Spontán matematika-tudásnövelés programozók esetében, avagy a véletlen kezelésének kockázatai pszeudo-kódokban

(Knowledge-increasing on the field of mathematics in a spontaneous way in case of programmers – or risks in the pseudo-code-based representations of the coincidence)

Pitlik László, Pitlik Marcell (MY-X team)

Kivonat: A tantárgyi kontúrokat feladó, alkotás-orientált (pl. IT)-képzés kapcsán felmerül, hogy minden ún. alapozó tantárgy (vö. matematika) esetében csak akkor és annyit adjunk/várjunk el oktatóként, amennyi adott alkotás kapcsán szükséges. Ez nem feltétlenül kevés, mert vannak komplexitási szintek melyek több alapozást igényelnek, de a szintézis nem várat magára korlátlanul, hanem quasi az első adandó alkalommal fellép ez a motiváló hatás. A matematikai tanítása kapcsán a cikk rámutat két hatásmechanizmusra a valós oktatásból véve ehhez a példát, a Monty-Hall-paradoxont. Az egyik hatás a véletlenszám-generálás kockázatait mutatja be ott, ahol a programozó (pszeudo-kód-tervező) szeretné elkerülni téves számkombinációk szűrését. A másik hatás a matematikai bizonyítások, látásmód megkerülni akarásának esetleges lehetetlenségét mutatja be/ki – azzal kiegészítve, hogy a felismerés forráskód-szinten megtörténhet anélkül is, hogy a matematikai általánosítás tudatosan megszületne.

Kulcsszavak: just-in-time tudásmenedzsment, nyers erő, tudatosság, feltételes valószínűségek

Abstract: IT-education can be realized in a form where the classic subjects can not be identified because the education is creation-driven. The basics (like mathematics) will be offered in this specific form in micro packages – in form of the just-in-time-knowledge-management where the Students have to learn the optimized necessary dose. This dose should not necessarily be little – there are complexity levels where the dose is relatively robust. It is important, that the success (the synthesis) might not be wait too long – but the first potential appropriate moment should be used for this kind of motivation. Based on real educational examples, the paper presents two mechanisms concerning the education of mathematics for IT-experts. The first one can be described as a form of risk increasing in case of random numbers without allowing wrong scenarios. Hereby, the probabilities can be influenced in an unexpected way. The other one is the awareness of mathematical feelings. In this case it is possible, that an IT-expert is capable of solving a general mathematical challenge in frame of a mostly brute force problem handling – but this experts must not be capable of generalizing these solutions.

Keywords: just-in-time knowledge management, brute force, awareness, conditional probabilities

# Bevezetés

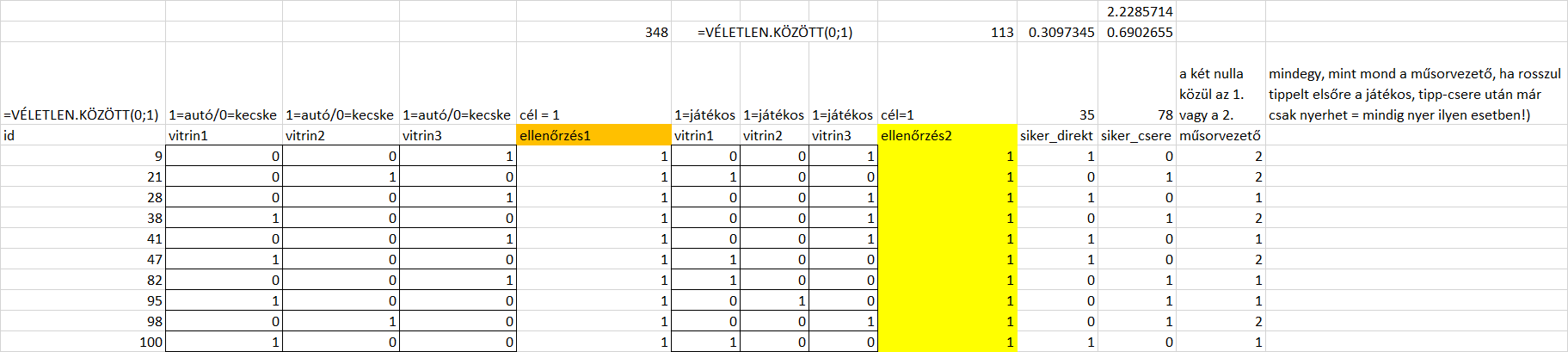
Előzmények:

* <https://miau.my-x.hu/bprof/kongruencia_demo.xlsx>
* <https://miau.my-x.hu/bprof/erobol.xlsx>
* <https://miau.my-x.hu/miau/192/naiv_ido.pdf>
* <http://miau.my-x.hu/miau/185/occams_razor_finetuned.doc>
* <https://miau.my-x.hu/bprof/monty-hall-paradoxon-tesztelese.xlsm>
* <https://miau.my-x.hu/bprof/TDK_kakukmarci_final.docx>

A tanítás célja nem lehet a raktárra tanulás kikényszerítése (1), vagyis a mindenkor tanított elemek minél hamarabb illik, hogy valós kihívások (2) kapcsán egymással kapcsolatba kerüljenek. Ha azt várjuk el az Occam-borotvája elvből kiindulva (vö. Kannibálok és Tudósok példája – occams\_razor\_finetuned.pdf), hogy egy adott feladat kezelése kapcsán a megoldások (mint objektumok) egyszerűségét a felhasznált matematikai apparátusok, ill. integrált robotok mennyiségei határozzák, mint attribútumok határozzák meg a minél egyszerűbb, annál jobb irányelvek keretei között, és a BSC/BPROF képzés esetében adott matematikai kihívásra a fenti Occam-optimalizáció eredményét jelentő megoldást KELL oktatni, akkor érdekes határterületi jelenségekbe futhatunk bele pl. a Monty-Hall-paradoxon erőből történő megoldásakor. Az egyik ilyen specialitás a véletlen reprezentációjának potenciális torzulási kockázata. A másik a matematikai kihívások spontán értelmezési képességének elkerülhetetlensége pszeudo-kódként értelmezhető, s egyben brute-force megoldást jelentő Excel-transzformációk esetén.

# A spontán matematika-tanulásról

A fentebb előzményként hivatkozott XLSM 33vs2 munkalapja az alábbi lépésekben garantálja az erőből (vö. brute-force-based) történő kezelését a Monty-Hall paradoxonnak (vö. <https://hu.wikipedia.org/wiki/Monty_Hall-paradoxon>). A paradoxon centrális kérdése, hogy egy látszólag véletlenszerű nyerési pozíció esetén (azaz három vitrin közül a megfelelő kiválasztásának megkísérlése után) vajon érdemes-e változtatni a kiindulási tippen, ha az egyik ki nem választott vitrinről kiderül, hogy ott biztosan nincs immár nyeremény?



1. Ábra: Helyesen reprezentált véletlen (forrás: saját számítások)

Az 1. ábra és a mögöttes XLSM-állomány több lépésben engedi eljutni a nyers erőt alkalmazó programozást előkészítő szakembert a bemutatott nézetig:

* Az első lépésben generálásra kerülnek (vö. VÉLETLENKÖZÖTT()-függvény) pl. 1000 sorban és 3 oszlopban (vitrin1-2-3) 0-s és 1-es érték-tripletek.
* Az első ellenőrzés (narancssárga-jel) mutatja, hogy pl. 348 esetben volt igaz, hogy a 3 oszlop közül csak egyben volt 1-es érték, amit a nyeremény helyeként kell értelmezni mostantól.
* A 348 elemű szűrleten újabb 3 oszlop definiálásával az előző lépéseket ismételve létre lehet hozni mindennemű matematikai tudás nélkül azokat (citromsárgával jelölt) sorokat, melyek esetén igaz, hogy itt is csak egy darab 1-es érték szerepel egy sorban.
* A 113 ilyen eset a 348-as véletlen kiindulási halmaznak éppúgy az 1/3-dát közelítő érték, mint ahogy ez a 348/1000 esetében megfigyelhető (1000 elemű mintából kiindulva) – s a végtelenben elvárva az egzakt 1/3-os arány kialakulását…
* A direkt, azaz a játékos eredeti választását nem változtató nyerési stratégia sikere a fenti példában 35/113, azaz ismét csak az 1/3-os arányhoz közeli. Sikerről akkor beszélhetünk, ha a játékos által választott vitrinben van a nyeremény (autó).
* S ezzel el is jutunk a kezdeti választás (vitrin) feladását jelentő stratégia értelmezéséhez. A műsorvezető által a két lehetséges még ki nem választott vitrin közül azt kell kiválasztani, ahol nincs autó, ill. valamelyiket, ha egyik opció sem jelent autót. Így az 1. ábra jobbszélső oszlopában szereplő 1-es érték a két, a játékos által ki nem választott vitrin közül az olvasás irány szerinti elsőt (bal oldalit) jelenti, míg a 2-es érték a másikat (a jobb oldalit).

A kérdés már csak az, milyen képlet kerüljön siker-csere oszlopba?

* A programozó természetesen a brute-force keretében előállíthatná még magát a végső mintázatot, ahol látható az alany első, ill. esetlegesen módosított választása, látható az immár leleplezett vitrin és persze a potenciális nyeremények. DE…
* Erre azonban nincs szükség, ha a programozó elsőként felismeri, hogy
  + Minden olyan esetben, amikor a siker-direkt oszlopban 1-es érték szerepelt, azaz változtatás nélküli esetben nyert volna a játékos (1/3 eséllyel), a siker-csere oszlopban kényszerűen 0-s értéknek kell állnia, lévén a játékos lerontotta saját pozícióját. S itt már nem számít, mely vitrint mutatja meg a műsorvezető – feltárva ennek tartalmát, azaz deklarálva a nyeremény hiányát.
  + A programozó (a programozást előkészítő pszeudo-kódgyártó) további feladata már „csak” az, hogy a fennmaradó (direkt esetben vesztes) esetekre vonatkozó HA/AKKOR szabályt is kialakítsa:
    - Ez a HA()-függvény(ág) azt kell, hogy kifejezze, mikor nyer és mikor veszít a játékos a műsorvezető döntésétől függően.
    - Azonban tudjuk, hogy a direkt nézetből szemlélt vesztes ágon mozogva a műsorvezető csak az egyetlen egy fennmaradó, nyereményt már nem tartalmazó vitrint választhatja, mert a nyertes vitrinre nem mutathat rá.
    - A játékosnak ezen az ágon pedig a saját, majd a műsorvezető döntése után nincs további választása – a fennmaradó (sem általa elsőre, sem a műsorvezető által ezt követően ki nem választott) harmadik vitrint KELL választania.
    - Mely vitrin azonban MINDENKOR ÉS KÉNYSZERŰEN a nyereményt KELL, hogy tartalmazza.

Ezzel a HA()-függvénytervezési lépéssorral a programozó megkerülhetetlenül levezette a Monty-Hall paradoxon matematikai megoldását, hiszen a siker-csere oszlopba írandó képlet már nem is képlet, hanem egy deklaráció: minden ilyen cella értéke legyen 1. Nem mellesleg ez kényszerűen nem más, mint, hogy a siker-csere cella értéke a siker-direkt-ág 0 értéke esetén legyen 1, azaz legyen ennek ellentettje.

Ha tehát a siker-direkt oszlop sikerarányáról belátható, hogy ennek végtelen szcenárió esetén az értéke 1/3-a az éppen addig vizsgált összes esetszámnak, akkor ennek ellentettje nem lehet más, mint 2/3. Így a játékosnak saját érdekében a műsorvezető információtöbblete nyomán kialakult ÚJ helyzetben változtatnia kell az eredeti választásán és a fennmaradó vitrint kell preferálnia. Ezzel a stratégiával a nyerési esélyei kétszer akkorára nőnek, mint a játékos első választásakor voltak.

A programozó matematikai tudás azonban nem kell, hogy szervesüljön, nem kell, hogy tudatosuljon. A programozó „csak” egyszerűen belátja a HA()-függvény két ágának egymás utáni kezelése keretében a számára triviális igazságot, mely egyben a matematikai igazság is.

A programozó nem mint matematikus jut el és látja be az igazságot, hanem az erőből való megoldás logikája quasi kikényszeríti belőle azt.

A programozónak nem kellett volna szűrleteket sem létrehoznia, elég lett volna egyetlen egy sornyi Excel-megoldást kialakítania, ha utána a véletlenszámokat (F9 = Calculate) a forráskódból (macro-ból) akarja értelmezni. A macro naplózta (loggolta) volna a beállításokat és a nyerés/vesztés állapotait. S ezeket a log-adatokat lehetett volna tetszőlegesen nagy ciklusváltozó mellett kiértékelni, ábrázolni (vö. log-logg-logg munkalapok). Az erőből való megoldás a sikerarány (a valószínűség) becsléséhez/levezetéséhez volt/lett volna csak szükséges ebben az esetben is, mert az egysoros esetértelmezés kapcsán már önmagában rá kellett volna jönni a fenti értelmezési kényszerpályára, amennyiben a programozó a siker-direkt állapot két ágát egymástól elkülönítve próbál HA()-függvényeket alkotni.

A konklúzió nem lehet más, mint az, hogy nincs fekete-fehér határvonal a matematikainak nevezhető idealizált tudás és az erőből megoldást kereső (a matematikai tudást minimalizálni akaró) stratégiák között, mert a valós példák kapcsán a két megközelítés egyetlen egy képletben tételesen és megkerülhetetlenül összeérhet. Innentől már csak a matematikai megoldás tudatosultsági szintje marad nyitott kérdésként, ami pedig annyira egyén és helyzetfüggő, hogy annak tárgyalása itt és most nem is kerül felvállalásra.

Felmerülhet a gyanú ennek kapcsán: Ha az összes eddigi szoftver minden forráskód-variánsa egy hatalmas korpuszban rendelkezésre állna, akkor nem kizárt, hogy programozók komoly matematikai kérdéseket oldottak meg ihletett állapotban – de talán sosem tudatosítva, mit is tettek a matematika érdekében/kapcsán/területén: vö. <https://hu.wikipedia.org/wiki/Sr%C3%ADniv%C3%A1sza_R%C3%A1m%C3%A1nudzsan>, ill. <https://en.wikipedia.org/wiki/Srinivasa_Ramanujan>

# A véletlen reprezentációjának kockázatai

A fentebb jelzett XLSM 3 munkalapja is impulzusokat ad a hiba-feltáró gondolkodásmód edzéséhez. A megértéshez továbbra is a Monty-Hall paradoxon erőből történő kezelését, mint célt kell szem előtt tartani. A három munkalap (önellenőrző szemléletmódja ellenére) olyan logikai hibákat mutat be, melyek felismerésének képessége (más és/vagy saját programozási feladataink előkészítő tesztelése) megkerülhetetlen elvárás egy-egy IT-végzettség megszerzése esetén:

* A mi-a-hiba munkalap J-oszlopa a kritikus tényező, lévén a műsorvezető NEM dönthet azon vitrin kapcsán, ahol deklaráltan a nyeremény található.
* A mi-a-hiba-2 munkalap Z oszlopában szereplő képlet esetén hibás az E-oszlopra való hivatkozás, aminek W-oszlopnak illene lennie, hiszen a végső siker számításakor a játékos esetleges (Y-oszlop által vezérelt) döntés-változását is figyelembe kell venni – ennek hiányában a sikerarányok jelentősen csökkennek.
* Az „aranyok” munkalapon a hiba nem triviális, látszólag minden ellenőrzés rendben van, s a log-logg-loggg vizualizációk (s az ezeket támogató macro-k) még sem vezetnek a stratégiaiváltás és az első döntés preferálása között elvárt 2-szeres eltéréshez.
  + Ennek oka a C4, D4, E4 cellákban keresendő.
  + Míg a C4 cella valóban klasszikus véletlenszámokat tartalmaz - vö. C4=VÉLETLEN.KÖZÖTT(0;1)
  + Addig már a D4 figyelembe veszi a C4 tartalmát annak érdekében, hogy elkerülhető legyen bármely sor esetén is az ellenőrző értékek általi hibajelzés - vö. D4=HA(C4=1;0;VÉLETLEN.KÖZÖTT(0;1))
  + Az E4 pedig immár a C4 és a D4 alakulását is figyeli – vö. E4=HA(SZUM(C4:D4)=0;1;0)
  + A fenti vízesés-szerű vezérlés eredményét a 106. sor, ill. maga a „csak” munkalap 1. sora mutatja be:
    - A vitrin1 választásának esélye nem 1/3 (hanem kb. 50%)
    - A vitrin2 választásának esélye nem 1/3 (hanem csak kb. 25%)
    - A vitrin3 választásának esélye nem 1/3 (hanem csak kb. 25%)
  + Mivel az „aranyok” munkalap egy fix vitrin-tartalom esetében (C2:D2:E2 = K:K:A) vizsgálja a lehetséges véletlenszerűen remélt játékos/műsorvezető döntéseket, így a vitrin1 esetében a nem-nyerés primer kb. 50%-os aránya ezen választás véletlenszerű lecserélése után a klasszikus 1/3-os arányokhoz képest erőteljesebben növeli a csere utáni sikerarányt: vö. sikertelen primer döntés 50+25=75, ill. 100-75=25, s végül 75/25=3 (vö. log-logg-loggg munkalapok).
  + További torzulást okozhatnak az R-S-T-oszlopok véletlenszámot használó képletei, mert ezek is úm. „figyelnek” – vö. R4=HA(H4=1;1;HA(C4=1;VÉLETLEN.KÖZÖTT(0;1);""))
  + A végső hatás pl. az U106;V106;W106 cellákban már érezhető…

Konklúziók: véletlenszám-generátor integrálása egy erőből megoldást kereső rendszerbe úgy, hogy arra is azonnal igyekszik figyelni a szakértő, hogy ne is generáljon egyetlen egy, az értelmezési intervallum-szabályokat sértő esetet, nagy eséllyel vezet oda, hogy a teljes véletlenszerűség egy/több hatás alá kerül a (feltételességek okán), ami végső soron sok fáradság árán téves megoldáshoz (arányszámokhoz, valószínűségekhez) vezet.

# Referenciák

…szövegközben…