Introduction to Algorithms and Data Structures

Dynamic Programming - The Bellman-Ford Algorithm for Shortest Paths

Shortest Paths in Graphs (Lecture 17)

- **Input:** A directed graph $G = (V, E)$, and a designated node s in V. We also assume that every node u in V is reachable from s. We are also given a length $\ell_e > 0$ for every edge e in E.

- **Output:** For every node u in V, a shortest path s~u from s to u.

Shortest Paths in Graphs (today)

- **Input:** A directed graph $G = (V, E)$, and designated nodes s, t in V. We also assume that every node u in V is reachable from s. We are also given a cost $c_e \in \mathbb{R}$ for every edge e in E.

- **Output:** A shortest path s~t from s to t.

Shortest Paths in Graphs (today)

Shortest Paths in Graphs (today)

- The difference is that the edge "lengths" can be *positive* or *negative*. In this context they are better interpreted as *costs*, and denoted by c_e or c_{uv}.

Shortest Paths in Graphs (today)

- The difference is that the edge “lengths” can be *positive* or *negative*. In this context they are better interpreted as *costs*, and denoted by c_e or c_{uv}.

- Motivation: e.g., Financial Networks

  positive costs (*costs* of transactions)
  negative costs (*profits* of transactions)

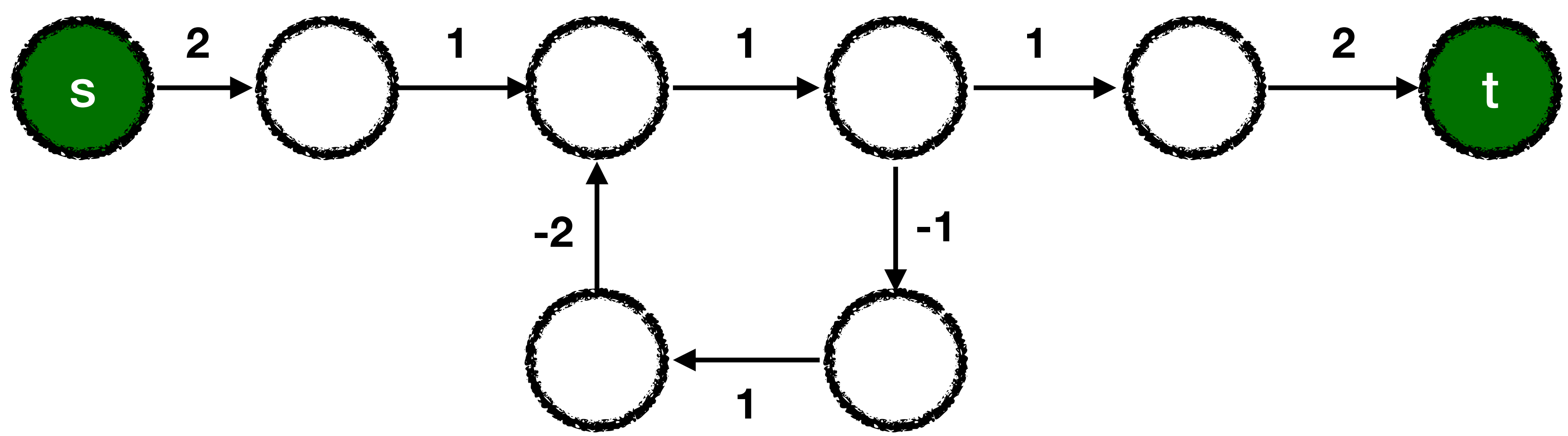
Shortest Paths in Graphs (today)

- **Input:** A directed graph $G = (V, E)$, and designated nodes s, t in V. We also assume that every node u in V is reachable from s. We are also given a cost $c_e \in \mathbb{R}$ for every edge e in E.

- **Output:** A shortest path $s \sim t$ from s to t. In other words, a path P that minimises $\sum_{(u,v) \in P} c_{uv}$

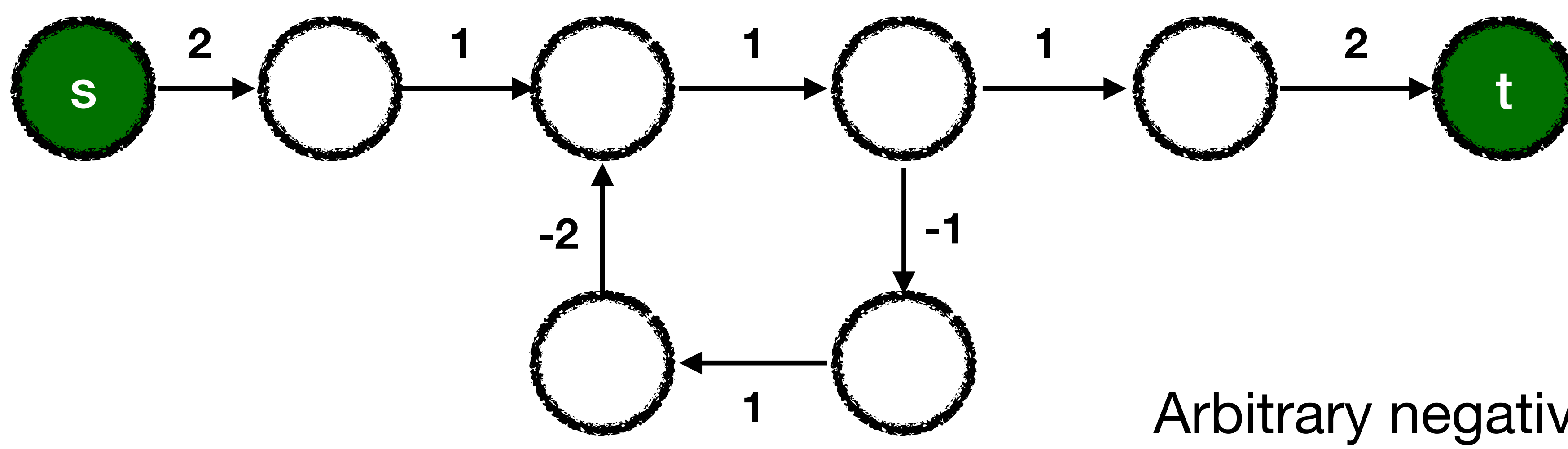
Negative Cycles

- Can we find a shortest path in the following graph?



Negative Cycles

- Can we find a shortest path in the following graph?



Arbitrary negative cost!

Shortest Paths in Graphs

- **Input:** A directed graph $G = (V, E)$, and designated nodes s, t in V. We also assume that every node u in V is reachable from s, and *that the graph does not have any negative cycles.* We are also given a cost $c_e \in \mathbb{R}$ for every edge e in E.

- **Output:** A shortest path $s \sim t$ from s to t. In other words, a path P that minimises $\sum_{(u,v) \in P} c_{uv}$

Why not Dijkstra?

Dijkstra's Algorithm

- For every node $v \in V - S$, we determine the shortest path that can be constructed by traveling along a path $s \sim u$ for $u \in S$, followed by (u, v).

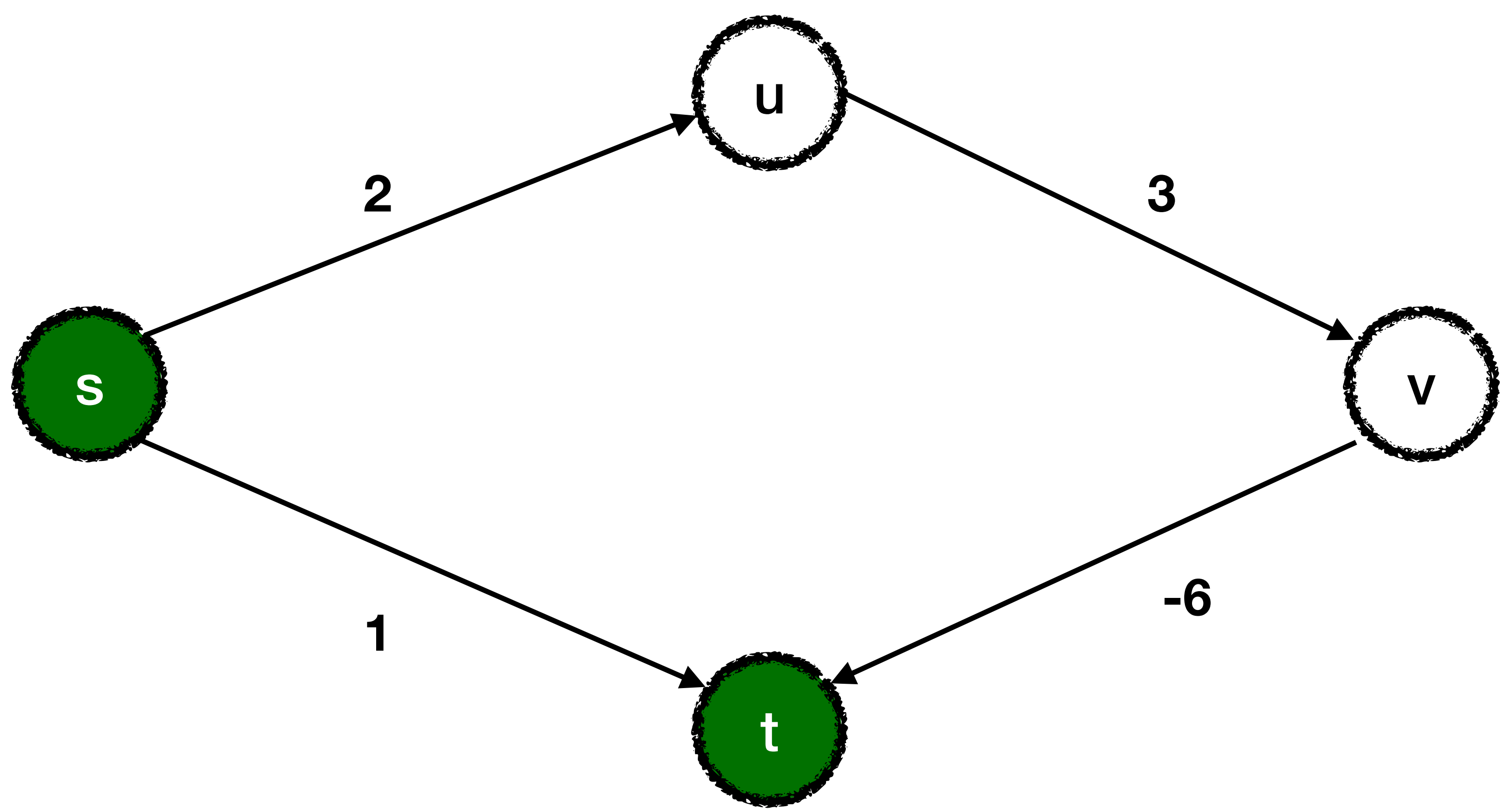
- In other words, we choose node $v \in V - S$ such that

$$d'(v) = \min_{e=(u,v):u \in S} d(u) + \ell_e$$

- Add v to S and define $d(v) = d'(v)$.

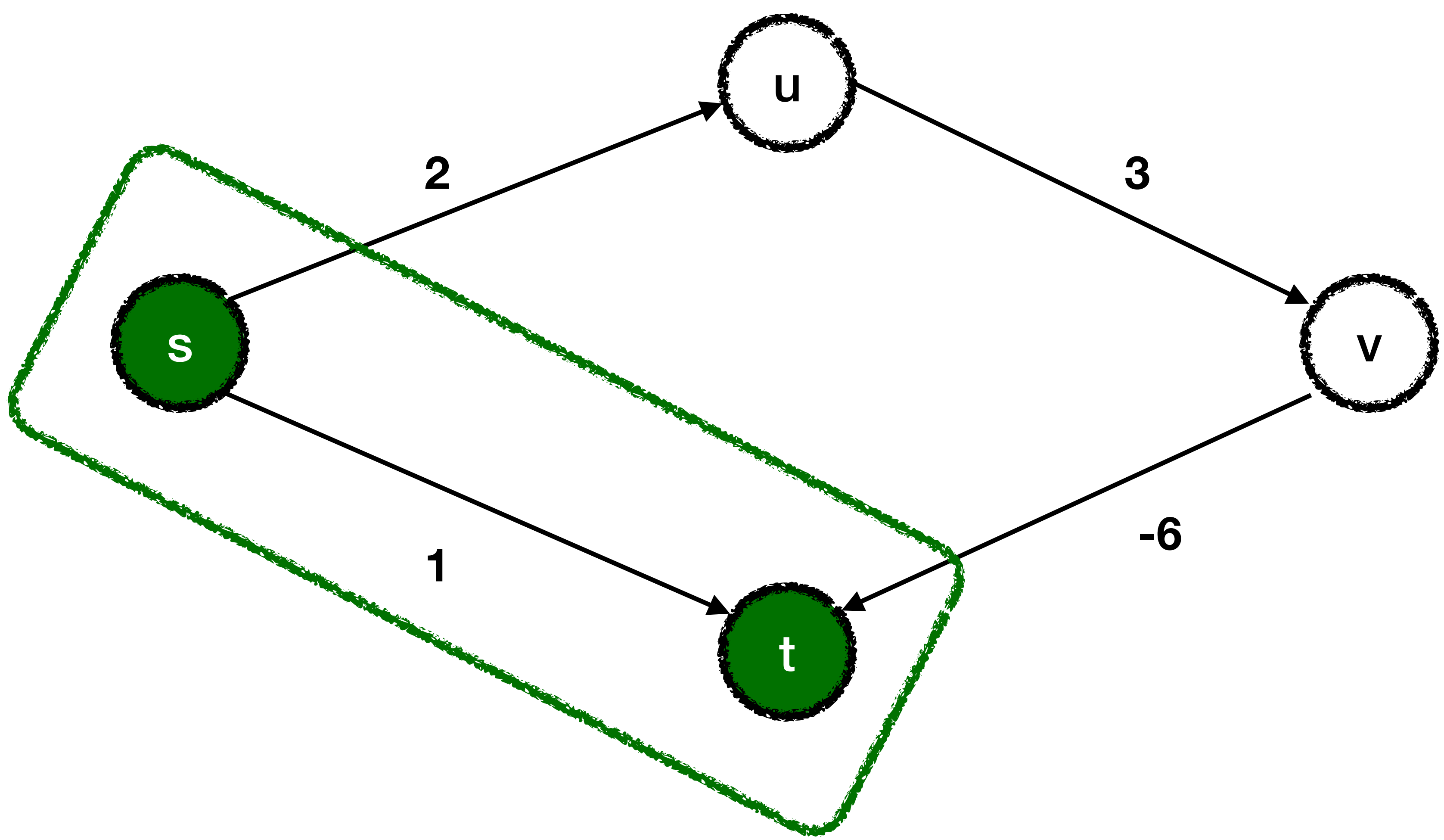
Why not Dijkstra?

- Which node would Dijkstra add in the following graph?



Why not Dijkstra?

- Which node would Dijkstra add in the following graph?

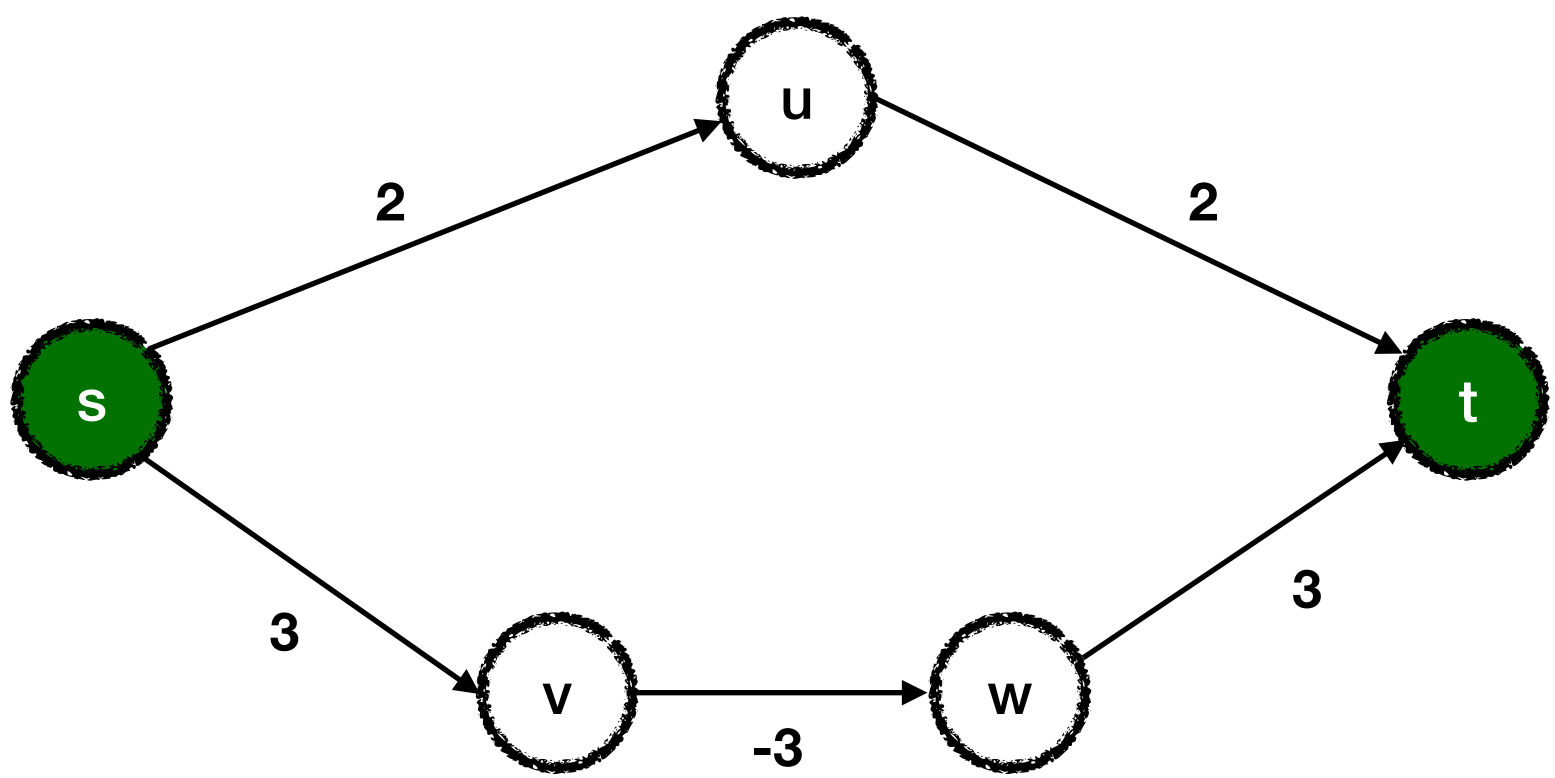


Maybe modified Dijkstra?

- **Idea:** "Get rid" of the negative costs by adding a large number $\mathscr{M}$ to all the edge costs.

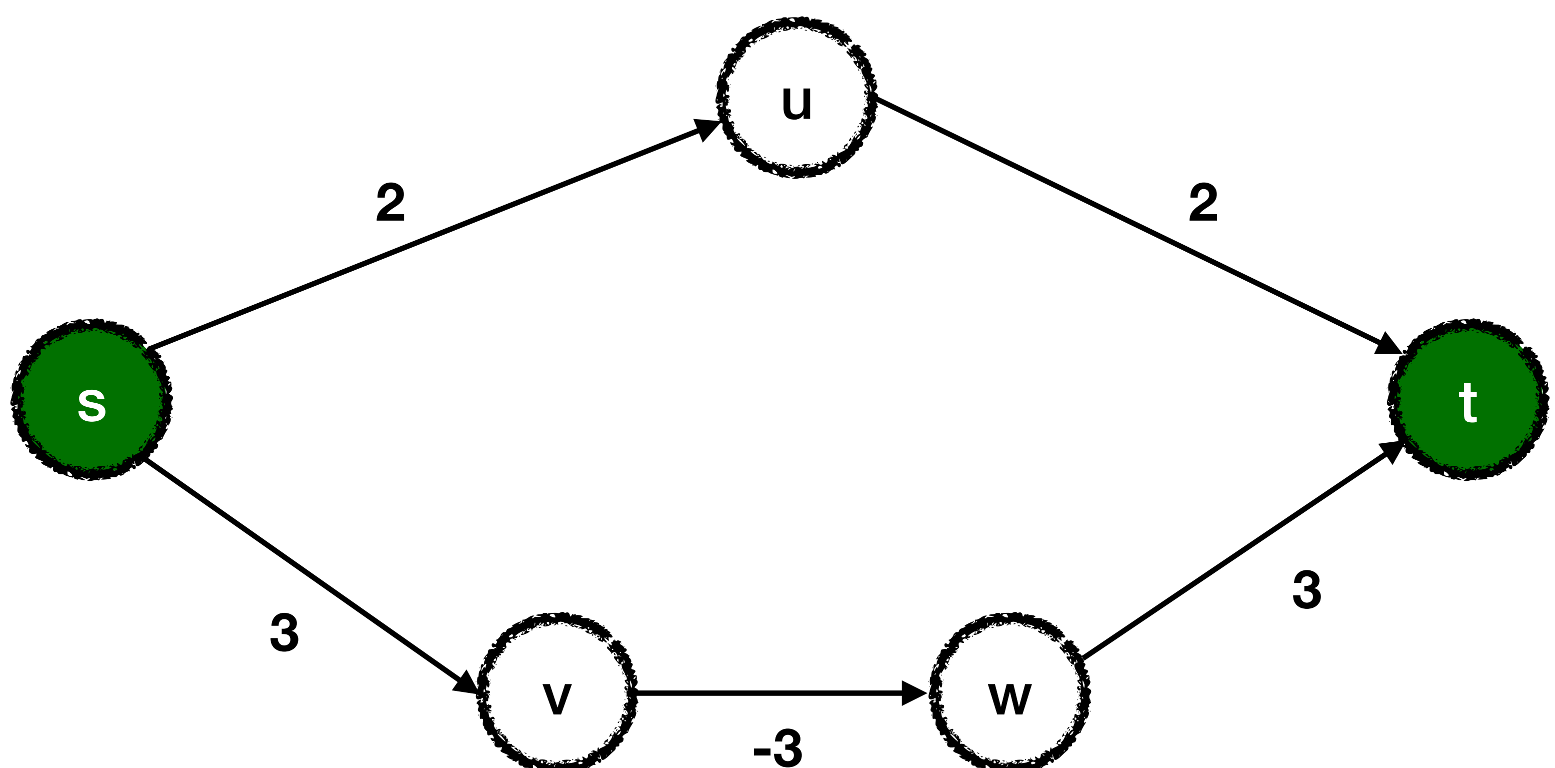
Maybe modified Dijkstra?

- **Idea:** "Get rid" of the negative costs by adding a large number $\mathscr{M}$ to all the edge costs.



Maybe modified Dijkstra?

- **Idea:** “Get rid” of the negative costs by adding a large number $\mathscr{M}$ to all the edge costs.



The shortest path changes!

A dynamic programming approach

- The algorithm that we will present next was developed by Bellman (1958) and Ford (1956).

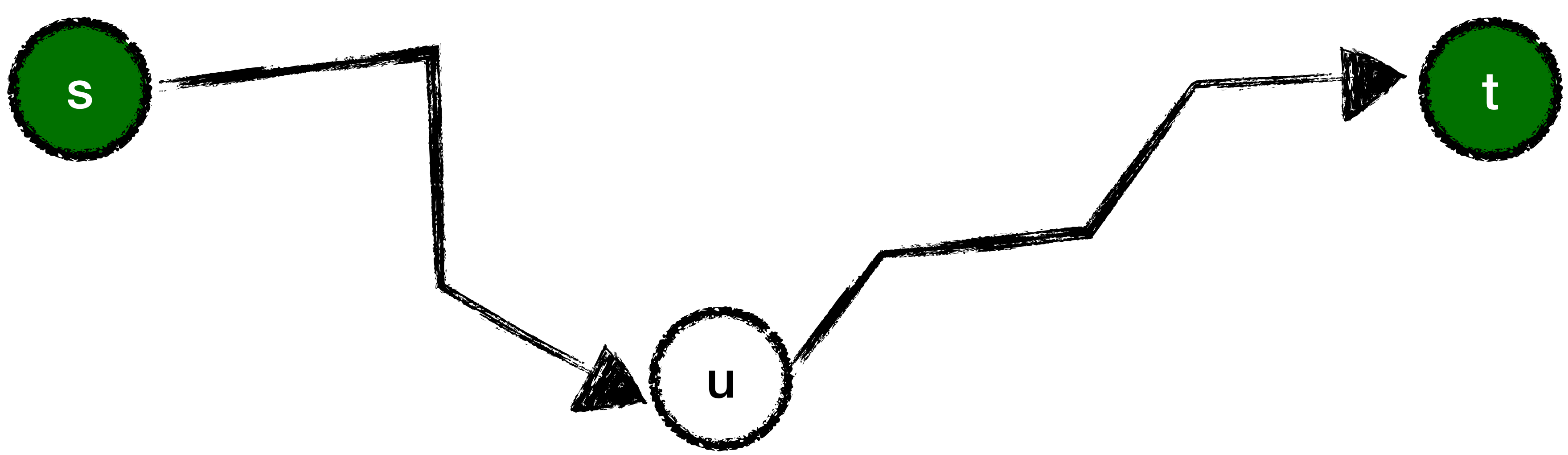
- Note that Dijkstra’s algorithm was published in 1959.

Why dynamic programming?

Let's look at a shortest path $s \sim t$ from s to t.

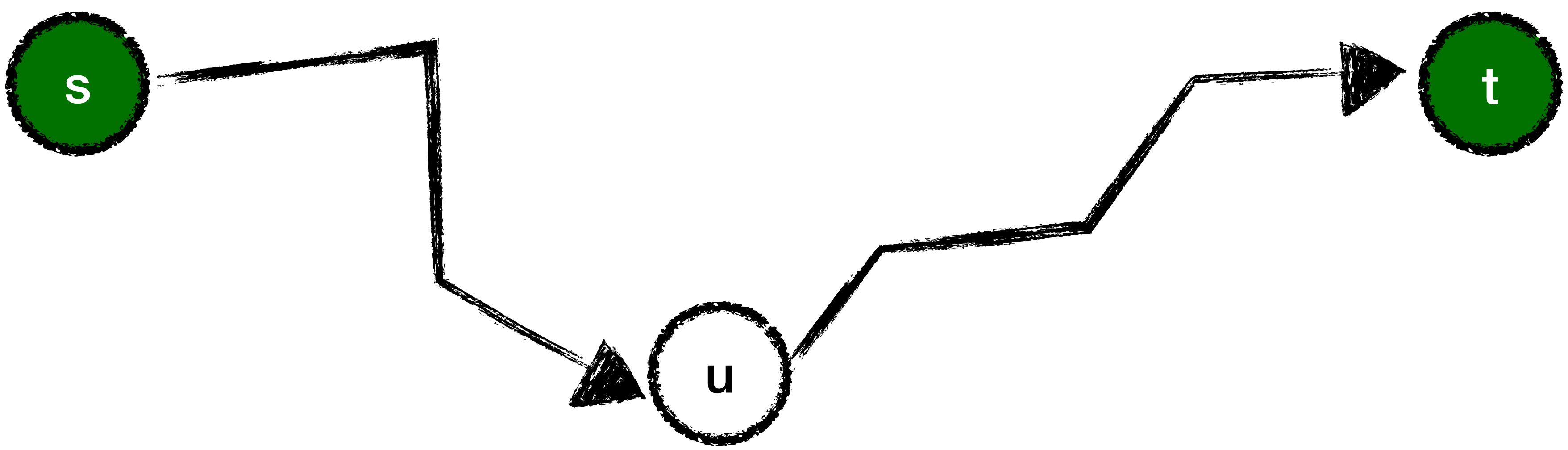
Why dynamic programming?

Let's look at a shortest path $s \sim t$ from s to t.



Why dynamic programming?

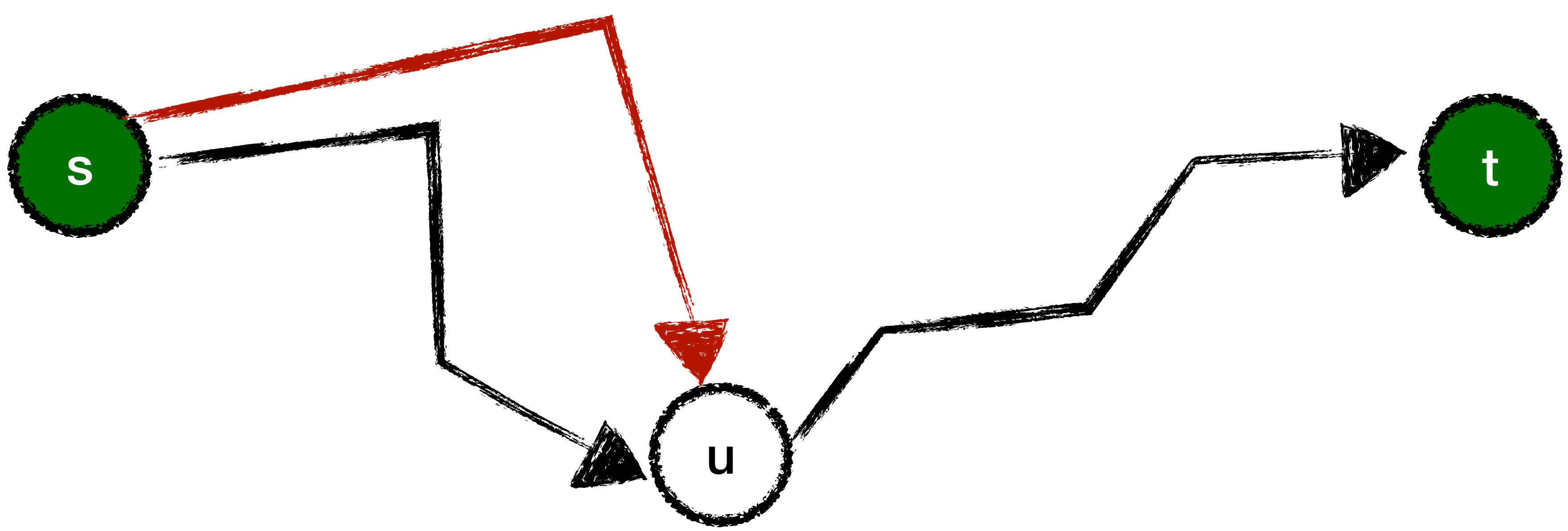
Let’s look at a shortest path s~t from s to t.



This consists of a shortest path s~u from s to t, and a shortest path u~t from u to t (why?).

Why dynamic programming?

Let's look at a shortest path $s{\sim}t$ from s to t.



This consists of a shortest path $s{\sim}u$ from s to t, and a shortest path $u{\sim}t$ from u to t (why?).

Why dynamic programming?

Let's look at a shortest path $s \sim t$ from s to t.

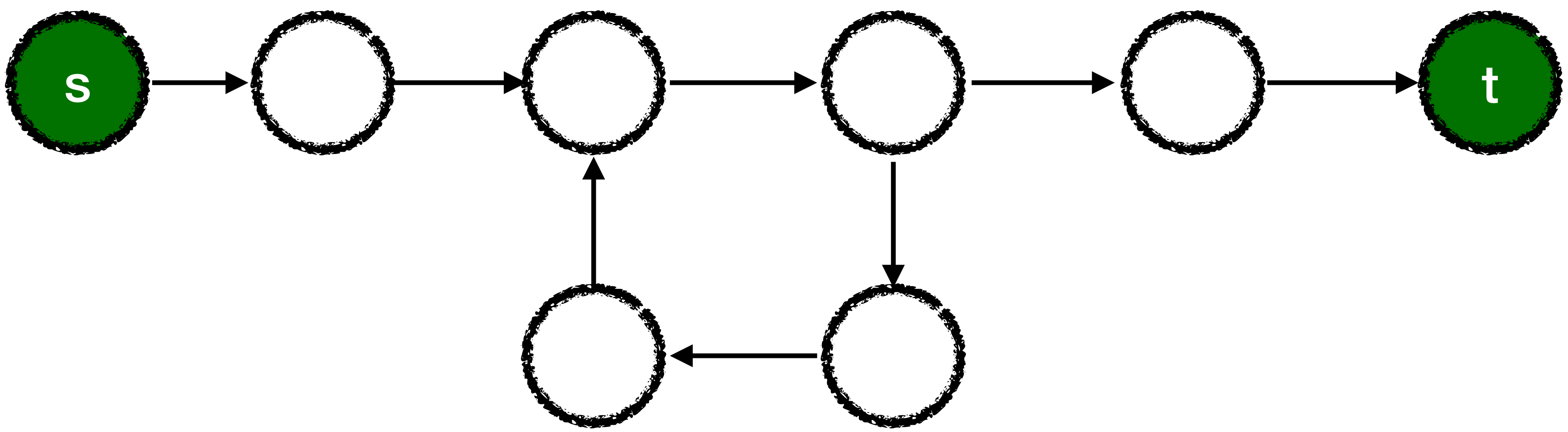
This consists of a shortest path $s \sim u$ from s to t, and a shortest path $u \sim t$ from u to t (why?).

Simple Observation

- **Observation:** If a graph *does not have any negative cycles*, then there is a shortest path $s \sim t$ from s to t that is simple, i.e., it does not repeat any nodes.

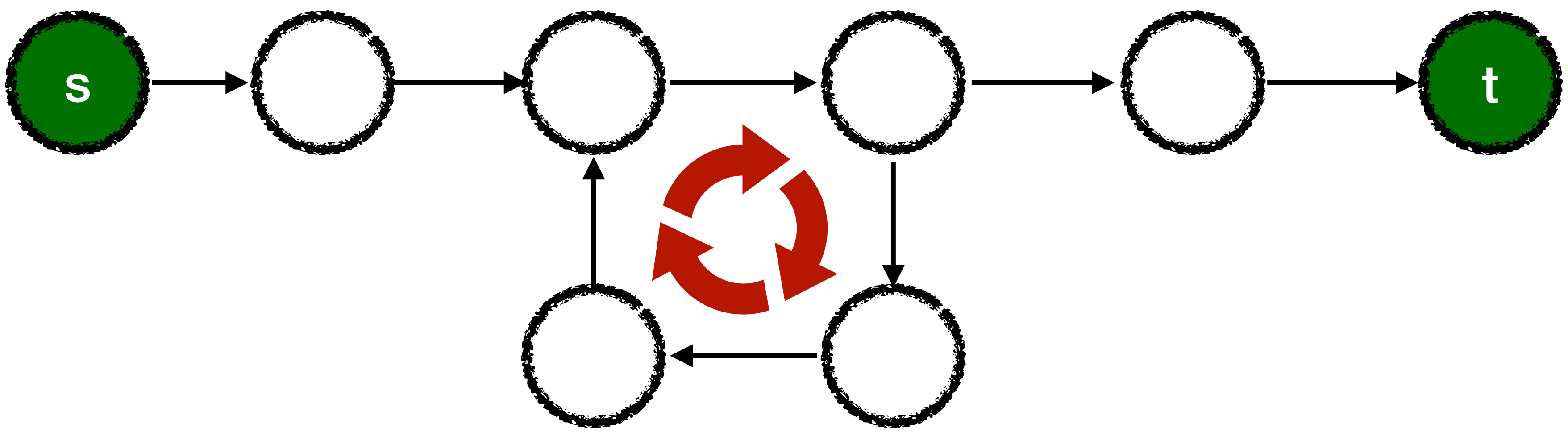
Simple Observation

- **Observation:** If a graph *does not have any negative cycles*, then there is a shortest path $s{\sim}t$ from s to t that is simple, i.e., it does not repeat any nodes.



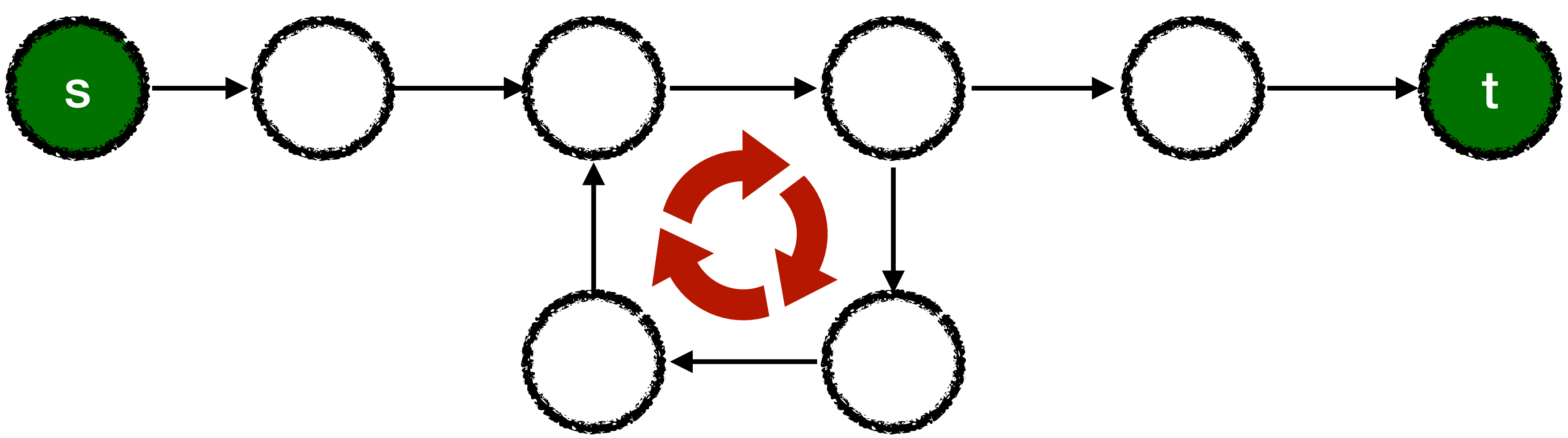
Simple Observation

- **Observation:** If a graph *does not have any negative cycles*, then there is a shortest path $s{\sim}t$ from s to t that is simple, i.e., it does not repeat any nodes.



Simple Observation

- **Observation:** If a graph *does not have any negative cycles*, then there is a shortest path $s \sim t$ from s to t that is simple, i.e., it does not repeat any nodes.



Adding the cycle cannot make the path shorter!

Simple Observation

- **Observation:** If a graph *does not have any negative cycles*, then there is a shortest path $s \sim t$ from s to t that is simple, i.e., it does not repeat any nodes.

Simple Observation

- **Observation:** If a graph *does not have any negative cycles*, then there is a shortest path s~t from s to t that is simple, i.e., it does not repeat any nodes.

- **Corollary:** The length of any shortest path s~t from s to t has at most $n - 1$ edges.

Setting up our subproblems

Setting up our subproblems

- Previously:

  Subset Sum: OPT(i,w) was the value of the optimal solution on the first i items and weight w.

  Weighted Interval Scheduling: OPT(i) was the value of the optimal solution on the first i intervals.

Setting up our subproblems

- Previously:

  Subset Sum: OPT(i,w) was the value of the optimal solution on the first i items and weight w.

  Weighted Interval Scheduling: OPT(i) was the value of the optimal solution on the first i intervals.

- We could try something similar for the “first” i nodes.

Setting up our subproblems

- Previously:

  Subset Sum: $OPT(i,w)$ was the value of the optimal solution on the first i items and weight w.

  Weighted Interval Scheduling: $OPT(i)$ was the value of the optimal solution on the first i intervals.

- We could try something similar for the “first” i nodes.
  - Could be made to work, but it seems complicated.

Setting up our subproblems

- Previously:

  Subset Sum: OPT(*i*,w) was the value of the optimal solution on the first *i* items and weight w.

  Weighted Interval Scheduling: OPT(*i*) was the value of the optimal solution on the first *i* intervals.

- We could try something similar for the “first” *i* nodes.
  - Could be made to work, but it seems complicated.
  - Instead, we will use the *number of edges*, rather than the set of nodes or edges.

Setting up our subproblems

Setting up our subproblems

- Let OPT(i,v) denote the minimum cost of a path v~t from node v to t that uses at most i edges.

Setting up our subproblems

- Let OPT(i,v) denote the minimum cost of a path v~t from node v to t that uses at most i edges.

- We could also use OPT(i,v) to denote the minimum cost of a path s~v from s to node v that uses at most i edges.

Setting up our subproblems

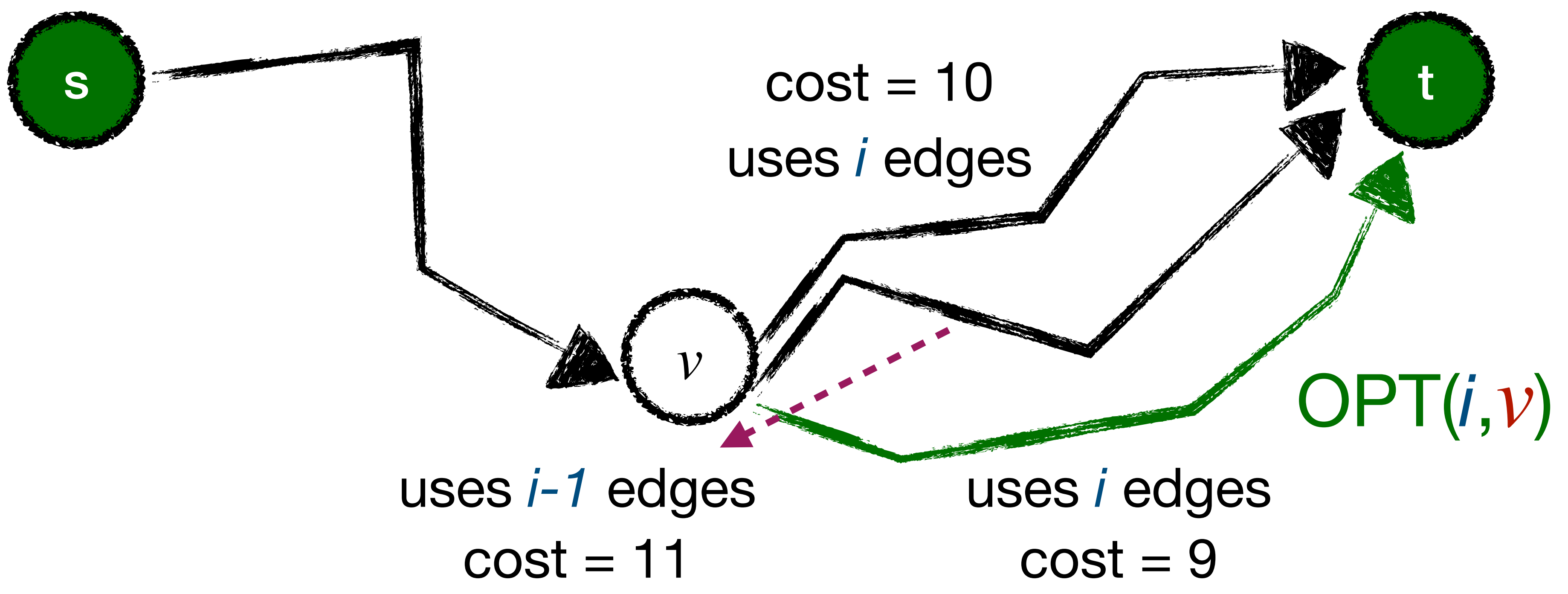
- Let OPT(i,v) denote the minimum cost of a path v~t from node v to t that uses at most i edges.

- We could also use OPT(i,v) to denote the minimum cost of a path s~v from s to node v that uses at most i edges.

  - This looks more like Dijkstra, but the former one is used in KT, because it fits better some of the other applications presented in the book.

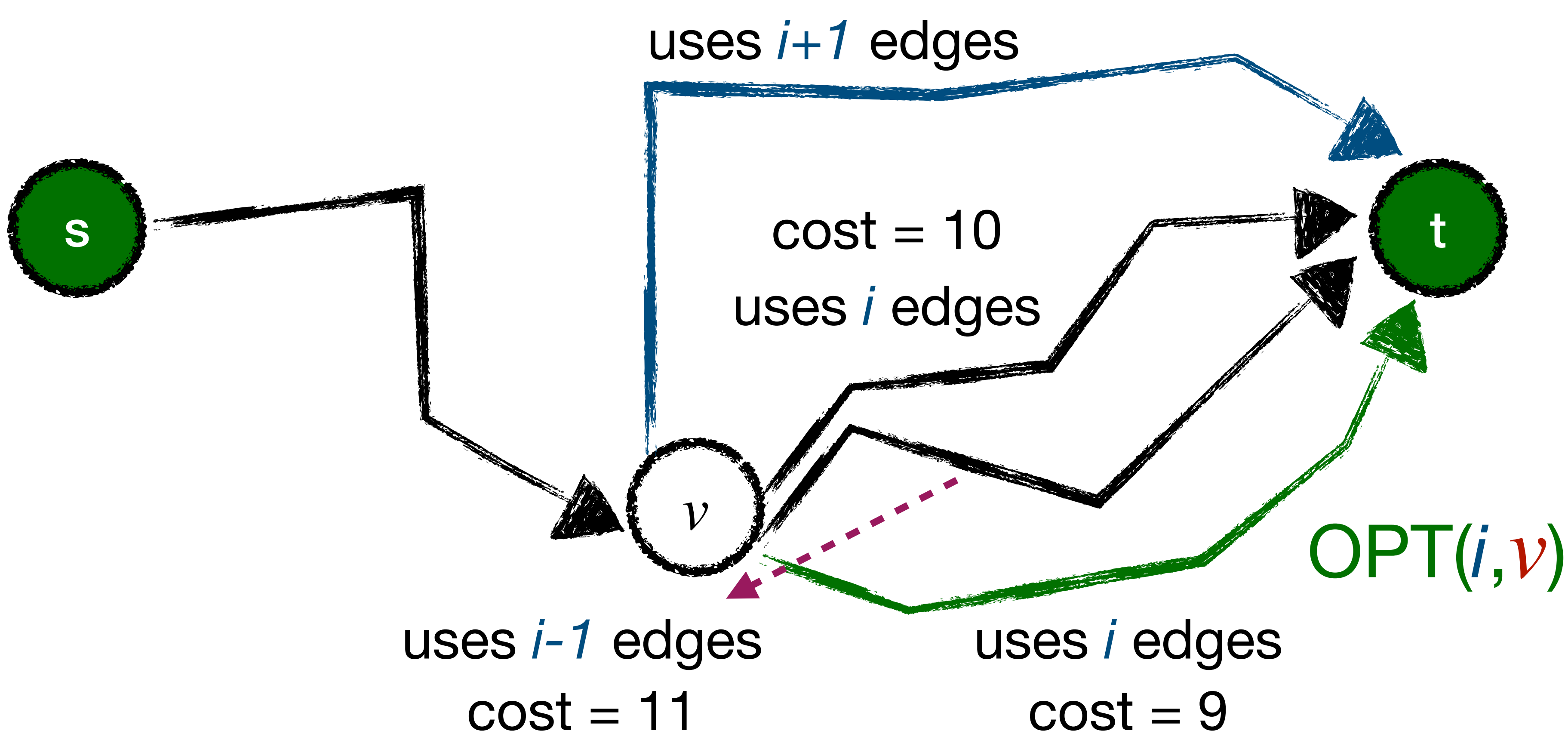
Setting up our subproblems

Let OPT(i,v) denote the minimum cost of a path v~t from node v to t that uses at most i edges.



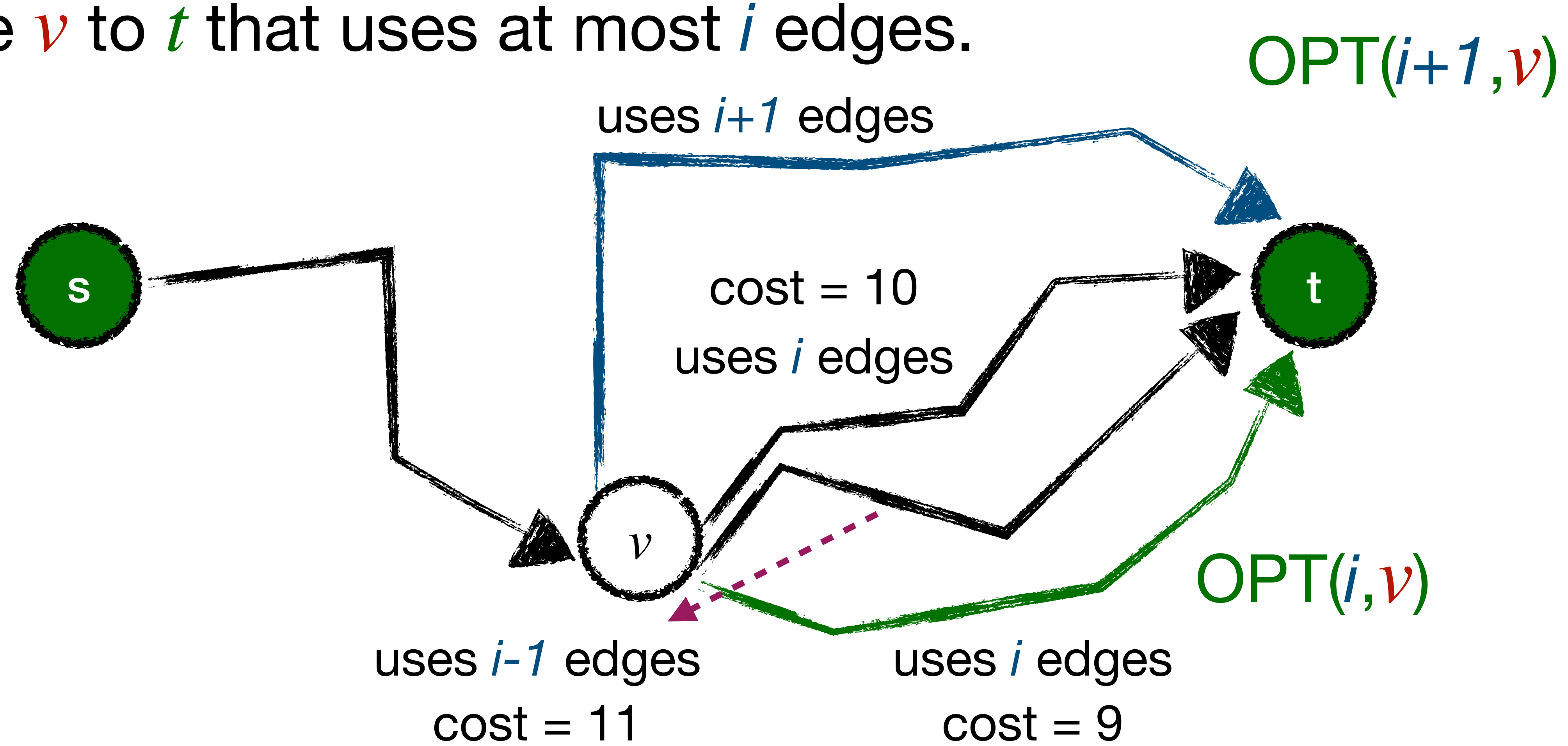
Setting up our subproblems

Let OPT(i,v) denote the minimum cost of a path v~t from node v to t that uses at most i edges.



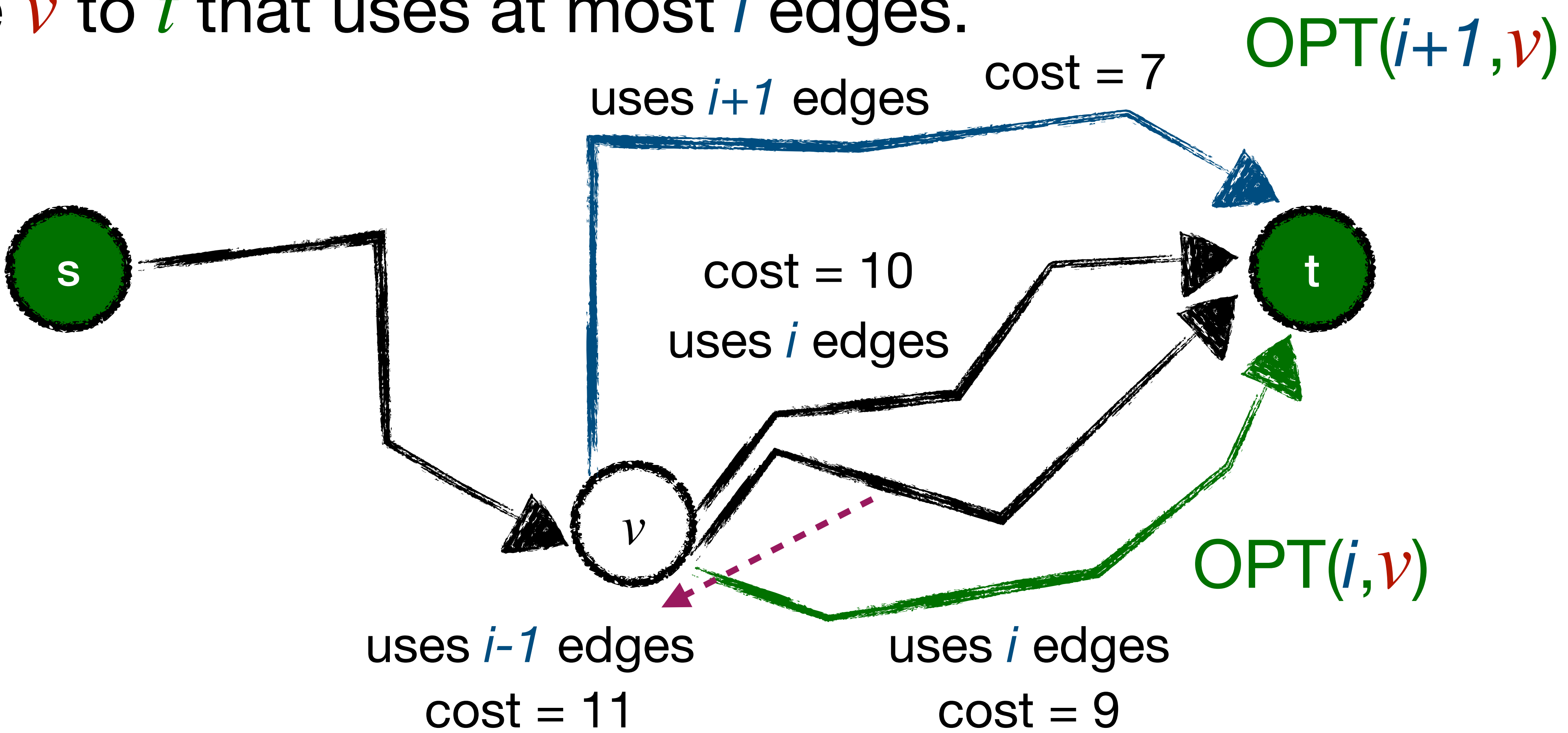
Setting up our subproblems

Let OPT(i,v) denote the minimum cost of a path v~t from node v to t that uses at most i edges.



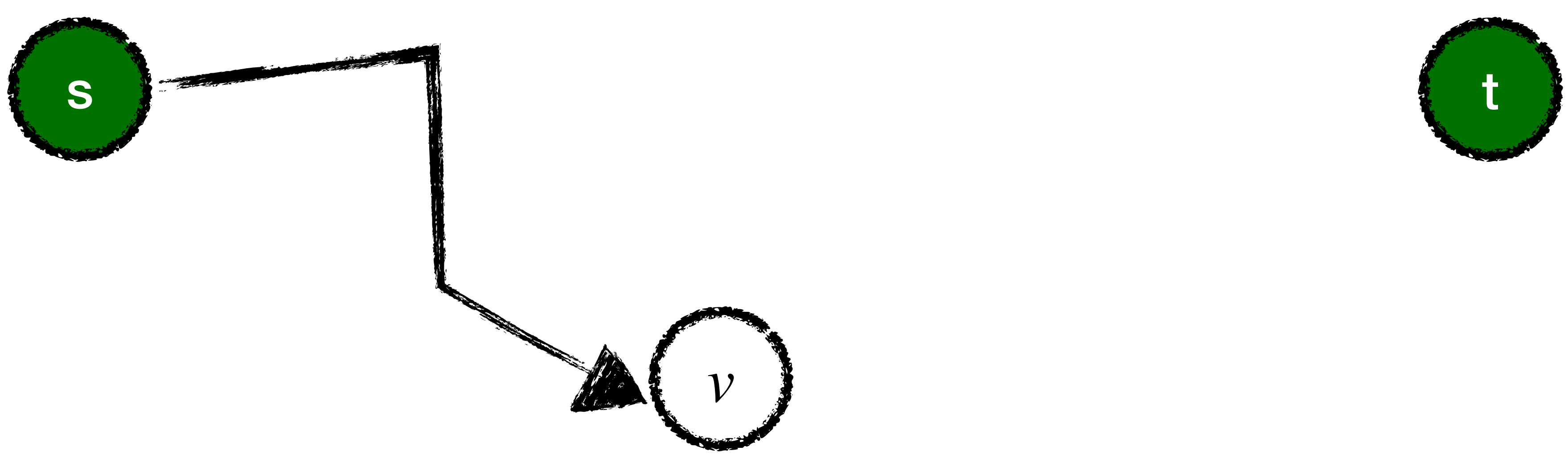
Setting up our subproblems

Let OPT(i,v) denote the minimum cost of a path v~t from node v to t that uses at most i edges.



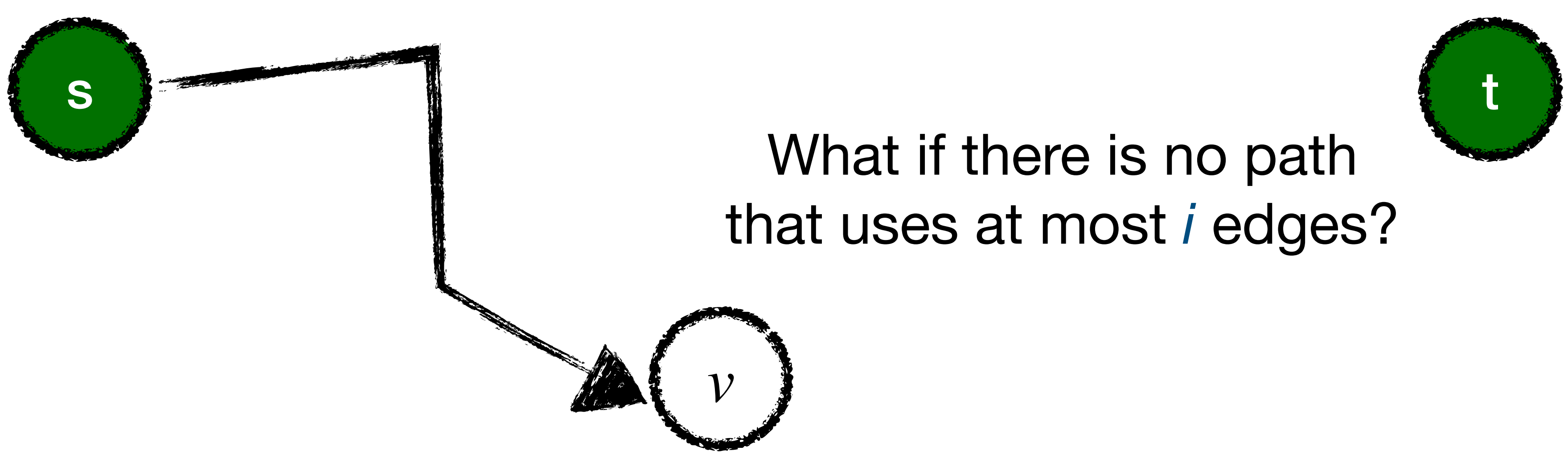
Setting up our subproblems

Let OPT(i,v) denote the minimum cost of a path v~t from node v to t that uses at most i edges.



Setting up our subproblems

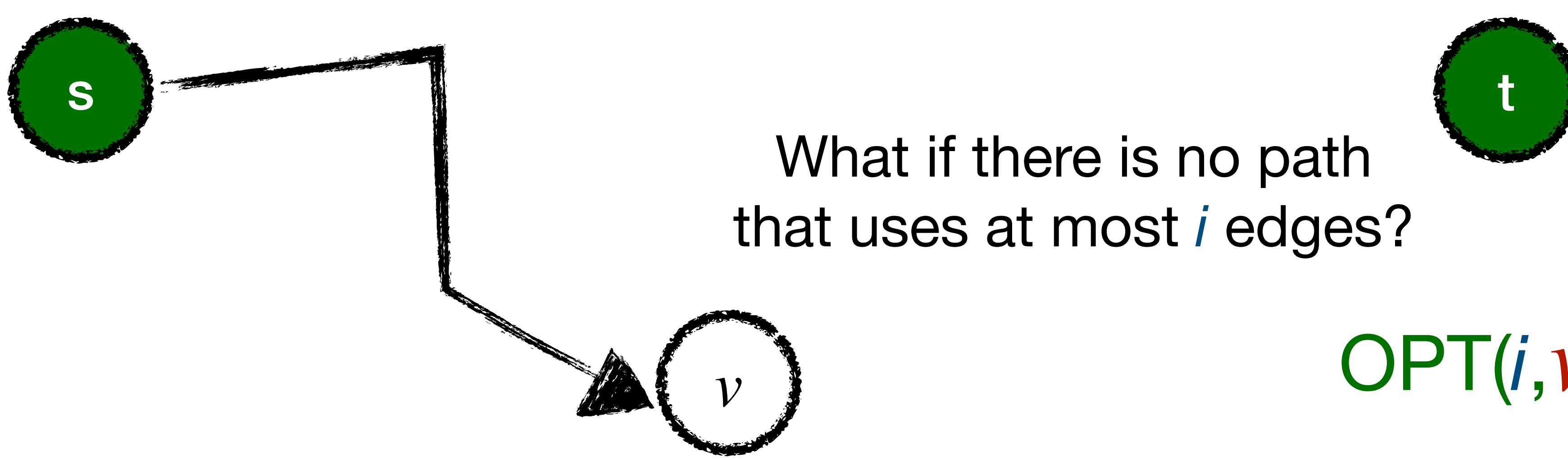
Let OPT(i,v) denote the minimum cost of a path v~t from node v to t that uses at most i edges.



What if there is no path that uses at most i edges?

Setting up our subproblems

Let OPT(i,v) denote the minimum cost of a path v~t from node v to t that uses at most i edges.



What if there is no path that uses at most i edges?

$\text{OPT}(i,v) = \infty$

Setting up our subproblems

- Let OPT(i,v) denote the minimum cost of a path v~t from node v to t that uses at most i edges.

Setting up our subproblems

- Let OPT(i,v) denote the minimum cost of a path v~t from node v to t that uses at most i edges.

- What is then the (global) solution to our problem?

Setting up our subproblems

- Let OPT(i,v) denote the minimum cost of a path $v \sim t$ from node v to t that uses at most i edges.
- What is then the (global) solution to our problem?
  - OPT(n-1,s)

Simple Observation

- **Observation:** If a graph *does not have any negative cycles*, then there is a shortest path $s \sim t$ from s to t that is simple, i.e., it does not repeat any nodes.

- **Corollary:** The length of any shortest path $s \sim t$ from s to t has at most $n - 1$ edges.

The recurrence relation

The recurrence relation

Let P a minimum-cost path using at most i edges from v to t with cost OPT(i,v).

The recurrence relation

Let P a minimum-cost path using at most i edges from v to t with cost OPT(i,v).

Case 1: P uses at most i-1 edges. Then OPT(i,v) = OPT(i-1,v).

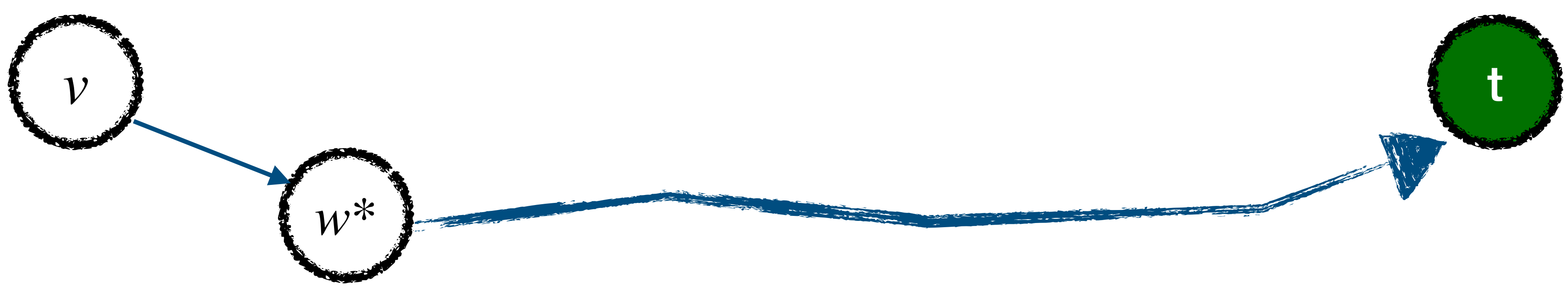
The recurrence relation

Let P a minimum-cost path using at most *i* edges from v to t with cost OPT(*i*,v).

Case 1: P uses at most *i-1* edges. Then OPT(*i*,v) = OPT(*i-1*,v).

Case 2: P uses exactly *i* edges. Let (v, w^*) be the first edge of P. Then OPT(*i*,v) = c_{vw^*} + OPT(*i-1*,w^*).

The recurrence relation

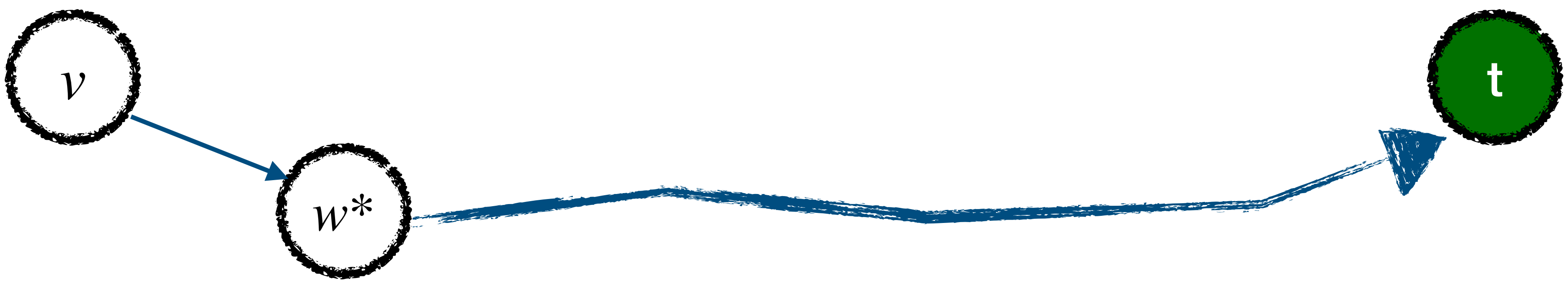


Let P a minimum-cost path using at most *i* edges from v to t with cost OPT(*i*,v).

Case 1: P uses at most *i-1* edges. Then OPT(*i*,v) = OPT(*i-1*,v).

Case 2: P uses exactly *i* edges. Let (v, w^*) be the first edge of P. Then OPT(*i*,v) = c_{vw^*} + OPT(*i-1*,w^*).

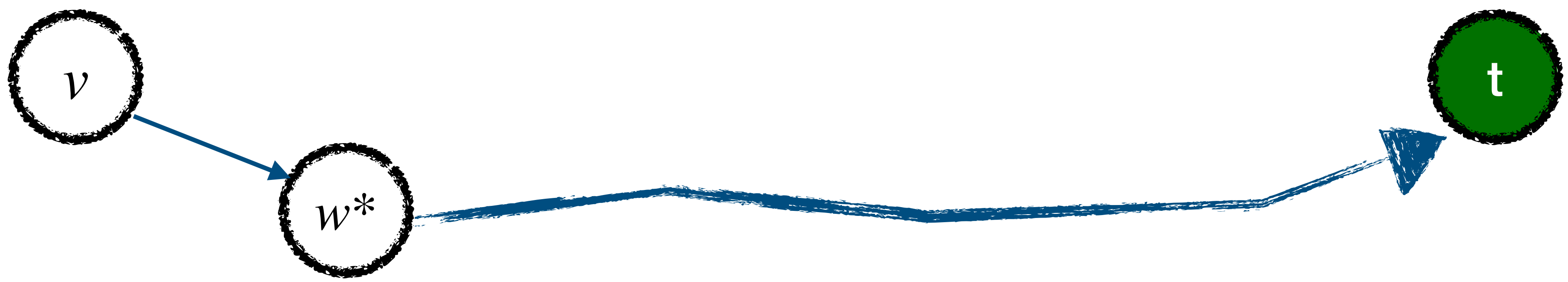
The recurrence relation



Let P a minimum-cost path using at most i edges from v to t with cost OPT(i,v).

Case 2: P uses exactly i edges. Let (v, w^*) be the first edge of P. Then OPT(i,v) = OPT(i-1,w^*).

The recurrence relation

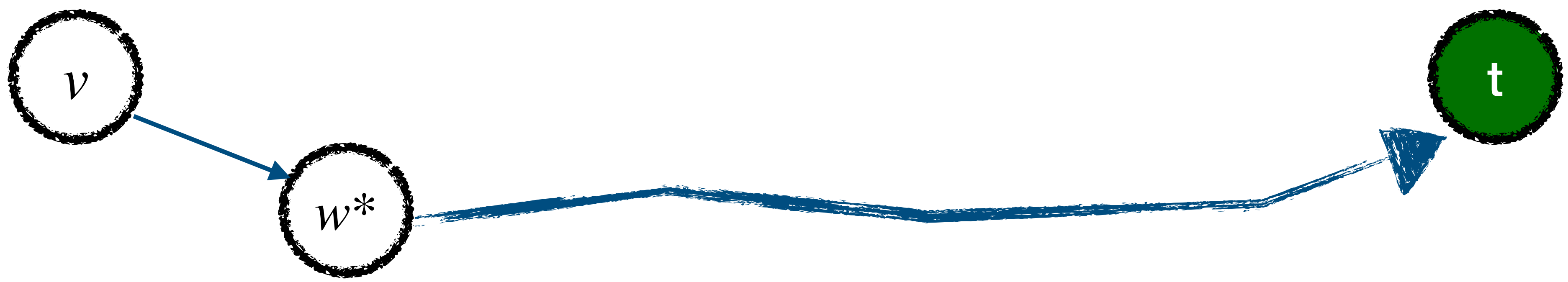


Let P a minimum-cost path using at most i edges from v to t with cost OPT(i,v).

Case 2: P uses exactly i edges. Let (v, w^*) be the first edge of P. Then OPT(i,v) = OPT(i-1,w^*).

We don't know w^*.

The recurrence relation



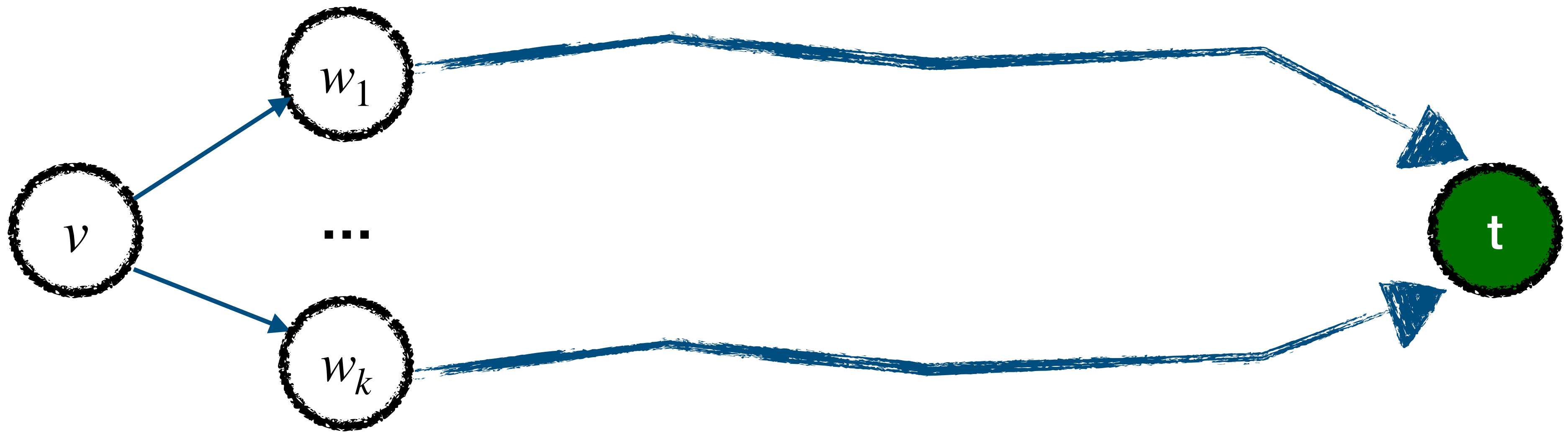
Let P a minimum-cost path using at most *i* edges from v to t with cost OPT(i,v).

Case 2: P uses exactly *i* edges. Let (v, w^*) be the first edge of P. Then OPT(i,v) = OPT(i-1,w^*).

We don't know w^*.

Take $\min\limits_{w \in N(v)} (c_{vw}$ + OPT(i-1,w))

The recurrence relation



Let P a minimum-cost path using at most i edges from v to t with cost OPT(i,v).

Case 2: P uses exactly i edges. Let (v, w^*) be the first edge of P. Then OPT(i,v) = OPT(i-1,w^*).

We don't know w^*.

Take $\min_{w \in N(v)} (c_{vw}$ + OPT(i-1,w))

The recurrence relation

Let P a minimum-cost path using at most i edges from v to t with cost OPT(i,v).

Case 1: P uses at most i-1 edges. Then OPT(i,v) = OPT(i-1,v).

Case 2: P uses exactly i edges.
Then $\mathrm{OPT}(i,v) = \min_{w \in N(v)} (c_{vw} + \mathrm{OPT}(i-1,w))$

The recurrence relation

Let P a minimum-cost path using at most i edges from v to t with cost OPT(i,v).

Case 1: P uses at most i-1 edges. Then OPT(i,v) = OPT(i-1,v).

Case 2: P uses exactly i edges.

Then OPT(i,v) = $\min_{w \in N(v)} (c_{vw} + \text{OPT}(i\text{-}1,w))$

Recurrence: OPT(i,v) = $\min\{\text{OPT}(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + \text{OPT}(i\text{-}1,w))\}$

The recurrence relation

Let P a minimum-cost path using at most i edges from v to t with cost OPT(i,v).

Case 1: P uses at most i-1 edges. Then OPT(i,v) = OPT(i-1,v).

Case 2: P uses exactly i edges.
Then OPT(i,v) = $\min_{w \in N(v)} (c_{vw} + \mathrm{OPT}(i-1, w))$

Recurrence: $\mathrm{OPT}(i,v) = \min\{\mathrm{OPT}(i-1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + \mathrm{OPT}(i-1,w))\}$

The Bellman-Ford Algorithm

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

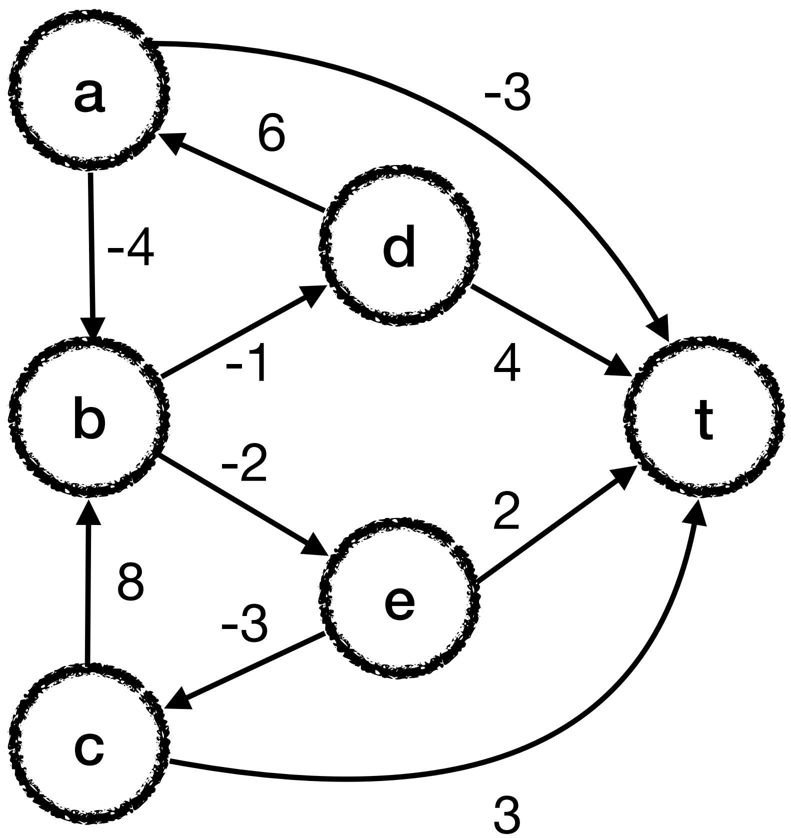
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t						
a						
b						
c						
d						
e						

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

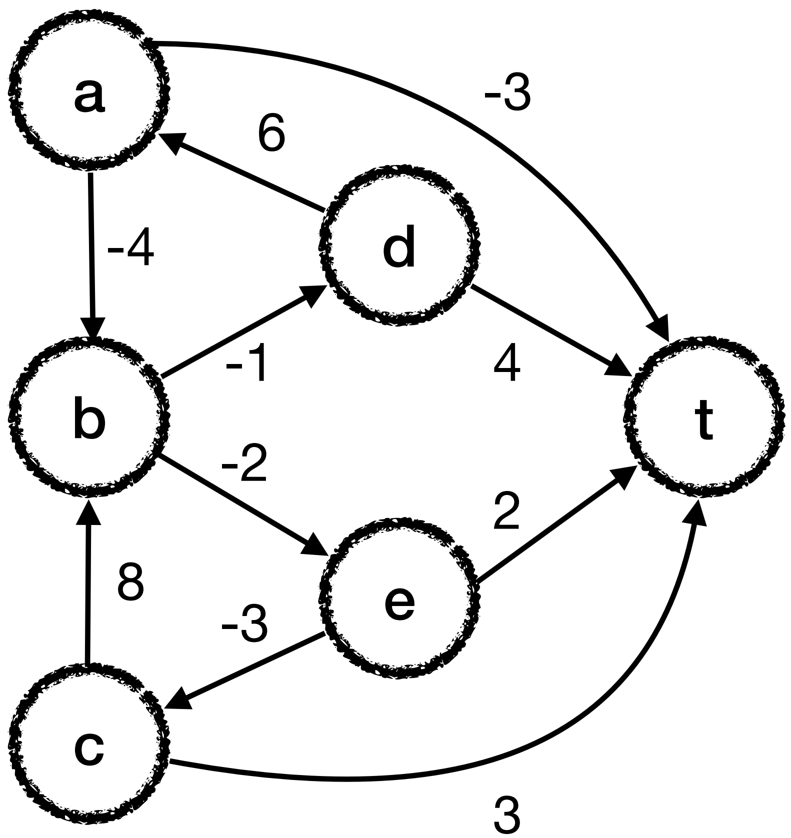
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t						
a						
b						
c						
d						
e						

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1,\ s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

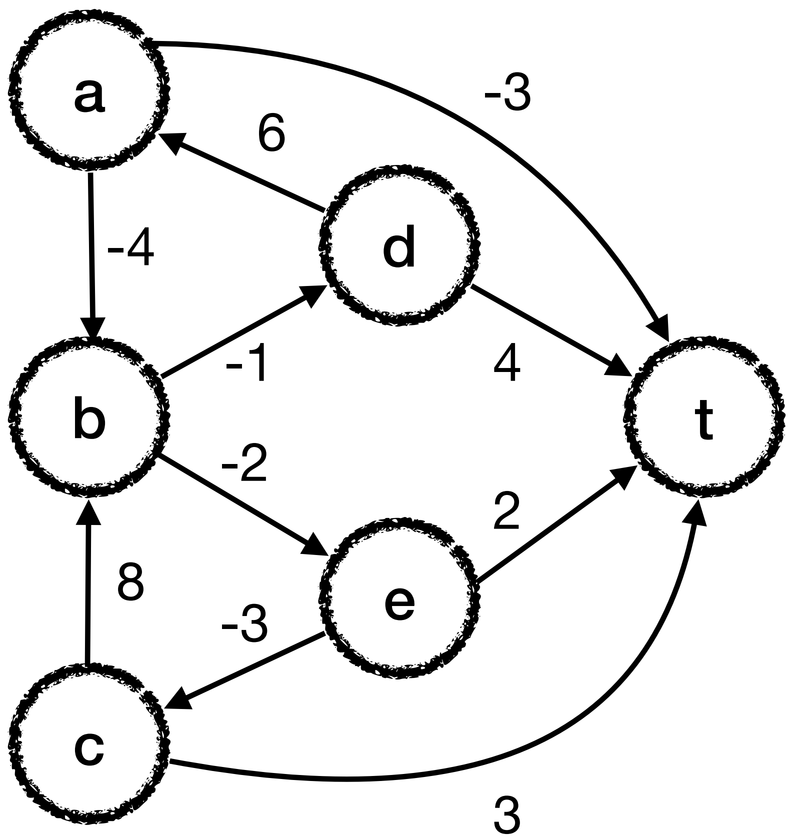
For $i = 1,2,\ldots,n-1$

For $v \in V$

$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0					
a	∞					
b	∞					
c	∞					
d	∞					
e	∞					

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

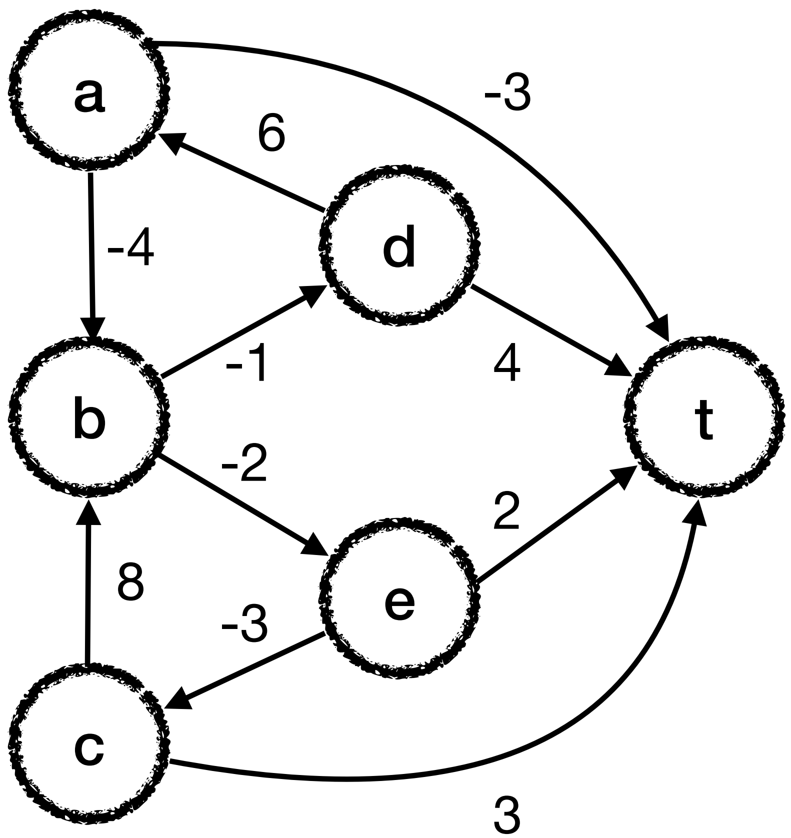
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0					
a	∞					
b	∞					
c	∞					
d	∞					
e	∞					

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

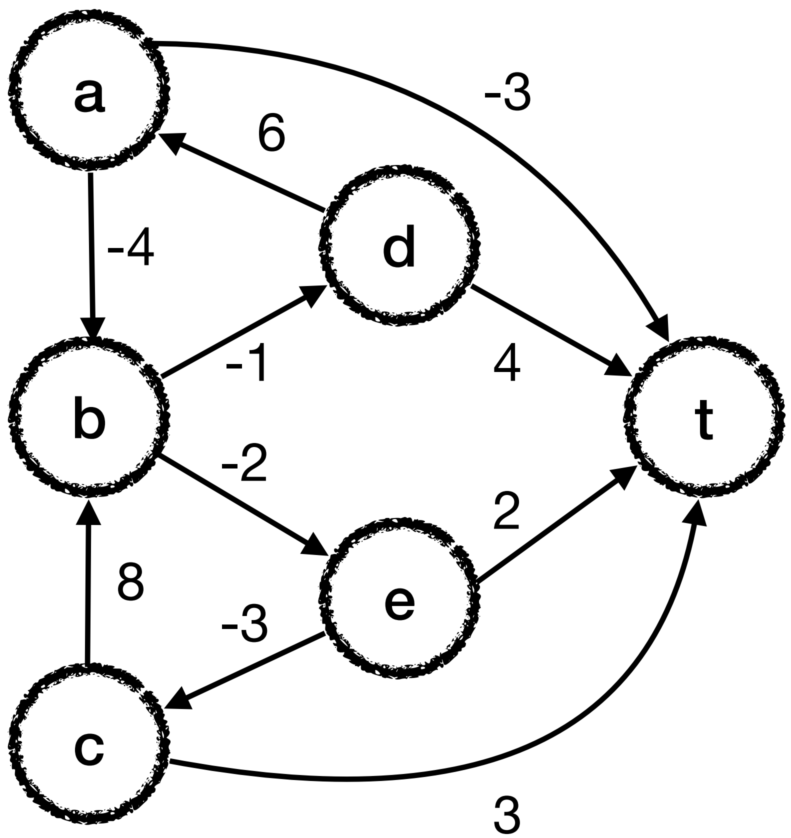
For $i = 1,2,\ldots,n-1$

For $v \in V$

$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0					
a	∞					
b	∞					
c	∞					
d	∞					
e	∞					

$$M(1,t) = \min\{M(0,t)\ ,\ \min_{w \in N(t)} (0 + M(0,w))\}$$

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

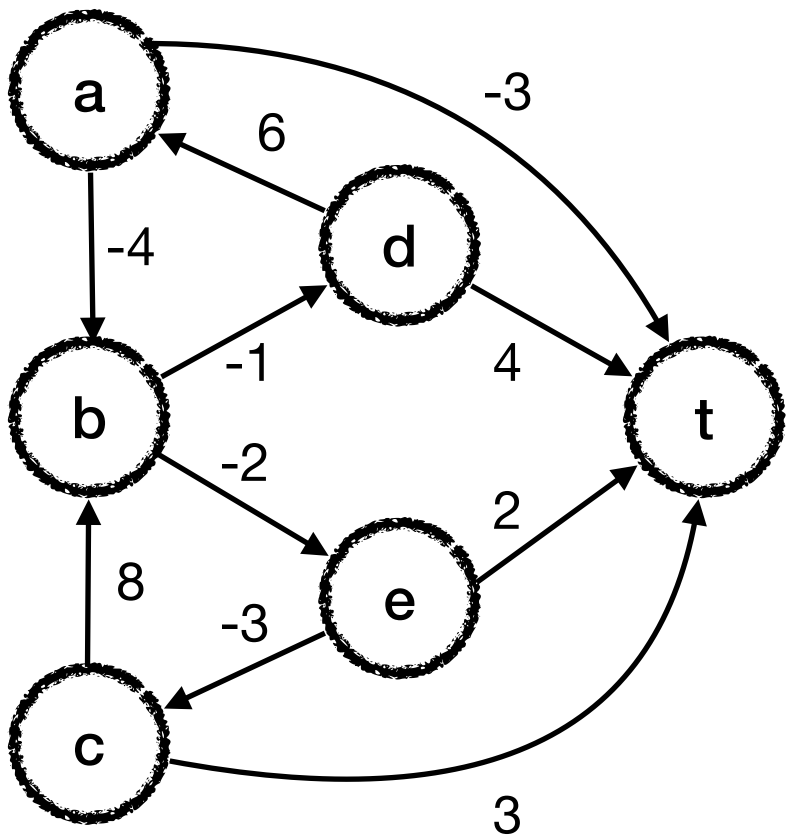
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0				
a	∞					
b	∞					
c	∞					
d	∞					
e	∞					

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

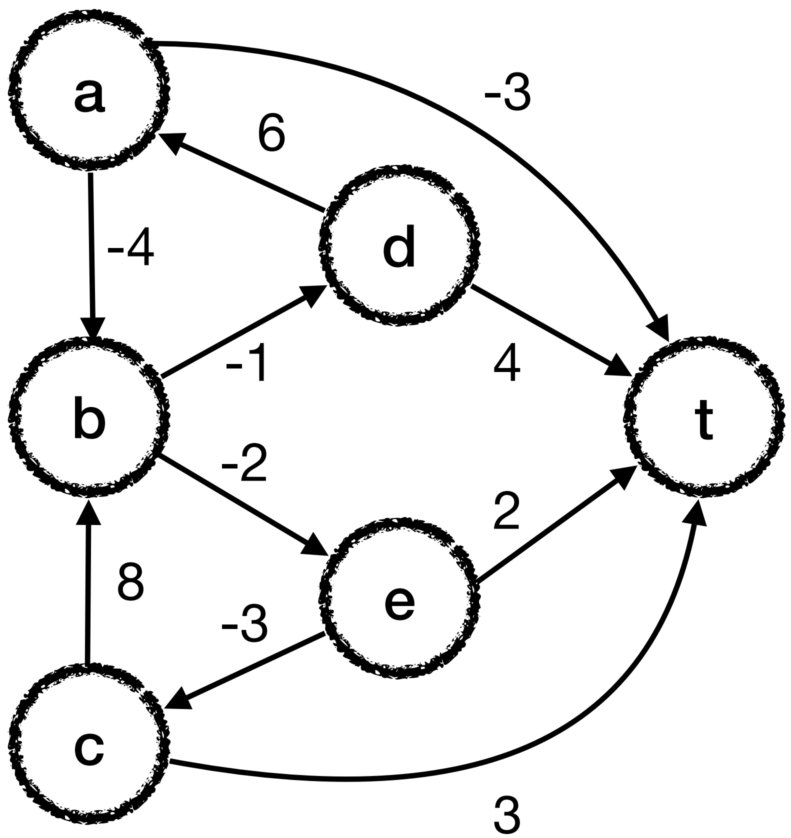
For $i = 1,2,\ldots,n-1$

For $v \in V$

$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0				
a	∞					
b	∞					
c	∞					
d	∞					
e	∞					

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

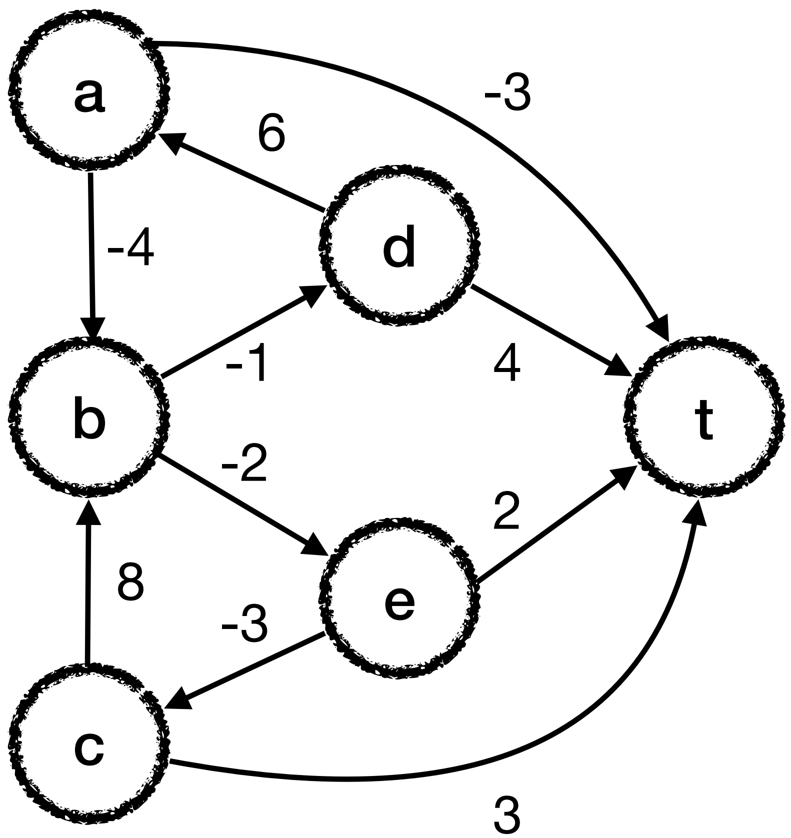
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min\limits_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0				
a	∞					
b	∞					
c	∞					
d	∞					
e	∞					

$$M(1,a) = \min\{M(0,a)\ ,\ \min_{w \in N(a)} (c_{aw} + M(0,w))\}$$

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

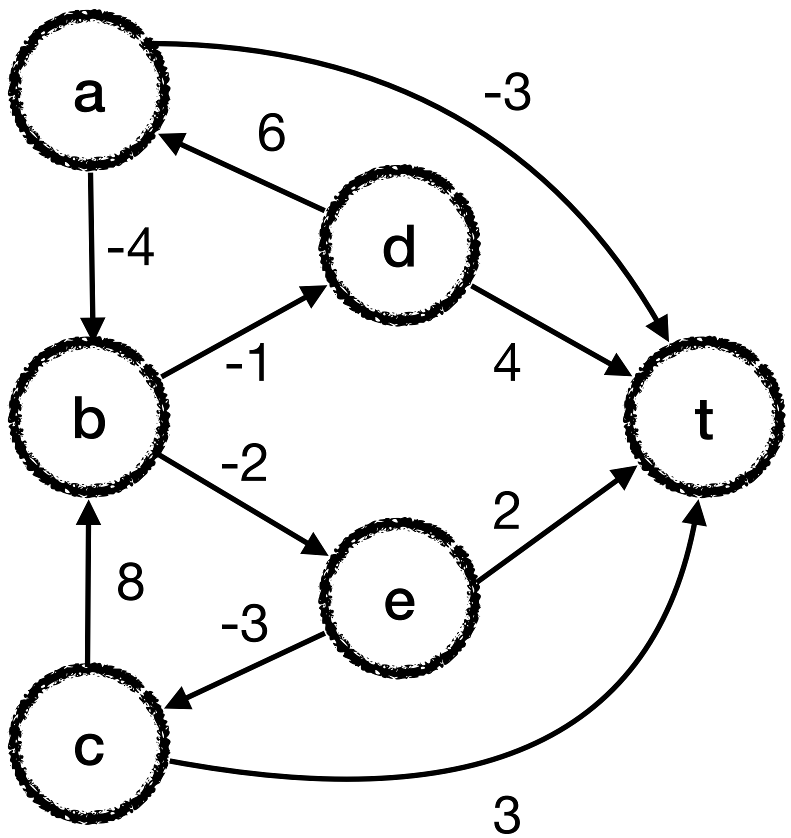
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0				
a	∞					
b	∞					
c	∞					
d	∞					
e	∞					

$$M(1,a) = \min\{M(0,a)\ ,\ \min_{w\in N(a)} (c_{aw} + M(0,w))\}$$

$$M(1,a) = c_{at} + M(0,t)$$

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

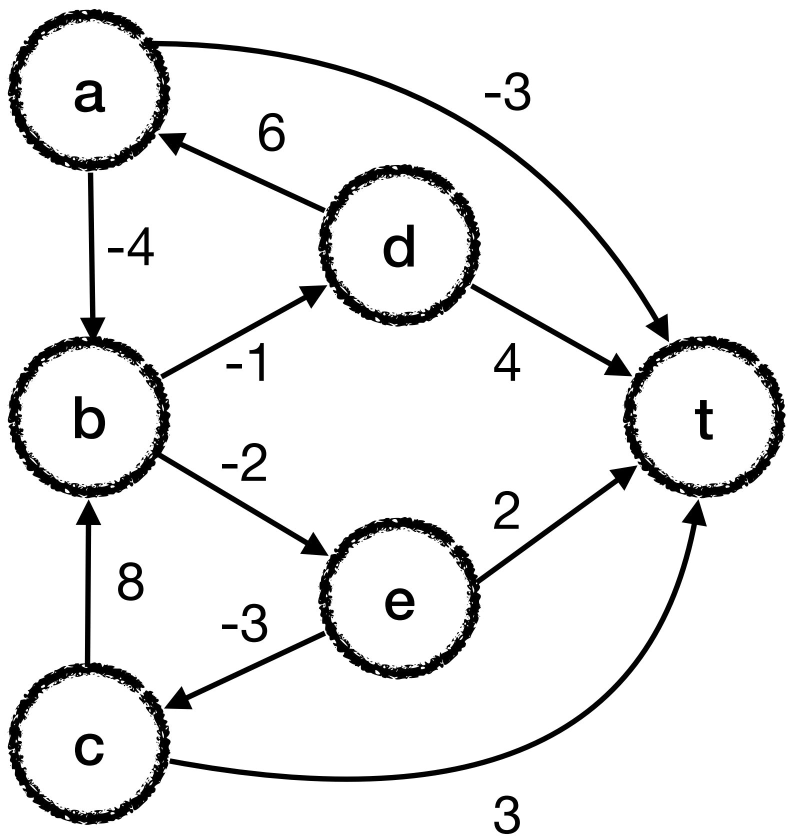
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w\in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0				
a	∞	-3				
b	∞					
c	∞					
d	∞					
e	∞					

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

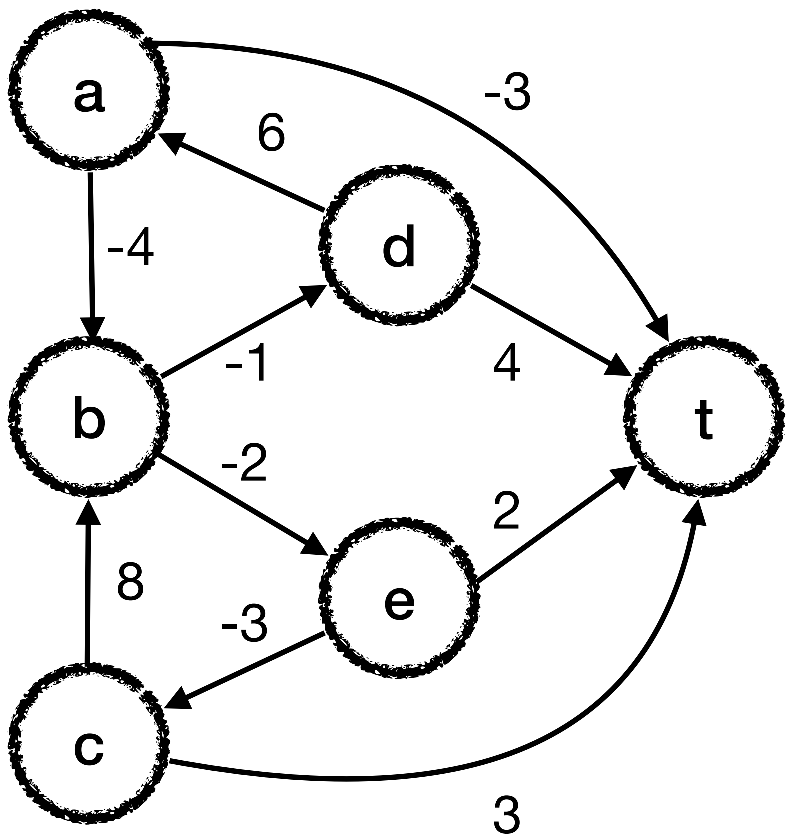
For $i = 1,2,\ldots,n-1$

For $v \in V$

$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0				
a	∞	-3				
b	∞					
c	∞					
d	∞					
e	∞					

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

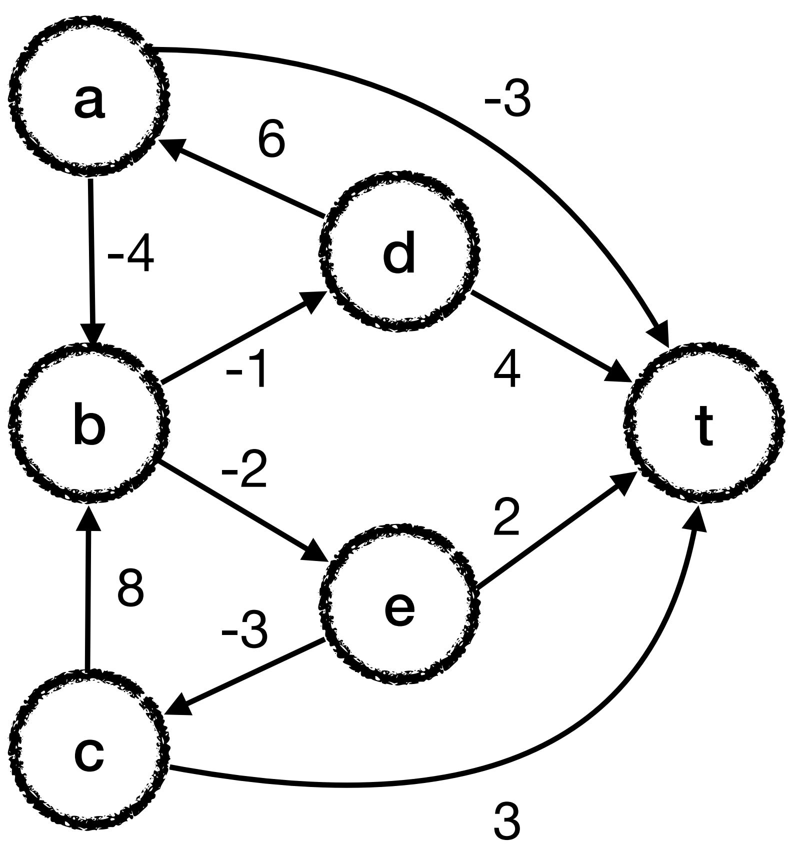
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0				
a	∞	-3				
b	∞					
c	∞					
d	∞					
e	∞					

$$M(1,b) = \min\{M(0,b)\ ,\ \min_{w \in N(b)} (c_{bw} + M(0,w))\}$$

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1,\ s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

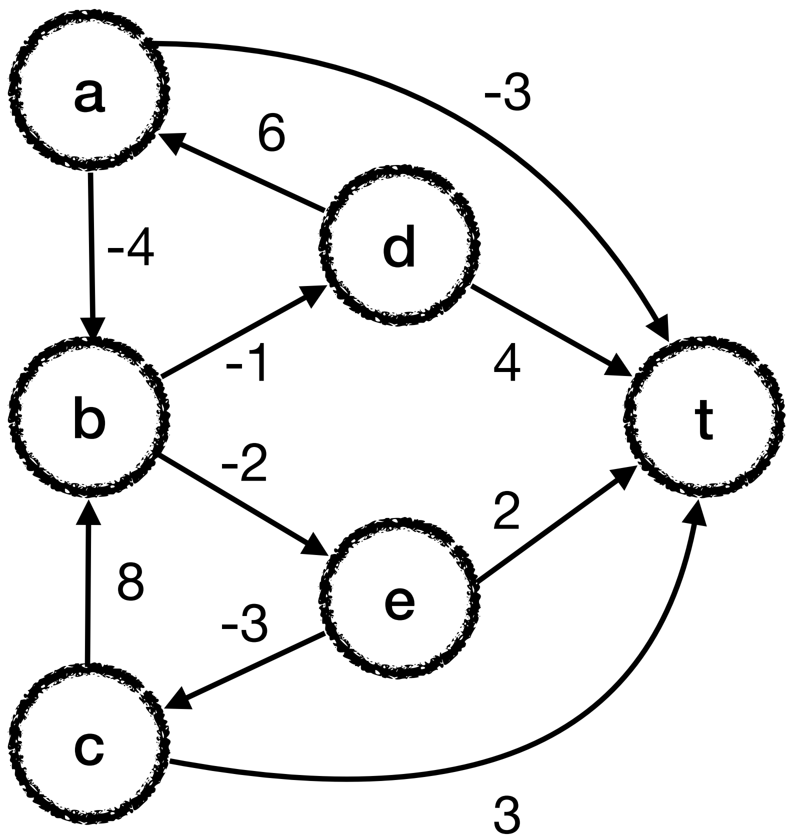
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0				
a	∞	-3				
b	∞					
c	∞					
d	∞					
e	∞					

$$M(1,\mathrm{b}) = \min\{M(0,\mathrm{b}),\ \min_{w \in N(b)} (c_{bw} + M(0,w))\}$$

$$M(1,\mathrm{b}) = \infty$$

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

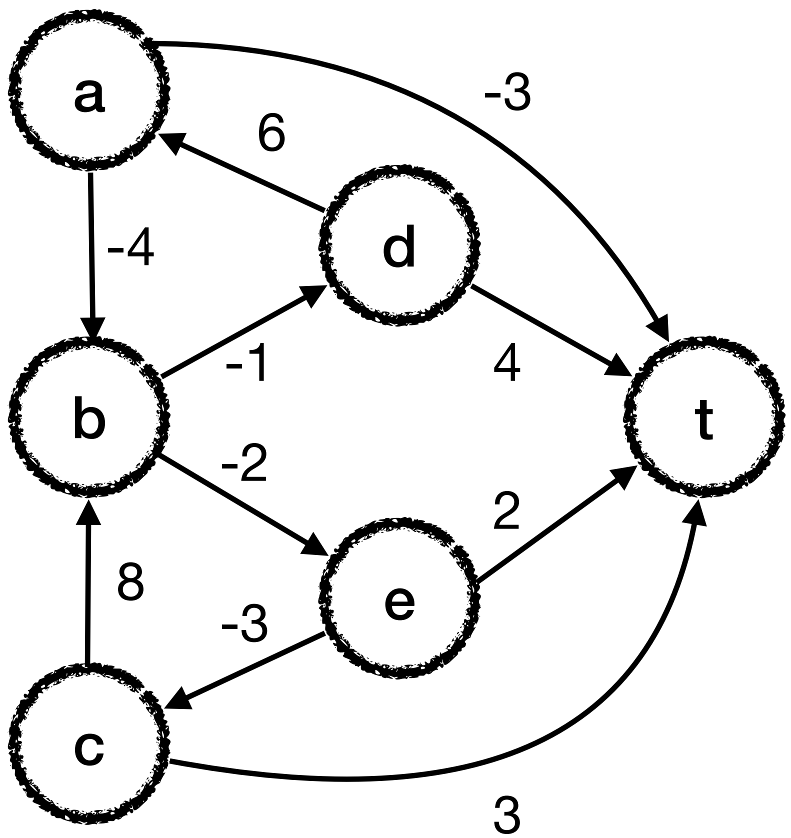
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v),\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0				
a	∞	-3				
b	∞	∞				
c	∞					
d	∞					
e	∞					

$$M(1,\mathrm{b}) = \min\{M(0,\mathrm{b}),\ \min_{w \in N(b)} (c_{bw} + M(0,w))\}$$

$$M(1,\mathrm{b}) = \infty$$

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1,\ s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

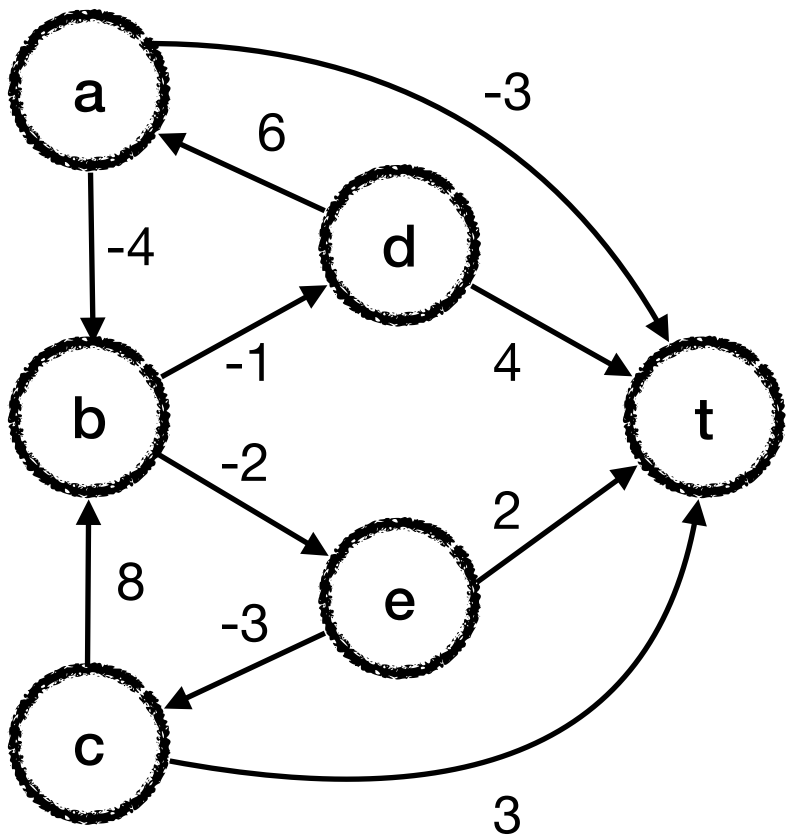
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v),\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0	0	0	0	
a	∞	-3	-3	-4		
b	∞	∞	0	-2		
c	∞	3	3	3		
d	∞	4	3	3		
e	∞	2	0	0		

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

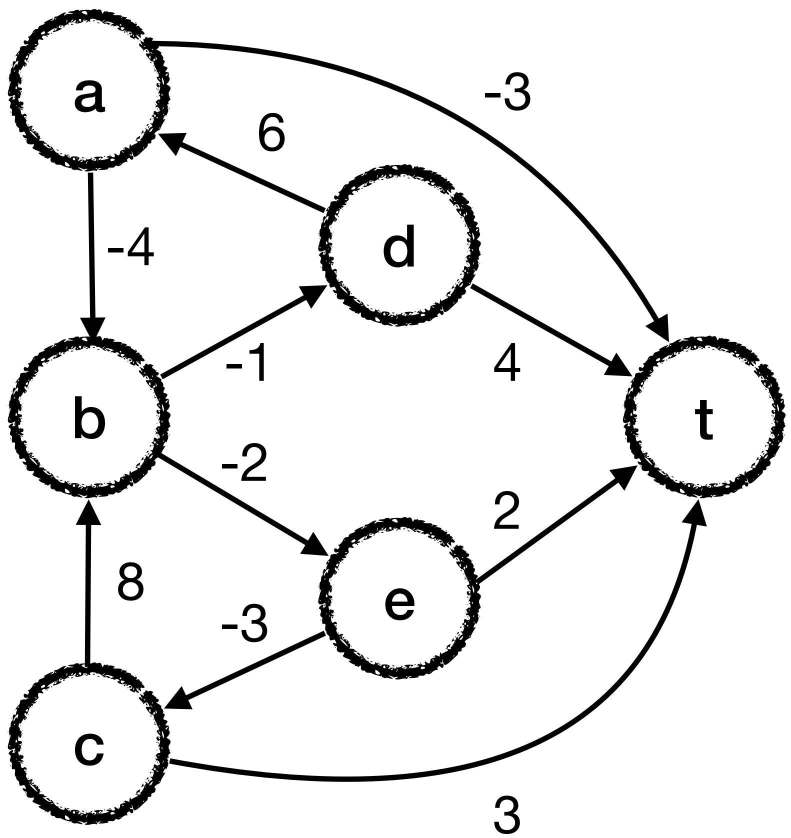
For $i = 1,2,\ldots,n-1$

For $v \in V$

$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0	0	0	0	
a	∞	-3	-3	-4		
b	∞	∞	0	-2		
c	∞	3	3	3		
d	∞	4	3	3		
e	∞	2	0	0		

$$M(4,\mathrm{a}) = \min\{M(3,\mathrm{a}) , \min_{w \in N(a)} (c_{aw} + M(3,\mathrm{w}))\}$$

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

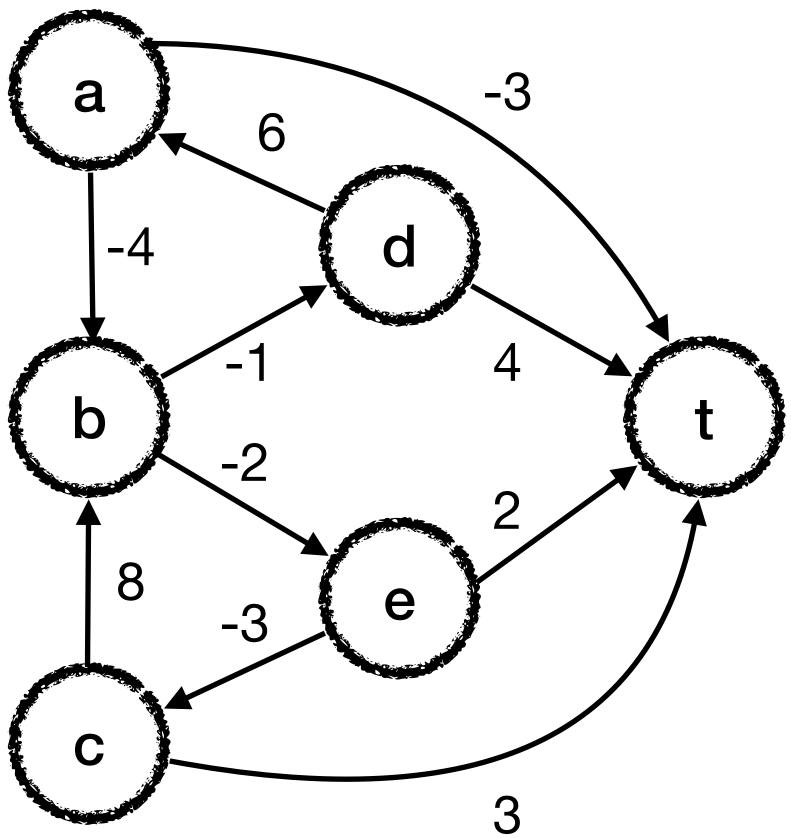
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v) , \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0	0	0	0	
a	∞	-3	-3	-4		
b	∞	∞	0	-2		
c	∞	3	3	3		
d	∞	4	3	3		
e	∞	2	0	0		

$$M(4,\mathrm{a}) = \min\{M(3,\mathrm{a})\,,\ \min_{w\in N(a)} (c_{aw} + M(3,\mathrm{w}))\}$$

$$c_{ab} + M(3,\mathrm{b}) = \text{-}6$$

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1,\ s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

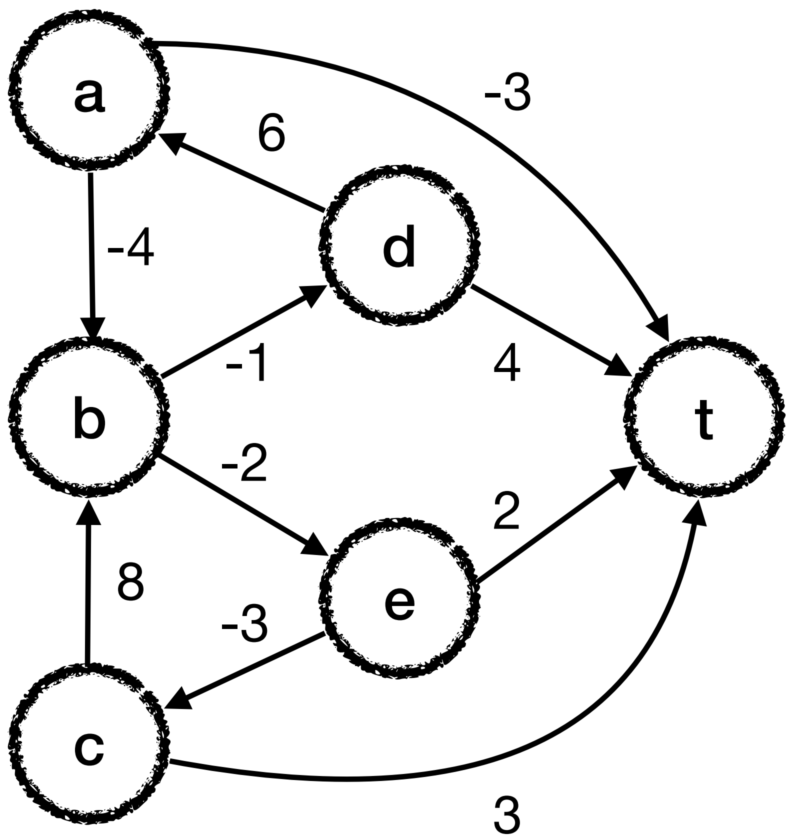
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\,,\ \min_{w\in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0	0	0	0	
a	∞	-3	-3	-4		
b	∞	∞	0	-2		
c	∞	3	3	3		
d	∞	4	3	3		
e	∞	2	0	0		

$$M(4,a) = \min\{M(3,a)\ ,\ \min_{w \in N(a)} (c_{aw} + M(3,w))\}$$

$$c_{ab} + M(3,b) = -6$$

$$c_{at} + M(3,t) = -3$$

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1,\ s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

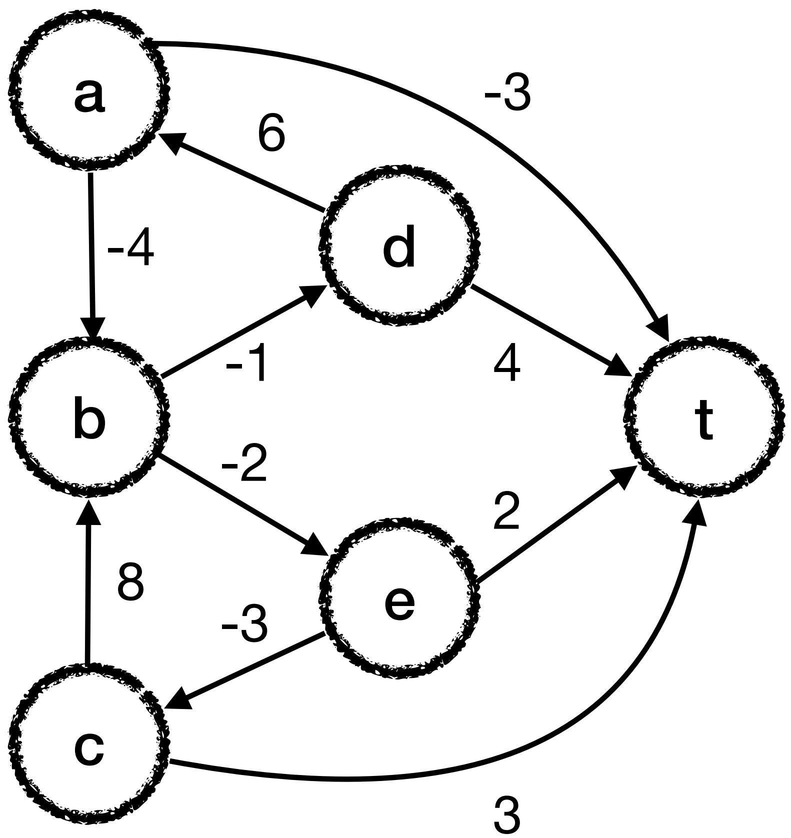
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0	0	0	0	
a	∞	-3	-3	-4	-6	
b	∞	∞	0	-2		
c	∞	3	3	3		
d	∞	4	3	3		
e	∞	2	0	0		

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

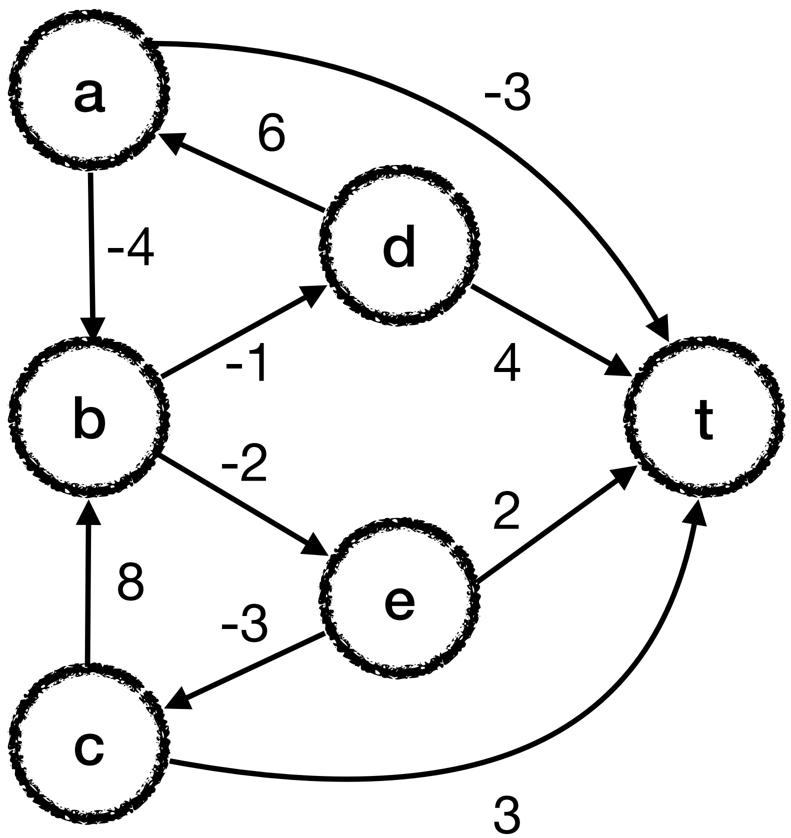
For $i = 1,2,\ldots,n-1$

For $v \in V$

$$M(i,v) = \min\{M(i\text{-}1,v)\,,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0	0	0	0	
a	∞	-3	-3	-4	-6	
b	∞	∞	0	-2		
c	∞	3	3	3		
d	∞	4	3	3		
e	∞	2	0	0		

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

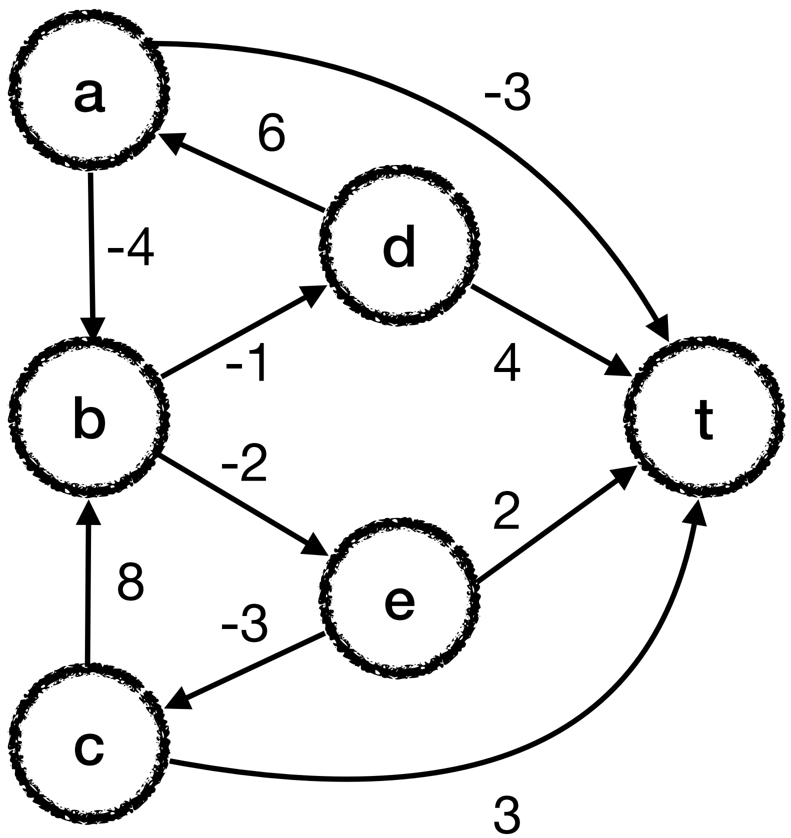
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0	0	0	0	
a	∞	-3	-3	-4	-6	
b	∞	∞	0	-2		
c	∞	3	3	3		
d	∞	4	3	3		
e	∞	2	0	0		

$$M(4,b) = \min\{M(3,b)\ ,\ \min_{w \in N(b)} (c_{bw} + M(3,w))\}$$

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

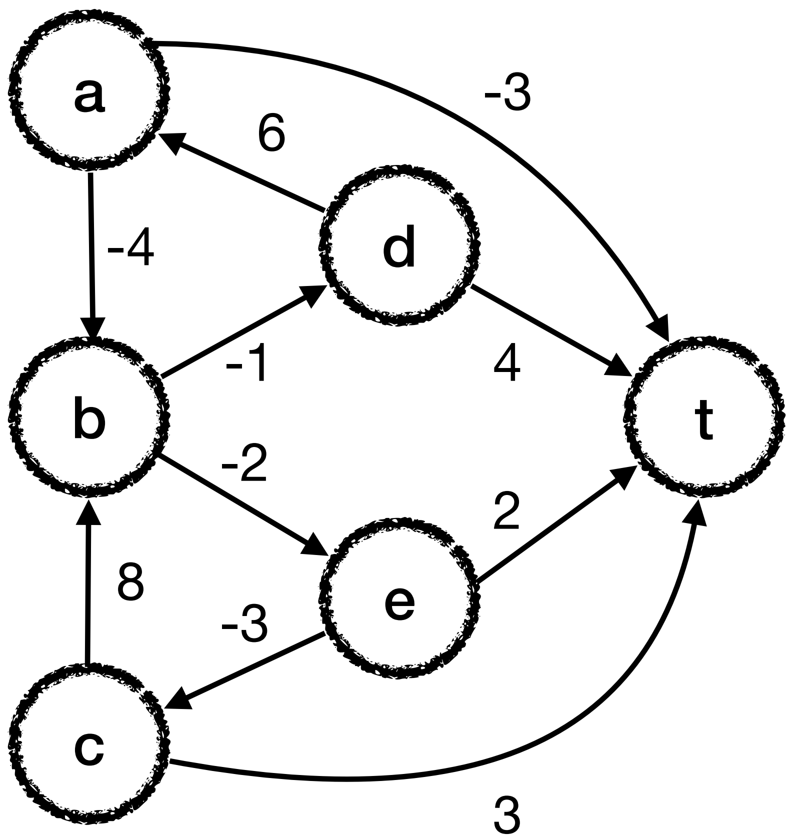
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0	0	0	0	
a	∞	-3	-3	-4	-6	
b	∞	∞	0	-2		
c	∞	3	3	3		
d	∞	4	3	3		
e	∞	2	0	0		

$$M(4,b) = \min\{M(3,b)\ ,\ \min_{w \in N(b)} (c_{bw} + M(3,w))\}$$

$$c_{bd} + M(3,d) = -1 + 3$$

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

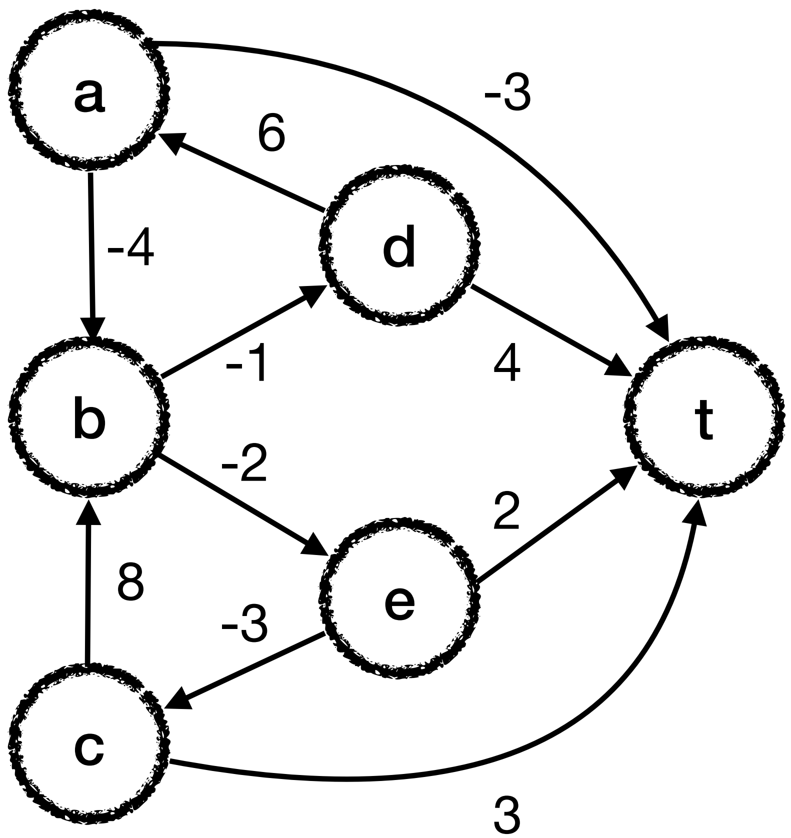
For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0	0	0	0	
a	∞	-3	-3	-4	-6	
b	∞	∞	0	-2		
c	∞	3	3	3		
d	∞	4	3	3		
e	∞	2	0	0		

$$M(4,b) = \min\{M(3,b)\ ,\ \min_{w \in N(b)} (c_{bw} + M(3,w))\}$$

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1,\ s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

For $i = 1,2,\ldots,n-1$

For $v \in V$

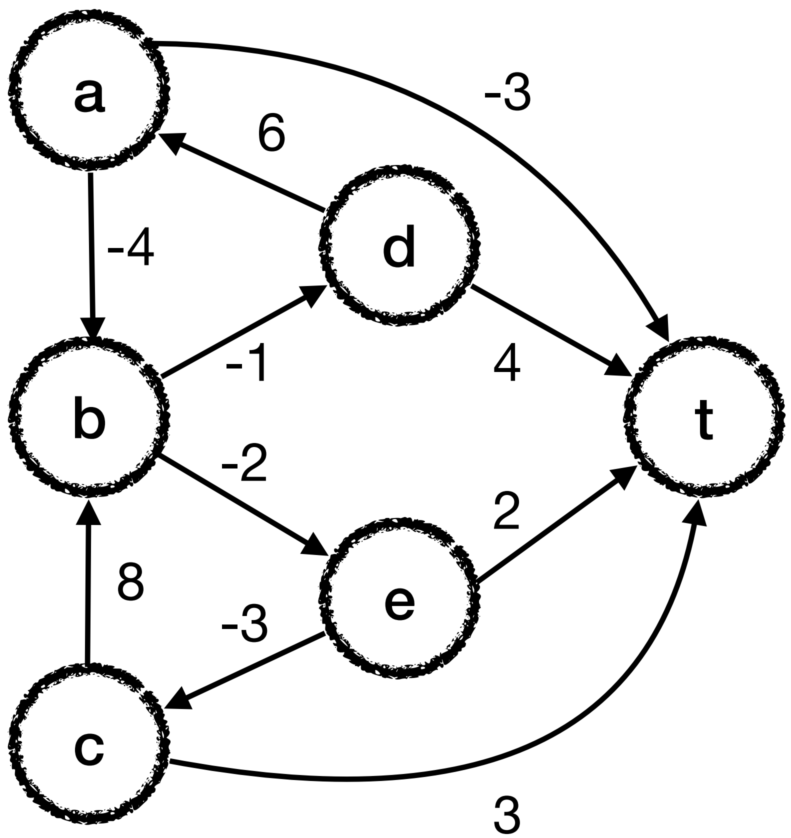
$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

$$c_{bd} + M(3,d) = -1 + 3$$

$$c_{be} + M(3,e) = -2 + 0$$

Example



	0	1	2	3	4	5
t	0	0	0	0	0	
a	∞	-3	-3	-4	-6	
b	∞	∞	0	-2	-2	
c	∞	3	3	3		
d	∞	4	3	3		
e	∞	2	0	0		

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

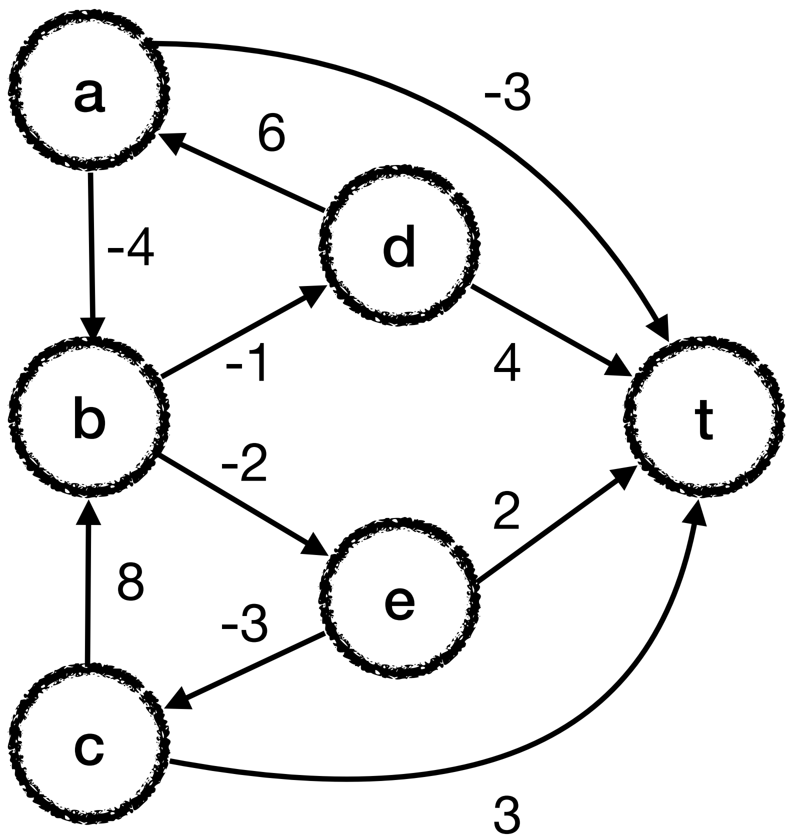
For $i = 1,2,\ldots,n-1$

For $v \in V$

$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	-3	-3	-4	-6	-6
b	∞	∞	0	-2	-2	-2
c	∞	3	3	3	3	3
d	∞	4	3	3	2	0
e	∞	2	0	0	0	0

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

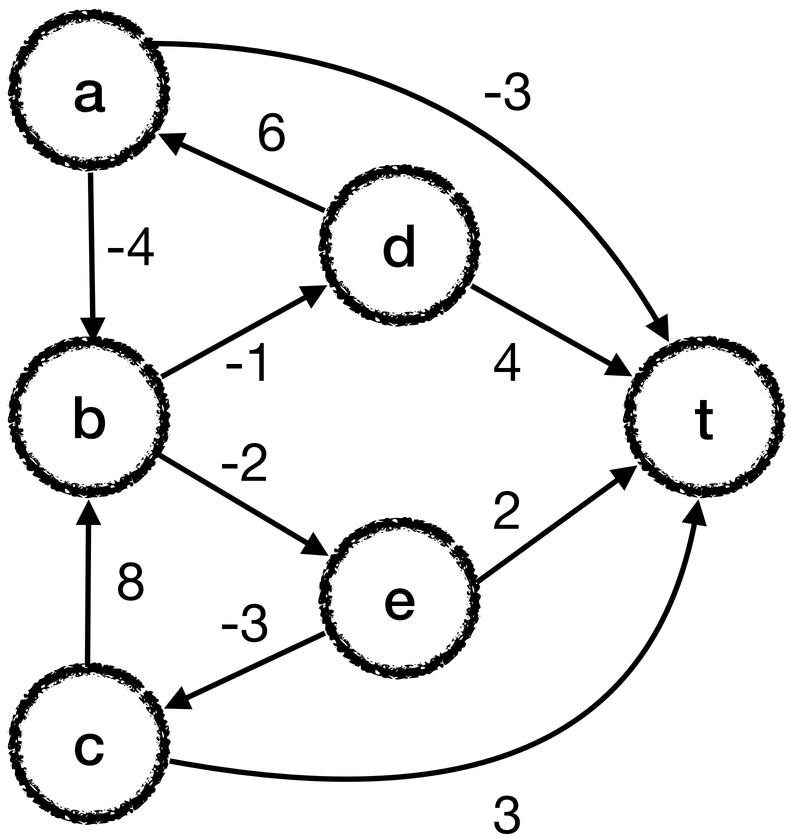
For $i = 1,2,\ldots,n-1$

For $v \in V$

$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	-3	-3	-4	-6	-6
b	∞	∞	0	-2	-2	-2
c	∞	3	3	3	3	3
d	∞	4	3	3	2	0
e	∞	2	0	0	0	0

We can find the actual paths via tracing backwards:

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

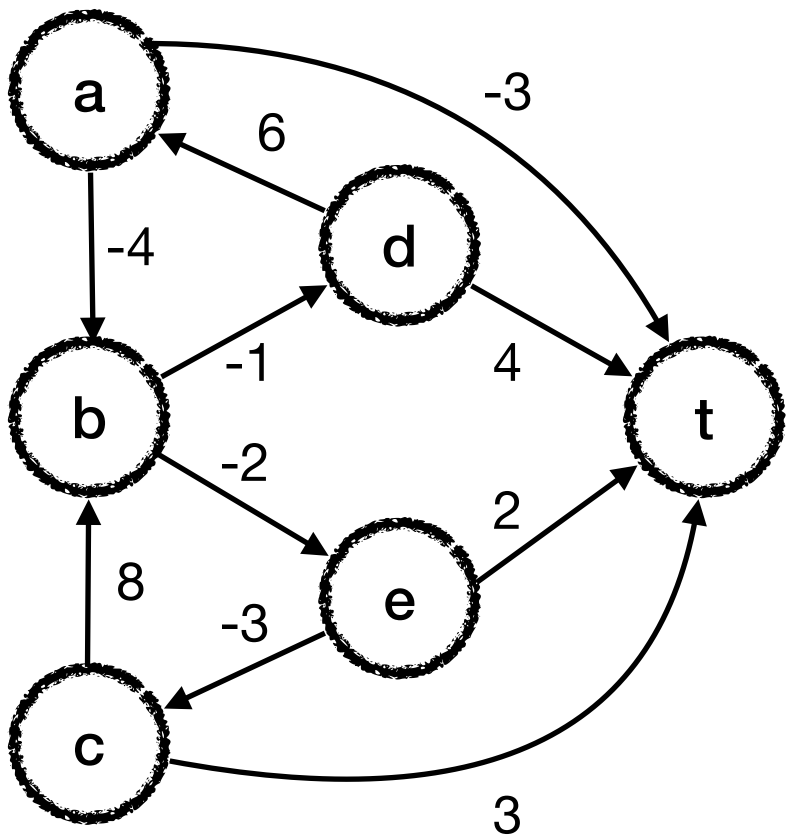
For $i = 1,2,\ldots,n-1$

For $v \in V$

$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$$

Return $M(n\text{-}1,s)$

Example



	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	-3	-3	-4	-6	-6
b	∞	∞	0	-2	-2	-2
c	∞	3	3	3	3	3
d	∞	4	3	3	2	0
e	∞	2	0	0	0	0

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

For $i = 1,2,\ldots,n-1$

For $v \in V$

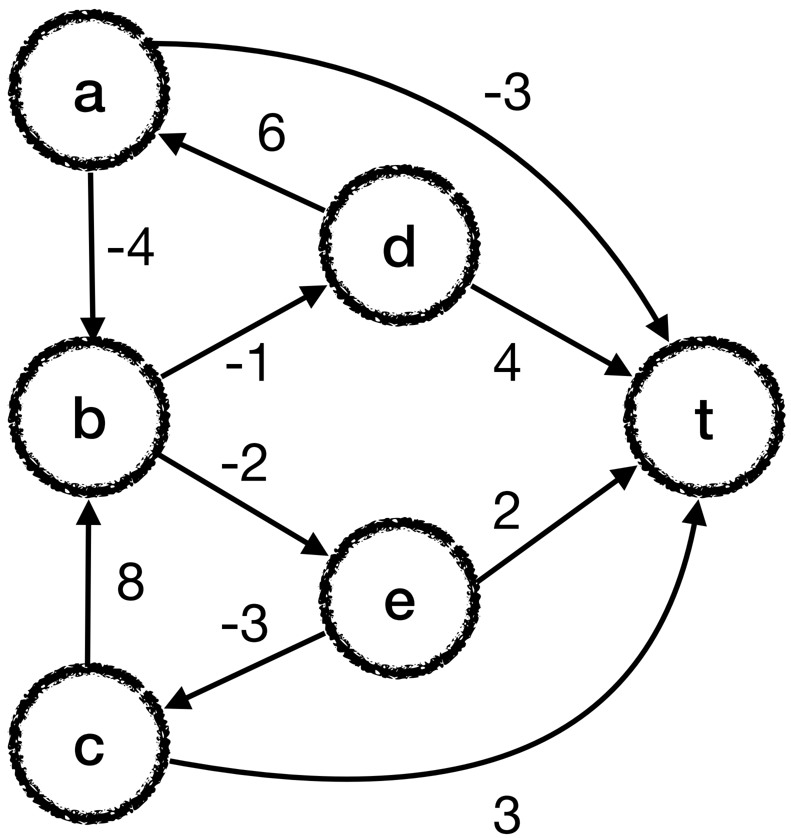
$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

We can find the actual paths via tracing backwards:

$$M(5,d) = \min\{M(4,d)\ ,\ \min_{w \in N(d)} (c_{dw} + M(4,w))\}$$

Example



	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	-3	-3	-4	-6	-6
b	∞	∞	0	-2	-2	-2
c	∞	3	3	3	3	3
d	∞	4	3	3	2	0
e	∞	2	0	0	0	0

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

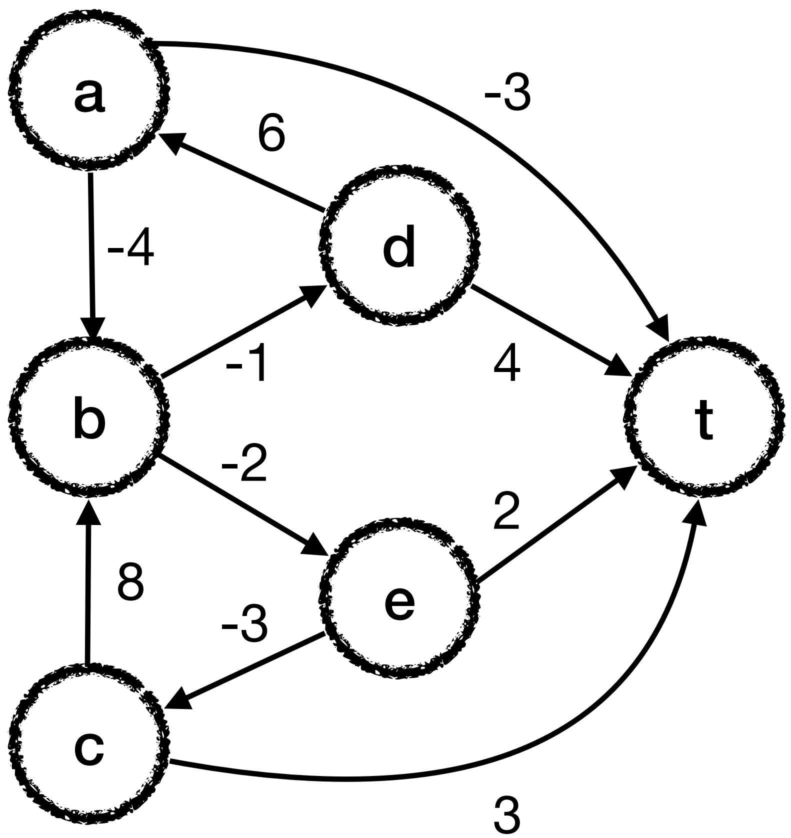
We can find the actual paths via tracing backwards:

$$M(5,\mathrm{d}) = \min\{M(4,\mathrm{d})\ ,$$

$$\min_{w \in N(d)} (c_{dw} + M(4,\mathrm{w}))\}$$

So the first edge must be (d, a).

Example



	0	1	2	3	4	5
t	0	0	0	0	0	0
a	∞	-3	-3	-4	-6	-6
b	∞	∞	0	-2	-2	-2
c	∞	3	3	3	3	3
d	∞	4	3	3	2	0
e	∞	2	0	0	0	0

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1,\ s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

We can find the actual paths via tracing backwards:

$$M(5,d) = \min\{M(4,d)\ ,$$

$$\min_{w \in N(d)} (c_{dw} + M(4,w))\}$$

So the first edge must be (d, a).

Next we consider $M(4, a)$, etc.

Running Time

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

For $i = 1,2,\ldots,n-1$

For $v \in V$

$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$$

Return $M(n\text{-}1,s)$

Running Time

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

$O(1)$

For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Running Time

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots, n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

$O(1)$ $O(n)$

For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Running Time

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

$O(1)$ $O(n)$

$n-1$ For $i = 1,2,\ldots,n-1$

For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$

Return $M(n\text{-}1,s)$

Running Time

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

$O(1)$ $O(n)$

$n-1$ For $i = 1,2,\ldots,n-1$

n For $v \in V$

$$M(i,v) = \min\{M(i\text{-}1,v) \,,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$$

Return $M(n\text{-}1,s)$

Running Time

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0,\cdots,n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

$O(1)$ $O(n)$

$n-1$ For $i = 1,2,\ldots,n-1$

n For $v \in V$

$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min\limits_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$ $O(n)$

Return $M(n\text{-}1,s)$

Running Time

ShortestPath (G, s, t).

* Let $n = |V|$ *\

Define 2-D Array $M[0, \cdots, n-1, s, v_1, v_2, \ldots, t]$

Initialise $M[0,t] = 0$, and $M[0,v] = \infty$ for all other $v \in V$.

$O(1)$ $O(n)$

$n-1$ For $i = 1,2,\ldots,n-1$

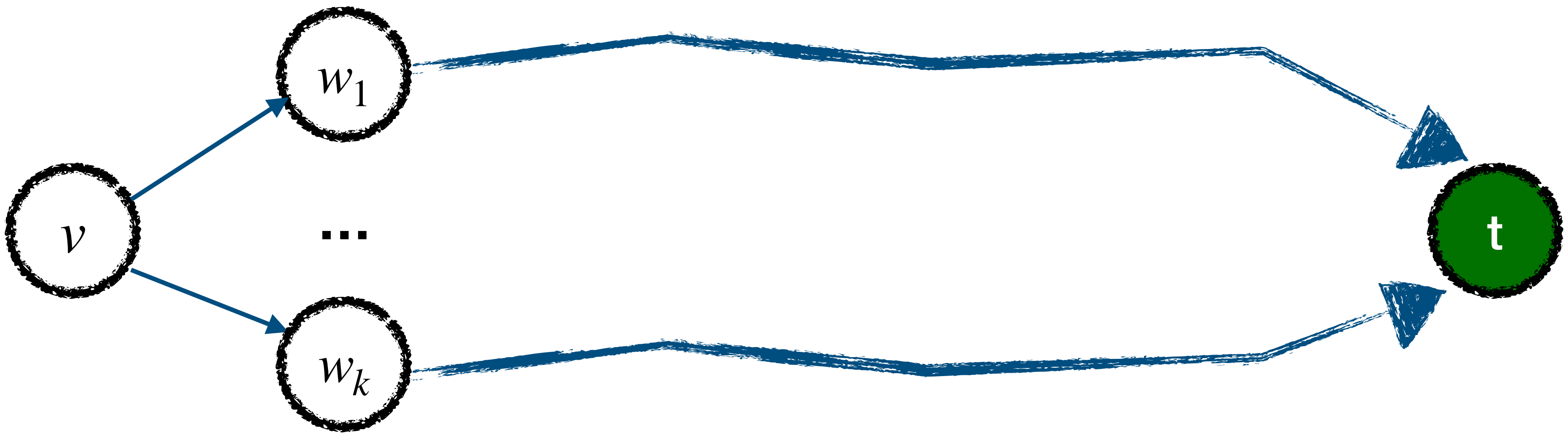
n For $v \in V$

$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\} \quad O(n)$$

Return $M(n\text{-}1,s)$

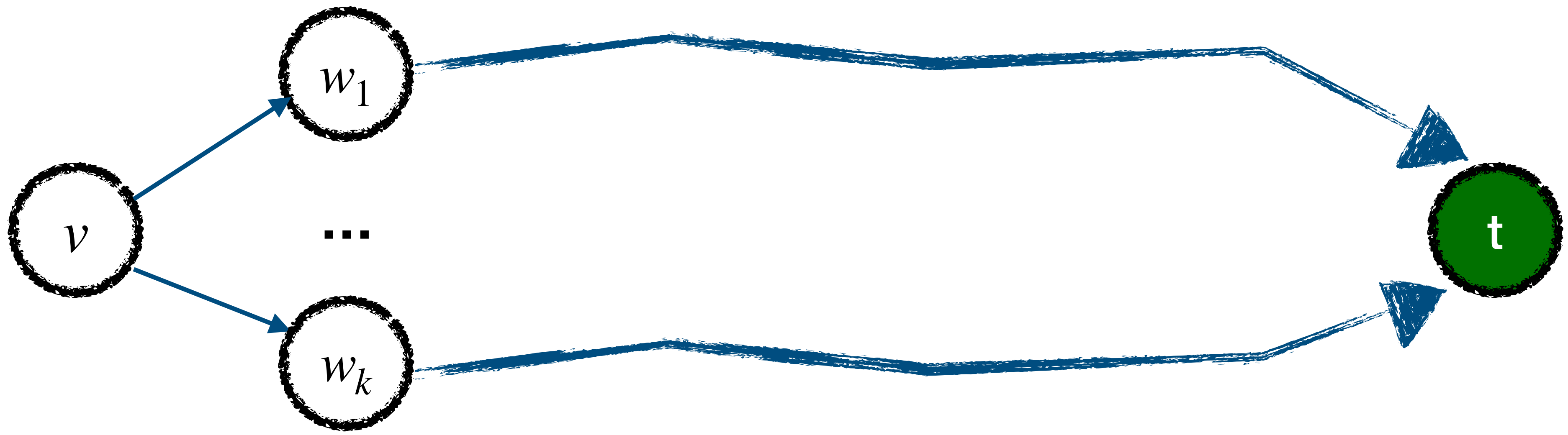
Overall: $O\left(n^3\right)$

Improved Analysis



$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$$

Improved Analysis



$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\}$$

Suffices to only check w such that $(v, w) \in E$.

Improved Analysis

Improved Analysis

Recall that $N(v)$ be the set of nodes w for which there is an edge (v, w), and let $n_v = |N(v)|$ be their number.

Improved Analysis

Recall that $N(v)$ be the set of nodes w for which there is an edge (v, w), and let $n_v = |N(v)|$ be their number.

$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\} \qquad \boxed{O(n_v)}$$

Improved Analysis

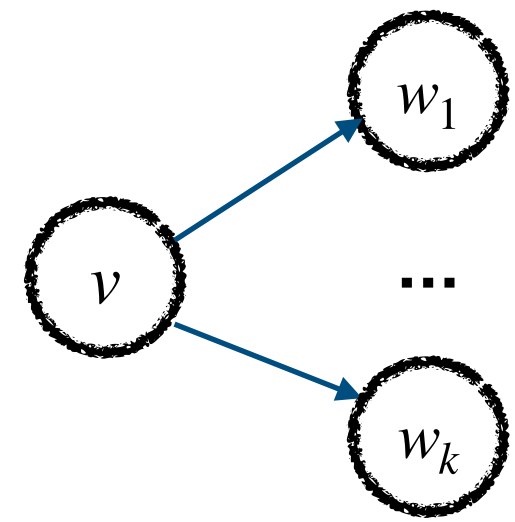
Recall that $N(v)$ be the set of nodes w for which there is an edge (v, w), and let $n_v = |N(v)|$ be their number.

$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\} \qquad \boxed{O(n_v)}$$

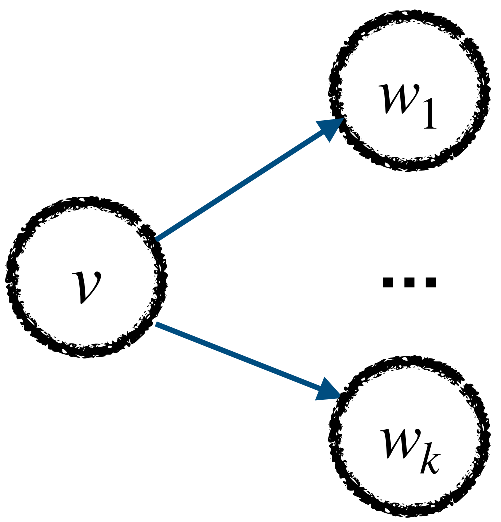
We have to compute an entry for every $v \in V$ and every index $i \in [0,n-1]$, so in total we need time $O\left(n \sum_{v \in V} n_v\right)$

Improved Analysis

How large is $\sum_{v \in V} n_v$?



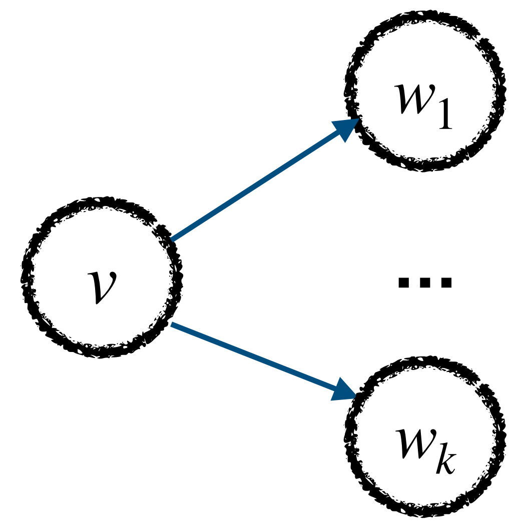
Improved Analysis



How large is $\sum_{v \in V} n_v$?

Each node v contributes exactly as many terms as the number of its outgoing edges (v, w).

Improved Analysis



How large is $\sum_{v \in V} n_v$?

Each node v contributes exactly as many terms as the number of its outgoing edges (v, w).

$$\sum_{v \in V} n_v = m$$

Improved Analysis

Let $N(v)$ be the set of nodes w for which there is an edge (v, w), and let $n_v = |N(v)|$ be their number.

$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\} \qquad \boxed{O(n_v)}$$

We have to compute an entry for every $v \in V$ and every index $i \in [0, n-1]$, so in total we need time $O\left(n \sum_{v \in V} n_v\right)$

Improved Analysis

Let $N(v)$ be the set of nodes w for which there is an edge (v, w), and let $n_v = |N(v)|$ be their number.

$$M(i,v) = \min\{M(i\text{-}1,v)\ ,\ \min_{w \in N(v)} (c_{vw} + M(i\text{-}1,w))\} \qquad \boxed{O(n_v)}$$

We have to compute an entry for every $v \in V$ and every index $i \in [0,n-1]$, so in total we need time $O\left(n \sum_{v \in V} n_v\right)$

Overall: $O(nm)$

Improved Memory Implementation

Improved Memory Implementation

- We need to store the 2D array M in memory, which takes space $\Theta(n^2)$.

Improved Memory Implementation

- We need to store the 2D array M in memory, which takes space $\Theta(n^2)$.
- Memory usage is a common problem with many dynamic programming algorithms.

Improved Memory Implementation

- We need to store the 2D array M in memory, which takes space $\Theta(n^2)$.
- Memory usage is a common problem with many dynamic programming algorithms.
- Here, we can instead use an 1D array M' of space $O(n)$.

Improved Memory Implementation

- We need to store the 2D array M in memory, which takes space $\Theta(n^2)$.
- Memory usage is a common problem with many dynamic programming algorithms.
- Here, we can instead use an 1D array M' of space $O(n)$.
- We will store only $M'[v]$ for every node $v \in V$ rather than $M'[i,v]$ for all i and all $v \in V$.

Improved Memory Implementation

- We need to store the 2D array M in memory, which takes space $\Theta(n^2)$.
- Memory usage is a common problem with many dynamic programming algorithms.
- Here, we can instead use an 1D array M' of space $O(n)$.
- We will store only $M'[v]$ for every node $v \in V$ rather than $M'[i,v]$ for all i and all $v \in V$.
    - This will be the length of the shortest path $v \sim t$ that we have found so far.

Improved Memory Implementation

- We will store only $M'[v]$ for every node $v \in V$ rather than $M'[i,v]$ for all i and all $v \in V$.
    - This will be the length of the shortest path $v \sim t$ that we have found so far.

Improved Memory Implementation

- We will store only $M'[v]$ for every node $v \in V$ rather than $M'[i,v]$ for all i and all $v \in V$.
    - This will be the length of the shortest path v~t that we have found so far.
- We still iterate over all i and all v as before, but i is only a *counter*.

Improved Memory Implementation

- We will store only $M'[v]$ for every node $v \in V$ rather than $M'[i,v]$ for all i and all $v \in V$.
  - This will be the length of the shortest path v~t that we have found so far.
- We still iterate over all i and all v as before, but i is only a *counter*.
- We use the following recurrence relation:

$$M[v] = \min\{M[v]\ ,\ \min_{w \in N(v)} (c_{vw} + M[w])\}$$

Improved Memory Implementation

- We will store only $M'[v]$ for every node $v \in V$ rather than $M'[i,v]$ for all i and all $v \in V$.
  - This will be the length of the shortest path $v \sim t$ that we have found so far.
- We still iterate over all i and all v as before, but i is only a *counter*.
- We use the following recurrence relation:

$$M[v] = \min\{M[v]\ ,\ \min_{w \in N(v)} (c_{vw} + M[w])\}$$

- That works, however more work is needed to recover the shortest paths!

Shortest Paths in Graphs

- **Input:** A directed graph $G = (V, E)$, and designated nodes s, t in V. We also assume that every node u in V is reachable from s, and *that the graph does not have any negative cycles.* We are also given a cost $c_e \in \mathbb{R}$ for every edge e in E.

- **Output:** A shortest path $s{\sim}t$ from s to t. In other words, a path P that minimises $\sum_{(u,v)\in P} c_{uv}$

Shortest Paths in Graphs

- **Input:** A directed graph $G = (V, E)$, and designated nodes s, t in V. We also assume that every node u in V is reachable from s, and *that the graph does not have any negative cycles.* We are also given a cost $c_e \in \mathbb{R}$ for every edge e in E.

- **Output:** A shortest path $s{\sim}t$ from s to t. In other words, a path P that minimises $\sum_{(u,v)\in P} c_{uv}$

What if the graph has negative cycles? Can we at least detect that?

Detecting Negative Cycles

- Can be done in time $O(mn)$.
- It is in fact usually included as a part of the Bellman-Ford algorithm.
- We will not cover this here, see KT 6.10 for the details if you are interested.

Reading and References

- Kleinberg and Tardos 6.8.
- CLRS 22.1.
- Roughgarden 18.1, 18.2.
- The Bellman-Ford visualiser: https://algorithms.discrete.ma.tum.de/graph-algorithms/spp-bellman-ford/index_en.html