A szakmai problémától a pszeudo-kódig, avagy a context-free szómágia és az Excel kapcsolata

(From the context-related problem to a pseudo-code, or magic of words in a context-free way leading to an Excel-demo)

Pitlik László, Biró Bálint, Pitlik Marcell, Pitlik László (jun) - MY-X team

Kivonat: A pszeudo-kódok elvi célja az kellene, hogy legyen, hogy annyira mentesítsék a tervezett algoritmusról szóló kommunikációt a szómágiától, amennyire csak lehetséges, de még ne kelljen semmilyen programnyelvi tudással rendelkezni ennek az átiratnak az értelmezéséhez az alapvetően programozni nem tudó Megrendelő esetében. Azaz olyan (minél inkább context-free) állapotba hozzuk egy adott probléma leírását, hogy quasi bármilyen programozási nyelven azonnal meg lehessen írni a szóban forgó algoritmus egy már hatásos verzióját. A cikk egyik célja, hogy egy élő példa többlépéses feldolgozását esettanulmányként bemutatva érzékeltesse a kontextustól való elszakadás lépéseinek mibenlétét, egy fajta (akár ad hoc) szabványosság felé közeledést és végül az Excel szerepét, mint demo-számításként pszeudo-megközelítésszerűséget nyújtani képes keretrendszert. A másik cél a programozó és a mindenkori Megrendelő közötti kommunikációs problémák egyes részleteinek bemutatása annak érdekében, hogy a leendő informatikusok minél több szimulált tapasztalat birtokában találkozzanak élő Megrendelőkkel. A cikk végül a konkrét probléma esetén arra a következtetésre jut, hogy előállhatnak esetek, amikor a megértési folyamat elakad…

Kulcsszavak: specifikáció, igényfelmérés, rendszerterv, tesztelés

Abstract: The principial goal of a pseudo-code should be the minimizing of the magic of words without needing any knowledge about programming languages – especially in case of the stakeholders. With other words: the problem description where a source code should be delivered must be transferred into a quasi context free description allowing source codes in arbitrary languages. The paper demonstrates a case study about a real problem and the communication process between the partners. This communication should ensure a quasi context free version of the problem description starting from a version where the real context and the IT-problem are mixed in a deep level and the basic version of the description is not complex enough to support the necessary clarification levels. This communication process will end in an Excel demo which might be seen as a kind of pseudo code – but unfortunately, not the final one – at all. Parallel, the case study presents typical scenarios of the communication processes between partners during the understanding phases. One of the conclusions is it is not trivial to realize a pseudo-code in any case…

Keywords: specification, requirements, system planning, software testing

# Bevezetés/előzmények

A pszeudo-kód irányából való közelítés:

* <https://miau.my-x.hu/bprof/autodidakta_kritikus_tanulas_mintazatai.docx>
* <https://miau.my-x.hu/bprof/autodidakta_kritikus_tanulas_mintazatai_p2.docx>
* <https://miau.my-x.hu/bprof/autodidakta_kritikus_tanulas_mintazatai_p3.docx>

A szómágia kezelésének irányából való közelítés:

* <http://miau.my-x.hu/miau/201/20150508.doc>
* <https://miau.my-x.hu/miau/quilt/Definitions_of_knowledge.docx>
* <https://miau.my-x.hu/miau/quilt/20Q.docx>
* <https://miau.my-x.hu/myx-free/index.php3?x=test1>
* <https://miau.my-x.hu/miau2009/index.php3?x=e0&string=letra>

# Kiindulási helyzet

„Lenne egy kérdésem. Pszeudo-kódban sikerült valamit kibogarásznom, de implementálni már nem tudom... Kinek van valamilyen ötlete a probléma megoldására?

Adott a következő adatstruktúra:

[[id1, 1,15,14],

[id2, 5,40,35],

[id3, 50,100,50],

[id4, 2,15,13]]

Az id ténylegesen egy azonosítót jelent. Az első szám egy szekvencia kezdőpontja, a második a végpontja, a harmadik szám pedig a kettő közötti pontok száma. Ez az összes elem egy adott referencia szekvenciára illeszkedik.

A feladat: a lehető legnagyobb lefedettség elérése a lehető legkevesebb szekvencia felhasználásával. Fehérje szekvenciákról van szó, amelyeknek nitrogén felőli végét N terminálisnak, szén felőli végét pedig C terminálisnak nevezik.

A pszeudo kódom valahogy így néz ki (a kisebb/nagyobb itt indexbeli különbséget jelent):

Két elem (x es y) kivétele a halmazból, majd a kettő összehasonlítása, ahol:

Ha x startpontja legalább 3-mal kisebb, mint y startpontja, akkor tartsd meg x-et, máskülönben töröld ki x-et es tartsd meg y-t! (Ez az esete a N terminálison túlnyúló szekvenciának).

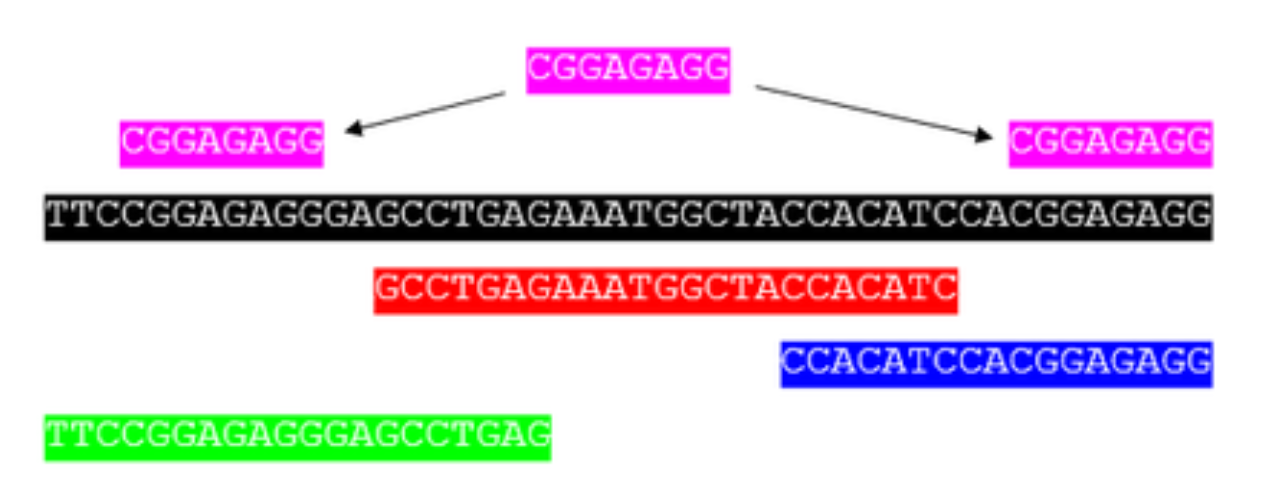
Ha x endpontja legalább hárommal nagyobb, mint y endpontja, akkor tartsd meg x-et, máskülönben töröld ki x-et es tartsd meg y-t! (Ez az esete a C terminálison túlnyúló szekvenciának).

Ha x kezdőpontja nagyobb, mint y végpontja, akkor tartsd meg mindkettőt! (Ez az esete a két független szekvenciának, ahol y “upstream” a referenciában).

Ha x végpontja kisebb, mint y kezdőpontja, akkor tartsd meg mindkettőt! (Ez az esete a két független szekvenciának, ahol y “downstream” a referenciában).

Valahogy így néz ki vizuálisan. A fekete szekvencia a referencia. Ezek nukleinsav szekvenciák, de az elv ugyanaz: <https://upload.wikimedia.org/wikipedia/commons/thumb/d/dd/Seqassemble.png/450px-Seqassemble.png?fbclid=IwAR32UDyEBH1vL_a4rCG2RZAuLRKiXX7P-txpE5iJRiuC-rB1b-jHm77e9aA>

(vö. alább integrált kép-melléklet)”



# Kiindulási helyzet és ennek első értelmezési kísérletei

Az alábbi korrektúra-verzió a Megrendelői szándékok megértését akadályozó rétegeket, vagyis a típusanomáliákat (ezek egy részét) mutatják be:

Lenne egy kérdésem. Pszeudo-kódban sikerült valamit kibogarásznom, de implementálni már nem tudom... Kinek van valamilyen ötlete a probléma megoldására?

Adott a következő adatstruktúra:

Ssz, kezd, vég, táv

[[id1, 1,15,14],

[id2, 5,40,35],

[id3, 50,100,50],

[id4, 2,15,13]]

Az id ténylegesen egy azonosítót jelent. Az első szám egy szekvencia kezdőpontja, a második a végpontja, a harmadik szám pedig a kettő közötti pontok száma (minek kell ezt tárolni, ha számítható). Ez az összes elem egy adott referencia-szekvenciára🡨ez mi is pontosan? Miért nincs erre is egy példa? Tekinthetjük tetszőlegesen hosszú alfabetikus/alfanumerikus/numerikus jelnek? illeszkedik.

A feladat: a lehető legnagyobb lefedettség🡨hogyan mérjük ezt a fogalmat? elérése a lehető legkevesebb referencia?-szekvencia 🡨más szekvencia? felhasználásával. Fehérje🡨irreleváns szómágia szekvenciákról van szó, amelyeknek nitrogén felőli végét N terminálisnak, szén felőli végét pedig C terminálisnak nevezik🡨irreleváns szómágia, már vélhetően a szekvencia szó is irreleváns szómágiának számít ott, ahol a pontos tartalomhoz a biotechnológiailag laikus laikus programozónak ennyi még kevés, de bőven elég lenne a mező típusát megadó leírás (pl. tetszőleges hosszúságú szöveg?).

A pszeudo kódom valahogy így néz ki (a kisebb/nagyobb itt indexbeli🡨nem lett pontosan bevezetve különbséget jelent):

Két elem (x es y 🡨van ezen változóknak köze a sárga-türkiz-lila-zöld színekhez? Leginkább pl. a türkizhez és/vagy a lilához?) kivétele a halmazból, majd a kettő összehasonlítása, ahol:

Ha x startpontja=kezdőpont? legalább 3-mal kisebb, mint y startpontja🡨a lila végpont nem kell semmihez?, akkor tartsd meg x-et, máskülönben töröld ki x-et es tartsd meg y-t! (Ez az esete a N terminálison túlnyúló szekvenciának).

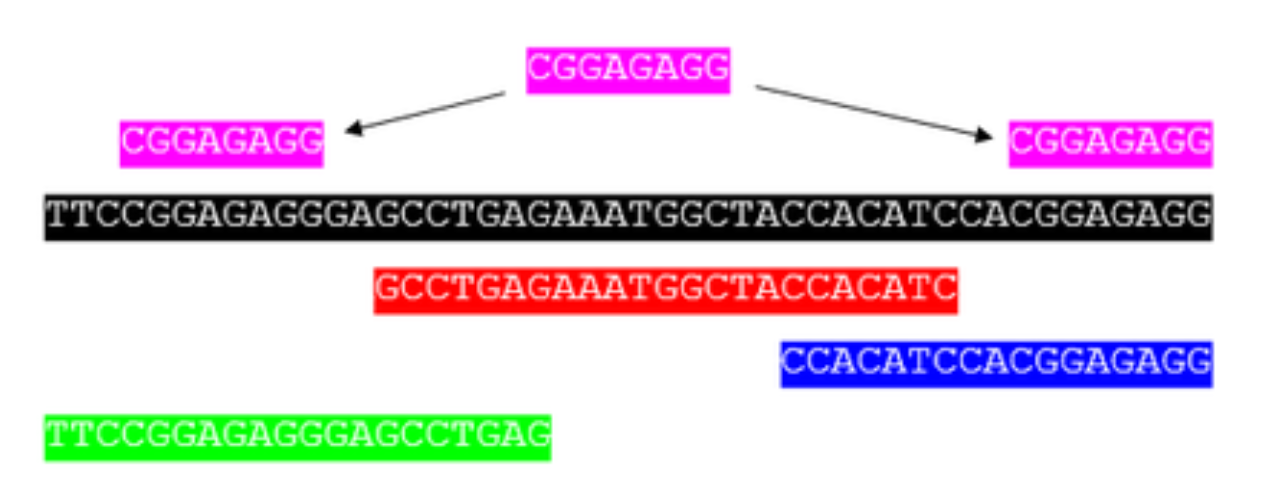
Ha x endpontja legalább hárommal nagyobb, mint y endpontja=végpont?, akkor tartsd meg x-et, máskülönben töröld ki x-et es tartsd meg y-t! (Ez az esete a C terminálison túlnyúló szekvenciának).

Ha x kezdőpontja nagyobb, mint y vég=end?pontja, akkor tartsd meg mindkettőt! (Ez az esete a két független szekvenciának, ahol y “upstream” a referenciában).

Ha x végpontja kisebb, mint y kezdőpontja, akkor tartsd meg mindkettőt! (Ez az esete a két független szekvenciának, ahol y “downstream” a referenciában🡨context free szempontból felesleges szómágia).

Valahogy így néz ki vizuálisan. A fekete szekvencia a referencia = szóköz nélküli alfabetikus jelsorozat, mely maximális hossza nincs meghatározva. Ezek nukleinsav szekvenciák, de az elv ugyanaz: <https://upload.wikimedia.org/wikipedia/commons/thumb/d/dd/Seqassemble.png/450px-Seqassemble.png?fbclid=IwAR32UDyEBH1vL_a4rCG2RZAuLRKiXX7P-txpE5iJRiuC-rB1b-jHm77e9aA>

(vö. alább integrált kép-melléklet, ahol a színeket be kellett volna emelni a fenti magyarázatba?)



# Pontosító szóbeli kommunikáció és ennek eredményei – I

A korrektúra verzióra nem született írásos válasz a Megrendelőtől, de a szóbeli egyeztetés azonnal létre jött és az alábbi eredményekre vezetett:

Lenne egy kérdésem. Pszeudo-kódban sikerült valamit kibogarásznom, de implementálni már nem tudom... Kinek van valamilyen ötlete a probléma megoldására?

Adott a következő adatstruktúra:

[[id1, 1,15,14],

[id2, 5,40,35],

[id3, 50,100,50], id3 lefedettség 50%, ha a referenciaszekvencia hossza 100 karakter

[id4, 2,15,13]]

Id-mátrix lefedettség = nem értelmezendő (de a többszörösen lefedett aminosavak/betűk egynek számítanak, tehát a lefedettség maximuma 100%)

40 és 50 között van egy lyuk (41-49 = 9 db betű)

Ideális kimenet/output: id1+id2+id3, mert id4 része id1-nek…

Id(végtelen)-ig tart, a referencia-szekvencia hossza kombinatorikailag limitálja, a valóságban 2-20 közötti a sorok száma…

Az id ténylegesen egy azonosítót (alobjektum-név) jelent. Az első szám egy szekvencia kezdőpontja, a második a végpontja, a harmadik szám pedig a kettő közötti pontok száma (alobjektum-hossza, de nem kell, mert számítható). Ez az összes elem egy adott referencia szekvenciára illeszkedik.

A feladat: a lehető legnagyobb lefedettség elérése a lehető legkevesebb szekvencia felhasználásával. Fehérje szekvenciákról van szó, amelyeknek nitrogén felőli végét N terminálisnak, szén felőli végét pedig C terminálisnak nevezik.

**Inputok**

Referencia szekvencia (OBJEKTUM) = tetszőleges hosszúságú alfabetikus karaktersor (fehérjeszekvencia 20 betűs karakterkészlettel dolgozik) – vö. demo-kép fekete-fehér sor

Id-mátrix: egyes sorai a referencia szekvencia ALOBJEKTUMA (a megfelelő kezdő és végpontokkal)

X = n. id-mátrix sor

Y = (n+1). id-mátrix sor

Index = l. alább

**Outputok**

Lefedettség = vö. előbb / a többszörösen lefedett betűk még fontosak lesznek/lehetnek

Legkevesebb vs. legtöbb = bármely id-variáció (n alatt a k) esetén számítandó a lefedettség és ezt kell osztani a felhasznált id-k számával?!

Vagyis az n alobjektum által legalább egyszer érintett betűk darabszáma osztva n-nel ad egy maximalizálandó skálát, ahol, ha azonos (adott kerekítési szabályok mellett!!!) a betűkösszszáma/n érték, akkor az n maximumát preferáljuk. Lehet, hogy több alternatív megoldás is kezelhet azonos! n-t!

S mindezen számítást a leggyorsabban… (vélelmezhetően nem tisztán kombinatorikai alapon)…

A pszeudo kódom valahogy így néz ki (a kisebb/nagyobb itt indexbeli különbséget jelent – egy id mátrix soron belüli x és x értékek, de az index fogalomra egyelőre nincs szükség):

Két elem (x es y) kivétele a halmazból, majd a kettő összehasonlítása, ahol: preferencia-számítás, ha az n-ek is azonosak a fenti számítás menet végén?!

Ha x (n) startpontja legalább 3[[1]](#footnote-1)-mal kisebb, mint y (n+1) startpontja, akkor tartsd meg x-et, máskülönben töröld ki x-et es tartsd meg y-t! (Ez az esete a N terminálison túlnyúló szekvenciának).

Ha x endpontja legalább hárommal nagyobb, mint y endpontja, akkor tartsd meg x-et, máskülönben töröld ki x-et es tartsd meg y-t! (Ez az esete a C terminálison túlnyúló szekvenciának).

Ha x kezdőpontja nagyobb, mint y végpontja, akkor tartsd meg mindkettőt! (Ez az esete a két független szekvenciának, ahol y “upstream” a referenciában).

Ha x végpontja kisebb, mint y kezdőpontja, akkor tartsd meg mindkettőt! (Ez az esete a két független szekvenciának, ahol y “downstream” a referenciában).

Mind a négy szabály alapján előáll egy preferencia-értéksor (4 elem): 0000-1111 = 2^4

Az erőből való megoldás alternatívája, mely alternativitás matematikailag bizonyítandó.

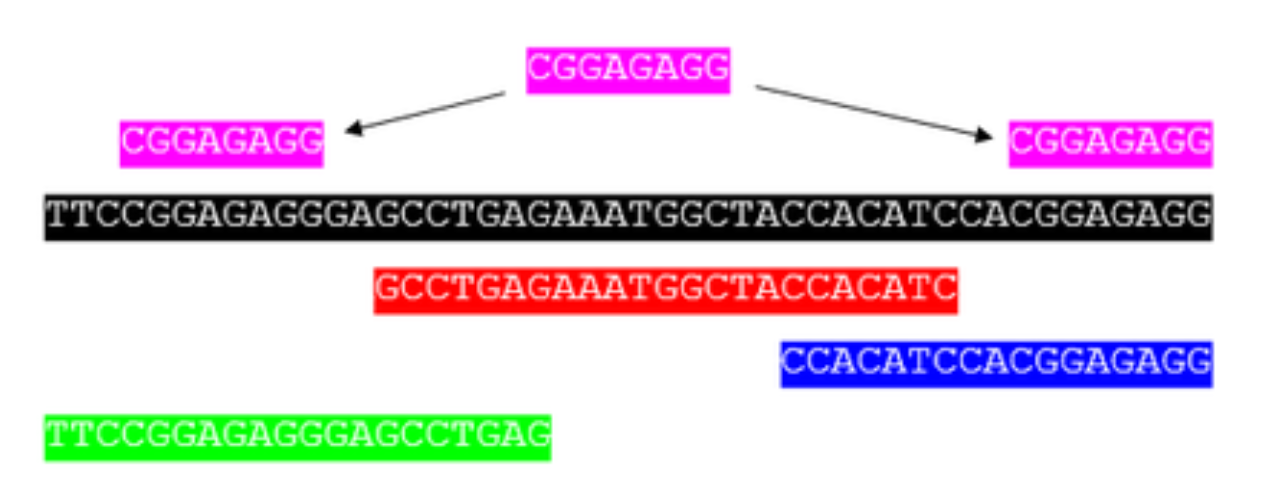
A fenti 4 szabály (szabálylánc) 2 elemű (id) megoldásokra igaz csak!

3+ elemű megoldások szabályrendszere még nem adott?!

MINTAPÉLDA (input és output) minden esetben szükséges lenne az első pillanattól kezdve!

Valahogy így néz ki vizuálisan. A fekete szekvencia a referencia. Ezek nukleinsav szekvenciák, de az elv ugyanaz: <https://upload.wikimedia.org/wikipedia/commons/thumb/d/dd/Seqassemble.png/450px-Seqassemble.png?fbclid=IwAR32UDyEBH1vL_a4rCG2RZAuLRKiXX7P-txpE5iJRiuC-rB1b-jHm77e9aA>

(vö. alább integrált kép-melléklet)



A korrektúra jelek a szóbeli egyeztetés folyamata alatt keletkeztek, de nem az olvasás szerinti sorrendben, hanem mindenkor ott rögzítve egy-egy gondolatfoszlányt, ahol az a leginkább adekvátnak tűnt a beszélgetés menetében. (Itt kell megjegyezni, hogy a kronologikus jegyzetelés lehetősége is adott lenne, ha pl. egy wiki-rendszerben szócikk formájában készülne egy beszélgetés jegyzőkönyve, ahol minden egyes önálló feljegyzési egység után mentés történne, melyek helye megtartaná a leginkább logikus helyre vonatkozó preferenciákat, de a szócikk laptörténete alapján a feljegyzések keletkezésének sorrendje is követhető lenne. Ideális esetben egy dokumentum keletkezéstörténete a laptörténet animáció-vezérlő jelként való értelmezését jelentené, vagyis a dokumentum korrektúrái úgy lennének bevillantva lépésről lépésre, mintha ezek most keletkeznének éppen – azzal a finomhangolással, hogy a laptörténet valós időben loggoló időpecsétjei helyett a történések a szimulált keletkezéstörténetben metronóm-szerűen, azaz nem időarányosan történhetnének – felgyorsítandó a történet információértékének kinyerését.)

A beszélgetés lényege az alapmegrendelésből hiányzó input-output példa szóbeli közelítése és későbbi tételes bekérések előkészítése volt. Már a matematika oktatás is úgy igyekszik kondicionálni a gyermekek gondolkodását, hogy egy gépnek nevezhető fekete doboz bejárati nyílásán (input) pl. bedobni enged két számot (pl. 3 és 5), majd a kijövő nyíláson megjelenik a példa kapcsán a 8-as érték. S a kérdés csak annyi: milyen műveletet végez a fekete doboz? Hasonló problémákkal van tele az Internet és ezeket a „találós kérdéseket” előszeretettel használják a HR-szakértők is adott pozícióra leginkább alkalmasnak vélt személyek kiválasztásához: vö. <https://miau.my-x.hu/bprof/potencialis_feladatok>

A szóbeliség keretében minden fogalmat, szimbólumot, változót tételesen és pontosan definiálni kellett, lehetőség szerint önálló példákkal kizárva a félreértéseket.

A szóbeliség feltárt egy ösztönös (alapvetően kombinatorikai, erőből való megoldást megalapozó) gondolkodásmódot és rámutatott arra, hogy a megrendelésben szereplő 4 szabály lényegében az erőből való megoldás egy potenciálisan gyorsabb, matematikailag kiérleltebb alternatívája. A matematikai alternatív megoldás előzetes és alapos bizonyítást igényelhet olyan esetben, amikor az algoritmus valóban tétre menően, tömegesen használnak majd olyan (főleg) laikusok, akik immár vakon fognak bízni az eredményekben.

A 4 szabályra alapozó algoritmus esetén a szóbeli egyeztetés után, ennek eredményeit elemezve öltött testet a vélelem, hogy a „kizárás” és „megtartás” kifejezések alapján lényegében az id-mátrix egyes sorainak, avagy az alobjektumoknak a megoldásból való kizárása, ill. a megoldáshalmazban való megtartása a vezérlés logikája, mely így egy addig futó algoritmust jelent, amíg további kizárásra már nem kerül sor abban a folyamatban, ahol az x és y, ill. az n. és az (n+1). id-mátrixsorok mibenléte (kijelölése) marad már csak kihívás. A mindenkori x és y kijelölésének mibenléte alapjaiban befolyásolja a futásgyorsaságot. Az x és y fogalmak léte arra enged következtetni, hogy nincs 3+ elemű megoldáskezelési ágra szüksége az algoritmusnak, mert a végső megoldásban annyi id-mátrixsor fog maradni, ami a kizárások után kell, hogy maradjon.

A 4-szabályra alapozó potenciális eredmény és a potenciális alternatív megoldásba torkollás utána esetleges preferenciák matematikai világa vélelmezhetően nem kényszerűen kell, hogy azonos legyen, mert a 4-szabályra alapozó algoritmus elvileg mindenkor egyetlen egy megoldást illene, hogy adjon, mely megoldás formálisan nem fejezi ki a lefedettségarány/alobjektumszám alapján azt a mutatószámot, mely maximalizálandó, s mely értéke kapcsán alternativitás előállhat, különösen, ha a kerekítés fogalmát is be kell vonni az értelmezésbe.

További releváns felismerés, hogy pl. a referencia szekvencia jelenségére semmi szükség, mert egyedül és kizárólag az id-mátrix, abból is csak a kezdő és a végpont koordinátája a való input, már a név sem releváns, ha a sorszámot információértéknek tekintjük, s a végpont-kezdőpont távolsága eleve nem szükséges. Az output pedig nem más, mint a bemeneti id-mátrix szűkített (kizárások utáni nézete), vagy egy új oszlop az id-mátrixban: adott sor esetén a nulla jelölje a kizárva, 1 jelölje a megtartva végállapotot (mely státuszváltozó egy egyszer bármilyen okból kizárt (0) sor esetén soha nem válhat elvileg visszavett = megtartott (1) állapotúvá.

További értelmezési lehetőségek merülnek fel akkor, ha a 4-es szabályt szentírásnak tekintjük és feladatként úgy kell megalkotni a szómágikus probléma-leírást, hogy annak (erőből dolgozni akaró) kombinatorikai és a 4-es szabályt követő lépései kényszerűen azonos eredményre vezessenek. Ugyanis egyelőre fennáll a veszélye annak, hogy: míg a kombinatorikai megoldás biztosan A Megoldást keresi és találja meg, addig a 4-es szabály alapján történő egy-egy kizárás után adott al-objektum már nem tud visszakerülni a megoldási térbe, így esetlegesen nem minden variáció kerül nagyító alá kényszerűen. Ennek kapcsán ismét felmerül az (n+1). elem mibenléte – ugyanis az id-mátrix sorainak sorrendje elvileg bármi lehet – VAGY az id-mátrix előrendezendő, pl. elsőként a kezdőértékek, majd azonosság esetén a végértékek alapján.

# Az ideális feladatleírás közelítése

Legyen adott egy táblázat, melynek 2 oszlopa és tetszőlegesen sok (n db) sora lehet, ahol a 2 oszlopból a mindenkor baloldaliban szereplő érték kényszerűen kisebb, mint a jobb oldali és mindkét érték pozitív egész szám, vagyis minden sor egy-egy természetes számokból álló sorozat első és utolsó elemét adja meg. (Valós helyzetben a sorok száma n 20 – ami a tesztelést könnyítő információ, de az algoritmust tetszőlegesen nagy sormennyiségre kell tervezni ettől még.)

Generáljuk a táblázat sorainak összes lehetséges részhalmazát. Egy lehetséges részhalmaz tartalmazhat 1 és n között tetszőleges darabszámú sort, a részhalmazok összes száma .   
Ld. <https://www.wolframalpha.com/input/?i=sum+from+k%3D1+to+n+%28n+choose+k%29>

Vezessük be a lefedettség fogalmát, mely a táblázati sorokból képzett egyes részhalmazokban a sorozatok által legalább egyszer érintett pozitív egész számok darabszámát jelenti (f) – vö. select distinct, ahol nem számít, egy adott pozitív egész szám hány táblázati sor kapcsán válik lefedetté. (A valóságközeliség érdekében szükséges megjegyezni, hogy a táblázatban jelölt sorozatok által érintett számok egy ismert (m) hosszúságú minta részei, így szemléletes lehet az egyes részhalmazok által generált lefedettséget az f/m aránnyal (%) is értelmezni.

Határozzuk meg ezek után a táblázati sorok halmazának összes lehetséges részhalmazára vonatkozóan az általuk generált lefedettségi mértéket (f).

Vezessük be a relatív lefedettség fogalmát, mely legyen: a táblázati sorok adott részhalmazához tartozó lefedettség (f) osztva az adott variánsban szerepet kapó sorok (k) számával (1kn).

**S végül a feladat maga: keressük a relatív lefedettségi arány (f/k) maximumát. Amennyiben több különböző részhalmaz is azonos relatív lefedettségi arányra vezetne, akkor preferálandó az a megoldás, melyhez több táblázati sort használtunk fel.**

Gondolatkísérlet – és kérdés

Vegyük az alábbi potenciális részhalmazokat:   
S1 esetében f=10, k=15 --> a rel. lefedettség 2/3   
S2 esetében f=10, k=10 --> a rel. lefedettség 1   
S3 esetében f=5, k=5 --> a rel. lefedettség 1

A fenti S1-S3 opciók közül a vázolt algoritmus először elveti S1-et (f/k maximalizálás), majd S3-at (k max.) és a győztes az S2.

Megjegyzés: talán didaktikusabb volna ugyanezen eredményt az alábbi algoritmussal elérni:   
elsődleges cél az f maximalizálása (S3 elvetése), majd ezt követően a k minimalizálása (S1 elvetése).

Vegyük ezt követően az előző három eset mellé az alábbi részhalmazt is:   
S4 esetében f=5, k=1 --> a rel. lefedettség 5

A vázolt algoritmus S1-S4 közül egyértelműen győztesnek hozza ki az utolsót, amely döntés megkérdőjelezhetőnek tűnik abból a szempontból, hogy racionális várakozás volna továbbra is az S2 megoldást tekinteni a referencia-szekvencia legjobb lefedésének.

Megjegyzés: az előző megjegyzésben vázolt algoritmus képes ezt biztosítani.

KÉRDÉS: a gondolatkísérlet előtti utolsó (kiemelt) bekezdésben vázolt algoritmus helyesen értelmezi az eredeti elemzői szándékot, s ha igen: valóban ez (S4>S2) a kívánatos elemzési cél?

# Példakényszer

A fentiek alapján egy konkrét példa kialakítása a Megrendelő által elkerülhetetlen, s íme, a példa maga (kontextushoz kötéssel együtt):

A5YKK6 a referencia szekvencia szakmai körökben használt id-jára vonatkozóan legyen ismert (input):

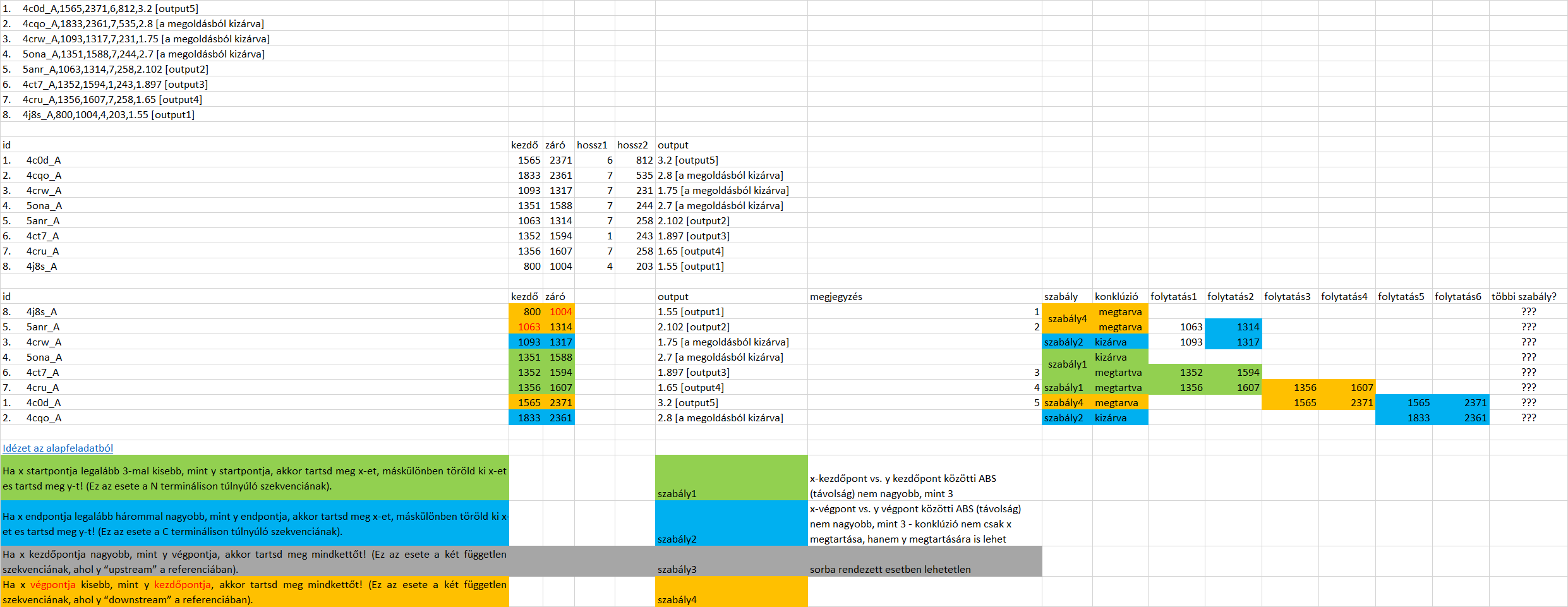
1. 4c0d\_A,1565,2371,6,812,3.2 [output5]
2. 4cqo\_A,1833,2361,7,535,2.8 [a megoldásból kizárva]
3. 4crw\_A,1093,1317,7,231,1.75 [a megoldásból kizárva]
4. 5ona\_A,1351,1588,7,244,2.7 [a megoldásból kizárva]
5. 5anr\_A,1063,1314,7,258,2.102 [output2]
6. 4ct7\_A,1352,1594,1,243,1.897 [output3]
7. 4cru\_A,1356,1607,7,258,1.65 [output4]
8. 4j8s\_A,800,1004,4,203,1.55 [output1]

Ahol immár csak a kezdő és a záró érték a releváns, minden más csak kontextusba helyezve nyer értelmet. A []-ben szereplő utalások már az alábbi output-listával való kapcsolatát írják le az input-soroknak:

Optimális output:

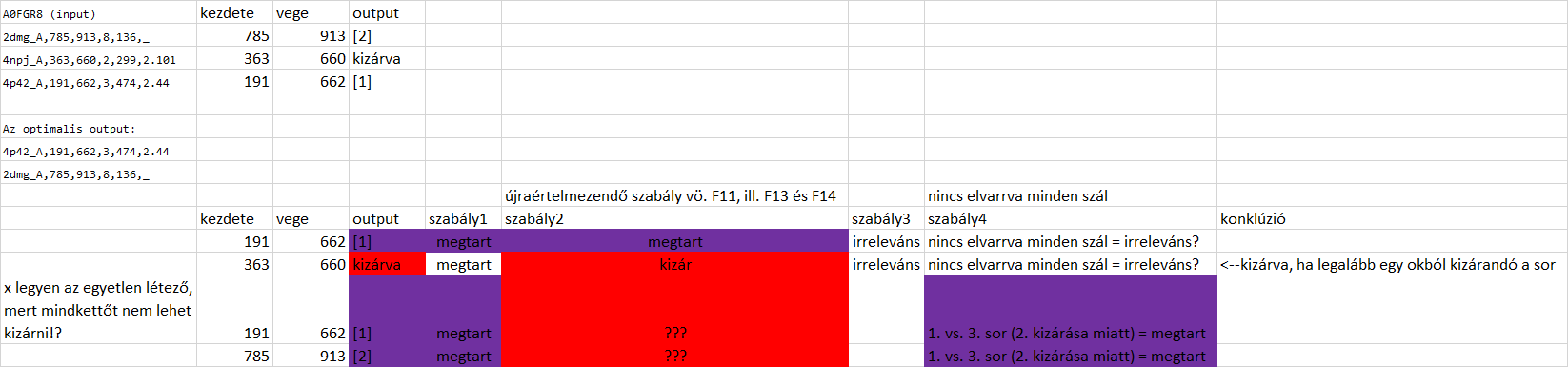
1. 4j8s\_A,800,1004,4,203,1.55
2. 5anr\_A,1063,1314,7,258,2.102
3. 4ct7\_A,1352,1594,1,243,1.897
4. 4cru\_A,1356,1607,7,258,1.65
5. 4c0d\_A,1565,2371,6,812,3.2

Részeredmények az id-mátrix megrendezése után, ahol a kezdő értékre vonatkozóan a legkisebbtől a legnagyobbig tart az egytényezős rendezés, s ahol ezután az optimális output-ként megadott sorrend tételesen visszaköszön egy-egy kizárással tarkítva úgy, hogy a megtartó és kizáró szabályok tételesen az output-sorhoz rendelhetők:

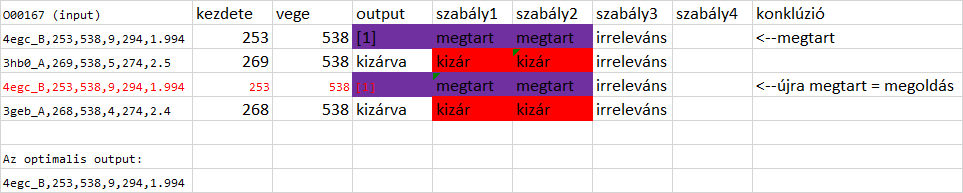


Problémák a látszólagos reprodukálhatósági siker mögött:

* mind a 4 szabályt egyszerre illene értelmezni
* a 3. szabály rendezett id-mátrix esetén nem is képes hatni
* az 1. és a 2. szabály szövege pontosítandó
* a 3-egység távolság ezek szerint mégis csak ténylegesen releváns
* a mi számít x-nek és mi számít y-nak adott szabály általi kizárás/megtartás után nem triviális (vö. folytatás1-2-3-4-5-6 vs. [1351,1588]-adatpár kezelése egy kizárás után (nem a kizárt az x és nem is az azt megelőző, hanem a [1351, 1588] lesz az x és az őt követő az y – vélelmezhetően)



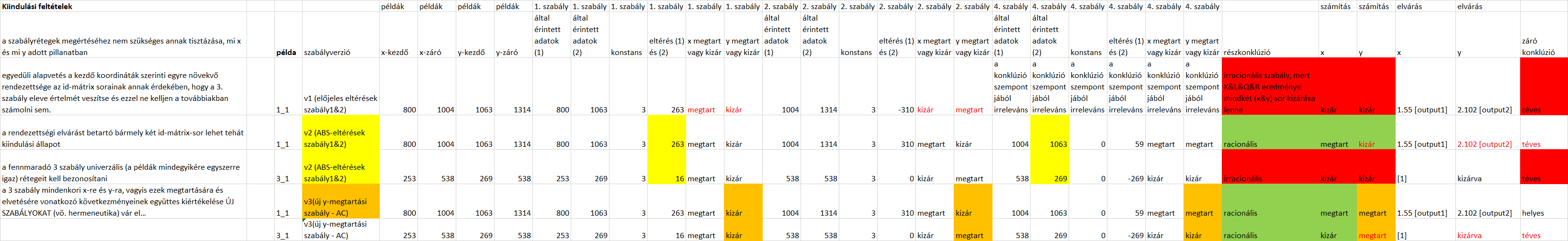
Konklúzió: a szabályok pontos szövege minden példára egységesen újra-gondolandó, finomhangolandó, a mi az x és mi az y vezérlés véglegesítendő ismét csak minden példára egységesen.



Konklúzió: a szabályok pontos szövege minden példára egységesen újra-gondolandó, finomhangolandó, a mi az x és mi az y vezérlés véglegesítendő ismét csak minden példára egységesen.

Példák forrása: <https://miau.my-x.hu/digeco/2020/2020osz/pseudo.xlsx>

Az XLSX-ben a példa1 munkalapon még a szabályok csak manuális vannak értelmezve. A példa2 és a példa3 esetén az XLSX már quasi pszeudo-kód, mert a szabályok képletként vannak megadva, még ha részben lyukasan is (csak az adott esetre érvényes ágra kialakítva)…



Példák forrása: <https://miau.my-x.hu/digeco/2020/2020osz/pseudo.xlsx>

A szabályrétegek egységesítése x és y kizárása és/vagy megtartása kapcsán már 2 példa után sem vezet racionális állapotokhoz.

Tehát a szabályok tartalma esetlegesen zavaros, még inkább a szabályok együtt-értelmezése hiányos/zavaros, ami akár a szabály-elvűséget, mint alternatív megoldást ki is zárhatja a teljes előkészítési folyamatból ezen a ponton?!

A mellékletek kapcsán felmerül annak értelmezési kényszere is, hogy az eredetileg 4-elemű szabályrendszer pl. párhuzamos vagy soros kapcsolású-e. Amennyiben (vö. fentebb) párhuzamos kapcsolású, azaz minden szabály önálló részkövetkeztetésekre vezet, melyek alapján egy eredő következtetést jelentő +1. szabálynak is lennie kell, akkor tudjuk, hogy minden szabály minden x-y-párra hat. A párhuzamosság egy speciális értelmezése lehet az az egyszerűsítés, hogy adott x vagy y akkor tartandó meg, ha erre legalább egy ok van, s a kizárásról, mint olyanról aktív következményként nem is kell beszélni, mert ami nincs megtartva, az ki van zárva. Ha soros kapcsolásról van szó, akkor a szabályok alkalmazásának lehet egy sorrendje és az az x és/vagy y, mely adott szabály kapcsán kizárásra kerül, többet elvileg nem válik részévé az outputhoz vezető útnak, azaz nem minden szabály hat rá.

A +1. szabály lehetne éppen mennyiségi jellegű is a minőségi jellegű lehetőségek mellett, ahol minőségi egy zárószabály akkor, ha pl. elég egyszer előfordulni egy kizár/megtart-állapotnak, hogy azt tekintsük eredmény-állapotnak, s mennyisége a zárószabály, ha az a végső konklúzió, ami többször fordul elő.

# Konklúziók

Mindazok számára, akik vették a komoly fáradságot és minden részletre kiterjedően (esetlegesen már saját ellenszámításokkal is) követték a fentebb leírtakat, világossá kellett, hogy váljon: egy szoftver előkészítése nem triviálisan sikert ígérő feladat. Bizonyos esetekben az sem segít, ha a Megrendelői oldal saját kódírásba kezd, mert ennek is a közösen megértett és elfogadott tesztesetek illene, hogy az alapját adják. A tesztesetek ráadásul nem lehetnek akármilyenek, mert minél több (ideális esetben minden) szélsőséges inputhelyzetet le kell, hogy fedjenek…

# Mellékletek

Megrendelői kód-kísérletek:

{

"cells": [

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"mylist=[['id1',100,200],['id2',100,199]]\n",

"first=mylist[0]\n",

"second=mylist[1]\n",

"xbeg=first[1]\n",

"ybeg=second[1]\n",

"xend=first[2]\n",

"yend=second[2]\n",

"result=[]\n",

"if xbeg>yend or xend<ybeg: #ket teljesen kulonobzo szekvencia, ezert mindkettore szukseg van\n",

" result.append(first)\n",

" result.append(second)\n",

"elif xbeg==ybeg:#a kezdopontok azonosak, ezert a nagyobb vegponttal rendelkezo (hosszabb) kell\n",

" if xend>yend:\n",

" result.append(first)\n",

" else:\n",

" result.append(second)\n",

"elif xend==yend:#a vegpontok azonosak, ezert a kisebb kezdoponttal rendelkezo (hosszabb) kell\n",

" if xbeg<ybeg:\n",

" result.append(first)\n",

" else:\n",

" result.append(second)\n",

"elif xbeg<ybeg:\n",

" if xend>yend:#a masodik szekvencia benne van az elsoben, csak a hosszabb kell\n",

" result.append(first)\n",

" if xend<yend:#atfedes, az elso szekvencia downstream, mig a masodik szekvencia upstream hosszabb\n",

" result.append(first)\n",

" result.append(second)\n",

"elif xbeg>ybeg:\n",

" if xend<yend:#az elso szekvencia benne van a masodikban, csak a hosszabb kell\n",

" result.append(second)\n",

" if xend>yend:#atfedes, az elso szekvencia upstream, mig a masodik szekvencia downstream hosszabb\n",

" result.append(first)\n",

" result.append(second)\n",

"print(result)"

]

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3",

"language": "python",

"name": "python3"

},

"language\_info": {

"codemirror\_mode": {

"name": "ipython",

"version": 3

},

"file\_extension": ".py",

"mimetype": "text/x-python",

"name": "python",

"nbconvert\_exporter": "python",

"pygments\_lexer": "ipython3",

"version": "3.7.4"

}

},

"nbformat": 4,

"nbformat\_minor": 2

}

mylist=[['id1',100,200],['id2',100,199]]  
first=mylist[0]  
second=mylist[1]  
xbeg=first[1]  
ybeg=second[1]  
xend=first[2]  
yend=second[2]  
result=[]  
if xbeg>yend or xend<ybeg: #ket teljesen kulonobzo szekvencia, ezert mindkettore szukseg van  
    result.append(first)  
    result.append(second)  
elif xbeg==ybeg:#a kezdopontok azonosak, ezert a nagyobb vegponttal rendelkezo (hosszabb) kell  
    if xend>yend:  
        result.append(first)  
    else:  
        result.append(second)  
elif xend==yend:#a vegpontok azonosak, ezert a kisebb kezdoponttal rendelkezo (hosszabb) kell  
    if xbeg<ybeg:  
        result.append(first)  
    else:  
        result.append(second)  
elif xbeg<ybeg:  
    if xend>yend:#a masodik szekvencia benne van az elsoben, csak a hosszabb kell  
        result.append(first)  
    if xend<yend:#atfedes, az elso szekvencia downstream, mig a masodik szekvencia upstream hosszabb  
        result.append(first)  
        result.append(second)  
elif xbeg>ybeg:  
    if xend<yend:#az elso szekvencia benne van a masodikban, csak a hosszabb kell  
        result.append(second)  
    if xend>yend:#atfedes, az elso szekvencia upstream, mig a masodik szekvencia downstream hosszabb  
        result.append(first)  
        result.append(second)  
print(result)

lista=[['id1',20,40],['id2',28,49]]  
first=lista[0]  
second=lista[1]  
def mapping(first, second):  
    result=[]  
    fbeg=first[1]  
    sbeg=second[1]  
    fend=first[2]  
    send=second[2]  
    if fbeg<=sbeg-3:  
        if fend<=send-3:  
            result.append(first)  
            result.append(second)  
        elif fend>=send+3:  
            result.append(first)  
        elif send-3<fend<send+3:  
            result.append(first)  
    elif fbeg>=sbeg+3:  
        if fend<=send-3:  
            result.append(second)  
        elif fend>=send+3:  
            result.append(first)  
            result.append(second)  
        elif send-3<fend<send+3:  
            result.append(second)  
    elif sbeg-3<fbeg<sbeg+3:  
        if fend<=send-3:  
            result.append(second)  
        elif fend>=send+3:  
            result.append(first)  
        elif send-3<fend<send+3:#doesnt matter which one is added, decision can be done by resolution  
            result.append(first)  
    return result

# Alternatív megoldások

* 4-es szabály
* Brute-force
* Select-distinct (lefedettségek mértékei és ezek szórása)
* Felvételi rendszer-analógia

1. Ez a konstans akár fontos szereppel is bírhat?! [↑](#footnote-ref-1)