

# COL 351 : ANALYSIS & DESIGN OF ALGORITHMS

## LECTURE 13

### GRAPH ALGORITHMS VI : DIJKSTRA'S ALGORITHM

AUG 21, 2026

|

ROHIT VAISH

# SINGLE - SOURCE SHORTEST PATH PROBLEM



# 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:**  $\text{dist}(s, v)$  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:**  $\text{dist}(s, v)$  for every vertex  $v \in V$ .

length of the shortest path from  $s$  to  $v$

# 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:**  $\text{dist}(s, v)$  for every vertex  $v \in V$ .

length of the shortest path from  $s$  to  $v$

sum of edge lengths

# SINGLE - SOURCE SHORTEST PATH PROBLEM

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

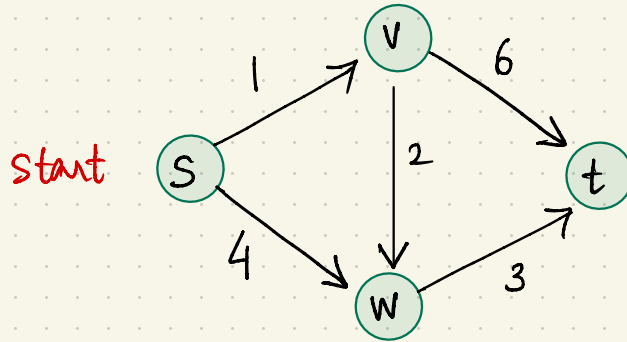
**output:**  $\text{dist}(s, v)$  for every vertex  $v \in V$ .

length of the shortest path from  $s$  to  $v$  if  $s \rightsquigarrow v$  path exists  
 $\infty$  otherwise

# SINGLE - SOURCE SHORTEST PATH PROBLEM

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

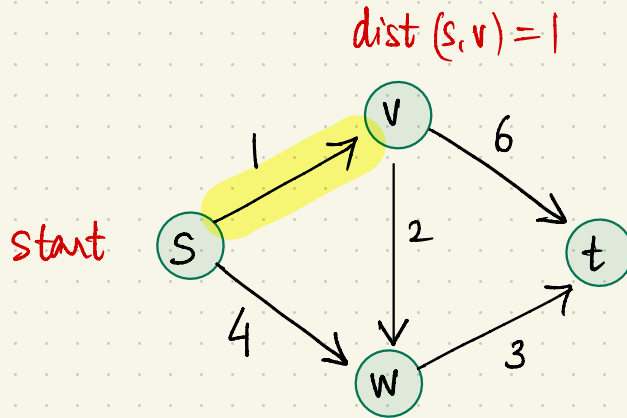
**output:**  $\text{dist}(s, v)$  for every vertex  $v \in V$ .



# SINGLE - SOURCE SHORTEST PATH PROBLEM

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

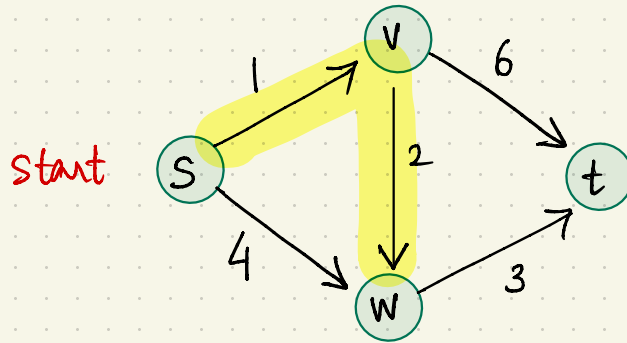
**output:**  $\text{dist}(s, v)$  for every vertex  $v \in V$ .



# 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:**  $\text{dist}(s, v)$  for every vertex  $v \in V$ .

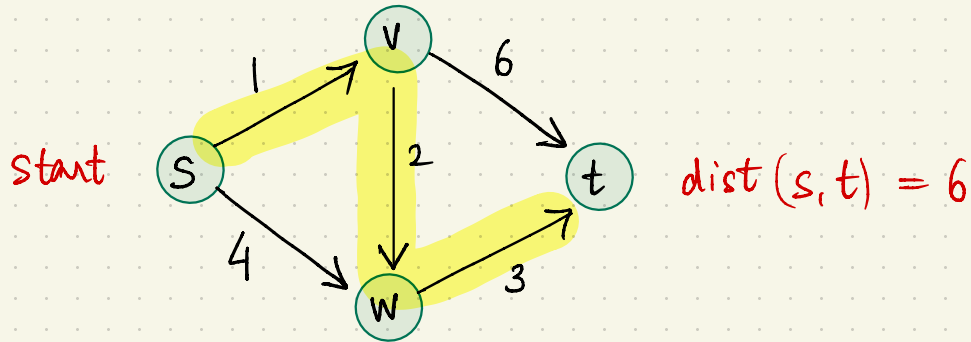


$\text{dist}(s, w) = 3$

# 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:**  $\text{dist}(s, v)$  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:**  $\text{dist}(s, v)$  for every vertex  $v \in V$ .

For negative lengths, use Bellman-Ford algorithm.

# SINGLE - SOURCE SHORTEST PATH PROBLEM



Doesn't BFS already compute shortest paths?

# SINGLE - SOURCE SHORTEST PATH PROBLEM



Doesn't BFS already compute shortest paths?

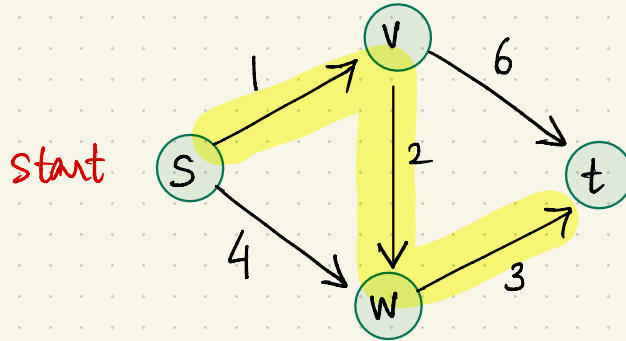
Yes, if length of a path = number of edges in it  
( $l_e = 1$  for every edge  $e$ )

# SINGLE - SOURCE SHORTEST PATH PROBLEM



Doesn't BFS already compute shortest paths?

Yes, if length of a path = **number** of edges in it  
( $l_e = 1$  for every edge  $e$ )



BFS fails

# SINGLE - SOURCE SHORTEST PATH PROBLEM

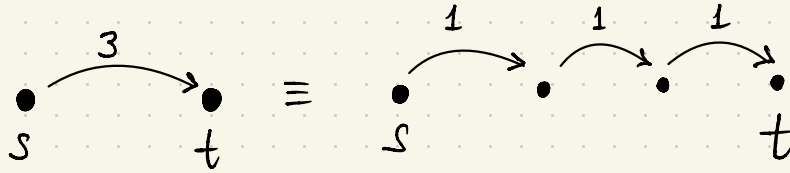


Isn't an edge with length  $l_e$  the same as  $l_e$  unit length edges?

# SINGLE - SOURCE SHORTEST PATH PROBLEM



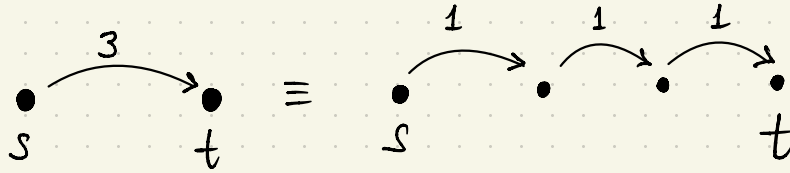
Isn't an edge with length  $l_e$  the same as  $l_e$  unit length edges?



# SINGLE - SOURCE SHORTEST PATH PROBLEM



Isn't an edge with length  $l_e$  the same as  $l_e$  unit length edges?

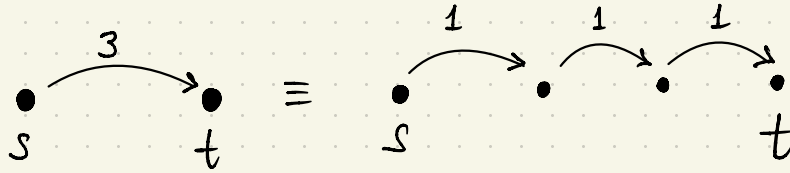


This representation is **faithful** but **wasteful**.

# SINGLE - SOURCE SHORTEST PATH PROBLEM



Isn't an edge with length  $l_e$  the same as  $l_e$  unit length edges?



This representation is **faithful** but **wasteful**.

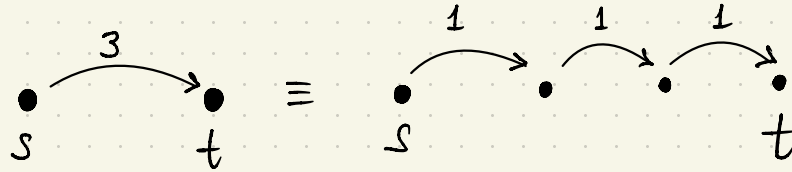
BFS time is linear in the size of the **blown up graph**



# SINGLE - SOURCE SHORTEST PATH PROBLEM



Isn't an edge with length  $l_e$  the same as  $l_e$  unit length edges?



This representation is **faithful** but **wasteful**.

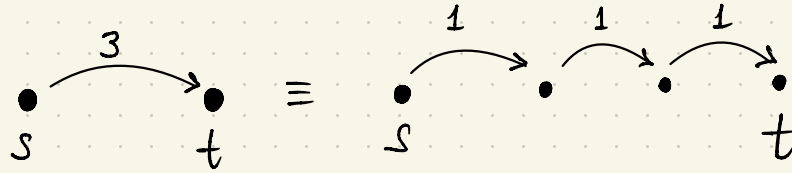
BFS time is linear in the size of the **blown up graph**

$$\sum_e l_e \gg n + m$$

# SINGLE - SOURCE SHORTEST PATH PROBLEM



Isn't an edge with length  $l_e$  the same as  $l_e$  unit length edges?



This representation is **faithful** but **wasteful**.

BFS time is linear in the size of the **blown up graph**

$$\sum_e l_e \gg n + m$$

**Solution** : Dijkstra's algorithm

# DIJKSTRA'S ALGORITHM

When  $\ell_e = 1$  for all edges  $e$ , Dijkstra's algorithm becomes breadth-first search.

# DIJKSTRA'S ALGORITHM

Initialize

$X := \{s\}$  vertices processed so far

# DIJKSTRA'S ALGORITHM

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

# DIJKSTRA'S ALGORITHM

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances



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

# DIJKSTRA'S ALGORITHM

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

$B[s] = \emptyset$  computed shortest paths

# DIJKSTRA'S ALGORITHM

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

$B[s] = \emptyset$  computed shortest paths

only to help with explanation



# DIJKSTRA'S ALGORITHM

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

$B[s] = \emptyset$  computed shortest paths

// main loop

while  $X \neq V$  need to grow  $X$

# DIJKSTRA'S ALGORITHM

Initialize

$X := \{s\}$  vertices processed so far

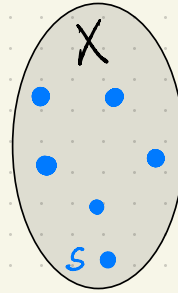
$A[s] = 0$  computed shortest-path distances

$B[s] = \emptyset$  computed shortest paths

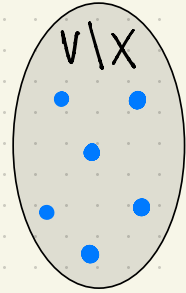
// main loop

while  $X \neq V$  need to grow  $X$

processed



not processed



# DIJKSTRA'S ALGORITHM

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

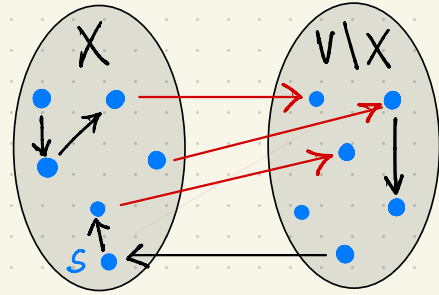
$B[s] = \emptyset$  computed shortest paths

// main loop

while  $X \neq V$  need to grow  $X$

processed

not processed



# DIJKSTRA'S ALGORITHM

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

$B[s] = \emptyset$  computed shortest paths

// main loop

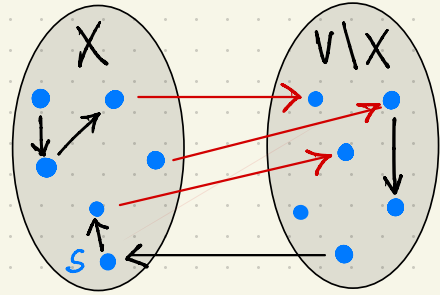
while  $X \neq V$  need to grow  $X$

among all edges  $(v, w) \in E$  with  $v \in X$  and  $w \notin X$ , pick one that minimizes

$$A[v] + l_{vw}$$

processed

not processed



# DIJKSTRA'S ALGORITHM

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

$B[s] = \emptyset$  computed shortest paths

// main loop

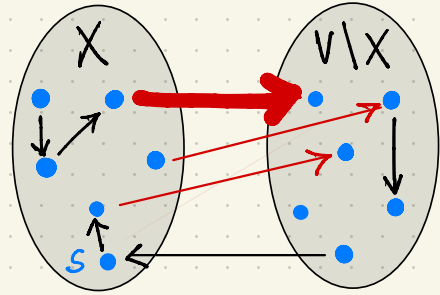
while  $X \neq V$  need to grow  $X$

among all edges  $(v, w) \in E$  with  $v \in X$  and  $w \notin X$ , pick one that minimizes

$$A[v] + l_{vw}$$

processed

not processed



# DIJKSTRA'S ALGORITHM

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

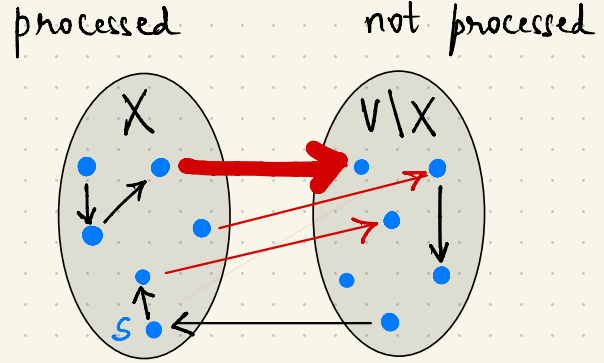
$B[s] = \emptyset$  computed shortest paths

// main loop

while  $X \neq V$  need to grow  $X$

among all edges  $(v, w) \in E$  with  $v \in X$  and  $w \notin X$ , pick one that minimizes

$A[v] + l_{vw}$  Dijkstra score



# DIJKSTRA'S ALGORITHM

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

$B[s] = \emptyset$  computed shortest paths

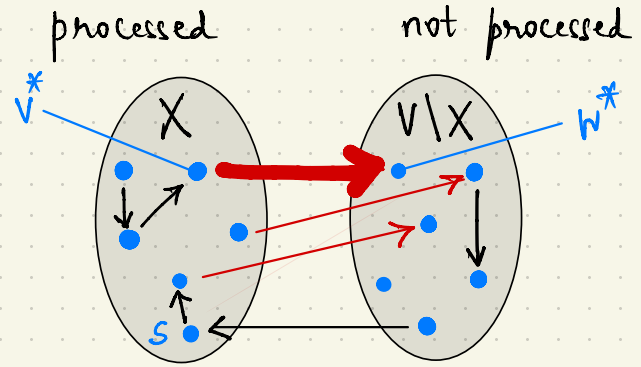
// main loop

while  $X \neq V$  need to grow  $X$

among all edges  $(v, w) \in E$  with  $v \in X$  and  $w \notin X$ , pick one that minimizes

$A[v] + l_{vw}$  Dijkstra score

Call it  $(v^*, w^*)$ .



# DIJKSTRA'S ALGORITHM

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

$B[s] = \emptyset$  computed shortest paths

// main loop

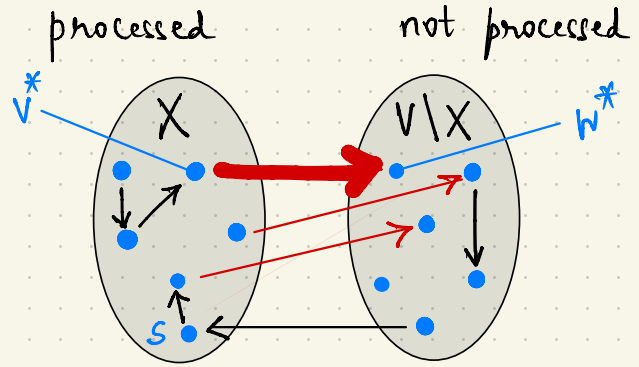
while  $X \neq V$  need to grow  $X$

among all edges  $(v, w) \in E$  with  $v \in X$  and  $w \notin X$ , pick one that minimizes

$A[v] + l_{vw}$  Dijkstra score

Call it  $(v^*, w^*)$ .

Add  $w^*$  to  $X$ .





# DIJKSTRA'S ALGORITHM

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

$B[s] = \emptyset$  computed shortest paths

// main loop

while  $X \neq V$  need to grow  $X$

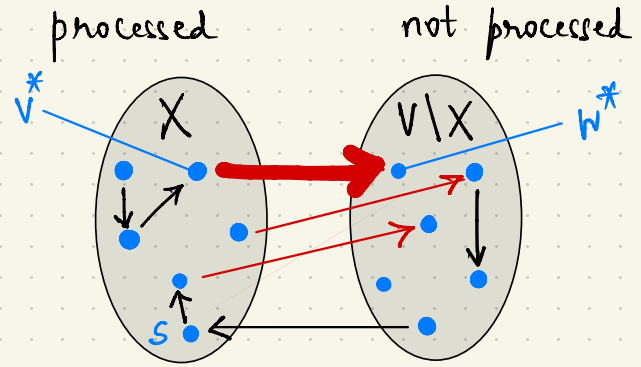
among all edges  $(v, w) \in E$  with  $v \in X$  and  $w \notin X$ , pick one that minimizes

$A[v] + l_{vw}$  Dijkstra score

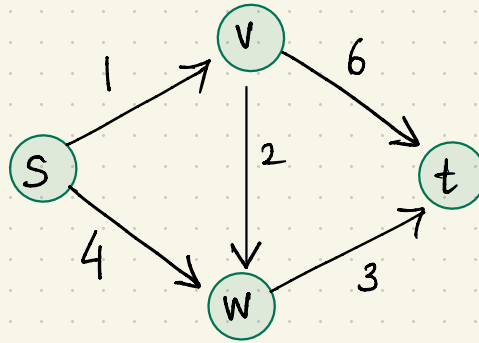
Call it  $(v^*, w^*)$ .

Add  $w^*$  to  $X$ .

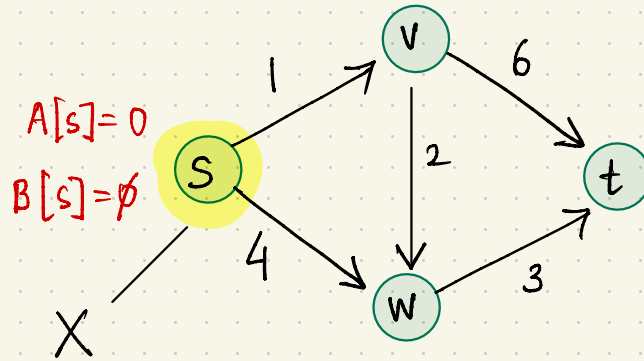
Set  $A[w^*] := A[v^*] + l_{v^*w^*}$  and  $B[w^*] := B[v^*] \cup (v^*, w^*)$ .



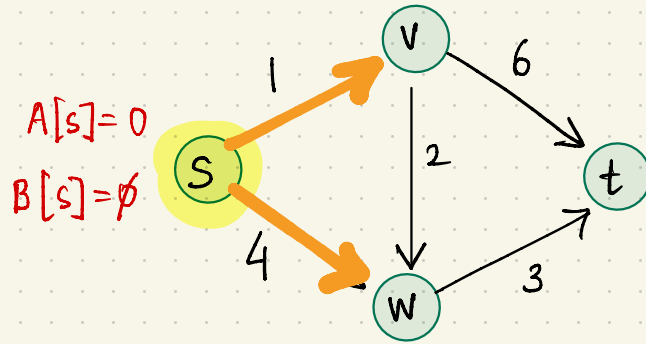
# DIJKSTRA'S ALGORITHM



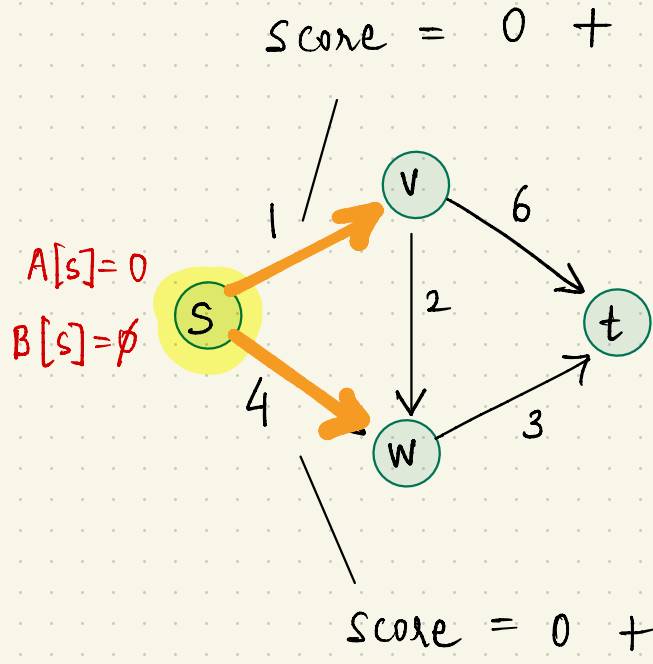
# DIJKSTRA'S ALGORITHM



# DIJKSTRA'S ALGORITHM

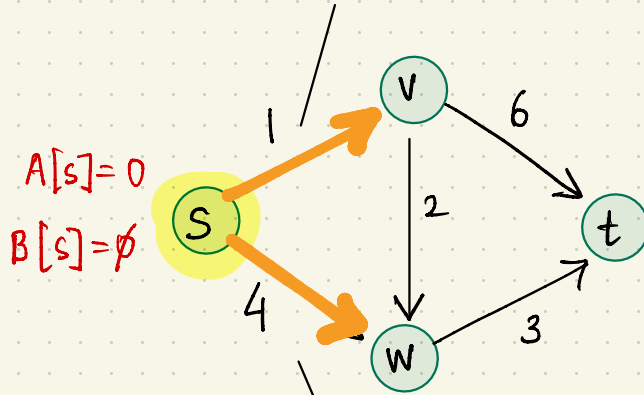


# DIJKSTRA'S ALGORITHM



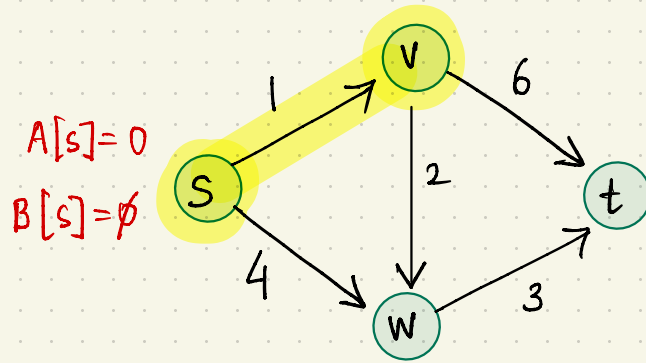
# DIJKSTRA'S ALGORITHM

$$\text{Score} = 0 + 1 \text{ (minimum)}$$

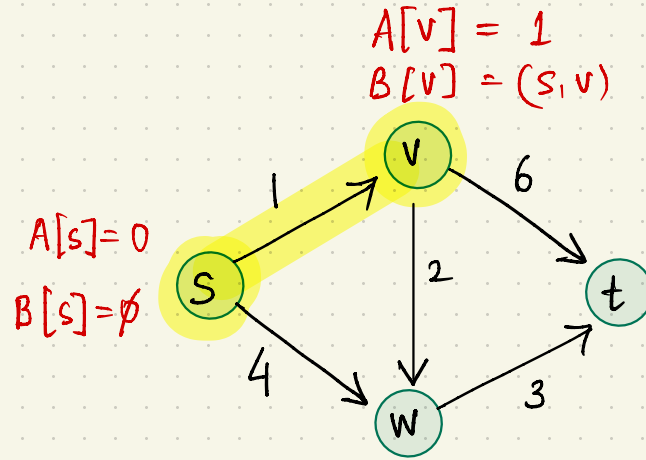


$$\text{Score} = 0 + 4$$

# DIJKSTRA'S ALGORITHM

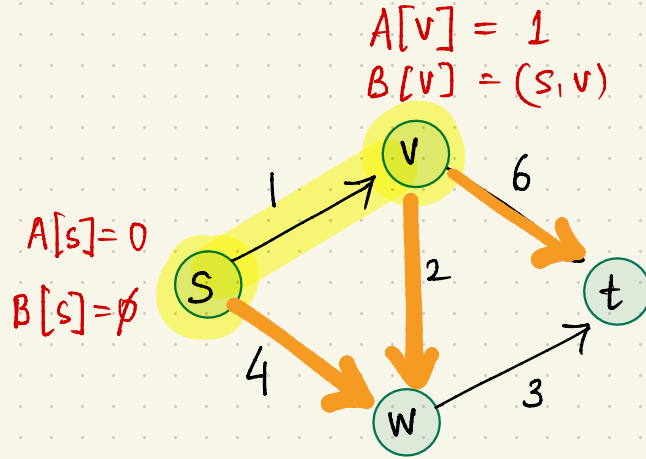


# DIJKSTRA'S ALGORITHM

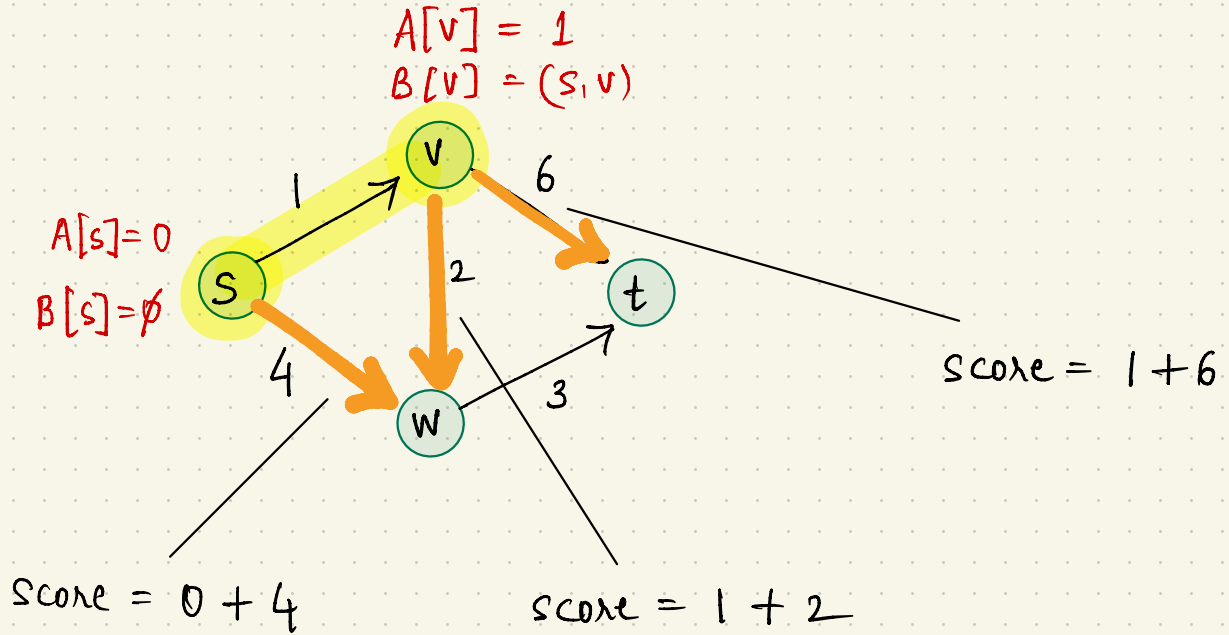




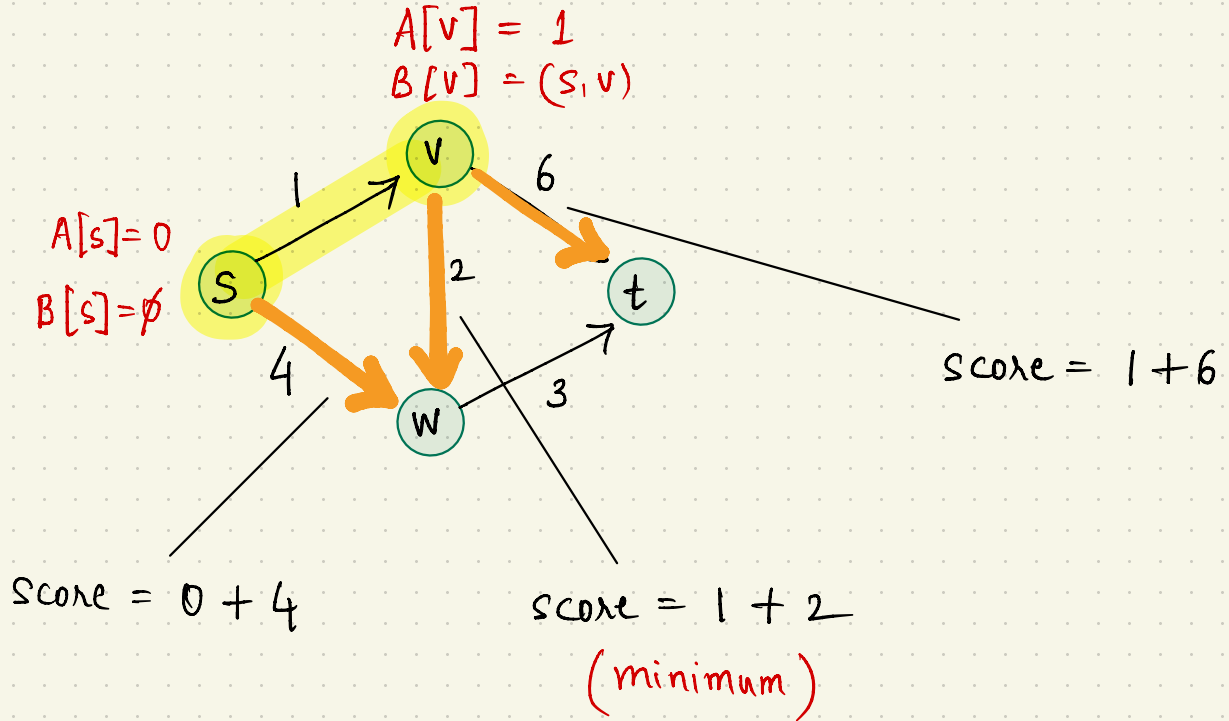
# DIJKSTRA'S ALGORITHM



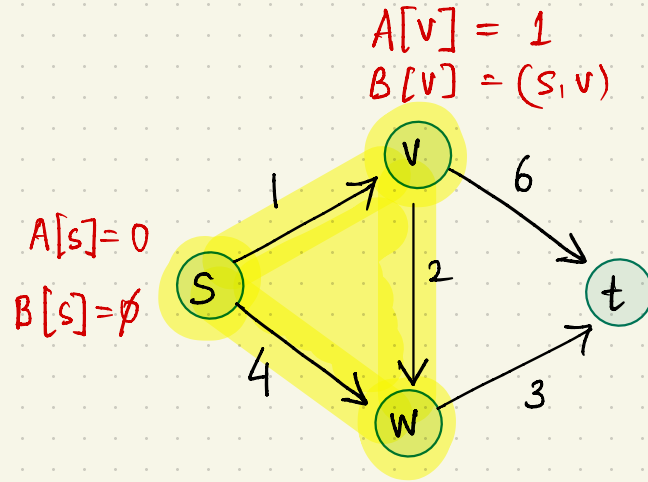
# DIJKSTRA'S ALGORITHM



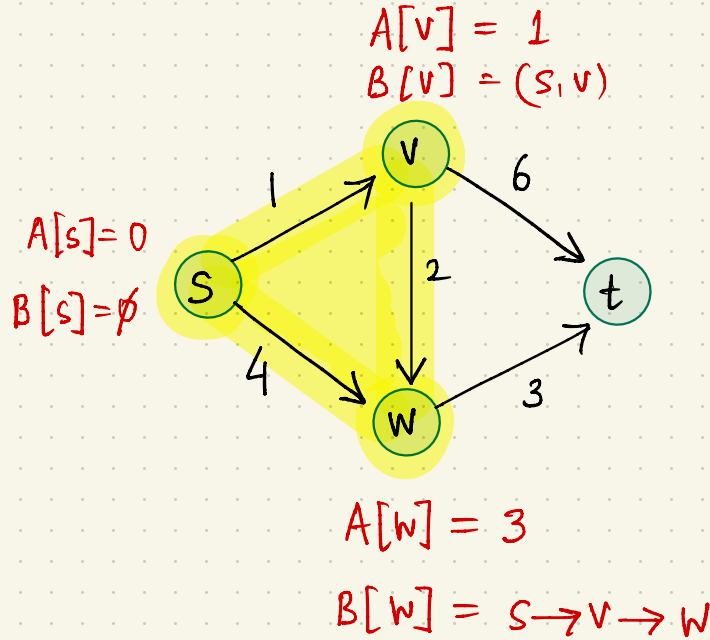
# DIJKSTRA'S ALGORITHM



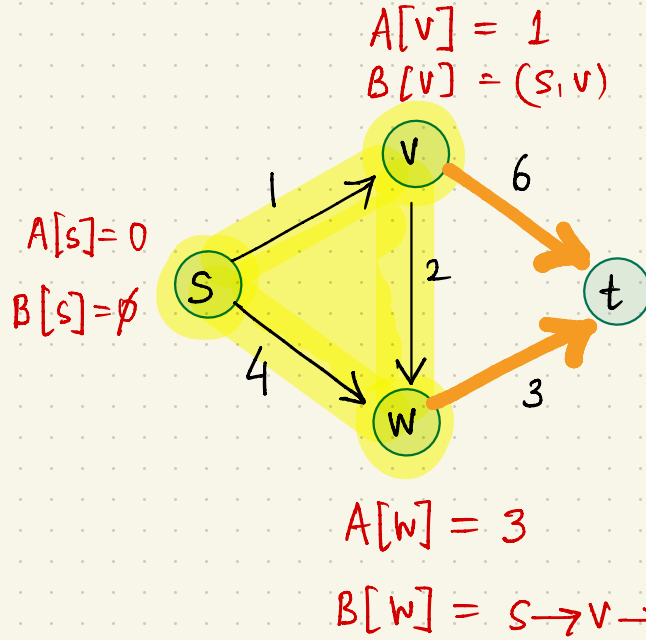
# DIJKSTRA'S ALGORITHM



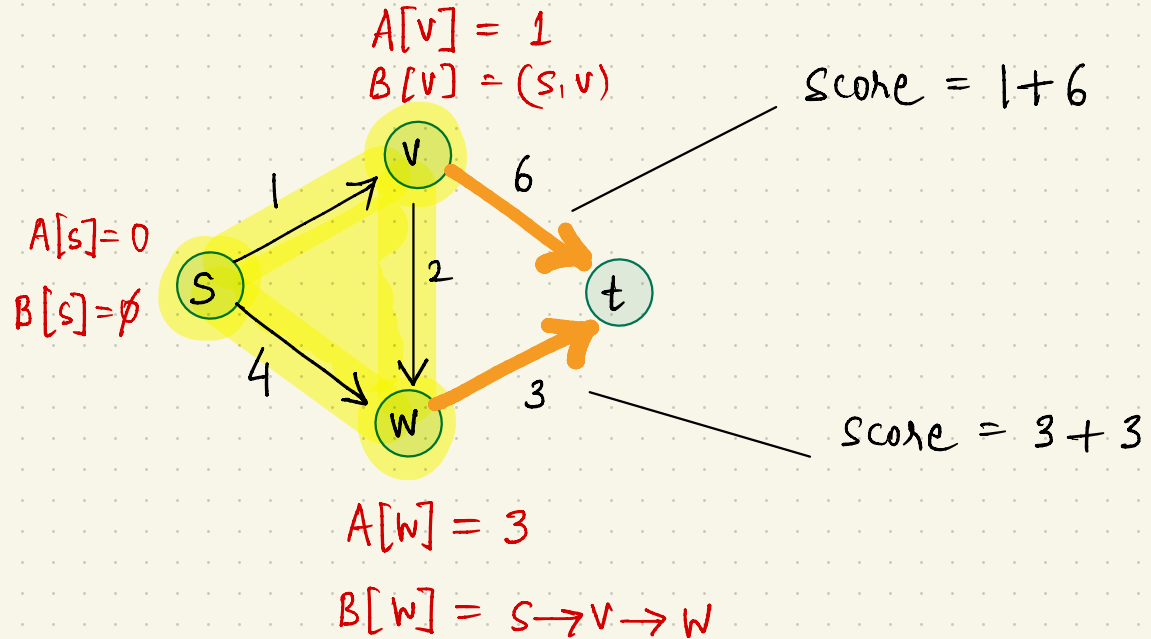
# DIJKSTRA'S ALGORITHM



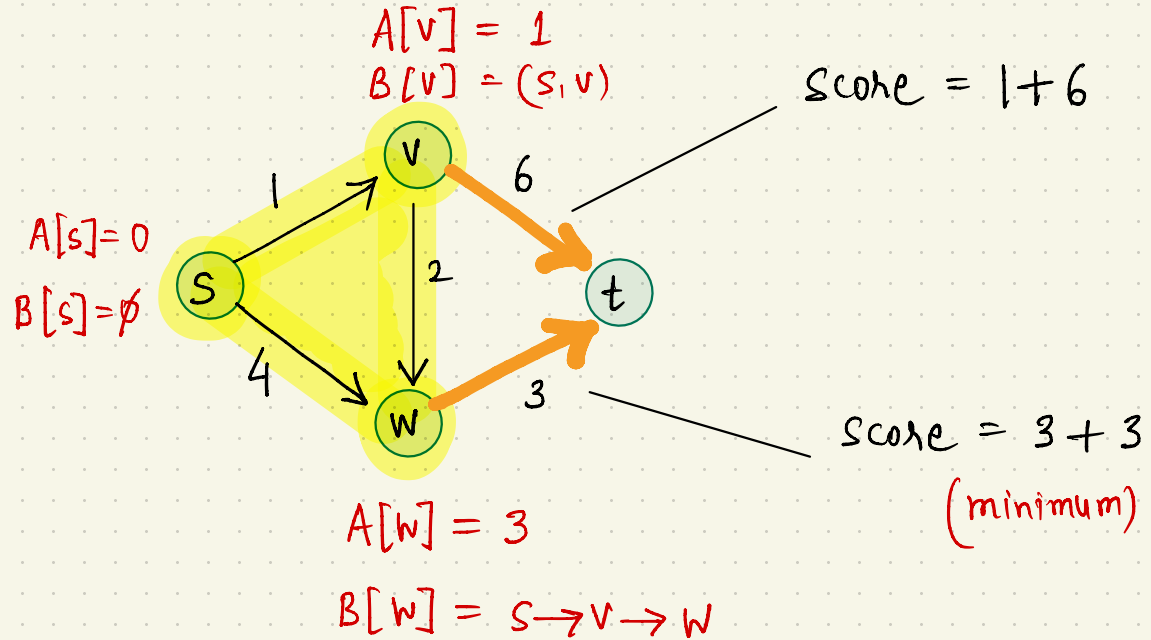
# DIJKSTRA'S ALGORITHM



# DIJKSTRA'S ALGORITHM

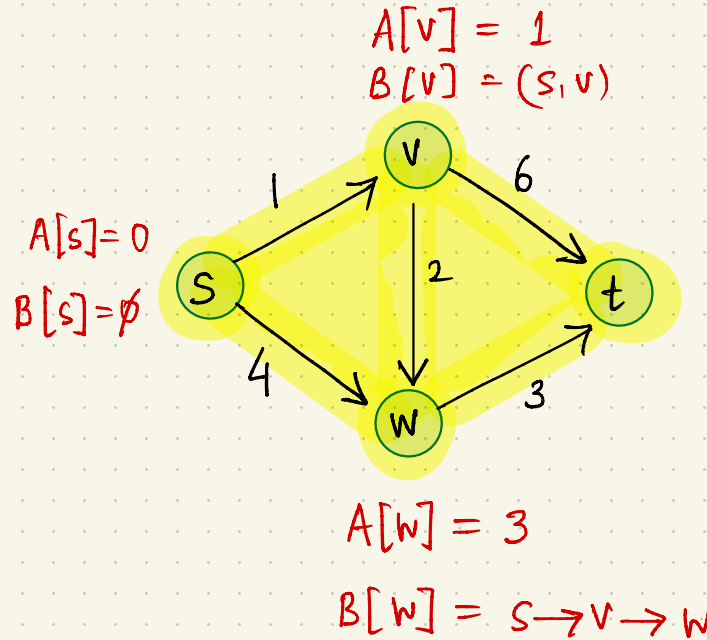


# DIJKSTRA'S ALGORITHM

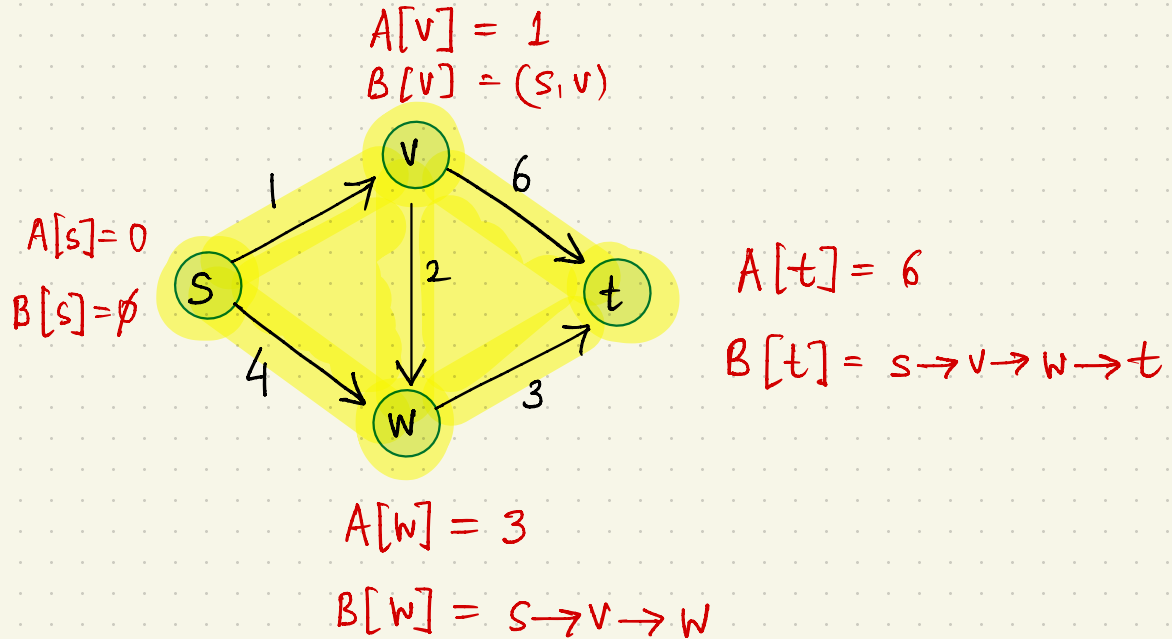




## DIJKSTRA'S ALGORITHM



# DIJKSTRA'S ALGORITHM

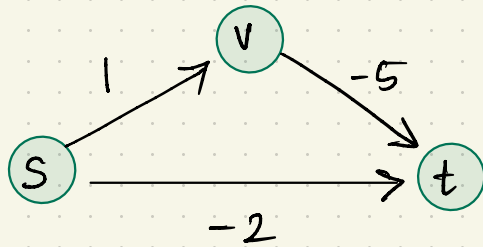




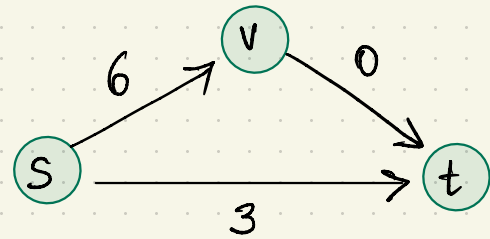
Negative edge lengths: what's the big deal?



Negative edge lengths: what's the big deal?

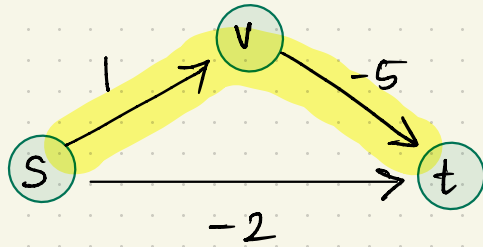


+5 per edge

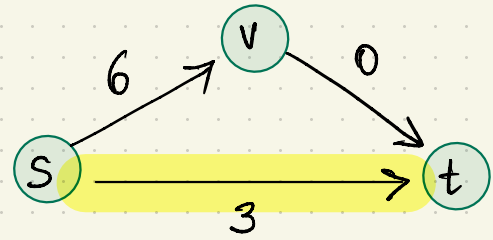




Negative edge lengths: what's the big deal?



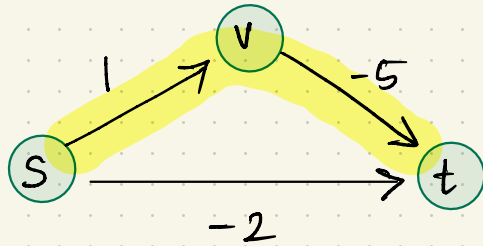
+5 per edge



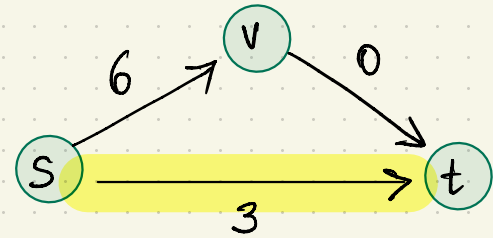
Doesn't preserve shortest paths!



Negative edge lengths: what's the big deal?



+5 per edge

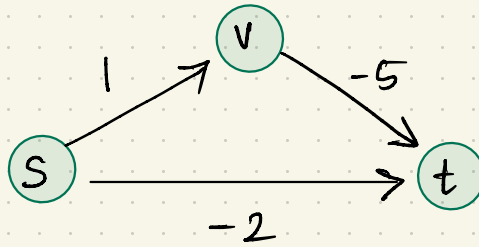


Doesn't preserve shortest paths!

Different paths have different number of edges.

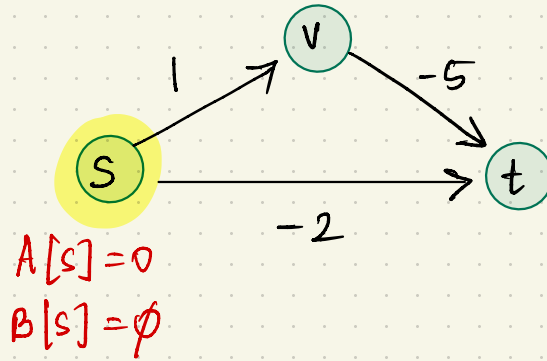


Never mind. Why don't we run Dijkstra anyway?





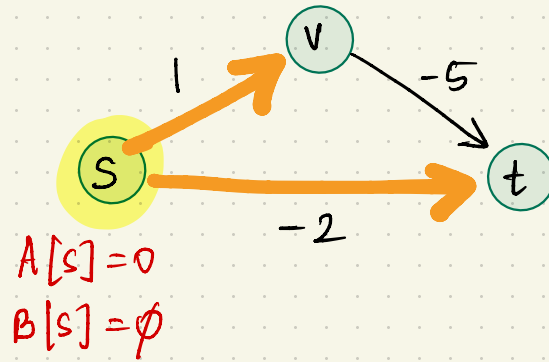
Never mind. Why don't we run Dijkstra anyway?





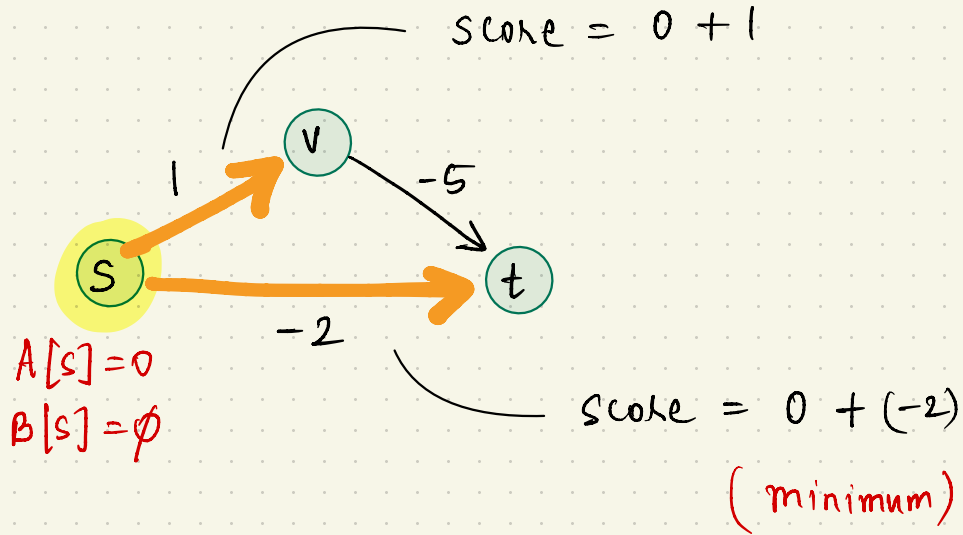


Never mind. Why don't we run Dijkstra anyway?



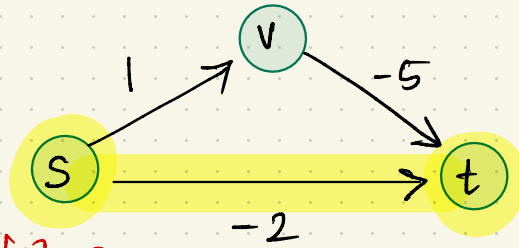


Never mind. Why don't we run Dijkstra anyway?





Never mind. Why don't we run Dijkstra anyway?

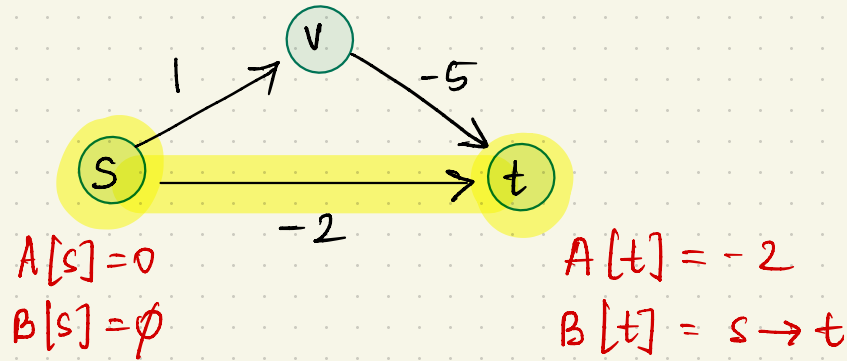


$$A[s] = 0$$

$$B[s] = \emptyset$$

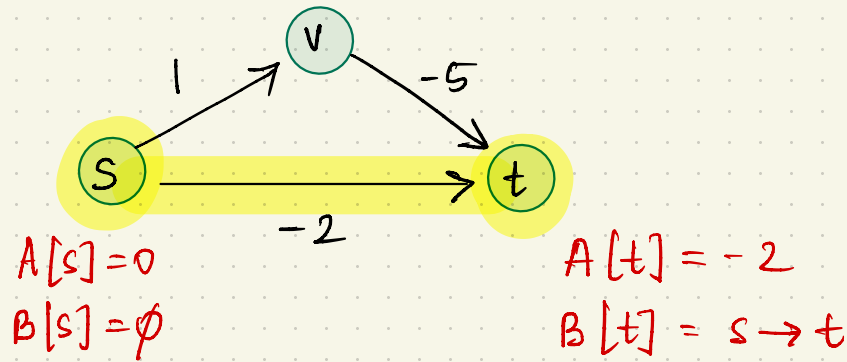


Never mind. Why don't we run Dijkstra anyway?





Never mind. Why don't we run Dijkstra anyway?



INCORRECT!

$s \rightarrow v \rightarrow t$  is shorter

# DIJKSTRA'S ALGORITHM: CORRECTNESS

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Theorem:** For every directed graph  $G = (V, E)$  with non negative edge lengths and for every starting vertex  $s$ , at the conclusion of Dijkstra's algorithm,

$$A[v] = \text{dist}(s, v).$$

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Theorem:** For every directed graph  $G = (V, E)$  with non negative edge lengths and for every starting vertex  $s$ , at the conclusion of Dijkstra's algorithm,

$$A[v] = \text{dist}(s, v).$$

length of shortest path  
from  $s$  to  $v$



# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Theorem:** For every directed graph  $G = (V, E)$  with non negative edge lengths and for every starting vertex  $s$ , at the conclusion of Dijkstra's algorithm,

$$A[v] = \text{dist}(s, v).$$

$$\text{length of } B[v] = \text{dist}(s, v).$$

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Theorem:** For every directed graph  $G = (V, E)$  with non negative edge lengths and for every starting vertex  $s$ , at the conclusion of Dijkstra's algorithm,

$$A[v] = \text{dist}(s, v).$$

$$\text{length of } B[v] = \text{dist}(s, v).$$

Proof by strong induction.

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$ .

(and length of  $B[v] = \text{dist}(s, v)$ )

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

Base case  $k=1$  :  $A[s] = 0 = \text{dist}(s, s)$  ✓

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

Base case  $k=1$  :  $A[s] = 0 = \text{dist}(s, s)$  ✓

/  
requires no negative length path  
from  $s$  to itself

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

Base case  $k=1$  :  $A[s] = 0 = \text{dist}(s, s)$  ✓

$B[s] = \emptyset$  ✓

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$ .

**Ind. step:** Assume for all previous iterations  $1, 2, \dots, k-1$

$A[v] = \text{dist}(s, v)$  and length of  $B[v] = \text{dist}(s, v)$ .



# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

**Ind. step:** Assume for all previous iterations  $1, 2, \dots, k-1$   
 $A[v] = \text{dist}(s, v)$  and length of  $B[v] = \text{dist}(s, v)$ .

Let  $w^*$  be the  $k^{\text{th}}$  vertex added to  $X$

say, through the edge  $(v^*, w^*)$  where  $v^* \in X$ ,  $w^* \notin X$ .

**Want:**  $A[w^*] = \text{dist}(s, w^*)$

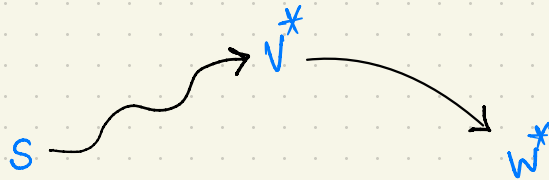
length of  $B[w^*] = \text{dist}(s, w^*)$ .

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

Ind. step :

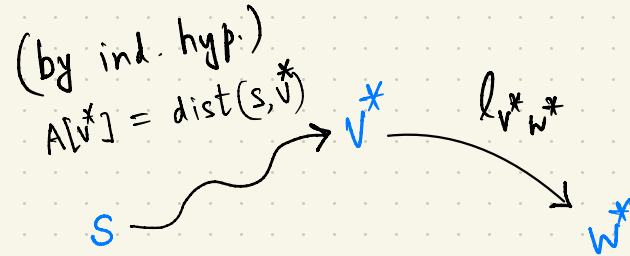


# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

Ind. step :

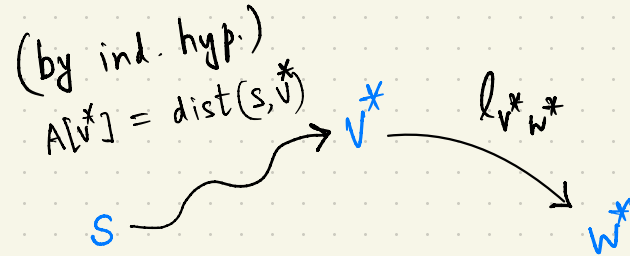


# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$ .

Ind. step :



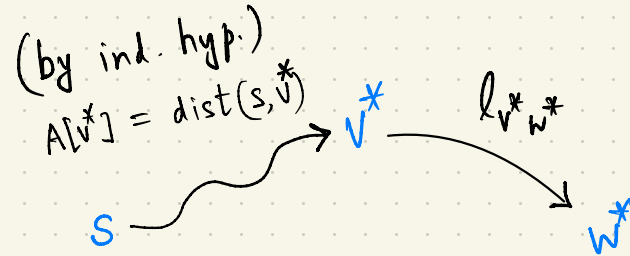
We set  $A[w^*] = A[v^*] + l_{v^*w^*}$  and  $B[w^*] = B[v^*] \cup (v^*, w^*)$ .

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

Ind. step :



We set  $A[w^*] = A[v^*] + l_{v^* w^*}$  and  $B[w^*] = B[v^*] \cup (v^*, w^*)$ .

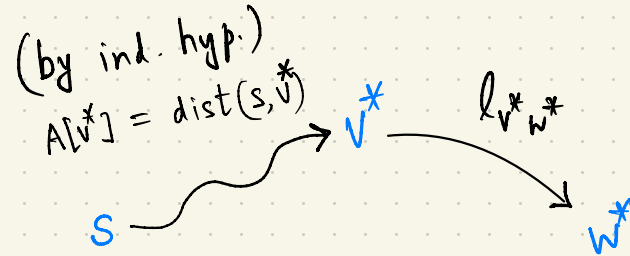
$$\Rightarrow \text{dist}(s, w^*) \leq A[v^*] + l_{v^* w^*}$$

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

Ind. step :



We set  $A[w^*] = A[v^*] + l_{v^* w^*}$  and  $B[w^*] = B[v^*] \cup (v^*, w^*)$ .

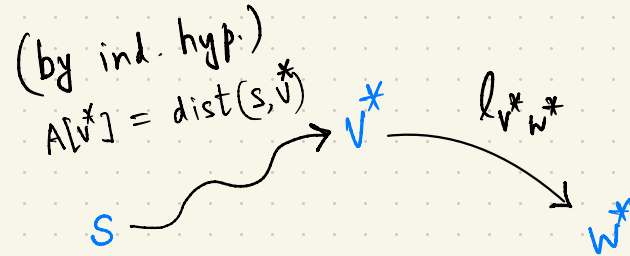
$$\Rightarrow \boxed{\text{dist}(s, w^*) \leq A[v^*] + l_{v^* w^*}} \quad \text{---} \quad \text{☺}$$

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

Ind. step :



We set  $A[w^*] = A[v^*] + l_{v^*w^*}$  and  $B[w^*] = B[v^*] \cup (v^*, w^*)$ .

$$\Rightarrow \boxed{\text{dist}(s, w^*) \leq A[v^*] + l_{v^*w^*}} \quad \text{---} \quad \text{😊}$$

Want : " "  $\geq$  " " "

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$ .

**Ind. step:** Suffices to show that **every**  $s-w^*$  path has length at least  $A[v^*] + l_{v^*w^*}$ .



# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$ .

**Ind. step:** Suffices to show that **every**  $s-w^*$  path has length at least  $A[v^*] + l_{v^*w^*}$ .

Let  $P$  be any  $s-w^*$  path.

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$ .

**Ind. step:** Suffices to show that **every**  $s - w^*$  path has length at least  $A[v^*] + l_{v^* w^*}$ .

Let  $P$  be any  $s - w^*$  path.

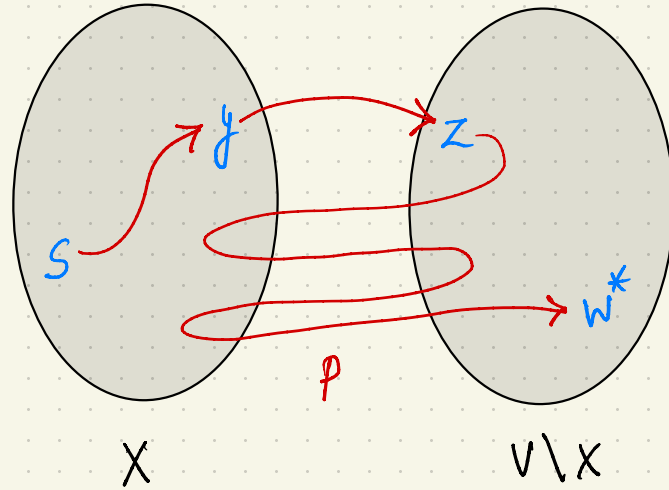
**Any such path must cross the frontier between  $X$  and  $V \setminus X$ .**

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

Ind. step :

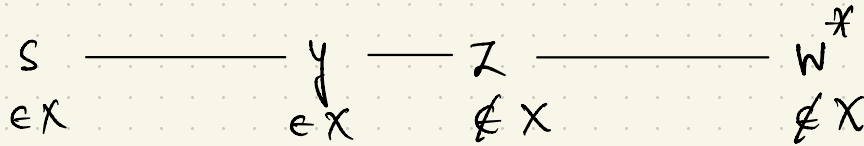


# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

**Ind. step:** Any  $s-w^*$  path  $P$  has the form :

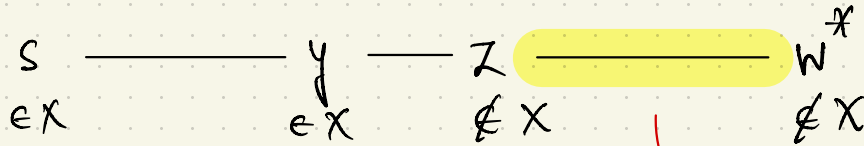


# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$ : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

**Ind. step:** Any  $s-w^*$  path  $P$  has the form:



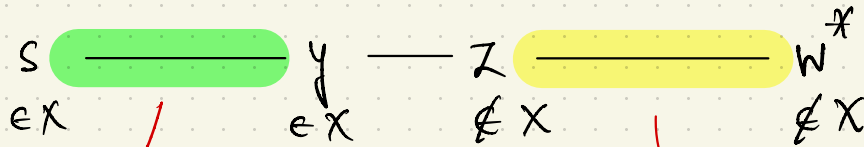
must have nonnegative length

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

**Ind. step:** Any  $s - w^*$  path  $P$  has the form :



lies entirely in  $X$

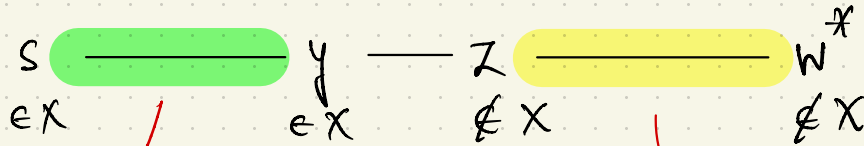
must have nonnegative length

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

**Ind. step:** Any  $s - w^*$  path  $P$  has the form :



lies entirely in  $X$

length  $\geq \text{dist}(s, y)$

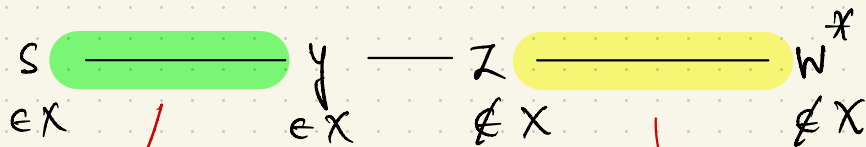
must have nonnegative length

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$ : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

**Ind. step:** Any  $s - w^*$  path  $P$  has the form:



lies entirely in  $X$

length  $\geq \text{dist}(s, y)$

$= A[y]$  (by ind. hyp.)

must have nonnegative length

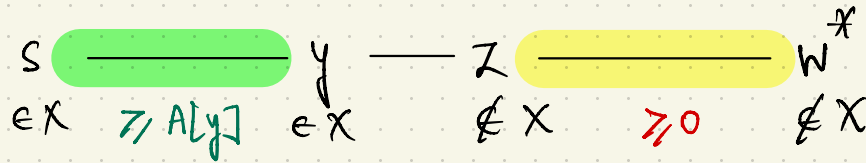


# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

**Ind. step:** Any  $s - w^*$  path  $P$  has the form :

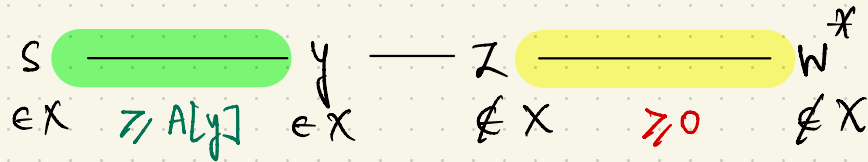


# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$ .

**Ind. step:** Any  $s \rightarrow w^*$  path  $P$  has the form :



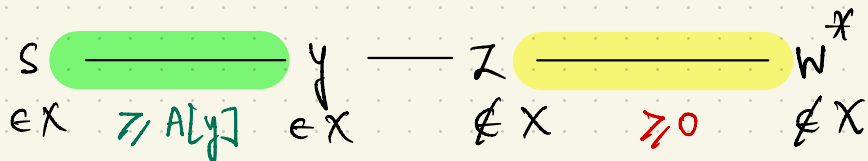
$$\begin{aligned} \text{length of path } P &= \text{length } s \rightsquigarrow y + \text{length } y \rightsquigarrow z + \text{length } z \rightsquigarrow w^* \\ &\geq A[y] + l_{yz} \end{aligned}$$

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$ : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

**Ind. step:** Any  $s \rightarrow w^*$  path  $P$  has the form:



$$\begin{aligned} \text{length of path } P &= \text{length } s \rightsquigarrow y + \text{length } y \rightsquigarrow z + \text{length } z \rightsquigarrow w^* \\ &\geq A[y] + l_{yz} \end{aligned}$$

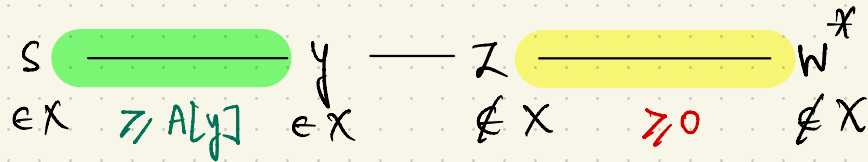
By Dijkstra score criterion:  $A[v^*] + l_{v^*w^*} \leq A[y] + l_{yz}$

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$ .

**Ind. step:** Any  $s \rightarrow w^*$  path  $P$  has the form :



$$\begin{aligned} \text{length of path } P &= \text{length } s \rightsquigarrow y + \text{length } y \rightsquigarrow z + \text{length } z \rightsquigarrow w^* \\ &\geq A[y] + l_{yz} \end{aligned}$$

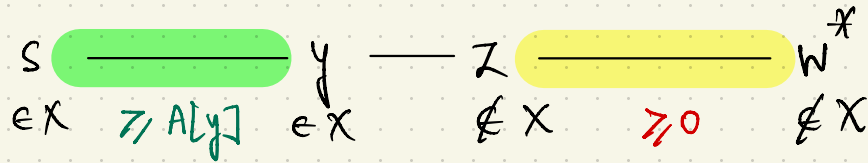
By **Dijkstra score criterion** :  $A[v^*] + l_{v^*w^*} \leq A[y] + l_{yz}$   
( $y, z$ ) was available to algorithm, but it chose ( $v^*, w^*$ ).

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

**Ind. step:** Any  $s \rightarrow w^*$  path  $P$  has the form :



$$\begin{aligned} \text{length of path } P &= \text{length } s \rightsquigarrow y + \text{length } y \rightsquigarrow z + \text{length } z \rightsquigarrow w^* \\ &\geq A[y] + l_{yz} \end{aligned}$$

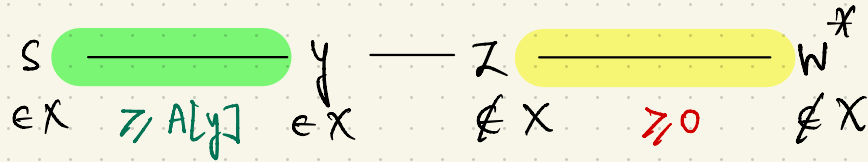
$$\begin{aligned} \text{By Dijkstra score criterion: } A[v^*] + l_{v^*w^*} &\leq A[y] + l_{yz} \\ &\leq \text{length of path } P \end{aligned}$$

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$ .

**Ind. step:** Any  $s \rightarrow w^*$  path  $P$  has the form :



$$\begin{aligned} \text{length of path } P &= \text{length } s \rightsquigarrow y + \text{length } y \rightsquigarrow z + \text{length } z \rightsquigarrow w^* \\ &\geq A[y] + l_{yz} \end{aligned}$$

By Dijkstra score criterion :

$A[v^*] + l_{v^*w^*} \leq A[y] + l_{yz}$

$\leq \text{length of path } P$

😊😊

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

**Ind. step:** By ☺☺

$$\text{dist}(s, w^*) \geq A[v^*] + l_{v^* w^*}$$

# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

**Ind. step:** By ☺☺

$$\text{dist}(s, w^*) \geq A[v^*] + l_{v^* w^*}$$

By ☺

$$\text{dist}(s, w^*) \leq A[v^*] + l_{v^* w^*}$$



# DIJKSTRA'S ALGORITHM: CORRECTNESS

**Proof:** (by strong induction on the number of iterations)

$P(k)$  : for the  $k^{\text{th}}$  vertex  $v$  added to  $X$ ,  $A[v] = \text{dist}(s, v)$

**Ind. step:** By ☺☺

$$\text{dist}(s, w^*) \geq A[v^*] + l_{v^* w^*}$$

By ☺

$$\text{dist}(s, w^*) \leq A[v^*] + l_{v^* w^*}$$

$\Rightarrow s \rightarrow v^* \rightarrow w^*$  is the shortest path to  $w^*$ .



# DIJKSTRA'S ALGORITHM: RUNNING TIME

# DIJKSTRA'S ALGORITHM: RUNNING TIME

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

$B[s] = \emptyset$  computed shortest paths

// main loop

while  $X \neq V$  need to grow  $X$

among all edges  $(v, w) \in E$  with  $v \in X$  and  $w \notin X$ , pick one that minimizes

$A[v] + l_{vw}$  Dijkstra score

Call it  $(v^*, w^*)$ .

Add  $w^*$  to  $X$ .

Set  $A[w^*] := A[v^*] + l_{v^*w^*}$  and  $B[w^*] := B[v^*] \cup (v^*, w^*)$ .

# DIJKSTRA'S ALGORITHM: RUNNING TIME

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

// main loop

while  $X \neq V$  need to grow  $X$

among all edges  $(v, w) \in E$  with  $v \in X$  and  $w \notin X$ , pick one that minimizes

$A[v] + l_{vw}$  Dijkstra score

Call it  $(v^*, w^*)$ .

Add  $w^*$  to  $X$ .

Set  $A[w^*] := A[v^*] + l_{v^*w^*}$

# DIJKSTRA'S ALGORITHM: RUNNING TIME

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

// main loop  # iterations =  $n-1$  ( $n = \#$  vertices)

while  $X \neq V$  need to grow  $X$

among all edges  $(v, w) \in E$  with  $v \in X$  and  $w \notin X$ , pick one that minimizes

$A[v] + l_{vw}$  Dijkstra score

Call it  $(v^*, w^*)$ .

Add  $w^*$  to  $X$ .

Set  $A[w^*] := A[v^*] + l_{v^*w^*}$

# DIJKSTRA'S ALGORITHM: RUNNING TIME

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

// main loop  $\# \text{ iterations} = n - 1$  ( $n = \# \text{ vertices}$ )

while  $X \neq V$  need to grow  $X$   $\# \text{ checks} = m$  ( $m = \# \text{ edges}$ )

among all edges  $(v, w) \in E$  with  $v \in X$  and  $w \notin X$ , pick one that minimizes

$A[v] + l_{vw}$  Dijkstra score

Call it  $(v^*, w^*)$ .

Add  $w^*$  to  $X$ .

Set  $A[w^*] := A[v^*] + l_{v^*w^*}$

# DIJKSTRA'S ALGORITHM: RUNNING TIME

Naive implementation :  $O(mn)$

# DIJKSTRA'S ALGORITHM: RUNNING TIME

Naive implementation :  $O(mn)$

Can we do better?



# DIJKSTRA'S ALGORITHM: RUNNING TIME

Initialize

$X := \{s\}$  vertices processed so far

$A[s] = 0$  computed shortest-path distances

How to speed up computation of "minimum"?

// main loop

while  $X \neq V$  need to grow  $X$

among all edges  $(v, w) \in E$  with  $v \in X$  and  $w \notin X$ , pick one that minimizes

$$A[v] + l_{vw}$$

Dijkstra score

Call it  $(v^*, w^*)$ .

Add  $w^*$  to  $X$ .

Set  $A[w^*] := A[v^*] + l_{v^*w^*}$

# DIJKSTRA'S ALGORITHM: RUNNING TIME

Naive implementation:  $O(mn)$

Can we do better?

Yes!  $O((m+n) \log n)$  with heaps.

(see "Additional Reading").