Μελετη και Υλοποιηση του Αλγοριθμου Floyd -Warshall

Γιαννακος Αντωνιος-Μαριος
2008021

Τμήμα Μηχανικών Πληροφορικής Τ.Ε
Σχολή Τεχνολογικών Εφαρμογών (έδρα: Σπάρτη)
Τ.Ε.Ι Πελοποννήσου

25.5.2016

Επίβλεψη Πτυχιακής και Προϊστάμενος Τμήματος
Καραγιωργος Γρηγορης

Σύμφωνα με απόφαση της Συνέλευσης του ΤΕΙ Πελοποννήσου οι φοιτητές που εκπονούν την πτυχιακή τους εργασία υποχρεούνται να συμπεριλαμβάνουν στις προκαταρκτικές σελίδες της εργασίας τους το παρακάτω κείμενο, υπογεγραμμένο από τους ίδιους.

«

ΔΗΛΩΣΗ ΜΗ ΛΟΓΟΚΛΟΠΗΣ ΚΑΙ ΑΝΑΛΗΨΗΣ ΠΡΟΣΩΠΙΚΗΣ ΕΥΘΥΝΗΣ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάση επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δε μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

Όνομα και Επώνυμο Συγγραφέα (Με Κεφαλαία):

……

Υπογραφή (Ολογράφως, χωρίς μονογραφή):

……

Ημερομηνία (Ημέρα – Μήνας – Έτος):

……

»

Περιεχόμενα

Κατάλογος Σχημάτων

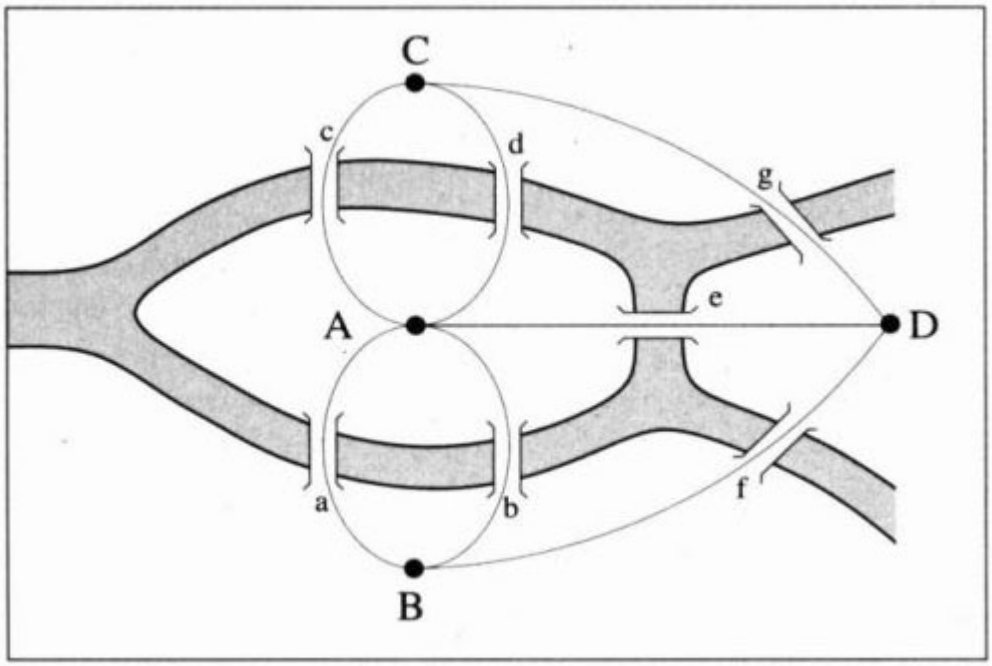
Περίληψη

Το πρόβλημα της εύρεσης της ελάχιστης διαδρομής αποτελεί ένα από τα θεμελιώδη προβλήματα της υπολογιστικής νοημοσύνης και ένα από τα βασικά προβλήματα προς επίλυση με χρήση των ηλεκτρονικών υπολογιστών σήμερα. Ο λόγος έχει να κάνει με την άμεση συσχέτισή του με προβλήματα δικτύων, τα οποία στην σημερινή εποχή πολλαπλασιάζονται ως προς την χρήση τους απο τον άνθρωπο (δίκτυα τηλεπικοινωνιών, μεταφοράς, διανομής ενέργειας, υπολογιστών, πληροφοριών κ.α.) σχεδόν με εκθετικό ρυθμό όσον αφορά τον όγκο αλλά και την πολυπλοκότητα τους.Επίσης, τα σύγχρονα συστήματα υποβοήθησης μετακινήσεων, με χρήση δορυφορικών συστημάτων πλοήγησης απαιτούν και αυτά την επίλυση παρόμοιων προβλημάτων για την εύρωστη λειτουργία τους. Σκοπός της περιγραφικής έρευνας είναι η μελέτη του **Floyd Warshall**. Εμπνευση του θέματος αποτέλεσε οτι αν και ο αλγόριθμος παρέχει σοβαρή υποστήριξη ανάμεσα στους γράφους, δεν υπάρχουν πολλές ελληνικές βιβλιογραφικές αναφορές που να τον περιγράφουν. Εξετάζεται η λειτουργία του έναντι άλλων σχετικών αλγορίθμων εύρεσης της ελάχιστης διαδρομής, αναλύονται τα χαρακτηριστικά του και υλοποιείται ο αλγόριθμος με σχετικό παράδειγμα. Ο αλγόριθμος τρέχει με χρόνο $O(V^3)$ με τη δυνατότητα χειρισμού αρνητικού βάρους. Η υλοποίησή του έγινε σε JAVA σε περιβάλλον Eclipse. Εξετάστηκε η συμπεριφορά του και σε περίπτωση αρνητικών κύκλων και εφαρμόστηκε παράδειγμα αναζήτησης ελάχιστης διαδρομής σε επιλεγμένους ελληνικούς προορισμούς λαμβάνοντας υπόψην τα πραγματικά δεδομένα χιλιομετρικών αποστάσεων των πόλεων κόμβων. Ως αποτέλεσμα της έρευνας διαπιστώνουμε ότι ο αλγόριθμος παράγει τα σωστά αποτελέσματα στον αναμενόμενο χρόνο, αποδεικνύοντας πως είναι ένας δυνατός και εύχρηστος αλγόριθμος για αντιμετώπιση προβλημάτων εύρεσης συντομότερης διαδρομής.

Κεφάλαιο 1

Εισαγωγή

Όταν το 1735, ο **Euler** αναρωτήθηκε αν ήταν δυνατό να βρεθεί ένα μονοπάτι μέσα απο τη πόλη του Königsberg [1], διασχίζοντας κάθε μία από τις επτά γέφυρές του ακριβώς μια φορά και τελικά επιστρέφοντας στο αρχικό σημείο εκκίνησης, έβαλε τα θεμέλια για αυτό που είναι σήμερα γνωστό θεωρητικά ως γράφος, το οποίο με τη σειρά του άνοιξε το δρόμο για την ανάπτυξη αλγορίθμων εύρεσης συντομότερης διαδρομής.



Σχήμα 1.1: Το Πρόβλημα των Επτά Γεφυρών του Königsberg

Το πρόβλημα εύρεσης της **συντομότερης διαδρομης** είναι ένα από τα πιο θεμελιώδη και πιο συχνά ανακύπτοντα προβλήματα γράφων, τόσο ως αυτούσια προβλήματα, όσο και ως υποπροβλήματα σε αρκετά πιο σύνθετα. Εκτός των προφανών εφαρμογών, όπως η προετοιμασία των διαγραμμάτων του χρόνου ταξιδιού και απόστασης, οι υπολογισμοί του συντομότερου δρόμου απαιτούνται συχνά στο χώρο των τηλεπικοινωνιών και των βιομηχανιών μεταφοράς, όπου πρέπει να αποστέλλονται μηνύματα ή οχημάτα μεταξύ

γεωγραφικών θέσεων όσο πιο γρήγορα ή όσο το δυνατόν φθηνότερα.

Άλλα παραδείγματα [3](σελ.661),[2](σελ.3), είναι οι προσομοιώσεις της πολύπλοκης ροής της κυκλοφορίας και τα εργαλεία σχεδιασμού αυτής, τα οποία στηρίζονται σε μεγάλο αριθμό επιμέρους προβλημάτων εύρεσης της ελάχιστης διαδρομής.

Περαιτέρω εφαρμογές περιλαμβάνουν πολλά πρακτικά προβλήματα ακέραιου προγραμματισμού. Οι υπολογισμοί της ελάχιστης διαδρομής χρησιμοποιούνται ως υπορουτίνες σε διαδικασίες επίλυσης για την υπολογιστική βιολογία (π.χ. ευθυγράμμιση της αλληλουχίας του Deoxyribonucleic acid - DNA), για τον σχεδιασμό Very Large Scale Integration - VLSI, το πρόβλημα συσκευασίας σακιδίου, καθώς και τα προβλήματα του μετακινούμενου πωλητή Traveling Salesman Problem - TSP και πολλά άλλα.

Η ντετερμινιστική [4] εκδοχή του προβλήματος λύνεται εύκολα, ωστόσο, στον πραγματικό κόσμο, η **αβεβαιότητα** είναι συχνή και πρέπει να αντιμετωπιστεί. Για παράδειγμα, σε ένα δίκτυο μεταφορών, η διαδρομή που παρουσιάζει την μικρότερη διάρκεια ταξιδιού όταν δεν υπάρχει κίνηση, ενδέχεται να είναι επιρρεπής σε ατυχήματα και συμφόρηση. Έτσι μπορεί να οδηγήσει σε δραστική αύξηση του χρόνου ταξιδιού κατά τη διάρκεια της ώρας αιχμής. Σε περιπτώσεις με σημαντικά αυξημένη αβεβαιότητα, η ντετερμινιστική προσέγγιση μπορεί να είναι ακατάλληλη.

Ένα σημαντικό σύνολο μοντέλων και αλγορίθμων επίλυσης του προβλήματος συντομότερης διαδρομής έχουν αναπτυχθεί ώστε να είναι δυνατή η χρήση τους σε αυτές τις διάφορετικές εφαρμογές. Όσον αφορά το ίδιο το πρόβλημα, **οι πιο συνηθισμένοι τύποι προβλήματος συντομότερης διαδρομής** είναι[2]:

> Το **πρόβλημα ενός ζεύγους (One Pair Shortest Path - OPSP)**. Αναζητά να μια συντομότερη διαδρομή από ένα συγκεκριμένο κόμβο - πηγή σε ένα καθορισμένο κόμβο προορισμού.
>
> Το **πρόβλημα από μια αφετηρία (Single Source Shortest Path - SSSP)**. Απαιτεί τον υπολογισμό μιας συντομότερης διαδρομής από ένα καθορισμένο κόμβο πηγή προς κάθε άλλο κόμβο στο γράφημα.
>
> Το **πρόβλημα όλων των ζευγών (All Pairs Shortest Path - APSP)**. Επιζητά την εύρεση των συντομότερων μονοπατιών μεταξύ όλων των ζευγών των κόμβων.

Συχνά, δεν απαιτείται ο υπολογισμός του σύνολου των συντομότερων διαδρομών καθ΄ αυτών, αλλά μόνο των αποστάσεων μεταξύ των κόμβων. Μόλις οι αποστάσεις γίνουν γνωστές οι διαδρομές μπορούν να προκύψουν εύκολα. Άλλοι υποτύποι του προβλήματος ασχολούνται με τροποποιημένους περιορισμούς στις διαδρομές, δηλαδή παραμέτρους όπως το βάρος της συντομότερης διαδρομής από κόμβο σε κόμβο, το **βάρος** της συντομότερης διαδρομής από κόμβο σε κόμβο μέσω τρίτου κόμβου, το συντομότερο μονοπάτι απο κόμβο σε κόμβο και ούτω καθεξής. Περαιτέρω ταξινομήσεις αφορούν στο ίδιο το γράφημα εισόδου. Η αξιοποίηση γνωστών δομικών ιδιότητων των γραφων εισόδου μπορεί να να

οδηγήσει σε απλούστερους ή/και πιο αποδοτικούς αλγορίθμους.

Ωστόσο, όσο μεγαλύτερο είναι ένα δίκτυο και όσο πιο πολύπλοκη καθίσταται η δομή του, ανάλογος είναι και ο βαθμός δυσκολίας που υπεισέρχεται κατά τις διαδικασίες επίλυσής του. Ως εκ τούτου, αρκετά προβλήματα που αφορούν την επίλυση δικτύων που αναπαρίστανται διαγραμματικά ως γράφοι, παραμένουν ακόμη άλυτα είτε λόγω του τεράστιου υπολογιστικού χρόνου που η επίλυσή τους προϋποθέτει, είτε λόγω περιορισμών μνήμης[16].

Η συμβολή της **θεωρίας των γράφων** για την επίλυση προβλημάτων σε δίκτυα είναι σημαντική, καθώς η δομή ενός γράφου είναι κατάλληλη για τη μοντελοποίηση των δικτύων, των αντικειμένων τους και των συνδέσεων που υφίστανται μεταξύ των αντικειμένων αυτών.

Ορίζοντας έναν γράφο απλά [5, 6, 3] είναι η οπτική αναπαράσταση των σχέσεων που αναπτύσσουν ορισμένες ποσότητες, αναπαραστώμενες ως σημεία, σχεδιασμένες σε σχέση με ένα σύνολο αξόνων. Αντίστοιχος ορισμός, αναφερόμενος στην οπτική αναπαράσταση, αναγνωρίζει τον γράφο ως απεικόνιση αποτελούμενη από ένα σύνολο σημείων (κορυφών ή κόμβων) που συνδέονται με γραμμές (ακμές). Στους κατευθυνόμενους ή προσανατολισμένους γράφους οι ακμές απεικονίζονται διανυσματικά.

Μία άλλη εκδοχή ορισμού ενός γράφου είναι, ένα σύνολο από κόμβους (κορυφές) που ενώνονται μεταξύ τους με ακμές και χαρακτηρίζεται από τον τρόπο με τον οποίο συνδέονται οι κορυφές (κόμβοι). Αν οι ακμές προσανατολίζονται οριζόμενες από διατεταγμένα ζεύγη κόμβων, τότε ο γράφος αποκαλείται κατευθυνόμενος. Αν οι ακμές δεν προσανατολίζονται, οριζόμενες απλώς από διμελή σύνολα και όχι διατεταγμένα ζεύγη, τότε αποκαλείται μη κατευθυνόμενος ή απλός. Επιπλέον στοιχεία για τον ορισμό ενός γράφου είναι η σύνδεση των ακμών του με κάποια αξία, οπότε αποκαλείται σταθμισμένος.

Είναι γνωστό οτι η εξέλιξη των μαθηματικών ορίζει τον ρυθμό ανάπτυξης τόσο της πληροφορικής όσο και των λοιπών επιστημονικών πεδίων.Έτσι και σε αυτή την περίπτωση παρ΄ότι τα **θεμέλια της θεωρίας των γράφων** τέθηκαν από τα μαθηματικά, γρήγορα οι βασικές αρχές υιοθετήθηκαν από τον επιστημινικό κόσμο για την επίλυση και μοντελοποίηση προβλημάτων με τη βοήθεια γράφων.

Στις αρχές του 1950, πολλές μέθοδοι εφευρέθηκαν που θα μπορούσαν να θεωρηθούν αρχικές εκδόσεις ενός αλγορίθμου εύρεσης του συντομότερου μονοπατιού. Οι Pollack Wiebenson [7](σελ.128) αξιολόγησαν μερικές από τις μεθόδους αυτές. Αναγνωρίζουν τον αλγόριθμο του Dantzig, με βάση την μέθοδο simplex, ως την πρώτη λύση που δόθηκε όταν το πρόβλημα πρωτοεμφανίστηκε στη βιβλιογραφία. Περιγράφουν επίσης ένα πολύ πιο αποδοτικό αλγόριθμο (πιστωμένο στον Minty), που στις μέρες μας μπορεί να θεωρηθεί ως ένας αλγόριθμος ρύθμισης ετικέτων.

Κατά τη διάρκεια του δεύτερου μισού της δεκαετίας του 1960, νέοι αλγόριθμοι αναπτύχθηκαν, ενώ βελτιώθηκαν οι υπάρχοντες από πλευράς υπολογιστικής αποδοτικότητας.

Για παράδειγμα, εισήχθη μια νέα μέθοδος για να βρεθεί η συντομότερη διαδρομή μεταξύ δύο σημείων σε ένα δίκτυο. Μια συντομότερη διαδρομή βρίσκεται διερευνώντας μια επιλογή τροχιών από τη προέλευση καθώς και από τον προορισμό. Η **επιλογή των τροχιών** αποφασίζεται δυναμικά με την επέκταση, ένα προς ένα, των μονοπάτιων που έχουν καλύψει την απόσταση αποδεκτά, τουλάχιστον σε ότι αφορά στο χρόνο.

Η μέθοδος αυτή μπορεί να θεωρηθεί ως η πρώτη αμφίδρομη μέθοδος. Λίγο αργότερα ο Hart παράγει τον A* ο οποίος εξακολουθεί να είναι ένας διάσημος αλγόριθμος σήμερα, ειδικά στην περιοχή της τεχνητής νοημοσύνης.

Κατά τη διάρκεια του 1970 αναπτύχθηκαν πολλές νέες τεχνικές και αλγόριθμοι αναπτύχθηκαν για αρκετές παραλλαγές του προβλήματος. Κατά τη διάρκεια της δεκαετίας του 1980 και 1990, πολλή δουλειά έγινε σχετικά με τη βελτίωση των υφιστάμενων αλγορίθμων, π.χ. προτείνοντας μια σειρά νέων δομών δεδομένων. Εισήχθησαν εφαρμογές, που διατηρούσαν την υπολογιστική αποτελεσματικότητα χρήσης του προγενέστερου αλγορίθμου ο οποίος τρέχει σε πολυωνυμικό χρόνο. Άλλοι ερευνητές δημοσίευσαν αλγορίθμους για τους οποίους ήταν αβέβαιο αν τρέχουν ή όχι σε πολυωνυμικό χρόνο.

Κατά τη διάρκεια των τελευταίων δέκα ετών, η έρευνα σχετικά με το εν λόγω πρόβλημα έχει επικεντρωθεί κυρίως στην αναζήτηση της συντομότερης διαδρομής από μια πηγή σε έναν προορισμό σε μεγάλης κλίμακας οδικούς χάρτες. Η εισαγωγή συστημάτων πλοήγησης αυτοκινήτου έχει οδηγήσει σε πολλές νέες τεχνικές. Η περισσότερη από την έρευνα επικεντρώθηκε στην προεπεξεργασία του πρώτου δικτύου.

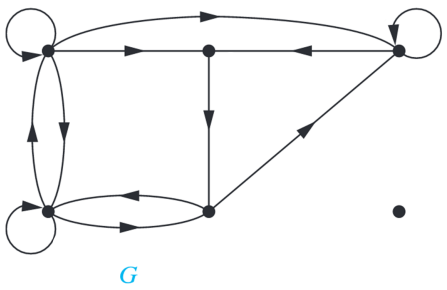
Κεφάλαιο 2

Γράφοι

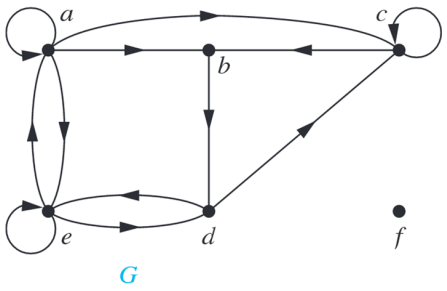
Οι γράφοι μπορούν να χρησιμοποιηθούν για την μοντελοποίηση φυσικών, βιολογικών, κοινωνικών και πληροφοριακών συστημάτων, με τρόπο που καθιστά ευκολότερη την κατανόηση και την ανάλυσή τους. Η πολυμορφικότητα, τους καθιστά κατάλληλους για αναπαράσταση **πάσης φύσεως δικτύων και συστημάτων**, καθώς και απαραίτητα εργαλεία για κάθε επιστημονικό κλάδο. Από τη πρωτοεμφάνισή τους, στο πρώτο μισό περίπου του 18ου αιώνα [1] μέχρι και σήμερα, από τον τομέα των διακριτών μαθηματικών, δεν έχει πάψει η εξέλιξη τους.

Σε αντίθεση με τη μελέτη μεμονωμένων χωρικών αντικειμένων, όπου οι σχέσεις μεταξύ τους **αναλύονται** με βασικό άξονα την εγγύτητα, η μελέτη χωρικών αντικειμένων που συναποτελούν ένα δίκτυο βασίζεται στις σχέσεις συνδεσιμότητας που αναπτύσσονται μεταξύ των επιμέρους στοιχείων του δικτύου. Εννοιολογικά, ένας γράφος σχηματίζεται από κορυφές και ακμές που συνδέουν τις κορυφές.[5]

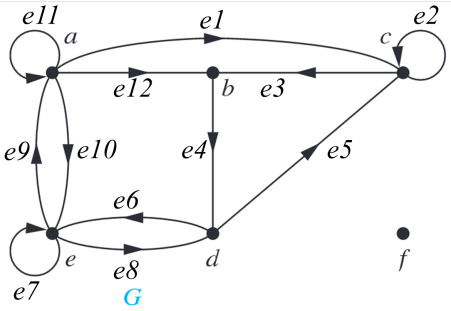
Τυπικά, ένα γράφημα είναι ένα ζεύγος συνόλων (V, E), όπου V είναι το σύνολο των κορυφών, ή αλλιώς σημείων ή κόμβων, και E είναι το σύνολο των ακμών, ή αλλιώς τόξων ή γραμμών, που σχηματίζονται από ζεύγη κορυφών. Το E είναι ένα πολλαπλό σύνολο, με άλλα λόγια, τα στοιχεία του μπορούν να συμβούν περισσότερες από μία φορές έτσι ώστε κάθε στοιχείο να έχει μια πολλαπλότητα[4, 1]. Συχνά, η επισήμανση των κορυφών γίνεται με γράμματα (για παράδειγμα: a, b, c,. . .) και αριθμούς (1, 2,. . .) ή και τα δύο μαζί(Σχήμα 2.1).



Σχήμα 2.1: Ορισμός ενός γράφου



Σχήμα 2.2: Ορισμός κορυφών γράφου



Σχήμα 2.3: Ορισμός ακμών γράφου

Στο Σχήμα 2.2 έχουμε V=$(a, b, ..., f)$ για τις κορυφές και στο 2.3
$E = (e1, e2, ..., e12) = [(a, c), (c, c), (c, b), (b, d),$
$(d, c), (d, e), (e, e), (e, d), (e, a), (a, e), (a, a)]$
για τις ακμές.

Αναλύοντας έναν γράφο[1]:

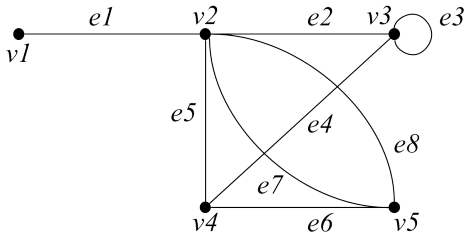
- Οι δύο κορυφές u και n είναι οι τελικές κορυφές της ακμής (u, v).
- Οι ακμές που έχουν τις ίδιες τελικές κορυφές είναι παράλληλες.
- Μια ακμή της μορφής (n, n) είναι ένας βρόχος.
- Ένας γράφος είναι απλός, αν δεν έχει παράλληλες ακμές ή βρόχους.
- Ένας γράφος χωρίς άκρα (δηλαδή τα E είναι κενά) είναι κενός.
- Ένας γράφος χωρίς κορυφές (δηλαδή V και E είναι κενά) είναι ένας μηδενικός γράφος.
- Ένας γράφος με μόνο μία κορυφή είναι ασήμαντος.
- Οι ακμές είναι όμορες, αν μοιράζονται μια κοινή κορυφή-άκρο.

- Δύο κορυφές u και n είναι όμορες εφόσον συνδέονται με μία ακμή, με άλλα λόγια, (u, v) είναι μια ακμή.
- Ο βαθμός της κορυφής v, γράφεται ως $d(v)$, είναι ο αριθμός των ακμών με τη v ως τελική κορυφή. Κατά συνθήκη, μετράμε ένα βρόχο δύο φορές ενώ οι παράλληλες ακμές συμβάλουν ξεχωριστά.
- Μια κορυφή εκκρεμές είναι μια κορυφή της οποίας ο βαθμός είναι 1
- Μια ακμή που έχει μια κορυφή εκκρεμές ως τελευταία κορυφή είναι μια ακμή εκκρεμές.
- Μια απομονωμένη κορυφή είναι μια κορυφή της οποίας ο βαθμός είναι 0.

Με βάση τα παραπάνω ένας γράφος G μπορεί να ορισθεί μαθηματικά ως $(V(G),E(G),w)$ που αποτελείται απο ένα μη κενό σύνολο κορυφών $V(G)$, ένα σύνολο $E(G)$ ακμών αποσυνδεδεμένο απο το $V(G)$, και μια συνάρτηση επίπτωσης W, που συσχετίζει με κάθε ακμή του G ένα μη διατεταγμένο ζεύγος κορυφών του G. Αν e μια ακμή και u και v οι κορυφές ώστε $w(e)=uv$, τότε η e θεωρείται οτι ενώνει τα u και v.

Δύο μαθηματικά παραδείγματα γράφων που θα βοηθήσουν στην κατανόηση των παραπάνω είναι τα ακόλουθα:

Παράδειγμα 1



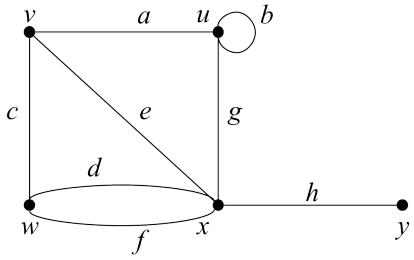
Σχήμα 2.4: Ο γράφος G

$G = (V(G), E(G), w)$ όπου
$V(G) = v_1, v_2, v_3, v_4, v_5$
$E(G) = e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8$

ενώ η w ορίζεται ως:

$w(e_1) = v_1v_2, w(e_2) = v_2v_3, w(e_3) = v_3v_3, w(e_4) = v_3v_4,$
$w(e_5) = y_2v_4, w(e_6) = v_4v_5, w(e_7) = v_5v_2, w(e_8) = v_2v_5,$

Παράδειγμα 2



Σχήμα 2.5: Ο γράφος H

$H = (V(H), E(H), w)$
όπου
$V(H) = u, v, w, x, y$
$E(H) = a, b, c, d, e, f, g, h$
ενώ η w ορίζεται ως:

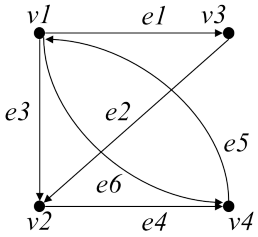
$w(a) = uv, w(b) = uu, w(c) = uw, w(d) = wx,$
$w(e) = ux, w(f) = wx, w(g) = ux, w(h) = xy,$

2.1 Τύποι γράφων

Εν γένει, οι γράφοι ταξινομούνται σε επιμέρους κατηγορίες βάση συγκεκριμμένων κριτηρίων που όμως τίθενται ανά περίπτωση, βάση των δεδομένων του προβλήματος. Οι κυριότερες κατηγορίες εξ΄ αυτών είναι οι ακόλουθες[5, 3, 8]:

Απλός γράφος. Ένας απλός γράφος ορίζεται ως ένα διάγραμμα $G = (V,\ E)$ αποτελούμενο από κόμβους και ακμές, ενώ μεταξύ δύο κόμβων του γράφου υφίσταται μία και μόνο μία ακμή. Ένας απλός γράφος δεν περιλαμβάνει κυκλικές ακμές. Επίσης στις ακμές δεν ορίζεται καμία κατεύθυνση μετακίνησης από κόμβο σε κόμβο.

Κατευθυνόμενος γράφος. (Σχήμα 2.6) Ένας γράφος ορίζεται ως κατευθυνόμενος γράφος $G = (V,\ A)$ όταν οι ακμές που συνδέουν τους κόμβους του είναι προσανατολισμένες προς μια κατεύθυνση (φορά), οπότε και αυτές με τη σειρά τους χαρακτηρίζονται ως κατευθυνόμενες ακμές ή τόξα. Ένα τόξο $a = (x,\ y)$ που ανήκει σε έναν κατευθυνόμενο γράφο, έχει κατεύθυνση από τον κόμβο x προς τον κόμβο y. Ο κόμβος y καλείται άμεσος διάδοχος του x και ο κόμβος x άμεσος προκάτοχος του κόμβου y.



Σχήμα 2.6: Κατευθυνόμενος γράφος

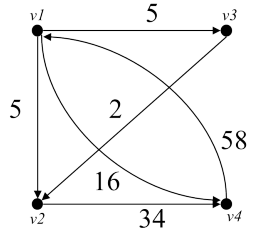
Μεικτός γράφος. Μεικτός καλείται ένας γράφος $G = (V, E, A)$ ο οποίος είναι δυνατό να περιλαμβάνει ταυτόχρονα κατευθυνόμενες και μη κατευθυνόμενες ακμές. Οι γράφοι αυτού του είδους συνιστούν ειδική περίπτωση γράφου.

Συνδεδεμένος και μη-συνδεδεμένος γράφος. Ως συνδεδεμένος χαρακτηρίζεται ένας γράφος, στον οποίο υπάρχουν ένα ή περισσότερα μονοπάτια μέσω των οποίων συνδέονται δύο οποιοιδήποτε κόμβοι του γράφου. Ως μη συνδεδεμένος χαρακτηρίζεται ένας γράφος, στον οποίο δεν υφίσταται απαραίτητα σύνδεση μεταξύ του συνόλου των κόμβων που περιλαμβάνονται στο γράφο.

Κανονικός γράφος. Ένας γράφος ορίζεται ως κανονικός όταν όλοι οι κόμβοι του έχουν ίσο αριθμό γειτονικών κόμβων. Στην περίπτωση αυτή, όλοι οι κόμβοι του γράφου έχουν ακριβώς τον ίδιο βαθμό.

Πλήρης γράφος. Καλούνται οι γράφοι όπου μεταξύ κάθε ζεύγους κόμβων υφίσταται οπωσδήποτε μια σύνδεση. Οι γράφοι αυτής της κατηγορίας, περιλαμβάνουν το μέγιστο δυνατό αριθμό ακμών.

Βεβαρημένος γράφος. (Σχήμα: 2.7) Είναι ο γράφος στις ακμές του οποίου ανατίθενται βάρη. Τα βάρη αυτά μπορεί να αναπαριστούν το κόστος μιας διαδρομής, το μήκος της, το χρόνο που απαιτείται για τη συνολική διάσχιση της συγκεκριμένης διαδρομής κ.λ.π.



Σχήμα 2.7: Βεβαρημένος Γράφος

Επίπεδος γράφος. Ένας γράφος καλείται επίπεδος όταν μπορεί να σχεδιαστεί στο επίπεδο χωρίς να διασταυρώνονται οι πλευρές του.

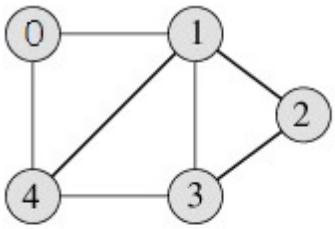
Διμερής γράφος. Ένας γράφος καλείται διμερής όταν οι κορυφές του μπορούν να διαιρεθούν σε δύο σύνολα έτσι ώστε κάθε στοιχείο του ενός να συνδέεται με κάποιο στοιχείο του άλλου.

Κυκλικός γράφος. Ένας γράφος καλείται κυκλικός όταν αποτελείται από έναν κύκλο, δηλαδή έναν κόμβο και μια κυκλική ακμή όπου η αρχή και το πέρας της είναι ο μοναδικός κόμβος του γράφου. Επίσης, ένας κυκλικός κόμβος ορίζεται και ως μια κλειστή αλυσίδα ακμών που συνδέονται μεταξύ τους, ενώ ο κόμβος αφετηρίας της πρώτης ακμής είναι ο ίδιος με τον κόμβο πέρατος της τελευταίας ακμής.

2.2 Αναπαράσταση γράφων

Κατά βάση, υπάρχουν δύο μέθοδοι αναπαράστασης γράφων στους υπολογιστές, η μήτρα ή ο πίνακας γειτνίασης και η λίστα γειτνίασης.[3](σελ.669),[4](σελ.589), [2](σελ.8)

Η **μήτρα γειτνίασης** είναι ένας δισδιάστατος πίνακας διαστάσεων V επί V, όπου V ο αριθμός των ακμών του γράφου. Για έναν απλό γράφο, αν ισχύει ότι adj[i][j] $= 1$ τότε υπάρχει μια ακμή που να συνδέει το κόμβο i με τον κόμβο j. Αντίστοιχα, αν η τιμή είναι μηδενική τότε δεν υπάρχει τέτοια ακμή. Τιμές μεγαλύτερες του 1 υποδηλώνουν οτι υπάρχουν περισσότερες της μιας ακμές που οι κορυφές αυτές πρόσκεινται. Το παράδειγμα που ακολουθεί παρουσιάζει αυτή την αναπαράσταση ενός απλού γράφου:



	0	1	2	3	4
0	0	1	0	0	1
1	1	0	1	1	1
2	0	1	0	1	0
3	0	1	1	0	1
4	1	1	0	1	0

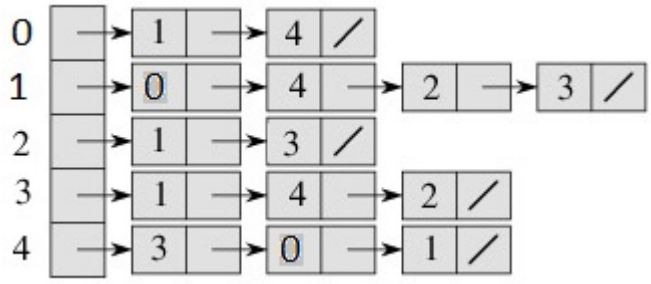
Σχήμα 2.8: Βεβαρημένος Γράφος

Στις **περιπτώσεις άλλων τύπων γράφων** η αναπαράσταση προσαρμόζεται κατάλληλα. Για παράδειγμα, στην αναπαράσταση ενός μη βεβαρημένου κατευθυνόμενου

γράφου, το ψηφίο 0 αντικαθίσταται με το σύμβολο του απείρου για την δήλωση των ακμών που δεν υπάρχουν και συνεπώς δεν είναι δυνατόν να ακολουθηθούν.

Στην **περίπτωση βεβαρημένων γράφων** το ψηφίο 1 αντικαθίσταται από το βάρος της διαδρομής (κόστος) εάν αυτό υπάρχει. Σε περίπτωση που δεν υπάρχει μια ακμή εισέρχεται το ψηφίο 0 (μη βεβαρημένος γράφος) ή το το άπειρο (βεβαρημένος γράφος), ώστε να υποδηλωθεί η απουσία της.

Στην **περίπτωση της λίστας γειτνίασης**, χρησιμοποιείται ένα διάνυσμα από συνδεδεμένες λίστες. Το μέγεθος του διανύσματος είναι ίσο με τον αριθμό των κορυφών. Αν υποτεθεί οτι το διάνυσμα είναι το array[], μια πληροφορία τύπου array[i] αναπαριστά τη συνδεδεμένη λίστα κορυφών που πρόσκεινται στην ακμή i Για τον γράφο του παραδείγματος, η αναπαράσταση με λίστα γειτνίασης είναι η ακόλουθη:



Σχήμα 2.9: Παράδειγμα απεικόνισης με λίστα γειτνίασης

Στα **θετικά της μεθόδου αναπαράστασης με πίνακα**[4](σελ.591) είναι η συμπυκνωμένη μορφή καθώς και η ενίσχυση του "*locality of reference*", της σωστής δηλαδή κατανομής τύπων αρχείων στη μνήμη. Στα μειονεκτήματα είναι η καθυστέρηση εύρεσης γειτονικών ακμών σε μια δεδομένη κορυφή καθως πρέπει να σκαναριστεί ολόκληρη η σειρά,πράγμα που καταλαμβάνει χώρο ανάλογο του αριθμού κορυφών ολόκληρου του γράφου.

Στα **θετικά της μεθόδου αναπαράστασης με λίστα**[4](σελ.591), είναι η ευκολία στην εφαρμογή και παρακολούθηση της, αφού η εύρεση γειτονικών ακμών σε μια δεδομένη κορυφή είναι τόσο απλή όσο το να διαβάσει κανείς τη λίστα. Στα μειονεκτήματα είναι η καθυστέρηση εύρεσης των ακμών μεταξύ δυο δεδομένων κορυφών που είναι ανάλογη του ελάχιστου βαθμού των δύο αυτών κορυφών, καθώς η χρήση λίστας για αναπαράσταση είναι βέλτιστη μόνο σε αραιούς γράφους.

Κεφάλαιο 3

Πρόβλημα εύρεσης ελάχιστης διαδρομής

Σε αυτό το κεφάλαιο εξετάζεται το πρόβλημα εύρεσης της συντομότερης διαδρομής μεταξύ των κορυφών σε ένα γράφο. Αποτυπώνεται το πρόβλημα στη γενική του μορφή με μια μαθηματική αποτύπωση και παρουσιάζονται εν συντομία οι αλγόριθμοι επίλυσης του.

Το πρόβλημα μπορεί να προκύψει για παράδειγμα στην κατάρτιση του πίνακα αποστάσεων μεταξύ δύο πόλεων, σε όλους του συνδυασμούς, σε έναν άτλα οδικού δικτύου.

Το πρόβλημα του συντομότερου μονοπατιού μπορεί να οριστεί για γράφους που είναι είτε προσανατολισμένοι είτε μη-προσανατολισμένοι.

Θεωρείται ότι δίδεται ένας **βεβαρημένος γράφος** $G = (V, E)$ με συνάρτηση βάρους $w : E \to \Re$ που αντιστοιχεί τις κορυφές του χάρτη με πραγματικά βάρη. Το ζητούμενο είναι η εύρεση για κάθε **ζεύγος κορυφών**, u, v το συντομότερο μονοπάτι (δηλαδή αυτό με το μικρότερο βάρος) από το u στο v, όπου το βάρος της διαδρομής είναι το άθροισμα των βαρών των περιεχόμενων ακμών.

Τυπικά, είναι επιθυμητή η μορφή ενός πίνακα: η τιμή που αντιστοιχεί στην γραμμή των u και στήλη των v θα πρέπει να είναι η συντομότερη διαδρομή από το u στο v. Μια ελάχιστη διαδρομή είναι αυτή που παρέχει το μικρότερο συνολικό βάρος στην διαδρομή μεταξύ δύο δεδομένων κορυφών. Για μια δεδομένη διαδρομή (v_1, v_2, v_3,...,v_m), m κόμβων, το μήκος δίνεται ως[9]:

$$w_{i,j} = \sum_{i=1}^{m-1} w_{v_i, v_{i+1}} \tag{3.1}$$

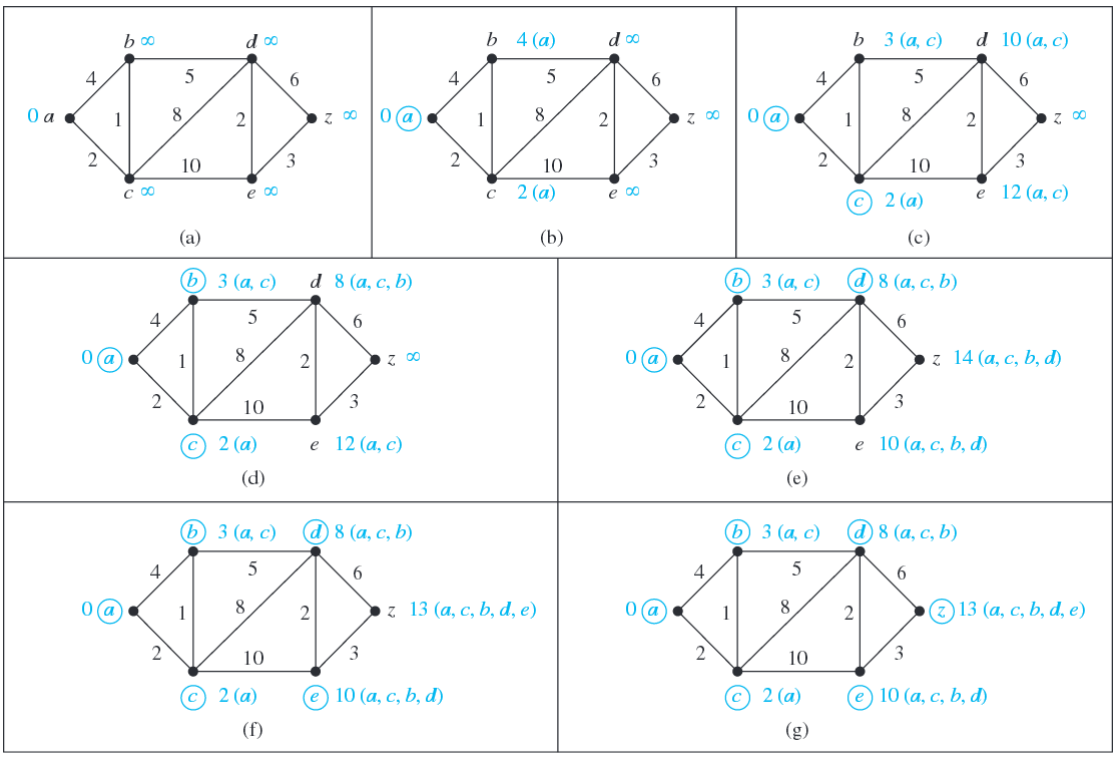
όπου $w_{i,j}$ το συνολικό μήκος (βάρος) και $w_{v_n, v_{n_{i+1}}}$ το βάρος από τη κορυφή v_i στην κορυφή v_{i+1} αντίστοιχα.

Για την επίλυση του προβλήματος της ελάχιστης διαδρομής, θεωρούμε τις παρακάτω παραδοχές:

Τα κόστη κάθε ακμής είναι ακέραιοι αυτή η απαίτηση εφαρμόζεται σε ορισμένους μόνο από τους αλγορίθμους επίλυσης, ένας από αυτούς είναι και ο αλγόριθμος της εργασίας. Σε περίπτωση που το πραγματικό κόστος της ακμής είναι ένας πραγματικός αριθμός με δεκαδικά ψηφία, μπορούμε να τον μετατρέψουμε σε ακέραιο με τον πολλαπλασιασμό του

με έναν κατάλληλο αριθμό. Φανταστικές τιμές σποκλείονται καθώς εισάγουν περιττές επιπλοκές όσον αφορά στην παράσταση των γράφων σε υπολογιστική απεικόνιση[13].

Υπάρχει μια κατευθυνόμενη διαδρομή μεταξύ των ζευγών των κορυφών υπό εξέταση ενώ ο γράφος δεν περιλαμβάνει αρνητικούς κύκλους. Το πρόβλημα της συντομότερης διαδρομής με αρνητικούς κύκλους είναι NP-hard (αδύνατο να συνταχθεί με πολυωνυμικό αλγόριθμο).



Σχήμα 3.1: Παράδειγμα υπολογισμού συντομότερης διαδρομής

Στην περίπτωση μη-κατευθυνόμενου γράφου με μη αρνητικά βάρη, είναι εύκολο να μετατραπεί σε κατευθυνόμενο γράφο.

Μπορούμε να λύσουμε ένα τέτοιο πρόβλημα με τη λειτουργία ενός κώδικα αλγορίθμου συντομότερης διαδρομής μοναδικής πηγής για $|V|$ φορές, μία φορά για κάθε κορυφή ως πηγή [3].

Αν όλα τα βάρη των ακμών είναι μη αρνητικά, μπορούμε να χρησιμοποιήσουμε τον αλγόριθμο του Dijkstra.

Αν χρησιμοποιηθεί η **εφαρμογή γραμμικής σειράς της ουράς ελάχιστης προτεραιότητας**, ο χρόνος εκτέλεσης είναι $O(V^3 + VE) = O(V^3)$[3](σελ.553). Η εφαρμογή του **δυαδικού ελάχιστου σωρού της ουράς ελάχιστης προτεραιότητας**[10, 11] αποδίδει ένα χρόνο λειτουργίας $O(VElogV)$, κάτι που αποτελεί μια βελτίωση, αν το γράφημα είναι αραιό. Εναλλακτικά, μπορεί να εφαρμοστεί η ουρά ελάχιστης προτεραιότητας με σωρό Fibonacci, αποδίδοντας ένα χρόνο λειτουργίας $O(V^2logV + VE)$.

Αν επιτρέπονται αρνητικά βάρη στις ακμές, δεν μπορεί πλέον να χρησιμοποιηθεί ο αλγόριθμος του Dijkstra. Αντ άυτού, πρέπει να εκτελεστεί ο πιο αργός Bellman-Ford [10]αλγόριθμος μία φορά από κάθε κορυφή. Ο χρόνος της προκύπτουσας λειτουργίας είναι $O(V^2E)$, ο οποίος σε ένα πυκνό γράφημα είναι $O(V^4)$.

Εδώ να σημειώσουμε ότι στην επιστήμη της πληροφορικής[4][3](σελ.205) το $Big\ O$ περιγράφει την απόδοση ή την πολυπλοκότητα ενός αλγορίθμου. Το $Big\ O$ χρησιμοποιείται για να περιγράψει τον μέγιστο χρόνο εκτέλεσης ενός αλγορίθμου ή το χώρο κατανάλωσης μνήμης στον εκάστοτε δίσκο.

Σε αντίθεση με τους αλγόριθμους από μία μόνο πηγή, που προϋποθέτουν μια λίστα γειτνίασης για την εκπροσώπηση του γράφηματος, οι περισσότεροι από τους αλγορίθμους σε χρήση για τη λύση του προβλήματος χρησιμοποιούν μια αναπαράσταση γειτνίασης με μορφή μήτρας.

Για λόγους ευκολίας, υποθέτουμε ότι οι κορυφές αριθμούνται $1, 2, ..., |V|$, έτσι ώστε η είσοδος είναι ένας $n \times n$ πίνακας W που αντιπροσωπεύει τα βάρη των ακμών ενός κατευθυνόμενου γράφηματος n-κορυφών $G = (V, E)$. Δηλαδή :

$$\omega_{i,j} = \begin{cases} 0 & \text{αν } i = j \text{ δηλαδή η διαγώνιος} \\ \text{βάρος εκάστοτε κατευθυνόμενης ακμής } w_{ij} & \text{αν } i \neq j, (i,j) \in E \\ \infty & \text{αν ι} \neq j, (i,j) \notin E \end{cases} \tag{3.2}$$

Το πρόβλημα καλείται μερικές φορές και ως πρόβλημα συντομότερου μονοπατιού ενός ζεύγους ώστε να διαφοροποιηθεί από τις παρακάτω περιπτώσεις άλλων προβλημάτων συντομότερου μονοπατιού[2]:

Μονής πηγής στο οποίο απαιτείται να βρεθούν τα συντομότερα μονοπάτια από έναν κόμβο πηγή προς όλους τους υπόλοιπους κόμβους στο γράφο.

Όλων των ζευγαριών, στο οποίο απαιτείται να βρεθούν τα συντομότερα μονοπάτια μεταξύ κάθε ζεύγους κόμβων στο γράφο.

3.1 Αλγόριθμοι επίλυσης

Παρακάτω περιγράφονται μερικοί από τους κυριότερους αλγόριθμους διάσχισης γράφων, που χρησιμοποιούνται για την επίλυση του προβλήματος της συντομότερης διαρομής, τα βασικά χαρακτηριστικά τους, η δομή και η λειτουργία τους [5] (σελ.25) [8](σελ.60) .

Ανάλογα με τις ιδιαιτερότητες και τις απαιτήσεις του εκάστοτε προβλήματος, εφαρμόζεται κατά περίπτωση ο αλγόριθμος που εξασφαλίζει την αποδοτικότερη διαχείρισή του. Αυτός είναι άλλωστε και ο λόγος που οι αλγόριθμοι διάσχισης γράφων εμφανίζονται διαφοροποιημένοι ως προς τον τύπο δεδομένων που κάθε φορά επεξεργάζονται και ως προς τις διαδικασίες επεξεργασίας των δεδομένων αυτών.

Γενικός αλγόριθμος - Generic algorithm[5](σελ.25). Η λειτουργία του γενικού αλγορίθμου βασίζεται στον επαναληπτικό έλεγχο των ακμών από τη κορυφή υπό εξέταση i και στην ρύθμιση της ετικέτας για τη κορυφή j, στην οποία μια δεδομένη ακμή

τερματίζει:

Για την αποθηκεύση των κορυφών που πρέπει να ελεγχθούν, χρησιμοποιείται ο κατάλογος V, που ονομάζεται λίστα υποψηφίων. Ο τρόπος με τον οποίο οι κορυφές που ανήκουν στη διαδρομή αποθηκεύονται στον κατάλογο αυτό, καθώς επίσης και η μέθοδος για τον καθορισμό της προσθήκης και της ανάκτησης των κορυφών από και προς αυτόν, είναι συχνά ο κύριος παράγοντας που διακρίνει τους διαφόρους επιμέρους αλγορίθμους υπό εξέταση. Στην περίπτωση του γενικού αλγόριθμου, ο κατάλογος των υποψηφίων είναι μια ουρά FIFO στην οποία εκτελούνται οι πράξεις της προσθήκης και της ανάκτησης μιας κορυφής για την αρχή ή το τέλος της διαδρομής.

Αναζήτηση κατά πλάτος - Breadth-First Search - BFS[18](σελ.26) [5](σελ.38). Ο αλγόριθμος BFS εκτελεί αναζήτηση ανά κόμβο του γράφου. Σε κάθε κόμβο η μόνη κίνηση που επιτρέπεται είναι προς τους όμορους κόμβους. Συνεπώς, εκκινώντας απο μια δεδομένη αρχή, ο αλγόρθμος επισκέπτεται τους γειτονικούς του. Έπειτα σε κάθε νέο κόμβο, ο αλγόριθμος συνεχίζει να επισκέπτεται τους γειτονικούς του, με μόνη διαφορά ότι πλέον επισκέπτεται μόνο αυτούς που δεν έχει ήδη επισκεπτεί. Παρακάτω παρουσιάζεται γραφικά η λειτουργία του αλγορίθμου.

Αναζήτηση κατά βάθος - Depth-First Search - DFS[18](σελ.26) [5](σελ.38). Ο αλγόριθμος αυτός είναι παρεμφερής του BFS. Η διαφορά έγκειται στο ότι ο BFS εκτελεί την έρευνα του διασχίζοντας κάθε φορά μια διαδρομή με εκκίνηση τον κόμβο αρχής και αφού υπολογίσει το βάρος του μονοπατιού επιστρέφει στη αρχή για να εκτελέσει την επόμενη διαδρομή απο τον ίδιο κόμβο αρχής. Έπειτα εκκινεί την ίδια διαδικασία απο τους όμορους αυτού κόμβους.

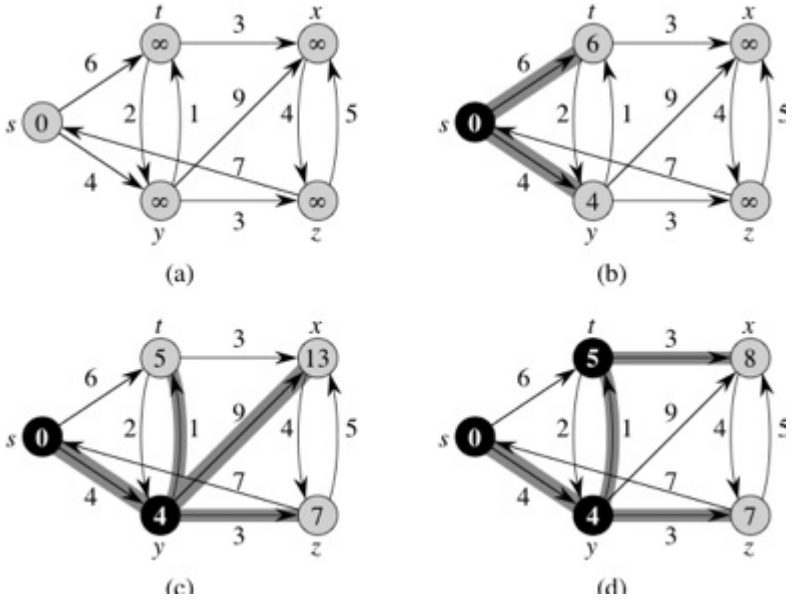
Αλγόριθμος Dijkstra[4](σελ.658)[10][18](σελ.45). Ο αλγόριθμος του Dijkstra είναι πιθανώς ο πιο γνωστός αλγόριθμος για την εύρεση της συντομότερης διαδρομής σε ένα κατευθυνόμενο γράφο. Η βασική διαφορά μεταξύ αυτού του αλγόριθμου και του γενικού αλγόριθμου είναι ο τρόπος με τον οποίο οι κορυφές επιλέγονται από τον κατάλογο των υποψηφίων. Η επιλεγμένη κορυφή είναι η κορυφή που έχει τη μικρότερη από όλες τις ετικέτες από τις διαθέσιμες κορυφές στον κατάλογο:

$$d_i = min(d_j) \tag{3.3}$$

Αυτό οδηγεί την κορυφή με δεδομένο βάρος, καθώς και όλες τις κορυφές που είναι στη διαδρομή από την αρχική ως τη επιλεγμένη κορυφή, να έχουν την ελάχιστη τιμή βάρους και να μην προστεθούν πάλι στον κατάλογο υποψηφίων. Ο συνολικός αριθμός των επαναλήψεων που ο αλγόριθμος του Dijkstra πρέπει να εκτελέσει για να επιλύσει το πρόβλημα συντομότερης διαδρομής είναι $O(V^2)$

Για ένα δεδομένο κόμβο-πηγή σε ένα γράφο ο αλγόριθμος βρίσκει το μονοπάτι με το ελάχιστο κόστος μεταξύ αυτού του κόμβου και οποιουδήποτε άλλου κόμβου στο γράφο. Μπορεί να χρησιμοποιηθεί, επίσης, για να βρεθούν τα κόστη των συντομότερων μονοπατιών από ένα μονό κόμβο αρχής σε ένα μονό κόμβο προορισμού σταματώντας τον αλγόριθμο με το που εντοπιστεί ο κόμβος προορισμού.

Αν οι κόμβοι του γράφου αναπαριστούν πόλεις και τα κόστη των ακμών αναπαριστούν αποστάσεις μεταξύ δύο πόλεων που συνδέονται μέσω δρόμου, ο αλγόριθμος Dijkstra μπορεί να χρησιμοποιηθεί για να βρει τη συντομότερη διαδρομή μεταξύ μιας πόλης και όλων των υπολοίπων πόλεων. Ως αποτέλεσμα, ο αλγόριθμος συντομότερου μονοπατιού χρησιμοποιείται ευρέως σε πρωτόκολλα δρομολόγησης δικτύων.



Σχήμα 3.2: Παράδειγμα αναζήτησης κατά πλάτος Dijkstra

Δεν είναι δυνατό να μειωθεί ο αριθμός των επαναλήψεων που εκτελούνται για να ελεγθούν οι ετικέτες, διότι αυτό δεν θα επέτρεπε να διασφαλιστεί η εύρεση της βέλτι-

στης λύσης. Κάθε ακμή θα πρέπει να ελέγχεται τουλάχιστον μία φορά. Η επιλογή μιας βέλτιστης δομής δεδομένων που αντιπροσωπεύει τον κατάλογο των υποψηφίων καθιστά δυνατή, με τη σειρά της, την αισθητή μειώση της υπολογιστικής πολυπλοκότητας της διαδικασίας επιλογής μιας κορυφής από το κατάλογο των υποψηφίων. Σωροί (επίσης γνωστοί ως ουρές προτεραιότητας) μπορούν να εξυπηρετήσουν ιδανικά το σκοπό αυτό. Χρησιμοποιώντας σωρούς Fibonacci μπορούμε να λύσουμε το πρόβλημα εύρεσης συντομότερης διαδρομής με τη χρήση του Dijkstra εκτελώντας $O(A + nlog(n))$ πράξεις.

Αλγόριθμος Bellman-Ford[5](σελ.39)[10][18](σελ.52). Ο αλγόριθμος Bellman-Ford ανήκει στους αλγόριθμους διόρθωσης της τιμής του βάρους που θεωρούν όλα τα βάρη για τις αποστάσεις των κορυφών σαν προσωρινά μέχρι την τελευταία επανάληψη, μετά την οποία όλα τα βάρη τίθενται σε βέλτιστες τιμές.

Αυτός ο αλγόριθμος παρέχει μια δυνατότητα για την επίλυση του προβλήματος συντομότερης διαδρομής σε γράφους με αρνητικά βάρη ακμών. Στη περίπτωση που διαπιστωθεί η ύπαρξη αρνητικού κύκλου, ο αλγόριθμος αποδίδει το ψεύδος ως αποτέλεσμα της λειτουργίας του. Σε αυτήν την περίπτωση ο αλγόριθμος Bellman-Ford μπορεί να εντοπίσει αρνητικούς κύκλους και να αναφέρει την ύπαρξή τους. Αυτός ο αλγόριθμος εκτελεί $V-1$ επαναλήψεις στις οποίες ελέγχει A ακμές. Η υπολογιστική πολυπλοκότητα του είναι ίση με $O(VA)$.

Ο αλγόριθμος υπολογίζει τα συντομότερα μονοπάτια από μία εκκινηση προς όλους τους υπόλοιπους κόμβους. Όπως φαίνεται και απο την έκφραση της πολυπλοκότητας του είναι βραδύτερος από τον αλγόριθμο Dijkstra για το ίδιο πρόβλημα, αλλά πιο ευέλικτος.

Όπως και στον αλγόριθμο Dijkstra, ο αλγόριθμος Bellman-Ford βασίζεται στο αξίωμα της χαλάρωσης, στο οποίο μία προσέγγιση της σωστής απόστασης αντικαθίσταται σταδιακά από περισσότερο ακριβείς τιμές μέχρι τελικά να βρεθεί η βέλτιστη λύση.

Και στους δύο αλγορίθμους η κατά προσέγγιση απόσταση του κάθε κόμβου είναι πάντα υπερεκτιμημένη σε σχέση με την πραγματική τιμή, και αντικαθίσταται από την ελάχιστη τιμή ενός νέου μονοπατιού κάθε φορά. Η διαφορά μεταξύ των αλγορίθμων αυτών έγκειται στο γεγονός ότι ο αλγόριθμος Dijkstra επιλέγει με άπληστη μέθοδο τον κόμβο με την ελάχιστη απόσταση ο οποίος δεν έχει ακόμα υποστεί επεξεργασία και εφαρμόζει την διαδικασία χαλάρωσης σε όλες τις γειτονικές ακμές. Από την άλλη, ο αλγόριθμος Bellman-Ford απλά εφαρμόζει τη διαδικασία χαλάρωσης σε όλες τις ακμές κάνοντάς το V-1 φορές, όπου V ο αριθμός των κόμβων στο γράφο.

Αλγόριθμος A*[18](σελ.48). Ένας αλγόριθμος A* είναι μια πιο γενική προσέγγιση σε σχέση με τον αλγόριθμο του Dijkstra για την εύρεση της συντομότερης διαδρομής μεταξύ δύο κόμβων σε ένα γράφο. Όπως αναφέρθηκε παραπάνω, ο αλγόριθμος του Dijkstra πραγματοποιεί πρώτα μια αναζήτηση breadth-first όπου ο επόμενος κόμβος που πρέπει να εξεταστεί είναι αυτός με την ελάχιστη απόσταση από τον κόμβο αφετηρίας (που αναφέρονται στο πίνακα αποστάσεων ή τη λίστα υποψηφίων).

Ο A* αλγόριθμος εισάγει μια διαδικασία για να προσδιορίσει τη σειρά με την οποία οι κόμβοι επιλέγονται κατά τη διαδικασία αναζήτησης. Αυτή η ευρετική είναι ένα άθροισμα δύο όρων. Ο πρωτος όρος είναι η απόσταση απο τον τρέχοντα k κόμβο (δίνεται ως $Dist[k]$), ενώ ο δεύτερος είναι μια εκτίμηση της απόστασης από τον κόμβο προορισμού j, που συνήθως συμβολίζεται ως $h(k)$. Στις περισσότερες υλοποιήσεις το $h(k)$ υπολογίζεται

ως η Ευκλείδεια απόσταση από τον εξεταζόμενο κόμβο προς τον κόμβο προορισμού. Εάν οι συντεταγμένες όλων των κόμβων είναι διαθέσιμες, η απόσταση μπορεί π.χ. να υπολογιστεί με το θεώρημα του Πυθαγόρα. Ο αλγόριθμος του Dijkstra μπορεί να θεωρηθεί ως ειδική περίπτωση του αλγόριθμου A* όπου $h(k) = 0$ για όλους τους k κόμβους. Ο A* χρησιμοποιεί μια best-first αναζήτηση και βρίσκει ένα μονοπάτι ελαχίστου κόστους από ένα δεδομένο αρχικό κόμβο σε έναν κόμβο στόχου (από έναν ή περισσότερους πιθανούς στόχους). Δεδομένου ότι ο A* διασχίζει το γράφημα, δημιουργεί ένα δέντρο μερικών διαδρομών. Τα φύλλα αυτού του δέντρου (που ονομάζεται ανοικτό σύνολο ή περιθωριακό) αποθηκεύονται σε μια ουρά προτεραιότητας που διατάσει τους κόμβους φύλλα βάσει μιας συνάρτησης κόστους, η οποία συνδυάζει μια ευρετική εκτίμηση του κόστους για την επίτευξη του στόχου και την απόσταση που διανύθηκε από τον αρχικό κόμβο. Συγκεκριμένα, η συνάρτηση κόστους είναι:

$$f(n) = g(n) + h(n) \tag{3.4}$$

Εδώ, $g(n)$ είναι το γνωστό κόστος μετακίνησης απο τον αρχικό κόμβο στον κόμβο n. Αυτή η τιμή παρακολουθείται από τον αλγόριθμο. Το $h(n)$ είναι μια ευρετική εκτίμηση του κόστους για τη μετακίνηση απο τον n σε οποιοδήποτε κόμβο στόχο. Για τον αλγόριθμο για την εύρεση της πραγματικής συντομότερης διαδρομής, η ευρετική συνάρτηση πρέπει να είναι παραδεκτή, πράγμα που σημαίνει ότι ποτέ δεν υπερεκτιμά το πραγματικό κόστος για τη μετακίνηση προς τον πλησιέστερο κόμβο στόχου. Η ευρετική λειτουργία είναι εξαρτώμενη από το πρόβλημα που λύνεται και πρέπει να παρέχεται από το χρήστη του αλγορίθμου.

Για παράδειγμα, σε μια εφαρμογή, όπως η δρομολόγηση, το $h(x)$ μπορεί να αντιπροσωπεύει την ευθεία απόσταση προς την αντίπαλη μεριά, δεδομένου ότι είναι φυσικά η μικρότερη δυνατή απόσταση μεταξύ δύο σημείων.

Αν η ευρετική h ικανοποιεί την πρόσθετη προϋπόθεση:

$$h(x) \leq d(x, y) + h(y) \tag{3.5}$$

για κάθε άκρο (x, y) του γραφήματος (όπου d δηλώνει το μήκος της εν λόγω ακμής), τότε η h ονομάζεται μονότονη, ή συνεπής. Σε μια τέτοια περίπτωση, ο A* μπορεί να εφαρμοστεί πιο αποτελεσματικά. Σε γενικές γραμμές, κανένας κόμβος δεν πρέπει να υποβληθεί σε επεξεργασία περισσότερο από μία φορά και ο A* είναι ισοδύναμος στη διαπέραση του με τον αλγόριθμο Dijkstra με το μειωμένο κόστος:

$$d'(x, y) = d(x, y) + h(y) - h(x) \tag{3.6}$$

Όπως όλοι οι αλγόριθμοι αναζήτησης με ενημέρωση, ψάχνει πρώτα τις διαδρομές που φαίνεται να είναι πιο πιθανό να οδηγήσουν προς τον στόχο.

Ξεκινώντας με τον αρχικό κόμβο, διατηρεί μια ουρά προτεραιότητας των κόμβων που θα διέλθουν, γνωστή ως ανοικτό σύνολο ή περιθώριο. Όσο μικρότερη η τιμή του $f(x)$ για ένα δεδομένο κόμβο x τόσο υψηλότερη η προτεραιότητα του. Σε κάθε βήμα του αλγορίθμου, ο κόμβος με το χαμηλότερο $f(x)$ αφαιρείται από την ουρά, οι τιμές f και g των γειτόνων του ενημερώνονται αναλόγως, και αυτοί οι γείτονες προστίθενται στην ουρά. Ο αλγόριθμος συνεχίζεται μέχρις ότου ένας κόμβος στόχος να έχει μια χαμηλότερη τιμή f από οποιονδήποτε άλλο κόμβο στην ουρά (ή έως ότου η ουρά να είναι άδεια). Η

αξία του στόχου είναι, στη συνέχεια το μήκος της συντομότερης διαδρομής, δεδομένου ότι ο στόχος είναι μηδέν σε μια παραδεκτή ευρετική.

Ο αλγόριθμος που περιγράφεται μέχρι τώρα μας δίνει μόνο το μήκος της συντομότερης διαδρομής. Για να βρεθεί η πραγματική αλληλουχία των βημάτων, ο αλγόριθμος μπορεί εύκολα να αναθεωρηθεί, έτσι ώστε κάθε κόμβος στο μονοπάτι να παρακολουθεί τον προκάτοχό του. Ο αλγόριθμος δηλαδή τρέχει ως εξής: ο τελικός κόμβος δείχνει τον προκάτοχό του, και ούτω καθεξής, μέχρις ότου ο προκάτοχος κάποιου κόμβου να ταυτιστεί με τον κόμβο εκκίνησης. Επιπλέον, εάν η ευρετική είναι μονότονη (ή σταθερή, βλέπε παρακάτω), ένα κλειστό σύνολο κόμβων μπορεί να χρησιμοποιηθεί για να κάνει την αναζήτηση πιο αποτελεσματική. Ο A* χρησιμοποιείται κατά κόρον για απλά προβλήματα εύρεσης μονοπατιών σε εφαρμογές όπως βιντεοπαιχνίδια, αλλά είχε πρωτοσχεδιαστεί για να χρησιμοποιηθεί ως αλγόριθμος γενικής αναζήτησης σε γράφο.

Βρίσκεται, επίσης, σε εφαρμογές ποικίλων προβλημάτων συμπεριλαμβανομένου του προβλήματος της ανάλυσης χρησιμοποιώντας στοχαστικές γραμματικές σε NPL

Αλγόριθμος Johnson[5](σελ.42). Ο αλγόριθμος Johnson βρίσκει τα συντομότερα μονοπάτια μεταξύ όλων των ζευγαριών των κόμβων σε ένα αραιό κατευθυνόμενο γράφο με βάρη στις ακμές του. Στον γράφο αυτό μπορούν να υπάρχουν αρνητικά βάρη αλλά όχι αρνητικοί κύκλοι. Ο αλγόριθμος δουλεύει χρησιμοποιώντας τον αλγόριθμο Bellman-Ford ώστε να υπολογίσει έναν μετασχηματισμό του γράφου ο οποίος αφαιρεί όλα τα αρνητικά βάρη επιτρέποντας να εφαρμοσθεί ο αλγόριθμος Dijkstra στον μετασχηματιζόμενο γράφο.

Αλγόριθμος Viterbi[4](σελ.409). Ο αλγόριθμος δυναμικού προγραμματισμού που βρίσκει την πιο πιθανή ακολουθία από κρυφά στάδια ονομαζόμενη ως μονοπάτι Viterby και δίνει ως αποτέλεσμα μια ακολουθία από παρατηρούμενα γεγονότα στο πλαίσιο των Μαρκοβιανών πηγών πληροφορίας και των κρυφών Μαρκοβιανών μοντέλων.

Ο αλγόριθμος εφαρμόζεται παγκοσμίως για την αποκωδικοποίηση συνελλικτικών κωδικών που χρησιμοποιούνται σε CDMA και GSM ψηφιακών κυψελών, dial-up modems, δορυφόρους, deepspace επικοινωνίες και σε ασύρματα δίκτυα.

Χρησιμοποιείται επίσης σε εφαρμογές αναγνώρισης φωνής, σύνθεση φωνής, εντοπισμού λέξης-κλειδί, υπολογιστικές γλώσσες και βιοπληροφορική.

Αλγόριθμος D'Esopo-Pape[17] .Ο αλγόριθμος D'Esopo-Pape χρησιμοποιεί τη λίστα υποψηφίων με τη μορφή μιας ουράς. Οι κορυφές που πρέπει να ελεγχθούν πάντα ανακτώνται από τη κορυφή της λίστας. Ωστόσο, ο τόπος που μια δεδομένη κορυφή προστίθεται στο κατάλογο των υποψήφιων ουσιαστικά εξαρτάται από το αν η κορυφή βρίσκεται ήδη σε αυτή τη λίστα. Αν ναι, προστίθεται στη κεφαλή της, αλλιώς - στο τέλος της λίστας.

Αλγόριθμος Small Label First algorithm (SLF)[17] . Ο αλγόριθμος (SLF) επιδιώκει να διαχειριστεί τον κατάλογο υποψηφίων με τέτοιο τρόπο ώστε οι κορυφές με μικρές ετικέτες να βρίσκονται όσο πιο κοντά στην κορυφή της όσο είναι δυνατό. Ο λόγος για αυτή τη λειτουργία είναι το γεγονός ότι όσο μικρότερη είναι η ετικέτα ενός κόμβου που ανακτάται από το κατάλογο υποψηφίων, τόσο μικρότερη είναι η πιθανότητα ότι αυτή η κορυφή θα διαβιβαστεί για άλλη μια φορά στη λίστα. Ο αλγόριθμος αυτός, προσπαθεί

να φτάσει τη χαρακτηριστική λειτουργία του αλγόριθμου του Dijkstra με χαμηλότερες υπολογιστικές δαπάνες.

Αλγόριθμος Large Label Last algorithm (LLL)[17] . Ο αλγόριθμος ((LLL)) επιχειρεί να επιτύχει λειτουργία που είναι παρόμοια με εκείνη του προηγούμενου αλγόριθμου χρησιμοποιώντας μια συγκεκριμμένη μέθοδο για την ανάκτηση των κορυφών από τη λίστα των υποψηφίων. Η προσθήκη των κορυφών στον κατάλογο υποψηφίων δεν καθορίζεται με οποιονδήποτε τρόπο, ωστόσο, η μέθοδος για την ανάκτηση τους από τον κατάλογο ορίζεται. Κάθε φορά, όταν μια κορυφή πρέπει να ληφθεί από το κατάλογο, η μέση τιμή των ετικετών των κορυφών της λίστας υπολογίζεται. Στη συνέχεια, η ετικέτα του κόμβου που βρίσκεται στην κεφαλή του καταλόγου συγκρίνεται με το μέσο όρο. Αν η ετικέτα της κορυφής είναι υψηλότερη από το μέσο όρο, η κορυφή κινείται προς το τέλος της λίστας. Διαφορετικά, η κορυφή επιστρέφεται ως εκείνη που χρήζει θεώρησης στην παρούσα επανάληψη.

Αλγόριθμος SLF/LLL. Ο SLF/LLL[17] . συνδυάζει την (SLF) μέθοδο για την προσθήκη των κορυφών στον κατάλογο υποψήφιων και τη μέθοδο ((LLL)) για την ανάκτησή τους από τον κατάλογο. Ο αλγόριθμος SLF/LLL απαιτεί χαμηλότερο αριθμό επαναλήψεων για να λύσει το πρόβλημα εύρεσης συντομότερης διαδρομής από τους αλγόριθμους που συνδυάζει. Αυτό γίνεται, όμως, με το κόστος του αυξημένου αριθμού των απαραίτητων υπολογισμών.

Αλγόριθμος Dantzig[4](σελ.897).Ένας άλλος αλγόριθμος για την εύρεση ενός συντομότερου μονοπατιού μεταξύ κάθε ζεύγους κορυφών σε ένα γράφημα είναι ο αλγόριθμος Dantzig.

Ο αλγόριθμος αυτός μοιάζει με τον αλγόριθμο Floyd-Warshall στο ότι εκτελούνται οι ίδιοι υπολογισμοί. Η σειρά, όμως, εκτέλεσης των υπολογισμών αυτών διαφέρει στους δυο αλγορίθμους. Κατ΄ ουσίαν, ενώ ο αλγόριθμος Floyd-Warshall εκκινεί απο το κόμβο αρχή προς το κόμβο στόχο εκτελώντας το πολλαπλασιασμό των πινάκων γειτνίασης απο την αρχή προς το τέλος, ο αλγόριθμος Dantzig κάνει το αντίθετο.

3.2 Πρόβλημα συντομότερου μονοπατιού όλων των ζευγαριών

Στο πρόβλημα συντομότερου μονοπατιού όλων των ζευγαριών, δίνεται ο σταθμισμένος κατευθυνόμενος γράφος χωρίς αρνητικούς κύκλους, και πρέπει να βρεθεί η συντομότερη διαδρομή μεταξύ κάθε ζεύγους κορυφών σε αυτόν. Οι κλασικοί αλγόριθμοι για τον υπολογισμό συντομότερου μονοπατιού όλων των ζευγαριών είναι οι Dijkstra και Floyd. Ο αλγόριθμος Dijkstra για την επίλυση του προβλήματος εύρεσης της συντομότερης διαδρομής μοναδικής εκκίνησης μπορεί να να εκτελεστεί μία φορά για κάθε κορυφή στο γράφημα ως κορυφή - πηγή. Εάν χρησιμοποιηθούν σωροί Fibonacci κατά την υλοποίηση των ουρών προτεραιότητας, ο αλγόριθμος αυτός έχει μειωμένο χρόνο για την επίλυση του προβλήματος όπως παρουσιάστηκε ήδη παραπάνω[10, 9].

Σε αντίθεση με τον αλγόριθμο του Dijkstra ο αλγόριθμος επίλυσης του πρόβληματος

συντομότερου μονοπατιού όλων των ζευγαριών που θεσπίστηκε από το Floyd μπορεί να επιλύσει γράφους με αρνητικά βάρη που συνδέονται με τα άκρα. Ο γράφος εντούτοις δεν πρέπει να περιλαμβάνει αρνητικούς κύκλους. Ο αλγόριθμος λειτουργεί με δυναμικό προγραμματισμό.

Αντίστοιχα το 1973 εισήχθη ένας άλλος αλγόριθμος επίλυσης του πρόβληματος συντομότερου μονοπατιού όλων των ζευγαριών που βασίζεται στον Dijkstra. Στο αλγόριθμο, αν τα βάρη που συνδέονται με ακμές είναι ανεξάρτητα και ταυτόσημα κατανεμημένα, ο αναμενόμενος χρόνος εκτέλεσης θα είναι $O(n^2(logn)$. Αντίστοιχα το 1980 εισήχθη και άλλος ένας παρόμοιος αλγόριθμος με αναμενόμενο χρόνο εκτέλεσης $O(lognhZ(1ogn))$[7].

Ο μαθηματικός ορισμός του προβλήματος είναι ο ακόλουθος[4]:

Λαμβάνοντας υπόψη ένα σταθμισμένο, κατευθυνόμενο γράφο $G(V,E)$ όπου V είναι μια ομάδα απο n κορυφές και E είναι ένα σύνολο από m ακμές. Ο γράφος εκπροσωπείται από ένα πίνακα γειτνίασης, του οποίου τα στοιχεία είναι $w[i,j]$, της ακμής$[i,j]$ για i, j =1,2, ..., n. Τα βάρη της ακμής μπορεί να είναι αρνητικά αλλά ο γράφος δεν πρέπει να περιέχει κανένα αρνητικό κύκλο. Για την αποθήκευση των βαρών των συντομότερων μονοπατιών, χρησιμοποιείται μια μήτρα απόστασης με στοιχεία $d[i,j]$, ως το βάρος του συντομότερου μονοπατιού από το j στο i, j = 1,2, ..., n. Επίσης χρησιμοποιείται ένας πίνακας προκατόχων για την κατασκευή και αποθήκευση των πιο κοντινών διαδρομών με στοιχεία $r[i, j]$, πού αντιπροσωπεύουν τον προκάτοχο του j ως προς την συντομότερη διαδρομή από το i στο j = 1,2, ..., n. Το πρόβλημα ορίζεται ως η ανάγκη για εύρεση των συντομότερων μονοπάτιων μεταξύ κάθε ζεύγους των κορυφών του γράφου.

Οι Feuerstein et al. πρότειναν έναν αλγόριθμο για την επίλυση του πρόβληματος εύρεσης του συντομότερου μονοπατιού όλων των ζευγαριών ενός γράφου. Ο αλγόριθμος χρησιμοποιούσε το θεώρημα διαχωρισμού και διαιρούταν σε τρεις περιπτώσεις:

- Στη πρώτη περίπτωση ο αριθμός των επαναλλήψεων για την εύρεση του συντομότερου μονοπατιού είναι μικρός σε σχέση με τον αριθμό των κορυφών του γράφου.
- Στην δευτερη και τρίτη περίπτωση το αντίθετο.

Ο λόγος πίσω από αυτή τη διάκριση είναι ότι όταν ο αριθμός των ερωτημάτων της επίλυσης είναι μικρός σε σύγκριση με τον αριθμό των κορυφών γραφήματος, η προεπεξεργασία και αλγόριθμος εύρεσης της συντομότερης διαδρομής μπορεί να χρησιμοποιηθεί. Αυτός λαμβάνει $O(nlogn)$ χρόνο προεπεξεργασίας, η οποία όμως πρέπει να λάβει χώρα μόνο μια φορά ανά ελάχιστη διαδρομή. Στη δεύτερη και τρίτη περίπτωση, η συντομότερη διαδρομή για όλα τα ζεύγη θα πρέπει να υπολογιστεί. Με q να αντιπροσωπεύει τον αριθμό των ερωτημάτων συντομότερης διαδρομής και v να παριστά τον αριθμό των κορυφών του γραφήματος.

3.3 Άλλες κατηγορίες προβλημάτων διερεύνησης γράφων

Το πρόβλημα του δέντρου ελάχιστης κάλυψης[5](σελ.17)[8](σελ.43). Εδώ δεν ψάχνουμε τη συντομότερη διαδρομή μεταξύ δύο κόμβων ενός δικτύου, αλλά ένα σύνολο κλάδων του δικτύου στο οποίο υπάρχει ένα μονοπάτι για κάθε ζεύγος κόμβων του δικτύου, ενώ το συνολικό μήκος των κλάδων αυτών είναι το ελάχιστο δυνατό. Για να γίνει αυτό, οι κλάδοι θα πρέπει να επιλεγούν με τέτοιο τρόπο που να σχηματίζεται ένα δένδρο (θυμηθείτε ότι δέντρο είναι ένα συνεκτικό δίκτυο χωρίς κύκλους) που συνδέει όλες τις κορυφές. Με άλλα λόγια, το πρόβλημα είναι να βρεθεί το συνεκτικό δένδρο με το ελάχιστο συνολικό μήκος. Το πρόβλημα αυτό έχει ορισμένες πολύ σπουδαίες πρακτικές εφαρμογές. Για παράδειγμα, είναι πολύ χρήσιμο στον προγραμματισμό δικτύων επικοινωνίας, όπου πρέπει να βρεθεί κάποιο δέντρο, που να συνδέει όλους τους κόμβους μεταξύ τους με κάποιο μονοπάτι με τον πιο οικονομικό τρόπο. Οι κόμβοι μπορεί να είναι τηλεπικοινωνιακοί σταθμοί, οι κλάδοι καλώδια μεταφοράς του σήματος και οι αποστάσεις το κόστος κατασκευής του αντίστοιχου κλάδου. Στα πλαίσια αυτά, το πρόβλημα του δέντρου ελάχιστης κάλυψης είναι ο προσδιορισμός εκείνων των κλάδων, που εξυπηρετούν όλους τους σταθμούς με το ελάχιστο συνολικό κόστος. Άλλα παραδείγματα είναι ο προγραμματισμός μεγάλων δικτύων μεταφοράς. Το πρόβλημα του δέντρου ελάχιστης κάλυψης μπορεί να λυθεί απ΄ ευθείας, όπου αρχίζοντας από οποιοδήποτε κόμβο θα οδηγηθούμε στο τέλος στη βέλτιστη λύση.

Το πρόβλημα της μέγιστης ροής[8](σελ.43). Το πρόβλημα της μέγιστης ροής μπορεί να περιγραφεί ως εξής. Δίνεται ένα συνεκτικό προσανατολισμένο δίκτυο που έχει μια πηγή και ένα δέκτη. Η ροή σε αυτό το δίκτυο ξεκινά από την πηγή και καταλήγει στο δέκτη. Όλοι οι υπόλοιποι κόμβοι είναι κόμβοι ενδιάμεσης μεταφοράς. Η ροή σε έναν κλάδο επιτρέπεται μόνο προς την κατεύθυνση που δείχνουν τα βέλη. Υποθέτουμε ότι η ροή κατά μήκος του κλάδου (i, j) από τον κόμβο i στον κόμβο j μπορεί να είναι οποιαδήποτε μη αρνητική ποσότητα, που δεν είναι μεγαλύτερη από την προκαθορισμένη δυναμικότητα ροής u. Στην πηγή όλοι οι κλάδοι απομακρύνονται από τον κόμβο αυτό. Στο δέκτη όλοι οι κλάδοι καταλήγουν προς αυτό τον κόμβο. Υποθέτουμε διατήρηση της ροής σε κάθε κόμβο (δηλαδή η ροή προς κάθε κόμβο είναι ίση με τη ροή από κάθε κόμβο), εκτός από την πηγή και το δέκτη. Αντικειμενικός σκοπός είναι να προσδιοριστεί το εφικτό σχέδιο ροής διαμέσου του δικτύου, που μεγιστοποιεί τη συνολική ροή από την πηγή στο δέκτη.

Το πρόβλημα της ροής ελάχιστου κόστους[8](σελ.43). Το πρόβλημα της ροής ελάχιστου κόστους κατέχει μία πολύ σημαντική θέση μεταξύ των μοντέλων βελτιστοποίησης δικτύων, τόσο επειδή περιλαμβάνει μία πολύ ευρεία ομάδα εφαρμογών αλλά και επειδή μπορεί να επιλυθεί πολύ αποτελεσματικά. Όπως και το πρόβλημα της μέγιστης ροής, εξετάζει ροή μέσω ενός δικτύου με περιορισμένες χωρητικότητες κλάδων. Όπως το πρόβλημα της συντομότερης διαδρομής, θεωρεί ένα κόστος (ή απόσταση) για ροή μέσω ενός κλάδου. Όπως το πρόβλημα μεταφοράς ή το πρόβλημα ανάθεσης, μπορούν να συμπεριληφθούν στο πρόβλημα πολλές πηγές και πολλοί δέκτες για την ροή, με αντίστοιχα

κόστη. Μάλιστα, όλα τα παραπάνω προβλήματα είναι ειδικές περιπτώσεις του προβλήματος ροής ελάχιστου κόστους

Προβλήματα εύρεσης βέλτιστης λύσης - Το πρόβλημα του πλανώδιου πωλητή[8](σελ.42)[5](σελ.14). Το πρόβλημα του πλανόδιου πωλητή περιλαμβάνει την εύρεση της (σχεδόν) μικρότερης διαδρομής που ενώνει ένα αριθμό περιοχών -πιθανόν εκατοντάδων- όπως δηλαδή κάνει ένας περιπλανώμενος πωλητής που επισκέπτεται διάφορες πόλεις με σκοπό να πουλήσει τα προϊόντα του. Αν ένας πωλητής, ξεκινάει από την πόλη του, και θέλει να επισκεφτεί ακριβώς μία φορά κάθε πόλη από μια δοθείσα λίστα πόλεων και να επιστρέψει πάλι στην πόλη του, είναι εύλογο για αυτόν να επιλέξει την σειρά με την οποία θα επισκεφτεί τις πόλεις έτσι ώστε η συνολική απόσταση την οποία θα διανύσει κατά την περιοδεία του να είναι όσο το δυνατόν μικρότερη. Έστω ότι γνωρίζει, για κάθε ζεύγος πόλεων, την απόσταση από την μια πόλη στην άλλη, τότε έχει όλα τα δεδομένα που του χρειάζονται για να βρει την ελάχιστη διαδρομή, αλλά δεν είναι καθόλου προφανές γι' αυτόν πώς να χρησιμοποιήσει τα δεδομένα αυτά προκειμένου να πάρει την απάντηση που ζητά. Το πρόβλημα μπορεί να μοντελοποιηθεί ως ένα μη-κατευθυνόμενο γράφο, έτσι ώστε οι πόλεις να αναπαριστάνονται με κόμβους του γράφου, τα μονοπάτια που συνδέουν τις πόλεις μεταξύ τους να είναι οι ακμές του γραφήματος και μια απόσταση μονοπατιού να είναι το μήκος της κάθε ακμής. Είναι ένα πρόβλημα ελαχιστοποίησης και εύρεσης του μικρότερου μονοπατιού, ξεκινώντας και τελειώνοντας σε ένα συγκεκριμένο κόμβο αφού έχει επισκεφθεί κάθε άλλο κόμβο ακριβώς μια φορά. Ως συνήθως το μοντέλο παριστάνεται από ένα πλήρες γράφο. Αν δεν υπάρχει διαδρομή μεταξύ δύο κόμβων-πόλεων, προσθέτουμε αυθαίρετα μια «μεγάλη» ακμή (το μήκος της να είναι πολύ μεγαλύτερο από όλες τις άλλες ακμές), για να ολοκληρωθεί το διάγραμμα χωρίς να επηρεαστεί η βέλτιστη περιήγηση.

Κεφάλαιο 4

Ο αλγόριθμος Floyd -Warshall

Στην ενότητα αυτή θα παρουσιαστεί η βασική θεωρία πίσω απο τον αλγόριθμο Floyd-Warshall, η λειτουργία του καθώς και ορισμένα παραδείγματα χρήσης του απο τη βιβλιογραφία.

Θα χρησιμοποιήσουμε μια διαφορετική διατύπωση του δυναμικού προγραμματισμού για την επίλυση του συγκεκριμμένου προβλήματος σε ένα κατευθυνόμενο γράφημα $G = (V, E)$.

Ο προκύπτων αλγόριθμος, γνωστός ως ο αλγόριθμος Floyd-Warshall, τρέχει σε $O(V^3)$ χρόνο. Όπως και πριν, αρνητικά βάρη ακμών μπορεί να υπάρχουν, αλλά υποθέτουμε ότι δεν υπάρχουν κύκλοι αρνητικού βάρους[4](σελ.645,693-699)[10](σελ.14)[5](σελ.41)[12, 13].

Στον αλγόριθμο Floyd-Warshall, [15]χρησιμοποιούμε ένα διαφορετικό χαρακτηρισμό της δομής της συντομότερης διαδρομής από ό, τι χρησιμοποιείται στους αλγορίθμους για το πρόβλημα όλων των ζευγών που βασίζονται στο πολλαπλασιαμό πινάκων.

Ο αλγόριθμος θεωρεί ¨ενδιάμεσες' κορυφές της συντομότερης διαδρομής, όπου μια ενδιάμεση κορυφή ενός απλού μονοπατιού $p = v_1, v_2, ..., v_i$ είναι οποιαδήποτε κορυφή του p εκτός των v_1, v_i, δηλαδή, κάθε κορυφή στο σύνολο $v_2, v_3, ..., v_{i-1}$.

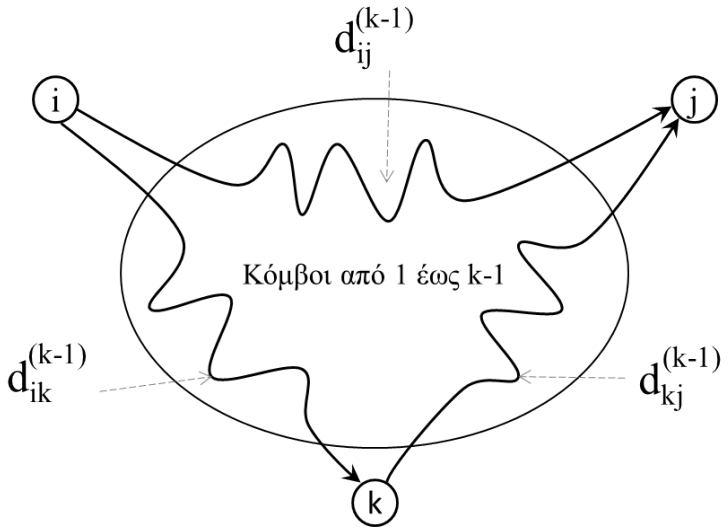
Ο αλγόριθμος Floyd-Warshall βασίζεται στην ακόλουθη παρατήρηση. Υπό την υπόθεση ότι οι κορυφές του G είναι $V = 1, 2, ..., n$, εξετάζεται ένα υποσύνολο $1, 2, ..., k$ των κορυφών για κάποιο k. Για κάθε ζεύγος κορυφών i, j, V, να εξετάζονται όλες οι διαδρομές από το i στο j των οποίων οι ενδιάμεσες κορυφές προέρχονται από το $1, 2, ..., k$ και έστω p ένα μονοπάτι ελάχιστου βάρους μεταξύ τους.

Ο αλγόριθμος Floyd-Warshall εκμεταλλεύεται μια σχέση μεταξύ του μονοπατιού p και των συντομότερων διαδρομών από το i στο j με όλες τις ενδιάμεσες κορυφές στο σύνολο $1, 2, ..., k-1$. Η σχέση εξαρτάται από το αν ή όχι το k είναι ενδιάμεση κορυφή του μονοπατιού p.

- Εάν το k δεν είναι μια ενδιάμεση κορυφή στο μονοπάτι p, τότε όλες οι ενδιάμεσες κορυφές του μονοπατιού p είναι στο σύνολο $1, 2, ..., k-1$. Έτσι, μια συντομότερη διαδρομή από την κορυφή i προς την κορυφή j με όλες τις ενδιάμεσες κορυφές στο σύνολο $1, 2, ..., k-1$ είναι επίσης μια συντομότερη διαδρομή από το i στο j με όλες τις ενδιάμεσες κορυφές του συνόλου $1, 2, ..., k$.

- Αν το k είναι μια ενδιάμεση κορυφή του μονοπάτιού p, τότε μπορούμε να σπάσουμε

το p, όπως φαίνεται στο σχήμα. Το p_1 είναι μια συντομότερη διαδρομή από το i στο k με όλες τις ενδιάμεσες κορυφές του συνόλου $1, 2, ..., k$. Επειδή η κορυφή k δεν είναι μια ενδιάμεση κορυφή της διαδρομής p_1 βλέπουμε ότι το p_1 είναι η συντομότερη διαδρομή από το i στο k με όλες τις ενδιάμεσες κορυφές στο σύνολο $1, 2, ..., k-1$. Ομοίως, το p_2 είναι μια συντομότερη διαδρομή από την κορυφή k προς κορυφή j με όλες τις ενδιάμεσες κορυφές στο σύνολο $1, 2, ..., k-1$[5](σελ.44).



Σχήμα 4.1: Γραφική απεικόνιση λογικής ενδιαμεσων κορυφών του αλγορίθμου

Με βάση τις παραπάνω παρατηρήσεις , ορίζουμε μια αναδρομική σύνθεση των εκτιμήσεων της συντομότερης διαδρομής. Έστω $d^k_{i,j}$ το βάρος ενός συντομότερου μονοπατιού από τη κορυφή i προς την κορυφή j για τις οποίες όλες οι ενδιάμεσες κορυφές βρίσκονται στο σύνολο $1, 2, ..., k$. Όταν $k = 0$, ένα μονοπάτι από την κορυφή i προς την κορυφή j χωρίς ενδιάμεση κορυφή αριθμημένη υψηλότερα από το 0 δεν έχει καμία ενδιάμεση κορυφή. Ένα τέτοιο μονοπάτι έχει το πολύ ένα άκρο, και ως εκ τούτου $d^{(0)}_{(i,j)} = w_{(i,j)}$. Ένας αναδρομικός ορισμός συνεπώς είναι:

$$d(i,j)^k = \left\{ \begin{array}{cc} w(i,j) & \text{if } k = 0 \\ \min((d(i,j)^{k-1}, d(i,k)^{k-1} + d(k,j)^{k-1}) & \text{if } k \geq 1 \end{array} \right\} \quad (4.1)$$

Ο αλγόριθμος Floyd-Warshall μπορεί να χρησιμοποιηθεί για να λύσει τα ακόλουθα προβλήματα, μεταξύ των άλλων:

Κοντύτερες πορείες στις κατευθυνόμενες γραφικές παραστάσεις. Για αυτό στην εργασία, στα βάρη όλων των ακρών ανατίθεται ο ίδιος θετικός αριθμός. Για τον αριθμό αυτό επιλέγεται συνήθως το ένα, έτσι ώστε το βάρος μιας πορείας να συμπίπτει με τον αριθμό της εκάστοτε άκρης κατά μήκος της πορείας[4].

Μεταβατική περάτωση από τις κατευθυνόμενες γραφικές παραστάσεις[14, 4]. Στην αρχική διατύπωση του αλγορίθμου Floyd-Warshall, η γραφική παράσταση δεν

έχει βάρη και αντιπροσωπεύεται από μια μήτρα γειτνίασης του Boole. Κατόπιν η λειτουργία προσθεσης αντικαθίσταται με τη λογική πύλη AND και η λειτουργία αφαίρεσης με τη λογική πύλη συνεπαγωγής.

Βέλτιστη δρομολόγηση[4] . Σε αυτήν την εφαρμογή ο σκοπός είναι η εύρεση της πορείας με τη μέγιστη ροή μεταξύ δύο κορυφών. Αυτό σημαίνει ότι, αντί να εξετάζονται τα ελάχιστα, εξετάζονται τα μέγιστα. Τα βάρη ακρών αντιπροσωπεύουν τους σταθερούς περιορισμούς στη ροή. Τα βάρη πορειών αντιπροσωπεύουν τις δυσχέρειες έτσι η προαναφερθείσα λειτουργία προσθηκών αντικαθίσταται από την ελάχιστη λειτουργία.

Αντιστροφή πραγματικών μήτρων[5](σελ.44)[4]. Ο αλγόριθμος λειτουργεί στις μήτρες και μπορεί να χρησιμοποιηθεί για να αναστρέψει μια πραγματική μήτρα. Αυτό ολοκληρώνεται με το τρέξιμο του αλγορίθμου και αντικαθιστώντας την προσθήκη από τον πολλαπλασιασμό και τα ελάχιστα από την προσθήκη.

Ο ψευδοκώδικας Floyd-Warshall:[10](σελ.16)

```
let dist be a |V|x|V| array of minimum distances initialized to infinity
for each vertex v
    dist[v][v] <- 0
for each edge (u,v)
     dist[u][v] <- w(u,v) // the weight of the edge (u,v)
for k from 1 to |V|
     for i from 1 to |V|
        for j from 1 to |V|
            if dist[i][k] + dist[k][j] < dist[i][j] then
            dist[i][j] <- dist[i][k] + dist[k][j]

```

Listing 4.1: FloydWarshall Pseudocode

4.1 Γράφοι με αρνητικούς κύκλους

Ο αλγόριθμος Floyd-Warshall[14, 13] είναι ένας απλός και ευρέως χρησιμοποιούμενος αλγόριθμος για τον υπολογισμό συντομότερων μονοπατιών μεταξύ όλων των ζευγών των κορυφών σε ένα γράφο. Αυτός ο αλγόριθμος έχει χειρότερη περίπτωση χρόνου εκτέλεσης τον $O(V^3)$ για γραφήματα με V κορυφές. Υπάρχουν διάφοροι αλγόριθμοι με καλύτερο χρονικό διάστημα χειρότερης περίπτωσης εκτέλεσης, με τον καλύτερο από αυτούς τους αλγορίθμους σήμερα να έχει μια επίτευξη εκτέλεσης της τάξης του $O(V^3 log(logV)/log^2 V)$ και αντίστοιχα $O(mn + log(logV))$.

Ωστόσο, αυτοί οι αλγόριθμοι είναι πολύ πιο περίπλοκοι από ό,τι ο αλγόριθμος Floyd-Warshall και περιλαμβάνουν πολύπλοκες δομές δεδομένων. Ως εκ τούτου, σε πολλές περιπτώσεις, ο αλγόριθμος Floyd-Warshall εξακολουθεί να είναι η καλύτερη επιλογή.

Ο αλγόριθμος Floyd-Warshall εξάγει το σωστό αποτέλεσμα εφ ́όσον δεν υπάρχουν αρνητικοί κύκλοι στο γράφημα εισόδου. Σε περίπτωση που υπάρχει ένα αρνητικός κύκλος, υπολογίζοντας μια συντομότερη (απλή) διαδρομή είναι ένα NP-hard πρόβλημα και ο αλγόριθμος Floyd-Warshall δεν θα εμφανίσει το σωστό αποτέλεσμα[13](σελ.2).

Σε αυτή τη περίπτωση ο αλγόριθμος Floyd-Warshall μάλλον, θα ανιχνεύσει την παρουσία ενός αρνητικού κύκλου ελέγχοντας ότι υπάρχει μια αρνητική εγγραφή στη διαγώνιο της μήτρας γειτνίασης.

Σε αυτή την περίπτωση, μπορεί να εφαρμοστεί ο παρακάτω ψευδοκώδικας για την ανίχνευση των αρνητικών κύκλων[10](σελ.10):

Είσοδος: Ένας κατευθυνόμενος γράφος G με $V(G) = 1, ..., n$ και βάρη $c : E(G) \to R$

Έξοδος: Ένας M πίνακας $n \times n$ έτσι ώστε κάθε στοιχείο $M[i, j]$ να περιέχει το μήκος του μικρότερου μονοπατιου από τον εκάστοτε κόμβο i σε κόμβο j

```
1     M[i, j] <- INF for every  i != j
2     M[i, i] <- 0 for every i
3     M[i, j] <- c((i, j)) for every (i, j) element of E(G)
4      for i <- 1 to n do
5          for j <- 1 to n do
6              for k <- 1 to n do
7                  if M[j, k] > M[j, i] + M[i, k] then M[j, k]<- M[j, i] + M
      [i, k]
8      for i <- 1 to n do
9          if M[i, i] < 0 then return(graph contains a negative cycle)
10
```

Listing 4.2: FloydWarshall Pseudocode Negative cycles

Κεφάλαιο 5

Υλοποίηση Αλγορίθμου

Στο προηγούμενο κεφάλαιο παρουσιάστηκε ο ψευδοκώδικας του αλγορίθμου όπως αναφέρεται στην βιβλιογραφία καθώς και ο τρόπος λειτουργίας αυτού. Με βάση την θεωρία που παρουσιάστηκε στα προηγούμενα κεφάλαια στο παρόν κεφάλαιο θα παρουσιαστεί η υλοποίηση του αλγορίθμου σε υπολογιστικό περιβάλλον καθώς και η χρήση αυτής για την επίλυση ενδεικτικών προβλημάτων εύρεσης της ελάχιστης διαδρομής. Ο αλγόριθμός Floyd-Warshall υλοποιήθηκε σε γλώσσα Java.Στις παρακάτω ενότητες, περιγράφεται η υλοποίηση του.

Η εκτέλεση του αλγόριθμου γίνεται με την κλάση FloydWarshall.java και εκτελούνται τα αποτελέσματα του πίνακα γειτνίασης μέσω της κλάσηςMatrix.java, ενώ η εκτύπωση της λίστας γειτνίασης γίνεται μέσω της κλάσης List.java. Οι δύο τελευταίες κλάσεις με την σειρά τους κληρονομούν τα στοιχεία της κλάσης Adjacency.java. Με βάση αυτά, ο κώδικας υλοποίησης του αλγορίθμου είναι ο ακόλουθος:

5.1 Κώδικας κλάσης FloydWarshall.java

Σχολιάζοντας την λειτουργία του αλγορίθμου διαπιστώνουμε ότι προσθέτει όλες τις κορυφές μία πρός μία στο σύνολο k ενδιάμεσων κόμβων. Επιλέγει κάθε πηγή i και κάθε προορισμό j ξεχωριστά. Εάν οι κορυφές k είναι το μικρότερο μονοπάτι από το i στο j τότε εισάγονται οι καινούριες τιμές στο dist .

```
1  package packageProject;
2
3  public class FloydWarshall {
4
5    private int vertex;
6    private int dist[][];
7
8    public FloydWarshall(int v,int dis[][]){
9
10     vertex = v;
11     dist = dis;
12     Algorithm();
13   }
```

```
14
15
16   public void Algorithm(){
17
18    // adds all vertices one by one to the k
19    // set of intermediate vertices.
20     for (int k = 0; k < vertex; k++){
21
22       //  select all vertices one by one as a source
23       for (int i = 0; i < vertex; i++){
24
25
26       //  select all vertices one by one as a destination
27         for (int j = 0; j < vertex; j++){
28
29
30       // if vertex k is in the smallest path
31       // from i to j update value dist[]
32           if ( dist[i][k] + dist[k][j] < dist[i][j])
33           dist[i][j] = dist[i][k] + dist[k][j];
34         }
35       }
36     }
37     setDist(dist);
38   }
39
40   public void setDist(int[][] dis){
41     dist = dis;
42   }
43
44
45   public int[][] getDist(){
46     return dist;
47   }
48 }
```

Listing 5.1: FloydWarshall.java

5.2 Κώδικας κλάσης Adjacency.java

Η κλάση Adjacency είναι αυτή η οποία ορίζει μία αφηρημένη κλάση τύπο abstract με όνομα μεθόδου display και στοιχεία εισόδου το πλήθος την κορυφών, τα στοιχεία του πίνακα και την άπειρη τιμή του infinity.

```
1  package packageProject;
2
3  public abstract class Adjacency {
4
5    public Adjacency() {}
6
7    public abstract void display(int vertex, int[][] dist, int inf);}
```

Listing 5.2: Adjacency.javay

5.3 Κώδικας κλάσης Matrix.java

Η αναπαράσταση της κλάση Matrix μας εμφανίζει τις τιμές κόστους του dist με την μορφή λεπτομεριακής ανπαράστασης επάνω σε πίνακα.

```
1  package packageProject;
2  public class Matrix extends Adjacency{
3
4    public Matrix(){}
5
6    @Override
7    public void display(int vertex, int[][] dist, int inf) {
8
9      System.out.print("\t    ");
10     for(int i=0;i<vertex;i++){
11       System.out.print((i+1)+"-");
12     }System.out.println();
13
14     for(int i=0;i<vertex;i++){
15       System.out.print("Vertex "+ (i+1) +" | ");
16       for(int j=0;j<vertex;j++){
17
18         if(dist[i][j] != inf /*&& i != j*/)
19           System.out.print(dist[i][j] + " ");
20         else
21           System.out.print("INF ");
22
23         if(j==vertex-1)
24           System.out.println();
25       }
26     }
27   }
28 }
```

Listing 5.3: Matrix.javax

5.4 Κώδικας κλάσης List.java

Η κλάση List δημιουργεί ακριβή αναπαράσταση της σύνδεσης των κόμβων με τις γειτονικές συνδέσεις τους και τον αριθμό του εκάστοτε βάρους τους.

```
1  package packageProject;
2
3  public class List extends Adjacency{
4
5    public List() {}
6
7    @Override
8    public void display(int vertex, int[][] dist, int inf) {
9
10     for(int i=0;i<vertex;i++){
11       System.out.print("Vertex: "+ (i+1) +"   ");
12       for(int j=0;j<vertex;j++){
13
14         if(dist[i][j] != inf && i != j)
15           System.out.print("-> "+(j+1)+"|"+dist[i][j]+"  ");
16
17         if(j==vertex-1)
18           System.out.println();
19       }
20     }
21   }
22 }
```

Listing 5.4: List.java

5.5 Κώδικας main.java

Εκτυπώνονται οι τιμές του εκάστοτε γράφου με την μοφή λίστας γειτνίασης και πίνακα γειτνίασης. Εισάγονται οι τιμές του γράφου στον αλγόριθμο Floyd Warshall και ξανά εκτελώντας την λίστα και τον πίνακα γειτνίασης πέρνουμε τις τελικές τιμές συντομότερης διαδρομής του γράφου μας.

```
1  package packageProject;
2
3  import java.util.Scanner;
4
5  public class Main {
6
7    public static Scanner input = new Scanner(System.in);
8    public static Adjacency aj;
9    public static FloydWarshall fw;
10
11   public static void main(String[] args) {
12
13     int INF = 99999;
```

```
14      int vertex = 0;
15      int[][] dist = null;
16
17      System.out.print("Chose what problem will follow: ");
18      int number = input.nextInt();
19
20
21      switch(number){
22        case 1:
23          vertex = 5;
24          dist = new int[][] {{0 , 3, INF, 1, INF},{3, 0, 2, INF, 2},
25            {INF, 2, 0, 7, INF},{1, INF, 7, 0, 5},{INF, 2, INF, 5, 0}};
26          break;
27        case 2:
28          vertex = 7;
29          dist = new int[][] {{0, INF , 11 , INF , INF , INF , INF},
30            {5, 0 , INF , 3 , 9 , INF ,INF},
31            {INF , INF , 0 , 4 , INF , INF , INF},
32            {INF , INF , INF, 0 , 8 , INF , INF},
33            {INF , INF , INF , INF , 0 , INF , 2},
34            {INF , INF , INF , 6 , 12 , 0 , INF},
35            {7 , INF, 10 , INF , INF , INF , 0}};
36          break;
37      }
38
39      System.out.println("\nGraph:");
40      System.out.println("*Adjacency List:");
41      aj = new List();
42      aj.display(vertex,dist,INF);
43
44
45      System.out.println("\n*Adjacency Matrix\n");
46      aj = new Matrix();
47      aj.display(vertex,dist,INF);
48
49
50      System.out.println("\n\nResult FloydWarshall:");
51      System.out.println("*Cost table:\n");
52
53      fw = new FloydWarshall(vertex,dist);
54
55      aj = new Matrix();
56      aj.display(vertex,dist,INF);
57    }
58  }
```

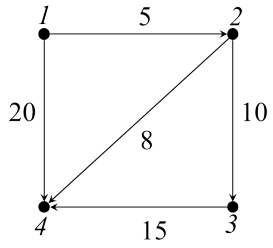
Listing 5.5: List.java

Κεφάλαιο 6

Προβλημάτα και Αποτελέσματα

6.1 Εισαγωγή προβλήματος

Η εισαγωγή του προβλήματος γίνεται με την εισαγωγή του πίνακα γειτνίασης του θεωρούμενου γράφου, με τα βάρη των ακμών σύνδεσης για τις υπάρχουσες ακμές, και τη τιμή του απείρου στην περίπτωση που η ακμή δεν υφίσταται. Για παράδειγμα, αν υποτεθεί ο ακόλουθος γράφος:

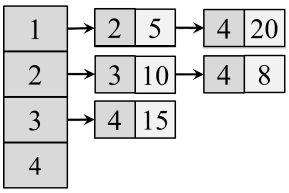


Σχήμα 6.1: Θεωρούμενος γράφος 1

Ο γράφος έχει αρχικό πίνακα:

$$\begin{Bmatrix} 0 & 5 & \text{INF} & 20 \\ \text{INF} & 0 & 10 & 8 \\ \text{INF} & \text{INF} & 0 & 15 \\ \text{INF} & \text{INF} & \text{INF} & 0 \end{Bmatrix}$$

Με λίστα:



Σχήμα 6.2: Λίστα γειτνίασης γράφου

Στα παρακάτω, παρουσιάζεται ο τρόπος εισαγωγής του προβλήματος προς επίλυση. Υπενθυμίζεται οτι σε περίπτωση αλλαγών στον αριθμό των κορυφών θα πρέπει ο χρήστης να αλλάξει και την αντίστοιχη τιμή στον βασικό αλγόριθμο υλοποίησης του Floyd-Warshall.

Εισάγονται τα παρακάτω:

```
int vertex = 4;
int dist[][] = {
  {0,5,INF,20}, {INF,0,10,8},
  {INF,INF,0,15},{INF,INF,INF,0}
};
```

Listing 6.1: main

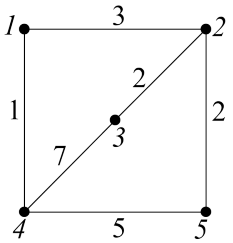
Ο αλγόριθμος σε αυτή τη περίπτωση επιστρέφει σαν αποτέλεσμα:

$$\begin{Bmatrix} 0 & 5 & 15 & 13 \\ INF & 0 & 10 & 8 \\ INF & INF & 0 & 15 \\ INF & INF & INF & 0 \end{Bmatrix}$$

6.2 Ενδεικτικά προβλήματα

Με στόχο την μελέτη του αλγορίθμου, καθώς και την διαπίστωση της σωστής λειτουργίας του κώδικα που παρήχθη για τις ανάγκες της εργασίας, μια σειρά απο προβλήματα (ενδεικτικά) θεωρήθηκαν προς επίλυση. Τα προβλήματα αφορούν σε θεωρούμενους γράφους με διαφορετικό αριθμό κορυφών, με στόχο να εξεταστεί η λειτουργία του αλγορίθμου κάτω απο διαφορετικές συνθήκες πολυπλοκότητας. Τέλος, θεωρείται και ένα πραγματικό πρόβλημα, με στόχο την τελική επιβεβαίωση της σωστής λειτουργίας του αλγορίθμου.

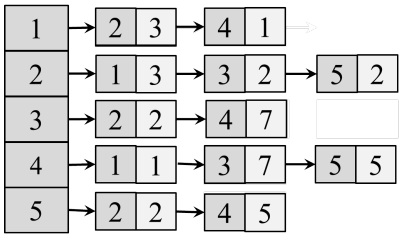
6.2.1 Πρόβλημα 1 - Μη Κατευθυνόμενος Γράφος



Σχήμα 6.3: Θεωρούμενος γράφος 2

$$\begin{Bmatrix} 0 & 3 & INF & 1 & INF \\ 3 & 0 & 2 & INF & 2 \\ INF & 2 & 0 & 7 & INF \\ 1 & INF & 7 & 0 & 5 \\ INF & 2 & INF & 5 & 0 \end{Bmatrix}$$

Η λίστα γειτνίασης είναι:

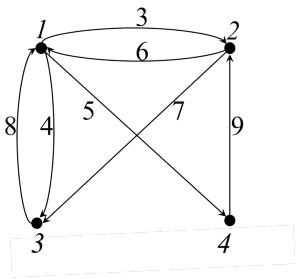


Σχήμα 6.4: Λίστα γειτνίασης γράφου

$$\left\{\begin{array}{ccccc} 0 & 3 & 5 & 1 & 5 \\ 3 & 0 & 2 & 4 & 2 \\ 5 & 2 & 0 & 6 & 4 \\ 1 & 4 & 6 & 0 & 5 \\ 5 & 2 & 4 & 5 & 0 \end{array}\right\}$$

6.2.2 Πρόβλημα 2 - Κατευθυνόμενος Γράφος(1)

Θεωρείται ο ακόλουθος κατευθυνόμενος γράφος:



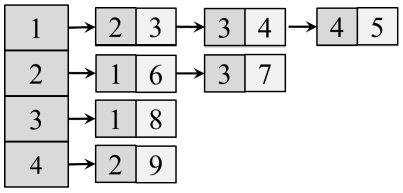
Σχήμα 6.5: Θεωρούμενος γράφος 3

Ο πίνακας γειτνίασης του γράφου είναι:

$$\left\{\begin{array}{cccc} 0 & 3 & 4 & 5 \\ 6 & 0 & 7 & \text{INF} \\ 8 & \text{INF} & 0 & \text{INF} \\ \text{INF} & 9 & \text{INF} & 0 \end{array}\right\}$$

Η λίστα γειτνίασης είναι:

Μετά την εφαρμογή του κώδικα της εργασίας επιστρέφεται σαν λύση η ακόλουθη:



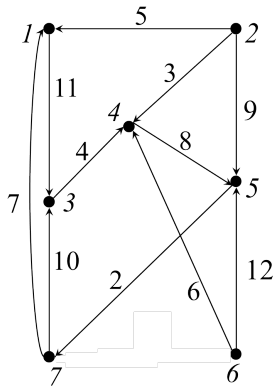
Σχήμα 6.6: Λίστα γειτνίασης γράφου

$$\begin{Bmatrix} 0 & 3 & 4 & 5 \\ 6 & 0 & 7 & 11 \\ 8 & 11 & 0 & 13 \\ 15 & 9 & 16 & 0 \end{Bmatrix}$$

Είναι εμφανές οτι ο αλγόριθμος άλλαξε την τιμή απο την υπάρχουσα στην ακμή 1-4 ενώ δημιούργησε ακμή μεταξύ των κόμβων 2-4, 3-2, 3-4, 4-1, 4-3 όπου προηγουμένως δεν υπήρχε, με αντίστοιχα βάρη 11, 11, 13, 15 16. (Η σείρά αναφοράς των ακμών αποτελεί και την κατέυθυνση του γράφου).

6.2.3 Πρόβλημα 3 - Κατευθυνόμενος Γράφος(2)

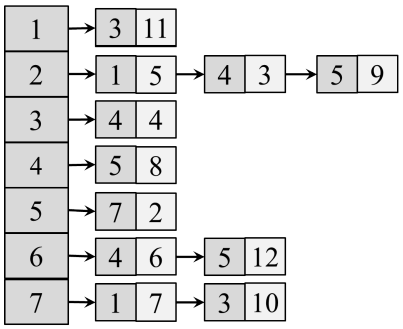
Ο γράφος που θεωρείται είναι ο ακόλουθος:



Σχήμα 6.7: Θεωρούμενος γράφος 4

Με πίνακα:

$$\begin{Bmatrix} 0 & INF & 11 & INF & INF & INF & INF \\ 5 & 0 & INF & 3 & 9 & INF & INF \\ INF & INF & 0 & 4 & INF & INF & INF \\ INF & INF & INF & 0 & 8 & INF & INF \\ INF & INF & INF & INF & 0 & INF & 2 \\ INF & INF & INF & 6 & 12 & 0 & INF \\ 7 & INF & 10 & INF & INF & INF & 0 \end{Bmatrix}$$



Σχήμα 6.8: Λίστα γειτνίασης γράφου

Η λίστα γειτνίασης είναι:

Ο κώδικας επιστρέφει σαν λύση:

$$\begin{Bmatrix} 0 & INF & 11 & 15 & 23 & INF & 25 \\ 5 & 0 & 16 & 3 & 9 & INF & 11 \\ 21 & INF & 0 & 4 & 12 & INF & 14 \\ 17 & INF & 20 & 0 & 8 & INF & 10 \\ 9 & INF & 12 & 16 & 0 & INF & 2 \\ 21 & INF & 24 & 6 & 12 & 0 & 14 \\ 7 & INF & 10 & 14 & 22 & INF & 0 \end{Bmatrix}$$

Σε αυτή την περίπτωση δημιουργήθηκαν οι ακόλουθες ακμές:

1-4 με βάρος 15, 1-6 με βάρος 23, 1-7 με βάρος 25,
2-3 με βάρος 16, 2-7 με βάρος 11, 3-1 με βάρος 21,
3-5 με βάρος 12, 3-7 με βάρος 14, 4-1 με βάρος 17,
4-3 με βάρος 20, 4-7 με βάρος 10, 5-1 με βάρος 9,
5-3 με βάρος 12, 5-4 με βάρος 16, 6-1 με βάρος 21,
6-3 με βάρος 24, 6-7 με βάρος 14, 7-4 με βάρος 14
7-5 με βάρος 22

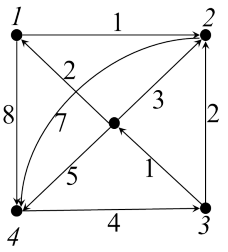
Επίσης δεν παρατηρείται καμία αλλαγή στα υπάρχοντα βάρη ακμών απο τον αρχικό πίνακα γειτνίασης.

6.2.4 Πρόβλημα 4 - Κατευθυνόμενος Γράφος(3)

Θεωρείται ο ακόλουθος γράφος:

Με πίνακα γειτνίασης:

$$\begin{Bmatrix} 0 & 1 & INF & 8 & INF \\ INF & 0 & 2 & 7 & INF \\ INF & INF & 0 & INF & 1 \\ INF & INF & 4 & 0 & INF \\ 2 & 3 & INF & 5 & 0 \end{Bmatrix}$$



Σχήμα 6.9: Θεωρούμενος γράφος 5

Μετά την εφαρμογή, ο κώδικας επιστρέφει σαν λύση τον ακόλουθο πίνακα:

$$\begin{Bmatrix} 0 & 1 & 3 & 8 & 4 \\ 5 & 0 & 2 & 7 & 3 \\ 3 & 4 & 0 & 6 & 1 \\ 7 & 8 & 4 & 0 & 5 \\ 2 & 3 & 5 & 5 & 0 \end{Bmatrix}$$

Εδώ δημιουργήθηκε ακμή μετάξυ των κόμβων 1-3, 1-5, 2-1, 2-5, 3-1, 3-2, 3-5, 4-1, 4-2, 4-5, 5-3, αντίστοιχα όπου αρχικά τέτοιες δεν υπήρχαν. Εντούτοις, δεν παρατηρείται καμία αλλαγή στα υπάρχοντα βάρη.

6.3 Εφαρμογή πραγματικού πρόβληματος

6.3.1 Πρώτη εφαρμογή σε πραγματικό πρόβλημα

Για την καλύτερη επαλήθευση της λειτουργίας του αλγορίθμου που παρήχθηκε, θεωρείται το παρακάτω πραγματικό πρόβλημα (με γνωστή λύση). Θεωρούνται οι ακόλουθες Ελληνικές πόλεις: Αθήνα, Πάτρα, Βόλος, Καλαμάτα, Τρίκαλα, Καρδίτσα, Λαμία.

Οι χιλιομετρικές (οδικές) αποστάσεις των πόλεων αυτών είναι γνωστές. Εντούτοις θεωρείται ένας γράφος όπου η Αθήνα, τα Τρίκαλα και η Καρδίτσα δεν είναι απαυθείας συνδεδεμένα, ήτοι δεν υπάρχει ακόμα διαδρομή που να τα συνδέει. Όλες οι υπόλοιπες συνδέσεις πόλεων υπάρχουν και μπορεί να υπάρξει κατεύθυνση και προς τις δύο κατευθύνσεις. Συνεπώς θεωρείται ένας γράφος με κορυφές τις πόλεις και αντίστοιχα βάρη ακμών τις χιλιομετρικές αποστάσεις μεταξύ τους με άπειρες αυτές μεταξύ Αθήνας - Καρδίτσας και Αθήνας - Τρικάλων. Συνεπώς ο πίνακας γειτνίασης θα είναι:

$$\begin{Bmatrix} 0 & 326 & 257 & \text{INF} & 214 & 218 & \text{INF} \\ 326 & 0 & 515 & 119 & 115 & 544 & 123 \\ 257 & 515 & 0 & 494 & 403 & 215 & 520 \\ \text{INF} & 119 & 494 & 0 & 91 & 279 & 27 \\ 214 & 115 & 403 & 91 & 0 & 188 & 117 \\ 218 & 577 & 215 & 279 & 188 & 0 & 305 \\ \text{INF} & 123 & 520 & 27 & 117 & 305 & 0 \end{Bmatrix}$$

Όπου η σειρά των πόλεων είναι η ακόλουθη: Αθήνα, Βόλος, Καλαμάτα, Καρδίτσα, Λαμία, Πάτρα, Τρίκαλα. Η εφαρμογή του αλγορίθμου αποδίδει τον ακόλουθο πίνακα:

$$\begin{Bmatrix} 0 & 326 & 257 & 305 & 214 & 218 & 331 \\ 326 & 0 & 515 & 119 & 115 & 544 & 123 \\ 257 & 515 & 0 & 494 & 403 & 215 & 520 \\ 305 & 119 & 494 & 0 & 91 & 279 & 27 \\ 214 & 115 & 403 & 91 & 0 & 188 & 117 \\ 218 & 577 & 215 & 279 & 188 & 0 & 305 \\ 331 & 123 & 520 & 27 & 117 & 305 & 0 \end{Bmatrix}$$

Προστέθηκαν λοιπόν οι χιλιομετρικές αποστάσεις που έλειπαν. Ο πίνακας στην εικόνα που ακολουθεί παρουσιάζει τις πραγματικές χιλιομετρικές αποστάσεις των πόλεων:

ΠΙΝΑΚΑΣ ΧΙΛΙΟΜΕΤΡΙΚΩΝ ΑΠΟΣΤΑΣΕΩΝ

	ΑΘΗΝΑ	ΒΟΛΟΣ	ΚΑΛΑΜΑΤΑ	ΚΑΡΔΙΤΣΑ	ΛΑΜΙΑ	ΠΑΤΡΑ	ΤΡΙΚΑΛΑ
ΑΘΗΝΑ	-	326	257	305	214	218	331
ΒΟΛΟΣ	326	-	515	119	115	544	123
ΚΑΛΑΜΑΤΑ	257	515	-	494	403	215	520
ΚΑΡΔΙΤΣΑ	305	119	494	-	91	279	27
ΛΑΜΙΑ	214	115	403	91	-	188	117
ΠΑΤΡΑ	218	544	215	279	188	-	305
ΤΡΙΚΑΛΑ	331	123	520	27	117	305	-

Σχήμα 6.10: Πραγματικές αποστάσεις Ελληνικών πόλεων

Είναι εμφανές οτι ο αλγόριθμος κατά την λύση του προβλήματος υπολόγισε τις πραγματικές αποστάσεις που έλειπαν μεταξύ των πόλεων. Αυτό συνεπάγεται οτι η λειτουργία του είναι σωστή. Υπενθυμίζουμε σε αυτό το σημείο οτι οι χιλιομετρικές αποστάσεις που παρουσιάζονται είναι οι ελάχιστες όπως μετρώνται στον χάρτη της χώρας. Ως εκ τούτου, η επίλυση του προβλήματος όντως θα έπρεπε να αποδώσει τις παραπάνω τιμές.

6.3.2 Δεύτερη εφαρμογή σε πραγματικό πρόβλημα

Στην δεύτερη και τελική δοκιμή του αλγορίθμου, στο πρόβλημα αποστάσεων των πόλεων τοποθετήθηκαν περισσότερες τιμές ίσες με άπειρο μεταξύ των αποστάσεων των πόλεων,

ώστε να αυξηθεί η δυσκολία για τον αλγόριθμο. Συνεπώς επιλέχθηκε ο πίνακας γειτνίασης να είναι ο ακόλουθος:

$$\begin{Bmatrix} 0 & INF & 257 & INF & 214 & 218 & INF \\ INF & 0 & 515 & 119 & 115 & 544 & 123 \\ 257 & 515 & 0 & 494 & 403 & 215 & 520 \\ INF & 119 & 494 & 0 & INF & 279 & 27 \\ 214 & 115 & 403 & INF & 0 & 188 & 117 \\ 218 & 577 & 215 & 279 & 188 & 0 & 305 \\ INF & 123 & 520 & 27 & 117 & 305 & 0 \end{Bmatrix}$$

Η εφαρμογή του αλγορίθμου αποδίδει τον ακόλουθο πίνακα:

$$\begin{Bmatrix} 0 & 326 & 257 & 305 & 214 & 218 & 331 \\ 326 & 0 & 515 & 119 & 115 & 544 & 123 \\ 257 & 515 & 0 & 494 & 403 & 215 & 520 \\ 305 & 119 & 494 & 0 & 91 & 279 & 27 \\ 214 & 115 & 403 & 91 & 0 & 188 & 117 \\ 218 & 577 & 215 & 279 & 188 & 0 & 305 \\ 331 & 123 & 520 & 27 & 117 & 305 & 0 \end{Bmatrix}$$

Είναι εμφανές οτι και σε αυτή την περίπτωση ο αλγόριθμος λειτουργεί με σωστό τρόπο καθότι υπολογίζονται ξανά οι χιλιομετρικές αποστάσεις που παρουσιάστηκαν παραπάνω όπως είναι στην αντίστοιχη εικόνα.

Κεφάλαιο 7

Συμπεράσματα

Η παρούσα εργασία εστίασε στην μελέτη, την παρουσίαση καθώς και την υλοποίση του αλγορίθμου Floyd -Warshal για την επίλυση προβλημάτων εύρεσης συντομότερης διαδρομής.

Κατά την υλοποίηση του αλγορίθμου διαπιστώθηκαν κάποιες παρατηρήσεις:

Σχολιάζοντας την δυνατότητα μη κατευθυνόμενου και κατευθυνόμενου γράφου ο πίνακας βαρών ενός μη κατευθυνόμενου γράφου είναι συμμετρικός, ενώ, ο κατευθυνόμενος γράφος λόγω της κατεύθυνσης των ακμών μας ορίζει πιο ατομικά την κάθε κίνηση στο γράφο.

Χρησιμοποιούνται δύο τρόποι αναπαράστασης του γράφου ο πίνακας και η λίστα γειτνίασης. Στην αναπαράσταση αυτών παρατηρήθηκε η λίστα γειτνίασης λόγω της απλότητας της μας παρέχει την ευκολία εισαγωγής και διαγραφής κόμβων και ακμών όπου είναι καλή για την αναπαράσταση ενός αραιού γράφου. Ενώ, ο πίνακας γειτνίασης είναι ιδανικός για την αναπαράσταση πυκνών γράφων, αφού δεν μας παρέχει επιπρόσθετες πληροφορίες.

Παρατηρήθηκαν περιπτώσεις κατευθυνόμενου γράφου στις οποίες μετά το τέλος του αλγορίθμου δημιουργήθηκαν καινούργιες τιμές και ακμές εκεί που δεν υπήρχαν και περιπτώσεις που δημιουργήθηκαν καινούργιες ακμές χωρίς να αλλάξουν τα βάρη των ήδη υπαρχουσών ακμών από τον αρχικό πίνακα γειτνίασης αντικαθιστώντας με τιμές βάρους κόστους.

Επίσης, εφαρμόζονται παραδείγματα πραγματικού γράφου χιλιομετρικών αποστάσεων ελληνικών πόλεων, διαπιστώνοντας ότι ο αλγόριθμος μπορεί να εφαρμοστεί και σε προβλήματα του πραγματικού κόσμου. Σκοπός του προβλήματος είναι η επαλήθευση των αποτελεσμάτων.

Χαρακτηριστικά αλγορίθμου:

Οι δυσκολίες που αντιμετωπίζει ο αλγόριθμος στον υπολογισμό της συντομότερης διαδρομής είναι οι αρνητικοί κύκλοι οι οποίοι μετατρέπουν σε NP-hard πρόβλημα και δεν εμφανίζει το σωστό αποτέλεσμα. Επίσης αυξάνει τις δυσκολίες επίτευξης του αλγορίθμου και η αβεβαιότητα εφαρμογής σε πραγματικό κόσμο.

Ο αλγόριθμος έχει την δυνατότητα της εύκολης κατανόησης.

Μελλοντική προσφορά:

Θεωρούμε ότι ένας τέτοιου τύπου αλγόριθμος λόγω της απλότητας και ευκολίας μάθησης θα μπορούσε πολύ εύκολα εκτός της εφαρμογής του σε δίκτυα και την εφαρμογή του στα ελληνικά σχολεία ως τρόπος εισαγωγής στον τομέα του αλγόριθμων.

Στοιχεία εξέλιξης που θα μπορούσε να βελτιώσει τον αλγόριθμο είναι η προσέγγιση του προς τους αρνητικούς κύκλους. Δίνοντας με αυτόν τον τρόπο μεγαλύτερες δυνατότητες στον αλγόριθμο για τον κόσμο των δικτύων.

Κατάλογος Σχημάτων

Bibliography

[1] Biggs, N.; Lloyd, E.; Wilson, R.(1986) , Graph Theory, Oxford University Press

[2] MEYER, U. (2004), Design and analysis of sequential and parallel single-source shortest paths algorithms, S.l, s.n.

[3] Rosen, Kenneth H. (2007) Boston: McGraw-Hill Higher Education, Discrete Mathematics and Its Applications, 7th Edition, MLA print

[4] Cormen T. H., Leiserson C. E.,Riverst R. L., Stein C.(2009), Introduction to Algorithms, 3d Edition, MIT Press

[5] Hochbaum D. S.(2014), Graph Algorithms and Network Flows, Lecture Notes for IEOR 266,

[6] Joyner D., Nguen M., Cohen N.(2011), Algorithmic Graph Theory, Version 7, Flooved

[7] Schrijver A. (2003), Combinatorial Optimization: Polyhedra and Efficiency, Springer, Google Books

[8] Jonathan L. G, Yellen J. (2003), Handbook of Graph Theory, Discrete Mathematics and Its Applications, CRC Press, ISBN 9780203490204, Google Books

[9] Glabowski ., Musznicki B., Nowak P., Zwierzykowski P. (2013), Efficiency Evaluation of Shortest Path Algorithms, AICT 2013, The Ninth Advanced International Conference on Telecommunications

[10] Tarau P. (2013), Dijkstra's Algorithm, Computer Science and Engineering, University of North Texas

[11] Chan T. M. (2007), More Algorithms for All-Pairs Shortest Paths in Weighted Graphs, University of Waterloo, Waterloo, ON, Canada, ISBN: 978-1-59593-631-8 doi¿10.1145/1250790.1250877

[12] Aini A., Salehipour A., (January 2012), Speeding up the Floyd–Warshall algorithm for the cycled shortest path problem, Applied Mathematics Letters, Volume 25, Issue 1, ISSN 0893-9659

[13] Hougardy S., (2010) The Floyd–Warshall algorithm on graphs with negative cycles, Information Processing Letters, Volume 110, ISSN 0020-0190

[14] Lehmann D. J., (1977), Algebraic structures for transitive closure, Theoretical Computer Science, Volume 4, Issue 1, ISSN 0304-3975

[15] Chegireddy C. R., Hamacher H. W., (1987), Algorithms for finding K-best perfect matchings, Discrete Applied Mathematics, Volume 18, Issue 2, ISSN 0166-218X

[16] v Kolosovskiy M. A. , (2009), Data structure for representing a graph: combination of linked list and hash table, Altai State Technical University, Russia

[17] Bertsekas D. P.,(1992), Linear network optimization: Algoritmhs and Codes, 2nd edition, MIT Press

[18] Papadopoulou C. (2011), Επεξεργασία και Αναπαράσταση Βέλτιστων Διαδρομών σε Οδικά Δίκτυα, ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟΠΟΛΥΤΕΧΝΕΙΟ