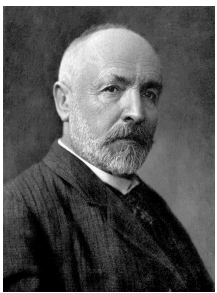
Μια κληρονομιά του Cantor

Διαγωνιοποίηση (Diagonalization)



Διαγωνιοποίηση: διαφέρουμε από κάθε στοιχείο μιας λίστας

Στόχος: να κατασκευάσουμε ένα αντικείμενο που δεν εμφανίζεται σε μια δοσμένη λίστα.

Η ιδέα είναι να εξασφαλίσουμε μια διαφορετική θέση διαφωνίας για κάθε στοιχείο της:

- Διαφέρουμε από το πρώτο στοιχείο στην πρώτη θέση.
- Διαφέρουμε από το δεύτερο στοιχείο στη δεύτερη θέση.
- Γενικά, διαφέρουμε από το *i*-οστό στοιχείο στην *i*-οστή θέση.

Μία διαφορά με κάθε στοιχείο αρκεί για να μην είμαστε κανένα από αυτά.

Η τεχνική ονομάζεται **διαγωνιοποίηση**, επειδή χρησιμοποιούμε τη διαγώνιο ενός πίνακα.

Ένα πεπερασμένο παράδειγμα: προτιμήσεις σε παιχνίδια

Πέντε φοιτητές δηλώνουν ποια παιχνίδια τούς αρέσουν.

	Σκάκι	Sudoku	Minecraft	Tetris	FIFA
Άννα	**Ναι**	Όχι	Ναι	Όχι	Ναι
Νίκος	Ναι	**Όχι**	Όχι	Ναι	Ναι
Ελένη	Όχι	Ναι	**Ναι**	Ναι	Όχι
Πέτρος	Ναι	Όχι	Ναι	**Ναι**	Όχι
Μαρία	Όχι	Ναι	Όχι	Ναι	**Όχι**

Ερώτημα: μπορούμε να κατασκευάσουμε ένα προφίλ προτιμήσεων διαφορετικό από καθένα από τα πέντε;

Ιδέα: αντιστρέφουμε τις απαντήσεις της διαγωνίου.

Η διαγώνιος δίνει ένα νέο προφίλ

	Σκάκι	Sudoku	Minecraft	Tetris	FIFA
Άννα	**Ναι**	Όχι	Ναι	Όχι	Ναι
Νίκος	Ναι	**Όχι**	Όχι	Ναι	Ναι
Ελένη	Όχι	Ναι	**Ναι**	Ναι	Όχι
Πέτρος	Ναι	Όχι	Ναι	**Ναι**	Όχι
Μαρία	Όχι	Ναι	Όχι	Ναι	**Όχι**
Νέο προφίλ	**Όχι**	**Ναι**	**Όχι**	**Όχι**	**Ναι**

Το νέο προφίλ διαφέρει:

- από της Άννας στο Σκάκι και του Νίκου στο Sudoku,
- από της Ελένης στο Minecraft και του Πέτρου στο Tetris,
- από της Μαρίας στο FIFA.

Μία διαφορά με κάθε γραμμή αρκεί: το νέο προφίλ δεν είναι καμία από τις υπάρχουσες γραμμές.

Τι ακριβώς εξασφαλίζει η κατασκευή;

Κωδικοποιούμε το «Ναι» με 1 και το «Όχι» με 0.
Αν a_{ij} είναι η απάντηση του i-οστού φοιτητή για το j-οστό παιχνίδι, ορίζουμε:

$$b_i = 1 - a_{ii}, \qquad i = 1, \ldots, 5.$$

Στο παράδειγμά μας:

$$\underbrace{(1,0,1,1,0)}_{\text{διαγώνιος}} \longmapsto \underbrace{(0,1,0,0,1)}_{\text{νέο προφίλ}}.$$

Για κάθε γραμμή i, το νέο προφίλ διαφέρει από αυτήν στη στήλη i.

Δεν χρειάζεται να διαφέρει σε όλες τις στήλες. Μία στήλη για κάθε γραμμή αρκεί.

Κατασκευάσαμε ένα δυνατό προφίλ προτιμήσεων. Δεν αποδείξαμε ότι υπάρχει φοιτητής με αυτό το προφίλ.

Από τις προτιμήσεις στις άπειρες δυαδικές ακολουθίες

Έστω μια λίστα άπειρων δυαδικών ακολουθιών:

	1	2	3	4	$\cdots$
s_1	a_{11}	a_{12}	a_{13}	a_{14}	$\cdots$
s_2	a_{21}	a_{22}	a_{23}	a_{24}	$\cdots$
s_3	a_{31}	a_{32}	a_{33}	a_{34}	$\cdots$
s_4	a_{41}	a_{42}	a_{43}	a_{44}	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

όπου $a_{ij} \in \{0, 1\}$.

Κατασκευάζουμε την ακολουθία $b = b_1 b_2 b_3 \cdots$ θέτοντας

$$\boxed{b_i = 1 - a_{ii}.}$$

Για κάθε i, έχουμε $b_i \neq a_{ii}$, επομένως $b \neq s_i$.

Η νέα ακολουθία δεν εμφανίζεται πουθενά στη λίστα.

Οπτικοποίηση της ακολουθίας

Ας δούμε την κατασκευή με συγκεκριμένα ψηφία:

	1	2	3	4	5	6	⋯
s_1	**0**	1	1	0	1	0	⋯
s_2	1	**1**	0	1	0	0	⋯
s_3	0	0	**1**	1	0	1	⋯
s_4	1	0	1	**0**	1	1	⋯
s_5	0	1	0	0	**0**	1	⋯
s_6	1	1	0	1	0	**1**	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱
b	**1**	**0**	**0**	**1**	**1**	**0**	⋯

Αντιστρέφουμε κάθε ψηφίο της διαγωνίου:

$$011001\cdots \quad \longmapsto \quad \boxed{b = 100110\cdots}$$

Γιατί αυτό αποδεικνύει μη αριθμησιμότητα;

Ένα άπειρο σύνολο είναι **αριθμήσιμο** αν τα στοιχεία του μπορούν να μπουν σε μια πλήρη λίστα

$$s_1, s_2, s_3, \ldots$$

Υποθέτουμε προς άτοπο ότι υπάρχει τέτοια πλήρης λίστα για όλες τις άπειρες δυαδικές ακολουθίες. Γράφουμε

$$s_i = (a_{i1}, a_{i2}, a_{i3}, \ldots), \qquad a_{ij} \in \{0, 1\}.$$

Κατασκευάζουμε μία νέα ακολουθία

$$b = (b_1, b_2, b_3, \ldots), \qquad b_i = 1 - a_{ii}.$$

Εφόσον η λίστα υποτίθεται ότι είναι πλήρης, θα υπήρχε κάποιος $k \geq 1$ τέτοιος ώστε $b = s_k$. Όμως, στη θέση k, η ακολουθία s_k έχει ψηφίο a_{kk}, ενώ

$$b_k = 1 - a_{kk} \neq a_{kk}.$$

Άρα $b \neq s_k$, αντίφαση.

Συμπέρασμα: το σύνολο $\{0, 1\}^{\mathbb{N}^+}$ είναι μη αριθμήσιμο.

Το δυναμοσύνολο των φυσικών αριθμών

Το

$$2^{\mathbb{N}} = \mathcal{P}(\mathbb{N})$$

είναι το σύνολο όλων των υποσυνόλων των φυσικών αριθμών.

Για παράδειγμα, τα παρακάτω είναι στοιχεία του $2^{\mathbb{N}}$:

$$\varnothing, \qquad \{0, 2, 4, 6, \ldots\}, \qquad \{1, 4, 9, 16, \ldots\}, \qquad \mathbb{N}.$$

Θέλουμε να απαντήσουμε:

Μπορούμε να βάλουμε όλα τα υποσύνολα της $\mathbb{N}$ σε μια λίστα

$$R_0, R_1, R_2, \ldots ?$$

Θεώρημα. Το $2^{\mathbb{N}}$ δεν είναι αριθμήσιμο.

Υποθέτουμε ότι υπάρχει πλήρης λίστα

Προς άτοπο, υποθέτουμε ότι η λίστα

$$R_0, R_1, R_2, \ldots$$

περιέχει κάθε υποσύνολο της $\mathbb{N}$.

Για κάθε γραμμή R_i, ρωτάμε αν ο αριθμός j ανήκει σε αυτήν:

	0	1	2	3	$\cdots$
R_0	•		•		$\cdots$
R_1		•		•	$\cdots$
R_2	•	•			$\cdots$
R_3			•	•	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

Η διαγώνιος αποτελείται από τις ερωτήσεις

$$0 \in R_0, \quad 1 \in R_1, \quad 2 \in R_2, \quad \ldots$$

Κατασκευάζουμε το διαγώνιο σύνολο

Ορίζουμε

$$D = \{\, n \in \mathbb{N} : n \notin R_n \,\}.$$

Με απλά λόγια, για κάθε n:

- αν $n \in R_n$, τότε δεν βάζουμε το n στο D,
- αν $n \notin R_n$, τότε βάζουμε το n στο D.

Άρα το D διαφέρει από κάθε R_k στο στοιχείο k:

$$k \in D \quad \Longleftrightarrow \quad k \notin R_k.$$

Επομένως

$$D \neq R_0, \quad D \neq R_1, \quad D \neq R_2, \quad \ldots$$

Εφόσον $D \subseteq \mathbb{N}$, το D είναι ένα από τα υποσύνολα της $\mathbb{N}$. Άρα, αν η λίστα

$$R_0, R_1, R_2, \ldots$$

ήταν πλήρης, θα έπρεπε να υπάρχει κάποιος $k \geq 0$ τέτοιος ώστε $D = R_k$.

Η αντίφαση: πού ακριβώς βρίσκεται;

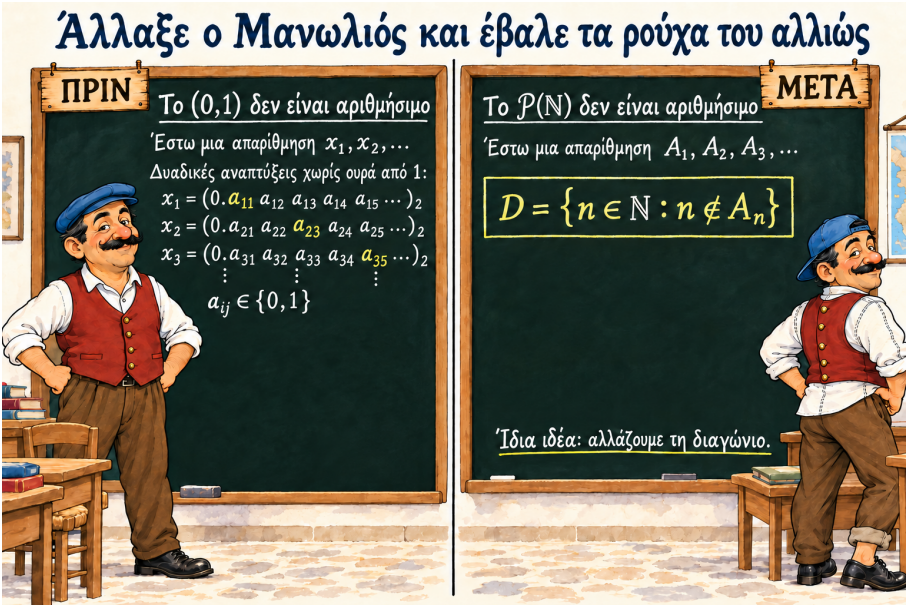
Τώρα ρωτάμε την κρίσιμη ερώτηση:

$$\boxed{\text{Ισχύει } k \in R_k;}$$

1. Αν $k \in D = R_k$, τότε, από τον ορισμό του D,
$$k \notin D.$$
Όμως $D = R_k$, άρα $k \notin R_k$: αντίφαση.
2. Αν $k \notin R_k$, τότε, από τον ορισμό του D,
$$k \in D.$$
Όμως $D = R_k$, άρα $k \in R_k$: αντίφαση.

Άρα δεν υπάρχει πλήρης απαρίθμηση των στοιχείων του $2^{\mathbb{N}}$.

Άλλαξε ο Μανωλιός και έβαλε τα ρούχα του αλλιώς



Πού κρύβεται η διαγωνιοποίηση στην απόδειξη του Halting problem;

Στη διαγωνιοποίηση κατασκευάζουμε ένα αντικείμενο που διαφέρει από το *i*-οστό αντικείμενο στην *i*-οστή θέση.

Στην απόδειξη για το πρόβλημα τερματισμού:

- τα αντικείμενα είναι **προγράμματα**,
- οι θέσεις αντιστοιχούν σε **εισόδους**,
- η ιδιότητα που αντιστρέφουμε είναι «**τερματίζει / δεν τερματίζει**».

Κατασκευάζουμε ένα πρόγραμμα που, στην είσοδο του *i*-οστού προγράμματος, συμπεριφέρεται αντίθετα από εκείνο όταν εκτελείται στον εαυτό του.

- Σημαντική σημείωση: τα προγράμματα είναι ναι μεν άπειρα, αλλά αριθμήσιμα. Ας κάνουμε μια παρένθεση, να δούμε γιατί.

Κάθε πρόγραμμα είναι μια πεπερασμένη λέξη

Ιδέα: ο πηγαίος κώδικας ενός προγράμματος είναι κείμενο **πεπερασμένου μήκους**.

Κάθε τέτοιο κείμενο μπορεί να κωδικοποιηθεί ως μια πεπερασμένη δυαδική λέξη:

$$P \quad \longmapsto \quad \mathsf{code}(P) \in \{0,1\}^*.$$

Επιλέγουμε μια κωδικοποίηση στην οποία διαφορετικά κείμενα προγραμμάτων έχουν διαφορετικούς κωδικούς.

Προσοχή στη διάκριση

Το πρόγραμμα έχει **πεπερασμένη περιγραφή**, ακόμη κι αν η εκτέλεσή του συνεχίζεται για πάντα!

```
while True: pass
```

Άρα αρκεί να δείξουμε ότι μπορούμε να βάλουμε **όλες τις πεπερασμένες δυαδικές λέξεις σε μία λίστα**.

Πώς ένα πρόγραμμα γίνεται δυαδική λέξη;

Εύλογο ερώτημα: Τί σημαίνει $P \longmapsto \text{code}(P) \in \{0,1\}^*$.;

Θεωρήστε το πρόγραμμα: $P =$ `print(1)`

Κάθε χαρακτήρας έχει μια δυαδική κωδικοποίηση. Για τους παρακάτω χαρακτήρες, η UTF-8 χρησιμοποιεί 8 bits ανά χαρακτήρα:

Χαρακτήρας	Κωδικοποίηση
`p`	01110000
`r`	01110010
`i`	01101001
`n`	01101110
`t`	01110100
`(`	00101000
`1`	00110001
`)`	00101001

Ενώνουμε τα bits με τη σειρά:

$$\text{code}(P) = \underbrace{01110000}_{\texttt{p}}\,\underbrace{01110010}_{\texttt{r}}\,\underbrace{01101001}_{\texttt{i}}\cdots\underbrace{00101001}_{\texttt{)}}.$$

Πεπερασμένο κείμενο $\Rightarrow$ πεπερασμένη δυαδική λέξη.

Πώς βάζουμε όλα τα προγράμματα σε μία λίστα;

Απαριθμούμε τις δυαδικές λέξεις **κατά μήκος** και, για κάθε μήκος, **λεξικογραφικά**:

$$\underbrace{\varepsilon}_{\text{μήκος } 0},\quad \underbrace{0,\ 1}_{\text{μήκος } 1},\quad \underbrace{00,\ 01,\ 10,\ 11}_{\text{μήκος } 2},\quad \underbrace{000,\ 001,\ \ldots,\ 111}_{\text{μήκος } 3},\quad \ldots$$

- Κάθε λέξη μήκους n εμφανίζεται έπειτα από **πεπερασμένα βήματα**: υπάρχουν μόνο

$$\sum_{k=0}^{n-1} 2^k = 2^n - 1$$

  λέξεις μικρότερου μήκους.
- Κρατάμε όσες λέξεις κωδικοποιούν συντακτικά έγκυρα προγράμματα: $P_1,\ P_2,\ P_3,\ \ldots$

Συμπέρασμα

Τα προγράμματα είναι αριθμήσιμα.

Τα προγράμματα μπορούν να χρησιμοποιηθούν ως δεδομένα

Κάθε πρόγραμμα P έχει μια πεπερασμένη περιγραφή $code(P)$. Συμβολίζουμε το $code(P)$ για συντομία με

$$\langle P \rangle$$

την κωδικοποίηση του προγράμματος P ως συμβολοσειρά.

Μπορούμε να καταγράψουμε όλα τα προγράμματα:

$$P_1, P_2, P_3, \ldots$$

Η έκφραση

$$P_i(\langle P_j \rangle)$$

σημαίνει: εκτελούμε το πρόγραμμα P_i με είσοδο την περιγραφή του προγράμματος P_j.

Ειδικά,

$$P_i(\langle P_i \rangle)$$

σημαίνει ότι δίνουμε στο P_i τη δική του περιγραφή ως είσοδο.

Ο πίνακας τερματισμού

Ορίζουμε μαθηματικά:

$$h_{ij} = \begin{cases} 1, & \text{αν το } P_i(\langle P_j \rangle) \text{ τερματίζει,} \\ 0, & \text{αν δεν τερματίζει.} \end{cases}$$

	$\langle P_1 \rangle$	$\langle P_2 \rangle$	$\langle P_3 \rangle$	$\langle P_4 \rangle$	$\cdots$
P_1	h_{11}	h_{12}	h_{13}	h_{14}	$\cdots$
P_2	h_{21}	h_{22}	h_{23}	h_{24}	$\cdots$
P_3	h_{31}	h_{32}	h_{33}	h_{34}	$\cdots$
P_4	h_{41}	h_{42}	h_{43}	h_{44}	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

- **Γραμμή *i*:** το πρόγραμμα που εκτελούμε.
- **Στήλη *j*:** η περιγραφή που δίνουμε ως είσοδο.
- **Διαγώνιος:** κάθε πρόγραμμα με είσοδο τον εαυτό του.

Ο πίνακας είναι καλά ορισμένος. Δεν έχουμε υποθέσει ότι μπορούμε να υπολογίσουμε τις τιμές του.

Η υπόθεση που θα οδηγήσει σε αντίφαση

Υποθέτουμε ότι υπάρχει αλγόριθμος H ο οποίος, για κάθε πρόγραμμα P και είσοδο x, **τερματίζει πάντα** και επιστρέφει:

$$H(\langle P\rangle, x) = \begin{cases} 1, & \text{αν το } P(x) \text{ τερματίζει,} \\ 0, & \text{αν το } P(x) \text{ δεν τερματίζει.} \end{cases}$$

Τότε μπορούμε να υπολογίσουμε οποιοδήποτε διαγώνιο στοιχείο:

$$h_{ii} = H(\langle P_i\rangle, \langle P_i\rangle).$$

Το επόμενο βήμα της διαγωνιοποίησης:

Αντιστρέφουμε τη συμπεριφορά πάνω στη διαγώνιο.

Η διαγώνια κατασκευή προγράμματος

Προς άτοπο, υποθέτουμε ότι το halts υπάρχει.

Κατασκευάζουμε ένα νέο πρόγραμμα diagonal, το οποίο δέχεται ως είσοδο ένα πρόγραμμα X:

diagonal(X) :
if $H(\langle X\rangle, \langle X\rangle) = 1$ **then**
loop για πάντα
else
τερμάτισε.

Δηλαδή το diagonal(X) κάνει το αντίθετο από αυτό που προβλέπει το υποθετικό πρόγραμμα H για το X όταν τρέχει με είσοδο τον εαυτό του.

Τι γίνεται στο diagonal(diagonal);

Θέτουμε ως είσοδο στο πρόγραμμα τον ίδιο του τον κώδικα:

$$\text{diagonal}(\text{diagonal}).$$

1. Αν
$$H(\langle \text{diagonal} \rangle, \langle \text{diagonal} \rangle) = \text{yes},$$
τότε το diagonal εκτελείται για πάντα.
2. Αν
$$H(\text{diagonal}, \text{diagonal}) = \text{no},$$
τότε το diagonal τερματίζει.

Άρα το πρόγραμμα τερματίζει αν και μόνο αν δεν τερματίζει:

$$\text{diagonal}(\text{diagonal}) \text{ τερματίζει} \iff \text{diagonal}(\text{diagonal}) \text{ δεν τερματίζει}.$$

Αυτό είναι αδύνατο.

Συμπέρασμα: δεν υπάρχει καθολικός ελεγκτής τερματισμού

Η αντίφαση προήλθε μόνο από την υπόθεση ότι υπάρχει ο αλγόριθμος για το halting problem, που τον καλούμε ως function H.

Άρα:

> Δεν υπάρχει αλγόριθμος που αποφασίζει το Halting Problem.

Η δομή είναι ακριβώς η ίδια με πριν:

	Μη αριθμησιμότητα του $2^{\mathbb{N}}$	Halting
Υποθέτουμε	πλήρη λίστα συνόλων	αλγόριθμο H
Αντιστρέφουμε	τη διαγώνιο	την πρόβλεψη για $H(\langle X\rangle, \langle X\rangle)$
Κατασκευάζουμε	σύνολο εκτός λίστας	πρόγραμμα που διαψεύδει τον ελεγκτή

Η διαγωνιοποίηση δεν είναι τέχνασμα: είναι μηχανή κατασκευής ενός αντικειμένου που ξεφεύγει από κάθε υποτιθέμενα πλήρη περιγραφή.