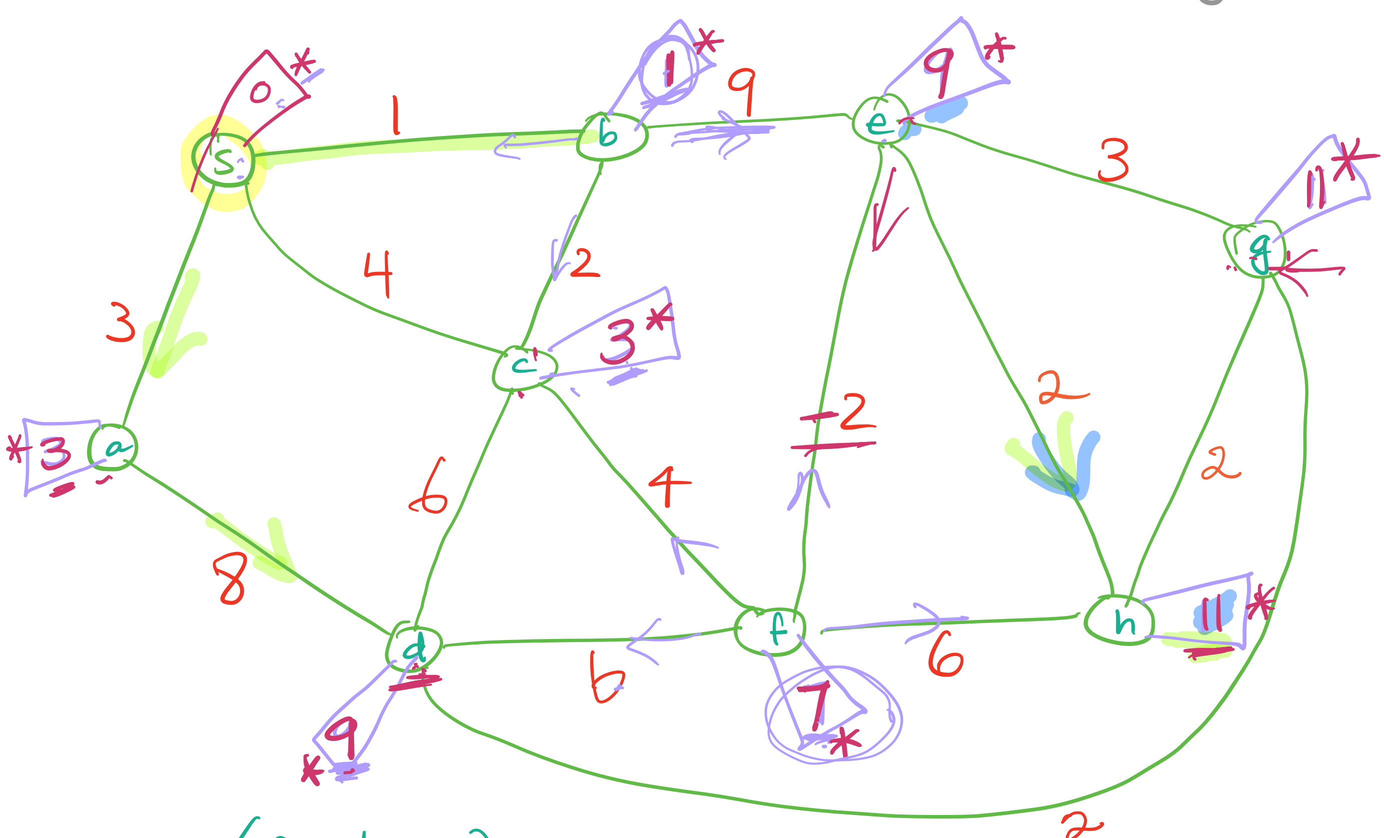
Single-Source Shortest Paths

Input: Graph $G=(V,E)$, weight function $w: E \to \mathbb{R}^+$, source s, $s \in V$

Goal: determine $\forall v \in V$ the shortest s-v path (ie path with min sum of edge weights)



relax(g, d, w)

if $d[g] > d[d] + w(d,g)$

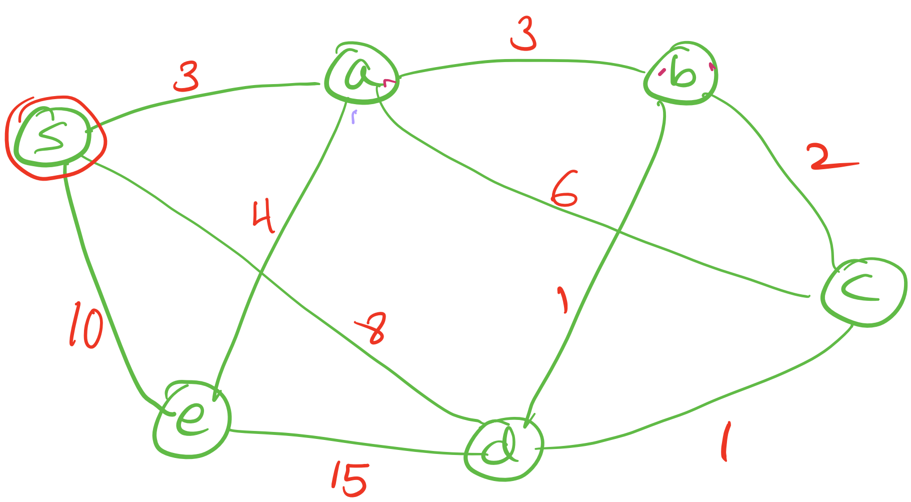
$d[g] = d[d] + w(d,g)$

For a path $\langle v_1, v_2, \ldots, v_k \rangle$, the weight of the path is $\sum_{i=1}^{k-1} w(v_i, v_{i+1})$

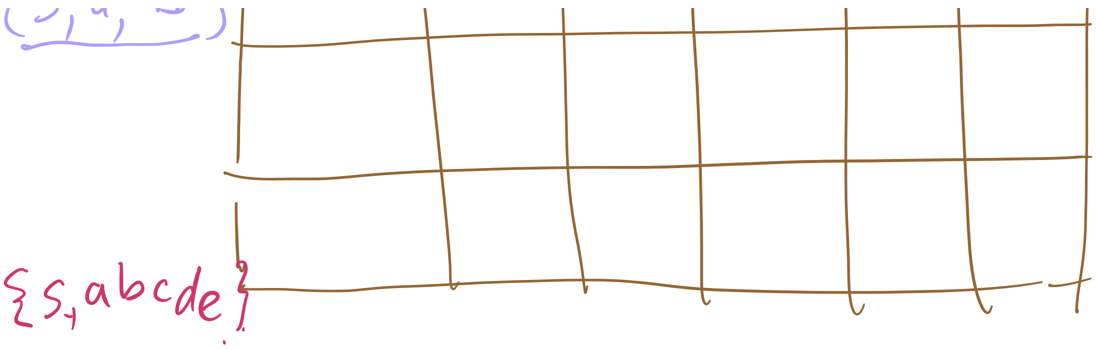
The shortest-path weight from u to v is denoted

$$\delta(u,v) = \begin{cases} \min w(p) : u \overset{p}{\rightsquigarrow} v & \text{if } \exists \text{ a path from } u \text{ to } v \\ \infty & : \text{otherwise} \end{cases}$$

A shortest u-v path is a u-v path that realizes the shortest-path weight.



d	s	a	b	c	d	e
$\emptyset$	0	∞	∞	∞	∞	∞
$\{s\}$		$\boxed{3}^{(s)}$	∞	∞	$8^{(s)}$	$\cancel{10}^{(s)}$
$\{s, a\}$			$6^{(a)}$	$9^{(a)}$	$8^{(s)}$	$7^{(a)}$
$\{s, a, b\}$						



$d_k[s,y]$ = min weight of a s-y-path that only visits, as intermediate vertices (vertices along the way) the k closest vertices to s.

Start by calculating $d_0[s,y]$ $\forall y$.

$d_1[s,y]$ = length of Shortest s-y path using {s }

$d_2[s,y]$ = shortest s-y path using {s, a }

$d_3[s,y]$ = Shortest s-y path using {s, a, b}

$d_4[s,y]$ = shortest s-y path using {s, a, b, e }

etc.

Dijkstra's Algorithm

- solves single-source shortest paths on weighted, directed or undirected graphs when edge weights are non-negative.

Dijkstra(G, w, s)

Initialize Single Source

set $S = \emptyset$

Put each $v \in V$ into a "special container" H.

While H is not empty

- extract vertex v from H that has minimum distance from s.
- relax all of v's neighbours that are still in H.

What kind of thing is H?

Operations:

init()

insert(v, K)

extractMin()

- removes and returns the (element, Key) pair with minimum Key value in H.

H is a Priority Queue ADT

- defined by the operations above.

How can H be implemented?