Graph Distances

Floyd-Warshall

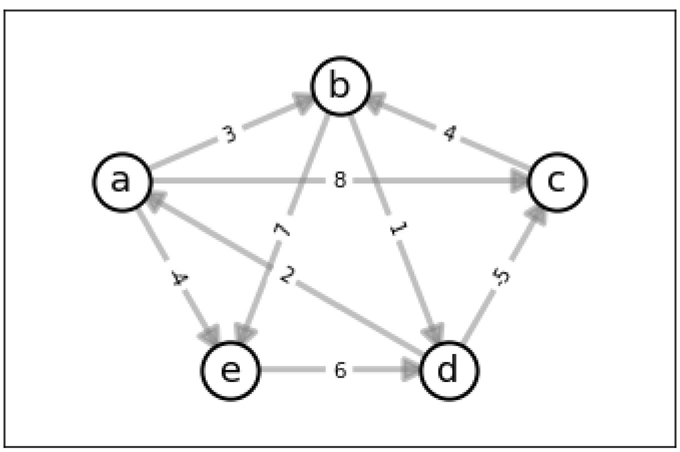
Given a weighted graph, determine all sources shortest path. This means you determine the shortest distance from any vertex to any other vertex on the graph.

Let n be the number of vertices and m be the number of edges of the graph.

1. Create an adjascency matrix D[n x n] where the initial values in the table are the weights of the edges, and the missing values are all infinity.
2. Iterate k over all n vertices of the graph.
3. Iterate i over all n vertices of the graph.
4. Iterate j over all n vertices of the graph.
5. For each (i, j)
   ```
   D[i, j] = min(D[i, j], D[i, k] + D[k, j])
   ```
6. If at the end one of the diagonal values is negative, then the graph has a negative cycle.

Note: to also save the vertices of the path instead of just the distances, create a second P[n x n] matrix. If the value of D[i, j] is updated, also update `P[i, j] = P[k, j]`

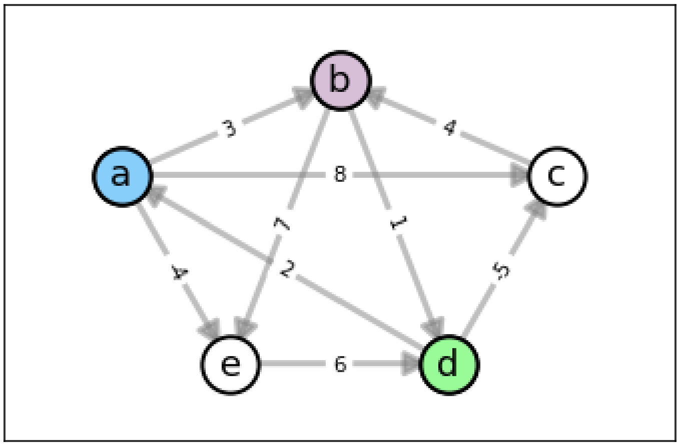
The progress of the algorithm is shown below. Note: due to the number of steps, outputs are only shown when a value is changed in the matrix.




```
Initial Table
```

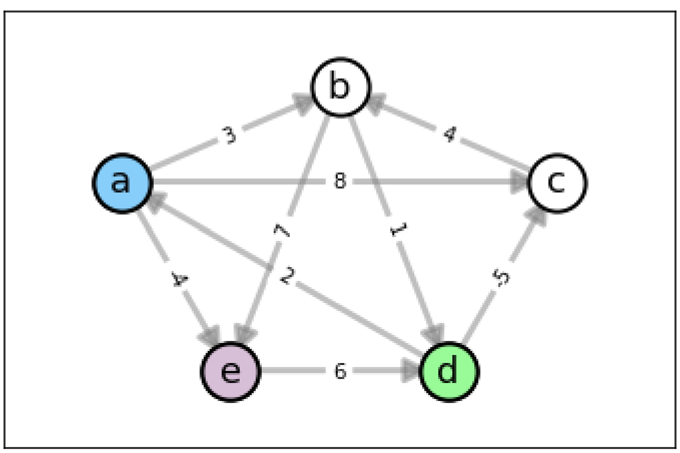
	a	b	c	d	e
a	0	3	8	inf	-4
b	inf	0	inf	1	7
c	inf	4	0	inf	inf
d	2	inf	-5	0	inf
e	inf	inf	inf	6	0

```
Begin evaluating 'a' on path
i = d, k = a, j = b
```



	a	b	c	d	e
a	0	3	8	inf	-4
b	inf	0	inf	1	7
c	inf	4	0	inf	inf
d	2	5	-5	0	inf
e	inf	inf	inf	6	0

```
i = d, k = a, j = e
```

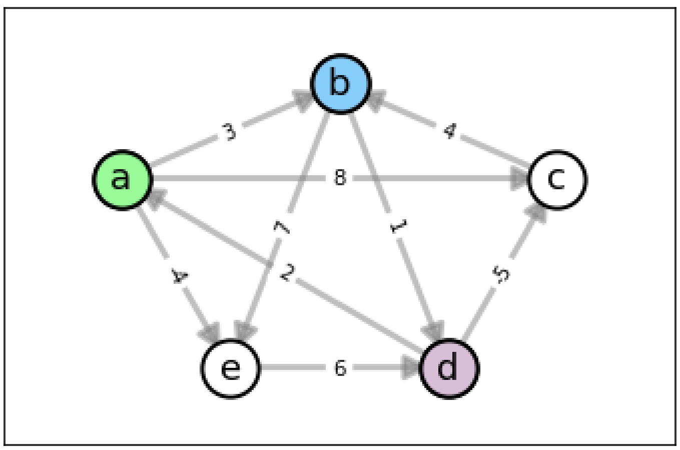


	a	b	c	d	e
a	0	3	8	inf	-4
b	inf	0	inf	1	7
c	inf	4	0	inf	inf
d	2	5	-5	0	-2
e	inf	inf	inf	6	0

```
------------------------------------------
Table after evaluating 'a' on path
```

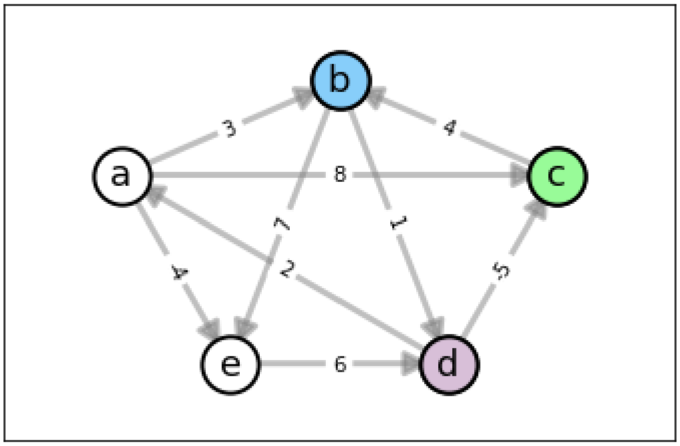
	a	b	c	d	e
a	0	3	8	inf	-4
b	inf	0	inf	1	7
c	inf	4	0	inf	inf
d	2	5	-5	0	-2
e	inf	inf	inf	6	0

```
------------------------------------------
Begin evaluating 'b' on path
i = a, k = b, j = d
```



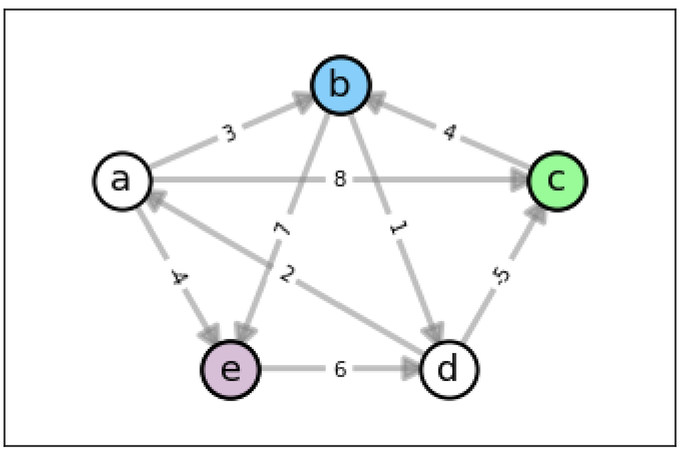
	a	b	c	d	e
a	0	3	8	4	-4
b	inf	0	inf	1	7
c	inf	4	0	inf	inf
d	2	5	-5	0	-2
e	inf	inf	inf	6	0

```
i = c, k = b, j = d
```



	a	b	c	d	e
a	0	3	8	4	-4
b	inf	0	inf	1	7
c	inf	4	0	5	inf
d	2	5	-5	0	-2
e	inf	inf	inf	6	0

```
i = c, k = b, j = e
```

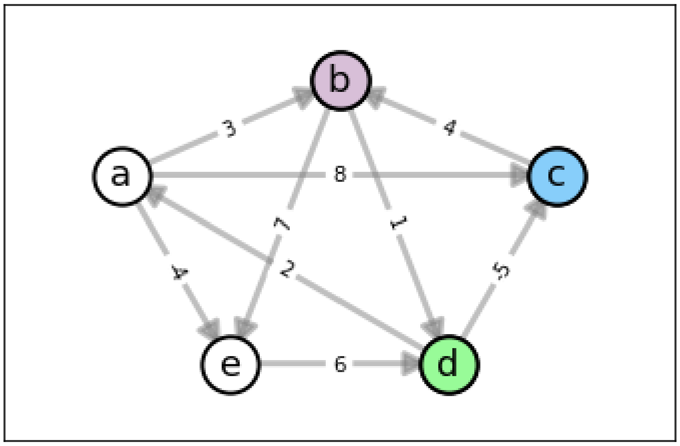


	a	b	c	d	e
a	0	3	8	4	-4
b	inf	0	inf	1	7
c	inf	4	0	5	11
d	2	5	-5	0	-2
e	inf	inf	inf	6	0

```
-------------------------------------------
Table after evaluating 'b' on path
```

	a	b	c	d	e
a	0	3	8	4	-4
b	inf	0	inf	1	7
c	inf	4	0	5	11
d	2	5	-5	0	-2
e	inf	inf	inf	6	0

```
-------------------------------------------
Begin evaluating 'c' on path
i = d, k = c, j = b
```

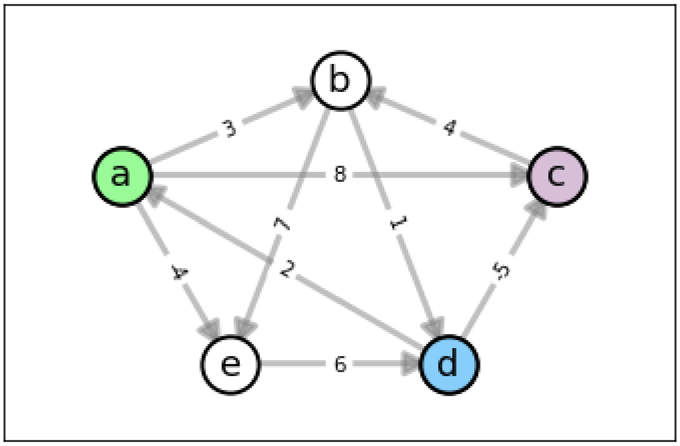


	a	b	c	d	e
a	0	3	8	4	-4
b	inf	0	inf	1	7
c	inf	4	0	5	11
d	2	-1	-5	0	-2
e	inf	inf	inf	6	0

```
-------------------------------------------
Table after evaluating 'c' on path
```

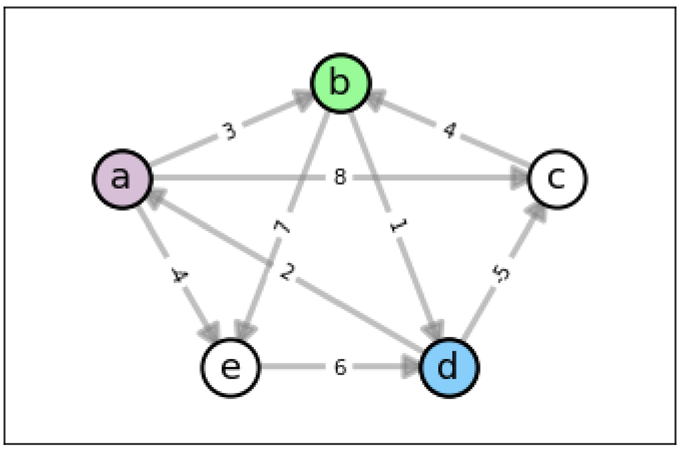
	a	b	c	d	e
a	0	3	8	4	-4
b	inf	0	inf	1	7
c	inf	4	0	5	11
d	2	-1	-5	0	-2
e	inf	inf	inf	6	0

```
-------------------------------------------
Begin evaluating 'd' on path
i = a, k = d, j = c
```



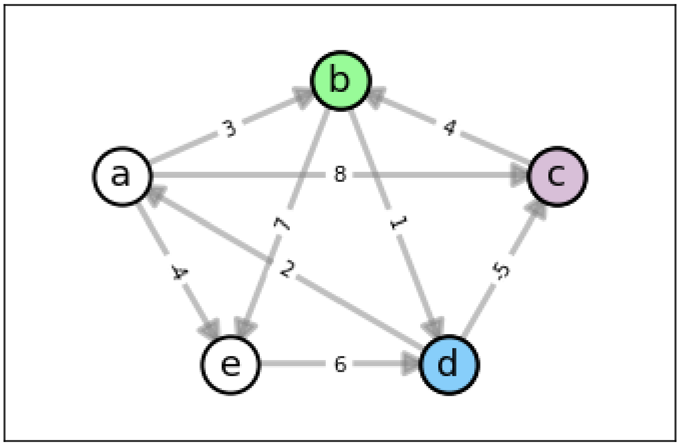
	a	b	c	d	e
a	0	3	-1	4	-4
b	inf	0	inf	1	7
c	inf	4	0	5	11
d	2	-1	-5	0	-2
e	inf	inf	inf	6	0

`i = b, k = d, j = a`



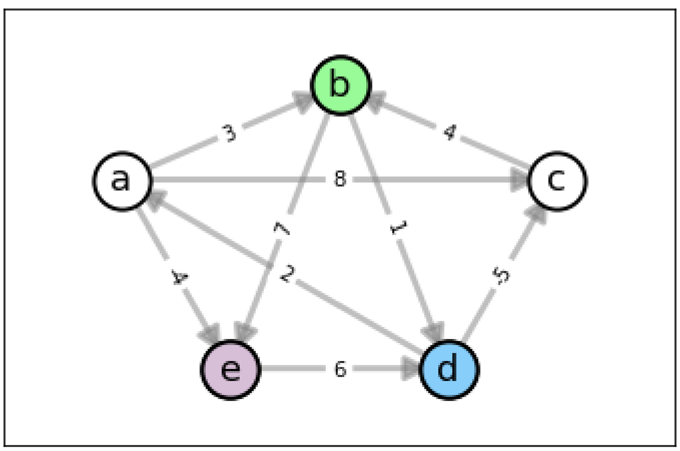
	a	b	c	d	e
a	0	3	-1	4	-4
b	3	0	inf	1	7
c	inf	4	0	5	11
d	2	-1	-5	0	-2
e	inf	inf	inf	6	0

`i = b, k = d, j = c`



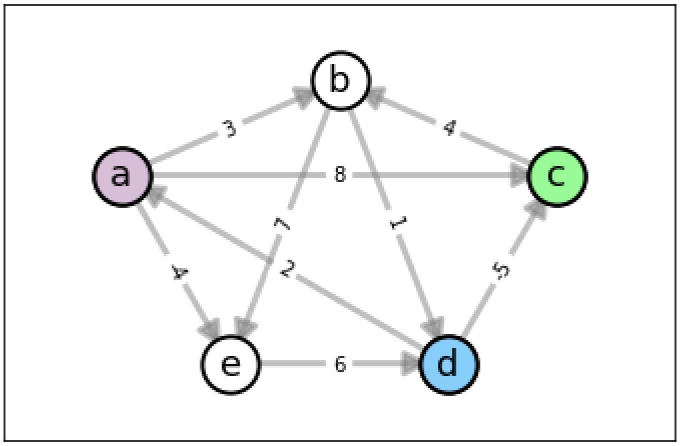
	a	b	c	d	e
a	0	3	-1	4	-4
b	3	0	-4	1	7
c	inf	4	0	5	11
d	2	-1	-5	0	-2
e	inf	inf	inf	6	0

`i = b, k = d, j = e`



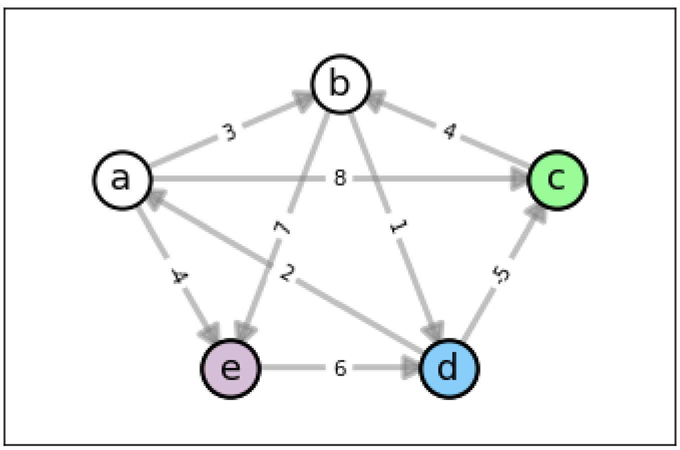
	a	b	c	d	e
a	0	3	-1	4	-4
b	3	0	-4	1	-1
c	inf	4	0	5	11
d	2	-1	-5	0	-2
e	inf	inf	inf	6	0

`i = c, k = d, j = a`



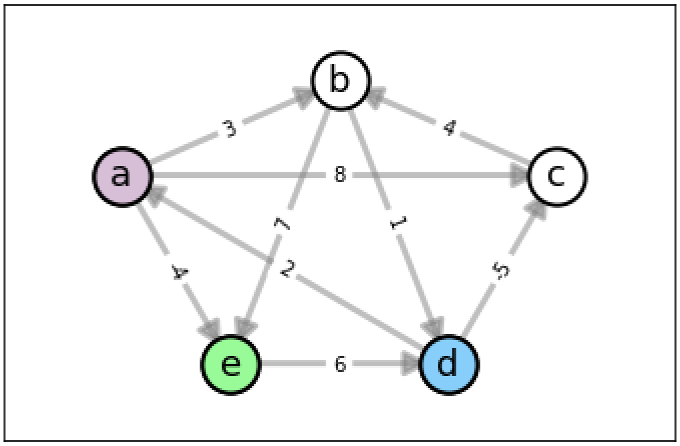
	a	b	c	d	e
a	0	3	-1	4	-4
b	3	0	-4	1	-1
c	7	4	0	5	11
d	2	-1	-5	0	-2
e	inf	inf	inf	6	0

`i = c, k = d, j = e`



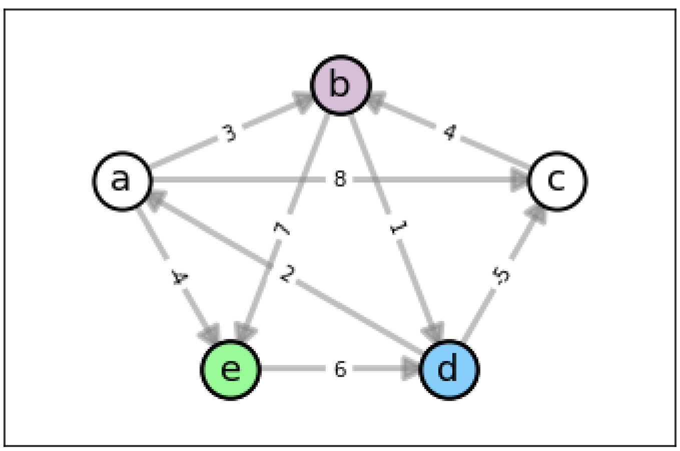
	a	b	c	d	e
a	0	3	-1	4	-4
b	3	0	-4	1	-1
c	7	4	0	5	3
d	2	-1	-5	0	-2
e	inf	inf	inf	6	0

`i = e, k = d, j = a`



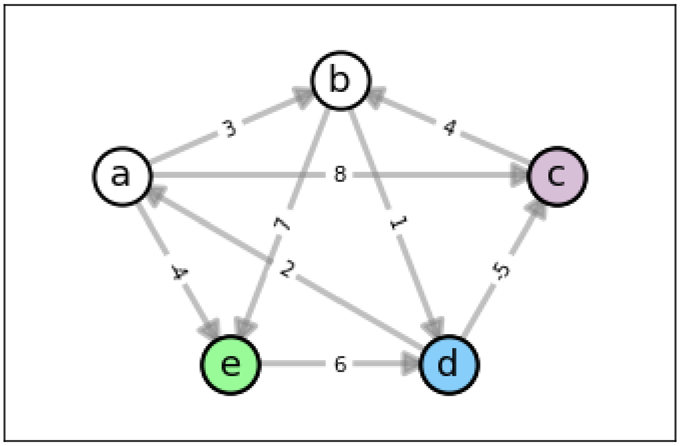
	a	b	c	d	e
a	0	3	-1	4	-4
b	3	0	-4	1	-1
c	7	4	0	5	3
d	2	-1	-5	0	-2
e	8	inf	inf	6	0

`i = e, k = d, j = b`



	a	b	c	d	e
a	0	3	-1	4	-4
b	3	0	-4	1	-1
c	7	4	0	5	3
d	2	-1	-5	0	-2
e	8	5	inf	6	0

`i = e, k = d, j = c`

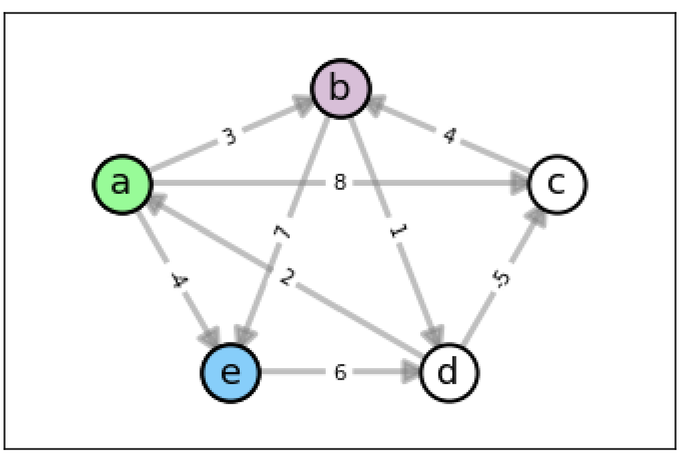


	a	b	c	d	e
a	0	3	-1	4	-4
b	3	0	-4	1	-1
c	7	4	0	5	3
d	2	-1	-5	0	-2
e	8	5	1	6	0

```
------------------------------------------
Table after evaluating 'd' on path
```

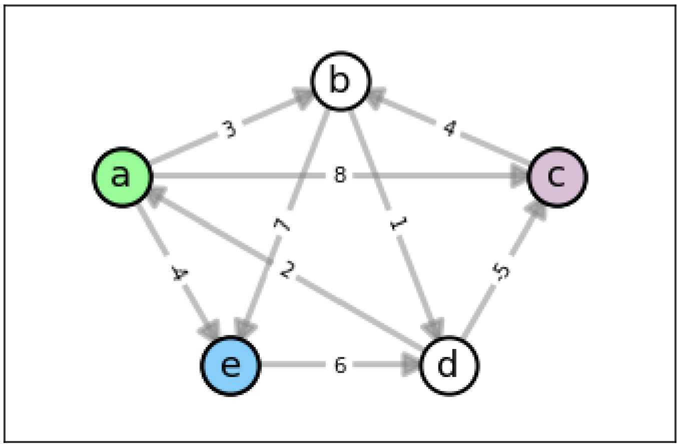
	a	b	c	d	e
a	0	3	-1	4	-4
b	3	0	-4	1	-1
c	7	4	0	5	3
d	2	-1	-5	0	-2
e	8	5	1	6	0

```
------------------------------------------
Begin evaluating 'e' on path
i = a, k = e, j = b
```



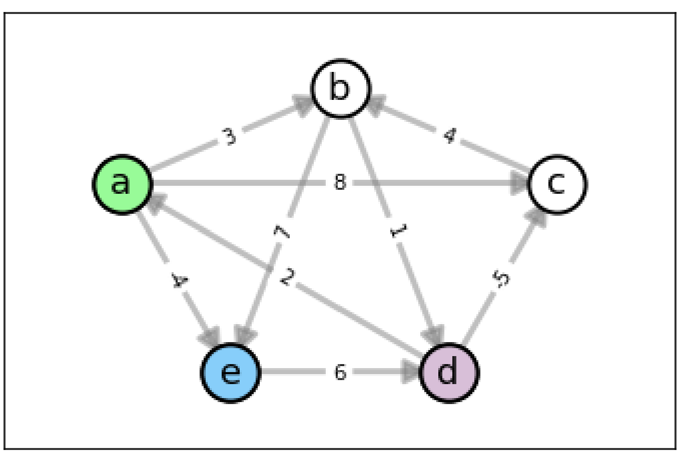
	a	b	c	d	e
a	0	1	-1	4	-4
b	3	0	-4	1	-1
c	7	4	0	5	3
d	2	-1	-5	0	-2
e	8	5	1	6	0

`i = a, k = e, j = c`



	a	b	c	d	e
a	0	1	-3	4	-4
b	3	0	-4	1	-1
c	7	4	0	5	3
d	2	-1	-5	0	-2
e	8	5	1	6	0

`i = a, k = e, j = d`



	a	b	c	d	e
a	0	1	-3	2	-4
b	3	0	-4	1	-1
c	7	4	0	5	3
d	2	-1	-5	0	-2
e	8	5	1	6	0

```
------------------------------------------
Table after evaluating 'e' on path
```

	a	b	c	d	e
a	0	1	-3	2	-4
b	3	0	-4	1	-1
c	7	4	0	5	3
d	2	-1	-5	0	-2
e	8	5	1	6	0

```
------------------------------------------
Final Table
```

	a	b	c	d	e
a	0	1	-3	2	-4
b	3	0	-4	1	-1
c	7	4	0	5	3
d	2	-1	-5	0	-2
e	8	5	1	6	0

```
Backtrace Matrix:
```

	a	b	c	d	e
a	-	c	d	e	a
b	d	-	d	b	a
c	d	c	-	b	a
d	d	c	d	-	a
e	d	c	d	e	-

To get the vertices of the shortest path between any two vertices i and j from the backtrace matrix P:

1. Let i and j be the first and last vertices of the path respectively: v_1 = i, v_n = j
2. Find the value of P[v_1, v_n] in the path matrix: v_{n-1} = P[v_1, v_n]
3. Find the value of P[v_1, v_{n-1}] in the path matrix: v_{n-2} = P[v_1, v_{n-1}]
4. Continue building the path until P[v_1, v_{n-k}] = v_1

For the matrix above, the path from a to c is as follows:

P[a, c] = d

P[a, d] = e

P[a, e] = a

Final path = a, e, d, c

Complexity

Time complexity: $O(n^3)$ where n is the number of vertices

Space complexity: $O(n^2)$ where n is the number of vertices