Κεφάλαιο 3

Αναπαράσταση και Διαπέραση Γραφημάτων

Περιεχόμενα

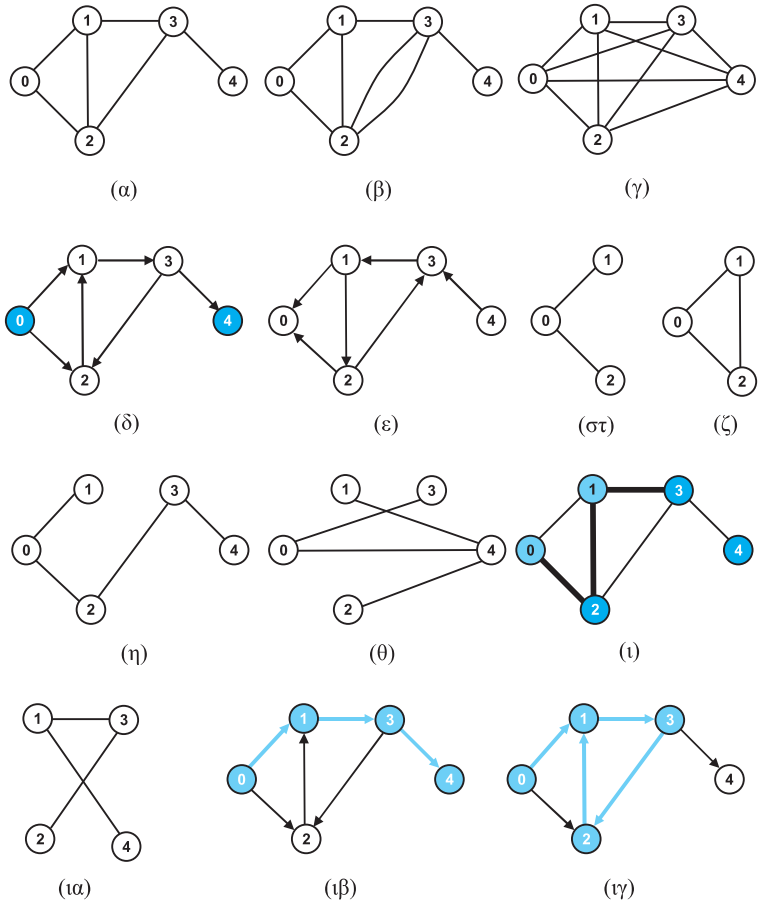
Τα γραφηματα αποτελούν αναπόσπαστο κομμάτι της Επιστήμης των Υπολογιστών, καθώς βρίσκουν πολυάριθμες εφαρμογές σε διαφορετικούς κλάδους της. Πράγματι, είναι δύσκολο να φανταστεί κανείς εφαρμογή που να μην τα χρησιμοποιεί! Στην συνέχεια θα ασχοληθούμε με την εσωτερική αναπαράσταση των γραφημάτων, τις μεθόδους διαπεράσεώς τους και τους βασικούς αλγορίθμους εξαγωγής των ποιοτικών χαρακτηριστικών τους.

3.1 Βασική Ορολογία

Ένα ***μη κατευθυνόμενο γράφημα (undirected graph)*** $G = (V, E)$ αποτελείται από ένα πεπερασμένο σύνολο ***κορυφών (vertices)*** V και ένα σύνολο ***ακμών (edges)*** E επί του V, ήτοι αδιάτακτα ζεύγη διακεκριμένων κορυφών (Σχήμα 3.1(α)). Έτσι, εάν $e = \{v, w\} \in E, v, w \in V$ λέμε πως η ακμή e *προσπίπτει στις* ή *έχει ως άκρα τις* κορυφές v, w. Το πλήθος των προσπιπτουσών ακμών μίας κορυφής v συνιστά τον ***βαθμό (degree)*** της $deg(v)$. Παρατηρήστε πως, εφ' όσον το E είναι σύνολο, μεταξύ δύο κορυφών δεν επιτρέπονται περισσότερες της μίας ακμές, ούτε βρόχοι· το γράφημα είναι, δηλαδή, ***απλό (simple)***. Όταν επιτρέπονται πολλαπλές διασυνδέσεις μεταξύ των κορυφών, τότε ομιλούμε για ***πολυγραφήματα (multigraphs)*** (Σχ. 3.1(β)), ενώ ως ***ψευδογραφήματα (pseudographs)*** χαρακτηρίζονται τα γραφήματα με βρόχους. Ένα απλό γράφημα χαρακτηρίζεται ως ***πλήρες (complete)***, όταν όλες οι κορυφές του συνδέονται μεταξύ τους και συμβολίζεται με K_n, όπου $n = |V|$ (Σχ. 3.1(γ)).

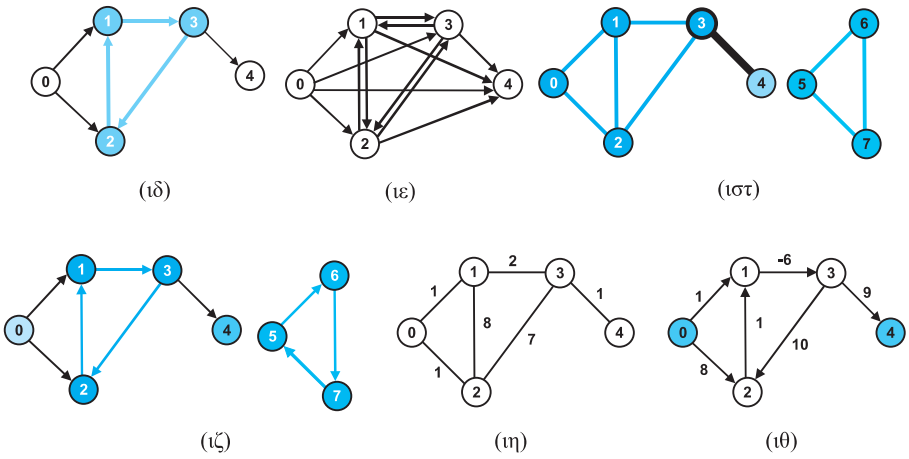
Ένα πεπερασμένο σύνολο κορυφών V και ένα σύνολο ***κατευθυνόμενων ακμών (directed edges)*** E επί του V —ήτοι διατεταγμένα ζεύγη διακεκριμένων κορυφών— συνιστούν ένα ***κατευθυνόμενο γράφημα (directed graph)*** $G = (V, E)$ (Σχ. 3.1(δ)). Έτσι, εάν $e = (v, w) \in E$, $v, w \in V$, λέμε πως η ακμή e κατευθύνεται από την v προς την w· πλέον, $(v, w) \neq (w, v)$. Το πλήθος των ακμών που ξεκινούν από μία κορυφή v αποτελούν τον ***βαθμό εξόδου (out-degree)*** της, ενώ το πλήθος των προσπιπτουσών ακμών της καλείται ***βαθμός εισόδου (in-degree)***. Κάθε κορυφή με βαθμό εισόδου 0 καλείται ***πηγή (source)***, ενώ οι κορυφές με βαθμό εξόδου 0 κατηγοριοποιούνται ως ***καταβόθρες (drains)*** (κορυφές 0 και 4, αντίστοιχα, στο εν λόγω Σχήμα). Δοθέντος ενός κατευθυνόμενου γραφήματος $G = (V, E)$, ορίζεται το ***ανάστροφο γράφημα (reverse graph)*** $G^R = (V, E^R)$, στο οποίο $(v, w) \in E^R$, $\forall (w, v) \in E$· τέτοιο γράφημα, για αυτό του Σχ. 3.1(δ), αποτελεί το εικονιζόμενο στο Σχ. 3.1(ε). Τέλος, κατά τρόπο ανάλογο ως προς τα μη κατευθυνόμενα, τα κατευθυνόμενα δύνανται να χαρακτηριστούν ως πολυγραφήματα και ψευδογραφήματα, ενώ χρησιμοποιούμε τον όρο πλήρη (απλά) κατευθυνόμενα γραφήματα σε περίπτωση που, μεταξύ δύο οποιωνδήποτε κορυφών, υφίσταται *ακριβώς μία* κατευθυνόμενη ακμή.

Τα παραδείγματα των ορισμών που θα ακολουθήσουν, αφορούν στο γράφημα του Σχ. 3.1(α): Ένα γράφημα $G' = (V', E')$ καλείται ***υπογράφημα (subgraph)*** ενός γραφήματος $G = (V, E)$, όταν $V' \subseteq V$ και $E' \subseteq E$ (Σχ. 3.1(στ)). Μάλιστα, όταν το σύνολο



Σχήμα 3.1: Στιγμιότυπα γραφημάτων.

E' περιλαμβάνει όλες τις ακμές του E που έχουν τα άκρα τους στο V', τότε ομιλούμε για ***επαγόμενο (induced)*** υπογράφημα (Σχ. 3.1(ζ)), ενώ, εάν $V' = V$, τότε αναφερόμαστε σε ***επικαλύπτον (spanning)*** υπογράφημα (Σχ. 3.1(η)). ***Συμπλήρωμα (complement)*** $\bar{G} = (V, \bar{E})$ ενός γραφήματος $G = (V, E)$ ορίζεται το γράφημα όπου το $\bar{E}$ συνίσταται από όλες τις ακμές του $K_{|V|}$ που δεν ανήκουν στο G —λ.χ., το Σχ. 3.1(θ) αποτελεί συμπλήρωμα του εν λόγω γραφήματος. Μία διαμέριση του συνόλου των κορυφών V σε δύο διαζευγμένα σύνολα V', V'' ορίζει μία ***τομή (cut)*** (V', V'') του γραφήματος· το παράδειγμα του Σχ. 3.1(ι) ορίζει την τομή $\{0, 1\}, \{2, 3, 4\}$. Οι δε ακμές που ενώνουν κορυφές του V' με κορυφές του V'' ονομάζονται ***ακμές τομής (cut edges)*** —στο παράδειγμά μας, είναι αυτές με έντονη διαγράμμιση. Επίσης, ένα γράφημα χαρακτηρίζεται ως ***διμελές (bipartite)***, εάν το σύνολο V των κορυφών του διαμερίζεται σε δύο διαζευγμένα υποσύνολα V' και V'', τέτοια ώστε κάθε ακμή του



Σχήμα 3.2: (συνέχεια) Στιγμιότυπα γραφημάτων.

γραφήματος να ενώνει κορυφή του πρώτου με κορυφή του δευτέρου. Το στιγμιότυπο του Σχ. 3.1(ια) αποτελείται από τα σύνολα $\{1, 2\}, \{3, 4\}$.

Μία ακολουθία $(v_1, v_2, \ldots, v_k)$ k κορυφών, με

$$\{v_i, v_{i+1}\} \in E, \quad \forall i \in [1, k-1],$$

αποτελεί ένα ***μονοπάτι (path)*** μήκους $k-1$. Όταν όλες οι κορυφές του μονοπατιού είναι διακεκριμένες, τότε έχουμε ***απλό μονοπάτι (simple path)***. Το μεγαλύτερο μήκος συντομότερου, άρα και απλού, μονοπατιού μεταξύ δύο κορυφών του γραφήματος αποτελεί την ***διάμετρο (diameter)*** του γραφήματος. Λόγου χάριν, η διάμετρος του γραφήματος στο Σχ. 3.1(α) είναι 3, και την καθορίζουν οι κορυφές 0 και 4. Ένα μονοπάτι γραφήματος καλείται ***μονοπάτι Euler***, εάν διαπερνά κάθε ακμή του γραφήματος *ακριβώς* μία φορά —παρατηρήστε ότι ένα τέτοιο γράφημα δύναται να σχεδιαστεί μονοκονδυλιά. Για παράδειγμα, η διαδοχική διαπέραση $4 \to 3 \to 2 \to 3 \to 1 \to 2 \to 0 \to 1$ των ακμών του Σχ. 3.1(β) συνιστά ένα μονοπάτι Euler. Σε περίπτωση που $v_1 = v_k$, η ακολουθία κορυφών χαρακτηρίζεται ως ***κύκλος (cycle)***, απλός ή μη. Ένας κύκλος που διαπερνά όλες τις ακμές ενός γραφήματος *ακριβώς* μία φορά αποκαλείται ***κύκλος Euler***. Λόγου χάριν, η διαπέραση $0 \to 1 \to 3 \to 4 \to 2 \to 0 \to 3 \to 2 \to 1 \to 4 \to 0$ των ακμών του Σχ. 3.1(γ) διαγράφει έναν κύκλο Euler. Στα κατευθυνόμενα γραφήματα ισχύουν οι ανάλογοι ορισμοί· με την διαφορά, πως χαρακτηρίζονται κατευθυνόμενα. Για παράδειγμα, στο Σχ. 3.1(ιβ), εικονίζεται το απλό, κατευθυνόμενο μονοπάτι $\{0, 1, 3, 4\}$, από την κορυφή 0 στην κορυφή 4, με ενδιάμεσες τις 3 και 4, το μονοπάτι $\{0, 1, 3, 2, 1\}$ του Σχ. 3.1(ιγ) δεν είναι απλό, ενώ το Σχ. 3.2(ιδ) παρουσιάζει τον (απλό) κύκλο $\{1, 3, 2, 1\}$. Δύο μονοπάτια (κύκλοι) καλούνται ***διαζευγμένα ως προς τις κορυφές (ακμές) (vertex (edge) disjoint)*** εάν δεν διαθέτουν κοινές κορυφές (ακμές). Η ***μεταβατική κλειστότητα (transitive closure)*** ενός κατευθυνόμενου γραφή-

ματος $G = (V, E)$ ορίζεται ως ένα γράφημα $G^* = (V, E^*)$, με $E \subseteq E^*$, στο οποίο δύο κορυφές $v, w \in V$ διασυνδέονται με μία κατευθυνόμενη ακμή e αν και μόνον αν υπάρχει στο G κατευθυνόμενο μονοπάτι που τα συνδέει. Το Σχήμα 3.2(ιε) αποτελεί την μεταβατική κλειστότητα του Σχ. 3.1(δ).

Ένα γράφημα ορίζεται ως ***συνεκτικό (connected)***, όταν για οποιαδήποτε δύο κορυφές υπάρχει μονοπάτι που τις συνδέει. Σε αντίθετη περίπτωση, καλείται ***μη συνεκτικό*** και αποτελείται από ένα σύνολο ***συνεκτικών συνιστωσών (connected components)***, ήτοι από μέγιστα συνεκτικά υπογραφήματα. Το μη κατευθυνόμενο γράφημα, λ.χ., του Σχ. 3.2(ιστ) αποτελείται από δύο συνεκτικές συνιστώσες: η μία περιλαμβάνει τις κορυφές $0, 1, 2, 3$ και 4 και η άλλη αποτελείται από τις κορυφές $5, 6, 7$.

Τα μη κατευθυνόμενα γραφήματα χαρακτηρίζονται ως ***δισυνεκτικά (biconnected)***, όταν μεταξύ δύο οποιαδήποτε κορυφών υπάρχουν διαζευγμένα, ως προς τις κορυφές, μονοπάτια που τις συνδέουν. Σε περίπτωση που το γράφημα δεν είναι δισυνεκτικό, τότε ως ***δισυνεκτικές συνιστώσες (biconnected components)*** ορίζονται τα μέγιστα υπογραφήματά του που διαθέτουν την ιδιότητα της δισυνεκτικότητας. Πρέπει να σημειωθεί, πως εάν ένα γράφημα είναι συνεκτικό, δεν είναι, κατ' ανάγκην και δισυνεκτικό, ενώ, είναι δυνατόν, μία κορυφή να ανήκει σε περισσότερες της μίας δισυνεκτικές συνιστώσες. Στο παράδειγμα του Σχήματος 3.2(ιστ), υπάρχουν τρεις δισυνεκτικές συνιστώσες, οι $\{0, 1, 2, 3\}, \{4\}$ και $\{5, 6, 7\}$. Μία ακμή καλείται ***γέφυρα (bridge)***, όταν η αφαίρεσή της αυξάνει το πλήθος των συνεκτικών συνιστωσών· η ακμή μεταξύ των κορυφών 3 και 4 στο Σχήμα 3.2(ιστ) αποτελεί τέτοιου είδους ακμή. Κατά αναλογία, μία κορυφή χαρακτηρίζεται ως ***σημείο αρθρώσεως (articulation point)*** ή ***κορυφή αποκοπής (cut vertex)***, όταν η απόσβεσή της επαυξάνει τον αριθμό των συνεκτικών συνιστωσών· π.χ., η κορυφή 3 του Σχ. 3.2(ιστ) αποτελεί σημείο αρθρώσεως. Αποδεικνύεται πως ένα γράφημα είναι δισυνεκτικό εάν και μόνον αν δεν υπάρχουν σημεία αποκοπής, ενώ κάθε μη δισυνεκτικό γράφημα συνίσταται από γέφυρες και δισυνεκτικές συνιστώσες.

Τα κατευθυνόμενα γραφήματα χαρακτηρίζονται ***ισχυρά συνεκτικά (strongly connected)***, όταν μεταξύ δύο οποιαδήποτε κορυφών v, w υπάρχουν δύο κατευθυνόμενα μονοπάτια, όχι απαραίτητα διαζευγμένα, $v \rightsquigarrow w, w \rightsquigarrow v$ που τις συνδέουν. Οπότε, οι ***ισχυρά συνεκτικές συνιστώσες (strongly connected components)*** ορίζονται ως τα μέγιστα υπογραφήματα με την ιδιότητα της ισχυρής συνεκτικότητας. Λόγου χάριν, το κατευθυνόμενο γράφημα του Σχήματος 3.2(ιζ) διαθέτει τις ακόλουθες τέσσερις ισχυρά συνεκτικές συνιστώσες, τις $\{0\}, \{1, 2, 3\}, \{4\}$ και $\{5, 6, 7\}$. Από την άλλη, τα κατευθυνόμενα γραφήματα χαρακτηρίζονται ***ασθενώς συνεκτικά (weakly connected)***, όταν, αγνοώντας την φορά των ακμών, αποδεικνύονται συνεκτικά.

Όταν ένα γράφημα είναι *άκυκλο* και *συνεκτικό* ονομάζεται ***δένδρο (tree)***, π.χ., θεωρήστε το στιγμιότυπο του Σχ. 3.1(θ), ενώ, εάν είναι *άκυκλο* και *μη συνεκτικό*, χαρακτηρίζεται ως ***δάσος (forest)***. Στα κατευθυνόμενα γραφήματα, η έλλειψη κατευθυνόμενων κύκλων τα κατηγοριοποιεί ως ***κατευθυνόμενα άκυκλα γραφήματα —ΚΑΓ (DAG)***. Για

παράδειγμα, εάν αλλάξουμε την φορά της ακμής που συνδέει τις κορυφές 2 και 3, το γράφημα του Σχήματος 3.1(δ) γίνεται ΚΑΓ. Ακόμη, ένα γράφημα καλείται ***πυκνό (dense)*** εάν $|E| = \Theta(|V|^2)$· διαφορετικά, έχουμε ένα ***αραιό (sparse)*** γράφημα. Τέλος, όταν σε κάθε ακμή συσχετίζουμε έναν αριθμό, το γράφημα αποκαλείται ***βεβαρημένο (weighted)***. Μάλιστα, στην περίπτωση των κατευθυνόμενων βεβαρημένων γραφημάτων συναντάται και ο όρος ***δίκτυο (network)***· τα γραφήματα των σχημάτων 3.2(ιη) και (ιθ) αποτελούν τέτοιες περιπτώσεις. ***Βάρος*** ή ***κόστος μονοπατιού*** ορίζεται το άθροισμα των βαρών των εμπεριεχομένων ακμών. Παραδείγματος χάριν, το κόστος του μονοπατιού από την κορυφή 0 στην κορυφή 3 μέσω της δύο, στο γράφημα του Σχήματος 3.2(ιη), είναι 8.

Κλείνοντας, παραθέτουμε τις γνωστότερες ποσοτικές συσχετίσεις στα γραφήματα, σημειώνοντας πως, στις παραγράφους που έπονται, θα ακολουθήσουμε την «συνήθη» τακτική της καταχρηστικής χρησιμοποιήσεως των συμβόλων V, E στην θέση των $|V|$ και $|E|$ αντιστοίχως.

Θεώρημα 3.1 *Σε κάθε μη κατευθυνόμενο γράφημα $G = (V, E)$ ισχύει ότι:*

$$\sum_{v \in V} \deg(v) = 2|E|,$$

στα κατευθυνόμενα $G = (V, E)$ έχουμε ότι:

$$\sum_{v \in V} \text{in-deg}(v) = \sum_{v \in V} \text{out-deg}(v) = |E|,$$

σε κάθε δένδρο $T = (V, E)$

$$|E| = |V| - 1,$$

ενώ για κάθε δάσος F t δένδρων:

$$|E| = |V| - t.$$

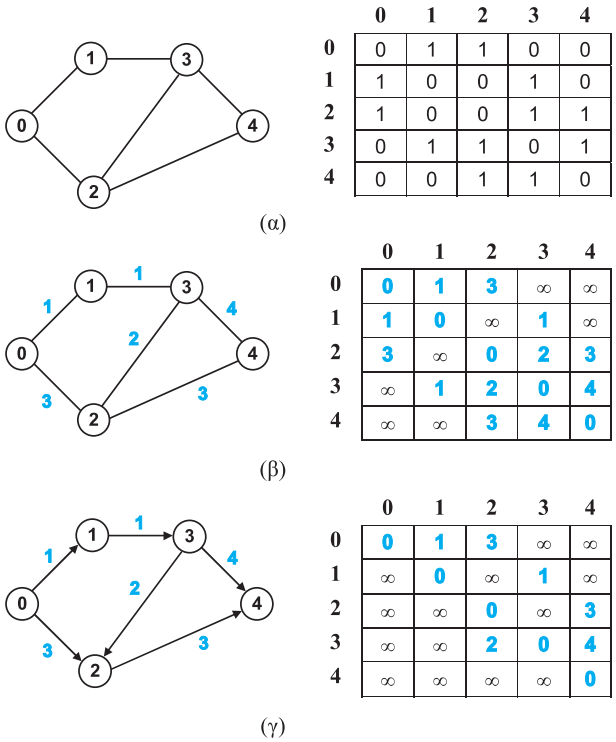
3.2 Αναπαράσταση Γραφημάτων

Το άθροισμα $V + E$ χαρακτηρίζει πλήρως την πολυπλοκότητα ενός γραφήματος. Κατά συνέπεια, θα αποτελεί μέτρο της αποτελεσματικότητας της εκάστοτε επιλογής «εσωτερικής» αναπαραστάσεως ενός γραφήματος. Υπάρχουν δύο κυρίαρχες αναπαραστάσεις, ο ***πίνακας γειτνιάσεως*** και η ***λίστα γειτνιάσεως***, εκάστη με τα μειονεκτήματα και τα πλεονεκτήματά της, αλλά σίγουρα καλύτερη από την τρίτη —και, κατά το μάλλον ή ήττον, λιγότερο ενδιαφέρουσα— μορφή του ***πίνακα ακμών (edge matrix)***.

Στην συζήτηση που ακολουθεί, χωρίς βλάβη της γενικότητας και για προγραμματιστικούς, κυρίως, λόγους, θα ακολουθήσουμε την σύμβαση πως οι κορυφές φέρουν,

ως ονόματα, τους ακεραίους $0, \ldots, V-1$. Κάτι τέτοιο είναι εφικτό, καθώς οποιαδήποτε ονοματολογία δύναται να μετασχηματιστεί στην εν λόγω μορφή μέσω μίας οποιαδήποτε δομής λεξικού. Για λόγους κατανοήσεως, κατά την περιγραφή ψευδοκώδικα, θα μεταχειριζόμαστε εναλλακτικά τους τύπους **int** και **vertex** για την κατάδειξη αντικειμένου-κορυφή.

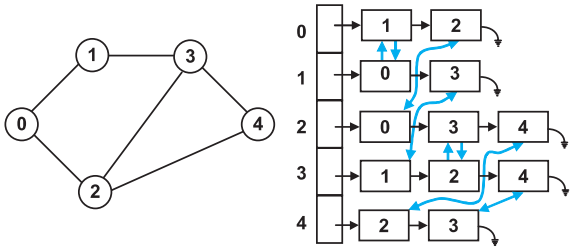
Πίνακας Γειτνιάσεως (Adjacency Matrix). Η προσέγγιση αυτή χρησιμοποιεί έναν δισδιάστατο δυαδικό (boolean) πίνακα A, διαστάσεων $V \times V$, όπου η τιμή '0' στο στοιχείο $A[v][w]$ δεικνύει την απουσία ακμής μεταξύ των κορυφών v, w, ενώ η τιμή '1' δηλοί την διασύνδεση των εν λόγω κορυφών. Προκύπτει πολύ εύκολα πως, στην περίπτωση των μη κατευθυνόμενων γραφημάτων, ο A είναι συμμετρικός ως προς την κύρια διαγώνιο, κάτι που δεν παρατηρείται στα κατευθυνόμενα γραφήματα. Στα βεβαρημένα γραφήματα, αντί για δυαδικό, χρησιμοποιούμε έναν γενικό πίνακα τιμών, με κάθε στοιχείο να αποθηκεύει το βάρος της αντίστοιχης ακμής. Το Σχήμα 3.3 παρουσιάζει στιγμιότυπα αυτής της αναπαραστάσεως.



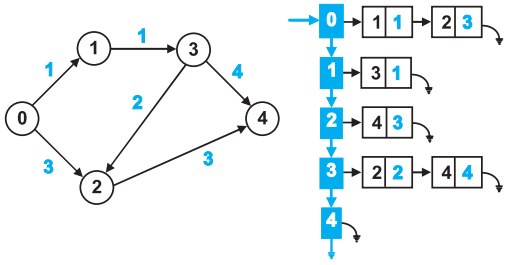
Σχήμα 3.3: Αναπαράσταση πίνακα γειτνιάσεως: (α) μη κατευθυνόμενο γράφημα, (β) βεβαρημένο γράφημα, και (γ) κατευθυνόμενο βεβαρημένο γράφημα.

Η αναπαράσταση με πίνακα είναι ιδανική για πυκνά γραφήματα, καθιστώντας τον έλεγχο διασυνδέσεως μεταξύ δύο οποιωνδήποτε κορυφών πράξη σταθερού κόστους. Επίσης, η αποτροπή ενθέσεως πολλαπλών ακμών μεταξύ του ιδίου ζεύγους κορυφών είναι εγγενής ιδιότητα της μεθόδου. Από την άλλη, συνιστά άσκοπη σπατάλη για αραιά γραφήματα, όπου το $V + E = \Theta(V)$, ενώ η εύρεση του βαθμού μίας κορυφής είναι αρκετά ακριβή πράξη, καθώς είναι γραμμική στο πλήθος των κορυφών του γραφήματος και όχι στο πλήθος των γειτονικών κορυφών. Τέλος, η αφαίρεση μίας κορυφής αποτελεί, επίσης, επίπονη πράξη, αφού αιτεί αναδιαμόρφωση του πίνακα.

Λίστα Γειτνιάσεως (Adjacency List). Η λίστα γειτνιάσεως αποτελεί την ασφαλή επιλογή για αραιά γραφήματα. Συγκεκριμένα, αποτελείται από έναν πρωτεύοντα πίνακα ή μία πρωτεύουσα λίστα σε δευτερεύουσες λίστες (Σχήμα 3.4). Κάθε στοιχείο της κύριας λίστας ή του κύριου πίνακα αντιστοιχεί σε μία κορυφή, την οποία συσχετίζει με την δευτερεύουσα λίστα των γειτονικών προς αυτήν κορυφών. Στην περίπτωση των μη κατευθυνόμενων γραφημάτων, κάθε ακμή πρέπει να αποθηκευτεί *και στις δύο* δευτερεύουσες λίστες των άκρων της· συνήθως δε, για λόγους αποδόσεως κατά την απόσβεση μίας κορυφής, οι δύο αυτές εγγραφές αλληλοσυσχετίζονται με δείκτες.

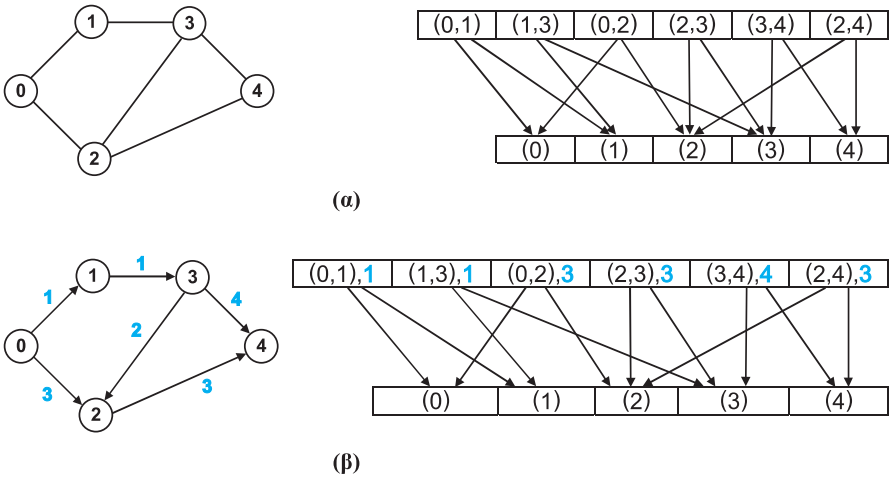


(α)



(β)

Σχήμα 3.4: Αναπαράσταση λίστας γειτνιάσεως: (α) μη κατευθυνόμενο γράφημα μέσω πρωτεύοντα πίνακα και συσχετίσεις συνδεδεμένων κορυφών και (β) βεβαρημένο κατευθυνόμενο γράφημα με πρωτεύουσα λίστα.



Σχήμα 3.5: Αναπαράσταση πίνακα ακμών: (α) μη κατευθυνόμενο γράφημα, (β) κατευθυνόμενο βεβαρημένο γράφημα.

Προκύπτει, πολύ εύκολα, πως η αναπαράσταση αυτή στοιχίζει $V+E$, ενώ η εύρεση των γειτόνων μίας κορυφής αποτελεί πράξη γραμμική στον βαθμό της. Από την άλλη, παύουν να είναι σταθερού κόστους πράξεις, η αποτροπή παράλληλων ακμών, η διαπίστωση εάν δύο κορυφές διασυνδέονται ή όχι, και η αφαίρεση μίας ακμής. Για την τελευταία πράξη, είθισται να μεταχειρίζεται κανείς μία βοηθητική δομή λεξικού (συνήθως, έναν πίνακα κατακερματισμού) για τον εντοπισμό της σε, σχεδόν, σταθερό χρόνο. Τέλος, η λίστα γειτνιάσεως εύκολα μπορεί να αναπαραστήσει βεβαρημένα γραφήματα: αρκεί να αποθηκευτεί το βάρος στις εγγραφές των δευτερευουσών λιστών.

Πίνακας Ακμών (Edge Matrix). Συνιστά την ευθύτερη αναπαράσταση ενός γραφήματος. Μεταχειρίζεται δύο πίνακες: στον ένα αποθηκεύονται οι ακμές, με τα τυχόν βάρη τους και ένδειξη εάν είναι κατευθυνόμενες ή όχι, και στον άλλο οι κορυφές, ενώ υπάρχουν δείκτες (pointers) μεταξύ των ακμών και των κορυφών-άκρων τους. Το Σχήμα 3.5 παρουσιάζει στιγμιότυπα αναπαραστάσεως μη κατευθυνόμενου και κατευθυνόμενου βεβαρημένου γραφήματος. Είναι, βέβαια, δυνατόν να εξοικονομηθεί ο δεύτερος πίνακας των κορυφών, εάν υιοθετηθεί η σύμβαση της ονομασίας των κορυφών με ακεραίους, καθώς δεν θα υπάρχει η ανάγκη δημιουργίας αντικειμένων-στιγμιοτύπων μιας ειδικής κλάσεως vertex για αυτές.

Το κύριο χαρακτηριστικό της είναι η άμεση πρόσβαση από τις ακμές στις προσκείμενες κορυφές. Όμως, η αντίστροφη πράξη της πρόσβασης των ακμών που είναι προσκείμενες σε δοθείσα κορυφή απαιτεί την εξαντλητική αναζήτηση του συνόλου των ακμών. Της ίδιας δυσκολίας είναι και ο έλεγχος διασύνδεσης δύο κορυφών, όπως και η διαγραφή κορυφής.

Συμβάσεις για την Περιγραφή ενός Γραφήματος. Στις Παραγράφους και τα Κεφάλαια που ακολουθούν, παρέχεται ένας ικανός αριθμός αλγορίθμων, που αφορούν τα γραφήματα, εκφρασμένων σε ψευδογλώσσα. Θα ακολουθηθεί η σύμβαση πως τα γραφήματα-είσοδοι σε αυτούς τους αλγορίθμους αποτελούν στιγμιότυπα της κλάσεως graph. Αυτή, στην βασική διάρθρωσή της, θα περιέχει τις μεταβλητές V, E του πλήθους των κορυφών και των ακμών του γραφήματος, αντιστοίχως, και, εάν χρησιμοποιείται η αναπαράσταση λίστας, ο πρωτεύων πίνακας λιστών γειτνιάσεως List, διαφορετικά, ο πίνακας γειτνιάσεως A. Εάν απαιτείται και ο πίνακας ακμών, αυτός θα δίδεται ως Edges. Κάθε κόμβος της List θα περιέχει την μεταβλητή v της κορυφής που στεγάζει, και, ενδεχομένως, το αντίστοιχο βάρος weight. Από την άλλη, ο Edges θα στεγάζει αντικείμενα της κλάσεως edge. Τα τελευταία θα περιέχουν τις μεταβλητές v, w των άκρων τους και, ενδεχομένως, το βάρος τους weight.

```
Algorithm graphSearchStart(graph g)
Input:  Γράφημα g
Output: Διαπέραση του g βάσει της διαδικασίας graphSearch

1.  int v, order = 0;   // η order χρησιμοποιείται για την διάταξη των κορυφών
2.  for (v = 0; v < g.V; v++)   // αρχικοποίηση ως ανεξέταστες
3.      g.pre[v] = −1;
4.  for (v = 0; v < g.V; v++)
5.    if (g.pre[v] == −1)
6.      graphSearch(g,v);
```

Αλγόριθμος 3.1: Εκκίνηση της διαδικασίας ψαξίματος graphSearch

3.3 Διαπεράσεις Μη Κατευθυνόμενων Γραφημάτων

Με τον όρο ***διαπέραση*** ή ***διάβαση γραφήματος*** (***graph traversal***) χαρακτηρίζεται η διαδικασία συστηματικής και εξαντλητικής εξερευνήσεως ενός γραφήματος, με την εξέταση των κορυφών και των ακμών του. Παρ' όλο που, όπως θα δούμε, οι εφαρμοζόμενες μέθοδοι είναι πολύ απλές και κατανοητές, εν τούτοις είναι τόσο ισχυρές που δύνανται να χρησιμοποιηθούν για την ανακάλυψη ιδιοτήτων, όπως, λ.χ., συνεκτικότητα ή ύπαρξη κύκλων, των υποκείμενων γραφημάτων.

Υποθέτουμε ότι οι εκάστοτε διαδικασίες ψαξίματος graphSearch εκκινούνται από έναν γενικό αλγόριθμο της μορφής του Αλγ. 3.1, κάνοντας χρήση ενός πίνακα pre, ο οποίος θα παρέχεται από την κλάση **graph** των γραφημάτων—χωρίς αυτό να είναι δεσμευτικό— και θα χρησιμεύει για την καταγραφή της σειράς επισκέψεως των κορυφών. Κατ' αυτόν τον τρόπο, εξασφαλίζεται η πλήρης διαπέραση των μη συνεκτικών γραφημάτων. Οι αριθμοί του pre είναι γνωστοί και ως σειρά ***προδιατάξεως (pre-order)***, καθώς, σε κάθε κορυφή, κατά την ανακάλυψή της, αποδίδεται ένας αριθμός-προτεραιότητα, προτού εξεταστούν οι τυχόν γειτονικές προς αυτήν κορυφές.

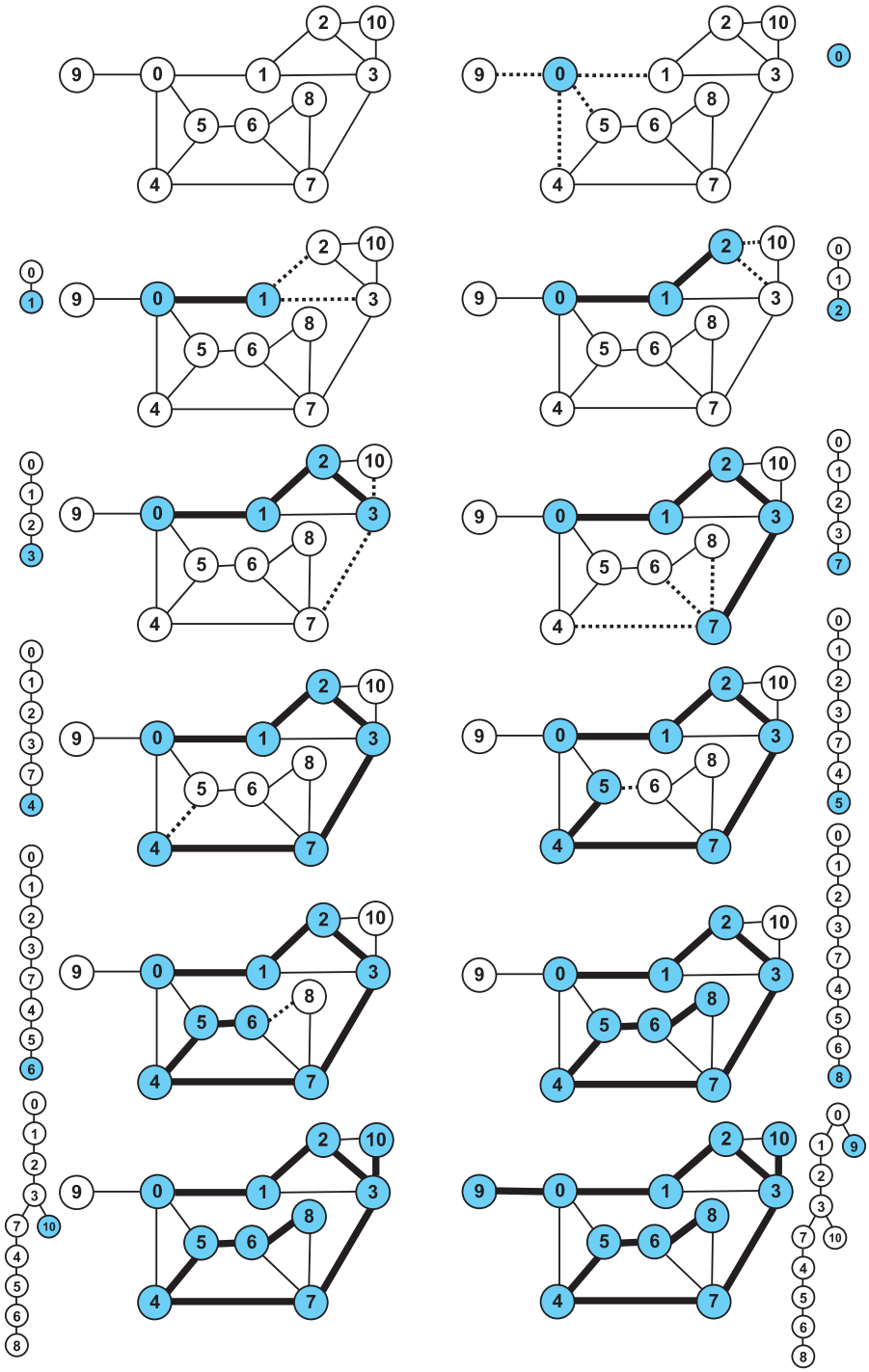
3.3.1 Αναζήτηση σε Βάθος (ΑσΒ)

Η ***αναζήτηση σε βάθος —ΑσΒ (depth first search —DFS)*** ενός γραφήματος G δίδει προτεραιότητα στις κορυφές που παραμένουν ανεξερεύνητες. Πιο συγκεκριμένα, κατά την εκκίνηση της διαδικασίας ΑσΒ, όλες οι κορυφές του υπό εξέταση γραφήματος σημειώνονται ως *ανεξέταστες* ή *ανεξερεύνητες*. Κατόπιν, μέχρι να επιθεωρηθούν όλες οι κορυφές, συστηματικά εφαρμόζουμε τα ακόλουθα δύο βήματα:

(α) επιλέγουμε μία ανεξερεύνητη κορυφή v και την κηρύσσουμε εξερευνημένη, και

(β) εάν η v διαθέτει μία ανεξέταστη γείτονα κορυφή w, μεταβαίνουμε στην w, την σημειώνουμε ως εξερευνημένη και, αναδρομικά, εφαρμόζουμε την διαδικασία αναζητήσεως σε κάποια γειτονική ανεξερεύνητη κορυφή της. Είθισται, εάν υπάρχουν περισσότερες της μίας ανεξερεύνητες γειτονικές κορυφές, να επιλέγεται αυτή με το μικρότερο όνομα. Η σύμβαση αυτή θα υιοθετηθεί για την συνέχεια.

Το Σχήμα 3.6 αποτελεί στιγμιότυπο τρεξίματος ΑσΒ, με κορυφή εκκινήσεως την 0. Οι «ακολουθούμενες» ακμές εικονίζονται με έντονο χρώμα, ενώ με κουκκίδες σημειώνονται οι εναλλακτικές επιλογές για την επόμενη επίσκεψη κορυφής. Παραπλεύρως, εικονίζεται η —δενδρικής μορφής— ακολουθία των αναδρομικών κλήσεων, η οποία είναι γνωστή και ως ***δένδρο ΑσΒ (DFS tree)***. Έτσι, από την 0, υπάρχει η δυνατότητα επισκέψεως των $1, 4, 5$ και 9. Από αυτές, λόγω συμβάσεως, διαλέγουμε την 1. Στην 1 έχουμε τις εναλλακτικές επιλογές των 2 και 3 και επιλέγουμε την 2 κ.ο.κ. μέχρι να φθάσουμε στην 4, όπου, για πρώτη φορά, θα έχουμε μοναδική επιλογή την 5, καθώς η 0 έχει ήδη εξερευνηθεί. Από την 5 μεταβαίνουμε, με υποχρεωτικό τρόπο, στην 6,



Σχήμα 3.6: ΑσΒ σε μη κατευθυνόμενο, συνεκτικό γράφημα.

από την 6 στην 8, επίσης αναγκαστικά. Από την 8, εφ' όσον δεν υπάρχουν άλλες ανεξέταστες γειτονικές κορυφές, οπισθοχωρούμε στην 6. Από την 6, για τον ίδιο λόγο, γυρίζουμε στην 5, από την 5 στην 4 κ.ο.κ. μέχρι την 3, η οποία διασυνδέεται με την ανεξέταστη 10. Από την 10, οπισθοχωρούμε στην 3, από την 3 στην 2, από την 2 στην 1, από την 1 στην 0. Η τελευταία συνδέεται με την 9. Μεταβαίνουμε στην 9, από την 9 πίσω στην 0 και η αναζήτηση ολοκληρώνεται.

Εάν χρησιμοποιείται η αναπαράσταση του γραφήματος με πίνακα γειτνιάσεως, τότε έχουμε τον Αλγόριθμο 3.2:

```
Algorithm dfsMatrix(graph g, vertex v)
Input:   Το γράφημα g και μία κορυφή εκκινήσεως v
Output:  ΑκΒ

1.  g.pre[v] = order++;        // βοηθητικός πίνακας προδιατάξεως
2.  for (w = 0; w < g.V; w++)
3.    if (g.A[v][w] != 0)      // υπάρχει ακμή (v, w)
4.      if (g.pre[w] == -1)    //  η w δεν έχει ανακαλυφθεί ακόμη
5.        dfsMatrix(g,w);
```

Αλγόριθμος 3.2: Αναζήτηση σε Βάθος για αναπαράσταση με πίνακα

ο οποίος επισκέπτεται, αναδρομικά, την πρώτη, κατά αύξουσα ονομασία, ανεξερεύνητη γειτονική κορυφή, συνδυάζοντας τις πληροφορίες διασυνδέσεως του πίνακα γειτνιάσεως A και επισκέψεως του βοηθητικού πίνακα pre.

Όταν το γράφημα αναπαρίσταται με λίστα γειτνιάσεως, τότε η διαδικασία περιγράφεται από τον Αλγ. 3.3:

```
Algorithm dfsList(graph g, vertex v)
Input:   Το γράφημα g και μία κορυφή εκκινήσεως v
Output:  ΑκΒ

1.  g.pre[v] = order++;        // βοηθητικός πίνακας προδιατάξεως
2.  for (x = g.List[v]; x != null; x = x.getNext()) // όσο υπάρχει ακμή (v, x.v)
3.    if (g.pre[x.v] == -1)    // η κορυφή x.v δεν έχει ανακαλυφθεί ακόμη
4.      dfsList(g,x.v);
```

Αλγόριθμος 3.3: Αναζήτηση σε Βάθος για αναπαράσταση με λίστα.

ο οποίος, εν αντιθέσει προς τον Αλγ. 3.2, ελέγχει *απ' ευθείας* εάν οι γειτονικές της εκάστοτε κορυφής v έχουν ήδη ανακαλυφθεί, και μόνον αυτές, σαρώνοντας την αντίστοιχη δευτερεύουσα λίστα List[v].

Θεώρημα 3.2 *Η διαπέραση ΑσΒ περατώνει σε χρόνο γραμμικό ως προς την πολυπλοκότητα του υπό εξερεύνηση γραφήματος G.*

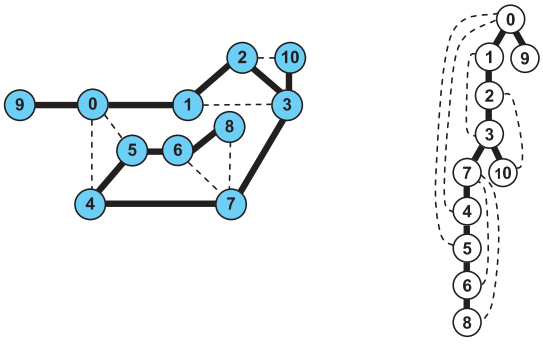
Απόδειξη. Προκύπτει άμεσα, παρατηρώντας πως κάθε κορυφή σημειώνεται ακριβώς μία φορά ως εξερευνημένη, ενώ κάθε ακμή διερευνάται το πολύ δύο φορές, μία από κάθε άκρο της, κατά τον έλεγχο, εάν το άλλο άκρο είναι ανεξερεύνητο ή όχι. ■

Θυμηθείτε, τώρα, πως η ακολουθία των αναδρομικών κλήσεων της ΑσΒ αναπαρίσταται με το δένδρο ΑσΒ. Έστω, λοιπόν, T_G το δένδρο ΑσΒ που προκύπτει από μία αναζήτηση σε βάθος ενός γραφήματος G. Βάσει αυτού, οι ακμές του G διακρίνονται σε:

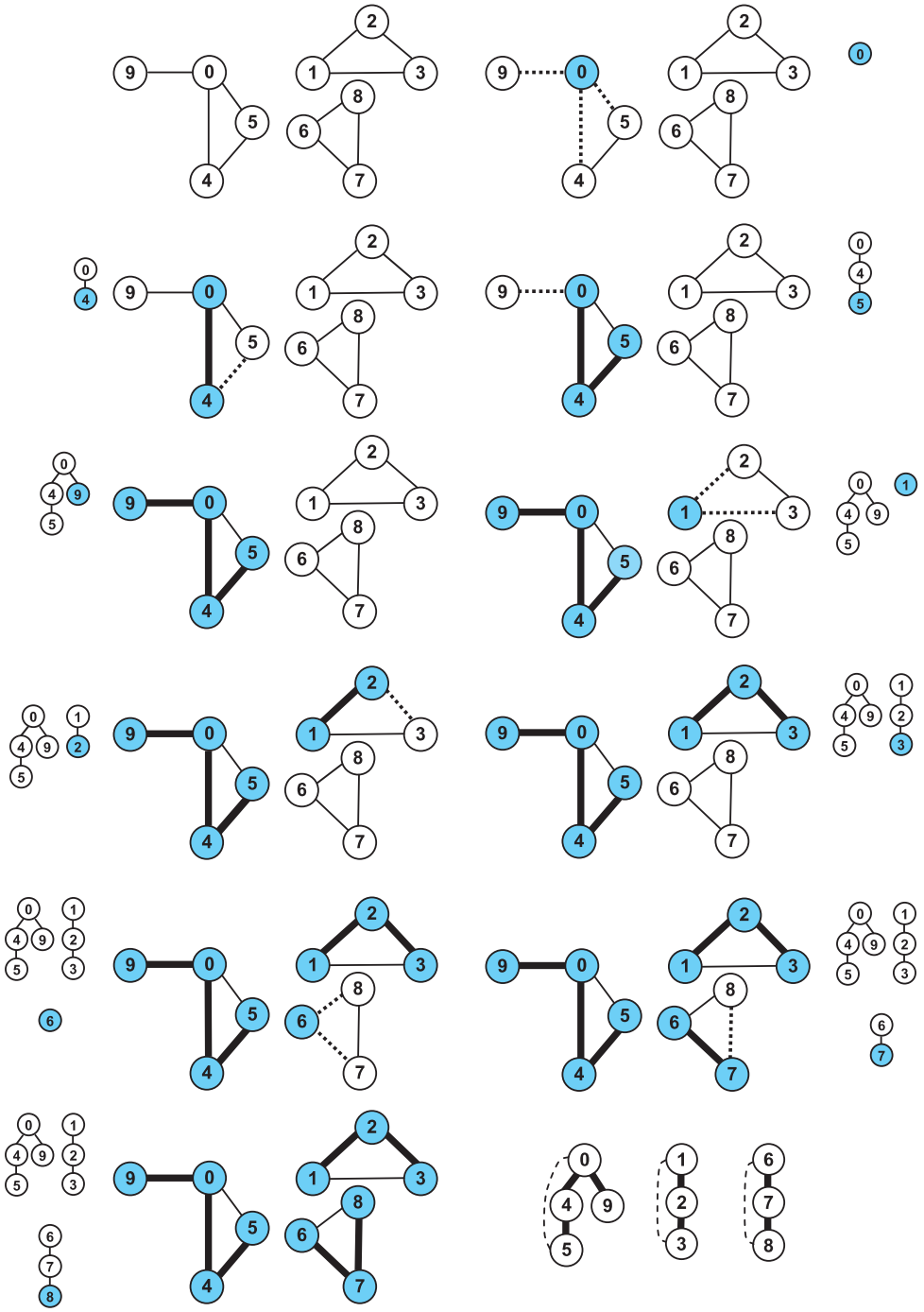
ακμές δένδρου (tree edges), εάν αποτελούν τμήμα του T_G, και

οπισθοακμές (back-edges), εάν συνδέουν κορυφές v, w που επιδεικνύουν σχέση προγόνου-απογόνου ως προς το T_G. Παρατηρήστε πως, σε αυτήν την περίπτωση, $G.\mathrm{pre}[v] < G.\mathrm{pre}[w]$.

Το Σχήμα 3.7 παρουσιάζει την κατηγοριοποίηση των ακμών για το παράδειγμα του Σχήματος 3.6. Αξίζει να σημειωθεί πως το ανάστροφο δένδρο T_G^R του T_G δύναται να κατασκευαστεί άμεσα, κατά την διάρκεια κλήσης είτε του Αλγ. 3.2 είτε του Αλγ. 3.3, μέσω ενός βοηθητικού πίνακα $G.T$: κάθε φορά που μεταβαίνουμε από μία κορυφή v σε μία γειτονική, ανεξερεύνητη κορυφή w, αναθέτουμε $G.T[w] = v$, ώστε να σημειωθεί πως πατέρας του w, στο δένδρο T_G, είναι η κορυφή v. Εάν το υπό εξερεύνηση γράφημα δεν είναι συνεκτικό, τότε η διαδικασία δύναται να αναπαρασταθεί από ένα *δάσος δένδρων ΑσΒ* (Σχήμα 3.8). Γενικά, πάντως, ένα δένδρο ΑσΒ διακρίνεται για το μεγάλο ύψος του και όχι τόσο για το άπλωμά του. Ήτοι, επιδεικνύει μικρούς, κατά κανόνα, βαθμούς εξόδου.



Σχήμα 3.7: Κατηγοριοποίηση των ακμών γραφήματος (δεξιά) βάσει του δένδρου ΑσΒ που αντιστοιχεί στην εικονιζόμενη ΑσΒ (αριστερά). Με παχιές ακμές, σημειώνονται οι ακμές δένδρου, ενώ, με λεπτές διακεκομμένες, οι οπισθοακμές.



Σχήμα 3.8: ΑσΒ σε μη κατευθυνόμενο, μη συνεκτικό γράφημα. Στο δεξιό κάτω μέρος του Σχήματος απεικονίζεται η κατηγοριοποίηση των συνεκτικών συνιστωσών βάσει του δάσους ΑσΒ.

Με τροποποίηση των Αλγορίθμων 3.2 και 3.3 καθίσταται δυνατή και η εύρεση των αριθμών ***μεταδιατάξεως (postorder)***, οι οποίοι αντιστοιχούν στην αντίστροφη σειρά (περατώσεως) της αναζητήσεως· με άλλα λόγια, κάθε νεο-ανακαλυφθείσα κορυφή λαμβάνει τον πρώτο διαθέσιμο αριθμό, *αφού* τερματίσουν οι αναδρομικές κλήσεις προς όλες τις τυχόν ανεξέταστες γειτονικές κορυφές. Αρκεί, λοιπόν, ο τρέχων διαθέσιμος αριθμός $order$ να σημειώνεται σε έναν επιπλέον πίνακα $post$, *μετά* την επεξεργασία των προσκείμενων κορυφών. Με διαθέσιμους και τους δύο πίνακες $post, pre$ είναι δυνατή η εύρεση της σχέσεως, ως προς την αντίστοιχη ΑσΒ, μεταξύ δύο κορυφών v, w:

Λήμμα 3.1 *Δοθέντων των πινάκων προδιατάξεως και μεταδιατάξεως* $post, pre$*,*

η v *είναι απόγονος της* w*, αν και μόνον αν* $pre(v) > pre(w)$ *και* $post(v) < post(w)$*.*

η v *είναι πρόγονος της* w*, αν και μόνον αν* $pre(v) < pre(w)$ *και* $post(v) > post(w)$*.*

η v *δεν σχετίζεται με την* w*, σε κάθε άλλη περίπτωση.*

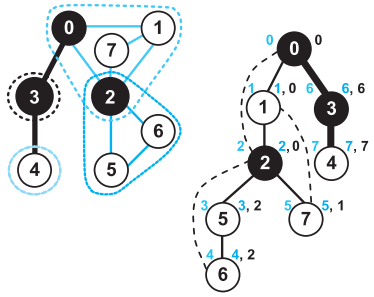
Απόδειξη. Άμεση από την διαδικασία αποδόσεως των αριθμών προδιατάξεως και μεταδιατάξεως. ■

Εφαρμογές. Η ΑσΒ δύναται να επιλύσει τα ακόλουθα προβλήματα σε μη κατευθυνόμενα γραφήματα:

Εντοπισμός κύκλου: Γίνεται άμεσα, αφ' ης στιγμής εντοπιστεί, με χρήση του πίνακα pre, η πρώτη οπισθοακμή.

Εύρεση απλού μονοπατιού μεταξύ δύο κορυφών v, w: Αρκεί να εκκινήσουμε από την v μία ΑσΒ. Εάν όντως υπάρχει μονοπάτι, η διαπέραση θα το εντοπίσει. Τυπική απόδειξη για τον ισχυρισμό αυτό προκύπτει πολύ εύκολα, με επαγωγή στο μήκος του μονοπατιού.

Επικαλύπτον Δένδρο ή Δάσος, Απλή Συνεκτικότητα: Εάν το γράφημα είναι συνεκτικό, τότε το δένδρο ΑσΒ συνιστά ένα επικαλύπτον δένδρο. Διαφορετικά, οι t διαφορετικές επανεκκινήσεις της ΑσΒ ανακαλύπτουν τα επικαλύπτοντα δένδρα των t συνεκτικών συνιστωσών. Οι τροποποιήσεις που απαιτούνται για την επίτευξη του σκοπού είναι ελάχιστες: Αρκεί να χρησιμοποιηθεί ένας βοηθητικός πίνακας cc, για την αντιστοίχιση κάθε κορυφής με την συνεκτική συνιστώσα που ανήκει, και μία βοηθητική μεταβλητή $ccnumber$. Στην γραμμή 1 του Αλγ. 3.2 ή του Αλγ. 3.3 πρέπει να αναθέτεται στην v ο τρέχων αριθμός συνιστώσας $ccnumber$ (δηλαδή, $\mathrm{cc}[v] = ccnumber$), ενώ, σε κάθε επανεκκίνηση της γραμμής 4 του Αλγ. 3.1, είναι ανάγκη να αυξάνεται ο $ccnumber$ κατά ένα.



Σχήμα 3.9: Σημεία αρθρώσεως (κορυφές 0, 2, 3) και δισυνεκτικές συνιστώσες (εντός διάστικτων καμπύλων). Με παχιές γραμμές σημειώνονται οι γέφυρες. Δεξιά έκαστου κόμβου, εικονίζεται ο αριθμός προδιατάξεως, ενώ, αριστερά, ο αριθμός low με τις ενδιάμεσες μεταβολές του.

Σημεία Αρθρώσεως: Μία κορυφή v αποτελεί σημείο αρθρώσεως, εφ' όσον είτε δεν υπάρχει απόγονός της στο δένδρο ΑσΒ που να διασυνδέεται με οπισθοακμή με έναν πρόγονο της v (και, άρα, όλα τα μονοπάτια που συνδέουν απογόνους της v με προγόνους της αναγκαστικά διέρχονται από την v) είτε η v είναι ρίζα του ΑσΒ με περισσότερα του ενός παιδιά —επομένως, η αποκοπή της v εμποδίζει τους κόμβους ενός υποδένδρου της v να συνδεθούν με τους κόμβους οποιουδήποτε άλλου υποδένδρου της.

Το Σχήμα 3.9 εμφαίνει αυτήν την σχέση: Η κορυφή 2 αποτελεί σημείο αρθρώσεως για τους κόμβους του αριστερού υποδένδρου της, όχι, όμως, και για την 7 του δεξιού υποδένδρου της, καθώς, για την τελευταία, υπάρχει εναλλακτική διαδρομή μέσω της οπισθοακμής (1,7). Η 2 είναι, επίσης, σημείο αρθρώσεως, ως ρίζα-ενδιάμεση κορυφή για τα δύο υποδένδρα της, όπως και η 3, καθώς αποτελεί ενδιάμεση για την 4. Ο Αλγ. 3.4 αποτελεί μία ενδεικτική υλοποίηση. Βασική προϋπόθεση του η κορυφή v της πρώτης κλήσεως να έχει βαθμό τουλάχιστον δύο· διαφορετικά, η v αναγορεύεται, λανθασμένα, ως σημείο αρθρώσεως.

Κεντρική βοηθητική δομή στον εν λόγω αλγόριθμο αποτελεί ο πίνακας low. Η τιμή $\mathrm{low}[v]$ θα ισούται με την μικρότερη από τις ακόλουθες τιμές: (α) τον αριθμό προδιατάξεως $\mathrm{pre}[v]$ της v και (β) τον μικρότερο αριθμό προδιατάξεως που μπορεί να ανακαλυφθεί από ένα απλό μονοπάτι π, το οποίο αποτελείται από ακμές απογόνων της v ακολουθούμενο *το πολύ* από μία οπισθοακμή. Από τον προηγούμενο ορισμό, καθίσταται φανερό πως μία κορυφή δεν είναι σημείο αρθρώσεως εάν $\mathrm{low}[w] < \mathrm{pre}[v]$, για κάθε παιδί w της v στο δένδρο ΑσΒ, αφού η τελευταία σχέση σημαίνει την ύπαρξη εναλλακτικών μονοπατιών από όλες τις κορυφές-απογόνους της στο δένδρο ΑσΒ, προς τις κορυφές-προγόνους της.

Ο υπολογισμός του $\mathrm{low}[v]$ γίνεται αναδρομικά, εξετάζοντας ξεχωριστά όλους τους γιους της v στο δένδρο ΑσΒ (γραμμή 5), κρατώντας την μικρότερη από τις τιμές (γραμ-

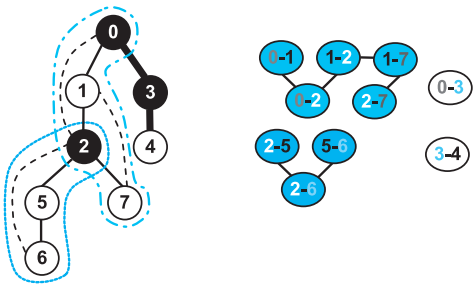
```
Algorithm articPoint(graph g, vertex v)
Input:  Γράφημα g με βοηθητικούς πίνακες pre, low, T
Output: Η λίστα artPoint συμπληρωμένη με τα σημεία αρθρώσεως

1.  g.low[v] = g.pre[v] = g.order++;      // αριθμός προδιατάξεως
2.  for (x = g.List[v]; x != null; x = x.getNext())
3.    if (g.pre[w = x.v] == −1){          // ανεξερεύνητη κορυφή
4.      g.T[w] = v;                       // σημείωση του v ως πατέρα του w
5.      articPoint(g,w);
6.      if (g.low[v] > g.low[w])          // υπάρχει οπισθοακμή προς πρόγονο
7.        g.low[v] = g.low[w];
8.      if (g.low[w] >= g.pre[v])
9.        g.artPoint.insLast(v);          // εύρεση σημείου αρθρώσεως
10.   }
11.   else if ((w != g.T[v]) && (g.low[v] > g.pre[w]))
12.     g.low[v] = g.pre[w];              // η (w, v) είναι οπισθοακμή
```

Αλγόριθμος 3.4: Εύρεση σημείων αρθρώσεως

μές 6–7), και τους αριθμούς προδιατάξεως όλων των κορυφών προσπελάσεων μέσω οπισθοακμών (γραμμές 11–12). Οπότε, εάν μετά την επεξεργασία ενός υποδένδρου, προκύψει ότι όλες οι κορυφές του δεν μπορούν να παρακάμψουν την v μέσω οπισθοακμής (γραμμή 8), τότε η v είναι και σημείο αρθρώσεως.

Επανερχόμενοι στο παράδειγμα του Σχήματος 3.9, ο υπολογισμός της τιμής $\mathrm{low}[0]$, αφού πάρει την αρχική τιμή 0, θα ολοκληρωθεί μετά τον υπολογισμό των $\mathrm{low}[1], \mathrm{low}[3]$, με αυτήν την σειρά. Η $\mathrm{low}[1]$ ξεκινά με την τιμή $\mathrm{pre}[1] = 1$. Εν συνεχεία, θα προκληθεί ο υπολογισμός της $\mathrm{low}[2]$. Η τελευταία αρχικοποιείται στην τιμή 2. Κατόπιν, γίνεται 0, λόγω της εξέτασης της οπισθοακμής (2,0) που οδηγεί στον κόμβο 0 με αριθμό προδιατάξεως 0. Εν συνεχεία, υπολογίζεται αναδρομικά η $\mathrm{low}[5]$, η οποία, από 3 γίνεται 2, καθώς η απόγονός της 6, μέσω της οπισθοακμής (6,2), συνδέεται με την 2 που έχει αριθμό προδιατάξεως $\mathrm{pre}[2] = 2$, ενώ η 5 δεν είναι σημείο αρθρώσεως, αφού $\mathrm{pre}[5] = 3 > \mathrm{low}[6] = 2$. Καθώς $\mathrm{low}[5] = 2 < \mathrm{low}[2] = 0$, δεν σημειώνεται καμμία μεταβολή στην $\mathrm{low}[2]$, μετά την ολοκλήρωση των υπολογισμών για την 5, αλλά, η 2 αναγορεύεται ως σημείο αρθρώσεως για το αριστερό της υποδένδρο, εφ' όσον $\mathrm{low}[5] = 2 \geq \mathrm{pre}[2] = 2$. Ακολούθως, η αναδρομική διαδικασία μεταβαίνει στην κορυφή 7 κ.ο.κ.

Η χρονική πολυπλοκότητα του αλγορίθμου είναι γραμμική, αφού, ουσιαστικά, αποτελεί παραλλαγή της ΑσΒ. Ως τελευταία σημείωση, αναφέρεται ότι, αρκετά εύκολα, ο Αλγ. 3.4 μπορεί να τροποποιηθεί, ώστε να ανακαλύπτει και τις γέφυρες. Σκεφτείτε πώς!



Σχήμα 3.10: Εντοπισμός δισυνεκτικών συνιστωσών μέσω κλάσεων ισοδυναμίας: Η αρχική ΑσΒ (αριστερά) και το βοηθητικό γράφημα B_G (δεξιά).

Λήμμα 3.2 *Ο εντοπισμός των σημείων αρθρώσεως και των γεφυρών ενός γραφήματος $G = (V, E)$ απαιτεί γραμμικό, στο μέγεθός του, χρόνο.*

Δισυνεκτικές Συνιστώσες: Η πρώτη αντιμετώπιση βασίζεται στην ακόλουθη σχέση ισοδυναμίας (Παράρτημα Α.2) L επί του συνόλου E των ακμών ενός γραφήματος $G = (V, E)$:

$$eLo,\ e, o \in E \quad \text{ααν} \quad e = o \text{ ή οι } e, o \text{ ανήκουν σε κάποιον απλό κύκλο του } G.$$

Η απόδειξη ότι η L είναι σχέση ισοδυναμίας είναι απλή. Το μεγαλύτερο ενδιαφέρον είναι ότι οι κλάσεις ισοδυναμίας της ταυτίζονται με τις δισυνεκτικές συνιστώσες τού G. Η τελευταία παρατήρηση προκύπτει άμεσα, από τον ορισμό μιας δισυνεκτικής συνιστώσας ως το μέγιστο υπογράφημα G' του G, στο οποίο, κάθε ζεύγος κορυφών v, w, διασυνδέονται με δύο, διαζευγμένα ως προς τις κορυφές και τις ακμές, μονοπάτια και, συνεπώς, υπάρχει απλός κύκλος που τα περιλαμβάνει. Επομένως, μία κορυφή $v \in V$ είναι σημείο αρθρώσεως αν και μόνον αν έχει προσπίπτουσες ακμές που ανήκουν σε δύο ή περισσότερες κλάσεις ισοδυναμίας, ενώ μία ακμή $e \in E$ είναι γέφυρα εάν και μόνον αν είναι το μόνο μέλος μιας κλάσεως ισοδυναμίας· ήτοι, $[e] = \{e\}$.

Κατά συνέπεια, εάν υπολογισθούν οι κλάσεις ισοδυναμίας, βάσει των παραπάνω παρατηρήσεων, είναι δυνατή η πλήρης κατηγοριοποίηση των ακμών και των κορυφών του γραφήματος. Ο εντοπισμός τους απαιτεί δύο ΑσΒ και ένα βοηθητικό γράφημα B_G (Σχήμα 3.10). Το μεν σύνολο κορυφών του B είναι σε 'ένα-προς-ένα' αντιστοιχία με το σύνολο των ακμών του G. Το δε σύνολο ακμών σχηματίζεται με την βοήθεια μίας ΑσΒ: Για κάθε οπισθοακμή b που ανακαλύπτεται, έστω $e_1, e_2, \ldots, e_j$ οι ακμές δένδρου του αντίστοιχου κύκλου. Προσθέτουμε στο B_G τις ακμές $(b, e_1), (b, e_2), \ldots, (b, e_j)$. Η ανακάλυψη της $(2, 0)$ στο παράδειγμά μας, έχει, ως συνέπεια, την διασύνδεση των κορυφών 0-1 και 1-2 του B_G με την 0-2, η εύρεση της $(6, 2)$ την ένωση των 5-6 και 2-5 με την 2-6, ενώ, όταν συναντούμε την $(7, 1)$, συνδέουμε την 1-7 με τις 1-2 και 1-7.

Το σύνολο των συνεκτικών συνιστωσών του B_G βρίσκεται σε 'ένα-προς-ένα' αντιστοιχία με το σύνολο των δισυνεκτικών συνιστωσών του G: Οι ακμές ενός κύκλου

c του G συνδέονται μεταξύ τους στο B_G, διαμέσου της κορυφής της οπισθοακμής, σχηματίζοντας, έτσι, ένα υπογράφημα «αστέρα» (star ή hub) s_c. Επιπλέον, οι κύκλοι που ανήκουν στην ίδια δισυνεκτική συνιστώσα, μοιράζονται, στο G, ακμές από ένα μονοπάτι, αποτελούμενο από ακμές δένδρου ΑσΒ. Αυτές οι κοινές ακμές έχουν, ως συνέπεια, την συνένωση των αντίστοιχων «αστέρων» μεταξύ τους. Ως εκ τούτου, μία ΑσΒ στο B_G δύναται να εντοπίσει τις δισυνεκτικές συνιστώσες του G. Συνοψίζοντας, έχουμε

Λήμμα 3.3 *Δοθέντος ενός μη κατευθυνόμενου γραφήματος $G = (V, E)$, το πρόβλημα του υπολογισμού των δισυνεκτικών συνιστωσών του είναι γραμμικό στο μέγεθος της εισόδου.*

Η πολυπλοκότητα της παραπάνω λύσεως εξαρτάται από το μέγεθος του παραγόμενου B_G. Το τελευταίο φράσσεται από το $O(nm)$, καθώς ένας κύκλος μπορεί να αποτελείται από $O(n)$ ακμές. Με αποτέλεσμα, η εν λόγω αντιμετώπιση να κοστίζει $O(nm)$. Είναι δυνατόν ο χρόνος να πέσει στο $O(n + m)$, εάν, αντί για το πλήρες B_G, παραχθεί ένα επικαλύπτον δάσος του S_G. Τότε, το μέγεθος του βοηθητικού γραφήματος περιορίζεται στο $O(m)$, και οι δύο ΑσΒ κοστίζουν $O(n+m)+O(m) = O(n+m)$. Η κατασκευή του βελτιωμένου S_G παραλείπεται ως (μία καλή) άσκηση.

Εναλλακτικά, μπορεί κανείς να επεκτείνει τον αλγόριθμο ευρέσεως των σημείων αρθρώσεως (Αλγ. 3.4), ώστε να εντοπίζονται και οι δισυνεκτικές συνιστώσες. Συγκεκριμένα, εάν ο έλεγχος της γραμμής 8 καταδειχθεί αληθινός, τότε η (v, w), μαζί με όλες τις ακμές που προσπελάθηκαν κατά την αναδρομική κλήση της γραμμής 5, συνιστούν μία δισυνεκτική συνιστώσα. Η παρατήρηση αυτή υποδεικνύει την χρήση μίας δομής στοίβας (α) για την εισαγωγή έκαστης ακμής (v, w) είτε προς νεο-ανακαλυφθείσα (ανεξέταστη, δηλαδή) είτε προς «προγονική», αλλά διάφορη της $g.T[v]$ (του πατέρα, δηλαδή, της v), κορυφή w και (β) την ανάκτηση όλων των ακμών-μελών της τρέχουσας δισυνεκτικής συνιστώσας, κάθε φορά που $\mathrm{low}[w] \geq \mathrm{pre}[v]$ —και άρα η v δεν μπορεί να παρακαμφθεί μέσω εναλλακτικού μονοπατιού— (γραμμή 8), με διαδοχικά ποπ από την στοίβα, μέχρις ότου προσπελαστεί η (v, w).

Καθώς κάθε ακμή ανήκει σε μία μόνο δισυνεκτική συνιστώσα, προκύπτει άμεσα πως η παραπάνω τροποποίηση διατηρεί την γραμμικότητα του Αλγ. 3.4. Αξίζει σε αυτό το σημείο να σημειωθεί ότι, εάν το γράφημα περιέχει απομονωμένες κορυφές (βαθμού, δηλαδή, 0), η παραπάνω περιγραφή δεν τις αναγορεύει ως (τετριμμένες) δισυνεκτικές συνιστώσες. Αυτή η «παράληψη» αφήνεται, εν είδει ασκήσεως, στον αναγνώστη. Ως προς την ορθότητά της, είναι δυνατή η τυπική απόδειξη με ισχυρά επαγωγή:

Ως βάση της επαγωγής, χρησιμοποιούμε την ορθότητα του αλγορίθμου στα δισυνεκτικά γραφήματα, καθώς, τότε, το δένδρο ΑσΒ θα διαθέτει ρίζα r με ένα μόνο παιδί και, συνεπώς, η r θα είναι η μόνη κορυφή που ικανοποιεί τον έλεγχο της γραμμής 8. Για την επαγωγική υπόθεση, θεωρούμε ότι ο αλγόριθμος δουλεύει σωστά για όλα τα γραφήματα $G = (V, E)$, διαθέτοντα το πολύ m δισυνεκτικές συνιστώσες. Οπότε,

πρέπει να δείξουμε ότι δουλεύει και για τα γραφήματα $G' = (V', E')$, με $m + 1$ δισυνεκτικές συνιστώσες. Αυτό προκύπτει, εάν παρατηρήσουμε πως, την πρώτη φορά που ο έλεγχος $\mathrm{low[w]} \geq \mathrm{pre[v]}$ αποδειχθεί σωστός, τότε η v είναι σημείο αρθρώσεως και η στοίβα περιέχει όλες τις ακμές, που είναι προσκείμενες σε απογόνους της v, πάνω από την (v, w). Καθώς, δε, καμμία κορυφή-απόγονος της v δεν είναι σημείο αρθρώσεως, έπεται πως όλες αυτές οι ακμές, μαζί με την (v, w), συνιστούν μία δισυνεκτική συνιστώσα. Μετά την ανάκτησή της, με τα διαδοχικά ποπ, το εναπομείναν τμήμα του γραφήματος περιέχει m δισυνεκτικές συνιστώσες, και, βάσει της επαγωγικής υποθέσεως, ο αλγόριθμος θα συμπεριφερθεί ορθά. Κατά συνέπεια:

Λήμμα 3.4 *Η τροποποιημένη εκδοχή του Αλγ. 3.4 υπολογίζει, εντός γραμμικού χρόνου, όλες τις δισυνεκτικές συνιστώσες κάθε γραφήματος $G = (V, E)$.*

3.3.2 Αναζήτηση κατά Πλάτος (ΑκΠ)

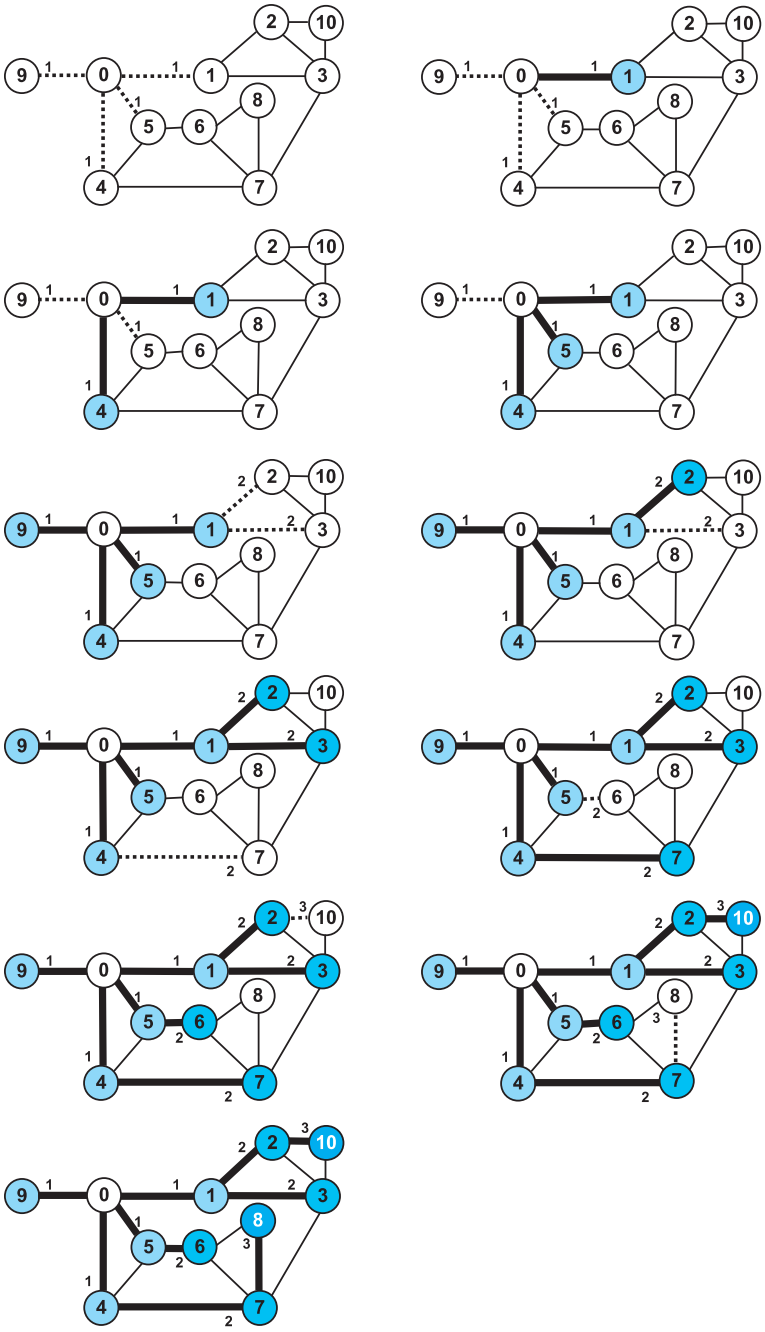
Η ***αναζήτηση κατά πλάτος —ΑκΠ (breadth first search —BFS)*** ενός γραφήματος G πετυχαίνει την εξερεύνηση των κορυφών του γραφήματος κατά αύξοντα μήκη ελαχίστων μονοπατιών από την αρχική κορυφή-αφετηρία. Όπως και στην ΑσΒ, κατά την εκκίνηση της διαδικασίας ΑκΠ, όλες οι κορυφές του εκάστοτε γραφήματος G σημειώνονται ως ανεξερεύνητες (Σχήμα 3.11). Επιλέγουμε μία κορυφή v ως αφετηρία της αναζητήσεως, την οποία και χαρακτηρίζουμε εξερευνημένη. Κατόπιν, μέχρι να επιθεωρηθούν όλες οι κορυφές, συστηματικά εφαρμόζουμε τα ακόλουθα δύο βήματα:

(α) Επισκεπτόμαστε κάθε ανεξέταστη γείτονα κορυφή w της v, σημειώνοντας την ως εξερευνημένη, και

(β) για κάθε κορυφή w που επισκεφτήκαμε στο βήμα (α), φροντίζουμε να εξετάσουμε τις γειτονικές της κορυφές, τηρώντας την διάταξη FiFo.

Ο Αλγόριθμος 3.5 αποτελεί την εκδοχή της ΑκΠ, για αναπαράσταση με πίνακα γειτνίασης. Με ελάχιστες αλλαγές στις γραμμές 5–6, κατά τρόπο παρόμοιο προς τον Αλγ. 3.3, προκύπτει η παραλλαγή για αναπαραστάσεις με λίστες γειτνιάσεως.

Παρατηρήστε πως η ΑκΠ σαρώνει τις κορυφές του G κατά «κύματα» (βήματα): Στο πρώτο κύμα, επισκεπτόμαστε όλες τις κορυφές που συνδέονται άμεσα με την αφετηρία. Κατά το δεύτερο κύμα, εξερευνούμε όσες κορυφές έχουν ελάχιστο μήκος μονοπατιού από την αφετηρία ίσο με δύο, στο τρίτο, αυτές που απέχουν απόσταση ίση με 3 κ.ο.κ. Αυτό συμβαίνει γιατί, κάθε χρονική στιγμή, η δομή της FiFo διαθέτει δύο ειδών κορυφές, οι ελάχιστες αποστάσεις των οποίων διαφέρουν κατά μία μονάδα. Συγκεκριμένα, κατά το i-στό κύμα, η FiFo περιλαμβάνει ένα σύνολο κορυφών, αποστάσεως i από την αφετηρία, ακολουθούμενο από 0 ή περισσότερες κορυφές, αποστάσεως $i + 1$. Μόλις εξαντληθούν οι κορυφές αποστάσεως i, ξεκινά το $(i + 1)$-στό κύμα, κατά το οποίο η ΑκΠ επισκέπτεται όλες όσες απέχουν $i + 1$, ενώ στην ουρά θα εισάγονται,



Σχήμα 3.11: ΑκΠ σε μη κατευθυνόμενο, συνεκτικό γράφημα. Οι χρωματισμοί καταδεικνύουν κόμβους ίδιας αποστάσεως από τον κόμβο εκκινήσεως.

```
Algorithm bfsMatrix(graph g, vertex v)
Input:   Το γράφημα g και μία κορυφή εκκινήσεως v
Output: ΑκΒ

1.  fifo f = new fifo();
2.  f.enqueue(v);
3.  g.pre[v] = order++;
4.  while (!f.isEmpty()){
5.     w = f.dequeue();
6.     for (x = 0; x < g.V; x++)
7.        if ((g.A[w][x] == 1) && (g.pre[x] == −1)){
8.           f.enqueue(x);
9.           g.pre[x] = order++;
10.       }
11. }
```

Αλγόριθμος 3.5: Αναζήτηση σε Πλάτος για αναπαράσταση με πίνακα

εάν υπάρχουν, κορυφές αποστάσεως $i + 2$. Καθίσταται, πλέον, φανερή και η επιλογή της ονομασίας αυτής της διαδικασίας αναζητήσεως· οι κορυφές διαμερίζονται σε εξερευνημένες και ανεξερεύνητες, και σε κάθε βήμα εξετάζονται *όλες* οι ανεξέταστες κορυφές που κείτονται κατά πλάτος του συνόρου.

Βάσει των παραπάνω, οι κορυφές εισάγονται και εξάγονται από την ουρά FiFo κατά σειρά αύξουσας (ελάχιστης) αποστάσεως από την κορυφή αφετηρία. Επομένως, η ΑκΠ εντοπίζει τα συντομότερα (ή, ισοδύναμα, φθηνότερα) μονοπάτια από την κορυφή ενάρξεως της αναζητήσεως προς όλες τις υπόλοιπες, θεωρώντας πως όλες οι ακμές έχουν «βάρος» ή «κόστος» ίσο με την μονάδα. Στο Κεφάλαιο 5 θα αντιμετωπιστεί η γενικότερη περίπτωση συντομότερων μονοπατιών σε βεβαρημένα γραφήματα.

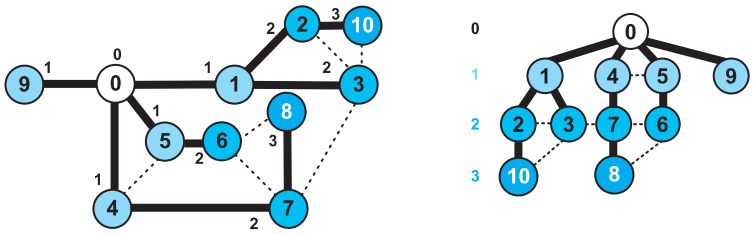
Θεώρημα 3.3 *Η διαπέραση ΑκΠ εξερευνά ένα γράφημα G σε χρόνο γραμμικό ως προς την πολυπλοκότητα του.*

Απόδειξη. Προκύπτει εάν παρατηρήσουμε πως κάθε ακμή ελέγχεται ακριβώς μία φορά, ενώ κάθε κορυφή εισάγεται στην και εξάγεται από την ουρά μόνο μία φορά. ■

Κατά τρόπο ανάλογο προς την ΑσΒ, και η διαδικασία ΑκΠ δύναται να χαρακτηριστεί από ένα δένδρο T_G, γνωστό ως ***δένδρο ΑκΠ (BFS tree)*** (Σχήμα 3.12), το οποίο αποτελείται από τις ακμές του γραφήματος που οδήγησαν σε νεο-ανακαλυφθείσες κορυφές και άρα προκάλεσαν τον υπολογισμό της αποστάσεώς τους.

Βάσει του T_G οι ακμές του αντίστοιχου γραφήματος διακρίνονται σε:

ακμές δένδρου (tree edges), εάν αποτελούν τμήμα του T_G,



Σχήμα 3.12: Κατηγοριοποίηση ακμών (δεξιά) βάσει της εικονιζόμενης ΑκΠ (αριστερά). Οι ακμές δένδρου σημειώνονται παχιές, ενώ οι διασταυρώσεως λεπτές και διακεκομμένες. Όλοι οι κόμβοι επιπέδου i απέχουν από τον κόμβο εκκινήσεως απόσταση i.

ακμές διασταυρώσεως (cross-edges), εάν συνδέουν κορυφές, που οι αριθμοί επιπέδου τους στο T_G διαφέρουν το πολύ κατά ένα, ενώ δεν επιδεικνύουν σχέση προγόνου-απογόνου.

Ο περιορισμός που διέπει τον αριθμό επιπέδου των άκρων των ακμών διασταυρώσεως προέρχεται από το γεγονός ότι κάθε χρονική στιγμή η FiFo περιέχει κορυφές, οι αποστάσεις των οποίων διαφέρουν κατά ένα, το πολύ. Επομένως, ακμή διασταύρωσης (v, w), που διαβαίνει δύο ή περισσότερα επίπεδα είναι αδύνατη, καθώς τούτο θα σήμαινε ότι στο γράφημα υπάρχει συντομότερο μονοπάτι για την w, μέσω της v, και δεν βρέθηκε, παρ' όλη την διαχείριση της FiFo (άτοπο).

Εάν το εν λόγω γράφημα δεν είναι συνεκτικό, τότε η διαδικασία δύναται να αναπαρασταθεί από ένα δάσος δένδρων ΑκΠ. Μέσω ενός βοηθητικού πίνακα $G.T$, με τον ίδιο τρόπο που ακολουθήθηκε και στην περίπτωση της ΑσΒ, δύναται κανείς να κατασκευάσει το ανάστροφο δένδρο T_G^R του T_G. Ως γενικό σχόλιο, πάντως, τα δένδρα ΑκΠ χαρακτηρίζονται από τον μεγάλο βαθμό εξόδου και όχι τόσο για το ύψος τους, καθώς, εν αντιθέσει προς την ΑσΒ, η ΑκΠ δεν «αδημονεί» να ακολουθήσει ανεξερεύνητες ακμές, αλλά διερευνά περιμετρικά στο σύνορο των ανακαλυφθεισών κορυφών.

Εφαρμογές. Η ΑκΠ δύναται να επιλύσει τα ακόλουθα προβλήματα σε μη κατευθυνόμενα γραφήματα:

Εντοπισμός κύκλου: Γίνεται εφ' όσον εντοπιστεί η πρώτη ακμή διασταυρώσεως.

Εύρεση απλού μονοπατιού μεταξύ δύο κορυφών v, w: Αρκεί να εκκινήσουμε από την v μία ΑκΠ. Εάν όντως υπάρχει μονοπάτι, η διαπέραση θα το εντοπίσει.

Επικαλύπτον Δένδρο ή Δάσος, Απλή Συνεκτικότητα: Εάν το γράφημα είναι συνεκτικό, τότε το δένδρο ΑκΠ συνιστά ένα επικαλύπτον δένδρο. Διαφορετικά, οι t

διαφορετικές επανεκκινήσεις της ΑκΠ ανακαλύπτουν τα επικαλύπτοντα δένδρα των t συνεκτικών συνιστωσών.

Συντομότερα Μονοπάτια από μία κορυφή-πηγή v: Η εκτέλεση μίας ΑκΠ σε ένα συνεκτικό γράφημα με κορυφή εκκινήσεως την v ανακαλύπτει τα συντομότερα, ως προς πλήθος ακμών, μονοπάτια προς κάθε άλλη κορυφή.

3.3.3 Γενίκευση

Μία προσεκτικότερη επισκόπηση των αλγορίθμων ΑσΒ και ΑκΠ καταδεικνύει πως η ειδοποιός διαφορά τους έγκειται στον τρόπο επιλογής της επόμενης ανεξερεύνητης κορυφής. Η μεν ΑσΒ, λόγω της αναδρομής, αναφέρεται σε επιλογή κορυφών βάσει μίας στοίβας (ουράς LiFo, δηλαδή), η δε ΑκΠ σε επίσκεψη βάσει μίας ουράς FiFo. Η παρατήρηση αυτή οδηγεί στον ακόλουθο Αλγόριθμο 3.6 «γενικευμένης» διαπεράσεως γραφημάτων, όπου κεντρικό σημείο αποτελεί η επιλογή της ουράς προτεραιότητας: ανάλογα με τον τρόπο που υλοποιείται η ένθεση και η απόσβεση από την ουρά, προκύπτει και η τελική διαπέραση του δένδρου.

Ως γενικό σχόλιο, πάντως, η ουρά προτεραιότητας περιέχει τις κορυφές που βρίσκονται στο σύνορο μεταξύ εξερευνημένων και ανεξερεύνητων κορυφών. Κάθε φορά που αφαιρείται μία κορυφή v από την ουρά, αφαιρείται η v από το σύνορο, επεκτεινόμενο κατά τις προσκείμενες σε αυτήν ανεξευρεύνητες κορυφές.

```
Algorithm pqSearchList(graph g, pqueue pq, vertex v)
Input:  Το γράφημα g, η ουρά προτεραιότητας pq και μία κορυφή εκκινήσεως v
Output: Γενικευμένη αναζήτηση

1.  pq.enqueue(v);            // ένθεση στην ουρά προτεραιότητας pq
2.  g.pre[v] = order++;       // σημείωση του «χρόνου» εξευρευνήσεως
3.  while (!pq.isEmpty()){
4.    w = pq.dequeue();       // επόμενη κορυφή βάσει της πολιτικής της ουράς
5.    g.T[w] = v;             // τοποθέτηση στο «αντίστροφο» δένδρο αναζητήσεως
6.    for (x = g.List[w]; x != null; x = x.getNext())
7.      if (g.pre[x.v] == -1){ // είναι ανεξερεύνητη
8.        pq.enqueue(x);      // ένθεση στην ουρά βάσει της πολιτικής της
9.        g.pre[x.v] = order++;
10.     }
11. }
```

Αλγόριθμος 3.6: Γενικευμένη αναζήτηση για αναπαράσταση με λίστα

3.4 Κατευθυνόμενα Γραφήματα

Τα κατευθυνόμενα γραφήματα, ως συνδυαστικά αντικείμενα, παρουσιάζουν μεγαλύτερη πολυπλοκότητα από τα αντίστοιχα μη κατευθυνόμενα. Για παράδειγμα, η συνεκτικότητα στα πρώτα προϋποθέτει διερεύνηση και προς τις δύο κατευθύνσεις. Στις Παραγράφους που ακολουθούν:

(α) θα εξετάσουμε τις διαπεράσεις στα κατευθυνόμενα γραφήματα,

(β) θα ασχοληθούμε με μία ειδική κατηγορία τους, τα κατευθυνόμενα άκυκλα, και

(γ) θα παρουσιάσουμε αλγορίθμους για ζητήματα διασυνδέσεως.

3.4.1 Διαπεράσεις Κατευθυνόμενων Γραφημάτων

Οι διαφορές που εντοπίζονται στην ΑσΒ και την ΑκΠ κατευθυνόμενων γραφημάτων, προέρχονται από το γεγονός πως, πλέον, τα αντίστοιχά τους δένδρα ΑσΒ και ΑκΠ είναι κατευθυνόμενα.

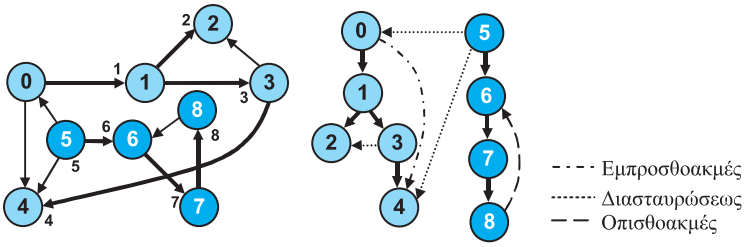
Αναζήτηση σε Βάθος. Δοθέντος ενός δένδρου ΑσΒ T_G, οι ακμές ενός γραφήματος $G = (V, E)$ δύνανται να κατηγοριοποιηθούν σε (συμβουλευτείτε το Σχήμα 3.13):

ακμές δένδρου (tree edges), εάν αποτελούν τμήμα του T_G,

εμπροστο-ακμές (forward-edges), εάν συνδέουν κορυφές που επιδεικνύουν σχέση προγόνου-απογόνου ως προς το T_G,

οπισθοακμές (back-edges), εάν συνδέουν κορυφές που επιδεικνύουν σχέση απογόνου-προγόνου ως προς το T_G, και

ακμές διασταυρώσεως (cross-edges), εάν συνδέουν κορυφές που δεν παρουσιάζουν σχέσεις απογόνου-προγόνου, εν σχέσει με το T_G.



Σχήμα 3.13: ΑσΒ και δένδρο ΑσΒ σε κατευθυνόμενο γράφημα.

Παρατηρήστε πως οι ακμές διασταυρώσεως, εξ ορισμού της ΑσΒ, *πάντοτε* κατευθύνονται από δεξιά προς τα αριστερά. Προκύπτει πολύ εύκολα, πως, μία ακμή (v, w), προς ήδη εξερευνημένη κορυφή w, είναι οπισθοακμή, εάν ο αριθμός μεταδιατάξεως $\mathrm{post}[w]$ της τελευταίας είναι μεγαλύτερος από τον αντίστοιχο $\mathrm{post}[v]$ της v. Σε αντίθετη περίπτωση, πρόκειται για ακμή διασταυρώσεως, εάν ο αριθμός προδιατάξεως $\mathrm{pre}[w]$ είναι μικρότερος του $\mathrm{pre}[v]$ ή για εμπροστο-ακμή, εάν $\mathrm{pre}[w] > \mathrm{pre}[v]$.

Οι παραπάνω παρατηρήσεις μπορούν εύκολα να ενσωματωθούν στην διαδικασία ΑσΒ, ώστε οι ακμές να κατηγοριοποιούνται, μόλις προσπελαστούν. Το πώς αφήνεται, εν είδει ασκήσεως, στον ενδιαφερόμενο αναγνώστη. Ως εφαρμογές της ΑσΒ αναφέρονται τα εξής:

Προσπελασιμότητα από μία κορυφή-πηγή v: Η εκτέλεση μίας ΑσΒ, με κορυφή εκκινήσεως την v, ανακαλύπτει όλες τις κορυφές, προσπελάσιμες μέσω κατευθυνόμενου μονοπατιού, από την v. Η ορθότητα του ισχυρισμού αυτού προκύπτει, πολύ εύκολα, με ισχυρά επαγωγή στο μέγεθος του γραφήματος, εάν παρατηρήσουμε ότι, αφού επιλεγεί η πρώτη ακμή $e \in E$ από την διαδικασία ΑσΒ, το σύνολο των προσπελάσιμων, από την v, κορυφών διαμοιράζεται σε αυτές που ανήκουν σε κύκλους και σε όσες κείτονται επί απλών μονοπατιών.

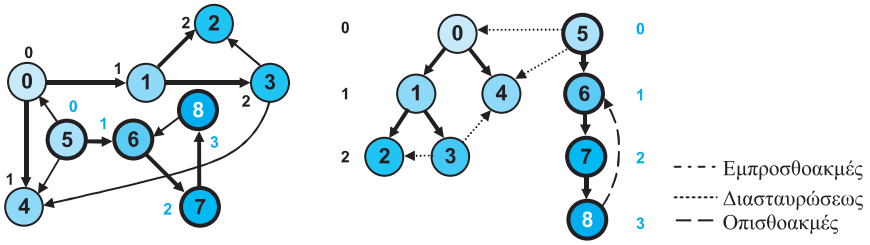
Εντοπισμός κύκλου: Γίνεται εφ' όσον εντοπιστεί μία οπισθοακμή. Για το αληθές αυτού, παρατηρήστε ότι, πρώτον, η ΑσΒ εντοπίζει όλες τις προσπελάσιμες ακμές, και, δεύτερον, η πρώτη κορυφή του κύκλου που εντοπίζει η ΑσΒ έχει τον μικρότερο αριθμό προδιατάξεως και, άρα, η προσκείμενη σε αυτή ακμή, που κλείνει τον κύκλο, είναι οπισθοακμή.

Ιδιότητες διασυνδέσεως: Εν αντιθέσει με τα μη κατευθυνόμενα, όπου κάθε δένδρο ΑσΒ αντιστοιχεί και σε συνεκτική συνιστώσα, στα κατευθυνόμενα γραφήματα, η ύπαρξη ακμών διασταυρώσεως καθιστά δυνατή την μετακίνηση μεταξύ των δένδρων ΑσΒ. Ως εκ τούτου, τα ζητήματα αυτής της κατηγορίας παρουσιάζουν μεγαλύτερη πολυπλοκότητα και, άρα, προσεκτικότερη επεξεργασία. Αυτός είναι ο λόγος που τους αφιερώνεται χωριστή παράγραφος στην συνέχεια του Κεφαλαίου.

Αναζήτηση κατά Πλάτος. Βάσει του αντίστοιχου δένδρου ΑκΠ, οι ακμές διακρίνονται σε (Σχήμα 3.14):

ακμές δένδρου (tree edges), εάν αποτελούν τμήμα του T_G, και, επομένως, αντιστοιχούν σε επιτυχή υπολογισμό,

ακμές διασταυρώσεως (cross-edges), εάν συνδέουν κορυφές που δεν παρουσιάζουν σχέσεις απογόνου-προγόνου ως προς το T_G, και



Σχήμα 3.14: ΑκΠ και δένδρο ΑκΠ σε κατευθυνόμενο γράφημα.

οπισθοακμές (back-edges), εάν συνδέουν κορυφές που επιδεικνύουν σχέση απογόνου-προγόνου ως προς το T_G.

Παρατηρήστε την απουσία εμπροστοακμών (γιατί;) Η κατευθυνόμενη ΑκΠ έχει λιγότερες εφαρμογές από ό,τι η ΑσΒ:

Προσπελασιμότητα-Συντομότερα Κατευθυνόμενα Μονοπάτια από μία κορυφή-πηγή v: Η εκτέλεση μίας ΑκΠ, με κορυφή εκκινήσεως την v, ανακαλύπτει όλες τις κορυφές, προσπελάσιμες μέσω κατευθυνόμενου μονοπατιού, από την v, κατά αύξουσα συντομότερη απόσταση. Η απόδειξη του ισχυρισμού αυτού ακολουθεί την αντίστοιχη των μη κατευθυνόμενων γραφημάτων.

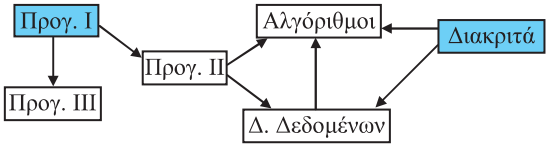
Εντοπισμός κύκλου: Επιτυγχάνεται με τον εντοπισμό της πρώτης οπισθοακμής.

Διάταξη κορυφών ΚΑΓ: Στην Παράγραφο που ακολουθεί θα περιγραφεί ένα είδος ΑκΠ που ταξινομεί, κατά τοπολογική διάταξη, τις κορυφές ενός ΚΑΓ.

3.4.2 Κατευθυνόμενα Άκυκλα Γραφήματα

Τα ***κατευθυνόμενα άκυκλα γραφήματα —ΚΑΓ (directed acyclic graphs —DAG)*** αποτελούν μία ιδιαίτερη χρήσιμη υποκατηγορία κατευθυνόμενων γραφημάτων, τα οποία μπορούν να θεωρηθούν και ως ένα είδος συμπαγών κατευθυνόμενων δένδρων. Κλασσικό παράδειγμα εμφανίσεως των ΚΑΓ αποτελεί ο προγραμματισμός εργασιών ενός σύνθετου έργου, τα οποία χαρακτηρίζονται από χρονικούς περιορισμούς ως προς την εκκίνηση και την περάτωσή τους: κάθε εργασία αντιστοιχείται με μία κορυφή, ενώ ο περιορισμός «*η επιτυχής ολοκλήρωση της εργασίας x επιτρέπει την εκκίνηση της εργασίας y*», δηλώνεται με τον σχεδιασμό μίας κατευθυνόμενης ακμής, από την κορυφή x στην κορυφή y.

Ως άλλο παράδειγμα, θα μπορούσε να αναφερθεί η αποτύπωση των περιορισμών που θέτουν τα προαπαιτούμενα ενός προγράμματος σπουδών. Το Σχήμα 3.15 αποτελεί ένα στιγμιότυπο τέτοιου προβλήματος, όπου μία ακμή από ένα μάθημα x σε άλλο



Σχήμα 3.15: Στιγμιότυπο ΚΑΓ. Με σκούρο χρώμα εμφαίνονται οι δύο πηγές του.

μάθημα y ερμηνεύεται ως «*η επιτυχής εξέταση στο* x *επιτρέπει την δήλωση του* y». Εάν, μάλιστα, φανταστούμε πως οι κόμβοι 'Αλγόριθμοι' και 'Δ. Δεδομένων' σπάνε στα τρία, τότε το προκύπτον γράφημα συνιστά ένα δάσος από κατευθυνόμενα δένδρα με επαναλαμβανόμενους κόμβους. Αυτό δικαιολογεί και τον ανωτέρω χαρακτηρισμό τους ως συμπαγές μορφές κατευθυνόμενων δένδρων.

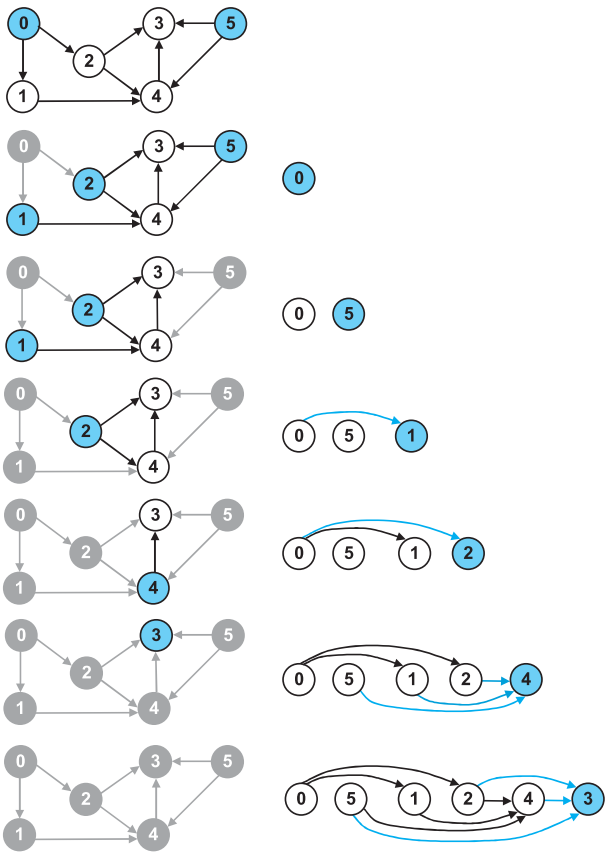
Τοπολογική Διάταξη

Συναφή ως τα ΚΑΓ αποτελεί η έννοια της ***τοπολογικής διατάξεως (topological sort)*** των κορυφών του: αιτείται η διάταξη των κορυφών, κατά τέτοιο τρόπο, ώστε κάθε κορυφή v να προηγείται όλων των κορυφών w με $(v, w) \in E$. Επανερχόμενοι στην περίπτωση των υποέργων, η δρομολόγησή τους βάσει της τοπολογικής διατάξεως εξασφαλίζει την ορθή εκτέλεση του έργου, καθώς ο αριθμός που λαμβάνει εκάστη εργασία είναι μεγαλύτερος από τους αντίστοιχους όλων των εργασιών, από την ολοκλήρωση των οποίων εξαρτάται. Αξίζει να αναφερθεί ότι είναι δυνατόν να υφίστανται περισσότερες της μίας τοπολογικές διατάξεις για ένα ΚΑΓ.

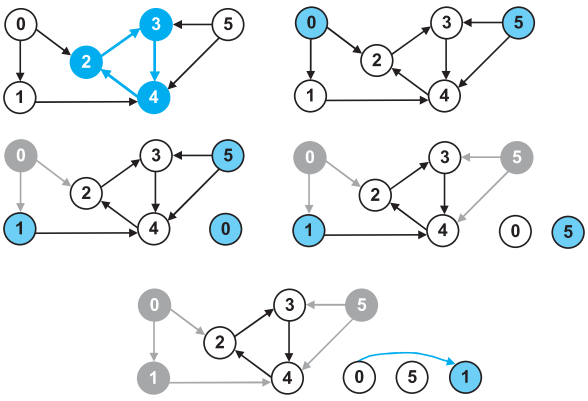
Στην συνέχεια, παρουσιάζονται δύο διαφορετικοί τρόποι ευρέσεως μίας τοπολογικής διατάξεως. Η μία βασίζεται στην παρουσία των πηγών και των καταβοθρών, και η δεύτερη, στην πληροφορία που παρέχει η σειρά μεταδιατάξεως κατά την εκτέλεση ΑσΒ.

Διαδοχικές Αφαιρέσεις Κορυφών. Η εν λόγω μέθοδος προκύπτει από την παρατήρηση ότι υπάρχει τοπολογική διάταξη εάν και μόνον αν δεν υπάρχουν κύκλοι στο γράφημα. Το ευθές της πρότασης αυτής είναι συνέπεια του ορισμού. Το ανάποδο, βασίζεται στην ύπαρξη πηγών και καταβοθρών στο ΚΑΓ, εξ αιτίας της απουσίας κύκλων: εάν δεν υπήρχαν, τότε, θα ήταν δυνατή η έξοδος από κάθε κορυφή, και, άρα, ξεκινώντας από οποιαδήποτε κορυφή θα καταλήγαμε ξανά σε αυτήν (άτοπο). Κατά συνέπεια, εάν το γράφημα είναι άκυκλο, υπάρχει μία πηγή, της αφαίρεση της οποίας δημιουργεί επίσης άκυκλο υπογράφημα.

Η τελευταία παρατήρηση συνιστά την ακόλουθη διαδικασία: Αρχικά, επιλέγουμε μία οποιαδήποτε πηγή s, αφαιρούμε τις (εξερχόμενες από αυτήν) ακμές της, αναφέρουμε την s ως επόμενη, στην διάταξη, κορυφή και εφαρμόζουμε αναδρομικά την διαδικασία στο προκύπτον γράφημα, έως ότου εξαντληθούν οι κορυφές. Εάν κάποιες



Σχήμα 3.16: Στιγμιότυπο τοπολογικής διατάξεως κορυφών ΚΑΓ με διαδοχικές διαγραφές κορυφών. Με χρωματισμένο κόμβο, σημειώνονται οι εκάστοτε πηγές του ΚΑΓ.



Σχήμα 3.17: Εντοπισμός κύκλου μέσω αφαιρέσεως κορυφών.

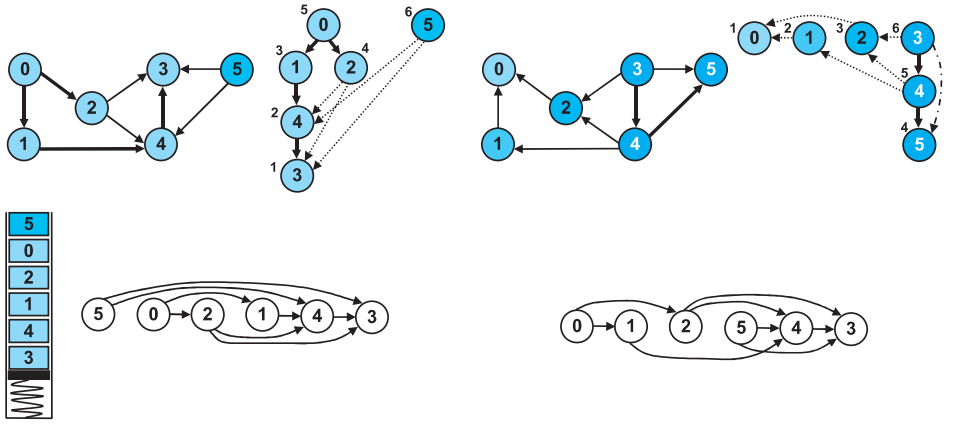
```
Algorithm topolSortList(graph g, int[] sort)
Input:  Το ΚΑΓ g και ο πίνακας διατάξεως sort
Output: Η τοπολογική διάταξη sort

1.  int[] aux;   // βοηθητικός πίνακας με τους βαθμούς εισόδου των κορυφών
2.  for (v = 0; v < g.V; v++) { // αρχικοποίηση των πινάκων
3.      aux[v] = 0;     // ο aux χρησιμοποιείται για τον εντοπισμό πηγών
4.      sort[v] = −1;
5.  }
6.  for (v = 0; v < g.V; v++)
7.    for (x = g.List[v]; x != null; x = x.getNext())
8.      aux[x.v]++;   // βρέθηκε μία ακόμη προσπίπτουσα ακμή προς την x.v
9.  fifo pq = new fifo();   // ουρά με τις πηγές
10. for (v = 0; v < g.V; v++)
11.   if (aux[v] == 0)
12.     pq.enqueue(v); // η v είναι πηγή
13. for (i = 0; !pq.isEmpty(); i++){
14.   sort[i] = (v = pq.dequeue());  // η v αποτελεί την i-στή κορυφή
15.     for (x = g.List[v]; x != null; x = x.getNext())
16.       if (−−aux[x.v] == 0) // αφαίρεση της (v,w) από τις προσπίπτουσες
17.         pq.enqueue(x.v);     // και τυχόν προσθήκη της w στο σύνολο των πηγών
18. }
```

Αλγόριθμος 3.7: Τοπολογική διάταξη με διαδοχικές αφαιρέσεις πηγών

κορυφές δεν εξερευνηθούν από την διαδικασία, τότε το γράφημα, όπως μαρτυρά και η παραπάνω πρόταση, *δεν* είναι ΚΑΓ, αλλά υπάρχει ένα υπογράφημα επί αλληλοεξαρτώμενων κορυφών.

Ο Αλγ. 3.7 αποτελεί μία άμεση υλοποίηση των ανωτέρω για αναπαράσταση με λίστες γειτνίασης. Παρατηρήστε πως με την χρήση του βοηθητικού πίνακα aux, για την δυναμική καταγραφή του βαθμού εισόδου των κορυφών, σε συνδυασμό με την δομή ουράς FiFo, είναι δυνατή η διάταξη των κορυφών. Οποιαδήποτε άλλη ουρά σταθερού χρόνου ενθέσεως και αποσβέσεως, όπως, λόγου χάριν, στοίβα, θα εξυπηρετούσε το ίδιο. Στιγμιότυπο τρεξίματος του αλγορίθμου εικονίζεται στο Σχήμα 3.16: Αρχικά, υπάρχουν δύο πηγές, οι 0 και 5. Από αυτές, επιλέγεται για αφαίρεση η 0, γεγονός που προκαλεί τον μηδενισμό του βαθμού εισόδου των 1 και 2. Κατόπιν, εξάγεται από την ουρά η 5 κ.ο.κ. Στο Σχήμα 3.17 απεικονίζεται η συμπεριφορά του αλγορίθμου, εάν στο προηγούμενο παράδειγμα αντιστραφεί η φορά της ακμής (3,4), ώστε να δημιουργεί ο κύκλος $(2, 3, 4)$.

Χρήση ΑσΒ. Η δεύτερη κατηγορία αλγορίθμων βασίζεται στο γεγονός ότι η σειρά με την οποία εγκαταλείπει οριστικά τις κορυφές μία ΑσΒ (η μεταδιάταξη κατά ΑσΒ,



(α) (β)

Σχήμα 3.18: Τοπολογική διάταξη (α) με χρήση στοίβας, (β) μέσω G^R. Παραπλεύρως των κορυφών, εικονίζονται οι αριθμοί μεταδιατάξεως.

δηλαδή, των κορυφών) αποτελεί, στην πραγματικότητα, την αντίστροφη τοπολογική διάταξη: εάν $v \leqslant w$, ως προς την μεταδιάταξη, τότε δεν είναι δυνατόν να υπάρξει ακμή (v, w), λόγω της απουσίας κύκλων στο γράφημα. Οπότε, μπορεί κανείς είτε να αποθηκεύει τις κορυφές σε μία στοίβα, όταν περατώνεται οριστικά η επεξεργασία τους, και μετά τον τερματισμό της ΑσΒ, να τις αναφέρει με διαδοχικά ποπ, (Σχήμα 3.18(α)) είτε να εφαρμόσει ΑσΒ για την εύρεση της μεταδιάταξεως των κορυφών στο ανάστροφο γράφημα G^R (Σχήμα 3.18(β)) είτε να εφαρμόσει ΑσΒ και, κατά σειρά μεταδιατάξεως, να αποδίδονται στις αντίστοιχες κορυφές, κατά φθίνουσα τιμή, οι αριθμοί $V - 1$ ως 0 (Αλγ. 3.8).

3.4.3 Εξαγωγή Ιδιοτήτων Διασύνδεσης

Μεταβατική Κλειστότητα

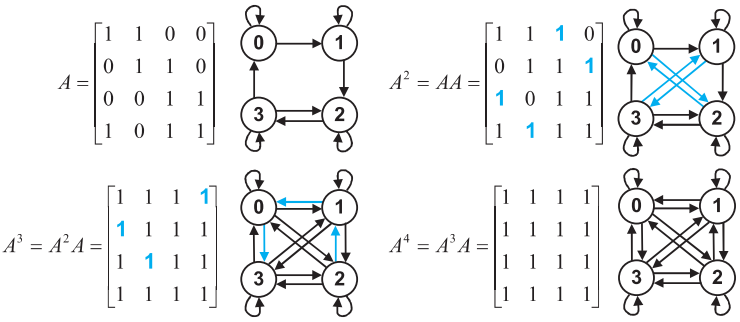
Η εύρεση της μεταβατικής κλειστότητας ενός γραφήματος αποτελεί ένα από τα πρώτα προβλήματα στα γραφήματα που επιλύθηκαν, καθώς, έτσι, αποκαλύπτονται όλες οι δυνατές διασυνδέσεις, άμεσες ή έμμεσες μέσω μονοπατιών, μεταξύ των κορυφών ενός γραφήματος. Η απλούστερη λύση συνίσταται από την εκτέλεση V ΑσΒ, μία για κάθε κορυφή v, ώστε να ανακαλυφθούν οι προσπελάσιμες από αυτήν κορυφές. Το κόστος είναι $O(V(V + E))$, στην χειρότερη περίπτωση, και η διαδικασία περιγράφεται από τον Αλγ. 3.9 για αναπαράσταση με λίστα γειτνιάσεως. Ο αλγόριθμος δύναται να τροποποιηθεί, με την βοήθεια δυναμικού προγραμματισμού, στην περίπτωση των ΚΑΓ, ώστε να εξοικονομηθεί χρόνος. Ο ενδιαφερόμενος αναγνώστης παραπέμπεται στην σχετική Άσκηση, στο τέλος του Κεφαλαίου.

```
Algorithm revTopolSort(graph g, int[] post, boolean[] found)
Input:  Γράφημα g, πίνακες μεταδιατάξεως post και επισκέψεως found
Output: Η αντίστροφη τοπολογική διάταξη post

1.  g.order = V−1; // βοηθητική μεταβλητή
2.  for (v = 0; v < g.V; v++){   // αρχικοποίηση
3.    post[v] = −1;
4.    found[v] = false;
5.  }
6.  for (v = 0; v < g.V; v++)
7.    if (!found(v))
8.      dfsListPost(g,v,post,found);

9.  static void dfsListPost(graph g, vertex v, int[] post, boolean[] found){
10.   found[v] = true;        // σημείωση της επισκέψεως
11.   for (x = g.List[v]; x != null; x = x.getNext())
12.     if (!found[x.v])      // επίσκεψη παιδιού
13.       dfsListPost(g,x.v,post,found);
14.   post[g.order−−] = v; // αριθμός μεταδιατάξεως
15. }
```

Αλγόριθμος 3.8: Τοπολογική διάταξη με ΑσΒ και αναπαράσταση λίστας

```
Algorithm transClosure(graph g, graph t)
Input:  Γράφημα εισόδου g και γράφημα t με πίνακα TrClosure για την κλειστότητα
Output: Το γράφημα της κλειστότητας t με τον TrClosure συμπληρωμένο

1.  for (v = 0; v < g.V; v++)  // ο TrClosure αρχικά είναι μηδενικός
2.    dfsListTC(g,t,v,v);

3.  static void dfsListTC(graph g, graph t, vertex w, vertex v){
4.    t.List[w].insertLast(v);
5.    for (x = g.List[v]; x != null; x = x.getNext())
6.      if (t.TrClosure[w][x.v] == 0){
7.        t.TrClosure[w][x.v] = 1;
8.        dfsListTC(g,t,w,x.v);
9.      }
10. }
```

Αλγόριθμος 3.9: Μεταβατική κλειστότητα με ΑσΒ σε αναπαράσταση λίστας



Σχήμα 3.19: Μεταβατική κλειστότητα μέσω πολλαπλασιασμού πινάκων.

Η ανωτέρω λύση είναι πολύ καλή για αραιά γραφήματα, καθώς η πολυπλοκότητά της είναι ανάλογη του V^2. Στα πυκνά, όμως, αυξάνει σε $O(V^3)$. Για την τελευταία κατηγορία γραφημάτων εφαρμόζεται, κατά κύριο λόγο, εναλλακτική μεθοδολογία, η οποία, βασιζόμενη στην ομοιότητα που παρουσιάζει με τον πολλαπλασιασμό λογικών πινάκων, επιτυγχάνει —στην πράξη— καλύτερους χρόνους. Έστω, λοιπόν, κατευθυνόμενο γράφημα $G = (V, E)$ με πίνακα γειτνιάσεως A. Θεωρούμε, στην συνέχεια, τον πίνακα $A \otimes A \equiv A^2$, όπου το $\otimes$ δηλοί τον (λογικό) πολλαπλασιασμό πινάκων Boole. Με άλλα λόγια, αντί της πράξεως του «κανονικού» πολλαπλασιασμού, εκτελείται το λογικό 'και' ($\wedge$ ή $\cdot$), και στην θέση της προσθέσεως, εφαρμόζεται η πράξη του 'ή' ($\vee$ ή $+$). Παρατηρήστε ότι ο A^2 διαθέτει 1 στην θέση $A^2[i][j]$, εάν υπάρχει τουλάχιστον μία κορυφή $k \neq i, j$, τέτοια ώστε $A[i][k] \wedge A[k][j] = 1$. Τουτ' έστιν, εάν υπάρχει κατευθυνόμενο μονοπάτι (i, k, j), μήκους 2, που συνδέει την i με την j, όπου η k είναι ενδιάμεση κορυφή. Συνεχίζοντας, κατά την ίδια λογική, ο πίνακας $A^2 \otimes A \equiv A^3$, φέρει 1 στις θέσεις $A^3[i][j]$, εάν υπάρχει κατευθυνόμενο μονοπάτι από την i στην j, μήκους 3, και γενικά, ο πίνακας A^m, φέρει 1 στις θέσεις $A^m[i][j]$, εάν υπάρχει κατευθυνόμενο μονοπάτι, μήκους m, από την i στην j.

Ως άμεση επέκταση των ανωτέρω, θεωρήστε ότι θέτουμε 1 σε όλες τις θέσεις της κύριας διαγωνίου του πίνακα A. Αυτό ισοδυναμεί με την προσθήκη βρόχων σε όλες τις κορυφές του αντίστοιχου γραφήματος G. Κατόπιν, δεν είναι καθόλου δύσκολο να δει κανείς ότι ο πίνακας A^2 έχει 1 σε μία θέση $A^2[i][j]$, εάν υπάρχει κατευθυνόμενο μονοπάτι, από την i στην j, μήκους *το πολύ* 2, ο A^3 έχει θεμένη στην τιμή 1 το $A^3[i][j]$, εάν η i συνδέεται με την j, με κατευθυνόμενο μονοπάτι μήκους *το πολύ* 3 κ.ο.κ. Αμέσως, προκύπτει το ερώτημα εάν υπάρχει κάποιο τέλος στην διαδικασία. Η απάντηση είναι καταφατική, καθώς δεν χρειάζεται να υπολογιστεί μεγαλύτερη του V δύναμη του (τροποποιημένου) πίνακα A· εφ' όσον ο A^i ανακαλύπτει μονοπάτια μήκους μικρότερου ή ίσου του i, και άρα, το πολύ $i + 1$ διακριτών κορυφών, απλό επιχείρημα τύπου ***αρχής περιστερώνα (pigeonhole principle)*** αποδεικνύει ότι τα μονοπάτια μήκους μεγαλύτερου του V δεν είναι απλά, αλλά περιλαμβάνουν, αναγκαστικά, και κύ-

κλους. Με συνέπεια, ο A^V να αντιστοιχεί και στην μεταβατική κλειστότητα. Το Σχήμα 3.19 επεξηγεί την όλη διαδικασία για γράφημα τεσσάρων κορυφών. Χρωματισμένες παρουσιάζονται οι αλλαγές τιμών και οι νέες ακμές.

Η πολυπλοκότητα της περιγραφείσας διαδικασίας είναι $O(V^4)$, εφ' όσον περιλαμβάνει V πολλαπλασιασμούς πινάκων, ενώ ένας πολλαπλασιασμός, μεταξύ πινάκων $V \times V$, κοστίζει $O(V^3)$ πράξεις. Η πολυπλοκότητα αυτή δύναται να βελτιωθεί σε $O(V^3 \log V)$, εάν κανείς υπολογίσει την εξής ακολουθία πινάκων, μήκους $O(\log V)$:

$$A^2, A^4 = A^2 \otimes A^2, A^8 = A^4 \otimes A^4, \ldots, A^k = A^{2^{j-1}} \otimes A^{2^{j-1}},$$

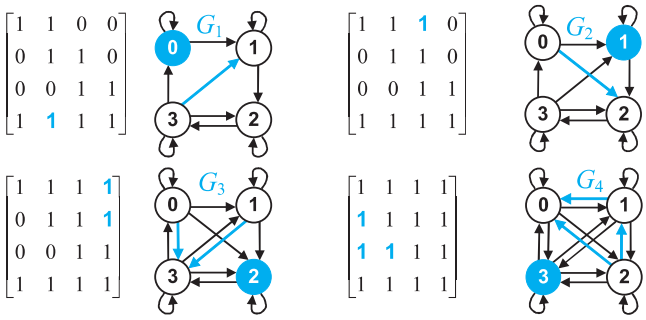
όπου $k = 2^j$ η μικρότερη δύναμη του δύο, ίση ή μεγαλύτερη του V. Έτσι, εξοικονομείται ένα ποσοστό $O(V/\log V)$ σε πράξεις.

Υπάρχει καλύτερη λύση από τις προαναφερόμενες; Η απάντηση είναι καταφατική: μάλιστα, εφαρμόζει την τεχνική του δυναμικού προγραμματισμού (§ 2.2), αποτελεί τον αλγόριθμο επιλογής για πυκνά γραφήματα και είναι γνωστός ως ***αλγόριθμος του Warshall***, από το όνομα του επιστήμονα που τον ανακάλυψε. Σύμφωνα με αυτόν, αφού δοθεί μία τυχαία διάταξη $v_1, v_2, \ldots, v_V$ στις κορυφές, κατασκευάζεται μία ακολουθία γραφημάτων $G_1, G_2, \ldots, G_V$, τέτοιων ώστε το G_i, $1 \leq i \leq V$ να περιέχει, *πρώτον*, όλες τις ακμές του G_{i-1} και, *δεύτερον*, κάθε ακμή της μορφής (v_k, v_j), εφ' όσον οι ακμές (v_k, v_i) και (v_i, v_j) ανήκουν αμφότερες στο G_{i-1}. Το τελευταίο, στην σειρά, γράφημα G_V αποτελεί την μεταβατική κλειστότητα G^* του G, αφού, εκ κατασκευής, θα περιέχει κατευθυνόμενες ακμές μεταξύ όλων των (διατεταγμένων) ζευγών κορυφών που διασυνδέονται με κάποιο κατευθυνόμενο μονοπάτι.

Ο Αλγ. 3.10 συνιστά άμεση υλοποίηση, πολυπλοκότητας $O(V^3)$, της παραπάνω

```
Algorithm warshallTC(graph g)
Input:  Γράφημα g με πίνακα κλειστότητας TrClosure
Output: Συμπληρωμένος TrClosure

1.  for (v = 0; v < g.V; v++)          // αρχικοποίηση του πίνακα κλειστότητας
2.    for (w = 0; w < g.V; w++)
3.      g.TrClosure[v][w] = g.A[v][w];
4.  for (v = 0; v < g.V; v++)
5.    g.TrClosure[v][v] = 1;
6.  for (i = 0; i < g.V; i++)          // υπολογισμός του i-στού γραφήματος
7.    for (v = 0; v < g.V; v++)        // υπολογισμός μεταβάσεως μέσω
8.      if (g.TrClosure[v][i] == 1)    // της κορυφής i
9.        for (w = 0; w < g.V; w++)
10.         if (g.TrClosure[i][w] == 1) g.TrClosure[v][w] = 1;
```

Αλγόριθμος 3.10: Μεταβατική κλειστότητα κατά Warshall



Σχήμα 3.20: Στιγμιότυπο μεταβατικής κλειστότητας κατά Warshall.

περιγραφής, αναπαριστώντας τα γραφήματα G_i με τον πίνακα TrClosure· μετά την αρχικοποίηση του πίνακα (γραμμές 1–5), σαρώνονται, κατά αύξοντα αριθμό, οι κορυφές και ελέγχεται μήπως υπάρχει μονοπάτι, μέσω της τρέχουσας κορυφής i. Παρατηρήστε την γραμμή 8, στην οποία εξετάζεται η ύπαρξη του δεύτερου μονοπατιού από την i στην w μόνο εάν υπάρχει μονοπάτι από την v στην i, εξοικονομώντας, έτσι, μία ανώφελη πράξη.

Λήμμα 3.5 *Ο αλγόριθμος του Warshall υπολογίζει την μεταβατική κλειστότητα κάθε γραφήματος $G = (V, E)$, εντός χρόνου $O(V^3)$.*

Απόδειξη. Η ορθότητα του αλγορίθμου προκύπτει, πολύ εύκολα, με επαγωγή στον αριθμό i της κορυφής που χρησιμοποιείται για τον σχηματισμό του G_i: Πριν το i-στό βήμα, η θέση TrClosure$[k][j]$ του πίνακα είναι θεμένη στην τιμή 1, αν υπάρχει μονοπάτι μεταξύ των κορυφών v_k, v_j, αποτελούμενο *αποκλειστικά* από κορυφές του συνόλου $\{v_1, v_2, \ldots, v_{i-1}\}$, ενώ, κατά το i-στό βήμα, μία θέση του TrClosure$[l][m]$ αποκτά την τιμή 1 μόνον εάν $(v_l, v_i), (v_i, v_m) \in E$ και, κατά συνέπεια, υπάρχει μονοπάτι μεταξύ των v_l, v_m, αποτελούμενο *αποκλειστικά* από κορυφές του συνόλου $\{v_1, v_2, \ldots, v_{i-1}, v_i\}$. Οπότε, μετά το V-στό βήμα, ο εν λόγω πίνακας αποτελεί και την μεταβατική κλειστότητα του G, καθώς περιλαμβάνει όλα τα κατευθυνόμενα μονοπάτια που σχηματίζουν οι κορυφές $\{v_1, v_2, \ldots, v_{V-1}, v_V\} = V$.

Η χρονική πολυπλοκότητα προκύπτει με επισκόπηση του Αλγ. 3.10. ■

Το Σχήμα 3.20 εικονίζει ένα στιγμιότυπο υπολογισμού μεταβατικής κλειστότητας. Αρχικά, στο πρώτο βήμα, ανακαλύπτεται το μοναδικό μονοπάτι, μήκους δύο, με ενδιάμεση κορυφή την 0. Κατόπιν, υπολογίζονται τα μονοπάτια που έχουν, ως ενδιάμεσες, είτε την 0 είτε την 1 είτε τις 0 και 1· υπάρχει μόνο ένα καινούργιο, το $(0, 1, 2)$. Έπειτα, προσθέτοντας την κορυφή 2, ανακαλύπτονται τα μονοπάτια $(0, 1, 2, 3)$ και $(1, 2, 3)$, μήκους τρία και δύο, αντίστοιχα. Τέλος, μετά και την θεώρηση της 3, κατασκευάζονται τα μονοπάτια $(2, 3, 0), (2, 3, 0, 1)$ και $(1, 2, 3, 0)$.

Κλείνοντας, αξίζει να αναφερθεί ότι η σχέση μεταξύ των προβλημάτων του πολλαπλασιασμού δύο λογικών πινάκων A, B, διαστάσεων $n \times n$, και του υπολογισμού της μεταβατικής κλειστότητας ενός γραφήματος G, με πίνακα γειτνιάσεως C, είναι πολύ στενή, καθώς το πρώτο πρόβλημα δύναται να αναχθεί στο δεύτερο, μέσω ενός έξυπνου μετασχηματισμού: Δοθέντων των A, B κατασκευάζουμε το γράφημα G με πίνακα γειτνιάσεως:

$$C = \begin{bmatrix} I & A & 0 \\ \mathbf{0} & I & B \\ \mathbf{0} & \mathbf{0} & I \end{bmatrix},$$

όπου $\mathbf{0}$ είναι ο μηδενικός πίνακας $n \times n$ και I ο μοναδιαίος πίνακας $n \times n$. Εάν προσπαθήσει κανείς να υπολογίσει την μεταβατική κλειστότητα G^*, παρατηρεί ότι:

$$C^* \equiv \begin{bmatrix} I & A & 0 \\ \mathbf{0} & I & B \\ \mathbf{0} & \mathbf{0} & I \end{bmatrix}^* = \begin{bmatrix} I & A & \mathbf{0} \\ \mathbf{0} & I & B \\ \mathbf{0} & \mathbf{0} & I \end{bmatrix}^2 = \begin{bmatrix} I & A & A \cdot B \\ \mathbf{0} & I & B \\ \mathbf{0} & \mathbf{0} & I \end{bmatrix}. \tag{3.1}$$

Με άλλα λόγια, ο πίνακας γειτνιάσεως του G^* εμπεριέχει τον ζητούμενο πίνακα $A \cdot B$. Επομένως, εάν ανακαλυφθεί γρήγορος αλγόριθμος για τον υπολογισμό της μεταβατικής κλειστότητας, τότε έχει βρεθεί τρόπος πολλαπλασιασμού δύο λογικών πινάκων, εντός της ίδιας πολυπλοκότητας. Το Κεφάλαιο 11 παρέχει ενδελεχή μελέτη του ζητήματος των αναγωγών, ενώ στο Κεφάλαιο 8 παρουσιάζονται αλγόριθμοι πολλαπλασιασμού πινάκων.

Ισχυρά Συνεκτικές Συνιστώσες

Η παρούσα Παράγραφος αποτελεί ισχυρή μαρτυρία για την δύναμη μίας τόσο απλής μεθόδου, όπως η ΑσΒ, καθώς υποστηρίζει τρεις γραμμικούς αλγορίθμους ευρέσεως των ισχυρά συνεκτικών συνιστωσών σε ένα κατευθυνόμενο γράφημα $G = (V, E)$.

Λύση Kosaraju. Ο πρώτος αλγόριθμος ανακαλύφθηκε από τον Kosaraju και απαιτεί δύο ΑσΒ και τον υπολογισμό του ανάστροφου γραφήματος G^R. Πιο συγκεκριμένα, αρχικά, εκτελείται μια ΑσΒ στο G^R. Η μεταδιάταξη (postorder) που προκύπτει χρησιμοποιείται, εν συνεχεία, για την διενέργεια μίας δεύτερης ΑσΒ, αυτήν την φορά στο G: κάθε φορά, δηλαδή, που «κολλά» η διαδικασία ΑσΒ, εκκινείται από την ανεξέταστη κορυφή με τον μεγαλύτερο αριθμό μεταδιατάξεως. Μετά το πέρας της διαδικασίας, ως γνωστόν, δημιουργείται ένα δάσος ΑσΒ $\mathcal{F}$. Τα δένδρα ΑσΒ του $\mathcal{F}$ αντιστοιχούν στις ισχυρά συνεκτικές συνιστώσες του G.

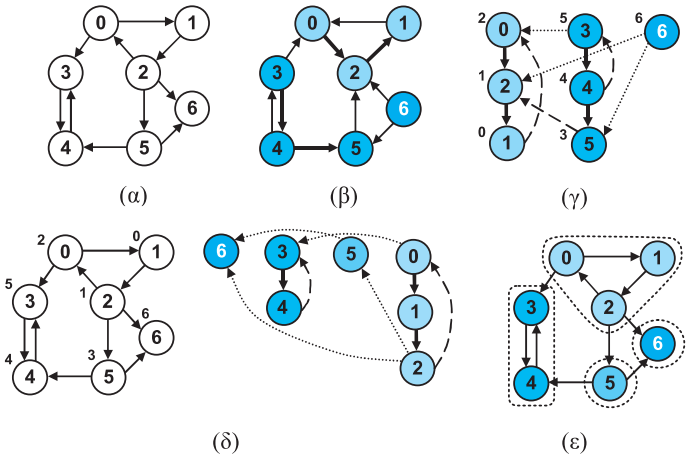
Η όλη διαδικασία περιγράφεται από τον Αλγόριθμο 3.11. Βασική προϋπόθεσή του αποτελεί η ύπαρξη της μεθόδου reverse, η οποία δημιουργεί το ανάστροφο γράφημα. Οι γραμμές 2–6 αποτελούν την πρώτη εκτέλεση ΑσΒ (μέθοδος dfsPostOrder)

```
Algorithm kosarajuSC(graph g)
Input:  Γράφημα g με βοηθητικούς πίνακες post, sc
Output: Ο πίνακας sc θα έχει για κάθε κορυφή την ισχυρά συνιστώσα της

1.  graph gr = g.reverse(); // η μέθοδος επιστρέφει το αντίστροφο γράφημα g^R
2.  for (v = 0; v < gr.V; v++)
3.    gr.sc[v] = −1;          // αρχικοποίηση
4.  for (v = 0; v < gr.V; v++)
5.    if (gr.sc[v] == −1)
6.      dfsPostOrder(gr,v); // υπολογισμός μεταδιατάξεως αντιστρόφου
7.  for (v = 0; v < g.V; v++)
8.    g.sc[v] = −1;
9.  for (v = gr.V−1; v >=0; v−−)     // διέλευση βάσει μεταδιατάξεως
10.   if (g.sc[gr.post[v]] == −1){   // η υ ανεξερεύνητη
11.     dfsPostOrder(g,gr.post[v]);
12.     g.strcompnum++;              // έναρξη νέας ισχυράς συνιστώσας
13.   }

14. static void dfsPostOrder(graph g, vertex v){
15.   g.sc[v] = g.strcompnum;   // απόδοση του αριθμού ισχυράς συνιστώσας
16.   for (x = g.List[v]; x != null; x = x.getNext())
17.     if (g.sc[x.v] == −1)
18.       dfsPostOrder(g,x.v);
19.   g.post[g.porder++] = v; // ο g.post αποθηκεύει «μετα-διατεταγμένες»
20. }                         // τις κορυφές
```

Αλγόριθμος 3.11: Ισχυρά συνεκτικές συνιστώσες βάσει Kosaraju

στο ανάστροφο γράφημα. Ως αποτέλεσμα, παράγεται ο πίνακας post, ο οποίος αποθηκεύει τις κορυφές κατά αύξοντα αριθμό μεταδιατάξεως. Αξίζει να τονισθεί, πως σε αυτήν την πρώτη ΑσΒ, η γραμμή 15 δεν παίζει κανέναν ρόλο. Κατόπιν, στις γραμμές 7–13, εκτελείται η δεύτερη ΑσΒ στο κανονικό, πλέον, γράφημα. Παρατηρήστε ότι οι εκκινήσεις της ΑσΒ γίνονται κατά φθίνουσα μεταδιάταξη (γραμμές 10–11). Κάθε επανεκκίνηση σημαίνει και νέα ισχυρά συνιστώσα (γραμμή 12), ενώ, κατά την διάρκεια μίας ΑσΒ, όσο ανακαλύπτονται νέες ανεξερεύνητες κορυφές, τις αποδίδεται ο τρέχων αριθμός ισχυράς συνιστώσας (γραμμή 15). Σε αυτήν την δεύτερη ΑσΒ, η γραμμή 19 δεν επηρεάζει τους υπολογισμούς.

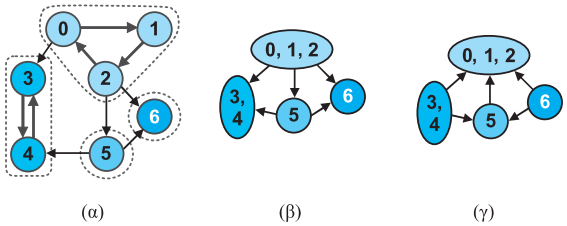
Τα Σχήματα 3.21(α)-(ε) εικονίζουν ένα στιγμιότυπο «τρεξίματος» του αλγορίθμου. Το γράφημα του Σχ. 3.21(β), αποτελεί το ανάστροφο του γραφήματος. Μετά την εκτέλεση της ΑσΒ (Σχ. 3.21(γ)), προκύπτει η μεταδιάταξη 1,2,0,5,4,3,6 των κορυφών. Βάσει αυτής, εκτελείται η δεύτερη ΑσΒ (Σχ. 3.21(δ)): Η εκκίνηση γίνεται από την



Σχήμα 3.21: Εντοπισμός ισχυρά συνεκτικών συνιστωσών βάσει Kosaraju.

κορυφή 6, καθώς έχει τον μεγαλύτερο αριθμό μεταδιατάξεως (6). Καθώς, από την 6 δεν είναι δυνατή περαιτέρω διερεύνηση του γραφήματος, η ΑσΒ επανεκκινείται από την 3, με τον αριθμό μεταδιατάξεως 5. Έτσι, ανακαλύπτεται η 4 και η διαδικασία «ξανακολλά». Οπότε, ξεκινούμε από την αμέσως διαθέσιμη ανεξέταστη κορυφή 5, με αριθμό 3. Και πάλι, παρουσιάζεται ανάγκη επανεκκινήσεως, και επιλέγεται η 0, καθώς ο αριθμός μεταδιατάξεως 2 είναι ο μεγαλύτερος από τους εναπομείναντες. Η τελευταία, αυτή, ΑσΒ ανακαλύπτει όλες τις κορυφές. Επομένως, στο Σχήμα 3.21(ε), εικονίζονται οι ισχυρά συνεκτικές συνιστώσες, αντιστοιχούσες στα δένδρα ΑσΒ.

Η διαισθητική εξήγηση του αλγορίθμου έχει ως εξής (Σχήμα 3.22): Έστω, προς στιγμήν, ότι οι ισχυρά συνεκτικές συνιστώσες του γραφήματος G ήταν γνωστές, εκ των προτέρων. Εάν κάθε συνιστώσα i αντικατασταθεί από έναν «υπερκόμβο» v_i, τότε προκύπτει ένα ΚΑΓ G'· σε αντίθετη περίπτωση, θα υπήρχε κάποιος κύκλος, και, άρα, αμφίδρομη επικοινωνία μεταξύ ισχυρά συνεκτικών συνιστωσών (άτοπο). Στο G' είναι δυνατός ο ορισμός αντίστροφης τοπολογικής διατάξεως. Οπότε, εάν οι «ισχυρές»



Σχήμα 3.22: Διαίσθηση αλγορίθμου Kosaraju: (α) το ΚΑΓ G' και (β) η αναστροφή του.

συνιστώσες εξεταστούν κατά αυτήν την αντίστροφη διάταξη, προκύπτει μία διαπέραση τους από τις περισσότερο προς τις λιγότερο «εξαρτώμενες» συνιστώσες. Επομένως, η διαδικασία αναζητήσεως δεν θα «παρεκκλίνει» σε κορυφές άλλης συνιστώσας. Λόγου χάριν, στο εν λόγω Σχήμα, μία νόμιμη αντίστροφη τοπολογική διάταξη είναι η $\{6\}, \{3,4\}, \{5\}, \{0,1,2\}$. Εάν εκτελεστεί μία ΑσΒ, ξεκινώντας από την 6, σεβόμενοι την διάταξη όταν παρουσιαστεί ανάγκη για επανεκκίνηση, τότε όλες οι συνιστώσες θα αναπαράγονται, η μία μετά την άλλη, πλήρως.

Τυπικότερα, η απόδειξη της ορθότητας έχει ως εξής:

Θεώρημα 3.4 *Ο αλγόριθμος του Kosaraju ανακαλύπτει τις ισχυρά συνεκτικές συνιστώσες ενός γραφήματος G σε γραμμικό, ως προς την πολυπλοκότητα του, χρόνο και χώρο.*

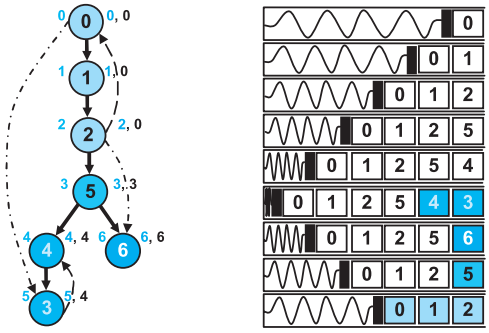
Απόδειξη. Η χρονική πολυπλοκότητα προκύπτει άμεσα, λόγω της γραμμικότητας των πράξεων της αναστροφής γραφήματος και της ΑσΒ. Ως προς το θέμα της ορθότητας, αρκεί να δειχθεί ότι δύο κορυφές v, w ανήκουν στο ίδιο δένδρο της δεύτερης ΑσΒ αν και μόνον αν ανήκουν στην ίδια ισχυρά συνεκτική συνιστώσα.

Εάν οι κορυφές ανήκουν στην ίδια συνιστώσα, τότε θα ανήκουν στο ίδιο δένδρο, καθώς, όταν συναντηθεί η πρώτη από τις κορυφές, δεν θα εγκαταλειφθεί οριστικώς, παρά μόνον εφ' όσον έχουν βρεθεί όλες οι προσπελάσιμες, από αυτήν, κορυφές.

Το αντίστροφο αποδεικνύεται ως εξής: Έστω ρ η ρίζα του δένδρου ΑσΒ των v, w. Επειδή υπάρχει κατευθυνόμενο μονοπάτι από την ρ προς την κορυφή, π.χ., v, συνεπάγεται πως υπάρχει μονοπάτι από την v προς την ρ στο αντίστροφο γράφημα G^R. Εξ αιτίας αυτού του γεγονότος, στο G^R πρέπει να υφίσταται και μονοπάτι από την ρ στη v, καθώς ο αριθμός μεταδιατάξεως της ρ (ως ρίζα του δένδρου) είναι μεγαλύτερος από τον αντίστοιχο της v. Κατά συνέπεια, στο γράφημα G, υπάρχουν κατευθυνόμενα μονοπάτια από την ρ στην v και από την v στην ρ. Ακολουθώντας την ίδια επιχειρηματολογία, αποδεικνύεται πως υπάρχουν κατευθυνόμενα μονοπάτια από την ρ στην w και από την w στην ρ. Με άλλα λόγια, αποδείξαμε ότι $v \leftrightarrows \rho \rightleftarrows w$ και, άρα, $v \leftrightarrows w$. ■

Βελτίωση Tarjan. Το βασικό μειονέκτημα του προηγούμενου αλγορίθμου είναι ο υπολογισμός του ανάστροφου γραφήματος και η εκτέλεση δύο ΑσΒ· ουσιαστικά, δηλαδή, απαιτεί τρεις σαρώσεις του G. Ο Tarjan ανακάλυψε έναν αλγόριθμο που απαιτεί μία απλή σάρωση του γραφήματος, εκμεταλλευόμενος το γεγονός πως η κατά σειρά μεταδιατάξεως επεξεργασία των κορυφών του γραφήματος, σε συνδυασμό με τις τυχόν οπισθοακμές που θα συναντηθούν, ανακαλύπτει τις συνιστώσες κατά αντίστροφη τοπολογική διάταξη.

Εν πολλοίς, ο εν λόγω αλγόριθμος θυμίζει τον αντίστοιχο εντοπισμού των σημείων αρθρώσεως (Αλγ. 3.4) και των δισυνεκτικών συνιστωσών. Εάν, λοιπόν, κατά την διάρκεια μιας ΑσΒ, ανακαλυφθεί για μία κορυφή v, μετά την περάτωση όλων των αναδρομικών κλήσεων που αυτή προκαλεί, ότι είναι η υψηλότερη κορυφή-απόληξη οπισθοακμής από απόγονό της, τότε η v αποτελεί σημείο «εισόδου» σε μία ισχυρά



Σχήμα 3.23: Παράδειγμα ευρέσεως ισχυρά συνεκτικών συνιστωσών κατά Tarjan.

συνεκτική συνιστώσα, αποτελούμενη από αυτήν και τους εμπλεκόμενους απογόνους της, λόγω του κατευθυνόμενου κύκλου, συνιστάμενου από ακμές ΑσΒ και οπισθοακμές.

Η περιγραφή του θα γίνει βάσει του στιγμιοτύπου του Σχ. 3.23. Αριστερά των κόμβων, εικονίζεται ο αριθμός προδιατάξεως και δεξιά, οι διαδοχικοί υπολογισμοί της τιμής low. Θυμηθείτε πως η τελευταία ισούται με την μικρότερη από τις ακόλουθες τιμές: (α) τον αριθμό προδιατάξεως $\mathrm{pre}[v]$ της v, και (β) τον μικρότερο αριθμό προδιατάξεως που μπορεί να ανακαλυφθεί από ένα απλό μονοπάτι π, το οποίο αποτελείται από ακμές απογόνων της v, ακολουθούμενο *το πολύ* από μία οπισθοακμή. Στο δεξί τμήμα του Σχήματος εικονίζεται η επικουρική δομή της στοίβας: Κάθε φορά που συναντάται μία ανεξέταστη κορυφή v, εισάγεται στην στοίβα. Όταν περατωθεί η επεξεργασία της, ελέγχεται εάν είναι σημείο εισόδου. Εάν ναι, τότε οι κορυφές που εμπεριέχονται στην στοίβα, από την αρχή της μέχρι και την v, αποτελούν μία ισχυρά συνιστώσα. Στο παράδειγμά μας, βάσει της ΑσΒ, εισάγονται διαδοχικά οι 0, 1, 2, 5, 4 και 3 στην στοίβα. Αφού ολοκληρωθεί η επεξεργασία της 3, δεν εξάγεται καμμία κορυφή από την στοίβα, καθώς δεν είναι σημείο εισόδου (pre[3]=5≠low[3]=4), λόγω της οπισθοακμής. Κατόπιν, η διαδικασία επιστρέφει στην 4, η οποία είναι αρχή συνιστώσας (pre[4]=low[4]=4) και εξάγεται με δύο ποπ από την στοίβα. Εν συνεχεία, επιστρέφει στην 5, από την 5 μεταβαίνει στην 6, όπου pre[6]=low[6]=6, και άρα είναι αρχή συνιστώσας, κ.ο.κ.

Ο Αλγ. 3.12 αποτελεί μία ευθεία υλοποίηση σε ψευδοκώδικα του αλγορίθμου Tarjan. Ο πίνακας sc χρησιμοποιείται για την καταγραφή των συνεκτικών συνιστωσών, ο πίνακας pre, για την καταγραφή των αριθμών προδιατάξεως, ο πίνακας low, για τον ελάχιστο αριθμό προδιατάξεως προσπελάσιμου προγόνου, και η στοίβα stack για την διατήρηση του τρέχοντος μονοπατιού. Την πρώτη φορά που θα εξεταστεί μία κορυφή, θα εισαχθεί στην στοίβα (γραμμή 4). Εν συνεχεία, εφαρμόζεται ο αλγόριθμος αναδρομικά για τις γειτονικές της κορυφές, εξετάζοντας μήπως υπάρχει απόγονος που την παρακάμπτει, μέσω οπισθοακμής (γραμμές 5–10). Μετά το πέρας όλων των

```
Algorithm tarjanSC(graph g, vertex w)
Input:  Γράφημα g με βοηθητικούς πίνακες pre, sc, low και στοίβα stack
Output: Ο πίνακας sc θα έχει για κάθε κορυφή την ισχυρά συνιστώσα της

1.  g.pre[w] = g.order++;     // αριθμός προδιατάξεως της w
2.  g.low[w] = g.pre[w];      // ο μικρότερος αριθμός προδιατάξεως που προκύπτει
3.  min = low[w];             // από οπισθοακμή προς πρόγονο
4.  g.stack.push(w);          // τοποθέτηση στην βοηθητική στοίβα
5.  for (x = g.List[w]; x != null; x = x.getNext()){
6.    if (pre[x.v] == −1)     // δεν έχει ανακαλυφθεί
7.      tarjanSC(g, x.v);
8.    if (g.low[x.v] < min)   // βρέθηκε οπισθοακμή απογόνου x της w
9.      min = g.low[x.v];     // προς πρόγονό της
10. }
11. if (min < g.low[w]){      // ενημέρωση του ελαχίστου αριθμού προδιατάξεως
12.   g.low[w] = min;         // απογόνου της w προς απόγονο της
13.   return;                 // και επιστροφή αφού δεν είναι αρχή συνιστώσας
14. }
15. do { // είναι αρχή συνιστώσας η οποία ανακτάται με διαδοχικά ποπ
16.   g.sc[(v = g.stack.pop()] = g.strcompnum;
17.   g.low[v] = g.V;         // τέλος της επεξεργασίας της
18. }
19. while (v != w);
20. g.strcompnum++;           // αύξηση του αριθμού συνιστωσών
```

Αλγόριθμος 3.12: Ισχυρά συνεκτικές συνιστώσες βάσει Tarjan

αναδρομικών κλήσεων, εάν, όντως, υπάρχει τέτοιος απόγονος, δεν γίνεται τίποτε (γραμμές 11–14). Διαφορετικά, ανακτάται το τρέχον μονοπάτι π μέσω διαδοχικών ποπ (γραμμές 15–19) και, τελικά, ενημερώνεται ο τρέχων διαθέσιμος αριθμός συνιστώσας (γραμμή 20).

Θεώρημα 3.5 *Ο αλγόριθμος του Tarjan ανακαλύπτει τις ισχυρά συνεκτικές συνιστώσες ενός γραφήματος G σε γραμμικό, ως προς την πολυπλοκότητά του, χρόνο και χώρο.*

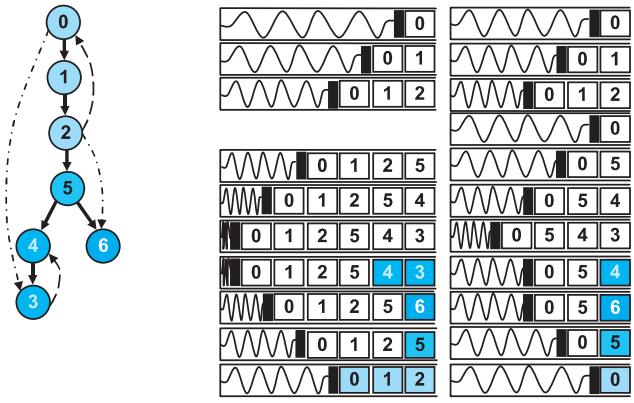
Απόδειξη. Παρατηρήστε ότι πράξεις της στοίβας είναι σταθερού χρόνου, ενώ κάθε κορυφή μία φορά εισάγεται και μία φορά εξάγεται από αυτήν. Επομένως, δεν είναι δύσκολο να αποδειχθεί η γραμμικότητα της πολυπλοκότητας του αλγορίθμου, καθώς αποτελεί επέκταση της ΑσΒ.

Η ορθότητά του στηρίζεται στο γεγονός ότι οι οπισθοακμές παρέχουν εναλλακτικές διαδρομές σε όλες τις κορυφές $z \in \pi$ του μονοπατιού δένδρου ΑσΒ π, που ορίζεται από τα δύο άκρα τους. Έστω, λοιπόν, v η υψηλότερη κορυφή του τρέχοντος μονοπατιού δένδρου ΑσΒ, όλοι οι απόγονοι w της οποίας δεν διαθέτουν οπισθοακμές που

να την ξεπερνούν, αλλά, το πολύ, κάποιες από αυτές, να διασυνδέονται μαζί της. Για κάθε τέτοιον απόγονο w της v υφίσταται ο εξής κατευθυνόμενος κύκλος: από την v στην w, μέσω ακμών δένδρου ΑσΒ, από την w στην πρώτη κορυφή x, απόγονο της w, διαθέτουσα οπισθοακμή που ξεπερνά την w (εξ ορισμού, υπάρχει τέτοια), από την x στην πρώτη κορυφή y, απόγονο της x, με οπισθοακμή που ξεπερνά την y κ.ο.κ. μέχρι να επιστρέψουμε —αναγκαστικά— πίσω στην v. Καθώς σε κάθε κορυφή διατηρείται ο αριθμός προδιατάξεως του υψηλότερου προγόνου της που είναι προσπελάσιμος από κάποιον απόγονό της, και οι κορυφές του τρέχοντος μονοπατιού εξερευνήσεως διατηρούνται στην στοίβα, έπεται ότι κάθε ισχυρά συνεκτική συνιστώσα είναι άμεσα διαθέσιμη μόλις διαπιστωθεί ότι κάποια κορυφή είναι σημείο εισόδου της. ■

Αλγόριθμος Gabow. Ο Gabow το 1999 έδειξε πώς μπορεί να εξαλειφθεί η ανάγκη του δεύτερου βοηθητικού πίνακα low με την κατάλληλη χρήση μίας δεύτερης στοίβας. Η πρώτη στοίβα αποθηκεύει, όπως και στον αλγόριθμο του Tarjan, το τρέχον μονοπάτι. Η δεύτερη στοίβα σκοπό έχει να εντοπίσει ποιες κορυφές είναι σημεία εισόδου σε ισχυρά συνεκτικές συνιστώσες. Ο αλγόριθμος, λοιπόν, ακολουθεί την εξερεύνηση ΑσΒ. Όταν εντοπιστεί μία ανεξέταστη κορυφή v, εισάγεται και στις δύο στοίβες. Εάν βρεθεί μία οπισθοακμή —και, συνεπώς, βρισκόμαστε εντός συνιστώσας— με διαδοχικά ποπ στην δεύτερη στοίβα, παραμένει μόνο η κορυφή-απόληξη της οπισθοακμής· ήτοι, η εγγύτερη προς την ρίζα του δένδρου ΑσΒ κορυφή του αντίστοιχου κύκλου. Κατά την περάτωση της επεξεργασίας μίας κορυφής v, ελέγχεται μήπως η v είναι στην κορυφή της δεύτερης στοίβας. Εάν ναι, τότε αυτό σημαίνει πως η v είναι σημείο εισόδου, και όλες οι κορυφές μονοπατιού ΑσΒ στην πρώτη στοίβα, από την κεφαλή της μέχρι και την v, αποτελούν ισχυρά συνεκτική συνιστώσα. Εάν όχι, τότε δεν πράττουμε τίποτε.

Το Σχήμα 3.24 επεξηγεί τον τρόπο δράσεως του αλγορίθμου. Διαδοχικά, οι $0, 1$ και 2 εισάγονται στις δύο στοίβες. Κατά την επεξεργασία της 2, λόγω της οπισθοακμής, οι



Σχήμα 3.24: Παράδειγμα ευρέσεως ισχυρά συνεκτικών συνιστωσών βάσει Gabow.

$1, 2$ αφαιρούνται από την δεύτερη. Κατόπιν, εισάγονται οι $5, 4, 3$. Με τον εντοπισμό της οπισθοακμής $(3, 4)$, αφαιρείται η 3 από την δεύτερη στοίβα. Κατά το πέρας της επεξεργασίας της 3, αφού δεν είναι κορυφαία στην δεύτερη στοίβα, οπισθοχωρούμε στην 4. Εκεί, καθώς ολοκληρώνονται οι υπολογισμοί, η 4 αναγορεύεται ως σημείο εισόδου, εφ' όσον είναι στην κεφαλή της δεύτερης στοίβας. Ως εκ τούτου, ανακτώνται, με διαδοχικά ποπ από την πρώτη στοίβα, οι $3, 4$, ως μέλη της πρώτης συνιστώσας που συναντούμε στο γράφημα. Κατόπιν, η αναδρομή οπισθοχωρεί στην 5, από την 5 μεταβαίνει στην 6 κ.ο.κ.

Ο Αλγ. 3.13 συνιστά μία περιγραφή του σε ψευδογλώσσα. Η stack αποτελεί την πρωτεύουσα στοίβα, ενώ η pstack την δεύτερη. Κατά την είσοδο, η τρέχουσα κορυφή w εισάγεται και στις δύο στοίβες (γραμμές 2–3). Όταν βρεθεί μία οπισθοακμή (άρα, είμαστε εντός ισχυρά συνεκτικής συνιστώσας), με διαδοχικά ποπ παραμένει μόνο η εγγύτερη προς την ρίζα του δένδρου AσB κορυφή (γραμμές 7–9). Κατά την ολοκλήρωση των υπολογισμών (γραμμή 10), εάν η w ταυτίζεται με την πρώτη κορυφή της pstack, τότε, κατά την διάρκεια της AσB, από την pstack έχουν εξαλειφθεί όλες οι κορυφές της τρέχουσας συνιστώσας, πλην αυτής που αποτελεί σημείο εισόδου στην συνιστώσα (γραμμές 12–13). Οπότε η stack εμπεριέχει τις κορυφές της ισχυρά

```
Algorithm gabowSC(graph g, vertex w)
Input:  Γράφημα g με βοηθητικούς πίνακες pre, sc και στοίβες stack, pstack
Output: Ο πίνακας sc θα έχει για κάθε κορυφή την ισχυρά συνιστώσα της

1.  g.pre[w] = g.order++;    // αριθμός προδιατάξεως
2.  g.stack.push(w);         // ένθεση στις στοίβες
3.  g.pstack.push(w);
4.  for (x = g.List[w]; x != null; x = x.getNext())
5.    if (g.pre[x.v] == −1)      // ανεξερεύνητος απόγονος
6.      gabowSC(g,x.v);
7.    else if (g.sc[x.v] == −1)    // απροσδιόριστος αριθμός συνιστώσας
8.      while (g.pre[g.pstack.top()] > g.pre[x.v])
9.        g.pstack.pop();
10. if (g.pstack.top() != w)  // δεν αποτελεί «κορυφή εισόδου»
11.   return;
12. else
13.   g.pstack.pop();         // αποτελεί «κορυφή εισόδου» σε συνιστώσα
14. do                        // ανάκτηση κορυφών συνιστώσας
15.   g.sc[(t=g.stack.pop())] = g.strcompnum;
16. while (t != w);
17. g.strcompnum++;       // ενημέρωση τρέχοντος αριθμού συνιστώσας
```

Αλγόριθμος 3.13: Ισχυρά συνεκτικές συνιστώσες βάσει Gabow

συνεκτικής συνιστώσας, τις οποίες ανακτούμε με διαδοχικά ποπ (γραμμές 10–16) και ενημερώνουμε τον επόμενο διαθέσιμο αριθμό συνιστώσας (γραμμή 17).

Θεώρημα 3.6 *Ο αλγόριθμος του Gabow ανακαλύπτει τις ισχυρά συνεκτικές συνιστώσες ενός γραφήματος G σε γραμμικό, ως προς την πολυπλοκότητά του, χρόνο και χώρο.*

Απόδειξη. Η χρονική πολυπλοκότητα προκύπτει άμεσα, αφού, κατ' ουσίαν, αποτελεί επέκταση της ΑσΒ, ενώ κάθε κορυφή εισάγεται και εξάγεται και από τις δύο στοίβες ακριβώς μία φορά. Η τυπική απόδειξη ορθότητας παραλείπεται. ■

3.5 Περίληψη Αποτελεσμάτων

Ο Πίνακας 3.5 συνοψίζει τους βασικότερους αλγορίθμους του παρόντος Κεφαλαίου και τις αντίστοιχες πολυπλοκότητές τους. Στην δεξιά στήλη, σημειώνονται τα τυχόν ιδιαίτερα γνωρίσματα καθενός.

Πρόβλημα	**Πολυπλοκότητα**	
Ασβ	$O(V+E)$	
ΑκΠ	$O(V+E)$	
Τοπολογική Διάταξη	$O(V+E)$	
Μεταβατική Κλειστότητα	$O(V(V+E))$ $O(V^4)$ $O(V^3 \log n)$ $O(V^3)$	(V ΑσΒ) (Πολ/σμός Δυαδικού Πίνακα) ($O(\log n)$ Τετραγωνισμοί) (Warshall)
Δισυνεκτικότητα	$O(V+E)$	
Ισχυρά Συνεκτικότητα	$O(V+E)$ $O(V+E)$ $O(V+E)$	(Kosaraju: 2 ΑσΒ) (Tarjan: 1 ΑσΒ, στοίβα, πίνακας) (Gabow: 1 ΑσΒ, 2 στοίβες)

Πίνακας 3.1: Σύνοψη πολυπλοκοτήτων βασικών αλγορίθμων επεξεργασίας γραφημάτων

Ασκήσεις

3.1 Αποδείξτε ότι κάθε πλήρες δυαδικό δένδρο T, ύψους h, διαθέτει το 90% των κόμβων του στα τρία τελευταία επίπεδα.

3.2 Έστω $d_1, d_2, \ldots, d_n, n > 1$ θετικοί ακέραιοι.

(α) Αποδείξτε ότι οι $d_1, d_2, \ldots, d_n$ αποτελούν τους βαθμούς των κορυφών ενός δένδρου T n κορυφών εάν και μόνον αν $\sum_{i=1}^{n} d_i = 2n - 2$.

(β) Σχεδιάστε έναν αλγόριθμο, ο οποίος, δεχόμενος, ως είσοδο, n ακεραίους $d_1, d_2,$ $, \ldots, d_n$, θα κατασκευάζει ένα δένδρο n κορυφών, με βαθμούς τους $d_1, d_2, \ldots, d_n$, εφ' όσον, βέβαια, αυτό είναι εφικτό.

3.3 Προτείνετε έναν αλγόριθμο εξερευνήσεως ενός λαβύρινθου.

3.4 Η ύπαρξη μόνο μίας οπισθοακμής σε ένα δένδρο ΑσΒ συνεπάγεται την ύπαρξη ενός μοναδικού κύκλου; Αιτιολογήστε πλήρως την απάντησή σας.

3.5 Έστω A ο πίνακας γειτνιάσεως ενός γραφήματος $G = (V, E)$. Ποια η σημασία του πίνακα $A \cdot A^T$;

3.6 Τροποποιήστε την διαδικασία ΑκΠ, ώστε να εντοπίζει το πλήθος των διακεκριμένων μονοπατιών μεταξύ δύο κορυφών εντός γραμμικού χρόνου. Μπορείτε να χρησιμοποιήσετε επιπλέον βοηθητικό χώρο.

3.7 Αποφανθείτε εάν είναι δυνατόν ή όχι

(α) η ΑκΠ σε ένα γράφημα $G = (V, E)$ να μην είναι σε θέση να εντοπίσει όλα τα συντομότερα μονοπάτια από μία κορυφή v προς όλες τις υπόλοιπες, αλλά μόνο μερικά.

(β) μία κορυφή v να έχει τοποθετηθεί μόνη της σε ένα δένδρο του δάσους ΑσΒ καίτοι διαθέτει μη μηδενικούς βαθμούς εισόδου και εξόδου.

3.8 Γράψτε ένα πρόγραμμα, το οποίο θα δέχεται ως είσοδο ένα αρχείο $V \times V$ στοιχείων (με αρίθμηση από 0 ως $V^2 - 1$), που περιγράφει ένα γράφημα ως εξής: εάν το i-στο στοιχείο είναι 1, τότε η κορυφή $i \div V$ συνδέεται με την κορυφή $i \mod V$. Διαφορετικά, εάν είναι 0, οι κορυφές δεν συνδέονται μεταξύ τους. Επιπρόσθετα, θα τού γνωστοποιείται εάν το γράφημα είναι κατευθυνόμενο ή όχι. Κατόπιν θα μπορεί να δημιουργεί (ανάλογα με την επιθυμία του χρήστη) την εσωτερική αναπαράσταση είτε με πίνακα γειτνιάσεως είτε με λίστα γειτνιάσεως.

3.9 Υλοποιήστε τις αναζητήσεις ΑσΒ και ΑκΠ, τόσο σε κατευθυνόμενα όσο και σε μη κατευθυνόμενα γραφήματα, χρησιμοποιώντας ως βάση τους αντίστοιχους ψευδοκώδικες του κεφαλαίου. Οι διαδικασίες αναζητήσεως θα πρέπει να είναι σε θέση να κατηγοριοποιούν τις ακμές του γραφήματος σε ακμές δένδρου, οπισθοακμές, κ.ο.κ. Επαληθεύστε τις υλοποιήσεις σας με τα αντίστοιχα παραδείγματα.

3.10 Προτείνετε έναν αλγόριθμο κατασκευής ενός επικαλύπτοντος δάσους E_G για το βοηθητικό γράφημα B_G του αλγορίθμου εντοπισμού δισυνεκτικών συνιστωσών σε ένα γράφημα $G = (V, E)$, (**Υπόδειξη.** Επεξεργαστείτε τις οπισθοακμές κατά αύξοντα

αριθμό προδιατάξεως του άνω άκρο τους, όπως φαίνεται στο δένδρο ΑσΒ. Για κάθε οπισθοακμή b, έστω $e_1, e_2, \ldots, e_j$ οι αντίστοιχες ακμές δένδρου ΑσΒ, διατεταγμένες από την χαμηλότερη προς την υψηλότερη. Προσθέτουμε στο E_G τις ακμές $(b, e_1), (b, e_2), \ldots, (b, e_k), 1 \leq k \leq j$, μέχρι να βρούμε την πρώτη e_k, η οποία έχει εμπλακεί ξανά σε προσθήκη ακμής.)

3.11 Έστω ένα σύνολο n εργασιών T, όπου για κάθε εργασία t_i είναι γνωστή η χρονική της διάρκεια χ_i. Δοθέντος του ΚΑΓ G των εξαρτήσεων μεταξύ των t_i, σχεδιάστε και αναλύστε έναν αλγόριθμο που υπολογίζει τον ελάχιστο χρόνο ολοκληρώσεως του T.

3.12 Περιγράψτε έναν γραμμικό αλγόριθμο που αποφαίνεται, εάν ένα γράφημα $G = (V, E)$ είναι διμελές ή όχι.

3.13 Αποδείξτε ότι το δένδρο ΑσΒ

(α) σε γράφημα μη κατευθυνόμενο αποτελείται μόνο από ακμές δένδρου και οπισθοακμές,

(β) δεν διαθέτει οπισθοακμές αν και μόνον αν το υποκείμενο μη κατευθυνόμενο γράφημα είναι άκυκλο, και

(γ) δεν διαθέτει οπισθοακμές αν και μόνον αν το υποκείμενο κατευθυνόμενο γράφημα είναι άκυκλο.

3.14 Έστω $G = (\{v_1, v_2 \ldots, v_n\}, \{e_1, e_2, \ldots, e_m\})$ κατευθυνόμενο γράφημα και $B_{n \times m}$ δισδιάστατος πίνακας τέτοιος ώστε:

$$B_{i,j} = \begin{cases} 1, & \text{η ακμή } e_j \text{ εισέρχεται στην κορυφή } v_i \\ -1, & \text{η ακμή } e_j \text{ εξέρχεται της κορυφής } v_i \\ 0, & \text{διαφορετικά.} \end{cases}$$

Τι δίδουν οι θέσεις του πίνακα $B \cdot B^T$;

3.15 Σχεδιάστε και αναλύστε έναν αλγόριθμο, ο οποίος, σε γραμμικό χρόνο, θα αποφαίνεται εάν μπορεί να αποδοθεί σε κάθε κορυφή v ενός γραφήματος $G = (V, E)$ μία ταμπέλα από το σύνολο $\{r, b\}$, ούτως ώστε γειτονικές κορυφές να διαθέτουν διαφορετικές ταμπέλες.

3.16 Προτείνετε μία επέκταση της ΑσΒ, ώστε να εντοπίζει κύκλο Euler σε ένα γράφημα, εάν αυτός υπάρχει.

3.17 *Κάλυμμα κορυφής* (*vertex cover*) ενός γραφήματος $G = (V, E)$ καλείται ένα σύνολο $V' \subseteq V$, εάν κάθε ακμή $e \in E$ είναι προσκείμενη σε μία τουλάχιστον κορυφή $v \in V'$.

(α) Βρείτε έναν αλγόριθμο, ο οποίος, εντός γραμμικού χρόνου $O(V + E)$, υπολογίζει το μέγεθος του μικρότερου καλύμματος κορυφής ενός δένδρου.

(β) Προτείνετε έναν αλγόριθμο που αποφαίνεται εάν υπάρχει ή όχι κάλυμμα κορυφής σε ένα μη κατευθυνόμενο γράφημα, αποτελούμενο από $\leq k$ ανεξάρτητες (μη γειτονικές) κορυφές.

3.18 Έστω $G = (V, E)$ ένα συνεκτικό, μη κατευθυνόμενο γράφημα, και e_1, e_2, e_3 τρεις ακμές του. Πώς μπορείτε να βρείτε εάν υπάρχει κύκλος που περιέχει τις e_1, e_2 αλλά όχι την e_3;

3.19 Δοθέντος ενός μη κατευθυνόμενου γραφήματος $G = (V, E)$, σε πόσο χρόνο είναι δυνατόν να διαπιστωθεί εάν είναι δένδρο ή όχι;

3.20 Δοθέντων των σημείων αρθρώσεως ενός γραφήματος $G = (V, E)$, περιγράψτε έναν αλγόριθμο εντοπισμού των δισυνεκτικών συνιστωσών.

3.21 Προτείνετε και αναλύστε έναν αλγόριθμο, ο οποίος, δεχόμενος ως είσοδο ένα μη κατευθυνόμενο γράφημα $G = (V, E)$, προσθέτει τον ελάχιστο αριθμό ακμών, ώστε το G να καταστεί δισυνεκτικό, εάν δεν είναι.

3.22 Έστω Σ ένα σύστημα ανισοτήτων της μορφής $x_i < x_j$. Σχεδιάστε έναν αλγόριθμο που αποφαίνεται εάν το Σ είναι επιλύσιμο ή όχι.

3.23 Περιγράψτε και αναλύστε έναν αλγόριθμο εντοπισμού όλων των κύκλων μήκους 4 σε μη κατευθυνόμενα γραφήματα.

3.24 Δοθέντος ενός ΚΑΓ $G = (V, E)$, όπου κάθε ακμή $e \in E$ φέρει μία ταμπέλα $l(e) \in \{0, 1\}$, αιτείται η ανάπτυξη ενός αλγορίθμου που, εντός χρόνου $O(V + E)$, υπολογίζει για κάθε κορυφή $v \in V$ το πλήθος $a(v)$ των εναλλασσόμενων μονοπατιών με σημείο εκκινήσεως την v. *Εναλλασσόμενο* χαρακτηρίζεται ένα μονοπάτι, όπου οι ταμπέλες εναλλάσσονται, καθώς μεταβαίνουμε από κορυφή σε κορυφή. Λόγου χάριν, τα $v_1 \xrightarrow{0} v_2$, $v_1 \xrightarrow{1} v_2$, $v_1 \xrightarrow{1} v_2 \xrightarrow{0} v_3$, $v_1 \xrightarrow{0} v_2 \xrightarrow{1} v_3 \xrightarrow{0} v_4 \xrightarrow{1} v_5$, αποτελούν στιγμιότυπα εναλλασσόμενων μονοπατιών.

3.25 Έστω $G = (V, E)$ ένα ΚΑΓ με μέγιστο μήκος μονοπατιού k. Περιγράψτε έναν αλγόριθμο που διαμερίζει το V σε $k + 1$, το πολύ, σύνολα, ξένα μεταξύ τους, τέτοια ώστε δύο κορυφές v, w ανήκουν στην ίδια διαμέριση εάν και μόνον αν δεν υπάρχουν μονοπάτια από το v στο w και από το w στο v.

3.26 (α) Να σχεδιαστεί και να αναλυθεί ένας αλγόριθμος που να ανακαλύπτει εάν ένα ΚΑΓ $G = (V, E)$ διαθέτει ή όχι μονοπάτι Hamilton.

(β) Σε ένα ΚΑΓ $G = (V, E)$ το μακρύτερο μονοπάτι καλείται *κρίσιμο*. Περιγράψτε και αναλύστε έναν αλγόριθμο εντοπισμού των κρίσιμων μονοπατιών.

3.27 Προτείνετε μία τροποποίηση της διαδικασίας ΑσΒ, ώστε να υπολογίζει την μεταβατική κλειστότητα ενός ΚΑΓ εισόδου. (**Υπόδειξη.** Συνδυάστε την ΑσΒ με την τεχνική του

δυναμικού προγραμματισμού, εκμεταλλευόμενοι το γεγονός ότι οι εμπροστοακμές και οι ακμές διασταυρώσεως οδηγούν σε κορυφές, για τις οποίες *όλα* τα μονοπάτια έχουν ήδη υπολογιστεί.)

3.28 Μετατρέψτε τον Αλγ. 3.4 των σημειώσεων σε πραγματικό κώδικα, ώστε να βρίσκει τα σημεία αρθρώσεως, τις γέφυρες και τις δισυνεκτικές συνιστώσες.

3.29 Υλοποιήστε τους αλγορίθμους ευρέσεως των ισχυρά συνεκτικών συνιστωσών των Kosaraju, Gabow και Tarjan (Αλγόριθμοι 3.11, 3.13 και 3.12 αντίστοιχα).

3.30 Ένα κατευθυνόμενο γράφημα $G = (V, E)$ καλείται ημι-συνεκτικό, εάν μεταξύ δύο οποιαδήποτε κορυφών $v, w \in V$ υπάρχει μονοπάτι $v \rightsquigarrow w$ ή $w \rightsquigarrow v$. Σχεδιάστε και αναλύστε έναν αλγόριθμο, ο οποίος αποφασίζει εάν ένα κατευθυνόμενο γράφημα είναι ημι-συνεκτικό ή όχι. (**Υπόδειξη.** Τροποποιήστε τον Αλγόριθμο Kosaraju.)

3.31 Σχεδιάστε έναν αλγόριθμο κόστους $O(E)$ υπολογισμού κύκλου Euler σε ένα μη κατευθυνόμενο γράφημα $G = (V, E)$. (**Υπόδειξη.** Θυμηθείτε το σχετικό Θεώρημα υπάρξεως ή όχι του κύκλου.)

3.32 Πώς αλλάζει ο πίνακας μεταβατικής κλειστότητας, εάν προσθέσουμε μία ακμή; Βασιζόμενοι σε αυτήν την παρατήρηση, προτείνετε μία τροποποίηση του αλγορίθμου Warshall (Αλγ. 3.10), ώστε να επεξεργάζεται «ημι-δυναμικά» γραφήματα (γραφήματα, δηλαδή, που επιτρέπουν την προσθήκη ακμών). Ποια η πολυπλοκότητα χειρότερης περιπτώσεως μίας ενθέσεως;

3.33 Κάθε ύφασμα αποτελείται από δύο ακολουθίες νημάτων H, V, οριζοντίων και καθέτων, αντίστοιχα, που περιπλέκονται μεταξύ τους. Το μοτίβο της πλέξης δύναται να παρασταθεί με έναν ορθογώνιο δυαδικό πίνακα A, διαστάσεως $|H| \times |V|$, όπου $A_{i,j} = 0$ εάν και μόνο αν το i-στό οριζόντιο νήμα h_i περνά κάτω από το j-στό κάθετο νήμα v_j. Δοθέντος του A, αιτείται να σχεδιαστεί αλγόριθμος που να αποφαίνεται εάν το αντίστοιχο μοτίβο πλέξης είναι σωστό· δηλαδή, εάν εφαρμοστεί, παράγεται ένα «μονοκόμματο» ύφασμα. (**Υπόδειξη.** Αναπαραστήστε τον A με ένα κατάλληλο διμελές κατευθυνόμενο γράφημα G, το οποίο θα είναι ισχυρά συνεκτικό, εάν η πλέξη είναι σωστή.)