ΠΛΗ30

ΕΝΟΤΗΤΑ 6: NP-πληρότητα

Μάθημα 6.1:
Θεωρία Πολυπλοκότητας και
Το Πρόβλημα της Ικανοποιησιμότητας (SAT)

Δημήτρης Ψούνης

ΠΕΡΙΕΧΟΜΕΝΑ

Α. Σκοπός του Μαθήματος

Οι στόχοι του μαθήματος είναι:

Επίπεδο Α
- (-)

Επίπεδο Β
- Το πρόβλημα του τερματισμού SAT και αλγόριθμοι επίλυσής του.

Επίπεδο Γ
- Κλάσεις Πολυπλοκότητας

Β. Θεωρία

1. Εισαγωγή

1. Είδη Προβλημάτων

Τα προβλήματα που λύνονται από υπολογιστές διαχωρίζονται σε κατηγορίες

- **Απόφασης:** Προβλήματα που απαντιούνται με ΝΑΙ/ΟΧΙ
  - Παράδειγμα: Δίνεται γράφος $G = (V, E)$, αφετηρία $s \in V$ και προορισμός $t \in V$. Υπάρχει μονοπάτι από το s στο t;
  - Υπόδειγμα Απάντησης: ΝΑΙ
- **Αναζήτησης:** Επιστρέφουν δομή που απεικονίζει μία λύση
  - Παράδειγμα: Δίνεται γράφος $G = (V, E)$, αφετηρία $s \in V$ και προορισμός $t \in V$. Ζητείται ένα μονοπάτι από το s στο t
  - Υπόδειγμα Απάντησης: Ένα Μονοπάτι (π.χ. $s = s - v_1 - v_6 - v_4 - t$)
- **Βελτιστοποίησης:** Επιστρέφουν δομή που απεικονίζει την βέλτιστη λύση
  - Παράδειγμα: Δίνεται γράφος με βάρη $G = (V, E, W)$, αφετηρία $s \in V$ και προορισμός $t \in V$. Ζητείται το συνομότερο μονοπάτι από το s στο t
  - Υπόδειγμα Απάντησης: Το μονοπάτι που είναι ελάχιστο ως προς το μήκος (π.χ. $s = s - v_1 - v_6 - v_4 - t$ με βάρος 20)

Β. Θεωρία

1. Εισαγωγή

1. Είδη Προβλημάτων

Τα προβλήματα που λύνονται από υπολογιστές διαχωρίζονται σε κατηγορίες

- **Απαρίθμησης:** Επιστρέφουν σύνολο από δομές που απεικονίζουν λύσεις
  - Παράδειγμα: Δίνεται γράφος $G = (V, E)$, αφετηρία $s \in V$ και προορισμός $t \in V$. Ποια τα μονοπάτια από το s στο t;
  - Υπόδειγμα Απάντησης: (Όλα τα μονοπάτια από το s στο t)

- **Άθροισης:** Επιστρέφουν πλήθος λύσεων
  - Παράδειγμα: Δίνεται γράφος $G = (V, E)$, αφετηρία $s \in V$ και προορισμός $t \in V$. Ζητείται το πλήθος των μονοπατιών από το s στο t
  - Υπόδειγμα Απάντησης: (Το πλήθος των μονοπατιών από το s στο t)

- Κάθε πρόβλημα έχει τις ισοδύναμες εκδοχές του ως πρόβλημα απόφασης, αναζήτησης, βελτιστοποίησης, απαρίθμησης και άθροισης.
- Στην συνέχεια του μαθήματος εστιάζουμε το ενδιαφέρον μας στα προβλήματα απόφασης.

Β. Θεωρία

1. Εισαγωγή

2. Μοντέλα Υπολογισμού

Αναφορικά με την πολυπλοκότητα χρόνου (δηλαδή πόσο χρόνο χρειάζεται για να τρέξει ένας αλγόριθμος θεωρούμε δύο μοντέλα στα οποία μπορούμε να εργασθούμε

- **Τις Ντετερμινιστικές Μηχανές Turing**
  - Δηλαδή αλγορίθμους οι οποίοι λειτουργούν με έναν ακολουθιακό τρόπο (όπως για παράδειγμα τα προγράμματα C,Pascal κ.λπ.).
  - Διατυπώνουμε έναν αλγόριθμο για μία ντετερμινιστική μηχανή Turing, διατυπώνοντας έναν ακολουθιακό αλγόριθμο.

- **Τις Μη Ντετερμινιστικές Μηχανές Turing**
  - Δηλαδή αλγορίθμους οι οποίοι λειτουργούν θεωρώντας ότι υπάρχει τέτοια μηχανή και μπορούμε να τις χρησιμοποιήσουμε για την επίλυση προβλημάτων.
  - Θα μελετήσουμε πως διατυπώνουμε ένα μη ντετερμινιστικό αλγόριθμο.

- Το πρόβλημα μας είναι η ισοδυναμία των παραπάνω μοντέλων.

Β. Θεωρία

2. Το πρόβλημα της Ικανοποιησιμότητας

1. Διατύπωση

Το πρόβλημα της ικανοποιησιμότητας (Satisfiability - SAT):
- **Είσοδος:** Δίνεται φόρμουλα φ σε κάνονική συζευκτική μορφή.
- **Ερώτημα:** Είναι η φ ικανοποιήσιμη;

Υπενθύμιση:
Μια φόρμουλα προτασιακής λογικής φ είναι σε κανονική συζευκτική μορφή (Κ.Σ.Μ.) αν είναι στην μορφή:

$$C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

δηλαδή αποτελείται από m προτάσεις που η κάθε πρόταση C_i είναι της μορφής:

$$z_{i_1} \vee z_{i_2} \vee \cdots \vee z_{i_n}$$

δηλαδή αποτελείται από i_n όρους με κάθε όρο να είναι είτε μία προτασιακη μεταβλητή x_j είτε άρνηση προτασιακής μεταβλητής $\overline{x_j}$.

Επίσης μία φόρμουλα λέμε ότι είναι ικανοποίησιμη, αν υπάρχει αποτίμηση των μεταβλητών (τιμές Α,Ψ) που να την κάνουν αληθή.

Β. Θεωρία

2. Το πρόβλημα της Ικανοποιησιμότητας

1. Διατύπωση

Παράδειγμα 1:
Η φόρμουλα SAT:

$$\varphi_1 = (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_2} \vee x_3)$$

Είναι ικανοποιήσιμη, για παράδειγμα με την αποτίμηση $x_1 = \text{Α}, x_2 = \text{Α}, x_3 = \text{Α}$

Παράδειγμα 2:
Η φόρμουλα SAT:

$$\varphi_2 = (\overline{x_1} \vee x_2) \wedge (x_1 \vee x_2 \vee x_3) \wedge (\overline{x_2}) \wedge (x_1 \vee x_2 \vee \overline{x_3})$$

Δεν είναι ικανοποιήσιμη.
- Για να αληθεύει η πρόταση $(\overline{x_2})$ πρέπει $x_2 = \Psi$
- Συνεπώς η πρόταση $(\overline{x_1} \vee x_2)$ είναι $(\overline{x_1} \vee \Psi)$ πρέπει $x_1 = \Psi$
- Συνεπώς η πρόταση $(x_1 \vee x_2 \vee x_3)$ είναι $(\Psi \vee \Psi \vee x_3)$ πρέπει $x_3 = \text{Α}$
- Συνεπώς η πρόταση $(x_1 \vee x_2 \vee \overline{x_3})$ είναι $(\Psi \vee \Psi \vee \Psi)$ άρα δεν είναι αληθής.

Άρα η φόρμουλα είναι ψευδής.

Β. Θεωρία

2. Το πρόβλημα της Ικανοποιησιμότητας

2. Το SAT λύνεται σε εκθετικό ντετερμινιστικό χρόνο.

Ο ακόλουθος ντετερμινιστικός αλγόριθμος λύνει το πρόβλημα της ικανοποιησιμότητας:

ΝΤΕΤΕΡΜΙΝΙΣΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ ΓΙΑ ΤΟ SAT
Είσοδος: Φόρμουλα φ σε ΚΣΜ
Έξοδος: ΝΑΙ/ΟΧΙ ανάλογα με το αν η φόρμουλα είναι ικανοποιήσιμη.
---
1. Κάνε την παραγωγή όλων των δυνατών αποτιμήσεων των προτασιακών μεταβλητών.
2. Έλεγξε διαδοχικά κάθε μία αποτίμηση αν ικανοποιεί την φόρμουλα
   1. Αν την ικανοποιεί, τερμάτισε απαντώντας ΝΑΙ.
   2. Αν δεν την ικανοποιεί προχώρα στην επόμενη αποτίμηση
3. Αν καμία αποτίμηση δεν ικανοποιεί την φόρμουλα τερμάτισε απαντώντας ΟΧΙ.

Σημείωση:
Ένας ντετερμινιστικός αλγόριθμος είναι ένας αλγόριθμος που μπορεί να υλοποιηθεί από μια γλώσσα προγραμματισμού (π.χ. C). Με βάση την θέση των Church-Turing θα υλοποιείται και από μία ντετερμινιστική μηχανή Turing.

Β. Θεωρία

2. Το πρόβλημα της Ικανοποιησιμότητας

2. Το SAT λύνεται σε εκθετικό ντετερμινιστικό χρόνο.

Ανάλυση Πολυπλοκότητας του Αλγορίθμου:

Θεωρώντας ότι η φόρμουλα έχει n μεταβλητές και m προτάσεις, έχουμε:

- Το βήμα 1 (κατασκευή όλων των αποτιμήσεων) υλοποιείται σε χρόνο $\Theta(n \cdot 2^n)$
- Στο βήμα 2 για κάθε αποτίμηση (2^n φορές)
  - Ελέγχουμε αν η αποτίμηση ικανοποιεί την φόρμουλα σε χρόνο ανάλογο με το μήκος της φόρμουλας O(mn)

  Στην χειρότερη περίπτωση (μη ικανοποίησιμη φόρμουλα) θα απαιτηθεί χρόνος $O(2^n \cdot mn)$
- Το βήμα 3 απαιτεί χρόνο $\Theta(1)$

Η συνολική πολυπλοκότητα είναι $O(2^n \cdot mn)$

Άρα η πολυπλοκότητα του αλγορίθμου είναι **εκθετική**. Συνεπώς ο ντετερμινιστικός αλγόριθμος που προτείναμε απαιτεί εκθετικό χρόνο.

Λέμε ότι το **SAT λύνεται σε Ντετερμινιστικό Εκθετικό Χρόνο.**

Β. Θεωρία

2. Το πρόβλημα της Ικανοποιησιμότητας

3. Το SAT λύνεται σε πολυωνυμικό ντετερμινιστικό χρόνο?

- Είναι ανοικτό ερώτημα αν μπορούμε να βρούμε κάποιον αλγόριθμο με καλύτερη πολυπλοκότητα
  - Έχουν προταθεί αλγόριθμοι που ρίχνουν την πολυπλοκότητα (κουρεύοντας κάποιες από τις αποτιμήσεις) σε $\Theta(1{,}9^n)$, $\Theta(1{,}8^n)$
  - Ωστόσο μέχρι σήμερα δεν έχει βρεθεί κάποιος αλγόριθμος που να είναι πολυωνυμικής πολυπλοκότητας
- Η ύπαρξη πολυωνυμικού ντετερμινιστικού αλγορίθμου για το SAT αποτελεί **ανοικτό ερώτημα**:
  - Δεν έχει βρεθεί πολυωνυμικός αλγόριθμος
  - Δεν έχει αποδειχθεί ότι τέτοιος αλγόριθμος δεν μπορεί να κατασκευασθεί

Είναι ανοικτό ερώτημα αν το **SAT λύνεται σε Ντετερμινιστικό Πολυωνυμικό Χρόνο.**

Β. Θεωρία

2. Το πρόβλημα της Ικανοποιησιμότητας

4. Το SAT λύνεται σε μη ντετερμινιστικό πολυωνυμικό χρόνο.

Μελετούμε την επίλυση του SAT από έναν μη ντετερμινιστικό αλγόριθμο (δηλαδή μια υλοποίηση ενός αλγορίθμου από μη ντετερμινιστική μηχανή Turing)

ΜΗ ΝΤΕΤΕΡΜΙΝΙΣΤΙΚΟΣ ΑΛΓΟΡΙΘΜΟΣ ΓΙΑ ΤΟ SAT
Είσοδος: Φόρμουλα φ σε ΚΣΜ
Έξοδος: ΝΑΙ/ΟΧΙ ανάλογα με το αν η φόρμουλα είναι ικανοποιήσιμη.
---
1. Επανέλαβε για i=1…n

   1. Μη Ντετερμινιστικά ανέθεσε στην μεταβλητή x_i τις τιμές x_i=Α και x_i=Ψ

2. Επαλήθευσε αν η αποτίμηση που έχει κατασκευασθεί ικανοποιεί την φόρμουλα.

Σημείωση:
Ο παραπάνω αλγόριθμος είναι μη ντετερμινιστικός. Συνεπώς κατασκευάζονται 2^n μονοπάτια υπολογισμού. Αν κάποια από τις ενσαρκώσεις απαντήσει ΝΑΙ, τότε η συνολική απάντηση είναι ΝΑΙ. Αν όλες απαντήσουν ΟΧΙ, τότε η συνολική απάντηση είναι ΟΧΙ.

Β. Θεωρία

2. Το πρόβλημα της Ικανοποιησιμότητας

4. Το SAT λύνεται σε μη ντετερμινιστικό πολυωνυμικό χρόνο.

Σχηματικά η εκτέλεση αυτού του αλγορίθμου είναι η εξής:

Ανάλυση Πολυπλοκότητας: Η επανάληψη απαιτεί n χρονικά βήματα. Έπειτα η επαλήθευση ότι η αποτίμηση ικανοποιεί την φόρμουλα γίνεται σε χρόνο O(nm). Συνεπώς η συνολική πολυπλοκότητα είναι O(nm)

Συνεπώς το **SAT λύνεται σε Μη Ντετερμινιστικό Πολυωνυμικό Χρόνο.**

Β. Θεωρία

2. Το πρόβλημα της Ικανοποιησιμότητας

4. Το SAT λύνεται σε μη ντετερμινιστικό πολυωνυμικό χρόνο.

Παρόλο που είναι εμφανές ότι ο μη ντετερμινιστικός αλγόριθμος που περιγράψαμε παράγει όλες τις δυνατές αποτιμήσεις (υποψήφιες λύσεις) και εξακριβώνει αν κάποια από αυτές ικανοποιεί την φόρμουλα:

- Αντί να λέμε «Η μη ντετερμινιστική Μ.Τ. παράγει σε πολυωνυμικό χρόνο όλες τις δυνατές λύσεις και επαληθεύει αν κάποια από αυτές ικανοποιεί την φόρμουλα»
- Θα λέμε «Η μη ντετερμινιστική Μ.Τ. **μαντεύει** την λύση σε πολυωνυμικό χρόνο και **επαληθεύει** ότι ικανοποιεί την φόρμουλα.»

Η ορολογία αυτή είναι πολύ σημαντική για την συνέχεια του μαθήματος και στις αποδείξεις που ζητούνται στις εξετάσεις.

Β. Θεωρία

2. Το πρόβλημα της Ικανοποιησιμότητας

5. Σύνοψη για το πρόβλημα SAT

Συνοψίζοντας για το πρόβλημα SAT:

Σχετικά με την επίλυση του από ντετερμινιστική μηχανή Turing:

- Το **SAT λύνεται σε Εκθετικό Ντετερμινιστικό Χρόνο.**
- Είναι ανοικτό ερώτημα αν το SAT λύνεται σε **Πολυωνυμικό Ντετερμινιστικό Χρόνο**

Σχετικά με την επίλυση του από μη ντετερμινιστική μηχανή Turing:

- Το **SAT λύνεται σε Μη Ντετερμινιστικό Πολυωνυμικό Χρόνο.**

Επίσης έχει αποδειχθεί ότι πολλά ακόμη προβλήματα έχουν παρόμοια συμπεριφορά με το SAT. Π.χ. για το πρόβλημα του Περιοδεύοντος Πωλητή TSP:

Σχετικά με την επίλυση του από ντετερμινιστική μηχανή Turing:

- Το **TSP λύνεται σε Εκθετικό Ντετερμινιστικό Χρόνο.**
- Είναι ανοικτό ερώτημα αν το TSP λύνεται σε **Πολυωνυμικό Ντετερμινιστικό Χρόνο**

Σχετικά με την επίλυση του από μη ντετερμινιστική μηχανή Turing:

- Το **TSP λύνεται σε Μη Ντετερμινιστικό Πολυωνυμικό Χρόνο.**

Τα προβλήματα που παρουσίαζουν παρόμοια συμπεριφορά είναι αρκετές δεκάδες χιλιάδες. Αυτό οδήγησε στην ανάπτυξη της θεωρίας πολυπλοκότητας.

Β. Θεωρία

3. Θεωρία Πολυπλοκότητας

1. Η κλάση P

Θεωρούμε όλα τα προβλήματα που λύνονται από ντετερμινιστική Μ.Τ. σε πολυωνυμικό χρόνο και τα ομαδοποιούμε σε ένα σύνολο, το οποίο ονομάζουμε P (polynomial deterministic time):

Ορίζουμε την κλάση προβλημάτων απόφασης P:

$$\boldsymbol{P = \{\text{Συνολο προβληματων που λυνονται σε πολυωνυμικο ντετερμινιστικο χρονο}\}}$$

Πιο τυπικά αν $DTIME(f(n))$ είναι το σύνολο των προβλημάτων που λύνονται σε ντετερμινιστικό χρόνο $O(f(n))$ τότε:

$$P = \bigcup_k DTIME(n^k)$$

Στην ενότητα 2, είδαμε πολλά προβλήματα που ανήκει στην κλάση P

Το πρόβλημα ShortestPath $\in P$ αφού επιλύεται από τον αλγόριθμο του Dijkstra σε $O(n^2 \log n)$
Το πρόβλημα MinimumSpanningTree $\in P$ αφού επιλύεται από τον αλγόριθμο του Prim σε $O(n^3)$

Ωστόσο για το πρόβλημα SAT δεν υπάρχει απόδειξη ούτε αν το SAT $\in P$ ούτε αν το SAT $\notin P$

Β. Θεωρία

3. Θεωρία Πολυπλοκότητας

2. Η κλάση NP

Θεωρούμε όλα τα προβλήματα που λύνονται από μη ντετερμινιστική Μ.Τ. σε πολυωνυμικό χρόνο και τα ομαδοποιούμε σε ένα σύνολο, το οποίο ονομάζουμε NP (non deterministic polynomial time):

Ορίζουμε την κλάση προβλημάτων απόφασης NP:

$$NP = \{\textbf{Συνολο προβληματων που λυνονται σε μη ντετερμινιστικο πολυωνυμικο χρονο}\}$$

Πιο τυπικά αν $NTIME(f(n))$ είναι το σύνολο των προβλημάτων που λύνονται σε μη ντετερμινιστικό χρόνο $O(f(n))$ τότε:

$$NP = \bigcup_k NTIME(n^k)$$

Για το πρόβλημα SAT έχουμε:

- $SAT \in NP$ αφού λύνεται με τον μη ντετερμινιστικό αλγόριθμο που περιγράψαμε

Ισχύει επίσης ότι κάθε ντετερμινιστική μηχανή Turing μπορεί να προσομοιωθεί από μία μη ντετερμινιστική μηχανή Turing άρα κάθε πρόβλημα που ανήκει στο P ανήκει και στο NP.

Β. Θεωρία

3. Θεωρία Πολυπλοκότητας

3. Η κλάση EXP

Θεωρούμε όλα τα προβλήματα που λύνονται από ντετερμινιστική Μ.Τ. σε εκθετικό χρόνο και τα ομαδοποιούμε σε ένα σύνολο, το οποίο ονομάζουμε EXP (exponential deterministic time):

Ορίζουμε την κλάση προβλημάτων απόφασης EXP:

$$EXP = \{\textbf{Συνολο προβληματων που λυνονται σε εκθετικο ντετερμινιστικο χρονο}\}$$

Πιο τυπικά αν $DTIME(f(n))$ είναι το σύνολο των προβλημάτων που λύνονται σε ντετερμινιστικό χρόνο $O(f(n))$ τότε:

$$EXP = \bigcup_k DTIME(2^{n^k})$$

Για το πρόβλημα SAT έχουμε:

- $SAT \in EXP$ αφού λύνεται με τον εκθετικό αλγόριθμο που περιγράψαμε.

Κάθε πρόβλημα που ανήκει στο P ανήκει και στο EXP αφού λύνεται σε χρόνο το πολύ εκθετικό.

Β. Θεωρία

3. Θεωρία Πολυπλοκότητας

4. $P \subseteq NP \subseteq EXP$

Το ακόλουθο θεώρημα μας εξασφαλίζει μια ιεραρχία στις κλάσεις πολυπλοκότητας:

Ισχύει ότι:

$$\boldsymbol{P \subseteq NP \subseteq EXP}$$

Απόδειξη:

- $\boldsymbol{P \subseteq NP}$ είναι προφανές, αφού κάθε ντετερμινιστική Μ.Τ. είναι εξ'ορισμού και μη ντετερμινιστική.
- $\boldsymbol{NP \subseteq EXP}$. Η απόδειξη στηρίζεται στην προσομοίωση μια μη ντετερμινιστικής Μ.Τ. Ν από μία ντετερμινιστική Μ ως εξής:
  - Η Ν είναι πολυωνυμικού χρόνου, άρα κάθε υπολογισμός της έχει πολυωνυμικό μήκος έστω $p=n^k$, όπου n το μέγεθος της εισόδου.
  - Κάθε υπολογισμός της Ν είναι μια ακολουθία από μη ντετερμινιστικές επιλογές. Αν είναι d ο βαθμός του μη ντετερμινισμού, τότε υπάρχουν d^p δυνατοί μη ντετερμινιστικοί υπολογισμοί.
  - Η Μ προσομοιώνει εξαντλητικά κάθε μη ντετερμινιστικό υπολογισμό διαπερνώντας όλο του δένδρο του μη ντετερμινιστικού υπολογισμού.
  - Συνεπώς ο χρόνος λειτουργίας της είναι $p \cdot d^p$, άρα εκθετικός
  - Συνεπώς $\boldsymbol{NP \subseteq EXP}$

Β. Θεωρία

3. Θεωρία Πολυπλοκότητας

5. NP-πληρότητα

Διαισθητικά σε μια κλάση προβλημάτων C ορίζουμε:

- C-πλήρη (C-Complete) τα προβλήματα της κλάσης που:
  - Είναι τα δυσκολότερα προβλήματα της κλάσης (υπό την έννοια ότι κάθε πρόβλημα της κλάσης είναι το πολύ τόσο δύσκολα όσο αυτά)
  - Είναι ισοδύναμα μεταξύ τους (δηλαδή αντίστοιχης υπολογιστικής δυσκολίας)

Έτσι για την κλάση NP, ορίζουμε ότι ένα πρόβλημα είναι **NP-πλήρες** (ή NP-Complete):

- Αν κάθε πρόβλημα στην κλάση NP, είναι το πολύ τόσο δύσκολο όσο αυτό.
- Και έχει αποδειχθεί από τον (Cook,1970) ότι:

  Το SAT είναι NP-πλήρες

- Συνεπώς οποιοδήποτε πρόβλημα του NP είναι το πολύ τόσο δύσκολο όσο το SAT!

Β. Θεωρία

4. NP-πληρότητα

1. Αποδείξεις NP-πληρότητας

Το ακόλουθο σκεπτικό είναι σημαντικό στα μαθήματα που ακολουθούν.

Για να αποδείξουμε ότι ένα πρόβλημα Π είναι NP-πλήρες, ακολουθούμε την εξής διαδικασία:

1. Αποδεικνύουμε ότι $\Pi \in NP$
   - Είτε δίνοντας μη ντετερμινιστική μηχανή Turing-μάντη που «μαντεύει» την λύση και έπειτα επαληθεύει ότι είναι όντως λύση του προβλήματος.
   - Είτε δίνοντας ντετερμινιστική μηχανή Turing-επαληθευτή που δεδομένης μιας λύσης (πιστοποιητικό) επαληθεύει σε πολυωνυμικό ντετερμινιστικό χρόνο ότι είναι λύση του προβλήματος.
2. Δίνουμε μια πολυωνυμική αναγωγή από ένα γνωστό NP-πλήρες πρόβλημα Π' στο πρόβλημα Π (Η αναγωγή συμβολίζεται με Π'≤Π)
   - Όπου δίνουμε έναν κανόνα μετασχηματισμού της εισόδου Ε' του γνωστού προβλήματος Π' σε είσοδο Ε του αγνώστου προβλήματος Π έτσι ώστε για κάθε στιγμιότυπο:

     Αποτέλεσμα του Π(Ε) **ισοδύναμο** με αποτέλεσμα του Π'(Ε΄)

     Και δείχνουμε ότι η κατασκευή θέλει πολυωνυμικό χρόνο

➢ Αν αποδείξουμε μόνο το 2ο σκέλος, τότε το πρόβλημα είναι NP-δύσκολο (NP-Hard)

Β. Θεωρία

4. NP-πληρότητα

1. Αποδείξεις NP-πληρότητας

Αρκετές δεκάδες χιλιάδες προβλημάτων από διαφορετικούς κλάδων των επιστημών έχουν αποδειχθεί NP-Complete

- Εμείς θα μελετήσουμε κάποιες οικογένειες προβλημάτων όπων:
  - Προβλήματα λογικής (π.χ. στο Μάθημα 6.2 θα δείξουμε ότι το πρόβλημα 3SAT είναι NP-πλήρες). Θα δώσουμε μία αναγωγή από το πρόβλημα SAT στο πρόβλημα 3SAT
  - Προβλήματα θεωρίας γράφων (π.χ. στο Μάθημα 6.3 θα δείξουμε ότι το πρόβλημα INDEPENDENT-SET είναι NP-πλήρες). Θα δώσουμε μία αναγωγή από το πρόβλημα 3SAT στο πρόβλημα INDEPENDENT-SET.
  - Προβλήματα θεωρίας συνόλων (π.χ. στο Μάθημα 6.5 θα δείξουμε ότι το πρόβλημα SET-COVER είναι NP-πλήρες). Θα δώσουμε μία αναγωγή από το πρόβλημα VERTEX-COVER στο πρόβλημα SET-COVER.

➢ Αν αποδείξουμε μόνο το 2ο σκέλος, τότε το πρόβλημα είναι NP-δύσκολο (NP-Hard)

Β. Θεωρία

4. NP-πληρότητα

2. Ιδιότητες NP-Complete Προβλημάτων

Όταν αποδείξουμε ότι ένα πρόβλημα είναι NP-Complete, τότε το πρόβλημα αυτό είναι εξίσου δύσκολο με το SAT και έχει τις ίδιες ιδιότητες με το SAT, δηλαδή:

- Δεν έχει ανακαλυφθεί πολυωνυμικός ντετερμινιστικός αλγόριθμος που το λύνει (δεν έχει αποδειχθεί ότι ανήκει στο P)
- Δεν έχει αποδειχθεί ότι δεν υπάρχει πολυωνυμικός ντετερμινιστικός αλγόριθμος που το λύνει (δεν έχει αποδειχθεί ότι δεν ανήκει στο P)
- Υπάρχει μη ντετερμινιστικός πολυωνυμικός αλγόριθμος που το λύνει (έχει αποδειχθεί ότι ανήκει στο NP)
- Υπάρχει ντετερμινιστικός εκθετικός αλγόριθμος που το λύνει (έχει αποδειχθεί ότι ανήκει στο EXP)

Β. Θεωρία

4. NP-πληρότητα

3. Το ανοικτό πρόβλημα P=NP

Συνεπώς έρχεται το μεγαλύτερο ανοικτό πρόβλημα της πληροφορικής σήμερα:

- Αν αποδειχθεί ότι οποιοδήποτε NP-πλήρες πρόβλημα λύνεται σε πολυωνυμικό χρόνο, τότε όλα τα προβλήματα της NP επίσης λύνονται σε πολυωνυμικό χρόνο. Συνεπώς $\boldsymbol{P = NP}$

- Αν αποδειχθεί ότι οποιοδήποτε NP-πλήρες πρόβλημα δεν λύνεται σε πολυωνυμικό χρόνο, τότε αυτό δεν θα ανήκει στο P, και θα ανήκει στο NP. Άρα $P \subset NP$ συνεπώς $\boldsymbol{P \neq NP}$

Άρα το μεγαλύτερο ανοικτό πρόβλημα διατυπώνεται ως εξής:

Ισχύει ότι $\boldsymbol{P = NP}$?

Του οποίου η λύση αμείβεται με 10^6\$ (www.claymath.org)

Β. Θεωρία

4. NP-πληρότητα

4. Η κλάση NP-Complete

Ορίζουμε:

Το σύνολο όλων των προβλημάτων που έχουν αποδειχθεί NP-πλήρη, συγκροτούν την κλάση προβλημάτων **NP-Complete**.

Συνοψίζοντας τα προβλήματα της κλάσης αυτής έχουν τις εξής ιδιότητες:

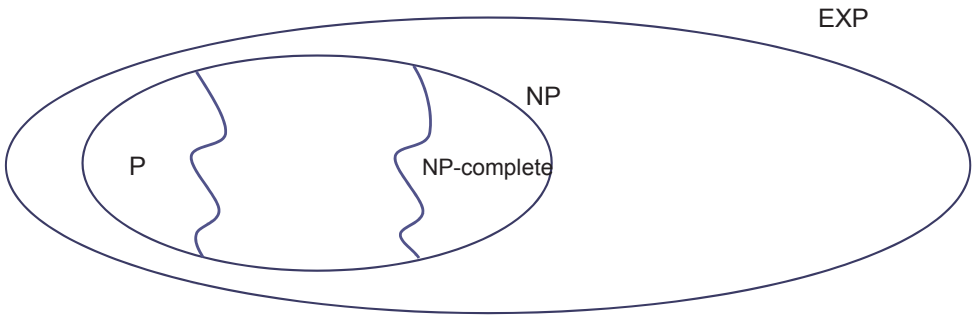
1. Λύνονται σε εκθετικό ντετερμινιστικό χρόνο (**ανήκουν στο EXP**)
2. Λύνονται σε πολυωνυμικό μη ντετερμινιστικό χρόνο (**ανήκουν στο NP**)
3. Δεν έχει αποδειχθεί ότι δεν λύνονται από ντετερμινιστικό πολυωνυμικό αλγόριθμο.
    1. Αν αποδειχθεί ότι ένα από αυτά δεν λύνεται σε πολυωνυμικό ντετερμινιστικό χρόνο, τότε κανένα δεν λύνεται σε ντετερμινιστικό πολυωνυμικό χρόνο
    2. Άρα $\boldsymbol{P \neq NP}$
4. Δεν έχει αποδειχθεί ότι λύνονται από ντετερμινιστικό πολυωνυμικό αλγόριθμο.
    1. Αν αποδειχθεί ότι ένα από αυτά λύνεται σε πολυωνυμικό ντετερμινιστικό χρόνο, τότε όλα λύνονται σε ντετερμινιστικό πολυωνυμικό χρόνο
    2. Άρα $\boldsymbol{P = NP}$
5. Όλα τα προβλήματα της κλάσης NP ανάγονται σε αυτά.

Β. Θεωρία

4. NP-πληρότητα

5. Ιεραρχία κλάσεων αν $P \neq NP$

Αν $\boldsymbol{P \neq NP}$ τότε ισχύει η εξής ιεραρχία των κλάσεων πολυπλοκότητας:



Έχουμε χρησιμοποιήσει τα σημεία της θεωρίας:

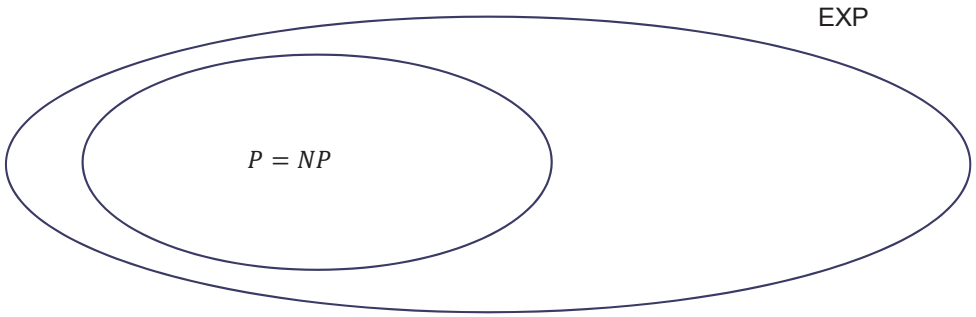
- $P \subseteq NP \subseteq EXP$
- $NP - complete \subseteq NP$

Β. Θεωρία

4. NP-πληρότητα

6. Ιεραρχία κλάσεων αν $P = NP$

Αν $\boldsymbol{P = NP}$ τότε ισχύει η εξής ιεραρχία των κλάσεων πολυπλοκότητας:



Γ. Ασκήσεις

Εφαρμογή 1

Ποιες από τις παρακάτω προτάσεις είναι: (α) Σωστές, (β) Λάθος (γ) Αληθές αν $P = NP$ (δ) Αληθές αν $P \neq NP$

Δίνεται ένα NP-πλήρες πρόβλημα Π:

1. Το Π δεν επιλύεται σε πολυωνυμικό χρόνο
2. Το Π επιλύεται σε πολυωνυμικό χρόνο.
3. Το Π λύνεται σε εκθετικό χρόνο.
4. Το Π λύνεται σε μη ντετερμινιστικό πολυωνυμικό χρόνο.

Γ. Ασκήσεις
Εφαρμογή 2

Ποιες από τις παρακάτω προτάσεις είναι: (α) Σωστές, (β) Λάθος (γ) Αληθές αν $P = NP$ (δ) Αληθές αν $P \neq NP$

Δίνεται ένα NP-πλήρες πρόβλημα Π:

1. Το Π ανάγεται στο SAT και αντίστροφα.
2. Υπάρχει αναγωγή από το πρόβλημα Shortest-Path στο πρόβλημα Π.
3. Το SAT είναι μη επιλύσιμο.
4. Αν το πρόβλημα Α είναι NP-πλήρες και υπάρχει αναγωγή από το πρόβλημα Β στο πρόβλημα Α, τότε και το Β είναι NP-πλήρες.

Γ. Ασκήσεις
Εφαρμογή 3

Ποιες από τις παρακάτω προτάσεις είναι: (α) Σωστές, (β) Λάθος (γ) Αληθές αν $P = NP$ (δ) Αληθές αν $P \neq NP$

1. Αν το πρόβλημα Α είναι NP-πλήρες και υπάρχει αναγωγή από το πρόβλημα Α στο πρόβλημα Β, τότε και το Β είναι NP-πλήρες.
2. Αν ένα πρόβλημα της κλάσης NP λύνεται σε πολυωνυμικό χρόνο, τότε κάθε πρόβλημα της κλάσης NP λύνεται σε πολυωνυμικό χρόνο.
3. Υπάρχει ντετερμινιστικός πολυωνυμικός αλγόριθμος για το πρόβλημα SAT
4. Δεν υπάρχει ντετερμινιστικός πολυωνυμικός αλγόριθμος για το πρόβλημα SAT.