Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Αλγόριθμοι & Πολυπλοκότητα – 7ο Εξάμηνο 2018-2019
Αβραμίδης Κλεάνθης – ΑΜ: 03115117

4η Γραπτή Άσκηση – Απαντήσεις

➢ *Θέμα 1*

Το πρόβλημα αποτελείται από ένα γράφημα με n κορυφές – χρήστες και m ακμές – δείκτες εμπιστοσύνης. Ζητείται να προτείνουμε στον χρήστη i τους χρήστες εκείνους με τους οποίους έχει μεγάλη «συνολική» εμπιστοσύνη, σύμφωνα με τη μετρική β, όπου η συνολική εμπιστοσύνη χτίζεται κατά μήκος ενός μονοπατιού του γραφήματος, σύμφωνα με τη σχέση $t(p) = \prod_{q=0}^{l-1} t(q, q+1) \geq \beta_l$.

Μπορούμε λοιπόν να λύσουμε το πρόβλημα μέσω ενός αλγόριθμου εύρεσης συντομότερων μονοπατιών, αρκεί να μετασχηματίσουμε κατάλληλα τα βάρη των ακμών. Για να αντιμετωπίσουμε το ζήτημα του πολλαπλασιασμού στο χτίσιμο συνολικής εμπιστοσύνης λογαριθμούμε τα βάρη:

$$t(p) = \prod_{q=0}^{l-1} t(q, q+1) \geq \beta_l \;\leftrightarrow\; \ln\big(t(p)\big) = \ln\left(\prod_{q=0}^{l-1} t(q, q+1)\right) = \sum_{q=0}^{l-1} \ln\big(t(q, q+1)\big) \geq \ln(\beta_l)$$

Θεωρούμε λοιπόν βάρος το λογάριθμο της αντίστοιχης εμπιστοσύνης. Όμως $t \in (0,1) \rightarrow \ln(t) < 0$. Έχουμε δηλαδή μόνο αρνητικές ακμές και δεν ορίζονται συντομότερα μονοπάτια. Συνεπώς θεωρούμε τα εξής:

$$\sum_{q=0}^{l-1} \ln\big(t(q, q+1)\big) \geq \ln(\beta_l) \;\leftrightarrow\; \sum_{q=0}^{l-1} -\ln\big(t(q, q+1)\big) \leq -\ln(\beta_l) \;\leftrightarrow\; \sum_{q=0}^{l-1} W(q, q+1) \leq -\ln(\beta_l) \quad (1)$$

Εν τέλει, θεωρούμε γράφημα με βάρη $W(i,j) = -\ln\big(t(i,j)\big) > 0$ (όπου $t = 0$ θέτουμε $W \rightarrow \infty$) και στόχος μας είναι, για δεδομένο κόμβο i να βρούμε το σύνολο των κόμβων που σχηματίζουν με αυτόν μονοπάτι μήκους $l \leq k$, με το l να επαληθεύει την ανισότητα (1). Ενδιαφερόμαστε επομένως για τους κόμβους εκείνους που απέχουν μέχρι k από τον αρχικό και συγκεκριμένα για τα συντομότερα μονοπάτια μέχρι τον κόμβο αυτό, όπως υποδεικνύει και η φορά της ανισότητας (1). Για το συγκεκριμένο πρόβλημα λειτουργεί αποδοτικά ο αλγόριθμος Bellman-Ford:

Έστω αρχικός κόμβος (δεδομένος χρήστης) i. Τρέχουμε τον αλγόριθμο ξεκινώντας από αυτόν και αποθηκεύουμε κάθε φορά τον αριθμό των ακμών που έχουμε διανύσει στο τρέχον μονοπάτι. Συγκεκριμένα, θα θεωρήσουμε $d(u,x)$ την ελάχιστη απόσταση που καταγράφεται από τον αλγόριθμο για τους κόμβους $i-u$ με το πολύ x ακμές. Αντί δηλαδή να κρατάμε την απόσταση κάθε κόμβου από τον αρχικό, σε κάθε επανάληψη θα κρατάμε την ελάχιστη απόσταση, συναρτήσει του αριθμού των ακμών του μονοπατιού. Η αναδρομική όπως αναφέρθηκε και στο μάθημα:

$$d(u,x) = \min\big\{d(u,x-1), \min\{d(v,x-1) + W(u,v)\}\big\} \quad (2)$$

Ο αλγόριθμος θα εκτελεστεί k φορές, με κάθε επανάληψη να βασίζεται στις τιμές που έχει αναθέσει η προηγούμενη. Κάθε φορά που ανανεώνεται μια d τιμή, εξετάζουμε την συνθήκη (1) και αν ισχύει, αποθηκεύουμε τον κόμβο-χρήστη ως καλή πρόταση. Η ιδιότητα του αλγόριθμου να αυξάνει ανά επανάληψη το πλήθος των ακμών για κάθε μονοπάτι μας εγγυάται το ακριβές μήκος του μονοπατιού κατά τις ανανεώσεις. Επομένως, μετά το πέρας k επαναλήψεων, θα έχουν υπολογιστεί όλες οι επιθυμητές αποστάσεις d για κάθε κόμβο του γραφήματος και θα έχει βρεθεί το ζητούμενο σύνολο χρηστών που συνιστούν καλές προτάσεις για τον αρχικό κόμβο.

Η πολυπλοκότητα της παραπάνω διαδικασίας είναι: $O(n)$ για την αρχικοποίηση των $d(u,0)$ με βάση τις επιταγές του αλγόριθμου Bellman-Ford (αρχική συνθήκη του DP) και $O(mk)$ για τις k επαναλήψεις, κατά τις οποίες περνάμε από όλες τις ακμές του γραφήματος. Στον χρόνο αυτό συνυπολογίζουμε και τις συγκρίσεις που γίνονται κατά το τρέξιμο του αλγόριθμου, μία για κάθε ανανέωση (σταθερός χρόνος). Συνολικά λοιπόν έχουμε $\boldsymbol{O(n+mk)}$ και θεωρώντας την ακριβότερη περίπτωση της εκφώνησης $O(k) = O(logn)$ καταλήγουμε με $O(n + mlogn)$.

- ***Θέμα 2***

a) Διατηρούμε ως δομή δεδομένων buckets, nC στον αριθμό. Αυτό διότι με C μέγιστο βάρος ακμής και $n-1$ πιθανές ανανεώσεις τιμών ανά κόμβο, προκύπτουν περίπου nC δυνατές αποστάσεις από τον αρχικό κόμβο. Κάθε bucket λοιπόν θα περιέχει τις κορυφές του γραφήματος που θα απέχουν την αντίστοιχη απόσταση από τον αρχικό. Αξιοποιούμε την ιδιότητα του αλγόριθμου Dijkstra να αυξάνει συνεχώς την απόσταση που εξετάζει (εφόσον έχουμε και θετικά πάντα βάρη) και ακολουθούμε την εξής διαδικασία:

Αρχικά κάθε κορυφή έχει απόσταση ∞ εκτός από την αρχική, που έχει απόσταση 0, ενώ βρισκόμαστε με δείκτη στο αριστερότερο bucket (απόσταση 0, περιέχει τον αρχικό κόμβο). Επαναληπτικά, μετακινούμε τον δείκτη δεξιά, μέχρι το πρώτο μη κενό bucket, αφαιρούμε μια κορυφή από αυτό και εκτελούμε μια επανάληψη Dijkstra, εισάγοντας τυχόν ανανεωμένες τιμές στα αντίστοιχα buckets. Το ότι οι αποστάσεις των κόμβων που εξετάζουμε συνεχώς αυξάνεται μας διασφαλίζει πως η διάσχιση των buckets θα είναι συνεπής.

Εφόσον η εισαγωγή και διαγραφή στοιχείων από αυτά γίνεται (μπορεί να γίνει) σε σταθερό χρόνο και σε συνολικά amortized $O(C)$, η πολυπλοκότητα καθορίζεται από τη διάσχιση των buckets, χρόνου nC στη χειρότερη περίπτωση, και του γραφήματος, m στη γενική περίπτωση. Συνολικά προκύπτει πολυπλοκότητα $\boldsymbol{O(nC+m)}$. Παρατηρούμε πως για σχετικά μικρές τιμές του C η δομή των buckets αποδίδει καλύτερα από την βασική υλοποίηση με heap.

b) Στη συγκεκριμένη περίπτωση όπου έχουμε εκθετικά περισσότερες πιθανές τιμές βάρους από πριν, ως δομή δεδομένων διατηρούμε την ουρά προτεραιότητας σε μορφή binary min-heap, όπου κάθε κόμβος του heap δείχνει σε μία δομή (πχ stack, λίστα) με τους κόμβους που έχουν την ίδια απόσταση από τον αρχικό. Η αποδοτικότητα της συγκεκριμένης δομής έγκειται στο ότι όλες οι δυνατές αποστάσεις από τον αρχικό κόμβο (άρα και ο αριθμός των κόμβων του heap) είναι ίσος με 2^C+2, διότι: Κάθε κόμβος ξεκινά από απόσταση ∞ και την στιγμή που θα ανανεωθεί για πρώτη φορά, συμβαίνουν τα εξής:

Η ανανέωση του κόμβου (έστω v) προκύπτει από relax μιας ακμής (u,v), όπου $dist(u,s)=\min(heap)$. Εκείνη τη στιγμή θα ισχύει επίσης $dist(v,s)=dist(u,s)+w(u,v)\leq dist(u,s)+2^C$. Αυτή η ιδιότητα ισχύει βέβαια και για κάθε κορυφή y του heap με πεπερασμένη τιμή: $dist(y,s)\leq dist(y_1,s)+2^C\leq dist(u,s)+2^C$. Αυτό προκύπτει επειδή κάθε y τοποθετήθηκε σε προηγούμενο χρόνο στο heap, συνεπώς $dist(y_1,s)\leq dist(u,s)$. Επομένως θα ισχύει η διπλή ανισότητα $dist(u,s)\leq dist(y,s)\leq dist(u,s)+2^C \quad \forall y: dist(y,s)\neq\infty$. Δηλαδή υπάρχουν πράγματι 2^C+1+1 δυνατές αποστάσεις, άρα και κόμβοι στο heap. Αυτό μας επιτρέπει να εκτελέσουμε τις απαραίτητες ενέργειες στο heap (εισαγωγή, εύρεση και διαγραφή ελαχίστου) σε χρόνο $O(\log(2^C+2))=O(C)$, άρα έχουμε συνολική πολυπλοκότητα $O(C)*(n+m)=\boldsymbol{O\big((n+m)C\big)}$.

- ***Θέμα 3***

a) Το πρόβλημα είναι ουσιαστικά μια παραλλαγή της πρώτης άσκησης, όπου είχαμε μοναδιαίο κόστος στις ακμές. Εδώ ωστόσο θα εφαρμόσουμε, στην ίδια λογική, μια παραλλαγή του αλγόριθμου Dijkstra: Ορίζουμε τον πίνακα αποστάσεων $d(u,x)$ ως την ελάχιστη απόσταση που καταγράφεται από τον αλγόριθμο για τους κόμβους $s-u$ με ακριβώς x κόστος. Θέλουμε για κάθε κορυφή του γραφήματος να βρούμε το $\min\{d(u,x)\}$ για $x\leq k$. Εκτελούμε τον αλγόριθμο για κόστη $\leq k$ και συμπληρώνουμε τις τιμές του πίνακα σύμφωνα με την σχέση:

$$\textbf{if } d(v,x+c_{uv})>d(u,x)+w(u,v) \textbf{ then } d(v,x+c_{uv})=d(u,x)+w(u,v)$$
$$\textbf{push\_to\_queue } (d(v,x+c_{uv}),v)$$

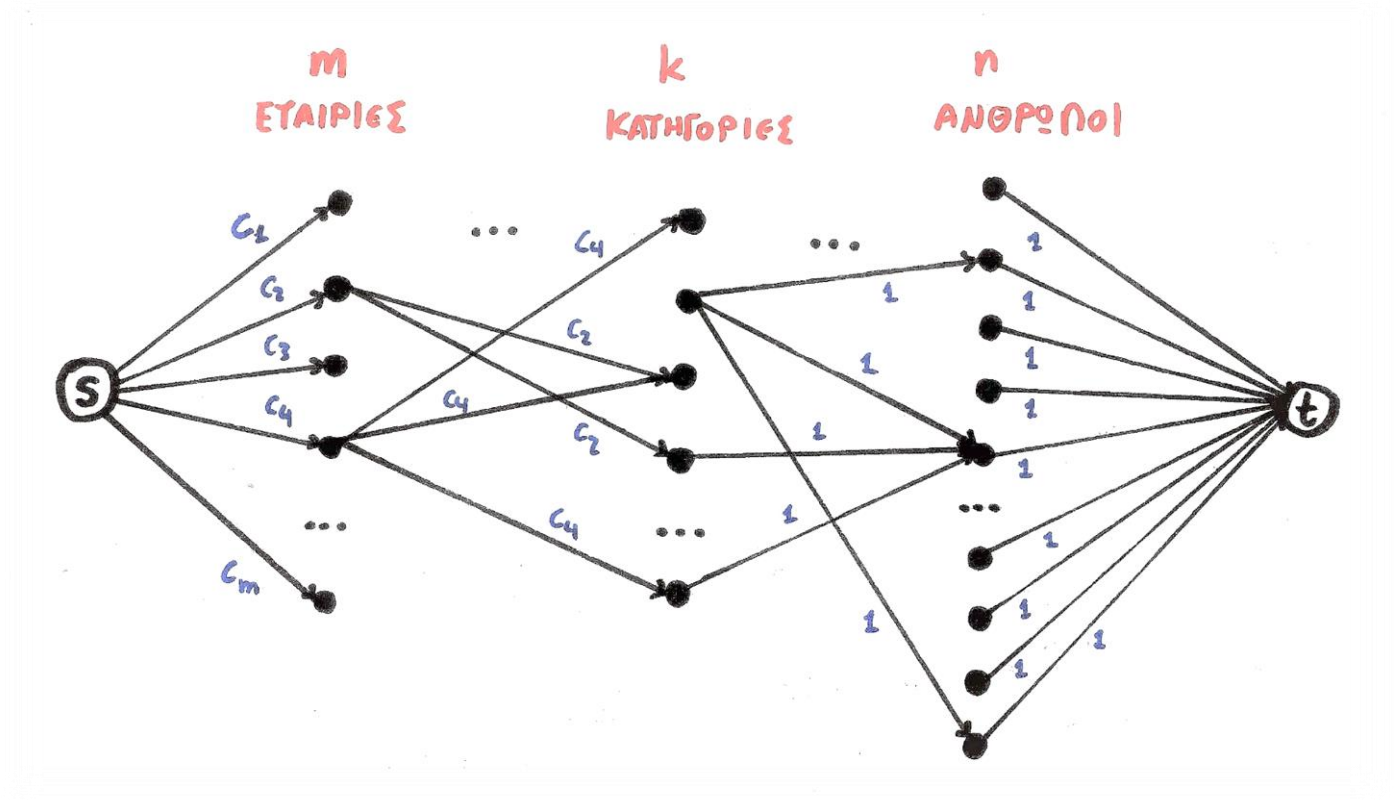
Προκειμένου να είναι ορθή η προσέγγιση, θα πρέπει να αποθηκεύουμε κάθε φορά στην ουρά την απόσταση του κόμβου μαζί με το εκάστοτε κόστος μονοπατιού. Θα θεωρούμε στην ουσία πως κάθε τέτοιο δυνατό ζεύγος <κόμβος, κόστος> θα είναι ξεχωριστός κόμβος. Αντίστοιχα βέβαια θα πρέπει να θεωρήσουμε και πολλαπλές ακμές, που είναι km στον αριθμό, μία για κάθε δυνατό κόστος μονοπατιού. Η διαδικασία δηλαδή συνοψίζεται στην εκτέλεση ενός Dijkstra στο ενισχυμένο αυτό γράφημα $G'(kn,km)$. Μετά το πέρας του θα έχουν υπολογιστεί όλες οι επιθυμητές d τιμές, επομένως υπολογίζουμε γραμμικά για κάθε κόμβο το $\min\{d(u,x)\}$ σε χρόνο nk. Σύμφωνα με την εκφώνηση έχουμε $k\leq n-1$, άρα η συνολική πολυπλοκότητα θα είναι $\boldsymbol{O\big(n^2 log n^2+mn\big)}$.

b) Για τα γενίκευση του παραπάνω προβλήματος θα χρησιμοποιήσουμε τον ίδιο αλγόριθμο. Η μόνη αλλαγή αφορά τις διαστάσεις του πίνακα d που τώρα θα είναι (n, C) αντί για (n, k), άρα η πολυπλοκότητα θα μεταβληθεί σε $\boldsymbol{O(CnlogCn + Cm)}$. Ο αλγόριθμος αυτός είναι ψευδοπολυωνυμικός, καθώς το C είναι στη γενική περίπτωση εκθετικό ως προς το μέγεθος της εισόδου και το ενισχυμένο γράφημα μεγαλώνει κατά μη αποδοτικό τρόπο.

➢ *Θέμα 4*

Το πρόβλημα αποτελείται από n ανθρώπους, k δημογραφικές κατηγορίες και m εταιρίες, κάθε μία από τις οποίες (έστω i) θα προβάλει το πολύ c_i διαφημίσεις τη μέρα σε ένα συγκεκριμένο υποσύνολο $S_i \subseteq \{1, \dots k\}$ ώστε τελικά κάθε άνθρωπος να βλέπει ακριβώς μία διαφήμιση τη μέρα. Θέλουμε να αποφανθούμε κατά πόσον είναι εφικτό αυτό και αν ναι, να βρούμε μια σχετική ανάθεση. Θα μετατρέψουμε το παραπάνω πρόβλημα σε πρόβλημα Μέγιστης Ροής και θα κατασκευάσουμε κατάλληλο γράφημα που να υπακούει στους περιορισμούς της εκφώνησης:

Θεωρούμε κορυφές για τους ανθρώπους, τις κατηγορίες και τις εταιρίες, συν 2 κορυφές s, t που αντιστοιχούν στην πηγή και την καταβόθρα του συστήματος. Συνολικά δηλαδή έχουμε γράφημα με $n + k + m + 2$ κορυφές. Καθορίζουμε τις ακμές: Συνδέουμε την s με κάθε εταιρία μέσω ακμής χωρητικότητας c_i, στην συνέχεια κάθε εταιρία με κάθε κατηγορία $\in S_i$ μέσω ακμής χωρητικότητας c_i (το πολύ c_i διαφημίσεις), κάθε άνθρωπο με κάθε κατηγορία στην οποία ανήκει μέσω ακμής χωρητικότητας 1 (στέλνεται το πολύ μία διαφήμιση) και τέλος την t με κάθε άνθρωπο μέσω ακμής χωρητικότητας 1 (βλέπεται το πολύ μία διαφήμιση). Προκύπτει το εξής γράφημα:



Πλέον μπορούμε να εφαρμόσουμε τον αλγόριθμο Ford-Fulkerson για τον υπολογισμό της $s - t$ μέγιστης ροής. Εν προκειμένω ενδιαφερόμαστε για την περίπτωση κατά την οποία η μέγιστη ροή είναι ίση με n, κάτι που συνεπάγεται πως κάθε άνθρωπος βλέπει μία διαφήμιση. Σε διαφορετική περίπτωση ο αλγόριθμος τερματίζει ανεπιτυχώς. Από κει και πέρα, η εύρεση της ανάθεσης γίνεται ως εξής: Επιλέγουμε μια εταιρία και για κάθε κατηγορία με την οποία έχει μη μηδενική ροή παρατηρούμε ποιοι άνθρωποι έχουν ροή προς αυτή. Από αυτούς διαλέγουμε έναν αριθμό ίσο με τη ροή της εταιρίας. Επαναλαμβάνουμε για κάθε εταιρία και καταλήγουμε με μια συνολική ζητούμενη ανάθεση.

Η δημιουργία του γραφήματος γίνεται σε χρόνο αντίστοιχο με το πλήθος των στοιχείων του, $O(mk + nk)$. Η εύρεση της ανάθεσης σύμφωνα με την παραπάνω περιγραφή γίνεται σε $O(mk + n)$. Ο αλγόριθμος Ford-Fulkerson από την άλλη έχει πολυπλοκότητα $O(Ef) = O\big((nk + mk + m + n) * n\big) = O\big(nk(m + n)\big)$, καθιστώντας αδιάφορες τις υπόλοιπες διαδικασίες από άποψη πολυπλοκότητας. Συνολικά λοιπόν, η πολυπλοκότητα είναι $\boldsymbol{O\big(nk(m + n)\big)}$.

- ***Θέμα 5***

3-PARTITION

Το πρόβλημα ανήκει στο NP, αφού δοθέντων τριών συνόλων $A_1,\ A_2,\ A_3$ μπορούμε γραμμικά να ελέγξουμε κατά πόσο το άθροισμα των στοιχείων τους είναι $w(A)/3$, δηλαδή αποτελούν 3-διαμέριση του Α. Για να αποδείξουμε τώρα ότι το πρόβλημα είναι NP-hard θα κάνουμε πολυωνυμική αναγωγή στο πρόβλημα PARTITION: Θεωρούμε πίνακα $A = \{w_1, w_2, \dots, w_n\}$ και κατασκευάζουμε σε γραμμικό χρόνο $A' = \left\{w_1, w_2, \dots, w_n, w' = \frac{w_1+w_2+\cdots+w_n}{2}\right\}$. Θα δείξουμε ότι υπάρχει 2-διαμέριση του A αν και μόνο αν υπάρχει 3-διαμέριση του A' ώστε να επιτύχουμε την αναγωγή. Παρατηρούμε πως κάθε 3-διαμέριση του A' θα θέτει το w' μόνο του σε σύνολο, αφού

$$\frac{w(A')}{3} = \frac{w_1 + w_2 + \cdots + w_n + w'}{3} = \frac{2w_1 + 2w_2 + \cdots + 2w_n + w_1 + w_2 + \cdots + w_n}{6} = \frac{w_1 + w_2 + \cdots + w_n}{2} = w'$$

Συνεπώς $A_1 = \{w'\}$ και για τα άλλα θα ισχύει προφανώς $A_2 = A_3 = \frac{w_1+w_2+\cdots+w_n}{2}$. Τα δύο τελευταία σύνολα στοιχειοθετούν μια 2-διαμέριση για το A. Επίσης, με δεδομένη αυτή τη διαμέριση μπορούμε, εφαρμόζοντάς τη στα n πρώτα στοιχεία του A', να λάβουμε 3-διαμέριση για το A'. Συνεπώς, το πρόβλημα 3-PARTITION είναι NP-complete.

AP-SS

Το πρόβλημα ανήκει στο NP, αφού δοθέντος συνόλου S μπορούμε σε γραμμικό χρόνο να ελέγξουμε την συνθήκη $B - x \leq w(S) \leq B$. Για να αποδείξουμε τώρα ότι το πρόβλημα είναι NP-hard θα κάνουμε πολυωνυμική αναγωγή στο πρόβλημα SUBSET SUM: Θεωρούμε $A_1 = \{w_1, w_2, \dots, w_n\}$, με παράμετρο W το γενικό στιγμιότυπο του SS. Θέτουμε $A = \{2w_1, 2w_2, \dots, 2w_n\},\ B = 2W,\ x = 1$ και δείχνουμε πως υπάρχει λύση του SS αν και μόνο αν υπάρχει λύση του AP-SS όπως ορίστηκαν παραπάνω. ***Ευθύ***: Έστω S_A η λύση του SS. Τότε, επιλέγοντας τους αντίστοιχους διπλάσιους αριθμούς λαμβάνουμε $w(S'_A) = 2W$ που ικανοποιεί την συνθήκη $2W - 1 \leq w(S'_A) \leq 2W$ του AP-SS. ***Αντίστροφο***: Έστω S'_A η λύση του AP-SS, δηλαδή ισχύει ότι $2W - 1 \leq w(S'_A) \leq 2W$. Το $w(S'_A)$ όμως πρέπει να είναι άρτιος, αφού κάθε στοιχείο του A' είναι άρτιος. Έτσι προκύπτει αναγκαστικά $w(S'_A) = 2W = 2w(S_A) \rightarrow w(S_A) = w(S'_A)/2$. Άρα οι αντίστοιχοι αριθμοί του A επιλύουν το SS. Συνεπώς, το πρόβλημα AP-SS είναι NP-complete.

AP-HC

Το πρόβλημα ανήκει στο NP, αφού δοθέντος ενός κύκλου του γραφήματος μπορούμε σε γραμμικό χρόνο να επαληθεύσουμε πρώτον ότι είναι κύκλος και δεύτερον ότι περιλαμβάνει 1-2 φορές κάθε κορυφή (κύκλος Hamilton). Για να αποδείξουμε τώρα ότι το πρόβλημα είναι NP-hard θα κάνουμε πολυωνυμική αναγωγή στο πρόβλημα HAMILTON CYCLE: Έστω γράφημα G και G' το γράφημα με ίδιες κορυφές και ακμές με το G συν μία κορυφή u' γειτονική $\forall u$. Θα δείξουμε πως υπάρχει κύκλος Hamilton στο G αν και μόνο αν υπάρχει AP-HC στο G'. ***Ευθύ***: Με δεδομένο κύκλο Hamilton στο G, η προσθήκη σε αυτόν των επιπλέον κορυφών του G' θα ικανοποιούσε τους περιορισμούς του AP-HC, αφού κάθε «παλιά» κορυφή θα βρίσκεται στον κύκλο 2 φορές και κάθε νέα από 1 φορά. ***Αντίστροφο***: Παρατηρούμε πως η ύπαρξη AP-HC στο G' επιβάλλει να περάσουμε 2 φορές από κάθε κορυφή του G, περνώντας διπλή φορά από τις επιπλέον ακμές του G', ώστε να συμπεριλάβουμε τις νέες κορυφές . Συνεπώς αν τις αφαιρέσουμε, θα περνάμε από 1 φορά κάθε κορυφή του αρχικού γραφήματος, δηλαδή το G θα έχει κύκλο Hamilton. Άρα, το πρόβλημα AP-HC είναι NP-complete αφού ανάγεται στο NP-complete πρόβλημα HAMILTON CYCLE.

C-SAT

Το πρόβλημα ανήκει στο NP, αφού για δεδομένη ανάθεση τιμών αληθείας μπορούμε σε γραμμικό χρόνο να αποφανθούμε επί της εγκυρότητας, με απλή αντικατάσταση. Για να αποδείξουμε τώρα ότι το πρόβλημα είναι NP-hard θα κάνουμε πολυωνυμική αναγωγή στο πρόβλημα 3-SAT: Θεωρούμε $\varphi = \bigwedge_{i=1}^{n}(l_{i1} \vee l_{i2} \vee l_{i3})$ μια 3-CNF πρόταση και θέλουμε να εξετάσουμε την ικανοποιησιμότητά της. Κατασκευάζουμε σε πολυωνυμικό χρόνο πρόταση $\psi = \varphi = \bigwedge_{i=1}^{n}(l_{i1} \vee l_{i2} \vee l_{i3} \vee x)$. Θα δείξουμε πως ψ ικανοποιήσιμη $\leftrightarrow$ φ ικανοποιήσιμη: Έστω ότι υπάρχει ανάθεση που ικανοποιεί την ψ. Τότε, αν $z = F$, τουλάχιστον ένα από τα υπόλοιπα literals θα είναι T, συνεπώς η ίδια ανάθεση θα ικανοποιεί την φ. Αν πάλι $z = T$, τότε το πολύ 2 από τα υπόλοιπα literals δε θα είναι T οπότε η αντίθετη ανάθεση αυτών θα ικανοποιεί τη φ. Αντίστροφα, αν υπάρχει ανάθεση που ικανοποιεί τη φ, θα περιέχει ένα τουλάχιστον T literal, συνεπώς με $z = F$ ικανοποιείται η ψ. Συνεπώς δείξαμε πως το C-SAT είναι NP-complete.

SET-PACKING

Το πρόβλημα ανήκει στο NP, αφού για συγκεκριμένα υποσύνολα μπορούμε να ελέγξουμε το αν είναι ανά 2 ξένα σε πολυωνυμικό χρόνο. Μια brute-force προσέγγιση είναι για κάθε ζεύγος σετ (n^2 στο σύνολο) να ελέγξουμε κάθε πιθανό ζεύγος στοιχείων (επίσης n^2), προκειμένου να εντοπίσουμε κοινά στοιχεία. Η συνολική πολυπλοκότητα είναι προφανώς πολυωνυμική. Για να αποδείξουμε τώρα ότι το πρόβλημα είναι NP-hard θα κάνουμε πολυωνυμική αναγωγή στο πρόβλημα Max-Independent-Set (MIS): Για κάθε κόμβο u γραφήματος G δημιουργούμε σύνολο S_u με όλες τις γειτονικές του ακμές. Θεωρούμε E το σύνολο των ακμών του G και τη σταθερά του SP k ίση με τη σταθερά του MIS. Έστω ότι έχουμε ανεξάρτητο σύνολο κορυφών με πληθάριθμο τουλάχιστον k. Κανένα ζεύγος του συνόλου αυτού δε θα έχει κοινή ακμή, συνεπώς κανένα ζεύγος των αντίστοιχων υποσυνόλων του SP δε θα έχει κοινό στοιχείο (ξένα). Αντίστροφα, αν έχουμε δεδομένα k τουλάχιστον τέτοια υποσύνολα του E, στο G θα έχουμε ίδιου πλήθους σύνολο κορυφών χωρίς κοινές ακμές. Άρα το G θα έχει ανεξάρτητο σύνολο κορυφών που θα ικανοποιεί τον περιορισμό του MIS. Συνεπώς, μέσω της αναγωγής, το πρόβλημα SET-PACKING είναι NP-complete.

CONSTRAINT SHORTEST PATH

Το πρόβλημα ανήκει στο NP, αφού για δοθέν μονοπάτι μπορούμε να ελέγξουμε αν πληροί τους περιορισμούς με μια απλή του διάσχιση. Για να αποδείξουμε τώρα ότι το πρόβλημα είναι NP-hard θα κάνουμε πολυωνυμική αναγωγή στο πρόβλημα KNAPSACK. Έστω n στοιχεία με βάρη w_i και κέρδη p_i. Θέλουμε να δείξουμε κατά πόσον υπάρχει υποσύνολό τους με $\sum w \leq W$ και $\sum p \geq P$. Μετασχηματίζουμε αυτό το στιγμιότυπο KNAPSACK σε ένα ειδικό για το πρόβλημά μας: Θεωρούμε γράφημα με κορυφές τα n στοιχεία που ορίσαμε με 2 κατευθυνόμενες ακμές για την αντιστοίχιση ενός στοιχείου με το επόμενο. Η μία θα έχει βάρος w_i και κόστος $P_s - p_i$ και θα αντιστοιχεί στην επιλογή του επόμενου αντικειμένου ενώ η άλλη θα έχει μηδενικό βάρος και κόστος P_s και θα αντιστοιχεί στη μη επιλογή του αντικειμένου. Θέτοντας W το βάρος των ακμών του γραφήματος και $n * P_s - P$ το κόστος, έχουμε μετασχηματίσει πλήρως το πρόβλημα σε εύρεση συντομότερου μονοπατιού με τους παραπάνω περιορισμούς. Μέσω του μετασχηματισμού λοιπόν δείξαμε πως το πρόβλημά μας είναι ισοδύναμο με το KNAPSACK, άρα NP-complete.

- ✓ ***Παρατηρήσεις επί των απαντήσεων:***