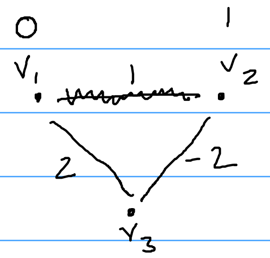
Dijkstra's algorithm may fail in the presence of negative weights.



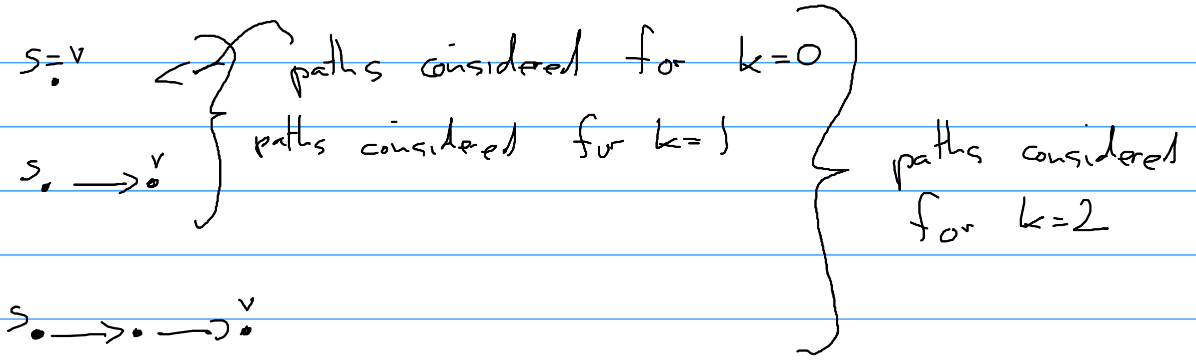
Dijkstra starting from v_1 adds the edge v_1v_2 to the tree, but v_1v_2 is not a shortest v_1-v_2-path in the graph.

Note that Dijkstra starting from v_2 works correctly.

5.3 Shortest Directed Paths (possibly with negative weights)

Consider a directed network (D, w) and $s \in V(D)$. Let $n = |V(D)|$. For $v \in V(D)$ and $k = 0, 1, 2, \ldots, n$, let

$\delta_k(v)$ be the length of a shortest (directed) s-v-path in (D, w) using at most k arcs



In the absence of negative cycles, the length of a shortest s-t-path is equal to $\delta_{n-1}(t)$.
Defining the length of a path that does not exist as ∞, we have

$$\delta_0(v) = \begin{cases} 0 & \text{if } v = s \\ \infty & \text{otherwise} \end{cases}$$

For $k \geq 1$, a shortest s-v-path using at most k arcs can take one of two forms:
either it has at most $k-1$ arcs, or it consists of an s-u-path using at most $k-1$ arcs plus the arc uv

case 1: $s \xrightarrow{\text{at most } k-1 \text{ arcs}} v$

case 2: $s \xrightarrow{\text{at most } k-1 \text{ arcs}} u \xrightarrow{\text{one arc}} v$

In the absence of negative cycles, for any $k \geq 1$,

$$\delta_k(v) = \min \Big\{ \delta_{k-1}(v), \underbrace{\min_{u \in V(D) \setminus \{v\}} \delta_{k-1}(u) + w(uv)}_{\text{length of a shortest } s\text{-}u\text{-path using at most } k-1 \text{ arcs plus the arc } uv} \Big\}$$

length of a shortest s-v-path using at most $k-1$ arcs

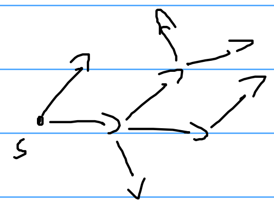
Algorithm (Bellman-Ford) Consider a directed network (D, w) and $s \in V(D)$. Let $n = |V(D)|$. The Bellman-Ford algorithm proceeds as follows:

1. For each $v \in V(D)$, set $\delta_0(v) = 0$ if $v = s$ and $\delta_0(v) = \infty$ otherwise.
2. Repeat the following for $k = 1, 2, \ldots, n-1$:
   For each $v \in V(D)$ do the following:
   a) Let $\delta_k(v) = \min \Big\{ \delta_{k-1}(v), \min_{u \in V(D) \setminus \{v\}} \delta_{k-1}(u) + w(uv) \Big\}$
   b) If $\delta_k(v) < \delta_{k-1}(v)$, let
   $$p(v) = \arg\min_{u \in V(D) \setminus \{v\}} \delta_{k-1}(u) + w(uv)$$

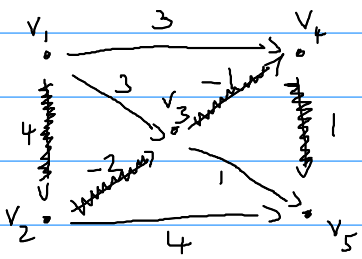
predecessor of v on a shortest s-v-path using at most k arcs

3. Repeat Step 2 one more time to compute $\delta_n(v)$ for all $v \in V(D)$. If for any $v \in V(D)$, $\delta_n(v) < \delta_{n-1}(v)$, then stop! (D,w) contains a cycle of negative length, and the algorithm cannot be used to find shortest s-t-paths in (D,w).

4. Otherwise, the set $\{p(v)v : v \in V(D)\}$ forms a spanning tree of the part of D reachable from s. All s-v-paths in this spanning tree are shortest s-v-paths in (D,w).



Example



run Bellman-Ford from v_1

$\delta_1(v_1) = \min\{\delta_0(v_1)\} = \delta_0(v_1)$

$\delta_1(v_2) = \min\{\delta_0(v_2), \delta_0(v_1) + w(v_1v_2)\}$
$= \min\{\infty, 0+4\}$
$= 4$

$v_1(4)$ $v_1(3)$, $v_2(-2)$ $v_1(3)$, $v_3(-1)$ $v_2(4)$, $v_3(1)$, $v_4(1)$

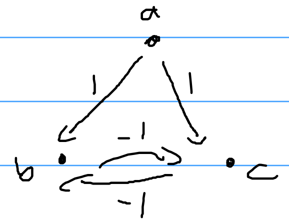
k	v_1	v_2	v_3	v_4	v_5
0	$\underline{0}$	∞	∞	∞	∞
1	0	$4(v_1)$	$3(v_1)$	$3(v_1)$	∞
2	0	4	$\underline{2(v_2)}$	$2(v_3)$	$4(v_3)$
3	0	4	2	$1(v_3)$	$3(v_3)$
4	0	4	$\underline{2}$	1	$2(v_4)$
5	0	4	2	1	2

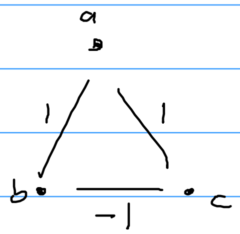
$\delta_1(v_3) = \min\{\delta_0(v_3), \delta_0(v_1) + w(v_1v_3), \delta_0(v_2) + w(v_2v_3)\}$
$= \min\{\infty, 0+3, \infty - 2\}$

$\delta_2(v_3) = \min\{\delta_1(v_3), \delta_1(v_1) + w(v_1v_3), \delta_1(v_2) + w(v_2v_3)\}$

no changes from row 4 to 5, so entries in row 4 are lengths of shortest v_1-t-paths for $t \in V(D)$

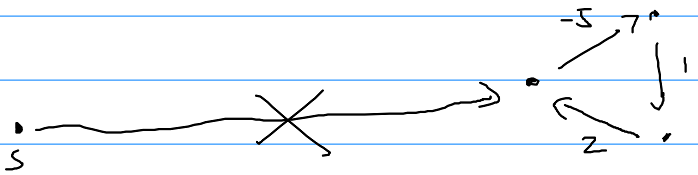
The running time of the Bellman-Ford algorithm is $O(|V(G)|^3)$.
Finding shortest paths in the presence of negative cycles is NP-hard.



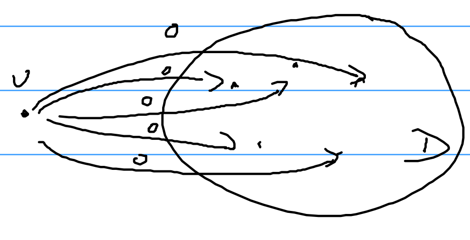


5.4 Directed Cycles of Negative Length

If the algorithm encounters a negative cycle, that cycle is contained in $\{p(v)v : v \in V(D)\}$.



If we want to find negative cycles, we start by adding a new vertex u and arcs of weight 0 from u to each original vertex. Then we run Bellman-Ford from u.



Example where we want to find negative cycles.

Given exchange rates r_{ij} for currencies i, j

Arbitrage exists if

$$r_{12} \cdot r_{23} \cdot r_{34} \cdot r_{41} > 1$$

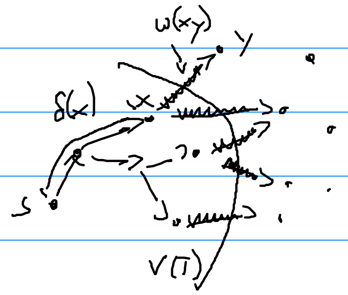
which happens if and only if

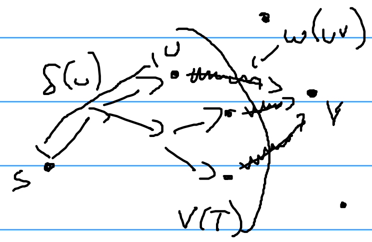
$$-\log(r_{12}) - \log(r_{23}) - \log(r_{34}) - \log(r_{41}) < 0$$

5.5 Longest Paths in Directed Acyclic Networks

<u>Algorithm</u> (Morávek) Consider a directed acyclic network (D, w) and $s \in V(D)$ such that there exists an s-v-path for every $v \in V(D)$. Let $<$ be a topological ordering of D. Morávek's algorithm starts from T with $V(T) = \{s\}$ and $A(T) = \emptyset$ and then repeats the following:

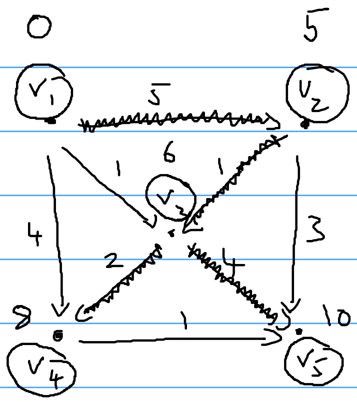
1. Let $v \in V(D) \setminus V(T)$ such that $v < u$ for all $u \in V(D) \setminus V(T) \setminus \{v\}$
2. Let $F = \{uv \in A(D) : u \in V(T)\}$. If $F = \emptyset$, stop.
3. For each $u \in V(T)$, let $\delta(u)$ be the length of the unique s-u-path in T
4. Let $uv \in F$ such that $\delta(u) + w(uv) = \max_{xy \in F} \delta(x) + w(xy)$
5. Add v to $V(T)$ and uv to $A(T)$





Dijkstra chooses xy to minimise $\delta(x) + w(xy)$

Morávek chooses v as the next vertex in the topological ordering and u to <u>maximise</u> $\delta(u) + w(uv)$



topological v_1, v_2, v_3, v_4, v_5

~~the~~ a longest v_1-v_5-path is v_1, v_2, v_3, v_5