Eötvös Loránd Tudományegyetem
Természettudományi Kar

Gráfokkal megoldható hétköznapi problémák

Szakdolgozat

Készítette
Vincze Ágnes Melitta

Konzulens
Héger Tamás

Budapest, 2015

Tartalomjegyzék

Bevezetés

Szakdolgozatom témájának olyan témakört szerettem volna választani, ami a mindennapi életünkben előfordul, valamint fontos szempont volt az is, hogy középiskolában egy szakkör, vagy fakultáció keretén belül megemlíthető legyen. Tapasztalataim alapján a diákokban gyakran merül fel a kérdés, hogy egy bizonyos témakört miért is kell nekik tanulniuk, tudják-e majd hasznosítani a későbbiekben a megszerzett tudást.

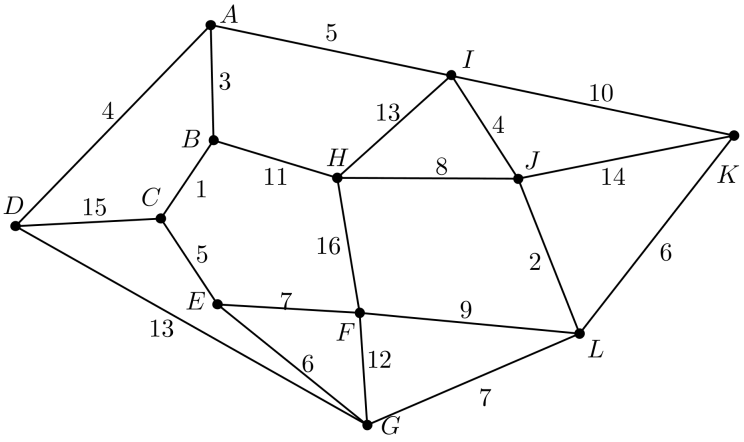
A dolgozatom fő témája a különböző útkeresési problémák, célom az, hogy megmutassam ezek sokszínűségét, valamint belátást biztosítsak megoldási hátterükbe. A hétköznapi életben ezek gyakran felmerülő kérdések, hiszen akár csak a nagymamát, vagy valamelyik rokonunkat szeretnénk meglátogatni, el kell döntenünk, hogy melyik útvonalat válasszuk. Egy nyaralás alkalmával is fontos szempont lehet a városok, múzeumok meglátogatási sorrendje úgy, hogy a körutazást a legrövidebb, legolcsóbb módon hajtsuk végre. Az útvonaltervezés során számolnunk kell a különböző utak megtételéhez szükséges idővel, benzinköltséggel, valamint egyéb felmerülő költségekkel, mint például az autópálya, vagy komphasználati díj. Fontos kérdés még, hogy például út-, vagy vízvezetékhálózat megépítése során milyen szakaszokon és milyen sorrendben építsük meg azokat.

A dolgozatom során először a minimális költségvetésű feszítőfákat tárgyalom, majd a legrövidebb utak, körutak keresésével fogok foglalkozni. Az egyszerűség kedvéért általában az egyszerű, irányítatlan gráfokat vizsgálom.

1. fejezet

Minimális költségű feszítőfa keresése

A király úgy dönt, hogy az országa városai közt kövezett úthálózatot építtet, hogy könnyebben közelíthessék meg azokat. Azonban a kincstárban kevés pénz van, ezért a népéhez fordult segítségért, így szólva hozzájuk: „Alattvalóim! Nagy jutalmat ajánlok annak, aki elkészíti nekem birodalmam leendő úthálózatának a térképét úgy, hogy az a legkedvezőbb legyen számomra. Ehhez segítségetekre lesz az a térkép, amelyet tanácsadóm készített a különböző utak megépítésének költségeivel. Fontos, hogy minden városomból minden városomba el lehessen jutni a leendő utakon haladva.”



1.1. ábra. Útépítési költségek

Egy optimista Kruskal[1] nevű alattvaló úgy gondolja, királya számára az lesz a legkedvezőbb, ha a legolcsóbb úthálózatot építi meg, mégpedig úgy, hogy költségük szerint növekvő sorban kellene megépíteni azokat. Amint összegyűjti a pénz egy útra, azt nyomban megépíti. Nyilván mindig a legolcsóbb utat fogja építeni, de ha a következő legolcsóbb út két olyan várost kötne össze, amelyek közt már lehetne kövezett utakon közlekedni, akkor inkább spórol a következő legolcsóbbra. Így állítása szerint, az ő úthálózata összesen 51 aranyába kerülne a királynak. Az alattvaló a fentiek alapján először nyilvánvalóan az 1 aranyba kerülő utat építi meg B és C város közt, aztután 2 aranyért L és J között, 3 aranyért A és B között. Amikor 4 aranyért építené az utat, észreveszi, hogy két ilyen út is van, ekkor tetszőleges sorrendben építi meg mindkettőt egymás után. Továbbiakban hasonlóan jár el az $5, 6$ aranyba kerülő utaknál is, majd a 7 aranyba kerülő útnál észreveszi, hogy hiába kerül két út is ugyanúgy 7 aranyba, csak az E, F városok közötti utat tudja megépíteni, mivel a G, L városok közt lévő út kört alkotna A, B, C, E, G, L, J, I városokkal. Végül már csak a 8 aranyba kerülő utat kell megépíteni ahhoz, hogy minden várost meg lehessen közelíteni a megépített úton haladva.

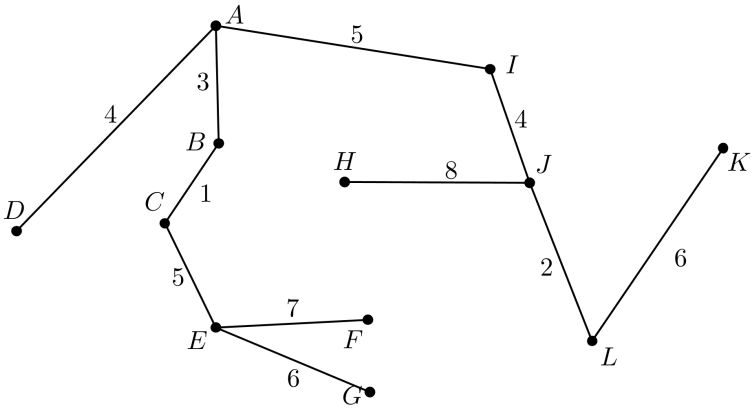
A pesszimista alattvaló szintén a legolcsóbb úthálózat megépítésére törekedve azt mondja, azért, hogy ne fordulhasson elő az, hogy egy nagyon drága utat szükségtelenül megépítünk, először jegyezzük fel, hogy mely utak azok, amiket biztosan nem fogunk megépíteni, mert drágák. Ezt addig tudja megtenni, míg már kénytelen egy utat megépíteni, mivel anélkül az út nélkül nem lehetne bizonyos városba eljutni kövezett utakon. Tehát megkeresi a legnagyobb költségűt, ez a 16 aranyba kerülő út H és F város között, és feljegyzi a nem megépítendő utak listájára. Ezek után a következő legdrágább úttal is így tesz, ez a 15 aranyba kerülő út D és C város közt. A következő legdrágább út (13 aranyért) keresésénél észreveszi, hogy két ilyen út is van, de mivel egyik építése sem szükséges, így mindkettőt feljegyzi. Tovább vizsgálva a városokat szintén feljegyzésre kerülnek a $12, 11, 10, 9$ aranyba kerülő utak. Azonban mikor eljut a 8 aranyba kerülő úthoz, észreveszi, hogy ezt meg kell építeni, mivel a H városba már nincs több szóba jöhető út. A 7 aranyba kerülő utaknál rájön, hogy az E, F városok között szintén meg kell építenie az utat, de G és L között még nem. A következő legdrágább út a 6, az 5, valamint a 4 aranyba kerülő, és sajnos mindhárom esetben mindkét utat meg kell építeni. Végül szomorúan veszi észre, hogy már nincs olyan út,

[1]Joseph Kruskal amerikai matematikus 1956-ban publikálta algoritmusát.

amit feljegyezhetne a nem megépítendő utak listájára, tehát az összes többi utat meg kell építeni. Az általa tervezett úthálózat szintén 51 aranyba kerül.

Egy Prim[2] nevű alattvaló úgy gondolja, hogy a legkedvezőbb a király számára, ha nem csak a legolcsóbb úthálózatot építik meg, hanem ha a fővárosból (A csúcs) indítja el az útépítést, és az utakat úgy építik, hogy mindig kapcsolódjanak egy már megépített úthoz. Így az építőanyag szállítása a már elkészült úton egyszerűbb lesz, mintha a földúton haladnának. Tehát a következő építési sorrendet javasolja: először építsék meg az utat 3 aranyért A és B város közt, 1 aranyért B és C között, 4 aranyért A, D között, majd tetszőleges sorrendben választhat az 5 aranyba kerülő út építésénél, hogy C, E, vagy A, I városok közti utat építsék előbb. Ezek után 4 aranyért építik I, J között, 2 aranyért J, L között, majd a 6 aranyba kerülő utakat szintén tetszőleges sorrendben építhetik, végül a 7, majd a 8 aranyba kerülő utakat építik meg. Tehát ez az úthálózat is 51 aranyba kerül.

A király úgy dönt, mivel mindhárom alattvaló ötlete azonos összegbe kerül, megnézi az általuk készített térképet a leendő úthálózatokról és ezek alapján dönt.



1.2. ábra. A három alattvaló úthálózata

Azonban mikor a király kezébe kerülnek a térképek, észreveszi, hogy mindhárom térkép megegyezik. A döntését az egyszerűbb építőanyag szállításra hivatkozva hozza meg, Prim alattvaló ötletét alkalmazva. Ekkor felmerült benne a kérdés, hogy tudná-e még kevesebb aranyból építeni az úthálózatot.

[2]Robert C. Prim amerikai matematikus 1957-ben írta le az algoritmusát.

A kérdés megválaszolásához következőkben tekintsünk a király országára gráfként és feleltessük meg a városokat csúcsoknak, a köztük lévő utakat pedig éleknek.

1.0.1. Definíció. Egy $G(V, E)$ gráf egy élsúlyozásán egy $w : E \to \mathbb{R}$ függvényt értünk. Ekkor egy f él súlyán (vagy költségén) a $w(f)$ értéket értjük. Egy $F \subseteq E(G)$ élhalmaz súlya $w(F) = \sum_{f \in F} w(f)$.

Vegyük észre, hogy a definícióban megengedünk negatív súlyokat is, de általában nemnegatív súlyokat alkalmazunk, mivel gyakran az alkalmazásokban csak erre van szükség, valamint az eljárások egy része ezt meg is követeli.

Olyan részgráfra van szükségünk, amely által bármely csúcsból bármely csúcs elérhető az éleken haladva, azaz egy összefüggő gráfra. Amennyiben a gráfban két csúcs között már van út, úgy nincs szükség köztük az él behúzására, hiszen célunk a költségek minimalizálása. Tehát egy olyan összefüggő részgráfot keresünk, amelyben nincsen kör, azaz fa gráfot. Mivel az élek mentén minden csúcsnak elérhetőnek kell lennie, vagyis az eredeti gráf összes csúcsát tartalmazza, így feladatunk egy minimális költségű feszítőfa keresése lesz. Erre a Véges matematika I című kurzuson már láttunk egy megoldást az optimista, illetve a pesszimista stratégia alkalmazásával, amit most felelevenítünk. Az optimista stratégia során a gráfban az éleket költségük szerint növekvő sorrendben húzzuk be úgy, hogy kör ne keletkezzen. A pesszimista stratégia pedig az, hogy töröljük ki a gráfból az éleket költségük növekvő sorrendjében, kivéve, ha a gráf szétesne, ekkor kénytelenek vagyunk behúzni azt az élt, amivel a gráf még összefüggő marad. Az előadáson azt is beláttuk, hogy az optimista és a pesszimista alattvaló is egy legolcsóbb feszítőfát fog megépíteni, azonban ennek a bizonyításnak az ismétlésétől most eltekintünk.

A következőkben nézzünk egy általánosabb, e két eljárás vegyes alkalmazását erre a feladatra, majd ennek a speciális eseteit tárgyaljuk különféle motivációkkal.

1.1. Piros-kék eljárás

Ezek a fejezetek főként a [9], valamint a [4] könyv alapján készültek.

Az eljárás során a gráf olyan éleit színezzük ki kékre, amelyek biztosan benne lesznek a készítendő minimális költségű feszítőfában, illetve pirosra azokat, amelyek semmiképp sem lesznek benne.

Kék szabály:	Vegyünk egy olyan nem üres X csúcshalmazt, melyből nem vezet ki kék él. Ekkor az X-ből kivezető élek közül az egyik színtelen minimális költségűt színezzük kékre.
Piros szabály:	Vegyünk a gráfban egy olyan kört, melyben nincs piros színű él. Ebben az egyik maximális költségű színtelen él pirosra színezhető.
Piros-kék eljárás:	Kezdetben egy színezetlen gráfból indulunk ki, majd az eljárás során a két szabályt tetszőleges sorrendben alkalmazzuk, amíg lehetséges.

A következőkben azt fogjuk megmutatni, hogy ha veszünk egy színezetlen gráfot és a két szabályt tetszőleges sorrendben alkalmazzuk rá, akkor végül a kék élek egy minimális költségű feszítő fát fognak alkotni.

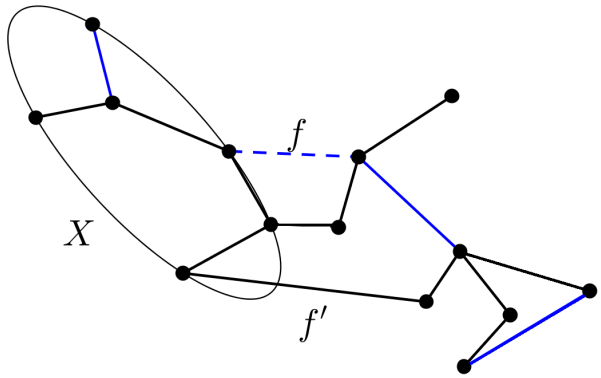
1.1.1. Definíció. Vegyük a gráfnak egy olyan élszínezését, melyben piros, kék és színtelen éleket engedünk meg. A színezés akkor *takaros*, ha van olyan minimális költségű feszítőfa, amely az összes kék élt tartalmazza, de nem tartalmaz egy piros élt sem.

1.1.2. Tétel. *A piros-kék eljárás működése során mindig takaros színezésünk van. Ezen felül a színezéssel sosem akadunk el : végül a gráf minden éle színes lesz. Ekkor a kék élek egy minimális költségű feszítőfát alkotnak.*

Bizonyítás. Legyen $G = (V, E)$ egy tetszőleges élsúlyokkal ellátott n csúcsú gráf. Induljunk ki egy takaros színezésből, például az megfelelő lesz, ahol minden él színtelen. Először megmutatjuk, hogy ha egy takaros színezésben egy színtelen élt megszínezünk pirosra vagy kékre a megfelelő szabályok szerint, akkor az új színezés is takaros lesz. Mivel az eddigi színezésünk takaros, van olyan F minimális költségű feszítő fa, mely az összes kék élt tartalmazza, és nem tartalmaz egyetlen piros élt sem.

Vegyünk egy f élt, amit a kék szabálynak megfelelően kékre festünk. Ha ez az f él része F-nek, akkor nyilvánvalóan f-et kékre színezve takaros színezést kapunk. Ha f nem éle F-nek, akkor biztosan van olyan p út F-ben ami f két végpontját összeköti, mivel F feszítő fa. Mivel f-re alkalmazható a kék szabály, van olyan X csúcshalmaz, melyből f kilép, de nem vezet ki belőle rajta kívül kék él. Vegyük a p útnak az X csúcshalmazból egy kilépő f' élét. Ez az él nem lehet kék, mivel a kék szabály szerint az X-ből

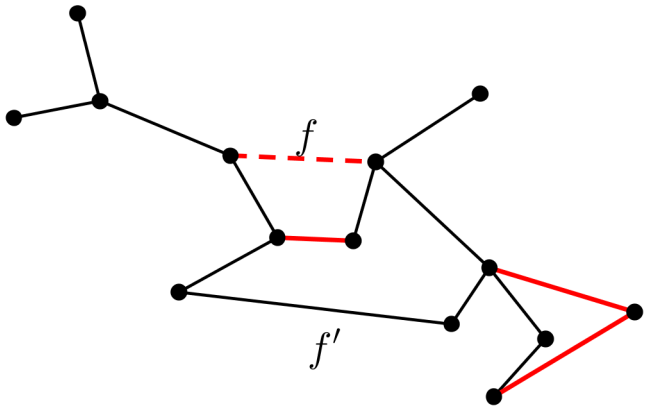
nem vezet ki kék él, és f súlya nem lehet nagyobb f' súlyánál, hiszen egy legkisebb súlyú élt kellett, hogy válasszunk. Az $F \cup \{f\}$ gráfban f' benne van egy körben (hiszen f' rajta van az f végpontjait összekötő úton), tehát $F' = (F \cup \{f\}) \setminus \{f'\}$ egy összefüggő, n csúcsú, $n-1$ élű gráf, tehát fa, súlya $w(F') = w(F) + w(f) - w(f') \leq w(F)$. Mivel F minimális súlyú, $w(F') = w(F)$ következik, tehát F' is minimális költségű feszítő fa. Ekkor F' tartalmaz minden kék élt (hiszen f' nem volt kék), és továbbra sem tartalmaz pirosat, tehát az új színezés valóban takaros.



1.3. ábra. Kék szabály

Most tegyük fel, hogy f-et pirosra színezzük a piros szabály szerint. Ha $f \notin F$, akkor ismét nyilvánvalóan takaros színezést kapunk. Vegyük azt az esetet, amikor $f \in F$, és vegyük azt a C kört, amely miatt alkalmazhattuk a piros szabályt, ebben f az egyetlen piros és egyben legnehezebb él. Az f él törlésével F két komponensre esik szét, viszont a $C \setminus \{f\}$ egy út f két végpontja közt, tehát van olyan f' él C-ben, mely a két komponens közt halad. Ez az f' él a piros szabály miatt nem lehet piros (olyan kört vettünk, amelyben nincs piros él) és súlya nem nagyobb f súlyánál, mivel a C élei közül egy legnagyobb súlyú élt kellett színeznünk. Tehát $F' = (F \cup \{f'\}) \setminus \{f\}$ egy fa, melyben minden kék él benne van és nem tartalmaz egyetlen piros élt sem. Súlya $w(F') = w(F) - w(f) + w(f') \leq w(F)$, tehát mivel F minimális költségű, így $w(F') = w(F)$, tehát F' egy, a színezés takarosságát igazoló feszítőfa.

Most belátjuk, hogy a színezéssel sosem akadunk el. Vegyünk egy színezetlen f élt. Mivel a színezésünk takaros, a kék élek erdőt alkotnak. Ha f mindkét végpontja ugyanabban a kék fában van, akkor a piros szabályt



1.4. ábra. Piros szabály

alkalmazhatjuk arra a körre, ami f-ből és a két végpontját összekötő kék útból áll. Ha f két végpontja különböző kék fákban van, akkor legyen az X halmaz az f él egyik végpontját tartalmazó fa csúcsainak halmaza. Ekkor X-ből nem vezet ki kék él, és a kivezető élek közül a legkisebb súlyút kékre színezhetjük.

Mivel beláttuk, hogy az eljárás működése során mindig takaros színezésünk van, azaz mindig lesz olyan minimális költségű feszítő fa, amely az összes kék élt tartalmazza, de egyetlen pirosat sem, valamint tudjuk, hogy az eljárás végére minden él színes lesz, ezért végül a minimális költségű feszítő fa minden éle kék lesz. Tehát az eljárás végén a kék élek valóban egy minimális költségű feszítő fát alkotnak.

□

Az eljárás alkalmazása során eddig tetszőleges sorrendben színeztük az éleket pirosra, illetve kékre, azonban hatékonyság szempontjából nem mindegy, mikor melyik szabályt használjuk. A következő algoritmusok ezen eljárás speciális esetei.

1.2. Kruskal algoritmus

A király feladatára a Kruskal nevű alattvaló által javasolt megoldás útépítési módszerét nevezzük Kruskal algoritmusnak. Az eljárás során a következő lépések alkalmazásával kapunk minimális költségű feszítő fát.

- Először megkeressük a gráfban az egyik legkisebb súlyú élt és a két végpontjával együtt belevesszük a részgráfunkba.
- Egy következő legkisebb súlyú éllel is hasonlóan járunk el.
- Ha a következő egyik legkisebb súlyú él a részgráfunkban kört alkotna, akkor ezt az élt nem vesszük bele a részgráfba, és tovább folytatjuk az eddigi lépéseket egészen addig, míg van még meg nem vizsgált él.

Vegyük észre, hogy ez az eljárás megegyezik a már ismertetett optimista stratégiával.

1.2.1. Állítás. *Ezzel az eljárással minimális költségű feszítő fát kapunk.*

Bizonyítás. Az algoritmus tulajdonképpen a piros-kék eljárást alkalmazza az alábbiak szerint. A Kruskal algoritmus által a részgráfba bevett éleket színezzük kékre, valamint azokat az éleket, amiket nem válogatott be, pirosra. Ez a színezés szabályos, ugyanis a kék élek erdőt alkotnak, mert az algoritmus a részgráfba csak olyan éleket vesz be, amik nem alkotnak kört egymással, tehát a részgráfban csak kék élek vannak. Egy f él kékre való színezésekor az erdő két komponensét kötjük össze ezzel az f éllel. A kék szabály értelmében f él kékre színezhető, mivel ha a két komponens közül az egyiket választjuk az X halmaznak, akkor belőle nem indulhat ki kék él, mert a komponensek közt nem futnak kék élek. Egy f élt akkor színezünk pirosra, ha végpontjai az erdő ugyanabban a komponensében vannak. Ezt megtehetjük, mivel a piros szabály értelmében vesszük azt a kört, ami az f él végpontjait összekötő útból és magából f-ből áll. Ekkor ebben a körben nincs piros él, mivel az út összes éle kék, f pedig még színtelen. Tehát az eljárás lépéseit alkalmazva, végül valóban egy minimális költségű feszítőfát kapunk eredményül. □

1.3. Prim algoritmus

A király feladatára a Prim nevű alattvaló által javasolt megoldás útépítési módszerét nevezzük Prim algoritmusnak. Az eljárás egy feszítő fát épít úgy, hogy minden lépésben egy csúccsal bővíti a már meglévő fát.

- Induljunk ki egy tetszőleges csúcsból, és az ebből kiinduló élek közül vegyük a legkisebb költségűt, így egy kétcsúcsú fához jutunk.

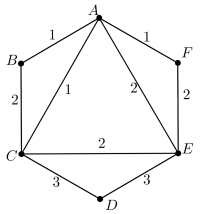
- A már meglévő fa és a gráf többi csúcsa között lévő élekből válasszunk ki egy legkisebb súlyút, és a nem fabeli csúcsával együtt vegyük hozzá a fához.
- Addig folytatjuk ezt az eljárást, míg van nem fabeli csúcs.

1.3.1. Állítás. *Az algoritmus során kapott feszítő fa minimális költségű.*

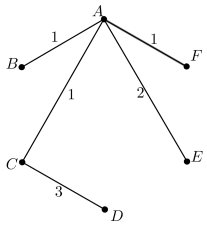
Bizonyítás. Prim algoritmusa valójában a kék szabályt alkalmazza úgy, hogy a kék fát bővíti. Legyen X a kék élek által lefedett csúcshalmaz. Ekkor megkeresi a legkisebb súlyú olyan élt, aminek egyik végpontja X-ben van, a másik pedig X-en kívül, ezt beszínezi kékre, és ezt az élt, valamint az X-en kívül eső végpontját pedig hozzáveszi X-hez.

A Prim algoritmus is egy minimális költségű feszítőfát alkot a kék szabály alkalmazásával, aminek helyességét már beláttuk. □

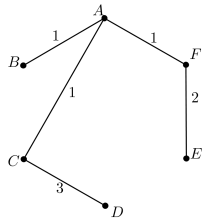
Fontos megjegyeznünk, hogy a Kruskal és Prim algoritmusok használatával nem minden esetben kapunk azonos minimális költségű feszítő fát, sőt algoritmusonként is kaphatunk különböző feszítő fákat többszöri futtatás esetében. Erre egy példával szolgálnak a következő ábrák:



1.5. ábra. Kiindulási gráf



1.6. ábra. Kruskal algoritmus által kapott feszítő fa



1.7. ábra. Prim algoritmus által kapott feszítő fa

2. fejezet

Legrövidebb utat kereső algoritmusok

Ez a fejezet főként a [8], valamint a [5] könyv alapján készült.

A királynak most már elég pénz gyűlt össze a kincstárában, így úgy dönt, hogy az összes utat megépítteti a városok között a könnyebb közlekedés érdekében. Azonban mikor az egyik tanácsadója meg szeretné látogatni az L városban élő rokonait, felmerül benne a kérdés, hogy melyik utat válassza.

Nézzük meg először, hogy mit is jelent a legrövidebb út élsúlyozott gráfok esetében.

2.0.1. Definíció. A *legrövidebb út* egy olyan út az gráf egy u és v csúcsa között, amelynek az élein lévő súlyok összege minimális.

Vegyük észre, hogy élsúlyozatlan esetben a legrövidebb út megegyezik a gráf u és v csúcsa közti legkevesebb élt tartalmazó útjával, mivel ebben az esetben az élek súlyait egységnyinek vehetjük.

2.0.2. Definíció. Az u csúcs v-től való *távolsága* az u, és a v csúcs között lévő utak közül egy legrövidebb út hossza, jelöljük ezt $d(u, v)$-vel. Amennyiben nincs út u és v közt, úgy $d(u, v) = \infty$.

2.0.3. Definíció. Vegyünk egy $G = (V, E)$ gráfban egy $s \in V$ csúcsot. Jelöljük egy $v \in V$ csúcsnak a becsült legrövidebb út hosszát s-től $\delta(v)$-vel. Egy $u \in V$ csúcs szomszédai közül v-t vizsgálva kiszámítjuk a hozzá vezető becsült legrövidebb út hosszát a következőképpen: $\delta(u) + w(u, v)$, amennyiben ez az összeg kisebb, mint a $\delta(v)$ értéke, akkor javítjuk. Ezt az eljárást nevezzük *közelítésnek*.

2.1. Szélességi bejárás

Ez a fejezet főként a [2], valamint a [5] könyvön alapul.

A király tanácsadója arra jött rá, hogy mivel bármelyik két szomszédos város közti út megtételéhez szükséges idő megegyezik, ezért célszerűbb azt az útvonalat választania, amelyik a legkevesebb városon megy keresztül. Ennek megtalálásában segít neki a következő algoritmus.

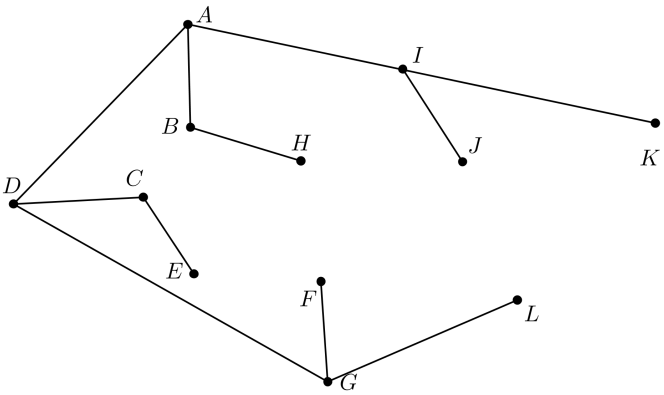
Jelöljük a csúcsok felfedezettségi állapotait fehér, szürke, fekete színnel. A fehér szín a még nem felfedezett csúcsokat jelenti, szürkével jelöljük az éppen megtalált csúcsokat, feketével pedig azokat a csúcsokat, amelyek összes közvetlen szomszédját már átvizsgáltuk, azaz minden közvetlen szomszédja fekete, vagy szürke. Az algoritmus során minden v szürke csúcsra egy (u, v) éllel való közelítést végzünk.

- Kezdetben minden csúcs fehér. Vegyünk egy tetszőleges kiindulási s csúcsot és színezzük szürkére, és legyen $\delta(s) = d(s, s) = 0$, valamint az összes többi $v \in V$ csúcsra $\delta(v) = \infty$.
- Vegyük a legrégebben megtalált v szürke csúcsot és keressük meg az összes még felfedezetlen közvetlen szomszédját, és azokat szürkére színezve végezzünk egy-egy közelítést. Ekkor v feketére színezhető.
- Ezt ismételjük addig, míg van szürke csúcs a gráfban.

Amennyiben az eljárás végén még van fehér csúcs, az azt jelenti, hogy a gráf nem összefüggő és ekkor egy újabb fehér csúcsra is lefuttatjuk az algoritmust.

A király tanácsadója az algoritmus segítségével először meghatározza a fővárosból az összes többi városba a legrövidebb utakat, majd így keresi meg az egyik legrövidebb utat L városba. Először kiindul a fővárosból és megkeresi az 1 egységnyi távolságra lévő szomszédait, ezek D, B, I városok. Majd sorra veszi ezek szomszédait kezdve D város azon közvetlen szomszédaival, amiket még nem talált meg, ezek C, G városok. Ezek után ugyanígy tesz B várossal is. Ennek egyetlen olyan szomszédja van, amit még nem talált meg, ez pedig H. I város még meg nem talált szomszédai J, K városok. Veszi a legrégebben megtalált olyan várost, aminek még nem kereste meg összes közvetlen szomszédját, ez C. Ennek egyetlen még nem vizsgált szomszédja van E. G városnak két olyan szomszédja van, amit még nem talált meg, az F és az L. A többi várost E, H, J, K, F, L sorban megvizsgálva, azt látja, hogy

már egyiknek sincs meg nem talált szomszédja. Végül arra a következtetésre jut, hogy számára egy megfelelő út lesz, ha A-ból D-n és G-n keresztül jut el L-be.



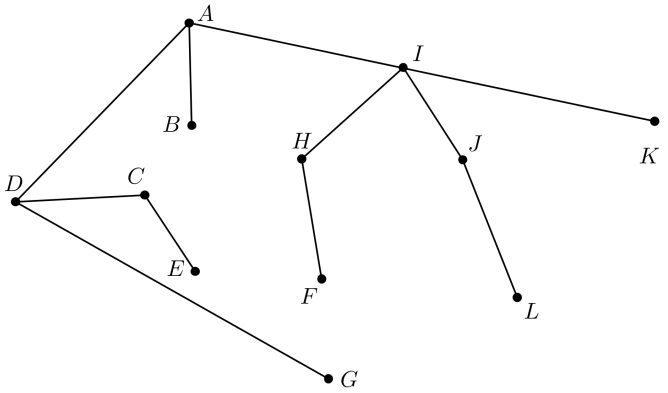
2.1. ábra. Szélességi bejárás eredménye az első esetben

Az algoritmus alkalmazása során, amikor egy v szürke csúcsnak keressük a szomszédait, nem kötöttük ki, hogy ezeket milyen sorrendben vegyük figyelembe, azaz milyen sorrendben színezzük szürkére őket. Ha más sorrendet választunk akkor más útvonalat kapunk. Lássuk milyen megoldásra jut a tanácsadó, ha más sorrendet választ.

Először ugyanúgy a fővárosból indul ki, de most szomszédai közül előbb I város azon közvetlen szomszédait keresi, amiket még nem talált meg, ezek H, J, K városok. Ezek után keresi D város szomszédait C-t és G-t, majd B városnak már nem talál olyan szomszédját, amit ne talált volna már meg korábban. A legrégebben megtalált város H, ennek még meg nem talált szomszédja F.J-nek új szomszédja L, majd K következik, de nem talál olyan vele szomszédos várost, amit még ne vizsgált volna meg. A következő C város még megtalálatlan szomszédja E. Végül a sorra következő G, F, L, E városoknak már minden szomszédjukat megtaláltuk. Így a tanácsadó talált egy másik megfelelő útvonalat A-ból I, J-n keresztül L-be, ami ugyancsak két városon át vezet.

Vegyük észre, hogy a két kiválasztott úthálózat nem ugyanaz, de a távolságok ugyanazok.

Mielőtt belátnánk, hogy az algoritmus tényleg működik, nézzük meg a legrövidebb utak néhány tulajonságát.



2.2. ábra. Szélességi bejárás eredménye a második esetben

Jegyezzük meg, hogy nemnegatív élsúlyozás esetén, két csúcs közt a legrövidebb séta egy út. A legrövidebb sétában nem érinthetünk egy csúcsot többször, hiszen egy csúcs két érintése közötti részt nyugodtan elhagyhatjuk, és ekkor továbbra is sétát kapunk a két csúcs között.

2.1.1. Állítás. (Háromszög-egyenlőtlenség) *Egy $G = V, E$ nemnegatív élsúlyozással ellátott gráfban bármely $s, u, v \in V$ csúcsokra igaz, hogy $d(s, v) \leq d(s, u) + d(u, v)$.*

Bizonyítás.

Egy s-ből v-be tartó legrövidebb útnak nem lehet nagyobb a súlya, mint bármely másik s és v közti útnak. Azaz s és v közti legrövidebb út súlya nem nagyobb, mint annak az útnak, amely az s és u közti legrövidebb útból, valamint az u és v közti legrövidebb útból áll. Ha nincs út s és u közt, akkor a definíció szerint $d(s, u) = \infty$, hasonlóan, ha u és v közt nincs út, akkor $d(u, v) = \infty$, valamint ekkor s és v közt sincs út, tehát $\infty = d(s, v) \leq d(s, u) + d(u, v) = \infty$ teljesül.

□

2.1.2. Állítás. (Felső korlát tulajdonság) *Minden $v \in V$ csúcsra fennáll $\delta(v) \geq d(s, v)$, és ha $\delta(v)$ egyszer eléri $d(s, v)$ értéket, attól fogva sosem változik meg.*

Bizonyítás.

Először a közelítő lépések száma szerinti indukcióval fogjuk belátni a $\delta(v) \geq d(s, v)$ egyenlőtlenséget. Kezdetben $\delta(v) \geq d(s, v)$ teljesül, mivel

$0 = \delta(s) \geq d(s,s)$, valamint $\delta(v) = \infty$ miatt $\delta(v) \geq d(s,v)$. A közelítés során csak $\delta(v)$ értéke változhat, de akkor is igaz marad rá az állítás, mivel

$$\begin{aligned}\delta(v) &= \delta(u) + w(u,v) \\ &\geq d(s,u) + w(u,v) \quad \text{indukciós feltevés miatt} \\ &\geq d(s,v) \quad \text{háromszög-egyenlőtlenség miatt}\end{aligned}$$

Most lássuk be, hogy ha $\delta(v)$ egyszer eléri $d(s,v)$ értéket, attól fogva sosem változik meg. $\delta(v)$ az alsó korlátot elérve tovább már nem csökkenhet, hiszen $\delta(v) \geq d(s,v)$, de nem is nőhet, mert a közelítés nem növeli δ értékét. □

2.1.1. Tétel. (Szélességi bejárás helyessége) *Legyen $G = (V,E)$ egy irányítatlan, élsúlyozatlan gráf, és tegyük fel, hogy egy $s \in V$ kezdőpontból kiindulva alkalmazzuk rá az algoritmust. Ekkor az algoritmus minden s-ből elérhető $v \in V$ csúcsot elér, valamint ezekre befejezéskor $\delta(v) = d(s,v)$ teljesül.*

Bizonyítás.

Először azt fogjuk belátni, hogy minden lépésben a szürke csúcsok becsült távolságának különbsége legfeljebb 1, és ha egy $u \in V$ csúcs korábban lett szürke, mint egy v csúcs, akkor $\delta(u) \leq \delta(v)$. A bizonyítás során minden lépésben az aktuálisan szürke csúcsokra nézzük, hogy teljesül-e az állítás. Kezdetben csak az s csúcs szürke, így teljesül rá az állítás, mivel önmagától vett távolsága 0. Ha az s csúcsot feketére színezzük, az azt jelenti, hogy megtaláltuk az összes közvetlen szomszédját. Ezek becsült távolsága s-től 1, így becsült távolságuk különbsége 0. Tegyük fel, hogy néhány lépésig igaz az állítás. Vegyük a legrégebben megtalált v szürke csúcsot, legyen $\delta(v) = k$, és nézzük ennek a szomszédait. Ekkor csak a k és $k+1$ becsült távolságra lévő csúcsok lehetnek szürkék. Az első $k+2$ becsült távolságra lévő csúcsot csak úgy találhatjuk meg, ha már az összes $k+1$ becsült távolságú csúcsot megtaláltuk, azaz mind szürke és ezek közül a legrégebben megtaláltnak keressük a szomszédait. Tehát a szürke csúcsok távolságának különbsége legfeljebb 1, ezzel az állítást igazoltuk.

Most lássuk be, hogy az algoritmus befejezésekor $\delta(v) = d(s,v)$ teljesül. Indirekt tegyük fel, hogy $\delta(v) \neq d(s,v)$. Egy $s \to v$ úton vegyünk egy olyan v csúcsot, aminek a legkisebb a $\delta(v)$ értéke, és $\delta(v) \neq d(s,v)$ teljesül. Tudjuk, hogy $\delta(v) < d(s,v)$ a felső korlát tulajdonság miatt nem teljesülhet. Nézzük

egy legrövidebb $s \to v$ úton a v csúcsot közvetlenül megelőző u csúcsot. Tudjuk, hogy

$$
\begin{aligned}
d(s,v) &\leq d(s,u)+1 \quad \text{háromszög-egyenlőtlenség miatt} \\
d(s,v) &= d(s,u)+1 \quad d(s,u) < d(s,v)\text{miatt} \\
\delta(u) &= d(s,u) \quad \text{mivel } v \text{ volt az első, amire nem teljesül az egyenlőség} \\
d(s,v) &= \delta(u)+1 \\
\delta(v) &> \delta(u)+1 \quad \delta(v) > d(s,v) \text{ feltevésünk miatt}
\end{aligned}
$$

Nézzük azt a lépést, amikor u-nak keressük a szomszédait.

- Ha ekkor v fehér, az azt jelenti, hogy még más csúcsból nem értük el, azaz becsült távolsága s-től: $\delta(v) = \delta(u)+1$. Ez pedig ellentmond annak az állításunknak, hogy $\delta(v) > \delta(u)+1$.
- Ha a v csúcs színe szürke, akkor már találtunk egy olyan w csúcsot, amit már u előtt megtaláltunk. Tehát $\delta(v) = \delta(w)+1$, de mivel $\delta(w) \leq \delta(u)$, ezért $\delta(v) \leq \delta(u)+1$, amivel szintén ellentmondásra jutottunk.
- Ha v csúcs ekkor már fekete volt, akkor u előtt már megtaláltuk, azaz $\delta(v) \leq \delta(u)$, ami pedig ismételten ellentmondás.

Tehát beláttuk, hogy $\delta(v) > d(s,v)$ nem teljesülhet, vagyis $\delta(v) = d(s,v)$ teljesül.

□

2.1.1. Dijkstra algoritmus

Ez a fejezet főként a [6], valamint a [5] könyvön alapul.

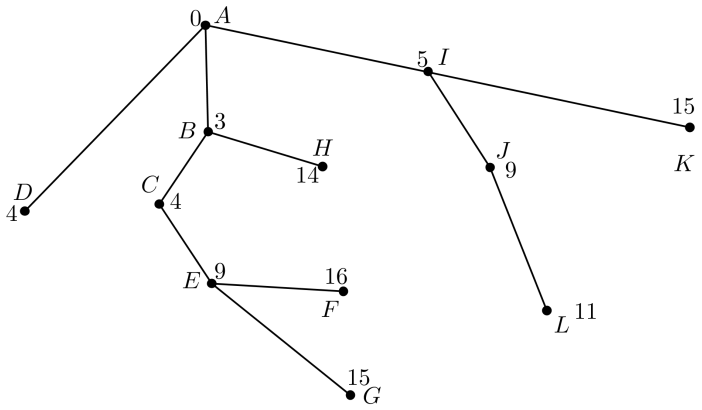
A király úgy dönt, hogy mivel nagyon sok pénzt költött az utak megépítésére, ezért a különböző városok közt az építési költséggel arányos úthasználati díjat vezet be. Tanácsadója amint meghalotta a király ötletét, azon kezd el gondolkodni, hogy vajon az előző módszerrel talált legrövidebb út egyben a legolcsóbb-e számára, ha rokonait szeretné meglátogatni. Ehhez először meg kell találnia a legolcsóbb utat a fővárosból L városba. Ebben a Dijkstra algoritmus nyújt neki segítséget.

Az algoritmus nemnegatív élsúlyozás esetén a legrövidebb utat adja meg az s kezdőcsúcsból az összes többi csúcsba. Tulajdonképpen most is egy szélességi keresést fogunk végrehajtani, az élek súlyának figyelembevételével. Alkalmazzuk az előző algoritmus során bevezetett színezést és vegyünk egy S halmazt, ami a fekete csúcsokat tartalmazza.

- Kezdetben az S halmaz üres. Vegyük a kiindulási s csúcsot és színezzük szürkére, ekkor $\delta(s) = d(s, s) = 0$, valamint az összes többi $v \in V$ csúcsra $\delta(v) = \infty$.
- Vegyük a legkisebb $\delta(v)$ értékű szürke csúcsot és keressük meg összes közvetlen szomszédját, valamint azokat szürkére színezve végezzünk rajtuk egy-egy közelítést v-ből. Ezek után v-t bevesszük S-be.
- Ezt ismételjük addig, míg az összes csúcs be nem kerül S-be.

Alattvalónk az algoritmus használatával a következő megoldásra jut. Először megnézi A (a főváros) szomszédait, ezek a D, B, I városok, majd kiszámítja az útiköltséget a fővárosból ezekbe a városokba és feljegyzi. Majd a költségek növekvő sorrendjében vizsgálja tovább a városokat. B szomszédai közül (C és H) azt veszi bele ebbe az S halmazba, amelyhez a legolcsóbb az út A és C, vagy A és H között. Az útiköltség C-be összesen 4 arany, H-ba 14 arany, azonban D városba is 4 arany az útiköltség, tehát választhat, hogy C-t, vagy D-t veszi be előbb S-be. A D város mellett döntve vizsgálja szomszédait, C-be egyelőre nem talál olcsóbb utat, viszont G-be lesz út D-n keresztül, így oda a becslés a legolcsóbb útra 17 arany, majd D-t is beveszi a halmazba. A következő legolcsóbb út, C-be vezet. Szomszédai közül egyedül E-t nem vette még be az S halmazba, oda az út 9 aranyba kerül, és mivel nincs más elérhető szomszéd, így C is belekerül S-be. A következő legolcsóbb út I-be vezet 5 aranyért, ennek szomszédai J, ahová 9 aranyat kell fizetni, valamint K, ahová pedig 15 arany a jelenleg talált legolcsóbb út, H városába I-n keresztül nem talál olcsóbb utat, több szomszédja pedig nincs,így I is belekerül S-be. A következő olyan városból aminek a szomszédait tekintheti ugyancsak kettő van E és J. E-t választva G-be talál egy olcsóbb utat, így E-n keresztül 15 aranyért tud G városába menni, valamint még egy még nem vizsgált szomszédja van ez F, ahová 16 arany az útdíj. Ezek után E-t is beveszi a halmazba. J még elérhető szomszédai közül csak L-be van olcsóbb út, ami 11 aranyba kerül, majd miután az összes szomszédos várost megnézte J-t is beveszi a halmazba. Ezek után az eddigiekhez hasonlóan sorra veszi

a városokat, de már sehová sem talál olcsóbb utat. A következő sorrendben kerülnek bele S-be a még megmaradt városok: L, H, G, K, F. Tehát a legolcsóbb út számára, ha az A fővárosból a I, J városokon keresztül megy L városba.



2.3. ábra. Dijkstra algoritmus eredménye. A csúcsokon szereplő számok az útiköltséget jelzik A csúcsból kiindulva.

2.1.2. Tétel. (Dijkstra algoritmus helyessége) *Ha egy $G = (V, E)$ nemnegatív élsúlyozással ellátott, irányított, vagy irányítatlan gráfban alkalmazzuk az algoritmust egy s kezdőpontból kiindulva, akkor annak befejezésekor minden $v \in V$-re $\delta(v) = d(s, v)$, azaz az algoritmus meghatározza a legrövidebb út hosszát s és a gráf egy v pontja között.*

Bizonyítás.

Meg fogjuk mutatni, hogy ha az S halmazba bevesszük a v csúcsot, akkor teljesülni fog a $\delta(v) = d(s, v)$ egyenlőség, valamint azt már a felső korlát tulajdonság miatt tudjuk, hogy az egyenlőség később is megmarad.

Indirekt tegyük fel, hogy a v az első olyan csúcs, amelyet ha S-be beleveszünk $\delta(v) \neq d(s, v)$ teljesül. A felső korlát tulajdonság miatt tudjuk, hogy $\delta(v) > d(s, v)$. Vizsgáljuk meg az s-ből v-be menő legrövidebb utat. Tudjuk, hogy $s \neq v$, hiszen az s csúcsot vettük be S-be először és $\delta(s) = d(s, s) = 0$ telejesül, ami ellentmond a feltevésünknek. Továbbá léteznie kell legalább egy útnak s és v között, különben a felső korlát miatt $\delta(v) \geq d(s, v) = \infty$, azaz $\delta(v) = d(s, v) = \infty$ lenne, ami szintén ellentmondás. Márpedig ekkor létezik egy legolcsóbb p út is s és v közt. Legyen ezen a p úton y az első olyan csúcs,

ami nincs benne S-ben, és legyen u a p úton az y előtti csúcs. Ekkor $u \in S$, és emiatt y szürke, hiszen mikor az u csúcsot bevettük S-be már megtaláltuk.

Mivel u bekerült S-be, ezért az y csúcsra végeztünk közelítést az (u, y) él mentén, ezért fennáll a $\delta(y) \leq \delta(u) + w(u, y)$ egyenlőtlenség, ugyanis, ha a közelítés előtt $\delta(y) > \delta(u) + w(u, y)$ fennáll, akkor a közelítés után $\delta(y) = \delta(u) + w(u, y)$ teljesül. Ha a közelítés előtt $\delta(y) \leq \delta(u) + w(u, y)$ fennállt, akkor ez utána sem változik, mivel sem $\delta(u)$, sem pedig $\delta(y)$ nem változik.

Továbbá mivel feltettük, hogy v volt az első S-be kerülő csúcs, amire nem teljesül az egyenlőség, ezért $\delta(u) = d(s, u)$, és így $\delta(y) \leq d(s, u) + w(u, y)$. Bontsuk fel p legolcsóbb utat $s \to y \to v$ részutakra. Ekkor p összköltsége annyi, mint az $s \to y$ részút költsége, és $y \to v$ út költsége együtt. Tegyük fel, hogy van az $s \to y$ részútnál egy olcsóbb út s-ből y-ba. Ezek szerint, ekkor az olcsóbb $s \to y$ részút költsége, és az $y \to v$ útnak a költsége együtt olcsóbb lesz, mint az eredeti legolcsóbb p út, ami pedig ellentmondás. Ebből következik, hogy az $s \to y$ részút egy legolcsóbb út, tehát $\delta(y) = d(s, y)$. Mivel az y csúcs v csúcs előtt van a p legrövidebb úton, valamint az élsúlyok nemnegatívak, ezért $d(s, v) \geq d(s, y)$. Tehát tudjuk, hogy

$$\begin{aligned} \delta(y) &\leq d(s, v) \\ &\leq \delta(v) \quad \text{felső korlát tulajdonság miatt} \end{aligned}$$

Ezzel ellentmondásra jutottunk, hiszen az algoritmus során a legkisebb δ értékű szürke csúcsot tesszük S-be, ami ezek szerint nem lehet v, hiszen $\delta(y) < \delta(v)$. $\square$

2.1.2. Bellman-Ford algoritmus

Ez a fejezet főként a [7], valamint a [5] könyvön alapul.

Most, hogy a tanácsadó már tudja, hogy mennyibe kerül neki az út a rokonaihoz, elkezd spórolni az útra. Mivel az útdíjakat drágának tartja, úgy dönt, hogy felvesz utasokat út közben, akik szintén hasonló irányba tartanak, és kitalálja, hogy mennyi pénzt kérjen utasonként. Felméri, hogy bizonyos városok között hány utasra számíthat, és ezt összeveti az útdíjakkal, valamint feltünteti a térképén. Így előfordulhat, hogy bizonyos szakaszokon negatív érték kerül feltüntetésre, mivel azokon az utakon az utasok által fizetett összeg több, mint az útdíj.

Az előző algoritmus, ha megengedünk negatív értékeket nem szolgáltat jó megoldást, ezért egy másik eljárást kell alkalmaznia, ez a Bellman-Ford

algoritmus. Az eljárás lényegében ugyanazt a stratégiát alkalmazza, mint a Dijkstra algoritmus, ha egy csúcshoz az addiginál kisebb δ érték tartozik, akkor javítja ezt az értéket. A különbség a negatív költségek figyelembe vételekor jelentkezik, ugyanis ekkor nem teljesül a háromszög-egyenlőtlenség, és így előfordulhat, hogy javítani tudunk egy már fekete csúcsra a kezdőpontból odavezető eddigi út költségén. Fontos megjegyeznünk, hogy a gráfban nem lehetnek negatív összsúlyú körök, hiszen a körön többször áthaladva a séták költségét mindig csökkenteni tudnánk. Irányítatlan gráfok esetében továbbra sem lehetnek negatív élek, ugyanis ekkor az élen oda-vissza haladva a költség tetszőlegesen csökkenthető.

Az algoritmus olyan irányított gráfokra alkalmazható, ahol negatív éleket is megengedünk, de negatív összköltségű kört nem. Az eljárás során éleken, utakon az irányított éleket, utakat értjük.

Az algoritmus alkalmazása során az éleket tetszőleges sorrendben vizsgálhatjuk és ha lehetséges javítunk az eddigi legrövidebb út értékén. Mivel a közelítések sorrendjét nem kötöttük ki, ezért csak az garantálható, hogy az első lépés végén az 1 élből álló legrövidebb utakat találjuk meg, majd a második lépés végén a 2 élből állóakat, és így tovább. Egy n csúcsú gráfban legfeljebb $n-1$ élből álló út lehet, ezért összesen legfeljebb $n-1$ lépést végzünk.

- Kezdetben vegyük a kiindulási s csúcsot. Legyen $\delta(s) = d(s,s) = 0$, valamint az összes többi $v \in V$ csúcsra $\delta(v) = \infty$.
- Vizsgáljuk meg az összes (u,v) élt tetszőleges sorrendben, és ha $\delta(v) > \delta(u) + w(u,v)$ akkor $\delta(v)$ értékét $\delta(u) + w(u,v)$ értékre javítjuk.
- Ismételjük meg az előző lépést még $n-2$-szer (összesen $n-1$-szer).

2.1.3. Tétel. (Bellman-Ford algoritmus helyessége) *Legyen G=(V,E) egy élsúlyozott, irányított, negatív összköltségű kört nem tartalmazó véges gráf, továbbá a kezdőcsúcs legyen s. Ekkor a Bellman-Ford algoritmus meghatározza minden $v \in V$ csúcsra legrövidebb út hosszát s-ből, azaz $\delta(v) = d(s,v)$ értéket.*

Bizonyítás. Először a lépések száma szerinti indukcióval belátjuk, hogy ha a $\delta(v) \neq \infty$, akkor van út s és v között, aminek távolsága $\delta(v)$. Kezdetben igaz az állítás, hiszen $\delta(s) = 0$, az összes többi v csúcsra pedig $\delta(v) = \infty$ (nincs él nélküli $s \to v$ út). Nézzük azt a lépést, amikor az eljárás során a

$\delta(v)$ értéket javítjuk az (u, v) irányított él mentén. Vegyük észre, hogy ekkor $\delta(v) < \infty$. Mivel az indukciós feltevés szerint van $\delta(u)$ hosszú út s és u csúcsok között, vegyük az $s \to u$ utat és egészítsük ki az (u, v) éllel, ekkor kapunk egy $\delta(v) = \delta(u) + w(u, v)$ értéket, és az $s \to v$, $\delta(u)$ hosszú út és az (u, v) él hossza együtt épp ennyi.

A következőkben azt fogjuk belátni, hogy i lépés után (egy lépés alatt azt értjük, hogy a gráf összes éle mentén javítunk) minden v csúcsra $\delta(v)$ éppen a legfeljebb i élből álló $s \to v$ utak közül a legkisebb súlyúnak a súlya. Vegyük a legfeljebb i élből álló $s \to v$ utak közül egy legkisebb súlyút, hívjuk p-nek. Legyen ezen az úton a u a v csúcsot közvetlen megelőző csúcs. Ekkor az u csúcsig tartó p' részút a legrövidebb legfeljebb $i-1$ élt tartalmazó $s \to u$ út, és p súlya éppen $w(p') + w(u, v)$. Az indukciós feltevés szerint $\delta(u) \leq w(p')$. Az i. lépés során javítunk az (u, v) él mentén, azaz $\delta(v)$ értéket összehasonlítjuk a $\delta(v) = \delta(u) + w(u, v)$ összeggel és, ha ez az összeg kisebb, javítjuk a $\delta(v)$ értéket. Tehát az i. lépés után $\delta(v) \leq \delta(u) + w(u, v) \leq w(p') + w(u, v) = w(p)$, de kisebb a felső korlát tulajdonság miatt nem lehet, tehát egyenlőség áll fenn.

Az $n-1$ lépés végrehajtása után minden csúcshoz megtaláltuk az odavezető, legfeljebb $n-1$ élből álló utak közül a legkisebb súlyút, és mivel nincs negatív összsúlyú kör a gráfban, minden csúcsba az odavezető legkisebb súlyú út legfeljebb $n-1$ élből áll, ezért a kapott érték valóban a távolság lesz, azaz $\delta(v) = d(s, v)$ teljesül minden v csúcsra.

□

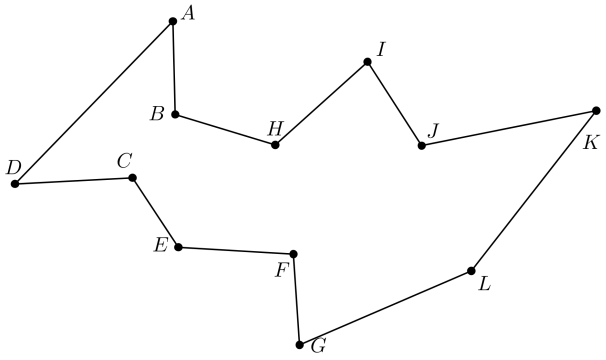
3. fejezet

Hamilton körök

Amikor a király az összes utat megépíttette, a kereskedő is gondolkodóba esett, hogy melyik utakon és milyen sorrendben látogassa meg a városokat. Neki mindegyik városba el kell szállítania az árut és nap végén az otthonába szeretne visszatérni, ahonnan elindult. Ehhez kell megtalálnia a legrövidebb körutat (ha egyáltalán van ilyen), hogy mielőbb hazaérhessen.

A feladat valójában Hamilton kör (azaz olyan kör, ami a gráf minden csúcsán pontosan egyszer megy át) keresése a gráfban.

A kereskedő problémájának egy lehetséges megoldását mutatja a következő ábra.



3.1. ábra. A kereskedő egy lehetséges körútja

3.1. Utazó ügynök probléma

Ez a fejezet főként a [3], valamint a [1] könyvön alapul.

Az úthasználati díj bevezetésével újabb probléma merült fel a kereskedőben. Most már nem csak az a célja, hogy mielőbb hazaérjen, hanem, hogy minél nagyobb nyereségre tegyen szert, így a legolcsóbb útvonalat kell megtalálnia.

Az utazóügynök feladata egy $G = (V, E)$ n csúcsú teljes, nemnegatív élköltségekkel ellátott gráfban legrövidebb olyan kör keresése, amely pontosan egyszer áthalad a gráf összes csúcsán, azaz egy legrövidebb Hamilton kör keresése. A továbbiakban az egyszerűség kedvéért csak irányítatlan gráfokkal foglalkozunk, valamint feltesszük, hogy az élköltségekre teljesül a háromszög-egyenlőtlenség: $w(u, v) + w(v, z) \geq w(u, z)$ minden u, v, z csúcsokra.

Ezek a feltételek sajnos nem teljesülnek a király úthálózatára, ennek következtében az alábbiakban bemutatásra kerülő eljárások közül nem mindegyik működik, ezért az eljárások szemléltetéseként új példagráfokat mutatok. Itt megjegyezném, hogy amennyiben nem tesszük fel, hogy teljesül a háromszög-egyenlőtlenség, akkor nem szükséges kikötnünk, hogy a gráf legyen teljes, hiszen a hiányzó éleket kellően nagy költséggel hozzávéve a gráfhoz már teljes gráfot kapunk, melyben a legolcsóbb körút egyben az eredeti gráf legolcsóbb körútja is lesz (amennyiben van benne). Például ezáltal használható az utazóügynök feladat Hamilton-kör keresésére élsúlyozatlan gráfokon úgy, hogy a már meglévő élek 1 súlyt kapnak, a hiányzó élek pedig 2-t. Ha az utazó ügynök feladatban az optimális körút hossza a csúcsszámmal egyenlő, akkor van Hamilton kör az eredeti gráfban, ha több, mint a csúcsszám, akkor nem lesz benne Hamilton kör. Ráadásul, ha az előbb említett élsúlyozást alkalmazzuk a gráfban, akkor még a háromszög-egyenlőtlenség is teljesülni fog. Mivel a Hamilton kör keresés közismerten nehéz feladat, ebből látszik, hogy az utazó ügynök feladata is az.

3.1.1. Levágó algoritmus

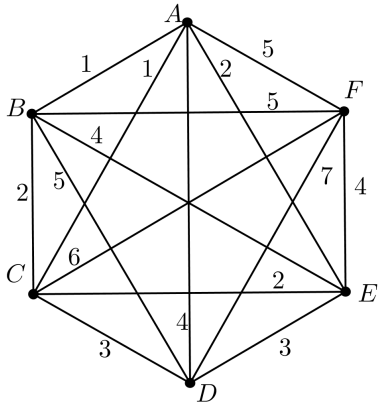
Az utazó ügynök problémára jelenleg még nem áll rendelkezésünkre olyan algoritmus, mely egy tetszőleges gráfban egyszerűen és gyorsan megadná az optimális körutat. A következőkben olyan algoritmusokat mutatunk be, amelyek ha nem is szolgálnak optimális eredménnyel, de ahhoz közeli értéket adnak. Ilyen például a most következő levágó algoritmus is, amely a következőképpen működik.

- Első lépésben keressünk egy minimális költségű F feszítőfát, például Kruskal algoritmussal.
- Éleit duplázzuk meg és nevezzük az így kapott gráfot F'-nek.
- Keressünk ebben a gráfban egy Euler sétát, ami biztosan létezik, hiszen a gráfunk összefüggő és benne a megduplázott élek miatt minden csúcs foka páros.
- Így persze lesz olyan csúcs, amin többször is áthaladtunk, de ezt a problémát könnyen orvosolhatjuk, ha ezeket az utakat levágjuk, azaz, ha a séta során i csúcsból olyan j csúcson kellene keresztül mennünk, amin korábban már áthaladtunk, akkor ezt a j csúcsot most kihagyjuk és i-ből közvetlenül k-ba, azaz a séta következő csúcsába megyünk. (az i-ből k-ba van út, mivel G teljes gráf)
- Az élek számát addig csökkentjük ezzel a módszerrel, míg már minden csúcsot pontosan egyszer érintünk a séta során, azaz egy H Hamilton kört kapunk.

3.1.1. Tétel. *A levágó algoritmussal kapott H Hamilton kör költsége nem haladja meg az optimális körút költségének kétszeresét.*

Bizonyítás. Az F' gráfon az Euler séta költsége kétszer annyi, mint az F minimális költségű feszítő fáé, mivel F'-et úgy kaptuk, hogy az F-beli éleket megdupláztuk. A háromszög-egyenlőtlenség miatt az élek levágásával kapott Hamilton kör költsége legfeljebb annyi, mint az F' gráfé, aminek költsége az F feszítőfa költségének a kétszeresével egyenlő. Azonban az F feszítőfa költsége kisebb, mint a legolcsóbb körúté, ugyanis, ha a legolcsóbb körút egyik élét elhagyjuk egy feszítőfát (pontosabban feszítő utat) kapunk. Nem biztos, hogy ez a feszítőfa minimális költségű, de költsége nem lehet kevesebb a minimális költségű feszítőfa költségénél. Tehát az algoritmus során előállított H Hamilton kör költsége legfeljebb kétszer akkora, mint az optimális körút költsége. □

A H Hamilton kör költsége azonban függ attól, hogy melyik minimális költségű feszítő fából indultunk ki, sőt attól is függhet, hogy melyik Euler sétát vesszük F'-ben. A 3.3. és a 3.4. ábrán lévő két minimális költségű feszítőfa összköltsége mindkét esetben 11, azonban az első esetben kapott Hamilton kör költsége 19, a második esetben pedig 17.



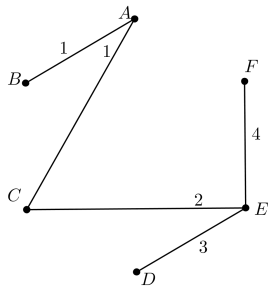
3.2. ábra. A kiindulási gráf

3.1.2. Christofides heurisztikája

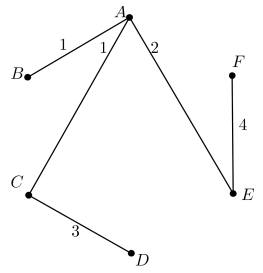
A most következő algoritmussal valamivel jobb értéket kapunk a körút költségére.

- Először keressünk a G gráfban egy minimális költségű F feszítőfát, például Kruskal algoritmussal.
- Vegyük F-ből a páratlan fokú csúcsokat, és legyen F' az F-ben a páratlan fokú csúcsok által feszített részgráfja G-nek.
- Keressünk G-ben az F' által feszített fa csúcsai közt egy minimális súlyú M teljes párosítást. (Ezt például Edmonds algoritmusával tehetjük meg.)
- Ehhez vegyük hozzá az eredeti F feszítőfát, így $F \cup M$ a mindkét gráfban lévő éleket duplán tartalmazza. Tehát $F \cup M$ minden foka páros, tehát van benne Euler séta.
- Mivel ismét vannak olyan csúcsok, melyeken többször áthaladtunk, ezért az előző alfejezetben ismertetett módszer alapján csökkentjük az élek számát, míg végül egy H Hamilton kört kapunk.

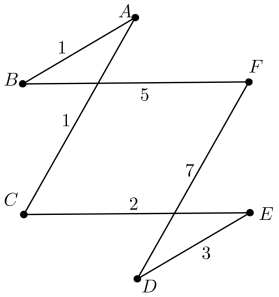
3.1.2. Tétel. *A Christofides heurisztikája által kapott H Hamilton kör költsége nem haladja meg az optimális körút költségének másfélszeresét.*



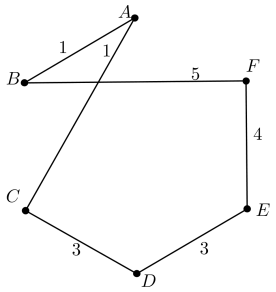
3.3. ábra. Egy minimális költségű feszítő fa a kiindulási gráfban



3.4. ábra. Egy másik minimális költségű feszítő fa a kiindulási gráfban



3.5. ábra. A levágó algoritmus által kapott Hamilton kör az első minimális költségű feszítő fából kiindulva



3.6. ábra. A levágó algoritmus által kapott Hamilton kör a második minimális költségű feszítő fából kiindulva

Bizonyítás. Legyen O egy optimális körút. Azt már láttuk, hogy O költsége nem lehet kevesebb, mint F költsége. Legyen $H_{F'}$ az az F'-beli Hamilton kör, amiben a csúcsok sorrendje megegyezik az optimális körút csúcsainak sorrendjével. Ekkor a háromszög-egyenlőtlenség miatt teljesül, hogy $w(H_{F'}) \leq w(O)$. Állítsuk elő $H_{F'}$-t két F'-beli teljes párosítás uniójaként. Ezt megtehetjük például úgy, hogy az egyik teljes párosítás (jelöljük M_1-gyel) álljon a Hamilton kör minden második éléből, a másik (jelöljük M_2-vel) pedig az összes többi élből.

$$\begin{aligned} 2w(M) &\leq w(M_1) + w(M_2) \\ &\leq w(H_{F'}) \\ &\leq w(O) \end{aligned}$$

Ekkor F és M költségének összege legfeljebb annyi, mint az optimális körút másfélszerese, valamint a háromszög-egyenlőtlenség miatt a H költsége is legfeljebb akkora, mint az $F \cup M$ költsége, ezzel az állítást bizonyítottuk. □

Megjegyzem, hogy míg az általam bemutatott felső korlátok csak olyan teljes gráfokra teljesülnek, amelyekre fennáll a háromszög-egyenlőtlenség, addig a most tárgyalásra kerülő alsó korlátok esetén ez a két feltétel nem szükséges.

3.1.3. 1-fa korlát

Az előző alfejezetekben már láttuk, hogy a minimális költségű feszítőfa súlya egy alsó korlát az optimális körút költségére. Most ezt az alsó korlátot fogjuk finomítani.

Vegyünk egy $v \in V$ csúcsot a $G = (V, E)$ gráfban. Számítsuk ki a v-ből kiinduló két legolcsóbb él költségének összegét, valamint határozzuk meg a $G \backslash v$ gráf minimális költségű feszítőfájának a súlyát, és vegyük a két eredmény összegét. Ezt az összeget szokás 1-fa korlátnak hívni, ami szintén egy alsó korlátot fog adni az optimális körút költségére.

3.1.1. Állítás. *Az 1-fa korlát egy alsó becslést ad az optimális körút költségére.*

Bizonyítás.

Legyen v egy tetszőleges csúcs, valamint jelöljük az optimális körutat O-val. Mivel O a gráf összes csúcsán áthalad pontosan egyszer, ezért minden csúcs foka, és így v foka is legalább 2. Ha ezt a v csúcsot kivesszük az O optimális körútból, a maradék út egy-egy feszítő fa $G \backslash v$-ben , aminek költsége legalább annyi, mint a $G \backslash v$ gráfban egy minimális költségű feszítőfa költsége. Ha a $G \setminus v$-ben lévő minimális költségű feszítő fához hozzáadjuk a v csúcs két legolcsóbb élének költségét, akkor ez az összeg nem lehet több, mint az $O \setminus v$ feszítő fa költségéhez a v-ből kiinduló két O-beli él költségét hozzáadva. Tehát az 1-fa korlát valóban egy alsó becslést ad.

□

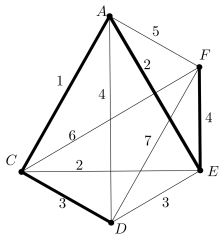
A következő ábrákon azt szeretném szemléltetni, hogy az 1-fa korlát függ v választásától.

A 3.7. ábrán $B = v$ választással az 1-fa korlát: $1+2+10 = 13$, a 3.8. ábrán $F = v$ választással $5 + 4 + 7 = 16$. Ha összehasonlítjuk a levágó algoritmus

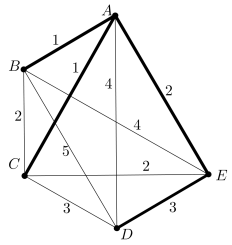
által kapott felső korlátot a most kapott alsó korláttal, akkor láthatjuk, hogy az optimális körút (O) költsége $16 \leq O \leq 17$.

Ezzel a mechanizmussal a baj az, hogy a $G \setminus v$-ben lévő minimális költségű feszítő fa alakja távol eshet az optimális körúttól, mivel igazából egy minimális költségű feszítő utat kellene keresnünk, amiben a kezdő és végpont kivételével minden csúcs foka pontosan 2, de a feszítő fában nagyfokú csúcsok is lehetnek.

Ezen a problémán próbál javítani a Held és Karp korlátja.



3.7. ábra. A 3.2 ábra alapján készült 1-fa korlát egy minimális költségű feszítőfája $B = v$ választással



3.8. ábra. A 3.2 ábra alapján készült 1-fa korlát egy minimális költségű feszítőfája $F = v$ választással

3.1.4. Held-Karp korlát

Válasszunk ki a $G \setminus v$-ben lévő feszítő fából egy nagyfokú u csúcsot és minden G-ben rá illeszkedő élének a súlyát növeljük meg egy tetszőleges M értékkel. Minden körút hossza pontosan $2M$-mel nő, hiszen u csúcsba egyszer belép és egyszer kilép, de az 1-fa korlát nőhet $2M$-nél többel, mert a feszítő fában u foka lehet több, mint 2. Tehát jobb alsó korláthoz jutunk. A továbbiakban ezt az eljárást alkalmazzuk egyszerre az összes csúcsra. Vezessük be a csúcsokra az $y : V \setminus v \to \mathbb{R}$ súlyfüggvényt, ekkor egy $(u, v) \in E$ élre $w(u, v)' = w(u, v) + y_u + y_v$. Jelöljük C'-vel az új élsúlyozással kapott 1-fa korlátot, ekkor a Held-Karp korlát a következő: $C' - 2 \sum_{u \in V \setminus v} y_u$.

3.1.2. Állítás. *A Held-Karp korlát egy alsó becslést ad az optimális körút költségére.*

Bizonyítás. Jelöljük az optimális körutat O-val, az 1-fa korlátot pedig C-vel, az új élsúlyozással kapott optimális körutat O' fogja jelölni, az új 1-fa

korlátot C'. Az 1-fa korlát bizonyításánál beláttuk, hogy $w(O) \geq w(C)$. Mivel minden körút költsége $2 \sum_{u \in V \setminus v} y_u$ értékkel nő, így $w(O) = w(O') - 2 \sum_{u \in V \setminus v} y_u$. Tehát $w(O) = w(O') - 2 \sum_{u \in V \setminus v} y_u \geq w(C') - \sum_{u \in V \setminus v} y_u$, ahogy állítottuk. $\square$

Held és Karp megállapították, hogy mivel az optimális körútban minden csúcs foka 2, ezért arra kell törekedni, hogy az 1-fa korlátban szereplő feszítő fa csúcsainak fokszámát úgy változtassuk meg az eljárás során, hogy azok minél közelebb legyenek az optimális körútéhoz. A javaslatuk a következő, minden u csúcsra változtassuk meg a súlyozást úgy, hogy a fokszámukból vonjunk ki 2-t, szorozzuk be egy tetszőleges t számmal és ezt az értéket adjuk hozzá az u csúcs eddigi súlyához. Ezzel a módszerrel a nagyfokú csúcsok fokszáma csökkeni, míg az elsőfokú csúcsoké pedig nőni fog, a 2-es fokszámra törekedve.

A módszer előnye, hogy mindig újabb súlyozást véve a csúcsokra, a korlát javítható. A tapasztalat azt mutatja, hogyha a t-t jól választjuk meg, akkor így kapott Held-Karp korlátok egy optimális Held-Karp korláthoz tartanak, bár az nem igaz, hogy mindig minden újabb súlyozást véve jobb korlátot kapunk.

Irodalomjegyzék

[1] Lovász-Pelikán-Vesztergombi: *Diszkrét matematika*, Typotex kiadó, Budapest, (2006)

[2] Katona Gyula Y.-Recski András-Szabó Csaba: *A számítástudomány alapjai*, Typotex kiadó, Budapest, (2006)

[3] Király Tamás, Kis Tamás, Szegő László:*Online jegyzet az Egészértékű Programozás I. és II. tárgyhoz*, (2013)

[4] Hajnal Péter: *Gráfelmélet*, Polygon, Szeged, (2003)

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein:*Új algoritmusok*, Scolar kiadó, Budapest, (2003)

[6] `http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm, 2015.05.21.`

[7] `http://en.wikipedia.org/wiki/Bellman-Ford_algorithm, 2015.05.21.`

[8] `http://tamop412.elte.hu/tananyagok/algoritmusok/index.html, 2015.05.21.`

[9] `http://www.cs.bme.hu/algel/pp10elo.pdf, 2015.04.15.`