Algoritmuselmélet 2. Zárthelyi dolgozat 2025. május 15.

A rendelkezésre álló idő: 90 perc.
Minden feladat egységesen 10 pontot ér. Az aláírás megszerzéséhez legalább 24 pont szükséges. A teljes pontszám eléréséhez a megoldás(oka)t indokolni kell!
Kérjük, írja fel a **nevét**, **NEPTUN-kódját** és a **gyakorlatvezető nevét** az összetűzött lapok jobb felső sarkába.

Általános alapelvek. A pontozási útmutató célja, hogy a javítók a dolgozatokat egységesen értékeljék. Ezért az útmutató minden feladat (legalább egy lehetséges) megoldásának főbb gondolatait és az ezekhez rendelt részpontszámokat közli. Az útmutatónak nem célja a feladatok teljes értékű megoldásának részletes leírása; a leírt lépések egy maximális pontszámot érő megoldás vázlatának tekinthetők. 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. Így például az anyagban szereplő ismeretek, definíciók, tételek puszta leírása azok alkalmazása nélkül nem ér pontot (még akkor sem, ha egyébként valamelyik leírt tény a megoldásban valóban szerephez jut). 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, részmegoldásért, amelyből a dolgozatban leírt gondolatmenet alkalmas kiegészítésével a feladat hibátlan megoldása volna kapható. Ha egy megoldó egy feladatra több, egymástól lényegesen különböző megoldást is elkezd, akkor legföljebb 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 dolgozatból nem derül ki, hogy a megoldó melyiket tartotta helyesnek, akkor a kevesebb pontot érő megoldáskezdeményt értékeljük (akkor is, ha ez a pontszám 0). Az útmutatóban szereplő részpontszámok szükség esetén tovább is oszthatók. Az útmutatóban leírttól eltérő jó megoldás természetesen maximális pontot ér, de bizonyítás nélkül csak az előadáson szereplő tételekre és állításokra lehet hivatkozni.

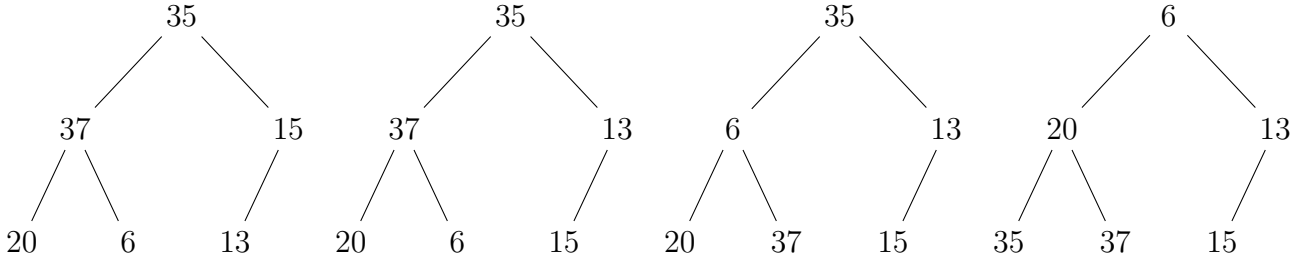
Az első 3 feladatban az algoritmusok lépéseinek bemutatásán kívül nem várunk további indoklást.

1. Építsen kupacot az órán tanult lineáris idejű módszerrel a közvetkező tömbből:

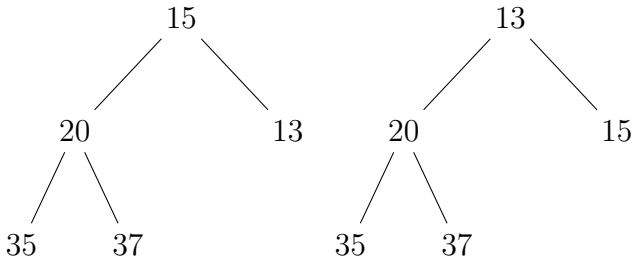
35	37	15	20	6	13

a) Ábrázolja a lépéseket *fa reprezentációban.*
b) A kapott kupacban végezzen el egy `MINTÖR` műveletet.

Megoldás: A kupacépítés lépései:



A `MINTÖR` lépései:



Pontozás:

A tömb fa alakja helyesen	**1 pont**
A kupacépítésnél alulról felfelé és jobbról balra végzi a kupacolt	**2 pont**
Jó fák	**4 pont**
A mintörnél jó elemet tesz a gyökérbe	**1 pont**
Jó végeredmény	**2 pont**

Ha a kupacot beszúrásokkal építi, akkor max **5 pont**

2. Nyitott címzésű hashelést használunk, a hash-tábla mérete $m = 11$, a hash-függvény pedig $h(k) = k \bmod 11$. Az ütközések kezelésére **lineáris próbát** alkalmazunk. A tábla állapota jelenleg a következő *(a $*$ a törölt jel)*:

0	1	2	3	4	5	6	7	8	9	10
44	56			16	38				*	21

a) Illessze be a következő két kulcsot a hash-táblába: $98, 33$. Minden beszúrás után ábrázolja a hash-tábla aktuális állapotát.
b) A kapott táblában végezze el a `TÖRÖL(44)` műveletet és ismét ábrázolja a hash-tábla aktuális állapotát.
c) Az így kapott táblában a `KERES(34)` során hány cellát vizsgál meg az algoritmus?

Megoldás: `BESZÚR(98)`: $h(98) = 10$, $*$ helyére be lehet szúrni

0	1	2	3	4	5	6	7	8	9	10
44	56			16	38				98	21

`BESZÚR(33)`: $h(33) = 0$

0	1	2	3	4	5	6	7	8	9	10
44	56			16	38			33	98	21

`TÖRÖL(44)`: $h(44) = 0$, a törölt jelet be kell tenni

0	1	2	3	4	5	6	7	8	9	10
*	56			16	38			33	98	21

`KERES(34)`: hány cellát vizsgál meg? $h(34) = 1$. A $*$-nál folytatni kell a keresést, és az első üres celláig kell menni.
Megvizsgált cellák száma: 6

Pontozás:

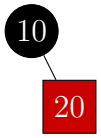
Jó próbasorozat	**1 pont**
Mindkét kulcs helyes beszúrása,	**2+2 pont**
Törlés helyes végrehajtása, $*$ beírása	**2 pont**
Keresés helyes lépései, vizsgált cellák száma	**2+1 pont**

3. Egy kezdetben üres piros–fekete fába sorban szúrja be a következő számokat: $10, 20, 30, 15, 18$. Minden egyes beillesztés után adja meg a fa állapotát (a csomópontok színeivel együtt). Ha forgatást alkalmaz, akkor ábrázolja a fát a forgatás előtt és után is.

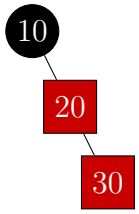
Megoldás: Az üres fekete levelek nélkül ábrázolva a fákat:

Beszúrás: 10

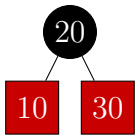
Beszúrás: 20



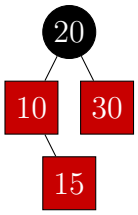
Beszúrás: 30 forgatás előtt



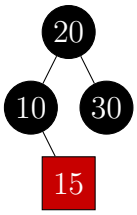
Beszúrás: 30 forgatás után



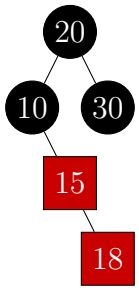
Beszúrás: 15 átszínezés előtt



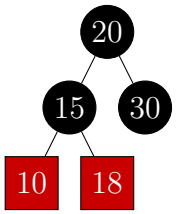
Beszúrás: 15 átszínezés után



Beszúrás: 18 forgatás előtt



Beszúrás: 18 forgatás után



Pontozás:

Az első két beszúrás (nem kell levonni, ha az első nincs külön ábrázolva)	**2 pont**
30 beszúrása	**3 pont**
15 beszúrása	**2 pont**
18 beszúrása	**3 pont**

Mindegy, hogy az üres levelek be vannak-e rajzolva.

4. Vegyük a következő *eldöntési problémát*:
Input: Egy egyszerű G gráf. **Kérdés:** Teljesül-e, hogy $\chi(G) \leq \omega(G)$?
$\chi(G)$ *a gráf kromatikus száma: minimum hány szín kell a csúcsok jó színezéséhez.*
$\omega(G)$ *a gráf klikkszáma: hány csúcsú a legnagyobb teljes részgráf.*
Mutassa meg, hogy a fenti probléma az **NP** osztályba tartozik.

Megoldás: Egy jó színezés k színnel és $\ell \geq k$ pontú klikk megfelelő tanú.	**4 pont**
Ha a válasz igen, akkor ilyenek biztosan vannak. *(Van olyan színezés, ami $\chi(G)$ színt használ és van olyan ponthalmaz, ami teljes részgráf, és amiben $\omega(G)$ pont van.)*	**1 pont**
Mindkettő triviálisan polinom méretű.	**1 pont**
Az ellenőrző algoritmus ellenőrzi:	
jó-e a színezés	**1 pont**
teljes részgráf-e a ponthalmaz	**1 pont**
$\ell \geq k$	**1 pont**
Ezek is triviálisan polinom időben végrehajthatóak.	**1 pont**

Megjegyzés: A BSZ tanulmányokból ismert, hogy $\chi(G) \geq \omega(G)$ minden gráfra teljesül, ezért minden megfelelő tanúnál $k = \ell$ lesz. De ez nem szükséges az indokláshoz.

5. Adott egy n elemű, csupa különböző számot tartalmazó S halmaz és egy k egész szám. Adjon algoritmust, amely kiírja az S halmaz k legkisebb elemét növekvő sorrendben. Ha $k \leq n/\log_2(n)$, akkor az algoritmus lépésszáma legyen $O(n)$.

Megoldás: Építsünk kupacot S elemeiből. **3 pont**
Végezzünk k db `MINTÖR`-t, sorban írjuk ki a minimumokat. **2 pont**
Kupacépítés elvégezhető $O(n)$ lépésben. **1 pont**
Egy `MINTÖR` elvégezhető $O(\log n)$ lépésben. **1 pont**
Az össz lépésszám $O(n) + k \cdot O(\log n)$. **1 pont**
Ha $k \leq n/\log_2(n)$, akkor ez $O(n)$. **2 pont**

6. Legyen a `2KÖR` probléma a következő:
Input: Egy $2v-1$ csúcsú irányítatlan G gráf.
Kérdés: Létezik-e a gráfban két olyan kör, hogy mindkettőnek a hossza v és pontosan egy közös csúcsuk van?

Adjon meg egy `H` $\prec$ `2KÖR` Karp-redukciót, ahol `H` a Hamilton-kör probléma.
(A redukciót elég legalább 3 pontú gráfokra megadni.)

Megoldás: Legyen $f(G)$ az a gráf, amit úgy kapunk G-ből, hogy egy tetszőleges x pontjához csatlakoztatunk egy v hosszú kört (v a G csúcsainak száma, a hozzáadott kör x-en kívüli többi pontja új pont) (*több más jó konstrukció is van*) **4 pont**
$f(G)$ polinom időben kiszámolható, ez triviális **1 pont**
Ha G-ben van Hamilton-kör, akkor ez és a hozzáadott új kör megfelelő részgráf $f(G)$-ben **2 pont**
Ha $f(G)$-ben van megfelelő részgráf, akkor van két v hosszú kör $f(G)$-ben, de csak az egyik tartalmazhat új pontot, a másik minden pontja G-ben van, tehát Hamilton-kör G-ben. **3 pont**

7. Algoritmisztán Birodalmában a *Törpék* és az *Óriások* külön nyilvántartást vezetnek a saját lakóikról, de mindkét helyen az `ÓriásTörpe` adatszerkezetben. A lakókat az adatszerkezetben a magasságuk azonosítja. Tudjuk, hogy minden törpe kisebb, mint bármelyik óriás és hogy nincs két egyforma magasság. Legyen n_1 a törpék, n_2 pedig az óriások aktuális létszáma.

Az `ÓriásTörpe` adatszerkezet rendelkezik a következő tulajdonságokkal:

- A *keresés*, *beszúrás* és *törlés* az adatszerkezetben $O(\log n)$ lépésben végrehajtható, ha n tárolt elemek száma.
- A két nyilvántartás egyesítése, vagyis az *unió* művelete is elvégezhető. Ilyenkor a két nyilvántartásból **egyetlen** `ÓriásTörpe` adatszerkezetben lévő nyilvántartást készítünk, amiben minden Törpe és Óriás is benne van. Ez a művelet legyen végrehajtható $O(\log(\max(n_1, n_2)))$ lépésben.

Adjon megfelelő `ÓriásTörpe` adatszerkezetet.
Az indoklásról ennél a feladatnál se feledkezzen el!

Megoldás: Tároljuk a (magasság, lakó) párokat egy 2-3-fában, ahol a kulcs a magasság. **2 pont**
A szokásos műveletek ebben elvégezhetőek $O(\log n)$ lépésben. **2 pont**
Az unió elvégzésénél tegyük fel, hogy a törpék fájának magassága nem nagyobb, mint az fájának magassága. A másik esetben hasonló az algoritmus.
Megvizsgáljuk a két fa magasságát (pl. a minimum keresésnél ennyi belső csúcsot vizsgálunk), legyen k a törpék, ℓ az óriások fájának magassága, $k \leq \ell$. **1 pont**
A gyökérből induljunk lefelé, mindig a legkisebb gyerek felé haladva jussunk el az $\ell - k$-adik belső csúcsig, ehhez vegyünk fel egy új gyereket, ami kisebb az eddigieknél és ez alá tesszük a törpék fáját, így minden törpe levél távolsága is ℓ lesz a gyökértől. **2 pont**
Ha így már 4 gyerek lenne, akkor a beszúrás algoritmusához hasonlóan csúcsvágásokkal elérhetjük, hogy egy 2-3-fát kapjunk. **1 pont**
Minden részfeladat lépésszáma $O(\ell)$, tehát összesen is $O(\ell) = O(\log(\max(n_1, n_2)))$ **2 pont**
Piros-fekete fával való próbálkozás esetén az első 4 pontot meg lehet adni.