

1. Rendezze a következő szavakat a radix rendezés segítségével: abc , acb , bca , bbc , acc , bac , baa .

Megoldás:

bca , baa , acb , abc , bbc , acc , bac

baa , bac , abc , bbc , bca , acb , acc

abc , acb , acc , baa , bac , bbc , bca

2. Honnan látszik A szomszédossági mátrixból, hogy hány csúcsa és hány éle van az irányítatlan gráfnak? Honnan látszanak a fokszámok?

Megoldás: Annyi csúcs van, ahány sor, azaz 5. $d(v_i)$ az i -edik sorban lévő 1-esek száma. $d(v_1) = 2$, $d(v_2) = 4$, $d(v_3) = 2$, $d(v_4) = d(v_5) = 3$. Az élek száma a fokszámtétel szerint a fokszámok összegének fele, tehát 7.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3. (a) Honnan látjuk, hogy a B szomszédossági mátrix irányított gráfhoz tartozik?

(b) Honnan látszik, hogy mennyi az egyes csúcsok ki- és be-foka?

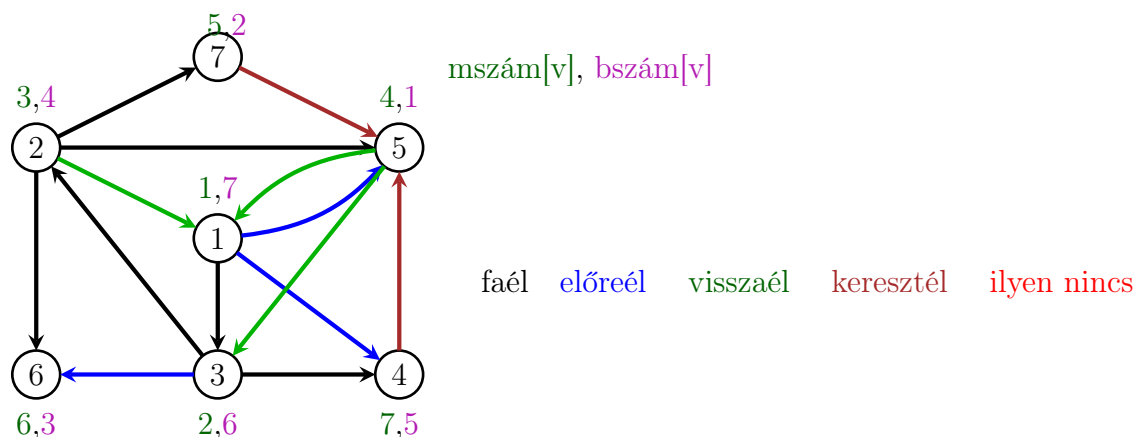
Megoldás: (a) Az irányítatlan gráfok szomszédossági mátrixa szimmetrikus, ez nem az. (b) $d_{ki}(v_i)$ az i -edik sorban lévő 1-esek száma. $d_{ki}(v_1) = 2$, $d_{ki}(v_2) = 1$, $d_{ki}(v_3) = 3$, $d_{ki}(v_4) = 1$, $d_{ki}(v_5) = 0$. $d_{be}(v_i)$ az i -edik oszlopban lévő 1-esek száma. $d_{be}(v_1) = 0$, $d_{be}(v_2) = 2$, $d_{be}(v_3) = 1$, $d_{be}(v_4) = d_{be}(v_5) = 2$.

4. Mennyi az egyes csúcsok ki-foka az **a**: b, c ; **b**: e, d ; **c**: d ; **d**: f ; **e**: f ; **f**: – szomszédossági lista által adott irányított gráfban? Mennyi az f csúcs be-foka?

Megoldás: A kifokok rendre: 2, 2, 1, 1, 1, 0. Az f befoka: 2.

5. 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

Megoldás:



6. Adjon egy $O(n)$ időigényű algoritmust n olyan egész számból álló sorozat rendezésére, melynek elemei az

(a) $\{1, \dots, 3n\}$ halmazból vannak!

(b) $\{1, \dots, n^3 - 1\}$ halmazból vannak!

Megoldás: (a) Használjunk ládarendezést $3n$ ládával! Ekkor a lépésszám $O(n + 3n) = O(n)$.

(b) Az egyszerű ládarendezés itt nem elég, használjuk a radix rendezést! Képzeld el a számokat az n alapú számrendszerben felírva. Ekkor mindegyik (legfeljebb) 3 jegyű. Ha ebben az alakban használjuk a radix rendezést, akkor 3 ládarendezés kell, mindegyiknél n láda, tehát az egész valóban $O(n)$ lépést használ.

Megjegyzés: Az n alapú számrendszerbe való átírás is megoldható $O(n)$ aritmetikai művelettel, hiszen minden számot maradékosan el kell osztani n -nel, a maradék adja az utolsó jegyet, a hányadost újra elosztjuk n -nel, az itteni maradék adja a következő számjegyet, a hányados meg az elsőt.

7. Egy gráf fordított éllistájában (szomszédossági listájában) minden csúcsnál a csúcsba bejövő élek vannak felsorolva. Adjon $O(n + m)$ lépésszámú algoritmust, ami az éllistából elkészíti a fordított éllistát.

Megoldás: A fordított éllistához először lemásoljuk a csúcsok tömbjét és mindegyikhez egy üres listát rendelünk. Ez eddig $O(n)$ lépés. Ez most azt a gráfot írja le, amiben ugyanazok a pontok vannak, de még egyetlen él sincs benne. Ezután végigmegyünk az eredeti éllistán. Amikor egy új (x, y) élet találunk, akkor a fordított éllistához hozzáadjuk az (y, x) életet, azaz az y csúcs listájának elejére illesztjük az x csúcsot. Egy ilyen beszúrás $O(1)$ lépést igényel (a láncolt lista elejére tesszük). Mivel eredetileg m él van, ez összesen $O(m)$ lépés, vagyis a teljes lépésszám $O(n + m)$.

8. Egy szomszédossági listával adott irányított G gráfban mindegyik csúcs színes, vagy piros vagy zöld (ez az információ egy, a csúcsokkal indexelt C tömbben adott) és adott a gráfban egy piros s és egy piros t csúcs. (a) Adjon $O(n + m)$ lépésszámú algoritmust, ami eldönti, hogy van-e az s -ből t -be olyan út, ami csak piros csúcsokon megy át. (b) Adjon $O(n(n + m))$ lépésszámú algoritmust, ami eldönti, hogy van-e s -ből t -be olyan út, ami legfeljebb egy zöld csúcson megy át.

Megoldás: (a) Először átalakítjuk a gráfot úgy, hogy a zöld csúcsok ne legyenek elérhetők, aztán a kapott új G_1 gráfban futtatjuk a DFS(G_1, s) mélységi bejárást az s csúcsból és ha közben elérjük t -t, akkor van ilyen út, különben pedig nincsen.

Lépésszám: A gráf átalakítása úgy történik, hogy végigmegyünk az éllistán és minden olyan élet törölök, aminek valamelyik vége zöld. Ez az éllista egyszeri végigjárásával megoldható és a szomszédok listájából egy él törlése konstans lépés, azaz ez összesen $O(n + m)$, az ezutáni DFS-ben pedig egy n csúcsú, legfeljebb m élű gráffal dolgozok, azaz ezen rész lépésszáma is $O(n + m)$.

Az algoritmus helyessége: Az új gráfban pontosan akkor van út s és t között, ha az eredetiben van csupa piros ilyen út, azt pedig a DFS jól eldönti, hogy G_1 -ben t elérhető-e s -ből.

(b) A gráf minden zöld z csúcsára megcsináljuk a következőt: pirossá nyilvánítjuk a z csúcsot és futtatjuk az (a) pont algoritmusát. Ezzel kiderítjük azt, hogy van-e olyan út s -ből t -be, amin csak eredetileg is piros csúcsok, illetve esetleg még z szerepel. Ez az eljárás minden zöld csúcstra $O(n + m)$ lépés, legfeljebb n -szer csináljuk meg, azaz összesen $O(n(n + m))$ lépés.

Megoldás részletesebben:

Jótanács: Mindig próbáljuk meg először az input módosításával elérni a kívánt eredményt. Ha az algoritmust módosítjuk, akkor az algoritmus helyességét és futási idejét újra bizonyítani kell. Ezért, ha van egy algoritmikus ötletünk, amit látszólag az algoritmus módosításával érhetnénk el, próbáljuk meg az input módosításával megoldani a problémát.

(a)

Az algoritmus leírása

1. Módosítsuk a gráfot úgy, hogy végigmegyünk a szomszédossági éllistán, és töröljük az összes nem piros csúcsot, valamint az ezekhez tartozó éleket.

2. Futtassunk mélységi bejárást (DFS) az így kapott G' gráfon, kiindulva az s csúcsból, és határozzuk meg, hogy létezik-e irányított út s -ből t -be G' -ben. Ha igen, akkor G -ben is van olyan út, amely kizárólag piros csúcsokat tartalmaz; ha nem, akkor nincs ilyen út.

Helyesség

- Ha G -ben létezik olyan út, amely csak piros csúcsokat tartalmaz, akkor ez az út megmarad G' -ben, így a DFS hatására az algoritmus talál egy csak piros irányított utat s -ből t -be.
- Fordítva, ha a DFS megtalál egy csak piros irányított utat s -ből t -be G' -ben, akkor az út eredetileg G -ben is létezett.

Futásidő

1. Az éllistán való végigmenetel, mely során minden csúcsot és az ezekhez tartozó éleket egyszer ellenőrizzük, $O(n + m)$ időt vesz igénybe.
2. A DFS futtatása G' -n, amely legfeljebb n csúcsból és m élből álló gráfon történik, szintén $O(n + m)$.
3. Így összességében az algoritmus futási ideje: $O(n + m)$.

(b)

Az algoritmus leírása

Először végigmegyünk a gráf szomszédossági éllistáján, és összegyűjtjük az összes zöld csúcsot. Ezt követően a következő stratégiát alkalmazzuk: minden egyes zöld z csúcsra megvizsgáljuk, hogy létezik-e olyan s -ből t -be vezető irányított út, amelyben z az egyetlen zöld csúcs, vagy akár egyáltalán nem fordulnak elő zöld csúcsok.

A konkrét megvalósítás lépései:

1. Vizsgáljuk azokat az utakat, amelyek tartalmazhatnak zöld csúcsot:
 - a) Minden zöld z csúcs esetén hozzunk létre egy G_z gráfot, amelyben z kivételével töröljük G -ből az összes zöld csúcsot az (a) részben ismertetett módon.
 - b) Minden G_z -ben futtassunk két DFS-t: az egyik s -ből z -be, a másik z -ből t -be. Amennyiben mindkét irányban létezik út, úgy a feladat megoldása IGEN.
2. Az (a) részben ismertetett módszerrel határozzuk meg, hogy létezik-e olyan út s -ből t -be G -ben, amely kizárólag piros csúcsokat tartalmaz. Ha igen, a válasz szintén IGEN.

Amennyiben bármelyik előző lépésben IGEN a válasz, a feladat eredménye IGEN, különben NEM.

Helyesség

- Ha létezik egy s -ből z -be vezető út, majd egy z -ből t -be vezető út, ezek egymás után "rakásával" egy s -ből t -be vezető út állítható elő. Bár az utak esetleg átmetszhetnek, az így keletkező irányított körök "levághatók" a két út uniójáról, így valóban egy egyszerű irányított út jön létre s -ből t -be, amely legfeljebb egy zöld pontot (z -t) tartalmaz.

- Ezután az érvelés analóg az (a) rész helyességi bizonyításával: ha G -ben létezik olyan s -ből t -be vezető út, amely legfeljebb egy zöld csúcsot tartalmaz, ezt G_z -nél algoritmusunk megtalálja.
- Fordítva, ha az algoritmus talál két megfelelő utat G_z -ben utat, akkor a fenti megjegyzés értelmében szerepelni fog egy, a feladat által kívánt út G -ben.

Futásidő

1. Az éllistán végighaladva és az összes G_z és G' gráfok előállítására legfeljebb $O((n-1)(m+n)) = O(n(m+n))$, hiszen a zöld pontos számossága legfeljebb $n-2$.
 2. Minden G_z gráfokon 2db DFS futtatása és G' -n egy darab DFS futtatása. Mind a G_z -k, mind G' legfeljebb n csúcsból és m élből áll, így az összes DFS futási ideje legfeljebb $(2n+1) \cdot O(m+n)$.
 3. Összességében az algoritmus futási ideje: $O((3n+1)(m+n)) = O(n(m+n))$.
9. Egy szomszédossági mátrixával adott irányított G gráfban szeretnénk meghatározni az összes olyan csúcsot, ahonnan egy adott t csúcs irányított úton elérhető. Adjon erre a feladatra $O(n^2)$ lépésszámú algoritmust.

Megoldás: Ha t elérhető irányított úton s -ből, akkor ha minden él irányát megfordítjuk, akkor s elérhető lesz irányított úton t -ből. Ennek a gráfnak a szomszédossági mátrixát könnyen előállíthatjuk az eredeti szomszédossági mátrixból, vegyük a transzponáltját. Az új gráfban futtassuk a $\text{DFS}(G, t)$ eljárást. A transzponált előállítása és a DFS lépésszáma is $O(n^2)$, ezért az össz lépésszám is $O(n^2)$. Ha a gráf éllistával lenne megadva, akkor az 5. feladat megoldását felhasználva hasonlóan belátható, hogy $O(n+m)$ lépésben megkaphatjuk a keresett csúcsokat.

10. Egy kezdő autóvezető a városban való közlekedése során szeretne gyakorlatának megfelelő útvonalat választani. Az úthálózat egy irányítatlan gráfként van megadva, a csúcsok a kereszteződések, az élek az utak, a csúcsoknál adott, hogy nehéz-e számára az a kereszteződés. (Az hogy nehéz, a kereszteződés tulajdonsága, nem azon múlik, merről érkezik oda és merre akar rajta áthaladni.)

Adjon algoritmust, amivel meg lehet határozni, hogy az autós az egyik adott csúcsnál levő otthonából mely csúcsokba tud autóval úgy eljutni, hogy útja során két nehéz csúcs soha nem jön közvetlenül egymás után. Az algoritmus lépésszáma szomszédossági listás megadás esetén legyen $O(n+m)$.

Megoldás: Töröljük ki az éllistából azokat az éleket, amelyek nehéz csomópontból nehéz csomópontba vezető utaknak felelnek meg. Ez az éllista egyszeri végigjárásával az éllista méretében lineáris lépésszámmal, tehát $O(n+m)$ lépésben megtehető. Ezután az így kapott éllistájú új G' gráfban futtassuk le a $\text{DFS}(G', v)$ algoritmust, ahol v a kezdő vezető otthonának megfelelő csúcs. G' -nek $n' = n$ csúcsa és $m' \leq m$ éle van, tehát a DFS $O(n' + m') = O(n+m)$ lépést vesz igénybe. Az algoritmus által elért csúcsok éppen azok, amelyekbe v -ből vezet út G' -ben, vagyis pont azok, amelyekbe az eredeti G gráfnak megfelelő eredeti úthálózatban v -ből el lehet jutni úgy, hogy ne következzen egymás után két nehéz kereszteződés.

11. Adjon $O(n+m)$ lépésszámú algoritmust a 8(b) feladatra.

Megoldás: Először lefuttatjuk a 8(a) feladat algoritmusát, hogy eldöntsük, hogy van-e csupa piros út. Ha van, akkor készen vagyunk.

Ezután, az algoritmus második részében eldöntjük, hogy van-e pontosan egy zöld csúcsot tartalmazó út úgy, hogy meghatározzuk azokat a zöld csúcsokat, akik s -ből elérhetők és akikből t is elérhető. Ehhez először töröljük a zöld csúcsokból kimenő éleket, ez $O(n+m)$ lépésben megtehető. Az így kapott G_1 gráfban lefuttatjuk a $\text{DFS}(G_1, s)$ algoritmust és az s -ből elérhető zöld csúcsokat feljegyezzük egy T_1 tömbben ($T_1 = 1$, ha a csúcs elérhető volt és zöld, különben meg $T_1 = 0$), ennek lépésszáma

$O(n + m')$, ahol m' az új gráf élszáma, de $m' \leq m$, ezért ez $O(n + m)$. Mivel a kimenő éleket töröltük a G gráfból, így pontosan azokat a zöld csúcsokat fedezi fel a DFS, amik piros csúcsokon keresztül elérhetőek.

A következő lépésben megfordítjuk G éleinek irányát és az új gráfban töröljük a zöld csúcsokból kimenő éleket. Legyen az így kapott gráf G_2 . Most a $\text{DFS}(G_2, t)$ algoritmust futtatva megkapjuk azon zöld csúcsokat, amik t -ből elérhetőek irányított úton, ezeket feljegyezzük egy T_2 tömbben. Az élek megfordítása miatt minden T_2 -ben 1-re állított zöld csúcsból van az eredeti gráfban irányított út t -be, továbbá a kimenő élek törlésével ez az út csak piros csúcsokon megy keresztül.

Ezután végigmegyünk a T_1 és T_2 tömbökön, és ha találunk olyan v csúcsot, amire $T_1[v] = T_2[v] = 1$, akkor s -ből el tudunk jutni irányított úton csak piros csúcsokat érintve ebbe a zöld v csúcsba, majd onnan t -be piros csúcsokon keresztül. Ha nincs ilyen metszet, akkor nincs ilyen irányított út pontosan egy zöld csúccsal. Mivel a gráfok módosítása ($O(n + m)$) és DFS kétszeri futtatása (mindegyik $O(n + m)$) után csak a két tömbön kell végighaladni és ez $O(n)$, így a lépésszám $O(m + n)$.

2. Megoldás: Vegyünk még egy példányát a G gráfnak, legyen ez G' . Minden G -beli v csúcsnak lesz megfelelője G' -ben, jelölje ezt v' . Ezután G -ben minden zöld csúcsból kimenő (v, w) élet töröljük ki, viszont adjuk hozzá helyette a (v, w') élet, majd G' -ben töröljük a zöld csúcsokból kimenő összes élet. Az így kapott gráf legyen H . Futtassuk a $\text{DFS}(H, s)$ algoritmust. Ha ez talál irányított utat t -be, vagy t' -be, akkor az egy megfelelő út lesz.

Lépésszám: H pontszáma $2n$, élszáma legfeljebb $2m$, ugyanis először minden élet megkettőzünk, majd néhányat áthelyezünk, végül G' -ből esetleg törünk is éleket. Ezen a mélységi keresés futtatása $O(2n + 2m) = O(n + m)$ lépés.

Helyesség: Ha van irányított út s -ből t -be, akkor az végig a módosított G -ben haladt, mivel G' -ből G -be nem lehet visszatérni. Mivel G -n belül zöld csúcsból nincs kimenő él, ezért az ilyen út nem tartalmaz zöld csúcsot. Ha van irányított út s -ből t' -be, akkor az egyszer kénytelen átlépni G -ből G -be és ilyenkor egy zöld ponton halad át. Mivel visszalépni már nem lehet G -be, és G' -ben nincs zöld csúcsból induló él, ezért az ilyen út pontosan egy zöld csúcsot tartalmaz. Könnyen látható a fordított irány is, azaz ha van megfelelő út az eredeti G gráfban, akkor lesz egy megfelelő út H -ban is.

Megjegyzés: Ennek a módszernek az alapötlete alkalmazható akkor is, ha olyan utat keresünk, amelyik legfeljebb $2, 3, \dots$ zöld csúcsot tartalmaz. (Ilyenkor több példány kell a gráfból.) Sőt kis módosítással akkor is, ha olyan utakat keresünk, ami pontosan $1, 2, 3, \dots$ zöld csúcsot tartalmaz. A lépésszám mindig $O(n + m)$ marad, feltéve, hogy az előírt zöld csúcsok száma konstans, nem függ n -től.

Megoldás részletesebben: Algoritmus leírása

Az első fázisban lefuttatjuk a 7(a) feladat algoritmusát, hogy eldöntsük, van-e csupa piros út. Ha ez a feltétel teljesül, akkor a probléma meg van oldva, a válasz: IGEN.

Ezután az algoritmus második fázisban eldöntjük, hogy létezik-e olyan út, amely pontosan egy zöld csúcsot tartalmaz. Ehhez meg kell határoznunk azokat a zöld csúcsokat, akik:

- (i) s -ből elérhetőek csak piros úton, és (ii) t -be vezet belőlük irányított, csak piros út.

A részletes lépések a következők:

(a) (i) rész: Az s -ből piros csúcsokon elérhető zöld csúcsok meghatározása

- i. Töröljük a zöld csúcsokból kimenő éleket az eredeti G gráfból, így kapjuk G_1 gráfot.

- ii. Az így kapott G_1 gráfban futtassuk le a $\text{DFS}(G_1, s)$ algoritmust. A bejárás során minden elérhető csúcs esetében, ha az csúcs zöld, akkor azt feljegyezzük egy T_1 tömbbe úgy, hogy

$$T_1[v] = \begin{cases} 1, & \text{ha } v \text{ elérhető és zöld,} \\ 0, & \text{különben.} \end{cases}$$

(b) (ii) rész: A t -ből elérhető zöld csúcsok meghatározása

- i. Fordítsuk meg az eredeti G gráf éleinek irányát, azaz gyártsuk le a fordított éllistát, mint azt láttuk 7 megoldásában.
- ii. Az új, megfordított gráfban ismét töröljük a zöld csúcsokból kimenő éleket, így kapva a G_2 gráfot.
- iii. Futtassuk le a $\text{DFS}(G_2, t)$ algoritmust, hogy meghatározzuk azon zöld csúcsokat, amelyeket elérhetünk t -ből G_2 -ben csak piros csúcsokon keresztül, azaz azon zöld pontok halmaza, melyekből t elérhető irányított piros úton. Ezeket jegyezzük fel egy T_2 tömbben, azaz

$$T_2[v] = \begin{cases} 1, & \text{ha } v \text{ elérhető és zöld,} \\ 0, & \text{különben.} \end{cases}$$

(c) (iii) rész: Végző ellenőrzés

- Végigmegyünk a T_1 és T_2 tömbökön. Ha találunk olyan v csúcsot, amelyre:

$$T_1[v] = T_2[v] = 1,$$

akkor algoritmusunk outputja IGEN.

Ha sem az első, sem a második fázis során nem kaptunk IGEN-t, akkor algoritmusunk outputja: NEM.

Helyesség

A 8. feladatra vonatkozó helyességi gondolatmenet adaptálható itt is.

Futásidő

- Első fázis: 8. feladatnál látottak alapján: $O(m + n)$
- Második fázis:
 - (i)/(a) $O(m + n)$
 - (i)/(b) $O(m' + n)$, ahol $m' \leq m$, így $O(m + n)$
 - (ii)/(a) $O(m + n)$, ahogy láttuk a 7 feladatban
 - (ii)/(b) $O(m + n)$
 - (ii)/(c) $O(m' + n)$, ahol $m' \leq m$, így $O(m + n)$
 - (iii) $O(n)$

Így összesen $O(m + n)$.

12. Adjunk algoritmust, mely egy éllistával megadott irányítatlan gráfban vagy talál egy kört, vagy igazolja a gráf körmentességét $O(n)$ időben (függetlenül attól, hogy m akár sokkal nagyobb is lehet, mint n)!

Megoldás: Módosítsuk a mélységi bejáró algoritmust úgy, hogy álljunk meg, ha a soron következő él mentén már bejárt csúcsba jutunk. Ha ez nem következik be, akkor a gráf körmentes. Ha igen, akkor az él visszaél és a meglevő feszítő erdőben a végpontjai között menő úttal együtt kört alkot. Bármely eset is következik be, a bejárás legfeljebb n él megtekintése után véget ér, hiszen egy erdőnk és esetleg még egy élünk van. A szélességi bejárás hasonlóan módosítható.

13. A szomszédossági listával adott egy összefüggő, irányított G gráf, melynek minden éle az $1, 2, \dots, k$ egész számok valamelyikével van súlyozva. Egy út értéke legyen az úton található élek súlyainak *maximuma*. Adott a gráf két csúcsa, x és y . Adjunk $O(m \log k)$ lépésszámú algoritmust annak meghatározására, hogy mennyi a lehető legkisebb értékű x -ből y -ba vezető út értéke.

Megoldás: Egy bináris kereséshez hasonló algoritmust fogunk használni. Először vizsgáljuk meg, hogy a $\frac{k}{2}$ -nél nagyobb súlyú élek eltávolításakor keletkező gráfban van-e irányított út x -ből y -ba. Ha van út az új módosított gráfban x -ből y -ba, akkor megvizsgáljuk, hogy kisebb élsúlyok esetén is lehet-e út x és y között, így a folyamatot megismételjük $\frac{k}{4}$ -re. Ha a $\frac{k}{2}$ -nél nagyobb súlyú élek törlésével már nem lesz út x és y között, akkor növeljük a keresendő súlyokat. Ebben az esetben a $\frac{3k}{4}$ -nél nagyobb súlyú éleket töröljük és folytatjuk az algoritmust. Az eljárás biztosan megtalálja a minimális l súlyú utat x és y között, mivel ha létezne egy legfeljebb $(l-1)$ súlyú út, akkor az $(l-1)$ súlyú éleknél nagyobb súlyok törlésével kapott gráfban x -ből y elérhető lenne, így az algoritmus nem az l súlyú úttal térne vissza. A lépésszáma $O(m \log k)$, mivel minden felezésnél legfeljebb m él törlését végezzük el, majd DFS-t futtatunk $O(n+m)$ lépés alatt, ezt pedig $\log k$ -szor hajtjuk végre.