2ο μέρος

Θεωρία Υπολογισμού

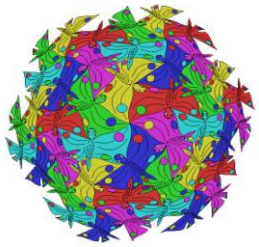
Theory of Computation

Υπολογισιμότητα - Computability

- Υπολογισιμότητα (Computability)
  - Τι μπορεί να υπολογιστεί και τι όχι;
- Υπολογιστική πολυπλοκότητα (Computational Complexity)
  - Τι μπορεί να υπολογιστεί γρήγορα και τι όχι;
  - Πόσο γρήγορα μπορεί να υπολογιστεί;

Τι είναι πολυπλοκότητα;

- Το 10110101110 είναι πιο πολύπλοκο από το 0000000000000 (Kolmogorov complexity)
- Τα θηλαστικά είναι πιο πολύπλοκα από τους ιούς.
- Το σκάκι είναι πιο πολύπλοκο από την τρίλιζα.
- Οι επικαλύψεις του Escher είναι πιο πολύπλοκες από τα τετράγωνα πλακάκια του μπάνιου μου.
- Οι πρώτοι αριθμοί είναι πιο πολύπλοκοι από τους περιττούς (υπολογιστική πολυπλοκότητα).



Τι είναι υπολογιστική πολυπλοκότητα;

- Ένας τρόπος για να συλλάβουμε γιατί οι πρώτοι αριθμοί είναι πιο πολύπλοκοι από τους περιττούς είναι η υπολογιστική πολυπλοκότητα.
- Το πρόβλημα

  «Δίνεται x. Είναι πρώτος;»

  είναι πιο δύσκολο από το πρόβλημα

  «Δίνεται x. Είναι περιττός;»

Τι είναι πρόβλημα;

- **1ο ΠΡΟΒΛΗΜΑ**: Υπάρχουν ακέραιοι x,y,z>0 και n>2 τέτοιοι ώστε $x^n+y^n=z^n$;
  Αυτό είναι το Θεώρημα του Fermat που απαντήθηκε αρνητικά πρόσφατα από τον Andrew Wiles.
- **2ο ΠΡΟΒΛΗΜΑ**: Γράψτε ένα πρόγραμμα που όταν του δίνουμε για είσοδο μια πολυωνυμική εξίσωση (π.χ. $x^3+y^3=z^3$) απαντά αν έχει ακέραια λύση ή όχι.
  Αυτό είναι το δέκατο πρόβλημα του David Hilbert που τέθηκε το 1900 και απαντήθηκε (αρνητικά) από τον Yuri Matiyasevitch το 1970.

Τι είναι πρόβλημα;

- Θα μιλήσουμε για υπολογιστικά προβλήματα, δηλαδή, προβλήματα που ζητάνε να βρούμε ένα αλγόριθμο (πρόγραμμα).
- Π.χ. το 2ο πρόβλημα (πρόβλημα Hilbert) είναι υπολογιστικό πρόβλημα, αλλά το 1ο πρόβλημα (Θεώρημα του Fermat) δεν είναι.

Ιστορία υπολογισιμότητας

- 1900: Ο David Hilbert ρωτάει αν μπορούν να αυτοματοποιηθούν τα μαθηματικά;
- 1930: Ο Kurt Godel δείχνει ότι αυτό δεν γίνεται με το περίφημο Θεώρημα της μη πληρότητας (Incompleteness Theorem).
- 1936: Ο Alan Turing ορίζει την έννοια του υπολογιστή και δείχνει ότι πολλά προβλήματα δεν μπορούν να επιλυθούν με συστηματικό τρόπο, δηλαδή δεν υπάρχει πρόγραμμα που να τα λύνει.

Μη επιλύσιμα προβλήματα – Πρόβλημα τερματισμού

- Το πρόγραμμα 3x+1:

```
While x!=1 do
    if (x is even) then x=x/2
    else x=3*x+1
```

- 7 → 22 → 11 → 34 → 17 → 52 → 26 → 13 → 40 → 20 → 10 → 5 → 16 → 8 → 4 → 2 → 1
- Πρόβλημα: Δίνεται x. Τερματίζει το πρόγραμμα;
- Πρόβλημα: Τερματίζει το πρόβλημα για κάθε φυσικό αριθμό x;
- Δεν γνωρίζουμε την απάντηση (είναι δηλαδή ανοικτά προβλήματα).

Μη επιλύσιμα προβλήματα – Πρόβλημα τερματισμού

- Το 3x+1 είναι ειδική περίπτωση του προβλήματος τερματισμού:
  Δίνεται πρόγραμμα και είσοδος. Τερματίζει το πρόγραμμα για αυτή την είσοδο;
  Μια ισοδύναμη παραλλαγή είναι:
  Δίνεται πρόγραμμα χωρίς είσοδο. Τερματίζει;
- Θεώρημα: Το πρόβλημα τερματισμού είναι μη επιλύσιμο. Δηλαδή, δεν υπάρχει αλγόριθμος που να απαντάει σε αυτή την ερώτηση.

Γιατί δεν είναι επιλύσιμο;

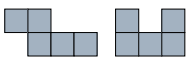
- Με εις άτοπον απαγωγή: Έστω ότι είναι επιλύσιμο, δηλαδή υπάρχει πρόγραμμα T τέτοιο ώστε «T(P,w) απαντά αν το P(w), δηλαδή το πρόγραμμα P με είσοδο w, τερματίζει ή όχι».
- Μπορούμε τότε να κατασκευάσουμε το πρόγραμμα S(P):

```
S(P)
    if T(P,P)=true then
        while true; // loop forever
```

- S(P) τερματίζει ↔ P(P) δεν τερματίζει
- S(S) τερματίζει ↔ S(S) δεν τερματίζει.
- Άτοπο, άρα το πρόγραμμα T δεν υπάρχει.

Μη επιλύσιμα προβλήματα – Πρόβλημα επικάλυψης

- Δίνεται πεπερασμένος αριθμός ειδών από πλακάκια, π.χ.



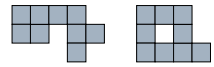
- Μπορούμε να καλύψουμε όλο το επίπεδο με τέτοια πλακάκια;
- Το πρόβλημα είναι μη επιλύσιμο. Δηλαδή, δεν υπάρχει πρόγραμμα, που να παίρνει για είσοδο τους τύπους πλακακιών και να απαντά την ερώτηση.

Μη επιλύσιμα προβλήματα – 10ο πρόβλημα του Hilbert

- Δίνεται Διοφαντική εξίσωση, δηλαδή ακέραια πολυωνυμική εξίσωση (π.χ. $x^2-2y^2=5$ ή $x^3+y^3=z^3$). Έχει λύση στους ακέραιους;
- Το πρόβλημα το έθεσε ο Hilbert το 1900. Το 1970 ο Yuri Matiyasevitch έδειξε ότι είναι μη επιλύσιμο. Δηλαδή, δεν υπάρχει πρόγραμμα, που να παίρνει για είσοδο μια εξίσωση και να απαντά την ερώτηση.

Μη επιλύσιμα προβλήματα – το πρόβλημα της σφαίρας.

- Δίνονται τετράγωνα. Είναι το σχήμα τοπολογικά ισόμορφο με δίσκο; Αν δηλαδή ήταν από πλαστελίνη, μπορούμε να την μετατρέψουμε σε δίσκο, χωρίς να σχίσουμε ή να κολλήσουμε τμήματα της; Η απάντηση είναι καταφατική για το πρώτο σχήμα και αρνητική για το δεύτερο.

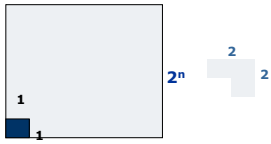


Μη επιλύσιμα προβλήματα – το πρόβλημα της σφαίρας.

- Το πρόβλημα αυτό για υψηλότερες διαστάσεις είναι μη επιλύσιμο.
- Αντίθετα για τις 3 διαστάσεις προτάθηκε πρόσφατα ένας αλγόριθμος.
- Το πρόβλημα σχετίζετε με το ερώτημα: Τι σχήμα έχει το σύμπαν μας;
- Σχετίζεται επίσης με την εικασία του Poincare (ένα από τα μεγάλα ανοικτά προβλήματα των μαθηματικών).

Άσκηση 3^n

- **Θέλουμε να πλακοστρώσουμε το δάπεδο μιας κουζίνας διαστάσεων 2^n X 2^n αφήνοντας ελεύθερη μια γωνιά 1x1 (για να περάσουν τα υδραυλικά).**
- **Τα πλακίδια που έχουμε στη διάθεσή μας έχουν γωνιακό σχήμα με εμβαδόν 3 μονάδες (ιδέ σχήμα)**
- **Ερώτημα: Μπορεί να γίνει η πλακόστρωση για οποιαδήποτε τιμή του n= 1, 2, 3,… ; Απόδειξη**
- **2^n**



Προσοχή στην Επαγωγή

(όμοια με φαινόμενο Domino, αλλά)

- **Εάν μια πρόταση ισχύει για ένα μεγάλο αριθμό περιπτώσεων αυτό ΔΕΝ σημαίνει ότι ισχύει και όλες τις περιπτώσεις**
- **Παράδειγμα 1: το πολυώνυμο n^2+n+41 γεννά πρώτους αριθμούς για n=0,…, 39. Για n=40 δίνει $1681=41^2$ και για n=41 δίνει πολλαπλάσιο του 41**
  - **Αποδεικνύεται ότι δεν υπάρχει ακέραιο πολυώνυμο που να παράγει τους πρώτους αριθμούς**
- **Παράδειγμα 2 (ακόμη χειρότερο):**

  Με κατάλληλο πρόγραμμα που τρέχουμε σε έναν υπολογιστή για κάμποσα χρόνια βλέπουμε ότι το πολυώνυμο $991n^2+1$ ΔΕΝ παράγει τέλεια τετράγωνα!
  - **Δυστυχώς μετά από περίπου 10^{38} δοκιμές συναντούμε το πρώτο n για το οποίο δεν ισχύει αυτό!**

n= 12.055.735.790.331.359.447.442.538.767

Ανακεφαλαίωση Θεωρίας Υπολογισιμότητας

- Δεν υπάρχουν αλγόριθμοι για τα προβλήματα:
  - Τερματισμού
  - Επικάλυψης
  - Hilbert
  - Σφαίρας
- Είναι ανοικτό το πρόβλημα 3x+1.
- Τα αποτελέσματα αυτά επηρέασαν σημαντικά τη σκέψη του σύγχρονου ανθρώπου. Δεν αποτελούν όμως σήμερα αντικείμενο εκτεταμένης έρευνας, γιατί τα βασικά ερωτήματα έχουν απαντηθεί.

Υπολογιστική Πολυπλοκότητα

- Η θεωρία υπολογιστικής πολυπλοκότητας ασχολείται κυρίως με το ερώτημα «πόσο γρήγορα μπορεί να υπολογιστεί;»
- Παράδειγμα: Οι αριθμοί Fibonacci 1, 1, 2, 3, 5, 8, 13, 21,… $F_n=F_{n-1}+F_{n-2}$
- Πρόβλημα: Δίνεται n, να υπολογιστεί το F_n
- Πόσο γρήγορο μπορεί να είναι το πρόγραμμα μας;

Αριθμοί Fibonacci – αναδρομικός αλγόριθμος

- F(int n)
  {
  if (n<=2) return ;
  else return F(n-1)+F(n-2);
  }
- Πόσο χρόνο θα πάρει να υπολογιστεί το F(n);
  Απάντηση: Περίπου 1.62^n βήματα. Ή όπως το συμβολίζουμε $O(1.62^n)$.
- Ο χρόνος εξαρτάται βέβαια από την ταχύτητα του υπολογιστή μας, το λειτουργικό σύστημα κλπ. αλλά πάνω από όλα από το $O(1.62^n)$.

Χρόνος εκτέλεσης αλγορίθμου

- Θεωρείστε 3 προγράμματα με αριθμό βημάτων O(2^n), O(n^2), και O(n) που το καθένα παίρνει 100 δευτερόλεπτα για να υπολογίσει το F(100).
- Πόσα δευτερόλεπτα θα πάρουν για να υπολογίσουν το F(n);

	2^n	*n^2*	*n*
n=100	*100*	*100*	*100*
n=101	*200*	*102*	*101*
n=110	*102400*	*121*	*110*
n=200	*??????*	*400*	*200*

Αριθμοί Fibonacci – καλύτερος αλγόριθμος

- a=1;
  b=1;
  for (i=2; i<=n; i++) {
  c=b; b=a+b; a=c;
  }
  return b;
- Χρόνος O(n);

Αριθμοί Fibonacci – ακόμα καλύτερος αλγόριθμος

- Μπορούμε να γράψουμε τον υπολογισμό σε μορφή πινάκων:

$$\begin{bmatrix} F(n) \\ F(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} F(n-1) \\ F(n-2) \end{bmatrix}$$

Από αυτό συμπεραίνουμε

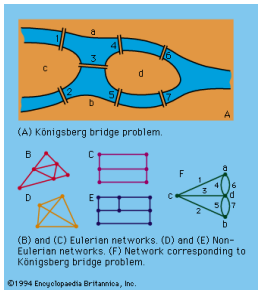
$$\begin{bmatrix} F(n) \\ F(n-1) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^{n-2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Και ο αριθμός των **αριθμητικών πράξεων** μειώνεται στο O(log n).

P =? NP

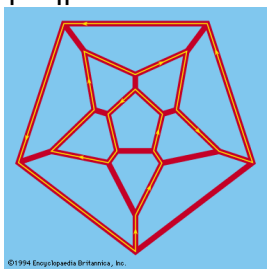
- Τι είναι πιο εύκολο; Να βρείτε τις λύσεις των ασκήσεων ή να τις αντιγράψετε;
- Πόσο πιο εύκολο είναι να βρούμε κάποια λύση από το να την επιβεβαιώσουμε;
- Αυτό είναι ουσιαστικά το P=?NP πρόβλημα, που αποτελεί το πιο σημαντικό ανοικτό πρόβλημα σήμερα. Στο http://www.claymath.org προσφέρονται 1εκ. δολάρια για τη λύση του.

Το πρόβλημα του Euler



Δίνεται γράφος. Υπάρχει τρόπος να περάσουμε από κάθε ακμή μια ακριβώς φορά;

Το πρόβλημα του Hamilton



Δίνεται γράφος. Υπάρχει τρόπος να περάσουμε από κάθε κορυφή μια ακριβώς φορά;

Euler --- Hamilton

- Το πρόβλημα του Euler είναι εύκολο. Μπορούμε γρήγορα να απαντήσουμε: Ελέγχουμε αν ο αριθμός των ακμών σε κάθε κόμβο είναι άρτιος.
  - Τέτοια προβλήματα που οι αλγόριθμοι τους χρειάζονται χρόνο $O(n)$, $O(n^2)$, $O(n^3)$ … ανήκουν στην κλάση P (polynomial time).
- Το πρόβλημα του Hamilton είναι πιο δύσκολο. Δεν γνωρίζουμε κανένα γρήγορο αλγόριθμο γι' αυτό. Ο καλύτερος γνωστός αλγόριθμος δεν διαφέρει ουσιαστικά από το να δοκιμάσουμε όλους τους συνδυασμούς --- που είναι n!=1.2.3…n.

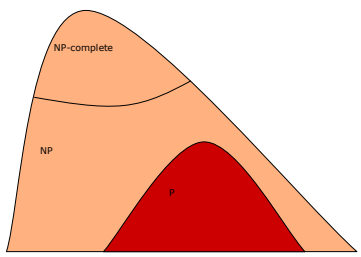
NP-complete προβλήματα

- Το πρόβλημα του Hamilton μπορεί να έχει γρήγορο αλγόριθμο. Δεν πιστεύουμε όμως ότι έχει. Ούτε καταφέραμε να αποδείξουμε κάτι τέτοιο.
- Το μόνο που μπορούμε να δείξουμε είναι ότι μια πλειάδα από προβλήματα που μας ενδιαφέρουν είναι της ίδιας δυσκολίας.
- Τα προβλήματα που είναι το ίδιο δύσκολα με το πρόβλημα του Hamilton τα λέμε NP-complete.

Κλάσεις πολυπλοκότητας

- **P (polynomial time)**: Το σύνολο των προβλημάτων που έχουν αλγόριθμο πολυωνυμικού χρόνου. Τα ταυτίζουμε με τα προβλήματα που μπορούμε να λύσουμε στην πράξη.
  - Το πρόβλημα του Euler ανήκει στο P
- **NP (nondeterministic polynomial time)**: Το σύνολο των προβλημάτων που μπορούμε να επιβεβαιώσουμε τη λύση τους (αν μας δοθεί) σε πολυωνυμικό χρόνο.
- **NP-complete**: Το υποσύνολο των πιο δύσκολων προβλημάτων του NP. Αν ένα από αυτά τα προβλήματα ανήκει στο P, τότε P=NP.
  - Το πρόβλημα του Hamilton είναι NP-complete.

Κλάσεις πολυπλοκότητας



Άλλα NP-complete προβλήματα

- Satisfiability
  - Δίνεται Boolean φόρμουλα $\varphi(x_1,\ldots,x_n)$. Υπάρχουν τιμές για τα $x_1,\ldots,x_n$ που να ικανοποιούν την φ;
- Partition
  - Δίνονται ακέραιοι $a_1,\ldots,a_n$. Μπορούν να χωριστούν σε δύο μέρη με ίσα αθροίσματα;
- Πάρα πολλά άλλα προβλήματα.

Προβλήματα πρώτων αριθμών

- Primality testing: Δίνεται ακέραιος n. Είναι πρώτος;
  - Σχετικά εύκολο. Ανήκει στο P όπως έδειξαν πρόσφατα κάποιο προπτυχιακοί Ινδοί φοιτητές.
- Factoring: Δίνεται ακέραιος n. Να βρεθούν οι πρώτοι παράγοντες του.
  - Δεν ξέρουμε αν είναι εύκολο ή δύσκολο. Πιστεύουμε ότι δεν είναι στο P, αλλά ούτε ότι είναι τόσο δύσκολο όσο τα NP-complete προβλήματα.
  - Για κβαντικούς υπολογιστές (που δεν έχουμε ακόμα καταφέρει να κατασκευάσουμε) ανήκει στο P.

Factoring και κρυπτογραφία

- RSA: Κρυπτογραφικό σχήμα για να στείλει η A (Alice) στον B (Bob) ένα μήνυμα m.
- Ο B διαλέγει 2 μεγάλους πρώτους αριθμούς p και q και ένα ακέραιο e. Υπολογίζει το γινόμενο n=pq.
- Ο B στέλνει στην A τα n και e.
- Η A στέλνει στον B τον αριθμό $c=m^e \pmod n$.
- Ο B υπολογίζει το m: $m=c^d \pmod n$, όπου το $d=e^{-1} \pmod{(p-1)(q-1)}$.
- Παράδειγμα: p=11, q=17, n=187, e=21, d=61, m=42, c=9

Αλγόριθμοι – πόσο γρήγοροι;

- Εκτός από κάποιες ειδικές περιπτώσεις, για κανένα πρόβλημα δεν γνωρίζουμε πόσο γρήγορα μπορεί να λυθεί.
- Ακόμα και για τον πολλαπλασιασμό αριθμών δεν γνωρίζουμε τον ταχύτερο αλγόριθμο.
- Ο σχολικός τρόπος πολλαπλασιασμού αριθμών με n ψηφία παίρνει $O(n^2)$ βήματα.
- Υπάρχουν καλύτεροι αλγόριθμοι που παίρνουν περίπου $O(n \log n)$ βήματα.
- Υπάρχει αλγόριθμος που παίρνει $O(n)$ βήματα; Αυτό είναι ανοικτό ερώτημα.

Κάποια σύγχρονα θέματα θεωρίας

- Διαδίκτυο (routing, congestion, game theory, cost allocation)
- Βιολογία (protein folding, genome, evolution).