

SZAKDOLGOZAT FELADAT

**Tokovics Dávid Tamás**

Mérnökinformatikus hallgató részére

Hálózati folyam feladatok automatikus generálása

A Számítástudomány alapjai és a Bevezetés a számítástudományba tárgyak anyaga a Hálózati folyamok elmélete. Minden félévben szükségünk van több konkrét példára, amit a hallgatóknak kell megoldani. Nem könnyű feladat rendszeresen előállítani ilyen példákat, hiszen nem lehet se túl könnyű, se túl nehéz, se túl kicsi, se túl nagy.

A hallgató feladata egy olyan alkalmazás létrehozása, ami ilyen példákat automatikusan előállít. A feladat első nehézsége annak definiálása, hogy mikor ,,jó'' egy ilyen példa.

A hallgató feladatának egy olyan szoftver létrehozása, ami tudja a következőket:

* Adott típusú gráfokból egy véletlen hálózat generálása.
* Egy véletlenül kiválasztott majdani minimális vágás generálása.
* Az élekhez véletlen kapacitások rendelése úgy, hogy a minimális vágás éppen a kiválasztott legyen.
* A folyam feladat megoldása.
* A feladat és megoldás grafikus megjelenítése.
* A feladat nehézségének megfelelően egy mérőszámot is rendeljen a feladathoz egy adott skálán.
* A feladat exportálható legyen LaTeX TikZ formátumba.

**Tanszéki konzulens:** Dr. Katona Gyula egyetemi docens

Budapest, 2022. október 08.

Dr. Katona Gyula   
*egyetemi docens*

*tanszékvezető*



Budapesti Műszaki és Gazdaságtudományi Egyetem

Villamosmérnöki és Informatikai Kar

Számítástudományi és Információelméleti Tanszék

Tokovics Dávid

Hálózati folyam feladatok automatikus generálása

Konzulens

2022. december 9.

Tartalomjegyzék

[Összefoglaló 1](#_Toc120961095)

[Abstract 2](#_Toc120961096)

[Bevezetés 3](#_Toc120961097)

[Hálózati folyam feladatok 5](#_Toc120961098)

[2.1 Hálózatok 5](#_Toc120961099)

[2.2 Folyamok 5](#_Toc120961100)

[2.3 Maximális folyam 5](#_Toc120961101)

[2.4 Minimális vágás 5](#_Toc120961102)

[Hálózatok generálása 6](#_Toc120961103)

[3.1 Csúcsok elrendezése 6](#_Toc120961104)

[3.2 Keret éleinek behúzása 8](#_Toc120961105)

[3.3 További élek hozzáadása 10](#_Toc120961106)

[3.3.1 s-ből vagy t-be mutató élek 10](#_Toc120961107)

[3.3.2 Azonos oszlopba mutató élek 10](#_Toc120961108)

[3.3.3 Eggyel előre mutató élek 11](#_Toc120961109)

[3.3.4 Kettővel előre mutató élek 12](#_Toc120961110)

[3.3.5 Eggyel vissza mutató élek 12](#_Toc120961111)

[3.4 Folyam létrehozása 15](#_Toc120961112)

[3.4.1 Utak keresése s-ből t-be 15](#_Toc120961113)

[3.4.2 Utak keresése visszaéllel 16](#_Toc120961114)

[3.4.3 Keletkezett folyamok 16](#_Toc120961115)

[3.5 Folyam átalakítása maximális folyam feladattá 17](#_Toc120961116)

[3.5.1 Vágás sorsolása 17](#_Toc120961117)

[3.5.2 Élek növelése a vágás éleinek kivételével 17](#_Toc120961118)

[3.5.3 Keletkezett feladatok 18](#_Toc120961119)

[Feladat nehézsége 20](#_Toc120961120)

[4.1 Élekkel kapcsolatos nehézségek 20](#_Toc120961121)

[4.1.1 Élek száma 20](#_Toc120961122)

[4.1.2 Nullélek száma 20](#_Toc120961123)

[4.1.3 Hosszú élek száma 20](#_Toc120961124)

[4.1.4 Visszafele mutató élek száma 20](#_Toc120961125)

[4.2 Csúcsok száma 20](#_Toc120961126)

[4.3 Vágáserősség 20](#_Toc120961127)

[4.4 Szórás 20](#_Toc120961128)

[4.5 Teljes képlet 20](#_Toc120961129)

[A program bemenetei 22](#_Toc120961130)

[5.1 Grafikus felület kezdőoldala 22](#_Toc120961131)

[5.2 Hálózat generálása paraméterek megadásával 22](#_Toc120961132)

[5.3 Hálózat generálása fájlból 23](#_Toc120961133)

[5.3.1 Hálózat generálása random gráffal 24](#_Toc120961134)

[5.3.2 Hálózat generálása előre megadott gráffal és vágással 25](#_Toc120961135)

[A program kimenetei 27](#_Toc120961136)

[6.1 Grafikus megjelenítés 27](#_Toc120961137)

[6.1.1 Feladat 27](#_Toc120961138)

[6.1.2 Megoldás 28](#_Toc120961139)

[6.1.3 Újragenerálás 31](#_Toc120961140)

[6.2 Txt fájl 31](#_Toc120961141)

[6.3 LaTeX fájl 32](#_Toc120961142)

[6.3.1 Program által generált LaTeX fájl 32](#_Toc120961143)

[6.3.2 Javított LaTeX fájl 34](#_Toc120961144)

[Összegzés 37](#_Toc120961145)

[Irodalomjegyzék 38](#_Toc120961146)

Hallgatói nyilatkozat

Alulírott **Tokovics Dávid**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

A képen szöveg látható

Automatikusan generált leírásKelt: Budapest, 2022. 12. 03.

...…………………………………………….

Tokovics Dávid

# Összefoglaló

A Bevezetés a számítástudományba és a Számítástudomány alapjai tárgyakban a Hálózati folyamok témakörében szükség van ellenőrizni a hallgatók tudását, ehhez pedig feladatokat kell készíteni. Ez azonban nem olyan egyszerű feladat, ezért nagy segítség egy olyan program, ami ezt a munkát elvégzi az oktatók helyett. Erről szól a dolgozat témája, amely egy program, ami hálózati folyam feladatokat generál automatikusan, bizonyos paraméterek megadásával.

A paraméterek egyrészt a gráf tulajdonságai, mégpedig, hogy hány sorból és hány csúcsból áll. Emellett meg kell adni egy feladatnehézséget is, és ezen paraméterekből alkotja meg a megfelelő hálózatot a program, amelyeket bevihetünk a grafikus felületen keresztül, vagy fájlból. Utóbbi esetben akár egy konkrét gráfra is futtathatjuk.

A feladat nehézségekor figyelembe vesszük az élek számát, a hosszú élek számát, a visszafele mutató élek számát, azt, hogy a hálózatból keletkező maximális folyamban hány él kapacitása 0, illetve a minimális vágás erősségét, és az élek kapacitásának szórását, és a csúcsok számát is.

A program kimenete 3 féle lehet, egy futtatáskor mindhármat megkapjuk. Van egy JFrame alapú grafikus megjelenítés, ahol a megoldás megjelenítésére is van lehetőség, valamint újra is lehet generálni a feladatot, ha számunkra nem megfelelő az eredmény. Emellett egy txt fájlt is kapunk eredményként, ahol szöveges formában van leírva a feladat megjelenítése. A harmadik, amit kimenetként kapunk az egy LaTeX TikZ formátumú forráskód.

# Abstract

jg

**1. fejezet**

# Bevezetés

Több szakon a BME-n és más egyetemeken is vannak tárgyak, amelyek a gráfok témakörére épülnek. Általában ennek a területnek része a hálózati folyamok elmélete is, és az ehhez kapcsolódó maximális folyam és minimális vágás feladatok. A dolgozat célja egy olyan program létrehozása volt, amely képes hálózati folyam feladatok automatikus generálására. A fő motiváció erre az volt, hogy ezt a programot aztán lehessen hasznosítani a Számítástudomány alapjai és a Bevezetés a számítástudományba tárgyakban, ugyanis ezen tárgyak témakörei közé tartoznak a hálózati folyamok. Szükség van ebben a témakörben a hallgatók tudásának ellenőrzésére, és ennek érdekében feladatok készítésére.

A feladatok létrehozása azonban nem olyan egyszerű dolog, mivel nem egyértelműen megállapítható egy létrehozott feladatnak a nehézsége. Nyilván nem szeretnénk például egy zárthelyi dolgozatban, ha a hallgató nagyon hamar megoldaná a feladatot, és azt sem, ha egy megoldhatatlannak tűnő feladattal találkozna zárthelyi írás közben. Viszont az sem ideális, ha a megfelelő feladat kidolgozására az oktatónak a kelleténél több ideje menne rá, hogy az szerinte pont jó nehézségű legyen, egyébként is a feladat létrehozásakor nehéz ellenőrizni, hogy valóban olyan nehéz-e, mint ahogy mi azt gondoljuk. Tehát az igazi kihívás az, hogy egy számunkra megfelelő nehézségű feladatot kreáljunk. Ezért jól jönne egy program, ami ezt megteszi helyettünk, vagyis egy általunk megadott nehézséggel hoz létre egy hálózati folyam feladatot, amelyet aztán a gyakorlatokon és a zárthelyi dolgozatokban fel lehet használni, ezzel pedig az oktatóknak rengeteg időt spórolva.

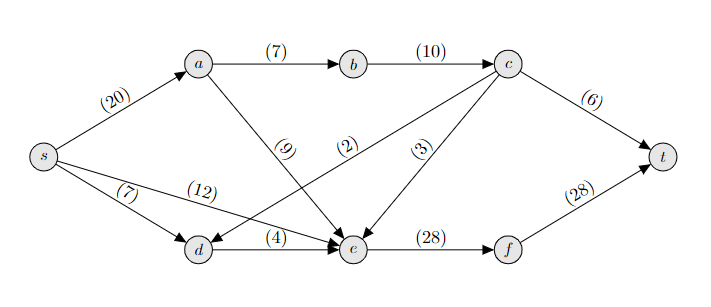
A dolgozat további részében szükség lesz először is tisztázni, hogy mik is azok a hálózatok, folyamok, hogyan lehet belőlük feladatot csinálni. Aztán szükség van annak az elemzésére, hogy mitől is lesz nehéz egy ilyen feladat, és hogyan tudunk a programunkkal egy kívánt nehézségű feladatot generálni. Ezután pedig arról is szó kell essen, hogy milyen módon tudjuk azt a programnak megmondani, hogy hogyan is nézzen ki a feladatunk, ugyanis nem csak egy féleképpen tudunk egy hálózatot generálni, lehet például több sora is az egyiknek, mint egy másiknak. Illetve ezen kívül a kívánt nehézség értékét is meg kell adnunk valamilyen módon a programunknak, amiből az aztán feladatot tud generálni. Végül pedig azt is tisztáznunk kell, hogy az eredményt mi milyen módon kapjuk meg, illetve azt, hogy tudjuk a későbbiekben, például egy zárthelyi dolgozatban felhasználni.

**2. fejezet**

# Hálózati folyam feladatok

## 2.1 Hálózatok

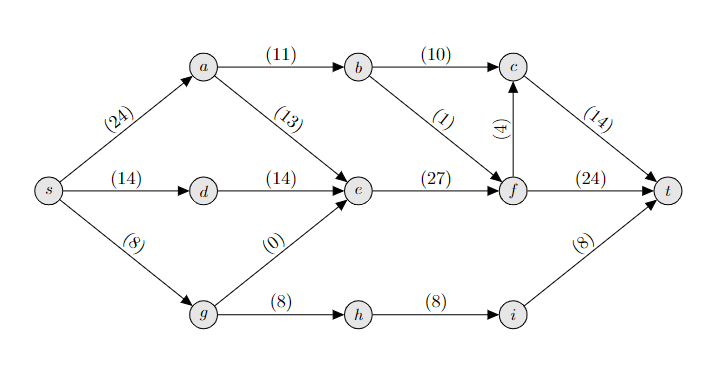
afas



**2.1. ábra: Hálózat**

## 2.2 Folyamok

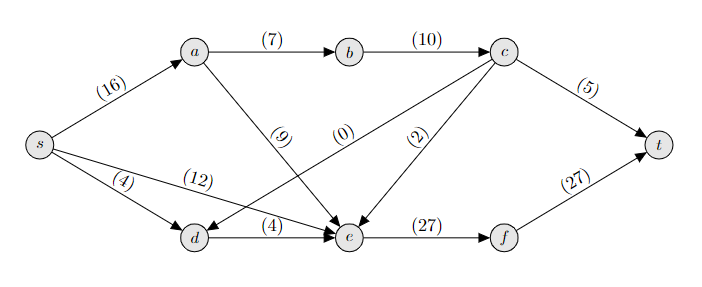
afas



**2.2. ábra: Folyam**

## 2.3 Maximális folyam

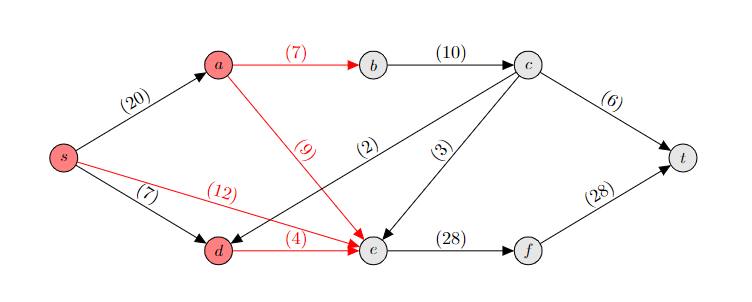
afas



**2.3. ábra: Maximális folyam**

## 2.4 Minimális vágás

afas



**2.4. ábra: Minimális vágás**

3. fejezet

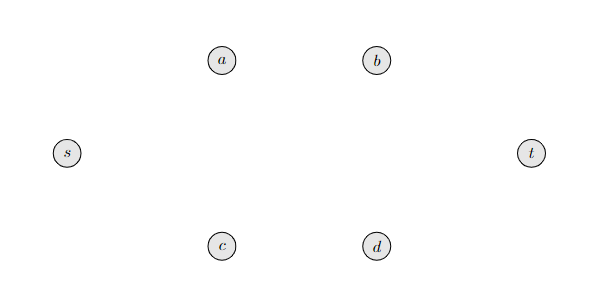
# Hálózatok generálása

A feladatokat generáló program csak a megadott paraméterek alapján gyárt hálózatokat, amelyeket aztán kiértékel nehézség szerint, és ezalapján eldönti, hogy elveti vagy megtartja azt. Később a megtartott hálózatot is ugyanúgy elvetheti, ha talál egy jobbat, mivel rengeteget generál, és mindegyik nehézségét összehasonlítja a kívánt értékkel, így, ha egy újonnan létrehozott hálózathoz tartozó nehézség közelebb áll a kívánt feladatnehézséghez, akkor azt fogjuk megtartani, és az eddig megtartott hálózatot elvetni. Ebben a fejezetben megnézzük, hogy hogyan is állítjuk elő ezeket a hálózatokat. Először legyártjuk a gráfot részekre lebontva, majd utána adunk kapacitásokat az éleknek, így hálózatot létrehozva belőle, amely, ha megfelelő nehézségűre sikerült, akkor később maga a feladat, mint a program eredménye is lehet belőle.

## 3.1 Csúcsok elrendezése

Első lépés az, hogy a megadott sor- és csúcsszámok alapján megalkotjuk a csúcsok elrendezését. A megadott paraméterek alapján mindig csak egyféle elrendezés lehetséges, ami pedig a következő: Ha páros számú sorral kell generálnunk, akkor valójában annak a gráfnak is páratlan számú sora lesz, viszont a középső sorban csak az s és t szerepel a bal és jobb oldalakon a sor szélén. A többi sorban pedig a megadott számú mínusz 2 – ami az s és a t – csúcs található elosztva úgy, hogy minden, a megadott számú sorban azonos mennyiségű legyen belőlük. Ha páratlan a megadott sorok száma, akkor a középső sorban a bal és a jobb szélen található s és t, valamint ebben a sorban 2-vel több a csúcsok száma, mint a többiben, egyébként pedig ebben az esetben is ugyanúgy a megadott számú sorban elosztjuk a megadott számú csúcsot úgy, hogy a sorokban azonos mennyiség legyen.

A legkisebb gráf, amit így alkothatunk az 2 sorból és 6 csúcsból áll, (ezt 2×2-es gráfként írjuk le) ez látható a 3.1. ábrán.



**3.1. ábra: 2×2-es csúcselrendezés**

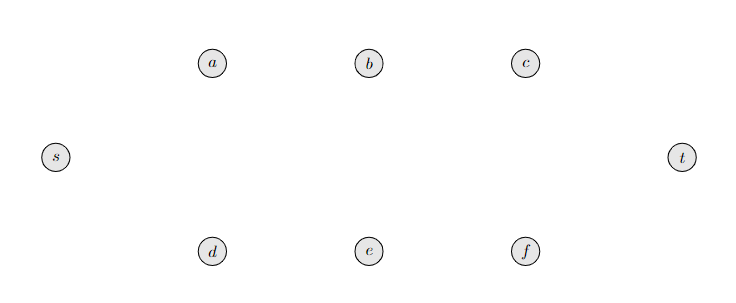
A legnagyobb gráf pedig, amit így létrehozhatunk az 5 soros, és 27 csúcsa van (5×5-ös), ez a 3.2. ábrán látható.



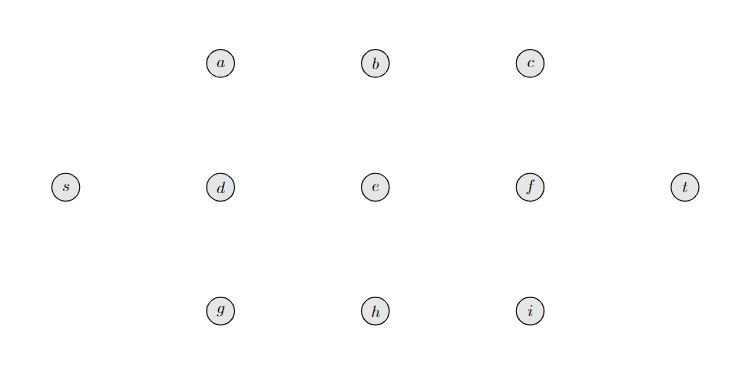
**3.2. ábra: 5×5-ös csúcselrendezés**

A két érték között sor- és oszlopszámban is bármilyen értékkel hozhatunk létre gráfot, vagyis a 2 és 5 mellett lehet 3 vagy 4 soros is, ugyanígy a 2 és 5 mellett állhat 3 vagy 4 oszlopból is, ahol az oszlopszámításban nem számoljuk bele az s és t oszlopait, csak a többit.

A fenti két csúcselrendezést ritkán fogjuk használni, a két leggyakoribb gráftípus a 2 csúcsból és 8 csúcsból álló 2×3-as, valamint a 3 sorból és 11 csúcsból álló 3×3-as lesz, amelyeket a 3.3. ábra és 3.4. ábra tartalmazza.



**3.3. ábra: 2×3-as csúcselrendezés**



**3.4. ábra: 3×3-as csúcselrendezés**

## 3.2 Keret éleinek behúzása

Ha megalkottuk a keretet, akkor a következő feladat, hogy éleket húzzunk be a gráfba. Ezt nem teljesen randomizált módon tesszük, ugyanis lesznek élek, amelyeket biztosan bele kell tegyünk minden egyes gráfba, és lesznek olyanok is, amiket viszont semmiképp nem húzunk be. A maradékot pedig az él típusától függően bizonyos valószínűségekkel húzzuk majd be.

Először nézzük meg azokat az éleket, amelyek biztosan részei lesznek a gráfnak. Az egyik ilyen él az s-ből minden sor első csúcsába mutató él. Szintén mindenképp behúzzuk minden csúcs utolsó sorából a t-be mutató éleket. Ezen kívül a harmadik éltípus, amiből az összeset biztosan hozzáadjuk a gráfhoz, azok a sorban a csúcsból a tőle jobbra található csúcsba húzott élek, tehát, hogy minden sorban balról jobbra a sor elejétől a végéig vezessen egy út. Így egy olyan gráfot kapunk, ahol s-ből t-be minden soron keresztül vezet út.

Ezek az élek tehát minden hálózati folyam feladatban szerepelni fognak. Így lesz a 2×3-as és 3×3-as csúcselrendezésből az alábbi két gráf, amit a 3.5. ábra és 3.6. ábra tartalmaz, ezekhez fogunk még behúzni további éleket.

A képen kard, lámpa látható

Automatikusan generált leírás

**3.5. ábra: Keret a 2×3-as csúcselrendezésben**

A képen kard, különböző, lámpa látható

Automatikusan generált leírás

**3.6. ábra: Keret a 3×3-as csúcselrendezésben**

## 3.3 További élek hozzáadása

Ha létrehoztuk a gráfot a megfelelő csúcselrendezéssel, és behúztuk a keret éleit, akkor jöhet a többi él behúzása is, amely most már bizonyos mértében randomizált módon fog működni, azalapján, hogy milyen fajta élt szeretnénk behúzni. Fontos, hogy két csúcs közé csak egy élt húzunk be, vagyis, ha már hozzá van adva a gráfhoz egy él, akkor ugyanazon két csúcs közé nem húzható be egy a másik irányba mutató él, vagyis a két csúcs közé behúzható két élnek a valószínűsége, hogy része lesz a gráfnak, az egymástól nem független. Fontos még azt is megjegyezni, hogy azokban a gráfokban, ahol páros számú sor van, (de valójában ugyanúgy páratlan ott is) azoknál az s és t sora feletti, illetve alatti sorok szomszédosnak számítanak a továbbiakban.

### 3.3.1 s-ből vagy t-be mutató élek

Ugyan az s-ből, illetve a t-be húztunk már be éleket, viszont van még pár, amit még hozzáadhatunk a gráfhoz. Az ilyenek a szomszédos, vagyis a tőlük eggyel feljebb vagy lejjebb található sorokban az s-ből a sor második csúcsába húzott, vagy a t-be a sor utolsó előtti csúcsából húzott élek.

A képen kard, modellt állás, különböző látható

Automatikusan generált leírás

**3.7. ábra: s-ből vagy t-be mutató élek**

Mindegyik ilyen fajta élre 1⁄3 az esély, hogy bekerül a gráfba. Mivel minden gráfban pontosan 4 ilyen fajta behúzható él van, ezért leggyakrabban 1 ilyen él lesz behúzva.

### 3.3.2 Azonos oszlopba mutató élek

A következő kategória azok az élek, amelyek valamelyik sorban egy csúcsból az eggyel alatta vagy felette lévő sorban, vele egy oszlopban található csúcsba húzott élek.

A képen kard látható

Automatikusan generált leírás

**3.8. ábra: Azonos oszlopba mutató élek**

Minden ilyen élt 1/4 eséllyel húzunk be, viszont 2 csúcs között 1/2 eséllyel lesz behúzva ilyen fajta él, csak azon belül 1/2 eséllyel az egyik, 1/2 eséllyel pedig a másik irányba.

### 3.3.3 Eggyel előre mutató élek

A következő fajta élek a valamelyik sorban egy csúcsból az eggyel alatta vagy felette lévő sorban, vele eggyel jobbra lévő oszlopban található csúcsba húzott élek.

A képen kard, különböző látható

Automatikusan generált leírás

**3.9. ábra: Eggyel előre mutató élek**

Az ilyen fajta éleket 1/3 eséllyel adjuk hozzá a gráfhoz, ebből egy kicsivel kevesebb lehetőség van, mint az előző kategóriából, de nagyobb esély van rá, hogy behúzzuk, így nagyjából ugyanannyi lesz a gráfban a kétfajtából.

### 3.3.4 Kettővel előre mutató élek

A következő kategória azok az élek, amelyek egy csúcsból az eggyel alatta vagy felette lévő sorban, vele kettővel jobbra lévő oszlopban található csúcsba húzott élek.

A képen kard látható

Automatikusan generált leírás

**3.10. ábra: Kettővel előre mutató élek**

Ebből a fajtából már nem szeretnénk olyan sokat belerakni, amúgy is kevesebb lehetőség van ilyen élek behúzására, és ezt 1/4 eséllyel tesszük meg.

### 3.3.5 Eggyel vissza mutató élek

A következő fajta élek a valamelyik sorban egy csúcsból az eggyel alatta vagy felette lévő sorban, vele eggyel balra lévő oszlopban található csúcsba húzott élek.

A képen kard, különböző látható

Automatikusan generált leírás

**3.11. ábra: Eggyel vissza mutató élek**

Visszafele mutató élekből kevesebb fog kelleni nekünk, mivel alapvetően s-ből t felé nem akarjuk a megfelelő utakat úgy bonyolítani, hogy túl sok él visszafele mutasson, így egy ilyen él behúzására az esély 1/6. Ezzel nagyjából fele annyi kellene legyen a számuk, mint az eggyel előre mutató éleknek.

**3.3.6 Kettővel vissza mutató élek**

A következő kategória azok az élek, amelyek egy csúcsból az eggyel alatta vagy felette lévő sorban, vele kettővel balra lévő oszlopban található csúcsba húzott élek.

A képen kard látható

Automatikusan generált leírás

**3.12. ábra: Kettővel vissza mutató élek**

Ebből lesz a legkevesebbre szükségünk, mivel se a hosszú élek, se a visszafele mutató élek nem olyan fontosak, hogy annyira sok legyen belőlük. 1/8 eséllyel húzunk be egy ilyet, vagyis ebből is nagyjából fele annyi kellene legyen, mint az előrefelé mutató párjából.

**3.3.7 Egyéb, nem használt éltípusok**

A fentiekben az összes éltípust felsoroltuk, amiket használunk a gráfunkban, és behúzunk a folyam elkészítésekor, viszont kimaradt pár élfajta, amikről nem esett szó, azokat pedig nem fogjuk használni a hálózatok generálásakor. Ilyen él például, ha s-ből egy szomszédos sor harmadik, vagy attól jobbra lévő csúcsába húzunk egy élt. Ugyanígy, ha egy szomszédos sorban egy csúcs nem az utolsó két csúcs egyike, akkor semmiképp nem húzunk be onnan a t-be egy élt. Emellett nem húzunk be az s-be vagy a t-ből visszamutató éleket, vagyis nem lehet a gráfunknak olyan éle, amelynek végpontja s vagy kezdőpontja t.

De nem csak s-sel vagy t-vel kapcsolatos élek vannak tiltólistán, ugyanis például olyan élt sem húzunk be, amely bármelyik oszlopból húzott csúcsból egy tőle legalább 3-mal balra vagy jobbra lévő csúcsba mutat. Emellett semmilyen élt nem húzunk be olyan csúcsok között sem, amelyek nem szomszédos sorokban találhatóak. Illetve azt már említettük, hogy két csúcs közé csak egy él húzható, így például egy sorban lévő két szomszédos csúcs közé sem húzhatunk be újabb élt, de igazából nem húzhatunk be egy sorban lévő két nem szomszédos él közé sem, mivel ezek is fednék egymást más élekkel.

Természetesen ezek a szabályok nem a hálózat definíciójából erednek, nem emiatt nem húzzuk be a felsorolt éleket, hanem kizárólag esztétikai szempontból az átláthatóság érdekében. Ahogy az sem, hogy kizárólag egyenes élekkel dolgozunk. Így is fogunk kapni meglehetősen bonyolult hálózatokat, amelyeket elég nehéz átlátni, de ezekkel a szabályokkal azért ez valamennyire kordában tartható. A lenti ábra tartalmaz pár élt ezekből a „tiltott” élekből.

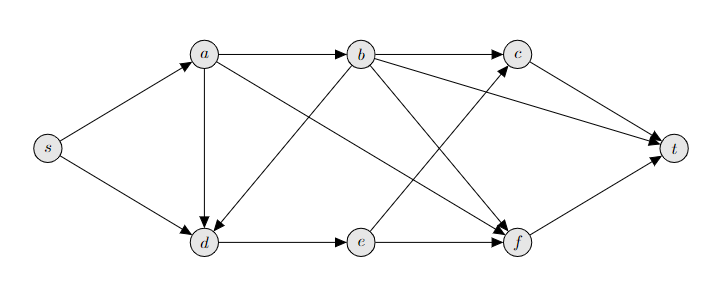
A képen kard látható

Automatikusan generált leírás

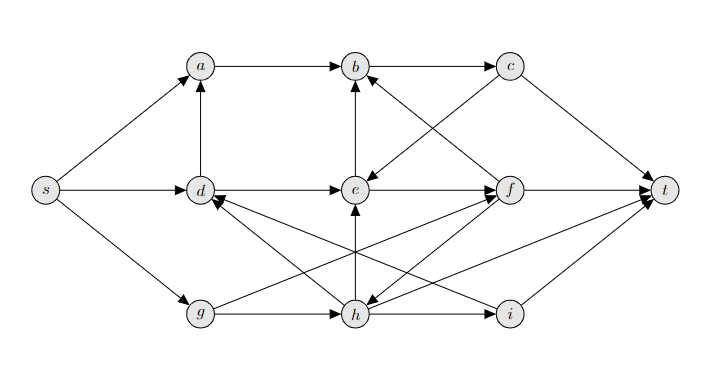
**3.13. ábra: Nem használt éltípusok**

**3.3.8 Az élek hozzáadásával generált gráfok**

asd



**3.14. ábra: Egy generált 2×3-as gráf**



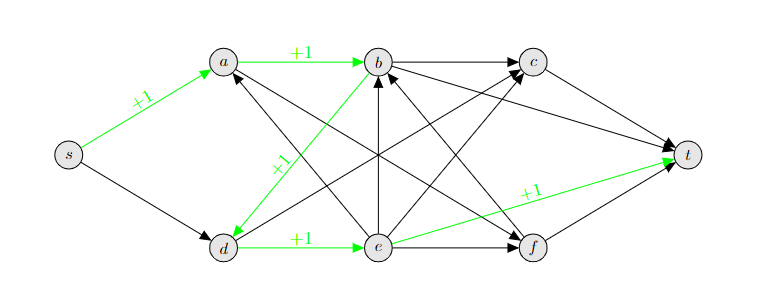
**3.15. ábra: Egy generált 3×3-as gráf**

## 3.4 Folyam létrehozása

asd

### 3.4.1 Utak keresése s-ből t-be

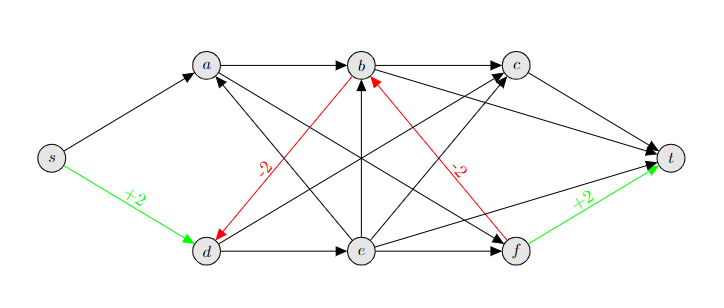
asd



**3.16. ábra: út élkapacitásainak növelése**

### 3.4.2 Utak keresése visszaéllel

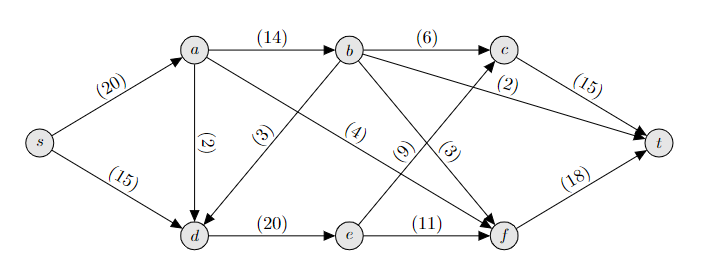
ads



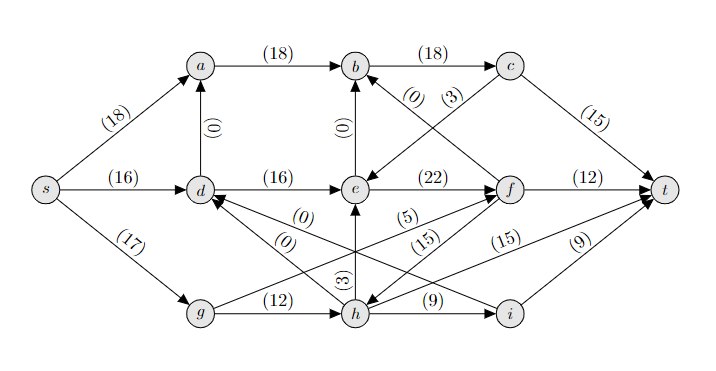
**3.17. ábra: visszaélek élkapacitásának csökkentése az útban**

### 3.4.3 Keletkezett folyamok

asd



**3.18. ábra: Egy generált 2×3-as folyam**



**3.19. ábra: Egy generált 3×3-as folyam**

## 3.5 Folyam átalakítása maximális folyam feladattá

ads

### 3.5.1 Vágás sorsolása

asd

### 3.5.2 Élek növelése a vágás éleinek kivételével

asd

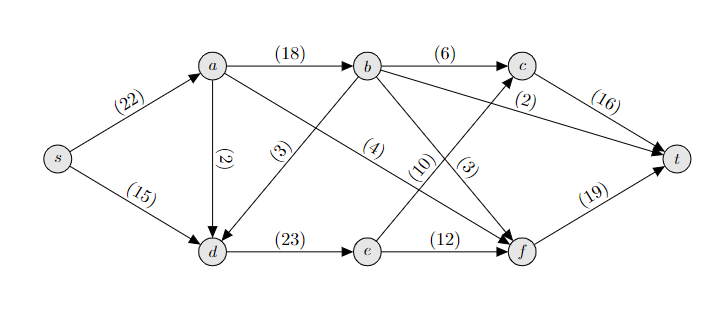
A képen síelés, színes, zászló, lejtő látható

Automatikusan generált leírás

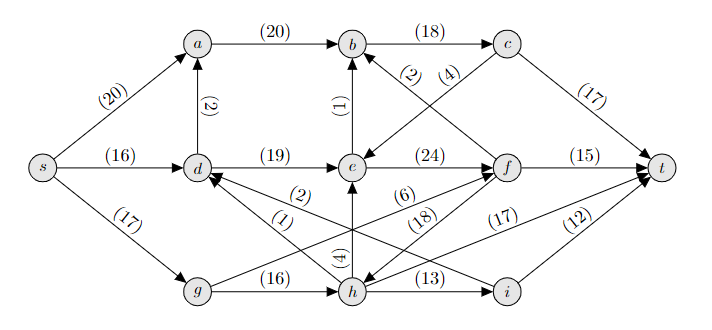
**3.20. ábra: Nem vágásélek kapacitásának növelése**

### 3.5.3 Keletkezett feladatok

asd



**3.21. ábra: Elkészült feladat 2×3-as hálózatra**



**3.22. ábra: Elkészült feladat 3×3-as hálózatra**

4. fejezet

# **Feladat nehézsége**

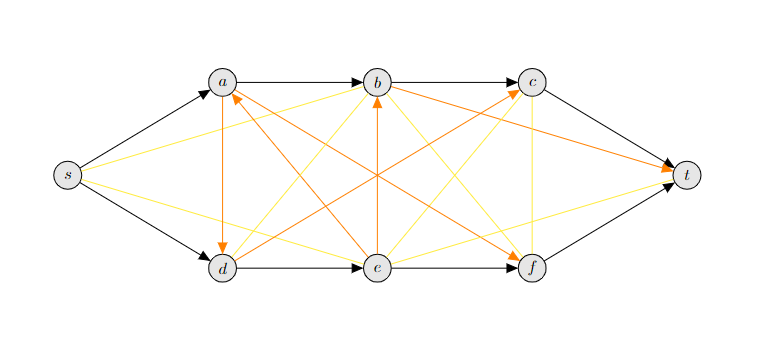
aca

## 4.1 Élekkel kapcsolatos nehézségek

adcaw

### 4.1.1 Élek száma

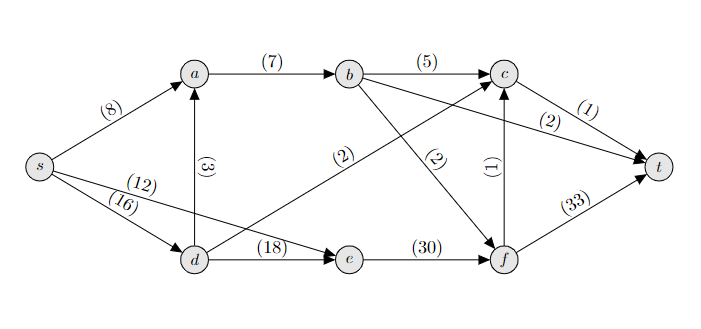
adasd



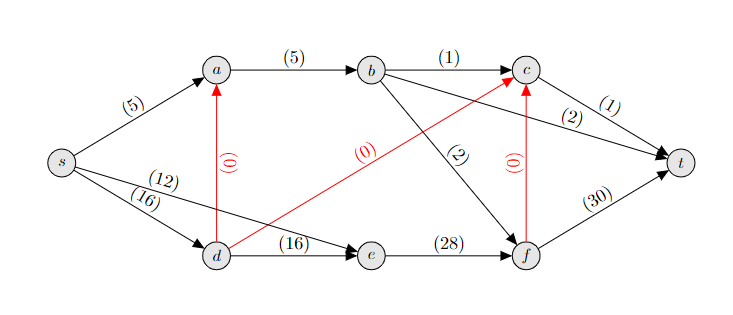
**4.1. ábra:** **Élek aránya a maximumhoz képest**

### 4.1.2 Nullélek száma

asd



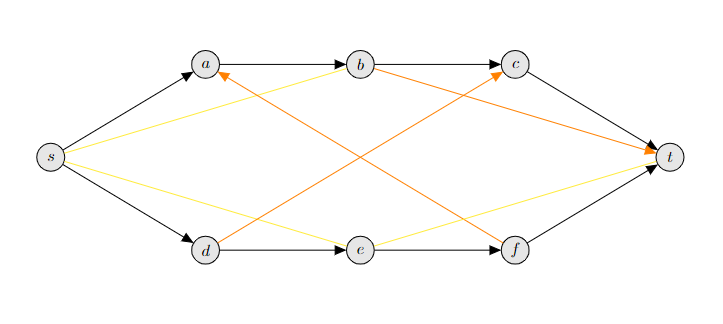
**4.2. ábra: Maximális folyam feladat**



**4.3. ábra: Nullélek a maximális folyamban**

### 4.1.3 Hosszú élek száma

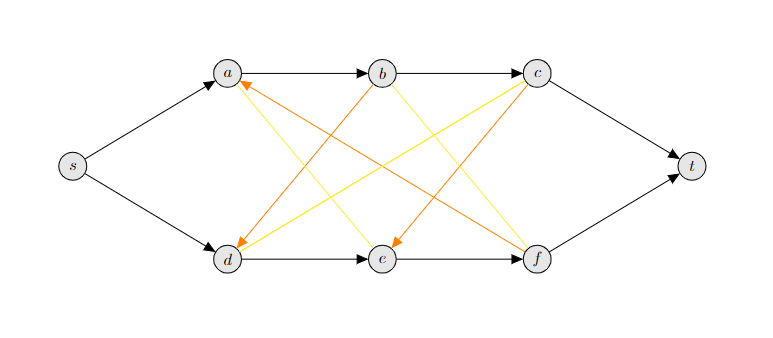
asd



**4.4. ábra: Hosszú élek aránya a maximumhoz képest**

### 4.1.4 Visszafele mutató élek száma

asd



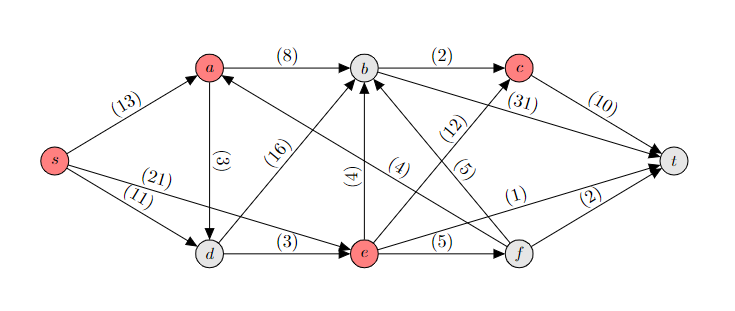
**4.5. ábra: Visszafele mutató élek aránya a maximumhoz képest**

## 4.2 Csúcsok száma

qsd

## 4.3 Vágáserősség

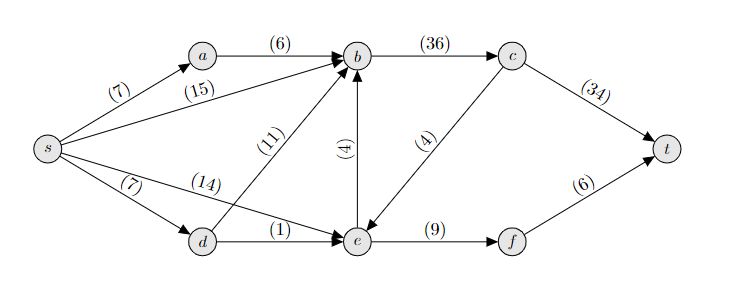
asd



**4.6. ábra: Vágás csúcsai**

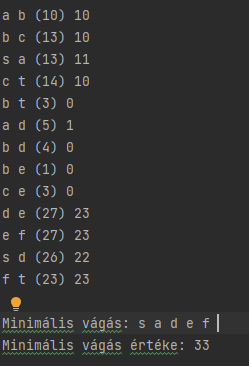
## 4.4 Szórás

ads



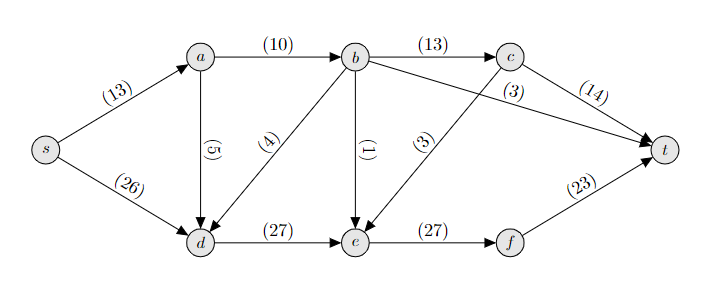
**4.7. ábra: Egyszerű hálózat szórás vizsgálatához**

## 4.5 Teljes képlet



2.9964165529656492 0.16666666666666666 0.23809523809523814 1.0 0.0 1.180775494846827

asdasd



**4.8. ábra: Egyszerű hálózat a képlet bemutatásához**

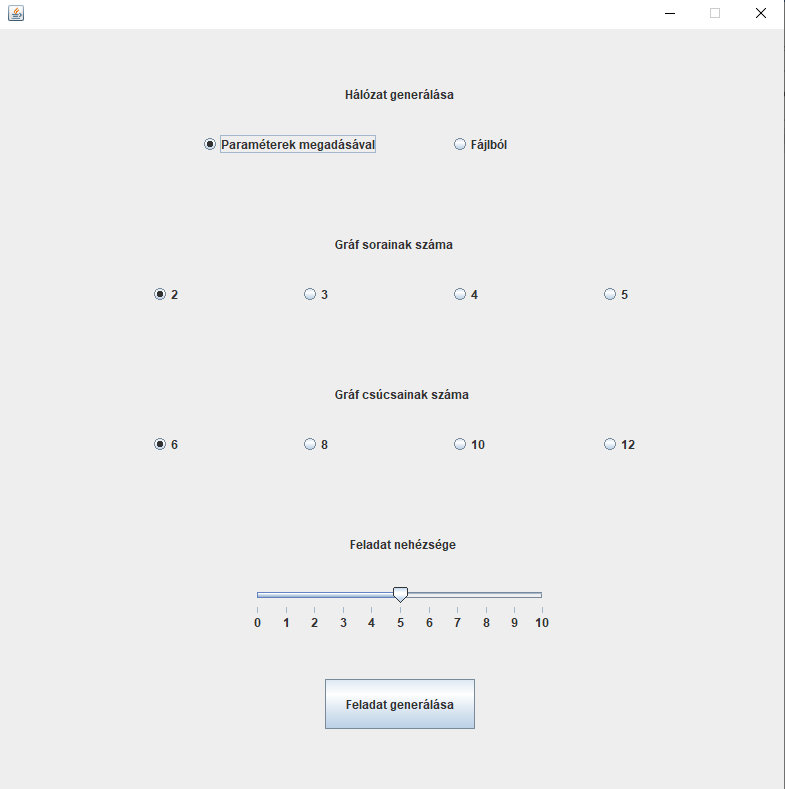
5. fejezet

# A program bemenetei

asd

## 5.1 Grafikus felület kezdőoldala

asd



**5.1. ábra: Kezdőképernyő**

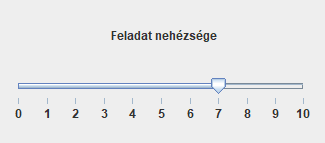
## 5.2 Hálózat generálása paraméterek megadásával

asd

A képen szöveg látható

Automatikusan generált leírás

**5.2. ábra: Gráf sorai és csúcsai számának megadása**



**5.3. ábra: Feladatnehézség megadása**

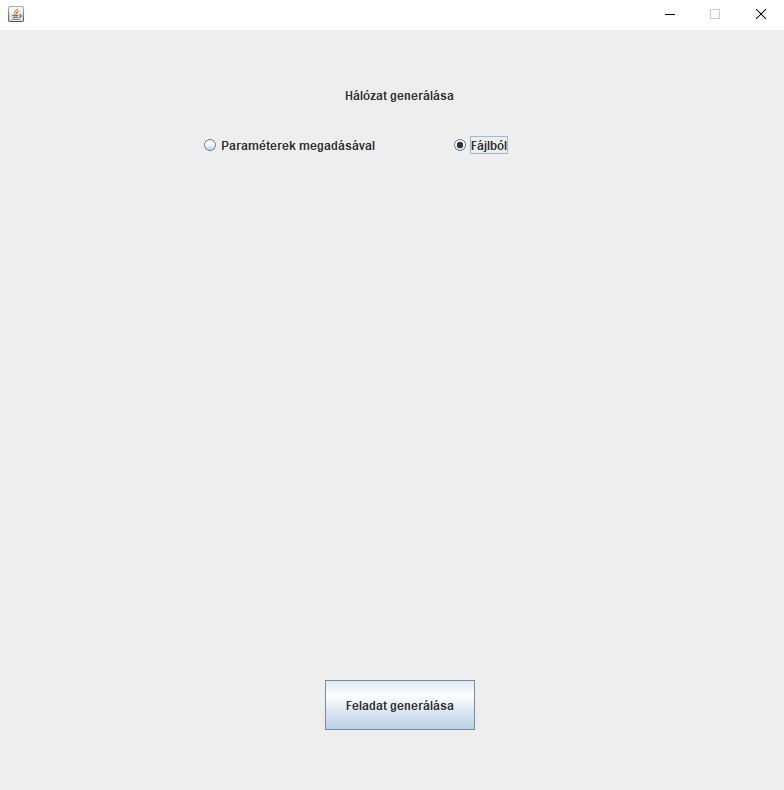
A képen különböző, különféle, változat látható

Automatikusan generált leírás

**5.4. ábra: Generált 3×4-es, 7-es nehézségű feladat**

## 5.3 Hálózat generálása fájlból

asd



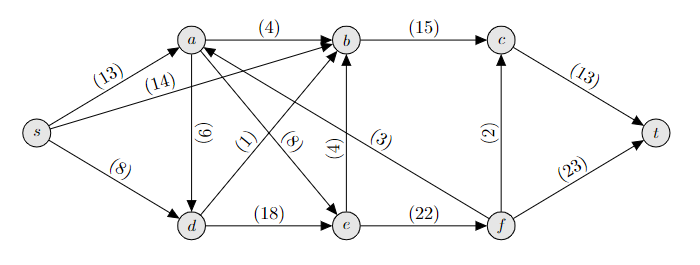
**5.5. ábra: Hálózat generálása fájlból**

### 5.3.1 Hálózat generálása random gráffal

asd



**5.6. ábra: Txt fájl tartalma random gráffal**



**5.7. ábra: Generált feladat random gráffal**

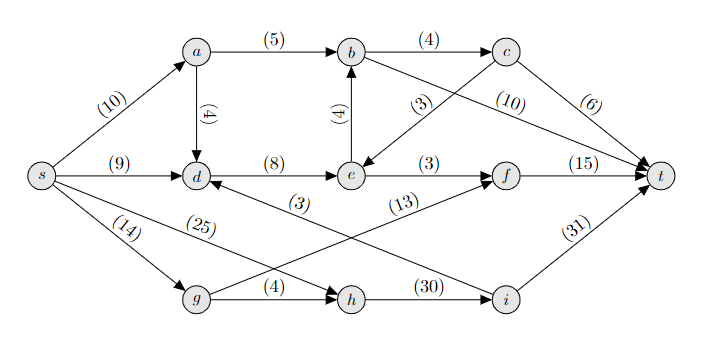
### 5.3.2 Hálózat generálása előre megadott gráffal és vágással

asd

A képen asztal látható

Automatikusan generált leírás

**5.8. ábra: Txt fájl tartalma egy megadoot gráffal és vágással**



**5.9. ábra: Generált feladat megadott gráffal és vágással**

6. fejezet

# A program kimenetei

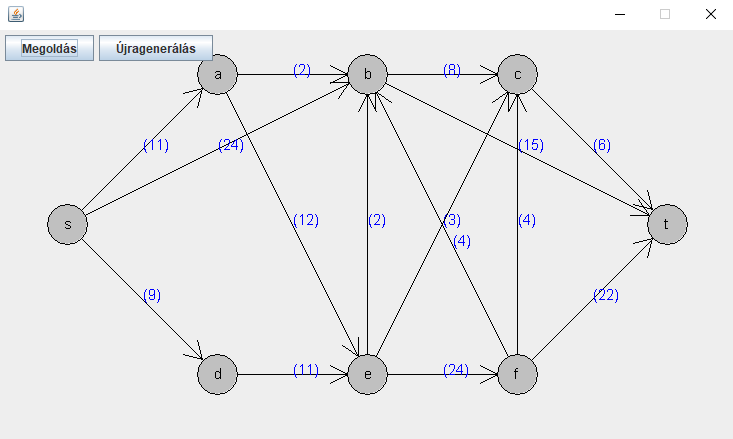
wd

## 6.1 Grafikus megjelenítés

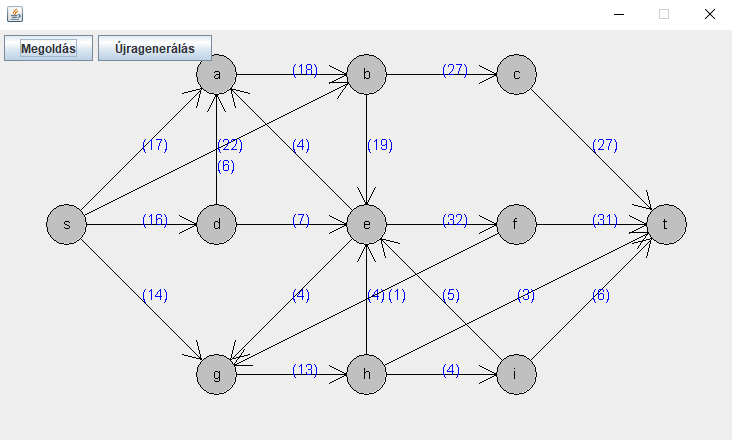
asd

### 6.1.1 Feladat

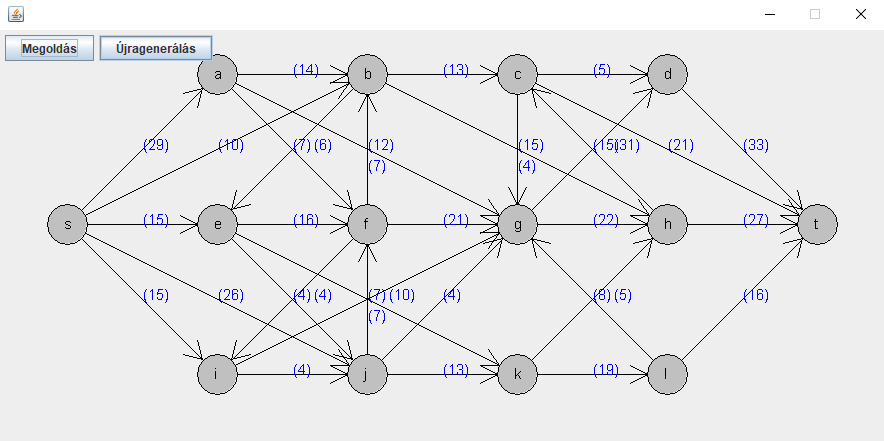
asd



**6.1. ábra: 2×3-as feladat grafikus megjelenítése**



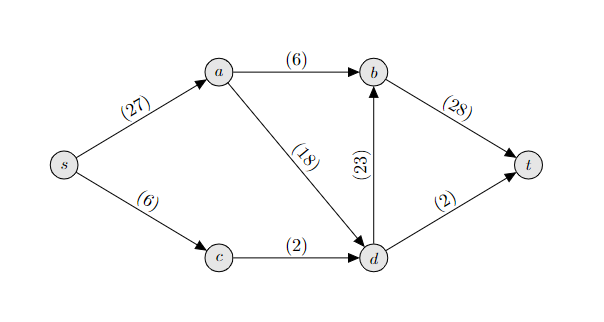
**6.2. ábra: 3×3-as feladat grafikus megjelenítése**



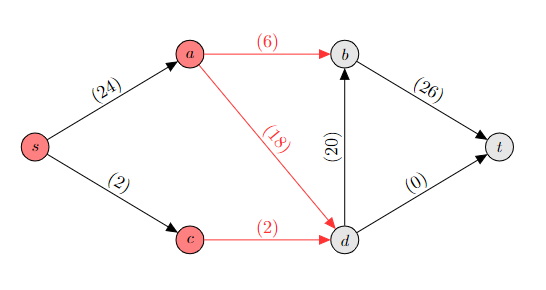
**6.3. ábra: 3×4-es feladat grafikus megjelenítése**

### 6.1.2 Megoldás

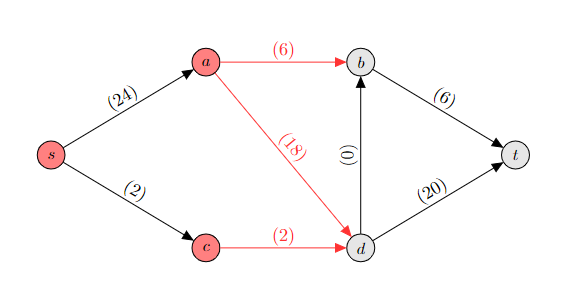
asd



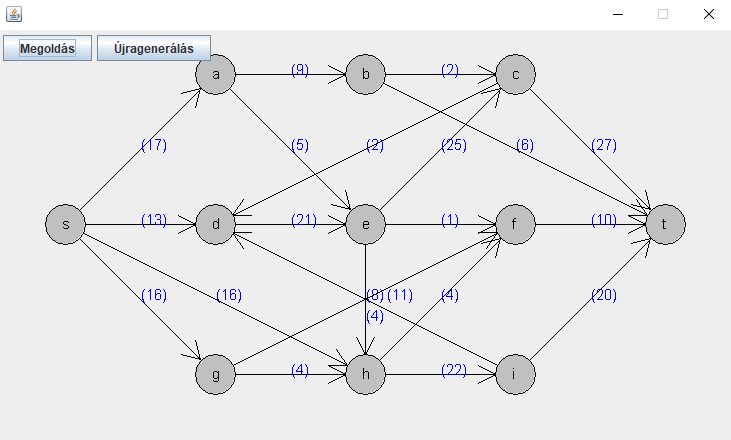
**6.4. ábra: Egyszerű hálózati folyam feladat**



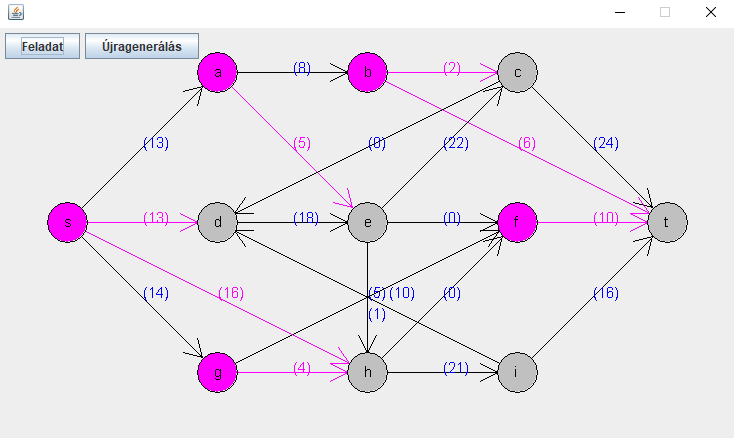
**6.5. ábra: Maximális folyam megoldás 1**



**6.6. ábra: Maximális folyam megoldás 2**



**6.7. ábra: Feladat megjelenítése grafikusan**



**6.8. ábra: Megoldás megjelenítése grafikusan**

### 6.1.3 Újragenerálás

asd

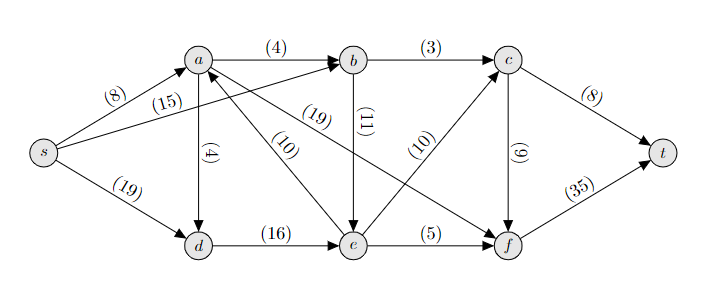
## 6.2 Txt fájl

asd

A képen asztal látható

Automatikusan generált leírás

**6.9. ábra: Txt fájl kimenet**



**6.10. ábra: Txt fájl által leírt feladat**

## 6.3 LaTeX fájl

asd

### 6.3.1 Program által generált LaTeX fájl

asd

\documentclass{article}

\usepackage{tikz}

\usetikzlibrary{arrows}

\tikzset{>=triangle 45}

\begin{document}

\begin{center}

    \tikz[->, csucs/.style={draw, fill=black!10, circle, minimum size={0.6cm}, inner sep=0cm, align=center, scale=0.9},

    scale=3]

    {

        \node [csucs] (a) at (1,2.4000000000000004) {$a$};

        \node [csucs] (b) at (2,2.4000000000000004) {$b$};

        \node [csucs] (c) at (3,2.4000000000000004) {$c$};

        \node [csucs] (s) at (0,1.6) {$s$};

        \node [csucs] (d) at (1,1.6) {$d$};

        \node [csucs] (e) at (2,1.6) {$e$};

        \node [csucs] (f) at (3,1.6) {$f$};

        \node [csucs] (t) at (4,1.6) {$t$};

        \node [csucs] (g) at (1,0.8) {$g$};

        \node [csucs] (h) at (2,0.8) {$h$};

        \node [csucs] (i) at (3,0.8) {$i$};

        \path

        (a) edge node [above=-2pt, pos=0.5, sloped] {(13)} (b)

        (b) edge node [above=-2pt, pos=0.5, sloped] {(9)} (c)

        (s) edge node [above=-2pt, pos=0.5, sloped] {(16)} (a)

        (c) edge node [above=-2pt, pos=0.5, sloped] {(16)} (t)

        (s) edge node [above=-2pt, pos=0.5, sloped] {(11)} (b)

        (a) edge node [above=-2pt, pos=0.5, sloped] {(5)} (d)

        (f) edge node [above=-2pt, pos=0.5, sloped] {(2)} (a)

        (b) edge node [above=-2pt, pos=0.5, sloped] {(5)} (d)

        (e) edge node [above=-2pt, pos=0.5, sloped] {(1)} (b)

        (b) edge node [above=-2pt, pos=0.5, sloped] {(10)} (f)

        (c) edge node [above=-2pt, pos=0.5, sloped] {(2)} (d)

        (f) edge node [above=-2pt, pos=0.5, sloped] {(4)} (c)

        (s) edge node [above=-2pt, pos=0.5, sloped] {(16)} (d)

        (d) edge node [above=-2pt, pos=0.5, sloped] {(17)} (e)

        (e) edge node [above=-2pt, pos=0.5, sloped] {(3)} (f)

        (f) edge node [above=-2pt, pos=0.5, sloped] {(5)} (t)

        (h) edge node [above=-2pt, pos=0.5, sloped] {(2)} (d)

        (h) edge node [above=-2pt, pos=0.5, sloped] {(4)} (e)

        (e) edge node [above=-2pt, pos=0.5, sloped] {(23)} (i)

        (g) edge node [above=-2pt, pos=0.5, sloped] {(9)} (h)

        (h) edge node [above=-2pt, pos=0.5, sloped] {(11)} (i)

        (s) edge node [above=-2pt, pos=0.5, sloped] {(6)} (g)

        (i) edge node [above=-2pt, pos=0.5, sloped] {(33)} (t)

        (s) edge node [above=-2pt, pos=0.5, sloped] {(24)} (h)

        (h) edge node [above=-2pt, pos=0.5, sloped] {(21)} (t)

        ;}

\end{center}

\end{document}

A képen szállítás, nap látható

Automatikusan generált leírás

**6.11. ábra: LaTeX fájllal létrehozott hálózat képe**

### 6.3.2 Javított LaTeX fájl

asd

\documentclass{article}

\usepackage{tikz}

\usetikzlibrary{arrows}

\tikzset{>=triangle 45}

\begin{document}

\begin{center}

    \tikz[->, csucs/.style={draw, fill=black!10, circle, minimum size={0.6cm}, inner sep=0cm, align=center, scale=0.9},

    scale=3]

    {

        \node [csucs] (a) at (1,2.4000000000000004) {$a$};

        \node [csucs] (b) at (2,2.4000000000000004) {$b$};

        \node [csucs] (c) at (3,2.4000000000000004) {$c$};

        \node [csucs] (s) at (0,1.6) {$s$};

        \node [csucs] (d) at (1,1.6) {$d$};

        \node [csucs] (e) at (2,1.6) {$e$};

        \node [csucs] (f) at (3,1.6) {$f$};

        \node [csucs] (t) at (4,1.6) {$t$};

        \node [csucs] (g) at (1,0.8) {$g$};

        \node [csucs] (h) at (2,0.8) {$h$};

        \node [csucs] (i) at (3,0.8) {$i$};

        \path

        (a) edge node [above=-2pt, pos=0.5, sloped] {(13)} (b)

        (b) edge node [above=-2pt, pos=0.5, sloped] {(9)} (c)

        (s) edge node [above=-2pt, pos=0.5, sloped] {(16)} (a)

        (c) edge node [above=-2pt, pos=0.5, sloped] {(16)} (t)

        (s) edge node [above=-2pt, pos=0.35, sloped] {(11)} (b)

        (a) edge node [above=-2pt, pos=0.7, sloped] {(5)} (d)

        (f) edge node [above=-2pt, pos=0.3, sloped] {(2)} (a)

        (b) edge node [above=-2pt, pos=0.5, sloped] {(5)} (d)

        (e) edge node [above=-2pt, pos=0.25, sloped] {(1)} (b)

        (b) edge node [above=-2pt, pos=0.5, sloped] {(10)} (f)

        (c) edge node [above=-2pt, pos=0.7, sloped] {(2)} (d)

        (f) edge node [above=-2pt, pos=0.5, sloped] {(4)} (c)

        (s) edge node [above=-2pt, pos=0.5, sloped] {(16)} (d)

        (d) edge node [above=-2pt, pos=0.5, sloped] {(17)} (e)

        (e) edge node [above=-2pt, pos=0.5, sloped] {(3)} (f)

        (f) edge node [above=-2pt, pos=0.5, sloped] {(5)} (t)

        (h) edge node [above=-2pt, pos=0.5, sloped] {(2)} (d)

        (h) edge node [above=-2pt, pos=0.5, sloped] {(4)} (e)

        (e) edge node [above=-2pt, pos=0.5, sloped] {(23)} (i)

        (g) edge node [above=-2pt, pos=0.5, sloped] {(9)} (h)

        (h) edge node [above=-2pt, pos=0.5, sloped] {(11)} (i)

        (s) edge node [above=-2pt, pos=0.5, sloped] {(6)} (g)

        (i) edge node [above=-2pt, pos=0.5, sloped] {(33)} (t)

        (s) edge node [above=-2pt, pos=0.5, sloped] {(24)} (h)

        (h) edge node [above=-2pt, pos=0.5, sloped] {(21)} (t)

        ;}

\end{center}

\end{document}

A képen szállítás látható

Automatikusan generált leírás

**6.12. ábra: Javított LaTeX fájllal létrehozott hálózat képe**

7. fejezet

# Összegzés

afasd

# Irodalomjegyzék

1. Szeszlér Dávid, Wiener Gábor: Bevezetés a számításelméletbe 2 jegyzet, <http://cs.bme.hu/bsz2/bsz2_jegyzet.pdf>
2. Wikipédia: Maximális áramlási probléma,
3. <https://hu.wikipedia.org/wiki/Maxim%C3%A1lis_%C3%A1raml%C3%A1si_probl%C3%A9ma>
4. Wikipedia: Flow network, <https://en.wikipedia.org/wiki/Flow_network>
5. <http://www.cs.columbia.edu/~bert/courses/3137/hw3_files/GraphDraw.java>