Algoritmuselmélet 10. előadás

Katona Gyula Y.
Budapesti Műszaki és Gazdaságtudományi Egyetem
Számítástudományi Tsz.
I. B. 137/b
kiskat@cs.bme.hu

2002 Március 25.

Szélességi bejárás

BFS: Breadth First Search

Szélességi bejárás

BFS: Breadth First Search
Meglátogatjuk az első csúcsot, majd ennek a csúcsnak az összes szomszédját.

Szélességi bejárás

BFS: Breadth First Search
Meglátogatjuk az első csúcsot, majd ennek a csúcsnak az összes szomszédját. Aztán ezen szomszédok összes olyan szomszédját, hol még nem jártunk, és így tovább.

Szélességi bejárás

BFS: Breadth First Search
Meglátogatjuk az első csúcsot, majd ennek a csúcsnak az összes szomszédját. Aztán ezen szomszédok összes olyan szomszédját, hol még nem jártunk, és így tovább.
megvalósítás $\Longrightarrow$ *sor* (FIFO-lista)

Szélességi bejárás

BFS: Breadth First Search
Meglátogatjuk az első csúcsot, majd ennek a csúcsnak az összes szomszédját. Aztán ezen szomszédok összes olyan szomszédját, hol még nem jártunk, és így tovább.
megvalósítás $\Longrightarrow$ *sor* (FIFO-lista)
Berakjuk be az épp meglátogatott csúcsot, hogy majd a megfelelő időben a szomszédaira is sort keríthessünk.

Szélességi bejárás

BFS: Breadth First Search
Meglátogatjuk az első csúcsot, majd ennek a csúcsnak az összes szomszédját. Aztán ezen szomszédok összes olyan szomszédját, hol még nem jártunk, és így tovább.
megvalósítás $\Longrightarrow$ *sor* (FIFO-lista)
Berakjuk be az épp meglátogatott csúcsot, hogy majd a megfelelő időben a szomszédaira is sort keríthessünk.
Általános lépés $\Longrightarrow$ vesszük a sor elején levő x csúcsot, töröljük a sorból, meglátogatjuk azokat az y szomszédait, amelyeket eddig még nem láttunk, majd ezeket az y csúcsokat a sor végére tesszük.

Szélességi bejárás

BFS: Breadth First Search
Meglátogatjuk az első csúcsot, majd ennek a csúcsnak az összes szomszédját. Aztán ezen szomszédok összes olyan szomszédját, hol még nem jártunk, és így tovább.
megvalósítás $\Longrightarrow$ *sor* (FIFO-lista)
Berakjuk be az épp meglátogatott csúcsot, hogy majd a megfelelő időben a szomszédaira is sort keríthessünk.
Általános lépés $\Longrightarrow$ vesszük a sor elején levő x csúcsot, töröljük a sorból, meglátogatjuk azokat az y szomszédait, amelyeket eddig még nem láttunk, majd ezeket az y csúcsokat a sor végére tesszük.

```
procedure bejár (* elvégzi a G irányított gráf szélességi bejárását *)
    begin
        for v := 1 to n do
            bejárva[v] := hamis;
        for v := 1 to n do
            if bejárva[v] = hamis then
              szb(v)
    end
```

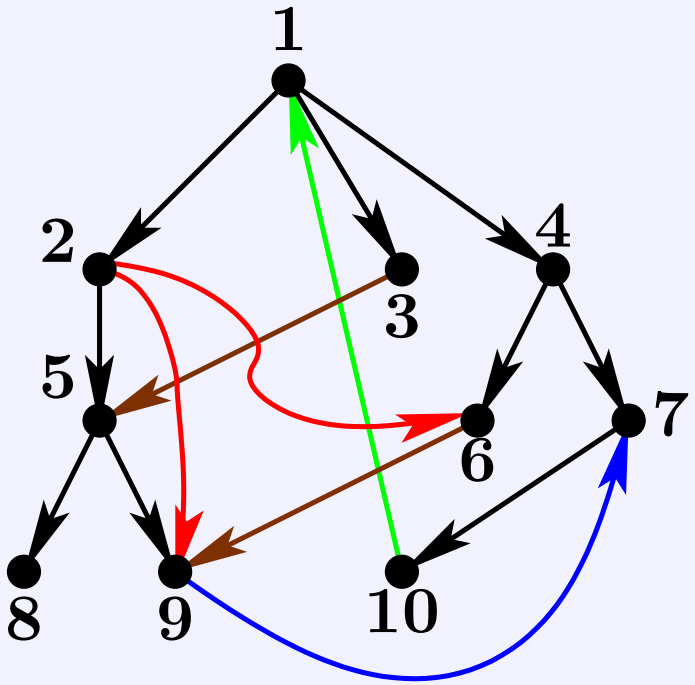
```
procedure szb (v: csúcs)
    var
        Q: csúcsokból álló sor;
        x, y: csúcsok;
    begin
        bejárva[v] := igaz;
        sorba(v, Q);
        while Q nem üres do begin
            x := első(Q);
            for minden x → y ∈ E élre do
                if bejárva[y] = hamis then begin
                    bejárva[y] := igaz;
(*)                 sorba(y, Q)
                end
        end
    end
```

```
procedure szb (v: csúcs)
    var
        Q: csúcsokból álló sor;
        x, y: csúcsok;
    begin
        bejárva[v] := igaz;
        sorba(v, Q);
        while Q nem üres do begin
            x := első(Q);
            for minden x → y ∈ E élre do
                if bejárva[y] = hamis then begin
                    bejárva[y] := igaz;
(*)                 sorba(y, Q)
                end
        end
    end
```

JAVA animáció: BFS

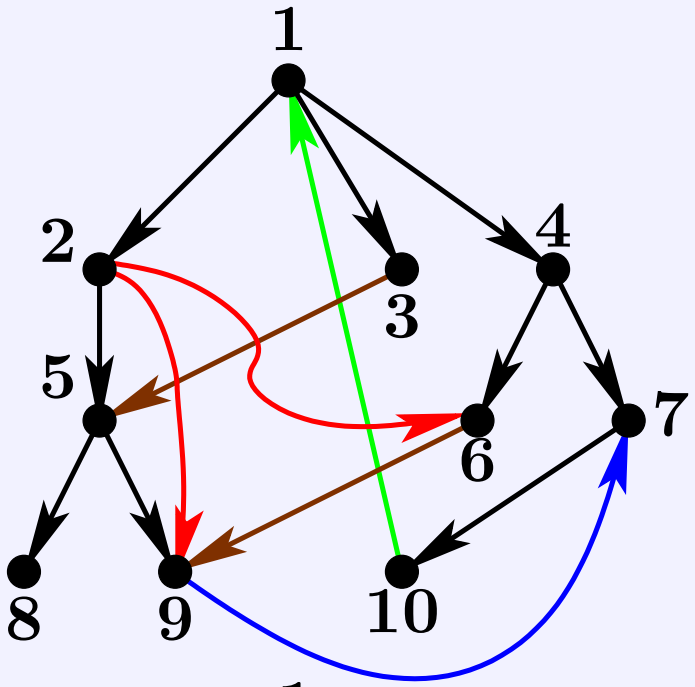
Faél:
megvizsgálásukkor még be nem járt pontba mutatnak

Faél:
megvizsgálásukkor még be nem járt pontba mutatnak



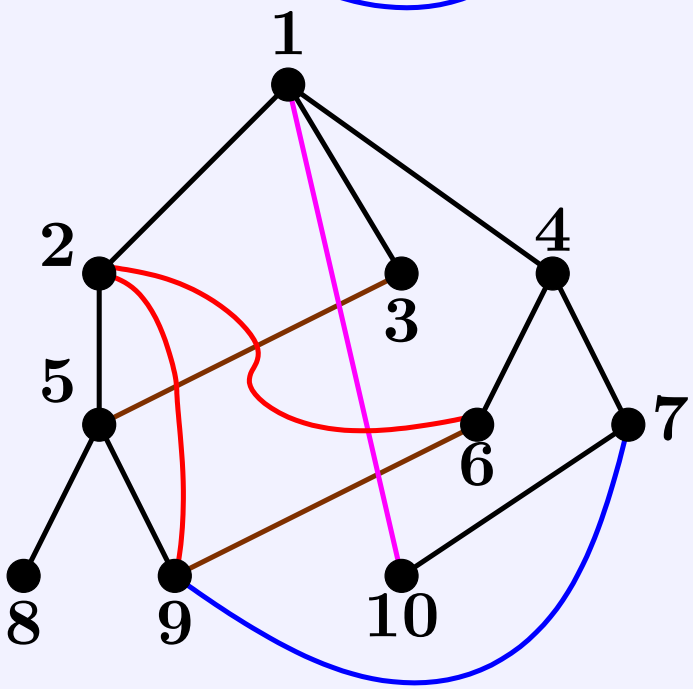
faél
ilyen nincs
visszaél
keresztél
keresztél

Faél:
megvizsgálásukkor még be nem járt pontba mutatnak



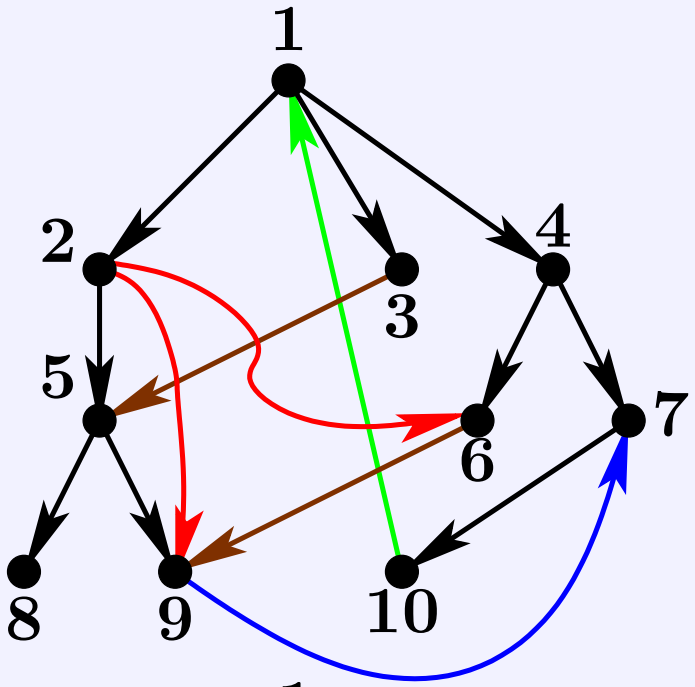
faél
ilyen nincs
visszaél
keresztél
keresztél

Irányítatlan esetben csak faél és keresztél lehet.



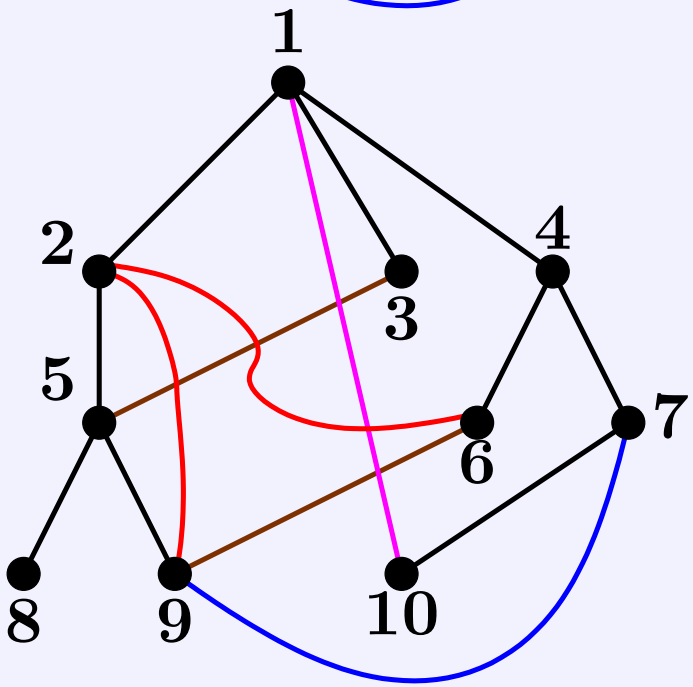
faél
ilyen nincs
ilyen nincs
keresztél
keresztél

Faél:
megvizsgálásukkor még be nem járt pontba mutatnak



faél
ilyen nincs
visszaél
keresztél
keresztél

Irányítatlan esetben csak faél és keresztél lehet.
Lépésszám: $O(n + e)$



faél
ilyen nincs
ilyen nincs
keresztél
keresztél

Legrövidebb utak súlyozatlan gráfokban

Ha minden él hossza egy $\Longrightarrow$ út hossza = élek száma

Legrövidebb utak súlyozatlan gráfokban

Ha minden él hossza egy $\Longrightarrow$ út hossza = élek száma
Szélességi kereséssel $\Longrightarrow$ Jelentse $D[v]$ a v csúcsnak az s-től való távolságát az s-gyökerű szélességi fában.

Legrövidebb utak súlyozatlan gráfokban

Ha minden él hossza egy $\Longrightarrow$ út hossza = élek száma
Szélességi kereséssel $\Longrightarrow$ Jelentse $D[v]$ a v csúcsnak az s-től való távolságát az s-gyökerű szélességi fában.

Legyen kezdetben $D[s] := 0$

Legrövidebb utak súlyozatlan gráfokban

Ha minden él hossza egy $\Longrightarrow$ út hossza = élek száma
Szélességi kereséssel $\Longrightarrow$ Jelentse $D[v]$ a v csúcsnak az s-től való távolságát az s-gyökerű szélességi fában.

Legyen kezdetben $D[s] := 0$
az *szb* eljárásba tegyük be a $D[y] := D[x] + 1;$ utasítást, miután elértük y-t.

Legrövidebb utak súlyozatlan gráfokban

Ha minden él hossza egy $\Longrightarrow$ út hossza = élek száma
Szélességi kereséssel $\Longrightarrow$ Jelentse $D[v]$ a v csúcsnak az s-től való távolságát az s-gyökerű szélességi fában.

Legyen kezdetben $D[s] := 0$
az *szb* eljárásba tegyük be a $D[y] := D[x] + 1;$ utasítást, miután elértük y-t.

Lépésszám: $O(n + e)$
Tétel. *Az előzőek szerint módosított szélességi bejárás végeztével teljesülnek a következők:*
1. Legyen $s = x_1, x_2, \ldots, x_n$ a csúcsoknak a szélességi bejárás szerinti sorrendje. Ekkor $D[x_1] \leq D[x_2] \leq \ldots \leq D[x_n]$.

Legrövidebb utak súlyozatlan gráfokban

Ha minden él hossza egy $\Longrightarrow$ út hossza = élek száma
Szélességi kereséssel $\Longrightarrow$ Jelentse $D[v]$ a v csúcsnak az s-től való távolságát az s-gyökerű szélességi fában.

Legyen kezdetben $D[s] := 0$
az *szb* eljárásba tegyük be a $D[y] := D[x] + 1;$ utasítást, miután elértük y-t.

Lépésszám: $O(n + e)$
Tétel. *Az előzőek szerint módosított szélességi bejárás végeztével teljesülnek a következők:*
1. Legyen $s = x_1, x_2, \ldots, x_n$ a csúcsoknak a szélességi bejárás szerinti sorrendje. Ekkor $D[x_1] \leq D[x_2] \leq \ldots \leq D[x_n]$.
2. Ha $x \to y$ éle G-nek, akkor $D[y] \leq D[x] + 1$.

Legrövidebb utak súlyozatlan gráfokban

Ha minden él hossza egy $\Longrightarrow$ út hossza = élek száma
Szélességi kereséssel $\Longrightarrow$ Jelentse $D[v]$ a v csúcsnak az s-től való távolságát az s-gyökerű szélességi fában.

Legyen kezdetben $D[s] := 0$
az *szb* eljárásba tegyük be a $D[y] := D[x] + 1;$ utasítást, miután elértük y-t.

Lépésszám: $O(n + e)$
Tétel. *Az előzőek szerint módosított szélességi bejárás végeztével teljesülnek a következők:*
1. Legyen $s = x_1, x_2, \ldots, x_n$ a csúcsoknak a szélességi bejárás szerinti sorrendje. Ekkor $D[x_1] \leq D[x_2] \leq \ldots \leq D[x_n]$.
2. Ha $x \to y$ éle G-nek, akkor $D[y] \leq D[x] + 1$.
3. $D[v] = d(s, v)$ teljesül minden $v \in V$ csúcsra.

Tétel. *1.* $D[x_1] \leq D[x_2] \leq \ldots \leq D[x_n]$.

Tétel. *1.* $D[x_1] \leq D[x_2] \leq \ldots \leq D[x_n]$.

Bizonyítás: A csúcsok az $s = x_1, x_2, \ldots, x_n$ sorrendben kerülnek bele a Q sorba $\Longrightarrow$ ebben a sorrendben is kerülnek ki a sorból

Tétel. *1.* $D[x_1] \leq D[x_2] \leq \ldots \leq D[x_n]$.

Bizonyítás: A csúcsok az $s = x_1, x_2, \ldots, x_n$ sorrendben kerülnek bele a Q sorba $\Longrightarrow$ ebben a sorrendben is kerülnek ki a sorból
ha $x \neq s$ csúcs előbb van mint $y \Longrightarrow apa(x)$ nem lehet később, mint $apa(y)$, hiszen ha előbb lenne, y-hoz előbb eljutottunk volna

Tétel. *1.* $D[x_1] \leq D[x_2] \leq \ldots \leq D[x_n]$.

Bizonyítás: A csúcsok az $s = x_1, x_2, \ldots, x_n$ sorrendben kerülnek bele a Q sorba $\Longrightarrow$ ebben a sorrendben is kerülnek ki a sorból
ha $x \neq s$ csúcs előbb van mint $y \Longrightarrow apa(x)$ nem lehet később, mint $apa(y)$, hiszen ha előbb lenne, y-hoz előbb eljutottunk volna
Indukció $\Longrightarrow$ Gyökérre $D[s] = 0$, fiaira mind nagyobb $\surd$

Tétel. *1.* $D[x_1] \leq D[x_2] \leq \ldots \leq D[x_n]$.

Bizonyítás: A csúcsok az $s = x_1, x_2, \ldots, x_n$ sorrendben kerülnek bele a Q sorba $\implies$ ebben a sorrendben is kerülnek ki a sorból
ha $x \neq s$ csúcs előbb van mint $y \implies apa(x)$ nem lehet később, mint $apa(y)$, hiszen ha előbb lenne, y-hoz előbb eljutottunk volna
Indukció $\implies$ Gyökérre $D[s] = 0$, fiaira mind nagyobb ✓
$D[x_i] = D[apa(x_i)] + 1$ és $D[x_{i+1}] = D[apa(x_{i+1})] + 1 \implies$

Tétel. *1.* $D[x_1] \leq D[x_2] \leq \ldots \leq D[x_n]$.

Bizonyítás: A csúcsok az $s = x_1, x_2, \ldots, x_n$ sorrendben kerülnek bele a Q sorba $\implies$ ebben a sorrendben is kerülnek ki a sorból
ha $x \neq s$ csúcs előbb van mint $y \implies apa(x)$ nem lehet később, mint $apa(y)$, hiszen ha előbb lenne, y-hoz előbb eljutottunk volna
Indukció $\implies$ Gyökérre $D[s] = 0$, fiaira mind nagyobb ✓
$D[x_i] = D[apa(x_i)] + 1$ és $D[x_{i+1}] = D[apa(x_{i+1})] + 1 \implies$
Ha a két apa különböző
$\implies D[apa(x_i)] \leq D[apa(x_{i+1})] \implies D[x_i] \leq D[x_{i+1}]$

Tétel. *1.* $D[x_1] \leq D[x_2] \leq \ldots \leq D[x_n]$.

Bizonyítás: A csúcsok az $s = x_1, x_2, \ldots, x_n$ sorrendben kerülnek bele a Q sorba $\implies$ ebben a sorrendben is kerülnek ki a sorból
ha $x \neq s$ csúcs előbb van mint $y \implies apa(x)$ nem lehet később, mint $apa(y)$, hiszen ha előbb lenne, y-hoz előbb eljutottunk volna
Indukció $\implies$ Gyökérre $D[s] = 0$, fiaira mind nagyobb ✓
$D[x_i] = D[apa(x_i)] + 1$ és $D[x_{i+1}] = D[apa(x_{i+1})] + 1 \implies$
Ha a két apa különböző
$\implies D[apa(x_i)] \leq D[apa(x_{i+1})] \implies D[x_i] \leq D[x_{i+1}]$
Ha pedig az apák megegyeznek $\implies D[x_i] = D[x_{i+1}]$

Tétel. *1.* $D[x_1] \leq D[x_2] \leq \ldots \leq D[x_n]$.

Bizonyítás: A csúcsok az $s = x_1, x_2, \ldots, x_n$ sorrendben kerülnek bele a Q sorba $\implies$ ebben a sorrendben is kerülnek ki a sorból
ha $x \neq s$ csúcs előbb van mint $y \implies apa(x)$ nem lehet később, mint $apa(y)$, hiszen ha előbb lenne, y-hoz előbb eljutottunk volna
Indukció $\implies$ Gyökérre $D[s] = 0$, fiaira mind nagyobb ✓
$D[x_i] = D[apa(x_i)] + 1$ és $D[x_{i+1}] = D[apa(x_{i+1})] + 1 \implies$
Ha a két apa különböző
$\implies D[apa(x_i)] \leq D[apa(x_{i+1})] \implies D[x_i] \leq D[x_{i+1}]$
Ha pedig az apák megegyeznek $\implies D[x_i] = D[x_{i+1}]$

Tétel. *2. Ha* $x \to y$ *éle* G*-nek, akkor* $D[y] \leq D[x] + 1$

Bizonyítás: Nézzük, hogy mi történik, amikor x kikerül a Q sorból, és éppen az (x, y) élet vizsgáljuk.

Tétel. *1.* $D[x_1] \leq D[x_2] \leq \ldots \leq D[x_n]$.

Bizonyítás: A csúcsok az $s = x_1, x_2, \ldots, x_n$ sorrendben kerülnek bele a Q sorba $\implies$ ebben a sorrendben is kerülnek ki a sorból
ha $x \neq s$ csúcs előbb van mint $y \implies apa(x)$ nem lehet később, mint $apa(y)$, hiszen ha előbb lenne, y-hoz előbb eljutottunk volna
Indukció $\implies$ Gyökérre $D[s] = 0$, fiaira mind nagyobb ✓
$D[x_i] = D[apa(x_i)] + 1$ és $D[x_{i+1}] = D[apa(x_{i+1})] + 1 \implies$
Ha a két apa különböző
$\implies D[apa(x_i)] \leq D[apa(x_{i+1})] \implies D[x_i] \leq D[x_{i+1}]$
Ha pedig az apák megegyeznek $\implies D[x_i] = D[x_{i+1}]$

Tétel. *2. Ha* $x \to y$ *éle* G*-nek, akkor* $D[y] \leq D[x] + 1$

Bizonyítás: Nézzük, hogy mi történik, amikor x kikerül a Q sorból, és éppen az (x, y) élet vizsgáljuk.
Ha bejárva$[y] = hamis \implies y$ apja x, vagyis $D[y] = D[x] + 1$

Tétel. *1.* $D[x_1] \leq D[x_2] \leq \ldots \leq D[x_n]$.

Bizonyítás: A csúcsok az $s = x_1, x_2, \ldots, x_n$ sorrendben kerülnek bele a Q sorba $\implies$ ebben a sorrendben is kerülnek ki a sorból
ha $x \neq s$ csúcs előbb van mint $y \implies apa(x)$ nem lehet később, mint $apa(y)$, hiszen ha előbb lenne, y-hoz előbb eljutottunk volna
Indukció $\implies$ Gyökérre $D[s] = 0$, fiaira mind nagyobb ✓
$D[x_i] = D[apa(x_i)] + 1$ és $D[x_{i+1}] = D[apa(x_{i+1})] + 1 \implies$
Ha a két apa különböző
$\implies D[apa(x_i)] \leq D[apa(x_{i+1})] \implies D[x_i] \leq D[x_{i+1}]$
Ha pedig az apák megegyeznek $\implies D[x_i] = D[x_{i+1}]$

Tétel. *2. Ha* $x \to y$ *éle* G*-nek, akkor* $D[y] \leq D[x] + 1$

Bizonyítás: Nézzük, hogy mi történik, amikor x kikerül a Q sorból, és éppen az (x, y) élet vizsgáljuk.
Ha bejárva$[y] = hamis \implies y$ apja x, vagyis $D[y] = D[x] + 1$
Különben y-t már korábban láttuk $\implies y$ apja előbb van mint x

Tétel. *1.* $D[x_1] \leq D[x_2] \leq \ldots \leq D[x_n]$.

Bizonyítás: A csúcsok az $s = x_1, x_2, \ldots, x_n$ sorrendben kerülnek bele a Q sorba $\implies$ ebben a sorrendben is kerülnek ki a sorból
ha $x \neq s$ csúcs előbb van mint $y \implies apa(x)$ nem lehet később, mint $apa(y)$, hiszen ha előbb lenne, y-hoz előbb eljutottunk volna
Indukció $\implies$ Gyökérre $D[s] = 0$, fiaira mind nagyobb ✓
$D[x_i] = D[apa(x_i)] + 1$ és $D[x_{i+1}] = D[apa(x_{i+1})] + 1 \implies$
Ha a két apa különböző
$\implies D[apa(x_i)] \leq D[apa(x_{i+1})] \implies D[x_i] \leq D[x_{i+1}]$
Ha pedig az apák megegyeznek $\implies D[x_i] = D[x_{i+1}]$

Tétel. *2. Ha* $x \to y$ *éle* G*-nek, akkor* $D[y] \leq D[x] + 1$

Bizonyítás: Nézzük, hogy mi történik, amikor x kikerül a Q sorból, és éppen az (x, y) élet vizsgáljuk.
Ha bejárva$[y] = hamis \implies y$ apja x, vagyis $D[y] = D[x] + 1$
Különben y-t már korábban láttuk $\implies y$ apja előbb van mint x
$\implies D[apa(y)] \leq D[x]$

Tétel. *1.* $D[x_1] \leq D[x_2] \leq \ldots \leq D[x_n]$.

Bizonyítás: A csúcsok az $s = x_1, x_2, \ldots, x_n$ sorrendben kerülnek bele a Q sorba $\implies$ ebben a sorrendben is kerülnek ki a sorból
ha $x \neq s$ csúcs előbb van mint $y \implies apa(x)$ nem lehet később, mint $apa(y)$, hiszen ha előbb lenne, y-hoz előbb eljutottunk volna
Indukció $\implies$ Gyökérre $D[s] = 0$, fiaira mind nagyobb ✓
$D[x_i] = D[apa(x_i)] + 1$ és $D[x_{i+1}] = D[apa(x_{i+1})] + 1 \implies$
Ha a két apa különböző
$\implies D[apa(x_i)] \leq D[apa(x_{i+1})] \implies D[x_i] \leq D[x_{i+1}]$
Ha pedig az apák megegyeznek $\implies D[x_i] = D[x_{i+1}]$

Tétel. *2. Ha* $x \to y$ *éle* G*-nek, akkor* $D[y] \leq D[x] + 1$

Bizonyítás: Nézzük, hogy mi történik, amikor x kikerül a Q sorból, és éppen az (x, y) élet vizsgáljuk.
Ha bejárva$[y] = hamis \implies y$ apja x, vagyis $D[y] = D[x] + 1$
Különben y-t már korábban láttuk $\implies y$ apja előbb van mint x
$\implies D[apa(y)] \leq D[x] \implies D[y] = D[apa(y)] + 1 \leq D[x] + 1$

Tétel. *3.* $D[v] = d(s, v)$ *teljesül minden* $v \in V$ *csúcsra.*

Bizonyítás: $d(s, v) \leq D[v]$ $\surd$

Tétel. *3.* $D[v] = d(s, v)$ *teljesül minden* $v \in V$ *csúcsra.*

Bizonyítás: $d(s, v) \leq D[v]$ $\surd$

Legyen $s = y_0, y_1, \ldots y_k = v$ egy minimális hosszúságú G-beli irányított út s-ből v-be.

Tétel. *3.* $D[v] = d(s, v)$ *teljesül minden* $v \in V$ *csúcsra.*

Bizonyítás: $d(s, v) \leq D[v]$ $\surd$

Legyen $s = y_0, y_1, \ldots y_k = v$ egy minimális hosszúságú G-beli irányított út s-ből v-be.

$\Longrightarrow$ az út éleire: $D[y_1] \leq D[s] + 1 = 1$, majd

$D[y_2] \leq D[y_1] + 1 \leq 2$

Tétel. *3.* $D[v] = d(s, v)$ *teljesül minden* $v \in V$ *csúcsra.*

Bizonyítás: $d(s, v) \leq D[v]$ $\surd$

Legyen $s = y_0, y_1, \ldots y_k = v$ egy minimális hosszúságú G-beli irányított út s-ből v-be.

$\Longrightarrow$ az út éleire: $D[y_1] \leq D[s] + 1 = 1$, majd

$D[y_2] \leq D[y_1] + 1 \leq 2 \quad \ldots \quad D[v] = D[y_k] \leq k = d(s, v) \implies \surd$

Tétel. *3.* $D[v] = d(s, v)$ *teljesül minden* $v \in V$ *csúcsra.*

Bizonyítás: $d(s, v) \leq D[v]$ $\surd$

Legyen $s = y_0, y_1, \ldots y_k = v$ egy minimális hosszúságú G-beli irányított út s-ből v-be.

$\Longrightarrow$ az út éleire: $D[y_1] \leq D[s] + 1 = 1$, majd

$D[y_2] \leq D[y_1] + 1 \leq 2 \quad \ldots \quad D[v] = D[y_k] \leq k = d(s, v) \implies \surd$

JAVA animáció: Legrövidebb út

Minimális költségű feszítőfák

Most irányítatlan gráfokkal foglalkozunk
kör és út $\Longrightarrow$ valóban egyszerű

Minimális költségű feszítőfák

Most irányítatlan gráfokkal foglalkozunk
kör és út $\Longrightarrow$ valóban egyszerű

Definíció. *(minimális költségű feszítőfa)* *Legyen $G = (V, E)$ egy összefüggő gráf. A G gráf egy körmentes összefüggő $F = (V, E')$ részgráfja a gráf egy* feszítőfája*. Legyen továbbá az éleken értelmezve egy $c : E \to \mathbb{R}$ súlyfüggvény. Ekkor a G gráf egy F feszítőfája* minimális költségű*, ha költsége (a benne szereplő élek súlyainak összege) minimális G összes feszítőfája közül.*

Minimális költségű feszítőfák

Most irányítatlan gráfokkal foglalkozunk
kör és út $\Longrightarrow$ valóban egyszerű

Definíció. *(minimális költségű feszítőfa)* *Legyen $G = (V, E)$ egy összefüggő gráf. A G gráf egy körmentes összefüggő $F = (V, E')$ részgráfja a gráf egy feszítőfája. Legyen továbbá az éleken értelmezve egy $c : E \to \mathbb{R}$ súlyfüggvény. Ekkor a G gráf egy F feszítőfája minimális költségű, ha költsége (a benne szereplő élek súlyainak összege) minimális G összes feszítőfája közül.*

Probléma:
Adott egy $G = (V, E)$ összefüggő irányítatlan gráf, és az élein értelmezett $c : E \to \mathbb{R}$ súlyfüggvény. Határozzuk meg a G egy minimális költségű feszítőfáját.

Minimális költségű feszítőfák

Most irányítatlan gráfokkal foglalkozunk
kör és út $\Longrightarrow$ valóban egyszerű

Definíció. *(minimális költségű feszítőfa)* *Legyen $G = (V, E)$ egy összefüggő gráf. A G gráf egy körmentes összefüggő $F = (V, E')$ részgráfja a gráf egy* feszítőfája*. Legyen továbbá az éleken értelmezve egy $c : E \to \mathbb{R}$ súlyfüggvény. Ekkor a G gráf egy F feszítőfája* minimális költségű*, ha költsége (a benne szereplő élek súlyainak összege) minimális G összes feszítőfája közül.*

Probléma:
Adott egy $G = (V, E)$ összefüggő irányítatlan gráf, és az élein értelmezett $c : E \to \mathbb{R}$ súlyfüggvény. Határozzuk meg a G egy minimális költségű feszítőfáját.

Például: Villamosvezetékek kiépítése

Fák tulajdonságai

Tétel.

1. Minden legalább kétpontú fában van olyan csúcs, amiből csak egy él megy ki (elsőfokú csúcs).

Fák tulajdonságai

Tétel.

1. Minden legalább kétpontú fában van olyan csúcs, amiből csak egy él megy ki (elsőfokú csúcs).

2. Bármely összefüggő gráf tartalmaz feszítőfát.

Fák tulajdonságai

Tétel.

1. Minden legalább kétpontú fában van olyan csúcs, amiből csak egy él megy ki (elsőfokú csúcs).

2. Bármely összefüggő gráf tartalmaz feszítőfát.

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

Fák tulajdonságai

Tétel.

1. *Minden legalább kétpontú fában van olyan csúcs, amiből csak egy él megy ki (elsőfokú csúcs).*
2. *Bármely összefüggő gráf tartalmaz feszítőfát.*
3. *Egy n-pontú összefüggő gráf akkor és csak akkor fa, ha $n - 1$ éle van.*
4. *Egy fa bármely két pontja között pontosan egy út vezet.*

Fák tulajdonságai

Tétel.

1. Minden legalább kétpontú fában van olyan csúcs, amiből csak egy él megy ki (elsőfokú csúcs).

2. Bármely összefüggő gráf tartalmaz feszítőfát.

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

4. Egy fa bármely két pontja között pontosan egy út vezet.

5. Legyen G egy súlyozott élű összefüggő gráf, F egy minimális költségű feszítőfája. Legyen $g=(u,v)$ a G-nek egy olyan éle, ami nem éle F-nek, és tegyük fel, hogy az F-beli u-ból v-be vezető úton van olyan g' él, amelyre $c(g) \leq c(g')$. Ekkor az F-ből a g hozzávételével és a g' elhagyásával adódó F' gráf is egy minimális költségű feszítőfa G-ben.

Fák tulajdonságai

Tétel.

1. Minden legalább kétpontú fában van olyan csúcs, amiből csak egy él megy ki (elsőfokú csúcs).

2. Bármely összefüggő gráf tartalmaz feszítőfát.

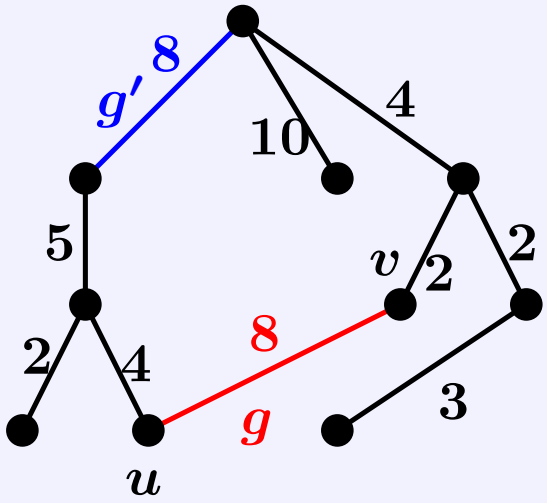
3. Egy n-pontú összefüggő gráf akkor és csak akkor fa, ha $n-1$ éle van.

4. Egy fa bármely két pontja között pontosan egy út vezet.

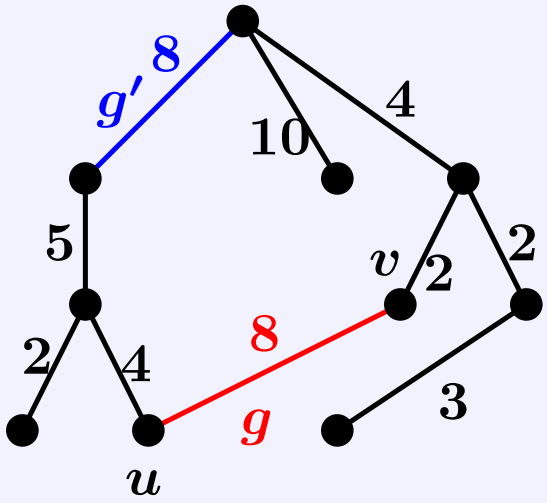
5. Legyen G egy súlyozott élű összefüggő gráf, F egy minimális költségű feszítőfája. Legyen $g=(u,v)$ a G-nek egy olyan éle, ami nem éle F-nek, és tegyük fel, hogy az F-beli u-ból v-be vezető úton van olyan g' él, amelyre $c(g) \leq c(g')$. Ekkor az F-ből a g hozzávételével és a g' elhagyásával adódó F' gráf is egy minimális költségű feszítőfa G-ben.

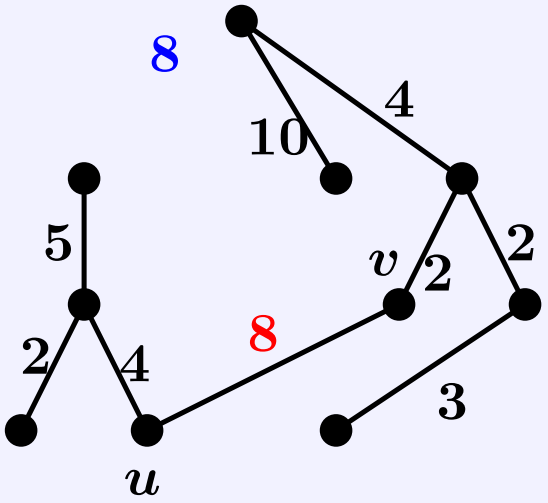
Bizonyítás: 1–4 volt már BSZ-ben $\surd$

Bizonyítás: 5.

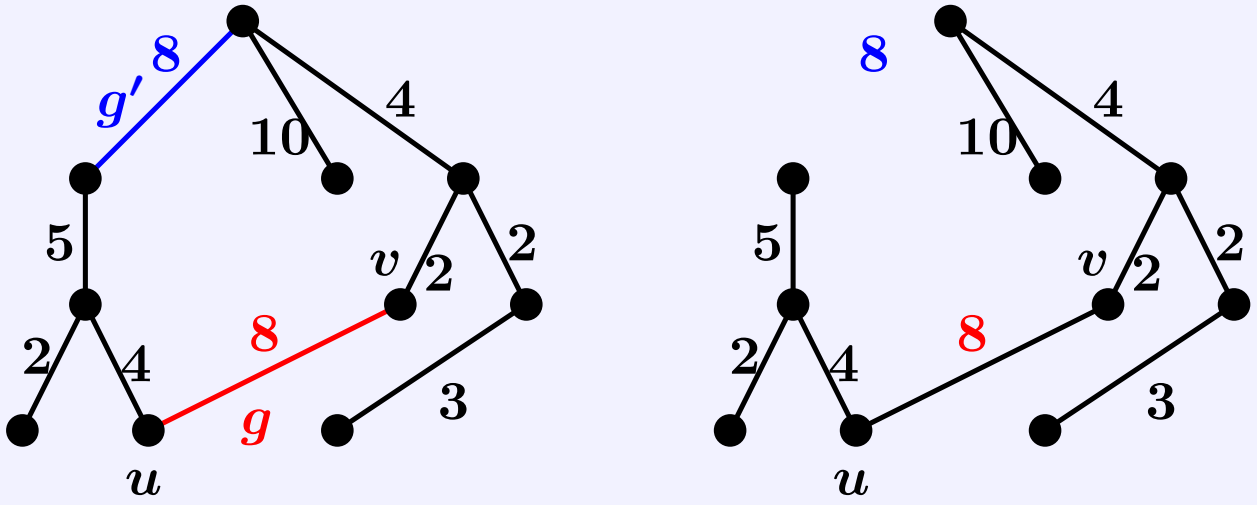


Bizonyítás: 5.



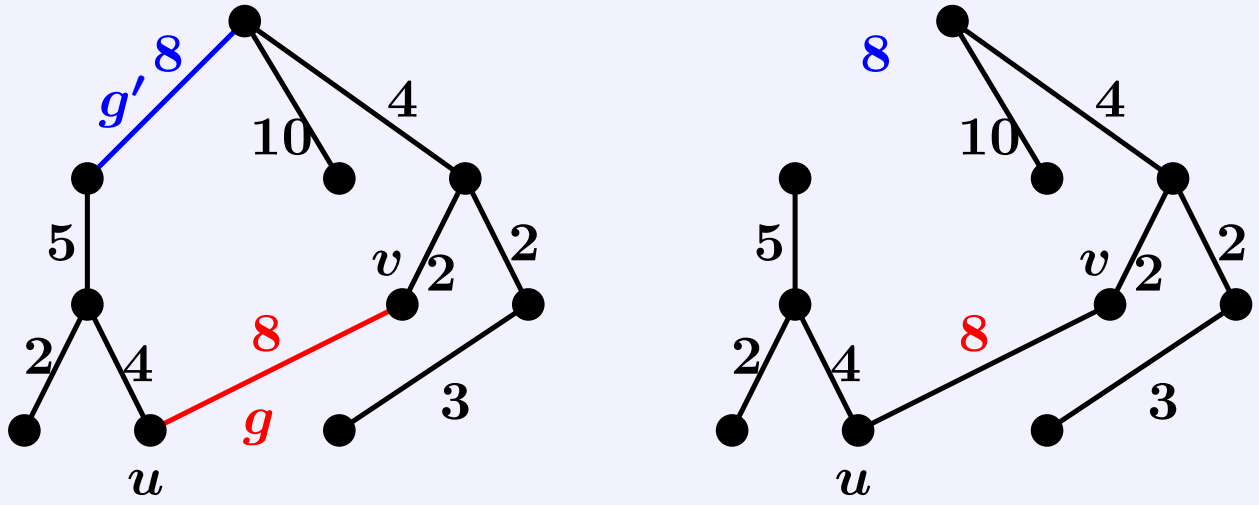


Bizonyítás: 5.



$F \cup \{g\}$ gráfban van olyan kör, amelynek g' éle $\Longrightarrow$ A g' törlésével kapott F' gráf összefüggő marad

Bizonyítás: 5.



$F \cup \{g\}$ gráfban van olyan kör, amelynek g' éle $\Longrightarrow$ A g' törlésével kapott F' gráf összefüggő marad

F' költsége nem nagyobb F költségénél

A piros-kék algoritmus

Sorra nézzük G éleit: bizonyosakat beveszünk a minimális feszítőfába, másokat pedig eldobunk.

A piros-kék algoritmus

Sorra nézzük G éleit: bizonyosakat beveszünk a minimális feszítőfába, másokat pedig eldobunk.
$\Longrightarrow$ színezzük a G éleit:
a *kék* élek belekerülnek a végeredményt jelentő minimális feszítőfába,

A piros-kék algoritmus

Sorra nézzük G éleit: bizonyosakat beveszünk a minimális feszítőfába, másokat pedig eldobunk.
$\Longrightarrow$ színezzük a G éleit:
a *kék* élek belekerülnek a végeredményt jelentő minimális feszítőfába,
a *pirosak* pedig nem

A piros-kék algoritmus

Sorra nézzük G éleit: bizonyosakat beveszünk a minimális feszítőfába, másokat pedig eldobunk.
$\Longrightarrow$ színezzük a G éleit:
a *kék* élek belekerülnek a végeredményt jelentő minimális feszítőfába,
a *pirosak* pedig nem
$\Longrightarrow$ Úgy színezünk, hogy az eddig kialakult (részleges) színezés mindig *takaros* legyen.

A piros-kék algoritmus

Sorra nézzük G éleit: bizonyosakat beveszünk a minimális feszítőfába, másokat pedig eldobunk.
$\Longrightarrow$ színezzük a G éleit:
a *kék* élek belekerülnek a végeredményt jelentő minimális feszítőfába,
a *pirosak* pedig nem
$\Longrightarrow$ Úgy színezünk, hogy az eddig kialakult (részleges) színezés mindig *takaros* legyen.

Definíció. (takaros színezés) *Tekintsük a súlyozott élű G gráf éleinek egy részleges színezését, melynél bármely él piros, kék vagy színtelen lehet. Ez a színezés takaros, ha van G-nek olyan minimális költségű feszítőfája, ami az összes kék élet tartalmazza, és egyetlen piros élet sem tartalmaz.*

KÉK SZABÁLY: Válasszunk ki egy olyan $\emptyset \neq X \subset V$ csúcshalmazt, amiből nem vezet ki kék él. Ezután egy legkisebb súlyú X-ből kimenő színezetlen élet fessünk kékre.

KÉK SZABÁLY: Válasszunk ki egy olyan $\emptyset \neq X \subset V$ csúcshalmazt, amiből nem vezet ki kék él. Ezután egy legkisebb súlyú X-ből kimenő színezetlen élet fessünk kékre.

PIROS SZABÁLY: Válasszunk G-ben egy olyan egyszerű kört, amiben nincs piros él. A kör egyik legnagyobb súlyú színtelen élét fessük pirosra.

Kezdetben G-nek nincs színes éle.

KÉK SZABÁLY: Válasszunk ki egy olyan $\emptyset \neq X \subset V$ csúcshalmazt, amiből nem vezet ki kék él. Ezután egy legkisebb súlyú X-ből kimenő színezetlen élet fessünk kékre.

PIROS SZABÁLY: Válasszunk G-ben egy olyan egyszerű kört, amiben nincs piros él. A kör egyik legnagyobb súlyú színtelen élét fessük pirosra.

Kezdetben G-nek nincs színes éle. $\Longrightarrow$ a két szabályt tetszőleges sorrendben és helyeken alkalmazzuk, amíg csak lehetséges.

KÉK SZABÁLY: Válasszunk ki egy olyan $\emptyset \neq X \subset V$ csúcshalmazt, amiből nem vezet ki kék él. Ezután egy legkisebb súlyú X-ből kimenő színezetlen élet fessünk kékre.

PIROS SZABÁLY: Válasszunk G-ben egy olyan egyszerű kört, amiben nincs piros él. A kör egyik legnagyobb súlyú színtelen élét fessük pirosra.

Kezdetben G-nek nincs színes éle. $\Longrightarrow$ a két szabályt tetszőleges sorrendben és helyeken alkalmazzuk, amíg csak lehetséges.

$\Longrightarrow$ piros-kék algoritmus

KÉK SZABÁLY: Válasszunk ki egy olyan $\emptyset \neq X \subset V$ csúcshalmazt, amiből nem vezet ki kék él. Ezután egy legkisebb súlyú X-ből kimenő színezetlen élet fessünk kékre.

PIROS SZABÁLY: Válasszunk G-ben egy olyan egyszerű kört, amiben nincs piros él. A kör egyik legnagyobb súlyú színtelen élét fessük pirosra.

Kezdetben G-nek nincs színes éle. $\Longrightarrow$ a két szabályt tetszőleges sorrendben és helyeken alkalmazzuk, amíg csak lehetséges.

$\Longrightarrow$ piros-kék algoritmus

Tétel. *A piros-kék eljárás működése során mindig takaros színezésünk van. Ezen felül a színezéssel sosem akadunk el: végül G minden éle színes lesz.*

KÉK SZABÁLY: Válasszunk ki egy olyan $\emptyset \neq X \subset V$ csúcshalmazt, amiből nem vezet ki kék él. Ezután egy legkisebb súlyú X-ből kimenő színezetlen élet fessünk kékre.

PIROS SZABÁLY: Válasszunk G-ben egy olyan egyszerű kört, amiben nincs piros él. A kör egyik legnagyobb súlyú színtelen élét fessük pirosra.

Kezdetben G-nek nincs színes éle. $\Longrightarrow$ a két szabályt tetszőleges sorrendben és helyeken alkalmazzuk, amíg csak lehetséges.

$\Longrightarrow$ piros-kék algoritmus

Tétel. *A piros-kék eljárás működése során mindig takaros színezésünk van. Ezen felül a színezéssel sosem akadunk el: végül G minden éle színes lesz.*

Bizonyítás: Belátjuk, hogy a színezés mindig takaros.

KÉK SZABÁLY: Válasszunk ki egy olyan $\emptyset \neq X \subset V$ csúcshalmazt, amiből nem vezet ki kék él. Ezután egy legkisebb súlyú X-ből kimenő színezetlen élet fessünk kékre.

PIROS SZABÁLY: Válasszunk G-ben egy olyan egyszerű kört, amiben nincs piros él. A kör egyik legnagyobb súlyú színtelen élét fessük pirosra.

Kezdetben G-nek nincs színes éle. $\Longrightarrow$ a két szabályt tetszőleges sorrendben és helyeken alkalmazzuk, amíg csak lehetséges.
$\Longrightarrow$ piros-kék algoritmus

Tétel. *A piros-kék eljárás működése során mindig takaros színezésünk van. Ezen felül a színezéssel sosem akadunk el: végül G minden éle színes lesz.*

Bizonyítás: Belátjuk, hogy a színezés mindig takaros. $\Longrightarrow$ kezdetben $\surd$

KÉK SZABÁLY: Válasszunk ki egy olyan $\emptyset \neq X \subset V$ csúcshalmazt, amiből nem vezet ki kék él. Ezután egy legkisebb súlyú X-ből kimenő színezetlen élet fessünk kékre.

PIROS SZABÁLY: Válasszunk G-ben egy olyan egyszerű kört, amiben nincs piros él. A kör egyik legnagyobb súlyú színtelen élét fessük pirosra.

Kezdetben G-nek nincs színes éle. $\Longrightarrow$ a két szabályt tetszőleges sorrendben és helyeken alkalmazzuk, amíg csak lehetséges.
$\Longrightarrow$ piros-kék algoritmus

Tétel. *A piros-kék eljárás működése során mindig takaros színezésünk van. Ezen felül a színezéssel sosem akadunk el: végül G minden éle színes lesz.*

Bizonyítás: Belátjuk, hogy a színezés mindig takaros. $\Longrightarrow$ kezdetben $\surd$
Tegyük fel, hogy egy takaros színezésünk van. Legyen F a G egy olyan minimális költségű feszítőfája, amely minden kék élet tartalmaz, és egyetlen pirosat sem. Tegyük fel továbbá, hogy ebben a helyzetben a gráf f élét festjük be.

Két eset van aszerint, hogy melyik szabályt használjuk:

A kék szabályt használjuk: $\Longrightarrow$ f kék lesz.

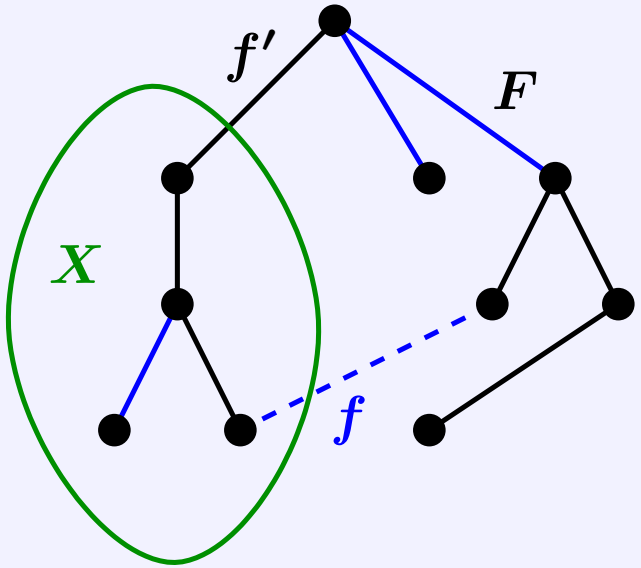
Két eset van aszerint, hogy melyik szabályt használjuk:

A kék szabályt használjuk: $\Longrightarrow$ f kék lesz.
Ha f éle F-nek ✓

Két eset van aszerint, hogy melyik szabályt használjuk:

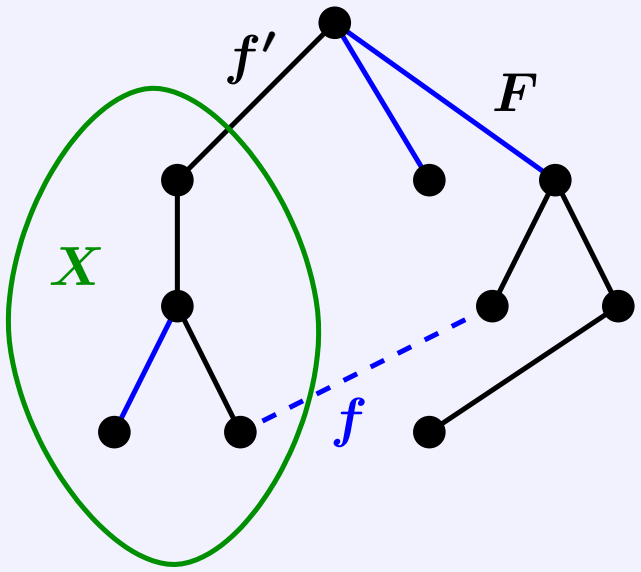
A kék szabályt használjuk: $\Longrightarrow$ f kék lesz.
Ha f éle F-nek ✓

Ha f nem éle F-nek $\Longrightarrow$ nézzük azt az $X \subset V$ csúcshalmazt, amire a kék szabályt alkalmaztuk.



Két eset van aszerint, hogy melyik szabályt használjuk:

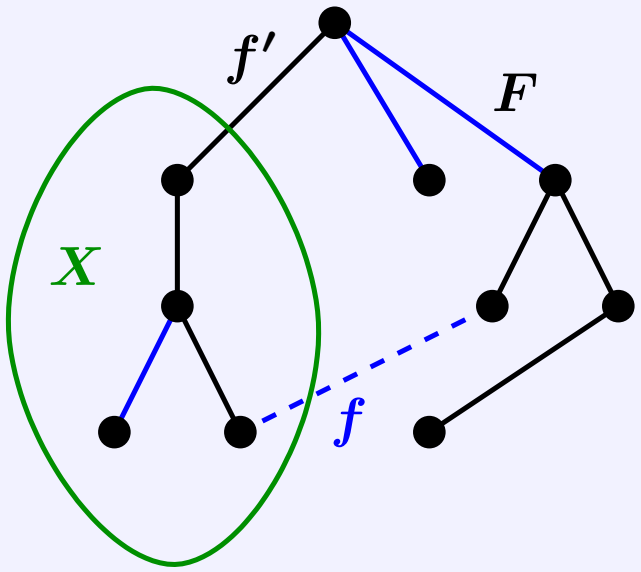
A kék szabályt használjuk: $\Longrightarrow$ f kék lesz.
Ha f éle F-nek ✓



Ha f nem éle F-nek $\Longrightarrow$ nézzük azt az $X \subset V$ csúcshalmazt, amire a kék szabályt alkalmaztuk.
Az F-ben van olyan út (mert feszítőfa), ami az f két végpontját összeköti. $\Longrightarrow$ Ezen az úton pedig van olyan f' él, ami kimegy X-ből, ugyanis f kilép X-ből.

Két eset van aszerint, hogy melyik szabályt használjuk:

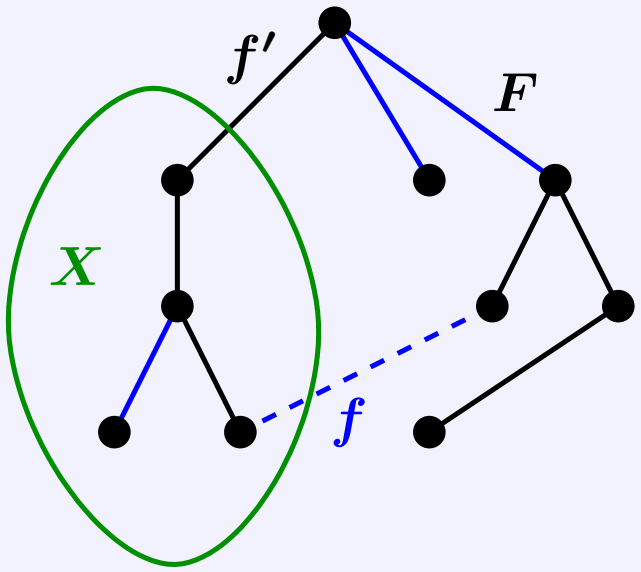
A kék szabályt használjuk: $\Longrightarrow$ f kék lesz.
Ha f éle F-nek ✓



Ha f nem éle F-nek $\Longrightarrow$ nézzük azt az $X \subset V$ csúcshalmazt, amire a kék szabályt alkalmaztuk.
Az F-ben van olyan út (mert feszítőfa), ami az f két végpontját összeköti. $\Longrightarrow$ Ezen az úton pedig van olyan f' él, ami kimegy X-ből, ugyanis f kilép X-ből.
Az F választása miatt f' nem lehet piros.

Két eset van aszerint, hogy melyik szabályt használjuk:

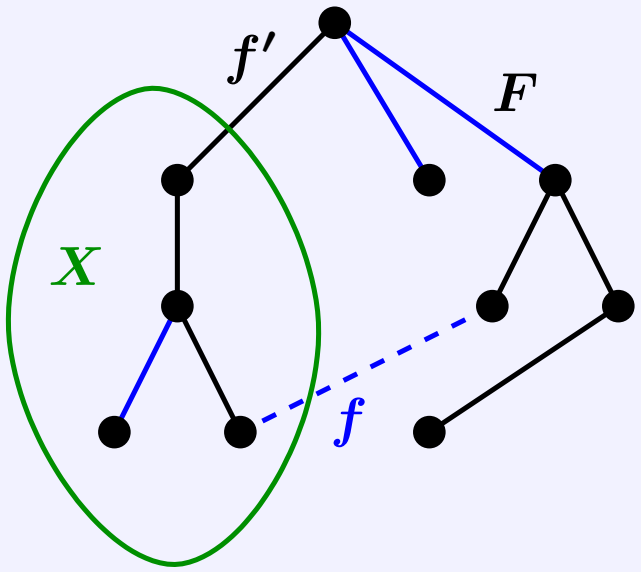
A kék szabályt használjuk: $\Longrightarrow$ f kék lesz.
Ha f éle F-nek ✓



Ha f nem éle F-nek $\Longrightarrow$ nézzük azt az $X \subset V$ csúcshalmazt, amire a kék szabályt alkalmaztuk.
Az F-ben van olyan út (mert feszítőfa), ami az f két végpontját összeköti. $\Longrightarrow$ Ezen az úton pedig van olyan f' él, ami kimegy X-ből, ugyanis f kilép X-ből.
Az F választása miatt f' nem lehet piros.
A kék szabály szerint kék sem lehet, továbbá $c(f') \geq c(f)$ is teljesül.

Két eset van aszerint, hogy melyik szabályt használjuk:

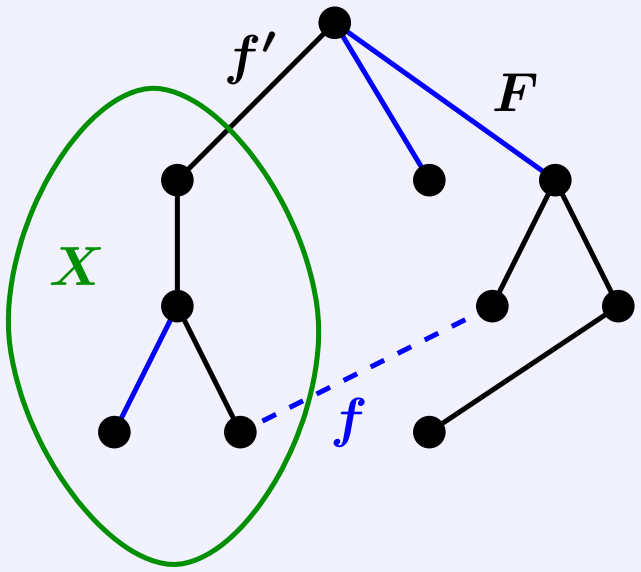
A kék szabályt használjuk: $\Longrightarrow$ f kék lesz.
Ha f éle F-nek ✓



Ha f nem éle F-nek $\Longrightarrow$ nézzük azt az $X \subset V$ csúcshalmazt, amire a kék szabályt alkalmaztuk.
Az F-ben van olyan út (mert feszítőfa), ami az f két végpontját összeköti. $\Longrightarrow$ Ezen az úton pedig van olyan f' él, ami kimegy X-ből, ugyanis f kilép X-ből.
Az F választása miatt f' nem lehet piros.
A kék szabály szerint kék sem lehet, továbbá $c(f') \geq c(f)$ is teljesül.
Legyen F' az F-ből az f' törlésével és az f hozzáadásával kapott gráf.

Két eset van aszerint, hogy melyik szabályt használjuk:

A kék szabályt használjuk: $\Longrightarrow$ f kék lesz.
Ha f éle F-nek ✓



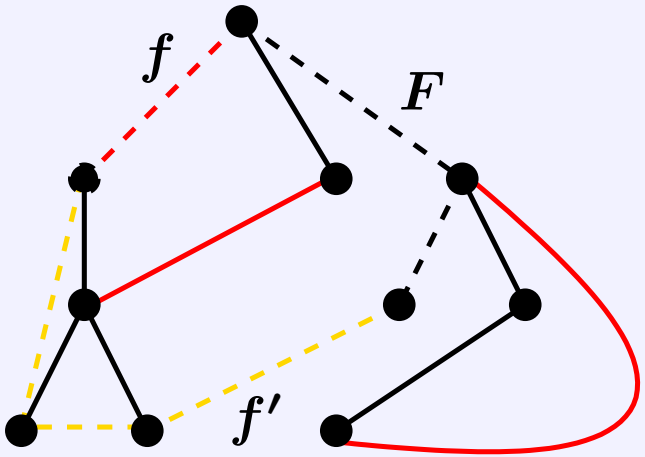
Ha f nem éle F-nek $\Longrightarrow$ nézzük azt az $X \subset V$ csúcshalmazt, amire a kék szabályt alkalmaztuk.
Az F-ben van olyan út (mert feszítőfa), ami az f két végpontját összeköti. $\Longrightarrow$ Ezen az úton pedig van olyan f' él, ami kimegy X-ből, ugyanis f kilép X-ből.
Az F választása miatt f' nem lehet piros.
A kék szabály szerint kék sem lehet, továbbá $c(f') \geq c(f)$ is teljesül.
Legyen F' az F-ből az f' törlésével és az f hozzáadásával kapott gráf.
$\Longrightarrow$ Eszerint F' egy minimális feszítőfa, tartalmaz minden kék élet és nem tartalmaz piros élet. ✓

A piros szabályt használjuk: Ekkor f piros lesz.

A piros szabályt használjuk: Ekkor f piros lesz. $\Longrightarrow$
Ha f nem éle F-nek ✓

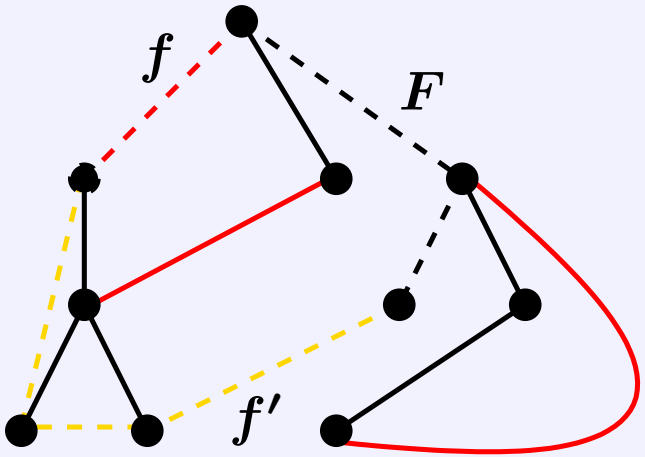
A piros szabályt használjuk: Ekkor f piros lesz. $\Longrightarrow$

Ha f nem éle F-nek ✓



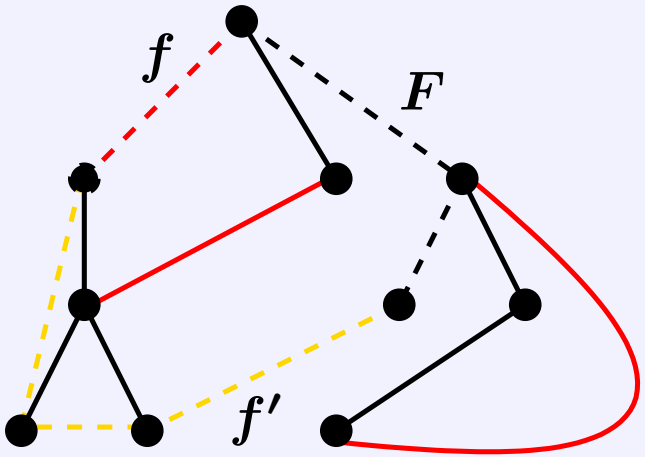
Ha $f \in F$, akkor az f törlésével az F két komponensre esik.

A piros szabályt használjuk: Ekkor f piros lesz. $\Longrightarrow$
Ha f nem éle F-nek ✓



Ha $f \in F$, akkor az f törlésével az F két komponensre esik.
$\Longrightarrow$ A körnek, amelyre a piros szabályt alkalmaztuk. van olyan $f' \neq f$ éle, ami a két komponens között fut.

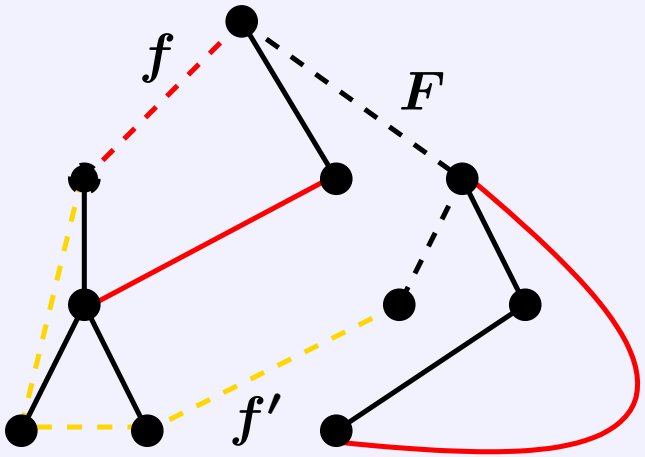
A piros szabályt használjuk: Ekkor f piros lesz. $\Longrightarrow$
Ha f nem éle F-nek ✓



Ha $f \in F$, akkor az f törlésével az F két komponensre esik.
$\Longrightarrow$ A körnek, amelyre a piros szabályt alkalmaztuk. van olyan $f' \neq f$ éle, ami a két komponens között fut.
A régi színezés takarossága és a piros szabály miatt az f' színtelen és $c(f') \leq c(f)$.

A piros szabályt használjuk: Ekkor f piros lesz. $\Longrightarrow$

Ha f nem éle F-nek ✓



Ha $f \in F$, akkor az f törlésével az F két komponensre esik.

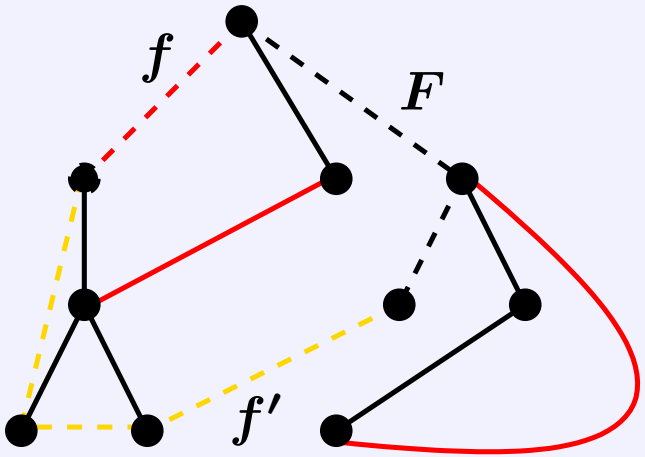
$\Longrightarrow$ A körnek, amelyre a piros szabályt alkalmaztuk. van olyan $f' \neq f$ éle, ami a két komponens között fut.

A régi színezés takarossága és a piros szabály miatt az f' színtelen és $c(f') \leq c(f)$.

az F-be f helyett f'-t véve a kapott F' egy minimális költségű feszítőfa lesz ✓

A piros szabályt használjuk: Ekkor f piros lesz. $\Longrightarrow$

Ha f nem éle F-nek ✓



Miért nem akadunk el soha?

Ha $f \in F$, akkor az f törlésével az F két komponensre esik.

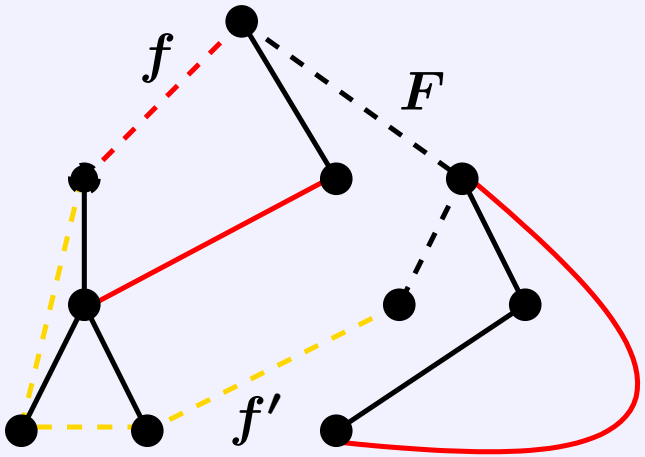
$\Longrightarrow$ A körnek, amelyre a piros szabályt alkalmaztuk. van olyan $f' \neq f$ éle, ami a két komponens között fut.

A régi színezés takarossága és a piros szabály miatt az f' színtelen és $c(f') \leq c(f)$.

az F-be f helyett f'-t véve a kapott F' egy minimális költségű feszítőfa lesz ✓

A piros szabályt használjuk: Ekkor f piros lesz. $\Longrightarrow$

Ha f nem éle F-nek ✓



Ha $f \in F$, akkor az f törlésével az F két komponensre esik.

$\Longrightarrow$ A körnek, amelyre a piros szabályt alkalmaztuk. van olyan $f' \neq f$ éle, ami a két komponens között fut.

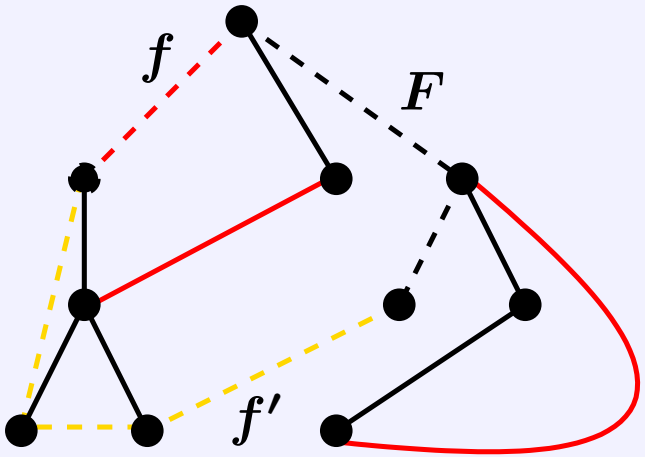
A régi színezés takarossága és a piros szabály miatt az f' színtelen és $c(f') \leq c(f)$.

az F-be f helyett f'-t véve a kapott F' egy minimális költségű feszítőfa lesz ✓

Miért nem akadunk el soha?

Tegyük fel, hogy van még egy f színtelen él.

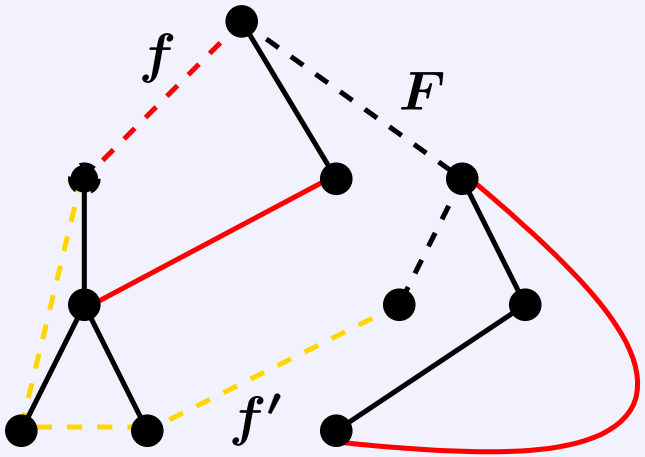
A piros szabályt használjuk: Ekkor f piros lesz. $\Longrightarrow$
Ha f nem éle F-nek ✓



Ha $f \in F$, akkor az f törlésével az F két komponensre esik.
$\Longrightarrow$ A körnek, amelyre a piros szabályt alkalmaztuk. van olyan $f' \neq f$ éle, ami a két komponens között fut.
A régi színezés takarossága és a piros szabály miatt az f' színtelen és $c(f') \leq c(f)$.
az F-be f helyett f'-t véve a kapott F' egy minimális költségű feszítőfa lesz ✓

Miért nem akadunk el soha?
Tegyük fel, hogy van még egy f színtelen él.
A színezés takaros $\Longrightarrow$ a kék élek egy erdőt alkotnak.

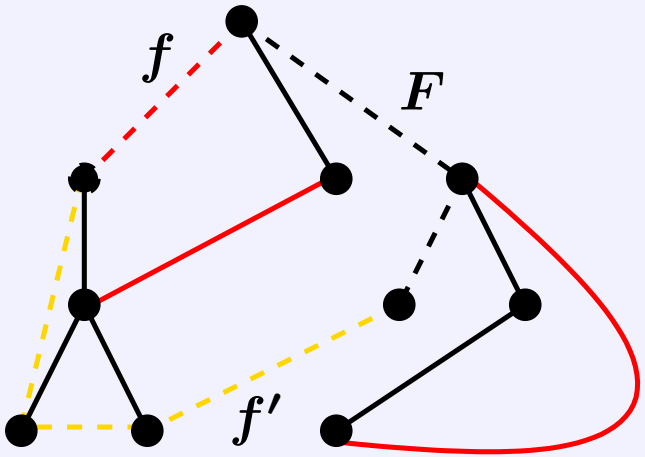
A piros szabályt használjuk: Ekkor f piros lesz. $\Longrightarrow$
Ha f nem éle F-nek ✓



Ha $f \in F$, akkor az f törlésével az F két komponensre esik.
$\Longrightarrow$ A körnek, amelyre a piros szabályt alkalmaztuk. van olyan $f' \neq f$ éle, ami a két komponens között fut.
A régi színezés takarossága és a piros szabály miatt az f' színtelen és $c(f') \leq c(f)$.
az F-be f helyett f'-t véve a kapott F' egy minimális költségű feszítőfa lesz ✓

Miért nem akadunk el soha?
Tegyük fel, hogy van még egy f színtelen él.
A színezés takaros $\Longrightarrow$ a kék élek egy erdőt alkotnak.
$\Longrightarrow$ Ha f végpontjai ugyanabban a kék fában vannak, akkor a piros szabály alkalmazható arra körre, aminek az élei f és az f végpontjait összekötő (egyetlen) kék út élei.

A piros szabályt használjuk: Ekkor f piros lesz. $\Longrightarrow$
Ha f nem éle F-nek ✓



Ha $f \in F$, akkor az f törlésével az F két komponensre esik.
$\Longrightarrow$ A körnek, amelyre a piros szabályt alkalmaztuk. van olyan $f' \neq f$ éle, ami a két komponens között fut.
A régi színezés takarossága és a piros szabály miatt az f' színtelen és $c(f') \leq c(f)$.
az F-be f helyett f'-t véve a kapott F' egy minimális költségű feszítőfa lesz ✓

Miért nem akadunk el soha?
Tegyük fel, hogy van még egy f színtelen él.
A színezés takaros $\Longrightarrow$ a kék élek egy erdőt alkotnak.
$\Longrightarrow$ Ha f végpontjai ugyanabban a kék fában vannak, akkor a piros szabály alkalmazható arra körre, aminek az élei f és az f végpontjait összekötő (egyetlen) kék út élei.
$\Longrightarrow$ Ha f különböző kék fákat köt össze, akkor pedig a kék szabály működik; X legyen az egyik olyan fa csúcshalmaza, amihez f csatlakozik. (Ez utóbbi esetben nem biztos, hogy f fog színt kapni a következő lépésben.)

Tétel. *Ha a piros-kék algoritmussal befestjük az összefüggő $G = (V, E)$ gráf minden élét, akkor a kék élek egy minimális költségű feszítőfa élei. Sőt, ez már akkor is igaz, amikor van $|V| - 1$ kék élünk (és esetleg van még színezetlen él).*

Tétel. *Ha a piros-kék algoritmussal befestjük az összefüggő $G = (V, E)$ gráf minden élét, akkor a kék élek egy minimális költségű feszítőfa élei. Sőt, ez már akkor is igaz, amikor van $|V| - 1$ kék élünk (és esetleg van még színezetlen él).*

Bizonyítás: Az első állítás $\Longleftarrow$ a végső színezés is takaros.

Tétel. *Ha a piros-kék algoritmussal befestjük az összefüggő $G = (V, E)$ gráf minden élét, akkor a kék élek egy minimális költségű feszítőfa élei. Sőt, ez már akkor is igaz, amikor van $|V| - 1$ kék élünk (és esetleg van még színezetlen él).*

Bizonyítás: Az első állítás $\Longleftarrow$ a végső színezés is takaros.
A második: végül összesen $|V| - 1$ kék él lesz. Ha már van ennyi, akkor több nem keletkezhet.

Prim, Kruskal és Borůvka módszerei

A recept *helyessége* szempontjából tehát közömbös a sorrend, *hatékonyság* szempontjából viszont nem.

Prim, Kruskal és Borůvka módszerei

A recept *helyessége* szempontjából tehát közömbös a sorrend, *hatékonyság* szempontjából viszont nem.

PRIM MÓDSZERE: Legyen s a G egy rögzített csúcsa. Minden egyes színező lépéssel az s-et tartalmazó F kék fát bővítjük. Kezdetben az F csúcshalmaza $\{s\}$, végül pedig az egész V.

Prim, Kruskal és Borůvka módszerei

A recept *helyessége* szempontjából tehát közömbös a sorrend, *hatékonyság* szempontjából viszont nem.

PRIM MÓDSZERE: Legyen s a G egy rögzített csúcsa. Minden egyes színező lépéssel az s-et tartalmazó F kék fát bővítjük. Kezdetben az F csúcshalmaza $\{s\}$, végül pedig az egész V. A következő kék élnek az egyik legkisebb súlyú élet választjuk azok közül, amelyek F-beli pontból F-en kívüli pontba mennek.

Prim, Kruskal és Borůvka módszerei

A recept *helyessége* szempontjából tehát közömbös a sorrend, *hatékonyság* szempontjából viszont nem.

PRIM MÓDSZERE: Legyen s a G egy rögzített csúcsa. Minden egyes színező lépéssel az s-et tartalmazó F kék fát bővítjük. Kezdetben az F csúcshalmaza $\{s\}$, végül pedig az egész V. A következő kék élnek az egyik legkisebb súlyú élet választjuk azok közül, amelyek F-beli pontból F-en kívüli pontba mennek.

KRUSKAL MÓDSZERE: A következő befestendő f él legyen mindig a legkisebb súlyú színtelen él.

Prim, Kruskal és Borůvka módszerei

A recept *helyessége* szempontjából tehát közömbös a sorrend, *hatékonyság* szempontjából viszont nem.

PRIM MÓDSZERE: Legyen s a G egy rögzített csúcsa. Minden egyes színező lépéssel az s-et tartalmazó F kék fát bővítjük. Kezdetben az F csúcshalmaza $\{s\}$, végül pedig az egész V. A következő kék élnek az egyik legkisebb súlyú élet választjuk azok közül, amelyek F-beli pontból F-en kívüli pontba mennek.

KRUSKAL MÓDSZERE: A következő befestendő f él legyen mindig a legkisebb súlyú színtelen él. Ha az f két végpontja ugyanazon kék fában van, akkor az él legyen piros, különben pedig kék.

Prim, Kruskal és Borůvka módszerei

A recept *helyessége* szempontjából tehát közömbös a sorrend, *hatékonyság* szempontjából viszont nem.

PRIM MÓDSZERE: Legyen s a G egy rögzített csúcsa. Minden egyes színező lépéssel az s-et tartalmazó F kék fát bővítjük. Kezdetben az F csúcshalmaza $\{s\}$, végül pedig az egész V. A következő kék élnek az egyik legkisebb súlyú élet választjuk azok közül, amelyek F-beli pontból F-en kívüli pontba mennek.

KRUSKAL MÓDSZERE: A következő befestendő f él legyen mindig a legkisebb súlyú színtelen él. Ha az f két végpontja ugyanazon kék fában van, akkor az él legyen piros, különben pedig kék.

BORŮVKA MÓDSZERE: Minden egyes kék fához válasszuk ki a legkisebb súlyú belőle kimenő (színtelen) élet. Színezzük kékre a kiválasztott éleket.

Prim módszere

Mindig a kék szabályt alkalmazzuk: Válasszuk X-nek a meglévő fa ponthalmazát.

Prim módszere

Mindig a kék szabályt alkalmazzuk: Válasszuk X-nek a meglévő fa ponthalmazát. A kék élek végig fát alkotnak.

Prim módszere

Mindig a kék szabályt alkalmazzuk: Válasszuk X-nek a meglévő fa ponthalmazát. A kék élek végig fát alkotnak.

procedure Prim (G: gráf; **var** F: élek halmaza);
    var
        U: csúcsok halmaza;
        u, v: csúcsok;
    begin
        $F := \emptyset$;
        $U := \{1\}$;
        while $U \neq V$ **do begin**
(∗)   legyen (u, v) egy legkisebb súlyú olyan él,
            melyre $u \in U$ és $v \in V \setminus U$;
            $F := F \cup \{(u, v)\}$;
            $U := U \cup \{v\}$
        end
    end

Prim módszere

Mindig a kék szabályt alkalmazzuk: Válasszuk X-nek a meglévő fa ponthalmazát. A kék élek végig fát alkotnak.

procedure Prim (G: gráf; **var** F: élek halmaza);
    var
        U: csúcsok halmaza;
        u, v: csúcsok;
    begin
        $F := \emptyset$;
        $U := \{1\}$;
        while $U \neq V$ **do begin**
($*$)   legyen (u, v) egy legkisebb súlyú olyan él,
            melyre $u \in U$ és $v \in V \setminus U$;
            $F := F \cup \{(u, v)\}$;
            $U := U \cup \{v\}$
        end
    end

Jól működik, mert piros-kék algoritmus.

Prim módszere

Mindig a kék szabályt alkalmazzuk: Válasszuk X-nek a meglévő fa ponthalmazát. A kék élek végig fát alkotnak.

procedure Prim (G: gráf; **var** F: élek halmaza);
    var
        U: csúcsok halmaza;
        u, v: csúcsok;
    begin
        $F := \emptyset$;
        $U := \{1\}$;
        while $U \neq V$ **do begin**
($*$)   legyen (u, v) egy legkisebb súlyú olyan él,
            melyre $u \in U$ és $v \in V \setminus U$;
            $F := F \cup \{(u, v)\}$;
            $U := U \cup \{v\}$
        end
    end

Jól működik, mert piros-kék algoritmus.

JAVA animáció: Prim módszere

Naiv implementáció

A gráf az élsúlyokat tartalmazó C adjacencia-mátrixával adott.

Naiv implementáció

A gráf az élsúlyokat tartalmazó C adjacencia-mátrixával adott.
Az épp aktuális U és $V \setminus U$ halmazok közt futó legkisebb súlyú élek kiválasztása $\Longrightarrow O(n^2)$ lépés

Naiv implementáció

A gráf az élsúlyokat tartalmazó C adjacencia-mátrixával adott.
Az épp aktuális U és $V \setminus U$ halmazok közt futó legkisebb súlyú élek kiválasztása $\Longrightarrow O(n^2)$ lépés
$\Longrightarrow$ Minden $V \setminus U$-beli csúcshoz tároljuk, hogy milyen messze van az U halmaztól

Naiv implementáció

A gráf az élsúlyokat tartalmazó C adjacencia-mátrixával adott.
Az épp aktuális U és $V \setminus U$ halmazok közt futó legkisebb súlyú élek kiválasztása $\Longrightarrow O(n^2)$ lépés
$\Longrightarrow$ Minden $V \setminus U$-beli csúcshoz tároljuk, hogy milyen messze van az U halmaztól

$$\text{KÖZEL}[i] = \begin{cases} * & \text{ha } i \in U \\ \text{egy az } i\text{-hez legközelebbi } U\text{-beli csúcs} & \text{ha } i \in V \setminus U \end{cases}$$

Naiv implementáció

A gráf az élsúlyokat tartalmazó C adjacencia-mátrixával adott.
Az épp aktuális U és $V \setminus U$ halmazok közt futó legkisebb súlyú élek kiválasztása $\Longrightarrow O(n^2)$ lépés
$\Longrightarrow$ Minden $V \setminus U$-beli csúcshoz tároljuk, hogy milyen messze van az U halmaztól

$$\text{KÖZEL}[i] = \begin{cases} * & \text{ha } i \in U \\ \text{egy az } i\text{-hez legközelebbi } U\text{-beli csúcs} & \text{ha } i \in V \setminus U \end{cases}$$

$$\text{MINSÚLY}[i] = \begin{cases} * & \text{ha } i \in U \\ C[i,j] & \text{ha KÖZEL}[i] = j \neq * \end{cases}$$

Naiv implementáció

A gráf az élsúlyokat tartalmazó C adjacencia-mátrixával adott.
Az épp aktuális U és $V \setminus U$ halmazok közt futó legkisebb súlyú élek kiválasztása $\Longrightarrow O(n^2)$ lépés
$\Longrightarrow$ Minden $V \setminus U$-beli csúcshoz tároljuk, hogy milyen messze van az U halmaztól

$$\text{KÖZEL}[i] = \begin{cases} * & \text{ha } i \in U \\ \text{egy az } i\text{-hez legközelebbi } U\text{-beli csúcs} & \text{ha } i \in V \setminus U \end{cases}$$

$$\text{MINSÚLY}[i] = \begin{cases} * & \text{ha } i \in U \\ C[i, j] & \text{ha KÖZEL}[i] = j \neq * \end{cases}$$

A következő kék él az $(i, \text{KÖZEL}[i])$ élek közül kerül majd ki $\Longrightarrow$ *kékes* élek

$$\text{KÖZEL}[i] := \begin{cases} * & \text{ha } i = 1 \\ 1 & \text{ha } i \neq 1 \end{cases}$$

$$\text{MINSÚLY}[i] := \begin{cases} * & \text{ha } i = 1 \\ C[i,1] & \text{ha } i \neq 1 \end{cases}$$

$$\text{KÖZEL}[i] := \begin{cases} * & \text{ha } i = 1 \\ 1 & \text{ha } i \neq 1 \end{cases}$$

$$\text{MINSÚLY}[i] := \begin{cases} * & \text{ha } i = 1 \\ C[i,1] & \text{ha } i \neq 1 \end{cases}$$

- *A következő kék él kiválasztása:* megkeressük a MINSÚLY[ ] tömb minimumát, $\Longrightarrow$ legrövidebb kékes él hossza $\Longrightarrow$ k-ba mutató

$$\text{KÖZEL}[i] := \begin{cases} * & \text{ha } i = 1 \\ 1 & \text{ha } i \neq 1 \end{cases}$$

$$\text{MINSÚLY}[i] := \begin{cases} * & \text{ha } i = 1 \\ C[i,1] & \text{ha } i \neq 1 \end{cases}$$

- *A következő kék él kiválasztása:* megkeressük a MINSÚLY[ ] tömb minimumát, $\Longrightarrow$ legrövidebb kékes él hossza $\Longrightarrow$ k-ba mutató
  A minimumkeresés költsége: $O(n)$ lépés

$$\text{KÖZEL}[i] := \begin{cases} * & \text{ha } i = 1 \\ 1 & \text{ha } i \neq 1 \end{cases}$$

$$\text{MINSÚLY}[i] := \begin{cases} * & \text{ha } i = 1 \\ C[i,1] & \text{ha } i \neq 1 \end{cases}$$

- *A következő kék él kiválasztása:* megkeressük a MINSÚLY[ ] tömb minimumát, $\Longrightarrow$ legrövidebb kékes él hossza $\Longrightarrow$ k-ba mutató
  A minimumkeresés költsége: $O(n)$ lépés
  a (KÖZEL$[k], k$) élet fogjuk F-be tenni, k-t pedig U-ba.

$$\text{KÖZEL}[i] := \begin{cases} * & \text{ha } i = 1 \\ 1 & \text{ha } i \neq 1 \end{cases}$$

$$\text{MINSÚLY}[i] := \begin{cases} * & \text{ha } i = 1 \\ C[i,1] & \text{ha } i \neq 1 \end{cases}$$

- *A következő kék él kiválasztása:* megkeressük a MINSÚLY[ ] tömb minimumát, $\Longrightarrow$ legrövidebb kékes él hossza $\Longrightarrow$ k-ba mutató
  A minimumkeresés költsége: $O(n)$ lépés
  a (KÖZEL$[k], k$) élet fogjuk F-be tenni, k-t pedig U-ba.
  $\Longrightarrow$ MINSÚLY$[k] :=$ KÖZEL$[k] := *$.

$$\text{KÖZEL}[i] := \begin{cases} * & \text{ha } i = 1 \\ 1 & \text{ha } i \neq 1 \end{cases}$$

$$\text{MINSÚLY}[i] := \begin{cases} * & \text{ha } i = 1 \\ C[i,1] & \text{ha } i \neq 1 \end{cases}$$

- *A következő kék él kiválasztása:* megkeressük a MINSÚLY[ ] tömb minimumát, $\Longrightarrow$ legrövidebb kékes él hossza $\Longrightarrow$ k-ba mutató
  A minimumkeresés költsége: $O(n)$ lépés
  a (KÖZEL$[k], k$) élet fogjuk F-be tenni, k-t pedig U-ba.
  $\Longrightarrow$ MINSÚLY$[k]$:= KÖZEL$[k] := *$.
- *A két tömb felfrissítése:* A $C[k,i]$ és a MINSÚLY$[i]$ értékeket ($i \in V \setminus U$) kell összevetni.

$$\text{KÖZEL}[i] := \begin{cases} * & \text{ha } i = 1 \\ 1 & \text{ha } i \neq 1 \end{cases}$$

$$\text{MINSÚLY}[i] := \begin{cases} * & \text{ha } i = 1 \\ C[i,1] & \text{ha } i \neq 1 \end{cases}$$

- *A következő kék él kiválasztása:* megkeressük a MINSÚLY[ ] tömb minimumát, $\Longrightarrow$ legrövidebb kékes él hossza $\Longrightarrow$ k-ba mutató
  A minimumkeresés költsége: $O(n)$ lépés
  a (KÖZEL$[k], k$) élet fogjuk F-be tenni, k-t pedig U-ba.
  $\Longrightarrow$ MINSÚLY$[k] :=$ KÖZEL$[k] := *$.
- *A két tömb felfrissítése:* A $C[k,i]$ és a MINSÚLY$[i]$ értékeket ($i \in V \setminus U$) kell összevetni. $\Longrightarrow$

```
if KÖZEL[i] ≠ * and C[k,i] < MINSÚLY[i] then begin
    KÖZEL[i] := k;
    MINSÚLY[i] := C[k,i]
end
```

$$\text{KÖZEL}[i] := \begin{cases} * & \text{ha } i = 1 \\ 1 & \text{ha } i \neq 1 \end{cases}$$

$$\text{MINSÚLY}[i] := \begin{cases} * & \text{ha } i = 1 \\ C[i,1] & \text{ha } i \neq 1 \end{cases}$$

- *A következő kék él kiválasztása:* megkeressük a MINSÚLY[ ] tömb minimumát, $\Longrightarrow$ legrövidebb kékes él hossza $\Longrightarrow$ k-ba mutató
  A minimumkeresés költsége: $O(n)$ lépés
  a (KÖZEL$[k], k$) élet fogjuk F-be tenni, k-t pedig U-ba.
  $\Longrightarrow$ MINSÚLY$[k] :=$ KÖZEL$[k] := *$.
- *A két tömb felfrissítése:* A $C[k,i]$ és a MINSÚLY$[i]$ értékeket ($i \in V \setminus U$) kell összevetni. $\Longrightarrow$

```
if KÖZEL[i] ≠ * and C[k,i] < MINSÚLY[i] then begin
    KÖZEL[i] := k;
    MINSÚLY[i] := C[k,i]
end
```

Lépésszám: Egy él színezés $O(n)$ $\Longrightarrow O(n^2)$

$$\text{KÖZEL}[i] := \begin{cases} * & \text{ha } i = 1 \\ 1 & \text{ha } i \neq 1 \end{cases}$$

$$\text{MINSÚLY}[i] := \begin{cases} * & \text{ha } i = 1 \\ C[i,1] & \text{ha } i \neq 1 \end{cases}$$

- *A következő kék él kiválasztása:* megkeressük a MINSÚLY[ ] tömb minimumát, $\Longrightarrow$ legrövidebb kékes él hossza $\Longrightarrow$ k-ba mutató
  A minimumkeresés költsége: $O(n)$ lépés
  a (KÖZEL$[k], k$) élet fogjuk F-be tenni, k-t pedig U-ba.
  $\Longrightarrow$ MINSÚLY$[k] :=$ KÖZEL$[k] := *$.
- *A két tömb felfrissítése:* A $C[k,i]$ és a MINSÚLY$[i]$ értékeket ($i \in V \setminus U$) kell összevetni. $\Longrightarrow$

```
if KÖZEL[i] ≠ * and C[k,i] < MINSÚLY[i] then begin
   KÖZEL[i] := k;
   MINSÚLY[i] := C[k,i]
end
```

Lépésszám: Egy él színezés $O(n)$ $\Longrightarrow O(n^2)$
JAVA animáció: Prim módszere

Kupacos-éllistás implementáció

Építsünk kupacot az aktuális U és $V \setminus U$ közötti élekből.

Kupacos-éllistás implementáció

Építsünk kupacot az aktuális U és $V \setminus U$ közötti élekből.
(néhány) MINTÖR-rel $O(\log(e))$ lépéssel kiválaszthatjuk a minimálisat, amit kékre színezünk

Kupacos-éllistás implementáció

Építsünk kupacot az aktuális U és $V \setminus U$ közötti élekből.
(néhány) MINTÖR-rel $O(\log(e))$ lépéssel kiválaszthatjuk a minimálisat, amit kékre színezünk
Megváltozott $U \implies$ BESZÚR-ral beszúrjuk az új éleket

Kupacos-éllistás implementáció

Építsünk kupacot az aktuális U és $V \setminus U$ közötti élekből.
(néhány) MINTÖR-rel $O(\log(e))$ lépéssel kiválaszthatjuk a minimálisat, amit kékre színezünk
Megváltozott $U \implies$ BESZÚR-ral beszúrjuk az új éleket
Nem törődünk azokkal az élekkel, amik így U-n belül mennek

Kupacos-éllistás implementáció

Építsünk kupacot az aktuális U és $V \setminus U$ közötti élekből.
(néhány) MINTÖR-rel $O(\log(e))$ lépéssel kiválaszthatjuk a minimálisat, amit kékre színezünk
Megváltozott U $\Longrightarrow$ BESZÚR-ral beszúrjuk az új éleket
Nem törődünk azokkal az élekkel, amik így U-n belül mennek
$\Longrightarrow$ ezért lehet, hogy MINTÖR-nél ilyet kapunk elsőre.

Kupacos-éllistás implementáció

Építsünk kupacot az aktuális U és $V \setminus U$ közötti élekből.
(néhány) MINTÖR-rel $O(\log(e))$ lépéssel kiválaszthatjuk a minimálisat, amit kékre színezünk
Megváltozott U $\Longrightarrow$ BESZÚR-ral beszúrjuk az új éleket
Nem törődünk azokkal az élekkel, amik így U-n belül mennek
$\Longrightarrow$ ezért lehet, hogy MINTÖR-nél ilyet kapunk elsőre.

Lépésszám: A kupac mérete sosem haladja meg e-t.

Kupacos-éllistás implementáció

Építsünk kupacot az aktuális U és $V \setminus U$ közötti élekből.
(néhány) MINTÖR-rel $O(\log(e))$ lépéssel kiválaszthatjuk a minimálisat, amit kékre színezünk
Megváltozott U $\Longrightarrow$ BESZÚR-ral beszúrjuk az új éleket
Nem törődünk azokkal az élekkel, amik így U-n belül mennek
$\Longrightarrow$ ezért lehet, hogy MINTÖR-nél ilyet kapunk elsőre.

Lépésszám: A kupac mérete sosem haladja meg e-t.
A kezdeti kupacépítés legfeljebb $O(e)$, az egyes műveletek végrehajtása pedig $O(\log e)$ időt vesz igénybe.

Kupacos-éllistás implementáció

Építsünk kupacot az aktuális U és $V \setminus U$ közötti élekből.
(néhány) MINTÖR-rel $O(\log(e))$ lépéssel kiválaszthatjuk a minimálisat, amit kékre színezünk
Megváltozott U $\Longrightarrow$ BESZÚR-ral beszúrjuk az új éleket
Nem törődünk azokkal az élekkel, amik így U-n belül mennek
$\Longrightarrow$ ezért lehet, hogy MINTÖR-nél ilyet kapunk elsőre.

Lépésszám: A kupac mérete sosem haladja meg e-t.
A kezdeti kupacépítés legfeljebb $O(e)$, az egyes műveletek végrehajtása pedig $O(\log e)$ időt vesz igénybe.
Összesen kevesebb, mint e darab BESZÚR és legfeljebb e darab MINTÖR műveletet végzünk

Kupacos-éllistás implementáció

Építsünk kupacot az aktuális U és $V \setminus U$ közötti élekből.
(néhány) MINTÖR-rel $O(\log(e))$ lépéssel kiválaszthatjuk a minimálisat, amit kékre színezünk
Megváltozott U $\Longrightarrow$ BESZÚR-ral beszúrjuk az új éleket
Nem törődünk azokkal az élekkel, amik így U-n belül mennek
$\Longrightarrow$ ezért lehet, hogy MINTÖR-nél ilyet kapunk elsőre.

Lépésszám: A kupac mérete sosem haladja meg e-t.
A kezdeti kupacépítés legfeljebb $O(e)$, az egyes műveletek végrehajtása pedig $O(\log e)$ időt vesz igénybe.
Összesen kevesebb, mint e darab BESZÚR és legfeljebb e darab MINTÖR műveletet végzünk $\Longrightarrow O(e \log e)$

Kupacos-éllistás implementáció

Építsünk kupacot az aktuális U és $V \setminus U$ közötti élekből.
(néhány) MINTÖR-rel $O(\log(e))$ lépéssel kiválaszthatjuk a minimálisat, amit kékre színezünk
Megváltozott $U \implies$ BESZÚR-ral beszúrjuk az új éleket
Nem törődünk azokkal az élekkel, amik így U-n belül mennek
$\implies$ ezért lehet, hogy MINTÖR-nél ilyet kapunk elsőre.

Lépésszám: A kupac mérete sosem haladja meg e-t.
A kezdeti kupacépítés legfeljebb $O(e)$, az egyes műveletek végrehajtása pedig $O(\log e)$ időt vesz igénybe.
Összesen kevesebb, mint e darab BESZÚR és legfeljebb e darab MINTÖR műveletet végzünk $\implies O(e \log e)$

Johnson: Kombináljuk a két ötletet, nyilvántartjuk a közeli csúcsokat, és d-kupacban tároljuk a kékes éleket

Kupacos-éllistás implementáció

Építsünk kupacot az aktuális U és $V \setminus U$ közötti élekből.
(néhány) MINTÖR-rel $O(\log(e))$ lépéssel kiválaszthatjuk a minimálisat, amit kékre színezünk
Megváltozott $U \implies$ BESZÚR-ral beszúrjuk az új éleket
Nem törődünk azokkal az élekkel, amik így U-n belül mennek
$\implies$ ezért lehet, hogy MINTÖR-nél ilyet kapunk elsőre.

Lépésszám: A kupac mérete sosem haladja meg e-t.
A kezdeti kupacépítés legfeljebb $O(e)$, az egyes műveletek végrehajtása pedig $O(\log e)$ időt vesz igénybe.
Összesen kevesebb, mint e darab BESZÚR és legfeljebb e darab MINTÖR műveletet végzünk $\implies O(e \log e)$

Johnson: Kombináljuk a két ötletet, nyilvántartjuk a közeli csúcsokat, és d-kupacban tároljuk a kékes éleket
$\implies$ ha $n^{1,5} \leq e \implies O(e)$