COL 351 : Analysis & Design of Algorithms

LECTURE 27

DYNAMIC PROGRAMMING VII:

BELLMAN FORD ALGORITHM (CONTD.)

OCT 13, 2026 | ROHIT VAISH

SINGLE-SOURCE SHORTEST PATH PROBLEM

input: A directed graph $G = (V, E)$, starting vertex s, a non negative length l_e for each edge $e \in E$

output: dist (s, v) for every vertex $v \in V$.

SINGLE-SOURCE SHORTEST PATH PROBLEM

input: A directed graph $G = (V, E)$, starting vertex s, a non negative length ℓ_e for each edge $e \in E$

output: dist (s, v) for every vertex $v \in V$.

What about negative-length edges?

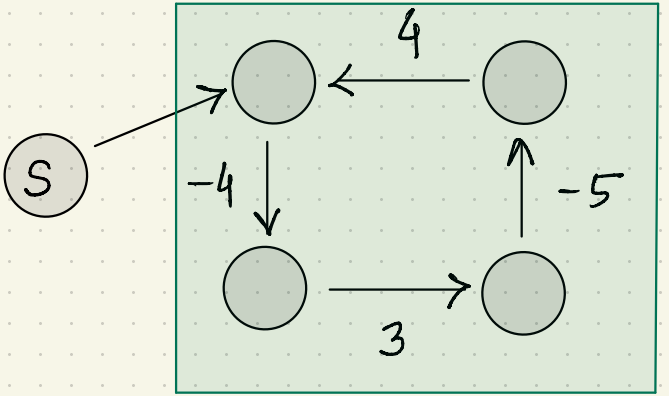
Dijkstra's algorithm fails ☹

NEGATIVE CYCLES

How to define shortest paths in the presence of negative cycles?

NEGATIVE CYCLES

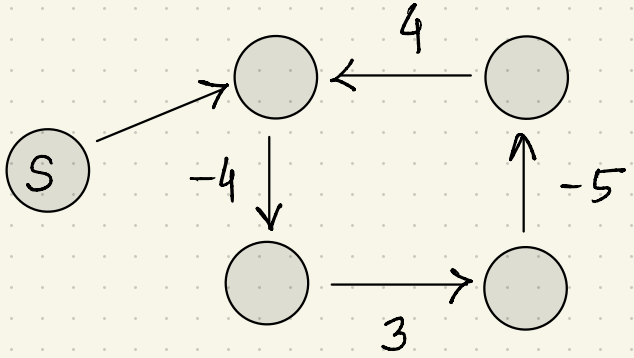
How to define shortest paths in the presence of negative cycles?



cycle with weight -2

NEGATIVE CYCLES

How to define shortest paths in the presence of negative cycles?

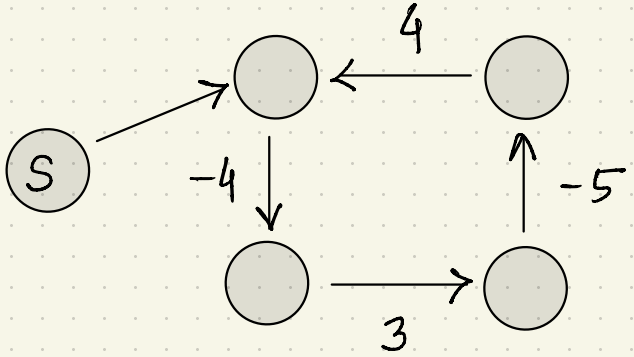


If we include cycles in shortest paths

then length of the shortest path is undefined (or $-\infty$).

NEGATIVE CYCLES

How to define shortest paths in the presence of negative cycles?

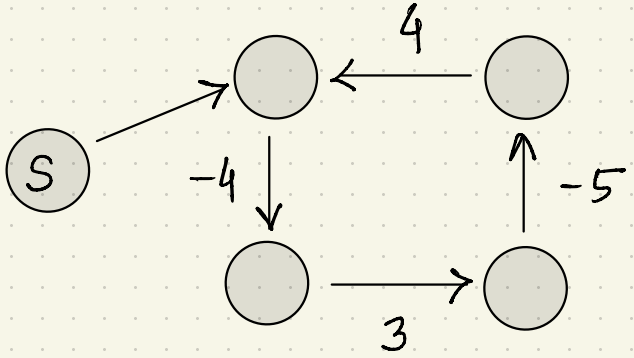


If we require shortest cycle-free path

then the problem is NP-complete (no known poly-time algo.)

NEGATIVE CYCLES

How to define shortest paths in the presence of negative cycles?



Solution: Allow negative length edges but not negative cycles.

NEGATIVE CYCLES

No negative cycles $\Rightarrow$ shortest $s - v$ path has at most $n-1$ edges

SINGLE-SOURCE SHORTEST PATH PROBLEM

input: A directed graph $G = (V, E)$ (possibly contains negative cycles), starting vertex s, a possibly negative length ℓ_e for each edge $e \in E$

SINGLE-SOURCE SHORTEST PATH PROBLEM

input: A directed graph $G = (V, E)$, starting vertex s, a possibly negative length ℓ_e for each edge $e \in E$

output: (1) for all destinations $v \in V$, compute the length of the shortest $s-v$ path, or

(2) output a negative cycle.

SINGLE-SOURCE SHORTEST PATH PROBLEM

input: A directed graph $G = (V, E)$, starting vertex s, a possibly negative length ℓ_e for each edge $e \in E$

output: (1) for all destinations $v \in V$, compute the length of the shortest $s - v$ path, or

(2) output a negative cycle.

excuse for failing to find a shortest path

OPTIMAL SUBSTRUCTURE

OPTIMAL SUBSTRUCTURE

No inherent sequentiality in the input (general graph)
but the output (i.e., a path) is sequential.

OPTIMAL SUBSTRUCTURE

No inherent sequentiality in the input (general graph)
but the output (i.e., a path) is sequential.

Intuition: Subpath of shortest path should itself be shortest.

OPTIMAL SUBSTRUCTURE

No inherent sequentiality in the input (general graph)
but the output (i.e., a path) is sequential.

Intuition: Subpath of shortest path should itself be shortest.

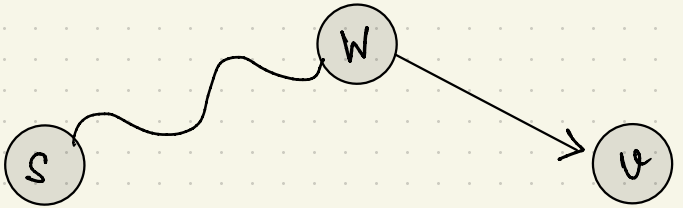
Difficulty: Not clear in what order to process the vertices.
How to define "smaller" or "larger" subproblems?

OPTIMAL SUBSTRUCTURE

No inherent sequentiality in the input (general graph)
but the output (i.e., a path) is sequential.

Intuition: Subpath of shortest path should itself be shortest.

Difficulty: Not clear in what order to process the vertices.
How to define "smaller" or "larger" subproblems?



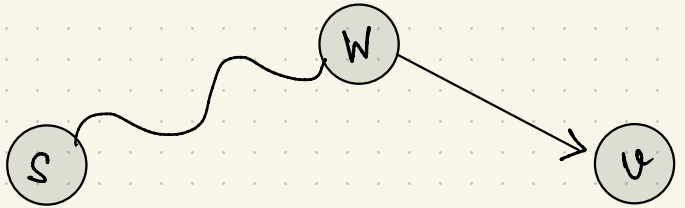
Possible that length of $s \rightsquigarrow w$ > length of $s \rightsquigarrow v$

OPTIMAL SUBSTRUCTURE

No inherent sequentiality in the input (general graph)
but the output (i.e., a path) is sequential.

Intuition: Subpath of shortest path should itself be shortest.

Difficulty: Not clear in what order to process the vertices.
How to define "smaller" or "larger" subproblems?





edges in $s \rightsquigarrow w$ $\leq$ # edges in $s \rightsquigarrow v$

OPTIMAL SUBSTRUCTURE

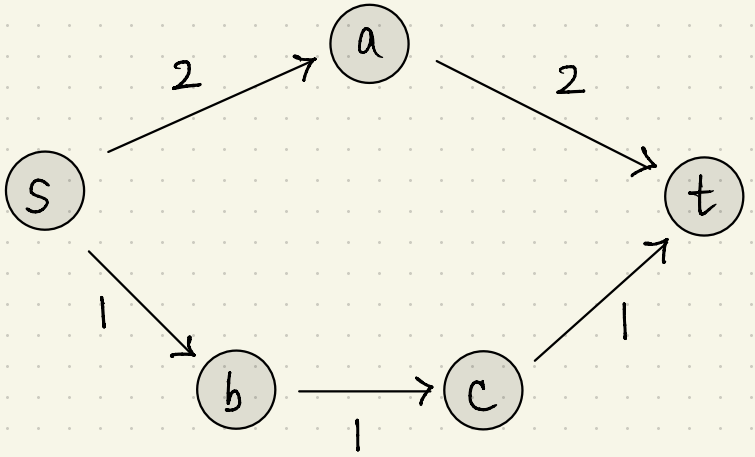
No inherent sequentiality in the input (general graph)
but the output (i.e., a path) is sequential.

Intuition: Subpath of shortest path should itself be shortest.

Difficulty: Not clear in what order to process the vertices.
How to define "smaller" or "larger" subproblems?

Key idea: Introduce an artificial parameter: # edges in a path

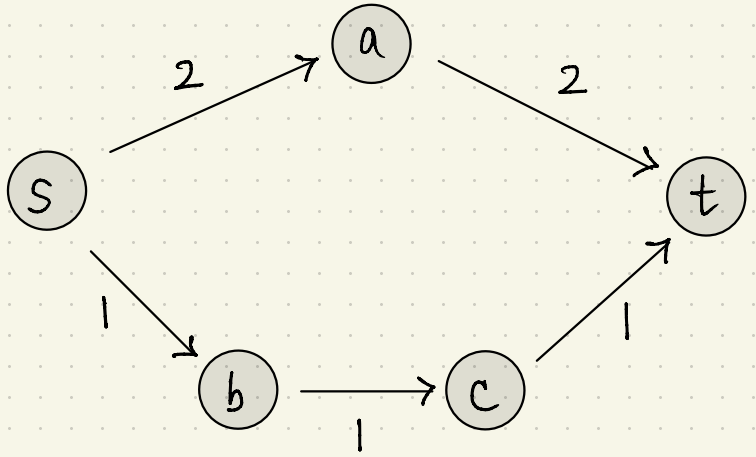
OPTIMAL SUBSTRUCTURE



Key idea: Introduce an artificial parameter: # edges in a path

OPTIMAL SUBSTRUCTURE

One subproblem for each destination and edge budget

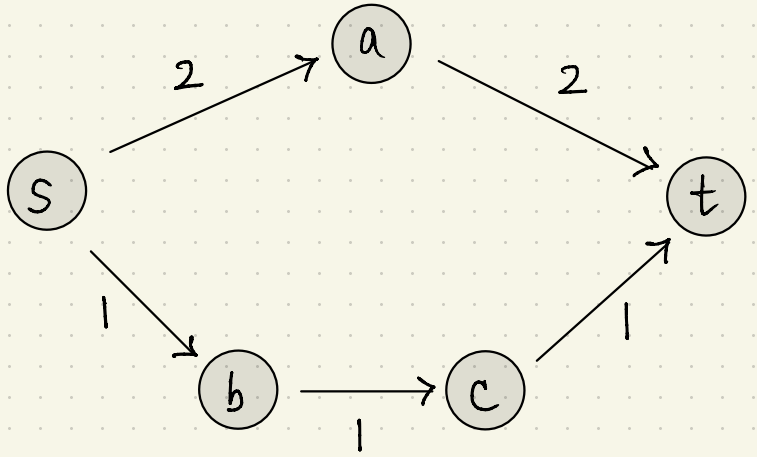


Key idea: Introduce an artificial parameter: # edges in a path

OPTIMAL SUBSTRUCTURE

s —— t paths

One subproblem for each destination and edge budget



Key idea: Introduce an artificial parameter: # edges in a path

OPTIMAL SUBSTRUCTURE

s —— t paths

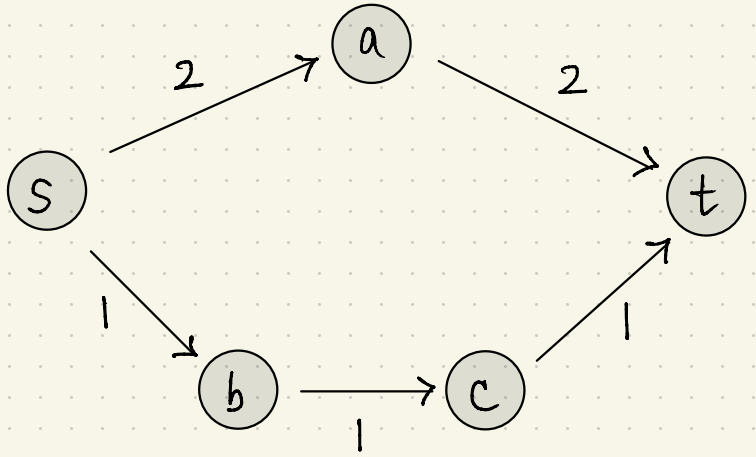
edges = 2

⇒ s — a — t

One subproblem

for each destination

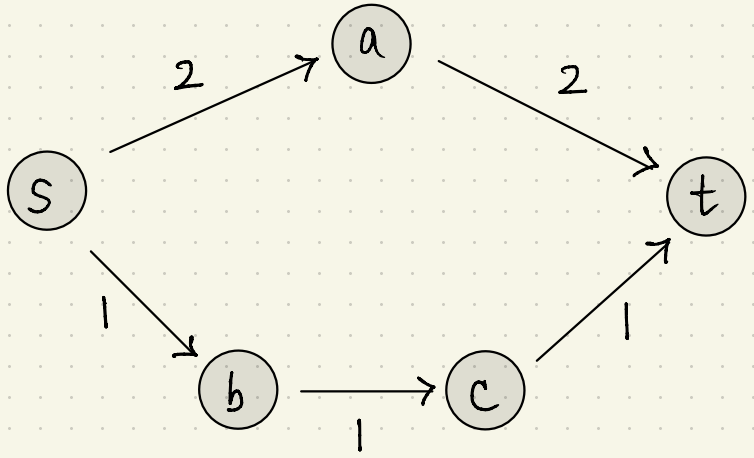
and edge budget



Key idea: Introduce an artificial parameter: # edges in a path

OPTIMAL SUBSTRUCTURE

One subproblem for each destination and edge budget



s —— t paths

edges = 2

⇒ s – a – t

edges = 3

⇒ s – b – c – t

Key idea: Introduce an artificial parameter: # edges in a path

OPTIMAL SUBSTRUCTURE

Lemma: Let $G = (V, E)$ be a directed graph with edge lengths l_e and starting vertex s.

OPTIMAL SUBSTRUCTURE

Lemma: Let $G = (V, E)$ be a directed graph with edge lengths l_e and starting vertex s.

For every destination $v \in V$ and every $i \in \{1, 2, \cdots\}$, let

P be the shortest $s-v$ path with at most i edges.

OPTIMAL SUBSTRUCTURE

Lemma: Let $G = (V, E)$ be a directed graph with edge lengths l_e and starting vertex s.

For every destination $v \in V$ and every $i \in \{1, 2, \cdots\}$, let

P be the shortest $s-v$ path with at most i edges.

Case I: P has $(i-1)$ or fewer edges

$\Rightarrow$ P is the shortest $s-v$ path with at most $(i-1)$ edges.

OPTIMAL SUBSTRUCTURE

Lemma: Let $G = (V, E)$ be a directed graph with edge lengths l_e and starting vertex s.

For every destination $v \in V$ and every $i \in \{1, 2, \cdots\}$, let

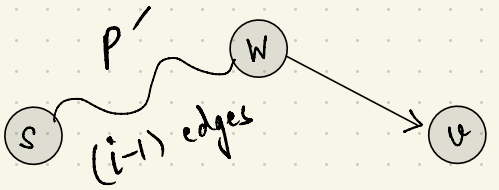
P be the shortest $s-v$ path with at most i edges.

Case I: P has $(i-1)$ or fewer edges

$\Rightarrow$ P is the shortest $s-v$ path with at most $(i-1)$ edges.

Case II: P has i edges

$\Rightarrow$ P' is the shortest $s-w$ path with at most $(i-1)$ edges



OPTIMAL SUBSTRUCTURE

For any given destination v, how many candidates are there for an optimal solution to a subproblem involving v?

OPTIMAL SUBSTRUCTURE

For any given destination v, how many candidates are there for an optimal solution to a subproblem involving v?

$1 + \text{in-degree}(v)$

OPTIMAL SUBSTRUCTURE

For any given destination v, how many candidates are there for an optimal solution to a subproblem involving v?

$$1 + \text{in-degree}(v)$$

inherited from Case I

from case II,

for each w s.t. $(w, v) \in E$.

THE RECURRENCE

Define $L_{i,v}$:= length of shortest $s \to v$ path with $\leq i$ edges

For every $v \in V$ and every $i \in \{0, 1, 2, \ldots\}$

$$L_{i,v} = \min \begin{cases} L_{(i-1),v} & \text{// Case I} \\ \min_{(w,v) \in E} L_{(i-1),w} + l_{w,v} & \text{// Case II} \end{cases}$$

THE RECURRENCE

Define $L_{i,v}$:= length of shortest $s \to v$ path with $\leq i$ edges

For every $v \in V$ and every $i \in \{0, 1, 2, \ldots\}$

$$L_{i,v} = \min \begin{cases} L_{(i-1),v} & \text{// Case I} \\ \min_{(w,v) \in E} L_{(i-1),w} + l_{w,v} & \text{// Case II} \end{cases}$$

Correctness: by the optimal substructure lemma

IF NO NEGATIVE CYCLES

IF NO NEGATIVE CYCLES

Recall that if input graph G has no negative cycle then shortest paths have $\leq (n-1)$ edges.

IF NO NEGATIVE CYCLES

Recall that if input graph G has no negative cycle then shortest paths have $\leq (n-1)$ edges.

$\Rightarrow$ edge budget $i \leq n-1$.

IF NO NEGATIVE CYCLES

Recall that if input graph G has no negative cycle then shortest paths have $\leq (n-1)$ edges.

$\Rightarrow$ edge budget $i \leq n-1$.

Subproblems : compute $L_{i,v}$ for all $i \in \{0, 1, \cdots, n-1\}$ and all $v \in V$.

IF NO NEGATIVE CYCLES

Recall that if input graph G has no negative cycle then shortest paths have $\leq (n-1)$ edges.

$\Rightarrow$ edge budget $i \leq n-1$.

Subproblems : compute $L_{i,v}$ for all $i \in \{0, 1, \cdots, n-1\}$ and all $v \in V$.

NOTE : $O(n^2)$ subproblems overall

$O(n)$ subproblems per destination

BELLMAN-FORD ALGORITHM

A := $n \times n$ dimensional array // budget $\times$ destinations

// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$

// recurrence

for $i \in \{1, 2, \ldots, n-1\}$

for each $v \in V$

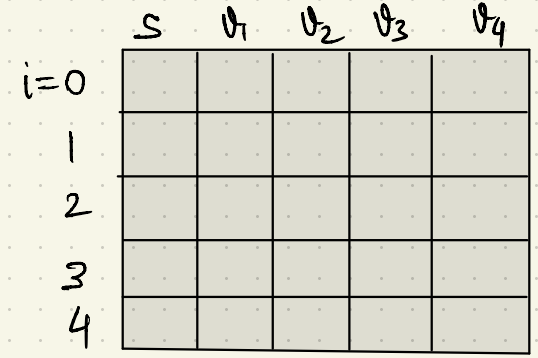
$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

BELLMAN-FORD ALGORITHM

A := $n \times n$ dimensional array // budget $\times$ destinations

// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$



// recurrence

for $i \in \{1, 2, \ldots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

BELLMAN-FORD ALGORITHM

A := $n \times n$ dimensional array // budget × destinations

// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$

	s	v_1	v_2	v_3	v_4
$i=0$	0				
1					
2					
3					
4					

// recurrence

for $i \in \{1, 2, \ldots, n-1\}$

  for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

BELLMAN-FORD ALGORITHM

A := $n \times n$ dimensional array // budget $\times$ destinations

// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$

	s	v_1	v_2	v_3	v_4
$i=0$	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1					
2					
3					
4					

// recurrence

for $i \in \{1, 2, \ldots, n-1\}$

  for each $v \in V$

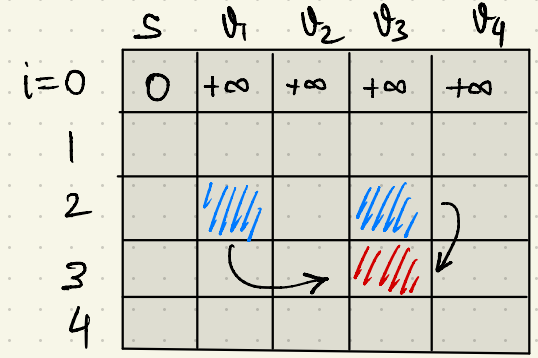
$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

BELLMAN-FORD ALGORITHM

A := $n \times n$ dimensional array // budget × destinations

// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$



// recurrence

for $i \in \{1, 2, \ldots, n-1\}$

for each $v \in V$

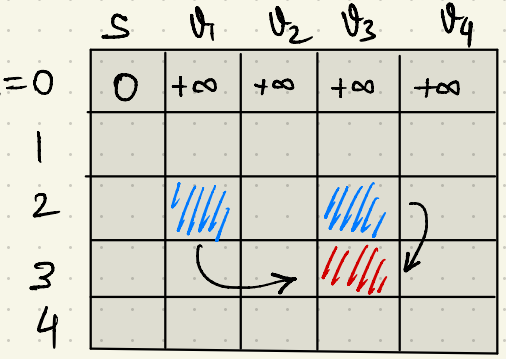
$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

BELLMAN-FORD ALGORITHM

A := $n \times n$ dimensional array // budget $\times$ destinations

// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$



// recurrence

for $i \in \{1, 2, \ldots, n-1\}$ — order is important

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

BELLMAN-FORD ALGORITHM

A := $n \times n$ dimensional array // budget $\times$ destinations

// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$

	s	v_1	v_2	v_3	v_4
$i=0$	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$
1					
2					
3					
4					

// recurrence

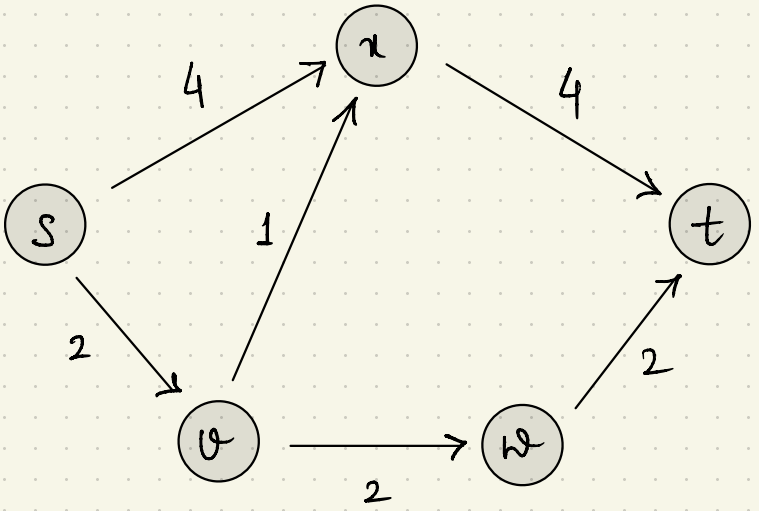
for $i \in \{1, 2, \ldots, n-1\}$ — order is important

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

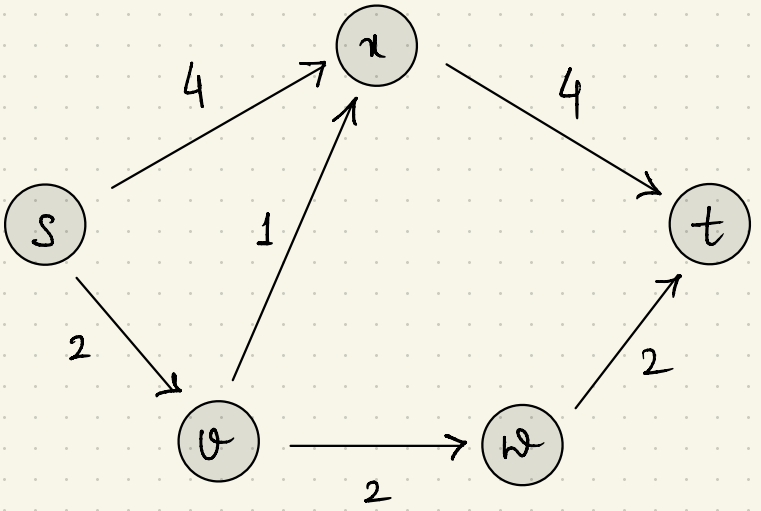
If G has no negative cycle, then the algorithm is correct.

EXAMPLE



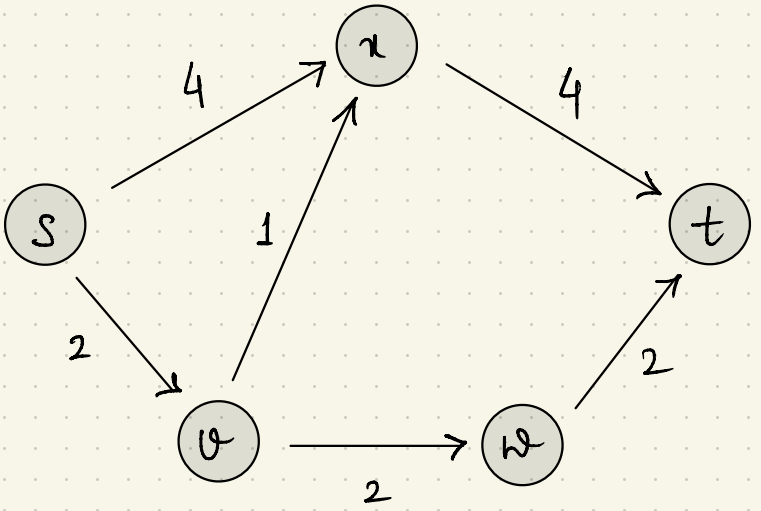
EXAMPLE

$i = 0$



EXAMPLE

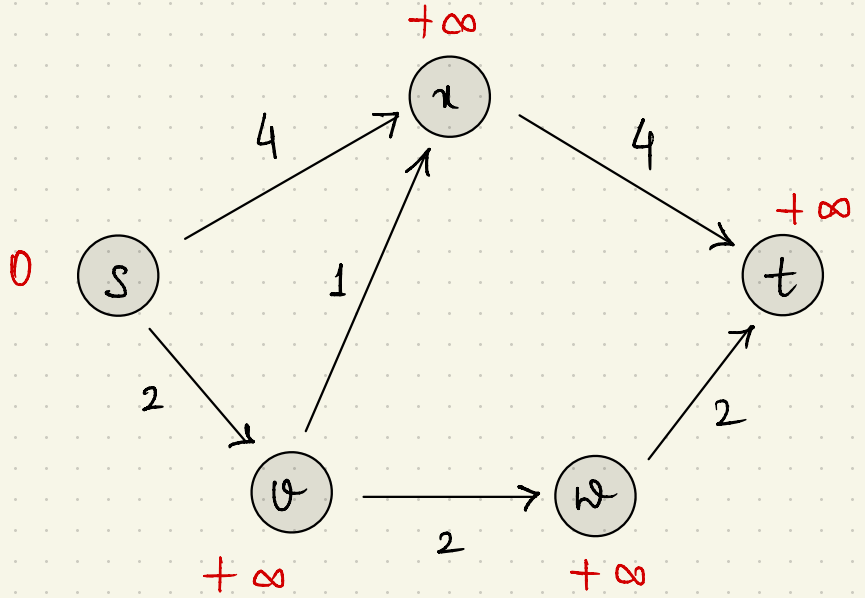
$i = 0$



// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$

EXAMPLE

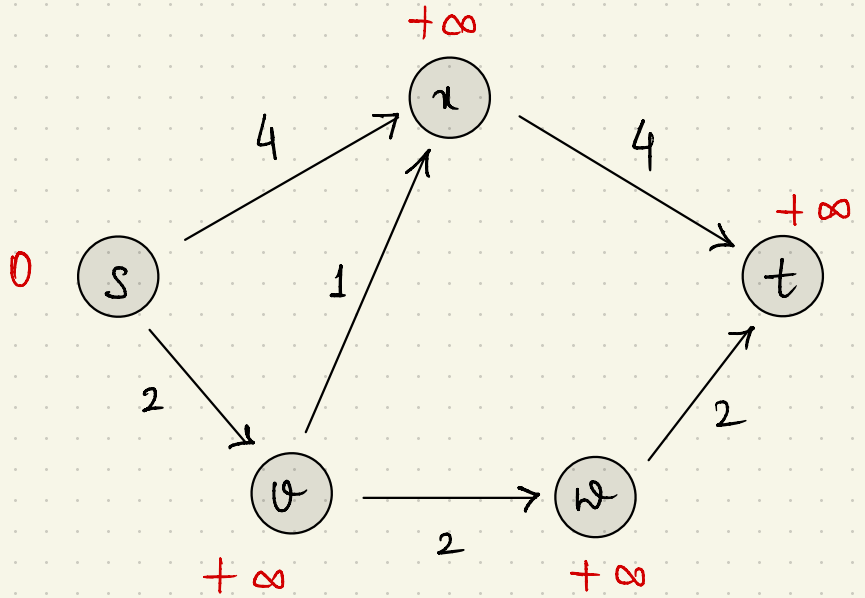


$i = 0$

// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$

EXAMPLE



$i = 0$

$i = 1$

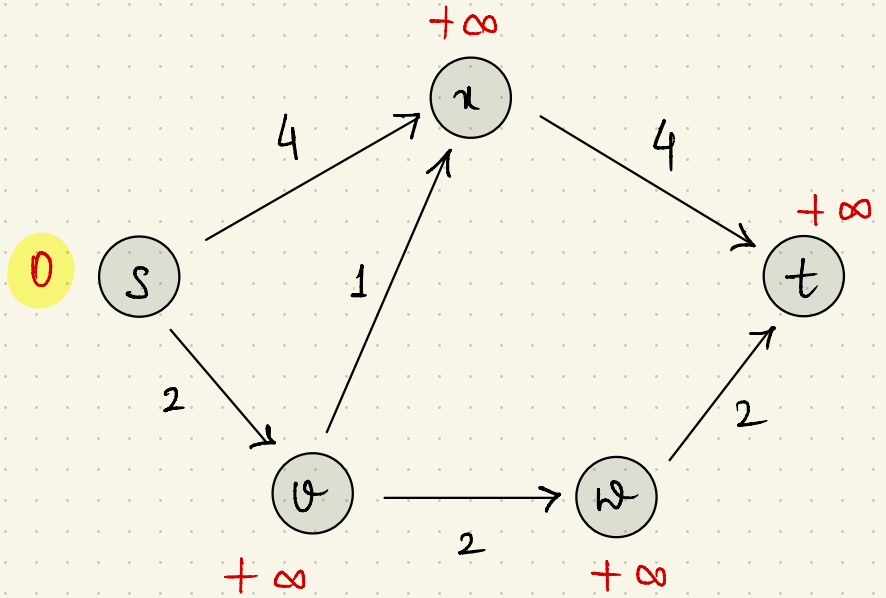
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE



i = 0

i = 1

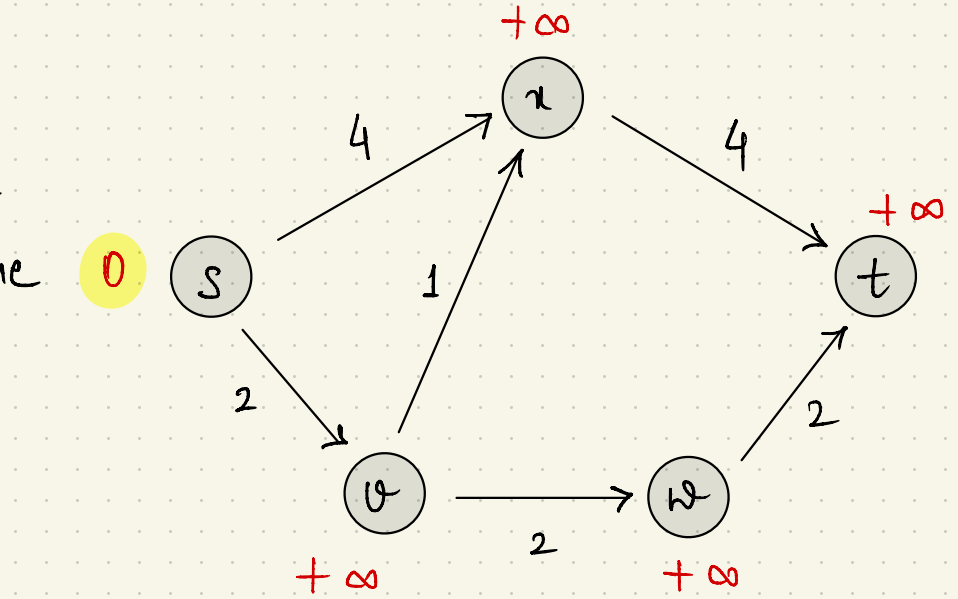
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i,v] = \min \begin{cases} A[i-1,v] \\ \min_{(w,v) \in E} A[i-1,w] + l_{w,v} \end{cases}$$

EXAMPLE



$i = 0$

$i = 1$

⇒ stays 0

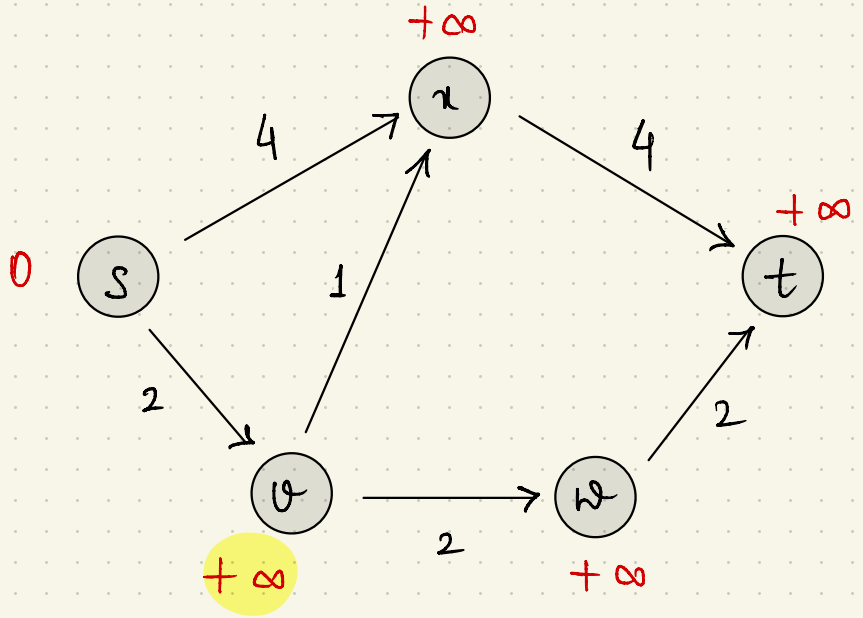
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE



i = 0

i = 1

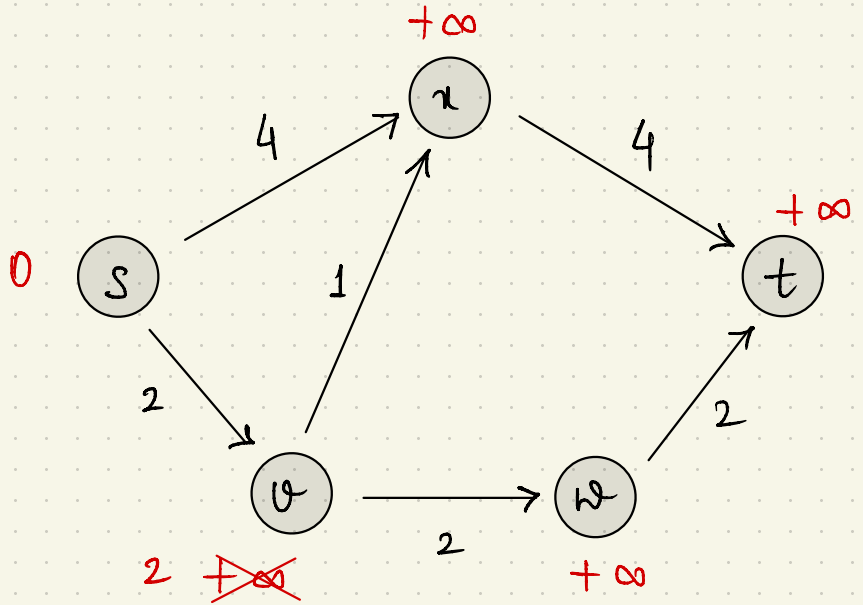
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE



i = 0

i = 1

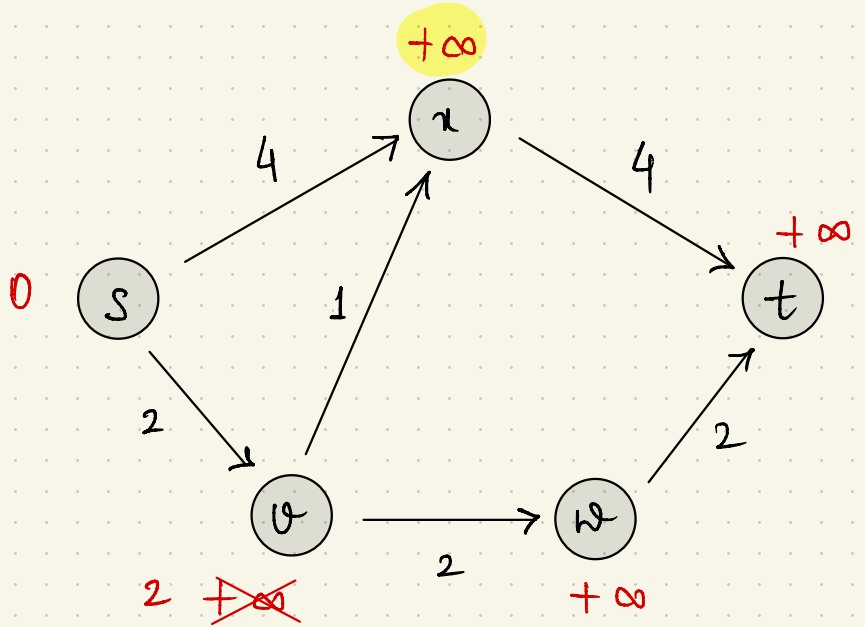
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE



i = 0

i = 1

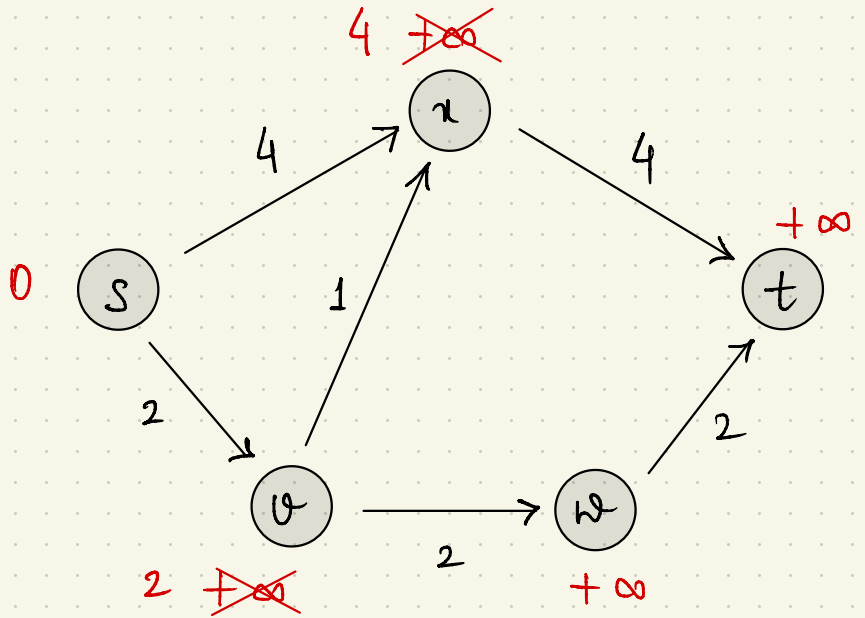
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i,v] = \min \begin{cases} A[i-1,v] \\ \min_{(w,v) \in E} A[i-1,w] + l_{w,v} \end{cases}$$

EXAMPLE



i = 0

i = 1

// recurrence

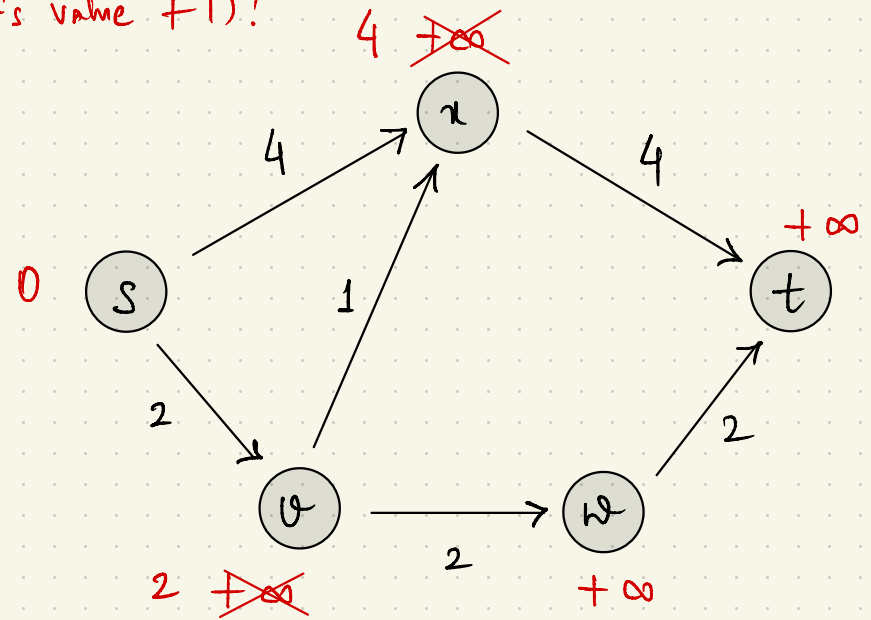
for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE

Why not 3 (v's value +1)?



i = 0

i = 1

// recurrence

for $i \in \{1, 2, \dots, n-1\}$

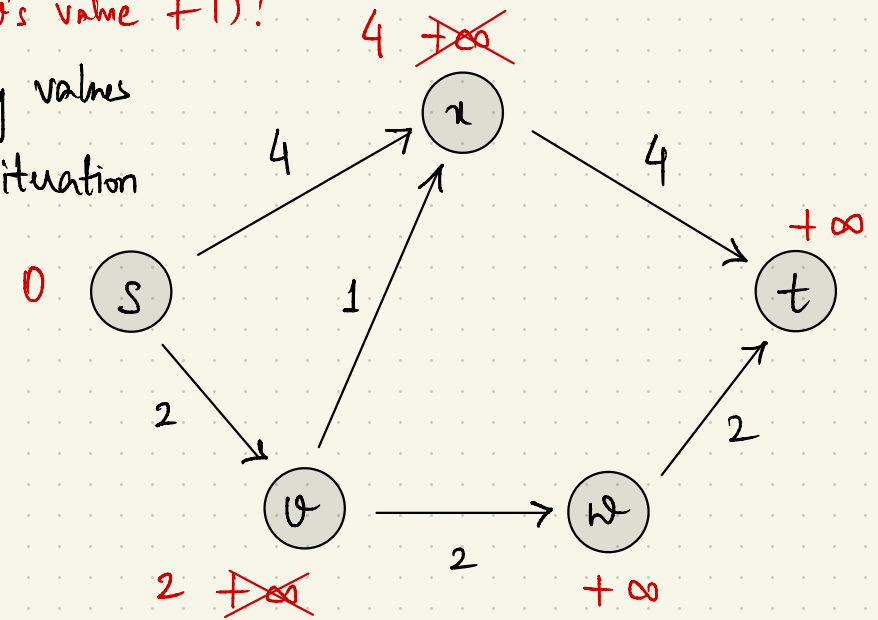
for each $v \in V$

$$A[i,v] = \min \begin{cases} A[i-1,v] \\ \min_{(w,v) \in E} A[i-1,w] + l_{w,v} \end{cases}$$

EXAMPLE

Why not 3 (v's value +1)?

Updates using values of previous iteration



i = 0

i = 1

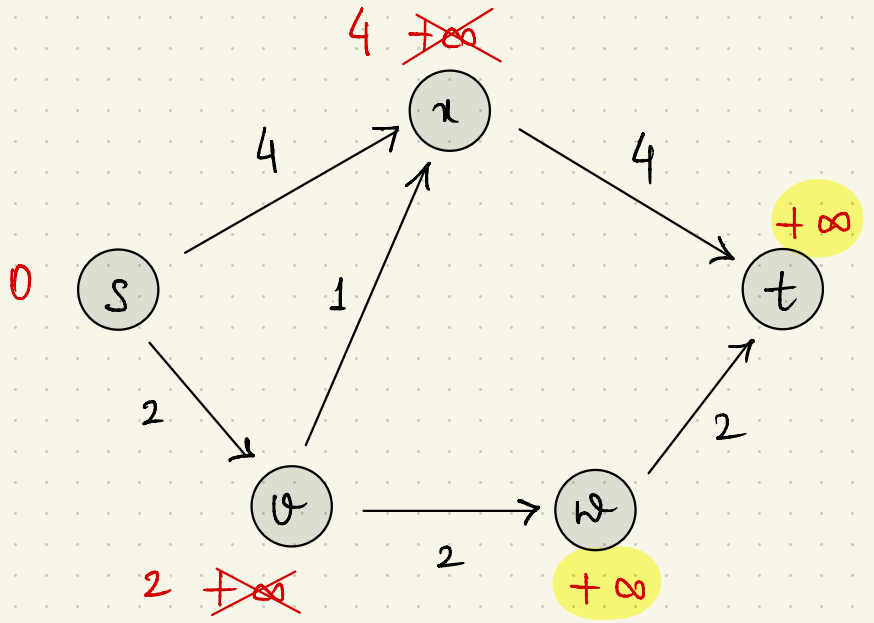
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE



i = 0

i = 1

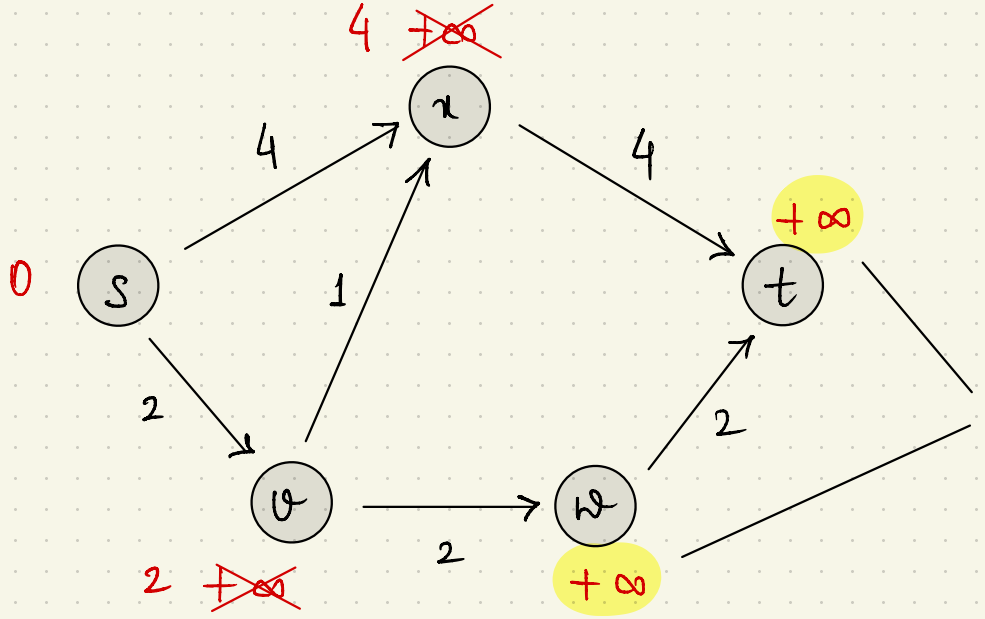
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i,v] = \min \begin{cases} A[i-1,v] \\ \min_{(w,v) \in E} A[i-1,w] + l_{w,v} \end{cases}$$

EXAMPLE



i = 0

i = 1

stay unchanged

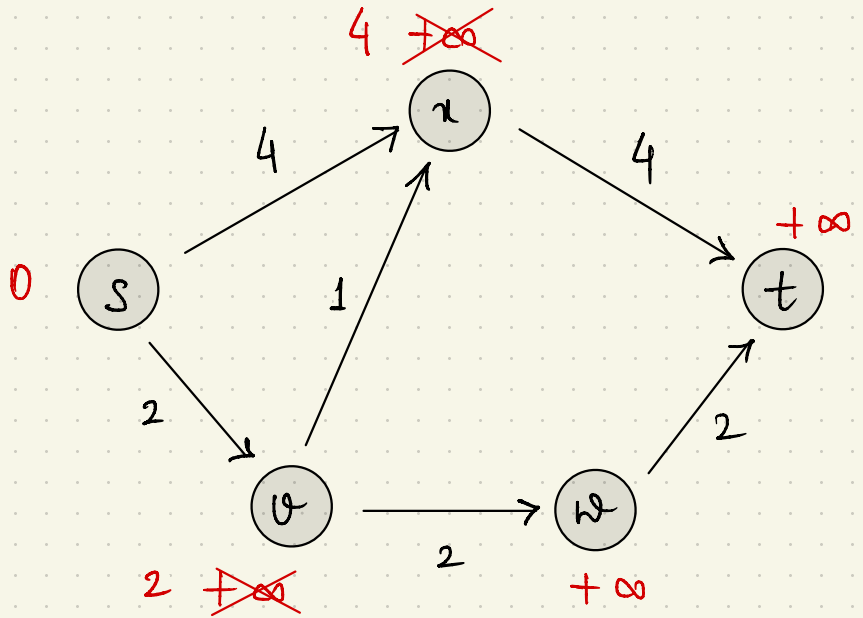
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE



i = 0

i = 1

i = 2

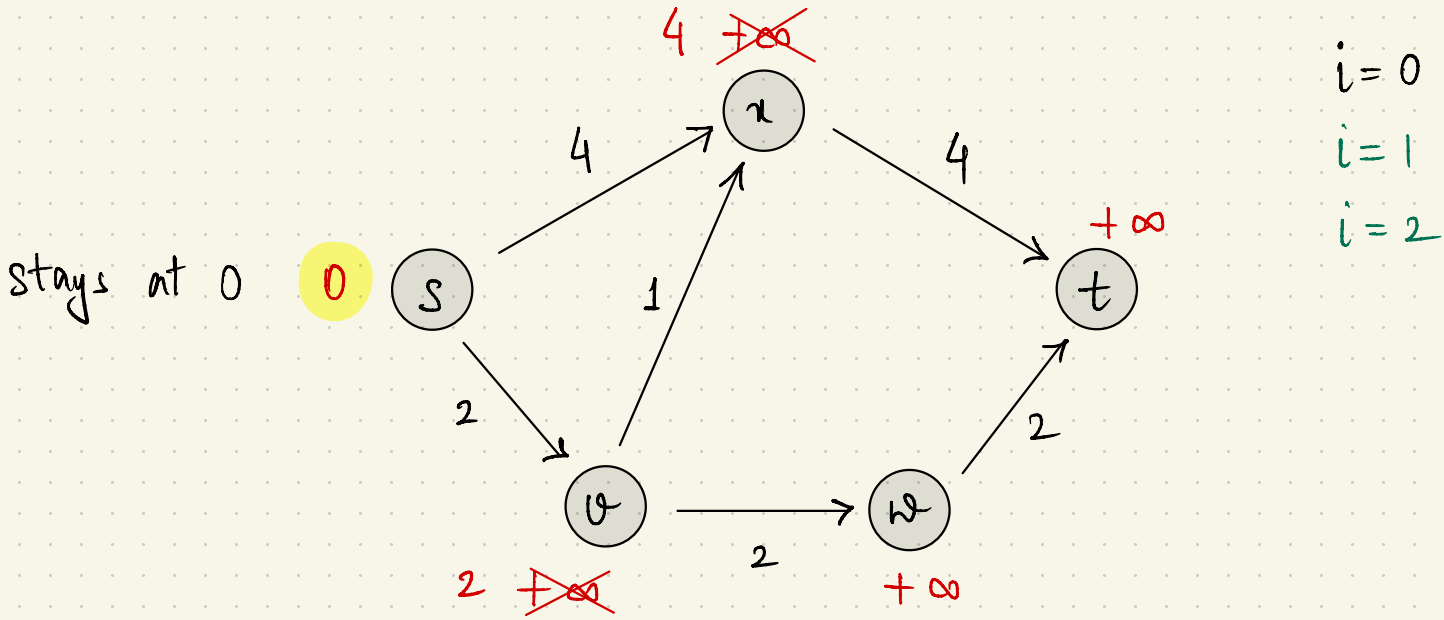
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i,v] = \min \begin{cases} A[i-1,v] \\ \min_{(w,v) \in E} A[i-1,w] + l_{w,v} \end{cases}$$

EXAMPLE



i = 0

i = 1

i = 2

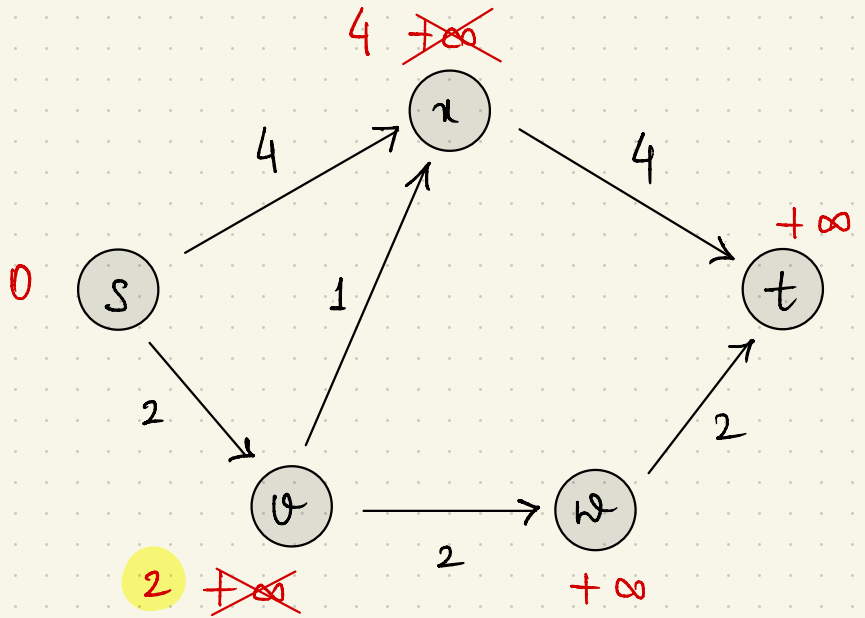
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE



i = 0

i = 1

i = 2

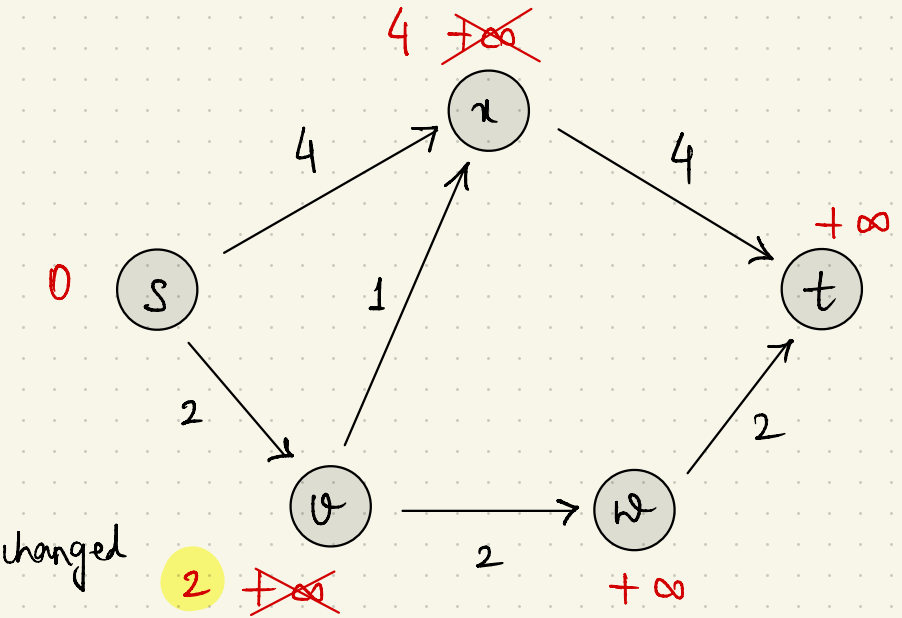
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i,v] = \min \begin{cases} A[i-1,v] \\ \min_{(w,v) \in E} A[i-1,w] + l_{w,v} \end{cases}$$

EXAMPLE



$i = 0$

$i = 1$

$i = 2$

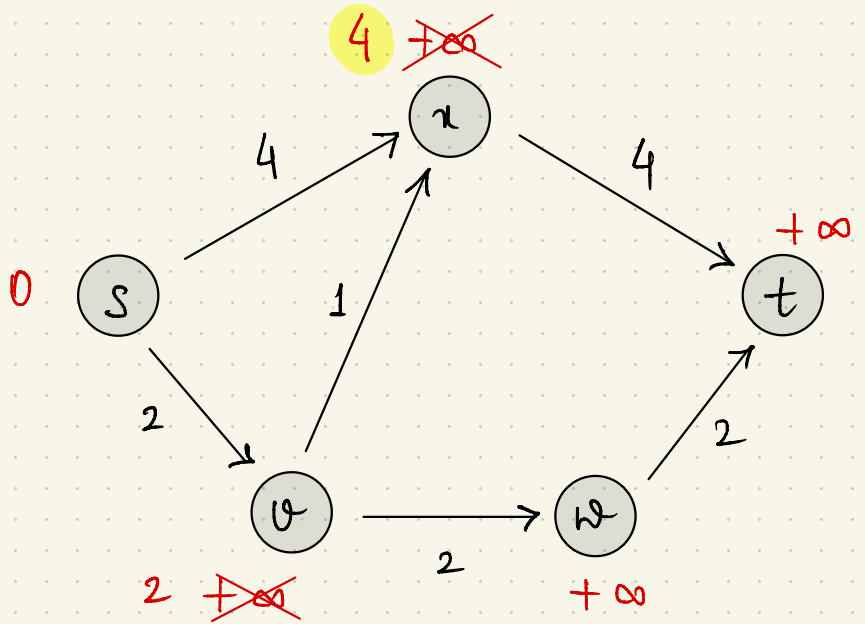
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE



i = 0

i = 1

i = 2

// recurrence

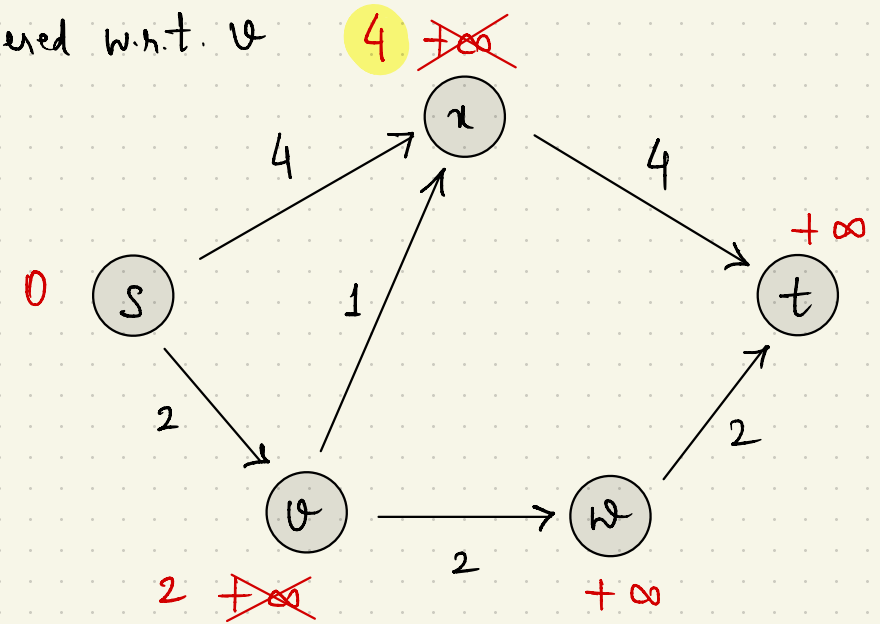
for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i,v] = \min \begin{cases} A[i-1,v] \\ \min_{(w,v) \in E} A[i-1,w] + l_{w,v} \end{cases}$$

EXAMPLE

Case II triggered w.r.t. v



$i = 0$

$i = 1$

$i = 2$

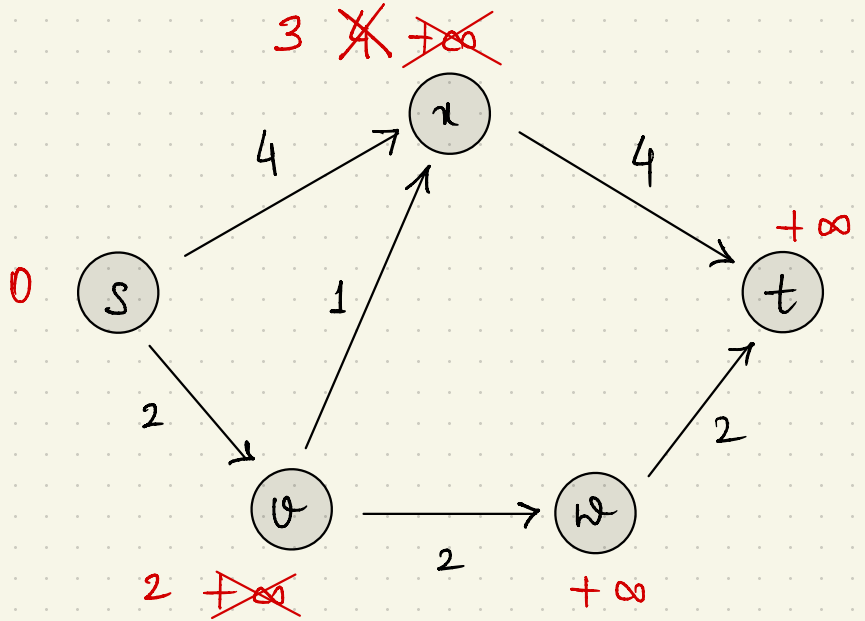
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE



$i = 0$

$i = 1$

$i = 2$

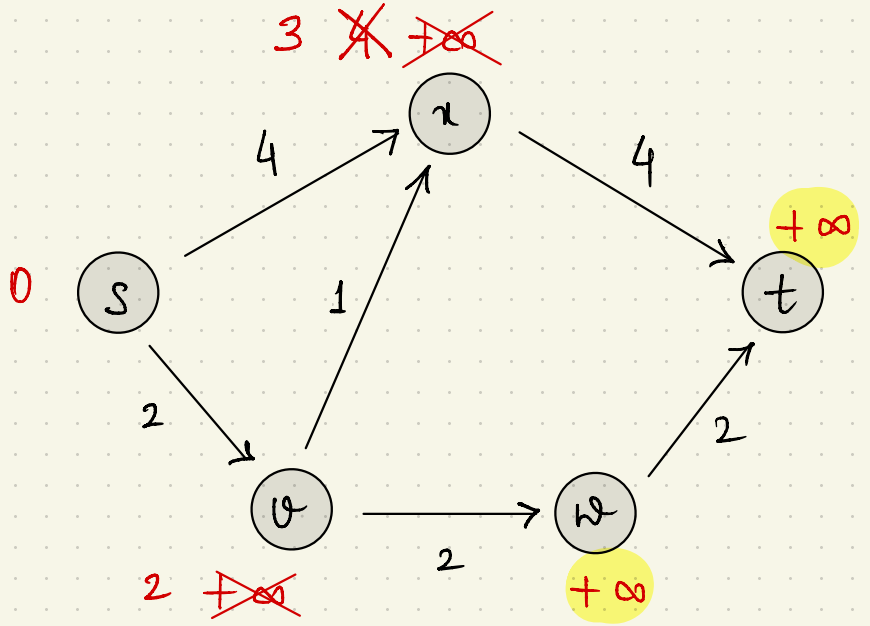
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE



$i = 0$

$i = 1$

$i = 2$

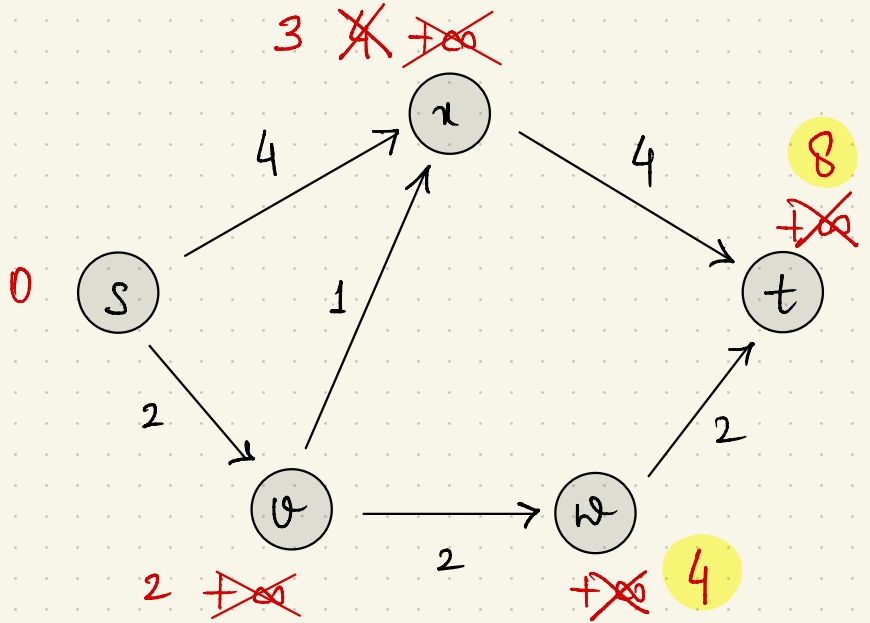
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE



$i = 0$

$i = 1$

$i = 2$

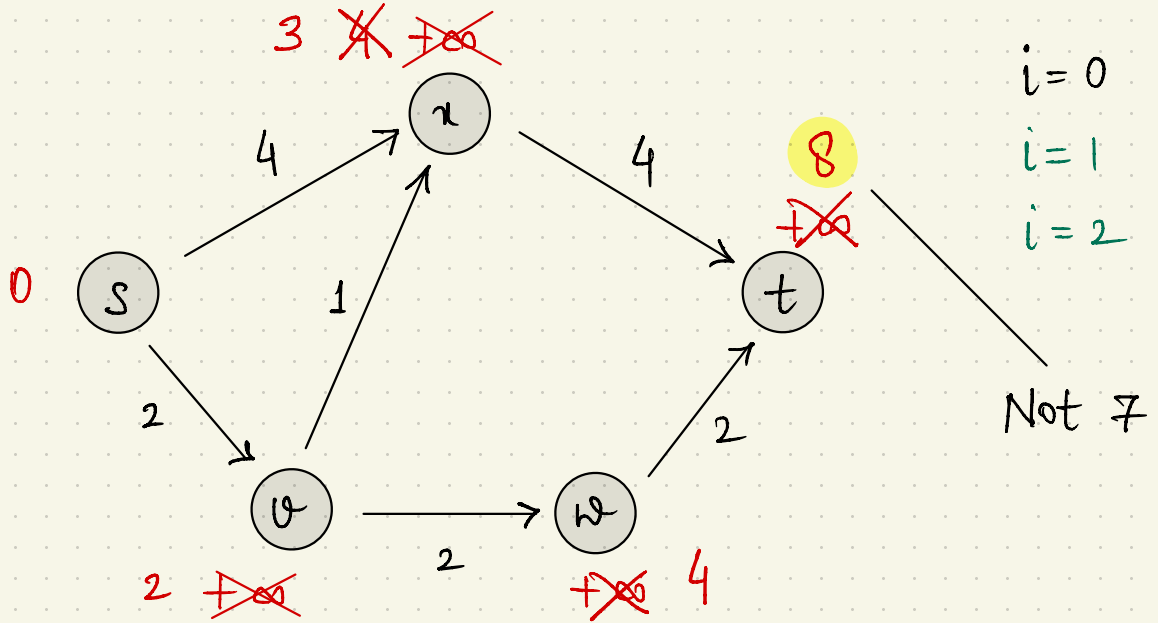
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE



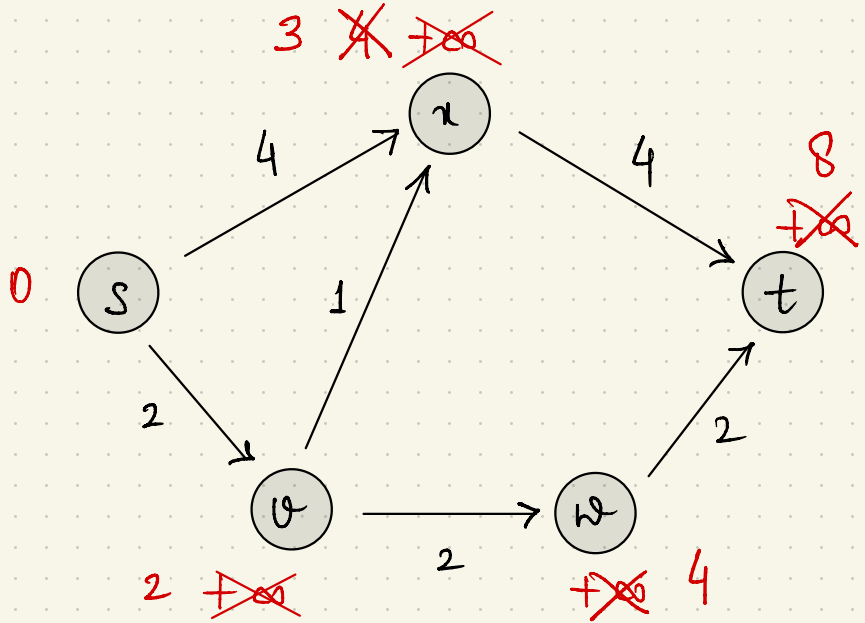
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE



$i = 0$

$i = 1$

$i = 2$

$i = 3$

// recurrence

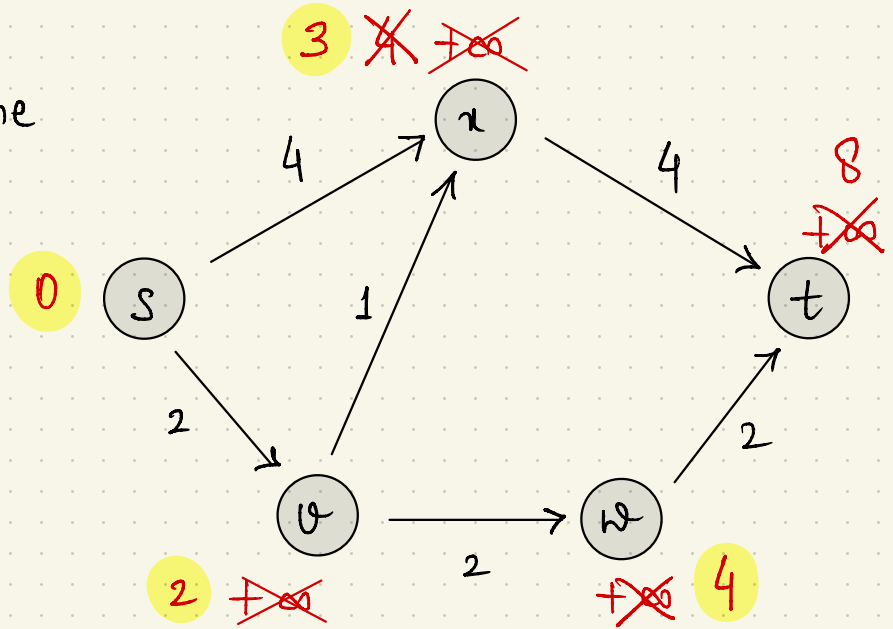
for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE

stay the same



i = 0

i = 1

i = 2

i = 3

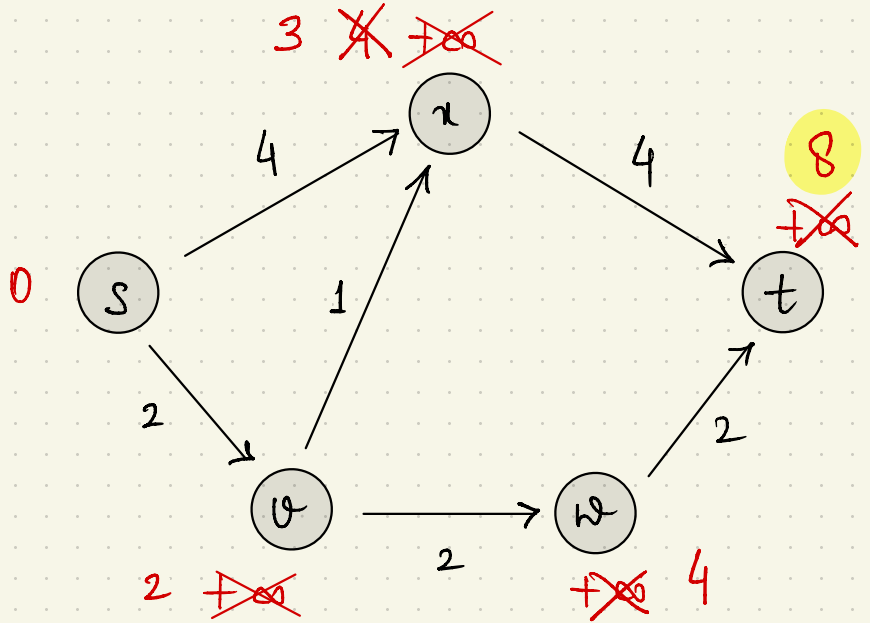
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i,v] = \min \begin{cases} A[i-1,v] \\ \min_{(w,v) \in E} A[i-1,w] + l_{w,v} \end{cases}$$

EXAMPLE



$i = 0$

$i = 1$

$i = 2$

$i = 3$

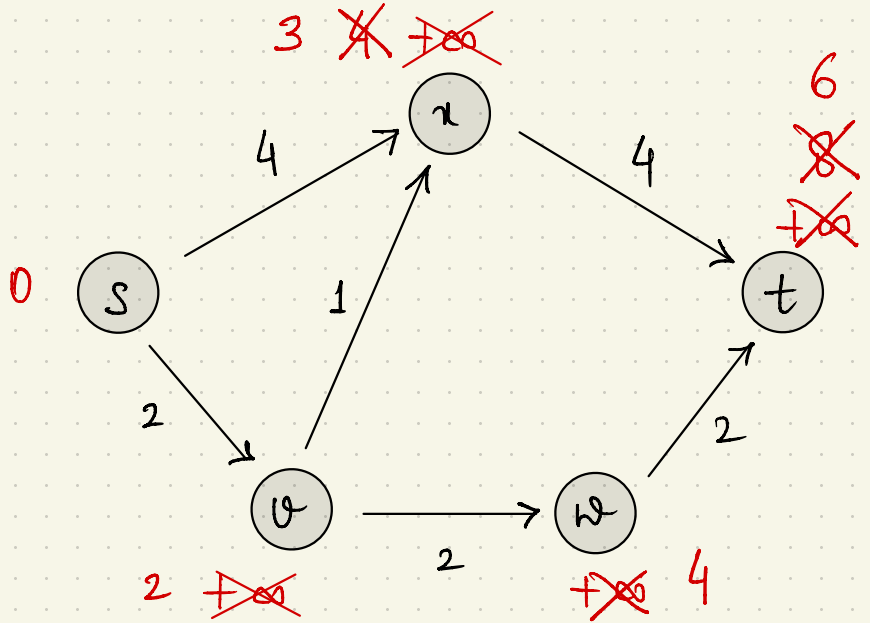
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE



$i = 0$

$i = 1$

$i = 2$

$i = 3$

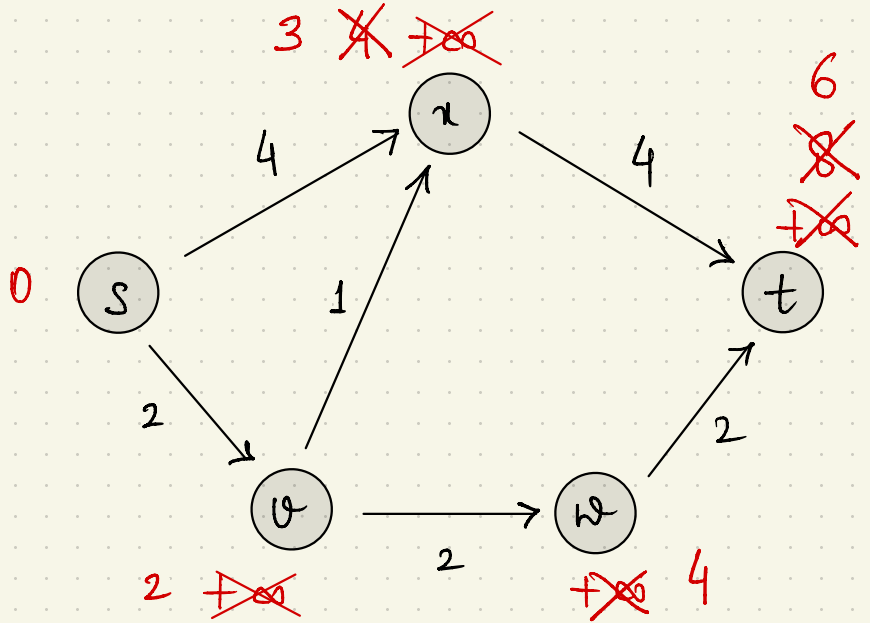
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

EXAMPLE



i = 0

i = 1

i = 2

i = 3

i = 4

// recurrence

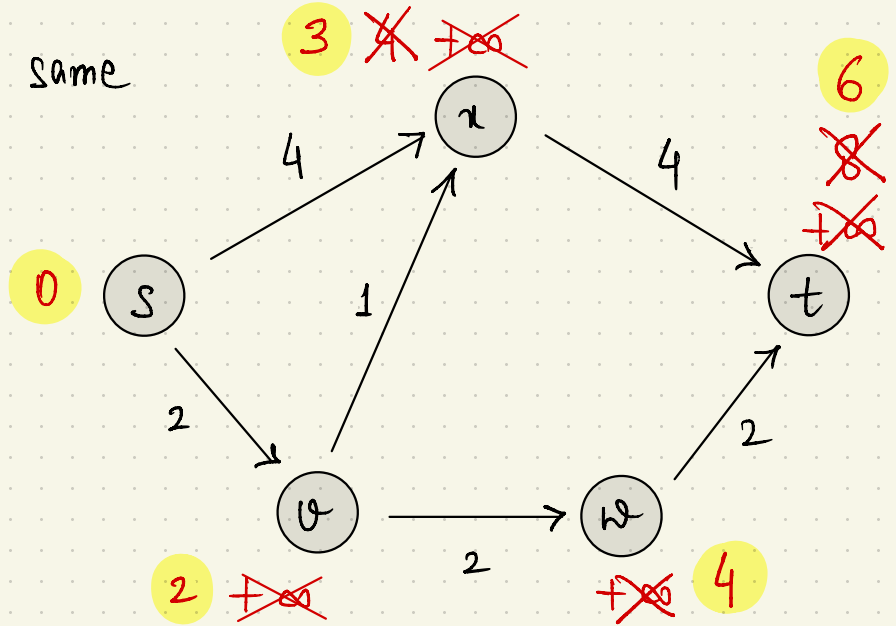
for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i,v] = \min \begin{cases} A[i-1,v] \\ \min_{(w,v) \in E} A[i-1,w] + l_{w,v} \end{cases}$$

EXAMPLE

all stay the same



i = 0

i = 1

i = 2

i = 3

i = 4

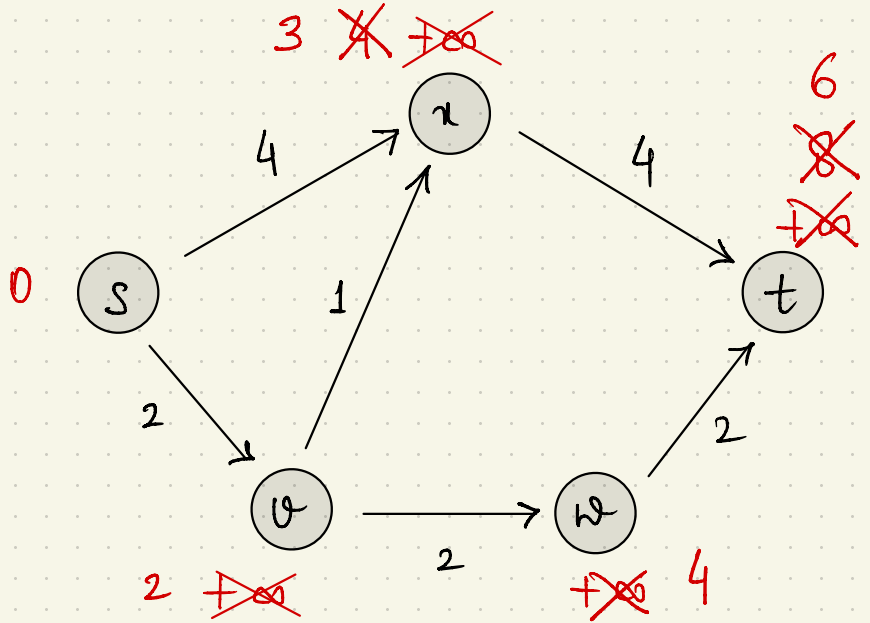
// recurrence

for $i \in \{1, 2, \dots, n-1\}$

for each $v \in V$

$$A[i,v] = \min \begin{cases} A[i-1,v] \\ \min_{(w,v) \in E} A[i-1,w] + l_{w,v} \end{cases}$$

EXAMPLE



i = 0

i = 1

i = 2

i = 3

i = 4

done!

BELLMAN-FORD ALGORITHM

A := $n \times n$ dimensional array // budget $\times$ destinations

// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$

// recurrence

for $i \in \{1, 2, \ldots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

BELLMAN-FORD ALGORITHM

A := $n \times n$ dimensional array // budget $\times$ destinations

// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$

// recurrence

Running time: ?

for $i \in \{1, 2, \ldots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

BELLMAN-FORD ALGORITHM

A := $n \times n$ dimensional array // budget $\times$ destinations

// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$

// recurrence

Running time: $O(mn)$

for $i \in \{1, 2, \ldots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

BELLMAN-FORD ALGORITHM

A := $n \times n$ dimensional array // budget $\times$ destinations

// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$

// recurrence

for $i \in \{1, 2, \ldots, n-1\}$

  for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + \ell_{w,v} \end{cases}$$

Running time: $O(mn)$

$$O\left(n \cdot \left(1 + \sum_{v} \text{in-degree}(v)\right)\right)$$

BELLMAN-FORD ALGORITHM

A := $n \times n$ dimensional array // budget $\times$ destinations

// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$

// recurrence

for $i \in \{1, 2, \ldots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

Running time: $O(mn)$

$1 + \text{in-deg}(v)$ subproblems for vertex v

BELLMAN-FORD ALGORITHM

A := $n \times n$ dimensional array // budget $\times$ destinations

// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$

// recurrence

for $i \in \{1, 2, \ldots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

Running time: $O(mn)$

$$O\left(n \cdot \left(\sum_{v} 1 + \text{in-deg}(v)\right)\right)$$

BELLMAN-FORD ALGORITHM

A := $n \times n$ dimensional array // budget $\times$ destinations

// base cases

$A[0, s] := 0$, $A[0, v] := +\infty$ for all $v \neq s$

// recurrence

for $i \in \{1, 2, \ldots, n-1\}$

for each $v \in V$

$$A[i, v] = \min \begin{cases} A[i-1, v] \\ \min_{(w,v) \in E} A[i-1, w] + l_{w,v} \end{cases}$$

Running time: $O(mn)$

$$O\left(n \cdot \left(\sum_{v} 1 + \text{in-deg}(v)\right)\right)$$

$$= O(n^2 + mn) = O(mn)$$

DETECTING NEGATIVE CYCLES

DETECTING NEGATIVE CYCLES

Claim: G has no negative cycle if and only if in the extended Bellman-Ford algorithm

$$A[n-1, v] = A[n, v] \quad \text{for all } v \in V.$$

DETECTING NEGATIVE CYCLES

Claim: G has no negative cycle if and only if in the extended Bellman-Ford algorithm

$$A[n-1, v] = A[n, v] \quad \text{for all } v \in V.$$

run the algorithm for one more iteration

$$i \in \{1, 2, \cdots, n-1, n\}$$

DETECTING NEGATIVE CYCLES

Claim: G has no negative cycle if and only if in the extended Bellman-Ford algorithm

$$A[n-1, v] = A[n, v] \quad \text{for all } v \in V.$$

Consequence: Can check existence of a negative cycle by running Bellman-Ford for an extra iteration

$\Rightarrow$ running time still $O(mn)$.

DETECTING NEGATIVE CYCLES

Claim: G has no negative cycle if and only if in the extended Bellman-Ford algorithm

$$A[n-1, v] = A[n, v] \quad \text{for all } v \in V.$$

NOTE: Claim is FALSE when the negative cycle is not reachable from s.



DETECTING NEGATIVE CYCLES

Claim : G has no negative cycle reachable from s

if and only if

in the extended Bellman-Ford algorithm

$$A[n-1, v] = A[n, v] \quad \text{for all } v \in V.$$

DETECTING NEGATIVE CYCLES

Claim : G has no negative cycle reachable from s

if and only if

in the extended Bellman-Ford algorithm

$$A[n-1, v] = A[n, v] \quad \text{for all } v \in V.$$



Add auxiliary vertex to detect negative cycle, reachable from s or otherwise.

PROOF OF CLAIM

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$

in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$ in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Rightarrow$) If no negative cycle reachable from s, then claim follows by correctness of standard Bellman-Ford.

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$ in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$)

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$ in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$) Assume there is a path from s to every other vertex.

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$ in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$) Assume there is a path from s to every other vertex.

[Exercise: Try the proof without this assumption]

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$ in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$) Assume there is a path from s to every other vertex.

Let $d(v)$ denote the common finite value of $A[n-1, v]$ and $A[n, v]$.

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$ in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$) Assume there is a path from s to every other vertex.

Let $d(v)$ denote the common finite value of $A[n-1, v]$ and $A[n, v]$.

Recall: $$A[n, v] = \min \left\{ A[n-1, v], \min_{(w, v) \in E} A[n-1, w] + \ell_{w,v} \right\}.$$

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$

in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$) Assume there is a path from s to every other vertex.

Let $d(v)$ denote the common finite value of $A[n-1, v]$ and $A[n, v]$.

Recall: $A[n, v] = \min \left\{ A[n-1, v], \min_{(w,v) \in E} A[n-1, w] + \ell_{w,v} \right\}$.

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$ in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$) Assume there is a path from s to every other vertex.

Let $d(v)$ denote the common finite value of $A[n-1, v]$ and $A[n, v]$.

Recall: $A[n, v] = \min \left\{ A[n-1, v], \min_{(w,v) \in E} A[n-1, w] + \ell_{w,v} \right\}$.

$A[n, v]$: $d(v)$ — $A[n-1, w]$: $d(w)$

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$ in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$) Assume there is a path from s to every other vertex. Let $d(v)$ denote the common finite value of $A[n-1, v]$ and $A[n, v]$.

Recall: $$d(v) = \min \left\{ A[n-1, v], \min_{(w,v) \in E} d(w) + \ell_{w,v} \right\}.$$

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$ in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$) Assume there is a path from s to every other vertex.

Let $d(v)$ denote the common finite value of $A[n-1, v]$ and $A[n, v]$.

Recall: $$d(v) = \min \left\{ A[n-1, v], \min_{(w,v) \in E} d(w) + \ell_{w,v} \right\}.$$

$$\Rightarrow \quad d(v) \leq d(w) + \ell_{w,v} \quad \text{for all } (w, v) \in E.$$

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$ in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$) Assume there is a path from s to every other vertex.

Let $d(v)$ denote the common finite value of $A[n-1, v]$ and $A[n, v]$.

Recall: $$d(v) = \min \left\{ A[n-1, v], \min_{(w,v) \in E} d(w) + \ell_{w,v} \right\}.$$

$$\Rightarrow \quad d(v) \leq d(w) + \ell_{w,v} \quad \text{for all } (w,v) \in E.$$

$$\Rightarrow \quad d(v) - d(w) \leq \ell_{w,v} \quad \text{for all } (w,v) \in E$$

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$

in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$) Assume there is a path from s to every other vertex.

Let $d(v)$ denote the common finite value of $A[n-1, v]$ and $A[n, v]$.

Then, $d(v) - d(w) \leq l_{w,v}$ for all $(w, v) \in E$

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$ in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$) Assume there is a path from s to every other vertex.

Let $d(v)$ denote the common finite value of $A[n-1, v]$ and $A[n, v]$.

Then, $d(v) - d(w) \leq l_{w,v}$ for all $(w, v) \in E$

Consider an arbitrary cycle C.

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$ in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$) Assume there is a path from s to every other vertex.

Let $d(v)$ denote the common finite value of $A[n-1, v]$ and $A[n, v]$.

Then, $d(v) - d(w) \leq l_{w,v}$ for all $(w, v) \in E$

Consider an arbitrary cycle C.

$$\sum_{(w,v) \in C} l_{w,v}$$

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$ in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$) Assume there is a path from s to every other vertex.

Let $d(v)$ denote the common finite value of $A[n-1, v]$ and $A[n, v]$.

Then, $d(v) - d(w) \leq l_{w,v}$ for all $(w, v) \in E$

Consider an arbitrary cycle C.

$$\sum_{(w,v) \in C} l_{w,v} \geqslant \sum_{(w,v) \in C} d(v) - d(w)$$

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$ in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$) Assume there is a path from s to every other vertex.

Let $d(v)$ denote the common finite value of $A[n-1, v]$ and $A[n, v]$.

Then, $d(v) - d(w) \leq l_{w,v}$ for all $(w, v) \in E$

Consider an arbitrary cycle C.

$$\sum_{(w,v) \in C} l_{w,v} \geqslant \sum_{(w,v) \in C} d(v) - d(w) = 0.$$

PROOF OF CLAIM

Claim: G has no negative cycle reachable from s $\iff$ in extended Bellman-Ford algorithm, $A[n-1, v] = A[n, v]$ $\forall v$.

Proof: ($\Leftarrow$) Assume there is a path from s to every other vertex.

Let $d(v)$ denote the common finite value of $A[n-1, v]$ and $A[n, v]$.

Then, $d(v) - d(w) \leq l_{w,v}$ for all $(w, v) \in E$

Consider an arbitrary cycle C.

$$\sum_{(w,v) \in C} l_{w,v} \geqslant \sum_{(w,v) \in C} d(v) - d(w) = 0.$$

