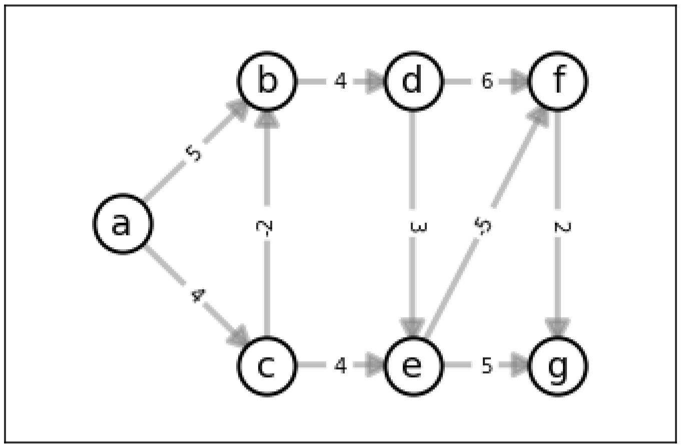
Graph Distances

Bellman Ford

Given a weighted graph, determine the single source shortest path. This means you determine the shortest distance from the starting vertex to any other vertices on the graph.

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

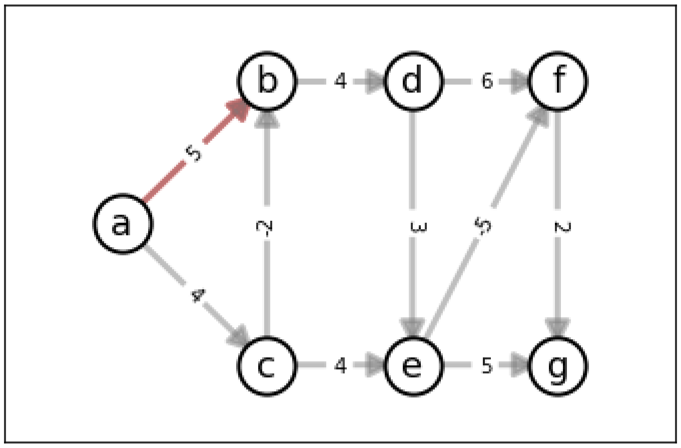
1. Create an array D[n]. Initialize the source value to zero and all remaining values to infinity. Each element of the array corresponds to the distance of that element from the source vertex. The distance from the source to itself is 0, and all others are unknown.
2. Iterate over this array $n-1$ times, each time performing the following actions.
3. Iterate over each edge of the graph. For each (source, destination) of the edge
   `D[destination] = min(D[source] + weight(source, destination), D[destination])`
4. If during one of the iterations, no changes are made to D, then you may terminate early.
5. If you run the algorithm 1 final time after $n-1$ iterations and a distance changes, then the graph has a negative cycle.



```
Edge order: [('a', 'b'), ('a', 'c'), ('b', 'd'), ('c', 'b'), ('c', 'e'), ('d', 'e'),
('d', 'f'), ('e', 'f'), ('e', 'g'), ('f', 'g')]
```

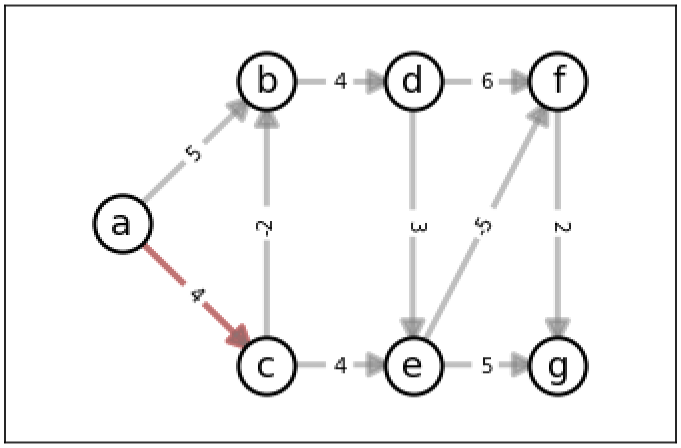
	a	b	c	d	e	f	g
0	0	inf	inf	inf	inf	inf	inf

```
--------------
ITERATION: 1
--------------
--------------
RELAX: a, b
```



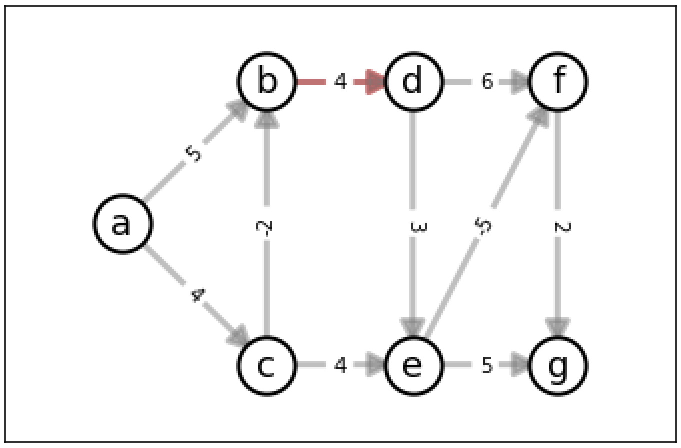
	a	b	c	d	e	f	g
1	0	5	inf	inf	inf	inf	inf

```
--------------
RELAX: a, c
```



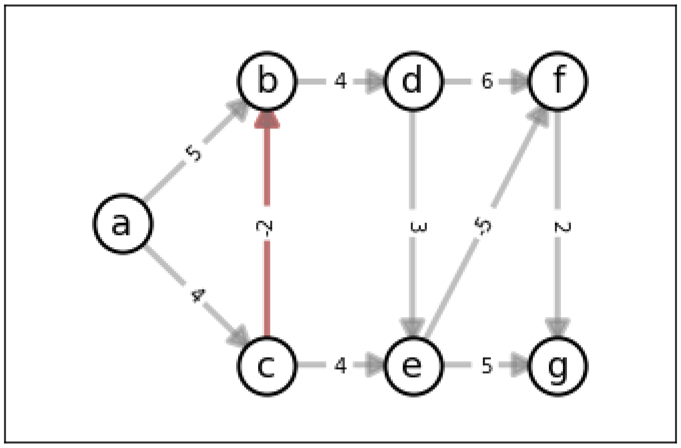
	a	b	c	d	e	f	g
1	0	5	4	inf	inf	inf	inf

```
--------------
RELAX: b, d
```



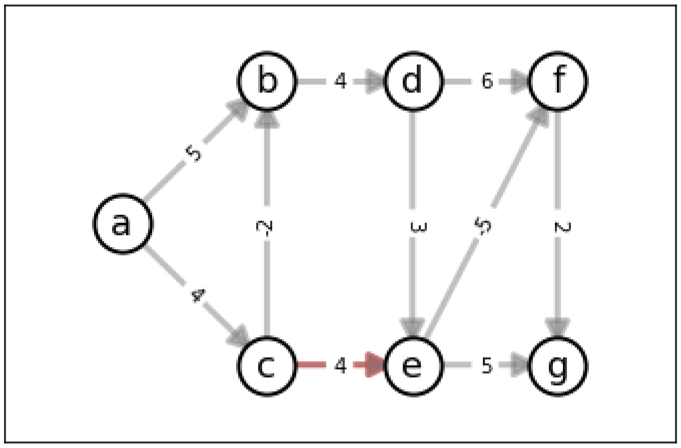
	a	b	c	d	e	f	g
1	0	5	4	9	inf	inf	inf

```
--------------
RELAX: c, b
```



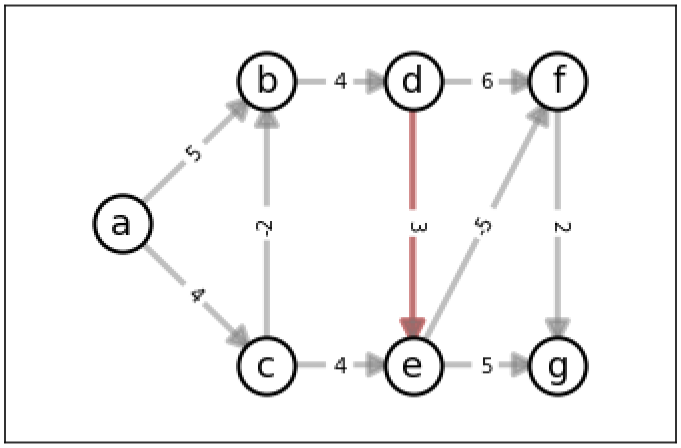
	a	b	c	d	e	f	g
1	0	2	4	9	inf	inf	inf

```
--------------
RELAX: c, e
```



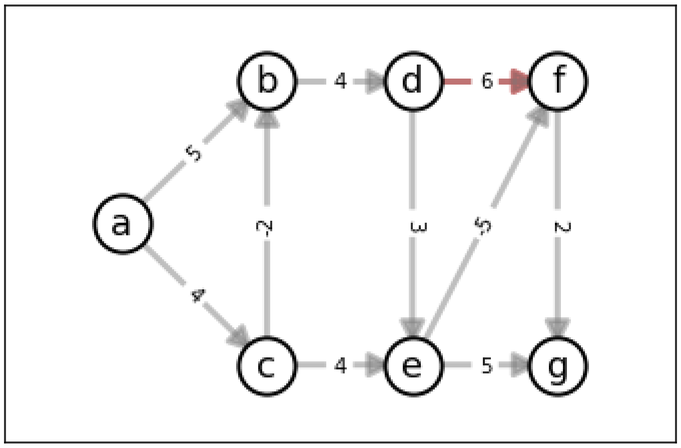
	a	b	c	d	e	f	g
1	0	2	4	9	8	inf	inf

```
--------------
RELAX: d, e
```



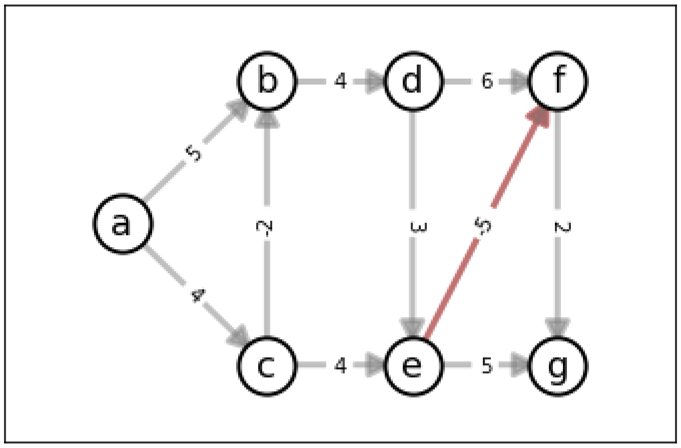
	a	b	c	d	e	f	g
1	0	2	4	9	8	inf	inf

```
--------------
RELAX: d, f
```



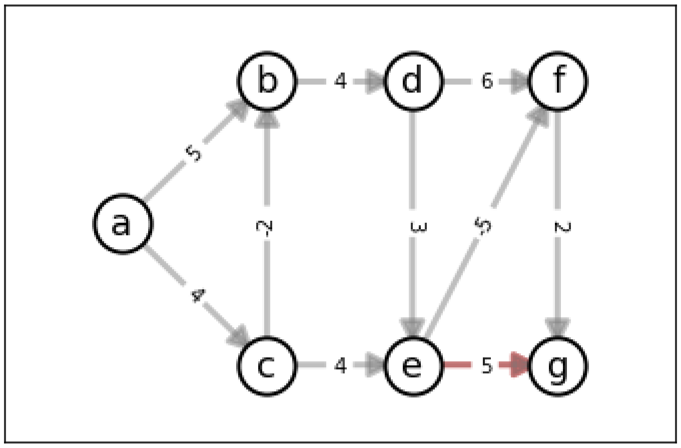
	a	b	c	d	e	f	g
1	0	2	4	9	8	15	inf

```
--------------
RELAX: e, f
```



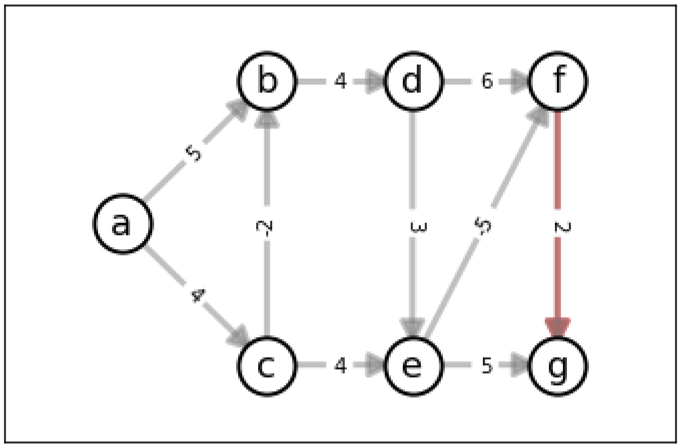
	a	b	c	d	e	f	g
1	0	2	4	9	8	3	inf

```
--------------
RELAX: e, g
```



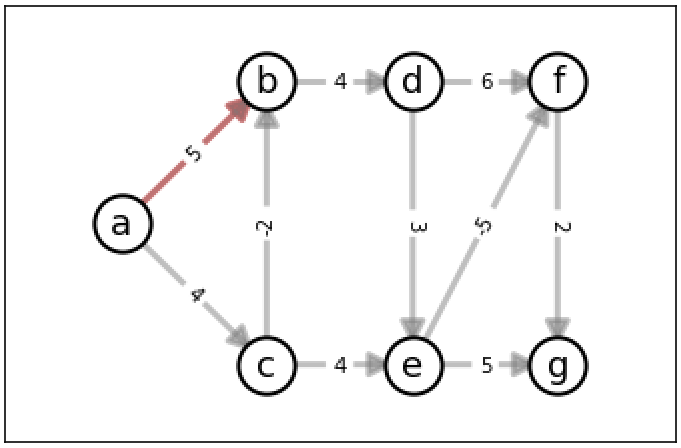
	a	b	c	d	e	f	g
1	0	2	4	9	8	3	13

```
--------------
RELAX: f, g
```



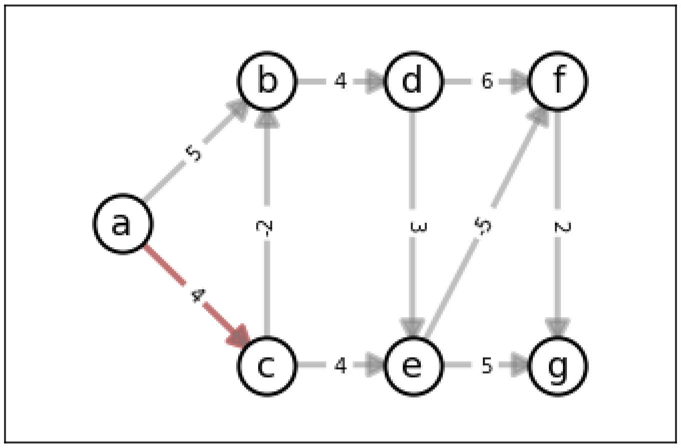
	a	b	c	d	e	f	g
1	0	2	4	9	8	3	5

```
--------------
ITERATION: 2
--------------
--------------
RELAX: a, b
```



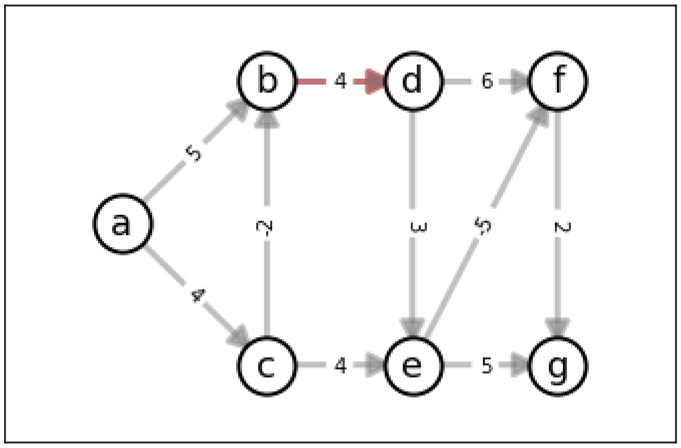
	a	b	c	d	e	f	g
2	0	2	4	9	8	3	5

```
--------------
RELAX: a, c
```



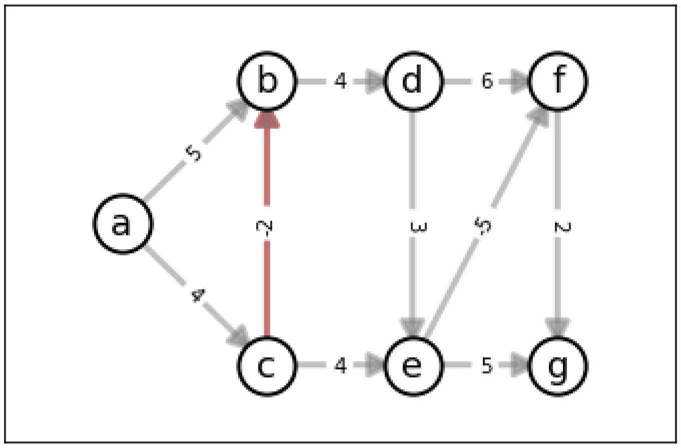
	a	b	c	d	e	f	g
2	0	2	4	9	8	3	5

```
--------------
RELAX: b, d
```



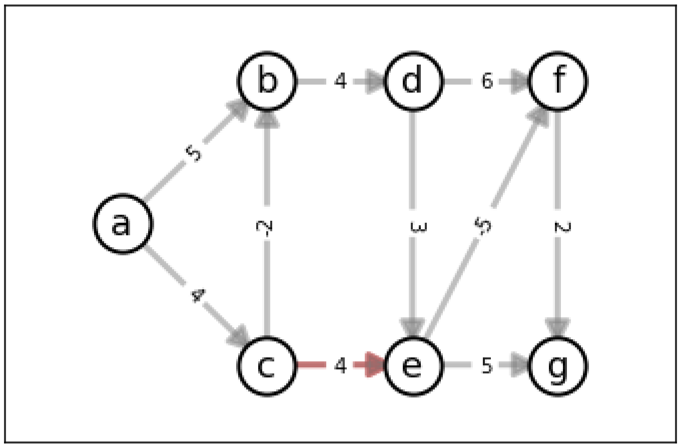
	a	b	c	d	e	f	g
2	0	2	4	6	8	3	5

```
--------------
RELAX: c, b
```



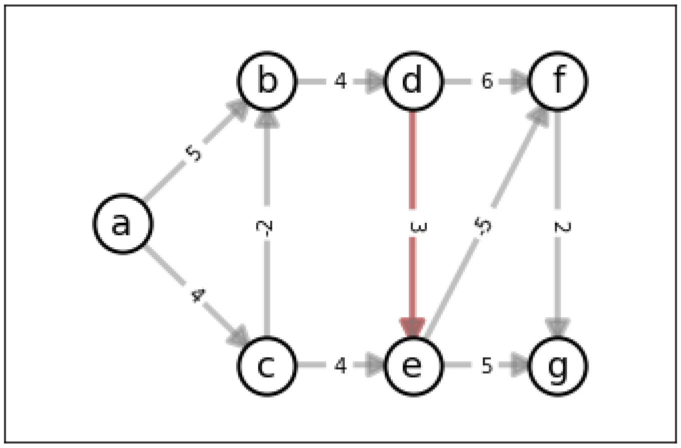
	a	b	c	d	e	f	g
2	0	2	4	6	8	3	5

```
--------------
RELAX: c, e
```



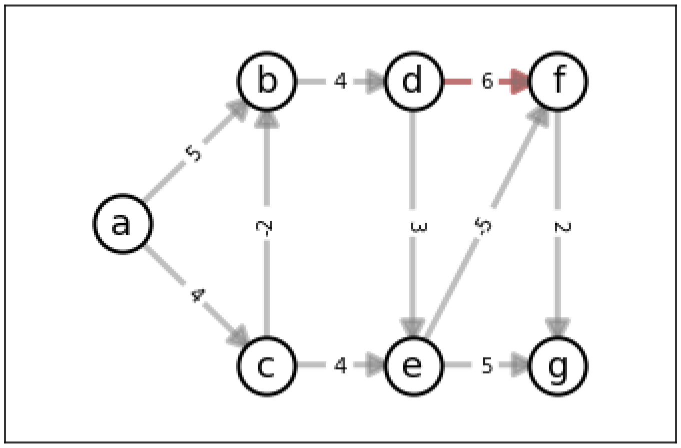
	a	b	c	d	e	f	g
2	0	2	4	6	8	3	5

```
--------------
RELAX: d, e
```



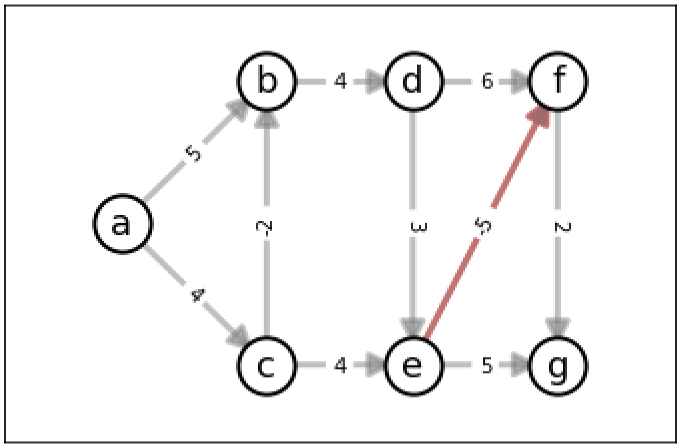
	a	b	c	d	e	f	g
2	0	2	4	6	8	3	5

```
--------------
RELAX: d, f
```



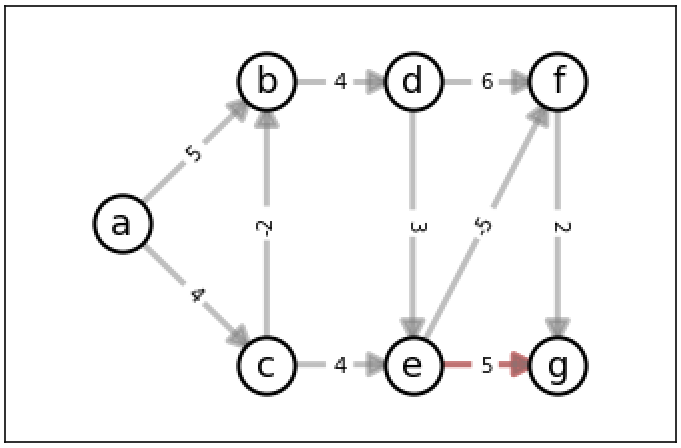
	a	b	c	d	e	f	g
2	0	2	4	6	8	3	5

```
--------------
RELAX: e, f
```



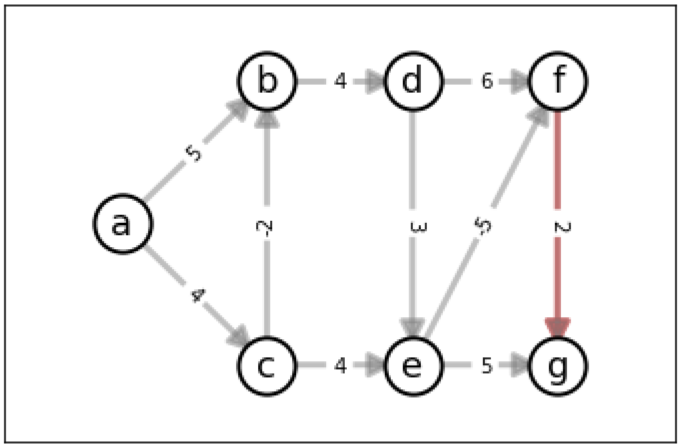
	a	b	c	d	e	f	g
2	0	2	4	6	8	3	5

```
--------------
RELAX: e, g
```



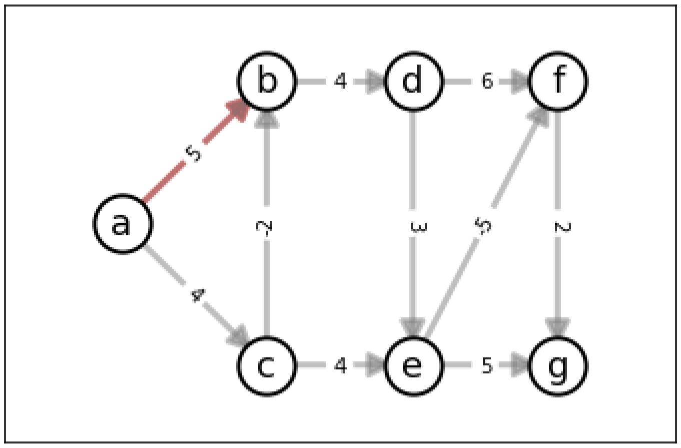
	a	b	c	d	e	f	g
2	0	2	4	6	8	3	5

```
--------------
RELAX: f, g
```



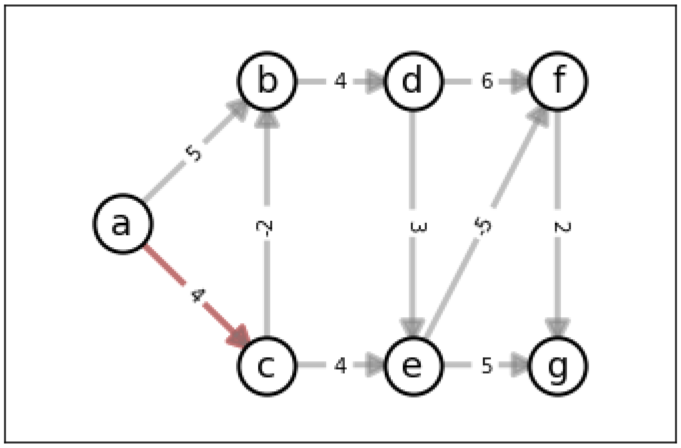
	a	b	c	d	e	f	g
2	0	2	4	6	8	3	5

```
--------------
ITERATION: 3
--------------
--------------
RELAX: a, b
```



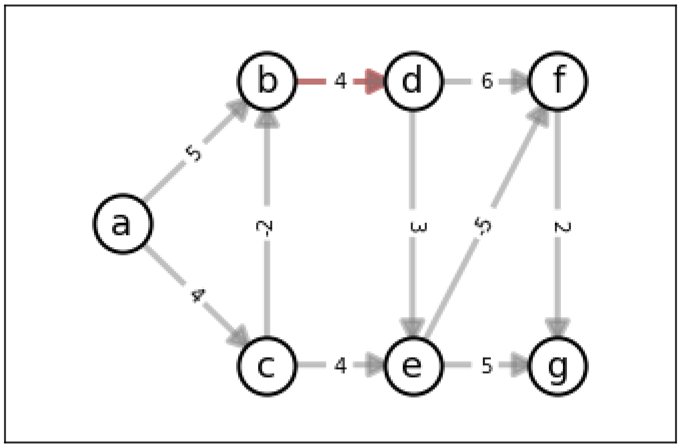
	a	b	c	d	e	f	g
3	0	2	4	6	8	3	5

```
--------------
RELAX: a, c
```



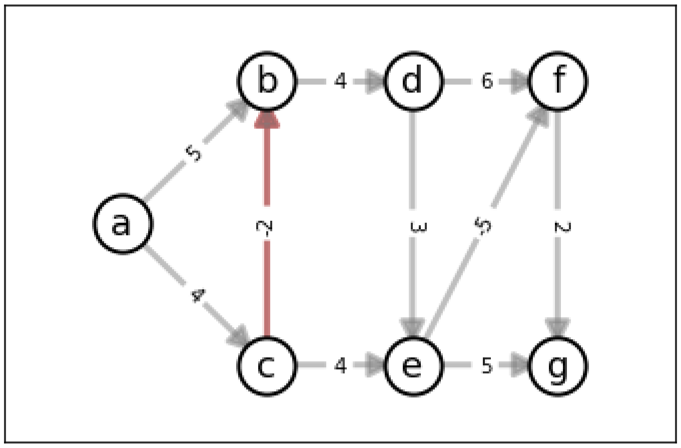
	a	b	c	d	e	f	g
3	0	2	4	6	8	3	5

```
--------------
RELAX: b, d
```



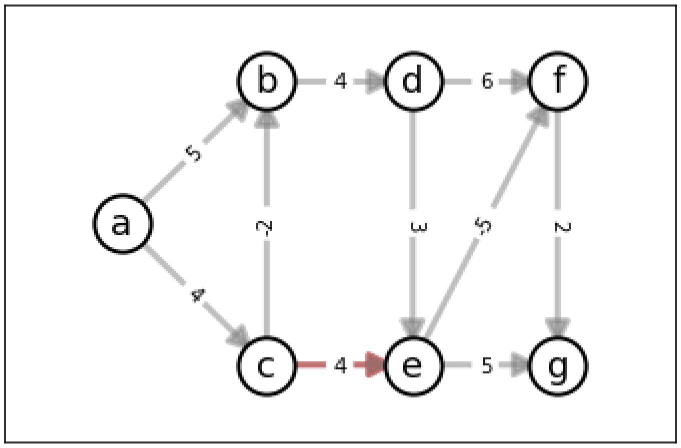
	a	b	c	d	e	f	g
3	0	2	4	6	8	3	5

```
--------------
RELAX: c, b
```



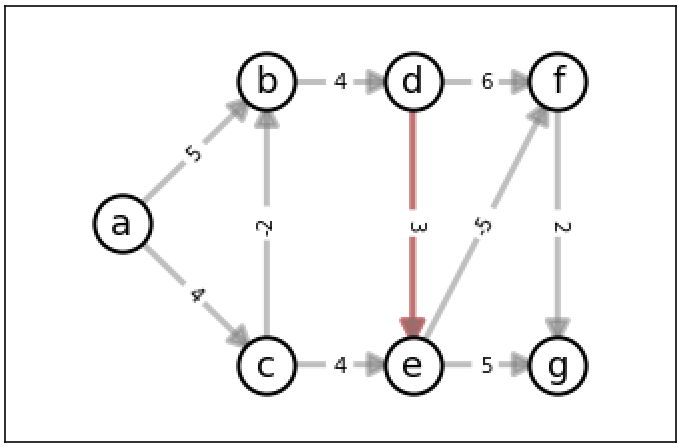
	a	b	c	d	e	f	g
3	0	2	4	6	8	3	5

```
--------------
RELAX: c, e
```



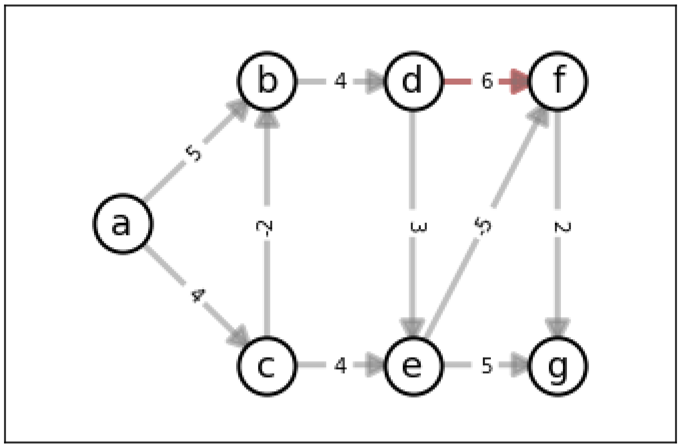
	a	b	c	d	e	f	g
3	0	2	4	6	8	3	5

```
--------------
RELAX: d, e
```



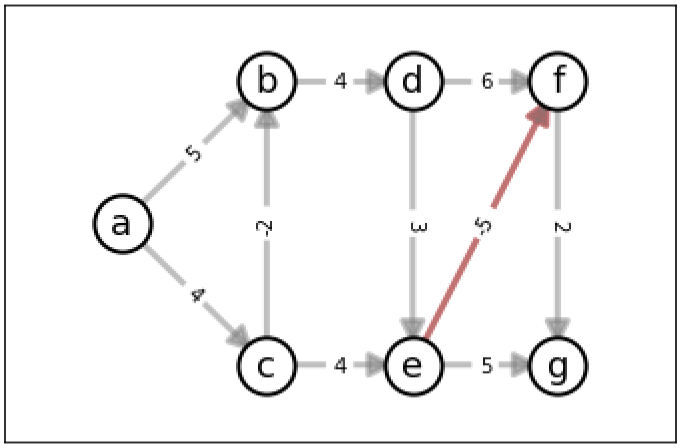
	a	b	c	d	e	f	g
3	0	2	4	6	8	3	5

```
--------------
RELAX: d, f
```



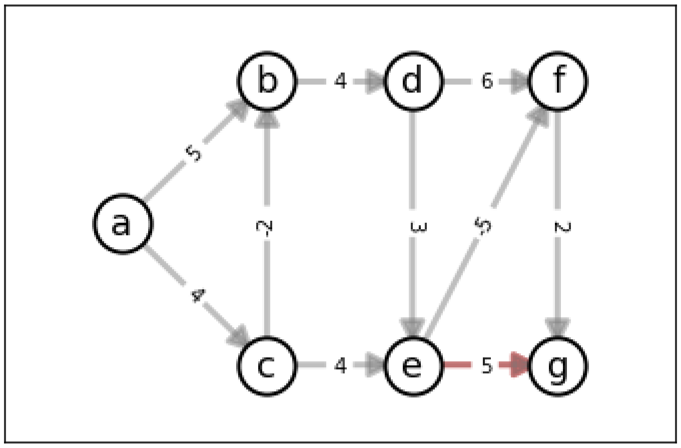
	a	b	c	d	e	f	g
3	0	2	4	6	8	3	5

```
--------------
RELAX: e, f
```



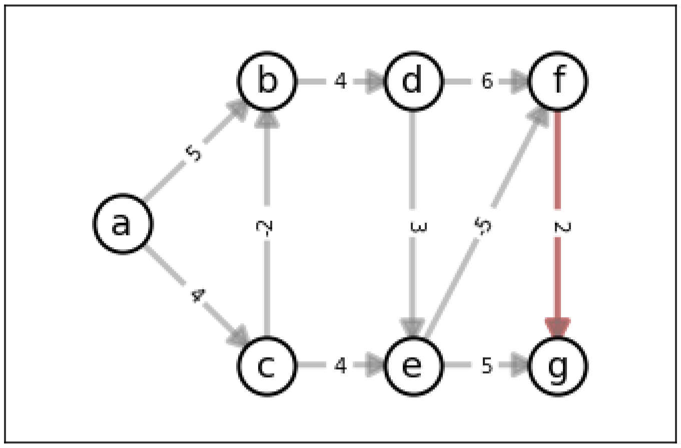
	a	b	c	d	e	f	g
3	0	2	4	6	8	3	5

```
--------------
RELAX: e, g
```



	a	b	c	d	e	f	g
3	0	2	4	6	8	3	5

```
--------------
RELAX: f, g
```



	a	b	c	d	e	f	g
3	0	2	4	6	8	3	5

```
FINAL SOLUTION:
```

	a	b	c	d	e	f	g
3	0	2	4	6	8	3	5

Complexity

Time complexity: $O(nm)$ where n is the number of vertices and m is the number of edges