

# Algoritmusok és adatszerkezetek II.

## régebbi vizsgakérdések.

Ásványi Tibor – asvanyi@inf.elte.hu

2026. január 3.

Struktogram készítésekor a feladat része

- a paraméterek típusának megadása,
- a cím szerinti paraméterátvétel szükség szerinti jelölése,
- *függvények esetében* a visszatérési típus megadása,
- a láncolt adatszerkezetek elemtípusának (pl. E1, E2, Node, Edge) UML leírása,
- a lokális tömbök, objektumok, deklarálása.

Amenyiben a feladatot konkrét adatszerkezetek (tömbök, objektumok, pointerek) szintjén fogalmaztuk meg, akkor a megoldást is ezen a szinten kell megadni. Ilyenkor tehát nem fogadható el absztrakt gráfokat, halmazokat, sorozatokat, címkéket stb. használó megoldás.

Ilyenkor is szabad használni vermeket, sorokat, prioritásos sorokat, ahogy azokat a múlt félévben definiáltuk. Ha ettől eltérünk, a megfelelő kódot is meg kell adni.

A megoldás során figyeljünk a feladatban adott idő- és tárбonyolultságra (time and space complexity)! Az egyébként helyes megoldás pontszámát is erősen lerontja, ha ezeket nem vesszük figyelembe.

Akik nem a múlt félévben tanulták az Algo1-et, vagy nem ebben a félévben az Algo2-t, figyeljenek a terminológia esetleges változásaira!

Az algoritmusok konkrét példákon való bemutatásánál a működést alapértelmezésben az előadáson tanultak szerint kell szemléltetni.

A vizsgákon négy plusz egy összetett feladat van,  $4*24+4=100$  pontért:

- kb. egy az első fejezetből (AVL fák, általános fák, B+ fák),
- kb. kettő a gráfalgoritmusokból és
- kb. egy az utolsó két fejezetből (sztring keresés, tömörítés).
- Az utolsó feladat 10 szakkifejezés fordítása magyarból angolra.

A ponthatárok: 85 → jeles ; 70 → jó ; 55 → közepes ; 40 → elégséges.

Az oktatók nevében kérem, hogy a szokásos szabályokon túl, a vizsgák során vegyék figyelembe az alábbiakat.

- A vizsga előtt ne jöjjönek be a terembe, amíg nem szólítjuk öröket!
- Azokra a helyekre üljenek, ahol találnak kérdéssort és üres papírt!
- A megoldáshoz csak kék (fekete és/vagy zöld) színnel író tollra, a diákokigazolványukra, hagyományos órára és esetleg egy üveg innivalóra lesz szükségük. Kérjük, hogy semmi egyéb ne legyen öröknél! Ennek a szabálynak a megszegése automatikusan elégtelen osztályzatot eredményez.
- Kérjük, vegyék le a sapkájukat a vizsga előtt!
- A táskájukat, hátizsákjukat, a kültérben szükséges ruhadarabjaikat és minden egyéb, a vizsgához szükségtelen kelléküket az üresesen maradt sorokba tegyék úgy, hogy azok a sorokban az átjárást ne akadályozzák!
- A vizsga során lehet kérdezni. mindenkit úgy ültetünk le, hogy oda tudunk hozzá menni, és közvetlenül tudunk vele kommunikálni. Ha nem egyértelmű ööknek a kérdés, vagy elakadnak egy kreatív feladat megoldásával, igyekszünk segítséget nyújtani, a továbbhaladáshoz ötletet adni.

Az alábbiakban vizsgakérdésekre adunk példákat.

# 1. Fák

## 1.1. AVL fák

**1.a,** A közönséges bináris keresőfákkal kapcsolatos fogalmakat ismertnek feltételezve, mondjuk ki az AVL fa meghatározásához szükséges definíciókat!

**1.b,** Adott a

{ [ (2) 3 ( { } 4 {5} ) ] 7 [ (8) 9 () ] } AVL fa. Rajzolja le a fát a csúcsok egyensúlyaival együtt! Szemléthesse az előadásról ismert módon a 7 törlését és a 6 beszúrását, **mindkét esetben az eredeti fára!** (Törléskor, indeterminisztikus esetben a jobb részfa minimumát használjuk!) Jelölje, ha ki kell egyensúlyozni, a kiegyensúlyozás helyét, és a kiegyensúlyozás után is rajzolja újra fát! A rajzokon jelölje a belső csúcsoknak az algoritmus által nyilvántartott egyensúlyait is, a szokásos módon!

**1.c,** Rajzolja le a hat általános kiegyensúlyozási séma közül azokat, amiket alkalmazott!

**2.a,** A bináris keresőfákkal kapcsolatos fogalmakat ismertnek feltételezve, mondja ki az AVL fa meghatározásához szükséges definíciókat!

**2.b,** Szemléthesse az 1 beszúrását és a 4 törlését, **mindkét esetben a { [ (2) 3 () ] 4 [ (5) 6 ( {7} 8 { } ) ] }** AVL fára! (Törléskor, indeterminisztikus esetben a jobb részfa minimumát használjuk!) Jelölje, ha ki kell egyensúlyozni, a kiegyensúlyozás helyét, és a kiegyensúlyozás után is rajzolja újra fát! A rajzokon jelölje a belső csúcsoknak az algoritmus által nyilvántartott egyensúlyait is, a szokásos módon!

**2.c,** Rajzolja le a hat általános kiegyensúlyozási séma közül azokat, amiket alkalmazott!

**3.a,** Adja meg az előadásról ismert **AVLremMin( $t, minp, d$ )** rekurzív eljárás és segédeljárásai struktogramjait, ami a  $t \neq \emptyset$  AVL fából kiveszi a legkisebb kulcsú csúcsot, és  $minp$ -be teszi a címét! A  $d$  logikai típusú paraméterben azt kapjuk meg, hogy a művelet hatására csökkent-e a fa magassága. A kiegyensúlyozási szabályokat megvalósító eljárások közül elég a  $(++, 0)$  esetet kódolót részletezni, a szabálynak megfelelő ábrával együtt.

**3.b,** Igaz-e, hogy  $MT(n) \in \Theta(\log n)$ ? Miért?

**4.a,** A bináris keresőfákkal kapcsolatos fogalmakat ismertnek feltételezve, mondja ki az AVL fa meghatározásához szükséges definíciókat!

**4.b,** Az AVL fák mérete és magassága között milyen összefüggést ismer? Mi ennek a jelentősége az AVL fák műveletei szempontjából? Mely műveletek hatékonyさága függ az AVL fa magasságától?

**4.c,** Rajzolja le az előadásról ismert módon az AVL fák kiegyensúlyozási

sémáit a  $(--, -)$ -os és a  $(--, +)$  esetekben! Mutassa be ezek működését egy-egy egyszerű példán, ahol azonban egyik, a sémákban jelölt részfa sem üres!

**4.d,** Adja meg a  $(--, -)$ -os kiegyensúlyozás struktogramját! Mekkora a műveletigénye?

**5.a,** Adjon olyan  $- 0, 1, 2, 3$  és  $4$  magasságú — AVL fákat, amik egyben Fibonacci fák is!

**5.b,** Definiálja a Fibonacci fák magassága és mérete közti összefüggést leíró rekurzív  $f_h$  függvényt!

**5.c,** Igaz-e, hogy adott magasságú kiegyensúlyozott fák között a Fibonacci fák a legkisebb méretűek? Miért?

**5.d,** Mondja ki az AVL fák mérete és magassága közti összefüggésre vonatkozó kettős egyenlőtlenséget, és írja le a bizonyítás vázlatát!

**6.a,** A bináris keresőfa fogalmát ismertnek feltételezve, mondja ki az AVL fa meghatározásához szükséges definíciókat!

**6.b,** Rajzolja le az  $\{ [ (1) 4 ( \{6\} 7 \{ \} ) ] 9 [ (11) 14 () ] \}$  AVL fát a csúcsok egyensúlyaival együtt!

**6.c,** Szemléltesse az előadásról ismert módon a legnagyobb kulcsú csúcs törlését és az 5 beszúrását, **mindkét esetben az eredeti fára!** Jelölje, ha ki kell egyensúlyozni, a kiegyensúlyozás helyét, és a kiegyensúlyozás után is rajzolja újra a fát! A rajzokon jelölje a belső csúcsoknak az algoritmus által nyilvántartott egyensúlyait is, a szokásos módon!

**6.d,** Rajzolja le a hat általános kiegyensúlyozási séma közül azokat, amiket alkalmazott!

**7.a,** Adja meg az előadásról ismert **AVLinsert**( $t, k, d$ ) rekurzív eljárás struktogramját, ami a  $t$  AVL fába beszúrja a  $k$  kulcsot! A  $d$  logikai típusú paraméterben azt adjuk vissza, hogy a művelet hatására növekedett-e a fa magassága. Az **AVLinsert**( $t, k, d$ ) segédeljárásai közül elég a **rightSubTreeGrow()** és a **balancePPp()** struktogramjait megadni, a megfelelő paraméterezzel, ez utóbbi esetében a megvalósított kiegyensúlyozási szabálynak  $(++, +)$  megfelelő ábrával együtt.

**7.b,** Igaz-e, hogy  $MT(n) \in \Theta(\log n)$ ? Miért?

**8.a,** Adja meg az előadásról ismert **AVLinsert**( $t, k, d$ ) rekurzív eljárás struktogramját, ami a  $t$  AVL fába beszúrja a  $k$  kulcsot! A  $d$  logikai típusú paraméterben azt adjuk vissza, hogy a művelet hatására növekedett-e a fa magassága. Az **AVLinsert**( $t, k, d$ ) segédeljárásai közül elég a **rightSubTreeGrow()** és a **balancePPm()** struktogramjait megadni, a megfelelő paraméterezzel,

ez utóbbi esetében a megvalósított kiegyensúlyozási szabálynak  $(++, -)$  megfelelő ábrával együtt.

**8.b,** Igaz-e, hogy  $MT(n) \in \Theta(\log n)$ ? Miért?

**9.a,** Adja meg az előadásról ismert **AVLInsert**( $t, k, d$ ) rekurzív eljárás struktogramját, ami a  $t$  AVL fába beszúrja a  $k$  kulcsot! A  $d$  logikai típusú paraméterben azt adjuk vissza, hogy a művelet hatására növekedett-e a fa magassága. Az **AVLInsert**( $t, k, d$ ) segédeljárásai közül elég a `leftSubTreeGrow()` és a `balanceMMm()` struktogramjait megadni, a megfelelő paraméterezővel, ez utóbbi esetében a megvalósított kiegyensúlyozási szabálynak  $(--, -)$  megfelelő ábrával együtt.

**9.b,** Igaz-e, hogy  $MT(n) \in \Theta(\log n)$ ? Miért?

**10.a,** Adja meg az előadásról ismert **AVLInsert**( $t, k, d$ ) rekurzív eljárás struktogramját, ami a  $t$  AVL fába beszúrja a  $k$  kulcsot! A  $d$  logikai típusú paraméterben azt adjuk vissza, hogy a művelet hatására növekedett-e a fa magassága. Az **AVLInsert**( $t, k, d$ ) segédeljárásai közül elég a `leftSubTreeGrow()` és a `balanceMMP()` struktogramjait megadni, a megfelelő paraméterezővel, ez utóbbi esetében a megvalósított kiegyensúlyozási szabálynak  $(--, +)$  megfelelő ábrával együtt.

**10.b,** Igaz-e, hogy  $MT(n) \in \Theta(\log n)$ ? Miért?

**11\*.** A  $t : \text{Node}^*$  típusú pointer egy láncoltan ábrázolt bináris fát azonosít. A fa csúcsaiban nincsenek „parent” pointerek. Írja meg a **Fibonacci**( $t$ ) rekurzív logikai függvényt, ami  $\Theta(|t|)$  műveletigénnyel eldönti, hogy a  $t$  Fibonacci fa-e! A függvény a fát ne változtassa meg!

## 1.2. Általános fák

**1.a,** Rajzolja le részletesen az  $\{1 [2 (5)] [3] [4 (6) (7)]\}$  általános fa binárisan láncolt reprezentációját! A rajzon a csúcsok szerkezete és a mutatók iránya is világosan látható legyen!

**1.b,** A fenti reprezentációhoz adja meg a fa csúcsai osztályának UML leírá-sát!

**1.c,** A  $t$  pointer egy binárisan láncolt általános fát azonosít. Írja meg a **faKi**( $t$ ) eljárást, ami kiírja a fát szöveges (zárójeles) alakban,  $\Theta(n)$  műveletigénnyel és  $O(h)$  tárigénnyel, ahol  $n$  a  $t$  fa mérete és  $h$  a magassága! A kiíratásnál elegendő csak kerek zárójeleket alkalmazni.

**2.a,** Rajzolja le részletesen az  $\{1 [2] [3 (4 \{5\}) (6 \{7\} \{8\}) (9)]\}$  általános fa binárisan láncolt reprezentációját! A rajzon a csúcsok szerkezete és a mutatók iránya is világosan látható legyen!

**2.b,** A fenti reprezentációhoz adja meg a fa csúcsai osztályának UML leírá-sát!

**2.c,** A  $t$  pointer egy binárisan láncolt általános fát azonosít. Írja meg a  $\text{height}(t)$  függvényeljárást, ami kiszámolja a  $t$  fa magasságát,  $\Theta(n)$  művelet-igénnnyel és  $O(h)$  tárígénnyel, ahol  $n$  a  $t$  fa mérete és  $h$  a magassága!

### 1.3. B+ fák

**1.a,** Egy  $d$ -edfokú B+ fa csúcsaiban 4 bájtos kulcsok és 6 bájtos pointerek vannak. A B+ fát mágneslemezen tároljuk, ahol a blokkméret 4096 bájt. Mekkorának érdemes választani a B+ fa  $d$  fokszámát? Miért?

**1.b,** Rajzolja le a  $[ (9\ 10)\ 11\ (12\ 13)\ 14\ (15\ 16\ 17)\ 18\ (19\ 20) ]$  negyedfokú B+ fát! Szemléltesse az előadásról ismert algoritmus szerint a 14 beszúrását!

**1.c,** Adott az  $\{ [ (1\ 2)\ 3\ (4\ 5) ]\ 7\ [ (11\ 15\ 20)\ 27\ (27\ 30) ] \}$  negyedfokú B+ fa. Rajzolja le a fát! Szemléltesse az előadásról ismert módon a 18 beszúrását, valamint a 30 és a 4 törlését, **mindhárom esetben az eredeti fára!**

**2.a,** A  $d$ -edfokú B+ fák **belső** csúcsainak milyen tulajdonságait ismeri?

$\{ [ (1\ 2)\ 3\ (5\ 6\ 7) ]\ 8\ [ (9\ 10)\ 11\ (12\ 13)\ 14\ (14\ 16\ 17)\ 18\ (19\ 20) ] \}$

**2.b,** Adott a fenti negyedfokú B+ fa. Rajzolja le a fát! **2.c,** Szemléltesse az előadásról ismert algoritmus szerint a 11, a 4 és a 15 beszúrását, **mindhárom esetben az eredeti fára!**

**3.a,** A  $d$ -edfokú B+ fák *levél* csúcsainak milyen tulajdonságait ismeri?

**3.b,**  $[ (9\ 10)\ 11\ (12\ 13\ 14)\ 15\ (15\ 16)\ 18\ (19\ 20) ]$

Rajzoluk le a fenti negyedfokú B+ fát! Szemléltessük az előadásról ismert algoritmus szerint a 11 beszúrását, és a 9, illetve a 19 törlését, **mindhárom esetben az eredeti fára!**

**4.a,** A  $d$ -edfokú B+ fák *levél* csúcsainak milyen tulajdonságait ismeri?

**4.b,**  $\{ [ (1\ 2)\ 3\ (5\ 6\ 7) ]\ 9\ [ (9\ 10)\ 11\ (12\ 13)\ 14\ (15\ 16\ 17)\ 18\ (19\ 20) ] \}$

Rajzoluk le a fenti negyedfokú B+ fát! Szemléltessük az előadásról ismert algoritmus szerint a 14 beszúrását, és az 1, illetve a 9 törlését, **mindhárom esetben az eredeti fára!**

**5.a,** A  $d$ -edfokú B+ fák **belső** csúcsainak milyen tulajdonságait ismeri?

$\{ [ (1\ 2)\ 3\ (5\ 6) ]\ 8\ (9\ 10\ 11)\ 12\ (12\ 13) ]\ 14\ [ (14\ 17)\ 18\ (19\ 20) ] \}$

**5.b,** Adott a fenti negyedfokú B+ fa. Rajzolja le a fát!

**5.c,** Szemléltesse az előadásról ismert algoritmus szerint a 8 beszúrását, valamint a 2 és a 14 törlését, **mindhárom esetben az eredeti fára!**

**6.a,** Hol helyezkednek el a B+ fában tárolt kulcshalmaz elemei? Mi a többi kulcs szerepe?

**6.b,** Tegyük fel, hogy adott egy  $d$ -edfokú,  $h$  magasságú  $B+$  fa! Adjon alsó és felső becslést a  $B+$  fában tárolt kulcsalmaz  $n$  méretére! (Indokolja is a becsléseket!)

**6.c,** Tegyük fel, hogy egy  $B+$  fára  $n = 10^9$  és  $d = 400$ ; legfeljebb mekkora lehet a fa  $h$  magassága, és miért?

**6.d,** Tegyük fel, hogy egy  $d$ -edfokú  $B+$  fában egy  $n$  méretű kulcsalmazt tárolunk! Adjon alsó és felső becslést a fa  $h$  magasságára! (Indokolja is a becsléseket!)

**6.e,** Milyen kapcsolatban áll a  $B+$  fa  $h$  magassága a keresés, a beszúrás és a törlés műveletigényével?

## 2. Gráfalgoritmusok

### 2.1. Gráfprezentációk

A gráfok reprezentációnál a gráfok csúcsait az egész Gráfalgoritmusok fejezetben az  $1 \dots n$  sorszámokkal azonosítjuk, ahol  $n \geq 1$  egész szám.

**1.** A  $C/1 : \mathbb{B}[n, n]$  bitmátrix egy irányított gráf szomszédossági mátrixos ábrázolása. Írja meg a **transzformál**( $C, G$ ) eljárást, ami előállítja a  $G/1 : Edge*[n]$  gráfot, ami a  $C$  mátrixszal reprezentált gráf szomszédossági listás reprezentációja! A szomszédossági listák egyszerű láncolt listák (fejelem nélküli, nemciklikus, egyirányú listák) legyenek!  $MT(n) \in \Theta(n^2)$ .

**2.a,** Rajzolja le a következő irányított gráf szomszédossági listás reprezentációját!<sup>1</sup> **1 → 2.** **2 → 1 ; 3.**

**2.b,**  $G/1 : Edge*[n]$  egy irányított, élsúlyozatlan gráf szomszédossági listás ábrázolása. A listák egyszerű láncolt listák (fejelem nélküli, nemciklikus, egyirányú listák). Adja meg az *Edge* osztály UML leírását!

**2.c,** Írja meg a **transzponál**( $G, GT$ ) eljárást, ami előállítja a  $GT/1 : Edge*[n]$  gráfot! Ez utóbbi a  $G/1 : Edge*[n]$  gráf transzponáltjának szomszédossági listás reprezentációja. A  $G$  gráfot ne változtassuk meg!  $MT(n, m), MS(n, m) \in \Theta(n + m)$ , ahol  $m$  a gráf éleinek száma.

**3.a,**  $G/1 : Edge*[n]$  egy irányított, élsúlyozatlan gráf szomszédossági listás ábrázolása. A  $G[i]$  listák egyszerű láncolt listák (fejelem nélküli, egyirányú, nemciklikus listák). Adja meg a listaelem típus UML leírását!

**3.b,** Írja meg a **kibeFokok**( $G, be, ki$ ) eljárást, ami minden  $u \in 1 \dots n$  csúcsra a  $be[u]$ -ban kiszámítja a csúcs bemeneti fokszámát,  $ki[u]$ -ban pedig a kimeneti fokszámát!  $MT(n, m) \in \Theta(n + m)$ ,  $MS(n, m) \in \Theta(1)$ , ahol  $m$  a gráf éleinek száma.

### 2.2. Szélességi keresés

**1.a,** Milyen gráfokon és mit számol ki a *Szélességi gráfkeresés*?

**1.b,** Adja meg az algoritmus absztrakt struktogramját!

**1.c,** A *Szélességi gráfkeresés* a gráf mely csúcsaiba talál optimális utat, és a végrehajtás során mikor?

**1.d,** Mit tud a *Szélességi gráfkeresés* műveletigényéről? (Indokolja is az állítást!)

**1.e,** Szemléltesse az algoritmust az **a** csúcsból indítva, az

<sup>1</sup> $u \rightarrow v_1; \dots; v_k$ . azt jelenti, hogy az  $u$  csúcs közvetlen rákövetkezői  $v_1, \dots, v_k$ .

**a – b ; d. b – c ; d. c – e. d – e.**

irányítatlan gráfon!<sup>2</sup> Rajzolja le az eredményül adódó szélességi fát is!

**2.a,** Milyen gráfokon és mit számol ki a *Szélességi gráfkeresés*?

**2.b,** Rajzolja le részletesen az alábbi irányított gráf (i) grafikus, (ii) csúcs-mátrixos és (iii) szomszédossági listás ábrázolását!<sup>3</sup>

**a → b ; d. b → c ; d. c → e. e → c ; d.**

**2.c,** Szemléthesse a *Szélességi gráfkeresést* az **a** csúcsból indítva, a fenti gráfon! Rajzolja le az eredményül adódó szélességi fát is!

**2.d,** Mekkora a *Szélességi gráfkeresés* műveletigénye, és miért?

**3.a,** Milyen gráfokon és mit számol ki a *Szélességi gráfkeresés*?

**3.b,** Rajzolja le részletesen az alábbi irányított gráf szomszédossági listás ábrázolását!<sup>4</sup> **1 → 2. 2 → 3; 1.**

**3.c,**  $G/1 : Edge*[n]$  egy élsúlyozatlan gráf szomszédossági listás ábrázolása. A  $G[i]$  listák egyszerű láncolt listák (fejelem nélküli, egyirányú, nemciklikus listák). Adja meg az *Edge* osztály UML leírását!

**3.d,** Írja meg a **BFS**( $G, s, d, \pi$ ) eljárást, ami a fenti módon ábrázolt  $G$  gráfra kiszámolja annak  $s$  gyökércsúcsú szélességi faját! A csúcsok  $d$  és  $\pi$  értékeit a megfelelő nevű tömbök reprezentálják! Ne feledkezzen meg a paraméterek típusának megadásáról sem!  $MT(n, m) \in \Theta(n + m)$ , ahol  $m$  a gráf éleinek száma.

### 2.3. Mélységi keresés és topologikus rendezés

**1.a,** Mit értünk egy irányított gráf csúcsainak topologikus rendezése alatt? Mondja ki és bizonyítsa be az ezzel kapcsolatos tételeit!

**1.b,** Mutassa be az alábbi gráf<sup>5</sup> csúcsai topologikus rendezésének a gráf mélységi bejárására épülő algoritmusát! (Az algoritmus szemléltetése során a nemdeterminisztikus esetekben mindenkor az alfabetikusan kisebb indexű csúcsot részesítse előnyben!)

**1.c,** Módosítsa egyetlen él behúzásával úgy a gráfot, hogy ne legyen topologikus rendezése! A módosított gráfnak miért nincs topologikus rendezése? Mikor derül ez ki a fent szemléltetett algoritmus végrehajtása során?

**1.d,** Mit tud a mélységi bejárásra épülő topologikus rendezés műveletigényéről? (Indokolja is az állítást!)

**a → b ; d. b → c ; d. c → e. d → e. f → c ; e.**

---

<sup>2</sup> $u - v_1; \dots; v_k$ . azt jelenti, hogy az *irányítatlan gráfban* az  $u$  csúcs  $u$ -nál nagyobb indexű szomszédai  $v_1, \dots, v_k$ . (Ezzel a jelöléssel a gráf minden élét csak egyszer tüntetjük fel.)

<sup>3</sup> $u \rightarrow v_1; \dots; v_k$ . azt jelenti, hogy az  $u$  csúcs közvetlen rákövetkezői  $v_1, \dots, v_k$ .

<sup>4</sup> $u \rightarrow v_1; \dots; v_k$ . azt jelenti, hogy a gráfban az  $u$  csúcs közvetlen rákövetkezői  $v_1, \dots, v_k$ .

<sup>5</sup> $u \rightarrow v_1; \dots; v_k$ . azt jelenti, hogy az  $u$  csúcs közvetlen rákövetkezői  $v_1, \dots, v_k$ .

**2.a,** Rajzolja le a *Mélységi gráfkeresés* absztrakt struktogramját! Mit tud a műveletigényéről? (Indokolja is az állítást!)

**2.b,** Adja meg az éltípusok definícióját és mondja ki az osztályozásukkal kapcsolatos tétele!

**2.c,** Szemléltesse a *Mélységi keresést* az alábbi irányított gráfon<sup>6</sup> úgy, hogy nemdeterminisztikus esetekben minden indexű csúcsot részesítse előnyben! Jelölje a bejárás során a különböző éltípusokat is!

$$a \rightarrow b ; d. \quad b \rightarrow c ; d. \quad c \rightarrow e. \quad d \rightarrow e. \quad e \rightarrow b. \quad f \rightarrow c ; e.$$

## 2.4. Minimális feszítőfák

### 2.4.1. Kruskal algoritmusa

**1.a,** Milyen gráfokon és mit számol ki a *Kruskal* algoritmus?

**1.b,** Szemléltesse a működését az előadásról ismert módon az alábbi gráfon!<sup>7</sup>

**1.c,** Mondja ki a biztonságos elekről és a minimális feszítőfákról szóló tétele! Definiálja a tételeben szereplő *vágás* és *könnyű él* fogalmakat!

**1.d,** Hogyan következik a *Kruskal* algoritmus helyessége ebből a tételeből?

**1.e,** Mekkora az algoritmusnak az előadásról ismert műveletigénye, és milyen feltételekkel?

$$a - b, 0 ; d, 1. \quad b - c, 5 ; d, 2 ; e, 3. \quad d - e, 4.$$

**2.a,** Milyen gráfokon és mit számol ki a *Kruskal* algoritmus?

**2.b,** Szemléltesse a működését az előadásról ismert módon az  
 $a - b, 3 ; d, 1. \quad b - c, 5 ; d, 2 ; e, 3. \quad c - e, 4. \quad d - e, 0.$   
gráfon!<sup>8</sup>

**2.c,** Adja meg a fő eljárás struktogramját! Magyarázza el a segédeljárások és a segédfüggvény működését!

**2.d,** Mekkora az algoritmus műveletigénye? Miért?

### 2.4.2. Prim algoritmusa

**1.a,** Milyen gráfokon és mit számol ki a Prim algoritmus?

**1.b,** Definiálja a súlyozott szomszédossági csúcsmátrix fogalmát!

**1.c,** Csak tömb adatszerkezeteket használva adja meg a  $\text{Prim}(C, r, d, \pi)$  eljárás struktogramját, ahol a gráfot a  $C[1..n, 1..n]$  súlyozott szomszédossági csúcsmátrix segítségével ábrázoltuk, és a fa építése az  $r$  csúcsból indul. A

<sup>6</sup> $u \rightarrow v_1; \dots; v_k$ . azt jelenti, hogy az  $u$  csúcs közvetlen rákövetkezői  $v_1, \dots, v_k$ .

<sup>7</sup> $u - v_1, w_1; \dots; v_k, w_k$ . azt jelenti, hogy a gráfban az  $u$  csúcs  $u$ -nál nagyobb indexű szomszédai  $v_1, \dots, v_k$ , és a megfelelő irányítatlan élek súlyai sorban  $w_1, \dots, w_k$ .

<sup>8</sup> $u - v_1, w_1; \dots; v_k, w_k$ . azt jelenti, hogy a gráfban az  $u$  csúcs  $u$ -nál nagyobb indexű szomszédai  $v_1, \dots, v_k$ , és a megfelelő irányítatlan élek súlyai sorban  $w_1, \dots, w_k$ .

segédeljárásokat is részletezze! Az eredményt, a csúcsok  $d$  és a  $\pi$  értékeit a  $d[1..n]$  és a  $\pi[1..n]$  vektorokban kapjuk.  $MT(n) \in O(n^2)$ ,  $MS(n) \in O(n)$

**2.a,** Milyen gráfokon és mit számol ki a *Prim* algoritmus?

**2.b,** Szemléltesse a működését az

**a – b, 0 ; d, 1. b – c, 5 ; d, 2 ; e, 3. c – e, 2. d – e, 2.**

irányítlan gráfon, a **d** csúcsból indítva!<sup>9</sup>.

**2.c,** Mondja ki a biztonságos élekről és a minimális feszítőfákról szóló tételt! Definiálja a tételben szereplő *vágás* és *könnyű él* fogalmakat! Hogyan következik a *Prim* algoritmus helyessége ebből a tételből?

**2.d,** Mekkora az algoritmusnak az előadásról ismert műveletigénye, és milyen feltételekkel?

## 2.5. Legrövidebb utak egy forrásból

### 2.5.1. Legrövidebb út kiírása

**1.** A Dijkstra algoritmus eredményeképpen megkaptuk a  $d/1:\mathbb{R}[n]$  és a  $\pi/1:\mathbb{N}[n]$  tömböket, amik a gráf csúcsainak a  $d$  és  $\pi$  értékeit tartalmazzák. (A gráf csúcsait az  $1..n$  sorszámokkal azonosítjuk.)

Tegyük fel, hogy a start csúcsból elérhető a  $v$  csúcs! Írja meg az **útkiíró**( $d, \pi, v$ ) **rekurzív** függvényt, ami a  $d[v]$  értékkel tér vissza, és kiírja a start csúcsból a  $v$  csúcsba vezető optimális utat a következő formátumban: Az úton tetszőleges  $x$  csúcs „ $x:d$ ” alakban jelenik meg, ahol  $d$  az  $x$  csúcs  $d$ -értéke. minden él „ $--w-->$ ” alakban jelenik meg, ahol  $w$  az él súlya. Adott a kiír( $y$ ) eljárás, ami tetszőleges  $y$  számot vagy sztringet kiír.

Ha például a gráfnak három csúcsa van,  $d[1..3] = \langle 4; 9; 0 \rangle$  és  $\pi[1..3] = \langle 3; 1; 0 \rangle$ , akkor a startcsúcsból a 2 indexű csúcsba vezető optimális út a következő alakban jelenjen meg:       $3:0--4-->1:4--5-->2:9$

**2.** A Dijkstra algoritmus eredményeképpen megkaptuk a  $d/1:\mathbb{R}[n]$  és a  $\pi/1:\mathbb{N}[n]$  tömböket, amik a gráf csúcsainak a  $d$  és  $\pi$  értékeit tartalmazzák. (A gráf csúcsait az  $1..n$  sorszámokkal azonosítjuk.)

Tegyük fel, hogy a start csúcsból elérhető a  $v$  csúcs! Írja meg az **útkiíró**( $d, \pi, v$ ) **nemrekurzív** függvényt, ami a  $d[v]$  értékkel tér vissza, és kiírja a start csúcsból a  $v$  csúcsba vezető optimális utat a következő formátumban: Az úton tetszőleges  $x$  csúcs „ $x:d$ ” alakban jelenik meg, ahol  $d$  az  $x$  csúcs  $d$ -értéke. minden él „ $--w-->$ ” alakban jelenik meg, ahol  $w$  az él súlya. Adott a kiír( $y$ ) eljárás, ami tetszőleges  $y$  számot vagy sztringet kiír.

---

<sup>9</sup> $u - v_1, w_1; \dots v_k, w_k$ . azt jelenti, hogy a gráfban az  $u$  csúcs  $u$ -nál nagyobb indexű szomszédai  $v_1, \dots, v_k$ , és a megfelelő irányítlan élek súlyai  $w_1 = w(u, v_1), \dots, w_k = w(u, v_k)$ . (Ezzel a jelöléssel minden élet csak egyszer tüntetünk fel.)

Ha például a gráfnak három csúcsa van,  $d[1..3] = \langle 4; 9; 0 \rangle$  és  $\pi[1..3] = \langle 3; 1; 0 \rangle$ , akkor a startcsúcsból a 2 indexű csúcsba vezető optimális út a következő alakban jelenjen meg:       $3 : 0 -- 4 --> 1 : 4 -- 5 --> 2 : 9$

### 2.5.2. A QBF (Queue-based Bellman-Ford vagy Breadth-first Scanning) algoritmus

**1.a,** Milyen gráfokon és mit számol ki a *QBF* algoritmus? Adja meg a struktogramjait általános esetben!

**1.b,** Mit értünk a fenti program futásának *menetei* alatt? Mi a menetekhez kapcsolódó alapvető tulajdonság?

**1.c,** Adjon az algoritmus műveletigényére aszimptotikus *felső* becslést, és indokolja is állítását!

**1.d,** Honnét ismerhető fel, hogy van-e a gráfban a startcsúcsból elérhető negatív kör? Hogyan lokalizálható egy ilyen negatív kör?

**1.e,** Szemléltesse az algoritmus működését az alábbi gráfon, az **a** csúcsból indítva!<sup>10</sup> Grafikusan adja meg a végeredményt, a szokásos módon!

$$\begin{array}{ll} a \rightarrow b, 2 ; d, 4. & b \rightarrow c, -1 ; d, 1. \\ & d \rightarrow e, -2. \end{array} \quad \begin{array}{l} c \rightarrow a, -1 ; d, 2 ; e, 2. \\ \qquad \qquad \qquad e. \end{array}$$

**1.f,** Hogyan változik az algoritmus működése, ha a  $c \rightarrow a$  él súlya -2? (Minden más változatlan.) Értelmezze az így kapott végeredményt!

**2.**  $G[1..n]$  egy irányított, élsúlyozott, egyszerű gráf szomszédossági listás ábrázolása. A  $G[i]$  listák egyszerű láncolt listák,  $s \in 1..n$ .

**2.a,** Írjuk meg a  $QBF(G, s, d, \pi)$  függvényt, ami a fenti  $G[1..n]$  és  $s$  szerint  $O(n * m)$  műveletigénnel kiszámolja (a  $d$  és a  $\pi$  tömbökben, a szokásos ábrázolással) a legrövidebb utakat minden  $s$ -ből elérhető csúcsra, feltéve, hogy a QBF algoritmus előfeltétele teljesül, de azt az esetet is a tanultak szerint kezeli, amikor ez nem teljesül.

**2.b,** Mi a függvény fent említett előfeltétele?

**2.c,** Hogyan kezeli azt az esetet, amikor ez az előfelétel nem teljesül?

**2.d,** Hogyan értelmezhető az eredménye ez utóbbi esetben?

**2.e,** Mit tud mondani a hatékonyságáról ebben az esetben? Indokolja is az állítását!

### 2.5.3. DAG lerövidebb utak egy forrásból

**1.a,** Írja le röviden, szóban vagy struktogrammal, azt a tanult algoritmust, mellyel irányított, súlyozott, *körmentes* gráfokra a leghatékonyabb módon

---

<sup>10</sup> $u \rightarrow v_1, w_1; \dots v_k, w_k$ . azt jelenti, hogy a gráfban az  $u$  csúcs közvetlen rákövetkezői  $v_1, \dots, v_k$ , és a megfelelő irányított élek súlyai sorban  $w_1, \dots, w_k$ .

határozhatjuk meg a start csúcsból a többi csúcsba vezető legrövidebb utak fáját! (Negatív élköltségek is megengedettek.)

**1.b,** Mekkora a műveletigénye? Miért?

**1.c,** Szemléltesse a működését az alábbi gráfon,<sup>11</sup> ahol a csúcsok topologikus rendezése  $\langle a,b,c,d,e,f \rangle$ , és a „b” a startcsúcs! A legrövidebb utak fáját rajzolja is le!

$$a \rightarrow d, 2; f, 3. \quad b \rightarrow c, 2; e, 4. \quad c \rightarrow d, -1; e, 1. \quad d \rightarrow e, 2; f, 2. \quad e \rightarrow f, -2.$$

#### 2.5.4. Dijkstra algoritmusa

**1.a,** Milyen gráfokon és mit számol ki a *Dijkstra* algoritmus? Mekkora a műveletigénye  $n$  csúcsú gráf esetén, ha a prioritásos sort rendezetlen tömbbel reprezentáljuk? Miért?

**1.b,** Szemléltesse a működését az alábbi irányítatlan gráfon az **a** csúcsból indítva, az előadásról ismert módon!<sup>12</sup> Rajzolja le a legrövidebb utak fáját is, ami az eredményből adódik!

$$a - b, 2; c, 1; d, 4. \quad b - d, 0. \quad c - d, 2; e, 2. \quad d - e, 1. \quad e.$$

**2.a,** Milyen gráfokon és mit számol ki a Dijkstra algoritmus? Adja meg a struktogramját!

**2.b,** Mit értünk a gráfok élsúlyozott szomszédossági listás ábrázolása alatt? Mekkora az algoritmus futási ideje az előbbi gráfprezentáció és a prioritásos sor bináris kupaccal való megvalósítása esetén? Miért?

**2.c,** Milyen állítás igaz, amikor egy tetszőleges csúcsot kiválasztunk kiterjesztésre? Miért?

### 2.6. Legrövidebb utak minden csúcspárra

**1.a,** Milyen gráfokon és mit számol ki a Floyd-Warshall algoritmus? Mekkora a műveletigénye  $n$  csúcsú gráf esetén? Miért?

**1.b,** Szemléltessük a működését az alábbi irányított gráfon a  $(D^{(0)}, \Pi^{(0)}), \dots, (D^{(3)}, \Pi^{(3)})$  mátrix párok megadásával!<sup>13</sup>

**1.c,** Rajzoljuk le a legrövidebb utak fáit, amiket az eredményből kiolvashatunk!

$$1 \rightarrow 3, 1. \quad 2 \rightarrow 1, 0 ; 3, 2. \quad 3 \rightarrow 1, 1 ; 2, 2.$$

---

<sup>11</sup>  $u \rightarrow v_1, w_1; \dots v_k, w_k$ . azt jelenti, hogy a gráfban az  $u$  csúcs közvetlen rákövetkezői  $v_1, \dots v_k$ ; sorban  $w_1, \dots w_k$  súlyokkal.

<sup>12</sup>  $u - v_1, w_1; \dots v_k, w_k$ . azt jelenti, hogy a gráfban az  $u$  csúcs  $u$ -nál nagyobb indexű szomszédai  $v_1, \dots v_k$ , és a megfelelő irányítatlan élek súlyai sorban  $w_1, \dots w_k$ .

<sup>13</sup> Az „ $u \rightarrow v_1, w_1; \dots v_k, w_k$ .” formula azt jelenti, hogy a gráf  $u$  csúcsából kivezető élek  $(u, v_1), \dots (u, v_k)$ , sorban  $w_1 = w(u, v_1), \dots w_k = w(u, v_k)$  súlyokkal.

**2.**  $G[1..n]$  egy irányított, élsúlyozott, egyszerű gráf szomszédossági listás ábrázolása. A  $G[i]$  listák egyszerű láncolt listák.

**2.a,** Írjuk meg a FloydWarshall( $G, D, \Pi$ ) függvényt, ami a fenti  $G[1..n]$  szerint  $\Theta(n^2)$  műveletigénnyel inicializálja a Floyd-Warshall algoritmus  $D[1..n, 1..n]$  és  $\Pi[1..n, 1..n]$  mátrixait, majd  $\Theta(n^3)$  műveletigénnyel kiszámolja a legrövidebb utakat minden csúcspárra, feltéve, hogy a Floyd-Warshall algoritmus előfeltétele teljesül, de azt az esetet is a tanultak szerint kezeli, amikor ez nem teljesül.

**2.b,** Mi a függvény fent említett előfeltétele?

**2.c,** Hogyan kezeli azt az esetet, amikor ez az előfelétel nem teljesül?

**2.d,** Hogyan értelmezhető az eredménye ez utóbbi esetben?

**2.e,** Mit tud mondani a hatékonyságáról ebben az esetben? Indokolja is az állítását!

**3.a,** Milyen gráfokon és mit számol ki a *Floyd-Warshall* algoritmus?

**3.b,** Tegyük fel, hogy a gráfnak  $n$  csúcsa van! Mi a  $\langle (D^{(k)}, \Pi^{(k)}) : k \in 0..n \rangle$  mátrix-pár sorozat definíciója? Mi a rekurzív képlete?

**3.c,** Adja meg az algoritmus struktogramját, feltéve, hogy a gráf szomszédsági mátrixszal adott! Mekkora a műveletigénye  $n$  csúcsú gráf esetén?

**3.d,** Miért elegendő egyetlen  $(D, \Pi)$  mátrix-pár a programban?

**4.a,** Milyen gráfokon és mit számol ki a Floyd-Warshall algoritmus?

**4.b,** Mekkora a műveletigénye  $n$  csúcsú gráf esetén? Miért?

**4.c,** Szemléltesse a működését az alábbi irányítatlan gráfon a  $(D^{(0)}, \Pi^{(0)}), \dots, (D^{(4)}, \Pi^{(4)})$  mátrix párok megadásával!<sup>14</sup>

**4.d,** Melyik az alábbi gráf legkisebb részgráfja, ami az összes optimális utat tartalmazza?

1 – 2, 3; 3, 1; 4, 4. 2 – 4, 0. 3 – 4, 1. 4.

## 2.7. Irányított gráf tranzitív lezártja

**1.a,** Mit jelent egy gráf tranzitív lezártja, amit Warshall algoritmusa számol ki?

**1.b,** Tegyük fel, hogy a gráfnak  $n$  csúcsa van! Mi a  $\langle T^{(k)} : k \in 0..n \rangle$  mátrix sorozat definíciója? Mi a rekurzív képlete?

**1.c,** Adja meg az algoritmus struktogramját! Mekkora a műveletigénye  $n$  csúcsú gráf esetén? Miért elegendő egyetlen  $T$  mátrix a programban?

**1.d,** Mutassa be az algoritmus működését a

<sup>14</sup> $u - v_1, w_1; \dots v_k, w_k$ . azt jelenti, hogy a gráfban az  $u$  csúcs  $u$ -nál nagyobb indexű szomszédai  $v_1, \dots, v_k$ , és a megfelelő irányítatlan élek súlyai sorban  $w_1, \dots, w_k$ .

$4 \rightarrow 3.$     $3 \rightarrow 2.$     $2 \rightarrow 1$  ;  $4.$   
irányított gráfon!

### 3. Sztring keresés (Mintaillesztés)

#### 3.1. Quick-search

- 1.a,** Mit számol ki a *Quick Search* algoritmus?
- 1.b,** Mi az előnye, illetve hátránya a naiv mintaillesztő algoritmussal összehasonlítva?
- 1.c,** Mutassa be a *Quick Search* algoritmus előkészítő eljárásának működését az  $\{A, B, C, D\}$  ábécé-vel az  $ABACABA$  mintán, és
- 1.d,** e mintát illesztő eljárását az  $ABABACABACABADABACABABA$  szövegen!
- 1.e,** Mekkora az egyes eljárások aszimptotikus műveletigénye?
- 2.a,** Mit számol ki a *Quick Search* algoritmus?
- 2.b,** Mi az előnye, illetve hátránya – műveletigény szempontjából – a KMP algoritmussal összehasonlítva?
- 2.c,** Magyarázza el, mit tárolunk az algoritmus során a *shift* tömbben!
- 2.d,** Mutassa be a *Quick Search* algoritmus előkészítő eljárásának működését az  $\{A, B, C, D\}$  ábécé-vel az  $ADABABA$  mintán, és
- 2.e,** e mintát illesztő eljárását az  $ADABACACACABADABABADABABA$  szövegen!
- 2.f,** Mekkora az egyes eljárások aszimptotikus műveletigénye?

#### 3.2. Knuth-Morris-Pratt (KMP)

- 1.a,** Definiálja a *KMP* algoritmusban a keresett  $P[0..m)$  mintához tartozó  $\pi$  függvényt! Adja meg a  $\pi$  függvény két alapvető tulajdonságát, és ezeket bizonyítsa is be!
- 1.b,** Adja meg a  $\pi$  függvényt a definíciója alapján a  $BABAABAB$  mintán!
- 1.c,** Szemléltesse a *KMP* algoritmus működését, ahogy a fenti minta előfordulásait keresi az  $ABABABAABABAABABABAABABBABA$  szövegen!
- 2.a,** Definiálja a *KMP* algoritmus  $\pi$  függvényét, majd szemléltesse a *KMP* algoritmus  $\text{init}(\pi, P)$  eljárásának működését az  $ABABADA$  mintán!
- 2.b,** Szemléltesse *KMP* algoritmussal e minta előfordulásainak keresését az  $ABABADABABADABABADABABA$  szövegen!

- 2.c,** Mi a  $\pi$  függvény szerepe a keresés során?
- 2.d,** Mi a *KMP* algoritmus előnye, illetve hátránya – műveletigény szempontjából – a naiv mintaillesztő algoritmussal összehasonlítva?
- 3.a,** Milyen feladatot old meg a *Knuth-Morris-Pratt (KMP)* algoritmus?
- 3.b,** Definiálja a  $\pi$  függvényt, majd adja meg a *BABABAB* mintán!
- 3.c,** Szemléltesse *KMP* algoritmussal e minta előfordulásainak keresését a *BABBABABABABBABABAAB* szövegen!
- 3.d,** Mi a  $\pi$  függvény szerepe a keresés során?
- 3.e,** Mi a *KMP* algoritmus előnye, illetve hátránya a *Quick-search* mintaillesztő algoritmussal összehasonlítva?
- 4.a,** Milyen feladatot old meg a Knuth-Morris-Pratt (KMP) algoritmus?
- 4.b,** Szemléltesse a KMP algoritmus  $\text{init}(\pi, P)$  eljárásának működését az *ABACABA* mintán és
- 4.c,** e mintát illesztő eljárását az *ABABACABACABACABABA* szövegen!
- 4.d,** Mekkora az egyes eljárások műveletigénye?
- 4.e,** Mi KMP algoritmus előnye, illetve hátránya a Quick-search mintaillesztő algoritmussal összehasonlítva?

## 4. Tömörítés

### 4.1. Huffman kód

- 1.a,** Szemléltesse a *Huffman kódolás* működését az *ÁBRÁBANÁBRA* szövegen! Adja meg a kódfát és a szótárat! Mekkora a *Huffman kódolással* tömörített kód hossza?
- 1.b.,** Mekkora lenne a tömörített kód *fix hosszú* karakterkódok esetén?
- 1.c,** Hogyan dekódolható egy *Huffman kóddal* tömörített szöveg?
- 1.d,** Milyen értelemben optimális a *Huffman kód*? Azt jelenti-e ez, hogy a *Huffman kódolás* a lehető legjobb tömörítés? Miért?
- 2.a,** Az *ELEVEJELESRERENDELVE* szövegen szemléltessük a *Huffman kódolás* működését! Adjuk meg a kódfát és a szótárat!
- 2.b,** Mekkora a *Huffman kódolással* tömörített kód hossza?
- 2.c,** Mekkora lenne a tömörített kód *fix hosszú* karakterkódok esetén?
- 2.d,** Hogyan dekódolható egy *Huffman kóddal* tömörített szöveg?
- 2.e,** Milyen értelemben optimális a *Huffman kód*? Azt jelenti-e ez, hogy a *Huffman kódolás* a lehető legjobb tömörítés? Miért?

## 4.2. LZW tömörítés

**1.a,** Adott az  $\{A, B, C\}$  ábécé. Szemléthesse a Lempel–Ziv–Welch (LZW) tömörítő algoritmus működését a  $CBABABABCBABABAB$  szövegen, majd a megfelelő kitömörítő algoritmusét az 1, 2, 3, 4, 6, 5, 9, 7, 11 kódon! Mindkét esetben adja meg a generált szótárat, és a tömörítetlen szövegen a részszavak és a kódok megfeleltetését!

**1.b,** Hogyan kezeli a kitömörítő algoritmus azt az esetet, amikor nem találja meg a szótárban az aktuális kódhoz tartozó sztringet?

**2.a,** Az  $\{A, B\}$  ábécével szemléthesse a Lempel–Ziv–Welch (LZW) tömörítő algoritmus működését az  $ABABABABA$  szövegen, majd a megfelelő kitömörítő algoritmusét az 1, 2, 2, 3, 6, 4, 7 kódon! Mindkét esetben adja meg a generált szótárat, és a tömörítetlen szövegen a részszavak és a kódok megfeleltetését!

**2.b,** Milyen értelemben optimális a Huffman kód? Hogyan lehetséges, hogy az LZW algoritmus a gyakorlatban gyakran gyorsabban és jobban tömörít?