Algoritmuselmélet 1. ZH

A rendelkezésre álló munkaidő 90 perc. Minden megoldást indokoljon! 2024. április 18.
Minden feladat egységesen 10 pontot ér. Az aláírás megszerzéséhez minimum 24 pontot kell elérni.
Kérjük, minden résztvevő 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.

1. Legyen $f(n) = 2024$ ha $n < 15$ és $f(n) = n \cdot 2^{n+10}$ különben. Valamint legyen $g(n) = 2023$ ha $n < 42$ és $g(n) = n^2 \cdot 2^n$ különben. Igaz-e, hogy
   (a) $f(n) \in O(g(n))$? (b) $g(n) \in O(f(n))$?

   Megoldás: $f(n) = 2^{10} \cdot n \cdot 2^n$ ha $n \geq 15$ **2 pont**
   (a) Ha $n \geq 42$, akkor $n < n^2$, ezért $f(n) < 2^{10} g(n)$. **2 pont**
   Tehát az $n_0 = 42$ és $c = 2^{10}$ választás mutatja, hogy $f(n) \in O(g(n))$. **2 pont**
   (b) Tegyük fel, hogy valamilyen $n \geq n_0$ és c esetén $n^2 \cdot 2^n \leq c \cdot 2^{10} \cdot n \cdot 2^n$. **2 pont**
   Ekkor $n \leq c \cdot 2^{10}$-nek teljesülni kell minden elég nagy n-re, ami nem igaz. **1 pont**
   Tehát nem igaz, hogy $g(n) \in O(f(n))$. **1 pont**

2. Adottak $c_1, c_2, \ldots, c_n$ különböző egész számok. Ezeket szeretnénk nagyság szerint rendezni növekvő, vagy csökkenő sorrendbe úgy, hogy a szokásos összehasonlítás helyett most a következő kérdéseket lehet feltenni 1 lépésben: Három kiválasztott

elem közül melyik esik a rendezés szerint a másik kettő közé? (Például, ha a c_i, c_j, c_k elemeket kérdezzük ahol $c_i = 3, c_j = 1, c_k = 7$, akkor a c_i elemet kapjuk válaszként.) Adjon $O(n \log n)$ kérdést használó algoritmust, aminek kimenete a csökkenő vagy növekvő sorrendben lévő számok. *(Azt nem kell az algoritmusnak meghatároznia, hogy csökkenő vagy növekvő-e a sorozat.)*

Megoldás: Először vegyük az első 3 elemet, c_1, c_2, c_3-at és kérdezzük meg, hogy melyik a középső. Ha pl. c_2, akkor a sorrend vagy $c_1 < c_2 < c_3$, vagy $c_1 > c_2 > c_3$, írjuk le őket balról jobbra c_1, c_2, c_3 sorrendben **2 pont**
Ezután a beszúrásos rendezéshez hasonlóan járunk. **2 pont**
Ha $c_1, c_2, \ldots, c_k$ sorba van rakva (növően vagy csökkenően), akkor a bináris kereséshez hasonlóan beszúrjuk c_{k+1}-et. **2 pont**
Először a $c_{k+1}, c_{\lceil k/2 \rceil - 1}, c_{\lceil k/2 \rceil}$ hármasra kérdezünk rá. A választ meghatározza, hogy c_{k+1} a sorrend bal vagy jobb felében van-e, vagy pedig épp megtaláltuk a helyét. **2 pont**
A szóbejövő helyek száma minden kérdés után felére csökken, így egy elem beszúrása $O(\log n)$ lépés. Az összes lépésszám $O(n \log n)$. **2 pont**

3. Adott egy n pontú m élű egyszerű, összefüggő, irányítatlan gráf. Adjon $O(n + m)$ futásidejű algoritmust, ami megtalál egy olyan pontot, amelyet a gráfból elhagyva a gráf összefüggő marad.

Megoldás: Futtassunk egy DFS-t a gráf tetszőleges pontjából. **2 pont**
Ennek vegyük az 1 befejezési számú csúcsát (itt le is állíthatjuk a DFS-t). **2 pont**
Ez a DFS fának egy levele lesz. **2 pont**
Ha ezt a levelet elhagyjuk a gráfból, akkor a DFS fa maradék élei mutatják, hogy a maradék gráf összefüggő marad. **2 pont**
A DFS lépésszáma a befejezési szám meghatározásával együtt is $O(n + m)$ **2 pont**
Ha DFS helyett pl. BFS fát veszünk, akkor pl. az utolsó meglátogatott pont lesz biztosan levél.

4. Egy irányított kört nem tartalmazó irányított gráfban (DAG) lefuttattuk a DFS algoritmust valamelyik pontjából indulva. Azt kaptuk, hogy az (u,v) élre teljesül, hogy mszám$(v) <$ mszám(u). Bizonyítsa be, hogy bszám$(v) <$ bszám(u) teljesül.

Megoldás: Mivel mszám$(v) <$ mszám(u), ezért (u,v) csak visszaél vagy keresztél lehet. **3 pont**
Ha visszaél lenne, akkor lenne a gráfban irányított kör, vagyis nem DAG. **4 pont**
Tehát (u,v) keresztél. **1 pont**
Keresztélre bszám$(v) <$ bszám(u) **2 pont**

5. Egy n hosszú 0-1 sorozatot olyan, legalább 4 hosszú darabokra kell szétvágni, hogy minden keletkezett részben az első két bit megegyezzen az utolsó két bittel. (*Például*

a 01001111110110 *sorozat felvágható 3 részre:* 01001, 1111, 10110.) Adjon $O(n^2)$ lépésszámú dinamikus programozást használó algoritmust, amely eldönti, hogy van-e ilyen szétvágás!

Megoldás: Legyenek a sorozat elemei $s_1, \ldots, s_n$.
Legyen $M[i]$ = IGAZ, ha a sorozat a első i bitjének van megfelelő szétvágása, különben $M[i]$ = HAMIS. **2 pont**
$M[0]$ = IGAZ, i = 1,2,3-ra legyen $M[i]$ = HAMIS. **1 pont**
$i \geq 4$-re $M[i]$ akkor IGAZ, ha valamely $0 \leq j < i-4$ esetén $M[j]$ = IGAZ, valamint $s_{j+1} = s_{i-1}$ és $s_{j+2} = s_i$ teljesülnek. **3 pont**
Ha van ilyen j, akkor az $s_1, \ldots, s_j$ rész felvágása az $s_{j+1}, \ldots, s_i$ darabbal az $s_1, \ldots, s_i$ egy felvágását adja. **1 pont**
Akkor és csak akkor van az egész sorozatnak megfelelő felvágása, ha $M[n]$ = IGAZ. **1 pont**
Adott i-re a megfelelő j keresése $O(n)$ lépés. Ez minden i-re elvégezve, az összes lépésszám $O(n^2)$. **2 pont**
Más jó megoldás, ami nem dinamikus programozást használ: **5 pont**
Az nem derül ki a feladadat szövegéből, hogy ha a sorozat első és utolsó két bitje megegyzik, akkor az 1 részre vágás megfelelő-e. Mindkét féle értelmezést elfogadjuk.

6. Egy irányított gráf éllistája az élek súlyaival:

a : $(b,6)$, $(c,5)$, $(e,8)$
b : $(a,6)$, $(e,1)$, $(f,2)$
c : $(b,2)$, $(f,4)$,
e : $(b,6)$, $(g,3)$
f : $(e,1)$, $(g,1)$
g :

Dijkstra algoritmusával határozza meg a-ból az összes többi csúcsba vezető legrövidebb út hosszát. *(Indokolni nem kell, de látszódjon, lépésenként hogyan változik a távolságokat tároló D[ ] tömb és a KÉSZ halmaz.)*

Megoldás:

	a	b	c	e	f	g	KÉSZ
1.	0	6	5	8	∞	∞	$\{a\}$
2.	0	6	5	8	9	∞	$\{a,c\}$
3.	0	6	5	7	8	∞	$\{a,c,b\}$
4.	0	6	5	7	8	10	$\{a,c,b,e\}$
5.	0	6	5	7	8	9	$\{a,c,b,e,f\}$
6.	0	6	5	7	8	9	$\{a,c,b,e,f,g\}$

Az a oszlopa csupa 0, (vagy nincs is felírva) **1 pont**
Az esetek nagy részében jól választja ki, hogy melyik csúcs kerüljön át a KÉSZ

halmazba. **2 pont**
Az esetek nagy részében jól végzi el a javítást. **2 pont**
Ha sok számolási hiba van, akkor csak az eddigi 5 pont jár.
Ha viszont minden számolás jó, beleértve, hogy jól választotta ki a minimálisat, akkor további **5 pont**
5-nél több elszámolásnál ezért a részért nem jár pont.

7. Adott egy $n \times n$-es mátrix. Adjon $O(n^2 \log n)$ összehasonlítást használó algoritmust, amely eldönti, van-e két olyan sor, amelyek az első elem kivételével megegyeznek, viszont az első elemük meg különböző!

Megoldás: Rendezzük a sorokat először a 2. oszlop szerint. **1 pont**
Bináris keresést használó beszúrásos rendezéssel vagy összefésüléses vagy kupacos rendezéssel ez $O(n \log n)$ lépés **1 pont**
Így egymás utáni blokkokba rendeztük a sorokat, hogy egy blokkban a 2. elem egyenlő. **1 pont**
Most rendezzük külön-külön a blokkokat a harmadik elem szerint, így olyan blokkokat kapunk, ahol a 2. és 3. oszlop elemei is egyformák. (Ehelyett rendezhetünk az egész 3. oszlop szerint a beszúrásos rendezés olyan változatával, ami az azonos elemek sorrendjét megtartja.) Ezután hasonlóan folytatjuk a többi oszlopra. **2 pont**
Egy oszlop szerint a rendezés lépésszáma

$$O(n_1 \log n_1) + \ldots + O(n_k \log n_k) \subseteq O\left(\left(\sum_{i=1}^{k} n_i\right) \cdot \log n\right) \subseteq O(n \log n),$$

ahol n_i a blokkok mérete. **2 pont**
Végül minden blokkban végigmegyünk az első oszlopon. Ha nem mind egyforma, akkor találtunk két megfelelő sort. (Ehelyett lassabb algoritmus is megfelelő lehet.) **1 pont**
Ennek lépésszáma összesen $O(n)$ **1 pont**
Az össz lépésszám $n \cdot O(n \log n) + O(n) \subseteq O(n^2 \log n)$. **1 pont**