Αλγόριθμοι και Πολυπλοκότητα

Γραφήματα

Δημήτρης Μιχαήλ

Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο

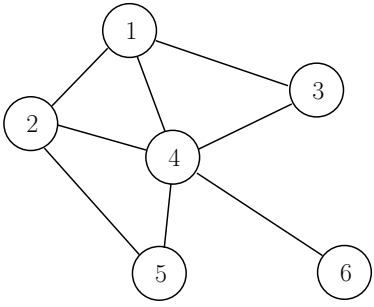
Γραφήματα

Μη Κατευθυνόμενο Γράφημα $G(V, E)$

- V σύνολο κόμβων
- E σύνολο ακμών (ζεύγοι κόμβων)

Κωδικοποίηση των σχέσεων ανά ζεύγη μεταξύ των αντικειμένων ενός συνόλου.

Παράμετροι Μεγέθους. $n = |V|$, $m = |E|$

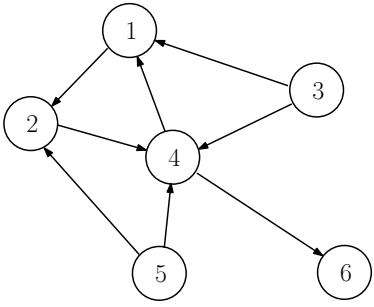


$$V = \{1, 2, 3, 4, 5, 6\}$$
$$E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 4\}, \{4, 3\}, \{2, 5\}, \{4, 5\}, \{4, 6\}\}$$
$$n = 6$$
$$m = 8$$

Κατευθυνόμενα Γραφήματα

Κατευθυνόμενο Γράφημα $G(V, E)$

- V σύνολο κόμβων
- E σύνολο ακμών (διατεταγμένα ζεύγη κόμβων)
- $n = |V|$, $m = |E|$



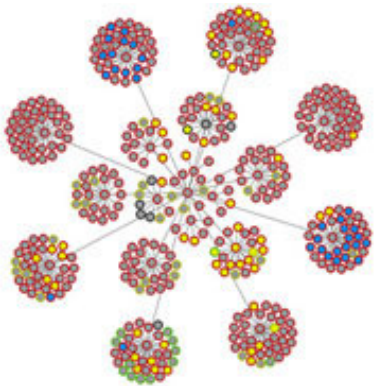
$$V = \{1, 2, 3, 4, 5, 6\}$$
$$E = \{(1, 2), (3, 1), (4, 1), (2, 4), (3, 4), (5, 2), (5, 4), (4, 6)\}$$
$$n = 6$$
$$m = 8$$

Παραδείγματα

Παγκόσμιος Ιστός

Κατευθυνόμενο γράφημα όπου

- κόμβοι είναι οι σελίδες (webpages)
- ακμές είναι οι σύνδεσμοι (hyperlinks)

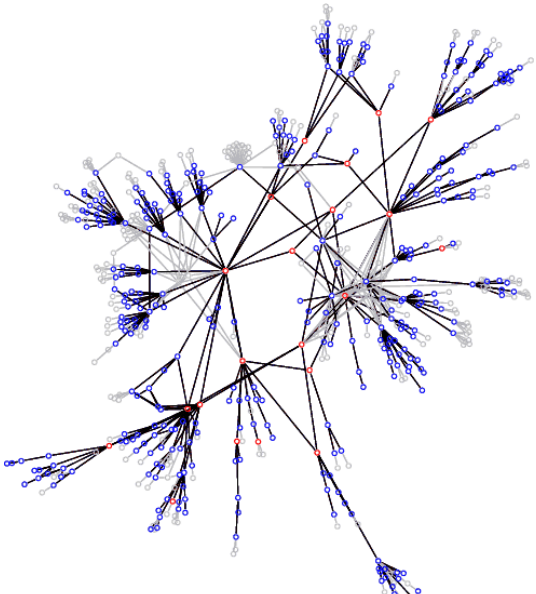


Παραδείγματα

Δίκτυα Επικοινωνιών

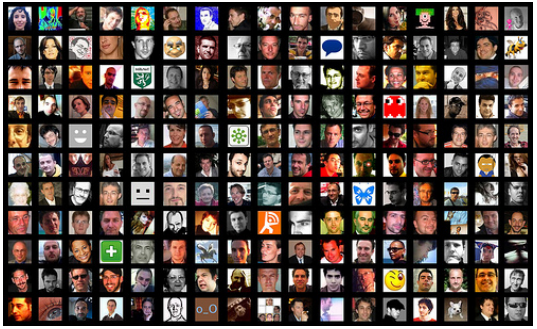
Κατευθυνόμενο γράφημα όπου

- κόμβοι είναι οι υπολογιστές
- ακμές που συνδέουν κόμβους u και v εαν υπάρχει άμεση φυσική σύνδεση μεταξύ τους



Παραδείγματα

Κοινωνικά Δίκτυα (social networks)

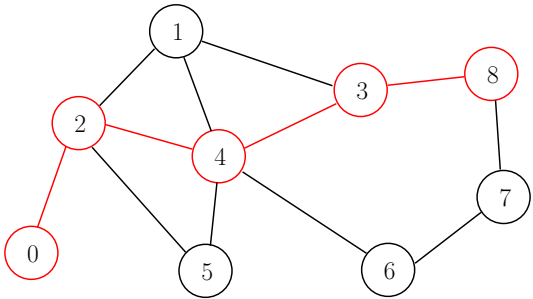


Κατευθυνόμενο γράφημα όπου

- κόμβοι είναι οι άνθρωποι
- ακμές συνδέουν κόμβους u και v εαν υπάρχει κάποια σχέση, π.χ φιλία, επαγγελματική σχέση, συνεργασία, κ.τ.λ

Ορισμοί σε (μη κατευθυνόμενα) Γραφήματα

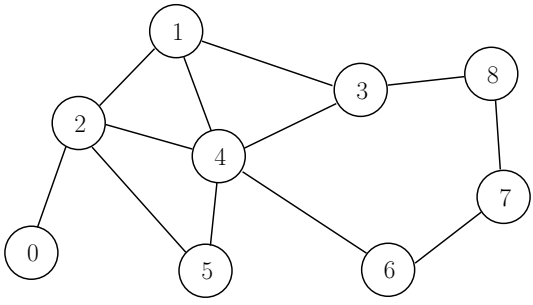
διαδρομή (path). Μια ακολουθία P από κόμβους $v_1, v_2, \ldots, v_{k-1}, v_k$ όπου κάθε διαδοχικό ζευγάρι v_i, v_{i+1} συνδέεται με μια ακμή.



$P = v_0, v_2, v_4, v_3, v_8$

Ορισμοί σε (μη κατευθυνόμενα) Γραφήματα

απλή διαδρομή (simple path). Μια διαδρομή ονομάζεται **απλή** αν όλες οι κορυφές της είναι διακριτές η μία από την άλλη.



απλή διαδρομή

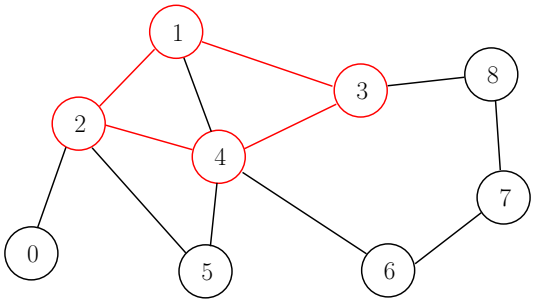
$$v_0, v_2, v_1, v_4, v_3, v_8$$

και μη απλή διαδρομή

$$v_0, v_2, v_1, v_4, v_2, v_1, v_3, v_8$$

Ορισμοί σε (μη κατευθυνόμενα) Γραφήματα

κύκλος (cycle). Μια διαδρομή $v_1, v_2, \ldots, v_{k-1}, v_k$ στην οποία $k > 2$, οι πρώτοι $k-1$ κόμβοι είναι όλοι διακριτοί και $v_1 = v_k$.

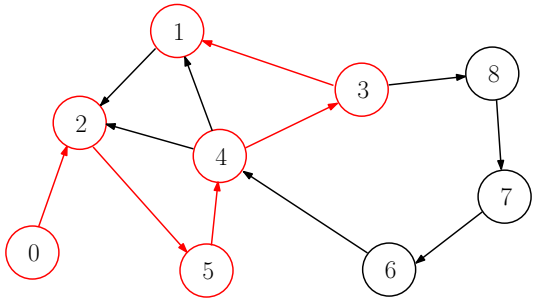


π.χ κύκλος

v_2, v_1, v_3, v_4, v_2

Σε Κατευθυνόμενα Γραφήματα;

Οι ορισμοί μεταφέρονται φυσιολογικά στα κατευθυνόμενα γραφήματα φροντίζοντας να διατρέχουμε τις ακμές ως προς τον προσανατολισμό τους.



π.χ διαδρομή

$$v_0, v_2, v_5, v_4, v_3, v_1$$

Συνεκτικότητα Γραφήματος

Μη-κατευθυνόμενα.
Ένα μη-κατευθυνόμενο γράφημα είναι **συνεκτικό** ή **συνδεδεμένο** (connected) αν, για κάθε ζευγάρι κόμβων u και v, υπάρχει διαδρομή από το u στο v.

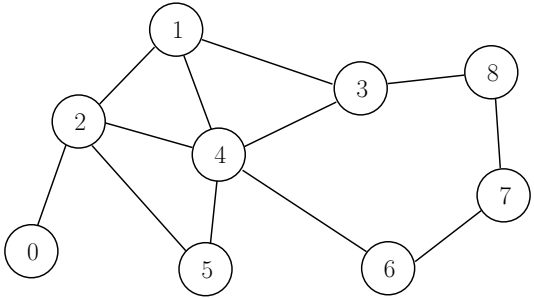
Κατευθυνόμενα.
Πιο πολύπλοκος ορισμός αφού γίνεται να υπάρχει διαδρομή από το u στον v αλλά όχι από το v στο u.

Ένα κατευθυνόμενο γράφημα είναι **αυστηρά συνεκτικό** (strongly connected) αν, για κάθε ζευγάρι κόμβων u και v, υπάρχει διαδρομή από το u στο v και διαδρομή από το v στο u.

Απόσταση

Απόσταση. Η απόσταση (distance) μεταξύ δύο κόμβων u και v είναι ο ελάχιστος αριθμός ακμών σε μια διαδρομή $u \rightsquigarrow v$.

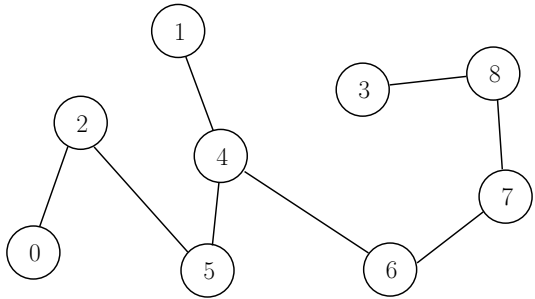
Άπειρη απόσταση. Εάν δεν υπάρχει διαδρομή μεταξύ δύο κόμβων χρησιμοποιούμε το σύμβολο ∞ για να το συμβολίσουμε.



Η απόσταση του v_5 και του v_7 είναι 3 αφού η συντομότερη διαδρομή είναι v_5, v_4, v_6, v_7.

Δέντρα

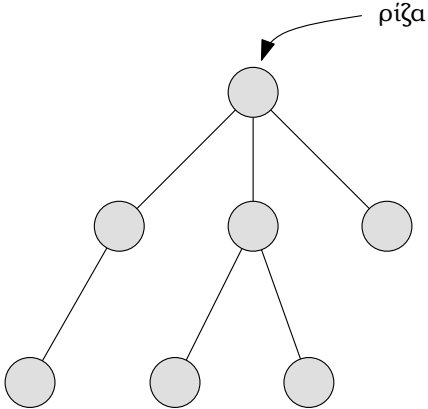
Ένα μη-κατευθυνόμενο γράφημα είναι **δέντρο** (tree) αν είναι συνεκτικό και δεν περιέχει κύκλο.



Η αφαίρεση οποιασδήποτε ακμής από το δέντρο θα το αποσυνδέσει (δεν θα είναι πλέον συνδεδεμένο ή συνεκτικό).

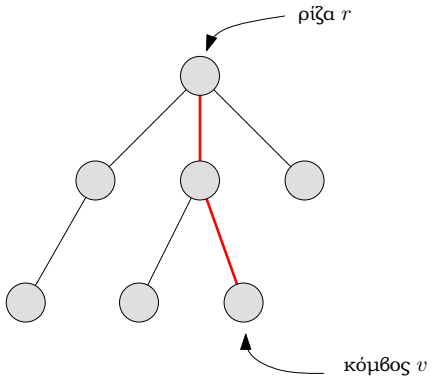
Δέντρα

δέντρο με ρίζα (rooted tree) Ορίζουμε ένα κόμβο ως ρίζα. Συνήθως ζωγραφίζουμε αυτόν τον κόμβο πιο ψηλά.



Δέντρα

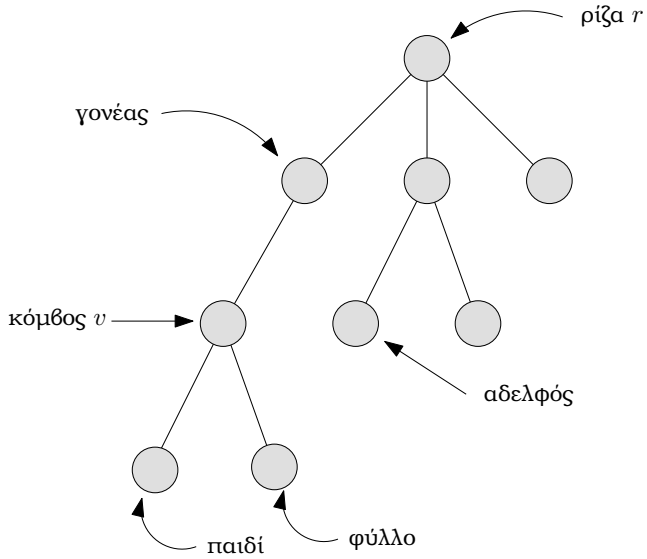
Μεταξύ ενός κόμβου *v* και της ρίζας *r* υπάρχει ένα μοναδικό μονοπάτι.



Ο αριθμός των ακμών του μονοπατιού αυτού καθορίζει το ύψος του κόμβου *v* στο δέντρο.

Δέντρα

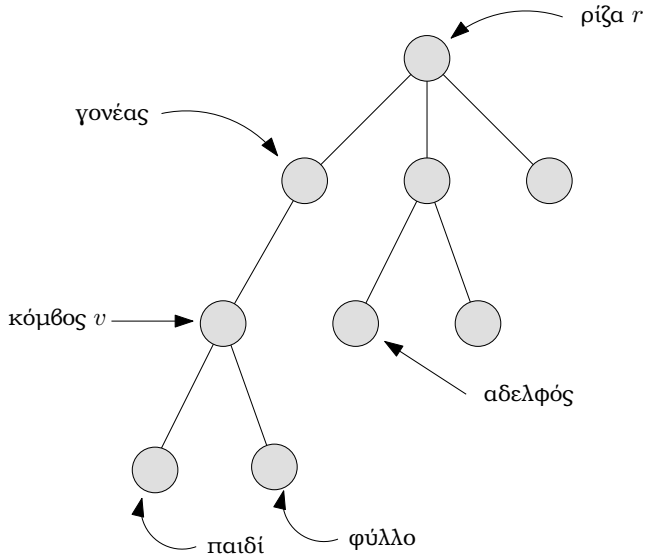
Έστω ένας κόμβος v και η ρίζα r.



Ο κόμβος ακριβώς πριν τον v στο μονοπάτι $r \rightsquigarrow v$ λέγεται πατέρας του v.

Δέντρα

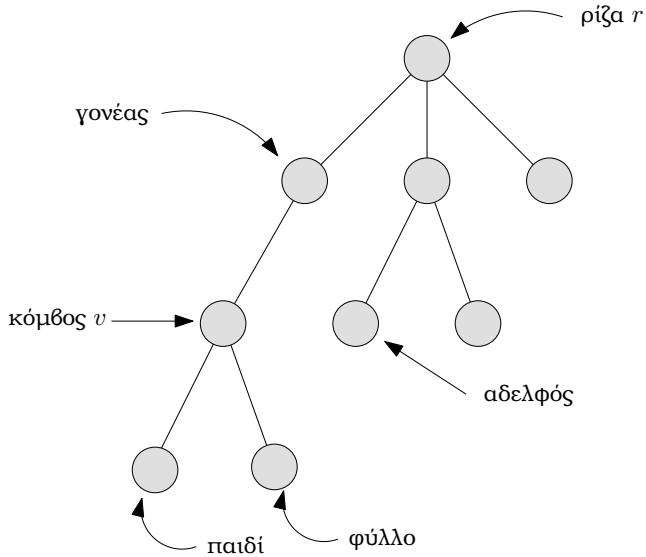
Έστω ένας κόμβος v και η ρίζα r.



Οι κόμβοι ακριβώς κάτω από τον v είναι τα παιδιά του v.

Δέντρα

Έστω ένας κόμβος *v* και η ρίζα *r*.

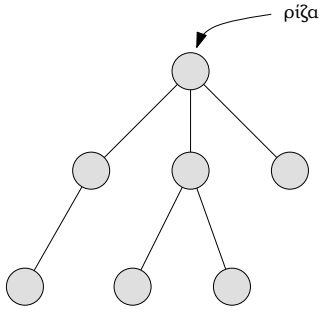


Κόμβος χωρίς απογόνους λέγεται *φύλλο* (leaf).

Δέντρα

Λήμμα

Κάθε δέντρο με n κόμβους έχει ακριβώς $n - 1$ ακμές.



Συνεκτικότητα Γραφήματος

Πρόβλημα προσδιορισμού συνεκτικότητας $s - t$.
Δεδομένου ενός γραφήματος $G(V, E)$ και δύο κόμβων s και t υπάρχει διαδρομή από τον s στον t στο γράφημα;

Συνεκτικότητα Γραφήματος

Πρόβλημα προσδιορισμού συνεκτικότητας $s - t$.

Δεδομένου ενός γραφήματος $G(V, E)$ και δύο κόμβων s και t υπάρχει διαδρομή από τον s στον t στο γράφημα;

Λαβύρινθος. Φανταστείτε ένα λαβύρινθο όπου κάθε κόμβος του γραφήματος είναι ένα δωμάτιο και κάθε ακμή ένας διάδρομος που ενώνει κόμβους (δωμάτια). Θέλουμε να βρούμε μια διαδρομή από ένα δωμάτιο s σε ένα άλλο δωμάτιο t.

Συνεκτικότητα Γραφήματος

Πρόβλημα προσδιορισμού συνεκτικότητας $s - t$.
Δεδομένου ενός γραφήματος $G(V, E)$ και δύο κόμβων s και t υπάρχει διαδρομή από τον s στον t στο γράφημα;

Πόσο αποδοτικά μπορούμε να λύσουμε το παραπάνω πρόβλημα; Θα δούμε σε υψηλό επίπεδο δύο φυσικούς αλγορίθμους για το πρόβλημα:

1. αναζήτηση πρώτα κατά πλάτος (breadth-first search, BFS)
2. αναζήτηση πρώτα κατά βάθος (depth-first search, DFS)

Αναζήτηση Πρώτα Κατά Πλάτος

Breadth-First Search, BFS

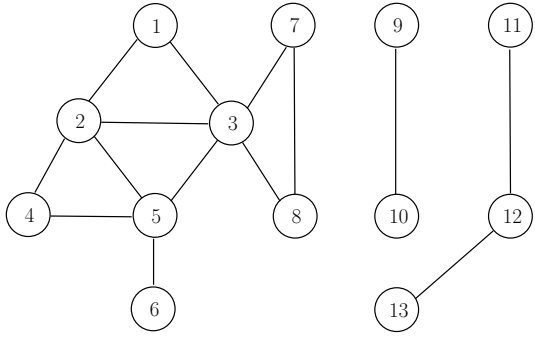
Επίπεδα. Εξερευνούμε από τον s προς όλες τις κατευθύνσεις προσθέτοντας κάθε φορά κόμβους κατά ”επίπεδα”.

Αλγόριθμος.

1. ξεκινάμε με τον κόμβο s
2. συμπεριλαμβάνουμε στο πρώτο επίπεδο αναζήτησης όλους τους κόμβους που συνδέονται με τον s με ακμή
3. περιλαμβάνουμε όλους τους επιπρόσθετους κόμβους που συνδέονται με ακμή με οποιονδήποτε κόμβο του πρώτου επιπέδου – αυτό είναι το δεύτερο επίπεδο.
4. συνεχίζουμε μέχρι να μην συναντήσουμε άλλους κόμβους

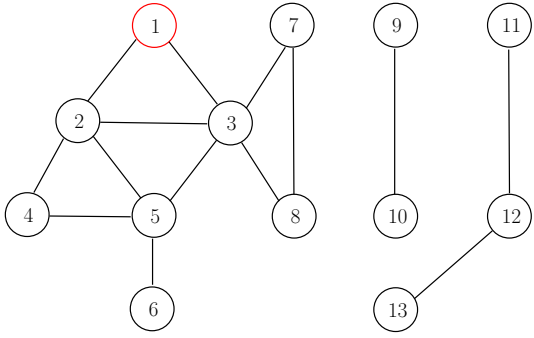
Αναζήτηση Πρώτα Κατά Πλάτος

Παράδειγμα BFS από κόμβο v_1



Αναζήτηση Πρώτα Κατά Πλάτος

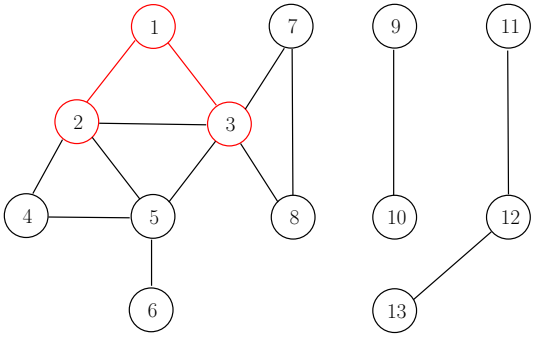
Παράδειγμα BFS από κόμβο v_1



1. κόμβος v_1 ως s

Αναζήτηση Πρώτα Κατά Πλάτος

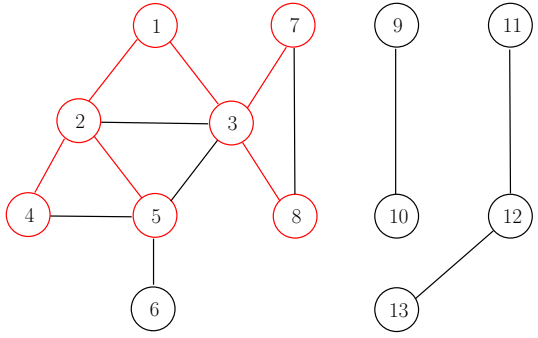
Παράδειγμα BFS από κόμβο v_1



1. κόμβος v_1 ως s
2. πρώτο επίπεδο αναζήτησης $\{v_2, v_3\}$

Αναζήτηση Πρώτα Κατά Πλάτος

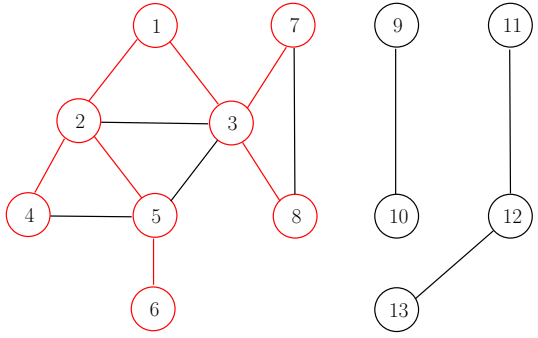
Παράδειγμα BFS από κόμβο v_1



1. κόμβος v_1 ως s
2. πρώτο επίπεδο αναζήτησης $\{v_2, v_3\}$
3. δεύτερο επίπεδο αναζήτησης $\{v_4, v_5, v_7, v_8\}$

Αναζήτηση Πρώτα Κατά Πλάτος

Παράδειγμα BFS από κόμβο v_1



1. κόμβος v_1 ως s
2. πρώτο επίπεδο αναζήτησης $\{v_2, v_3\}$
3. δεύτερο επίπεδο αναζήτησης $\{v_4, v_5, v_7, v_8\}$
4. τρίτο επίπεδο αναζήτησης $\{v_6\}$

Επίπεδο L_0. Ορίζουμε ως επίπεδο μηδέν το σύνολο $\{s\}$ που περιέχει μόνο τον κόμβο s.

Επίπεδο L_{j+1}. Αν έχουμε ορίσει τα επίπεδα $L_0, L_1, \ldots, L_j$, τότε το επίπεδο L_{j+1} αποτελείται από όλους τους κόμβους που

1. δεν ανήκουν σε προηγούμενο επίπεδο και
2. διαθέτουν ακμή με κόμβο του επιπέδου L_j.

Λόγω κατασκευής του αλγορίθμου καταλήγουμε στο εξής αποτέλεσμα:

Λήμμα

Για κάθε $j \geq 0$, το επίπεδο L_j που παράγεται από τον αλγόριθμο BFS αποτελείται από όλους τους κόμβους που έχουν ακριβώς *απόσταση* j από τον κόμβο s. Υπάρχει διαδρομή από τον κόμβο s προς τον κόμβο t αν και μόνον αν ο t εμφανίζεται σε κάποιο επίπεδο.

Παραγωγή Δέντρου. Ο αλγόριθμος BFS παράγει με πολύ φυσικό τρόπο ένα δέντρο T με ρίζα τον κόμβο s για το σύνολο των κόμβων που μπορούν να προσεγγιστούν από τον s.

- για κάθε κόμβο $v \neq s$ θεωρήστε την στιγμή όπου για πρώτη φορά "ανακαλύπτεται" ο v από τον αλγόριθμο BFS.
- συμβαίνει όταν εξετάζεται κάποιος κόμβος u του επιπέδου L_j και διαπιστώνεται πως υπάρχει ακμή (u, v) προς τον κόμβο v, ο οποίος δεν έχει φανεί μέχρι τότε.
- την στιγμή αυτή προσθέτουμε την ακμή (u, v) στο δέντρο T, δηλαδή ο u γίνεται γονέας του v.

Λήμμα

Έστω T ένα δέντρο αναζήτησης πρώτα κατά πλάτος, και έστω x και y κόμβοι του T που ανήκουν στα επίπεδα L_i και L_j αντίστοιχα, και (x, y) μια ακμή του G. Τότε τα i και j διαφέρουν το πολύ κατά 1.

Λήμμα

Έστω T ένα δέντρο αναζήτησης πρώτα κατά πλάτος, και έστω x και y κόμβοι του T που ανήκουν στα επίπεδα L_i και L_j αντίστοιχα, και (x, y) μια ακμή του G. Τότε τα i και j διαφέρουν το πολύ κατά 1.

Απόδειξη

□

Λήμμα

Έστω T ένα δέντρο αναζήτησης πρώτα κατά πλάτος, και έστω x και y κόμβοι του T που ανήκουν στα επίπεδα L_i και L_j αντίστοιχα, και (x, y) μια ακμή του G. Τότε τα i και j διαφέρουν το πολύ κατά 1.

Απόδειξη

Θα χρησιμοποιήσουμε εις άτοπον απαγωγή.

□

Λήμμα

Έστω T ένα δέντρο αναζήτησης πρώτα κατά πλάτος, και έστω x και y κόμβοι του T που ανήκουν στα επίπεδα L_i και L_j αντίστοιχα, και (x, y) μια ακμή του G. Τότε τα i και j διαφέρουν το πολύ κατά 1.

Απόδειξη

Θα χρησιμοποιήσουμε εις άτοπον απαγωγή.

Ας υποθέσουμε πως τα i και j διαφέρουν περισσότερο από 1, συγκεκριμένα ότι $i < j - 1$.

□

Λήμμα

Έστω T ένα δέντρο αναζήτησης πρώτα κατά πλάτος, και έστω x και y κόμβοι του T που ανήκουν στα επίπεδα L_i και L_j αντίστοιχα, και (x, y) μια ακμή του G. Τότε τα i και j διαφέρουν το πολύ κατά 1.

Απόδειξη

Θα χρησιμοποιήσουμε εις άτοπον απαγωγή.

Ας υποθέσουμε πως τα i και j διαφέρουν περισσότερο από 1, συγκεκριμένα ότι $i < j - 1$.

Θεωρήστε τώρα το σημείο του αλγορίθμου BFS που εξετάζονται οι ακμές του κόμβου x. Επειδή ο x ανήκει στο L_i, οι μόνοι κόμβοι που ανακαλύπτονται από το x ανήκουν στο επίπεδο L_{i+1} και σε προηγούμενα από αυτό.

□

Λήμμα

Έστω T ένα δέντρο αναζήτησης πρώτα κατά πλάτος, και έστω x και y κόμβοι του T που ανήκουν στα επίπεδα L_i και L_j αντίστοιχα, και (x, y) μια ακμή του G. Τότε τα i και j διαφέρουν το πολύ κατά 1.

Απόδειξη

Θα χρησιμοποιήσουμε εις άτοπον απαγωγή.

Ας υποθέσουμε πως τα i και j διαφέρουν περισσότερο από 1, συγκεκριμένα ότι $i < j - 1$.

Θεωρήστε τώρα το σημείο του αλγορίθμου BFS που εξετάζονται οι ακμές του κόμβου x. Επειδή ο x ανήκει στο L_i, οι μόνοι κόμβοι που ανακαλύπτονται από το x ανήκουν στο επίπεδο L_{i+1} και σε προηγούμενα από αυτό.

Αφού ο y είναι γειτονικός κόμβους του x, θα πρέπει να ανήκει στο επίπεδο L_{i+1} ή σε προηγούμενο και άρα $j \leq i + 1$ που είναι άτοπο. □

Συνεκτική Συνιστώσα

Connected Component

Ορισμός. Το σύνολο των κόμβων που ανακαλύπτονται από τον BFS είναι οι κόμβοι που μπορούν να προσεγγιστούν από τον κόμβο αφετηρίας s. Θα ονομάσουμε το σύνολο αυτό R **συνεκτική συνιστώσα** (connected component) του G που περιέχει τον s.

Αναζήτηση Πρώτα Κατά Βάθος

Depth-First Search, DFS

Τι προσέγγιση θα ακολουθούσατε αν το γράφημα ήταν πραγματικά ένας λαβύρινθος και περπατούσατε σε αυτόν;

- θα ξεκινούσατε από τον s, και θα δοκιμάζατε την πρώτη ακμή έξω από αυτόν, στον κόμβο v.
- θα συνεχίζατε έτσι μέχρι να φτάσετε σε κάποιο ”αδιέξοδο”, έναν κόμβο του οποίου έχετε εξερευνήσει όλους του γείτονες
- κατόπιν θα γυρίζατε προς τα πίσω μέχρι να φτάσετε σε κάποιον κόμβο με ανεξερεύνητο γείτονα, και θα συνεχίζατε από εκεί.

Αναζήτηση Πρώτα Κατά Βάθος

Depth-First Search, DFS

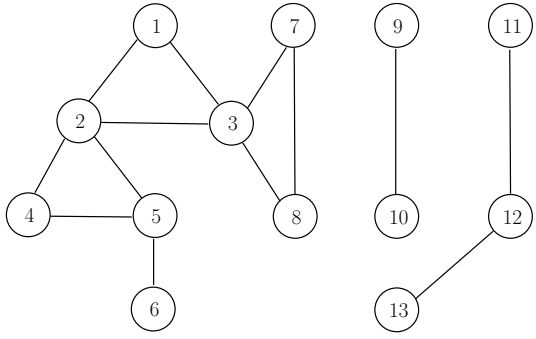
DFS(u)

Σημείωσε ότι το u "εξερευνήθηκε" και πρόσθεσε το u στην συνεκτική συνιστώσα R
for *κάθε ακμή* (u, v) **do**
  if *ο κόμβος* v *δεν έχει "εξερευνηθεί"* **then**
    κάλεσε αναδρομικά την DFS(v)
  end
end

Συνεκτικότητα $s - t$. Θέτουμε για όλους τους κόμβους πως δεν έχουν εξερευνηθεί και καλούμε την DFS(s).

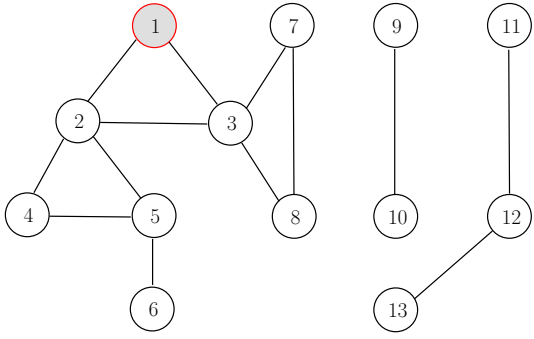
Αναζήτηση Πρώτα Κατά Βάθος

Παράδειγμα DFS από κόμβο v_1



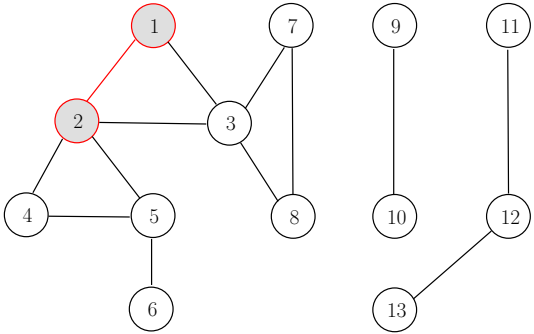
Αναζήτηση Πρώτα Κατά Βάθος

Παράδειγμα DFS από κόμβο v_1



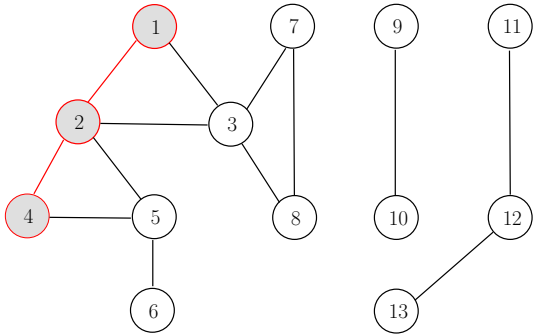
Αναζήτηση Πρώτα Κατά Βάθος

Παράδειγμα DFS από κόμβο v_1



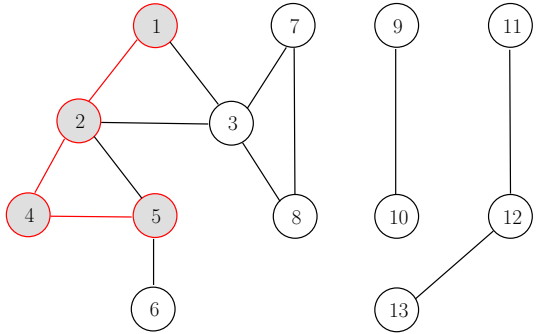
Αναζήτηση Πρώτα Κατά Βάθος

Παράδειγμα DFS από κόμβο v_1



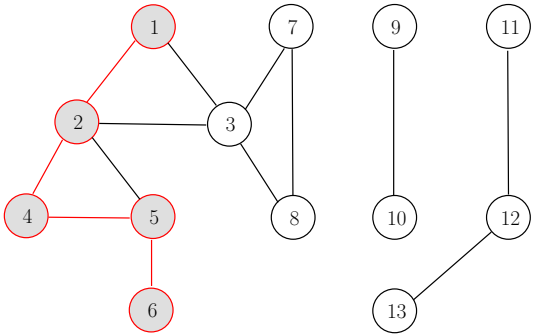
Αναζήτηση Πρώτα Κατά Βάθος

Παράδειγμα DFS από κόμβο v_1



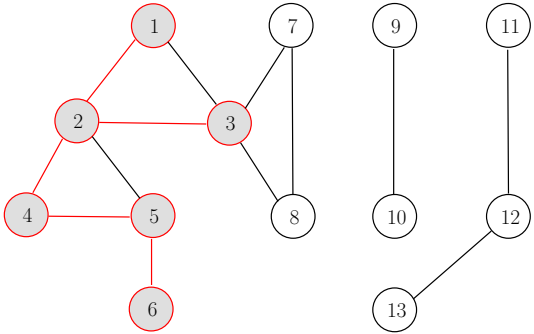
Αναζήτηση Πρώτα Κατά Βάθος

Παράδειγμα DFS από κόμβο v_1



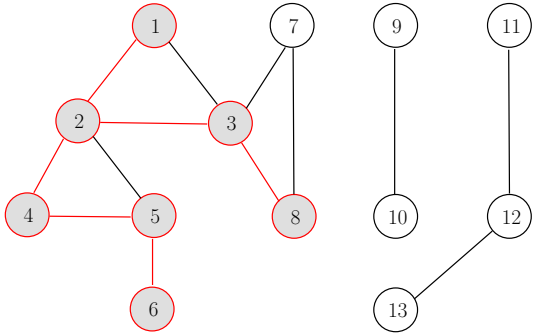
Αναζήτηση Πρώτα Κατά Βάθος

Παράδειγμα DFS από κόμβο v_1



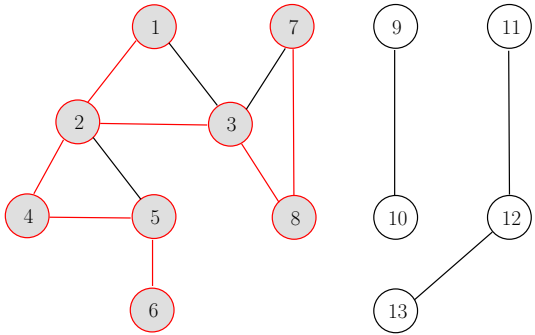
Αναζήτηση Πρώτα Κατά Βάθος

Παράδειγμα DFS από κόμβο v_1



Αναζήτηση Πρώτα Κατά Βάθος

Παράδειγμα DFS από κόμβο v_1



Αναπαράσταση Γραφημάτων

Ως τώρα έχουμε δει τα γραφήματα ως μαθηματικά αντικείμενα.

Αναπαράσταση. Πως όμως αναπαριστάμε γραφήματα στον υπολογιστή;

Πίνακες και Λίστες. Θα χρησιμοποιήσουμε πίνακες και λίστες για διάφορες αναπαραστάσεις.

Αναπαράσταση Γραφημάτων

Δύο βασικές λύσεις. Υπάρχουν δύο βασικές αναπαραστάσεις ενός γραφήματος $G(V, E)$ στον υπολογιστή:

1. μήτρα γειτνίασης (adjacency matrix)
2. λίστα γειτνίασης (adjacency list)

Γιατί όχι μια αναπαράσταση; Έστω ένα γράφημα $G(V, E)$ με $n = |V|$ και $m = |E|$. Ανάλογα με την σχέση των n και m, κάθε αναπαράσταση έχει άλλα πλεονεκτήματα και μειονεκτήματα.

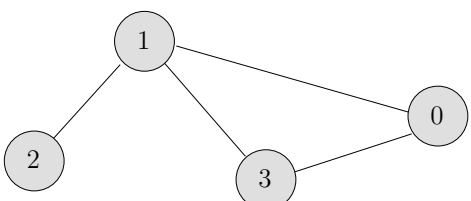
Αριθμός ακμών. Εαν δεν επιτρέπουμε πολλές ακμές μεταξύ ενός ζευγαριού κόμβων, ο αριθμός ακμών σε ένα γράφημα είναι το πολύ $\binom{n}{2} \leq n^2$. Επίσης πολλές φορές εργαζόμαστε με συνδεδεμένα γραφήματα όπου $m \geq n - 1$.

Αναπαράσταση Γραφημάτων

Μήτρα Γειτνίασης (adjacency matrix)

Έστω το γράφημα $G(V, E)$ όπου $V = \{0, 1, \ldots, n-1\}$.

- διδιάστατος πίνακα A με διαστάσεις $n \times n$
- η θέση του πίνακα i, j υποδηλώνει την ύπαρξη ή όχι της ακμής (i, j)
- εαν $e = (i, j) \in E$ τότε $A[i][j] = 1$ αλλιώς 0
- για μη-κατευθυνόμενα γραφήματα ο A είναι συμμετρικός



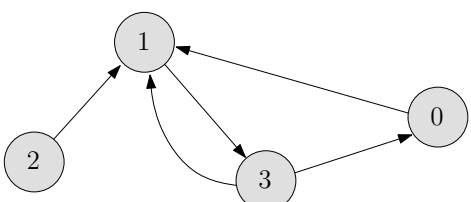
$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Αναπαράσταση Γραφημάτων

Μήτρα Γειτνίασης (adjacency matrix)

Έστω το γράφημα $G(V, E)$ όπου $V = \{0, 1, \ldots, n-1\}$.

- διδιάστατος πίνακα A με διαστάσεις $n \times n$
- η θέση του πίνακα i, j υποδηλώνει την ύπαρξη ή όχι της ακμής (i, j)
- εαν $e = (i, j) \in E$ τότε $A[i][j] = 1$ αλλιώς 0
- για μη-κατευθυνόμενα γραφήματα ο A είναι συμμετρικός



$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

Ιδιότητες

Ερώτημα ύπαρξης ακμής. Μπορούμε να ελέγξουμε την ύπαρξη μιας δεδομένης ακμής $e = (u, v)$ σε σταθερό χρόνο $\mathcal{O}(1)$.

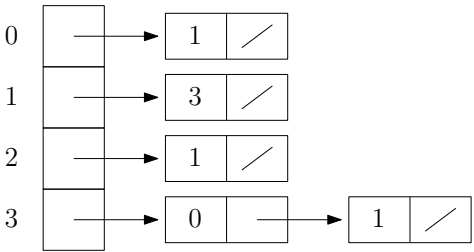
Εξέταση προσκείμενων ακμών. Για να εξετάσουμε διαδοχικά όλες τις ακμές που είναι προσκείμενες σε ένα κόμβο u, πρέπει να ελέγξουμε για κάθε κομβου στο σύνολο $V \setminus \{u\}$ την τιμή του πίνακα A. Χρειαζόμαστε δηλαδή χρόνο $\mathcal{O}(n)$.

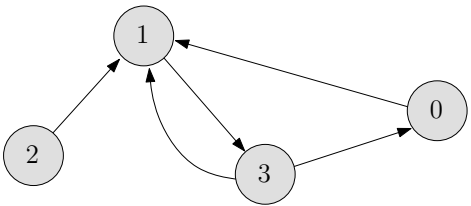
Η λειτουργία αυτή είναι πολύ συνήθης στους αλγορίθμους γραφημάτων.

Χώρος. Η αναπαράσταση αυτή χρειάζεται χώρο $\mathcal{O}(n^2)$. Ακόμη και όταν ο αριθμός των ακμών m είναι μικρότερος από n^2.

Έστω το γράφημα $G(V, E)$ όπου $V = \{0, 1, \ldots, n-1\}$.

- μονοδιάστατος πίνακα A με μια θέση για κάθε κόμβο
- κάθε θέση του πίνακα A είναι μια λίστα ακμών που εξέρχονται από τον αντίστοιχο κόμβο





Ιδιότητες

Ορισμός. Ορίζουμε τον βαθμό ενός κόμβου u συμβολίζοντας τον ως d_u το πλήθος των εξερχόμενων ακμών του κόμβου u.

Ερώτημα ύπαρξης ακμής. Ο έλεγχος ύπαρξης της ακμής $e = (u, v)$ δεν είναι πλέον σταθερός, χρειάζεται διάσχιση της λίστας εξερχόμενων ακμών του κόμβου u. Χρειαζόμαστε χρόνο $\mathcal{O}(d_u)$.

Ιδιότητες

Ορισμός. Ορίζουμε τον βαθμό ενός κόμβου u συμβολίζοντας τον ως d_u το πλήθος των εξερχόμενων ακμών του κόμβου u.

Ερώτημα ύπαρξης ακμής. Ο έλεγχος ύπαρξης της ακμής $e = (u, v)$ δεν είναι πλέον σταθερός, χρειάζεται διάσχιση της λίστας εξερχόμενων ακμών του κόμβου u. Χρειαζόμαστε χρόνο $\mathcal{O}(d_u)$.

Εξέταση προσκείμενων ακμών. Μπορούμε αποδοτικά διασχίζοντας την λίστα εξερχόμενων ακμών του κόμβου u σε χρόνο $\mathcal{O}(d_u)$.

Ιδιότητες

Ορισμός. Ορίζουμε τον βαθμό ενός κόμβου u συμβολίζοντας τον ως d_u το πλήθος των εξερχόμενων ακμών του κόμβου u.

Ερώτημα ύπαρξης ακμής. Ο έλεγχος ύπαρξης της ακμής $e = (u, v)$ δεν είναι πλέον σταθερός, χρειάζεται διάσχιση της λίστας εξερχόμενων ακμών του κόμβου u. Χρειαζόμαστε χρόνο $\mathcal{O}(d_u)$.

Εξέταση προσκείμενων ακμών. Μπορούμε αποδοτικά διασχίζοντας την λίστα εξερχόμενων ακμών του κόμβου u σε χρόνο $\mathcal{O}(d_u)$.

Χώρος. Η αναπαράσταση αυτή χρειάζεται χώρο $\mathcal{O}(n + m)$.

- $\mathcal{O}(n)$ για τον πίνακα
- $\mathcal{O}(m)$ για τις λίστες, αφού κάθε κόμβος έχει d_u εξερχόμενες ακμές και $\sum_{v \in V} d_v = 2m = \mathcal{O}(m)$.

Υλοποίηση Αναζήτησης Πρώτα Κατά Πλάτος

Με την χρήση ουράς FIFO

Δομές. Μια ουρά Q, ένας πίνακας discovered με θέση για κάθε κόμβο και ένα δέντρο T.

Algorithm 2: BFS(u)

```
T = ∅;
discovered[s] = true;
for v ∈ V \ {s} do
    discovered[u] = false;
end
Q = {s};
while Q έχει στοιχεία do
    αφαίρεσε το στοιχείο u από την ουρά Q;
    for κάθε ακμή (u, v) do
        if discovered[v] = false then
            discovered[v] = true;
            πρόσθεσε την ακμή (u, v) στο BFS δέντρο T;
            πρόσθεσε τον κόμβο v στην ουρά Q;
        end
    end
end
```

Αρχικοποίηση. Η αρχικοποίηση του πίνακα discovered χρειάζεται $\mathcal{O}(n)$ χρόνο. Η ουρά και το δέντρο αρχικοποιούνται σε σταθερό χρόνο.

Χρόνος ανά κόμβο. Για κάθε κόμβο $v \in V$ κάνουμε κάτι που πέρνει σταθερό χρόνο για κάθε εξερχόμενη ακμή και άρα χρειαζόμαστε χρόνο όσο και ο βαθμός του, δηλαδή $\mathcal{O}(d_v)$.

Ξέρουμε όμως πως $\sum_{v \in V} d_v = 2m = \mathcal{O}(m)$.

Συνολικός χρόνος $\mathcal{O}(n + m)$.

Υλοποίηση Αναζήτησης Πρώτα Κατά Βάθος

Η υλοποίηση της αναζήτησης πρώτα κατά βάθος (DFS) μπορεί να γίνει είτε με στοίβα είτε χρησιμοποιώντας αναδρομή.

Υλοποίηση Αναδρομής. Θυμηθείτε πως για να υποστηρίξουν αναδρομή οι γλώσσες προγραμματισμού χρησιμοποιούν μία στοίβα (call stack).

Υλοποίηση Αναζήτησης Πρώτα Κατά Βάθος

Με την χρήση στοίβας

Δομές. Μια στοίβα S και ένας πίνακας explored με θέση για κάθε κόμβο.

Algorithm 3: DFS(s)

```
for v ∈ V do
    explored[u] = false;
end
όρισε την S ως στοίβα με ένα στοιχείο s;
while S έχει στοιχεία do
    αφαίρεσε το στοιχείο u από την στοίβα S;
    if explored[u] = false then
        explored[u] = true;
        for κάθε ακμή (u, v) do
            πρόσθεσε τον κόμβο v στην στοίβα S;
        end
    end
end
```

Αρχικοποίηση. Η αρχικοποίηση του πίνακα explored χρειάζεται $\mathcal{O}(n)$ χρόνο. Η αρχικοποίηση της στοίβας χρειάζεται $\mathcal{O}(1)$ χρόνο.

Λειτουργίες Στοίβας. Οι λειτουργίες της στοίβας (PUSH, POP) χρειάζονται σταθερό χρόνο. Για να βρούμε λοιπόν τον χρόνο πρέπει να βρούμε πόσες τέτοιες λειτουργίες κάνουμε.

Προσθήκη στην Στοίβα. Αρκεί να μετρήσουμε πόσες φορές προσθέτουμε ένα στοιχείο στην στοίβα καθώς κάθε κόμβος πρέπει να προστεθεί μια φορά για κάθε φορά που μπορεί να διαγραφεί από την στοίβα.

Χρόνος ανά κόμβο. Κάθε κόμβος $v \in V$ προστίθεται κάθε φορά που γίνεται εξερεύνηση ενός από τους d_v γειτονικούς κόμβους του.

Ο συνολικός αριθμός κόμβων που προστίθενται στη στοίβα είναι το πολύ

$$\sum_{v \in V} d_v = 2m = \mathcal{O}(m).$$

Συνολικός χρόνος $\mathcal{O}(n + m)$.

Έχοντας ορίσει την συνεκτική συνιστώσα ενός συγκεκριμένου κόμβου s, μπορούμε πλέον να μιλήσουμε για το σύνολο των συνεκτικών συνιστωσών (connected components).

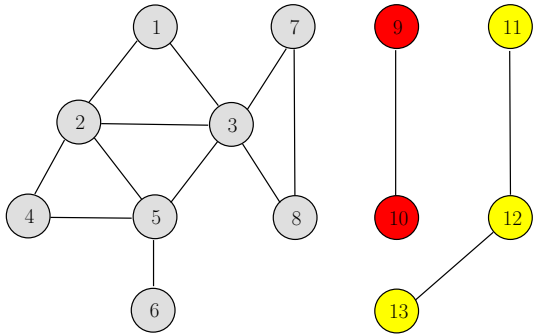
Λήμμα

Για οποιουσδήποτε δύο κόμβους s και t σε ένα γράφημα, οι συνεκτικές συνιστώσες τους είναι είτε ταυτόσημες ή ξένες.

Μπορούμε να βρούμε σε γραμμικό χρόνο $\mathcal{O}(n+m)$ όλες τις συνεκτικές συνιστώσες εκτελώντας πολλές φορές έναν αλγόριθμο διάσχισης από διαφορετικούς κόμβους.

Σύνολο Συνεκτικών Συνιστωσών

Connected Components



Αρκούν 3 διασχίσεις π.χ DFS(v_1), DFS(v_9), DFS(v_{11})

Κατευθυνόμενα Γραφήματα
Directed Graphs

Φυσικά ανάλογα. Οι αλγόριθμοι που είδαμε ως τώρα έχουν φυσικά ανάλογα σε κατευθυνόμενα γραφήματα.

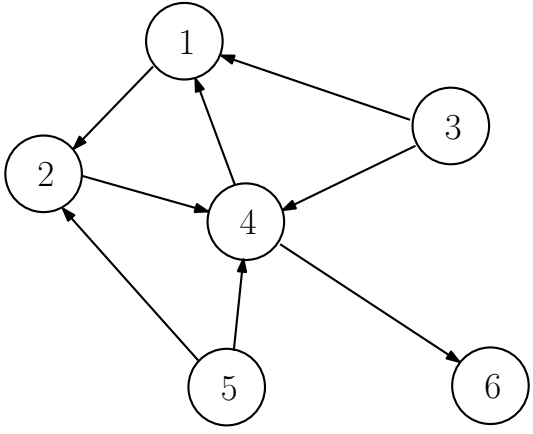
Αναπαράσταση. Πολλές φορές αναπαριστάμε τα κατευθυνόμενα γραφήματα με μια ενισχυμένη μορφή της λίστας γειτνίασης. Για κάθε κόμβο κρατάμε δύο λίστες, τις εξερχόμενες ακμές και τις εισερχόμενες ακμές.

Διάσχιση. Οι αλγόριθμοι BFS και DFS υλοποιούνται με τον ίδιο τρόπο, αρκεί να προσέχουμε να διασχίζουμε ακμές μόνο προς την σωστή κατεύθυνση.

Συνεκτικότητα σε Κατευθυνόμενα Γραφήματα

Directed Graphs

Εκτελώντας μια διάσχιση από ένα κόμβο s σε ένα κατευθυνόμενο γράφημα $G(V, E)$ βρίσκουμε όλους τους κόμβους προς τους οποίους υπάρχει διαδρομή από τον s.

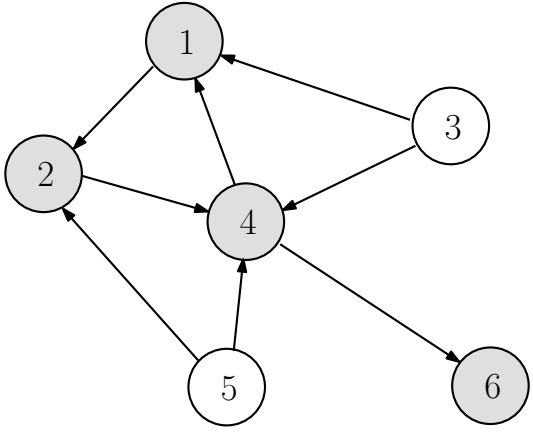


π.χ εκτελώντας DFS(v_1) βρίσκουμε τους κόμβους v_2, v_4, και v_6.

Συνεκτικότητα σε Κατευθυνόμενα Γραφήματα

Directed Graphs

Εκτελώντας μια διάσχιση από ένα κόμβο s σε ένα κατευθυνόμενο γράφημα $G(V, E)$ βρίσκουμε όλους τους κόμβους προς τους οποίους υπάρχει διαδρομή από τον s.

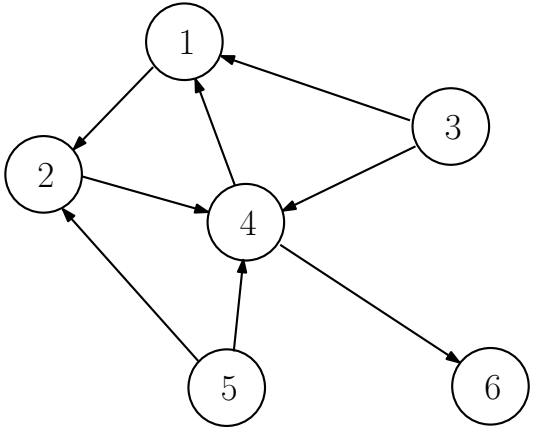


π.χ εκτελώντας DFS(v_1) βρίσκουμε τους κόμβους v_2, v_4, και v_6.

Συνεκτικότητα σε Κατευθυνόμενα Γραφήματα

Directed Graphs

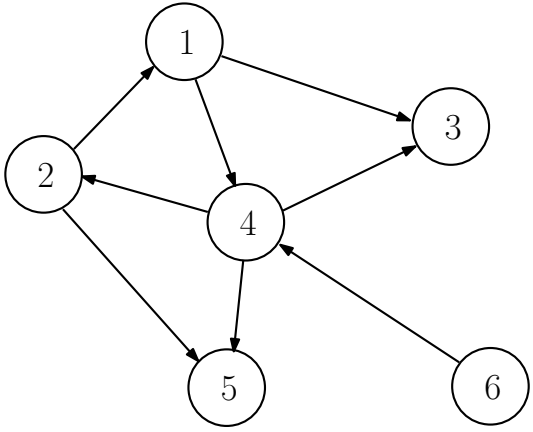
Για να βρούμε τους κόμβους που έχουν διαδρομή προς ένα κόμβο s σε ένα κατευθυνόμενο γράφημα $G(V, E)$, μπορούμε να αντιστρέψουμε όλες τις ακμές ώστε να λάβουμε το γράφημα G^{rev}.



Συνεκτικότητα σε Κατευθυνόμενα Γραφήματα

Directed Graphs

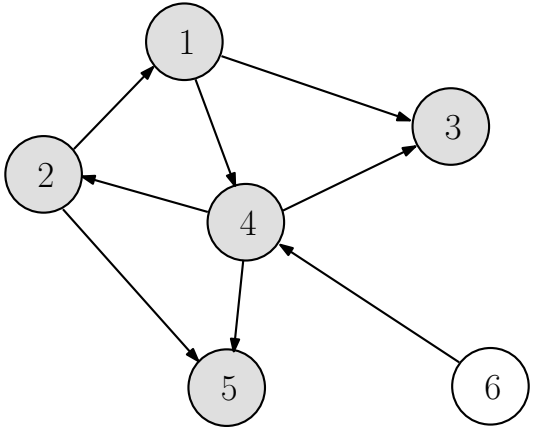
Για να βρούμε τους κόμβους που έχουν διαδρομή προς ένα κόμβο s σε ένα κατευθυνόμενο γράφημα $G(V, E)$, μπορούμε να αντιστρέψουμε όλες τις ακμές ώστε να λάβουμε το γράφημα G^{rev}.



Συνεκτικότητα σε Κατευθυνόμενα Γραφήματα

Directed Graphs

Για να βρούμε τους κόμβους που έχουν διαδρομή προς ένα κόμβο s σε ένα κατευθυνόμενο γράφημα $G(V, E)$, μπορούμε να αντιστρέψουμε όλες τις ακμές ώστε να λάβουμε το γράφημα G^{rev}.



π.χ εκτελώντας DFS(v_1) βρίσκουμε τους κόμβους v_2, v_3, v_4, και v_5.

Ισχυρή Συνεκτικότητα

Θυμηθείτε πως ένα κατευθυνόμενο γράφημα είναι **ισχυρά συνδεδεμένο** (strongly connected) αν, για κάθε ζεύγος κόμβων u και v, υπάρχει διαδρομή από το u στο v και διαδρομή από το v στο u.

Αμοιβαία Προσπελάσιμοι. Θα λέμε δύο κόμβους u και v, αμοιβαία προσπελάσιμους (mutually reachable) εαν $u \rightsquigarrow v$ και $v \rightsquigarrow u$.

Λήμμα

Εαν οι κόμβοι u και v είναι αμοιβαία προσπελάσιμοι και οι κόμβοι v και w είναι αμοιβαία προσπελάσιμοι, τότε οι κόμβοι u και w είναι αμοιβαία προσπελάσιμοι.

Απόδειξη

Απλά φτιάξτε τις διαδρομές... □

Ισχυρή Συνεκτικότητα

Πως μπορούμε να βρούμε εάν ένα κατευθυνόμενο γράφημα είναι ισχυρά συνδεδεμένο;

Γραμμικός Αλγόριθμος.

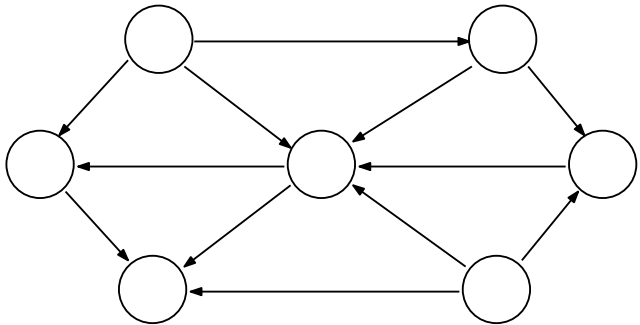
1. διάλεξε ένα τυχαίο κόμβο $s \in V$
2. τρέξε BFS από τον s στο γράφημα G και στο G^{rev}
3. εάν μια από τις δύο αναζητήσεις αποτύχει να προσπελάσει όλους τους κόμβους, τότε προφανώς το G δεν είναι ισχυρά συνδεδεμένο
4. αλλιώς είναι, αφού για οποιοδήποτε ζευγάρι κόμβων u και v ισχύει
   - $u \rightsquigarrow s$ και $s \rightsquigarrow v$ και άρα $u \rightsquigarrow v$
   - $v \rightsquigarrow s$ και $s \rightsquigarrow u$ και άρα $v \rightsquigarrow u$

Η αντιστροφή του γραφήματος γίνεται σε γραμμικό χρόνο. Το ίδιο και οι δύο BFS αναζητήσεις.

Κατευθυνόμενο Ακυκλικό Γραφήμα

Directed Acyclic Graph (DAG)

Όταν ένα κατευθυνόμενο γράφημα δεν έχει κύκλους, ονομάζεται *κατευθυνόμενο ακυκλικό γράφημα*.



Κατευθυνόμενο Ακυκλικό Γραφήμα

Directed Acyclic Graph (DAG)

Χρησιμότητα. Τα κατευθυνόμενα ακυκλικά γραφήματα είναι ιδανικά για να εκφράσουν *σχέσεις προτεραιότητας* και *εξαρτήσεις*.

Παράδειγμα.

- ένα σύνολο από εργασίες με ετικέτες $\{1, 2, \ldots, n\}$ που πρέπει να εκτελεστούν
- εξαρτήσεις εργασιών όπου για ορισμένα ζευγάρια i και j η εργασία i πρέπει να πραγματοποιηθεί πριν από την εργασία j

Κατευθυνόμενο Ακυκλικό Γραφήμα

Directed Acyclic Graph (DAG)

Χρησιμότητα. Τα κατευθυνόμενα ακυκλικά γραφήματα είναι ιδανικά για να εκφράσουν *σχέσεις προτεραιότητας* και *εξαρτήσεις*.

Φτιάχνουμε ένα γράφημα

- όπου υπάρχει ένας κόμβος για κάθε εργασία
- υπάρχει μια κατευθυνόμενη ακμή (i, j) κάθε φορά που η εργασία i πρέπει να εκτελεστεί πριν από την εργασία j

Για να έχει κάποιο νόημα η σχέση προτεραιότητας, το γράφημα που προκύπτει πρέπει να είναι ένα DAG.

Σε αυτή την περίπτωση μας ενδιαφέρει μια σειρά εκτέλεσης των εργασιών.

Τοπολογική Διάταξη

Directed Acyclic Graph (DAG)

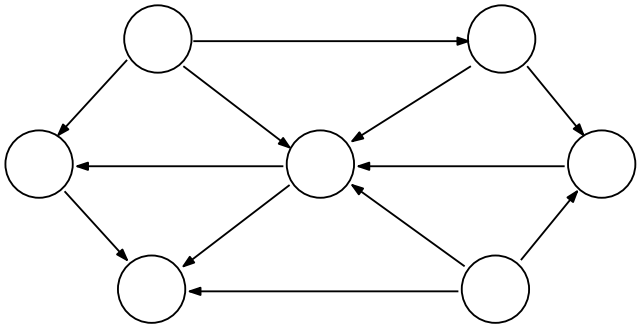
Ορισμός. Για ένα κατευθυνόμενο γράφημα G, λέμε ότι η **τοπολογική διάταξη** (topological ordering) του G είναι μια ταξινόμηση των κόμβων σε μορφή $v_1, v_2, \ldots, v_n$ έτσι ώστε για κάθε ακμή (v_i, v_j) να ισχύει $i < j$.

Με άλλα λόγια όλες οι ακμές δείχνουν ”προς τα εμπρός” στην διάταξη.

Τοπολογική Διάταξη

Directed Acyclic Graph (DAG)

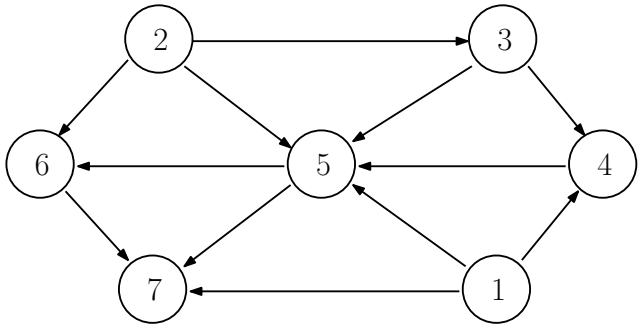
Ορισμός. Για ένα κατευθυνόμενο γράφημα G, λέμε ότι η **τοπολογική διάταξη** (topological ordering) του G είναι μια ταξινόμηση των κόμβων σε μορφή $v_1, v_2, \ldots, v_n$ έτσι ώστε για κάθε ακμή (v_i, v_j) να ισχύει $i < j$.



Τοπολογική Διάταξη

Directed Acyclic Graph (DAG)

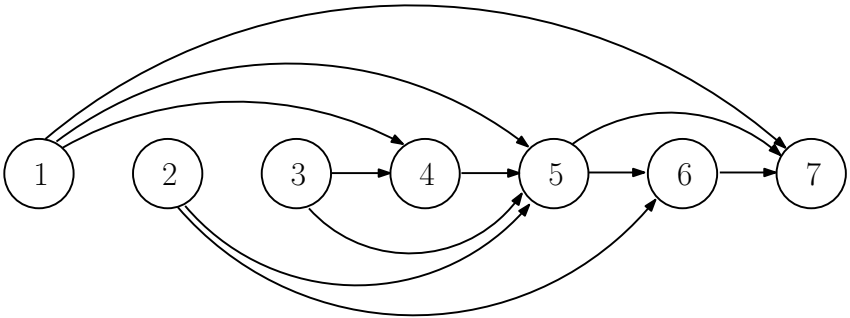
Ορισμός. Για ένα κατευθυνόμενο γράφημα G, λέμε ότι η **τοπολογική διάταξη** (topological ordering) του G είναι μια ταξινόμηση των κόμβων σε μορφή $v_1, v_2, \ldots, v_n$ έτσι ώστε για κάθε ακμή (v_i, v_j) να ισχύει $i < j$.



Τοπολογική Διάταξη

Directed Acyclic Graph (DAG)

Ορισμός. Για ένα κατευθυνόμενο γράφημα G, λέμε ότι η **τοπολογική διάταξη** (topological ordering) του G είναι μια ταξινόμηση των κόμβων σε μορφή $v_1, v_2, \ldots, v_n$ έτσι ώστε για κάθε ακμή (v_i, v_j) να ισχύει $i < j$.



Λήμμα

Εάν ένα γράφημα G διαθέτει τοπολογική διάταξη, τότε το G είναι γράφημα DAG.

Λήμμα

Εάν ένα γράφημα G διαθέτει τοπολογική διάταξη, τότε το G είναι γράφημα DAG.

Απόδειξη

□

Λήμμα

Εάν ένα γράφημα G διαθέτει τοπολογική διάταξη, τότε το G είναι γράφημα DAG.

Απόδειξη

Θα χρησιμοποιήσουμε εις άτοπον απαγωγή.

□

Λήμμα

Εάν ένα γράφημα G διαθέτει τοπολογική διάταξη, τότε το G είναι γράφημα DAG.

Απόδειξη

Θα χρησιμοποιήσουμε εις άτοπον απαγωγή.

Έστω πως το G έχει μια τοπολογική διάταξη $v_1, v_2, \ldots, v_n$ και ένα κύκλο C.

□

Λήμμα

Εάν ένα γράφημα G διαθέτει τοπολογική διάταξη, τότε το G είναι γράφημα DAG.

Απόδειξη

Θα χρησιμοποιήσουμε εις άτοπον απαγωγή.

Έστω πως το G έχει μια τοπολογική διάταξη $v_1, v_2, \ldots, v_n$ και ένα κύκλο C.

Έστω επίσης v_i ο κόμβος με τον μικρότερο δείκτη στον κύκλο C και v_j ο κόμβος ακριβώς πριν από τον v_i στον κύκλο.

□

Λήμμα

Εάν ένα γράφημα G διαθέτει τοπολογική διάταξη, τότε το G είναι γράφημα DAG.

Απόδειξη

Θα χρησιμοποιήσουμε εις άτοπον απαγωγή.

Έστω πως το G έχει μια τοπολογική διάταξη $v_1, v_2, \ldots, v_n$ και ένα κύκλο C.

Έστω επίσης v_i ο κόμβος με τον μικρότερο δείκτη στον κύκλο C και v_j ο κόμβος ακριβώς πριν από τον v_i στον κύκλο.

Λόγω της υπόθεσης υπάρχει η ακμή $(v_j, v_i) \in C$ και άρα λόγω της τοπολογικής διάταξης $j < i$. Όμως το i είναι ο κόμβος με τον μικρότερο δείκτη, άτοπο. □

Υπολογισμός Τοπολογικής Διάταξη

Αρχικός Κόμβος. Μπορούμε να χαρακτηρίσουμε κάπως τον πρώτο κόμβο μιας τοπολογικής διάταξης;

Υπολογισμός Τοπολογικής Διάταξη

Αρχικός Κόμβος. Μπορούμε να χαρακτηρίσουμε κάπως τον πρώτο κόμβο μιας τοπολογικής διάταξης;

Λήμμα

Σε κάθε DAG G υπάρχει κόμβος v χωρίς εισερχόμενες ακμές.

Υπολογισμός Τοπολογικής Διάταξη

Αρχικός Κόμβος. Μπορούμε να χαρακτηρίσουμε κάπως τον πρώτο κόμβο μιας τοπολογικής διάταξης;

Λήμμα

Σε κάθε DAG G υπάρχει κόμβος v χωρίς εισερχόμενες ακμές.

Απόδειξη

Έστω πως κάθε κόμβος έχει τουλάχιστον μια εισερχόμενη ακμή.

Διάλεξε ένα κόμβο στην τύχη και άρχισε να πηγαίνεις ανάποδα ακολουθώντας μία εισερχόμενη ακμή. Επειδή όλοι οι κόμβοι έχουν τέτοια ακμή, μπορούμε να κάνουμε αυτή την διαδικασία επ' αόριστον.

Επειδή οι κόμβοι είναι n, μετά από $n+1$ βήματα θα έχουμε επισκεπτεί κάποιον κόμβο 2 φορές και άρα θα έχουμε βρει ένα κύκλο. □

Τοπολογική Διάταξη

Directed Acyclic Graph (DAG)

Λήμμα

Εάν ένα γράφημα G είναι DAG τότε έχει τοπολογική διάταξη.

Τοπολογική Διάταξη

Directed Acyclic Graph (DAG)

Λήμμα

Εάν ένα γράφημα G είναι DAG τότε έχει τοπολογική διάταξη.

Απόδειξη

□

Λήμμα

Εάν ένα γράφημα G είναι DAG τότε έχει τοπολογική διάταξη.

Απόδειξη

Θα το αποδείξουμε με επαγωγή στον αριθμό των κόμβων.

□

Λήμμα

Εάν ένα γράφημα G είναι DAG τότε έχει τοπολογική διάταξη.

Απόδειξη

Θα το αποδείξουμε με επαγωγή στον αριθμό των κόμβων.

Προφανώς ισχύει για ένα γράφημα με 1 ή 2 κόμβους. Έστω λοιπόν πως ισχύει για ένα γράφημα με n κόμβους.

□

Λήμμα

Εάν ένα γράφημα G είναι DAG τότε έχει τοπολογική διάταξη.

Απόδειξη

Θα το αποδείξουμε με επαγωγή στον αριθμό των κόμβων.

Προφανώς ισχύει για ένα γράφημα με 1 ή 2 κόμβους. Έστω λοιπόν πως ισχύει για ένα γράφημα με n κόμβους.

Έστω λοιπόν ένα DAG με $n+1$ κόμβους. Ξέρουμε πως υπάρχει κόμβος v χωρίς εισερχόμενες ακμές. Τοποθετούμε τον v πρώτο στην τοπολογική διάταξη.

□

Λήμμα

Εάν ένα γράφημα G είναι DAG τότε έχει τοπολογική διάταξη.

Απόδειξη

Θα το αποδείξουμε με επαγωγή στον αριθμό των κόμβων.

Προφανώς ισχύει για ένα γράφημα με 1 ή 2 κόμβους. Έστω λοιπόν πως ισχύει για ένα γράφημα με n κόμβους.

Έστω λοιπόν ένα DAG με $n+1$ κόμβους. Ξέρουμε πως υπάρχει κόμβος v χωρίς εισερχόμενες ακμές. Τοποθετούμε τον v πρώτο στην τοπολογική διάταξη.

Το γράφημα $G - \{v\}$ είναι DAG αφού η διαγραφή κόμβου δεν μπορεί να δημιουργήσει κύκλους που δεν υπάρχουν ήδη. Λόγω επαγωγικής υπόθεσης έχει τοπολογική διάταξη. Βάζουμε μετά τον v τους υπόλοιπους κόμβους με την σειρά αυτής της διάταξης.

□

Λήμμα

Εάν ένα γράφημα G είναι DAG τότε έχει τοπολογική διάταξη.

Απόδειξη

Θα το αποδείξουμε με επαγωγή στον αριθμό των κόμβων.

Προφανώς ισχύει για ένα γράφημα με 1 ή 2 κόμβους. Έστω λοιπόν πως ισχύει για ένα γράφημα με n κόμβους.

Έστω λοιπόν ένα DAG με $n+1$ κόμβους. Ξέρουμε πως υπάρχει κόμβος v χωρίς εισερχόμενες ακμές. Τοποθετούμε τον v πρώτο στην τοπολογική διάταξη.

Το γράφημα $G - \{v\}$ είναι DAG αφού η διαγραφή κόμβου δεν μπορεί να δημιουργήσει κύκλους που δεν υπάρχουν ήδη. Λόγω επαγωγικής υπόθεσης έχει τοπολογική διάταξη. Βάζουμε μετά τον v τους υπόλοιπους κόμβους με την σειρά αυτής της διάταξης.

Το αποτέλεσμα είναι τοπολογική διάταξη για τον G. □

Η απόδειξη είναι ουσιαστικά και ένας αλγόριθμος κατασκευής.

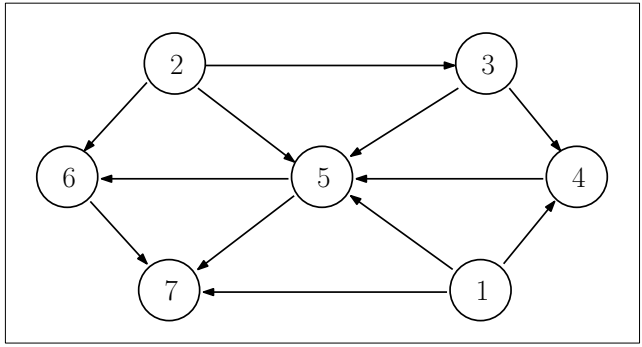
Algorithm 4: Υπολογισμός Τοπολογικής Διάταξης G

Βρες ένα κόμβο v χωρίς εισερχόμενες ακμές και τοποθέτησε τον στην πρώτη θέση.;
Διέγραψε τον v από το G.;
Υπολόγισε αναδρομικά μια τοπολογική διάταξη του $G - \{v\}$ και πρόσθεσε αυτή την διάταξη μετά τον κόμβο v.;

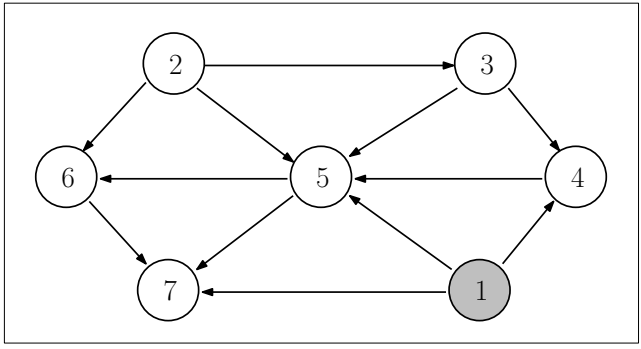
Χρόνος. Ο παραπάνω αλγόριθμος τρέχει σε χρόνο $\mathcal{O}(n^2)$ αφού

- ψάχνει να βρει ένα κόμβο χωρίς εισερχόμενες ακμές σε χρόνο $\mathcal{O}(n)$
- κάνει αυτή τη διαδικασία για όλους τους κόμβος, το πολύ $\mathcal{O}(n)$ φορές

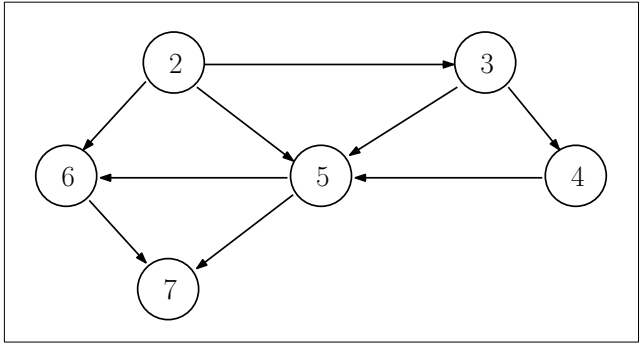
Αλγόριθμος
Τοπολογική Διάταξη



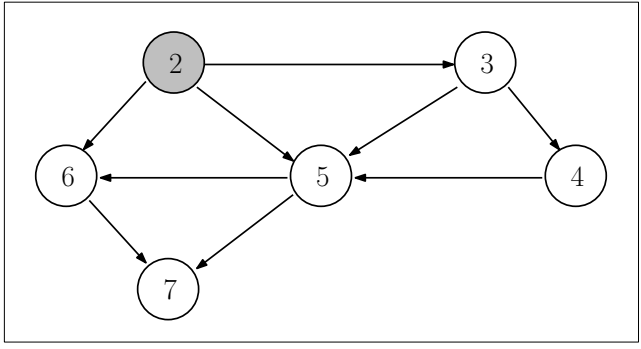
Αλγόριθμος
Τοπολογική Διάταξη



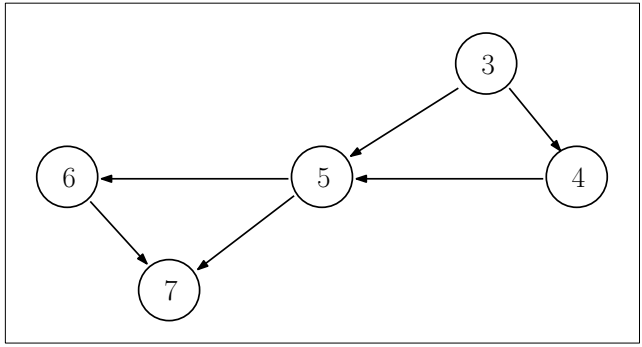
Αλγόριθμος
Τοπολογική Διάταξη

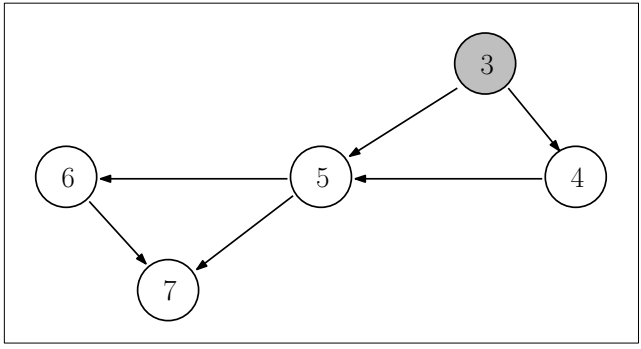


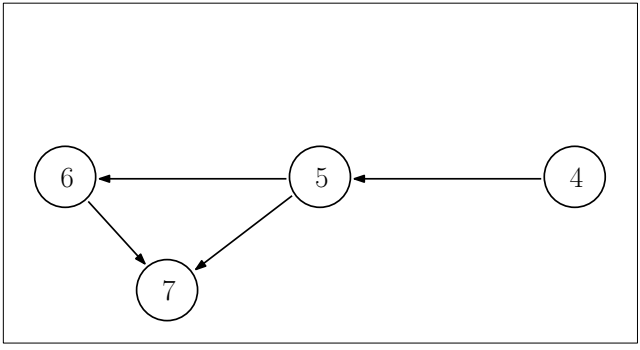
Αλγόριθμος
Τοπολογική Διάταξη

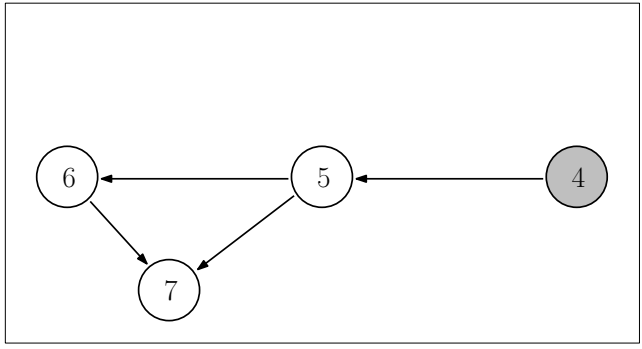


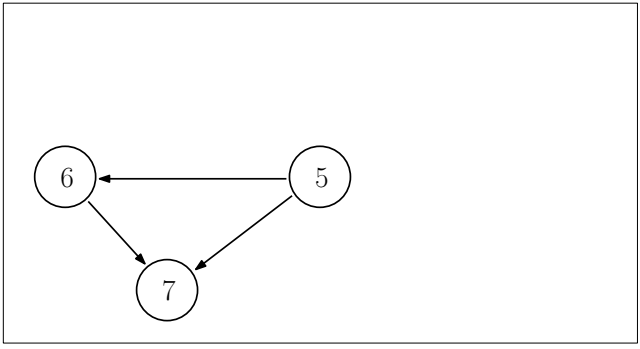
Αλγόριθμος
Τοπολογική Διάταξη

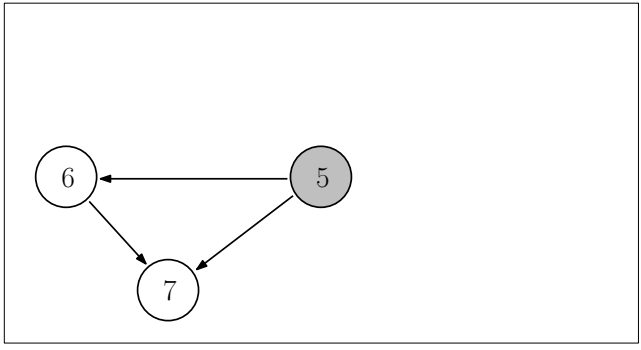


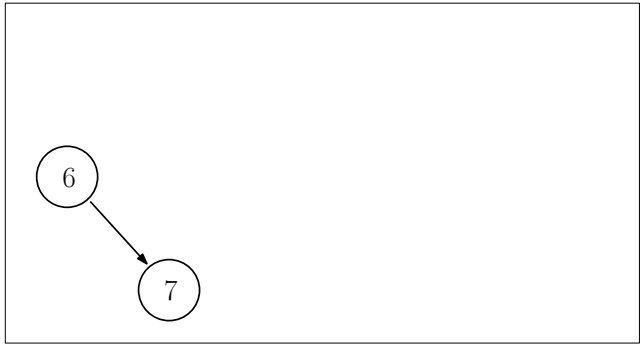

3
6
5
4
7

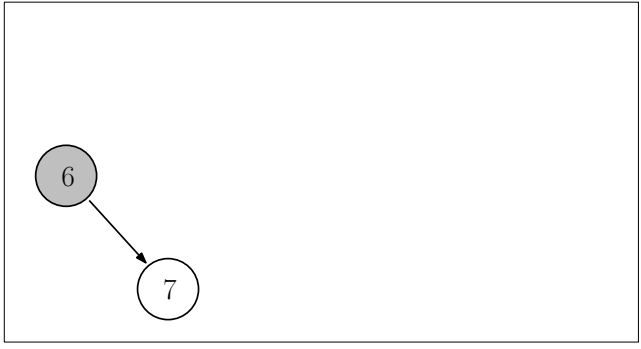

6
5
4
7


6
5
4
7


6
5
7

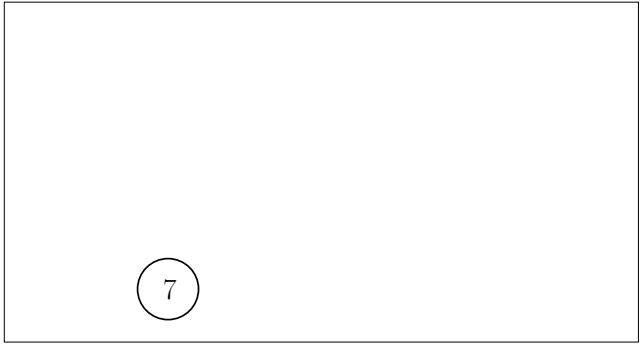

6
5
7


6
7


6
7

Αλγόριθμος

Τοπολογική Διάταξη



Algorithm 5: Υπολογισμός Τοπολογικής Διάταξης G

αρχικοποίησε ένα πίνακα d που περιέχει το αριθμό των εισερχόμενων ακμών κάθε κόμβου.;
αρχικοποίηση μιας λίστας S με τους κόμβους $v \in V$ που έχουν $d[v] = 0$.;
έστω μετρητής $i = 1$;
while *η λίστα S δεν είναι άδεια* **do**
    αφαίρεσε τον πρώτο κόμβο $u \in S$ από την λίστα S.;
    δώσε στον u το νούμερο i και αύξησε το i κατά ένα.;
    for *κάθε εξερχόμενη ακμή* (u, v) **do**
        μείωσε κατά ένα το $d[v]$;
        if $d[v] = 0$ **then**
            πρόσθεσε τον v στην λίστα S;
        end
    end
    διέγραψε τον u.
end

Χρόνος. Ο παραπάνω αλγόριθμος τρέχει σε χρόνο $\mathcal{O}(m + n)$ αφού κάνει σταθερή δουλειά ανά ακμή. Η αρχικοποίηση θέλει και αυτή γραμμικό χρόνο.