ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

Αλγόριθμοι Βελτιστοποίησης της Δικτυακής Ροής και Εφαρμογές

Μαρία Π.Ρέππα

Πτυχιακή Εργασία

Ηράκλειο, Ιούλιος 2004

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΜΑΘΗΜΑΤΙΚΩΝ

Συγγραφέας:

Μαρία Π. Ρέππα
Τμήμα Μαθηματικών
Πανεπιστήμιο Κρήτης

Εξεταστική Επιτροπή:

Μιχάλης Κολουντζάκης, Αναπληρωτής Καθηγητής
Επόπτης

Μανώλης Βάβαλης, Αναπληρωτής Καθηγητής
Μέλος

Γιώργος Ζουράρης, Επίκουρος Καθηγητής
Μέλος

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΕΠΙΣΤΗΜΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Πτυχιακή Εργασία

Αλγόριθμοι Βελτιστοποίησης της Δικτυακής Ροής Και Εφαρμογές

Μαρία Π. Ρέππα

Περιεχόμενα

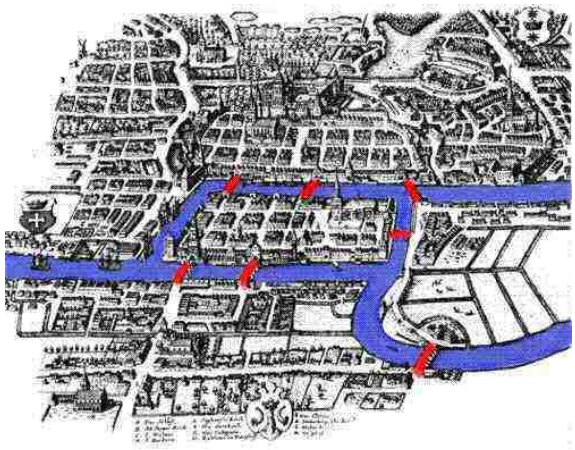
Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή στη Θεωρία Γράφων

1.1 Χρησιμότητα και Εφαρμογές της Θεωρίας Γράφων

Η μελέτη των γραφημάτων ξεκινάει το 1736 όταν ο Euler έλυσε το πρόβλημα αν δεδομένου του χάρτη της πόλης Konigsberg της Γερμανίας, είναι δυνατόν κάποιος να διασχίσει και τις 7 γέφυρες του ποταμού Pregel και να επιστρέψει εκεί απ'οπου ξεκίνησε χωρίς να περάσει δύο φορές από την ίδια γέφυρα.

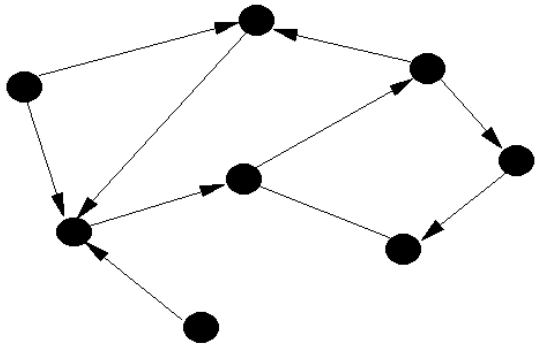


Σχήμα 1.1: Το πρόβλημα του Euler

Πολλά προβλήματα μοντελοποιούνται μέσω γραφημάτων και λύνονται με την βοήθεια της θεωρίας γράφων. Παρακάτω αναφέρονται μερικά χαρακτηριστικά παραδείγματα.

- Δέντρα παραγόμενα από γραφήματα.

Σε ένα χωράφι υπάρχουν 5 βρύσες παροχής νερού, από τις οποίες η η μία είναι η κεντρική παροχή, ενώ οι άλλες τροφοδοτούνται από την πρώτη. Με ποιόν τρόπο θα τοποθετηθούν οι σωλήνες που θα ενώνουν τις βρύσες έτσι ώστε το συνολικό κόστος να είναι το μικρότερο δυνατό· (Πρόβλημα εύρεσης ελάχιστου δέντρου - Shortest Spanning Tree.)



- Σχεδίαση Δικτύων - Τηλεπικοινωνίες.

  Όταν υπάρχουν διακριτά σημεία (κορυφές) και συνδέσεις μεταξύ δύο κόμβων (ακμές) και το πρόβλημα έγκειται στο να σχεδιάσουμε ένα κατάλληλο δίκτυο για να συνδέσουμε διάφορους κόμβους με το ελάχιστο δυνατό κόστος και με τρόπο που να ικανοποιούνται διάφορα κριτήρια, τότε έχουμε ένα πρόβλημα σχεδίασης κατάλληλου δικτύου. Παραδείγματος χάριν μπορούμε να σχεδιάσουμε ένα δίκτυο με ελάχιστο κόστος, το ποιο λειτουργεί ακόμα μετά από καταστροφή μιας οποιαδήποτε σύνδεσης.

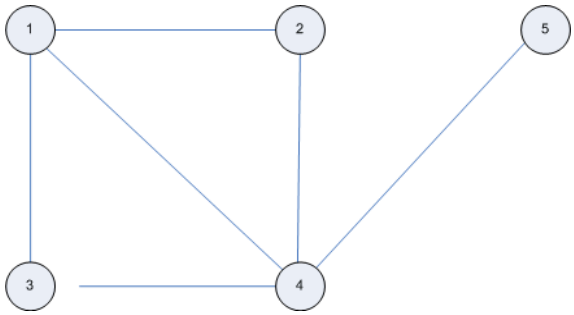
  Τέτοια προβλήματα γίνονται όλο και πιο σημαντικά σήμερα, ειδικότερα στο χώρο των τηλεπικοινωνιών. Στην απλούστερη περίπτωση, δεδομένων των κόμβων και του κόστους κάθε σύνδεσης, το πρόβλημα είναι ποιες συνδέσεις πρέπει να χρησιμοποιηθούν ώστε να ελαχιστοποιηθεί το συνολικό κόστος και ταυτόχρονα εξασφαλίζοντας ότι το δίκτυο συνδέεται.

- Συντομότερο μονοπάτι.

  Στο παρακάτω γράφημα πώς μπορούμε να βρούμε τον συντομότερο δρόμο από την κορυφή 1 στην κορυφή 5;

  Σε ένα μικρό παράδειγμα μπορούμε να βρούμε το συντομότερο μονοπάτι από την κορυφή 1 στην κορυφή 5 μετά από παρατήρηση, ή με την απαρίθμηση όλων των μονοπατιών από την κορυφή 1 στην κορυφή 5. Εντούτοις εάν το γράφημα ήταν πολύ μεγαλύτερο, αυτό δεν θα ήταν εφικτό.

  Το πρόβλημα της εύρεσης του συντομότερου μονοπατιού ενός γραφήματος λύνει ο αλγόριθμος του Dijsktra (αρχές της δεκαετίας του '60).



Ένα παράδειγμα της χρησιμότητας του αλγορίθμου εύρεσης του συντομότερου μονοπατιού είναι η εφαρμογή του σε ένα οδικό δίκτυο έτσι ώστε μία εταιρία μεταφορών να έχει την δυνατότητα να προγραμματίσει τις διαδρομές για τα οχήματα παράδοσης, βάσει της μικρότερης απόστασης μεταξύ των στάσεων, ή του μικρότερου χρόνου μεταξύ των στάσεων.

- Προγραμματισμός πολυεθνικού φόρου.

  Για να εξηγήσουμε το πρόβλημα προγραμματισμού πολυεθνικού φόρου, θα χρησιμοποιήσουμε σαν παράδειγμα την δομή μιας πολυεθνικής επιχείρησης εκμετάλλευσης με τρεις θυγατρικές που εδρεύουν στην Ιαπωνία, το Χογκ Κογκ και την Σιγκαπούρη αντίστοιχα.

  Οι οικονομικές ροές μεταξύ αυτών των επιχειρήσεων μπορούν να ταξινομηθούν σε δύο κατηγορίες:

  1. μερίσματα (που πληρώνονται στους μετόχους) και
  2. τόκοι (στα δάνεια).

  Όταν μια οικονομική ροή που εμπίπτει σε μια από αυτές τις κατηγορίες εμφανίζεται μεταξύ δύο επιχειρήσεων από διαφορετικές χώρες, η επεξεργασία εκείνης της ροής (σχετικά με το ποσό που πληρώνεται σαν φόρος σε κάθε χώρα) ελέγχεται από μια συνθήκη μεταξύ των δύο χωρών. Τα διαφορετικά ζευγάρια των χωρών συνάπτουν διαφορετικές συνθήκες που αμέσως αυξάνει τη δυνατότητα πληρωμής διαφορετικού φόρου ανάλογα με τη διαδρομή που ακολουθεί η οικονομική ροή. Το πρόβλημα αυτό μπορεί να αναπαρασταθεί από ένα γράφημα με βάρη. Οι κορυφές θα αντιπροσωπεύουν τις χώρες όπου εδρεύει η κάθε εταιρία και οι ακμές την οικονομική συνθήκη που διέπει τις χώρες ανά δύο με βάρη το ποσοστό φόρου που πρέπει να πληρώσει η μία στην άλλη. Παραδείγματος χάριν, μια πληρωμή τόκου από την ιαπωνική θυγατρική άμεσα στην πολυεθνική επιχείρηση εκμετάλλευσης μπορεί να είναι χειρότερη εάν η πληρωμή γίνει στη θυγατρική του Χογκ Κογκ και έπειτα η θυγατρική του Χογκ Κογκ πληρώσει αυτή τους τόκους στην πολυεθνική επιχείρηση.

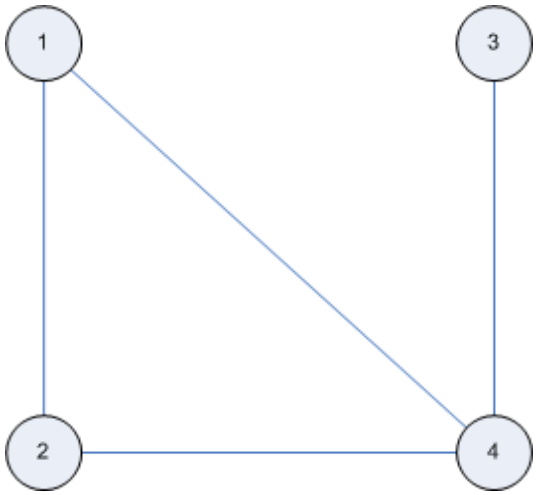
1.2 Ορισμοί

Θα συνεχίσουμε με μία εισαγωγή στην θεωρία γραφημάτων με τους βασικούς ορισμούς και με αρκετά παραδείγματα, έτσι ώστε να γίνουν εύκολα κατανοητά όσα θα περιγράψουμε παρακάτω.

Ορισμός 1. *Ένα **γράφημα** G αποτελείται από κορυφές και ακμές. Είναι ένα ζεύγος (V, E), όπου V είναι πεπερασμένο σύνολο από κορυφές και το E ένα σύνολο από ζεύγη (διατεταγμένα ή μη) κορυφών. Δείτε σχήμα 1.2. **Ακμή** ορίζεται να είναι ένα σύνολο από δύο κορυφές (u, v) με $u, v \in V$, οι οποίες ενώνονται με μία γραμμή.*

γράφημα

ακμή



Σχήμα 1.2: Μη-κατευθυνόμενο γράφημα.

Ένα γράφημα μπορεί να είναι κατευθυνόμενο ή μη.

Ορισμός 2. *Αν είναι **κατευθυνόμενο**, τότε κάθε ακμή του, είναι ένα διατεταγμένο ζεύγος κορυφών, για παράδειγμα η ακμή (u, v) ξεκινάει από το u στο v ενώ η (v, u) από το v στο u. Επίσης, αν το γράφημα είναι κατευθυνόμενο οι ακμές που συνδέονται με κάποια κορυφή χωρίζονται σε εξερχόμενες εάν ξεκινούν από την κορυφή αυτή ή εισερχόμενες αν καταλήγουν σε αυτή την κορυφή. Δείτε σχήμα 1.3.*

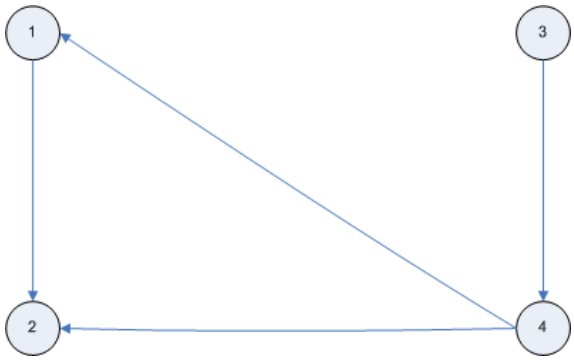
κατευθυνόμενο γράφημα

εξερχόμενες - εισερχόμενες κορυφές

Ορισμός 3. *Αν το γραφημα είναι **μη-κατευθυνόμενο**, τότε η ακμή είναι ένα δισύνολο κορυφών $\{u, v\}$ και μπορούμε να την διασχίσουμε με όποιο τρόπο θέλουμε, από το u στο v και ανάποδα.*

μη-κατευθυνόμενο γράφημα

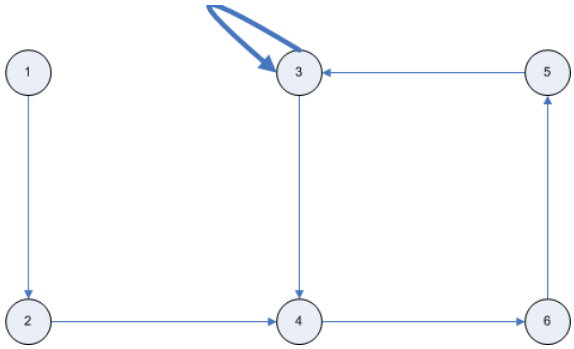
Λέμε ότι μία κορυφή u συνδέεται με μία άλλη v αν υπάρχει ακμή από το u στο v.



Σχήμα 1.3: Κατευθυνόμενο γράφημα.

Ορισμός 4. *Αν μία κορυφή συνδέεται με τον εαυτό της τότε η ακμή που ξεκινά και καταλήγει στην ίδια κορυφή λέγεται* ***βρόγχος*** *(loop). Δείτε σχήμα 1.4.* *βρόγχος*

Ορισμός 5. ***Μονοπάτι*** *στο G είναι ένα υπογράφημα $P = G(V', E')$ του γραφήματος G με $V' = \{x_0, x_1, x_2, ..., x_k\}$ και $E' = \{x_0x_1, x_1x_2, ..., x_{k-1}x_k\}$. Τα $\{x_0, x_1, ..., x_k\}$ είναι κάποιες από τις κορυφές του G και το σύνολο E' περιέχει κάποιες ακμές από το E. Δείτε σχήμα 1.5.* *μονοπάτι*



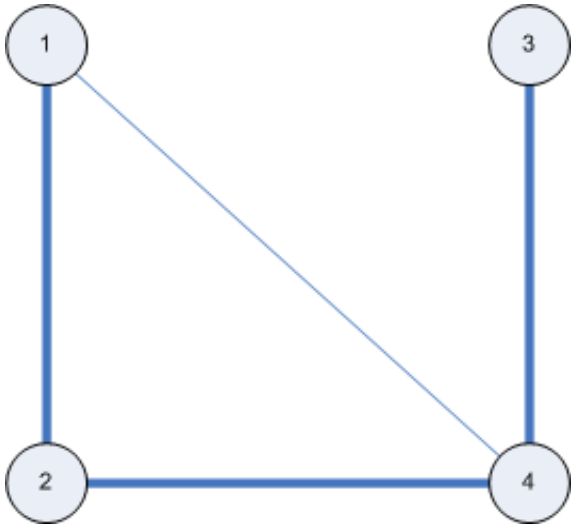
Σχήμα 1.4: Βρόγχος.

Δηλαδή, μονοπάτι είναι μία σειρά από κορυφές ενός γραφήματος όπου κάθε κορυφή συνδέεται με την επόμενη.

Ορισμός 6. ***Κύκλος*** *ονομάζεται ένα μονοπάτι P το οποίο ξεκινάει και καταλήγει στην ίδια κορυφή. $P = x_0x_1x_2...x_k - 1x_0, k \geq 3$. Δείτε σχήματα 1.7 και 1.8.* *κύκλος*

Το **μήκος κύκλου** είναι το πλήθος των ακμών από τις οποίες αποτελείται. *μήκος ενός κύκλου*

Ορισμός 7. *Ένα γράφημα λέγεται* ***συνεκτικό*** *αν κάθε ζεύγος από κορυφές συνδέεται μέσω ενός μονοπατιού, δηλαδή αν ακολουθώντας ένα μονοπάτι, μπορούμε να ξεκινήσουμε από μία οποιαδήποτε κορυφή του G και να καταλήξουμε σε οποιαδήποτε άλλη κορυφή του G.* *συνεκτικό γράφημα*



Σχήμα 1.5: Μονοπάτι σε μη-κατευθυνόμενο γράφημα.

Ορισμός 8. *Αν G γράφημα, ένα σύνολο κορυφών του V_1 ονομάζεται* ***συνεκτική συνιστώσα*** *του G αν κάθε κορυφή του V_1 είναι προσιτή από οποιαδήποτε άλλη, και κάθε κορυφή x που είναι προσιτή σε κάποια κορυφή του V_1, ανήκει στο V_1.*

συνεκτική συνιστώσα

Ορισμός 9. *Ένα γράφημα με περισσότερες από μία συνεκτικές συνιστώσες, ονομάζεται* ***μη-συνεκτικό*** *. Δείτε σχήμα 1.9.*

μη-συνεκτικό γράφημα

Ορισμός 10. ***Δάσος*** *είναι ένα γράφημα χωρίς κύκλους ή βρόγχους. Αν ένα δάσος είναι συνεκτικό, τότε ονομάζεται* ***δέντρο*** *. Δείτε σχήματα 1.11 και 1.10.*

δάσος
δέντρο

Ορισμός 11. *Κάθε γράφημα συνεκτικό γράφημα περιέχει ένα υπογράφημα με τις ίδιες κορυφές, συνεκτικό και χωρίς κύκλους, δηλαδή δέντρο. Αυτό ονομάζεται* ***επικαλύπτον δέντρο****.Δείτε σχήμα 1.12.*

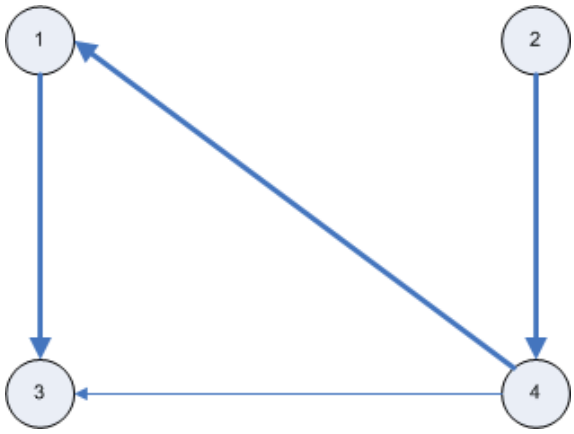
επικαλύπτου δέντρο

Γραφήματα με βάρη είναι εκείνα τα γραφήματα των οποίων κάθε ακμή σχετίζεται με ένα **βάρος** που δίνεται από μία συνάρτηση βάρους

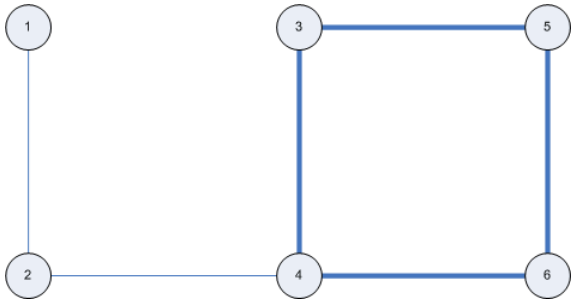
βάρος

$$w : E \to \mathbb{R}.$$

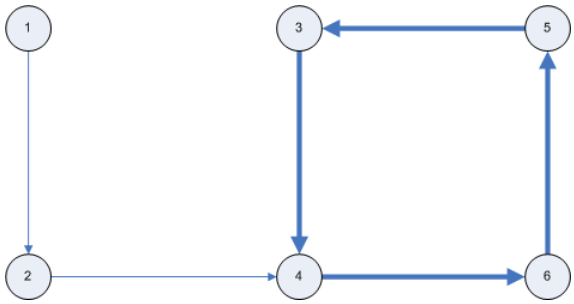
Για παράδειγμα δείτε σχ. 1.13.



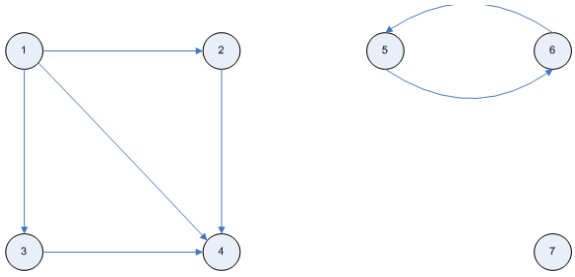
Σχήμα 1.6: Μονοπάτι σε κατευθυνόμενο γράφημα.



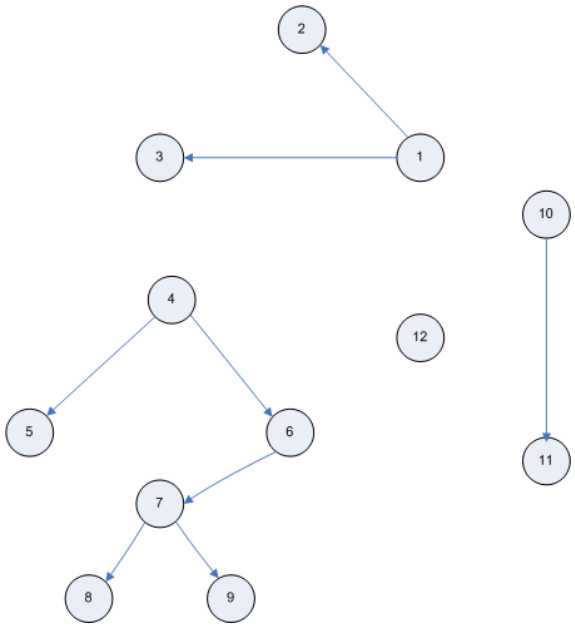
Σχήμα 1.7: Κύκλος σε μη-κατευθυνόμενο γράφημα μήκους 3.



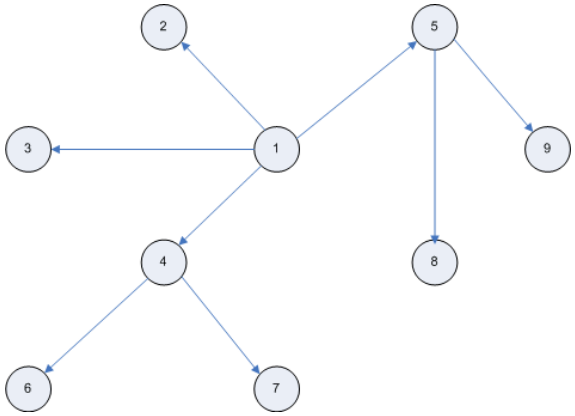
Σχήμα 1.8: Κύκλος σε κατευθυνόμενο γράφημα μήκους 3.



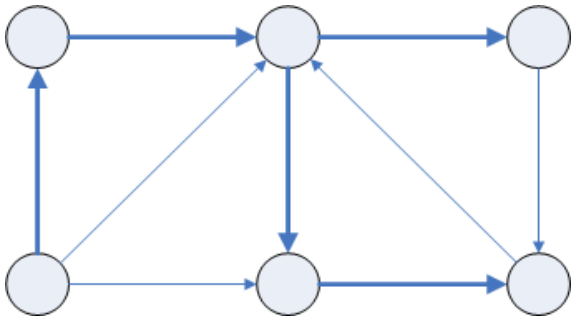
Σχήμα 1.9: Μη συνεκτικό γράφημα.



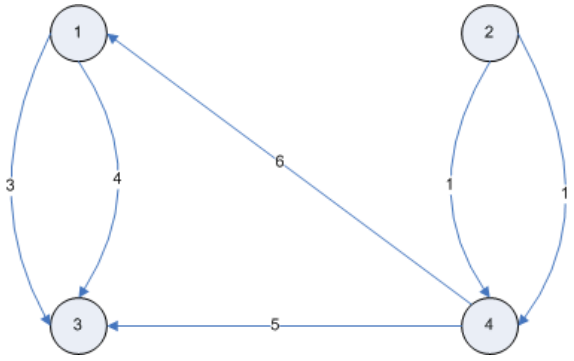
Σχήμα 1.10: Δάσος.



Σχήμα 1.11: Δέντρο.



Σχήμα 1.12: Οι ακμές με μεγαλύτερο πλάτος αποτελούν το επικαλύπτον δέντρο του γραφήματος.



Σχήμα 1.13: Κατευθυνόμενο γράφημα με βάρη.

Κεφάλαιο 2

Αναπαράσταση δεδομένων

Προκειμένου να επεξεργαστούμε γραφήματα πρέπει αρχικά να αποφασίσουμε πώς θα τα αναπαραστήσουμε στον υπολογιστή. Το πρώτο βήμα στην αναπαράσταση ενός γραφήματος είναι να αντιστοιχήσουμε τις κορυφές με ακέραιους αριθμούς μεταξύ 0 και $V-1$. Ο κύριος λόγος για αυτό είναι η γρήγορη πρόσβαση στις πληροφορίες που αντιστοιχούν σε κάθε κορυφή.

2.1 Δύο κύριοι τρόποι αναπαράστασης

Υπάρχουν δύο κύριοι τρόποι αναπαράστασης ενός γραφήματος : με ένα σύνολο από συνδεδεμένες λίστες ή με ένα πίνακα συνδεσμολογίας.

2.1.1 Πίνακας Συνδεσμολογίας

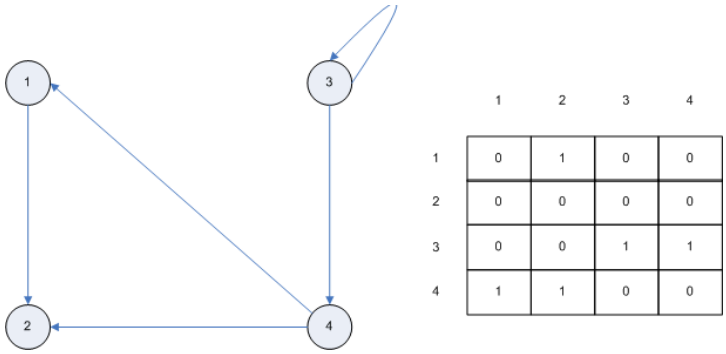
Έστω ένα γράφημα $G = (V, E)$ με ν κορυφές, $V = v_1, v_2, ..., v_n$. Η πιο απλή αναπαράσταση χρησιμοποιεί έναν πίνακα $n \times n$ από μηδενικά και άσσους που δίνεται από την σχέση

$$Adj(i,j) = \begin{cases} 1, & \alpha\nu \quad (i,j) \in E \\ 0, & \alpha\lambda\lambda\iota\omega\varsigma \end{cases} \tag{2.1}$$

Δείτε Σχήμα 2.1.1. Προφανώς ο αριθμός των άσσων του πίνακα είναι ίσος με το πλήθος των ακμών του γραφήματος.

Μπορούμε να αναπαραστήσουμε και μη-κατευθυνόμενα γραφήματα αν θεωρήσουμε τις ακμές σαν μη-διατεταγμένα δισύνολα v_i, v_j και τότε ο πίνακας θα είναι συμμετρικός ως προς τη διαγώνιο. Σε αυτή την περίπτωση οι άσσοι είναι διπλάσιοι από το πλήθος των ακμών του γραφήματος.

Ένα πλεονέκτημα του πίνακα συνδεσμολογίας είναι ότι μπορούμε σε σταθερό χρόνο να αποφασίσουμε αν μια κορυφή i συνδέεται με την j. Διατρέχουμε την ι-οστή γραμμή και αν βρούμε την κορυφή j, συμπεραίνουμε ότι υπάρχει



Σχήμα 2.1: Αναπαράσταση με πίνακα συνδεσμολογίας.

η ακμή (i, j) που συνδέει την κορυφή i με την j. Σε ένα κατευθυνόμενο γράφημα πρέπει να εξετάσουμε και την ι-οστή στήλη για να δούμε εάν υπάρχει η ακμή (j, i).

Για γραφήματα με βάρη στην θέση κάθε άσσου θα τοποθετούμε το βάρος της αντίστοιχης ακμής και όπου δεν υπάρχει ακμή θα τοποθετούμε το σύμβολο ∞.

Εφόσον ο πίνακας συνδεσμολογίας έχει $|V|^2$ σύμβολα, ο αναγκαίος χώρος για την αναπαράσταση ενός γραφήματος, *ανεξαρτήτως του πλήθους των ακμών του*, είναι $O(|V|^2)$[1]. Αν το γράφημα έχει σχετικά λίγες ακμές, δηλαδη αν $|E| \ll |V|^2$, τότε τα περισσότερα στοιχεία του πίνακα συνδεσμολογίας είναι 0 ή ∞. Τότε ονομάζουμε αυτό το γράφημα **αραιό** . Δείτε Σχήμα 2.2.

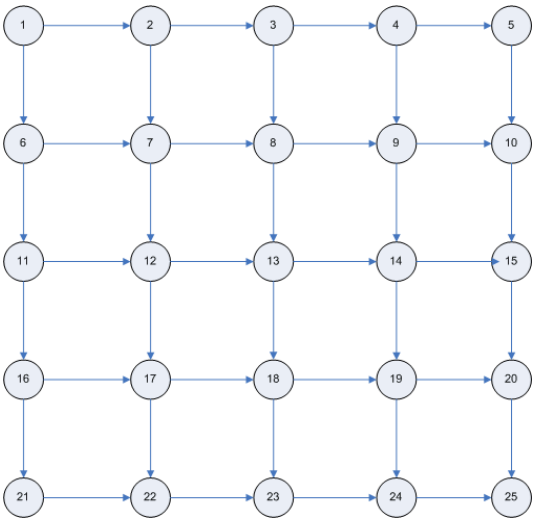
αραιό γράφημα

2.1.2 Λίστα Συνδεσμολογίας

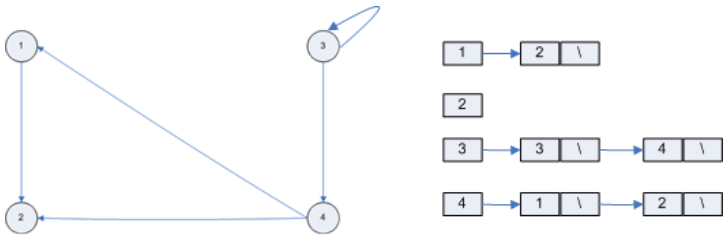
Η αναπαράσταση με συνδεδεμένες λίστες αποτελείται από ένα διάνυσμα Adj από $|V|$ λίστες, μία για κάθε κορυφή του συνόλου κορυφών V. Για κάθε $u \in V$, η λίστα $Adj(u)$ περιέχει όλους τους γείτονες τις κορυφής u με τυχαία σειρά. Κάθε στοιχείο της λίστας έχει δύο πληροφορίες, μια κορυφή και ένα ¨δείκτη¨, ο οποίος δείχνει την επόμενη κορυφή της λίστας. Η λίστα τελειώνει όταν βρεθεί δείκτης με τιμή NULL, που σημαίνει ότι δεν υπάρχει άλλη κορυφή που να συνδέεται με την u. Μπορούμε να προσθέσουμε ή να αφαιρέσουμε μία κορυφή από τη λίστα προσαρμόζοντας τους δείκτες σύμφωνα με το Σχήμα 2.4.

Αν το γράφημα είναι κατευθυνόμενο, το άθροισμα του μήκους όλων των λιστών είναι $|E|$, αφού μια ακμή της μορφής (u, v) παρίσταται με την εμφάνιση του v στην $Adj(u)$. Αν είναι μη-κατευθυνόμενο, το άθροισμα του μήκους όλων των λιστών είναι $2|E|$, αφού η (u, v) είναι ακμή χωρίς κατεύθυνση και

[1] $O(g(n)) = \{f(n)$: υπάρχουν θετικές σταθερές c και n_0 έτσι ώστε $0 \leq f(n) \leq cg(n)$ για κάθε $n \geq n_0$



Σχήμα 2.2: Αραιό γράφημα.



Σχήμα 2.3: Αναπαράσταση με συνδεδεμένη λίστα.

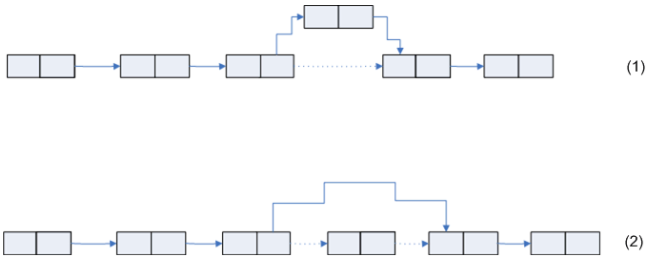
έτσι το v εμφανίζεται στη λίστα του u και αντίστροφα. Για κατευθυνόμενα και μη γραφήματα η αναπαράσταση με λίστα έχει χρειάζεται $\Theta(V + E)$ χώρο αποθήκευσης[2].

Ένα πιθανό μειονέκτημα είναι ότι δεν υπάρχει συντομότερος τρόπος για να καθοριστεί αν μία ακμή (u, v) υπάρχει στο γράφημα ή όχι από το να ψάξουμε για το v στην $Adj(u)$.

2.2 Σύγκριση Πίνακα και Λίστας Συνδεσμολογίας

Προκειμένου να κάνουμε την κατάλληλη επιλογή του τρόπου αναπαράστασης ενός γραφήματος, είναι απαραίτητο να γνωρίζουμε αν θέλουμε να κερδίσουμε σε χώρο αποθήκευσης ή σε χρόνο εκτέλεσης.

[2] $\Theta(g(n)) = \{f(n) :$ υπάρχουν θετικές σταθερές c_1, c_2 και n_0 έτσι ώστε $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ για κάθε $n \geq n_0$



Σχήμα 2.4: Πρόσθεση (1) και αφαίρεση (2) κορυφής από μια συνδεδεμένη λίστα.

2.2.1 Σύγκριση σε σχέση με τον χρόνο πρόσβασης στα δεδομένα και τον χώρο αποθήκευσης

Οι λειτουργίες που περιγράφονται παρακάτω χρησιμοποιούνται εκτενώς σε πολλούς αλγόριθμους για κατευθυνόμενα γραφήματα :

1. **βρες την ακμή** (u, v)

   Όταν χρησιμοποιούμε πίνακα συνδεσμολογίας, μπορούμε να βρούμε την ακμή (u, v) σε σταθερό χρόνο.
   Όταν χρησιμοποιούμε συνδεδεμένη λίστα, η χειρότερη περίπτωση είναι ό χρόνος εκτέλεσης να είναι $O(|A(v)|)$, αφού $|A(v)|$ είναι το μήκος της συνδεδεμένης λίστας που περιέχει τους γείτονες του v, δηλαδή $|A(v)| = deg(v)$. Το μέγιστο αυτού είναι $O(|V|)$.

2. **απαρίθμησε όλες τις κορυφές**

   Για να απαριθμήσουμε τις ακμές σε ένα γράφημα που έχει αναπαρασταθεί με πίνακα συνδεσμολογίας πρέπει να εξετάσουμε όλα τα στοιχεία του πίνακα, $|V|^2$. Άρα η χειρότερη περίπτωση είναι ό χρόνος εύρεσης όλων των κορυφών να είναι $O(|V|^2)$.
   Ενώ, αν χρησιμοποιήσουμε συνδεδεμένη λίστα, πρέπει να εξετάσουμε $|V|$ λίστες. Σε όλες υπάρχουν $|E|$ ακμές. Άρα στη χειρότερη περίπτωση χρειαζόμαστε χρόνο $O(|V| + |E|)$, που μπορεί να είναι σημαντικά μικρότερο από $O(|V|^2)$.

3. **απαρίθμησε τις εξερχόμενες ακμές από την** v

   Για να απαριθμήσουμε τις ακμές που εξέρχονται από την κορυφή v, πρέπει να εξετασουμε την $v-$οστη γραμμή του πίνακα, άρα χρειάζεται $O(|V|)$.

   Η απαρίθμηση των ακμών που ξεκινούν από την κορυφή v με συνδεδεμένη λίστα ολοκληρώνεται σε χρόνο $O(|A(v)|)$, αφού το μόνο που

χρειάζεται είναι να διασχίσουμε την $v-$οστή λίστα και να δούμε τα περιεχόμενα της.

4. **απαρίθμησε τις εισερχόμενες ακμές στην** v

   Για να απαριθμήσουμε τις ακμές που εισέρχονται από την κορυφή v, πρέπει να εξετασουμε την $v-$οστή στήλη του πίνακα, άρα χρειάζεται $O(|V|)$.

   Αν χρησιμοποιούνται συνδεδεμένες λίστες πρέπει να διατρέξουμε όλες τις λίστες και να εντοπίσουμε στις λίστες ποιον κορυφών υπάρχει η κορυφή v. Άρα ο χρόνος που χρειάζεται είναι $O(|V| + |E|)$.

	Τρόπος Αναπαράστασης	
Λειτουργία	Συνδεδεμένη λίστα	Πίνακας συνδεσμολογίας
Βρες την (u,v)	O(1)	O (| A (u) |)
Απαρίθμηση όλων των ακμών	O(|V|^2)	O(|V| + |E|)
Απαρίθμηση όλων των ακμών που ξεκινούν από το v	O(|V|)	O (| A (u) |)
Απαρίθμηση όλων των ακμών που καταλήγουν στο w	O(|V|)	O(|V| + |E|)

Σχήμα 2.5: Σύγκριση λίστας και πίνακα συνδεσμολογίας σε σχέση με τον χρόνο πρόσβασης στα δεδομένα.

Παρατηρούμε ότι αν το γράφημα δεν έχει βάρη, στην αναπαράσταση με πίνακα αντί να χρησιμοποιούμε μία λέξη μνήμης για κάθε στοιχείο του πίνακα, χρειαζόμαστε μόνο ένα bit.

Αν υποθέσουμε ότι τα ονόματα όλων των ακμών και των κορυφών χρειάζονται τον ίδιο χώρο αποθήκευσης, τότε μια συνδεδεμένη λίστα χρειάζεται πιθανώς λιγότερο χώρο αποθήκευσης απ'οτι ένας πίνακας συνδεσμολογίας όταν ισχύει η παρακάτω σχέση :

$$|E| < \frac{|V|^2 - 2|V|}{2}.$$

Για παράδειγμα, αν ένα γράφημα έχει 10 κορυφές, τότε συμφέρει από άποψη χώρου αποθήκευσης να το αναπαραστήσουμε με λίστα συνδεσμολογίας, αν

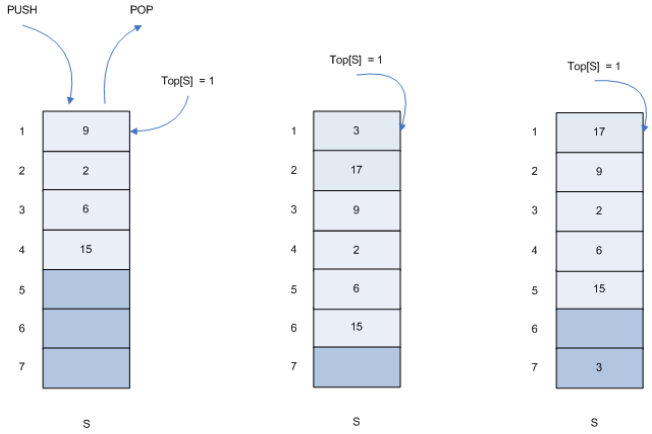
έχει λιγότερες από $\frac{10^2-2\cdot 10}{2} = 40$ ακμές.

2.3 Δομές Δεδομένων

Παρακάτω θα περιγράψουμε τις δομές δεδομένων που χρησιμοποιήσαμε για την υλοποίηση των αλγορίθμων Depth-First-Search, Breadth-First-Search, Maximum Flow τους οποίους περιγράφουμε παρακάτω.

2.3.1 Στοίβα

Μια στοίβα S είναι ένα δυναμικό σύνολο στο οποίο μπορούμε να αφαιρούμε στοιχεία με προκαθορισμένο τρόπο. Στη στοίβα αφαιρείται το στοιχείο που



Σχήμα 2.6: Στοίβα.

προστέθηκε πιο πρόσφατα. Λειτουργεί με την μέθοδο εξυπηρέτησης $LIFO$ (last in - first out). Η λειτουργία προσθήκης ενός στοιχείου ονομάζεται PUSH, ενώ η διαγραφή ονομάζεται POP.

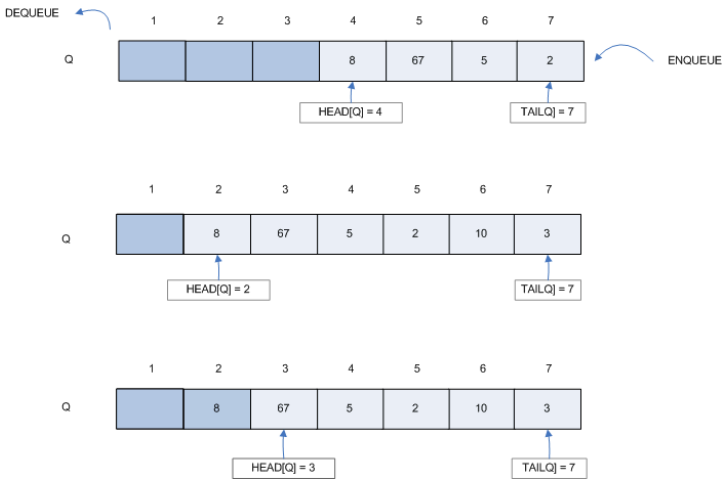
Επίσης μπορούμε να βλέπουμε ποιο στοιχείο βρίσκεται στην κεφαλή με την λειτουργία TOP. Όταν η στοίβα είναι άδεια $Top[S] = NULL$.

Τα στοιχεία της βρίσκονται μόνο στις ανοιχτόχρωμες θέσεις. Στην αρχή η στοίβα έχει 4 στοιχεία και $Top[S] = 9$. Όταν καλέσουμε τις λειτουργίες $PUSH(S, 17)$ και $PUSH(S, 3)$ η στοίβα έχει την δεύτερη μορφή. Έπειτα καλούμε την λειτουργία $POP(S)$ και η στοίβα αφαιρεί το στοιχείο που προστέθηκε πιο πρόσφατα και βρίσκεται στην κεφαλή, δηλαδή το 3. Αν και το 3 εμφανίζεται στο διάνυσμα, δεν ανήκει πλέον στη στοίβα και στην κεφαλή της βρίσκεται το 17.

2.3.2 Ουρά

Μια ουρά Q είναι επίσης ένα δυναμικό σύνολο στο οποίο μπορούμε να αφαιρούμε στοιχεία με προκαθορισμένο τρόπο. Στην ουρά αφαιρείται το

στοιχείο είχε προστεθεί παλαιότερα. Λειτουργεί με την μέθοδο εξυπηρέτησης $FIFO$ (first in - first out), δηλαδή ακριβώς όπως εξυπηρετούνται οι πελάτες σε μια συνηθισμένη ούρα τράπεζας. Η λειτουργία προσθήκης ενός στοιχείου ονομάζεται ENQUEUE, ενώ η διαγραφή ονομάζεται DEQUEUE. Η λίστα είναι άδεια όταν το τελευταίο της στοιχείο ταυτίζεται με το πρώτο, δηλαδή όταν $head[Q] = tail[Q]$.



Σχήμα 2.7: Ουρά.

Η παραπάνω ουρά έχει αρχικά 4 στοιχεία. Καλούμε την $ENQUEUE(Q, 10)$ και $ENQUEUE(Q, 3)$ και παίρνουμε την δεύτερη μορφή. Η τελευταία μορφή της ουράς είναι αποτέλεσμα της $DEQUEUE(Q)$.

Κεφάλαιο 3

Διασχίζοντας ένα γράφημα

Το να διασχίσει κανείς ένα γράφημα σημαίνει να επισκεφθεί όλες τις κορυφές του με συστηματικό τρόπο. Θα δούμε δύο ευρέως γνωστούς τρόπους παρακάτω, τον Depth-First-Search και τον Breadth-First-Search. Και οι δύο μετά το τέλος τους δημιουργούν επικαλύπτοντα δέντρα (βλέπε Ορισμό 11) με ιδιότητες χρήσιμες στους αλγόριθμους για γραφήματα.

3.1 Depth-First-Search

Έστω ότι βρισκόμαστε στην κορυφή v ενός μη-κατευθυνόμενου γραφήματος. Το επόμενο βήμα είναι να επισκεφθούμε μια κορυφή που συνδέεται με την v και που δεν την έχουμε ξαναεπισκεφθεί. Αν δεν υπάρχει τέτοια κορυφή, επιστρέφουμε στην κορυφή που βρισκόμασταν αμέσως πριν πάμε στην v. Αυτό το βήμα επαναλαμβάνεται μέχρι να περάσουμε από όλες τις κορυφές της συνιστώσας στην οποία βρισκόμαστε. Αυτός ο αλγόριθμος δεν επιτρέπει να επισκεφθούμε πάνω από μια φορά μια κορυφή, εκτός αν χρησιμοποιήσουμε ακμές από τις οποίες έχουμε ξαναπεράσει προηγουμένως. Οι ακμές που διατρέχουμε ακολουθώντας τον DFS σχηματίζουν ένα επικαλύπτον δέντρο για κάθε συνεκτική συνιστώσα του γραφήματος. Το σύνολο των δέντρων αυτών ονομάζεται $DFS-$ επικαλύπτον δάσος F.

3.1.1 Αλγόριθμος

1. $i \leftarrow 1$
2. $F \leftarrow \emptyset$
3. **για κάθε** $v \in V\{$
4. $\quad DFI(v) \leftarrow 0$
5. $\quad$ **όσο** υπάρχουν u με $DFI(v) = 0\{$

6. $DFS(v)\}$
7. έξοδος F
8. διαδικασία $DFS(v)\{$
9. $DFI(v) \leftarrow i$
10. $i = i + 1$
11. **για κάθε** $v' \in A(v)\{$
12. **αν** $DFI(v') = 0\{$
13. $F \leftarrow F \cup \{(v, v')\}$
14. $DFS(v')$
15. $\}$
16. $\}$

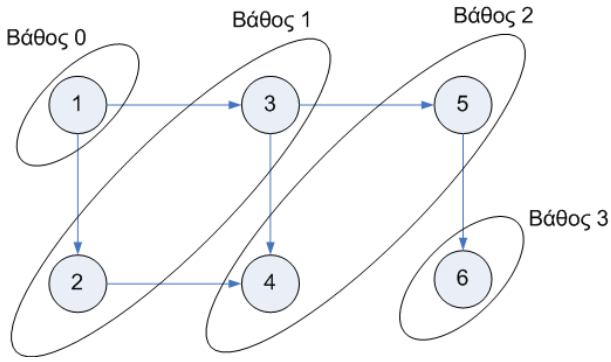
Το πρόγραμμα αυτό παίρνει σαν είσοδο την λίστα συνδεσμολογίας κάθε ακμής του γραφήματος G. Η έξοδος είναι το σύνολο ακμών F. Χρησιμοποιούμε το διάνυσμα $DFI(v)(Depth - First - Index)$ το οποίο αρχικά είναι 0 για κάθε κορυφή και μετά το τέλος του προγράμματος η σειρά με την επισκεφθήκαμε κάθε κορυφή v.
Σχετικά με το βήμα στη σειρά 11, θα πρέπει να προσδιορίσουμε με ποιον τρόπο επιλέγουμε κάθε φορά την v'. Συνήθως η απλούστερη μέθοδος είναι να επιλέγουμε τις κορυφές με την σειρά που είναι αποθηκευμένες στην συνδεδεμένη λίστα.

3.1.2 Πολυπλοκότητα

Για κάθε $v \in V$ η διαδικασία $DFS(v)$ καλείται μόνο μία φορά γιατί μετά την πρώτη εκτέλεση, $DFS(v) = 0$. Αν εξαιρέσουμε τις αναδρομικές κλήσεις της διαδικασίας $DFS(v)$, ο χρόνος που ξοδεύεται από την $DFS(v)$ είναι ανάλογος με το πλήθος των ακμών που συμπίπτουν στην v ή αν το γράφημα είναι κατευθυνόμενο, στο πλήθος των ακμών που ξεκινούν από την v. Άρα οι κλήσεις της $DFS(v)$ χρειάζονται συνολικό χρόνο ανάλογο με $|E|$. Από την άλλη η σειρά 3 απαιτεί $O(|V|)$, καθώς ελέγχει για περισσότερες από μία συνεκτικές συνιστώσες. Η σειρά 6 απαιτεί $O(|E|)$ βήματα. Συμπερασματικά η πολυπλοκότητα του αλγορίθμου είναι $O(max(|V|, |E|))$

3.2 Breadth-First-Search

Με αυτόν τον αλγόριθμο επισκεπτόμαστε πρώτα τις κορυφές με βάθος 0 (π.χ δηλαδή την ρίζα αν μιλάμε για δέντρο), έπειτα αυτές με βάθος 1 (γείτονες της ρίζας) κ.ο.κ. Επειδή αναφερόμαστε σε γράφημα είναι απαραίτητο στην αρχή να επιλέγουμε την κορυφή από την οποία θα ξεκινήσει ο αλγόριθμος.



3.2.1 Αλγόριθμος

1. **για κάθε** $v \in V\{$
2. $BFI(v) \leftarrow 0$
3. $i \leftarrow 1$
4. $BFI(u) \leftarrow i$
5. $S \leftarrow \emptyset$
6. πρόσθεσε το u στην ουρά
7. **όσο** η ουρά δεν είναι άδεια $\{$
8. αφαίρεσε μία κορυφή από την ουρά, ονόμασε την w
9. **για κάθε** $v \in A(w)\{$
10. **αν** $BFI(v) = 0\{$
11. $BFS(v) = i + 1$
12. $i = i + 1$
13. πρόσθεσε το v στην ουρά
14. πρόσθεσε την (w, v) στο S
15. $\}$

16. }

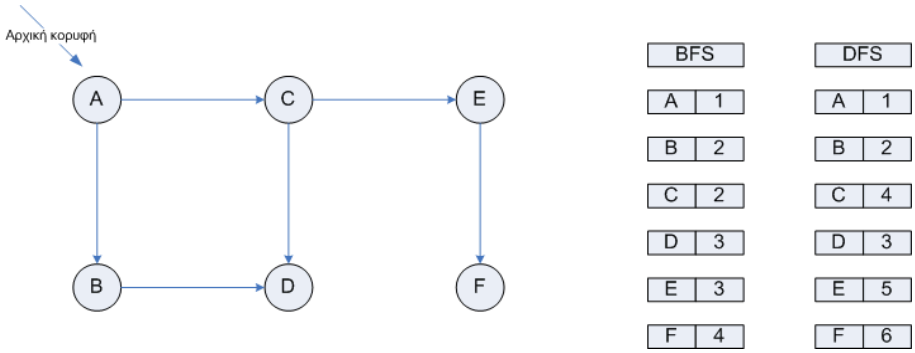
17. έξοδος S

18. }

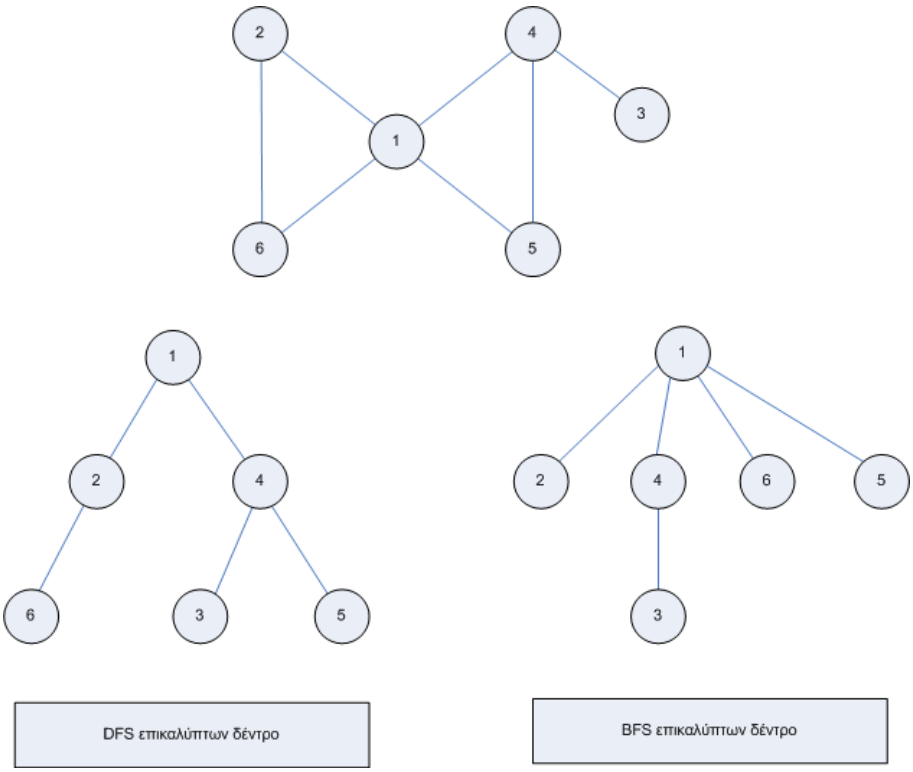
(S είναι το σύνολο των ακμών που αποτελούν το $BFS-$επικαλύπτον δέντρο.)

3.2.2 Πολυπλοκότητα

Η ανάλυση είναι όμοια με την ανάλυση του Depth-First-Search και η πολυπλοκότητα είναι επίσης $O(max(|V|, |E|))$.



Σχήμα 3.1: Σειρά με την οποία επισκεπτόμαστε τις κορυφές του γραφήματος με τους αλγόριθμους BFS και DFS.



Σχήμα 3.2: BFS και DFS επικαλύπτοντα δέντρα.

Κεφάλαιο 4

Εύρεση συντομότερου μονοπατιού. Ο αλγόριθμος του Dijkstra

Ο αλγόριθμος του Dijkstra βρίσκει το συντομότερο μονοπάτι από μία κορυφή σε μία άλλη (ή σε όλες τις άλλες) σε ένα γράφημα με βάρη, κατευθυνόμενο ή μη, με n κορυφές. Εάν το γράφημα είναι μη κατευθυνόμενο, αντικαθιστούμε κάθε ακμή (u, v) με δύο άλλες κατευθυνόμενες (u, v) και (v, u). Κάθε κορυφή (v) ακολουθείται από ένα αριθμό $L(v)$. Αρχικά $L(v) = w((u, v))$, όπου $w((u, v))$ είναι το βάρος της ακμής (u, v) και (u) είναι η κορυφή από την οποία ξεκινάει το μονοπάτι. Αν $u \neq v$ και $(u, v) \notin E$ τότε $w((u, v)) = \infty$ και $w((u, v)) = 0$. Κατά τον τερματισμό του αλγορίθμου το $L(v)$ για κάθε $v \in V$ είναι ίσο με το μήκος του συντομότερου μονοπατιού από το (u) στο (v). Ο αλγόριθμος λειτουργεί κατασκευάζοντας ένα σύνολο $T \subseteq V$. Κάθε στιγμή το συντομότερο μονοπάτι από το (u) στο (v) περιέχει ακμές μόνο από το T, έτσι ώστε όταν $T = V$ ο αλγόριθμος σταματάει.

4.1 Αλγόριθμος

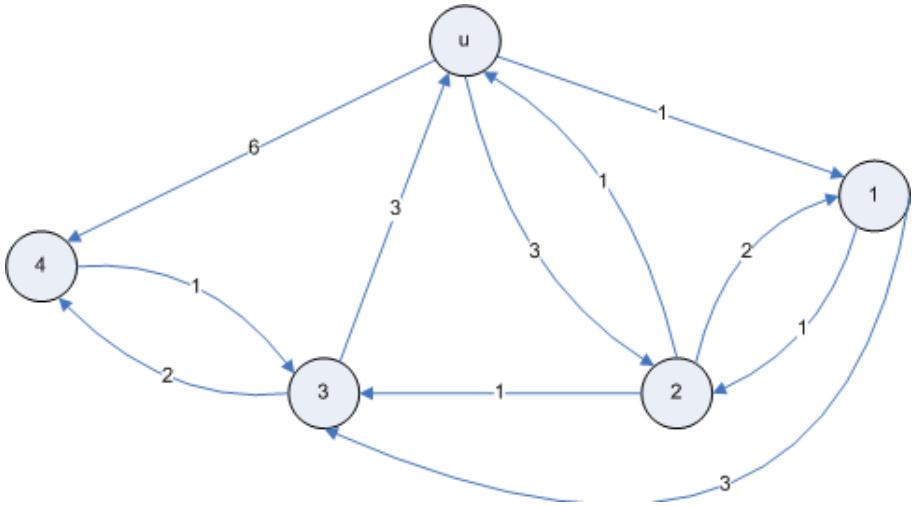
1. **για κάθε** $v \neq u, \quad L(v) \leftarrow w((u, v))$
2. $L(u) = 0$
3. $T \leftarrow \{u\}$
4. **όσο** $T \neq V\{$
5. βρες $v' \notin T$ τέτοια ώστε για κάθε $v \notin T L(v') \leq L(v)$
6. $T \leftarrow T \cup \{u'\}$
7. **για κάθε** $v \notin T$

8. **αν** $L(v) > L(v') + w((v', v))$

9. τότε $L(v) = L(v') + w((v', v))$

4.2 Η εκτέλεση του αλγορίθμου

Στον πίνακα που ακολουθεί βλέπουμε τα βήματα του αλγορίθμου του Dijkstra για το γράφημα 4.1.



Σχήμα 4.1: Dijkstra.

Επανάληψη	v΄	L(u)	L(1)	L(2)	L(3)	L(4)	T
0	-	0	1	3	oo	6	{u}
1	1	0	1	2	4	6	{u,1}
2	2	0	1	2	3	6	{u,1,2}
3	3	0	1	2	3	5	{u,1,2,3}
4	4	0	1	2	3	5	V

Σχήμα 4.2: Εκτέλεση του Dijkstra.

4.3 Η εγκυρότητα του αλγορίθμου

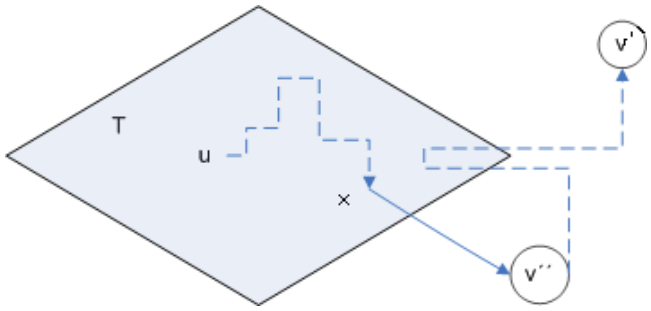
Ο αλγόριθμος του Dijkstra βρίσκει το συντομότερο μονοπάτι από την κορυφή u προς κάθε άλλη κορυφή του γραφήματος.

Απόδειξη. Πρώτα αποδεικνύουμε με επαγωγή ως προς το μέγεθος του T ότι :

1. για κάθε $v \in T$, το $L(v)$ είναι ίσο με το μήκος του συντομότερου μονοπατιού από το u στο v, και

2. για κάθε $v \notin T$, το $L(v)$ είναι ίσο με το μήκος του συντομότερου μονοπατιού από το u στο v, το οποίο εκτός από την κορυφή v, περνάει μόνο από κορυφές στο Τ.

Στην γραμμή 3 βλέπουμε ότι $|T| = 1$ και ότι οι γραμμές 1 και 2 ικανοποιούν τα (α) και (β). Στην σειρά 6 βρίσκεται το επαγωγικό βήμα, όπου η v' προστίθεται στο Τ και έχει την μικρότερη απόσταση από το u σε σχέση με τις υπόλοιπες κορυφές που δεν ανήκουν ακόμα στο Τ.
Υποθέτουμε ότι πριν το v' προστεθεί στο Τ, το $L(v')$ είναι ίσο με το συντομότερο μονοπάτι από το u στο v', το οποίο εκτός από την v', περνάει μόνο από κορυφές στο Τ.
Έστω ότι όταν το v' προστίθεται στο Τ, το *$L(v')$ δεν είναι ίσο με το συντομότερο μονοπάτι από το u στο v'.*



Άρα το μονοπάτι αυτό πρέπει να περιέχει τουλάχιστον μία κορυφή που δεν ανήκει στο Τ. Έστω ότι η v'' είναι η πρώτη τέτοια κορυφή που συναντάμε όταν διατρέχουμε αυτό το μονοπάτι από το u.
Τότε το μήκος αυτό του μονοπατιού από το u στο v'' (το οποίο είναι το συντομότερο μονοπάτι από το u στο v'' και περιέχει κορυφές μόνο του Τ εκτός από την v'' - αλλιώς θα υπήρχε ακόμα πιο σύντομο) είναι μικρότερο από $L(v')$. Από την επαγωγική υπόθεση το $L(v'')$ είναι η απόσταση από το u στο v'' άρα $L(v') > L(v'')$ τη στιγμή που η κορυφή v' προστίθεται στο Τ. Αυτό είναι άτοπο, γιατί στην γραμμή 5 του αλγορίθμου ψάχνουμε v' τέτοιο ώστε $L(v') \leq L(v'')$, άρα δεν υπάρχει πιο σύντομο μονοπάτι από το $L(v')$ όταν το v' προστίθεται στο Τ. Άρα το (α) ικανοποιείται και το (β) από την γραμμή 7 . □

4.4 Πολυπλοκότητα

Θα δούμε ότι ο αλγόριθμος του Dijkstra τρέχει σε χρόνο $O(|V|^2)$. Στην γραμμή 5 για να αποφασίσουμε ποια κορυφή θα επιλέξουμε χρειάζονται $O(|V|)$ συγκρίσεις, ενώ για την γραμμή 7 απαιτούνται όχι περισσότερες από $|V|$ αναθέσεις. Οι γραμμές 5 και 7 περιλαμβάνονται στο "όσο" που ξεκινάει από την γραμμή 4 και τελειώνει στην τελευταία γραμμή, το οποίο εκτελείται $|V| - 1$ φορές. Άρα καταλήξαμε ότι ο αλγόριθμος του Dijkstra τρέχει σε χρόνο $O(|V|^2)$.

Αν θέλουμε να βρούμε το συντομότερο μονοπάτι από μια συγκεκριμένη κορυφή α σε μια άλλη β, τότε στη γραμμή 4 πρέπει να γίνεται ο έλεγχος αν η κορυφή β ανήκει στο Τ.

Κεφάλαιο 5

Ροή σε Δίκτυα

5.1 Προβλήματα Δικτυακής Ροής - Εφαρμογές

Υπάρχουν πολλά προβλήματα που λύνονται εύκολα εάν τα μοντελοποιήσουμε όχι σαν απλά γραφήματα αλλά σαν δίκτυα.

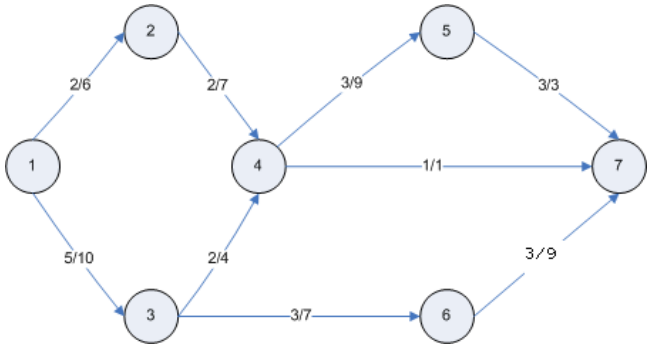
Θα ξεκινήσουμε με ένα απλό παράδειγμα. Έστω ότι έχουμε ένα δίκτυο ύδρευσης το οποίο αποτελείται από σωλήνες διαφορετικών διατομών, συνδεδεμένων μεταξύ τους μέσα στους οποίους ρέει νερό. Σε κάθε σημείο ένωσης δύο ή περισσότερων σωλήνων υπάρχουν διακόπτες οι οποίοι ρυθμίζουν την ροή του νερού σε κάθε έναν από τους σωλήνες.
Είναι προφανές ότι αλλάζοντας των συνδυασμό ρυθμίσεων στους διακόπτες, μπορούμε να πετύχουμε διαφορετική ροή στο δίκτυο, π.χ. 10^6 κυβικά εκατοστά νερού. Αν κλείσουμε όλους τους διακόπτες έχουμε μηδενική ροή, ενώ αν ανοίξουμε κάποιους η ροή στο δίκτυο αλλάζει. Ένα πρώτο ερώτημα που προκύπτει, το οποίο θα ορίσουμε αυστηρά παρακάτω και ονομάζεται Πρόβλημα Μέγιστης Ροής, είναι το εξής : ¨Ποια είναι η μέγιστη ροή που μπορούμε να επιτύχουμε από ένα δοσμένο σημείο του δικτύου προς ένα άλλο; ¨.

5.2 Παράσταση δικτύου σε μορφή γραφήματος

Ένα δίκτυο μπορεί να παρασταθεί με ένα γράφημα με συγκεκριμένα χαρακτηριστικά. Στο παράδειγμα που αναφέραμε παραπάνω κάθε σημείο ένωσης των σωλήνων είναι μία κορυφή του γραφήματος που παριστά αυτό το δίκτυο και κάθε σωλήνας μία ακμή με βάρος την χωρητικότητα του, δηλαδή με τον όγκο νερού που μπορεί να περάσει από μία κάθετη διατομή του σωλήνα ανά μονάδα χρόνου με σταθερές συνθήκες.
Το γράφημα που παριστά ένα δίκτυο πληρεί τις παρακάτω προϋποθέσεις:

1. Είναι συνεκτικό κατευθυνόμενο γράφημα με βάρη μη αρνητικούς πραγματικούς αριθμούς.
2. Δεν έχει βρόγχους.
3. Έχει δύο ¨ειδικές¨ κορυφές, μία με μόνο εξερχόμενες ακμές, την **πηγή** s (source) και μία μόνο με εισερχόμενες ακμές, την **δεξαμενή** t (sink).
4. Ισχύει ο *Νόμος του Kirchhoff*, δηλαδή ο ρυθμός με τον οποίο εισέρχεται ένα αγαθό σε μία κορυφή πρέπει να είναι ίσος με τον ρυθμό με τον οποίο εξέρχεται από την ίδια κορυφή.

Το βάρος κάθε ακμής είναι μη αρνητικό και λέγεται **χωρητικότητα** (capacity) $c(u,v) \geq 0$. Αν $(u,v) \notin E$, τότε $c(u,v) = 0$. *χωρητικότητα*



Σχήμα 5.1: Δίκτυο (ροή / χωρητικότητα) .

Ορισμός 12. *Ο ρυθμός με τον οποίο ρέουν τα αγαθά σε ένα δίκτυο $G = (V, E)$, λέγεται **ροή** f (flow) και μπορεί να παρασταθεί από ένα ξεχωριστό σύνολο από βάρη, τέτοιο ώστε η ροή σε κάθε ακμή να είναι μικρότερη ή ίση από την χωρητικότητα της, $f(u,v) \leq c(u,v)$.* ***ροή***
Στο Σχήμα 5.1 βλέπουμε ένα δίκτυο, όπου η ροή κάθε ακμής είναι ο αριθμός αριστερά, ενώ δεξιά φαίνεται η χωρητικότητα της. Πιο συγκεκριμένα, η ροή από μία κορυφή σε μία άλλη είναι μία πραγματική συνάρτηση $f : V \times V \to \mathbb{R}$ και ικανοποιεί τα παρακάτω :

1. $-\infty \leq f(u,v) \leq c(u,v)$
2. $f(u,v) = -f(v,u)$
3. *Για κάθε $v \in V \setminus \{s,t\}$ απαιτούμε $\sum_{u \in V} f(u,v) = 0$*

Ορισμός 13. *Η **ολική ροή** ενός δικτύου είναι το άθροισμα των ροών από κάθε κορυφή του γραφήματος προς την δεξαμενή και είναι ίση με* *ολική ροή*

$$|f| = \sum_{u \in V} f(u,t)$$

Αφού είδαμε τους απαραίτητους ορισμούς σχετικά με τα δίκτυα, θα δώσουμε έναν αυστηρό ορισμό του προβλήματος μέγιστης ροής.

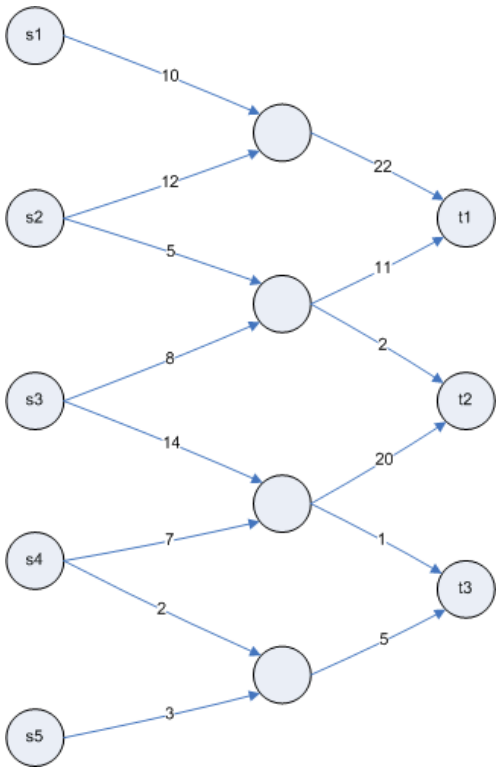
Πρόβλημα Μέγιστης Ροής

Ορισμός 14. *Η εύρεση της μεγαλύτερης ολικής ροής σε ένα δίκτυο με πηγή s και δεξαμενή t, ονομάζεται* ***Πρόβλημα Μέγιστης Ροής****.*

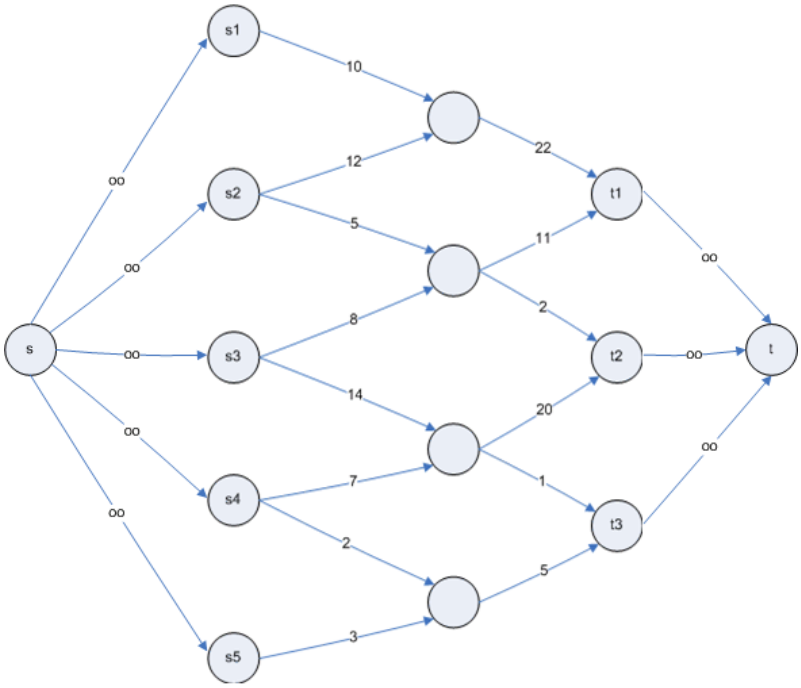
Θα θέλαμε δηλαδή να υπάρχει ένας "τρόπος" έτσι ώστε για κάθε δίκτυο που μας δίνεται να μπορεί να βρεθεί η κατάλληλη ροή σε κάθε ακμή του γραφήματος έτσι ώστε να μεγιστοποιηθεί η ολική ροή του δικτύου.

5.2.1 Δίκτυα με περισσότερες από μία πηγές ή δεξαμενές.

Ένα δίκτυο μπορεί να έχει περισσότερες από μία πηγές ή δεξαμενές. Μπορούμε εύκολα να αναγάγουμε αυτό το πρόβλημα στο προηγούμενο δημιουργώντας στο υπάρχον δίκτυο μια υπερπηγή s (supersource) και προσθέτοντας κατευθυνόμενες ακμές (s, s_i) από την υπερπηγή προς κάθε πηγή με χωρητικότητα $c(s, s_i) = \infty$. Αντίστοιχα, δημιουργούμε μία υπερδεξαμενή t (supersink) με τις ανάλογες ακμές από τις δεξαμενές προς την υπερδεξαμενή επίσης με άπειρη χωρητικότητα. Έτσι το πρόβλημα ανάγεται σε αυτό με ένα s και ένα t.



Σχήμα 5.2: Δίκτυο με πολλαπλές πηγές και δεξαμενές.



Σχήμα 5.3: Δίκτυο με μία πηγή και μία δεξαμενη.

5.2.2 Η ροή μεταξύ συνόλων από κορυφές

Μέχρι τώρα συναντήσαμε την συνάρτηση ροής $f(u, v)$ με u, v κορυφές. Παρακάτω θα δούμε την $f(X, y)$ όπου X σύνολο με κορυφές και y κορυφή και την $f(X, Y)$ με X, Y σύνολα κορυφών. Τότε η ροή $f(X, Y)$ θα είναι ίση με το άθροισμα όλων των ροών που προκύπτουν όταν αντικαταστήσουμε τα σύνολα X, Y με όλα τα στοιχεία τους.

$$f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y)$$

Ομοίως, η χωρητικότητα $c(X, Y)$ ορίζεται να είναι ίση με το άθροισμα όλων των χωρητικοτήτων που προκύπτουν όταν αντικαταστήσουμε τα σύνολα X, Y με όλα τα στοιχεία τους.

$$c(X, Y) = \sum_{x \in X} \sum_{y \in Y} c(x, y)$$

Επίσης, ο συμβολισμός $V - s$ σηματοδοτεί το σύνολο $V \setminus \{s\}$.

Λήμμα 1. *Αν $G(V, E)$ είναι ένα δίκτυο και f είναι μία ροή του, τότε για $X, Y, Z \subseteq V$ με $X \cap Y = \emptyset$, ισχύουν τα παρακάτω:*

1. $f(X, X) = 0$
2. $f(X, Y) = -f(Y, X)$
3. $f(X \cup Y, Z) = f(X, Z) + f(Y, Z)$ *και*
4. $f(Z, X \cup Y) = f(Z, X) + f(Z, Y)$

Απόδειξη. 1.

$$f(X, X) = \sum_{x_1 \in Q} \sum_{x_2 \in Q} f(x_1, x_2)$$

Αν $x_1 = x_2$ τότε $f(x_1, x_2) = 0$.

Αν $x_1 \neq x_2$ τότε επειδή $f(x_1, x_2) = -f(x_2, x_1)$, στο άθροισμα θα εμφανίζονται η ροή $f(x_1, x_2)$ και η $f(x_2, x_1)$ οι οποίες θα αλληλοαναιρούνται αφού έχουν αντίθετες τιμές.

2.

$$f(X, Y) = \sum_{x \in X} \sum_{y \in Y} f(x, y) = \sum_{x \in X} \sum_{y \in Y} -f(y, x) = -\sum_{x \in X} \sum_{y \in Y} f(y, x) = -f(y, x)$$

3.

$$\begin{aligned} f(X \cup Y, Z) &= \sum_{a \in X \cup Y} \sum_{b \in Z} f(a, b) \\ &= (\sum_{a \in X} + \sum_{a \in Y}) \sum_{b \in Z} f(a, b) \qquad Q, Ux'enametax'utouc \\ &= \sum_{a \in X} \sum_{b \in Z} f(a, b) + \sum_{a \in Y}) \sum_{b \in Z} f(a, b) \\ &= f(X, Z) + f(Y, Z). \end{aligned}$$

4. Ομοίως με 3.

□

Τώρα με τη χρήση του Λήμματος 1 μπορούμε να αποδείξουμε ότι η ολική ροή είναι ίση με την ροή από την πηγή προς όλες τις κορυφές του γραφήματος.

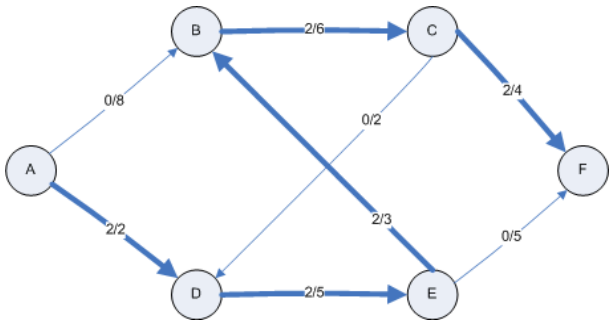
$$\begin{aligned} |f| &= f(V, t) \\ &= f(V, V) - f(V, V - t) && (Ιδιοτητα3) \\ &= -f(V, V - t) && (Ιδιοτητα1) \\ &= f(V - t, V) && (Ιδιοτητα2) \\ &= f(s, V) + f(V - t - s, V) && (Ιδιοτητα3) \\ &= f(s, V) \end{aligned}$$

5.2.3 Η Μέθοδος Ford-Fulkerson

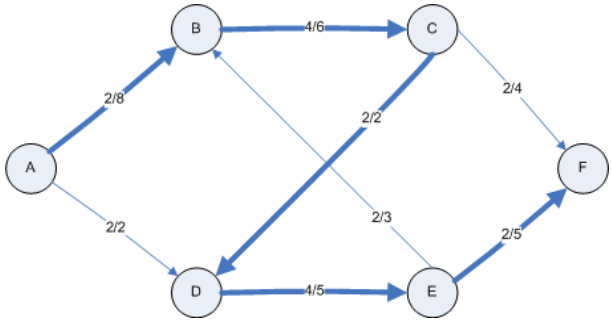
Εισαγωγή - Βασικές έννοιες

Η *Μέθοδος Ford-Fulkerson* χρησιμοποιείται για την λύση του προβλήματος μέγιστης ροής. Υπάρχουν διάφοροι αλγόριθμοι που υλοποιούν την παραπάνω γενική μέθοδο με διαφορετικούς τρόπους.

Η μέθοδος βασίζεται σε τρεις έννοιες : το **υπολοίπον δίκτυο, τα επαυξάνοντα μονοπάτια** και τις **τομές**.
Αυτές οι έννοιες χρησιμοποιούνται στο πολύ σημαντικό Θεώρημα 1, το οποίο συνδέει την μέγιστη ολική ροή σε ένα δίκτυο, με τη ροή κατά μήκος μίας τομής στο δίκτυο αυτό και με το ενδεχόμενο το υπολοίπον δίκτυο ενός δικτύου να μην έχει κανένα επαυξάνον μονοπάτι.

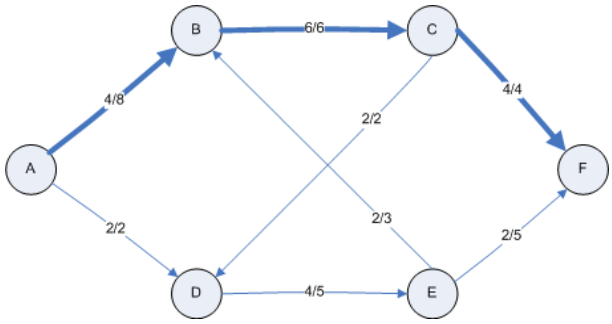


Σχήμα 5.4: Αύξηση ροής κατά μήκος του μονοπατιού ADEBCF .



Σχήμα 5.5: Αύξηση ροής κατά μήκος του μονοπατιού ABCDEF .

Η μέθοδος αυτή είναι επαναληπτική. Ο σκοπός είναι να μεγιστοποιηθεί η συνολική ροή του δικτύου. Έτσι ξεκινάμε με μηδενική ροή και σε κάθε βήμα της μεθόδου αυξάνουμε την συνολική ροή βρίσκοντας ένα επαυξάνον μονοπάτι το οποίο ουσιαστικά είναι ένα μονοπάτι από την πηγή προς την δεξαμενή, κατά μήκος του οποίου μπορούμε να αυξήσουμε την ροή, τηρώντας



Σχήμα 5.6: Αύξηση ροής κατά μήκος του μονοπατιού ABCF .

πάντα τους κανόνες σχετικά με τα δίκτυα. Επαναλαμβάνουμε μέχρι να μην μπορούμε να βρούμε άλλο επαυξάνον μονοπάτι στο δίκτυο. Τέλος θα χρησιμοποιήσουμε το θεώρημα 1 για να δείξουμε ότι μετά το τέλος της μεθόδου θα έχουμε πράγματι βρει την μέγιστη συνολική ροή.
Τα Σχήματα 5.4, 5.5 και 5.6 δείχνουν την σταδιακή αυξηση της ροής σε ένα δίκτυο στο οποίο εφαρμόζεται η Μέθοδος Ford-Fulkerson.

Η γενική περιγραφή της μεθόδου είναι η ακόλουθη ($G(V, E)$ είναι ένα γράφημα, s, t οι κορυφές του).
Ford-Fulkerson-Method$(G(V, E), s, t)$

1. αρχικοποίηση ροής σε κάθε ακμή του δικτύου $f(u, v) = 0$ για κάθε $u, v \in V$
2. όσο υπάρχει επαυξάνον μονοπάτι p {
3. αύξηση της ροής f κατά μήκος του μονοπατιού p }
4. επιστρέφει την ροή f (η οποία είναι η μέγιστη)

Υπολοίποντα δίκτυα

Διαισθητικά ένα υπολοίπον δίκτυο αποτελείται από εκείνες τις ακμές ενός δικτύου, οι οποίες επιτρέπουν μεγαλύτερη ροή.

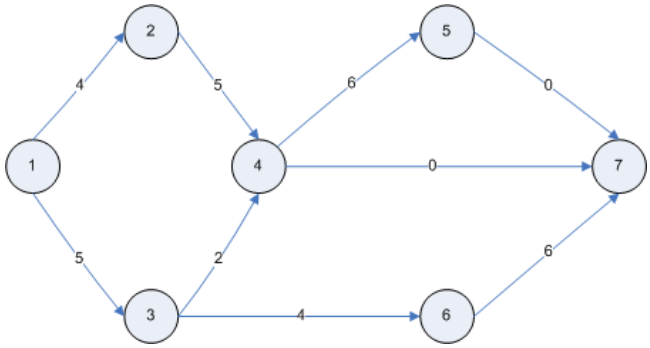
αχρησιμοποίητη χωρητικότητα ακμής

Ορισμός 15. *Δεδομένου ενός δικτύου $G(V, E)$ και μιας ακμής $(u, v) \in G$, ορίζουμε σαν **αχρησιμοποίητη χωρητικότητα** της ακμής (u, v) τη ροή εκείνη που μπορεί να προστεθεί προχωρώντας από το u στο v και υπακούοντας στον κανόνα που απαγορεύει η ροή να είναι μεγαλύτερη από την αντίστοιχη χωρητικότητα $c(u, v)$:*

$$c_f(u, v) = c(u, v) - f(u, v)$$

Δείτε Σχήμα 5.7.

Για παράδειγμα αν $c(u,v) = 12$, $f(u,v) = 8$ τότε $c_f(u,v) = 12 - 8 = 4$.



Σχήμα 5.7: Αχρησιμοποίητη χωρητικότητα του δικτύου.

Είπαμε προηγουμένως ότι η μέθοδος Ford-Fulkerson χρησιμοποιεί τα επαυξάνοντα μονοπάτια ενός δικτύου, δηλαδή τα μονοπάτια κατά μήκος των οποίων μπορεί να αυξηθεί η ροή. Για να βρούμε ένα τέτοιο μονοπάτι μπορούμε να διατρέξουμε μία ακμή για παράδειγμα από το u στο v, σύμφωνα με την κατεύθυνση που ήδη έχει, οπότε και η ροή της θα είναι $f(u,v)$ ή να διατρέξουμε την ακμή αυτή με αντίθετη φορά, και η ροή σε αυτήν την περίπτωση θα είναι $f(v,u) = -f(u,v)$.

Σε περίπτωση που η ροή κατά μήκος μιας ακμής έχει αρνητική τιμή, η αχρησιμοποίητη χωρητικότητα είναι μεγαλύτερη από την χωρητικότητα της ακμής, για παράδειγμα αν $c(u,v) = 12$, $f(u,v) = -8$ τότε $c_f(u,v) = 12 - (-8) = 20$.

Ορισμός 16. *Το **υπολοίπου γράφημα** ενός γραφήματος G που επάγεται από την ροή f είναι το $G_f(V, E_f)$ όπου* *υπολοίπου γράφημα*

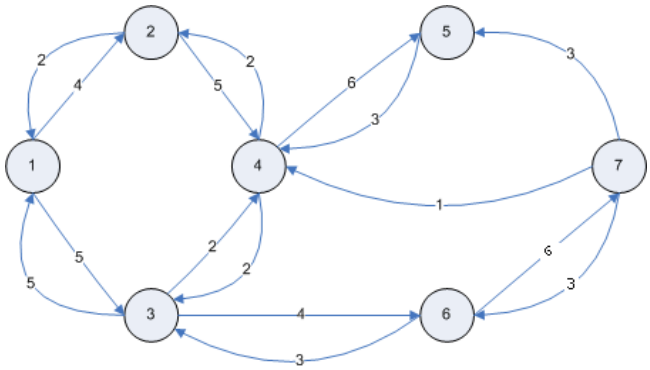
$$E_f = \{(u,v) \in V \times V : c_f(u,v) > 0\}$$

Δείτε Σχήμα 5.8.

Σημειώνουμε ότι η (u,v), δηλαδή η ακμή από το u στο v, μπορεί να είναι ακμή στο G_f ακόμα κι αν δεν ήταν η ακμή στο G. Αυτό μπορεί να συμβεί αν η (v,u), δηλαδή η ακμή από το v στο u ήταν ακμή στο G. Δηλαδή αν το πρώτο γράφημα G περιέχει την ακμή (u,v), το υπολοίπον γράφημα G_f μπορεί να περιέχει και την ακμή (u,v) αλλά και την (v,u). Δηλαδή, το G_f μπορεί να έχει το πολύ δύο φορές όσες ακμές έχει το G.

$$|E_f| \leq 2|E|$$

Το G_f είναι από μόνο του ένα δίκτυο με χωρητικότητες που δίνονται από την



Σχήμα 5.8: Το Υπολοίπον Δίκτυο του δικτύου στο Σχήμα 5.1.

συνάρτηση χωρητικότητας c_f.
Το επόμενο λήμμα συνδέει την ροή στο πρώτο δίκτυο με την αχρησιμοποίητη ροή.

Λήμμα 2. *Έστω ένα δίκτυο $G(V, E)$ με πηγή s και δεξαμενή t και έστω f ροή στο G. Αν G_f είναι το υπολοίπον γράφημα που επάγεται από τη ροή f και αν f' ροή στο G_f, τότε το άθροισμα των ροών $f + f'$ αποτελεί ροή στο G με ολική ροή $|f + f'| = |f| + |f'|$.*

Απόδειξη. Πρώτα θα δείξουμε ότι ισχύουν οι ιδιότητες 1-3 του Ορισμού 12 για την $(f + f')$.

Σημειώνουμε εδώ ότι $f'(u, v) \leq c_f(u, v)$ για κάθε $u, v \in V$.

$$\begin{aligned}(f + f')(u, v) &= f(u, v) + f'(u, v) \\ &\leq f(u, v) + (c(u, v) - f(u, v)) \\ &= c(u, v).\end{aligned}$$

$$\begin{aligned}(f + f')(u, v) &= f(u, v) + f'(u, v) \\ &= -f(v, u) - f'(v, u) \\ &= -(f(v, u) + f'(v, u)) \\ &= -(f + f')(u, v).\end{aligned}$$

Για κάθε $u \in V \setminus \{s, t\}$,

$$\begin{aligned}\sum_{v \in V}(f + f')(u, v) &= \sum_{v \in V}(f(u, v) + f'(u, v)) \\ &= \sum_{v \in V} f(u, v) + \sum_{u \in V} f'(u, v) \\ &= 0 + 0 \\ &= 0.\end{aligned}$$

Τέλος έχουμε,

$$\begin{aligned} |f + f'| &= \sum_{v \in V} (f + f')(s, v) \\ &= \sum_{v \in V} (f(s, v) + f'(s, v)) \\ &= \sum_{v \in V} f(s, v) + \sum_{v \in V} f'(s, v) \\ &= |f| + |f'|. \end{aligned}$$

□

Επαυξάνοντα μονοπάτια

επαυξάνον μονοπάτι
αχρησιμοποίητη χωρητικότητα

Ένα **επαυξάνον μονοπάτι** p είναι ένα μονοπάτι από το s στο t σε ένα υπολοίπον δίκτυο. Ονομάζουμε το μεγαλύτερο ποσό αύξησης της δικτυακής ροής που μπορούμε να πετύχουμε κατά μήκος ενός αυξανόμενου μονοπατιού p, **αχρησιμοποίητη χωρητικότητα** του p και και δίνεται από την σχέση:

$$c_f(p) = min\{c_f(u, v) : (u, v) \in p\}$$

Λήμμα 3. *Έστω ένα δίκτυο $G(V, E)$ με f ροή στο G και p ένα επαυξάνον μονοπάτι στο G_f. Ορίζουμε την συνάρτηση $f_p : V \times V \to \mathbb{R}$*
$f_p(u, v) = c_f(p)$,αν (u, v) είναι στο p
$f_p(u, v) = -c_f(p)$,αν (v, u) είναι στο p
$f_p(u, v) = 0$, αλλιώς
Τότε η f_p είναι ροή στο G_f με $|f_p| = c_f(p) > 0$

Απόδειξη. Πρώτα θα δείξουμε ότι ισχύουν οι ιδιότητες 1-3 του Ορισμού 12 για την f_p .

1. Όταν οι ακμές (u, v) και (u, v)δεν είναι στο p τότε $f_p(u, v) = 0$, όταν το (u, v) είναι στο p τότε η $f_p(u, v) = c_f(p)$ και όταν το (v, u) είναι στο p, η $f_p(u, v) = -c_f(p)$. Άρα $f_p \leq c_f(p)$.

2.

$$\begin{aligned} f_p(u, v) &= min\{c_f(u, v) : (u, v) \in p\} \\ &= -(-(min\{c_f(u, v) : (u, v) \in p\})) \\ &= -(min\{c_f(v, u) : (u, v) \in p\}) \\ &= -f_p(v, u). \end{aligned}$$

3. Θέλουμε για κάθε $v \in V \setminus \{s,t\} : \sum_{u \in V} f_p(u,v) = 0$.

   Το p είναι ένα επαυξάνον μονοπάτι στο G_f, άρα κάθε κορυφή $v \in p \setminus \{s,t\}$ έχει ακριβώς μία εισερχόμενη ακμή, έστω $(u_1, v) \in p$ και μία εξερχόμενη, έστω $(v, u_2) \in p$. Άρα

   $$\sum_{u \in V} f(u,v) = f_p(u_1,v) + f_p(u_2,v) = f_p(u_1,v) - f_p(v,u_2) = c_f(p) - c_f(p) = 0.$$

   Τέλος, από Ορισμό 13 έχουμε ότι $|f_p| = \sum_{u \in V} f_p(u,t)$. Ομώς, το p είναι μονοπάτι, άρα θα υπάρχει μόνο μία ακμή του, από την κορυφή u στην δεξαμενή t, άρα $\sum_{u \in V} f_p(u,t) = f_p(u,t)$. Η $f_p(u,t) = c_f(p)$ επειδή (u,t) είναι ακμή στο p αναγκαστικά αφού η δεξαμενή t έχει μόνο εισερχόμενες κορυφές και δεν είναι 0 ακριβώς επειδή η (u,t) είναι ακμή στο p.

□

Πόρισμα 1. *Έστω ένα δίκτυο $G(V,E)$ με f ροή στο G και p ένα επαυξάνον μονοπάτι στο G_f. Η f_p όπως ορίστηκε παραπάνω. Ορίζουμε μία συνάρτηση $f' : V \times V \to \mathbb{R}, f' = f + f_p$. Τότε η f' θα είναι η ροή στο G με*

$$|f'| = |f| + |f_p| > |f|$$

Τομές

Ορισμός 17. *Μία **τομή** (S,T) σε ένα δίκτυο $G(V,E)$ με πηγή s και δεξαμενή t, είναι μία διαμέριση του συνόλου κορυφών V σε S και $T = V \setminus S$ τέτοια ώστε $s \in S$ και $t \in T$. Η ροή και η χωρητικότητα κατά μήκος της τομής είναι $f(S,T)$ και $c(S,T)$.* τομή

Το παρακάτω λήμμα δείχνει ότι η συνολική ροή ενός δικτύου ισούται με την ροή κατά μήκος μίας τομής στο ίδιο δίκτυο.

Λήμμα 4. *Έστω ένα δίκτυο $G(V,E)$ με πηγή s και δεξαμενή t και έστω (S,T) μια τομή στο G. Τότε η ροή κατά μήκος της τομής (S,T) είναι $f(S,T) = |f|$.*

Απόδειξη. Χρησιμοποιώντας το Λήμμα 1 έχουμε

$$\begin{aligned} f(S,T) &= f(S,V) - f(S,S) \\ &= f(S,V) \\ &= f(s,V) + f(S-s,V) \\ &= f(s,V) \\ &= |f|. \end{aligned}$$

Η τέταρτη ισότητα ισχύει από τον ορισμό της ροής(Ορισμός 12) (ιδιότητα 3). □

Πόρισμα 2. *Το μέγεθος οποιασδήποτε ροής σε ένα δίκτυο είναι άνω φραγμένο από την χωρητικότητα οποιασδήποτε τομής στο δίκτυο αυτό.*

$$|f| \leq c(S,T)$$

Απόδειξη
Αν (S,T) είναι μία τομή στο δίκτυο G και αν f είναι μία ροή στο ίδιο δίκτυο, τότε από το προηγούμενο Λήμμα 4 έχουμε

$$\begin{aligned} |f| &= f(S,T) \\ &= \sum_{u\in S}\sum_{v\in T} f(u,v) \\ &\leq \sum_{u\in S}\sum_{v\in T} c(u,v) \\ &= c(S,T). \end{aligned}$$

Το Θεώρημα Max-flow min-cut

Θεώρημα 1 (Max-flow min-cut). *Αν f είναι ροή σε ένα δίκτυο $G(V,E)$ με πηγή s και δεξαμενή t, τότε τα παρακάτω είναι ισοδύναμα:*

1. *η f είναι η μέγιστη ροή στο G.*
2. *Το υπολοίπου δίκτυο G_f δεν περιέχει κανένα αυξανόμενο μονοπάτι.*
3. *$|f| = c(S,T)$ για κάποια τομή (S,T) του G.*

Απόδειξη. $(1) \Rightarrow (2)$: Έστω ότι f είναι η μέγιστη ολική ροή στο G αλλά το G_f έχει ένα επαυξάνον μονοπάτι. Το άθροισμα των ροών $f + f_p$ είναι (από Πόρισμα 1) μεγαλύτερο από την $|f|$, ροή στο G. Άτοπο, γιατί υποθέσαμε ότι η f είναι η μέγιστη ροή στο G.
$(2) \Rightarrow (3)$: Έστω ότι το G_f δεν έχει κανένα αυξανόμενο μονοπάτι, δηλαδή δεν υπάρχει μονοπάτι από το s στο t. Ορίζουμε το $S = \{u \in V$:υπάρχει μονοπάτι στο G_f από το s στο $v\}$ και $T = V \setminus S$. Η διαμέριση (S,T) είναι μία τομή τέτοια ώστε $s \in S$ και $t \notin S$ αφού δεν υπάρχει μονοπάτι από το s στο T στο G_f. Για κάθε ζεύγος από κορυφές $u \in S$, $v \in T$, έχουμε $f(u,v) = c(u,v)$, διαφορετικά θα ίσχυε ότι $(u,v) \in E_f$ και $v \in S$. Από το Λήμμα 4, $|f| = f(S,T) = c(S,T)$.
$(3) \Rightarrow (1)$: Από Πόρισμα 2 έχουμε ότι $|f| \leq c(S,T)$ για όλες τις τομές στο G. Άρα αν ισχύει $|f| = c(S,T)$ αυτό σημαίνει ότι η ροή είναι η μέγιστη δυνατή. □

Η μέθοδος Ford-Fulkerson

Μετά από την περιγραφή των βασικών εννοιών που χρησιμοποιούνται στην μέθοδο $Ford - Fulkerson$, είναι η κατάλληλη στιγμή για να περιγράψουμε με περισσότερη λεπτομέρεια τον αλγόριθμο $Ford - Fulkerson$.

Η βασική ιδέα της μεθόδου είναι σε κάθε βήμα να βρίσκουμε κάποιο επαυξάνον μονοπάτι p και να αυξάνουμε την ροή f κατά μήκος αυτού του μονοπατιού τόσο όσο είναι και η αχρησιμοποίητη ροή του $c_f(p)$.

Η υλοποίηση της μεθόδου που περιγράφουμε παρακάτω, υπολογίζει την μέγιστη δυνατή ροή σε ένα δίκτυο $G(V, E)$ ξεκινώντας με μηδενική ροή σε κάθε ακμή του δικτύου $f(u, v) = 0$ για κάθε $u, v \in V$. Αν δύο κορυφές δεν συνδέονται μεταξύ τους, ορίζουμε η ροή κατά μήκος κάθε τέτοιας ακμής και να είναι 0. Η αχρησιμοποίητη ροή $c_f(p)$ υπολογίζεται ως $c_f(u, v) = c(u, v) - f(u, v)$, όπως στον Ορισμό 15. Ο αλγόριθμος σταματάει όταν δεν υπάρχει άλλο επαυξάνον μονοπάτι στο υπολοίπον δίκτυο G_f και τότε η ροή f είναι η μέγιστη.

Ford-Fulkerson-Method(G,s,t)

1. **για κάθε** ακμή $(u, v) \in E$
2. $\quad f(u, v) = 0 = f(v, u)$
3. **όσο** υπάρχει επαυξάνον μονοπάτι από το s στο t στο υπολοίπον δίκτυο G_f
4. $\quad c_f(p) \leftarrow min\{c_f(p)(u, v) : (u, v)$ που ανήκει στο $p\}$
5. $\quad$ **για κάθε ακμή** (u, v) στο p
6. $\quad\quad f(u, v) \leftarrow f(u, v) + c_f(p)$
7. $\quad\quad f(v, u) \leftarrow -f(u, v)$

Ανάλυση της πολυπλοκότητας της μεθόδου Ford-Fulkerson.

Ο χρόνος εκτέλεσης του αλγορίθμου έχει άμεση σχέση με τον τρόπο τον οποίο θα ορίσουμε το επαυξάνον μονοπάτι και τον τρόπο εύρεσής του. Ο αλγόριθμος εκτελείται σε χρόνο της τάξης $O(|E| \cdot |f^*|)$, όπου $|f^*|$ η μέγιστη ροή που βρίσκουμε μετά την εκτέλεση του αλγορίθμου για κάποιο δίκτυο $G(V, E)$. Θα υποθέσουμε ότι οι ροές και οι χωρητικότητες είναι ακέραιοι αριθμοί. Αν όμως είναι ρητοί, μπορούν να γίνουν ακέραιοι με τον κατάλληλο μετασχηματισμό. Με την παραπάνω υπόθεση, η προσέγγιση του χρόνου αναλύεται ως εξής :

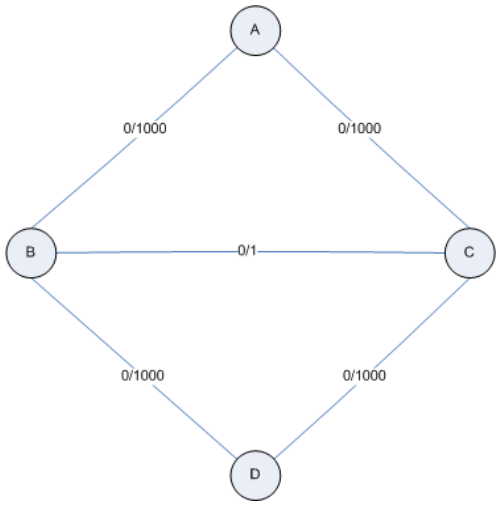
1. Οι δύο πρώτες γραμμές του αλγορίθμου απαιτούν χρόνο $\Theta(|E|)$ γιατί πρέπει να δοθούν αρχικές τιμές σε όλες τις ακμές του γραφήματος.

2. Στις γραμμές 4 - 7 εκτελείται μία ανακύκλωση το πολύ $|f^*|$ φορές, επειδή η ροή f αρχικά είναι μηδενική παντού και σε κάθε επανάληψη αυξάνεται τουλάχιστον κατά μία μονάδα.

3. Ο χρόνος εύρεσης ένος επαυξάνοντος μονοπατιού στο υπολοίπον δίκτυο, χρησιμοποιώντας είτε BFS είτε DFS, είναι $O(|E|)$.

Όταν το $|f^*|$ είναι μικρό, τότε ο χρόνος εκτέλεσης του αλγορίθμου είναι μικρός επίσης. Το δίκτυο του σχήματος 5.9 μας δείχνει ότι τα πράγματα δεν είναι πάντα τόσο καλά.

Ακόμα δεν έχουμε ορίσει τον τρόπο να διαλέγουμε επαυξάνοντα μονοπάτια στο υπολοίπον δίκτυο. Διαισθητικά θα μπορούσε κάποιος να διαλέξει το μεγαλύτερο σε μήκος μονοπάτι, αυτό δηλαδή που αποτελείται από το μεγαλύτερο πλήθος κορυφών. Με αυτόν τον τρόπο, στο παρακάτω δίκτυο ο αλγόριθμος θα δίαλεγε την πρώτη φορά το $ABCD$ και η ροή θα αυξανόταν κατά μία μονάδα. Το δεύτερο μονοπάτι μπορεί να ήταν το $ACBD$ και πάλι η ροή θα μεγάλωνε κατά μία μονάδα. Αυτό θα συνεχιζόταν και τελικά θα χρειάζονταν 2000 επαναλήψεις για να βρεθεί η μέγιστη ολική ροή. Αυτή όμως δεν είναι η βέλτιστη λύση μιας και η μέγιστη ροή μπορεί να βρεθεί μετά από μόνο δύο βήματα, αυξάνοντας την ροή στα μονοπάτια ABD και ACD κατά 1000 μονάδες.



Σχήμα 5.9:

Ο Αλγόριθμος Edmonds-Karp

Οι Edmonds και Karp πρότειναν για την εύρεση των επαυξάνοντων μονοπατιών, να χρησιμοποιείται ο αλγόριθμος Breadth-First-Search (BFS) ο

οποίος διατρέχει ένα γράφημα κατά πλάτος. Με αυτόν τον τρόπο θα επιλέγεται κάθε φορά το συντομότερο μονοπάτι από το s στο t, όπου το μήκος κάθε ακμής του μονοπατιού θα είναι μονάδα. Κατά μήκος αυτού του μονοπατιού αυξάνουμε την ροή όσο μας επιτρέπει η χωρητικότητα του. Θα αποδείξουμε παρακάτω ότι αυτός ο αλγόριθμος τρέχει σε χρόνο $O(|V| \cdot |E^2|)$.

Λήμμα 5 (Υπομονοπάτια των συντομότερων μονοπατιών, είναι επίσης συντομότερα μονοπάτια). *Δεδομένου ενός κατευθυνόμενου γραφήματος με βάρη $G = (V, E)$ με συνάρτηση βάρους $w : E \to \mathbf{R}$, έστω $p =< v_1, v_2, ..., v_k >$ ένα συντομότερο μονοπάτι από την κορυφή v_1 στην v_k και για κάθε i και j τέτοια ώστε $1 \leq i \leq j \leq k$, έστω $p_{ij} =< v_i, v_{i+1}, ..., v_j >$ ένα υπομονοπάτι του p από την κορυφή v_i στην v_j. Τότε το p_{ij} είναι ένα συντομότερο μονοπάτι από από την v_i στην v_j.*

Απόδειξη. Αν αποσυνθέσουμε το p σε υπομονοπάτια, θα έχουμε ότι $w(p) = w(p_{1i}) + w(p_{ij}) + w(p_{jk})$. Έστω ότι υπάρχει μονοπάτι p'_{ij} από την κορυφή v_i στην v_j με βάρος $w(p_{1i}) + w(p'_{ij}) + w(p_{jk})$ μικρότερο από $w(p)$. Άτοπο γιατί υποθέσαμε ότι το p είναι το συντομότερο μονοπάτι από την κορυφή v_1 στην v_k. □

Πόρισμα 3. *Έστω $G = (V, E)$ γράφημα με βάρη, με συνάρτηση βάρους $w : E \to \mathbf{R}$. Αν υποθέσουμε ότι το συντομότερο μονοπάτι του γραφήματος από την πηγή s προς μία κορυφή v, μπορεί να γραφτεί σαν $s \rightsquigarrow^{p'} u \to v$ για κάποια κορυφή u και μονοπάτι p', τότε το μήκος του συντομότερου μονοπατιού από την πηγή προς την v είναι $\delta(s, v) = \delta(s, u) + w(u, v)$.*

Απόδειξη.

$$\begin{aligned} \delta(s, v) &= w(p) \\ &= w(p') + w(u, v) \\ &= \delta(s, u) + w(u, v). \end{aligned}$$

□

Πόρισμα 4. *Δεδομένου ενός κατευθυνόμενου γραφήματος με βάρη $G = (V, E)$ με συνάρτηση βάρους $w : E \to \mathbf{R}$ και πηγή s, για κάθε ακμή $(u, v) \in \mathbf{E}$, ισχύει ότι $\delta(s, v) \leq \delta(s, u) + w(u, v)$.*

Απόδειξη. Το συντομότερο μονοπάτι p από την s στην v δεν έχει περισσότερο βάρος από κάθε άλλο μονοπάτι από το s στο v. Ειδικότερα, το μονοπάτι p δεν έχει περισσότερο βάρος από το μονοπάτι που διαλέγει ένα συντομότερο μονοπάτι από το s στο u και μετά ακολουθεί την ακμή (u, v). □

Λήμμα 6. *Αν τρέξουμε τον αλγόριθμο Edmonds-Karp για ένα δίκτυο $G = (V, E)$ με πηγή s και δεξαμενή t, τότε για όλες τις κορυφές $v \in V \setminus \{s, t\}$, το μήκος του συντομότερου μονοπατιού $\delta_f(s, u)$ στο υπολοίπου γράφημα G_f αυξάνεται με κάθε αύξηση της ροής. ($\delta_f(s, u)$ είναι το μήκος του συντομότερου μονοπατιού από το u στο v στο G_f, όπου κάθε ακμή έχει μοναδιαίο μήκος)*

Απόδειξη. Έστω ότι για κάποια κορυφή $v \in V \setminus \{s, t\}$, υπάρχει κάποια αύξηση ροής που έχει σαν αποτέλεσμα το $\delta_f(s, u)$ να ελαττωθεί και έστω f η ροή πριν την αύξηση και f' η επαυξημένη ροή. Τότε,

$$\delta'_f(s, u) < \delta_f(s, u).$$

Υποθέτουμε χωρίς βλάβη της γενικής περίπτωσης ότι $\delta'_f(s, v) < \delta'_f(s, u)$ για κάθε κορυφή $v \in V \setminus \{s, t\}$ τέτοια ώστε $\delta'_f(s, u) < \delta_f(s, u)$. Ομοίως, υποθέτουμε ότι για όλες τις κορυφές $v \in V \setminus \{s, t\}$, αν $\delta'_f(s, u) < \delta'_f(s, v) \Rightarrow \delta_f(s, u) \leq \delta'_f(s, u)$.

Παίρνουμε το συντομότερο μονοπάτι p' στο G'_f της μορφής $s \curvearrowright u \to v$ με την u να προηγείται της v σε αυτό το μονοπάτι.

Από το Πόρισμα 3 θα πρέπει να έχουμε $\delta'_f(s, u) = \delta'_f(s, v) - 1$ αφού η ακμή (u, v) ανήκει στο p', που είναι το συντομότερο μονοπάτι από το s στο u.

Με δεδομένες τις κορυφές u και v, μπορούμε να θεωρήσουμε τη ροή από το u στο v πριν την αύξηση της ροής στο G_f. Αν $f(u, v) < c(u, v)$, τότε έχουμε,

$$\begin{aligned} \delta(s, v) &\leq \delta_f(s, u) + 1 \\ &\leq \delta'_f(s, u) + 1 \\ &= \delta(s, u). \end{aligned}$$

Αυτό όμως είναι άτοπο γιατί στην αρχή υποθέσαμε ότι η αύξηση ροής προκαλεί μείωση της απόστασης από το s στο v.
Άρα $f(u, v) = c(u, v)$, που σημαίνει ότι $(u, v) \notin E_f$.
Το επαυξάνον μονοπάτι p που επιλέχθηκε από το G_f για να προκύψει το G'_f πρέπει να περιέχει την ακμή (v, u) αφού η $(u, v) \notin E'_f$ και η $(u, v) \notin E_f$ όπως μόλις δείξαμε. Αυτό σημαίνει ότι αυξάνοντας τη ροή κατά μήκος του p, μειώνει τη ροή κατά μήκος της ακμής (u, v) και η v εμφανίζεται πριν από την u στο p. Αφού το p είναι το συντομότερο μονοπάτι από το s στο t, τα ̈υπομονοπάτια" του θα είναι ακόμα συντομότερα μονοπάτια και έτσι έχουμε $\delta_f(s, u)0 = \delta_f(s, v) + 1$. Συνεπώς με τη βοήθεια του Λήμματος 6 έχουμε,

$$\begin{aligned} \delta_f(s, v) &= \delta_f(s, u) - 1 \\ &\leq \delta'_f(s, u) - 1 \\ &= \delta'_f(s, v) - 2 \\ &< \delta'_f(s, v) \end{aligned}$$

που αντιτίθεται με την αρχική υπόθεση. □

Το επόμενο Θεώρημα φράσει το πλήθος των επαναλήψεων του αλγόριθμου Edmonds-Karp.

Θεώρημα 2. *Αν ο αλγόριθμος Edmonds-Karp τρέξει για ένα δίκτυο $G = (V, E)$ με πηγή s και δεξαμενή t, τότε ο συνολικός αριθμός επαυξήσεων της ολικής ροής που προκαλεί ο αλγόριθμος είναι $O(|V| \cdot |E|)$.*

Απόδειξη. Λέμε ότι μία ακμή (u, v) σε ένα επαυξάνον μονοπάτι p είναι **καθοριστική** αν $c_f(p) = c_f(u, v)$. *καθοριστική ακμή*

Μετά από κάθε επαύξηση της ροής κατά μήκος ενός επαυξάνοντως μονοπατιού, κάθε καθοριστική ακμή εξαλείφεται από το G_f. Επιπλέον, τουλάχιστον μία ακμή σε κάθε επαυξάνον γράφημα πρέπει να είναι καθοριστική.
Έστω u και v συνδεδεμένες κορυφές. Πόσες φορές μπορεί μία ακμή να είναι καθοριστική κατά την διάρκεια του αλγορίθμου Edmonds-Karp; Αφού τα επαυξάνοντα μονοπάτια είναι και τα συντομότερα, την πρώτη φορά που η (u, v) είναι καθοριστική, έχουμε

$$\delta_f(s, v) = \delta_f(s, u) + 1$$

Μόλις αυξηθεί η ροή, η ακμή (u, v) εξαλείφεται από το G_f. Δεν μπορεί να ξαναεμφανιστεί σε κάποιο άλλο επαυξάνον μονοπάτι αν δεν μειωθεί πρώτα η ροή από το u στο v και αυτό συμβαίνει μόνο αν η (v, u) εμφανιστεί σε ένα επαυξάνον μονοπάτι. Αν f' είναι η ροή όταν αυτό το γεγονός συμβεί, τότε θα έχουμε

$$\delta'_f(s, v) = \delta'_f(s, u) + 1$$

Αφού $\delta_f(s, v) \leq \delta'_f(s, v)$ από το Λήμμα 4 έχουμε ότι

$$\begin{aligned} \delta'_f(s, u) &= \delta'_f(s, v) + 1 \\ &\geq \delta_f(s, v) + 1 \\ &= \delta_f(s, u) + 2 \end{aligned}$$

Άρα από τη στιγμή που η ακμή (u, v) γίνει καθοριστική μέχρι την επόμενη στιγμή που θα ξαναγίνει η απόσταση της u από την πηγή αυξάνεται τουλάχιστον κατά δύο μονάδες. Αρχικά, η απόσταση της u από την πηγή είναι τουλάχιστον 1 και μέχρι (αν ποτέ) να μην είναι προσιτή από την πηγή, η απόσταση αυτή είναι το πολύ $|V| - 2$. Άρα η (u, v) μπορεί να γίνει καθοριστική το πολύ $O(|V|)$ φορές. Αφού υπάρχουν $O(|E|)$ ζεύγη κορυφών που μπορεί να συνδέονται με μία ακμή του υπολοίποντος γραφήματος, ο συνολικός αριθμός των καθοριστικών ακμών κατά την διάρκεια εκτέλεσης του αλγορίθμου Edmonds-Karp είναι $O(|V| \cdot |E|)$. Κάθε επαυξάνον μονοπάτι έχει τουλάχιστον μία καθοριστική ακμή, έτσι το θεώρημα ισχύει. □

Αφού κάθε επανάληψη του Ford-Fulkerson απαιτεί χρόνο $O(|E|)$ όταν τα επαυξάνοντα μονόπάτια βρίσκονται με BFS, ο συνολικός χρόνος που απαιτεί ο Edmonds-Karp είναι $O(|V| \cdot |E|^2)$.

Πρέπει επίσης να συμπληρώσουμε ότι ο χρόνος εκτέλεσης του αλγορίθμου εξαρτάται από την αναπαράσταση του δικτύου στον υπολογιστή, κάτι το οποίο θα εξηγήσουμε αναλυτικά σε επόμενη ενότητα.

Πιο αναλυτικά...

ΒΗΜΑ Α : Δημιουργούμε ένα γράφημα $G_f = (V, E_f)$ ως εξής :

1. αν η ροή της ακμής (i, j) είναι ίση με τη χωρητικότητα της, τότε $(j, i) \in E_f$
2. αν η ροή της ακμής (i, j) είναι μηδενική, τότε $(i, j) \in E_f$
3. αν η ροή της ακμής (i, j) έχει θετική τιμή, τότε $(i, j), (j, i) \in E_f$

ΒΗΜΑ Β : Εφαρμόζουμε τον αλγόριθμο Breadth-First-Search στο επαυξάνον δίκτυο $G_f = (V, E_f)$ με αρχική κορυφή την πηγή s για να βρούμε ένα κατευθυνόμενο μονοπάτι με το ελάχιστο δυνατό πλήθος ακμών, από την πηγή προς την δεξαμενή. Αν δεν υπάρχει τέτοιο, προχωράμε στο ΒΗΜΑ Δ. Αν όχι πάμε στο ΒΗΜΑ Γ.

ΒΗΜΑ Γ : Το μονοπάτι που βρήκαμε στο προηγούμενο βήμα είναι ένα επαυξάνον μονοπάτι, άρα αυξάνουμε την ροή κατα μήκος αυτού το μονοπατιου και επιστρέφουμε στο ΒΗΜΑ Α.

ΒΗΜΑ Δ : Έχουμε έχουμε βρει την μέγιστη ροή και ο αλγόριθμος τελειώνει.

Κεφάλαιο 6

Εφαρμογές του αλγόριθμου μεγιστοποίησης της δικτυακής ροής. Ταίριασματα σε διμερή γραφήματα.

Πολλά προβλήματα που εκ πρώτης όψεως δεν φαίνεται να έχουν σχέση με δίκτυα, βρίσκουν εύκολα λύση αν μετατραπούν σε τέτοια και εφαρμοστούν πάνω τους γνωστοί αλγόριθμοι από την θεωρία γραφημάτων. Ένα από αυτά είναι το εξής: Δεδομένου ενός πλήθους ανδρών και γυναικών, πώς μπορούμε να βρούμε τα περισσότερα δυνατά ζευγάρια με δεδομένες τις σχέσεις γνωριμίας ανάμεσα τους.

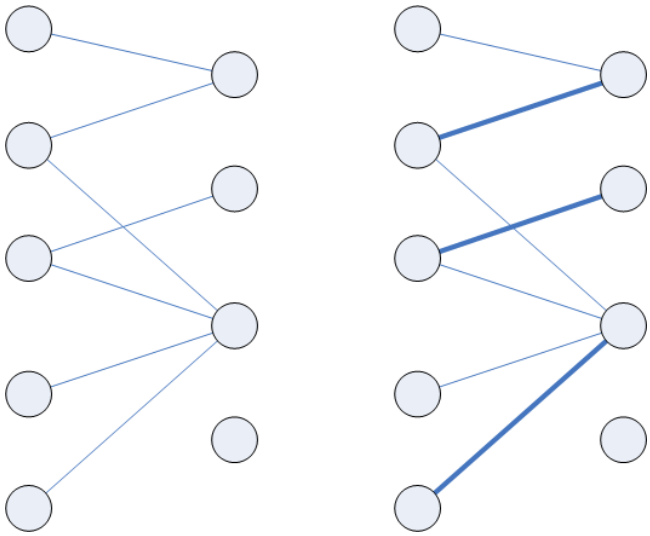
6.1 Ορισμοί

Ορισμός 18. *Έστω μη κατευθυνόμενο γράφημα $G = (V, E)$. Το G λέγεται **διμερές γράφημα** αν το σύνολο των κορυφών V μπορεί να χωριστεί σε δύο υποσύνολα V_1 και V_2 τέτοια ώστε να υπάρχουν μόνο ακμές της μορφής $(v_1, v_2), v_1 \in V_1, v_2 \in V_2$.* *διμερές γράφημα*

Ορισμός 19. ***Ταίριασμα** είναι ένα υποσύνολο ακμών $M \subseteq E$ τέτοιο ώστε για κάθε κορυφή $v \in V$ το πολύ μία ακμή του M καταλήγει στο v. $|M|$ είναι το πλήθος των ακμών που ανήκουν στο M. Δείτε Σχήμα 6.1.* *ταίριασμα*

Ορισμός 20. *Λέμε ότι μία κορυφή $v \in V$ έχει **ταίρι** μέσω του ταιριάσματος M αν υπάρχει ακμή του M που προσπίπτει στην v, αλλίως η v δεν έχει ταίρι.* *ταίρι*

Ορισμός 21. ***Μέγιστο ταίριασμα** ονομάζεται ένα ταίριασμα με το μέγιστο δυνατό πλήθος κορυφών.* *μέγιστο ταίριασμα*



Σχήμα 6.1: Ταίριασμα ενός διμερούς γραφήματος.

6.2 Λύση του προβλήματος εύρεσης μέγιστου ταιριάσματος σε διμερή γραφήματα

Μία διαδεδομένη λύση στο πρόβλημα εύρεσης μέγιστου ταιριάσματος σε διμερή γραφήματα είναι η χρήση του αλγορίθμου εύρεσης μέγιστης ροής σε ένα δίκτυο. Για το λόγο αυτό κατασκευάζουμε ένα δίκτυο που αντιστοιχεί στο γράφημα που θέλουμε να εξετάσουμε.

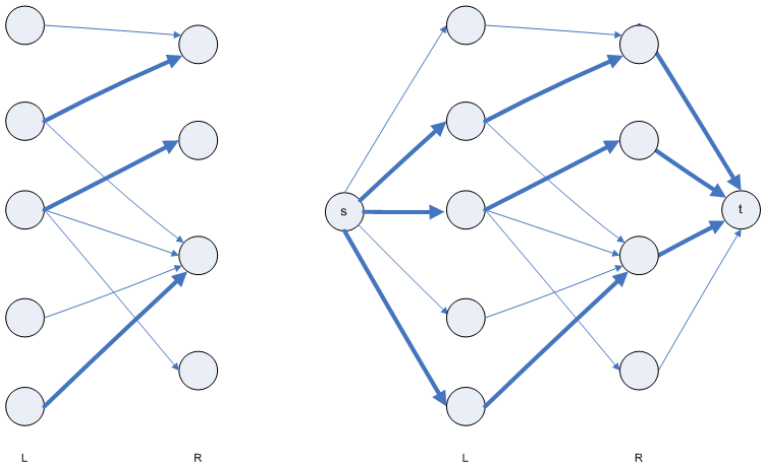
Ορισμός 22. ***Αντίστοιχο δίκτυο*** *$G' = (V', E')$. Ορίζουμε δύο καινούργιες κορυφές s, t να είναι αντίστοιχα η πηγή και η δεξαμενή του αντίστοιχου δικτύου και $V' = V \cup \{s, t\}$. Αν τα δύο σύνολα του V είναι τα L, R τότε ορίζουμε* αντίστοιχο δίκτυο

$$E' = \{(s, u) : u \in L\} \cup \{(u, v) : u \in L, v \in R, (u, v) \in E\} \cup \{(v, t) : v \in R\}.$$

Η χωρητικότητα κάθε ακμής στο E θέτουμε να είναι ίση με τη μονάδα. Δείτε Σχήμα 6.2.

Θεώρημα 3. *Έστω $G = (V, E)$ ένα διμερές γράφημα με διαμέριση κορυφών $V = L \cup R$, και $G' = (V', E')$ το αντίστοιχο δίκτυο. Αν M είναι ένα ταίριασμα του G υπάρχει ροή f στο G' με ακέραια τιμή τέτοια ώστε $|f| = |M|$. Αντίστροφα, αν f είναι ροή στο G' με ακέραια τιμή τότε υπάρχει ταίριασμα με $|M| = |f|$.*

Απόδειξη. Πρώτα θα δείξουμε ότι ένα ταίριασμα M στο G αντιστοιχεί σε ροή f με ακέραια τιμή. Ορίζουμε την f ως εξής. Αν $(u, v) \in M$, τότε $f(s, u) = f(u, v) = f(v, t) = 1$ και $f(u, s) = f(v, u) = f(t, v) = -1$. Για τις υπόλοιπες ακμές $(u, v) \in E', f(u, v) = 0$. Διαισθητικά, κάθε ακμή $(u, v) \in M$ αντιστοιχεί σε μία μονάδα ροής στο G' η οποία διατρέχει το μονοπάτι $s \to u \to v \to t$.



Σχήμα 6.2: Αντίστοιχο δίκτυο ενός ταιριάσματος σε διμερές γράφημα. Όλες οι ακμές έχουν χωρητικότητα 1 και ροή 0, εκτός από αυτές με έντονη γραμμή, από τις οποίες περνάει ροή με τιμή 1.

.

Επιπλέον τα μονοπάτια δεν έχουν άλλες κοινές κορυφές εκτός από την πηγή s και την δεξαμενή t. Η f είναι ροή γιατί προκύπτει από την επαύξηση της ροής σε κάθε μονοπάτι. Επίσης, η ροή κατά μήκος της τομής $(L \cup \{s\}, R \cup \{t\}$ είναι ίση με $|M|$, άρα από Λήμμα 4 $|f| = |M|$.
Αντίστροφα, αν f είναι ροή με ακέραια τιμή στο G' και

$$M = \{(u, v) : u \in L, v \in R, f(u, v) > 0\}.$$

Κάθε κορυφή $u \in L$ έχει μόνο μία εισερχόμενη ακμή, την (s, u) με χωρητικότητα 1. Άρα, από κάθε $u \in L$ περνάει εισερχόμενη το πολύ μία μονάδα ροής εάν και μόνο εάν υπάρχει $v \in R$ τέτοια ώστε $f(u, v) = 1$. Άρα το πολύ μία ακμή εξέρχεται από κάθε $u \in L$ με ροή 1. Άρα το M είναι ένα ταίριασμα. Για να δείξουμε ότι $|M| = |f|$, παρατηρούμε για κάθε ακμή $u \in L$ με ζεύγος υπάρχει η (s, u) με $f(s, u) = 1$, και για κάθε ακμή $(u, v) \in E - M$ έχουμε $f(u, v) = 0$. Συνεπώς από το Λήμμα 1, από ιδιότητες της ροής και γνωρίζοντας ότι δεν υπάρχουν ακμές από το L στο t έχουμε,

$$\begin{aligned} |M| &= f(L, R) \\ &= f(L, V') - f(L, L) - f(L, s) - f(L, t) \\ &= 0 - 0 + f(s, L) - 0 \\ &= f(s, V') \\ &= |f| \end{aligned}$$

□

Θεώρημα 4. *Αν η συνάρτηση χωρητικότητας c παίρνει μόνο ακέραιες τιμές, τότε η μέγιστη ροή που παράγεται από την μέθοδο Ford-Fulkerson είναι ακέραιος αριθμός. Επιπλέον, για κάθε κορυφή $u, v \in V$, η τιμή της $f(u, v)$ είναι ακέραια.*

Απόδειξη. Με επαγωγή ως προς το πλήθος των επαναλήψεων. □

Πόρισμα 5. *Το πλήθος των ακμών ενός μέγιστου ταιριάσματος σε ένα διμερές γράφημα G είναι ίσο με την τιμή της μέγιστης ροής του αντίστοιχου δικτύου G'.*

Απόδειξη. Έστω Μ μέγιστο ταίριασμα σε ένα διμερές γράφημα G και η ροή του αντίστοιχου δικτύου G' η οποία δεν είναι μέγιστη. Τότε υπάρχει μέγιστη ροή f' με $|f'| > |f|$. Η f' αντιστοιχεί σε ένα ταίριασμα M' του G με $|M'| = |f'| \geq |f| = |M|$. Άρα το M δεν είναι το μέγιστο ταίριασμα. □

6.3 Πολυπλοκότητα

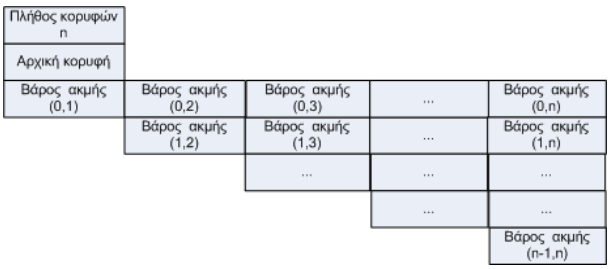
Αποδείξαμε ότι δεδομένου ενός διμερούς γραφήματος G, μπορούμε να βρούμε το μέγιστο ταίριασμα του δημιουργώντας το αντίστοιχο δίκτυο G' και εφαρμόζοντας τον αλγόριθμο εύρεσης μέγιστης ροής Edmongs-Karp. Έτσι πετυχαίνουμε το μέγιστο ταίριασμα M. Αφού κάθε ταίριασμα σε διμερές γράφημα έχει πλήθος στοιχείων το μικρότερο από τα L, R άρα είναι $O(|V|)$, η τιμή της μέγιστης ροής στο αντίστοιχο δίκτυο G' είναι $O(|V|)$. Άρα μπορούμε να βρούμε το μέγιστο ταίριασμα σε ένα διμερές γράφημα σε χρόνο $O(|V| \cdot |E|)$.

Κεφάλαιο 7

Σχόλια

Για της ανάγκες αυτής της εργασίας υλοποιήθηκαν οι παρακάτω αλγόριθμοι :

1. Εύρεση συντομότερου μονοπατιού Dijkstra.
2. Depth First Search.
3. Εύρεση συντομότερου μονοπατιού διατρέχοντας το γράφημα με τον BFS.
4. Εύρεση μέγιστης ροής (Αλγόριθμος Edmonds - Karp).
5. Εύρεση μέγιστου ταιριάσματος σε διμερές γράφημα.



Σχήμα 7.1: Είσοδος στον αλγόριθμο του Dijkstra. Το γράφημα είναι μη-κατευθυνόμενο. Όσες ακμές δεν υπάρχουν έχουν άπειρο βάρος.

Ο αλγόριθμος Breadth First Search Shortest Path παίρνει σαν είσοδο ένα αρχείο ίδιας μορφής με αυτό του σχήματος 7.2.

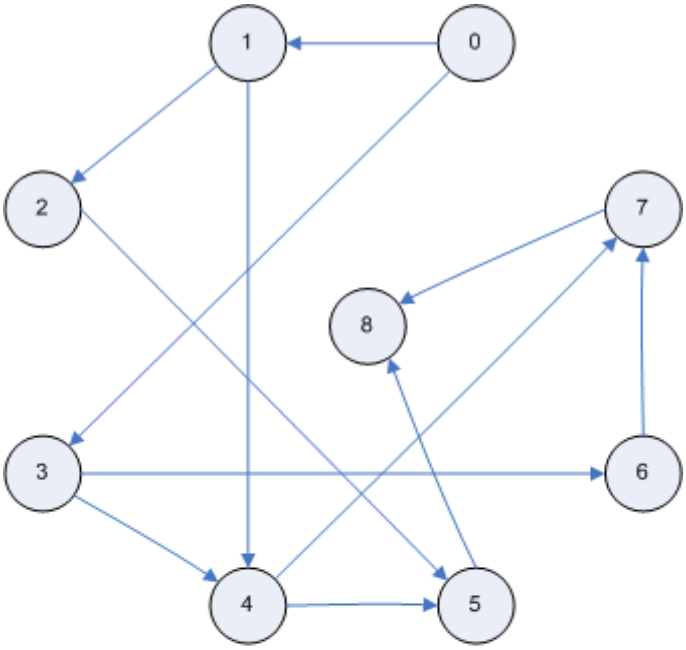
Τέλος, υλοποιήθηκαν και κάποιες ρουτίνες οι οποίες έχουν σαν είσοδο ένα γράφημα ή ένα δίκτυο και παρουσιάζουν τα χαρακτηριστικά του. Για παράδειγμα, το πλήθος των ακμών ενός γραφήματος, αν μια κορυφή συνδέεται με μια άλλη, τις εισερχόμενες και εξερχόμενες ακμές μια κορυφής. Επίσης, το συνολικό ποσό ροής που εισέρχεται και εξέρχεται από μια κορυφή ενός δικτύου, η αχρησιμοποίητη χωρητικότητα κάθε ακμής κ.α.

Πλήθος κορυφών n						
Αρχική κορυφή						
Βαθμός κορυφης 0, d^0	1ος Γείτονας κορυφής 0, n_1	...	d^0οστος Γείτονας κορυφής 0	Βάρος ακμής (0, 0)	...	Βάρος ακμής (0, n_1)
Βαθμός κορυφης 1, d^1	1ος Γείτονας κορυφής 1, n_2	...	d^1οστος Γείτονας κορυφής 1	Βάρος ακμής (1, 0)	...	Βάρος ακμής (1, n_2)
...	...	...	...	...	...	...
Βαθμός κορυφης n-1, d^{n-1}	1ος Γείτονας κορυφής n-1, n_{n-1}	...	d^{n-1}οστος Γείτονας κορυφής n-1	Βάρος ακμής (n-1, 0)	...	Βάρος ακμής (n-1, n_{n-1})

Σχήμα 7.2: Είσοδος στον αλγόριθμο DFS. Το γράφημα είναι κατευθυνόμενο. Όσες ακμές δεν υπάρχουν έχουν μηδενικό βάρος.

Πλήθος κορυφών αριστερής πλευράς N												
Βαθμός κορυφης 0, d^0	1ος Γείτονας κορυφής 0	...	d^0οστος Γείτονας κορυφής 0	Χωρητικότητα ακμής (0, 0)	...	Χωρητικότητα ακμής (0, n_1)	Ροή ακμής (0, 0)	...	Ροή ακμής (0, n_{d0})	forward(0, 0)	...	forward (0, n_{d0})
Βαθμός κορυφης 1, d^1	1ος Γείτονας κορυφής 1	...	d^1οστος Γείτονας κορυφής 1	Χωρητικότητα ακμής (1, 0)	...	Χωρητικότητα ακμής (1, n_2)	Ροή ακμής (1, 0)	...	Ροή ακμής (1, n_{d1})	forward(1, 0)	...	forward(1, n_{d1})
...	...	...	...	...	...	...	...	...	...	...	...	...
Βαθμός κορυφης n-1, d^{N-1}	1ος Γείτονας κορυφής N-1	...	d^{N-1}οστος Γείτονας κορυφής N-1	Χωρητικότητα ακμής (N-1, 0)	...	Χωρητικότητα ακμής (N-1, n_{N-1})	Ροή ακμής (N-1, 0)	...	Ροή ακμής (N-1, n_{dN-1})	forward(N-1, 0)	...	forward(N-1, n_{dN-1})

Σχήμα 7.3: Είσοδος στον αλγόριθμο εύρεσης μέγιστης ροής. Κάθε ακμή (i, j) συμπληρώνεται από μια άλλη $(j, i) \in E$ με μηδενική χωρητικότητά και ροή και με $forward(j, i) = -1$, ενώ η ακμή (i, j) έχει $forward(j, i) = 1$.

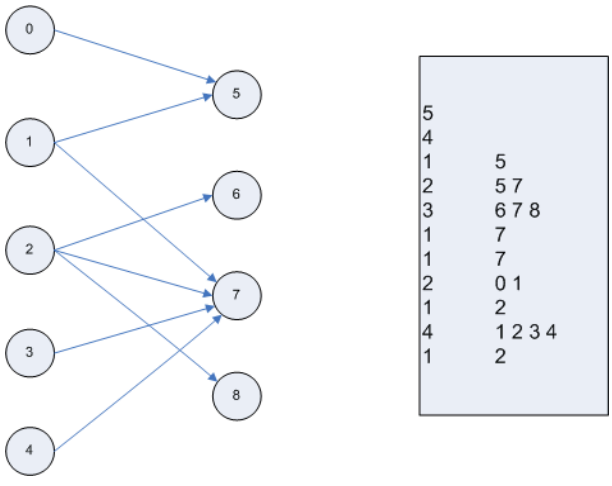



```
9
2       1 3     1 1     0 0     1 1
3       0 2 4   0 1 1   0 0 0   -1 1 1
2       1 5     0 1     0 0     -1 1
3       0 4 6   0 1 1   0 0 0   -1 1 1
4       1 3 5 7 0 0 1 1 0 0 0 0 -1 -1 1 1
3       2 4 8   0 0 1   0 0 0   -1 -1 1
2       3 7     0 1     0 0     -1 1
3       4 6 8   0 0 1   0 0 0   -1 -1 1
2       5 7     0 0     0 0     -1 -1
```

Σχήμα 7.4: Αρχείο εισόδου του αλγόριθμου εύρεσης μέγιστης ροής.

Πλήθος κορυφών αριστερής πλευράς N1			
Πλήθος κορυφών δεξιάς πλευράς N2			
Βαθμός κορυφης 0, d^0	1ος Γείτονας κορυφής 0	...	d^0οστος Γείτονας κορυφής 0
Βαθμός κορυφης 1, d^1	1ος Γείτονας κορυφής 1	...	d^1οστος Γείτονας κορυφής 1
...	...	...	...
Βαθμός κορυφης n-1, d^{N1-1}	1ος Γείτονας κορυφής N1-1	...	d^{N1-1}οστος Γείτονας κορυφής N1-1
Βαθμός κορυφης 0, d^{N1}	1ος Γείτονας κορυφής N1	...	d^{N1}οστος Γείτονας κορυφής N1
Βαθμός κορυφης 1, d^{N1+1}	1ος Γείτονας κορυφής N1+1	...	d^{N1+1}οστος Γείτονας κορυφής N1+1
...	...	...	...
Βαθμός κορυφης N1+N2-1, $d^{N1+N2-1}$	1ος Γείτονας κορυφής N1+N2-1	...	$d^{N1+N2-1}$οστος Γείτονας κορυφής N1+N2-1

Σχήμα 7.5: Είσοδος στον αλγόριθμο εύρεσης μέγιστου ταιριάσματος σε διμερές γράφημα. Το γράφημα είναι ένα διμερές κατευθυνόμενο γράφημα.



Σχήμα 7.6: Διμερές γράφημα και αρχείο εισόδου για τον αλγόριθμο εύρεσης μέγιστου ταιριάσματος σε διμερές γράφημα .

Βιβλιογραφία

[1] Robert Sedgewick, *Algorithms in C, Princeton University 1990.*

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, *Introduction to algorithms, The MIT Press.*

[3] Alan Gibbons, *Algorithmic Graph Theory, Cambridge University Press.*