A számítástudomány alapjai

Gráfbejárások és legrövidebb utak

2022. szeptember 20.

Munkaterv

Munkaterv

- Irányítatlan gráfok esetén a csúcsok közötti elérhetőséget a komponensek ill. feszítő erdő segítségével tudjuk kompakt módon leírni. Irányított gráfokban ez a struktúra ennél jóval bonyolultabb.

Munkaterv

- Irányítatlan gráfok esetén a csúcsok közötti elérhetőséget a komponensek ill. feszítő erdő segítségével tudjuk kompakt módon leírni. Irányított gráfokban ez a struktúra ennél jóval bonyolultabb.
- Irányított gráfban fogjuk azt vizsgálni, hogy egy megadott csúcsból a gráf milyen más csúcsai érhetők el, majd az elérhető csúcsokba keresünk legrövidebb utat.

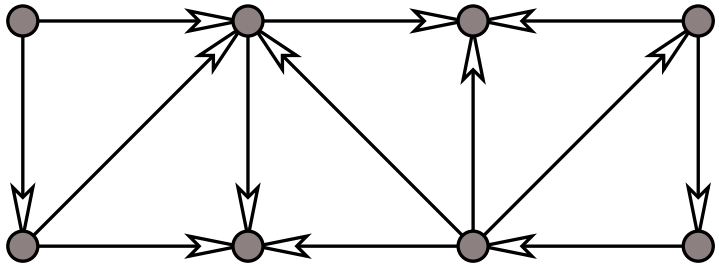
Munkaterv

- Irányítatlan gráfok esetén a csúcsok közötti elérhetőséget a komponensek ill. feszítő erdő segítségével tudjuk kompakt módon leírni. Irányított gráfokban ez a struktúra ennél jóval bonyolultabb.
- Irányított gráfban fogjuk azt vizsgálni, hogy egy megadott csúcsból a gráf milyen más csúcsai érhetők el, majd az elérhető csúcsokba keresünk legrövidebb utat.
- Ennek során (többek között) egy újfajta módszert látunk a feszítőfa (ill. feszítő erdő) keresésére.

Munkaterv

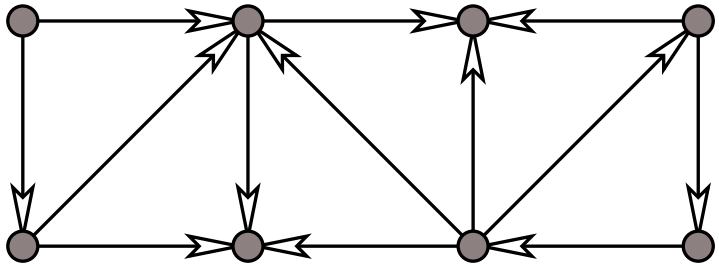
- Irányítatlan gráfok esetén a csúcsok közötti elérhetőséget a komponensek ill. feszítő erdő segítségével tudjuk kompakt módon leírni. Irányított gráfokban ez a struktúra ennél jóval bonyolultabb.
- Irányított gráfban fogjuk azt vizsgálni, hogy egy megadott csúcsból a gráf milyen más csúcsai érhetők el, majd az elérhető csúcsokba keresünk legrövidebb utat.
- Ennek során (többek között) egy újfajta módszert látunk a feszítőfa (ill. feszítő erdő) keresésére.
- A továbbiakban irányított gráfokkal foglalkozunk: irányítatlan gráf esetén arra az irányított gráfra gondolunk, amit az irányítatlanból az élek oda-vissza irányításával kapunk. Ezzel a módszerrel az irányított gráfra kapott eredmények az irányítatlan esetre is könnyen átültethetők.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\longrightarrow$ elért $\longrightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\longrightarrow$ elért $\longrightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

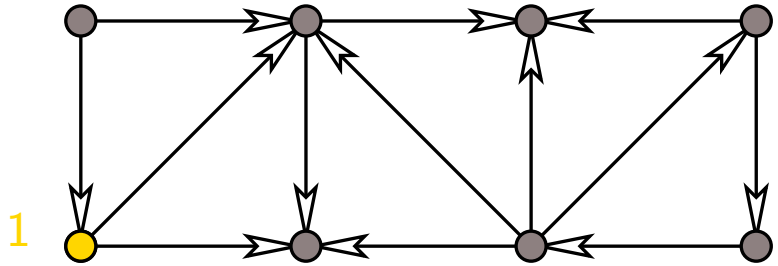
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\longrightarrow$ elért $\longrightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

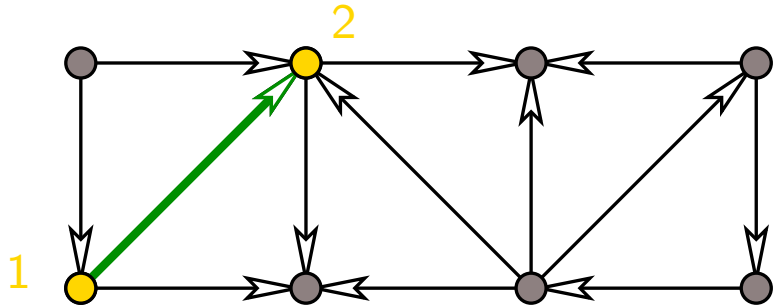
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\longrightarrow$ elért $\longrightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

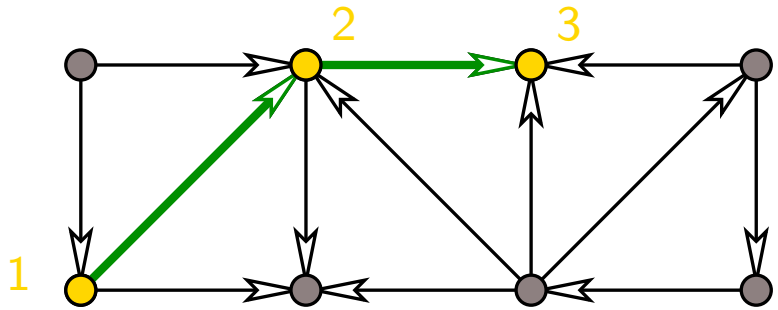
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\longrightarrow$ elért $\longrightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

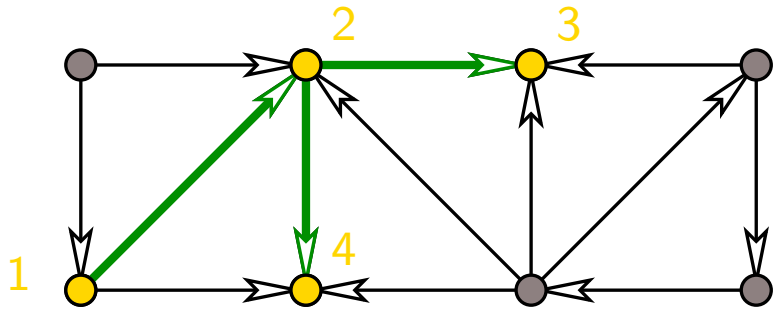
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\longrightarrow$ elért $\longrightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

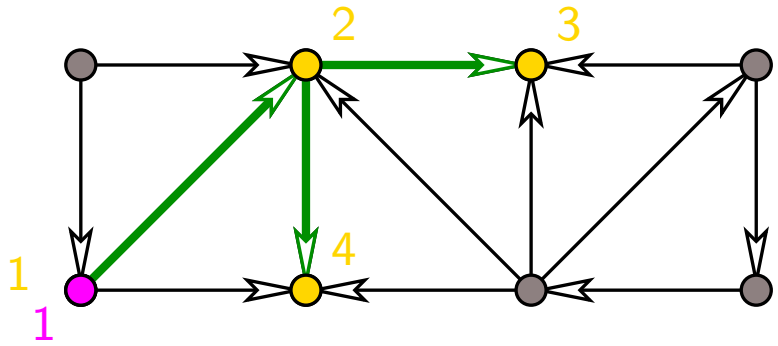
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\longrightarrow$ elért $\longrightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

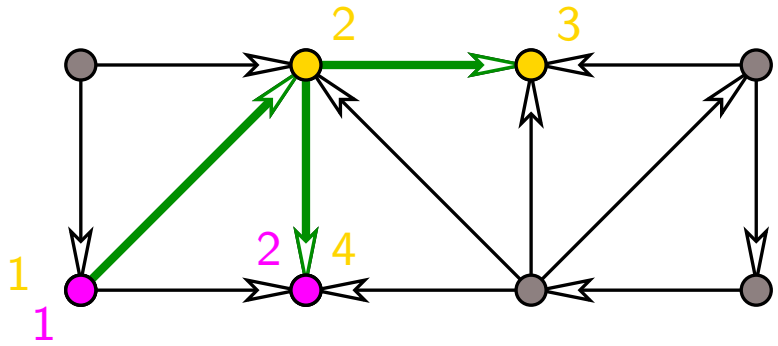
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen $\longrightarrow$ elért $\longrightarrow$ befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

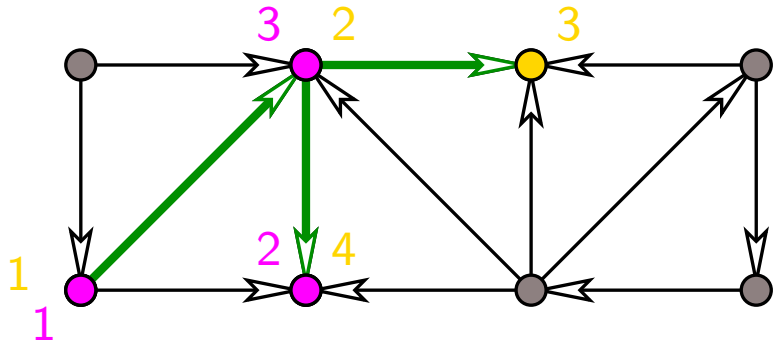
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

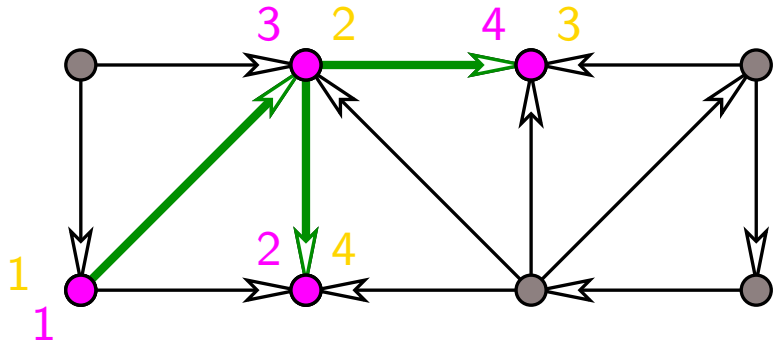
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

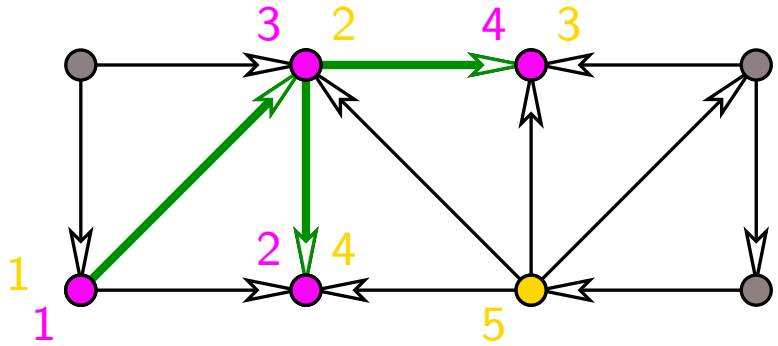
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

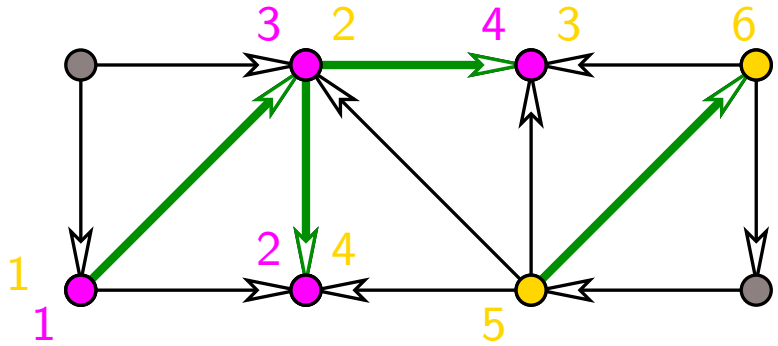
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

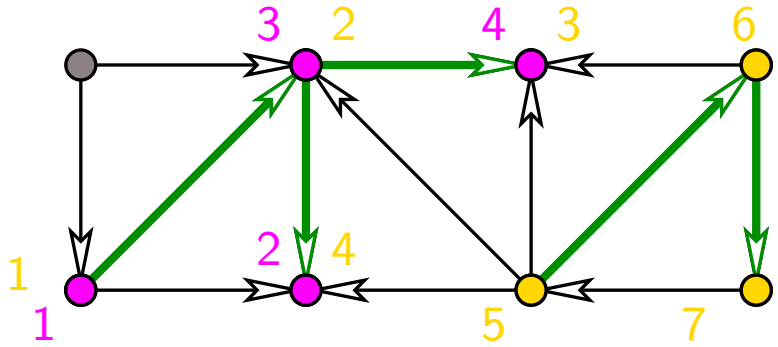
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

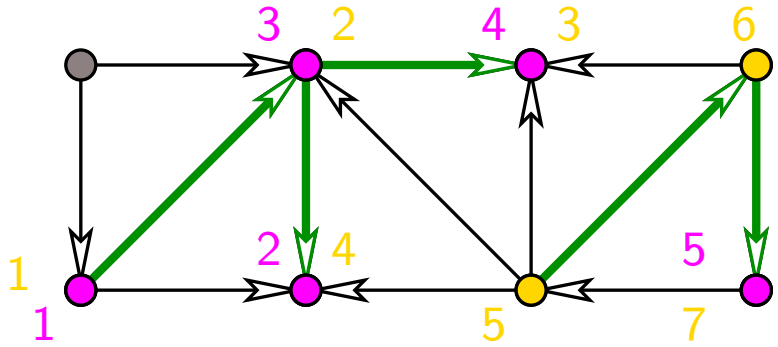
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

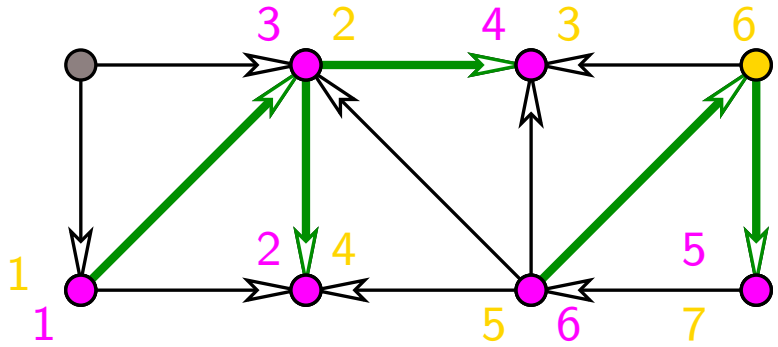
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

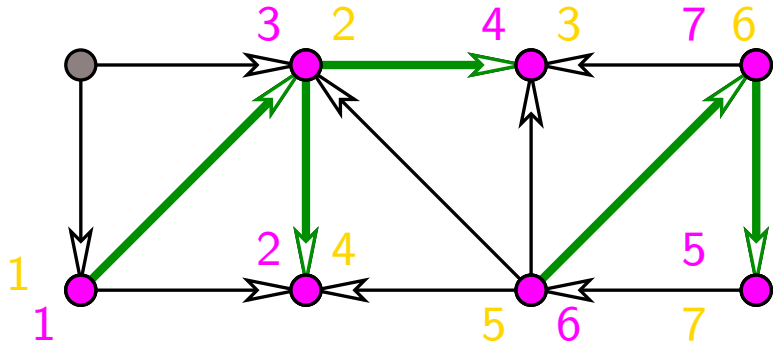
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

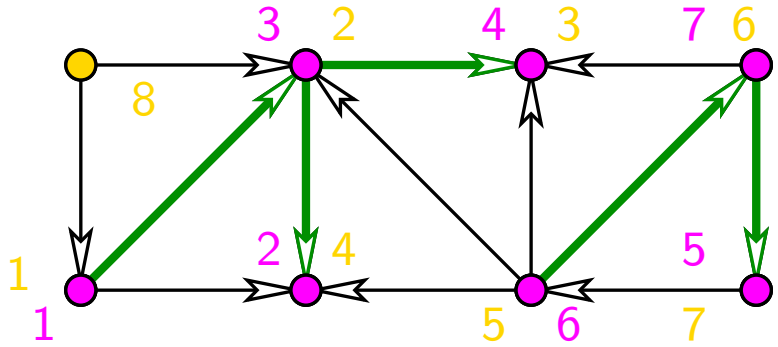
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

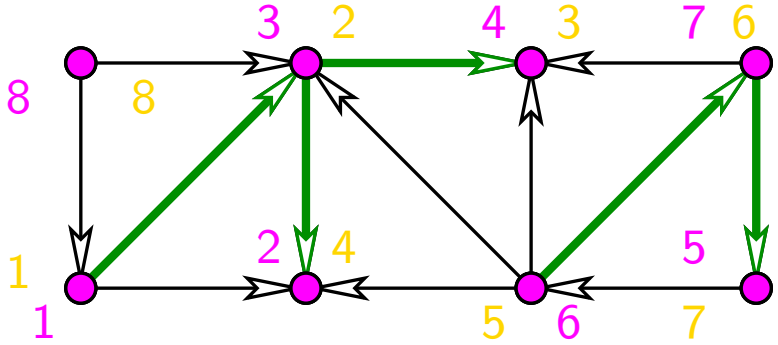
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

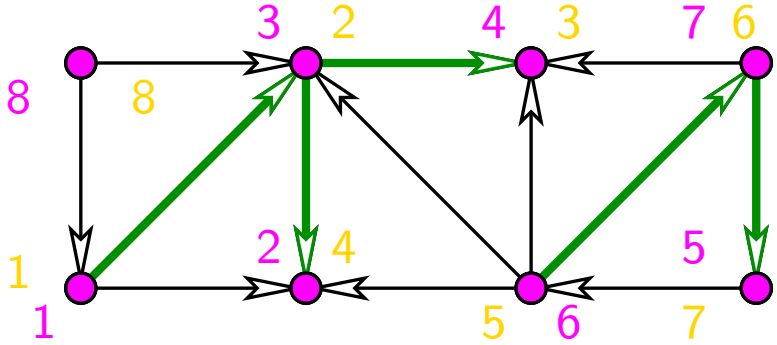
(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Általános gráfbejárás & BFS



A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

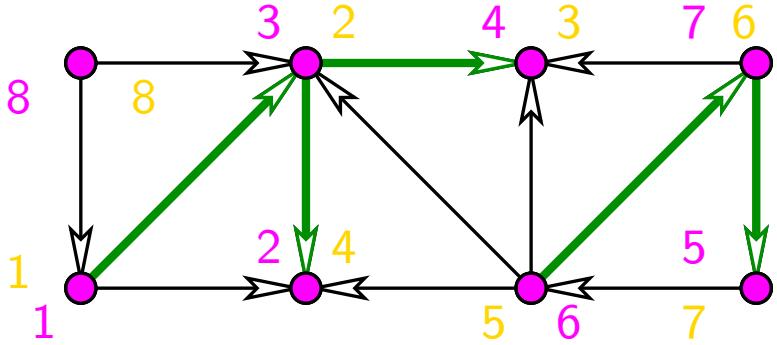
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

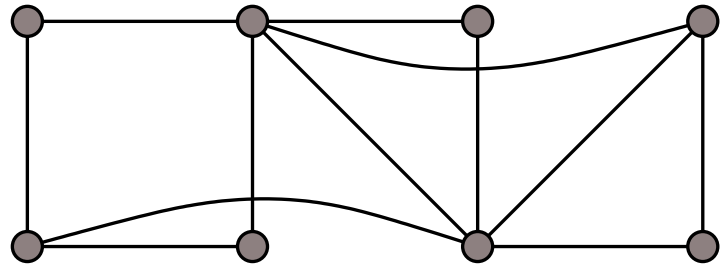
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

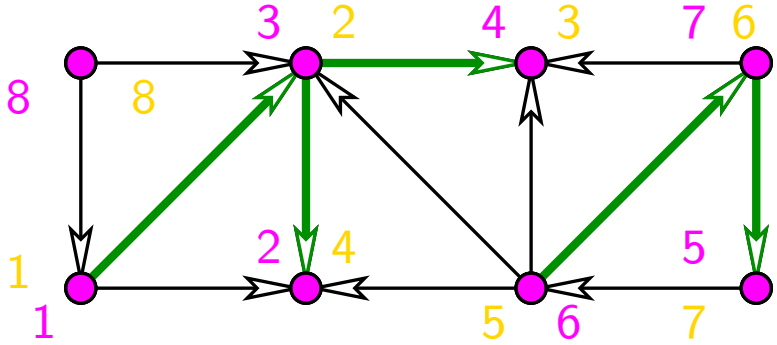
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

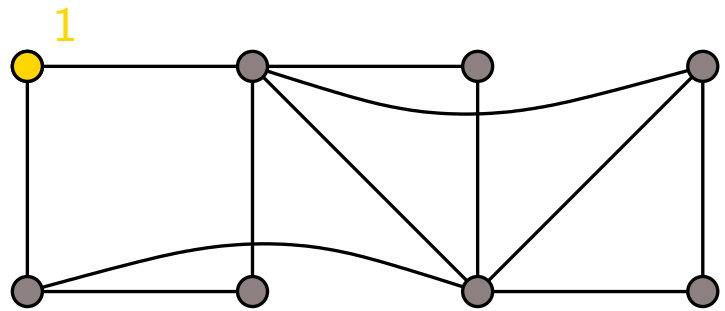
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

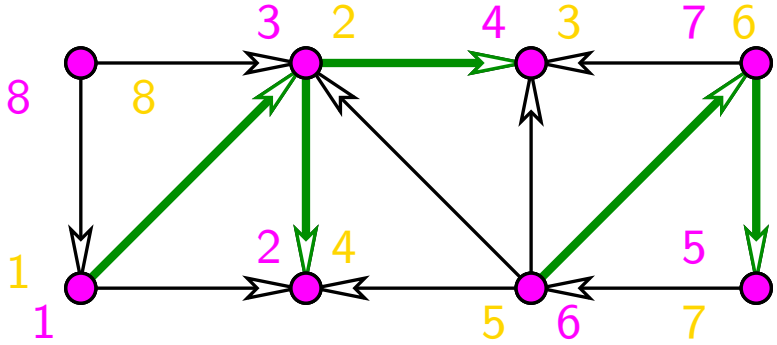
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

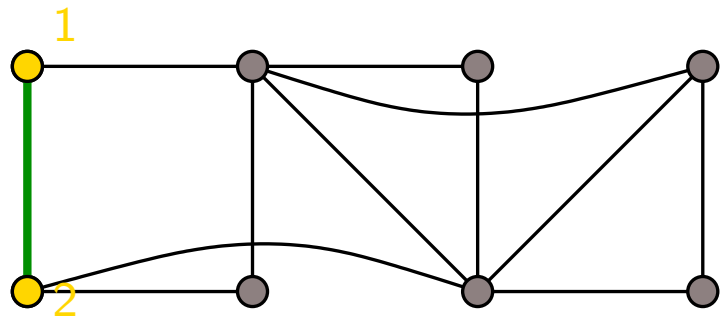
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

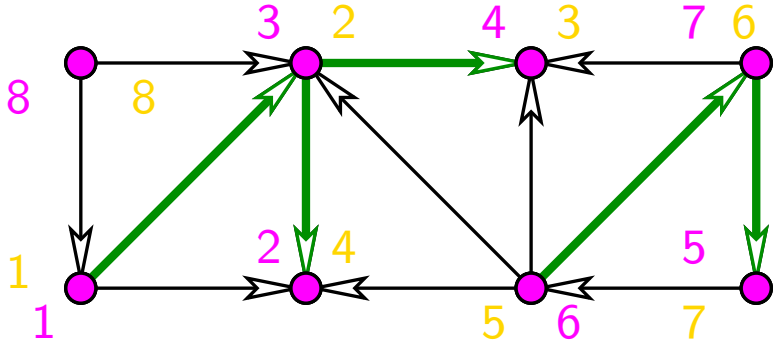
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

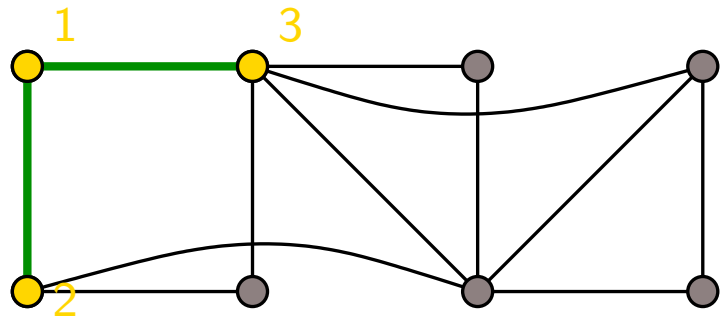
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

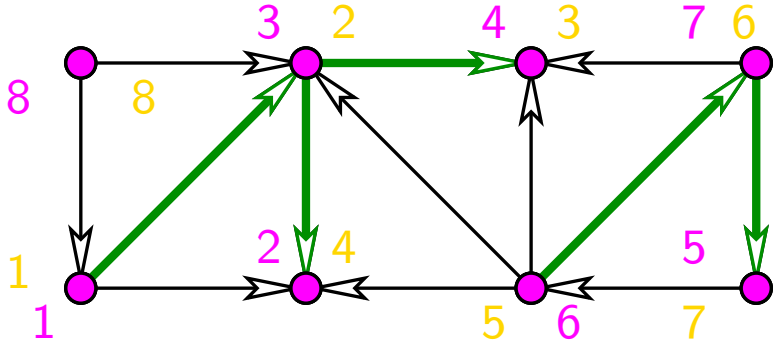
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

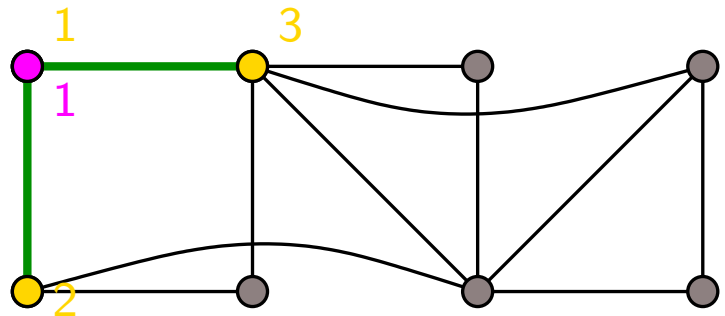
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

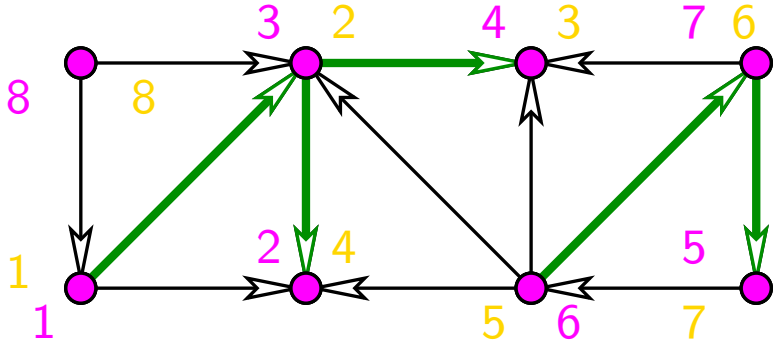
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

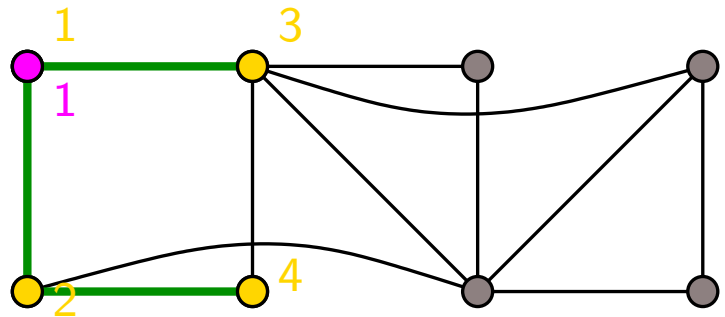
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

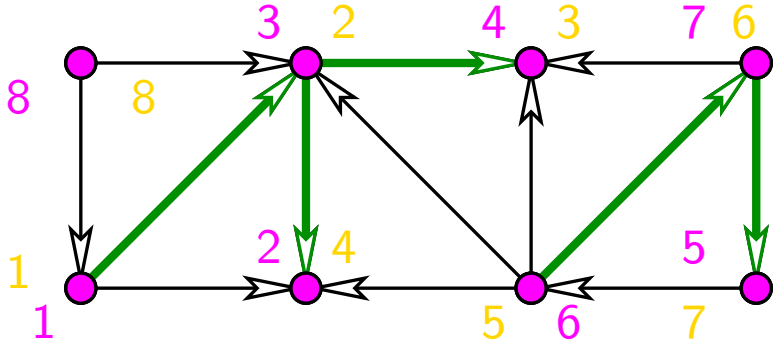
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

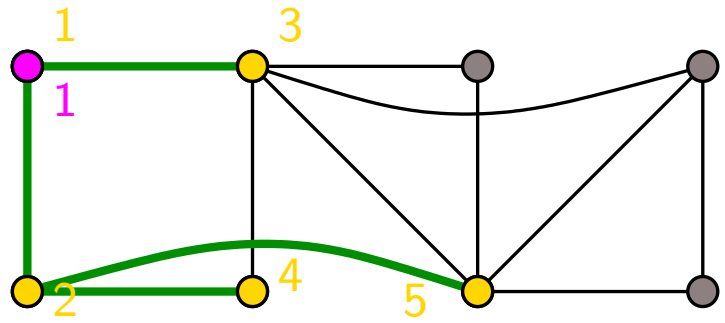
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

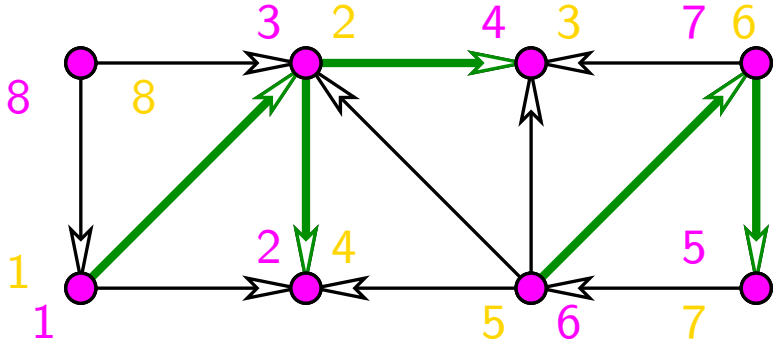
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

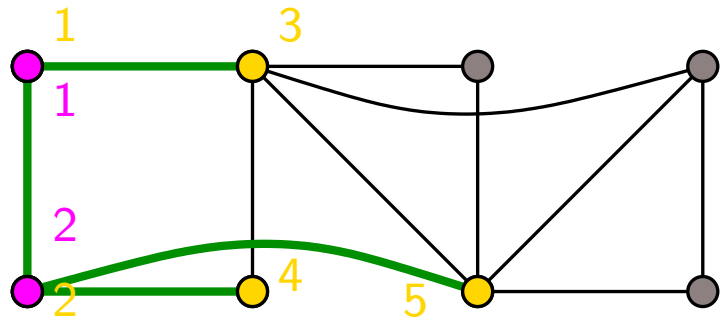
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

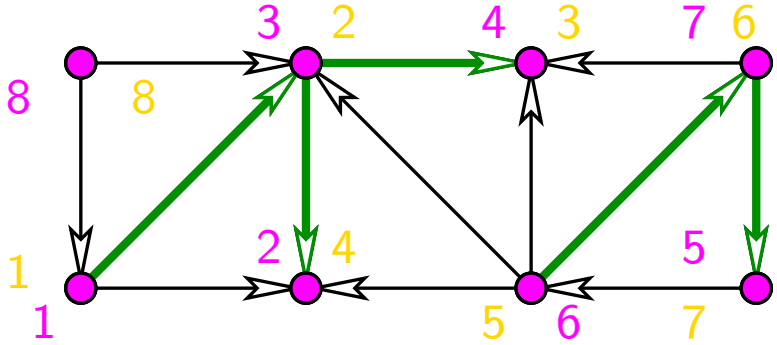
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

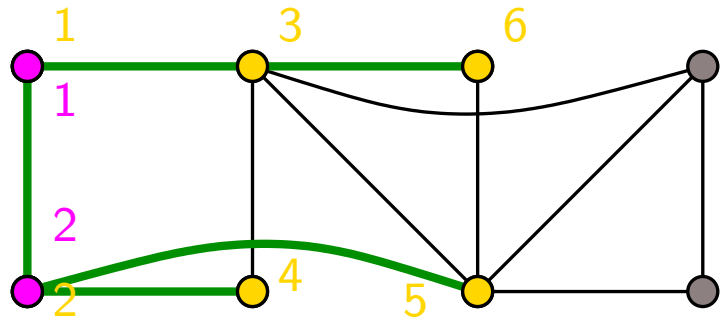
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

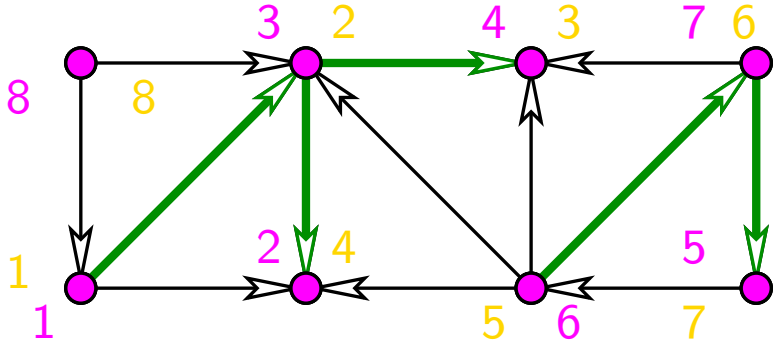
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

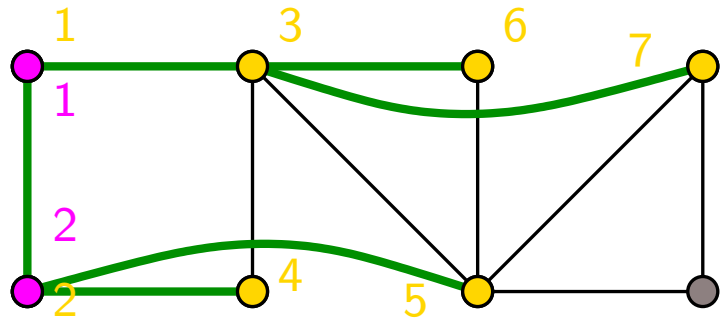
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

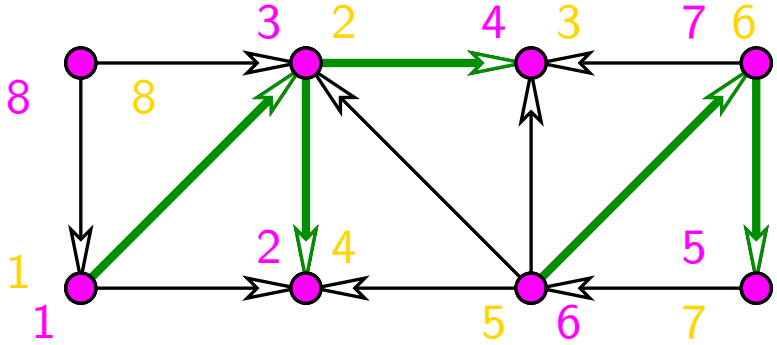
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

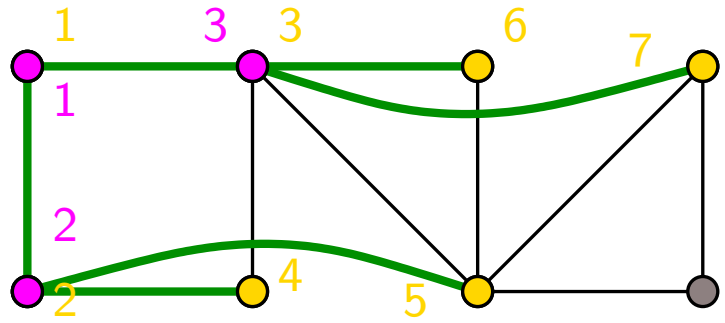
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

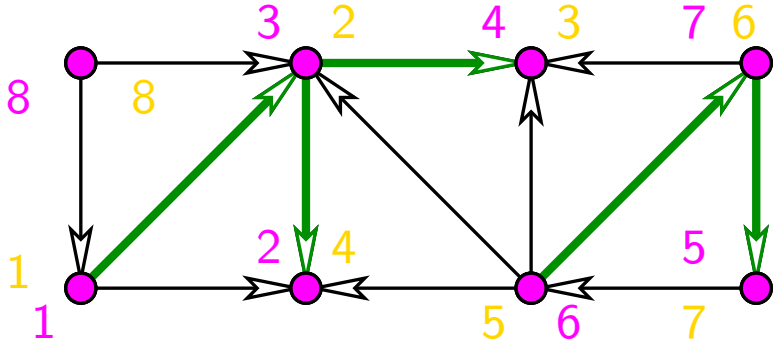
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

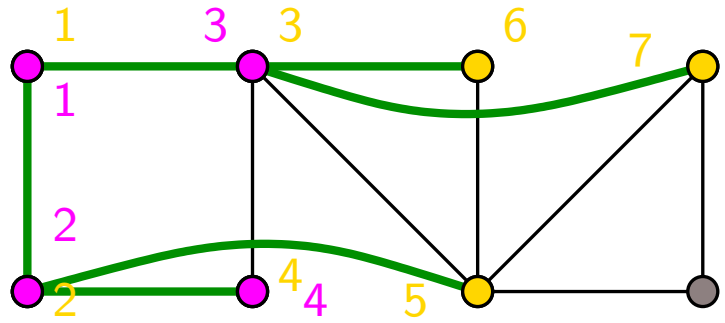
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

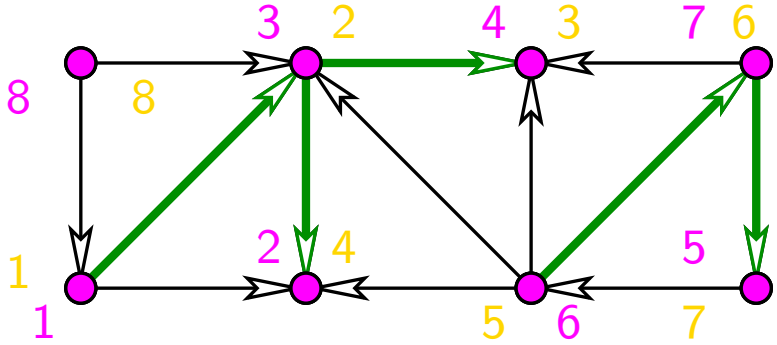
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

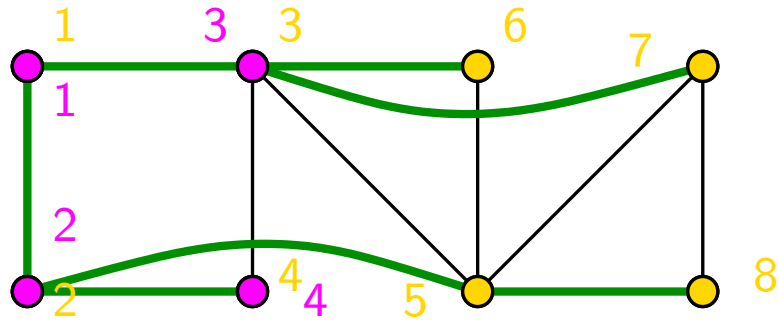
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

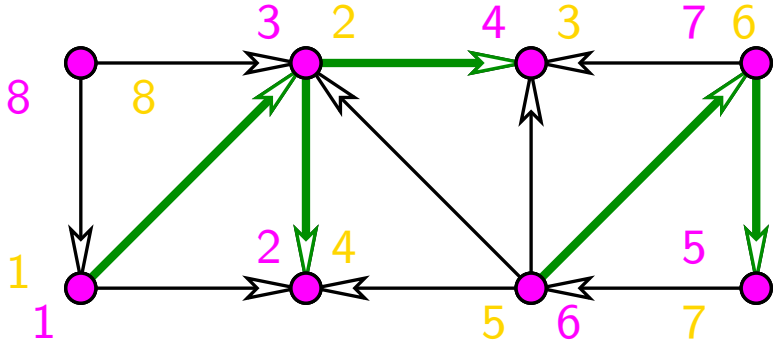
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

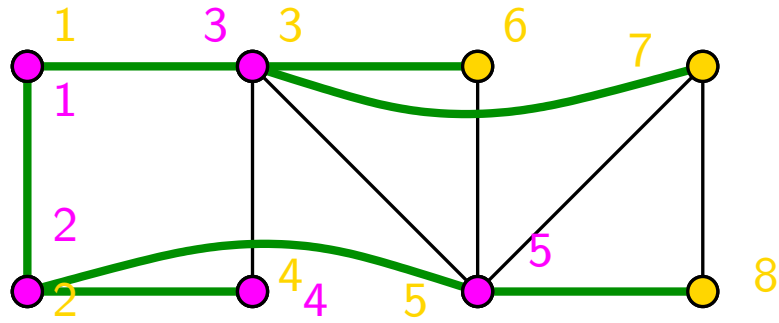
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

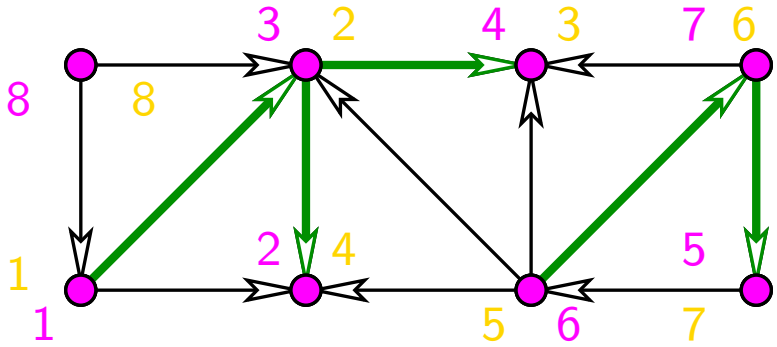
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

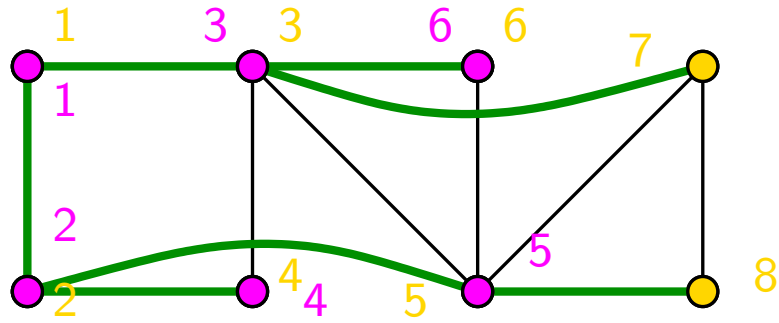
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

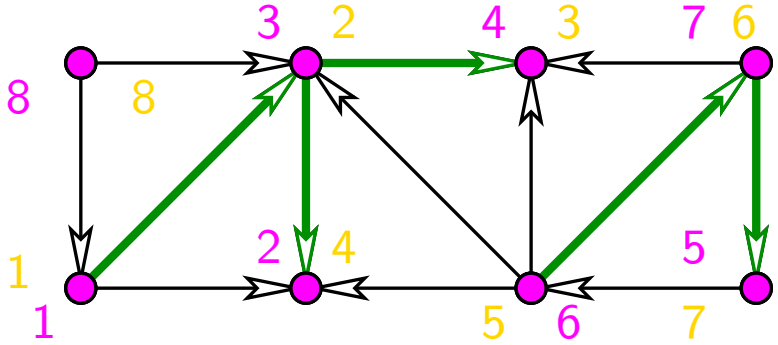
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

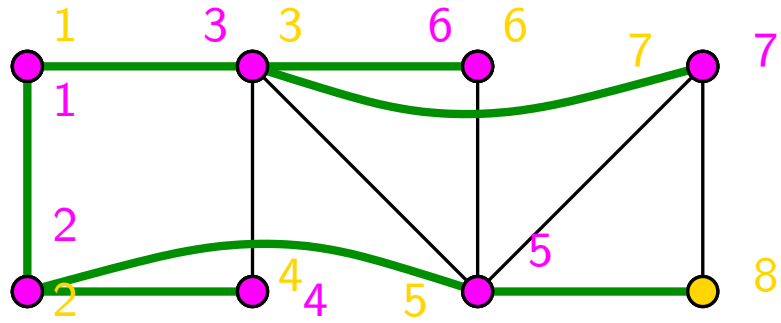
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

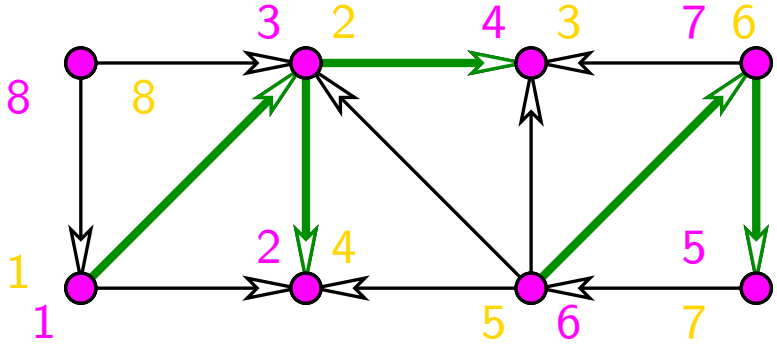
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

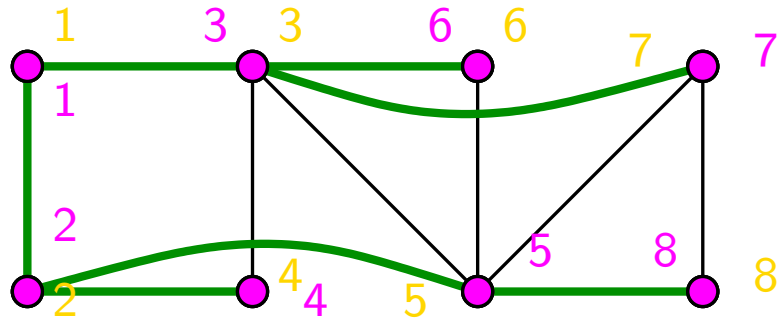
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS





A gráfbejárási algoritmus az inputgráf csúcsait és éleit fedezi fel. Minden csúcs az eléretlen → elért → befejezett állapotokat veszi fel. A bejárás akkor ér véget, amint minden csúcs befejezetté vált.

1. Van elért csúcs. Választunk egyet, mondjuk u-t.

(1a) Ha van olyan uv él, amire v eléretlen, akkor v elértté válik.

(1b) Ha nincs ilyen uv él, akkor u befejezetté válik.

2. Nincs elért csúcs.

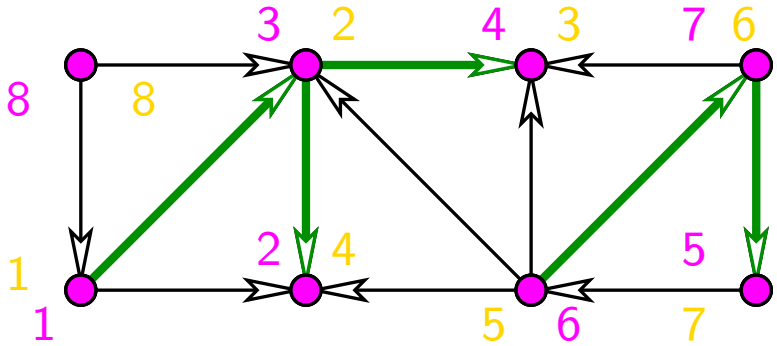
(2a) Ha van eléretlen u csúcs, akkor u-t elértté tesszük.

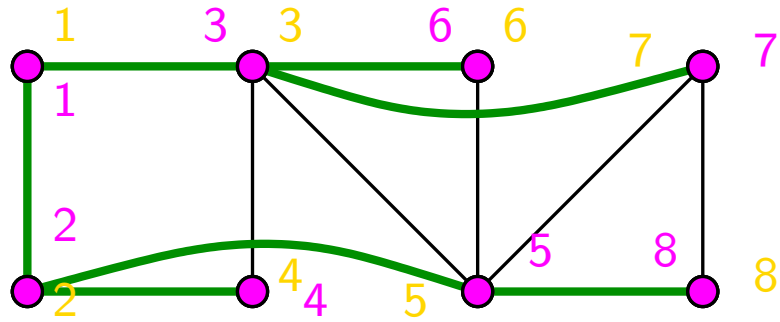
(2b) Ha nincs eléretlen csúcs (azaz $\forall$ csúcs befejezett), akkor END.

Szélességi bejárás (BFS) szabálya:

Az 1. esetben mindig a legkorábban elért u-t választjuk.

Általános gráfbejárás & BFS

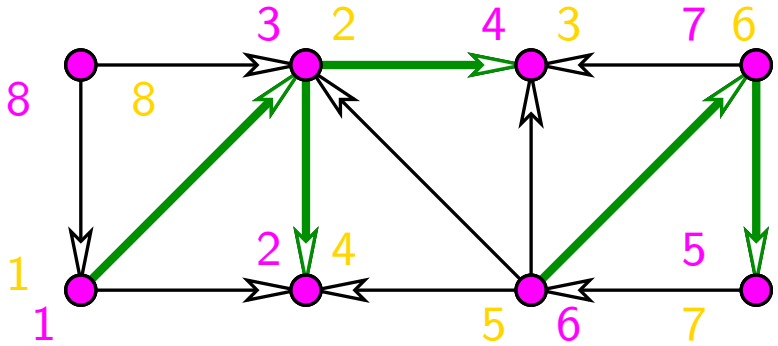


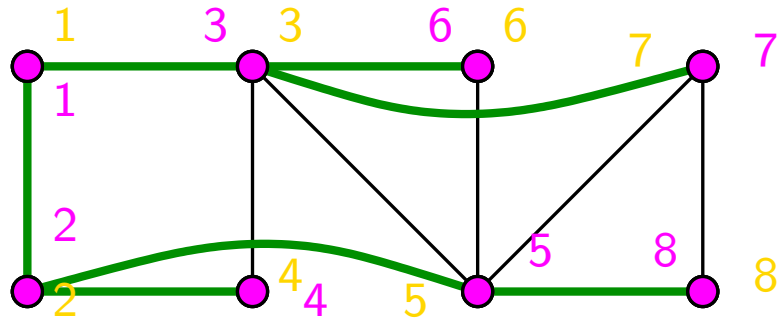


Input: $G = (V, E)$ (ir/ir.tatlan) gráf, ($v \in V$ gyökérpont[1]).

[1] A gyökér kezdetben elért állapotú, ezért kivétel az általános szabály alól.

Általános gráfbejárás & BFS

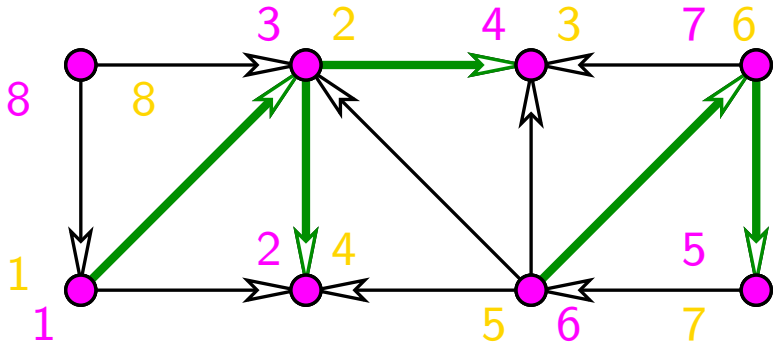


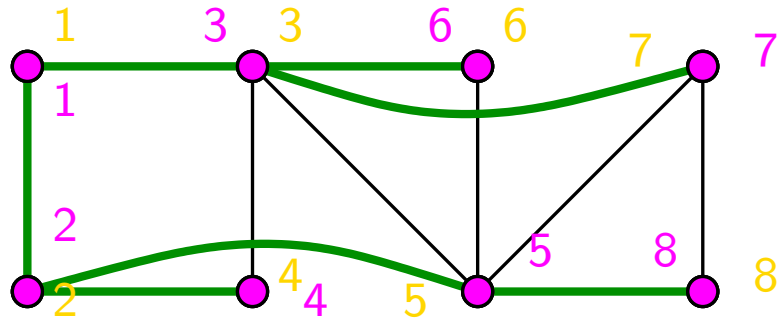


Input: $G = (V, E)$ (ir/ir.tatlan) gráf, ($v \in V$ gyökérpont[1]).
Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.

[1] A gyökér kezdetben elért állapotú, ezért kivétel az általános szabály alól.

Általános gráfbejárás & BFS





Input: $G = (V, E)$ (ir/ir.tatlan) gráf, ($v \in V$ gyökérpont[1]).

Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.

(2) Az élek osztályozása:

faél: Olyan él, ami mentén egy csúcs elértté vált.

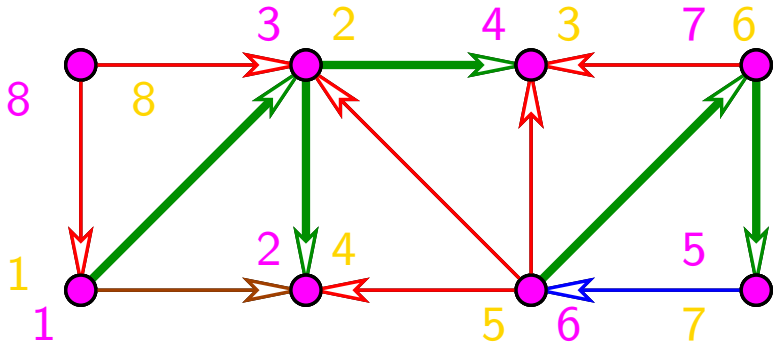
uv **előreél:** nem faél, de *u*-ból *v*-be faélekből irányított út vezet.

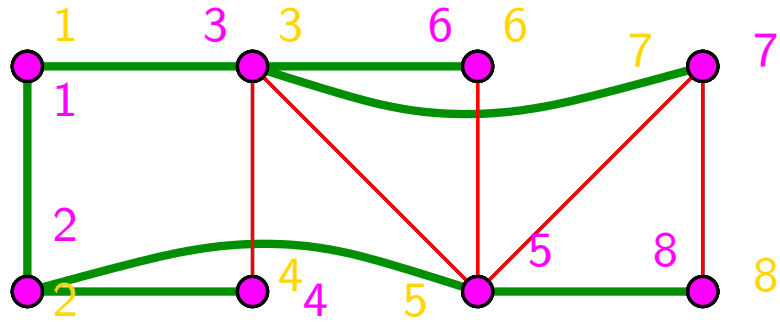
uv **visszaél:** *v*-ből *u*-ba faélekből irányított út vezet.

keresztél: minden más él (*u* és *v* közt nincs leszármazási viszony).

[1] A gyökér kezdetben elért állapotú, ezért kivétel az általános szabály alól.

Általános gráfbejárás & BFS





Input: $G = (V, E)$ (ir/ir.tatlan) gráf, ($v \in V$ gyökérpont[1]).

Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.

(2) Az élek osztályozása:

faél: Olyan él, ami mentén egy csúcs elértté vált.

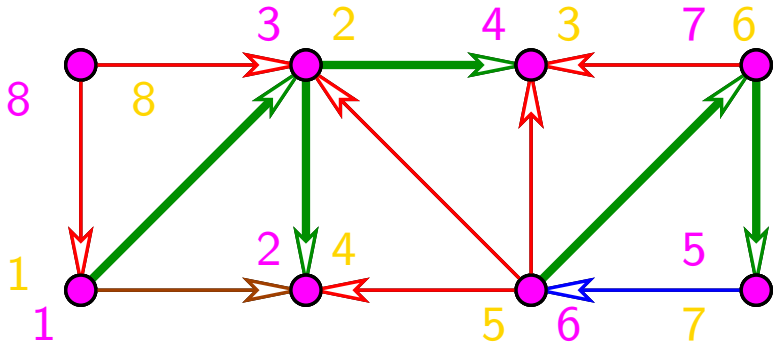
uv **előreél:** nem faél, de *u*-ból *v*-be faélekből irányított út vezet.

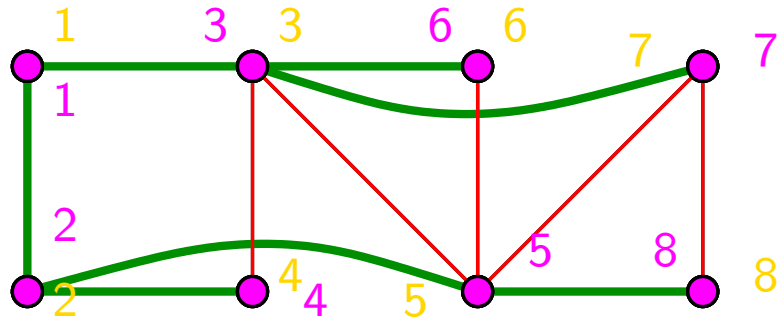
uv **visszaél:** *v*-ből *u*-ba faélekből irányított út vezet.

keresztél: minden más él (*u* és *v* közt nincs leszármazási viszony).

[1] A gyökér kezdetben elért állapotú, ezért kivétel az általános szabály alól.

Általános gráfbejárás & BFS





Input: $G = (V, E)$ (ir/ir.tatlan) gráf, ($v \in V$ gyökérpont[1]).

Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.

(2) Az élek osztályozása:

faél: Olyan él, ami mentén egy csúcs elértté vált.

uv **előreél:** nem faél, de *u*-ból *v*-be faélekből irányított út vezet.

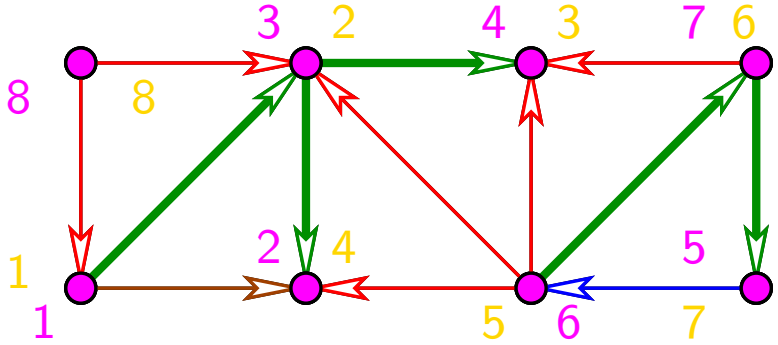
uv **visszaél:** *v*-ből *u*-ba faélekből irányított út vezet.

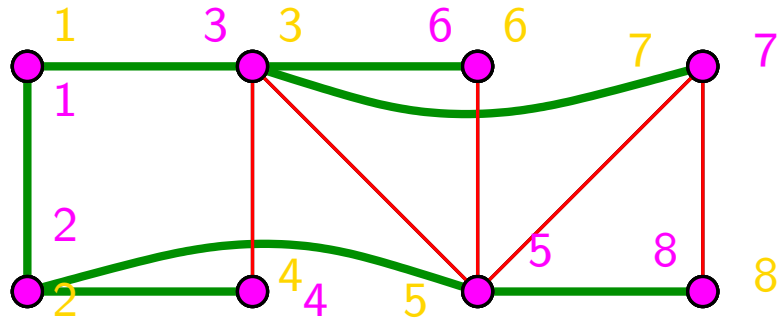
keresztél: minden más él (*u* és *v* közt nincs leszármazási viszony).

(3) A **bejárás fája**: a faélek alkotta részgráf.

[1] A gyökér kezdetben elért állapotú, ezért kivétel az általános szabály alól.

Általános gráfbejárás & BFS





Input: $G = (V, E)$ (ir/ir.tatlan) gráf, ($v \in V$ gyökérpont[1]).

Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.

(2) Az élek osztályozása:

faél: Olyan él, ami mentén egy csúcs elértté vált.

uv **előreél:** nem faél, de *u*-ból *v*-be faélekből irányított út vezet.

uv **visszaél:** *v*-ből *u*-ba faélekből irányított út vezet.

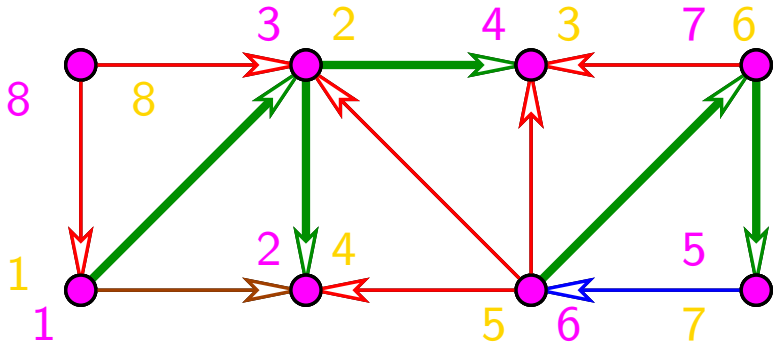
keresztél: minden más él (*u* és *v* közt nincs leszármazási viszony).

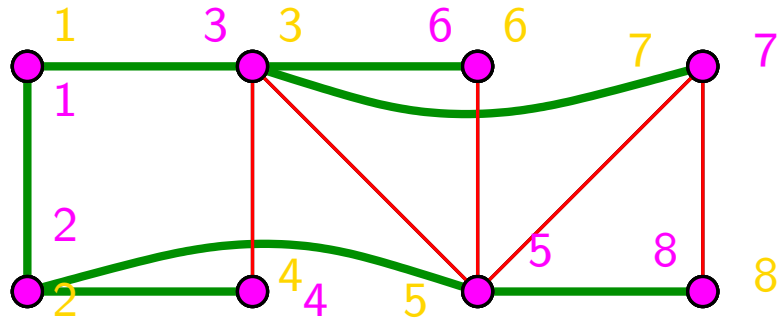
(3) A **bejárás fája**: a faélek alkotta részgráf.

(A bejárás fája valójában egy gyökereiből kifelé irányított erdő.)

[1] A gyökér kezdetben elért állapotú, ezért kivétel az általános szabály alól.

Általános gráfbejárás & BFS





Input: $G = (V, E)$ (ir/ir.tatlan) gráf, ($v \in V$ gyökérpont[1]).

Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.

(2) Az élek osztályozása:

faél: Olyan él, ami mentén egy csúcs elértté vált.

uv **előreél:** nem faél, de *u*-ból *v*-be faélekből irányított út vezet.

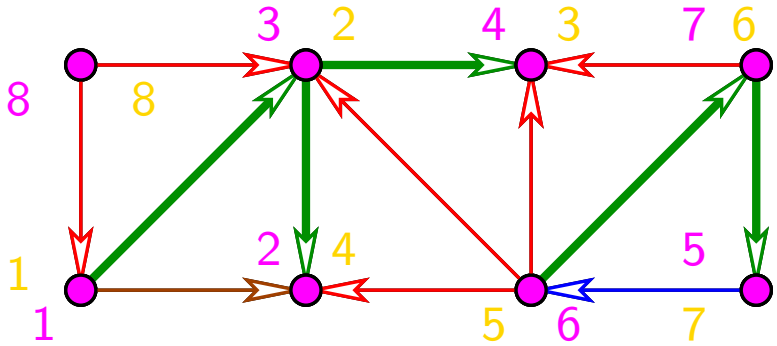
uv **visszaél:** *v*-ből *u*-ba faélekből irányított út vezet.

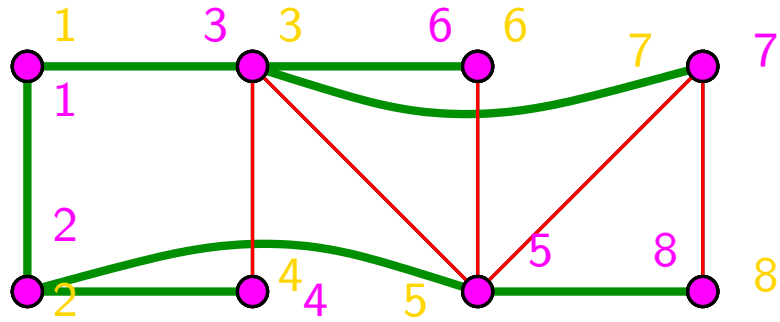
keresztél: minden más él (*u* és *v* közt nincs leszármazási viszony).

(3) A **bejárás fája**: a faélek alkotta részgráf.

[1] A gyökér kezdetben elért állapotú, ezért kivétel az általános szabály alól.

Általános gráfbejárás & BFS





Input: $G = (V, E)$ (ir/ir.tatlan) gráf, ($v \in V$ gyökérpont[1]).

Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.

(2) Az élek osztályozása:

faél: Olyan él, ami mentén egy csúcs elértté vált.

uv **előreél:** nem faél, de *u*-ból *v*-be faélekből irányított út vezet.

uv **visszaél:** *v*-ből *u*-ba faélekből irányított út vezet.

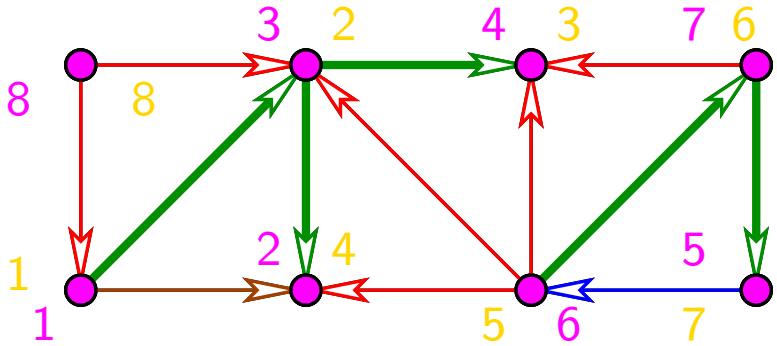
keresztél: minden más él (*u* és *v* közt nincs leszármazási viszony).

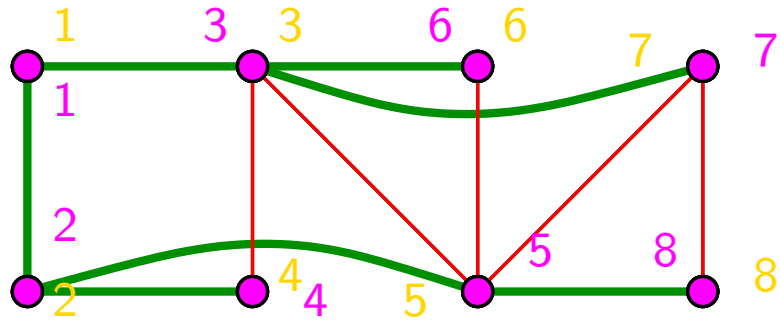
(3) A **bejárás fája**: a faélek alkotta részgráf.

Megf: Irányítatlan esetben az előreél és a visszaél ugyanazt jelenti.

[1] A gyökér kezdetben elért állapotú, ezért kivétel az általános szabály alól.

Általános gráfbejárás & BFS

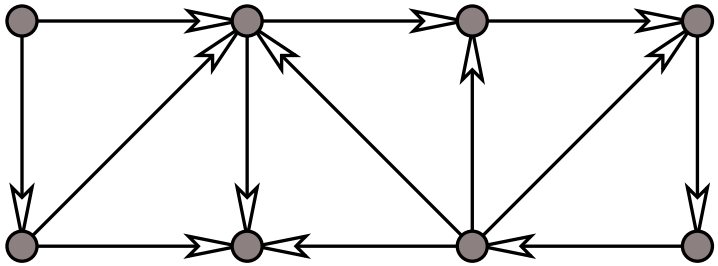




Input: $G = (V, E)$ (ir/ir.tatlan) gráf, ($v \in V$ gyökérpont[1]).
Output: (1) A csúcsok **elérési** és **befejezési** sorrendje.
(2) Az élek osztályozása:
faél: Olyan él, ami mentén egy csúcs elértté vált.
uv **előreél:** nem faél, de *u*-ból *v*-be faélekből irányított út vezet.
uv **visszaél:** *v*-ből *u*-ba faélekből irányított út vezet.
keresztél: minden más él (*u* és *v* közt nincs leszármazási viszony).
(3) A **bejárás fája**: a faélek alkotta részgráf.
Megf: Irányítatlan esetben az előreél és a visszaél ugyanazt jelenti.
Terminológia: Ha a bejárás fájában *u*-ból *v*-be irányított út vezet, akkor *u* a *v* ős és *v* az *u* leszármazottja. A faél és az előreél tehát ősből leszármazottba, a visszaél leszármazottból ősbe vezet.

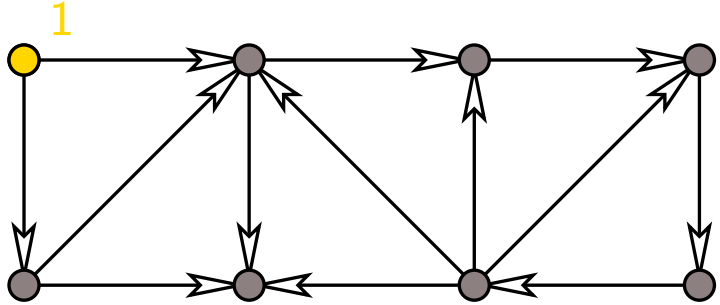
[1] A gyökér kezdetben elért állapotú, ezért kivétel az általános szabály alól.

A BFS tulajdonságai



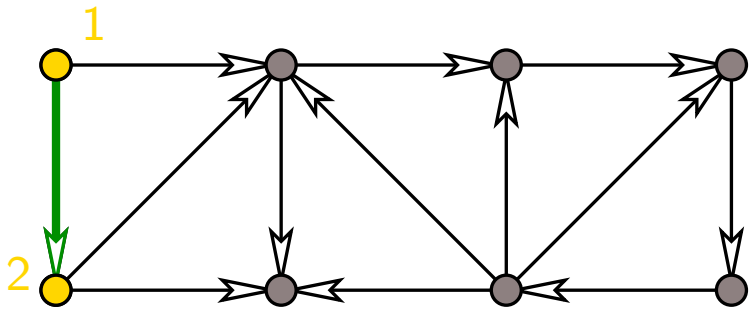
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



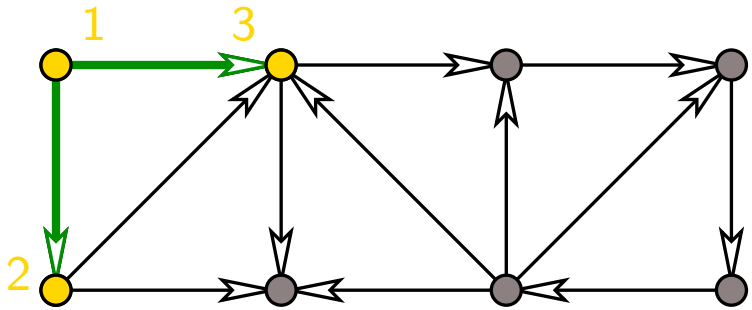
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



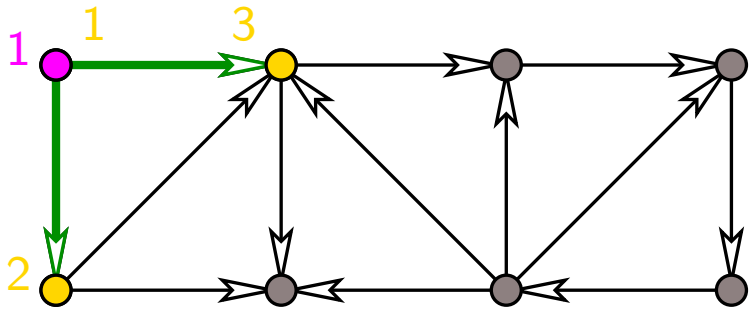
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



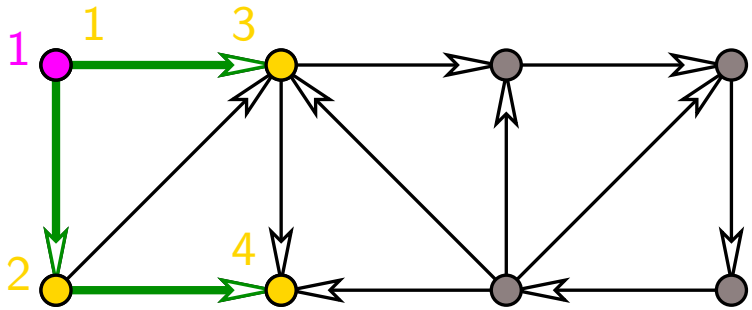
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



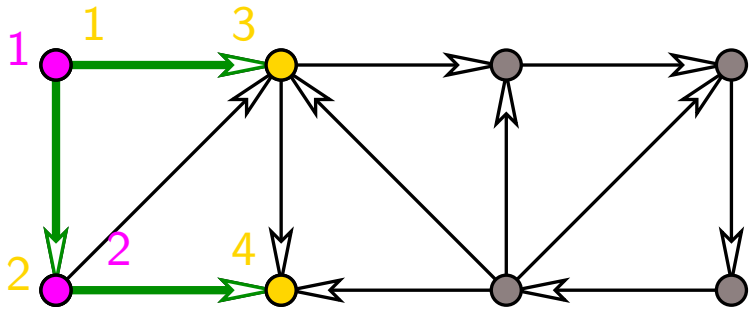
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



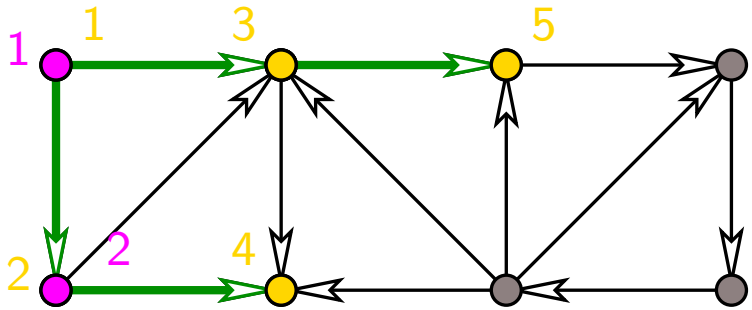
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



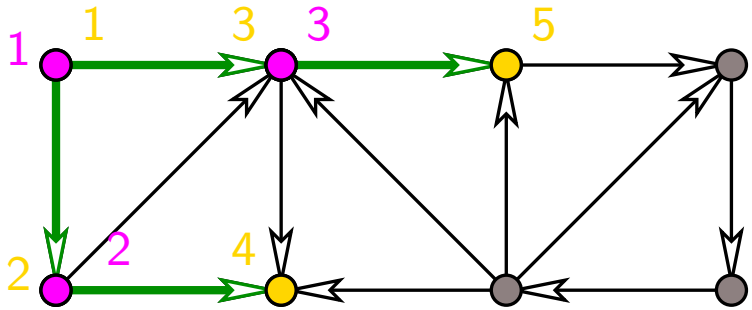
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



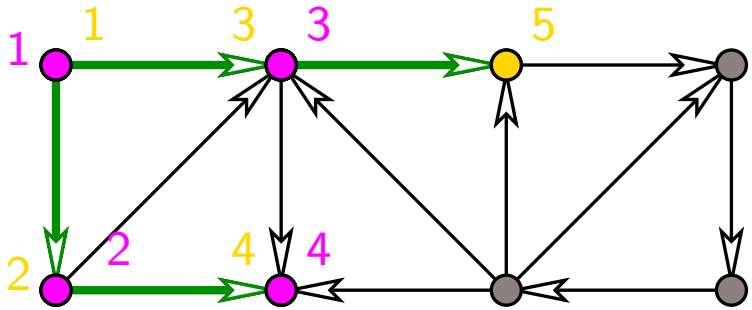
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



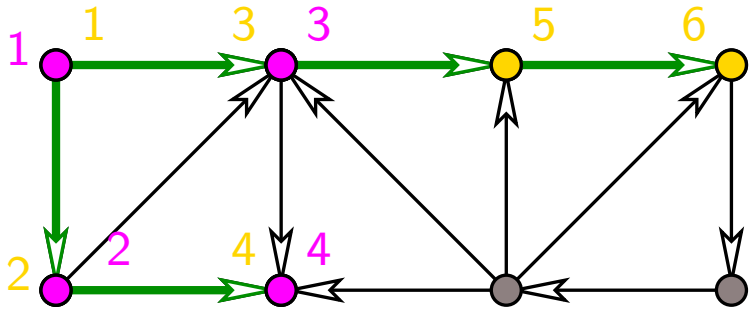
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



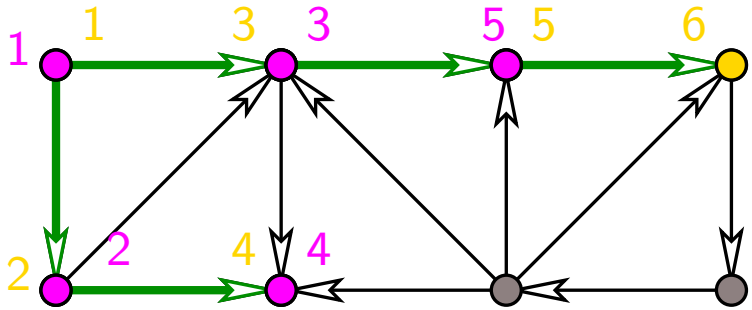
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



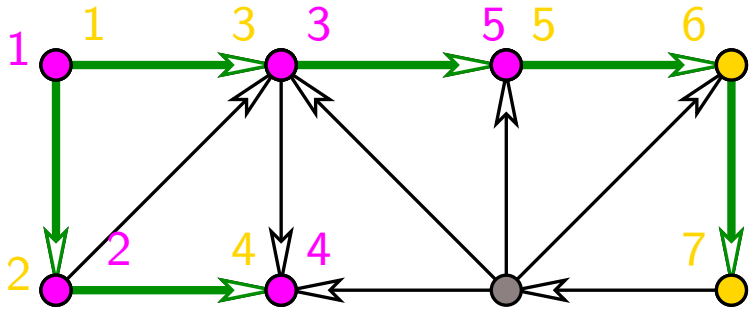
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



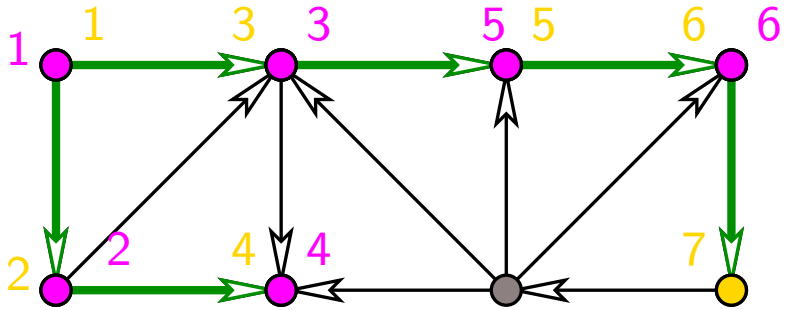
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



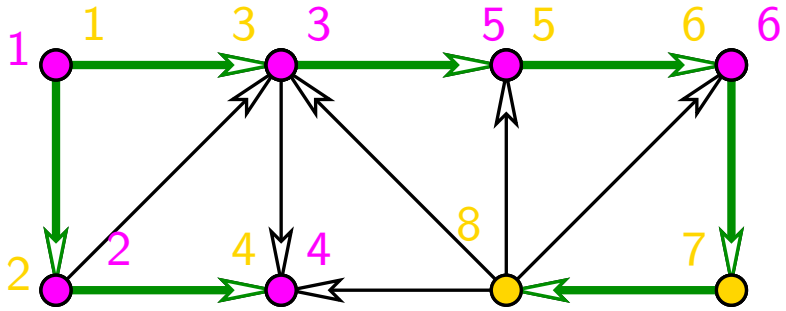
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



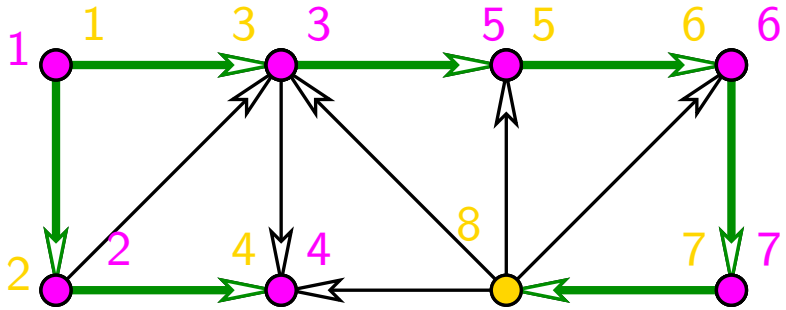
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



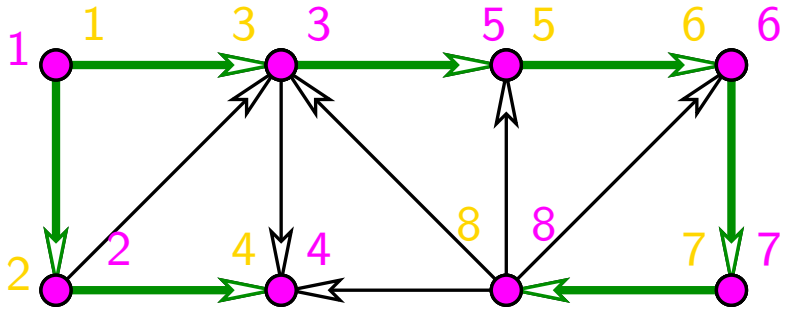
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



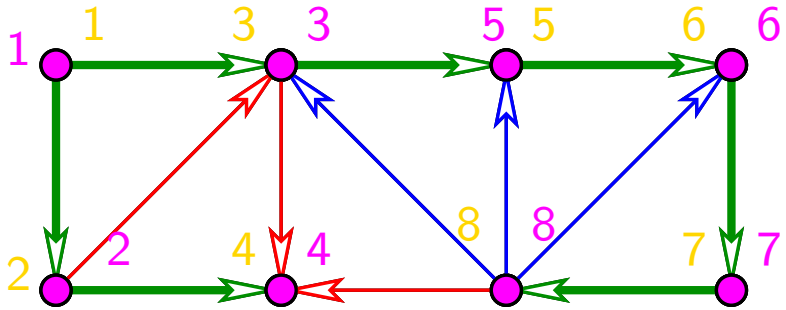
Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



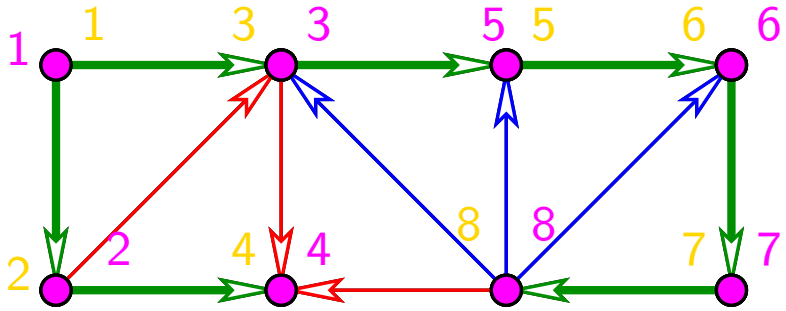
Nézzük meg egy **irányított** gráf BFS bejárását is.

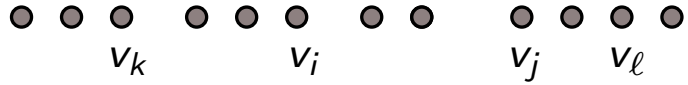
A BFS tulajdonságai



Nézzük meg egy **irányított** gráf BFS bejárását is.

A BFS tulajdonságai



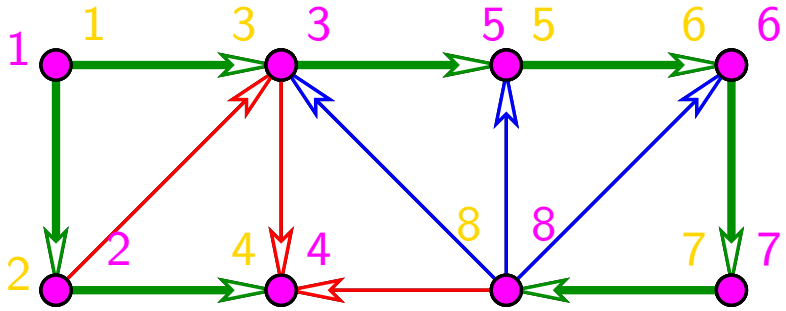


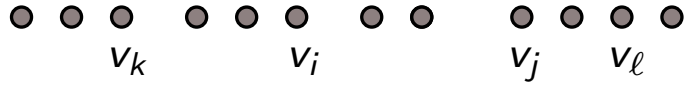
Nézzük meg egy **irányított** gráf BFS bejárását is.

Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

A BFS tulajdonságai





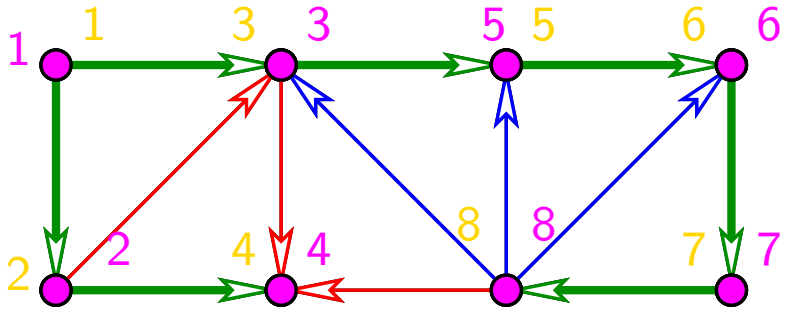
Nézzük meg egy **irányított** gráf BFS bejárását is.

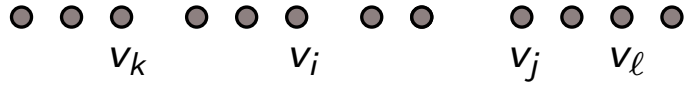
Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

Biz: A v_i-t befejezésének pillanatában v_i minden gyereke elért, de v_j-nek még egy gyereke sem az. Ezért v_j gyerekeit a v_i csúcs befejezése után érjük el, majd ezt követően fejezzük be v_j-t. □

A BFS tulajdonságai



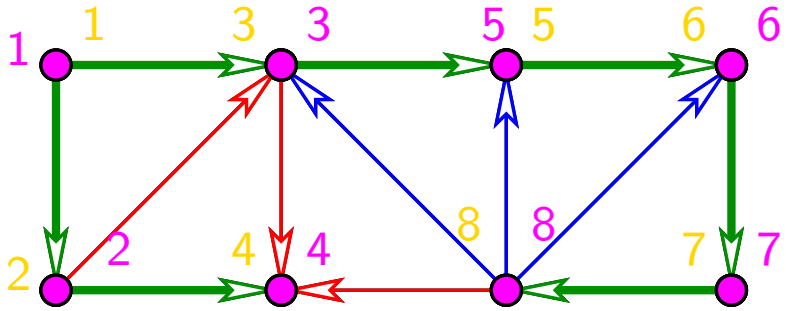


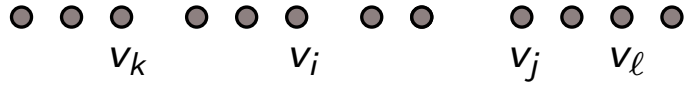
Nézzük meg egy **irányított** gráf BFS bejárását is.

Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

A BFS tulajdonságai





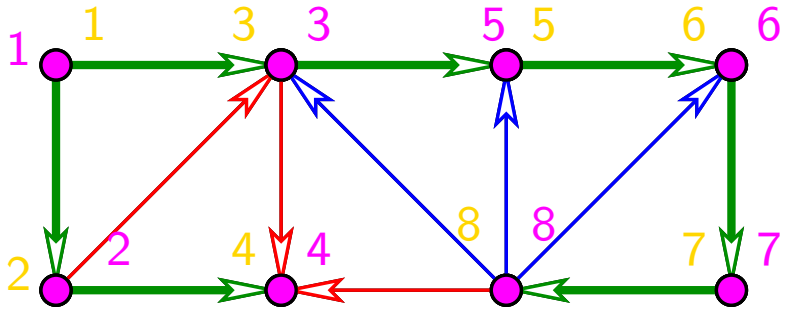
Nézzük meg egy **irányított** gráf BFS bejárását is.

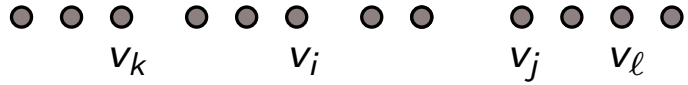
Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

(2) **Az elérési és befejezési sorrend (BFS esetén) megegyezik.**

A BFS tulajdonságai





Nézzük meg egy **irányított** gráf BFS bejárását is.

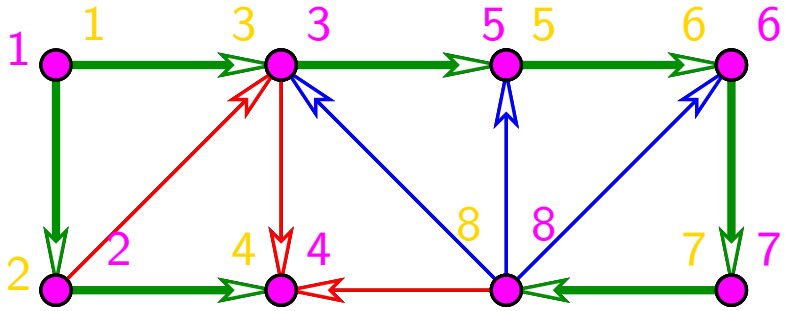
Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

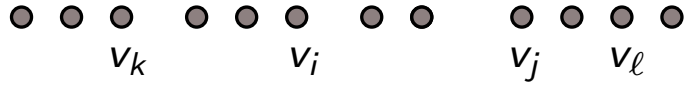
(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

(2) **Az elérési és befejezési sorrend (BFS esetén) megegyezik.**

Biz: Ha v_i-t korábban érjük el, mint v_j-t, akkor (1) miatt v_i-t korábban is fejezzük be v_j-nél. Ezért bármely két csúcs sorrendje ugyanaz az elérési sorrendben mint befejezési sorrendben. Tehát az elérési sorrendnek meg kell egyeznie a befejezési sorrenddel. □

A BFS tulajdonságai





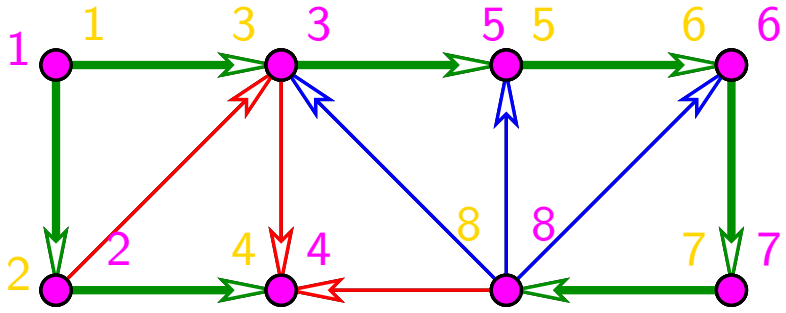
Nézzük meg egy **irányított** gráf BFS bejárását is.

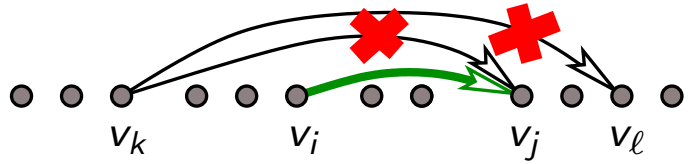
Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

(2) **Az elérési és befejezési sorrend (BFS esetén) megegyezik.**

A BFS tulajdonságai





Nézzük meg egy **irányított** gráf BFS bejárását is.

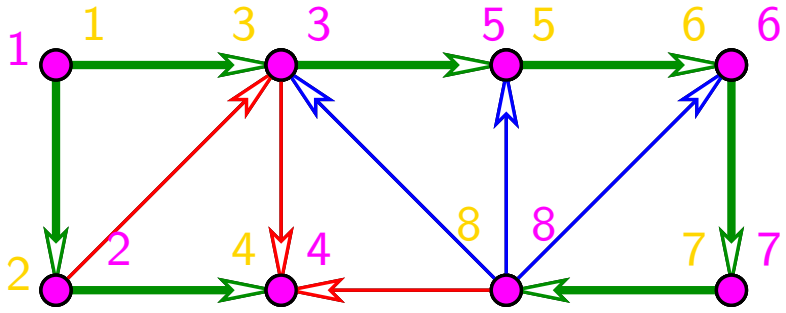
Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

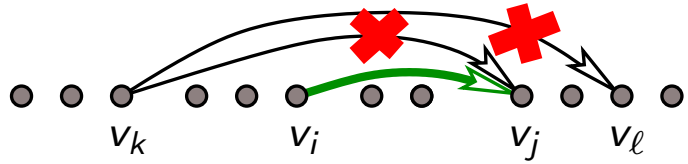
(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

(2) **Az elérési és befejezési sorrend (BFS esetén) megegyezik.**

(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

A BFS tulajdonságai





Nézzük meg egy **irányított** gráf BFS bejárását is.

Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

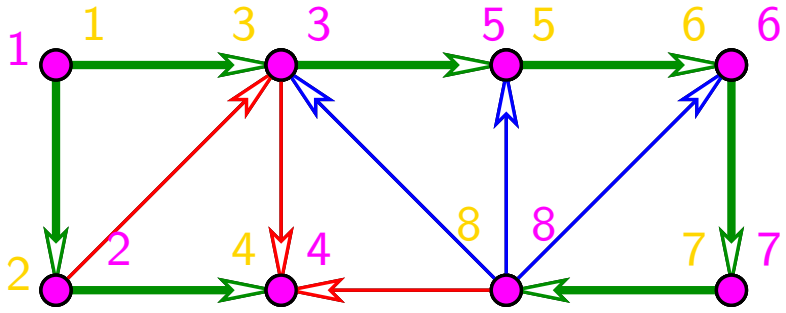
(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

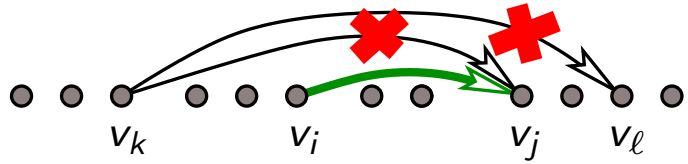
(2) **Az elérési és befejezési sorrend (BFS esetén) megegyezik.**

(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

Biz: Ha $v_k v_\ell \in E(G)$, akkor v_ℓ szülője v_k vagy egy v_k-t megelőző csúcs. (1) miatt v_j szülője sem következhet v_k után, vagyis v_i nem lehet v_j szülője. □

A BFS tulajdonságai





Nézzük meg egy **irányított** gráf BFS bejárását is.

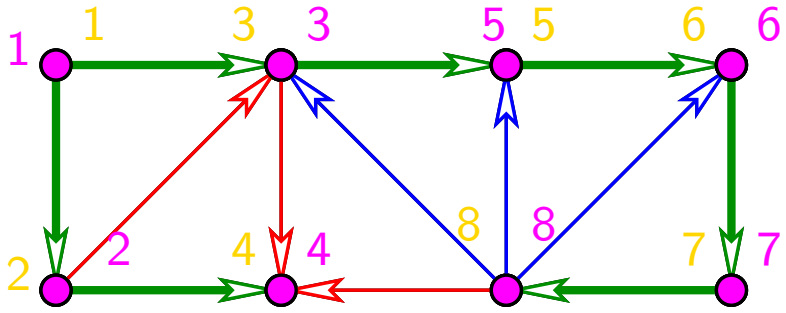
Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

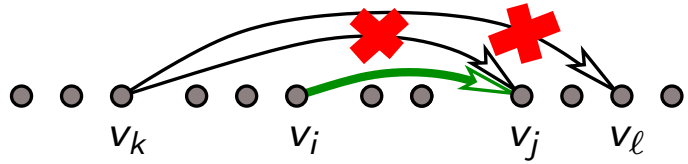
(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

(2) **Az elérési és befejezési sorrend (BFS esetén) megegyezik.**

(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

A BFS tulajdonságai





Nézzük meg egy **irányított** gráf BFS bejárását is.

Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

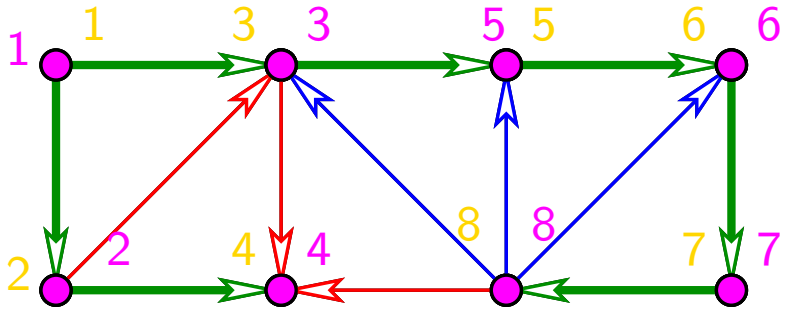
(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

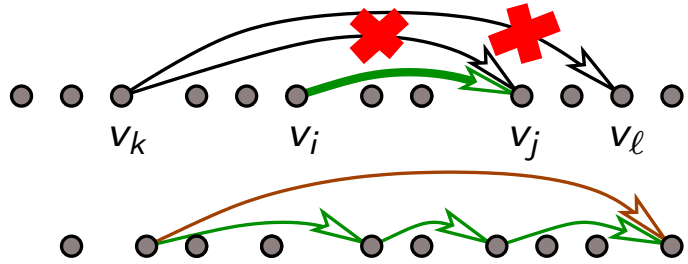
(2) **Az elérési és befejezési sorrend (BFS esetén) megegyezik.**

(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

(4) **Nincs előreél.** (Irányítatlan eset: csak faél és keresztél van.)

A BFS tulajdonságai





Nézzük meg egy **irányított** gráf BFS bejárását is.

Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

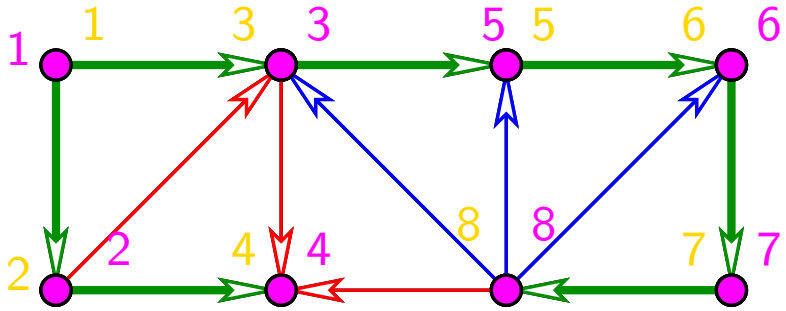
(2) **Az elérési és befejezési sorrend (BFS esetén) megegyezik.**

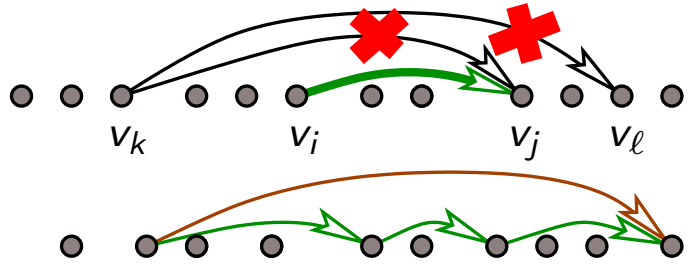
(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

(4) **Nincs előreél.** (Irányítatlan eset: csak faél és keresztél van.)

Biz: Indirekt: ha $v_i v_j$ előreél lenne, akkor v_i-ből v_j-be irányított út vezetne a BFS-fában, és $v_i v_j$ ennek a faélekből álló útnak az utolsó élét átugrná. □

A BFS tulajdonságai





Nézzük meg egy **irányított** gráf BFS bejárását is.

Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

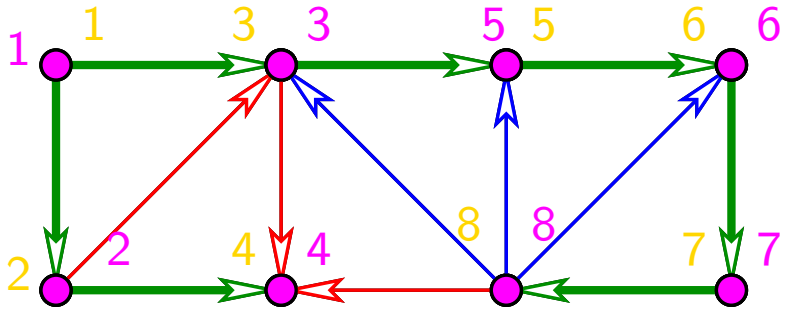
(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

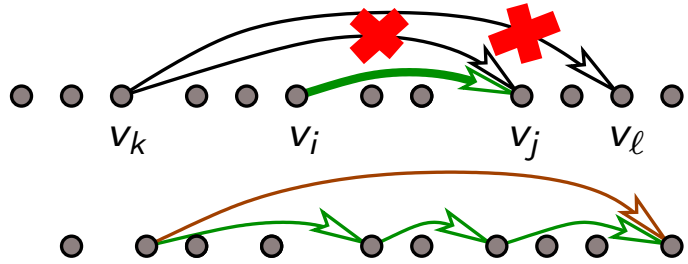
(2) **Az elérési és befejezési sorrend (BFS esetén) megegyezik.**

(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

(4) **Nincs előreél.** (Irányítatlan eset: csak faél és keresztél van.)

A BFS tulajdonságai





Nézzük meg egy **irányított** gráf BFS bejárását is.

Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

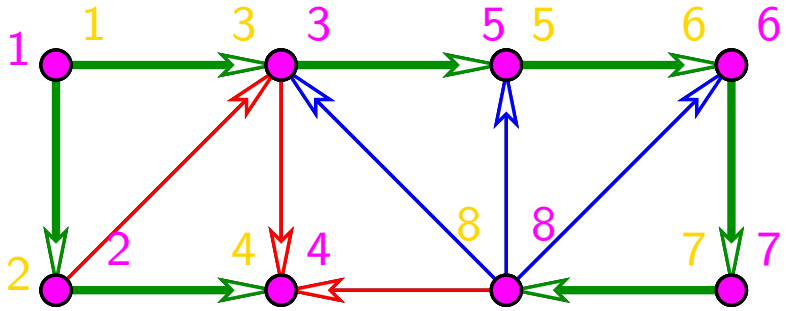
(2) **Az elérési és befejezési sorrend (BFS esetén) megegyezik.**

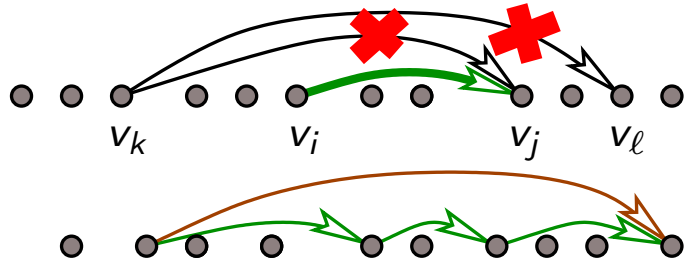
(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

(4) **Nincs előreél.** (Irányítatlan eset: csak faél és keresztél van.)

(5) Ha a BFS-fában k-élű irányított út vezet u-ból v-be, akkor G-ben nincs k-nál kevesebb élű uv-út.

A BFS tulajdonságai





Nézzük meg egy **irányított** gráf BFS bejárását is.

Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

(2) **Az elérési és befejezési sorrend (BFS esetén) megegyezik.**

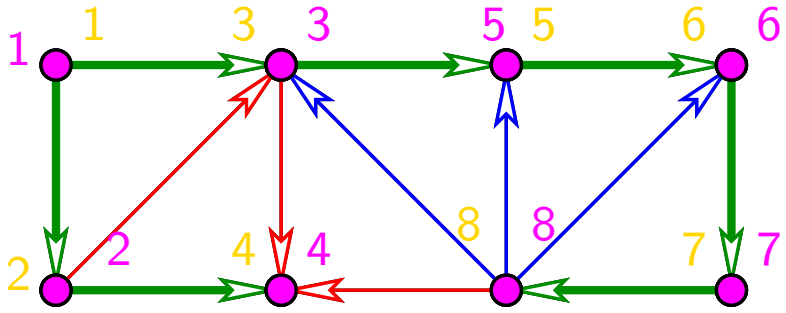
(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

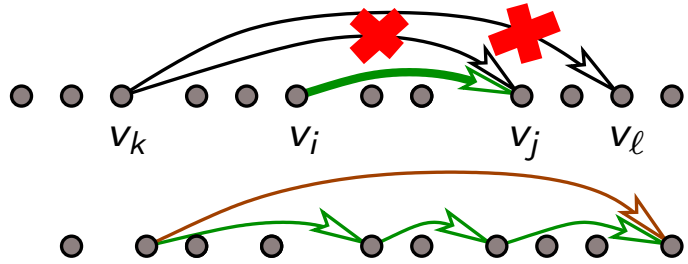
(4) **Nincs előreél.** (Irányítatlan eset: csak faél és keresztél van.)

(5) Ha a BFS-fában k-élű irányított út vezet u-ból v-be, akkor G-ben nincs k-nál kevesebb élű uv-út.

Biz: Ha lenne a BFS fa-beli útnál kevesebb élű út G-ben, akkor lenne olyan gráfél, ami faélt ugrik át. □

A BFS tulajdonságai





Nézzük meg egy **irányított** gráf BFS bejárását is.

Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

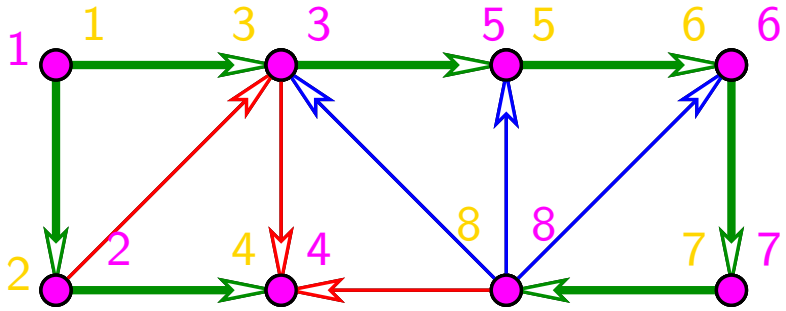
(2) **Az elérési és befejezési sorrend (BFS esetén) megegyezik.**

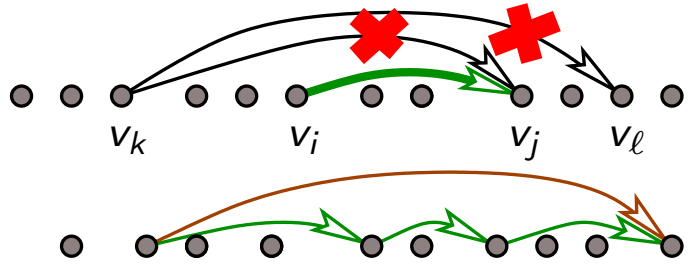
(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

(4) **Nincs előreél.** (Irányítatlan eset: csak faél és keresztél van.)

(5) Ha a BFS-fában k-élű irányított út vezet u-ból v-be, akkor G-ben nincs k-nál kevesebb élű uv-út.

A BFS tulajdonságai





Nézzük meg egy **irányított** gráf BFS bejárását is.

Állítás: Tfh $G = (V, E)$ BFS bejárása után a csúcsok elérési sorrendje $v_1, v_2, \ldots, v_n$. Ekkor az alábbiak teljesülnek.

(1) Ha $i < j$, akkor v_i-t hamarabb fejezzük be, mint v_j-t, továbbá v_i gyerekei megelőzik v_j gyerekeit az elérési sorrendben.

(2) **Az elérési és befejezési sorrend (BFS esetén) megegyezik.**

(3) **Gráfél nem ugorhat át faélt**: ha $k < i < j \leq \ell$ és $v_i v_j$ faél, akkor $v_k v_\ell$ nem lehet gráfél.

(4) **Nincs előreél.** (Irányítatlan eset: csak faél és keresztél van.)

(5) Ha a BFS-fában k-élű irányított út vezet u-ból v-be, akkor G-ben nincs k-nál kevesebb élű uv-út.

(6) **A BFS-fa egy legrövidebb utak fája:** a BFS-fa v_1 gyökeréből bmely v_i csúcsba vezető faút a G egy legkevesebb élű $v_1 v_i$-útja.

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\ell(P) = \sum_{e \in E(P)} \ell(e)$.

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\ell(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\ell(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\ell(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\ell(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)
Az ℓ hosszfüggvény nemnegatív, ha $\ell(e) \geq 0$ teljesül minden e élre.
Az ℓ hosszfv konzervatív, ha G-ben $\nexists$ negatív összhosszú ir. kör.

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\ell(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\ell(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)
Az ℓ hosszfüggvény nemnegatív, ha $\ell(e) \geq 0$ teljesül minden e élre.
Az ℓ hosszfv konzervatív, ha G-ben $\nexists$ negatív összhosszú ir. kör.
Cél: Legrövidebb út keresése irányított/irányítatlan gráfban.

Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\ell(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\ell(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)
Az ℓ hosszfüggvény nemnegatív, ha $\ell(e) \geq 0$ teljesül minden e élre.
Az ℓ hosszfv konzervatív, ha G-ben $\nexists$ negatív összhosszú ir. kör.
Cél: Legrövidebb út keresése irányított/irányítatlan gráfban.
Megf: Ha $\ell(e) = 1$ a G minden e élére, akkor $\ell(P)$ a P élszáma.
Ezért a BFS-fa minden gyökérből elérhető csúcsba tartalmaz egy legrövidebb utat a gyökérből, azaz a szélességi bejárás tekinthető egy legrövidebb utat kereső algoritmusnak is.

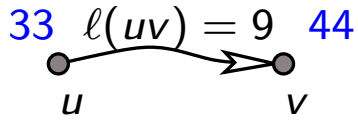
Legrövidebb utak

Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\ell(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\ell(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)
Az ℓ hosszfüggvény nemnegatív, ha $\ell(e) \geq 0$ teljesül minden e élre.
Az ℓ hosszfv konzervatív, ha G-ben $\nexists$ negatív összhosszú ir. kör.
Cél: Legrövidebb út keresése irányított/irányítatlan gráfban.
Megf: Ha $\ell(e) = 1$ a G minden e élére, akkor $\ell(P)$ a P élszáma. Ezért a BFS-fa minden gyökérből elérhető csúcsba tartalmaz egy legrövidebb utat a gyökérből, azaz a szélességi bejárás tekinthető egy legrövidebb utat kereső algoritmusnak is.
Def: Adott G (ir) gráf, $\ell : E(G) \to \mathbb{R}$ hosszfv. és $r \in V(G)$.
(r, ℓ)-felső becslés olyan $f : V(G) \to \mathbb{R}$ függvény, ami felülről becsli minden csúcs r-től mért távolságát: $dist_\ell(r, v) \leq f(v)\ \forall v \in V(G)$.

Legrövidebb utak

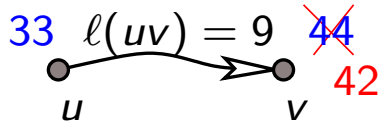
Def: Adott G (ir) gráf és $\ell : E(G) \to \mathbb{R}$ hosszfüggvény esetén egy P út hossza a P éleinek összhossza: $\ell(P) = \sum_{e \in E(P)} \ell(e)$.
Az u és v csúcsok távolsága a legrövidebb uv-út hossza:
$dist_\ell(u, v) := \min\{\ell(P) : P \ uv\text{-út}\}$ ($\nexists uv$-út $\Rightarrow dist_\ell(u, v) = \infty$.)
Az ℓ hosszfüggvény nemnegatív, ha $\ell(e) \geq 0$ teljesül minden e élre.
Az ℓ hosszfv konzervatív, ha G-ben $\nexists$ negatív összhosszú ir. kör.
Cél: Legrövidebb út keresése irányított/irányítatlan gráfban.
Megf: Ha $\ell(e) = 1$ a G minden e élére, akkor $\ell(P)$ a P élszáma. Ezért a BFS-fa minden gyökérből elérhető csúcsba tartalmaz egy legrövidebb utat a gyökérből, azaz a szélességi bejárás tekinthető egy legrövidebb utat kereső algoritmusnak is.
Def: Adott G (ir) gráf, $\ell : E(G) \to \mathbb{R}$ hosszfv. és $r \in V(G)$.
(r, ℓ)-felső becslés olyan $f : V(G) \to \mathbb{R}$ függvény, ami felülről becsli minden csúcs r-től mért távolságát: $dist_\ell(r, v) \leq f(v)\ \forall v \in V(G)$.
Triviális (r, ℓ)-felső becslés: $f(v) = \begin{cases} 0 & v = r \\ \infty & v \neq r \end{cases}$.
Pontos (r, ℓ)-felső becslés: $f(v) = dist_\ell(r, v)\ \forall v \in V(G)$.

Az élmenti javítás



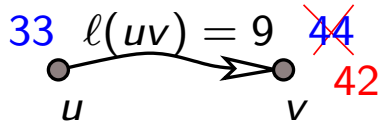
Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Az élmenti javítás



Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Az élmenti javítás

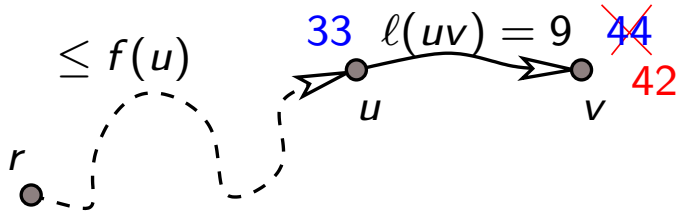


Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv konzervatív és $f(r) = 0$.

Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.

Az élmenti javítás

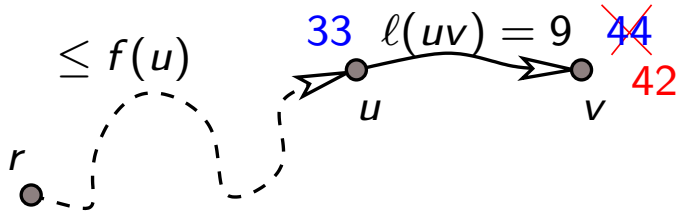


Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv konzervatív és $f(r) = 0$.
Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.

Biz: Azt kell megmutatni, hogy van olyan rv-út, aminek a hossza legfeljebb $f(u) + \ell(uv)$. Ha egy legrövidebb ru-utat kiegészítünk az uv éllel, akkor olyan rv-élsorozatot kapunk, aminek az összhossza $dist_\ell(r, u) + \ell(uv) \leq f(u) + \ell(uv)$. ,,Könnyen" látható, hogy az élhosszfv konzervativitása miatt ha van x összhosszúságú rv-élsorozat, akkor van legfeljebb x összhosszúságú rv-út is. Ezek szerint van legfeljebb $f(u) + \ell(u, v)$ hosszúságú uv-út is, azaz az émj után szintén (r, ℓ)-fb-t kapunk. □

Az élmenti javítás

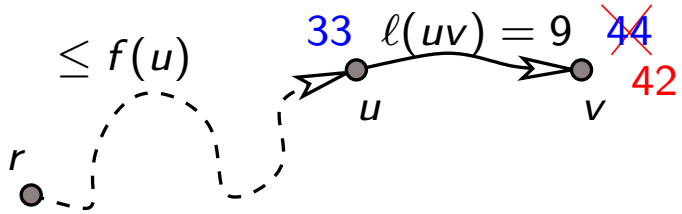


Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f *uv*-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv konzervatív és $f(r) = 0$.

Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.

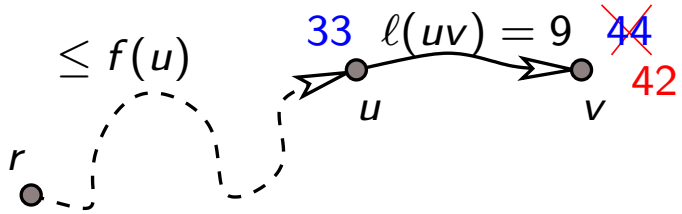
Az élmenti javítás



Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f *uv*-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv konzervatív és $f(r) = 0$.
Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.
(2) f (r, ℓ)-fb (pontos) $\iff$ (f-en $\nexists$ érdemi élmenti javítás).

Az élmenti javítás

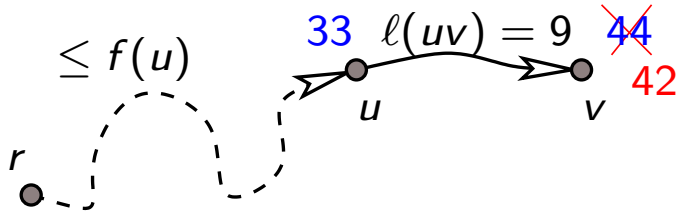


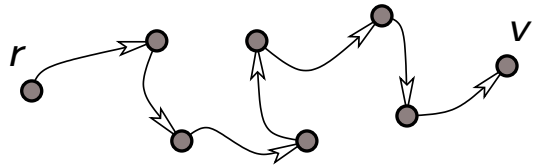
Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f *uv*-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv konzervatív és $f(r) = 0$.
Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.
(2) f (r, ℓ)-fb (pontos) $\iff$ (f-en $\nexists$ érdemi élmenti javítás).

Biz: $\Rightarrow$: Ha f pontos, akkor biztosan nincs rajta érdmei élmenti javítás: ha volna, akkor egy felső becslés a pontos érték alá csökkenne, így az élmenti javítás nem (r, ℓ)-fb-t eredményezne.

Az élmenti javítás



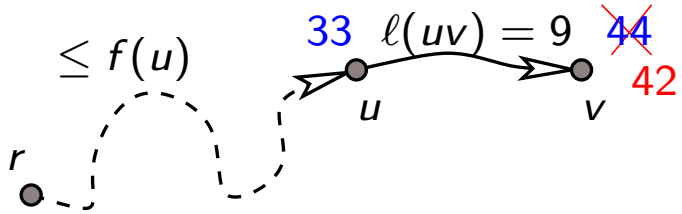


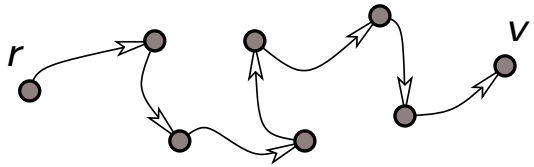
Def: Tfh f egy (r,ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u)+\ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv konzervatív és $f(r) = 0$.
Ekkor (1) Az f (r,ℓ)-fb élmenti javítása mindig (r,ℓ)-fb-t ad.
(2) f (r,ℓ)-fb (pontos) $\iff$ (f-en $\nexists$ érdemi élmenti javítás).

Biz: $\Rightarrow$: Ha f pontos, akkor biztosan nincs rajta érdmei élmenti javítás: ha volna, akkor egy felső becslés a pontos érték alá csökkenne, így az élmenti javítás nem (r,ℓ)-fb-t eredményezne.
$\Leftarrow$: Legyen $v \in V(G)$ tetsz, és legyen P egy legrövidebb rv-út. A P egyik éle mentén sincs érdemi élmenti javítás, ezért P minden u csúcsára pontos a felső becslés: $f(u) = dist_\ell(r,u)$. Ez igaz az út utolsó csúcsára, a tetszőlegesen választott v-re is. □

Az élmenti javítás

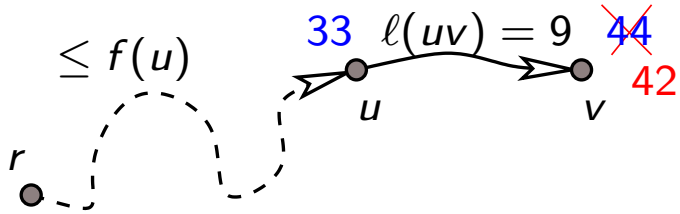


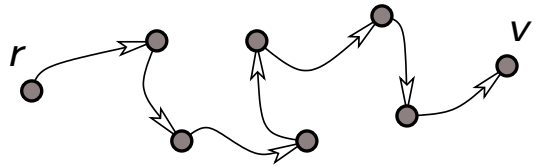


Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f *uv*-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv konzervatív és $f(r) = 0$.
Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.
(2) f (r, ℓ)-fb (pontos) $\iff$ (f-en $\nexists$ érdemi élmenti javítás).

Az élmenti javítás





Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

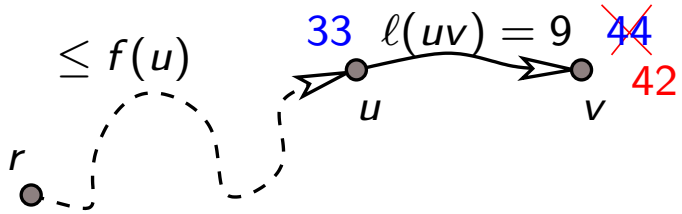
Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv konzervatív és $f(r) = 0$.
Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.
(2) f (r, ℓ)-fb (pontos) $\iff$ (f-en $\nexists$ érdemi élmenti javítás).

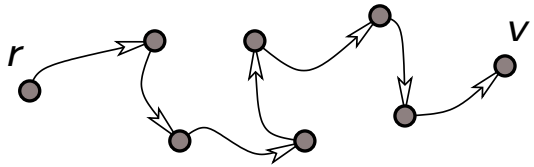
Köv: Adott G, konzervatív ℓ és $r \in V(G)$ esetén ha kiindulunk a triviális (r, ℓ)-fb-ből, és addig végzünk émj-kat, amíg lehet, akkor a végén megkapjuk minden csúcs r-től való távolságát.

Kérdés: Milyen sorrendben végezzül az émj-kat, ha garantáltan gyorsan kell végeznünk a feladattal?

Bemelegítés Fontos spec. eset: nemnegatív élhosszok.

Az élmenti javítás

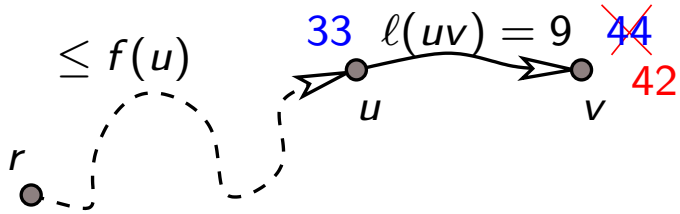


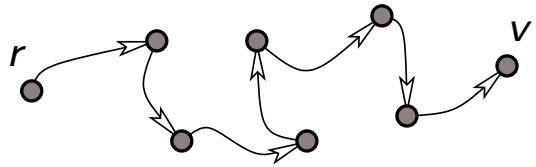


Def: Tfh f egy (r, ℓ)-fb és $uv \in E(G)$. Az f *uv*-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u) + \ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv konzervatív és $f(r) = 0$.
Ekkor (1) Az f (r, ℓ)-fb élmenti javítása mindig (r, ℓ)-fb-t ad.
(2) f (r, ℓ)-fb (pontos) $\iff$ (f-en $\nexists$ érdemi élmenti javítás).

Az élmenti javítás



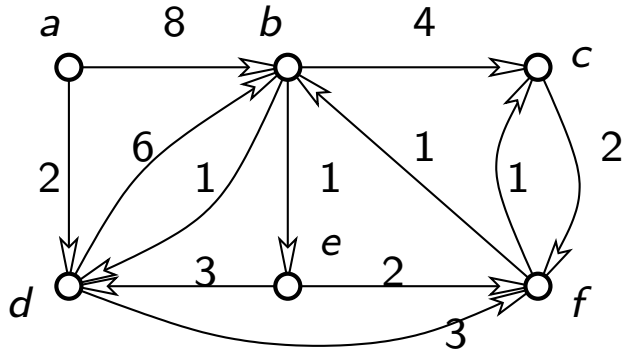


Def: Tfh f egy (r,ℓ)-fb és $uv \in E(G)$. Az f uv-élmenti javítása az az f', amire $f'(z) = \begin{cases} f(z) & z \neq v \\ \min\{f(v), f(u)+\ell(uv)\} & z = v \end{cases}$

Megf: Tfh az $\ell : E(G) \to \mathbb{R}$ hosszfv konzervatív és $f(r) = 0$.
Ekkor (1) Az f (r,ℓ)-fb élmenti javítása mindig (r,ℓ)-fb-t ad.
(2) f (r,ℓ)-fb (pontos) $\iff$ (f-en $\nexists$ érdemi élmenti javítás).

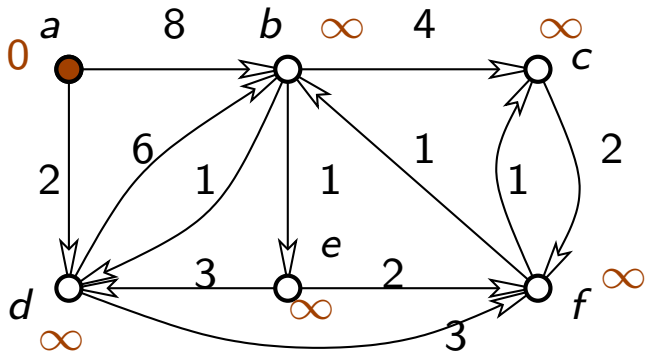
Dijkstra-algoritmus: Input: $G = (V,E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r,v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r,ℓ)-fb.
Az i-dik fázis:
1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.
Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:
1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.
Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



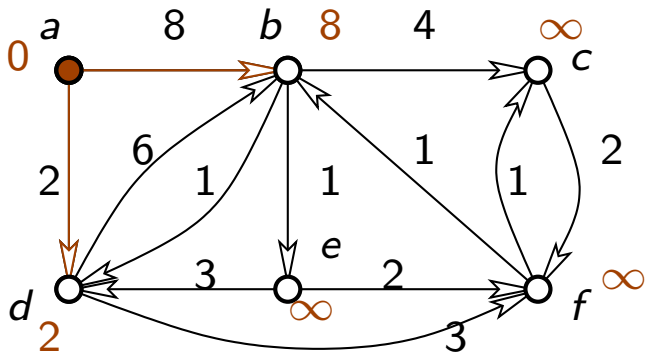
a	b	c	d	e	f
0	∞	∞	∞	∞	∞

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



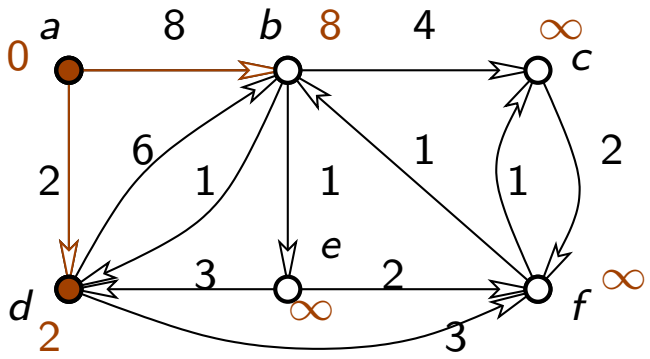
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



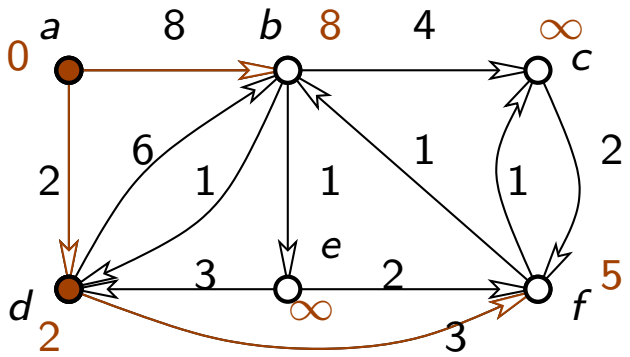
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



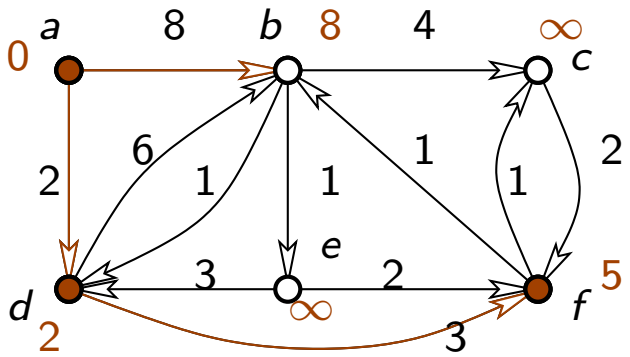
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



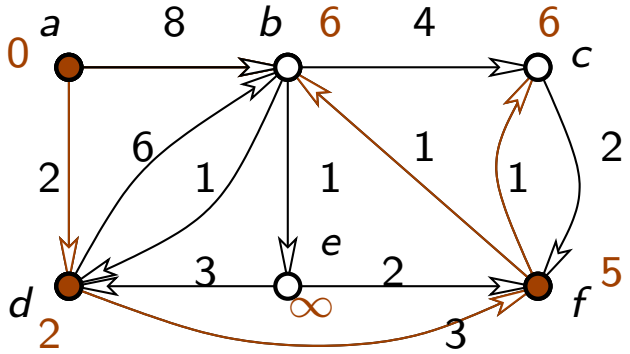
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



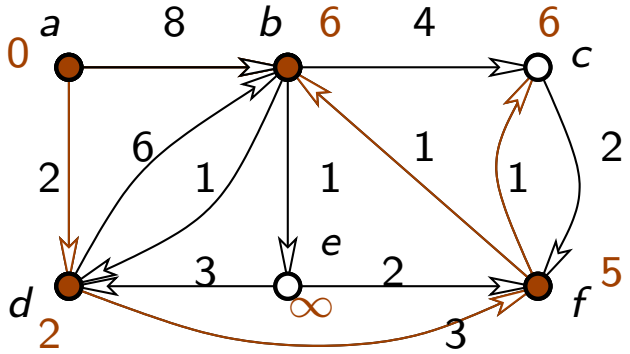
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



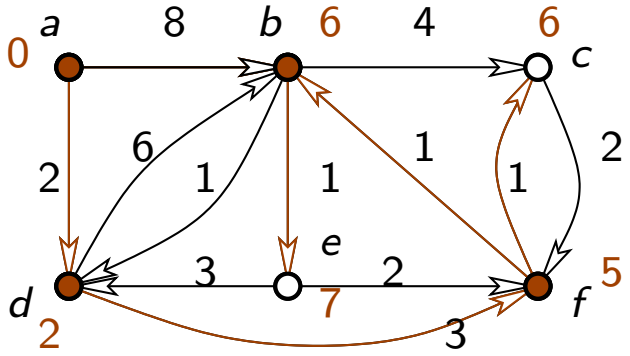
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



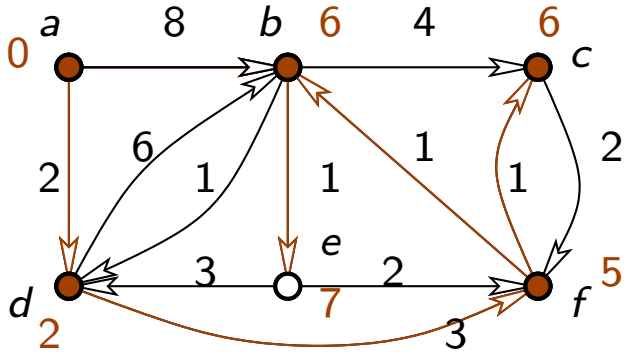
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5
0	6	6	2	7	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



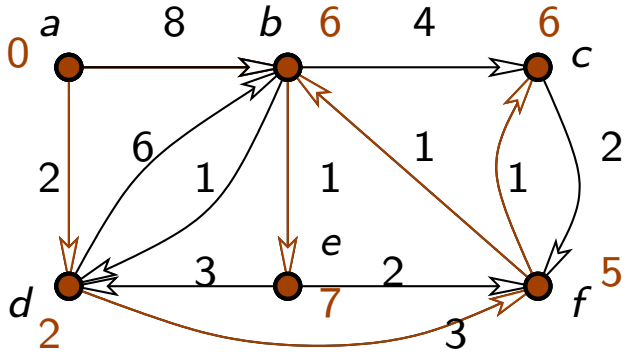
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5
0	6	6	2	7	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



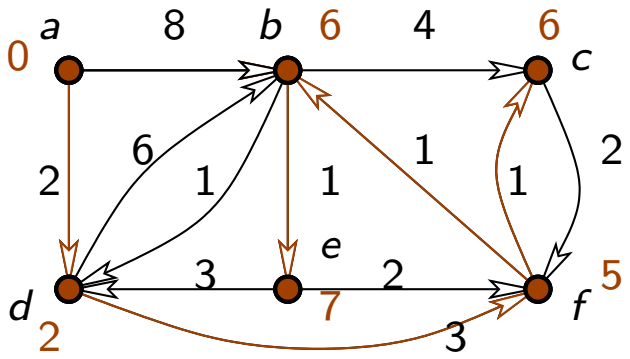
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5
0	6	6	2	7	5
0	6	6	2	7	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Dijkstra, egy példán



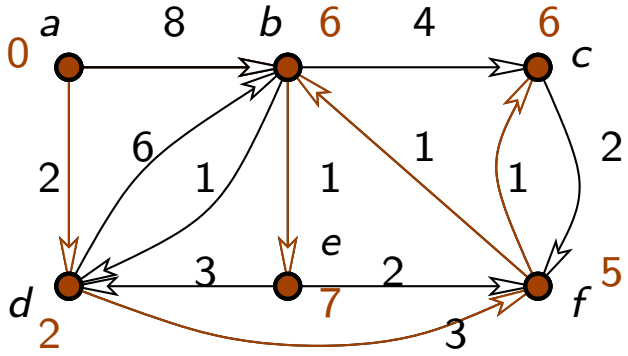
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5
0	6	6	2	7	5
0	6	6	2	7	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.
Megf: Ha v-be vezet megjelölt él, akkor vezet r-ből v-be megjelölt éleken út, és ennek hozza megegyezik $f_{|V|}(v)$-vel.

Dijkstra, egy példán



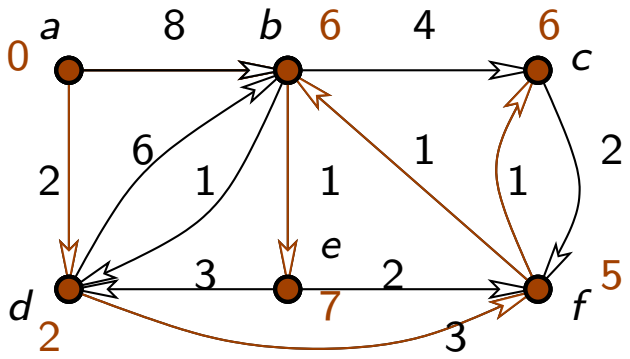
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5
0	6	6	2	7	5
0	6	6	2	7	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.
Megf: Ha v-be vezet megjelölt él, akkor vezet r-ből v-be megjelölt éleken út, és ennek hozza megegyezik $f_{|V|}(v)$-vel.
Biz: $f_{|V|}(r) = 0$, és a megjelölt élek mentén haladva az $f_{|V|}$ érték az élhosszal növekszik. □

Dijkstra, egy példán



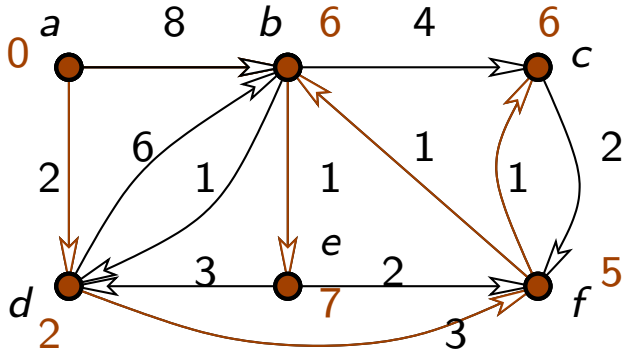
a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5
0	6	6	2	7	5
0	6	6	2	7	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.
Megf: Ha v-be vezet megjelölt él, akkor vezet r-ből v-be megjelölt éleken út, és ennek hozza megegyezik $f_{|V|}(v)$-vel.

Dijkstra, egy példán



a	b	c	d	e	f
0	∞	∞	∞	∞	∞
0	8	∞	2	∞	∞
0	8	∞	2	∞	5
0	6	6	2	∞	5
0	6	6	2	7	5
0	6	6	2	7	5

Dijkstra-algoritmus: Input: $G = (V, E)$, $\ell : E \to \mathbb{R}_+$, $r \in V$.
Output: $dist_\ell(r, v)$ $\forall v \in V$ Működés: $U_0 := \emptyset$, f_0 a triv. (r, ℓ)-fb.
Az i-dik fázis:

1. Legyen $U_i := U_{i-1} \cup \{u_i\}$, ahol u_i olyan csúcs a $V \setminus U_{i-1}$ halmazból, amelyre $f_{i-1}(v)$ minimális.
2. f_i: f_{i-1} élmenti javítása minden U_i-ből kivezető $u_i x$ élen.

Output: $f_{|V|}$. Megjelöljük a végső $f_{|V|}(v)$ értékeket beállító éleket.

Megf: Ha v-be vezet megjelölt él, akkor vezet r-ből v-be megjelölt éleken út, és ennek hozza megegyezik $f_{|V|}(v)$-vel.

Köv: Ha a Dijkstra-algoritmus helyes, akkor az algoritmus végén a megjelölt élek egy legrövidebb utak fáját alkotják r gyökérrel.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$.
Biz: Az i-dik fázisban $f_i(u_i) \leq f_i(u_{i+1})$ teljesült az u_i választása miatt. Ezek után $f_i(u_i)$ már nem változott: $f_{|V|}(u_i) = f_i(u_i)$.
Ugyan $f_i(u_{i+1})$ még csökkenhetett, de csak az $u_i u_{i+1}$ él mentén történt javítás miatt, hiszen az $(i+1)$-dik fázisban u_{i+1} bekerül az U_i halmazba, és a hozzá tartozó (r, ℓ)-fb már nem csökken tovább.
Ekkor $f_{i+1}(u_{i+1}) = \min\{f_i(u_{i+1}), f_i(u_i) + \ell(u_i u_{i+1})\} \geq f_i(u_i)$, mivel $\ell(u_i u_{i+1}) > 0$. Ezért $f_{|V|}(u_i) = f_i(u_i) \leq f_{i+1}(u_{i+1}) = f_{|V|}(u_{i+1})$ □

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.
Biz: Tegyük fel, hogy $u_iu_j \in E(G)$ a G egy tetsz. éle. Ha $i > j$, akkor (2) miatt $f_{|V|}(u_i) \geq f_{|V|}(u_j)$, ezért az u_iu_j mentén történő javítás nem tudja $f_{|V|}(u_j)$-t csökkenteni, hisz $\ell(u_iu_j)$ pozitív.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.

Biz: Tegyük fel, hogy $u_iu_j \in E(G)$ a G egy tetsz. éle. Ha $i > j$, akkor (2) miatt $f_{|V|}(u_i) \geq f_{|V|}(u_j)$, ezért az u_iu_j mentén történő javítás nem tudja $f_{|V|}(u_j)$-t csökkenteni, hisz $\ell(u_iu_j)$ pozitív.
Ha pedig $i < j$, akkor az i-dik fázisban megtörtént az u_iu_j mentén történő javítás, és ezt követően $f(u_i)$ nem változott, azaz $f_{|V|}(u_i) = f_i(u_i)$. A másik (r, ℓ)-fb pedig csak tovább csökkenhetett a későbbi émj-ok során $f_{|V|}(u_j) \leq f_i(u_j)$. Ezért az u_iu_j él mentén sem az i-dik fázisban, sem később nincs érdemi javítás. □

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.

Tétel: A Dijkstra algoritmus helyesen működik, azaz G minden csúcsára igaz, hogy $dist(r, v) = f_{|V|}(v)$.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.

Tétel: A Dijkstra algoritmus helyesen működik, azaz G minden csúcsára igaz, hogy $dist(r, v) = f_{|V|}(v)$.

Biz: A Dijkstra-algoritmus az f_0 triviális (r, ℓ)-fb-ből indul ki, és élmenti javításokat alkalmaz. Így minden f_i (speciálisan $f_{|V|}$ is) (r, ℓ)-fb lesz. A fenti (3)-as megfigyelés miatt $f_{|V|}$-n nem végezhető érdemi élmenti javítás. Ezért egy korábbi (2)-es megfigyelés miatt $f_{|V|}$ pontos (r, ℓ)-fb, azaz $f_{|V|}(v) = dist_\ell(r, v)$ $\forall v \in V(G)$. □

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.

Tétel: A Dijkstra algoritmus helyesen működik, azaz G minden csúcsára igaz, hogy $dist(r, v) = f_{|V|}(v)$.

Dijkstra helyessége

Megf: Tfh $u_1, u_2, \ldots, u_n$ a G csúcsainak sorrendje a Dijkstra-algoritmus végrehajtása után.
(1) Ekkor $f_{|V|}(u_i) \leq f_{|V|}(u_{i+1})$ teljesül $\forall 1 \leq i \leq n$.
(2) $f_{|V|}(u_1) \leq f_{|V|}(u_2) \leq \ldots \leq f_{|V|}(u_n)$
(3) A Dijkstra-algoritmus outputjaként kapott $f_{|V|}$-n élmenti javítás nem tud változtatni.

Tétel: A Dijkstra algoritmus helyesen működik, azaz G minden csúcsára igaz, hogy $dist(r, v) = f_{|V|}(v)$.

„Lépésszámanalízis”: Ha a G gráfnak n csúcsa és m éle van, akkor a Dijkstra-algoritmus n-szer keresi meg legfeljebb n szám minimumát, ami összességében legfeljebb $konst \cdot n^2$ lépést igényel. Ezen kívül legfeljebb m élmenti javítást végez, ami $konst' \cdot m$ lépés. Összességében tehát legfeljebb $konst'' \cdot (n^2 + m)$ lépésre van szükség, az algoritmus hatékony.

Mit tanultunk ma?

- Gráfbejárás fogalma
  (csúcsok evolúciója, élek osztályozása, bejárás fája)
- BFS (elérési és befejezési sorrend megegyezik, nincs se faélt átugró gráfél, se előreél, de van legrövidebb utak fája)
- Legrövidebb út keresése (r, ℓ)-fb élmenti javításával, Dijkstra

Mit tanultunk ma?

- Gráfbejárás fogalma
  (csúcsok evolúciója, élek osztályozása, bejárás fája)
- BFS (elérési és befejezési sorrend megegyezik, nincs se faélt átugró gráfél, se előreél, de van legrövidebb utak fája)
- Legrövidebb út keresése (r, ℓ)-fb élmenti javításával, Dijkstra

Köszönöm a figyelmet!