Συντομότερα Μονοπάτια παραδείγματα

Διδάσκοντες:

Αρ. Παγουρτζής, Δ. Φωτάκης,
Δ. Σούλιου, Παν. Γροντάς

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

Εθνικό Μετσόβιο Πολυτεχνείο

Ταξιδεύοντας με ηλεκτρικό αυτοκίνητο

Θεωρούμε κατευθυνόμενο γράφημα $G(V, E, w)$, με n κορυφές (ή πόλεις), m ακμές και θετικά μήκη $w : E \to \mathbb{N}_+$ στις ακμές, το οποίο αποτελεί μοντέλο του οδικού δικτύου μιας χώρας. Θέλουμε να ταξιδέψουμε από την πόλη s στην πόλη t. Το αυτοκίνητό μας είναι ηλεκτρικό και έχει αυτονομία α (δηλ. η απόσταση που διανύει το αυτοκίνητο μεταξύ δύο διαδοχικών φορτίσεων δεν μπορεί να ξεπερνά το α). Ευτυχώς σε κάποιες (όχι όλες τις) πόλεις υπάρχουν σταθμοί φόρτισης. Συγκεκριμένα, $C \subset V$ είναι το σύνολο των πόλεων που διαθέτουν σταθμό φόρτισης. Ευτυχώς, τουλάχιστον, οι πόλεις s και t διαθέτουν σταθμούς φόρτισης (δηλ. $s, t \in C$). Στα (α) και (β) παρακάτω, να αιτιολογήσετε την ορθότητα και την υπολογιστική πολυπλοκότητα των αλγορίθμων που θα διατυπώσετε.

(α) Εξετάζουμε αρχικά την περίπτωση που η αυτονομία του αυτοκινήτου μας είναι αρκετά μεγάλη, ώστε να μην απαιτείται ενδιάμεση φόρτιση για το ταξίδι μας από την s στην t. Θέλουμε όμως, στη διαδρομή, να επισκεφθούμε τουλάχιστον μία ακόμη πόλη με σταθμό φόρτισης, για να επιβεβαιώσουμε ότι οι σταθμοί φόρτισης στις ενδιάμεσες πόλεις είναι συμβατοί. Να διατυπώσετε αποδοτικό αλγόριθμο που υπολογίζει το συντομότερο $s - t$ μονοπάτι που διέρχεται από τουλάχιστον μία πόλη του C διαφορετική από τις s και t (υποθέτουμε ότι $|C| \geq 3$).

(β) Θεωρούμε ότι η αυτονομία α του αυτοκινήτου μας είναι σχετικά μικρή και αναμένεται να λειτουργήσει περιοριστικά για τη διαδρομή που θέλουμε να ακολουθήσουμε. Να διατυπώσετε αποδοτικό αλγόριθμο που υπολογίζει το συντομότερο $s - t$ μονοπάτι υπό τον περιορισμό ότι η απόσταση μεταξύ δύο διαδοχικών πόλεων με σταθμό φόρτισης στη διαδρομή μας δεν ξεπερνά την αυτονομία α του αυτοκινήτου μας.

(α) Εξετάζουμε αρχικά την περίπτωση που η αυτονομία του αυτοκινήτου μας είναι αρκετά μεγάλη, ώστε να μην απαιτείται ενδιάμεση φόρτιση για το ταξίδι μας από την s στην t. Θέλουμε όμως, στη διαδρομή, να επισκεφθούμε τουλάχιστον μία ακόμη πόλη με σταθμό φόρτισης, για να επιβεβαιώσουμε ότι οι σταθμοί φόρτισης στις ενδιάμεσες πόλεις είναι συμβατοί. Να διατυπώσετε αποδοτικό αλγόριθμο που υπολογίζει το συντομότερο $s - t$ μονοπάτι που διέρχεται από τουλάχιστον μία πόλη του C διαφορετική από τις s και t (υποθέτουμε ότι $|C| \geq 3$).

- Εφαρμόζουμε τον αλγόριθμο Dijkstra στο γράφημα G προκειμένου να υπολογίσουμε τις αποστάσεις $d(c_i)$ όλων των πόλεων $c_i \in C$ που διαθέτουν σταθμό φόρτισης από τον αρχικό κόμβο s.
- Δημιουργούμε ένα νέο γράφημα G' που είναι ουσιαστικά το G και στο οποίο έχουμε αντιστρέψει τη φορά όλων των ακμών. Ως αρχικό κόμβο θεωρούμε τώρα τον t.
- Εφαρμόζουμε ξανά τον Dijkstra στο G' προκειμένου να υπολογίσουμε τις αποστάσεις $d'(c_i)$ όλων των πόλεων που διαθέτουν σταθμό φόρτισης από τον αρχικό κόμβο t.
- Αναζητούμε το $min_{i \in C}(d(c_i) + d'(c_i))$
- Η πολυπλοκότητα είναι $O(mlogn)$ ή $O(m + nlogn)$.

(β) Θεωρούμε ότι η αυτονομία α του αυτοκινήτου μας είναι σχετικά μικρή και αναμένεται να λειτουργήσει περιοριστικά για τη διαδρομή που θέλουμε να ακολουθήσουμε. Να διατυπώσετε αποδοτικό αλγόριθμο που υπολογίζει το συντομότερο $s-t$ μονοπάτι υπό τον περιορισμό ότι η απόσταση μεταξύ δύο διαδοχικών πόλεων με σταθμό φόρτισης στη διαδρομή μας δεν ξεπερνά την αυτονομία α του αυτοκινήτου μας.

- Για κάθε $c_i \in C$ υπολογίζουμε τα συντομότερα μονοπάτια από το s στο c_i και από το c_i στο t καθώς και από το c_i στο c_j για κάθε $c_i, c_j \in C$. Τέλος βρίσκουμε το συντομότερο μονοπάτι από το s στο t.
- Δημιουργούμε νέο γράφο με κόμβους όλες τις πόλεις που διαθέτουν σταθμό φόρτισης (s, t, c_i). Προσθέτω ως ακμές τα συντομότερα μονοπάτια που υπολόγισα αρκεί να μην ξεπερνούν το α. Εφαρμόζω Dijkstra στο νέο γράφο και παίρνω το συντομότερο μονοπάτι από το s στο t
- Η πολυπλοκότητα είναι $O(|C|mlogn)$ ή $O(|C|(m+nlogn))$.

Επιβεβαίωση και αναπροσαρμογή συντομότερων μονοπατιών

Θεωρούμε ένα (ισχυρά συνεκτικό) κατευθυνόμενο γράφημα $G(V, E, w)$ με n κορυφές, m ακμές, και (ενδεχομένως αρνητικά) μήκη w στις ακμές. Συμβολίζουμε με $d(u, v)$ την απόσταση των κορυφών u και v στο G.

(α) Δίνονται n αριθμοί $\delta_1, \ldots, \delta_n$, όπου κάθε δ_k (υποτίθεται ότι) ισούται με την απόσταση $v_1 - v_k$ στο G. Να διατυπώσετε αλγόριθμο γραμμικού χρόνου που ελέγχει αν τα $\delta_1, \ldots, \delta_n$ πράγματι ανταποκρίνονται στις αποστάσεις των κορυφών από την v_1, δηλαδή αν για κάθε $v_k \in V$, ισχύει ότι $\delta_k = d(v_1, v_k)$. Αν αυτό αληθεύει, ο αλγόριθμός σας πρέπει να υπολογίζει και να επιστρέφει ένα Δέντρο Συντομότερων Μονοπατιών με ρίζα τη v_1.

(β) Υποθέτουμε ότι έχουμε υπολογίσει τις αποστάσεις $d(v_i, v_j)$ μεταξύ κάθε (διατεταγμένου) ζεύγους κορυφών $(v_i, v_j) \in V \times V$. Στη συνέχεια, το μήκος μιας ακμής $e = (x, y)$ μειώνεται σε $w'(x, y) < w(x, y)$. Να διατυπώσετε αλγόριθμο με χρόνο εκτέλεσης $O(n^2)$ που αναπροσαρμόζει τις αποστάσεις μεταξύ όλων των κορυφών (εφόσον βέβαια η μείωση δεν δημιουργεί κύκλο αρνητικού μήκους!).

(γ) Τι αλλάζει, σε σχέση με το (β), αν το μήκος μιας ακμής $e = (x, y)$ αυξηθεί σε $w'(x, y) > w(x, y)$; Μπορείτε να επεκτείνετε τον αλγόριθμο του (β) σε αυτή την περίπτωση; Αν ναι, να περιγράψετε την επέκταση του αλγορίθμου, αν όχι, να εξηγήσετε συνοπτικά τις βασικές διαφορές / δυσκολίες.

Ο αλγόριθμος είναι μια απλή παραλλαγή του BFS

- Ξεκινάμε το BFS από την κορυφή v_1 με $p(v_1) = NULL$
- Έστω ότι είμαστε στην κορυφή v_i και έστω v_j γειτονάς του.
- Αν $\delta_i + w_{ij} < \delta_j$ οι αποστάσεις είναι λάθος. Αν $\delta_i + w_{ij} = \delta_j$, σημειώνουμε την κορυφή και ανανεώνουμε τον πίνακα προγόνων $p[j] = i$. Αλλιώς, προχωράμε.
- Αν ολοκληρωθεί η διαδικασία ελέγχουμε τον πίνακα προγόνων εάν έχει σε κάποια θέση εκτός την πρώτη $NULL$. Αν έχει, τότε απορρίπτουμε τις δοθείσες αποστάσεις. Αλλιώς, επιστρέφουμε τον πίνακα προγόνων που αποτελεί το ΔΣΜ.
- Η πολυπλοκότητα είναι $O(n + m)$.

- Μειώνεται το μήκος της ακμής $e = (x, y)$ σε $w'(x, y) < w(x, y)$. Άρα για κάθε ζεύγος κορυφών v_i, v_j υπάρχει περίπτωση να μειωθεί η απόσταση μεταξύ τους μόνο εάν το μήκος του μονοπατιού που περνά από την e είναι μικρότερο από το δοθέν $d(v_i, v_j)$.
- Το μήκος αυτό είναι $d(v_i, x) + w'(x, y) + d(y, v_j)$ και υπολογίζεται προφανώς σε ένα υπολογιστικό βήμα. Έτσι εφόσον έχουμε $O(n^2)$ ζεύγη, η πολυπλοκότητα του αλγορίθμου είναι $O(n^2)$.

Αποφεύγοντας τη συμφόρηση

Στις μεγαλουπόλεις, πολλοί άνθρωποι μετακινούνται κάθε πρωί από τα προάστια προς το κέντρο (και αντίστροφα το απόγευμα). Επειδή σχεδόν όλοι χρησιμοποιούν συντομότερες διαδρομές (με βάση την απόσταση σε km) για τις μετακινήσεις τους, κάθε πρωί κατά τις ώρες αιχμής, όλοι οι δρόμοι που ανήκουν στις συντομότερες διαδρομές από τα μεγάλα προάστια προς το κέντρο είναι μποτιλιαρισμένοι. Επιδιώκοντας να ελαχιστοποιήσουμε τον χρόνο μετακίνησης, θέλουμε να υπολογίσουμε τη συντομότερη διαδρομή που αποφεύγει όλους αυτούς τους μποτιλιάρισμένους δρόμους.

Θεωρούμε λοιπόν ένα κατευθυνόμενο γράφημα $G(V, E, w)$, με n κορυφές, m ακμές και (μη αρνητικά) μήκη w στις ακμές, μια αφετηρία $s \in V$ (το προάστιο όπου κατοικούμε) και έναν τελικό προορισμό $t \in V$ (ο χώρος εργασίας μας στο κέντρο της πόλης). Θεωρούμε ότι μια ακμή e είναι *ανεπιθύμητη* αν ανήκει σε κάποιο συντομότερο $s - t$ μονοπάτι. Να διατυπώσετε έναν αποδοτικό αλγόριθμο που με είσοδο το γράφημα $G(V, E, w)$ και τις κορυφές s και t, υπολογίζει μια συντομότερη $s-t$ διαδρομή στο υπογράφημα του G που δεν περιλαμβάνει καμία ανεπιθύμητη ακμή (ή διαπιστώνει ότι δεν υπάρχει τέτοια διαδρομή). Να εξηγήσετε λεπτομερώς πως θα υλοποιηθεί κάθε βήμα του αλγόριθμου και να υπολογίσετε τον χρόνο εκτέλεσης που προκύπτει.

- Πρέπει να βρούμε όλες τις ακμές που ανήκουν σε κάποιο συντομότερο μονοπάτι από τον s στον t και να τις αφαιρέσουμε, ώστε να βρούμε μετά το συντομότερο μονοπάτι χωρίς αυτές.
- Αν έχουμε αφαιρέσει όλες τις *ανεπιθύμητες* ακμές (για παράδειγμα, αν έχουμε θέσει το βάρος τους ίσο με ∞), τότε μπορούμε να τρέξουμε τον αλγόριθμο Dijkstra για να βρούμε το συντομότερο μονοπάτι στο τροποποιημένο γράφημα. Αν σε κάποια επανάληψη πριν φθάσουμε στο t, το $D[u] = min_{v \notin S}\{D[v]\} = \infty$, τότε προφανώς δεν υπάρχει κανένα άλλο μονοπάτι $s - t$ (κι επομένως ούτε συντομότερο) αφού το γράφημα δεν είναι πλέον συνεκτικό.
- Μένει να υλοποιήσουμε τον αλγόριθμο εύρεσης των *ανεπιθύμητων* ακμών.

- Πρώτα, τρέχουμε τον αλγόριθμο Dijkstra στο γράφημά μας με αρχική κορυφή την s ώστε να υπολογίσουμε τις ελάχιστες αποστάσεις από αυτή την κορυφή προς όλες τις υπόλοιπες. Τώρα, όλες οι κορυφές είναι εξοπλισμένες με τις τελικές εκτιμήσεις $D[v] = d(s, v)$.
- Στο δεύτερο βήμα, τρέχουμε έναν τροποποιημένο BFS από το t: Ξεκινώντας από το t εξετάζουμε όλες τις εισερχόμενες σε αυτό ακμές. Ελέγχουμε για ποιές από αυτές ισχύει ότι $D[v] = D[t] + w(v, t)$. Αυτές προφανώς είναι *ανεπιθύμητες* ακμές αφού υπάρχει μονοπάτι $s - v$ το οποίο αν επεκταθεί με την ακμή (v, t) θα αποτελεί μονοπάτι $s - t$ με την ελάχιστη απόσταση. Προσθέτουμε τις κορυφές αυτές στην ουρά του τροποποιημένου BFS και τις σημειώνουμε υπό εξερεύνηση. Συνεχίζουμε με την επόμενη κορυφή της ουράς, διαγράφοντας μία κορυφή όταν εξερευνήσουμε τις εισερχόμενες ακμές της και ελέγχοντας αν μία κορυφή είναι ήδη υπό εξερεύνηση πριν μπει στην ουρά (όπως στον BFS).

Η πολυπλοκότητα υπολογίζεται ως εξής:

- Εκτέλεση Dijkstra με αρχική κορυφή την s: $O(m + n \log n)$
- Υπολογισμός λιστών εισερχόμενων ακμών για κάθε κορυφή από τις λίστες γειτνίασης: $O(m)$
- Εκτέλεση τροποποιημένου BFS για εύρεση *ανεπιθύμητων* ακμών: $O(n + m)$
- Αφαίρεση *ανεπιθύμητων* ακμών: $O(m)$
- Εκτέλεση Dijkstra στο τροποποιημένο γράφημα για εύρεση συντομότερου μονοπατιού χωρίς *ανεπιθύμητες* ακμές: $O(m + n \log n)$

Άρα συνολικά: $O(m + n \log n)$

Προτάσεις Φίλων

Σε μια πλατφόρμα κοινωνικής δικτύωσης, θέλουμε να υλοποιήσουμε ένα σύστημα πρότασης φίλων. Για κάθε ζευγάρι συνδεδεμένων χρηστών i και j, οι αναλυτές της συμπεριφοράς των χρηστών έχουν εκτιμήσει την "εμπιστοσύνη" $t(i,j) \in (0,1)$ που δείχνει ο i για τον j με βάση τη συχνότητα αλληλεπίδρασής τους. Η εκτίμηση της "εμπιστοσύνης" δεν είναι κατ' ανάγκη συμμετρική (δηλ. μπορεί να ισχύει ότι $t(i,j) \neq t(j,i)$), ενώ αν δύο χρήστες i και j δεν είναι συνδεδεμένοι, τότε $t(i,j) = t(j,i) = 0$. Οι αναλυτές της συμπεριφοράς των χρηστών έχουν υπολογίσει ζεύγη παραμέτρων $(2,\beta_2),\ldots,(k,\beta_k)$, και θεωρούν ότι ο j αποτελεί καλή πρόταση για τον i, αν υπάρχει "κατευθυνόμενο μονοπάτι" $p = (u_0 = i, u_1, \ldots, u_\ell = j)$ μήκους ℓ, $2 \leq \ell \leq k$, που συνδέει τον i με τον j και εξασφαλίζει συνολική "εμπιστοσύνη" τουλάχιστον β_ℓ, δηλ. $t(p) = \prod_{q=0}^{\ell-1} t(q,q+1) \geq \beta_\ell$.

Χρειάζεται να υλοποιήσουμε έναν αποδοτικό αλγόριθμο που υπολογίζει όλες τις καλές προτάσεις φίλων για δεδομένο χρήστη i. Πρέπει να αιτιολογήσουμε την ορθότητα του αλγορίθμου και πρέπει να εκτιμήσουμε την υπολογιστική του πολυπλοκότητα. Για την εκτίμηση της πολυπλοκότητας, θεωρούμε ότι το πλήθος των χρηστών είναι n, το πλήθος των συνδέσεων μεταξύ των χρηστών είναι m, και ότι το k είναι πολύ μικρό σε σχέση με το n (π.χ., το k μπορεί να είναι σταθερό ή $O(\log n)$).

- Πρέπει να βρούμε όλες τις καλές προτάσεις φίλων για τον i ανάμεσα σε όλους του υπόλοιπους χρήστες.
- Για κάθε χρήστη j, αν βρούμε ένα μονοπάτι μήκους $2 \leq \ell \leq k$ για το οποίο η συνολική εκτίμηση εμπιστοσύνης είναι μεγαλύτερη από β_ℓ, πρέπει να τον προσθέσουμε στις καλές προτάσεις.
- Αρκεί να βρούμε για κάθε j και για κάθε μήκος μονοπατιού (=πλήθος ακμών) τον μονοπάτι με τη μεγαλύτερη συνολική εκτίμηση και αν αυτό πληροί τις προϋποθέσεις της παραμέτρου που αντιστοιχεί στο μήκος του, τότε προφανώς είναι καλή πρόταση. Διαφορετικά, αν δεν τις πληροί για κανένα μήκος μονοπατιού, δεν είναι.
- Το πρόβλημα παραπέμπει στο πρόβλημα Υπολογισμού Συντομότερων Μονοπατιών από μία κορυφή προς όλες τις άλλες. Αλλά πώς πρέπει να το διαμορφώσουμε;

- Για κάθε j και κάθε περίπατο p από το i σε αυτό, θέλουμε να βρούμε το μέγιστο $t(p) = \Pi_{q=0}^{q=\ell-1} t(q, q+1)$. Δηλαδή το μέγιστο $\log t(p) = \sum_{q=0}^{q=\ell-1} \log t(q, q+1)$, δηλαδή το ελάχιστο $-\log t(p) = \sum_{q=0}^{q=\ell-1} \log \frac{1}{t(q,q+1)}$.
- Επομένως, αν σχηματίσουμε το γράφημα όλων των n ατόμων και θέσουμε βάρος κάθε ακμής $e = (u, v)$ το $w(u, v) = \log \frac{1}{t(u,v)}$ (θετικό αφού $t(u, v) \in (0, 1)$) και δεν έχουμε καμία ακμή αν $t(u, v) = 0$, μετασχηματίζουμε το πρόβλημα έτσι ώστε να αναζητάμε συντομότερους περίπατους από το i προς κάθε j μήκους $\ell \leq k$, που να έχουν συνολικό βάρος $\leq \log \frac{1}{\beta_\ell}$.

- Στις διαφάνειες του μαθήματος έχουμε δει τον αλγόριθμο Bellman-Ford σε μορφή δυναμικού προγραμματισμού που υπολογίζει το συντομότερο μονοπάτι από την αρχική στον u με το πολύ r ακμές $D[u, r] = min\{D[u, r-1], min_{v:(v,u)\in E} D[v, r-1] + w(v, u)\}$.
- Τρέχουμε παραλλαγή του αλγόριθμου έτσι ώστε να υπολογίζει τους συντομότερους περίπατους μηκους ακριβώς l. Η αναδρομή τώρα θα είναι $D[u, r] = min_{v:(v,u)\in E} D[v, r-1] + w(v, u)$. Αυτό που μας επιτρέπει να πάρουμε σωστά αποτελέσματα είναι ότι κάθε επανάληψη βασίζεται στις προηγούμενες τιμές ώστε να εξασφαλίζουμε ότι το μονοπάτι που υπολογίζουμε δεν ξεπερνά το δείκτη της επανάληψης.

- Μένει να τροποποιήσουμε λίγο ακόμα τον αλγόριθμο ώστε να μας δίνει τις καλές προτάσεις φίλων. Κάθε φορά που ανανεώνεται μία απόσταση $D[u, r]$ εξετάζουμε αν είναι $\leq \log \frac{1}{\beta_r}$ και μόνο τότε την αποθηκεύουμε ως καλή πρόταση. Προφανώς το μήκος κάθε περίπατου μεγαλώνει κατά ένα σε κάθε επανάληψη.
- Στο τέλος έχουμε ελέγξει όλα τα δυνατούς συντομότερους περίπατους με $\ell \leq k$ μήκος προς κάθε κόμβο j και έχουμε βρει τις καλές προτάσεις φίλων.

- Ο αλγόριθμος μας επιτρέπει να αναγνωρίζουμε ¨προβληματικά' στιγμιότυπα του προβλήματος, τα οποία μπορούμε να θεωρήσουμε ως περιπτώσεις ανάλογες των κύκλων αρνητικού μήκους. Τέτοιες περιπτώσεις μπορεί να προκύψουν αν τα β_l δεν αποτελούν αύξουσα ακολουθία σε σχέση με τον δείκτη l. Αρνητικός κύκλος θεωρείται σε αυτή την περίπτωση ένας περίπατο,ς που αρχικά επισκέπτεται έναν κόμβο και δεν τον θεωρεί καλή πρόταση και μετά ξαναγυρίζει σε αυτόν και τον θεωρεί καλή πρόταση.
- Μπορούμε να αναγνωρίζουμε τέτοιες περιπτώσεις αποθηκεύοντας από τον πίνακα αναδρομής $D[u, r]$ τον κόμβο, από τον οποίο φτάσαμε στον u ως κόμβο πατέρα. Ακολουθώντας αναδρομικά την αντίστροφη διαδρομή, μπορούμε να αναγνωρίζουμε τους κόμβους, οι οποίοι έγιναν καλές προτάσεις, επειδή συμμετέχουν σε κύκλους, ενώ πριν δεν ήταν καλές προτάσεις(σε περίπατο μικρότερου μήκους).

Η πολυπλοκότητα υπολογίζεται ως εξής:

- Η αρχικοποίηση όλων των αποστάσεων $D[u, 0]$ γίνεται σε $O(n)$ χρόνο.
- Σε κάθε επανάληψη, εξετάζουμε όλες τις ακμές από μία φορά (για κάθε κόμβο όλες τις γειτονικές του) και κάνουμε σταθερό αριθμό συγκρίσεων σε $O(m)$. Οι επαναλήψεις είναι συνολικά k άρα το δεύτερο μέρος γίνεται σε $O(km)$.

Επομένως έχουμε συνολική πολυπλοκότητα:

$$O(n + km)$$