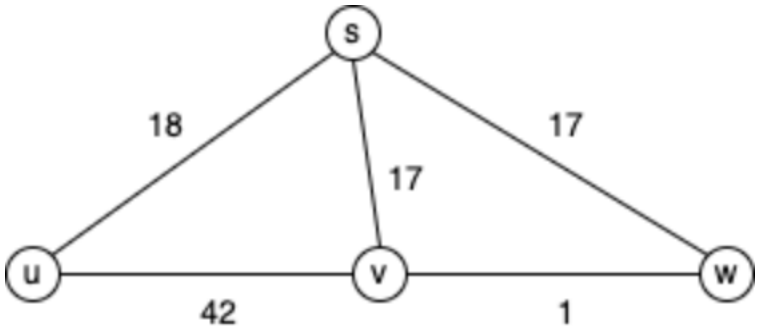
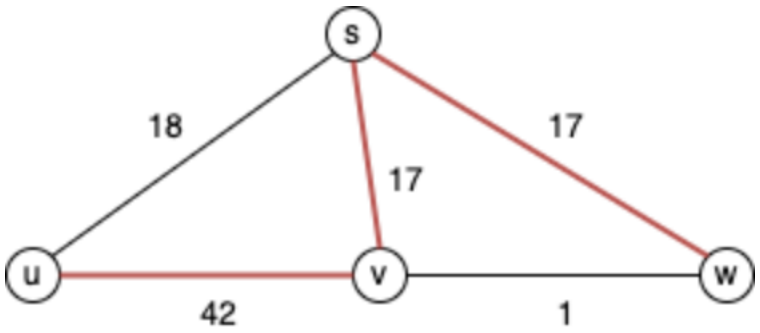
Αλγόριθμοι και Πολυπολοκότητα
3η σειρά Ασκήσεων
Βασίλης Βαρσαμής-el18033

Άσκηση 1

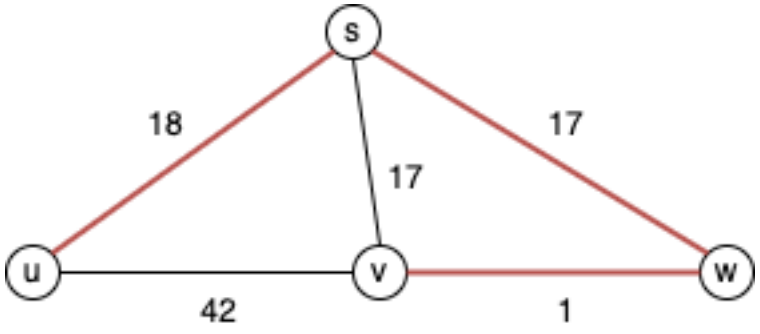
(α) Η άπληστη στρατηγική δεν είναι βέλτιστη και αυτό φαίνεται στο παρακάτω αντιπαράδειγμα:



Στην παραπάνω εικόνα η άπληστη στρατηγική για το $T^*(s,2)$ θα δώσει (λανθασμένα) το συνδετικό δέντρο που φαίνεται παρακάτω με βάρος 76 (οι ακμές με κόκκινο είναι αυτές που ανήκουν στο δέντρο)



Ωστόσο το σωστό συνδετικό δέντρο με βάρος 36 είναι αυτό που φαίνεται παρακάτω:



Το αντιπαράδειγμα βασίζεται στο ότι αν πάρουμε τις φθηνότερες ακμές που προσπίπτουν στην s, μπορεί να αναγκαστούμε στη συνέχεια για να μην κλείσουμε κύκλο να πάρουμε μια πολύ “βαριά” ακμή ή ισοδύναμα να μην πάρουμε μια πολύ “φθηνή” ακμή επειδή κλείνει κύκλο με τις ακμές που προσπίπτουν στο s και διαλέξαμε άπληστα.

(β) Η ιδέα στην οποία βασίζεται ο αλγόριθμος είναι η εξής:

Ο αλγόριθμος του Kruskal επιλέγει τις ακμές με βάση την αύξουσα διάταξη των βαρών τους.Έτσι κάποιες ακμές που προσπίπτουν στο s μπορεί να “χάσουν”, δηλαδή να επιλεγούν στη θέση τους φθηνότερες ακμές με αποτέλεσμα το T^* να έχει λιγότερες από k προσπίπτουσες ακμές στο s, ή να έχουμε πολλές φθηνές ακμές που “κερδίζουν” άλλες , με αποτέλεσμα το T^* να έχει παραπάνω από k προσπίπτουσες ακμές. Πρέπει λοιπόν να βρούμε έναν ακέραιο x που αν τον προσθέσουμε στα βάρη των ακμών που προσπίπτουν στην s, τότε ο αλγόριθμος του Kruskal θα πάρει ακριβώς k προσπίπτουσες στο s ακμές.Σημειώνουμε ότι το x μπορεί να είναι θετικός ή αρνητικός. Δηλαδή αν T' είναι το ΕΣΔ που προκύπτει αν προσθέσουμε στις προπίπτουσες στο s ακμές το x, τότε αν το x είναι το κατάλληλο $T' \equiv T^*(s,k)$.Με το να μετακινούμε στο πίνακα τις ακμές που προσπίπτουν στο s, είναι ισοδύναμο το να αλλάζουμε σε όλες τα βάρη κατάλληλα με σκοπό για επιλογή κατάλληλου x να “αναγκάσουμε τον Kruskal να επιλέξει αυτές και καμία άλλη. Όσο περισσότερες ακμές θέλουμε τόσο μικρότερο πρέπει να ναι το x (μπορεί και αρνητικό) και όσο λιγότερες ακμές θέλουμε τόσο πιο μεγάλο θα γίνεται το x.Μπορούμε να επιτύχουμε έτσι οποιοδήποτε k.Τα k που έχει νόημα να εξετάσουμε είναι αυτά για τα οποία όταν το προσθέσουμε σε μία ακμή από τις προσπίπτουσες στο s, γίνεται ίσο το βάρος με μία άλλη στο γράφο.Αυτό συμβαίνει διότι για να “χάσει” μια ακμή πρέπει να αντικατασταθεί κάποια άλλη, οπότε για να διατηρηθεί το κόστος πρέπει το $e_s + k = e_g$, δηλαδή $k = e_g - e_s$ Για τον εντοπισμό του κατάλληλου x θα εκτελέσουμε binary search. Ο αλγόριθμος είναι ο εξής:
Ορίζουμε το σύνολο $K \in \{e_g - e_s : e_g \in G \wedge e_s \in E_s\}$ που αποτελείται από αυτές τις διαφορές.

1) Ταξινομούμε το K σε $O(|E||A|log(|E||A|))$
2) Binary Search στο K για εύρεση x σε $O(2log(|E|)$

3) Kruskal σε$O(|E|log|E|)$, έχοντας αυξήσει τα βάρη κατά x
4) Αν έχω k μικρότερο μειώνω το x, αλλιώς το αυξάνω.

Συνολικά: $O(|E||A|log(|E|)+|E|log^2|E|)$

Άσκηση 2

1. Ο αλγόριθμος είναι μια τροποποίηση του αλγόριθμου Bellman-Ford ο οποίος τώρα θα βρίσκει τα μακρύτερα μονοπάτια από τον s σε όλους τους υπόλοιπους κόμβους με την προυπόθεση ότι κάθε υπο-μονοπάτι θα έχει θετικό μήκος. Ψάχνουμε το μακρύτερο μονοπάτι γιατί αυτός είναι ο "καλύτερος" τρόπος να φτάσουμε από έναν κόμβο σε έναν άλλο ή ισοδύναμα ο τρόπος που μαζεύει τους περισσότερους πόντους. Αν ο "καλύτερος" τρόπος για τον κόμβο t είναι αρνητικός σημαίνει ότι και όλοι άλλοι θα είναι επίσης, οπότε το s-t δεν θα είναι r-ασφαλές.Αντίθετα,αν είναι θετικός τότε βρήκαμε μια r-ασφαλή διαδρομή από το s στο t.

Οι τροποποιήσεις που κάνουμε στον Bellmann-Ford είναι η εξής:

1) Αρχικοποιούμε το $dist[s]=r$ και για κάθε άλλο κόμβο u κάνουμε $dist[u]=-\infty$
2) Ανανεώνουμε κάποιο label αν για την ακμή $(u,v)\in E$ έχουμε ότι $dist[u]+w(u,v)>dist[v]$ και επίσης $dist[u]+w(u,v)>0$ διότι δεν θέλουμε να μπορούμε να φτάσουμε σε κάποιο ενδιάμεσο κόμβο με αρνητική ζωή (ανεξάρτητα από το αν μετά θα μπορούσε να γίνει θετική).

Αφού δεν έχουμε κύκλους θετικού μήκους που θα οδηγούσε σε όλες και καλύτερα labels στους κόμβους που τον αποτελούν η πολυπλοκότητα του αλγορίθμου είναι αυτή του Bellman-Ford που είναι $O(|V||E|)$.

2. Η ιδέα εδώ είναι ότι αν κάποιος κύκλος θετικού μήκους είναι προσβάσιμος με θετικούς πόντους από τον s τότε φτάνοντας σε αυτόν μπορούμε να τον διασχίσουμε όσες φορές είναι απαραίτητο με σκοπό να αυξήσουμε τους

πόντους κατάλληλα ώστε να έχουμε αρκετούς να φτάσουμε μέχρι το t και έτσι η διαδρομή να είναι r-ασφαλής. Βρίσκουμε κύκλο θετικού μήκους ακριβώς όπως βρίσκουμε κύκλο αρνητικού μήκους στον κλασικό Bellmann-Ford: μετά το πέρας των $|V-1|$ επαναλήψεων εκτελούμε άλλη μία επανάληψη, έχουμε κύκλο θετικού μήκους αν κάποιο label αλλάξει σε αυτόν τον γύρο.Τον κύκλο τον εντοπίζουμε από τον πίνακα prev που κρατάει τους πατεράδες των κόμβων. Αν δεν μπορούμε να φτάσουμε τον κύκλο με θετικούς πόντους από το s, τότε κατά την τελευταία επανάληψη δεν θα αλλάξει κανένα label και έτσι τον αν φτάνουμε στο t θα εξαρτάται απο το $dist[t]$.

3. Έχοντας τους παραπάνω αλγόριθμους η προφανής λύση για το ερώτημα είναι να αναζητήσουμε γραμμικά το r μέχρι να φτάσουμε στο πρώτο r για το οποίο θα έχουμε r-ασφαλή διαδρομή. Η πολυπλοκότητα αυτής της ιδέας είναι $O(r|V||E|)$ που είναι ψευδοπολυωνυμική στο μέγεθος της εισόδου.

Θα δείξουμε ότι μπορούμε να βελτιώσουμε το παραπάνω σε $O(\log(r)|V||E|)$:

Εφαρμόζουμε την ιδέα των εκθετικών αλμάτων και σε κάθε εκτέλεση του Bellman-Ford θέτουμε r = επόμενη δύναμη του 2. Θα φτάσουμε σε ένα σημείο που για $r=2^i$ έχουμε βρει r-ασφαλή διαδρομή στον γράφο. Λόγω της μονοτονίας του προβλήματος, αυτό σημαίνει ότι για κάθε άλλο $r'>2^i$ υπάρχει επίσης r'-ασφαλής διαδρομή και ότι για κανένα $r''<2^{i-1}$ δεν υπάρχει, διότι αν υπήρχε θα είχαμε σταματήσει στο $(i-1)-οστο$ βήμα. Έτσι περιορίζουμε τα υποψήφια r στο διάστημα $[2^{i-1},2^i]$ και κάνουμε binary search εκεί η οποία πετυχαίνει ακριβώς λόγω της μονοτονίας που περιγράψαμε πρηγουμένως. Ο χρόνος και για τις δύο διεργασίες είναι $O(log(r))$ όποτε τελικά η πολυπλοκότητα είναι $O(\log(r)|V||E|)$.

Άσκηση 3

1. Αποδεικνύουμε πρώτα το παρακάτω Λήμμα:

Λήμμα 1

Έστω το μονοπάτι των κόμβων $s, u_1, u_2, \ldots, u_n, t$ τότε το παρακάτω άπληστο κριτήριο δίνει βέλιτστη λύση:

- $\forall i$ αν $c(u_i) < c(u_{i+1})$ τότε γέμισε όλο το ντεπόζιτο στο u_j
- $\forall j$ αν $c(u_i) \geq c(u_{i+1})$ τότε γέμισε όσο χρειάζεται για να πας από το u_j στο u_{j+1}

Απόδειξη

Στην πρώτη περίπτωση αν η βέλτιστη λύση στο u_j δεν γεμίσει το ντεπόζιτο, τότε μπορούμε να το γεμίσουμε στο u_j και να βάλουμε λιγότερο στο u_{j+1} χωρίς να αλλάζει το συνολικό κόστος.Αυτό μας θυμίζει το Fractional Knapsack: αφού σε κάθε σταθμό μπορούμε να βάλουμε οποιοδήποτε κλάσμα λίτρων βενζίνης του B με $f_i B$ με $f_i \in [0,1]$. Τότε για το κόστος που θα είναι $(f_i c_i + f_{i+1} c_{i+1})B = ((f_i + \varepsilon)c_i + (f_{i+1} - \frac{c_i}{c_{i+1}}\varepsilon)c_{i+1})B$

Στην πρώτη περίπτωση: αν για την βέλτιστη έχουμε $f_i < 1$ τότε για ε = $1 - f_i$ Έχουμε από την παραπάνω ισότητα ίδιο κόστος και $f_i' = 1$ και $f_{i+1}' = f_{i+1} - \frac{c_i}{c_{i+1}}(1 - f_i) < f_{i+1}$, δηλαδή αυξάνουμε το f_i και μειώνουμε το f_{i+1} κατάλληλα και παίρνουμε το ίδιο κόστος για την άπληστη επιλογή. Ομοίως αν $b_i < f_i$ παίρνουμε $\varepsilon = b_i - f_i < 0$ οπότε το f_i μειώνεται και το f_{i+1} αυξάνεται κατάλληλα.

Ως προς τον αλγόριθμο παρατηρούμε τα εξής:

Ορίζουμε $prev(i)$ ως το σταθμό $j \leq i$ στο μονοπάτι από το s στο i που έχει την φθηνότερη βενζίνη και για τον οποίο το άθροισμα των ακμών δεν υπερβαίνει την χωρητικότητα του ντεπόζιτου.Ομοίως ορίζουμε $next(i)$ ως το σταθμό $j > i$ που έχει την φθηνότερη βενζίνη και για τον οποίο το άθροισμα των ακμών του μονοπατιού έως εκεί δεν υπερβαίνει την χωρητικότητα του ντεπόζιτου.Οι ισοπαλίες λύνονται επιλέγοντας το κόμβο πιο κοντά στο

τέλος.Έστω τώρα ότι για κάποιο $i : prev(i) = i$ και j η τελευταία στάση για βενζίνη που κάναμε πριν το i. Τότε $c(i) \leq c(j)$ και επομένως μια βέλτιστη λύση θα προέλθει, λόγω του Λήμματος που αποδείξαμε, βάζοντας όσο ακριβώς χρειάζεται για να πάμε από το i στο j.Αν έχουμε τους πίνακες next και prev, τότε η οικονομικότερη διαδρομή για να πάμε από ένα σημείο για το οποίο $prev[i] = i$ στο άλλο μπορεί να βρεθεί με τον παρακάτω αλγόριθμο:

1) Αν απέχουμε λιγότερο από Β (σε βενζίνη) από το κοντινότερο σημείο για το οποίο $prev(i) = i$, τότε βάλε τόση ακριβώς βενζίνη ώστε να φθάσεις σε αυτό με άδειο ντεπόζιτο.
2) Αλλιώς γέμισε το ντεπόζιτο και πήγαινε στο $next(i)$, κάνε το $i, next(i)$

Ο παραπάνω αλγόριθμος είναι ένα κριτήριο για το πως πρέπει να πάμε από το ένα σημείο για το οποίο $prev[i] = i$ στο επόμενο.Αν έχουμε $prev[]$ και $next[]$ τότε κάνει $O(n)$ για να βρει διαδρομή ελαχίστου κόστους.Πως υπολογίζουμε όμως τους πίνακες αυτούς;

Μια προφανής λύση είναι να τους υπολογίσουμε σε χρόνο $O(n^2)$.

Μπορούμε να βελτιώσουμε αυτήν την πολυπλοκότητα ως εξής:

Στην ουσία για να βρούμε το σημείο για το οποίο $prev[i] = i$ αυτό που πρέπει να κάνουμε είναι να κοιτάμε σε ένα παράθυρο μήκους B, όπως ορίστηκε στο prev άλλωστε πρέπει ο σταθμός να απέχει το πολύ Β από το i.

Προχωράμε το παράθυρο ένα βήμα την φορά από το 1 έως το n και για κάθε σταθμό μέσα στο παράθυρο τον βάζουμε σε μια ουρά προτεραιότητας (η κεφαλή της ουράς είναι το στοιχείο στο παράθυρο με το ελάχιστο κόστος).Όταν λοιπόν βάλουμε την i στην ουρά, η κεφαλή αυτής είναι το $prev[i]$. Όταν περάσει το παράθυρο και αφαιρέσουμε την i τότε η κεφαλή της ουράς θα είναι το $next(i)$.Κάθε εισαγωγή και διαγραφή στην ουρά γίνεται σε $O(logn)$ οπότε συνολικά θα έχουμε πολυπλοκότητα $O(nlogn)$.Η ιδέα του παραπάνω αλγόριθμου είναι ότι αν κάποια στιγμή βρούμε δύο σημεία i, j για τα οποία $prev[i] = i$ και $prev[j] = j$ και για τα οποία ανάμεσα τους δεν υπάρχουν άλλα τέτοια σημεία (χβτγ) που απέχουν παραπάνω από Β σε βενζίνη, τότε πρέπει να βρω τον σταθμό με την φθηνότερη βενζίνη μετά το i σε απόσταση το πολύ Β δηλαδή το $next[i]$, για

το οποίο ξέρω ότι $c[i] \leq c[next[i]]$ και επομένως λόγω του λήμματος μια βέλτιστη στρατηγική είναι να γεμίσω το ντεπόζιτο και να πάω στην $next$.

2.

Έστω $D(u, b)$, το ελάχιστο κόστος για να πάμε από το u στο t αρχίζοντας με b μονάδες βενζίνης.Λόγω του Λήμματος 1, σε κάθε κορυφή που εξετάζουμε,είτε θα γεμίσουμε το ντεπόζιτο είτε θα βάλουμε τόσο ώστε να πάμε στο επόμενο με άδειο.

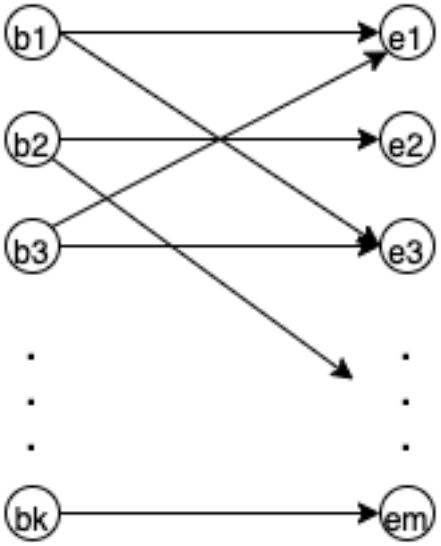
$$D(u, b) = \min_{v:b_{uv} \leq B} \{D(v,0) + (b_{uv} - b)c(u), D(v, B - b_{uv}) + (B - b)c(u)\}$$

Ζητάμε το $D(s,0)$. Λόγω του Λήμματος 1 το b που είναι η υπολοιπόμενη βενζίνη σε κάθε σταθμό, δεν θα πάρει όλες τις τιμές από 0 έως Β.Για κάθε κόμβο έχουμε $O(n)$ (μέγιστο μονοπάτι έχει n-1 ακμές). Επομένως το state έχει $O(n^2)$ καταστάσεις και κάθε κατάσταση για να υπολογιστεί χρειάζεται $O(n)$ λόγω του min. Επομένως έχουμε πολυπλοκότητα $O(n^3)$.Σημειώνουμε ότι με την παραπάνω αναδρομική σχέση ανάγουμε το πρόβλημα σε πρόβλημα υπολογισμού συντομότερων διαδρομών στο οποίο οι κορυφές είναι τα (u,b) , οι ακμές είναι $((u, b), (v,0))$ και $((u, b), (v, B - b_{uv}))$ και τα βάρη είναι τα υπόλοιπα.Στον χώρο αυτό ο αλγόριθμος του Dijkstra θα δώσει πολυπλοκότητα $O(n^3)$.

Άσκηση 4

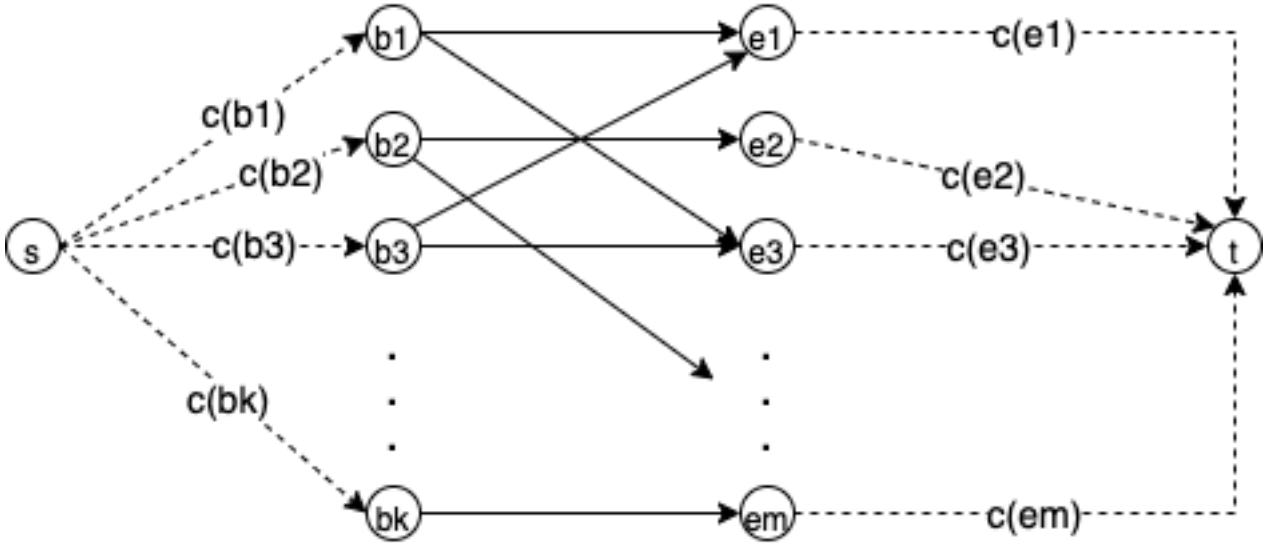
Για το πρόβλημα παρατηρούμε ότι κάθε φορά σε κάποια βάση εξτρεμιστών b_i ή σε κάποια δύναμη εξτρεμιστών e_i θα επιτίθεται η δύναμη στρατού που είναι πιο κοντά σε αυτό. Επομένως για κάθε βάση εξτρεμιστών και για κάθε δύναμη εξτρεμιστών αναθέτουμε ένα κόστος, το οποίο είναι η ελάχιστη απόσταση από κάποια δύναμη στρατού. Αναπαριστούμε την συνθήκη ότι κάθε δύναμη εξτρεμιστών συνδέεται με όλες τις βάσεις που απέχουν το πολύ απόσταση d από αυτούς, ως ένα διμερές κατευθυνόμενο γράφημα που από μια έχει τις βάσεις και από την άλλη τις δυνάμεις εξτρεμιστών.Το in-degree κάθε κόμβου εξτρεμιστών αντιστοιχεί στο σε πόσες βάσεις μπορεί να πάει η συγκεκριμένη δύναμη.

Παρακάτω φαίνεται ένα παράδειγμα του διμερούς γράφου που περιγράφουμε:



Στο παραπάνω γράφημα αρκεί να καλύψουμε όλες τις ακμές επιλέγοντας κορυφές που ελαχιστοποιούν το κόστος.Αυτό συμβαίνει γιατί για να κατακτήσουμε μια δύναμη εξτρεμιστών πρέπει είτε να να την αιχμαλωτίσουμε απευθείας ή να καταστρέψουμε όλες τις βάσεις το πολύ d κοντά της.

Για να το πετύχουμε αυτό τροποποιούμε το γράφημα όπως παρακάτω:



Στο παραπάνω γράφημα όποιες ακμές δεν αναγράφουν τιμή έχουν άπειρο κόστος, το $c(b_i)$ είναι το ελάχιστο κόστος για την κατάρριψη της βάσης b_i και το $c(e_i)$ το ελάχιστο κόστος για την αιχμαλώτισης της e_i ομάδας εξτρεμιστών.

Η ελάχιστη τομή στο παραπάνω γράφημα ισοδυναμεί με το ελάχιστο απαιτούμενο κόστος τους προβλήματος.

Η ελάχιστη τομή δεν θα πάρει καμία από τις μη διακεκομένες ακμές γιατί όπως προαναφέραμε έχουν άπειρη χωρητικότητα.Επομένως κάθε ακμή της τομής ισοδυναμεί με την επιλογή του κόστους της αντίστοιχης κορυφής στο διμερές. Όλες οι ακμές στο διμερές θα καλυφθούν, διότι αν δεν καλυφθούν αυτό σημαίνει ύπαρξη μονοπατιού από το s στο t το οποίο είναι άτοπο διότι πρόκειται περί τομής.Την εύρεση της ελάχιστης τομής την βρίσκουμε εκτελώντας κάποιον αλγόριθμο max-flow/min cut στον παραπάνω γράφο.
Η κατασκευή του γράφουν που περιγράψαμε παραπάνω γίνεται σε γραμμικό χρόνο, οπότε η πολυπλοκότητα του αλγορίθμου εξαρτάται από τον αλγόριθμο που θα επιλέξουμε για την μέγιστη ροή/ελάχιστη τομή.

Άσκηση 5

Τακτοποίηση Ορθογωνίων Παραλληλογράμμων

Το πρόβλημα ανήκει στο NP διότι χωρίζοντας το ορθογώνιο σε τετράγωνα εμβαδού 1 και επειδή τα x_i είναι πολυωνυμικά συσχετισμένα με το n μπορούμε σε πολυωνυμικό χρόνο να ελέγξουμε αν καλύπτονται όλα και αν υπάρχουν επικαλύψεις.

Θα αποδείξουμε ότι το πρόβλημα είναι NP-πλήρες με αναγωγή από το Partition.

Έστω instance του partition με $A = \{a_1, a_2, a_3, \ldots, a_n\}$ και
$w(A) = a_1 + a_2 + a_3 + \ldots + a_n$

Φτιάχνουμε ορθογώνια διαστάσεων $1 \times a_1, 1 \times a_2, \ldots, 1 \times a_n$ και ορίζουμε το B έχει διαστάσεις $2 \times \frac{w(A)}{2}$, άρα προφανώς εμβαδόν ίσο με το άθροισμα των a_i.Με βάση τα παραπάνω παρατηρούμε τα εξής:

- Αν το Partition απαντήσει θετικά για το σύνολο A, σημαίνει ότι αυτό μπορεί να διαμεριστεί σε δύο συμπληρωματικά υποσύνολα A_1 και A_2 τέτοια ώστε
$$w(A_1) = w(A_2) = \frac{w(A)}{2}$$
Το παραλληλόγραμμο εμβαδού B αποτελείται από δύο γραμμές, με εμβαδόν $\frac{w(A)}{2}$, η μία θα πάρει όλα τα παραλληλόγραμμα που αντιστοιχούν στο A_1 και η άλλη τα παραλληλόγραμμα που αντιστοιχούν στο A_2
- Αντίστροφα μία θετική απάντηση στο ΤΟΠ ισοδυναμεί με μια θετική απάντηση στο partition για τον ίδιο ακριβώς λόγο.

Μέγιστη Τομή με Βάρη στις Κορυφές

Το πρόβλημα ανήκει στο NP και το πιστοποιητικό είναι οι κορυφές που ανήκουν σε ένα από τα δύο σύνολα της τομής.Μπορούμε σε τετραγωνικό χρόνο να υπολογίσουμε το συνολικό βάρος που διασχίζει την τομή και να αποφανθούμε αν είναι μεγαλύτερο του B ή όχι.

Θα αποδείξουμε ότι το πρόβλημα είναι NP-πλήρες ανάγοντας το από το Partition.

Έστω ένα στιγμιότυπο του Partition με $A = \{a_1, a_2, a_3, \ldots, a_n\}$ και $w(A) = a_1 + a_2 + a_3 + \ldots + a_n$
Φτιάχνουμε τον πλήρη γράφο με n κορυφές και βάρη που ανήκουν στο A και παίρνουμε επίσης $B = \frac{w^2(A)}{4}$
Επειδή το γράφημα είναι πλήρες το βάρος οποιασδήποτε τομής του έχει μέγεθος $W = \sum_{u \in S} w(u) \sum_{v \in A-S} w(v) = w(S)(w(A) - w(S))$

Το W μεγιστοποιείται για $w(S) = \frac{w(A)}{2}$ με τιμή $W_{max} = \frac{w^2(A)}{4}$

Εφόσον ζητάμε $W \geq B$ και $B = \frac{w^2(A)}{4}$ έχουμε ότι $W = W_{max}$ που σε αυτήν την περίπτωση είναι $w(S) = \frac{w(A)}{2}$, άρα η τομή αποτελεί partition του A.

Αραιό Υπογράφημα

Το πρόβλημα ανήκει στο NP διότι δεδομένου του συνόλου S μπορούμε να βρούμε σε πολυωνυμικό χρόνο πόσες ακμές υπάρχουν ανάμεσα στους κόμβους του S.Θα δείξουμε ότι το πρόβλημα είναι NP-πλήρες ανάγοντας το από το Ανεξάρτητο Σύνολο:

Παίρνουμε τον εξής ορισμό του Ανεξάρτητου Συνόλου: δοσμένου ενός γράφου $G(V,E)$ και ενός φυσικού $g \in \mathbb{N}$ υπάρχει υποσύνολο $I \subseteq V$ με $|I| \geq g$ τέτοιο ώστε $\forall u, v \in I$ με $u \neq v$ και $(u,v) \notin E$.

- Αν το Ανεξάρτητο Σύνολο δώσει θετική απάντηση τότε τετριμμένα και το Αραιό Υπογράφημα με την ίδια ακριβώς είσοδο θα δώσει επίσης θετική απάντηση

- Αν το Αραιό Υπογράφημα δώσει θετική απάντηση σημαίνει ότι υπάρχει $S \subseteq V$ τέτοιο ώστε το επαγόμενο υπογράφημα $G[S]$ να έχει τουλάχιστον k κορυφές και το πολύ k ακμές. Έστω n κορυφές και m ακμές του $G[S]$. Από την ανισότητα του Turan έχουμε ότι $\alpha(G) \geq \frac{n}{1+d_m}$

  Όπου $d_m = \frac{2m}{n}$ ο μέσος βαθμός του G και $\alpha(G)$ το ανεξάρτητο σύνολο μέγιστου μήκος στο G.

Επειδή $m \leq k$ και $n \geq k$ έχουμε ότι $1 + d_m \leq 3$ και συνεπώς $\frac{n}{1+d_m} \geq \frac{k}{3}$

Άρα $\alpha(G) \geq \frac{k}{3}$.

Επομένως αν το Αραιό Υπογράφημα δώσει θετική απάντηση για είσοδο $G(V,E)$ και $k = 3g$ τότε υπάρχει υποσύνολο $I \subseteq V$ με $|I| \geq 3g/3 = g$ κορυφές που σημάνει ότι και το Independent Set απαντάει θετικά.

Συνάθροιση Προτιμήσεων

Το πρόβλημα ανήκει στο NP διότι δοθείσης μιας μετάθεσης μπορούμε σε πολυωνυμικό χρόνο να ελέγξουμε με πόσα ζευγάρια στο C διαφωνεί.

Θα δείξουμε ότι το πρόβλημα είναι NP-πλήρες με αναγωγή από το Minimum Feedback Arc Set ,το οποίο είναι NP-πλήρες(αναγωγή από Vertex Cover).
Το MFAS διατυπώνεται ως εξής: δοθέντος ενός κατευθυνόμενου γραφήματος $G(V, E)$ και ενός θετικού ακεραίου k, υπάρχει σύνολο $S \subseteq E$ με $|S| \leq k$, τέτοιο ώστε η αφαίρεση από το V των ακμών που ανήκουν στο S να αφήνει το G ακυκλικό;

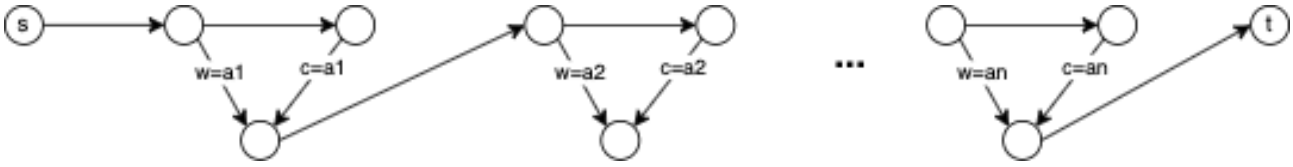
- Αν το MFAS δώσει θετική απάντηση τότε: για κάθε κορυφή του G φτιάχνουμε ένα a_i και για κάθε κατευθυνόμενη ακμή $(a_i, a_j) \in E$ με $i \neq j$ φτιάχνουμε έναν περιορισμό στο σύνολο C. Το $G(V, E - S)$ είναι ένα DAG το οποίο διαφέρει το πολύ με k ακμές με το $G(V, E)$ και η τοπολογική του ταξινόμηση μας δίνει την μετάθεση κορυφών που το πετυχαίνει αυτό.Εφόσον υπάρχει 1-1 αντιστοιχία των κορυφών του γραφήματος στο MFAS με το σύνολο A στο PrefAggr και των ακμών του γραφήματος στο MFAS με το σύνολο C, αυτό σημαίνει ότι και το PrefAggr θα απαντήσει επίσης θετικά.
- Αν το PrefAggr δώσει θετική απάντηση, αυτό σημαίνει ότι υπάρχει μια μετάθεση στο A που διαφωνεί με το C το πολύ κατά k. Η μετάθεση αυτή μπορεί να αναπαρασταθεί ως ένα DAG διότι κάθε a_i έχει όλες τις μπροστά ακμές που αντιστοιχούν σε στοιχεία a_j με $\pi(j) > \pi(i)$ $i \neq j$ και καμία άλλη.Φτιάχνουμε λοιπόν ένα γράφημα με κορυφές στο A και ακμές στο C , για το οποίο όπως προαναφέραμε έχουμε βρει ένα DAG που διαφέρει το πολύ κατά k με το C. Αυτό σημαίνει ότι βρήκαμε ένα σύνολο ακμών το πολύ μεγέθους k που όταν τις αφαιρώ από το γράφημα (τις αφαιρώ γιατί διαφωνούν με το C) έχω DAG, πράγμα το οποίο είναι ακριβώς ο ορισμός του MFAS, επομένως και το MFAS θα απαντήσει θετικά.

Συντομότερο Μονοπάτι με Περιορισμούς

Το πρόβλημα ανήκει στο NP διότι δοσμένου ενός μονοπατιού μπορούμε να επιβεβαιώσουμε σε πολυωνυμικό χρόνο αν ικανοποιεί τους περιορισμούς του W και του C

Θα δείξουμε ότι το πρόβλημα είναι NP-πλήρες με αναγωγή από το Partition.

Έστω στιγμιότυπο του Partition με $A = \{a_1, a_2, a_3, \ldots, a_n\}$ και $w(A) = a_1 + a_2 + a_3 + \ldots + a_n$, άρτιο.
Υπάρχει διαμέριση του σε δύο υποσύνολα A_1, A_2 ξένα μεταξύ τους με $w(A_1) = w(A)/2 = w(A_2)$ αν και μόνο αν $w(A_1) \leq w(A)/2$ και $w(A_2) \leq w(A)/2$. Αυτό συμβαίνει διότι το άθροισμα τους μας δίνει $w(A)$.
Φτιάχνουμε τον γράφο που φαίνεται παρακάτω στον οποίο τα μεγέθη που δεν εικονίζονται είναι 0.



Στο παραπάνω γράφημα που είναι DAG παρατηρούμε ότι κάθε μονοπάτι από το s στο t περνάει από ακριβώς μία από το ζευγάρι της αριστερής (με label w) και της δεξιάς (με label c) ακμών.Συνεπώς αν το Partition δώσει θετική απάντηση τότε θετική θα είναι και η απάντηση για το Μονοπάτι με περιορισμούς με $C = W = w(A)/2$ και αντίστροφα αν για το παραπάνω γράφημα απαντήσουμε θετικά, τότε έχουμε βρει μια διαμέριση του A σε δύο σύνολα με το ίδιο άθροισμα.