Algoritmikus vagy véletlen?
I. rész

Talán mindenki elgondolkodott már azon, mit is jelent a véletlen fogalma. Például dobjunk fel egy pénzérmét 16-szor, és írjuk fel a kapott fejek és írások sorozatát. Tételezzük fel, hogy 4 kísérletben a következőket kapjuk:

(1) FFFFFFFFFFFFFFFF;
(2) FFFIFFIFFFFIFFFF;
(3) FIFIFIFIFIFIFIFI;
(4) FIFFFIFFIIFIFFII.

Az (1) esetben biztosan arra gyanakszunk, hogy az érme cinkelt, még (2) esetén is túl kevés az írások száma. Nem hisszük el a (3) eredményt sem, annak ellenére, hogy a dobásoknak pontosan a fele fej.

A (4) esettel szemben azonban valószínűleg semmiféle kételyt nem támasztanánk. Gondoljuk viszont meg, hogy mind a 4 eset egy-egy a lehetséges 2^{16} féle kimenetelből, valószínűségük tehát azonos.

Így a jelenség nem a valószínűségszámítás nyelvén tűnik magyarázhatónak, inkább az algoritmusokén. Arról van szó, hogy az (1)–(3) eseményeket túl szabályosnak, egyszerű algoritmussal leírhatónak tartjuk, szemben a szabálytalan, „véletlenszerű" (4) esettel. Például (1) az „írj le 16 fejet", (3) az „írj le 8 FI-t" utasítással megadható, de még (2) is, a három írás helyének megadásával, míg (4)-nél ilyen utasítás nincs.

A cikkben foglalkozni fogunk az algoritmikusság és a véletlenszerűség összehasonlításával, az utóbbit jellemezzük, mint az előbbi ellentétét, s eközben az algoritmusok világának néhány meglepő, paradoxonszerű tulajdonságát fogjuk megismerni, sőt Gödel híres tételét is be tudjuk majd bizonyítani.

Kezdés előtt egy figyelmeztetés. Mivel az algoritmusok, a logika területe csak igen fáradságosan és terjedelmesen tehető precízzé, gyakran fogunk intuíciónkra hagyatkozni. Talán nem lesz ez nagyon zavaró annak ellenére, hogy ki fog derülni, hogy olyan intuitívan egyszerű dolgok, mint pl. egy program végtelen ciklusba esése, számítógéppel kezelhetetlen probléma.

A bolha

Lássunk bevezetőül egy elemi feladatot: egy bolhát kell a sötétben elkapnunk úgy, hogy minden másodpercben átugrik a bolha egy másik helyre (nem látjuk, hogy honnan hová), mi pedig megpróbáljuk eltalálni azt a helyet és rácsapni. A bolhát csak akkor láthatjuk meg, ha elkaptuk.

Célunk olyan stratégia kidolgozása, hogy véges idő alatt (amely a bolha szerencséjétől függően tetszőlegesen hosszú lehet) biztosan elkapjuk a bolhát, ha:

I. A bolha a természetes számokon ($\mathbb{N} = \{0, 1, 2 \ldots\}$) ugrál, a 0-ból indulva, ismeretlen $v \in \mathbb{N}$ „sebességgel", azaz másodpercenként v (egész) mezővel jobbra ugorva.

II. A bolha ismét $\mathbb{N}$-en ugrál, ismeretlen $v \in \mathbb{N}$ sebességgel, az ismeretlen $x_0 \in \mathbb{N}$ pontból indulva.

III. A bolha a sík természetes számokból álló koordinátájú pontjain ugrál, $v = (v_x, v_y)$ ismeretlen sebességgel, az ismeretlen (x_0, y_0) pontból indulva.

IV. A bolha egy „végtelen dimenziós téren" ugrál, azaz olyan téren, amelynek pontjait, vagyis a bolha pillanatnyi helyzetét egy természetes számokból álló végtelen sorozat adja meg. A bolha a $(0, 0, 0 \ldots)$ pontból induljon, és $(v_1, v_2, v_3, \ldots)$ sebességvektorának véges sok v_i kivételével minden komponense 0 legyen. (Másképpen: a bolha sebességvektora egy véges sorozat – ha a végéről a végtelen sok 0-t elhagyjuk.)

V. A bolha $\mathbb{N}$-en menekül, és bármely t időpontban helyét egy számítógép által végrehajtott algoritmus adja meg, a t értékének függvényében.

VI. A bolháról semmit nem tudunk, véletlenszerűen ugrál $\mathbb{N}$-en.

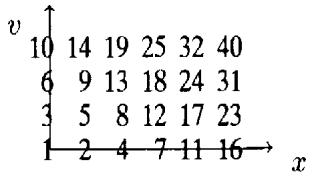
Feladat. *Továbbolvasás előtt adjunk stratégiát az I–IV esetekre.*

* * *

Ellenőrzésképpen, és mivel később is szükségünk lesz rá, következzenek a megoldások.

I. A t időpontban a bolha a $v \cdot t$ mezőn tartózkodik, tehát ha az 1. másodpercben a $v = 0$, a másodikban a $v = 1, \ldots,$ esetet próbáljuk ki, azaz a t-edik másodpercben a $t \cdot (t-1)$-edik mezőre csapunk, biztosan elkapjuk a bolhát.

II. Most a bolha $(x_0 + vt)$-n tartózkodik, célunk az összes (x_0, v) pár kipróbálása legalább egy-egy időpontban. Például az ábrán látható bejárást választhatjuk, $t = 1$ esetén a $v = 0$, $x_0 = 0$, $t = 2$ esetén $v = 0$, $x_0 = 1$, $t = 3$-ra $v = 1$, $x_0 = 0, \ldots$ stb. eseteket kipróbálva.



Fontos kiemelni a lényegét annak, amit csináltunk: kölcsönösen-egyértelmű leképzést adtunk meg $\mathbb{N}$ és $\mathbb{N}\times\mathbb{N}$ (azaz a számpárok) között, amelyet jelöljünk $\varphi:\mathbb{N}\times\mathbb{N}\to\mathbb{N}$-nel, inverzét $\Psi=(\Psi_1,\Psi_2):\mathbb{N}\to\mathbb{N}\times\mathbb{N}$-nel. Így a t-edik másodpercben p1. az $(x_0,v)=\Psi(t-1)$ kezdeti értékekhez tartozó pozíciót próbálhatjuk ki, azaz a $\Psi_1(t-1)+\Psi_2(t-1)\cdot t$ pozíciót.

III. Ezek után egyszerű a megoldás: 4 paraméterünket (v_x,v_y,x_0,y_0) kódolhatjuk a $\varphi\big(\varphi(x_0,y_0),\varphi(v_x,v_y)\big)$ képlettel és a bolhát az összes lehetséges kezdeti feltétel esetén elkapjuk. A t időponthoz az $x_0=\Psi_1\big(\Psi_1(t-1)\big)$, $v_x=\Psi_1\big(\Psi_2(t-1)\big)$, $y_0=\Psi_2\big(\Psi_1(t-1)\big)$, $v_y=\Psi_2\big(\Psi_2(t-1)\big)$ kezdeti értékek tartozzanak.

IV. Ismét megfoghatjuk a bolhát, ha fel tudjuk sorolni lehetséges sebességvektorait, a véges számsorozatokat, azaz $\mathbb{N}$-nel „kódolni" tudjuk azokat.

Tegyük ezt a következőképpen. Írjuk fel a számsorozatot vesszőkkel elválasztva, azaz 11 jel: a $0-9$ számjegyek és a vessző (,) felhasználásával. Tekintsük az így felírt jelsorozatban a vesszőt is számjegynek, így egy 11-es számrendszerbeli számot kapunk.

Megvan tehát a felsorolás, ha $\mathbb{N}$ számainak 11-es számrendszerbeli felírásából vissza tudjuk állítani a számsorozatot. Problémánk egyedül azzal lehet, hogy két vessző következhet egymás után egy 11-es számrendszerbeli számban. Ekkor tekintsük egyszerűen 0-nak a hiányzó számértéket.

Ez a leképzés már nem egy-egy értelmű (pl.: $0,0$-t állítja elő , $0_{\underset{\smile}{11}}=110$ és , $_{\underset{\smile}{11}}=10$ is, azonban igaz az, hogy minden időpontban kipróbálunk egy sebességvektort, és előbb-utóbb ki is próbáljuk az összeset, így a bolhát elkapjuk.

Az algoritmusok felsorolhatósága

Most térjünk rá az V–VI. esetekre. Van-e egyáltalán különbség a kettő között? Vajon nem adható-e meg minden „ugrálási sorozat" egy programmal?

Ehhez először is tisztáznunk kell, mi is lehet a számítógép, amelyet bolhánk használ, azaz egy számunkra kezelhető, egyszerű de elég általános modellt kell alkotnunk. A gép álljon egy memóriából, amely a bemeneti és számítás közbeni adatokat, illetve a végeredményt tárolja, és egy programtárból, amely a végrehajtandó utasításokat.

Legyen a memória korlátlanul nagy, azaz minden program használhasson tetszőlegesen sok memóriarekeszt, és minden memóriarekesz tárolhasson egy tetszőlegesen nagy természetes számot. A rekeszek legyenek sorszámozva.

A gép programja álljon véges sok, egyesével növekedve sorszámozott sorból, minden sor a következő 5 utasítás egyikét tartalmazhatja:

(1) NÖVEL_k - a k-adik memóriarekesz tartalmát eggyel növeli;
(2) CSÖKKENT_k - a k-adik memóriarekesz tartalmát eggyel csökkenti;
(3) $\text{KONSTANS}_k(n)$ - az n számot a k-adik memóriarekeszbe helyezi;
(4) $\text{UGRIK}_k(n_1,n_2)$ - ha a k-adik rekesz tartalma 0, az n_1-edik, különben az n_2-edik soron folytatja a végrehajtást;
(5) STOP - program vége

Ellenőrizhető, hogy ez az egyszerű gép is képes mindarra, amire „valódi" testvérei, amelyek persze „csak" korlátos memóriával rendelkeznek.

Feladat. *(Csak programozásban járatosabbak részére!)*

Írjunk programrészt a k-adik és az l-edik rekeszek összeadására. Az így kapott $\texttt{ÖSSZEAD}(k,l)$ programrész segítségével pedig szorozzuk össze a k-adik és az l-edik rekeszek tartalmát. (Elkészíthetők további, a személyi számítógépen ismert (pl. BASIC) utasítások is a bolha számítógépén.)

Ahhoz, hogy a bolha valóban alkalmazni tudja a gépet, már csak arra van szüksége, hogy programja a következő lépést időegység alatt elkészítse. Egy idealizációt (a végtelen memória feltételezését) már tettünk abból a célból, hogy minden program, méretétől függetlenül végrehajtható legyen, kössük ki még azt is, hogy időegység alatt szolgáltasson is eredményt. Ez az idealizáció sem tűnik „lehetetlenebbnek", mint a végtelen memória.

Közeledjünk az V. esethez a szokásos módon: próbáljuk meg felsorolni a bolha lehetséges stratégiáit.

1. tétel. *A programok beszámozhatóak, azaz létezik olyan leképezés, amely minden természetes számhoz egy programot rendel, és minden programot előállít.*

Bizonyítás. Láttuk, hogy van olyan leképezés, amely minden $\mathbb{N}$-beli számhoz egy véges sorozatot rendel. Elegendő tehát a véges sorozatokat továbbképezni programokká úgy, hogy minden program előálljon valamely véges sorozatból. Ekkor a két leképezés egymásutánja sorszámozza be a programokat.

Tekintsünk egy programot, és hagyjuk el a sorszámozást (nincs a sorszámra szükség, hiszen úgyis egyesével növekszik). Írjuk ezután az utasításait egy sorba, az utasításokat és paramétereket vesszővel elválasztva. Pl. a

0 `NÖVEL`$_1$

1 `UGRIK`$_2(0,2)$

2 `STOP`

programból a `NÖVEL ,1,UGRIK,2,0,2,STOP` sorozat keletkezik. Ez számsorozattá alakítható, ha az utasításokat kódoljuk, pl.

0 – `NÖVEL`

1 – `CSÖKKENT`

2 – `KONSTANS`

3 – `UGRIK`

4 – `STOP`

Így minden programnak megfelel egy számsorozat, a fentinek a $0, 1, 3, 2, 0, 2, 4$.

Nézzük meg, hogyan tudunk fordítva, minden sorozathoz egy programot rendelni.
Mindjárt az első számmal baj van: ha 4-nél nagyobb, nem felel meg neki utasítás. Ezen segíthetünk, ha minden 4-nél nagyobb vagy vele egyenlő számhoz a `STOP`-ot rendeljük, ha utasítás helyén áll. Ezután már van egy utasításunk. Tudjuk, hogy hány paramétere van (`STOP`-nak 0, `NÖVEL` , `CSÖKKENT`-nek 1, `KONSTANS`-nak 2, `UGRIK`-nak 3), így a sorozat következő megfelelő elemei lesznek e paraméterek, a következő sorozatelem pedig az új utasítás. Baj csak a végén lehet, ha egy utasítás paraméterei hiányoznak, mert előbb ért véget a sorozat. Ekkor nullákkal pótoljuk a hiányzó értékeket. Így minden sorozathoz hozzárendeltünk egy programot, és minden programot elő is állítottunk.□

Ez megadja a választ az V. esetre: a t-edik másodpercben a t-edik programot a t bemenettel kell végrehajtani, így a bolhát előbb-utóbb elkaphatjuk. Ez a gondolatmenet még egyszerűen számítógépesíthetőnek is tűnik, hiszen a k számból előállítható a k-adik program, és ennek utasításait tudjuk szimulálni egy másik gépen. Erre később visszatérünk.

Cantor átlós eljárása, a megállási függvény

Vizsgáljuk most a VI. esetet. Ha a bolha szerencsés, ill. kémei vannak, megteheti, hogy mindig amellé a mező mellé lép, amelyre csapnánk, ezzel magának örök életet biztosítva. Tehát nemleges lesz a válasz e kérdésre, ami azt is jelenti, hogy nem lehet minden számsorozatot számítógéppel előállítani.

Ha csak a $0-9$ helyekre ugrál a bolha, akkor is annyiféle végtelen sorozatot tud előállítani, mint ahány valós szám van $[0,1]$-ben: írjuk egyszerűen tizedespont mögé sorra számjegyekként a bolha által elfoglalt mezők számát. Biztosan sokak számára ismert, hogy $[0,1]$ intervallum valós számainak számossága nagyobb, mint $\mathbb{N}$-é, nincsen olyan $\mathbb{N} \to [0,1]$ függvény, amely az egész $[0,1]$-et felveszi.

Tulajdonképpen ezt láttuk be az előbb, ám a módszert, Cantor átlós eljárását, célszerű még egyszer meggondolni. Tételezzük fel, hogy $[0,1]$ valós számai felsorolhatók, például:

$$0, a_{11}a_{12}a_{13}\dots$$
$$0, a_{21}a_{22}a_{23}\dots$$
$$\vdots$$

A bolhaüldözéssel analóg módon az a_{tt} átlót vizsgáljuk. A $0,(a_{11}+1)(a_{22}+1)\dots$ szám (ahol most $9+1$ definíció szerint 0) nem szerepelhet a felsorolásban, hiszen ha abban a t-edik lenne, t-edik jegyének $a_{tt} \neq a_{tt}+1$-nek kellene lennie.

Térjünk most vissza az V. esetre. Mi van, ha a bolha ekkor is kémkedik? Ha esetleg mi is számítógépet veszünk igénybe, előre megtudhatja annak programját, és ezt csak egy utasítással kell kiegészítenie a meneküléséhez: eggyel növelnie kell a végeredményt, hasonlóan az előző átlós eljáráshoz.

Az előbb viszont meg tudtuk adni a stratégiát a bolha elfogására. Hol lehet a hiba? A válasz csak az lehet, hogy üldöző módszerünk programmal nem kiszámítható (tudjuk már, hogy vannak ilyen sorozatok). A k-adik program azonban szimulálható a k szám ismeretében (aki nem hiszi, járjon utána: készítsen valamely gépen programot e feladatra). A másik lehetőség, hogy esetleg nem tudunk időben elkészülni következő lépésünkkel. Mi van, ha pl. a

0 `UGRIK`$_0(0,0)$

programot tekintjük éppen? Ez soha nem szolgáltathat eredményt, hiszen végtelen ciklusba esik, elrontja tehát a mi algoritmusunkat is.

Ez jó magyarázatnak tűnik, ám mi lenne, ha kicsit ügyeskednénk és csak azokat a programokat hajtanánk végre, amelyek nem esnek végtelen ciklusba. Legyen pl.

$$H(k,n) = \begin{cases} 0, & \text{ha a } k\text{-adik program az } n \text{ bemenetre végeredménnyel megáll,} \\ 1, & \text{ha végtelen ciklusba esik.} \end{cases}$$

Először tehát $H(k,n)$-t kell kiszámítanunk, pl. kiszűrni a fentihez hasonló sorokat - (aki nem hiszi írjon rá algoritmust). Ha ez 0, végrehajtjuk a programot, ha nem, egy időegységnyit pihenhetünk (vagy pl. a 0-ra csapunk).

Mivel már tudjuk, hogy ez lehetetlen, a következő konklúzióra kell jutnunk:

2. Tétel. *$H(k,n)$ számítógéppel nem kiszámítható.*

Bizonyítás. Tulajdonképpen már elhangzott, de emeljük ki még egyszer a lényeget. Legyen $f(k,n)$ a k-adik program kimenete az n bemenet esetén, ha létezik, különben legyen pl. 0. (A bolhaüldözés során tehát $f(k,k)$-t kell meghatároznunk.) Ha $H(k,k)$ kiszámítható volna, akkor $f(k,k)$ is, hiszen először $H(k,k)$-t meghatározva a k-adik programot csak akkor kell végrehajtani, ha valóban eredményt is szolgáltat.

Legyen $g(k) = f(k,k) + 1$. Indirekt feltevésünk szerint ez is egy program, pl. az l-edik.
Így $f(l,l) = g(l) = f(l,l) + 1$, ellentmondás. □

A számítógépek védelmében

Mielőtt azt gondolnánk, a számítástechnikusoknak bealkonyult, sosem lesz mesterséges intelligencia, hiszen íme egy feladat: „könnyedén" megtaláljuk egy programban a végtelen ciklusokat, ám a gép mégsem boldogul vele; mi lenne, ha a program a következő volna: soroljuk fel az összes számnégyest (pl. a $\varphi\big(\varphi(x,y),\varphi(z,n)\big)$ segítségével), és addig maradjunk ciklusban, amíg az $x^n + y^n = z^n$ összefüggés fenn nem áll köztük. A híres Fermat sejtés tehát csak speciális esete $H(k,n)$ meghatározásának!

Általában is érdekes vizsgálni azt az esetet, amikor a bolha menekülőprogramja diofantoszi egyenletek (egész együtthatós polinomok) egész gyökeit adja meg. Hilbert (1862-1943), korának nagy matematikusa, az 1900-as Nemzetközi Matematikus konferenciára általa fontosnak ítélt feladatok egy listáját készítette el. A 10. problémája éppen az volt, hogyan dönthető el egy diofantoszi egyenletről, hogy van-e gyöke. Sokak közreműködésével csak 1961-től kezdett el érni annak bizonyítása, hogy nincs általános módszer (azaz számítógépesíthető algoritmus) e feladatra.

Még egy érv a most kissé ellentmondásosnak tűnő algoritmuselmélet védelmében. Tulajdonképpen „megbukott" az az idealizáció, hogy minden program időegység alatt végrehajtható. A bolhának csak olyan programválasztása lehet, amely minden bemenetre 1 időegység alatt kimenetet szolgáltat. Ha nekünk egy kicsit jobb gépünk van, soronként szimulálhatjuk a lehetséges programokat, és ha annyi lépéssel, amennyinél többet a bolha nem tud időegység alatt végrehajtani, nem kapunk eredményt, kiszűrtük a programot a lehetséges menekülési stratégiák közül. A bolha viszont a mi programunkat nem hajthatja végre, hiszen az ő gépe lassabb a miénknél.