Discrete Optimization
Lecture 1

Shortest paths

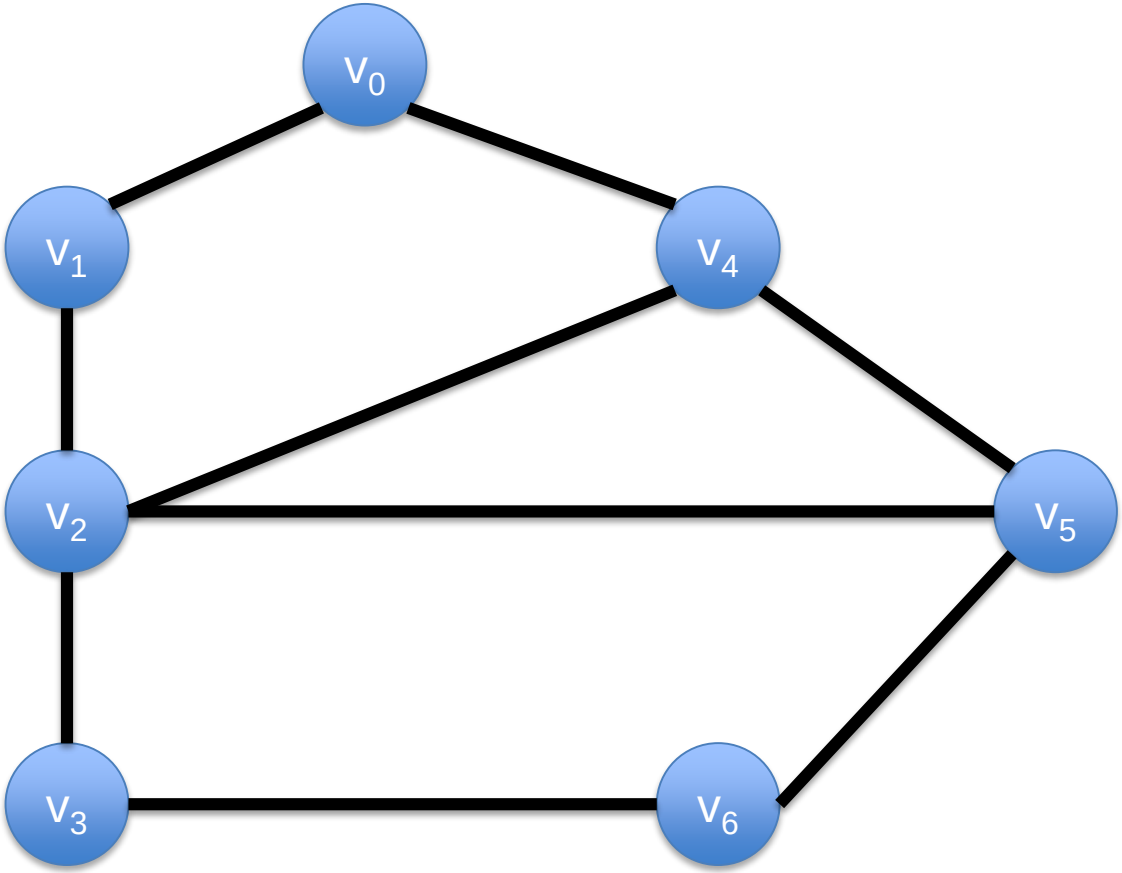
Slides courtesy of M. Pawan Kumar

Slides available online https://project.inria.fr/2015ma2827/

Outline

- Graph Preliminaries
  - Undirected Graphs
  - Directed Graphs
- Complexity Preliminaries
- Shortest Path Algorithms

Undirected Graphs

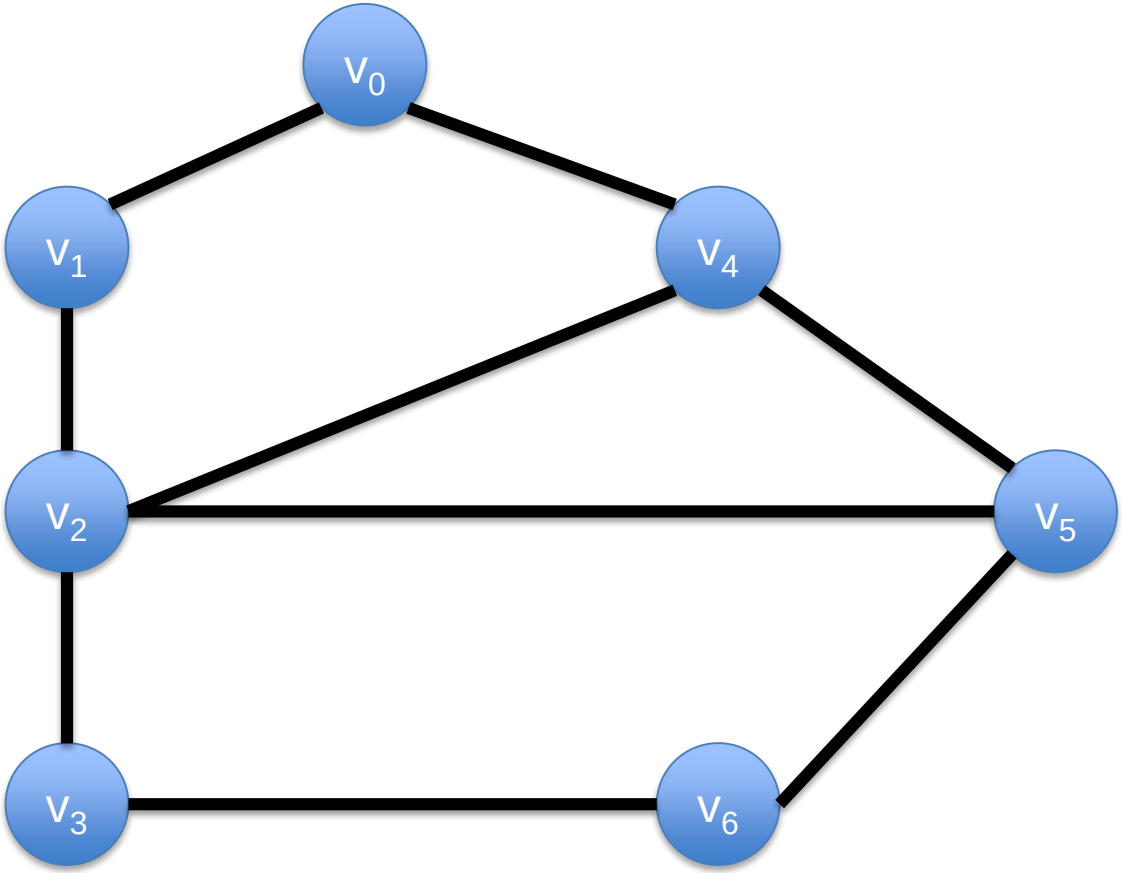


G = (V, E)

‘n’ vertices or nodes V

‘m’ edges E: unordered pairs from V

Neighboring or Adjacent Vertices

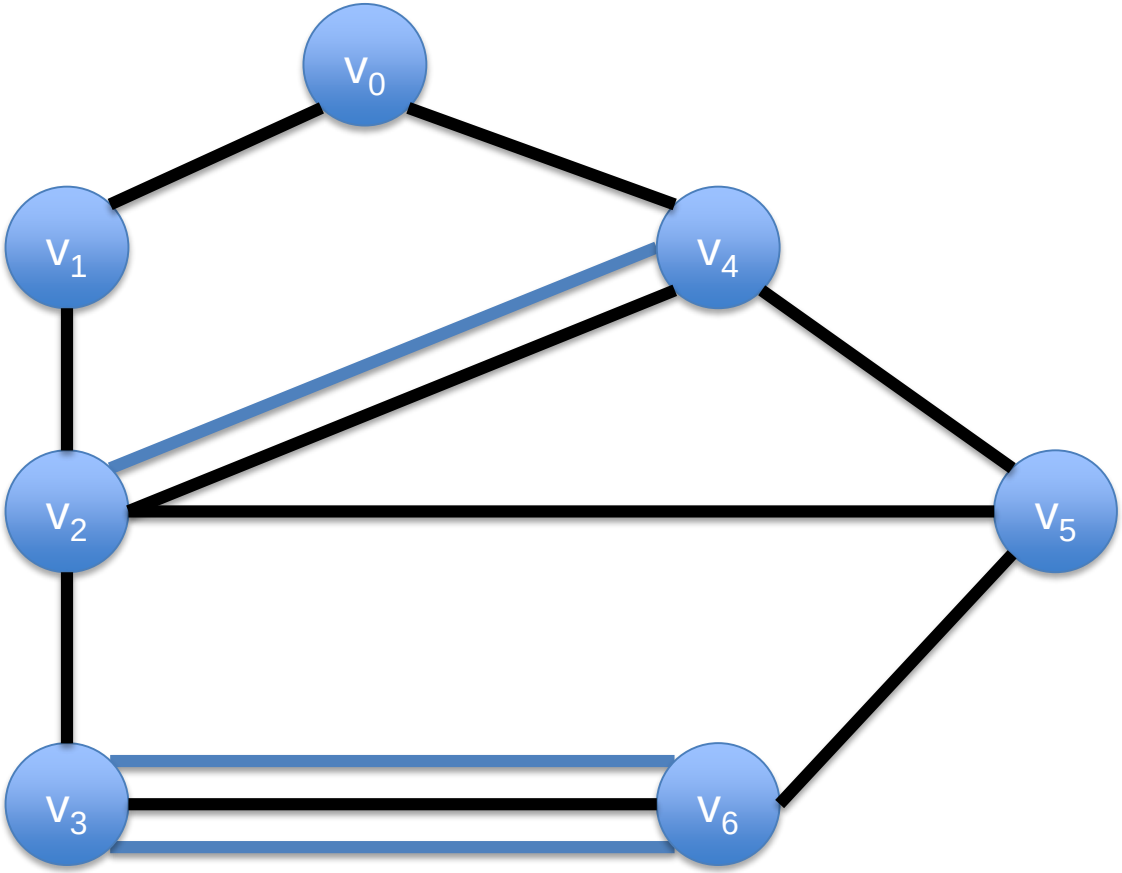


G = (V, E)

Connected by an edge e = (u,v) = (v,u).

‘v_0’ and ‘v_1’ adjacent, ‘v_0’ and ‘v_5’ not adjacent, …

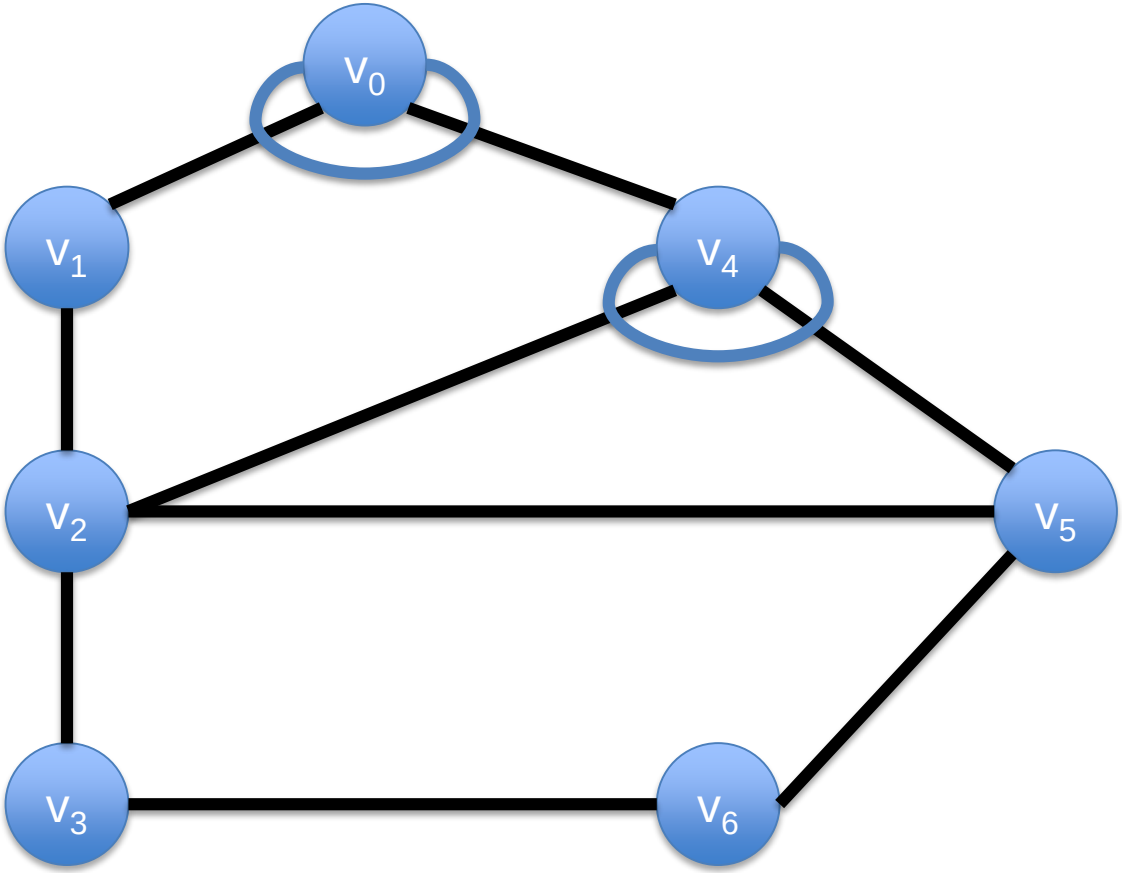
Parallel Edges



G = (V, E)

Represented by the same pair of vertices.

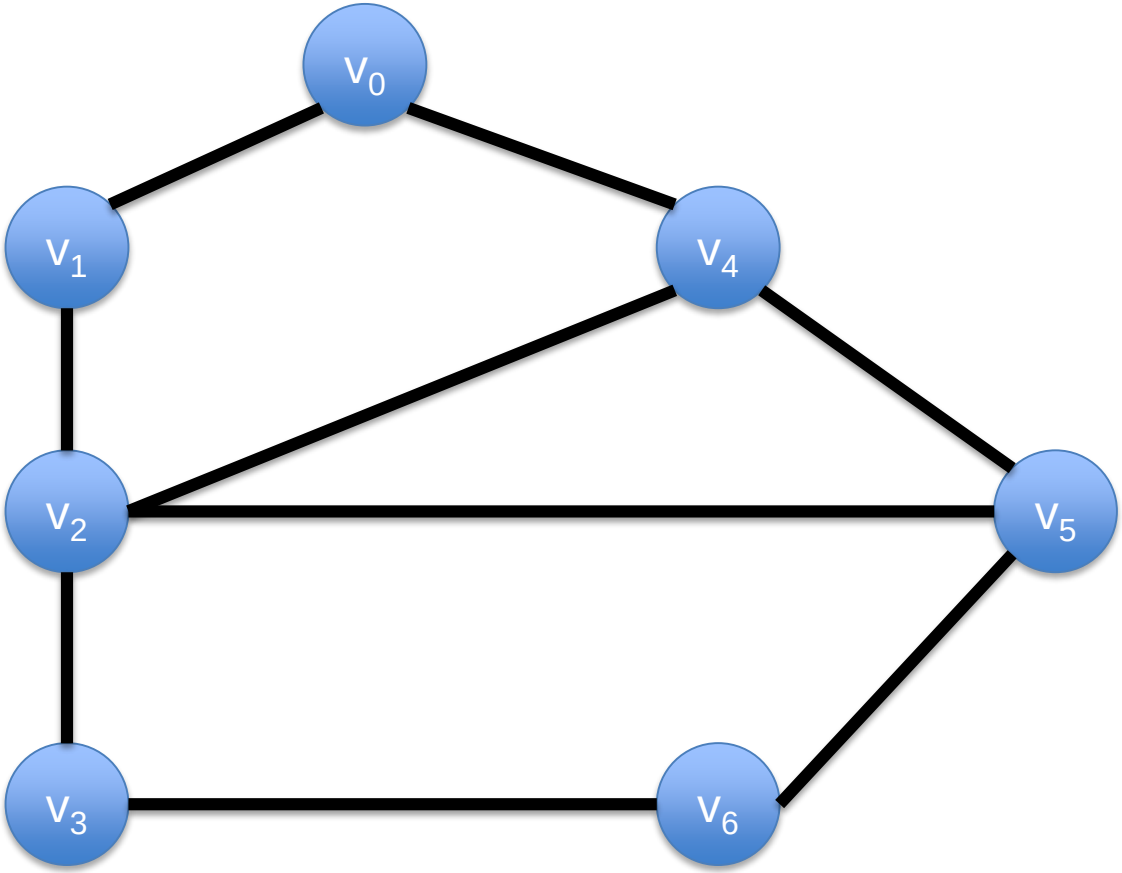
Loops



G = (V, E)

Edges that connect a vertex to itself.

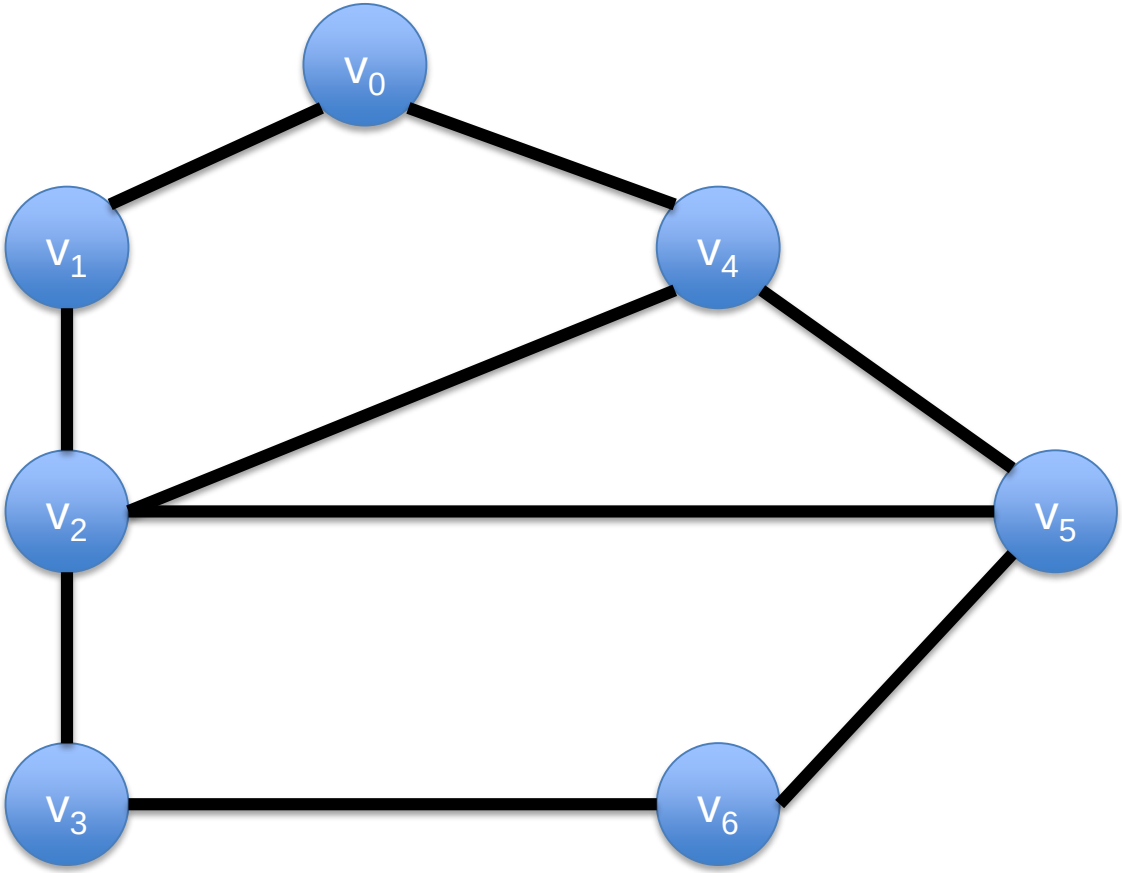
Simple Graphs



G = (V, E)

Graphs without parallel edges and without loops.

Degree of a Vertex

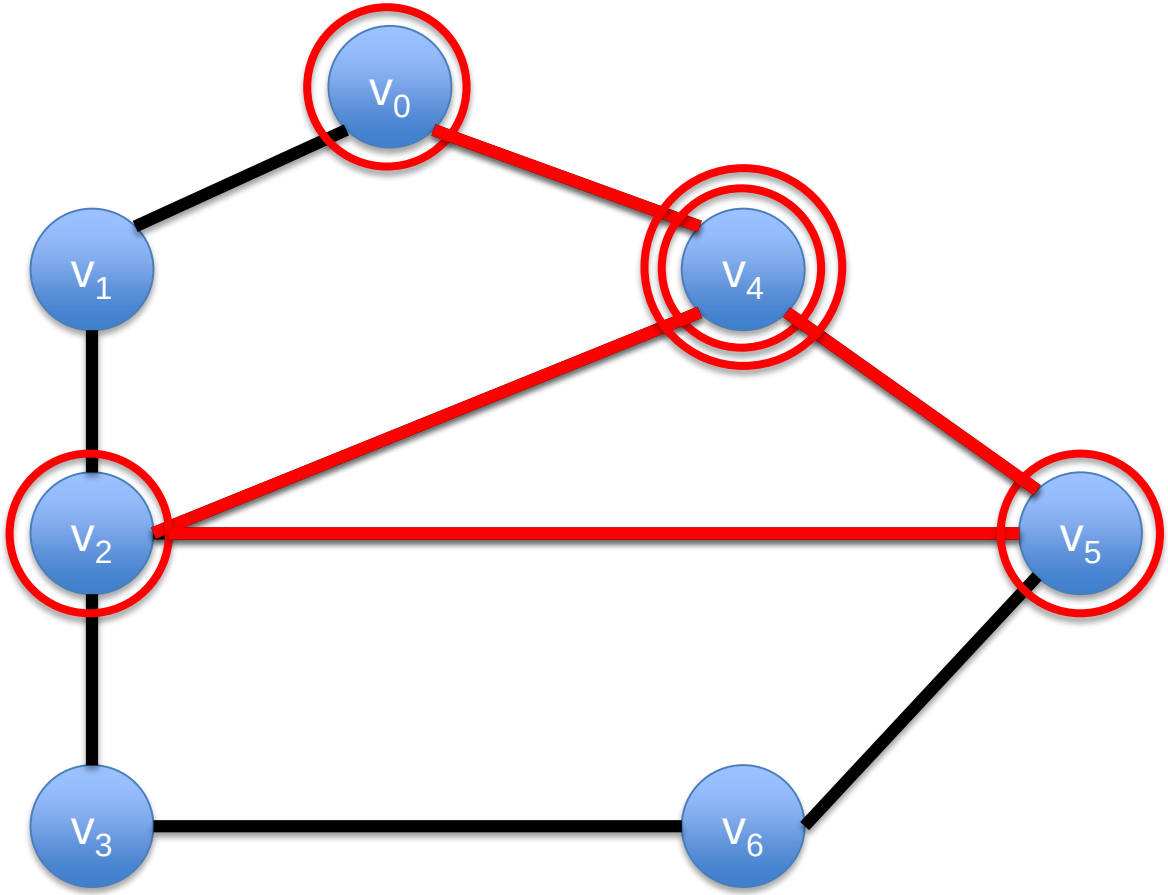


G = (V, E)

Number of edges incident on the vertex.

$\deg(v_0) = 2$, $\deg(v_1) = 2$, $\deg(v_4) = 3$, …

Walk

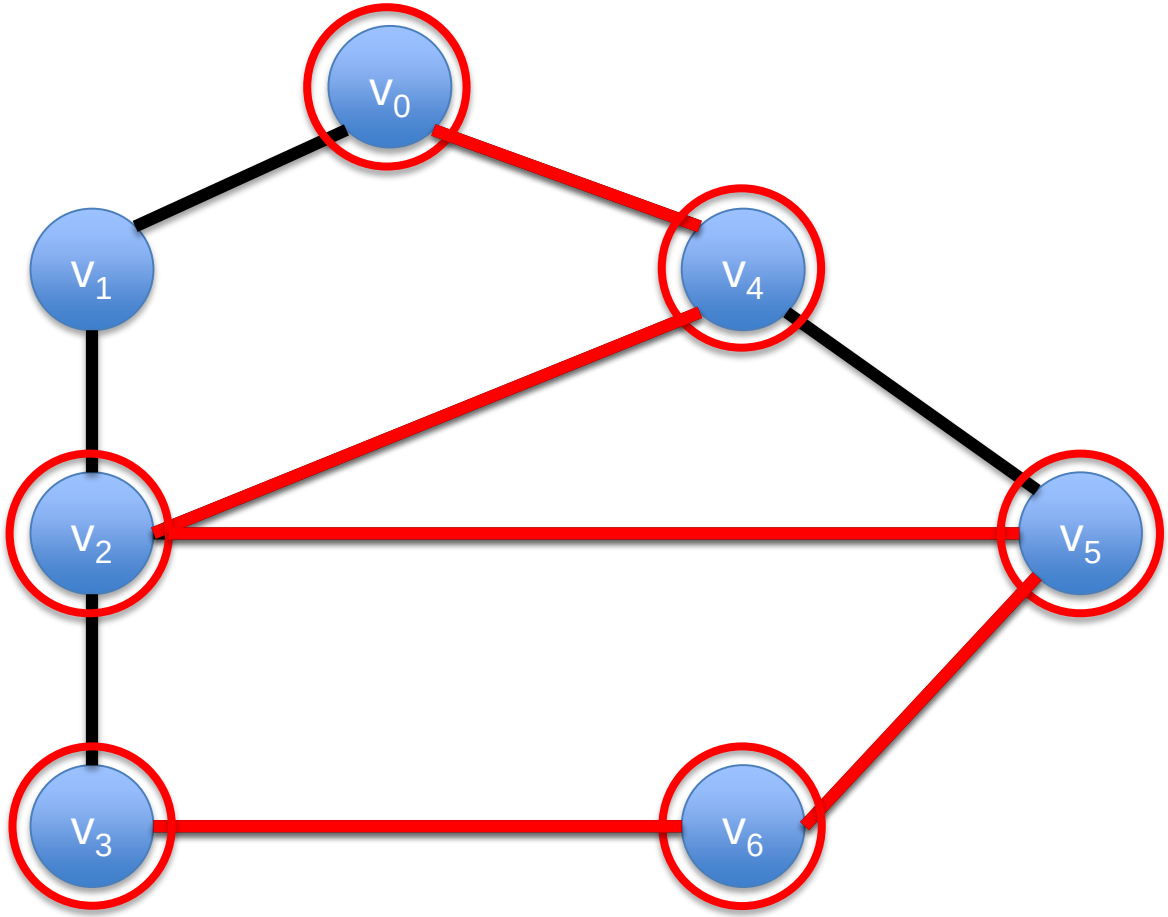


G = (V, E)

Sequence $P = (v_0,e_1,v_1,\ldots,e_k,v_k)$, $e_i = (v_{i-1},v_i)$

v_0, (v_0,v_4), v_4, (v_4,v_2), v_2, (v_2,v_5), v_5, (v_5,v_4), v_4

Path

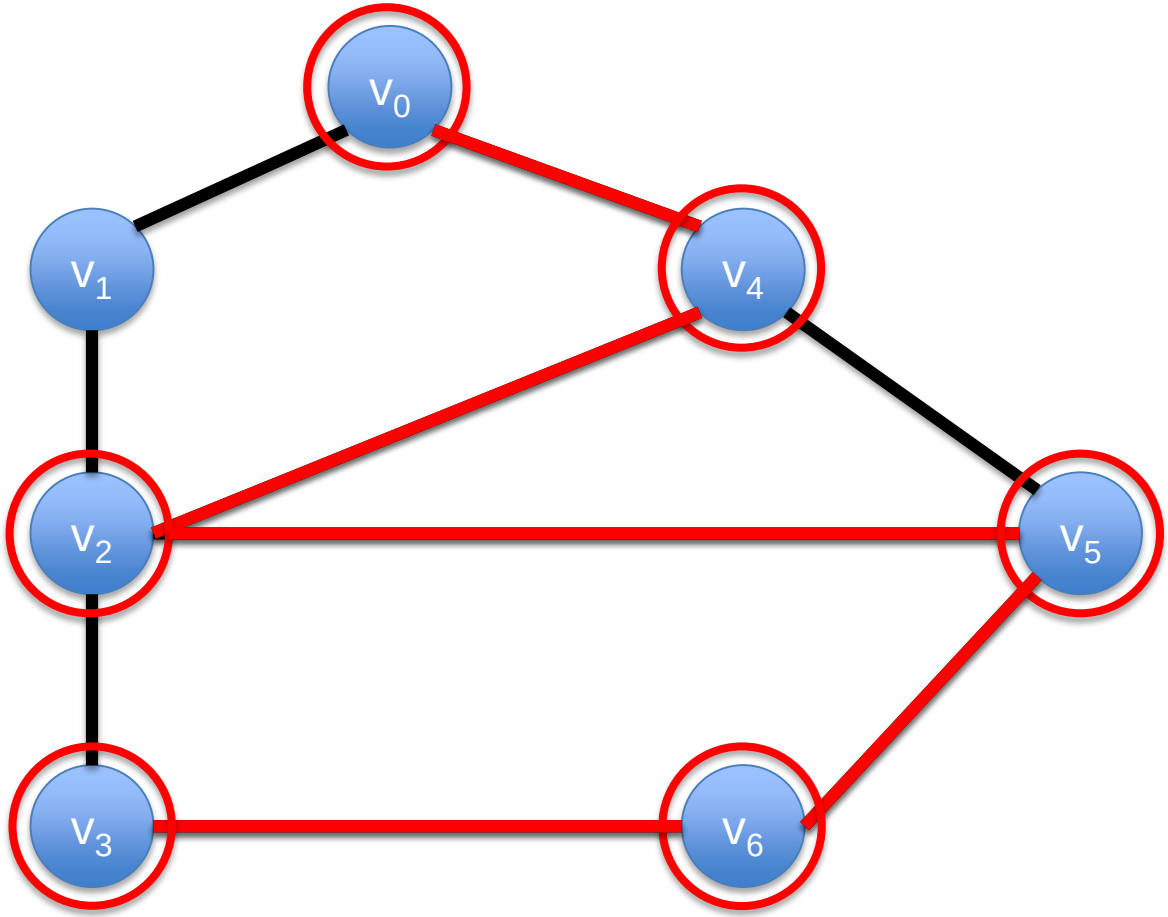


G = (V, E)

Sequence $P = (v_0, e_1, v_1, \ldots, e_k, v_k)$, $e_i = (v_{i-1}, v_i)$

Vertices $v_0, v_1, \ldots, v_k$ are distinct

Length of a Walk

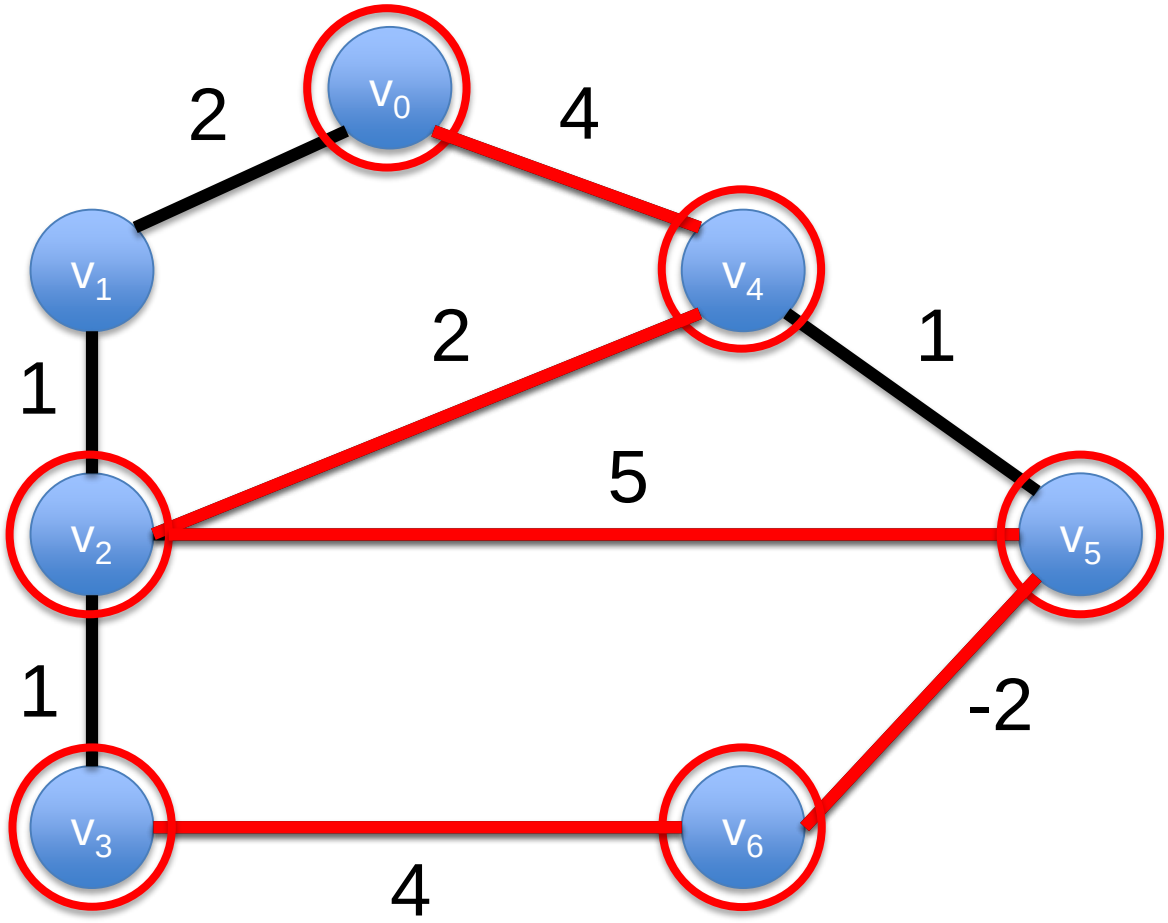


G = (V, E)

Number of edges: k

Length of above walk = 5

Length of a Walk



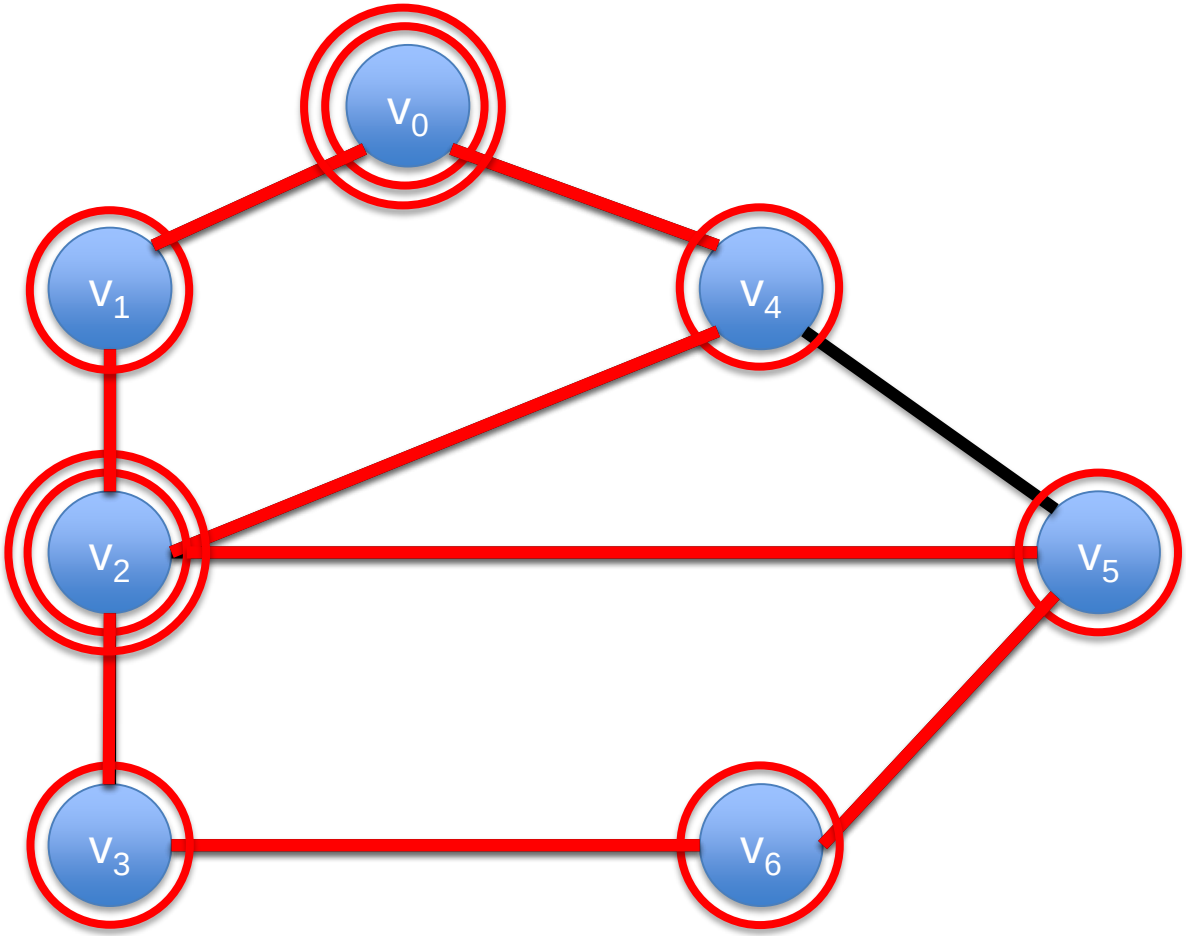
G = (V, E)

l: E ➔ R

Sum of lengths of all edges: $\Sigma_i\ l(e_i)$

Length of above walk = 4+2+5-2+4 = 13

Closed Walk

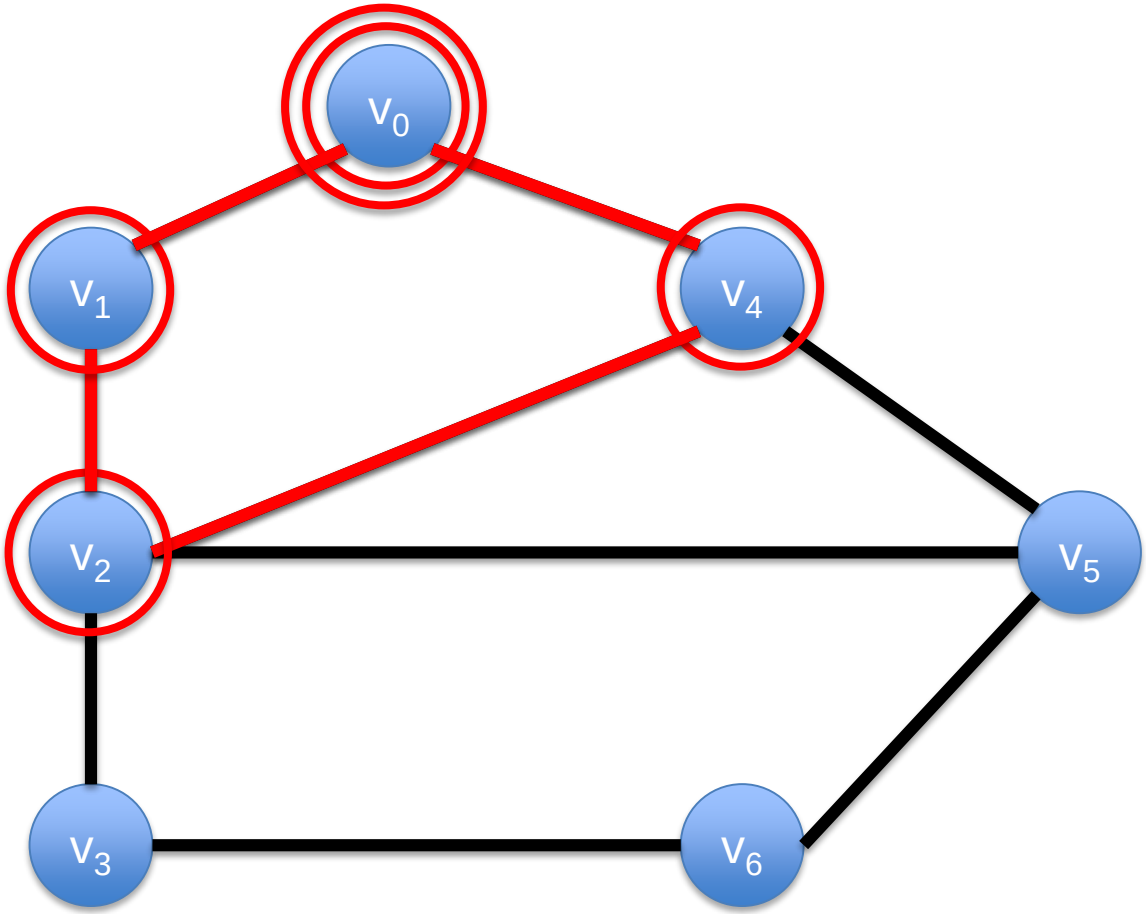


G = (V, E)

Sequence $P = (v_0, e_1, v_1, \ldots, e_k, v_k)$, $e_i = (v_{i-1}, v_i)$

$v_0 = v_k$

Circuit



G = (V, E)

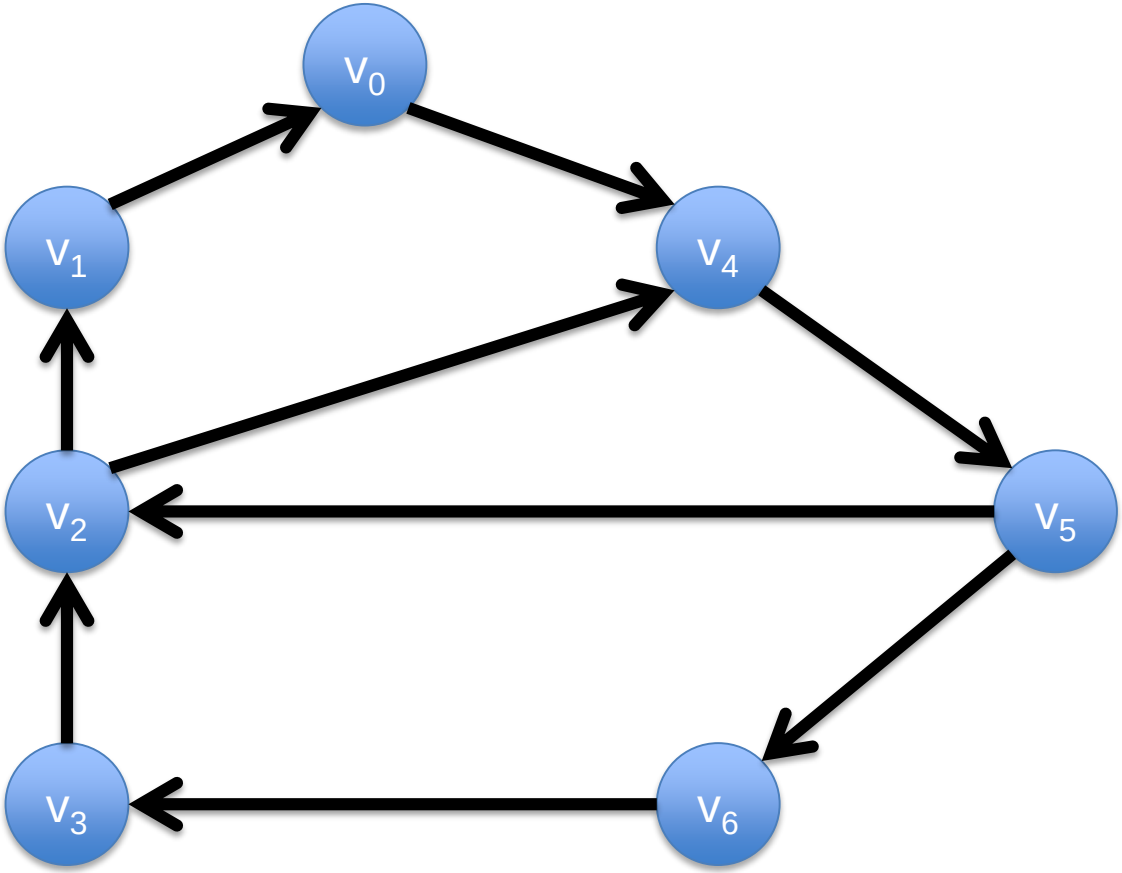
Sequence P = $(v_0,e_1,v_1,\ldots,e_k,v_k)$, $e_i = (v_{i-1},v_i)$

$v_0 = v_k$ Vertices $v_0,v_1,\ldots,v_{k-1}$ are distinct

Outline

- Graph Preliminaries
  - Undirected Graphs
  - **Directed Graphs**
- Complexity Preliminaries
- Shortest Path Algorithms

Directed Graphs (Digraphs)

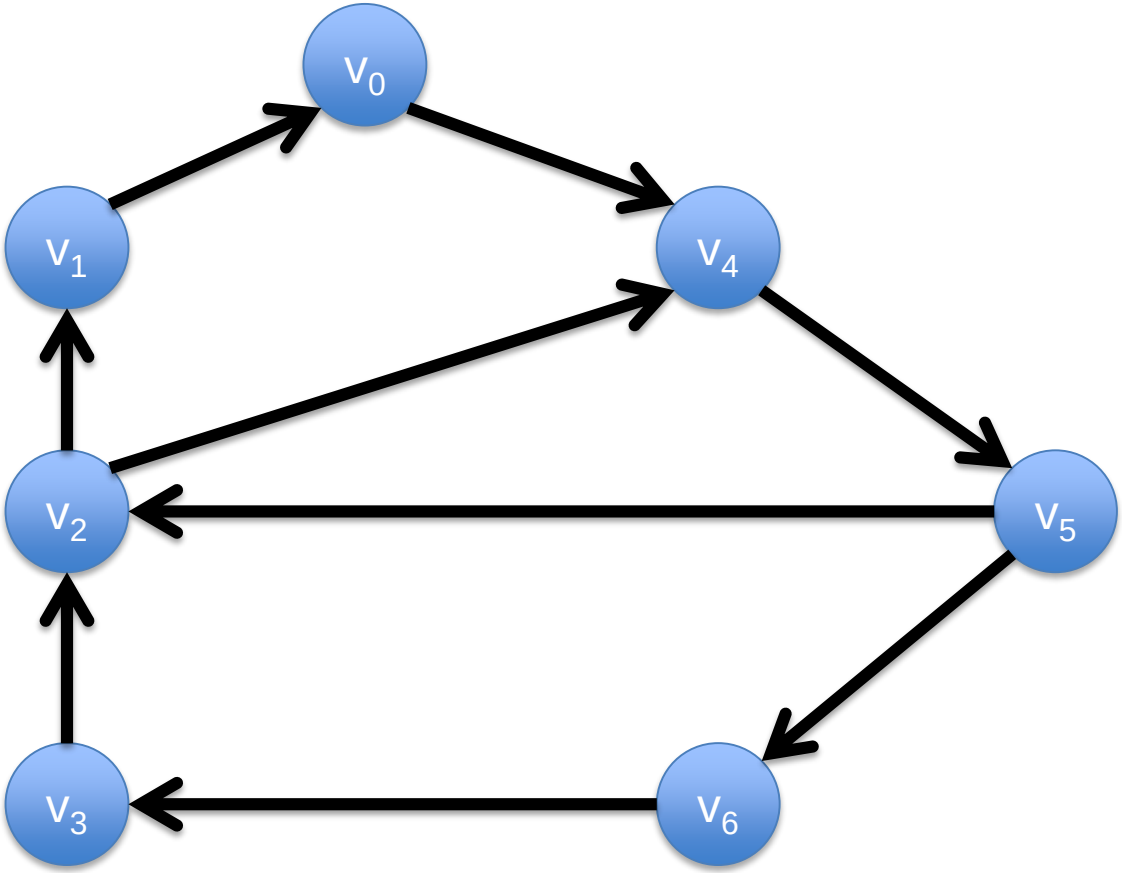


D = (V, A)

‘n’ vertices or nodes V

‘m’ arcs A: ordered pairs from V

Neighboring or Adjacent Vertices

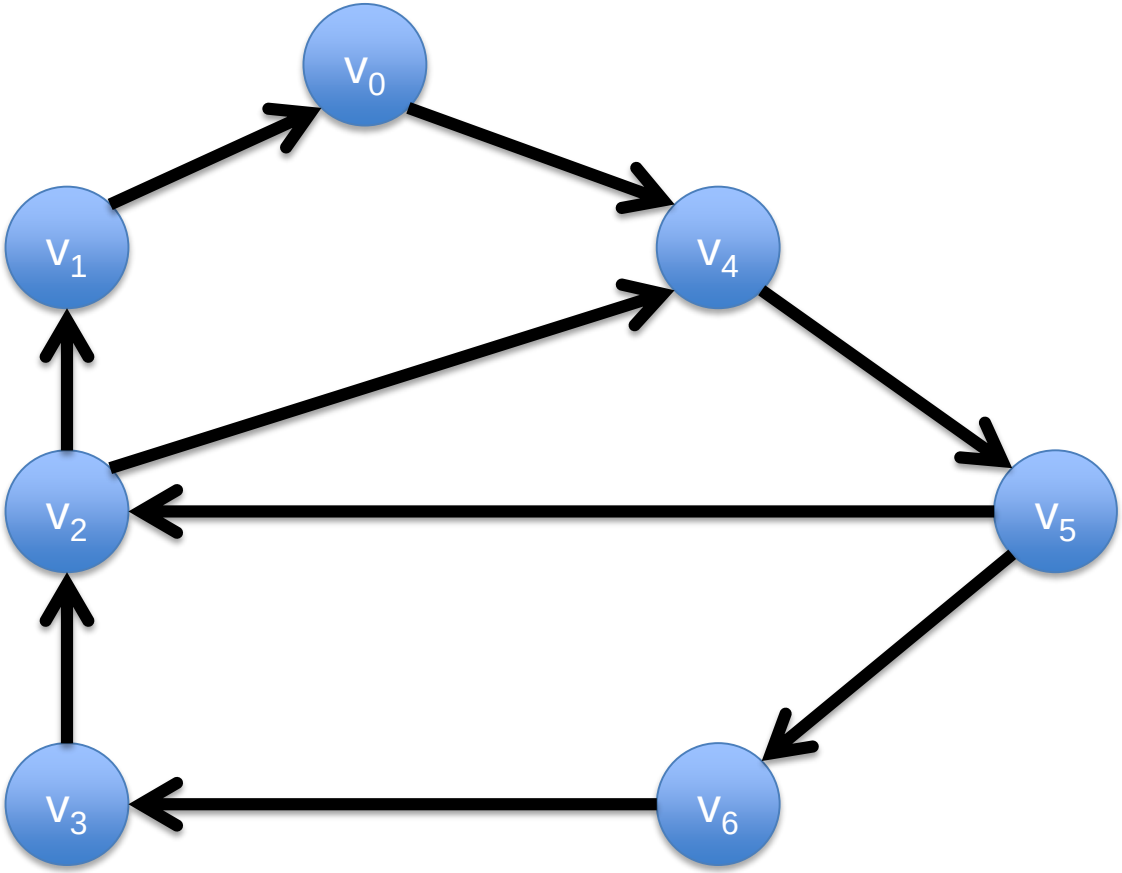


D = (V, A)

Connected by an arc a = (u,v).

‘v_0’ is the inneighbor of ‘v_4’

Neighboring or Adjacent Vertices

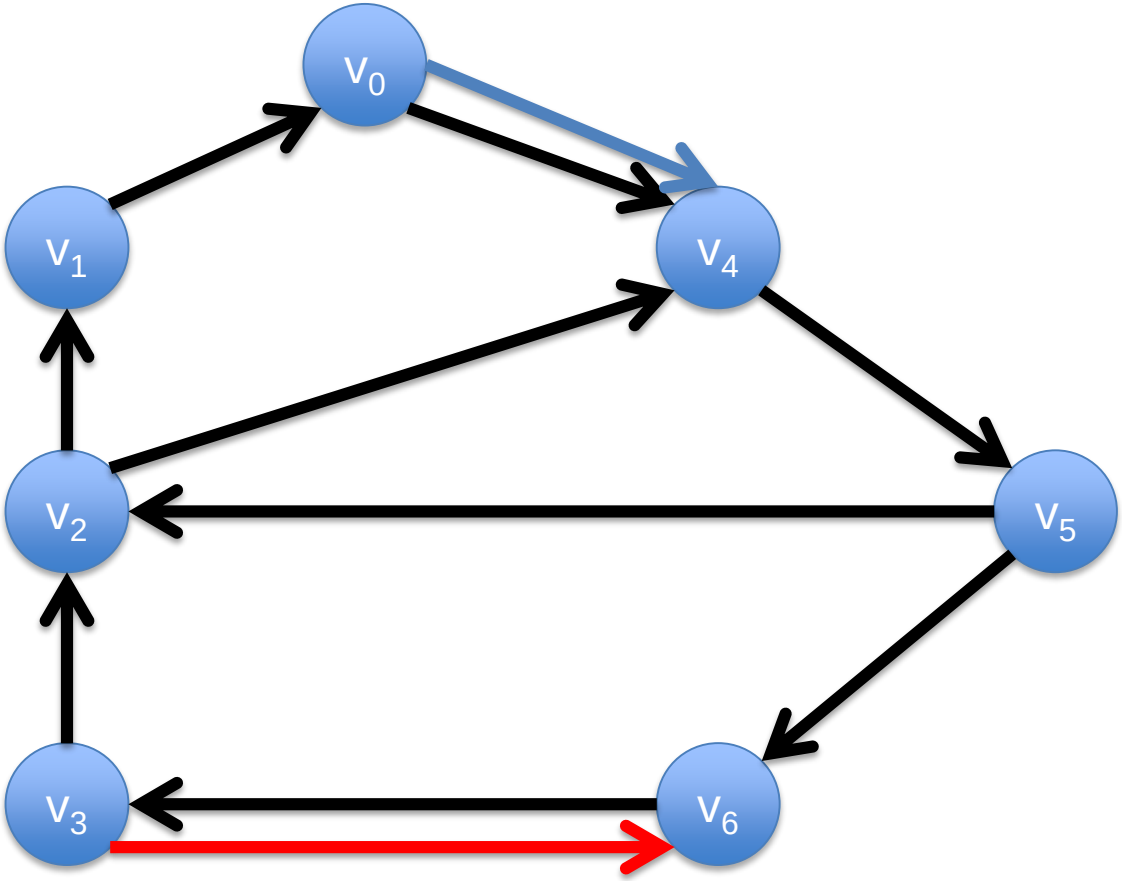


D = (V, A)

Connected by an arc a = (u,v).

‘v_4’ is the outneighbor of ‘v_0’

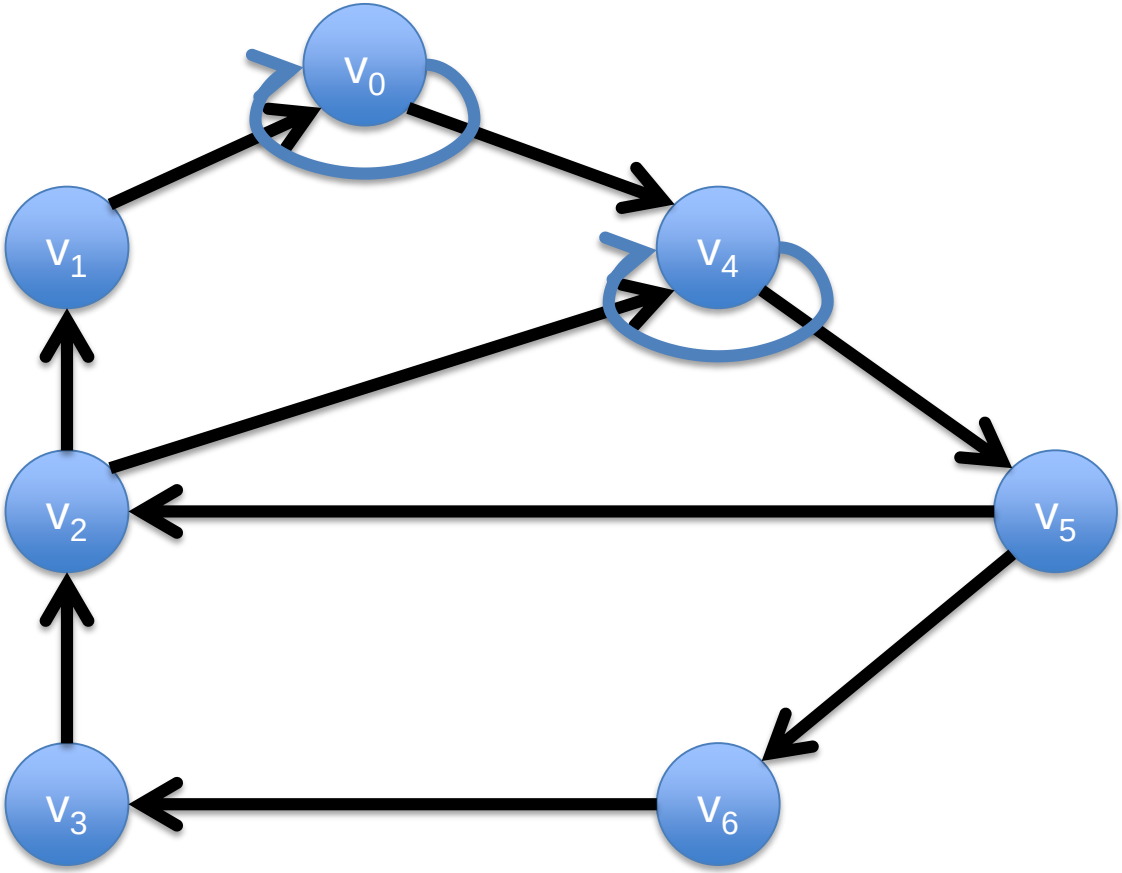
Parallel Arcs



D = (V, A)

Represented by the same ordered pair of vertices.

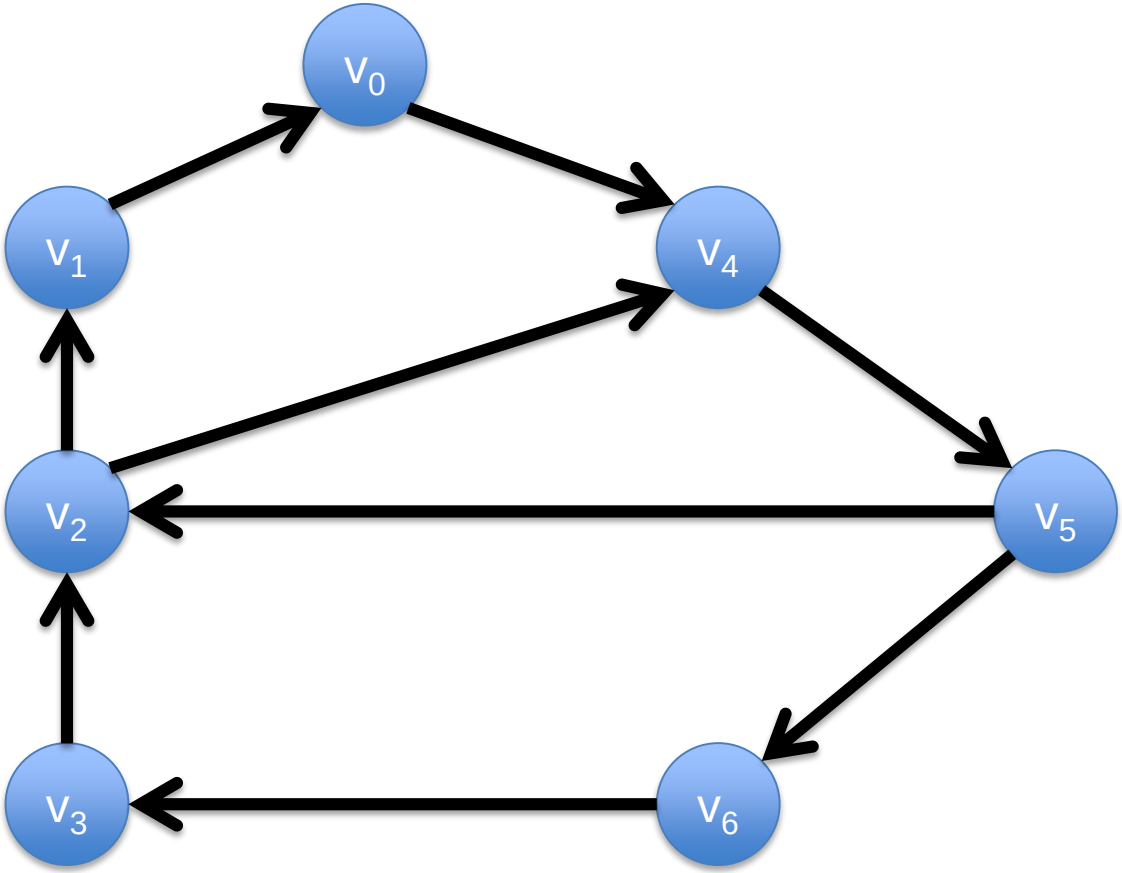
Loops



D = (V, A)

Arcs that connect a vertex to itself.

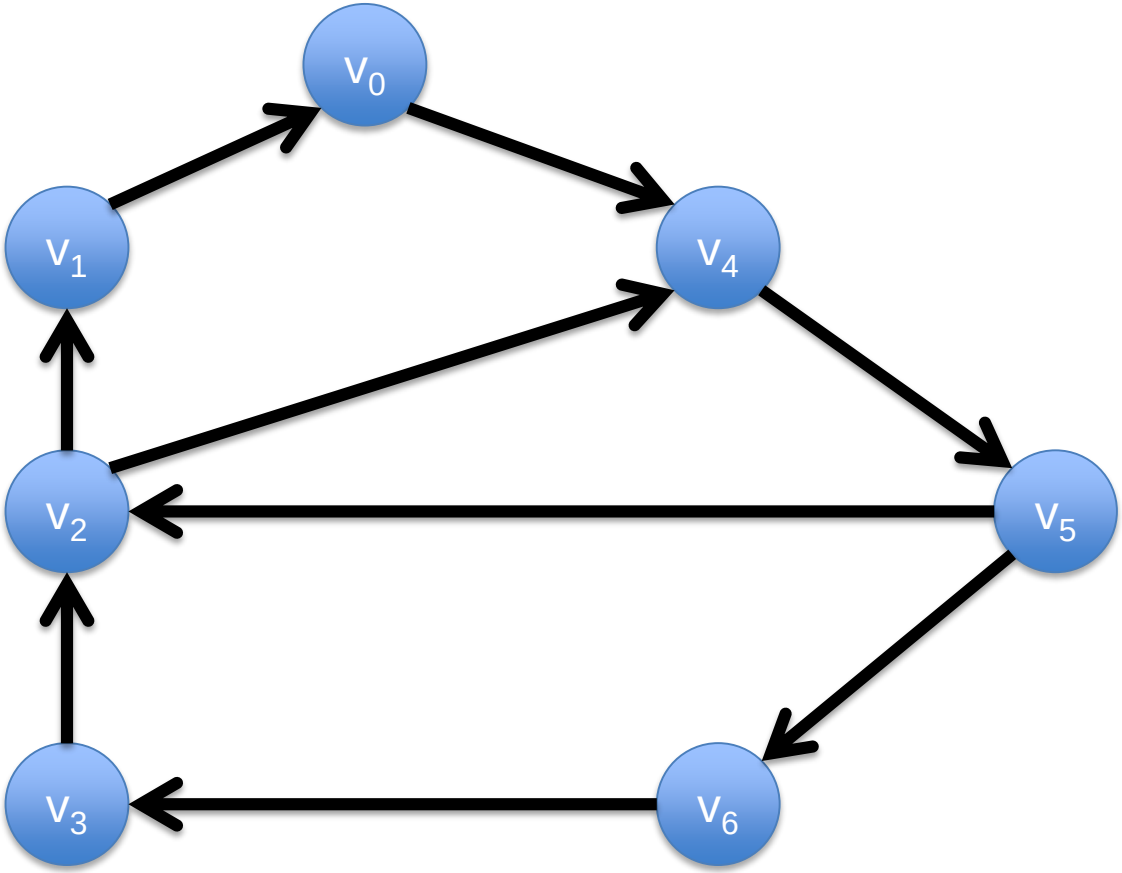
Simple Graphs



D = (V, A)

Graphs without parallel arcs and without loops.

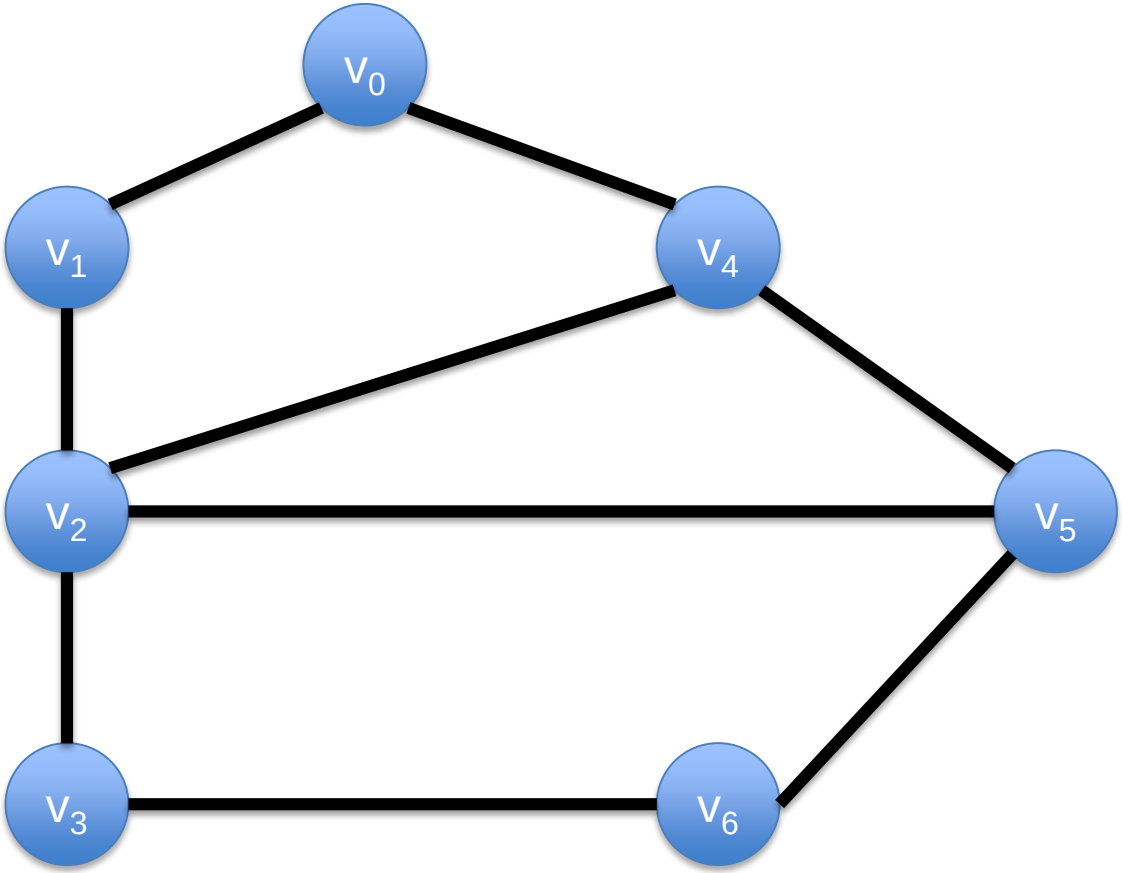
Underlying Undirected Graph



D = (V, A)

Graphs obtained by ignoring orientation of arcs.

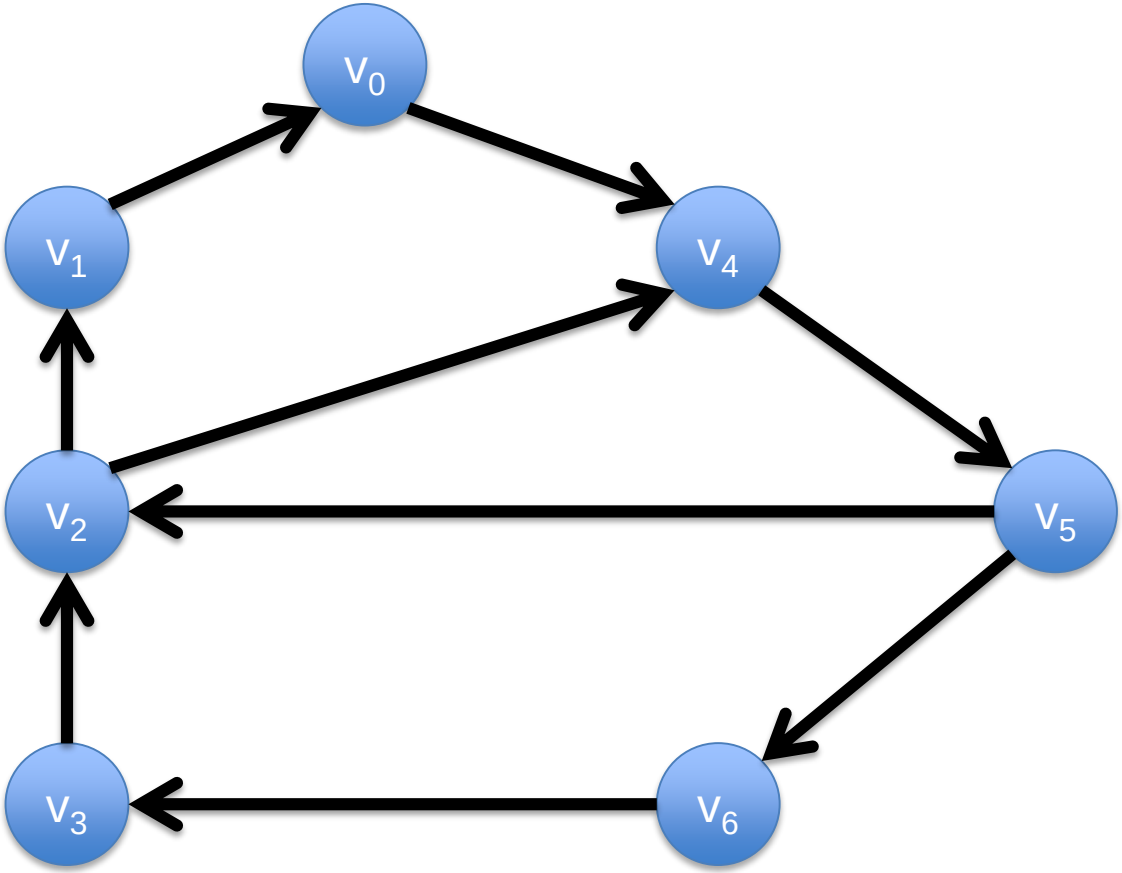
Underlying Undirected Graph



G = (V, E)

Graphs obtained by ignoring orientation of arcs.

Indegree of a Vertex

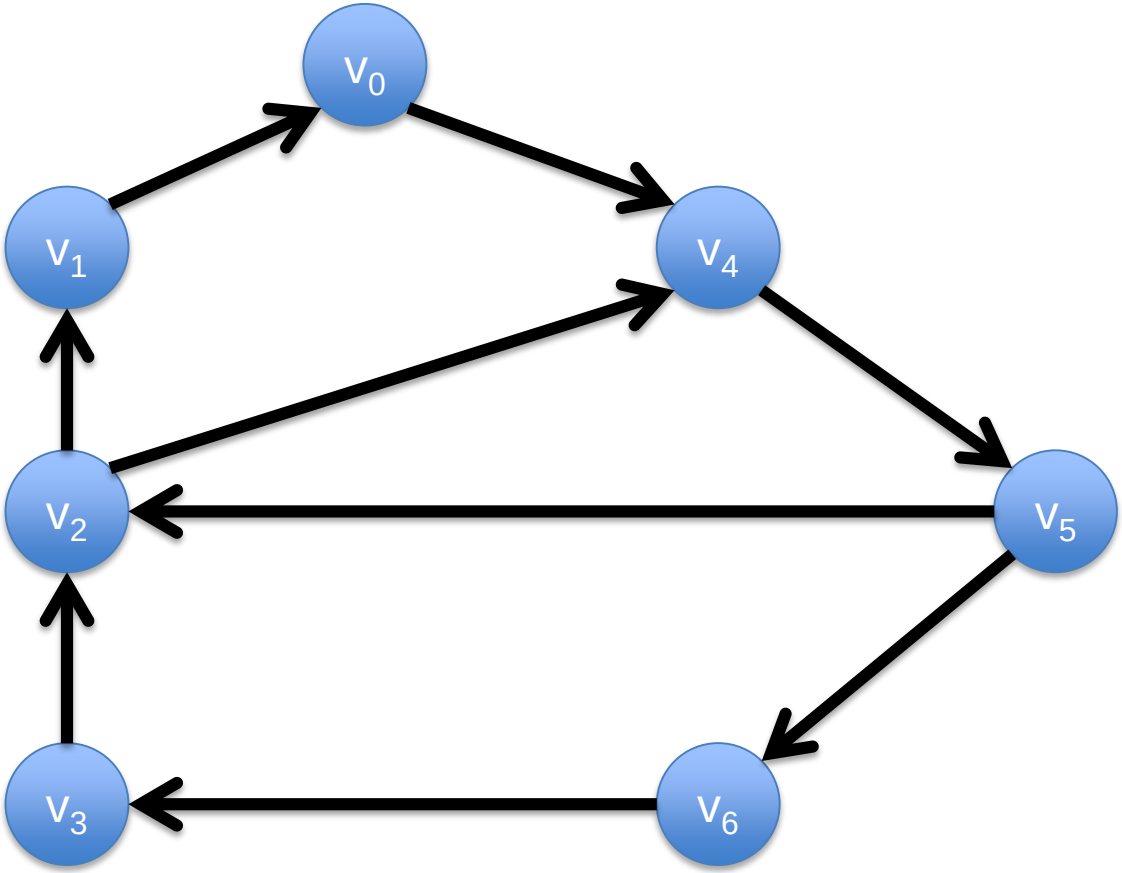


D = (V, A)

Number of arcs entering the vertex.

$indeg(v_0) = 1$, $indeg(v_1) = 1$, $indeg(v_4) = 2$, …

Outdegree of a Vertex

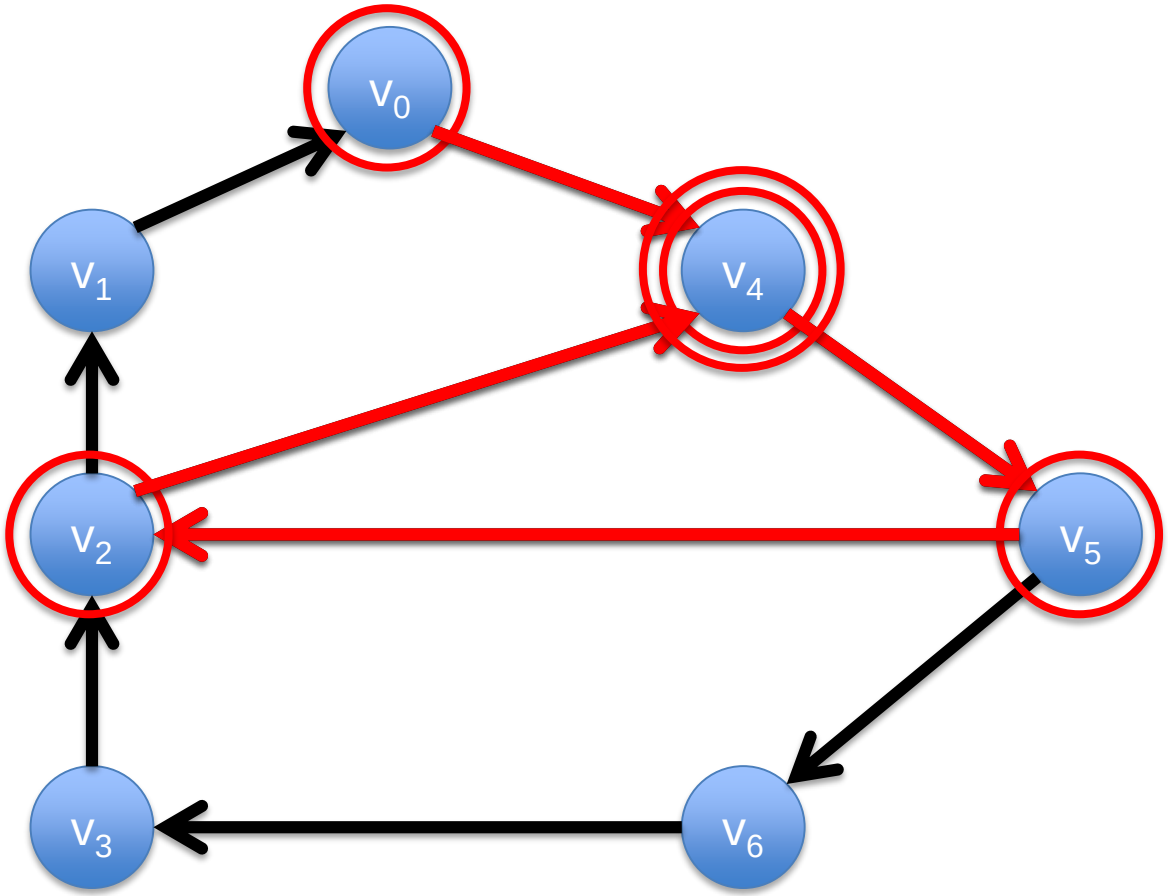


D = (V, A)

Number of arcs leaving the vertex.

outdeg(v_0) = 1, outdeg(v_1) = 1, outdeg(v_2) = 2, …

Walk

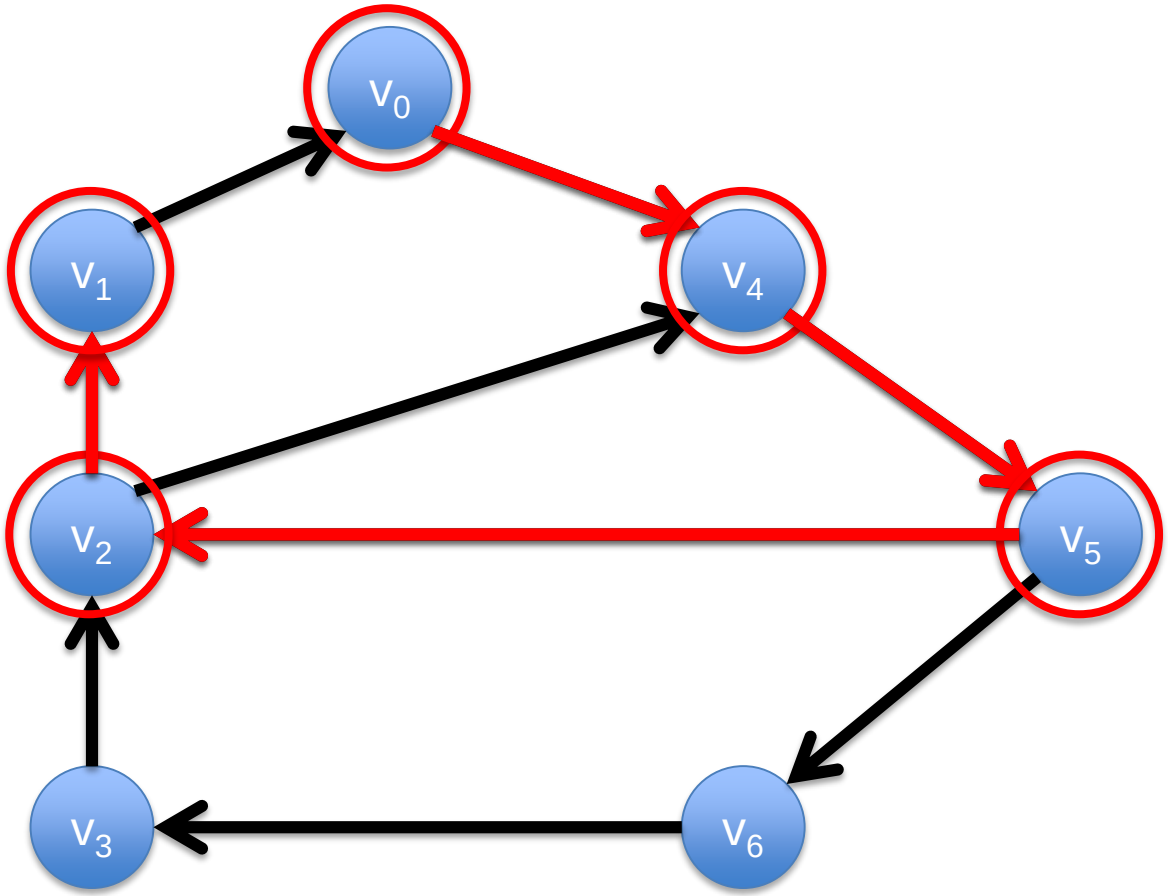


D = (V, A)

Sequence $P = (v_0,a_1,v_1,\ldots,a_k,v_k)$, $a_i = (v_{i-1},v_i)$

v_0, (v_0,v_4), v_4, (v_4,v_5), v_5, (v_5,v_2), v_2, (v_2,v_4), v_4

Path

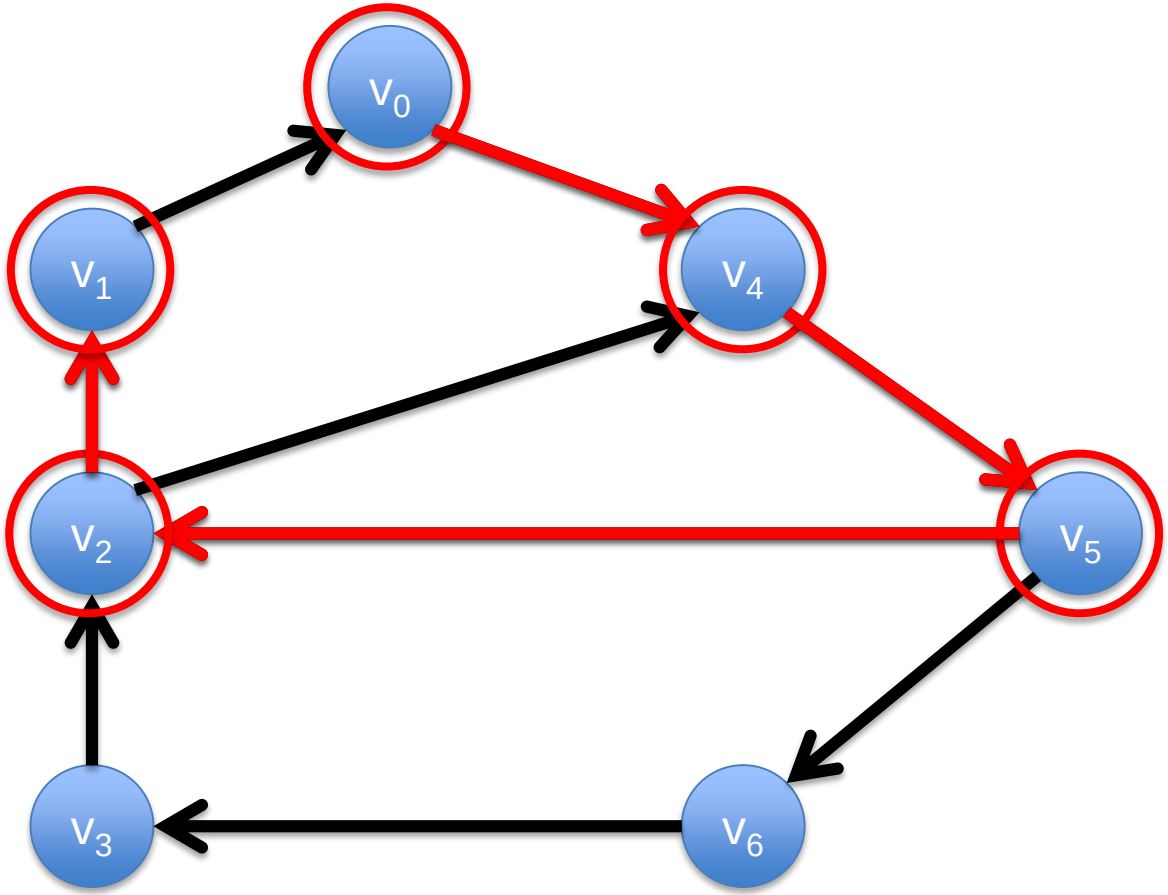


D = (V, A)

Sequence $P = (v_0,a_1,v_1,\ldots,a_k,v_k)$, $a_i = (v_{i-1},v_i)$

Vertices $v_0,v_1,\ldots,v_k$ are distinct

Length of a Walk

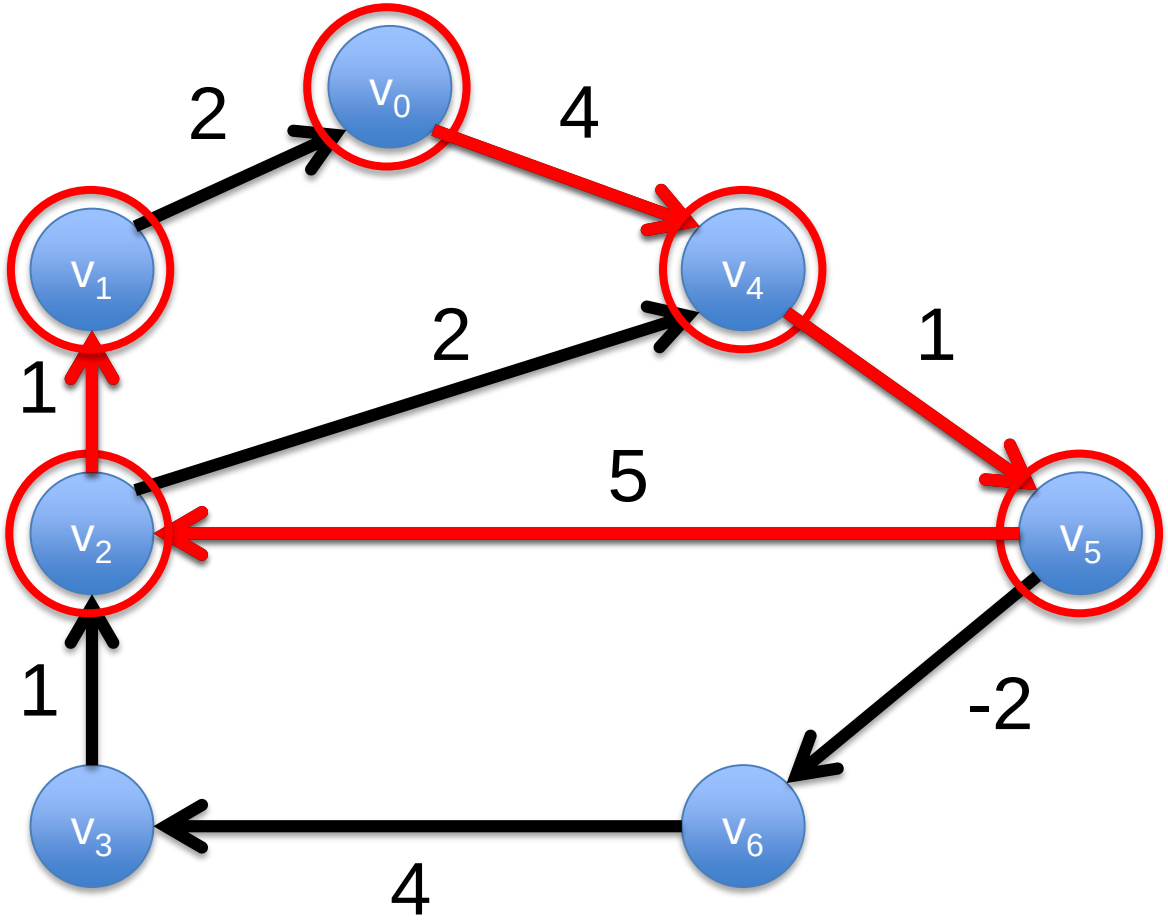


D = (V, A)

Number of arcs: k

Length of above walk = 4

Length of a Walk



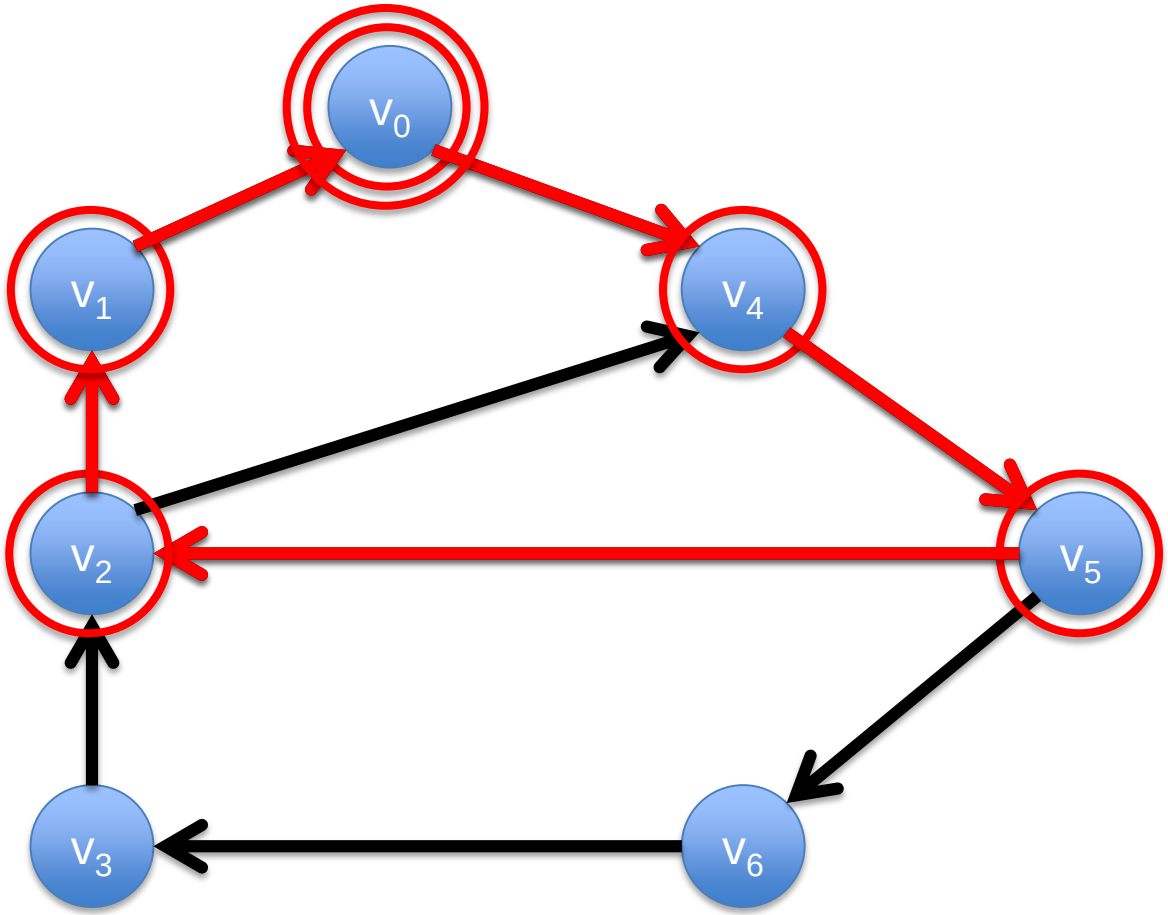
D = (V, A)

l: E ➔ R

Sum of lengths of all edges: $\Sigma_i\ l(e_i)$

Length of above walk = 4+1+5+1 = 11

Closed Walk

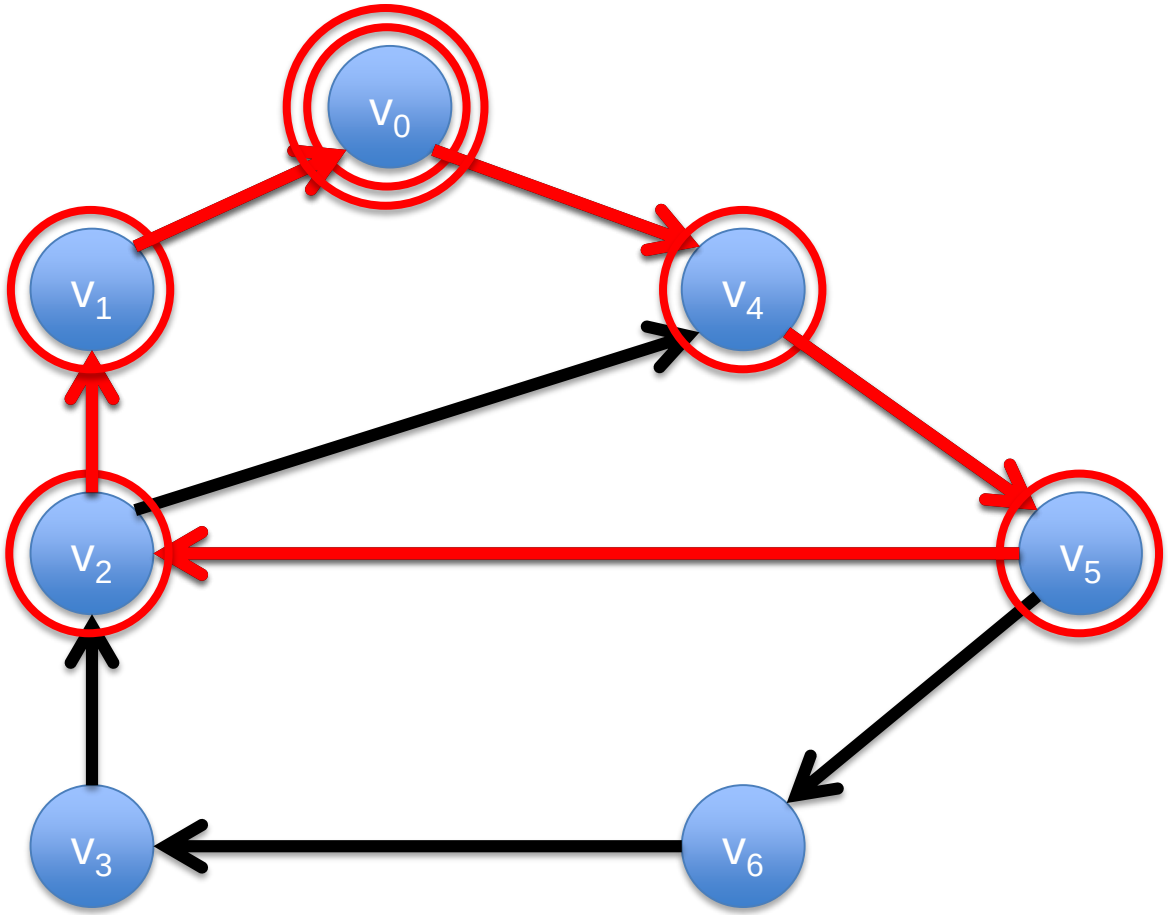


D = (V, A)

Sequence $P = (v_0, a_1, v_1, \ldots, a_k, v_k)$, $a_i = (v_{i-1}, v_i)$

$v_0 = v_k$

Circuit



D = (V, A)

Sequence $P = (v_0,a_1,v_1,\ldots,a_k,v_k)$, $a_i = (v_{i-1},v_i)$

$v_0 = v_k$ Vertices $v_0,v_1,\ldots,v_{k-1}$ are distinct

Outline

- Graph Preliminaries
- **Complexity Preliminaries**
- Shortest Path Algorithms

$$f(n) \in O(g(n))$$

There exists $k > 0$ and n_0 such that

$$f(n) \leq k\, g(n)$$

for all $n \geq n_0$

$f(n) = 5n^2 + 3n\log(n)$

$g(n) = n^2$

✓

$f(n) = 5n^3 + 3n\log(n)$

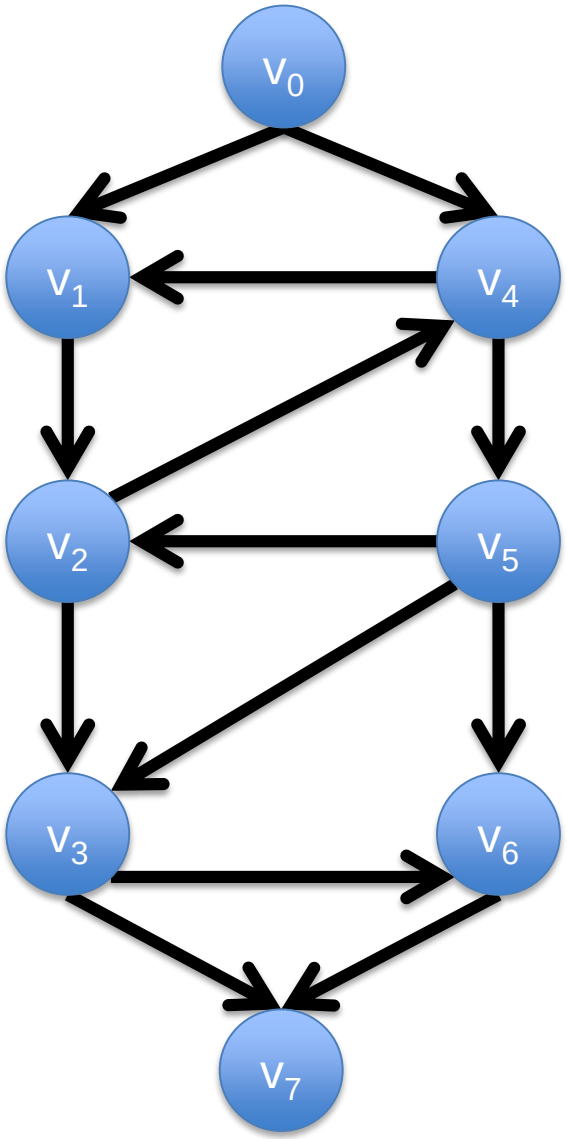
$g(n) = n^2$

✗

Outline

- Graph Preliminaries
- Complexity Preliminaries
- **Shortest Path Algorithms**
  - Breadth-first Search
  - Dijkstra's Method
  - Bellman-Ford Method
  - Floyd-Warshall Method

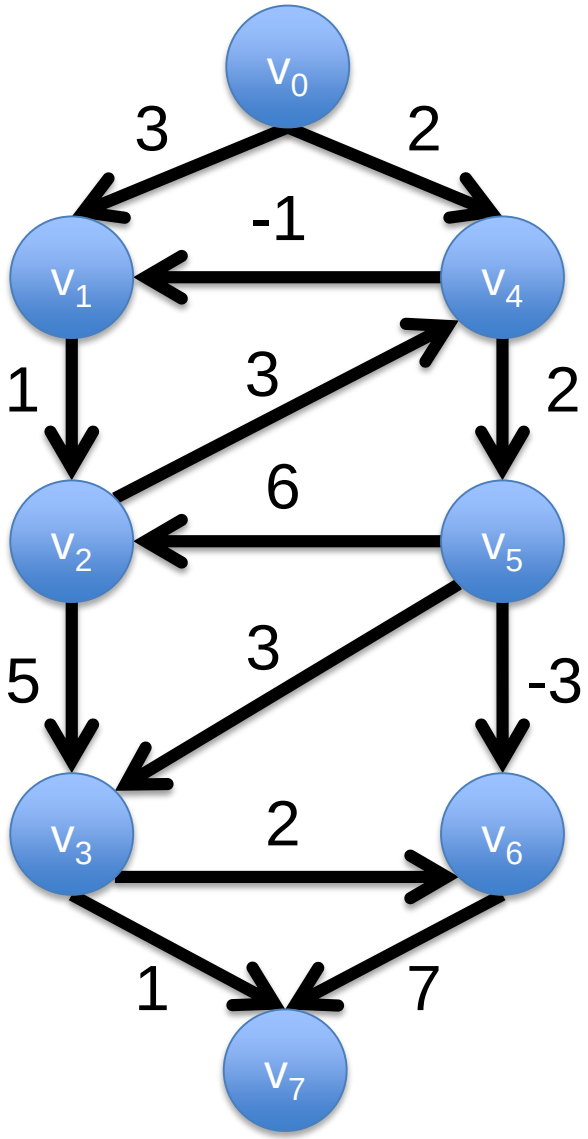
The Shortest Path Problem



Find the shortest path from s to t

Length of path =
Number of arcs

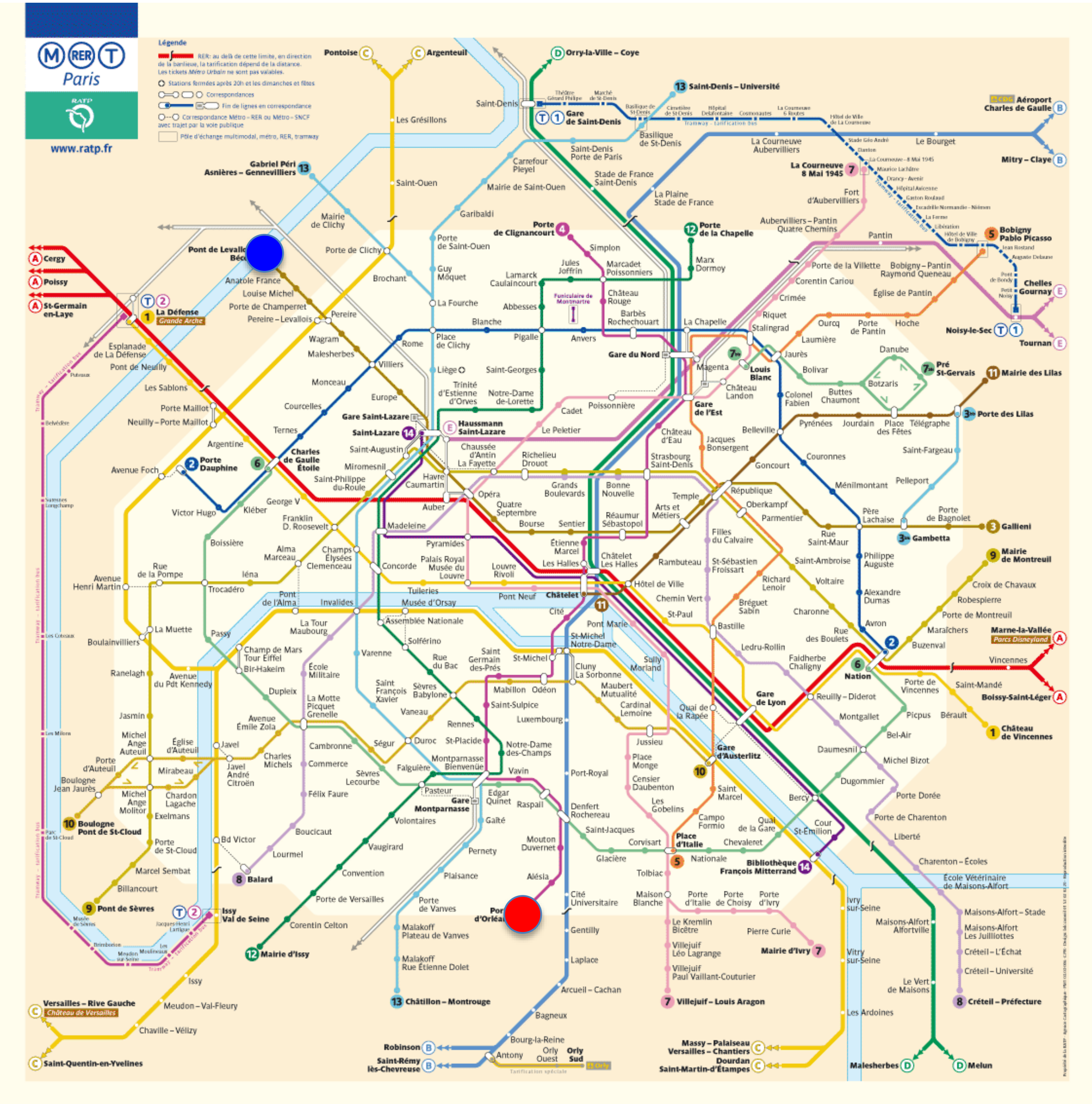
The Shortest Path Problem



Find the shortest path from s to t

Length of path =
Σ Length of arcs

Applications



Minimize number of stops (lengths = 1)

Minimize amount of time (positive lengths)

Applications

Convert 1 ounce of gold to US dollars

1 GOLD = 455.2 * 0.6677 * 1.0752 = 327.28 USD

Take log to convert products to sums (possibly negative lengths)

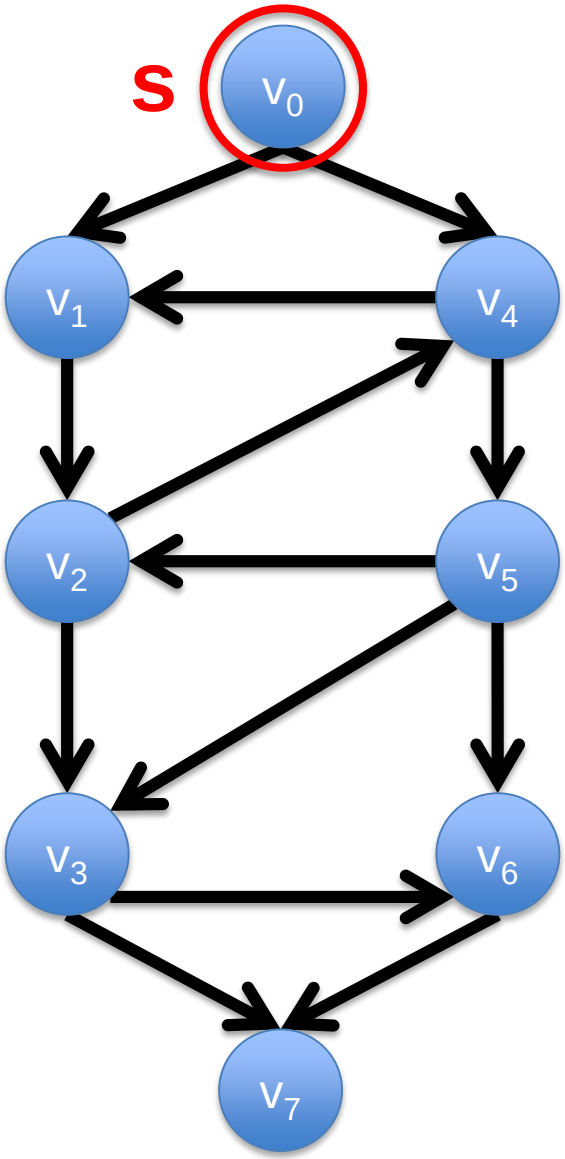
	GBP	EUR	JPY	CHF	USD	GOLD
GBP	1.0	0.6853	0.005290	0.4569	0.6368	208.1
EUR	1.4599	1.0	0.007721	**0.6677**	0.9303	304.028
JPY	189.05	129.52	1.0	85.4694	120.4	39346.7
CHF	2.1904	1.4978	0.011574	1.0	1.3941	**455.2**
USD	1.5714	**1.0752**	0.008309	0.7182	1.0	327.25
GOLD	0.004816	0.003295	0.0000255	0.002201	0.003065	1.0

Example courtesy of Robert Sedgewick

Outline

- Graph Preliminaries
- Complexity Preliminaries
- Shortest Path Algorithms
  - **Breadth-first Search**
  - Dijkstra's Method
  - Bellman-Ford Method
  - Floyd-Warshall Method

Breadth-First Search



D = (V, A)

Assumption: all lengths = 1

Length of path = Number of arcs

|V| = n, |A| = m

Breadth-First Search

Queue is empty. All nodes unvisited.

Push ‘s’ to queue. Set length(s) = 0.

While (‘t’ is unvisited)

Pop u from queue.

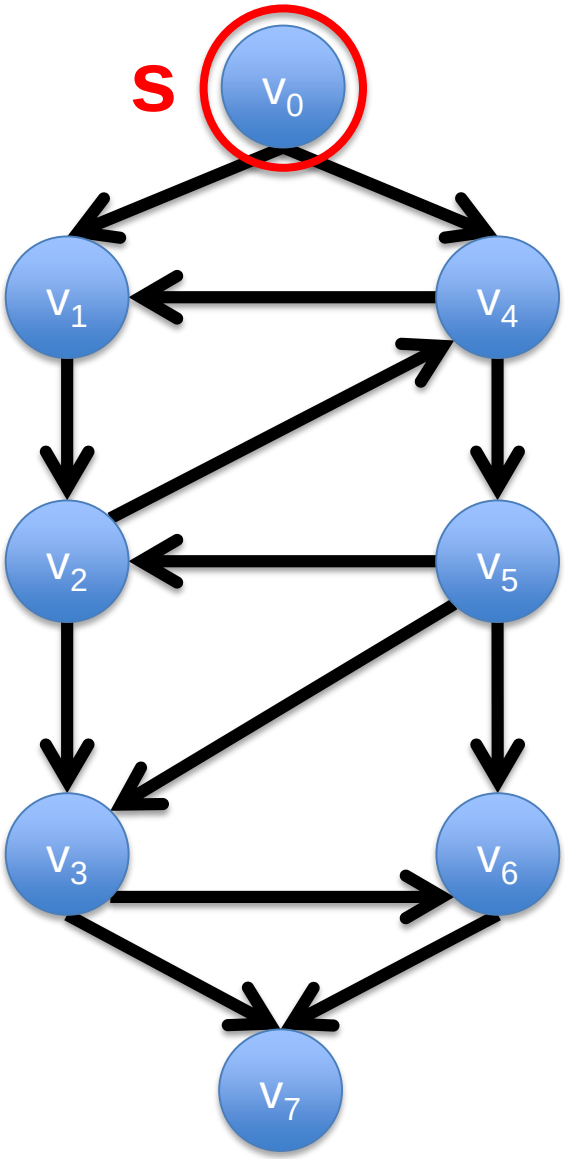
Length(unvisited outneighbors) = Length(u)+1

Push unvisited outneighbors to queue

End

Shortest path to all vertices from ‘s’

Breadth-First Search



Populate the queue with ‘s’

Length of ‘s’ = 0.

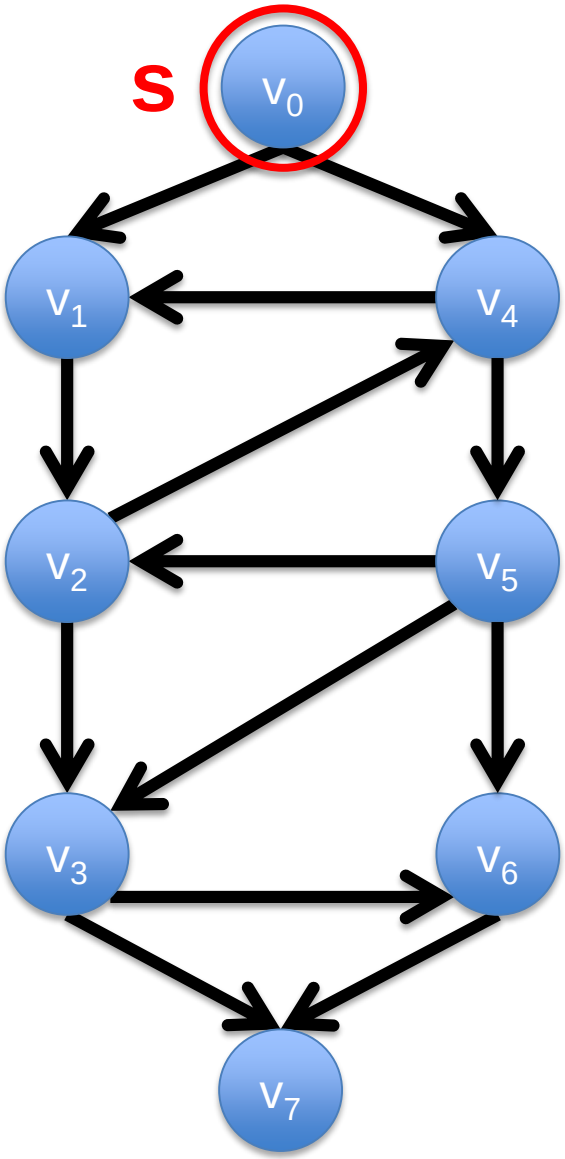
Queue

v_0							

Length

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0							

Breadth-First Search



Pop from the queue. $u = v_0$

Length(unvisited outneighbors) = Length(v) + 1

Push unvisited outneighbors to queue.

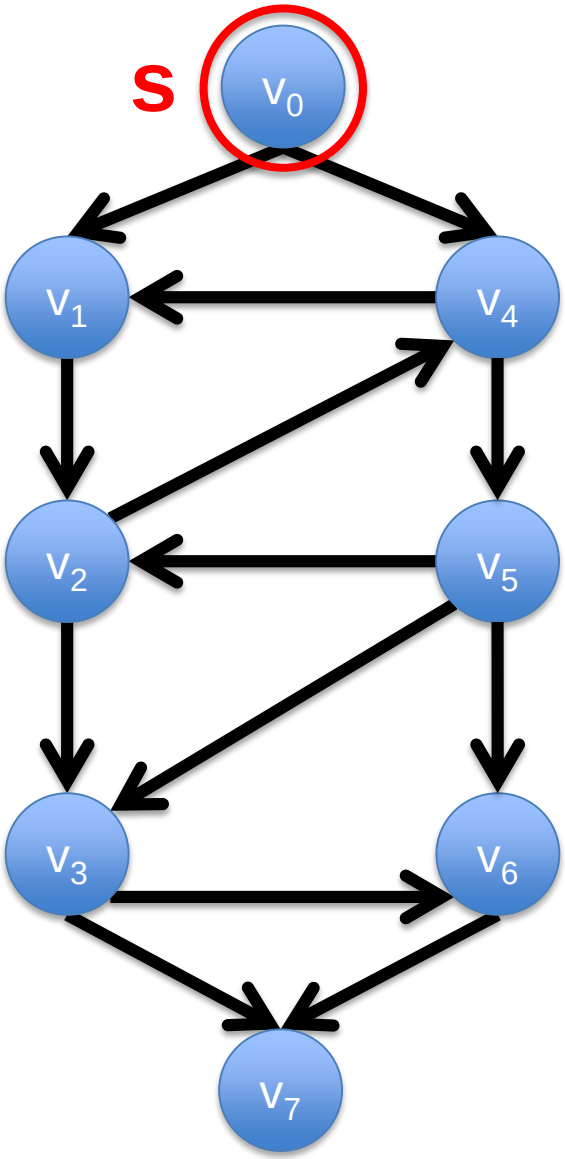
Queue

v_1	v_4						

Length

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	1			1			

Breadth-First Search



Pop from the queue. $u = v_1$

Length(unvisited outneighbors) = Length(v) + 1

Push unvisited outneighbors to queue.

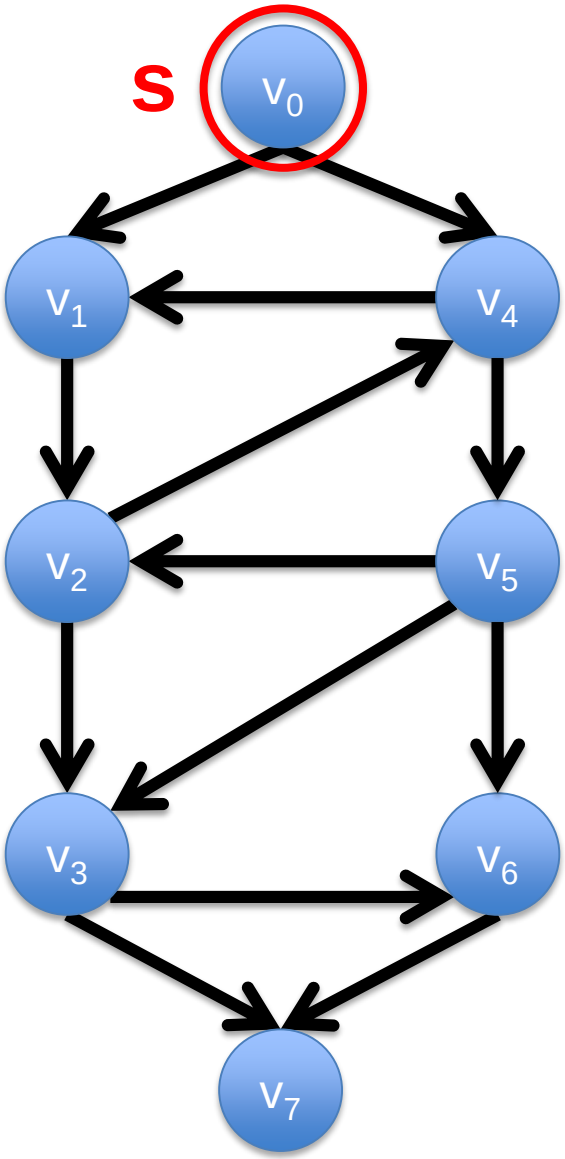
Queue

v_4	v_2						

Length

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	1	2		1			

Breadth-First Search



Pop from the queue. $u = v_4$

Length(unvisited outneighbors) = Length(v) + 1

Push unvisited outneighbors to queue.

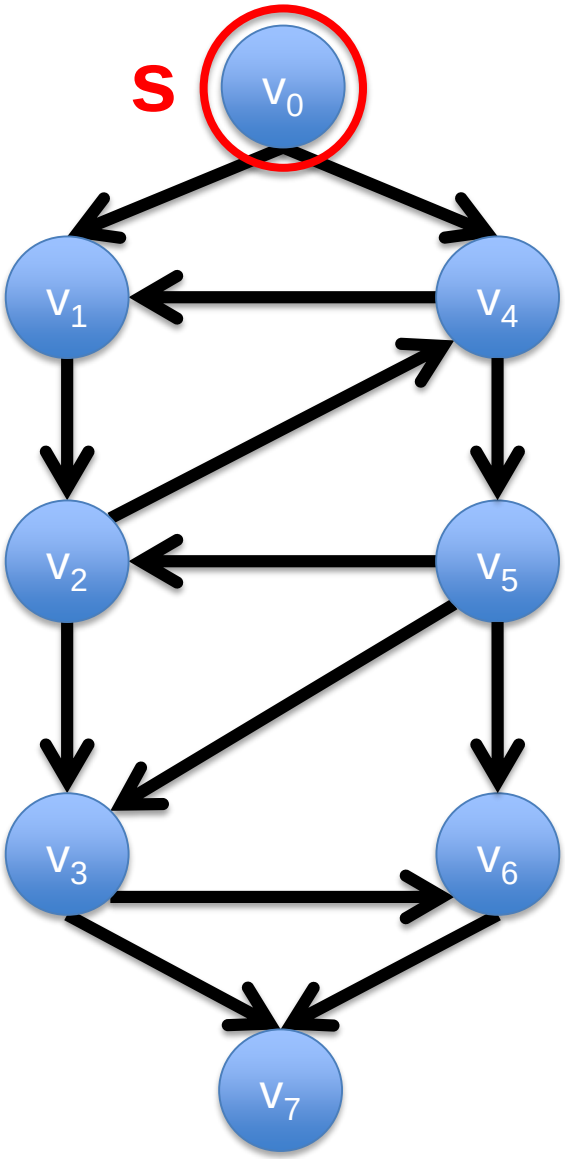
Queue

v_2	v_5						

Length

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	1	2		1	2		

Breadth-First Search



Pop from the queue. $u = v_2$

Length(unvisited outneighbors) = Length(v) + 1

Push unvisited outneighbors to queue.

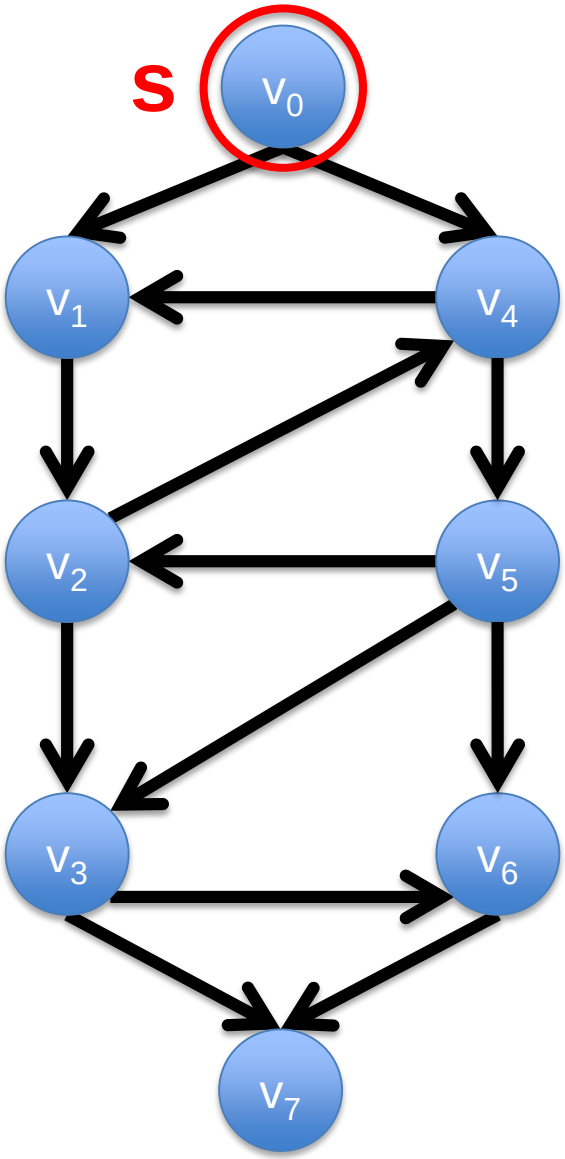
Queue

v_5	v_3						

Length

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	1	2	3	1	2		

Breadth-First Search



Pop from the queue. $u = v_5$

Length(unvisited outneighbors) = Length(v) + 1

Push unvisited outneighbors to queue.

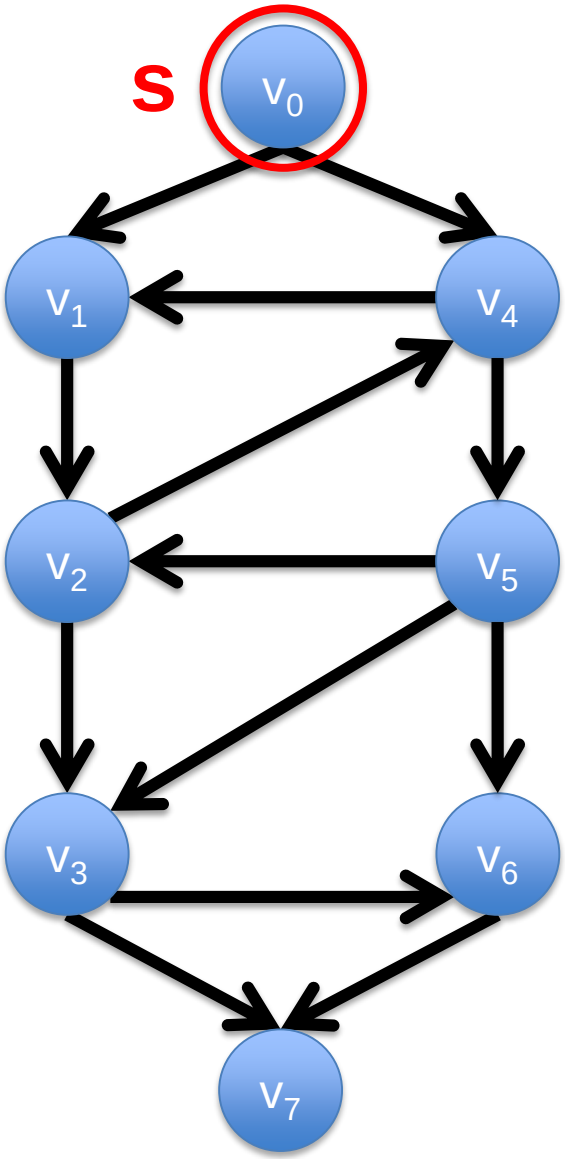
Queue

v_3	v_6						

Length

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	1	2	3	1	2	3	

Breadth-First Search



Pop from the queue. $u = v_3$

Length(unvisited outneighbors) = Length(v) + 1

Push unvisited outneighbors to queue.

Queue

v_6							

Length

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	1	2	3	1	2	3	4

Breadth-First Search

Queue is empty. All nodes unvisited.

Push ‘s’ to queue. Set length(s) = 0.

While (‘t’ is unvisited)

Pop u from queue.

Length(unvisited outneighbors) = Length(u)+1

Push unvisited outneighbors to queue

End

Shortest path to all vertices from ‘s’

Analysis

Time Complexity : O(n + m)

Queue operations
At most ‘n’ elements in the queue

In the worst case, all arcs will be visited once

How fast is O(n + m)?

Complexity of this type is called linear.

Just reading the data from disk is also linear.

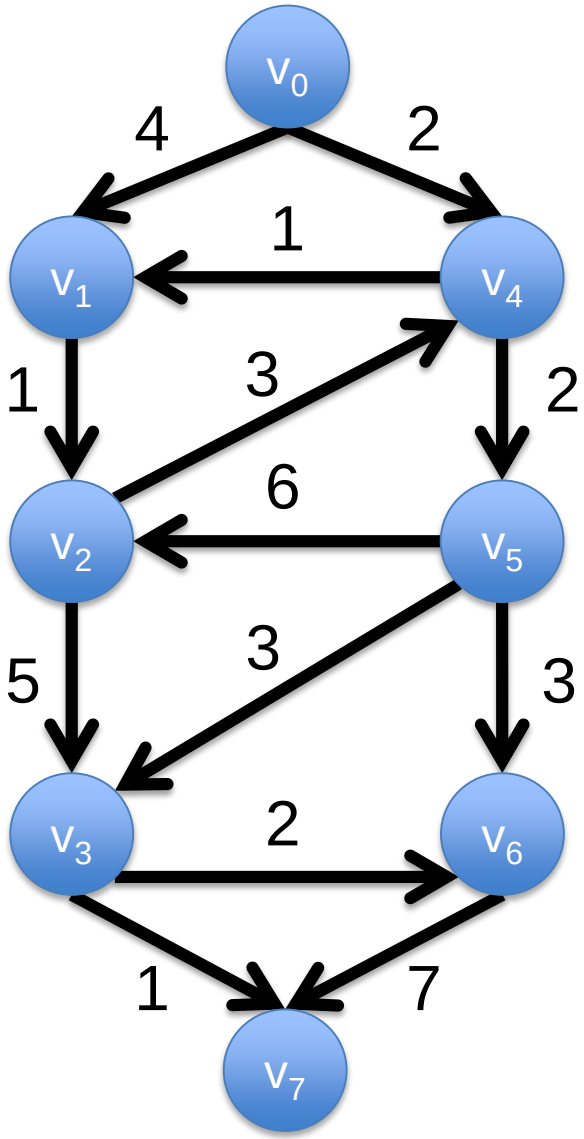
My laptop does 10^7 addition in 2 seconds (python)

Graphs with 10^5 edges are tractable.

Outline

- Graph Preliminaries
- Complexity Preliminaries
- Shortest Path Algorithms
  - Breadth-first Search
  - **Dijkstra's Method**
  - Bellman-Ford Method
  - Floyd-Warshall Method

Dijkstra's Method



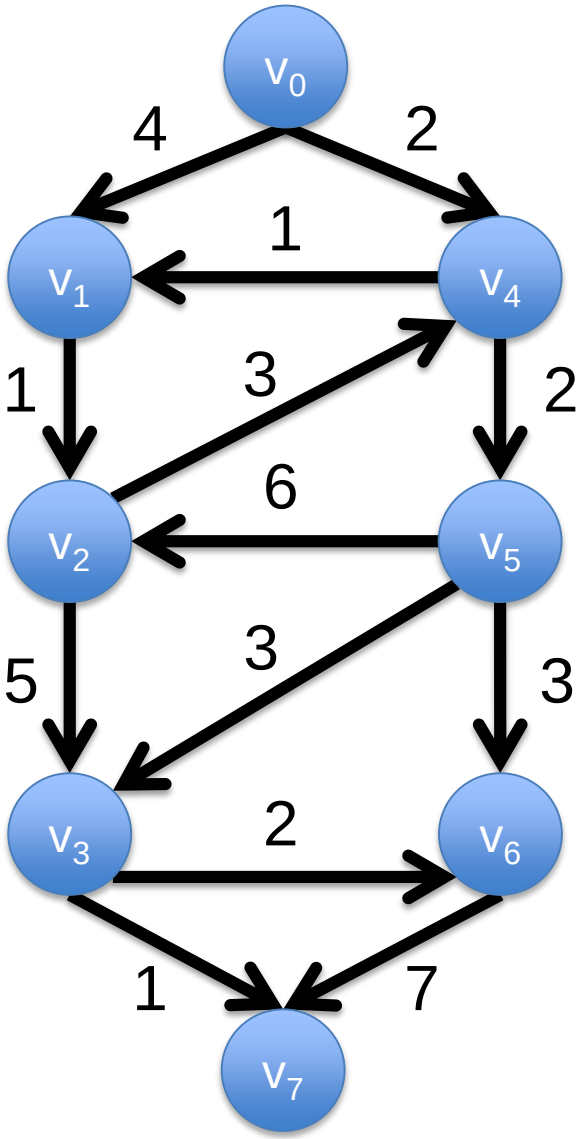
D = (V, A)

Assumption: No negative arc lengths

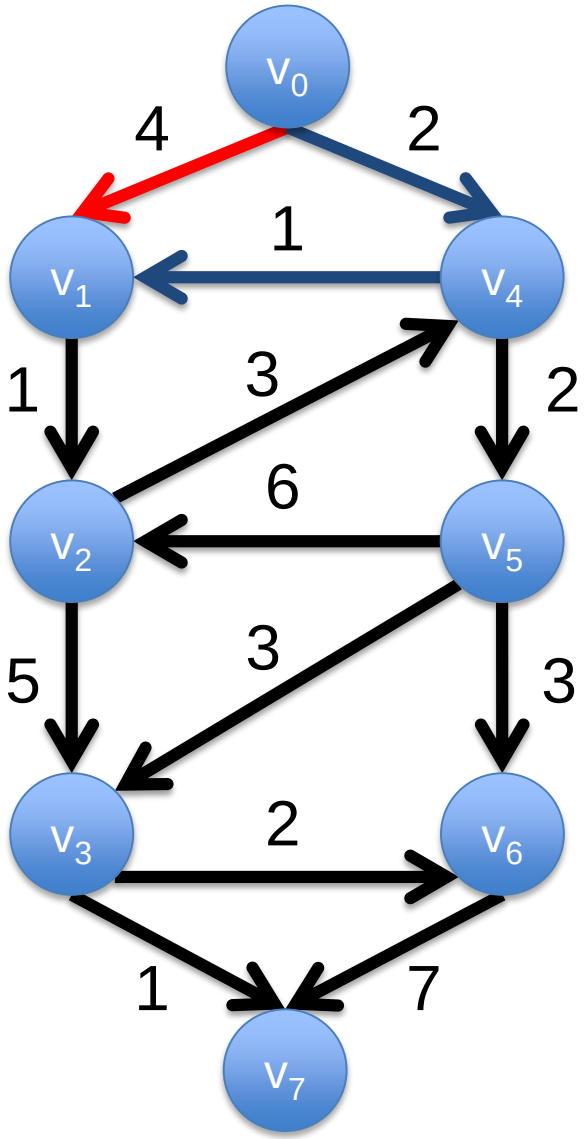
Length of path =
Σ arc lengths = Σ l(u,v)

|V| = n, |A| = m

Breadth-First Search Fails



Breadth-First Search Fails



Counter-example !!

Blue with shorter than red.

Dijkstra's Method

Set $d(v) = \infty$ for all $v \in V$. Set $U = V$.

Set $d(s) = 0$. Set $u = s$.

While u is not equal to t

Choose $u = \text{argmin}_{v \in U} d(v)$

For each v such that $(u,v) \in A$

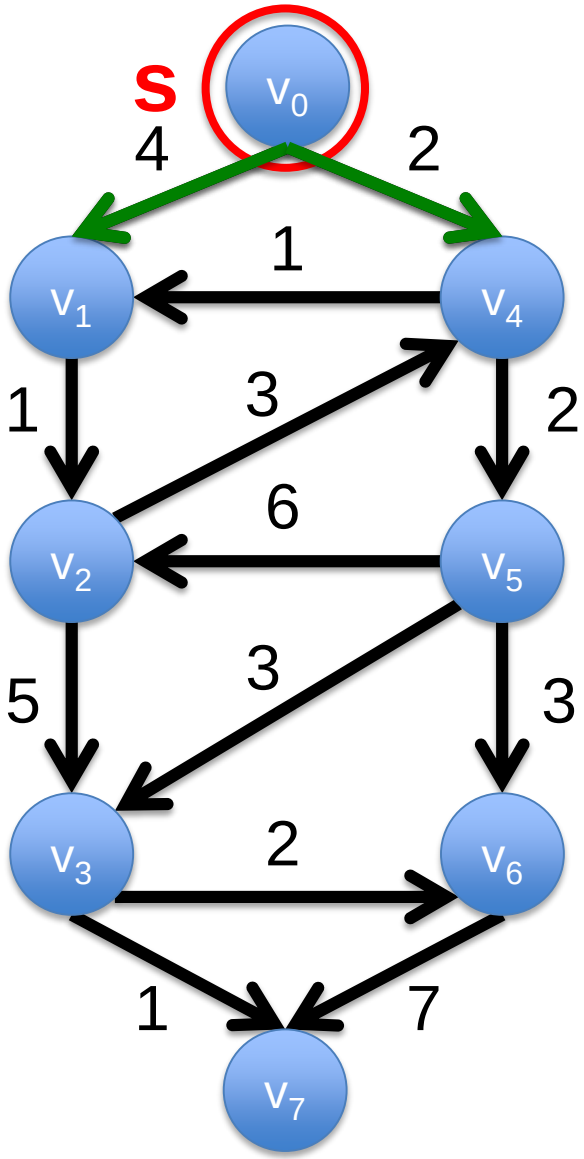
$d(v) = \min \{ d(v), d(u) + l(u,v)\}$

End

Set $U = U \setminus \{u\}$

End

Dijkstra's Method



Set U = V; Set d(s) = 0

Choose u = $\text{argmin}_{v \in U}$ d(v)

For each v such that (u,v) $\in$ A

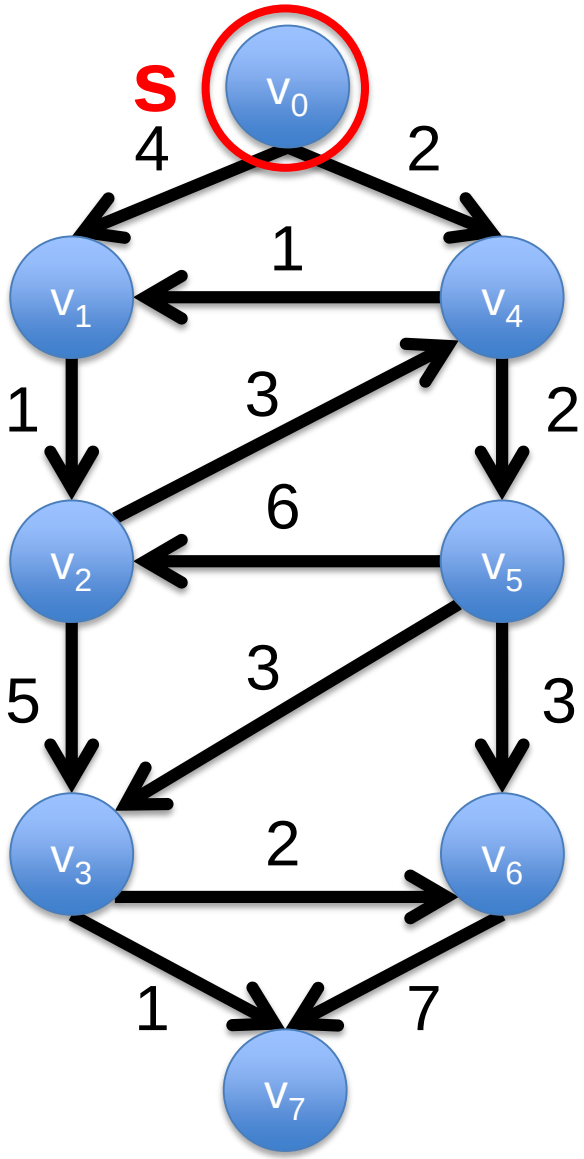
d(v) = min { d(v), d(u) + l(u,v)}

End

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	∞	∞	∞	∞	∞	∞	∞

Dijkstra's Method



Set U = V

Choose u = $\text{argmin}_{v \in U}$ d(v)

For each v such that (u,v) $\in$ A

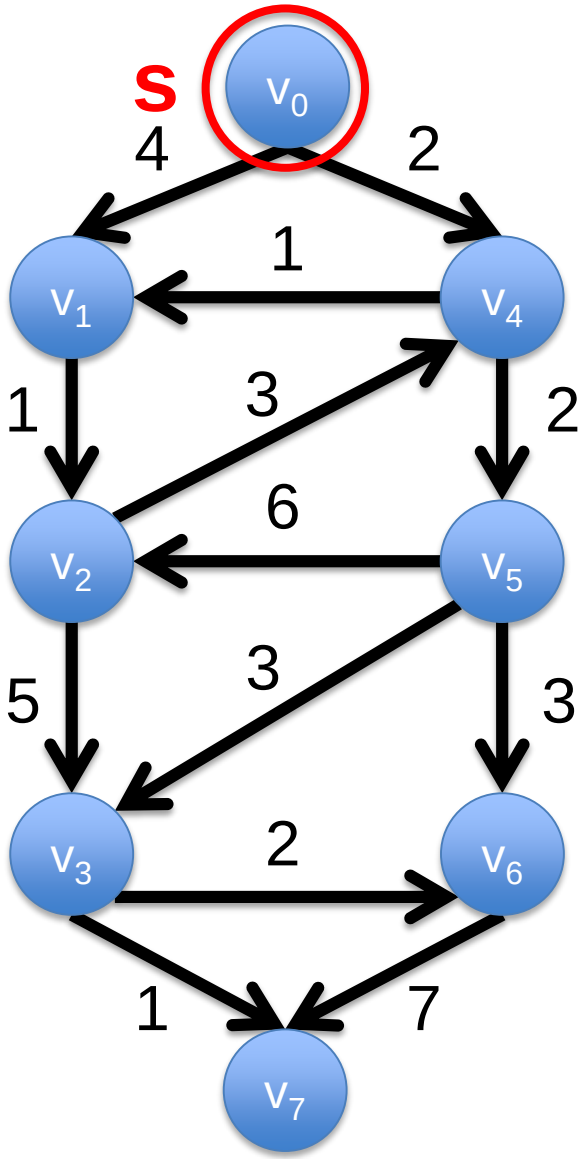
d(v) = min { d(v), d(u) + l(u,v)}

End

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	4	∞	∞	2	∞	∞	∞

Dijkstra's Method

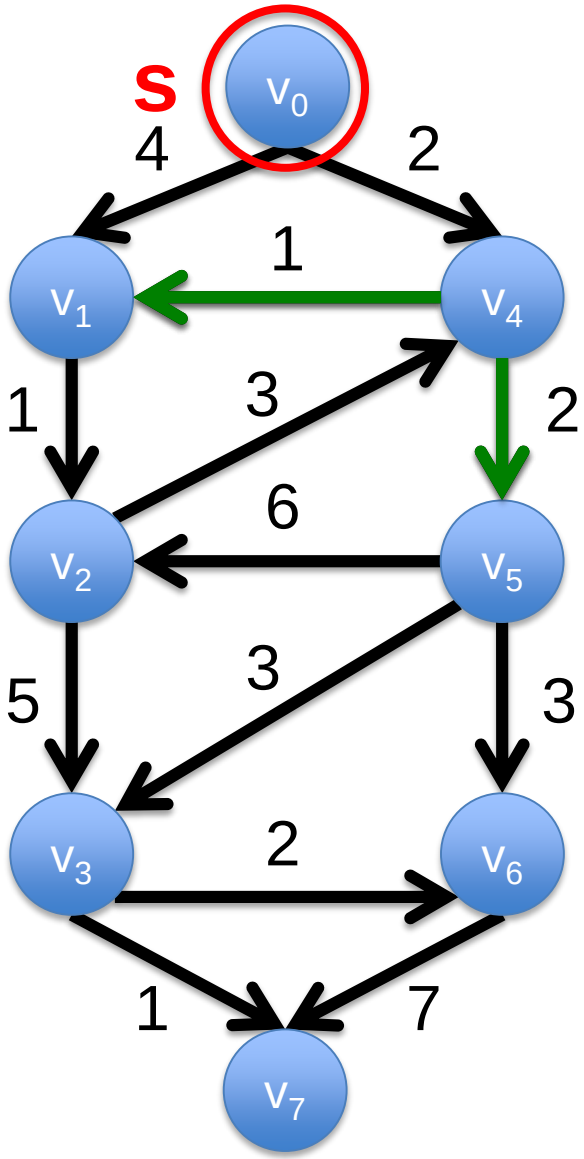


Set U = U \ {u}

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	4	∞	∞	2	∞	∞	∞

Dijkstra's Method



Set U = U \ {u}

Choose u = $\text{argmin}_{v \in U}$ d(v)

For each v such that (u,v) $\in$ A

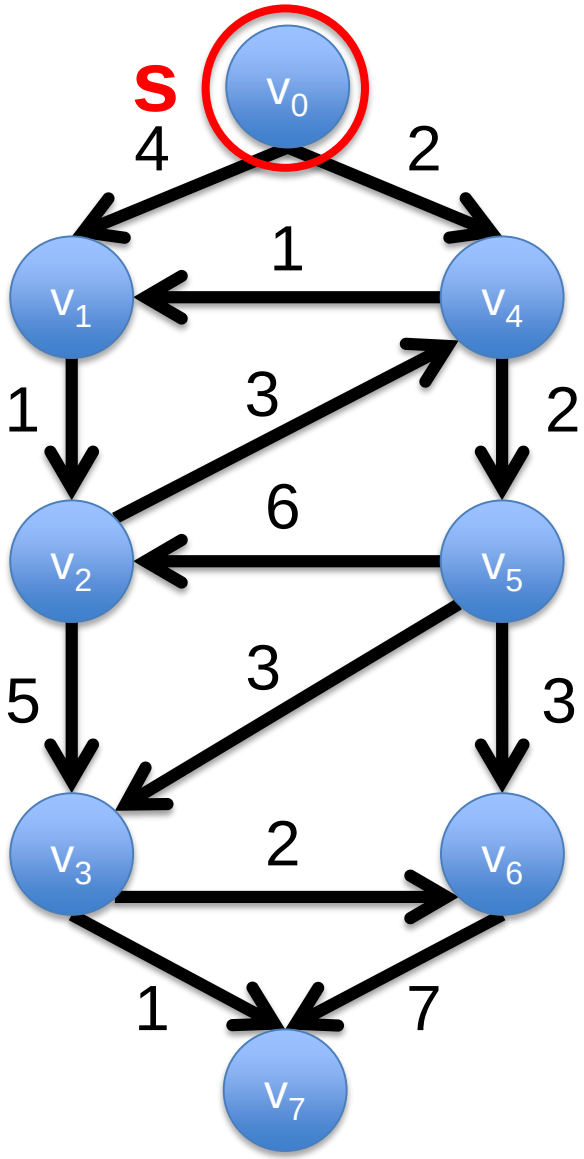
d(v) = min { d(v), d(u) + l(u,v)}

End

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	4	∞	∞	2	∞	∞	∞

Dijkstra's Method



Set U = U \ {u}

Choose u = $\text{argmin}_{v \in U}$ d(v)

For each v such that (u,v) $\in$ A

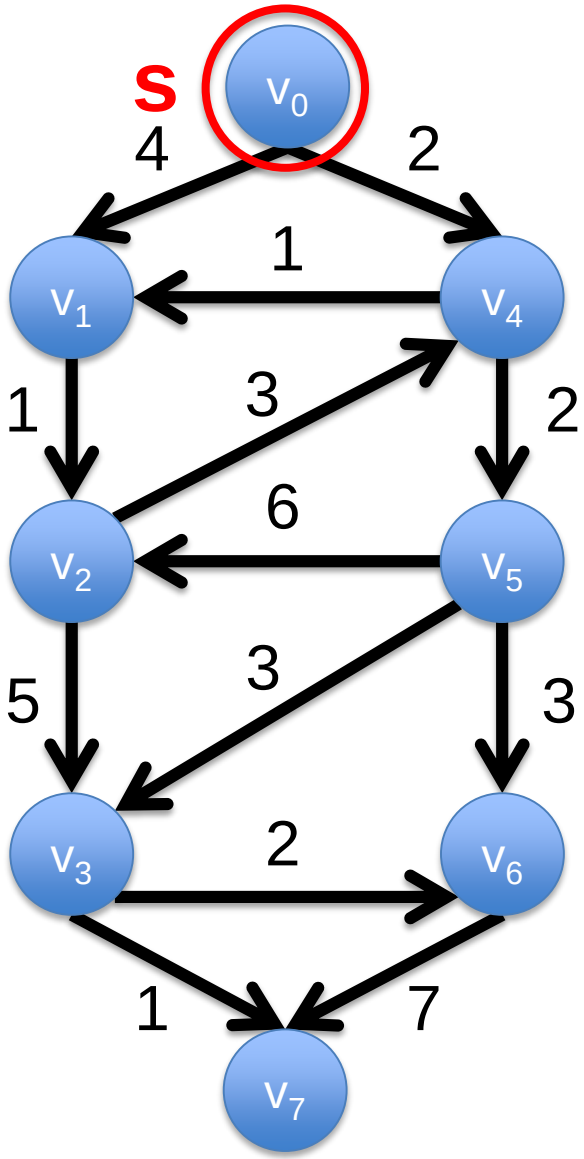
d(v) = min { d(v), d(u) + l(u,v)}

End

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	4	∞	∞

Dijkstra's Method

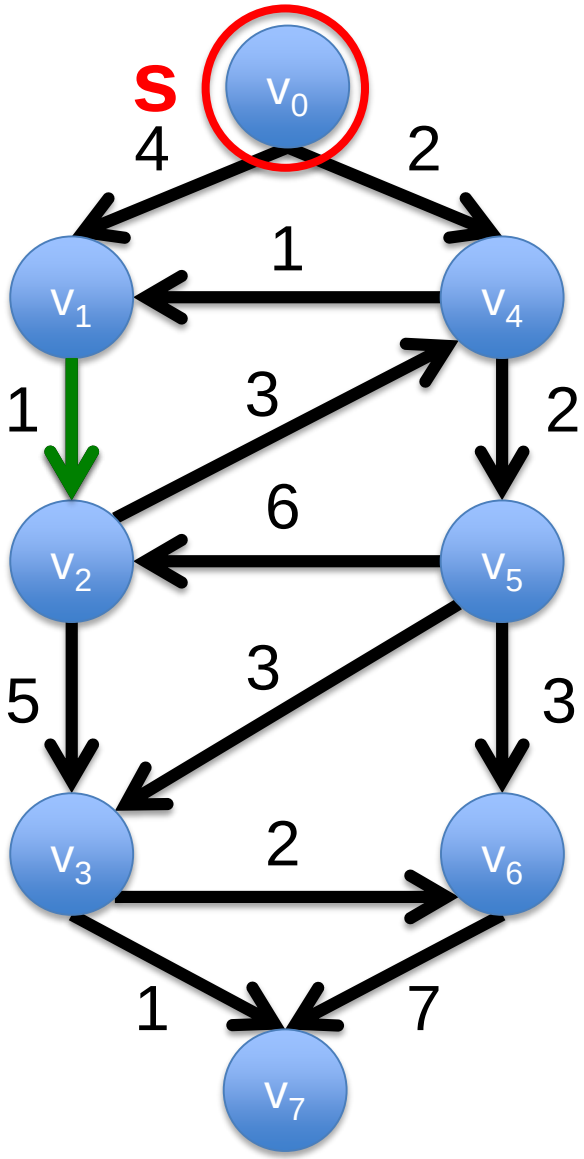


Set U = U \ {u}

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	4	∞	∞

Dijkstra's Method



Set U = U \ {u}

Choose u = $\text{argmin}_{v \in U}$ d(v)

For each v such that (u,v) $\in$ A

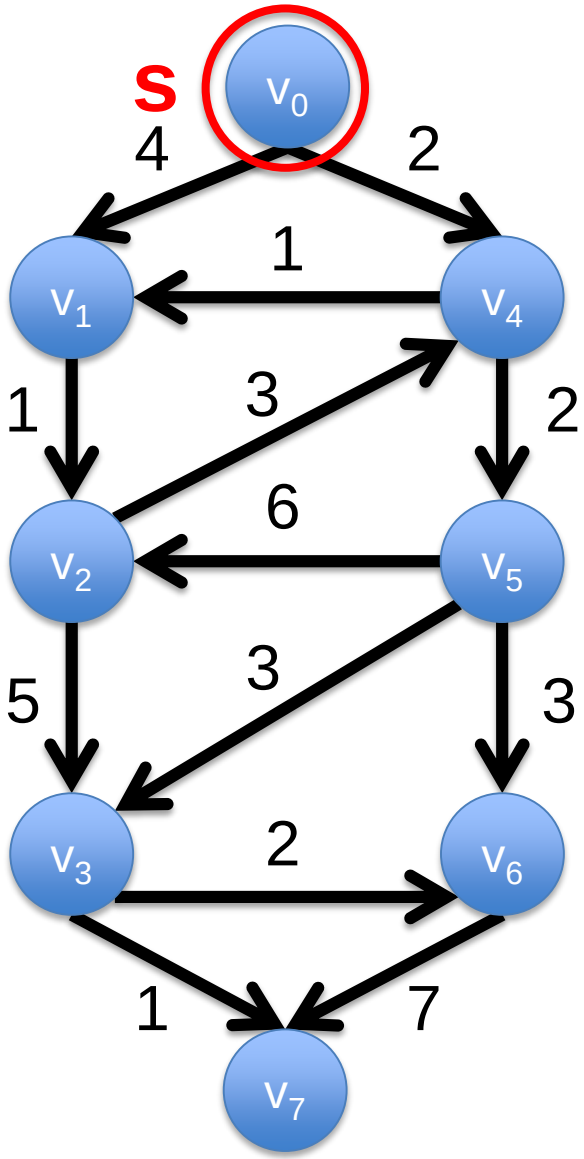
d(v) = min { d(v), d(u) + l(u,v)}

End

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	4	∞	∞

Dijkstra's Method



Set U = U \ {u}

Choose u = $\text{argmin}_{v \in U}$ d(v)

For each v such that (u,v) $\in$ A

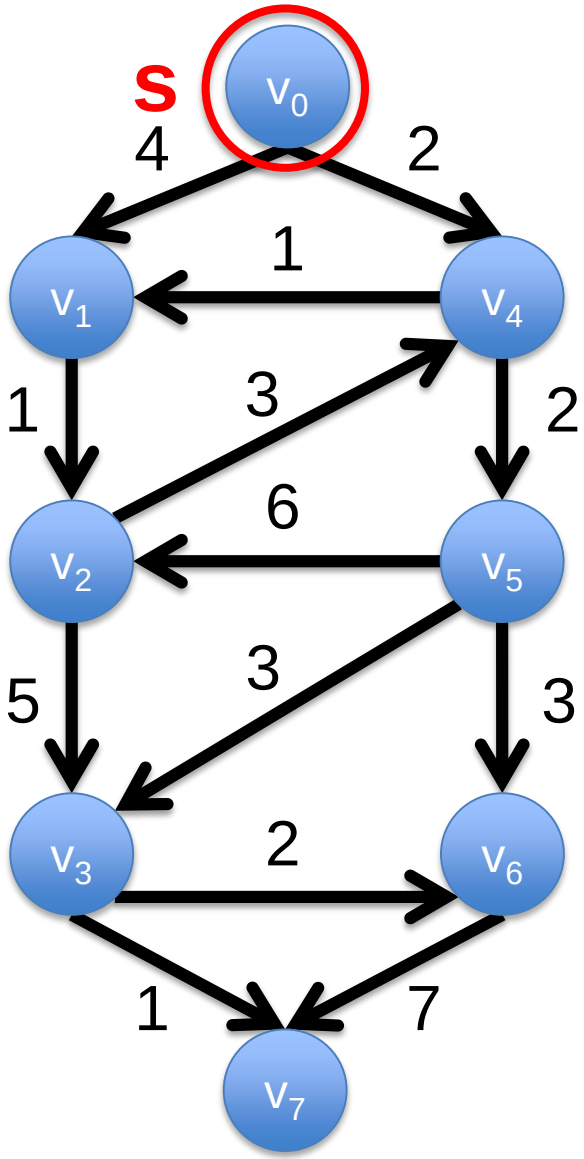
d(v) = min { d(v), d(u) + l(u,v)}

End

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	2	4	∞	∞

Dijkstra's Method

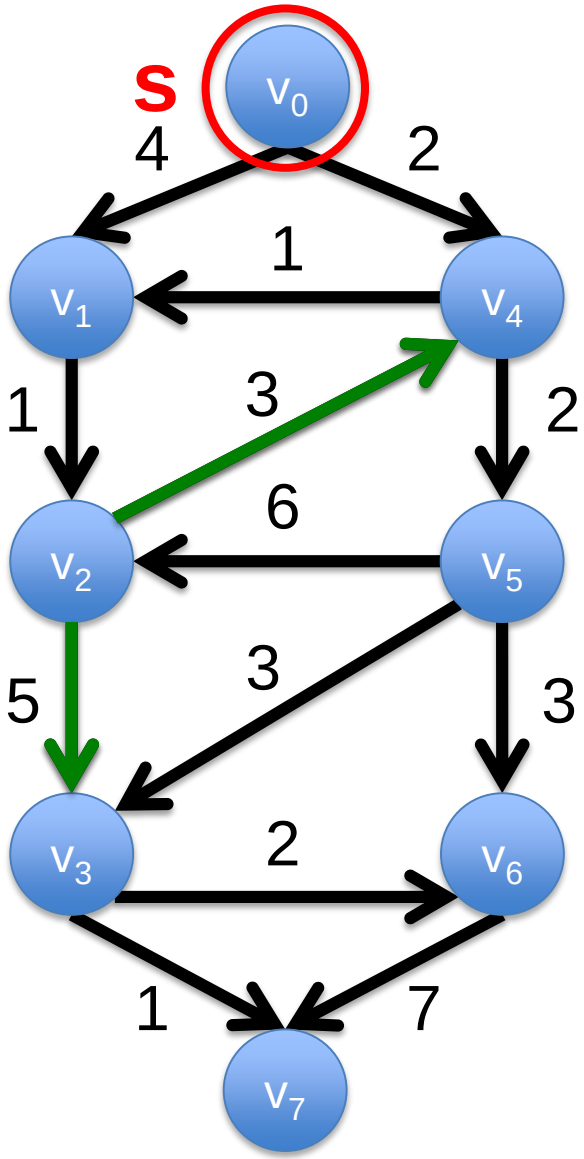


Set U = U \ {u}

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	2	4	∞	∞

Dijkstra's Method



Set U = U \ {u}

Choose u = $\text{argmin}_{v \in U}$ d(v)

For each v such that (u,v) $\in$ A

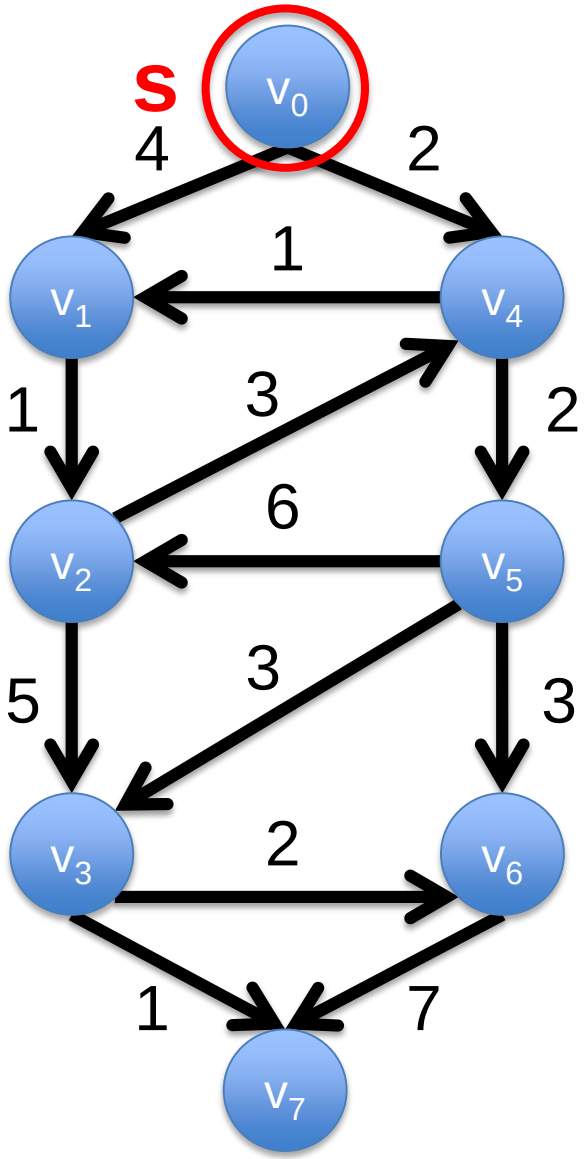
d(v) = min { d(v), d(u) + l(u,v)}

End

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	2	4	∞	∞

Dijkstra's Method



Set U = U \ {u}

Choose u = $\text{argmin}_{v \in U}$ d(v)

For each v such that (u,v) $\in$ A

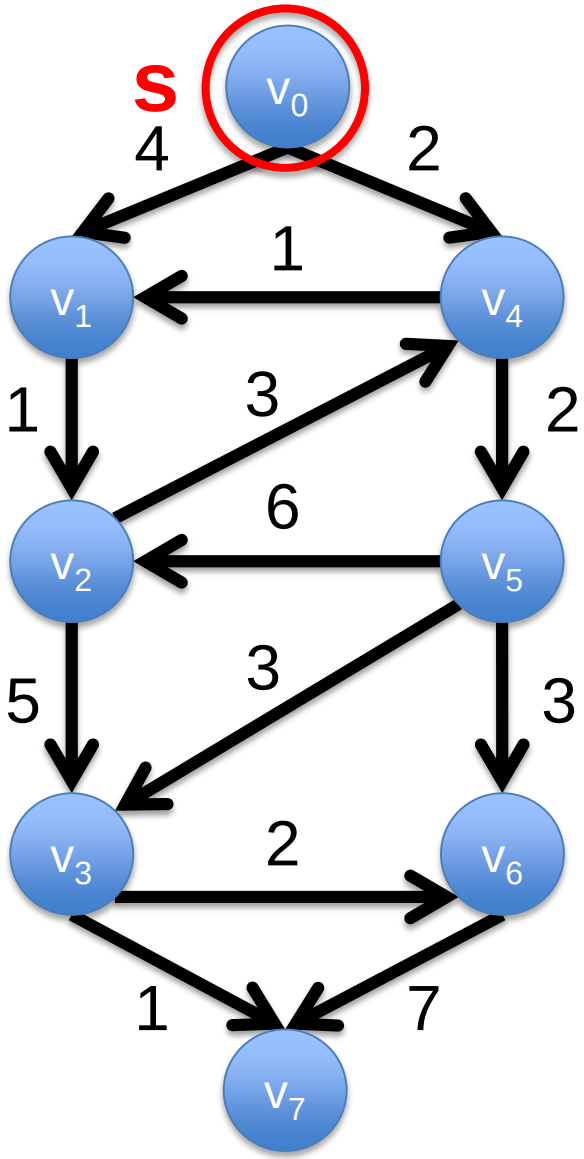
d(v) = min { d(v), d(u) + l(u,v)}

End

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	9	2	4	∞	∞

Dijkstra's Method

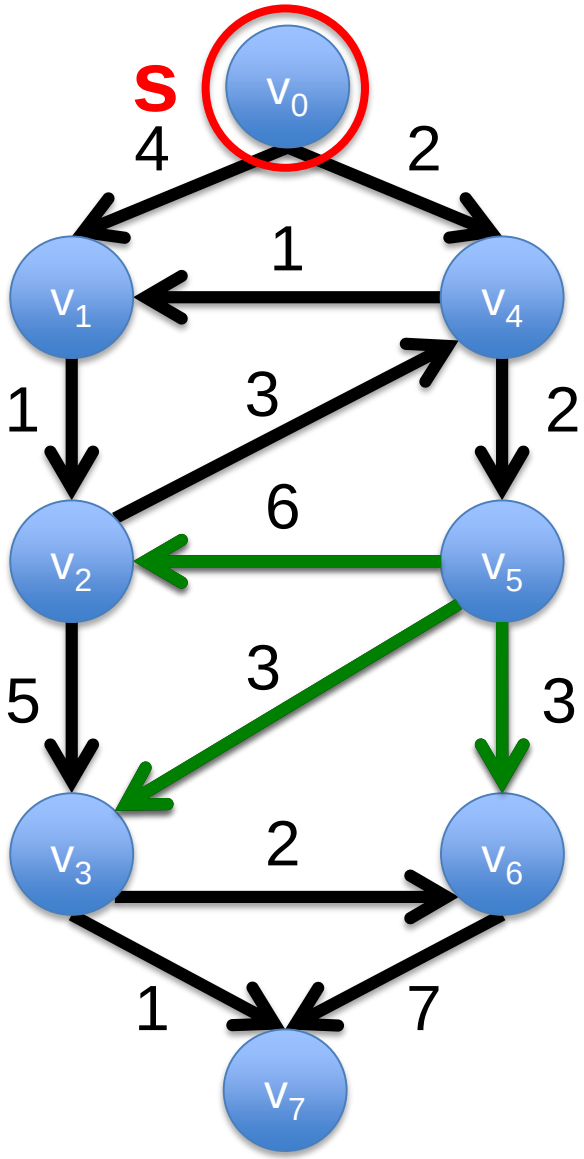


Set U = U \ {u}

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	9	2	4	∞	∞

Dijkstra's Method



Set U = U \ {u}

Choose u = $\text{argmin}_{v \in U}$ d(v)

For each v such that (u,v) $\in$ A

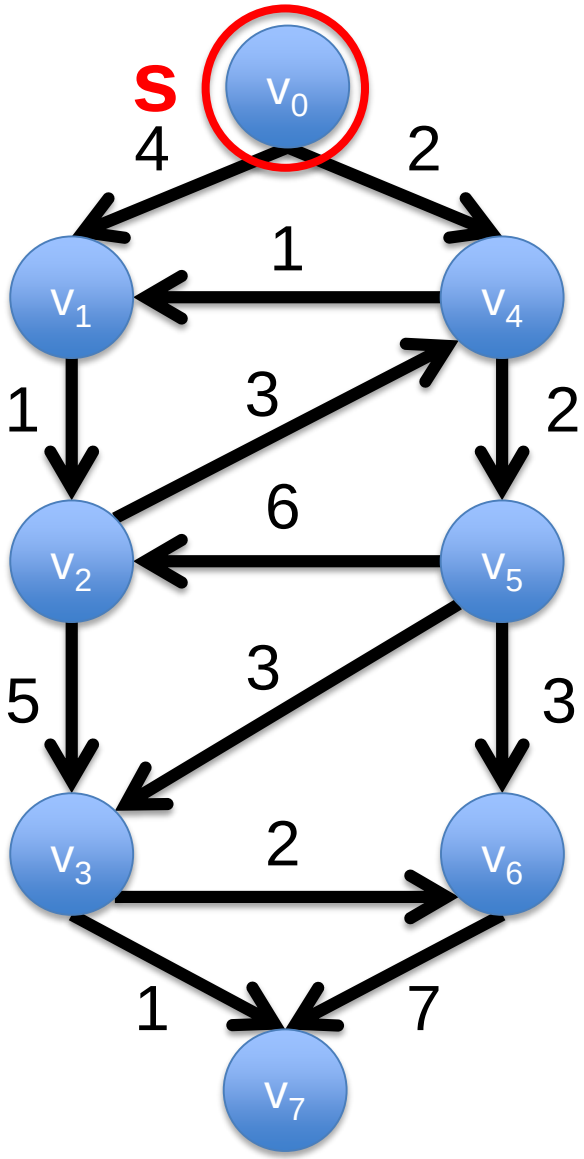
d(v) = min { d(v), d(u) + l(u,v)}

End

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	9	2	4	∞	∞

Dijkstra's Method



Set U = U \ {u}

Choose u = $\text{argmin}_{v \in U}$ d(v)

For each v such that (u,v) $\in$ A

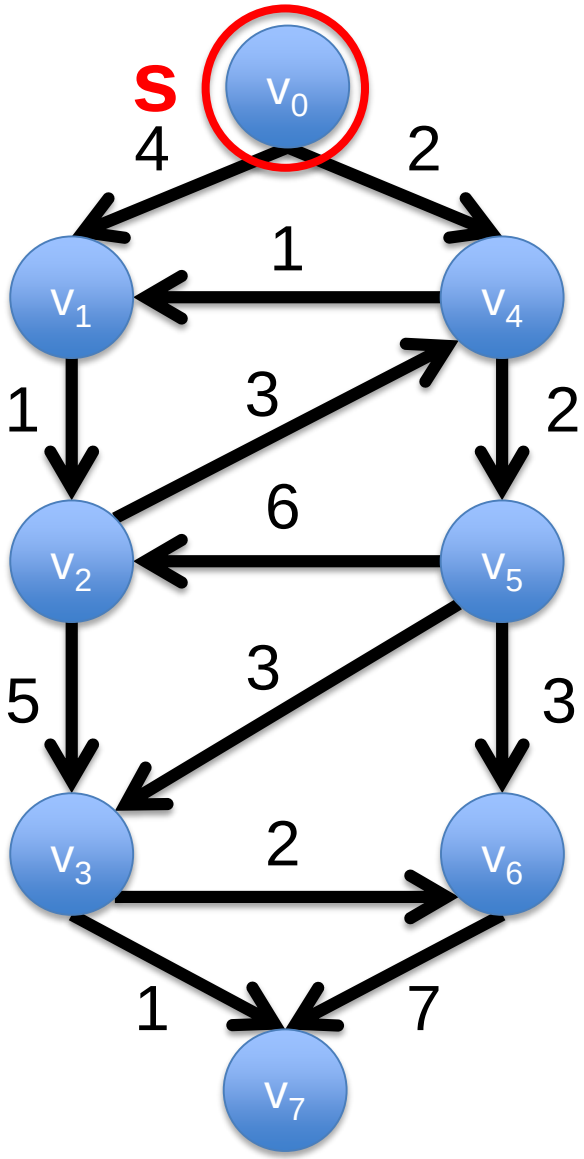
d(v) = min { d(v), d(u) + l(u,v)}

End

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	2	4	7	∞

Dijkstra's Method

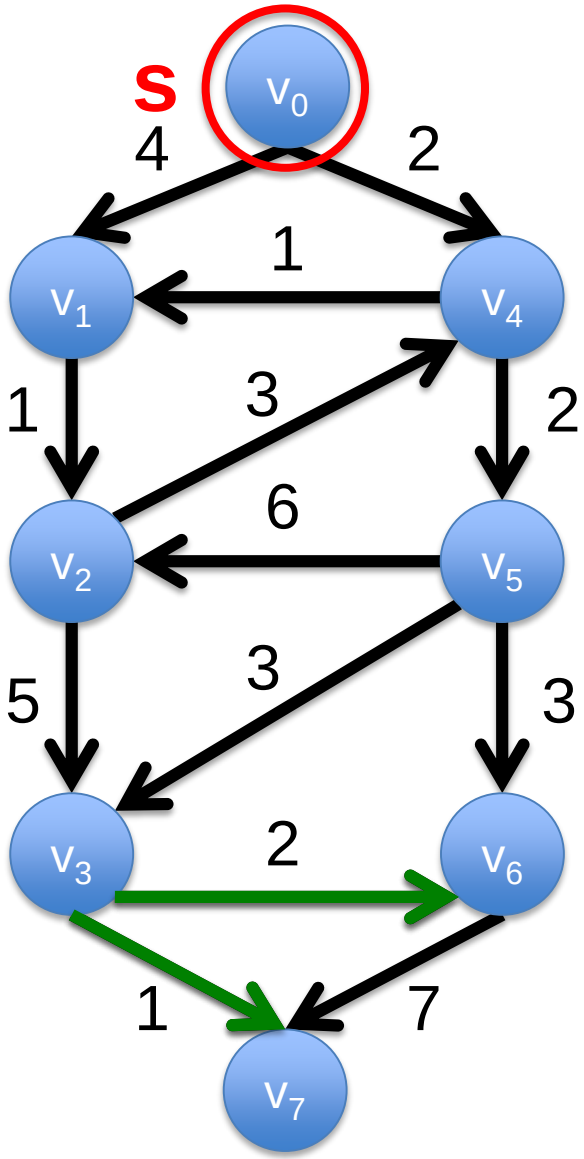


Set U = U \ {u}

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	2	4	7	∞

Dijkstra's Method



Set U = U \ {u}

Choose u = $\text{argmin}_{v \in U}$ d(v)

For each v such that (u,v) $\in$ A

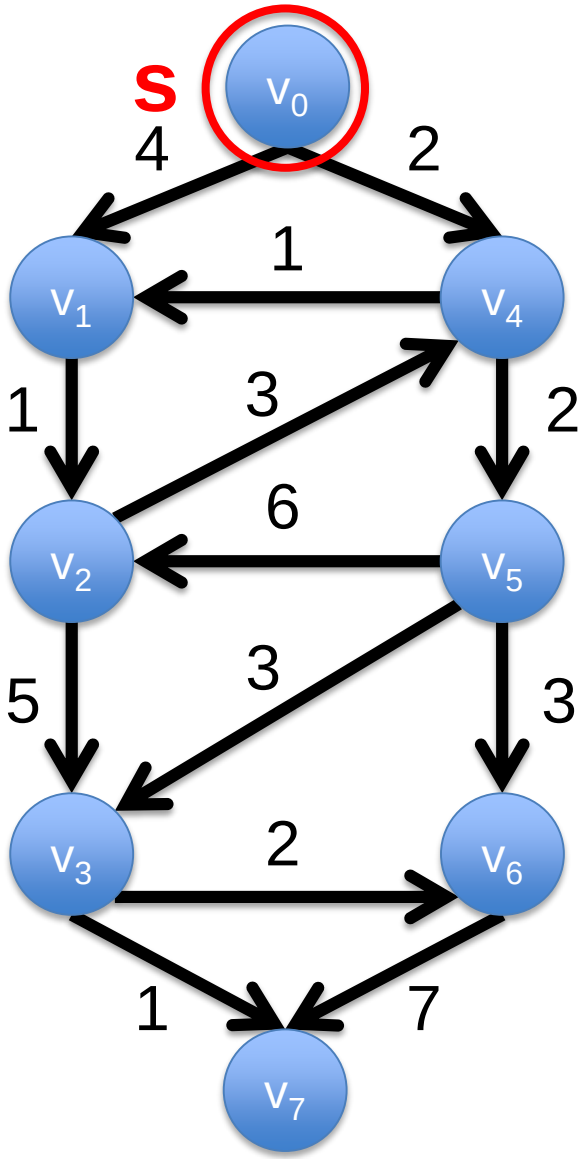
d(v) = min { d(v), d(u) + l(u,v)}

End

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	2	4	7	∞

Dijkstra's Method



Set U = U \ {u}

Choose u = $\text{argmin}_{v \in U}$ d(v)

For each v such that (u,v) $\in$ A

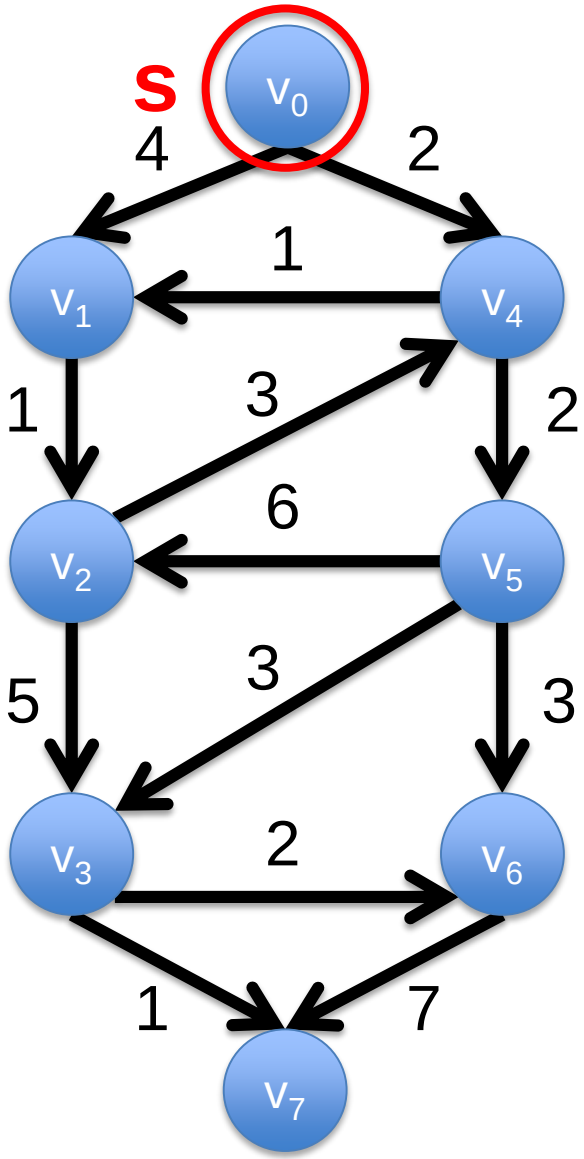
d(v) = min { d(v), d(u) + l(u,v)}

End

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	2	4	7	8

Dijkstra's Method

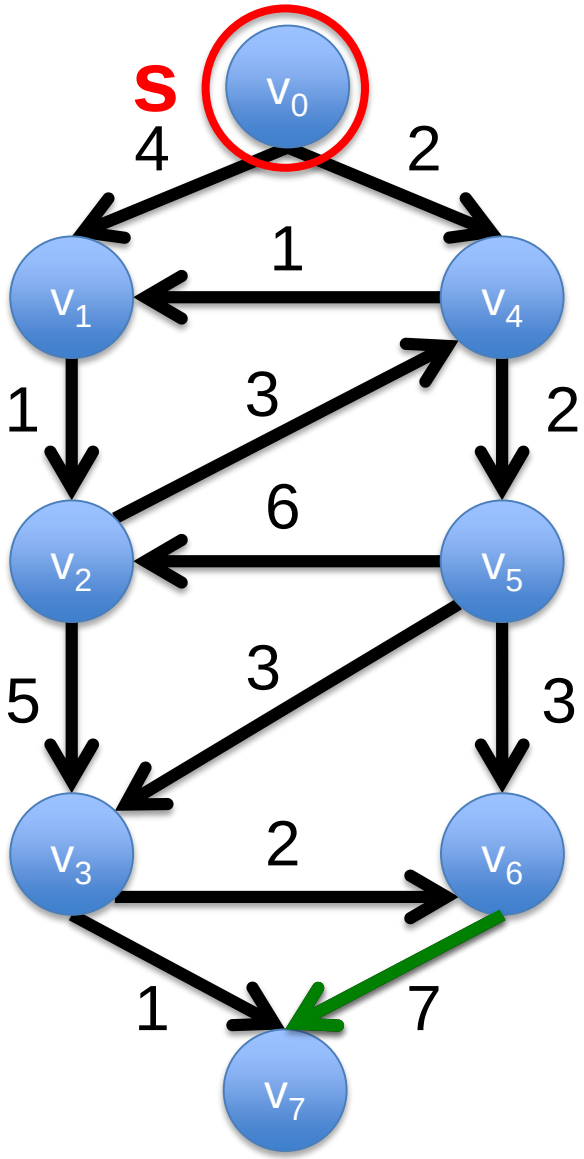


Set U = U \ {u}

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	2	4	7	8

Dijkstra's Method



Set U = U \ {u}

Choose u = $\text{argmin}_{v \in U}$ d(v)

For each v such that (u,v) $\in$ A

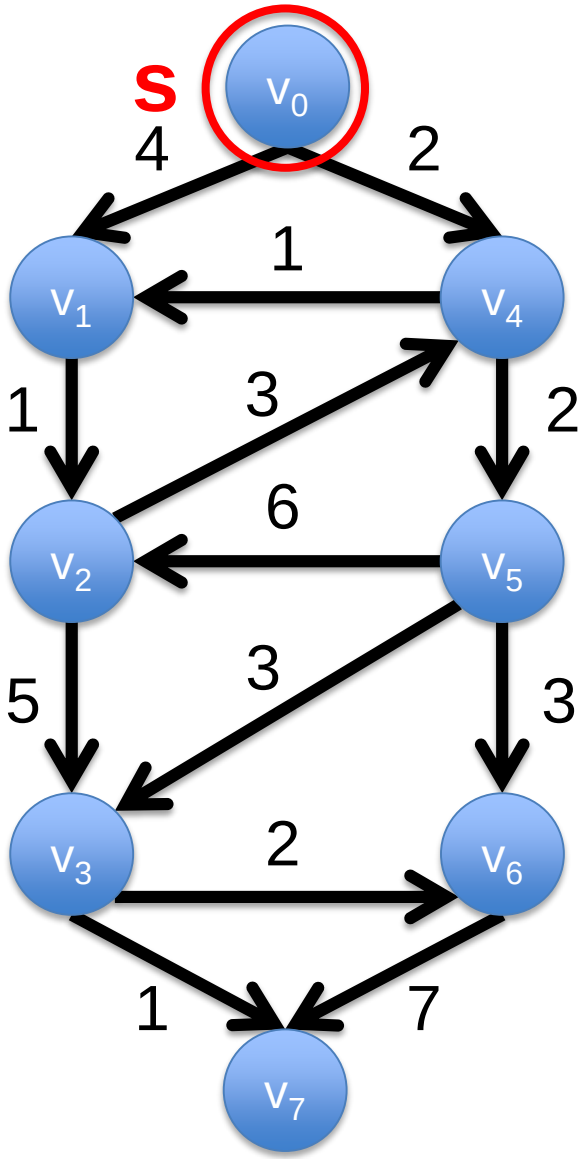
d(v) = min { d(v), d(u) + l(u,v)}

End

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	2	4	7	8

Dijkstra's Method



Set U = U \ {u}

Choose u = $\text{argmin}_{v \in U}$ d(v)

For each v such that (u,v) $\in$ A

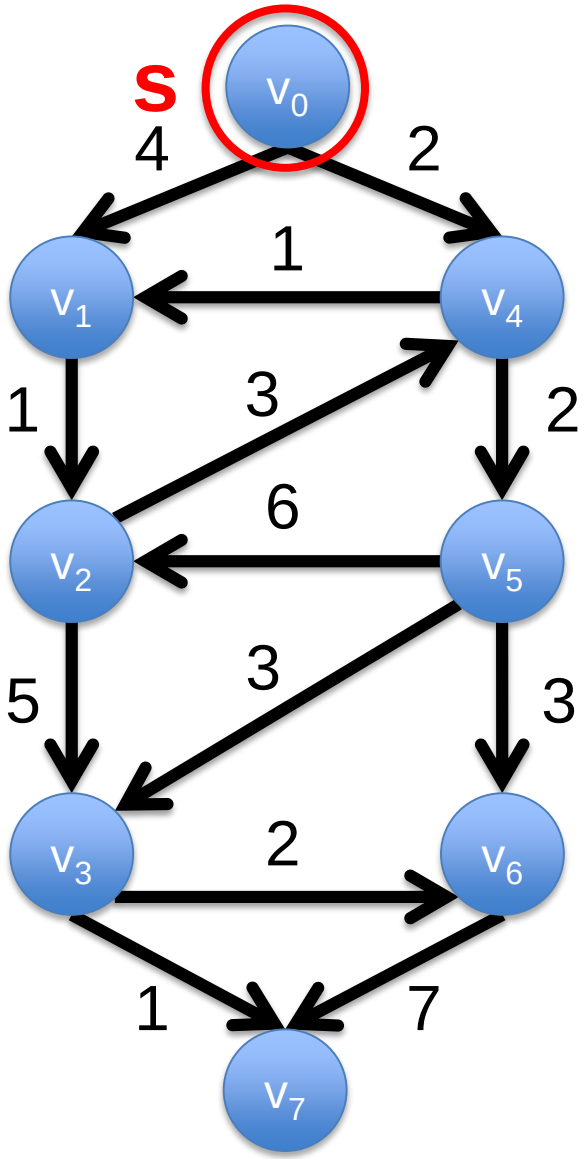
d(v) = min { d(v), d(u) + l(u,v)}

End

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	2	4	7	8

Dijkstra's Method

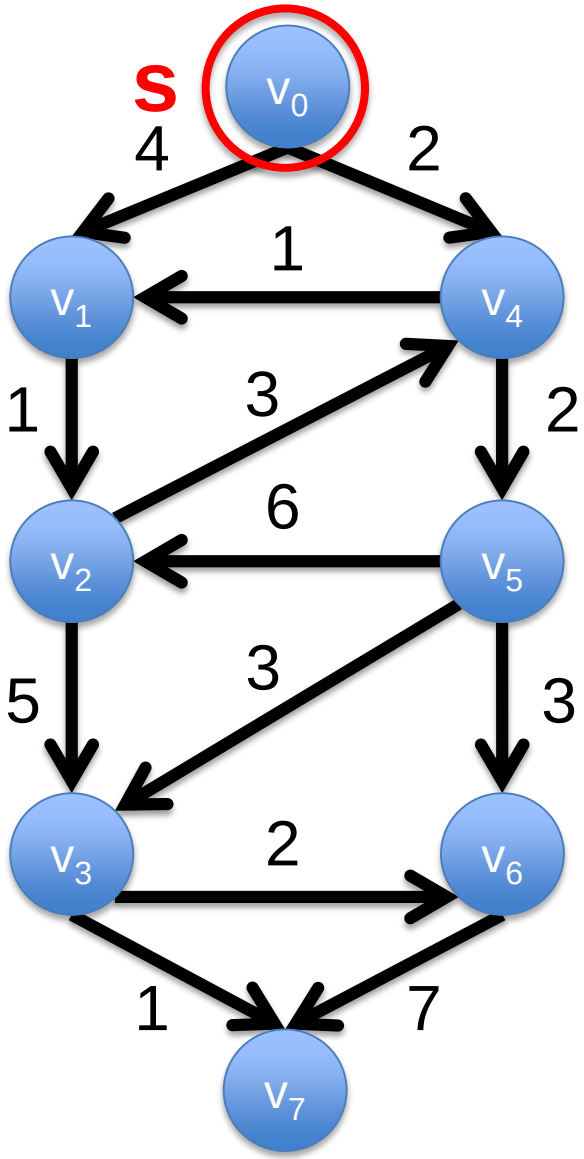


Set U = U \ {u}

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	2	4	7	8

Dijkstra's Method



Set U = U \ {u}

Choose u = $\text{argmin}_{v \in U}$ d(v)

Stop.

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	2	4	7	8

Summary

Set $d(v) = \infty$ for all $v \in V$. Set $U = V$.

Set $d(s) = 0$. Set $u = s$.

While u is not equal to t

Choose $u = \text{argmin}_{v \in U} d(v)$

For each v such that $(u,v) \in A$

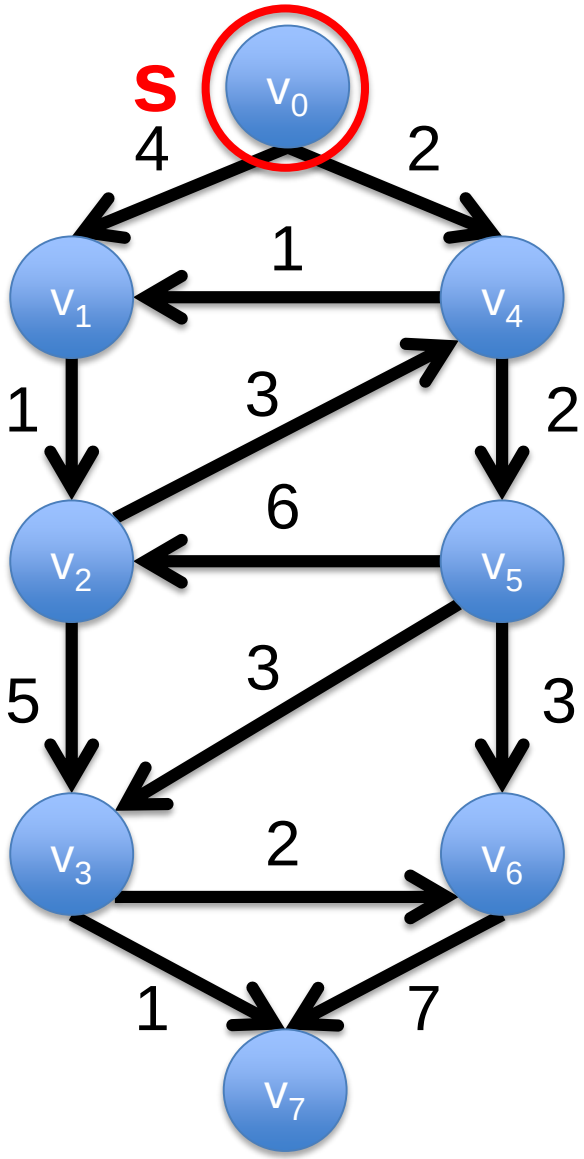
$d(v) = \min \{ d(v), d(u) + l(u,v)\}$

End

Set $U = U \setminus \{u\}$

End

Dijkstra's Method - Path



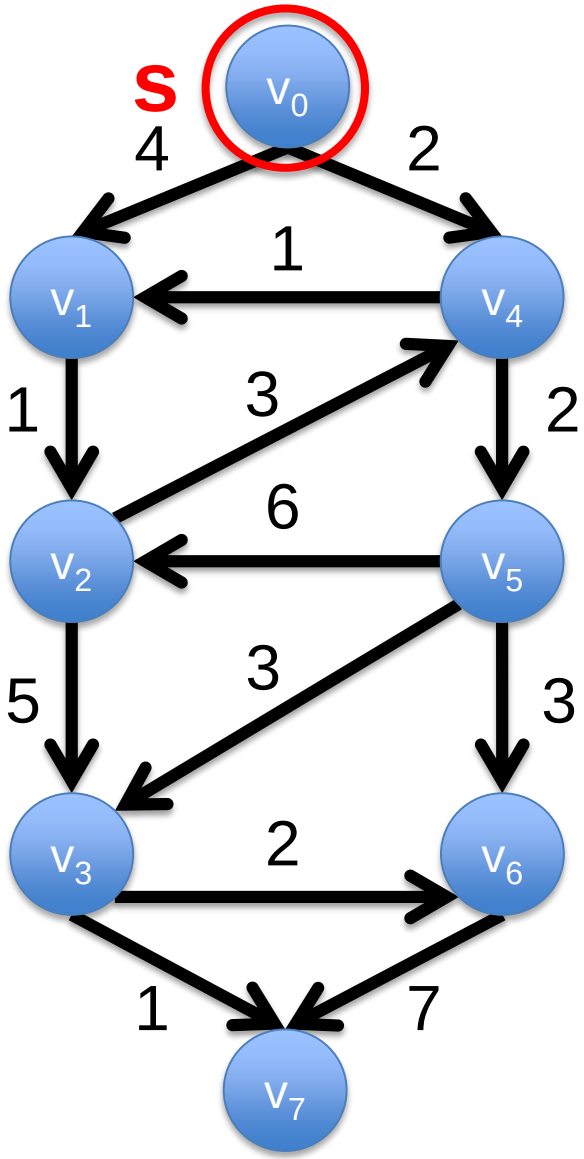
Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
∞	∞	∞	∞	∞	∞	∞	∞

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
∞	∞	∞	∞	∞	∞	∞	∞

Dijkstra's Method - Path



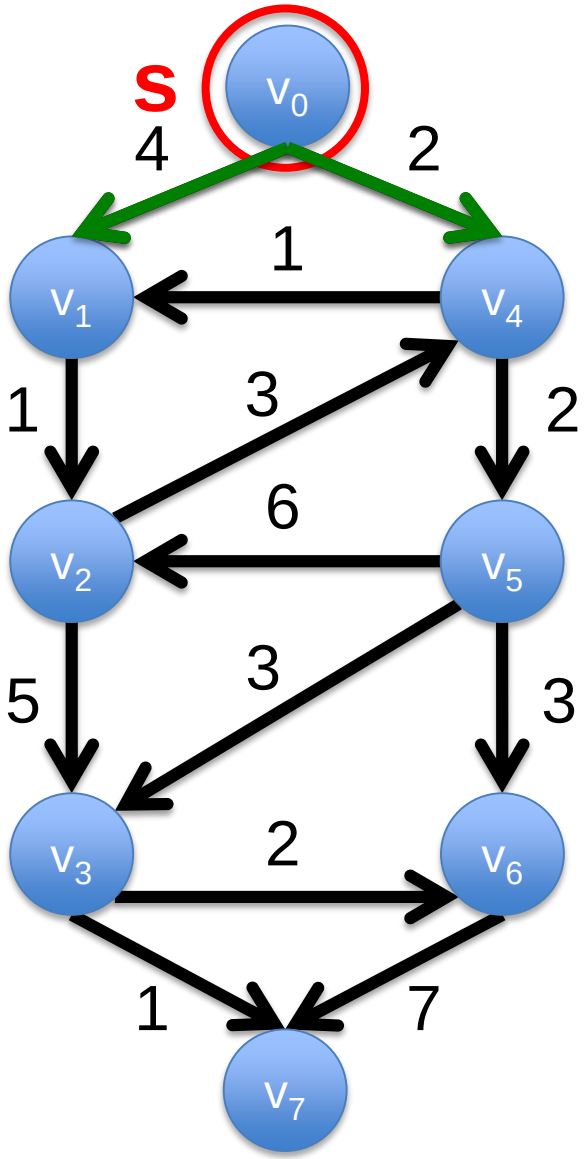
Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	∞	∞	∞	∞	∞	∞	∞

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	∞	∞	∞	∞	∞	∞	∞

Dijkstra's Method - Path



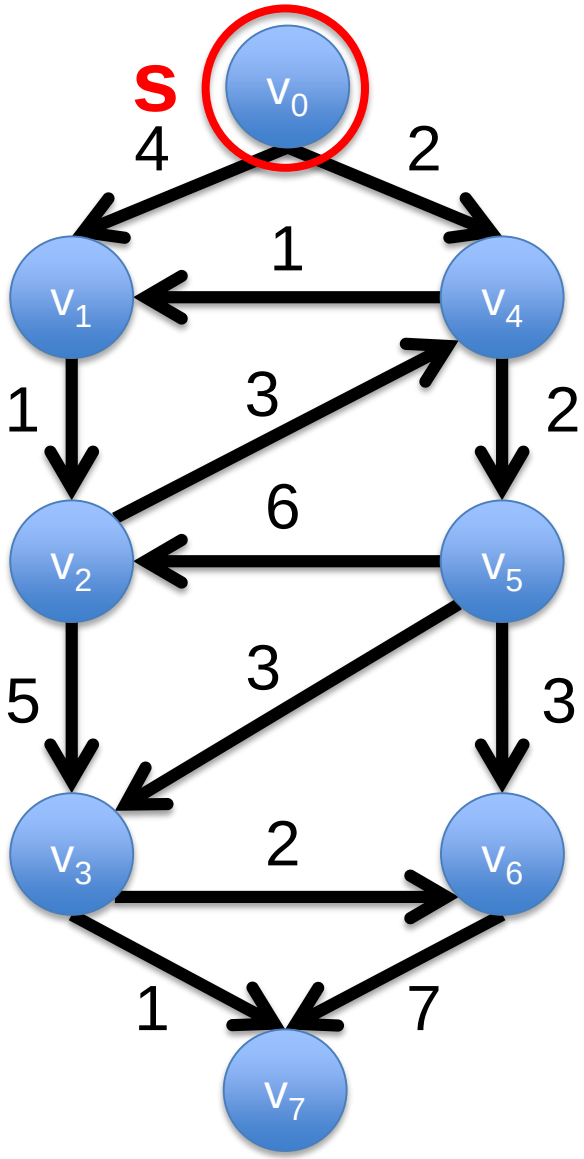
Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	∞	∞	∞	∞	∞	∞	∞

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	∞	∞	∞	∞	∞	∞	∞

Dijkstra's Method - Path



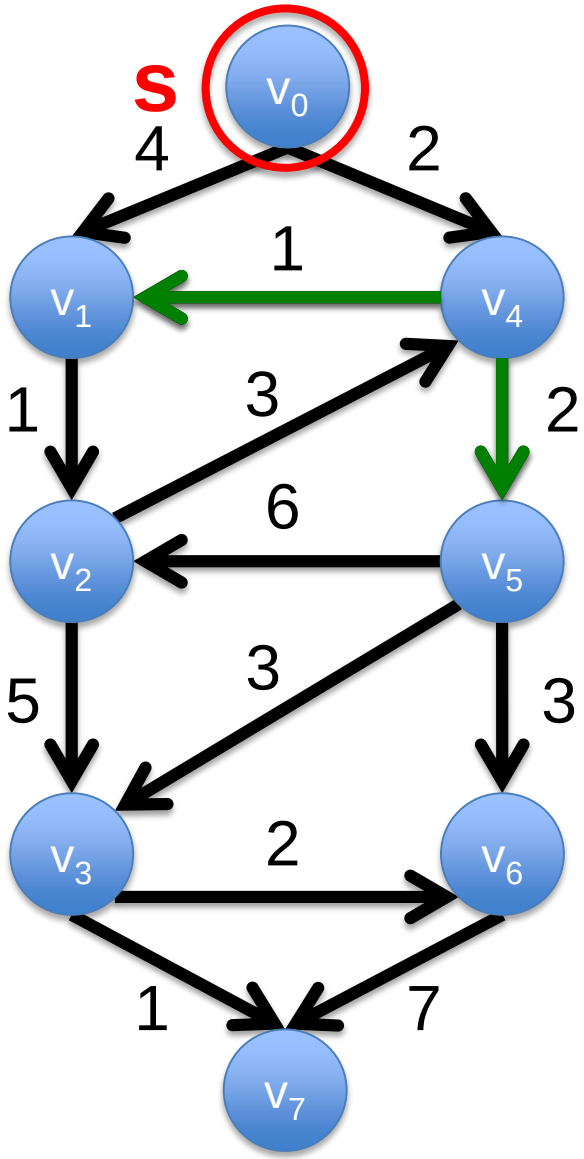
Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	∞	∞	v_0	∞	∞	∞

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	4	∞	∞	2	∞	∞	∞

Dijkstra's Method - Path



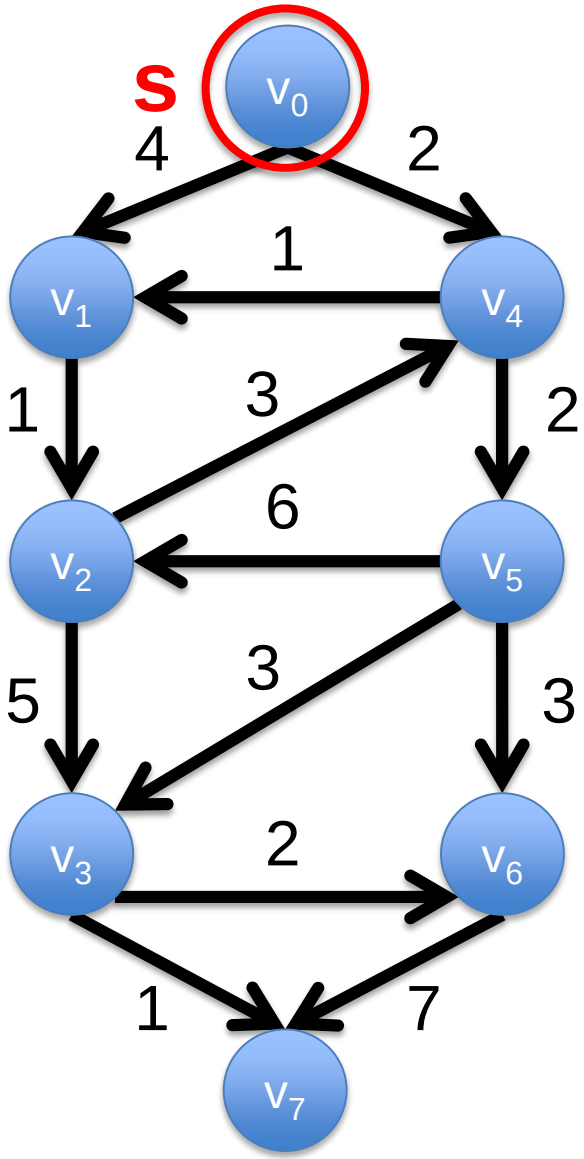
Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	∞	∞	v_0	∞	∞	∞

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	4	∞	∞	2	∞	∞	∞

Dijkstra's Method - Path



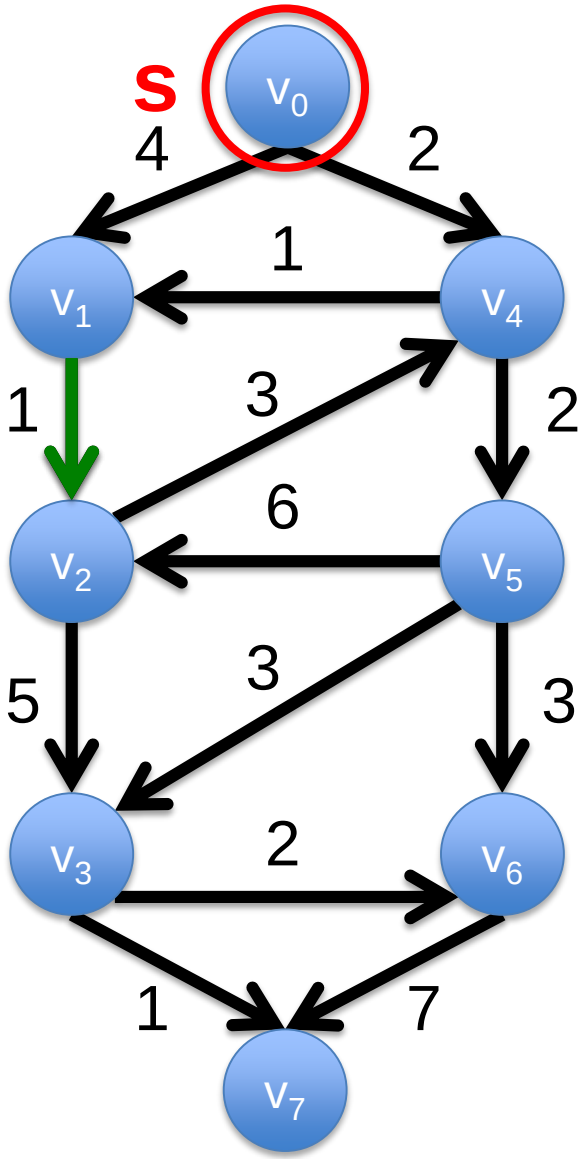
Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_4	∞	∞	v_0	v_4	∞	∞

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	4	∞	∞

Dijkstra's Method - Path



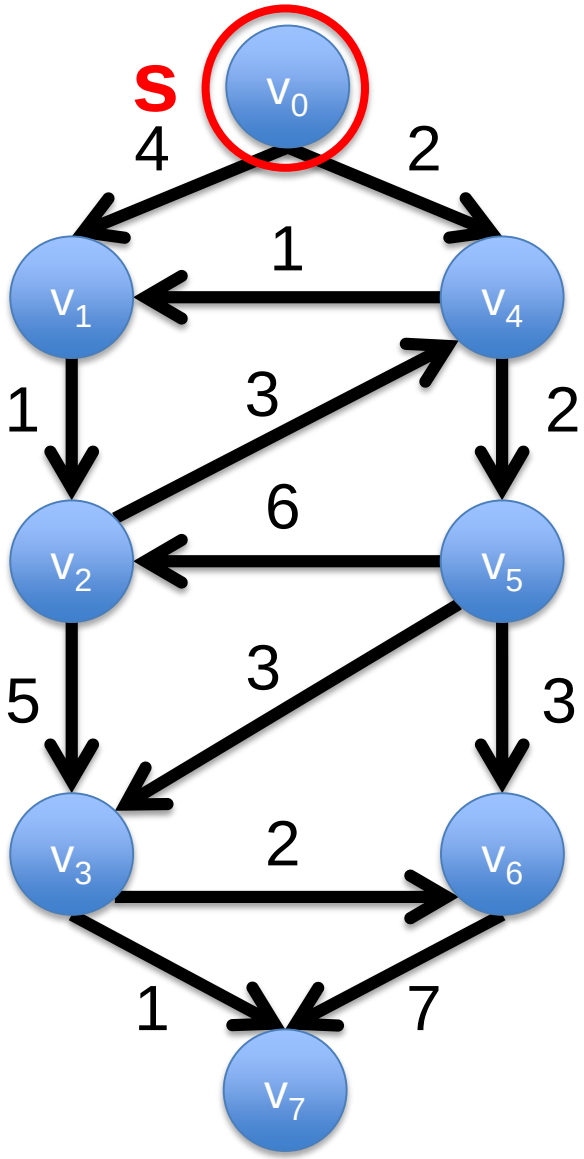
Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_4	∞	∞	v_0	v_4	∞	∞

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	4	∞	∞

Dijkstra's Method - Path



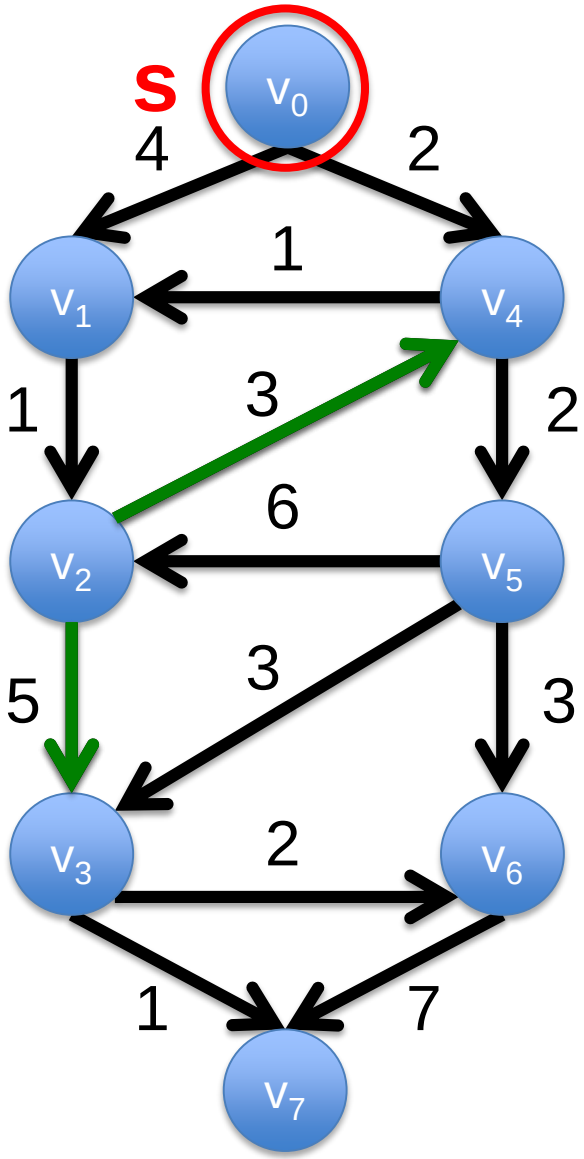
Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_4	v_1	∞	v_0	v_4	∞	∞

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	2	4	∞	∞

Dijkstra's Method - Path



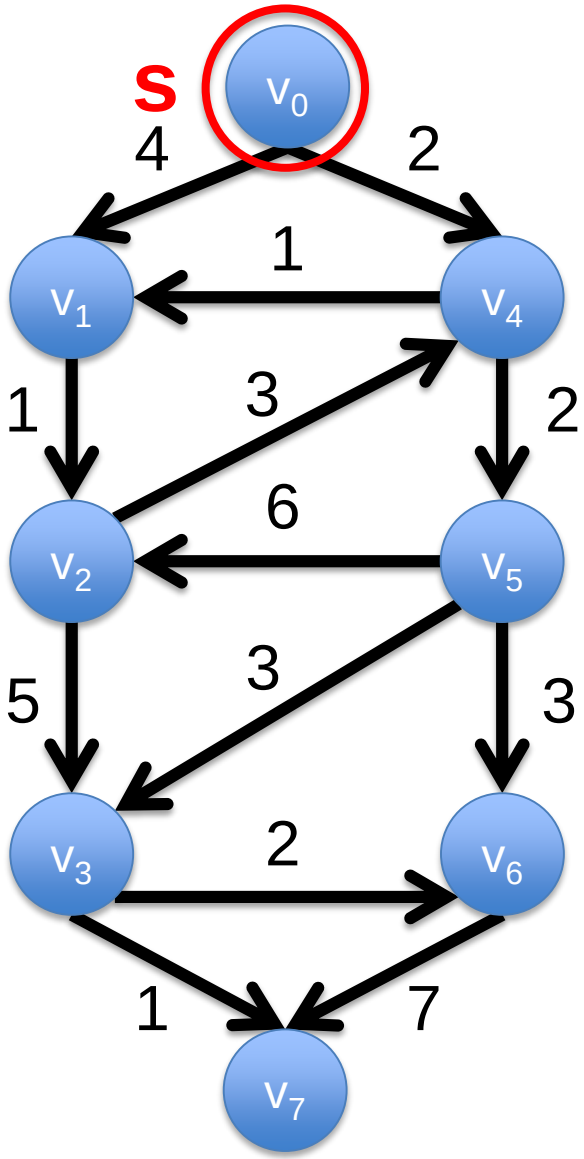
Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_4	v_1	∞	v_0	v_4	∞	∞

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	2	4	∞	∞

Dijkstra's Method - Path



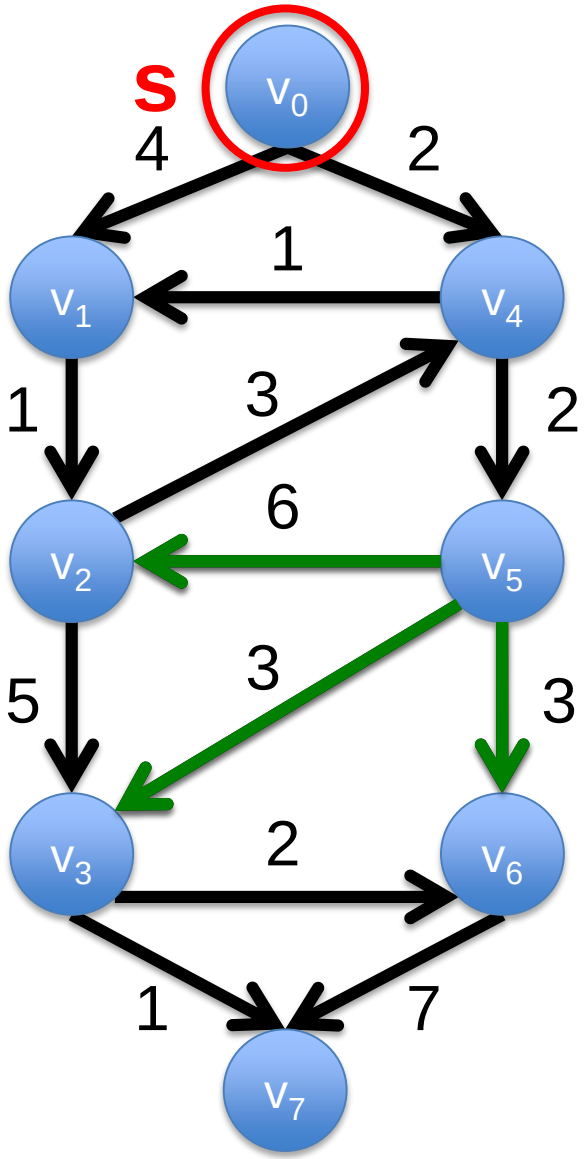
Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_4	v_1	v_2	v_0	v_4	∞	∞

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	9	2	4	∞	∞

Dijkstra's Method - Path



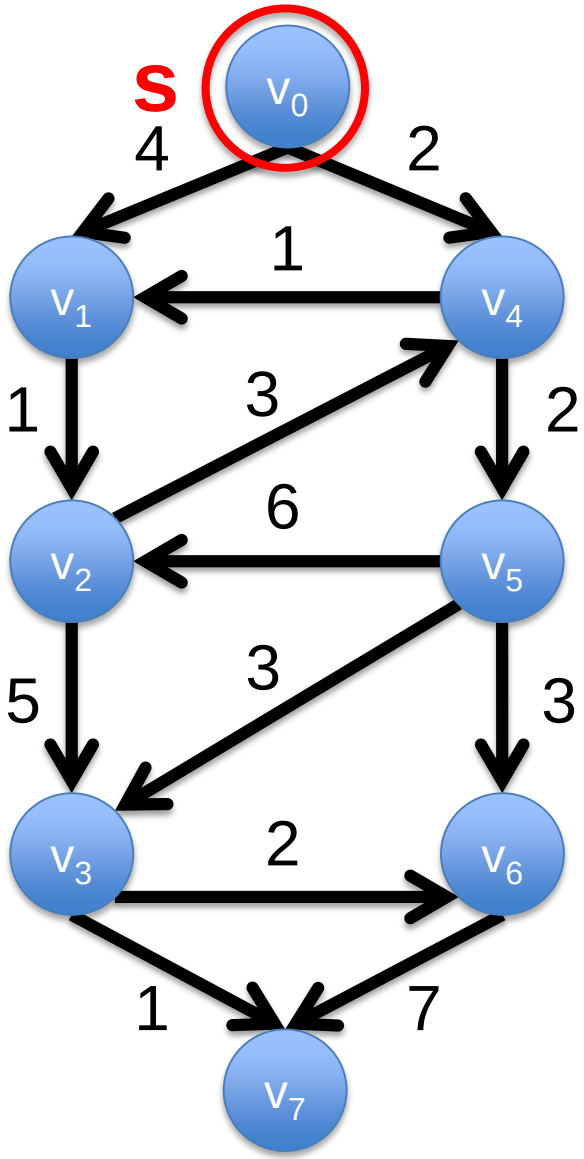
Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_4	v_1	v_2	v_0	v_4	∞	∞

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	9	2	4	∞	∞

Dijkstra's Method - Path



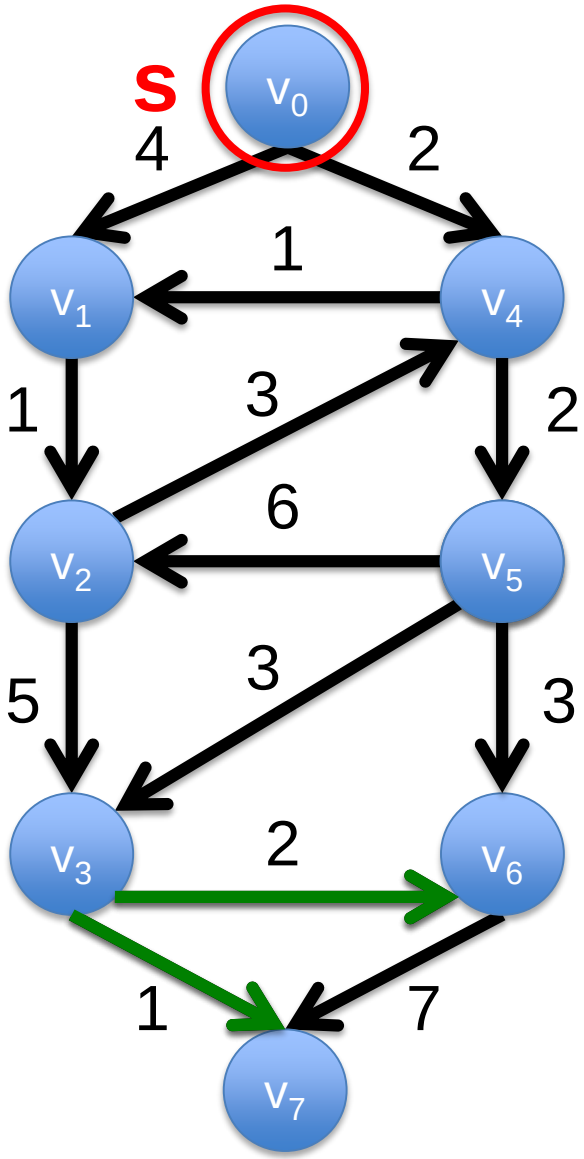
Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_4	v_1	v_5	v_0	v_4	v_5	∞

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	2	4	7	∞

Dijkstra's Method - Path



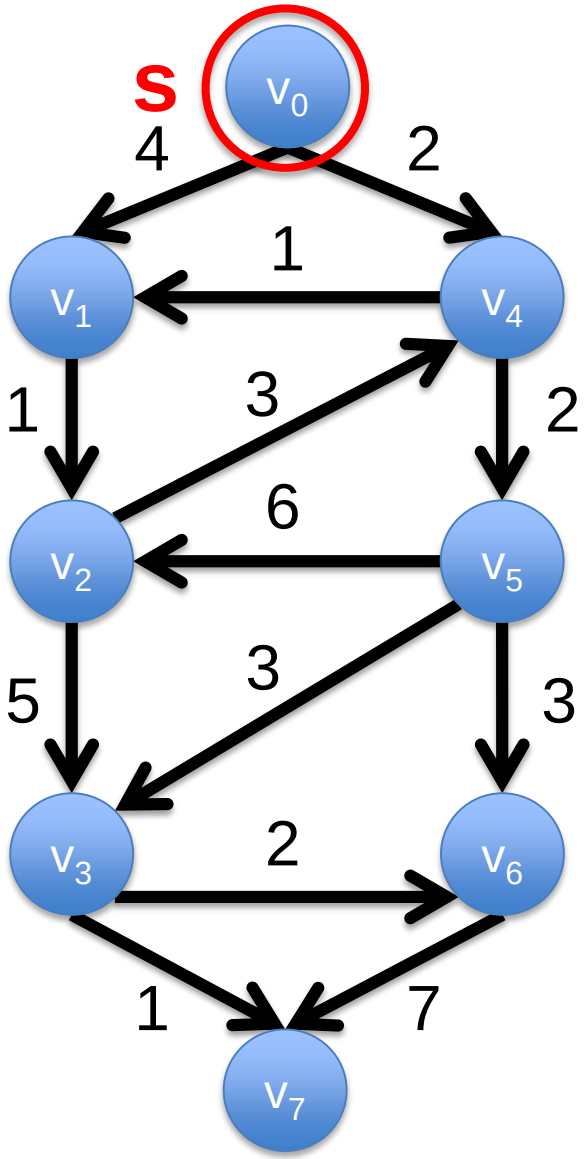
Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_4	v_1	v_5	v_0	v_4	v_5	∞

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	2	4	7	∞

Dijkstra's Method - Path



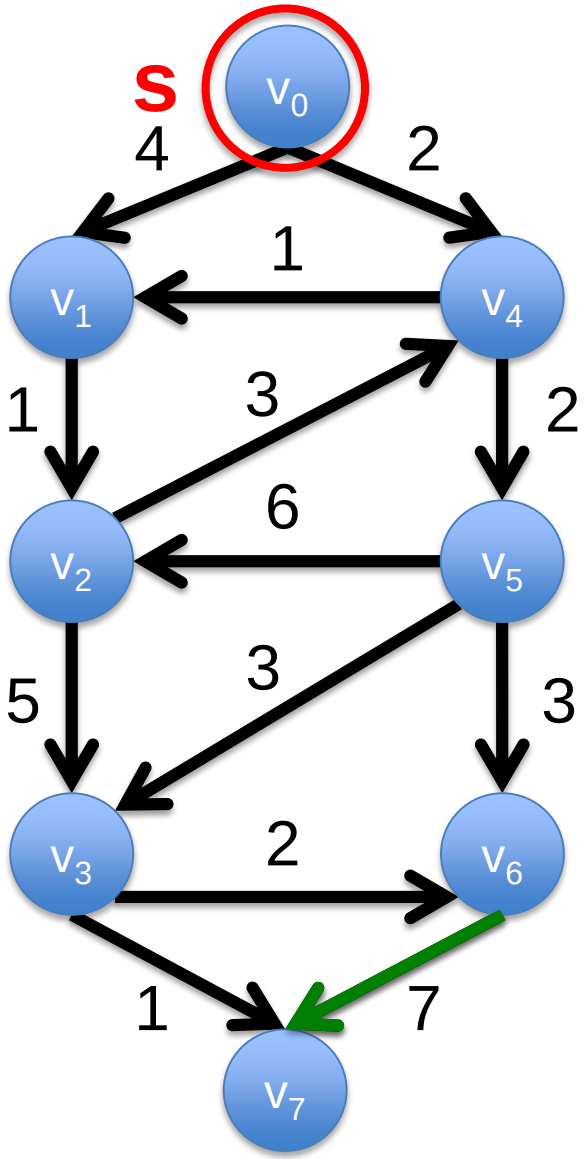
Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_4	v_1	v_5	v_0	v_4	v_5	v_3

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	2	4	7	8

Dijkstra's Method - Path



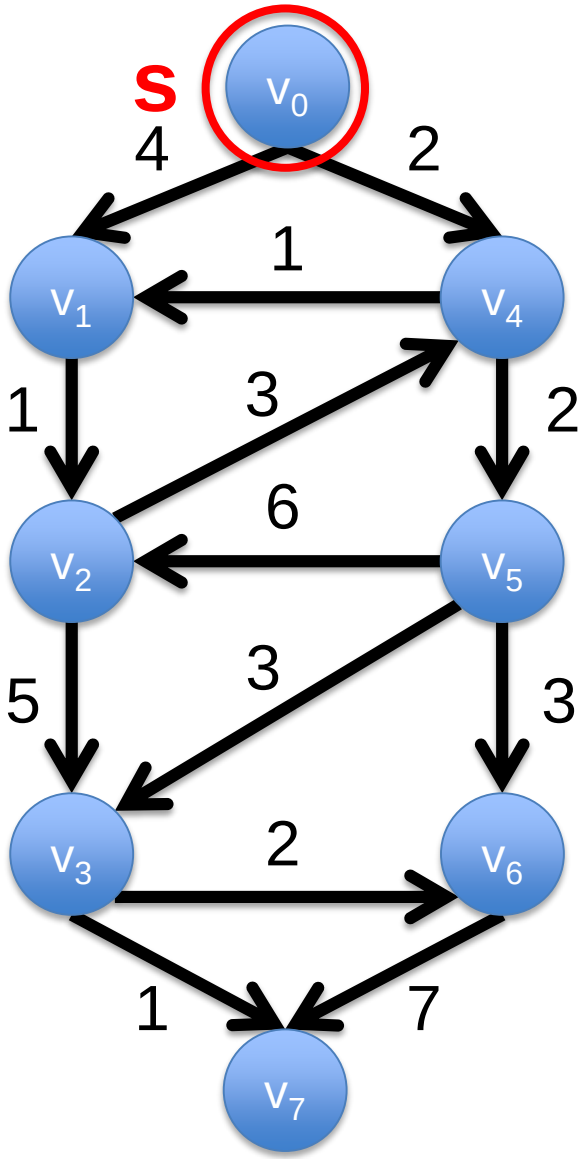
Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_4	v_1	v_5	v_0	v_4	v_5	v_3

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	2	4	7	8

Dijkstra's Method - Path



Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_4	v_1	v_5	v_0	v_4	v_5	v_3

d: Over-estimation of distance

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	2	4	7	8

Time Complexity

Maximum ‘n’ iterations

Each iteration is O(n)

Total time complexity = $O(n^2)$

Graphs with up to 1000 nodes

Can we do faster?

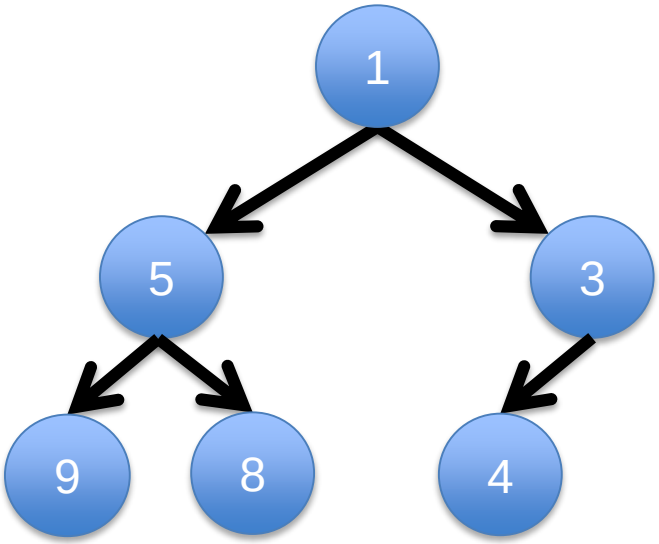
Yes, we can do O((n + m) log n)

We need a data structure to store d:
priority queue or heap

It allows 2 operations on elements:

1) Popping the minimum in O(log n)
2) Modifying an element in O(log n)

Data structure: heap



Binary tree, where each node is smaller than its children

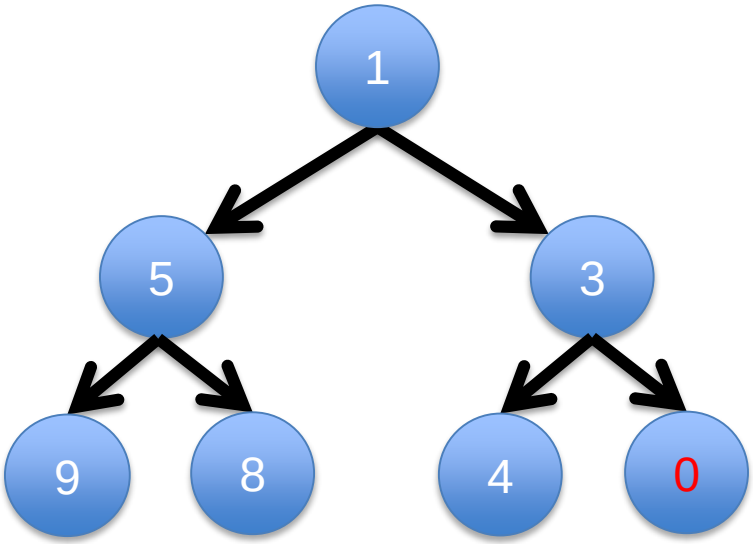
Minimum is always at the root

Shape: balanced tree

Binary tree can be implemented with an array:

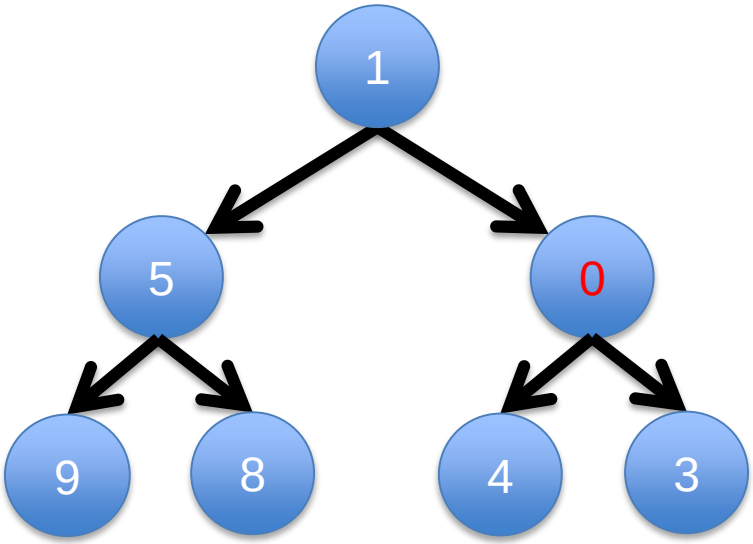
1	2	3	4	5	6
1	5	3	9	8	4

Heap: adding elements



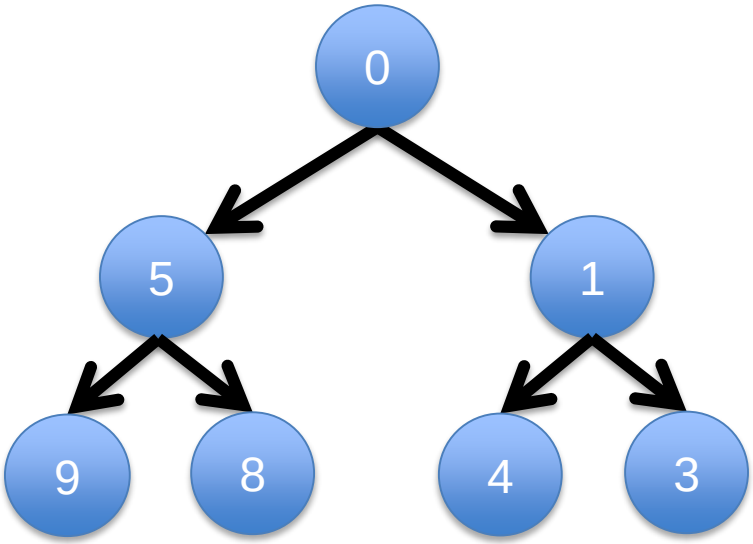
When adding an element we check its parent to keep the heap properties.

Heap: adding elements



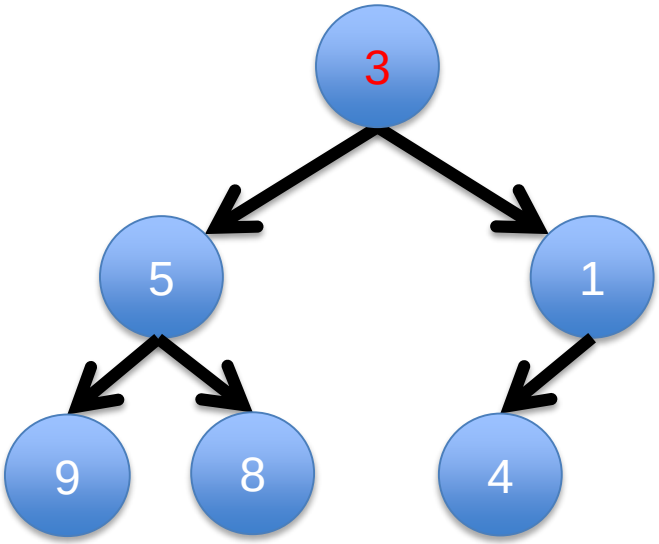
When adding an element we check its parent to keep the heap properties.

Heap: adding elements



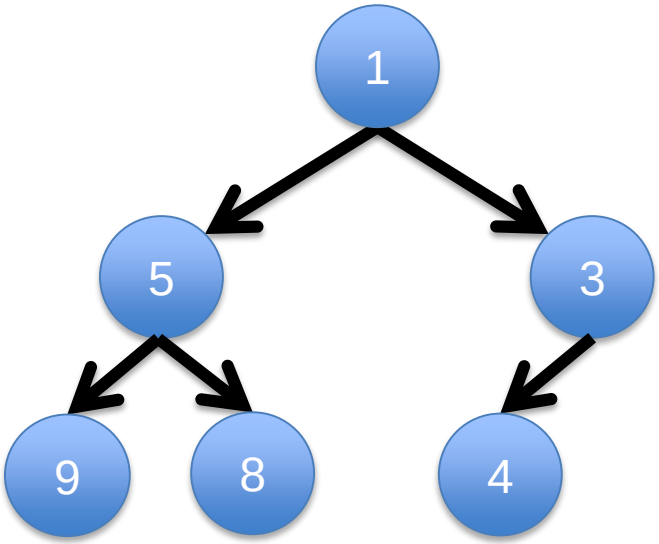
When adding an element we check its parent to keep the heap properties.

Heap: popping elements



When popping an element we replace the root with the last element.
We check its children to keep the heap properties.

Heap: popping elements



When popping an element we replace the root with the last element.
We check its children to keep the heap properties.

Dijkstra's Method with heap

Set $d(v) = \infty$ for all $v \in V$. Set $U = V$.

Set $d(s) = 0$. Set $u = s$.

Init heap

While u is not equal to t

    Pop from heap $u = \text{argmin}_{v \in U} d(v)$

    For each v such that $(u,v) \in A$

        $d(v) = \min \{ d(v), d(u) + l(u,v)\}$

        Update d(v) in the heap

    End

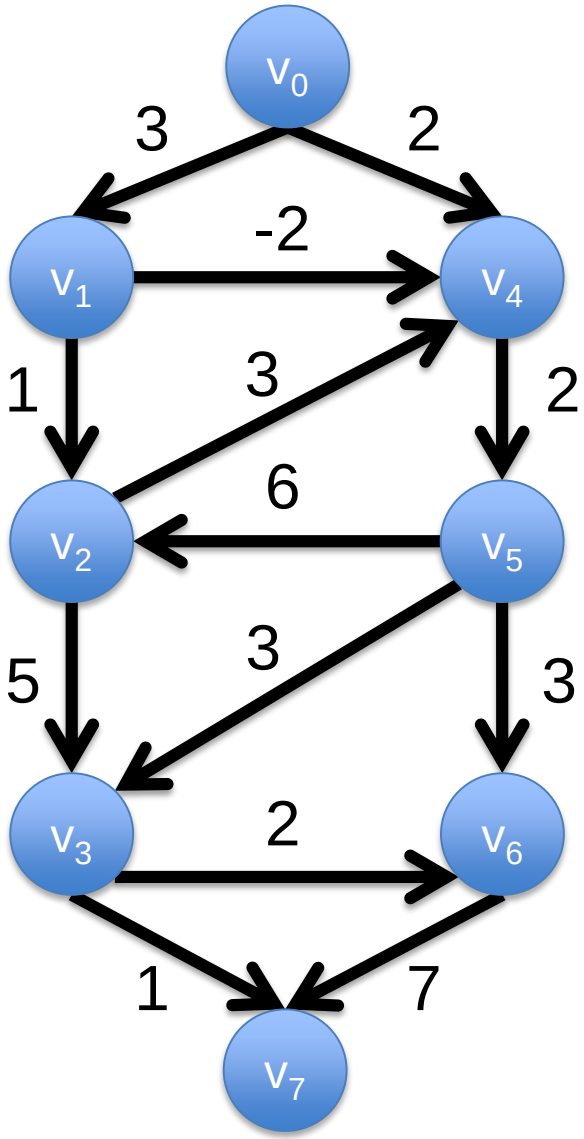
    Set $U = U \setminus \{u\}$

End

Outline

- Graph Preliminaries
- Complexity Preliminaries
- Shortest Path Algorithms
  - Breadth-first Search
  - Dijkstra's Method
  - **Bellman-Ford Method**
  - Floyd-Warshall Method

Bellman-Ford Method



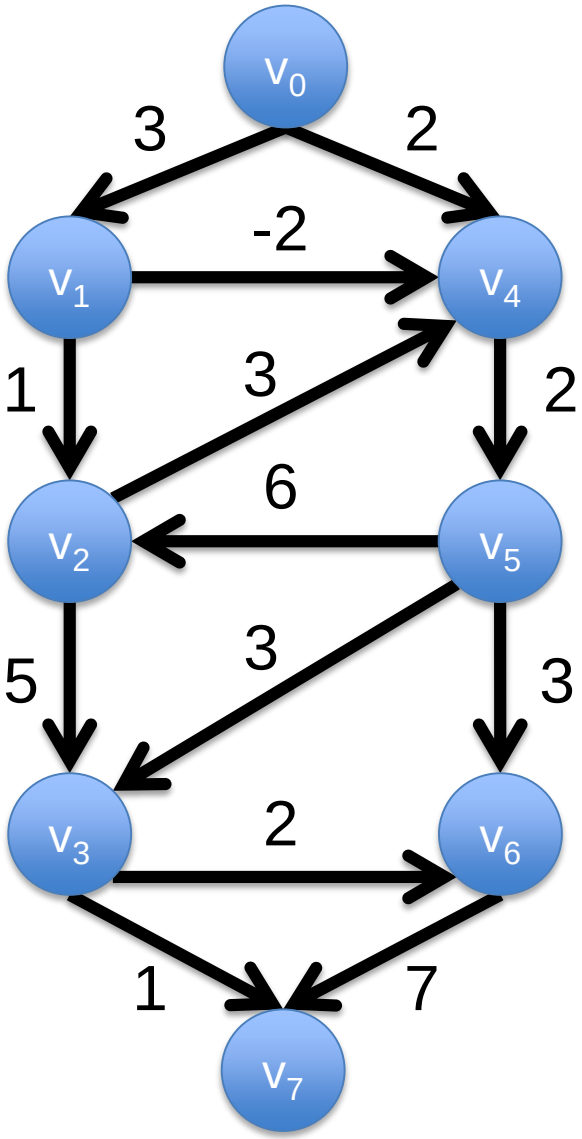
D = (V, A)

Assumption: No negative length directed circuits.

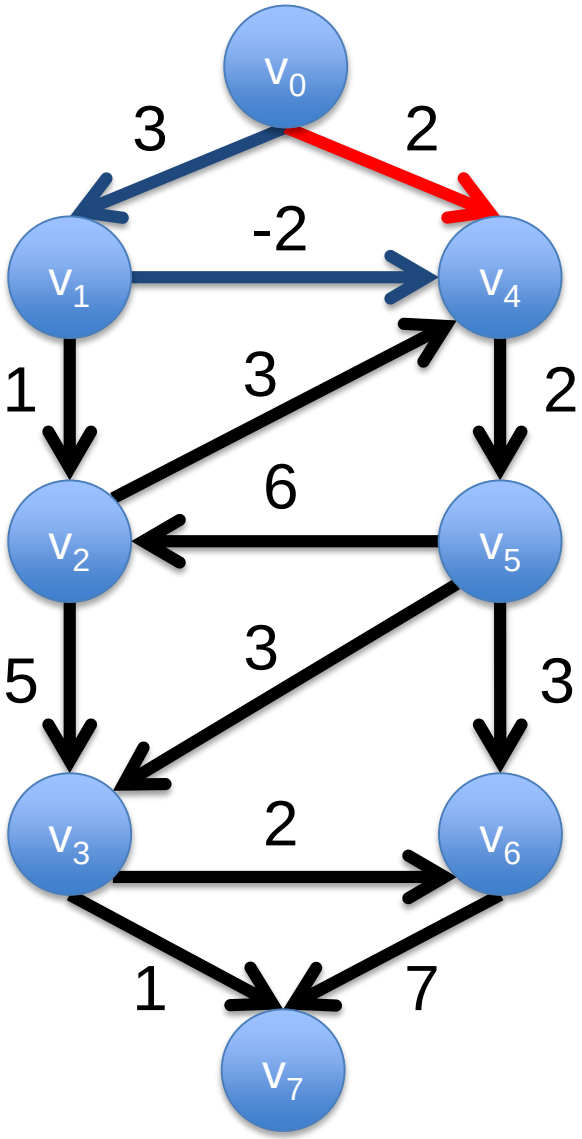
Length of path =
Σ arc lengths = Σ l(u,v)

|V| = n, |A| = m

Dijkstra's Method Fails



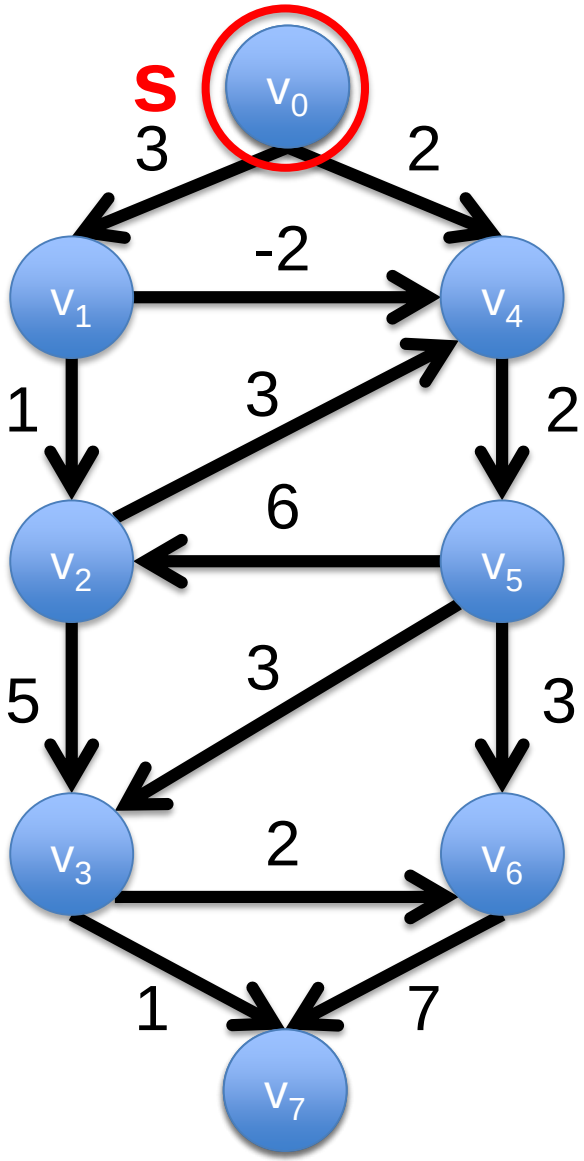
Dijkstra's Method Fails



If $s = v_0$, $u = v_4$ in the first iteration

d(u) > **dist(u)**

Bellman-Ford Method



Let $d_k(t)$ = minimum length s-t walk traversing **at most** k arcs.

$d_{k+1}(t)$ = minimum of

$d_k(t)$,

$\min_{(u,t) \in A} d_k(u) + l(u,t)$

Bellman-Ford Method

$k = 0.\ d_k(s) = 0$

While $(k < n)$

$d_{k+1}(t) = d_k(t)$

For each $(u,v) \in A$

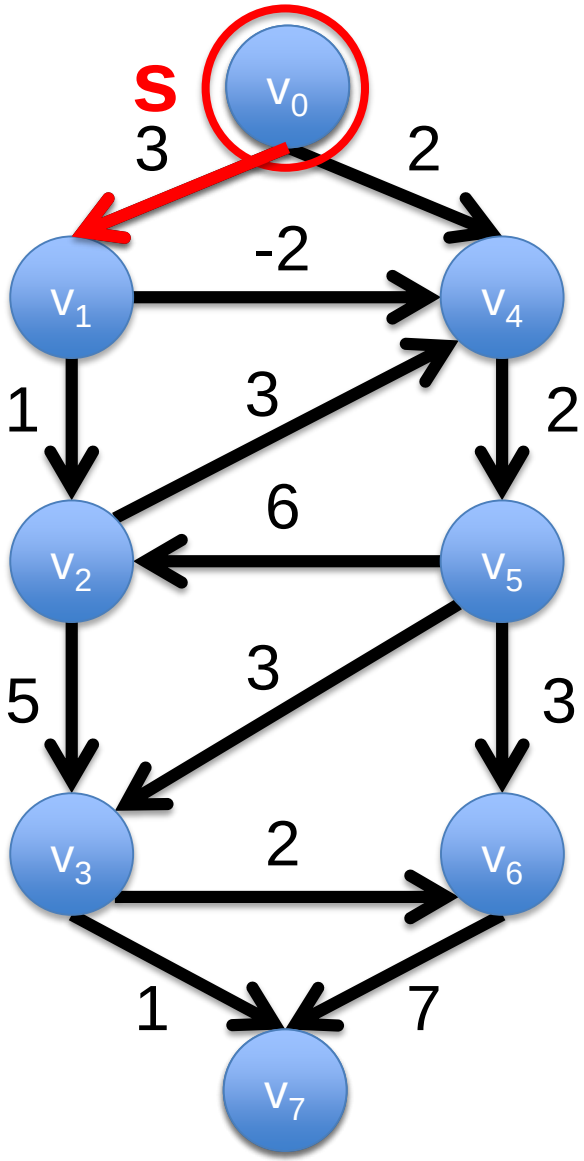
$d_{k+1}(v) = \min \{ d_{k+1}(v), d_k(u) + l(u,v) \}$

End

$k = k + 1$

End

Bellman-Ford Method



k = 0

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	∞	∞	∞	∞	∞	∞	∞

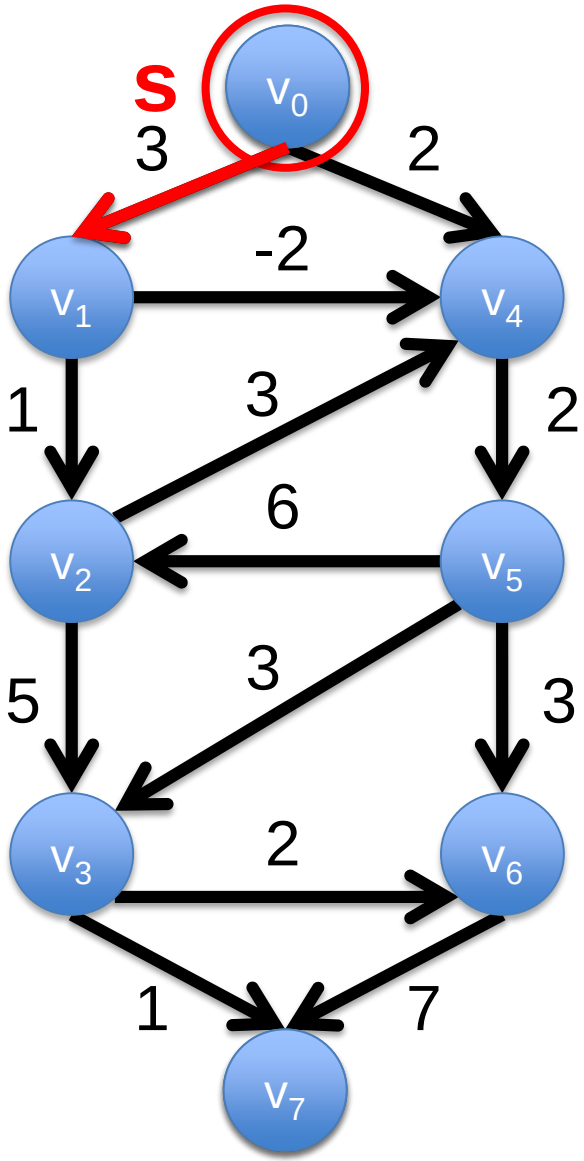
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	∞	∞	∞	∞	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A							

Bellman-Ford Method



k = 0

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	∞	∞	∞	∞	∞	∞	∞

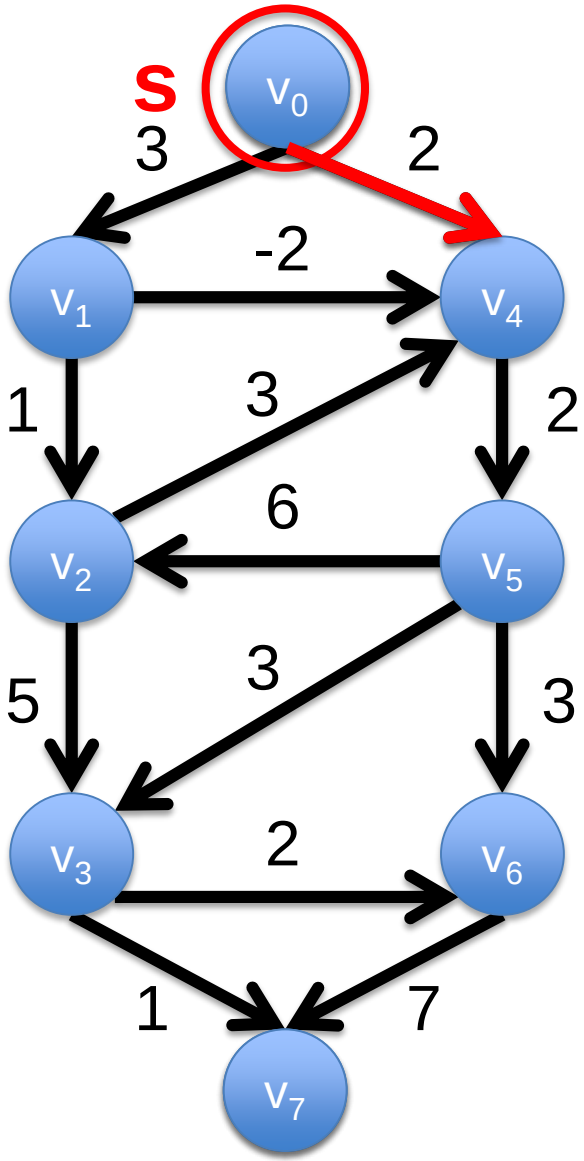
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	∞	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0						

Bellman-Ford Method



k = 0

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	∞	∞	∞	∞	∞	∞	∞

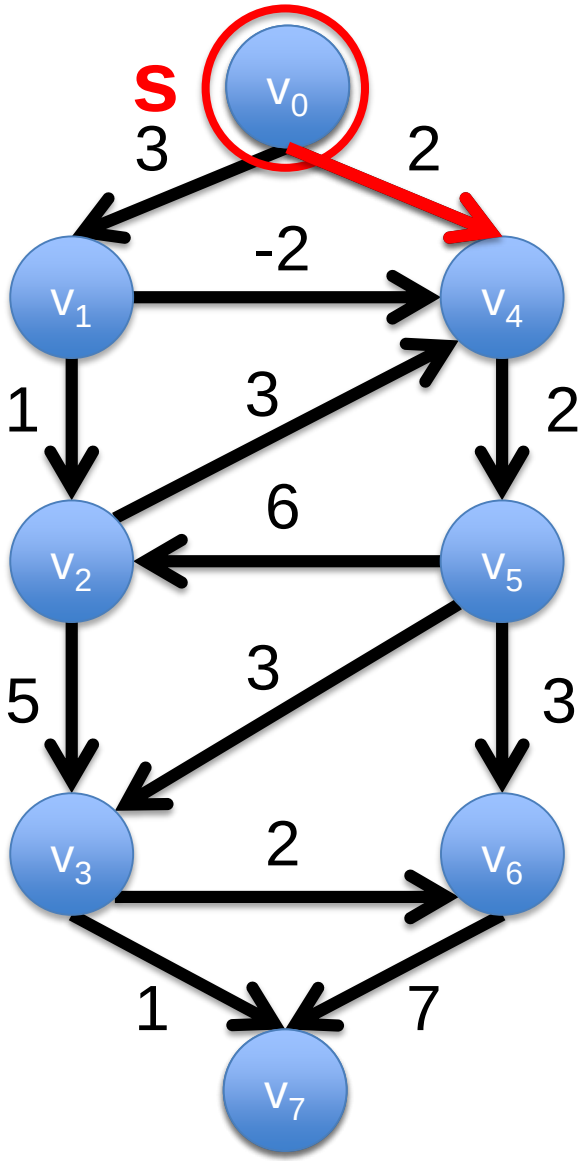
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	∞	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0						

Bellman-Ford Method



k = 0

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	∞	∞	∞	∞	∞	∞	∞

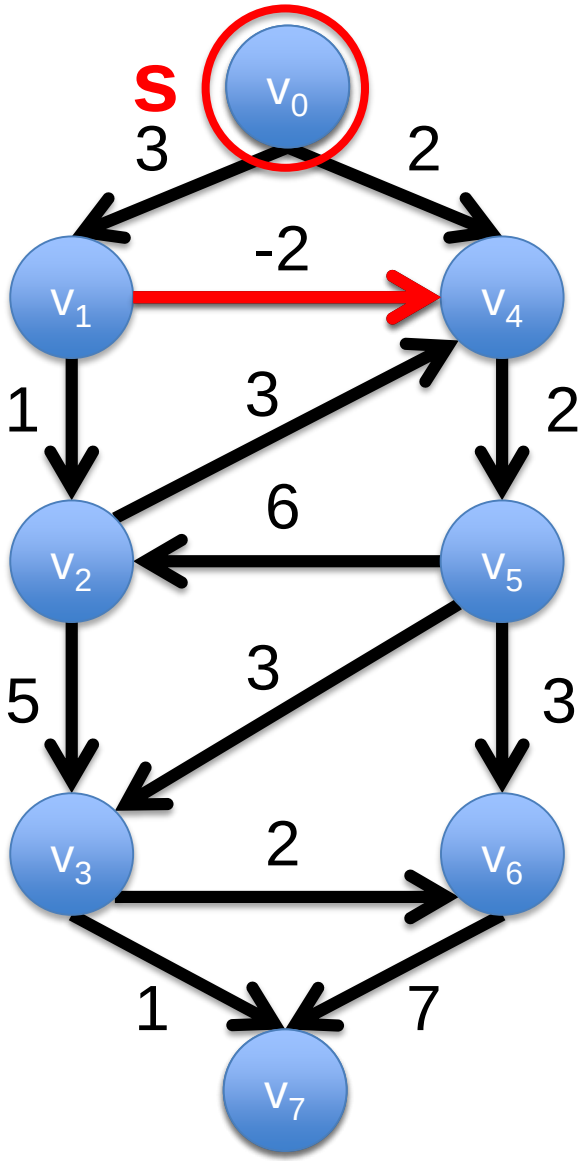
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0			v_0			

Bellman-Ford Method



k = 0

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	∞	∞	∞	∞	∞	∞	∞

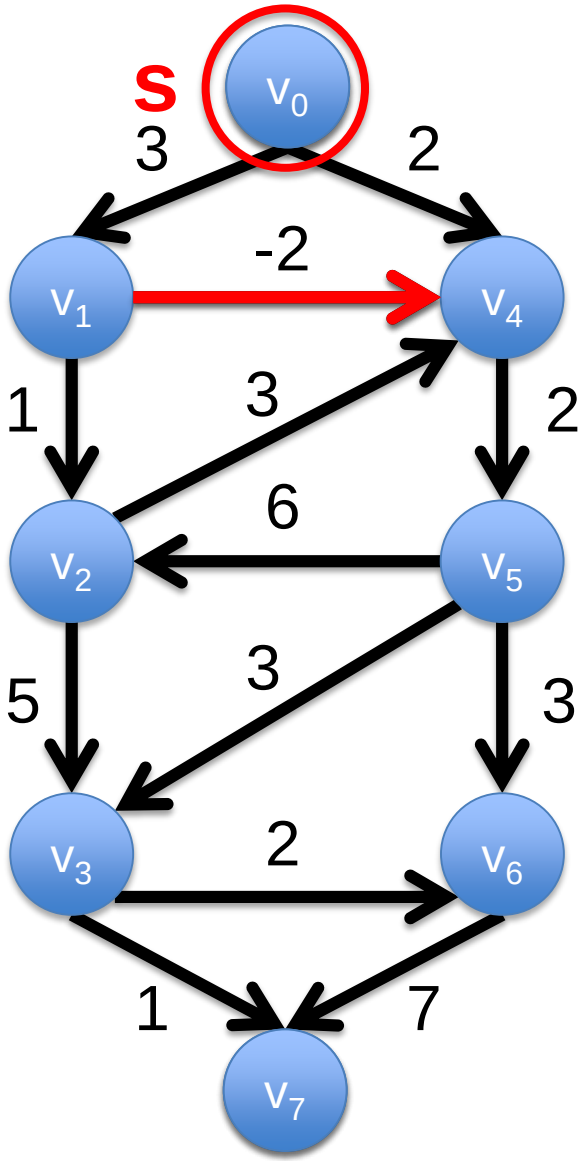
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0			v_0			

Bellman-Ford Method



k = 0 All other d_{k+1} are ∞

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	∞	∞	∞	∞	∞	∞	∞

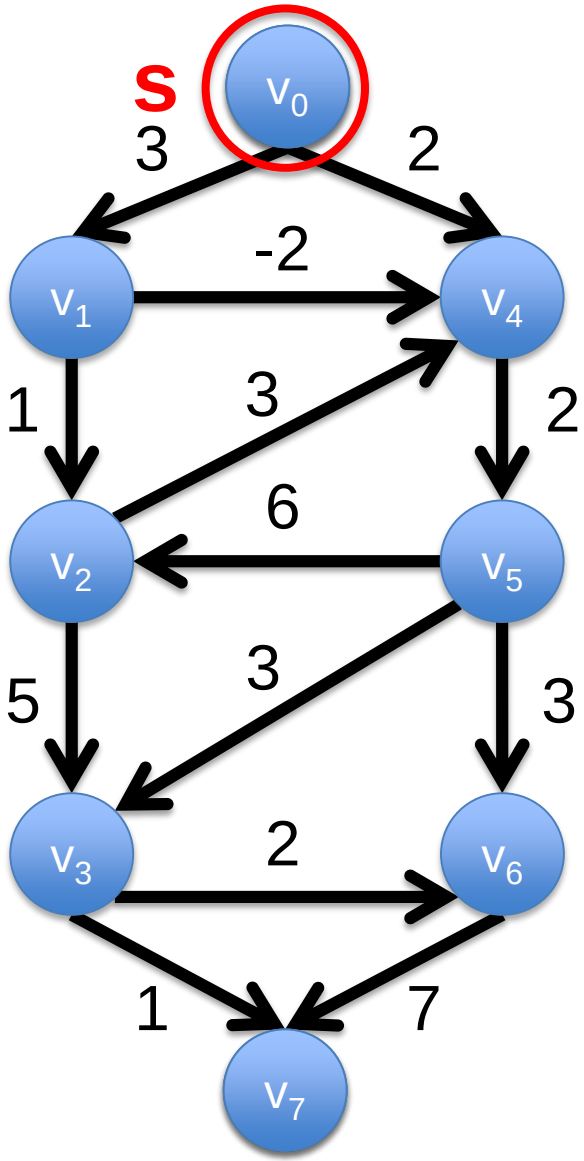
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0			v_0			

Bellman-Ford Method



$k = k + 1 = 1$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

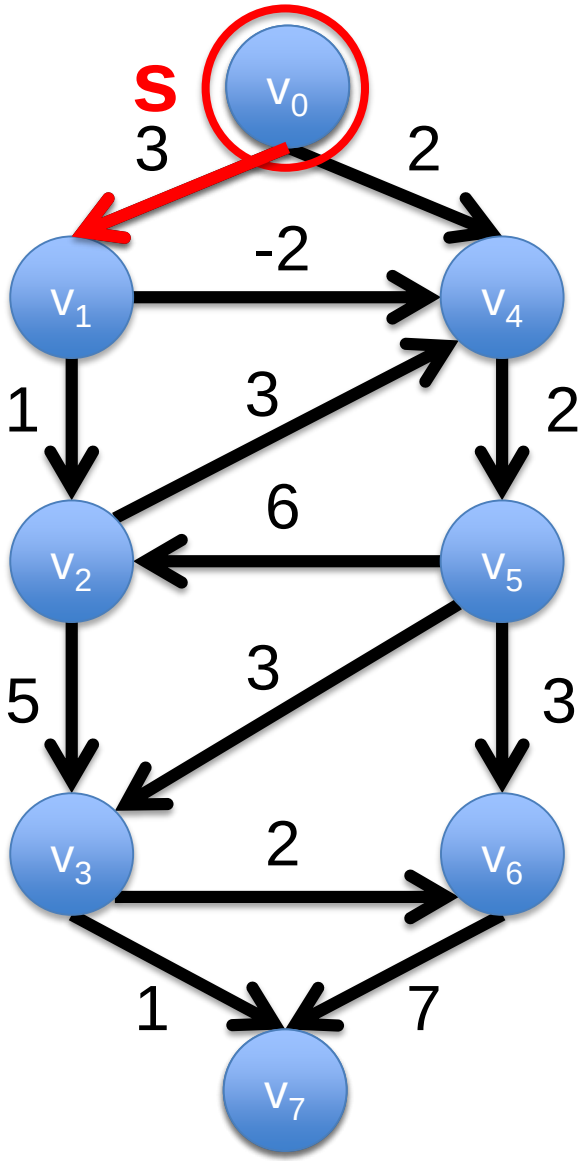
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0			v_0			

Bellman-Ford Method



$k = k + 1 = 1$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

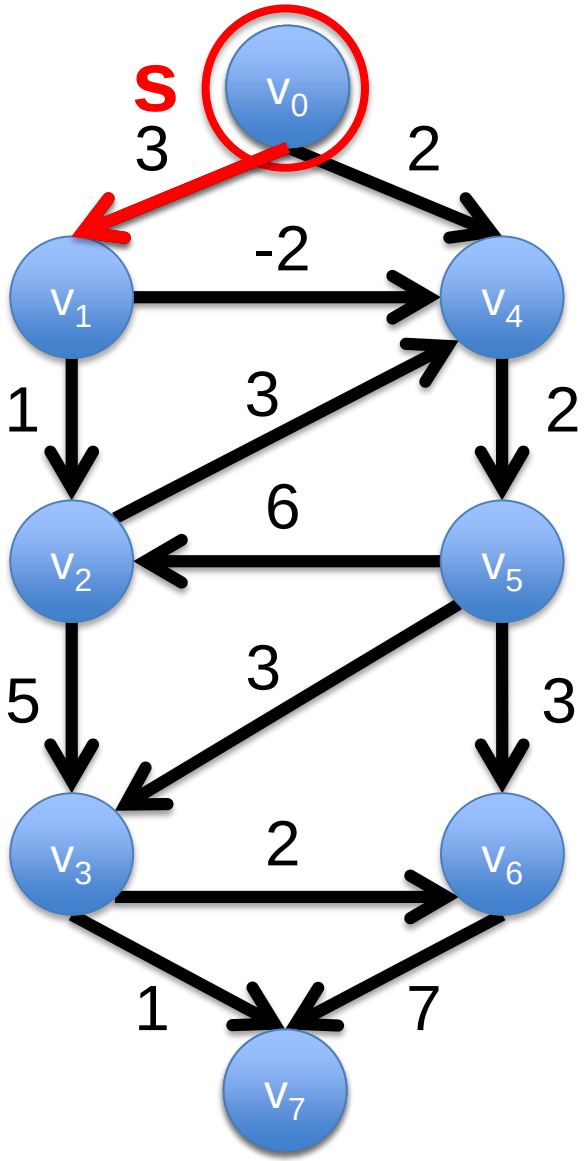
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0			v_0			

Bellman-Ford Method



$k = k + 1 = 1$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

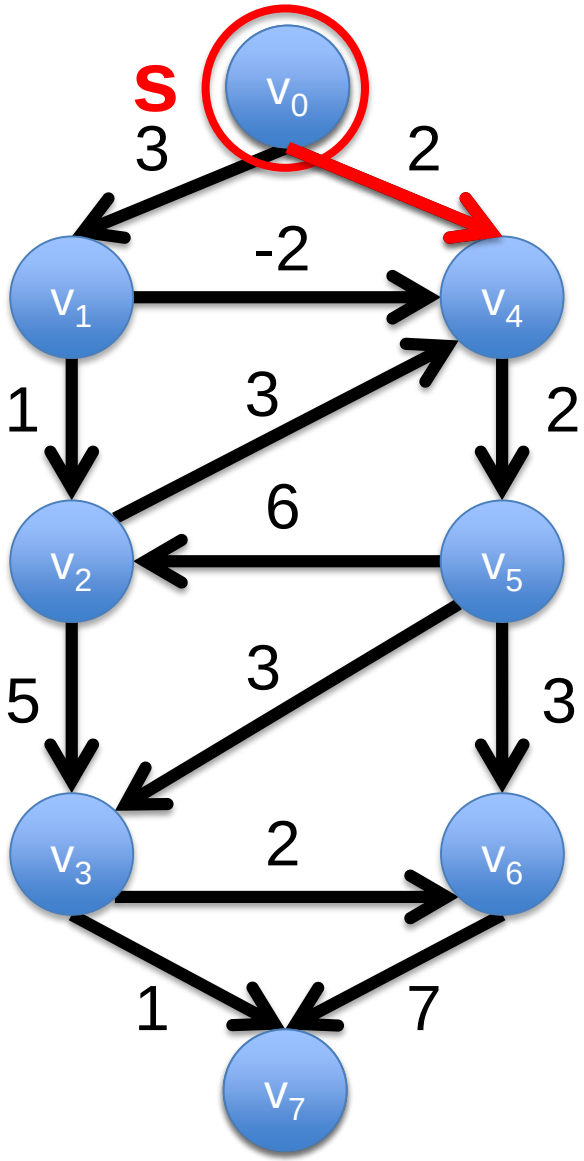
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0			v_0			

Bellman-Ford Method



$k = k + 1 = 1$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

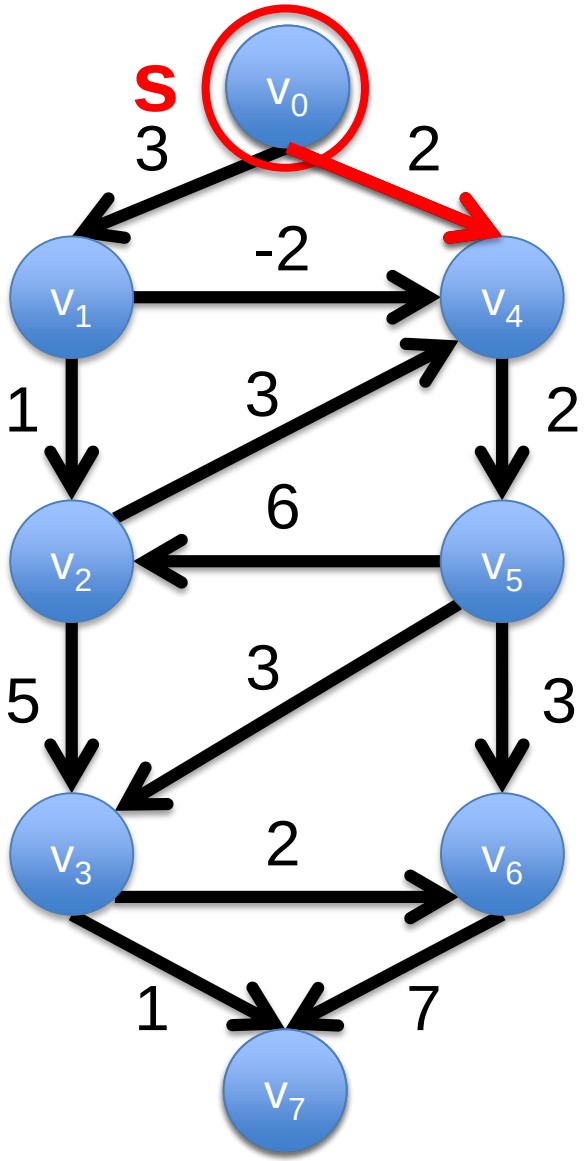
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0			v_0			

Bellman-Ford Method



$k = k + 1 = 1$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

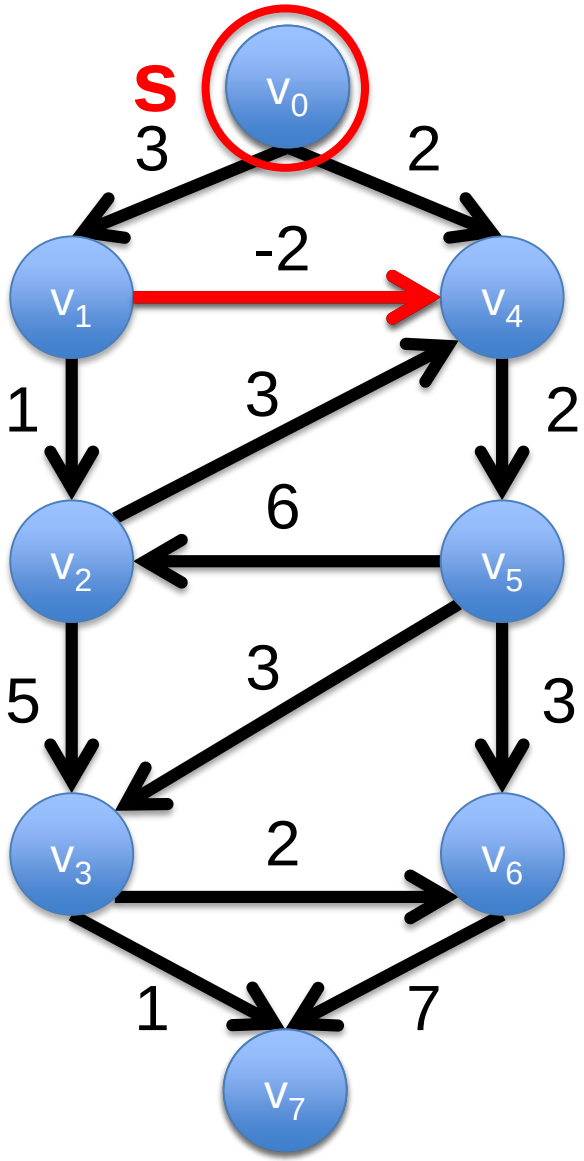
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0			v_0			

Bellman-Ford Method



$k = k + 1 = 1$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

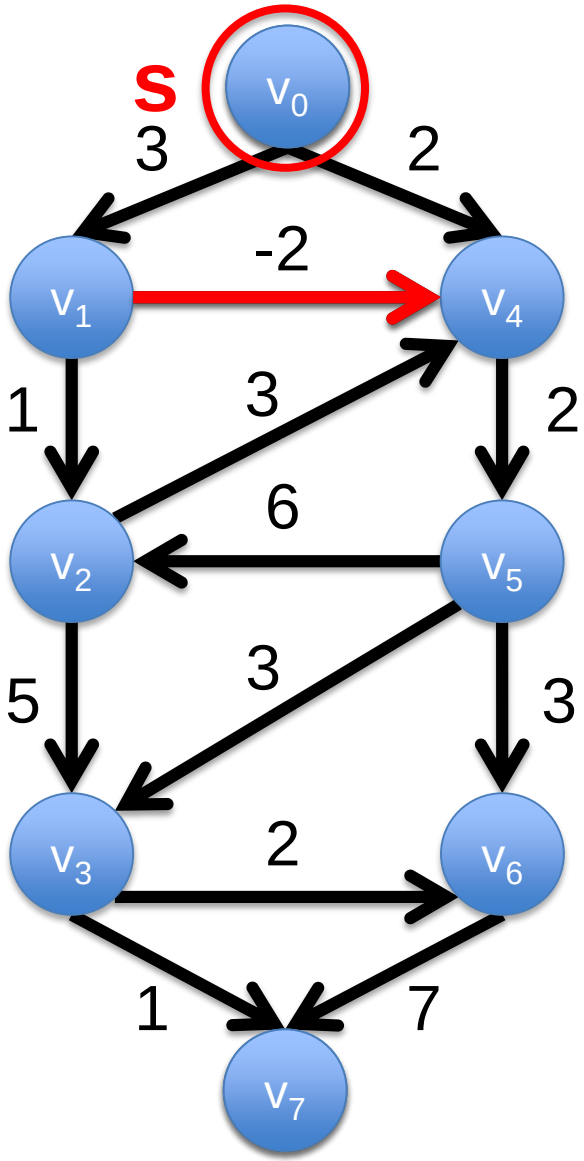
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0			v_0			

Bellman-Ford Method



$k = k + 1 = 1$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

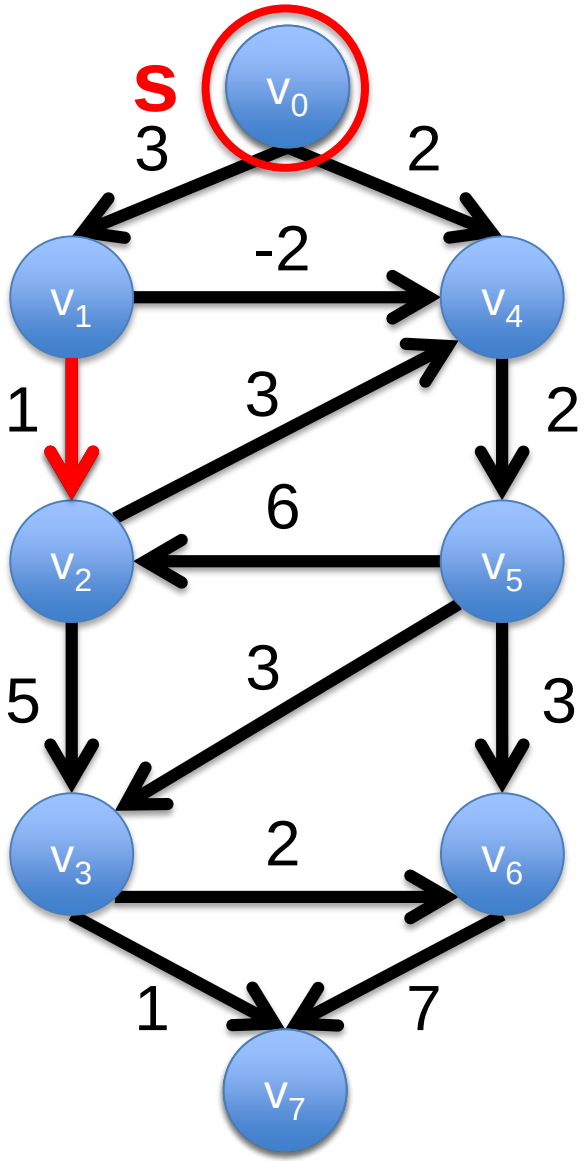
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	1	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0			v_1			

Bellman-Ford Method



$k = k + 1 = 1$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

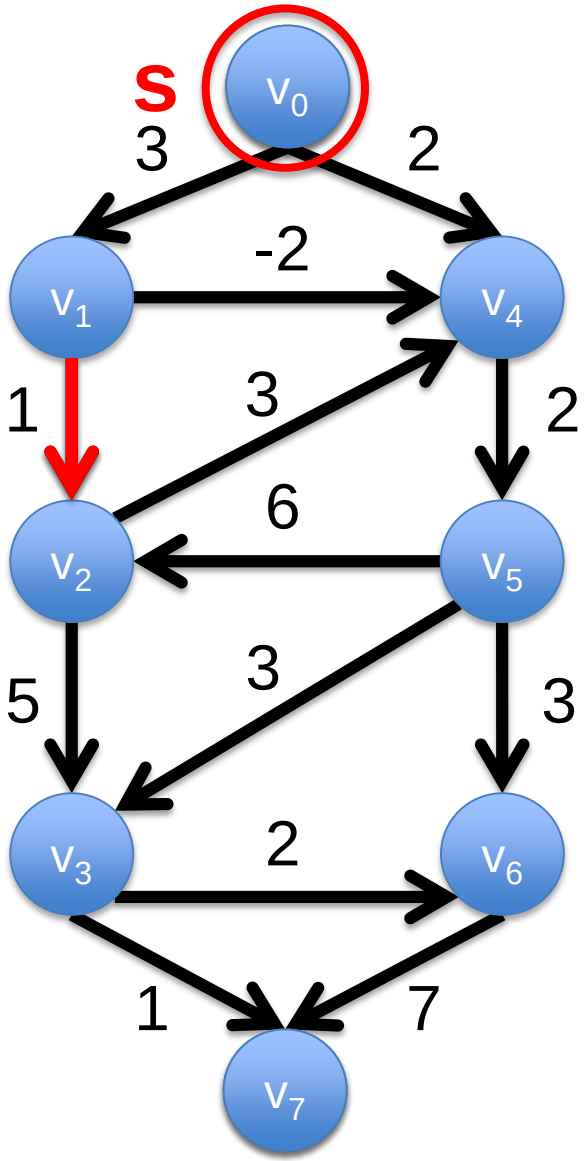
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	1	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0			v_1			

Bellman-Ford Method



$k = k + 1 = 1$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

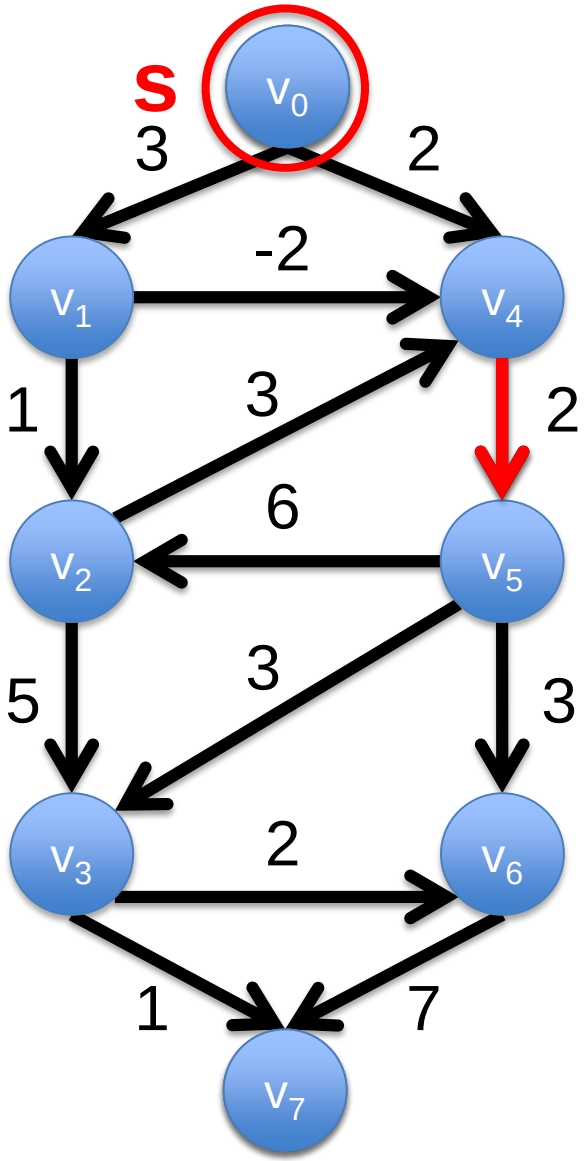
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1		v_1			

Bellman-Ford Method



$k = k + 1 = 1$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

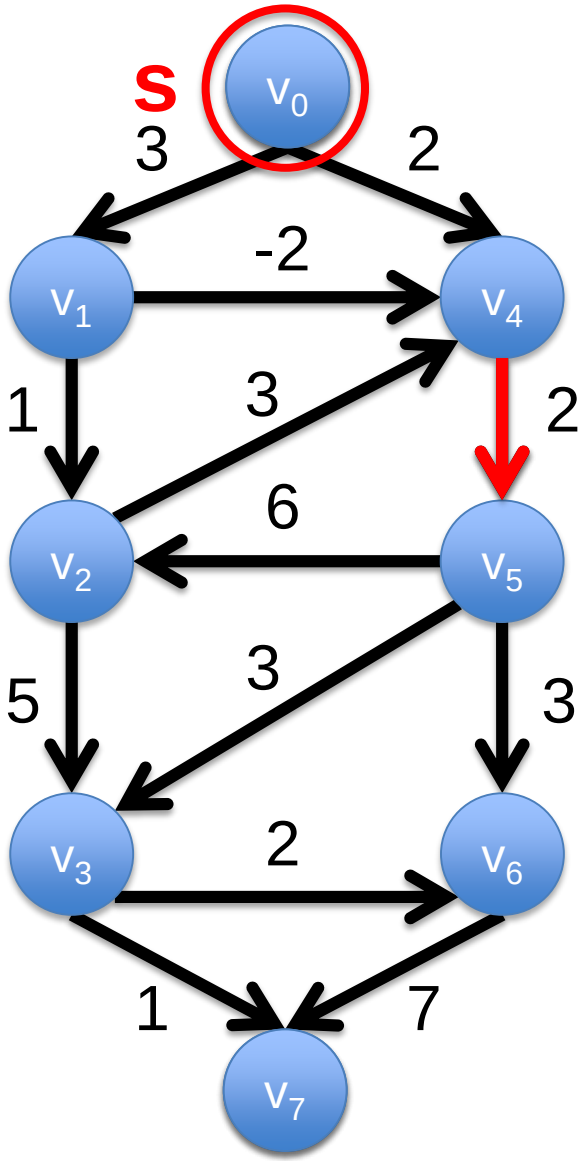
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	∞	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1		v_1			

Bellman-Ford Method



$k = k + 1 = 1$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	∞	∞	2	∞	∞	∞

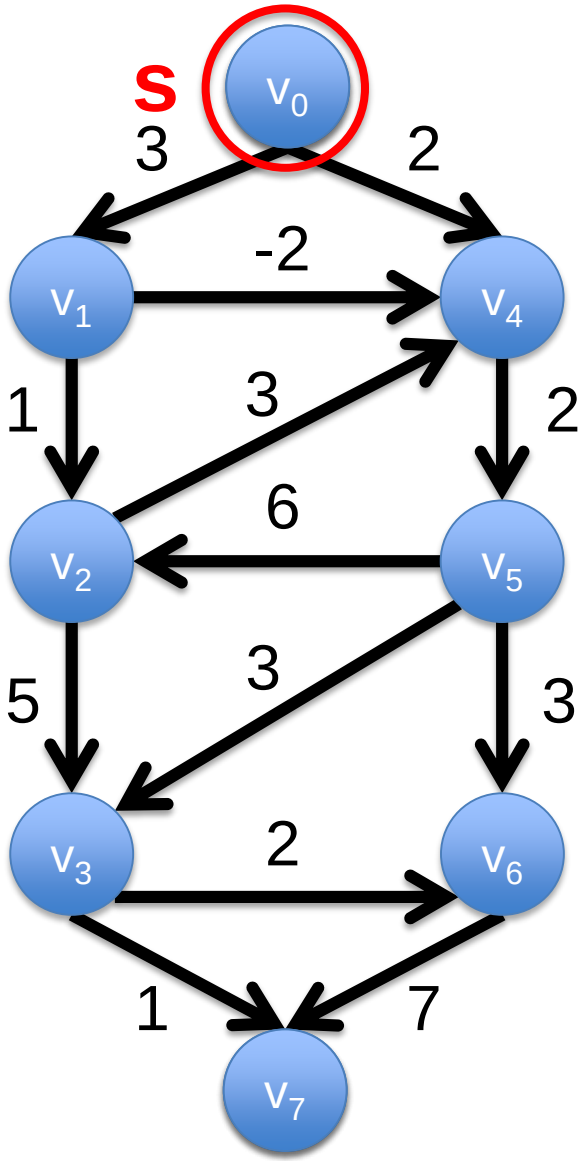
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1		v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

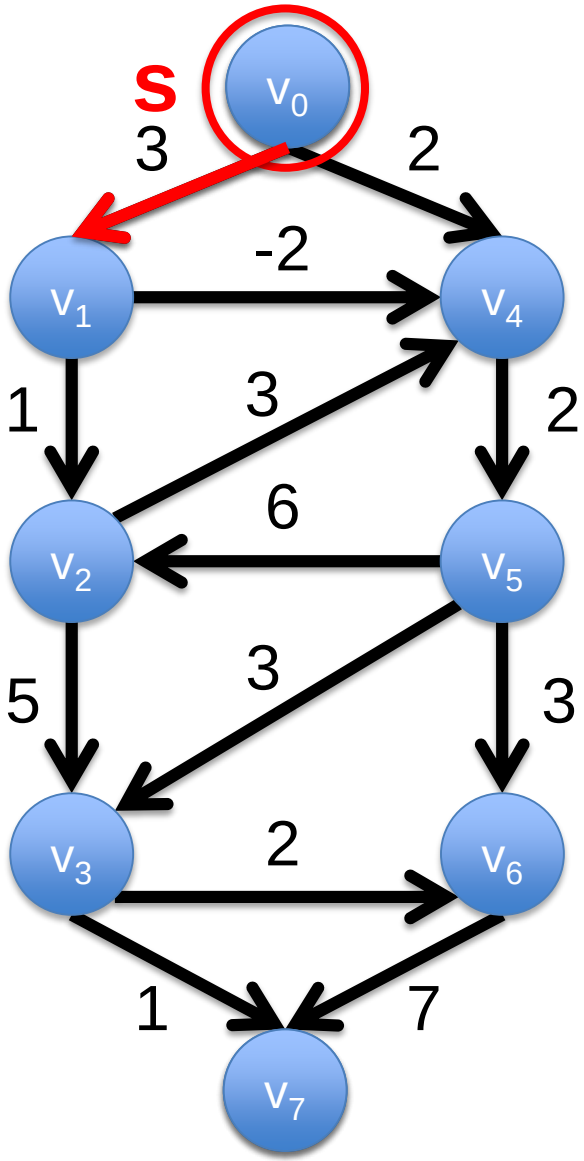
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1		v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

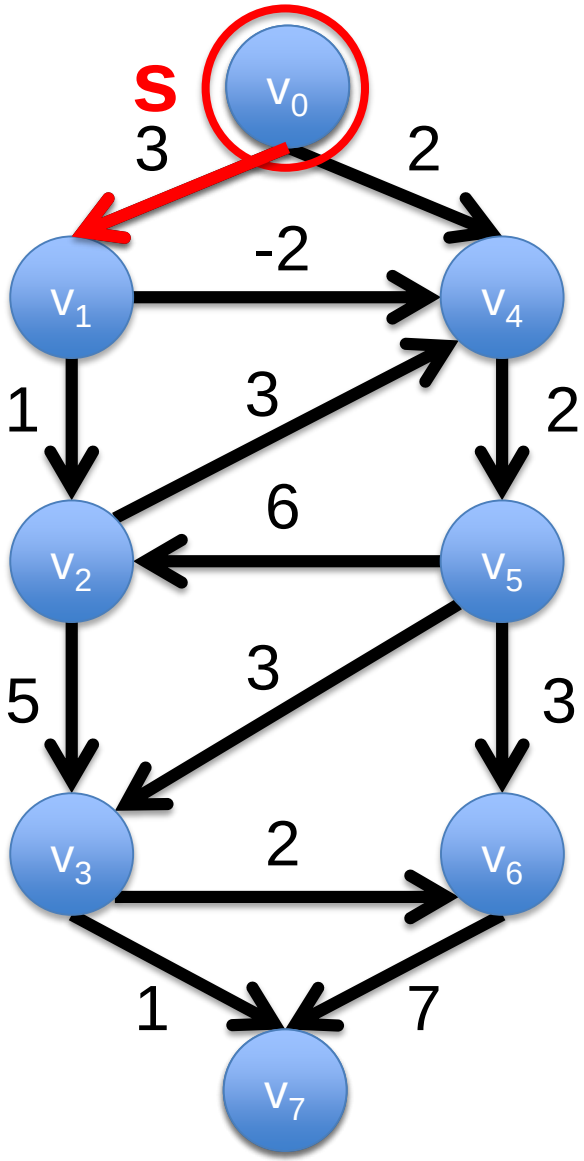
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1		v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

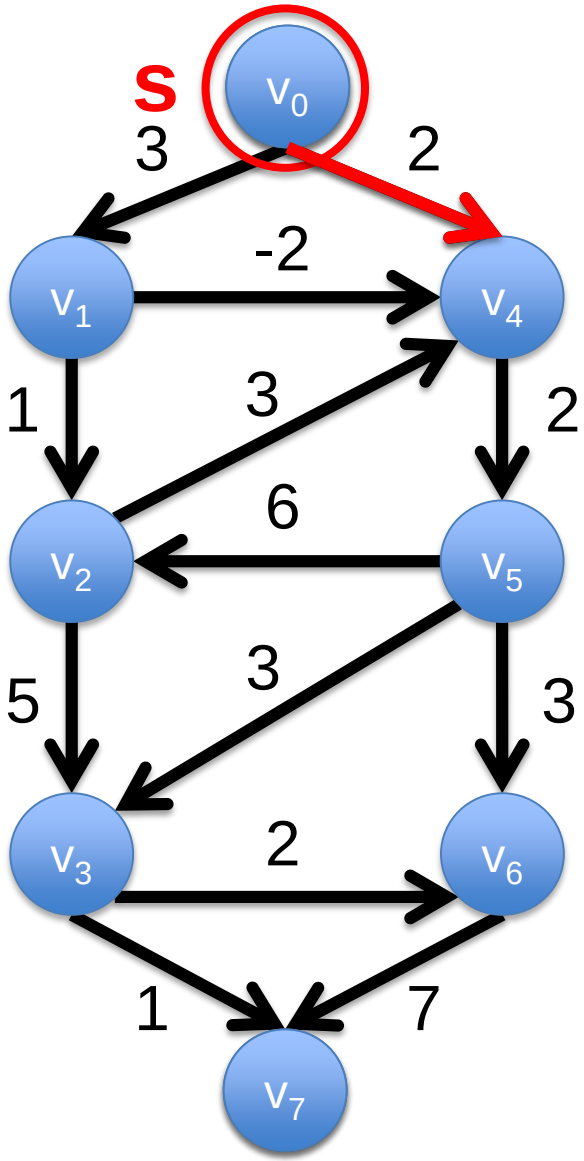
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1		v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

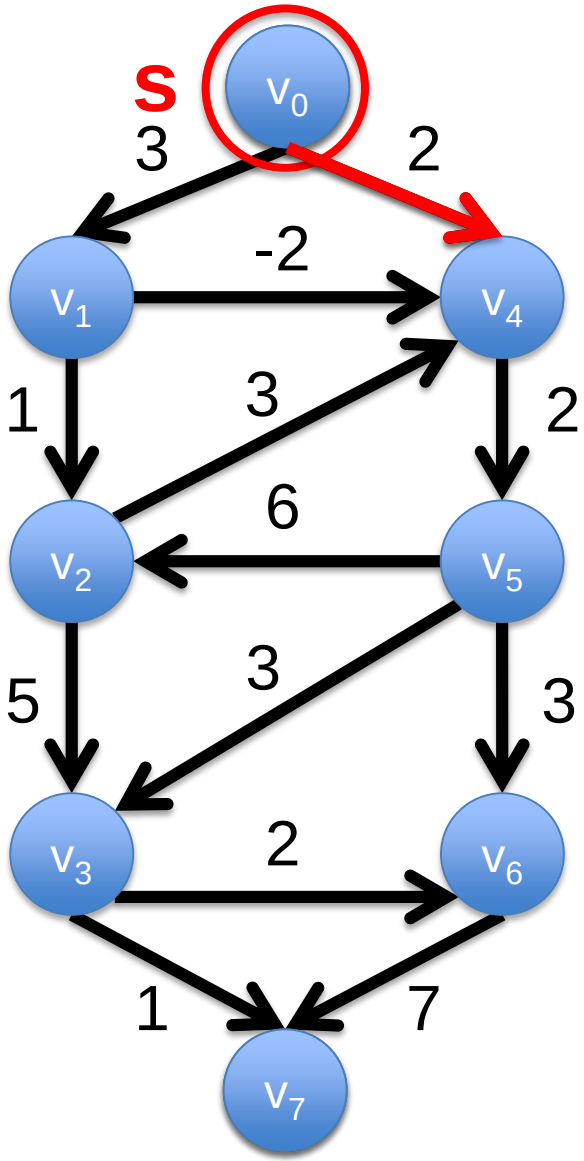
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1		v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

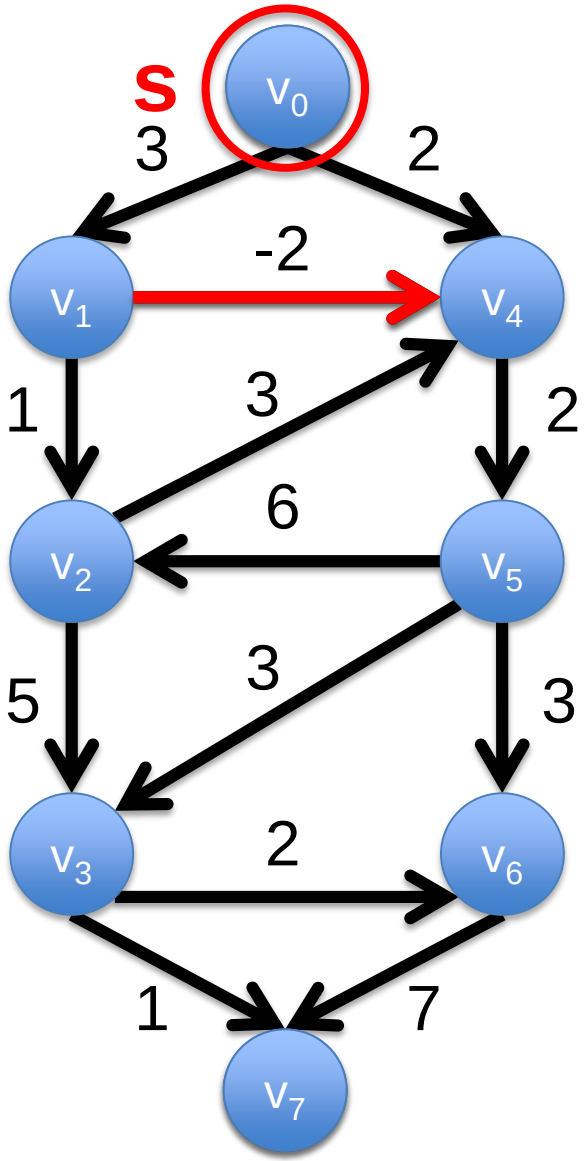
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1		v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

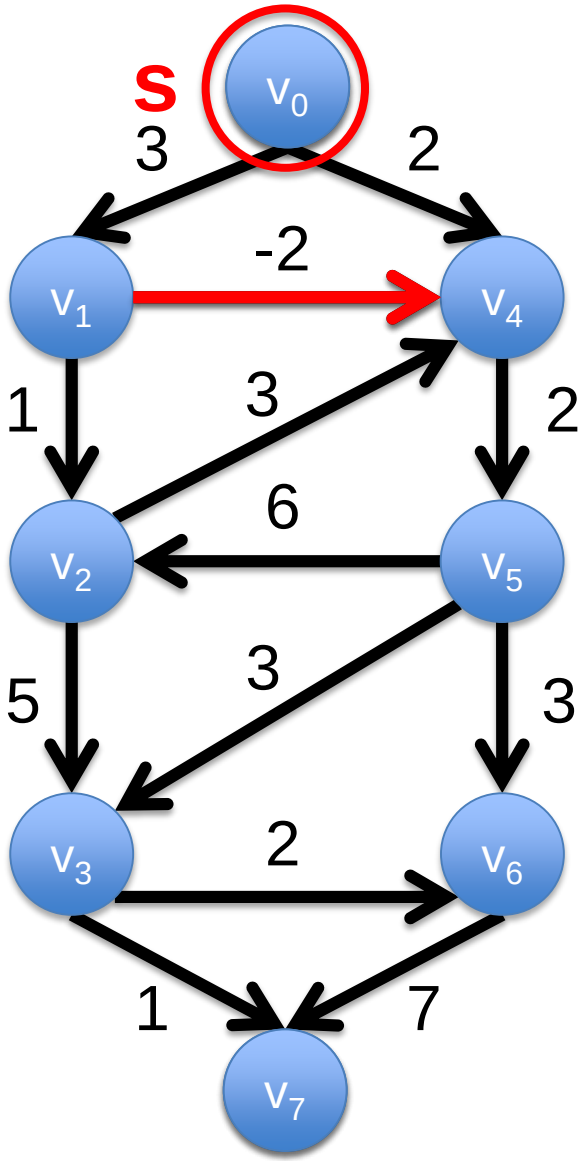
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1		v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

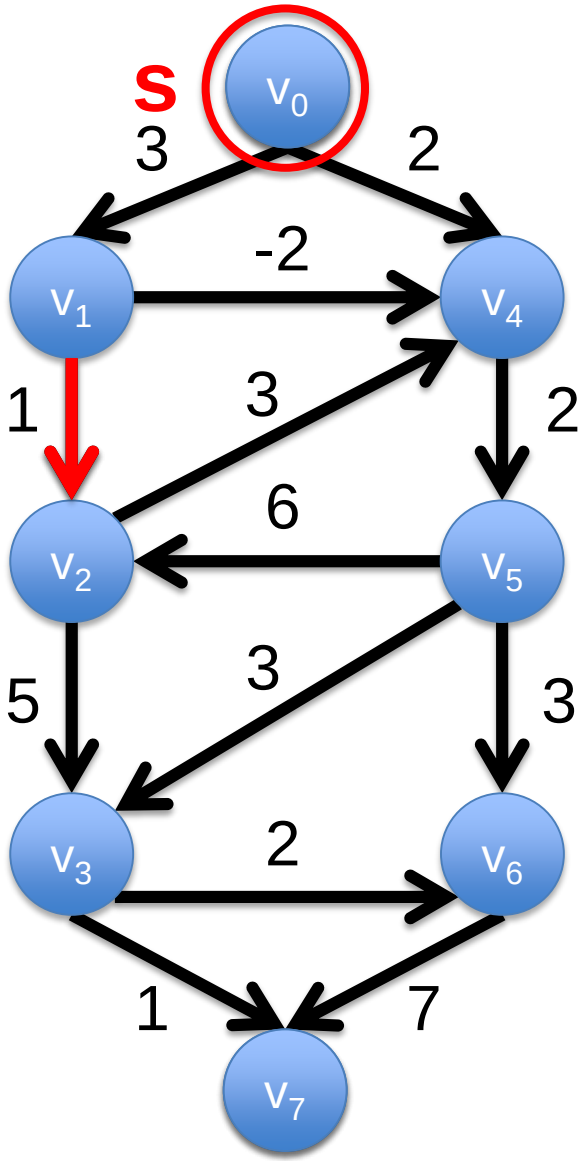
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1		v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

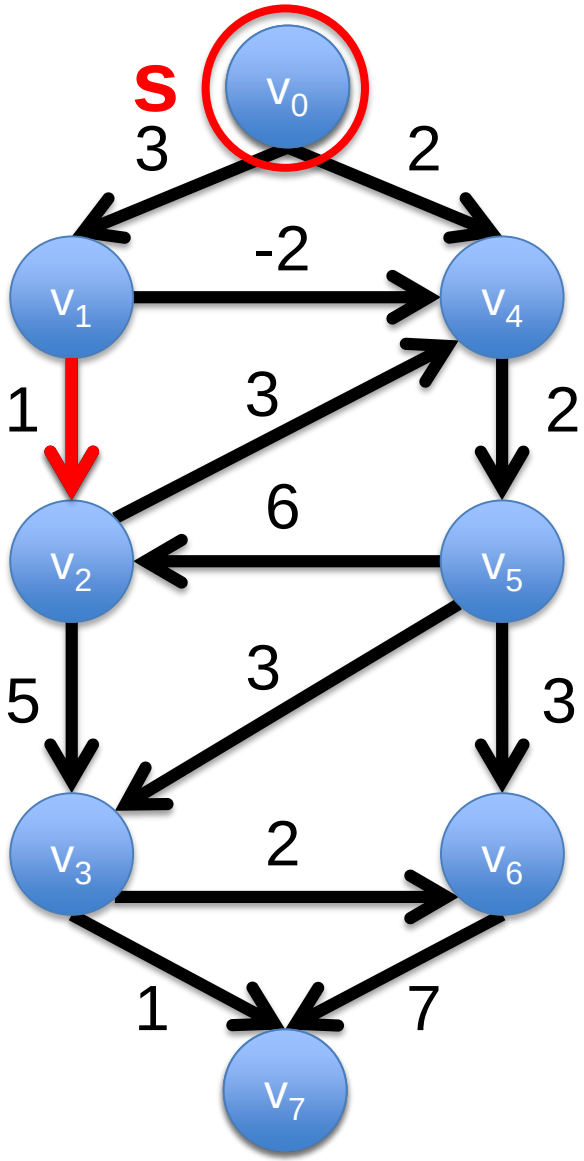
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1		v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

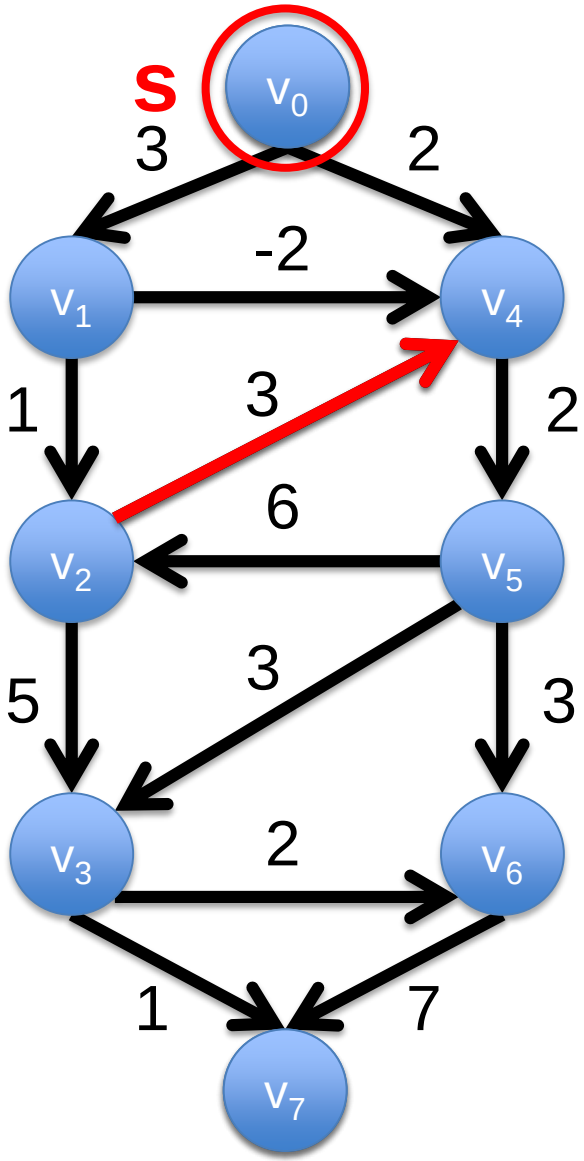
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1		v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

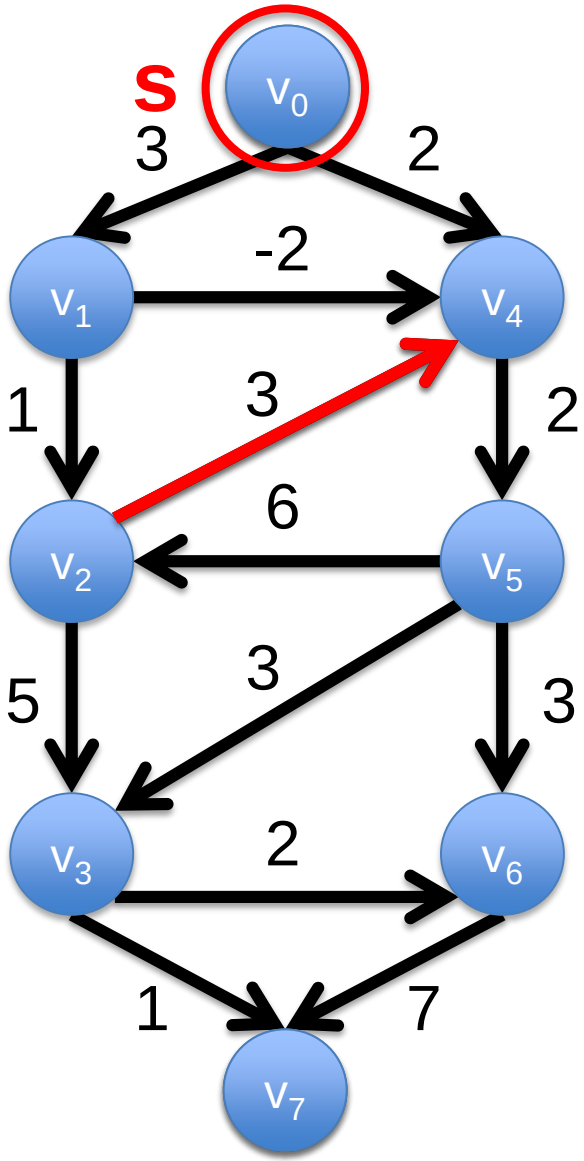
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1		v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

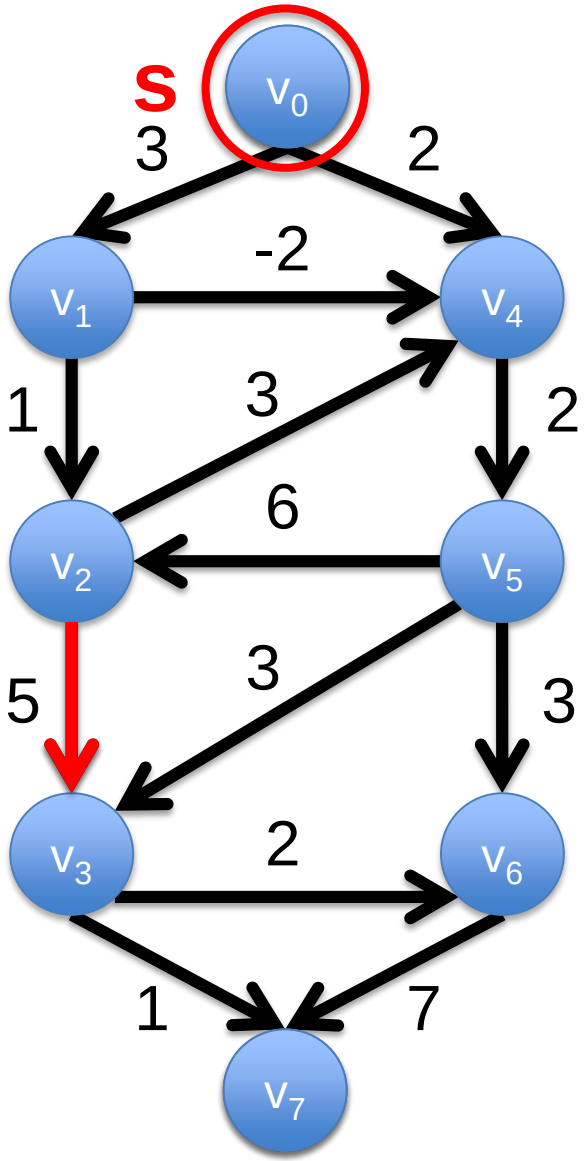
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1		v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

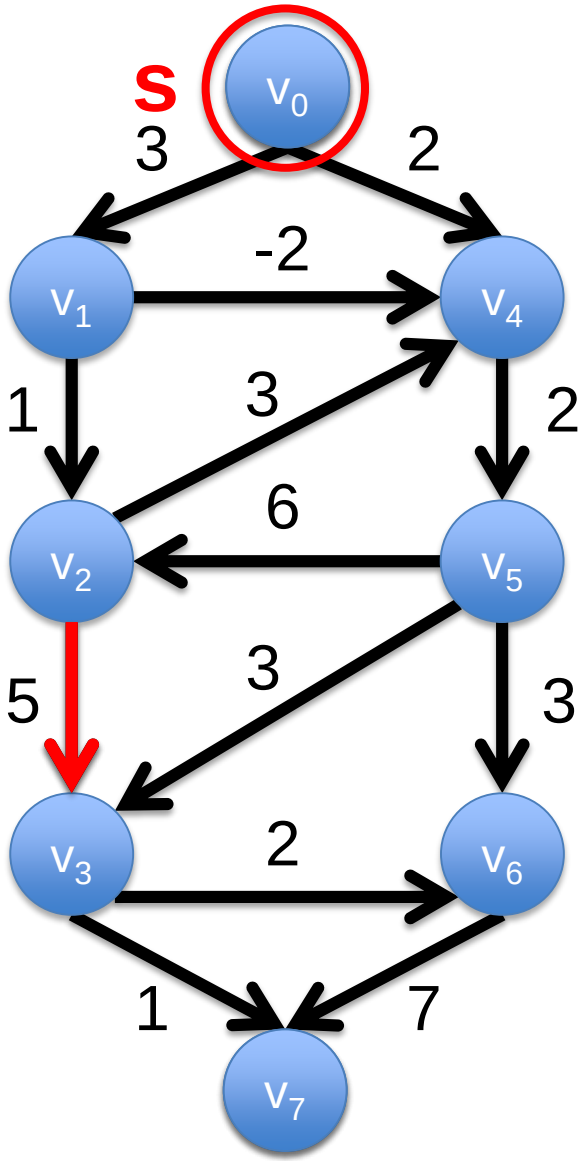
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1		v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

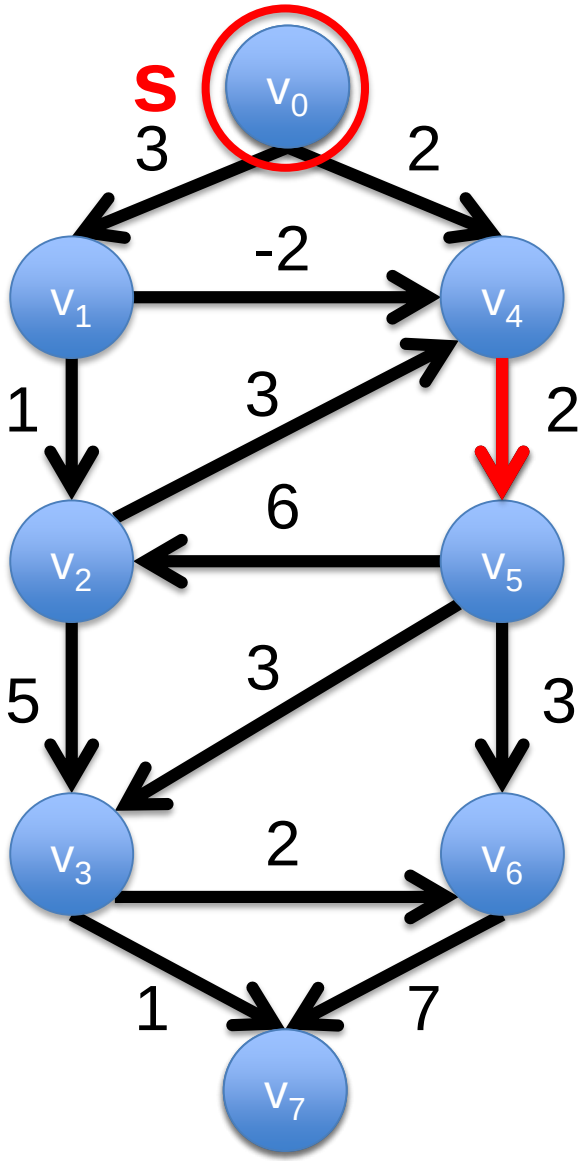
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	9	1	4	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1	v_2	v_1	v_4		

Bellman-Ford Method



k = k +1 = 2

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

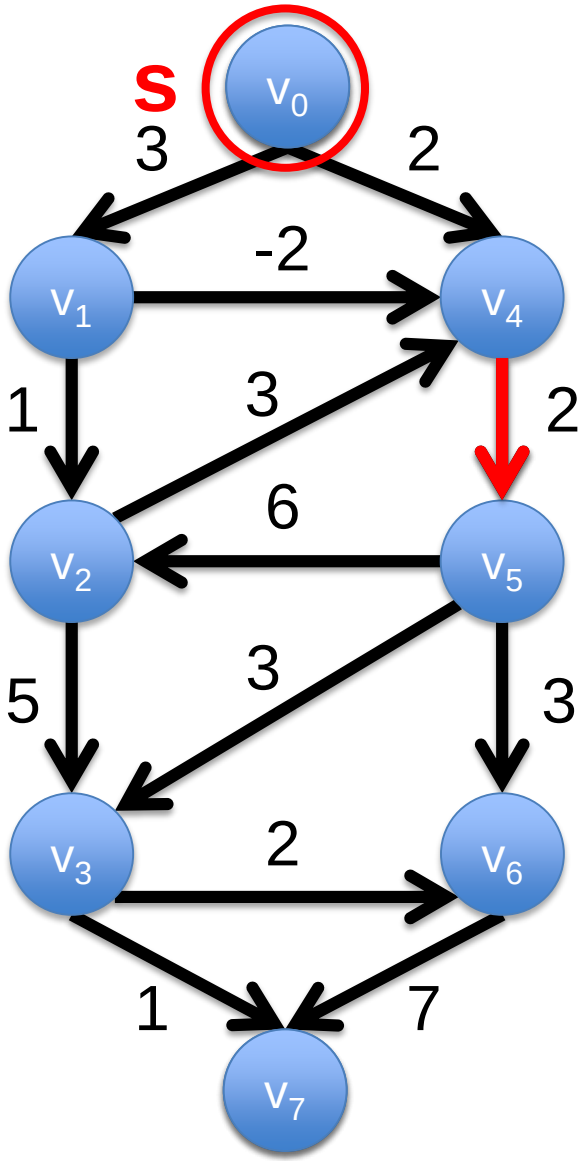
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	9	1	4	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1	v_2	v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

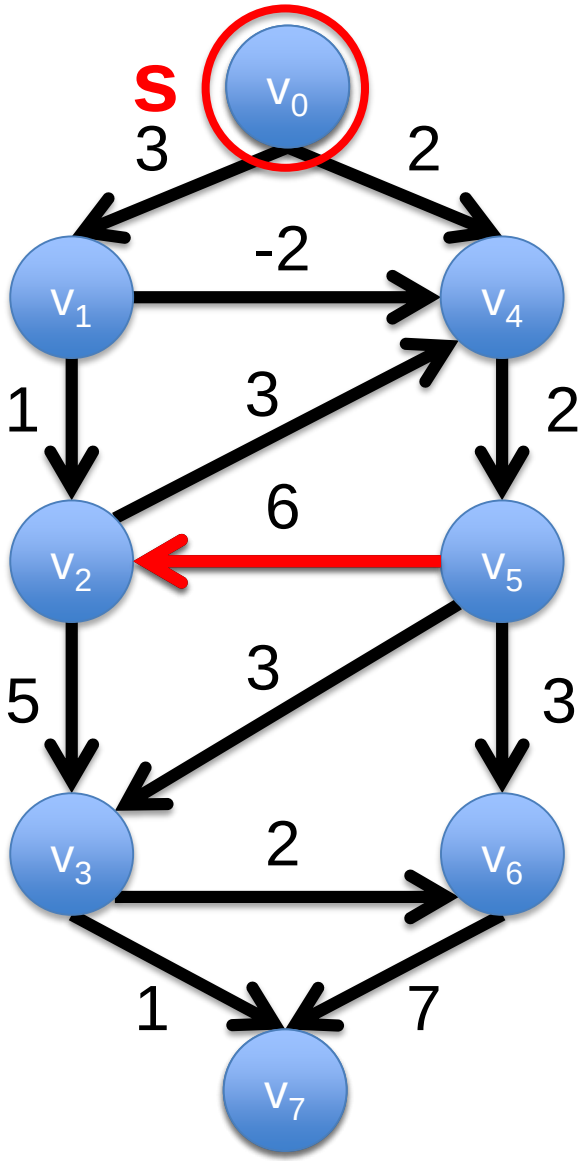
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	9	1	3	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1	v_2	v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

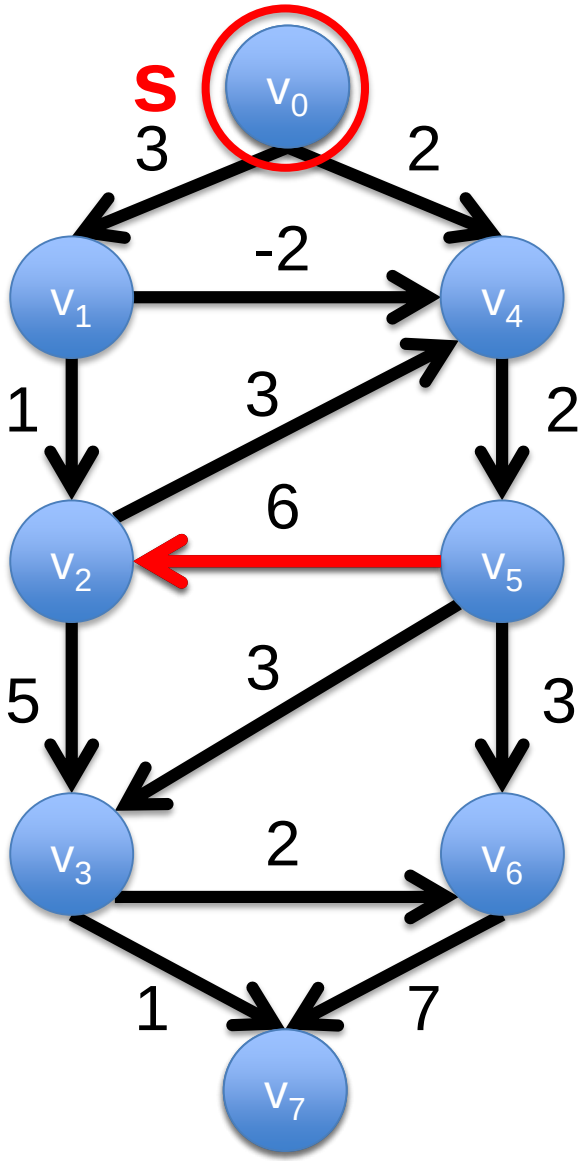
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	9	1	3	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1	v_2	v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

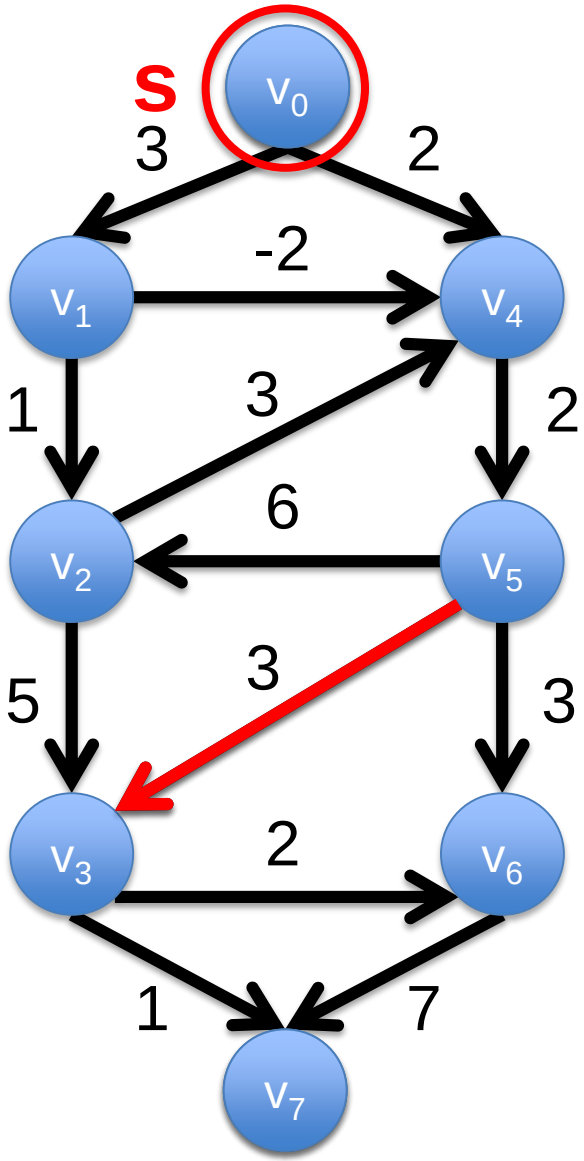
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	9	1	3	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1	v_2	v_1	v_4		

Bellman-Ford Method



k = k +1 = 2

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

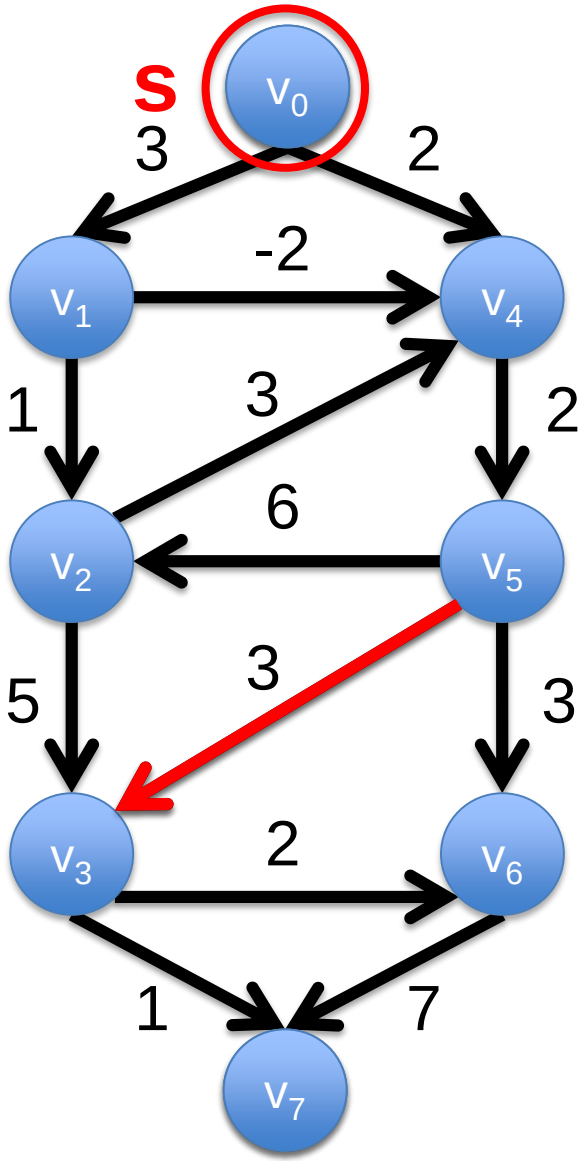
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	9	1	3	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1	v_2	v_1	v_4		

Bellman-Ford Method



k = k +1 = 2

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

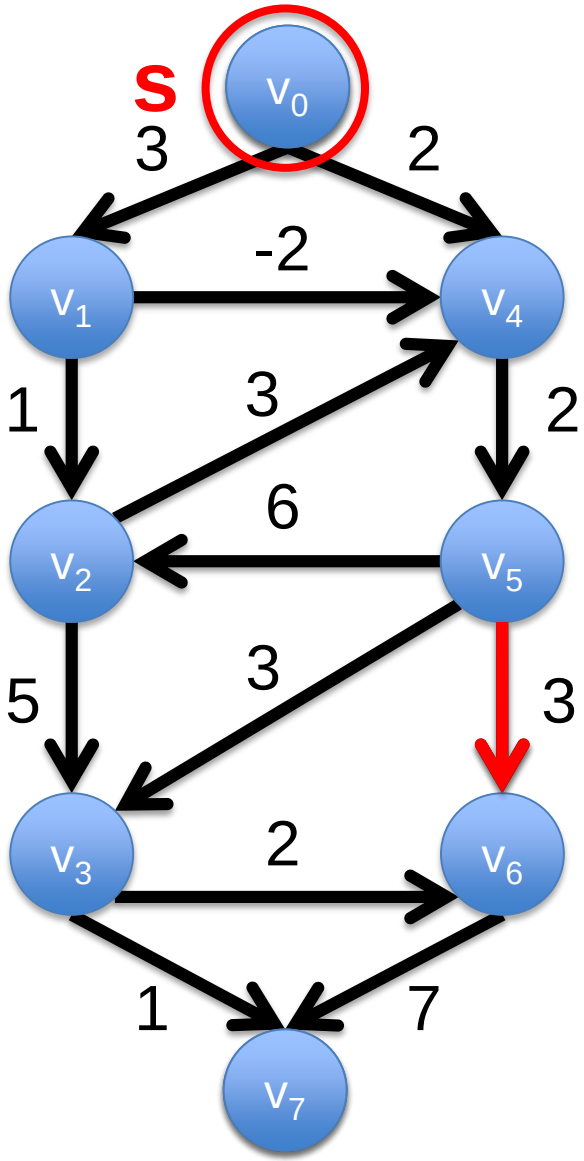
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	1	3	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1	v_5	v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

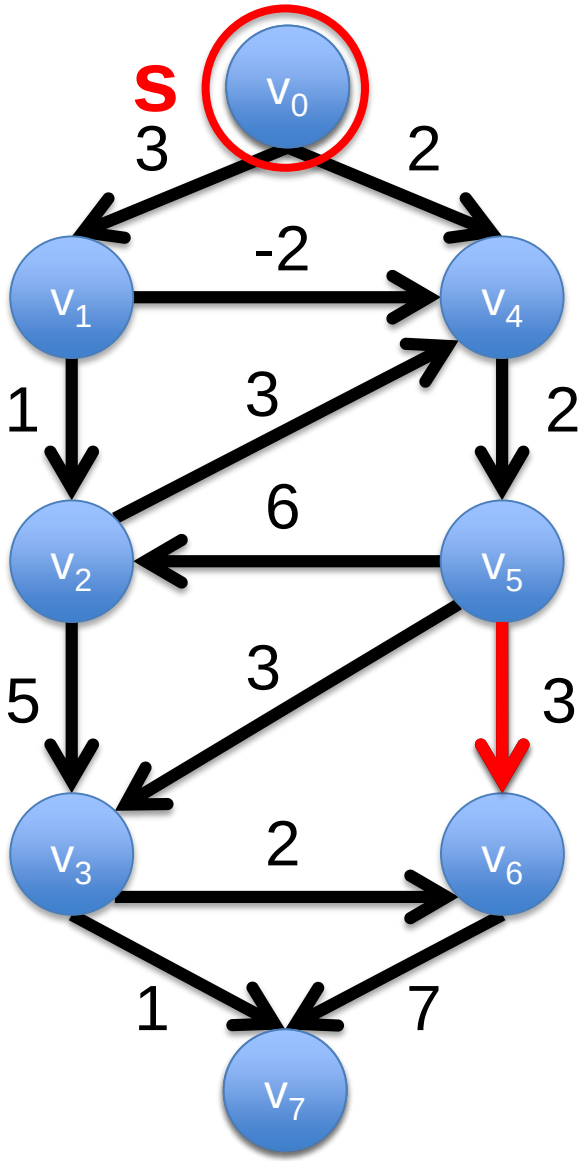
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	1	3	∞	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1	v_5	v_1	v_4		

Bellman-Ford Method



$k = k + 1 = 2$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	∞	1	4	∞	∞

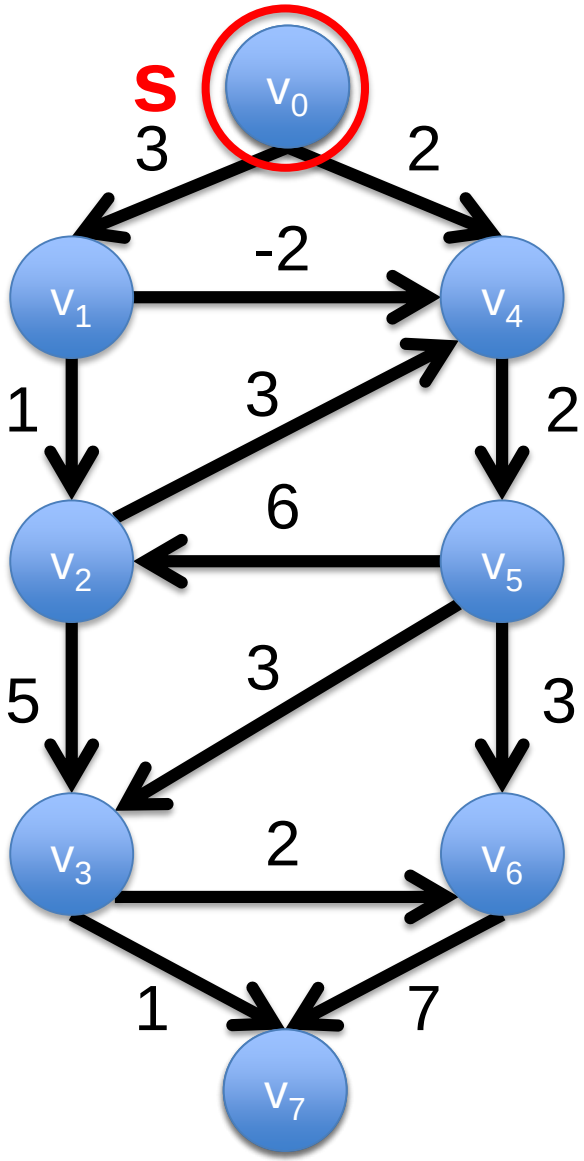
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	1	3	7	∞

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1	v_5	v_1	v_4	v_5	

Bellman-Ford Method



$k = k + 1 = 3$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	7	1	3	7	∞

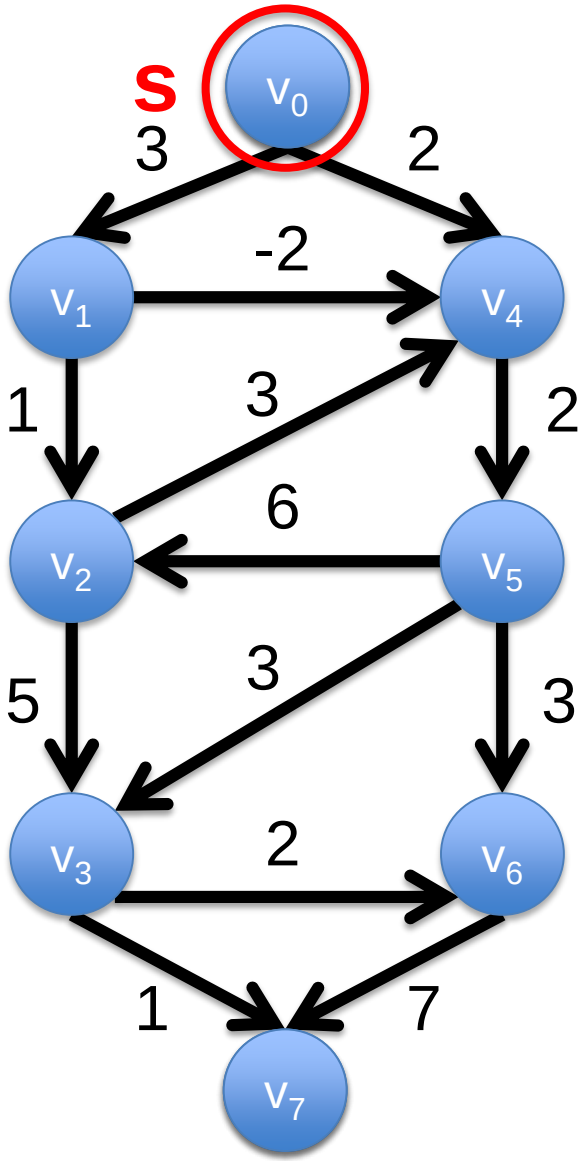
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	6	1	3	6	8

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1	v_5	v_1	v_4	v_5	v_3

Bellman-Ford Method



$k = k + 1 = 4$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	6	1	3	6	8

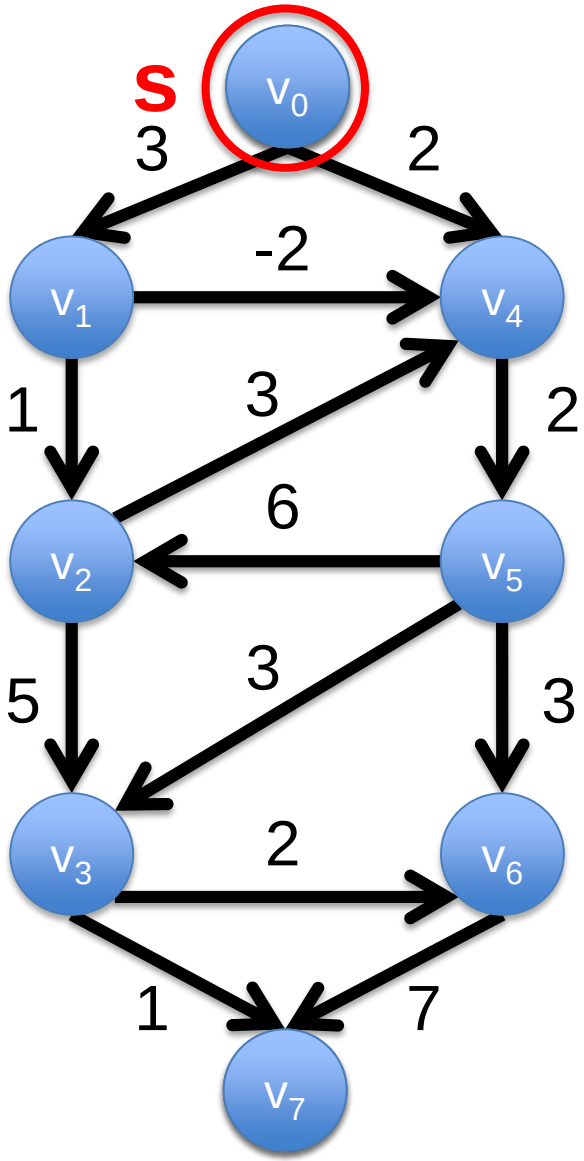
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	6	1	3	6	7

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1	v_5	v_1	v_4	v_5	v_3

Bellman-Ford Method



$k = k + 1 = 5$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	6	1	3	6	7

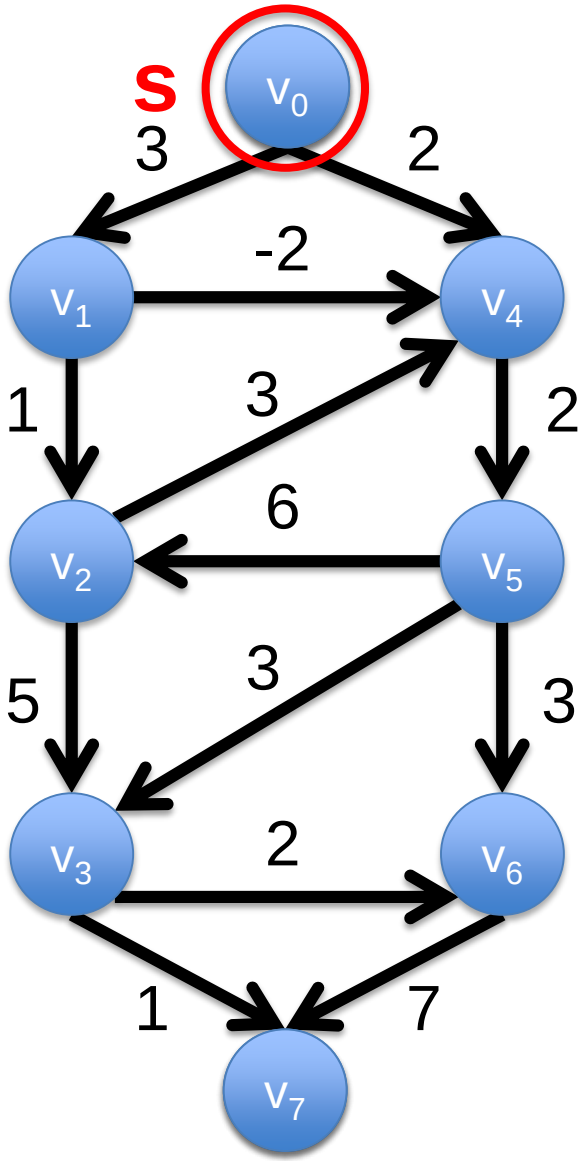
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	6	1	3	6	7

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1	v_5	v_1	v_4	v_5	v_3

Bellman-Ford Method



$k = k + 1 = 6$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	6	1	3	6	7

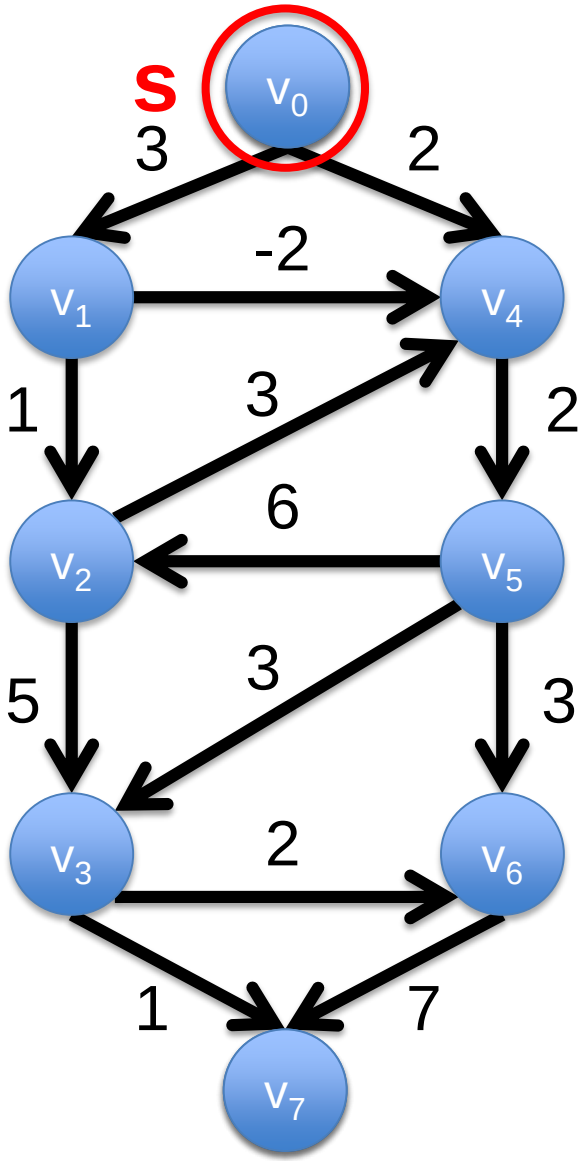
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	6	1	3	6	7

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1	v_5	v_1	v_4	v_5	v_3

Bellman-Ford Method



$k = k + 1 = 7$

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	6	1	3	6	7

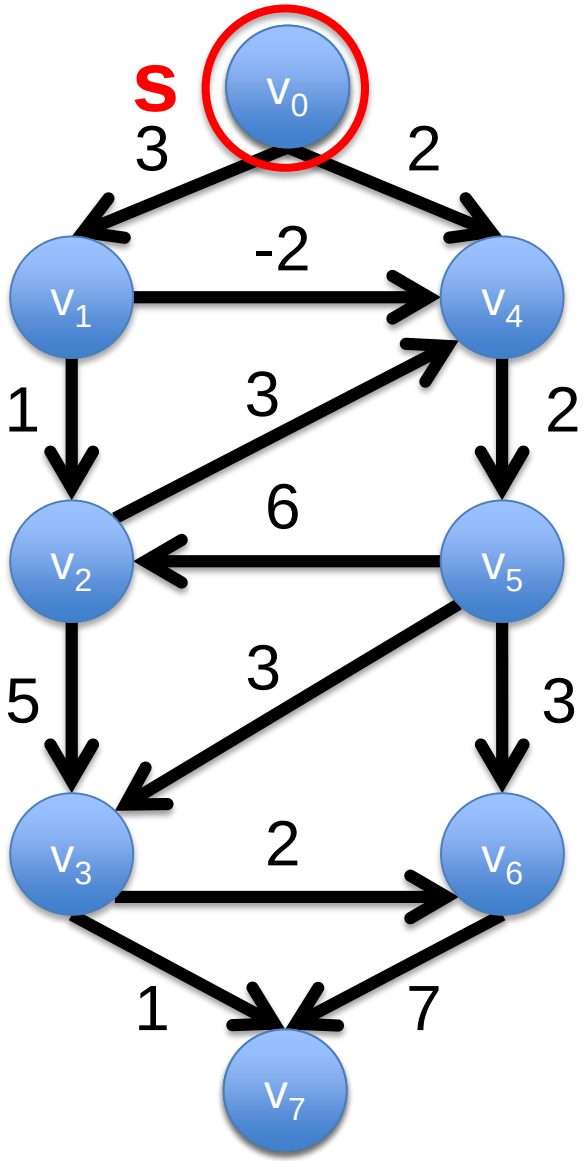
d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	6	1	3	6	7

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1	v_5	v_1	v_4	v_5	v_3

Bellman-Ford Method



$k = k + 1 = 8$

Stop after 'n' iterations

d_k

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	6	1	3	6	7

d_{k+1}

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
0	3	4	6	1	3	6	7

Predecessor

v_0	v_1	v_2	v_3	v_4	v_5	v_6	v_7
N/A	v_0	v_1	v_5	v_1	v_4	v_5	v_3

Summary

$k = 0.\ d_k(s) = 0$

While $(k < n)$

$d_{k+1}(t) = d_k(t)$

For each $(u,v) \in A$

$d_{k+1}(v) = \min \{ d_{k+1}(v), d_k(u) + l(u,v) \}$

End

$k = k + 1$

End

Time Complexity

O(n m)

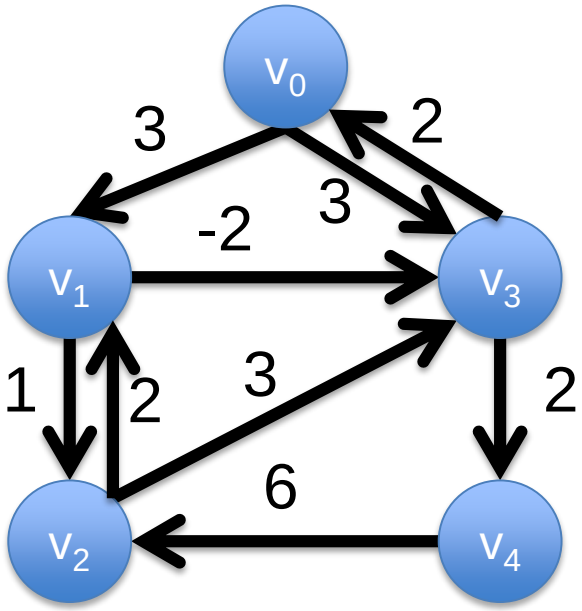
Outer iteration
Run ‘n’ times

Inner Iteration
O(m) operations

Outline

- Graph Preliminaries
- Complexity Preliminaries
- Shortest Path Algorithms
  - Breadth-first Search
  - Dijkstra's Method
  - Bellman-Ford Method
  - **Floyd-Warshall Method**

Floyd-Warshall Method



All-pairs shortest paths.

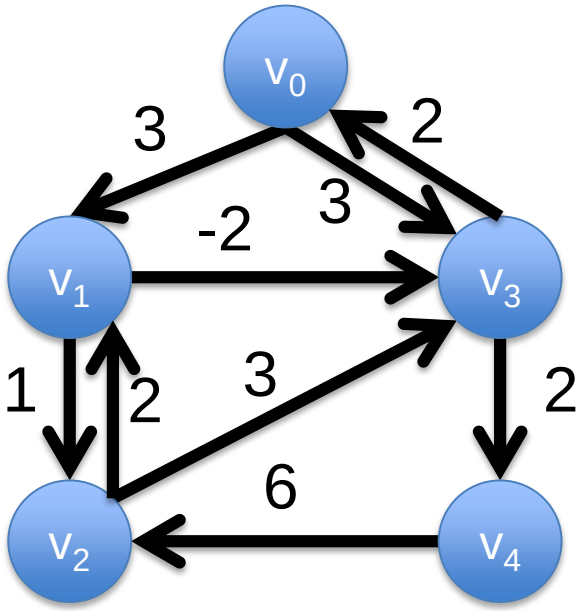
D = (V, A)

Assumption: No negative length directed circuits.

Length of path =
Σ arc lengths = Σ l(u,v)

|V| = n, |A| = m

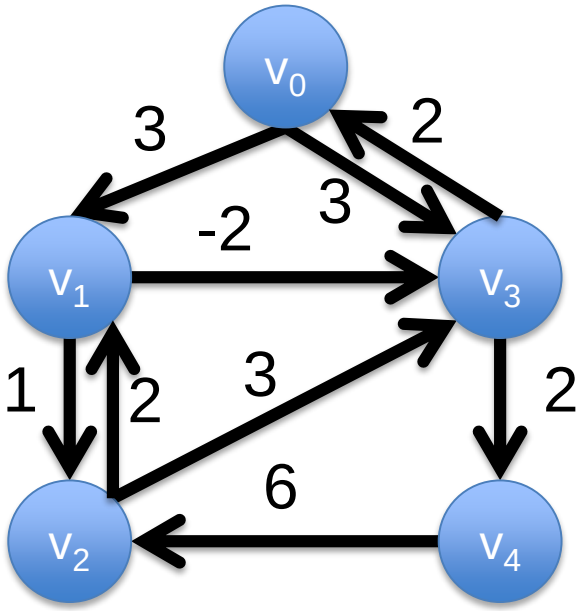
Bellman-Ford Method's Complexity



$O(n^2m)$

Apply Bellman-Ford 'n' times

Floyd-Warshall Method



Let $d_k(s,t)$ = minimum length s-t walk using only $\{s,t,v_0,\ldots,v_{k-1}\}$.

$d_{k+1}(s,t)$ = minimum of

$d_k(s,t)$,

$d_k(s,v_k) + d_k(v_k,t)$

Floyd-Warshall Method

$k = 0$. $d_k(i,j) = l(i,j)$ if $(i,j) \in A$, otherwise ∞

While ($k < n$)

For each $i \in V$

For each $j \in V$

$d_{k+1}(i,j) = \min \{ d_k(i,j), d_k(i,v_{k-1}) + d_k(v_{k-1},j)\}$

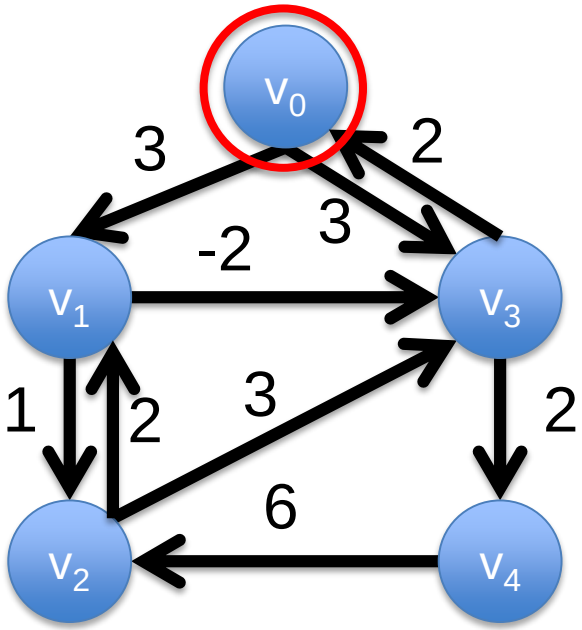
End

End

$k = k + 1$

End

Floyd-Warshall Method



$d_1(v_0,v_1)$ = minimum of $d_0(v_0,v_1)$, $d_0(v_0,v_0) + d_0(v_0,v_1)$

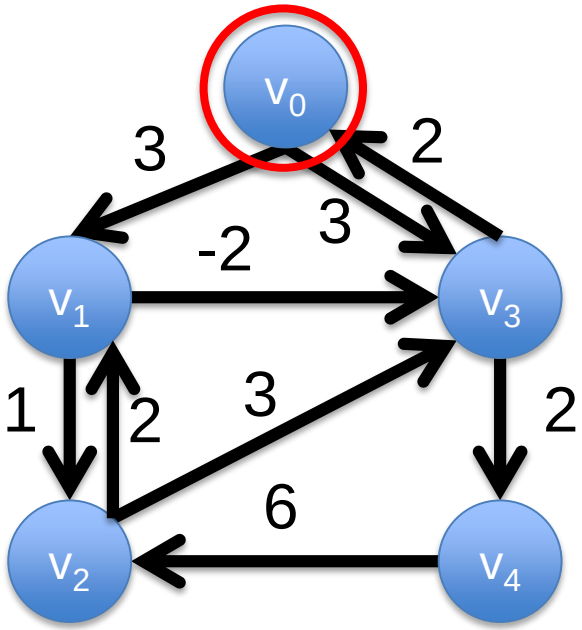
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0				
	0			
		0		
			0	
				0

Floyd-Warshall Method



$d_1(v_0,v_2)$ = minimum of $d_0(v_0,v_2)$, $d_0(v_0,v_0) + d_0(v_0,v_2)$

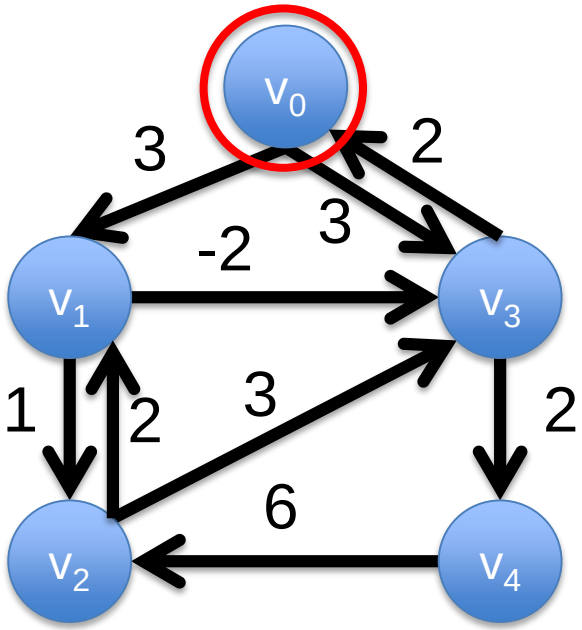
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3			
	0			
		0		
			0	
				0

Floyd-Warshall Method



$d_1(v_0,v_3)$ = minimum of $d_0(v_0,v_3)$, $d_0(v_0,v_0) + d_0(v_0,v_3)$

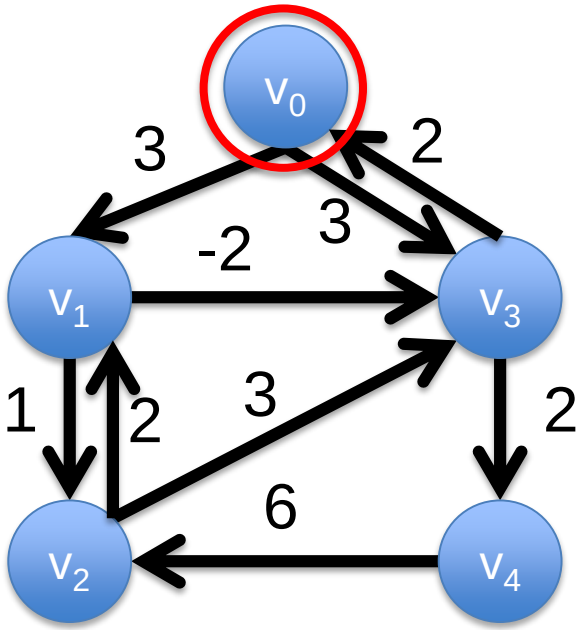
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞		
	0			
		0		
			0	
				0

Floyd-Warshall Method



d_0

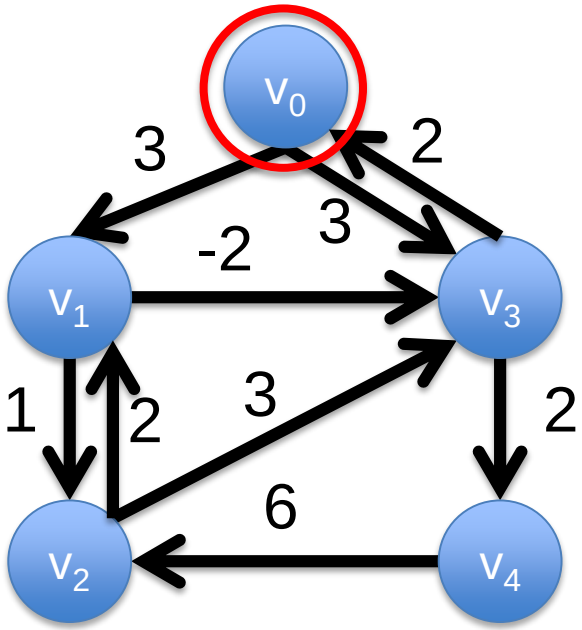
0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	
	0			
		0		
			0	
				0

$d_1(v_0,v_4)$ = minimum of $d_0(v_0,v_4)$, $d_0(v_0,v_0) + d_0(v_0,v_4)$

Floyd-Warshall Method



$d_1(v_1,v_0)$ = minimum of $d_0(v_1,v_0)$, $d_0(v_1,v_0) + d_0(v_0,v_0)$

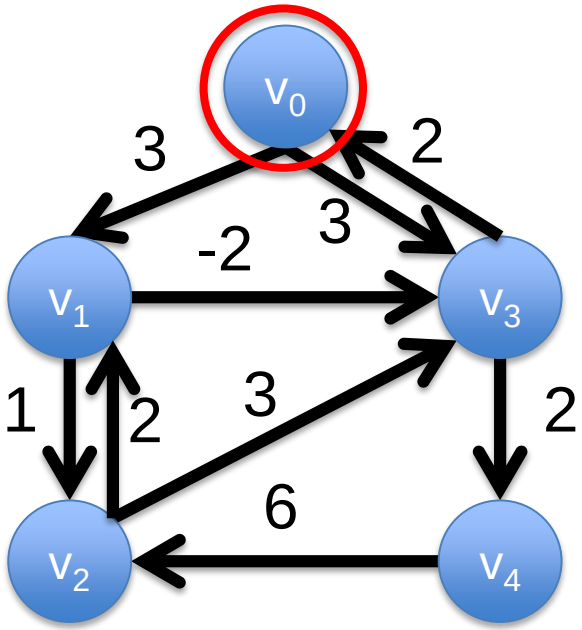
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
	0			
		0		
			0	
				0

Floyd-Warshall Method



$d_1(v_1,v_2)$ = minimum of $d_0(v_1,v_2)$, $d_0(v_1,v_0) + d_0(v_0,v_2)$

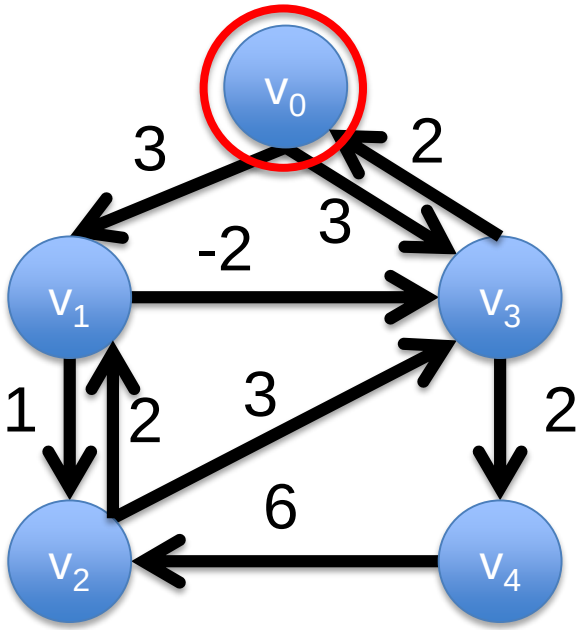
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0			
		0		
			0	
				0

Floyd-Warshall Method



$d_1(v_1,v_3)$ = minimum of $d_0(v_1,v_3)$, $d_0(v_1,v_0) + d_0(v_0,v_3)$

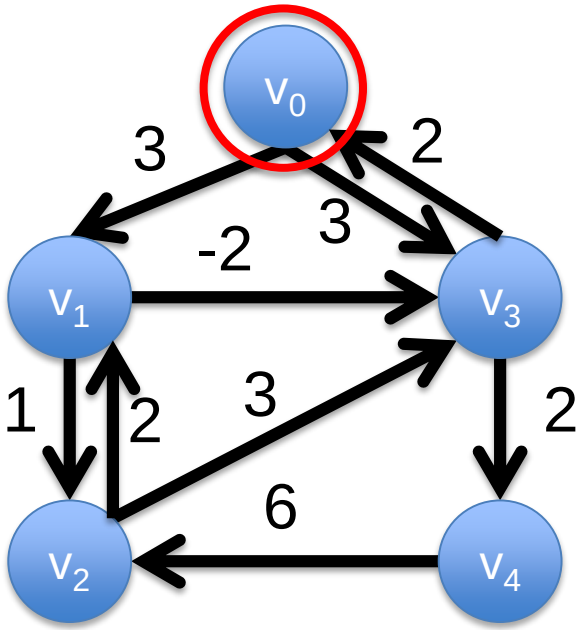
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0	1		
		0		
			0	
				0

Floyd-Warshall Method



$d_1(v_1,v_4)$ = minimum of $d_0(v_1,v_4)$, $d_0(v_1,v_0) + d_0(v_0,v_4)$

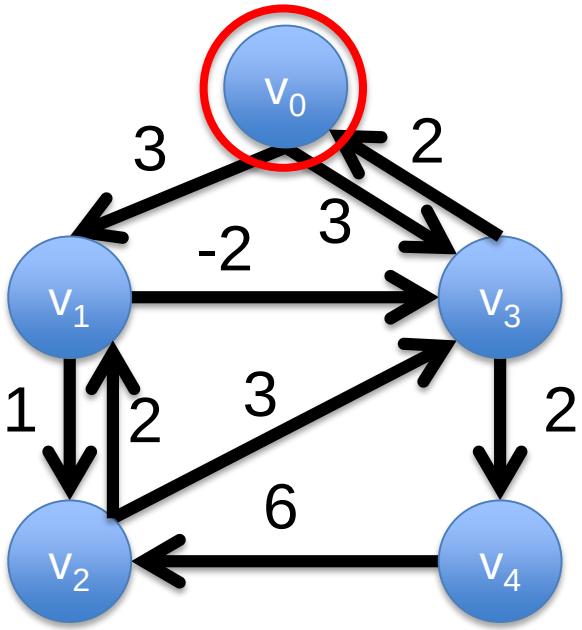
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0	1	-2	
		0		
			0	
				0

Floyd-Warshall Method



d_0

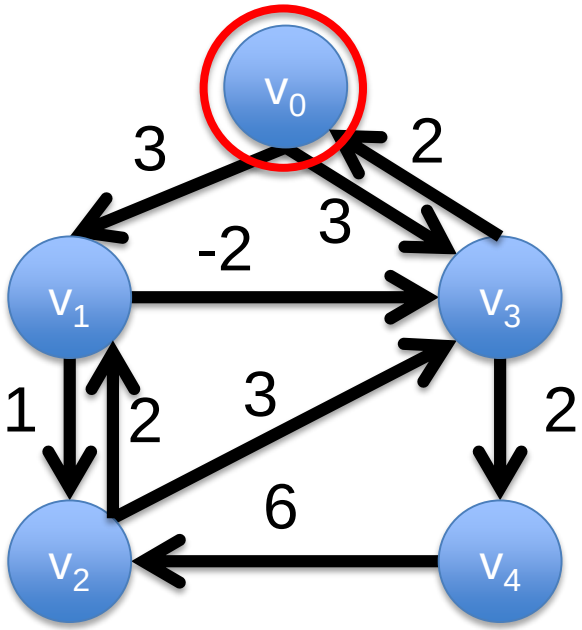
0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0	1	-2	∞
		0		
			0	
				0

$d_1(v_2,v_0)$ = minimum of $d_0(v_2,v_0)$, $d_0(v_2,v_0) + d_0(v_0,v_0)$

Floyd-Warshall Method



$d_1(v_2,v_1)$ = minimum of $d_0(v_2,v_1)$, $d_0(v_2,v_0) + d_0(v_0,v_1)$

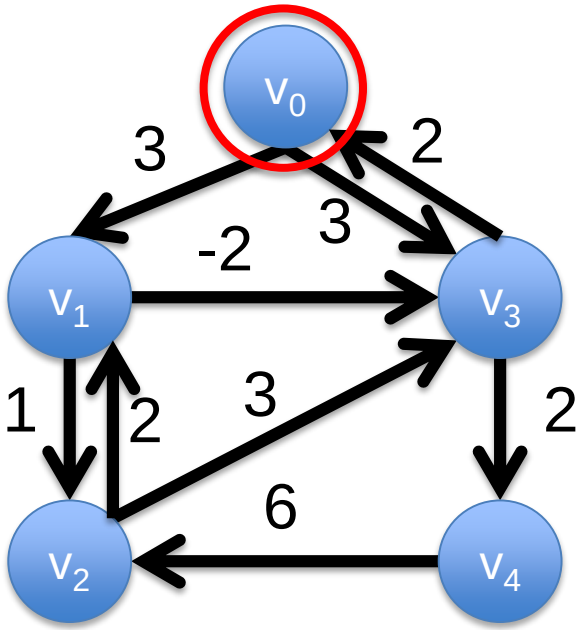
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞		0		
			0	
				0

Floyd-Warshall Method



$d_1(v_2,v_3)$ = minimum of $d_0(v_2,v_3)$, $d_0(v_2,v_0) + d_0(v_0,v_3)$

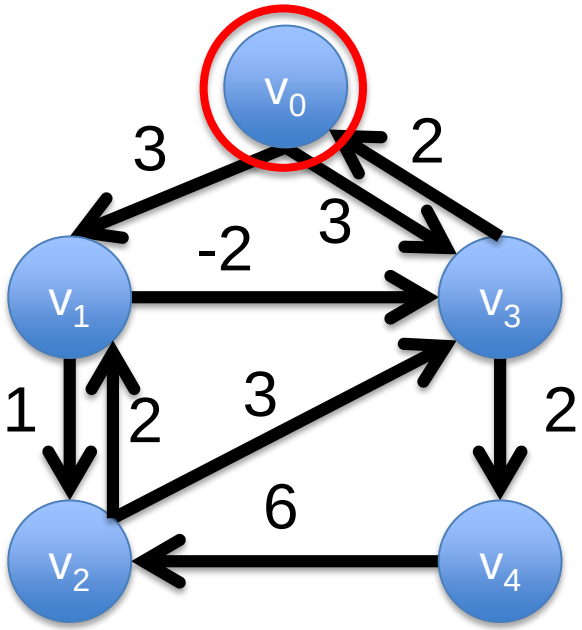
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0		
			0	
				0

Floyd-Warshall Method



$d_1(v_2,v_4)$ = minimum of $d_0(v_2,v_4)$, $d_0(v_2,v_0) + d_0(v_0,v_4)$

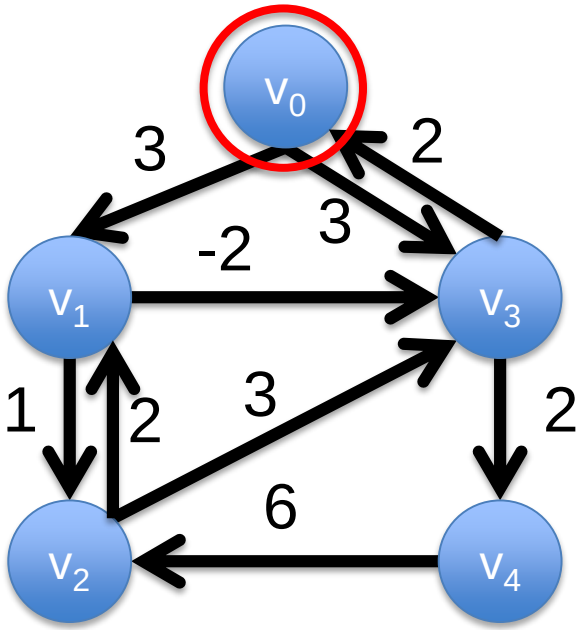
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	
			0	
				0

Floyd-Warshall Method



d_0

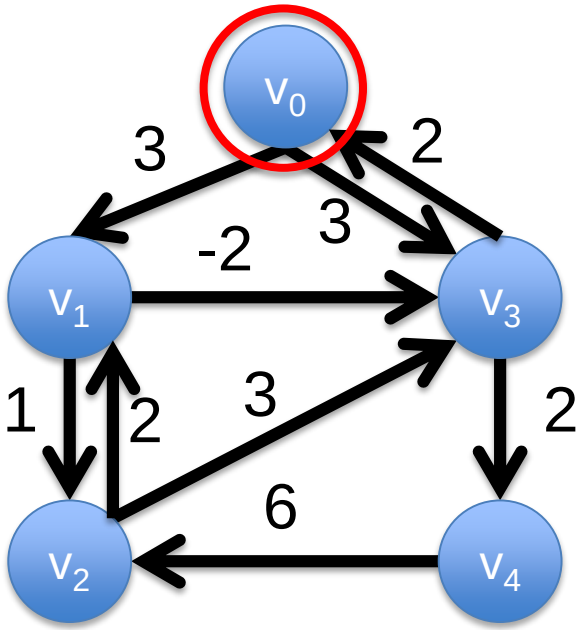
0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
			0	
				0

$d_1(v_3,v_0)$ = minimum of $d_0(v_3,v_0)$, $d_0(v_3,v_0) + d_0(v_0,v_0)$

Floyd-Warshall Method



$d_1(v_3,v_1)$ = minimum of $d_0(v_3,v_1)$, $d_0(v_3,v_0) + d_0(v_0,v_1)$

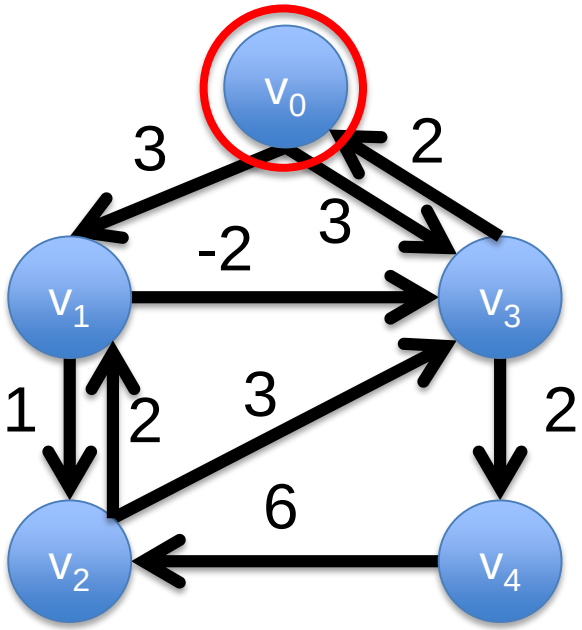
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2			0	
				0

Floyd-Warshall Method



$d_1(v_3,v_2)$ = minimum of $d_0(v_3,v_2)$, $d_0(v_3,v_0) + d_0(v_0,v_2)$

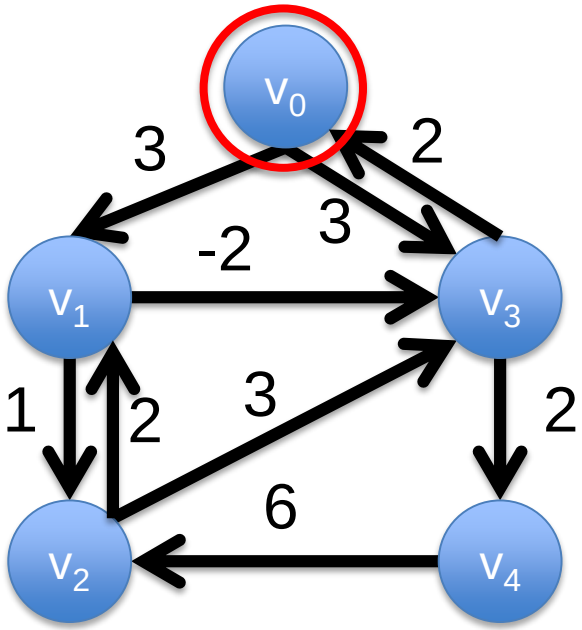
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5		0	
				0

Floyd-Warshall Method



$d_1(v_3,v_4)$ = minimum of $d_0(v_3,v_4)$, $d_0(v_3,v_0) + d_0(v_0,v_4)$

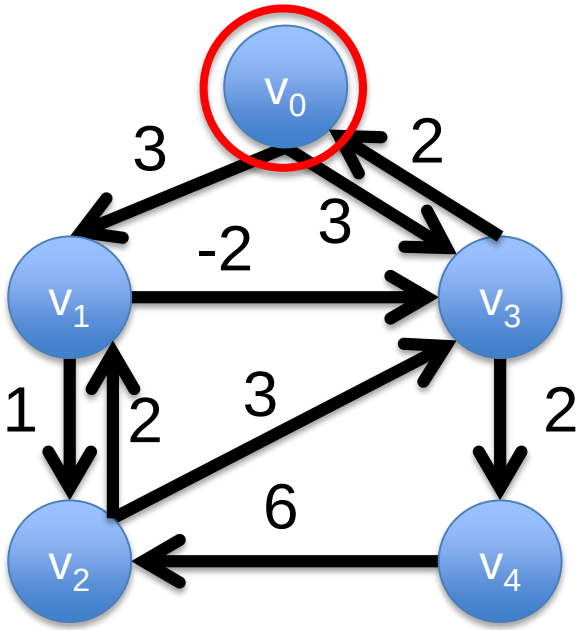
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	
				0

Floyd-Warshall Method



$d_1(v_4,v_0)$ = minimum of $d_0(v_4,v_0)$, $d_0(v_4,v_0) + d_0(v_0,v_0)$

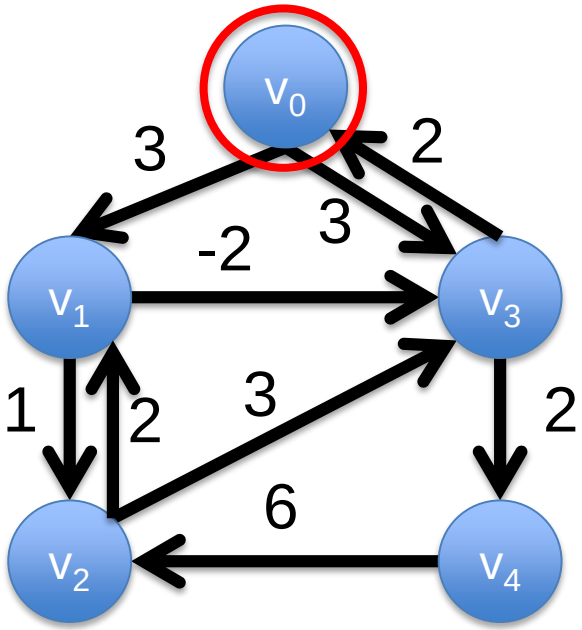
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
				0

Floyd-Warshall Method



$d_1(v_4,v_1)$ = minimum of $d_0(v_4,v_1)$, $d_0(v_4,v_0) + d_0(v_0,v_1)$

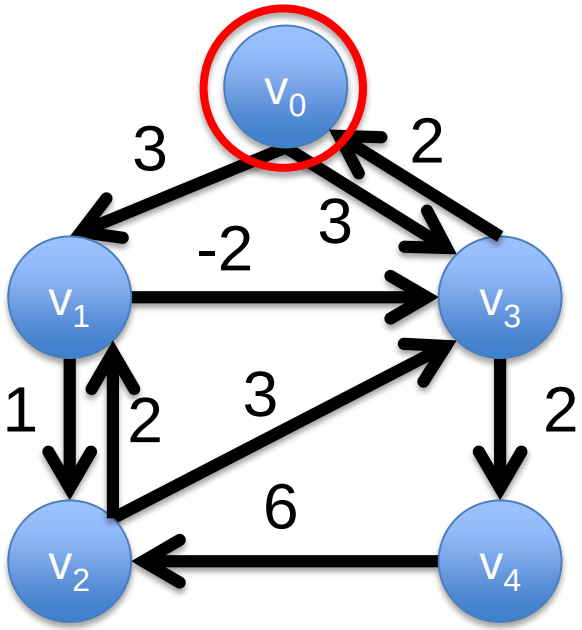
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞				0

Floyd-Warshall Method



$d_1(v_4,v_2)$ = minimum of $d_0(v_4,v_2)$, $d_0(v_4,v_0) + d_0(v_0,v_2)$

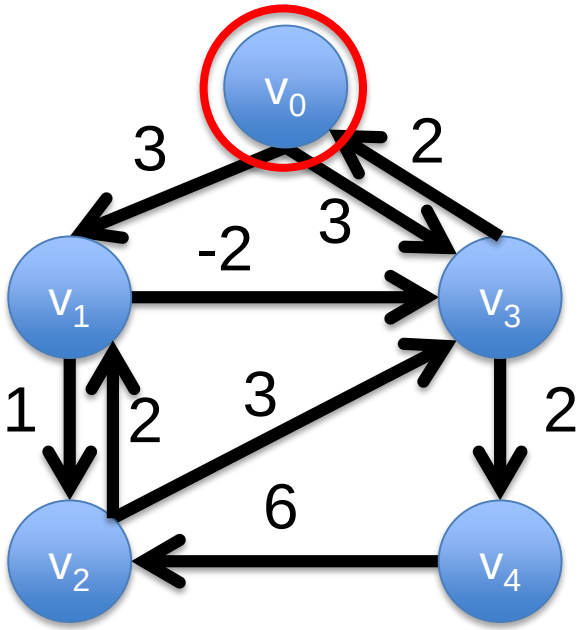
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞			0

Floyd-Warshall Method



$d_1(v_4,v_3)$ = minimum of $d_0(v_4,v_3)$, $d_0(v_4,v_0) + d_0(v_0,v_3)$

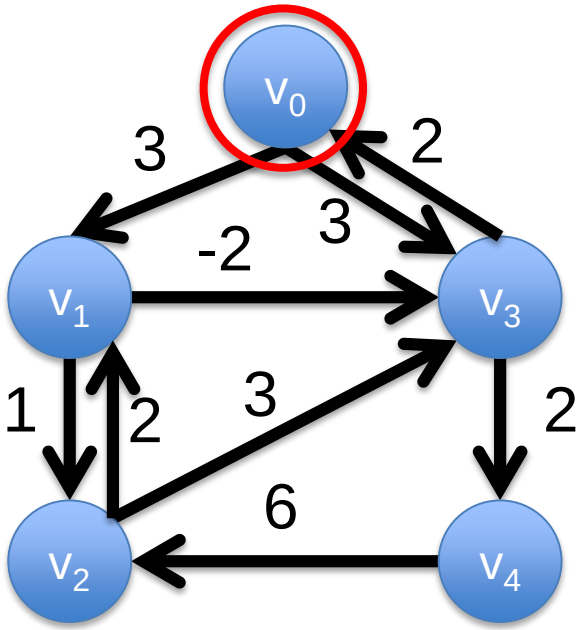
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6		0

Floyd-Warshall Method



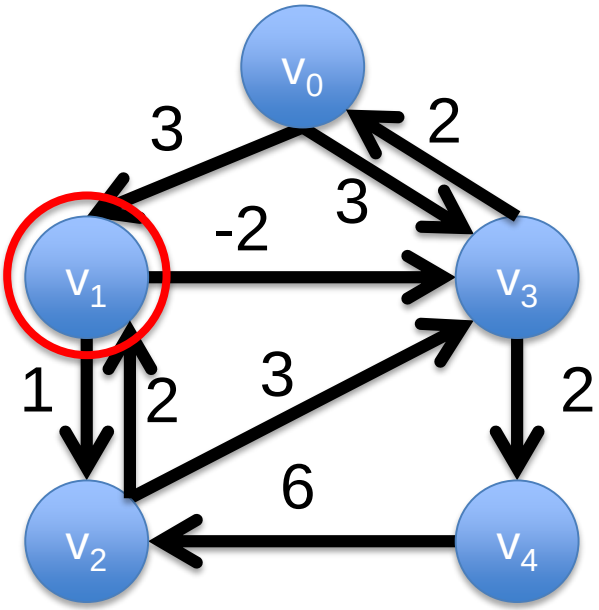
d_0

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	∞	∞	0	2
∞	∞	6	∞	0

d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

Floyd-Warshall Method



d_1

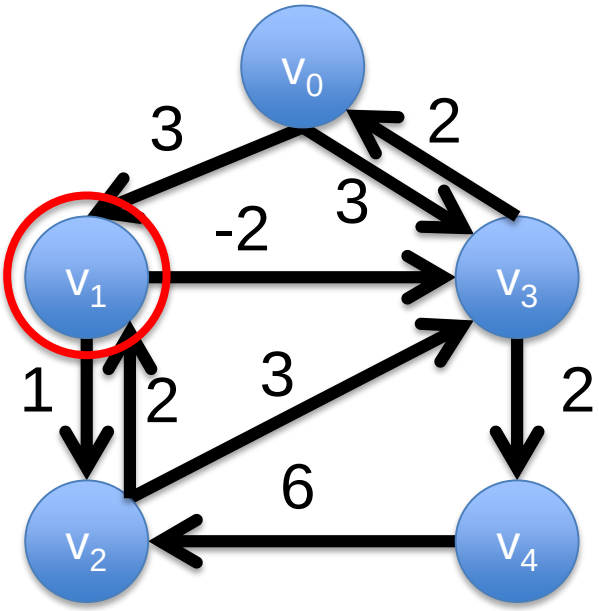
0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0				
	0			
		0		
			0	
				0

$d_2(v_0,v_1)$ = minimum of $d_1(v_0,v_1)$, $d_1(v_0,v_1) + d_1(v_1,v_0)$

Floyd-Warshall Method



$d_2(v_0,v_2)$ = minimum of $d_1(v_0,v_2)$, $d_1(v_0,v_1) + d_1(v_1,v_2)$

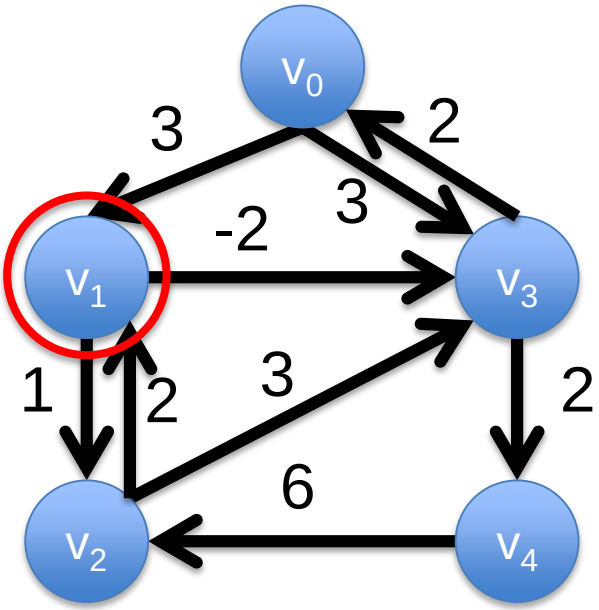
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3			
	0			
		0		
			0	
				0

Floyd-Warshall Method



$d_2(v_0,v_3)$ = minimum of $d_1(v_0,v_3)$, $d_1(v_0,v_1) + d_1(v_1,v_3)$

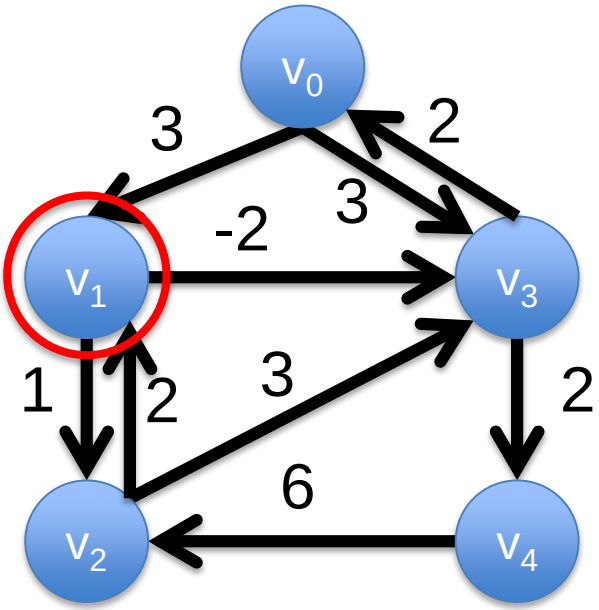
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4		
	0			
		0		
			0	
				0

Floyd-Warshall Method



$d_2(v_0,v_4)$ = minimum of $d_1(v_0,v_4)$, $d_1(v_0,v_1) + d_1(v_1,v_4)$

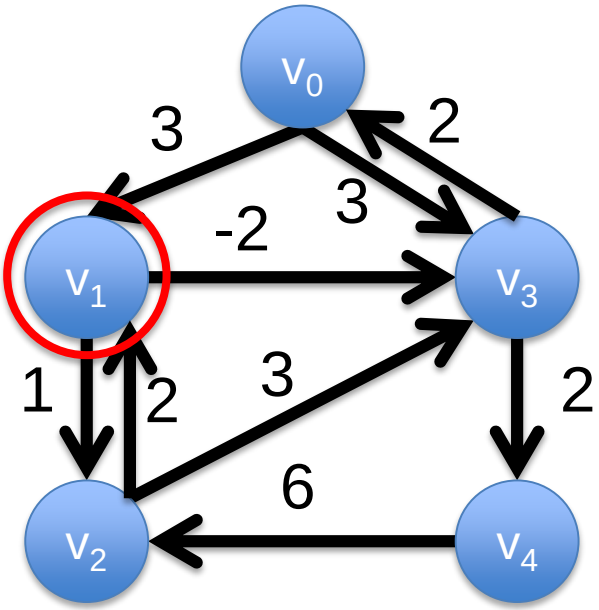
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	
	0			
		0		
			0	
				0

Floyd-Warshall Method



$d_2(v_1,v_0)$ = minimum of $d_1(v_1,v_0)$, $d_1(v_1,v_1) + d_1(v_1,v_0)$

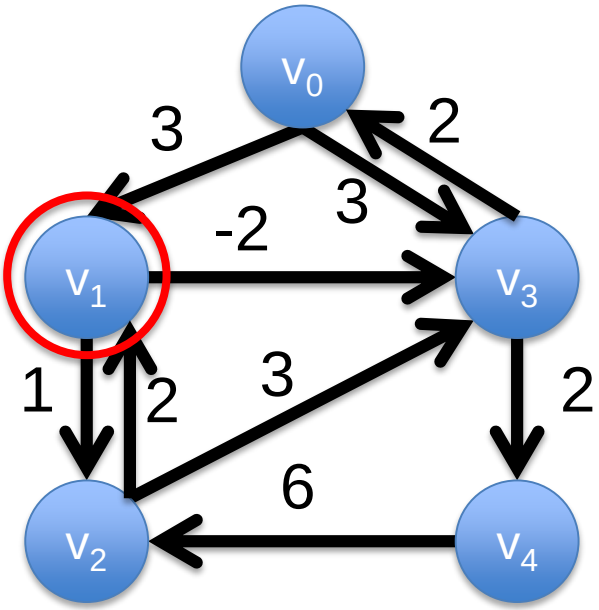
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
	0			
		0		
			0	
				0

Floyd-Warshall Method



$d_2(v_1,v_2)$ = minimum of $d_1(v_1,v_2)$, $d_1(v_1,v_1) + d_1(v_1,v_2)$

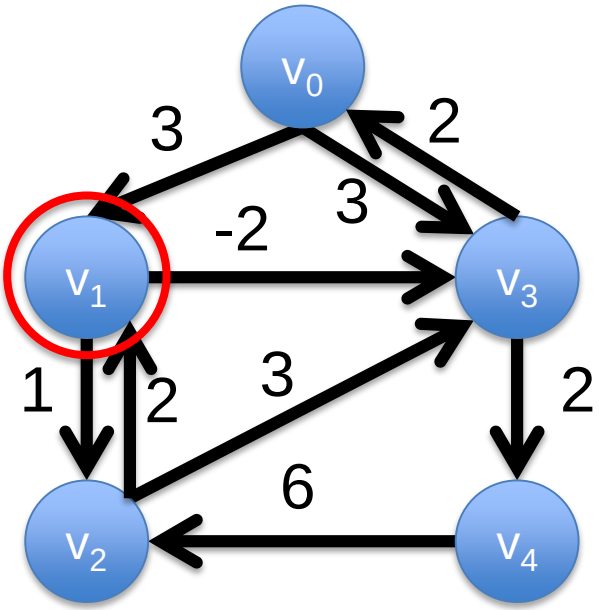
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
∞	0			
		0		
			0	
				0

Floyd-Warshall Method



$d_2(v_1,v_3)$ = minimum of $d_1(v_1,v_3)$, $d_1(v_1,v_1) + d_1(v_1,v_3)$

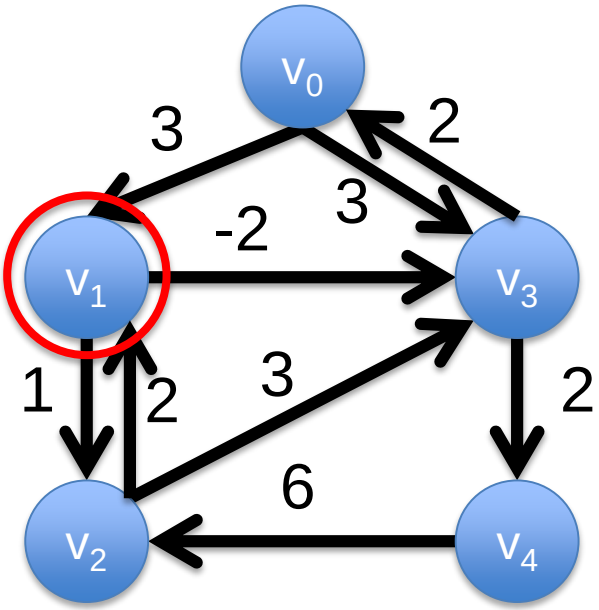
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
∞	0	1		
		0		
			0	
				0

Floyd-Warshall Method



d_1

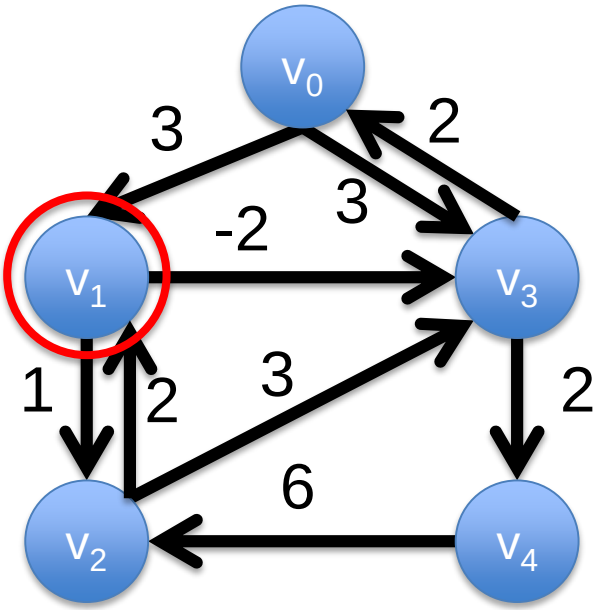
0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
∞	0	1	-2	
		0		
			0	
				0

$d_2(v_1,v_4)$ = minimum of $d_1(v_1,v_4)$, $d_1(v_1,v_1) + d_1(v_1,v_4)$

Floyd-Warshall Method



d_1

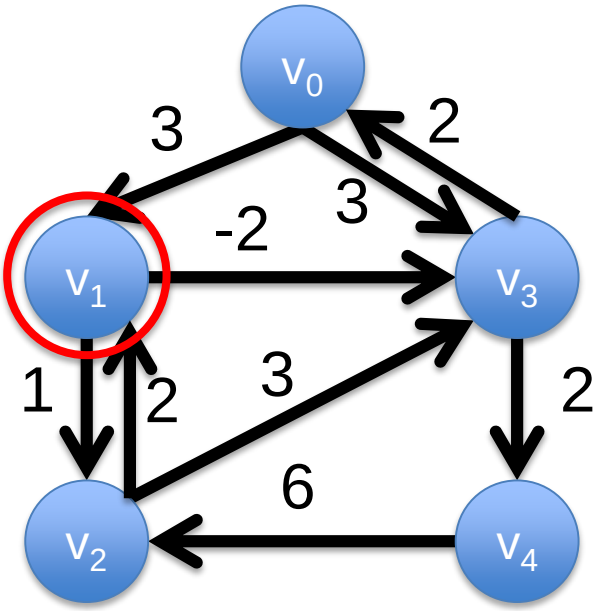
0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
∞	0	1	-2	∞
		0		
			0	
				0

$d_2(v_2,v_0)$ = minimum of $d_1(v_2,v_0)$, $d_1(v_2,v_1) + d_1(v_1,v_0)$

Floyd-Warshall Method



$d_2(v_2,v_1)$ = minimum of $d_1(v_2,v_1)$, $d_1(v_2,v_1) + d_1(v_1,v_1)$

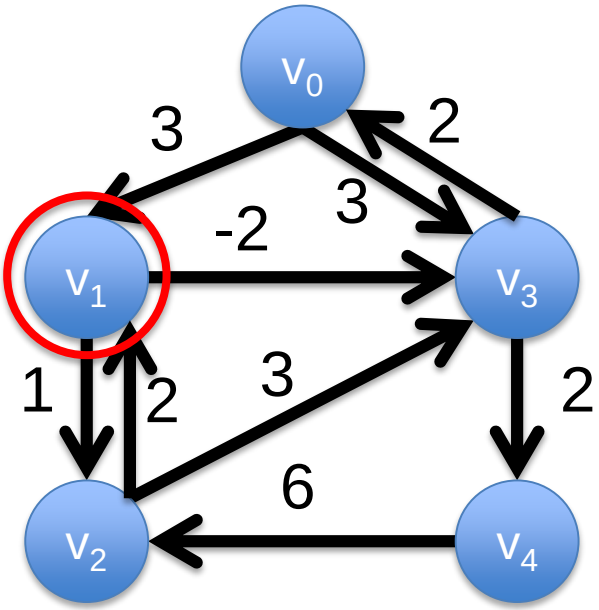
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
∞	0	1	-2	∞
∞		0		
			0	
				0

Floyd-Warshall Method



$d_2(v_2,v_3)$ = minimum of $d_1(v_2,v_3)$, $d_1(v_2,v_1) + d_1(v_1,v_3)$

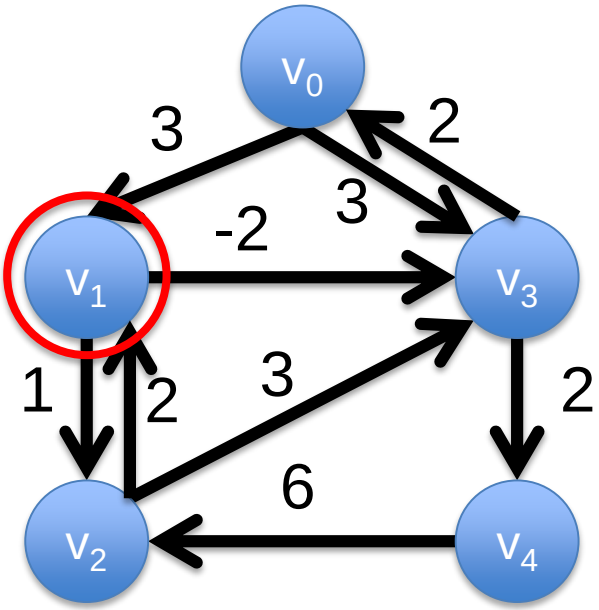
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
∞	0	1	-2	∞
∞	2	0		
			0	
				0

Floyd-Warshall Method



$d_2(v_3,v_0)$ = minimum of $d_1(v_3,v_0)$, $d_1(v_3,v_1) + d_1(v_1,v_0)$

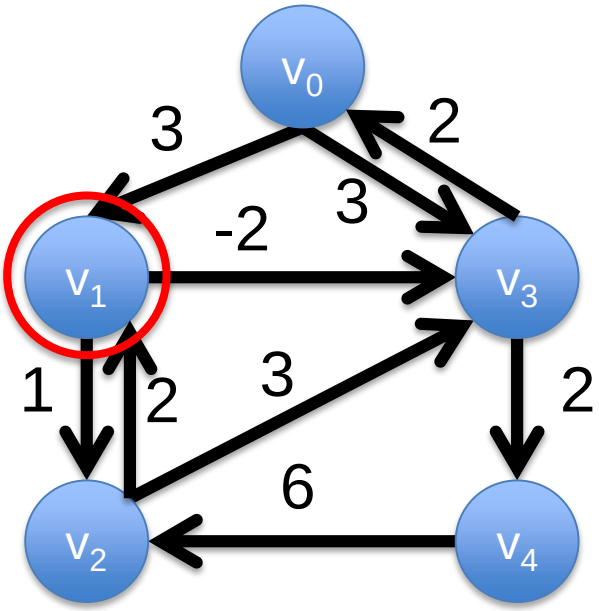
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
∞	0	1	-2	∞
∞	2	0	0	∞
			0	
				0

Floyd-Warshall Method



$d_2(v_3,v_1)$ = minimum of $d_1(v_3,v_1)$, $d_1(v_3,v_1) + d_1(v_1,v_1)$

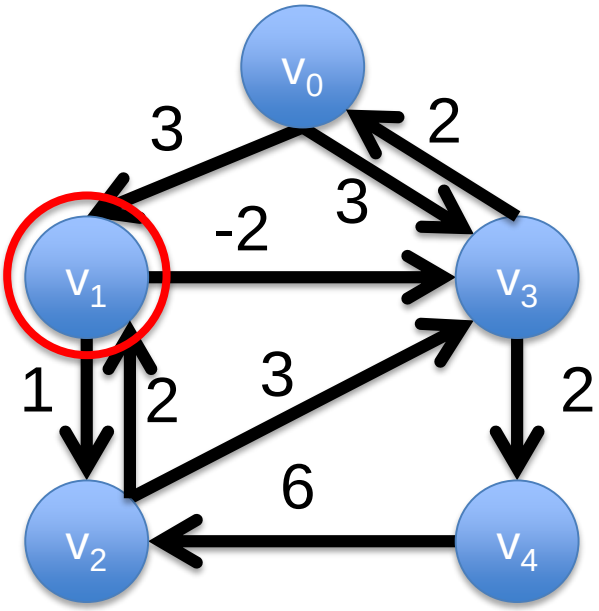
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
∞	0	1	-2	∞
∞	2	0	0	∞
2			0	
				0

Floyd-Warshall Method



$d_2(v_3,v_2)$ = minimum of $d_1(v_3,v_2)$, $d_1(v_3,v_1) + d_1(v_1,v_2)$

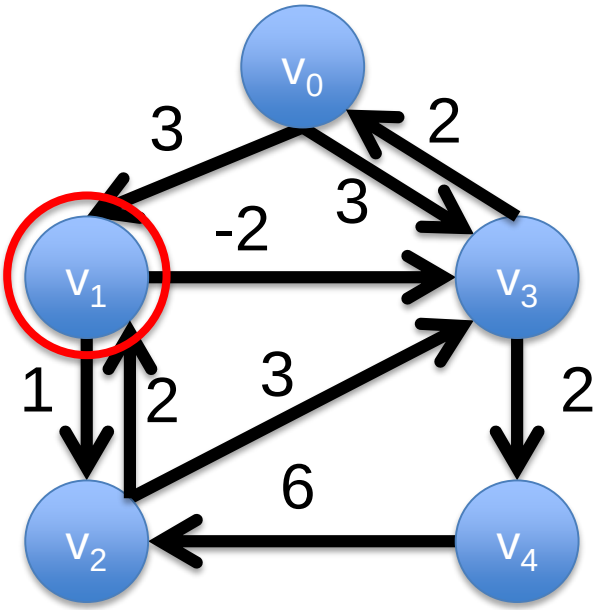
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
∞	0	1	-2	∞
∞	2	0	0	∞
2	5		0	
				0

Floyd-Warshall Method



$d_2(v_3,v_4)$ = minimum of $d_1(v_3,v_4)$, $d_1(v_3,v_1) + d_1(v_1,v_4)$

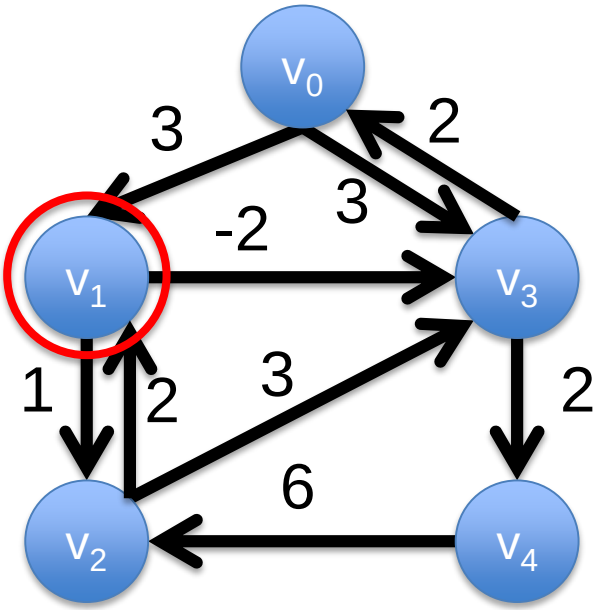
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
∞	0	1	-2	∞
∞	2	0	0	∞
2	5	6	0	
				0

Floyd-Warshall Method



$d_2(v_4,v_0)$ = minimum of $d_1(v_4,v_0)$, $d_1(v_4,v_1) + d_1(v_1,v_0)$

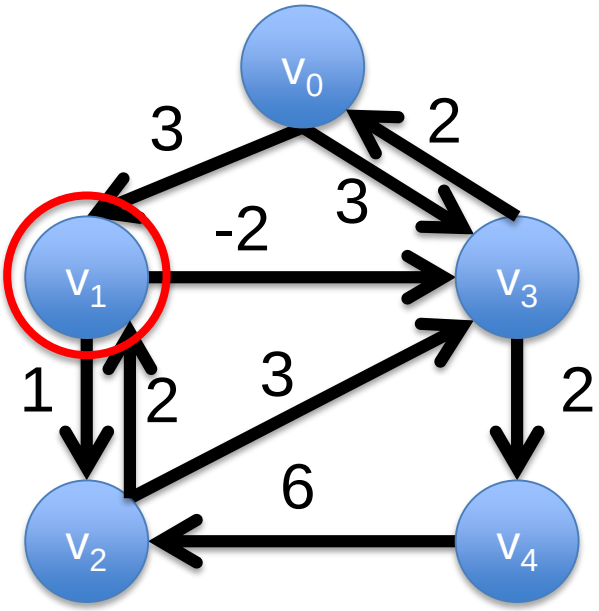
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
∞	0	1	-2	∞
∞	2	0	0	∞
2	5	6	0	2
				0

Floyd-Warshall Method



d_1

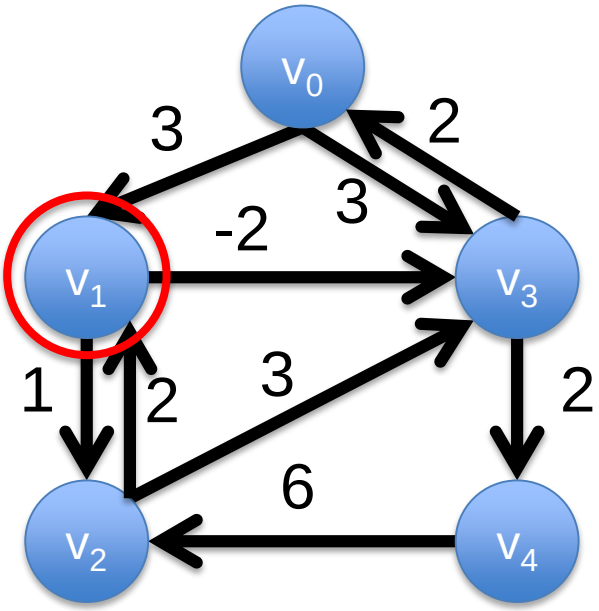
0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
∞	0	1	-2	∞
∞	2	0	0	∞
2	5	6	0	2
∞				0

$d_2(v_4,v_1)$ = minimum of $d_1(v_4,v_1)$, $d_1(v_4,v_1) + d_1(v_1,v_1)$

Floyd-Warshall Method



$d_2(v_4,v_2)$ = minimum of $d_1(v_4,v_2)$, $d_1(v_4,v_1) + d_1(v_1,v_2)$

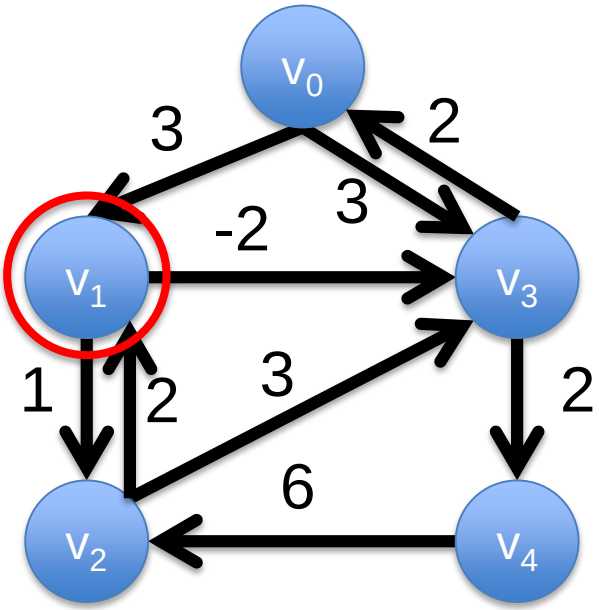
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
∞	0	1	-2	∞
∞	2	0	0	∞
2	5	6	0	2
∞	∞			0

Floyd-Warshall Method



$d_2(v_4,v_3)$ = minimum of $d_1(v_4,v_3)$, $d_1(v_4,v_1) + d_1(v_1,v_3)$

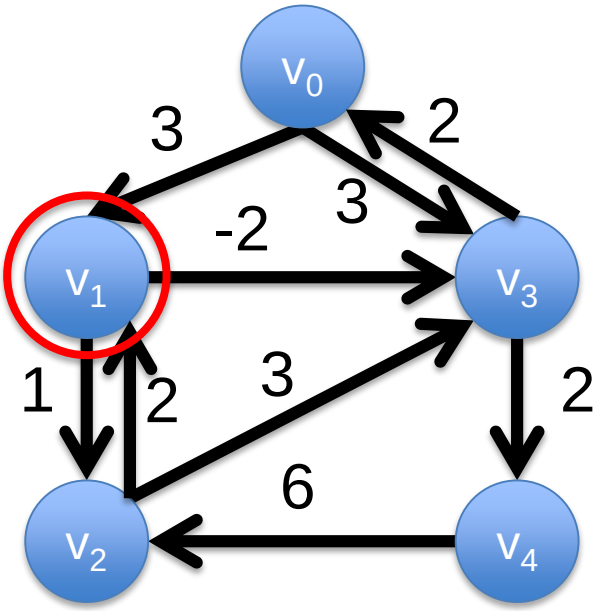
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
∞	0	1	-2	∞
∞	2	0	0	∞
2	5	6	0	2
∞	∞	6		0

Floyd-Warshall Method



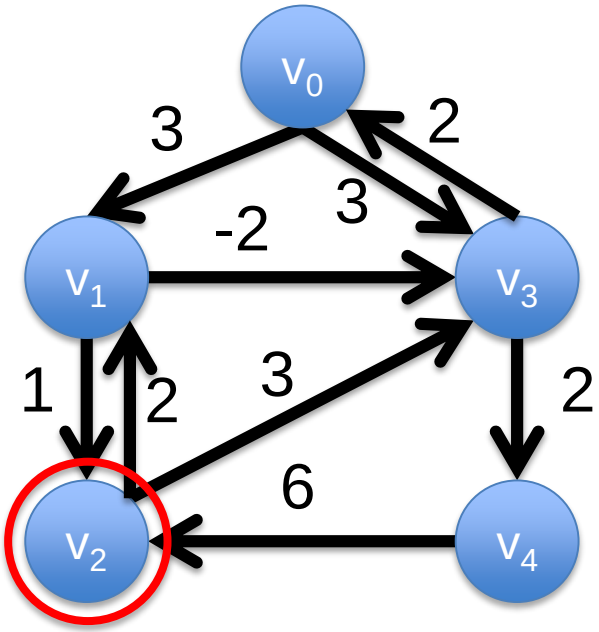
d_1

0	3	∞	3	∞
∞	0	1	-2	∞
∞	2	0	3	∞
2	5	∞	0	2
∞	∞	6	∞	0

d_2

0	3	4	1	∞
∞	0	1	-2	∞
∞	2	0	0	∞
2	5	6	0	2
∞	∞	6	∞	0

Floyd-Warshall Method



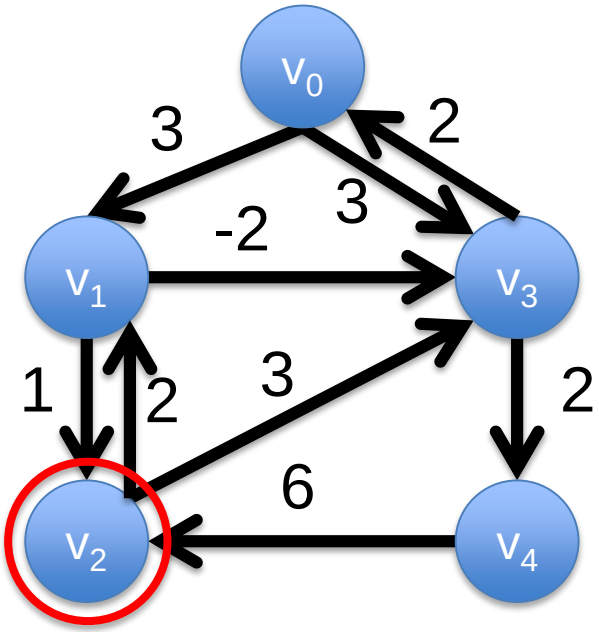
d_2

0	3	4	1	∞
∞	0	1	-2	∞
∞	2	0	0	∞
2	5	6	0	2
∞	∞	6	∞	0

d_3

0				
	0			
		0		
			0	
				0

Floyd-Warshall Method



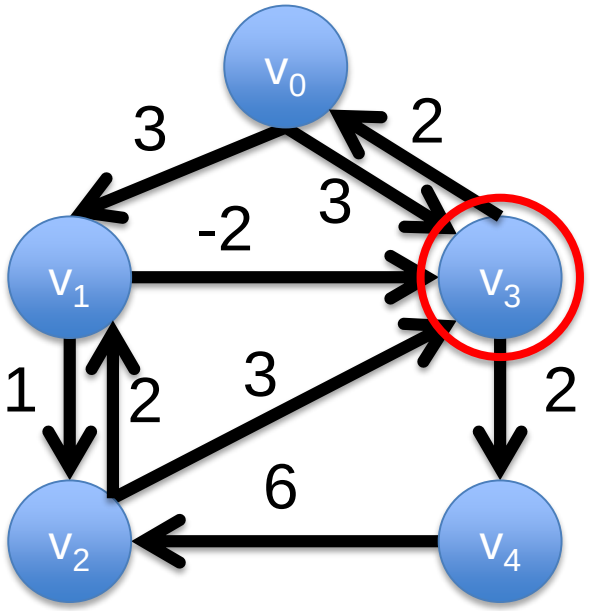
d_2

0	3	4	1	∞
∞	0	1	-2	∞
∞	2	0	0	∞
2	5	6	0	2
∞	∞	6	∞	0

d_3

0	3	4	1	∞
∞	0	1	-2	∞
∞	2	0	0	∞
2	5	6	0	2
∞	8	6	6	0

Floyd-Warshall Method



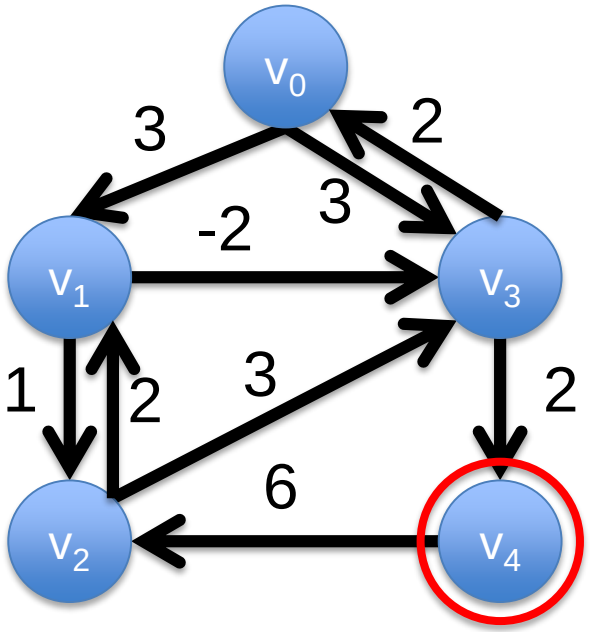
d_3

0	3	4	1	∞
∞	0	1	-2	∞
∞	2	0	0	∞
2	5	6	0	2
∞	8	6	6	0

d_4

0	3	4	1	3
0	0	1	-2	0
2	2	0	0	2
2	5	6	0	2
8	8	6	6	0

Floyd-Warshall Method



d_4

0	3	4	1	3
0	0	1	-2	0
2	2	0	0	2
2	5	6	0	2
8	8	6	6	0

d_5

0	3	4	1	3
0	0	1	-2	0
2	2	0	0	2
2	5	6	0	2
8	8	6	6	0

Summary

$k = 0$. $d_k(i,j) = l(i,j)$ if $(i,j) \in A$, otherwise ∞

While ($k < n$)

For each $i \in V$

For each $j \in V$

$d_{k+1}(i,j) = \min \{ d_k(i,j), d_k(i,v_{k-1}) + d_k(v_{k-1},j)\}$

End

End

$k = k + 1$

End

Time Complexity

$O(n^3)$

Outer iteration
Run ‘n’ times

Inner Iteration
$O(n^2)$ operations: O(1) for each pair of vertices