Αλγόριθμος Dijkstra

Ο αλγόριθμος του Dijkstra χρησιμοποιείται για την εύρεση τη συντομότερης διαδρομής από έναν κόμβο προς όλους του υπόλοιπους σε έναν γράφο με μη αρνητικά βάρη.

Αρχικά, θέτουμε την απόσταση του αρχικού κόμβου από τον εαυτό του ίση με 0. Έπειτα, θέτουμε την απόσταση όλων των άλλων κόμβων από τον αρχικό, ίση με έναν πολύ μεγάλο αριθμό (θεωρητικά άπειρο). Επιλέγουμε τον κόμβο με τη μικρότερη απόσταση από αυτόν, ο οποίος ΔΕΝ ΕΧΕΙ ΑΚΟΜΑ ΕΠΕΞΕΡΓΑΣΤΕΙ και ενημερώνουμε τις αποστάσεις των γειτονικών του κόμβων αν βρούμε μικρότερες μέσω του κόμβου αυτού. Επαναλαμβάνουμε έως ότου επεξεργαστούν όλοι οι κόμβοι ή μέχρι να βρούμε τη διαδρομή προς τον επιθυμητό κόμβο.

Πολυπλοκότητα $\Rightarrow O(|V|^2)$ για ουρά με πίνακα
$\Rightarrow O(|E|\cdot\log|V|)$ για ουρά με δυαδικό σωρό
$\Rightarrow O(|E| + |V|\cdot\log|V|)$ για ουρά Fib

Αλγόριθμος Bellman – Ford

Ο αλγόριθμος Bellman – Ford χρησιμοποιείται για την εύρεση της συντομότερης διαδρομής από έναν κόμβο προς όλους τους υπόλοιπους, σε έναν γράφο ακόμα και αν αυτός έχει αρνητικά βάρη. Επιπρόσθετα, ελέγχει αν υπάρχουν αρνητικοί κύκλοι και κατά πόσο εφόσον υπάρχουν, επηρεάζουν την ύπαρξη λύσης.

Αρχικά, θέτουμε την απόσταση του αρχικού κόμβου από τον εαυτό του ίση με 0. Έπειτα, θέτουμε την απόσταση όλων των άλλων κόμβων από τον αρχικό, ίση με έναν πολύ μεγάλο αριθμό (θεωρητικά άπειρο). Για κάθε ακμή στο γράφημα, ελέγχουμε αν μπορούμε να μειώσουμε την απόσταση προς τον κόμβο-προορισμό μέσω του κόμβου - πηγή. Επαναλαμβάνουμε το παραπάνω $|V| - 1$ φορές (όπου $|V|$ είναι το πλήθος των κόμβων). Εάν μπορούμε να μειώσουμε την απόσταση σε κάποια ακμή μετά την επανάληψη, τότε υπάρχει αρνητικός κύκλος και άρα δεν υπάρχει νόημα η αναζήτηση συντομότερης διαδρομής.

Πολυπλοκότητα $\Rightarrow O(V\cdot E)$

Αλγόριθμος Prim

Ο αλγόριθμος Prim χρησιμοποιείται για την εύρεση του ελάχιστου συνδετικού δένδρου ενός συνεκτικού γράφου με βάρη. Το ΕΣΔ είναι ένα υπογράφημα που συνδέει όλους τους κόμβους του γράφου με το μικρότερο δυνατό συνολικό κόστος διαδρομής.

Αρχικά, θέτουμε την απόσταση όλων των κόμβων από τον αρχικό στο άπειρο (θεωρητικά), εκτός από την απόσταση από τον εαυτό του, η οποία ισούται με 0. Εν συνεχεία, προσθέτουμε τον κόμβο με το μικρότερο κόστος στο ΕΣΔ και ενημερώνουμε τις αποστάσεις των γειτονικών κόμβων. Επαναλαμβάνουμε έως ότου όλοι οι κόμβοι του γράφου, συμπεριλαμβάνονται στο ΕΣΔ.

Πολυπλοκότητα ⇒ O(E·log(V)) με χρήση ουράς προτεραιότητας.
(Όπου E, το πλήθος των ακμών του γράφου)

Αλγόριθμος Kruskal

Ο αλγόριθμος Kruskal χρησιμοποιείται για την εύρεση του ελάχιστου συνδετικού δέντρου ενός συνεκτικού γράφου. Ο αλγόριθμος Kruskal λειτουργεί με τη λογική της επιλογής των ακμών με το μικρότερο βάρος, προσθέτοντάς τις μία προς μία στο ΕΣΔ, υπό την προϋπόθεση ότι δεν δημιουργούν κύκλους.

Αρχικά, ταξινομούμε όλες τις ακμές σε αύξουσα σειρά, με γνώμονα το βάρος τους. Έπειτα, για κάθε ακμή ελέγχουμε αν οι κόμβοι u και v ανήκουν στην ίδια συνιστώσα. Αν όχι, προσθέτουμε την ακμή στο ΕΣΔ και ενώνουμε τις συνιστώσες. Συνεχίζουμε μέχρι το ΕΣΔ να περιέχει |V| - 1 ακμές.

Πολυπλοκότητα ⇒ O(E·log(E))

Αλγόριθμος Hill climbing

Ο αλγόριθμος Hill Climbing χρησιμοποιείται για την εύρεση της καλύτερης λύσης σε προβλήματα μεγιστοποίησης ή ελαχιστοποίησης.

Αρχικά, ορίζουμε τον τρέχοντα κόμβο ως τη ρίζα του δένδρου. Όσο ο τρέχων κόμβος δεν είναι κόμβος στόχος βρίσκουμε τα παιδιά του και εντοπίζουμε αυτό με την ελάχιστη υπόλοιπη απόσταση από τον στόχο. Εάν ο τρέχων

κόμβος δεν έχει παιδιά ή το παιδί που βρέθηκε έχει μεγαλύτερη τιμή ευρετικής συνάρτησης από τον τρέχων κόμβο, βρήκαμε έναν στόχο! Ορίζουμε ως τρέχων κόμβο το παιδί του προηγούμενου τρέχοντα κόμβου και επαναλαμβάνουμε.

Αλγόριθμος Best – First

Ο Best First μοιάζει με τον Hill Climbing, μόνο που εδώ επεκτείνουμε όχι τον καλύτερο κόμβο από τα παιδιά του κόμβου που είμαστε, αλλά τον καλύτερο κόμβο από όλους τους κόμβους που βρίσκονται στο μέτωπο αναζήτησης του δένδρου (συνεπώς κρατάμε στο μέτωπο αναζήτησης όλα τα παιδιά του κόμβου που επεκτείνουμε).

Ο αλγόριθμος αναζήτησης Best - First Search είναι πιθανότερο να παράγει μικρότερα μονοπάτια από την αρχική κατάσταση στο στόχο.

Αρχικά, κατασκευάζουμε το μέτωπο αναζήτησης που περιέχει τη ρίζα του δένδρου (αρχική κατάσταση). Έως ότου το μέτωπο αναζήτησης αδειάσει ή βρεθεί ένας στόχος, εξετάζουμε κατά πόσο ο πρώτος κόμβος στη λίστα είναι κόμβος στόχος. Εάν είναι, ανακοινώνουμε επιτυχία, εάν δεν είναι, τότε βρίσκουμε τα παιδιά του και τα προσθέτουμε στο μέτωπο αναζήτησης το οποίο ταξινομούμε σε αύξουσα σειρά ευρετικής. Ελέγχουμε κατά πόσο ο πρώτος κόμβος του μετώπου αναζήτησης έχει παιδιά και αν δεν έχει τότε τον αφαιρούμε από το μέτωπο αναζήτησης.

Αλγόριθμος British Museum

Βρίσκουμε όλα τα μονοπάτια από την αρχική κατάσταση σε όλες τις καταστάσεις «στόχους» και επιλέγουμε το καλύτερο μονοπάτι.

Μπορούμε να βρούμε όλα τα μονοπάτια με αναζήτηση κατά βάθος ή αναζήτηση κατά-πλάτος. Μόνο που εδώ δεν σταματάμε όταν βρούμε λύση. Συνεχίζουμε μέχρι να βρούμε όλες τις λύσεις για να επιλέξουμε τη καλύτερη.

Αρχικά, δημιουργούμε ένα χώρο λύσεων που περιλαμβάνει όλες τις πιθανές λύσεις του προβλήματος. Κάθε λύση εξετάζεται μία προς μία (συνήθως σειριακά). Αν βρεθεί η σωστή λύση, η αναζήτηση σταματά.
Αν εξαντληθεί ο χώρος των λύσεων χωρίς να βρεθεί λύση, τότε το πρόβλημα είναι άλυτο.

Αλγόριθμος Branch and Bound

Ο αλγόριθμος Branch and χρησιμοποιείται για την επίλυση προβλημάτων βελτιστοποίησης, όπου το ζητούμενο είναι να βρεθεί η καλύτερη λύση ανάμεσα σε ένα σύνολο πιθανών λύσεων.

Αρχικά, κατασκευάζουμε μία λίστα από μονοπάτια η οποία είναι αρχικά κενή. Εν συνεχεία, εάν το πρώτο μονοπάτι οδηγεί σε στόχο το κρατάμε σαν πιθανή λύση. Εάν είναι καλύτερο από κάποια προηγούμενη λύση, τότε πάλι το κρατάμε σαν πιθανή λύση. Εάν το πρώτο μονοπάτι δεν οδηγεί σε στόχο, ή υπάρχουν άλλα μονοπάτια δεν έχουν ακόμη οδηγήσει σε στόχο και έχουν μικρότερο κόστος από το μονοπάτι που ήδη βρήκαμε, βγάζουμε το πρώτο μονοπάτι από τη λίστα φτιάχνουμε μονοπάτια που μπορούν να φτιαχτούν από το μονοπάτι που βγάλαμε επεκτείνοντας το κατά ένα βήμα και βάζουμε τα νέα μονοπάτια στη λίστα. Ταξινομούμε σε αύξουσα σειρά όλα τα μονοπάτια στη λίστα σύμφωνα με το κόστος του κάθε μονοπατιού (από την αρχική κατάσταση στο τελευταίο κόμβο του μονοπατιού) και «κλαδεύουμε» τα μονοπάτια που έχουν κόστος μεγαλύτερο από ένα όριο που διασφαλισμένα δεν μπορεί να οδηγήσει σε βέλτιστη λύση. Εάν βρήκαμε ένα μονοπάτι που οδηγεί σε κόμβο στόχο τότε ανακοινώνουμε μερική επιτυχία αλλιώς ανακοινώνουμε αποτυχία. Εάν βρήκαμε ένα μονοπάτι που οδηγεί σε κόμβο στόχο τότε ανακοινώνουμε επιτυχία αλλιώς ανακοινώνουμε αποτυχία.

Αλγόριθμος A*

Ακολουθούμε τη λογική του Branch and Bound και επεκτείνουμε το μονοπάτι με τον καλύτερο από όλους τους κόμβους που βρίσκονται στο μέτωπο αναζήτησης του δένδρου. Για το σκοπό αυτό χρησιμοποιούμε τη σύνθετη ευρετική συνάρτηση $F(k) = g(k) + h(k)$, όπου $g(k)$ η απόσταση της k από την αρχική κατάσταση, η οποία είναι πραγματική και γνωστή.
Όπου $h(k)$, μία εκτίμηση της απόστασης της k από το στόχο (μέσω μιας ευρετικής συνάρτησης).