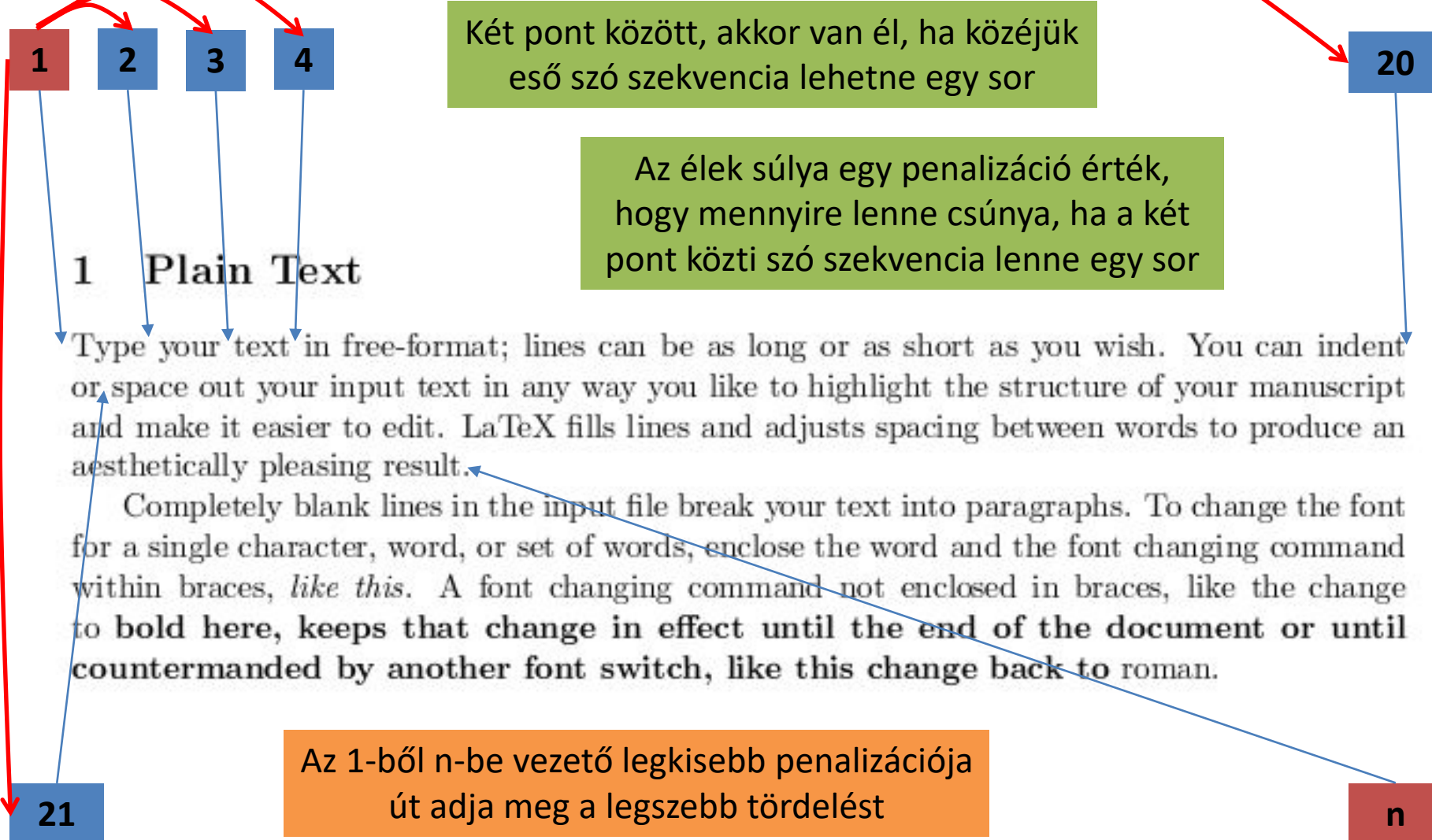




Typesetting in TeX



Applications

Shortest-paths is a broadly useful **problem-solving model**

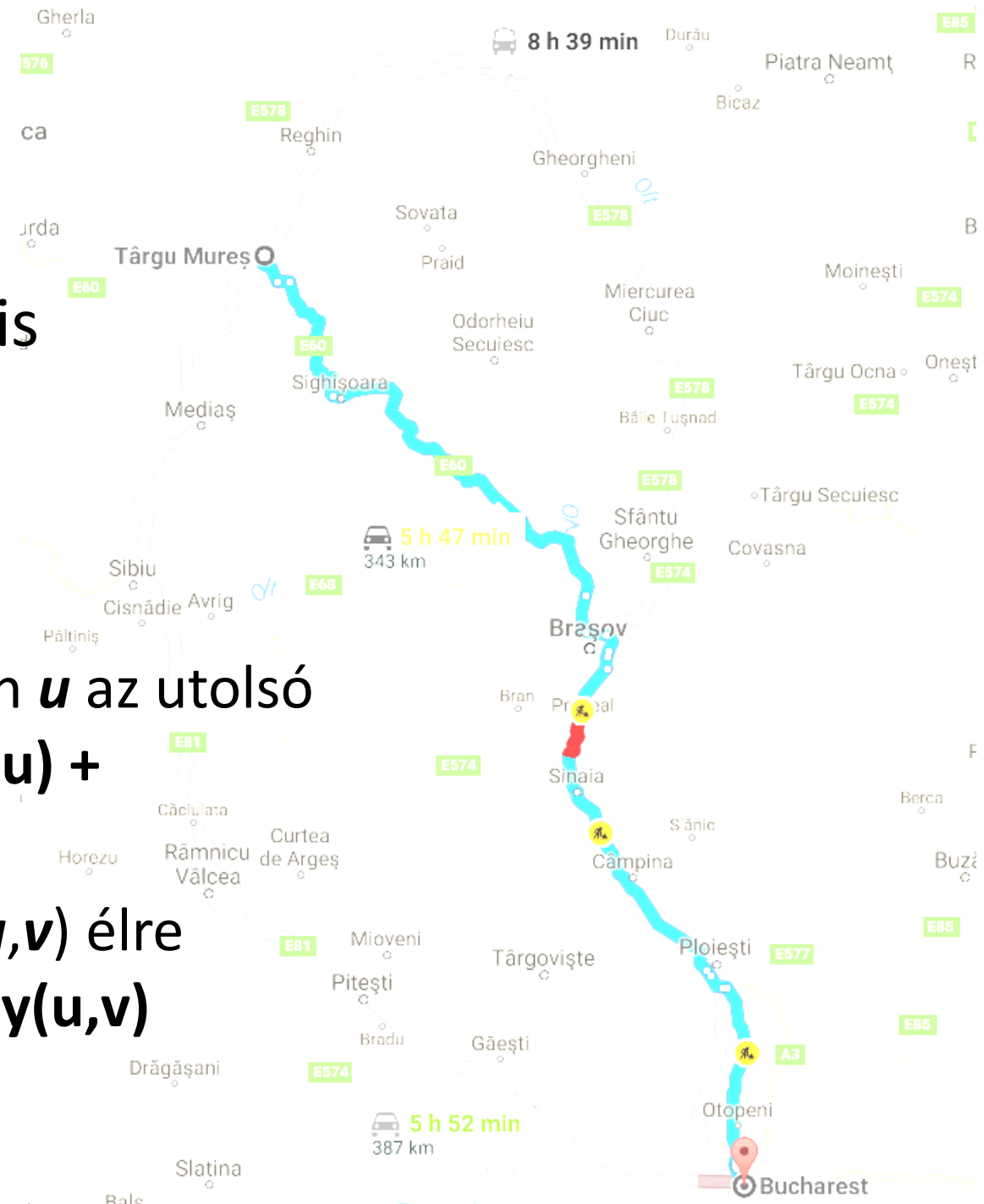
- **Maps**
- Robot navigation.
- Texture mapping.
- Typesetting in TeX.
- Urban traffic planning.
- Optimal pipelining of VLSI chip.
- Subroutine in advanced algorithms.
- Telemarketer operator scheduling.
- Routing of telecommunications messages.
- Approximating piecewise linear functions.
- Network routing protocols (OSPF, BGP, RIP).
- **Exploiting arbitrage opportunities in currency exchange.**
- Optimal truck routing through given traffic congestion pattern.

Optimális út problémák

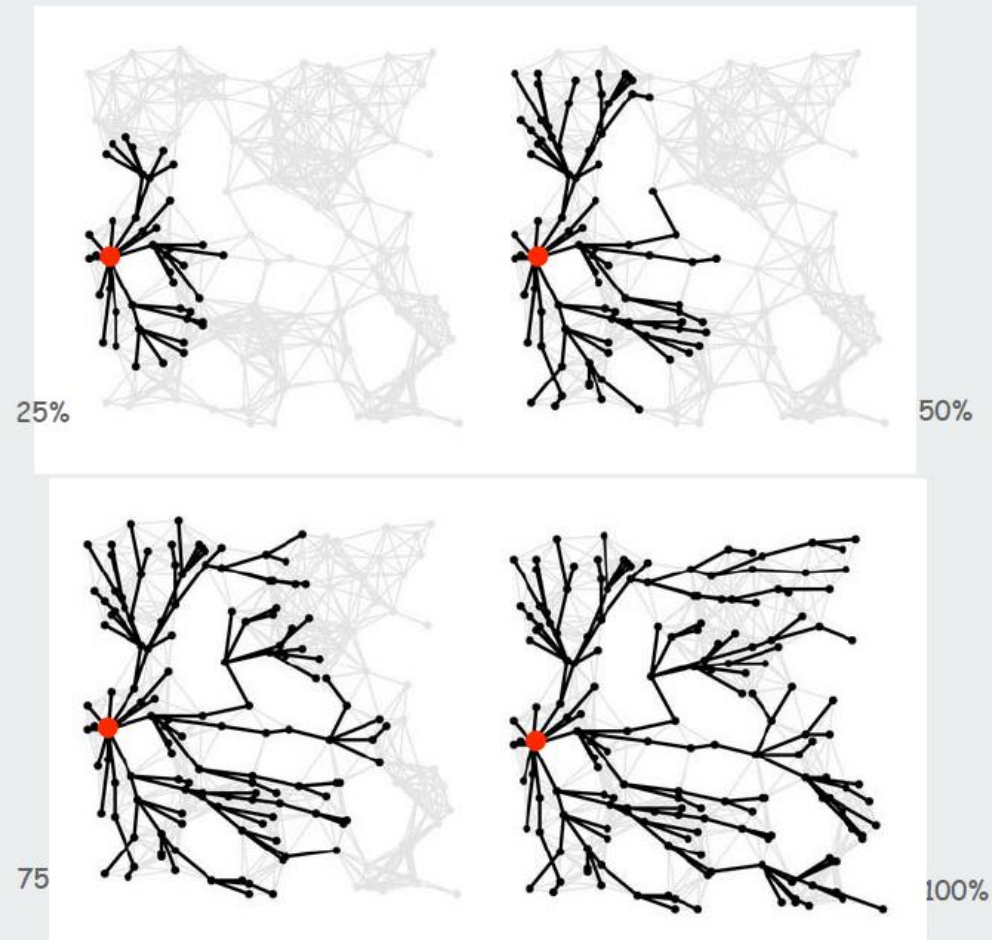
- Egy pontból induló *legrövidebb* utak
 - ~~KÖR~~ (Viterbi)
 - KÖR, ~~NEGATÍV-ÉL~~ (Dijkstra)
 - KÖR, NEGATÍV-ÉL, ~~NEGATÍV-KÖR~~ (Bellman-Ford)
- Bármely pontpár közötti *legrövidebb* utak
 - KÖR, NEGATÍV-ÉL, ~~NEGATÍV-KÖR~~ (Floyd)
- Egy pontból induló *leghosszabb* utak
 - ~~KÖR~~ (PERT módszer)

Optimalitás alapelve

- A legrövidebb utak (lru) részútjai is legrövidebb utak.
 - s : kiinduló pont
 - Ha s -ből v -be tartó legrövidebb úton u az utolsó előtti állomás, akkor $lru(s,v) = lru(s,u) + \text{súly}(u,v)$
 - Ha s a kezdőcsúcs, akkor bármely (u,v) élre fennáll, hogy $lru(s,v) \leq lru(s,u) + \text{súly}(u,v)$



Shortest Path Tree



Az s pontból induló legrövidebb utak egy s gyökerű fát alkotnak

Fokozatos közelítés technikája

- Minden v csúcsra nyilvántartunk egy $d[v]$ értéket, amely az s -ből v -be vezető per-pillanat legrövidebb út súlya.
- $apa[v]$ a per-pillanat legrövidebb út utolsó előtti állomása

Finomítás az (u,v) él mentén

ha $d[v] > d[u] + \text{súly}(u,v)$ **akkor**

$d[v] = d[u] + \text{súly}(u,v)$

$apa[v] = u$

$d[u]$ -ra alapozott javítás $d[v]$ -n.

vége ha

Mindhárom algoritmus élek menti finomítások sorozata, melyek révén a pontok d -értékei fokozatosan ezek lru -értékeihez konvergálnak.

Fokozatos közelítésre épülő

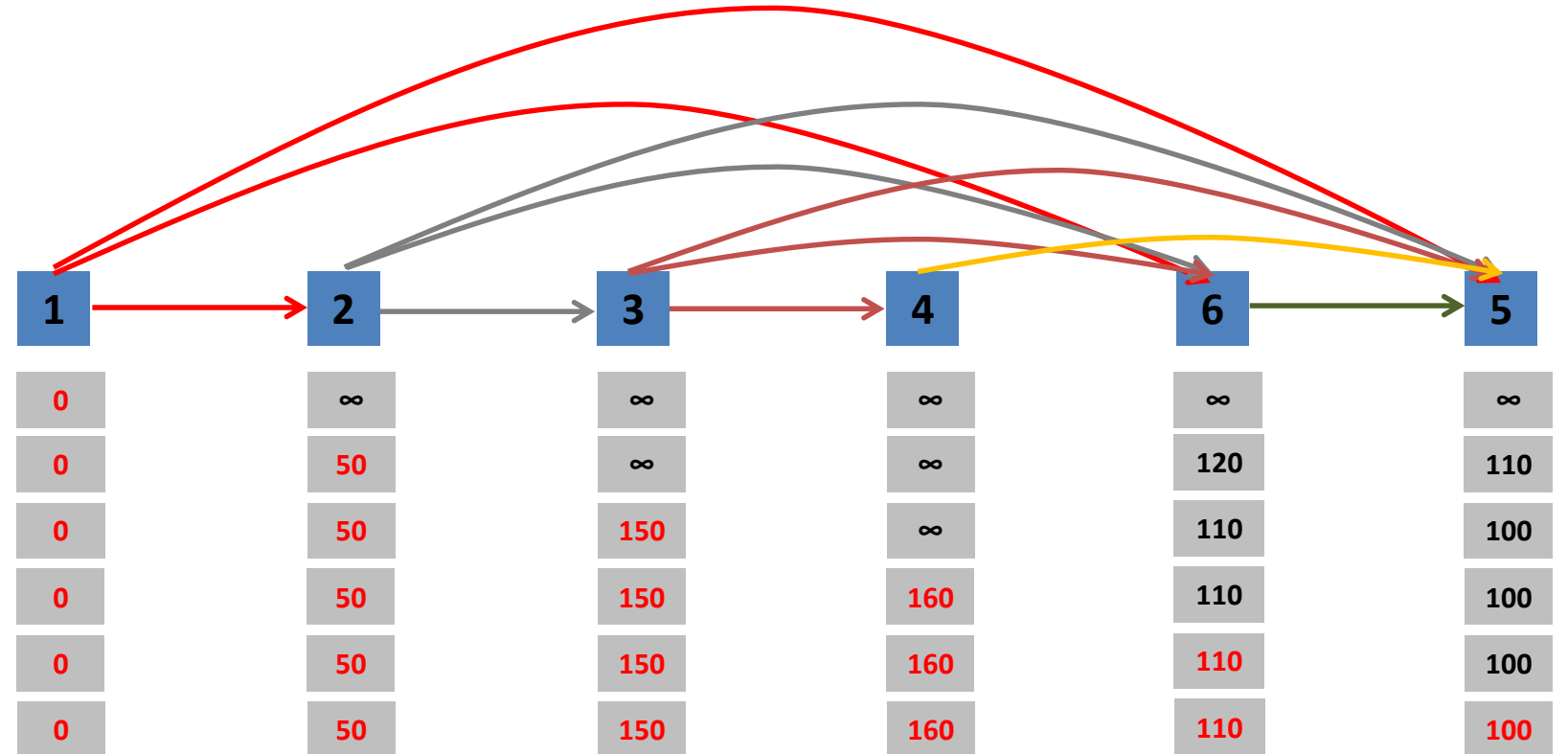
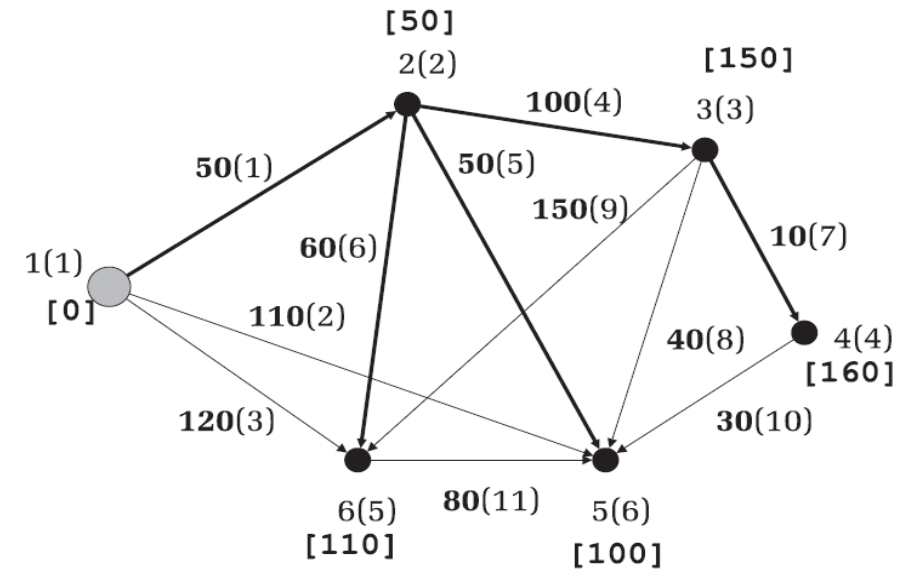
DP algoritmusok (1)

- A **v**-be vezető legrövidebb út felépíthető a **v** be-szomszédjaihoz vezető legrövidebb utakból, a megfelelő be-élek mentén való közelítések révén.
 - Ha **körmentes**, akkor **topologikus sorrend** biztosítja, hogy minden pontot megelőzzenek a be-szomszédjai (VITERBI)
- Ha a gráfnak **nincs negatív éle**, akkor, a **v**-be vezető legrövidebb út felépíthető a **v** azon be-szomszédjaihoz vezető legrövidebb utakból, amelyekhez kisebb a legrövidebb út.
 - **Hosszuk szerint növekvő sorrendben** határozzuk meg a legrövidebb utakat (DIJKSTRA)

Fokozatos közelítésre épülő DP algoritmusok (2)

- $d[s] = 0$; $d[v] = \infty$ (minden többi pontra)
 - Minden lépésben egy újabb pontot csatolunk a lru-fához;
- VITERBI
 - A pontokat topo-sorrendben tekintjük;
 - A kurrens pont d -értékét elkönyveljük lru-értéknek (csatoljuk), és finomítunk a ki-élei mentén.
- DIJKSTRA
 - Minden lépésben a legkisebb d -értékű csatolatlan pontot csatoljuk (d -értékét elkönyveljük lru-értéknek), és finomítunk a ki-élei mentén.
- A kurrens pont lru-értékének elkönyvelése azon alapszik, hogy az előzőleg csatolt pontokra épülő finomítások garantáltan eredményezték már.

VITERBI



VITERBI

SZÜRKE: topo-sorrend szerinti kurrens pont;
 SZÜRKE/FEKETE: garantált lru, és végleges apa értékek;
 FÉLKÖVÉR: kurrens lépésben frissült értékek.

	1	2	3	4	5	6		1	2	3	4	5	6
d	0	∞	∞	∞	∞	∞	→	0	50	∞	∞	110	120
apa	0	0	0	0	0	0		0	1	0	0	1	1

	1	2	3	4	5	6		1	2	3	4	5	6
d	0	50	∞	∞	110	120	→	0	50	150	∞	100	110
apa	0	1	0	0	1	1		0	1	2	0	2	2

	1	2	3	4	5	6		1	2	3	4	5	6
d	0	1	150	∞	100	110	→	0	50	150	160	100	110
apa	0	1	2	0	2	2		0	1	2	3	2	2

	1	2	3	4	5	6		1	2	3	4	5	6
d	0	50	150	160	100	110	→	0	50	150	160	100	110
apa	0	1	2	3	2	2		0	1	2	3	2	2

	1	2	3	4	5	6		1	2	3	4	5	6
d	0	50	150	160	100	110	→	0	50	150	160	100	110
apa	0	1	2	3	2	2		0	1	2	3	2	2

	1	2	3	4	5	6		1	2	3	4	5	6
d	0	50	150	160	100	110	→	0	50	150	160	100	110
apa	0	1	2	3	2	2		0	1	2	3	2	2

VITERBI

Az s pont topo-sorrendbeli indexe

minden $i \leftarrow i_s, n-1$ **végezd**

$u \leftarrow \text{topo}[i]$

minden $v \in \text{szomszéd}(u)$ **végezd**

ha $d[u] + \text{súly}(u, v) < d[v]$ **akkor**

$\text{apa}[v] \leftarrow u$

$d[v] \leftarrow d[u] + \text{súly}(u, v)$

vége ha

vége minden

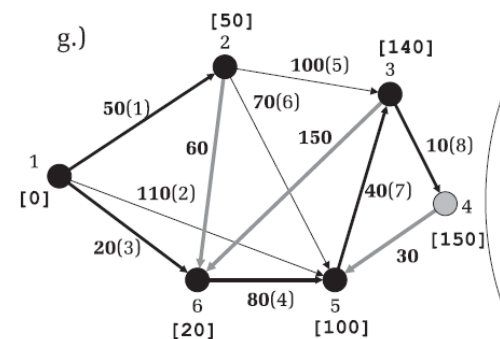
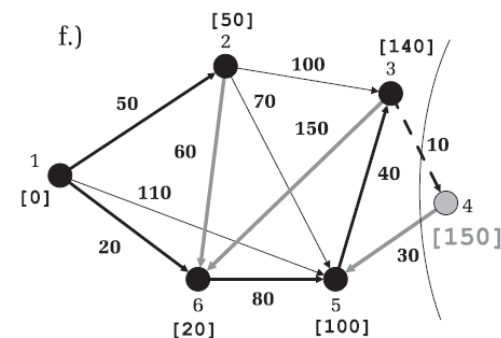
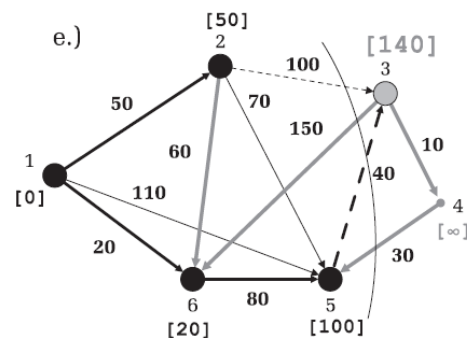
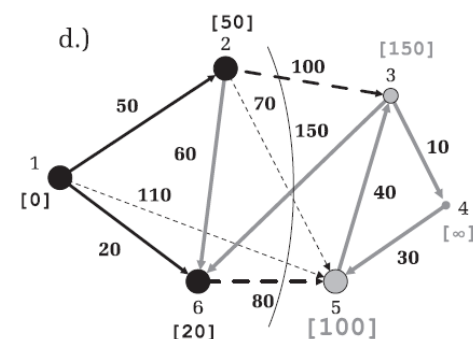
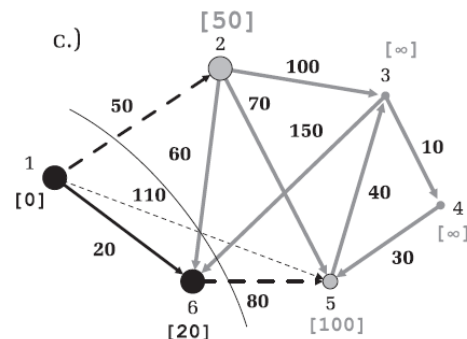
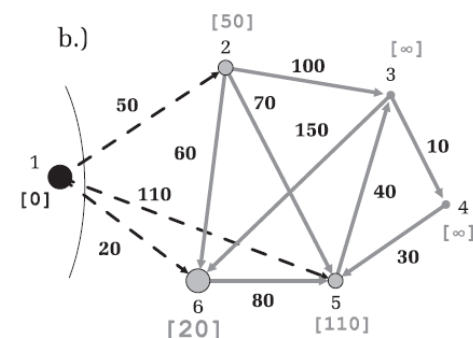
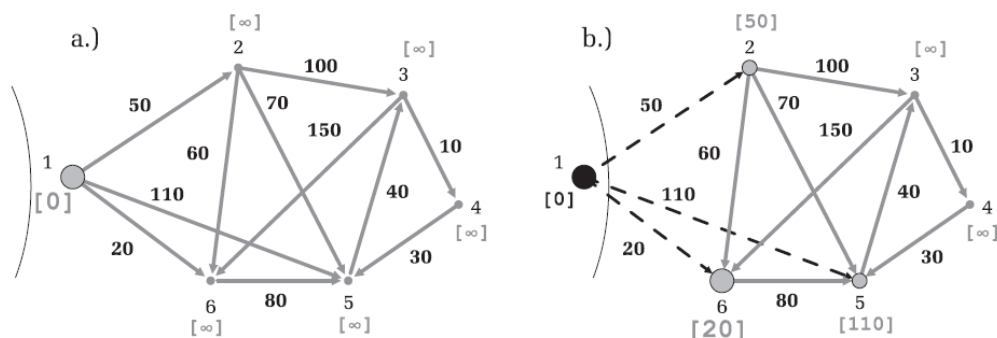
vége minden

- The question of whether computers can think is like the question of whether submarines can swim.
- Do only what only you can do.



Edger Dijkstra
Turing award 1972

DIJKSTRA



Minden lépésben a megnagyított szürke pont a csatolandó, azaz a legkisebb d-értékű ([...]) még csatolatlan pont.

A már csatolt pontok feketék. Ezek d-értéke már lru-értéknek számít.

Az éppen csatolandó/csatolt pont ki-élei mentén finomítunk. Észrevehető, hogy elég csak olyan ki-élek mentén finomítani, amelyek még csatolatlan pontok fele mutatnak (ezek azok, amelyek a körív jelölte vágásban vannak; a vágás elválasztja a már csatolt pontok alkotta lru-fát a még csatolandó pontoktól).

FEHÉR/SZÜRKE: az lru-fához még csatolandó pontok;

SZÜRKE: a kurrens lépésben csatolandó pont (legkisebb d-értékű csatolatlan pont);

SZÜRKE/FEKETE: garantált lru, és végleges apa értékek;

FÉLKÖVÉR: kurrens lépésben frissült értékek.

	1	2	3	4	5	6		1	2	3	4	5	6
d	0	∞	∞	∞	∞	∞	→	0	50	∞	∞	110	20
apa	0	0	0	0	0	0		0	1	0	0	1	1

	1	2	3	4	5	6		1	2	3	4	5	6
d	0	50	∞	∞	110	20	→	0	50	∞	∞	100	20
apa	0	1	0	0	1	1		0	1	0	0	6	1

	1	2	3	4	5	6		1	2	3	4	5	6
d	0	50	∞	∞	100	20	→	0	50	150	∞	100	20
apa	0	1	0	0	6	1		0	1	2	0	6	1

	1	2	3	4	5	6		1	2	3	4	5	6
d	0	50	150	∞	100	20	→	0	50	140	∞	10	20
apa	0	1	2	0	6	1		0	1	5	0	6	1

	1	2	3	4	5	6		1	2	3	4	5	6
d	0	50	140	∞	100	20	→	0	50	140	150	100	20
apa	0	1	5	0	6	1		0	1	5	3	6	1

	1	2	3	4	5	6		1	2	3	4	5	6
d	0	50	140	150	100	20	→	0	50	140	150	100	20
apa	0	1	5	3	6	1		0	1	5	3	6	1

DIJKSTRA

függvény DIJKSTRA}(G,s)

$Q \leftarrow V(G)$

minden $v \in Q$ **végezd**

$d[v] \leftarrow \infty$

vége minden

$d[s] \leftarrow 0$

$apa[s] \leftarrow 0$

amíg $Q \neq \emptyset$ **végezd**

$u \leftarrow \text{KIVESZ_MIN}(Q)$

ha $u = 0$ **akkor**

ugorj

vége ha

minden $v \in \text{szomszéd}(u)$ **végezd**

ha $(v \in Q \text{ ÉS } (d[u] + \text{súly}(u,v) < d[v]))$ **akkor**

$apa[v] \leftarrow u$

$d[v] \leftarrow d[u] + \text{súly}(u,v)$

vége ha

vége minden

vége amíg

vissza apa

vége DIJKSTRA

Akkor térít 0-t vissza, ha nincs több
nem-végtelen d-értékű csatolandó pont.

Viterbi és Dijkstra váll váll-mellett

minden $i \leftarrow 1, n$ végezd

$u \leftarrow \text{topo}[i]$

u ki-szomszédjai száma

$u \leftarrow \text{min_d_erteku_csatolatlan}(d, n)$

minden $j \leftarrow 1, \text{szL}[u][0]$ végezd

$v \leftarrow \text{szL}[u][j]$

u j-edik ki-szomszédja

ha $(d[u] + \text{suly}[u][v] < d[v])$ akkor

$d[v] \leftarrow d[u] + \text{suly}[u][v]$

$\text{apa}[v] \leftarrow u$

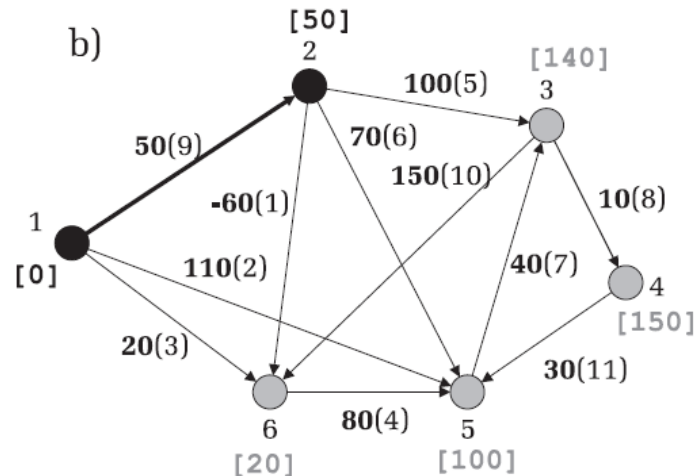
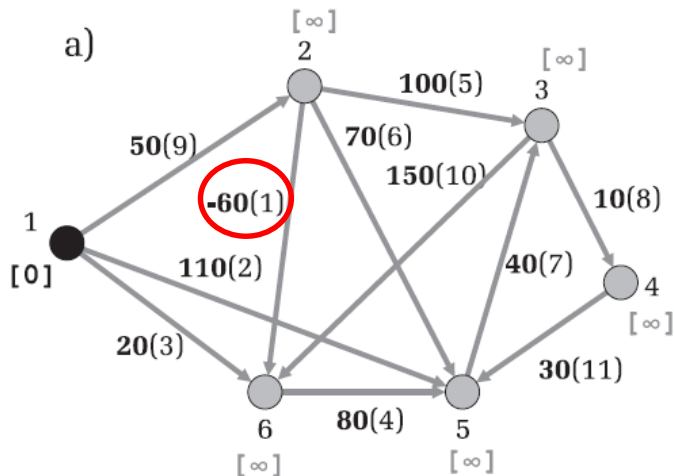
vége ha

vége minden

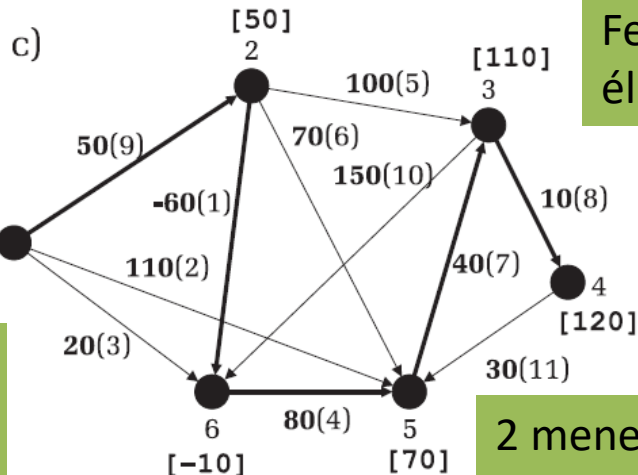
vége minden



BELLMAN-FORD (élek többszöri átjárása)



Bár a direkt él 6-ba rövidebb mint 2-be, a negatív súly miatt, a 6-ba vezető lru az 1-2-6. (Dijkstra mellé fogna)



Fekete ponttal és megvastagított éllel jelöltük az lru-fát.

Újra és újra finomítunk az összes él mentén, ezek ()-ben megjelölt sorrendje szerint.

2 menet után beállnak az lru-értékek; A 3-dik menetben derül ez ki.

Megvastagítottuk azokat a
finomításokat, amelyek lru-
értéket állítanak be.

	Élek	1	2	3	4	5	6
0. menet		0	∞	∞	∞	∞	∞
1. menet	1:(2,6) [-60]	0	∞	∞	∞	∞	∞
	2:(1,5) [110]	0	∞	∞	∞	110	∞
	3:(1,6) [20]	0	∞	∞	∞	110	20
	4:(6,5) [80]	0	∞	∞	∞	100	20
	5:(2,3) [100]	0	∞	∞	∞	100	20
	6:(2,5) [70]	0	∞	∞	∞	100	20
	7:(5,3) [40]	0	∞	140	∞	100	20
	8:(3,4) [10]	0	∞	140	150	100	20
	9:(1,2) [50]	0	50	140	150	100	20
	10:(3,6) [150]	0	50	140	150	100	20
	11:(4,5) [30]	0	50	140	150	100	20
2. menet	1:(2,6) [-60]	0	50	140	150	100	-10
	2:(1,5) [110]	0	50	140	150	100	-10
	3:(1,6) [20]	0	50	140	150	100	-10
	4:(6,5) [80]	0	50	140	150	70	-10
	5:(2,3) [100]	0	50	140	150	70	-10
	6:(2,5) [70]	0	50	140	150	70	-10
	7:(5,3) [40]	0	50	110	150	70	-10
	8:(3,4) [10]	0	50	110	120	70	-10
	9:(1,2) [50]	0	50	110	120	70	-10
	10:(3,6) [150]	0	50	110	120	70	-10
	11:(4,5) [30]	0	50	110	120	70	-10

BELLMAN-FORD

függvény BELLMAN_FORD($G(V,E)$, s , $d[1..n]$, $apa[1..n]$, n)

minden $v \in V(G)$ végezd

$d[v] \leftarrow \infty$

vége minden

$d[s] \leftarrow 0$

$apa[s] \leftarrow 0$

$i \leftarrow 0$

végezd

volt_közelítés $\leftarrow$ HAMIS

minden $(u,v) \in E(G)$ végezd

ha $d[u] + \text{súly}(u,v) < d[v]$ akkor

$apa[v] \leftarrow u$

$d[v] \leftarrow d[u] + \text{súly}(u,v)$

volt_közelítés $\leftarrow$ IGAZ

vége ha

vége minden

$i \leftarrow i + 1$

amíg $(\text{volt_közelítés} = \text{IGAZ}) \text{ ÉS } (i < n)$

ha $(\text{volt_közelítés} = \text{IGAZ}) \text{ ÉS } (i = n)$ akkor

vissza HAMIS

különben

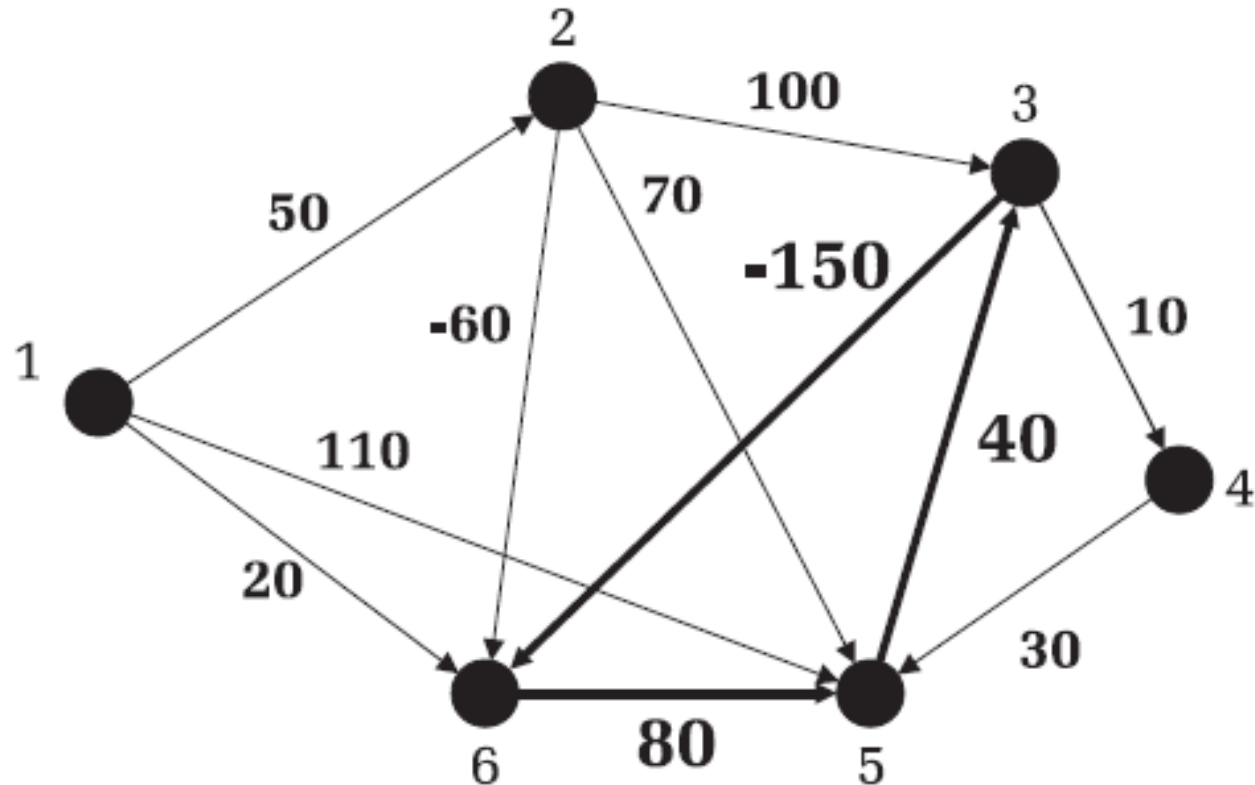
vissza IGAZ

vége ha

vége BELLMAN_FORD

Ha az n -edik menetben is lenne finomítás, akkor a gráfban van negatív kör.

BELLMAN-FORD (negatív kör)

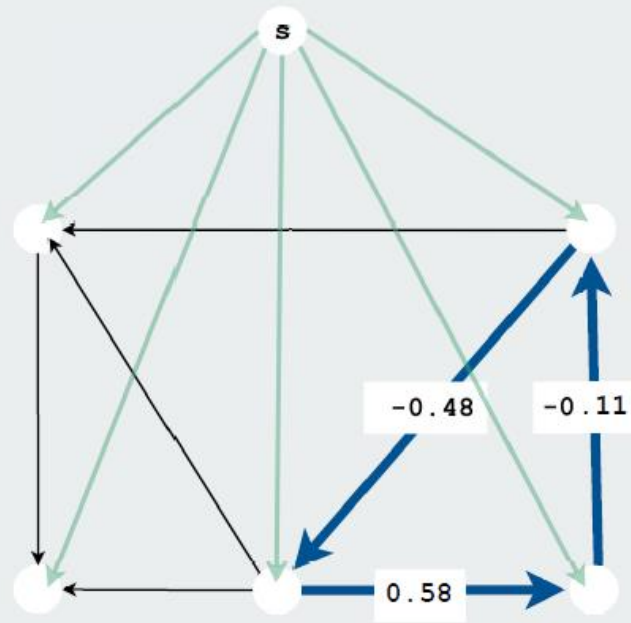


Minden további menetben a negatív kör menti pontokhoz az lru érték csökken a kör összsúlyával

Negative cycle detection

Goal. Identify a negative cycle (reachable from any vertex).

Solution. Add 0-weight edge from artificial source s to each vertex v .
Run Bellman-Ford from vertex s .



BELLMAN-FORD-MORE

Shortest paths with negative weights: Bellman-Ford-Moore algorithm

Observation. If $\text{dist}[v]$ doesn't change during phase i ,
no need to relax any edge leaving v in phase $i+1$.

FIFO implementation.

Maintain **queue** of vertices whose distance changed.

be careful to keep at most one copy of each vertex on queue

Running time.

- still could be proportional to EV in worst case
- much faster than that in practice

Szemléltető PÉLDA

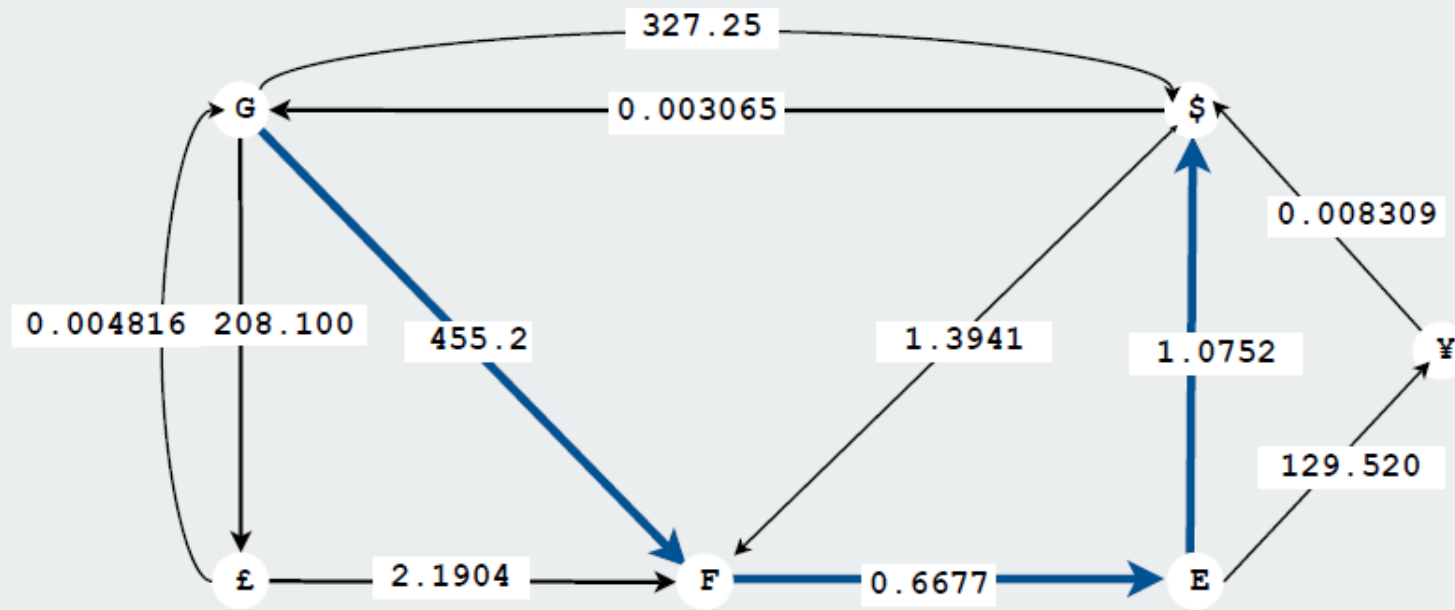
Currency conversion. Given currencies and exchange rates, what is best way to convert one ounce of gold to US dollars?

- 1 oz. gold $\Rightarrow$ \$327.25.
- 1 oz. gold $\Rightarrow$ £208.10 $\Rightarrow$ $\Rightarrow$ \$327.00. [208.10 $\times$ 1.5714]
- 1 oz. gold $\Rightarrow$ 455.2 Francs $\Rightarrow$ 304.39 Euros $\Rightarrow$ \$327.28. [455.2 $\times$.6677 $\times$ 1.0752]

Currency	£	Euro	¥	Franc	\$	Gold
UK Pound	1.0000	0.6853	0.005290	0.4569	0.6368	208.100
Euro	1.4599	1.0000	0.007721	0.6677	0.9303	304.028
Japanese Yen	189.050	129.520	1.0000	85.4694	120.400	39346.7
Swiss Franc	2.1904	1.4978	0.011574	1.0000	1.3941	455.200
US Dollar	1.5714	1.0752	0.008309	0.7182	1.0000	327.250
Gold (oz.)	0.004816	0.003295	0.0000255	0.002201	0.003065	1.0000

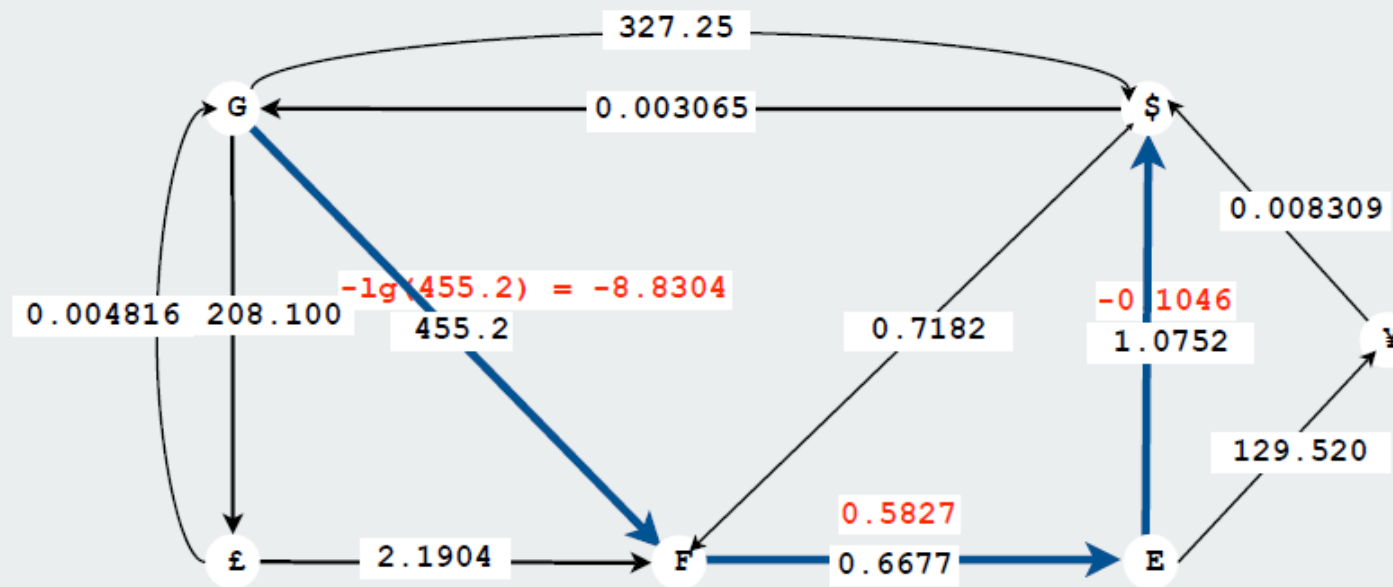
Graph formulation.

- Vertex = currency.
- Edge = transaction, with weight equal to exchange rate.
- Find path that maximizes **product** of weights.



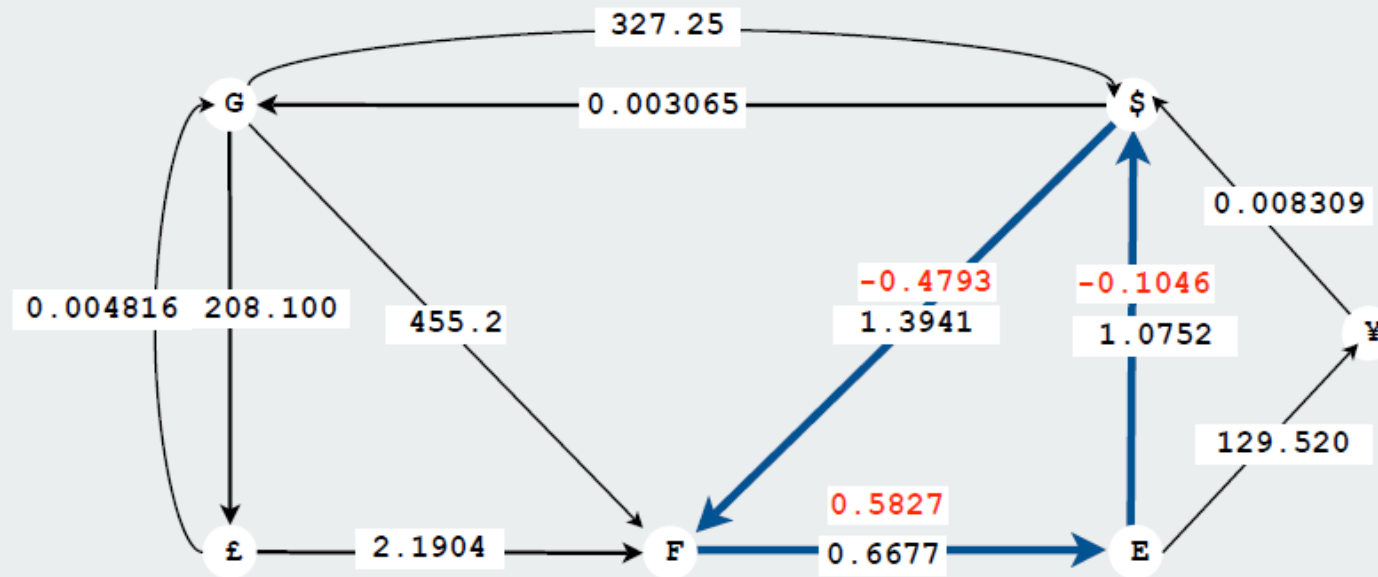
Reduce to shortest path problem by taking logs

- Let $\text{weight}(v-w) = -\lg(\text{exchange rate from currency } v \text{ to } w)$
- multiplication turns to addition
- Shortest path with costs c corresponds to best exchange sequence.



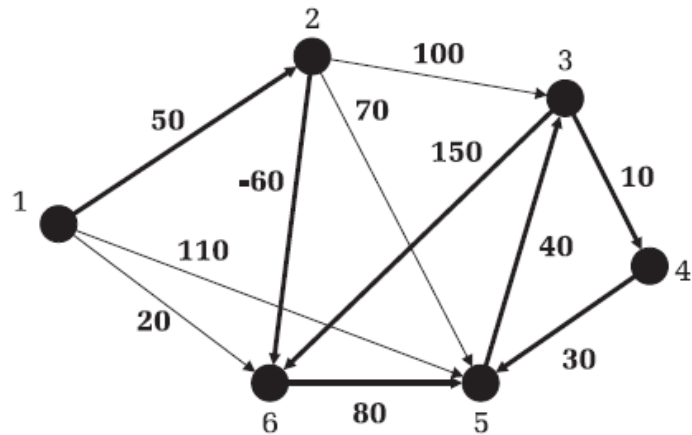
Is there an arbitrage opportunity in currency graph?

- Ex: \$1 $\Rightarrow$ 1.3941 Francs $\Rightarrow$ 0.9308 Euros $\Rightarrow$ \$1.00084.
- Is there a negative cost cycle?
- Fastest algorithm is valuable!



$$-0.4793 + 0.5827 - 0.1046 < 0$$

Legrövidebb utak minden pontpár között



- FLOYD algoritmus
 - Dinamikus programozás
 - A legfentebb $\{1, 2, \dots, k\}$ köztes állomásokat tartalmazó lru-akból ($d_k[1..n][1..n]$) felépítjük a legfentebb $\{1, 2, \dots, k, k+1\}$ köztes állomásokat tartalmazókat ($d_{k+1}[1..n][1..n]$)
 - $d_0[1..n][1..n]$ a súly-mátrix (egyetlen köztes állomást sem tartalmazó lru-ak)
 - Észrevehető, hogy d_{k+1} -vel felülírható d_k .
 - A d tömb kezdetben a súly mátrixot (d_0) tartalmazza, majd frissítjük n -szer
 - A k -adik lépésben d éppen a d_k -t tartalmazza

FLOYD algoritmus



minden $k \leftarrow 1, n$ végezd

minden $i \leftarrow 1, n$ végezd

minden $j \leftarrow 1, n$ végezd

ha ($i \neq k$ ÉS $j \neq k$) akkor

ha ($d[i][k] + d[k][j] < d[i][j]$) akkor

$d[i][j] \leftarrow d[i][k] + d[k][j]$

vége ha

vége ha

vége minden

vége minden

vége minden

Azért írható felül d_k a d_{k+1} -vel, mert a k . lépésben a k . oszlop és sor elemeire alapozva csak a többi elemet ($i \neq k$ ÉS $j \neq k$) kell frissíteni.

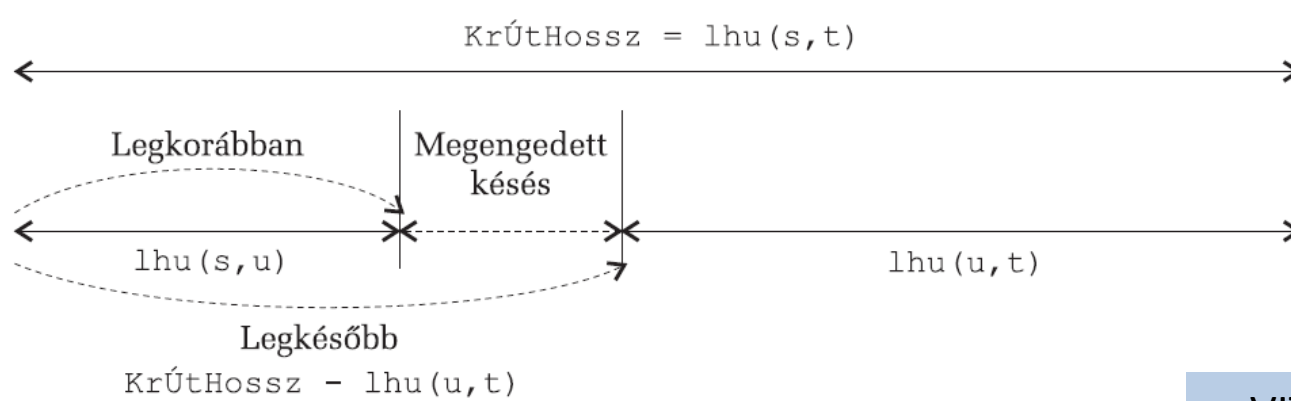
A d tömbben a lru -ak hosszai generálódnak. Ha egy $(n+1)$ -edik menetben is történné finomítás, akkor ez negatív kört jelezne.

Ha minden frissítéskor eltárolnánk egy bizonyos „apa tömbbe” a kurrens k -t ($apa[i][j]=k$), akkor ebből könnyen visszanyerhetők lennének maguk a legrövidebb utak is.

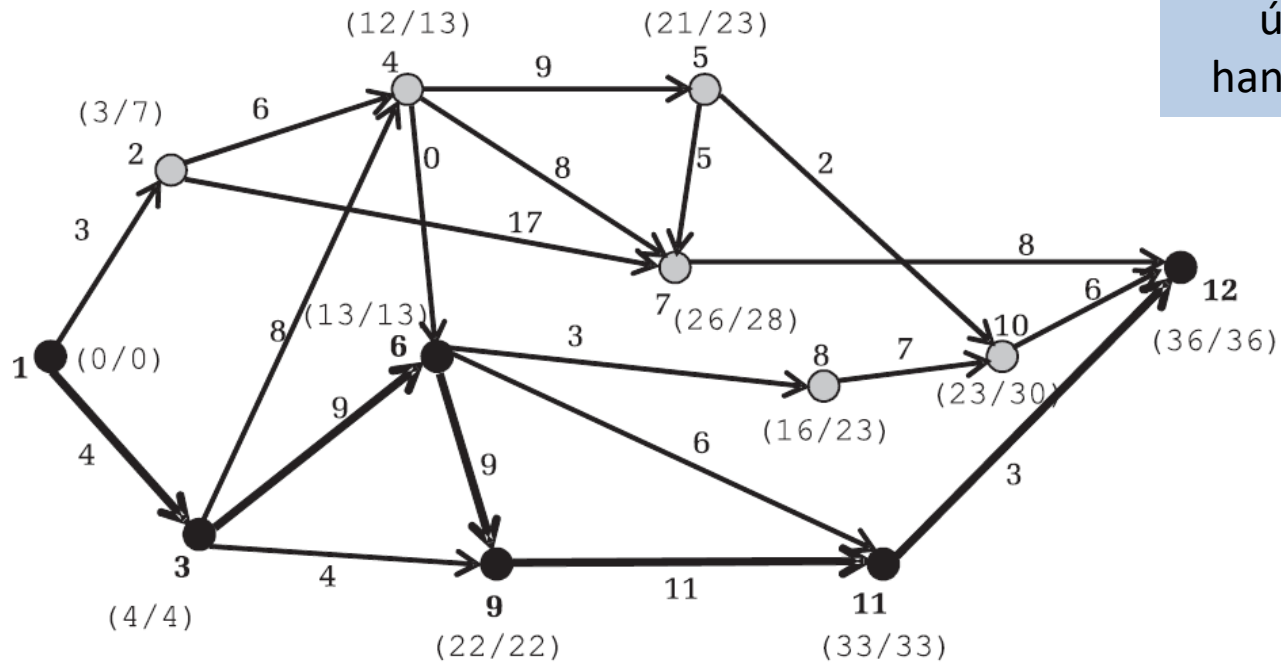
d_0	1	2	3	4	5	6		d_1	1	2	3	4	5	6
1	0	50	∞	∞	110	20		1	0	50	∞	∞	110	20
2	∞	0	100	∞	70	-60		2	∞	0	100	∞	70	-60
3	∞	∞	0	10	∞	150	→	3	∞	∞	0	10	∞	150
4	∞	∞	∞	0	30	∞		4	∞	∞	∞	0	30	∞
5	∞	∞	40	∞	0	∞		5	∞	∞	40	∞	0	∞
6	∞	∞	∞	∞	80	0		6	∞	∞	∞	∞	80	0
d_1	1	2	3	4	5	6		d_2	1	2	3	4	5	6
1	0	50	∞	∞	110	20		1	0	50	150	∞	110	-10
2	∞	0	100	∞	70	-60		2	∞	0	100	∞	70	-60
3	∞	∞	0	10	∞	150	→	3	∞	∞	0	10	∞	150
4	∞	∞	∞	0	30	∞		4	∞	∞	∞	0	30	∞
5	∞	∞	40	∞	0	∞		5	∞	∞	40	∞	0	∞
6	∞	∞	∞	∞	80	0		6	∞	∞	∞	∞	80	0
d_2	1	2	3	4	5	6		d_3	1	2	3	4	5	6
1	0	50	150	∞	110	-10		1	0	50	150	160	110	-10
2	∞	0	100	∞	70	-60		2	∞	0	100	110	70	-60
3	∞	∞	0	10	∞	150	→	3	∞	∞	0	10	∞	150
4	∞	∞	∞	0	30	∞		4	∞	∞	∞	0	30	∞
5	∞	∞	40	∞	0	∞		5	∞	∞	40	50	0	190
6	∞	∞	∞	∞	80	0		6	∞	∞	∞	∞	80	0
d_3	1	2	3	4	5	6		d_4	1	2	3	4	5	6
1	0	50	150	160	110	-10		1	0	50	150	160	110	-10
2	∞	0	100	110	70	-60		2	∞	0	100	110	70	-60
3	∞	∞	0	10	∞	150	→	3	∞	∞	0	10	40	150
4	∞	∞	∞	0	30	∞		4	∞	∞	∞	0	30	∞
5	∞	∞	40	50	0	190		5	∞	∞	40	50	0	190
6	∞	∞	∞	∞	80	0		6	∞	∞	∞	∞	80	0
d_4	1	2	3	4	5	6		d_5	1	2	3	4	5	6
1	0	50	150	160	110	-10		1	0	50	150	160	110	-10
2	∞	0	100	110	70	-60		2	∞	0	100	110	70	-60
3	∞	∞	0	10	40	150	→	3	∞	∞	0	10	40	150
4	∞	∞	∞	0	30	∞		4	∞	∞	70	0	30	220
5	∞	∞	40	50	0	190		5	∞	∞	40	50	0	190
6	∞	∞	∞	∞	80	0		6	∞	∞	120	130	80	0
d_5	1	2	3	4	5	6		d_6	1	2	3	4	5	6
1	0	50	150	160	110	-10		1	0	50	110	120	70	-10
2	∞	0	100	110	70	-60		2	∞	0	60	70	20	-60
3	∞	∞	0	10	40	150	→	3	∞	∞	0	10	40	150
4	∞	∞	70	0	30	220		4	∞	∞	70	0	30	220
5	∞	∞	40	50	0	190		5	∞	∞	40	50	0	190
6	∞	∞	120	130	80	0		6	∞	∞	120	130	80	0

A KRITIKUS ÚT MÓDSZERE

- Egy körmentes irányított súlyozott gráfban, amelynek van egy kitüntetett forrás pontja (s) és egy kitüntetett nyelő pontja (t), kritikus útnak az s-ből t-be vezető leghosszabb utat nevezzük.
 - Egy ilyen gráfot tevékenység gráfként is felfoghatunk.
 - s: munkálatok elkezdésének momentuma
 - t: munkálatok befejezésének momentuma
 - élek: tevékenységek (súlyuk: a tevékenység időigénye)
 - egy pont képviselte esemény akkor következik be, ha minden be-éle képviselte tevékenység befejeződött



VITERBI
leghosszabb
útra
hangolva



A kritikus út menti pontokra a megengedett késés 0. Ha késne valamelyik kritikus út menti tevékenység, akkor késne a teljes munkálat

