Αλγόριθμοι και Πολυπλοκότητα

Εθνικό Μετσόβιο Πολυτεχνείο

7ο εξάμηνο

Σ.Η.Μ.Μ.Υ. & Σ.Ε.Μ.Φ.Ε.

http://www.corelab.ece.ntua.gr/courses/

9η ενότητα: Υπολογισιμότητα, Θεωρία Πολυπλοκότητας

Διδάσκοντες: Στάθης Ζάχος – Άρης Παγουρτζής

Υπολογισιμότητα

- Αριστοτέλης
- Leibniz
- Cantor
- Frege
- Russel
- Hilbert
- Godel
- Turing
- Church
- Kleene

Θέση του Church

Ολα τα γνωστά και τα "άγνωστα" μοντέλα της έννοιας "υπολογιστός" είναι μηχανιστικά ισοδύναμα (effectively equivalent).

Δηλαδή, δοθέντος ενός αλγορίθμου σε ένα μοντέλο για μία συγκεκριμένη συνάρτηση f, μπορούμε μηχανιστικά (με τη βοήθεια μηχανής) να κατασκευάσουμε αλγόριθμο σε ένα άλλο μοντέλο για την ίδια συνάρτηση f.

Ύπαρξη μη-υπολογιστών συναρτήσεων

- Υπάρχουν άπειρα μεν, αλλά μόνο αριθμήσιμα (*countable*) διαφορετικά προγράμματα. Εκτός αυτού μπορούμε χρησιμοποιώντας κωδικοποίηση να τα απαριθμήσουμε μηχανιστικά (*effectively enumerate*)[3]

- Από την άλλη μεριά όμως, ξέρουμε ότι υπάρχουν μη αριθμήσιμες άπειρες (*uncountable*) διαφορετικές συναρτήσεις. Αυτό αποδεικνύεται με διαγωνιοποίηση (*diagonalization*), ανάλογη με αυτή που χρησιμοποιούμε για να δείξουμε ότι το σύνολο $\mathbb{R}$ είναι μη αριθμήσιμο[4]

Θεώρημα: το Halting Problem είναι μη-αποκρίσιμο

Απόδειξη. Έστω ότι $\pi_0, \pi_1, \pi_2, \ldots$ είναι μια μηχανιστική απαρίθμηση (*effective enumeration*) όλων των προγραμμάτων. Ας υποθέσουμε ότι το **HP** είναι επιλύσιμο. Τότε κατασκευάζουμε ένα πρόγραμμα π, που ελέγχει αν το πρόγραμμα π_n με είσοδο n σταματάει ή όχι και ανάλογα με την απάντηση σε αυτόν τον έλεγχο, το πρόγραμμα π σταματάει αν το $\pi_n(n)$ δεν σταματάει, και αντιστρόφως:

$$\pi\colon \text{read}(n); \textbf{if } \pi_n(n) \text{ terminates } \textbf{then } \text{loop\_forever } \textbf{else } \text{halt}$$

Αποκρισιμότητα (Decidability) Καταγραψιμότητα (Listability)

Ορισμός 12.1.2. Ένα σύνολο S λέγεται αποκρίσιμο ή υπολογιστό ή επιλύσιμο (*decidable, computable, solvable*) αν και μόνο αν υπάρχει ένας αλγόριθμος που σταματάει ή μια υπολογιστική μηχανή που δίνει έξοδο «ναι» για κάθε είσοδο $a \in S$ και έξοδο «όχι» για κάθε είσοδο $a \notin S$.

Ορισμός 12.1.3. Ένα σύνολο S λέγεται καταγράψιμο (με μηχανιστική γεννήτρια) (*listable, effectively generatable*) αν και μόνο αν υπάρχει μια 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.

Υπολογιστικά Μοντέλα

- προγράμματα Pascal
- προγράμματα Pascal χωρίς αναδρομή (αφαίρεση αναδρομής με χρήση στοίβας)
- προγράμματα Pascal χωρίς αναδρομή και χωρίς άλλους τύπους δεδομένων εκτός από τους φυσικούς αριθμούς (επιτυγχάνεται με κωδικοποιήσεις)
- προγράμματα WHILE (μόνη δομή ελέγχου το WHILE)
- προγράμματα GOTO και IF
- Assembler-like RAM (random access machine), URM (universal register machine)
- SRM (single register machine) ένας καταχωρητής
- Μηχανή Turing (πρόσβαση μόνο σε μια κυψέλη "cell" της ταινίας κάθε φορά)

Χαρακτηριστικά Υπολογιστικών Μοντέλων

- ντετερμινιστική πολυπλοκότητα σε διακριτά βήματα
- πεπερασμένο σύνολο εντολών που εκτελούνται από επεξεργαστή
- απεριόριστη μνήμη

Άλλα Μοντέλα Υπολογισμού

- παραλλαγές από μηχανές Turing
- Thue: κανόνες επανεγγραφής (re-writing rules)
- Post: κανονικά συστήματα (normal systems)
- Church: λογισμός λ (λ-calculus)
- Curry: συνδυαστική λογική (combinatory logic)
- Markov: M. αλγόριθμοι
- Kleene: γενικά αναδρομικά σχήματα (general recursive schemes)
- Shepherdson-Sturgis, Elgott: URM, SRM, RAM, RASP
- Σχήματα McCarthy (If ... then ... else ... $\Rightarrow$ LISP)

Θεώρημα 12.2.1. *f είναι TM υπολογιστή ανν*

- *f είναι WHILE-υπολογιστή*
- *f είναι GOTO-υπολογιστή*
- *f είναι PASCAL-υπολογιστή*
- *f είναι μερικά αναδρομική (partial recursive)*

Παραλλαγές Μηχανών Turing

- πολλές ταινίες, μνήμη πλέγματος (grid memory), μνήμη περισσοτέρων διαστάσεων
- μεγαλύτερο Σ
- πολλές παράλληλες κεφαλές
- μη ντετερμινιστικές μεταβάσεις
- μίας κατευθύνσεως, απείρου μήκους ταινία
- εγγραφή και κίνηση της κεφαλής σε κάθε βήμα

Ίδια υπολογιστική δυνατότητα, όχι όμως και αποδοτικότητα (efficiency)

Αφηρημένη Θεωρία Πολυπλοκότητας

Η χρονική πολυπλοκότητα ενός αλγορίθμου που επιλύει κάποιο υπολογιστικό πρόβλημα είναι μία αύξουσα συνάρτηση $T(n)$ όπου n το μέγεθος της εισόδου $(input)$. Συγκεκριμένα:

$$T(n) = \max\{\#\text{steps for input}x \mid |x| = n\}$$

Θεωρία Πολυπλοκότητας: μέγεθος εισόδου

Το μέγεθος του input εξαρτάται φυσικά από την αναπαράστασή του. Όμως αν θεωρήσουμε ότι:

- Στην αναπαράστασή του δεν χρησιμοποιούνται σύμβολα που να μην εκφράζουν τίποτα ή να εκφράζουν πληροφορίες που δεν χρειάζονται.
- Κατά την κωδικοποίηση (αντιστοίχηση των εισόδων σε αριθμούς), οι αριθμοί που μένουν αχρησιμοποίητοι είναι «λίγοι». Συγκεκριμένα υπάρχει πολυώνυμο $p(n)$ ώστε για κάθε n, ο n-οστός μεγαλύτερος κωδικός να είναι μικρότερος από $p(n)$.
- Για την αναπαράσταση αριθμών χρησιμοποιείται το δυαδικό, το δεκαδικό ή οποιοδήποτε άλλο σύστημα εκτός από το εναδικό, τότε:

Κάθε αναπαράσταση (*encoding*) της εισόδου μπορεί να διαφέρει μόνο πολυωνυμικά από μία άλλη. Αυτό σημαίνει ότι αν η χρονική πολυπλοκότητα ενός προβλήματος είναι πολυωνυμική, τότε οποιαδήποτε αναπαράσταση για την είσοδο ενός στιγμιοτύπου (*instance*) του προβλήματος και να χρησιμοποιήσουμε, η $T(n)$ παραμένει πολυωνυμική, αφού η σύνθεση πολυωνύμων είναι πολυώνυμο.

Θεωρία Πολυπλοκότητας: αποδοτικότητα αλγορίθμου

Ορισμός 13.1.1. Αν υπάρχει κάποιο πολυώνυμο P τέτοιο ώστε: $\forall n, T(n) \leq p(n)(T(n) = O(poly))$ τότε λέμε ότι ο αλγόριθμος είναι αποδοτικός (*efficient*), δηλαδή επιλύει το πρόβλημα σε πολυωνυμικό χρόνο (*Edmonds '68*).

Έτσι λοιπόν, όταν εξετάζουμε την «αφηρημένη» πολυπλοκότητα ενός προβλήματος, πρέπει να έχουμε κατά νου τα εξής:

- Δεν μας ενδιαφέρει κάποιο συγκεκριμένο υπολογιστικό μοντέλο.
- Δεν μας ενδιαφέρει κάποια συγκεκριμένη κωδικοποίηση της εισόδου.
- Δεν μας ενδιαφέρει ο συγκεκριμένος βαθμός του πολυωνύμου.

Non-Determinism

- searching:

```
choose a[i]
verify : if a[i]=x then found
```

- sorting:

```
choose a permutation
verify : a[i]<a[i+1] for all i
```

Non-Determinism: SAT

Παράδειγμα 13.5.3. Το πρόβλημα SAT. Έστω x_i προτασιακές μεταβλητές που παίρνουν τιμή True ή False. Ονομάζουμε:

- literals, τους όρους $x_i, \neg x_i$,
- clauses, τις διαζεύξεις (disjunctions) από literals $literal_1 \vee literal_2 \vee \ldots \vee literal_m$
- CNF (Conjuctive Normal Form), την ακόλουθη μορφή:

$$clause_1 \wedge clause_2 \wedge \ldots \wedge clause_n$$

Ορίζουμε σαν πρόβλημα της Ικανοποιησιμότητας (*SATisfiability*) το εξής:

SAT
Δεδομένα: Μία boolean έκφραση σε CNF
Ερώτηση: Υπάρχει ανάθεση τιμών στις μεταβλητές που να ικανοποιεί την έκφραση (δηλαδή η έκφραση να αποτιμάται σε True);

Optimization vs. Decision Problems

- TSP
- Subgraph Isomorphism
- SAT

Η Κλάση P

$$T(n) = \max_{|x|=n}\{\text{steps to decide } x\}$$

Ένα DTM πρόγραμμα M καλείται πρόγραμμα πολυωνυμικού χρόνου αν υπάρχει ένα πολυώνυμο p τέτοιο ώστε: $\forall n \in Z^+ : T_M(n) \leq p(n)$.

$$P = \{L \mid \exists \text{ πολυωνυμικού χρόνου DTM που αποκρίνεται για την γλώσσα } L\}$$

$$P = \{L \mid \exists \text{ πολυωνυμικού χρόνου DTM που αποδέχεται την γλώσσα } L\}$$

Η Κλάση NP

$$T(n) = \begin{cases} \max\limits_{|x|=n}\{\min \#\ \text{βημάτων για αποδοχή } x\}, & \text{αν κάποιο τέτοιο } x \text{ αποδεκτό} \\ 1, & \text{αλλιώς} \end{cases}$$

Ένα NDTM πρόγραμμα M είναι πολυωνυμικού χρόνου αν:

$$\exists \text{ πολυώνυμο } p : \ T(n) \leq p(n), \forall n$$

$$NP = \{L \mid \exists\ NDTM \text{ πρόγραμμα } M \text{ το οποίο μέσα σε πολυωνυμικό χρόνο αποδέχεται τη γλώσσα } L\}$$

Σχέση P και NP

- $P \subseteq NP$
- $NP \subseteq DEXPTIME$

Αναγωγή κατά Karp

$$A \leq_m^p B : \exists f \in P_f, \forall x (x \in A \iff f(x) \in B)$$

Ιδιότητες:

1. Ανακλαστική: $A \leq_m^p A$.

2. Μεταβατική: Αν $A \leq_m^p B$ και $B \leq_m^p C$ τότε $A \leq_m^p C$. Αυτό αποδεικνύεται εύκολα ως εξής:

   - $A \leq_m^p B : \exists f \in P_f, \forall x (x \in A \iff f(x) \in B)$
   - $B \leq_m^p C : \exists g \in P_f, \forall x (x \in B \iff g(x) \in C)$

Ιδιότητες Αναγωγής κατά Karp (i)

1. Ανακλαστική: $A \leq_m^p A$.

2. Μεταβατική: Αν $A \leq_m^p B$ και $B \leq_m^p C$ τότε $A \leq_m^p C$. Αυτό αποδεικνύεται εύκολα ως εξής:

   - $A \leq_m^p B : \exists f \in P_f, \forall x(x \in A \iff f(x) \in B)$
   - $B \leq_m^p C : \exists g \in P_f, \forall x(x \in B \iff g(x) \in C)$

   Δηλαδή:$\exists f, g \in P_f, \forall x((x \in A \iff f(x) \in B) \iff g(f(x)) \in C)$.
   Άρα

   $$\exists h \in P_f, \forall x(x \in Ax \iff h(x) \in C)$$

   όπου h είναι η σύνθεση των f, g και είναι πολυωνυμική αφού η σύνθεση πολυωνύμων είναι πολυώνυμο. Συνεπώς $A \leq_m^p C$.

Ιδιότητες Αναγωγής κατά Karp (ii)

3. Αν $A \leq_m^p B$ και $B \leq_m^p A$ τότε $A \equiv_m^p B$, και λέμε ότι τα προβλήματα A, B είναι ισοδύναμα ως προς $\leq_m^p$, (π.χ. αν $A,\ B \in P \Rightarrow A \equiv_m^p B$).

4. Αν $A \leq_m^p B$ και $B \in P \Rightarrow A \in P$. Αυτό αποδεικνύεται εύκολα ως εξής: Αφού $A \leq_m^p B$ σημαίνει ότι υπάρχει συνάρτηση f υπολογίσιμη σε πολυωνυμικό χρόνο τέτοια ώστε:

$$\forall x(x \in A \iff f(x) \in B)$$

Hardness - Completeness

Ένα πρόβλημα L είναι **NP-complete** ως προς $\leq_m^p$ αν:

$$(L \in NP) \wedge (\forall L' \in NP : \; L' \leq_m^p L)$$

Λήμμα 13.6.1. Αν $L_1 \leq_m^p L_2$, το L_1 είναι **NP-complete** και $L_2 \in NP$ τότε το L_2 είναι **NP-complete**.

Αναγωγή κατά Cook

Λέμε ότι ένα πρόβλημα A ανάγεται κατά Cook σε ένα πρόβλημα B και συμβολίζουμε, $A \leq_T^P B$, αν το A μπορεί να αποφασιστεί από μία πολυωνυμικού χρόνου ντετερμινιστική μηχανή Turing η οποία χρησιμοποιεί ένα **μαντείο** (*oracle*) για το B. Αυτό σημαίνει ότι η DTM μπορεί να κάνει οσεσδήποτε ερωτήσεις για οποιοδήποτε στιγμιότυπο του B και να πάρει στιγμιαία σωστές απαντήσεις.

$$P^A = \{L \mid L \leq_T^P A\}$$

Αναγωγή κατά Cook (συν.)

Ιδιότητες:

- αναλαστική,
- μεταβατική,
- $\equiv_T^P$

Έννοιες:

- NP-hard ως προς $\leq_T^P$
- NP-complete ως προς $\leq_T^P$