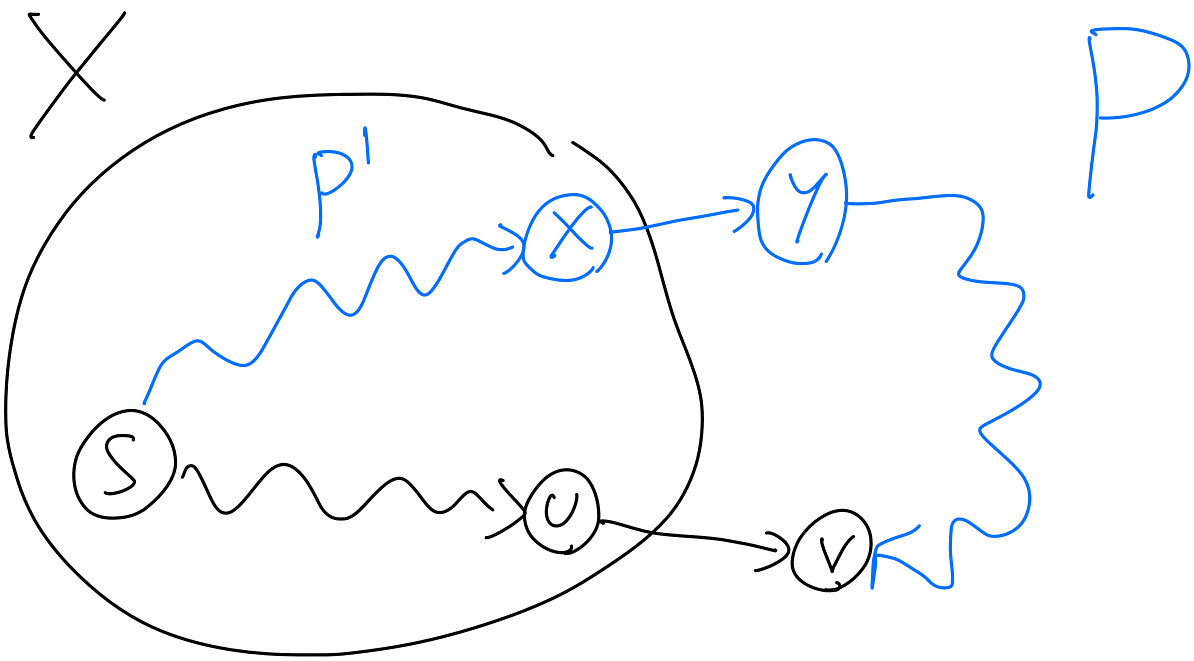
- Consider any path P from s to v
We need to show that P is not shorter than $\pi(v)$:



let (x,y) be the first arc in P that traverses X and let P' be the sub-path from s to x

$$len(P) \geq len(P') + w(x,y)$$

↳ w is non negative

$$\geq len(x) + w(x,y)$$

↳ ind. hypothesis

$\geq \pi(y)$

↳ def. of π

$\geq \pi(v)$

↳ Dijkstra selected v instead of y

Dijkstra with heaps

(almost identical to Prim's implementation with heaps)

Dijkstra(G, s)

$X = \emptyset$

H = empty heap

key$(s) = 0$

for each $v \neq s$ do

  key$(v) = +\infty$

for each $v \in V$ do

  insert v into H

while H is non-empty do

$w^* = \text{extractMin}(H)$

add w^* to X

$\text{len}(w^*) = \text{key}(w^*)$

// update heap

for every edge (w^*, y) s.t. $y \notin X$ do

delete y from H

$\text{key}(y) = \min\{\text{key}(y), \text{len}(w^*) + w(w^*, y)\}$

insert y into H

Complexity: $O((m+n)\log n)$

there are $O(m+n)$ operations on heaps

The (general) SSSP problem

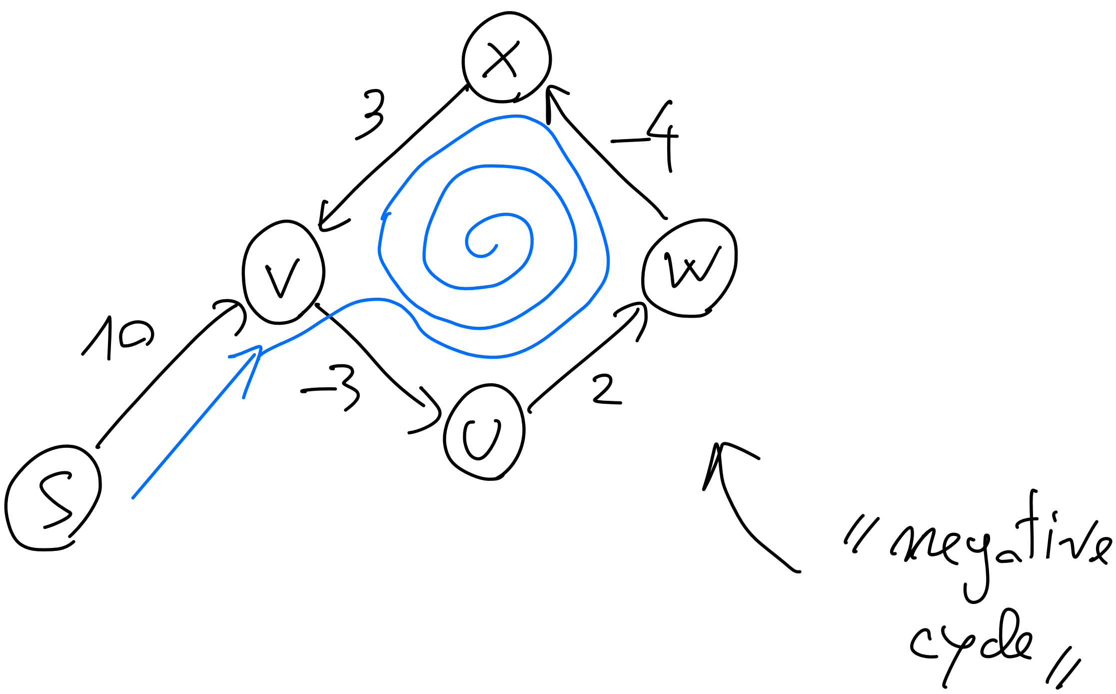
that is, graphs can have edge with negative weights

who cares about negative weights?

1) in road networks traversing one edge comes with a reward/bonus → weights represent more general costs than just distance

2) compute a profitable sequence of financial transactions

With negative weights we must be careful about what we even <u>mean</u> by "shortest paths":



$$\mathrm{dist}(s, v) = ?$$

there is no shortest s-v path! → dist(s,v) undefined

So, how about forbidding negative cycles?
(that is, compute shortest cycle-free/simple paths)

Problem now is well-defined, but is NP-hard → no polynomial-time algorithm (unless P=NP)

Then:

Single-Source Shortest Paths (revised version)

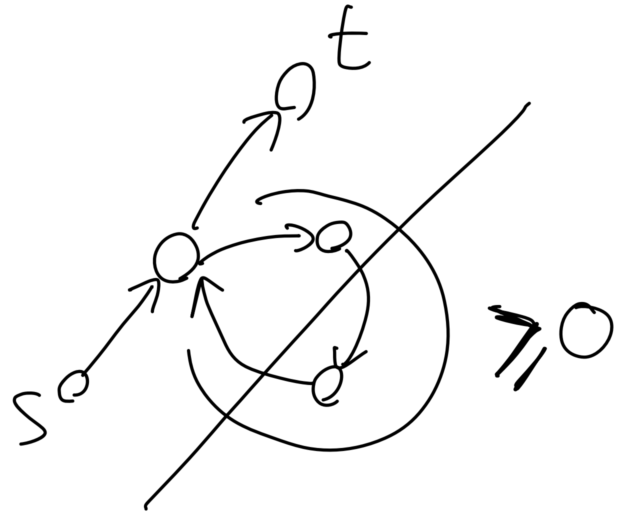
input: a directed, weighted graph $G=(V,E)$ and a source vertex $s \in V$

output: one of the following:

a) $dist(s,v)$ $\forall$ vertex $v \in V$

b) a declaration that G contains a negative cycle

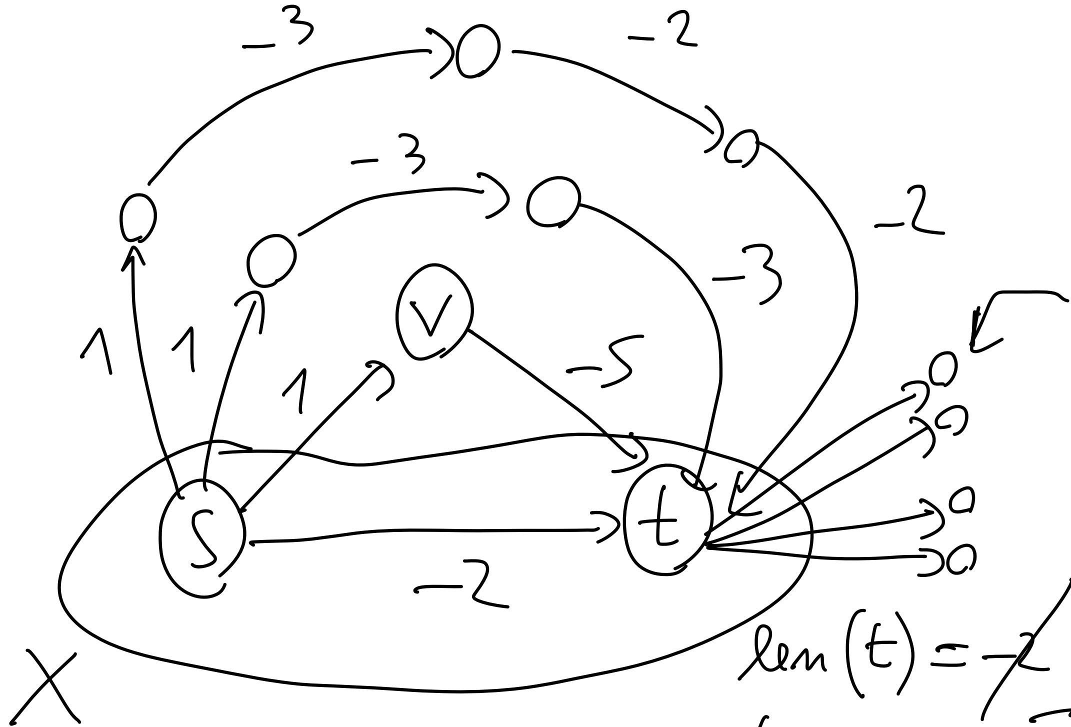
Observation: can a shortest path contain a cycle?
not negative-weight cycles, but not positive-weight either:



What about 0-weight cycles? We can remove all of them, and therefore wlog we can assume to compute cycle-free shortest paths, which have $\leq n-1$ edges

What needs to be changed in Dijkstra's alg. to deal with negative-weights edges?

intuition:



len(t) = ~~-2~~ ~~-4~~ ~~-8~~ -6

should be just an estimated distance

problem: Dijkstra's alg. never revisits/updates its decisions, but it should!
for all vertices!
how many times? $\leq n-1$ edges
$\Rightarrow$ $n-1$ times should be enough

Bellman-Ford (G, s) (1955)

input: directed graph G with edge weights $w: E \to \mathbb{R}$
and a source vertex $s \in V$

output: either $\text{dist}(s,v)$ $\forall v \in V$ or a declaration
that G contains a negative cycle

$\left.\begin{array}{l} \text{len}(s) = 0 \\ \text{len}(v) = +\infty \quad \forall v \neq s \end{array}\right\}$ initial estimated distances

for $n-1$ iterations do
    for each edge $(u,v) \in E$ do
        \\ update the estimated distance (a.k.a. "relax" edge (u,v))
        $\text{len}(v) = \min\{\text{len}(v), \text{len}(u) + w(u,v)\}$
for each edge $(u,v) \in E$ do
    if $\text{len}(v) > \text{len}(u) + w(u,v)$ then
        \\ some distance changed in the n-th iteration
        return "G contains a negative cycle"

Complexity: $O(m \cdot n)$

Example :