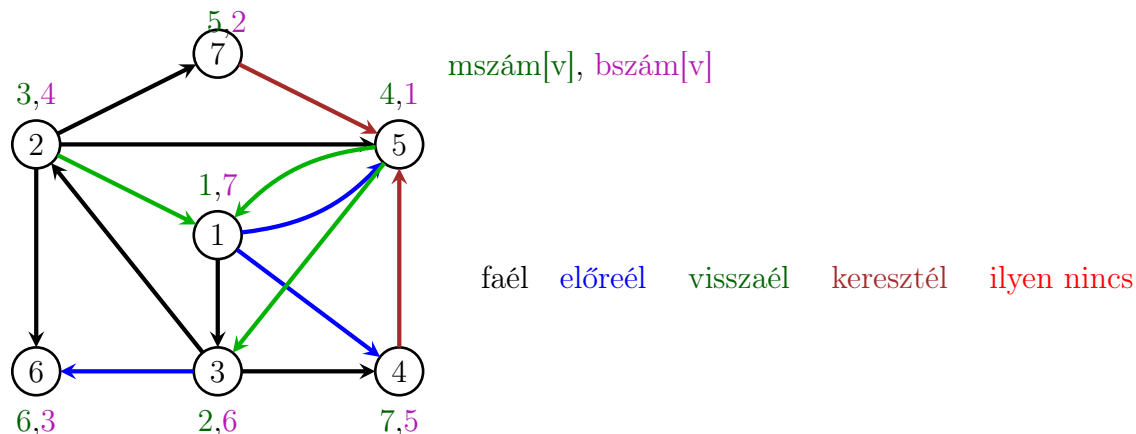


1. Futtassa le a DFS-MAIN(G)-t az alábbi (szomszédossági listával adott) irányított gráfban úgy, hogy a csúcsok és a szomszédok vizsgálatakor az éllistában adott sorrendet követi! Adja meg a csúcsok mélységi és befejezési számát is! **1**: 3, 4, 5; **2**: 1, 5, 7, 6; **3**: 2, 4, 6; **4**: 5; **5**: 3, 1; **6**: –; **7**: 5 (Eddig a múlt gyakorlaton már megcsináltuk.)
Mely élek lesznek faélek? Mely élek lesznek előreélek, visszaélek és keresztélek?

Megoldás:



2. Éllistájukkal adottak az alábbi G_1 és G_2 irányított gráfok.

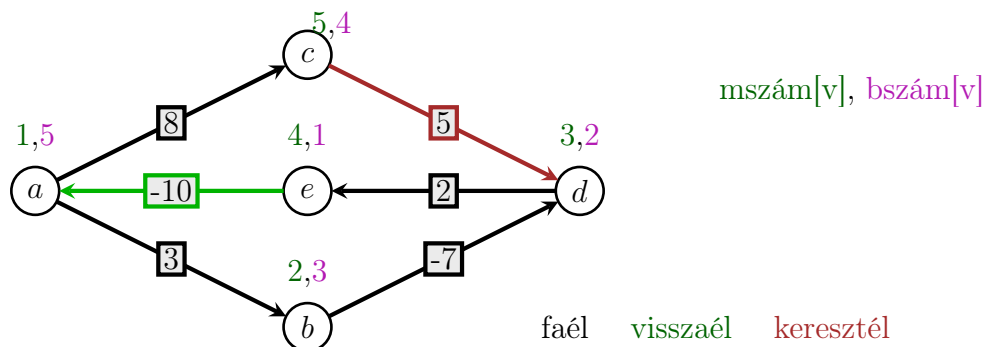
G_1 : **a**:b(3),c(8); **b**:d(-7); **c**:d(5); **d**:e(2); **e**:a(-10);

G_2 : **a**:g(2),f(10); **b**:a(-2),g(1); **c**:–; **d**:–; **e**:c(5),d(6); **f**:e(7); **g**:f(1), e(8);

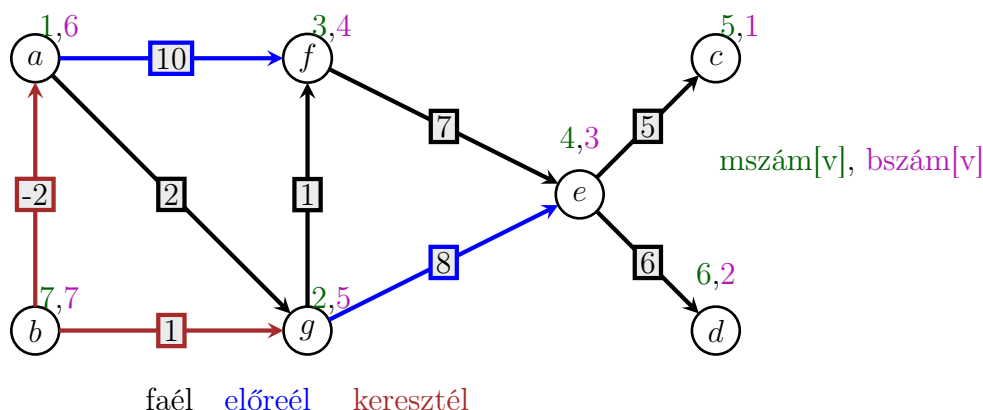
(a) Döntsük el mélységi bejárás segítségével erről a két gráfról, hogy DAG-e.

(b) Amelyik gráf DAG, abban adjunk meg egy topologikus sorrendet és határozzuk meg az a jelű csúcsból a többi csúcsba vezető legrövidebb utak hosszát.

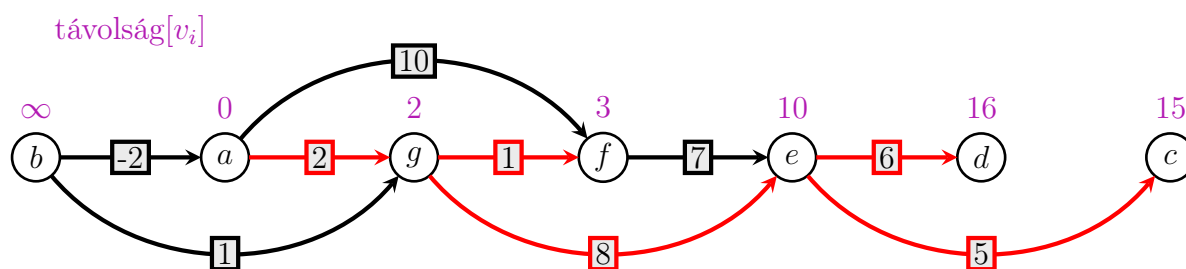
Megoldás: (a) Nem DAG, mert a DFS során találunk visszaélet.



(b) A gráf DAG, mert nincs visszaél.



Topologikus sorrend: b, a, g, f, e, d, c , amit a DFS futtatásával kapunk, ha a befejezési számok szerint csökkenő sorrendben írjuk fel a csúcsokat.



Az e csúcsnál a bejövő minimális él lehetne fe is ge helyett.

3. Egy éllistával adott irányított G gráfban néhány csúcs piros, néhány csúcs kék, a többi csúcs pedig színtelen. (A színezés egy, a csúcsokkal indexelt C tömbben adott). Melyik tanult algoritmus egyszeri futásával lehet $O(n + m)$ lépésben megoldani az alábbi feladatokat és hogyan?

- (a) El akarjuk dönteni, hogy az egyik adott p_1 piros csúcsból mik az elérhető kék csúcsok.
 (b) Egy adott piros p_1 csúcsához meg akarjuk keresni a legközelebbi kék csúcs(ka)t.

Megoldás: (a) Futassunk BFS-t vagy DFS-t p_1 -ből újratezdés nélkül. Azok a kék csúcsok alkotják a keresett halmazt, amelyek ezen futás során bejárttá válnak.

Ez azért igaz, mert mindkét algoritmus pontosan azokat a csúcsokat járja be (újratezdés nélkül) a kezdőcsúcsból, amikbe van irányított út a gráfban a kezdőcsúcsból.

Az eljárás $O(n + m)$, mert mindkét algoritmus futásideje ennyi és ezután már csak végig kell menni a bejárva tömbön és ki kell olvasni a bejárt kék csúcsokat, ennek lépésszáma pedig $O(n)$.

(b) Futassunk BFS-t p_1 -ből újratezdés nélkül, ennek végére minden csúcsra megkapjuk a p_1 -től való távolságot. Végigmenve ezután a távolságok tömbjén minimumot keresünk a kék csúcsokhoz tartozó értékek között, majd kiírjuk azokat a kék csúcsokat, ahol ez a megtalált legkisebb távolság felvevődött.

Ez azért jó, mert a BFS helyesen határozza meg a távolságokat akkor, amikor a távolság alatt az utat alkotó élek számát értjük.

Az eljárás $O(n + m)$, mert a BFS futásideje ennyi és ezután már csak végig kell menni a távolságok tömbjén egyszer a minimum megkereséséhez, majd még egyszer azon kékek kiírásához, akiknek ez a távolsága. Ez utóbbi két lépés $O(n)$ -s, ezért az egész futás $O(n + m)$.

4. Egy hat pontú irányított gráf csúcsait egy mélységi bejárás a, c, f, e, d, b sorrendben járja be, a befejezési számok pedig ezek: $a: 6; b: 5; c: 4; d: 3; e: 2; f: 1$.

Lehetséges-e, hogy a gráfban van él (a) f -ből e -be? (b) d -ből e -be?

Megoldás: (a) Mivel $mszám[f] = 3 < mszám[e] = 4$, ezért biztos, hogy amikor f -et bejárjuk, akkor e -t még nem látogattuk meg. Ha lenne él f -ből e -be, akkor f -ből még tudnánk tovább lépni (például e -be), így nem lehetne f az elsőnek befejezett csúcs.

(b) Ez lehetséges. Pl. ha a gráf éllistája $\mathbf{a} : c, b; \mathbf{b} : -; \mathbf{c} : f, e, d; \mathbf{d} : e; \mathbf{e} : -; \mathbf{f} : -$, akkor a DFS éppen a megadott mélységi és befejezési számokat adják. A (d, e) él ekkor keresztél lesz.

5. Éllistájával adott egy G irányított gráf, amiben nincsen irányított kör. Adott továbbá a gráf egy s csúcsa és szeretnénk meghatározni a gráf összes v csúcsa esetén az s -ből v -be vezető utak számát. Adjon erre a feladatra $O(n + m)$ lépésszámú algoritmust.

Megoldás: Jelöljük a gráfot $G = (V, E)$ -vel. Felveszünk egy, a csúcsokkal indexelt tömböt, amelyet **darab**-bal jelölünk. A célunk az, hogy **darab** $[v]$ az $s \rightarrow v$ utak számát adja meg. Ennek egy lehetséges módja a következő. Meghatározzuk a gráf egy topologikus sorrendjét. Ha a v csúcs s

előtt következik a topologikus sorrendben, akkor $\text{darab}[v] = 0$, mert ekkor nem lehet $s \rightarrow v$ út a topologikus sorrend definíciója szerint. Továbbá $\text{darab}[s] = 1$ (s -ből s -be pontosan egy út megy, az üres út, mivel a gráfban nincs irányított kör). Ezek ismeretében az s után következő v csúcsok esetén pedig a topologikus sorrendben előre felé haladva a

$$\text{darab}[v] = \sum_{u \in V : (u,v) \in E} \text{darab}[u] \quad (1)$$

összefüggés adja meg az $s \rightarrow v$ utak számát, ahol a $\text{darab}[u]$ -k már ismertek, hiszen ha van $u \rightarrow v$ él, akkor u biztos, hogy v előtt van a topologikus sorrendben. Ez a formula azért igaz, mert ha van $u \rightarrow v$ él, akkor s -ből u -n át v -be pontosan annyi út megy (olyan módon, hogy u -ból közvetlenül v -be lép), mint s -ből u -ba, így az u -kra összegezve megkapjuk az $s \rightarrow v$ utak számát (itt megint használtuk, hogy élek csak a topologikus sorrend szerint előre felé mehetnek).

Az algoritmus implementációja és lépésszáma: a topologikus sorrendről tudjuk, hogy $O(n + m)$ lépésben meghatározható. Az (1) lépéshez tudni kell, hogy melyik u -kból megy egy adott v -be él, amihez elkészítjük az eredeti éllistából a fordított éllistát, ami $O(n + m)$ lépés (lásd előző feladatsor 5. feladat). Ezután minden v csúcsnál legfeljebb annyi u csúcsra kell megnézni a darab -értékeket, ahány bejövő éle van v -nek, és ezeket csúcsonként összeadni, vagyis ez és az egész algoritmus is $O(n + m)$ lépés.

Megjegyzés: az (1) összefüggéssel definiált algoritmus tulajdonképpen dinamikus programozás, lásd később.

6. A 3. feladat folytatása:

(c) El akarjuk dönteni, hogy mindegyik kék csúcs elérhető-e legalább egy pirosból (azaz igaz-e, hogy minden kék csúcsra van olyan piros csúcs ahonnan ő elérhető).

(d) Meg akarjuk keresni a legrövidebb olyan utat a gráfban, ami piros csúcsból kék csúcsba vezet.

Megoldás: (c) Adjunk a G gráfhoz egy új v csúcsot. Ebből vezessünk (irányított) éleket a G minden piros csúcsához. A kapott gráf legyen G_1 . A G_1 éllistája megkapható G éllistájából $O(n + m)$ lépésben úgy, hogy felveszünk egy $n + 1$ méretű tömböt, aminek az első n cellájába átmásoljuk G éllistáját, az utolsó cellába pedig az új csúcs éleit rakjuk be. A csúcsok száma az új gráfban $n + 1$ lesz, az éleké legfeljebb $n + m$. Ezután bejárást indítunk G_1 -ben v -ből. Ez lehet DFS vagy BFS is. A v -ből elérhető kék csúcsok K halmazában pontosan azok a kék csúcsok lesznek, amelyek G -ben piros csúcsból elérhetők, mert a G -beli piros csúcsból induló utak megfelelnek a G_1 -beli v -ből induló utaknak. A K -beli csúcsokat megjelöljük, és végül ellenőrizzük - ez $O(n)$ munka - hogy K az összes kék csúcs halmaza-e. Az új éllista elkészítése $O(n + m)$, a bejárás költsége $O(1 + n + n + m) = O(n + m)$, a kék csúcsok ellenőrzése pedig $O(n)$.

(d) Itt az előzőekben felépített G_1 gráfban v -ből induló BFS-t hajtunk végre. Minden w csúcs mellé odaírjuk a v -től való G_1 -beli távolságát, legyen ez a távolság d_w . Ez eddig $O(n + m)$ munka. Ezt követően meghatározzuk, a $d_w - 1$ mennyiségek minimumát, ahol w befutja a kék csúcsok halmazát. Ez további $O(n)$ munka. Ezután még a megtalált minimumhoz tartozó utat is meg kell találnunk, de ezt meg tudjuk határozni, mert minden csúcsra feljegyeztük a BFS során, hogy melyik csúcsból ért ide a legrövidebb út és ezen élek mentén vissza tudunk lépkedni v -be. Ennek költsége is $O(n)$, mert legfeljebb n csúcsból áll a legrövidebb út, az algoritmus összköltsége tehát $O(n + m)$. Azért helyes az algoritmus, mert a G -beli pirosból kékbe vezető utak pont egy éllel rövidebbek, mint a G_1 -beli v -ből induló kékbe érkező utak.

7. Cirkuszi akrobaták egymás vállára állva minél nagyobb tornyot szeretnének létrehozni (a toronyban minden szinten csak egy akrobata lesz). Esztétikai és gyakorlati szempontok miatt egy ember vállára csak olyan állhat, aki nála alacsonyabb és könnyebb is. A cirkuszban n akrobata van, adott mindegyikük magassága és súlya.

(a) Adjon algoritmust, amely $O(n^2)$ lépésben megadja a lehetséges legtöbb emberből álló torony összeállítását.

(b) Adjon algoritmust, amely $O(n^2)$ lépésben megadja a lehetséges legmagasabb torony (hány centiméter magas) összeállítását.

Megoldás: Vegyünk egy olyan G gráfot, melyben a csúcsok az akrobatáknak felelnek meg és két tetszőleges $(v_i$ és $v_j)$ csúcs között pontosan akkor fut irányított él, ha a j . akrobata állhat az i . akrobata vállán. Vegyünk fel továbbá egy f (mint föld) csúcsot, húzzunk belőle élt minden másik csúcsba (a föld vállán bárki állhat). Vegyük észre, hogy G gráf DAG (az egymásra állás feltételei miatt) és a benne lévő, f -ből induló utak megfeleltethetők tornyoknak. Úgy kell ügyeskednünk az élek súlyozásával, hogy az ilyen utak hossza pont a tornyok (kérdezett) magassága legyen. Ezután mindkét probléma megoldható G -ben lefuttatott f -ből leghosszabb út kereséssel.

a) Legyen minden él súlya 1. Az a torony, aminek a tetején az i . akrobata áll, legfeljebb $\text{tavolsag}[v_i]$ ember magas lehet (mivel f -ből indulva nem létezik $\text{tavolsag}[v_i] - 1$ -nél több csúcsból (akrobatából) álló út v_i -be). Keressük meg a legnagyobb $\text{tavolsag}[v_i]$ -t, ehhez fog tartozni a legmagasabb torony. Maga a torony úgy kapható meg, hogy feljegyezzük a leghosszabb út keresés során minden csúcsnál azt is, hogy honnan érkezett a leghosszabb út és ennek mentén vissza tudjuk követni, hogy melyik akrobaták állnak a toronyban.

b) Legyen minden $v_i \rightarrow v_j$ él súlya a j . akrobata magassága, és hasonlóan az $f \rightarrow v_j$ él súlya is legyen a j . akrobata magassága. Ekkor egy út hossza éppen meg fog egyezni az úton található akrobaták magasságának összegével, így az a torony, aminek a tetején az i . akrobata áll, legfeljebb $\text{tavolsag}[v_i]$ magas lesz (mivel f -ből indulva nem létezik hosszabb (összesen magasabb akrobatából) álló út v_i -be). Keressük meg a legnagyobb $\text{tavolsag}[v_i]$ -t, majd magát a tornyot ugyanúgy kapjuk, mint az (a) részben.

A gráf szomszédossági mátrixának (vagy éllistájának) megadása $O(n^2)$ lépés, mert minden akrobatapárt meg kell vizsgálnunk, a leghosszabb út keresés $O(n^2)$ vagy $O(n + m)$, a megadás módjától függően, a maximum megtalálása pedig $O(n)$ lépés. Mivel a gráf egyszerű, $m \in O(n^2)$, így a teljes algoritmusok is beleférnek $O(n^2)$ -be.

Megoldás részletesebben: (a)

Az algoritmus leírása:

1. Hozzunk létre egy élsúlyozott gráfot G , ahol a csomópontok az akrobatákat reprezentálják. Él vezet akrobata_i -ből akrobata_j -be, ha akrobata_j alacsonyabb és könnyebb akrobata_i -nél (azaz akrobata_j állhat akrobata_i vállán), és az él súlya akrobata_j magasságával egyezik meg.
2. Használjunk mélységi keresést (DFS) és befejezési időt a G topologikus sorrendjének meghatározására. (A helyességi részben látni fogjuk, hogy G egy irányított aciklikus gráf (DAG), így létezik topologikus rendezés.)
3. Adjunk hozzá egy segéd csomópontot G -hez, és húzzunk éleket ettől az összes akrobatához, ahol az élek súlya az adott akrobata magassága. Ezt a gráfot jelöljük G' -vel.
4. Használjuk a **leghosszabb út megtalálása élsúlyozott DAG-ban algoritmust** a segéd csomópontból G' -ben a leghosszabb út meghatározására. A talált legnagyobb érték adja a lehető legmagasabb torony magasságát.

Helyesség:

G egy DAG, mivel ha létezne irányított kör, akkor találhatnánk olyan akrobatákat $a_1, a_2, \dots, a_l$, ahol a_{i+1} alacsonyabb, mint a_i ($1 \leq i \leq l - 1$), és a_l alacsonyabb, mint a_1 , ami ellentmondás, hiszen így a_1 kisebb lenne önmagánál. Ez bizonyítja, hogy létezik topologikus sorrend, így alkalmazhatjuk a keresett algoritmust. A G' -ben a segéd csomópontból kiinduló utak megfelelnek az akrobaták tornyainak, és az ilyen utak hossza megegyezik a megfelelő torony magasságával. Az algoritmus a leghosszabb ilyen utat találja meg, így helyes az eredmény.

Futási idő:

1. Összehasonlítjuk az összes akrobatapárt, azaz $\binom{n}{2}$ párt, így az $O(n^2)$ idő alatt elkészíthető G szomszédsági listája. G n csúcsból és $O(n^2)$ élből áll.
2. A DFS futási ideje $O(n + m)$ (ha a gráfot szomszédsági listával adjuk meg), így összesen $O(n^2)$, mivel $m \leq n^2$.
3. $O(n)$ idő a G' szomszédsági listájának létrehozására (1 csúcs és n él hozzáadása).
4. A **leghosszabb út megtalálása élsúlyozott DAG-ban algoritmus** futási ideje $O(n' + m')$, ahol $n' = n + 1$, $m' = m + n \leq n^2 + n$, így összesen $O(n^2)$.

(b): Az optimális torony visszafejtéséhez tároljuk minden akrobatánál az előző akrobatát, amelyből a legjobb magasság adódott. Így visszafelé haladva rekonstruálható a teljes torony.

Második megoldás az (a) részre:

Az algoritmus leírása:

1. Rendezzük az akrobatákat növekvő súly szerint. Ha két akrobatának azonos a súlya, akkor növekvő magasság szerint rendezzük őket.
2. Alkalmazzunk egy dinamikus programozási megoldást, amelyben $\text{tor}[i]$ az i -edik akrobatával végződő legmagasabb torony magasságát tárolja.
3. Kezdetben minden $\text{tor}[i]$ értéket az i -edik akrobata magasságára állítunk.
4. Minden i -re végigmegyünk az előző $j < i$ indexeken, és ha akrobata_j magassága kisebb, mint akrobata_i magassága, akkor frissítjük $\text{tor}[i]$ értékét:

$$\text{tor}[i] = \max(\text{tor}[i], \text{tor}[j] + \text{magassg}[i]) \quad (2)$$

5. Az optimális torony magassága a tor tömb maximuma.

Helyesség:

Mivel először növekvő súly szerint rendeztük az akrobatákat, így garantáljuk, hogy egy későbbi akrobata csak egy korábbira állhat, ha a magassági feltétel is teljesül. A dinamikus programozás garantálja, hogy az eddig elérhető legmagasabb tornyot építjük fel lépésenként.

Futási idő:

1. Az akrobaták rendezése $O(n \log n)$ időben történik.
2. A dinamikus programozás minden i esetén végignézi az összes előző akrobatát, így össz $O(n^2)$ időt igényel.
3. Az optimális torony meghatározása $O(n)$ időt vesz igénybe.
Azaz összesen a futásidő $O(n^2)$.

8. Egy irányított G gráfban hagyjuk el a forrásokat ($d_{be}(v) = 0$) és a nyelőket ($d_{ki}(v) = 0$). A maradék gráfban ismét hagyjuk el a forrásokat és a nyelőket, ezt ismételjük, amíg lehet. Bizonyítsuk be, hogy akkor és csak akkor kapunk üres gráfot, ha G -ben nincs irányított kör!

Megoldás: Ha G -ben van egy irányított kör, akkor annak egyik csúcsa sem forrás és nem is nyelő, és ez akkor sem változik, ha a gráfból elhagyjuk a forrásokat és a nyelőket, tehát G -t a források és nyelők elhagyásának ismétlésével nem lehet kiüríteni.

Ha viszont G -ben nincs irányított kör, vegyük G egy topologikus sorrendjét. A topologikus sorrend szerinti első csúcs mindig forrás, az utolsó mindig nyelő, így ezeket az eljárás első lépésében mindenképpen elhagyjuk (elképzeltethető, hogy ezek egybeestek, de csak akkor, ha a gráf egyetlen izolált

pontból állt). Ha a források és nyelők elhagyása után még van csúcsa a gráfnak, akkor az eredeti topologikus sorrend szerinti első meghagyott csúcs mindenképpen forrássá válik, mert aki előtte volt a topologikus sorrendben, azt elhagytuk, így belé már nem mehet él. Hasonlóan az utolsó meghagyott csúcs nyelővé válik, mert belőle kifelé már nem mehet él. Így ezeket a következő lépésben szintén el fogjuk hagyni (az összes többi, forrássá vagy nyelővé vált csúccsal együtt). Ezért minden lépésben el tudunk hagyni legalább két csúcsot, kivéve ha már csak egyetlen (izolált) csúcsa van a gráfnak, amit szintén el tudunk hagyni. Így a gráfot az eljárást legfeljebb $\lfloor n/2 \rfloor + 1$ lépésben ismételve ki lehet üríteni.

9. Legyen adott egy $n \times n$ pixelből álló fekete-fehér kép. Szeretnénk a képen a bal felső saroktól a jobb alsó sarokig egy jobbra-lefele haladó határvonalat húzni úgy, hogy a vonaltól jobbra-felfele eső fekete, valamint a vonaltól balra-lefele eső fehér pixelek számának az összege a lehető legkisebb legyen. Oldjuk meg ezt a feladatot $O(n^2)$ időben!

Megoldás: Készítsünk egy gráfot (éllistas megadást), amiben csúcsok a pixelek (mint téglalapok) sarkai és élek a pixelek határai. A függőlegesek élsúlya legyen a tőle jobbra levő fekete és a tőle balra levő fehér pixelek számának összege. A vízszintes élek súlya viszont legyen 0. Minden függőleges élet irányítsunk lefelé, a vízszinteseket pedig jobbra. Így definiálva az élek súlyát, látható, hogy a bal felső saroktól a jobb alsó sarokig jobbra-lefele haladó határvonal összsúlya éppen a minimalizálandó szám, mert minden rossz helyen levő pixelt pontosan egyszer számolok meg.

Az élek irányításából következően a gráf DAG. Csúcsainak száma $(n+1)^2 = O(n^2)$, éleinek száma legfeljebb kétszer ennyi (mert minden csúcsból legfeljebb két él lép ki), azaz $m = O(n^2)$. A függőleges élek súlyát meg tudjuk határozni $O(n^2)$ lépésben, ha soronként balról jobbra haladunk. Egy sorban először megszámoljuk a fekete pixelek számát ($O(n)$), ez lesz a bal szélső él súlya. Az i -edik pixel jobb oldalán lévő él súlya pedig 1-gyel kisebb a bal oldalán levőénél, ha a pixel fekete, és 1-gyel nagyobb, ha fehér. Így $O(n)$ lépésben megkapjuk egy sorban a súlyokat, azaz az összes élsúly is megvan $O(n^2)$ lépésben. A gráfban megkereshetjük a minimális súlyú utat $O(n' + m') = O(n^2)$ lépésben.

10. Egy városban 1000 ember lakik, mindenki minden nap elmondja az ismerőseinek az összes előző nap megtudott hírt (Az ismeretségek kölcsönösek.) Tudjuk, hogy olyan az ismeretségek hálózata, hogy előbb-utóbb mindenki megtud mindent. Bizonyítsuk be, hogy van 90 olyan ember, hogy ha ők egyszerre megtudnak valamit, akkor legkésőbb 10 nap múlva mindenki megtudja!

Megoldás: Tekintsük az ismeretségek által meghatározott (irányítatlan) gráfot. Ebben létezik egy leghosszabb út, $P = (x_0, x_1, \dots, x_d)$. Futtassuk le a DFS(G, x_0) algoritmust. Mivel előbb-utóbb mindenki megtud mindent, ezért a gráf összefüggő, így a DFS egy fát eredményez. Fontos észrevétel, hogy P élei mind faélek lesznek, az út egyik vége x_0 , a másik pedig x_d a DFS-fa egyik levele. (Ha a csúcsokat úgy rendezzük sorba, hogy először $x_0, x_1, \dots, x_d$ jön, akkor a DFS először végigmegy ezen a P -n és a x_d -nek 1 lesz a befejezési száma.) A feszítőfában minden más pontba egyetlen út vezet, ennek éleinek száma az illető pont távolsága x_0 -tól a fában. Legyen V_i a x_0 -tól a fában i távolságra lévő pontok halmaza, ezek egymástól páronként diszjunkt halmazok. Ha a V_i halmazban lévő emberek megtudnak egy hírt, akkor 10 nap alatt megtudják a $V_{i+1}, \dots, V_{i+10}$ halmazban levők is. Ráadásul azt is belátjuk, hogy ha $i \leq 10$, akkor a $V_0, V_1, \dots, V_{i-1}$ halmazokban levők is. Tegyük fel, hogy van közülük olyan aki nem, legyen ez y . Tudjuk, hogy $x_i \in V_i$. Ez azt jelenti, hogy a fában x_i és y távolsága nagyobb, mint 10. Így az $x_d, x_{d-1}, \dots, x_i$ utat az y -ba vezető úttal folytatva egy hosszabb utat kapnánk, mint a leghosszabb út, ami ellentmondás.

Legyen $k = 0, 1, 2, \dots, 10$ -re $W_k = V_k \cup V_{k+11} \cup V_{k+22} \cup \dots$. Mivel a V_i halmazok páronként diszjunktak, a W_k halmazok is azok. Ha valamelyik napon W_k -ban mindenki megtudja a hírt, akkor 10 nap múlva mindenki meg fogja tudni. Tudjuk, hogy $\sum_{k=0}^{10} |W_k| = 1000$. Nem lehet, hogy minden $|W_k| \geq 91$, hiszen $91 \cdot 11 = 1001 > 1000$. Vagyis valamelyik $|W_k| \leq 90$. Nekik kell elmondani a hírt.

2. megoldás (Drótos Márton):

Mivel a hír végül mindenkinek eljut, az ismeretségi hálózat összefüggő. Tekintsük a hálózat csúcs-pontjait (az embereket) és vegyünk egy T gyökeres feszítőfát az 1000 csúcson tetszőleges gyökérrel. A gyökértől mért távolság (azaz a szint) alapján rendezzük a csúcsokat.

Most az alábbi eljárást alkalmazzuk az S halmaz (a kiválasztott személyek halmaza) kialakítására:

- Amíg T tartalmaz olyan csúcsot, melynek szintje legalább 10, válasszunk ki egy olyan v csúcsot, amely a jelenleg elérhető legmélyebb szinten van.
- Az v -ből haladva a gyökér irányába, válasszuk ki azt a csúcsot, amely pontosan 10 szinttel feljebb van. Jelöljük ezt a csúcsot u -val (ha v szintje $L \geq 10$, akkor u szintje $L - 10$).
- Adjuk hozzá u -t az S halmazhoz, majd távolítsuk el T -ből az u gyökerű induló teljes részfat.

Ha a maradék feszítőfa legmélyebb szintje kevesebb, mint 10, akkor válasszuk ki a maradék feszítőfa gyökerét is, és tegyük az S halmazba.

Mivel minden egyes lépés során az u -ból induló részfa legalább 11 csúcsot tartalmaz (hiszen a u és a v közötti út $10 + 1 = 11$ csúcsot érint), így a folyamat során legfeljebb 90 csúcsot választunk ki, mielőtt a feszítőfa tetején legfeljebb 10 csúcs maradna. Viszont őket biztosan el tudjuk érni az utolsó kiválasztott csúcsból.

Ez azt jelenti, hogy ha egyszerre az S halmaz minden tagja megtud egy hírt, akkor az információ naponta terjed a szomszédos csúcsok között, így legkésőbb 10 nap múlva a hír eljut az egész feszítőfára, azaz az egész hálózatra.

Megjegyzés: 89 embernek nem mindig elég elmondani a hírt. Vegyünk egy teljes gráfot 100 ponton, 90 pontjához $(v_1, \dots, v_{90})$ csatoljunk egy-egy 10 hosszú utat (ez 10-10 új pont). Összesen 1000 pont lesz. Viszont, ha valamelyik v_i és a hozzá tartozó úton egyetlen ember sem tudja meg a hírt az első nap, akkor az út végén lévő ember biztos nem fogja megtudni 10 nap múlva. Mivel 90 ilyen út van (v_i -t az úthoz számolva), mindegyiken valakinek kell tudni a hírt.

11. A 10. feladat megoldására a ChatGPT a következő választ adta. Vizsgáld meg, hogy jó-e ez a megoldás!

ChatGPT: Tekintsük a várost egy **gráfként**, ahol a csúcsok az embereket, az élek pedig az ismeretségeket reprezentálják. Mivel mindenki minden nap elmondja az összes eddig megtudott hírt az ismerőseinek, a hírterjedés egy **diszkrét dinamikus folyamat** a gráfon. Az adott feltétel szerint a gráf **erősen kapcsolt**, azaz előbb-utóbb mindenkihez eljut minden hír.

A hírterjedés sebessége

Egy ember egy nap alatt az összes ismerősének továbbadja a hírt. Ez azt jelenti, hogy ha egy adott napon egy csúcshalmaz birtokában van egy hírnek, akkor a következő napra a szomszédai is megkapják azt. A terjedés egy **hullámszerű folyamat**, amely naponta egy-egy lépésben halad előre a gráfban.

A terjedési idő becslése

A hírterjedés maximális idejét a gráf **átmérője** becsüli meg. Az átmérő, jelölve mint D , a **leg-hosszabb legrövidebb út** a gráf bármely két csúcsa között:

$$D = \max_{u,v \in V} d(u,v), \quad (3)$$

ahol $d(u,v)$ az u és v közötti legrövidebb út hossza. Kisvilág-hálózatok esetében, mint az emberi kapcsolatok hálózata, az átmérő tipikusan kicsi, azaz $O(\log n)$. Mivel $n = 1000$, az átmérő jellemzően 6 és 10 között van.

Létezik 90 fős kezdőcsoport 10 lépéses lefedéssel

A célunk az, hogy találjunk **90 olyan embert**, akik egy hír megtudásakor **legfeljebb 10 nap alatt mindenkihez eljuttatják azt**.

Mivel a gráf átmérője legfeljebb 10, minden ember legfeljebb 10 lépésre van a hálózat többi tagjától. Ha tehát egy adott pillanatban egy jól elosztott S méretű halmaz, ahol $|S| = 90$, egyszerre birtokolja a hírt, akkor a hír az átmérőnyi idő alatt az egész hálózatban elterjed.

Mivel a 90 fős csoportot úgy választhatjuk meg, hogy megfelelő lefedést adjon a hálózatban, biztosítható, hogy az információ legfeljebb 10 lépés alatt elér mindenkit.

Következtetés

Mivel a gráf átmérője legfeljebb 10, és létezik egy olyan 90 fős kezdőcsoport, amely hatékonyan lefedi a hálózatot, következik, hogy ha ez a 90 ember egy adott napon megtud egy hírt, akkor **legfeljebb 10 nap múlva mindenkihez eljut**. Ez bizonyítja az állítást.

Megoldás: A ChatGPT megoldása azon alapul, hogy az ismeretségek által meghatározott gráf rendelkezik az úgynevezett *kisvilág-tulajdonsággal*. Megfigyelések azt mutatják, hogy a való életben az ilyen hálózatokban az átmérő kicsi, azaz bármely két pont távolsága kicsi. Hogy pontosan mekkora, az persze mindig függ a konkrét hálózattól, de sokszor emlegetik a 6-os átmérőt nagyon nagy gráfokra is. (https://en.wikipedia.org/wiki/Small-world_network)

Ez tehát nem egy bizonyított matematikai állítás, itt nem használható. A gyakorlat kontextusából viszont az is világos, hogy a feladatot nem csak ilyen gráfokra, hanem tetszőleges gráfra be szeretnénk látni. Ezzel viszont a ChatGPT nem is próbálkozik.

Ha tényleg feltesszük, hogy csak a legfeljebb 6 (vagy 10) átmérőjű gráfokra akarunk bizonyítást, akkor viszont a megoldás annyi, hogy bármely embernek elmondjuk a hírt, akkor 6 (vagy 10) nap alatt megtudja mindenki. Vagyis semmi szükség a ChatGPT által keresett „jól elosztott” S halmazra. (Azt egyébként nem is mondja meg, hogy mit is jelent ez és hogy találunk ilyet. Tehát a ChatGPT megoldása még ilyen értelmezésben is erősen hiányos.)

Érdekes, hogy ha felhívjuk a ChatGPT figyelmét rá, hogy mi tetszőleges gráfra meg akarjuk oldani a problémát, akkor is lényegében ugyanezt a megoldást javasolja, csak akkor semmilyen értelmezésben nem jó a bizonyítás.