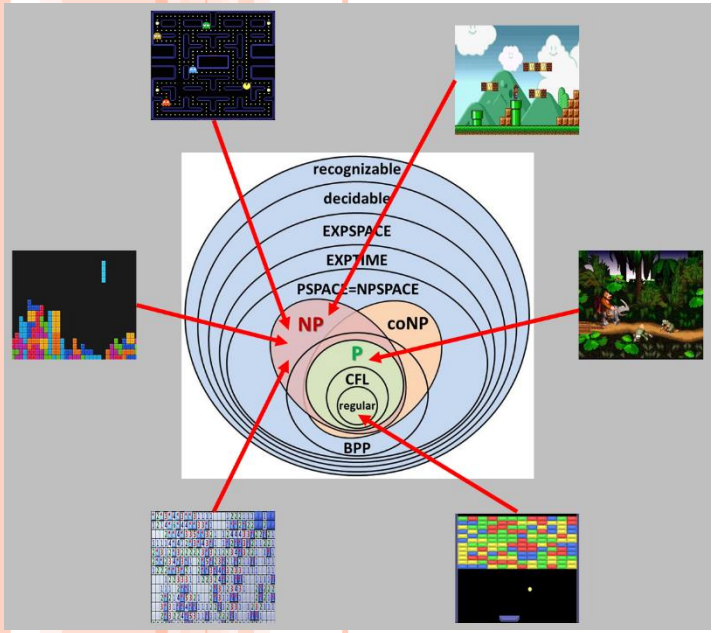
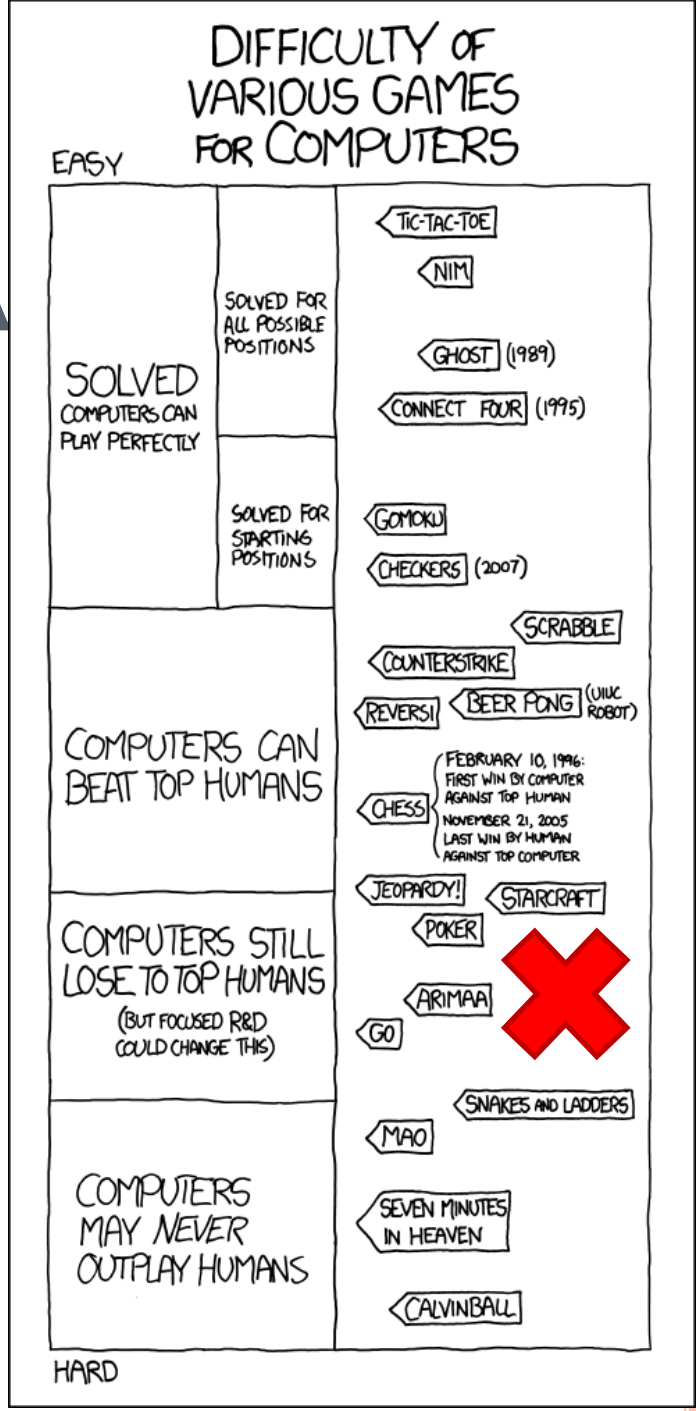
Υπολογιστική Πολυπλοκότητα

Υπολογισιμότητα – Πολυπλοκότητα



Εισαγωγή

Αν. Καθηγητής
Τσιχλας Κωνσταντινος



Σχετικα με το μαθημα…

- Βοηθός Μαθήματος: Χρήστος Κωνσταντόπουλος (st1079362@ceid.upatras.gr)
- Ώρες Γραφείου: Οποτεδήποτε (στείλτε email και κανονίζουμε) – Τρίτη 11-12
- email: ktsichlas@upatras.gr
- Δικτυακός Τόπος Μαθήματος: https://eclass.upatras.gr/courses/CEID1140/
- Ώρες Μαθήματος:
  - Πέμπτη 9:00-11:00 (Β)
  - Παρασκευή 12:00-14:00 (Β)
- Τρόπος Εξέτασης:
  - Τελική Εξέταση

Κάποιες Συμβουλές

- Διακόψτε με ερωτήσεις (μερικές φορές ξεφεύγω)
- Παρακολουθήστε τις διαλέξεις (ελπίζω να έχουν πλάκα)
- Λύστε τις ασκήσεις (ας είναι ήδη λυμένες)
- Αν δεν καταλαβαίνετε κάτι ελάτε στο γραφείο μου (ίσως το καταλάβουμε μαζί)

Κινούμενο χέρι: Θέλετε κάτι να πείτε σε σχέση με αυτά που ***λέω τώρα***

Σταθερό χέρι: Ερώτηση ή σχόλιο γενικής φύσης

Κάποιες Συμβουλές (Ίσχυαν επί COVID)

- Διακόψτε με ερωτήσεις (μερικές φορές ξεφεύγω)
- Παρακολουθήστε τις διαλέξεις (ελπίζω να έχουν πλάκα)
- Λύστε ασκήσεις (ας είναι ήδη λυμένες)
- Αν δεν καταλαβαίνετε κάτι στείλτε ένα email και είτε σας το εξηγώ ή κανονίζουμε ένα skype, ή ελάτε στο γραφείο μου (ίσως το καταλάβουμε μαζί)

Θέλετε κάτι να πείτε σε σχέση με αυτά που ***λέω τώρα***

Ερώτηση ή σχόλιο γενικής φύσης

Περι τινοσ δεν προκειται

«Οι Η/Υ δεν μπορούν να λύσουν τα πάντα»

Προφανή άλυτα προβλήματα:

- Πως θα γίνω εκατομμυριούχος; (;)
- Πώς θα ενοποιήσω την κβαντική θεωρία με την βαρύτητα; (;)
- Πώς θα δω αν υπάρχει Θεός; (;)

Τα προβλήματα αυτά δεν είναι σωστά ορισμένα.

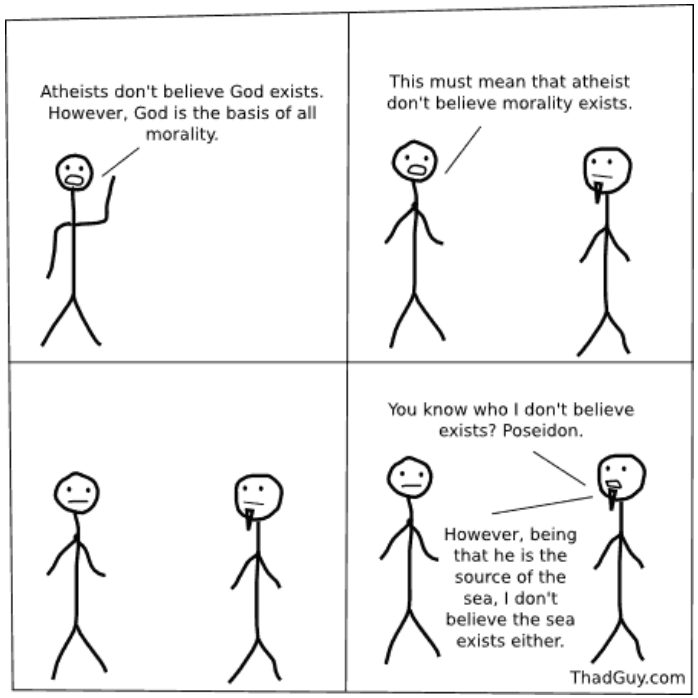
- Δεν είναι «**υπολογιστικά προβλήματα**»

- Ένα σωστά ορισμένο πρόβλημα θα πρέπει να περιγράφει την έξοδο για κάθε δυνατή είσοδο.

Περι τινοσ προκειται

- Πώς λύνουμε προβλήματα;
  Τεχνικές Απόδειξης
- Ποιοι είναι οι θεμελιώδεις περιορισμοί των Η/Υ;
  Υπολογισιμότητα/Αποφασισιμότητα
- Τι κάνει τα προβλήματα δύσκολα/εύκολα;
  Κατηγοριοποίηση Προβλημάτων
- Τι πόρους χρειαζόμαστε για να υπολογίσουμε κάτι;
  Χρόνος / Χώρος / «Υλικό» / Πολυπλοκότητα

Η Πραγματικοτητα



Η Υπολογιστική Πολυπλοκότητα είναι:

- Αποδείξεις
- Περισσότερες αποδείξεις
- Ακόμα περισσότερες αποδείξεις

και....

- Αποδείξεις (καθώς και κάποιες «ωραίες» ιδέες)

Αν και θεωρειται μαθηματικο μαθημα…

- Όχι παραγώγους ή ολοκληρώματα
- Όχι δυωνυμικοί συντελεστές
- Όχι πολύπλοκοι υπολογισμοί
- Όχι πιθανότητες
- Κυρίως: κάποια μαθηματική σημειογραφία και η δύναμη της λογικής σκέψης

Ελεγχος Απειριζουσας Επαναληψης

Είσαι βαθμολογητής στην Java

- Οι φοιτητές δίνουν μία εργασία foo.java
  - Έχεις ένα σύνολο αρχείων για έλεγχο 1.inp… 100.inp
  - Κάθε σωστό σημείο παίρνει 1 βαθμό
  - Για κάθε άπειρη επανάληψη η ποινή είναι 3 βαθμοί

- Θέλουμε να αυτοματοποιήσουμε την βαθμολόγηση
  - Γράφουμε ένα πρόγραμμα ILT.java με δύο εισόδους:
  - Τα αρχεία foo.java και i.inp – και ελέγχει αν

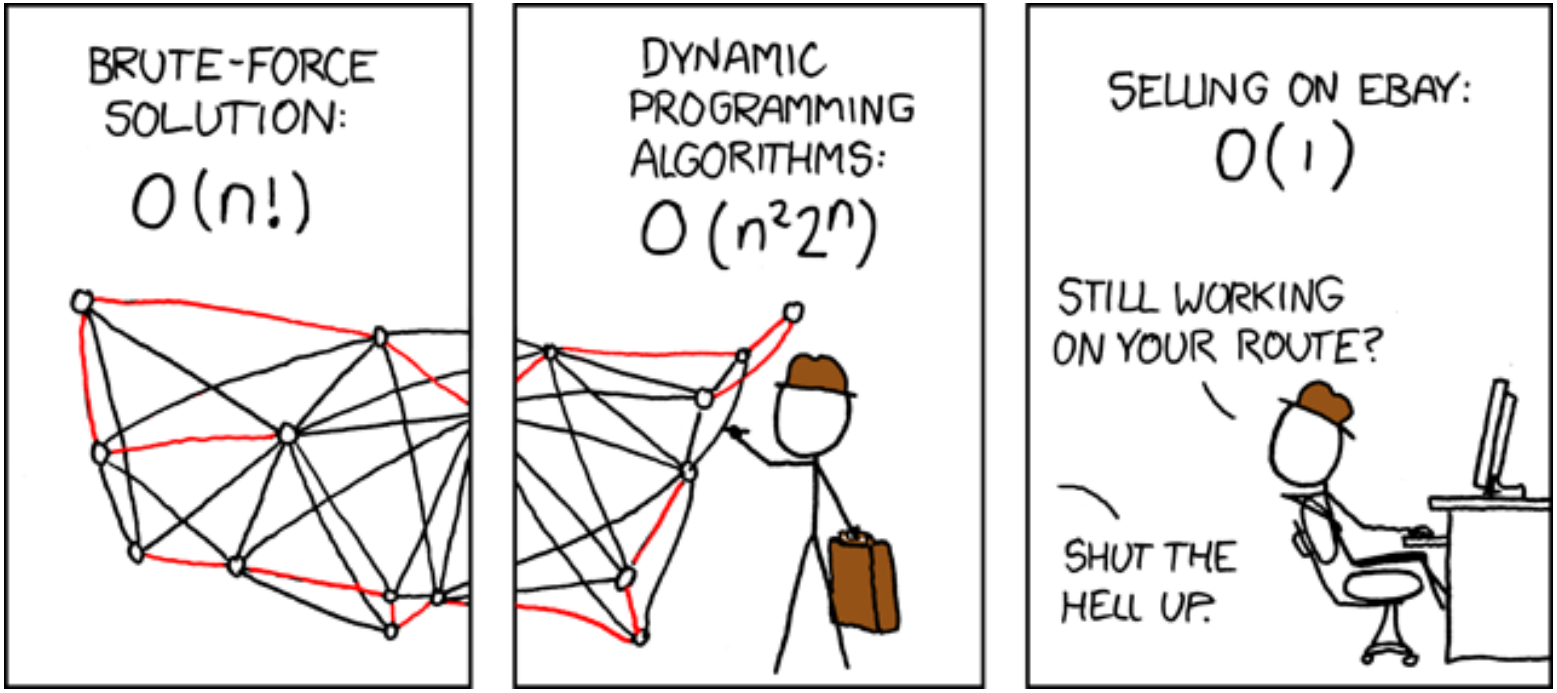
Το foo.java μπαίνει σε άπειρη επανάληψη με είσοδο i.inp

Δυστυχως (ευτυχως) ειναι αδυνατον (ΜΑΛΛΟΝ!!!!)

- Το πρόγραμμα ILT.java δεν μπορεί να γραφεί!!!

- Αυτό δεν οφείλεται σε κάποιο περιορισμό της Java, απλά η εργασία αυτή δεν μπορεί να γίνει από υπολογιστή
- Πολύ περίεργο που ένα καλώς ορισμένο υπολογιστικό πρόβλημα δεν μπορεί να λυθεί από υπολογιστή!!!!!

Το Πρόβλημα του Περιοδεύοντος Πωλητή



Μα Καλά, Κανείς Δεν το Έχει Λύσει Μέχρι Τώρα;

- Το TSP θεωρείται δύσκολο πρόβλημα
- «Φαίνεται» ότι υπάρχει κάποιο όριο στο πόσο γρήγορα υπολογίζουμε. Το όριο αυτό τίθεται από το μοντέλο υπολογισμού
- Προσέγγιση???
  - Ναι – για ευκλείδιες αποστάσεις (1+ε) από βέλτιστη (1996)
  - Για ασύμμετρες αποστάσεις $\frac{\log n}{\log \log n}$ από βέλτιστη (2010)

Θεωρια Υπολογισμου

Υπολογισιμότητα

- *Τι μπορούμε να υπολογίσουμε;*
- *Μπορεί ένας Η/Υ να λύσει οποιοδήποτε πρόβλημα δοθέντος αρκετού χρόνου και χώρου;*

Πολυπλοκότητα

- *Πόσο γρήγορα μπορούμε να λύσουμε ένα πρόβλημα;*
- *Πόσος χώρος χρειάζεται για να λύσουμε ένα πρόβλημα;*

Αυτόματα

- *Τι προβλήματα μπορούμε να λύσουμε με πολύ λίγο χώρο;*

Θεωρια Υπολογισμου

Υπολογισιμότητα

Πολυπλοκότητα

Αυτόματα

Τι προβλήματα λύνει ένας Η/Υ;

Όχι όλα!!!

π.χ. Δοθέντος ενός προγράμματος σε Java δεν μπορούμε να ελέγξουμε αν θα τερματίσει σωστά!

Ο έλεγχος ορθότητας προγραμμάτων είναι **αδύνατος**!

(Ο καημός της Microsoft και δικός μας!)

Θεωρια Υπολογισμου

Υπολογισιμότητα

Πολυπλοκότητα

Αυτόματα

Τι προβλήματα λύνει ένας Η/Υ;

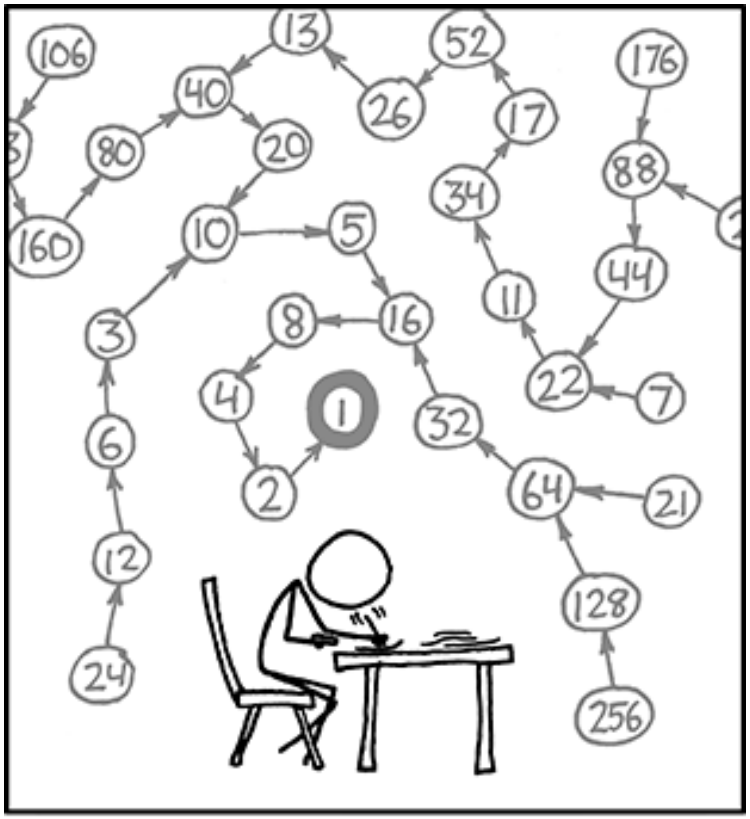
Ο έλεγχος τερματισμού είναι αδύνατος! Για παράδειγμα (Collatz Conjecture):

```
input n;
assume n>1;
while (n !=1) {
   if (n is even)
     n := n/2;
  else
     n := 3*n+1;
}
```

Κανείς δεν ξέρει αν αυτό το πρόγραμμα τερματίζει σε όλες τις εισόδους!

17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

Collatz Conjecture



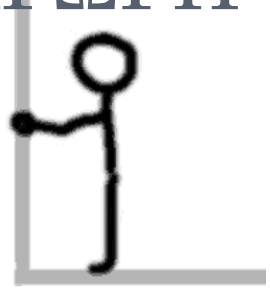
THE COLLATZ CONJECTURE STATES THAT IF YOU PICK A NUMBER, AND IF IT'S EVEN DIVIDE IT BY TWO AND IF IT'S ODD MULTIPLY IT BY THREE AND ADD ONE, AND YOU REPEAT THIS PROCEDURE LONG ENOUGH, EVENTUALLY YOUR FRIENDS WILL STOP CALLING TO SEE IF YOU WANT TO HANG OUT.

x=10?

Μια πρωτη Γευση Υπολογισιμοτητας

- Έστω $P(I)$ ένα πρόγραμμα, όπου P τρέχει σε είσοδο I.
- Έστω ότι το πρόγραμμα ILT (το java πρόγραμμα για έλεγχο τερματισμού)
- Το $ILT(P, I)$ απαντάει «άπειρη επανάληψη» αν το $P(I)$ μπαίνει σε άπειρη επανάληψη, διαφορετικά λέει «τερματίζει».
- Θα κατασκευάζουμε ένα πρόγραμμα ΚΛΑΨΕ (αποΚΛείεται νΑ δουλέΨΕι) χρησιμοποιώντας το πρόγραμμα ILT.

Αποδειξη με εις ατοπο απαγωγη



Το ΚΛΑΨΕ(P) κάνει τα εξής:

1. Καλεί $ILT(P, P)$.
2. Αν ILT απαντά «άπειρη επανάληψη», τότε τερμάτισε.
3. Αν ILT απαντά «τερματίζει», τότε μπες σε άπειρη επανάληψη.

Τι γίνεται αν τρέξουμε ΚΛΑΨΕ(ΚΛΑΨΕ);

- Αν η εκτέλεση τερματίσει, τότε στη γραμμή 1 ILT θα απαντούσε «τερματίζει», και άρα θα πηγαίναμε στην γραμμή 3 και…OOPS!
- Αν η εκτέλεση δεν τερματίσει, τότε μετά την γραμμή 1 θα πηγαίναμε στην γραμμή 2 και…OOPS!

Έχουμε **άτοπο** και άρα το ILT **δεν μπορεί να υπάρξει**

Θα την ξαναδουμε αργοτερα…

- Η απόδειξη αυτή είναι ένα από τα πιο σημαντικά και θεμελιώδη αποτελέσματα στην Επιστήμη της Πληροφορικής.

- Για να την καταλάβετε προσπαθήστε να την περιγράψετε σε κάποιον άλλον.

Θεωρια Υπολογισμου

Υπολογισιμότητα

Πολυπλοκότητα

Αυτόματα

- **Πόσο γρήγορα μπορούμε να υπολογίσουμε μία συνάρτηση;**
- **Πόσο χώρο χρειαζόμαστε;**

- Υπολογισμός πολυωνυμικού χρόνου
- Ανταιτιοκρατικός πολυωνυμικού χρόνου (NP)
- Προσέγγιση, Τυχαιότητα

Συναρτήσεις που δεν μπορούν να υπολογισθούν γρήγορα:

- Εφαρμογή σε ασφάλεια
  - Γρήγορη κρυπτογράφηση,
  - Η αποκρυπτογράφηση δεν είναι γρήγορη

- Κρυπτογραφία RSA

Ενα απλο παραδειγμα

7 x 11 = ?

Το πρόβλημα του

Πολλαπλασιασμού

(απάντηση: 77)

Ενα ακομα απλο παραδειγμα

? x ? = 77

Το πρόβλημα της

Παραγοντοποίησης

(απάντηση: 7,11)

Ενα πιο μεγαλο παραδειγμα

$$\begin{matrix}163473364580925384844313388386509085984178367003309231218111085238933310010450815121211816751157 9\end{matrix} \times \begin{matrix}1900871281664822113126851573935413975471896789968515493666638539088027103802104498957191261465571\end{matrix} = ?$$

Απάντηση:

3107418240490043721350750035888567930037346022842727545720161948823206440518081504556346829671723286782437916272838033415471073108501919548529007337724822783525742386454014691736602477652346609

Το αντιστροφο ομως;

? X ? = 3107418240490043721350750035888567930037346022842727545720161948823206440518081504556346829671723286782437916272838033415471073108501919548529007337724822783525742386540146917366024776523466 09

Αριθμός RSA 200. Οι παράγοντες βρέθηκαν μόλις το 2005 έπειτα από 5 μήνες ημερολογιακού χρόνου (80 AMD Opteron CPUs) – 20000$ το βραβείο

RSA Challenge 2048 – 617 ψηφια – 200.000$ βραβειο

25195908475657893494027183240048398571429282126204032027777**13783604366202070759555626401852588078440691829064124951508218929855914917618450280848912007284499268739280728777673597141834727026189637501497182469116507761337985909570009733045974880842840179742910064245869181719511874612151517265463228221686998754918242243363725908514186546204357679842338718477444792073993423658482382428119816381501067481045166037730605620161967625613384414360383390441495263443219011465754445417842402092461651572335077870774981712577246796292638635637328991215483143816789988504044536402352738195137863656439121201039712282212072035 7**

RSA-576	174	576	US$10,000	December 3, 2003
RSA-180 [*]	180	596		May 8, 2010
RSA-190 [*]	190	629		November 8, 2010
RSA-640	193	640	US$20,000	November 2, 2005
RSA-200 [*] ?	200	663		May 9, 2005
RSA-210 [*]	210	696		September 26, 2013[8]
RSA-704 [*]	212	704	US$30,000	July 2, 2012
RSA-220 [*]	220	729		May 13, 2016
RSA-230 [*]	230	762		August 15, 2018
RSA-232 [*]	232	768		February 17, 2020[9]
RSA-768 [*]	232	768	US$50,000	December 12, 2009
RSA-240 [*]	240	795		Dec 2, 2019[10]
RSA-250 [*]	250	829		Feb 28, 2020[11]
RSA-260	260	862		

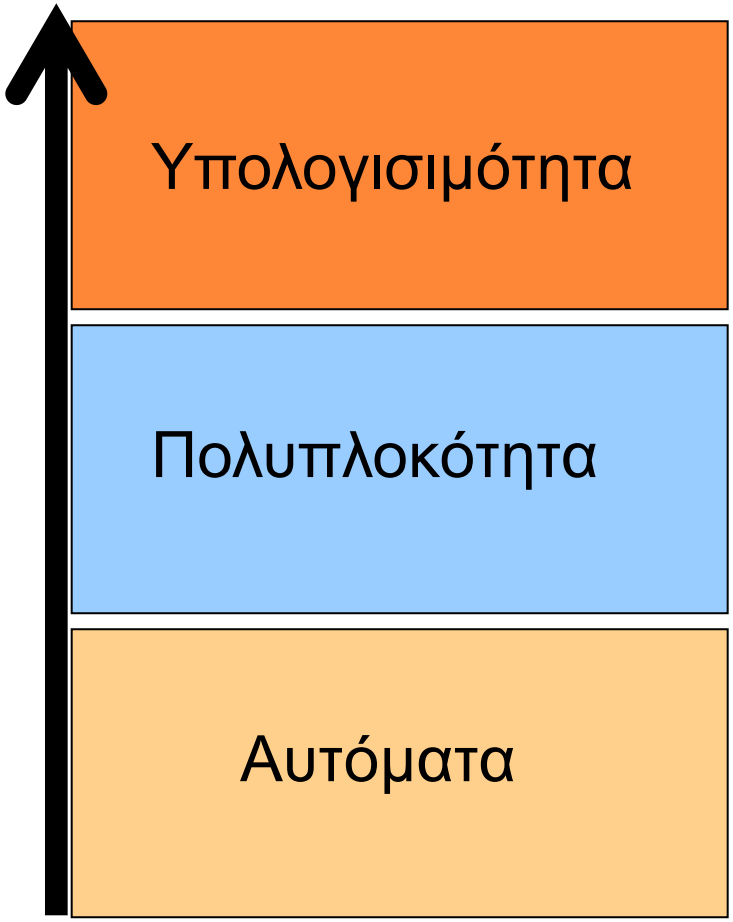
Θεωρια Υπολογισμου

Αυτόματα:
- Θεμελιώσεις υπολογισμού
- Μαθηματικές μέθοδοι
- Απλότητα

Έχουν ήδη γίνει στο μάθημα ***Θεωρία Υπολογισμού***.
Όπου χρειάζεται θα υπενθυμίζουμε.

ΑΥΞΗΜΕΝΗ ΠΟΛΥΠΛΟΚΟΤΗΤΑ

Θεωρια Υπολογισμου



Τι υπολογίζουμε;
- Γενικές έννοιες υπολογισιμότητας
- Μη υπολογίσιμες συναρτήσεις

Τι μπορούμε να υπολογίσουμε γρήγορα;
- Γρήγοροι αλγόριθμοι, πολυωνυμικός χρόνος
- Προβλήματα που δεν λύνονται γρήγορα
- Κρυπτογραφία

Τι υπολογίζουμε με μικρό χώρο;
- Σταθερός χώρος (+σωρός)
- Εύρεση αλφαριθμητικών, επαλήθευση υλικού κτλ.

Θεωρια Υπολογισμου

ΑΥΞΗΜΕΝΗ ΠΟΛΥΠΛΟΚΟΤΗΤΑ

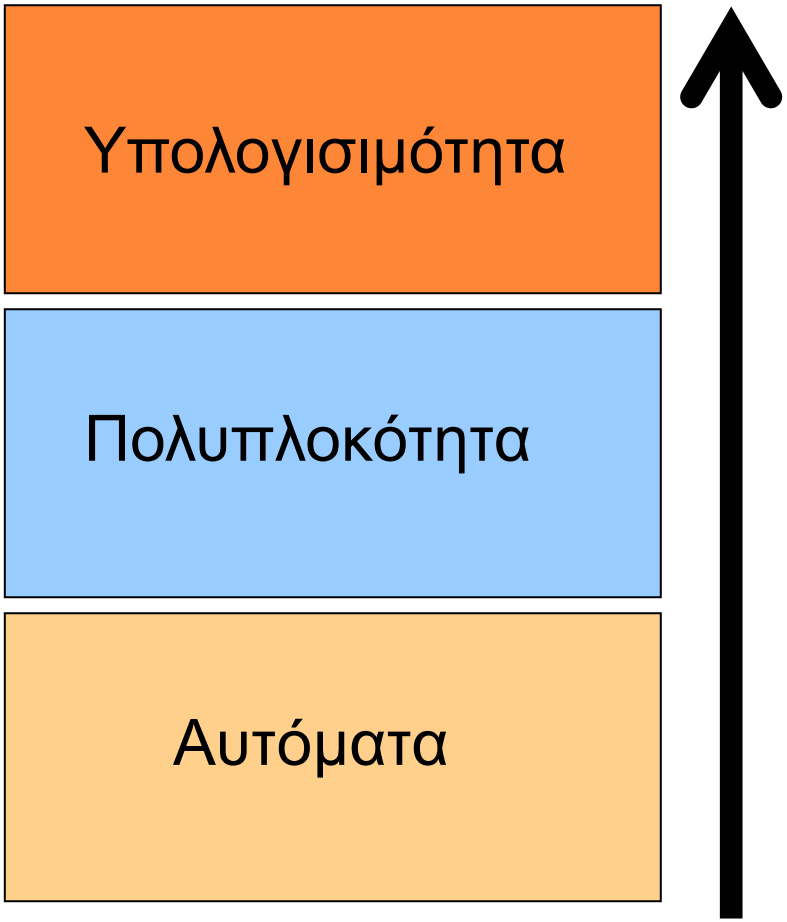
Μηχανές Turing (Δεκαετία ’40)

- Η πιο γενική έννοια υπολογισμού
- Η θέση των Church-Turing
- Περιορισμοί στον υπολογισμό:
  Μη υπολογίσιμες συναρτήσεις

Κίνητρο από Μαθηματικά:

- Μπορούμε να λύσουμε μαθηματικά προβλήματα μεθοδολογικά;
- Το θεώρημα του Godel: **ΟΧΙ!!!!**
- Ακόμα και οι πιο δυνατοί υπολογιστές δεν μπορούν να λύσουν κάποια προβλήματα

Θεωρια Υπολογισμου

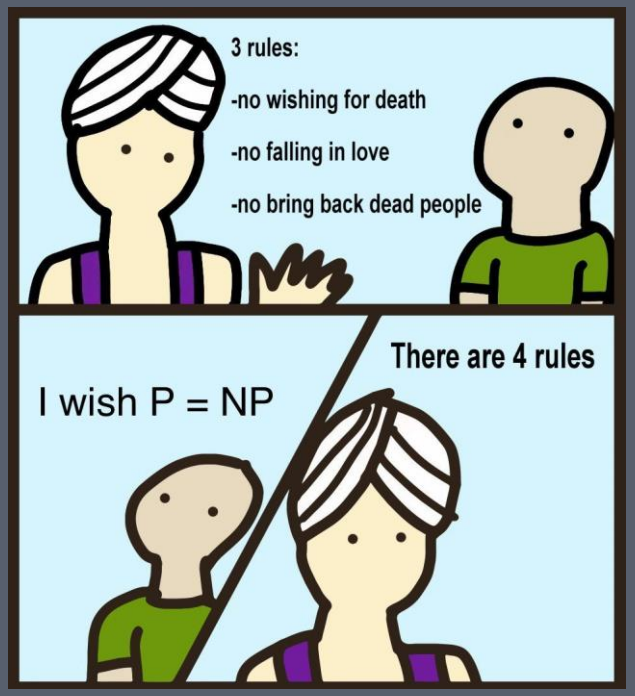


Πολυπλοκότητα: Εβδομάδα 7-13

Υπολογισιμότητα:Εβδομάδα 2-7

Μηχανές Turing: Εβδομάδα 1-2

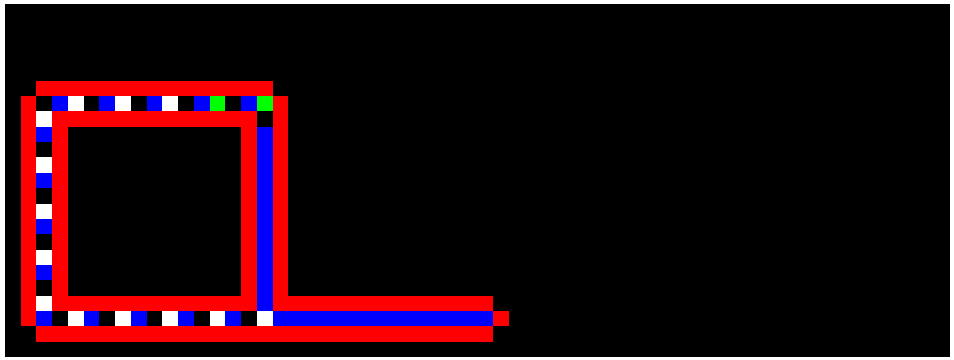
Εισαγωγή: Σήμερα



Κάτι που έχει πλάκα από «άλλο» μάθημα

Θεώρημα Αναδρομής

1. Οι ζωντανοί οργανισμοί είναι μηχανές
2. Οι ζωντανοί οργανισμοί μπορούν να αυτοαναπαράγονται
3. Οι μηχανές δεν μπορούν να αυτοαναπαράγονται



Quines

Πώς ένα πρόγραμμα εκτυπώνει τον εαυτό του;

```
main()
{
printf(“Hello World”);
}
```

```
main()
{
printf(“main() { printf(\”Hello World\”);}
}
```

??????????

Quines

http://www.nyx.net/~gthompso/quine.htm

Ένα παράδειγμα.

Ένα πιο απλό:

```
#include <stdio.h>
main(){char *c="#include <stdio.h> main(){char *c=%c%s%c;printf(c,34,c,34);}";printf(c,34,c,34);}
```

Διαλογικά Αποδεικτικά Συστήματα (Πιθανοκρατικά)

Η Marla έχει μία κόκκινη κάλτσα και μία κίτρινη κάλτσα. Ο φίλος της ο Arthur έχει αχρωματοψία και δεν την πιστεύει ότι οι κάλτσες έχουν διαφορετικό χρώμα. Πώς θα τον πείσει ότι πράγματι έχουν διαφορετικό χρώμα;

Επίσης: Αποδείξεις με Μηδενική Γνώση (Zero-Knowledge Proofs) - Αυθεντικοποίηση

http://www.wisdom.weizmann.ac.il/~naor/PAPERS/sudoku_abs.html

Δύο περιπολίες συναντιούνται σε ένα στρατόπεδο χωρίς να ξέρει ο ένας για τον άλλο ότι κάνουν περιπολία. Και οι δύο ξέρουν το σύνθημα που είναι ένας ακέραιος αριθμός στο διάστημα [1,52]. Τι θα κάνουν αν:

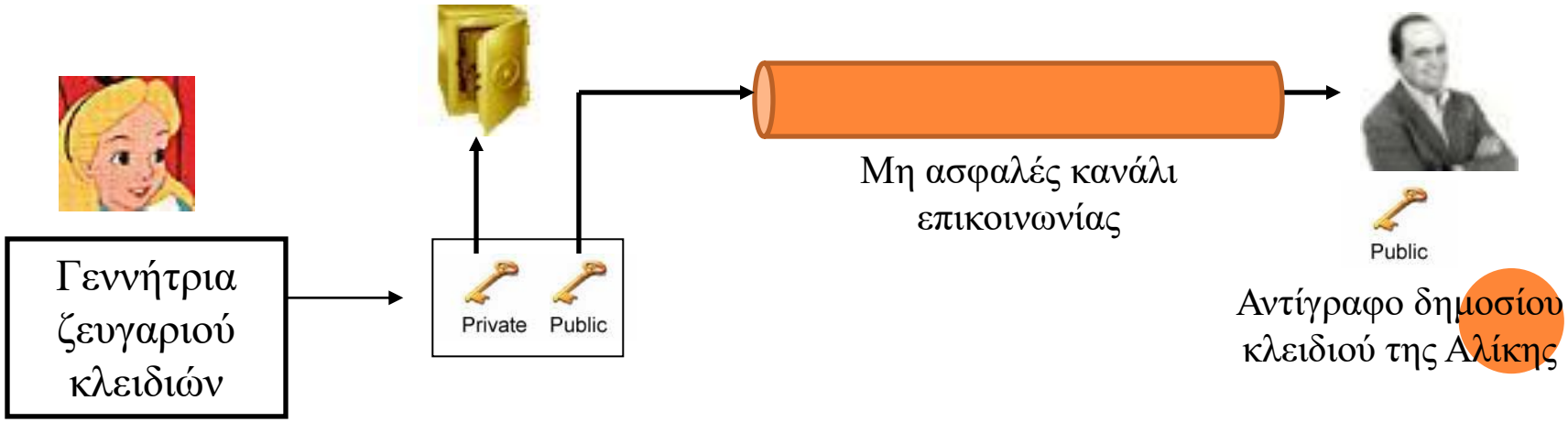
1. Έχουν μία τράπουλα και μία τσάντα.
2. Αν ένας σκαστός φαντάρος εμφανίστηκε μπροστά τους και βάλουν αυτόν να αποφασίσει αν είναι γνήσια περίπολα και όχι κατάσκοποι χωρίς και αυτός να μάθει τον αριθμό.

Κρυπτογραφία Δημοσίου Κλειδίου

- Παραλήπτης (Αλίκη) παράγει ένα ζευγάρι κλειδιών:
  - Δημόσιο Κλειδί $\mathbf{K_{PU}}$ 

    - Δεν χρειάζεται να είναι κρυφό
    - Δίνεται σε όλους τους αποστολείς (όπως ο Μπομπ)
  - Ιδιωτικό Κλειδί $\mathbf{K_{PR}}$ 

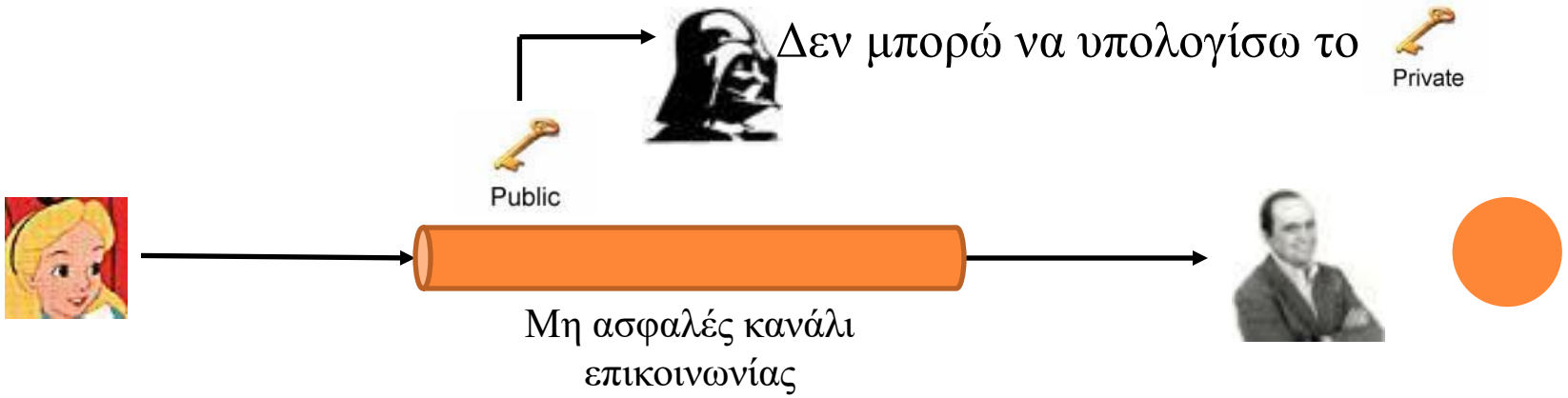
    - Είναι κρυφό και το γνωρίζει μόνο η Αλίκη



Βασικές Αρχές

Ο αντίπαλος δεν μπορεί να υπολογίσει το ιδιωτικό κλειδί από το αντίστοιχο δημόσιο κλειδί

- Θεωρητικά μπορεί αλλά είναι υπολογιστικά ανέφικτο
- Δεν μπορεί να διαβάσει τα μηνύματα που είναι κρυπτογραφημένα με το δημόσιο κλειδί



Και κατι πιο εξωτικο

- Εναλλακτικά μοντέλα Υπολογισμού:
  - Φούσκες σαπουνιού
  - Κβαντικός Υπολογισμός
  - Αναλογικός Υπολογισμός και Σχετικότητα
  - Κβαντική Βαρύτητα
  - Ταξίδι στον Χρόνο
  - …

Ύλη (1)

Θεωρία Υπολογισιμότητας

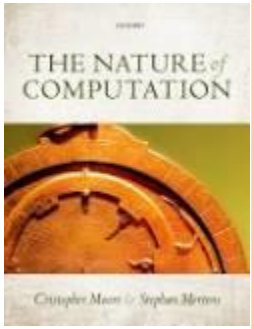
- Church-Turing
- Διαγνωσιμότητα
- Αναγωγές
- Σύνθετα Ζητήματα
  - Θεώρημα Αναδρομής
  - Διαγνωσιμότητα Λογικών Θεωριών
  - Εισαγωγή στην Πολυπολοκότητα κατά Kolmogorov (Μ.Ε.)

Ύλη (2)

Πολυπλοκότητα

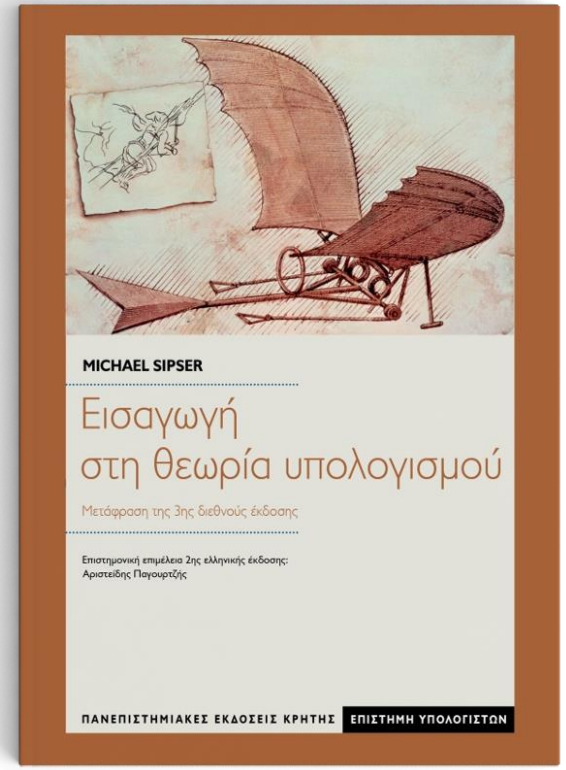
- Χρονική πολυπλοκότητα
- Χωρική Πολυπλοκότητα (Μ.Ε.)
- Θεώρημα Ιεραρχίας, Σχετικοποίηση, Κυκλώματα (Μ.Ε.)
- Προσεγγιστικοί Αλγόριθμοι (Μ.Ε.)
- Πιθανοκρατικοί Αλγόριθμοι
- Διαλογικά Αποδεικτικά Συστήματα (Μ.Ε.)
- Παράλληλος Υπολογισμός
- Κρυπτογραφία – Ψευδοτυχαιτότητα (Μ.Ε.)

The Nature of Computation (Μ.Ε.)



Βιβλιο

Εισαγωγή στη Θεωρία Υπολογισμού. Μ. Sipser.

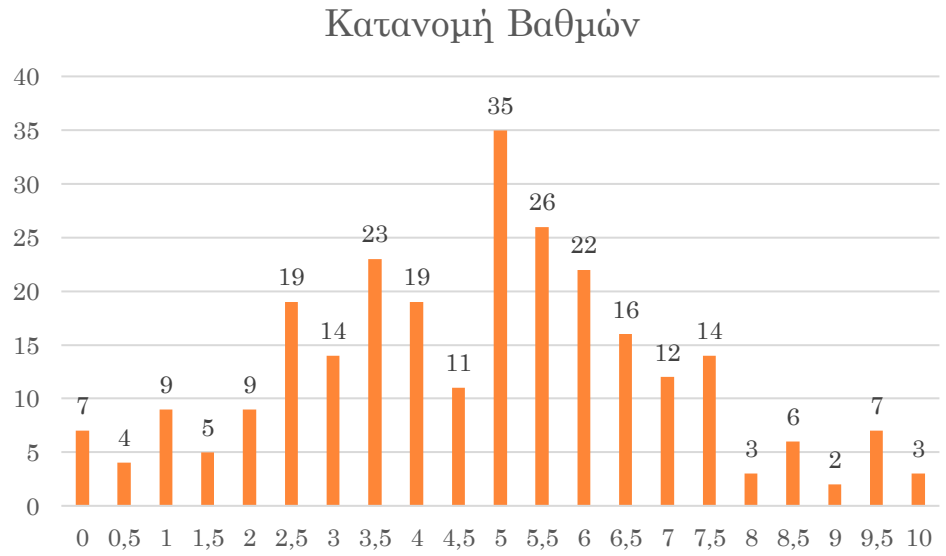


Μετα από Όσα Είπαμε: Στόχοι Μαθήματος

- Να κατανοήσουμε ότι ο Η/Υ έχει όρια.
  - Δεν μπορούμε να τα λύσουμε όλα.
- Να κατανοήσουμε ότι τα προβλήματα έχουν μία εγγενή δυσκολία.
  - Ακόμα και αυτά που λύνονται επί της αρχής μπορεί να απαιτούν πολύ χρόνο
- Να δώσουμε εργαλεία κατηγοριοποίησης
- Να πιέσουμε τους εαυτούς μας να σκεφτούν και να κατανοήσουν κάτι πολύπλοκο. ☺

Αξιολόγηση

- Τελική Εξέταση
  - Στόχος: να καταλάβω ότι έχετε κατανοήσει κάποιες βασικές έννοιες.
- Εξετάσεις Ιουνίου 2023.
  - Ποσοστό Επιτυχίας: 54,9%



Όχι άλλο.....
Φτάνει!!!!!!!!!

Αυτόματα και Γλώσσες

Μία πολύ μικρή υπενθύμιση (για δικιά σας χρήση)

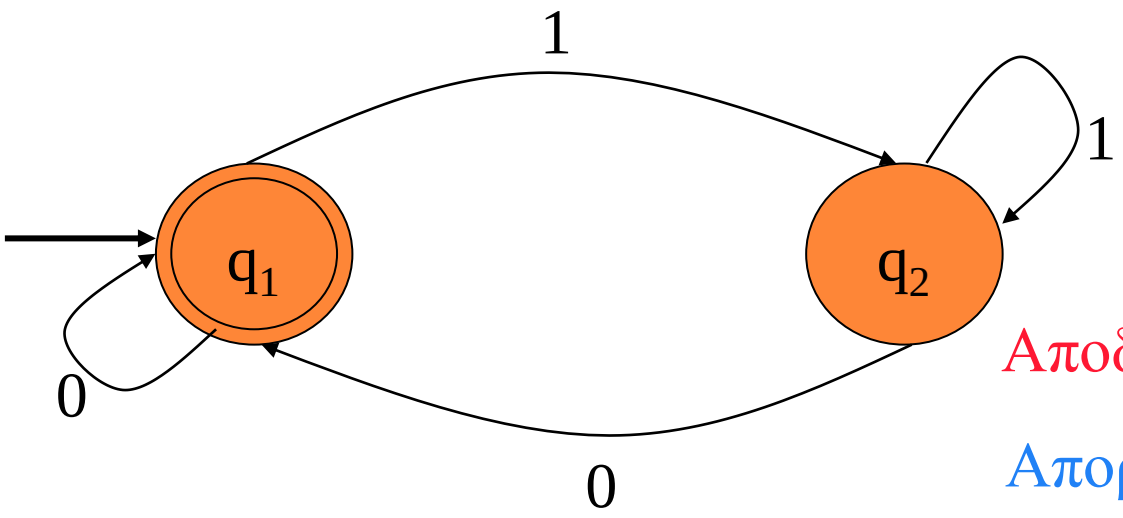
Η ΈΝΝΟΙΑ ΤΟΥ ΥΠΟΛΟΓΙΣΤΙΚΟΎ ΜΟΝΤΈΛΟΥ

- Οι υπολογιστές είναι ετερογενείς και αρκετά πολύπλοκοι ώστε να τους αναλύσουμε μαθηματικά απευθείας.

- Το υπολογιστικό μοντέλο είναι ένας ιδεατός υπολογιστής, ο οποίος όμως εμπεριέχει τα θεμελιώδη χαρακτηριστικά των πραγματικών υπολογιστών

Αυτόματο

- Μοντελοποιεί έναν υπολογιστή με εξαιρετικά περιορισμένη μνήμη.



Αποδέχεται: 11110

Απορρίπτει: 11111

Κανονικές Εκφράσεις

- Κανονικές πράξεις που επιτρέπουν την αναπαράσταση γλωσσών.

Έστω Σ={0,1}. Τότε:

- $\Sigma^*\{1\}\Sigma^* = \{w|$ η w περιέχει τουλάχιστον ένα 1$\}$ (Σώρευση)
- $(\Sigma\Sigma\Sigma)^* = \{w|$ το μήκος του w είναι πολλαπλάσιο του 3$\}$ (Συναρμογή)
- $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$ (Ένωση)

Παράδειγμα

Α={καλό,κακό} και Β={αγόρι,κορίτσι}

Α ∪ Β={καλό,κακό,αγόρι,κορίτσι}

ΑΒ={καλόαγόρι,κακόαγόρι,καλόκορίτσι,κακόκορίτσι}

Α*={ε,καλό,κακό,καλόκακό,καλόκαλό,κακόκαλό,κακόκακό,καλόκακόκαλό,…}

Ασυμφραστικές Γραμματικές

Έχουν την δυνατότητα αναδρομικών ορισμών.

Παράδειγμα:

A→0A1

A→B

B→#

π.χ. 000000#111111

Αυτόματα Στοίβας

- Η αναγνώριση των ασυμφραστικών γραμματικών γίνεται με τα αυτόματα στοίβας
- Αυτά είναι αυτόματα με την επιπλέον δυνατότητα χρήσης μίας στοίβας άπειρου μεγέθους
- Πιο ισχυρά από τα απλά αυτόματα