Αλγόριθμοι και Πολυπλοκότητα
3η Σειρά Γραπτών Ασκήσεων

Ελισάβετ Παπαδοπούλου,
Α.Μ.: 03120190

Άσκηση 1: Υπολογισιμότητα

Η κλάση RE περιέχει όλα τα ημιαποκρίσιμα υπολογιστικά προβλήματα, δηλαδή αυτά για τα οποία υπάρχει αλγόριθμος που τα ημιαποφασίζει. Με τον όρο **ημι-αποφασίζει**, εννοούμε πως, στην περίπτωση που η απάντηση σε κάποιο ζητούμενο ενός προβλήματος είναι ναι, τότε υπάρχει κάποιος αλγόριθμος ο οποίος το καθορίζει σε πεπερασμένο χρόνο και χωρίς εσφαλμένες επιβεβαιώσεις. Αν όμως η απάντηση είναι όχι, ο αλγόριθμος δεν είναι απαραίτητο ότι θα τερματίσει.
Ισοδύναμα, η κλάση RE απαρτίζεται από τα προβλήματα απόφασης για τα οποία μια μηχανή Turing μπορεί να απαριθμήσει όλες τις θετικές απαντήσεις, δηλαδή όλα τα "ναι".

(a) Για να αποδείξουμε ότι το Πρόβλημα Διοφαντικών Εξισώσεων, -όπου η είσοδος είναι μια πολυωνυμική εξίσωση με ακέραιους συντελεστές και πεπερασμένο αριθμό αγνώστων, και το ζητούμενο αν έχει ακέραιες λύσεις ή όχι- ανήκει στην κλάση RE (Recursively Enumerable) θα εργαστούμε ως εξής:
Αρχικά, υποθέτουμε ότι υπάρχει ένα σύνολο αριθμών K, το οποίο προσδιορίζεται από την Διοφαντική Εξίσωση $p(x, y_1, y_2, ..., y_k) = 0$. Εδώ, το πρόβλημα απαντάται με "ναι", αν το αποτέλεσμα p ισούται με 0 και με όχι σε οποιαδήποτε άλλη περίπτωση. Έτσι, αν εναλλάξουμε όλους τους πιθανούς μοναδικούς συνδυασμούς αναθέσεων τιμών στις μεταβλητές $x, y_1, y_2, ..., y_k$, και καταγράφουμε κάθε τιμή του x που αντιστοιχεί σε αποτέλεσμα $p = 0$, τελικά θα έχουμε απαριθμήσει όλα τα μέλη του σετ K. Ωστόσο, είναι σημαντικό να σημειωθεί ότι δεν είναι πάντα εφικτό να βρούμε όλες τις λύσεις μιας Διοφαντικής Εξίσωσης, καθώς μερικές εξισώσεις μπορεί να έχουν άπειρες λύσεις ή να είναι ανεπίλυτες.

(b) Το **πρόβλημα τερματισμού HP** μπορεί να οριστεί ως η απόφαση του αν ένα αυθαίρετο υπολογιστικό πρόγραμμα θα σταματήσει να τρέχει ή θα συνεχίσει να τρέχει για πάντα.
Ένα πρόβλημα είναι RE-πλήρες αν:

- ανήκει στην κλάση RE και
- κάθε πρόβλημα της κλάσης RE μπορεί να αναχθεί στο πρόβλημα αυτό με τρόπο που είναι συνεπής στους περιορισμούς της κλάσης.

Δηλαδή, αν μπορεί να λυθεί το δωθέν πρόβλημα, τότε μπορεί να λυθεί κάθε πρόβλημα της κλάσης RE.
Για να αποδείξουμε πως το Halting Problem (HP) ανήκει στην κλάση RE, θα ανάγουμε αρχικά το Πρόβλημα των Διοφαντικών Εξισώσεων σε αυτό ως εξής:
Η αναγωγή του προβλήματος των Διοφαντικών Εξισώσεων στο Halting Problem προϋποθέτει την απόδειξη ότι η επίλυση του ενός προβλήματος μπορεί να οδηγήσει στην επίλυση του άλλου. Πιο συγκεκριμένα, αυτό σημαίνει ότι εάν καταφέρουμε να λύσουμε το Halting Problem, τότε θα μπορούμε επίσης να λύσουμε το πρόβλημα της εύρεσης λύσεων σε Διοφαντικές Εξισώσεις.
Θεωρούμε ένα αυθαίρετο Turing Machine, δηλαδή μια τυχαία περίπτωση του HP. Θεωρούμε επίσης έναν αλγόριθμο που μετατρέπει μια Διοφαντική Εξίσωση σε ένα πρόγραμμα, έτσι ώστε το πρόγραμμα να τερματίζει αν και μόνο αν η εξίσωση έχει λύση.
Βάζοντας το πρόγραμμα στην μηχανή Turing δηλαδή, αν το μηχάνημα τερματίσει με κάποια είσοδο, τότε η Διοφαντική Εξίσωση που του ανατέθηκε έχει ακέραιη λύση. Αν δεν τερματίσει, η εξίσωση δεν έχει

τέτοια λύση.
Συμβολικά, θεωρούμε μια μηχανή H του Halting Problem, P ένα πρόγραμμα που βρίσκει αν κάποια Διοφαντική Εξίσωση έχει λύση, και I μια οποιαδήποτε Διοφαντική Εξίσωση. Τότε, για την μηχανή H θα έχω:

$$H(P,I) = \begin{cases} \text{"ναι"} & \text{αν το πρόγραμμα } P \text{ τερματίζει με την είσοδο } I, \text{ δηλαδή αν το P(I) = "ναι",} \\ \text{"όχι"} & \text{αν το πρόγραμμα } P \text{δεν τερματίζει με την είσοδο } I \text{ , δηλαδή το P(I) δεν τερματίζει.} \end{cases}$$

Με αυτή την απλή αναγωγή, αποδεικνύεται πως αν υπήρχε αλγόριθμος ο οποίος μπορεί να αποφασίσει αν μια οποιαδήποτε Διοφαντική Εξίσωση έχει λύση ή όχι, τότε αυτός θα μπορούσε να χρησιμοποιηθεί για να λύσει το Halting Problem. Αφού έχει αποδειχθεί πως το HP είναι μη αποφάσιμο, ακολουθεί πως δεν μπορεί να υπάρχει τέτοιος αλγόριθμος για τις Διοφαντικές Εξισώσεις. Άρα στην περίπτωση που το HP λυνόταν, τότε θα λυνόταν και το Πρόβλημα των Διοφαντικών Εξισώσεων.
Άρα, αν υποθέσουμε ότι D είναι το σύνολο των εξισώσεων που έχουν ακέραιες λύσεις, ισχύει η ισοδυναμία:

$$I \in D \iff H(P,I) = \text{"ναι"}$$

Τώρα, θα αποδείξουμε πως κάθε σετ που ανήκει στην RE κλάση ανήκει και στο σετ που ορίζει το Πρόβλημα Διοφαντικών Εξισώσεων.
Έστω K ένα σετ που ορίζεται από μια Διοφαντική Εξίσωση. δηλαδή υπάρχει κάποιο p τέτοιο ώστε $x \in K \iff \exists y_1 \exists y_2 ... \exists y_k \ldots$ τ.ω. $p(x, y_1, y_2, ..., y_k) = 0$. Έστω τώρα ότι υπάρχει κάποιος αλγόριθμος απόφασης που απαντάει στο αν μια Διοφαντική Εξίσωση $p(x, y_1, y_2, ..., y_k) = 0$ ικανοποιείται. Τότε, αποφασίζοντας για την τιμή του x, μπορούμε να εφαρμόσουμε τον αλγόριθμο απόφασης για να απαντήσουμε στο αν το $x \in K$. Όμως, αυτό σημαίνει πως η υπόθεση ότι υπάρχει κάποιος αλγόριθμος ο οποίος αποφασίζει αν ένα πολυώνυμο ισούται με μηδέν, υποννοεί πως κάθε RE σετ αριθμών K, αν είναι διοφαντικό, μπορεί να έχει απόφαση. Αυτό γνωρίζουμε ότι είναι άτοπο. Έτσι, αποδεικνύεται πως κάθε RE πρόβλημα είναι Πρόβλημα Διοφαντικών Εξισώσεων.
Με αυτούς τους δύο ισχυρισμούς, μπορούμε πλέον να αποδείξουμε πως πράγματι, κάθε πρόβλημα που ανήκει στην κλάση RE μπορεί να αναχθεί στο πρόβλημα τερματισμού HP.

(c) Ένα πρόβλημα θεωρείται RE-δύσκολο (RE-hard) αν κάθε πρόβλημα της κλάσης RE (Recursively Enumerable) μπορεί να αναχθεί σε αυτό.
Για να αποδείξουμε ότι το Πρόβλημα Καθολικού Τερματισμού είναι RE-δύσκολο, πρέπει να δείξουμε ότι κάθε πρόβλημα στην κλάση RE μπορεί να αναχθεί σε αυτό. Αυτό μπορεί να γίνει με την επίδειξη ότι η επίλυση του Προβλήματος Καθολικού Τερματισμού θα επέτρεπε την επίλυση οποιουδήποτε άλλου προβλήματος στην κλάση RE.
Στο Πρόβλημα Καθολικού Τερματισμού, δίνεται μια μηχανή Turing M και ζητείται αν η M τερματίζει για όλες τις εισόδους ή όχι. Εάν καταφέρουμε να αποφασίσουμε αυτό το πρόβλημα για κάθε μηχανή Turing, τότε μπορούμε να αποφασίσουμε για κάθε πρόβλημα στην κλάση RE, καθώς κάθε τέτοιο πρόβλημα μπορεί να αναπαρασταθεί από μια μηχανή Turing της παρακάτω μορφής:

$$M(P,I) = \begin{cases} \text{"ναι"} & \text{αν το πρόγραμμα } P \text{ τερματίζει με την είσοδο } I, \\ \text{"όχι"} & \text{αν το πρόγραμμα } P \text{δεν τερματίζει με την είσοδο } I. \end{cases}$$

Ωστόσο, το Πρόβλημα Καθολικού Τερματισμού είναι αποδεδειγμένα μη επιλύσιμο. Αυτό σημαίνει ότι δεν υπάρχει γενικός αλγόριθμος που μπορεί να αποφασίσει για κάθε δυνατή μηχανή Turing M αν θα τερματίσει ή όχι για όλες τις εισόδους.
Η απόδειξη αυτή έγγυται στην θεώρηση ενός προγράμματος C, το οποίο ορίζεται ως εξής:

Algorithm 1 C(X)

```
1: if M(X, X) == "ναι" then
2:     loop forever.
3: else
4:     halt.
5: end if
```

Στο πρόγραμμα αυτό, αν θέσουμε όπου X τον εαυτό του, τότε ουσιαστικά δηλώνεται πως, αν η μηχανή M που έχουμε θεωρήσει ως καθολική Μηχανή Turing βρει πως το πρόγραμμα C με είσοδο C τερματίζει, τότε το C(C) θα τρέχει για πάντα. Αν από την άλλη βρει πως τρέχει για πάντα, τότε αυτό θα τερματίζει. Αυτό αποτελεί ένα παράδοξο που αποδεικνύει την μη αποφασιμότητα του προβλήματος αυτού.
Επομένως, το Πρόβλημα Καθολικού Τερματισμού είναι RE-δύσκολο και μη επιλύσιμο, καθώς η επίλυσή του θα επέτρεπε την επίλυση οποιουδήποτε προβλήματος στην κλάση RE, αλλά δεν υπάρχει γενικός αλγόριθμος που μπορεί να το επιλύσει.

Άσκηση 2: Πολυπλοκότητα - Αναγωγές

(a) Για να περιγράψουμε αναγωγή πολυωνυμικού χρόνου ($\leq p$) από το πρόβλημα **UnSat**, δηλαδή το συμπλήρωμα του προβλήματος **Satisfiability**) στο πρόβλημα **NoLargeClique**, θα ορίσουμε αρχικά κάποιους όρους:

- **Αναγωγή Πολυωνυμικού Χρόνου:** Ύπαρξη μιας υπολογίσιμης συνάρτησης πολυωνυμικού χρόνου εκτέλεσης, που μετατρέπει στιγμιότυπα του ενός προβλήματος σε στιγμιότυπα του άλλου.
- **UnSat**: Το συμπλήρωμα του προβλήματος SAT, δηλαδή η απόφαση του αν μια πρόταση δεν έχει ερμηνεία.
- **NoLargeClique:** Σε ένα γράφο G, ζητείται αν κάθε κλίκα του έχει μέγεθος το πολύ k. Πρόκειται για το συμπλήρωμα του προβλήματος **Clique Problem**.

Πλέον, μπορούμε να κάνουμε την αναγωγή ως εξής:
Έστω πως το πρόβλημα της **NoLargeClique** μπορεί να διαγνωστεί σε πολυωνυμικό χρόνο
Έστω επίσης ένας SAT τύπος: $(a_1 \vee \ldots \vee f_1) \wedge \ldots \wedge (a_k \vee \ldots \vee d_k)$
Δημιουργείται ένας γράφος του οποίου οι κόμβοι είναι κάθε μια μεταβλητή του τύπου και οι αρνήσεις τους, και ακμή ανάμεσα σε δύο κόμβους υπάρχει αν οι μεταβλητές που τους αντιστοιχούν είτε είναι αντίθετες μεταξύ τους, είτε ανήκουν σε διαφορετικές φράσεις και δεν είναι αντίθετες μεταξύ τους.
Ορίζουμε τον ακέραιο k ίσο με τον αριθμό των όρων, και πλέον μπορούμε να τρέξουμε τον αλγόριθμο ύπαρξης κλίκας στον γράφο που παράγεται:

- Αν ο αλγόριθμος απαντήσει θετικά, δηλαδή πως πράγματι δεν υπάρχει κλίκα μεγαλύτερη από k, τότε ο τύπος είναι ανικανοποίητος. Αυτό ισχύει, καθώς μια τέτοια κλίκα θα αντιπροσώπευε μια εφικτή ανάθεση τιμών στις μεταβλητές που θα ικανοποιούσε την έκφραση.
- Αν ο αλγόριθμος απαντήσει αρνητικά τότε ο τύπος μπορεί να ικανοποιηθεί.

Η αναγωγή αυτή είναι πολυωνυμικού χρόνου, καθώς ο γράφος G και ο ακέραιος k μπορούν να δημιουργηθούν από την αρχική λογική έκφραση σε χρόνο που είναι πολυωνυμικός στο μέγεθος της έκφρασης.

(b) Έστω C μια κλάση πολυπλοκότητας και Π ένα πρόβλημα που είναι C-πλήρες. Αυτό σημαίνει ότι:

- Το Π ανήκει στην κλάση C.
- Κάθε άλλο πρόβλημα στην κλάση C μπορεί να αναχθεί στο Π μέσω μιας αναγωγής $\leq R$ (π.χ., πολυωνυμικού χρόνου).

Το συμπλήρωμα του προβλήματος Π, που ονομάζουμε Π′, είναι ένα πρόβλημα όπου οι απαντήσεις για όλες τις πιθανές εισόδους είναι το αντίθετο από τις απαντήσεις του Π. Θέλουμε να δείξουμε ότι το Π′ είναι co-C-πλήρες, δηλαδή ότι:

- Π′ πρέπει να ανήκει στην κλάση co-C. Αυτό συμβαίνει από τον ορισμό της co-C, καθώς τα προβλήματα στην co-C είναι ακριβώς τα συμπληρωματικά των προβλημάτων στην C.
- Κάθε άλλο πρόβλημα στην co-C μπορεί να αναχθεί στο Π′. Αυτό μπορεί να αποδειχθεί ως εξής: αν έχουμε ένα πρόβλημα Π_1 στην co-C, τότε το συμπλήρωμά του Π_1' ανήκει στην C. Επειδή το Π είναι C-πλήρες, υπάρχει αναγωγή από το Π_1' στο Π. Αυτή η αναγωγή μπορεί να χρησιμοποιηθεί για να δημιουργήσει μια αναγωγή από το Π_1 (το συμπλήρωμα του Π_1') στο Π′, καθώς το Π′ είναι το συμπλήρωμα του Π.

Ξεκινάμε αναφέροντας ότι το πρόβλημα SAT είναι NP-πλήρες. Αυτό σημαίνει πως κάθε πρόβλημα της κλάσης NP μπορεί να αναχθεί στο SAT μέσω μιας αναγωγής πολυωνυμικού χρόνου. Το πρόβλημα UnSat είναι το συμπλήρωμα του, και είναι coNP-πλήρες.
Από την προηγούμενη αναγωγή πολυωνυμικού χρόνου από το UnSat στο NoLargeClique, και συμπεραίνουμε πως το δεύτερο είναι coNP-hard, και άρα πως το LargeClique NP-hard.
Για να είναι το NoLargeClique coNP-πλήρες, πρέπει να αποδείξουμε ότι ανήκει στην κλάση coNP και ότι κάθε άλλο πρόβλημα στην κλάση coNP μπορεί να αναχθεί σε αυτό. Η αναγωγή από το UnSat στο NoLargeClique, μαζί με την απόδειξη ότι το NoLargeClique ανήκει στην κλάση coNP (πράγμα που συμβαίνει επειδή η απόρριψη του προβλήματος μπορεί να επιβεβαιωθεί σε πολυωνυμικό χρόνο), αρκεί για να αποδείξει ότι το NoLargeClique είναι coNP-πλήρες.

(c) Για να αποδείξουμε ότι αν ένα NP-πλήρες πρόβλημα ανήκει στην κλάση $NP \cap coNP$ τότε $NP = coNP$, αρχικά

1) **Κλάση NP:** Ένα πρόβλημα ανήκει στην κλάση NP αν υπάρχει μια μη-αποφασιστική Turing Μηχανή που μπορεί να επιλύσει το πρόβλημα σε πολυωνυμικό χρόνο. Δηλαδή, ένα πιθανό "ναι" αποτέλεσμα, μπορεί να επαληθευτεί σε πολυωνυμικό χρόνο. Με άλλα λόγια, αν δωθεί μια πιθανή λύση σε ένα NP πρόβλημα, μπορεί να ελεγχθεί αν αυτή η λύση είναι σωστή σε πολυωνυμικό χρόνο.

2) **Κλάση coNP:** Ένα πρόβλημα ανήκει στην κλάση coNP αν το συμπληρωματικό του ανήκει στην NP. Δηλαδή, αν μια απάντηση "όχι" μπορεί να επαληθευτεί σε πολυωνυμικό χρόνο.

3) **NP-πλήρη προβλήματα:** Ένα πρόβλημα είναι NP-πλήρες αν ανήκει στην κλάση NP και κάθε άλλο πρόβλημα της κλάσης NP μπορεί να αναχθεί σε αυτό σε πολυωνυμικό χρόνο.

Αν υποθέσουμε ότι υπάρχει ένα NP-πλήρες πρόβλημα Pr, το οποίο ανήκει επίσης στην κλάση $NP \cap coNP$, αυτό σημαίνει ότι για αυτό το πρόβλημα μπορεί να επαληθευτεί τόσο ένα πιθανό "ναι", αλλά και ένα πιθανό "όχι" σε πολυωνυμικό χρόνο.
Δεδομένου ότι κάθε πρόβλημα NP μπορεί να αναχθεί σε ένα NP-πλήρες πρόβλημα σε πολυωνυμικό χρόνο, και αφού το δεδομένο πρόβλημα ανήκει στην $NP \cap coNP$ κλάση, αυτό σημαίνει πως κάθε πρόβλημα στην κλάση NP μπορεί επίσης να ανήκει στην coNP. Αυτό οδηγεί στο συμπέρασμα πως $NP = coNP$.
Δηλαδή:
Έστω τυχαίο

$$X \in NP \Rightarrow X \leq pPr, Pr = coNP - \pi\lambda\rho\varepsilon\varsigma \Rightarrow X \in coNP$$
$$X \in coNP \Rightarrow X \leq pPr, Pr = NP - \pi\lambda\rho\varepsilon\varsigma \Rightarrow X \in NP$$

Άρα,

$$(X \in NP \iff X \in coNP) \Rightarrow NP = coNP$$

(d) Για να αποδείξουμε ότι το πρόβλημα NAE3SAT (Not-All-Equal 3 SAT) είναι NP-πλήρες, ορίζουμε αρχικά το πρόβλημα ως εξής:

1 **NAE3SAT:** Δίνεται ένας τύπος προτασιακού λογισμού σε μορφή 3-SAT, και ζητείται αν υπάρχει ανάθεση η οποία σε κάθε clause ικανοποιεί τουλάχιστον 1 και το πολύ 2 literals.

Αρχικά, είναι προφανές πως το πρόβλημα ανήκει στην κλάση NP, καθώς, αν δοθεί μια ανάθεση τιμών στις μεταβλητές, μπορούμε να ελέγξουμε γρήγορα (σε πολυωνυμικό χρόνο) αν σε κάθε clause ικανοποιεί τουλάχιστον ένα και το πολύ δύο literals.
Για να δείξουμε πως το NAE3SAT είναι NP-πλήρες, θα πρέπει να ανάγουμε πολυωνυμικά κάποιο άλλο NP-πλήρες πρόβλημα σε αυτό. Ένα τέτοιο πρόβλημα μπορεί να είναι το **3SAT**, το οποίο είναι γνωστό NP-πλήρες πρόβλημα. Για την αναγωγή:
Έστω ο τύπος 3-SAT

$$(a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$$

Για να μετασχηματίσουμε τον παραπάνω τύπο σε τύπο NAE3SAT, για κάθε ένα από τα k clauses του σχηματίζουμε νέα clauses και δημιουργούμε επιπλέον μεταβλητές, μοναδικές για αυτά, προκειμένου να εξασφαλίσουμε ότι σε κάποια ανάθεση που ικανοποιεί το 3-SAT πρόβλημα, δεν θα μπορούν να ικανοποιούνται λιγότερα από ένα και παραπάνω από δύο literals ανά clause. Το clause 1 για παράδειγμα μετασχηματίζεται ως εξής:

$$(a_1 \vee b_1 \vee e_1) \wedge (a_1 \vee c_1 \vee \overline{e_1})$$

Με την νέα μεταβλητή e_j για καθένα από τα C_j clauses, εξασφαλίζεται πως:

- **Οποιαδήποτε ανάθεση ικανοποιούσε τα αρχικά clauses, και άρα την αρχική πρόταση, θα ικανοποιεί και τον μετασχηματισμό τους.** Αυτό συμβαίνει διότι υπάρχει κατάλληλη ανάθεση των τιμών των μεταβλητών e_j τέτοια, ώστε σε όλα τα clauses να ικανοποιούνται από ένα εώς δύο literals. Δηλαδή, μια πιθανή ανάθεση στο παραπάνω παράδειγμα θα ήταν η:

$$a_1 = TRUE, b_1 = TRUE, c_1 = FALSE, e_1 = FALSE$$

- **Οποιαδήποτε ανάθεση που ικανοποιεί τον μετασχηματισμό της πρότασης ικανοποιεί και την αρχική πρόταση.** Αυτό μπορεί να φανεί από το ότι, για κάθε clause C_j που ικανοποιείται στον μετασχηματισμό, ισχύει πως υπάρχει κάποιο $x_j = TRUE, x \in \{a, b, c\}$, γεγονός το οποίο αρκεί για την ικανοποίηση του αντίστοιχου clause από το οποίο αυτό δημιουργήθηκε. Επομένως, η αρχική πρόταση ικανοποιείται.

Ο παραπάνω μετασχηματισμός, όπως φάνηκε, μπορεί να γίνει σε πολυωνυμικό χρόνο. Άρα, ισχύει πως:

$$\left.\begin{matrix} 3-SAT \leq pNAE3SAT, \\ 3-SAT \in NP-complete \end{matrix}\right\} \Rightarrow NAE3SAT \in NP \tag{1}$$

(e) Προκειμένου να αποδείξουμε πως το πρόβλημα Επιλογής Αντιπροσώπων είναι NP-πλήρες, αρκεί να αποδείξουμε πως:

1 **Το πρόβλημα ανήκει στην κλάση NP.** Αυτό μπορεί να αποδειχθεί εύκολα, καθώς, με δεδομένη κάποια δοσμένη λύση ένα σύνολο R, τ.ω. $|R| \leq k$, μπορούμε σε πολυωνυμικό χρόνο να επαληθεύσουμε αν πράγματι κάθε ομάδα U_i έχει κάποιον αντιπρόσωπο στο σύνολο, δηλαδή αν $U_i \cap R \neq \emptyset \quad i \leq r$.

2 **Κάποιο άλλο NP-πλήρες πρόβλημα μπορεί να αναχθεί σε αυτό σε πολυωνυμικό χρόνο.** Για αυτό το βήμα, ένα κατάλληλο NP-Πλήρες πρόβλημα είναι το Κάλυμμα Συνόλων (Set Cover).
Στο πρόβλημα αυτό, δίνεται ένα σύνολο U, μια συλλογή S από υποσύνολα του U και ένας ακέραιος k. Ζητείται να βρεθεί αν υπάρχει κάποιο σύνολο C που απαρτίζεται από τα υποσύνολα του S, με $|C| \leq k$, τ.ω. η ένωση των συνόλων στο C να καλύπτει όλο το U.
Για την αναγωγή από το πρόβλημα αυτό στο πρόβλημα Επιλογής Αντιπροσώπων, θεωρούμε ότι το U στο πρόβλημα Συνόλου Κάλυψης αντιστοιχεί στο σύνολο ατόμων U στο πρόβλημα Επιλογής Αντιπροσώπων. Θεωρούμε επίσης κάθε υποσύνολο του S στο πρόβλημα Συνόλου Κάλυψης ως μια ομάδα U_i στο πρόβλημα Επιλογής Αντιπροσώπων. Αν υπάρχει λύση C για το πρόβλημα του Συνόλου Κάλυψης, η οποία καλύπτει όλο το U με $|C| \leq k$, τότε αντίστοιχα υπάρχει λύση σύνολο R στο Επιλογής Αντιπροσώπων (το οποίο προκύπτει από τα υποσύνολα στο C) το οποίο αντιπροσωπεύει κάθε ομάδα U_i.

Παρατηρούμε τελικά, πως ισχύει ότι:

$$\left.\begin{array}{l}\text{Κάλυμα\_Συνόλων} \leq p \text{ Επιλογή\_Αντιπροσώπων,}\\ \text{Κάλυμα\_Συνόλων} \in NP - complete\end{array}\right\} \Rightarrow \text{ Επιλογή\_Αντιπροσώπων} \in NP \qquad (2)$$

Άσκηση 3: Προσεγγιστικοί Αλγόριθμοι

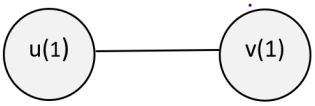
(a) Δίνεται ένας αλγόριθμος προσέγγισης για το Weighted Vertex Cover και ζητείται να αποδείξουμε ότι επιτυγχάνει λόγο προσέγγισης 2, και πιο συγκεκριμένα θα πρέπει να δείξουμε ότι ικανοποιούνται οι τρεις παρατηρήσεις που δίνονται:

1 **Το C καλύπτει όλες τις ακμές:** Η παρατήρηση αυτή επαλήθεύεται, καθώς μέσα στο while loop εξετάζεται κάθε φορά κάποια από τις ακμές που δεν έχουν καλυφθεί, ενώ ο αλγόριθμος για κάθε μία ακμή που εξετάζει αποθηκεύει τουλάχιστον την μία από τις κορυφές της στο C.

2 **Το συνολικό βάρος του C είναι μικρότερο ή ίσο με $2\sum_{e\in E} c(e)$:** Αυτή η παρατήρηση ισχύει, καθώς κάθε ακμή e συμβάλλει στο βάρος C με το ποσό $c(e)$, και κάθε κορυφή στο C μπορεί να συμμετέχει σε το πολύ δύο τέτοιες ακμές. Έτσι, το συνολικό βάρος του C δεν μπορεί να ξεπερνάει το διπλάσιο του συνολικού βάρους που ανατίθεται στις ακμές.

3 **Το συνολικό βάρος της βέλτιστης λύσης είναι μεγαλύτερο ή ίσο με $\sum_{e\in E} c(e)$:** Αυτό συμβαίνει, διότι ο αλγόριθμος θα εξετάσει μέσα στο while loop όλες τις ακάλυπτες ακμές, και για κάθε μία ακμή e το βάρος $c(e)$ αντιπροσωπεύει το ελάχιστο απαραίτητο που μπορεί να ανατεθεί σε μια κορυφή ώστε να καλυφθεί η ακμή. Έτσι, κάθε βέλτιστη λύση θα πρέπει τουλάχιστον να αντιστοιχίζει αυτό το βάρος σε κάθε ακμή για να καλύψει όλες τις ακμές.

Παράδειγμα Γραφήματος:
Ένα απλό γράφημα, που δείχνει ότι ο αλγόριθμος μπορεί να δώσει λύση με διπλάσιο βάρος από την βέλτιστη λύση είναι το ακόλουθο:



Εδώ, ο αλγόριθμος θα επιλέξει να βάλει και τις δύο ακμές στο Vertex Cover C, κάνοντας το τελικό βάρος ίσο με **2**. Στην βέλτιστη λύση ωστόσο, συμπεριλαμβάνεται μόνο ο ένας από τους δύο κόμβους, και το τελικό βάρος είναι ίσο με **1**.

(b) Για να αποδείξουμε ότι δεν υπάρχει πολυωνυμικός αλγόριθμος που επιτυγχάνει λόγο προσέγγιση k για το πρόβλημα του Πλανώδιου Πωλητή (Traveling Salesman Problem - TSP), εκτός αν ισχύει $P = NP$, θα χρησιμοποιήσουμε μια παραλλαγή της κλασικής αναγωγής από το πρόβλημα του Hamilton Cycle στο TSP.

1 **Hamilton Cycle Problem:** Δίνεται ένας γράφος $G(V, E)$, και ζητείται αν υπάρχει ένας κύκλος που περνά ακριβώς μια φορά από κάθε κορυφή του γράφου.

2 **Τροποποίηση Αναγωγής Hamilton Cycle στο TSP:** Δημιουργούμε μια είσοδο για το TSP από έναν γράφο του Hamilton Cycle Problem ως εξής:

- Αν δύο κορυφές είναι συνδεδεμένες στον γράφο του Hamilton Cycle, τότε η απόσταση τους στο TSP είναι 1.
- Αν οι δύο κορυφές δεν είναι συνδεδεμένες, η απόστασή τους στο TSP είναι $k + 1$, όπου το k είναι η σταθερά προσέγγισης.

Κάθε ακμή στοον κύκλο Hamilton έχει μήκος 1 στο TSP. Έτσι, αν ο αρχικός γράφος έχει κάποιον Hamilton κύκλο, τότε θα υπάρχει κάποιο μονοπάτι στον γράφο TSP με συνολικό μήκος ίσο με τον αριθμό των κορυφών του γράφου.
Αν δεν υπάρχει κύκλος Hamilton, τότε οποιαδήποτε διαδρομή στο TSP θα πρέπει να περιλαμβάνει

τουλάχιστον μία ακμή με μήκος $k+1$, και επομένως το συνολικό μήκος θα είναι μεγαλύτερο από τον αριθμό των κορυφών συν k.

3 **Λόγος Προσέγγισης** k**:** Αν υπήρχε κάποιος πολυωνυμικός προσεγγιστικός αλγόριθμος, με λόγο προσέγγισης k για το TSP, τότε θα μπορούσε να λυθεί και το πρόβλημα Hamilton Cycle. Δηλαδή, αν βρισκόταν κάποια διαδρομή με συνολικό μήκος μικρότερο ή ίσο με τον αριθμό των κορυφών, τότε θα υπήρχε κάποιος κύκλος Hamilton.
Έτσι, θα οδηγούμασταν στην πολυωνυμική λύση ενός NP-πλήρους προβλήματος (Hamilton Cycle), δηλαδή ενός προβλήματος στο οποίο αποδεδειγμένα μπορούν αναχθούν όλα τα υπόλοιπα NP προβλήματα. Έτσι, αναγκαστικά θα ίσχυε πως $P = NP$.