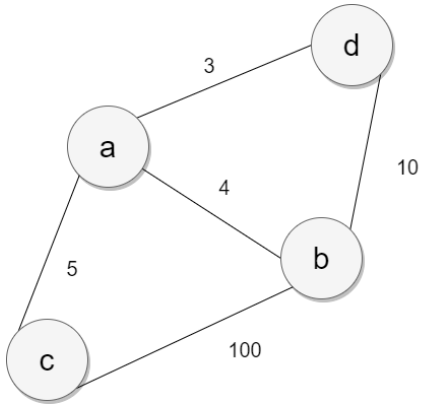
Αλγόριθμοι και Πολυπλοκότητα

3η Σειρά Γραπτών Ασκήσεων

7ο Εξάμηνο, Ακαδημαϊκό Έτος 2021 – 2022

Ονοματεπώνυμο	Αριθμός Μητρώου
Στεφανάκης Γεώργιος	el18436

Άσκηση 1η – Ελάχιστο Συνδετικό Δέντρο με Περιορισμούς



α) Παρατηρούμε ότι στον παραπάνω συνεκτικό μη-κατευθυνόμενο γράφο, για $k = 2$, και για τον κόμβο b, το άπληστο κριτήριο θα επιλέξει για το T^* τις ακμές 3, 4, 100 με συνολικό βάρος 107. Ωστόσο, θα μπορούσαμε να επιλέξουμε τις ακμές 3, 5, 10 με συνολικό βάρος 18, πολύ μικρότερο από το 107.

β) Για την κατασκευή ενός ελάχιστου συνδετικού δένδρου $T^*(s, k)$, όπου η κορυφή s θα έχει βαθμό k, θα κάνουμε τα εξής:

- Προσθέτουμε μία σταθερά C στις ακμές που προσπίπτουν στον s. Είναι προφανές ότι όσο μεγαλύτερη τιμή επιλέξουμε για το C, τόσο λιγότερες είναι οι ακμές που προσπίπτουν στον s θα συμπεριληφθούν στο MST, αφού μάλλον θα είναι βαρύτερες από άλλες ακμές του γράφου.
- Εφαρμόζουμε δυαδική αναζήτηση στις τιμές της σταθεράς C για να βρούμε τη βέλτιστη τιμή της.

Για να βρούμε λοιπόν τον βαθμό k της κορυφής s, θα αυξάνουμε τις ακμές που προσπίπτουν στην s κατά μία σταθερά C και ύστερα θα βρίσκουμε το MST του νέου γράφου που προκύπτει. Εάν η σταθερά είναι πολύ αρνητική τότε το MST θα συμπεριλαμβάνει πολλές από αυτές τις ακμές, αν όχι όλες. Αντιθέτως, εάν η σταθερά είναι πολύ μεγάλη, θα συμπεριληφθούν οι ελάχιστες από αυτές απαιτούμενες ακμές ώστε να κατασκευαστεί το MST, σε συνδυασμό με άλλες ακμές του γράφου. Συνεπώς αρκεί να εξετάσουμε τις τιμές της σταθεράς C που βρίσκονται στο σύνολο $\{e - z: e \in E \wedge z \in A\}$, όπου E το σύνολο των βαρών των ακμών του γράφου και A το σύνολο των βαρών των ακμών που προσπίπτουν στον κόμβο s. Η διαδικασία περιγράφεται παρακάτω:

- Ταξινόμηση του συνόλου των τιμών του C για την εφαρμογή της δυαδικής αναζήτησης. $\mathrm{O}(|E||A| \log |E||A|)$
- Δυαδική αναζήτηση στο σύνολο των C και εφαρμογή Kruskal στον νέο γράφο. $\mathrm{O}(\log |E||A| \times |E| \log |E|) \leq \mathrm{O}(\log |E||E| \times |E| \log |E|) = \mathrm{O}(2 \log |E| \times |E| \log |E|) = \mathrm{O}(|E| \log^2 |E|)$

Ο αλγόριθμος έχει συνολική πολυπλοκότητα $\mathrm{O}(|E||A| \log |E||A|) + \mathrm{O}(|E| \log^2 |E|)$.

Άσκηση 2η – Σχεδιασμός Videogame

1. Με είσοδο έναν άκυκλο λαβύρινθο $G(V, E, p)$, θέλουμε να αποφασίσουμε εάν ο G είναι r-ασφαλής. Αυτό σημαίνει ότι έχουμε ένα πρόβλημα απόφασης κατά το οποίο πρέπει να ελέγξουμε εάν ένας παίχτης με αρχική δύναμη r μπορεί να φτάσει από τον κόμβο s στον κόμβο t χωρίς να πεθάνει. Θεωρούμε πίνακα d με πλήθος στοιχείων όσοι και οι κόμβοι του G, αρχικοποιημένο με $-\infty$ σε όλες τις θέσεις του. Επαναληπτικά $|V| - 1$ φορές, θα ακολουθήσουμε τη λογική του αλγόριθμου Bellman – Ford ως εξής:
   - Για κάθε ακμή $(u, v) \in E$, ελέγχουμε εάν η μέχρι στιγμής δύναμη που μπορεί να έχει ο παίκτης φθάνοντας στον v μπορεί να είναι μεγαλύτερη από θα είχε εάν κατέφθανε στον κόμβο v από άλλο μονοπάτι. Εάν ικανοποιούνται οι συνθήκες $d[u] + p[v] > 0$ και $d[v] < d[u] + p[v]$, τότε ανανεώνουμε την τιμή του $d[v] = d[u] + p[v]$.
   - Στο τέλος κάθε επανάληψης, εάν προκύψει ότι $d[t] > 0$ τότε συμπεραίνουμε ότι μπορεί ο παίχτης να φτάσει στο τέλος του λαβυρίνθου με θετική δύναμη και άρα ο γράφος είναι r-ασφαλής.

Ο αλγόριθμος έχει πολυπλοκότητα $O(|V||E|)$.

2. Σε αυτό το ερώτημα, αρκεί να ελέγξουμε εάν κάποια από τις κορυφές των κύκλων που ενισχύουν τη δύναμη του παίκτη και αν η κορυφή t είναι προσβάσιμη από κάποιον κόμβο αυτού του κύκλου. Εάν αυτό ισχύει, τότε υπάρχει ασφαλής διαδρομή και ο γράφος είναι r-ασφαλής.
   - Εκτελούμε τον αλγόριθμο του προηγούμενου ερωτήματος και ελέγχουμε εάν $d[t] > 0$. Αν ισχύει, τότε ο λαβύρινθος είναι ασφαλής. Εάν όχι, τότε εκτελούμε έναν επιπλέον γύρο του αλγόριθμου του πρώτου ερωτήματος και εάν αλλάξει κάποια τιμή στον πίνακα d, τότε υπάρχει θετικός κύκλος ο οποίο ενισχύει τη δύναμη του παίκτη.
   - Έτσι εντοπίζουμε θετικούς κύκλους και μπορούμε να ελέγξουμε αν μπορούμε μέσω κάποιου κόμβου που ανήκει σε έναν από αυτούς να φτάσουμε στον t. Τα βάρη των ακμών δεν μας ενδιαφέρουν σε αυτό το σημείο καθώς μπορεί ο παίκτης να αυξήσει τη δύναμή του όσο επιθυμεί μέσα από τον θετικό κύκλο που εντοπίσαμε.

Η συνολική πολυπλοκότητα του αλγορίθμου είναι $O(|V||E|)$.

3. Μπορούμε να εφαρμόσουμε δυαδική αναζήτηση στο πεδίο τιμών του r έτσι ώστε να βρούμε την ελάχιστη τιμή του ώστε ο G να είναι r-ασφαλής. Αρχικά θα βρούμε την ελάχιστη δύναμη του 2 τέτοια ώστε αν το r πάρει αυτή την τιμή, ο λαβύρινθος θα γίνει μη ασφαλής. Έστω i αυτή η επανάληψη. Τότε, θα εφαρμόσουμε δυαδική αναζήτηση στο διάστημα $[2^{i-1}, 2^i]$. Εάν κάποια τιμή στο διάστημα δίνει ασφαλή γράφο, επαναλαμβάνουμε την αναζήτηση στο αριστερό ημιδιάστημα (συμπεριλαμβανομένου και του δεξιού άκρου) ενώ αν δίνει μη ασφαλή γράφο, κάνουμε αναζήτηση στο δεξί ημιδιάστημα για μεγαλύτερες τιμές του r. Λόγω της δυαδικής αναζήτησης, που θα κάνει το πολύ $\log r$ επαναλήψεις, θα έχουμε συνολική πολυπλοκότητα ίση με $O(|V||E| \log r)$.

Άσκηση 3η – Ταξίδι σε Περίοδο Ενεργειακής Κρίσης

α) Θα προσπαθήσουμε να γεμίσουμε με όσο το δυνατόν περισσότερη βενζίνη στους σταθμούς που έχουν καλύτερες τιμές και όσο το δυνατόν λιγότερη βενζίνη σε εκείνους που την έχουν πιο ακριβή. Αφού το ντεπόζιτό μας είναι χωρητικότητας B, προφανώς πρέπει σε όλες τις ακμές τα βάρη να είναι το πολύ ίσα με B. Αριθμούμε τις πόλεις με δείκτες που δείχνουν τη σειρά με την οποία ο δρομολογεί το μονοπάτι ο βέλτιστος αλγόριθμος ως $s, u_2, \dots, t$. Έτσι, μπορούμε να ακολουθήσουμε το εξής άπληστο κριτήριο:

- Εάν για μία πόλη i ισχύει ότι $c(i) < c(i+1)$ τότε θα γεμίσουμε όλο το ντεπόζιτο εκεί
- Εάν $c(i) \geq c(i+1)$ θα γεμίσουμε το ντεπόζιτο όσο χρειάζεται απλά για να φτάσουμε στην πόλη $i+1$

Μπορούμε να δείξουμε ότι αυτό το άπληστο κριτήριο μας δίνει και μία βέλτιστη λύση του προβλήματος, καθώς αν στην πόλη i ο βέλτιστος αλγόριθμος δεν γεμίσει όλο το ντεπόζιτο, τότε μπορούμε να αυξήσουμε την ποσότητα στην πόλη i και να την ελαττώσουμε στην επόμενη πόλη μειώνοντας έτσι το συνολικό κόστος της βενζίνης για την ίδια ποσότητα.

Έτσι πλησιάζουμε τον βέλτιστο αλγόριθμο στον άπληστο αλγόριθμο χωρίς να αυξάνεται το ποσό που ξοδεύουμε για βενζίνη. Αντίστοιχα, αν $c(i) \geq c(i+1)$ και ο optimal δεν γεμίσει όλο το ντεπόζιτο στην i, τότε μπορούμε να μειώσουμε την ποσότητα που γεμίζει ο βέλτιστος στην i-οστή ώστε να γίνει ίση με εκείνη που χρειάζεται για να φτάσουμε στην επόμενη πόλη, στην οποία μπορούμε να αγοράσουμε πιο φθηνή βενζίνη ενώ πάλι καταλήγουμε με την ίδια ποσότητα. Συνεπώς ο άπληστος αλγόριθμος είναι και βέλτιστος.

Τώρα μένει να βρούμε πως ενώ είμαστε σε μία πόλη j μπορούμε να βρούμε τον προηγούμενο σταθμό i, με $i \leq j$, με τη φθηνότερη βενζίνη από όσου σταθμούς ικανοποιούν τη συνθήκη $d_{ji} \leq B$. Επιπλέον, θέλουμε έναν τρόπο να μπορούμε να βρούμε τον αμέσως επόμενο σταθμό i, $j \leq i$, με τη φθηνότερη βενζίνη από αυτούς που ικανοποιούν τη συνθήκη $d_{ij} \leq B$. Δηλαδή, εάν έχουμε ισοπαλίες σε σταθμούς, πρέπει να επιλέγουμε αυτόν που είναι πιο κοντά στον κόμβο στόχο t.

Αφού φτιάξουμε τους πίνακες prev και next των οποίων τη λειτουργία περιγράψαμε προηγουμένως, θα ορίσουμε ένα παράθυρο (sliding window) που περιέχει κορυφές με συνολικό κόστος το πολύ B. Σε αυτό το παράθυρο, οι πόλεις με την ιδιότητα $\mathrm{prev}[i] = i$ είναι σημαντικές καθώς εάν επιλέξουμε ένα παράθυρο από τη θέση i και B μονάδες πίσω, η πόλη με την φθηνότερη βενζίνη είναι η i. Συνεπώς, προσπαθούμε να πάμε από μία πόλη i με $\mathrm{prev}[i] = i$ σε μία πόλη $j > i$ τέτοια ώστε $\mathrm{prev}[j] = j$. Οπότε διακρίνουμε τις εξής δύο περιπτώσεις:

- Αν $d_{ij} \leq B$ τότε γεμίζουμε τόσο όσο χρειαζόμαστε ώστε να φτάσουμε οριακά στην πόλη j
- Αλλιώς, αν $d_{ij} > B$ τότε θα γεμίσουμε πλήρως το ντεπόζιτο και θα μεταβούμε στην πόλη $\mathrm{next}[i]$

Οι παραπάνω πίνακες μπορούν να κατασκευαστούν αποδοτικά με σωρό ελαχίστου ο οποίος αξιοποιεί τα κόστη της βενζίνης σε κάθε σταθμό, σε χρόνο $\mathrm{O}(n \log n)$ ενώ ο συνολικός χρόνος για την εκτέλεση του αλγορίθμου είναι $\mathrm{O}(\log n)$ λόγω των προσθαφαιρέσεων στον σωρό επί $\mathrm{O}(n)$ καθώς διατρέχουμε μία φορά τους κόμβους, άρα συνολικά $\mathrm{O}(n \log n)$.

β) Το πρόβλημα με δυναμικό προγραμματισμό μπορεί να αναχθεί σε πρόβλημα συντομότερου μονοπατιού. Μπορούμε να χρησιμοποιήσουμε λογική Bellman-Ford ώστε για κάθε σταθμό του γράφου να υπολογίσουμε την συντομότερη διαδρομή μέσω της οποίας μπορούμε να φτάσουμε σε αυτόν με βενζίνη b. Για αυτό το λόγο, πρέπει να κοιτάξουμε όλα τα μονοπάτια που οδηγούν σε κάποιον κόμβο και όλους τους δυνατούς συνδυασμούς βενζίνης που προκύπτουν και να επιλέξουμε το βέλτιστο.

$$\mathrm{dp}[\mathrm{station}][b] = \min_{\substack{\forall (u,\mathrm{station}) \in E \\ 0 \leq b \leq B \\ 0 \leq w \leq b}} \begin{cases} \mathrm{dp}[\mathrm{station}][b] \\ \mathrm{dp}[u][w] + c(u) \times (b - w + C) \end{cases}$$

Ο DP πίνακας αρχικοποιείται με $+\infty$ σε όλες τις θέσεις του εκτός από τις θέσεις $\mathrm{dp}[s][x]$ που αρχικοποιούνται με x. Η λύση είναι η τιμή της θέσης $\mathrm{dp}[t][0]$ καθώς μας ενδιαφέρει να φτάσουμε στον κόμβο t με βενζίνη μηδέν. Με τον παραπάνω αλγόριθμο ελέγχουμε όλους τους πιθανούς συνδυασμούς αγοράς βενζίνης από κάθε σταθμό. Η πολυπλοκότητά του είναι $\mathrm{O}(|V||V|B^2) = \mathrm{O}(|V|^2\, B^2)$.

Άσκηση 4η - Επαναφορά της Ομαλότητας στη Χώρα των Αλγορίθμων

Το πρόβλημα μπορεί να αναχθεί σε πρόβλημα εύρεσης Min Cut – Max Flow. Για να κλείσει ο στρατός μία βάση ή να αιχμαλωτίσει έναν εξτρεμιστή, το ελάχιστο κόστος προκύπτει απ' το να χρησιμοποιήσει τον κοντινότερο στρατιώτη στον στόχο, αφού το κόστος είναι ανάλογο της απόστασης μεταξύ στρατιώτη και εξτρεμιστή ή βάσης. Θα κατασκευάσουμε έναν γράφο με πλήθος κόμβων ίσο με $m + k + 2$, όπου m το πλήθος των εξτρεμιστών, k το πλήθος των βάσεων και επιπλέον τους κόμβους s και t. Ύστερα θα συνδέσουμε κάθε κορυφή που αντιστοιχεί σε εξτρεμιστή με κάθε κόμβο βάσης που η μεταξύ τους απόσταση είναι μικρότερη ή ίση από d. Κάθε τέτοια ακμή θα έχει άπειρη χωρητικότητα. Επίσης θα συνδέσουμε κάθε εξτρεμιστή με τον κόμβο s, με χωρητικότητα το ελάχιστο κόστος για τον στρατό για να αιχμαλωτίσει

τον αντίστοιχο εξτρεμιστή, δηλαδή με την ελάχιστη απόσταση στρατιώτη από την εξτρεμιστή στην κεντρική αρτηρία. Επιπλέον, θα συνδέσουμε κάθε κόμβο που αντιστοιχεί σε βάση με την κορυφή t με χωρητικότητα το ελάχιστο κόστος για τον στρατό για να κλείσει τη συγκεκριμένη βάση, δηλαδή με την ελάχιστη απόσταση στρατιώτη από τη συγκεκριμένη βάση στην κεντρική αρτηρία.

Για την κατασκευή αυτού του γράφου, πρέπει αρχικά να ταξινομήσουμε τους εξτρεμιστές, τις βάσεις και τους στρατιώτες με βάση τη θέση τους στην κεντρική αρτηρία ώστε να έχουμε τα ελάχιστα κόστη που χρειάζονται για να συνδέσουμε εξτρεμιστές με τον κόμβο s και τις βάσεις με τον κόμβο t.

Το ζητούμενο αποτέλεσμα είναι το $s-t$ Min – Cut καθώς έτσι θα εξασφαλίσουμε ότι θα αιχμαλωτιστούν όλοι οι εξτρεμιστές είτε μέσω της καταστροφής κάποιας βάσης, είτε μεμονωμένα. Το Min – Cut θα περιέχει μόνο ακμές από το s σε εξτρεμιστές και από το t σε βάσεις εξτρεμιστών αφού οι ακμές μεταξύ εξτρεμιστών και βάσεων έχουν άπειρη χωρητικότητα.

Η ταξινόμηση των παραπάνω έχει πολυπλοκότητα ίση με $\mathrm{O}\big((n+m+k)\log(n+m+k)\big)$ και η κατασκευή του γράφου μπορεί να γίνει σε γραμμικό χρόνο. Το Min – Cut για να υπολογισθεί, μπορούμε να χρησιμοποιήσουμε Max – Flow το οποίο μπορούμε να υπολογίσουμε με Edmonds – Karp σε $\mathrm{O}(V^2E)$.

Άσκηση 5η - Αναγωγές και NP-Πληρότητα

Τακτοποίηση Ορθογωνίων Παραλληλογράμμων

Μπορούμε σε πολυωνυμικό χρόνο να ελέγξουμε ότι το παραλληλόγραμμο B καλύπτεται εξ' ολοκλήρου και ότι δεν υπάρχουν επικαλύψεις χωρίζοντας το B σε τετράγωνα εμβαδού ίσα με 1. Αυτό ισχύει καθώς τα $x_1, x_2, \dots, x_n$ είναι πολυωνυμικά μεγάλα σε σχέση με την είσοδο. Θα ανάγουμε το 2-Partition στο πρόβλημά μας για να αποδείξουμε πως είναι NP-πλήρες. Αν έχουμε είσοδο στο Partition $A=\{a_1, a_2, \dots, a_n\}$ τότε μπορούμε να φτιάξουμε n ορθογώνια διαστάσεων $1\times a_1, 1\times a_2, \dots, 1\times a_n$. Επιπλέον, μπορούμε να θεωρήσουμε ότι το ορθογώνιο B έχει διαστάσεις $2\times\frac{\mathrm{sum}\,A}{2}$ στο δικό μας πρόβλημα. Έτσι, αν τα ορθογώνια χωρέσουν μέσα στο B τότε, λόγω των διαστάσεων που δώσαμε στο B, θα έχουμε χωρίσει τα ορθογώνια σε δύο γραμμές με ίσο μήκος, χωρίς επικαλύψεις. Άρα, θα έχουμε χωρίσει το σύνολο A σε δύο συμπληρωματικά σύνολα με ίσο άθροισμα στοιχείων. Συνεπώς, το πρόβλημά μας απαντάει θετικά στην περίπτωση που υπάρχει Partition.

Μέγιστη Τομή με Βάση στις Κορυφές

Το πρόβλημα ανήκει στο NP καθώς αν γνωρίζουμε τις κορυφές που ανήκουν στο S, μπορούμε σε τετραγωνικό χρόνο να ελέγξουμε αν το άθροισμα των βαρών των ακμών που διασχίζουν την τομή είναι μεγαλύτερο του B. Θα κάνουμε ξανά αναγωγή 2-Partition πρόβλημα στο δικό μας για να δείξουμε ότι είναι NP-πλήρες. Έστω $A=\{w_1, w_2, \dots, w_n\}$ η είσοδος του Partition. Φτιάχνουμε τον πλήρη γράφο με κόμβους βάρους $w_1, w_2, \dots, w_n$. Θέλουμε τα βάρη που ανήκουν στο $(S,\ S\setminus V)$ να είναι μεγαλύτερα ή ίσα του B. Όμως αυτό είναι:

$$\sum_{\substack{u\in S\\ v\notin S}} w_u w_v = \sum_{u\in S} w_u \sum_{v\in V\setminus S} w_v = w(S)\big(w(A)-w(S)\big)$$

Καθώς έχουμε πλήρες γράφημα (clique). Συνεπώς αρκεί να ψάξουμε που μεγιστοποιείται η τομή, δηλαδή η παράσταση $f(x)=x(w(A)-x)$ για $x=\frac{w(A)}{2}$. Η συνάρτηση αυτή παρουσιάζει μέγιστο στο $x=\frac{w(A)}{2}$ την τιμή $\frac{A^2}{4}$, άρα επιλέγουμε το B ίσο με αυτήν. Έτσι η τομή θα αποτελεί partition του A. Συνεπώς, αν θέλουμε διαμερίσουμε ένα σύνολο A σε δύο

ισοβαρή σύνολα, αρκεί να μεγιστοποιήσουμε την τομή ενός γράφου με κόμβους όσοι και το πλήθος των στοιχείων του συνόλου A.

Συντομότερο Μονοπάτι με Περιορισμούς

Το πρόβλημα ανήκει στην κλάση NP αφού αν έχουμε δοσμένο το $s-t$ μονοπάτι, μπορούμε σε γραμμικό χρόνο να ελέγξουμε εάν απαντάει θετικά στο ερώτημα. Θα ανάγουμε το Knapsack στο πρόβλημά μας ώστε να αποδείξουμε ότι είναι NP-πλήρες. Το Knapsack έχει ως είσοδο n στοιχεία, με βάρος w_i και κόστος p_i το καθένα. Θέλουμε να επιβεβαιώσουμε ότι υπάρχει υποσύνολο αντικειμένων με βάρος μικρότερο ή ίσο του B και συνολικό κέρδος μεγαλύτερο ή ίσο του P. Θα κατασκευάσουμε έναν κατευθυνόμενο γράφο με κορυφές $x_0, x_1, \dots, x_n$ όπου $x_0 = s$ η αρχική κορυφή και τα υπόλοιπα στοιχεία τα αντικείμενα που μπορούμε να βάλουμε στο σακίδιο. Για κάθε ζεύγος αντικειμένων x_{i-1}, x_i θα δημιουργήσουμε ακμή e βάρους $w(e) = w_i$ και $c(e) = P_{sum} - p_i$, η οποία αντιστοιχεί στο να επιλέξουμε το i-οστό αντικείμενο, και μία ακμή e' βάρους $w(e') = 0$ και $c(e') = P_{sum}$. Επιπλέον, θέτουμε $x_0 = s$, $x_n = t$, $W = B$, $C = n\,P_{sum} - P$. Τότε, αν υπάρχει επιλογή αντικειμένων που φτιάχνουν αποδεκτό σακίδιο με βάση τους περιορισμούς τότε θα υπάρχει και μονοπάτι $s-t$ που ικανοποιεί τους περιορισμούς. Αν επιλέξουμε αντικείμενα $a_1, a_2, \dots, a_n$ να προστεθούν στο σακίδιο θα έχουμε:

$$w(a_1) + w(a_2) + \cdots + w(a_k) \le B = W$$

$$c(a_1) + c(a_2) + \cdots + c(a_k) + L \ge P \Rightarrow n\,P_{sum} - p_1 - p_2 - \cdots - p_k \ge P \Rightarrow p_1 + p_2 + \cdots + p_k \le n\,P_{sum} - P = C$$

Όπου L το κόστος των υπολοίπων ακμών που δεν επιλέχθηκαν.