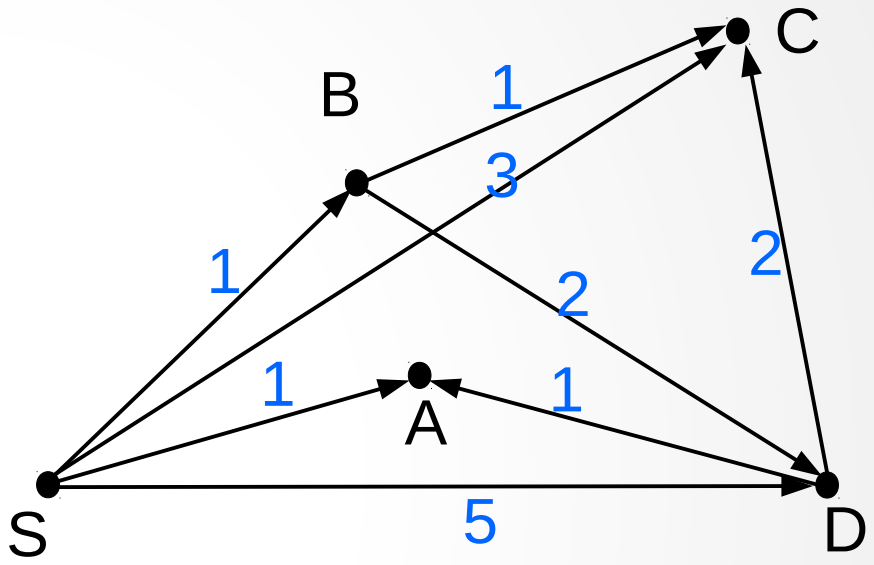
Dijkstra's algorithm for weighted graphs

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*)



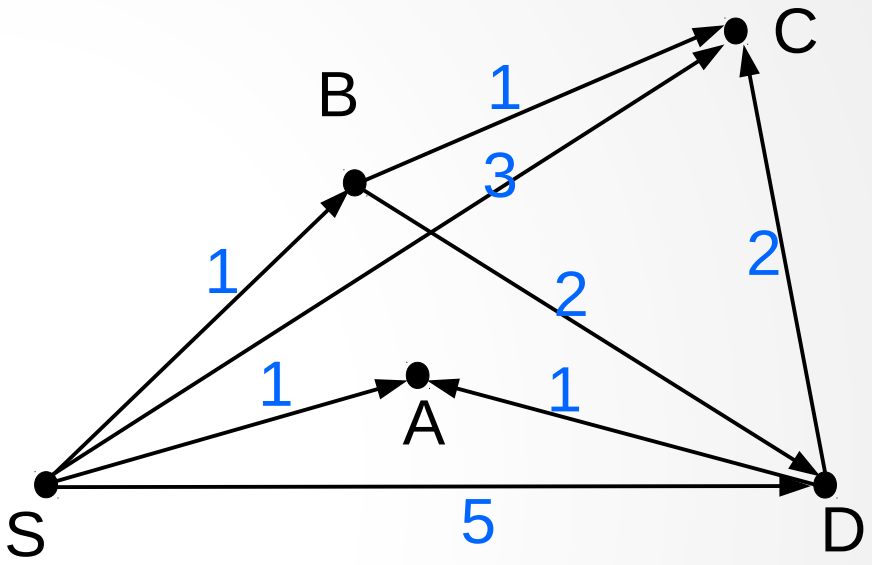
0 ∞ ∞ ∞ ∞

priority queue: S, A, B, C, D

	S	A	B	C	D
parent	None	None	None	None	None
dist.	0	infty	infty	infty	infty

Dijkstra's algorithm for weighted graphs

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*)



0 ∞ ∞ ∞ ∞

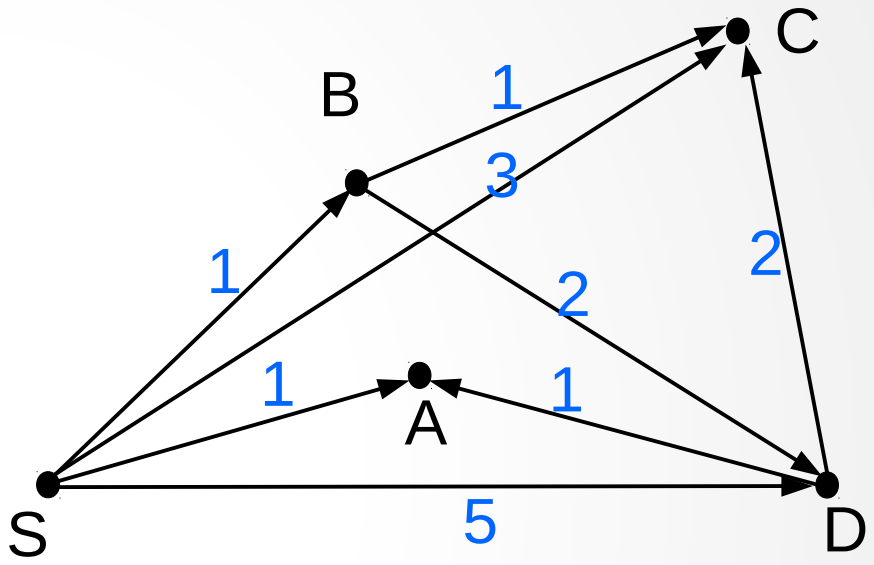
priority queue: ~~S~~, A, B, C, D

dequeued: S
adjacent to S:

	S	A	B	C	D
parent	None	None	None	None	None
dist.	0	infty	infty	infty	infty

Dijkstra's algorithm for weighted graphs

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*)



∞ ∞ ∞ ∞

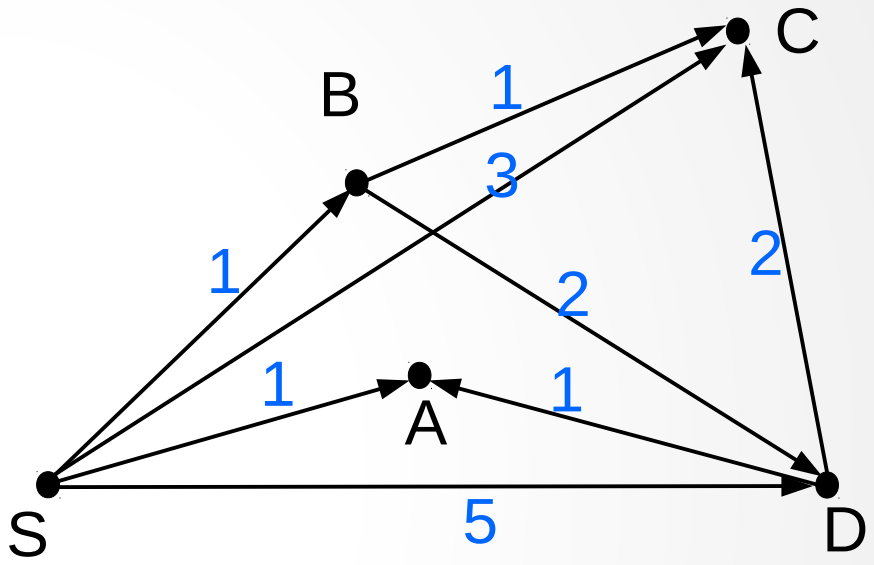
priority queue: A, B, C, D

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

	S	A	B	C	D
parent	None	None	None	None	None
dist.	0	infty	infty	infty	infty

Dijkstra's algorithm for weighted graphs

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*)



∞ ∞ ∞ ∞

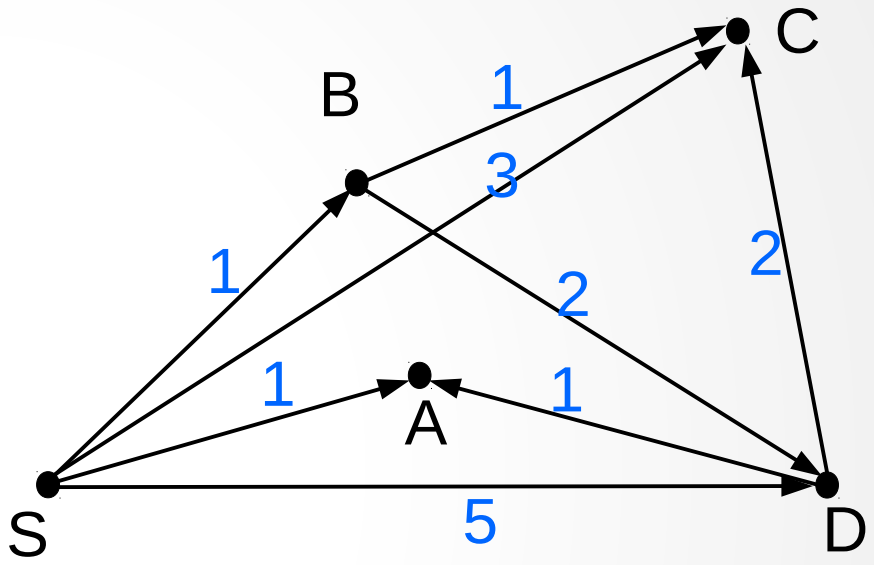
priority queue: A, B, C, D

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

	S	A	B	C	D
parent	None	None	None	None	None
dist.	0	infty	infty	infty	infty

Dijkstra's algorithm for weighted graphs

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*)



1 ∞ ∞ ∞

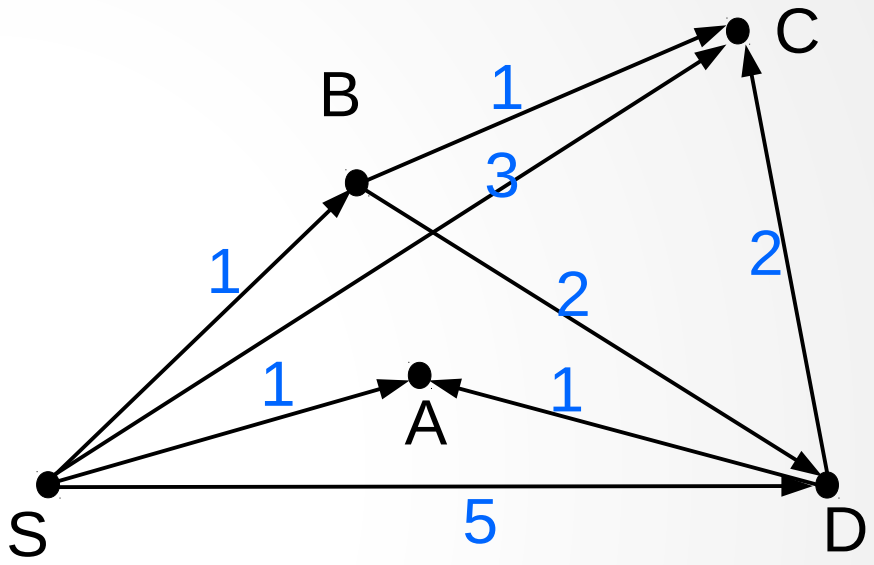
priority queue: A, B, C, D

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

	S	A	B	C	D
parent	None	S	None	None	None
dist.	0	1	infty	infty	infty

Dijkstra's algorithm for weighted graphs

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*)



1 ∞ ∞ ∞

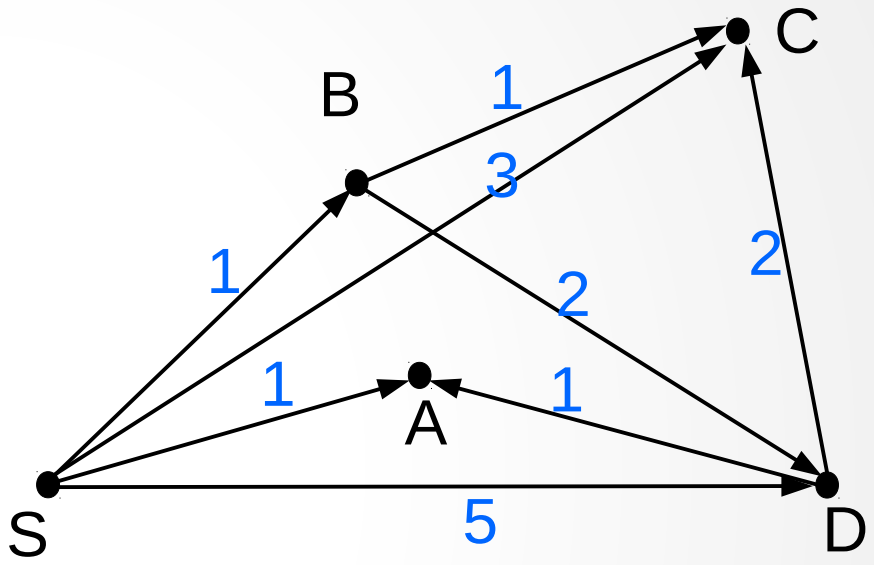
priority queue: A, B, C, D

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

	S	A	B	C	D
parent	None	S	None	None	None
dist.	0	1	infty	infty	infty

Dijkstra's algorithm for weighted graphs

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*)



1 1 ∞ ∞

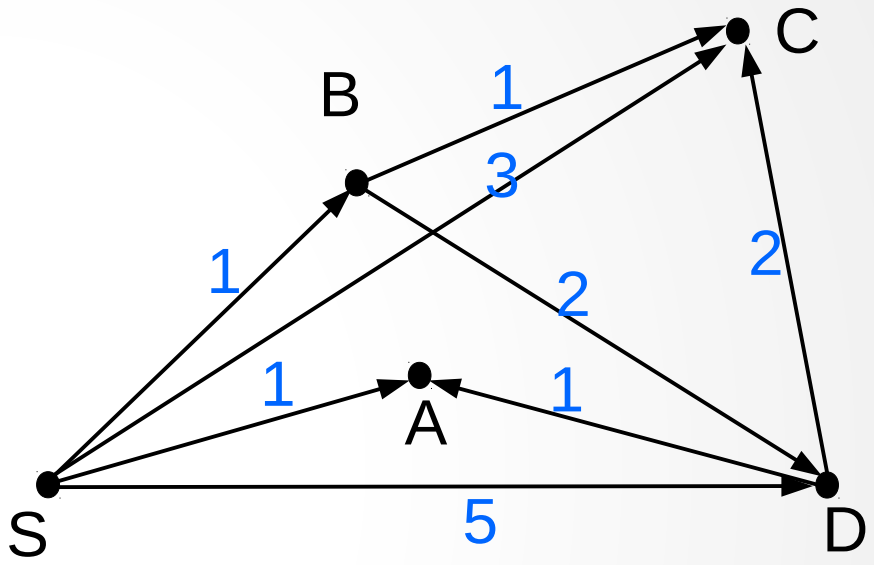
priority queue: A, B, C, D

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

	S	A	B	C	D
parent	None	S	S	None	None
dist.	0	1	1	infty	infty

Dijkstra's algorithm for weighted graphs

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*)



1 1 ∞ ∞

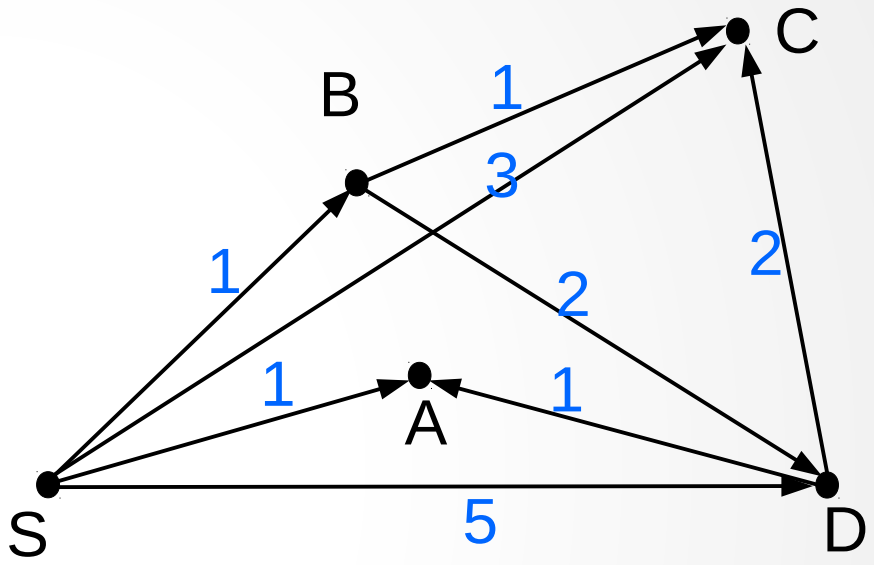
priority queue: A, B, C, D

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

	S	A	B	C	D
parent	None	S	S	None	None
dist.	0	1	1	infty	infty

Dijkstra's algorithm for weighted graphs

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*)



1 1 3 ∞

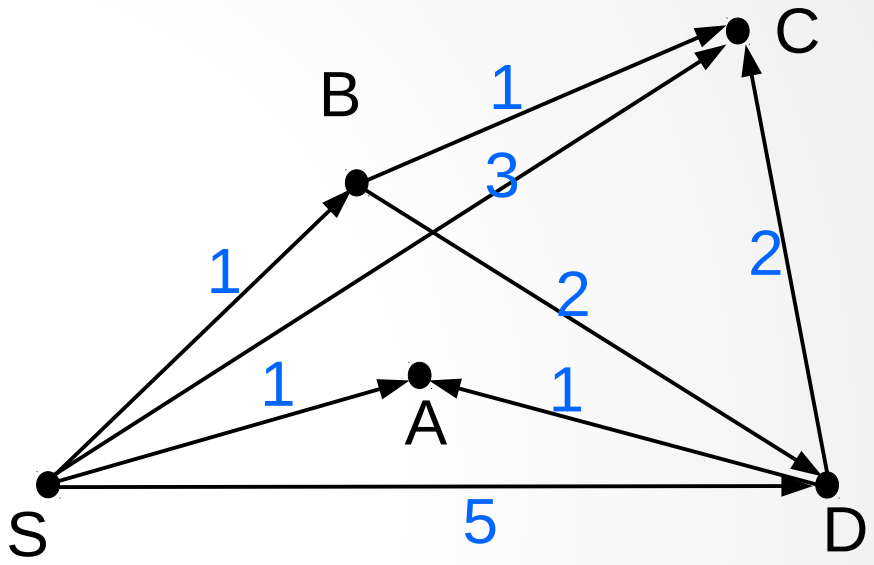
priority queue: A, B, C, D

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

	S	A	B	C	D
parent	None	S	S	S	None
dist.	0	1	1	3	infty

Dijkstra's algorithm for weighted graphs

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*)



1 1 3 ∞

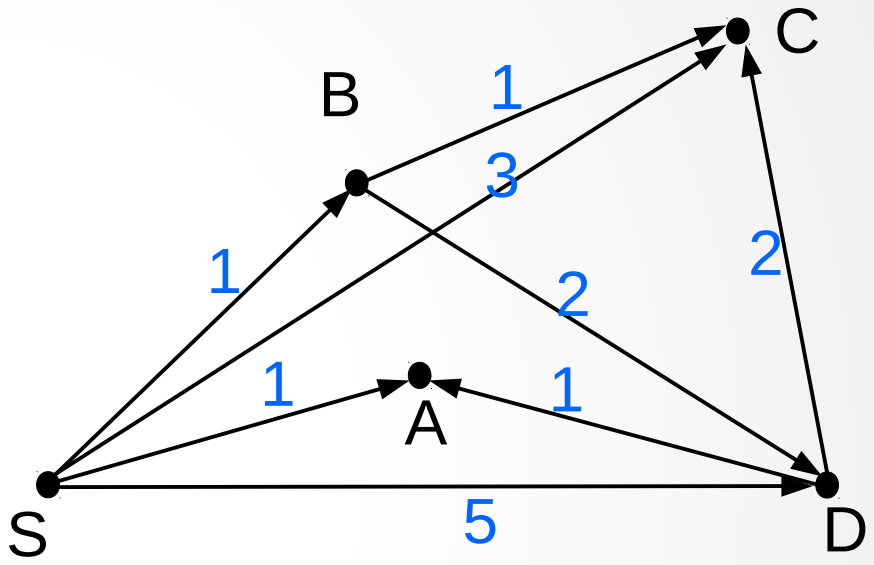
priority queue: A, B, C, D

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

	S	A	B	C	D
parent	None	S	S	S	None
dist.	0	1	1	3	infty

Dijkstra's algorithm for weighted graphs

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*)



1 1 3 5

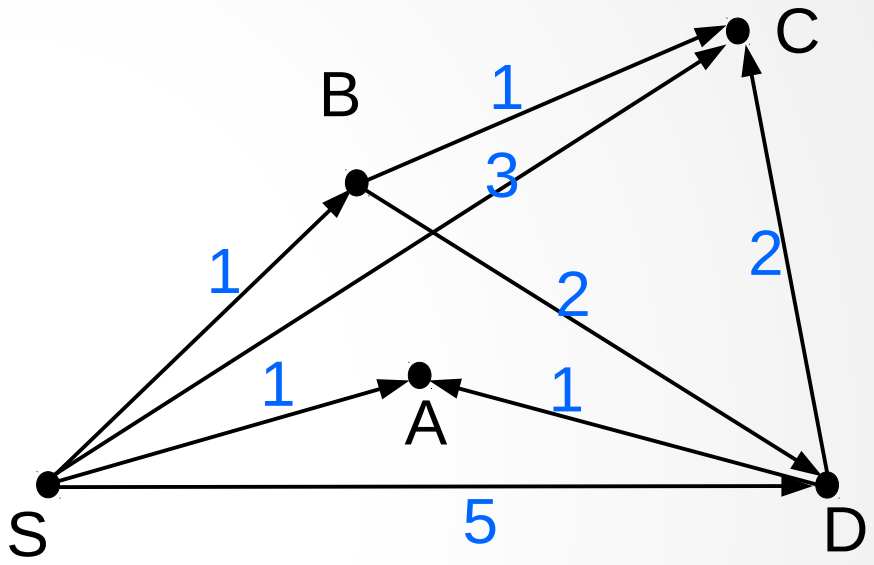
priority queue: A, B, C, D

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

	S	A	B	C	D
parent	None	S	S	S	S
dist.	0	1	1	3	5

Dijkstra's algorithm for weighted graphs

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*)



1 1 3 5

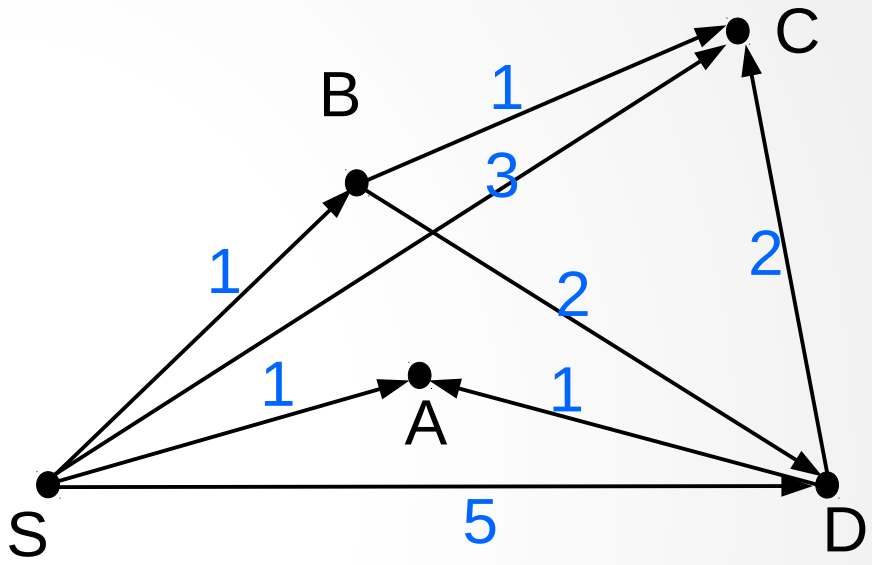
priority queue: A, B, C, D

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

	S	A	B	C	D
parent	None	S	S	S	S
dist.	0	1	1	3	5

Dijkstra's algorithm for weighted graphs

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*)



1 1 3 5

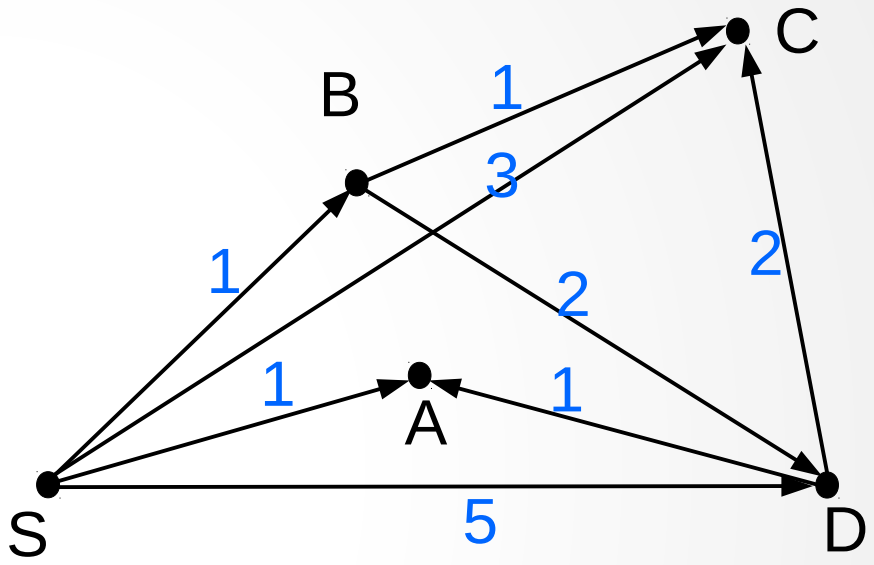
priority queue: A, B, C, D

dequeued:
adjacent to :

	S	A	B	C	D
parent	None	S	S	S	S
dist.	0	1	1	3	5

Dijkstra's algorithm for weighted graphs

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*)



1 1 3 5

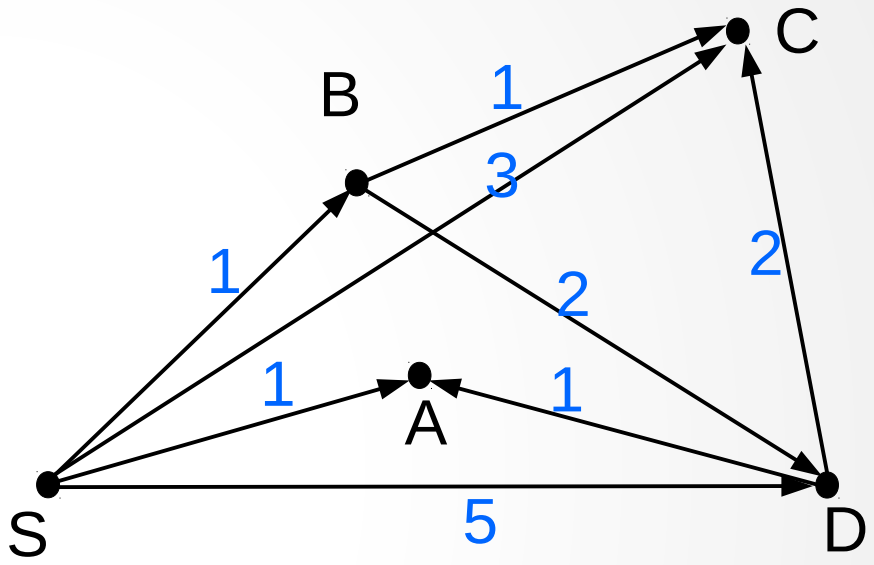
priority queue: ~~A~~, B, C, D

dequeued: A
adjacent to A:

	S	A	B	C	D
parent	None	S	S	S	S
dist.	0	1	1	3	5

Dijkstra's algorithm for weighted graphs

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*)



1 3 5

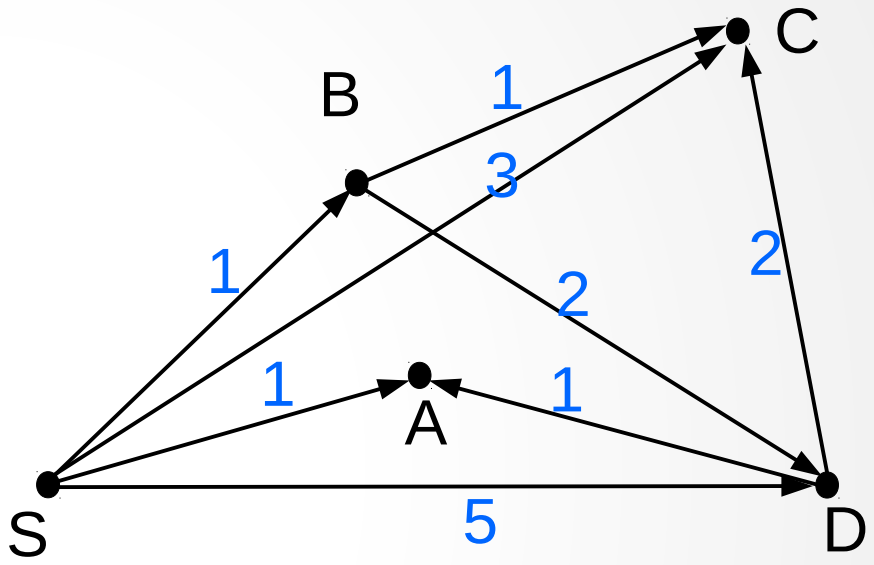
priority queue: B, C, D

dequeued: A
adjacent to A:

	S	A	B	C	D
parent	None	S	S	S	S
dist.	0	1	1	3	5

Dijkstra's algorithm for weighted graphs

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*)



1 3 5

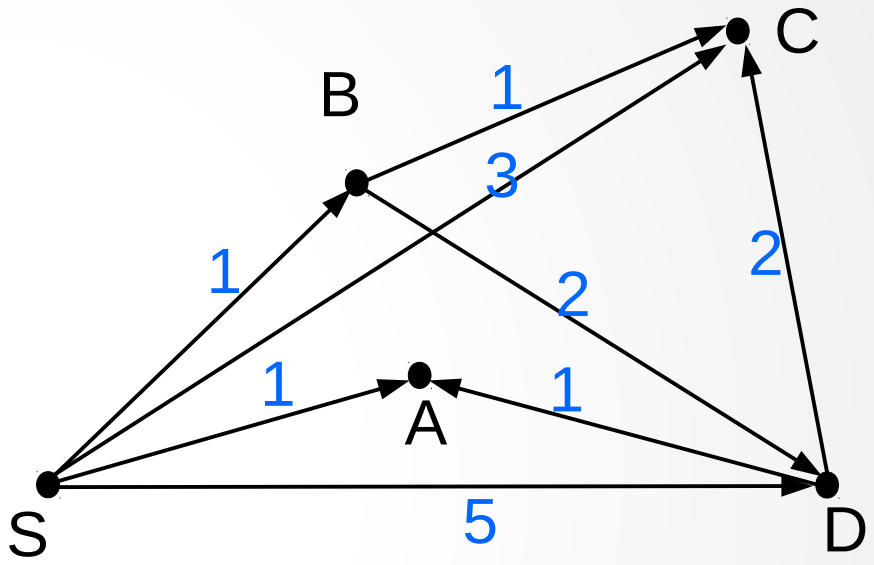
priority queue: B, C, D

dequeued:
adjacent to :

	S	A	B	C	D
parent	None	S	S	S	S
dist.	0	1	1	3	5

Dijkstra's algorithm for weighted graphs

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*)



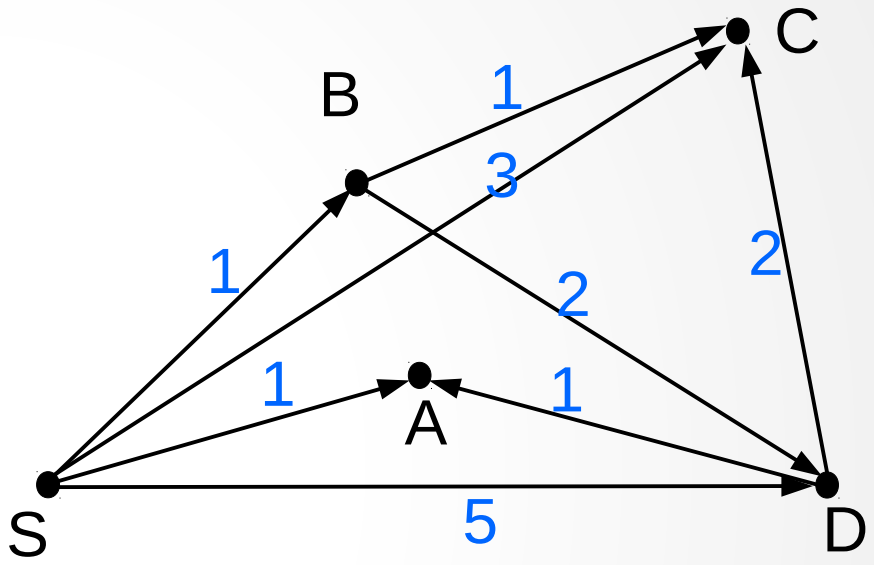
1 3 5
priority queue: ~~B~~, C, D

Dequeued: B
adjacent to B: C,D

	S	A	B	C	D
parent	None	S	S	S	S
dist.	0	1	1	3	5

Dijkstra's algorithm for weighted graphs

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*)



3 5

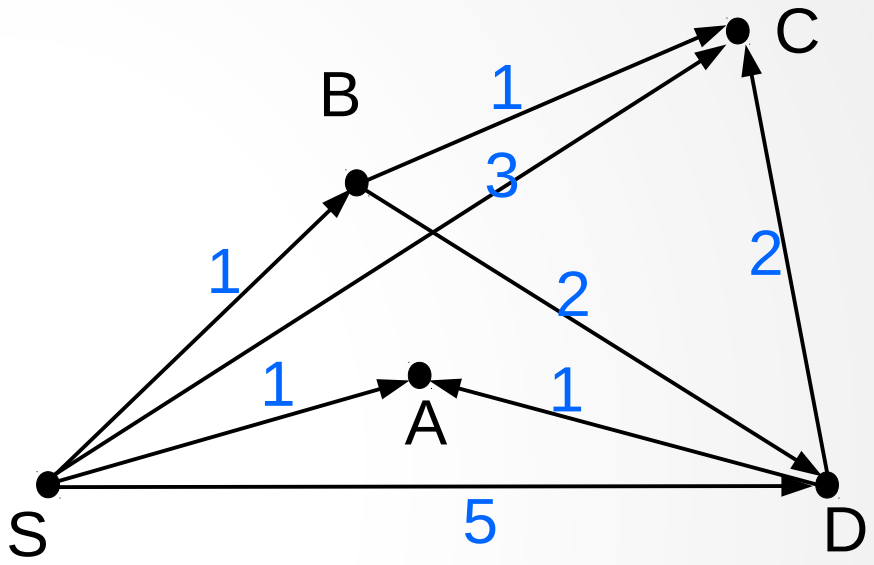
priority queue: C, D

Dequeued: B
adjacent to B: C,D

	S	A	B	C	D
parent	None	S	S	S	S
dist.	0	1	1	3	5

Dijkstra's algorithm for weighted graphs

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*)



3 5

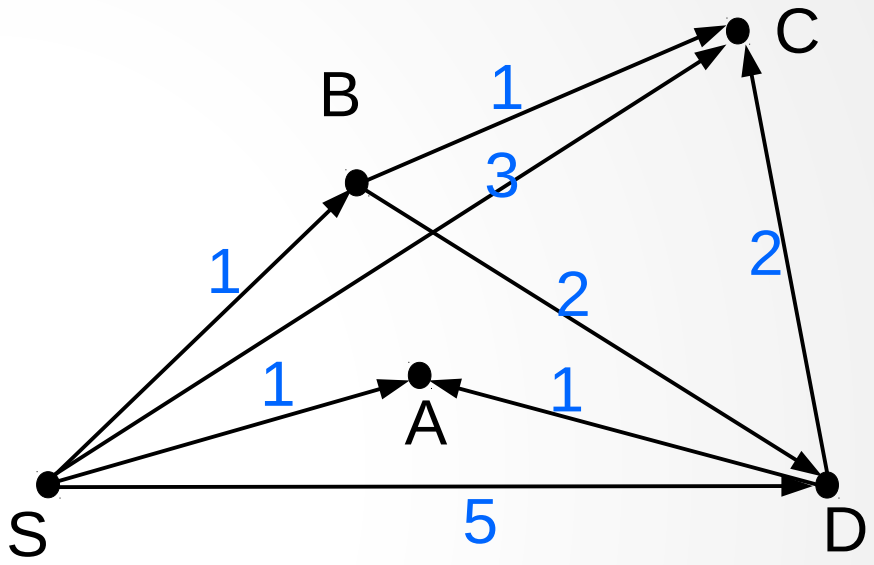
priority queue: C, D

Dequeued: B
adjacent to B: C,D

	S	A	B	C	D
parent	None	S	S	S	S
dist.	0	1	1	3	5

Dijkstra's algorithm for weighted graphs

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*)



3 5

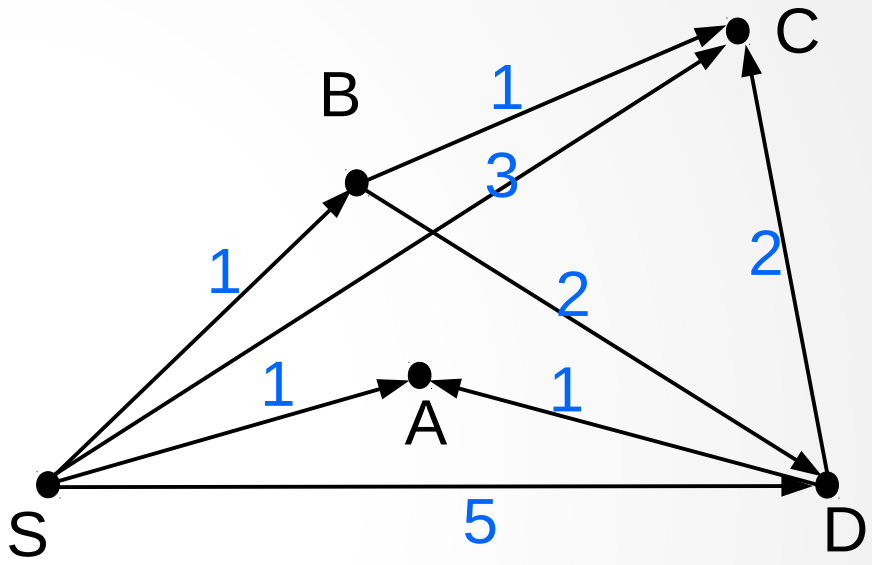
priority queue: C, D

Dequeued: B
adjacent to B: C,D

	S	A	B	C	D
parent	None	S	S	~~S~~ B	S
dist.	0	1	1	~~3~~ 2	5

Dijkstra's algorithm for weighted graphs

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*)



2 5

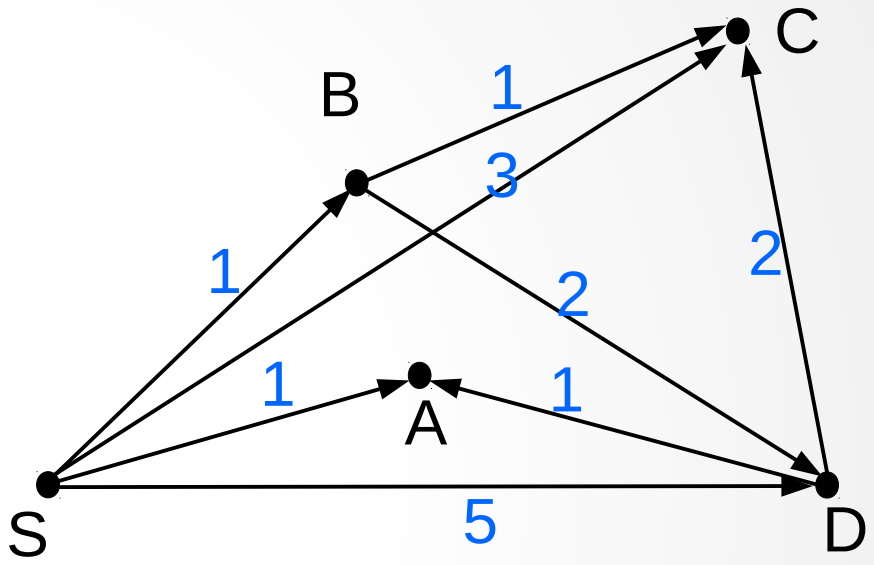
priority queue: C, D

Dequeued: B
adjacent to B: C,D

	S	A	B	C	D
parent	None	S	S	B	S
dist.	0	1	1	2	5

Dijkstra's algorithm for weighted graphs

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*)



2 5

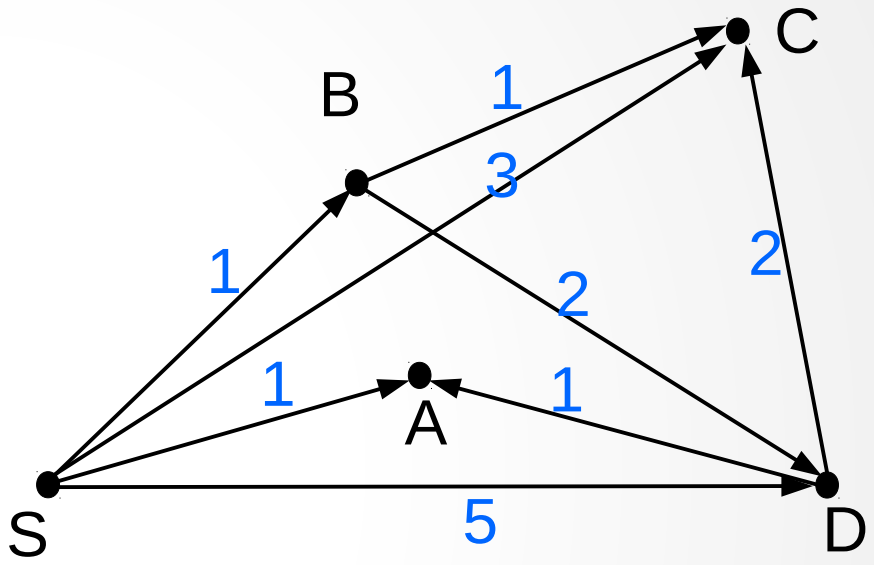
priority queue: C, D

Dequeued: B
adjacent to B: C,D

	S	A	B	C	D
parent	None	S	S	B	~~S~~ B
dist.	0	1	1	2	~~5~~ 3

Dijkstra's algorithm for weighted graphs

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*)



2 3

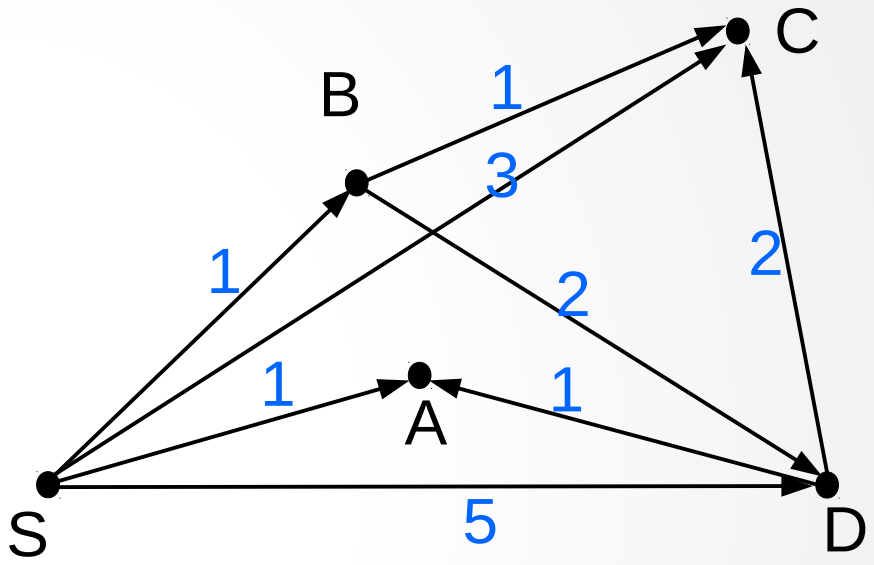
priority queue: C, D

Dequeued: B
adjacent to B: C,D

	S	A	B	C	D
parent	None	S	S	B	B
dist.	0	1	1	2	3

Dijkstra's algorithm for weighted graphs

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*)



2 3

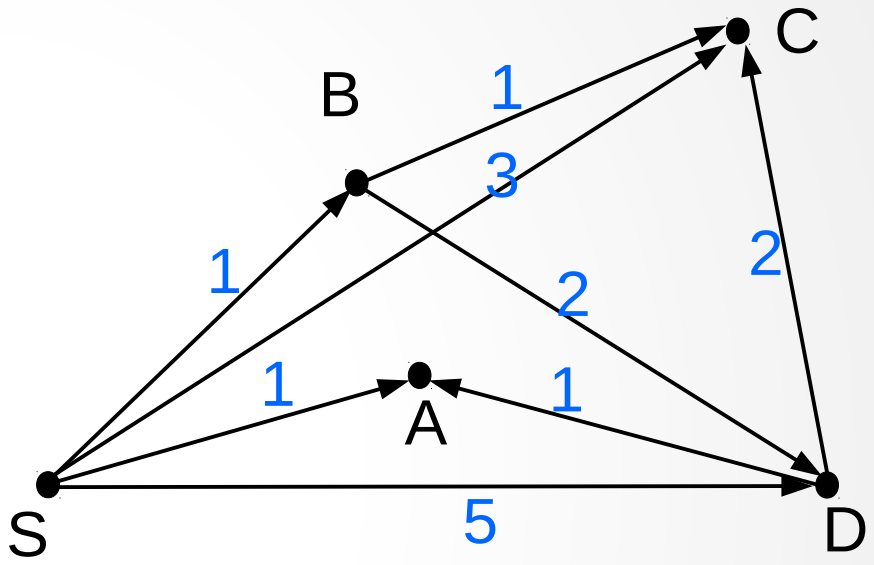
priority queue: C, D

Dequeued: B
adjacent to B: C,D

	S	A	B	C	D
parent	None	S	S	B	B
dist.	0	1	1	2	3

Dijkstra's algorithm for weighted graphs

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*)



2 3

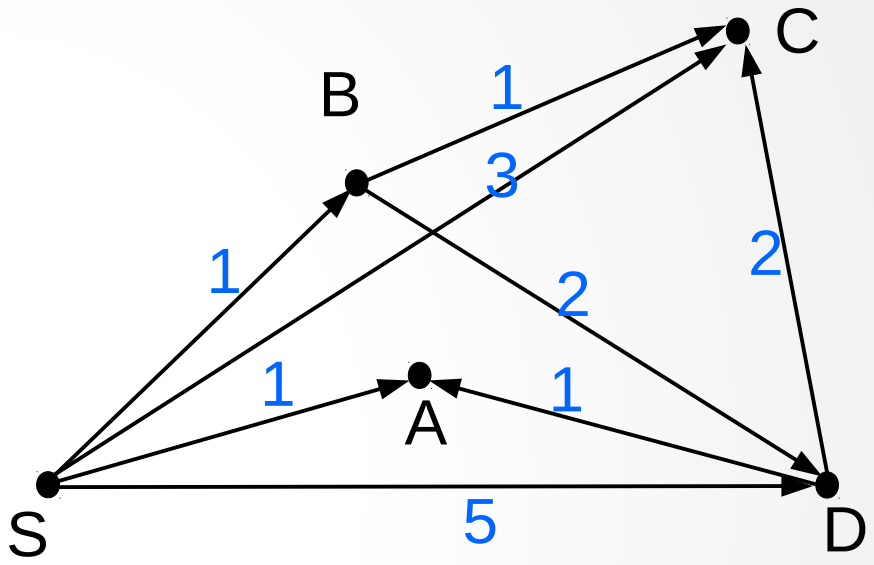
priority queue: C, D

Dequeued:
adjacent to :

	S	A	B	C	D
parent	None	S	S	B	B
dist.	0	1	1	2	3

Dijkstra's algorithm for weighted graphs

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*)



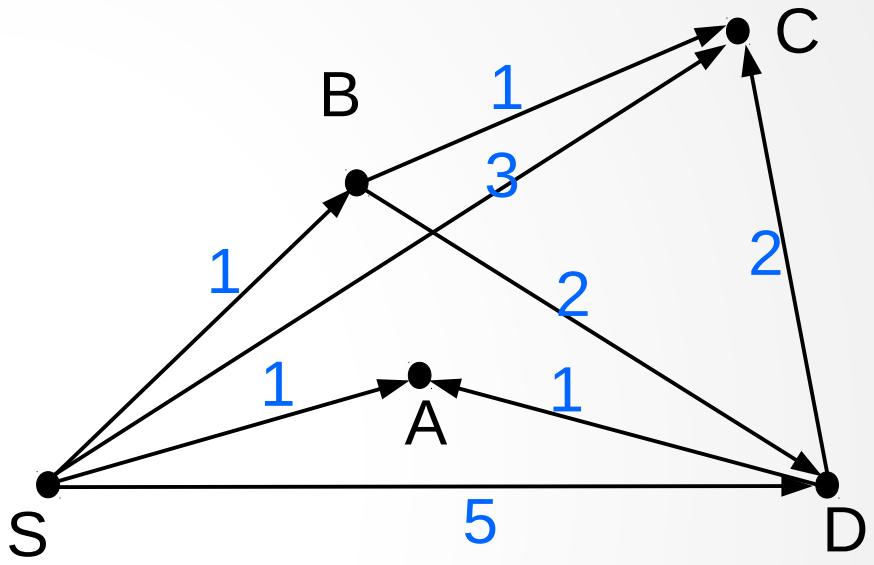
2 3
priority queue: ~~C~~, D

Dequeued: C
adjacent to C:

	S	A	B	C	D
parent	None	S	S	B	B
dist.	0	1	1	2	3

Dijkstra's algorithm for weighted graphs

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*)



3

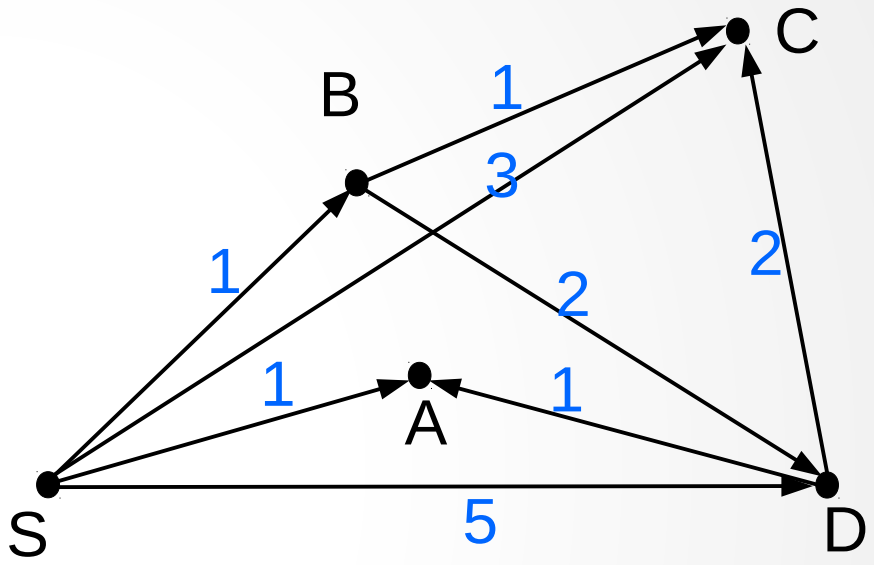
priority queue: D

Dequeued: C
adjacent to C:

	S	A	B	C	D
parent	None	S	S	B	B
dist.	0	1	1	2	3

Dijkstra's algorithm for weighted graphs

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*)



3

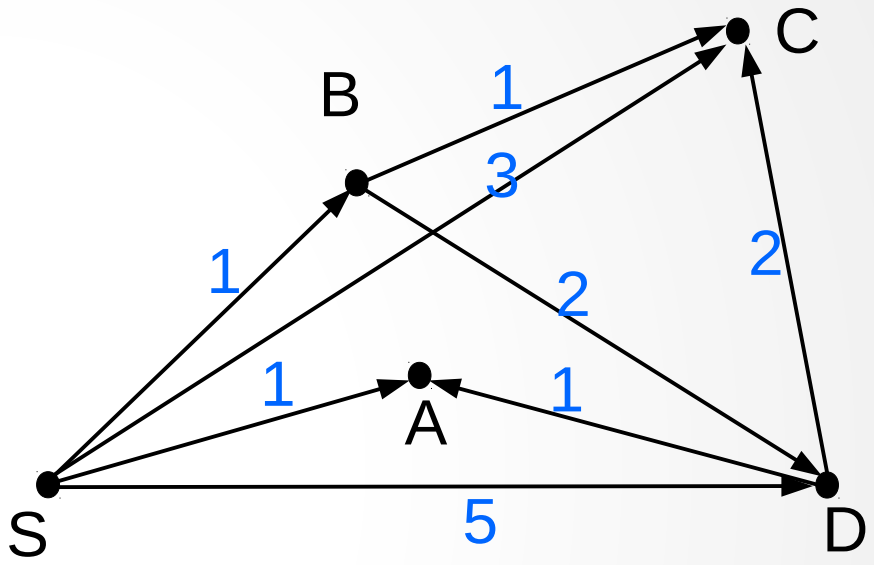
priority queue: D

Dequeued:
adjacent to :

	S	A	B	C	D
parent	None	S	S	B	B
dist.	0	1	1	2	3

Dijkstra's algorithm for weighted graphs

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*)



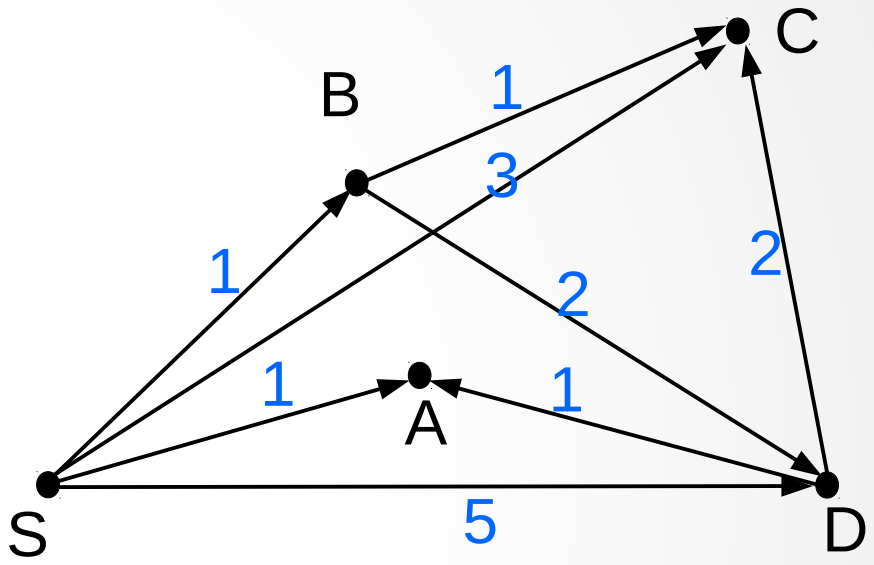
3
priority queue: ~~D~~

Dequeued: D
adjacent to D: A

	S	A	B	C	D
parent	None	S	S	B	B
dist.	0	1	1	2	3

Dijkstra's algorithm for weighted graphs

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*)



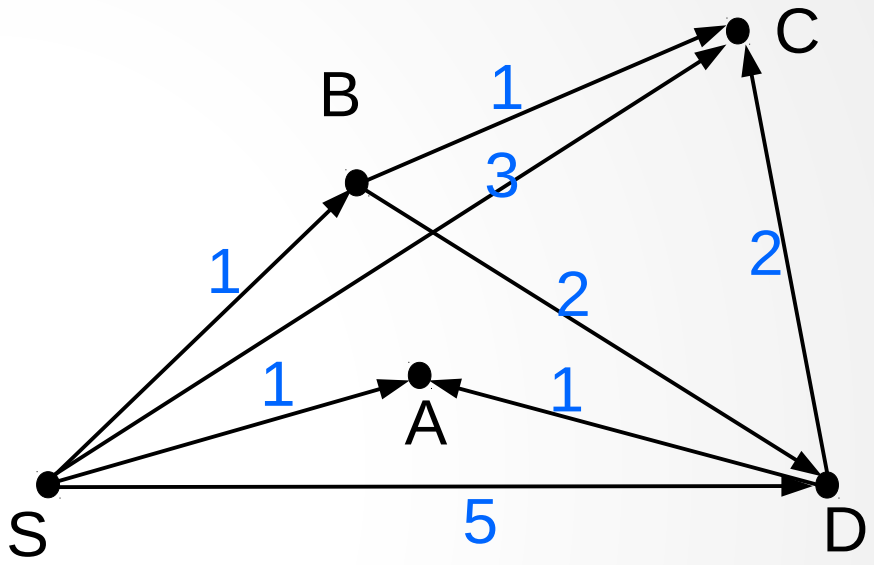
Dequeued: D
adjacent to D: A

priority queue:

	S	A	B	C	D
parent	None	S	S	B	B
dist.	0	1	1	2	3

Dijkstra's algorithm for weighted graphs

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*)



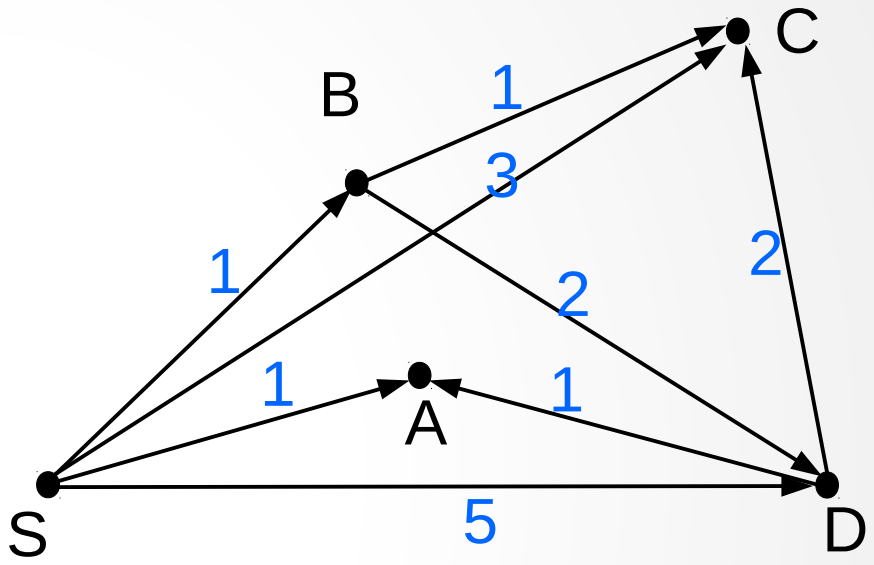
Dequeued: D
adjacent to D: A

priority queue:

	S	A	B	C	D
parent	None	S	S	B	B
dist.	0	1	1	2	3

Dijkstra's algorithm for weighted graphs

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*)



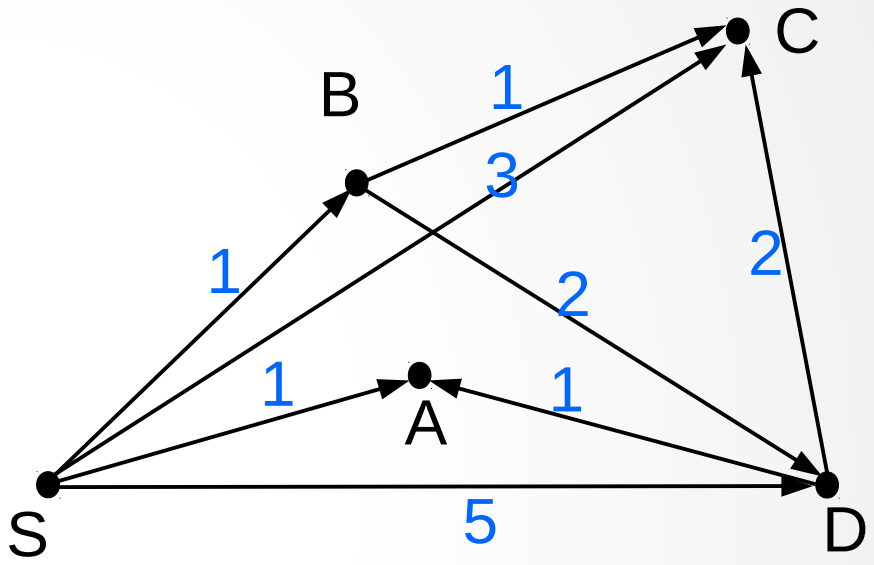
Dequeued: D
adjacent to D: A

priority queue:

	S	A	B	C	D
parent	None	S	S	B	B
dist.	0	1	1	2	3

Dijkstra's algorithm for weighted graphs

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*)



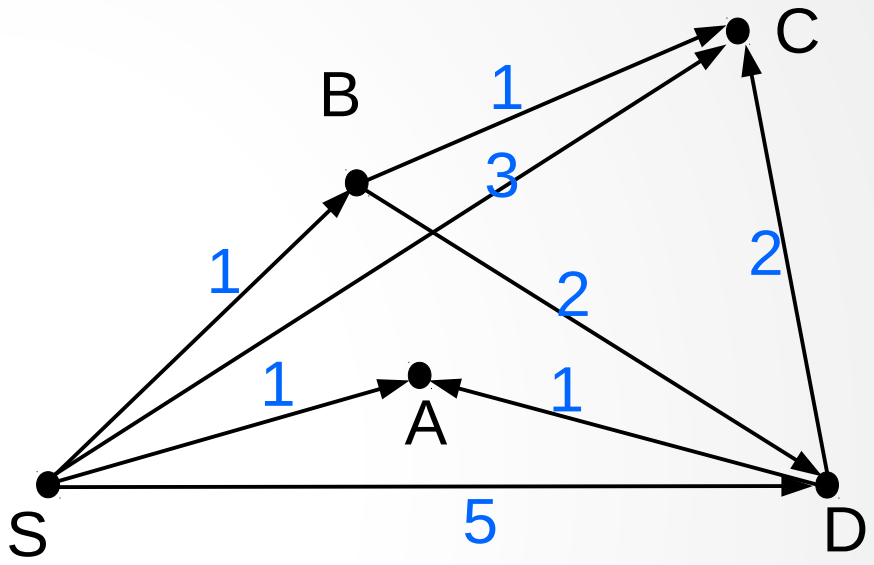
Dequeued:
adjacent to :

priority queue:

	S	A	B	C	D
parent	None	S	S	B	B
dist.	0	1	1	2	3

Dijkstra's algorithm for weighted graphs

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*)



STOP

priority queue:

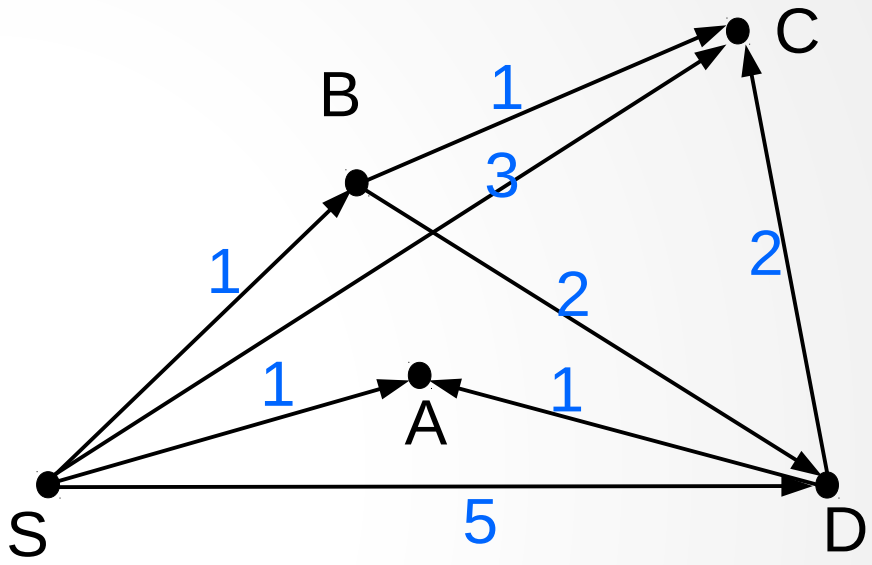
	S	A	B	C	D
parent	None	S	S	B	B
dist.	0	1	1	2	3

Dijkstra's algorithm for weighted graphs

The table is ready to be used.

For example,
the shortest path from S to D is
S → B → D

the shortest path from S to C is
S → B → C



	S	A	B	C	D
parent	None	S	S	B	B
dist.	0	1	1	2	3