# GRÁFOK ÉS ALGORITMUSOK ELMÉLETE – VIZSGAKÉRDÉSEK

## Matematika BSc Elemző szakirány II. év 1. félév

*Az írásbeli vizsgán öt kérdést kell kidolgozni, A kérdések az alábbiak közül kerülnek kiválasztásra, a műfaji változatosság figyelembe vételével. A feladatokban szereplő adatok változhatnak. Algoritmusok leírására egyaránt használható pszeudokód és struktogram. A vizsgakérdések döntő hányada megválaszolható a honlapomon, az első fül alatt elérhető jegyzet (http://people.inf.elte.hu/fekete/algoritmusok_jegyzet/) alapján, kisebb részükre az előadásokon hangzott el a válasz.*

1.) Írja le a Hanoi tornyai problémát, adja meg a megoldó algoritmusát. Adja meg a lépésszám rekurzív egyenletét, az egyenlet megoldását, és bizonyítsa annak helyességét.

2.) Definiálja a helyes zárójelezés fogalmát két különböző módon. Adja meg a helyes zárójelezés „programozási tételét” rövid szöveges magyarázattal kísérve. Az algoritmus, amelynek szekvenciális inputja egy helyes zárójelezés karaktersorozata, az outputra kiírja az összetartozó zárójelpárok sorszámát. (A sorszám-párok sorrendje nincs előírva.)

3.) Írja meg a *Verembe* (*V* [1..*n*], *k*, *d*, *hiba*), a *Veremből* (*V* [1..*n*], *k*, *x*, *hiba*) műveletet a verem tömbös ábrázolása esetén. A *k* paraméter a verem felső elemének indexe ($1 \le k \le n$), illetve értéke nulla, ha a verem üres. Az első művelet esetén a *d* értéket helyezzük a verembe, ha még nincs tele a verem. A második művelet esetén a kivett értéket az *x*-be helyezzük, feltéve, hogy a verem nem üres. Ne feledjük el a hiba változót az egyes műveletek sikerének megfelelően beállítani.

4.) Az *l* pointer egy egyszerű (fejlem nélküli) listának az első elemére mutat. Fordítsuk meg az elemek sorrendjét helyben, segédmemória használata nélkül!

5.) Az *l* pointer egy fejelemes egyirányú lista fejelemére mutat. Az *akt* pointer a lista aktuális elemére mutat, vagy pedig NIL az értéke (éppen nincs aktuális elem értelmezve). Ha a lista üres, akkor *akt* értéke NIL. Írja, meg a *Töröl* (*l, akt, r*) eljárást, amely a lista aktuális elemét törli! Az *r* pointer típusú változóban adjuk vissza a „kiláncolt” elem mutatóját, illetve NIL-t, ha a törlés nem hajtható végre. Az *akt* változó a törölt elem utáni elemre mutasson a művelet sikeres végrehajtása után.

6.) Egy bináris fának 10 csúcsa van, azonosítóik: *A*, *B*, …, *J*. A fa gyökere *A*; a többi csúcs egy szülő bal vagy jobb gyereke: *B=bal*(*A*), *C=jobb*(*A*), *D=bal*(*B*), *E=jobb*(*B*), *F=jobb*(*C*), *G=bal*(*E*), *H=bal*(*F*), *I=jobb*(*F*) és *J=jobb*(*G*). Rajzolja le a bináris fát és írja le a preorder, inorder és posztorder bejárás eredményt! Adja meg az inorder bejárás ADS-szintű rekurzív algoritmusát!

7.) Adott a *t* pointer által mutatott gyökeres bináris fa, amely lehet üres is. A fa egy csúcsát olyan összetett adatelem ábrázolja, amely az érték mellett a bal és jobb gyerek mutatóját tartalmazza. Írja meg azt az eljárást, amely (egy pointer alaptípusú sor adatszerkezet használatával) szintfolytonos sorrendben írja ki a fa csúcsaiban található értékeket!

8.) Definiálja a kupac ADS-szintű fogalmát. Adja meg a tömbös ábrázolás módját és a szülő – gyerek indexfüggvényeket, minkét irányban! Rajzoljon fel egy (nem triviális kitöltésű) példát $n = 12$ esetére. Adja meg a rajzon illusztrálva, szöveges magyarázat kíséretében a kupaccal megvalósított elsőbbségi sor *törlő* és *beszúró* műveletének működését.

9.) Adja meg a bináris keresés két algoritmusát. Az *A*[1..n] rendezett tömbben keresünk egy megadott elemet. Az első esetben „három-ágú” elágazást használunk, míg a második esetben „két-ágú” elágazással végigkeressük a tömböt (mint a barkochbában) és a végén rákérdezünk arra az elemre, amelyhez a keresés jutott.

10.) Írja le a bináris keresőfa fogalmát és legalapvetőbb tulajdonságát! Építsen bináris keresőfát a következő adatokból: 60, 30, 40, 110, 10, 50, 90, 70, 120, 80, 100, 20! Adj meg a *Min* (*t*) eljárást iteratív változatban, amely a *t* pointer által mutatott bináris keresőfában megkeresi a legkisebb kulcsértékű rekordot. Ha a fa üres, akkor NIL-t ad vissza az eljárás, egyébként pedig a megtalált csúcs pointere a visszaadott érték. Hogyan működik ezen a fán az eljárás?

11.) Írja le a bináris keresőfa fogalmát és legalapvetőbb tulajdonságát! Építsen bináris keresőfát a következő adatokból: 60, 30, 40, 110, 10, 50, 90, 70, 120, 80, 100, 20! Magyarázza el a K*övetkező* (*t*, *p*) művelet működésének elvét. Ez az eljárás a *t* pointer által mutatott bináris keresőfában a *p* mutatóval címzett elem érték szerinti rákövetkezőjének a pointerét adja vissza! (A *p* mutató értéke garantáltan helyes.) Ha a rákövetkező nem létezik, akkor a visszaadott mutató értéke NIL. A magyarázatban arra a három esetre térjen ki, amikor a p pointer rendre a 60, az 50, illetve a 120 kulcsértéket tartalmazó csúcsra mutat.

12.) Írja le a bináris keresőfa fogalmát és legalapvetőbb tulajdonságát! Építsen bináris keresőfát a következő adatokból: 40, 110, 30, 60, 90, 10, 50, 20, 120, 80, 70, 100! Hajtsa végre a 60-as érték törlését, a tanult algoritmus alapján! Rajzolja le a bináris keresőfa törlés utáni állapotát! Magyarázza el az alkalmazott eljárást!

13.a) Építsen AVL-fát a következő adatokból: 80, 50, 100, 30, 110, 60, 10, 90, 40, 70, 20! Ha elromlott a fa kiegyensúlyozása, lássa el címkékkel a csúcsokat és állítsa helyre az AVL-tulajdonságot a megfelelő forgatással! Adja meg az alkalmazott forgatás általános sémáját is!

b) Építsen AVL-fát a következő adatokból: 40, 20, 90, 60, 30, 100, 80, 110, 10, 50, 70! Ha elromlott a fa kiegyensúlyozása, lássa el címkékkel a csúcsokat és állítsa helyre az AVL-tulajdonságot a megfelelő forgatással! Adja meg az alkalmazott forgatás általános sémáját is!

14.) Adja meg annak a transzformációjának a sémáját (a magasságértékek feltüntetésével), amely az AVL-fa (++,-)-os típusú, beszúrást követő „elromlását” helyreállítja! Mutassa meg, hogy a forgatások utáni fa (i) keresőfa, (ii) AVL-fa és (iii) ennek magassága megegyezik a beszúrás előtti magassággal.

15.) Írja meg a buborékrendezés algoritmusát. Adja meg az összehasonlítások (Ö(*n*)) számát és a cserék minimális (mCs(*n*)) és maximális (MCs(*n*)) számát.

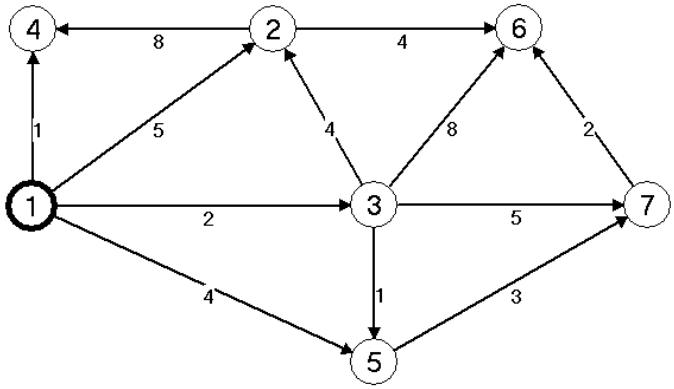
16.) Írja le a maximumkiválasztó rendezés algoritmusát úgy, hogy a maximumkiválasztást is megadja. Mennyi az összehasonlítások (Ö(*n*)) száma? Mennyi az elemekre vonatkozó értékadások (mozgatások) minimális és maximális (mM(*n*) és MM(*n*)) száma, ha egy csere három értékadásnak számít?

17.) Adja meg a tömbös beszúró rendezés algoritmusát! Határozza meg, hogy hány összehasonlítást (Ö(*n*)) és elemmozgatást (M(*n*)) végez az eljárás a legkedvezőbb, illetve a legkedvezőtlenebb esetben, adott *n* hosszúságú tömbre (mÖ(*n*), mM(*n*), MÖ(*n*), MM(*n*))?

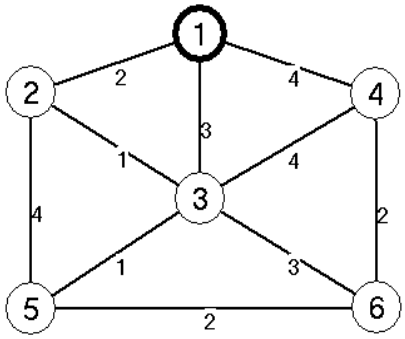
18.) Adja meg a beszúró rendezés algoritmusát fejelemes listára. Az l pointer a lista fejelemére mutat. Fűzzön magyarázatot az eljáráshoz! Milyen stratégiát alkalmaz az eljárás a lista egészének a feldolgozásában?

19.) Hajtsa végre a kupacrendezést egy 6-elemű ábrasorozaton. 1. ábra: a kupac bináris fája szintfolytonosan feltöltve a 7, 10, 4, 3, 2, 12, 5,1, 11, 6, 8, 9 számokkal. 2-3-4. ábra: a kupac szintenkénti kialakítása. A 4. ábra már a kész kupacot mutatja, és ezen jelölje be, hogy hogyan kerül a maximális elem a helyére. Az 5. ábrán a maximális elem már a rendezés szerinti helyén van; jelölje be ezen a következő legnagyobb elem helyrevitelének lépéseit. A 6. ábrán már a két legnagyobb elem a végleges helyére került; végül jelölje be a következő elemmel kapcsolatos lépéseket.

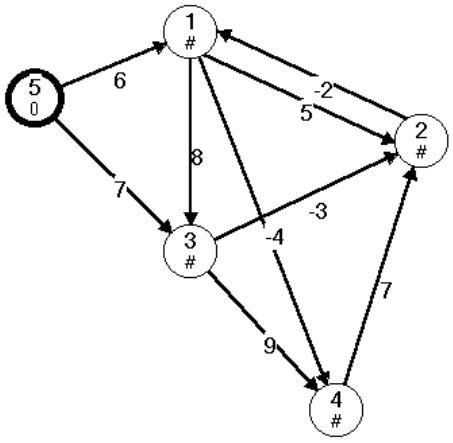
20.) Szemléltesse a Dijkstra algoritmus működését az alábbi gráfon! Használja erre az ábrát, egyértelmű jelöléseket alkalmazva (a csúcs kiválasztásának sorszáma; a költségek sora, közülük az utolsó kiválasztáskor bekeretezve; szülő pointerek, ahol végül egy érvényes)! A kiinduló csúcs legyen az 1-es címkéjű csúcs. A feszítőfát tüntesse ki a rajzon!



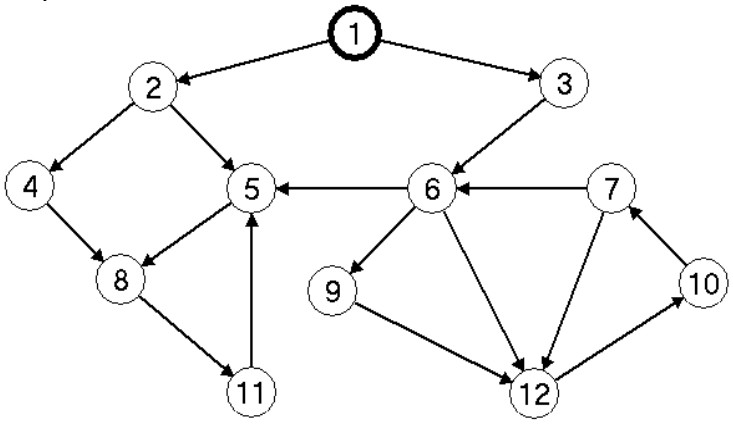

21.) Szemléltesse a Prim algoritmus működését az alábbi gráfon! Használja erre az ábrát, egyértelmű jelöléseket alkalmazva (a csúcs kiválasztásának sorszáma; a költségek sora, közülük az utolsó kiválasztáskor bekeretezve; szülő pointerek, ahol végül egy érvényes)! A kiinduló csúcs legyen az 1-es címkéjű csúcs. A feszítőfát tüntesse ki rajzban!



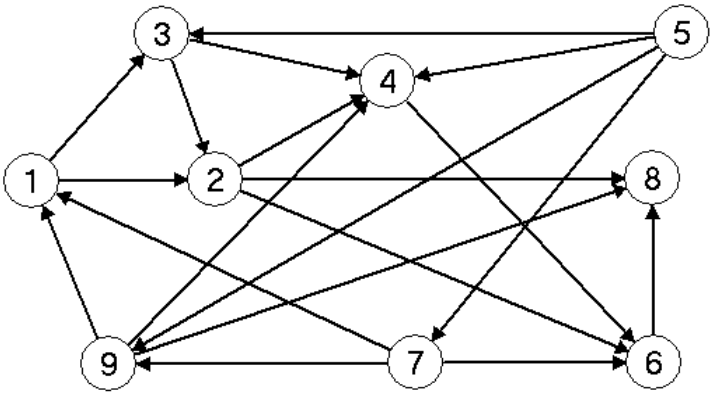

22.) Szemléltesse a Bellman-Ford algoritmus működését az alábbi gráfon! Startcsúcs az 5-ös címkéjű csúcs. A gráfban nincs „negatív kör”, ezt már nem kell ellenőrizni. Elég, ha összesen három ábrát vesz fel. A kiinduló ábrán szemléltetheti a kezdő költségértékek megadását és az első iterációs menet hatását. A második és a harmadik ábrán a második és harmadik iteráció eredményét szemléltesse, majd elég azt leírni, hogy mi lesz a negyedik iteráció hatása az eredményre.



23.) Az alábbi gráfon az 1-es címkéjű csúcsból indulva lefuttatunk egy mélységi bejárást. A csúcsok szomszédjainak a feldolgozási sorrendje legyen a szomszéd csúcsok címkéi szerint növekvően rendezett. Jelölje a gráfon, a bejárás során kapott mélységi és befejezési számokat, illetve az éltípusokat!




24.) Adja meg az alábbi gráf egy topologikus rendezését a mélységi bejárást és vermet alkalmazó eljárással! A mélységi bejárás végrehajtását szemléltesse az alábbi ábrán! A bejárás induljon az 1-es csúcsból és a rákövetkező csúcsot válassza mindig a lexikografikus rendezésnek megfelelően. Ha visszajut a kiinduló pontba, akkor folytassa a legkisebb sorszámú, még nem bejárt csúccsal!




25.) Adjon olyan algoritmust, amely meghatározza egy irányított gráf éllistás ábrázolása esetén a gráf csúcsainak kimeneti fokát és bemeneti fokát. A program ezeket az értékeket a *kifok* [1..*n*] és *befok* [1..*n*] tömbökbe írja be, amelyeket a csúcsok sorszámával indexeltünk.

26.) Írja le a gráfok szélességi bejárásának stratégiáját és adja meg az algoritmust is, a szükséges magyarázattal! Adja meg és indokolja az eljárás műveletigényét!

27.) Jellemezze a minimális költségű utak keresésének stratégiáját abban az esetben, ha a gráf élein csak nem-negatív költségek szerepelnek! Adja meg a Dijkstra-algoritmust, két változatát is. Az elsőben a költségek tömbjén történik a következő nem-kész (még nyitott) csúcs kiválasztása. A második, absztraktabb változatban elsőbbségi sor szerepel. Mennyi az algoritmus műveletigénye a tömbös esetben és mennyi akkor, ha az elsőbbségi sort kupaccal valósítjuk meg? Mit mondhatunk sűrű gráfok esetén a második változat költségére?

28.) Adja meg a Bellman-Ford algoritmust, és röviden írja le működésének elvét! Fogalmazza meg, hogy milyen feladat megoldására használható az algoritmus! (Mi teljesül a *k*-adik menet után?) Hogyan lehet az algoritmus végrehajtási idejét esetleg rövidíteni? Hogyan lehet az alkalmazhatóságnak a gráfra vonatkozó feltételét ellenőrizni?

29.) Írja le a gráfok mélységi bejárásának stratégiáját! Adja meg az algoritmust is, a szükséges magyarázattal. (Nem szükséges tételeket kimondani és bizonyítani.) Definiálja és illusztrálja kis példákkal a bejárás szerinti éltípusokat. Jelölje meg az algoritmusban azt a helyet, ahol ez megtörténhetne. Indokolja a mélységi bejárás műveletigényét!

30.) DAG topologikus rendezése. - Bizonyítsa be, hogy egy gráf pontosan akkor nem DAG (irányított, de nem körmentes), ha tetszőleges mélységi keresés talál benne visszaélt. - Adja meg egy gráf topologikus rendezésének definícióját! - Írja le és bizonyítsa a DAG tulajdonság és a topologikus rendezés kapcsolatáról szóló tételt és (még az előtt) - az ide tartozó lemmát (amely egy bizonyos tulajdonságú csúcs létezését állítja DAG-ban)! (A kérdésben nem szerepel a DAG topologikus rendezésének vermes algoritmusa!)

31.) Nyílt címzéssel, a $h(k) = k \ (mod\ 11)$ hasítófüggvénnyel helyezzük az $A[0..10]$ tömbben a 29, 61, 49, 50, 13, 39, 60, 76 sorozatot. Adjuk meg a keletkezett táblát az alábbi esetekben:
a) lineáris próbálás (balra lép)
b) négyzetes próbálás (felváltva jobbra-balra lép)
c) kettős hash-elés, ahol $h'(k) = k \ (mod\ 7) + 1$ (balra lép)

32.) Két módon is kézzel rendezünk névsorba 100 dolgozatot az asztalon.
(i) Beszúró rendezéssel visszük a helyére sorban a 2 – 100. dolgozatot pontosan úgy, ahogyan a tanult algoritmus működik, a dolgozatokat egyesével mozgatva. Egy dolgozat mozgatása 3 mp-et igényel. Mennyi a legrosszabb esetben a rendezés ideje (óra, perc, mp)?
(ii) Edényrendezéssel először „hasheljük" a dolgozatokat a hallgató nevének első betűje alapján. A létrejövő edények száma 20, és mindbe ugyanannyi (= 5) dolgozat jut. Az egyes edényekben (20 egymás utáni) beszúró rendezéssel állítjuk sorba az ott szereplő dolgozatokat. Utána összefűzzük az összes dolgozatot – mindet egyenként mozgatva – egyetlen rendezett sorba. Mennyi a rendezés ideje a legrosszabb esetben (óra, perc, mp), ha ismét 3 mp-et igényel egy dolgozat mozgatása?

33. a) Adott a háromjegyű bináris számok következő sorozata: 010, 101, 011, 110, 000, 001, 100. Rendezzük a számokat az egy tömbben dolgozó „RADIX előre" edényrendezéssel! Adjuk meg a tömb tartalmát az egyes pozíciók feldolgozása után!

b) Adott a háromjegyű bináris számok következő sorozata: 110, 011, 010, 101, 000, 100, 001. Rendezzük a számokat a felváltva két tömbben dolgozó „RADIX vissza" edényrendezéssel! Adjuk meg rendre az 1., 2. és 3. pozíció szerinti hasítás eredményét tartalmazó megfelelő (A vagy B) tömböt, végül pedig a rendezettséget mutató A tömböt!

c) Adott a három karakteres értékek következő sorozata: bcb, cba, aac, bbb, cab, acc, caa. Rendezzük a számokat a visszafelé haladó edényrendezés listás változatával! Adjuk meg a listák (absztrakt szinten sorozatok) tartalmát a szét- ill. összefűzések után!