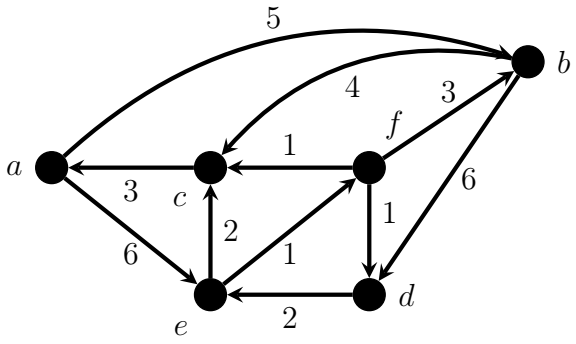
1. Egy irányított gráf csúcshalmaza $\{a,b,c,d,e,f\}$, az élek és súlyaik pedig az alábbiak: $c(a,b) = 5$, $c(a,e) = 6$, $c(b,c) = 4$, $c(b,d) = 6$, $c(c,a) = 3$, $c(d,e) = 2$, $c(e,c) = 2$, $c(e,f) = 1$, $c(f,b) = 3$, $c(f,c) = 1$, $c(f,d) = 1$. Dijkstra módszerével határozza meg a-ból az összes többi csúcsba vezető legrövidebb út hosszát. (Indokolni nem kell, de látszódjon, lépésenként hogyan változik a $D[\ ]$ és a $P[\ ]$ tömb és a KÉSZ halmaz, illetve mi az egyes pontok távolsága a-tól.)

Megoldás:



D:

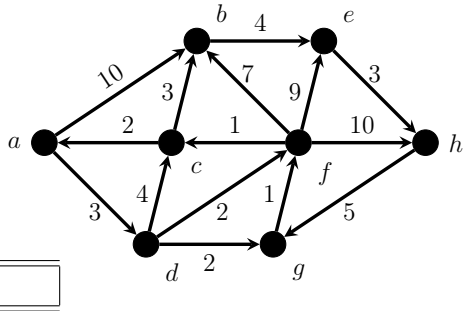
	a	b	c	d	e	f	KÉSZ
1.	0	5	∞	∞	6	∞	$\{a\}$
2.	0	5	9	11	6	∞	$\{a,b\}$
3.	0	5	8	11	6	7	$\{a,b,e\}$
4.	0	5	8	8	6	7	$\{a,b,e,f\}$
5.	0	5	8	8	6	7	$\{a,b,e,f,c\}$
6.	0	5	8	8	6	7	$\{a,b,e,f,c,d\}$

P:

	a	b	c	d	e	f
1.	–	a	–	–	a	–
2.	–	a	b	b	a	–
3.	–	a	e	b	a	e
4.	–	a	e	f	a	e
5.	–	a	e	f	a	e
6.	–	a	e	f	a	e

---

2. Dijkstra-algoritmussal határozza meg a-ból az összes többi pontba vezető legrövidebb út hosszát és magukat a legrövidebb utakat is. (Indokolni nem kell, de látszódjon, lépésenként hogyan változik a $D[\ ]$ és a $P[\ ]$ tömb és a KÉSZ halmaz, illetve mi az egyes pontok távolsága a-tól.)



Megoldás:

D:

	a	b	c	d	e	f	g	h	KÉSZ
1.	0	10	∞	3	∞	∞	∞	∞	$\{a\}$
2.	0	10	7	3	∞	5	5	∞	$\{a,d\}$
3.	0	10	6	3	14	5	5	15	$\{a,d,f\}$
4.	0	10	6	3	14	5	5	15	$\{a,d,f,g\}$
5.	0	9	6	3	14	5	5	15	$\{a,d,f,g,c\}$
6.	0	9	6	3	13	5	5	15	$\{a,d,f,g,c,b\}$
7.	0	9	6	3	13	5	5	15	$\{a,d,f,g,c,b,e\}$
8.	0	9	6	3	13	5	5	15	$\{a,d,f,g,c,b,e,h\}$

P:

	a	b	c	d	e	f	g	h
1.	–	a	–	a	–	–	–	–
2.	–	a	d	a	–	d	d	–
3.	–	a	f	a	f	d	d	f
4.	–	a	f	a	f	d	d	f
5.	–	c	f	a	f	d	d	f
6.	–	c	f	a	b	d	d	f
7.	–	c	f	a	b	d	d	f
8.	–	c	f	a	b	d	d	f

3. Adjuk meg az összes olyan minimális élszámú irányított gráfot (élsúlyokkal együtt), amely(ek)re az alábbi táblázat a Dijkstra-algoritmusban szereplő D[ ] tömb változásait mutathatja. Adja meg a legrövidebb utakat tartalmazó P[ ] tömb állapotait is.

v_1	v_2	v_3	v_4	v_5	v_6
0	2	6	∞	∞	7
0	2	5	9	∞	6
0	2	5	6	9	6
0	2	5	6	8	6
0	2	5	6	7	6

Megoldás:

4. Mátrixával adott egy város úthálózatának élsúlyozott, irányított gráfja: a csúcsok a csomópontok, az élek a csomópontok közötti közvetlen utak, az élek súlya pedig azt mutatja, hogy mennyi az átlagos idő, ami az út megtételéhez autóval szükséges.

Megoldás: A minimális élszámú gráfokban csak azok az élek szerepelnek, amelyek mentén valamelyik lépésben javítás történt. A harmadik lépésben választási lehetőségünk van, v_4 illetve v_6 közül melyik csúcsot tegyük a KÉSZ halmazba, két minimális gráf lesz. A szomszédsági mátrixok élsúlyokkal és a P

$$\begin{pmatrix} 0 & 2 & 6 & * & * & 7 \\ * & 0 & 3 & 7 & * & 4 \\ * & * & 0 & 1 & 4 & * \\ * & * & * & 0 & 2 & * \\ * & * & * & * & 0 & * \\ * & * & * & * & 1 & 0 \end{pmatrix}$$

P :

–	v_2	v_3	v_4	v_5	v_6
–	v_1	v_1	–	–	v_1
–	v_1	v_2	v_2	–	v_2
–	v_1	v_2	v_3	v_3	v_2
–	v_1	v_2	v_3	v_4	v_2
–	v_1	v_2	v_3	v_6	v_2

$$\begin{pmatrix} 0 & 2 & 6 & * & * & 7 \\ * & 0 & 3 & 7 & * & 4 \\ * & * & 0 & 1 & 4 & * \\ * & * & * & 0 & 1 & * \\ * & * & * & * & 0 & * \\ * & * & * & * & 2 & 0 \end{pmatrix}$$

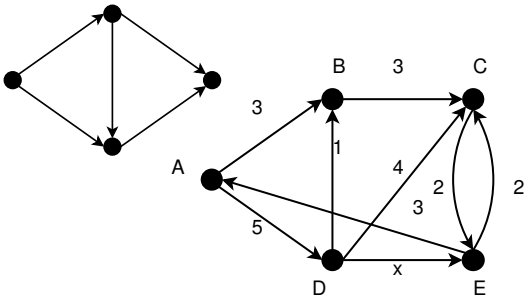
P :

v_1	v_2	v_3	v_4	v_5	v_6
–	v_1	v_1	–	–	v_1
–	v_1	v_2	v_2	–	v_2
–	v_1	v_2	v_3	v_3	v_2
–	v_1	v_2	v_3	v_6	v_2
–	v_1	v_2	v_3	v_4	v_2

Útfelújítások miatt a következő héten le fogják zárni a város két csomópontját, a-t és b-t (ezeken nem lehet autóval áthaladni). Adott a gráfban két kijelölt csúcs, S és T és azt szeretnénk eldönteni, hogy az a és b csomópontok lezárása miatt növekedni fog-e az S-ből T-be eljutás ideje és ha igen, akkor mennyivel. (Tételezzük fel, hogy a közvetlen utakhoz rendelt átlagos idők nem változnak a lezárások következtében.) Melyik tanult algoritmust lehet alkalmazni, hogyan és miért, ha $O(n^2)$ lépésben meg akarjuk oldani ezt a feladatot (a szokásos módon n a csomópontok számát jelöli)?

Megoldás: Először futtassuk le a Dijkstra-algoritmust S-ből kiindulva, ami megadja az eredeti gráfban az S-ből az összes többi csúcsba, köztük a T-be vezető legrövidebb út hosszát. Ez a legrövidebb $S \to T$ út meghatározásával együtt $O(n^2)$ lépés mátrixos megadás esetén. Ezután töröljük ki a gráfban az a-ba és b-be vezető, valamint az ezekből a csúcsokból kifelé vezető éleket. Ez $O(n)$ lépés, hiszen csak a mátrix két sorában és két oszlopában kell a korábbi élsúlyok helyére ∞-ket beírni. Az így kapott gráfnak még mindig n csúcsa van, így S-ből ebben a gráfban indítva Dijkstra algoritmusát, $O(n^2)$ lépésben megkapjuk az új gráfban a legrövidebb út hosszát S-ből T-be (és minden más csúcsba is). Az új gráfban az S-ből a T-be legrövidebb út az eredeti gráfban a legrövidebb olyan $S \to T$ útnak felel meg, ami nem megy át sem az a, sem a b csomóponton. Ezért az új gráfban a legrövidebb $S \to T$ út hosszából az eredeti gráfbeli legrövidebb $S \to T$ út hosszát kivonva megkapjuk, hogy mennyivel változott az S-ből T-be való eljutás ideje; ez a különbség nem lehet negatív, és ha 0-val egyenlő, akkor az eljutás ideje nem változott. Ez $O(1)$ lépés, így a teljes eljárás $O(n^2)$ lépés.
Megjegyzés: Ha $S \notin \{a, b\}$, akkor elég kitörölni az a-ba és a b-be bejövő éleket, illetve ha $T \notin \{a, b\}$, akkor elég kitörölni az a-ba és a b-be bejövő éleket (de mindenképpen ki kell törölni legalább az összes kimenőt vagy az összes bejövőt. A lényeg az, hogy az a és b csúcsokon ne lehessen keresztülhaladni, illetve ha $S \in \{a,b\}$ vagy $T \in \{a,b\}$, akkor az új gráfban nem lehet eljutni S-ből T-be).

5. Rendeljen hozzá élsúlyokat az alábbi gráf éleihez úgy, hogy a keletkező gráfban Dijkstra algoritmusa rosszul számolja ki a legrövidebb utak hosszait.



Megoldás: Jelölje a gráf csúcsait balról jobbra a,b,c,d az éleit (a,b),(a,c),(b,c),(b,d),(c,d). Legyen (a,b) súlya 3, (a,c) súlya 2, (b,c) súlya -2, a többi él súlya tetszőleges. A gráfban nincs negatív kör, mert DAG. A csúcsok közötti távolságok tehát értelmesek. Az a-ból indított Dijkstra az a-c távolságra 2-t ad (az a után közvetlenül a c kerül a KÉSZ-be), pedig az a-c távolság valójában 1.

6. Dijsktra-algoritmussal határozza meg az alábbi gráfban az A pontból az összes többi pontba menő legrövidebb utak hosszát az x pozitív valós paraméter függvényében. Minden lépésnél írja fel a távolságokat tartalmazó D tömb állapotát és a KÉSZ halmaz elemeit.

Megoldás: (Vázlat.) Az első három bővítő lépés után a KÉSZ halmaz $\{A,B,D\}$ lesz, az x értékétől függetlenül. A $D[\,]$ tömb értékei ekkor rendre $0,3,6,5,x+5$. Két fő eset van innen aszerint, hogy C vagy E lesz-e a KÉSZ következő eleme. Ha $x<1$ akkor E a következő, ha $x=1$ akkor mindkét eset lehetséges, ha pedig $x>1$ akkor C kerül negyediknek a KÉSZ halmazba. Ezeket az eseteket végigszámolva azt kapjuk, hogy a végső $D[\,]$ tömb értékei $0,3,6,5,x+5$, ha $x\le 3$, különben pedig, ha $x>3$, akkor $0,3,6,5,8$.

Ha $x<1$, akkor a számolás:

D :

	A	B	C	D	E	KÉSZ
1.	0	3	∞	5	∞	$\{A\}$
2.	0	3	6	5	∞	$\{A,B\}$
3.	0	3	6	5	$x+5$	$\{A,B,D\}$
4.	0	3	$\min\{6,x+7\}=6$	5	$x+5$	$\{A,B,D,E\}$
5.	0	3	6	5	$x+5$	$\{A,B,D,E,C\}$

Ha $x\ge 1$, akkor a számolás:

D :

	A	B	C	D	E	KÉSZ
1.	0	3	∞	5	∞	$\{A\}$
2.	0	3	6	5	∞	$\{A,B\}$
3.	0	3	6	5	$x+5$	$\{A,B,D\}$
4.	0	3	6	5	$\min\{8,x+5\}$	$\{A,B,D,C\}$
5.	0	3	6	5	$\min\{8,x+5\}$	$\{A,B,D,C,E\}$

7. Mátrixával adott egy város úthálózatának összefüggő, élsúlyozott, irányított gráfja: a csúcsok a csomópontok, az élek a csomópontok közötti közvetlen utak, az élek súlya pedig azt mutatja, hogy mennyi idő alatt tud az adott szakaszon egy biciklis futár végigmenni. Egy, az f csúcsban tartózkodó biciklis futár azt a feladatot kapja, hogy a nála levő két csomagot a lehető leggyorsabban kézbesítse ki a város b és c csomópontjaiba (az mindegy, hogy milyen sorrendben kézbesít). Melyik tanult algoritmust lehet alkalmazni és hogyan, hogy $O(n^2)$ lépésben meghatározzuk, hogy milyen sorrendben kell a futárnak a csomagokat leadnia és mennyi a legrövidebb idő, ami alatt teljesíteni tudja a feladatát?

Megoldás: Jelölje a G input gráf u és v csúcsai között időben mért (legrövidebb) távolságot $d(u,v)$ (amiről nem tesszük fel, hogy szimmetrikus). Ekkor a keresett leggyorsabb kézbesítési idő a $d(f,a)+d(a,b)$ és a $d(f,b)+d(b,a)$ értékek közül a nem nagyobb: vagy a-ba megy először a futár, utána b-be, vagy fordítva, először b-be utána a-ba. Az értékek megkaphatók a G gráfban három Dijkstra-futtatással, egyet f-ből, egyet a-ból, egyet b-ből indítva. Ezek mindegyikének $O(n^2)$ a költsége; ennyi lesz az összköltség is.

8. A mátrixával adott G irányított gráf élei között van egy negatív súlyú él, a többi él súlya pozitív. A gráfban nincs negatív súlyú kör. Adjon $O(n^2)$ lépésszámú algoritmust az $s \in V(G)$ pontból az összes többi pontba vezető legrövidebb utak meghatározására.

   Megoldás: Ha nincs negatív összsúlyú kör, akkor az egyik legrövidebb séta biztosan út. Legyen (u,v) a negatív él. A legrövidebb út vagy átmegy az (u,v) élen vagy nem. Ha nem megy át, akkor elég egy legrövidebb utat keresni $G-(u,v)$-ben. Ha átmegy, akkor először keresünk egy legrövidebb utat s-ből u-ba, majd egy legrövidebb utat v-ből t-be. Ezt két utat az (u,v) éllel összekötve kapjuk a legrövidebb olyan élsorozatot s-ből t-be, ami átmegy az (u,v) élen. Lehetséges, hogy ez nem út, mert az út első fele és második fele is átmegy ugyanazon a ponton. Mivel azonban nincs negatív kör, ezért könnyen látható, ilyenkor van egy legfeljebb ilyen hosszú út, ami nem megy át az (u,v) élen.

   Összefoglalva, csináljunk egy Dijkstrát $G-(u,v)$-ben s-ből is és v-ből is. Minden t pontra a legrövidebb út

   $$\min\{d(s,t), d(s,u) + c(u,v) + d(v,t)\}.$$

9. Adott egy G egyszerű irányított gráf éllistával, nemnegatív élsúlyokkal. Legyen v_1 egy tetszőleges csúcsa G-nek. Adjunk minél hatékonyabb módszert a v_1-hez legközelebbi 100 csúcs megtalálására!

   Megoldás: A Dijkstra-algoritmus kupacot nem használó változatának első 100 menetét hajtsuk végre! Ekkor a KÉSZ halmazba „a” 100 legközelebbi csúcs kerül. A módszer fontosabb összetevőinek a költsége: kezedti táblázatkitöltés: $O(n)$, 100 minimumkeresés: $100O(n) = O(n)$, a táblázat módosítása 100-szor: $100O(n) = O(n)$. Az összköltség tehát $O(n)$. (Ez kisebb, mint az input mérete: nem kell az egész éllistát végigolvasni.) Ha a 2-kupacos változatot használjuk, rosszabbul járunk, mert az $O(n)$ kulcsmódosítás összköltsége $O(n \log n)$ lesz. Meggondolható, hogy $d = [\sqrt{n}]$-nel d-kupacot használva szintén $O(n)$ összköltségű algoritmust kapunk.