ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΠΟΛΥΠΛΟΚΟΤΗΤΑ

3η Σειρά Ασκήσεων

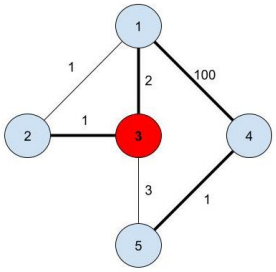
Ακαδημαϊκό έτος 2021-2022

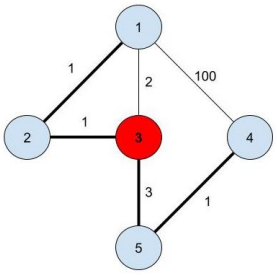
Κοκκινάκης Παναγιώτης – Α.Μ.: 03118115

Άσκηση 1: Ελάχιστο Συνδετικό Δέντρο με Περιορισμούς

Α) Θα χρησιμοποιήσουμε αντιπαράδειγμα για να δείξουμε ότι η άπληστη στρατηγική η οποία συμπεριλαμβάνει στο συνδετικό δέντρο τις k ακμές μικρότερου βάρους που προσπίπτουν στην s, δεν οδηγεί πάντα στη βέλτιστη λύση.

Στο παρακάτω παράδειγμα παρατηρούμε ότι με τη χρήση του άπληστου αλγορίθμου (αριστερά) για τον κόμβο 3 και βαθμό 2 το συνολικό βάρος του συνεκτικού δέντρου είναι 105, ενώ το ελάχιστο δέντρο για τα δοσμένα s και k (δεξιά) έχει βάρος 6.





Β)

- Διαγράφουμε τον κόμβο s και όλες τις ακμές του
- Εφαρμόζουμε τον αλγόριθμο Kruskal και βρίσκουμε τα ελάχιστα συνεκτικά δέντρα για κάθε συνιστώσα που προέκυψε, τα οποία συμβολίζουμε C_1, C_2 C_3, …, C_m.
- Προσθέτουμε στον γράφο τον κόμβο s και για κάθε συνιστώσα C, επιλέγουμε την ακμή ελάχιστου βάρους που τη συνδέει με τον κόμβο s. Εδώ διακρίνουμε τις εξής περιπτώσεις:
    1. m > k, δηλάδη οι συνιστώσες που προέκυψαν είναι περισσότερες από τον ζητούμενο βαθμό του κόμβου s, οπότε και το πρόβλημα δεν έχει λύση καθώς δεν υπάρχει τρόπος να φτιαχτεί συνεκτικό δέντρο αν ο βαθμός του s δεν είναι τουλάχιστον ίσος με τις υπόλοιπες συνιστώσες.
    2. m = k, οπότε και προσθέτουμε τις ελαφρύτερες ακμές μεταξύ της κάθε συνιστώσας και του s.
    3. m < k, οπότε και ξεκινάμε την ίδια διαδικασία με την περίπτωση 2 μέχρι να δημιουργηθεί ένα συνεκτικό δέντρο. Στη συνέχεια για κάθε ακμή του s που δεν έχουμε προσθέσει και με φθίνουσα σειρά βαρών,

εφαρμόζουμε DFS μέχρι να δημιουργηθεί κύκλος και βρίσκουμε την βαρύτερη ακμή του. Στη συνέχεια αποθηκεύουμε την τιμή της διαφοράς του βάρους της εκάστοτε ακμής με την βαρύτερη ακμή, την ίδια την ακμή του s και τη βαρύτερη ακμή του κύκλου. Ταξινομούμε τη λίστα των ακμών και των διαφορών σε φθίνουσα σειρά με βάση την διαφορά και επιλέγουμε με τη σειρά τα στοιχεία της. Για κάθε στοιχείο ελέγχουμε αν υπάρχει στον πίνακα ακόμη η βαρύτερη ακμή στο συνεκτικό δέντρο. Εφόσον υπάρχει το αντικαθιστούμε με την πρώτη ακμή του s που συναντάμε στη λίστα. Επαναλαμβάνουμε μέχρι ο βαθμός του s να είναι ίσος με k.

Ορθότητα:

Ο αλγόριθμος δίνει ορθό αποτέλεσμα αφού σε κάθε περίπτωση επιλέγει τις ακμές του s που συμβάλλουν το λιγότερο δυνατό στο βάρος του MST.

Πολυπλοκότητα:

Το πρώτο σκέλος του αλγορίθμου, δηλαδή η εύρεση των συνεκτικών δέντρων των συνιστωσών, απαιτεί τη χρήση του αλγορίθμου Kruskal μία φορά, ενώ το δέυτερο σκέλος απαιτεί τη χρήση DFS συνολικά k – m φορές, την ταξινόμηση της λίστας των ακμών και μια γραμμική διάσχιση της (μπορούμε να χρησιμοποιήσουμε κατάλληλη δομή δεδομένων όπως hash map για να ελέγχουμε αν αφαιρέσαμε κάποια ακμή σε σταθερό χρόνο). Συνεπώς η συνολική πολυπλοκότητα θα είναι: $O(ElogE + (m-k)(V+E) + (k-m)log(k-m) + (k-m))$, όπου $m \leq V$ οπότε η πολυπλοκότητα γίνεται: $O(ElogE + (V-k)(V+E))$.

Άσκηση 2: Σχεδιασμός Videogame

1) Κάνουμε BFS στον γράφο διατηρώντας ταυτόχρονα έναν πίνακα αποστάσεων d στον οποίο κρατάμε την ελάχιστη θετική ενέργεια με την οποία μπορεί να φτάσει ο παίκτης στον συγκεκριμένο κόμβο. Αρχικοποιούμε τον πίνακα αυτόν με r στη θέση s και ∞ στις υπόλοιπες προτού τον διατρέξουμε και ξεκινάμε να ελέγχουμε. Σε κάθε βήμα του BFS, για κάθε ζεύγος κόμβων v, u που ενώνονται με ακμή ελέγχουμε αν ισχύει ότι: d[u] > d[v]+p[u] (για να ελέγξουμε αν υπάρχει ελαχιστοποίηση από το συγκεκριμένο μονοπάτι) και αν ισχύει ότι: d[v]+p[u] > 0 (για να διασφαλίσουμε ότι ο παίκτης μπορεί να φτάσει ζωντανός). Εφόσον ικανοποιούνται αυτές οι συνθήκες αλλάζουμε το d[u] → d[v]+p[u]. Μόλις ολοκλήρωθεί η διάσχιση του λαβυρίνθου ελεγχουμε την τιμή του d[t]. Εάν είναι διάφορη του απείρου σημαίνει ότι έχει βρεθέι ασφαλές μονοπάτι στον λαβύρινθο.

Πολυπλοκότητα:

Ο αλγόριθμος κάνει μία διάσχιση BFS και σε κάθε βήμα σταθερό χρόνο για τις προσβάσεις και τις αλλαγές στον πίνακα d. Επομένως η συνολική πολυπλοκότητα είναι O(V+E).

2) Αρχικά χρησιμοποιούμε τον αλγόριθμο Bellman-Ford στον γράφο του λαβυρίνθου για να εντοπίσουμε τους θετικούς κύκλους και στη συνέχεια τρέχουμε τον αλγόριθμο του ερωτήματος 1. Ελέγχουμε αν η τιμή του d[t] είναι ίση με το άπειρο. Εάν δεν είναι, ο αλγόριθμος ολόκληρώνεται και ο λαβύρινθος είναι r-ασφαλής. Σε αντίθετη περίπτωση, ελέγχουμε την τιμή των κύκλων που πήραμε από τον BF για να δούμε αν είναι προσβάσιμοι από την αφετηρία και κάνουμε DFS στους κόμβους τους για να ελέγξουμε αν συνδέονται με τον κόμβο t. Αν υπάρχει μονοπάτι που τους συνδέει σημαίνει πως ανεξάρτητα από το βάρος του, μπορει ο παίκτης να φτάσει στους κύκλους και στη συνέχεια να πάρει αρκετη ενέργεια ώστε να φτάσει στο τέρμα.

Πολυπλοκότητα:

Ο αλγόριθμος χρησιμοποιεί μία φορά τον αλγόριθμο BF, μία φορά τον αλγόριθμο του ερωτήματος 1 και ένα DFS επομένως η συνολική πολυπλοκότητα θα είναι: O(VE+V+E).

3) Χρησιμοποιούμε δυαδική αναζήτηση στις πιθανές τιμές του r και ελέγχουμε αν για κάθε τιμή είναι r-ασφαλής με τους αλγορίθμους των ερωτημάτων 1 και 2 (αναλόγως την περίπτωση). Για να βρούμε το εύρος των τιμών στο οποίο εφαρμοζουμε τον αλγόριθμο του ερωτήματος 1 με μηδενική αρχή κρατώντας και αρνητικές τιμές ενέργειας και την ελάχιστη τιμή του πίνακα ανα πάσα στιγμή. Η τιμή αυτή αποτελεί την μέγιστη τιμή με την οποία μπορεί να χρειαστεί να ξεκινήσουμε, οπότε είναι και το μέγιστο της δυαδικής αναζήτησης.

Άσκηση 3: Ταξίδι σε Περίοδο Ενεργειακής Κρίσης

1) Ο αλγόριθμος μας έχει τα εξής βήματα:

- Ψάχνουμε για κάθε κόμβο τον «κυρίαρχο» του στο μονοπάτι s-t, ώστε να βρούμε την φθηνότερη πιθανή τιμή της βενζίνης.
- Υπολογίζουμε την απόσταση του τρέχοντος κόμβου από τον κυρίαρχο και ελέγχουμε αν αυτή είναι μικρότερη από B. Αν είναι τότε γεμίζουμε όσο χρειάζεται και φτάνουμε στον κυρίαρχο κόμβο. Αλλιώς γεμίζουμε εντελώς και βρίσκουμε τον κόμβο με την ελάχιστη τιμή σε διάστημα απόστασης μέχρι B, ώστε να επαναλάβουμε τη διαδικασία εκεί.

Πολυπλοκότητα: ο αλγόριθμος έχει πολυπλοκότητα $O(n^2)$ αφού για κάθε ακμή από την οποία περνάμε ελέγχουμε τις ακμές μέχρι να βρούμε την «κυρίαρχη» και την ελάχιστη στο μονοπάτι.

Άσκηση 4: Επαναφορά της Ομαλότητας στη Χώρα των Αλγορίθμων

Ανάγουμε το συγκεκριμένο πρόβλημα σε πρόβλημα εύρεσης της μέγιστης ροής. Δημιουργούμε γράφο G που έχει m + k + 2 κόμβους, (m για κάθε βάση, k για κάθε εξτρεμιστή, μία πηγή s και μία καταβόθρα t). Ενώνουμε με ακμές την πηγή s με κάθε κόμβο που αναπαριστά εξτρεμιστή, και θέτουμε ως μέγιστη ροή το ελάχιστο κόστος για την αιχμαλωσία του εκάστοτε στρατιώτη. Κάνουμε το ίδιο με ακμές από τους κόμβους των βάσεων προς την καταβόθρα t. Προφανώς για να ελαχιστοποιήσουμε το κόστος καταστροφής μιας βάσης, ή αιχμαλωσίας ενός εξτρεμιστή χρησιμοποιούμε τον στρατιώτη που βρίσκεται πιο κοντά στην εκάστοτε θέση. Τέλος, ενώνουμε τους κόμβους που αναπαριστούν βάσεις και εξτρεμιστές με ακμές μεταξύ τους, εάν η απόσταση ενός εξτρεμιστή και μίας βάσης είναι μικρότερη από το δοσμένο d. Στις ακμές αυτές θέτουμε μέγιστη ροή το άπειρο.

Με τον τρόπο αυτό η μέγιστη ροή του γράφου θα είναι η λύση του ζητουμένου αφού τα bottlenecks χρησιμεύουν καθώς αναπαριστούν επιθέσεις χαμηλού κόστους. Χρησιμοποιούμε επομένως τον αλγόριθμο Ford-Fulkerson στον γράφο και παίρνουμε το ελάχιστο κόστος.

Πολυπλοκότητα:

Η δημιουργία του γράφου γίνεται σε γραμμικό χρόνο εάν έχουμε βρει τα ελάχιστα κόστη, τα οποία βρίσκουμε σε λογαριθμικό χρόνο με δυαδική αναζήτηση στους στρατιώτες, ενώ ο αλγόριθμος Ford-Fulkerson χρειάζεται O(Ef) χρόνο (f η μέγιστη ροή από την πηγή). Συνεπώς η συνολική πολυπλοκότητα του αλγορίθμου είναι O((m+k)logn + (m+k)f) (ο G έχει αριθμό ακμών της τάξης του m+k).

Άσκηση 5: Αναγωγές και NP-Πληρότητα

Τακτοποίηση Ορθογωνίων Παραλληλογράμμων:

Θα ανάγουμε το πρόβλημα στο γνωστό πρόβλημα 2-Partition.

Αρχικά δείχνουμε ότι το πρόβλημα ανήκει στο NP:

Πιστοποιητικό → η διάταξη των παραλληλογράμμων που χρειάζεται γραμμικό χρόνο για να ελεγχθεί επομένως το πρόβλημα ανήκει στο NP.

Μετασχηματισμός: Θεωρούμε ότι έχουμε n ορθογώνια A_i με εμβαδά $a_i \times \varepsilon$, (ε αμελητέα μικρό). Ψάχνουμε να βρούμε ορθογώνιο B τ.ω. να έχει εμβαδόν $t \times 2\varepsilon$, για το οποίο να ισχύει ότι $t = \frac{1}{2} \sum a_i$. Έτσι αν μπορούμε να βρούμε κατάλληλο t, έχουμε κάνει το A partition σε 2 ισομεγέθη υποσύνολα.