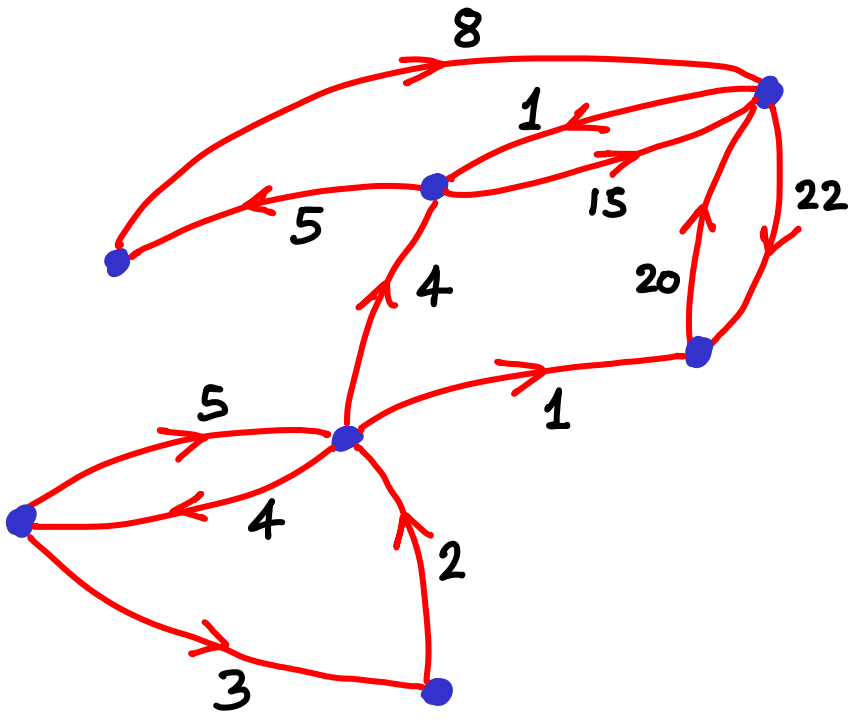
SINGLE SOURCE SHORTEST PATHS

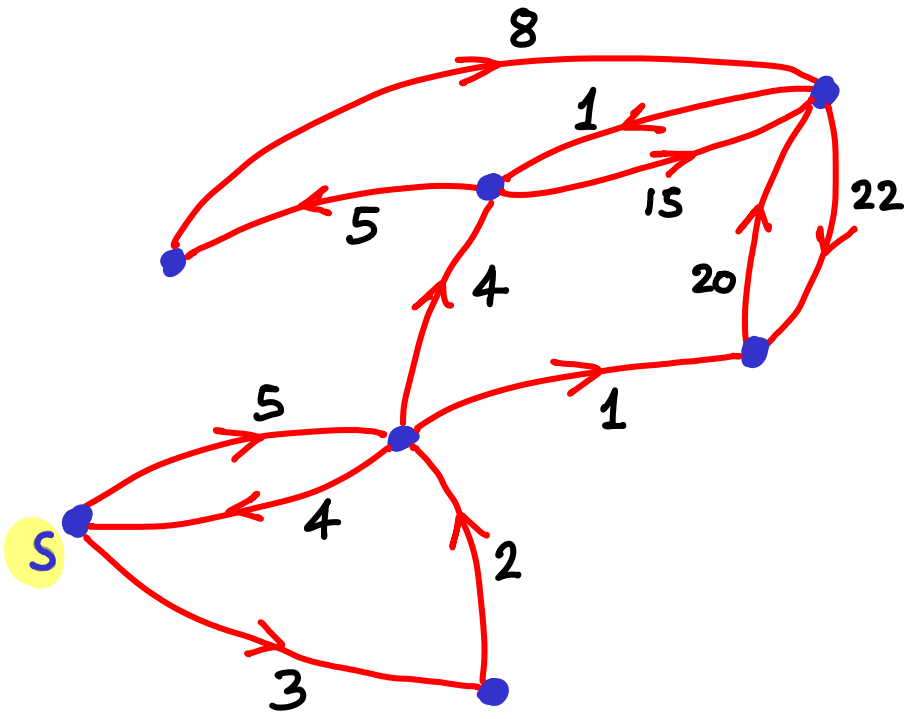
SINGLE SOURCE SHORTEST PATHS



Generally assume a directed graph
(can make undirected→directed easily)

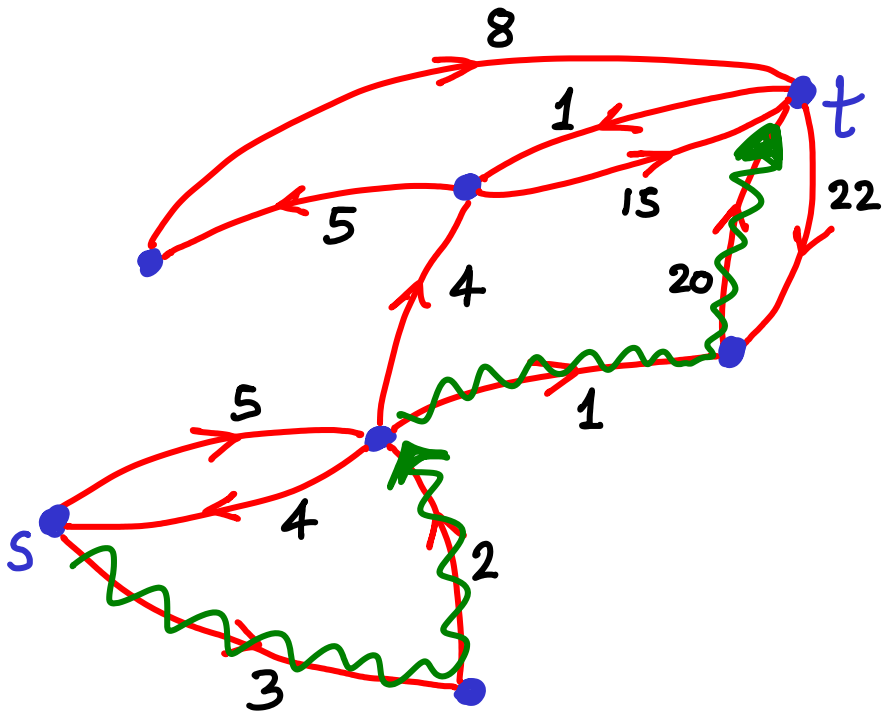
SINGLE SOURCE SHORTEST PATHS

s



SINGLE SOURCE SHORTEST PATHS

s

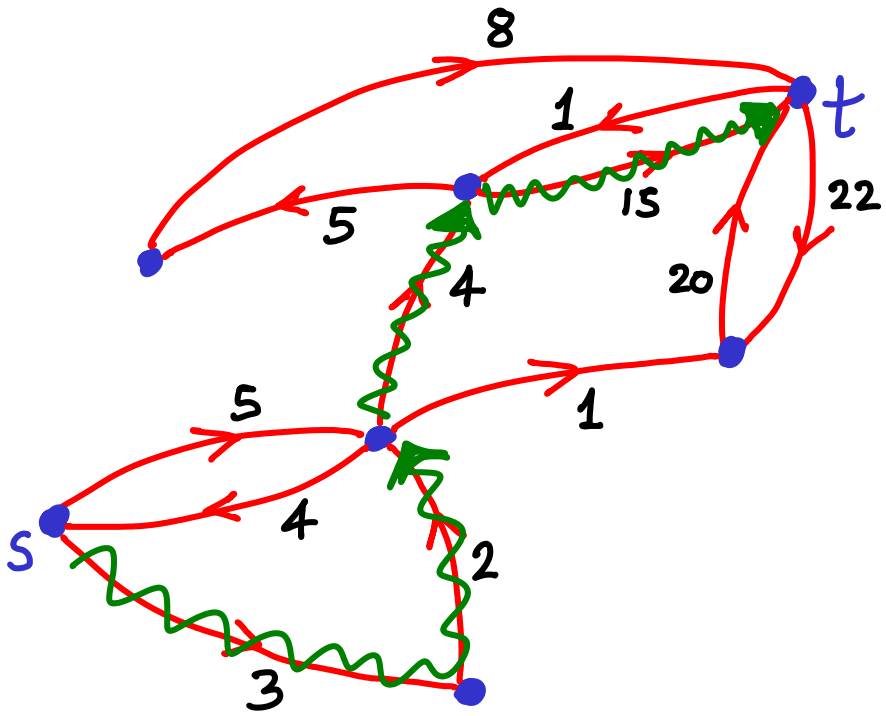


paths from s to t

3 + 2 + 1 + 20

SINGLE SOURCE SHORTEST PATHS

s



paths from s to t

3 + 2 + 1 + 20

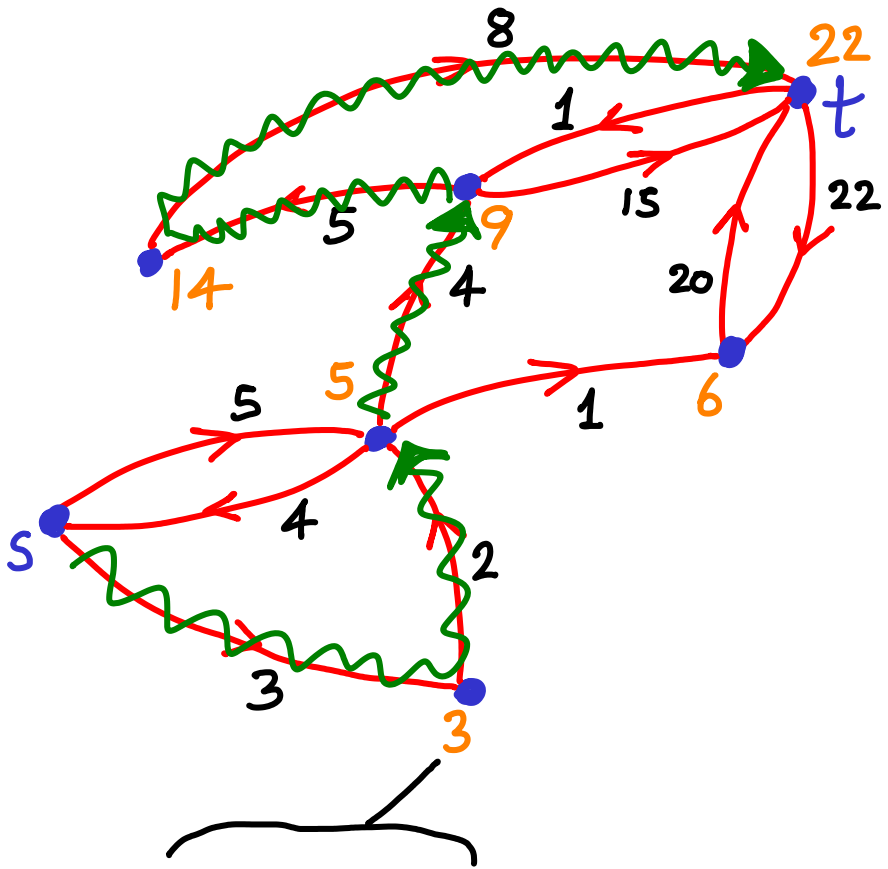
3 + 2 + 4 + 15

not greedy or BFS

although this is an extension of BFS (weights = 1)

SINGLE SOURCE SHORTEST PATHS

s



tot. length of path from s

paths from s to t

not greedy or BFS

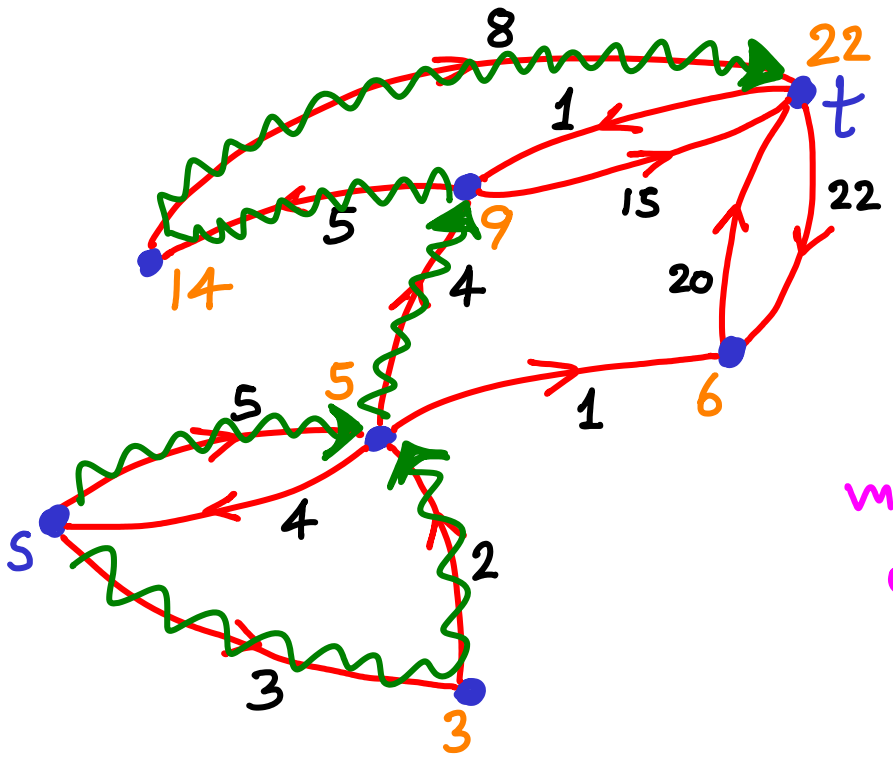
3 + 2 + 1 + 20

3 + 2 + 4 + 15

3 + 2 + 4 + 5 + 8 = 22

SINGLE SOURCE SHORTEST PATHS

s



paths from s to t

not greedy or BFS

3 + 2 + 1 + 20

3 + 2 + 4 + 15

3 + 2 + 4 + 5 + 8 = 22

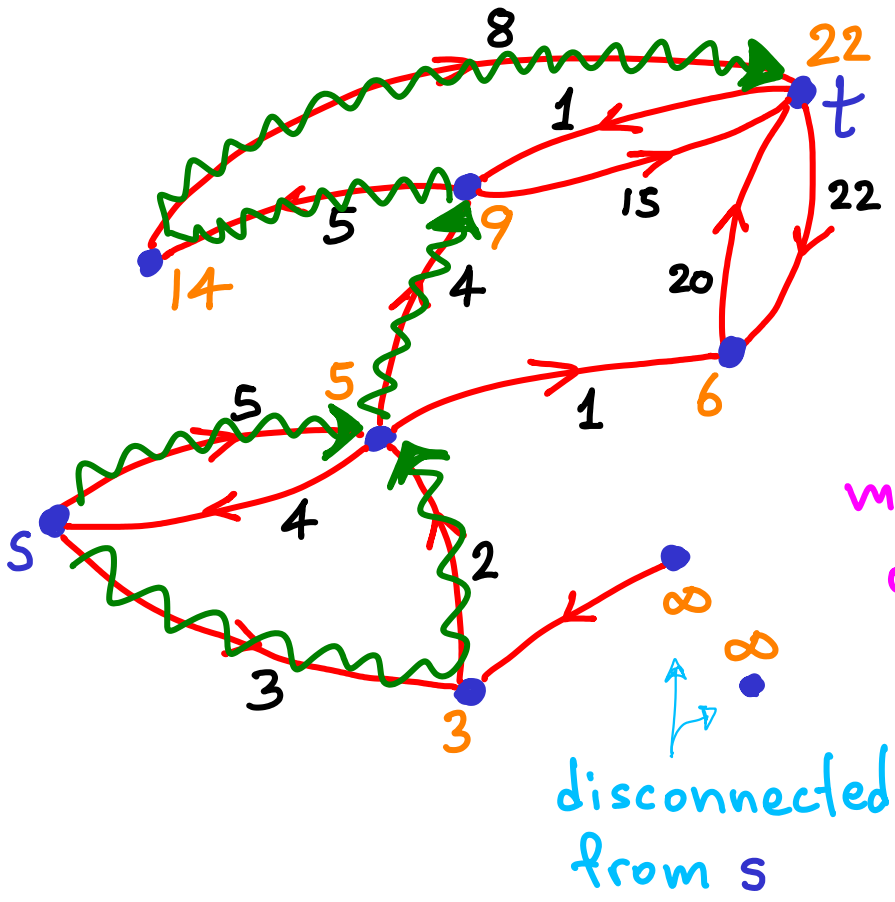
multiple options

5 + 1 + ⋯

5 + 4 + ⋯ = 22

SINGLE SOURCE SHORTEST PATHS

s



paths from s to t

not greedy or BFS

3 + 2 + 1 + 20

3 + 2 + 4 + 15

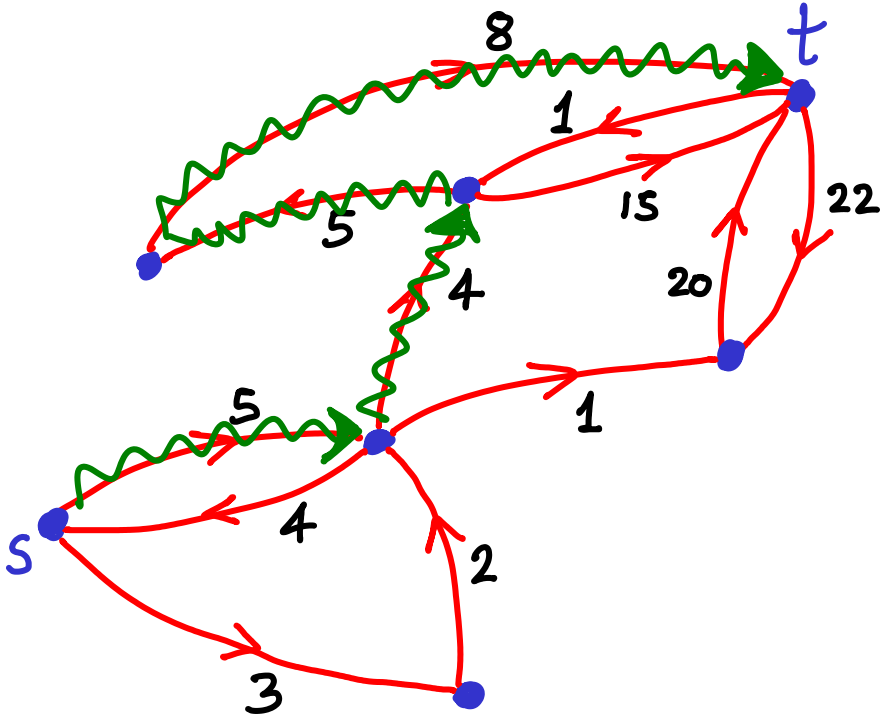
3 + 2 + 4 + 5 + 8 = 22

multiple options { 5 + 1 + ⋯ = 22

5 + 4 + ⋯

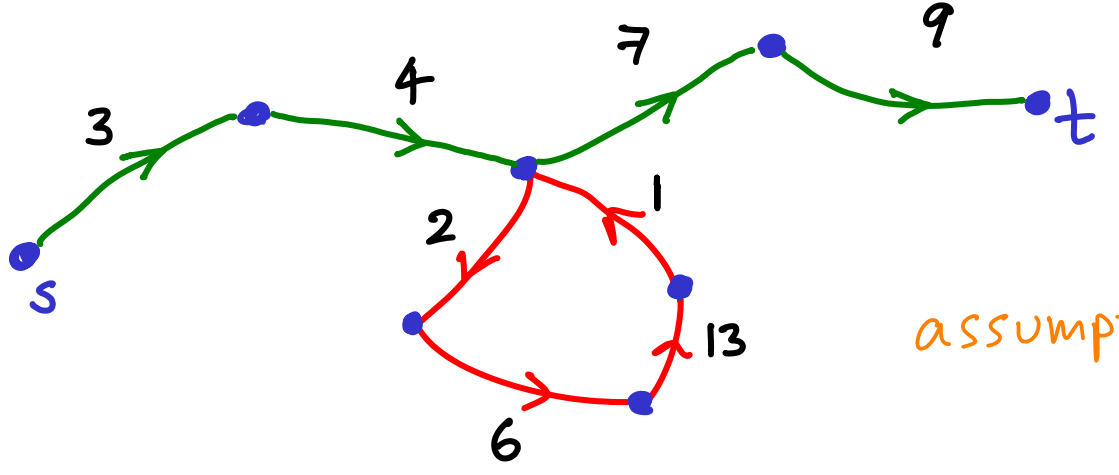
SINGLE SOURCE SHORTEST PATHS

s



Observations

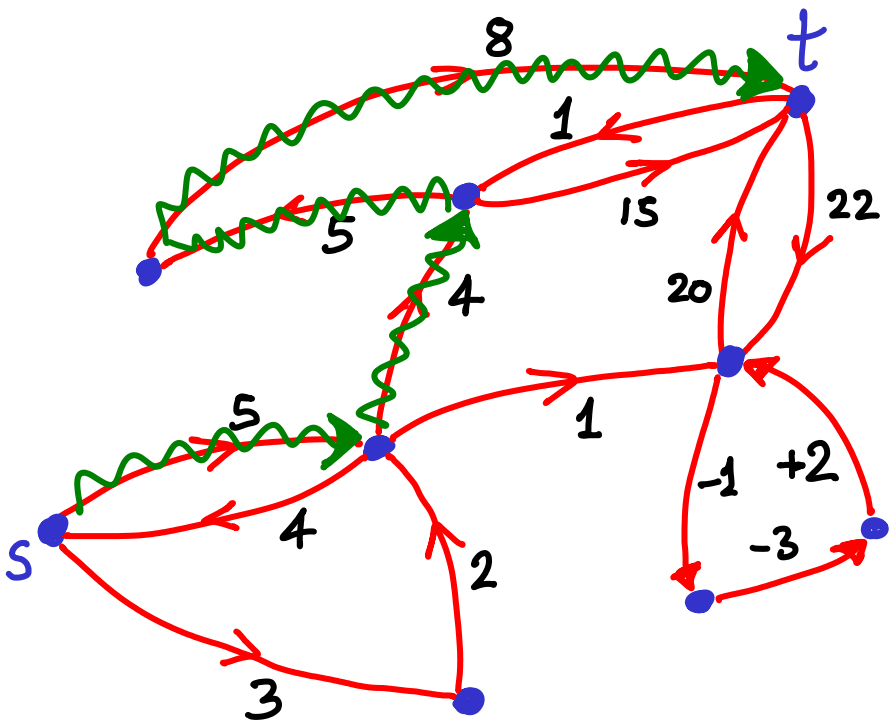
- No cycles in $s \rightarrow t$ (shortest path)



assumption?

SINGLE SOURCE SHORTEST PATHS

s



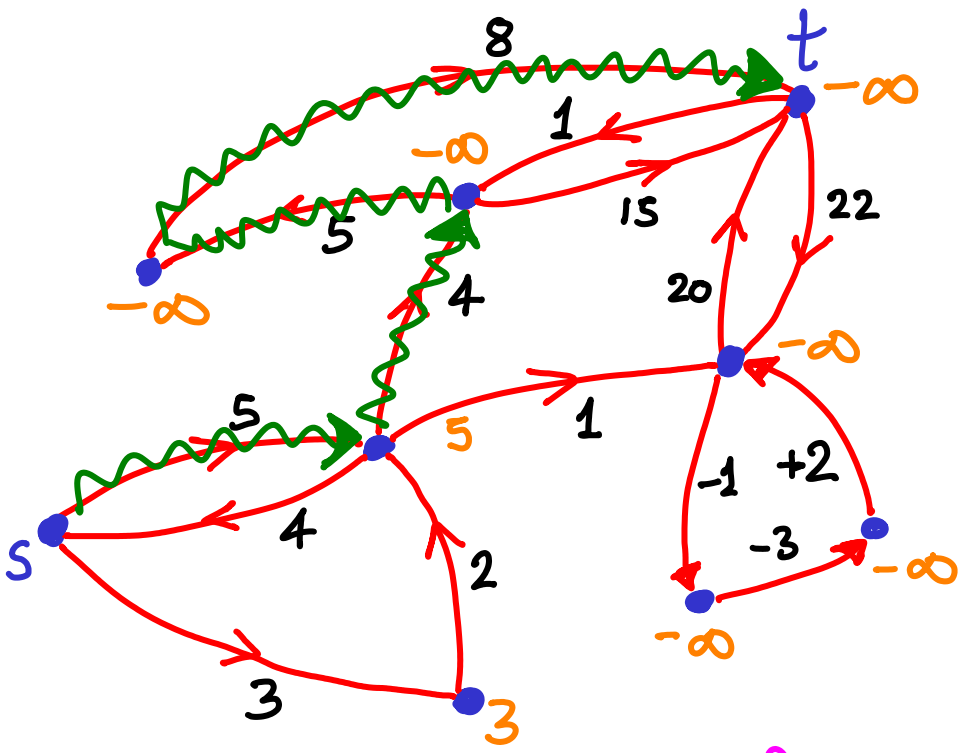
Observations

- No cycles in $s \rightsquigarrow t$
- Negative weights ~OK, unless they form a negative cycle in G

$\sum(\text{cycle}) < 0$

SINGLE SOURCE SHORTEST PATHS

s



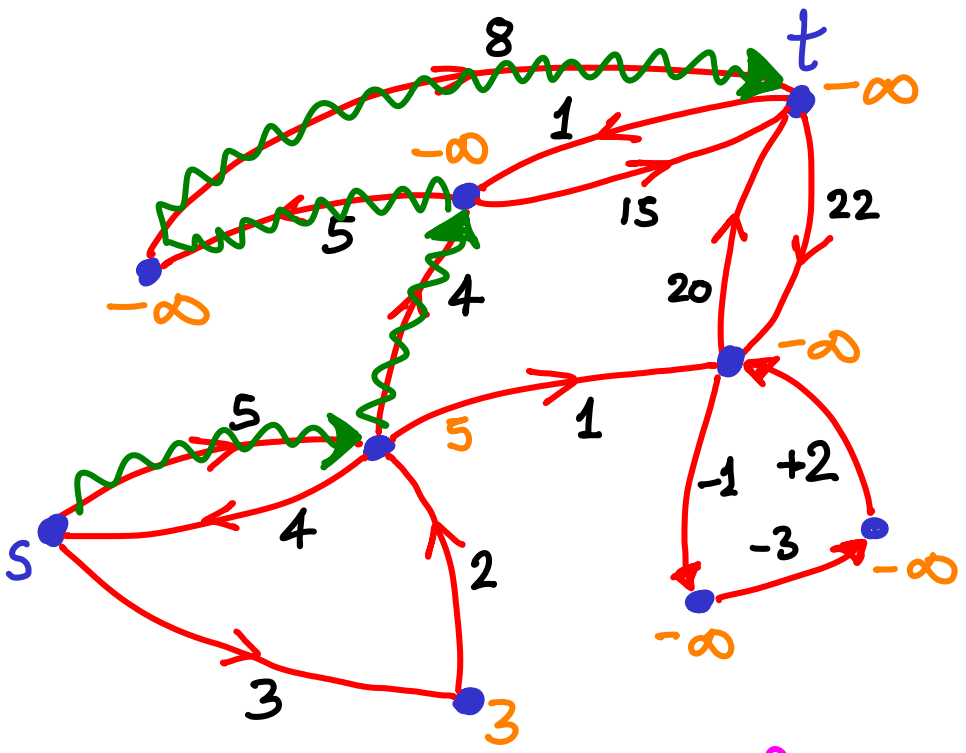
Observations

- No cycles in $s \rightsquigarrow t$
- Negative weights ~OK, unless they form a negative cycle in G

Any vertex reachable from a negative cycle gets a score of $-\infty$ } assuming cycle can be reached from s

SINGLE SOURCE SHORTEST PATHS

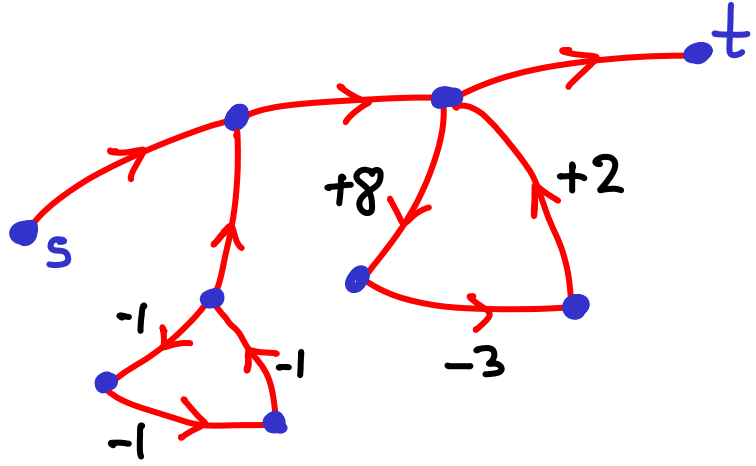
s



Any vertex reachable from a negative cycle gets a score of $-\infty$

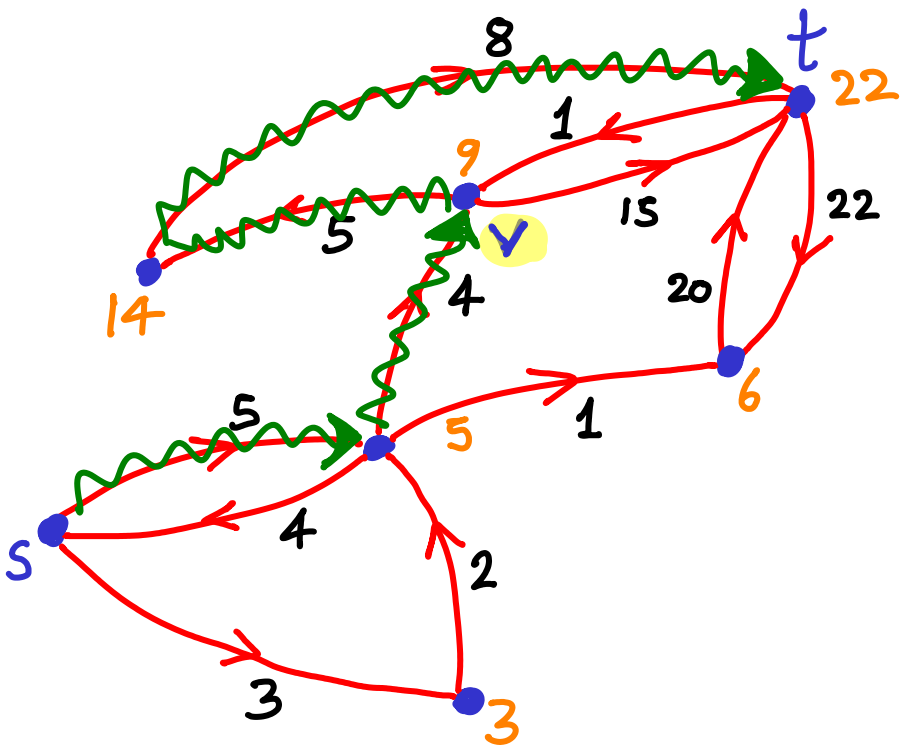
Observations

- No cycles in $s \rightsquigarrow t$
- Negative weights ~OK, unless they form a negative cycle in G



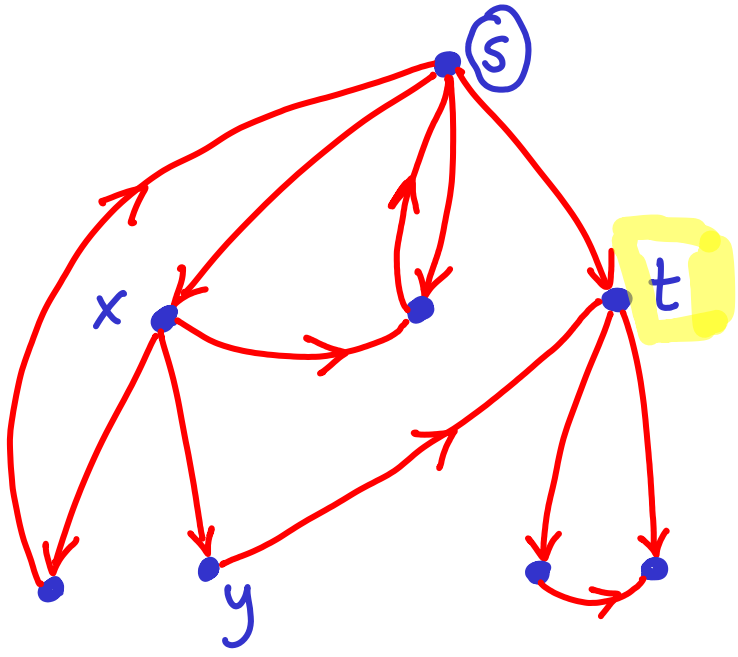
SINGLE SOURCE SHORTEST PATHS

s



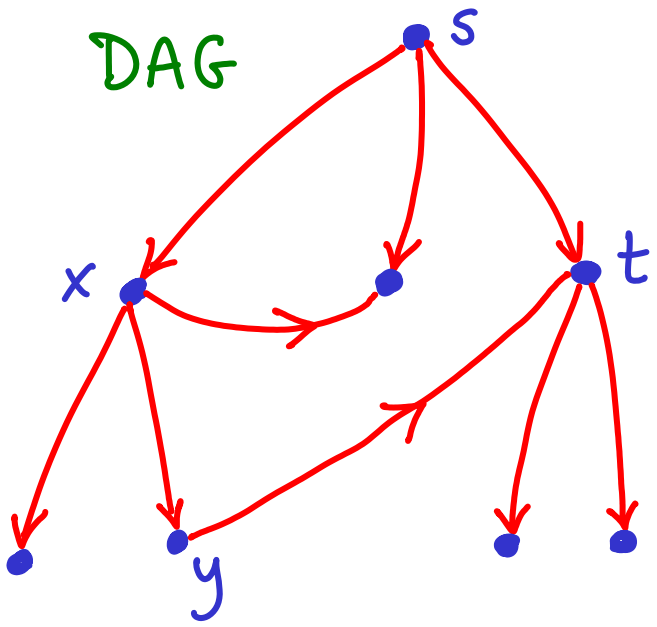
Observations

- No cycles in $s \rightsquigarrow t$
- Negative weights ~OK, unless they form a negative cycle in G
- shortest path $s \rightsquigarrow v \rightsquigarrow t$ contains
  - shortest path $s \rightsquigarrow v$ (9)
  - &
  - shortest path $v \rightsquigarrow t$ (13)



there may be multiple shortest paths

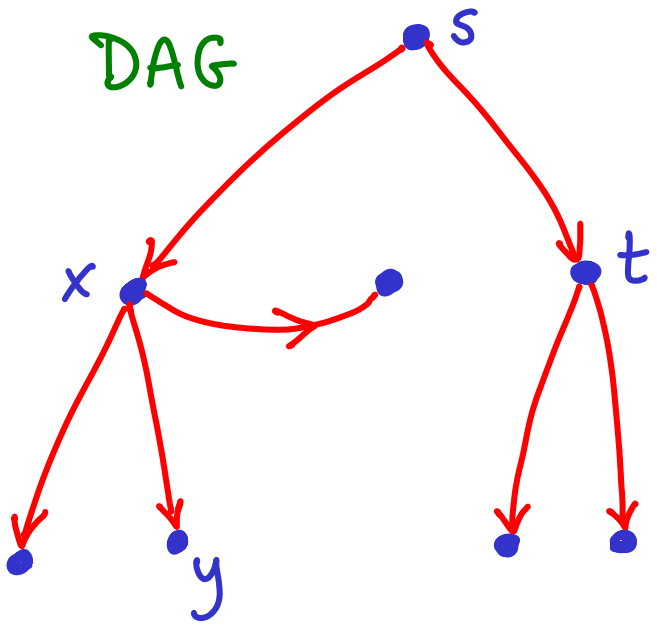
e.g. $s \to t$ or $s \to x \to y \to t$



there may be multiple shortest paths

e.g. $s \to t$ or $s \to x \to y \to t$

All shortest paths from s to V
can be represented in a DAG



there may be multiple shortest paths

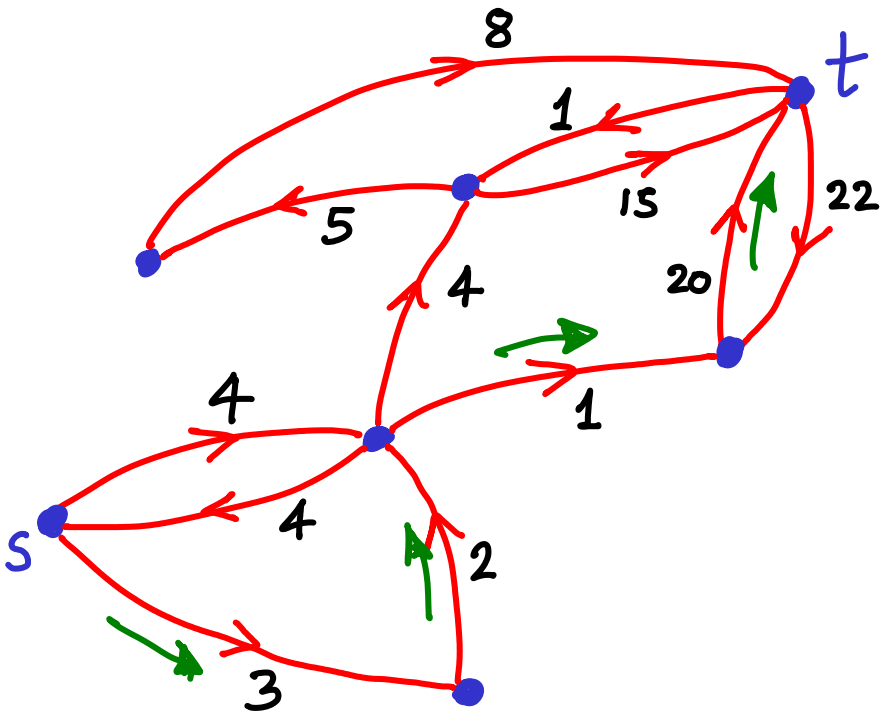
e.g. $s \to t$ or $s \to x \to y \to t$

All shortest paths from s to V can be represented in a DAG

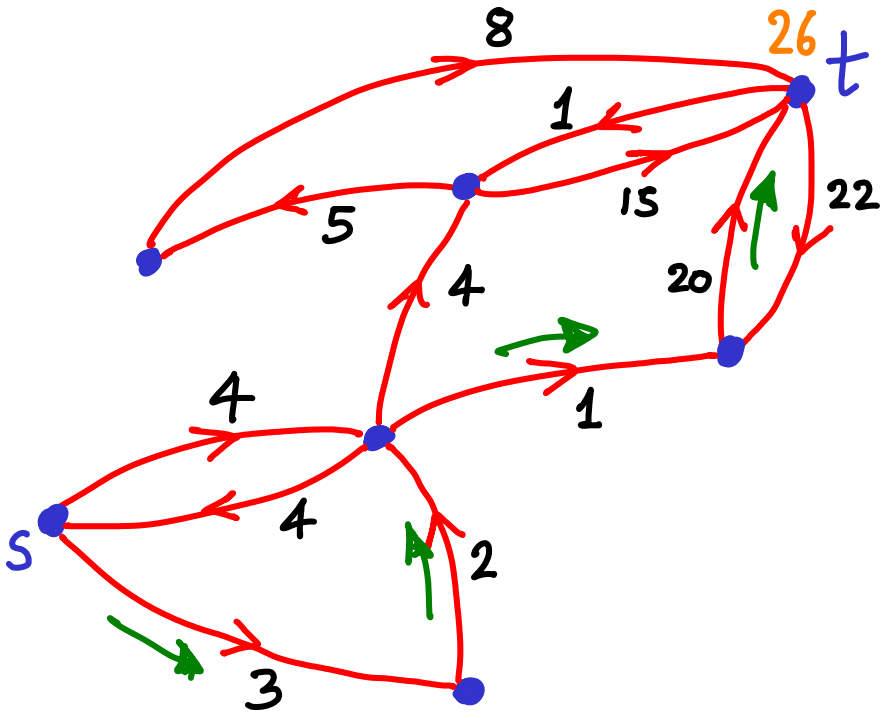
DAG → tree : arbitrarily keep one path to each vertex

"shortest paths tree"

(similar to picking one BFS/DFS search)

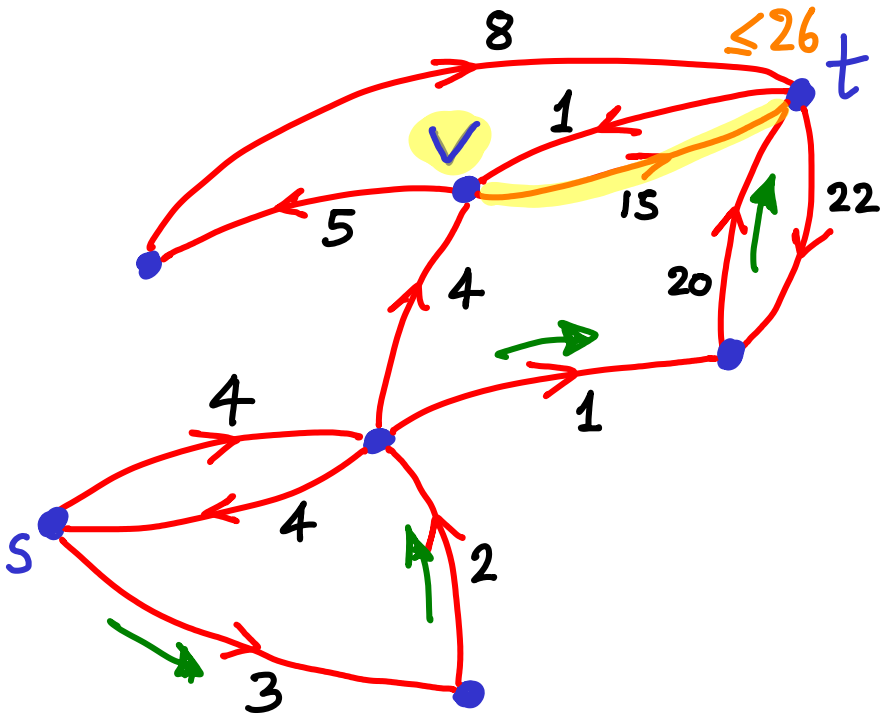


By exploring some path from s to t we get a path score (e.g. 26)



By exploring some path from s to t we get a path score (e.g. 26)

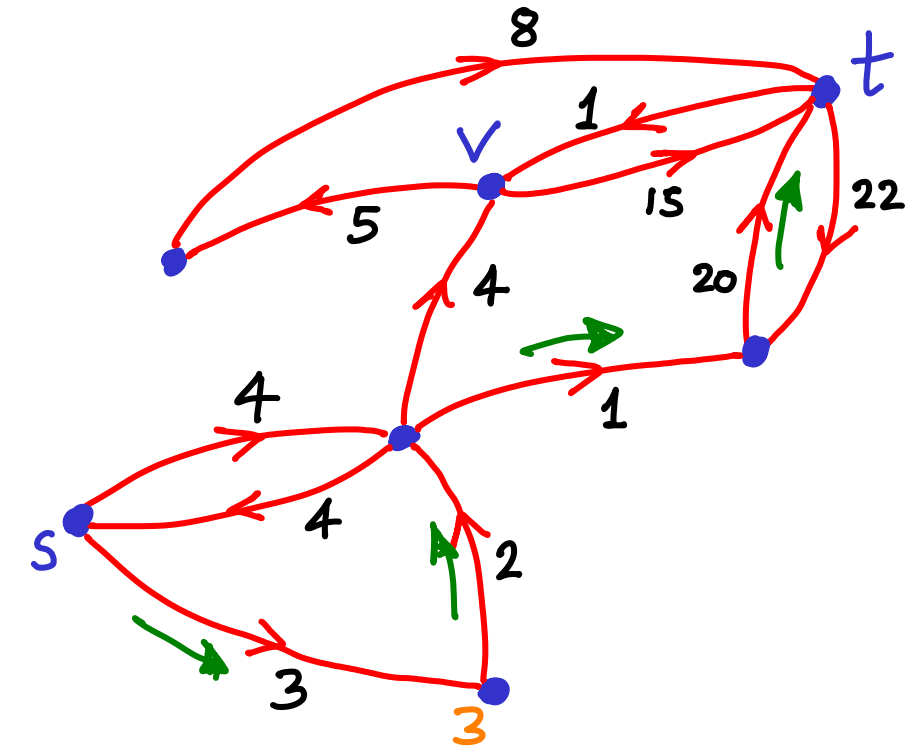
the score of t is 26, which may only decrease as we explore more options.



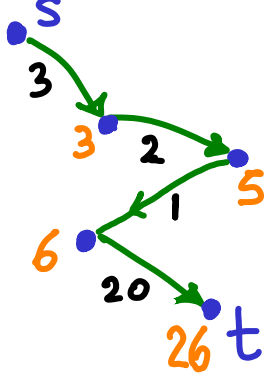
By exploring some path from s to t we get a path score (e.g. 26)

the score of t is 26, which may only decrease as we explore more options.

If we update the score of v: $d(v)$
& $\exists$ edge $v \to t$
then we can possibly improve $d(t)$:
$d(v) + w(v,t) < d(t)$?



shortest paths tree

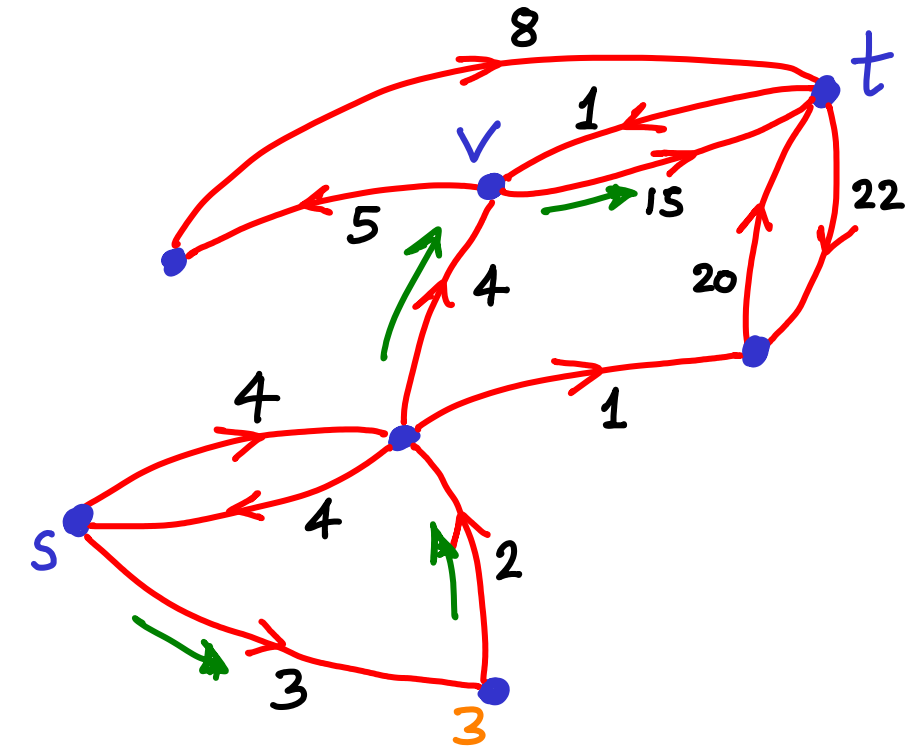


change tree

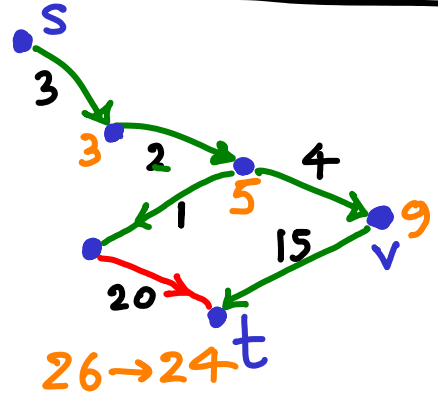
By exploring some path from s to t we get a path score (e.g. 26)

the score of t is 26, which may only decrease as we explore more options.

If we update the score of v: d(v)
& ∃ edge v→t
then we can possibly improve d(t):
$d(v) + w(v,t) < d(t)$?



shortest paths tree

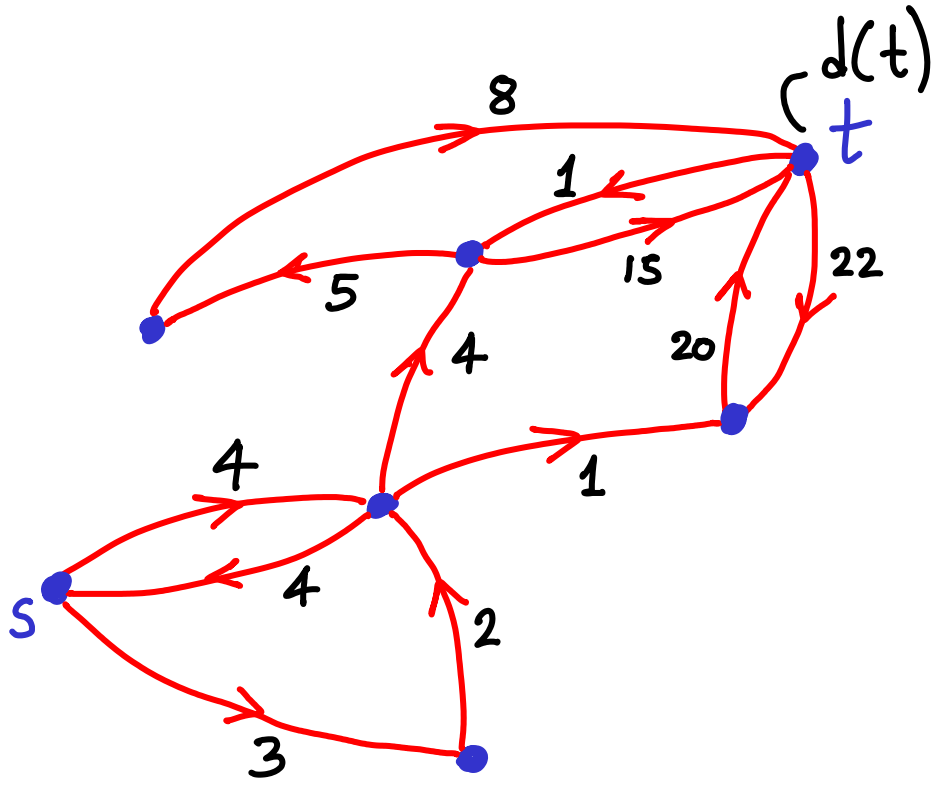


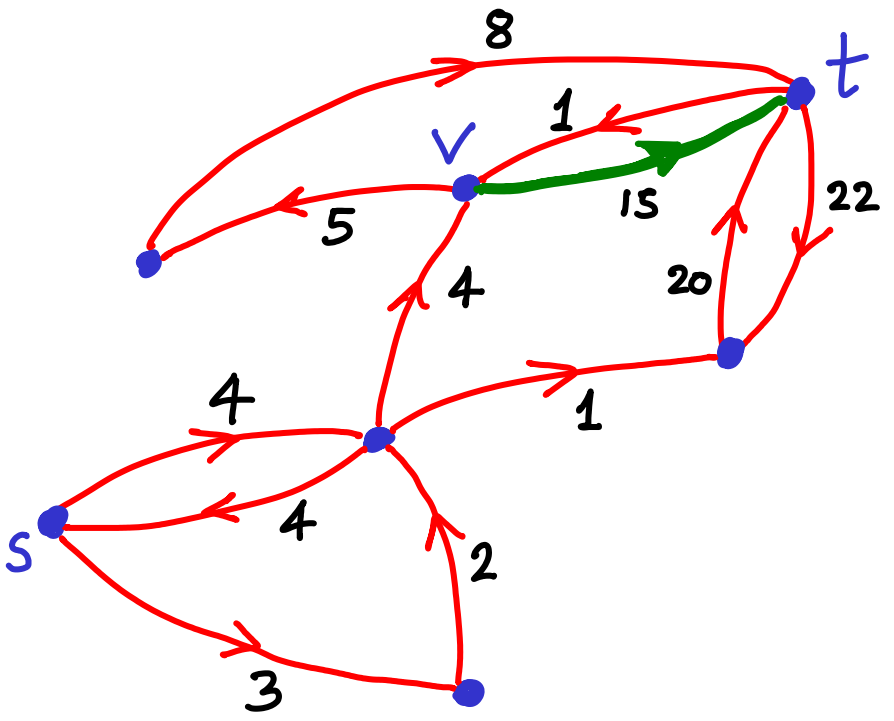
change tree

By exploring some path from s to t we get a path score (e.g. 26)

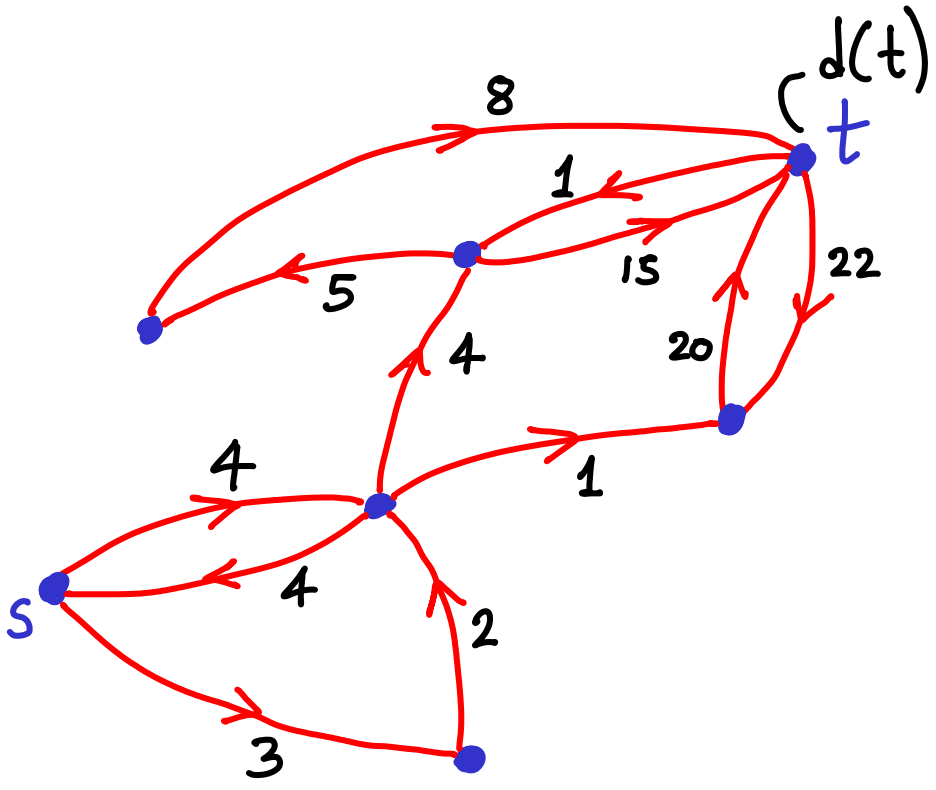
the score of t is 26, which may only decrease as we explore more options.

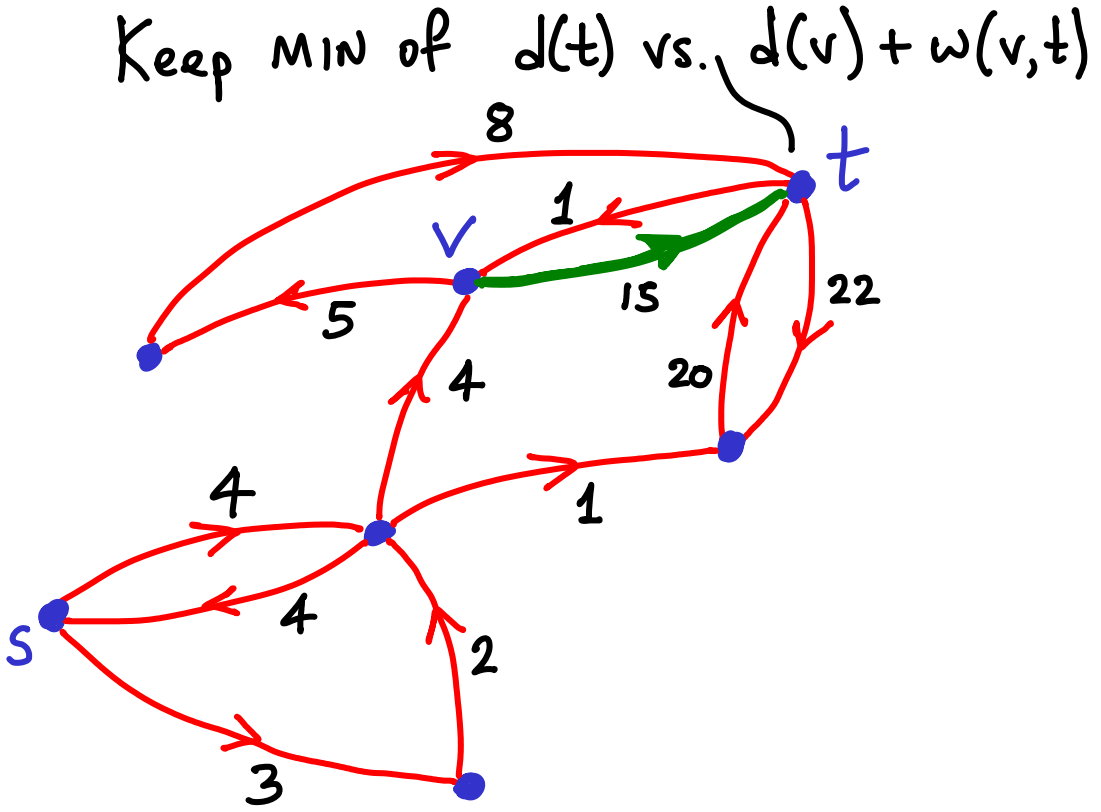
If we update the score of v: d(v)
& ∃ edge v→t
then we can possibly improve d(t):
$d(v)+w(v,t) < d(t)$?



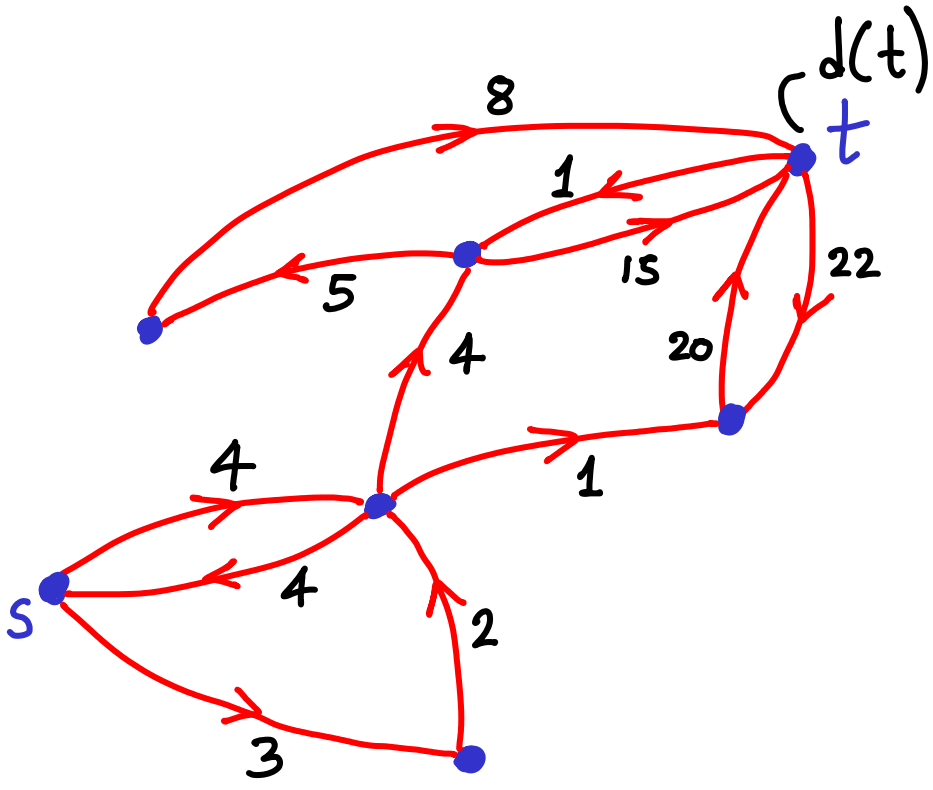


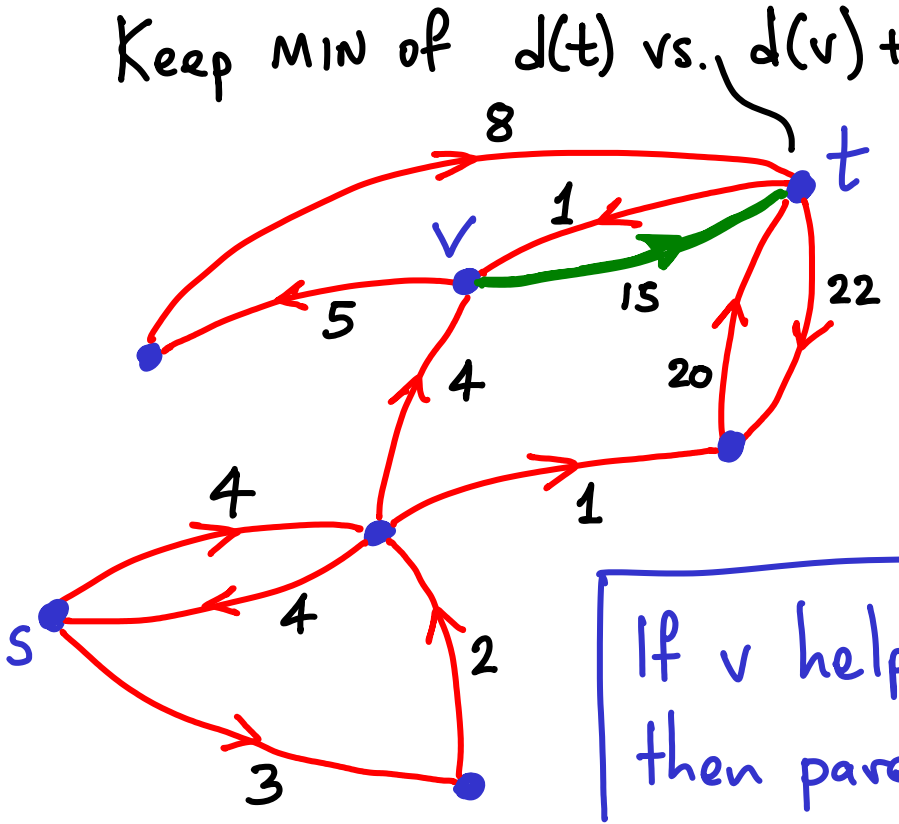
Relax(v,t) : checking if score of t can be improved (lowered) by using $s \to v \to t$





Relax(v,t) : checking if score of t can be improved (lowered) by using s → v → t



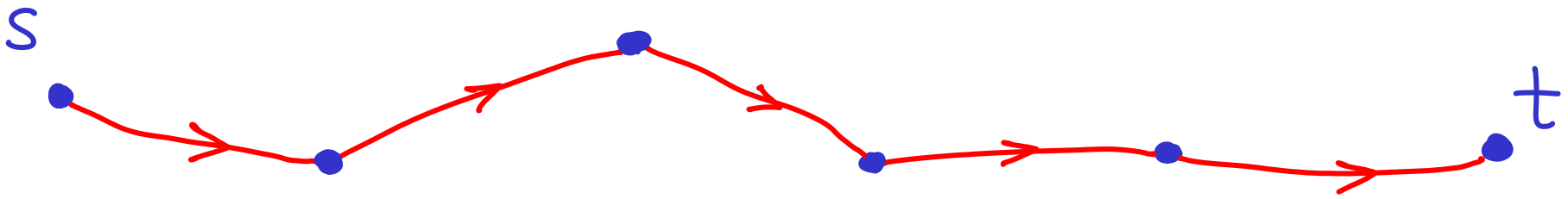


If v helps,
then parent(t) = v

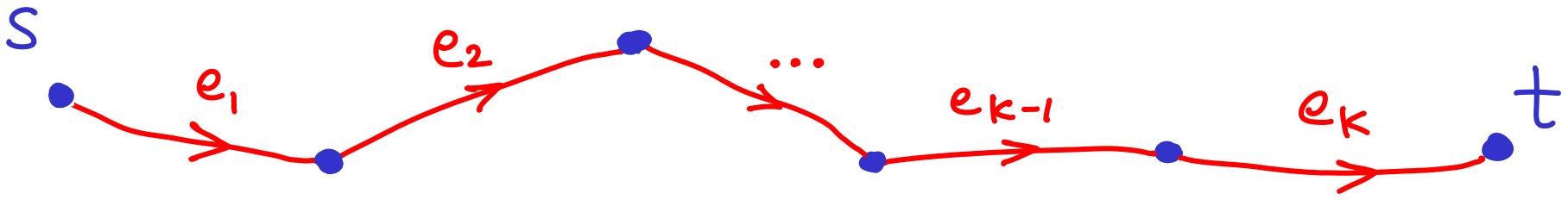
Relax(v,t) : checking if score of t can be improved (lowered) by using s → v → t

Assume this is a shortest path from s to t

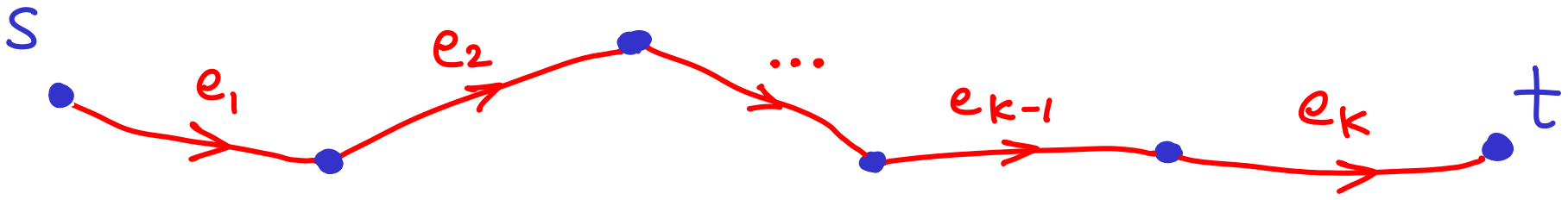
unknown but exists



Assume this is a shortest path from s to t unknown but exists

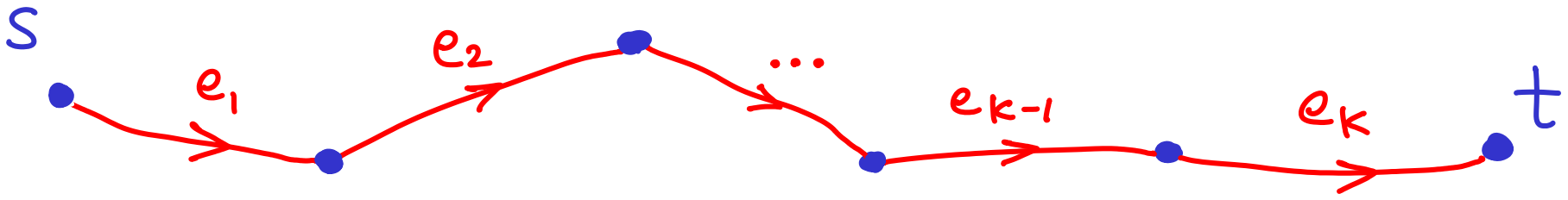


Assume this is a shortest path from s to t unknown but exists



Suppose we have an algorithm based on relaxing edges.

Assume this is a shortest path from s to t unknown but exists

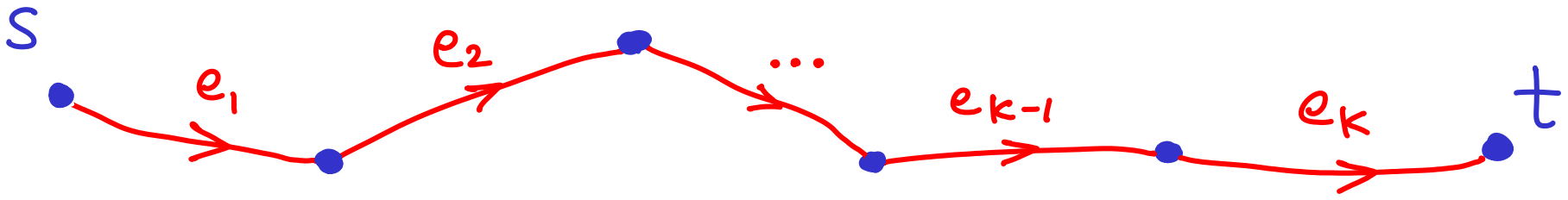


Suppose we have an algorithm based on relaxing edges.

If we relax e_1 before e_2 before ... before e_{k-1} before e_k

then ... ?

Assume this is a shortest path from s to t unknown but exists



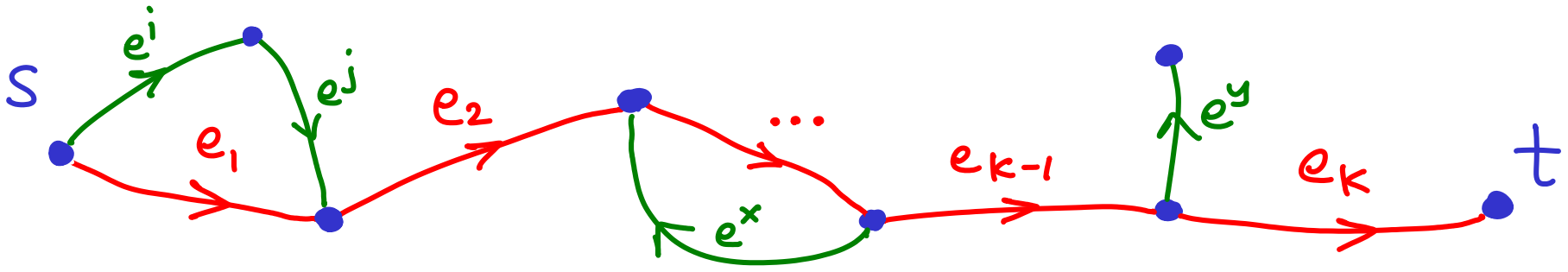
Suppose we have an algorithm based on relaxing edges.

If we relax e_1 before e_2 before ... before e_{k-1} before e_k

then we will correctly compute $d(t)$

Assume this is a shortest path from s to t

unknown but exists



Suppose we have an algorithm based on relaxing edges.

If we relax e_1 before e_2 before ... before e_{k-1} before e_k

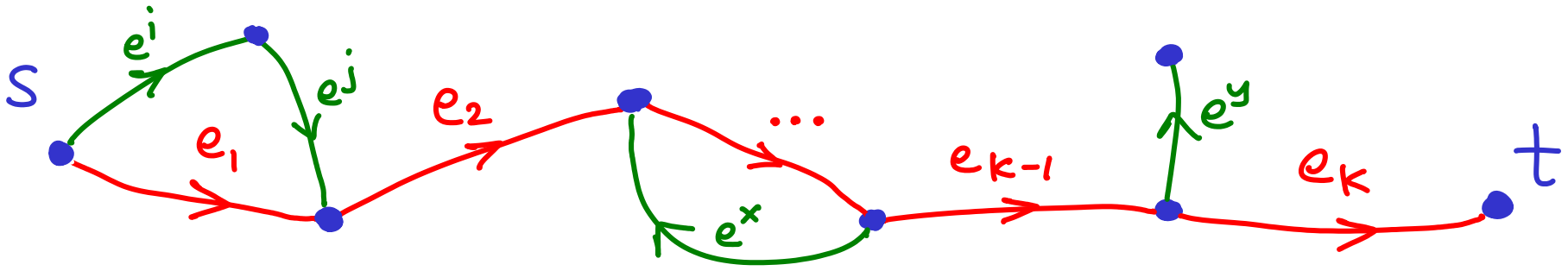
then we will correctly compute $d(t)$

arbitrary

Relax sequence : $e^x\ e_1\ e^j\ e^y\ e_2\ e^x e^i\ e_k\ e_{k-1}\ e_1 e^x\ e_k\ e^y$

Assume this is a shortest path from s to t

unknown but exists



Suppose we have an algorithm based on relaxing edges.

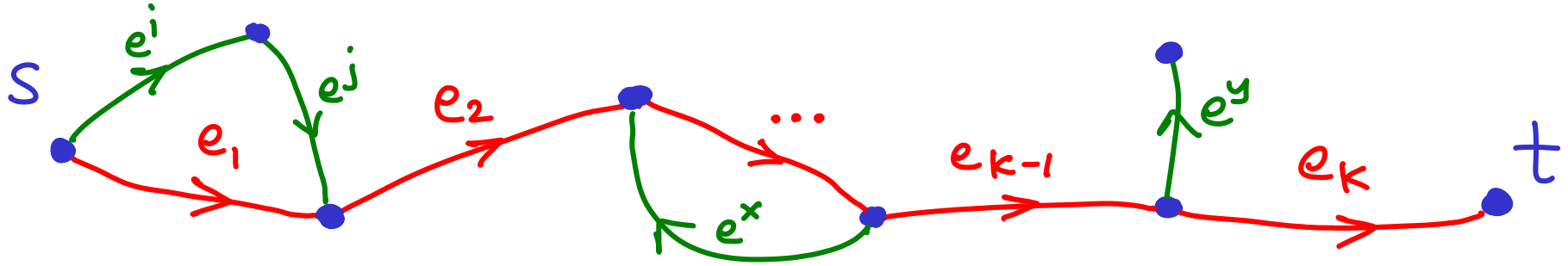
If we relax e_1 before e_2 before ... before e_{k-1} before e_k

then we will correctly compute d(t)

Relax sequence : $e^x\ e_1\ e^j\ e^y\ e_2\ e^x e^i\ e_k\ e_{k-1}\ e_1 e^x\ e_k\ e^y$: OK

(don't care if we relax other edges or the same ones repeatedly)

Assume this is a shortest path from s to t unknown but exists



Suppose we have an algorithm based on relaxing edges.

If we relax e_1 before e_2 before ... before e_{k-1} before e_k

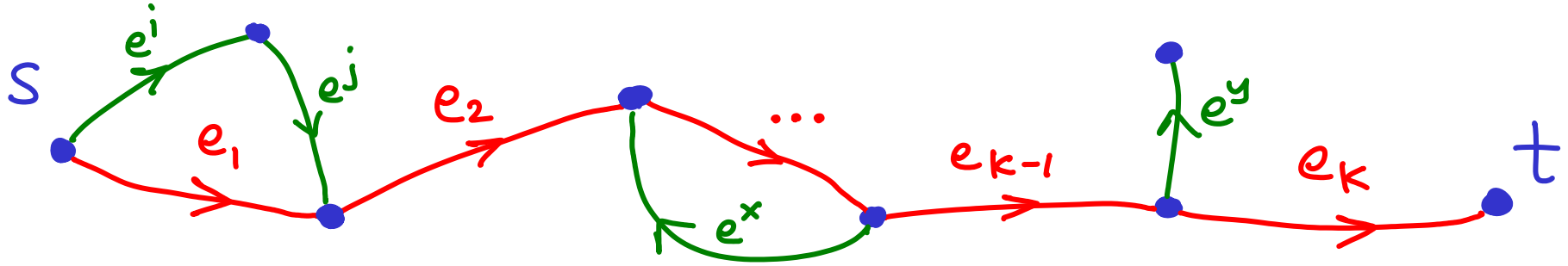
then we will correctly compute d(t) WHY?

Relax sequence : $e^x\ e_1\ e^j\ e^y\ e_2\ e^x e^i\ e_k\ e_{k-1}\ e_1 e^x\ e_k\ e^y$: OK

(don't care if we relax other edges or the same ones repeatedly)

Assume this is a shortest path from s to t

unknown but exists



Suppose we have an algorithm based on relaxing edges.

If we relax e_1 before e_2 before ... before e_{k-1} before e_k

then we will correctly compute d(t) [by INDUCTION]

Relax sequence : $e^x\ e_1\ e^j\ e^y\ e_2\ e^x e^i\ e_k\ e_{k-1}\ e_1 e^x\ e_k\ e^y$: OK

(don't care if we relax other edges or the same ones repeatedly)