in V$, $v\text{->}d \geq \delta(s,v)$.

Επαγωγικό Βήμα: Θα δείξουμε ότι ο ισχυρισμός ισχύει μετά την εκτέλεση k+1 λειτουργιών Enqueue() στην Q.

Έστω v ο λευκός κόμβος που εισάγεται κατά την εκτέλεση της (k+1)-οστής λειτουργίας Enqueue() στην Q και έστω ότι ο v εντοπίστηκε στη λίστα γειτνίασης ενός κόμβου u.

Σύμφωνα με την επαγωγική υπόθεση ισχύει ότι $u\text{->}d \geq \delta(s,u)$.

Από τον κώδικα προκύπτει ότι $v\text{->}d = u\text{->}d+1 \geq \delta(s,u)+1 \geq \delta(s,v)$.

Στη συνέχεια ο v προστίθεται στην ουρά για μια και μοναδική φορά (αφού ταυτόχρονα χρωματίζεται γκρίζος) και έτσι το v->d δεν αλλάζει ξανά και άρα η επαγωγική υπόθεση εξακολουθεί να ισχύει.

Διάσχιση «Κατά πλάτος»

Λήμμα 11.6: Έστω ότι σε κάποια χρονική στιγμή της εκτέλεσης της BFS(), η ουρά Q περιέχει τους κόμβους $\langle v_1, v_2, \ldots v_r \rangle$, όπου ο v_1 είναι ο πρώτος κόμβος (κεφαλή) της Q και ο v_r είναι ο τελευταίος κόμβος της Q. Τότε, $v_r\text{->}d \leq v_1\text{->}d + 1$ και $v_1\text{->}d \leq v_2\text{->}d \leq \ldots \leq v_{r-1}\text{->}d$ (δηλαδή ανά πάσα στιγμή υπάρχουν το πολύ δύο διαφορετικές τιμές στα πεδία d των κόμβων που περιέχονται στην ουρά Q).

Απόδειξη: Με επαγωγή στο πλήθος των Enqueue() & Dequeue() που εκτελούνται στην Q.

Βάση Επαγωγής: Όταν η ουρά περιέχει μόνο τον κόμβο s, ο ισχυρισμός ισχύει προφανώς.

Επαγωγική Υπόθεση: Έστω ότι ο ισχυρισμός ισχύει μετά από την εκτέλεση k λειτουργιών στην Q.

Επαγωγικό Βήμα: Θα δείξω ότι ο ισχυρισμός ισχύει μετά από την εκτέλεση k+1 λειτουργιών στην ουρά. Διακρίνω περιπτώσεις:

1. Η (κ+1) λειτουργία στην Q είναι Dequeue() $\Rightarrow$ ο κόμβος v_1 διαγράφεται από την Q. Από επαγωγική υπόθεση $v_1\text{->}d \leq v_2\text{->}d$ και $v_r\text{->}d \leq v_1\text{->}d + 1$. Επομένως, $v_r\text{->}d \leq v_2\text{->}d+1$, ενώ οι υπόλοιπες ανισότητες δεν επηρεάζονται.
2. Η (k+1) λειτουργία στην Q είναι Enqueue(). Στην ουρά προστίθεται ένας κόμβος v_{r+1}, ο οποίος έστω ότι βρίσκεται στη λίστα γειτνίασης του κόμβου u. Ο u είναι ο κόμβος που έχει εξαχθεί πιο πρόσφατα από την ουρά και αν v_1 είναι ο τρέχον κόμβος που είναι πρώτος στην ουρά, από την επαγωγική υπόθεση ισχύει ότι $u\text{->}d \leq v_1\text{->}d$. Επομένως, $v_{r+1}\text{->}d = u\text{->}d + 1 \leq v_1\text{->}d + 1$. Από επαγωγική υπόθεση, $v_r\text{->}d \leq u\text{->}d + 1 = v_{r+1}\text{->}d$.

Οι υπόλοιπες ανισότητες δεν επηρεάζονται.

Διάσχιση «Κατά πλάτος»

Πόρισμα 11.7: Έστω ότι κατά την εκτέλεση της BFS() προστίθενται στην ουρά οι κόμβοι v_i και v_j και ότι ο v_i προστίθεται στην Q πριν από τον v_j. Τότε, τη χρονική στιγμή που προστίθεται ο v_j ισχύει ότι v_i->d $\leq$ v_j->d.

Ορθότητα της BFS()

Θεώρημα 11.8: (α) Η BFS() διασχίζει όλους τους κόμβους $v \in V$ που είναι προσπελάσιμοι από τον s. Κατά τον τερματισμό της ισχύει ότι (β) v->d = δ(s,v), για κάθε κόμβο $v \in V$ και (γ) για κάθε κόμβο $v \in V$ που είναι προσπελάσιμος από τον s, ένα από τα συντομότερα μονοπάτια από τον s στον v αποτελείται από το συντομότερο μονοπάτι από τον s ως τον v->p ακολουθούμενο από την ακμή (v->p, v).

Ορθότητα της Διάσχισης «Κατά Πλάτος»

Απόδειξη Θεωρήματος 11.8

- Έστω ότι για κάποιο κόμβο δεν υπολογίζεται σωστά το d. Έστω v ο κόμβος με την ελάχιστη δ(s,v) ο οποίος λαμβάνει μια λανθασμένη τιμή d. Προφανώς, $v \neq s$ και επίσης ο v είναι προσπελάσιμος από τον s (αφού διαφορετικά δ(s,v) = ∞ > v->d ότι τιμή και να έχει το v->d).
- Λόγω Λήμματος 11.5, v->d $\geq$ δ(s,v) $\Rightarrow$ v->d > δ(s,v).
- Έστω u ο αμέσως προηγούμενος κόμβος σε ένα συντομότερο μονοπάτι από τον s στον v. Άρα, δ(s,v) = δ(s,u)+1.
- Αφού δ(s,u) < δ(s,v) και λόγω του τρόπου που επιλέχθηκε ο v, θα πρέπει να ισχύει πως u->d = δ(s,u).
- Επομένως προκύπτει ότι: v->d > δ(s,v) = u->d + 1 (1)
- Εξετάζουμε τι συμβαίνει τη χρονική στιγμή που η BFS() διαγράφει τον u από την Q. Διακρίνουμε περιπτώσεις (και αποδεικνύουμε ότι σε κάθε μια από αυτές καταστρατηγείται η ανισότητα (1), το οποίο είναι άτοπο).

Ορθότητα της Διάσχισης «Κατά Πλάτος»

Απόδειξη Θεωρήματος 11.8 (συνέχεια)

1. Ο v είναι λευκός. Τότε, ο v εισάγεται στην ουρά λόγω του u και έτσι v->d = u->d + 1 (το οποίο αντιτίθεται στην (1)).
2. Ο v είναι γκρίζος. Ο v χρωματίστηκε γκρίζος κατά την αφαίρεση από την ουρά κάποιου κόμβου w, ο οποίος αφαιρέθηκε από την Q νωρίτερα από τον u. Επίσης ισχύει ότι v->d = w->d + 1. Από Πόρισμα 11.7 $\Rightarrow$ w->d $\leq$ u->d. Επομένως, v->d $\leq$ u->d + 1 (το οποίο και πάλι αντιτίθεται στην (1)).
3. Ο v είναι μαύρος. Ο v έχει ήδη αφαιρεθεί από την ουρά. Βάσει Πορίσματος 11.7 $\Rightarrow$ v->d $\leq$ u->d (το οποίο και πάλι αντιτίθεται στην (1)).

- Συμπεραίνουμε ότι v->d = δ(s,v) για κάθε κόμβο v $\in$ V.
- Όλοι οι κόμβοι που είναι προσπελάσιμοι από τον s θα πρέπει να εντοπιστούν (γιατί διαφορετικά v->d = ∞ για όσους κόμβους δεν εντοπίζοναι, το οποίο αντιτίθεται στον ισχυρισμό που μόλις δείξαμε).
- Από κώδικα προκύπτει πως αν v->p = u, τότε v->d = u->d + 1.
- Επομένως, αν πάρουμε ένα συντομότερο μονοπάτι από τον s μέχρι τον u = v->p και προσθέσουμε την ακμή (v->p, v) παίρνουμε ένα συντομότερο μονοπάτι από τον s μέχρι τον v.