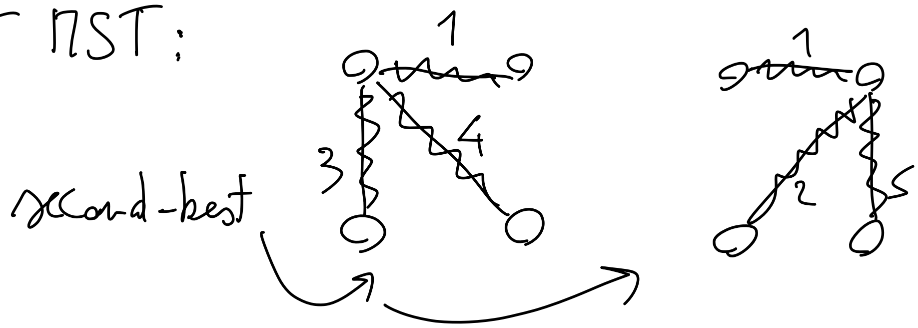
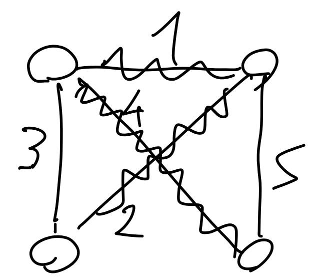
Second-best MST:





MST

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 every $v \in V$ do

  insert v into H

while H is non-empty do

  $w^* =$ extractMin(H)

  add w^* to X

$len(w^*) = key(w^*)$
// update heap
for every edge (w^*, y) s.t. $y \notin X$ do
    delete y from H
    $key(y) = \min\{key(y), len(w^*) + w(w^*, y)\}$
    insert y into H

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

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

Exercise: consider a directed graph with nonnegative weights. Under what conditions $\exists$ a unique shortest path from $s \in V$ to $t \in V$?

a) when all weights are distinct positive integers
b) when all weights are distinct powers of 2
c) when a) and the graph is cycle-free

The (general) SSSP problem

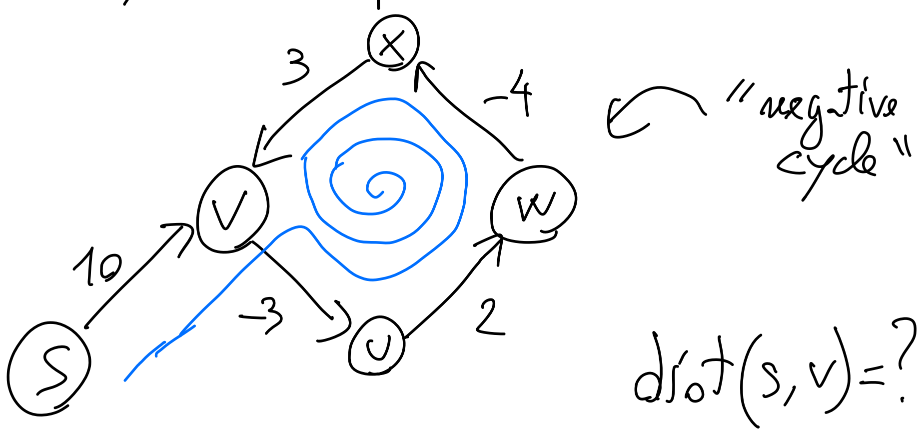
that is, graphs can have edges with negative weights

who cares about negative weights?

1) in road networks traversing one edge comes with a reward/bonus $\longrightarrow$ weights represent a more general cost 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"



$dist(s,v) = ?$

there is no shortest s-v path! $\Longrightarrow$ $dist(s,v)$ is undefined (or, $-\infty$)

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

Problem now is well-defined, but is <u>NP-hard</u>

$\longrightarrow$ no polynomial algorithm (unless $P=NP$)

Then:

Single-Source Shortest Paths (revised version)

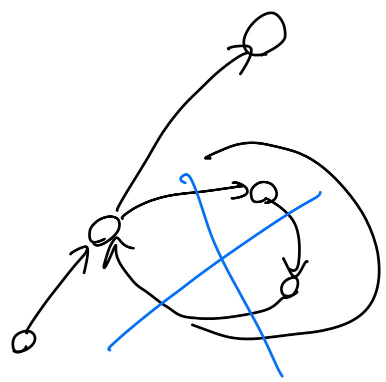
input: a directed, weighted graph $G = (V, E)$ and a source vertex $s \in V$

output: one of the following:

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

b) a declaration that G contains a negative cycle

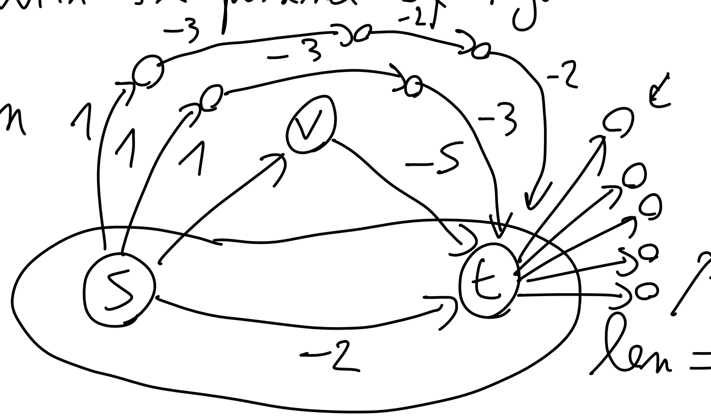
Observation: Can a shortest path contain a cycle? not negative-weight cycles, but not positive-weight either:



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 algorithm to deal with the presence of negative-weight edges?

intution



should be an estimated distance

len = ~~-2~~ ~~-4~~ ~~-5~~ -6

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

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 $dist(s,v)$ $\forall v \in V$ or a declaration that G contains a negative cycle

$len(s) = 0$
$len(v) = +\infty \quad \forall v \neq s$) initial estimated distances

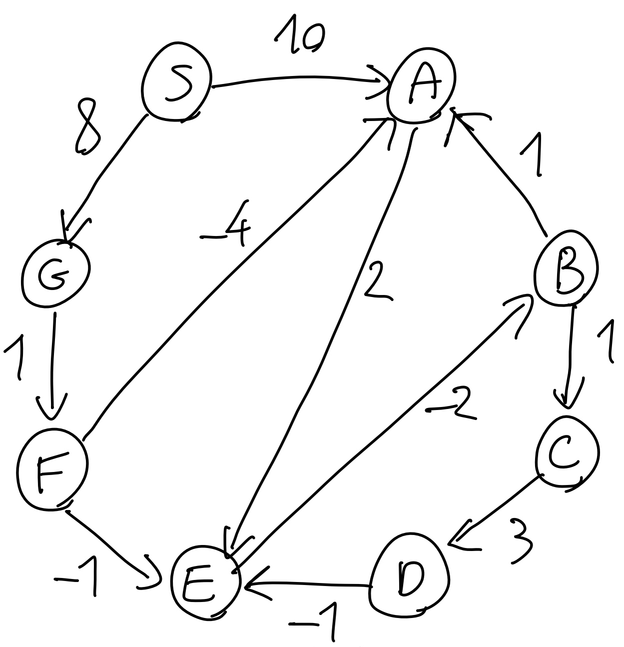
for $n-1$ iterations do
  for each edge $(u,v) \in E$ do
    $len(v) = \min\{len(v), len(u) + w(u,v)\}$

updates the distance estimate → "relax" the edge (u,v)

```
for each edge (u,v) ∈ E do
    if len(v) > 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:



iterations

Vertices

	0	1	2	3	4	5	6	7	last
S	0	0	0	0	0	0	0	0	0
A	∞	10	10	5	5	5	5	5	5
B	∞	∞	∞	10	6	5	5	5	5
C	∞	∞	∞	∞	11	7	6	6	6
D	∞	∞	∞	∞	∞	14	10	9	9
E	∞	∞	12	8	7	7	7	7	7
F	∞	∞	9	9	9	9	9	9	9
G	∞	8	8	8	8	8	8	8	8

Comments:

- it's more "distributed" than Dijkstra $\Rightarrow$ has played a prominent role in the evolution of the Internet routing protocols

- Has been the fastest alg. for SSSP until 2022, when a near-linear algorithm was published