Μηχανές Turing

Turing Machines (TM)

- Απλός ιδεατός υπολογιστής (υπολογιστικό μοντέλο).
- Πεπερασμένη συσκευή με δυνητικά άπειρη ταινία.
- Η ταινία υποδιαιρείται σε κύτταρα που περιέχουν 0 ή 1.
- Σε κάθε χρονική στιγμή η κεφαλή της TM βρίσκεται στο «τρέχον» κύτταρο.
- Βασικές λειτουργίες μιας TM:
  - Διαβάζει το τρέχον κύτταρο.
  - Γράφει 1 ή 0 στο τρέχον κύτταρο.
  - Κάνει τρέχον το αμέσως αριστερότερο ή δεξιότερο κύτταρο.

Παράδειγμα

$M = (K, \Sigma, \delta, s, \{h\})$
$K = \{q_0, q_1, h\}$

$\Sigma = \{1, \sqcup, \rhd\}$

$s = q_0$

Συνάρτηση μετάβασης:

q	σ	$\delta(q,\sigma)$
q_0	1	$(q1, \sqcup)$
q_0	$\sqcup$	$(h, \sqcup)$
q_0	$\rhd$	$(q_0, \rightarrow)$
q_1	1	$(q_0, 1)$
q_1	$\sqcup$	$(q_0, \rightarrow)$
q_1	$\rhd$	$(q_1, \rightarrow)$

Υπολογισμός:

$$
\begin{aligned}
(q_1, \rhd \underline{\sqcup}1111) & \vdash_M (q_0, \rhd \sqcup\underline{1}111) \\
& \vdash_M (q_1, \rhd \sqcup\underline{\sqcup}111) \\
& \vdash_M (q_0, \rhd \sqcup\sqcup\underline{1}11) \\
& \vdash_M (q_1, \rhd \sqcup\sqcup\underline{\sqcup}11) \\
& \vdash_M (q_0, \rhd \sqcup\sqcup\sqcup\underline{1}1) \\
& \vdash_M (q_1, \rhd \sqcup\sqcup\sqcup\underline{\sqcup}1) \\
& \vdash_M (q_0, \rhd \sqcup\sqcup\sqcup\sqcup\underline{1}) \\
& \vdash_M (q_1, \rhd \sqcup\sqcup\sqcup\sqcup\underline{\sqcup}) \\
& \vdash_M (q_0, \rhd \sqcup\sqcup\sqcup\sqcup\sqcup\underline{\sqcup}) \\
& \vdash_M (h, \rhd \sqcup\sqcup\sqcup\sqcup\sqcup\underline{\sqcup})
\end{aligned}
$$

Βασικές TM

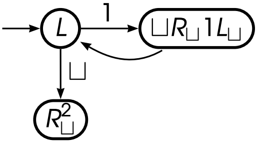
- Μηχανές εγγραφής συμβόλων: π.χ. αν $1 \in \Sigma$ τότε $M_1 : 1$.
- Μηχανές μετακίνησης κεφαλής: $M_{\leftarrow} : L$, $M_{\rightarrow} : R$
- Χτίσιμο σύνθετων μηχανών από απλούστερες:

$$
\begin{tikzcd}
M_1 \arrow[r, "0"] \arrow[d, "1"] & M_2 \\
M_3 &
\end{tikzcd}
$$

- Μηχανές $R_{\sqcup}$, $R_{\overline{\sqcup}}$, $L_{\sqcup}$, $L_{\overline{\sqcup}}$
- $R^{\curvearrowleft \sqcup} \xrightarrow{1} L1$

Παραδείγματα TM

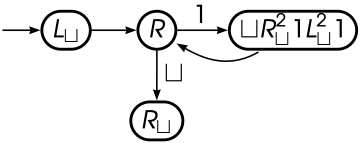
Μηχανή δεξιάς μετατόπισης της εισόδου ($\sqcup w\underline{\sqcup} \rightarrow \sqcup \sqcup w\underline{\sqcup}$):



$$
\begin{array}{ccccc}
\sqcup & 1 & 1 & \underline{\sqcup} & \sqcup \\
\sqcup & 1 & \underline{1} & \sqcup & \sqcup \\
\sqcup & 1 & \underline{\sqcup} & 1 & \sqcup \\
\sqcup & \underline{1} & \sqcup & 1 & \sqcup \\
\sqcup & \underline{\sqcup} & 1 & 1 & \sqcup \\
\underline{\sqcup} & \sqcup & 1 & 1 & \sqcup \\
\sqcup & \sqcup & 1 & 1 & \underline{\sqcup}
\end{array}
$$

Παραδείγματα TM

Μηχανή αντιγραφής της εισόδου: $(\sqcup w \underline{\sqcup} \longrightarrow \sqcup w \sqcup w \underline{\sqcup})$



$\sqcup$	1	1	$\underline{\sqcup}$	$\sqcup$	$\sqcup$	$\sqcup$
$\underline{\sqcup}$	1	1	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$
$\sqcup$	$\underline{1}$	1	$\sqcup$	$\sqcup$	$\sqcup$	$\sqcup$
$\sqcup$	$\underline{1}$	1	$\sqcup$	1	$\sqcup$	$\sqcup$
$\sqcup$	1	$\underline{1}$	$\sqcup$	1	$\sqcup$	$\sqcup$
$\sqcup$	1	$\underline{1}$	$\sqcup$	1	1	$\sqcup$
$\sqcup$	1	1	$\underline{\sqcup}$	1	1	$\sqcup$
$\sqcup$	1	1	$\sqcup$	1	1	$\underline{\sqcup}$

Υπολογισμοί με TM

Συμβάσεις

- Αρχική συνολική κατάσταση: $(s, \triangleright \underline{\sqcup} w)$.
- Η είσοδος w δεν περιέχει κενά.
- Δύο καταστάσεις τερματισμού: $H = \{y, n\}$.
- y : συνολική κατάσταση αποδοχής.
- n : συνολική κατάσταση απόρριψης.
- Η μηχανή δέχεται την είσοδο w αν παράγει μια συνολική κατάσταση αποδοχής.
- Η μηχανή απορρίπτει την είσοδο w αν παράγει μια συνολική κατάσταση απόρριψης.

TM σαν αναγνωριστές γλώσσας

Έστω $\Sigma_0 \subseteq \Sigma - \{\sqcup, \triangleright\}$. Λέμε ότι μια TM M αποφασίζει μια γλώσσα $L \subseteq \Sigma_0^*$ ανν $\forall w \in \Sigma_0^*$:

- η M δέχεται την w, αν $w \in L$,
- η M απορρίπτε την w, αν $w \notin L$.

Μια γλώσσα ονομάζεται αναδρομική αν υπάρχει μια TM που την αποφασίζει.

ΠΡΟΣΟΧΗ: Μια TM αποφασίζει μια γλώσσα αν ΠΑΝΤΑ ΤΕΡΜΑΤΙΖΕΙ είτε σε μια συνολική κατάσταση αποδοχής, είτε σε μια συνολική κατάσταση απόρριψης.

TM σαν υπολογιστές συναρτήσεων

Έστω συνάρτηση $f : \Sigma_0^* \longrightarrow \Sigma_0^*$ $(w \longrightarrow f(w))$. Λέμε ότι μια TM M υπολογίζει τη συνάρτηση f αν για κάθε $w \in \Sigma_0^*$ η μηχανή τερματίζει με ταινία $\triangleright \sqcup f(w)$.

Μια συνάρτηση f λέγεται αναδρομική αν υπάρχει TM που υπολογίζει την f. Γράφουμε τότε $M(w) = f(w)$.

Αναδρομικά απαριθμίσιμες γλώσσες

Λέμε ότι μια TM M ημιαποφασίζει μια γλώσσα $L \subseteq \Sigma_0^*$ αν για κάθε $w \in \Sigma_0^*$ ισχύει ότι $w \in L$ αν και μόνο άν η M τερματίζει με είσοδο w.

Μια γλώσσα λέγεται αναδρομικά απαριθμίσιμη αν και μόνο αν υπάρχει TM που την ημιαποφασίζει.

ΠΡΟΣΟΧΗ: Αν μια γλώσσα ημιαποφασίζεται από μια TM τότε για τα στοιχεία της γλώσσας η TM πάντα τερματίζει, όμως για τα στοιχεία του Σ_0^* η TM ΔΕΝ ΤΕΡΜΑΤΙΖΕΙ ΠΟΤΕ.

Παράδειγμα: η μηχανή $R^{\curvearrowleft\overline{a}}$ ημιαποφασίζει τη γλώσσα

$$L = \{w \mid w \in \{a, b\}^* \text{ και } w \text{ περιέχει τουλάχιστον ένα } a\}$$

Αναδρομικά απαριθμίσιμες γλώσσες

Παρατηρήσεις

- Οι TM που ημιαποφαζίζουν γλώσσες δεν είναι αναγνωριστές γλωσσών.
- Αν η είσοδος δεν ανήκει στη γλώσσα η TM δεν θα τερματίσει ποτέ κι έτσι δεν ξέρουμε ποτέ αν έχουμε περιμένει αρκετά για μια απάντηση.
- Αντίθετα, ένα πεπερασμένο αυτόματο τερματίζει ΠΑΝΤΑ.

Θεωρήματα:

- Αν μια γλώσσα είναι αναδρομική τότε είναι αναδρομικά απαριθμίσιμη (το αντίστροφο δεν ισχύει).
- Αν μια L είναι αναδρομική τότε η $\overline{L}$ είναι αναδρομική.

Παραλλαγές TM

Παραλλαγές των TM με την ίδια υπολογιστική δυνατότητα, όχι όμως και αποδοτικότητα είναι:

- πολλές ταινίες, μνήμη πλέγματος, μνήμη περισσότερων διαστάσεων,
- μεγαλύτερο Σ,
- πολλές παράλληλες κεφαλές,
- μη ντετερμινιστικές μεταβάσεις,
- . . .

Καθολική TM

Κάθε TM μπορεί να κωδικοποιηθεί από ένα φυσικό αριθμό κάτω από ένα κατάλληλο σχήμα κωδικοποίησης. Όμοια κωδικοποιείται και η είσοδος οποιασδήποτε μηχανής.

Μια καθολική TM U δέχεται σαν είσοδο τις κωδικοποιήσεις μιας οποιασδήποτε TM M και της εισόδου της w. Στη συνέχεια η U «τρέχει» την M με είσοδο w. Η U τερματίζει αν και μόνο άν η M τερματίζει.

Οι φυσικοί αριθμοί είναι ένα μετρήσιμο σύνολο. Άρα υπάρχουν μετρήσιμα πολλές κωδικοποιήσεις TM. Οι γλώσσες όμως είναι γενικά μη μετρήσιμα σύνολα (γιατί;). Άρα υπάρχουν γλώσσες που δεν μπορούν να αποφασιστούν από TM.

Θέση των Church - Turing

Όλα τα γνωστά και άγνωστα ντετερμινιστικά υπολογιστικά μοντέλα είναι μεταξύ τους ισοδύναμα.

Το παράνω είναι θέση ή αλλιώς αίτημα, π.χ. το αίτημα των παραλλήλων ευθειών στην Ευκλείδια γεωμετρία. Ένα αίτημα γίνεται δεκτό χωρίς απόδειξη.

Είναι μάλλον απίθανο να προταθεί στο μέλλον ένα υπολογιστικό μοντέλο ισχυρότερο από τις TM.

Υπολογιστικά μοντέλα

Δεν χρειάζεται να καθορίσουμε ένα συγκεκριμένο υπολογιστικό μοντέλο για τη λύση κάποιου προβλήματος.

Αν ένα πρόβλημα λύνεται από κάποιο υπολογιστικό μοντέλο τότε θα λύνεται και από οποιοδήποτε άλλο με το πολύ πολυωνυμική απώλεια χρόνου.

Μερικά ντετερμινιστικά υπολογιστικά μοντέλα:

- προγράμματα Pascal, JAVA, . . . με ή χωρίς αναδρομή,
- προγράμματα WHILE (μόνη δομή ελέγχου το WHILE),
- προγράμματα GOTO και IF,
- assembler-like RAM, URM (universal register machine,
- single register machine (ένας καταχωρητής),
- TM με πρόσβαση σε ένα μόνο κύτταρο της ταινίας σε κάθε βήμα.

Μηχανιστική απαρίθμιση προγραμμάτων

- Αν $\Sigma = \{a_1, a_2, \ldots, a_m\}$ είναι το αλφάβητο μιας γλώσσας προγραμματισμού, κάθε πρόγραμμα ανήκει στο Σ^*.
- Όμως $\Sigma^* = \cup_{n=0}^{\infty} \Sigma_n$ και κάθε Σ_n (σύνολο συμβολοσειρών του Σ με μήκος n) είναι πεπερασμένο.
- Τα στοιχεία κάθε συνόλου μπορούν να διαταχθούν αλφαβητικά οπότε μπορούμε να θεωρήσουμε την ακόλουθη απαρίθμηση για το Σ^*:

$$\begin{array}{ll} \Sigma_0: & \{\epsilon\} \\ \Sigma_1: & \{a_1, a_2, \ldots, a_m\} \\ \Sigma_2: & \{a_1a_1, a_1a_2, \ldots, a_1a_m, \ldots, a_ma_m\} \\ \vdots & \end{array}$$

- Η παραπάνω απαρίθμιση μπορεί να γίνει με πρόγραμμα.
- Με κατάλληλη χρήση compiler για τον έλεγχο ορθότητας μπορούμε να κατασκευάσουμε μηχανιστικά μια απαρίθμιση των συντακτικά ορθών n προγραμμάτων

Το πρόβλημα τερματισμού

Halting Problem (HP)

- Ο μεταγλωττιστής ελέγχει για τη συντακτική ορθότητα των προγραμμάτων.
- Τι γίνεται όμως με τα λάθη χρόνου εκτέλεσης;
- Υπάρχει πρόγραμμα που μπορεί να ελέγχει αν ένα συντακτικά ορθό πρόγραμμα θα σταματήσει κάποτε ή αν θα τρέχει για πάντα;
- Δυστυχώς τέτοιο πρόγραμμα δεν υπάρχει!
- Το HP είναι μή επιλύσιμο!

Το HP είναι μη επιλύσιμο

- Έστω ότι $\pi_0, \pi_1, \ldots$ είναι μια μηχανιστική απαρίθμηση όλων των προγραμμάτων μιας γλώσσας προγραμματισμού.
- Υποθέτουμε ότι το HP είναι επιλύσιμο.
- Μπορούμε δηλαδή να αποφανθούμε αν το πρόγραμμα π_i με είσοδο j σταματάει.
- Κατασκευάζουμε ένα πρόγραμμα π που ελέγχει αν το πρόγραμμα π_n με είσοδο n ($\pi_n(n)$) σταματάει ή όχι.
- Το πρόγραμμα π, ανάλογα με τον προηγούμενο έλεγχο, σταματάει αν το $\pi_n(n)$ δεν σταματάει, και αντιστρόφως:

  `read(n);` **`if`** $\pi_n(n)$ `terminates` **`then`** `loop_forever` **`else`** `halt`
- Το π είναι πρόγραμμα της γλώσσας συνεπώς είναι κάποιο π_i.
- Το $\pi_i(i)$ σταματάει αν και μόνο αν το $\pi(i)$ σταματάει και αυτό συμβαίνει αν και μόνο αν το $\pi_i(i)$ δεν σταματάει. Αντίφαση.
- Άρα το HP είναι μη επιλύσιμο!

Παρατηρήσεις

- Μπορούμε να κατασκευάσουμε μηχανιστικά μια άπειρη λίστα όλων των προγραμμάτων με την αντίστοιχη είσοδο για την οποία σταματούν.
- Επιλύουμε έτσι το HP;
- Φυσικά όχι! Αν το $\pi_k(n)$ δεν έχει εμφανιστεί στη λίστα, δεν ξέρουμε αν θα προστεθεί αργότερα ή αν δε θα εμφανισθεί ποτέ!
- Το HP είναι αναδρομικά απαριθμήσιμο

Πολλά άλλα προβλήματα είναι μη επιλύσιμα.

Αποκρίσιμα και καταγράψιμα σύνολα

Ένα σύνολο S λέγεται αποκρίσιμο ή υπολογίσιμο ή επιλύσιμο ή αναδρομικό (decidable, computable, solvable, recursive) ανν υπάρχει αλγόριθμος που σταματάει ή μια υπολογιστική μηχανή που δίνει έξοδο «ναι» για κάθε είσοδο $a \in S$ και έξοδο «όχι» για κάθε είσοδο $a \notin S$.

Ένα σύνολο S λέγεται καταγράψιμο ή αναδρομικά απαριθμήσιμο (listable, effectively generatable, recursively enumerable) ανν υπάρχει γεννήτρια διαδικασία ή μηχανή που καταγράφει όλα τα στοιχεία του S. Στην, πιθανώς άπειρη, λίστα εξόδου επιτρέπονται οι επαναλήψεις και δεν υπάρχει περιορισμός για την διάταξη των στοιχείων.

Μερικές απλές ιδιότητες

1. Αν το S είναι αποκρίσιμο τότε και το $\overline{S}$ είναι αποκρίσιμο.
2. Αν το S είναι αποκρίσιμο τότε το S είναι και καταγράψιμο.
3. Αν το S και $\overline{S}$ είναι καταγράψιμα τότε το S είναι αποκρίσιμο.
4. Αν το S είναι καταγράψιμο με γνησίως αύξουσα διάταξη τότε το S είναι αποκρίσιμο.

Υποδείξεις:

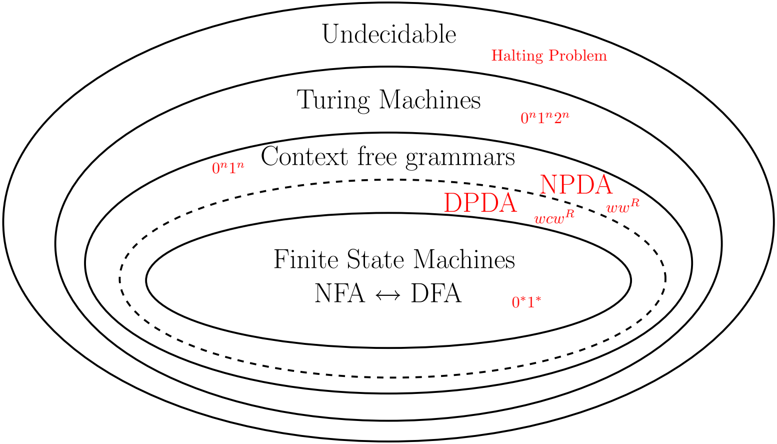
1. Στο πρόγραμμα που αποκρίνεται για το S αντέστρεψε το «ναι» και το «όχι» πριν από την έξοδο.
2. Τρέξε το πρόγραμμα που αποκρίνεται για το S διαδοχικά για όλους τους φυσικούς αριθμούς.
3. Τρέξε βήμα-βήμα παράλληλα τα προγράμματα που καταγράφουν τα S και $\overline{S}$ μέχρι η είσοδος να εμφανιστεί σε κάποια λίστα εξόδου.
4. Τρέξε το πρόγραμμα που καταγράφει το S μέχρι η είσοδος να εμφανιστεί ή ξεπεραστεί στη λίστα εξόδου.

Θεώρημα Rice

Οποιαδήποτε μη τετριμένη ιδιότητα των προγραμμάτων είναι ένα μη επιλύσιμο πρόβλημα!

- Ένα πρόγραμμα τερματίζει με μια δεδομένη είσοδο;
- Ένα πρόγραμμα χωρίς είσοδο τερματίζει;
- Για ένα πρόγραμμα υπάρχει κάποια είσοδος για την οποία το πρόγραμμα τερματίζει;
- Ένα πρόγραμμα τερματίζει για κάθε πιθανή είσοδο;
- Δύο διαφορετικά προγράμματα τερματίζουν με τις ίδιες εισόδους;
- ...

Το σύμπαν της υπολογισιμότητας



Θεωρία Πολυπλοκότητας

- Στη Θεωρία Υπολογισμού, μας ενδιαφέρει μόνον αν ένα πρόβλημα είναι υπολογίσιμο ή όχι.
- Αν ένα πρόβλημα είναι υπολογίσιμο, τότε είναι αδιάφορο τι ποσότητα αγαθών πρέπει να διατεθεί για να λυθεί το πρόβλημα.
- Στη Θεωρία Πολυπλοκότητας θεωρούμε μόνο υπολογίσιμα προβμήματα και προσπαθούμε να δούμε αν λύνονται κάτω από περιορισμούς.
- Οι υπολογιστικοί πόροι που συνήθως θεωρούνται περιορισμένοι είναι ο χρόνος υπολογισμού και ο επιπλέον χώρος μνήμης για ενδιάμεσα αποτελέσματα.
- Αυτοί οι περιορισμοί, καθώς και άλλα χαρακτηριστικά των υπολογισμών, ορίζουν κλάσεις πολυπλοκότητας.

Βασικοί ορισμοί

$\mathrm{TIME}(t(n))$ ανήκουν τα προβλήματα που μπορούν να επιλυθούν από ντετερμινιστική TM σε χρόνο $t(n)$.

$\mathrm{NTIME}(t(n))$ ανήκουν τα προβλήματα που μπορούν να επιλυθούν από μη ντετερμινιστική TM σε χρόνο $t(n)$.

$\mathrm{SPACE}(s(n))$ ανήκουν τα προβλήματα που μπορούν να επιλυθούν από ντετερμινιστική TM με χρήση επιπλέον χώρου $s(n)$.

$\mathrm{NSPACE}(s(n))$ ανήκουν τα προβλήματα που μπορούν να επιλυθούν από μη ντετερμινιστική TM με χρήση επιπλέον χώρου $s(n)$.

Ορισμοί κλάσεων πολυπλοκότητας

$$\begin{aligned}
\mathrm{P} &= \bigcup_{i\geq 1} \mathrm{TIME}(n^i)\\
\mathrm{NP} &= \bigcup_{i\geq 1} \mathrm{NTIME}(n^i)\\
\mathrm{PSPACE} &= \bigcup_{i\geq 1} \mathrm{SPACE}(n^i)\\
\mathrm{NPSPACE} &= \bigcup_{i\geq 1} \mathrm{NSPACE}(n^i)\\
\mathrm{L} &= \mathrm{SPACE}(\log n)\\
\mathrm{NL} &= \mathrm{NSPACE}(\log n)\\
\mathrm{EXP} &= \bigcup_{i\geq 1} \mathrm{TIME}(2^{n^i})\\
\mathrm{EXPSPACE} &= \bigcup_{i\geq 1} \mathrm{DSPACE}(2^{n^i})
\end{aligned}$$

Βασικές ιδιότητες

- Κάθε ντετερμινιστική TM μπορεί να θεωρηθεί ως μη ντετερμινιστική με μια μόνο επιλογή σε κάθε βήμα:
  - $\mathrm{TIME}(f(n)) \subseteq \mathrm{NTIME}(f(n))$
  - $\mathrm{SPACE}(f(n)) \subseteq \mathrm{NSPACE}(f(n))$
- Σε χρόνο $f(n)$ δεν μπορεί να εξεταστεί χώρος (αριθμός θέσεων στην ταινία της TM) παραπάνω από $f(n)$:
  - $\mathrm{TIME}(f(n)) \subseteq \mathrm{SPACE}(f(n))$
  - $\mathrm{NTIME}(f(n)) \subseteq \mathrm{SPACE}(f(n))$
- Υπάρχει η εξής ιεραρχία:

$$\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{PSPACE} = \mathrm{NPSPACE}$$

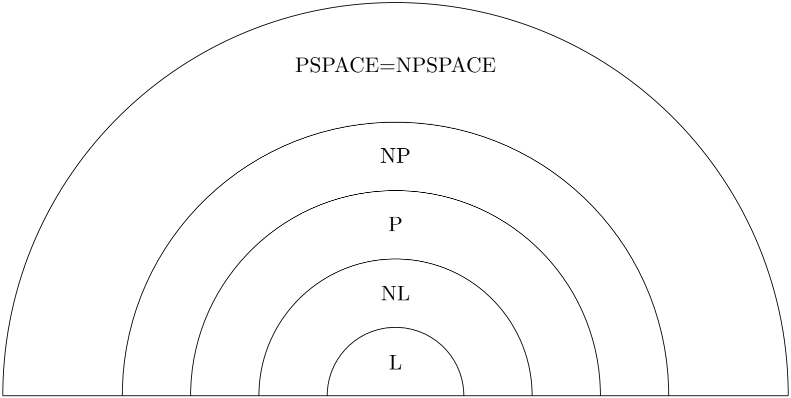
Ανοικτά προβλήματα

$$\mathrm{L} \supseteq \mathrm{NL} \supseteq \mathrm{P} \supseteq \mathrm{NP} \supseteq \mathrm{PSPACE}$$

Έπαθλο 1.000.000$ από το Clay Mathematics Institute:

$$\mathrm{P} \stackrel{?}{=} \mathrm{NP}$$

Το σύμπαν των κλάσεων πολυπλοκότητας



Αναγωγές

Το πρόβλημα «τι θα φάει μια μπλέ αγελάδα;» έχει τη λύση «φαγητό για μπλέ αγελάδες». Προκύπτει τώρα το πρόβλημα «τι θα φάει μια ροζ αγελάδα;». Μια λύση αυτού του προβλήματος είναι «κλείνουμε τη μύτη της ροζ αγελάδας, περιμένουμε να γίνει μπλέ και στη συνέχεια την ταΐζουμε με φαγητό για μπλέ αγελάδες».

Μια αναγωγή συνδέει μεταξύ τους προβλήματα με υπολογιστικά «εύκολο» τρόπο. Εύκολες συναρτήσεις και προβλήματα είναι αυτά που υπολογίζονται σε πολυωνυμικό χρόνο.

FP είναι το σύνολο των συναρτήσεων που υπολογίζεται από ντετερμινιστική TM σε πολυωνυμικό χρόνο.

Αναγωγή $\leq_m^P$(κατά Karp): Το πρόβλημα A ανάγεται στο πρόβλημα B:

$$A \leq_m^P B : \ \exists f \in \mathrm{FP} : \ (x \in A \Leftrightarrow f(x) \in B), \ \forall x \in A$$

Δύσκολα και πλήρη προβλήματα

Λέμε ότι μια κλάση γλωσσών C είναι κλειστή ως προς μια αναγωγή $\leq$ αν $A \leq B$ και $B \in C \Rightarrow A \in C$

Κλάσεις πολυπλοκότητας κλειστές ως προς την αναγωγή κατά Karp ($\leq_m^P$) είναι μεταξύ άλλων οι P, PSPACE, EXP, EXPSPACE.

Λέμε ότι το πρόβλημα A είναι C-δύσκολο (C-hard) ως προς την αναγωγή $\leq$, αν $\forall B \in C: \; B \leq A$. Η έννοια της δυσκολίας δίνει ένα κάτω όριο για την πολυπλοκότητα ενός προβλήματος: «το πρόβλημα A είναι τουλάχιστο τόσο δύσκολο όσο οποιοδήποτε πρόβλημα μιας C».

Λέμε ότι το πρόβλημα A είναι C-πλήρες (C-complete) ως προς την αναγωγή $\leq$, αν A είναι C-hard ως προς $\leq$ και $A \in C$.

Πλήρη προβλήματα μέσα στο σύμπαν της πολυπλοκότητας