Mélységi Bejárás, Topológikus sorrend, Pert feladat

Papp László

BME

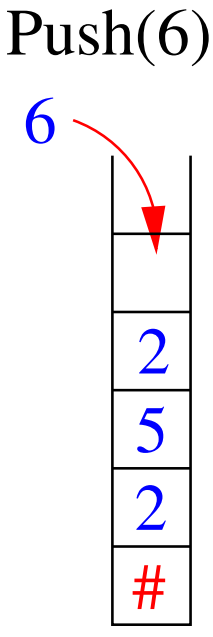
2022. szeptember 30.

Verem

A **verem** egy olyan adatszerkezet, amelyben azonos típusú adatok tárolhatóak úgy, hogy minden egyes lépésben a veremben található adatok közül csak a legutoljára bekerült hozzáférhető.

Verem operációk:

- **Push(x)** A verem tetejére helyezzük az *x* adatot.
- **Pop()** A verem tetején lévő adatot kivesszük és kiolvassuk.



A verem alján célszerű tartani egy speciális kezdőszimbólumot (#) amivel tudjuk tesztelni a verem ürességét. Ha a Pop() #-et ad vissza akkor megtudjuk, hogy a verem üres és gyorsan vissza is tesszük #-et Push(#)-tel.

Mélységi bejárás (DFS = Depth First Search)

„Úgy járom be a gráfot, hogy mindíg a legutoljára bejárt egy még nem bejárt szomszédjába lépek, ha még van ilyen.”
A bejárás során a bejárási fát, és két mennyiséget fogunk számolni. Ezek pedig:

- $m(v)$ a v csúcs **mélységi száma**, ami azt mondja meg, hogy a v csúcsot hányadjára járja be a bejárás.
- $b(v)$ a v csúcs **befejezési száma**, ami azt mondja meg, hogy a v csúcsot hányadjára látogatja meg utoljára a bejárás.

Megjegyzés: A mélységi szám igazából ugyan az mint az elérési szám amit a bejárások általános tárgyalásánál vettünk.

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

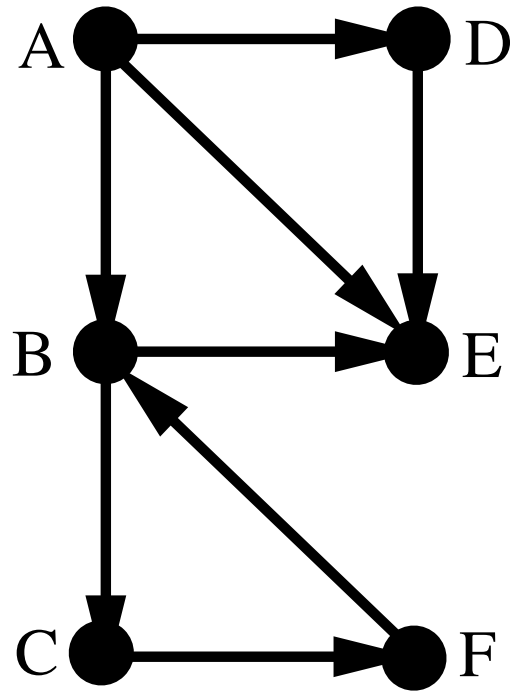
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

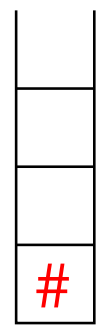
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=1,
j=1,
v=A

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

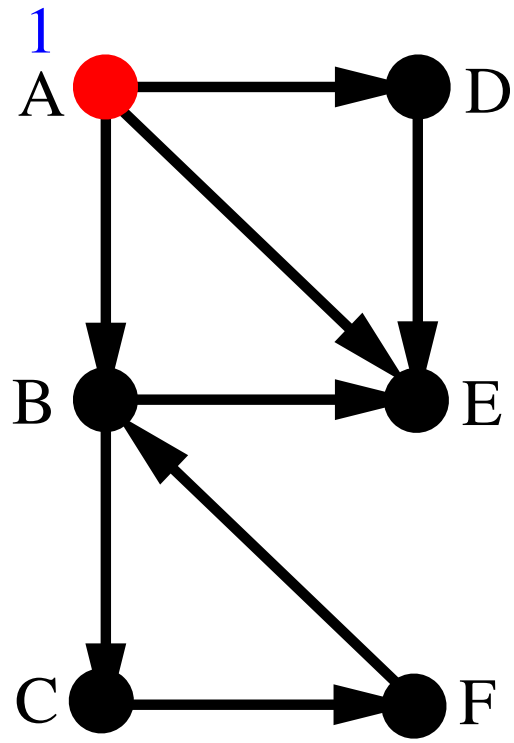
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

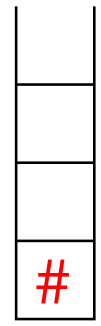
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=2,
j=1,
v=A

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

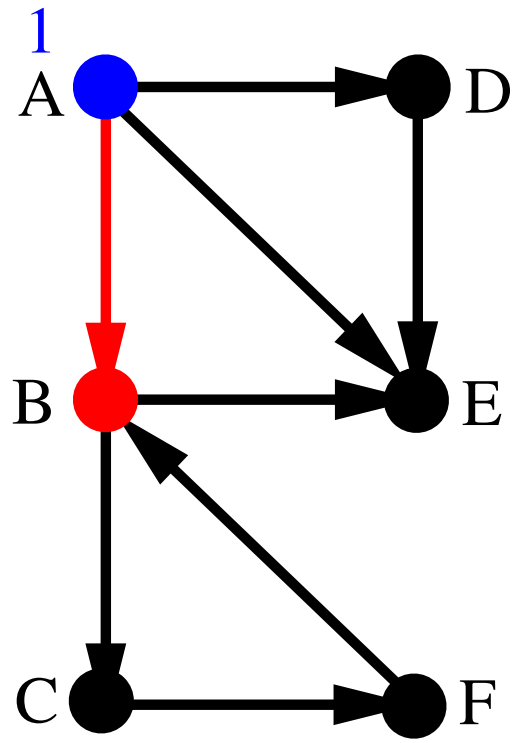
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

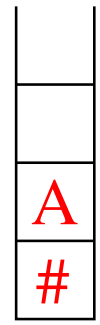
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=2,
j=1,
v=B

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

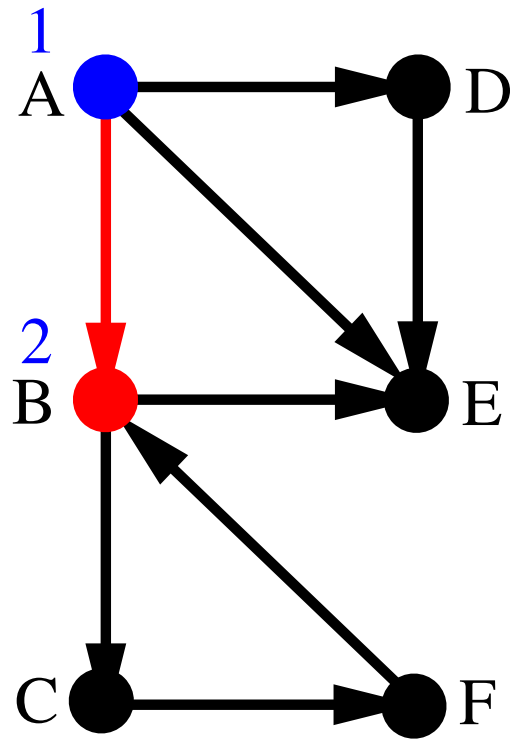
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

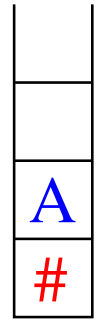
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=3,
j=1,
v=B

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

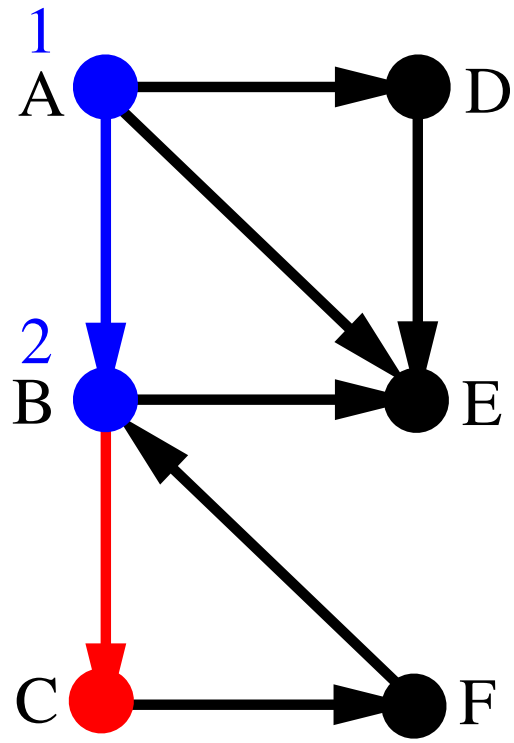
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

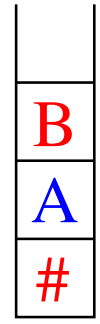
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=3,
j=1,
v=C

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

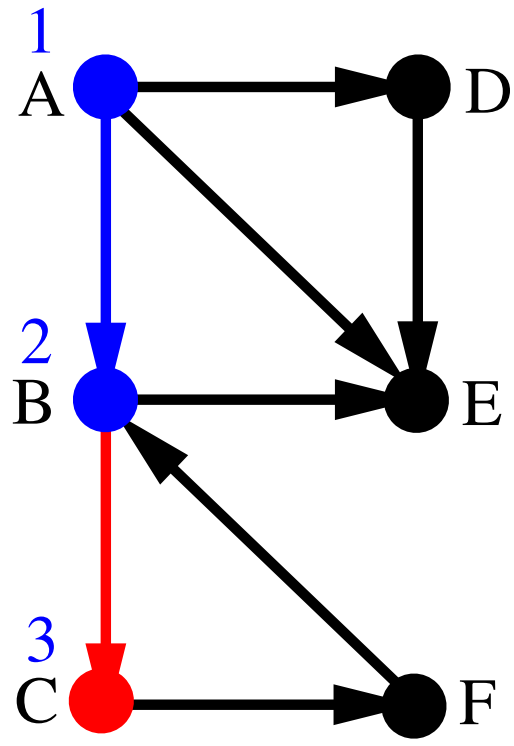
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

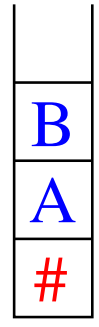
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=4,
j=1,
v=C

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

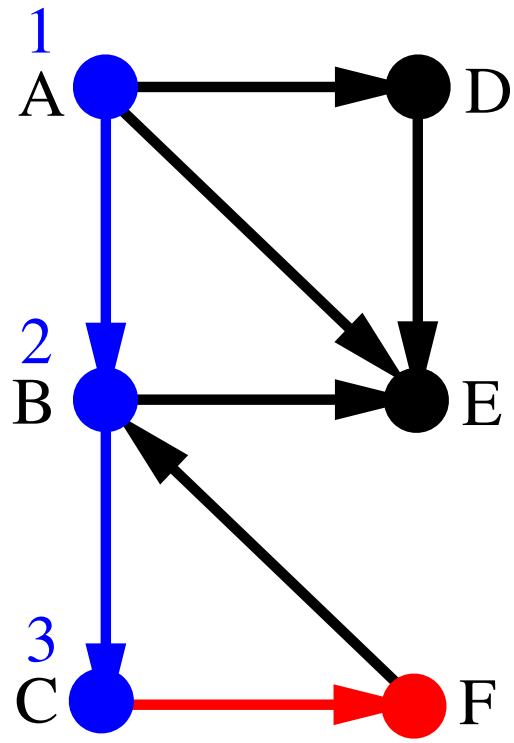
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

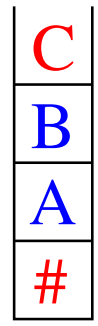
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=4,
j=1,
v=F

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

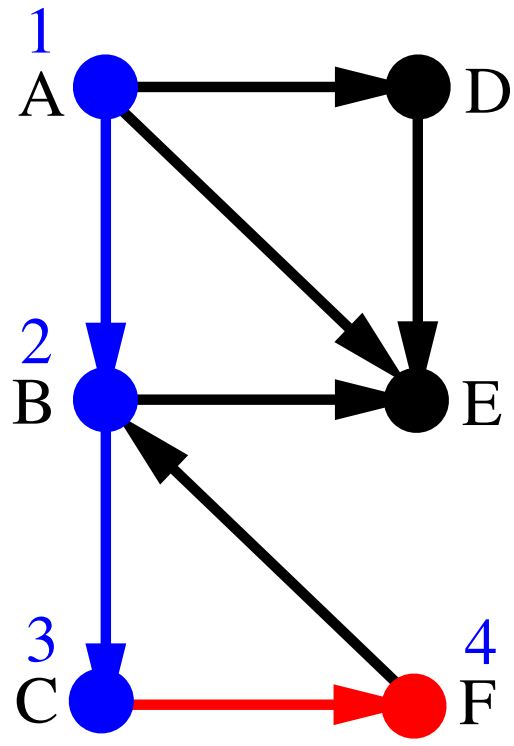
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

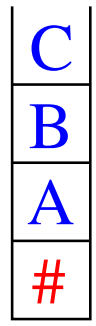
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=5,
j=1,
v=F

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

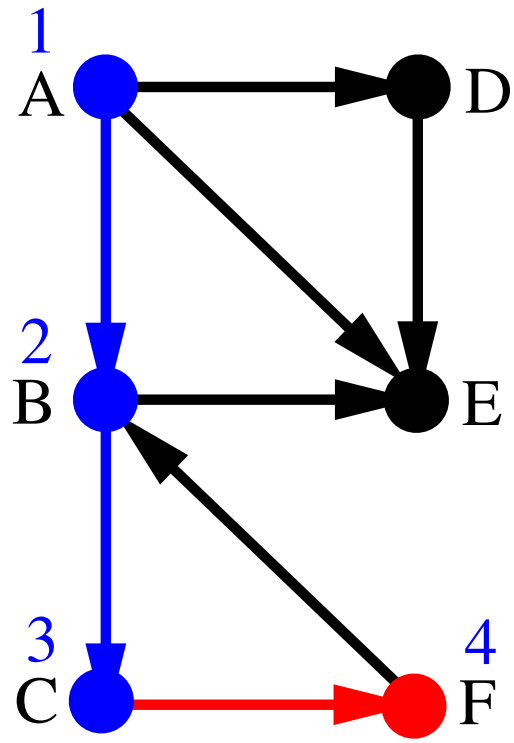
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

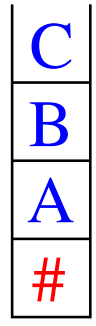
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=5,
j=1,
v=F

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

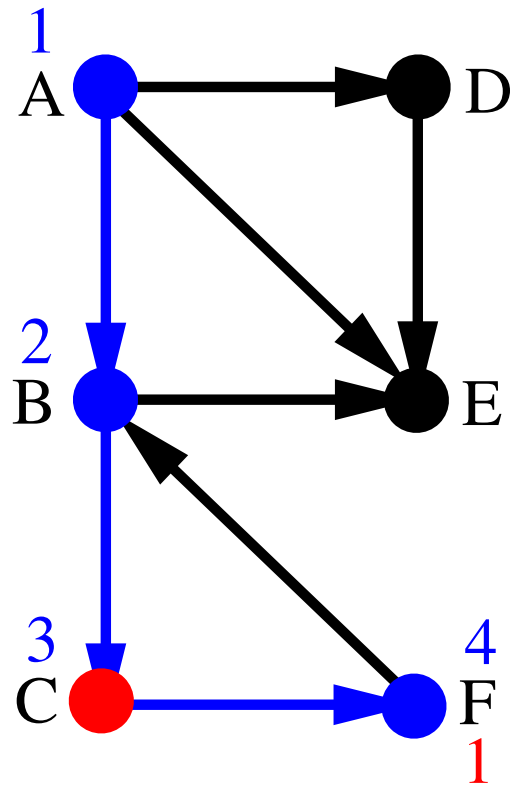
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

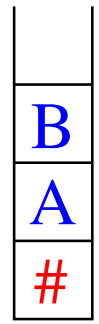
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=5,
j=2,
v=C

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

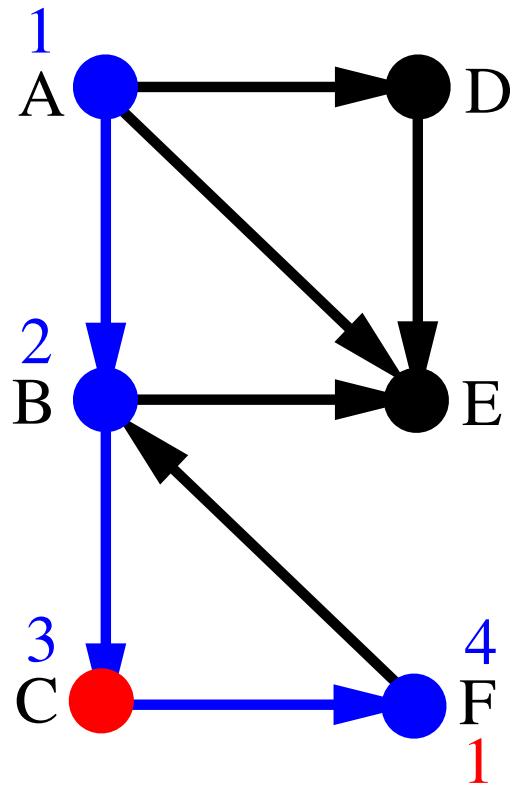
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

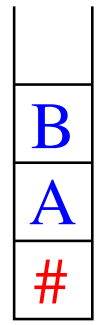
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=5,
j=2,
v=C

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

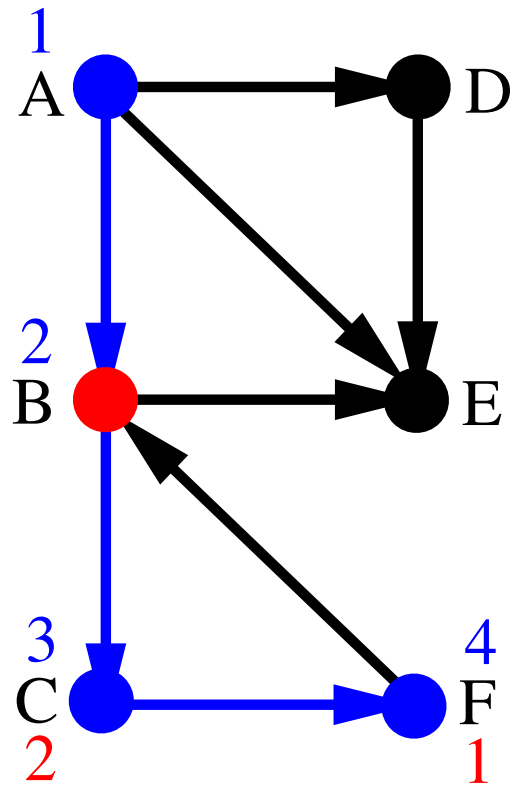
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

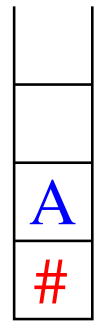
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=5,
j=3,
v=B

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

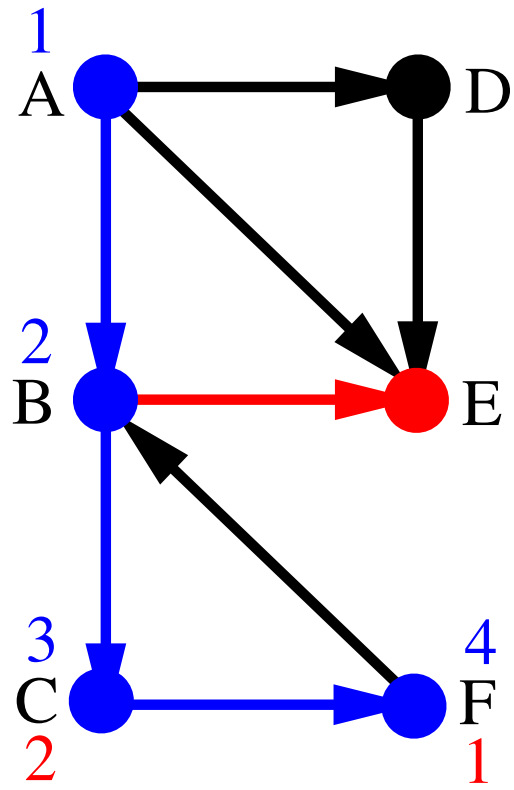
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

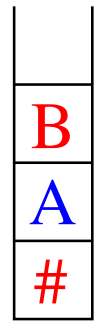
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=5,
j=3,
v=E

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

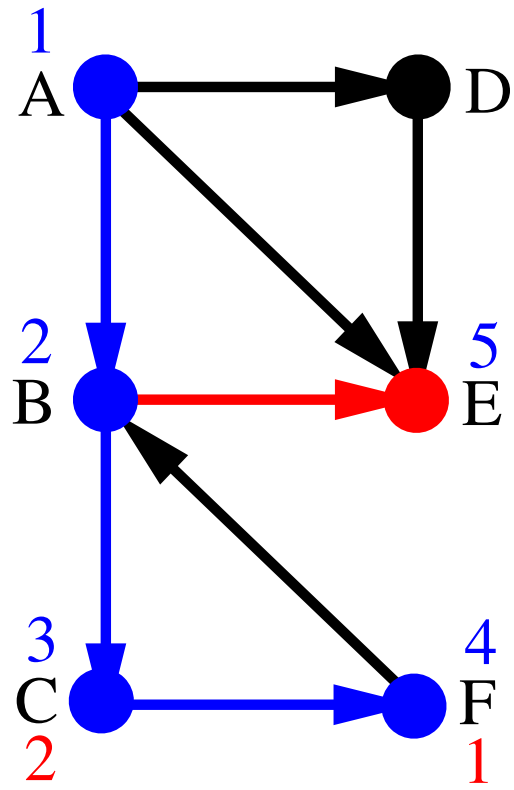
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

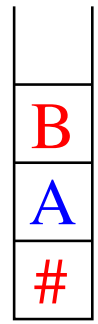
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=6,
j=3,
v=E

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

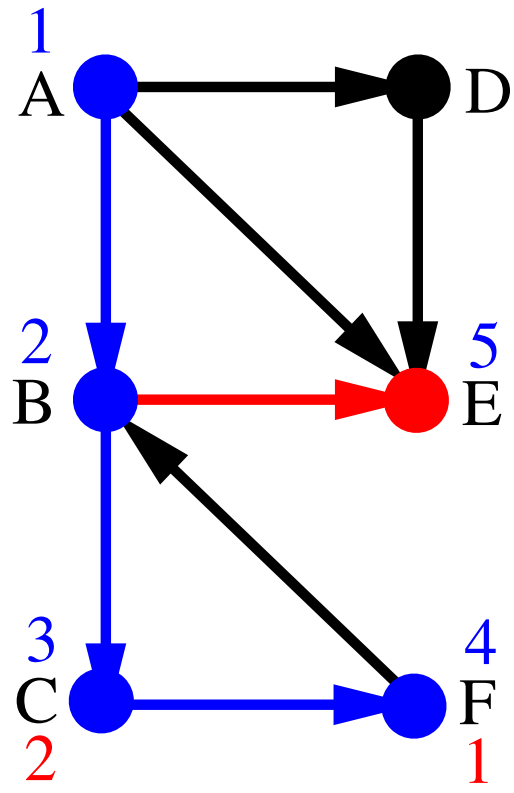
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

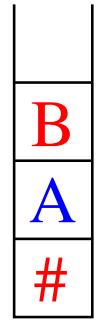
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=6,
j=3,
v=E

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

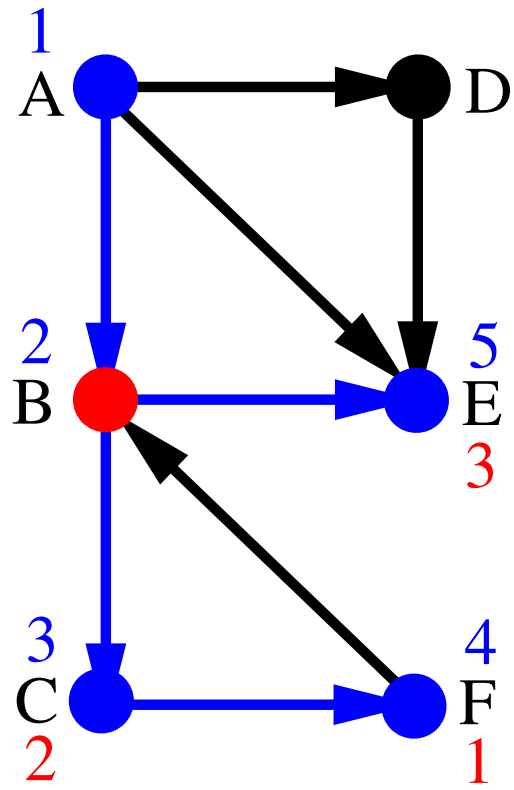
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

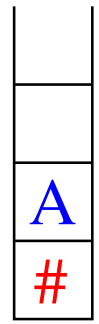
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=6,
j=4,
v=B

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

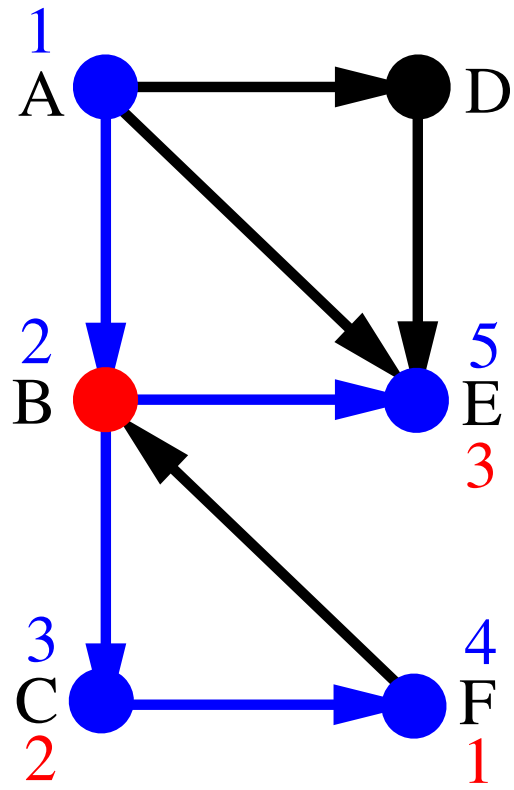
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

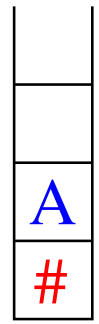
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=6,
j=4,
v=B

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

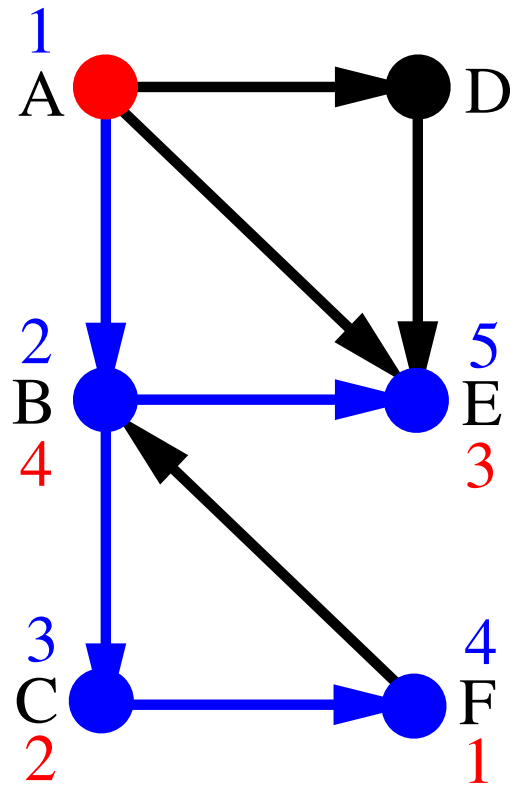
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

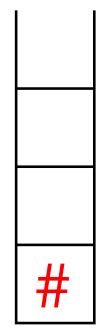
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=6,
j=5,
v=A

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

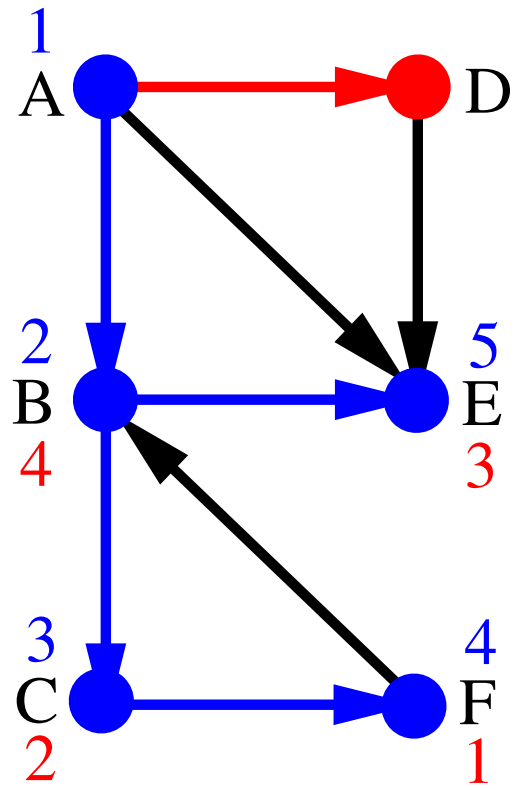
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

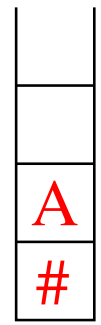
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=6,
j=5,
v=D

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

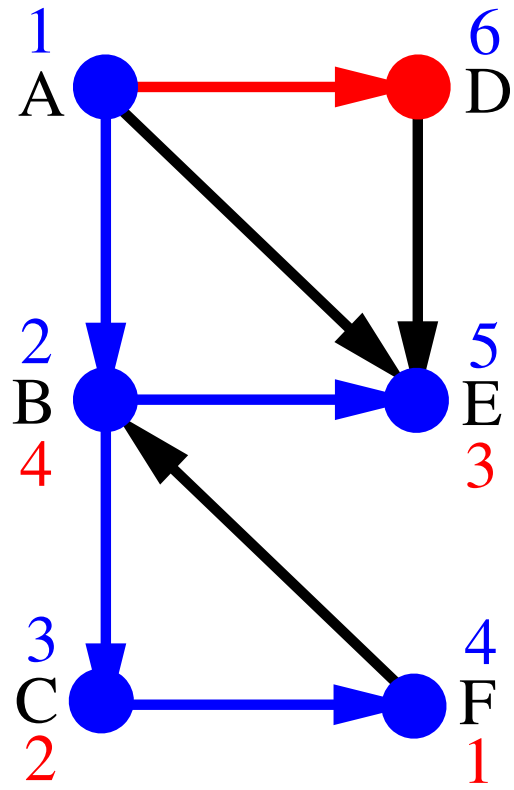
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

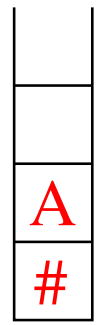
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=7,
j=5,
v=D

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

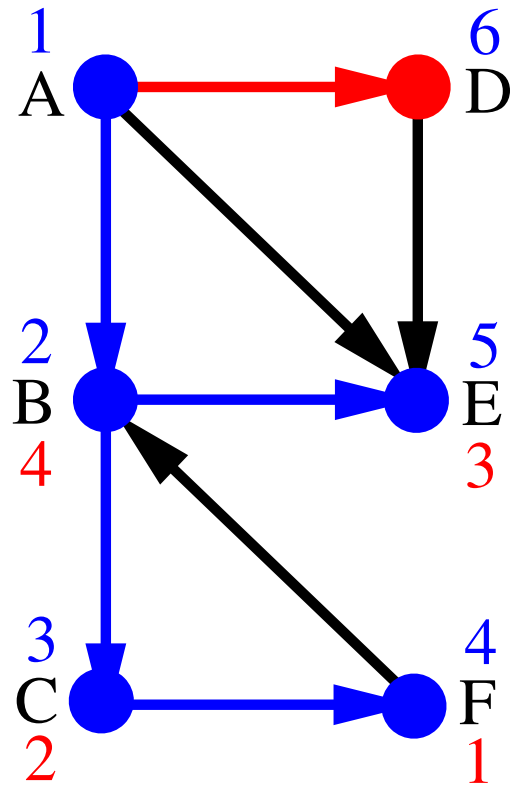
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

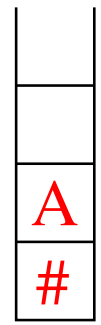
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=7,
j=5,
v=D

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

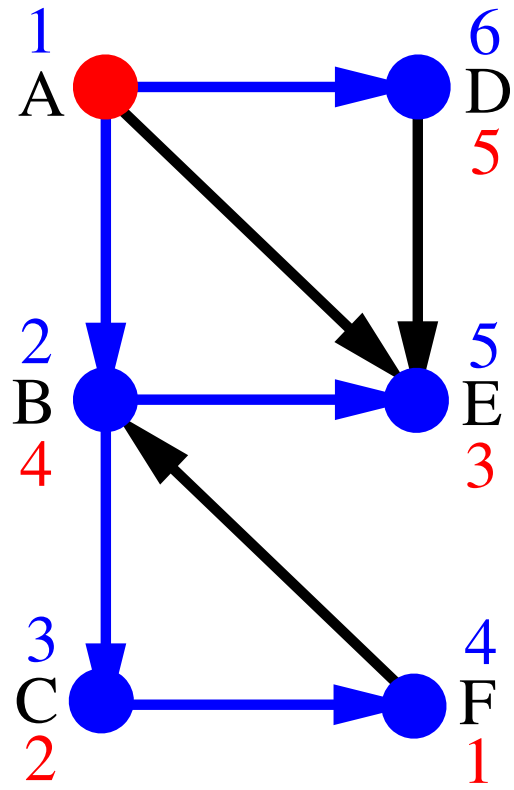
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

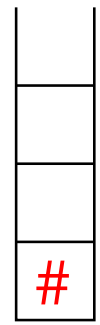
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=7,
j=6,
v=A

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

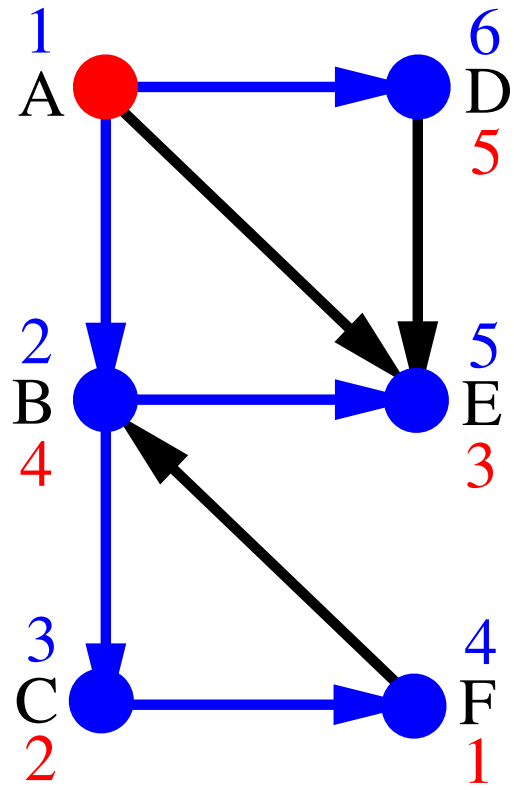
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

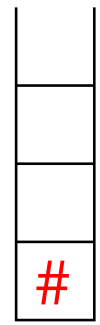
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=7,
j=6,
v=A

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

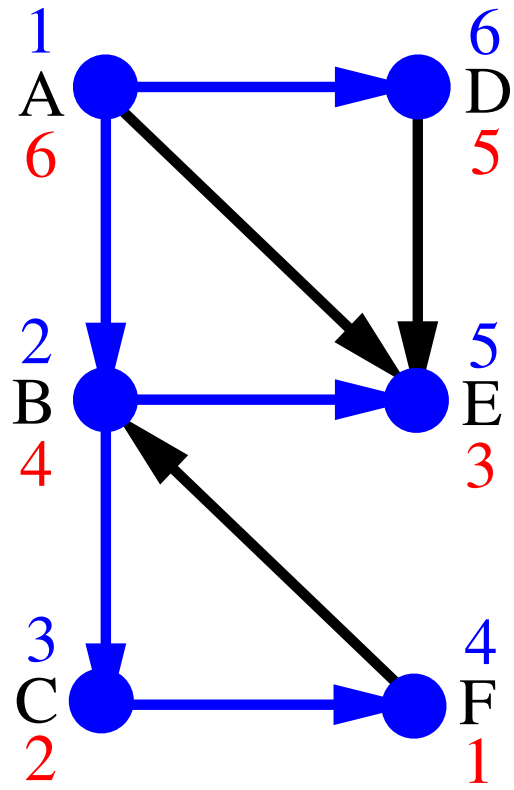
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

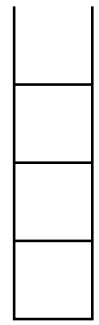
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





i=7,
j=7,
v=A

Mélységi bejárás algoritmusa példával

Input: G irányított vagy irányítatlan gráf és egy $s \in V(G)$ gyökércsúcs.

0. Legyen $i := 1$, $j := 1$, $v := s$ és létrehozok egy vermet # kezdőszimbólummal.

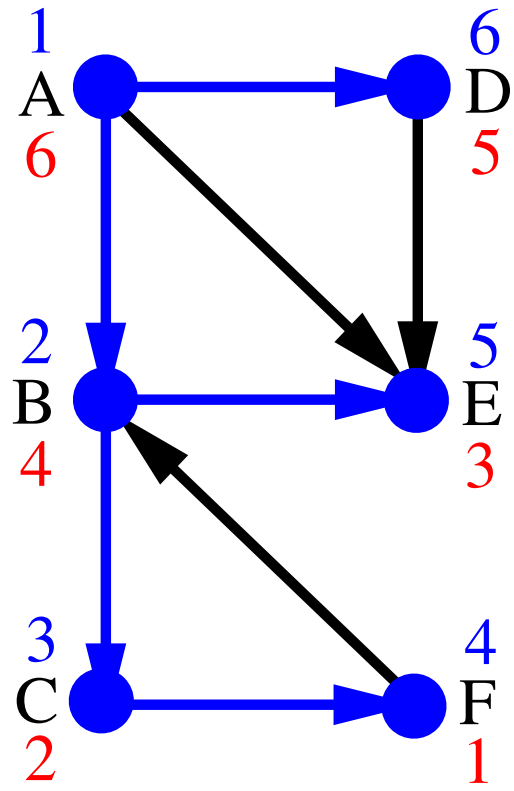
1. Bejárom a v csúcsot, $m(v) := i$; $i := i + 1$.

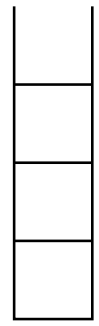
2. Ha v-nek van még egy be nem járt (ki)szomszédja, akkor ezt jelöljük u-val, Push(v), $\{v, \vec{u}\}$ faél, $v := u$ és ugorjunk az 1-es lépésre.

3. Legyen $b(v) := j$; $j := j + 1$. Ha Pop() egy csúcsot ad vissza, akkor ez az új v és ugorjunk a 2-es lépésre.

4. Ha a gráf minden csúcsát bejártuk ($m(v) = |V(G)|$) akkor STOP, különben Push(#) és legyen v egy még nem bejárt csúcs, $i := i + 1$, $j := j + 1$ és ugorjunk az 1-es lépésre.

Példa: A-ból indított DFS





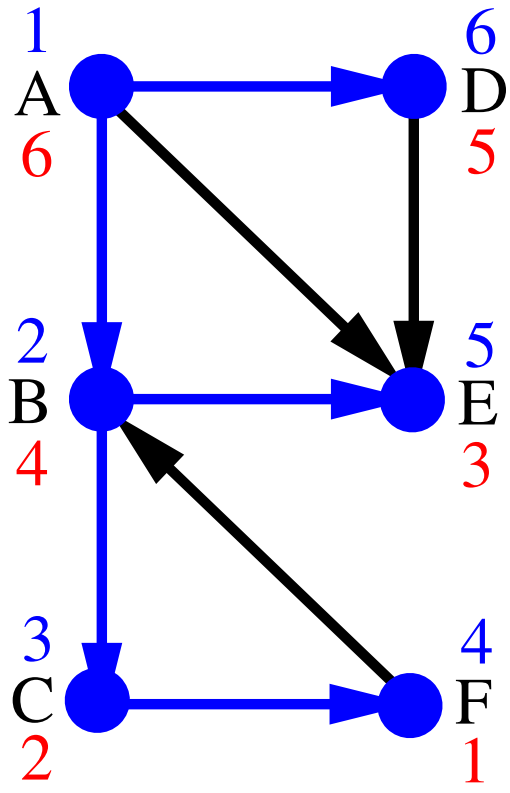
i=7,
j=7,
v=A

Mélységi bejárás verem nélkül

A mélységi keresés algoritmusát le lehet futtatni verem használata nélkül is és a gyakorlatban nem célszerű vermet használni.
Vegyük észre, hogy amikor a 3-as lépésben a veremhez fordulunk azért az információért, hogy melyik csúcsra kell az aktuális x csúcsról ugrani, akkor a verem minden esetben egy olyan y csúcsot fog visszadni amire $\{x,\vec{y}\}$ faél és ez számunkra a kérdés idejében már ismert, vagy azt tudjuk meg a veremből, hogy nincs hova visszaugrani. Olyan x csúcs amire $\{x,\vec{y}\}$ faél ráadásul csak egy lehet.
Tehát amikor az y-nak elfogynak a nem bejárt (ki)-szomszédai, akkor mondhatjuk azt, hogy azon az x-en folytatjuk amiből y-t először elértük, tehát amire $\{x,\vec{y}\}$ faél. Ebben az esetben x-et az y szülőjének szokták nevezni.

Élek osztályozása

A kék élek faélek. Az FB él visszaél, az AE él előreél. A DE keresztél.

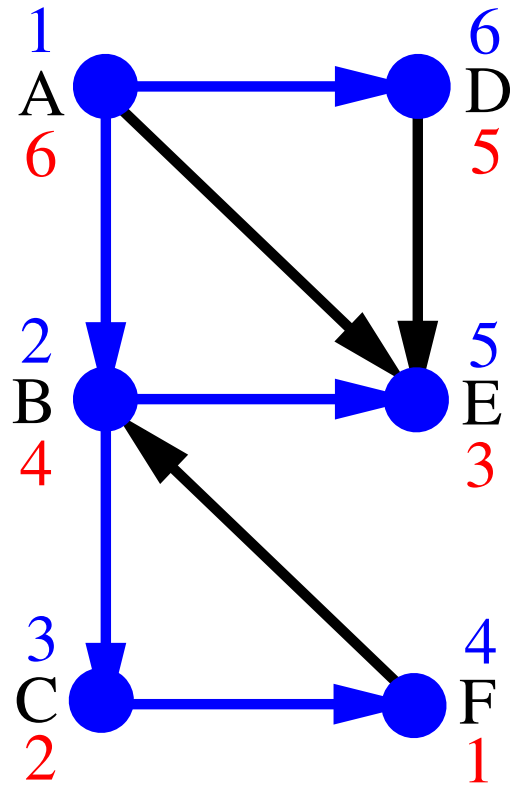


Állítás

- Az uv él előreél vagy faél ha $m(u) < m(v)$ és $b(u) > b(v)$.
- Az uv visszaél ha $m(u) > m(v)$ és $b(u) < b(v)$.
- Az uv keresztél ha $m(u) > m(v)$ és $b(u) > b(v)$.

Élek osztályozása

A kék élek faélek. Az FB él visszaél, az AE él előreél. A DE keresztél.



Állítás

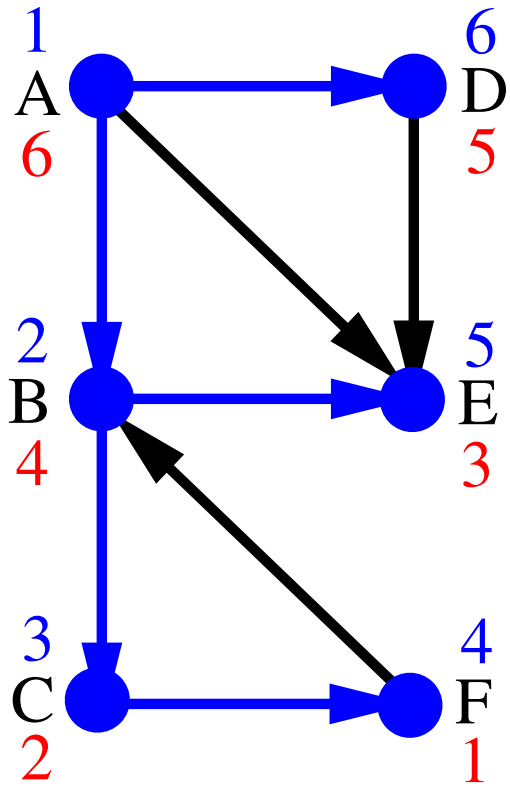
- Az uv él előreél vagy faél ha $m(u) < m(v)$ és $b(u) > b(v)$.
- Az uv visszaél ha $m(u) > m(v)$ és $b(u) < b(v)$.
- Az uv keresztél ha $m(u) > m(v)$ és $b(u) > b(v)$.

Az $m(u) < m(v)$ és $b(u) < b(v)$ eset nem lehetséges, mivel ha u-ba előbb érkezem akkor az uv él miatt átlépek v-be u befejezése előtt, így v-t előbb fogom befejezni mint u-t!

Következmény: Irányítatlan gráf esetén nincsenek keresztélek, hiszen ebben az esetben $uv = vu$ és az $m(u) > m(v)$ és $b(u) > b(v)$ esetből a kitiltott esetet kapjuk!

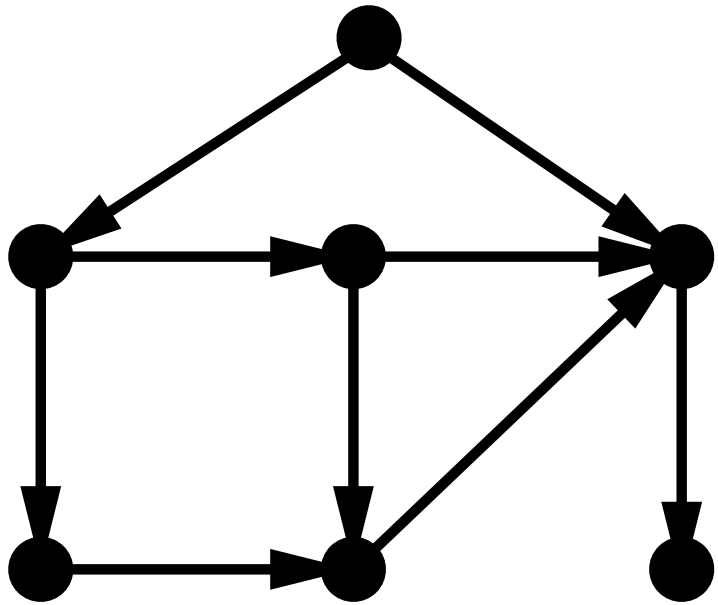
Élek osztályozása: Bizonyítás

Bizonyítás: Az u-ból v-be pontosan akkor van irányított út a mélységi bejárás fájában ha $m(u) < m(v)$ és $b(u) > b(v)$, hiszen egy ilyen irányított út mentén haladva a mélységi számok folyamatosan nőnek, a befejezési számok pedig csökkennek. Ebből és a faél, előreél, visszaél és keresztélek definíciójából következik az állítás.



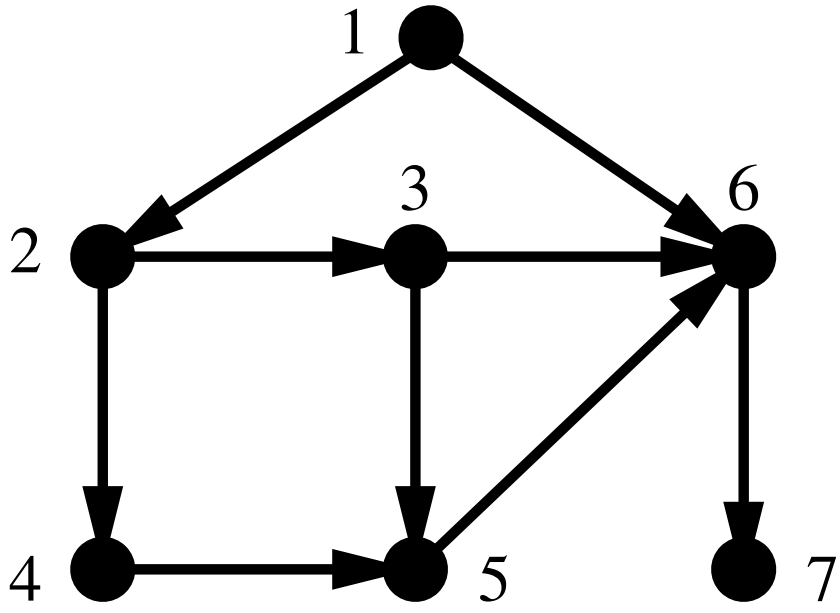
Aciklikus gráfok, Topológikus sorrend

Definíció: Egy irányított gráf **aciklikus**, másnéven **DAG (Directed Acyclic Graph)**, ha nem tartalmaz irányított kört részgráfként.



Aciklikus gráfok, Topológikus sorrend

Definíció: Egy irányított gráf **aciklikus**, másnéven **DAG (Directed Acyclic Graph)**, ha nem tartalmaz irányított kört részgráfként.



Definíció: A G irányított gráf csúcsainak egy $v_1, v_2, v_3, \ldots, v_n$ sorbarendezése **topológikus sorrend** ha minden irányított élre igaz, hogy a kezdőpontja megelőzi a végpontját a sorrendben.

Tétel

Ha G irányított gráf, akkor az alábbi négy tulajdonság ekvivalens:

(i) G aciklikus.

(ii) G tetszőleges mélységi bejárásában nem található visszaél.

(iii) G egy konkrét mélységi bejárásában nem található visszaél.

(iv) G-nek van topológikus sorrendje.

Bizonyítás

(i) $\Longrightarrow$ (ii): Ha egy mélységi bejárásban lenne uv visszaél akkor a visszaél defje szerint a mélységi fában van egy vu irányított út ami az uv éllel együtt irányított kört adna.
(ii) $\Longrightarrow$ (iii): Ha semelyikben sincs akkor nyilván egyben sem.
(iii) $\Longrightarrow$ (iv): Tekintsük azt a mélységi bejárást amiben nincs visszaél. Ez minden csúcshoz hozzárendel egy befejezési számot. Rendezzük a csúcsokat a befejezési számuk szerinti csökkenő sorrendbe! (Ez egyértelmű mivel két különböző csúcs két különböző befejezési számot kap.) A gráfban erre a bejárásra nézve egy uv él faél, előreél vagy keresztél és mindhárom esetén $b(u) > b(v)$ így a fordított sorrendben u sorszáma kisebb mint a v-jé. Ezért ez a sorrend topológikus sorrend.
(iv) $\Longrightarrow$ (i): A topológikus sorrendben minden él kezdőpontja kisebb sorrendű mint a végpontja, így nem lehet a gráfban irányított kör. □

Tétel következményei

Tétel: Ha G irányított gráf, akkor az alábbi négy tulajdonság ekvivalens:

(i) G aciklikus.

(ii) G tetszőleges mélységi bejárásában nem található visszaél.

(iii) G egy konkrét mélységi bejárásában nem található visszaél.

(iv) G-nek van topológikus sorrendje.

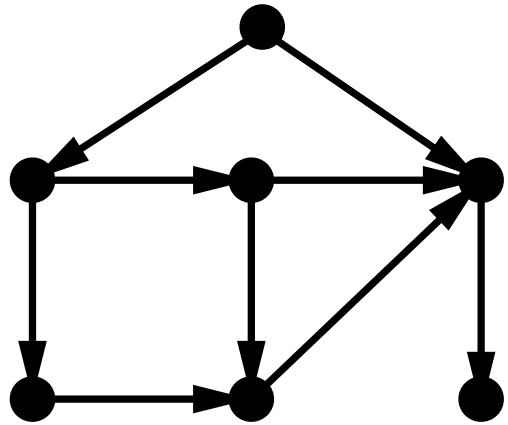
Ezen tétel következményei:

- Csakis aciklikus gráfnak van Topológikus sorrendje.
- Topológikus sorrendet tudunk készíteni a DFS algoritmussal a befejezési számozás megfordításával.
- El tudjuk dönteni egy irányított gráfról, hogy aciklikus-e a DFS algoritmussal: Lefuttatunk egy mélységi keresést és ha nincs visszaél, akkor aciklikus, különben nem az.

Topológikus sorrend keresése máshogy

Állítás

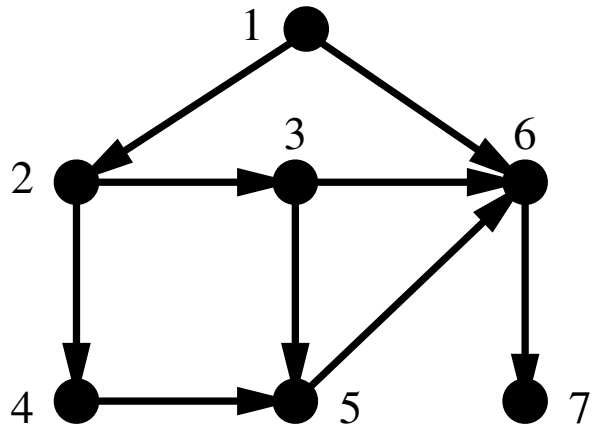
Minden aciklikus gráfban van nyelő és forrás.



Topológikus sorrend keresése máshogy

Állítás

Minden aciklikus gráfban van nyelő és forrás.

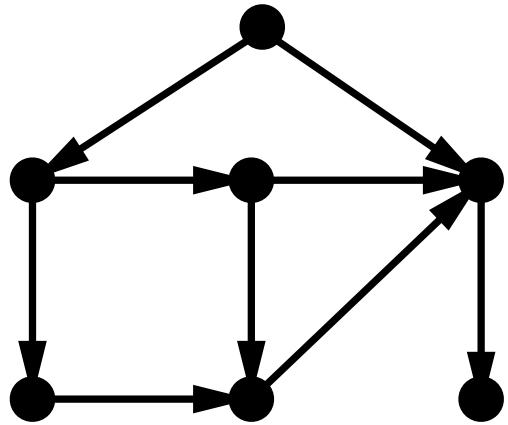


Bizonyítás: DAG-nak van topológikus sorrendje, az első csúcs a sorrendben forrás, az utolsó nyelő. □

Topológikus sorrend keresése máshogy

Állítás

Minden aciklikus gráfban van nyelő és forrás.



Bizonyítás: DAG-nak van topológikus sorrendje, az első csúcs a sorrendben forrás, az utolsó nyelő. □

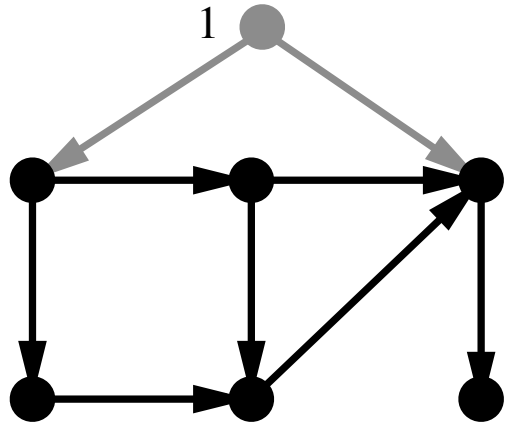
Algoritmus topológikus sorrend keresésre

Legyen a topológikus sorrend első csúcsa az egyik forrás, ezt hagyjuk el, az így kapott gráf továbbra is aciklikus így van forrása, legyen egy ilyen forrás a második csúcs, hagyjuk el, majd így tovább amíg az üres gráfot nem kapjuk.

Topológikus sorrend keresése máshogy

Állítás

Minden aciklikus gráfban van nyelő és forrás.



Bizonyítás: DAG-nak van topológikus sorrendje, az első csúcs a sorrendben forrás, az utolsó nyelő. □

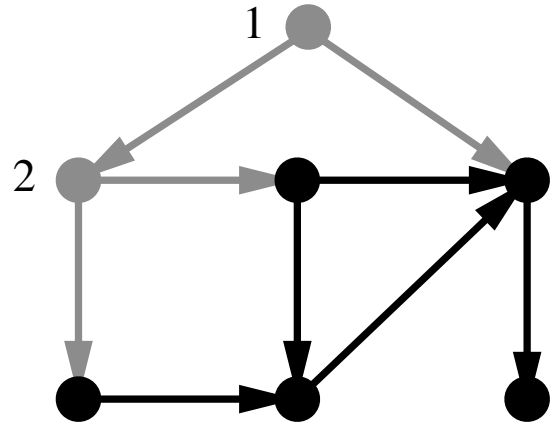
Algoritmus topológikus sorrend keresésre

Legyen a topológikus sorrend első csúcsa az egyik forrás, ezt hagyjuk el, az így kapott gráf továbbra is aciklikus így van forrása, legyen egy ilyen forrás a második csúcs, hagyjuk el, majd így tovább amíg az üres gráfot nem kapjuk.

Topológikus sorrend keresése máshogy

Állítás

Minden aciklikus gráfban van nyelő és forrás.



Bizonyítás: DAG-nak van topológikus sorrendje, az első csúcs a sorrendben forrás, az utolsó nyelő. □

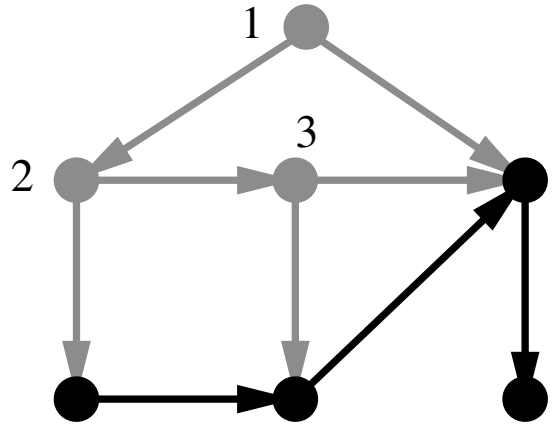
Algoritmus topológikus sorrend keresésre

Legyen a topológikus sorrend első csúcsa az egyik forrás, ezt hagyjuk el, az így kapott gráf továbbra is aciklikus így van forrása, legyen egy ilyen forrás a második csúcs, hagyjuk el, majd így tovább amíg az üres gráfot nem kapjuk.

Topológikus sorrend keresése máshogy

Állítás

Minden aciklikus gráfban van nyelő és forrás.



Bizonyítás: DAG-nak van topológikus sorrendje, az első csúcs a sorrendben forrás, az utolsó nyelő. □

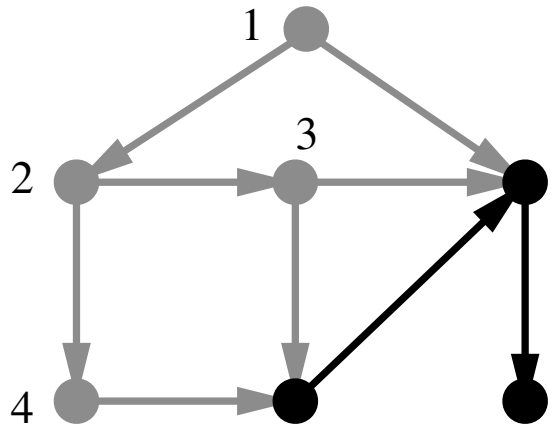
Algoritmus topológikus sorrend keresésre

Legyen a topológikus sorrend első csúcsa az egyik forrás, ezt hagyjuk el, az így kapott gráf továbbra is aciklikus így van forrása, legyen egy ilyen forrás a második csúcs, hagyjuk el, majd így tovább amíg az üres gráfot nem kapjuk.

Topológikus sorrend keresése máshogy

Állítás

Minden aciklikus gráfban van nyelő és forrás.



Bizonyítás: DAG-nak van topológikus sorrendje, az első csúcs a sorrendben forrás, az utolsó nyelő. □

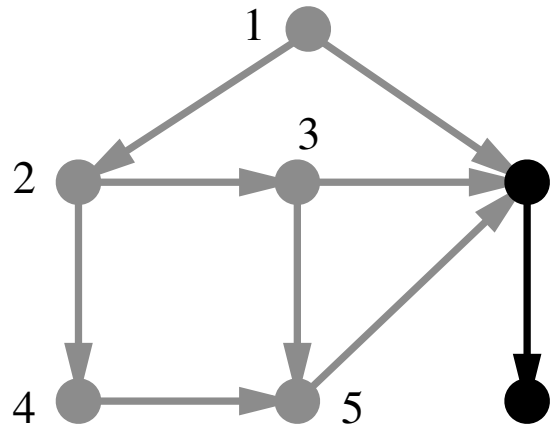
Algoritmus topológikus sorrend keresésre

Legyen a topológikus sorrend első csúcsa az egyik forrás, ezt hagyjuk el, az így kapott gráf továbbra is aciklikus így van forrása, legyen egy ilyen forrás a második csúcs, hagyjuk el, majd így tovább amíg az üres gráfot nem kapjuk.

Topológikus sorrend keresése máshogy

Állítás

Minden aciklikus gráfban van nyelő és forrás.



Bizonyítás: DAG-nak van topológikus sorrendje, az első csúcs a sorrendben forrás, az utolsó nyelő. □

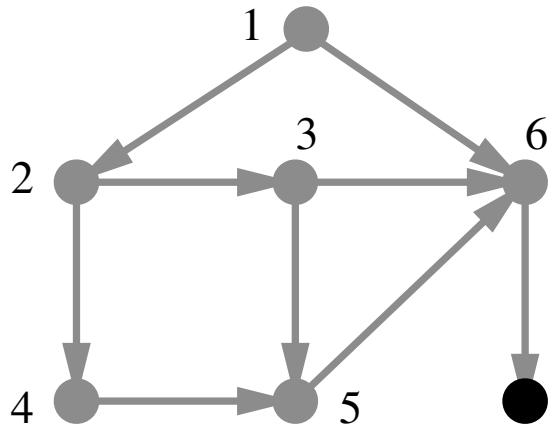
Algoritmus topológikus sorrend keresésre

Legyen a topológikus sorrend első csúcsa az egyik forrás, ezt hagyjuk el, az így kapott gráf továbbra is aciklikus így van forrása, legyen egy ilyen forrás a második csúcs, hagyjuk el, majd így tovább amíg az üres gráfot nem kapjuk.

Topológikus sorrend keresése máshogy

Állítás

Minden aciklikus gráfban van nyelő és forrás.



Bizonyítás: DAG-nak van topológikus sorrendje, az első csúcs a sorrendben forrás, az utolsó nyelő. □

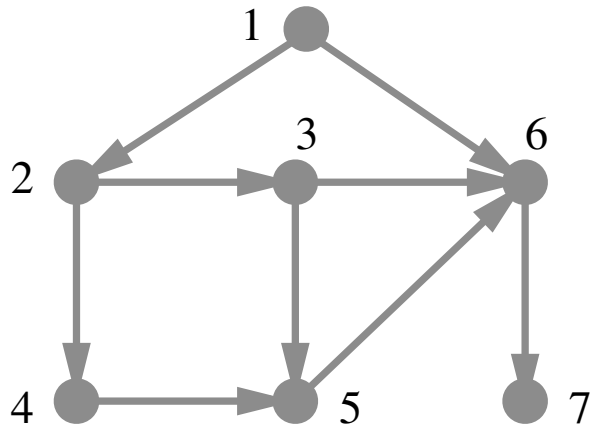
Algoritmus topológikus sorrend keresésre

Legyen a topológikus sorrend első csúcsa az egyik forrás, ezt hagyjuk el, az így kapott gráf továbbra is aciklikus így van forrása, legyen egy ilyen forrás a második csúcs, hagyjuk el, majd így tovább amíg az üres gráfot nem kapjuk.

Topológikus sorrend keresése máshogy

Állítás

Minden aciklikus gráfban van nyelő és forrás.



Bizonyítás: DAG-nak van topológikus sorrendje, az első csúcs a sorrendben forrás, az utolsó nyelő. □

Algoritmus topológikus sorrend keresésre

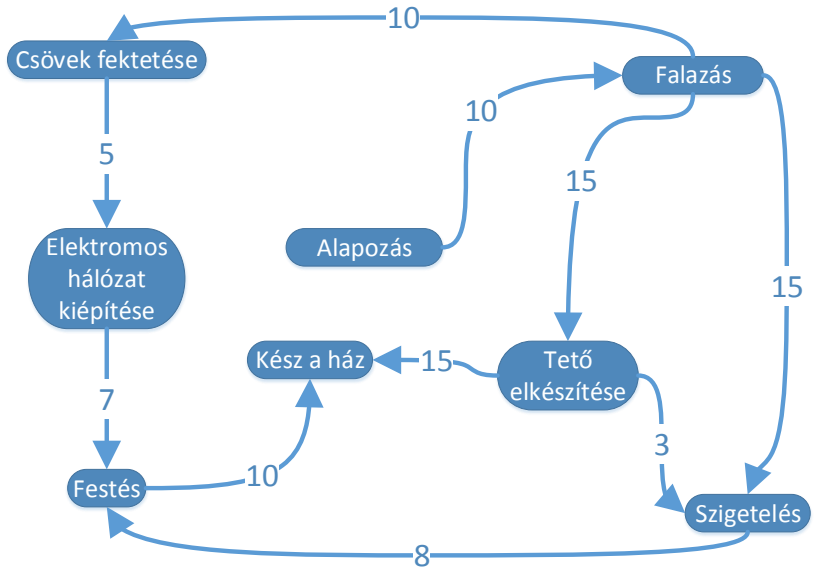
Legyen a topológikus sorrend első csúcsa az egyik forrás, ezt hagyjuk el, az így kapott gráf továbbra is aciklikus így van forrása, legyen egy ilyen forrás a második csúcs, hagyjuk el, majd így tovább amíg az üres gráfot nem kapjuk.

PERT, Program Evaluation and Review Technique

Feladat:

Adott egy projekt ami részfeladatokból (tevékenységek) áll. Ezeket mind el kell végeznünk. Azonban néhány részfeladat megfelelő szintig tartó elvégzése követelménye annak, hogy egy másik feladatot elkezdjünk.

Ezt a következő irányított gráffal kódoljuk: Csúcsai a tevékenységek és a t hosszú $(\overrightarrow{A, B})$ irányított él pontosan akkor eleme az élhalmaznak ha a B tevékenységet legkorábbán az A tevékenység kezdőidpontja után t idővel kezdhetjük el.



Kérdések:

- Mikor lehet egyáltalán végrehajtani a feladatokat?

Kérdések:

- Mikor lehet egyáltalán végrehajtani a feladatokat? **Válasz: Ha a gráf aciklikus!**

Kérdések:

- Mikor lehet egyáltalán végrehajtani a feladatokat? **Válasz: Ha a gráf aciklikus!**
- Mikor tudjuk leghamarabb elkezdeni a legutolsó tevékenységet, azaz mennyi az átfutási idő?

Kérdések:

- Mikor lehet egyáltalán végrehajtani a feladatokat? **Válasz: Ha a gráf aciklikus!**
- Mikor tudjuk leghamarabb elkezdeni a legutolsó tevékenységet, azaz mennyi az átfutási idő?**Válasz: A megadott irányított gráfban a leghoszabb irányított út hossza.**

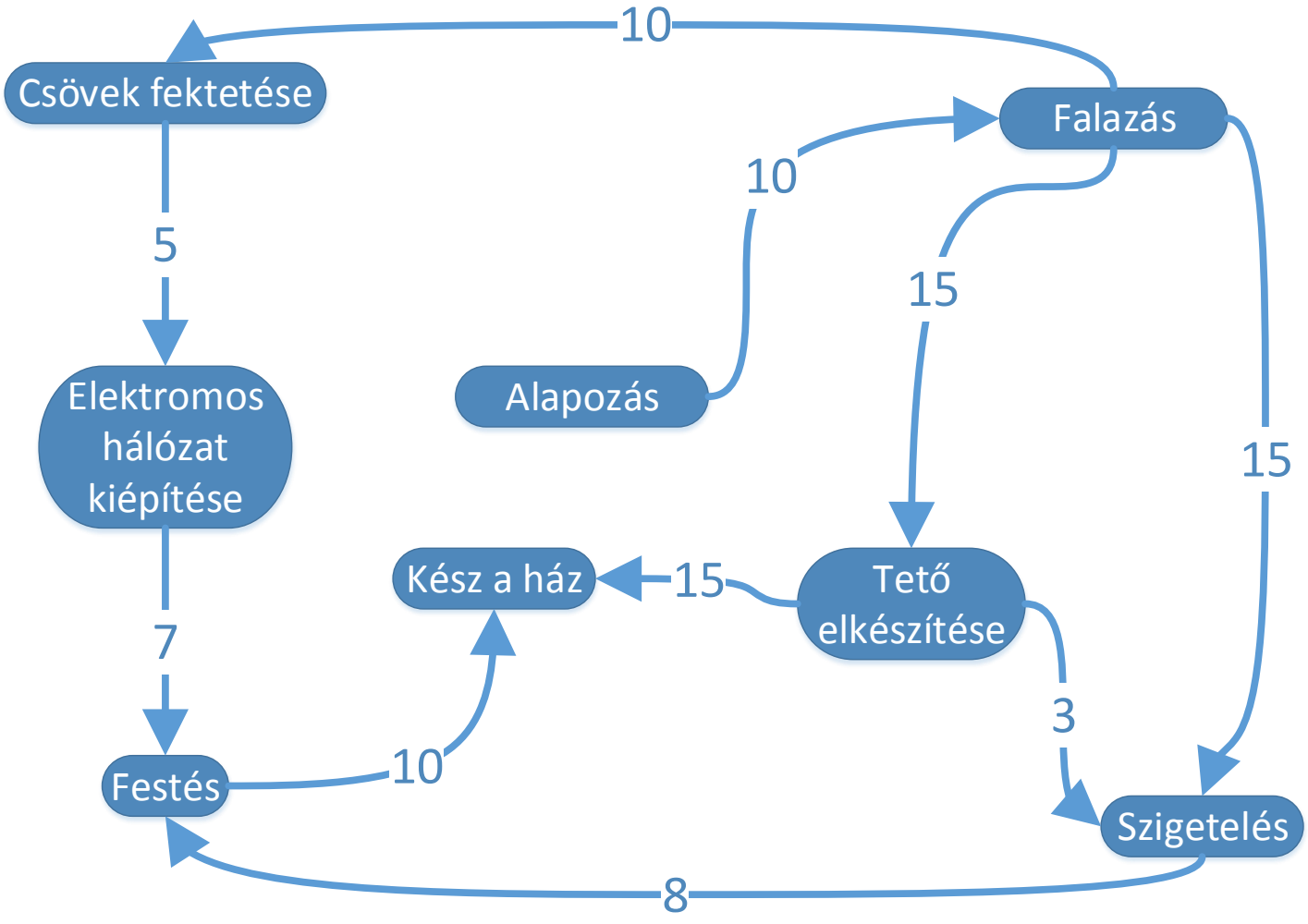
Kérdések:

- Mikor lehet egyáltalán végrehajtani a feladatokat? **Válasz: Ha a gráf aciklikus!**
- Mikor tudjuk leghamarabb elkezdeni a legutolsó tevékenységet, azaz mennyi az átfutási idő?**Válasz: A megadott irányított gráfban a leghoszabb irányított út hossza.**
- Mikor tudjuk elkezdeni leghamarabb a különböző tevékenységeket? **Válasz: A megadott irányított gráfban az adott tevékenységbe vezető leghoszabb irányított út hossza.**

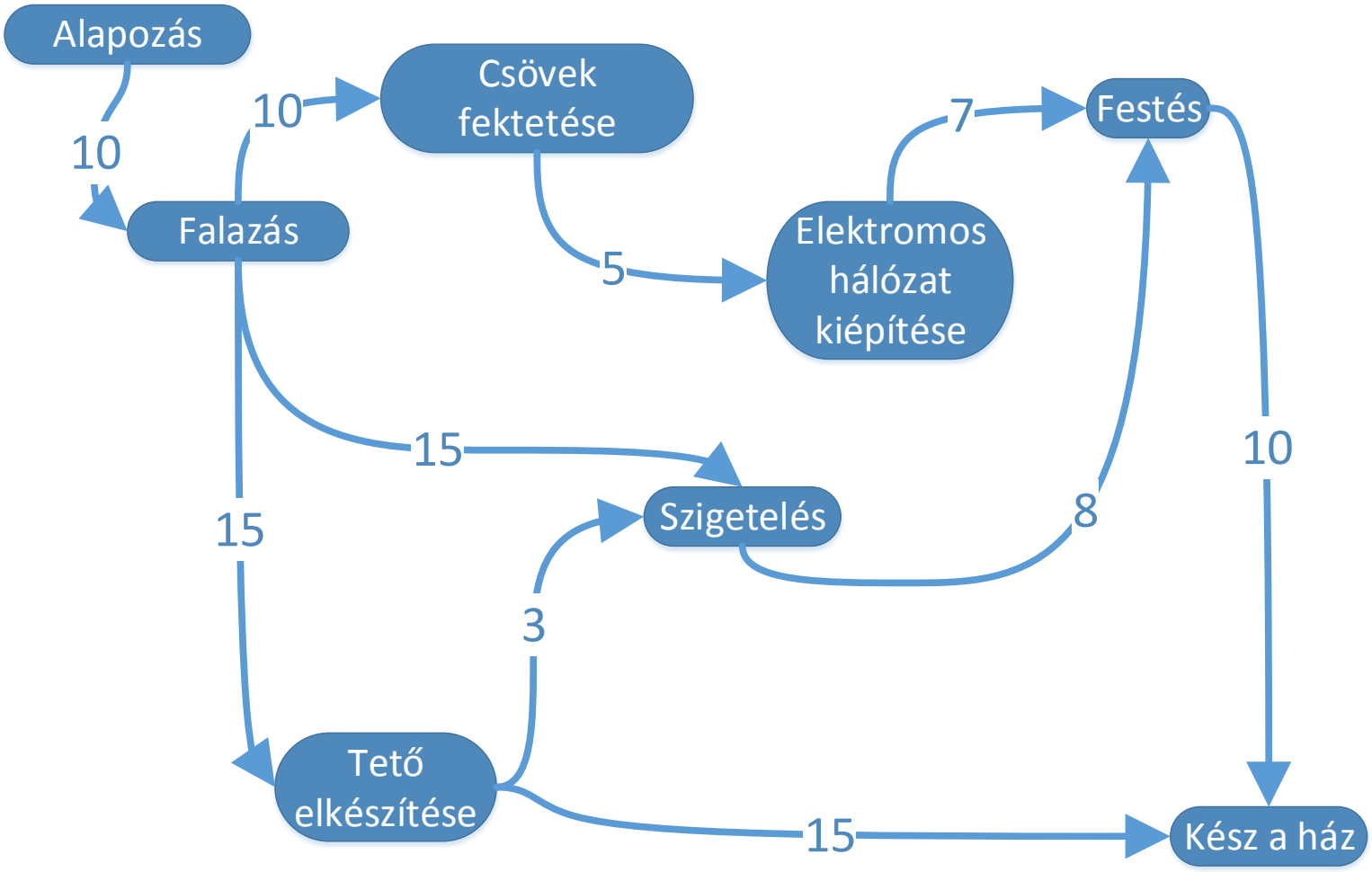
Kérdések:

- Mikor lehet egyáltalán végrehajtani a feladatokat? **Válasz: Ha a gráf aciklikus!**
- Mikor tudjuk leghamarabb elkezdeni a legutolsó tevékenységet, azaz mennyi az átfutási idő?**Válasz: A megadott irányított gráfban a leghoszabb irányított út hossza.**
- Mikor tudjuk elkezdeni leghamarabb a különböző tevékenységeket? **Válasz: A megadott irányított gráfban az adott tevékenységbe vezető leghoszabb irányított út hossza.**
- Melyek azok a tevékenységek, melyek csúszása esestén a projekt is csúszik, azaz tolódik a legutoljára elkezdett tevékenység kezdési ideje?

PERT példa

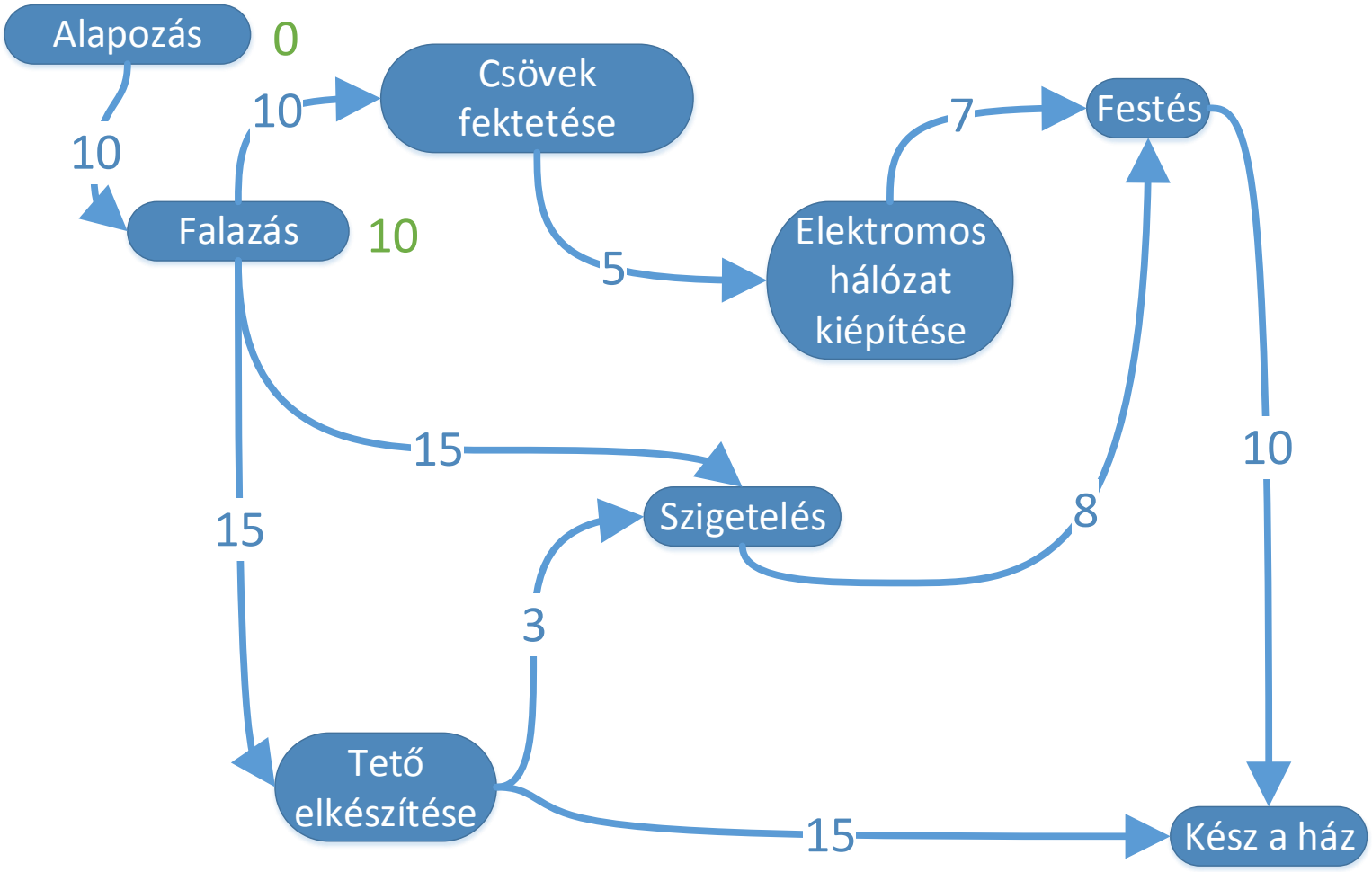

10
Csövek fektetése
Falazás
10
5
15
Elektromos hálózat kiépítése
Alapozás
15
Kész a ház
15
Tető elkészítése
7
3
Festés
10
Szigetelés
8

PERT példa



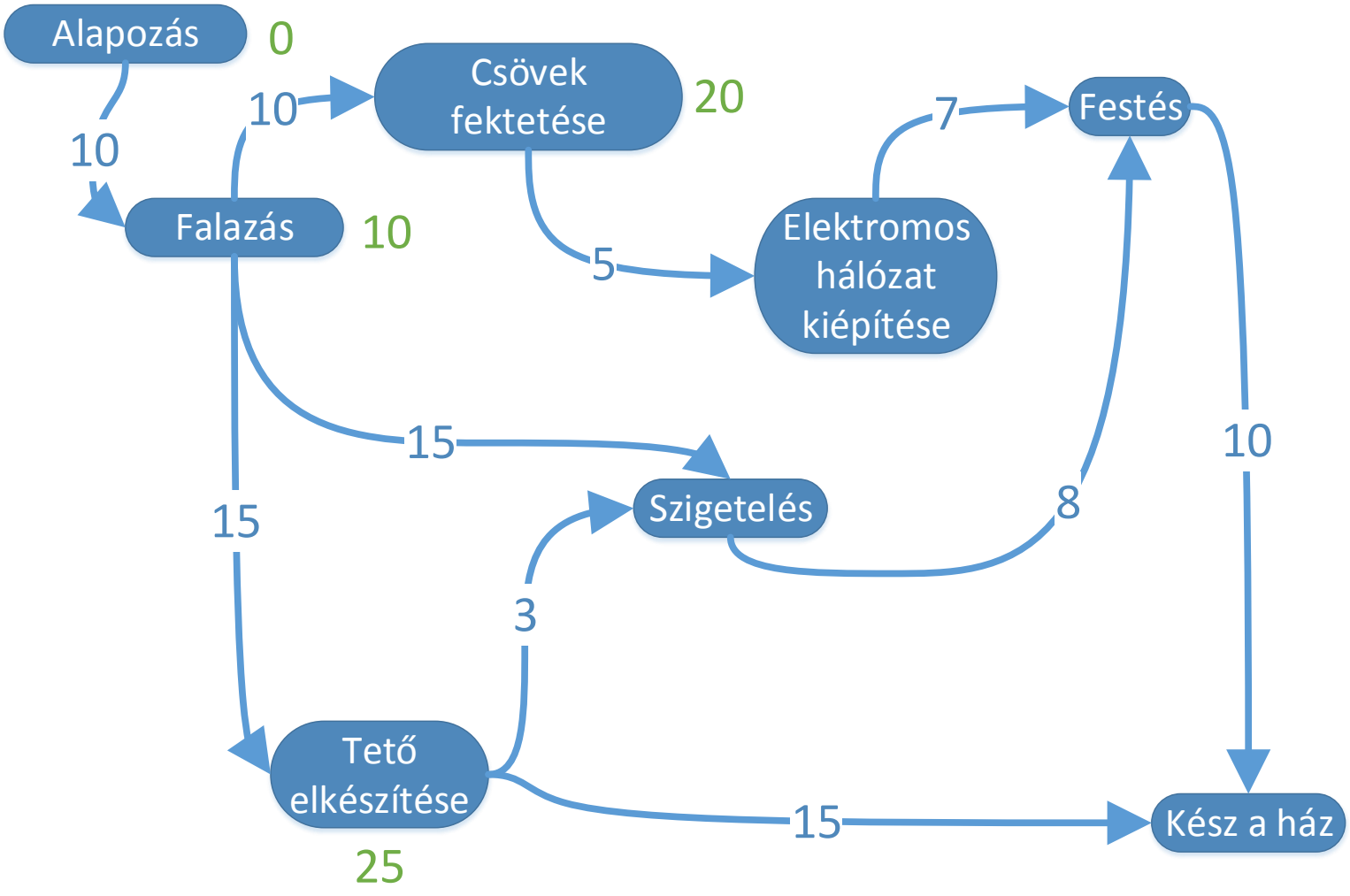
A csúcsok bal→jobb sorrendje a topológikus sorrend.

PERT példa



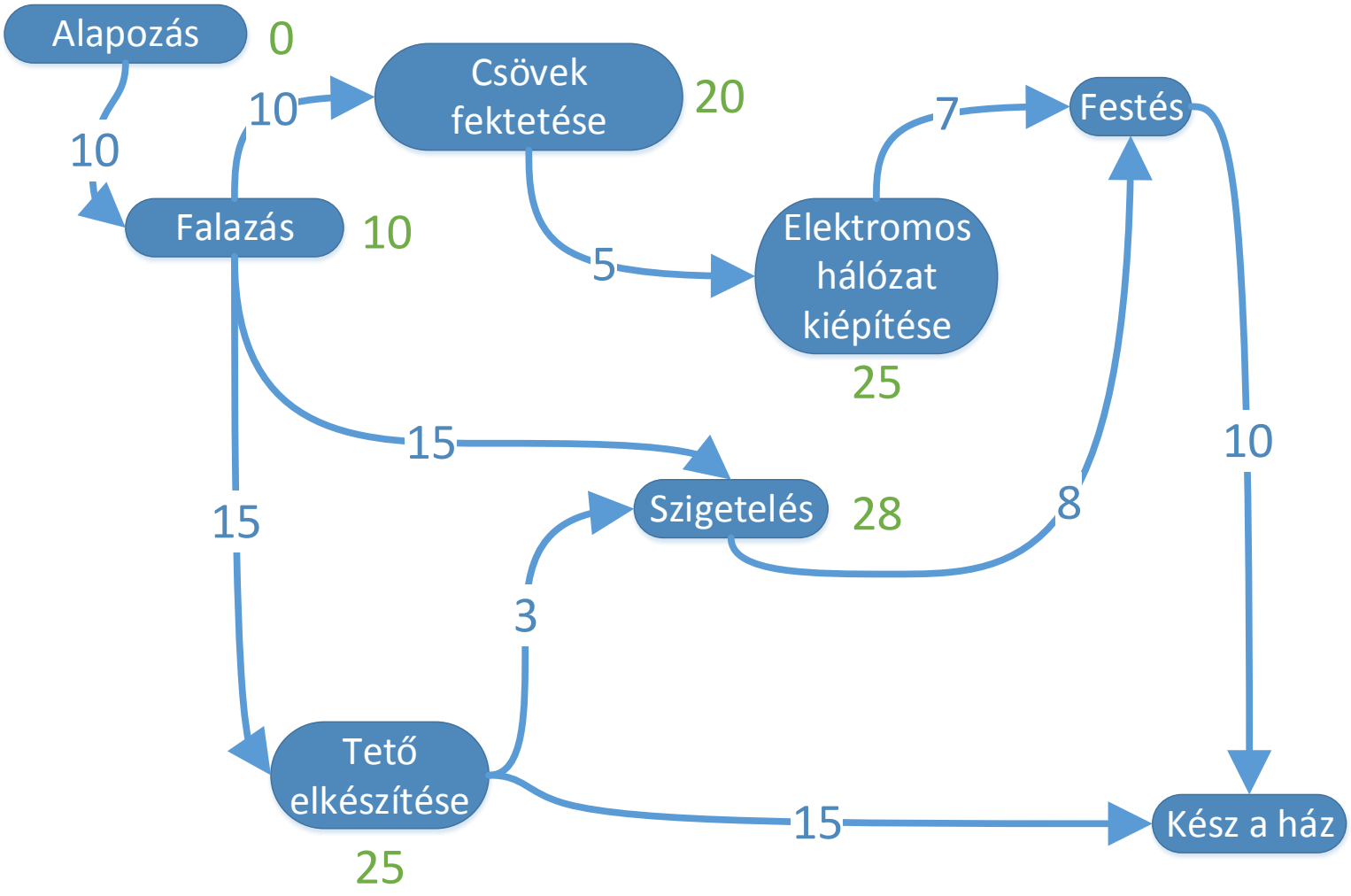
A csúcsok bal→jobb sorrendje a topológikus sorrend.

PERT példa



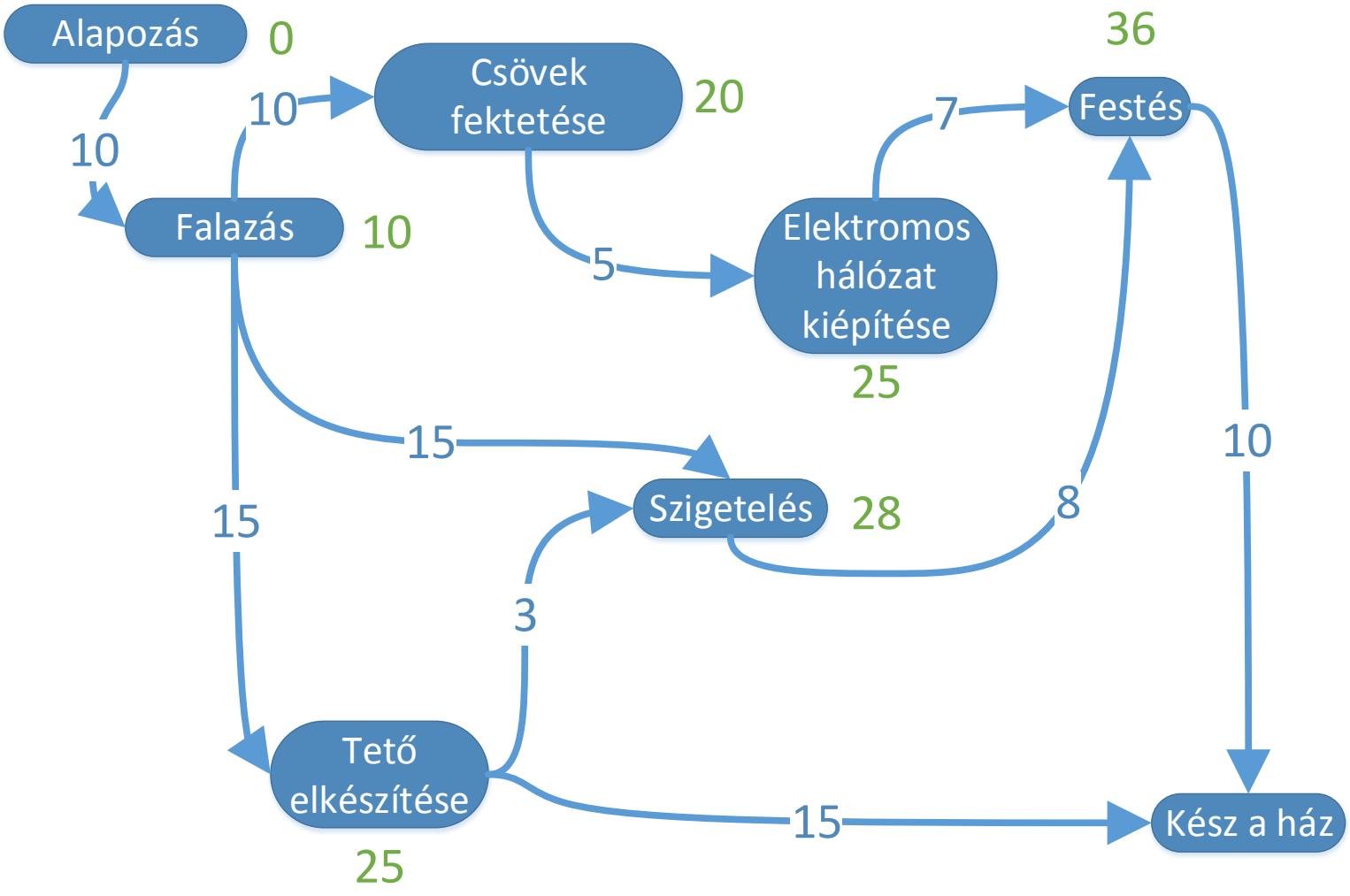
A csúcsok bal→jobb sorrendje a topológikus sorrend.

PERT példa



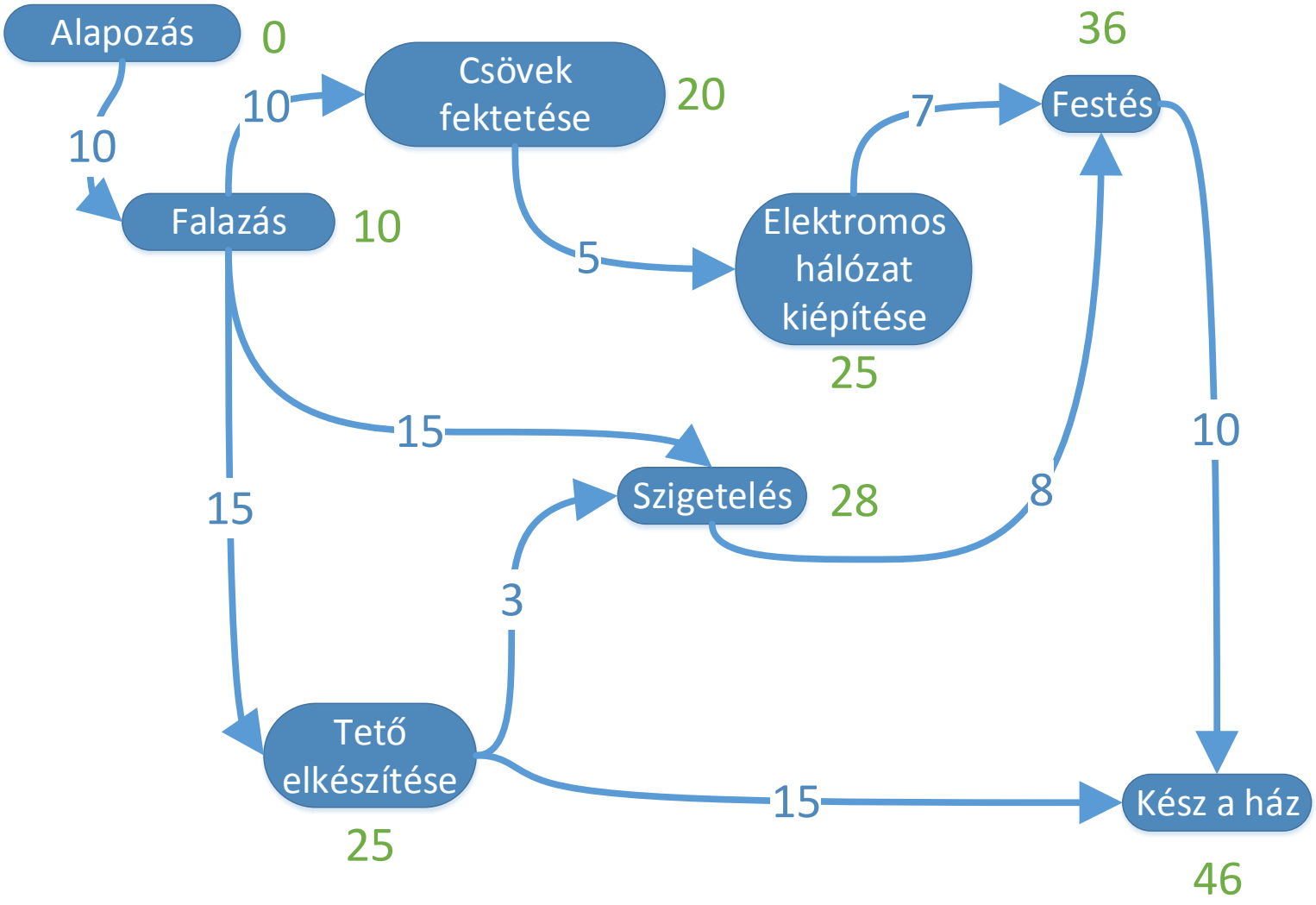
A csúcsok bal→jobb sorrendje a topológikus sorrend.

PERT példa



A csúcsok bal→jobb sorrendje a topológikus sorrend.

PERT példa



A csúcsok bal→jobb sorrendje a topológikus sorrend.

Algoritmus leghoszabb út keresésére DAGban:

Input: Egy $\overrightarrow{G}$, a feladatból készített DAG és egy l nem negatív élhosszfüggvény, ahol $\overrightarrow{G}$-ben pontosan egy forrás van. (Ha több forrás is lenne, akkor kiegészítjük a gráfot egy új csúccsal amiből irányított 0 hosszú éleket húzunk minden forrásba. Így az újonnan felvett csúcs az egyetlen forrás.)
Elkészítjük a $\overrightarrow{G}$ egy topológikus sorrendjét. Ilyen van mert $\overrightarrow{G}$ DAG. Legyen ez a topológikus sorrend $v_1, v_2, v_3, \ldots, v_n$ és legyen $k(v_1) := 0$.
A v_i-ken a topológikus sorrend szerint végigmenve:

$$k(v_i) := \max_{\overrightarrow{(v_j,v_i)} \in E(G)} (k(v_j) + l(\overrightarrow{(v_j, v_i)}))$$

Output: $k : V(\overrightarrow{G}) \to \mathbb{R}$ függvény, amely megmondja minden csúcsra a benne végetérő leghoszabb irányított út hosszát.
Megjegyzés: Mivel a csúcsok sorrendezése egy topológikus sorrend, ezért az értékadás jobb oldalán álló $k(v_j)$ értékek már mind ki vannak számolva, hiszen csak kisebb sorrendű csúcsból megy él nagyobb sorrendűbe.

Kritikus tevékenységek, kritikus út

A PERT feladat megoldása során az előző algoritmus futtatjuk és a $\overrightarrow{G}$ gráfban az l élhosszfüggvényre nézve megkapjuk a leghoszabb út hosszát.

Kritikus tevékenységek, kritikus út

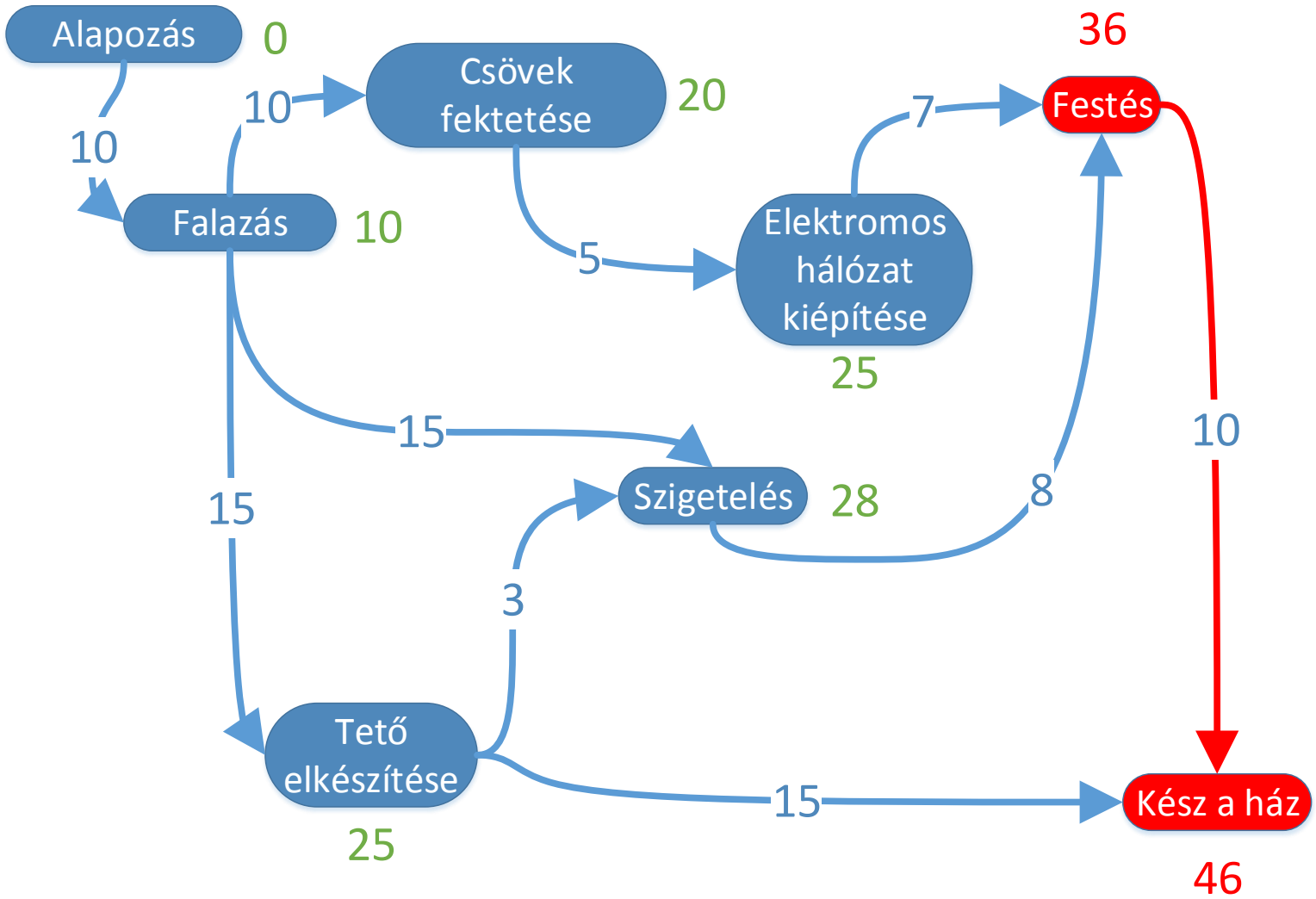
A PERT feladat megoldása során az előző algoritmus futtatjuk és a $\overrightarrow{G}$ gráfban az l élhosszfüggvényre nézve megkapjuk a leghoszabb út hosszát.
Ez a mennyiség pontosan akkor fog nőni ha egy olyan él hosszát növelem meg aki beletartozik valamelyik leghoszabb útba. Ez motíválja az alábbi definíciót:

Definíció: Ha egy PERT feladatot a $\overrightarrow{G}$ gráf és l élhosszfüggvény ír le, akkor a $\overrightarrow{G}$ gráfnak az l élhosszfüggvényre nézve vett leghoszabb útját **kritikus** útnak nevezzük.

Definíció: Egy élet vagy egy csúcsot (tevékenységet) **kritikusnak** nevezünk ha a kritikus út egy éle vagy csúcsa.

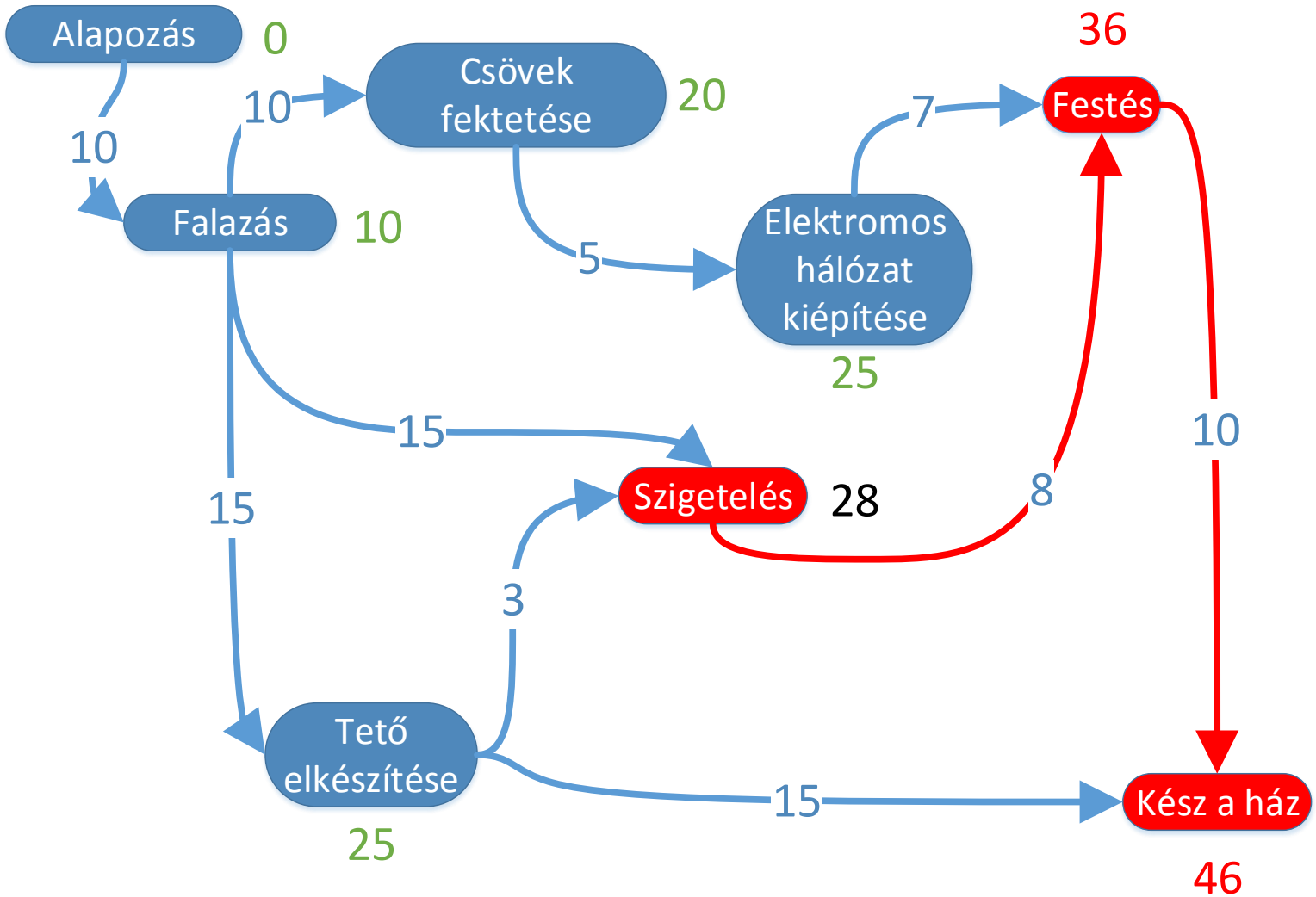
Megjegyzés: A PERT feladatban a kritikus tevékenységek pontosan azok, melyeket ha a lehetséges legkorábbi kezdési időben nem kezdünk el, akkor a projekt befejezése későbbre csúszik.

Kritikus tevékenységek megkeresése



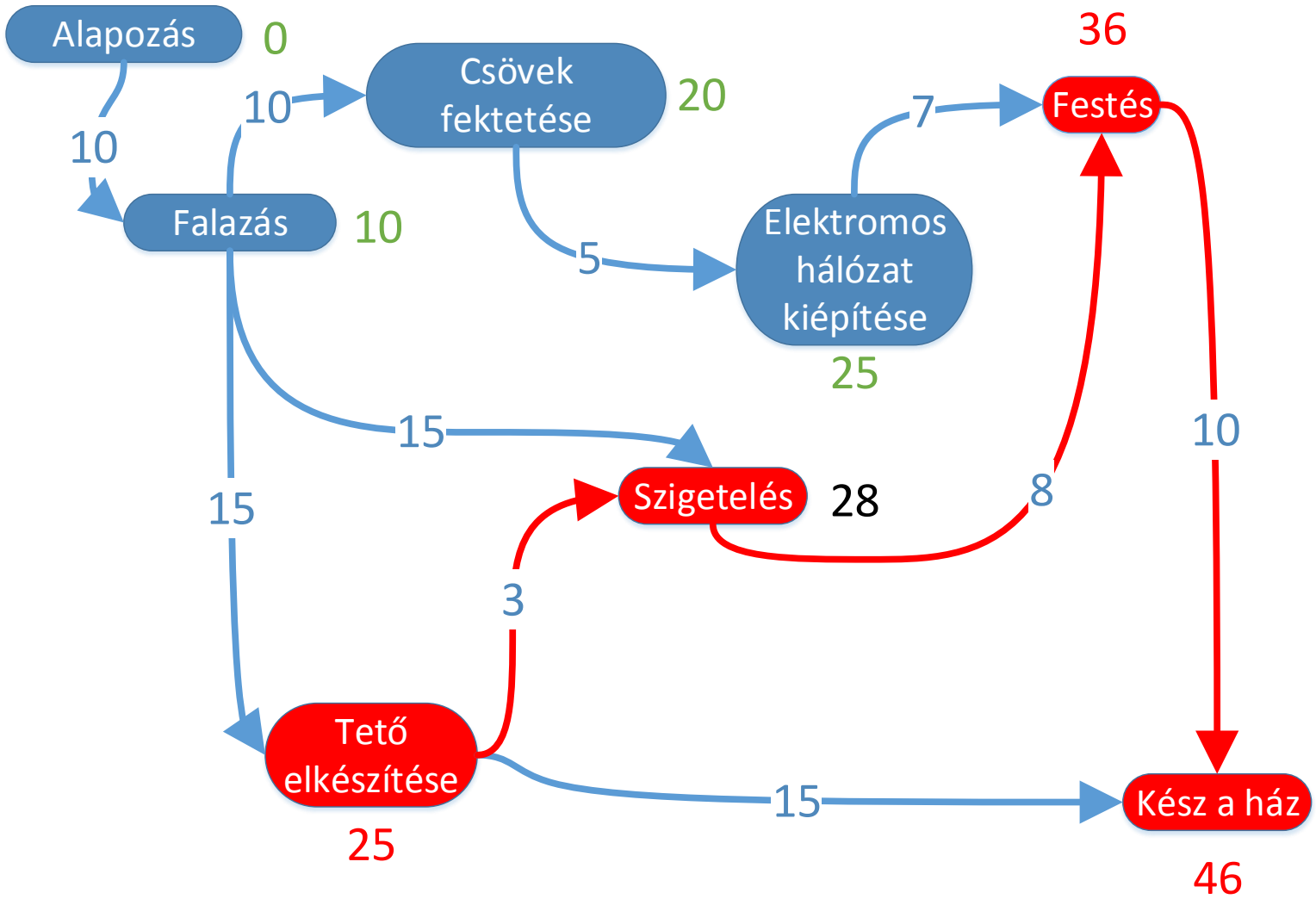
A nyelő kritikus. Topológikus sorrend mentén visszafele haladunk. Ha a v_i csúcs kritikus és $k(v_i) = k(v_j) + l(\overrightarrow{(v_j, v_i)})$, akkor a v_j csúcsot és $\overrightarrow{(v_j, v_i)}$ élet is kritikusnak nyílvánítjuk.

Kritikus tevékenységek megkeresése



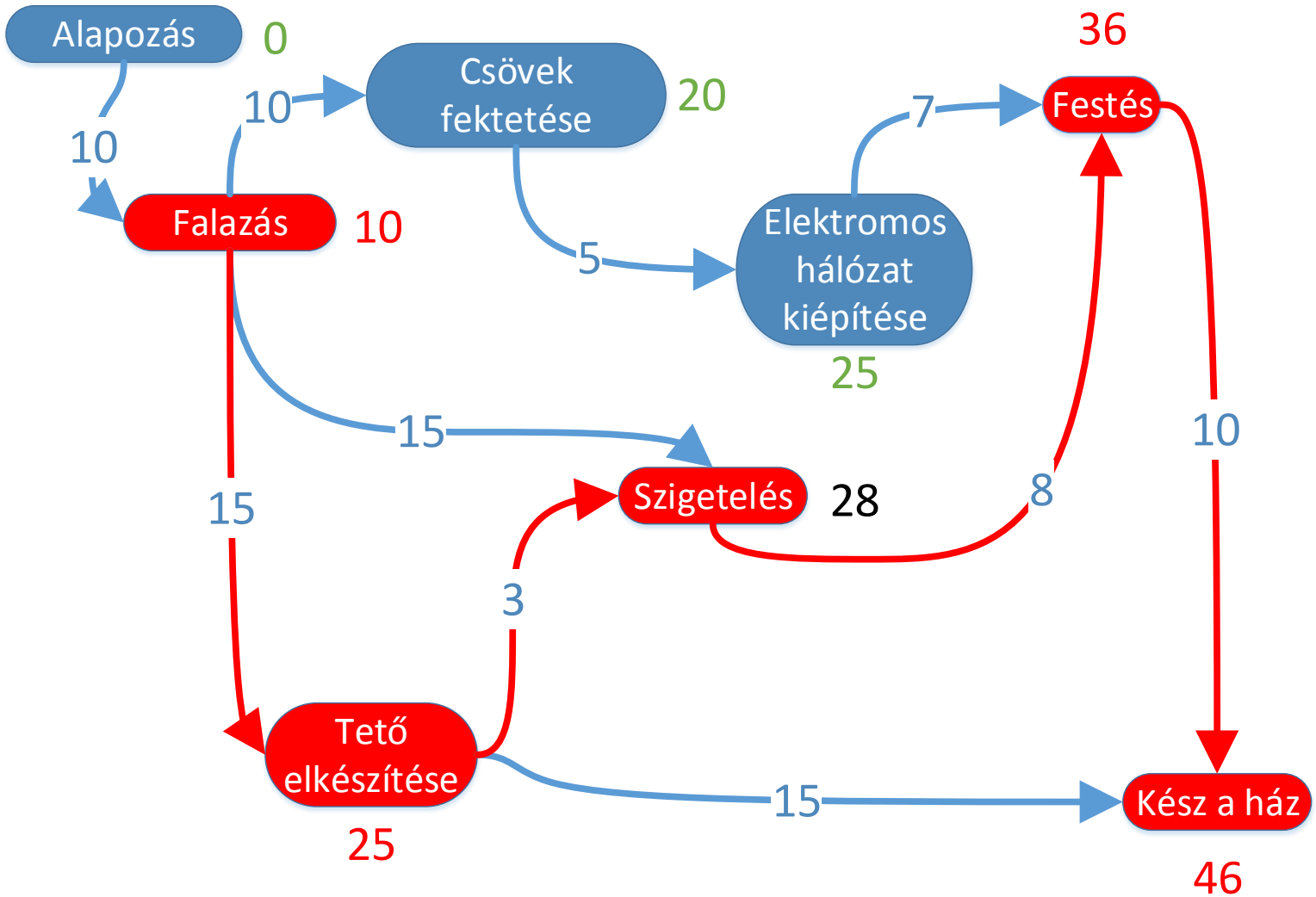
A nyelő kritikus. Topológikus sorrend mentén visszafele haladunk. Ha a v_i csúcs kritikus és $k(v_i) = k(v_j) + l(\overrightarrow{(v_j, v_i)})$, akkor a v_j csúcsot és $\overrightarrow{(v_j, v_i)}$ élet is kritikusnak nyílvánítjuk.

Kritikus tevékenységek megkeresése



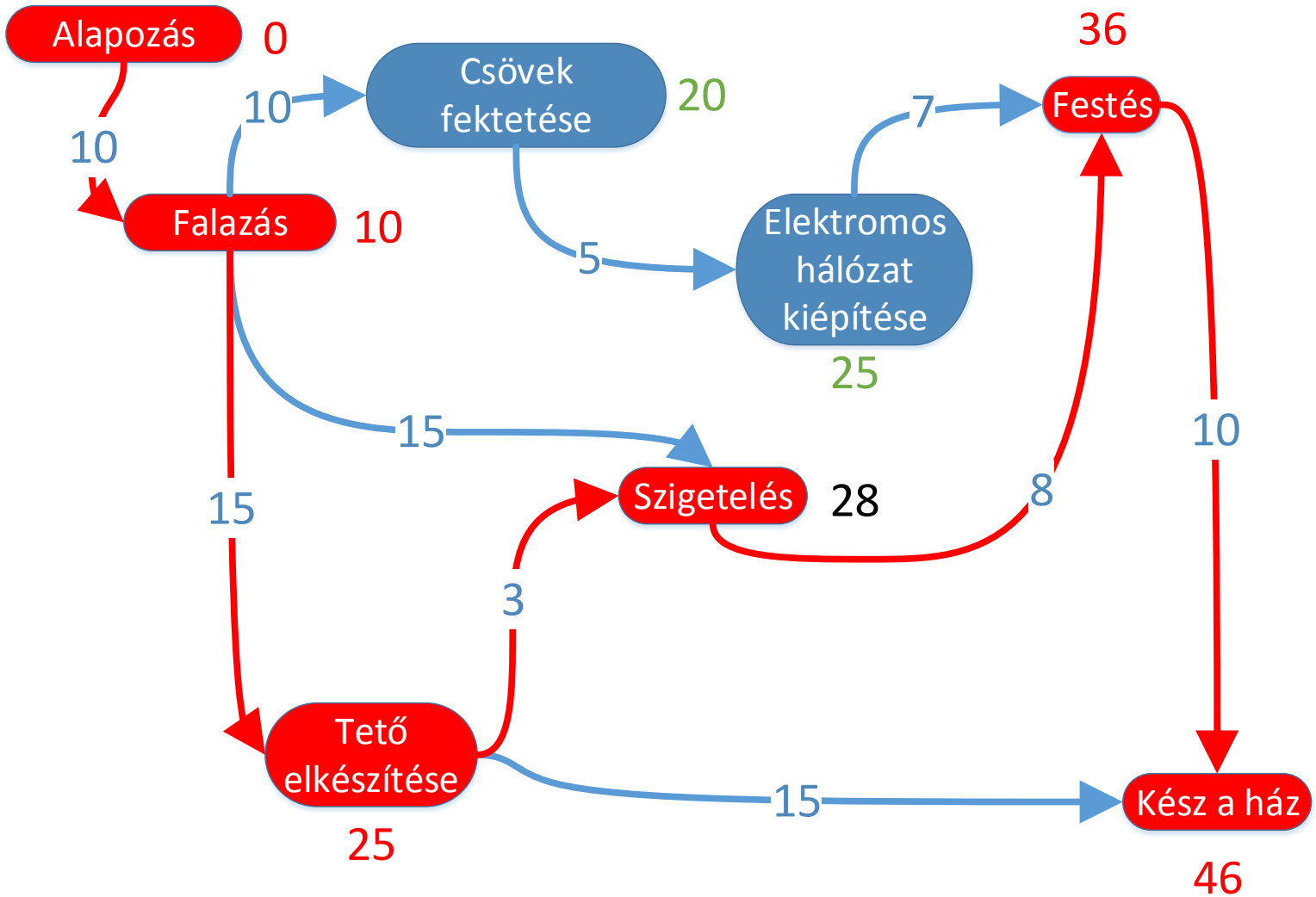
A nyelő kritikus. Topológikus sorrend mentén visszafele haladunk. Ha a v_i csúcs kritikus és $k(v_i) = k(v_j) + l(\overrightarrow{(v_j, v_i)})$, akkor a v_j csúcsot és $\overrightarrow{(v_j, v_i)}$ élet is kritikusnak nyílvánítjuk.

Kritikus tevékenységek megkeresése



A nyelő kritikus. Topológikus sorrend mentén visszafele haladunk. Ha a v_i csúcs kritikus és $k(v_i) = k(v_j) + l(\overrightarrow{(v_j, v_i)})$, akkor a v_j csúcsot és $\overrightarrow{(v_j, v_i)}$ élet is kritikusnak nyílvánítjuk.

Kritikus tevékenységek megkeresése



A nyelő kritikus. Topológikus sorrend mentén visszafele haladunk. Ha a v_i csúcs kritikus és $k(v_i) = k(v_j) + l(\overrightarrow{(v_j, v_i)})$, akkor a v_j csúcsot és $\overrightarrow{(v_j, v_i)}$ élet is kritikusnak nyílvánítjuk.

PERT algoritmus kritikus utak és tevékenységek meghatározásával

Input: Egy $\overrightarrow{G}$, a feladatból készített DAG és egy l nemnegatív élhosszfüggvény, ahol $\overrightarrow{G}$-ben pontosan egy forrás és egy nyelő van. (Ha több lenne, akkor új forrás illetve nyelő.)
Elkészítjük a $\overrightarrow{G}$ egy topológikus sorrendjét. Ilyen van mert $\overrightarrow{G}$ DAG. Legyen ez a topológikus sorrend $v_1, v_2, v_3, \ldots, v_n$ és legyen $k(v_1) := 0$.
A v_i-ken a topológikus sorrend szerint végigmenve:

$$k(v_i) := \max_{\overrightarrow{(v_j,v_i)} \in E(G)} (k(v_j) + l(\overrightarrow{(v_j, v_i)}))$$

Ezután a nyelőt kritikusnak nyílvánítjuk. A v_i-ken a topológikus sorrend fordítottja szerint végigmegyünk. Ha a v_i csúcs kritikus és $k(v_i) = k(v_j) + l(\overrightarrow{(v_j, v_i)})$, akkor a v_j csúcsot és $\overrightarrow{(v_j, v_i)}$ élet is kritikusnak nyílvánítjuk.
Output: $k : V(\overrightarrow{G}) \to \mathbb{R}$ legkorábbi kezdési idők, kritikus utak és kritikus tevékenységek.