

Bonyolultságelmélet

Monday 26th September, 2016, 19:02

Dacára egyszerű szintaxisának, a RAM-gép is **Turing-teljes** számítási modell: minden „hagyományos” programozási nyelven megvalósítható algoritmusra létezik RAM program is.

Univerzális RAM-gép

Dacára egyszerű szintaxisának, a RAM-gép is **Turing-teljes** számítási modell: minden „hagyományos” programozási nyelven megvalósítható algoritmusra létezik RAM program is.

Így például RAM interpretert is írhatunk RAM nyelven...

Univerzális RAM-gép

Dacára egyszerű szintaxisának, a RAM-gép is **Turing-teljes** számítási modell: minden „hagyományos” programozási nyelven megvalósítható algoritmusra létezik RAM program is.

Így például RAM interpretert is írhatunk RAM nyelven...

Tétel

Létezik olyan U **univerzális RAM-program**, amelynek ha adott egy M RAM-program és annak egy x bemenete, úgy viselkedik, mint M az x -en, vagyis $U(M; x) = M(x)$.

Univerzális RAM-gép

Dacára egyszerű szintaxisának, a RAM-gép is **Turing-teljes** számítási modell: minden „hagyományos” programozási nyelven megvalósítható algoritmusra létezik RAM program is.

Így például RAM interpretert is írhatunk RAM nyelven...

Tétel

Létezik olyan U **univerzális RAM-program**, amelynek ha adott egy M RAM-program és annak egy x bemenete, úgy viselkedik, mint M az x -en, vagyis $U(M; x) = M(x)$.

Természetesen M kódolva adott U számára (utasítások, tömbcímkézések... számokkal).

A megállási probléma – első eldönthetetlen problémánk

MEGÁLLÁS

- Input: M program, x input.
- Output: Megáll-e M az x -en futtatva?

A megállási probléma – első eldönthetetlen problémánk

MEGÁLLÁS

- Input: M program, x input.
- Output: Megáll-e M az x -en futtatva?

Tétel

A MEGÁLLÁS probléma eldönthetetlen.

A megállási probléma – első eldönthetetlen problémánk

MEGÁLLÁS

- Input: M program, x input.
- Output: Megáll-e M az x -en futtatva?

Tétel

A MEGÁLLÁS probléma eldönthetetlen.

Bizonyítás

A megállási probléma – első eldönthetetlen problémánk

MEGÁLLÁS

- Input: M program, x input.
- Output: Megáll-e M az x -en futtatva?

Tétel

A MEGÁLLÁS probléma eldönthetetlen.

Bizonyítás

Tegyük fel, hogy MEGÁLLÁS eldönthető egy M_0 programmal.

A megállási probléma – első eldönthetetlen problémánk

MEGÁLLÁS

- Input: M program, x input.
- Output: Megáll-e M az x -en futtatva?

Tétel

A MEGÁLLÁS probléma eldönthetetlen.

Bizonyítás

Tegyük fel, hogy MEGÁLLÁS eldönthető egy M_0 programmal.
Legyen D a következő (RAM-programot váró) program:

$$D(M) = \text{if } M_0(M; M) \text{ then } \nearrow \text{ else halt.}$$

A megállási probléma – első eldönthetetlen problémánk

MEGÁLLÁS

- Input: M program, x input.
- Output: Megáll-e M az x -en futtatva?

Tétel

A MEGÁLLÁS probléma eldönthetetlen.

Bizonyítás

Tegyük fel, hogy MEGÁLLÁS eldönthető egy M_0 programmal.
Legyen D a következő (RAM-programot váró) program:

$$D(M) = \text{if } M_0(M; M) \text{ then } \nearrow \text{ else halt.}$$

Ekkor:

$$D(D) = \nearrow \Leftrightarrow M_0(D; D) = \text{„igen”} \Leftrightarrow D(D) \neq \nearrow,$$

ami ellentmondás.

Ha már ismerünk egy eldönthetetlen problémát, másokról is meg tudjuk mutatni, hogy eldönthetetlenek.

Visszavezetés: eldönthetlenség bizonyítása

Ha már ismerünk egy eldönthetetlen problémát, másokról is meg tudjuk mutatni, hogy eldönthetetlenek.

Hogyan?

Azt használjuk fel, hogy ha

Ha már ismerünk egy eldönthetetlen problémát, másokról is meg tudjuk mutatni, hogy eldönthetetlenek.

Hogyan?

Azt használjuk fel, hogy ha

- A eldönthetetlen és

Ha már ismerünk egy eldönthetetlen problémát, másokról is meg tudjuk mutatni, hogy eldönthetetlenek.

Hogyan?

Azt használjuk fel, hogy ha

- A eldönthetetlen és
- $A \leq_{\mathcal{R}} B$,

Ha már ismerünk egy eldönthetetlen problémát, másokról is meg tudjuk mutatni, hogy eldönthetetlenek.

Hogyan?

Azt használjuk fel, hogy ha

- A eldönthetetlen és
- $A \leq_{\mathcal{R}} B$,

akkor B is eldönthetetlen.

Állítás

Az alábbi probléma algoritmikusan eldönthetetlen.

Állítás

Az alábbi probléma algoritmikusan eldönthetetlen.

- **Adott:** M RAM-program.

Állítás

Az alábbi probléma algoritmikusan eldönthetetlen.

- **Adott:** M RAM-program.
- **Kérdés:** M megáll-e minden bemenetén?

Állítás

Az alábbi probléma algoritmikusan eldönthetetlen.

- **Adott:** M RAM-program.
- **Kérdés:** M megáll-e minden bemenetén?

Bizonyítás

Adott M és x esetén legyen M_x olyan program, hogy az M_x minden y inputjára:

$$M_x(y) = M(x).$$

Állítás

Az alábbi probléma algoritmikusan eldönthetetlen.

- **Adott:** M RAM-program.
- **Kérdés:** M megáll-e minden bemenetén?

Bizonyítás

Adott M és x esetén legyen M_x olyan program, hogy az M_x minden y inputjára:

$$M_x(y) = M(x).$$

(azaz: M_x tetszőleges y input esetén futtassa M -et x -en.)

Állítás

Az alábbi probléma algoritmikusan eldönthetetlen.

- **Adott:** M RAM-program.
- **Kérdés:** M megáll-e minden bemenetén?

Bizonyítás

Adott M és x esetén legyen M_x olyan program, hogy az M_x minden y inputjára:

$$M_x(y) = M(x).$$

(azaz: M_x tetszőleges y input esetén futtassa M -et x -en.)

Így

$$M(x) \neq \nearrow \Leftrightarrow M_x \text{ megáll minden bemenetén.}$$

Állítás

Az alábbi probléma algoritmikusan eldönthetetlen.

- **Adott:** M RAM-program.
- **Kérdés:** M megáll-e minden bemenetén?

Bizonyítás

Adott M és x esetén legyen M_x olyan program, hogy az M_x minden y inputjára:

$$M_x(y) = M(x).$$

(azaz: M_x tetszőleges y input esetén futtassa M -et x -en.)

Így

$$M(x) \neq \nearrow \Leftrightarrow M_x \text{ megáll minden bemenetén.}$$

Ez az $M; x \mapsto M_x$ hozzárendelés kiszámítható, tehát rekurzív visszavezetés; és mivel az eldönthetetlen MEGÁLLÁS problémát visszavezettük erre a problémára, ez is eldönthetetlen.

ÜRES INPUTON MEGÁLLÁS

Legyen ÜRES INPUTON MEGÁLLÁS a következő probléma:

- Input: egy M program.
- Output: megáll-e M az üres inputon (azaz mikor minden input regisztert 0-val inicializálunk)?

ÜRES INPUTON MEGÁLLÁS

Legyen ÜRES INPUTON MEGÁLLÁS a következő probléma:

- Input: egy M program.
- Output: megáll-e M az üres inputon (azaz mikor minden input regisztert 0-val inicializálunk)?

Eldönthetetlen!

Visszavezetjük rá a MEGÁLLÁS problémát.

ÜRES INPUTON MEGÁLLÁS

Legyen ÜRES INPUTON MEGÁLLÁS a következő probléma:

- Input: egy M program.
- Output: megáll-e M az üres inputon (azaz mikor minden input regisztert 0-val inicializálunk)?

Eldönthetetlen!

Visszavezetjük rá a MEGÁLLÁS problémát.

Adott M -hez és x -hez legyen M_x az a program, ami tetszőleges y inputra

ÜRES INPUTON MEGÁLLÁS

Legyen ÜRES INPUTON MEGÁLLÁS a következő probléma:

- Input: egy M program.
- Output: megáll-e M az üres inputon (azaz mikor minden input regisztert 0-val inicializálunk)?

Eldönthetetlen!

Visszavezetjük rá a MEGÁLLÁS problémát.

Adott M -hez és x -hez legyen M_x az a program, ami tetszőleges y inputra

- ha y az üres input, akkor futtatja M -et x -en;

ÜRES INPUTON MEGÁLLÁS

Legyen ÜRES INPUTON MEGÁLLÁS a következő probléma:

- Input: egy M program.
- Output: megáll-e M az üres inputon (azaz mikor minden input regisztert 0-val inicializálunk)?

Eldönthetetlen!

Visszavezetjük rá a MEGÁLLÁS problémát.

Adott M -hez és x -hez legyen M_x az a program, ami tetszőleges y inputra

- ha y az üres input, akkor futtatja M -et x -en;
- egyébként mondjuk megáll.

ÜRES INPUTON MEGÁLLÁS

Legyen ÜRES INPUTON MEGÁLLÁS a következő probléma:

- Input: egy M program.
- Output: megáll-e M az üres inputon (azaz mikor minden input regisztert 0-val inicializálunk)?

Eldönthetetlen!

Visszavezetjük rá a MEGÁLLÁS problémát.

Adott M -hez és x -hez legyen M_x az a program, ami tetszőleges y inputra

- ha y az üres input, akkor futtatja M -et x -en;
- egyébként mondjuk megáll.

Ez az M_x az M ; x párból elkészíthető és nyilván
 M megáll x -en $\Leftrightarrow M_x$ megáll üres inputon.

ÜRES INPUTON MEGÁLLÁS

Legyen ÜRES INPUTON MEGÁLLÁS a következő probléma:

- Input: egy M program.
- Output: megáll-e M az üres inputon (azaz mikor minden input regisztert 0-val inicializálunk)?

Eldönthetetlen!

Visszavezetjük rá a MEGÁLLÁS problémát.

Adott M -hez és x -hez legyen M_x az a program, ami tetszőleges y inputra

- ha y az üres input, akkor futtatja M -et x -en;
- egyébként mondjuk megáll.

Ez az M_x az M ; x párból elkészíthető és nyilván

M megáll x -en $\Leftrightarrow M_x$ megáll üres inputon.

Tehát $\text{MEGÁLLÁS} \leq_{\mathcal{R}} \text{ÜRES INPUTON MEGÁLLÁS}$, így ez utóbbi is eldönthetetlen probléma.

A MEGÁLLÁS, MINDENEN MEGÁLLÁS és ÜRESSZÓN MEGÁLLÁS problémák eldönthetetlensége szerint. . .

A MEGÁLLÁS, MINDENEN MEGÁLLÁS és ÜRESSZÓN MEGÁLLÁS problémák eldönthetetlensége szerint. . .

- **nincs** olyan algoritmus, mely inputként kap egy **JAVA forráskódot** és hozzá egy **inputot**, és megválaszolja, hogy a megadott program megáll-e a megadott inputon;

A MEGÁLLÁS, MINDENEN MEGÁLLÁS és ÜRESSZÓN MEGÁLLÁS problémák eldönthetetlensége szerint. . .

- **nincs** olyan algoritmus, mely inputként kap egy **JAVA forráskódot** és hozzá egy **inputot**, és megválaszolja, hogy a megadott program megáll-e a megadott inputon;
- **nincs** olyan algoritmus, mely inputként kap egy forráskódot és megválaszolja, hogy a program eshet-e végtelen ciklusba egyáltalán,

A MEGÁLLÁS, MINDENEN MEGÁLLÁS és ÜRESSZÓN MEGÁLLÁS problémák eldönthetetlensége szerint. . .

- **nincs** olyan algoritmus, mely inputként kap egy **JAVA forráskódot** és hozzá egy **inputot**, és megválaszolja, hogy a megadott program megáll-e a megadott inputon;
- **nincs** olyan algoritmus, mely inputként kap egy forráskódot és megválaszolja, hogy a program eshet-e végtelen ciklusba egyáltalán, vagy akár csak azt, hogy paraméter nélkül elindítva végtelen ciklusba fog-e esni!

Mit láttunk eddig?

A MEGÁLLÁS, MINDENEN MEGÁLLÁS és ÜRESSZÓN MEGÁLLÁS problémák eldönthetetlensége szerint. . .

- **nincs** olyan algoritmus, mely inputként kap egy **JAVA forráskódot** és hozzá egy **inputot**, és megválaszolja, hogy a megadott program megáll-e a megadott inputon;
- **nincs** olyan algoritmus, mely inputként kap egy forráskódot és megválaszolja, hogy a program eshet-e végtelen ciklusba egyáltalán, vagy akár csak azt, hogy paraméter nélkül elindítva végtelen ciklusba fog-e esni!

A helyzet azonban ennél sokkal rosszabb. . .

Definíció

Egy A probléma **rekurzívan felsorolható** (vagy Turing-felismerhető), ha van olyan M RAM-program, amire

$$M(x) = \begin{cases} \text{ACCEPT} & , \text{ ha } x \in A \end{cases}$$

Definíció

Egy A probléma **rekurzívan felsorolható** (vagy Turing-felismerhető), ha van olyan M RAM-program, amire

$$M(x) = \begin{cases} \text{ACCEPT} & , \text{ ha } x \in A \\ \nearrow & , \text{ egyébként.} \end{cases}$$

Definíció

Egy A probléma **rekurzívan felsorolható** (vagy Turing-felismerhető), ha van olyan M RAM-program, amire

$$M(x) = \begin{cases} \text{ACCEPT} & , \text{ ha } x \in A \\ \nearrow & , \text{ egyébként.} \end{cases}$$

A rekurzíván felsorolható problémák osztályát **RE** jelöli.

Definíció

Egy A probléma **rekurzívan felsorolható** (vagy Turing-felismerhető), ha van olyan M RAM-program, amire

$$M(x) = \begin{cases} \text{ACCEPT} & , \text{ ha } x \in A \\ \nearrow & , \text{ egyébként.} \end{cases}$$

A rekurzíván felsorolható problémák osztályát **RE** jelöli.

Nyilván $\mathbf{R} \subseteq \mathbf{RE}$.

Definíció

Egy A probléma **rekurzívan felsorolható** (vagy Turing-felismerhető), ha van olyan M RAM-program, amire

$$M(x) = \begin{cases} \text{ACCEPT} & , \text{ ha } x \in A \\ \nearrow & , \text{ egyébként.} \end{cases}$$

A rekurzívan felsorolható problémák osztályát **RE** jelöli.

Nyilván $\mathbf{R} \subseteq \mathbf{RE}$. (Hiszen egy RAM programot ACCEPT-től különböző válasz **helyett** végtelen ciklusba egyszerű küldeni.)

Tétel

$$\mathbf{R} \subsetneq \mathbf{RE}.$$

Tétel

$$\mathbf{R} \subsetneq \mathbf{RE}.$$

Bizonyítás

Az eldönthetetlen MEGÁLLÁS probléma rekurzívan felsorolható, pl. azzal a RAM-programmal, mely futtatja az univerzális RAM-programot az input $M; x$ páron, majd ha az megáll, ACCEPT választ ad.

A „rosszabb helyzet”: Rice tétele

Rice tétele

A rekurzívan felsorolható problémák egyetlen nemtriviális tulajdonsága sem dönthető el algoritmikusan.

A „rosszabb helyzet”: Rice tétele

Rice tétele

A rekurzívan felsorolható problémák egyetlen nemtriviális tulajdonsága sem dönthető el algoritmikusan.

Vagyis: ha $\emptyset \neq \mathcal{C} \subsetneq \mathbf{RE}$ a rekurzívan felsorolható problémák egy nemtriviális osztálya, akkor a következő probléma eldönthetetlen: adott egy M program, igaz-e, hogy $L(M) \in \mathcal{C}$?

A „rosszabb helyzet”: Rice tétele

Rice tétele

A rekurzívan felsorolható problémák egyetlen nemtriviális tulajdonsága sem dönthető el algoritmikusan.

Vagyis: ha $\emptyset \neq \mathcal{C} \subsetneq \mathbf{RE}$ a rekurzívan felsorolható problémák egy nemtriviális osztálya, akkor a következő probléma eldönthetetlen: adott egy M program, igaz-e, hogy $L(M) \in \mathcal{C}$?

Másképp mondva: automatikusan **semmi nemtriviálisat** nem tudunk eldönteni egy program **viselkedéséről** a **forráskódjának** ismeretében.

A „rosszabb helyzet”: Rice tétele

Rice tétele

A rekurzívan felsorolható problémák egyetlen nemtriviális tulajdonsága sem dönthető el algoritmikusan.

Vagyis: ha $\emptyset \neq \mathcal{C} \subsetneq \mathbf{RE}$ a rekurzívan felsorolható problémák egy nemtriviális osztálya, akkor a következő probléma eldönthetetlen: adott egy M program, igaz-e, hogy $L(M) \in \mathcal{C}$?

Másképp mondva: automatikusan **semmi nemtriviálisat** nem tudunk eldönteni egy program **viselkedéséről** a **forráskódjának** ismeretében.

Ezért válnak szükségessé például

A „rosszabb helyzet”: Rice tétele

Rice tétele

A rekurzívan felsorolható problémák egyetlen nemtriviális tulajdonsága sem dönthető el algoritmikusan.

Vagyis: ha $\emptyset \neq \mathcal{C} \subsetneq \mathbf{RE}$ a rekurzívan felsorolható problémák egy nemtriviális osztálya, akkor a következő probléma eldönthetetlen: adott egy M program, igaz-e, hogy $L(M) \in \mathcal{C}$?

Másképp mondva: automatikusan **semmi nemtriviálisat** nem tudunk eldönteni egy program **viselkedéséről** a **forráskódjának** ismeretében.

Ezért válnak szükségessé például kommentek,

A „rosszabb helyzet”: Rice tétele

Rice tétele

A rekurzívan felsorolható problémák egyetlen nemtriviális tulajdonsága sem dönthető el algoritmikusan.

Vagyis: ha $\emptyset \neq \mathcal{C} \subsetneq \mathbf{RE}$ a rekurzívan felsorolható problémák egy nemtriviális osztálya, akkor a következő probléma eldönthetetlen: adott egy M program, igaz-e, hogy $L(M) \in \mathcal{C}$?

Másképp mondva: automatikusan **semmi nemtriviálisat** nem tudunk eldönteni egy program **viselkedéséről** a **forráskódjának** ismeretében.

Ezért válnak szükségessé például kommentek, tervezési minták használata,

A „rosszabb helyzet”: Rice tétele

Rice tétele

A rekurzívan felsorolható problémák egyetlen nemtriviális tulajdonsága sem dönthető el algoritmikusan.

Vagyis: ha $\emptyset \neq \mathcal{C} \subsetneq \mathbf{RE}$ a rekurzívan felsorolható problémák egy nemtriviális osztálya, akkor a következő probléma eldönthetetlen: adott egy M program, igaz-e, hogy $L(M) \in \mathcal{C}$?

Másképp mondva: automatikusan **semmi nemtriviálisat** nem tudunk eldönteni egy program **viselkedéséről** a **forráskódjának** ismeretében.

Ezért válnak szükségessé például kommentek, tervezési minták használata, statikus és dinamikus tesztelés. . .

További eldönthetetlen problémák

További eldönthetetlen problémák

Hilbert 10. problémája

További eldönthetetlen problémák

Hilbert 10. problémája

- **Adott:** $p(x_1, \dots, x_n)$ egész együtthatós polinom.

További eldönthetetlen problémák

Hilbert 10. problémája

- **Adott:** $p(x_1, \dots, x_n)$ egész együtthatós polinom.
- **Kérdés:** Létezik-e egész értékű zérushelye?

További eldönthetetlen problémák

Hilbert 10. problémája

- **Adott:** $p(x_1, \dots, x_n)$ egész együtthatós polinom.
- **Kérdés:** Létezik-e egész értékű zérushelye?

Tétel (Matijasevič)

Hilbert 10. problémája algoritmikusan megoldhatatlan.

További eldönthetetlen problémák

Hilbert 10. problémája

- **Adott:** $p(x_1, \dots, x_n)$ egész együtthatós polinom.
- **Kérdés:** Létezik-e egész értékű zérushelye?

Tétel (Matijasevič)

Hilbert 10. problémája algoritmikusan megoldhatatlan.

Post megfelelezési problémája

További eldönthetetlen problémák

Hilbert 10. problémája

- **Adott:** $p(x_1, \dots, x_n)$ egész együtthatós polinom.
- **Kérdés:** Létezik-e egész értékű zérushelye?

Tétel (Matijasevič)

Hilbert 10. problémája algoritmikusan megoldhatatlan.

Post megfelelezési problémája

- **Adott:** $u_1, v_1, \dots, u_n, v_n \in \Sigma^*$.

További eldönthetetlen problémák

Hilbert 10. problémája

- **Adott:** $p(x_1, \dots, x_n)$ egész együtthatós polinom.
- **Kérdés:** Létezik-e egész értékű zérushelye?

Tétel (Matijasevič)

Hilbert 10. problémája algoritmikusan megoldhatatlan.

Post megfelelőzési problémája

- **Adott:** $u_1, v_1, \dots, u_n, v_n \in \Sigma^*$.
- **Kérdés:** Létezik-e olyan $i_1, \dots, i_k$ ($k > 0$, $1 \leq i_j \leq n$) sorozat, hogy

$$u_{i_1} \dots u_{i_k} = v_{i_1} \dots v_{i_k} ?$$

További eldönthetetlen problémák

Hilbert 10. problémája

- **Adott:** $p(x_1, \dots, x_n)$ egész együtthatós polinom.
- **Kérdés:** Létezik-e egész értékű zérushelye?

Tétel (Matijasevič)

Hilbert 10. problémája algoritmikusan megoldhatatlan.

Post megfelelőkezési problémája

- **Adott:** $u_1, v_1, \dots, u_n, v_n \in \Sigma^*$.
- **Kérdés:** Létezik-e olyan $i_1, \dots, i_k$ ($k > 0$, $1 \leq i_j \leq n$) sorozat, hogy

$$u_{i_1} \dots u_{i_k} = v_{i_1} \dots v_{i_k} ?$$

Tétel (Post)

A fenti probléma algoritmikusan megoldhatatlan.

Példa

Ha pl. $\Sigma = \{0, 1\}$ és a szavak

$$u_1 = 01 \qquad v_1 = 0$$

$$u_2 = 110010 \qquad v_2 = 0$$

$$u_3 = 1 \qquad v_3 = 1111$$

$$u_4 = 11 \qquad v_4 = 01,$$

van-e megoldás?

Példa

Ha pl. $\Sigma = \{0, 1\}$ és a szavak

$$u_1 = 01 \qquad v_1 = 0$$

$$u_2 = 110010 \qquad v_2 = 0$$

$$u_3 = 1 \qquad v_3 = 1111$$

$$u_4 = 11 \qquad v_4 = 01,$$

van-e megoldás?

Van, pl. 1, 3, 2, 4, 4, 3:

$$u_1 u_3 u_2 u_4 u_4 u_3 = 01 \cdot 1 \cdot 110010 \cdot 11 \cdot 11 \cdot 1 = 01111001011111,$$

$$v_1 v_3 v_2 v_4 v_4 v_3 = 0 \cdot 1111 \cdot 0 \cdot 01 \cdot 01 \cdot 1111 = 01111001011111.$$

Egy dominó probléma

- **Adott:** Dominó típusok véges halmaza.

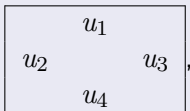
Egy dominó probléma

- **Adott:** Dominó típusok véges halmaza.
- **Kérdés:** Kirakható-e ezekkel a dominókkal az egész sík hézagmentesen?

Egy dominó probléma

- **Adott:** Dominó típusok véges halmaza.
- **Kérdés:** Kirakható-e ezekkel a dominókkal az egész sík hézagmentesen?

Típus:

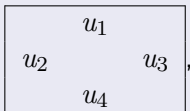


, $u_i \in \Sigma^*$.

Egy dominó probléma

- **Adott:** Dominó típusok véges halmaza.
- **Kérdés:** Kirakható-e ezekkel a dominókkal az egész sík hézagmentesen?

Típus:



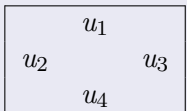
, $u_i \in \Sigma^*$.

A dominók nem forgathatóak.

Egy dominó probléma

- **Adott:** Dominó típusok véges halmaza.
- **Kérdés:** Kirakható-e ezekkel a dominókkal az egész sík hézagmentesen?

Típus:



$$u_i \in \Sigma^*.$$

A dominók nem forgathatóak.

Tétel

A dominó probléma algoritmikusan megoldhatatlan.

Egy megoldatlan probléma

- **Adott:** m pozitív egész szám.

Egy megoldatlan probléma

- **Adott:** m pozitív egész szám.
- **Kérdés:** Kongruens-e m ?

Egy megoldatlan probléma

- **Adott:** m pozitív egész szám.
- **Kérdés:** Kongruens-e m ?

Tehát azt kérdezzük, hogy létezik-e olyan derékszögű háromszög, mely oldalai racionális hosszúságúak és melynek területe m .

Egy megoldatlan probléma

- **Adott:** m pozitív egész szám.
- **Kérdés:** Kongruens-e m ?

Tehát azt kérdezzük, hogy létezik-e olyan derékszögű háromszög, mely oldalai racionális hosszúságúak és melynek területe m .

Példa

1, 2, 3, 4 **nem** kongruensek.

Egy megoldatlan probléma

- **Adott:** m pozitív egész szám.
- **Kérdés:** Kongruens-e m ?

Tehát azt kérdezzük, hogy létezik-e olyan derékszögű háromszög, mely oldalai racionális hosszúságúak és melynek területe m .

Példa

1, 2, 3, 4 **nem** kongruensek. 5 kongruens (Fibonacci):

$$a = \frac{3}{2}, \quad b = \frac{20}{3}, \quad c = \frac{41}{6}$$

Egy megoldatlan probléma

- **Adott:** m pozitív egész szám.
- **Kérdés:** Kongruens-e m ?

Tehát azt kérdezzük, hogy létezik-e olyan derékszögű háromszög, mely oldalai racionális hosszúságúak és melynek területe m .

Példa

1, 2, 3, 4 **nem** kongruensek. 5 kongruens (Fibonacci):

$$a = \frac{3}{2}, \quad b = \frac{20}{3}, \quad c = \frac{41}{6}$$

Nem ismert, hogy a probléma algoritmikusan eldönthető-e.

Állítás

Ha A rekurzív, akkor $\overline{A}$ is az.

Állítás

Ha A rekurzív, akkor $\overline{A}$ is az.

Ötlet

Az ACCEPT és a REJECT sorok cseréjével.

Állítás

Ha A rekurzív, akkor $\overline{A}$ is az.

Ötlet

Az ACCEPT és a REJECT sorok cseréjével.

Állítás

Egy A probléma pontosan akkor rekurzív, ha A és $\overline{A}$ rekurzívan felsorolhatók.

Állítás

Ha A rekurzív, akkor $\overline{A}$ is az.

Ötlet

Az ACCEPT és a REJECT sorok cseréjével.

Állítás

Egy A probléma pontosan akkor rekurzív, ha A és $\overline{A}$ rekurzívan felsorolhatók.

Ötlet

Állítás

Ha A rekurzív, akkor $\overline{A}$ is az.

Ötlet

Az ACCEPT és a REJECT sorok cseréjével.

Állítás

Egy A probléma pontosan akkor rekurzív, ha A és $\overline{A}$ rekurzívan felsorolhatók.

Ötlet

- A rekurzív $\Rightarrow A$ rekurzívan felsorolható

Állítás

Ha A rekurzív, akkor $\overline{A}$ is az.

Ötlet

Az ACCEPT és a REJECT sorok cseréjével.

Állítás

Egy A probléma pontosan akkor rekurzív, ha A és $\overline{A}$ rekurzívan felsorolhatók.

Ötlet

- A rekurzív $\Rightarrow A$ rekurzívan felsorolható
 A rekurzív $\Rightarrow \overline{A}$ rekurzív, így $\overline{A}$ rekurzívan felsorolható

Állítás

Ha A rekurzív, akkor $\overline{A}$ is az.

Ötlet

Az ACCEPT és a REJECT sorok cseréjével.

Állítás

Egy A probléma pontosan akkor rekurzív, ha A és $\overline{A}$ rekurzívan felsorolhatók.

Ötlet

- A rekurzív $\Rightarrow A$ rekurzívan felsorolható
 A rekurzív $\Rightarrow \overline{A}$ rekurzív, így $\overline{A}$ rekurzívan felsorolható
- Tegyük fel, hogy A és $\overline{A}$ rekurzívan felsorolhatók.

Állítás

Ha A rekurzív, akkor $\overline{A}$ is az.

Ötlet

Az ACCEPT és a REJECT sorok cseréjével.

Állítás

Egy A probléma pontosan akkor rekurzív, ha A és $\overline{A}$ rekurzívan felsorolhatók.

Ötlet

- A rekurzív $\Rightarrow A$ rekurzívan felsorolható
 A rekurzív $\Rightarrow \overline{A}$ rekurzív, így $\overline{A}$ rekurzívan felsorolható
- Tegyük fel, hogy A és $\overline{A}$ rekurzívan felsorolhatók. Ekkor A felismerhető egy M , $\overline{A}$ pedig egy $\overline{M}$ programmal.

Állítás

Ha A rekurzív, akkor $\overline{A}$ is az.

Ötlet

Az ACCEPT és a REJECT sorok cseréjével.

Állítás

Egy A probléma pontosan akkor rekurzív, ha A és $\overline{A}$ rekurzívan felsorolhatók.

Ötlet

- A rekurzív $\Rightarrow A$ rekurzívan felsorolható
 A rekurzív $\Rightarrow \overline{A}$ rekurzív, így $\overline{A}$ rekurzívan felsorolható
- Tegyük fel, hogy A és $\overline{A}$ rekurzívan felsorolhatók. Ekkor A felismerhető egy M , $\overline{A}$ pedig egy $\overline{M}$ programmal. Szimuláljuk M -et és $\overline{M}$ -et párhuzamosan!

Definíció

Legyen $\mathcal{C}$ problémák egy osztálya. Ekkor

$$\text{co}\mathcal{C} = \{\bar{L} : L \in \mathcal{C}\}.$$

Definíció

Legyen $\mathcal{C}$ problémák egy osztálya. Ekkor

$$\text{co}\mathcal{C} = \{\overline{L} : L \in \mathcal{C}\}.$$

Példák

Definíció

Legyen $\mathcal{C}$ problémák egy osztálya. Ekkor

$$\text{co}\mathcal{C} = \{\overline{L} : L \in \mathcal{C}\}.$$

Példák

- coR : azon problémák, melyek komplementere rekurzív

Definíció

Legyen $\mathcal{C}$ problémák egy osztálya. Ekkor

$$\text{co}\mathcal{C} = \{\overline{L} : L \in \mathcal{C}\}.$$

Példák

- **coR**: azon problémák, melyek komplementere rekurzív
- **coRE**: azon problémák, melyek komplementere rekurzívan felsorolható

Következmény

$$\mathbf{R} = \mathbf{coR}$$

$$\mathbf{RE} \neq \mathbf{coRE}$$

$$\mathbf{R} = \mathbf{RE} \cap \mathbf{coRE}$$

Láttunk rá bizonyítékot, hogy minden „realisztikus” számítási modell szimulálható RAM-gépen, lényegi időigény-romlás nélkül.

Láttunk rá bizonyítékot, hogy minden „realisztikus” számítási modell szimulálható RAM-gépen, lényegi időigény-romlás nélkül.

Most bevezetünk egy nemrealisztikus számítási modellt.

Nemdeterminizmus

Láttunk rá bizonyítékot, hogy minden „realisztikus” számítási modell szimulálható RAM-gépen, lényegi időigény-romlás nélkül.

Most bevezetünk egy nemrealisztikus számítási modellt.

Egy **nemdeterminisztikus** RAM-programban

Láttunk rá bizonyítékot, hogy minden „realisztikus” számítási modell szimulálható RAM-gépen, lényegi időigény-romlás nélkül.

Most bevezetünk egy nemrealisztikus számítási modellt.

Egy **nemdeterminisztikus** RAM-programban

- több különböző programsor is kaphatja **ugyanazt a sorszámot**

Láttunk rá bizonyítékot, hogy minden „realisztikus” számítási modell szimulálható RAM-gépen, lényegi időigény-romlás nélkül.

Most bevezetünk egy nemrealisztikus számítási modellt.

Egy **nemdeterminisztikus** RAM-programban

- több különböző programsor is kaphatja **ugyanazt a sorszámot**
- egy **lehetséges futás** minden lépésében a PC által kijelölt sorszámú lehetséges utasítások **egyike** hajtódik végre

Láttunk rá bizonyítékot, hogy minden „realisztikus” számítási modell szimulálható RAM-gépen, lényegi időigény-romlás nélkül.

Most bevezetünk egy nemrealisztikus számítási modellt.

Egy **nemdeterminisztikus** RAM-programban

- több különböző programsor is kaphatja **ugyanazt a sorszámot**
- egy **lehetséges futás** minden lépésében a PC által kijelölt sorszámú lehetséges utasítások **egyike** hajtódik végre
- (utána a PC eggyel nő vagy ugrás esetén a megfelelő értékre áll be)

Láttunk rá bizonyítékot, hogy minden „realisztikus” számítási modell szimulálható RAM-gépen, lényegi időigény-romlás nélkül.

Most bevezetünk egy nemrealisztikus számítási modellt.

Egy **nemdeterminisztikus** RAM-programban

- több különböző programsor is kaphatja **ugyanazt a sorszámot**
- egy **lehetséges futás** minden lépésében a PC által kijelölt sorszámú lehetséges utasítások **egyike** hajtódik végre
- (utána a PC eggyel nő vagy ugrás esetén a megfelelő értékre áll be)

A program **elfogadja** az inputot, ha **van olyan** lehetséges futás, mely ACCEPT utasítással terminál, egyébként **elutasítja** azt

Nemdeterminizmus

Láttunk rá bizonyítékot, hogy minden „realisztikus” számítási modell szimulálható RAM-gépen, lényegi időigény-romlás nélkül.

Most bevezetünk egy nemrealisztikus számítási modellt.

Egy **nemdeterminisztikus** RAM-programban

- több különböző programsor is kaphatja **ugyanazt a sorszámot**
- egy **lehetséges futás** minden lépésében a PC által kijelölt sorszámú lehetséges utasítások **egyike** hajtódik végre
- (utána a PC eggyel nő vagy ugrás esetén a megfelelő értékre áll be)

A program **elfogadja** az inputot, ha **van olyan** lehetséges futás, mely ACCEPT utasítással terminál, egyébként **elutasítja** azt

<https://www.youtube.com/watch?v=1ufECeWtN34>

Nemdeterminisztikus generálás

Egy bit

1. $a := 0$
1. $a := 1$
2. ...

Egy n -bites szám

function ND(n)

1. $r := 0$
2. if $n == 0$ **return** r
3. $r := r + r$
4. $r := r + 1$
4. $r := r + 0$
5. $n := n - 1$
6. goto 2

Példa: HAMILTON-ÚT

- **Input:** $\vec{G} = (V, E)$ (irányított) gráf. Feltehető, hogy $V = \{1, \dots, n\}$.
- **Output:** Van-e $\vec{G}$ -ben Hamilton-út (azaz minden csúcsot érintő út)?

```
for  $i := 1 \dots n$  do  
     $W[i] := \text{ND}(1 + \lfloor \log n \rfloor)$   
for  $i := 1 \dots n - 1$  do  
    if  $(W[i], W[i + 1]) \notin E$  then REJECT  
for  $i := 1 \dots n$  do  
    for  $j := 1 \dots n$  do  
        if  $W[j] == i$  then BREAK  
    if  $j > n$  then REJECT  
ACCEPT
```


Nemdeterminisztikus időigény

Egy nemdeterminisztikus program **időigénye** $f(n)$, ha bármely n méretű inputon **tetszőleges** lehetséges futás $f(n)$ lépésben véget ér.

Nemdeterminisztikus időigény

Egy nemdeterminisztikus program **időigénye** $f(n)$, ha bármely n méretű inputon **tetszőleges** lehetséges futás $f(n)$ lépésben véget ér.

HAMILTON-ÚT időigénye

Az előző fólia algoritmusának időigénye például **négyzetes**:

Nemdeterminisztikus időigény

Egy nemdeterminisztikus program **időigénye** $f(n)$, ha bármely n méretű inputon **tetszőleges** lehetséges futás $f(n)$ lépésben véget ér.

HAMILTON-ÚT időigénye

Az előző fólia algoritmusának időigénye például **négyzetes**:

- nemdeterminisztikusan generál n darab, egyenként $\approx \log n$ -bites számot, ez $n \log n$ lépés;

Nemdeterminisztikus időigény

Egy nemdeterminisztikus program **időigénye** $f(n)$, ha bármely n méretű inputon **tetszőleges** lehetséges futás $f(n)$ lépésben véget ér.

HAMILTON-ÚT időigénye

Az előző fólia algoritmusának időigénye például **négyzetes**:

- nemdeterminisztikusan generál n darab, egyenként $\approx \log n$ -bites számot, ez $n \log n$ lépés;
- (determinisztikusan) ellenőrzi, hogy az n szám ebben a sorrendben valóban egy séta-e a gráfban, ez n lépés;

Nemdeterminisztikus időigény

Egy nemdeterminisztikus program **időigénye** $f(n)$, ha bármely n méretű inputon **tetszőleges** lehetséges futás $f(n)$ lépésben véget ér.

HAMILTON-ÚT időigénye

Az előző fólia algoritmusának időigénye például **négyzetes**:

- nemdeterminisztikusan generál n darab, egyenként $\approx \log n$ -bites számot, ez $n \log n$ lépés;
- (determinisztikusan) ellenőrzi, hogy az n szám ebben a sorrendben valóban egy séta-e a gráfban, ez n lépés;
- (determinisztikusan) ellenőrzi, hogy minden csúcs előfordul-e a sétában, ez n^2 lépés.

Nemdeterminisztikus időigény

Egy nemdeterminisztikus program **időigénye** $f(n)$, ha bármely n méretű inputon **tetszőleges** lehetséges futás $f(n)$ lépésben véget ér.

HAMILTON-ÚT időigénye

Az előző fólia algoritmusának időigénye például **négyzetes**:

- nemdeterminisztikusan generál n darab, egyenként $\approx \log n$ -bites számot, ez $n \log n$ lépés;
- (determinisztikusan) ellenőrzi, hogy az n szám ebben a sorrendben valóban egy séta-e a gráfban, ez n lépés;
- (determinisztikusan) ellenőrzi, hogy minden csúcs előfordul-e a sétában, ez n^2 lépés.

Megjegyzés

Az utolsó ellenőrzés hatékonyabban is elvégezhető, most a pszeudokód egyszerűségét tartottuk szem előtt.

- **Input:** konjunktív normálformájú formula

- **Input:** konjunktív normálformájú formula
- **Output:** kielégíthető-e?

- **Input:** konjunktív normálformájú formula
- **Output:** kielégíthető-e?

Algoritmus

- **Input:** konjunktív normálformájú formula
- **Output:** kielégíthető-e?

Algoritmus

- minden változónak nemdeterminisztikusan beállítjuk az értékét 1-re vagy 0-ra (lineáris időigény);

- **Input:** konjunktív normálformájú formula
- **Output:** kielégíthető-e?

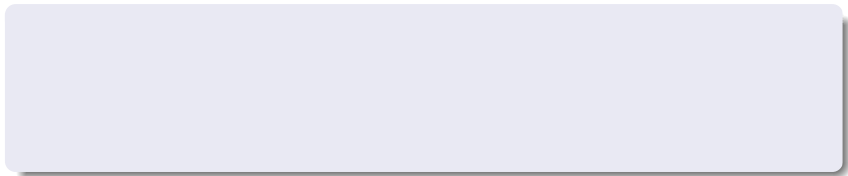
Algoritmus

- minden változónak nemdeterminisztikusan beállítjuk az értékét 1-re vagy 0-ra (lineáris időigény);
- determinisztikusan ellenőrizzük, hogy a kapott kiértékelés kielégítő-e (lineáris időigény);

- **Input:** konjunktív normálformájú formula
- **Output:** kielégíthető-e?

Algoritmus

- minden változónak nemdeterminisztikusan beállítjuk az értékét 1-re vagy 0-ra (lineáris időigény);
- determinisztikusan ellenőrizzük, hogy a kapott kiértékelés kielégítő-e (lineáris időigény);
- ha igen, ACCEPT, egyébként REJECT választ adunk.



- **Input:** $G = (V, E)$ (irányítatlan) gráf.

- **Input:** $G = (V, E)$ (irányítatlan) gráf.
- **Output:** kiszínezhető-e G **helyesen**, vagyis adhatunk-e minden csúcsnak egy-egy színt egy háromelemű színhalmazból, hogy szomszédos csúcsok különböző színt kapjanak?

- **Input:** $G = (V, E)$ (irányítatlan) gráf.
- **Output:** kiszínezhető-e G **helyesen**, vagyis adhatunk-e minden csúcsnak egy-egy színt egy háromelemű színhalmazból, hogy szomszédos csúcsok különböző színt kapjanak?

Algoritmus

- **Input:** $G = (V, E)$ (irányítatlan) gráf.
- **Output:** kiszínezhető-e G **helyesen**, vagyis adhatunk-e minden csúcsnak egy-egy színt egy háromelemű színhalmazból, hogy szomszédos csúcsok különböző színt kapjanak?

Algoritmus

- minden csúcsnak nemdeterminisztikusan adunk egy színt az $\{R, G, B\}$ halmazból;

- **Input:** $G = (V, E)$ (irányítatlan) gráf.
- **Output:** kiszínezhető-e G **helyesen**, vagyis adhatunk-e minden csúcsnak egy-egy színt egy háromelemű színhalmazból, hogy szomszédos csúcsok különböző színt kapjanak?

Algoritmus

- minden csúcsnak nemdeterminisztikusan adunk egy színt az $\{R, G, B\}$ halmazból;
- determinisztikusan ellenőrizzük, hogy a kapott színezés helyes-e;

- **Input:** $G = (V, E)$ (irányítatlan) gráf.
- **Output:** kiszínezhető-e G **helyesen**, vagyis adhatunk-e minden csúcsnak egy-egy színt egy háromelemű színhalmazból, hogy szomszédos csúcsok különböző színt kapjanak?

Algoritmus

- minden csúcsnak nemdeterminisztikusan adunk egy színt az $\{R, G, B\}$ halmazból;
- determinisztikusan ellenőrizzük, hogy a kapott színezés helyes-e;
- ha igen, ACCEPT, egyébként REJECT választ adunk.

Az **NP** osztályba mindazok a problémák tartoznak, melyek eldönthetők **polinom időigényű nemdeterminisztikus programmal**.

Az **NP** osztályba mindazok a problémák tartoznak, melyek eldönthetők **polinom időigényű nemdeterminisztikus programmal**.

Azaz, $L \in \mathbf{NP}$, ha van olyan M polinom időkorlátos nemdeterminisztikus program, melyre

Az **NP** osztályba mindazok a problémák tartoznak, melyek eldönthetők **polinom időigényű nemdeterminisztikus programmal**.

Azaz, $L \in \mathbf{NP}$, ha van olyan M polinom időkorlátos nemdeterminisztikus program, melyre

- ha $x \in L$, akkor M -nek **létezik** elfogadó futása x -en, és

Az **NP** osztályba mindazok a problémák tartoznak, melyek eldönthetők **polinom időigényű nemdeterminisztikus programmal**.

Azaz, $L \in \mathbf{NP}$, ha van olyan M polinom időkorlátos nemdeterminisztikus program, melyre

- ha $x \in L$, akkor M -nek **létezik** elfogadó futása x -en, és
- ha $x \notin L$, akkor M -nek minden futása elutasítja x -et.

Az **NP** osztály

Az **NP** osztályba mindazok a problémák tartoznak, melyek eldönthetők **polinom időigényű nemdeterminisztikus programmal**.

Azaz, $L \in \mathbf{NP}$, ha van olyan M polinom időkorlátos nemdeterminisztikus program, melyre

- ha $x \in L$, akkor M -nek **létezik** elfogadó futása x -en, és
- ha $x \notin L$, akkor M -nek minden futása elutasítja x -et.

Példa

HAMILTON-ÚT, SAT és 3 – SZÍNEZÉS **NP**-beli problémák.

Az **NP** osztály

Az **NP** osztályba mindazok a problémák tartoznak, melyek eldönthetők **polinom időigényű nemdeterminisztikus programmal**.

Azaz, $L \in \mathbf{NP}$, ha van olyan M polinom időkorlátos nemdeterminisztikus program, melyre

- ha $x \in L$, akkor M -nek **létezik** elfogadó futása x -en, és
- ha $x \notin L$, akkor M -nek minden futása elutasítja x -et.

Példa

HAMILTON-ÚT, SAT és 3 – SZÍNEZÉS **NP**-beli problémák.

Mivel minden $f(n)$ időigényű program felfogható $f(n)$ időigényű nemdeterminisztikus programként is, így nyilván $\mathbf{P} \subseteq \mathbf{NP}$.

Összefoglalás

- Megismertük az univerzális programot.
- Megismertük a megállási problémát és láttuk, hogy eldönthetetlen.
- Megismertük a MINDENEN MEGÁLLÁS problémát és visszavezetéssel láttuk, hogy szintén eldönthetetlen.
- Megismertük az ÜRES INPUTON MEGÁLLÁS problémát és visszavezetéssel láttuk, hogy szintén eldönthetetlen.
- Megismertük, mit jelent, hogy egy probléma rekurzívan felsorolható (avagy Turing-felismerhető, félig eldönthető).
- **RE**: a rekurzívan felsorolható problémák osztálya.
- Láttuk, hogy a megállási probléma rekurzívan felsorolható, és ezért $R \subsetneq RE$.
- Rice tétele: a rekurzívan felsorolható problémák egyetlen nemtriviális tulajdonsága se dönthető el.
- Megismertük Hilbert tizedik problémáját, Post megfeleltetési problémáját, melyek szintén csak félig eldönthetők.
- coC : a C -beli problémák komplementereinek osztálya.
- Láttuk, hogy $R = RE \cap coRE$, $R = coR$ és $RE \neq coRE$.
- Megismertük a nemdeterminisztikus algoritmusokat és azok időigényét.
- **NP**: a nemdeterminisztikusan polinomidőben eldönthető problémák osztálya.
- Láttuk, hogy HAMILTON-ÚT, SAT és 3 – SZÍNEZÉS **NP**-beliek.
- Láttuk, hogy $P \subseteq NP$.