

A munkaidő 90 perc. A VÁLASZOKAT INDOKOLNI KELL.
Hivatkozni csak az előadáson tanultakra lehet.

1. Az alábbi függvények közül pontosan egyre igaz, hogy $O(n^2)$ -es.

$$\log(n^2) + \frac{n(n-1)(n-2)}{2010} \quad 2020n \log n + \frac{2^n \cdot n(n-1)}{3^n} \quad 20n^2(\log n)^2 + 8$$

(a) Válassza ki ezt a függvényt és lássa be megfelelő c konstans és n_0 küszöbérték megadásával, hogy $O(n^2)$ -es.

(b) A másik két függvény egyikéről (szabadon választhat, hogy melyikről) bizonyítsa be, hogy az nem $O(n^2)$ -es.

Megoldás:

(a) A második függvény $O(n^2)$ -es, mert

$$2020n \log n + \frac{2^n \cdot n(n-1)}{3^n} \leq 2020n \log n + n(n-1) \leq 2020n^2 + n^2 = 2021n^2$$

Mindkét egyenlőtlenség $n \geq 1$ esetén igaz, az első becslésnél azt használtuk, hogy $\frac{2^n}{3^n} \leq 1$, a másodikonál pedig azt, hogy $n \log n \leq n^2$ és $n(n-1) \leq n^2$.

A fentiek miatt $c = 2021$ és $n_0 = 1$ jó választás $O(n^2)$ definíciójában.

(b) A harmadik függvényről indirekt bizonyítással látjuk be, hogy nem $O(n^2)$.

Ha $O(n^2)$ -es volna, akkor létezne olyan $c > 0$ konstans és $n_0 \geq 1$ küszöb, hogy $20n^2(\log n)^2 + 8 \leq c \cdot n^2$ áll fenn, ha $n \geq n_0$.

Innen azonban

$$c \cdot n^2 \geq 20n^2(\log n)^2 + 8 \geq 20n^2(\log n)^2$$

adódik, amit n^2 -tel leosztva kapjuk, hogy

$$c \geq 20 \cdot (\log n)^2$$

teljesül, ha $n \geq n_0$, ami ellentmondás, mert $\log n$ minden határon túl nő, nem maradhat semmilyen c konstans alatt örökké.

Pontozás vázlata

(a) A jó függvény kiválasztása: **1 pont**
Jó becslések: **2 pont**
Jó c és n_0 megadása: **2 pont**

(b) Jó indirekt elindulás: **1 pont**
Helyes becslések, amik az ellentmondáshoz vezetnek: **2 pont**
Az ellentmondás elmagyarázása: **2 pont**

A (b) részben az első függvény kiválasztása esetén is hasonlóan oszlanak el a pontok.

Az O pusztán definíciójának felírásáért nem jár pont.

2. Az $A[1 : n]$ tömb pozitív számokat tartalmaz (a számok nem feltétlenül egészek és lehetnek 1-nél kisebb értékek is). Szeretnénk megtalálni azt az $A[i : j]$ résztömböt ($1 \leq i \leq j \leq n$), melyben a számok szorzata a lehető legnagyobb. Adjon $O(n)$ lépésszámú, dinamikus programozást használó algoritmust az elérhető legnagyobb érték megtalálására az alábbi részfeladatok megoldásával: $M[j]$ adja meg a legnagyobb elérhető szorzatot, ha a résztömb utolsó cellája j .

Megoldás:

A részfeladatokat $i = 1, 2, \dots, n$ sorrendben oldjuk meg. **1 pont**

$M[1] = A[1]$, mert ha egy résztömb az első cellában végződik, akkor az csak az első elemből áll.

1 pont

Az $i \geq 2$ esetben $M[i] = \max(A[i], M[i - 1] \cdot A[i])$, **2 pont**

mert az i -edik cellában kétféleképpen végződhet egy résztömb:

- a résztömb csak az i -edik cellából áll, ekkor a szorzat $A[i]$
- az i -edik cella előtt még más cellákat is használunk, a legnagyobb ilyen szorzat értéke a korábbi celláig kapható legnagyobb szorzat és az aktuális elem, azaz $A[i]$ szorzata **3 pont**

Az elérhető legnagyobb érték az $M[i]$ értékek maximuma, mert így figyelembe vesszük az összes lehetséges pozíciót a résztömb utolsó cellájára. **1 pont**

Az algoritmus lépésszáma azért $O(n)$, mert az n részfeladat mindegyike konstans lépésben megoldható és a maximumkeresés is $O(n)$ az n méretű tömbben. **1 + 1 pont**

3. Egy **irányítatlan** G gráf csúcsai A, B, C, D, E, F, H . Mélységi bejárást (DFS-t) futtatva a D csúcsból kiindulva a feszítőfába a DA, AB, AC, DE, EF, FH élek kerülnek be ebben a sorrendben.

(a) Lehetséges-e, hogy a G gráfban van AE él?

(b) Lehetséges-e, hogy a G gráfban van EH él?

Megoldás: (a) A DFS futása során visszalépünk az A csúcsból a D -be és majd csak ezután járjuk be az E csúcsot és ha lett volna AE él, akkor legkésőbb az A szomszédainak vizsgálatakor E -be mentünk volna még az előtt, hogy A -ból visszalépünk, **4 pont**
tehát biztos nincsen AE és a gráfban. **1 pont**

(b) Az EH élet a H csúcs szomszédainak vizsgálata során vizsgáljuk először, de ekkor az E csúcs már be van járva, ezért az él megléte nem változtatja meg a DFS futását, **4 pont**
ezért lehetséges, hogy van EH él a gráfban **1 pont**

Indoklás nélküli helyes tippre nem jár pont.

4. Éllistájával adott egy n csúcsú irányítatlan gráf és abban két kijelölt csúcs, A és B . A gráf minden csúcsa ki van színezve vagy pirosra vagy kékre, ez az információ egy S tömbben adott, amely a csúcsokkal van indexelve és ahol $S[v]$ a v csúcs színét adja meg. A gráf egy élet tarkának nevezzük, ha egyik végpontja piros, a másik pedig kék. Adjon $O(n+m)$ lépésszámú algoritmust, ami meghatározza, hogy van-e olyan út az A csúcsból a B csúcsba, ami csupa tarka élből áll és ha van ilyen, akkor azt is megmondja, hogy hány tarka élből áll a legrövidebb ilyen út.

Megoldás:

Az az ötlet, hogy a tarka élek gráfjában kell legrövidebb utat keresni: **1 pont**
ALGORITMUS (a fenti ötlet egy lehetséges pontos megvalósítása)

(a) Kitérünk minden nem tarka élet a gráfból **1 pont**

Ezt úgy lehet megtenni, hogy végigmegyünk az éllistán és ha egy w csúcs benne van egy v csúcs szomszédai között, de v és w színe megegyezik, akkor w -t töröljük v szomszédai közül: **1 pont**

(b) Az új G_1 gráfban futtatunk BFS -t A -ból (újrakezdés nélkül) **2 pont**

és ha B nem járódik be eközben, akkor nincsen megfelelő út, ha pedig B bejáródik, akkor a B -hez tartozó távolság adja meg a keresett élszámot. **1 pont**

HELYESSÉG

A módosítás után pontosan a csupa tarka élekből álló utak maradnak meg az új gráfban **1 pont**
és a BFS meghatározza ezek közül a legrövidebbet. **1 pont**

LÉPÉSSZÁM

A módosítás $O(n + m)$, mert egyszer kell végigmenni az $O(n + m)$ méretű éllistán és minden bejegyzésnél konstans sok munkát végzünk. **1 pont**

Az új gráfban n csúcs és $m' \leq m$ él van, a BFS lépésszáma pedig $O(n + m')$, azaz $O(n + m)$.

1 pont

5. A következő nyáron sok minket érdeklő fesztivál lesz, azonban sajnos ezek időpontjai között vannak átfedések. Ha egy fesztiválra elmegyünk, azon az első naptól az utolsóig ott akarunk lenni, de másnap már mehetünk egy újabbra. A szóba jövő f fesztivál mindegyikéről tudjuk, hogy melyik nap kezdődik és melyik nap végződik, célunk hogy minél több napot töltsünk fesztiválokön. Fogalmazza meg a feladatot egy gráfelméleti problémaként és adjon $O(f^2)$ lépésszámú algoritmust a fesztiválok egy ilyen kiválasztására.

Megoldás:

GRÁFELMÉLETI ÁTFOGALMAZÁS ÖTLETE

A gráf csúcsai a fesztiválok és akkor van irányított él a v_i fesztiválból a v_j fesztiválba, ha v_i utolsó napja megelőzi v_j első napját (azaz el tudunk menni v_i után v_j -re). **2 pont**

Az a gondolat, hogy minden élhez az egyik fesztivál hosszát rendeljük és leghosszabb utat akarunk keresni: **1 pont**

ALGORITMUS (a fenti ötlet megvalósítása pontosan) és HELYESSÉG

A fenti gráf egy DAG, mert egy gráfbeli, legalább egy élből álló úthoz tartozó fesztiválsorozat kezdőnapjai szigorúan növe sorozatot alkotnak. **1 pont**

A pontos megvalósításhoz vegyünk fel még egy csúcsot, amiből minden fesztiválcsúcsba vezet egy irányított él. Mivel ebbe az új csúcsba nem vezet él, ezért a gráf DAG marad. **1 pont**

Mindegyik élhez rendeljük hozzá annak a fesztiválnak a hosszát, amibe az él mutat. **1 pont**

Így a gráfban az olyan utak, amik az extra csúcsból indulnak megfelelnek a lehetséges legális fesztiválsorozatoknak és egy ilyen út hossza a megfelelő fesztiválokön töltött összes idővel egyezik meg. **1 pont**

Mivel a gráf DAG, használjuk a DAG-ban leghosszabb utat kereső algoritmust az extra csúcsból futtatva (az utak nyomonkövetésével), így minden fesztiválcsúcsba megkapjuk az ezzel a fesztivállal végződő leghosszabb utat. A fesztiválokhoz így kapott értékek maximumát megkeresve, az adott fesztiválhoz tartozó leghosszabb út adja a legjobb kiválasztást. **1 pont**

LÉPÉSSZÁM

A gráf szomszédossági mátrixának elkészítése $O(f^2)$ lépés, mert minden fesztivál-párt egyszer kell összehasonlítani (ez $O(f^2)$ lépés, ezzel megvan az $(n + 1)$ -szer $(n + 1)$ -es mátrix n -szer n -es része), az extra csúcsához tartozó sort pedig $O(f)$ lépésben fel tudjuk tölteni. **1 pont**

A kapott gráfnak $n = f + 1$ csúcsa van, a tanult algoritmus így $O(n^2) = O((f + 1)^2) = O(f^2)$ lépésben fut. **1 pont**

6. Dijkstra algoritmusát futtatjuk egy, az A, B, C, E, F, G csúcsokból álló irányítatlan gráfon. Az alábbi táblázat az $D[]$ tömb változását mutatja a futás közben. Melyek azok az élek, amik biztosan szerepelnek a gráfban és mi ezeknek a súlya?

	A	B	C	E	F	G
1.	0	∞	∞	3	∞	1
2.	0	∞	5	2	4	1
3.	0	7	3	2	4	1
4.	0	6	3	2	4	1
5.	0	5	3	2	4	1

Megoldás:

A futás A -ból indult, mert az A csúcsnál szerepel 0 az első sorban. **1 pont**

Az első sorból tudjuk, hogy A -ból E -be és G -be van él, ezek súlya rendre 3 és 1, mert az első sor a

kezdőcsúcsból kivezető élek súlyát adja meg.

1 + 1 pont

Kezdetben a KÉSZ halmaz csak az A pontból áll, később mindig a legkisebb $D[\]$ értékű bekerül a KÉSZ-be.

2 pont

A későbbi sorok vizsgálata során akkor lehet egy xy él meglétére következtetni, ha azt látjuk, hogy x KÉSZ-be kerülése után a következő sorra áttérve y értéke csökken és ekkor azt is tudjuk, hogy az xy él súlya $D[y]$ - az az érték, amivel x a KÉSZ-be került.

2 pont

(Ezt nem muszáj így általános alakban leírni, az is megfelelő, ha más módon derül ki, hogy hogyan következtetjük ki az éleket és a súlyukat (például egyesével elmagyarázzuk). Az a lényeg, hogy szükséges az indoklás és az indoklásból ki kell derülnie, hogy melyik értéket miből találtuk ki.)

Az 1. sor alapján G kerül a KÉSZ-be, a 2. sorra áttérve látjuk, hogy a biztosan meglevő élek és súlyaik: $c(GC) = 4, c(GE) = 1, c(GF) = 3$ és E kerül a KÉSZ-be.

1 pont

A 3. sorból látjuk, hogy $c(EB) = 5, c(EC) = 1$, és C kerül a KÉSZ-be,

1 pont

A 4. sorból kiderül, hogy $c(CB) = 3$, és F kerül a KÉSZ-be. Az 5. sorból pedig $c(FB) = 1$.

1 pont

7. Adott egy pozitív egész számokból álló összeg: $a_1 + a_2 + \dots + a_n$. Az összeadás jelek közül szorzásra cserélhetjük bármelyikeket, de csak úgy, hogy ne legyenek szomszédos szorzások (azaz minden szám legfeljebb egy szorzásban szerepelhet). Adjon $O(n)$ futásidejű algoritmust, ami meghatározza, hogy mekkora az ilyen cserékkel kapható számok maximuma. Például az $1 + 4 + 3 + 2 + 3 + 4 + 2$ összegből kapható maximum $29 = 1 + 4 \cdot 3 + 2 + 3 \cdot 4 + 2$.

Megoldás:

Dinamikus programozást használunk, minden $1 \leq i \leq n$ esetén a következő részfeladatot oldjuk meg:

$M[i]$ = az $a_1 + a_2 + \dots + a_i$ összeg módosításával elérhető legnagyobb olyan érték

2 pont

Ezeket a feladatokat $i = 1, 2, \dots, n$ sorrendben oldjuk meg.

1 pont

$M[1] = a_1$, mert ez az egyetlen érték, ami előállhat, illetve

$M[2] = \max(a_1 \cdot a_2, a_1 + a_2)$, mert ezek az egyetlen értékek, amik előállhatnak.

1 pont

Ha $i \geq 3$:

Két lehetőség van:

- az utolsó műveleti jel $+$: ekkor a legjobb, amit kaphatunk $M[i - 1] + a_i$
- az utolsó műveleti jel \cdot : ekkor a legjobb, amit kaphatunk $M[i - 2] + a_{i-1} \cdot a_i$, mert ekkor az utolsó előtti művelet biztosan $+$

3 pont

Ezek közül kell a nagyobbat venni, ez lesz $M[i]$ értéke.

1 pont

A keresett érték M definíciója szerint $M[n]$.

1 pont

A lépésszám azért $O(n)$, mert n feladatot oldunk meg és mindegyik konstans lépésben megvan.

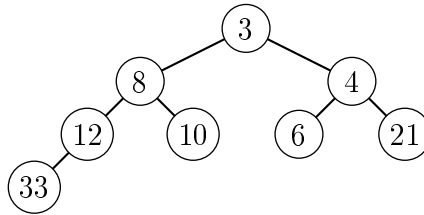
1 pont

A munkaidő 90 perc. A VÁLASZOKAT INDOKOLNI KELL.
Hivatkozni csak az előadáson tanultakra lehet.

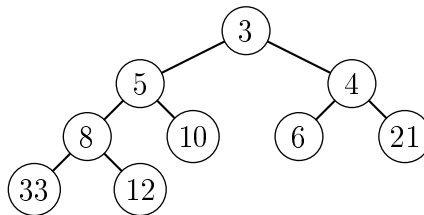
1. (a) Rajzolja fel a 3, 8, 4, 12, 10, 6, 21, 33 tömbbel adott kupacot bináris fa alakban. *(Itt indoklás nem szükséges.)*
(b) Szűrje be ebbe a kupacba (a fa alakban) az 5-t, majd hajtson végre egy MINTÖR-t, a megoldása során látszódnak az egyes lépései a műveleteknek.

Megoldás:

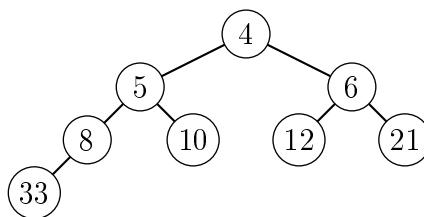
(a)



(b) Az 5-t a 12 jobb gyerekeként illesztjük be a fába, majd felfele mozgatjuk, felcserélve a szülővel addig, amíg már a szülője kisebb lesz nála, azaz 12-vel és 8-cal kell cseréni, de 3-mal már nem. Ezt kapjuk a végén:



A MINTÖR a 3 helyére teszi a 12-t, majd a 12-t addig cseréli fel a kisebb gyerekével, amíg már nem lesz magánál kisebb gyereke, ebben az esetben ez akkor következik be, amikor a 12 levélbe kerül, azaz a 4-gyel és a 6-tal kellett cserélni. Ezt kapjuk:



A pontozás vázlata

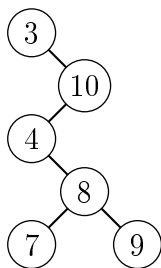
- (a) Jól felrajzolt fa: **2 pont**
(b) Jó helyre illesztte be az új elemet: **1 pont**
Kiderül, hogy felfele mozgatja az elemet (akár azért, mert lerajzolja lépésenként vagy jelöli a cseréket vagy leírja, hogy mi történik): **1 pont**
Jó kupac a végén: **2 pont**
MINTÖR-nél a 12-t felviszi a gyökérbe: **1 pont**
Kiderül, hogy lefele mozgatja az elemet (akár azért, mert lerajzolja lépésenként vagy jelöli a cseréket vagy leírja, hogy mi történik): **1 pont**
Jó kupac a végén: **2 pont**

2. Egy bináris keresőfa preorder bejárása során a fa csúcsait 3, 10, 4, 8, 7, 9 sorrendben látogatjuk meg. Rajzolja fel ezt a hat csúcsú bináris keresőfát, ahol ez megtörténhetett, majd lássa be, hogy a fa csak így nézhet ki.

Megoldás:

Jó fa:

3 pont



A gyökérben a 3-nak kell állnia, mert a preorder bejárás a gyökeret látja elsőnek.

A 3-tól balra nincsen senki, mert a bináris keresőfa tulajdonság miatt ott csak 3-nál kisebb számok lehetnek, de olyan nem szerepel. Tehát az összes többi elem a 3-as szám jobb fájában van és csak az a kérdés, hogy milyen elrendezésben.

A 10, 4, 8, 7, 9 számok közül a 10-et látjuk először, ezért 10 van a 3-as szám jobb részfájának a gyökerében és mindenki más a 10-től balra van, mert mindegyik szám kisebb nála.

A 4, 8, 7, 9 számok közül a 4-et látjuk először, ezért ez lesz a gyökérben és tőle jobbra van a másik három, mert nagyobbak nála.

A 8, 7, 9 számok közül pedig a 8-t látjuk először, ezért ez lesz a gyökérben és tőle balra van a 7 és jobbra a 9, megint csak a bináris keresőfa tulajdonság miatt.

Az indoklásban jól hivatkozza a preorder bejárást (a számok sorrendjében elől álló elem lesz a gyökérben):

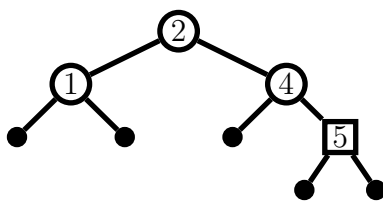
3 pont

Az indoklásban jól hivatkozza a bináris keresőfa tulajdonságot (a számok melyik oldalára esnek a gyökérnek):

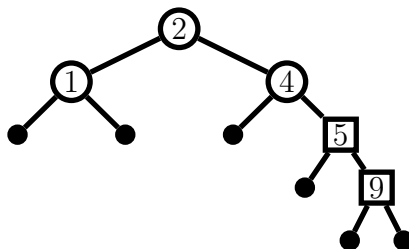
4 pont

Önmagában a preorder bejárás illetve a bináris keresőfa tulajdonság felírására nem jár pont, csak arra, ha ezekből a feladat szempontjából releváns következtetésekre jut a hallgató.

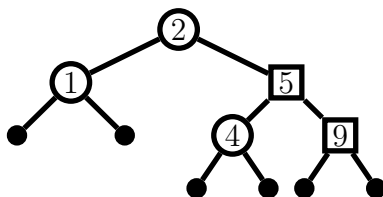
3. Szúrja be a következő piros-fekete fába a 9-et. (A \bigcirc csúcs fekete, a \square pedig piros.) A megoldásban látszódnak melyik műveleteket végzi, de indokolni nem kell.



Megoldás: Először beszúrjuk naiv módon a 9-et:

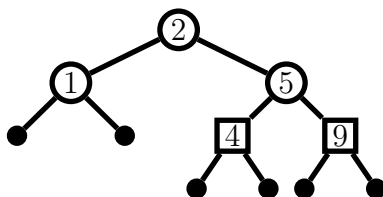


Az 5-ös csúcs piros, testvére fekete, ezért átszínezni nem lehet. Mivel az 5 és 9 azon oldali gyerek, forgatni kell, hogy az 5 feljebb kerüljön:



4 pont

Átszínezés:



3 pont

4. Egy kezdetben üres 2-3-fába az $1, 2, \dots, n$ számokat szűrtük be ebben a sorrendben. Bizonyítsa be, hogy a keletkezett fában a háromgyerekes csúcsok száma $O(\log n)$.

Megoldás: Háromgyerekes csúcs bármikor is csak úgy tud keletkezni, hogy egy kétgyerekes csúcsnak keletkezik harmadik gyereke (akár a levél szinten az új elem beszúrásával, akár feljebb). **2 pont**

A növekvő sorrend miatt a beszűrt elem mindig az aktuális fa legjobboldalibb eleme lesz, mert a levelek rendezettek, ezért a legalsó nem-levél szinten csak a jobboldali elem lehet harmadfokú.

2 pont

A fentebbi szinteken is csak a legjobboldalibb csúcs lehet háromgyerekes, mert a csúcsvágások is mindig a legjobboldalibb csúcsnál történnek. **3 pont**

Vagyis szintenként egy háromgyerekes lehet a keletkezett fában, de szintből $O(\log n)$ van a legvégén.

3 pont

A fa magasságának felírása csak akkor ér pontot, ha ez az érték valamilyen módon szerepet kap a megoldás során.

5. Lássa be, hogy az alábbi eldöntési probléma coNP-ben van:

Input: G egyszerű, összefüggő, irányítatlan gráf

Kérdés: Igaz-e, hogy G minden feszítőfájában van olyan csúcs, aminek a feszítőfában négy szomszédja van (azaz a csúcs a fában negyedfokú)?

Megoldás: Azt kell belátni, hogy a probléma komplementere NP-beli, azaz van rövid, gyorsan ellenőrizhető tanú a komplementer probléma „igen” inputjaira. (Vagy úgy is lehet mondani, hogy azt kell belátni, hogy van rövid, gyorsan ellenőrizhető tanú a „nem” inputokra.) **1 pont**

A probléma komplementerében az a kérdés, hogy igaz-e, hogy G -nek van olyan feszítőfája, amiben nincsen negyedfokú csúcs. (Vagy úgy is lehet mondani, hogy arra kell tanú, hogy egy G -nek van olyan feszítőfája, amiben minden csúcs foka 4-től különböző.) **2 pont**

Jó tanú lesz egy F feszítőfa. **2 pont**

A tanú mérete nem nagyobb, mint az input mérete, azaz $O(n)$ méretű. **1 pont**

Az ellenőrzés során ezeket kell megnézni: **2 pont**

- F minden éle szerepel G -ben?

- F összefüggő: BFS-sel eldönthető
- F -nek $v - 1$ éle van (ahol v a gráf pontszáma)?
- F -ben egyik fokszám sem 4?

A BFS $O(v + e) \subseteq O(n)$ lépés, a másik három ellenőrzés F éleinek egyszeri végigjárása során végrehajtható. Vagyis az ellenőrzés $O(n)$ lépés. **2 pont**

6. P-ben van vagy NP-teljes az alábbi NP-beli eldöntési feladat? (Azt a tényt fel szabad használni, hogy ez a feladat NP-ben van.)

Input: G irányítatlan gráf szomszédossági mátrixával és k pozitív egész szám

Kérdés: Igaz-e, hogy G -ben van k méretű klikk és k elemű független ponthalmaz is?

Megoldás: Adunk egy visszavezetést egy NP-teljes problémáról erre a feladatra, ez az NP-beliséggel együtt az Iszonyú Hasznos Tétel miatt már bizonyítja az NP-teljességet. **1 pont**

Jó választás lesz a MAXFTLN probléma. **1 pont**

Ha $k > v$, akkor $G' = G$ és $k' = k$ (azaz az input nem változik semmit), egyébként pedig a redukció során egy (G, k) párhoz azt a (G', k') párt rendeljük, amiben $k' = k$ és G' -t úgy kapjuk G -ből, hogy felveszünk G mellé egy k -as klikket, aminek minden csúcsát összekötjük G minden csúcsával. **3 pont**

(G', k') előállítás gyors, mert a k -as klikk hozzávétele $O(v^2)$ -ben megvan, ami biztosan $O(n)$. **1 pont**

Ha $k > v$ (ahol v a csúcsok száma G -ben), akkor nincsen se k -as független, se k -as klikk G -ben, azaz ilyenkor a redukció 2. pontjában kívánt ekvivalencia fennáll.

Az érdekes esetben ($k \leq v$) pedig:

- ha G -ben van k -as független ponthalmaz, akkor G' -ben van k -as független ponthalmaz és k -as klikk is, mert a k -as független ponthalmaz megmarad G -ben is, az újonnan felvett k -as klikk pedig biztosítja a k -as klikket. **2 pont**
- ha G' -ben van k -as klikk és k -as független ponthalmaz is, akkor a k -as független ponthalmaz G -n belül kell, hogy legyen, mert az új csúcsok minden G -belivel és egymással is össze vannak kötve. **2 pont**

Ha valaki nem kezeli a $k > v$ esetet külön, de ad egy jó redukciót $k \leq v$ esetre, ami minden részletében helyes, akkor az 9 pontot kap.

Ha MAXKLIKK-ről csinál valaki jó redukciót az is tökéletes persze (ekkor k izolált csúcsot kell felvenni G mellé).

7. Ha adott n szám, akkor hívjuk közülük középső elemnek a rendezés szerinti $\lceil n/2 \rceil$ -ediket. Kezdetben adottak az a_1, a_2, \dots, a_n egész számok, amikről tudjuk, hogy az a_1 a középső elem, egyébként a számok rendezetlenek. Ezekből építsen fel egy adatszerkezetet $O(n)$ összehasonlítás felhasználva úgy, hogy a felépített adatszerkezetben az alábbi két műveletet k tárolt elem esetén $O(\log k)$ lépésben lehessen valósítani:

BESZÚR: egy új elemet illeszt az adatszerkezetbe,

KÖZÉPTÖR: az aktuális középső elemet törli.

Megoldás: Két kupacot használunk, az egyik egy min-kupac (ahogy órán tanultuk), ebben lesznek a középső elemnél nagyobb számok, a másik pedig egy max-kupac, ebben lesz a középső elem és a nála kisebb számok. A max-kupacban az a tulajdonság áll fenn a számokra, hogy minden elem nagyobb, mint a gyerekei, a műveletek pedig hasonlóan megvalósíthatók, mint a min-kupacban. **2 pont**

Az a_1 -nél nagyobb elemekből a tanult módszerrel $O(n)$ -ben lehet megépíteni a min-kupacot, hasonlóan lehet megépíteni a max-kupacot is $O(n)$ -ben az a_1 -nél kisebb elemekből és a_1 -ből. **1 pont**

Mindig igaz, hogy a max-kupacban vagy ugyanannyi elem van, mint a min-kupacban vagy eggyel több és az is mindig igaz, hogy a max-kupac gyökerében a középső elem található. **1 pont**

BESZŰR esetén az új elemet összehasonlítom a max-kupac gyökerében levő középső elemmel és ha kisebb nála, akkor a max-kupacba, ha meg nagyobb nála, akkor a min-kupacba szűrom be. **1 pont**

Ha ezután a két kupac mérete több, mint eggyel eltér (azaz a különbség 2), akkor a nagyobb méretű kupacban (ez csak a max-kupac lehet) MAXTÖR-t csinállok (annak mintájára, ahogy a MINTÖR volt a min-kupacban), majd a kivett elemet beszűrom a min-kupacba (a tanult beszűrés eljárással).

Hasonlóan járok el akkor is, amikor a beszűrés után a min-kupac mérete nagyobb lesz a max-kupac méreténél: ekkor MINTÖR-t csinállok a min-kupacban, majd a kivett elemet beszűrom a max-kupacba. **1 pont**

Mindegyik lépés $O(\log k)$ -ban megvalósítható, mert a kupacok mérete $O(k)$. **1 pont**

KÖZÉPTÖR során a max-kupac gyökerét kell törölni, azaz egy MAXTÖR-t kell csinálni, majd ha a min-kupac mérete ettől nagyobbá válik, mint a max-kupacé, akkor egy hagyományos MINTÖR-t kell csinálni a min-kupacban, majd a kivett elemet be kell szűrni a max-kupacba. **2 pont**

Mindegyik lépés $O(\log k)$ -ban megvalósítható, mert a kupacok mérete $O(k)$. **1 pont**