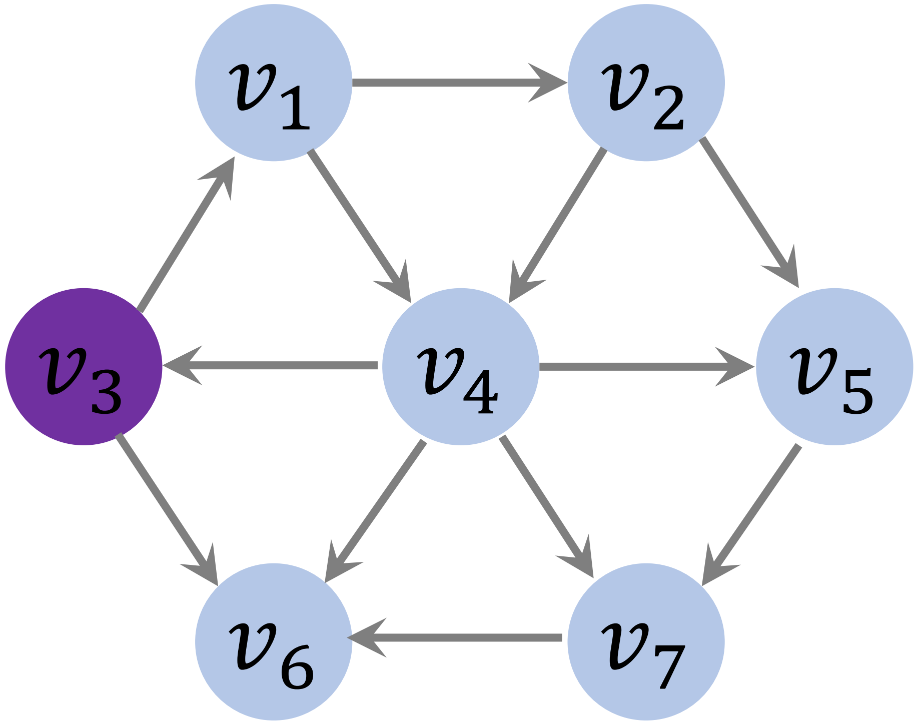
Finding Shortest Paths in Unweighted Graphs

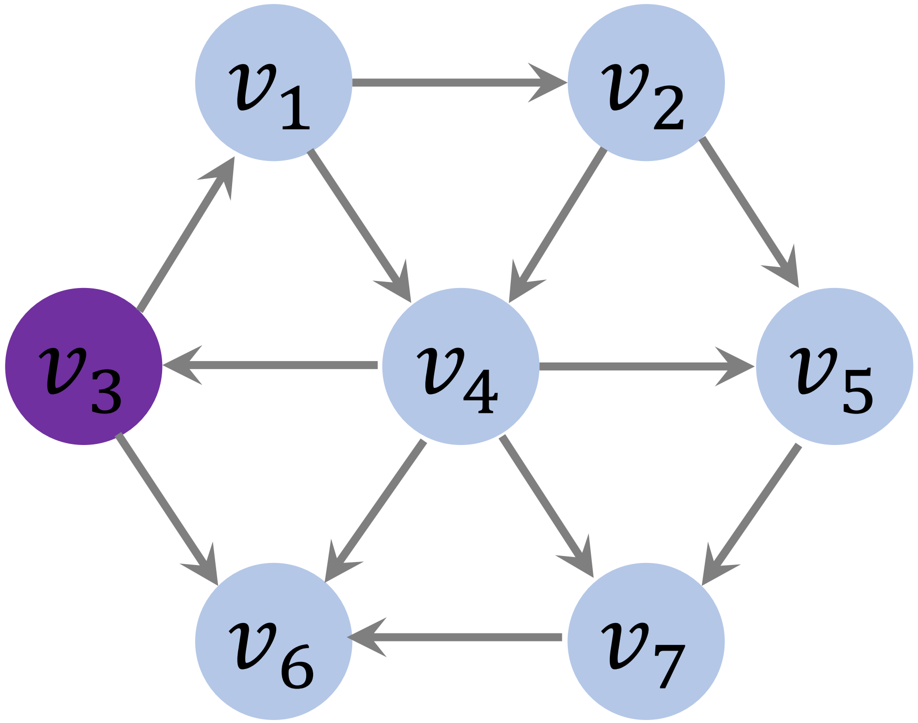
Shusen Wang

Shortest Path in Unweighted Graphs



- Weights are all ones.
- Weights of nonexistent edges are ∞.
- Easier problem: finding shortest path in unweighted graph.
- Harder problem: finding shortest path in weighted graph.

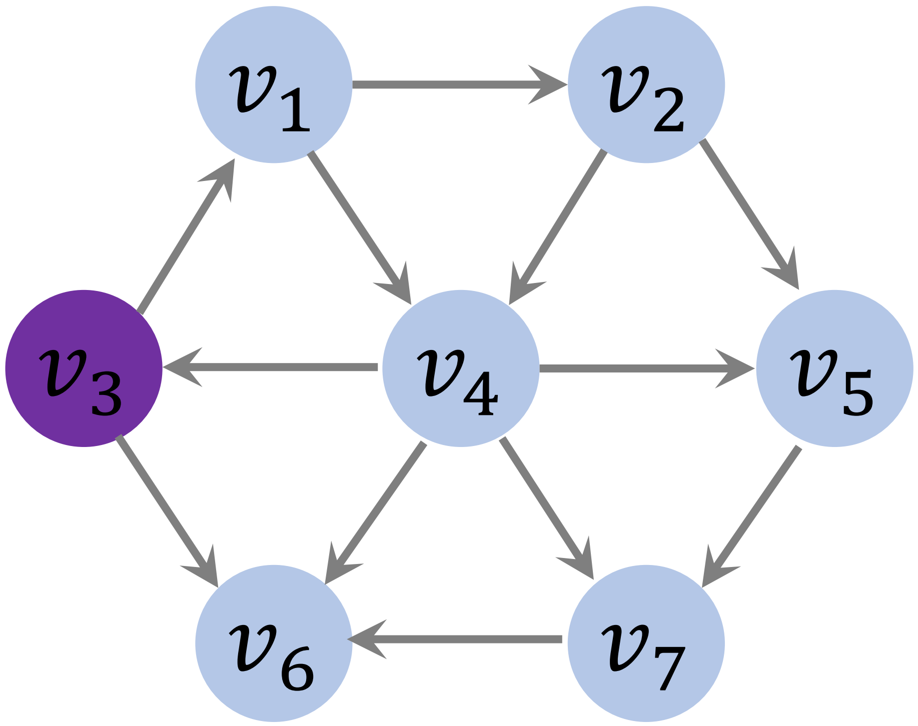
Shortest Path in Unweighted Graphs



vertex	dist	path
v_1	**1**	v_3
v_2	**2**	v_1
v_3	**0**	0
v_4	**2**	v_1
v_5	**3**	v_2
v_6	**1**	v_3
v_7	**3**	v_4

Algorithm

Preparations

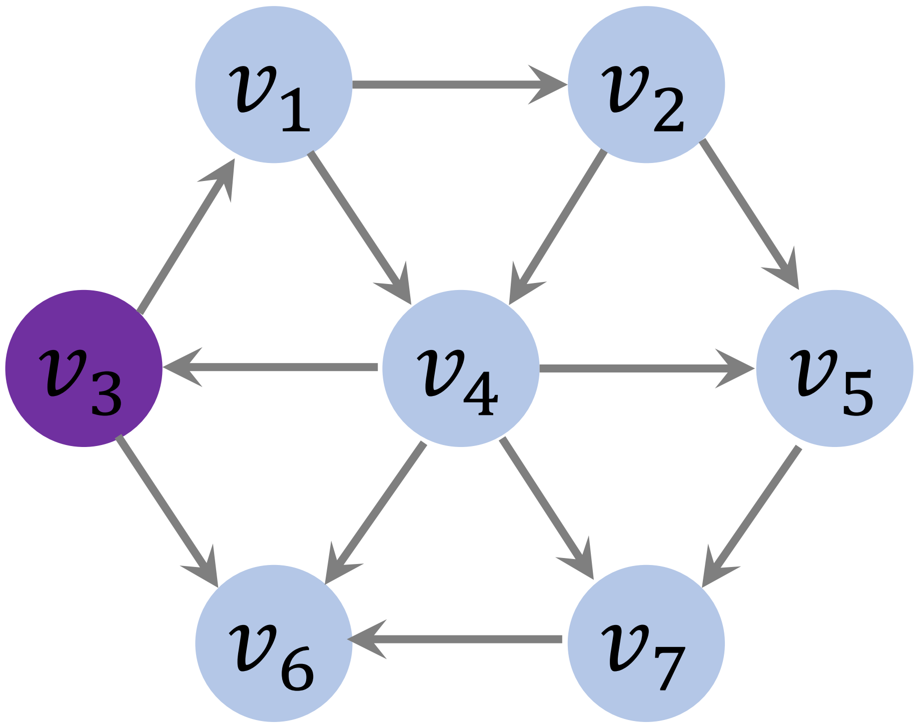


- v_3 is the source.

Queue:

vertex	visit	dist	path
v_1	no	∞	0
v_2	no	∞	0
v_3	no	∞	0
v_4	no	∞	0
v_5	no	∞	0
v_6	no	∞	0
v_7	no	∞	0

Preparations

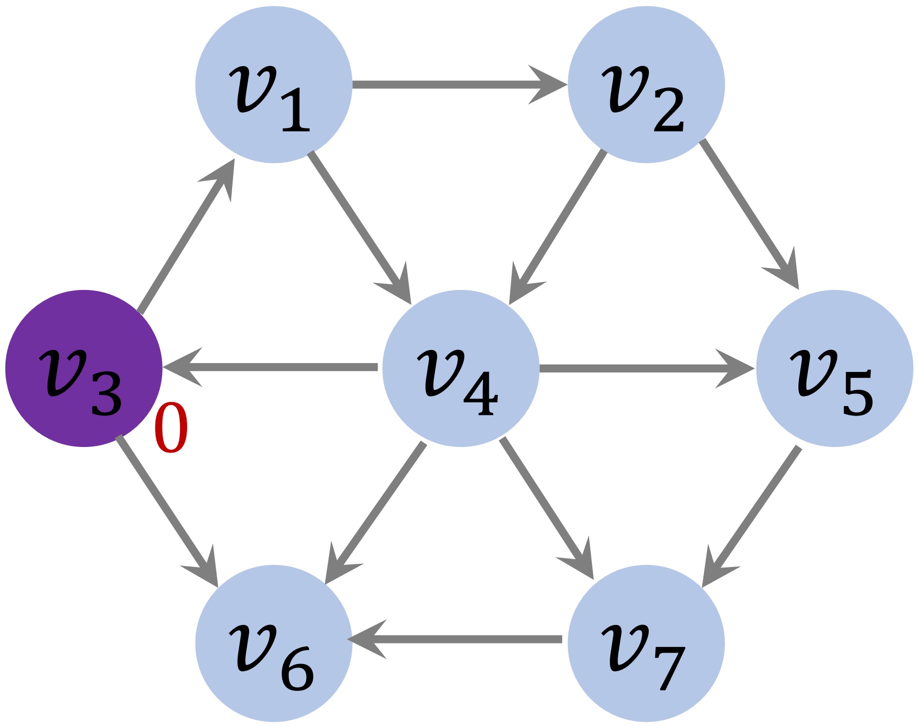


- v_3 is the source.

Queue:

vertex	visit	dist	path
v_1	no	∞	0
v_2	no	∞	0
v_3	no	∞	0
v_4	no	∞	0
v_5	no	∞	0
v_6	no	∞	0
v_7	no	∞	0

Initial State

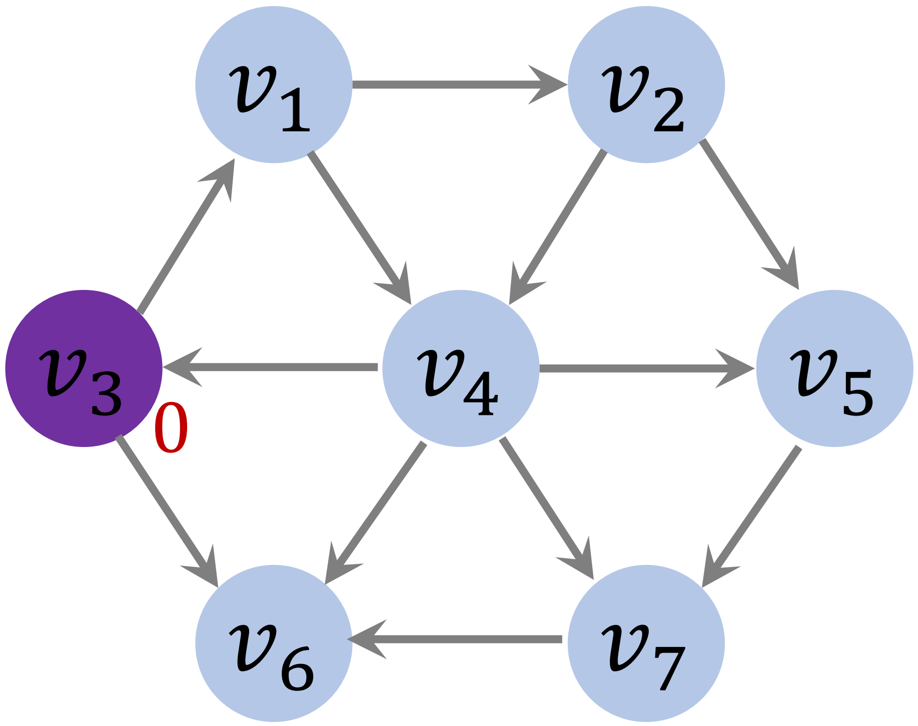


- v_3 is the source.
- Mark v_3 as ”visited”.
- Set v_3’s distance to 0.

Queue:

vertex	visit	dist	path
v_1	no	∞	0
v_2	no	∞	0
v_3	**yes**	**0**	0
v_4	no	∞	0
v_5	no	∞	0
v_6	no	∞	0
v_7	no	∞	0

Initial State

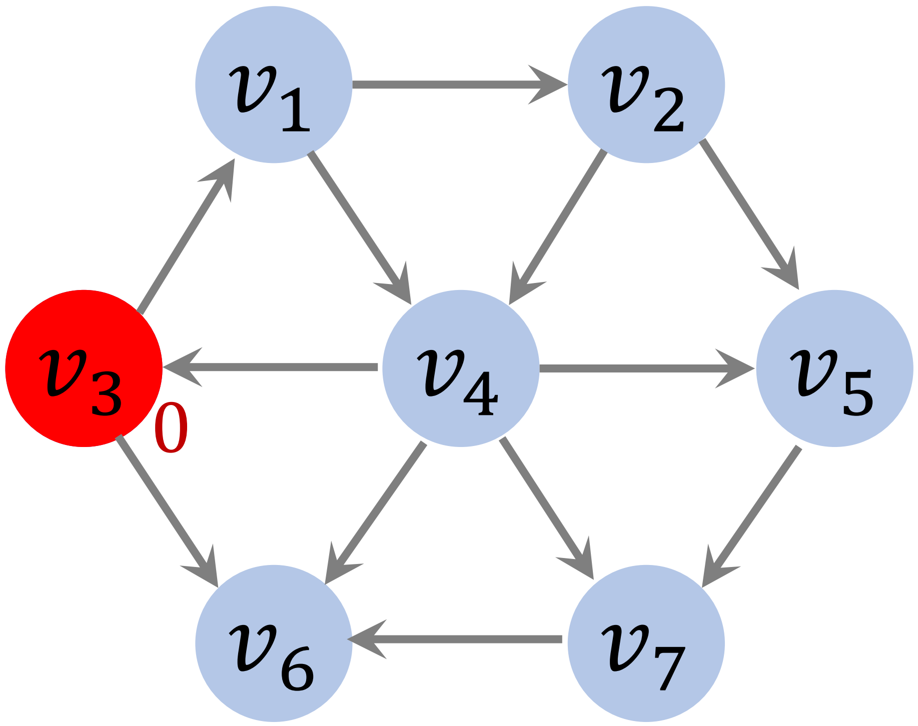


- enqueue(v_3).

Queue:

vertex	visit	dist	path
v_1	no	∞	0
v_2	no	∞	0
v_3	**yes**	**0**	0
v_4	no	∞	0
v_5	no	∞	0
v_6	no	∞	0
v_7	no	∞	0

Iteration 1

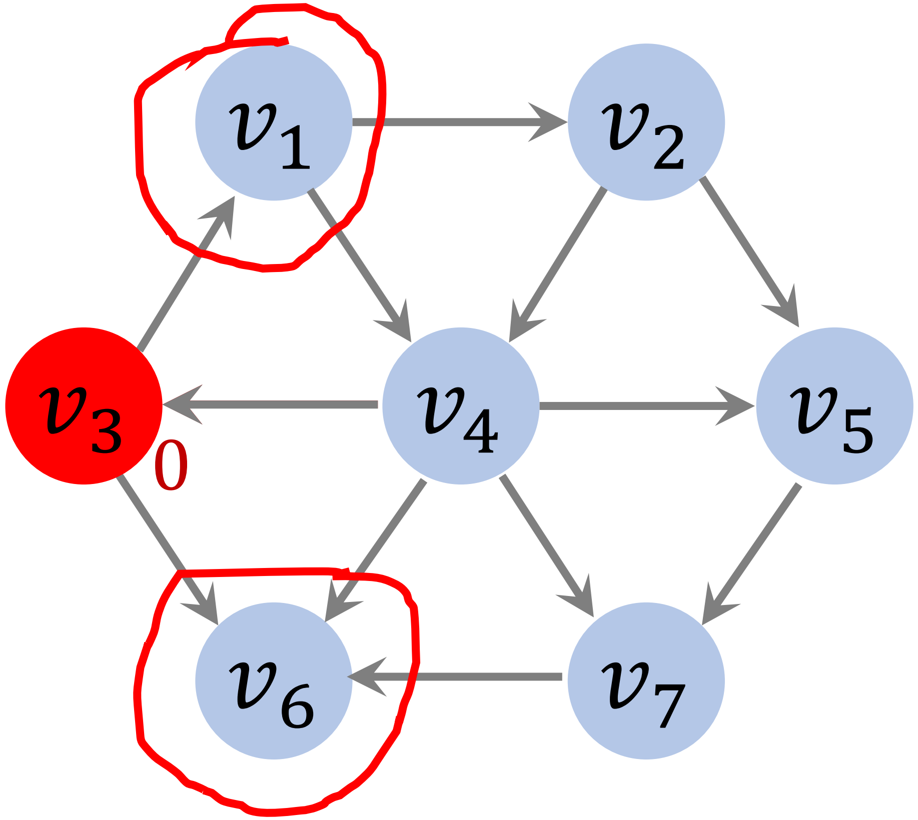


- $v_3 \leftarrow$ dequeue().

Queue:

vertex	visit	dist	path
v_1	no	∞	0
v_2	no	∞	0
v_3	**yes**	**0**	0
v_4	no	∞	0
v_5	no	∞	0
v_6	no	∞	0
v_7	no	∞	0

Iteration 1

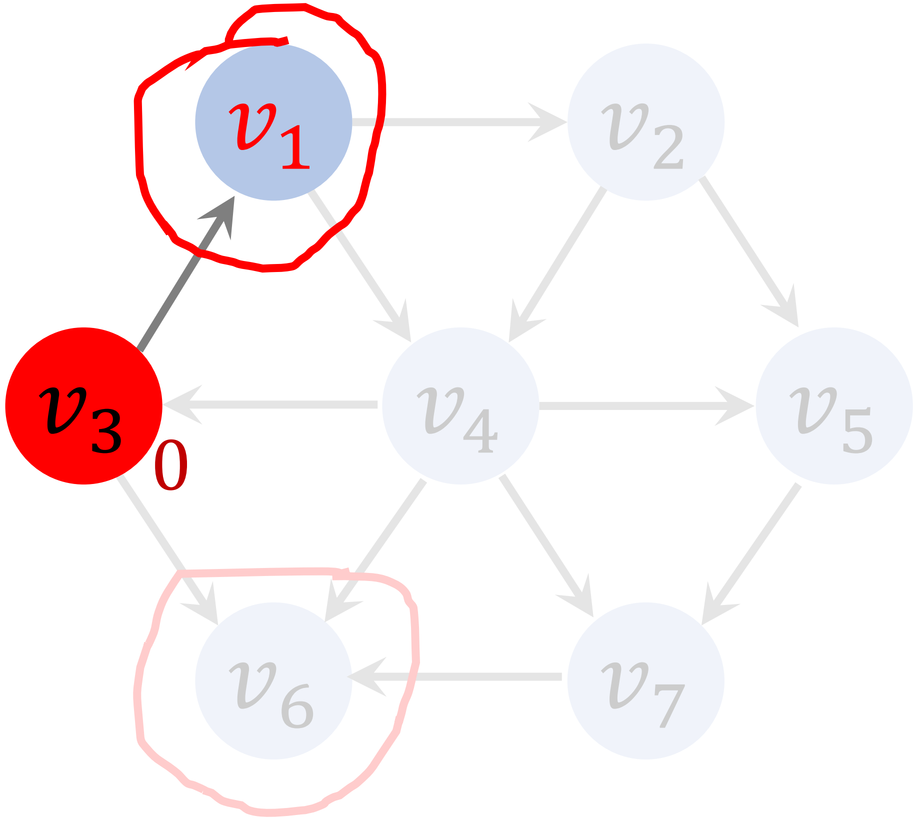


- $v_3 \leftarrow$ dequeue().
- Find adjacent vertices of v_3: v_1 and v_6.

Queue:

vertex	visit	dist	path
v_1	no	∞	0
v_2	no	∞	0
v_3	**yes**	**0**	0
v_4	no	∞	0
v_5	no	∞	0
v_6	no	∞	0
v_7	no	∞	0

Iteration 1(A)

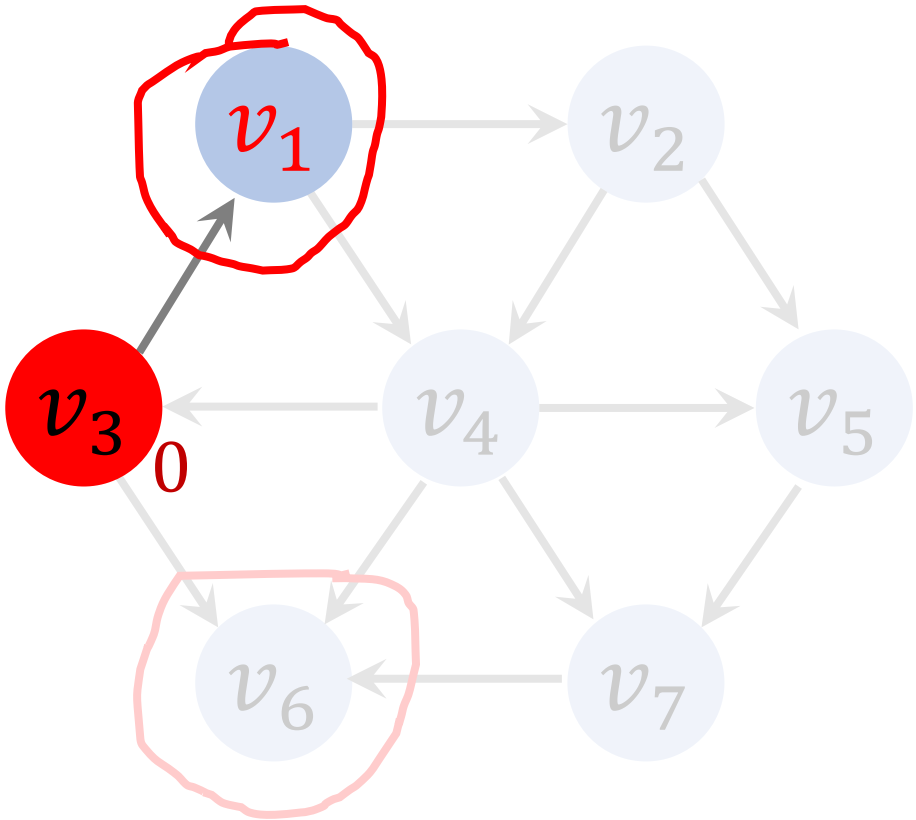


- Has v_1 been visited?

Queue:

vertex	visit	dist	path
v_1	no	∞	0
v_2	no	∞	0
v_3	**yes**	**0**	0
v_4	no	∞	0
v_5	no	∞	0
v_6	no	∞	0
v_7	no	∞	0

Iteration 1(A)

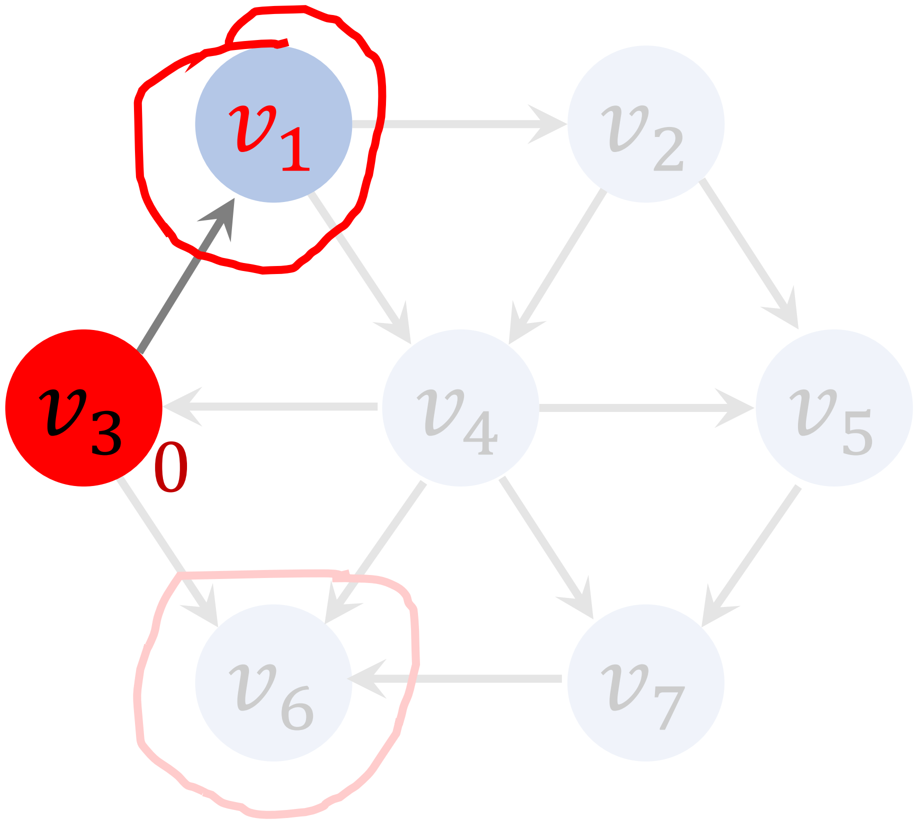


- Has v_1 been visited?
- No.
- ➔ Work on v_1.

Queue:

vertex	visit	dist	path
v_1	no	∞	0
v_2	no	∞	0
v_3	**yes**	**0**	0
v_4	no	∞	0
v_5	no	∞	0
v_6	no	∞	0
v_7	no	∞	0

Iteration 1(A)



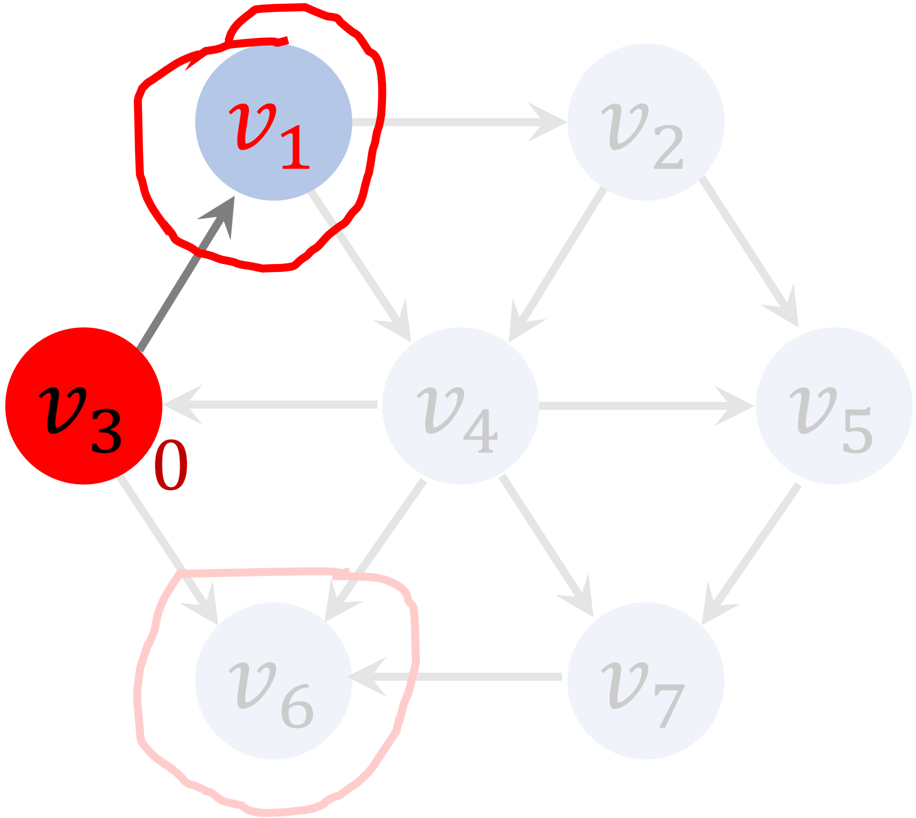
- enqueue(v_1).

Queue:

v_1

vertex	visit	dist	path
v_1	no	∞	0
v_2	no	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	no	∞	0
v_7	no	∞	0

Iteration 1(A)



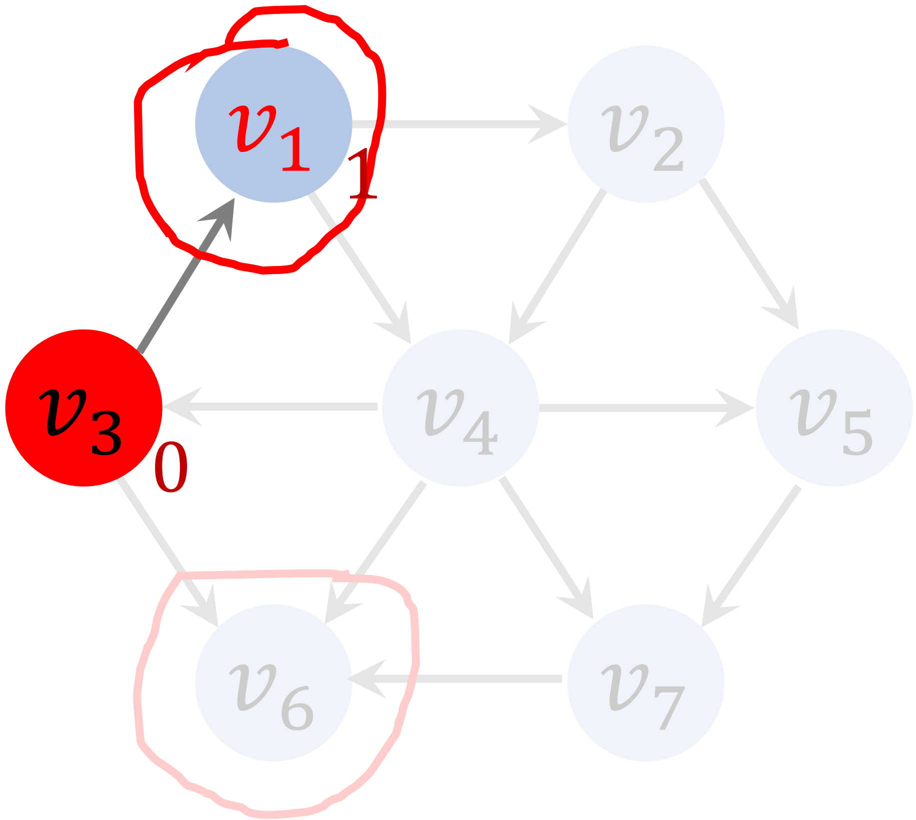
- visit[1] = true.

Queue:

v_1

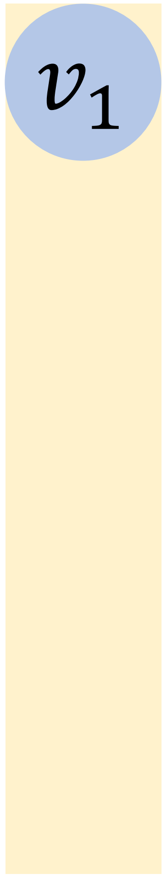
vertex	visit	dist	path
v_1	yes	∞	0
v_2	no	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	no	∞	0
v_7	no	∞	0

Iteration 1(A)



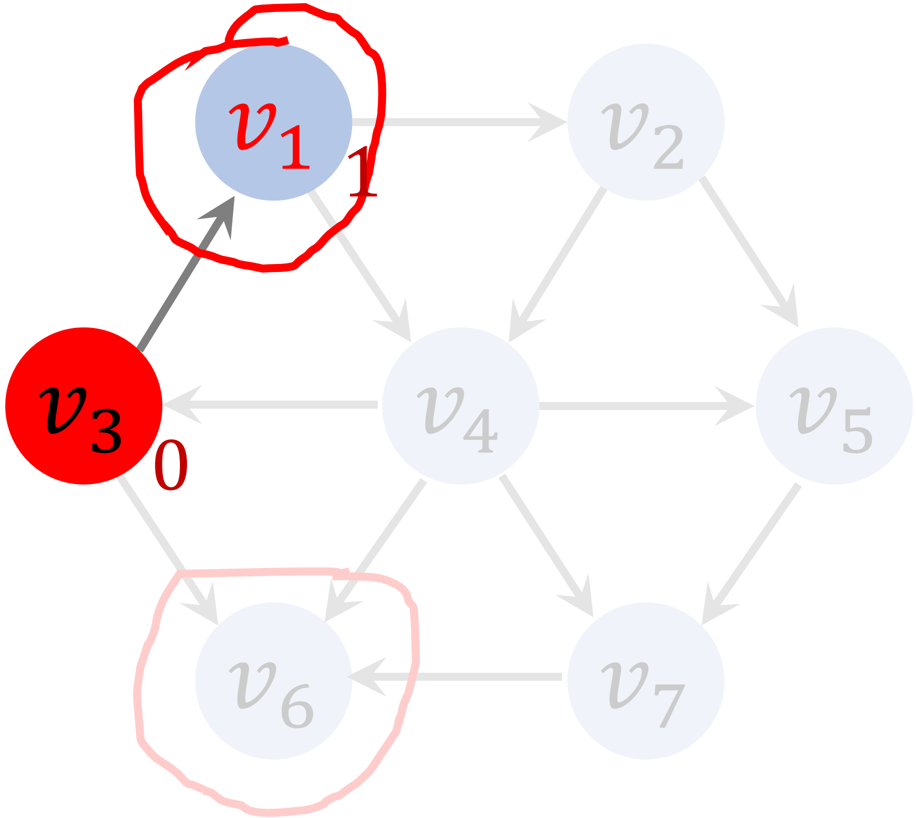
- visit[1] = true.
- dist[1] = dist[3] + 1 = 1.

Queue:



vertex	visit	dist	path
v_1	yes	1	0
v_2	no	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	no	∞	0
v_7	no	∞	0

Iteration 1(A)



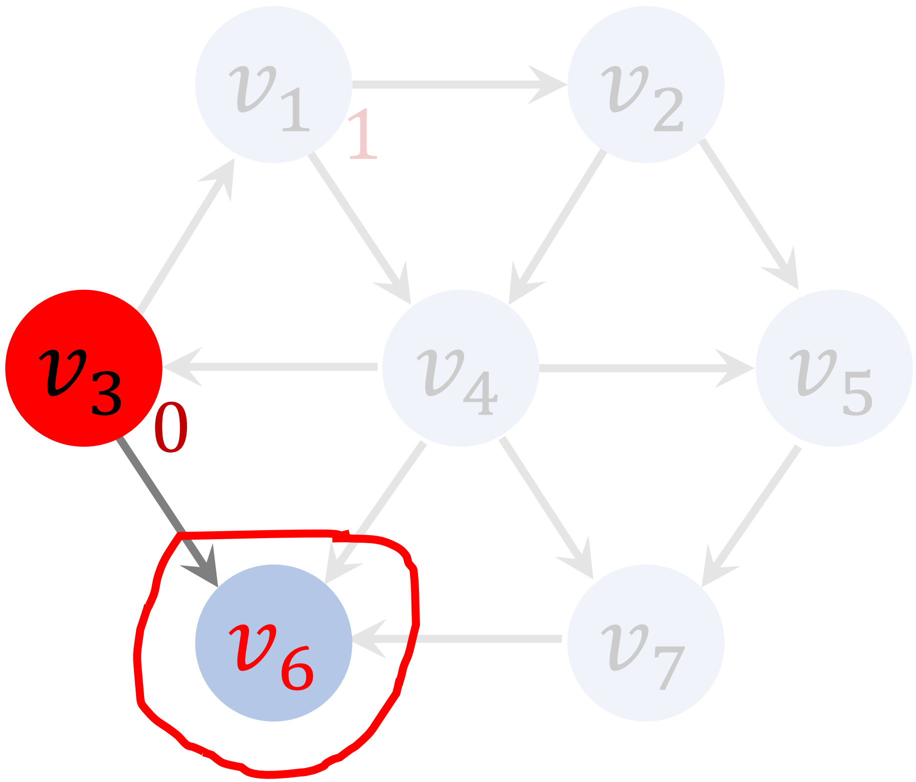
- visit[1] = true.
- dist[1] = dist[3] + 1 = 1.
- path[1] = v_3.

Queue:

v_1

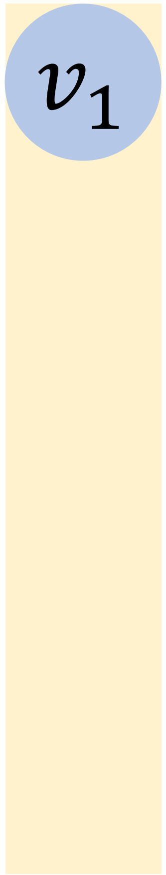
vertex	visit	dist	path
v_1	yes	1	v_3
v_2	no	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	no	∞	0
v_7	no	∞	0

Iteration 1(B)



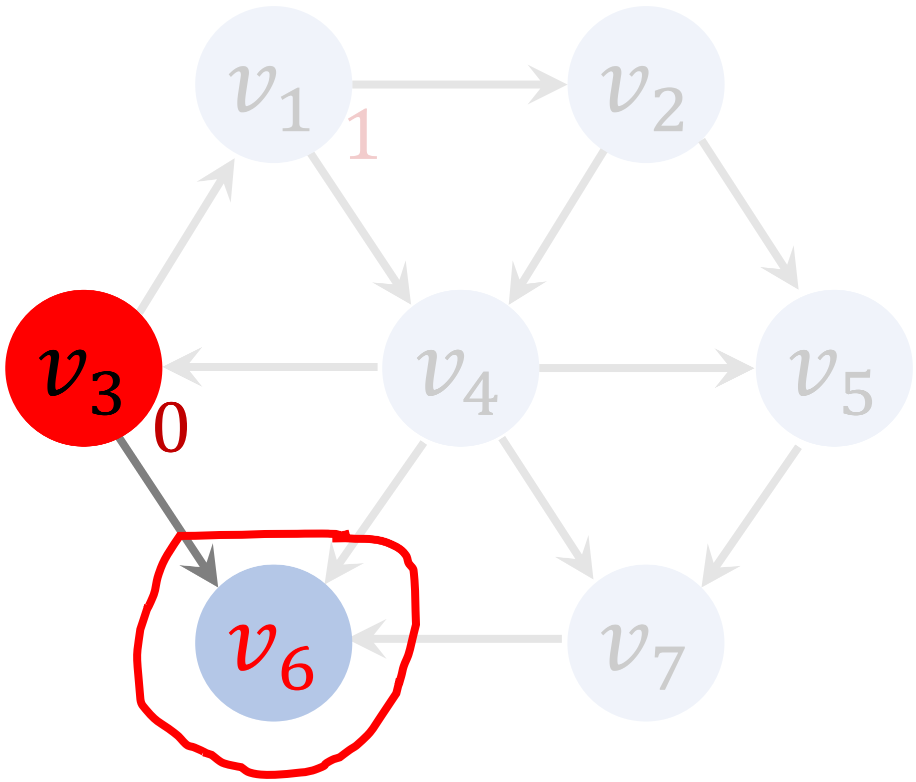
- Has v_6 been visited?

Queue:

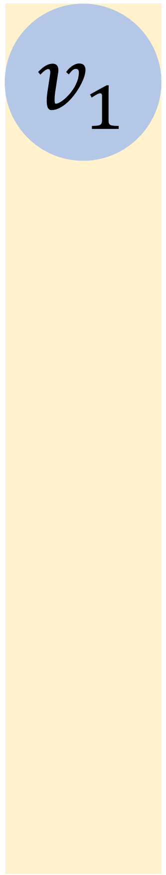


vertex	visit	dist	path
v_1	yes	1	v_3
v_2	no	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	no	∞	0
v_7	no	∞	0

Iteration 1(B)



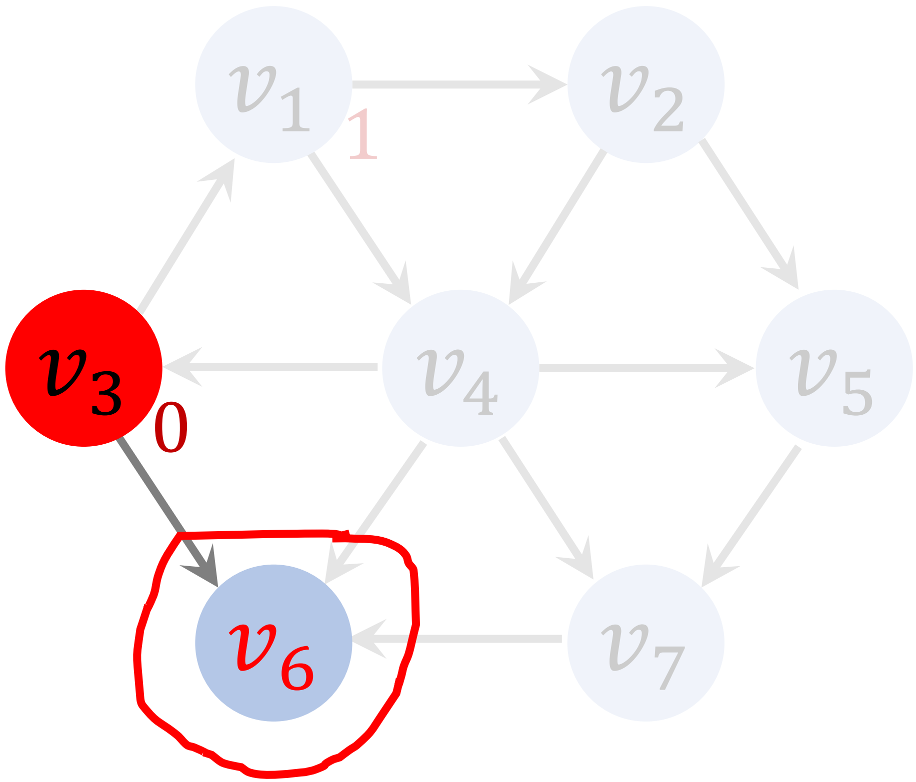
Queue:



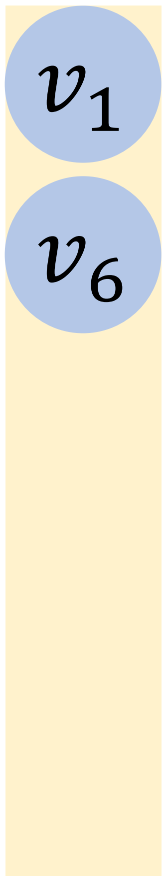
vertex	visit	dist	path
v_1	yes	1	v_3
v_2	no	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	no	∞	0
v_7	no	∞	0

- Has v_6 been visited?
- No.
- ➔ Work on v_6.

Iteration 1(B)



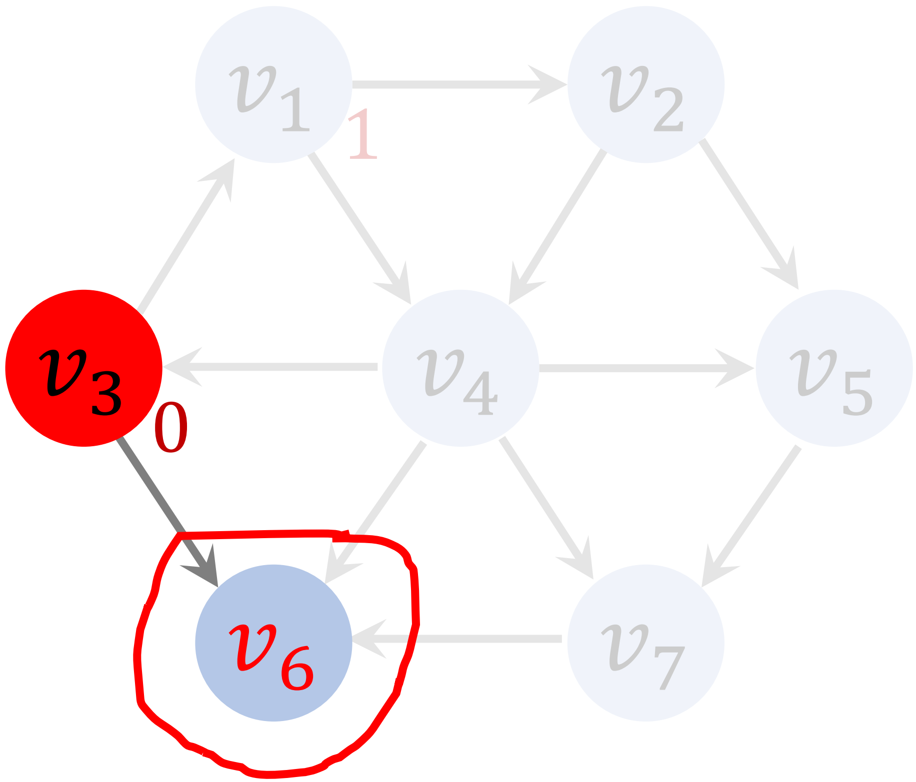
Queue:



vertex	visit	dist	path
v_1	yes	1	v_3
v_2	no	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	no	∞	0
v_7	no	∞	0

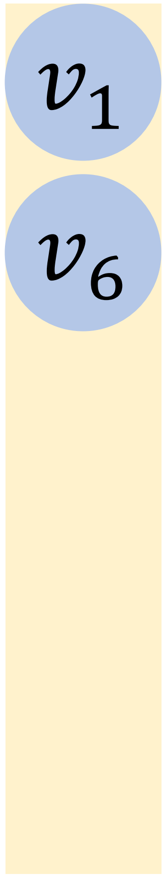
- enqueue(v_6).

Iteration 1(B)



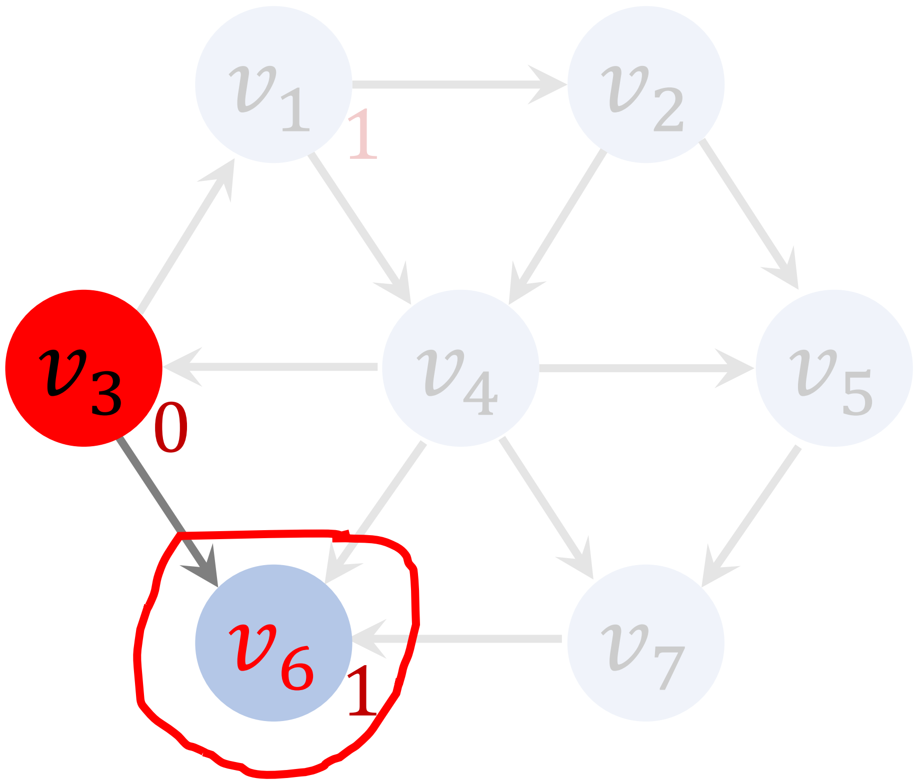
- visit[6] = true.

Queue:



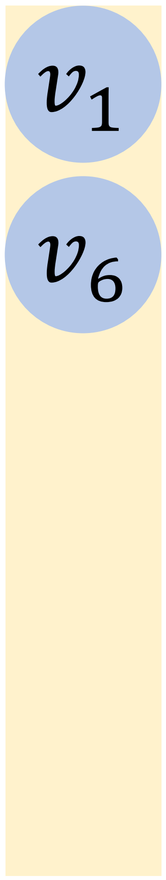
vertex	visit	dist	path
v_1	yes	1	v_3
v_2	no	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	yes	∞	0
v_7	no	∞	0

Iteration 1(B)



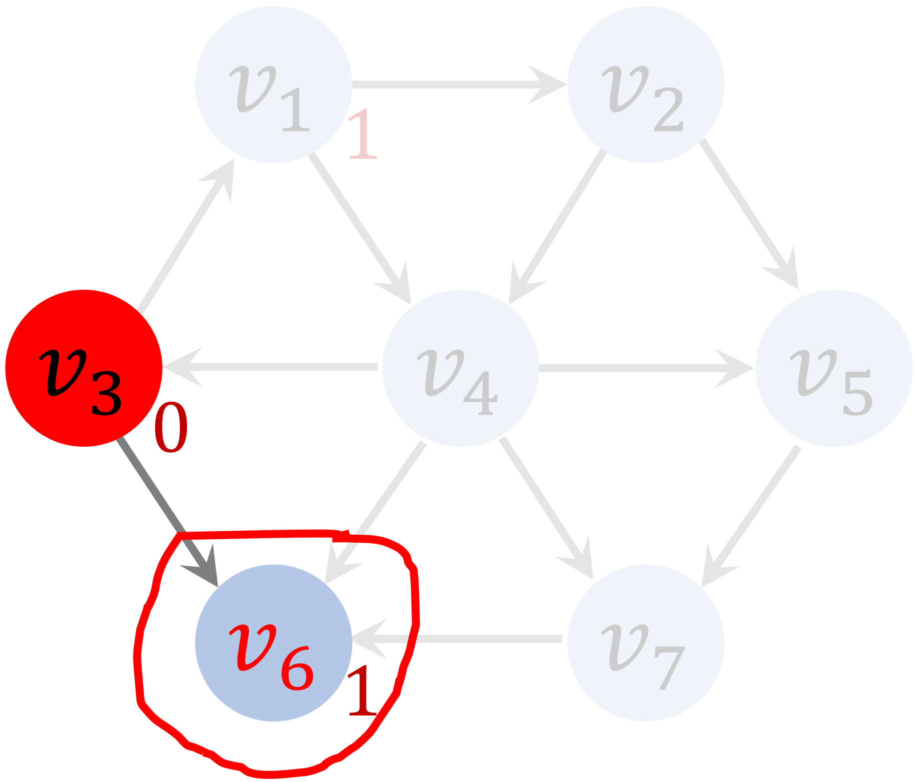
- visit[6] = true.
- dist[6] = dist[3] + 1 = 1.

Queue:



vertex	visit	dist	path
v_1	yes	1	v_3
v_2	no	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	yes	1	0
v_7	no	∞	0

Iteration 1(B)



Queue:

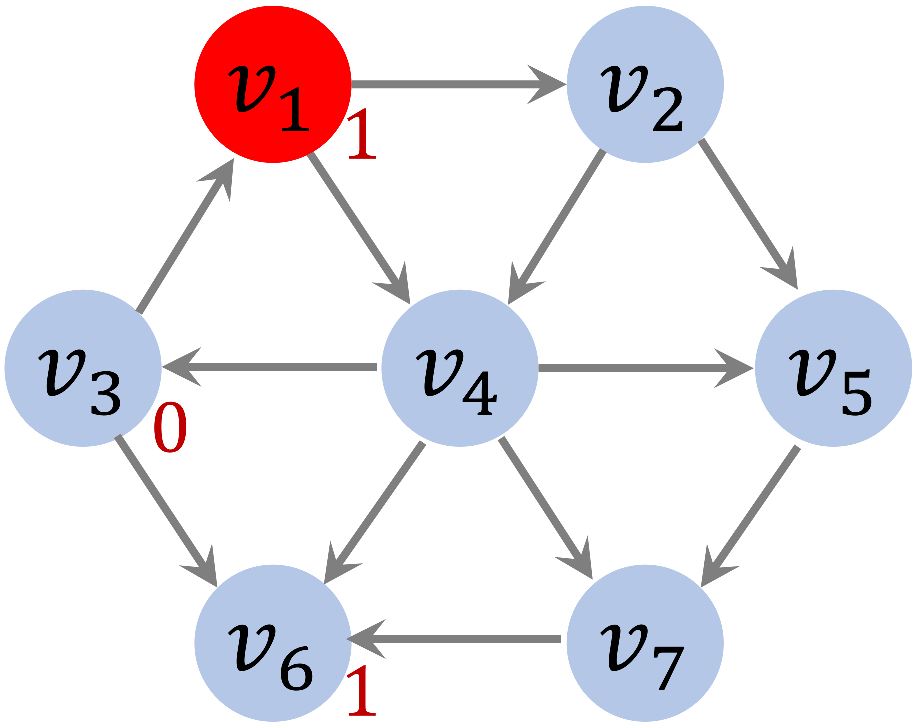
v_1

v_6

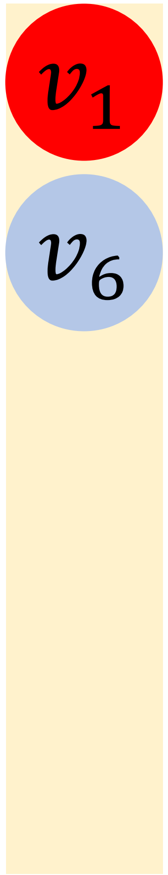
vertex	visit	dist	path
v_1	yes	1	v_3
v_2	no	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

- $\text{visit}[6] = \text{true}.$
- $\text{dist}[6] = \text{dist}[3] + 1 = 1.$
- $\text{path}[6] = v_3.$

Iteration 2



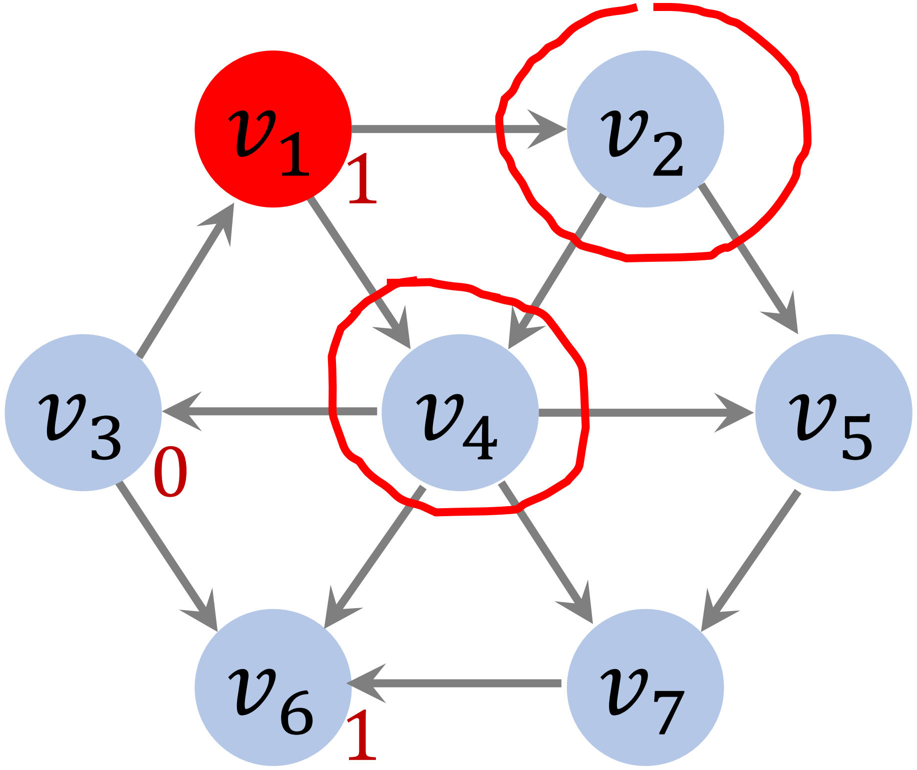
Queue:



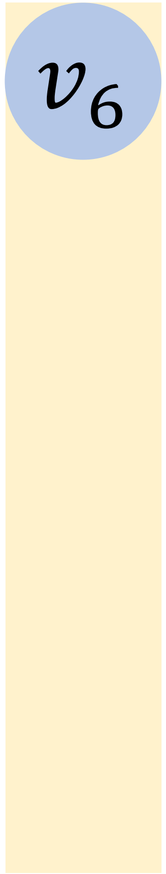
vertex	visit	dist	path
v_1	yes	1	v_3
v_2	no	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

- $v_1 \leftarrow$ dequeue().

Iteration 2



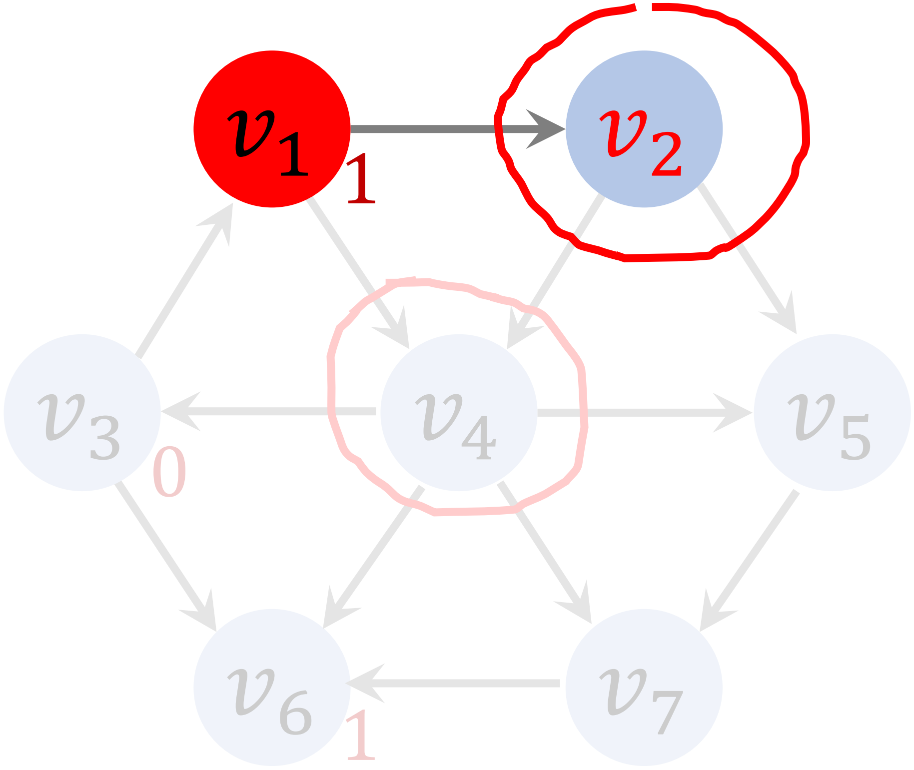
Queue:



vertex	visit	dist	path
v_1	yes	1	v_3
v_2	no	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

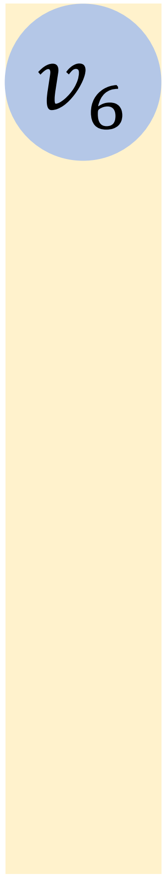
- $v_1 \leftarrow$ dequeue().
- Find adjacent vertices of v_1:
  v_2 and v_4.

Iteration 2(A)



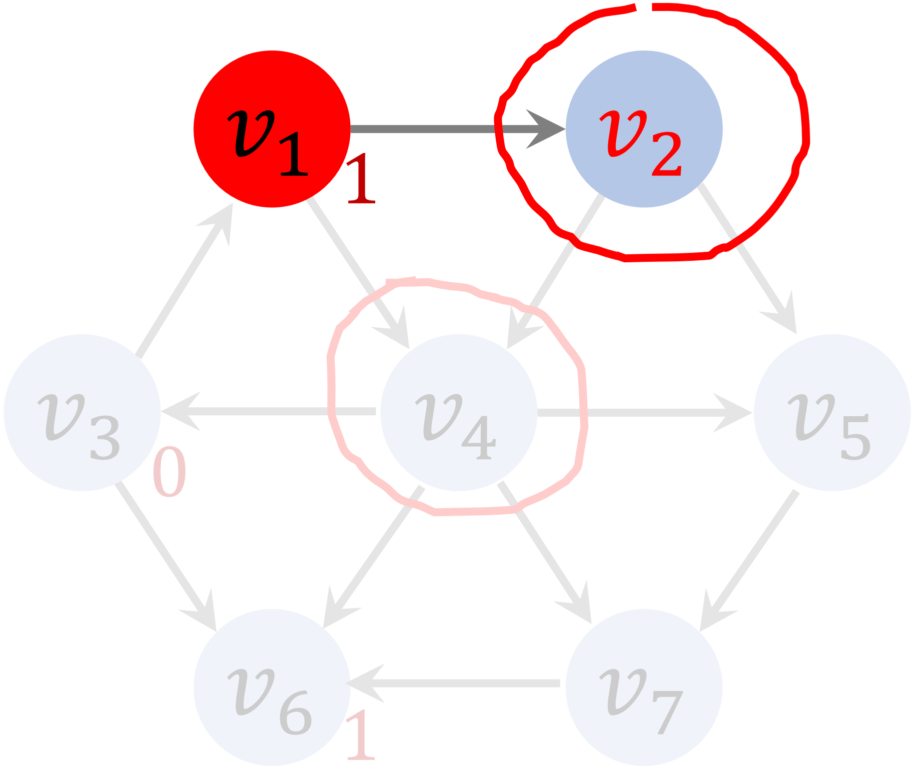
- Has v_2 been visited?

Queue:

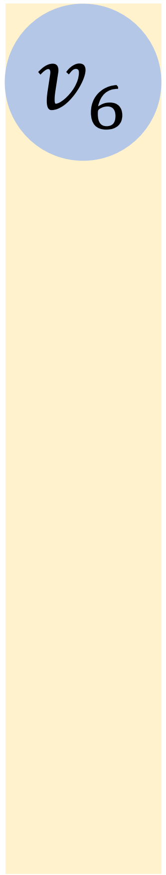


vertex	visit	dist	path
v_1	yes	1	v_3
v_2	no	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

Iteration 2(A)



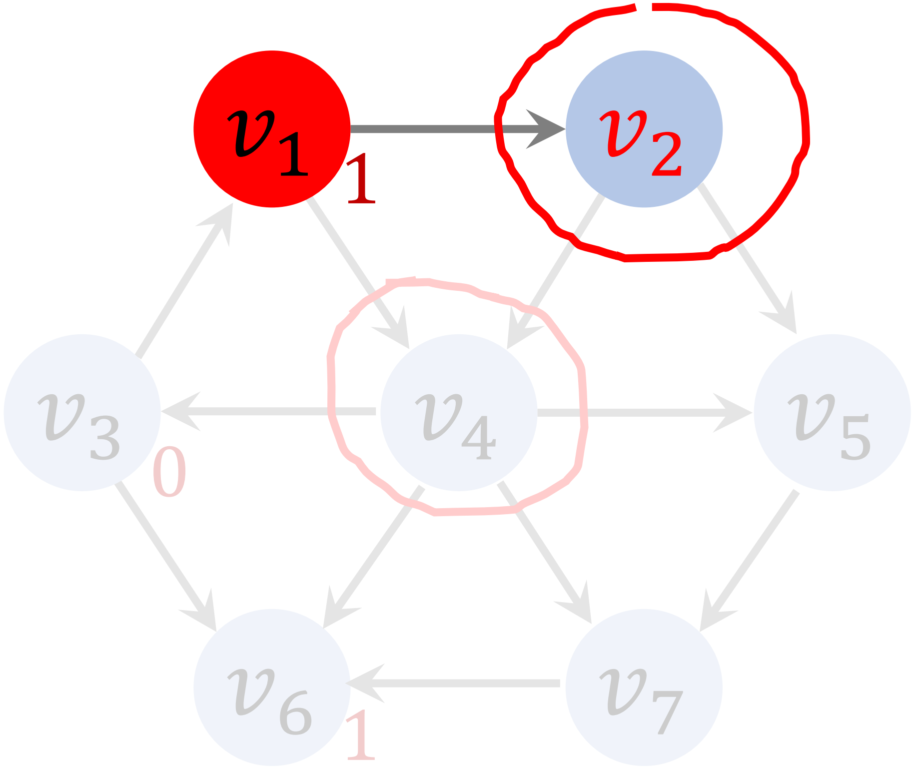
Queue:



vertex	visit	dist	path
v_1	yes	1	v_3
v_2	no	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

- Has v_2 been visited?
- No.
- ➔ Work on v_2.

Iteration 2(A)

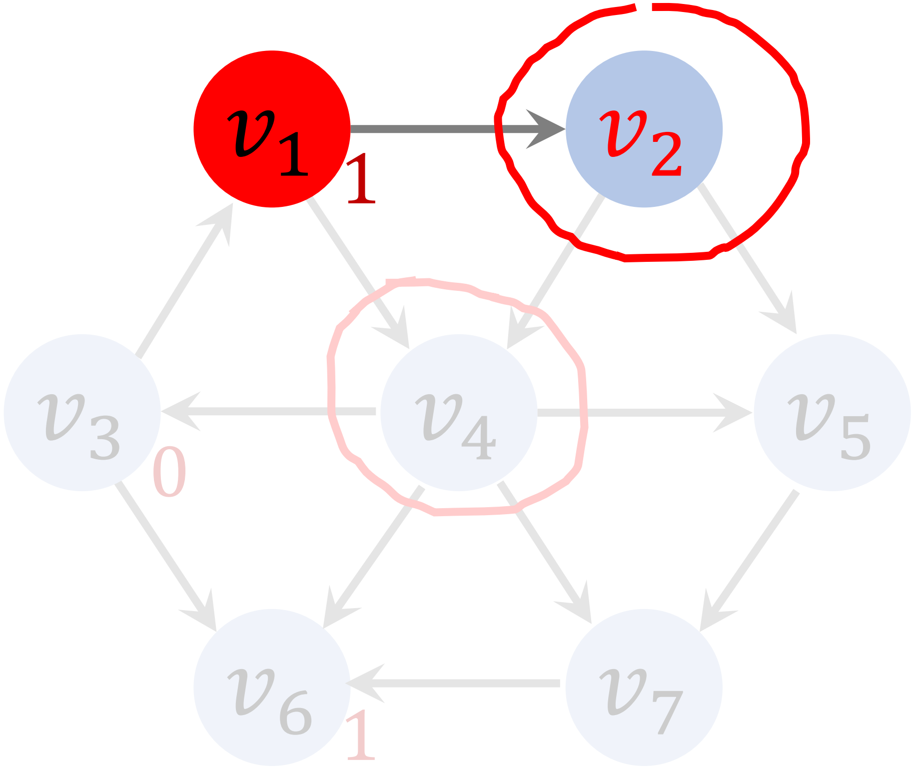


Queue:

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	no	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

- enqueue(v_2).

Iteration 2(A)

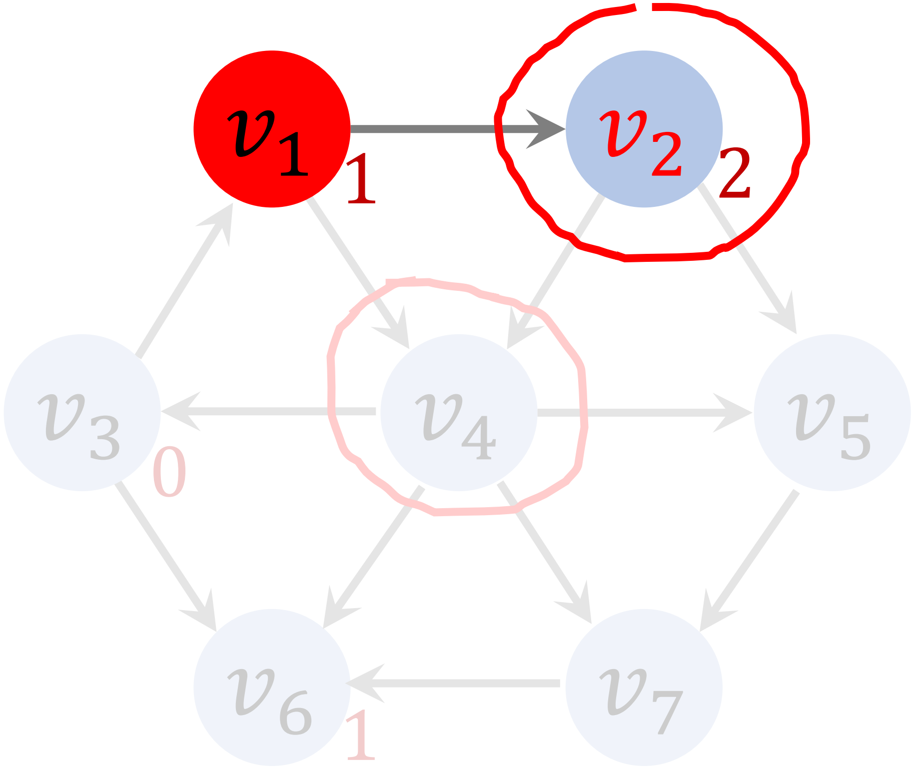


Queue:

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	∞	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

- visit[2] = true.

Iteration 2(A)

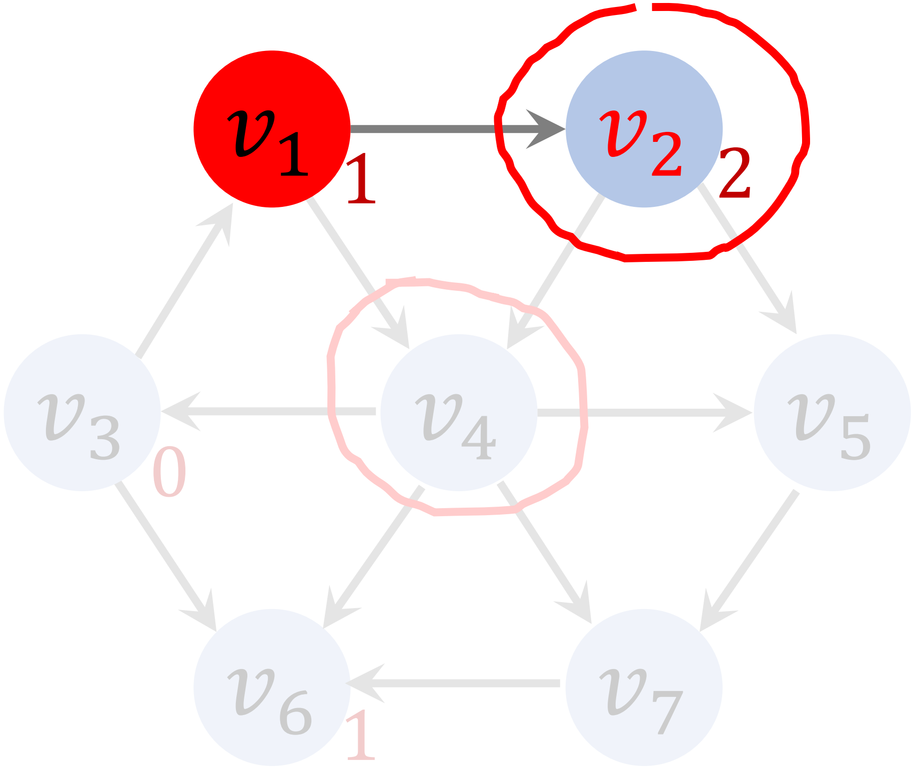


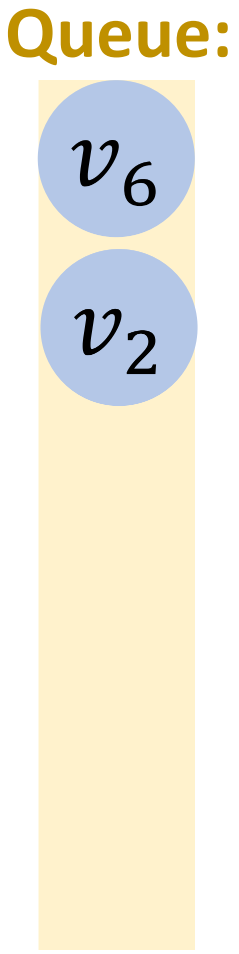
Queue:

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	0
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

- visit[2] = true.
- dist[2] = dist[1] + 1 = 2.

Iteration 2(A)

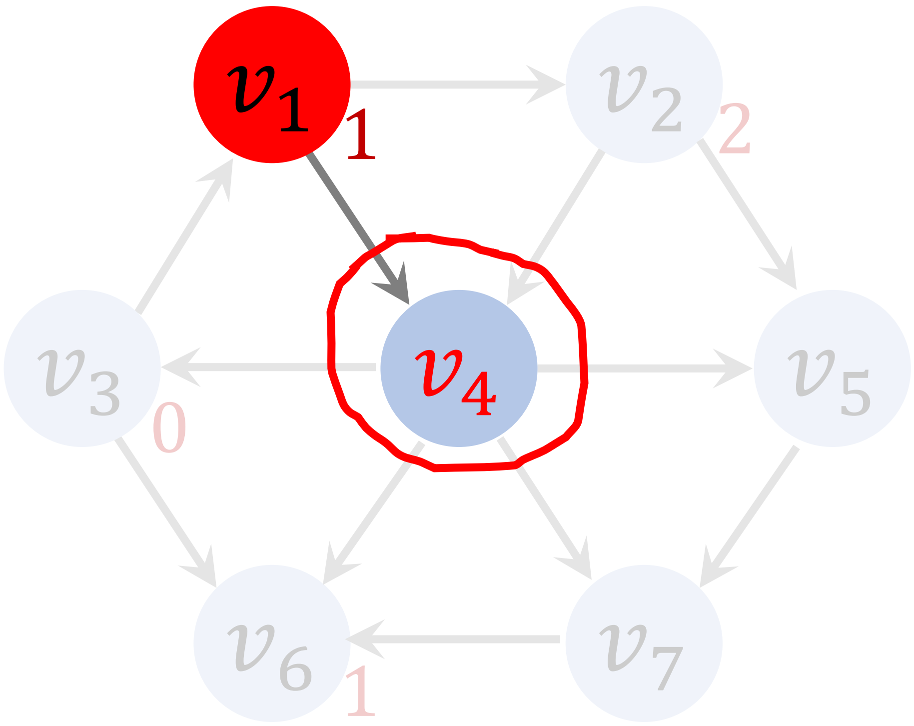




vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

- $\text{visit}[2] = \text{true}$.
- $\text{dist}[2] = \text{dist}[1] + 1 = 2$.
- $\text{path}[2] = v_1$.

Iteration 2(B)



- Has v_4 been visited?

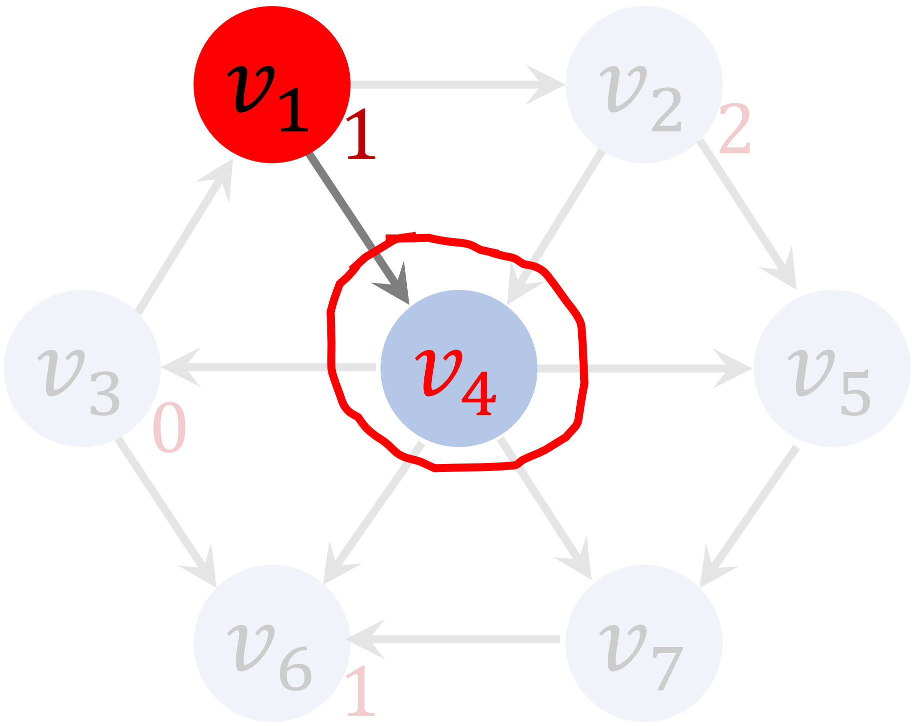
Queue:

v_6

v_2

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

Iteration 2(B)

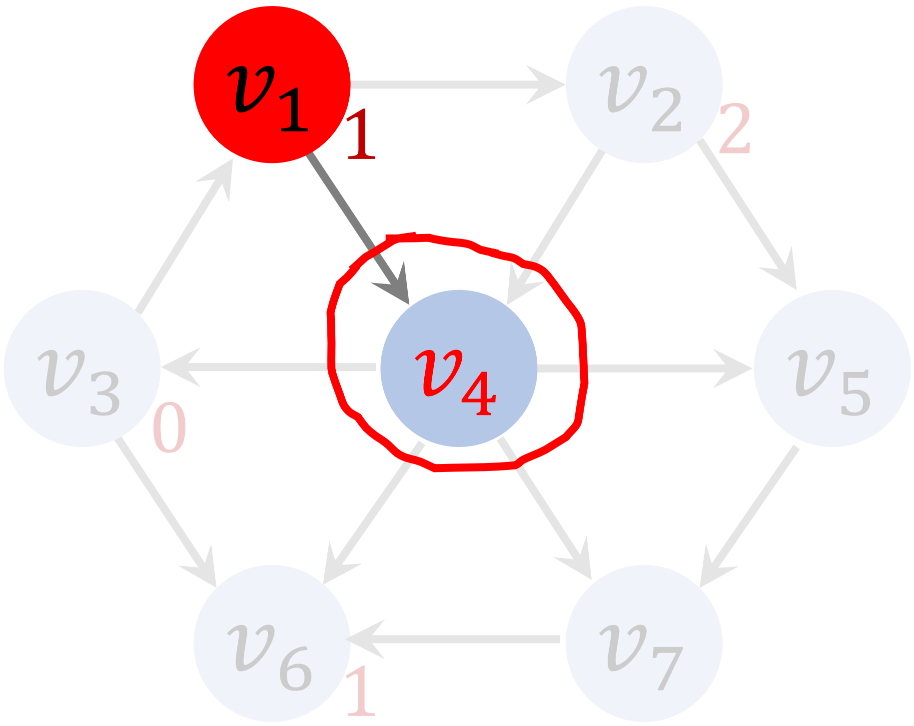


Queue:

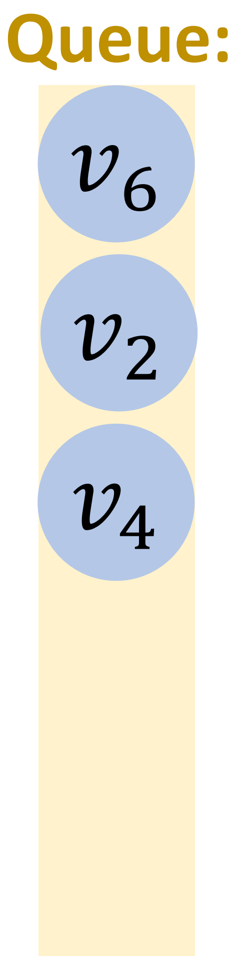
vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

- Has v_4 been visited?
- No.
- ➔ Work on v_4.

Iteration 2(B)

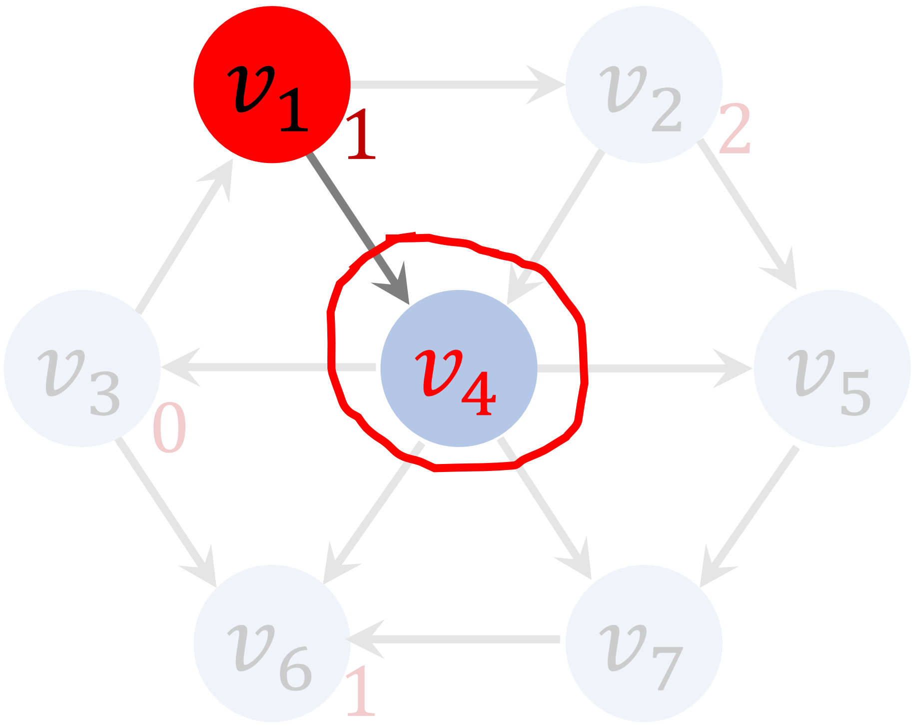


- enqueue(v_4).



vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	no	∞	0
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

Iteration 2(B)



Queue:

v_6

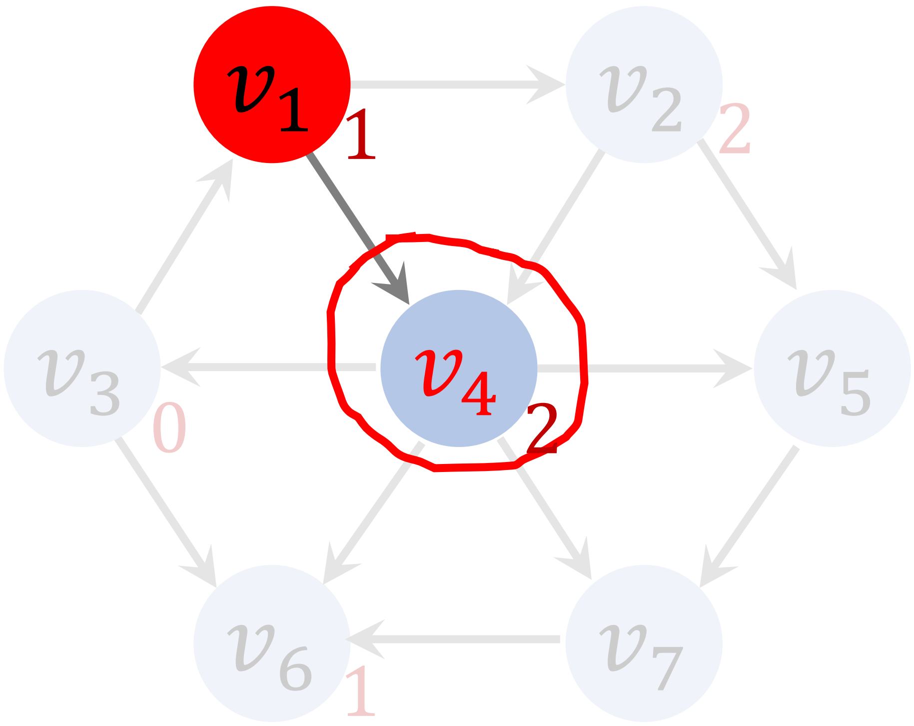
v_2

v_4

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	∞	0
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

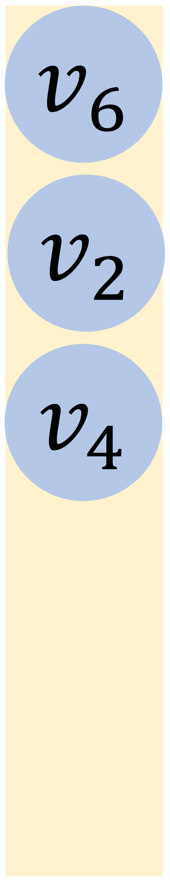
- visit[4] = true.

Iteration 2(B)



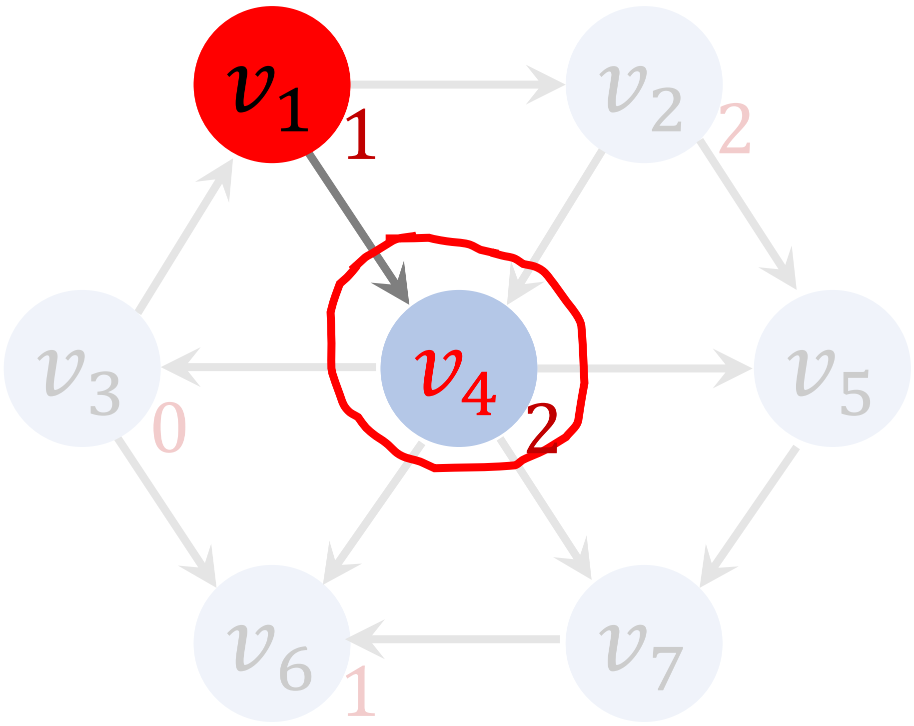
- visit[4] = true.
- dist[4] = dist[1] + 1 = 2.

Queue:



vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	0
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

Iteration 2(B)



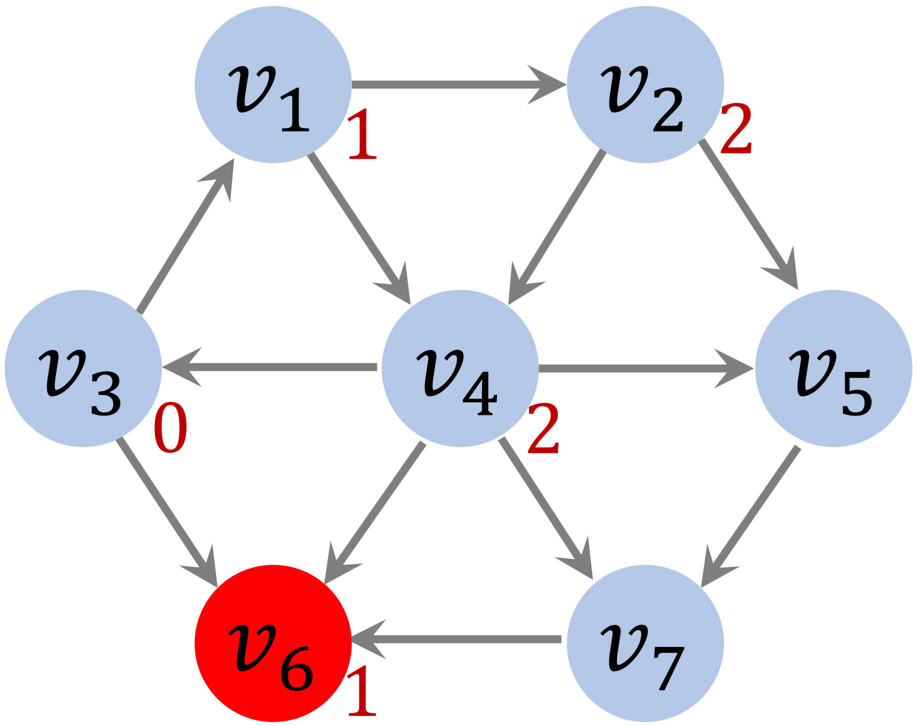
Queue:

v_6
v_2
v_4

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

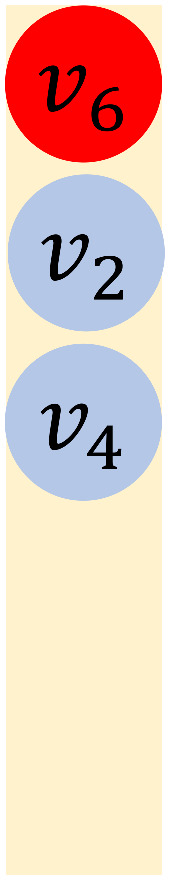
- $\text{visit}[4] = \text{true}$.
- $\text{dist}[4] = \text{dist}[1] + 1 = 2$.
- $\text{path}[4] = v_1$.

Iteration 3



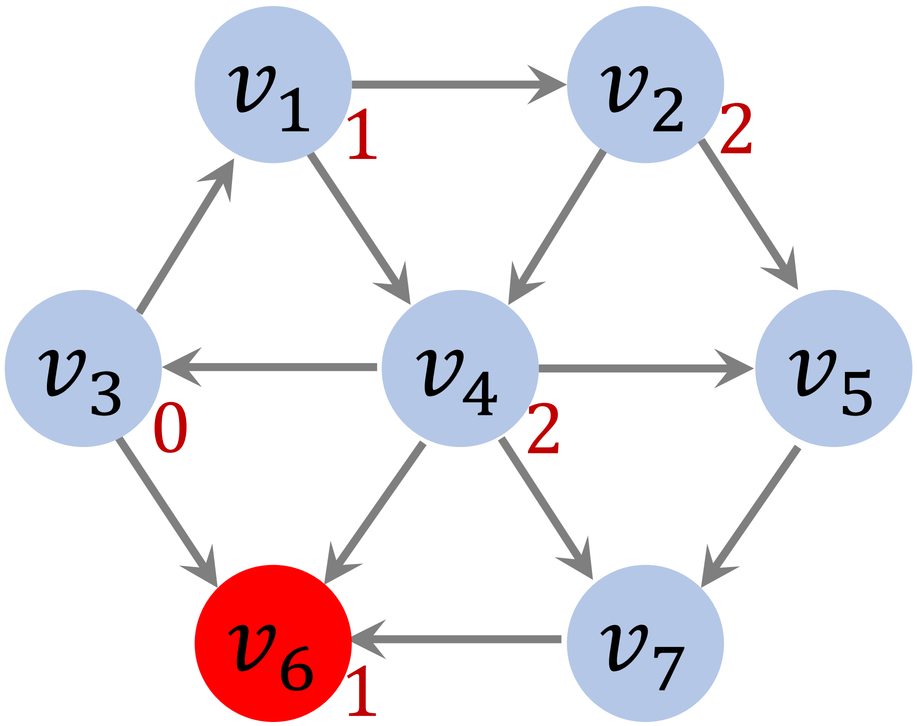
- $v_6 \leftarrow$ dequeue().

Queue:



vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

Iteration 3



Queue:

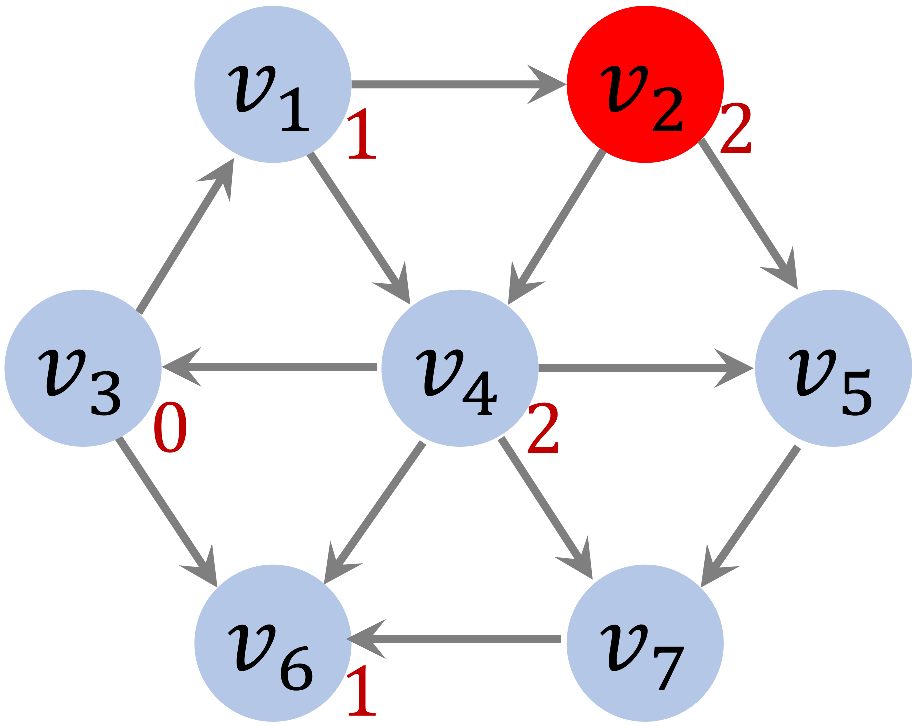
v_2

v_4

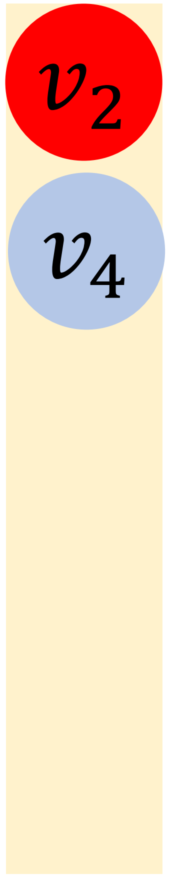
vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

- $v_6 \leftarrow$ dequeue().
- v_6 has no adjacent vertex.
- ➔ Ignore v_6.

Iteration 4



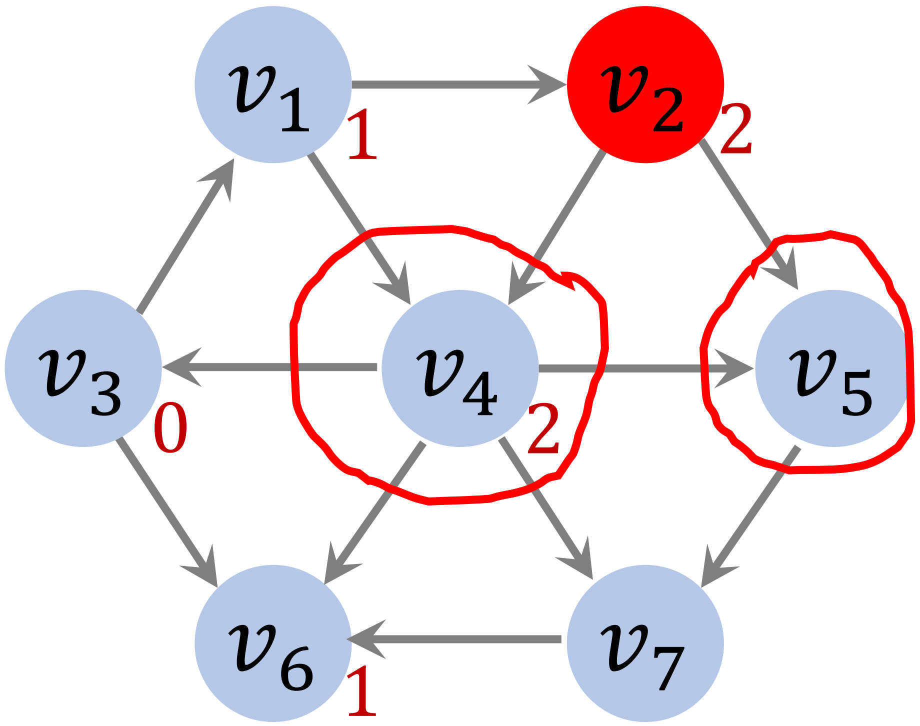
Queue:



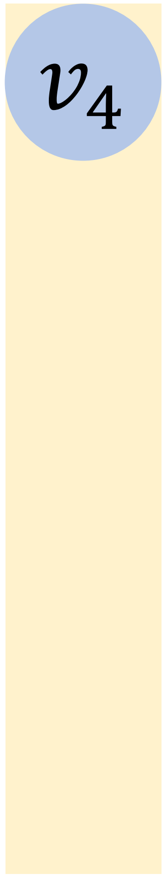
vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

- $v_2 \leftarrow$ dequeue().

Iteration 4



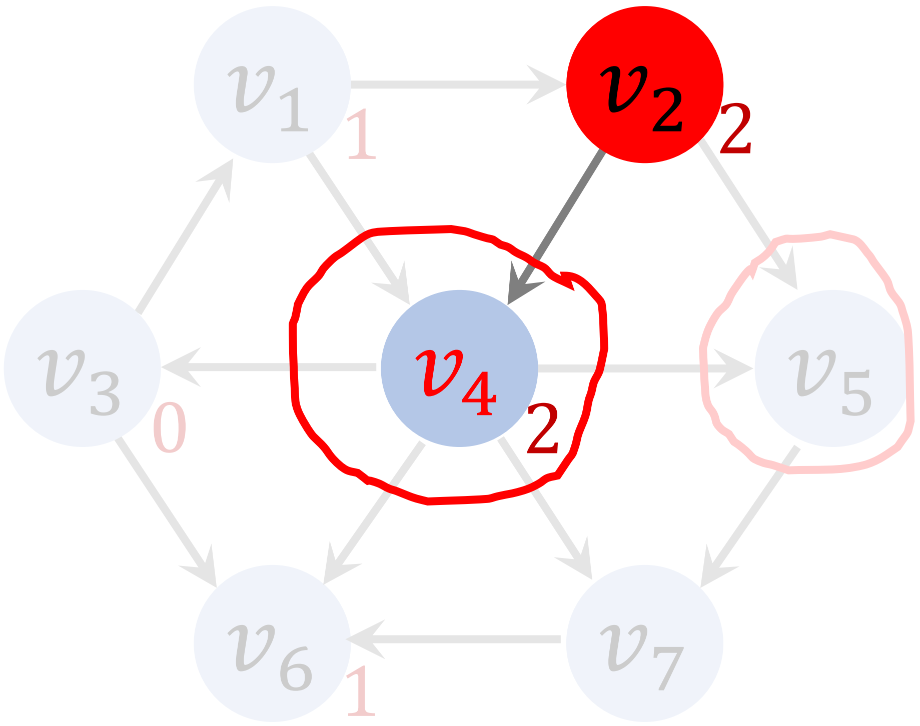
Queue:



vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

- $v_2 \leftarrow$ dequeue().
- Find adjacent vertices of v_2:
  v_4 and v_5.

Iteration 4(A)



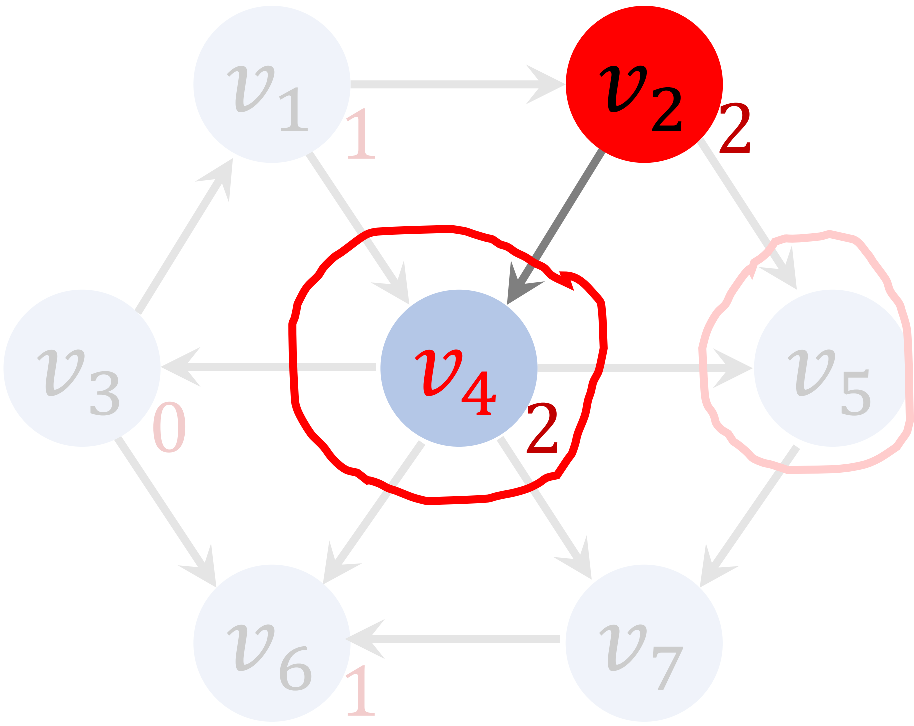
- Has v_4 been visited?

Queue:

v_4

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

Iteration 4(A)



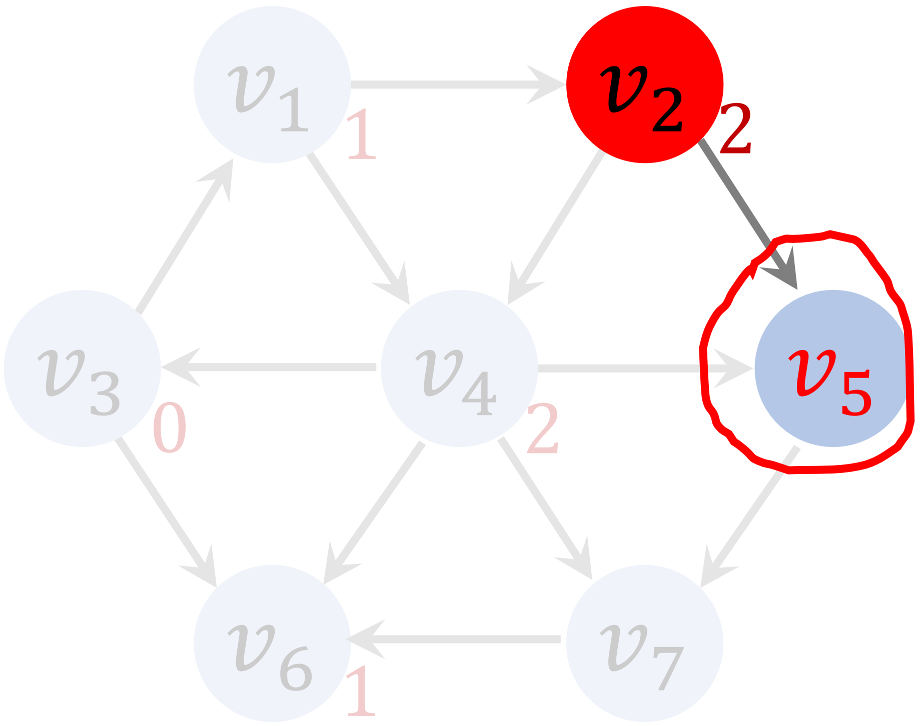
Queue:

v_4

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

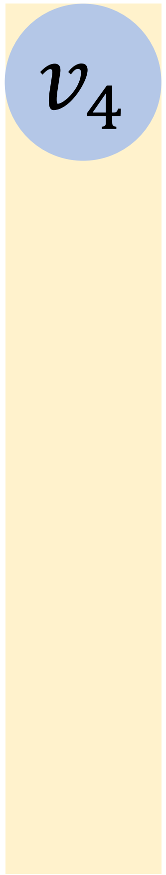
- Has v_4 been visited?
- Yes.
- ➔ Ignore v_4.

Iteration 4(B)



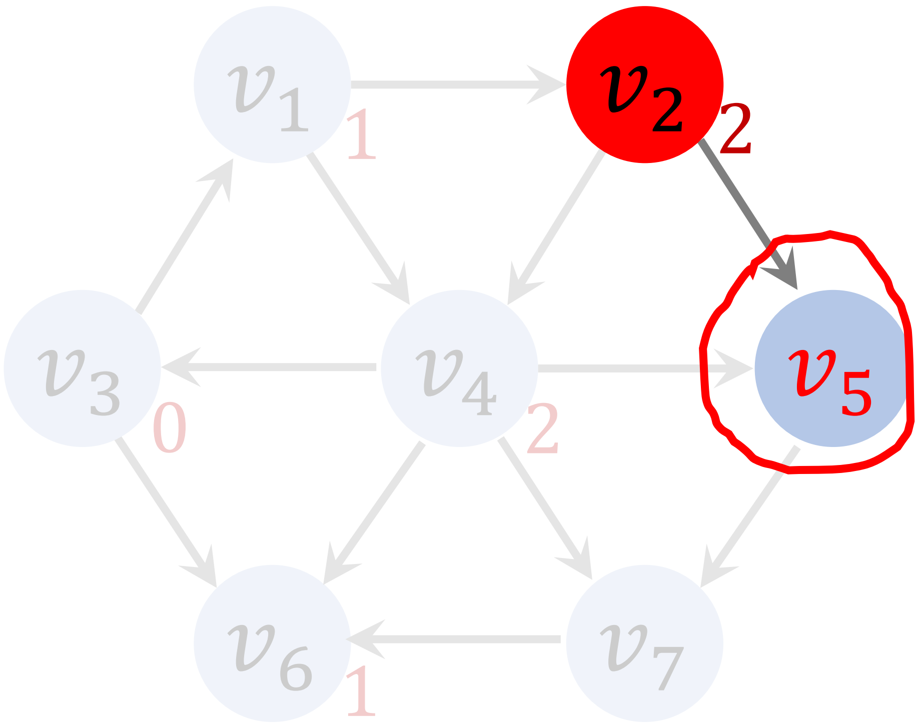
- Has v_5 been visited?

Queue:

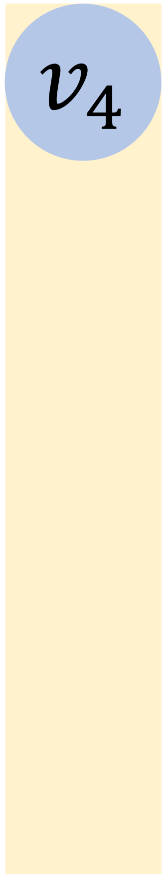


vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

Iteration 4(B)



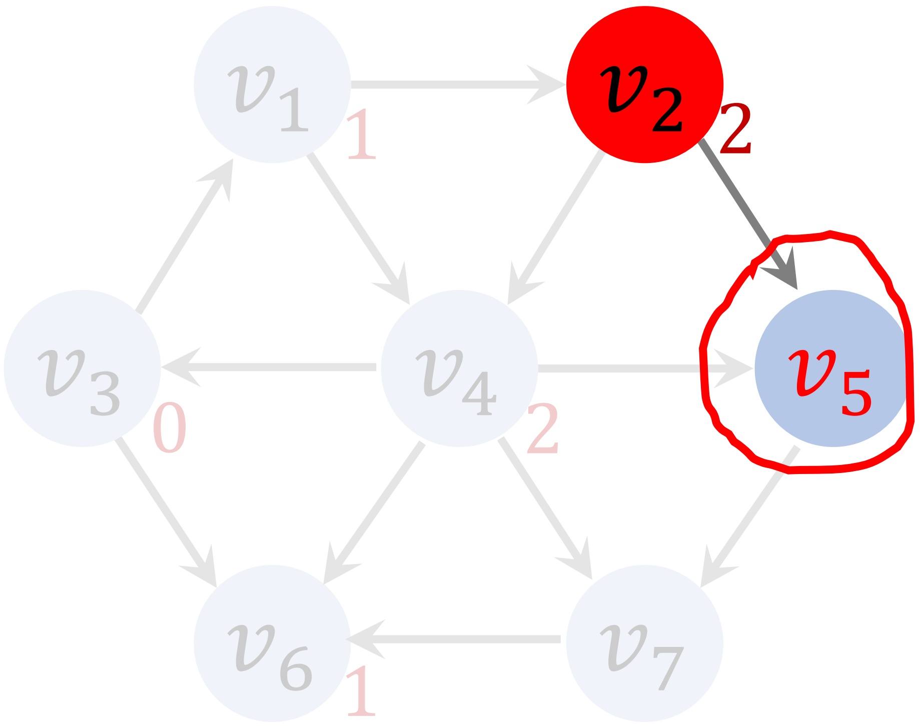
Queue:



vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

- Has v_5 been visited?
- No.
- ➔ Work on v_5.

Iteration 4(B)

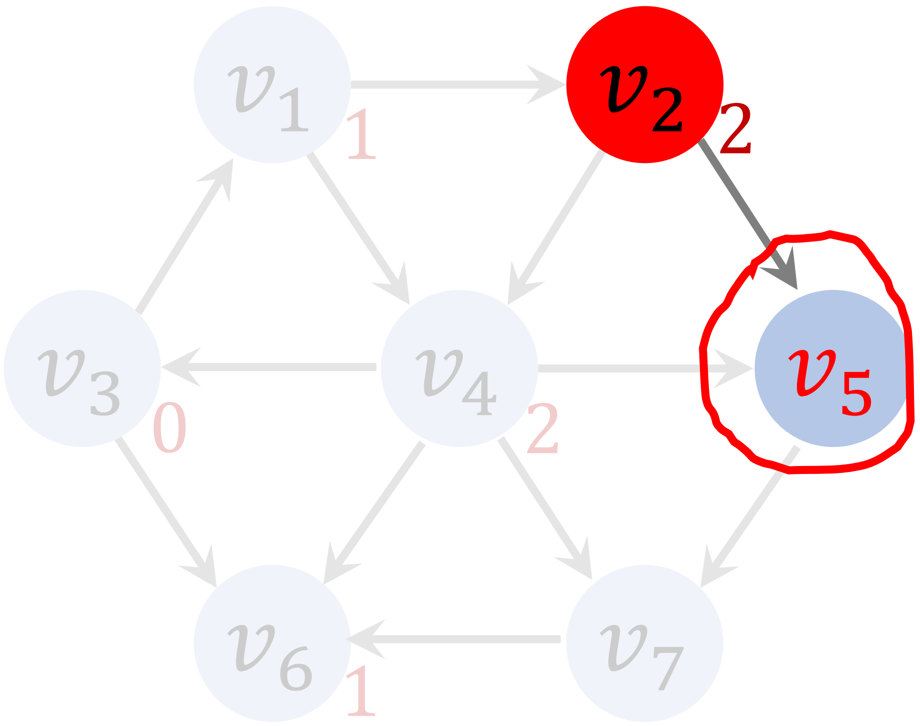


Queue:

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	no	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

- enqueue(v_5).

Iteration 4(B)

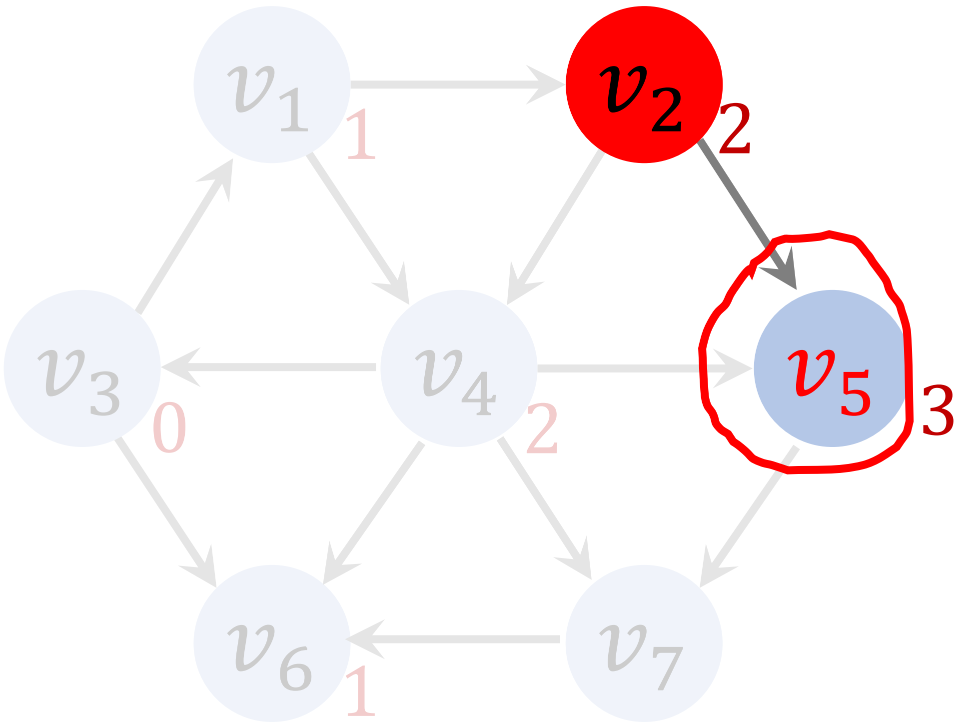


- visit[5] = true.

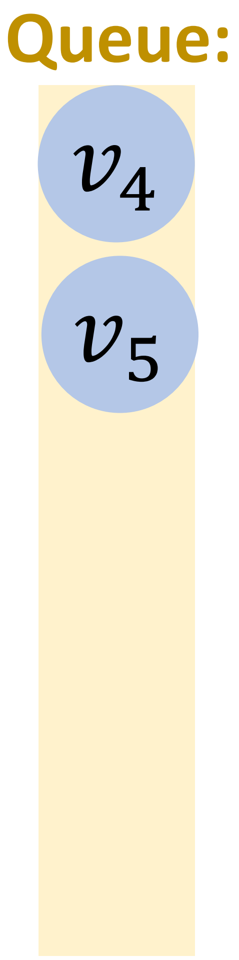
Queue:

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	∞	0
v_6	yes	1	v_3
v_7	no	∞	0

Iteration 4(B)

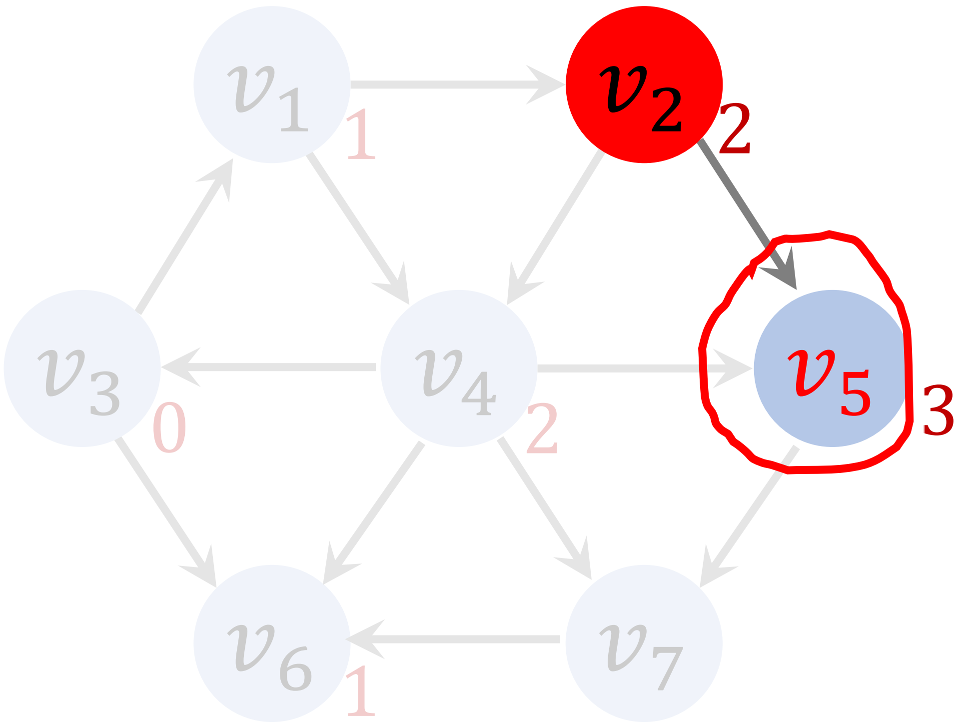


- visit[5] = true.
- dist[5] = dist[2] + 1 = 3.



vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	0
v_6	yes	1	v_3
v_7	no	∞	0

Iteration 4(B)

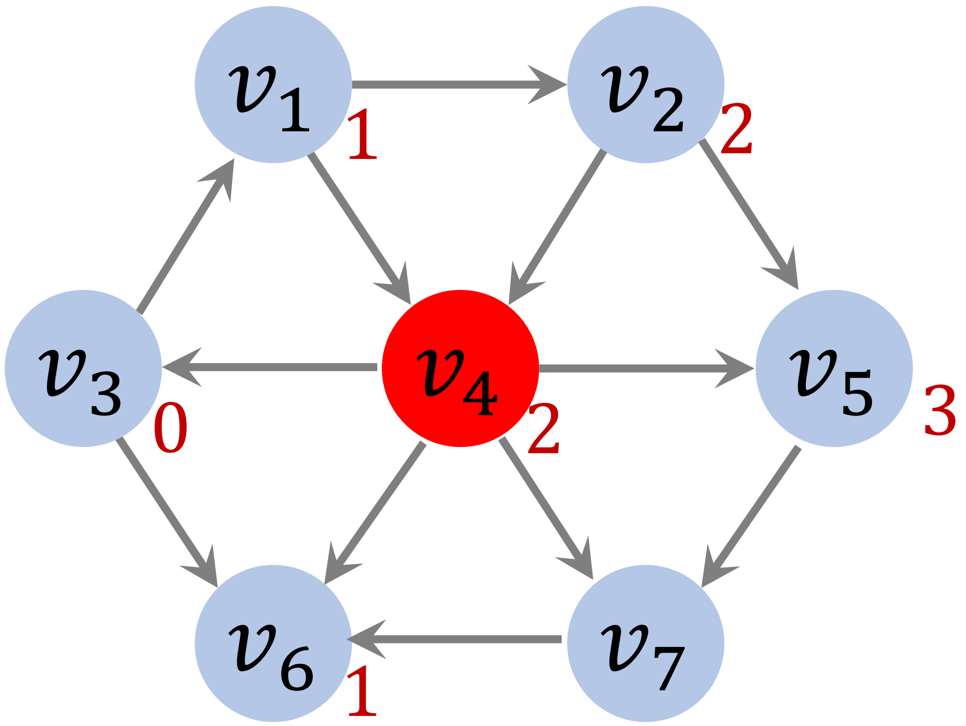


Queue:

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	no	∞	0

- $\text{visit}[5] = \text{true}.$
- $\text{dist}[5] = \text{dist}[2] + 1 = 3.$
- $\text{path}[5] = v_2.$

Iteration 5

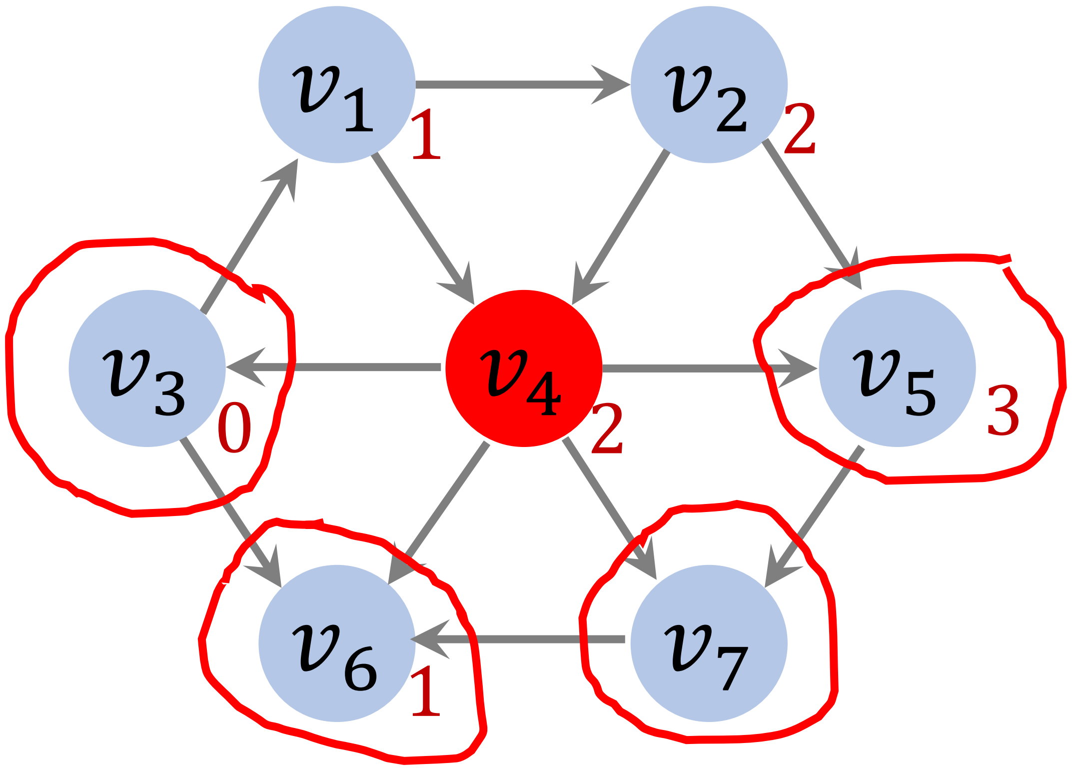


- $v_4 \leftarrow$ dequeue().

Queue:

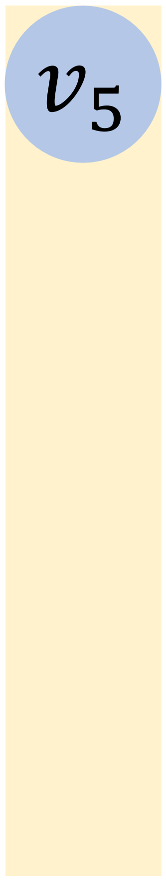
vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	no	∞	0

Iteration 5



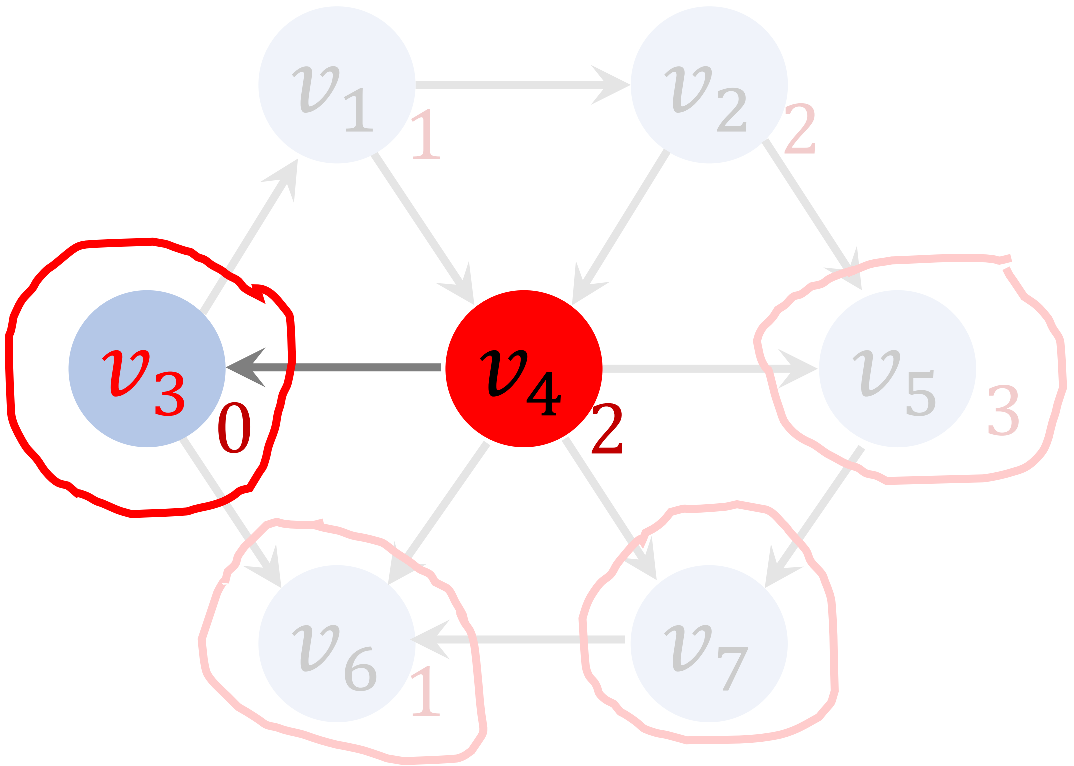
- $v_4 \leftarrow$ dequeue().
- Find adjacent vertices of v_4:
  v_3, v_5, v_6, and v_7.

Queue:

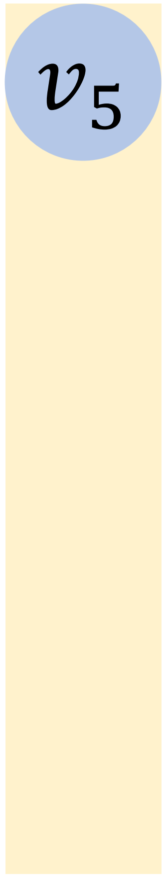


vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	no	∞	0

Iteration 5(A)



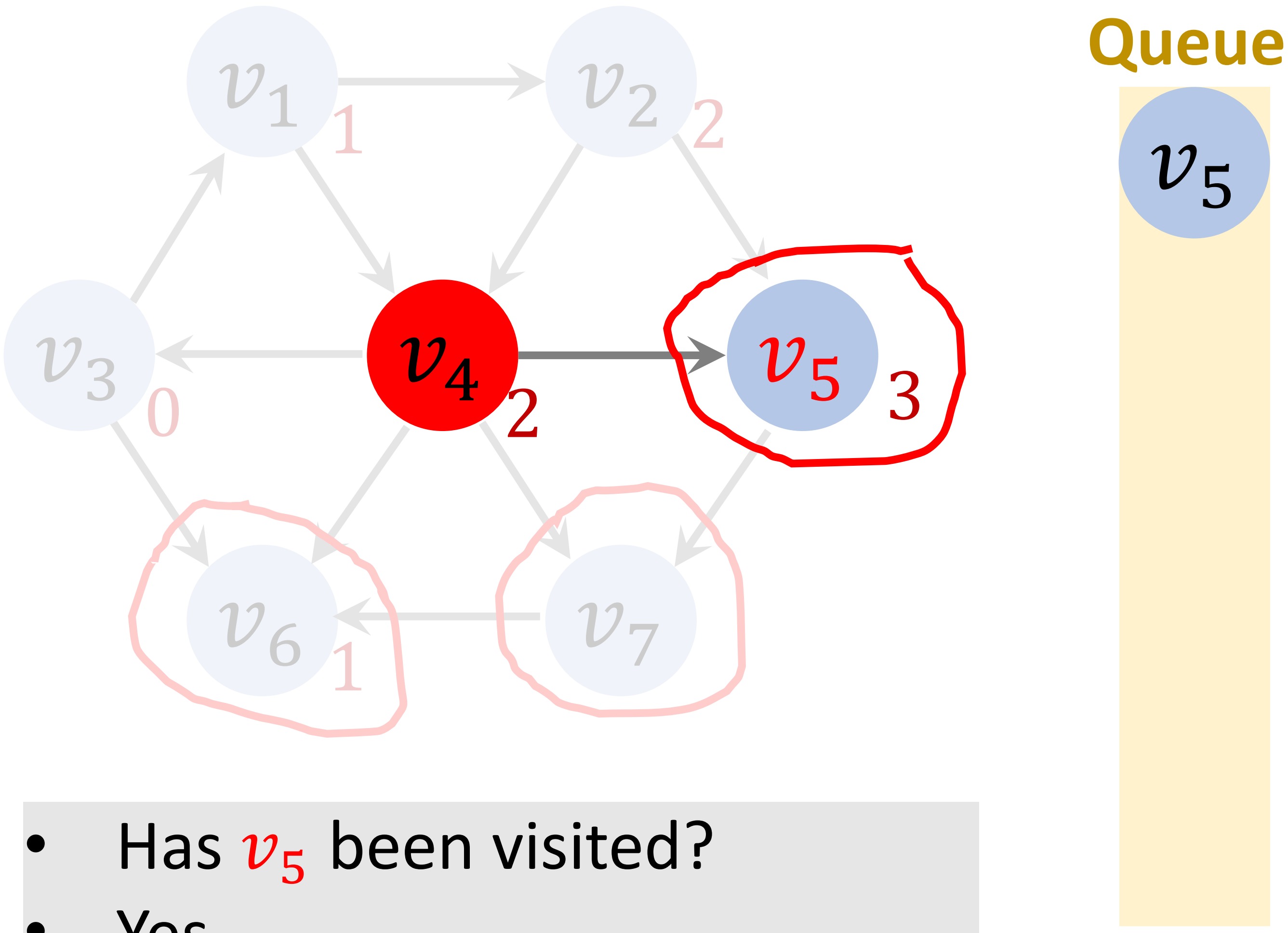
Queue:



vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	no	∞	0

- Has v_3 been visited?
- Yes.
- ➔ Ignore v_3.

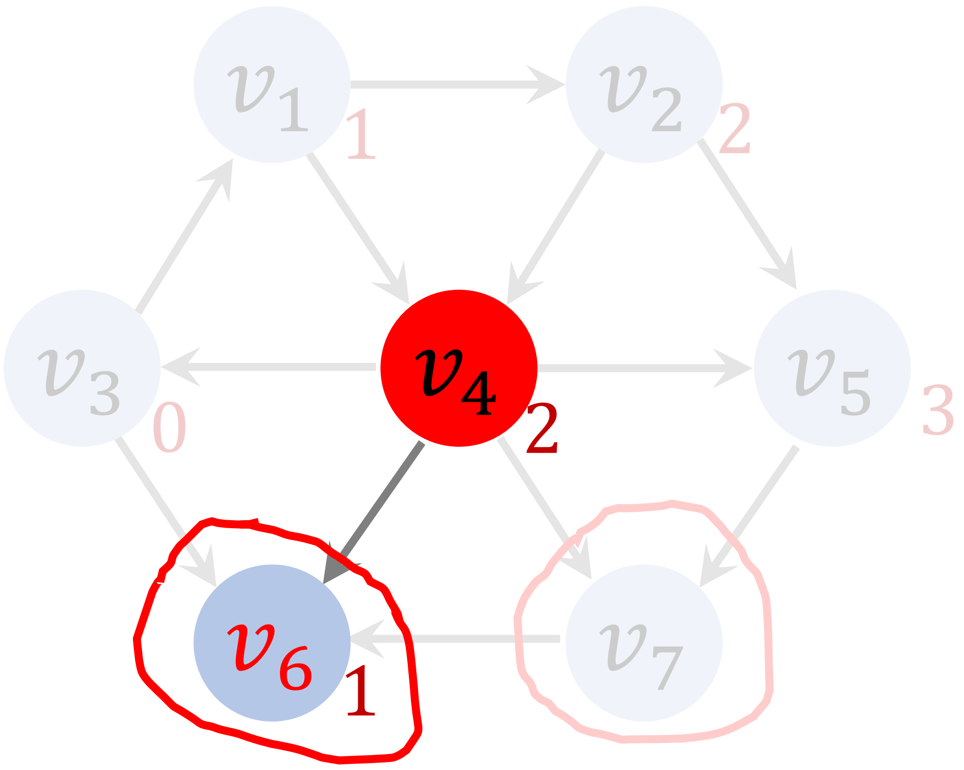
Iteration 5(B)



vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	no	∞	0

- Has v_5 been visited?
- Yes.
- ➔ Ignore v_5.

Iteration 5(C)



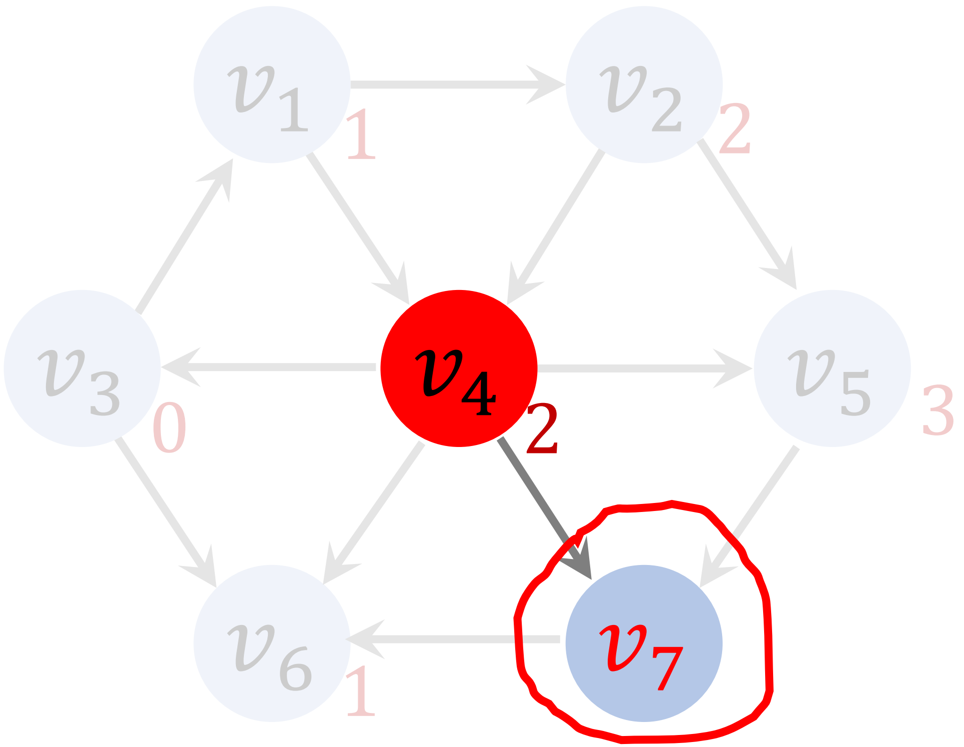
- Has v_6 been visited?
- Yes.
- ➔ Ignore v_6.

Queue:

v_5

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	no	∞	0

Iteration 5(D)



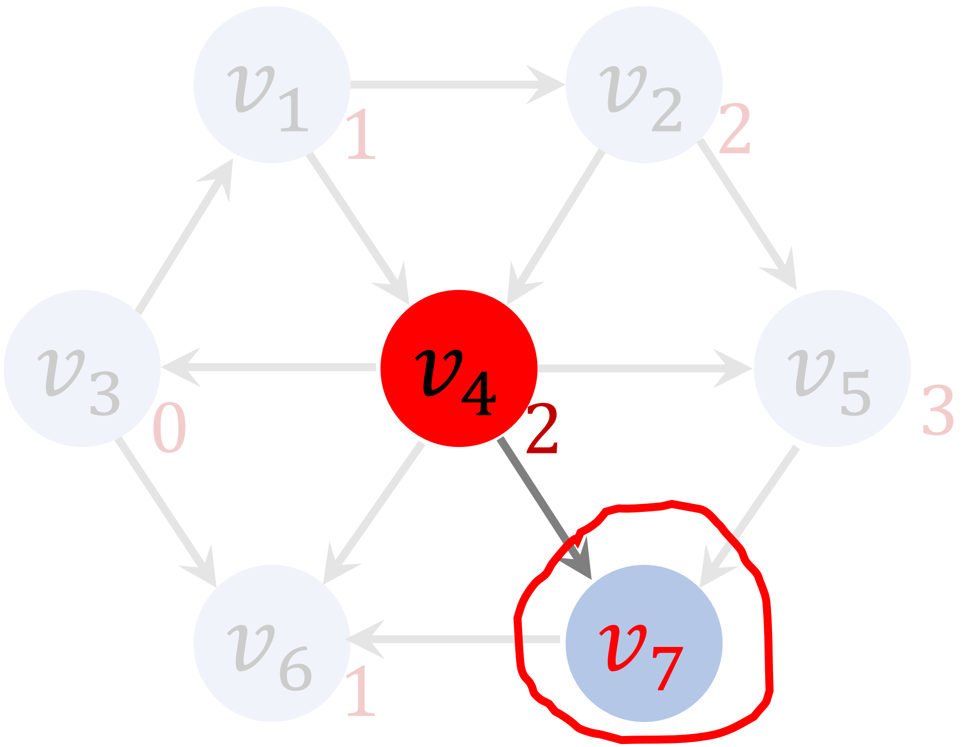
- Has v_7 been visited?

Queue:

v_5

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	no	∞	0

Iteration 5(D)



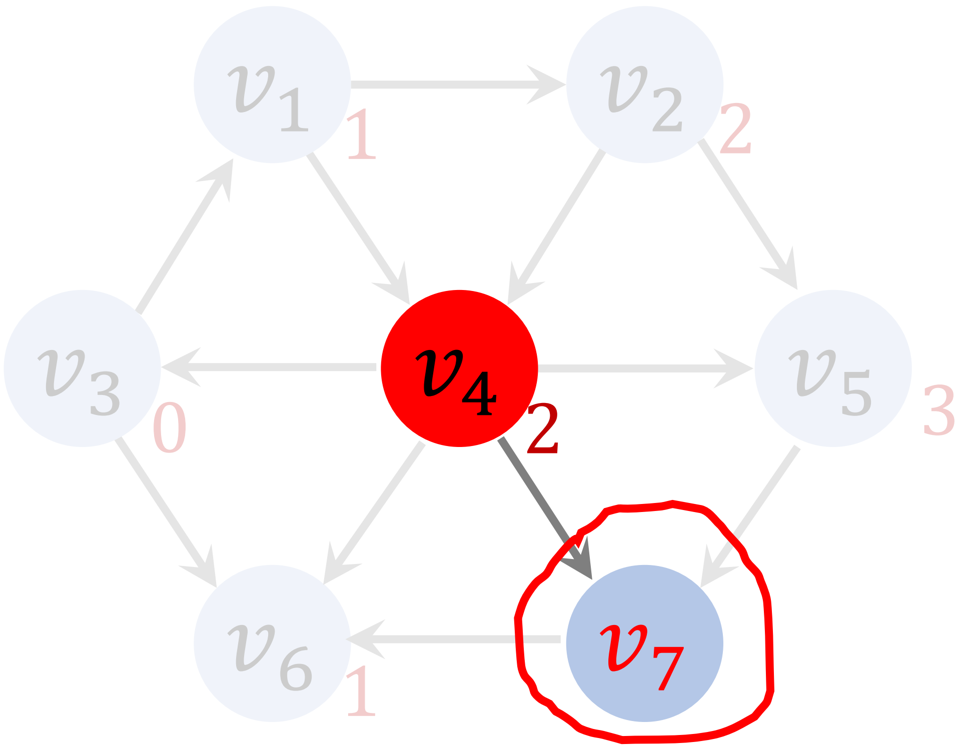
- Has v_7 been visited?
- No.
- ➔ Work on v_7.

Queue:

v_5

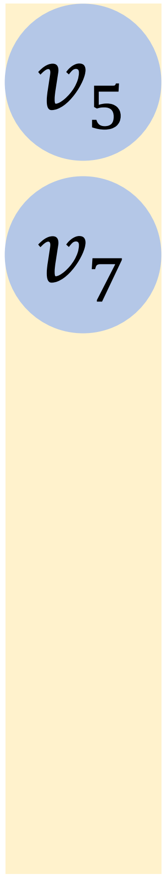
vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	no	∞	0

Iteration 5(D)



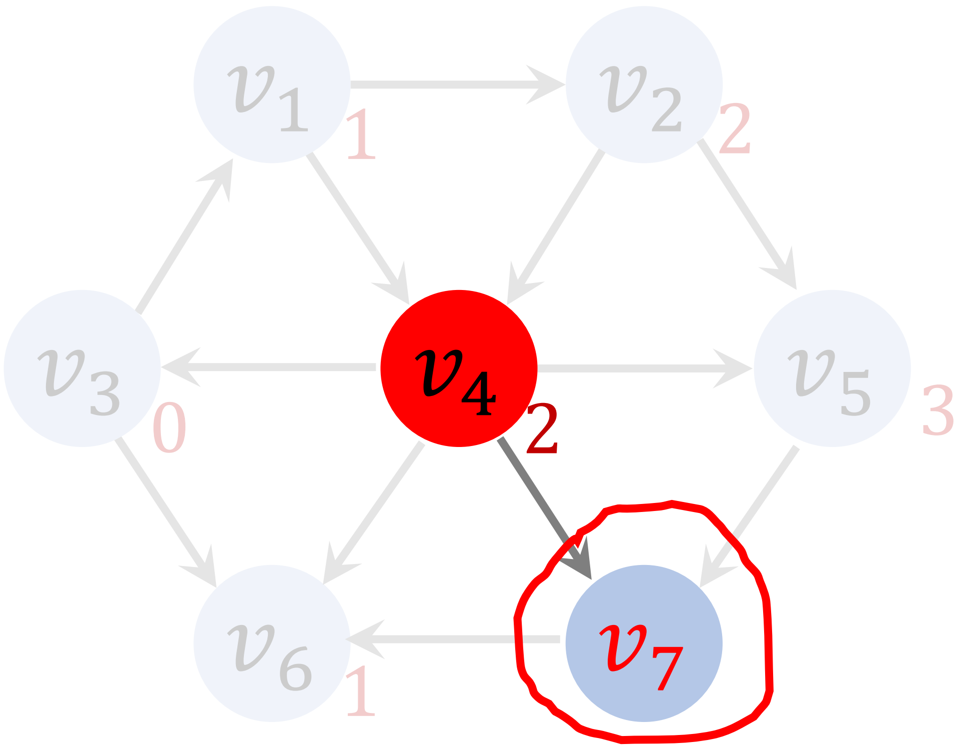
- enqueue(v_7).

Queue:

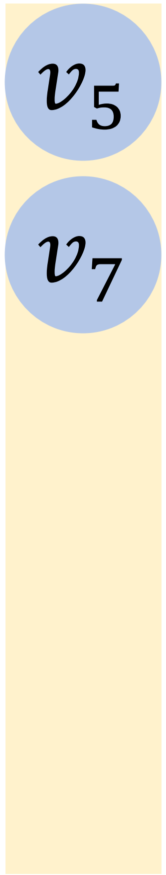


vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	no	∞	0

Iteration 5(D)



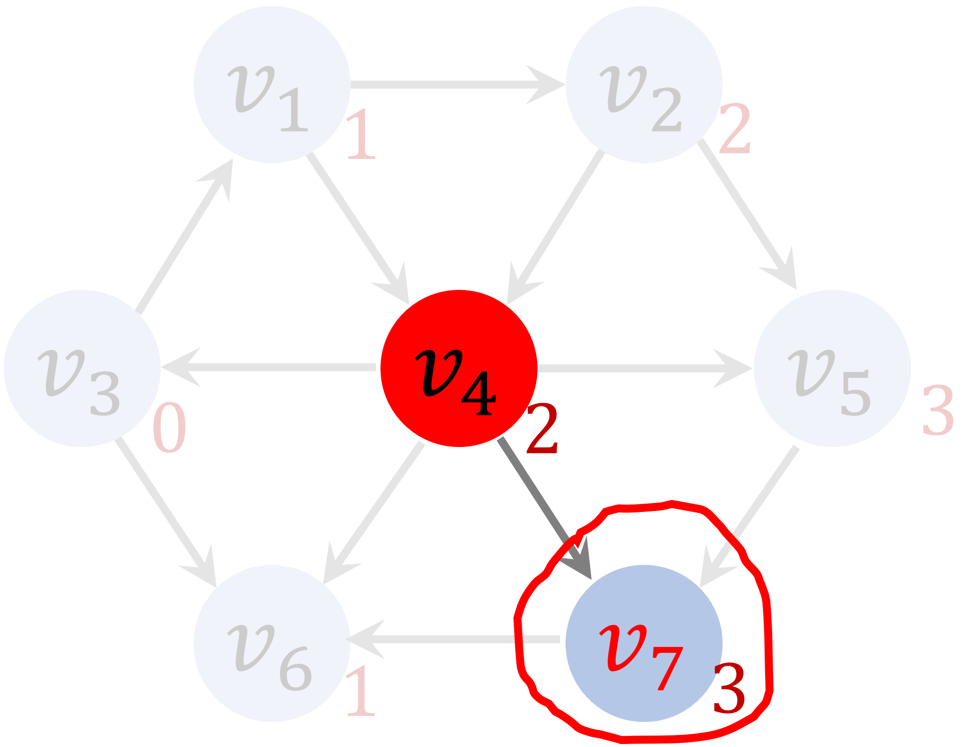
Queue:



vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	yes	∞	0

- $\text{visit}[7] = \text{true}.$

Iteration 5(D)



- visit[7] = true.
- dist[7] = dist[4] + 1 = 3.

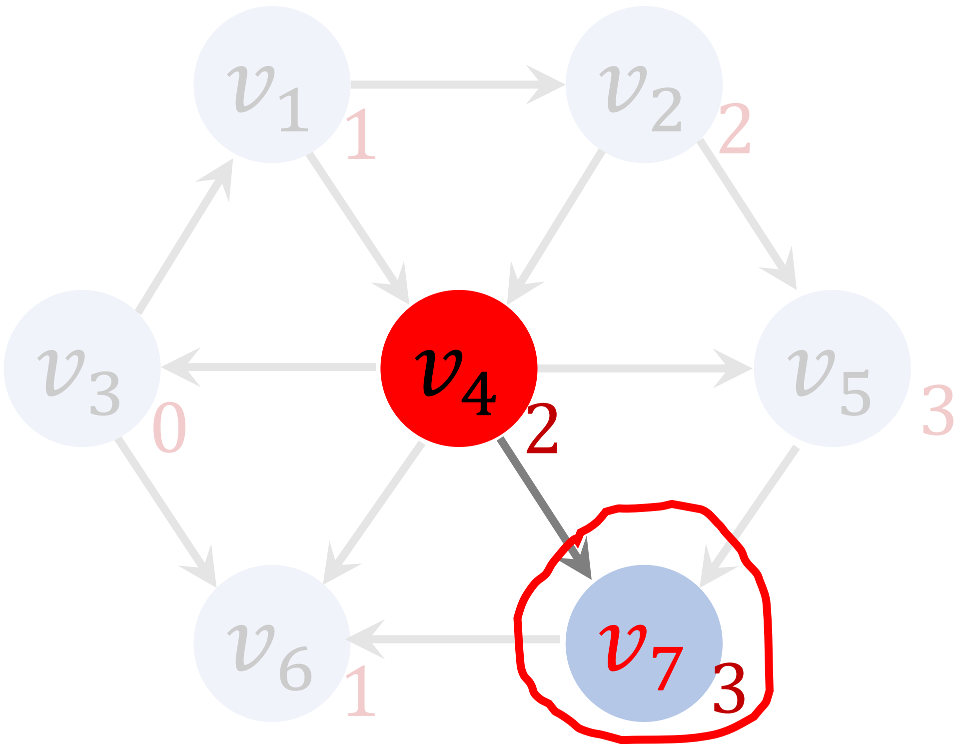
Queue:

v_5

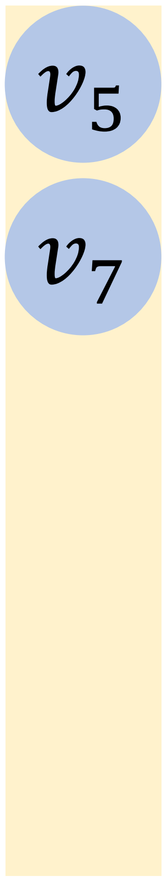
v_7

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	yes	3	0

Iteration 5(D)



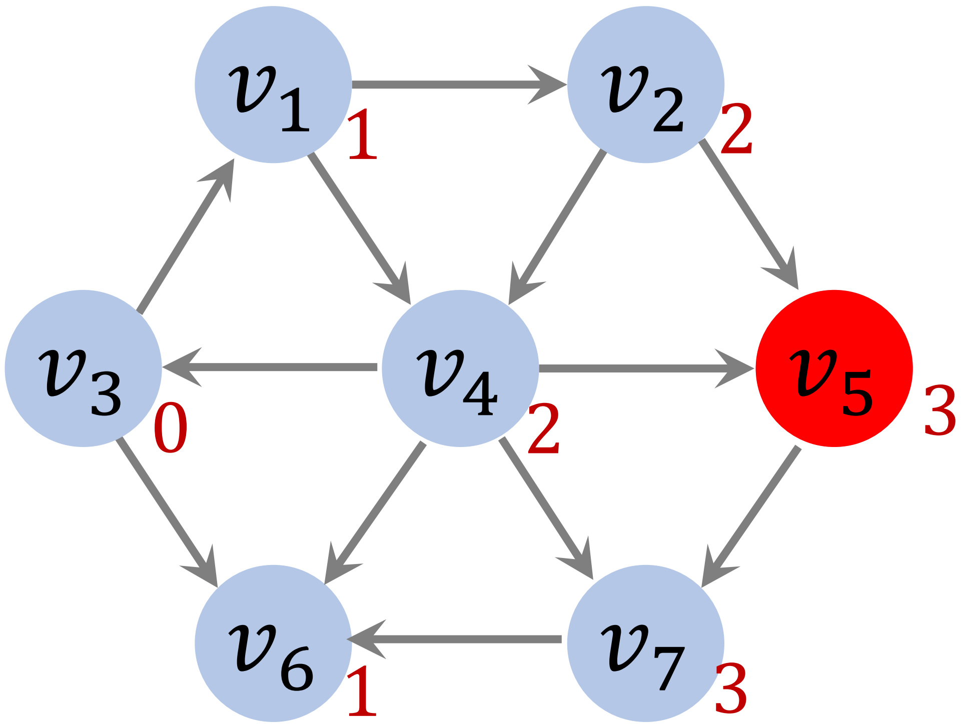
Queue:



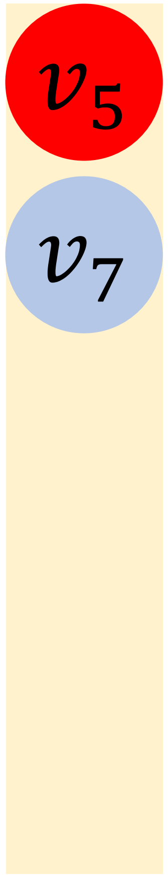
vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	yes	3	v_4

- visit[7] = true.
- dist[7] = dist[4] + 1 = 3.
- path[7] = v_4.

Iteration 6



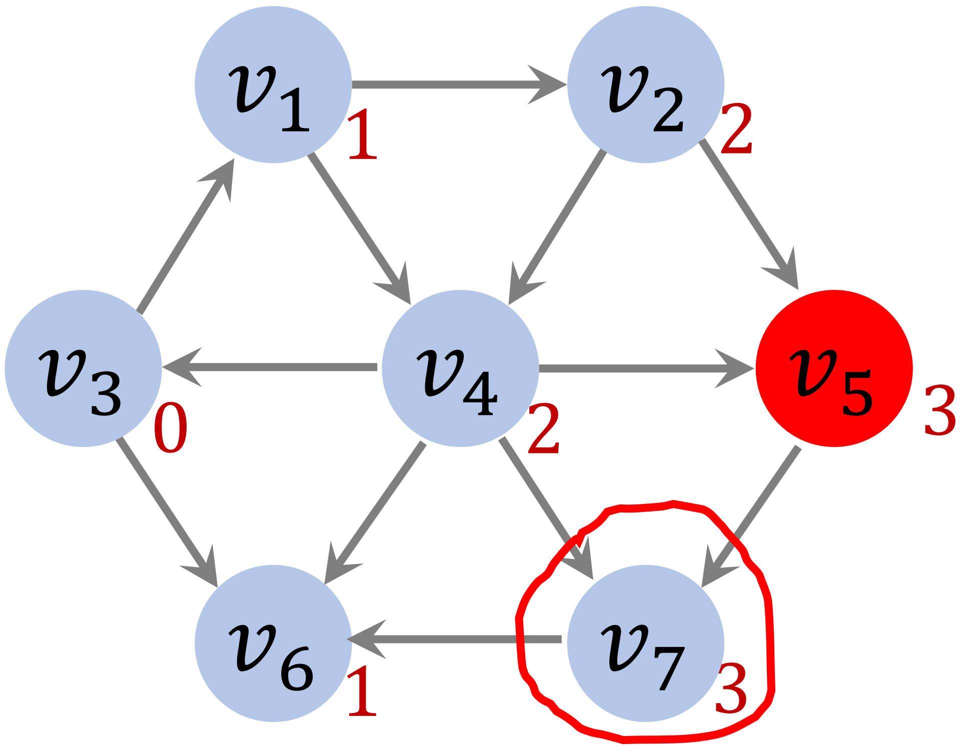
Queue:

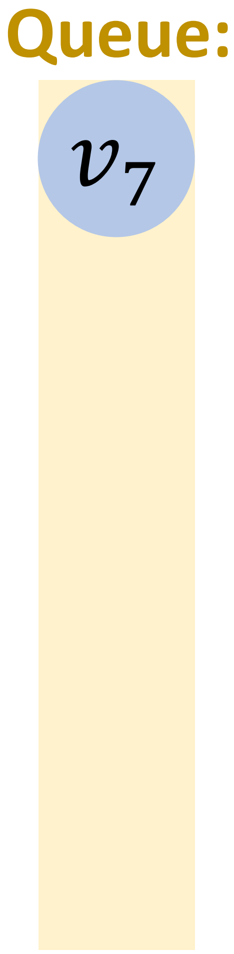


vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	yes	3	v_4

- $v_5 \leftarrow$ dequeue().

Iteration 6

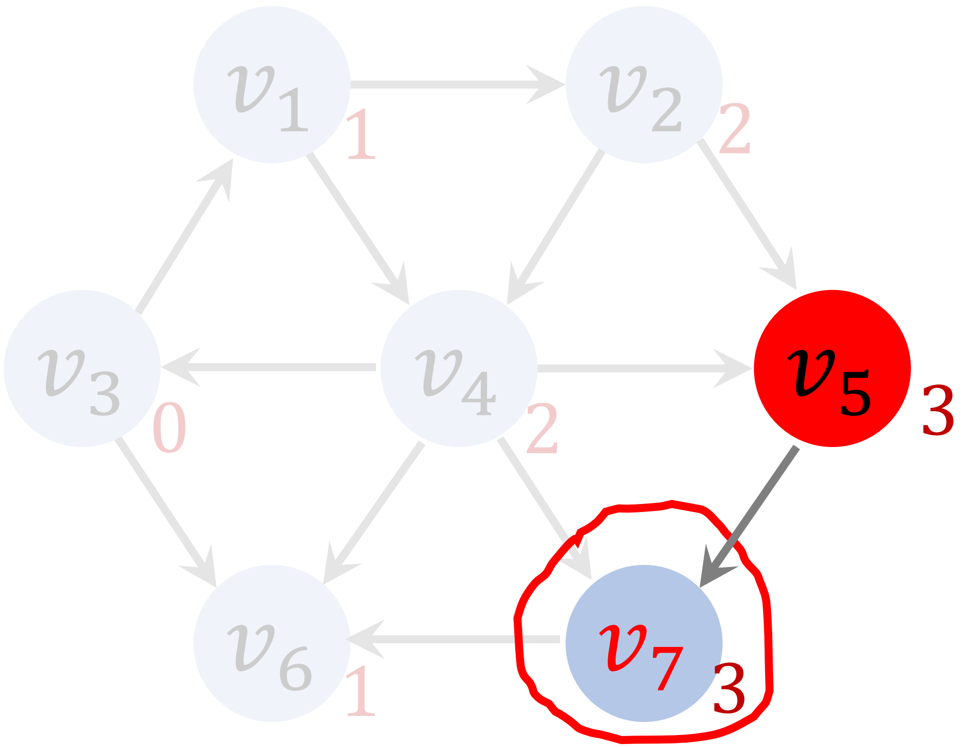




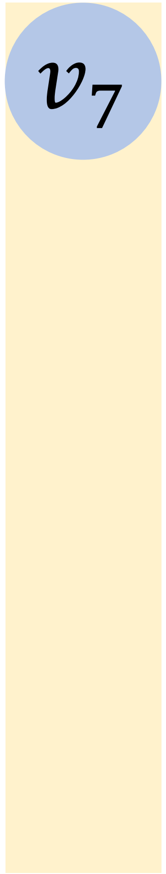
vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	yes	3	v_4

- $v_5 \leftarrow$ dequeue().
- Find adjacent vertices of v_5: v_7.

Iteration 6(A)



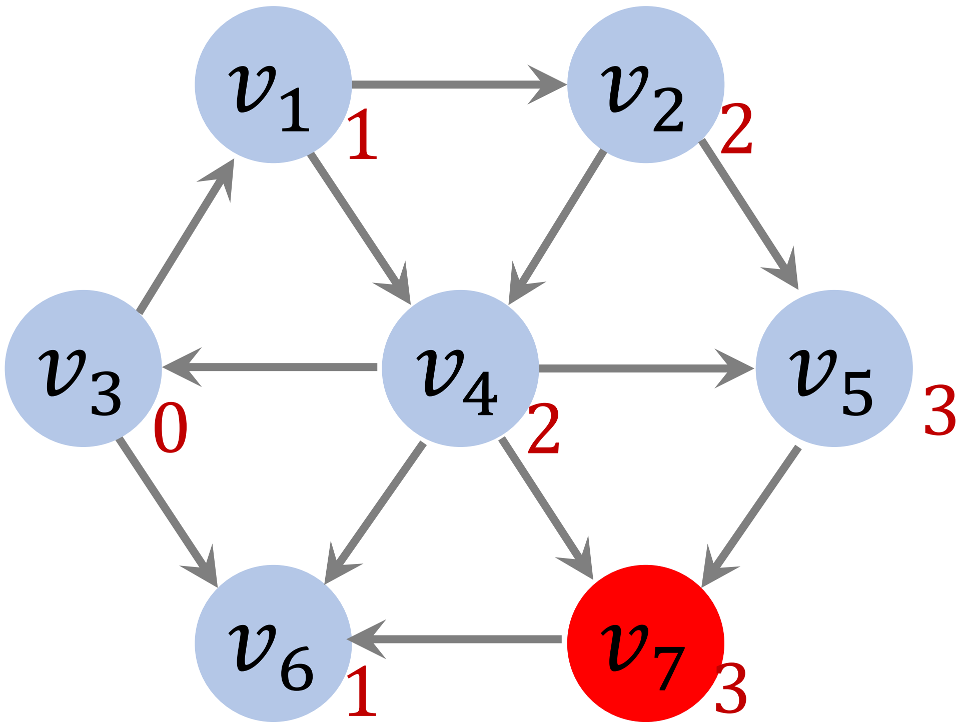
Queue:



vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	yes	3	v_4

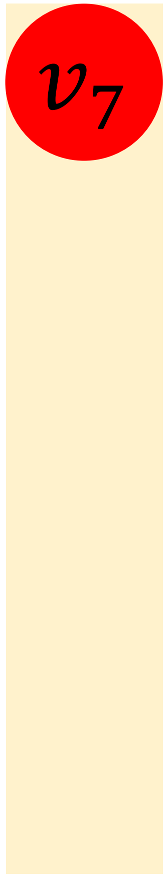
- Has v_7 been visited?
- Yes.
- ➔ Ignore v_7.

Iteration 7



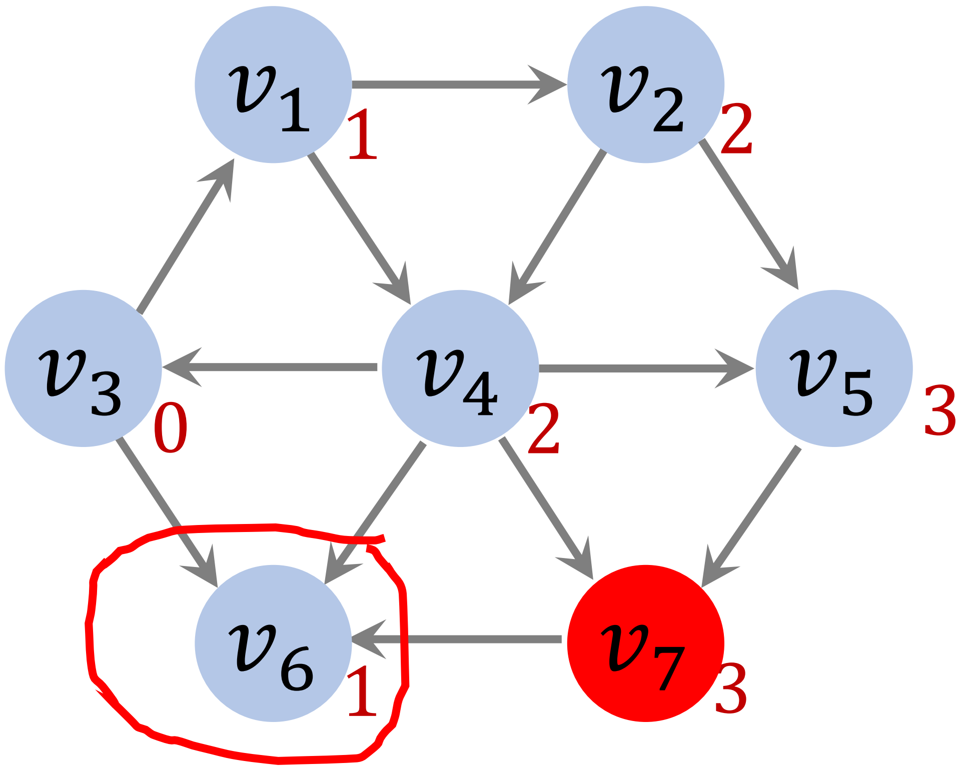
- $v_7 \leftarrow$ dequeue().

Queue:



vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	yes	3	v_4

Iteration 7

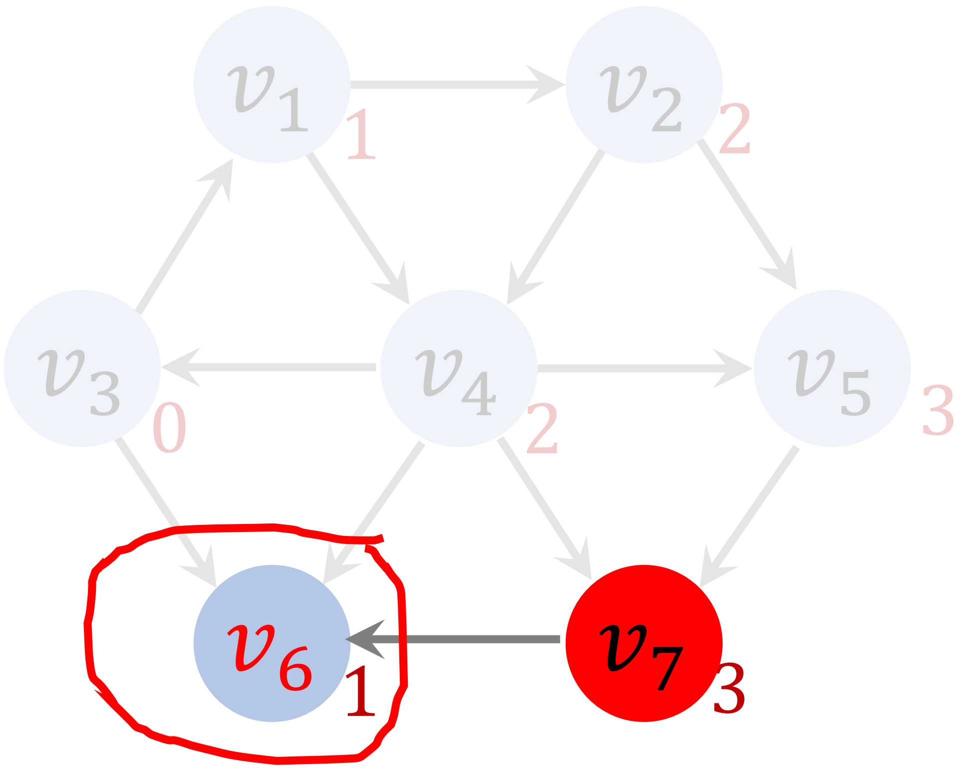


- $v_7 \leftarrow$ dequeue().
- Find adjacent vertices of v_7: v_6.

Queue:

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	yes	3	v_4

Iteration 7(A)

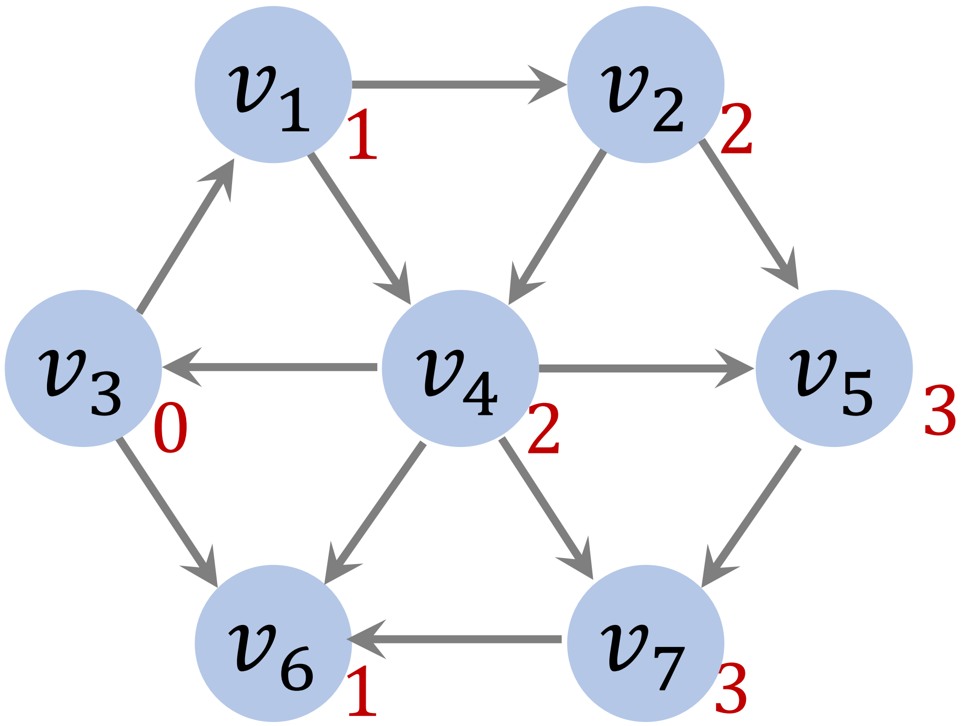


Queue:

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	yes	3	v_4

- Has v_6 been visited?
- Yes.
- ➔ Ignore v_6.

End of Procedure



- The queue is empty.
- ➔ End of procedure.

Queue:

vertex	visit	dist	path
v_1	yes	1	v_3
v_2	yes	2	v_1
v_3	yes	0	0
v_4	yes	2	v_1
v_5	yes	3	v_2
v_6	yes	1	v_3
v_7	yes	3	v_4

Pseudo Code

Inputs: vertices $\mathcal{V}$, edges $\mathcal{E}$, and the source vertex s.

1. Initialize an empty queue.

Pseudo Code

Inputs: vertices $\mathcal{V}$, edges $\mathcal{E}$, and the source vertex s.

1. Initialize an empty queue.
2. For each vertex $v \in \mathcal{V}$:
   a. Set $\text{visit}[v] = \text{false}$.
   b. Set $\text{dist}[v] = \infty$.
   c. Set $\text{path}[v] = 0$.

vertex	**visit**	**dist**	**path**
v_1	false	∞	0
v_2	false	∞	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$
v_n	false	∞	0

Pseudo Code

Inputs: vertices $\mathcal{V}$, edges $\mathcal{E}$, and the source vertex s.

1. Initialize an empty queue.
2. For each vertex $v \in \mathcal{V}$:
   a. Set $\text{visit}[v] = \text{false}$.
   b. Set $\text{dist}[v] = \infty$.
   c. Set $\text{path}[v] = 0$.
3. $\text{enqueue}(s)$.
4. Set $\text{visit}[s] = \text{true}$ and $\text{dist}[s] = 0$.

Pseudo Code (Cont.)

5. While the queue is not empty:
   a. $v \leftarrow \text{dequeue}()$.
   b. $\mathcal{S} \leftarrow \{u \mid e_{vu} \in \mathcal{E}$ and $\text{visit}[u] = \text{true}\}$.

Pseudo Code (Cont.)

5. While the queue is not empty:
   a. $v \leftarrow \text{dequeue}()$.
   b. $\mathcal{S} \leftarrow \{u \mid e_{vu} \in \mathcal{E}$ and $\text{visit}[u] = \text{true}\}$.
   c. For each $u \in \mathcal{S}$:
      i. $\text{enqueue}(u)$.
      ii. $\text{visit}[u] = \text{true}$.
      iii. $\text{dist}[u] = \text{dist}[v] + 1$.
      iv. $\text{path}[u] = v$.

Pseudo Code (Cont.)

5. While the queue is not empty:
   a. $v \leftarrow \text{dequeue}()$.
   b. $\mathcal{S} \leftarrow \{u \mid e_{vu} \in \mathcal{E}$ and $\text{visit}[u] = \text{true}\}$.
   c. For each $u \in \mathcal{S}$:
      i. $\text{enqueue}(u)$.
      ii. $\text{visit}[u] = \text{true}$.
      iii. $\text{dist}[u] = \text{dist}[v] + 1$.
      iv. $\text{path}[u] = v$.

Outputs: $\text{dist}[v]$ and $\text{path}[v]$, for all $v \in \mathcal{V}$.

Time Complexity

Time Complexity

The time complexity is $O(|\mathcal{V}| + |\mathcal{E}|)$. (Why?)

First, the initialization has $O(|\mathcal{V}|)$ time complexity.

Time Complexity

The time complexity is $O(|\mathcal{V}| + |\mathcal{E}|)$. (Why?)

First, the initialization has $O(|\mathcal{V}|)$ time complexity.

Second, queue operations has a total of $O(|\mathcal{V}|)$ time complexity.

- Every vertex is enqueued and dequeued exactly once.
- Enqueue and dequeue operations have constant time complexity.

Time Complexity

The time complexity is $O(|\mathcal{V}| + |\mathcal{E}|)$. (Why?)

First, the initialization has $O(|\mathcal{V}|)$ time complexity.

Second, queue operations has a total of $O(|\mathcal{V}|)$ time complexity.

Third, every edge is touched once; thus $O(|\mathcal{E}|)$ time complexity.

- Once vertex v is dequeued, all the edges from v are touched.
- Every vertex is dequeued only once.
- Thus every edge is touched only once.

Thank You!