

**Általános alapelvek.**

A pontozási útmutató célja, hogy a javítók a dolgozatokat egységesen értékeljék. Az útmutatónak ebből kifolyólag nem célja a feladatok teljes értékű megoldásának részletes leírása.

Az útmutatóban feltüntetett részpontszámok csak akkor járnak a megoldónak, ha a kapcsolódó gondolat egy áttekinthető, világosan leírt és megindokolt megoldás egy lépéseként szerepel a dolgozatban. A részpontszámok szükség esetén tovább is oszthatók.

Annak mérlegelése, hogy az útmutatóban feltüntetett pontszám a fentiek figyelembevételével a megoldónak (részben vagy egészében) jár-e, teljes mértékben a javító hatásköre. Részpontszám jár minden olyan ötletért, rész megoldásért, amely egy megoldásban érdemi szerephez juthat és amelyből a dolgozatban leírt gondolatmenet alkalmas kiegészítésével a feladat hibátlan megoldása volna kapható. Nem ér pontot azonban például az anyagban szereplő ismeretek, definíciók, tételek puszta leírása azok alkalmazása nélkül (még akkor sem, ha egyébként valamelyik leírt tény a megoldásban valóban szerephez jut).

Az útmutatóban leírttól eltérő jó megoldás természetesen maximális pontot ér, de bizonyítás nélkül csak az előadáson szereplő tételekre és állításokra lehet hivatkozni.

Ha egy megoldó egy feladatra több, egymástól lényegesen különböző megoldást is elkezd, akkor legfeljebb az egyikre adható pontszám. Ha mindegyik leírt megoldás vagy megoldásrészlet helyes vagy helyessé kiegészíthető, akkor a legtöbb részpontot érő megoldáskezdeményt értékeljük. Ha azonban több megoldási kísérlet között van helyes és (lényeges) hibát tartalmazó is, továbbá a dolgozathból nem derül ki, hogy a megoldó melyiket tartotta helyesnek, akkor a kevesebb pontot érő megoldáskezdeményt értékeljük (akkor is, ha ez a pontszám 0).

1. Az alábbi függvényeket rendezzük olyan sorrendbe, hogy ha  $f_i$  után közvetlenül  $f_j$  következik a sorban, akkor  $f_i \in O(f_j)$  teljesüljön, és alkalmas  $c$  konstans és  $n_0$  küszöbindex megadásával bizonyítsuk is be, hogy ez a sorrend helyes.

$$f_1(n) = \frac{2n^2}{\log_2 n}, \quad f_2(n) = n^2 - n, \quad f_3(n) = \frac{2026}{\sqrt{n}} + 5$$

Megfelelő (azaz  $c \cdot f_j(n)$ ) alakú felső becslésekre történő érdemi törekvés: 2 pont

Helyes felső becslések az egyes lépésekben, azaz az  $1/\sqrt{n}$ -nel és az  $1/\log_2 n$ -nel való elbánás, valamint a  $2n^2/\log_2 n$ , illetve a  $-n$  behozása: 1+1+1+2 pont

Az egyes lépésekben használt felső becslésekhez a helyes küszöbindexek megállapítása: 2 pont

Megfelelő végső konstansszorzók és küszöbindexek kiolvasása: 1 pont

Pusztán annak megállapítására, hogy a helyes sorrend  $f_3, f_1, f_2$ , vagy a nagy ordó definíciójának alkalmazás nélküli kimondására nem jár pont. Ha viszont egy megoldó a helyes sorrend felírására valamilyen intuitív érvelést ad, akkor arra az utolsó 8 pontból legfeljebb 2-t kaphat meg.

#### EGY LEHETSÉGES HELYES MEGOLDÁS

Megmutatjuk, hogy a keresett sorrend  $f_3, f_1, f_2$ .

Mivel

$$\frac{2026}{\sqrt{n}} + 5 \underset{\substack{\leq \\ \uparrow \\ \text{ha } n \geq 1, \\ \text{akkor } \sqrt{n} \geq 1, \\ \text{és így } \frac{1}{\sqrt{n}} \leq 1}}{\leq} 2026 + 5 = 2031 \underset{\substack{\leq \\ \uparrow \\ \text{ha } n > 1, \\ \text{akkor } \log_2 n > 0, \\ \text{és ha } n > 0, \\ \text{akkor } \log_2 n \leq n^2; \\ \text{vagyis ha } n > 1, \\ \text{akkor } 1 \leq \frac{n^2}{\log_2 n}}{\leq} \frac{2031}{2} \cdot \frac{2n^2}{\log_2 n},$$

ezért a nagy ordó definíciójában például a  $c = 2031/2$  és az  $n_0 = 2$  értékeket választva látjuk  $f_3 \in O(f_1)$  teljesül.

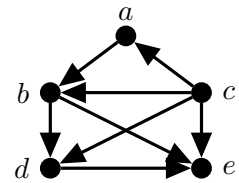
Mivel

$$\frac{2n^2}{\log_2 n} \underset{\substack{\leq \\ \uparrow \\ \text{ha } n \geq 2, \\ \text{akkor } \log_2 n \geq 1, \\ \text{és így } \frac{1}{\log_2 n} \leq 1}}{\leq} 2n^2 \underset{\substack{\leq \\ \uparrow \\ \text{ha } n \geq 3, \\ \text{akkor } 0 \leq n(n-3), \\ \text{azaz } 0 \leq n^2 - 3n}}{\leq} 2n^2 + (n^2 - 3n) = 3(n^2 - n),$$

ezért a nagy ordó definíciójában például a  $c = 3$  és az  $n_0 = 3$  értékeket választva látjuk  $f_1 \in O(f_2)$  teljesül.

2. Van-e a jobboldali irányított gráfnak olyan mélységi bejárása, amiben az  $ab$  él

- (a) keresztél;  
 (b) visszaél?



EGY LEHETSÉGES HELYES MEGOLDÁS

(a) Van, például a következő.



(5 pont)

(b) Mivel a gráf fenti mélységi bejárásban nem keletkezett visszaél, ezért az előadáson tanult tétel szerint semelyik mélységi bejárás során sem fog visszaél keletkezni. (5 pont)

Pusztán annak megállapítására, hogy az (a) kérdésre igen, a (b) kérdésre pedig nem a válasz, nem jár pont. Az első 5 pont abban az esetben jár, ha a megoldásból egyértelműen kiderül, hogy hogyan zajlik egy olyan mélységi bejárás, amiben az  $ab$  él keresztél. A (b) feladatban természetesen meg lehet esetenként is vizsgálni a mélységi bejárásokat, és érvelni amellett, hogy egyik esetben sem keletkezik visszaél. Arra is lehet hivatkozni, hogy a gráf egy DAG – amit meg lehet indokolni például egy topologikus sorrend megadásával –, és így a tanultak szerint semelyik mélységi bejárása során nem keletkezik visszaél (3+2 pont). De elegendő meggyőzően érvelni amellett, hogy a gráfban nincsen olyan irányított kör, ami az  $ab$  élt tartalmazná; ebből is következik, hogy az  $ab$  él nem lehet visszaél, hiszen egy visszaél a megfelelő faélekkel egy irányított kört alkotna (3+2 pont).

Ha egy megoldó nem tudja megindokolni sem az (a), sem a (b) kérdésre a választ, de a megoldásból egyértelműen kiderül, hogy tudja, hogy hogyan kell futtatni a mélységi bejárást, az 2 pontot kapjon.

3. Egy egyetem épületében a földszinti előadótermeket alagsori folyosók is összekötik, és egy mátrixban adott, hogy mely termek között vezet ilyen folyosó. Reggel a tudomásunkra jutott, hogy néhány folyosót felújítás miatt lezártak, és azt is kiderítettük, hogy pontosan melyeket. Jelölje  $n$  az előadótermek számát. Melyik tanult algoritmust lehet alkalmazni, hogyan és miért, ha  $O(n^2)$  lépésben el akarjuk dönteni, hogy át tudunk-e menni az első óránkról a másodikra az alagsori folyosókon keresztül?

Gráf definiálása (csúcsok a termek, élek a le nem zárt folyosók): *3 pont*

Az a kérdés, hogy vezet-e út az első teremből a másodikba: *2 pont*

Szélességi vagy mélységi bejárást használunk: *2 pont*

Lépésszám (szomszédossági mátrix elkészítése, bejárás, és ebből teljes lépésszám): *1+2+0 pont*

#### EGY LEHETSÉGES HELYES MEGOLDÁS

Legyenek a  $G$  gráf csúcsai a termek, és két terem között pontosan akkor vezessen (irányítatlan) él, ha a két terem között van olyan folyosó, ami nincs lezárva. Az a kérdés, hogy vezet-e ebben a gráfban út az első teremnek megfelelő csúcsból a másodikba. Ezt egy, az első teremnek megfelelő csúcsból indított szélességi (vagy mélységi) bejárással el tudjuk dönteni. Ha  $G$ -nek a szomszédossági mátrixát készítjük el, akkor annak a mérete  $O(n^2)$  lesz. Mivel  $G$ -nek  $n$  csúcsa van, ezért a szélességi (vagy a mélységi) bejárás lépésszáma is  $O(n^2)$  lesz. Vagyis a teljes algoritmus lépésszáma  $O(n^2) + O(n^2) = O(n^2)$ .

4. Szomszédossági mátrixával adott egy élsúlyozott, irányított  $G$  gráf, melyben minden él súlya pozitív. Adott továbbá a gráfnak két csúcsa,  $s$  és  $t$ , úgy, hogy nincs él  $s$ -ből  $t$ -be. Melyik tanult algoritmust lehet alkalmazni, hogyan és miért, ha  $O(n^2)$  lépésben szeretnénk meghatározni azt a legkisebb (negatív) súlyt, amivel az  $st$  irányított élt a  $G$  gráfba behúzáva még nem keletkezik negatív összsúlyú irányított kör?

Legrövidebb  $ts$ -utat keresünk: *2 pont*

Annak indoklása, hogy a válasz a legrövidebb  $ts$ -út hosszának a  $(-1)$ -szerese, ha van  $ts$ -út: *3 pont*

Annak az esetnek a kezelése, ha nincs  $ts$ -út: *1 pont*

Megfelelő algoritmus kiválasztása: *1 pont*

Algoritmus használhatóságának megindoklása: *1 pont*

Lépésszám: *2 pont*

#### EGY LEHETSÉGES HELYES MEGOLDÁS

Az új gráfban az  $st$  élt nem tartalmazó tetszőleges kör összsúlya pozitív (hiszen  $G$  minden élének a súlya pozitív). Az  $st$  élt tartalmazó körök összsúlya pedig éppen az  $st$  él súlyának és egy  $G$ -beli  $ts$ -út hosszának az összege lesz. Vagyis a keresett legkisebb súly éppen egy  $G$ -beli legrövidebb  $ts$ -út hosszának a  $(-1)$ -szerese, feltéve, hogy  $G$ -ben létezik  $ts$ -út. Ha  $G$ -ben nem létezik  $ts$ -út, akkor az  $st$  él behúzásával nem jön létre kör, tehát a válasz  $-\infty$ . Mivel  $G$ -ben minden él súlya pozitív, ezért használhatjuk a ( $t$  csúcsból indított) Dijkstra-algortmust, aminek a lépésszáma (szomszédossági mátrixszal való megadás esetén)  $O(n^2)$ .

5. A jobbra látható éllistával adott élsúlyozott, irányított gráfban elkezdjük futtatni a Bellman–Ford-algoritmust a  $c$  csúcsból induló legrövidebb utak meghatározására. Az utoljára lefuttatott fázis eredménye szintén jobbra látható.

**a:**  $b(-1)$ ,  $d(0)$ ;  
**b:**  $a(2)$ ;  
**c:**  $a(5)$ ,  $d(3)$ ;  
**d:**  $b(-2)$

- (a) Fejezzük be az algoritmus futtatását.  
 (b) Határozzunk meg egy legrövidebb  $ca$ -utat, illetve annak a hosszát.  
 (Egyértelműen derüljön ki, mikor mit csinál az algoritmus.)

$a$	$b$	$c$	$d$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$5\ c$	$1\ d$	$0\ *$	$3\ c$

(a) A táblázat, illetve a kitöltéséhez szükséges részszámítások alább láthatók.

$a$	$b$	$c$	$d$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$5\ c$	$1\ d$	$0\ *$	$3\ c$
$3\ b$	$1\ d$	$0\ *$	$3\ c$
$3\ b$	$1\ d$	$0\ *$	$3\ c$

(1+1 pont)

$$\begin{aligned} \text{táv}_k(a) &= \min \{ \text{táv}_{k-1}(b) + w(ba), \text{táv}_{k-1}(c) + w(ca) \} = \\ &= \min \{ 1+2, 0+5 \} = 3 \end{aligned}$$

előző( $a$ ) =  $b$

$$\begin{aligned} \text{táv}_k(b) &= \min \{ \text{táv}_{k-1}(a) + w(ab), \text{táv}_{k-1}(d) + w(db) \} = \\ &= \min \{ 5+(-1), 3+(-2) \} = 1 \end{aligned}$$

előző( $b$ ) =  $d$

$$\text{táv}_k(c) = 0$$

előző( $c$ ) =  $*$

$$\begin{aligned} \text{táv}_k(d) &= \min \{ \text{táv}_{k-1}(a) + w(ad), \text{táv}_{k-1}(c) + w(cd) \} = \\ &= \min \{ 5+0, 0+3 \} = 3 \end{aligned}$$

előző( $d$ ) =  $c$

$$\begin{aligned} \text{táv}_{k+1}(a) &= \min \{ \text{táv}_k(b) + w(ba), \text{táv}_k(c) + w(ca) \} = \\ &= \min \{ 1+2, 0+5 \} = 3 \end{aligned}$$

előző( $a$ ) =  $b$

$$\begin{aligned} \text{táv}_{k+1}(b) &= \min \{ \text{táv}_k(a) + w(ab), \text{táv}_k(d) + w(db) \} = \\ &= \min \{ 3+(-1), 3+(-2) \} = 1 \end{aligned}$$

előző( $b$ ) =  $d$

$$\text{táv}_{k+1}(c) = 0$$

előző( $c$ ) =  $*$

$$\begin{aligned} \text{táv}_{k+1}(d) &= \min \{ \text{táv}_k(a) + w(ad), \text{táv}_k(c) + w(cd) \} = \\ &= \min \{ 3+0, 0+3 \} = 3 \end{aligned}$$

előző( $d$ ) =  $c$

(5 pont)

Mivel az utolsó (most  $(k+1)$ -edikkel jelölt) fázisban nem történt változás az utolsó előttihez képest, ezért az algoritmus leáll, és ezzel megkaptuk a legrövidebb utakat  $c$ -ből a többi csúcsba. (2 pont)

(b) Egy legrövidebb  $ca$ -út (az előző értékekből visszaírva):  $cdba$ , és ennek hossza 3. (1+0 pont)

Az algoritmus futtatásának dokumentálásáért járó 5 pontot azzal lehet megszerezni, ha egyértelműen kiderül, hogy milyen lépéseket tesz az algoritmus; ebből 4 pont jár az egyes csúcsokba vezető legrövidebb utak hosszainak kiszámolására, és 1 pont jár az előző értékek meghatározására. Egy-két számolási hiba 1 pont levonást, több pedig 2 pont levonást jelent. Természetesen az is maximális pontot érhet, ha egy megoldó az elejétől futtatja a Bellman–Ford-algoritmust, és ebben az algoritmus leállítását azzal indokolja, hogy a 4. (ellenőrző) fázis az utolsó. Ha egy megoldó az utolsó fázisban átírja az előző( $d$ ) értékét  $c$ -ről  $a$ -ra, akkor az 1 pont levonást jelent (ez ugyanis elvi hiba).

6. Kirándulásunkra egy  $b$  súlyt elbíró hátizsákba szeretnénk tárgyakat pakolni, ahol  $b$  egy adott pozitív egész szám. Ehhez  $n$ -féle tárgy közül választhatunk, és minden tárgyból 0, 1 vagy 2 darabot vihetünk magunkkal. Adott minden tárgy súlya: az  $i$ -ediké  $s_i > 0$ . Minden tárgyhoz adott, hogy mekkora hasznosságot jelent számunkra, ha belőle egyet, illetve ha belőle kettőt pakolunk a hátizsákba: az  $i$ -edik tárgyra ezek a hasznosságok  $v_i > 0$ , illetve  $w_i > 0$  (a  $v_i$  és  $w_i$  értékek egymástól függetlenek). Adjunk  $O(nb)$  lépésszámú algoritmust annak meghatározására, hogy mi a leghasznosabb olyan pakolás értéke, aminél még nem szakad le a hátizsák.

Részfeladatok megfogalmazása: (2 pont)

Tetszőleges  $i \in \{0, 1, \dots, n\}$  és  $j \in \{0, 1, \dots, b\}$  esetén legyen  $T[i, j]$  az a legnagyobb hasznosság, amit az első  $i$ -féle tárgyból be tudunk pakolni egy  $j$  súlyt elbíró hátizsákba anélkül, hogy az leszakadna.

Inicializálás: (1 pont)

Legyen tetszőleges  $i \in \{0, 1, \dots, n\}$  és  $j \in \{0, 1, \dots, b\}$  esetén  $T[0, j] = T[i, 0] = 0$ .

Kitöltési szabály: (3 pont)

Tetszőleges  $i \in \{1, \dots, n\}$  és  $j \in \{1, \dots, b\}$  esetén

$$T[i, j] = \begin{cases} T[i-1, j], & \text{ha } s_i > j, \\ \max \{T[i-1, j], T[i-1, j-s_i] + v_i\}, & \text{ha } s_i \leq j \text{ és } 2s_i > j, \\ \max \{T[i-1, j], T[i-1, j-s_i] + v_i, T[i-1, j-2s_i] + w_i\}, & \text{különben.} \end{cases}$$

Kitöltési sorrend: (1 pont)

„Fentről lefelé”, soronként „balról jobbra”, azaz  $i$  megy 1-től  $n$ -ig, és minden  $i$  esetén  $j$  megy 1-től  $b$ -ig.

Válasz kiolvasása: (1 pont)

A keresett érték  $T[n, b]$ .

Helyesség bizonyítása: (1 pont)

Az  $i$ -edik féle tárgyról azt kell eldöntenünk, hogy belőle 0, 1 vagy 2 darabot tegyünk-e a  $j$ -kapacitású hátizsákba (már ha egyáltalán belefér a hátizsákba).

Ha 0 darabot pakolunk el belőle, akkor az első  $(i-1)$ -féle tárgyból akarunk minél nagyobb hasznosságot elérni  $j$  súlykorláton belül.

Ha 1 darabot pakolunk el belőle (már ha ennyit elbír a hátizsákunk), akkor már csak  $j - s_i$  súlyt rakhatunk a hátizsákba az első  $(i-1)$ -féle tárgyból, és így akarunk minél nagyobb hasznosságot.

Ha 2 darabot pakolunk el belőle (már ha ennyit elbír a hátizsákunk), akkor már csak  $j - 2s_i$  súlyt rakhatunk a hátizsákba az első  $(i-1)$ -féle tárgyból.

Ezen lehetőségek közül a leghasznosabb lesz a  $T[i, j]$  értéke.

Az inicializálás jelentése, hogy 0-féle tárgyból, vagy egy 0 súlyt elbíró hátizsákba 0 hasznosságot tudunk bepakolni.

Lépésszám: (1 pont)

Egy  $(n+1) \times (b+1)$ -es táblázatot töltünk ki, és minden mező kitöltéséhez  $O(1)$  lépés kell. A válasz kiolvasása  $O(1)$  lépés. Ez összesen  $O((n+1)(b+1)) \cdot O(1) + O(1) = O(nb)$  lépés.

Természetesen az is jó, ha egy megoldó a kitöltési szabályban azt mondja, hogy ha a képletben szereplő tagok második indexe negatív, akkor azt a tagot nem vesszük figyelembe. Ha egy megoldó érdemi kísérletet tesz a kitöltési szabály elmagyarázására, akkor a helyesség bizonyítására járó 1 pontot kapja meg. Ha egy megoldó csak annyit mond, hogy ez a hátizsák probléma, az nem ér pontot (főleg, mert ez nem a klasszikus hátizsák probléma).

7. Egy  $n$ -csúcsú fának minden élét valahogy megirányítottuk; az így kapott  $G$  gráf éllistával adott. Adjunk  $O(n)$  lépésszámú algoritmust egy legtöbb élű irányított út meghatározására.

Egy új $s$ csúcs felvétele:	1 pont
Megfelelő élsúlyozás:	1 pont
Leghosszabb utakat keresünk $s$ -ből:	1 pont
Ennek indoklása:	1 pont
Megfelelő algoritmus kiválasztása:	1 pont
Algoritmus használhatóságának megindoklása:	2 pont
Lépésszám (élek száma, éllista elkészítése, algoritmus futtatása, összegzés):	1+1+1+0 pont

#### EGY LEHETSÉGES HELYES MEGOLDÁS

Vegyünk fel az eredeti  $G$  gráfhoz egy új  $s$  csúcsot, és vezessünk  $s$ -ből minden más csúcsba egy 0-súlyú élt. Az eredeti élek súlya legyen 1. Legyen az így kapott gráf  $G'$ . Erre az élsúlyozásra nézve kell  $G'$ -ben leghosszabb utakat keresnünk  $s$ -ből. Ezek közül a leghosszabb  $G'$ -beli  $s$ -ből induló útból  $s$ -et elhagyva éppen egy keresett legtöbb élű irányított utat kapunk  $G$ -ben. Az eredeti  $G$  gráfban nem volt irányított kör (sőt, irányítatlan értelemben sem volt kör), hiszen egy fában nincs kör, az új 0-befokú csúcs felvételével pedig nem hozhattunk létre irányított kört. Tehát  $G'$  egy DAG, ezért használhatjuk az ( $s$  csúcsból indított) „DAG-os algoritmust” leghosszabb utak meghatározására. A  $G$  gráfnak  $n-1$  éle volt, vagyis a  $G'$  gráfnak  $m' = (n-1) + n = 2n-1$  éle lesz. Tehát a  $G'$  gráf éllistájának elkészítése  $O(n)$  lépés: az eredeti gráf éllistájában szereplő  $n-1$  darab élnek 1 súlyt adunk, és létrehozunk az  $s$  csúcsnak is egy láncolt listát, amiben a  $G$  gráf minden csúcsát felsoroljuk 0 súllyal. A „DAG-os algoritmus” futtatása  $O(n' + m') = O((n+1) + (2n-1)) = O(n)$  lépés. Ez összesen  $O(n) + O(n) = O(n)$  lépés.

Természetesen az is helyes megoldás, ha a  $G'$  gráfban nem adunk az éleknek súlyt (avagy minden élnek 1 súlyt adunk), és legtöbb élből álló irányított utakat keresünk  $s$ -ből – csak ekkor a keresett  $G$ -beli legtöbb élű út hossza éppen eggyel lesz rövidebb mint egy  $G'$ -beli leghosszabb  $s$ -ből induló irányított út.