Θεωρία Υπολογισμού

Αλγόριθμοι και Πολυπλοκότητα

2003-04

Αλγόριθμοι και Πολυπλοκότητα

- Υποθέστε ότι έχουμε δύο διαφορετικούς αλγόριθμους που λύνουν το ίδιο πρόβλημα. Ποιός αλγόριθμος είναι πιο γρήγορος;
- Υπάρχουν δύο τρόποι για να απαντήσουμε την παραπάνω ερώτηση:
  - Πειράματα με υποθετικά ή πραγματικά δεδομένα.
  - Μαθηματική ανάλυση χρησιμοποιώντας την έννοια της **χρονικής πολυπλοκότητας** ενός αλγόριθμου.
- Θα ασχοληθούμε **μόνο με χρονική πολυπλοκότητα**. Υπάρχει όμως και η **χωρική πολυπλοκότητα** που είναι εξίσου σημαντική.

Χρονική Πολυπλοκότητα

- Η χρονική πολυπλοκότητα ενός αλγόριθμου περιγράφεται από μια συνάρτηση του **μεγέθους της εισόδου.**
- **Ορισμός.** Η **χρονική πολυπλοκότητα χειρίστης περίπτωσης** ενός αλγόριθμου είναι μια συνάρτηση

$$T : \mathcal{N} \to \mathcal{N}$$

όπου $\mathcal{N}$ είναι το σύνολο των φυσικών αριθμών. Για κάθε φυσικό αριθμό n, η συνάρτηση T δίνει το μέγιστο αριθμό μονάδων χρόνου που απαιτείται ώστε να εκτελεστεί ο αλγόριθμος για οποιαδήποτε είσοδο μεγέθους n.

Χρονική Πολυπλοκότητα

- Συνήθως υποθέτουμε ότι κάθε βασική εντολή χρειάζεται σταθερό χρόνο για να εκτελεστεί π.χ. 1 χρονική μονάδα.
- Οι **βασικές εντολές** είναι συνήθως αναθέσεις, συγκρίσεις, εντολές εισόδου/εξόδου, εντολές επιστροφής, και εντολές εκχώρησης μνήμης.

Το Μέγεθος της Εισόδου

- Ποιά είναι κατάλληλη παράμετρος για τη μέτρηση του μεγέθους της εισόδου ενός αλγορίθμου;

Παραδείγματα:

- Ταξινόμηση μιας λίστας ακεραίων: το μήκος της λίστας.
- Πολλαπλασιασμός δύο δυαδικών αριθμών: το μήκος της δυαδικής τους αναπαράστασης.
- Εύρεση μονοπατιού σε γράφο: ο αριθμός των κορυφών και ο αριθμός των ακμών του γράφου.

Παράδειγμα

- **function** SUMMATION(*sequence*) **returns** an integer
  local *sequence*: array of integers, *sum*: integer
  $sum \leftarrow 0$
  for $i \leftarrow 1$ **to** LENGTH(*sequence*) **do**
  $sum \leftarrow sum + sequence[i]$
  end
  return *sum*

- **Μέγεθος εισόδου:** n, το μήκος της ακολουθίας.
- **Χρονική Πολυπλοκότητα:** $T(n) = 3n + 2$.

Παράδειγμα

- **function** FIND_13(*sequence*)
  returns the position in the sequence containing the number 13
  local *sequence*: array of integers, *pos*: integer
  $pos \leftarrow 1$
  while $sequence[pos] \neq 13$ **and** $pos \leq$ LENGTH(*sequence*) **do**
  $pos \leftarrow pos + 1$
  end
  if $pos \leq$ LENGTH(*sequence*) **then return** *pos*
  else return -1

- **Μέγεθος εισόδου:** n, το μήκος της ακολουθίας.
- **Χρονική Πολυπλοκότητα:** $T(n) = 3n + 3$.

Ρυθμός Αύξησης

- Όταν υπολογίζουμε την χρονική πολυπλοκότητα χειρίστης περίπτωσης ενός αλγορίθμου μας ενδιαφέρει ο **ρυθμός αύξησης** της, και όχι οι λεπτομέρειες.
- Για παράδειγμα, στην περίπτωση της χρονικής πολυπλοκότητας $T(n) = 3n^2 + 5n + 2$, ο **πιο σημαντικός όρος** είναι ο $3n^2$. Για μεγάλες τιμές του n, οι όροι $5n$ και 2 είναι σχετικά ασήμαντοι συγκρινόμενοι με τον όρο $3n^2$. Μπορούμε επίσης να αγνοήσουμε τον συντελεστή 3 και να θεωρήσουμε το n^2 σαν τον πιο σημαντικό όρο της $T(n)$.
- Αυτή η ανάλυση λέγεται **ασυμπτωτική ανάλυση αλγορίθμων**.

Ο συμβολισμός $O(\cdot)$

- **Ορισμός.** Έστω $f : \mathcal{N} \to \mathcal{N}$ μια συνάρτηση. Η **τάξη (order)** της f είναι το σύνολο όλων των συναρτήσεων $g : \mathcal{N} \to \mathcal{N}$ με την ιδιότητα ότι υπάρχουν φυσικοί αριθμοί $c > 0$ και $d \geq 0$ τέτοιοι ώστε, για όλα τα $n \in \mathcal{N}$, $g(n) \leq cf(n) + d$.
- Η τάξη της f συμβολίζεται με $O(f)$.
- Αν για δύο συναρτήσεις f και g έχουμε $f \in O(g)$ και $g \in O(f)$, τότε γράφουμε $f \asymp g$.
- Η σχέση $\asymp$ είναι **σχέση ισοδυναμίας**, άρα διαμερίζει το σύνολο όλων των συναρτήσεων $f : \mathcal{N} \to \mathcal{N}$ σε κλάσεις ισοδυναμίας.
- Η κλάση ισοδυναμίας μια συνάρτησης f ως προς τη σχέση $\asymp$ λέγεται **ρυθμός αύξησης (rate of growth)** της f.

Παράδειγμα

- Θεωρείστε το πολυώνυμο $f(n) = 31n^2 + 17n + 3$. Είναι $f \in O(n^2)$ επειδή $f(n) \leq 48n^2 + 3$. Επίσης $n^2 \in O(f(n))$ επειδή $n^2 \leq f(n) + 0$.
- Αρα $n^2 \asymp 31n^2 + 17n + 3$.
- Γενικά, κάθε πολυώνυμο της μορφής

$$f(n) = a_d n^d + a_{d-1} n^{d-1} + \cdots + a_1 n + a_0$$

όπου $a_i \geq 0$ για κάθε i, και $a_d > 0$, ισχύει ότι $f \asymp O(n^d)$.

- **Πρόταση.** Όλα τα πολυώνυμα με τον ίδιο βαθμό έχουν τον ίδιο ρυθμό αύξησης. Αν το πολυώνυμο g έχει μεγαλύτερο βαθμό από το πολυώνυμο f τότε ο ρυθμός αύξησης του g είναι μεγαλύτερος από τον ρυθμό αύξησης του f.

Εκθετικός Ρυθμός Αύξησης

- Έστω η συνάρτηση $f(n) = 2^n$. Η f έχει ρυθμό αύξησης **μεγαλύτερο από κάθε πολυώνυμο**.
- Αλλες συναρτήσεις όπως $5^n, n^n, n!, 2^{n^2}, 2^{2^n}$ έχουν ακόμα μεγαλύτερο ρυθμό αύξησης.

Χρονική Πολυπλοκότητα

- Όταν υπολογίζουμε χρονική πολυπλοκότητα χειρίστης περίπτωσης, μας ενδιαφέρει μόνο η **ρυθμός αύξησης** της συνάρτησης T.
- Η χρονική πολυπλοκότητα δίνεται στη μορφή $O(T)$ όπου T είναι η απλούστερη συνάρτηση από όλες τις συναρτήσεις που έχουν τον ίδιο ρυθμό αύξησης.
- **Παράδειγμα:** $O(n^2)$ αντί για $O(5n^2 + 3)$, ή $O(2^n)$ αντί για $O(3 \cdot 2^n + 17)$.

Παράδειγμα

- **function** SUMMATION(*sequence*) **returns** an integer
  local *sequence*: array of integers, *sum*: integer
  $sum \leftarrow 0$
  for $i \leftarrow 1$ **to** LENGTH(*sequence*) **do**
  $sum \leftarrow sum + sequence[i]$
  end
  return *sum*

- **Μέγεθος εισόδου:** n, το μήκος της ακολουθίας.
- **Χρονική Πολυπλοκότητα:** $O(n)$

Παράδειγμα

- **function** FIND_13(*sequence*)
  returns the position in the sequence containing the number 13
  local *sequence*: array of integers, *pos*: integer
  $pos \leftarrow 1$
  while $sequence[pos] \neq 13$ **and** $pos \leq$ LENGTH(*sequence*) **do**
  $pos \leftarrow pos + 1$
  end
  if $pos \leq$ LENGTH(*sequence*) **then return** *pos*
  else return -1

- **Μέγεθος εισόδου:** n, το μήκος της ακολουθίας.
- **Χρονική Πολυπλοκότητα:** $O(n)$

Παράδειγμα

- **Αλγόριθμος** NDFAtoDFA
  Είσοδος: Μη ντετερμινιστικό πεπερασμένο αυτόματο $M = (K, \Sigma, \Delta, s, F)$
  Έξοδος: Ντετερμινιστικό πεπερασμένο αυτόματο $M' = (K', \Sigma, \delta', s', F')$
  Μέθοδος:

$$K' = 2^K, \ s' = E(s)$$
$$F' = \{Q \subseteq K : \ Q \cap F \neq \emptyset\}$$
$$\delta'(Q, \sigma) = \bigcup \{E(p) : \ p \in K \ kai \ (q, \sigma, p) \in \Delta \ gia \ k'apoio \ q \in Q\}$$

Αν q είναι μια κατάσταση του K, τότε το $E(q)$ είναι το σύνολο όλων των καταστάσεων στις οποίες μπορεί να μεταβεί το M από την q χωρίς να διαβάσει κανένα σύμβολο (δηλ. να διαβάσει μόνο e):

$$E(q) = \{p \in K : \ (q, e) \vdash_M^* (p, e)\}$$

Παράδειγμα

- Για την περίπτωση ενός μη ντετερμινιστικού πεπερασμένου αυτόματου $M = (K, \Sigma, \Delta, s, F)$, το **μέγεθος** του M δίδεται από τις εξής παραμέτρους:
  - Πληθικός αριθμός του συνόλου K.
  - Πληθικός αριθμός του συνόλου Σ.
  - Πληθικός αριθμός του συνόλου Δ.

Παράδειγμα

- Η υπολογιστική πολυπλοκότητα του NDFAtoDFA μπορεί να υπολογιστεί ως εξής.
- Ο υπολογισμός του συνόλου K' έχει χρονική πολυπλοκότητα $O(|K|2^{|K|})$.
- Για κάθε κατάσταση $q \in K$, ο υπολογισμός του συνόλου $E(q)$ έχει χρονική πολυπλοκότητα $O(|K| + |\Delta|)$ αν το αυτόματο M παριστάνεται από ένα γράφο με $|K|$ κορυφές και $|\Delta|$ ακμές. Αρα ο υπολογισμός της s' έχει χρονική πολυπλοκότητα $O(|K| + |\Delta|)$.

Παράδειγμα

- Ο υπολογισμός του συνόλου F' έχει χρονική πολυπλοκότητα $O(|K|^2 2^{|K|})$.
- Ο υπολογισμός της δ έχει χρονική πολυπλοκότητα

$$O(2^{|K|}|K|^4|\Delta|^2|\Sigma|).$$

- Τελικά η χρονική πολυπλοκότητα του αλγόριθμου είναι

$$O(2^{|K|}|K|^4|\Delta|^2|\Sigma|).$$

Υπολογιστική Πολυπλοκότητα

- Πολλά προβλήματα, αν και μπορούν να λυθούν θεωρητικά από κάποιο αλγόριθμο, στην πράξη ο αλγόριθμος αυτός παίρνει **πάρα πολύ χρόνο**.

Παραδείγματα:

- Το πρόβλημα μετατροπής ενός μη ντετερμινιστικού αυτομάτου σε ντετερμινιστικό.
- Το πρόβλημα του πλανόδιου πωλητή (traveling salesman problem, TSP).
- ...

Το πρόβλημα TSP

- Δίνεται ένα σύνολο $\{c_1, \ldots, c_n\}$ από πόλεις, και ένας $n \times n$ πίνακας μη αρνητικών ακεραίων, όπου ο ακέραιος d_{ij} δηλώνει την απόσταση μεταξύ των πόλεων c_i και c_j.
- Ζητείται να βρούμε το συντομότερο δρομολόγιο που καλύπτει όλες τις πόλεις δηλ. την 1-1 και επί αντιστοιχία

$$\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$$

(όπου π_i είναι διαισθητικά η i-οστή πόλη του δρομολογίου) που ελαχιστοποιεί την ποσότητα

$$c(\pi) = d_{\pi(1)\pi(2)} + d_{\pi(2)\pi(3)} + \cdots + d_{\pi(n-1)\pi(n)} + d_{\pi(n)\pi(1)}.$$

Το πρόβλημα TSP

- Τα δυνατά δρομολόγια του πωλητή είναι $n!$.
- Η χρονική πολυπλοκότητα ενός απλού αλγόριθμου που εξετάζει όλα τα δρομολόγια θα είναι $O(n!)$. Μπορεί ένας τέτοιος αλγόριθμος να θεωρηθεί **πρακτικός**;
- Ποιοί αλγόριθμοι μπορούν να θεωρηθούν πρακτικοί;

Πολυωνυμικοί Αλγόριθμοι

- Οι αλγόριθμοι που έχουν **πολυωνυμική χρονική πολυπλοκότητα** (δηλ. $O(n^k)$, όπου n είναι το **μήκος** της εισόδου και k μια σταθερά) μπορούν να θεωρηθούν **πρακτικοί αλγόριθμοι**.
- Το θεωρητικό μοντέλο υπολογισμού που χρησιμοποιούμε για να μελετήσουμε τους πολυωνυμικούς αλγόριθμους είναι η πρότυπη μηχανή Turing.
- Επίσης μπορούμε να χρησιμοποιήσουμε οποιαδήποτε επέκταση της μηχανής Turing που μπορεί να προσομοιωθεί από την πρότυπη μηχανή με απώλεια αποδοτικότητας που δεν υπερβαίνει ένα πολυώνυμο (άρα **αποκλείονται οι μη ντετερμινιστικές μηχανές Turing**).

Κριτική

- Είναι πρακτικός ένας αλγόριθμος με πολυπλοκότητα n^{100} ή $10^{100}n^2$;
- Γιατί να μην θεωρούμε πρακτικό ένα αλγόριθμο με πολυπλοκότητα $n^{\log\log n}$;
- Γιατί να μην χρησιμοποιούμε πολυπλοκότητα μέσης περίπτωσης (average case complexity);

Η κλάση $\mathcal{P}$

- **Ορισμός.** Μια μηχανή Turing $M = (K, \Sigma, \delta, s, H)$ λέγεται **πολυωνυμικά φραγμένη** αν υπάρχει ένα πολυώνυμο $p(n)$ τέτοιο ώστε ισχύει η ακόλουθη πρόταση: Για κάθε συμβολοσειρά εισόδου x, δεν υπάρχει configuration C τέτοιο ώστε

$$(s, \triangleright \sqcup x) \vdash_M^{p(|x|)+1} C.$$

Με άλλα λόγια, η μηχανή τερματίζει πάντα μετά από το πολύ $p(n)$ βήματα όπου n είναι το μήκος της εισόδου.

- Μια γλώσσα λέγεται **πολυωνυμικά αποφασίσιμη** αν και μόνο αν υπάρχει μια πολυωνυμικά φραγμένη μηχανή Turing που την αποφασίζει.
- Η κλάση όλων των πολυωνυμικά αποφασίσιμων γλώσσων παριστάνεται από το σύμβολο $\mathcal{P}$.

Η κλάση $\mathcal{P}$

- Η κλάση $\mathcal{P}$ αντιπροσωπεύει όλους τους πρακτικούς αλγόριθμους και όλα τα προβλήματα που μπορούν να λυθούν αποδοτικά.
- **Θεώρημα.** Η κλάση $\mathcal{P}$ είναι κλειστή ως προς την πράξη του συμπληρώματος.

Η κλάση $\mathcal{P}$

- **Θεώρημα.** Έστω η αναδρομική γλώσσα

$$E = \{\text{“}M\text{”}\,\text{“}w\text{”} : \text{ η } M \text{ δέχεται την είσοδο } w \text{ μετά από το πολύ } 2^{|\,\text{“}w\text{”}\,|} \text{ βήματα}\}.$$

Η γλώσσα E δεν ανήκει στην κλάση $\mathcal{P}$.

Απόδειξη

- Θα χρησιμοποιήσουμε την τεχνική της διαγωνιοποίησης.
- Υποθέτουμε ότι $E \in \mathcal{P}$. Τότε $E_1 \in \mathcal{P}$ όπου

$$E_1 = \{\text{“}M\text{”} : \text{ η } M \text{ δέχεται την είσοδο “}M\text{” μετά από το πολύ } 2^{|\,\text{“}M\text{”}\,|} \text{ βήματα}\}.$$

Επίσης $\overline{E_1} \in \mathcal{P}$ όπου

$$\overline{E_1} = \{\text{“}M\text{”} : \text{ η } M \text{ δεν δέχεται την είσοδο “}M\text{” μετά από το πολύ } 2^{|\,\text{“}M\text{”}\,|} \text{ βήματα}\}.$$

Απόδειξη

- Υπάρχει πολυωνυμικά φραγμένη μηχανή Turing M_1 που αποφασίζει την $\overline{E_1}$. Δηλ. υπάρχει πολυώνυμο $p(n)$ έτσι ώστε:
  - η M_1 τερματίζει πάντα μετά από το πολύ $p(n)$ βήματα όπου n είναι το μήκος της εισόδου.
  - η M_1 δέχεται όλες τις αναπαραστάσεις μηχανών Turing που δεν δέχονται τις αναπαραστάσεις του εαυτού τους μετά από το πολύ 2^n βήματα.
- Παρατηρήστε ότι υπάρχει θετικός ακέραιος n_0 τέτοιος ώστε

$$p(n) \leq 2^n, \text{ για κάθε } n \geq n_0.$$

- Υποθέτουμε ότι το μήκος της αναπαράστασης της M_1 είναι τουλάχιστον n_0.

Απόδειξη

- **Ερώτηση.** Δέχεται ή απορρίπτει η M_1 την αναπαράσταση “M_1” του εαυτού της;
- Αν η M_1 δέχεται την “M_1”, τότε “M_1” $\in \overline{E_1}$. Αρα η M_1 δεν δέχεται την “M_1” μετά από το πολύ $2^{|\,\text{“}M_1\text{”}\,|}$ βήματα.
- Όμως η M_1 τερματίζει πάντα μετά από $p(n)$ βήματα όπου n είναι το μήκος της εισόδου, και $2^{|\,\text{“}M_1\text{”}\,|} \geq p(|\,\text{“}M_1\text{”}\,|)$. Δηλ. η M_1 δεν δέχεται την “M_1”. Αντίφαση!
- Όμοια καταλήγουμε σε αντίφαση αν υποθέσουμε ότι η M_1 δεν δέχεται την “M_1”.
- Η υπόθεση που μας οδήγησε σε αντίφαση είναι η $E \in \mathcal{P}$. Αρα $E \notin \mathcal{P}$.

Προβλήματα Απόφασης (Decision Problems)

- HALTING PROBLEM: Έστω μηχανή Turing M και συμβολοσειρά εισόδου w. Δέχεται η M την w;
- REACHABILITY: Έστω κατευθυνόμενος γράφος $G \subseteq V \times V$, όπου $V = \{v_1, \ldots, v_n\}$ είναι ένα πεπερασμένο σύνολο κορυφών, και $v_i, v_j \in V$. Υπάρχει μονοπάτι ανάμεσα στις κορυφές v_i και v_j;

Αναπαραστάσεις Προβλημάτων Απόφασης

- Συνήθως αναπαριστάνουμε προβλήματα απόφασης με κατάλληλες γλώσσες. Για παράδειγμα:
- HALTING PROBLEM:

$$H = \{\text{“}M\text{”}\,\text{“}w\text{”} : \text{ η μηχανή Turing } M \text{ τερματίζει με είσοδο } w \ \}$$

- REACHABILITY:

$$R = \{\kappa(G)\mathbf{b}(i)\mathbf{b}(j) : \text{ υπάρχει ένα μονοπάτι από την κορυφή } v_i \text{ στην κορυφή } v_j \text{ του γράφου } G$$

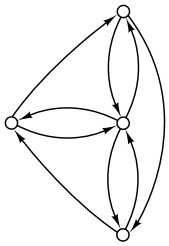
όπου $\mathbf{b}(i)$ είναι η δυαδική αναπαράσταση του ακέραιου i, και $\kappa(G)$ είναι μια αναπαράσταση του γράφου G από μια συμβολοσειρά (π.χ. ο G μπορεί να παρασταθεί από μια συμβολοσειρά που κωδικοποιεί τον πίνακα γειτνίασης του G).

Προβλήματα Απόφασης

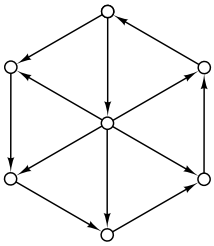
- Έστω αλφάβητο Σ και γλώσσα $L \subseteq \Sigma^*$:
  Πρόβλημα απόφασης για μια γλώσσα (DECISION PROBLEM FOR L). Έστω συμβολοσειρά $x \in \Sigma^*$. Ανήκει η x στην L;
- Το πρόβλημα REACHABILITY ανήκει στην κλάση $\mathcal{P}$.
  Μπορεί να λυθεί από ένα depth-first-search αλγόριθμο σε χρόνο $O(|\,V\,| + |\,E\,|)$ (V είναι το σύνολο των κορυφών και E είναι το σύνολο των ακμών του γράφου G.).
- EULERIAN CYCLE: Έστω γράφος G. Είναι ο G γράφος του Euler; Δηλ. υπάρχει στον G ένα κλειστό μονοπάτι που χρησιμοποιεί κάθε ακμή ακριβώς μια φορά;
  Το μονοπάτι του προβλήματος μπορεί να περνάει από κάθε κορυφή πολλές φορές (ή και καθόλου αν η κορυφή είναι απομονωμένη).

Παράδειγμα

- Ο γράφος (a) είναι γράφος του Euler ενώ ο (b) δεν είναι.



(a)



(b)

Γράφοι του Euler

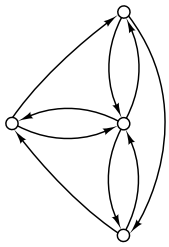
- **Θεώρημα.** Ένας γράφος $G \subseteq V \times V$ είναι γράφος του Euler αν και μόνο αν έχει τις ακόλουθες ιδιότητες:
  - (a)] Για κάθε ζευγάρι κορυφών $u, v \in V$ καμιά από τις οποίες δεν είναι απομονωμένη, υπάρχει μονοπάτι από την u στην v.
  - Όλες οι κορυφές έχουν ίσο αριθμό εισερχόμενων και εξερχόμενων ακμών.
- Το πρόβλημα EULERIAN CYCLE ανήκει στην κλάση $\mathcal{P}$ επειδή οι παραπάνω ιδιότητες μπορούν να ελεχθούν σε πολυωνυμικό χρόνο.

Αλλα Ενδιαφέροντα Προβλήματα

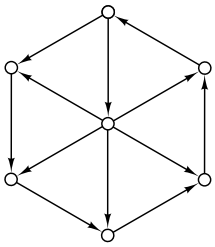
- HAMILTON CYCLE: Έστω γράφος G. Ειναι ο G γράφος του Hamilton; Δηλ. υπάρχει ένα κύκλος που περνάει από όλες τις κορυφές ακριβώς μια φορά;
- Κύκλοι όπως τους παραπάνω λέγονται κύκλοι του Hamilton.
- Δεν γνωρίζουμε αν το πρόβλημα HAMILTON CYCLE ανήκει στην κλάση $\mathcal{P}$. Μέχρι σήμερα κανείς δεν έχει βρεί πολυωνυμικό αλγόριθμο γι αυτό το πρόβλημα.

Παράδειγμα

- Οι παρακάτω γράφοι είναι γράφοι του Hamilton.



(a)



(b)

Προβλήματα Βελτιστοποίησης

- TSP: Δίνεται ένα σύνολο $\{c_1, \ldots, c_n\}$ από πόλεις, και ένας $n \times n$ πίνακας μη αρνητικών ακεραίων, όπου ο ακέραιος d_{ij} δηλώνει την απόσταση μεταξύ των πόλεων c_i και c_j.
- Ζητείται να βρούμε το συντομότερο δρομολόγιο που καλύπτει όλες τις πόλεις δηλ. την 1-1 και επί αντιστοιχία

$$\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$$

(όπου π_i είναι διαισθητικά η i-οστή πόλη του δρομολογίου) που ελαχιστοποιεί την ποσότητα

$$c(\pi) = d_{\pi(1)\pi(2)} + d_{\pi(2)\pi(3)} + \cdots + d_{\pi(n-1)\pi(n)} + d_{\pi(n)\pi(1)}.$$

Προβλήματα Βελτιστοποίησης

- Κάθε πρόβλημα βελτιστοποίησης μπορεί να μετατραπεί σε ένα πρόβλημα απόφασης εισάγοντας άλλη μια παράμετρο B σαν φράγμα στη συνάρτηση κόστους. Για παράδειγμα, το πρόβλημα TSP μπορεί να εκφραστεί σαν πρόβλημα απόφασης ως εξής.
- TSP: Έστω ακέραιος $n \geq 2$, ένας $n \times n$ πίνακας αποστάσεων d_{ij}, και ένας ακέραιος $B \geq 0$. Υπάρχει 1-1 και επί αντιστοιχία

$$\pi : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, n\}$$

τέτοια ώστε

$$c(\pi) = d_{\pi(1)\pi(2)} + d_{\pi(2)\pi(3)} + \cdots + d_{\pi(n-1)\pi(n)} + d_{\pi(n)\pi(1)} \leq B;$$

είναι ελάχιστη.
- Αν το προβλήμα απόφασης TSP δεν μπορεί να λυθεί σε πολυωνυμικό χρόνο, τότε ούτε το πρόβλημα βελτιστοποίησης TSP μπορεί!

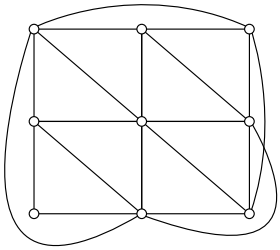
Προβλήματα Βελτιστοποίησης

- INDEPENDENT SET: Έστω μη κατευθυνόμενος γράφος $G \subseteq V \times V$ και ακέραιος $K \geq 2$. Υπάρχει $C \subseteq V$ με $|C| \geq K$ τέτοιο ώστε για όλες τις κορυφές $v_i, v_j \in C$, δεν υπάρχει ακμή που να συνδέει τις v_i και v_j;
- CLIQUE: Έστω μη κατευθυνόμενος γράφος $G \subseteq V \times V$ και ακέραιος $K \geq 2$. Υπάρχει $C \subseteq V$ με $|C| \geq K$ τέτοιο ώστε για όλες τις κορυφές $v_i, v_j \in C$, υπάρχει ακμή που συνδέει τις v_i και v_j;

Προβλήματα Βελτιστοποίησης

- VERTEX COVER: Έστω μη κατευθυνόμενος γράφος $G \subseteq V \times V$ και ακέραιος $K \geq 2$. Υπάρχει $C \subseteq V$ με $|C| \leq K$ τέτοιο ώστε το C καλύπτει όλες τις ακμές του G;

Παράδειγμα



Το Πρόβλημα partition

- PARTITION: Έστω ένα σύνολο n θετικών ακεραίων $a_1, \ldots, a_n$ που παριστάνονται στο δυαδικό σύστημα. Υπάρχει $P \subseteq \{1, \ldots, n\}$ τέτοιο ώστε $\sum_{i \in P} a_i = \sum_{i \notin P} a_i$;
- Έστω $H = (\sum_{i=1}^{n} a_i)/2$. Αν ο H δεν είναι ακέραιος τότε η απάντηση στο πρόβλημα είναι "όχι".
- Αν ο H είναι ακέραιος, ορίζουμε τα σύνολα $B(i)$ για κάθε i, $0 \leq i \leq n$:

$$B(i) = \{b \leq H : \ b \text{ είναι το άθροισμα των στοιχείων κάποιου}$$

$$\text{υποσυνόλου του } \{a_1, \ldots, a_n\} \}$$

Αν ξέρουμε το σύνολο $B(n)$ μπορούμε να λύσουμε το πρόβλημα PARTITION απλώς ελέγχοντας αν $H \in B(n)$.

Το Πρόβλημα partition

- Ο παρακάτω αλγόριθμος υπολογίζει το $B(n)$:
  $B(0) := \{0\}$
  for $i = 1, 2, \ldots, n$ do
    $B(i) := B(i-1)$
    for $j = a_i, a_{i+1}, a_{i+2}, \ldots, H$ do
      if $j - a_i \in B(i-1)$ then add j to $B(i)$
  Η χρονική πολυπλοκότητα του παραπάνω αλγόριθμου είναι $O(nH)$.
  Είναι αυτός ο αλγόριθμος πολυωνυμικός;

Το Πρόβλημα partition

- Αν κάθε ακέραιος της εισόδου παριστάνεται από το πολύ m bits (δηλαδή είναι το πολύ 2^{m-1}), τότε

$$H \leq n \cdot 2^{m-1}.$$

Αρα ο παραπάνω αλγόριθμος **δεν είναι πολυωνυμικός!**

Το Πρόβλημα unary partition

- UNARY PARTITION: Έστω ένα σύνολο n θετικών ακεραίων $a_1, \ldots, a_n$ που παριστάνονται στο μοναδιαίο σύστημα. Υπάρχει $P \subseteq \{1, \ldots, n\}$ τέτοιο ώστε $\sum_{i \in P} a_i = \sum_{i \notin P} a_i$;
- Τώρα ο H είναι $O(n)$ άρα το πρόβλημα UNARY PARTITION ανήκει στη κλάση $\mathcal{P}$ (λύνεται σε χρόνο $O(n^2)$)!
- **Σύμβαση:** Υποθέτουμε ότι οι ακέραιοι θα παριστάνονται πάντα στο δυαδικό σύστημα.
- Θα μπορούσαμε επίσης να χρησιμοποιήσουμε το δεκαδικό σύστημα, το δεκαεξαδικό ή οποιοδήποτε άλλο σύστημα αρίθμησης με βάση > 1. Οι αναπαραστάσεις ενός ακεραίου σ' αυτά τα συστήματα σχετίζονται μεταξύ τους με γραμμικό τρόπο.

Πεπερασμένα Αυτόματα και Κανονικές Εκφράσεις

- EQUIVALENCE OF DETERMINISTIC FINITE AUTOMATA: Έστω δύο ντετερμινιστικά πεπερασμένα αυτόματα M_1 και M_2. Είναι $L(M_1) = L(M_2)$;
- Το παραπάνω πρόβλημα ανήκει $\mathcal{P}$. Μπορούμε να βρούμε (σε πολυωνυμικό χρόνο) τα ισοδύναμα αυτόματα με ελάχιστο αριθμό καταστάσεων και μετά να τα συγκρίνουμε.

Πεπερασμένα Αυτόματα και Κανονικές Εκφράσεις

- Δεν ξέρουμε αν τα παρακάτω προβήματα ανήκουν στην κλάση $\mathcal{P}$:
- EQUIVALENCE OF NON DETERMINISTIC FINITE AUTOMATA: Έστω δύο μη ντετερμινιστικά πεπερασμένα αυτόματα M_1 και M_2. Είναι $L(M_1) = L(M_2)$;
- EQUIVALENCE OF REGULAR EXPRESSIONS: Έστω δύο κανονικές εκφράσεις R_1 και R_2. Είναι $L(R_1) = L(R_2)$;

Προτασιακή Λογική (Λογική του Boole)

- Έστω $Q = \{x_1, x_2, \ldots, x_n\}$ ένα πεπερασμένο σύνολο μεταβλητών που μπορούν να πάρουν τιμές $\top$ (**true**) και $\bot$ (**false**).
- Για κάθε μεταβλητή $x \in X$, η **άρνηση** της x παριστάνεται με $\bar{x}$. Επίσης ορίζουμε το σύνολο $\bar{Q} = \{\bar{x} : \ x \in X\}$.
- Τα στοιχεία του συνόλου $X \cup \bar{X}$ ονομάζονται **literals** (**θετικά literals** ή **αρνητικά literals** ανάλογα με το αν ανήκουν στο X ή στο $\bar{X}$).
- Ένα **clause** C είναι ένα μη κενό σύνολο $\{l_1, l_2, \ldots, l_n\}$ από literals.
- Κάθε clause παριστάνει μια **διάζευξη** δηλ. αν $C = \{l_1, l_2, \ldots, l_n\}$, τότε ισοδύναμα μπορούμε να γράφουμε

$$C \ = \ l_1 \vee l_2 \vee \cdots \vee l_n.$$

Προτασιακή Λογική (Λογική του Boole)

- Ένας **τύπος του Boole σε συζευκτική κανονική μορφή (Boolean formula in conjunctive normal form)** είναι ένα σύνολο από clauses.
- Αν
$$F = \{ \{l_{11}, \ldots, l_{1n_1}\}, \{l_{21}, \ldots, l_{2n_2}\}, \ldots, \{l_{m1}, \ldots, l_{mn_m}\} \}$$
είναι ένας τύπος σε συζευκτική κανονική μορφή, τότε ισοδύναμα μπορούμε να γράφουμε
$$F = (l_{11} \vee \cdots \vee l_{1n_1}) \wedge (l_{21} \vee \cdots \vee l_{2n_2}) \wedge \cdots \wedge (l_{m1} \vee \cdots \vee l_{mn_m}).$$

Παράδειγμα

- Έστω $Q = \{x_1, x_2, x_3\}$.
- Το σύνολο $\{x_1, \bar{x_2}, x_3\}$ είναι ένα clause που μπορεί επίσης να γραφεί σαν $x_1 \vee \bar{x_2} \vee x_3$.
- Το σύνολο
$$\{x_1 \vee \bar{x_2} \vee x_3,\ \bar{x_1},\ x_2 \vee \bar{x_2}\}$$
είναι ένας τύπος του Boole που μπορεί επίσης να γραφεί σαν
$$(x_1 \vee \bar{x_2} \vee x_3) \wedge \bar{x_1} \wedge (x_2 \vee \bar{x_2}).$$

Η Προτασιακή Λογική - Σημασιολογία

- Έστω F ένας τύπος του Boole σε συζευκτική κανονική μορφή με σύνολο μεταβλητών $Q = \{x_1, x_2, \ldots, x_n\}$.
- Μια **ανάθεση τιμών αλήθειας (truth assignment)** T για την F είναι μια συνάρτηση
$$T : X \to \{\top, \bot\}.$$
- Μια ανάθεση τιμών αλήθειας T **ικανοποιεί (satisfies)** τον τύπο F αν για κάθε clause $C \in F$ υπάρχει τουλάχιστον μια μεταβλητή x_i τέτοια ώστε
  - $T(x_i) = \top$ και $x_i \in C$, ή
  - $T(x_i) = \bot$ και $\bar{x_i} \in C$.
- Ένας τύπος του Boole λέγεται **ικανοποιήσιμος (satisfiable)** αν υπάρχει μια ανάθεση τιμών αλήθειας που τον ικανοποιεί.

Παράδειγμα

- Έστω ο τύπος του Boole
$$F = \{x_1 \vee \bar{x_2} \vee x_3,\ \ \bar{x_1},\ \ x_2 \vee \bar{x_2}\}.$$
- Η ανάθεση τιμών αλήθειας T με $T(x_1) = \top, T(x_2) = \top$ και $T(x_3) = \top$ δεν ικανοποιεί τον F.
- Η ανάθεση τιμών αλήθειας T με $T(x_1) = \bot, T(x_2) = \top$ και $T(x_3) = \top$ ικανοποιεί τον F.
- Ο τύπος F είναι ικανοποιήσιμος.

Παράδειγμα

- Έστω ο τύπος του Boole
$$F' = \{x_1 \vee x_2 \vee x_3,\ \ \bar{x_1} \vee x_2,\ \ \bar{x_2} \vee x_3,\ \ \bar{x_3} \vee x_1,\ \ \bar{x_1} \vee \bar{x_2} \vee \bar{x_3}\}.$$
Δεν υπάρχει ανάθεση ανάθεση τιμών αλήθειας που να ικανοποιεί τον F'.
- Ο τύπος F' δεν είναι ικανοποιήσιμος.

Το Πρόβλημα satisfiability

- SATISFIABILITY: Έστω τύπος του Boole F. Είναι ο F ικανοποιήσιμος;
- Δεν γνωρίζουμε αν υπάρχει πολυωνυμικός αλγόριθμος για το πρόβλημα SATISFIABILITY.

Το Πρόβλημα 2-satisfiability

- 2-SATISFIABILITY: Έστω τύπος του Boole F με την ιδιότητα ότι κάθε clause έχει το πολύ δύο literals. Είναι ο F ικανοποιήσιμος;
- Υπάρχει πολυωνυμικός αλγόριθμος για το πρόβλημα 2-SATISFIABILITY. Αρα το πρόβλημα 2-SATISFIABILITY ανήκει στην κλάση $\mathcal{P}$.

Αλγόριθμος για το Πρόβλημα 2-satisfiability

- Algorithm 2-SAT(F)
  - Αν ο F περιέχει δύο clauses $\{x\}$ και $\{\bar{x}\}$ τότε δεν είναι ικανοποιήσιμος.
  - Επανέλαβε τα παρακάτω βήματα μέχρι ο F να μην περιέχει κανένα clause που έχει μόνο ένα literal. Η διαδικασία αυτή λέγεται διαδικασία **εκκαθάρισης.**
    - Διάλεξε ένα clause C που έχει μόνο ένα literal. Αν $C = \{x_i\}$, τότε βάλε $T(x_i) = \top$ και απλοποίησε τον τύπο F. Αν $C = \{\bar{x_i}\}$, τότε βάλε $T(x_i) = \bot$ και απλοποίησε τον τύπο F.
    - Αν μετά την απλοποίηση, ο F περιέχει δύο clauses $\{x\}$ και $\{\bar{x}\}$ τότε δεν είναι ικανοποιήσιμος.

Αλγόριθμος για το Πρόβλημα 2-satisfiability

- Τώρα ο F περιέχει μόνο clauses με ακριβώς δύο literals.
  Διάλεξε μια μεταβλητή x, υπέθεσε ότι $T(x) = \top$, και απλοποίησε τον τύπο F.
  Τώρα έχουμε ένα τύπο που περιέχει clauses με μόνο ένα literal. Αρα μπορούμε να επαναλάβουμε τη διαδικασία εκκαθάρισης του βήματος 2 ώστε να απαλειφθούν όλα τα clauses αυτής της μορφής.
  Επανέφερε τον τύπο F στην αρχική του μορφή, υπέθεσε ότι $T(x) = \bot$, και επανέλαβε τη διαδικασία εκκαθάρισης.
  Αν και στις δύο περιπτώσεις η εκκαθάριση αποτύχει, τότε ο τύπος F **δεν είναι ικανοποιήσιμος.**

– Αν κάποια από τις δύο εκκαθαρίσεις πετύχει, τότε βάζουμε στο $T(x)$ την ανάλογη τιμή και συνεχίζουμε με την επόμενη μεταβλητή.

Αλγόριθμος για το Πρόβλημα 2-satisfiability

- Ο αλγόριθμος 2-SAT είναι πολυωνυμικός. (Γιατί; Ποιά είναι η ακριβής χρονική πολυπλοκότητα του;)
- Αρα το πρόβλημα 2-SATISFIABILITY ανήκει στην κλάση $\mathcal{P}$.

Παράδειγμα

- Έστω ο τύπος

$$\{\ x_1 \vee x_2,\ x_3 \vee \bar{x_2},\ x_1,\ \bar{x_1} \vee \bar{x_2},\ x_3 \vee x_4,\ \bar{x_3} \vee x_5,\ \bar{x_4} \vee \bar{x_5},\ x_4 \vee \bar{x_3}\ \}.$$

- Ο F ικανοποιείται από την παρακάτω ανάθεση τιμών αλήθειας:

$$T(x_1) = \top,\ T(x_2) = \bot,\ T(x_3) = \bot,\ T(x_4) = \top,\ T(x_5) = \bot$$

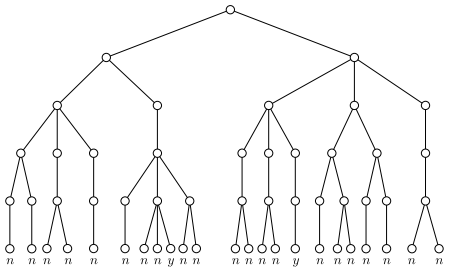
Η Κλάση $\mathcal{NP}$

- **Ορισμός.** Μια μη ντετερμινιστική μηχανή Turing $M = (K, \Sigma, \Delta, H)$ λέγεται **πολυωνυμικά φραγμένη** αν υπάρχει ένα πολυώνυμο $p(n)$ τέτοιο ώστε για κάθε συμβολοσειρά εισόδου x, δεν υπάρχει configuration C της M τέτοια ώστε

$$(s, \triangleright \sqcup x) \vdash_M^{p(|x|)+1} C.$$

  Με άλλα λόγια, δεν υπάρχει υπολογισμός της μηχανής που να συνεχίζει μετά από το πολύ $p(n)$ βήματα όπου n είναι το μήκος της εισόδου.
- Η κλάση όλων των γλωσσών που αποφασίζονται από πολυωνυμικά φραγμένες μη ντετερμινιστικές μηχανές Turing παριστάνεται από το σύμβολο $\mathcal{NP}$.

Υπολογισμοί με Μη Ντετερμινιστικές Μηχανές



Η Κλάση $\mathcal{NP}$

- Το πρόβλημα SATISFIABILITY μπορεί να λυθεί από μια μη ντετερμινιστική μηχανή Turing με δύο ταινίες ως εξής:
  – Έλεγξε εάν η είσοδος είναι η κωδικοποίηση ενός τύπου του Boole σε συζευκτική κανονική μορφή. Ταυτόχρονα υπολόγισε τον αριθμό των μεταβλητών n της συμβολοσειράς εισόδου, και μετέτρεψε την δεύτερη ταινία σε $\triangleright I^n$.
  – **Το μη ντετερμινιστικό βήμα:** Με μη ντετερμινιστικό τρόπο, αντικατέστησε κάθε I της δεύτερης ταινίας με $\top$ ή $\bot$.
  – Υπολόγισε την τιμή αλήθειας του τύπου εισόδου, θεωρώντας την συμβολοσειρά της δεύτερης ταινίας σαν ανάθεση τιμών αλήθειας.
  – Αν ο τύπος εισόδου είναι ικανοποιήσιμος, τότε μεταπήδησε σε κατάσταση αποδοχής αλλιώς σε κατάσταση απόρριψης.

Η Κλάση $\mathcal{EXP}$

- **Ορισμός.** Μια μηχανή Turing $M = (K, \Sigma, \Delta, H)$ λέγεται **εκθετικά φραγμένη** αν υπάρχει ένα πολυώνυμο $p(n)$ τέτοιο ώστε για κάθε συμβολοσειρά εισόδου x, δεν υπάρχει configuration C της M τέτοια ώστε

$$(s, \triangleright \sqcup x) \vdash_M^{2^{p(|x|)}+1} C.$$

- Με άλλα λόγια, δεν υπάρχει υπολογισμός της μηχανής που να συνεχίζει μετά από το πολύ $2^{p(n)}$ βήματα όπου n είναι το μήκος της εισόδου.
- Η κλάση όλων των γλωσσών που αποφασίζονται από εκθετικά φραγμένες μηχανές Turing παριστάνεται από το σύμβολο $\mathcal{EXP}$.

Οι Κλάσεις $\mathcal{P}$, $\mathcal{NP}$ και $\mathcal{EXP}$

- **Θεώρημα.** $\mathcal{P} \subseteq \mathcal{NP} \subseteq \mathcal{EXP}$
- **Απόδειξη:** Η σχέση $\mathcal{P} \subseteq \mathcal{NP}$ είναι προφανής επειδή κάθε πολυωνυμικά φραγμένη μηχανή Turing μπορεί να θεωρηθεί ότι είναι μη ντετερμινιστική.
- Για να αποδείξουμε τη σχέση $\mathcal{NP} \subseteq \mathcal{EXP}$ υποθέτουμε ότι έχουμε μια γλώσσα $L \in \mathcal{NP}$ και μια πολυωνυμικά φραγμένη μη ντετερμινιστική μηχανή Turing M που αποφασίζει την L.

Απόδειξη

- Μια ντετερμινιστική μηχανή Turing M' που αποφασίζει την L προσομοιώνει όλους τους δυνατούς υπολογισμούς μήκους 1 της M, όλους τους δυνατούς υπολογισμούς μήκους 2 της M, κ.ο.κ. μέχρι υπολογισμούς μήκους $p(n)+1$.
- Σ' αυτό το σημείο ή θα έχει βρεθεί ένας υπολογισμός που αποδέχεται ή όλοι οι υπολογισμοί απορρίπτουν.
- Αν η σταθερά r μας δίνει το **βαθμό του μη ντετερμινισμού** της M, τότε η M' μπορεί να προσομοιώσει την λειτουργία της M στη συμβολοσειρά εισόδου μήκους n σε χρόνο που είναι το πολύ

$$\sum_{l=1}^{p(n)+1} r^l = \frac{r^{p(n)+2}-1}{r-1} - 1 =$$

$$= r^{p(n)+1} = 2^{(p(n)+1)\log r}$$

Οι Κλάσεις $\mathcal{P}$, $\mathcal{NP}$ και $\mathcal{EQP}$

- **Θεώρημα.** $\mathcal{P} \subset \mathcal{EXP}$
- **Απόδειξη:** Η γλώσσα E ανήκει στην κλάση $\mathcal{EXP}$ αλλά όχι στην $\mathcal{P}$.
- **Ανοικτό Πρόβλημα:** Ποιά από τις παρακάτω σχέσεις $\subseteq$ είναι στην πραγματικότητα $\subset$;

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

Ενας Αλλος Χαρακτηρισμός της Κλάσης $\mathcal{NP}$

- Οι μη ντετερμινιστικοί αλγόριθμοι που λύνουν προβήματα όπως SATISFIABILITY, TSP κλπ. έχουν όλοι παρόμοια μορφή:
  - Με μη ντετερμινιστικό τρόπο να παράγεις μια συμβολοσειρά s.
  - Έλεγξε αν η συμβολοσειρά s ικανοποιεί μια ιδιότητα που έχει σχέση με το πρόβλημα και την είσοδο.
- Συμβολοσειρές όπως η s λέγονται **πιστοποιητικά ή μάρτυρες (certificates or witness).**

Ενας Αλλος Χαρακτηρισμός της Κλάσης $\mathcal{NP}$

- **Ορισμός.** Έστω Σ ένα αλφάβητο, και ";" ένα σύμβολο που δεν ανήκει στο Σ. Θεωρήστε μια γλώσσα $L' \subseteq \Sigma^*;\Sigma^*$. Η L' είναι **πολυωνυμικά ισορροπημένη** αν υπάρχει πολυώνυμο $p(n)$ τέτοιο ώστε αν $x;y \in L'$ τότε $|y| \leq p(|x|)$.
- **Θεώρημα.** Αν $L \subseteq \Sigma^*$ είναι μια γλώσσα όπου $; \notin \Sigma$ και $|\Sigma| \geq 2$. Τότε $L \in \mathcal{NP}$ αν και μόνο αν υπάρχει πολυωνυμικά ισορροπημένη γλώσσα $L' \subseteq \Sigma^*;\Sigma^*$ τέτοια ώστε $L' \in \mathcal{P}$, και

$$L = \{x : \text{ υπάρχει } y \in \Sigma^* \text{ τέτοια ώστε } x;y \in L'\}.$$

- **Διαισθητικά:** Η συμβολοσειρά y είναι πιστοποιητικό για το ότι η συμβολοσειρά x ανήκει στην κλάση L.

Απόδειξη

Αν

- Αν μια τέτοια γλώσσα L' υπάρχει, τότε μια μη ντετερμινιστική μηχανή Turing μπορεί να αποφασίσει την L σε πολυωνυμικό χρόνο δοκιμάζοντας όλα τα πιθανά πιστοποιητικά (το μήκος τους δίνεται από ένα γνωστό πολυώνυμο) και χρησιμοποιώντας την ντετερμινιστική μηχανή που αποφασίζει την L'.

Μόνο Αν

- Κάθε μη ντετερμινιστική μηχανή Turing που αποφασίζει την L μας δίνει πιστοποιητικά για την L. Το πιστοποιητικό κάθε συμβολοσειράς εισόδου x είναι ο υπολογισμός που δέχεται την x.

Τα προβλήματα linear και integer programming

- LINEAR PROGRAMMING: Έστω ένα σύστημα γραμμικών ανισώσεων με n μεταβλητές και ακέραιους συντελεστές. Έχει το σύστημα λύση στο χώρο των ρητών αριθμών;
- INTEGER PROGRAMMING: Έστω ένα σύστημα γραμμικών ανισώσεων με n μεταβλητές και ακέραιους συντελεστές. Έχει το σύστημα **ακέραια λύση**;
- Το πρόβλημα LINEAR PROGRAMMING ανήκει στην κλάση $\mathcal{P}$ (Khachiyan, 1979).
- Το πρόβλημα INTEGER PROGRAMMING ανήκει στην κλάση $\mathcal{NP}$ αλλά δεν ξέρουμε αν υπάρχει πολυωνυμικός αλγόριθμος για το πρόβλημα INTEGER PROGRAMMING.

Μελέτη

- Κεφάλαιο 6 του βιβλίου
  Harry R. Lewis και Χρίστος Χ. Παπαδημητρίου, Elements of the Theory of Computation, 2nd edition, Prentice Hall, 1998.