#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

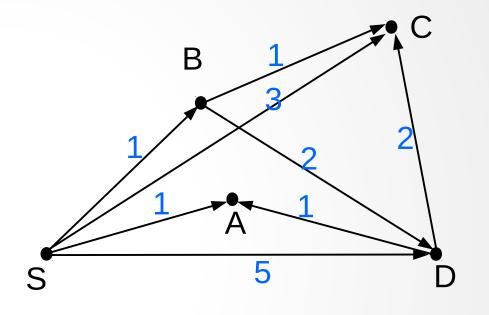
for each vertex *w* adjacent to *v*:

### **if** *w*'s distance > (*v*'s distance + weight(*v*,*w*):

set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: S, A, B, C, D

	S	А	В	С	D
parent	None	None	None	None None	
dist.	0	infty	infty	infty	infty



 $\infty \propto \infty \propto \infty$ 

#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

### while priority queue is not empty:

dequeue a vertex v with the shortest distance

for each vertex *w* adjacent to *v*:

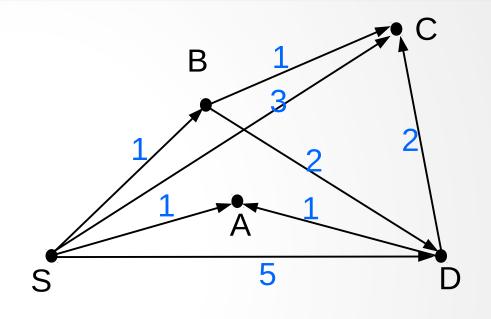
```
if w's distance > (v's distance + weight(v,w):
```

set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: S, A, B, C, D

deq	ueue	d:	S
adja	acent	to	S:

		priori	ly queue	∴ <u>≯</u> , A,	B, C, D
	S	A	В	С	D
parent	None	None	None	None	None
dist.	0	infty	infty	infty	infty



 $\infty \quad \infty \quad \infty \quad \infty \quad \infty$ 

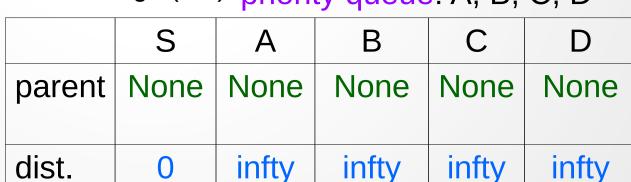
#### Edgar Dijkstra's algorithm:

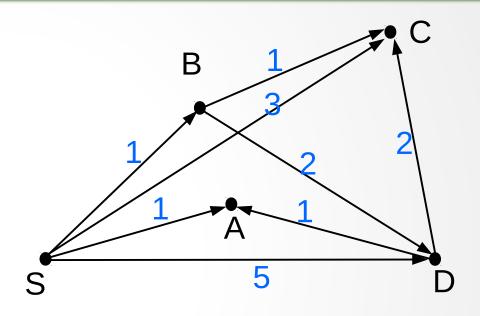
set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first). while priority queue is not empty:

- dequeue a vertex *v* with the shortest distance
- **for** each vertex *w* adjacent to *v*:
  - if w's distance > (v's distance + weight(v,w):
     set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: A, B, C, D

dequeued: S	
adjacent to S: A,B,C,D	pa





 $\mathbf{0}$ 

 $\infty$   $\infty$ 

 $\infty$ 

#### Edgar Dijkstra's algorithm:

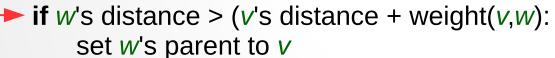
set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

#### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

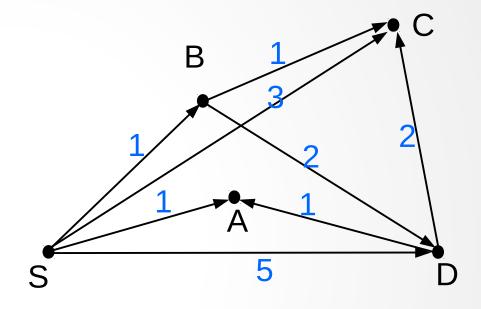
for each vertex *w* adjacent to *v*:

а



set w's distance to v's dist. + weight(v,w) priority queue: A, B, C, D

lequeued: S		S	A	B
adjacent to S: A,B,C,D	parent	None	None	None
	dist.	0	infty	infty



R

 $\mathbf{0}$  $\infty$   $\infty$  $\infty$ 

 $\square$ 

None

infty

C

None

infty

#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

#### while priority queue is not empty:

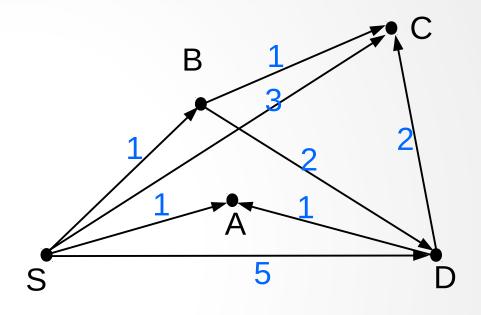
dequeue a vertex *v* with the shortest distance

for each vertex *w* adjacent to *v*:

- if w's distance > (v's distance + weight(v,w):
- set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: A, B, C, D

dequeued: S		S	Α	В	С	D
adjacent to S: A,B,C,D	parent	None	S	None	None	None
	dist.	0	1	infty	infty	infty



 $\infty$   $\infty$   $\infty$ 

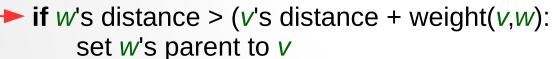
#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

#### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

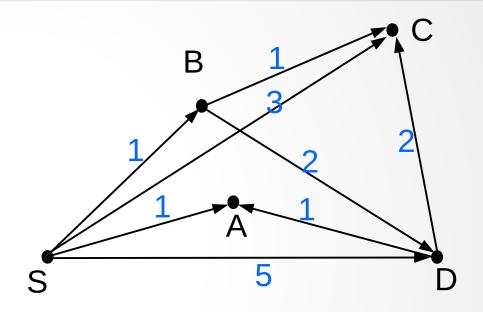
for each vertex *w* adjacent to *v*:



 $1 \propto \infty \infty$ 

set w's distance to v's dist. + weight(v,w) priority queue: A, B, C, D

dequeued: S		S	А	В	С	D
adjacent to S: A,B,C,D	parent	None	S	None	None	None
	dist.	0	1	infty	infty	infty



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

#### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

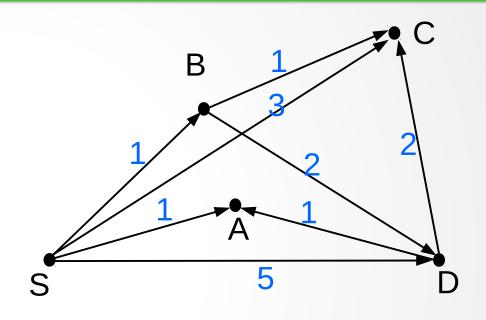
for each vertex *w* adjacent to *v*:

**if** *w*'s distance > (*v*'s distance + weight(*v*,*w*):

 $\rightarrow$  set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: A, B, C, D

dequeued: S		S	А	В	С	D
adjacent to S: A,B,C,D	parent	None	S	S	None	None
	dist.	0	1	1	infty	infty



 $\infty$   $\infty$ 

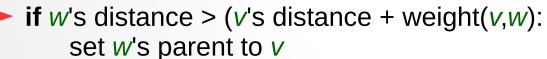
#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

#### while priority queue is not empty:

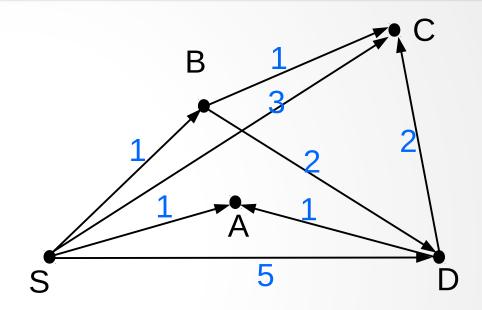
dequeue a vertex *v* with the shortest distance

for each vertex *w* adjacent to *v*:



set w's distance to v's dist. + weight(v,w) priority queue: A, B, C, D

dequeued: S		S	А	В	С	D
adjacent to S: A,B,C,D	parent	None	S	S	None	None
	dist.	0	1	1	infty	infty



 $\infty$   $\infty$ 

#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

#### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

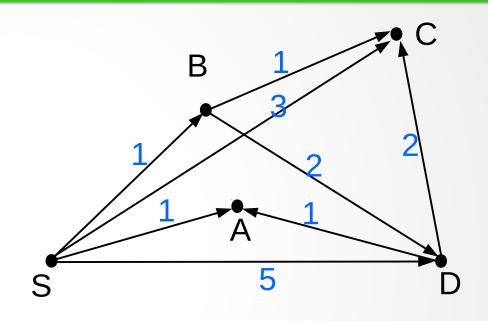
for each vertex *w* adjacent to *v*:

**if** *w*'s distance > (*v*'s distance + weight(*v*,*w*):

 $\rightarrow$  set *w*'s parent to *v* 

set w's distance to v's dist. + weight(v,w) priority queue: A, B, C, D

dequeued: S		S	А	В	С	D
adjacent to S: A,B,C,D	parent	None	S	S	S	None
	dist.	0	1	1	3	infty



3

 $\infty$ 

#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

#### while priority queue is not empty:

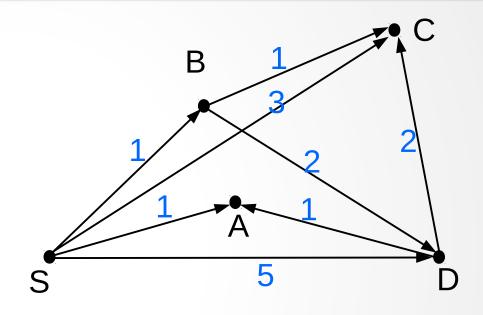
dequeue a vertex *v* with the shortest distance

for each vertex *w* adjacent to *v*:



set w's distance to v's dist. + weight(v,w) priority queue: A, B, C, D

dequeued: S		S	А	В	С	D
adjacent to S: A,B,C,D	parent	None	S	S	S	None
	dist.	0	1	1	3	infty



3

 $\infty$ 

#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

#### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

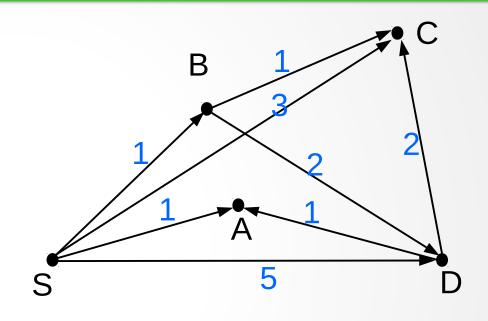
for each vertex *w* adjacent to *v*:

**if** *w*'s distance > (*v*'s distance + weight(*v*,*w*):

 $\rightarrow$  set *w*'s parent to *v* 

 $\rightarrow$  set w's distance to v's dist. + weight(v,w) priority queue: A, B, C, D

2 ·bauauah		S	А	В	С	D
dequeued: S adjacent to S: A,B,C,D	parent	None	S	S	S	S
	dist.	0	1	1	3	5



3

1 1

#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first). while priority queue is not empty:

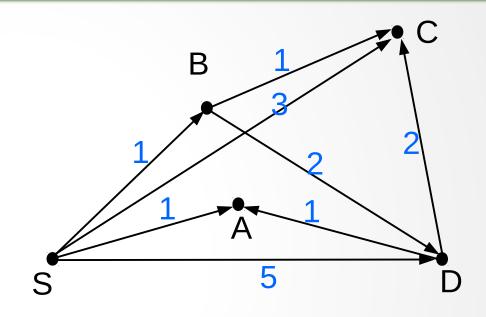
dequeue a vertex *v* with the shortest distance

for each vertex *w* adjacent to *v*:

if w's distance > (v's distance + weight(v,w):
 set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: A, B, C, D

2 ·bauauah		S	А	В	С	D
dequeued: S adjacent to S: A,B,C,D	parent	None	S	S	S	S
	dist.	0	1	1	3	5



3

1 1

- 5

#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

### while priority queue is not empty:

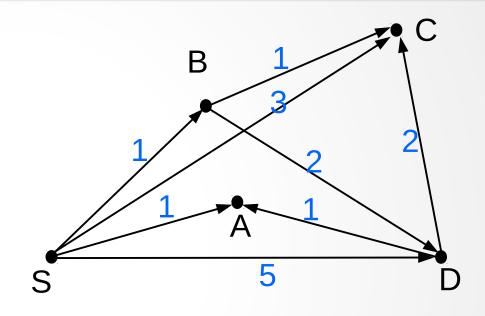
dequeue a vertex v with the shortest distance

for each vertex *w* adjacent to *v*:

if w's distance > (v's distance + weight(v,w): set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: A, B, C, D

dequeued:		S	А	В	С	D	
adjacent to :	parent	None	S	S	S	S	
	dist.	0	1	1	3	5	



5

#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

### while priority queue is not empty:

dequeue a vertex v with the shortest distance

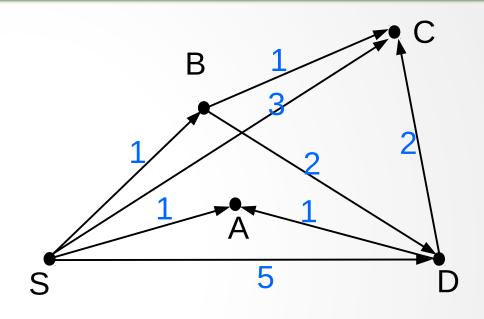
for each vertex *w* adjacent to *v*:

if w's distance > (v's distance + weight(v,w):
 set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: A. B. C. D

deq	ueueo	A:b
adja	acent	to A:

J C / Phoney queue. A, D, C, D							
	S	А	В	С	D		
parent	None	S	S	S	S		
dist.	0	1	1	3	5		



1 1 3 5

#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first). while priority queue is not empty: dequeue a vertex v with the shortest

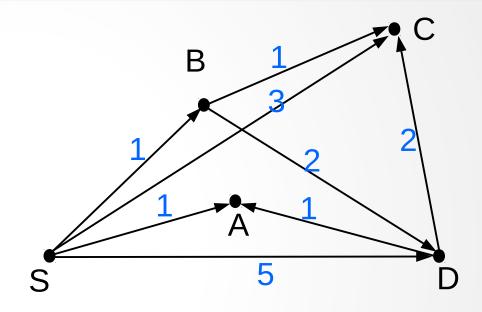
distance

for each vertex *w* adjacent to *v*:

if w's distance > (v's distance + weight(v,w):
 set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: B, C, D

		S	А	В	С	D
dequeued: A adjacent to A:	parent	None	S	S	S	S
	dist.	0	1	1	3	5



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

### while priority queue is not empty:

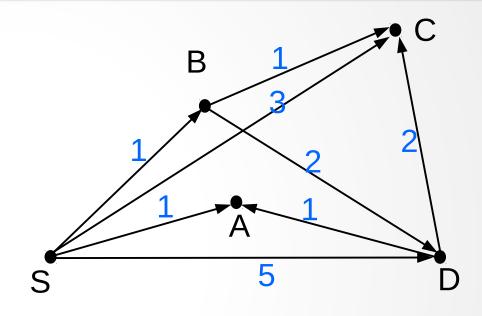
dequeue a vertex *v* with the shortest distance

for each vertex *w* adjacent to *v*:

if w's distance > (v's distance + weight(v,w):
 set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: B, C, D

dequeued.		S	А	В	С	D
dequeued: adjacent to :	parent	None	S	S	S	S
	dist.	0	1	1	3	5



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

for each vertex *w* adjacent to *v*:

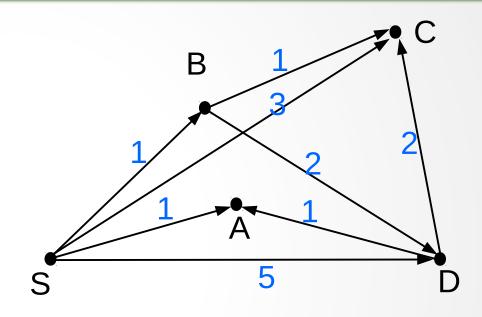
**if** *w*'s distance > (*v*'s distance + weight(v,w):

set w's parent to v

set w's distance to v's dist. + weight(v, w) priority queue:  $\mathbb{B}$ , C,

Deq	lueue	d: B	
adja	icent t	to B:	C,D

5 aist. + w	eigni( <i>v,w)</i>	priori	ty queue	E:₿, C,	D
	S	А	В	С	D
parent	None	S	S	S	S
dist.	0	1	1	3	5



#### Edgar Dijkstra's algorithm:

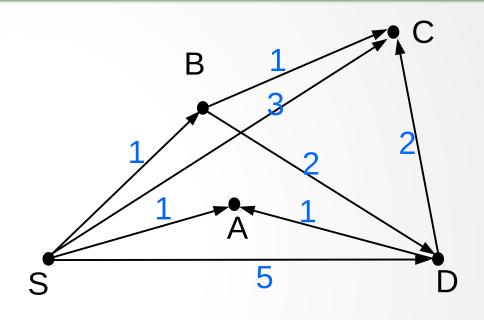
set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first). while priority queue is not empty:

- dequeue a vertex *v* with the shortest distance
- for each vertex *w* adjacent to *v*:
  - if w's distance > (v's distance + weight(v,w):
     set w's parent to v

set w's distance to v's dist. + weight(v, w) priority queue: C, E

Dequeued: B adjacent to B: C,D

uist. + weight( <i>v</i> , <i>w</i> ) priority queue: C, D								
	S	А	В	С	D			
parent	None	S	S	S	S			
dist.	0	1	1	3	5			



3

-5

#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

#### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

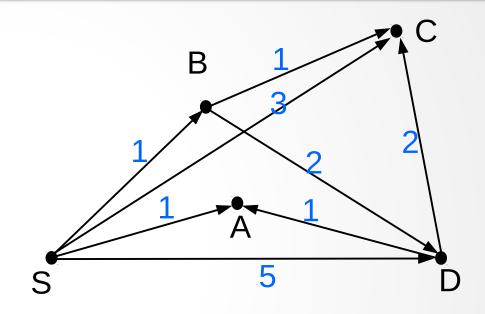
for each vertex *w* adjacent to *v*:

#### if w's distance > (v's distance + weight(v,w): set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: C. D

Dequeued: B	
adjacent to B: C,D	

phony queue. C, D							
	S	А	В	С	D		
parent	None	S	S	S	S		
dist.	0	1	1	3	5		



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

#### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

for each vertex *w* adjacent to *v*:

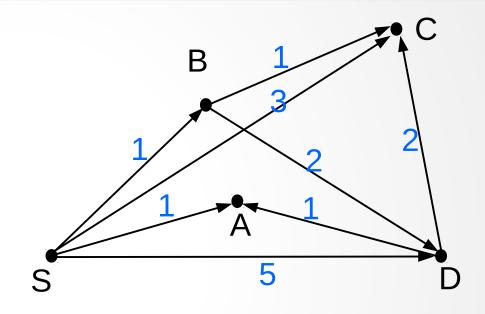
**if** *w*'s distance > (*v*'s distance + weight(*v*,*w*):

 $\rightarrow$  set *w*'s parent to *v* 

 $\rightarrow$  set w's distance to v's dist. + weight(v,w) priority queue: C, D

Dequeued: B adjacent to B: C,D

phoney queue. C, D							
	S	А	В	С	D		
parent	None	S	S	B	S		
dist.	0	1	1	2	5		



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

#### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

for each vertex *w* adjacent to *v*:

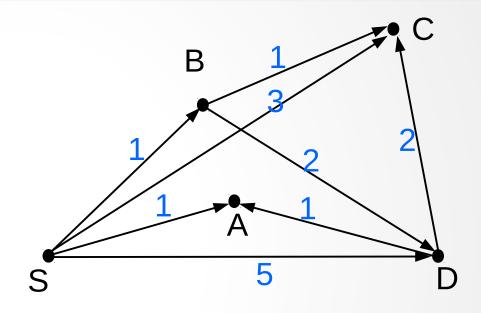
**if** *w*'s distance > (*v*'s distance + weight(*v*,*w*):

 $\rightarrow$  set *w*'s parent to *v* 

 $\rightarrow$  set w's distance to v's dist. + weight(v,w) priority queue: C, D

Dequeued: B adjacent to B: C,D

dist. • weight(v,w) priority queue: C, D							
	S	А	В	С	D		
parent	None	S	S	В	S		
dist.	0	1	1	2	5		



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

#### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

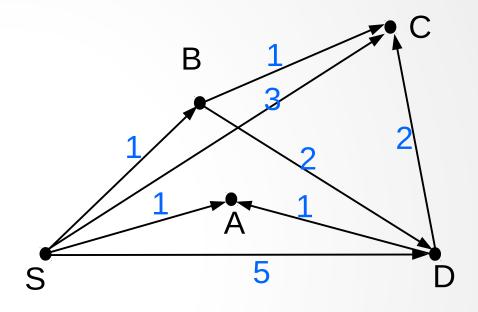
for each vertex *w* adjacent to *v*:

#### if w's distance > (v's distance + weight(v,w): set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: C. D

Deq	lueued:	B	
adja	acent to	) B:	C,D

	o the phoney queue. O, D						
	S	А	В	С	D		
parent	None	S	S	В	B		
dist.	0	1	1	2	53		



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

#### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

for each vertex *w* adjacent to *v*:

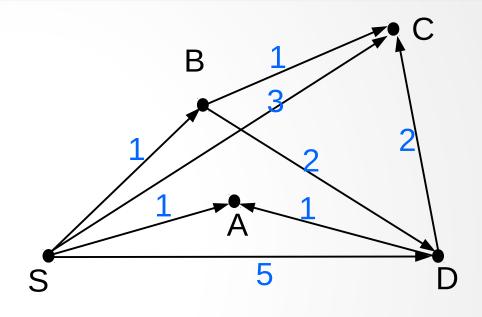
**if** *w*'s distance > (*v*'s distance + weight(*v*,*w*):

 $\rightarrow$  set *w*'s parent to *v* 

 $\rightarrow$  set w's distance to v's dist. + weight(v,w) priority queue: C, D

Dequeued: B adjacent to B: C,D

	o v phony queue. O, D						
	S	А	В	С	D		
parent	None	S	S	В	В		
dist.	0	1	1	2	3		



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first). while priority queue is not empty: dequeue a vertex v with the shortest

distance

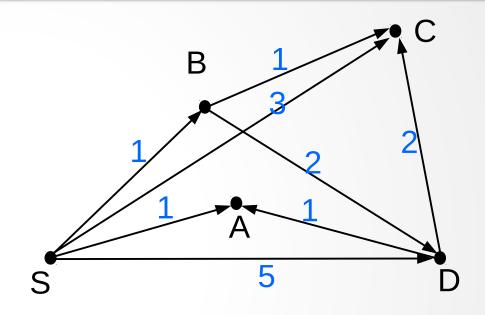
for each vertex *w* adjacent to *v*:

if w's distance > (v's distance + weight(v,w):
 set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: C, D

Dequeued: B adjacent to B: C,D

	J ( ) phoney queue. C, D						
	S	А	В	С	D		
parent	None	S	S	В	В		
dist.	0	1	1	2	3		



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

for each vertex *w* adjacent to *v*:

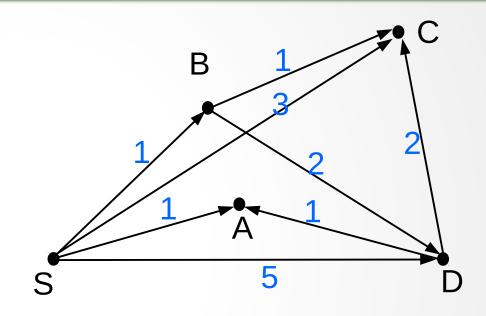
if w's distance > (v's distance + weight(v,w):
 set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: C. D

D					Ч	•
$\boldsymbol{\nu}$	C	4	u	u	U	•

adjacent to :

	o ( ) phoney queue. O, D						
	S	А	В	С	D		
parent	None	S	S	В	В		
dist.	0	1	1	2	3		



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

for each vertex *w* adjacent to *v*:

if w's distance > (v's distance + weight(v,w):
 set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue:  $\bigotimes$  D

S

Dequeued: C adjacent to C:

	S	А	В	С	D		
parent	None	S	S	В	В		
dist.	0	1	1	2	3		

B

5

#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first). while priority queue is not empty: dequeue a vertex v with the shortest

distance

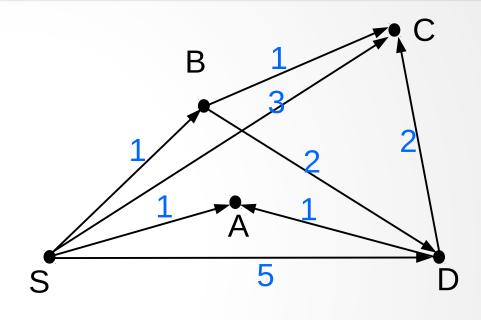
for each vertex *w* adjacent to *v*:

if w's distance > (v's distance + weight(v,w):
 set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: D

Dequeued: C adjacent to C:

o the phone queue. D							
	S	А	В	С	D		
parent	None	S	S	В	В		
dist.	0	1	1	2	3		



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

for each vertex *w* adjacent to *v*:

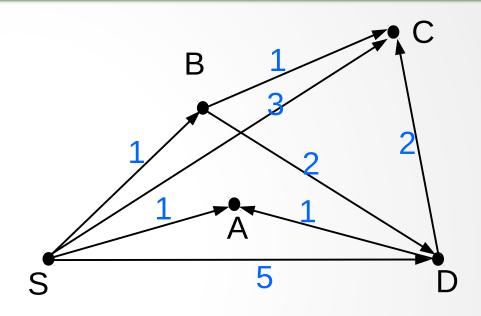
if w's distance > (v's distance + weight(v,w):
 set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue: D

Dog		Ind.
Deq	uei	led:

adjacent to :

	o v phony queue. D							
	S	А	В	С	D			
parent	None	S	S	В	В			
dist.	0	1	1	2	3			



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

for each vertex *w* adjacent to *v*:

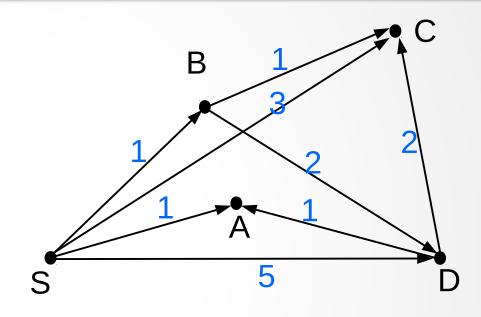
**if** *w*'s distance > (*v*'s distance + weight(*v*,*w*):

set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue:  $\bigotimes$ 

Dequeued: D adjacent to D: A

	S	А	В	С	D			
parent	None	S	S	В	В			
dist.	0	1	1	2	3			



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first). while priority queue is not empty: dequeue a vertex v with the shortest

distance

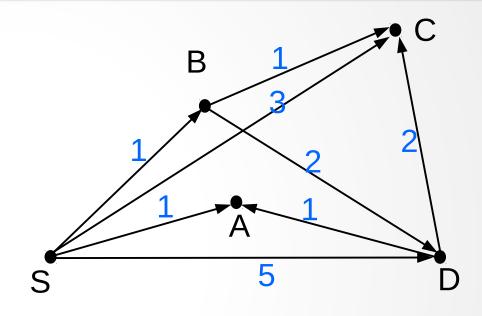
for each vertex *w* adjacent to *v*:

if w's distance > (v's distance + weight(v,w):
 set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue:

Dequeued: D adjacent to D: A

o ( / phoney queue.						
	S	А	В	С	D	
parent	None	S	S	В	В	
dist.	0	1	1	2	3	



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

for each vertex *w* adjacent to *v*:

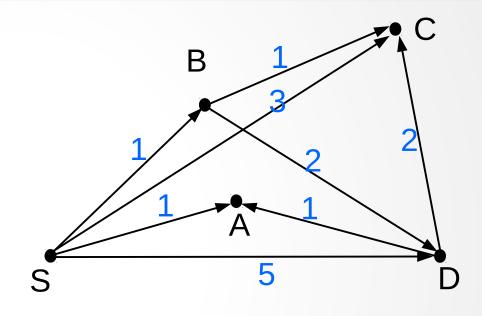
### **if** w's distance > (v's distance + weight(v,w):

set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue:

Dequeued: D adjacent to D: A

phony quouol						
	S	А	В	С	D	
parent	None	S	S	В	В	
dist.	0	1	1	2	3	



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first). while priority queue is not empty: dequeue a vertex v with the shortest distance

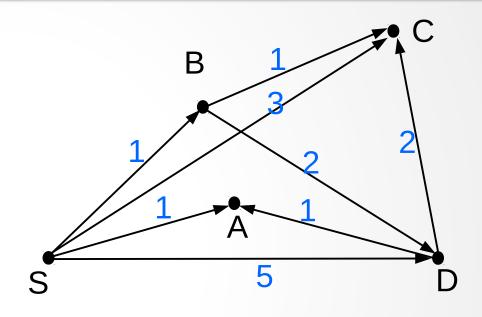
**for** each vertex *w* adjacent to *v*:

if w's distance > (v's distance + weight(v,w):
 set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue:

Dequeued: D adjacent to D: A

	S	А	В	С	D		
parent	None	S	S	В	В		
dist.	0	1	1	2	3		



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

#### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

for each vertex *w* adjacent to *v*:

**if** *w*'s distance > (*v*'s distance + weight(*v*,*w*):

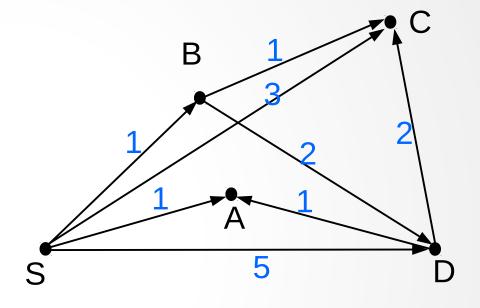
#### set w's parent to v

Dequeued:

adjacent to :

set w's distance to v's dist. + weight(v,w) priority queue:

	S	А	В	С	D
parent	None	S	S	В	В
dist.	0	1	1	2	3



#### Edgar Dijkstra's algorithm:

set all vertices to have parent None. set distance for all vertices to *infinity* set distance for source vertex to 0 insert all vertices into a priority queue (by distance, smallest first).

#### while priority queue is not empty:

dequeue a vertex *v* with the shortest distance

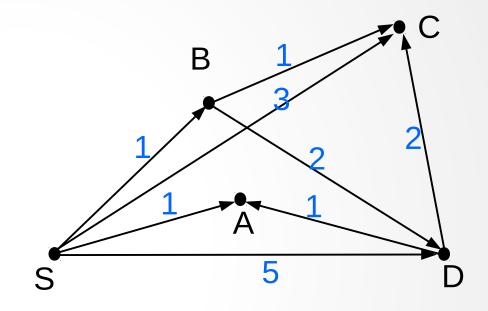
for each vertex *w* adjacent to *v*:

**if** *w*'s distance > (*v*'s distance + weight(*v*,*w*):

set w's parent to v

set w's distance to v's dist. + weight(v,w) priority queue:

S1	TOP

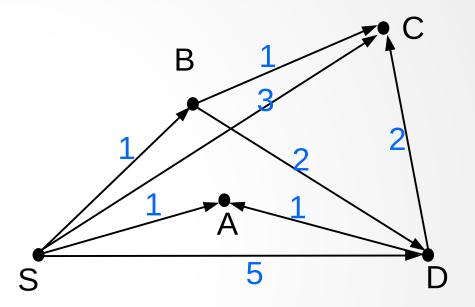


	S	А	В	С	D
parent	None	S	S	В	В
dist.	0	1	1	2	3

The table is ready to be used.

```
For example,
the shortest path from S to D is
S \rightarrow B \rightarrow D
```

```
the shortest path from S to C is S \rightarrow B \rightarrow C
```



	S	А	В	С	D
parent	None	S	S	В	В
dist.	0	1	1	2	3