# NP-teljesség röviden

Bucsay Balázs

earthquake[at]rycon[dot]hu

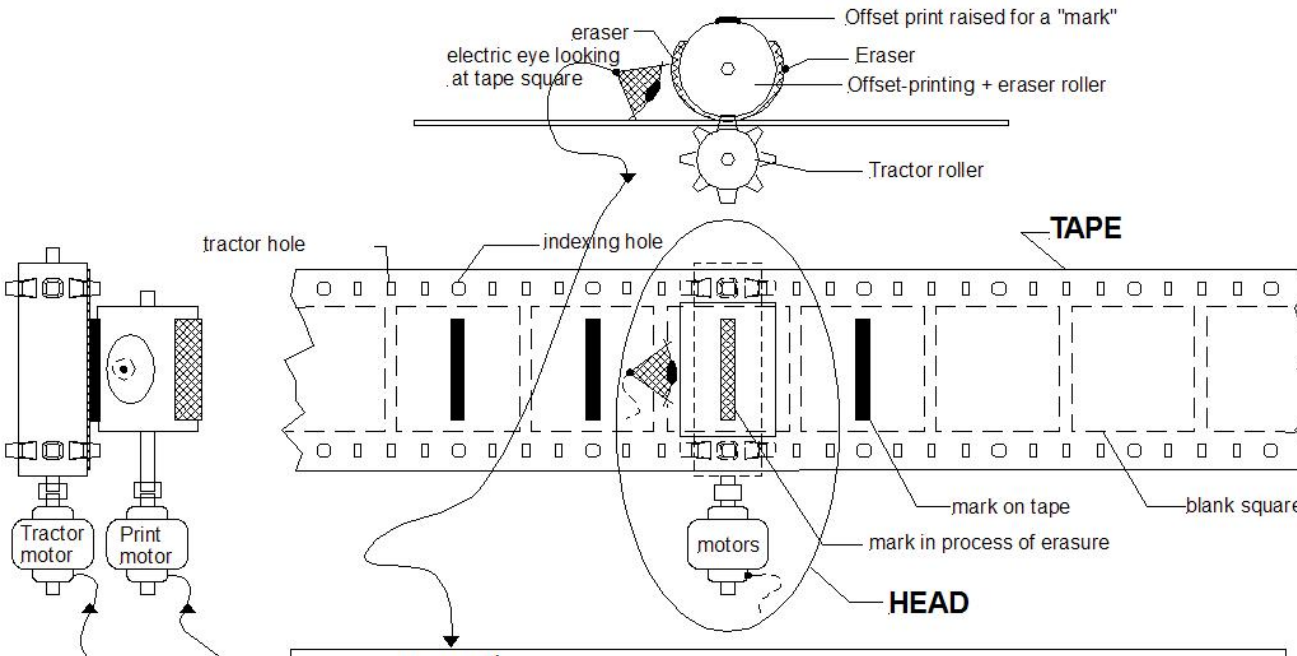
http://rycon.hu

# Turing gépek 1/3

## Mi a turing gép?

- **1. Definíció.** *[Turing gép] Egy Turing-gép formálisan egy* $M = (K, \Sigma, \delta, s)$ *rendezett négyessel adható meg, ahol* $K$ *állapotok egy véges halmaza (ezek az imént említett utasítások),* $s \in K$ *a kezdőállapot,* $\Sigma$ *pedig betűk egy véges halmaza (azt mondjuk, hogy* $\Sigma$ *az* $M$ *gép ábécéje). Mindig feltesszük, hogy* $K$ *és* $\Sigma$ *diszjunktak, és hogy* $\Sigma$ *tartalmazza a* $\sqcup$ *(üres) és a* $\triangleright$ *(kezdet) szimbólumokat. Végezetül,* $\delta$ *az átmenetfüggvény, mely a* $K \times \Sigma$ *halmazból a* $(K \cup \{h, \text{"}igen\text{"}, \text{"}nem\text{"}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$ *halmazba képez. Mindig feltesszük, hogy a* $h$ *(megállási állapot), "igen" (elfogadó állapot) és "nem" (elutasító állapot), tovább a* $\leftarrow$ *("lépés balra"),* $\rightarrow$ *("lépés jobbra") és* $-$ *("helyben marad") mozgási irányok nem elemei a* $K \cup \Sigma$ *halmaznak.*

# Turing gépek 2/3




**TABLE**

| | Current state A: | | | Current state B: | | | Current state C: | | |
|---|---|---|---|---|---|---|---|---|---|
| | Write symbol: | Move tape: | Next state: | Write symbol: | Move tape: | Next state: | Write symbol: | Move tape: | Next state: |
| tape symbol is 0: | 1 | R | **B** | 1 | L | **A** | 1 | L | **B** |
| tape symbol is 1: | 1 | L | **C** | 1 | R | **B** | 1 | N | HALT |

A fanciful mechanical Turing machine's TAPE and HEAD. The TABLE instructions might be on another "read only" tape, or perhaps on punch-cards. Usually a "finite state machine" is the model for the TABLE.

Turing gép ábra

# Turing gépek 3/3

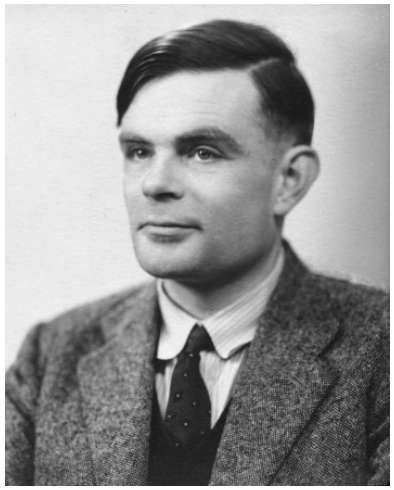
## Gép működése:

- 1. Olvasás a szalagról
- 2. Szimbólum szerint szabály alkalmazása vagyis:
  - 2a. Új szimbólum visszírása, vagy eredeti meghagyása
  - 2b. fej mozgatása jobbra vagy balra egyet.
- 3. Állapotváltás vagy STOP szimbólum
- 4. Ugrás az 1. pontra, ha nem volt STOP szimbólum

# Alan Mathison Turing

## Wikipedia:

- `Alan Mathison Turing` (1912. június 23. – 1954. június 7.) brit matematikus, a modern számítógéptudomány egyik atyja. Nagy hatással volt az algoritmus és a számítógépes adatfeldolgozás hivatalos koncepciójának kidolgozására az általa feltalált Turing-gép. Szabályba foglalta a ma már széles körben elfogadott Church–Turing-tézist, ami alapján tudniillik minden más számítási modell és más gyakorlati számítási modell azonos a Turing-géppel vagy annak egy részegységével. A II. világháború alatt sikeres erőfeszítéseket tett a német rejtjelkódok feltörésére. A háború után az egyik legkorábbi digitális számítógépen dolgozott, és később közreadott egy provokatív írást, a gondolatébresztő „Tudnak a gépek gondolkodni?” címűt.



Alan Mathison Turing

# RAM modell

Random Access Machine (véletlen elérésű gép)

- Rendelkezik memóriával
- Memóriában bárhova bármikor írhat illetve olvasható
- Így tárolhat a memóriában eredményeket
- A memóriában tárolt eredményeket felhasználhatja újabb eredményekhez

Ez a gép áll a legközelebb a Neumann elvekhez illetve a manapság használt PC-khez

Bevezethető egy ekvivalencia reláció, amivel ezt a modellt ekvivalenssé tehetjük más modellekkel, így a Turing géppel is, vagy a PC-vel. Ez azt jelenti, hogy polinom időveszteséggel, de az algoritmusok lefutnak egyik vagy másik gépen is, vagyis lényegi különbség nem igazán van köztük, vagyis bármely matematikai problémára amelyre megkonstruálható algoritmus, az leprogramozható Turing gépre is akár, ami az egyik legegyszerűbb modell.

# Nemdeterminisztikus Turing gép

Lényegében megegyezik az eredeti (determinisztikus) Turing gép definíciójával

Különbségek:

- Egy szimbólumhoz tartozhat 0, 1 vagy több szabály
- A szabályok közül a gép bármelyiket választhatja ha több van
- A választás előre nem megmondható, a gép az aktuális pozícióban tud csak dönteni
- Amit egy determinisztikus Turing gép elfogad (STOP állapotba kerül) azt ő is.

# Bonyolultsági osztályok

## Jelölésmód:

- $TIME(f(n))$ -nel jelöljük azon algoritmusok osztályát, amelyek legfeljebb $f(n)$ időbonyolultságúak
- $NTIME(f(n))$ -nel jelöljük azon algoritmusok osztályát, amelyek legfeljebb $f(n)$ időbonyolultságúak egy nemdeterminisztikus Turing gépen
- $SPACE(f(n))$ -nel jelöljük azon algoritmusok osztályát, amelyek legfeljebb $f(n)$ tárbonyolultságúak
- $NSPACE(f(n))$ -nel jelöljük azon algoritmusok osztályát, amelyek legfeljebb $f(n)$ tárbonyolultságúak egy nemdeterminisztikus Turing gépen

## Példa:

- Polinom időbonyolulságú algoritmusok osztálya: $P = TIME(n^k)$, ahol $\exists k \in \mathbb{N}, \forall n \in \mathbb{N} - re$
- Nem determinisztikus Turing gépen polinom időben lefutó algoritmusok osztálya: $NP = NTIME(n^k)$, ahol $\exists k \in \mathbb{N}, \forall n \in \mathbb{N} - re$ (P időben ellenőrizhetők)
- NP-teljes bonyolultsági osztály azon problémák osztálya, amely problémák legalább olyan nehezek mint bármelyik NP-beli probléma

# Döntési és optimalizálási problémák

Döntési problémáknál a felvetett kérdésre két válasz létezhet, az igen vagy a nem. Még az optimalizálási problémáknál a válasz egy érték vagy egy teljes struktúra is lehet.

Példák:

- LEGRÖVIDEBB-ÚT: optimalizálási, megadja a legrövidebb utat a gráfban
- LEGRÖVIDEBB-ÚT(E): döntési, egy útról megadja, hogy a legrövidebb út-e
- HAMILTON-KÖR: optimalizálási, megad egy Hamilton kört a gráfban
- HAMILTON-KÖR(E): döntési, egy részgráfról megmondja, hogy Hamilton kör-e a gráfban

Minden optimalizálási problémához létezik döntési probléma

A döntési problémák gyengébbek mint az optimalizálási problémák, viszont erős kapcsolat van köztük

Az NP-teljesség elmélete csak a döntési problémákkal foglalkozik.

# Visszavezetések 1/2

## Rövid deiníciók:

- Eset: a probléma egy bemenete
- $A$ és $B$ két különböző probléma
- $\alpha$ az $A$ probléma esete, $\beta$ a $B$ egy esete
- $f(x)$ egy visszavezető algoritmus

## $A$ probléma visszavezethető $B$-re ha:

- $A$ bármelyik $\alpha$ esetét $f(\alpha)$-val $B$ egy $\beta$ esetére vezethetjük vissza polinomiális időben
- Ha $A\alpha$ esete $f(\alpha)B$ problémában "igaz" a döntési probléma megoldása, akkor $A$ problémában az $\alpha$ eset is "igaz" lesz.
- Ezzel $A$ problémát visszavezettük a $B$ problémára.
- Ha minden esetre igaz ez, akkor $A$ ugyanolyan könnyű mint $B$

# Visszavezetések 2/2

Ha $B$ probléma nehézséget akarjuk bizonyítani akkor más a dolgunk:

Szükségünk van egy nehéz problémára, amihez viszonyíthatunk

Ez a probléma lesz a SAT probléma, ami NP-teljes

Fogjuk a SAT (vagy vele ekvivalens) problémát és a $B$ probléma esetere visszavezetjük

Indirekten bizonyítjuk, hogy létezik polinomiális algoritmus $B$-re

Ezzel beláttuk, hogy $B$ is legalább olyan nehéz mint a rá visszavezetett NP-teljes probléma

# $P \stackrel{?}{\neq} NP$ 1/2

NP-beli problémák minden esetére létezik P-beli tanú, vagyis P időben tudjuk ellenőrizni a probléma esetének helyességét

A visszavezetési függvények felhasználhatók ekvivalencia relációknak, melyek osztályai a bonyolultsági osztályok természetesen.

Ezzel az ekvivalencia relációval megmutathatjuk, hogy minden NP-teljes probléma nehézsége ekvivalens polinom idő erejéig

Ebből következik, hogy ha egy NP-teljes problémára találunk P-beli algoritmust, akkor az összesre találtunk

A hierarchia összeomlik és $P = NP$ lesz.

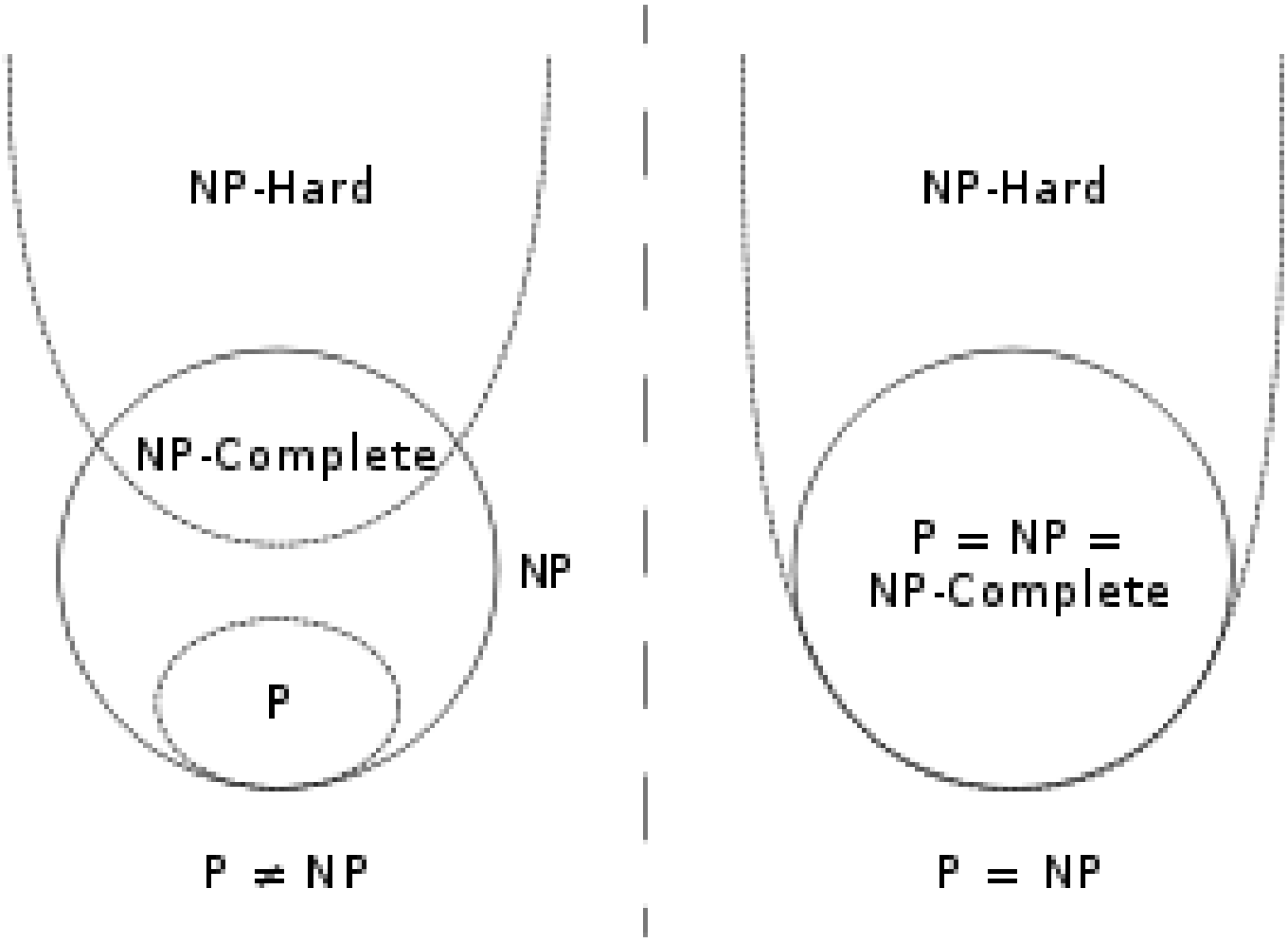
Ilyen megoldást nem sikerült a utóbbi évtizedekben találni, úgy hiszik, hogy $P \neq NP$

**2. Definíció.** *[Polinomális visszavezető függvény] Azt mondjuk, hogy* $L_1$ *probléma visszavezethető* $L_2$ *problémára (jelölése:* $L_1 \leq_p L_2$*), ha létezik olyan* $f$ *polinom időben kiszámítható függvény, melyre minden* $x \in L_1$ *esetén* $f(x) \in L_2$

**3. Definíció.** *[NP-teljes] Egy* $L$ *probléma NP-teljes, ha:* $L \in NP$ *és* $\forall L' \in NP$*-re teljesül hogy:* $L' \leq_p L$

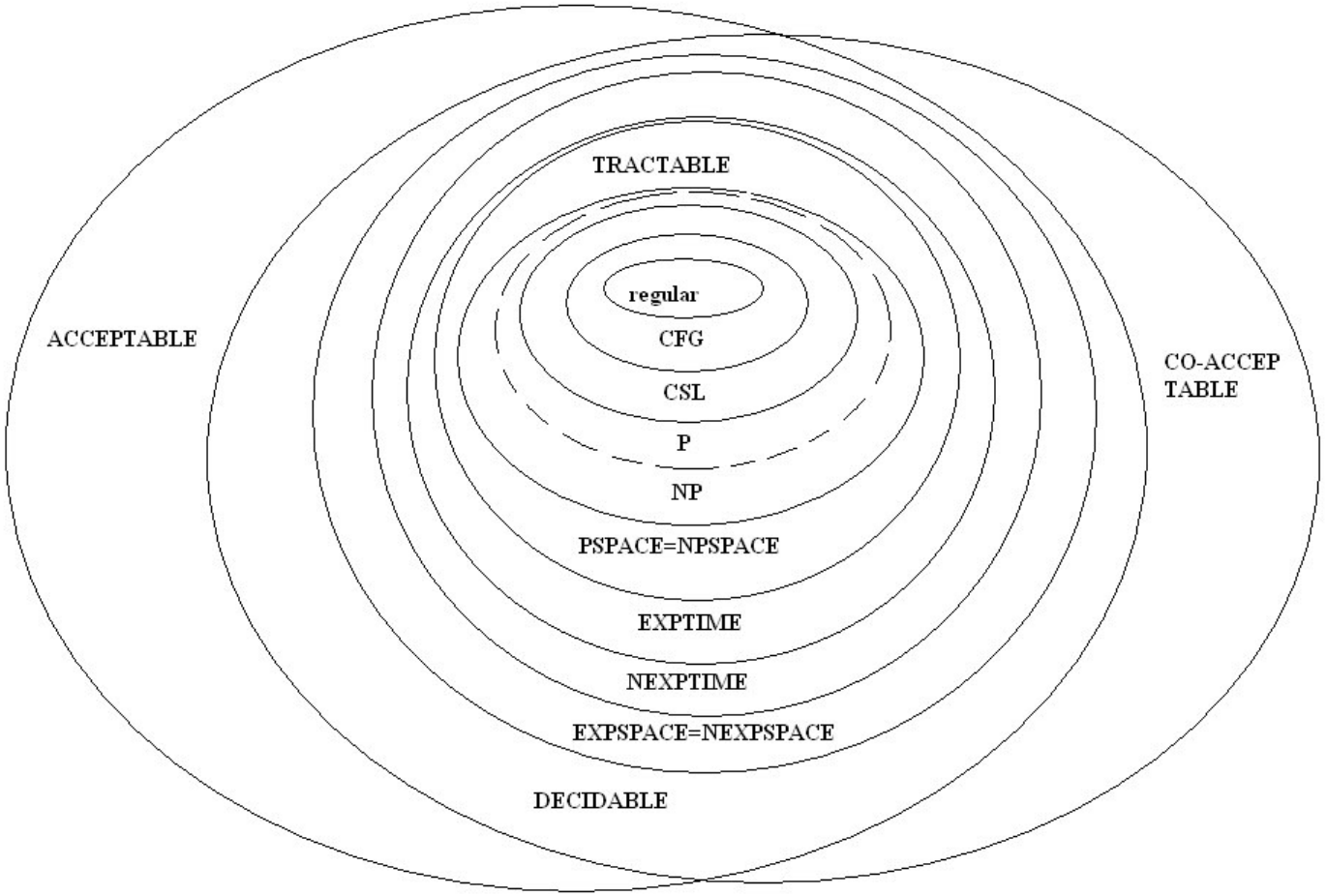
***1. Tétel.*** *[P=NP] Ha létezik polinomiális időben megoldható NP-teljes probléma, akkor P=NP. Más szóval: ha létezik NP-ben polinom időben nem megoldható probléma, akkor egyetlen NP-teljes probléma sem polinomiális.*

# Hierarchia




Egyszerűsített, lehetséges hierarchiák

# Chomsky hierarchia




Chomsky hierarchia

# Vége

Köszönöm a figyelmet.