Algoritmusok és adatszerkezetek II. régebbi vizsgakérdések.

Ásványi Tibor – asvanyi@inf.elte.hu

2019. december 10.

Az eljárásokat és függvényeket megfelelően elnevezett és paraméterezett struktogramok segítségével adjuk meg! Ne feledkezzünk meg a skalár típusú, cím szerint átvett paraméterek szükség szerinti jelöléséről sem! A változókat alapértelmezésben a struktogramra vonatkozóan lokálisnak tekintjük.

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

A vizsgákon négy feladat van: kb. egy kérdés az első fejezetből (AVL fák, általános fák, B+ fák), kb. két kérdés a gráfalgoritmusokból és kb. egy kérdés az utolsó két fejezetből (sztring keresés, tömörítés).

Mindegyik feladat 25 pontot ér. A ponthatárok:
85 → jeles ; 70 → jó ; 55 → közepes ; 40 → elégséges.

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 az
{ [(2) 3 (4 {5})] 7 [(8) 9] } AVL fa. Rajzolja le a fát a csúcsok egyensúlyaival együtt! Szemléltesse 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, Szemléltesse 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 struktogramját, ami a $t \neq \oslash$ 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 $MT(n) \in \Theta(\lg n)$? 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 hatékonysága 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 t : 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 Fibonnacci fa-e! A függvény a fát ne változtassa meg!

1.2. Általános fák

1.a, Rajzolja le az (1 (2 (5)) (3) (4 (6) (7))) általános fa binárisan láncolt reprezentációját!
1.b, 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!

2.a, Rajzolja le az (1 (2) (3 (4 (5)) (6 (7) (8)) (9))) általános fa binárisan láncolt reprezentációját!
2.b, A t pointer egy binárisan láncolt általános fát azonosít. Írja meg a height(t) függvényeljárást, ami kiszámolja a t fa magasságát, $\Theta(n)$ műveletigénnyel és $O(h)$ tárigénnyel, 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 helyezkenek 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 kulcshalmaz 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 kulcshalmazt tároltunk! 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. Gráfreprezentációk

1. A $C[1..n, 1..n]$ bitmátrix egy irányított gráf szomszédossági mátrixos ábrázolása. Írja meg a **transzformál**(C, n, G) eljárást, ami előállítja a $G[1..n]$ gráfot, ami a $C[1..n, 1..n]$ 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 $G[1..n]$ egy irányított gráf szomszédossági listás ábrázolása. Írja meg a **transzponál**(G, n, GT) eljárást, ami előállítja a $GT[1..n]$ gráfot, ami a $G[1..n]$ gráf transzponáltjának szomszédossági listás reprezentációja! A szomszédossági listák egyszerű láncolt listák (fejelem nélküli, nemciklikus, egyirányú listák). $MT(n, m) \in \Theta(n + m)$, ahol m a gráf éleinek száma.

3. $G[1..n]$ egy irányított gráf szomszédossági listás ábrázolása. A $G[i]$ listák egyszerű láncolt listák (fejelem nélküli, egyirányú, nemciklikus listák). Adja meg a listaelem típus leírását! Írja meg a **kibeFokok**(G, n, be, ki) eljárást, ami minden $u \in 1..n$ csúcsra a $be[u]$-ban kiszámítja a csúcs bemeneti fokszámát, $ki[u]$-ban pedig a kimeneti fokszámát! $MT(n, m) \in \Theta(n + m)$, ahol m a gráf éleinek száma.

2.2. Szélességi keresés

1.a, 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 **a − b ; d. b − c ; d. c − e. d − e.** irányítatlan gráfon![1] Rajzolja le az eredményül adódó szélességi fát is!

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 tételt! **1.b,** Mutassa be az alábbi gráf [2] csúcsai topologikus rendezésének a gráf mélységi bejárására épülő algoritmusát! (Az algoritmus szemléltetése során a nemdeterminisztikus esetekben mindig 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 ki ez a fent szemléltetett algoritmus végrehajtása során? **1.d,** Mit tud a mélységi bejárásra épülő topologikus rendezés műveletigényéről?

[1] $u - v_1; \dots v_k.$ 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.)

[2] $u \to v_1; \dots v_k.$ azt jelenti, hogy az u csúcs közvetlen rákövetkezői $v_1, \dots v_k$.

(Indokolja is az állítást!)
a → b; d. b → c; d. c → e. d → e. f → c; e.

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 tételt! **2.c,** Szemléltesse a *Mélységi keresést* az alábbi irányított gráfon[3] úgy, hogy nemdeterminisztikus esetekben mindig a kisebb indexű csúcsot részesítse előnyben! Jelölje a bejárás során a különböző éltípusokat is!

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

2.4. Minimális feszítőfák

2.4.1. Kruskal algoritmusa

1.a, Mit számol ki a *Kruskal* algoritmus? **1.b,** Szemléltesse a működését az alábbi gráfon![4] (Elég az „él – feszítő erdő" párosok sorozatát megadni.) **1.c,** Mondja ki a biztonságos élekről és a minimális feszítőfákról szóló tételt! Definiálja a tételben szereplő *vágás* és *könnyű él* fogalmakat! **1.d,** Hogyan következik a *Kruskal* algoritmus helyessége ebből a tételből? **1.e,** Mekkora az algoritmusnak az előadásról ismert műveletigénye, és milyen feltételekkel?

a – b, 0 ; d, 1. b – c, 5 ; d, 2 ; e, 3. d – e, 4.

2.a, Mit számol ki a *Kruskal* algoritmus? **2.b,** Szemléltesse a működését az **a – b, 3 ; d, 1. b – c, 5 ; d, 2 ; e, 3. c – e, 4. d – e, 0.** gráfon![5] (Elég az „él – feszítő erdő" párosok sorozatát megadni.) **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, Mit számol ki a Prim algoritmus? Definiálja a súlyozott szomszédossági csúcsmátrix fogalmát! **1.b,** Csak *tömb* adatszerkezeteket használva adja meg a Prim(C, r, d, π) 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 segédeljárásokat is részletezze! Az eredményt, a csúcsok

[3] $u \to v_1; \dots v_k$. azt jelenti, hogy az u csúcs közvetlen rákövetkezői $v_1, \dots v_k$.

[4] 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 $w_1, \dots w_k$.

[5] 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 $w_1, \dots w_k$.

d és a π értékeit a $d[1..n]$ és a $\pi[1..n]$ vektorokban kapjuk. $T(n) \in O(n^2)$, $M(n) \in O(n)$

2.a, 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.** irányítatlan gráfon, a **d** csúcsból indítva![6]. **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 $d[1..n]$ és a $\pi[1..n]$ tömböket, amik a gráf csúcsainak d és π értékeit tartalmazzák. Írjuk meg az **útkiíró**(d, π, 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 ehhez a kiír(y) eljárás, ami tetszőleges y paramétert 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`

2.5.2. Sor-alapú Bellman-Ford algoritmus

1.a, Mit számol ki a *Sor-alapú (Queue-based) Bellman-Ford* algoritmus? Adja meg a struktogramját! **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, a **b** csúcsból indítva![7]

b → c, 2 ; e, 4. c → d, -1 ; e, 1. d → b, -1 ; e, 2 ; f, 2. e → f, -2.

[6] $u - v_1, w_1; \dots v_k, w_k$. azt jelenti, hogy a gráfban az u csúcs u-nál nagyobb indexű szomszédai $v_1, \dots v_k$, és a megfelelő irányítatlan élek súlyai $w_1 = w(u, v_1), \dots w_k = w(u, v_k)$. (Ezzel a jelöléssel minden élet csak egyszer tüntetünk fel.)

[7] $u \to v_1, w_1; \dots v_k, w_k$. azt jelenti, hogy a gráfban az u csúcs közvetlen rákövetkezői $v_1, \dots v_k$, és a megfelelő irányított élek súlyai sorban $w_1, \dots w_k$.

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 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, [8] ahol a csúcsok topologikus rendezése ⟨a,b,c,d,e,f⟩, és a „b" a startcsúcs! A legrövidebb utak fáját rajzolja is le!

a → d, 2; f, 3. b → c, 2; e, 4. c → d, -1; e, 1. d → e, 2; f, 2. e → f, -2.

2.5.4. Dijkstra algoritmusa

1.a, 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![9]. Rajzolja le a legrövidebb utak fáját is, ami az eredményből adódik!

a – b, 2; c, 1; d, 4. b – d, 0. c – d, 2; e, 2. d – e, 1. e.

2.a, 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 gráfreprezentáció 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, 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 $(D^{(0)}, \Pi^{(0)}), \ldots, (D^{(3)}, \Pi^{(3)})$ mátrix párok megadásával![10]. **1.c,** Rajzoljuk le a legrövidebb utak fáit, amiket az eredményből kiolvashatunk!

1 → 3, 1. 2 → 1, 0 ; 3, 2. 3 → 1, 1 ; 2, 2.

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

[8] $u \to v_1, w_1; \ldots v_k, w_k$. azt jelenti, hogy a gráfban az u csúcs közvetlen rákövetkezői $v_1, \ldots v_k$; sorban $w_1, \ldots w_k$ súlyokkal.

[9] $u - v_1, w_1; \ldots v_k, w_k$. azt jelenti, hogy a gráfban az u csúcs u-nál nagyobb indexű szomszédai $v_1, \ldots v_k$, és a megfelelő irányítatlan élek súlyai sorban $w_1, \ldots w_k$.

[10] Az „$u \to v_1, w_1; \ldots v_k, w_k$." formula azt jelenti, hogy a gráf u csúcsából kivezető élek $(u, v_1), \ldots (u, v_k)$, sorban $w_1 = w(u, v_1), \ldots w_k = w(u, v_k)$ súlyokkal.

FloydWarshall(G, D, Π) eljárást, ami $G[1..n]$ szerint $\Theta(n^2)$ műveletigénnyel inicializálja 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! .

3.a, 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 szomszédossági mátrixszal adott! Mekkora a műveletigénye n csúcsú gráf esetén? **3.d,** Miért elegendő egyetlen (D, Π) mátrix-pár a programban?

4.a, 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 $(D^{(0)}, \Pi^{(0)}), \ldots, (D^{(4)}, \Pi^{(4)})$ mátrix párok megadásával![11]. **4.d,** Melyik az alábbi gráf legkisebb részgráfja, ami az összes optimális utat tartalmazza?

$1 - 2, 3; 3, 1; 4, 4. \quad 2 - 4, 0. \quad 3 - 4, 1. \quad 4.$

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 $\mathbf{4 \to 3.} \quad \mathbf{3 \to 2.} \quad \mathbf{2 \to 1 \ ; 4.}$ 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 **(a)** 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?

[11] $u - v_1, w_1; \ldots v_k, w_k.$ azt jelenti, hogy a gráfban az u csúcs u-nál nagyobb indexű szomszédai $v_1, \ldots v_k$, és a megfelelő irányítatlan élek súlyai sorban $w_1, \ldots w_k$.

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

3.2. Knuth-Morris-Pratt (KMP)

1.a, Definiálja a *KMP* algoritmusban a keresett $P[1..m]$ mintához tartozó $next$ függvényt (*nem* a struktogramot)! Adja meg a $next$ függvény három alapvető tulajdonságát, és kettőt bizonyítson is be! **1.b,** Adja meg a $next$ függvényt a definíciója alapján a $BABAABAB$ mintán! **1.c,** Szemléltesse a *KMP* algoritmus működését, ahogy a fenti minta előfordulásait keresi az $ABABABAABABAABABABAABABBABA$ szövegben!

2.a, Definiálja a *KMP* algoritmus $next$ függvényét (nem a struktogramot), majd adja meg az $ABABADA$ mintán! **2.b,** Szemléltesse *KMP* algoritmussal e minta előfordulásainak keresését az $ABABADABABADABABABADABADABA$ szövegben! **2.c,** Mi a $next$ függvény szerepe a keresés során? **2.d,** Mi a *KMP* algoritmus előnye, illetve hátránya – műveletigény szempontjából – a *Quick-search* mintaillesztő algoritmussal összehasonlítva?

3.a, Milyen feladatot old meg a *Knuth-Morris-Pratt (KMP)* algoritmus? **3.b,** Definiálja a $next$ függvényt (nem a struktogramot), majd adja meg a $BABABAB$ mintán! **3.c,** Szemléltesse *KMP* algoritmussal e minta előfordulásainak keresését a $BABBABABABABBABABABAAB$ szövegben! **3.d,** Mi a $next$ 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,** Szemléltessük a KMP algoritmus init($next\ldots$) eljárásának működését az $ABACABA$ mintán és **4.c,** e mintát illesztő eljárását az $ABABACABACABABACABABA$ 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. Szemléltesse 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 szemléltesse 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?