27. MINIMÁLIS KÖLTSÉGŰ FESZÍTŐFÁK

Az irányítás nélküli élsúlyozott gráfokon megfogalmazható feladat az, amely a csúcsokat összekötő *minimális költségű fa* (*feszítőfa*) megkeresését tűzi ki célul. Ez a feladat hasonlóan egyszerű és alapvetően fontos, mint az útkeresés, szintén komoly gyakorlati alkalmazásokkal a háttérben.

A probléma megjelenése egy időszakban, a villamosítás éveiben elég gyakori volt. Ha egy terület villamosítását kell megoldani a lehető legkisebb költséggel, akkor a feladat minimális összköltségű vezetékrendszer tervezése megadott helységek között.

A modellünk legyen irányítás nélküli, súlyozott gráf, ahol a városoknak megfeleltetjük a gráf pontjait, az éleknek pedig a tervezett, két várost összekötő villamos vezetéket. Az élek irányítás nélküliek az elektromos áram irányítatlan tulajdonsága miatt, és súlyozottak, ahol az élek költségei legyenek a becsült építési költségek.

Az itt következő egyetlen nagy fejezet – elméleti fogalmak rövid áttekintés után – három nagyobb egységből áll. A fejezet felépítése fordított sorrendet követ, mint az elmélet fejlődése. A feladat két ismert megoldásának, *Prim*, illetve *Kruskal algoritmusának* ismertetését megelőzi a *piros-kék eljárás* leírása, amely időben utoljára született.

Az előbb említett két megoldó algoritmus a probléma megoldására közvetlenül implementálható eljárásokat ad. Az általános és többszörösen nem-determinisztikus piros-kék algoritmus viszont inkább az elméleti megközelítés műfajába tartozik, ugyanis az eddig ismert megoldó eljárásokat mind magában foglalja lehetőségként.

Az általános eljárás és a két megoldó algoritmus ismertetését rövid bevezető jellegű, elméleti előkészítés előzi meg.

Definíciók:

- *Részgráf*: Legyen $G = (V, E)$ irányítatlan gráf. A $G' = (V', E')$ gráfot a G részgráfjának nevezzük, ha $V' \subseteq V$ és $E' \subseteq E$, továbbá $\forall\, (u, v) \in E' : u, v \in V'$.
- *Feszítőfa*: Legyen $G = (V, E)$ irányítatlan, összefüggő, véges gráf. A G egy körmentes, összefüggő $F = (V, E')$ részgráfját a G egy feszítőfájának nevezzük. (A jelölésből látható, hogy F és G csúcsainak halmaza megegyezik.)
- *Minimális költségű feszítőfa*: Legyen $G = (V, E, c)$ irányítatlan, összefüggő, élsúlyozott, véges gráf a $c: V \times V \to \mathbb{R}$ költségfüggvénnyel. Ekkor $F = (V, E')$ feszítőfa a G egy minimális költségű feszítőfája, ha költsége $C(F) = \sum_{(u,v)\in E'} c(u, v)$ minimális a G feszítőfái között, azaz $C(F) = \min\{C(H) \mid H \text{ a } G \text{ feszítőfája}\}$.

Feladat: Adott egy $G = (V, E)$ irányítatlan, összefüggő, élsúlyozott, véges gráf. Határozzuk meg G egy minimális költségű feszítőfáját.

A továbbiakban tekintsünk néhány fákkal kapcsolatos tulajdonságot, amelyek a későbbi bizonyítások során hasznosak lehetnek.

Állítás: Minden legalább kétpontú fában van elsőfokú csúcs.

Bizonyítás: Tekintsük $u = \langle v_0, v_1, \dots, v_k \rangle$ egyik leghosszabb utat a fában. Ha v_0-ból menne él egy olyan csúcsba, amely nem eleme $\{v_1, \dots, v_k\}$ halmaznak, akkor u nem lenne a leghosszabb út, ha v_0-ból menne él egy olyan csúcsba, amely eleme $\{v_1, \dots, v_k\}$ halmaznak, akkor az útban lenne kör, tehát nem lenne fa. Így azt kaptuk, v_0 elsőfokú csúcs.

Állítás: Minden összefüggő $G = (V, E)$ gráfnak van feszítőfája.

Bizonyítás: Ha a gráfban van kör, elhagyjuk az egyik élét. Ezt véges sokszor ismételve körmentes, összefüggő V csúcshalmazú gráfot kapunk, tehát feszítőfát.

Állítás: Egy n pontú összefüggő gráf fa akkor, és csak akkor, ha $n - 1$ éle van.

Bizonyítás:

$\Longrightarrow$: Ha egy n pontú fából törlünk egy elsőfokú csúcsot és a hozzá tartozó élt, akkor egy $n - 1$ pontú fát kapunk. Ezt ismételve, $n - 1$-szer lehet elsőfokú csúcsot elhagyni a hozzá tartozó éllel együtt, mivel a végén már csak egyetlen csúcs marad, tehát az eredeti fának n-1 éle volt.

$\Longleftarrow$: Legyen F egy n pontú, $n - 1$ élű összefüggő gráf, továbbá legyen F' egy feszítőfája F-nek. Az előbb igazoltak szerint F'-nek is $n - 1$ éle van, tehát $F = F'$.

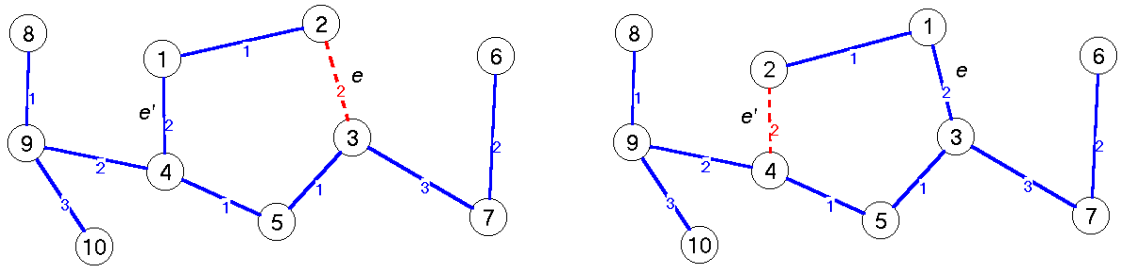
Állítás: Egy fa bármely két pontja között pontosan egy út vezet.

Bizonyítás: Indirekt tegyük fel, hogy u-ból v-be két út vezet, ekkor u-ból v-be elmegyek az egyik úton, majd visszajövök a másik úton, akkor legkésőbb u-ba jutva találok egy olyan csúcsot, amely eleme az első útnak, tehát kört találtam.

Állítás: Legyen $G = (V, E)$ gráfnak $F = (V, E')$ egy minimális költségű feszítőfája, továbbá, legyen $e = (u, v) \in E$ a G-nek egy olyan éle, ami nem éle F-nek ($e \notin E'$). Tegyük fel, hogy az F-beli u-ból v-be vezető úton van olyan e' él ($e' \notin E'$), amelyre $c(e) \leq c(e')$. F-ből az e él hozzá vételével és az e' elhagyásával kapott F' gráf is minimális költségű feszítőfája G-nek.

Bizonyítás: Vegyük hozzá F-hez e élt, ekkor a kapott gráfban van olyan kör, amelynek e' éle. Az e' törlésével kapott gráf tehát összefüggő marad és éleinek a száma is ugyanannyi, mint F éleinek a száma, így F' is feszítőfája G-nek. Továbbá $c(F') \leq c(F)$, mivel $c(e) \leq c(e')$, azaz egy nem nagyobb költségű éllel cseréltünk le egy élt.

Szemléltessük az utolsó állítást a 27.1. ábrán. Legyenek $u = 2$, $v = 3$ csúcsok, továbbá $e = (2, 3)$, $e' = (1, 4)$ az állításban említett élek. Az állítás szerint, ha e' él helyett e élt vesszük fel a feszítőfa éle közé, akkor áttérünk a G-nek egy másik minimális költségű feszítőfájára.



27.1. ábra. Az élcsere hatása a feszítőfára

27.1. A piros-kék eljárás

A fejezetben tárgyalt algoritmusok közös vonása, hogy valamilyen módszer szerint sorra veszik a gráf éleit, és egyes éleket bevesznek a kialakuló minimális költségű feszítőfába, másokat pedig nem. Ezen algoritmusok általánosításaként *Robert E. Tarjan* adott egy szép, nem determinisztikus eljárást, melyet *piros-kék eljárás*ként emlegetnek. A szemléletes tárgyalás érdekében az éleket szokás beszínezni, innen származik a módszer neve is.

A módszer kékre színezi a minimális költségű feszítőfába bekerülő élt, és pirosra színezi azokat az éleket, amelyek már biztosan nem kerülnek be a fába. Az élek színezése során két

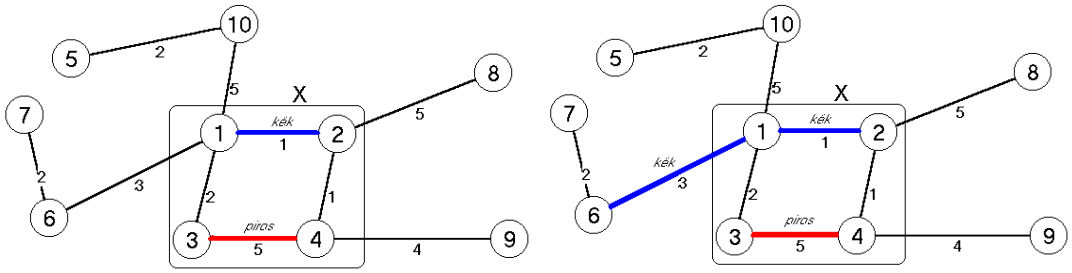
szabályt fogunk alkalmazni a piros szabályt és a kék szabályt. A két szabályt tetszőleges sorrendben és tetszőleges helyen alkalmazhatjuk, akár véletlenített módon.

A később ismertetésre kerülő algoritmusokat (Prim, Kruskal) tekinthetjük úgy is, mint a piros-kék eljárás egy-egy specializált változatait. Az algoritmus ismertetése előtt bevezetjük a szükséges definíciókat.

Definíciók:

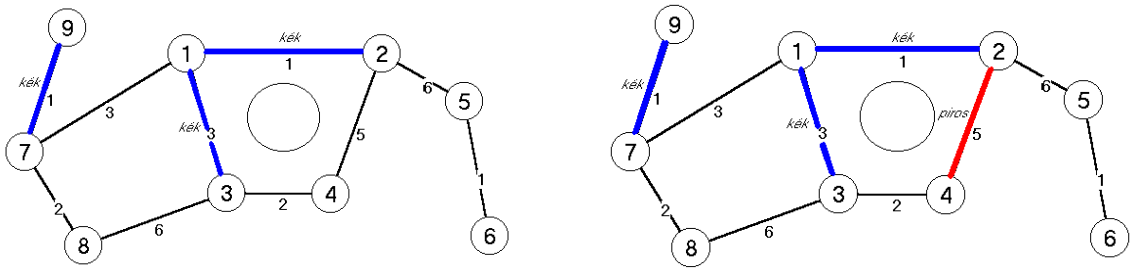
- *Megfelelő színezés*: Tekintsük a $G = (V, E)$ irányítatlan, súlyozott véges gráf éleinek egy színezését, amelynél egy él lehet piros, kék vagy színtelen. Ez a színezés megfelelő, ha létezik G-nek olyan minimális költségű feszítőfája, ami az összes kék élt tartalmazza, de egyetlen piros élt sem tartalmaz.
- Kék szabály: Válasszunk ki egy olyan $X \subset V$, $X \neq \emptyset$ csúcshalmazt, amiből nem vezet ki kék él. Ezután egy legkisebb súlyú W-ből kimenő színtelen élt fessünk kékre.

  Tekintsük a 27.2. ábrán szereplő példán a kék szabály egy alkalmazását. Legyen $X = \{1, 2, 3, 4\}$ halmaz. Látható, hogy X-ből nem vezet ki kék él. Színezzük kékre az X-ből "kivezető" egyik legkisebb súlyú élt, amely most a 3-as súlyú $(1, 6)$ él.



27.2. ábra. A kék szabály alkalmazása

- *Piros szabály*: Válasszunk G-ben egy olyan egyszerű kört, amiben nincs piros él. A kör egyik legnagyobb súlyú színtelen élét színezzük pirosra. A szabály egy alkalmazását illusztráljuk a 27.3. ábrán. A szabályban említett kör legyen $\langle 1, 2, 3, 4 \rangle$, amely nem tartalmaz piros élt. Keressük meg a kör egyik legnagyobb súlyú élét, amely az 5-ös súlyú $(2, 4)$ él. Színezzük pirosra.



27.3. ábra. A piros szabály alkalmazása

A fenti szabályok ismeretében a piros-kék eljárás könnyen megfogalmazható. Legyen kezdetben a $G = (V, E)$ irányítatlan, súlyozott, összefüggő, véges gráf minden éle színtelen. Alkalmazzunk a két szabályt tetszőleges sorrendben és helyen, amíg csak lehetséges.

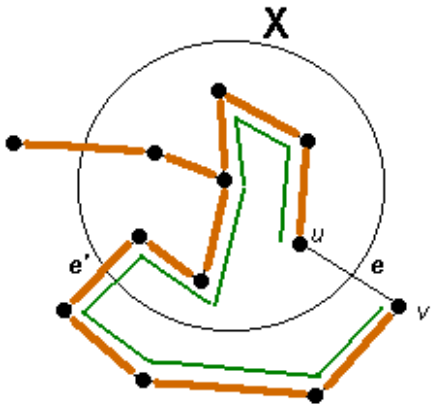
Állítás: Legyen $G = (V, E)$ irányítatlan, súlyozott, összefüggő, véges gráf, és $n = |V|$. Ekkor

1. a piros-kék eljárás során *a színezés mindig megfelelő marad*;
2. a színezéssel *sosem akadunk el*, ameddig G minden éle színes nem lesz;

3. ha beszíneztük G minden élét, akkor *a kék élek G egy minimális költségű feszítőfájának éleit adják*, sőt már $n-1$ kékre színezett él után is megkaptuk az említett feszítőfát.

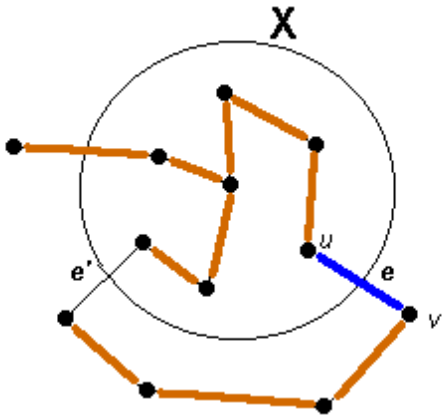
Bizonyítás:

- Teljes indukcióval lássuk be az állítást. Kezdetben, amikor minden él színtelen nyilván teljesül a megfelelő színezés. Továbbiakban tegyük fel, hogy egy olyan állapotban vagyunk, amelyre teljesül a megfelelő színezés. Legyen F a G egy olyan minimális költség feszítőfája, amely az összes, jelenleg kékre színezett élt tartalmazza, és egyetlen, jelenleg pirosra színezett élt sem tartalmaz. Tegyük fel, hogy az eljárás következő lépése során az $e=(u,v)\in E$ élt színeztük be. Két eset lehet attól függően, hogy melyik szabályt alkalmaztuk.
    - *A kék szabályt alkalmaztuk*: ekkor nyilván e színe kék lett.
        a) Ha e éle F-nek, akkor F mutatja, hogy megfelelő a színezés.
        b) Ha e nem éle F-nek, akkor tekintsük az $X\subset V$ halmazt, amire a kék szabályt alkalmaztuk. Az F-ben $\exists\ u\rightsquigarrow v$ út, hiszen F feszítőfa, továbbá ezen az úton van olyan e' él, ami kimegy X-ből. (Ugyanis e-t színeztük kékre, tehát a kék szabály értelmében e egyik vége X-en belül, a másik vége X-en kívül van. Továbbá az említett $u\rightsquigarrow v$ F-beli út, egy X-beli és egy X-en kívüli pontot köt össze, tehát valahol ki kell lépnie X-ből. Lásd 27.4. ábra.)



27.4. ábra. Gráf, amelyben a kijelölt él nem része a feszítőfának (a feszítőfa színesen jelölve)

Vizsgáljuk, milyen lehet e' színe. Piros nem lehet, mivel része F-nek, kék sem lehet, mivel a kék szabályt alkalmaztuk, amely szerint X-nek olyannak kell lennie, amiből nem vezet ki kék él. Tehát e' színtelen. Továbbá $c(e)\leq c(e')$, mivel a kék szabály szerint az X-ből kimenő egyik legkisebb súlyú élt kell választani, és mi e-t választottuk. Alkalmazhatjuk a korábbi állítást, mely szerint F-ből e' törlésével és e hozzá vételével kapott új F' gráf is a G egy minimális költségű feszítőfája. Tehát F' igazolja, hogy e kékre színezésével a színezés továbbra is megfelelő marad (27.5. ábra).

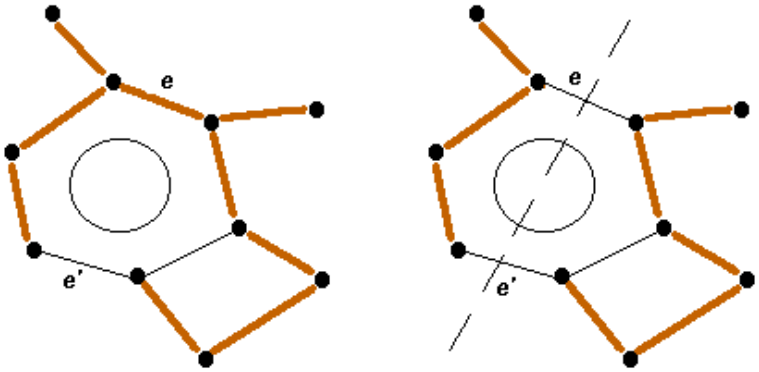


27.5. ábra. Egy élcserével kialakított új feszítőfa (színesen jelölve)

- *A piros szabályt alkalmaztuk*: ekkor nyilván e színe piros lett.

  a) Ha e nem éle F-nek, akkor F mutatja, hogy megfelelő a színezés.

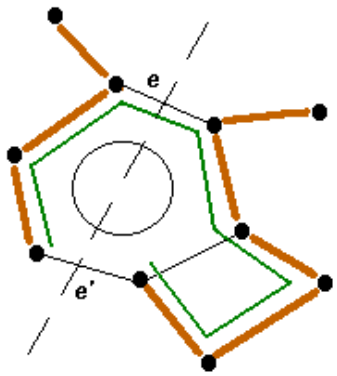
  b) Ha e éle F-nek, akkor a pirosra színezés azt jelenti, hogy e továbbiakban már nem lehet éle az eljárás során előállítás alatt lévő minimális feszítőfának, tehát a megfelelő színezés bizonyításához át kell térni egy másik minimális feszítőfára. Az e F-ből való törlésével F két komponensre esik szét. Tekintsük azt a kört, amelyre a piros szabályt alkalmaztuk, ennek van olyan e' éle, amelyik a két komponenst összeköti és nem éle F-nek. (Ugyanis a két komponenst összekötő e-től különböző élnek lennie kell, mivel kör mentén vizsgálódunk, és egy körbeli él elhagyásával az összefüggőség nem szűnhet meg. Továbbá, ha nem lenne ilyen e' él, ami nem éle F-nek, az azt jelenti, hogy a kör minden éle F éle is, tehát kör lenne a fában. Lásd 27.6. ábra.)



27.6. ábra. Egy él elhagyásával keletkező két feszítőfa (színesen jelölve)

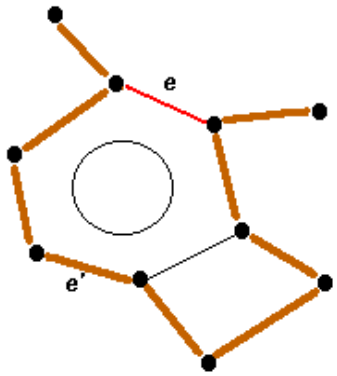
Vizsgáljuk, milyen lehet e' színe. Nem lehet kék, mivel nem éle F-nek és feltettük, hogy a színezés megfelelő, amit F mutat. Nem lehet piros, mivel a piros szabály értelmében, olyan kört kell választani, amiben nincs piros él. Tehát e' színtelen. Továbbá $c(e') \leq c(e)$, mivel a piros szabály alkalmazása során e-t választottuk színezésre, amely szerint a kör egyik legnagyobb súlyú élét kell pirosra színezni. Az e' végpontjait összekötő F-beli út tartalmazza e élt. (Ugyanis e törlése előtt F feszítőfa volt, és a korábbi állítás szerint, bármely két pontja között pontosan egy út vezet. Azonban most két olyan részre esett szét, amelynek egyik komponensében van e' egyik vége, a másik komponensében e'másik

vége. F-ben a két komponens között az átjárást éppen az e él biztosította, tehát az említett útnak át kell haladnia az e élen. Lásd 27.7. ábra.)



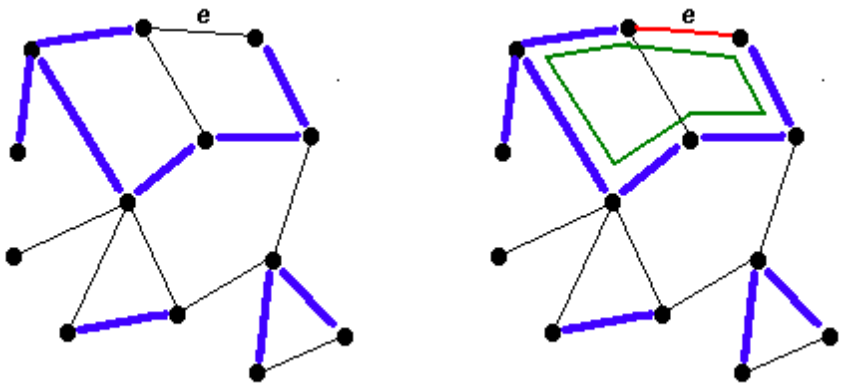
27.7. ábra. Az él két végpontját összekötő út (zölddel jelölve)

Alkalmazhatjuk a korábban belátott állítást, mely szerint F-ből e törlésével és e' hozzá vételével kapott új F' gráf is G egy minimális költségű feszítőfája. Tehát F' igazolja, hogy e pirosra színezésével a színezés továbbra is megfelelő marad (27.8. ábra).



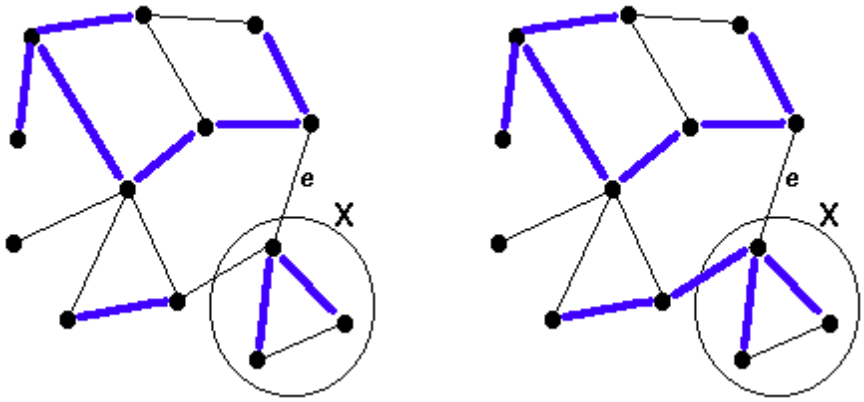
27.8. ábra. A kijelölt él pirosra színezésével kialakult új feszítőfa

- Most belátjuk, hogy a színezéssel sosem akadunk el, ameddig G minden éle színes nem lesz. Tegyük fel, hogy G-nek még nem minden éle színes. Legyen e egy színtelen él. A színezés megfelelősége miatt a kék élek egy erdőt alkotnak (de lehet, hogy már egy fát, ekkor az alábbi 1. eset alkalmazható), az erdő fáit nevezzük kék fáknak. Két eset lehetséges:
    - Ha e két végpontja ugyanabban a kék fában van (27.9. ábra). Ekkor a piros szabályt alkalmazhatjuk arra a körre, aminek az éleit úgy kapjuk, hogy az e két végpontját összekötő egyetlen kék úthoz hozzávesszük e-t.



27.9.a ábra. Amennyiben az él két végpontja ugyanabban a fában van, a piros szabályt alkalmazzuk (zölddel jelölve a kialakult kör)

- Ha e két végpontja különböző kék fában van. Ekkor a kék szabály alkalmazható a következőképpen. X legyen az egyik olyan kék fa csúcsainak halmaza, amelyikben benne van e egyik vége. Ebből a kék fából, azaz X-ből biztosan megy ki él (legalább e), e kimenő élek közül az egyik legkisebb súlyú (nem biztos, hogy e) kékre színezhető.



27.9.b ábra. Amennyiben az él két végpontja más fában van, a kék szabályt alkalmazzuk

- A harmadik állítás szerint, végül megkapjuk G egy minimális költségű feszítőfáját. Ez rögtön következik abból, hogy a végső színezés is megfelelő. Az állítás második része szerint, az eljárást elegendő addig folytatni, míg $n-1$ kék él nem lesz. A korábbi állítás szerint, a feszítőfának összesen $n-1$ éle van, tehát ha már van $n-1$ kék élünk, akkor a továbbiakban több nem is keletkezhet.

Tehát a piros és kék szabályt tetszőleges helyen és sorrendben alkalmazva, végül minimális költségű feszítőfát kapunk, azonban hatékonysági szempontból megfontolandó melyik szabályt mikor és hol alkalmazzuk. A következő algoritmusokat a piros-kék eljárás egy-egy speciális esetének is tekinthetjük.

27.2. A Prim-algoritmus

A Prim algoritmus minden lépésben a kék szabályt alkalmazza egy s kezdőcsúcsból kiindulva. Az algoritmus működése során egyetlen kék fát tartunk nyilván, amely folyamatosan növekszik, míg végül minimális költségű feszítőfa nem lesz.

Kezdetben a kék fa egyetlen csúcsból áll, a kezdőcsúcsból, majd minden lépés során, a kék fát tekintve a kék szabályban szereplő X halmaznak, megkeressük az egyik legkisebb súlyú élt (mohó stratégia), amelynek egyik vége eleme a kék fának (X-ben van), a másik vége viszont nem (azaz nem eleme X-nek). Az említett élt hozzá vesszük a kék fához, azaz az élt kékre színezzük, és az él X-en kívüli csúcsát hozzávesszük az X-hez.

27.2.1. Az absztrakt szintű algoritmus

Az algoritmus megvalósításának a kulcsa az X-ből kimenő egyik legkisebb súlyú él meghatározása. Ehhez használjunk egy minimum választó elsőbbségi (prioritásos) sort ($minQ$), amelyben a fához még nem tartozó (még nem eleme X-nek) csúcsokat tároljuk az X-től való távolsággal, mint kulcs értékkel. A távolság elnevezéséből adódóan és a korábbi algoritmusokhoz hasonlóan, jelöljük a kulcsot egy $v \in V$ csúcs esetén $d[v]$-vel.

Egy $v \in V$ csúcs esetén az X-től való távolság, azaz a $d[v]$ legyen azon élek közül a minimális súlyú él súlya, amely v és egy X-beli csúcs között halad. Amennyiben nem létezik él v és egy tetszőleges X-beli csúcs között, legyen $d[v] = \infty$.

A korábbi algoritmusokhoz hasonlóan, a $P[1 \ldots n]$ tömbbe tároljuk el egy csúcs feszítőfabeli megelőzőjét (szülőjét), amelynek segítségével bejárható a fa.

Az algoritmus elvénél, azt mondtuk, hogy kezdetben a kék fa legyen egyetlen pont, a kezdőcsúcs. Most az X-től való távolság fogalmának bevezetésével, azt mondhatjuk, hogy kezdetben X legyen az üres halmaz, amelytől a kezdőcsúcs nulla távolságra van, az összes többi csúcs pedig végtelen távolságra. Az algoritmus leírásában az X halmazt explicite nem ábrázoljuk, hanem $X = V \setminus minQ$.

Az algoritmus minden lépésében kivesszük a $minQ$ (egyik) legkisebb kulcsú elemét (az X-ből kimenő egyik legkisebb súlyú él X-en kívüli csúcsát), azaz a készülő feszítőfához, X-hez hozzávesszük az illető csúcsot. Majd az X-en kívüli csúcsok X-től való távolságát, mint invariáns tulajdonságot karban kell tartani. Nyilván elegendő az X-be újonnan bekerült csúcs szomszédjainak az X-től való távolságát módosítani (ha szükséges), mivel egy v csúcs úgy kerülhet közelebb X-hez, hogy valamelyik u szomszédja bekerül az X-be. Ekkor v távolsága a következőképpen alakul:

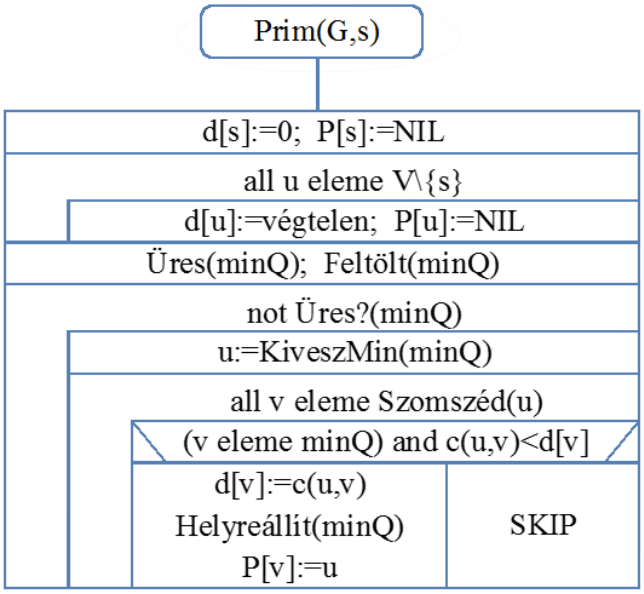
- ha $d[v] = \infty$, akkor most legyen $d[v] = c(u, v)$;
- ha $d[v] < \infty$, akkor már létezik v-nek olyan w szomszédja, amely eleme X-nek tehát $d[v]$ akkor változik, ha az (u, v) élen keresztül v közelebb van X-hez, mint (v, w) él esetén.

Eközben a $P[1 \ldots n]$ szülőségi tömböt is karban kell tartani.

Tehát a használt típusok és adatszerkezetek:

- $P[1 \ldots n]$ tömb: egy csúcs feszítőfabeli szülőcsúcsának a tárolására.
- $minQ$: $(d[v], v)$ párokból álló minimumválasztó elsőbbségi sor, ahol $d[v]$ értéke a kulcs.

Az algoritmus ADS szintű megvalósítása látható a 27.10. ábrán.



Műveletigény

rendezettlen tömb	**kupac**
1	
"(n-1)-szer"	
n-1	
0+n (X kinullázása)	n
"n-szer"	
n*n	n*logn
"e-szer"	
e*1	e*1
e*1	e*1
0	e*logn
e*1	e*1

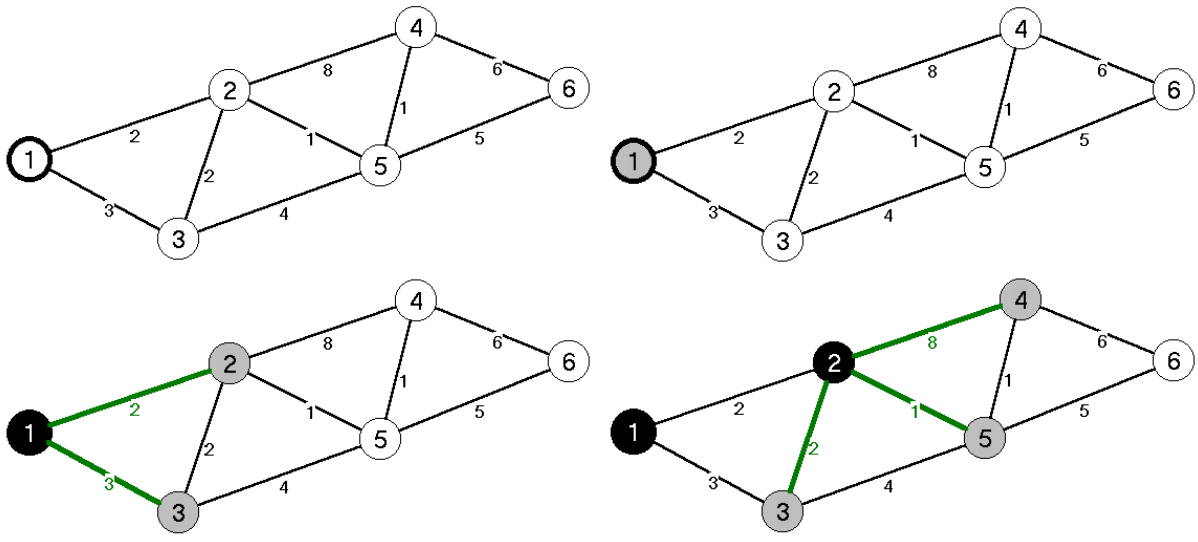
27.10. ábra. A Prim algoritmus és műveletigénye

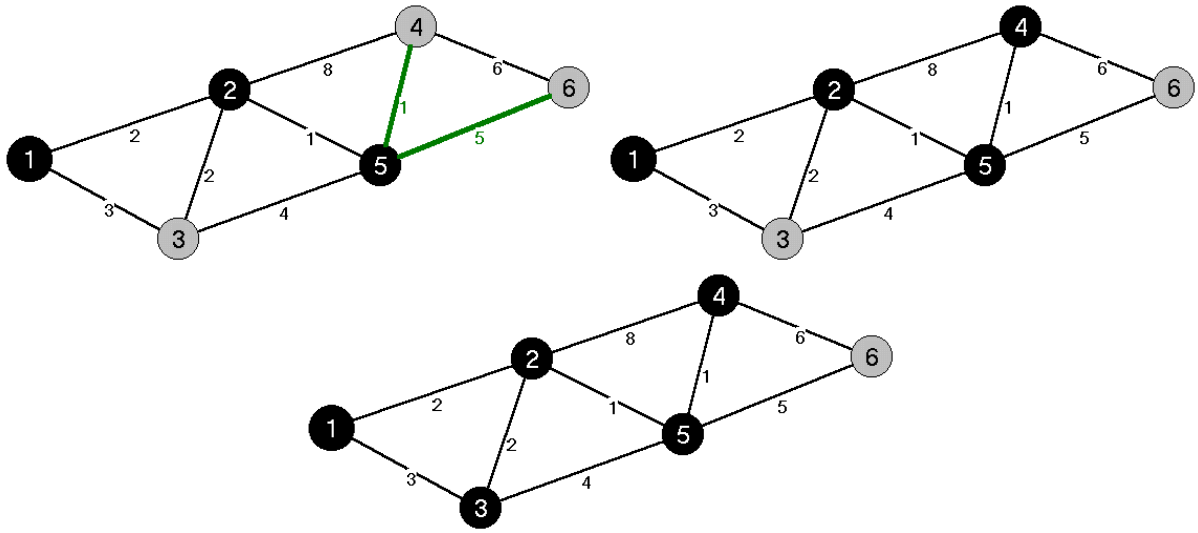
27.2.2. Az algoritmus szemléltetése

Nézzük meg egy példán, ADS szinten az algoritmus működését a 27.10. ábrán. A szemléltetés érdekében színezzük a csúcsokat a következőképpen:

- *Fehér*: a csúcs eleme a $minQ$-nak és nincs X-beli szomszédja, azaz még nem került "látótávolságba", tehát az X-től való távolsága végtelen.
- *Szürke*: a csúcs eleme a $minQ$-nak, de létezik X-beli szomszédja, tehát a távolsága már kisebb, mint végtelen.
- *Fekete*: a csúcs kikerült a $minQ$-ból, azaz bekerült X-be.

A példában a kezdőcsúcs legyen az 1-es csúcs. Az inicializáló lépés után az 1-es csúcs kivételével minden csúcs távolsága (az X halmaztól) legyen végtelen, az 1-es csúcs távolsága pedig legyen 0, az X legyen az üres halmaz.





27.11. ábra. A Prim algoritmus lépésenkénti végrehajtása

Az első lépésben kivesszük a $minQ$-ból az 1-es csúcsot (mivel az 1-es csúcs távolsága a legkisebb az X-től), tehát $X = \{1\}$, majd az 1-es csúcs szomszédai (2 és 3) kerülnek közelebb az X-hez. Ezek távolsága $d[2] = 2$ és $d[3] = 3$.

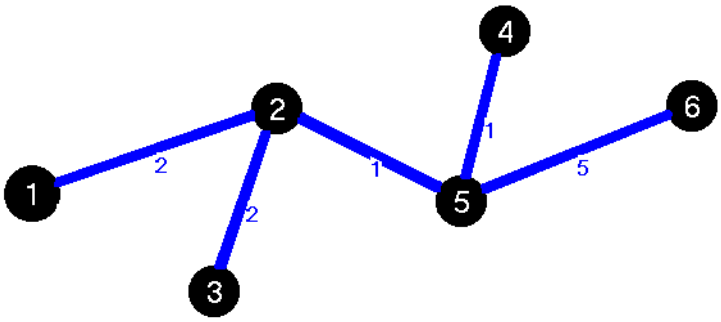
A második lépésben a 2-es csúcs kerül be az X halmazba (feljegyezve az 1-es csúcsot, mint fabeli szülőt), mivel közelebb van X-hez, mint a 3-as csúcs. Ezután a 2-es szomszédai kerülnek "látótávolságba". Megfigyelhető, hogy a 3-as csúcs $d[3] = 3$ távolságra volt az X-től, de most közelebb került $d[2] = 2$, a $(2, 3)$ él figyelembe vételével, $X = \{1,2\}$.

A harmadik lépésben az $X = \{1, 2\}$ halmazhoz a legközelebb lévő csúcs ($d[3] = 2$, $d[4] = 8$, $d[5] = 1$), az 5-ös csúcs kerül az X halmazba (feljegyezve szülőként a 2-es csúcsot). Az 5-ös (még X-hez nem tartozó) szomszédai, a 4-es és 6-os csúcsok kerülnek közelebb az X-hez.

A negyedik lépésben a nem fekete csúcsok közül a legkisebb távolságú, a 4-es csúcs kerül az X-be. $X = \{1, 2, 4, 5\}$. A 4-es szomszédai, a "4-esen keresztül" már nem kerülnek X-hez közelebb.

Az ötödik lépésben a 3-as csúcs kerül az X-be. A 3-asnak már nincs is nem fekete (nem X-beli) szomszédja.

Végül az utolsó lépésben a 6-os csúcs kerül az X halmazba. A menetközben feljegyzett szülőcsúcsok segítségével meghatározható a feszítőfa (27.12. ábra).



27.12. ábra. A Prim algoritmus futtatása által kialakított feszítőfa

27.2.3. Az algoritmus a reprezentáció szintjén

Vizsgáljuk meg a prioritásos sor ($minQ$) megvalósításának két, természetes módon adódó lehetőségét, ahogy a Dijkstra algoritmusnál is már láttuk:

1. A prioritásos sort valósítsuk meg *rendezetlen tömbbel*, azaz a prioritásos sor legyen maga a $d[1 \dots n]$ tömb. Ekkor a minimum kiválasztására egy *feltételes minimum keresést* kell alkalmazni, amelynek a műveletigénye $\Theta(n)$. A Feltölt($minQ$) és a Helyreállít($minQ$) absztrakt műveletek megvalósítása pedig egy SKIP-pel történik.

   Az algoritmus ADT leírásában az szerepel, hogy a $minQ$-ból kiveszünk egy elemet, azonban a $minQ$-t egy tömbbel valósítjuk meg, amelynek a mérete nem változik. Tehát osztályozni kell a csúcsokat aszerint, hogy a $minQ$-ban vannak-e még, vagy már bekerültek az X halmazba. Legyen egy $X[1 \dots n]$ tömb az alábbi módon definiálva:

   $$X(i) = \begin{cases} 0 & \text{ha } i \notin X \\ 1 & \text{ha } i \in X \end{cases}$$

   Az X tömböt kezdetben ki kell nullázni, majd menet közben karban kell tartani. Amint kikerül egy csúcs a $minQ$-ból, az X tömbben a csúcsnak megfelelő helyre 1-est kell írni.

2. Kupac adatszerkezet használatával is reprezentálhatjuk a prioritásos sort. Ekkor a Feltölt($minQ$) eljárás, egy kezdeti kupacot épít, amelynek a műveletigénye lineáris. Azonban most a $d[1 \dots n]$ tömb változása esetén a kupacot is karban kell tartani, mivel a kulcs érték változik. Ezt a Helyreállít($minQ$) eljárás teszi meg, amely a csúcsot a gyökér felé "szivárogtatja" fel, ha szükséges (mivel a kulcs értékek csak csökkenhetnek). Ennek a műveletigénye $\log n$-es.

   Ennél az ábrázolásnál is vezessünk be egy segédtömböt, a $HOL[1 \dots n]$ tömböt, amely megmutatja, hogy egy csúcs hol helyezkedik el a kupacban (a kupacot $[1 \dots 2n]$ tömbben valósítsuk meg), illetve legyen 0, ha az illető csúcs már nem eleme a minQ-nak. A HOL tömb felhasználásával egy csúcs prioritásos sorban való keresésének műveletigényét konstansra csökkenthetjük. A HOL tömböt a $minQ$ változásakor szintén karban kell tartani.

Megjegyzés: Nem szükséges kezdeti kupacot építeni, felesleges a kupacba rakni a végtelen távolságú elemeket. Kezdetben csak a kezdőcsúcs legyen a kupacban, majd amikor először „elérünk” egy csúcsot és a távolsága már nem végtelen, elég akkor berakni a kupacba.

27.2.4. Műveletigény

A prioritásos sor fenti két megvalósítása esetén, a 27.10. ábrán láthatóan megfelelő módon alakul az algoritmus műveletigénye.

A belső ciklust célszerű globálisan kezelni, ekkor mondható, hogy összesen legfeljebb annyiszor fut le, ahány éle van a gráfnak.

1. Rendezetlen tömb esetén:
   $$T(n) = O(1 + n - 1 + 1 + 0 + n^2 + n + e) = O(n^2 + e) = O(n^2)$$
2. Kupac esetén:
   $$T(n) = O(1 + n - 1 + 1 + n + n \cdot \log n + n + e \cdot \log n) = O\big((n + e) \cdot \log n\big)$$

A Dijkstra algoritmusnál már említett következmény itt is érvényes, azaz *sűrű gráf esetén* csúcsmátrix és rendezetlen tömb, *ritka gráf esetén* éllista és kupac.

27.3. A Kruskal-algoritmus

A Kruskal algoritmus mindkét szabályt alkalmazza a feszítőfa létrehozására, itt azonban egyértelmű él kiválasztási stratégiát alkalmazunk, amely meghatározza, mely szabályt kell alkalmaznunk.

Kezdetben legyen n db kék fa, azaz a gráf minden csúcsa egy-egy (egy pontból álló) *kék fa*, és legyen minden él színtelen. Minden lépés során kiválasztjuk az egyik legkisebb súlyú színtelen élt. Ha a kiválasztott él két végpontja különböző kék fában van, akkor színezzük kékre, különben (az él két vége azonos kék fában van, tehát a kék fa éleivel kört alkot) színezzük pirosra.

A fentiekből kitűnik, hogy a Kruskal algoritmust is tekinthetjük a piros-kék eljárás egy speciális esetének, ahol az élek színezésének a sorrendje egyfajta mohó stratégia szerint történik ("még mohóbb", mint a Prim algoritmusnál). Ugyanis

- Amikor egy e élt pirosra színezünk, akkor arra az egyszerű körre alkalmazható a piros szabály, amelynek élei az e, és az e két végpontját összekötő kék út élei. Ez egy egyszerű kör, mivel pontosan egy e végpontjait összekötő kék út létezik, továbbá az e kivételével, minden éle kék, tehát e színezése előtt nem tartalmazott piros élt. Így teljesülnek a piros szabály feltételei.
- Amikor egy e élt kékre színezünk, akkor e két kék fát köt össze, F_1-et és F_2-őt. A kék fák definíciójából következik, hogy F_1 csúcsainak halmazából nem vezet ki kék él. Legyen $X = \{F_1 \text{ csúcsainak a halmaza}\}$, ekkor az e él lesz az egyik legkisebb súlyú X-ből kimenő színtelen él, mivel e az egyik legkisebb súlyú (nem csak X-ből kimenő) színtelen él. Tehát teljesülnek a kék szabály feltételei.

27.3.1. Az absztrakt szintű algoritmus

Az algoritmus absztrakt szintjén, diszjunkt halmazokkal való műveleteket fogunk végezni. Tekintsük a kék fák csúcsainak (diszjunkt) halmazait (ezek a halmazok osztályozzák V-t). Amikor az egyik legkisebb súlyú színtelen élt kiválasztjuk, el kell dönteni, hogy a két végpontja azonos vagy különböző halmazban vannak-e. Majd a választól függően:

- Ha azonos halmazban vannak, akkor a kiválasztott élt színezzük pirosra.
- Ha különböző halmazban vannak, akkor a kiválasztott élt színezzük kékre, és a két különböző halmazt vonjuk össze, azaz a két halmaz helyett legyen egy halmaz, amely megegyezik a két halmaz uniójával.

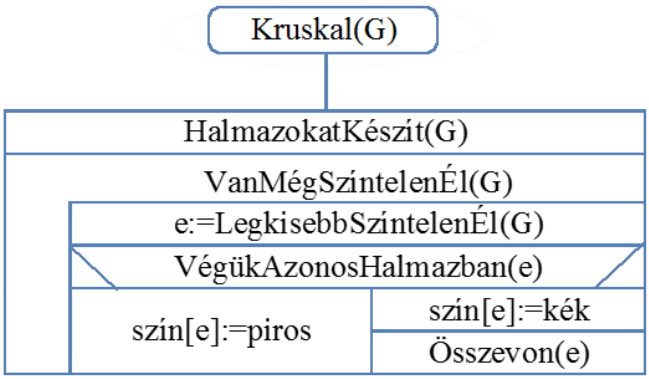
Az algoritmust akkor áll le, ha már nincs színtelen él (leállhatna már akkor is, ha az előbb következne be, hogy beszínezett $n - 1$ db kék élt). Mivel véges sok élünk van, és minden lépésben beszínezünk egyet, így $|E|$ lépés után az algoritmus biztosan befejezi a működését.

Az algoritmusban a következő absztrakt műveleteket szeretnénk használni:

- $\text{HalmazokatKészít}(G)$: Elkészíti a kezdeti n db, pontosan egy csúcsot tartalmazó diszjunkt halmazokat.
- $\text{Összevon}(e)$: Az e él két végpontja által reprezentált halmazokat összevonja.
- $szín[e] := \ldots$: Az e él színét változtatja meg az értékadás jobb oldalán szereplő színre.
- $\text{VanMégSzíntelenÉl}(G)$: igazat ad vissza, ha G-ben még van színtelen él, egyébként hamisat.

- VégükAzonosHalmazban(e): igazat ad vissza, ha e két végpontja azonos halmazban van, egyébként hamisat.
- LegkisebbSzíntelenÉl(G): Visszaadja a legkisebb súlyú színtelen élt.

Az algoritmus megvalósítása a 27.13. ábrán látható.

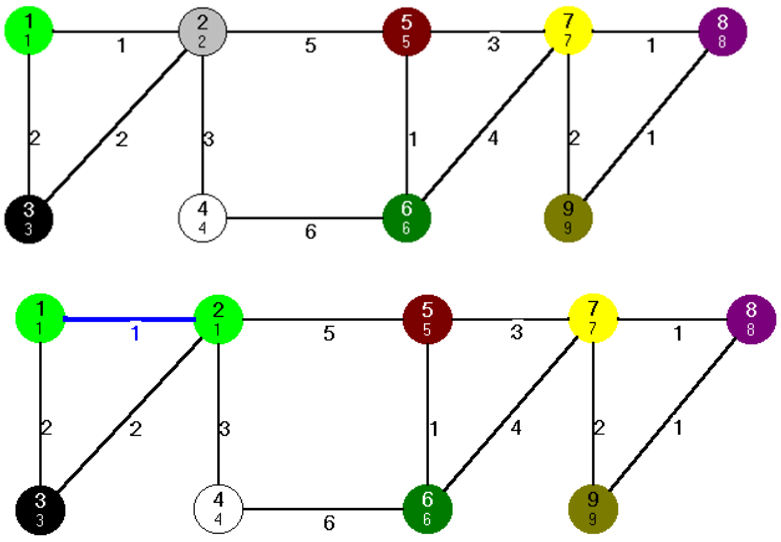


27.13. ábra. A Kruskal algoritmus

27.3.2. Az algoritmus szemléltetése

A következő példában a csúcsok osztályokhoz (halmazokhoz/kék fához) való tartozását színezéssel illetve címkézéssel oldottuk meg. Az azonos színű csúcsok, azonos osztályba tartoznak. Tudjuk, hogy az osztályok reprezentálhatók egy-egy elemükkel, ezért az ábrán (a csúcs címkéje alatt), feltüntettük azon osztály egy reprezentáló elemének a címkéjét, amelyhez az illető csúcs tartozik. Tehát azok a csúcsok tartoznak egy osztályba (azonos kék fához), amelyeknél a címkéjük alatt megjelenő, méretét tekintve kisebb szám azonos.

Az inicializáló lépés után, minden él színtelen és minden csúcs külön osztályt alkot (27.13. ábra). Az első lépésben kiválasztjuk az egyik legkisebb súlyú élt, legyen ez (1,2), és az 1-es ill. 2-es csúcsokat tartalmazó (egyelemű) halmazokat összevonjuk egyetlen $H = \{1, 2\}$ halmazzá. Az új halmaz reprezentáns eleme legyen az 1-es csúcs.



27.14. ábra. A Kruskal algoritmus inicializálása és első lépése

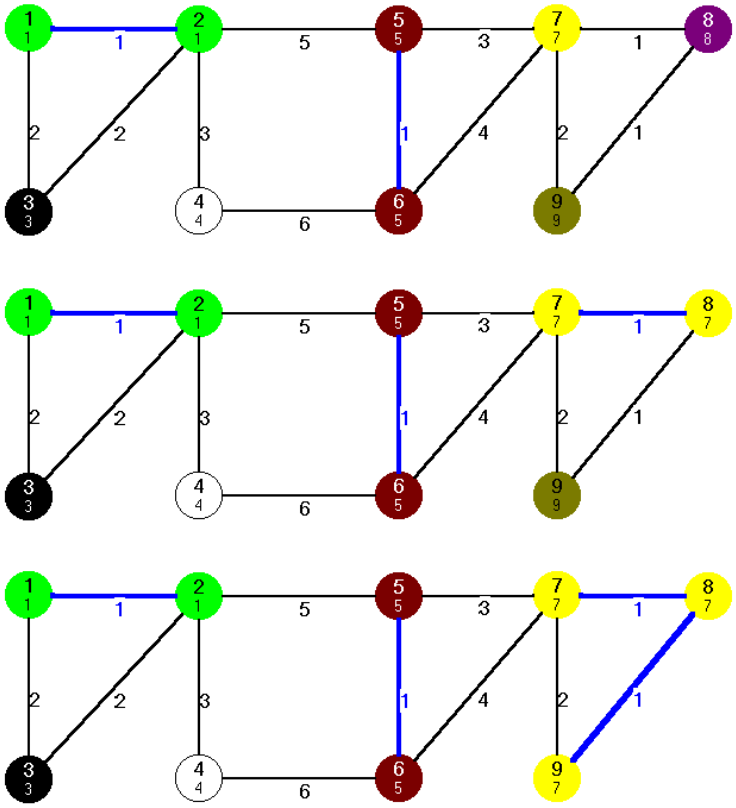
A következő lépésben, az első lépéshez hasonlóan járunk el az 5-ös és 6-os csúcsokkal. A harmadik lépésben, még mindig egyelemű halmazokat vonunk össze, most a 7-es és 8-as

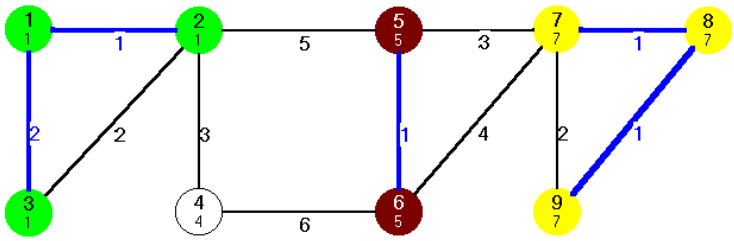
csúcsok osztályait. A negyedik lépésben a kiválasztásra kerülő $(8,9)$ él, még mindig két különböző kék fát köt össze, így kékre kell színezni és a $H_1 = \{7,8\}$ és $H_2 = \{9\}$ halmazokat össze kell vonni a $H = \{7,8,9\}$ halmazzá. Az ötödik lépésben már nincs 1-es súlyú él. A következő egyik legkisebb súlyú él, valamelyik 2-es súlyú él lesz. Mi most válasszuk az $(1,3)$ élt, amelyet kékre színezünk, és a végpontjainak megfelelő halmazokat összevonjuk. Eddig csak a kék szabályt alkalmaztuk, ahogy ez a 27.14. ábrán látható.

A hatodik lépésben kiválasztott $(2,3)$ él két végpontja azonos kék fához tartozik, ezért színezzük pirosra (27.15. ábra). A hetedik lépésben ismét a piros szabályt alkalmazzuk, most a $\langle 7,8,9\rangle$ körre, amelynek következtében a (7,9) él piros lesz.

A nyolcadik lépésben a $(2,4)$ élt választjuk ki, és a kék szabályt alkalmazhatjuk az $X = \{1,2,3\}$ halmazra (27.16. ábra). Tehát a $(2,4)$-es élt kékre színezzük, aminek következtében azonos kék fába kerülnek az $\{1,2,3,4\}$-es csúcsok. A Kruskal algoritmusnak megfelelően, a kék fák nyilvántartására, vonjuk össze őket egy halmazba. A kilencedik lépésben mindenképpen az $(5,7)$ élt kell választanunk, mert ez az egyetlen 3-as súlyú színtelen él. Az élt színezzük kékre, és a $H_1 = \{5,6\}$, $H_2 = \{7,8,9\}$ halmazokat vonjuk össze. A tízedik lépésben az egyetlen 4-es súlyú színtelen él kerül kiválasztásra, amelynek két végpontja azonos osztályba esik, ezért pirosra színezzük.

A tizenegyedik lépésben a (2,5) él a legkisebb súlyú színtelen él. Mivel a 2-es és 5-ös csúcsok különböző osztályokhoz tartoznak, így az élt színezzük kékre, és a $H_1 = \{1,2,3,4\}$, $H_2 = \{5,6,7,8,9\}$ halmazokat vonjuk össze! A halmazok összevonása után már csak egy H={1,2,3,4,5,6,7,8,9} osztályunk (kék fánk) maradt. A továbbiakban már nem alkalmazhatjuk a kék szabályt, azaz megkaptunk egy minimális költségű feszítőfát, amelynek élei: $(1,3)$, $(1,2)$, $(2,4)$, $(2,5)$, $(5,6)$, $(5,7)$, $(7,8)$, $(8,9)$



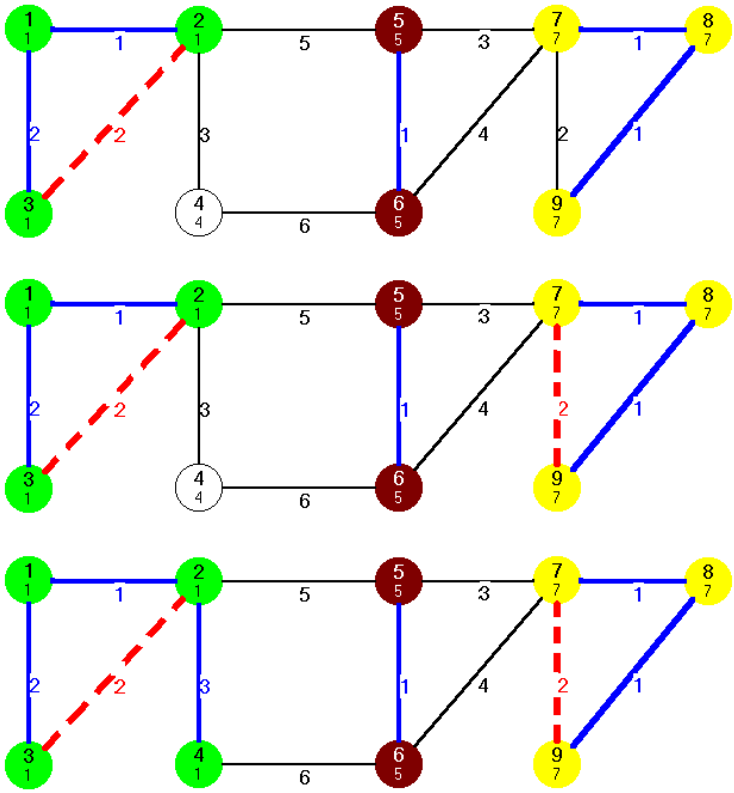


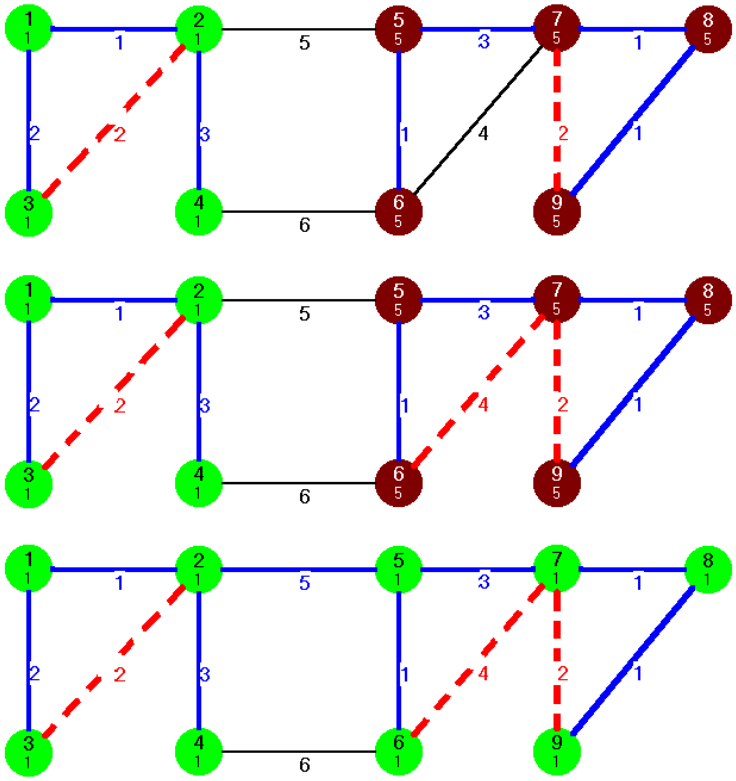
27.15. ábra. A kék szabály alkalmazásai

Az ADT szintű leírás szerint még maradt egy lépés, mivel még van egy színtelen él $(4, 6)$. Természetesen ezt az élt már csak pirosra színezhetjük (27.17. ábra).

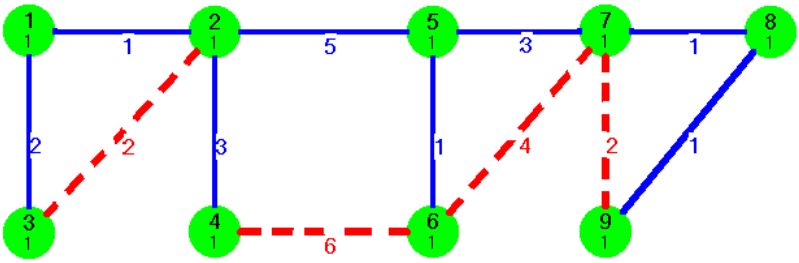
27.3.3. Műveletigény

Az ábrázolás szintjén nem tárgyaljuk az algoritmust. Jobban szemügyre véve a Kruskal algoritmust, a műveletigénye a diszjunkt halmazok megvalósításától függ. Amennyiben az éleket egy kupac adatszerkezetben tároljuk az élsúlyokkal, mint kulccsal, egy él kivétele $O(\log e)$, e él kivétele $O(e \log e)$. Tehát jó lenne olyan ábrázolást választani a diszjunkt halmazoknak, hogy a teljes algoritmus műveletigénye $O(e \log e)$ maradjon. Ilyen reprezentáció létezik; ez az *UNIÓ-HOLVAN* adatszerkezet (nem tárgyaljuk).





27.16. ábra. A feszítőfa kialakításának további lépései



27.17. ábra. A Kruskal algoritmus záró lépése