

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 pusztán 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 legfőbb 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 leírtól eltérő jó megoldás természetesen maximális pontot ér, de bizonyítás nélkül csak az előadáson szereplő tételekre és állításokra lehet hivatkozni.

1. Legyen

$$f(n) = n^3 + 5n \log_2^2 n, \quad g(n) = n^3 \log_2 n.$$

Igazoljuk vagy cáfoljuk, hogy $f(n) = O(g(n))$.

Megoldás: A reláció igaz, azaz $n^3 + 5n \log_2^2 n \in O(n^3 \log_2 n)$.

Ehhez belátjuk, hogy létezik olyan c valós szám és n_0 egész szám, hogy $n \geq n_0$ esetén:

2 pont

$$f(n) = n^3 + 5n \log_2^2 n \leq c \cdot n^3 \log_2 n = c \cdot g(n).$$

Azt fogjuk belátni, hogy $c = 6$ és $n_0 = 2$ kielégíti a fenti feltételt.

2 pont

Ezután nézzük a következő egyenlőtlenségsort:

$$n^3 + 5n \log_2^2 n \leq 6n^3 \leq 6n^3 \log_2 n.$$

2+2 pont

Az első egyenlőtlenség azért igaz, mert $n \geq 2$ esetén $\log_2 n \leq n$,

1 pont

míg a második azért, mert $n \geq 2$ esetén $\log_2 n \geq 1$.

1 pont

2. Egy vállalat 6 karakter hosszú termékazonosítókat használ. Az azonosítók kizárólag a következő 8 nagybetűből állhatnak:

O, Z, A, B, E, M, S, T

A vállalatnál ezekre a karakterekre nem a szokásos ABC sorrend érvényes, hanem egy speciális, fontossági sorrend:

O < Z < A < B < E < M < S < T

Adott pontosan 8 darab, ilyen szabály szerint képzett, 6 karakter hosszú termékazonosító:

[BASTET, AMETSA, ZEMOBE, SOMBTA, AZAZAZ, MESOBE, TAMTOM, ZEBESA]

Rendezzük a fontossági sorrendet használva lexikografikus sorrendbe ezeket az azonosítókat a Radix rendezés segítségével. A megoldást részletesen fejtsük ki, azaz minden fázis után írjuk le a termékazonosítók aktuális sorrendjét.

Pontozás Ha tudja, hogy radix rendezésben oszloponként (karakterenként) kell rendezni.

4 pont

Ha tudja, hogy hátulról előre kell.

2 pont

Helyes számolás.

4 pont

Ha *sima ABC sorrendjét alkalmazza, max.*

8 pont

Megoldás: A radix rendezést a jobb szélső (6. karakter) pozíciótól kezdjük:

A karakterek rendezett sorrendje (a kisebb értékű előrébb kerül) a 6. karakter utáni rendezés után:

AZAZAZ, AMETSA, SOMBTA, ZEBESA, ZEMOBE, MESOBE, TAMTOM, BASTET.

5. karakter utáni rendezés végén:

TAMTOM, AZAZAZ, ZEMOBE, MESOBE, BASTET, AMETSA, ZEBESA, SOMBTA.

4. karakter utáni rendezés végén:

ZEMOBE, MESOBE, AZAZAZ, SOMBTA, ZEBESA, TAMTOM, BASTET, AMETSA.

3. karakter utáni rendezés végén:

AZAZAZ, ZEBESA, AMETSA, ZEMOBE, SOMBTA, TANTOM, MESOBE, BASTET.

2. karakter utáni rendezés végén:

SOMBTA, AZAZAZ, TANTOM, BASTET, ZEBESA, ZEMOBE, MESOBE, AMETSA.

1. karakter utáni rendezés végén:

ZEBESA, ZEMOBE, AZAZAZ, AMETSA, BASTET, MESOBE, SOMBTA, TANTOM.

3. Adott egy fekete-fehér, pixeles kép, amelyen egymás melletti m darab épület látható. A kép egy $n \times m$ méretű mátrixként van reprezentálva. A mátrix minden eleme 0 vagy 1 (, ahol az 1 az épületeket, a 0 az égboltot jelenti)

- Az oszlopok az épületeket reprezentálják, soronként felfelé haladva az épületek aljától a tetejük felé.
- Minden oszlopban az 1-esek alulról kezdődnek és egybefüggőek: azaz minden oszlopban először szerepelhet néhány 1, majd egy összefüggő 0-s szakasz a kép tetejéig (de 0 után már nem lehet újra 1).

A feladat az, hogy határozzuk meg annak az oszlopnak az indexét, amelyben a legtöbb egymás felett álló 1-es található (azaz a legmagasabb épület helyét). Ha több ilyen oszlop is van, bármelyik helyes válasznak tekinthető.

Adjunk meg egy algoritmust, amely legfeljebb $O(m \log n)$ mezőt vizsgál meg a mátrixban, és meghatározza a legmagasabb épületet.

Pontozás:

jó algoritmus

6 pont

indoklás

2 pont

lépésszám

2 pont

Megjegyzés: Létezik $O(m + n)$ lépésszámú algoritmus is, ami persze szintén elfogadható.

Megoldás: Ha az adott oszlopban az első 1-es a r -edik sorban (1-gyel kezdve a sorok indexelését) található, akkor az oszlop "épületmagassága" $n - r + 1$.

Algoritmus: Egy bináris keresés típusú algoritmussal tudjuk megoldani a feladatot.

1. Menjünk végig minden oszlopon (m db).

2. **Bináris keresésszerű algoritmus:** Az aktuális oszlopra alkalmazzuk a következőt a sorindexek tartományán $[0, n - 1]$:

- Állítsuk be $alsó = 1$ és $felső = n$.
- Ismételjük, míg $alsó < felső$:
 - Számoljuk ki a $középső = \lceil (alsó + felső) / 2 \rceil$ indexet.
 - Ha $A[középső][c] = 1$, akkor $felső = középső$; különben $alsó = középső + 1$.
- A végén $alsó$ jelöli az első 1-es helyét (vagy n , ha nincs 1-es).

3. **Magasság számítása:** Az oszlop magassága $h = n - alsó + 1$.

4. **Maximum keresése:** Válasszuk ki a tanult maximum kereső algoritmussal azt az oszlopot, amelyiknél h maximális.

Helyesség és futási idő: Az oszlopok értékei monoton nőnek (0-k, majd 1-esek), így a bináris keresésszerű algoritmus garantáltan jól működik és legfeljebb $O(\lceil \log_2 n \rceil)$ mezőt vizsgál meg. Ezt m oszlopon futtatjuk, majd a kapott m érték közül választjuk ki a maximumot, de ehhez már nem kell újabb mezőket megvizsgálni. Tehát az algoritmus összesen $O(m \log n)$ mezőt vizsgál meg.

4. Az alábbi irányított gráfokat, G_1 -et és G_2 -t szomszédsági listáik írják le. Futtassunk mélységi keresést G_1 -en és G_2 -n és számítsuk ki minden csúcslétségi és befejezési számát, majd az órán tanult módszer használatával döntsük el, hogy melyik gráf tartalmaz irányított kört, azaz melyik nem irányított aciklikus gráf, és adjunk meg egy topologikus rendezést abban a gráfban, amelyben ez lehetséges.

- G_1 : **a:** b, c; **b:** d; **c:** d; **d:** e; **e:** a.
- G_2 : **a:** g, f; **b:** a, g; **c:** -; **d:** -; **e:** c, d; **f:** e; **g:** f, e.

Megoldás:

G_1

Csúcs	Mélységi	Befejezési
a	1	5
b	2	3
c	5	4
d	3	2
e	4	1

3 pont

Egy él $x \rightarrow y$ visszaél, ha a mélységi szám(x) > mélységi szám(y) és befejezési szám(x) < befejezési szám(y), így G_1 -ben az $e \rightarrow a$ él visszaél, azaz G_1 nem DAG. 1 pont

G_2

Csúcs	Mélységi	Befejezési
a	1	6
g	2	5
f	3	4
e	4	3
c	5	1
d	6	2
b	7	7

3 pont

Mivel G_2 -ben nincs visszaél, ez egy DAG.

1 pont

A topologikus sorrendet a befejezési számok szerinti csökkenő sorrend adja.

1 pont

Topologikus sorrend: b, a, g, f, e, d, c .

1 pont

5. Adott egy irányított, élsúlyozott gráf $G = (V, E)$, amely Algoritmisztán városait és közúti útvonalait reprezentálja. Minden él $e \in E$ két várost köt össze egy adott irányban, és súlya $w(e)$ azt az időt jelenti, amely az adott irányú utazáshoz szükséges. A gráf éllistával van megadva.

Két barát, akik jelenleg Algoritmisztán két különböző városában tartózkodnak (jelölje ezeket A és B), találkozót szeretne szervezni egy harmadik algoritmisztáni városban. A találkozás feltételei a következők:

- Mindketten ugyanabban az időpillanatban indulnak el,
- mindketten a gráf élei mentén haladhatnak csak, az él által adott irányban, az élhez tartozó idő alatt és a városokban nem kell várniuk az utazás során (pl. kocsival mennek),
- nem hagyják el Algoritmisztán területét, azaz a gráf csúcsai között kell mozogniuk.

A cél az, hogy megtaláljuk azt a várost $C \in V$, ahol a két barát a *leg hamarabb találkozhat* (azaz a közös indulástól a találkozásig eltelt idő a legkevesebb). Nem kell, hogy egyszerre érjenek oda, az egyik barát várhat a másikra. Ha több ilyen város is van, bármelyik elfogadható megoldás.

Adjunk meg egy algoritmust, amely legfeljebb $O((m+n)\log n)$ idő alatt meghatároz egy ilyen találkozási helyet, ahol n a gráfban lévő csúcsok számát jelenti.

Megoldás:

Algoritmus:

- I. Futtassuk Dijkstra algoritmusát a gráfban G az A kezdőpontból, hogy minden $v \in V$ esetén kiszámoljuk $d_A(v)$, azaz az A -ból v -be vezető legrövidebb (idejű) út hosszát. 2 pont
- II. Futtassuk Dijkstra algoritmusát a B kezdőpontból, így meghatározzuk $d_B(v)$ értékeit. 1 pont
- III. Minden $v \in V$ esetén számoljuk ki $T(v) = \max\{d_A(v), d_B(v)\}$. 2 pont
- IV. Válasszuk azt a $C \in V$ -t, amelyre $T(C)$ minimális, és adjuk vissza ezt C -t. 1 pont

Indoklások:

- Miért a Dijkstra?

- $T(v)$ indoklása

2 pont

Lépésszám:

A Dijkstra algoritmus éllista esetén $O((m+n)\log n)$ időben fut, és mivel kétszer hajtjuk végre (egyszer A -ból, egyszer B -ből), az össz idő $O((m+n)\log n)$. A $T(v)$ értékek számítása és a minimum kiválasztása $O(n)$ időt vesz igénybe, így a teljes algoritmus futási ideje $O((m+n)\log n)$. 2 pont

6. Egy hosszú autótúra készülünk, amelyet a 0 kilométernél kezdünk meg. Az út mentén n darab szálloda található, amelyek a következő, növekvő sorrendben megadott kilométerpontokon helyezkednek el:

$$0 < a_1 < a_2 < \dots < a_n.$$

(Itt a_i az indulási ponttól mért távolságot jelöli kilométerben, $i = 1, 2, \dots, n$. A számok nem feltétlen egészek.) Az út során csak ezek közül néhány kiválasztott helyen állunk meg, máshol nem, azonban az utolsó szállodánál (a_n) kötelezően meg kell állnunk, mivel az a célállomásunk. Ideális esetben naponta 350 kilométert szeretnénk utazni. A tényleges napi megtett távolság $x \geq 0$ esetén az adott naphoz tartozó *büntetés* a következőképpen alakul:

$$(350 - x)^4.$$

(**Fontos:** az x lehet kisebb és nagyobb is 350-nél; a képlet mindkét esetet bünteti, mivel az abszolút eltérés negyedik hatványát méri.)

Adjunk olyan $O(n^2)$ futásidejű algoritmust, amely meghatározza, hogy mely szállodákban érdemes megállni úgy, hogy az összes napra eső büntetések összege minimális legyen, míg elérjük a célállomást! (Például a 0, 300, 420, 770 esetben az optimális megállóhelyek a 420 és 770.)

Megoldás: **Algoritmus:** Legyen $a_0 = 0$, és definiáljuk a **Bünt** tömböt (részfeladatokat), ahol **Bünt**[j] a minimális összes büntetés, hogy eljussunk a j . szállodáig ($j = 0, 1, \dots, n$). **3 pont**

A kezdet és a továbblépés képlete:

$$\text{Bünt}[0] = 0, \quad \text{Bünt}[j] = \min_{0 \leq i < j} \left\{ \text{Bünt}[i] + \left(350 - (a_j - a_i) \right)^4 \right\}, \quad j = 1, \dots, n.$$

indoklással:

Egy **előző** tömb segítségével nyomon a szokásos módon követjük, melyik i -nél vette fel a minimumot. **1 pont**

A végén **Bünt**[n] adja a minimális büntetés mértékét és az **előző** tömbben visszalépkedve megkapjuk a konkrét szállodákat. **1 pont**

Lépésszám: Minden j ($1 \leq j \leq n$) esetén az összes i ($0 \leq i < j$) lehetőséget végigvizsgáljuk, így egy konkrét j -re **Bünt**[j] kiszámításának a futásideje $O(n)$, az algoritmus össz futásideje $O(n^2)$. **1 pont**

7. Adott éllistával egy irányított gráf. Adjunk $O(m + n)$ futásidejű algoritmust, ami eldönti, hogy van-e olyan csúcs a gráfban, ahonnan minden csúcs elérhető irányított úton. (n - szokás szerint - a gráfban lévő csúcsok számát, m pedig az élek számát jelöli.)

[*Segítség: próbáljuk meg a feladatot először egy DAG-ra megoldani.*]

Megoldás:

Algoritmus:

- (a) Futtassunk egy DFS-t az egész gráfon. Az utolsóként befejezett csúcsot nevezzük **esetleg**-nek.
 (b) Indítsunk egy újabb DFS-t az **esetleg** csúcsból.
 (c) Amennyiben a második DFS minden csúcsot elér, akkor **esetleg** egy olyan csúcs, ahonnan a gráf összes csúcsa elérhető. Egyébként nincs ilyen csúcs. **4 pont**

Helyesség:

Használjuk azt a tanult tételt, hogy a $DFS(G, v)$ pontosan azokat a csúcsokat éri el, ahova irányított út van v -ből és még nem lettek felfedezve.

Ha DFS-t többször kell „újrakezdeni”, akkor az utolsó újrakezdesnél elért pontokba biztosan nem vezet irányított út az előzőleg elértékből a fenti tétel szerint. Így csak az utolsó "elkezdesnél" elért csúcsok lehetnek esélyesek, hogy van belőlük irányított út a többiekhez. Ha ezen pontok közül bármely w -ből van irányított út az összes többibe, akkor **esetleg**-ből is van, hiszen **esetleg**-ből van w -be (mivel **esetleg** befejezési száma a legnagyobb, az utolsó „újrakezdes” **esetleg**-ből történt) és összefűzzük w -ből más csúcsokba menő utakkal, így lesz egy irányított élsorozat (tehát út is). **5 pont**

Lépésszám:

Az első DFS bejárása $O(n + m)$ időt vesz igénybe, a második DFS szintén $O(n + m)$ időben fut, így az algoritmus teljes futásideje $O(n + m)$. **1 pont**

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 legfőljebb 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 dolgozathoz 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.

1. Tegyük fel, hogy minden $n \geq 2025$ esetén teljesül:

$$2 \cdot n \leq f(n) \leq 10 \cdot n \cdot \log_2 n,$$

és minden $n \geq 428$ esetén teljesül:

$$2 \cdot \log_2 n \leq g(n) \leq 10^5 \cdot \sqrt{n}.$$

Mutassa meg megfelelő $c > 0$ és $n_0 \in \mathbb{N}$ konstansok megadásával, hogy $f(n) + g(n) \in O(n \log_2 n)$.

Megoldás: A feltételek szerint minden $n \geq 2025$ esetén

$$f(n) \leq 10 \cdot n \cdot \log_2 n \quad \text{és} \quad g(n) \leq 10^5 \cdot \sqrt{n}.$$

2 pont

Ezért

$$f(n) + g(n) \leq 10 \cdot n \cdot \log_2 n + 10^5 \cdot \sqrt{n}.$$

1 pont

Mivel $10^5 \cdot \sqrt{n} \leq 10^5 \cdot n \cdot \log_2 n$, igaz, ha $n \geq 2$ (tehát $n \geq 2025$ esetén is), ezért

2 pont

$$f(n) + g(n) \leq 10 \cdot n \cdot \log_2 n + 10^5 \cdot n \cdot \log_2 n = (10^5 + 10) \cdot n \cdot \log_2 n.$$

2 pont

teljesül, ha $n_0 = 2025$ és $c = 10^5 + 10$, ezért

2 pont

$$f(n) + g(n) \in O(n \log_2 n).$$

1 pont

2. Tegyük fel, hogy egy rendezett tömb n különböző egész számot tartalmaz az $\{1, 2, 3, \dots, n, n + 1, n + 2\}$ halmazból, azaz pontosan **két** szám hiányzik. Adjon algoritmust, ami megtalálja a két hiányzó számot és legfeljebb $O(\log n)$ összehasonlítást használ.

Megoldás: Legyen $A[1 \dots n]$ a rendezett tömb, amely az $\{1, 2, \dots, n + 2\}$ elemeiből két hiányzó szám ($m_1 < m_2$) mellett tartalmaz n elemet. Ekkor $i < m_1$ esetén $A[i] = i$, $m_1 \leq i < m_2 - 1$ esetén $A[i] = i + 1$, és $m_2 \leq i - 1$ esetén pedig $A[i] = i + 2$.

2 pont

Ha az m_2 határa egygyel elcsúszik.

-1 pont

Először keressük a legkisebb i indexet, amelyre $A[i] < i$.

1 pont

Ezt a bináris kereséshez hasonlóan a tesszük. A középsőre megnézzük, hogy $A[i] = i$ teljesül-e. Ha igen, akkor nagyobb indexűek között folytatjuk, ha nem akkor a kisebb indexűeknél.

2 pont

Ezután $m_1 = i$ (ha $i = 1$ és $A[1] \neq 1$, akkor $m_1 = 1$; ha minden i esetén $A[i] = i$, akkor $m_1 = n + 1$).

1 pont

A zárójeles rész hiánya miatt nem kell levonni.

Most keressük a legkisebb $j \geq m_1$ indexet, amelyre $A[j] < j + 1$.

2 pont

Ezt az előző bináris kereséshez hasonlóan találjuk meg. Ekkor $m_2 = j + 1$ (ha nincs ilyen j , akkor $m_2 = n + 2$). 1 pont

Mivel mindkét keresés $O(\log n)$ összehasonlítást igényel, az egész algoritmus $O(\log n)$ összehasonlítást használ. 1 pont

3. Adott két tömb, mindkettőben n különböző egész szám van, de a két tömbnek lehetnek közös elemei. A tömbök nem feltétlenül rendezettek. Tervezzen olyan algoritmust, amely kiírja azokat a számokat, amik mindkét tömbben szerepelnek vagy jelzi, ha nincsen a tömböknek közös eleme. Az algoritmus legfeljebb $O(n \log n)$ összehasonlítást használjon.

Megoldás: Legyen A és B a két tömb, melyek mindegyikében n különböző egész szám szerepel. Az algoritmus a következő lépésekből áll (futásidőkkel):

1. Rendezés: Rendezzük az A tömböt $O(n \log n)$ időben (pl. összefésüléssel). 3 pont

2. Keresés: Vegyük sorra a B tömb elemeit, és minden $x \in B$ esetén végezzünk bináris keresést a rendezett A -ban, hogy ellenőrizzük, x szerepel-e benne. 3 pont

Mivel bináris keresés $O(\log n)$ összehasonlítást használ egy elemre, a teljes keresési fázis $O(n \log n)$ lesz. 2 pont

3. Kimenet: Írjuk ki az összes olyan x -et, amelyet a keresés megtalált. Ha egyetlen közös elem sem található, jelezzük, hogy nincs közös elem. 2 pont

4. Az alábbi szomszédossági mátrix segítségével adott egy súlyozott, irányított gráf, melyben topologikus rendezés a csúcsok a, b, c, d, e, f, g sorrendje. (Ezt nem kell belátni.)

	a	b	c	d	e	f	g
a	0	∞	-2	∞	∞	∞	∞
b	∞	0	∞	-3	∞	2	∞
c	∞	∞	0	1	4	∞	∞
d	∞	∞	∞	0	∞	∞	-1
e	∞	∞	∞	∞	0	-2	∞
f	∞	∞	∞	∞	∞	0	3
g	∞	∞	∞	∞	∞	∞	0

- (a) Az órán tanult, a topologikus sorrendet használó eljárással számítsa ki a legrövidebb utak hosszát a b csúcsból az összes többi csúcsba.

- (b) Az (a) pontban használt eljárás segítségével határozzon meg egy legrövidebb utat a b pontból az f pontba.

Megoldás:

Topologikus sorrend szerint megyünk végig:

1 pont

a: a a forrás előtt van, így $D[a] = \infty$.

1 pont

b: Forrás: $D[b] = 0$.

1 pont

c: Bejövő élek:

$$a \rightarrow c: w(a,c) = -2, \quad D[a] + (-2) = \infty - 2 = \infty;$$

Tehát $D[c] = \infty$.

1 pont

d: Bejövő élek:

$$b \rightarrow d: w(b,d) = -3, \quad D[b] + (-3) = 0 - 3 = -3;$$

$$c \rightarrow d: w(c,d) = 1, \quad D[c] + 1 = \infty + 1 = \infty.$$

Ezért $D[d] = -3$ (és $\text{elozo}[d] = b$).

1 pont

e: Bejövő élek:

$$c \rightarrow e: w(c,e) = 4, \quad D[c] + 4 = \infty;$$

Így $D[e] = \infty$.

1 pont

f: Bejövő élek:

$$b \rightarrow f: w(b,f) = 2, \quad D[b] + 2 = 0 + 2 = 2;$$

$$e \rightarrow f: w(e,f) = -2, \quad D[e] + (-2) = \infty - 2 = \infty.$$

Tehát $D[f] = 2$ (és $\text{elozo}[f] = b$).

1 pont

g: Bejövő élek:

$$d \rightarrow g: w(d,g) = -1, \quad D[d] + (-1) = -3 - 1 = -4;$$

$$f \rightarrow g: w(f,g) = 3, \quad D[f] + 3 = 2 + 3 = 5;$$

Így $D[g] = \min\{-4, 5\} = -4$ (és $\text{elozo}[g] = d$).

1+1 pont

Eredmény: Legrövidebb utak b forrástól: *Nem kell feltétlen külön leírni, ha már ki van számolva.*

$$\begin{aligned} D[a] &= \infty \quad (a \text{ nem érhető el}); \\ D[b] &= 0; \\ D[c] &= \infty; \\ D[d] &= -3; \\ D[e] &= \infty; \\ D[f] &= 2; \\ D[g] &= -4. \end{aligned}$$

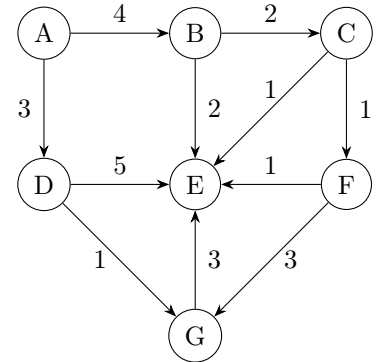
(b) Legrövidebb út $b \rightarrow f$:

Mivel $D[f] = 2$ és $\text{elozo}[f] = b$, a legrövidebb út egyszerűen: $b \rightarrow f$, súlya 2.

1 pont

5. Egy irányított, élsúlyozott gráf csúcsait és éleit a jobb oldali ábra mutatja.

Futtassa a Dijkstra-algoritmust az A pontból kiindulva.



- (a) Határozza meg az összes többi csúcsba vezető legrövidebb utak hosszát!
- (b) Adjon meg egy-egy legrövidebb utat A -ból a következő csúcsokhoz: C, E, G .

(Indokolni nem kell, de látszódjon, hogy lépésenként hogyan változik a D és a P tömb illetve a $KÉSZ$ halmaz.)

Megoldás:

It.	Új csúcs KÉSZ-ben	$D(\cdot)$							$P(\cdot)$						
		A	B	C	D	E	F	G	A	B	C	D	E	F	G
0	A	0	4	∞	3	∞	∞	∞	-	A	-	A	-	-	-
1	D	0	4	∞	3	8	∞	4	-	A	-	A	D	-	D
2	B	0	4	6	3	6	∞	4	-	A	B	A	B	-	D
3	G	0	4	6	3	6	∞	4	-	A	B	A	B	-	D
4	C	0	4	6	3	6	7	4	-	A	B	A	B	C	D
5	E	0	4	6	3	6	7	4	-	A	B	A	B	C	D
6	F	0	4	6	3	6	7	4	-	A	B	A	B	C	D

6 pont

(a) Legrövidebb utak hosszai A -ból:

1 pont

$$\begin{aligned} A &: 0, \\ B &: 4, \\ C &: 6, \\ D &: 3, \\ E &: 6, \\ F &: 7, \\ G &: 4. \end{aligned}$$

(b) Egy-egy legrövidebb út A -ból:

- $\underline{A \rightarrow C}$: $P(C) = B$ és $P(B) = A \Rightarrow$ út: $A \rightarrow B \rightarrow C$ (összsúly: $4 + 2 = 6$).
- $\underline{A \rightarrow E}$: $P(E) = B$ és $P(B) = A \Rightarrow$ út: $A \rightarrow B \rightarrow E$ (összsúly: $4 + 2 = 6$).
- $\underline{A \rightarrow G}$: $P(G) = D$ és $P(D) = A \Rightarrow$ út: $A \rightarrow D \rightarrow G$ (összsúly: $3 + 1 = 4$).

1 pont

1 pont

1 pont

6. Az egyetem újonnan szerződöttetett burkolója egy különleges feladaton töri a fejét: Egy hosszú folyosó $3 \times n$ -es padlóját kell lefedni 3×1 -es járólappal (a járólapok keresztben és hosszában is elhelyezhetők, tehát 3×1 -es vagy 1×3 -es alakban is használhatók), az alábbi feltételekkel:

- a járólapok nem fedhetik egymást,
- nem lóghatnak túl a padló szélén, és
- a teljes padlót pontosan le kell fedni.

Adjon meg egy $O(n)$ lépésben működő algoritmust, amely kiszámolja, hogy hányféleképpen lehet szabályosan lefedni a padlót ezekkel a feltételekkel.

Megoldás: $T(i)$ jelentése: megadja annak a számát, hányféleképpen lehet lefedni egy $3 \times i$ -es padlót 3×1 és 1×3 járólappal a feladatban adott feltételekkel. **1 pont**

Sorrend: Az értékeket az $i = 0$ -tól $i = n$ -ig kell kiszámítani, tehát az értékek növekvő sorrendben kerülnek kiszámításra. **1 pont**

Kezdő lépések: **2 pont**

$$\begin{aligned} T(0) &= 1 \quad (\text{üres padló esetén 1 triviális lefedés}), \\ T(1) &= 1 \quad (\text{egy } 3 \times 1 \text{ padló egyetlen módon fedhető le}), \\ T(2) &= 1 \quad (\text{egy } 3 \times 2 \text{ padlót csak kizárólag } 3 \times 1\text{-es járólappal lehet lefedni}). \end{aligned}$$

Vége: Miután elértük $i = n$ -t, a padló lefedési lehetőségeinek száma $T(n)$ lesz, ezt adjuk vissza az algoritmus eredményeként. **1 pont**

Tovább lépés: Minden i -re, ahol $3 \leq i \leq n$ esetén:

$$T(i) = T(i - 1) + T(i - 3),$$

attól függően, hogy az i -edik 3×1 -es vagy 1×3 -as volt. **2 pont**

Helyesség: Az i -edik „oszlopban” lévő 3 négyzetet vagy egy 3×1 -es lappal fedjük le, vagy pedig 3 db 1×3 -as fedi le őket. Más lehetőség nincs. **2 pont**

Futásidő: Mivel a kezdőértékek beállítása $O(1)$ és a tovább lépési ciklus $i = 3$ -tól $i = n$ -ig szintén minden i -re $O(1)$ idejű (hiszen csak 2 előző értéket kell kiolvasni és összeadni a tömbben), az algoritmus futási ideje: $O(n)$. **1 pont**

7. Algoritmisztán térképe egy szomszédsági mátrixszal adott irányított gráf, melynek csúcsai a városok, irányított élei pedig a városok között vezető közvetlen utak. Az utak használatáért általában fizetni kell, 2025 útszakasz kivételével ez az ár pozitív, de erre a 2025 útra nulla. Az algoritmisztáni árhivatal azt a szabályt hozta, hogy egy utazó egy út során legfeljebb három ingyenes útszakaszt használhat, egyébként összedől az ország költségvetése. Adjon $O(n^2)$ lépésszámú algoritmust, ami meghatározza a legolcsóbb olyan utat egy adott A városból egy adott B városba, ami legfeljebb három ingyenes utat használ.

Megoldás: Legyen $G = (V, E)$ az eredeti, szomszédsági mátrixos irányított gráf (a városokat csúcsok, az utak élek jelölik), és legyen $F \subseteq E$ az ingyenes útszakaszok halmaza. A szabály azt írja elő, hogy egy utazó legfeljebb 3 ingyenes útszakaszt használhat egy út során.

Algoritmus:

Minden olyan S részhalmazon iterálunk, melyre $S \subseteq F$ és $|S| = 3$, ezek száma $\binom{2025}{3}$, ami egy konstans. **2 pont**
1 pont

- (a) Egy adott S -re módosítjuk a gráfot úgy, hogy az új élhalmaz $E_S = \{e \in E \mid e \notin F\} \cup S$ tartalmazza az összes fizetős élt, valamint az S -ben kiválasztott ingyenes éleket. **2 pont**
- (b) Minden módosított gráfra $G_S = (V, E_S)$ futtassuk a Dijkstra algoritmust A városból B városba, és jelöljük $C(S)$ -sel a minimális útköltséget. **2 pont**
- (c) Válasszuk ki a legkisebb költséget: $C^* = \min\{C(S) \mid S \subseteq F; |S| = 3\}$. **1 pont**

Futásidő: Mivel az ingyenes élekből választott részhalmazok száma konstans, és egy Dijkstra algoritmus a szomszédsági mátrix miatt $O(n^2)$ lépést igényel, a teljes algoritmus futása $O(n^2)$. **2 pont**

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 pusztán 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 legfőljebb 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 dolgozatról 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.

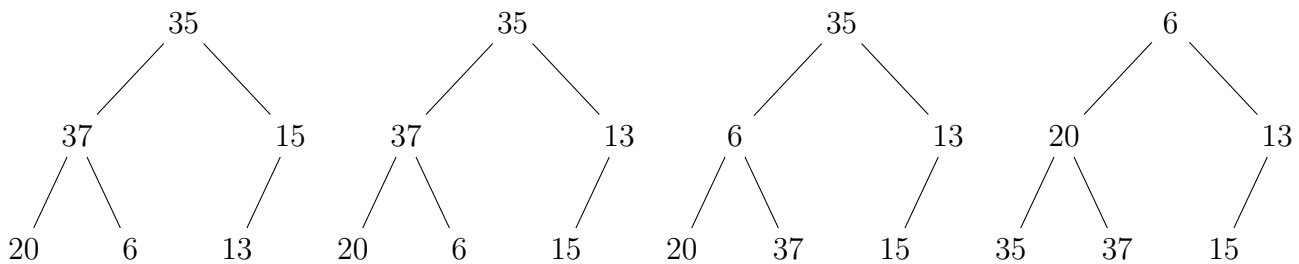
Az első 3 feladatban az algoritmusok lépéseinek bemutatásán kívül nem várunk további indoklást.

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

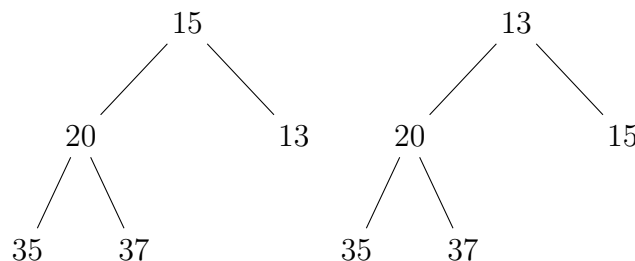
35	37	15	20	6	13
----	----	----	----	---	----

- 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 mintőrnél jó elemet tesz a gyökérbe | 1 pont |
| Jó végeredmény | 2 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

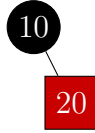
3. Egy kezdetben üres piros-fekete fába sorban szúrja be a következő számokat: 10, 20, 30, 15, 18. Minden egyes beillesztés után adja meg a fa állapotát (a csomópontok színeivel együtt). Ha forgatást alkalmaz, akkor ábrázolja a fát a forgatás előtt és után is.

Megoldás: Az üres fekete levelek nélkül ábrázolva a fákat:

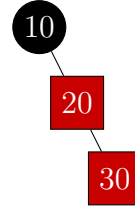
Beszúrás: 10



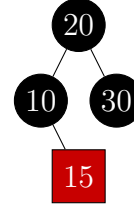
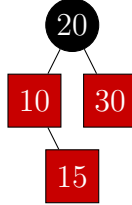
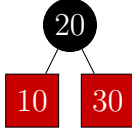
Beszúrás: 20



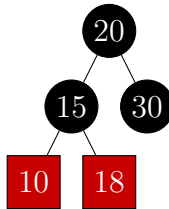
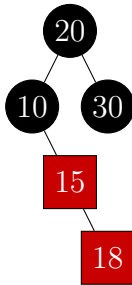
Beszúrás: 30 forgatás előtt



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 $\ell - k$ -edik 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.

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!

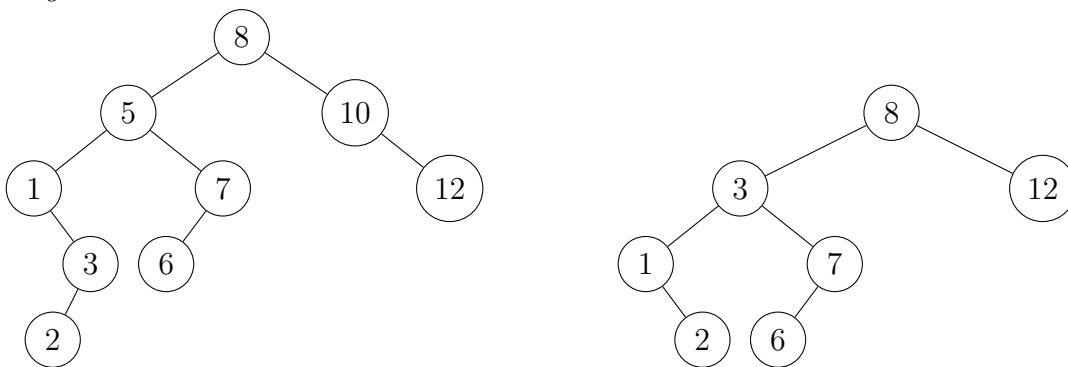
Kérjük, írja fel a nevét, NEPTUN-kódját és a gyakorlatvezető nevét az összetűzött lapok jobb felső sarkába.

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 pusztán 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 dolgozathoz 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. Egy kezdetben üres bináris keresőfába sorban szúrja be a következő elemeket: 8, 5, 10, 1, 3, 7, 12, 6, 2
 - a) Rajzolja fel a fát az összes beszúrás elvégzése után.
(A köztes állapotokat nem muszáj felrajzolni.)
 - b) Végezze el a TÖRÖL(10) és TÖRÖL(5) műveleteket, majd ábrázolja a kapott fát.

Megoldás:

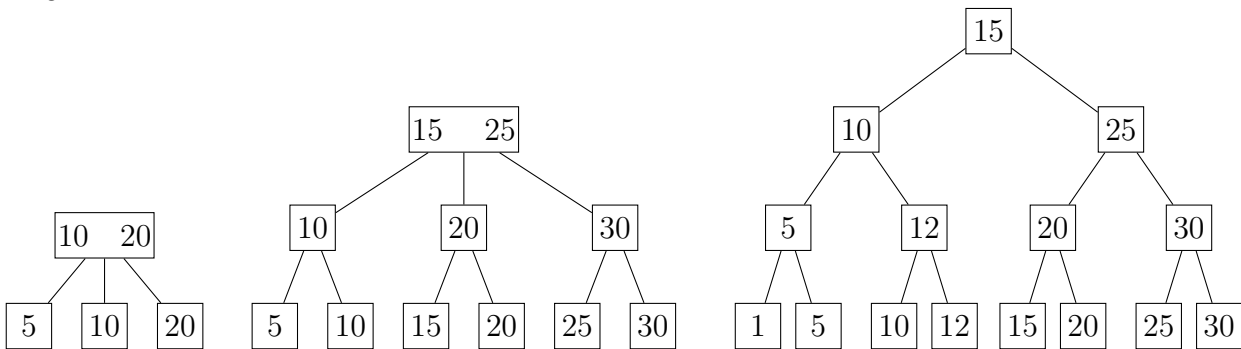


Pontozás: a) rész fája helyesen
b) jó törlések

6 pont
2+2 pont

2. Egy üres 2-3 fába szúrja be sorban a következő kulcsokat: 10, 20, 5, 15, 25, 30.
 - a) Rajzolja fel a fa állapotát a 10, 20, 5 beszúrása után és az összes beszúrás elvégzése után is.
(Ne feledkezzen meg a belső csúcsok irányjelzőiről sem.)
 - b) A kapott fába szúrja be még a 12 és 1 kulcsokat is, majd rajzolja fel a kapott fa állapotát.

Megoldás:



Pontozás: a) rész két fája helyesen
 b) végső fa
 Ha csak az irányjelzők rosszak legalább

2+3 pont
 5 pont
 6 pont

3. Nyitott címzésű hashelést használunk, a hash-tábla mérete $m = 11$, a hash-függvény pedig $h(k) = k \bmod 11$. Az ütközések kezelésére **kvadratikus próbát** alkalmazunk.

A tábla állapota jelenleg a következő ($a * a$ törölt jel):

0	1	2	3	4	5	6	7	8	9	10
22		46	*				30	41	53	

- a) A **kvadratikus próba** esetén az x kulcs beszúrásakor mi próbasorozat általános alakja?
- b) Illessze be a következő három kulcsot a hash-táblába: 35, 13, 74. Minden beszúrás után ábrázolja a hash-tábla aktuális állapotát.
- c) A kapott táblában végezze el a TÖRÖL(41) műveletet és ismét ábrázolja a hash-tábla aktuális állapotát.

Megoldás:

a) A kvadratikus próba próbasorozata: $h(x) + 1^2, -1^2, +2^2, -2^2, +3^2, \dots \pmod{11}$ **2 pont**

b) **1. Beszúrás: 35**

0	1	2	3	4	5	6	7	8	9	10
22		46	35				30	41	53	

2 pont

2. Beszúrás: 13

0	1	2	3	4	5	6	7	8	9	10
22	13	46	35				30	41	53	

2 pont

3. Beszúrás: 74

0	1	2	3	4	5	6	7	8	9	10
22	13	46	35	74			30	41	53	

2 pont

c) TÖRÖL(41) művelet

0	1	2	3	4	5	6	7	8	9	10
22	13	46	35	74			30	*	53	

2 pont

4. Lássuk be, hogy az alábbi eldöntési probléma **NP**-ben van.

Input: Összefüggő G gráf, k és ℓ egész számok, ahol $k \geq 2025$ és $\ell \geq 2025$

Kérdés: Létezik-e G -ben olyan k méretű teljes részgráf (klikk) és ℓ méretű független ponthalmaz, amelyeknek van közös csúcsa?

Megoldás: Jó tanú egy k méretű klikk és egy ℓ méretű független csúcshalmaz, aminek van közös pontja. **4 pont**

Ez polinom méretű, hiszen nem nagyobbak, mint a gráf (elég azt mondani, hogy trivi) **1 pont**

Ellenőrző algoritmus ellenőrzi, hogy

a teljes gráf teljes-e és részgráf-e **1 pont**

a független csúcshalmaz független **1 pont**

mindkettőben van 2025 csúcs **1 pont**

van közös csúcs **1 pont**

minden triviálisan megy polinom időben **1 pont**

5. Adott egy T bináris fa. Azt akarjuk ellenőrizni, hogy T minden x csúcsára teljesül-e: x baloldali részfájának és x jobboldali részfájának magassága legfeljebb eggyel tér el egymástól.

Adjon erre $O(n)$ futásidejű algoritmust, ahol n a T fa csúcsainak száma.

Megoldás: Dinamikus programozást használunk, minden x csúcsra kiszámítjuk a részfa $m(x)$ magasságát. *Nem kell levonni, ha nem hivatkozik explicit a DP-re, de amúgy jó az eljárás.* **3 pont**
(A levelekre $m(x) = 0$, *nem kell levonni, ha hiányzik.*)

postorder sorrendben haladunk **2 pont**

egy új csúcsra ellenőrizzük, hogy a két gyerekre teljesül-e a feltétel, azaz $|m(\text{bal}(x)) - m(\text{jobb}(x))| \leq 1$ (ha nem, megállunk) **2 pont**

az új csúcsra $m(x) = \max(m(\text{bal}(x)), m(\text{jobb}(x))) + 1$ (képlet helyett a magyarázat is jó) **2 pont**

a postorder lépésszáma $O(n)$, egy csúcsra konstans sok lépés **1 pont**

6. Legyen a **KBFÉL-SZÍN** probléma a következő:

Input: Egy $2p$ csúcsú G gráf

Kérdés: Kiszínezhetők-e a G gráf csúcsai $p + 3$ színnel úgy, hogy a szomszédos csúcsok különböző színt kapnak?

Adjon meg egy **3-SZÍN** \prec **KBFÉL-SZÍN** Karp-redukciót.

Megoldás: Legyen $f(G)$ az a gráf, amit a p pontú G gráfból úgy kapunk, hogy hozzáveszünk p új pontot, amik egy teljes gráfot alkotnak és mindegyiket hozzákötjük G minden pontjához (egy jó ábra is elég) **5 pont**

Ez polinom időben számolható, hiszen $2p$ csúcs és legfeljebb $(2p)^2$ él lesz (elég azt mondani, hogy trivi) **1 pont**

Ha G színezhető 3 színnel, akkor $f(G)$ minden új csúcsára használjunk külön színt, így $p + 3$ szín lesz. **2 pont**

Ha $f(G)$ színezhető $p + 3$ színnel, akkor minden új csúcsnak különbözik a színe egymástól is és G pontjainak színétől is, így G csúcsaira csak 2 szín marad. **2 pont**

7. Adjon polinomiális algoritmust a következő feladat eldöntésére:

Input: Egy N pozitív egész szám tízes számrendszerben felírva

Kérdés: Léteznek-e olyan $a \geq 2$ és $b \geq 2$ egész számok, hogy $a^b = N$?

Megoldás: Az input mérete $n = \lceil \log_{10}(N) \rceil$ (nem baj, ha nem pontos a kifejezés)

1 pont

Ha van ilyen b , akkor $b = \log_a(N) \leq \log_2(N) \leq c \cdot n$.

3 pont

Minden ilyen b -re külön eldöntjük, hogy van-e hozzá megfelelő a .

1 pont

Ezt a bináris kereséshez hasonlóan csináljuk, először kiszámoljuk pl., hogy $a_1 = N/2$ esetén a_1^b kisebb, vagy nagyobb N -nél, majd a válasz függvényében a bináris kereséshez hasonlóan folytatjuk.

(amúgy jobb becslés a_1 -re az N számjegyeinek számának $\log(10)/\log(2) < 3,33$ -szere) **3 pont**

Indoklás a lépésszáma... **2 pont**

2 pont