ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΠΟΛΥΠΛΟΚΟΤΗΤΑ

3η ΣΕΙΡΑ ΓΡΑΠΤΩΝ ΑΣΚΗΣΕΩΝ

Αλέξανδρος Κουλάκος

03118144

ΣΗΜΜΥ, 7ο Εξάμηνο

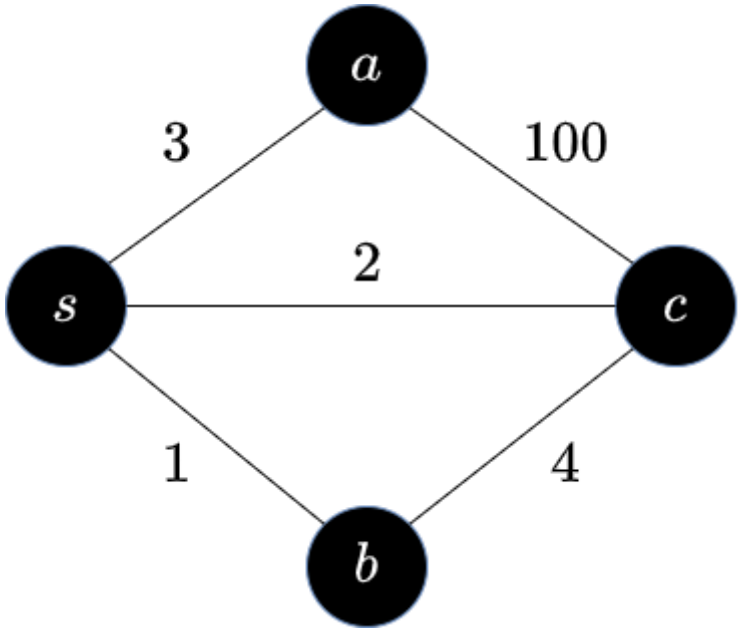
Φεβρουάριος, 2022

Άσκηση 1

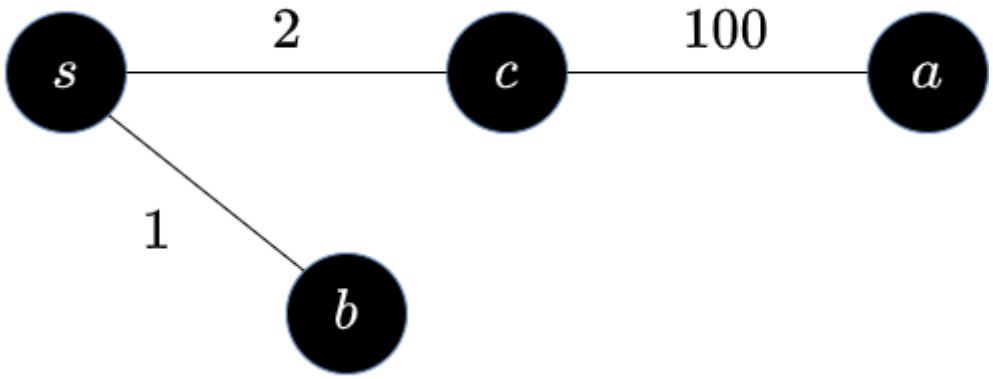
$(\boldsymbol{a})$

Αναζητούμε αντιπαράδειγμα.

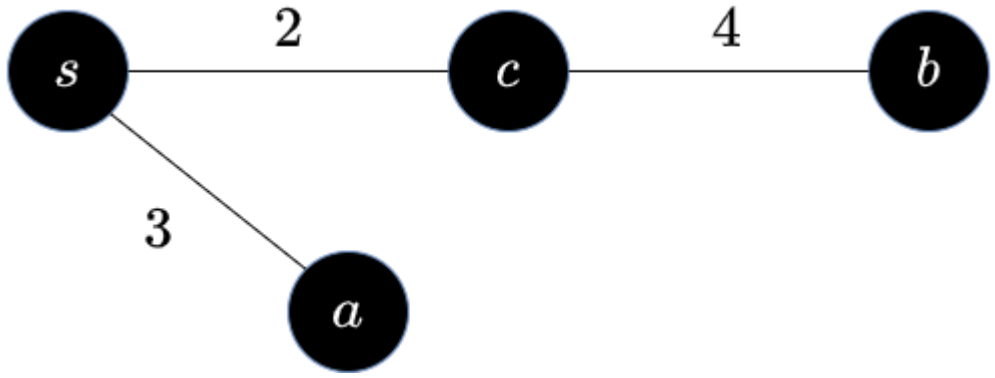
Θεωρούμε το παρακάτω γράφημα G το οποίο είναι πράγματι συνεκτικό και μη κατευθυνόμενο, σύμφωνα με τις προδιαγραφές της εκφώνησης:



Θεωρούμε $k = 2$, τότε η άπληστη στρατηγική κατά την οποία συμπεριλαμβάνουμε στο συνδετικό δέντρο τις δύο «φθηνότερες» ακμές που προσπίπτουν στην s, δηλαδή την $s - b$ και την $s - c$, παράγει το ακόλουθο συνδετικό δέντρο T με $w(T) = 1 + 2 + 100 = 103$:



Ωστόσο, διαισθητικά γνωρίζουμε ότι η βέλτιστη λύση περιγράφεται από το παρακάτω δέντρο T^* με $w(T^*) = 2 + 3 + 4 = 9$:



Παρατηρούμε ότι $w(T) > w(T^*)$, το οποίο πιστοποιεί ότι το δέντρο T (για την κατασκευή του οποίου χρησιμοποιείται το άπληστο κριτήριο) δεν είναι το ελάχιστο συνεκτικό δέντρο στο οποίο η κορυφή s έχει βαθμό 2. Συνεπώς, η άπληστη σταρατηγική δεν εγγυάται βέλτιστη λύση για το εν λόγω πρόβλημα.

Σχόλιο Στο γράφημα G που θεωρήσαμε, επιλέγοντας τις ακμές $s - b$ και $s - c$, όπως υπαγορεύει το άπληστο κριτήριο, η κορυφή s αποκτά βαθμό 2, συνεπώς η ακμή $s - a$ κόστους 3 δεν μπορεί να επιλεγεί, διότι θα αυξήσει κατά 1 το βαθμό της s. Έτσι, για να εξασφαλιστεί η συνεκτικότητα, στο δέντρο T πρέπει να συμπεριληφθεί η ακμή $c - a$ η οποία έχει κόστος 100 και αυξάνει ραγδαία το συνολικό κόστος του T, γεγονός που οδηγεί σε μη βέλτιστη λύση.

($\boldsymbol{\beta}$)

Έστω A το σύνολο των ακμών που προσπίπτουν στην κορυφή s. Αναζητούμε κατάλληλη σταθερά K η οποία προστιθέμενη σε κάθε $w(a)$, όπου $a \in A$, μας δίνει το επιθυμητό συνδετικό δέντρο.

Η γενική ιδέα πίσω από αυτό βασίζεται στον ισχυρισμό ότι το επιθυμητό συνδετικό δέντρο για κάθε βαθμό $1 \leq k \leq |A|$ προκύπτει ως το ελάχιστο συνδετικό δέντρο του αρχικού γραφήματος G όπου οι ακμές $a \in A$ έχουν επαυξηθεί αλγεβρικά κατά K. Πράγματι, για K πολύ αρνητικό στο συνδετικό δέντρο περιλαμβάνονται όλες οι ακμές που προσπίπτουν στην s και για K πολύ θετικό περιλαμβάνονται οι λιγότερες δυνατές. Προφανώς, κάθε βαθμός k θα ανήκει ανάμεσα στο ελάχιστο και μέγιστο πλήθος ακμών που περιλαμβάνονται.

Η επόμενη παρατήρηση σχετίζεται με το πλήθος των διαφορετικών τιμών του K που έχει νόημα να εξετάσουμε. Καταλήγουμε στο συμπέρασμα ότι ο βαθμός του κόμβου s στο επαγόμενο ελάχιστο συνδετικό δέντρο αλλάζει μόνο για εκείνες τις τιμές του K στις οποίες μια ακμή $a \in A$ αποκτά το ίδιο βάρος με κάποια ακμή $e \in E$, οπότε οι ενδιαφέρουσες τιμές του K ανήκουν στο σύνολο $X = \{w(e) - w(a): a \in A \text{ και } e \in E\}$, όπου αν το ίδιο άθροισμα $w(e) - w(a)$ εμφανίζεται πολλαπλές φορές, στο X κρατείται μόνο μια φορά. Στην ουσία, το σύνολο X περιλαμβάνει τις τιμές που πρέπει να αφαιρεθούν ή να προστεθούν σε κάθε $w(a)$, όπου $a \in A$, έτσι ώστε το βάρος της ακμής a να ισούται με το βάρος κάποιας άλλης ακμής e, για κάθε $e \in E$. Πράγματι, για να τροποποιηθεί το επαγόμενο ελάχιστο συνδετικό δέντρο, αρκεί μια ακμή του να αντικασταθεί με μια άλλη. Αυτό σημαίνει πως για κατάλληλο K, μια ακμή που προσπίπτει στην κορυφή s αποκτά το πολύ ίδιο βάρος με κάποια άλλη ακμή που προυπήρχε στο δέντρο και κάτι τέτοιο συμβαίνει αν το K ισούται με τη διαφορά των βαρών αυτών των δύο ακμών.

Έχοντας στη διάθεσή μας μια τιμή του K, μπορούμε να την προσθέσουμε σε κάθε $w(a)$, όπου $a \in A$, και εφαρμόζοντας τον αλγόριθμο του $Kruskal$ να αποφανθούμε σε χρόνο $O(|E|\log|E|)$ αν η κορυφή s στο επαγόμενο ελάχιστο συνδετικό δέντρο έχει βαθμό ίσο με k. Αν ισχύει κάτι τέτοιο, αρκεί να ελέγξουμε μικρότερες τιμές του k, αλλιώς αρκεί να ελέγξουμε μεγαλύτερες τιμές. Παρατηρούμε, λοιπόν, ότι μπορούμε να λύσουμε το πρόβλημα βελτιστοποίησης ανάγοντάς το στο αντίστοιχο

πρόβλημα απόφασης και εφαρμόζοντας $binary\ search$ στις τιμές του K βρίσκουμε το ελάχιστο συνδετικό δέντρο στο οποίο η κορυφή s έχει βαθμό ίσο με k. Για να εφαρμόσουμε $binary\ search$, απαραίτητη προϋπόθεση είναι το σύνολο X να είναι ταξινομημένο (σε αύξουσα σειρά). Χρειάζεται χρόνος $O(|X|\log|X|) = O(|E||A|\log(|E||A|))$ για την ταξινόμηση (π.χ. με $merge\ sort$). Επίσης, το πρόβλημα απόφασης επιλύεται $O(\log|X|) = O(\log(|E||A|)) = O(\log|E|^2) = O(\log|E|)$ φορές, διότι $|A| \leq |E|$. Άρα, η συνολική πολυπλοκότητα του αλγορίθμου μας είναι $O(|E||A|\log(|E||A|) + |E|\log^2|E|)$.

Άσκηση 2

1.

Θεωρούμε έναν πίνακα d με $|V|$ θέσεις. Το στοιχείο $d[u]$ εκφράζει τη μέγιστη εναπομείνουσα ενέργεια του παίκτη, αφού διασχίσει το δωμάτιο u. Έτσι, αρχικοποιούμε $d[s] = r$ και $d[u] = -\infty$, για κάθε $u \in V - \{s\}$. Στην ουσία, θέλουμε να σαρώσουμε όλα τα δυνατά μονοπάτια από την κορυφή s στην κορυφή t (από υπόθεση ξέρουμε ότι θα υπάρχει τουλάχιστον ένα) μέσω μιας επαναλαμβανόμενης DFS. Για κάθε ακμή $u \to v$ που βρίσκεται σε ένα από αυτά τα μονοπάτια ελέγχουμε αν ικανοποιούνται οι συνθήκες $d[u] + p(v) > 0$ και $d[u] + p(v) > d[v]$, δηλαδή ελέγχουμε αν μπορούμε να διασχίσουμε «ζωντανοί» το δωμάτιο v με μεγαλύτερη εναπομείνουσα ενέργεια από εκείνη που εξασφάλιζε κάποια προηγούμενη διάσχιση του δωματίου v (το καλύτερο σενάριο). Αν ισχύουν τα παραπάνω, ανανεώνουμε τον πίνακα d, έτσι ώστε $d[v] = d[u] + p(v)$. Διαφορετικά, η ετικέτα $d[v]$ παραμένει ως είχε. Είναι προφανές ότι στο τέλος του αλγορίθμου για όσες κορυφές u ισχύει $d[u] = c$, όπου c θετικό και πεπερασμένο, τότε αυτές οι κορυφές είναι προσβάσιμες μέσω της s και ο παίκτης μπορεί να τις διασχίσει παραμένονας «ζωντανός» με μέγιστη εναπομείνουσα ενέργεια $c > 0$. Αντίθετα, για όσες κορυφές u ισχύει $d[u] = -\infty$, τότε αυτές είτε δεν είναι προσβάσιμες μέσω της s είτε ο παίκτης δεν μπορεί να τις διασχίσει παραμένοντας «ζωντανός». Έτσι, με δεδομένο ότι η t είναι προσβάσιμη από την s, για να αποφανθούμε αν ο Λαβύρινθος G είναι $r - \text{ασφαλής}$, αρκεί να εξετάσουμε την τιμή $d[t]$ και να βεβαιωθούμε ότι $d[t] > 0$. Η πολυπλοκότητα του αλγορίθμου μας είναι $O(|V||E|)$, διότι για κάθε κορυφή που επισκεπτόμαστε ($O(|V|)$ το πλήθος) εξετάζουμε όλες τις ακμές που εξέρχονται από αυτή ($O(|E|)$ το πλήθος) για να σαρώσουμε όλα τα δυνατά μονοπάτια.

2.

Χάριν συντομίας, θα ονομάζουμε τους κύκλους που ενισχύουν την ενέργεια του παίκτη ως θετικούς κύκλους. Έστω u κορυφή του γραφήματος G που ανήκει σε κάποιο θετικό κύκλο (εφόσον υπάρχει τέτοιος στο γράφημα). Τότε, αρκεί να δείξουμε ότι υπάρχει $r-ασφαλής$ διαδρομή $s-u$, δηλαδή ο παίκτης μπορεί να διασχίσει «ζωντανός» το δωμάτιο u ξεκινώντας από το s, και ότι επίσης υπάρχει διαδρομή $u-t$. Αυτό πρακτικά σημαίνει ότι ο παίκτης μπορεί να διασχίσει οσοδήποτε φορές το θετικό κύκλο, δηλαδή να αυξήσει απεριόριστα πολύ την ενέργειά του, ώστε να μην μπορεί κανένας συνδυασμός τεράτων να του αφαιρέσει τη ζωή.

Επαναλαμβάνουμε τον αλγόριθμο του ερωτήματος 1 και ελέγχουμε την τιμή $d[t]$. Αν $d[t]>0$, τότε υπάρχει $r-ασφαλής$ διαδρομή $s-t$ και ανακοινώνουμε αποτυχία. Διαφορετικά, επαναλαμβάνουμε τον ίδιο αλγόριθμο στο γράφημα με τον ανανεωμένο πίνακα d, εφαρμόζοντας τις ίδιες συνθήκες χαλάρωσης. Σκοπός της δεύτερης επανάληψης του αλγορίθμου είναι να εντοπίσουμε αν υπάρχουν θετικοί κύκλοι. Σημειώνουμε ότι η ύπαρξη θετικών κύκλων εξασφαλίζει ότι κάθε επιπλέον διάσχισή τους αυξάνει την εναπομείνουσα ενέργεια. Συνεπώς, για να εντοπίσουμε ένα θετικό κύκλο αρκεί να ελέγξουμε αν τροποποιείται ο πίνακας d. Πράγματι, έστω ότι ενημερώνεται η τιμή $d[u]$, τότε η κορυφή u ανήκει σε θετικό κύκλο και η διαδρομή $s-u$ είναι προφανώς $r-ασφαλής$. Στη συνέχεια, αρκεί να ελέγξουμε αν η κορυφή t είναι προσβάσιμη μέσω της u το οποίο γίνεται με μια απλή DFS χρησιμοποιώντας ως αρχικό κόμβο τον u. Έτσι, η συνολική πολυπλοκότητα του αλγορίθμου μας είναι $O(|V||E|)$ για κάθε εφαρμογή του αλγορίθμου του ερωτήματος 1 και $O(|V|+|E|)$ για την DFS, δηλαδή συνολικά $O(|V||E|)$.

3.

Τα ερωτήματα 1 και 2 αφορούν την επίλυση προβλημάτων απόφασης. Με δεδομένο τον αλγόριθμο του ερωτήματος 2, που λύνει ένα γενικότερο πρόβλημα απόφασης, και παρατηρώντας ότι το πρόβλημα έχει κάποιου είδους μονοτονία ως προς το r, μπορούμε να λύσουμε το εν λόγω πρόβλημα βελτιστοποίησης εφαρμόζοντας $binary\ search$ στο r και λύνοντας το πρόβλημα απόφασης για αυτή την τιμή του r. Για να εφαρμόσουμε $binary\ search$ στο r, πρέπει να θέσουμε ένα άνω φράγμα στις τιμές που λαμβάνει το r, οπότε μπορούμε να πούμε ότι η μέγιστη τιμή του r ισούται με το άθροισμα των ενεργειών που αφαιρούνται από όλα τα δωμάτια τεράτων στο Λαβύρινθο (αυτό αντιστοιχεί στο σενάριο όπου όλες οι διαδρομές από την s στην t περιλαμβάνουν μόνο δωμάτια τεράτων ή άδεια δωμάτια). Έστω R αυτός ο αριθμός, τότε το R αρχικοποιείται στο 0 και υπολογίζεται εύκολα με μια απλή DFS με αρχικό κόμβο τον s, όπου σε κάθε δωμάτιο τέρατος το R αυξάνεται κατά την ενέργεια που αφαιρείται από τον παίκτη όταν διαβεί αυτό το δωμάτιο. Η πολυπλοκότητα για αυτή τη διαδικασία είναι $O(|V| + |E|)$. Προφανώς, ένα κάτω φράγμα για το r είναι το 0 (υποθέτουμε την καλύτερη περίπτωση, όπου δεν υπάρχει κανένα δωμάτιο τέρατος). Έτσι, το r λαμβάνει τις ακέραιες τιμές στο διάστημα $[0, R]$, $R > 0$. Εφαρμόζουμε $binary\ search$ στο r και λύνουμε το αντίστοιχο πρόβλημα απόφασης. Αν η διαδρομή $s - t$ είναι $r - ασφαλής$, τότε μας ενδιαφέρει να ελέγξουμε μικρότερες τιμές του r (για να δούμε αν μπορούμε να πετύχουμε κάτι καλύτερο). Διαφορετικά, εξετάζουμε μεγαλύτερες τιμές του r κοκ. Το πρόβλημα απόφασης επιλύεται $O\big(\log(R + 1) = O(\log R)\big)$ φορές, άρα η συνολική πολυπλοκότητα του αλγορίθμου είναι $O(|V| + |E| + |V||E| \log R) = O(|V||E| \log R)$.

Άσκηση 3

1.

Ονομάζουμε $s = v_1$ και $t = v_n$, έτσι ώστε όλες οι κορυφές να ακολουθούν ομοιόμορφη αρίθμηση.

Για την περίπτωση όπου το γράφημα G συνιστά ένα (σταθερό) μονοπάτι, ο αλγόριθμος που θα ακολουθήσουμε είναι άπληστος. Η ιδέα πίσω από την άπληστη λογική είναι η εξής:

Έστω $v_{i_1}, v_{i_2}, \dots, v_{i_k}$, με $i_1 < i_2 < \cdots < i_k$ και $i_1, i_2, \dots, i_k \in \{1,2, \dots, n\}$, οι σταθμοί στους οποίους ανεφοδιαζόμαστε. Τότε, για κάθε $j < i_k$ κάνουμε ένα από τα ακόλουθα:

(i) Αν $c(v_j) < c(v_{j+1})$ γεμίζουμε το ντεπόζιτο.

(ii) Αν $c(v_j) \geq c(v_{j+1})$ γεμίζουμε τόση βενζίνη όση χρειάζεται για να φτάσουμε στην πόλη v_{j+1} (προφανώς, φτάνουμε στη v_{j+1} με άδειο ντεπόζιτο).

Τέλος, στην πόλη v_{i_k}, που αποτελεί και το τελευταίο σταθμό ανεφοδιασμού, γεμίζουμε όσο χρειάζεται για να φτάσουμε στην πόλη v_n με άδειο ντεπόζιτο, όπως υπαγορεύει η άσκηση.

Η παραπάνω πρόταση μας δίνει ένα άπληστο κριτήριο για να επιλέγουμε την ποσότητα βενζίνης που θα γεμίζουμε σε κάθε σταθμό. Μένει, λοιπόν, να προσδιορίσουμε την ακολουθία σταθμών $v_{i_1}, v_{i_2}, \dots, v_{i_k}$ όπου θα γίνεται ο ανεφοδιασμός.

Για να γίνει κατανοητή η συνέχεια του αλγορίθμου μας, εισάγουμε πρώτα κάποιους βοηθητικούς συμβολισμούς.

• Για όλες τις κορυφές v_i, v_j, όπου $i < j$, που ανήκουν στο μονοπάτι $(v_1, \dots, v_n)$, μπορούμε να ορίσουμε την απόστασή τους d_{ij} ως εξής: $d_{ij} = \sum_{k=i}^{j-1} b\big((v_k, v_{k+1})\big)$. Επίσης, $d_{ii} = 0$, για κάθε $i = 1, \dots, n$. Με αυτό τον τρόπο, ορίζουμε αποστάσεις μεταξύ πόλεων που δε συνδέονται απαραίτητα με απευθείας ακμή στο μονοπάτι $s - t$.

• Θεωρούμε τη συνάρτηση $prev(i) = \min_{j \leq i}\{c(v_j) : d_{ji} \leq B\}$ η οποία επιστρέφει την πόλη v_j που έχει τη μικρότερη τιμή βενζίνης από όλες τις

πόλεις που βρίσκονται στο μονοπάτι $v_1 - v_i$ και μέσω της οποίας η πόλη v_i είναι προσβάσιμη με το πολύ ένα γέμισμα στη v_j (αυτό εξασφαλίζεται από τη συνθήκη $d_{ji} \leq B$).

• Θεωρούμε τη συνάρτηση $next(i) = \min_{j>i}\{c(v_j): d_{ij} \leq B\}$ η οποία επιστρέφει την πόλη που έχει τη μικρότερη τιμή βενζίνης από όλες τις πόλεις που βρίσκονται στο μονοπάτι $v_{i+1} - v_n$ και η οποία είναι προσβάσιμη με το πολύ ένα γέμισμα στην πόλη v_i.

• Στην ειδική περίπτωση όπου $prev(i) = i$, θα ονομάζουμε την πόλη i ως «σημείο αναφοράς».

Ο καθορισμός των «σημείων αναφοράς» είναι πολύ σημαντικός για την επίλυση του προβλήματος, επειδή κάθε «σημείο αναφοράς» στην ουσία αποτελεί ένα τοπικό ελάχιστο της τιμής της βενζίνης, καθώς σαρώνουμε το μονοπάτι $s - t$ από αριστερά προς τα δεξιά. Έτσι, η λογική είναι να προσδιορίσουμε όλα τα «σημεία αναφοράς» και να μετακινούμαστε από το ένα σημείο στο άλλο χρησιμοποιώντας βοηθητικά, αν χρειαστεί, τις πόλεις που επιστρέφει η συνάρτηση $next()$. Η μετακίνηση από το ένα «σημείο αναφοράς» στο άλλο φαίνεται να οδηγεί αλγοριθμικά στη σωστή κατεύθυνση, διότι αν v_i, v_j «σημεία αναφοράς» με $i < j$, τότε λόγω του ορισμού της συνάρτησης $prev()$ ισχύει ότι $c_j \leq c_i$, δηλαδή οδηγούμαστε σε πόλεις με όλο και μικρότερες τιμές βενζίνης.

Για να μεταβούμε από το «σημείο αναφοράς» i στο διαδοχικό «σημείο αναφοράς» j ελέγχουμε την απόσταση d_{ij}. Αν $d_{ij} \leq B$, δηλαδή αν η μετακίνηση μπορεί να συμβεί με ένα μόνο γέμισμα, τότε γεμίζουμε όσο χρειάζεται στην πόλη i και μεταβαίνουμε απευθείας στην πόλη j με άδειο ντεπόζιτο. Όμως, αν $d_{ij} > B$, τότε γεμίζουμε B λίτρα βενζίνης και μεταβαίνουμε στην πόλη $next(i)$. Επαναλαμβάνουμε την ίδια διαδικασία θέτοντας $i \leftarrow next(i)$ και $j \leftarrow j$.

Η παραπάνω διαδικασία δικαιολογεί ότι υπάρχει βέλτιστη λύση στην οποία μπορούμε να φτάσουμε με άδειο ντεπόζιτο σε κάθε «σημείο αναφοράς».

Υποθέτοντας το χειρότερο σενάριο όπου πρέπει να γεμίσουμε σε όλους τους σταθμούς, άρα η παραπάνω υπορουτίνα καλείται n φορές, ο απαιτούμενος χρόνος είναι $O(n)$. Για να υπολογίσουμε τη συνολική

υπολογιστική πολυπλοκότητα του αλγορίθμου, μένει να εκτιμήσουμε τους χρόνους υπολογισμού των $prev(i)$ και $next(i)$.

Προς το σκοπό αυτό, θεωρούμε ουρά προτεραιότητας στην οποία εισάγουμε πόλεις v_i με προτεραιότητα $c_i = c(v_i)$. Συγκεκριμένα, ξεκινάμε με την πόλη v_1 (φάση 1) και αποθηκεύουμε στην ουρά όλες τις πόλεις v_i για τις οποίες ισχύει ότι $d_{1i} \leq B$. Την πρώτη φορά που εισάγεται η πόλη v_i στην ουρά, αν καλέσουμε τη λειτουργία $\min()$, αυτή θα μας επιστρέψει την πόλη $prev(i)$. Αυτό ισχύει, διότι την πρώτη φορά που η πόλη v_i εισέρχεται στην ουρά, αυτό σημαίνει ότι υπάρχει πόλη v_j με $j \leq i$, έτσι ώστε $d_{ji} \leq B$ και για κάθε $k < j$ να ισχύει $d_{ki} > B$ (ειδάλλως δε θα ήταν η πρώτη φορά όπου η πόλη v_i εισάγεται στην ουρά). Δηλαδή, η v_j είναι η πιο απομακρυσμένη πόλη από τη v_i για την οποία ισχύει ότι $d_{ji} \leq B$, άρα και για κάθε $j < l < i$ (αν υπάρχει) θα ισχύει $d_{li} \leq B$. Μετά το τέλος της φάσης i, διαγράφουμε την πόλη v_i. Άρα, μόλις διαγράψουμε την πόλη v_i, αν καλέσουμε τη λειτουργία $\min()$, αυτή θα μας επιστρέψει την πόλη $next(i)$. Παρατηρούμε ότι συμβαίνουν n εισαγωγές στην ουρά και n διαγραφές, όπου κάθε λειτουργία απαιτεί $O(\log n)$. Συνεπώς, ο υπολογισμός των $prev(i)$ και $next(i)$ για κάθε $i = 1, \dots, n$ απαιτεί $O(n \log n)$. Τελικά, η συνολική πολυπλοκότητα του αλγορίθμου είναι $O(n + n \log n) = O(n \log n)$.

Σχόλιο Αν θέλουμε να επιστρέφεται και το ελάχιστο κόστος ανεφοδιασμού, μπορούμε να εισάγουμε μια μεταβλητή $cost$ η οποία αρχικοποιείται στο 0 και ενημερώνεται κάθε φορά που γεμίζουμε βενζίνη. Δηλαδή, αν βάλουμε g_i λίτρα βενζίνης στην πόλη v_i, $i = 1, \dots, n-1$, τότε η μεταβλητή $cost$ αυξάνεται κατά $g_i \cdot c(v_i)$, δηλαδή γίνεται η ανάθεση $cost \leftarrow cost + g_i \cdot c(v_i)$.

2.

Για την περίπτωση όπου το G μπορεί να είναι οποιοδήποτε κατευθυνόμενο γράφημα, ο αλγόριθμός μας θα ακολουθεί τη λογική του δυναμικού προγραμματισμού (DP).

Αρχικά, συμβολίζουμε με $D(v_i, g)$ το ελάχιστο κόστος για να μεταβούμε από την πόλη v_i στην πόλη t ξεκινώντας με g λίτρα βενζίνης. Τότε, απομονώνουμε όλες τις πόλεις v_j που είναι προσβάσιμες από την v_i με ένα το πολύ γέμισμα (δηλαδή $d_{ij} \leq B$) και για καθεμία από αυτές συγκρίνουμε την τιμή βενζίνης $c(v_j)$ με την τιμή βενζίνης $c(v_i)$.

• Αν $c(v_i) < c(v_j)$, τότε η καλύτερη στρατηγική είναι να γεμίσουμε B λίτρα στην πόλη v_i , δηλαδή να βάλουμε $B - g$ λίτρα βενζίνης, και να φτάσουμε στην πόλη v_j με $B - d_{ij}$ λίτρα στο ντεπόζιτο. Έτσι, για τον ανεφοδιασμό δαπανούμε $(B - g) \cdot c(v_i)$ ευρώ

• Αν $c(v_i) \geq c(v_j)$, τότε η καλύτερη στρατηγική είναι να γεμίσουμε όσο χρειάζεται στην πόλη v_i, δηλαδή να βάλουμε $d_{ij} - g$ λίτρα βενζίνης, και να φτάσουμε στην πόλη v_j με άδειο ντεπόζιτο. Έτσι, για τον ανεφοδιασμό δαπανούμε $(d_{ij} - g) \cdot c(v_i)$ ευρώ.

Επαναλαμβάνουμε την παραπάνω διαδικασία για όλες τις προσβάσιμες πόλεις v_j μέσω της v_i και επιλέγουμε τη στρατηγική με το ελάχιστο κόστος. Τυπικά, λοιπόν, η αναδρομική σχέση που περιγράφει το πρόβλημα είναι:

$$D(v_i, g) = \min_{\substack{j \ s.t. \\ d_{ij} \leq B}} \begin{Bmatrix} D(v_j, B - d_{ij}) + (B - g) \cdot c(v_i), & c(v_i) < c(v_j) \\ D(v_j, 0) + (d_{ij} - g) \cdot c(v_i), & c(v_i) \geq c(v_j) \ \kappa\alpha\iota \ d_{ij} \geq g \end{Bmatrix}$$

Προφανώς ισχύει $D(t, g) = 0$, για κάθε πιθανό g.

Με βάση την παραπάνω αναδρομική σχέση, κατασκευάζουμε ένα γράφημα H το οποίο έχει ως κορυφές κάθε ζεύγος της μορφής (v_i, g), όπου v_i κορυφή του G και g η ποσότητα βενζίνης με την οποία φτάνουμε στην πόλη v_i. Οι κορυφές (v_i, g_i) και (v_j, g_j) συνδένται μέσω ακμής e, με φορά από την πρώτη στη δεύτερη, όπου $e = (B - g) \cdot c(v_i)$ ή $e = (d_{ij} - g) \cdot c(v_i)$ ανάλογα με την περίπτωση στην οποία εμπίπτει η αναδρομική σχέση. Κάθε ακμή εκφράζει έναν τρόπο να μεταβούμε από

την πόλη v_i φτάνοντας σε αυτή με g_i λίτρα βενζίνης στην πόλη v_j διαθέτοντας g_j λίτρα βενζίνης, όπου $g_j = B - d_{ij}$ ή $g_j = 0$, πάλι ανάλογα με την περίπτωση στην οποία εμπίπτει η αναδρομική σχέση.

Έτσι, η λύση του προβλήματος δίνεται από το ελάχιστο κόστος μονοπατιού που επιστρέφει ο αλγόριθμος του $Dijkstra$ στο γράφημα H, οπότε και η πολυπλοκότητα του αλγορίθμου μας θα ταυτίζεται με την πολυπλοκότητα του $Dijkstra$. Για να την εκτιμήσουμε, απομένει να υπολογίσουμε το πλήθος των κορυφών και ακμών που εμφανίζονται στο γράφημα H. Επί της ουσίας, το πλήθος των κορυφών δίνεται από το μέγεθος του διδιάστατου πίνακα D. Παρατηρούμε καταρχάς ότι αυτός ο πίνακας θα έχει το πολύ n^2 μη κενές θέσεις και αυτό ισχύει, διότι έχουμε n πόλεις και σε κάθε πόλη μπορούμε να φτάσουμε με n διαφορετικές τιμές βενζίνης. Άρα, έχουμε το πολύ n^2 κορυφές και με δεδομένο ότι από κάθε πόλη (που αναπαρίσταται μερικώς από μια κορυφή) είναι προσβάσιμες n άλλες πόλεις θα έχουμε και το πολύ $n \cdot n^2 = n^3$ ακμές. Τελικά, η πολυπλοκότητα του αλγορίθμου μας είναι $O(n^3 + n^2 \log(n^2)) = O(n^3 + n^2 \log n) = O(n^3)$, διότι $\log n < n$.

Άσκηση 4

Αρχικά, σημειώνουμε ότι για να καμφθεί η αντίσταση μιας ομάδας εξτρεμιστών που βρίσκεται στη θέση e_i, όπου $i = 1, \dots, m$, αρκεί οι δυνάμεις στρατού είτε να αιχμαλωτίσουν άμεσα τους εξτρεμιστές είτε να καταστρέψουν όλες τις βάσεις που τους συντηρούν, δηλαδή όλες τις βάσεις σε απόσταση μικρότερη ή ίση του d από τη θέση της ομάδας των εξτρεμιστών. Ανεξάρτητα από τη μέθοδο επίθεσης που υιοθετείται κάθε φορά, η βέλτιστη στρατηγική, δηλαδή αυτή που εξασφαλίζει το ελάχιστο κόστος για την εξαπολυόμενη επίθεση, είναι να επιλέγουμε κάθε φορά την ομάδα στρατιωτών που βρίσκεται πλησιέστερα στην αντίστοιχη ομάδα εξτρεμιστών ή στην αντίστοιχη βάση.

Από τα παραπάνω προκύπτει πλέον το εξής (σημαντικό) ερώτημα:

«Για να καμφθεί η αντίσταση μιας ομάδας εξτρεμιστών συμφέρει ο στρατός να αιχμαλωτίσει άμεσα τη συγκεκριμένη ομάδα ή να καταστρέψει όλες τις βάσεις που τη συντηρούν;»

Η απάντηση δίνεται παρακάτω:

Έστω $c_{capture} = c_1$ το κόστος της άμεσης αιχμαλωσίας και $c_{destruction} = c_2$ το κόστος της καταστροφής όλων των βάσεων για τη συγκεκριμένη ομάδα εξτρεμιστών. Προφανώς, όπως ειπώθηκε παραπάνω, το c_1 ισούται με την ελάχιστη απόσταση κάποιου διαθέσιμου στρατιώτη από τη θέση της ομάδας εξτρεμιστών και το c_2 ισούται με το άθροισμα των ελάχιστων αποστάσεων των διαθέσιμων στρατιωτών από κάθε βάση που μας ενδιαφέρει. Τότε, διακρίνουμε τις ακόλουθες περιπτώσεις:

• Αν $c_2 \leq c_1$, τότε επιλέγεται η επίθεση υπό μορφή καταστροφής όλων των βάσεων που συντηρούν την εν λόγω ομάδα εξτρεμιστών και το ελάχιστο κόστος επίθεσης ισούται με c_2. Πράγματι, αυτό είναι αναμενόμενο και επιθυμητό, διότι η καταστροφή βάσεων ενδέχεται να συνεισφέρει και στην κάμψη της αντίστασης μιας άλλης ομάδας εξτρεμιστών που ανεφοδιάζονται από τις ίδιες βάσεις.

• Αν $c_2 > c_1$, τότε είναι λογικό να επιλέγεται η επίθεση υπό μορφή άμεσης αιχμαλωσίας της εν λόγω ομάδας εξτρεμιστών και το ελάχιστο κόστος επίθεσης ισούται με c_1. Ωστόσο, αν σκεφτούμε λίγο γενικότερα,

ενδέχεται να παρατηρήσουμε ότι ακόμα κι αν ισχύει $c_2 > c_1$, η καταστροφή όλων των βάσεων που συντηρεί την εν λόγω ομάδα εξτρεμιστών ίσως εξασφαλίζει ότι κάμπτεται η αντίσταση και τουλάχιστον μιας άλλης ομάδας εξτρεμιστών με μικρότερο συνολικό κόστος, οπότε σε αυτή την περίπτωση συμφέρει να επιλέξουμε την επίθεση υπό μορφή καταστροφής όλων των βάσεων που συντηρούν την εν λόγω ομάδα εξτρεμιστών με ελάχιστο κόστος c_2.

Από τα παραπάνω καταλήγουμε στο συμπέρασμα ότι για την κάμψη της αντίστασης μιας ομάδας εξτρεμιστών θα χρειαστεί κόστος το πολύ ίσο με με το κόστος άμεσης αιχμαλώτισης. Έτσι, μπορούμε να φανταστούμε ένα δίκτυο $G(V, E, b)$ με $m + k + 2$ κορυφές (k κορυφές: μία για κάθε βάση ανεφοδιασμού, m κορυφές: μία για κάθε ομάδα εξτρεμιστών και καθεμιά από τις δύο τελευταίες κορυφές s, t αντιστοιχεί σε μια ομαδοποίηση όλων των δυνάμεων στρατού). Ορίζουμε (κατευθυνόμενες) ακμές από την κορυφή s σε κάθε βάση ανεφοδιασμού (*χωρητικότητας* ίση με το ελάχιστο κόστος καταστροφής αυτής της βάσης), από κάθε ομάδα εξτρεμιστών προς την κορυφή t (*χωρητικότητας* ίση με το ελάχιστο κόστος άμεσης αιχμαλωσίας της εν λόγω ομάδας). Τέλος, για να περιγράψουμε τη σχέση ανεφοδιασμού μιας ομάδας εξτρεμιστών από μια βάση, ορίζουμε ακμές που ξεκινούν από τη βάση και καταλήγουν στην ομάδα εξτρεμιστών (*χωρητικότητας* ίση με τη χωρητικότητα της ακμής που προσπίπτει στην αντίστοιχη βάση). Συνεπώς, είναι φανερό ότι στο δίκτυο G που κατασκευάσαμε, η εξόντωση μιας ομάδας εξτρεμιστών ισοδυναμεί είτε με κορεσμό της ακμής που συνδέει την ομάδα εξτρεμιστών με την κορυφή t είτε με κορεσμό όλων των ακμών που καταλήγουν στην εν λόγω ομάδα εξτρεμιστών (και εκ κατασκευής προέρχονται από βάσεις ανεφοδιασμού). Αυτό σημαίνει ότι ψάχνουμε το φθηνότερο συνδυασμό ακμών από ομάδες εξτρεμιστών προς την t και από την s προς βάσεις ανεφοδιασμού, ώστε να καλυφθούν οι ακμές από βάσεις ανεφοδιασμού προς ομάδες εξτρεμιστών. Η παραπάνω δήλωση δικαιολογεί τον ισχυρισμό ότι το πρόβλημα μπορεί να αναχθεί σε πρόβλημα εύρεσης ελάχιστης τομής ($min\ cut$) το οποίο έχει την ίδια λύση με το πρόβλημα μέγιστης ροής ($max\ flow$). Το τελευταίο επιλύεται π.χ. με τον αλγόριθμο $Ford - Fulkerson$ εφαρμόζοντας σε αυτόν βελτιώσεις $Edmonds - Karp$. Όσον αφορά την υπολογιστική πολυπλοκότητα του αλγορίθμου μας, θα ισούται με το χρόνο κατασκευής

του δικτύου G, το οποίο απαιτεί γραμμικό χρόνο, εφόσον πρώτα ταξινομήσουμε τις αποστάσεις των βάσεων διαχωρισμού, των ομάδων εξτρεμιστών και των δυνάμεων στρατού (π.χ. με $mergesort$) σε χρόνο $O\big((m+n+k)\log(m+n+k)\big)$. Άρα, συνολικά για την πολυπλοκότητα έχουμε $O\big((m+k+2)+(m+n+k)\log(m+n+k)\big) = O\big((m+n+k)\log(m+n+k)\big)$.

Άσκηση 5

• Τακτοποίηση ορθογωνίων παραλληλογράμμων

Καταρχάς, το πρόβλημα ανήκει στην κλάση NP, διότι αν μας δοθεί μια υπόδειξη της λύσης, δηλαδή μια διάταξη των ορθογωνίων $A_1, \dots, A_n$ στο επίπεδο και οι ακριβείς διαστάσεις του ορθογωνίου B, μπορούμε με ένα γραμμικό πέρασμα όλων των ορθογωνίων να ελέγξουμε αν αυτά ανήκουν στο B και δεν επικαλύπτονται μεταξύ μας. Πιο τυπικά, αν μας δοθεί ένα πιστοποιητικό του προβλήματος (το οποίο εγγυημένα θα έχει πολυωνυμικό μέγεθος, επειδή τα $x_1, \dots, x_n$ είναι πολυωνυμικά συσχετιζόμενα με το n), μπορούμε να επαληθεύσουμε την εγκυρότητά του σε πολυωνυμικό χρόνο.

Αν φανταστούμε το πρόβλημα λίγο πιο γεωμετρικά, στην ουσία αναρωτιόμαστε αν μπορούμε να διαμερίσουμε ένα ορθογώνιο B ορισμένου εμβαδού σε n ορθογώνια A_i μοναδιαίου μήκους (ξένα μεταξύ τους). Έτσι, είναι λογικό να δείξουμε ότι το εν λόγω πρόβλημα είναι $NP - complete$ ανάγοντας σε αυτό το πρόβλημα $Partition$, το οποίο ξέρουμε επίσης από θεωρία ότι είναι $NP - complete$.

Θεωρούμε το σύνολο $X = \{x_1, \dots, x_n\}$ το οποίο αποτελεί $instance$ του $Partition$. Κάθε x_i, $i = 1, \dots, n$, αναπαριστά το πλάτος του ορθογωνίου A_i και με δεδομένο ότι όλα τα A_i έχουν μοναδιαίο μήκος, μπορούμε να ισχυριστούμε ότι το x_i επί της ουσίας εκφράζει το εμβαδόν του ορθογωνίου A_i. Υποθέτουμε, στη συνέχεια, ότι το ορθογώνιο B έχει διαστάσεις $2 \times b$, δηλαδή έχει διπλάσιο μήκος από κάθε A_i. Αναζητούμε κατάλληλη τιμή του b. Επιλέγουμε $b = \frac{1}{2}\sum_{i=1}^{n} x_i$, τότε αν τα n ορθογώνια μπορούν να τοποθετηθούν στο B χωρίς επικαλύψεις, αυτό θα σημαίνει ότι λόγω του πλάτους του B, υπάρχει $A \subset X$ έτσι ώστε $sum(A) = b = sum(X \setminus A)$, δηλαδή λύνεται το $Partition$.

• Μέγιστη τομή με βάρη στις κορυφές

Καταρχάς, το πρόβλημα ανήκει στην κλάση NP, διότι αν μας δοθεί μια υπόδειξη της λύσης, δηλαδή μια τομή $(S, V \setminus S)$, μπορούμε με μια σάρωση του γραφήματος G να ελέγξουμε αν το συνολικό βάρος των ακμών που διασχίζουν την τομή είναι τουλάχιστον B. Πιο τυπικά, αν μας δοθεί ένα πιστοποιητικό του προβλήματος (το οποίο εγγυημένα θα έχει πολυωνυμικό φράγμα, λόγω του πληθάριθμου του συνόλου V), μπορούμε να επαληθεύσουμε την εγκυρότητά του σε πολυωνυμικό χρόνο (π.χ. με μια διάσχιση DFS).

Έστω $w(S, V \setminus S)$ το συνολικό βάρος των ακμών που διασχίζουν την τομή $(S, V \setminus S)$. Τότε, θα ισχύει:

$$w(S, V \setminus S) = \sum_{\substack{u \in S, \\ v \in V \setminus S}} w(u)w(v) = \sum_{u \in S} w(u) \cdot \sum_{v \in V \setminus S} w(v) \qquad (1)$$

Η τελευταία ισότητα ισχύει επειδή το γράφημα G είναι πλήρες, οπότε κάθε κορυφή συνδέεται με όλες τις άλλες.

Θέτουμε $w(X) = \sum_{u \in X} w(u)$, δηλαδή με $w(X)$ συμβολίζουμε το συνολικό βάρος των κορυφών που ανήκουν στο σύνολο X.

Έτσι, η σχέση (1) απλοποιείται ως εξής:

$$w(S, V \setminus S) = \sum_{u \in S} w(u) \cdot \left(\sum_{v \in V} w(v) - \sum_{v \in S} w(v) \right) = w(S) \cdot \big(w(V) - w(S) \big)$$

Αρκεί να μεγιστοποιήσουμε την παραπάνω ποσότητα.

Θεωρούμε τη συνάρτηση $f(x) = x(K - x)$, όπου $K = w(V)$ και $x = w(S)$ για την οποία ισχύει ότι $f(x) \leq K^2/4$. Το $K^2/4$ είναι ολικό μέγιστο της συνάρτησης και δίνεται για $x = w(S) = K/2 = w(V)/2$. Τότε, θα ισχύει ότι $w(V \setminus S) = w(V) - w(S) = w(V)/2$. Άρα, σκοπός είναι να διαμερίσουμε το σύνολο $\{w(u) : u \in V\}$ σε δύο ισοβαρή σύνολα. Αυτή η παρατήρηση μας παρακινεί να ανάγουμε το πρόβλημα $2 - Partition$ στο δικό μας. Συνεπώς, μπορούμε να ξεκινήσουμε με ένα $instance$ του $2 - Partition$ το οποίο πρακτικά είναι ένα σύνολο από θετικούς ακέραιους με πληθάριθμο όσο και το πλήθος των κορυφών του

διαστήματος και στη συνέχεια να αντιστοιχίσουμε κάθε ακέραιο στο βάρος μιας κορυφής του γραφήματος G.