

Projekt dolgozat

Hozzárendeléses feladat modellezése Python-ben

Hallgatók:

Károlyi Krisztián, Kristó Csongor, Mezei Róbert

Csíkszereda

2023

Tartalomjegyzék

1.	Kivonat	2
2.	Bevezetés	3
3.	Elméleti háttér	4
3.1	Hozzárendelési feladatok	4
3.2	A matematikai háttér és modell	5
3.2.1	Első eset: $n = m$	6
3.2.2	Második eset: $n < m$	6
3.2.3	Harmadik eset: $n > m$	6
3.3	Python	6
3.4	Tkinter	7
3.5	SciPy	7
4.	Gyakorlati tanulmány (az alkalmazás bemutatása)	7
4.1	Rendszerkövetelmények	7
4.2	A program használata, működése	8
5.	Előrejezlés	10
6.	Tesztelés	10
7.	Következtetések	10
8.	Könyvészet	11

Hozzárendelési feladat modellezése Excel-ben

1. Kivonat

A dolgozatunk egy olyan hozzárendelési problémára segít optimális megoldást találni lineáris programozási módszerrel, ahol n tanuló vizsgázik, m db tétel közül húzhatnak egyet, és mindegyik tanulónak ismerjük az adott tételre való felkészültségét, amelyek jegyben $(1 - 10)$ vannak megadva, n , m és a jegyek a felhasználó által tetszőlegesen módosíthatóak.

A probléma fő kérdései pedig, amelyekre megoldást ad a program, többek között: a legszerencsésebb és legszerencsétlenebb húzások átlaga, illetve a húzási kombinációk elemzése, valamint az átlagokra való előrejelzés, mozgóátlag segítségével.

Erre egy interaktív, felhasználóbarát programot készítettünk Python nyelvben, a Tkinter GUI modul segítségével.

Végül a tesztelésre is nagy hangsúlyt fektettünk, hisz enélkül egyetlen program sem adható át a megrendelőnek.

Kulcs szavak: operációkutatás, hozzárendelési feladat, Python, Tkinter, lineáris programozás, optimalizáció

JEL kód: C44, C61, C67

2. Bevezetés

A kiindulási feladat a következő: Négy hallgató vizsgázik. A tanár kiválaszt 6 tételt, amelyekből a hallgatók véletlenszerűen kiválasztanak egyet-egyet. A hallgatók felkészültségét az egyes tételekből az alábbi táblázat tartalmazza:

1. táblázat – az alap feladatot szemléltető táblázat

Hallgató	Tétel					
	I.	II.	III.	IV.	V.	VI.
A	9	6	9	5	7	9
B	6	7	7	4	5	4
C	9	9	5	7	5	10
D	8	5	8	7	5	14

A legszerencsésebb húzás esetén mennyi lesz a hallgatók osztályzatának átlaga? Legszerencsésebb húzásnak azt tekintjük, ha a hallgatók által elért összeredmény a lehető legnagyobb. Lehetséges-e, hogy minden hallgató azt a tételt húzza, amelyből a legjobban felkészült? A legszerencsétlenebb húzás esetén előfordulhat-e, hogy egyik hallgató sem kap 4-est?

Az 1. táblázat szemlélteti, hogy a probléma felírható egy egyszerű, $n \times m$ -es méretű mátrixként, amely a hozzárendelési feladatok alapja. Dolgozatunk során a hallgatókat betűk helyett (ABC...) helyett $H1 \dots Hn$ sorszámmal jelöltük, míg a tételeket $T1 \dots Tn$ -nel.

Ha a feltett kérdésekre magunktól, számítógépes program nélkül keresnénk a választ, az a mátrix méretétől függően rengeteg időbe és energiába kerülhet, ezért sokkal célszerűbb egy számítógépes szoftvert használni, például az Excelben található Solver modul is alkalmas erre a célra. Viszont, ha az egyén nem jártas annak használatában, vagy nem szeretne időt eltölteni azzal, hogy felírja a feladatot egy Solver által megoldható problémaként, akkor jól jöhet egy olyan segédprogram például, amit mi készítettünk. Emellett az Excel-nél sokkal hatékonyabb és gyorsabb például egy Python nyelven írt program.

3. Elméleti háttér

3.1 Hozzárendelési feladatok

A hozzárendelési feladat (angolul: assignment problem) egy alapvető kombinatorikai optimalizációs feladat. Tekintsük az alábbi problémát, mely elvégzendő feladatok optimális kiosztásáról szól: Adott n számú dolgozó és ugyanennyi elvégzendő feladat. A feladatok mindegyikét el tudja látni bármelyik dolgozó, csak ezt nem ugyanakkora költséggel teszik (költség lehet például az egyes feladatok elvégzéséhez szükséges idő az egyes munkások esetén). Jelölje C a költségmátrixot, ennek az $n \times n$ típusú mátrixnak a c_{ij} eleme megadja, hogy mekkora a j -edik munka elvégzésének a költsége, ha azt az i -edik dolgozó hajtja végre. A feladat célja általában: Osszuk szét a dolgozók között az összes feladatot úgy, hogy minden dolgozó pontosan egy munkát kapjon, és a munkavégzés összköltsége minimális legyen!

2. Táblázat: Együtthatók mátrixa (példa)

Hallgató (i)	Tétel (j)					
	T ₁	T ₂	T ₃	T ₄	T ₅	T ₆
H ₁ (A)	1	0	0	0	0	0
H ₂ (B)	0	1	0	0	0	0
H ₃ (C)	0	0	0	0	0	1
H ₄ (D)	0	0	1	0	0	0

Tehát esetünkben a költségek (C_{ij}) a tanulók jegyét jelentik, hogy milyen szinten tanulták meg az adott tételt, míg az együtthatók mátrixának elemei (X_{ij}) azt mutatják meg, hogy az adott tételt kihúzta-e az adott tanuló. Ezek szorzatainak összege adja a célfüggvényt, ami Excelben lényegében a két táblázat szorzatösszege lesz. A 2. táblázatban például az látható, hogy ebben a megoldásban A hallgató az első tételt, B hallgató a másodikat, C hallgató a hatodikat, és végül a D hallgató a harmadik tételt kellene húzza ahhoz, hogy a legjobb átlagot ériék el, a feltételek teljesítése mellett.

3.2 A matematikai háttér és modell

Általános feladat: n db hallgató vizsgázik. A tanár kiválaszt m db tételt, amelyekből a hallgatók véletlenszerűen választanak 1-1-et. A hallgatók felkészültségét az egyes tételekből az alábbi táblázat tartalmazza:

3. táblázat: matematikai modell

Hallgatók	Tételek					
	T_1	T_2	...	T_j	...	T_m
H_1	C_{11}	C_{12}	...	C_{1j}	...	C_{1m}
H_2	C_{21}	C_{22}	...	C_{2j}	...	C_{2m}
...
H_i	C_{i1}	C_{i2}	...	C_{ij}	...	C_{im}
...
H_n	C_{n1}	C_{n2}	...	C_{nj}	...	C_{nm}

A következő matematikai feltételek segítségével oldottuk meg a feladatot:

- A jegyek mátrixának (C) bármely eleme 4 és 10 közötti értéket vehet fel. Bármely sor- és oszlopindex (i, j) 1 és n , illetve 1 és m közötti természetes szám lehet

$$\forall i, j \Rightarrow C_{ij} \in [4, 10], C_{ij} \in \mathbb{N}$$

$$i \in [1, n]; j \in [1, m]; i, j \in \mathbb{N}$$

- A segédmátrix bármely eleme 0 vagy 1 lehet:

$$x_{ij} = \begin{cases} 0, & \text{ha } H_i \text{ hallgató nem a } T_j \text{ tételt húzza} \\ 1, & \text{ha } H_i \text{ hallgató a } T_j \text{ tételt húzza} \end{cases}$$

- A célfüggvény (z) X és C mátrix szorzatösszege lesz

$$z = \sum_{i=1}^n \sum_{j=1}^m x_{ij} * C_{ij}$$

- A legszerencsésebb húzásnál a cél: $z \rightarrow \max$, Legszerencsétlenebb húzásnál: $z \rightarrow \min$

A továbbiakban a megszorításokat attól függően határoztuk meg, hogy mennyi tanuló (n) és tétel (m) van.

3.2.1 Első eset: $n = m$

Mindegyik hallgató 1 tételt húz és mindegyik tétel egyszer van kihúzva.

- Feltétel a hallgatókra: $\sum_{j=1}^m x_{ij} = 1$, ahol $i = \{1, 2, \dots, n\}$
- Feltétel a tételekre: $\sum_{i=1}^n x_{ij} = 1$, ahol $j = \{1, 2, \dots, m\}$

3.2.2 Második eset: $n < m$

Mindegyik hallgató egy tételt húz, mindegyik tétel legfeljebb egyszer van kihúzva.

- Feltétel a hallgatókra: $\sum_{j=1}^m x_{ij} = 1$, ahol $i = \{1, 2, \dots, n\}$
- Feltétel a tételekre: $\sum_{i=1}^n x_{ij} \leq 1$, ahol $j = \{1, 2, \dots, m\}$

3.2.3 Harmadik eset: $n > m$

Mindegyik hallgató 1 tételt húz, mindegyik tétel legalább $\lfloor \frac{n}{m} \rfloor$ -szer van kihúzva

- Feltétel a hallgatókra: $\sum_{j=1}^m x_{ij} = 1$, ahol $i = \{1, 2, \dots, n\}$
- Feltétel a tételekre: $\sum_{i=1}^n x_{ij} \geq \lfloor \frac{n}{m} \rfloor$, ahol $j = \{1, 2, \dots, m\}$

3.3 Python

“A Python egy általános célú, nagyon magas szintű programozási nyelv, melyet Guido van Rossum holland programozó kezdett el fejleszteni 1989 végén, majd hozott nyilvánosságra 1991-ben. A nyelv tervezési filozófiája az olvashatóságot és a programozói munka megkönnyítését helyezi előtérbe a futási sebességgel szemben. Például a behúzások szintaktikailag is fontosak.

A Python többek között a funkcionális, az objektumorientált, az aspektusorientált az imperatív és a procedurális programozási paradigmákat támogatja. Dinamikus típusokat és automatikus

memóriakezelést használ, ilyen szempontból hasonlít a Scheme, Perl és Ruby nyelvekhez, emellett szigorú típusrendszerrel rendelkezik. Erőssége a gazdag szabványos programkönyvtár.

A Python úgynevezett interpreteres (értelmezős) nyelv, ami azt jelenti, hogy nincs különválasztva a forrás- és tárgykód, a megírt program máris futtatható, ha rendelkezünk a Python értelmezővel. A Python értelmezőt számos géptípusra és operációs rendszerre elkészítették, továbbá számtalan kiegészítő könyvtár készült hozzá, így rendkívül széles körben használhatóvá vált.

Az egyik legnépszerűbb programozási nyelv. Nyitott, közösségalapú fejlesztési modellt mutat fel, amit a közhasznú Python Software Foundation felügyel, ami a nyelv definícióját a CPython referenciaimplementációval gondozza.”

(forrás: Wikipédia [1])

3.4 Tkinter

A Python nyelvhez többféle GUI (Graphical User Interface) eszköztár is tartozik, ebből a legnépszerűbb a Tkinter. „A grafikus könyvtár (library) olyan szoftvereszközkészlet, amely különböző GUI-elemek funkcionalitását meghatározó osztályok gyűjteményét tartalmazza. Ezek a grafikus könyvtárak általában C/C++ nyelven íródnak. Sokukat importálható modulok formájában átültették Pythonra.”

forrás: Salamon Júlia [2]

A tkinter modul importálása után máris használhatjuk a könyvtárat, és parancsokon keresztül hozzáadhatjuk, manipulálhatjuk a programhoz tartozó grafikus komponenseket.

3.5 SciPy

A SciPy egy ingyenes és nyílt forráskódú könyvtár Python nyelvhez, tudományos, illetve mérnöki számítások elvégzésének megkönnyítése céljából.

Többek közt optimalizáció, lineáris algebra, integráció, interpoláció, speciális függvények, FFT, képfeldolgozás modulokat is tartalmaz. Egyébként nemzetközi konferenciákon egyeztetnek a fejlesztők és felhasználók a modul fejlesztése kapcsán.

Projectünk során a SciPy modul linprog utasítását használtuk a lineáris programozási feladatok megoldáshoz.

4. Gyakorlati tanulmány (az alkalmazás bemutatása)

A program vezérlését igyekeztünk a lehető legegyszerűbben kialakítani, hogy a használata egyértelmű legyen a felhasználóknak, tehát csak a legfontosabb adatbeviteli mezőket és grafikus

komponenseket helyeztük el benne. A továbbiakban ismertetjük a legfontosabb tudnivalókat a program működéséről, leginkább felhasználói szempontból, minimális programozási háttér mellett.

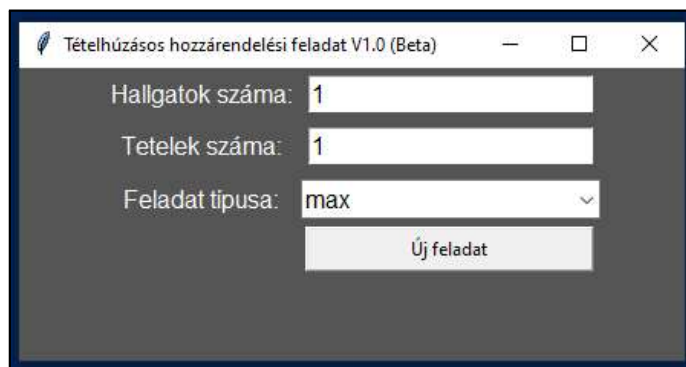
4.1 Rendszerkövetelmények

A program elméletileg minden olyan eszközön tud futni, amely rendelkezik Python 3.X értelmezővel, operációsrendszertől függetlenül. Fontos, hogy a SciPy modul telepítve legyen, különben nem indul el az alkalmazás. Emellett az ajánlott felbontás legalább 1366 x 768 annak érdekében, hogy a program grafikus vezérlőelemei megfelelően tudjanak megjelenni. (a tesztelés részben olvasható, hogy mi a probléma a kisebb kijelzőkkel).

4.2 A program használata, működése

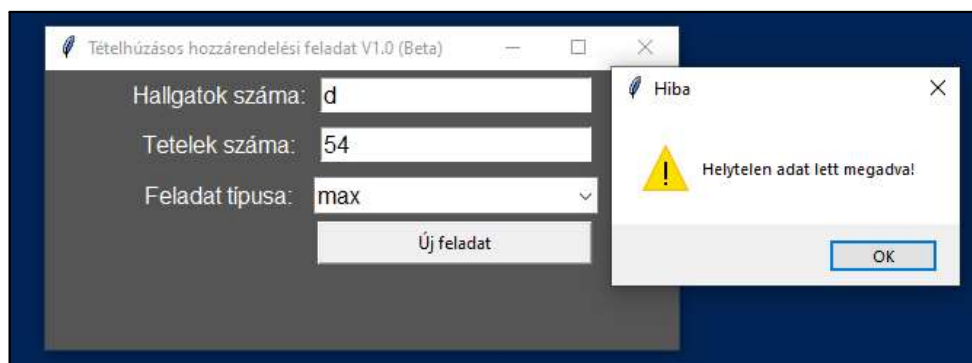
Mivel a Python egy értelmezős nyelv, ezért a programokat alapból parancssor (vagy terminál) segítségével lehet futtatni, ahol ki kell adni a *Python programfájl.py* parancsot. Az egyszerűbb indítás miatt készítettünk a program mappájában egy kötegelt állományt (progam.bat), amely tartalmazza a fentebb említett parancsot, csupán rá kell kattintani.

Az alkalmazás betöltése után megjelenik a következő felület:

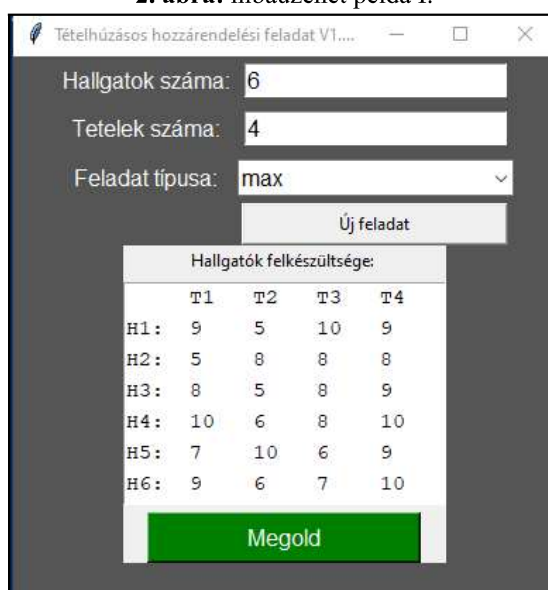


1. ábra: főmenü (első szakasz)

Itt kell megadni a hallgatók és a tételek számát, valamint itt van lehetőség a legrosszabb vagy legjobb húzás közül választani (a max a legjobb, a min a legrosszabb húzást jelenti). Ha megfelelően adtuk meg az adatokat (1 és 99 közötti természetes számok, ellenkező esetben hibaüzenetet ad a program), az Új feladat gomb megnyomása után megjelenik egy formázott szöveges mező, ahol a hallgatók egyes tételekből való felkészültségét tudjuk megadni. Fontos, hogy a jegyek között legalább egy szóköz legyen, mint elválasztó karakter.



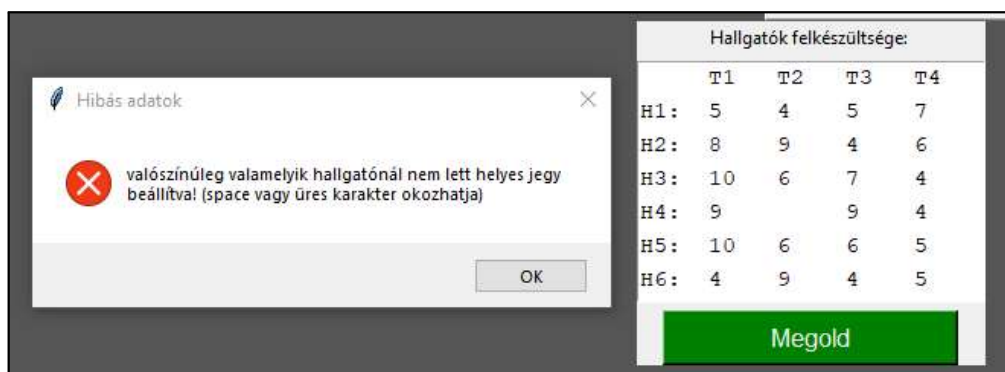
2. ábra: hibaüzenet példa I.



3. ábra: főmenü (második szakasz)

Látszik, hogy az Új feladat gomb automatikusan feltölti a táblázatot (költségmátrixot) véletlenszerű, 4 és 10 közötti jegyekkel. Ezeket tudjuk módosítani természetesen.

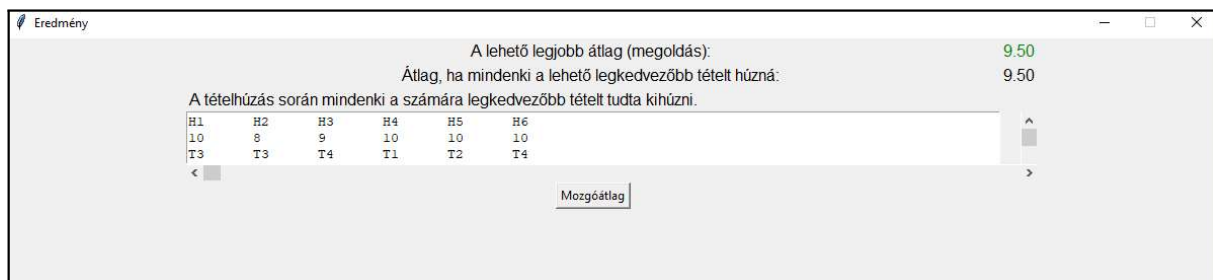
Egyébként a szöveges mező 20x25-ös táblázatig akkora helyet vesz fel, amekkora szükséges, hogy kiferjen az összes sor és oszlop. Ennél hallgató sor vagy tétel esetén megjelenik egy vertikális, illetve egy horizontális görgetősáv. Ha nem adunk meg valahol jegyet, vagy nem számot adunk meg a táblázatban, a program figyelmeztet, ha a zöld gombra kattintunk:



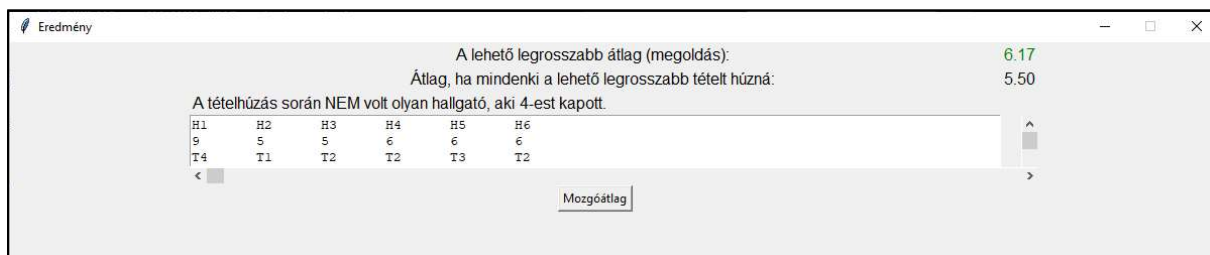
4. ábra: Hibaüzenet példa II.

Ha helyes adatok szerepelnek a táblázatban, a Megold feliratú gombra kattintva felugrik egy új ablak az eredménnyel, ahol zöld betűszínnel látszódik az, hogy a lineáris programozási feladatnak a legoptimálisabb megoldása (a 3.2-es fejezetben kifejtett modell) szerint mennyi lenne a vizsgázó csoport átlaga. Alatta pedig a korlátozási feltételek nélküli átlag (sormaximumok vagy sorminimumok átlaga) is látszik. Minimalizálás esetén leírja, hogy kapott-e valaki 4-est, valamint maximalizálás esetén azt, hogy minden hallgató ki tudta-e húzni a számára legkedvezőbb tételt.

Ugyanakkor egy táblázatban látható, hogy melyik hallgató melyik tételt kell kihúzza a megoldás szerint. A következő két ábra a 3. ábrán látható feladat megoldásait szemlélteti (sorrend: hallgató, jegy, tétel sorszáma).



5. ábra: Megoldás a legjobb húzás esetén



6. ábra: Megoldás a legrosszabb húzás esetén

Fontos tudni, hogyha ugyanarra a feladatra (ugyanaz a költségmátrix mellett) szeretnénk mindkét (min/max) megoldást megkapni, elég, ha a legördülő listában átállítjuk a min/max értéket és úgy

nyomjuk meg a zöld gombot. Ha megnyomjuk előtte az új feladat gombot, akkor sajnos felülírja az előző értékeket a táblázatban.

5. Előrejelzés (mozgóátlag)

6. Tesztelés

Mivel a Tkinter méretezési parancsok nem feltétlen relatív mértékegységeket használnak, programot a következő felbontásokon teszteltük:

- 800x600: A beviteli táblázat legfeljebb 16x16-os méretig fér el a képernyőn. Több hallgató esetén a Megold gomb nem fér el, és több tétel nem látjuk az összes oszlopot.
- 1024x768: Itt már bármennyi hallgató (sor) is elfér, mivel 20 felett mindenképpen megjelenik a görgetősáv, a gomb látszódik. Viszont maximum 20 tétel fér el, mivel 25 darab tétel arányosan nő a szövegdoboz szélessége.
- 1366x768: Ennél a felbontásnál minden tökéletesen látszódik és működik a felhasználói felületen.
- 1920x1080: Ugyanúgy viselkedik, mint az 1366x768-as felbontásnál

7. Következtetések

A programot sikerült úgy elkészítenünk, hogy minden – a bevezető részben levő - kérdést megválaszoljon, és az alkalmazás rendeltetésszerű használat mellett stabilan működik.

Pár kattintással megoldhat a felhasználó egy tételhúzásos hozzárendelési feladatot, mindezt egy átlátható, könnyen kezelhető program segítségével, időt és energiát megspórolva.

A program készítése során sokat tanultunk Python nyelvből. A Tkinter számunkra újdonság volt, de hamar ráérezünk annak használatára.

Egyértelmű számunkra, hogy sokkal hatékonyabban működik a program a jelenlegi formájában, mint az előző dolgozat során készített VBA – Excel verzió.

8. Könyvészet

[1] Wikipédia, Python (programozási nyelv), letöltés ideje: 2023.04.27.

[https://hu.wikipedia.org/wiki/Python_\(programoz%C3%A1si_nyelv\)](https://hu.wikipedia.org/wiki/Python_(programoz%C3%A1si_nyelv))

[2] Salamon Júlia, Operációkutatás 9. előadás

http://web.csik.sapientia.ro/moodle/pluginfile.php/4594/mod_resource/content/1/Eloadas9.docx?forcedownload=1