Shortest Paths

Input: Directed graph $G = (V, E)$,

each edge (u,v) in E has a weight $wt(u,v) > 0$,

fixed vertex s (source).

Output: for each vertex v:

$\delta(s,v)$ = length of a shortest path from s to v.

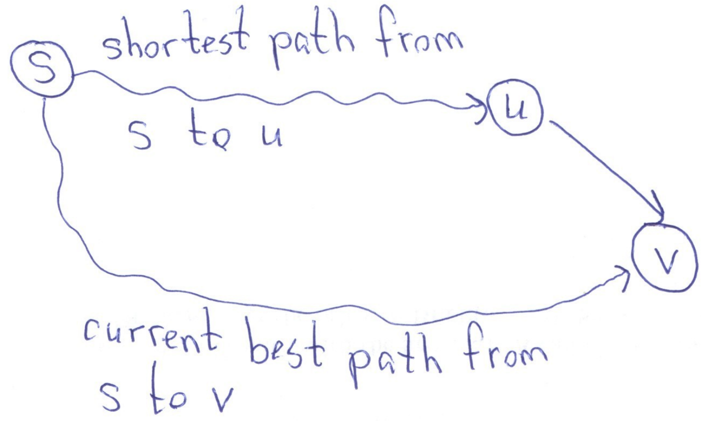
Approach: for each vertex v, maintain variable

$d(v)$ = length of a shortest path from s to v found so far.

Start: $d(s) = 0$, for every vertex $v \neq s$: $d(v) = \infty$.

Loop: Pick a vertex u for which $d(u) = \delta(s,u)$.

For each edge (u,v):



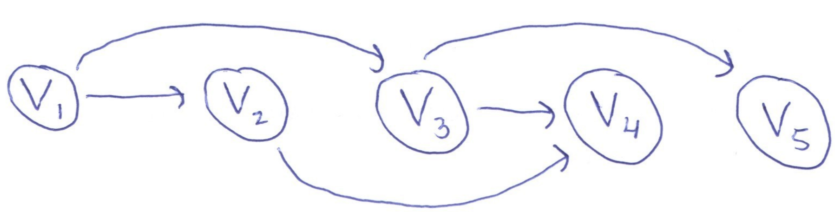
$$d(v) = \min\left(d(v),\ d(u) + wt(u,v)\right).$$

<u>Question</u>: How to pick vertex u? How do we know that $d(u) = \delta(s,u)$?

We start with a simple case: Assume G is acyclic.

<u>Step 1</u>: Topologically sort G; denote the resulting numbering of the vertices by $v_1, v_2, \ldots, v_n$.

Recall: if we put the vertices on a line, then all edges go from left to right



Step 2: Assume the source s is v_1.

$d(s) = 0$;

for $i = 2$ to n : $d(v_i) = \infty$;

for $i = 1$ to n :

  $u = v_i$; // claim: $d(u) = \delta(s, u)$

  for each edge (u, v):

    if $d(u) + wt(u,v) < d(v)$:

      $d(v) = d(u) + wt(u,v)$

What to do if, for example, $s = v_{23}$:

remove $v_1, \dots, v_{22}$ (and their outgoing edges) and run the algorithm for $v_{23}, \dots, v_n$

Running time: $O(|V| + |E|)$.

Correctness:

Claim: for $i = 1, \ldots, n$:

* at the start of iteration i: $d(v_i) = \delta(s, v_i)$
* during and after iteration i: $d(v_i)$ does not change.

① At any moment: $\delta(s, v_i) \le d(v_i)$.

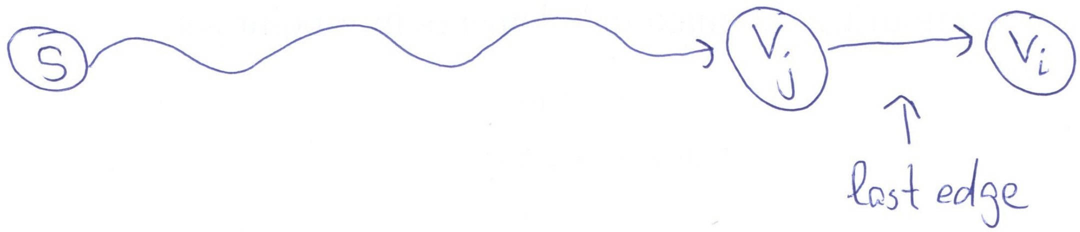
Proof: Either $d(v_i) = \infty$ or $d(v_i)$ is equal to the length of some path from s to v_i. $\square$

② Assume at some moment, $d(v_i)$ becomes equal to $\delta(s, v_i)$. Then, during the rest of the algorithm, $d(v_i)$ does not change.

Proof: From the algorithm: $d(v_i)$ can only decrease.

From ①: $d(v_i) \geq \delta(s, v_i)$ at any moment. $\square$

3. Let $2 \leq i \leq n$. Consider the shortest path from s to v_i.



If $d(v_j) = \delta(s, v_j)$ at the beginning of iteration j, then $d(v_i) = \delta(s, v_i)$ at the end of iteration j.

Proof: Observe: $\delta(s, v_i) = \delta(s, v_j) + wt(v_j, v_i)$.

At the end of iteration j:

$$\begin{aligned} d(v_i) &\leq d(v_j) + wt(v_j, v_i) \\ &= \delta(s, v_j) + wt(v_j, v_i) \\ &= \delta(s, v_i) \\ &\leq d(v_i) \end{aligned}$$

from 1

$\therefore\ d(v_i) = \delta(s, v_i)$.

$\square$

Proof of Claim on page 93: induction on i.

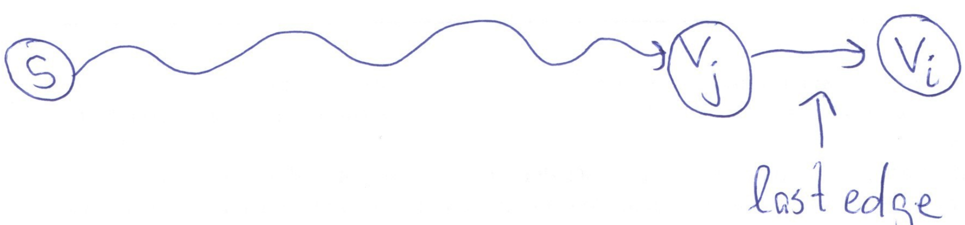
$i=1$: at the start of iteration 1:

$$\left.\begin{array}{l} d(v_1) = d(s) = 0 \\ \delta(s,v_1) = \delta(s,s) = 0 \end{array}\right\} \therefore\ d(v_1) = \delta(s,v_1).$$

from ②: $d(v_1)$ never changes.

Assume the claim is true for $1, 2, \ldots, i-1$.

Consider iteration i. Consider the shortest path from s to v_i:



Since $1 \le j \le i-1$: at the start of iteration j,

$d(v_j) = \delta(s,v_j)$.

From ③: at the end of iteration j: $d(v_i) = \delta(s,v_i)$.

From ②: at the start of iteration i: $d(v_i) = \delta(s,v_i)$

and $d(v_i)$ never changes again.

$\square$

Shortest paths in a directed graph $G=(V,E)$.

each edge (u,v) in E has a weight $wt(u,v)>0$.

source vertex s.

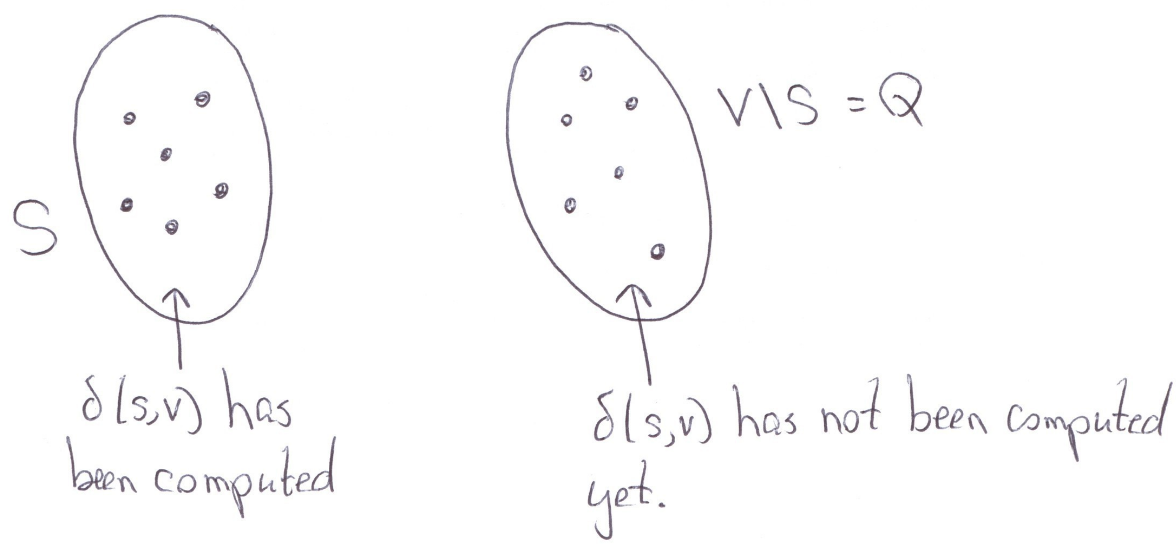
<u>Goal</u>: $\delta(s,v)$ for each vertex v.

<u>Approach</u>: ① for each vertex v, maintain a variable

$d(v)$ = length of a shortest path from s to v found so far.

② maintain $S \subseteq V$ such that for all $v \in S$:

$d(v) = \delta(s,v)$ (i.e., we know $\delta(s,v)$)



Start: $S = \phi$, $Q = V$,

$d(s) = 0$,

$d(v) = \infty$ for each vertex $v \neq s$.

One iteration: grow S, by moving one vertex u from Q to S.

Which vertex u do we move:

u = vertex of Q for which $d(u)$ is minimum.

later, we prove: for this vertex u: $d(u) = \delta(s,u)$.

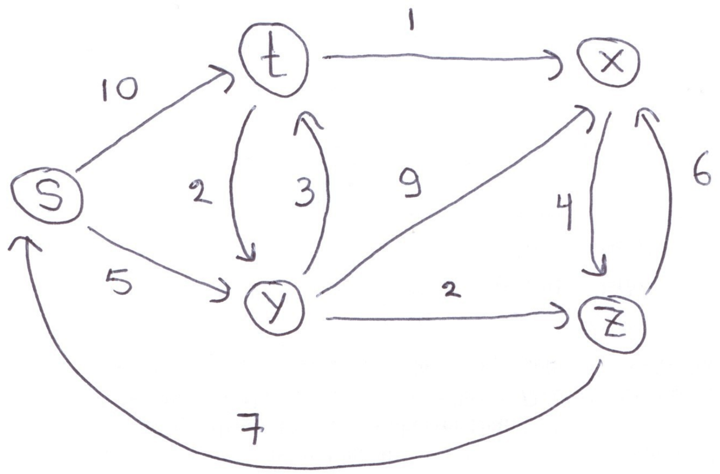
for each edge (u,v):

$d(v) = \min\big(d(v), d(u) + wt(u,v)\big)$.

Dijkstra (1959)

for each $v \in V$: $d(v) = \infty$;
$d(s) = 0$; $S = \phi$; $Q = V$;
while $Q \neq \phi$:
    u = vertex in Q for which $d(u)$ is minimum; (*)
    delete u from Q;
    insert u into S;
    for each edge (u,v):
        if $d(u) + wt(u,v) < d(v)$:
            $d(v) = d(u) + wt(u,v)$

(*) We will prove later that $d(u) = \delta(s,u)$



Q	s	t	x	y	z
d	0	∞	∞	∞	∞

$u = s$

$\delta(s,s) = d(s) = 0$

delete s from Q

update $d(t)$ and $d(y)$

Q	t	x	y	z
d	10	∞	5	∞

$u = y$, $\delta(s,y) = d(y) = 5$

delete y from Q

update $d(t), d(x), d(z)$

Q	t	x	z
d	8	14	7

$u = z$

$\delta(s,z) = d(z) = 7$

delete z from Q

update $d(x)$ and $d(s)$

Q	t	x
d	8	13

$u = t$

$\delta(s,t) = d(t) = 8$

delete t from Q

update $d(x)$ and $d(y)$

Q	x
d	9

$u = x$

$\delta(s,x) = d(x) = 9$

delete x from Q

update $d(z)$

$Q = \phi$; done.

<u>Running time</u> : $n = |V|$, $m = |E|$

Store Q in a min-heap, where the key of each vertex v is $d(v)$.

<u>Initialization</u>: $O(n)$ (this includes the time to build a heap storing $Q = V$)

<u>One iteration</u>:

* find u and delete it from Q:

  extract_min : $O(\log n)$ time

* for each edge (u,v): update $d(v)$:

  decrease_key : $O(\log n)$ time

Total time for one iteration:

$$O(\log n) + O(\text{outdegree}(u) \cdot \log n)$$

Total running time:

$$O(n) + O\left(\sum_{u \in V} (\log n + \text{outdegree}(u) \cdot \log n)\right)$$

$$= O(n \log n + m \log n)$$

$$= O((n+m) \log n).$$