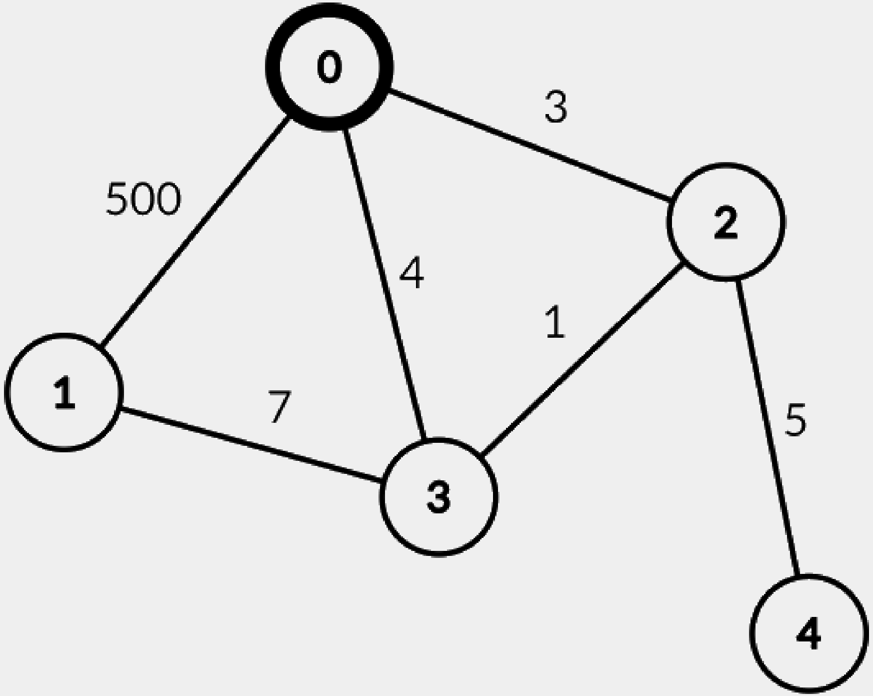
Prims to Dijkstras

Into & Live Coding

Prims Minimum Spanning Tree

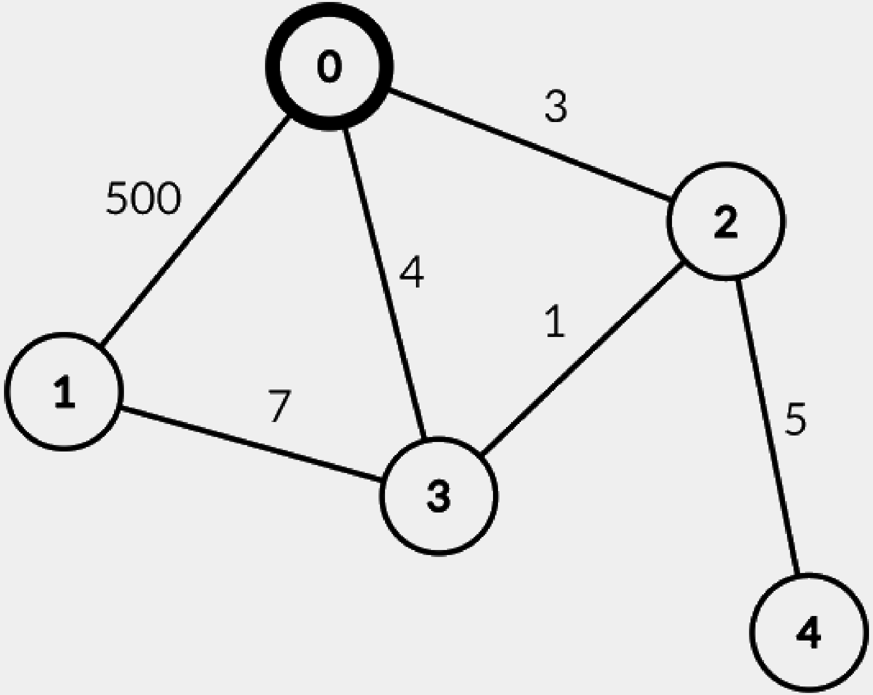
- Uses an Indexable Priority Queue.
- Uses “Eager” prims, so only considers best edge in PQ, which is why the PQ is indexable.

Prims Example



ID	edgeTo	distTo	marked	pq (index, value)
0	null	0	FALSE	
1	null	∞	FALSE	
2	null	∞	FALSE	
3	null	∞	FALSE	
4	null	∞	FALSE	

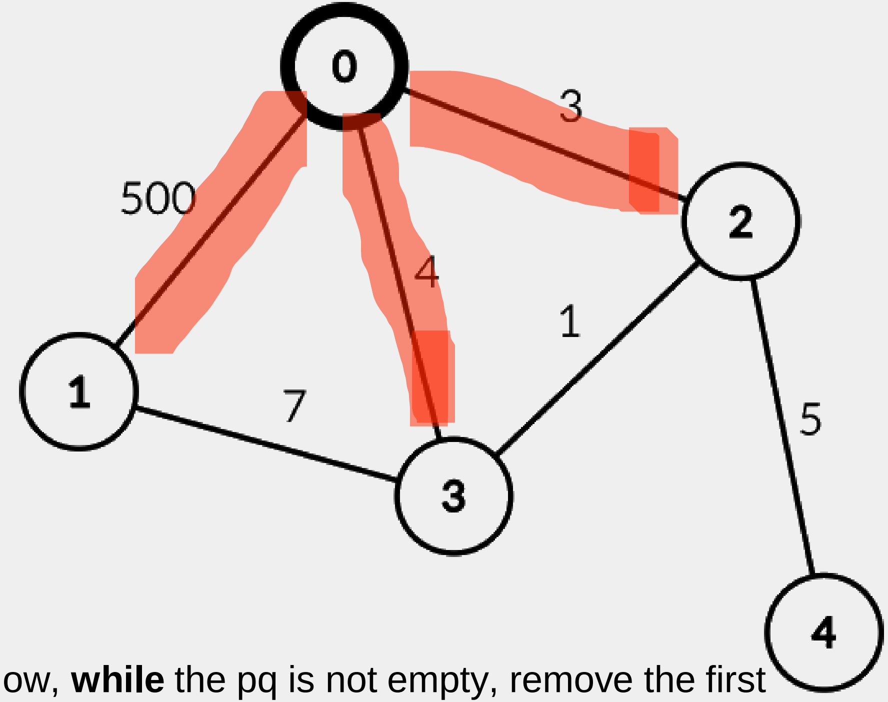
Prims Example



Add the start node to the pq with a distance of 0

ID	edgeTo	distTo	marked	pq (index, value)
0	null	0	FALSE	(0, 0)
1	null	∞	FALSE	
2	null	∞	FALSE	
3	null	∞	FALSE	
4	null	∞	FALSE	

Prims Example

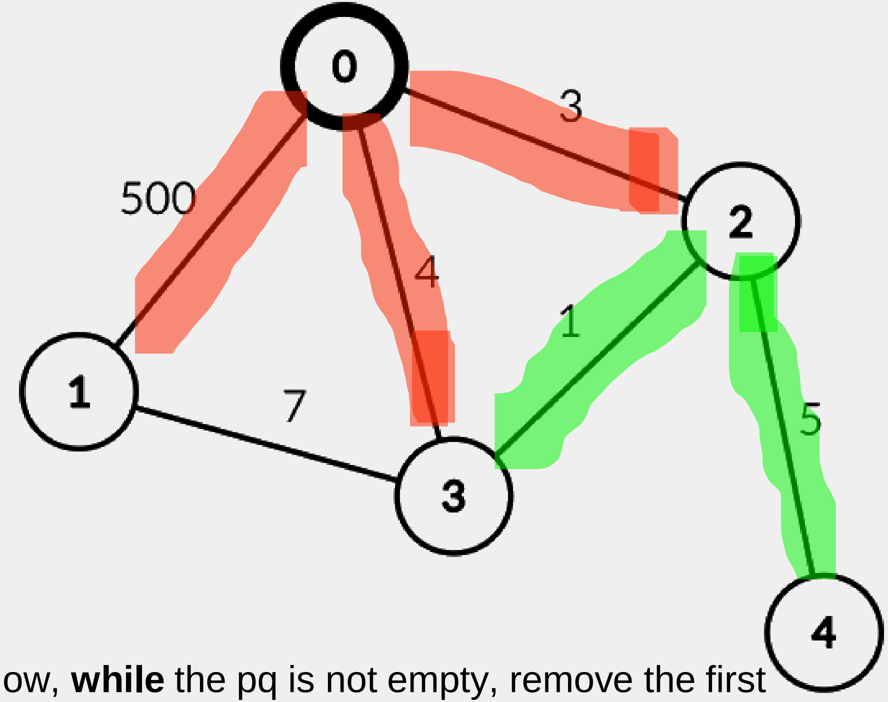


Now, **while** the pq is not empty, remove the first element of the pq and **consider** all of its edges

Removed ID 0 of weight 0

ID	edgeTo	distTo	marked	pq (index, value)
0	null	0	TRUE	(2, 3)
1	(0, 1)	500	FALSE	(3, 4)
2	(0, 2)	3	FALSE	(1, 500)
3	(0, 3)	4	FALSE	
4	null	∞	FALSE	

Prims Example

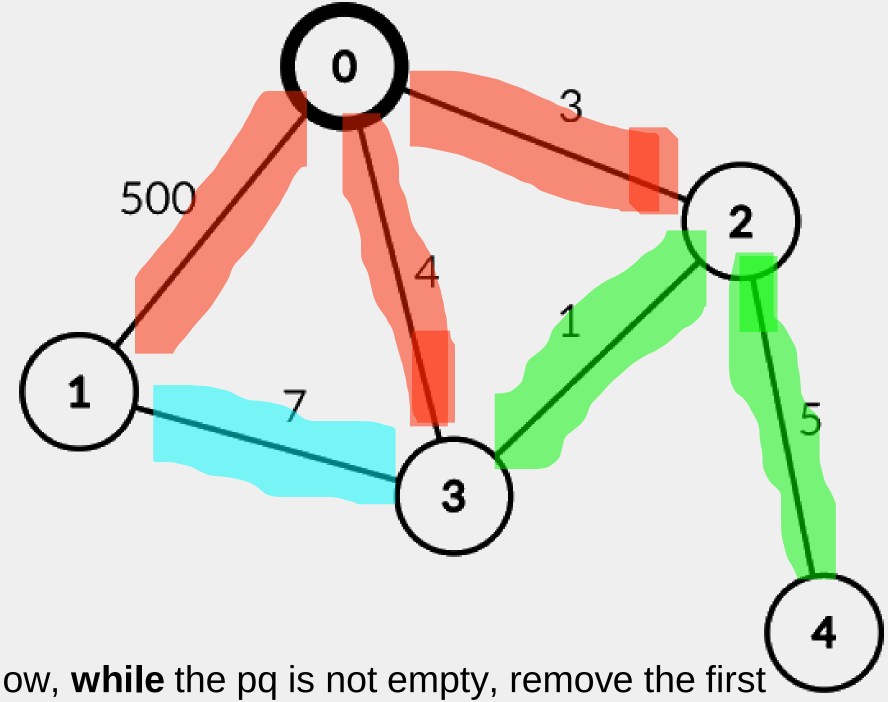


Now, **while** the pq is not empty, remove the first element of the pq and **consider** all of its edges

Removed ID 2 of weight 3

ID	edgeTo	distTo	marked	pq (index, value)
0	null	0	TRUE	(3, 1)
1	(0, 1)	500	FALSE	(4, 5)
2	(0, 2)	3	TRUE	(1, 500)
3	(2, 3)	1	FALSE	
4	(2, 4)	5	FALSE	

Prims Example

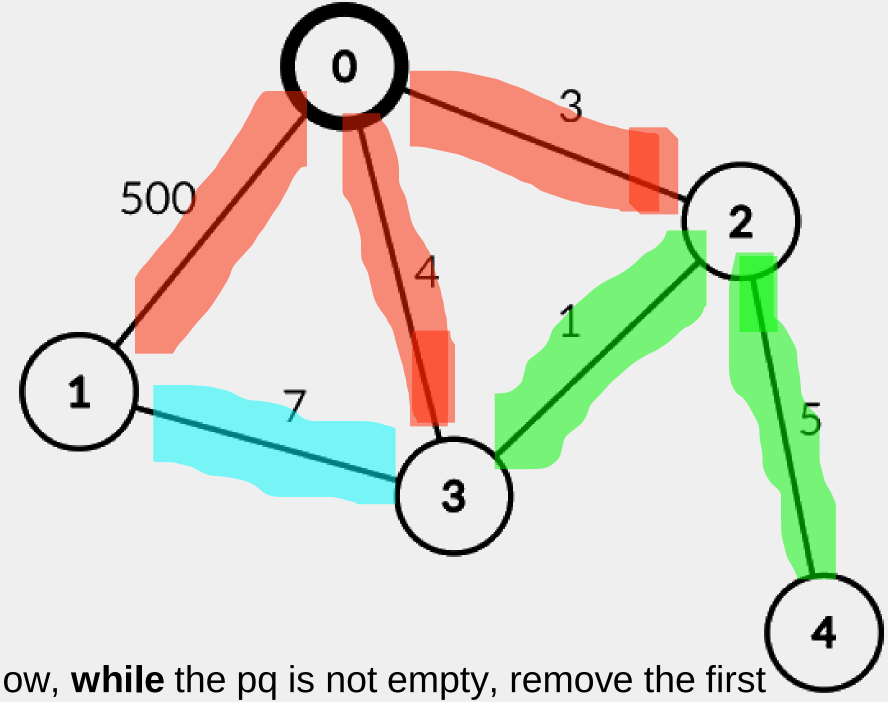


ID	edgeTo	distTo	marked	pq (index, value)
0	null	0	TRUE	(4, 5)
1	(3, 1)	7	FALSE	(1, 7)
2	(0, 2)	3	TRUE	
3	(2, 3)	1	TRUE	
4	(2, 4)	5	FALSE	

Now, **while** the pq is not empty, remove the first element of the pq and **consider** all of its edges

Removed ID 3 of weight 1

Prims Example

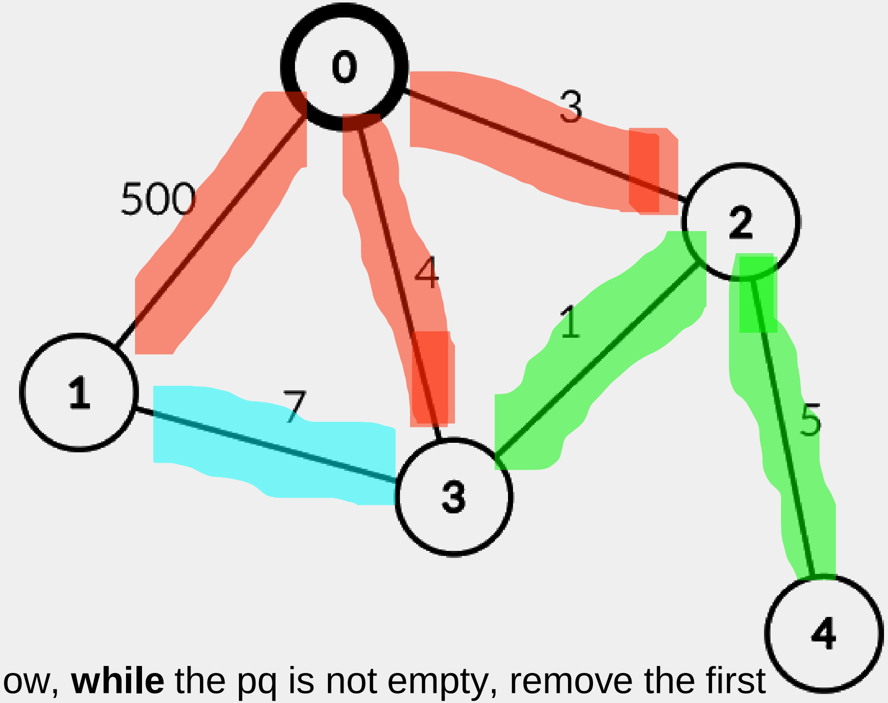


Now, **while** the pq is not empty, remove the first element of the pq and **consider** all of its edges

Removed ID 4 of weight 5

ID	edgeTo	distTo	marked	pq (index, value)
0	null	0	TRUE	(1, 7)
1	(3, 1)	7	FALSE	
2	(0, 2)	3	TRUE	
3	(2, 3)	1	TRUE	
4	(2, 4)	5	TRUE	

Prims Example



Now, **while** the pq is not empty, remove the first element of the pq and **consider** all of its edges

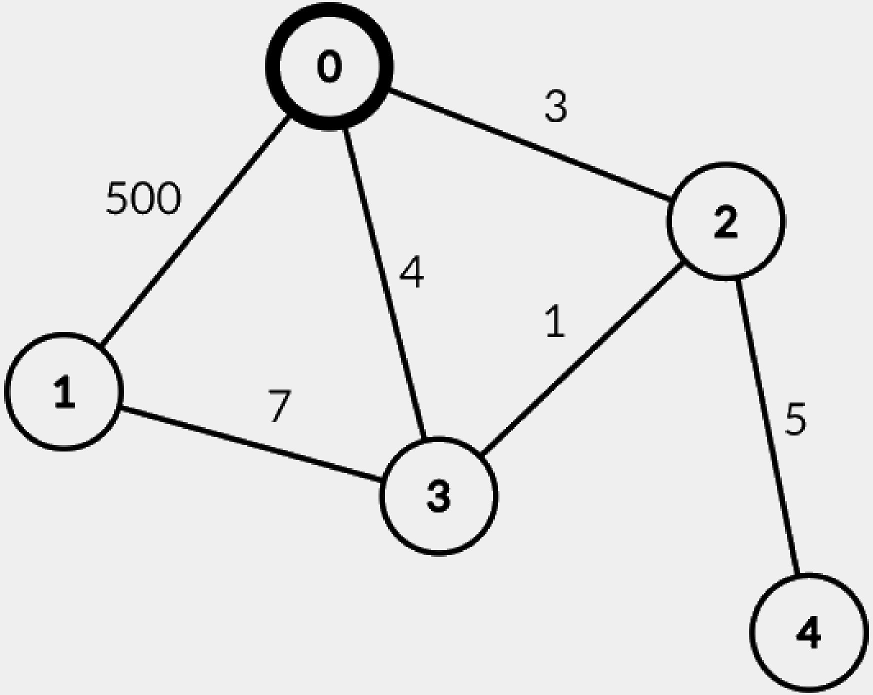
Removed ID 1 of weight 7

ID	edgeTo	distTo	marked	pq (index, value)
0	null	0	TRUE	
1	(3, 1)	7	TRUE	
2	(0, 2)	3	TRUE	
3	(2, 3)	1	TRUE	
4	(2, 4)	5	TRUE	

Dijkstras

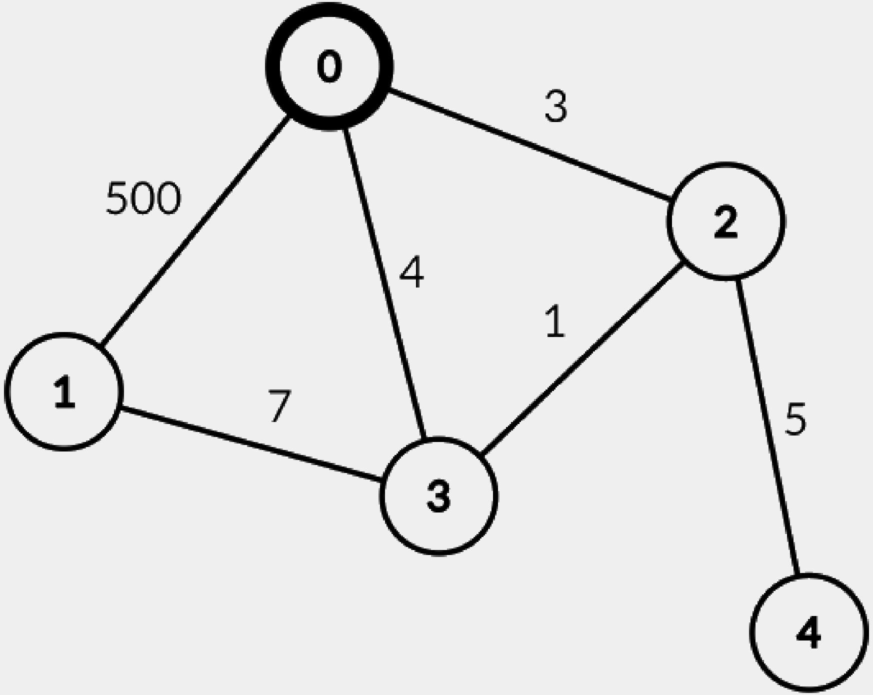
- Also Uses an Indexable Priority Queue.
- Primary difference from Prims is that it considers the total weight of the path to all nodes **from the source.**

Dijkstras Example



ID	edgeTo	distTo	marked	pq (index, value)
0	null	0	FALSE	
1	null	∞	FALSE	
2	null	∞	FALSE	
3	null	∞	FALSE	
4	null	∞	FALSE	

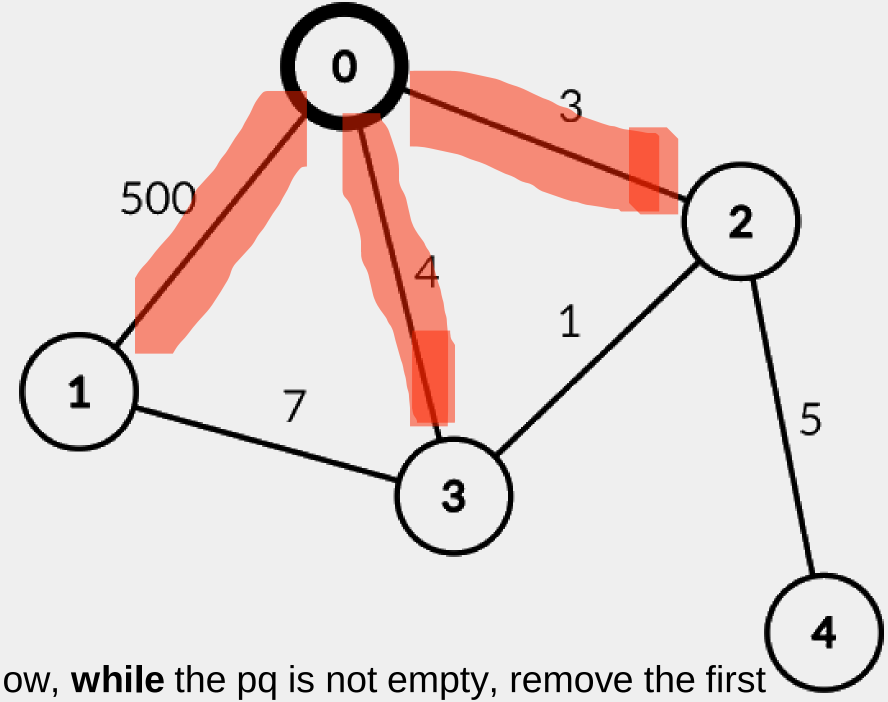
Dijkstras Example



Add the start node to the pq with a distance of 0

ID	edgeTo	distTo	marked	pq (index, value)
0	null	0	FALSE	(0, 0)
1	null	∞	FALSE	
2	null	∞	FALSE	
3	null	∞	FALSE	
4	null	∞	FALSE	

Dijkstras Example



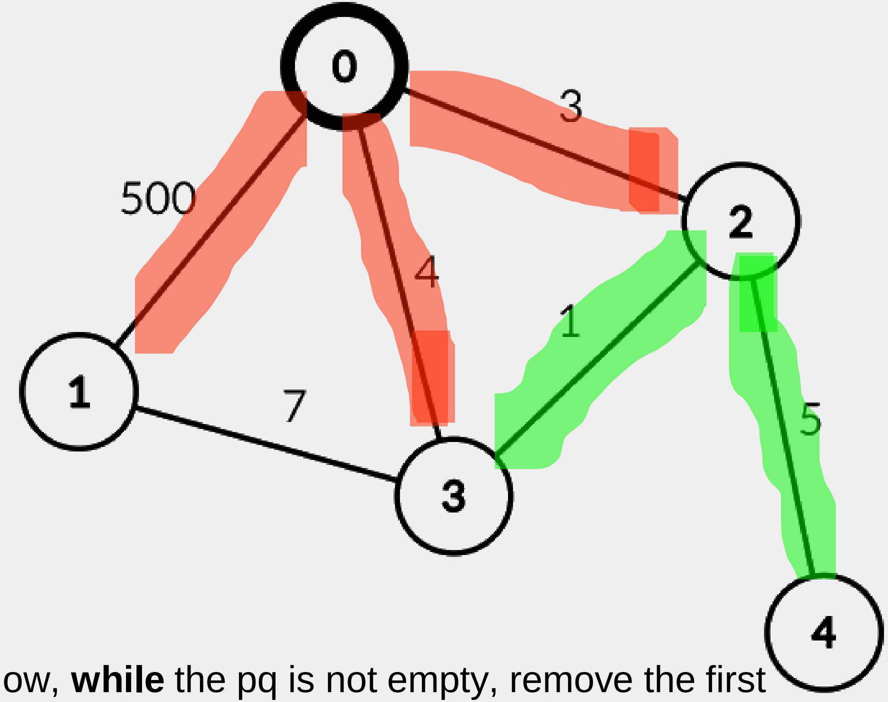
Now, **while** the pq is not empty, remove the first element of the pq and **consider** all of its edges.

This time, weight is calculated as a factor of how far it is from 0, not just the single edge's weight.

Removed ID 0 of weight 0

ID	edgeTo	distTo	marked	pq (index, value)
0	null	0	TRUE	(2, 3)
1	(0, 1)	500	FALSE	(3, 4)
2	(0, 2)	3	FALSE	(1, 500)
3	(0, 3)	4	FALSE	
4	null	∞	FALSE	

Dijkstras Example

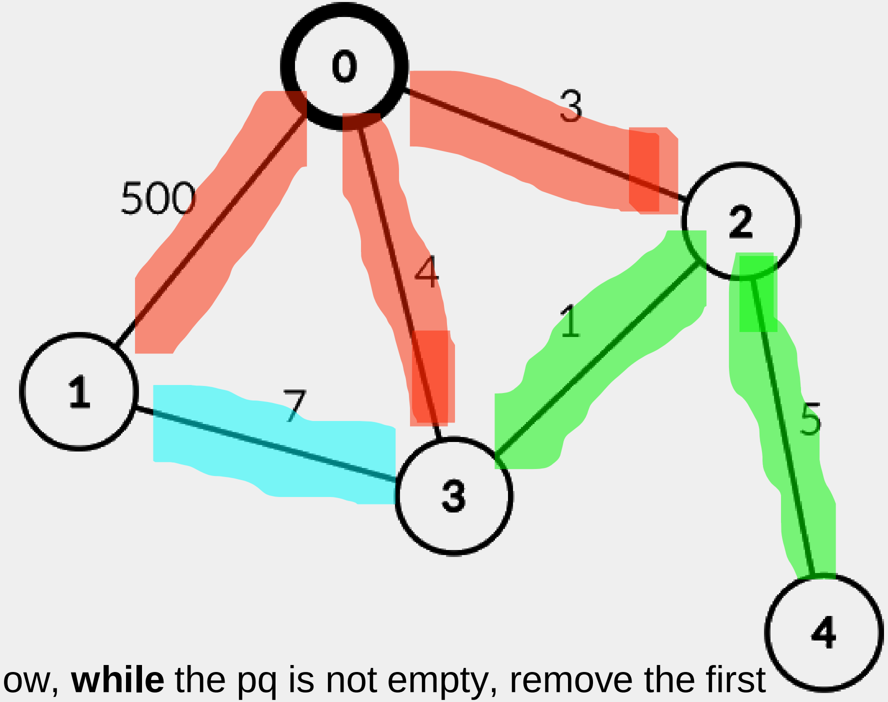


Now, **while** the pq is not empty, remove the first element of the pq and **consider** all of its edges

Removed ID 2 of weight 3

ID	edgeTo	distTo	marked	pq (index, value)
0	null	0	TRUE	(3, 4)
1	(0, 1)	500	FALSE	(4, 8)
2	(0, 2)	3	TRUE	(1, 500)
3	(2, 3)	4	FALSE	
4	(2, 4)	8	FALSE	

Dijkstras Example

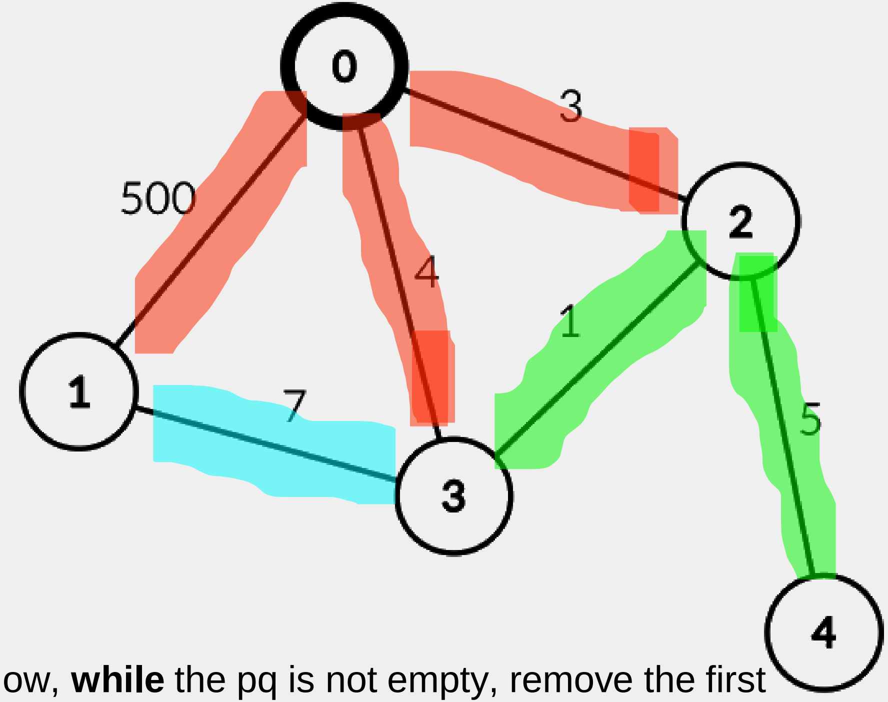


Now, **while** the pq is not empty, remove the first element of the pq and **consider** all of its edges

Removed ID 3 of weight 4

ID	edgeTo	distTo	marked	pq (index, value)
0	null	0	TRUE	(4, 8)
1	(0, 1)	11	FALSE	(1, 11)
2	(0, 2)	3	TRUE	
3	(2, 3)	4	TRUE	
4	(2, 4)	8	FALSE	

Dijkstras Example

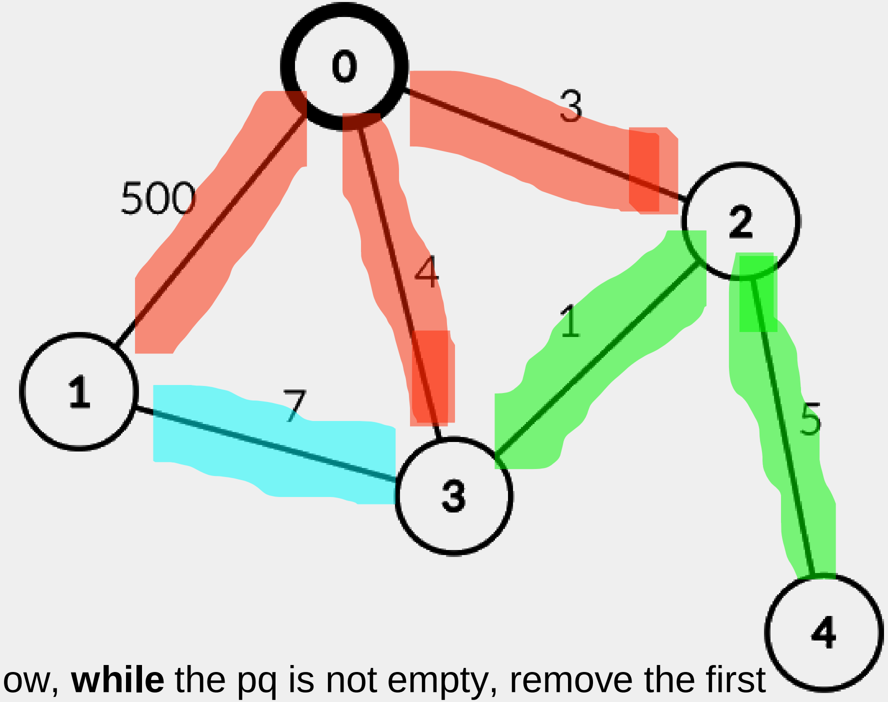


Now, **while** the pq is not empty, remove the first element of the pq and **consider** all of its edges

Removed ID 4 of weight 8

ID	edgeTo	distTo	marked	pq (index, value)
0	null	0	TRUE	(1, 11)
1	(0, 1)	11	FALSE	
2	(0, 2)	3	TRUE	
3	(2, 3)	4	TRUE	
4	(2, 4)	8	TRUE	

Dijkstras Example



Now, **while** the pq is not empty, remove the first element of the pq and **consider** all of its edges

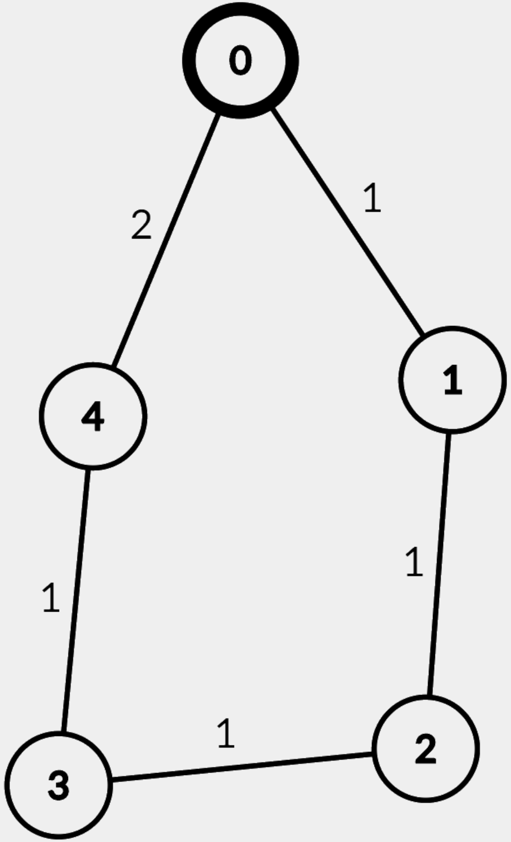
Removed ID 1 of weight 11

ID	edgeTo	distTo	marked	pq (index, value)
0	null	0	TRUE	
1	(0, 1)	11	TRUE	
2	(0, 2)	3	TRUE	
3	(2, 3)	4	TRUE	
4	(2, 4)	8	TRUE	

So What?

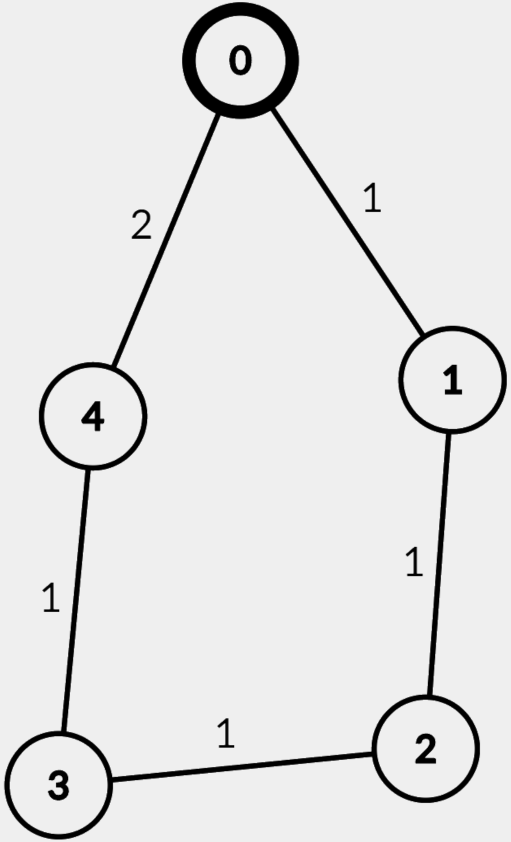
- We got the same result. What’s the difference?
- That’s just because we were lucky. Here’s a different example.

Different Example



- What will Prims do?
- What will Dijkstras do?

Different Example



- Prims will say that the edges of weight 1 are always better than edges of higher weights.

- Dijkstras will consider the weight of each edge in regards to its full path from the source (0), therefore 2 < 4 and the best path to 4 is through 0.

Now what?

- Since Prims is so close to Dijkstra, we’re going to:
  - Reverse Engineer Prims so we understand how it works.
  - Convert Prims to Dijkstras