Mohó algoritmusok

Hajnal Péter

Bolyai Intézet, TTIK, SZTE, Szeged

2021. ősz

Alapötlet

Alapötlet

• Nagyon sok optimalizálási probléma esetén az output egyedi döntések sorozataként alakul ki.

Alapötlet

- Nagyon sok optimalizálási probléma esetén az output egyedi döntések sorozataként alakul ki.
- Természetes, hogy az aktuális döntés esetén az akkor legjobbnak tűnő döntést hozzuk meg.

Alapötlet

- Nagyon sok optimalizálási probléma esetén az output egyedi döntések sorozataként alakul ki.
- Természetes, hogy az aktuális döntés esetén az akkor legjobbnak tűnő döntést hozzuk meg.
- Ez nem szükségszerűen vezet el a legjobb outputhoz.

Alapötlet

• Nagyon sok optimalizálási probléma esetén az output egyedi döntések sorozataként alakul ki.
• Természetes, hogy az aktuális döntés esetén az akkor legjobbnak tűnő döntést hozzuk meg.
• Ez nem szükségszerűen vezet el a legjobb outputhoz. Gyakran felismerhetjük, hogy egy helyileg optimális döntésünket felül kell értékelnünk és egy kevésbé jó irányba kellett volna haladnunk, ami később hozza meg a gyümölcsét.

Alapötlet

• Nagyon sok optimalizálási probléma esetén az output egyedi döntések sorozataként alakul ki.
• Természetes, hogy az aktuális döntés esetén az akkor legjobbnak tűnő döntést hozzuk meg.
• Ez nem szükségszerűen vezet el a legjobb outputhoz. Gyakran felismerhetjük, hogy egy helyileg optimális döntésünket felül kell értékelnünk és egy kevésbé jó irányba kellett volna haladnunk, ami később hozza meg a gyümölcsét.
• Egy algoritmust akkor nevezünk mohó algoritmusnak ha sose „visszakozik”. Olyan döntéssorozaton keresztül számol ki egy outputot, amely minden lépésben a legjobbnak tűnik, és ezen döntésekhez végig ragaszkodik.

Alapötlet

• Nagyon sok optimalizálási probléma esetén az output egyedi döntések sorozataként alakul ki.
• Természetes, hogy az aktuális döntés esetén az akkor legjobbnak tűnő döntést hozzuk meg.
• Ez nem szükségszerűen vezet el a legjobb outputhoz. Gyakran felismerhetjük, hogy egy helyileg optimális döntésünket felül kell értékelnünk és egy kevésbé jó irányba kellett volna haladnunk, ami később hozza meg a gyümölcsét.
• Egy algoritmust akkor nevezünk mohó algoritmusnak ha sose „visszakozik”. Olyan döntéssorozaton keresztül számol ki egy outputot, amely minden lépésben a legjobbnak tűnik, és ezen döntésekhez végig ragaszkodik.
• Példákon keresztül ismerhetjük meg ezen algoritmus tervezési elvet.

Szünet

Emlékeztető: Feszítőfák

Emlékeztető: Feszítőfák

Definíció: Feszítő részgráf

S a G gráf feszítő részgráfja G-nek, ha S csak élek elhagyásával megkapható G-ből.

Emlékeztető: Feszítőfák

Definíció: Feszítő részgráf

S a G gráf feszítő részgráfja G-nek, ha S csak élek elhagyásával megkapható G-ből.
Azaz: S pontosan akkor feszítő részgráf, ha egy részgráf és $V(S) = V(G)$.

Emlékeztető: Feszítőfák

Definíció: Feszítő részgráf

S a G gráf feszítő részgráfja G-nek, ha S csak élek elhagyásával megkapható G-ből.
Azaz: S pontosan akkor feszítő részgráf, ha egy részgráf és $V(S) = V(G)$.

Észrevétel

Ha G-nek van feszítő fa részgráfja, akkor összefüggő.

Emlékeztető: Feszítőfák

Definíció: Feszítő részgráf

S a G gráf feszítő részgráfja G-nek, ha S csak élek elhagyásával megkapható G-ből.
Azaz: S pontosan akkor feszítő részgráf, ha egy részgráf és $V(S) = V(G)$.

Észrevétel

Ha G-nek van feszítő fa részgráfja, akkor összefüggő.

Segédtétel

G akkor és csak akkor összefüggő, ha G-nek van feszítő fa részgráfja.

Emlékeztető: Feszítőfák

Definíció: Feszítő részgráf

S a G gráf feszítő részgráfja G-nek, ha S csak élek elhagyásával megkapható G-ből.
Azaz: S pontosan akkor feszítő részgráf, ha egy részgráf és $V(S) = V(G)$.

Észrevétel

Ha G-nek van feszítő fa részgráfja, akkor összefüggő.

Segédtétel

G akkor és csak akkor összefüggő, ha G-nek van feszítő fa részgráfja.

Definíció: Feszítőfa.

F a G gráf feszítőfája, ha egy feszítő fa részgráfja G-nek.

Ismeretek a fagráfokról

Ismeretek a fagráfokról

Lemma

Egy legalább kétpontú fában legalább két 1-fokú csúcs van.

Ismeretek a fagráfokról

Lemma

Egy legalább kétpontú fában legalább két 1-fokú csúcs van. //
Egy fa 1-fokú csúcsait leveleknek nevezzük.

Ismeretek a fagráfokról

Lemma

Egy legalább kétpontú fában legalább két 1-fokú csúcs van. //
Egy fa 1-fokú csúcsait leveleknek nevezzük.

Tétel

Egy n pontú fának pontosan $n-1$ éle van.

Ismeretek a fagráfokról

Lemma

Egy legalább kétpontú fában legalább két 1-fokú csúcs van. //
Egy fa 1-fokú csúcsait leveleknek nevezzük.

Tétel

Egy n pontú fának pontosan $n-1$ éle van.

Tétel*

Egy n pontú, c komponensű körmentes (erdő) gráfnak pontosan $n-c$ éle van.

Ismeretek a fagráfokról

Lemma

Egy legalább kétpontú fában legalább két 1-fokú csúcs van. //
Egy fa 1-fokú csúcsait leveleknek nevezzük.

Tétel

Egy n pontú fának pontosan $n-1$ éle van.

Tétel*

Egy n pontú, c komponensű körmentes (erdő) gráfnak pontosan $n-c$ éle van.

Észrevétel

Az n pontú teljes gráfnak n^{n-2} feszítőfája van

Ismeretek a fagráfokról

Lemma

Egy legalább kétpontú fában legalább két 1-fokú csúcs van. //
Egy fa 1-fokú csúcsait leveleknek nevezzük.

Tétel

Egy n pontú fának pontosan $n-1$ éle van.

Tétel*

Egy n pontú, c komponensű körmentes (erdő) gráfnak pontosan $n-c$ éle van.

Észrevétel

Az n pontú teljes gráfnak n^{n-2} feszítőfája van (Cayley-tétel),

Ismeretek a fagráfokról

Lemma

Egy legalább kétpontú fában legalább két 1-fokú csúcs van. //
Egy fa 1-fokú csúcsait leveleknek nevezzük.

Tétel

Egy n pontú fának pontosan $n-1$ éle van.

Tétel*

Egy n pontú, c komponensű körmentes (erdő) gráfnak pontosan $n-c$ éle van.

Észrevétel

Az n pontú teljes gráfnak n^{n-2} feszítőfája van (Cayley-tétel), de mindegyiknek ugyanannyi/$n-1$ éle van.

Az alapkérdés

Az alapkérdés

- Legyen G egy nem negatív élköltségekkel rendelkező gráf: $c : E(G) \to \mathbb{R}_+$.

Az alapkérdés

- Legyen G egy nem negatív élköltségekkel rendelkező gráf: $c : E(G) \to \mathbb{R}_+$.

- c természetes módon kiterjeszthető élhalmazokra: $R \subset E(G)$ esetén $c(R) = \sum_{e:e\in R} c(e)$.

Az alapkérdés

- Legyen G egy nem negatív élköltségekkel rendelkező gráf: $c : E(G) \to \mathbb{R}_+$.

- c természetes módon kiterjeszthető élhalmazokra: $R \subset E(G)$ esetén $c(R) = \sum_{e:e\in R} c(e)$.

Legkisebb költségű feszítőfa meghatározása

Adott egy G összefüggő (egyszerű) gráf. Keressük meg a legolcsóbb összefüggő FESZÍTŐ részgráfját.

Az alapkérdés

- Legyen G egy nem negatív élköltségekkel rendelkező gráf: $c : E(G) \to \mathbb{R}_+$.

- c természetes módon kiterjeszthető élhalmazokra: $R \subset E(G)$ esetén $c(R) = \sum_{e:e\in R} c(e)$.

Legkisebb költségű feszítőfa meghatározása

Adott egy G összefüggő (egyszerű) gráf. Keressük meg a legolcsóbb összefüggő FESZÍTŐ részgráfját.

- Az optimumot nyilvánvalóan egy feszítőfa élhalmaza adja.

Az alapkérdés

- Legyen G egy nem negatív élköltségekkel rendelkező gráf: $c : E(G) \to \mathbb{R}_+$.

- c természetes módon kiterjeszthető élhalmazokra: $R \subset E(G)$ esetén $c(R) = \sum_{e:e\in R} c(e)$.

Legkisebb költségű feszítőfa meghatározása

Adott egy G összefüggő (egyszerű) gráf. Keressük meg a legolcsóbb összefüggő FESZÍTŐ részgráfját.

- Az optimumot nyilvánvalóan egy feszítőfa élhalmaza adja. (Miért?

Az alapkérdés

- Legyen G egy nem negatív élköltségekkel rendelkező gráf: $c : E(G) \to \mathbb{R}_+$.

- c természetes módon kiterjeszthető élhalmazokra: $R \subset E(G)$ esetén $c(R) = \sum_{e:e\in R} c(e)$.

Legkisebb költségű feszítőfa meghatározása

Adott egy G összefüggő (egyszerű) gráf. Keressük meg a legolcsóbb összefüggő FESZÍTŐ részgráfját.

- Az optimumot nyilvánvalóan egy feszítőfa élhalmaza adja. (Miért? Ugye emlékszünk, c nemnegatív költségfüggvény.)

Az alapötlet: Mohóság

Az alapötlet: Mohóság

- Kiindulunk az $F = \emptyset$ körmentes élhalmazból.

Az alapötlet: Mohóság

- Kiindulunk az $F = \emptyset$ körmentes élhalmazból. $H = E(G)$ az összes él halmaza, az eddig nem vizsgált/hátralévő élek halmaza.

Az alapötlet: Mohóság

• Kiindulunk az $F = \emptyset$ körmentes élhalmazból. $H = E(G)$ az összes él halmaza, az eddig nem vizsgált/hátralévő élek halmaza. F és H az algoritmus során dinamikusan változó élhalmazok.

Az alapötlet: Mohóság

• Kiindulunk az $F = \emptyset$ körmentes élhalmazból. $H = E(G)$ az összes él halmaza, az eddig nem vizsgált/hátralévő élek halmaza. F és H az algoritmus során dinamikusan változó élhalmazok. Az output, a leálláskori F.

Az alapötlet: Mohóság

• Kiindulunk az $F = \emptyset$ körmentes élhalmazból. $H = E(G)$ az összes él halmaza, az eddig nem vizsgált/hátralévő élek halmaza. F és H az algoritmus során dinamikusan változó élhalmazok. Az output, a leálláskori F.

• Minden lépésben H egy ígéretes élét kiválasztjuk.

Az alapötlet: Mohóság

• Kiindulunk az $F = \emptyset$ körmentes élhalmazból. $H = E(G)$ az összes él halmaza, az eddig nem vizsgált/hátralévő élek halmaza. F és H az algoritmus során dinamikusan változó élhalmazok. Az output, a leálláskori F.

• Minden lépésben H egy ígéretes élét kiválasztjuk. Megvizsgáljuk, vagy F-be beválasztjuk, vagy eldobjuk.

Az alapötlet: Mohóság

• Kiindulunk az $F = \emptyset$ körmentes élhalmazból. $H = E(G)$ az összes él halmaza, az eddig nem vizsgált/hátralévő élek halmaza. F és H az algoritmus során dinamikusan változó élhalmazok. Az output, a leálláskori F.

• Minden lépésben H egy ígéretes élét kiválasztjuk. Megvizsgáljuk, vagy F-be beválasztjuk, vagy eldobjuk.

• **Ígéretes:**

Az alapötlet: Mohóság

• Kiindulunk az $F = \emptyset$ körmentes élhalmazból. $H = E(G)$ az összes él halmaza, az eddig nem vizsgált/hátralévő élek halmaza. F és H az algoritmus során dinamikusan változó élhalmazok. Az output, a leálláskori F.

• Minden lépésben H egy ígéretes élét kiválasztjuk. Megvizsgáljuk, vagy F-be beválasztjuk, vagy eldobjuk.

• **Ígéretes:** H legolcsóbb éle.

Az alapötlet: Mohóság

• Kiindulunk az $F = \emptyset$ körmentes élhalmazból. $H = E(G)$ az összes él halmaza, az eddig nem vizsgált/hátralévő élek halmaza. F és H az algoritmus során dinamikusan változó élhalmazok. Az output, a leálláskori F.

• Minden lépésben H egy ígéretes élét kiválasztjuk. Megvizsgáljuk, vagy F-be beválasztjuk, vagy eldobjuk.

• **Ígéretes:** H legolcsóbb éle. // A természetes döntés.

Az alapötlet: Mohóság

• Kiindulunk az $F = \emptyset$ körmentes élhalmazból. $H = E(G)$ az összes él halmaza, az eddig nem vizsgált/hátralévő élek halmaza. F és H az algoritmus során dinamikusan változó élhalmazok. Az output, a leálláskori F.

• Minden lépésben H egy ígéretes élét kiválasztjuk. Megvizsgáljuk, vagy F-be beválasztjuk, vagy eldobjuk.

• **Ígéretes:** H legolcsóbb éle. // A természetes döntés.

• **Mohóság:**

Az alapötlet: Mohóság

• Kiindulunk az $F = \emptyset$ körmentes élhalmazból. $H = E(G)$ az összes él halmaza, az eddig nem vizsgált/hátralévő élek halmaza. F és H az algoritmus során dinamikusan változó élhalmazok. Az output, a leálláskori F.

• Minden lépésben H egy ígéretes élét kiválasztjuk. Megvizsgáljuk, vagy F-be beválasztjuk, vagy eldobjuk.

• **Ígéretes:** H legolcsóbb éle. // A természetes döntés.

• **Mohóság:** F csak nőhet.

Az alapötlet: Mohóság

• Kiindulunk az $F = \emptyset$ körmentes élhalmazból. $H = E(G)$ az összes él halmaza, az eddig nem vizsgált/hátralévő élek halmaza. F és H az algoritmus során dinamikusan változó élhalmazok. Az output, a leálláskori F.

• Minden lépésben H egy ígéretes élét kiválasztjuk. Megvizsgáljuk, vagy F-be beválasztjuk, vagy eldobjuk.

• **Ígéretes:** H legolcsóbb éle. // A természetes döntés.

• **Mohóság:** F csak nőhet. Korábbi beválasztásainakat nem bíráljuk felül.

Kruskal-algoritmus

Kruskal-algoritmus

Kruskal-algoritmus (1956)

Kruskal-algoritmus

Kruskal-algoritmus (1956)

(I) // Inicializálás

Kruskal-algoritmus

Kruskal-algoritmus (1956)

(I) // Inicializálás
$F = \emptyset$

Kruskal-algoritmus

Kruskal-algoritmus (1956)

(I) // Inicializálás
$F = \emptyset$

(R) // Rendezési lépés

Kruskal-algoritmus

Kruskal-algoritmus (1956)

(I) // Inicializálás
$F = \emptyset$

(R) // Rendezési lépés
Rendezzük sorba $E(G)$ elemeit növekvő költségek szerint:

$$E(G): \quad e_1, e_2, e_3, \ldots, e_{m-1}, e_m,$$

ahol $c(e_1) \leq c(e_2) \leq c(e_3) \leq \ldots \leq c(e_{m-1}) \leq c(e_m)$.

Kruskal-algoritmus

Kruskal-algoritmus (1956)

(I) // Inicializálás
$F = \emptyset$

(R) // Rendezési lépés
Rendezzük sorba $E(G)$ elemeit növekvő költségek szerint:

$$E(G): \quad e_1, e_2, e_3, \ldots, e_{m-1}, e_m,$$

ahol $c(e_1) \leq c(e_2) \leq c(e_3) \leq \ldots \leq c(e_{m-1}) \leq c(e_m)$.

(D) // Döntési lépések

Kruskal-algoritmus

Kruskal-algoritmus (1956)

(I) // Inicializálás
$F = \emptyset$

(R) // Rendezési lépés
Rendezzük sorba $E(G)$ elemeit növekvő költségek szerint:

$$E(G): \quad e_1, e_2, e_3, \ldots, e_{m-1}, e_m,$$

ahol $c(e_1) \leq c(e_2) \leq c(e_3) \leq \ldots \leq c(e_{m-1}) \leq c(e_m)$.

(D) // Döntési lépések
$i = 1, 2, 3, \ldots, m-1, m$ esetén vizsgáljuk meg e_i-t:

Kruskal-algoritmus

Kruskal-algoritmus (1956)

(I) // Inicializálás
$F = \emptyset$

(R) // Rendezési lépés
Rendezzük sorba $E(G)$ elemeit növekvő költségek szerint:

$$E(G): \quad e_1, e_2, e_3, \ldots, e_{m-1}, e_m,$$

ahol $c(e_1) \leq c(e_2) \leq c(e_3) \leq \ldots \leq c(e_{m-1}) \leq c(e_m)$.

(D) // Döntési lépések
$i = 1, 2, 3, \ldots, m-1, m$ esetén vizsgáljuk meg e_i-t:

Kruskal-algoritmus

Kruskal-algoritmus (1956)

(I) // Inicializálás
$F = \emptyset$

(R) // Rendezési lépés
Rendezzük sorba $E(G)$ elemeit növekvő költségek szerint:

$$E(G): \quad e_1, e_2, e_3, \ldots, e_{m-1}, e_m,$$

ahol $c(e_1) \leq c(e_2) \leq c(e_3) \leq \ldots \leq c(e_{m-1}) \leq c(e_m)$.

(D) // Döntési lépések
$i = 1, 2, 3, \ldots, m-1, m$ esetén vizsgáljuk meg e_i-t:

- Ha $F \cup \{e_i\}$ körmentes, akkor $F \leftarrow F \cup \{e_i\}$. Ha $i < m$, akkor $i \leftarrow i + 1$.

Kruskal-algoritmus

Kruskal-algoritmus (1956)

(I) // Inicializálás
$F = \emptyset$

(R) // Rendezési lépés
Rendezzük sorba $E(G)$ elemeit növekvő költségek szerint:

$$E(G): \quad e_1, e_2, e_3, \ldots, e_{m-1}, e_m,$$

ahol $c(e_1) \leq c(e_2) \leq c(e_3) \leq \ldots \leq c(e_{m-1}) \leq c(e_m)$.

(D) // Döntési lépések
$i = 1, 2, 3, \ldots, m-1, m$ esetén vizsgáljuk meg e_i-t:

- Ha $F \cup \{e_i\}$ körmentes, akkor $F \leftarrow F \cup \{e_i\}$. Ha $i < m$, akkor $i \leftarrow i + 1$.
- Ha $F \cup \{e_i\}$ tartalmaz kör élhalmazát, akkor F „marad”. Ha $i < m$, akkor $i \leftarrow i + 1$.

Az alaptétel

Az alaptétel

Tétel

A Kruskal algoritmus outputja egy optimális (minimális költségű) feszítőfa élhalmaza.

Az alaptétel

Tétel

A Kruskal algoritmus outputja egy optimális (minimális költségű) feszítőfa élhalmaza.

Legyen G, c egy tetszőleges összefüggő, élköltségekkel rendelkező gráf.

Az alaptétel

Tétel

A Kruskal algoritmus outputja egy optimális (minimális költségű) feszítőfa élhalmaza.

Legyen G, c egy tetszőleges összefüggő, élköltségekkel rendelkező gráf.

Jelölés

Legyen $o_1, o_2, \ldots, o_s$ a Kruskal-algoritmus által (ebben a sorrendben) kiválasztott első s él (o_i-k költség szerint növekvő sorrendben következnek).
Legyen $T = \{t_1, \ldots, t_S\}$ tetszőleges S-elemű körmentes élhalmaz, ahol az élek sorrendje a költség szerinti növekvő sorrend.

Az alaptétel élesítése

Az alaptétel élesítése

Tétel+

(i) Ha $|T| = S = s$, akkor $c(o_1) \leq c(t_1)$, $c(o_2) \leq c(t_2)$, $\ldots, c(o_s) \leq c(t_s)$,

(ii) Ha $|T| = S > s$, akkor a Kruskal-algoritmus o_s után még választ o_{s+1} élt.

Az alaptétel élesítése

Tétel+

(i) Ha $|T| = S = s$, akkor $c(o_1) \leq c(t_1)$, $c(o_2) \leq c(t_2)$, $\ldots, c(o_s) \leq c(t_s)$,

(ii) Ha $|T| = S > s$, akkor a Kruskal-algoritmus o_s után még választ o_{s+1} élt.

- A tételt

Az alaptétel élesítése

Tétel+

(i) Ha $|T| = S = s$, akkor $c(o_1) \leq c(t_1)$, $c(o_2) \leq c(t_2)$, $\ldots, c(o_s) \leq c(t_s)$,

(ii) Ha $|T| = S > s$, akkor a Kruskal-algoritmus o_s után még választ o_{s+1} élt.

- A tételt $(i)_s$ és $(ii)_s$ teljes indukcióval igazoljuk.

Az alaptétel élesítése

Tétel+

(i) Ha $|T| = S = s$, akkor $c(o_1) \leq c(t_1)$, $c(o_2) \leq c(t_2)$, $\ldots, c(o_s) \leq c(t_s)$,

(ii) Ha $|T| = S > s$, akkor a Kruskal-algoritmus o_s után még választ o_{s+1} élt.

- A tételt $(i)_s$ és $(ii)_s$ teljes indukcióval igazoljuk.
- Az állítás:

$$(i)_1 \Rightarrow (ii)_1 \Rightarrow (i)_2 \Rightarrow (ii)_2 \Rightarrow (i)_3 \Rightarrow (ii)_3 \Rightarrow \ldots$$

Az alaptétel élesítése

Tétel+

(i) Ha $|T| = S = s$, akkor $c(o_1) \leq c(t_1)$, $c(o_2) \leq c(t_2)$, $\ldots, c(o_s) \leq c(t_s)$,

(ii) Ha $|T| = S > s$, akkor a Kruskal-algoritmus o_s után még választ o_{s+1} élt.

- A tételt $(i)_s$ és $(ii)_s$ teljes indukcióval igazoljuk.
- Az állítás:

$$(i)_1 \Rightarrow (ii)_1 \Rightarrow (i)_2 \Rightarrow (ii)_2 \Rightarrow (i)_3 \Rightarrow (ii)_3 \Rightarrow \ldots$$

- Az $(i)_1$ eset nyilvánvaló.

Az alaptétel élesítése

Tétel+

(i) Ha $|T| = S = s$, akkor $c(o_1) \leq c(t_1)$, $c(o_2) \leq c(t_2)$, $\ldots, c(o_s) \leq c(t_s)$,

(ii) Ha $|T| = S > s$, akkor a Kruskal-algoritmus o_s után még választ o_{s+1} élt.

- A tételt $(i)_s$ és $(ii)_s$ teljes indukcióval igazoljuk.
- Az állítás:

$$(i)_1 \Rightarrow (ii)_1 \Rightarrow (i)_2 \Rightarrow (ii)_2 \Rightarrow (i)_3 \Rightarrow (ii)_3 \Rightarrow \ldots$$

- Az $(i)_1$ eset nyilvánvaló.
- Az indukció lépés egy LEMMÁN múlik.

A LEMMA és bizonyítása

A LEMMA és bizonyítása

Lemma

Legyen F, F' két körmentes élhalmaz a V, n elemű csúcshalmazon. Tegyük fel, hogy $|F| < |F'|$. Ekkor található olyan $F' - F$-beli e él. hogy $F \cup \{e\}$ is körmentes legyen.

A LEMMA és bizonyítása

Lemma

Legyen F, F' két körmentes élhalmaz a V, n elemű csúcshalmazon. Tegyük fel, hogy $|F| < |F'|$. Ekkor található olyan $F' - F$-beli e él. hogy $F \cup \{e\}$ is körmentes legyen.

- Legyen G_F az a körmentes gráf (erdő), amely csúcshalmaza V és élhalmaza F.

A LEMMA és bizonyítása

Lemma

Legyen F, F' két körmentes élhalmaz a V, n elemű csúcshalmazon. Tegyük fel, hogy $|F| < |F'|$. Ekkor található olyan $F' - F$-beli e él. hogy $F \cup \{e\}$ is körmentes legyen.

- Legyen G_F az a körmentes gráf (erdő), amely csúcshalmaza V és élhalmaza F.
- Ismert, hogy G_F-nek $n - |F|$ komponense van:

A LEMMA és bizonyítása

Lemma

Legyen F, F' két körmentes élhalmaz a V, n elemű csúcshalmazon. Tegyük fel, hogy $|F| < |F'|$. Ekkor található olyan $F' - F$-beli e él. hogy $F \cup \{e\}$ is körmentes legyen.

- Legyen G_F az a körmentes gráf (erdő), amely csúcshalmaza V és élhalmaza F.
- Ismert, hogy G_F-nek $n - |F|$ komponense van: $c(G_F) = n - |F|$.

A LEMMA és bizonyítása

Lemma

Legyen F, F' két körmentes élhalmaz a V, n elemű csúcshalmazon. Tegyük fel, hogy $|F| < |F'|$. Ekkor található olyan $F' - F$-beli e él. hogy $F \cup \{e\}$ is körmentes legyen.

- Legyen G_F az a körmentes gráf (erdő), amely csúcshalmaza V és élhalmaza F.
- Ismert, hogy G_F-nek $n - |F|$ komponense van: $c(G_F) = n - |F|$.
- Hasonlóan legyen $G_{F'}$ az a körmentes gráf (erdő), amely csúcshalmaza V és élhalmaza F'.

A LEMMA és bizonyítása

Lemma

Legyen F, F' két körmentes élhalmaz a V, n elemű csúcshalmazon. Tegyük fel, hogy $|F| < |F'|$. Ekkor található olyan $F' - F$-beli e él. hogy $F \cup \{e\}$ is körmentes legyen.

- Legyen G_F az a körmentes gráf (erdő), amely csúcshalmaza V és élhalmaza F.
- Ismert, hogy G_F-nek $n - |F|$ komponense van: $c(G_F) = n - |F|$.
- Hasonlóan legyen $G_{F'}$ az a körmentes gráf (erdő), amely csúcshalmaza V és élhalmaza F'.
- $G_{F'}$-nek $n - |F'|$ komponense van:

A LEMMA és bizonyítása

Lemma

Legyen F, F' két körmentes élhalmaz a V, n elemű csúcshalmazon. Tegyük fel, hogy $|F| < |F'|$. Ekkor található olyan $F' - F$-beli e él. hogy $F \cup \{e\}$ is körmentes legyen.

- Legyen G_F az a körmentes gráf (erdő), amely csúcshalmaza V és élhalmaza F.
- Ismert, hogy G_F-nek $n - |F|$ komponense van: $c(G_F) = n - |F|$.
- Hasonlóan legyen $G_{F'}$ az a körmentes gráf (erdő), amely csúcshalmaza V és élhalmaza F'.
- $G_{F'}$-nek $n - |F'|$ komponense van:

$c(G_{F'}) = n - |F'| < n - |F| = c(G_F)$.

A LEMMA és bizonyítása

Lemma

Legyen F, F' két körmentes élhalmaz a V, n elemű csúcshalmazon. Tegyük fel, hogy $|F| < |F'|$. Ekkor található olyan $F' - F$-beli e él. hogy $F \cup \{e\}$ is körmentes legyen.

- Legyen G_F az a körmentes gráf (erdő), amely csúcshalmaza V és élhalmaza F.
- Ismert, hogy G_F-nek $n - |F|$ komponense van: $c(G_F) = n - |F|$.
- Hasonlóan legyen $G_{F'}$ az a körmentes gráf (erdő), amely csúcshalmaza V és élhalmaza F'.
- $G_{F'}$-nek $n - |F'|$ komponense van:
$c(G_{F'}) = n - |F'| < n - |F| = c(G_F)$.
- Ez csak úgy képzelhető el, ha egy e F'-beli él G_F két különböző komponensének egy-egy csúcsát köti össze.

A LEMMA és bizonyítása

Lemma

Legyen F, F' két körmentes élhalmaz a V, n elemű csúcshalmazon. Tegyük fel, hogy $|F| < |F'|$. Ekkor található olyan $F' - F$-beli e él. hogy $F \cup \{e\}$ is körmentes legyen.

- Legyen G_F az a körmentes gráf (erdő), amely csúcshalmaza V és élhalmaza F.
- Ismert, hogy G_F-nek $n - |F|$ komponense van: $c(G_F) = n - |F|$.
- Hasonlóan legyen $G_{F'}$ az a körmentes gráf (erdő), amely csúcshalmaza V és élhalmaza F'.
- $G_{F'}$-nek $n - |F'|$ komponense van:
$c(G_{F'}) = n - |F'| < n - |F| = c(G_F)$.
- Ez csak úgy képzelhető el, ha egy e F'-beli él G_F két különböző komponensének egy-egy csúcsát köti össze.
- Speciálisan $e \notin F$

A LEMMA és bizonyítása

Lemma

Legyen F, F' két körmentes élhalmaz a V, n elemű csúcshalmazon. Tegyük fel, hogy $|F| < |F'|$. Ekkor található olyan $F' - F$-beli e él. hogy $F \cup \{e\}$ is körmentes legyen.

- Legyen G_F az a körmentes gráf (erdő), amely csúcshalmaza V és élhalmaza F.
- Ismert, hogy G_F-nek $n - |F|$ komponense van: $c(G_F) = n - |F|$.
- Hasonlóan legyen $G_{F'}$ az a körmentes gráf (erdő), amely csúcshalmaza V és élhalmaza F'.
- $G_{F'}$-nek $n - |F'|$ komponense van: $c(G_{F'}) = n - |F'| < n - |F| = c(G_F)$.
- Ez csak úgy képzelhető el, ha egy e F'-beli él G_F két különböző komponensének egy-egy csúcsát köti össze.
- Speciálisan $e \notin F$ és $F \cup \{e\}$ is körmentes.

A LEMMÁból $(i)_s \Rightarrow (ii)_s$

A LEMMÁból $(i)_s \Rightarrow (ii)_s$

- Feltehetjük, hogy $|T| = s + 1$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$, ahol az indexelés sorrendje költség szerinti novekvő sorrend.

A LEMMÁból $(i)_s \Rightarrow (ii)_s$

- Feltehetjük, hogy $|T| = s + 1$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$, ahol az indexelés sorrendje költség szerinti novekvő sorrend.
- Legyen $O = \{o_1, o_2, \ldots, o_s\}$.

A LEMMÁból $(i)_s \Rightarrow (ii)_s$

- Feltehetjük, hogy $|T| = s + 1$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$, ahol az indexelés sorrendje költség szerinti novekvő sorrend.
- Legyen $O = \{o_1, o_2, \ldots, o_s\}$. Ekkor $|O| = s < s + 1 = |T|$.

A LEMMÁból $(i)_s \Rightarrow (ii)_s$

- Feltehetjük, hogy $|T| = s + 1$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$, ahol az indexelés sorrendje költség szerinti novekvő sorrend.
- Legyen $O = \{o_1, o_2, \ldots, o_s\}$. Ekkor $|O| = s < s + 1 = |T|$.
- A LEMMA alapján, van olyan t_i, hogy $t_i \notin O$ és $O \cup \{t_i\}$ körmentes.

A LEMMÁból $(i)_s \Rightarrow (ii)_s$

- Feltehetjük, hogy $|T| = s + 1$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$, ahol az indexelés sorrendje költség szerinti novekvő sorrend.
- Legyen $O = \{o_1, o_2, \ldots, o_s\}$. Ekkor $|O| = s < s + 1 = |T|$.
- A LEMMA alapján, van olyan t_i, hogy $t_i \notin O$ és $O \cup \{t_i\}$ körmentes.
- A Kruskal-algoritmus minden élt megvizsgál, t_i-t is.

A LEMMÁból $(i)_s \Rightarrow (ii)_s$

- Feltehetjük, hogy $|T| = s + 1$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$, ahol az indexelés sorrendje költség szerinti novekvő sorrend.
- Legyen $O = \{o_1, o_2, \ldots, o_s\}$. Ekkor $|O| = s < s + 1 = |T|$.
- A LEMMA alapján, van olyan t_i, hogy $t_i \notin O$ és $O \cup \{t_i\}$ körmentes.
- A Kruskal-algoritmus minden élt megvizsgál, t_i-t is.
- O mellé vagy O egy részhalmaza mellé t_i-t beválasztja az algoritmus.

A LEMMÁból $(i)_s \Rightarrow (ii)_s$

- Feltehetjük, hogy $|T| = s + 1$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$, ahol az indexelés sorrendje költség szerinti novekvő sorrend.
- Legyen $O = \{o_1, o_2, \ldots, o_s\}$. Ekkor $|O| = s < s + 1 = |T|$.
- A LEMMA alapján, van olyan t_i, hogy $t_i \notin O$ és $O \cup \{t_i\}$ körmentes.
- A Kruskal-algoritmus minden élt megvizsgál, t_i-t is.
- O mellé vagy O egy részhalmaza mellé t_i-t beválasztja az algoritmus. Készen vagyunk.

A LEMMÁból $(i)_s \Rightarrow (ii)_s$

- Feltehetjük, hogy $|T| = s + 1$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$, ahol az indexelés sorrendje költség szerinti novekvő sorrend.
- Legyen $O = \{o_1, o_2, \ldots, o_s\}$. Ekkor $|O| = s < s + 1 = |T|$.
- A LEMMA alapján, van olyan t_i, hogy $t_i \notin O$ és $O \cup \{t_i\}$ körmentes.
- A Kruskal-algoritmus minden élt megvizsgál, t_i-t is.
- O mellé vagy O egy részhalmaza mellé t_i-t beválasztja az algoritmus. Készen vagyunk.
- Ha t_i vizsgálatánál nem O egy részhalmaza az aktuális kiválasztott élhalmaz, akkor is készen vagyunk.

A LEMMÁból $(i)_s, (ii)_s \Rightarrow (i)_{s+1}$

A LEMMÁból $(i)_s, (ii)_s \Rightarrow (i)_{s+1}$

- Legyen $O = \{o_1, o_2, \ldots, o_{s+1}\}$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$.

A LEMMÁból $(i)_s, (ii)_s \Rightarrow (i)_{s+1}$

- Legyen $O = \{o_1, o_2, \ldots, o_{s+1}\}$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$.
- $O^- = \{o_1, o_2, \ldots, o_s\}$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$ esetét már tárgyaltuk.

A LEMMÁból $(i)_s, (ii)_s \Rightarrow (i)_{s+1}$

- Legyen $O = \{o_1, o_2, \ldots, o_{s+1}\}$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$.

- $O^- = \{o_1, o_2, \ldots, o_s\}$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$ esetét már tárgyaltuk. Gondoljuk végig az előző gondolatmenetet.

A LEMMÁból $(i)_s, (ii)_s \Rightarrow (i)_{s+1}$

- Legyen $O = \{o_1, o_2, \ldots, o_{s+1}\}$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$.

- $O^- = \{o_1, o_2, \ldots, o_s\}$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$ esetét már tárgyaltuk. Gondoljuk végig az előző gondolatmenetet.

- Láttuk, hogy o_{s+1} létezése szükségszerű és ez az elem legkésőbb t_{s+1} vizsgálatánál kiválasztásra kerül.

A LEMMÁból $(i)_s, (ii)_s \Rightarrow (i)_{s+1}$

- Legyen $O = \{o_1, o_2, \ldots, o_{s+1}\}$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$.

- $O^- = \{o_1, o_2, \ldots, o_s\}$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$ esetét már tárgyaltuk. Gondoljuk végig az előző gondolatmenetet.

- Láttuk, hogy o_{s+1} létezése szükségszerű és ez az elem legkésőbb t_{s+1} vizsgálatánál kiválasztásra kerül.

- A Kruskal-algoritmus vizsgálatait/választásait a költségek sorrendje alapján végzi.

A LEMMÁból $(i)_s, (ii)_s \Rightarrow (i)_{s+1}$

- Legyen $O = \{o_1, o_2, \ldots, o_{s+1}\}$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$.

- $O^- = \{o_1, o_2, \ldots, o_s\}$ és $T = \{t_1, t_2, \ldots, t_{s+1}\}$ esetét már tárgyaltuk. Gondoljuk végig az előző gondolatmenetet.

- Láttuk, hogy o_{s+1} létezése szükségszerű és ez az elem legkésőbb t_{s+1} vizsgálatánál kiválasztásra kerül.

- A Kruskal-algoritmus vizsgálatait/választásait a költségek sorrendje alapján végzi. Tehát

$$c(o_{s+1}) \leq c(t_{s+1}),$$

ahogy bizonyítani kellett.

A Kruskal-algoritmus analízise

A Kruskal-algoritmus analízise

- A rendezési lépés költsége

$$\mathcal{O}(|E| \log |E|),$$

A Kruskal-algoritmus analízise

- A rendezési lépés költsége

$$\mathcal{O}(|E| \log |E|),$$

ahogy majd látni fogjuk.

A Kruskal-algoritmus analízise

- A rendezési lépés költsége

$$\mathcal{O}(|E| \log |E|),$$

ahogy majd látni fogjuk.

- A maradék részben $|E|$ darab körmentesség tesztelést végzünk egy élhalmazon.

A Kruskal-algoritmus analízise

- A rendezési lépés költsége

$$\mathcal{O}(|E| \log |E|),$$

ahogy majd látni fogjuk.

- A maradék részben $|E|$ darab körmentesség tesztelést végzünk egy élhalmazon. Minden esetben az élhalmaz mérete legfeljebb $|V|$.

A Kruskal-algoritmus analízise

- A rendezési lépés költsége

$$\mathcal{O}(|E| \log |E|),$$

ahogy majd látni fogjuk.

- A maradék részben $|E|$ darab körmentesség tesztelést végzünk egy élhalmazon. Minden esetben az élhalmaz mérete legfeljebb $|V|$. Egy teszt költsége $\mathcal{O}(|V|)$.

A Kruskal-algoritmus analízise

- A rendezési lépés költsége

$$\mathcal{O}(|E| \log |E|),$$

ahogy majd látni fogjuk.

- A maradék részben $|E|$ darab körmentesség tesztelést végzünk egy élhalmazon. Minden esetben az élhalmaz mérete legfeljebb $|V|$. Egy teszt költsége $\mathcal{O}(|V|)$.

- A teljes futás költsége

$$\mathcal{O}(|E| \log |E|) + |E| \cdot \mathcal{O}(|V|) = \mathcal{O}(|E| \cdot |V|).$$

A Kruskal-algoritmus analízise

- A rendezési lépés költsége

$$\mathcal{O}(|E| \log |E|),$$

ahogy majd látni fogjuk.

- A maradék részben $|E|$ darab körmentesség tesztelést végzünk egy élhalmazon. Minden esetben az élhalmaz mérete legfeljebb $|V|$. Egy teszt költsége $\mathcal{O}(|V|)$.

- A teljes futás költsége

$$\mathcal{O}(|E| \log |E|) + |E| \cdot \mathcal{O}(|V|) = \mathcal{O}(|E| \cdot |V|).$$

- Egy naív implementáció analízisét végeztük el. Ügyesebb megoldások is vannak.

Szünet

Az alapkérdés

Az alapkérdés

A kérdés

Az alapkérdés

A kérdés

Adott

Az alapkérdés

A kérdés

Adott

(i) $\overrightarrow{G}$ irányított gráf,

Az alapkérdés

A kérdés

Adott

(i) $\overrightarrow{G}$ irányított gráf,

(ii) $\ell : E(\overrightarrow{G}) \to \mathbb{R}_{++}$ hosszfüggvény az éleken,

Az alapkérdés

A kérdés

Adott

(i) $\overrightarrow{G}$ irányított gráf,

(ii) $\ell : E(\overrightarrow{G}) \to \mathbb{R}_{++}$ hosszfüggvény az éleken,

(iii) s, t két kitüntetett csúcs.

Az alapkérdés

A kérdés

Adott

(i) $\overrightarrow{G}$ irányított gráf,

(ii) $\ell : E(\overrightarrow{G}) \to \mathbb{R}_{++}$ hosszfüggvény az éleken,

(iii) s, t két kitüntetett csúcs.

Határozzuk meg s és t távolságát.

Emlékeztető

Emlékeztető

Definíció: Irányított gráf

G irányított gráf:

Emlékeztető

Definíció: Irányított gráf

G irányított gráf: $V = V(\overrightarrow{G})$ véges csúcshalmaz,

Emlékeztető

Definíció: Irányított gráf

G irányított gráf: $V = V(\overrightarrow{G})$ véges csúcshalmaz, $E = E(\overrightarrow{G})$ véges élhalmaz,

Emlékeztető

Definíció: Irányított gráf

G irányított gráf: $V = V(\overrightarrow{G})$ véges csúcshalmaz, $E = E(\overrightarrow{G})$ véges élhalmaz, két illeszkedési reláció: kifut, befut, amelykre minden élhez pontosan egy végpont illeszkedik.

Emlékeztető

Definíció: Irányított gráf

G irányított gráf: $V = V(\overrightarrow{G})$ véges csúcshalmaz, $E = E(\overrightarrow{G})$ véges élhalmaz, két illeszkedési reláció: kifut, befut, amelykre minden élhez pontosan egy végpont illeszkedik.

$vK\overrightarrow{e}$ jelentése: a v csúcsból kifut az $\overrightarrow{e}$ él. $vB\overrightarrow{e}$ jelentése: a v csúcsba befut az $\overrightarrow{e}$ él.

Emlékeztető (folytatás)

Emlékeztető (folytatás)

Definíció: Irányított séta irányított gráfban

$\overrightarrow{uv}$-séta $\overrightarrow{G}$-ben:

$$\overrightarrow{S} : u = w_0, \overrightarrow{e}_1, w_1, \overrightarrow{e}_2, \ldots, w_{L-1}, \overrightarrow{e}_L, w_L = v,$$

Emlékeztető (folytatás)

Definíció: Irányított séta irányított gráfban

$\overrightarrow{uv}$-séta $\overrightarrow{G}$-ben:

$$\overrightarrow{S} : u = w_0, \overrightarrow{e}_1, w_1, \overrightarrow{e}_2, \ldots, w_{L-1}, \overrightarrow{e}_L, w_L = v,$$

ahol $w_i \in V$ $(i = 0, 1, \ldots, L)$,

Emlékeztető (folytatás)

Definíció: Irányított séta irányított gráfban

$\overrightarrow{uv}$-séta $\overrightarrow{G}$-ben:

$$\overrightarrow{S} : u = w_0, \overrightarrow{e}_1, w_1, \overrightarrow{e}_2, \ldots, w_{L-1}, \overrightarrow{e}_L, w_L = v,$$

ahol $w_i \in V$ $(i = 0, 1, \ldots, L)$, $\overrightarrow{e}_i \in E$ $(i = 1, \ldots, L)$,

Emlékeztető (folytatás)

Definíció: Irányított séta irányított gráfban

$\overrightarrow{uv}$-séta $\overrightarrow{G}$-ben:

$$\overrightarrow{S} : u = w_0, \overrightarrow{e}_1, w_1, \overrightarrow{e}_2, \ldots, w_{L-1}, \overrightarrow{e}_L, w_L = v,$$

ahol $w_i \in V$ $(i = 0, 1, \ldots, L)$, $\overrightarrow{e}_i \in E$ $(i = 1, \ldots, L)$, e_i kifut w_{i-1}-ből és befut w_i-be $(i = 1, \ldots, L)$.

Emlékeztető (folytatás)

Definíció: Irányított séta irányított gráfban

$\overrightarrow{uv}$-séta $\overrightarrow{G}$-ben:

$$\overrightarrow{\mathcal{S}} : u = w_0, \overrightarrow{e}_1, w_1, \overrightarrow{e}_2, \ldots, w_{L-1}, \overrightarrow{e}_L, w_L = v,$$

ahol $w_i \in V$ $(i = 0, 1, \ldots, L)$, $\overrightarrow{e}_i \in E$ $(i = 1, \ldots, L)$, e_i kifut w_{i-1}-ből és befut w_i-be $(i = 1, \ldots, L)$.

Az $\overrightarrow{\mathcal{S}}$ $\overrightarrow{uv}$-séta gráfelméleti hossza L.

Emlékeztető (folytatás)

Emlékeztető (folytatás)

Séta hossza élsúlyozott gráfban

Az $\overrightarrow{S}$ $\overrightarrow{uv}$-séta súlyozott hossza

$$\sum_{i=1}^{L} \ell(\overrightarrow{e}_i).$$

Emlékeztető (folytatás)

Séta hossza élsúlyozott gráfban

Az $\overrightarrow{S}$ $\overrightarrow{uv}$-séta súlyozott hossza

$$\sum_{i=1}^{L} \ell(\overrightarrow{e}_i).$$

Távolság élsúlyozott gráfban

$d(u, v)$ az u és v csúcsok távolsága, a legrövidebb uv séta hossza.

Emlékeztető (folytatás)

Séta hossza élsúlyozott gráfban

Az $\overrightarrow{\mathcal{S}}$ $\overrightarrow{uv}$-séta súlyozott hossza

$$\sum_{i=1}^{L} \ell(\overrightarrow{e}_i).$$

Távolság élsúlyozott gráfban

$d(u, v)$ az u és v csúcsok távolsága, a legrövidebb uv séta hossza.

Észrevétel

A legrövidebb uv séta egy út lesz.

Kezdeti megjegyzések

Kezdeti megjegyzések

- Feltesszük, hogy nincs hurokél.

Kezdeti megjegyzések

- Feltesszük, hogy nincs hurokél. Feltesszük, hogy u-ból v-be legfeljebb egy él vezet.

Kezdeti megjegyzések

- Feltesszük, hogy nincs hurokél. Feltesszük, hogy u-ból v-be legfeljebb egy él vezet. Azaz feltesszük, hogy $\overrightarrow{G}$ egyszerű.

Kezdeti megjegyzések

- Feltesszük, hogy nincs hurokél. Feltesszük, hogy u-ból v-be legfeljebb egy él vezet. Azaz feltesszük, hogy $\overrightarrow{G}$ egyszerű.
- Feltesszük, hogy minden csúcs elérhető s-ből irányított sétával/úttal.

Gráfelméleti távolság esete: súlyozatlan eset

Mielőtt a legrövidebb út problémáját tárgyaljuk, megnézzük a gráfelméleti távolságra vonatkozó problémát.

Gráfelméleti távolság esete: súlyozatlan eset

Mielőtt a legrövidebb út problémáját tárgyaljuk, megnézzük a gráfelméleti távolságra vonatkozó problémát. Azaz mi történik, ha egy séta/út hossza az élek/lépések megszámolásából adódik.

Gráfelméleti távolság esete: súlyozatlan eset

Mielőtt a legrövidebb út problémáját tárgyaljuk, megnézzük a gráfelméleti távolságra vonatkozó problémát. Azaz mi történik, ha egy séta/út hossza az élek/lépések megszámolásából adódik.

- Egy erősebb feladatot oldunk meg.

Gráfelméleti távolság esete: súlyozatlan eset

Mielőtt a legrövidebb út problémáját tárgyaljuk, megnézzük a gráfelméleti távolságra vonatkozó problémát. Azaz mi történik, ha egy séta/út hossza az élek/lépések megszámolásából adódik.

- Egy erősebb feladatot oldunk meg. $(\overrightarrow{G}, s)$ lesz az input.

Gráfelméleti távolság esete: súlyozatlan eset

Mielőtt a legrövidebb út problémáját tárgyaljuk, megnézzük a gráfelméleti távolságra vonatkozó problémát. Azaz mi történik, ha egy séta/út hossza az élek/lépések megszámolásából adódik.

• Egy erősebb feladatot oldunk meg. $(\overrightarrow{G}, s)$ lesz az input. Meghatározzuk azon pontok S halmazát, amelyekbe vezet s-ből induló irányított út.

Gráfelméleti távolság esete: súlyozatlan eset

Mielőtt a legrövidebb út problémáját tárgyaljuk, megnézzük a gráfelméleti távolságra vonatkozó problémát. Azaz mi történik, ha egy séta/út hossza az élek/lépések megszámolásából adódik.

- Egy erősebb feladatot oldunk meg. $(\overrightarrow{G}, s)$ lesz az input.

Meghatározzuk azon pontok S halmazát, amelyekbe vezet s-ből induló irányított út.

- Ezek halmaza $S = S_0 \dot{\cup} S_1 \dot{\cup} \ldots \dot{\cup} S_L$ lesz,

Gráfelméleti távolság esete: súlyozatlan eset

Mielőtt a legrövidebb út problémáját tárgyaljuk, megnézzük a gráfelméleti távolságra vonatkozó problémát. Azaz mi történik, ha egy séta/út hossza az élek/lépések megszámolásából adódik.

- Egy erősebb feladatot oldunk meg. $(\overrightarrow{G}, s)$ lesz az input.

Meghatározzuk azon pontok S halmazát, amelyekbe vezet s-ből induló irányított út.

- Ezek halmaza $S = S_0 \dot{\cup} S_1 \dot{\cup} \ldots \dot{\cup} S_L$ lesz, ahol S_i pontosan azokat a pontokat tartalmazza, amelyek gráfelméleti távolsága s-től i.

Gráfelméleti távolság esete: súlyozatlan eset

Mielőtt a legrövidebb út problémáját tárgyaljuk, megnézzük a gráfelméleti távolságra vonatkozó problémát. Azaz mi történik, ha egy séta/út hossza az élek/lépések megszámolásából adódik.

- Egy erősebb feladatot oldunk meg. $(\overrightarrow{G}, s)$ lesz az input. Meghatározzuk azon pontok S halmazát, amelyekbe vezet s-ből induló irányított út.
- Ezek halmaza $S = S_0 \dot{\cup} S_1 \dot{\cup} \ldots \dot{\cup} S_L$ lesz, ahol S_i pontosan azokat a pontokat tartalmazza, amelyek gráfelméleti távolsága s-től i.
- L jelöli az s-ből induló leghosszabb út hosszát.

Gráfelméleti távolság esete: súlyozatlan eset

Mielőtt a legrövidebb út problémáját tárgyaljuk, megnézzük a gráfelméleti távolságra vonatkozó problémát. Azaz mi történik, ha egy séta/út hossza az élek/lépések megszámolásából adódik.

- Egy erősebb feladatot oldunk meg. $(\overrightarrow{G}, s)$ lesz az input. Meghatározzuk azon pontok S halmazát, amelyekbe vezet s-ből induló irányított út.
- Ezek halmaza $S = S_0 \dot{\cup} S_1 \dot{\cup} \ldots \dot{\cup} S_L$ lesz, ahol S_i pontosan azokat a pontokat tartalmazza, amelyek gráfelméleti távolsága s-től i.
- L jelöli az s-ből induló leghosszabb út hosszát.
- S és $\overline{S} = V(G) - S$ között az összes él $\overline{S}$-ből indul és S-be vezet.

Gráfelméleti távolság esete: súlyozatlan eset

Mielőtt a legrövidebb út problémáját tárgyaljuk, megnézzük a gráfelméleti távolságra vonatkozó problémát. Azaz mi történik, ha egy séta/út hossza az élek/lépések megszámolásából adódik.

• Egy erősebb feladatot oldunk meg. $(\overrightarrow{G}, s)$ lesz az input. Meghatározzuk azon pontok S halmazát, amelyekbe vezet s-ből induló irányított út.

• Ezek halmaza $S = S_0 \dot{\cup} S_1 \dot{\cup} \ldots \dot{\cup} S_L$ lesz, ahol S_i pontosan azokat a pontokat tartalmazza, amelyek gráfelméleti távolsága s-től i.

• L jelöli az s-ből induló leghosszabb út hosszát.

• S és $\overline{S} = V(G) - S$ között az összes él $\overline{S}$-ből indul és S-be vezet. Ez bizonyítja, hogy s-ből irányított út nem léphet ki S-ből.

Szélességi keresés

Szélességi keresés

Szélességi keresés

Szélességi keresés

(I) // Inicializálás // Legyen $S_0 = \{s\}$.

Szélességi keresés

Szélességi keresés

(I) // Inicializálás // Legyen $S_0 = \{s\}$.

Szélességi keresés

Szélességi keresés

(I) // Inicializálás // Legyen $S_0 = \{s\}$.

(C) // Ciklus // Amíg $S_i \neq \emptyset$

$$S_{i+1} = \{x \in V(G) - (S_0 \cup \ldots \cup S_i) : \text{ van olyan } \sigma \in S_i, \text{ hogy } \overrightarrow{\sigma x} \in E(G)\}$$

és $i \leftarrow i + 1$.

Szélességi keresés

Szélességi keresés

(I) // Inicializálás // Legyen $S_0 = \{s\}$.

(C) // Ciklus // Amíg $S_i \neq \emptyset$

$$S_{i+1} = \{x \in V(G) - (S_0 \cup \ldots \cup S_i) : \text{ van olyan } \sigma \in S_i, \text{ hogy } \overrightarrow{\sigma x} \in E(G)\}$$

és $i \leftarrow i + 1$.

Tétel

A fenti algoritmus outputja korrekt.

Szélességi keresés

Szélességi keresés

(I) // Inicializálás // Legyen $S_0 = \{s\}$.

(C) // Ciklus // Amíg $S_i \neq \emptyset$

$$S_{i+1} = \{x \in V(G) - (S_0 \cup \ldots \cup S_i) : \text{ van olyan } \sigma \in S_i, \text{ hogy } \overrightarrow{\sigma x} \in E(G)\}$$

és $i \leftarrow i + 1$.

Tétel

A fenti algoritmus outputja korrekt.

Szélességi keresés

Szélességi keresés

(I) // Inicializálás // Legyen $S_0 = \{s\}$.

(C) // Ciklus // Amíg $S_i \neq \emptyset$

$$S_{i+1} = \{x \in V(G) - (S_0 \cup \ldots \cup S_i) : \text{ van olyan } \sigma \in S_i, \text{ hogy } \overrightarrow{\sigma x} \in E(G)\}$$

és $i \leftarrow i + 1$.

Tétel

A fenti algoritmus outputja korrekt.

(i) Ha $x \in S_i$, akkor s és x gráfelméleti távolsága i.

Szélességi keresés

Szélességi keresés

(I) // Inicializálás // Legyen $S_0 = \{s\}$.

(C) // Ciklus // Amíg $S_i \neq \emptyset$

$$S_{i+1} = \{x \in V(G) - (S_0 \cup \ldots \cup S_i) : \text{ van olyan } \sigma \in S_i, \text{ hogy } \overrightarrow{\sigma x} \in E(G)\}$$

és $i \leftarrow i + 1$.

Tétel

A fenti algoritmus outputja korrekt.

(i) Ha $x \in S_i$, akkor s és x gráfelméleti távolsága i.

(ii) Ha $x \notin S$, akkor nem létezik irányított út s-ből x-be.

Szélességi kereső fa

Szélességi kereső fa

A fenti algoritmusnak vegyük a következő változatát:

Szélességi kereső fa

A fenti algoritmusnak vegyük a következő változatát:

• Ha a (C) ciklusban egy x csúcs a S_{i+1} halmazba sorolódik, akkor az algoritmusnak találnia kell egy $\sigma \in S_i$ csúcsot és egy $\overrightarrow{\sigma x}$ élt. A „bizonyító" éleket egy F élhalmazba gyűjtsük össze.

Szélességi kereső fa

A fenti algoritmusnak vegyük a következő változatát:

• Ha a (C) ciklusban egy x csúcs a S_{i+1} halmazba sorolódik, akkor az algoritmusnak találnia kell egy $\sigma \in S_i$ csúcsot és egy $\overrightarrow{\sigma x}$ élt. A „bizonyító" éleket egy F élhalmazba gyűjtsük össze.

Tétel

A $G|_S$ gráfban F egy $\mathcal{F}$ feszítőfa élhalmaza. $\mathcal{F}$ az s csúccsal egy gyökeres fa, amelyben minden él s-től „elfele" vezet.

Szélességi kereső fa

A fenti algoritmusnak vegyük a következő változatát:

• Ha a (C) ciklusban egy x csúcs a S_{i+1} halmazba sorolódik, akkor az algoritmusnak találnia kell egy $\sigma \in S_i$ csúcsot és egy $\overrightarrow{\sigma x}$ élt. A „bizonyító" éleket egy F élhalmazba gyűjtsük össze.

Tétel

A $G|_S$ gráfban F egy $\mathcal{F}$ feszítőfa élhalmaza. $\mathcal{F}$ az s csúccsal egy gyökeres fa, amelyben minden él s-től „elfele" vezet.

Minden $x \in S$ csúcsra pontosan egy $\mathcal{F}$-beli $\overrightarrow{sx}$-út van. Ez az út egy legrövidebb (gráfelméleti értelemben) út, amely s-ből x-be vezet.

Súlyozott eset: Az alapötlet

Súlyozott eset: Az alapötlet

- Egy általánosabb problémát vizsgálunk.

Súlyozott eset: Az alapötlet

- Egy általánosabb problémát vizsgálunk.
- Legyen $s \in S \subset V(\overrightarrow{G})$ egy csúcshalmaz. Erre az S csúcshalmazra a következő problémát tűzzük ki

Súlyozott eset: Az alapötlet

- Egy általánosabb problémát vizsgálunk.
- Legyen $s \in S \subset V(\overrightarrow{G})$ egy csúcshalmaz. Erre az S csúcshalmazra a következő problémát tűzzük ki
  (i) Ha $v \in S$, akkor határozzuk meg a legrövidebb $\overrightarrow{sv}$ utat S-en belül.

Súlyozott eset: Az alapötlet

- Egy általánosabb problémát vizsgálunk.
- Legyen $s \in S \subset V(\overrightarrow{G})$ egy csúcshalmaz. Erre az S csúcshalmazra a következő problémát tűzzük ki

(i) Ha $v \in S$, akkor határozzuk meg a legrövidebb $\overrightarrow{sv}$ utat S-en belül.

(ii) Ha $v \notin S$, akkor határozzuk meg a legrövidebb $\overrightarrow{sv}$ utat, amely S-en belül halad, kivéve az utolsó lépését (ami persze v-be, S-en kívülre lép).

Súlyozott eset: Az alapötlet

- Egy általánosabb problémát vizsgálunk.
- Legyen $s \in S \subset V(\overrightarrow{G})$ egy csúcshalmaz. Erre az S csúcshalmazra a következő problémát tűzzük ki
  - (i) Ha $v \in S$, akkor határozzuk meg a legrövidebb $\overrightarrow{sv}$ utat S-en belül.
  - (ii) Ha $v \notin S$, akkor határozzuk meg a legrövidebb $\overrightarrow{sv}$ utat, amely S-en belül halad, kivéve az utolsó lépését (ami persze v-be, S-en kívülre lép).
- A megoldást úgy képzelhetjük, hogy minden csúcshoz egy „címkét” gondolunk, ami a kérdezett információt tartalmazza.

Súlyozott eset: Az alapötlet (folytatás)

Súlyozott eset: Az alapötlet (folytatás)

- A kiinduló feladat $S = \{s\}$ lesz, ami nagyon egyszerű. A megoldásra mint egy „kiinduló címkézés” gondolunk.

Súlyozott eset: Az alapötlet (folytatás)

- A kiinduló feladat $S = \{s\}$ lesz, ami nagyon egyszerű. A megoldásra mint egy „kiinduló címkézés” gondolunk.
- A megoldás lényege, hogyan mindig növelni fogjuk az aktuális S halmazt (mohóság).

Súlyozott eset: Az alapötlet (folytatás)

- A kiinduló feladat $S = \{s\}$ lesz, ami nagyon egyszerű. A megoldásra mint egy „kiinduló címkézés” gondolunk.
- A megoldás lényege, hogyan mindig növelni fogjuk az aktuális S halmazt (mohóság). Az új feladat megoldására mint „a címkézés update-eléseként” hivatkozunk.

Súlyozott eset: Az alapötlet (folytatás)

- A kiinduló feladat $S = \{s\}$ lesz, ami nagyon egyszerű. A megoldásra mint egy „kiinduló címkézés” gondolunk.
- A megoldás lényege, hogyan mindig növelni fogjuk az aktuális S halmazt (mohóság). Az új feladat megoldására mint „a címkézés update-eléseként” hivatkozunk.
- A növekedés során egy S-en kívüli pontot kell választani. Itt kell „ügyesnek” lenni.

Súlyozott eset: Az alapötlet (folytatás)

- A kiinduló feladat $S = \{s\}$ lesz, ami nagyon egyszerű. A megoldásra mint egy „kiinduló címkézés" gondolunk.
- A megoldás lényege, hogyan mindig növelni fogjuk az aktuális S halmazt (mohóság). Az új feladat megoldására mint „a címkézés update-eléseként" hivatkozunk.
- A növekedés során egy S-en kívüli pontot kell választani. Itt kell „ügyesnek" lenni.
- Az $S = V(\overrightarrow{G})$ elérésekor megoldottuk a legrövidebb út problémáját.

Dijkstra-algoritmus

Dijkstra-algoritmus

Dijkstra-algoritmus

Dijkstra-algoritmus

Dijkstra-algoritmus

(I) // Inicializálás

Dijkstra-algoritmus

Dijkstra-algoritmus

(I) // Inicializálás
Legyen $S = \{s\}$, $c(s) = 0$.

Dijkstra-algoritmus

Dijkstra-algoritmus

(I) // Inicializálás

Legyen $S = \{s\}$, $c(s) = 0$.

$v \notin S$, $\overrightarrow{sv} \in E$ esetén $c(v) = \ell(\overrightarrow{sv})$.

Dijkstra-algoritmus

Dijkstra-algoritmus

(I) // Inicializálás

Legyen $S = \{s\}$, $c(s) = 0$.

$v \notin S$, $\overrightarrow{sv} \in E$ esetén $c(v) = \ell(\overrightarrow{sv})$.

$v \notin S$, $\overrightarrow{sv} \notin E$ esetén $c(v) = \infty$.

Dijkstra-algoritmus

Dijkstra-algoritmus

(I) // Inicializálás

Legyen $S = \{s\}$, $c(s) = 0$.

$v \notin S$, $\overrightarrow{sv} \in E$ esetén $c(v) = \ell(\overrightarrow{sv})$.

$v \notin S$, $\overrightarrow{sv} \notin E$ esetén $c(v) = \infty$.

(B) // Bővítés

Dijkstra-algoritmus

Dijkstra-algoritmus

(I) // Inicializálás
Legyen $S = \{s\}$, $c(s) = 0$.
$v \notin S$, $\overrightarrow{sv} \in E$ esetén $c(v) = \ell(\overrightarrow{sv})$.
$v \notin S$, $\overrightarrow{sv} \notin E$ esetén $c(v) = \infty$.

(B) // Bővítés
Legyen $m \notin S$ azon (egyik) csúcs, amely címkéje a legkisebb az S halmazon kívül.

Dijkstra-algoritmus

Dijkstra-algoritmus

(I) // Inicializálás

Legyen $S = \{s\}$, $c(s) = 0$.

$v \notin S$, $\overrightarrow{sv} \in E$ esetén $c(v) = \ell(\overrightarrow{sv})$.

$v \notin S$, $\overrightarrow{sv} \notin E$ esetén $c(v) = \infty$.

(B) // Bővítés

Legyen $m \notin S$ azon (egyik) csúcs, amely címkéje a legkisebb az S halmazon kívül.

$S \leftarrow S \cup \{m\}$.

Dijkstra-algoritmus

Dijkstra-algoritmus

(I) // Inicializálás
Legyen $S = \{s\}$, $c(s) = 0$.
$v \notin S$, $\overrightarrow{sv} \in E$ esetén $c(v) = \ell(\overrightarrow{sv})$.
$v \notin S$, $\overrightarrow{sv} \notin E$ esetén $c(v) = \infty$.

(B) // Bővítés
Legyen $m \notin S$ azon (egyik) csúcs, amely címkéje a legkisebb az S halmazon kívül.
$S \leftarrow S \cup \{m\}$.

(U) // Update

Dijkstra-algoritmus

Dijkstra-algoritmus

(I) // Inicializálás
Legyen $S = \{s\}$, $c(s) = 0$.
$v \notin S$, $\overrightarrow{sv} \in E$ esetén $c(v) = \ell(\overrightarrow{sv})$.
$v \notin S$, $\overrightarrow{sv} \notin E$ esetén $c(v) = \infty$.

(B) // Bővítés
Legyen $m \notin S$ azon (egyik) csúcs, amely címkéje a legkisebb az S halmazon kívül.
$S \leftarrow S \cup \{m\}$.

(U) // Update
Csak azon $n \notin S$ csúcsok címkéje változhat, amelyekre $\overrightarrow{mn} \in E$:

$$c_{\text{új}}(n) = \min\{c_{\text{régi}}(n), c(m) + \ell(\overrightarrow{mn})\}.$$

Dijkstra-algoritmus

Dijkstra-algoritmus

(I) // Inicializálás
Legyen $S = \{s\}$, $c(s) = 0$.
$v \notin S$, $\overrightarrow{sv} \in E$ esetén $c(v) = \ell(\overrightarrow{sv})$.
$v \notin S$, $\overrightarrow{sv} \notin E$ esetén $c(v) = \infty$.

(B) // Bővítés
Legyen $m \notin S$ azon (egyik) csúcs, amely címkéje a legkisebb az S halmazon kívül.
$S \leftarrow S \cup \{m\}$.

(U) // Update
Csak azon $n \notin S$ csúcsok címkéje változhat, amelyekre $\overrightarrow{mn} \in E$:

$$c_{\text{új}}(n) = \min\{c_{\text{régi}}(n), c(m) + \ell(\overrightarrow{mn})\}.$$

Vissza a (B) lépéshez.

Az algoritmus helyessége

Az algoritmus helyessége

Tétel

A Dikstra-algoritmusában az eredeti és minden update-elt címkézés

Az algoritmus helyessége

Tétel

A Dikstra-algoritmusában az eredeti és minden update-elt címkézés

(i) S-beli értékei megadják az odavezető legrövidebb út hosszát (amely optimum S-en belül is megvalósítható)

Az algoritmus helyessége

Tétel

A Dikstra-algoritmusában az eredeti és minden update-elt címkézés

(i) S-beli értékei megadják az odavezető legrövidebb út hosszát (amely optimum S-en belül is megvalósítható)

(ii) S-en kívüli értékei megadják az odavezető legrövidebb út hosszát azon utak között, amelyek az utolsó csúcs kivételével S-en belül haladnak.

Az algoritmus helyessége

Tétel

A Dikstra-algoritmusában az eredeti és minden update-elt címkézés

(i) S-beli értékei megadják az odavezető legrövidebb út hosszát (amely optimum S-en belül is megvalósítható)

(ii) S-en kívüli értékei megadják az odavezető legrövidebb út hosszát azon utak között, amelyek az utolsó csúcs kivételével S-en belül haladnak.

- $|S|$-re vonatkozó teljes indukcióval bizonyítunk.

Az algoritmus helyessége

Tétel

A Dikstra-algoritmusában az eredeti és minden update-elt címkézés

(i) S-beli értékei megadják az odavezető legrövidebb út hosszát (amely optimum S-en belül is megvalósítható)

(ii) S-en kívüli értékei megadják az odavezető legrövidebb út hosszát azon utak között, amelyek az utolsó csúcs kivételével S-en belül haladnak.

• $|S|$-re vonatkozó teljes indukcióval bizonyítunk. $|S| = 1$ eset nyilvánvaló.

Az algoritmus helyessége

Tétel

A Dikstra-algoritmusában az eredeti és minden update-elt címkézés

(i) S-beli értékei megadják az odavezető legrövidebb út hosszát (amely optimum S-en belül is megvalósítható)

(ii) S-en kívüli értékei megadják az odavezető legrövidebb út hosszát azon utak között, amelyek az utolsó csúcs kivételével S-en belül haladnak.

- $|S|$-re vonatkozó teljes indukcióval bizonyítunk. $|S| = 1$ eset nyilvánvaló.
- Egy update után csak a következő két állítást kell belátni:

Az algoritmus helyessége

Tétel

A Dikstra-algoritmusában az eredeti és minden update-elt címkézés

(i) S-beli értékei megadják az odavezető legrövidebb út hosszát (amely optimum S-en belül is megvalósítható)

(ii) S-en kívüli értékei megadják az odavezető legrövidebb út hosszát azon utak között, amelyek az utolsó csúcs kivételével S-en belül haladnak.

- $|S|$-re vonatkozó teljes indukcióval bizonyítunk. $|S| = 1$ eset nyilvánvaló.
- Egy update után csak a következő két állítást kell belátni:

(i) m címkéje korrekt,

Az algoritmus helyessége

Tétel

A Dikstra-algoritmusában az eredeti és minden update-elt címkézés

(i) S-beli értékei megadják az odavezető legrövidebb út hosszát (amely optimum S-en belül is megvalósítható)

(ii) S-en kívüli értékei megadják az odavezető legrövidebb út hosszát azon utak között, amelyek az utolsó csúcs kivételével S-en belül haladnak.

- $|S|$-re vonatkozó teljes indukcióval bizonyítunk. $|S| = 1$ eset nyilvánvaló.
- Egy update után csak a következő két állítást kell belátni:

(i) m címkéje korrekt,

(ii) $x \notin S$ címkéje korrekt.

Bizonyítás

Bizonyítás

Minden s-ből induló $s \in X$-en kívülre vezető utat „szétszedhetünk” három részre az alábbi módon:

Bizonyítás

Minden s-ből induló $s \in X$-en kívülre vezető utat „szétszedhetünk” három részre az alábbi módon:

X-ben haladó kezdeti rész „+” kilépés X-ből „+” további rész.

Bizonyítás: m címkéje korrekt

Bizonyítás: m címkéje korrekt

$m \notin S_{\text{régi}}$. Így tetszőleges $\overrightarrow{sm}$-utat „szétszedhetjük” $S_{\text{régi}}$-re alapulva a fenti módon: $S_{\text{régi}}$-ben haladó rész „+” kilépés $S_{\text{régi}}$-ből „+” utolsó rész.

Bizonyítás: m címkéje korrekt

$m \notin S_{\text{régi}}$. Így tetszőleges $\overrightarrow{sm}$-utat „szétszedhetjük” $S_{\text{régi}}$-re alapulva a fenti módon: $S_{\text{régi}}$-ben haladó rész „$+$” kilépés $S_{\text{régi}}$-ből „$+$” utolsó rész.

1. eset: Amikor az utolsó rész nemüres, akkor a megfelelő út hossza nagyobb mint m címkéje (ℓ pozitív értékű és m-et speciális módon választottuk).

Bizonyítás: m címkéje korrekt

$m \notin S_{\text{régi}}$. Így tetszőleges $\overrightarrow{sm}$-utat „szétszedhetjük” $S_{\text{régi}}$-re alapulva a fenti módon: $S_{\text{régi}}$-ben haladó rész „$+$” kilépés $S_{\text{régi}}$-ből „$+$” utolsó rész.

1. eset: Amikor az utolsó rész nemüres, akkor a megfelelő út hossza nagyobb mint m címkéje (ℓ pozitív értékű és m-et speciális módon választottuk).

2. eset: Amikor az utolsó rész üres, akkor a megfelelő címke az indukciós feltevés miatt korrekt.

Bizonyítás: $x \notin S$ címkéje korrekt

Tetszőleges $\overrightarrow{sx}$-utat „szétszedhetjük” S-re alapulva a fenti módon: S-ben haladó rész „+” kilépés S-ből „+” utolsó rész.

Bizonyítás: $x \notin S$ címkéje korrekt

Tetszőleges $\overrightarrow{sx}$-utat „szétszedhetjük” S-re alapulva a fenti módon: S-ben haladó rész „+” kilépés S-ből „+” utolsó rész. Csak azok az utak érdekelnek, amely utolsó része üres. Ismét két esetet vizsgálunk.

Bizonyítás: $x \notin S$ címkéje korrekt

Tetszőleges $\overrightarrow{sx}$-utat „szétszedhetjük” S-re alapulva a fenti módon: S-ben haladó rész „+” kilépés S-ből „+” utolsó rész. Csak azok az utak érdekelnek, amely utolsó része üres. Ismét két esetet vizsgálunk.

1. eset: A kilépés nem m-ből történik. Ekkor $c_{\text{régi}}(x)$ az indukciós lépés alapján becsli a hosszat.

Bizonyítás: $x \notin S$ címkéje korrekt

Tetszőleges $\overrightarrow{sx}$-utat „szétszedhetjük” S-re alapulva a fenti módon: S-ben haladó rész „+” kilépés S-ből „+” utolsó rész. Csak azok az utak érdekelnek, amely utolsó része üres. Ismét két esetet vizsgálunk.

1. eset: A kilépés nem m-ből történik. Ekkor $c_{\text{régi}}(x)$ az indukciós lépés alapján becsli a hosszat.

2. eset: A kilépés m-ből történik. Ekkor $c(m) + \ell(\overrightarrow{mx})$ becsli a hosszat.

Bizonyítás: $x \notin S$ címkéje korrekt

Tetszőleges $\overrightarrow{sx}$-utat „szétszedhetjük” S-re alapulva a fenti módon: S-ben haladó rész „+” kilépés S-ből „+” utolsó rész. Csak azok az utak érdekelnek, amely utolsó része üres. Ismét két esetet vizsgálunk.

1. eset: A kilépés nem m-ből történik. Ekkor $c_{\text{régi}}(x)$ az indukciós lépés alapján becsli a hosszat.

2. eset: A kilépés m-ből történik. Ekkor $c(m) + \ell(\overrightarrow{mx})$ becsli a hosszat.

Az update-elt címke optimális értéket ad.

Az algoritmus elemzése

Az algoritmus elemzése

- Nyilván legfeljebb $|V| - 1$-szer végzünk el bővítő lépést.

Az algoritmus elemzése

- Nyilván legfeljebb $|V| - 1$-szer végzünk el bővítő lépést.

- Nyilván legfeljebb $|E|$-szer végzünk számításokat, amik címke átíráshoz vezethetnek.

Az algoritmus elemzése

- Nyilván legfeljebb $|V| - 1$-szer végzünk el bővítő lépést.
- Nyilván legfeljebb $|E|$-szer végzünk számításokat, amik címke átíráshoz vezethetnek.
- Az algoritmus lépésigénye

$$\mathcal{O}(|V| \cdot |E|).$$

Az algoritmus elemzése

- Nyilván legfeljebb $|V| - 1$-szer végzünk el bővítő lépést.
- Nyilván legfeljebb $|E|$-szer végzünk számításokat, amik címke átíráshoz vezethetnek.
- Az algoritmus lépésigénye

$$\mathcal{O}(|V| \cdot |E|).$$

- Van „okosabb” algoritmus is.

Végső megjegyzések

Végső megjegyzések

- Minden véges címke mögött lehet egy él, amely azt jelenti, hogy „hogyan jutottunk ide”.

Végső megjegyzések

- Minden véges címke mögött lehet egy él, amely azt jelenti, hogy „hogyan jutottunk ide”.

- Ha egy update megváltoztatja a címke értékét, akkor a jelző élt is megváltoztatjuk az algoritmus logikája szerint.

Végső megjegyzések

• Minden véges címke mögött lehet egy él, amely azt jelenti, hogy „hogyan jutottunk ide”.

• Ha egy update megváltoztatja a címke értékét, akkor a jelző élt is megváltoztatjuk az algoritmus logikája szerint.

• Az algoritmus végén a szélességi kereső fa egy súlyozott változatát is megkapjuk.

Vége van!

Köszönöm a figyelmet!