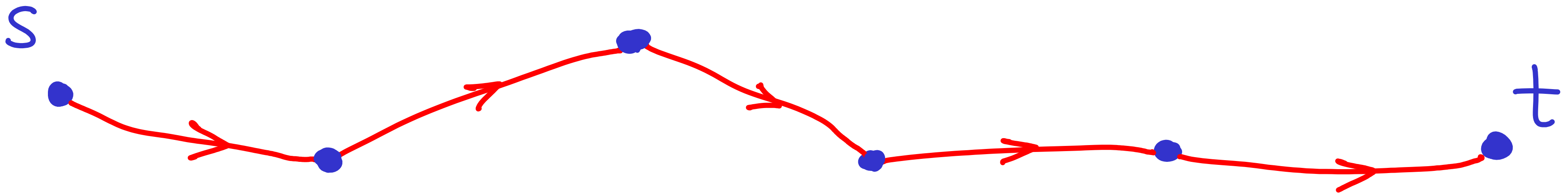
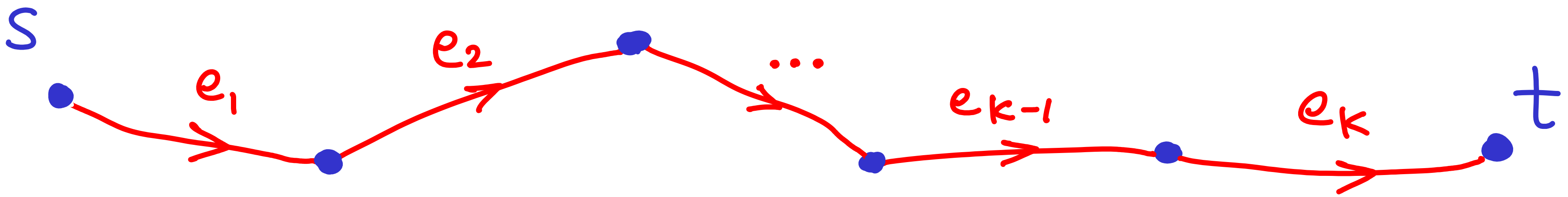
Assume this is a shortest path from s to t

unknown but exists



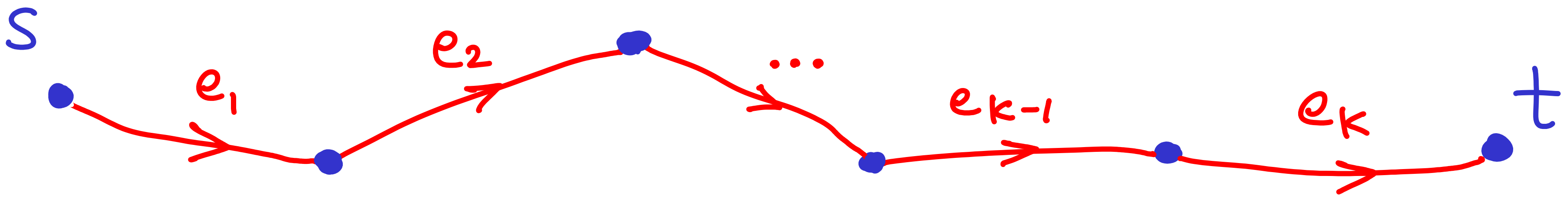
Assume this is a shortest path from s to t

unknown but exists



Assume this is a shortest path from s to t

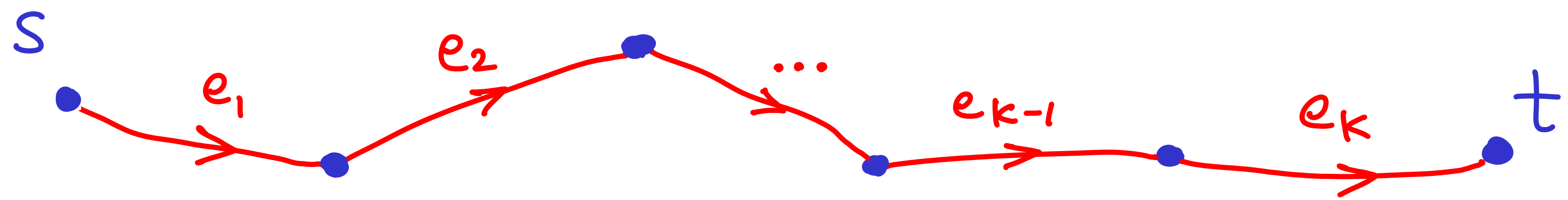
unknown but exists



Suppose we have an algorithm based on relaxing edges.

Assume this is a shortest path from s to t

unknown but exists

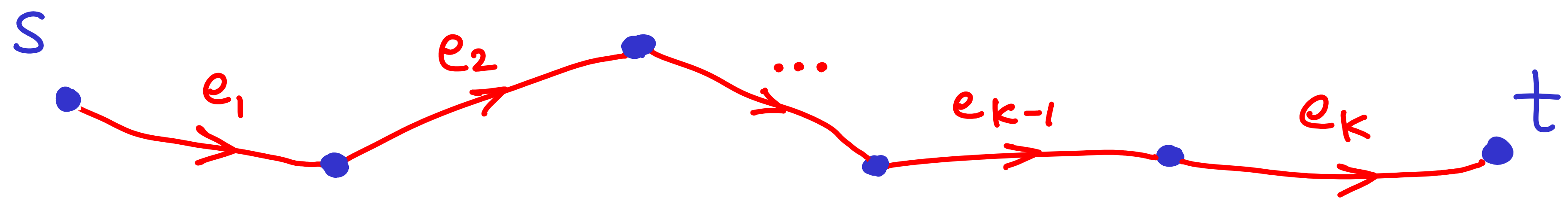


Suppose we have an algorithm based on relaxing edges.

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

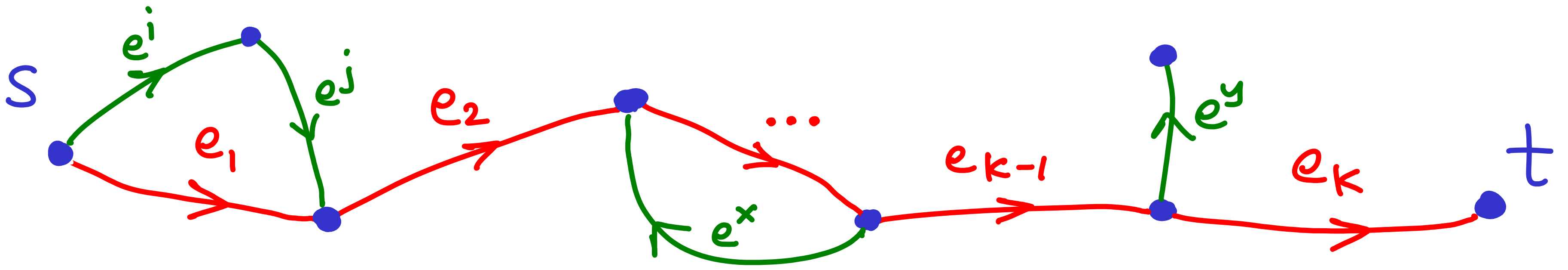


Suppose we have an algorithm based on relaxing edges.

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

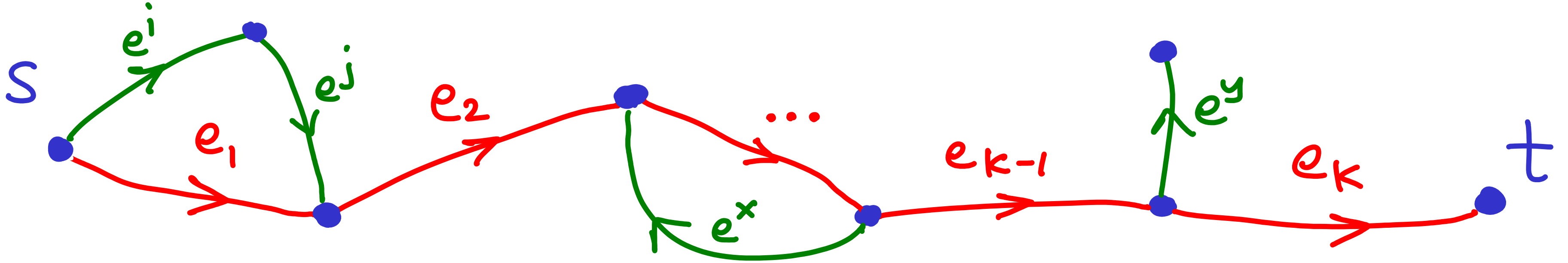
then we will correctly compute $d(t)$

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

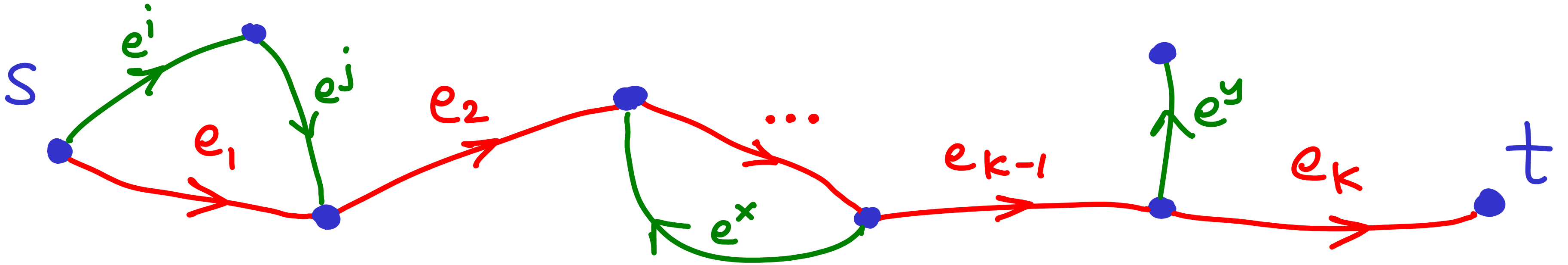
then we will correctly compute $d(t)$

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.

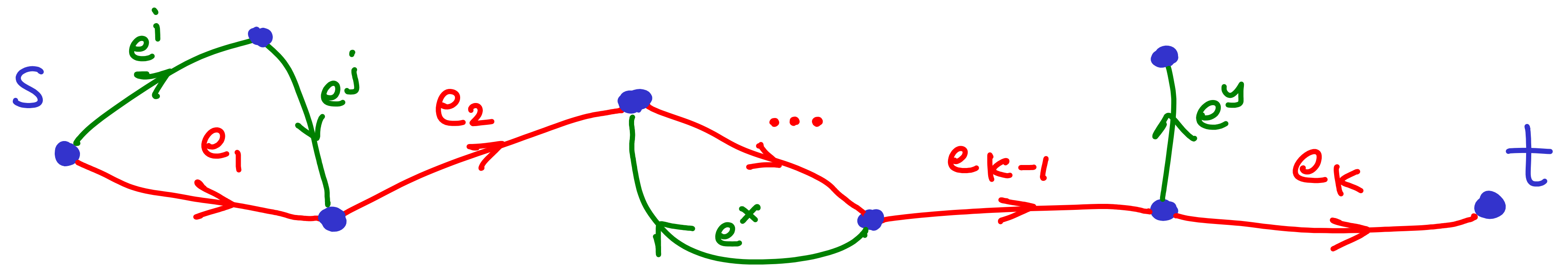
If we relax e_1 before e_2 before ... before e_{k-1} before e_k

then we will correctly compute $d(t)$ $\boxed{\text{WHY?}}$

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 $d(t)$ $\boxed{\text{by INDUCTION}}$

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)

BELLMAN-FORD ALGORITHM

simple, but made even simpler

(not identifying negative cycles)

Finds a shortest path from s to ALL vertices

BELLMAN-FORD ALGORITHM

simple, but made even simpler
(not identifying negative cycles)

L. Ford (1956) → slower variant

R. Bellman (1958)

Finds a shortest path from s to ALL vertices

BELLMAN-FORD ALGORITHM

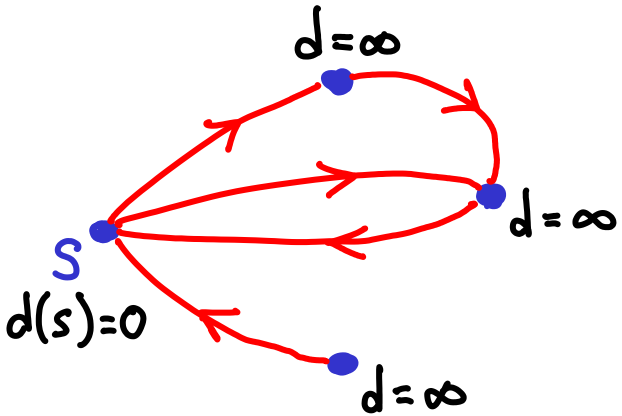
simple, but made even simpler
(not identifying negative cycles)

A. Shimbel (1954) → slower variant
L. Ford (1956) → slower variant
E. Moore (1957) → non-negative weights
R. Bellman (1958)

Finds a shortest path from s to ALL vertices

BELLMAN-FORD ALGORITHM

1) set score of s : zero
   set score of ≠s : ∞
   set parent of ≠s : null

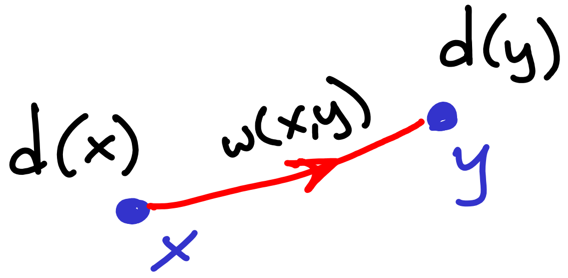


BELLMAN-FORD ALGORITHM

1) set score of s : zero
   set score of $\neq s$: ∞
   set parent of $\neq s$: null

2) for $i = 1$ to $V-1$
   RELAX every edge in G

BELLMAN-FORD ALGORITHM

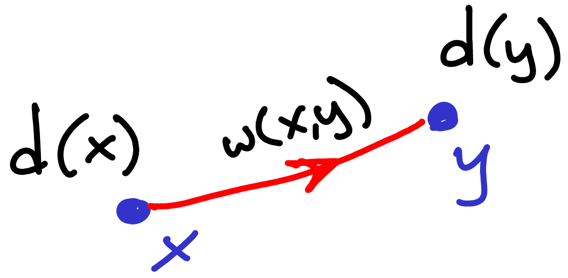


1) set score of s : zero
   set score of $\neq s$: ∞
   set parent of $\neq s$: null

2) for $i = 1$ to $V-1$
   RELAX every edge in G

RELAX an edge $x \to y$
if $d(x) + w(x,y) < d(y)$
then $d(y) = d(x) + w(x,y)$
parent$(y) = x$

BELLMAN-FORD ALGORITHM time?



1) set score of s : zero
set score of $\neq s$: ∞
set parent of $\neq s$: null

2) for $i = 1$ to $V-1$

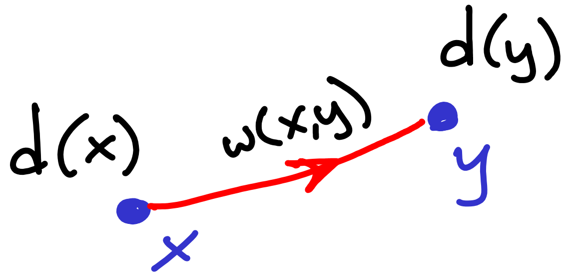
RELAX every edge in G

RELAX an edge $x \to y$

if $d(x) + w(x,y) < d(y)$

then $d(y) = d(x) + w(x,y)$
$parent(y) = x$

BELLMAN-FORD ALGORITHM $O(V \cdot E)$



1) set score of s : zero
   set score of $\neq s$: ∞
   set parent of $\neq s$: null

2) for $i = 1$ to $V-1$

   RELAX every edge in G

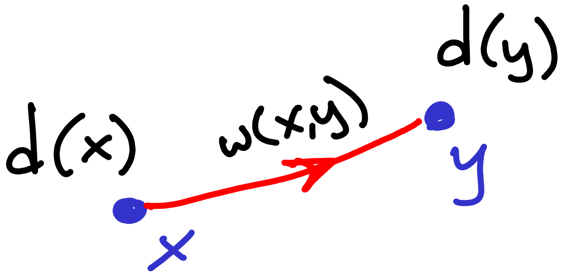
RELAX an edge $x \to y$

if $d(x) + w(x,y) < d(y)$

then $d(y) = d(x) + w(x,y)$
$\quad$ parent$(y) = x$

BELLMAN-FORD ALGORITHM $O(V \cdot E)$

1) set score of s : zero
   set score of $\neq$s : ∞
   set parent of $\neq$s : null



Why does this work?

2) for $i = 1$ to $V-1$

   RELAX every edge in G

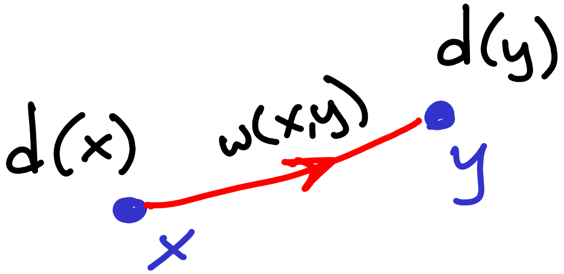
RELAX an edge $x \to y$

if $d(x) + w(x,y) < d(y)$

then $d(y) = d(x) + w(x,y)$
     parent$(y) = x$

BELLMAN-FORD ALGORITHM $O(V \cdot E)$

1) set score of s : zero
set score of $\neq s$: ∞
set parent of $\neq s$: null



2) for $i = 1$ to $V-1$

RELAX every edge in G

RELAX an edge $x \to y$

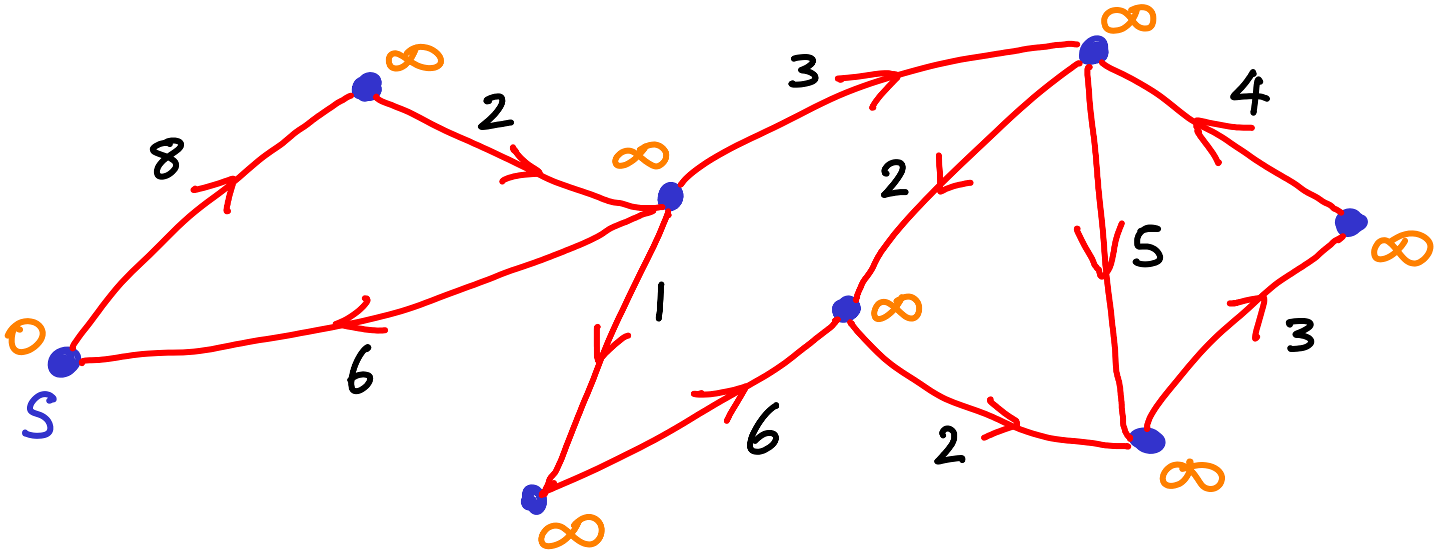
if $d(x) + w(x,y) < d(y)$

then $d(y) = d(x) + w(x,y)$
$\text{parent}(y) = x$

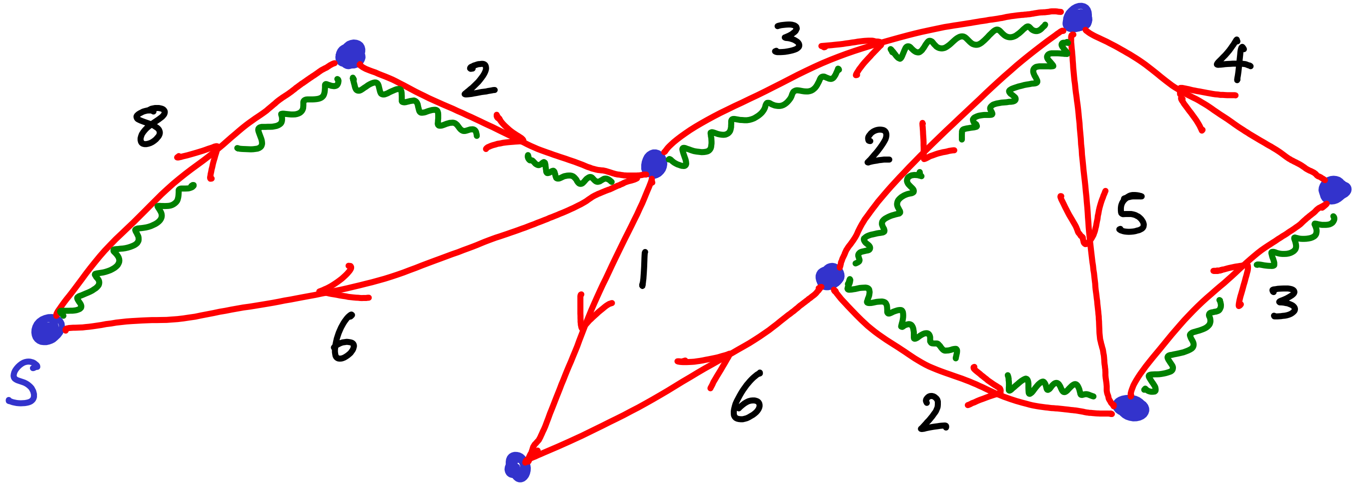
Why does this work?

FOR ANY t

In iteration i
we will get $d(p_i^t)$
on some shortest path p^t
$= \{p_1^t, p_2^t, p_3^t, \ldots, p_k^t = t\}$
from $s \to t$

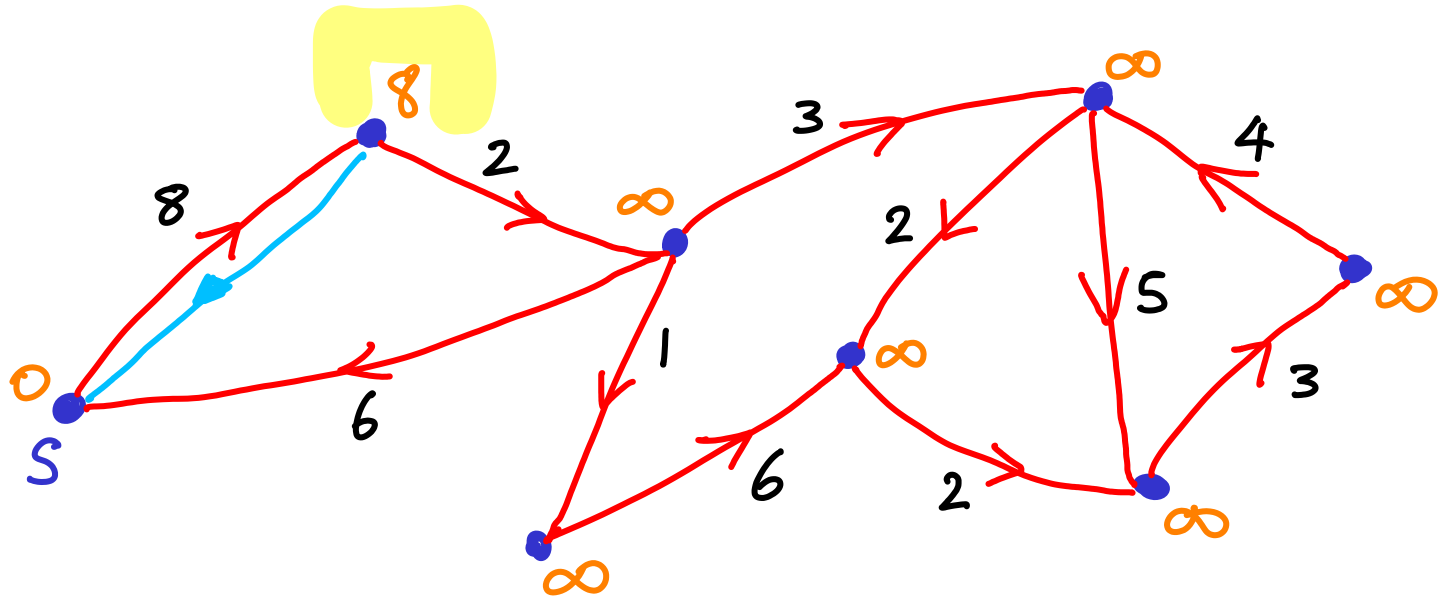


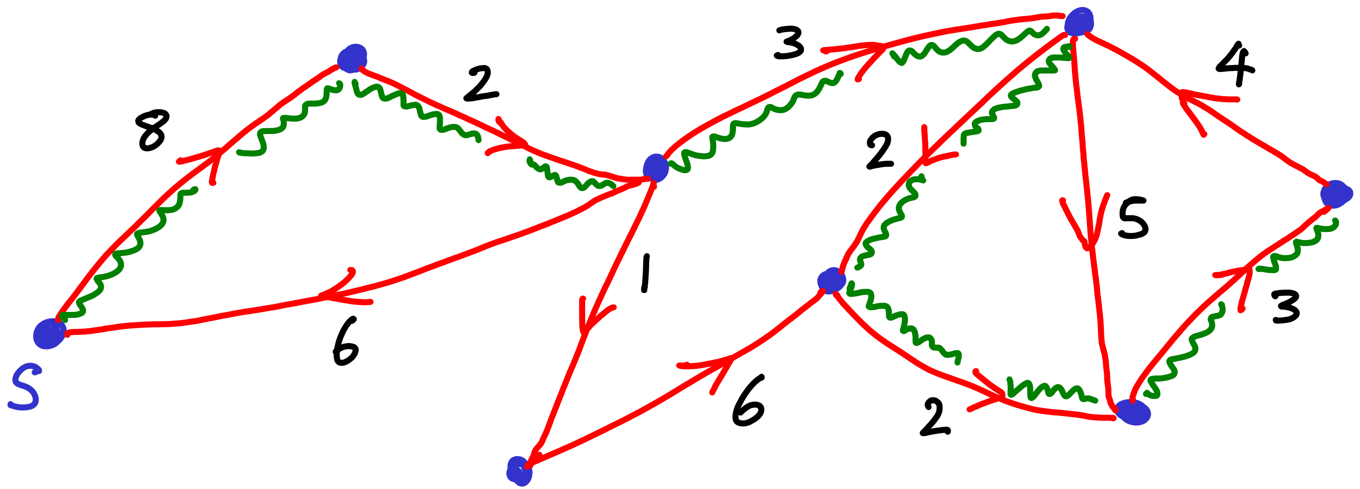
0
S
8
6
2
1
3
2
6
2
5
4
3
∞
∞
∞
∞
∞
∞
∞



S
8
6
2
1
3
2
6
2
5
4
3

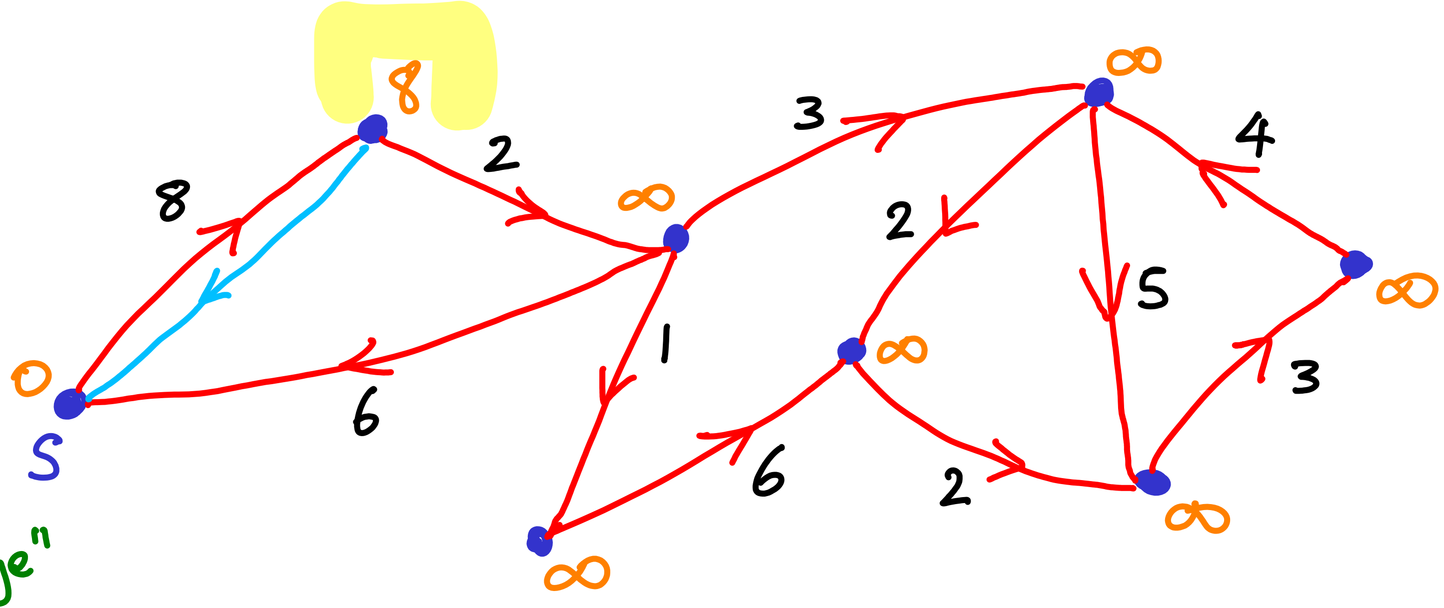
$i = 1$

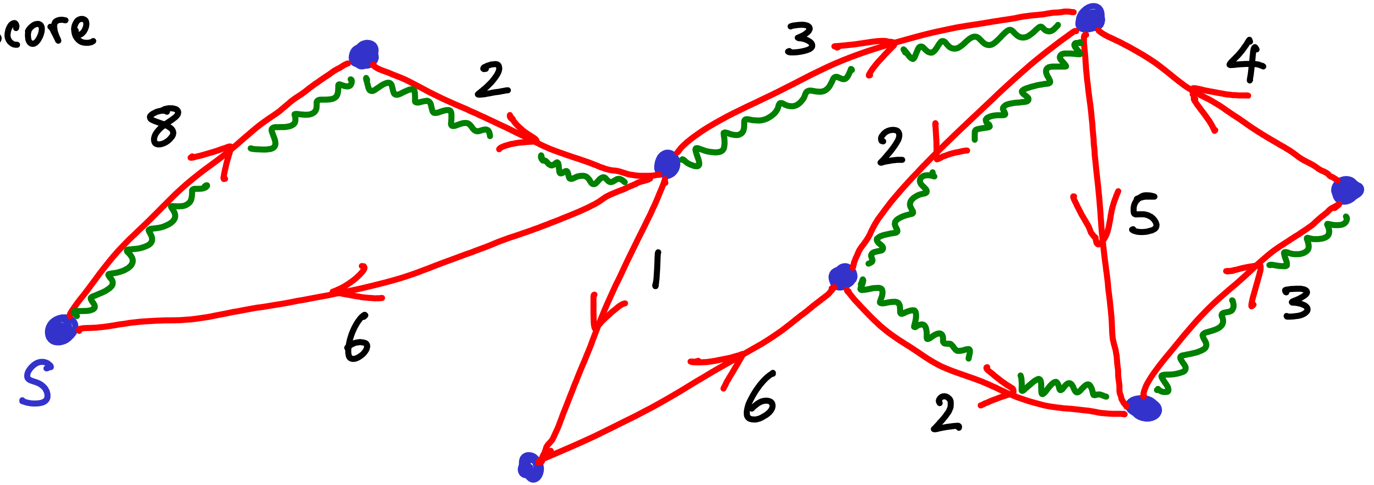




$i=1$

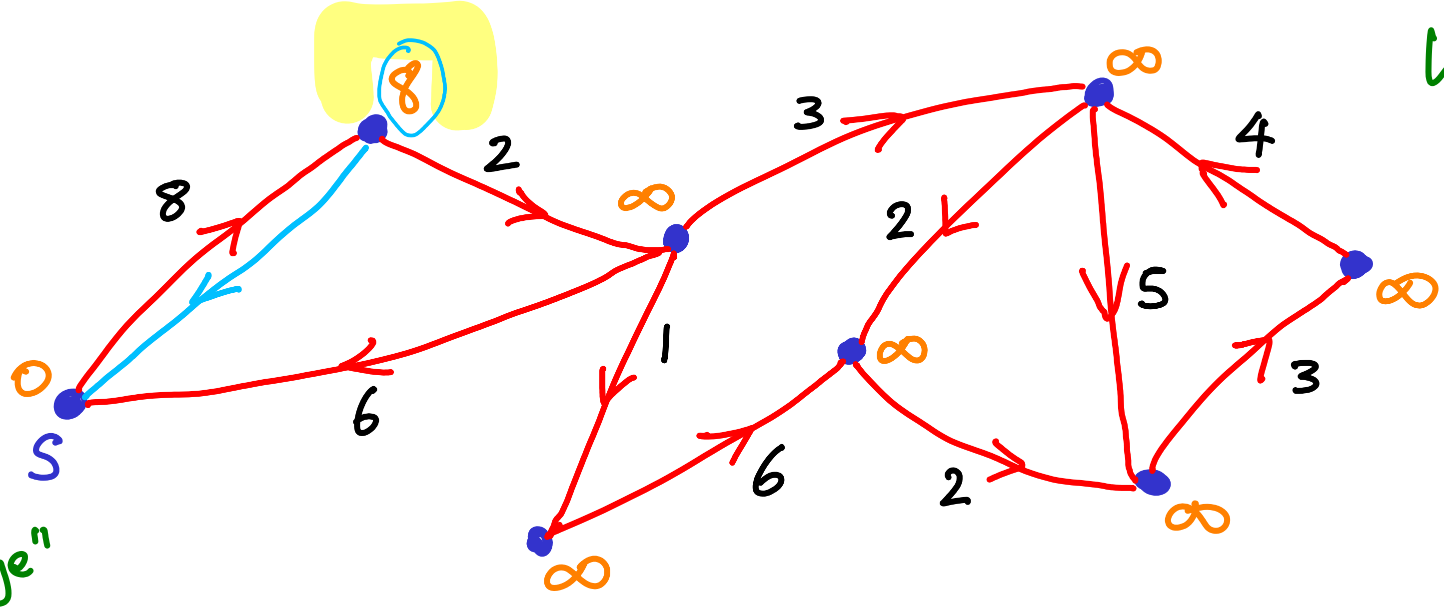
depending on order of processing edges in "RELAX every edge" we might get only 1 finite score or actually be done.

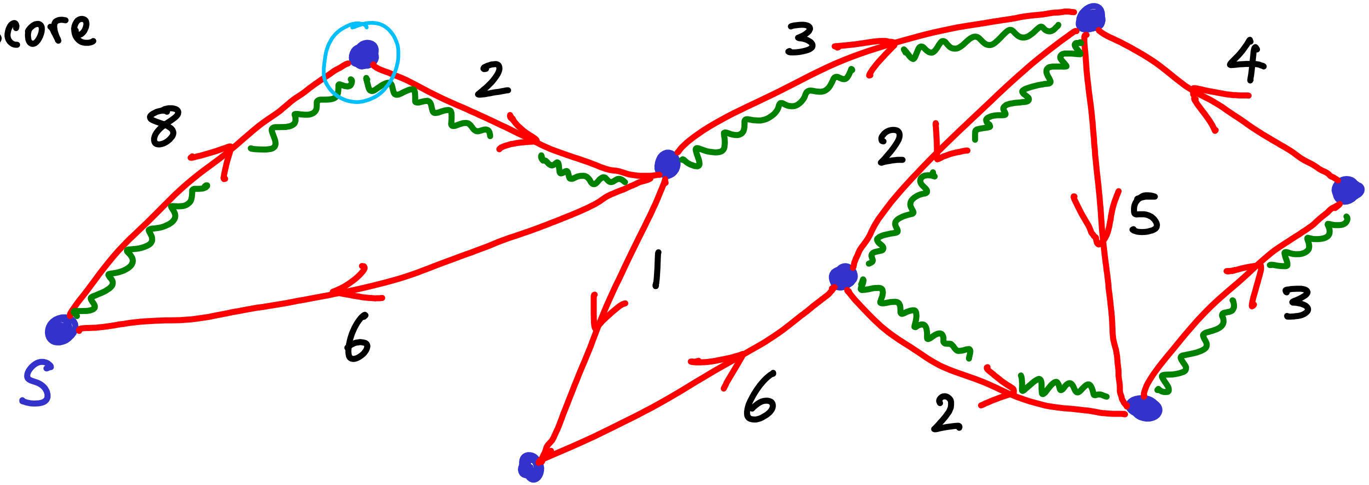




$i = 1$

depending on order of processing edges in "RELAX every edge" we might get only 1 finite score or actually be done.

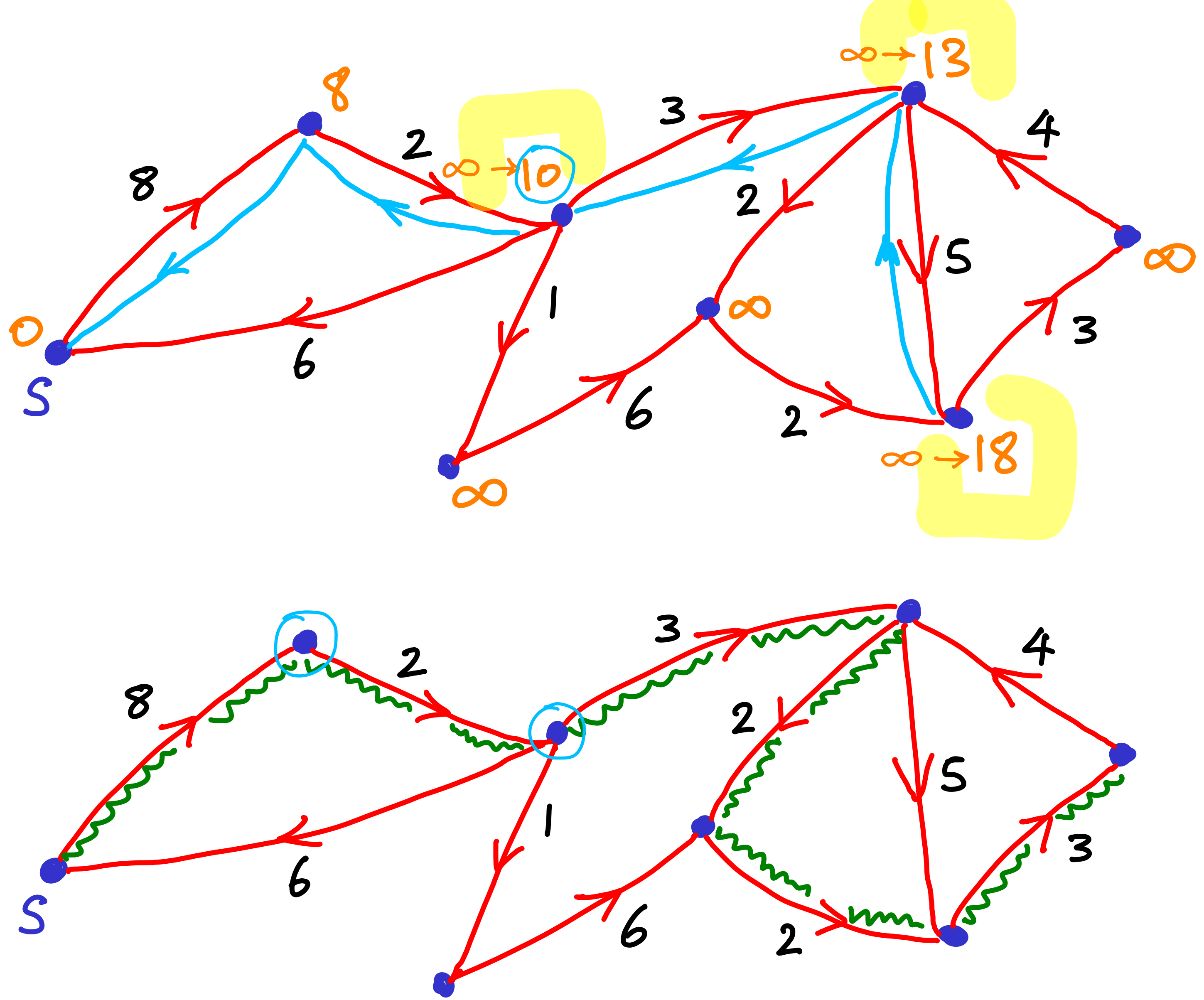




What matters is that we get final score of a vertex on shortest path to target

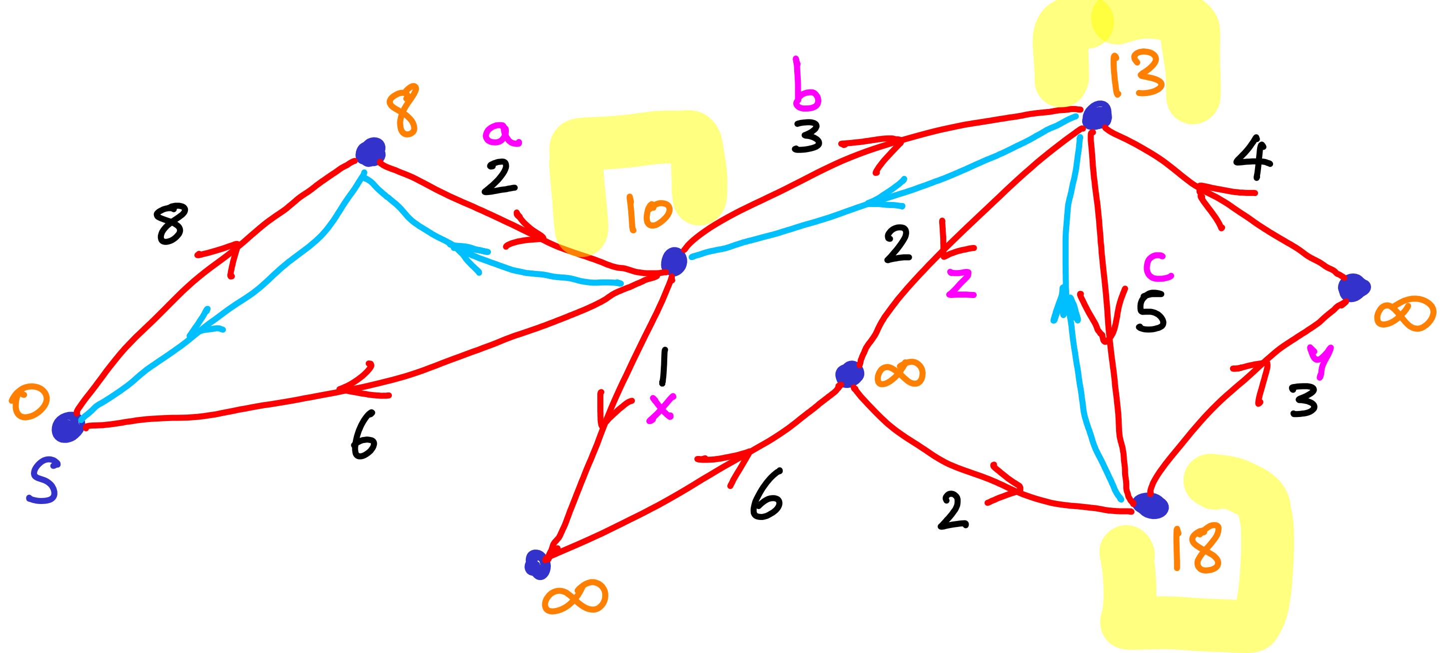
& in fact we extend a chain of such vertices.

$i = 2$



order of relaxing?

$i = 2$

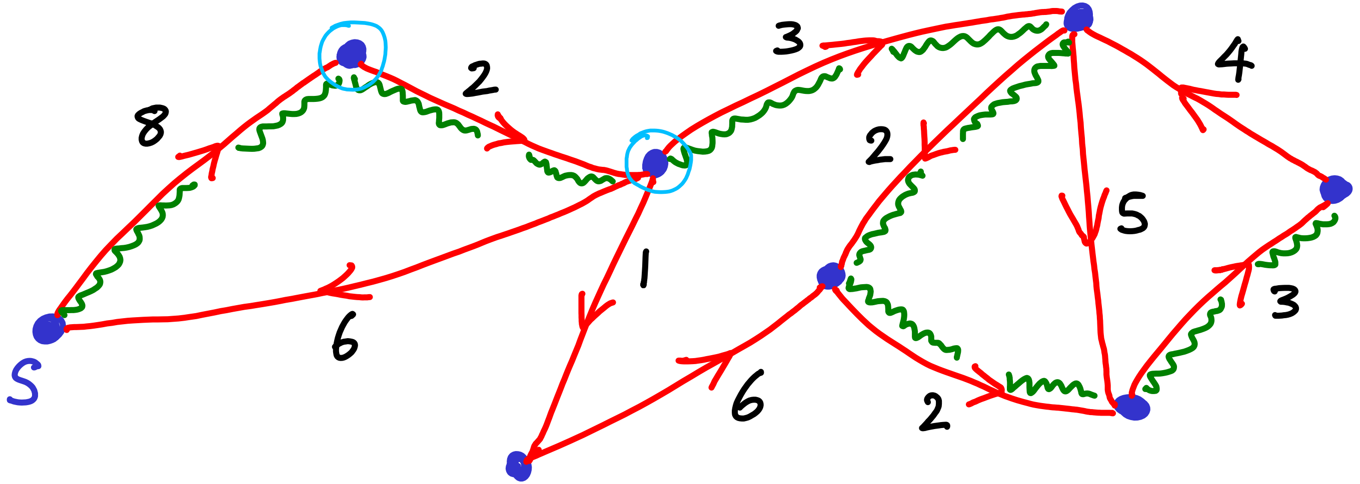


order of relaxing?

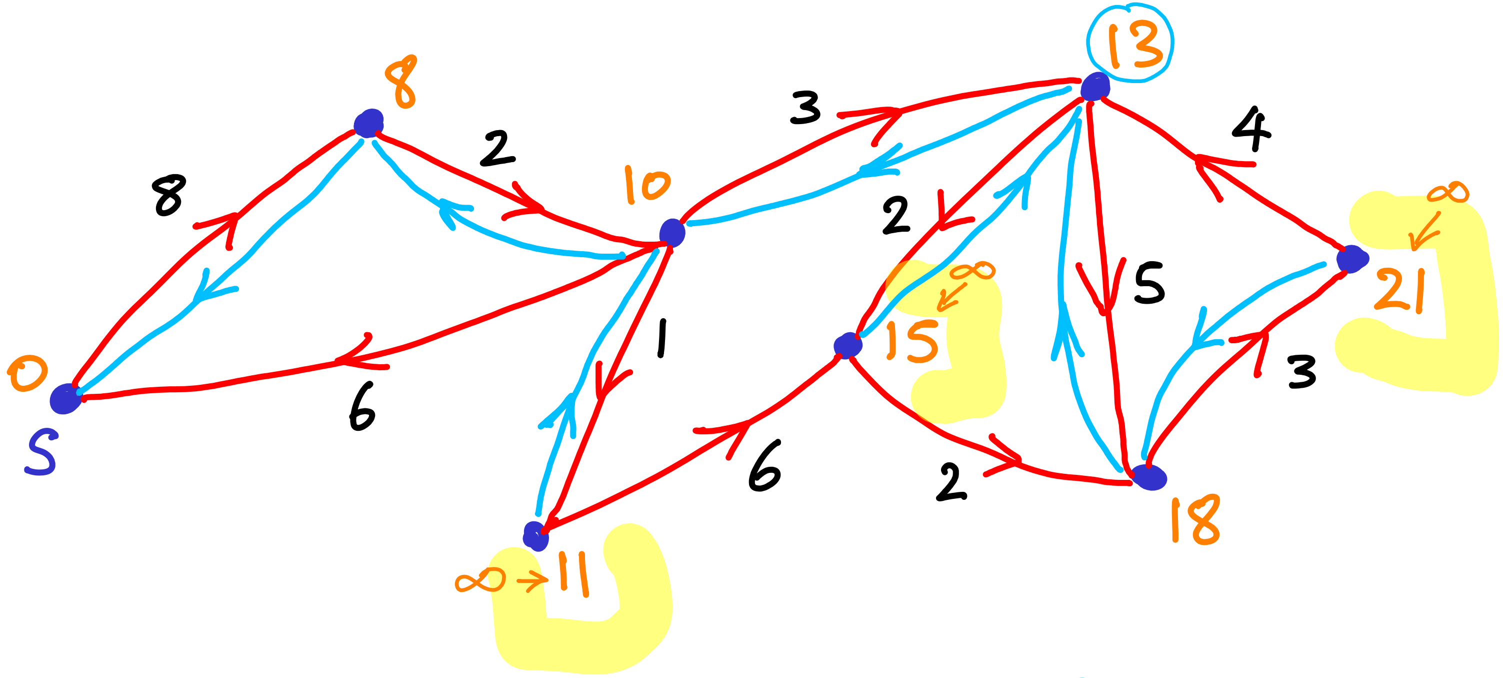
x...a...b...c

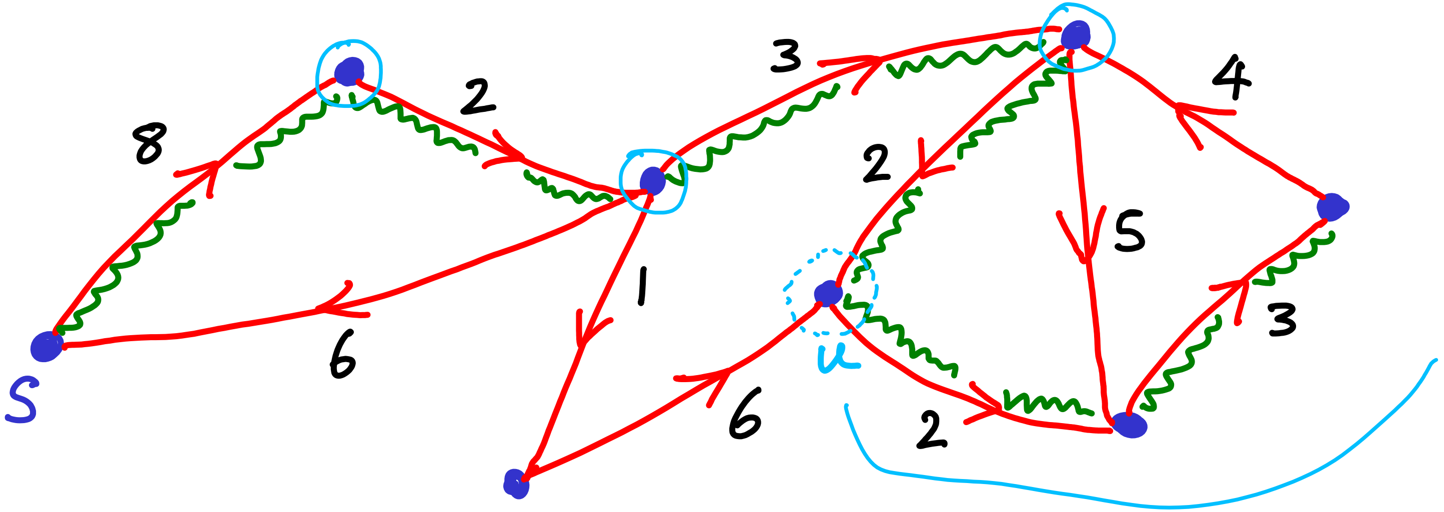
y...c

z...b



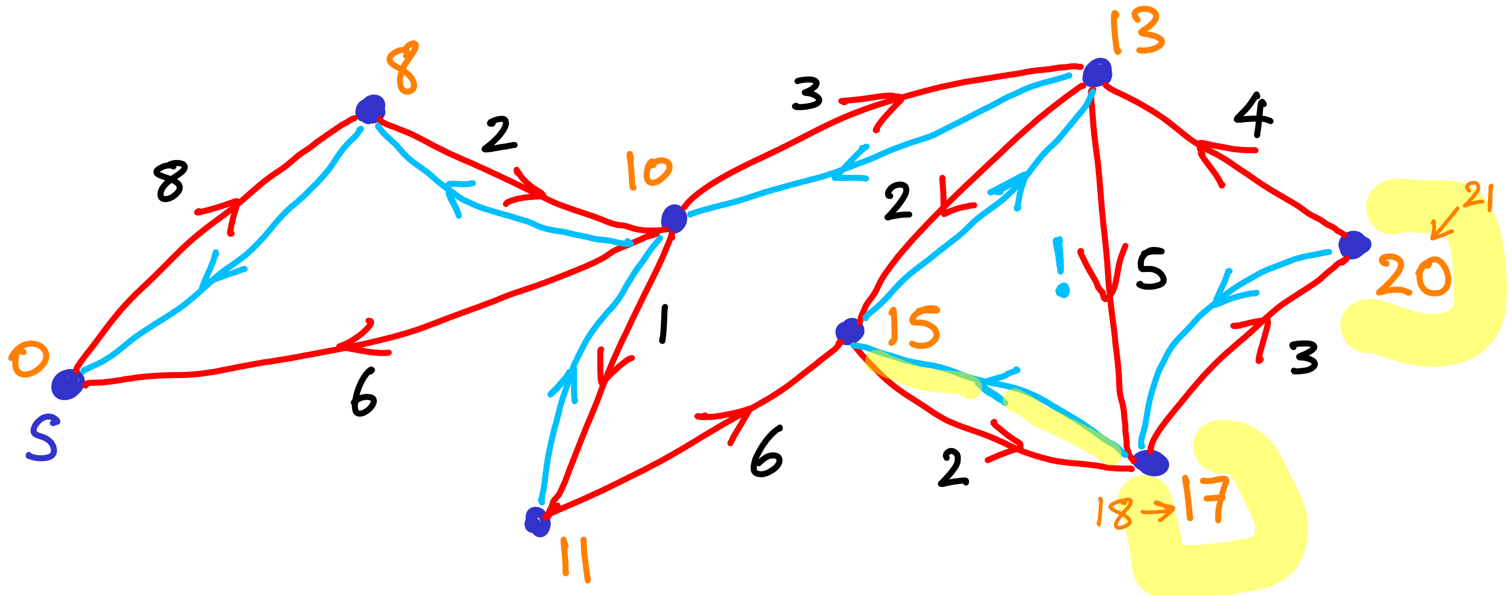
$i = 3$

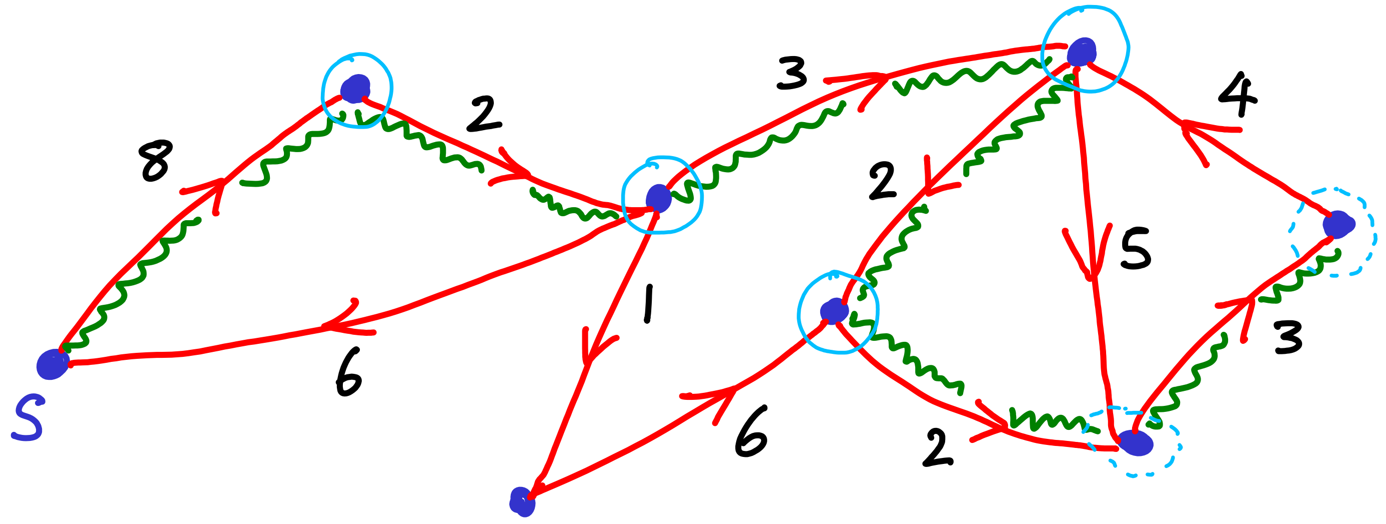




we have found shortest path to u ahead of schedule

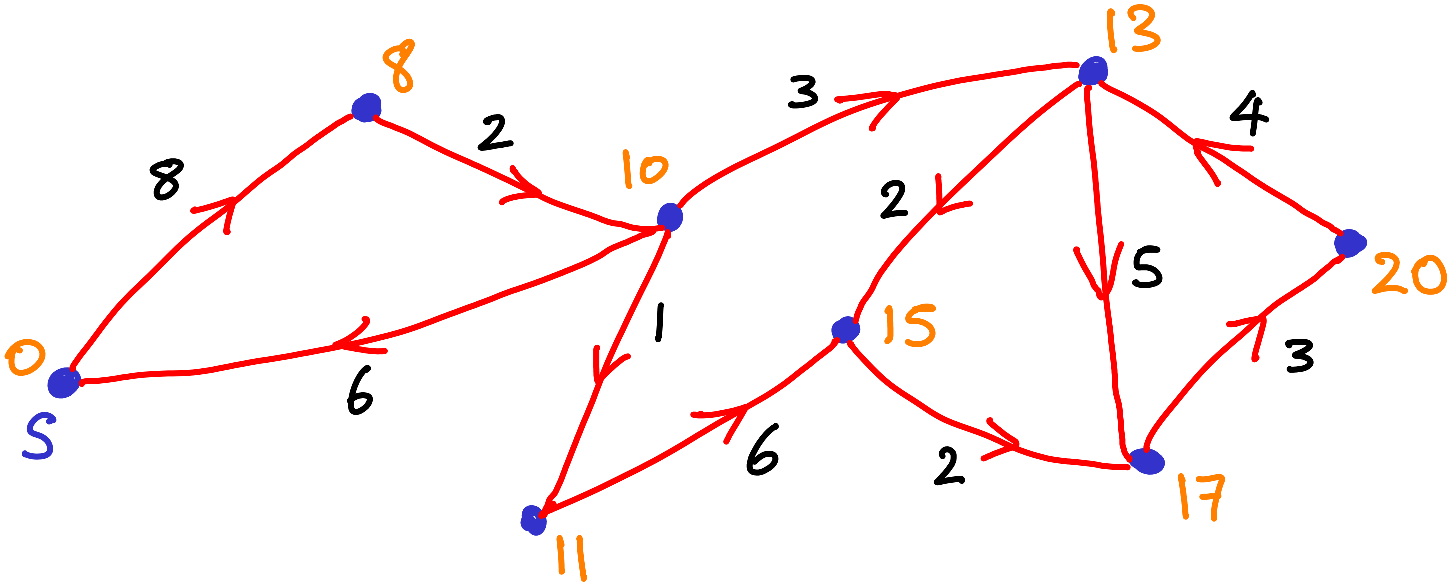
$i = 4$

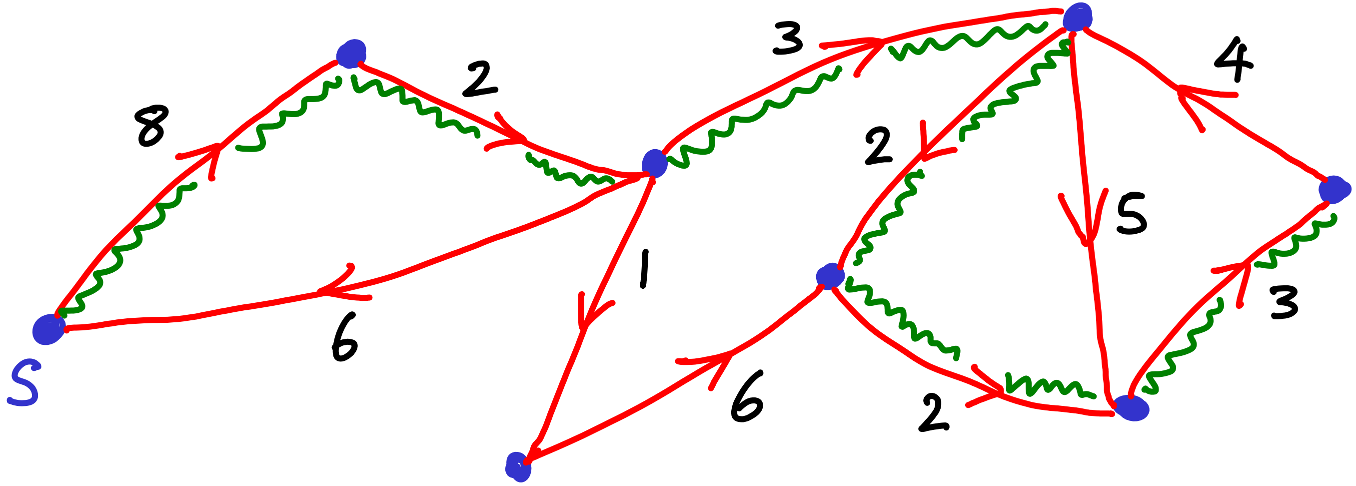




$i = 5, 6, 7$

redundant





BELLMAN-FORD ALGORITHM

Works for negative weights

& can detect negative cycles. (how?)

(see CLRS)

BELLMAN-FORD ALGORITHM

Works for negative weights

& can detect negative cycles.

scores will still be changing after $i = V-1$

(see CLRS)