Kruskal's Algorithm

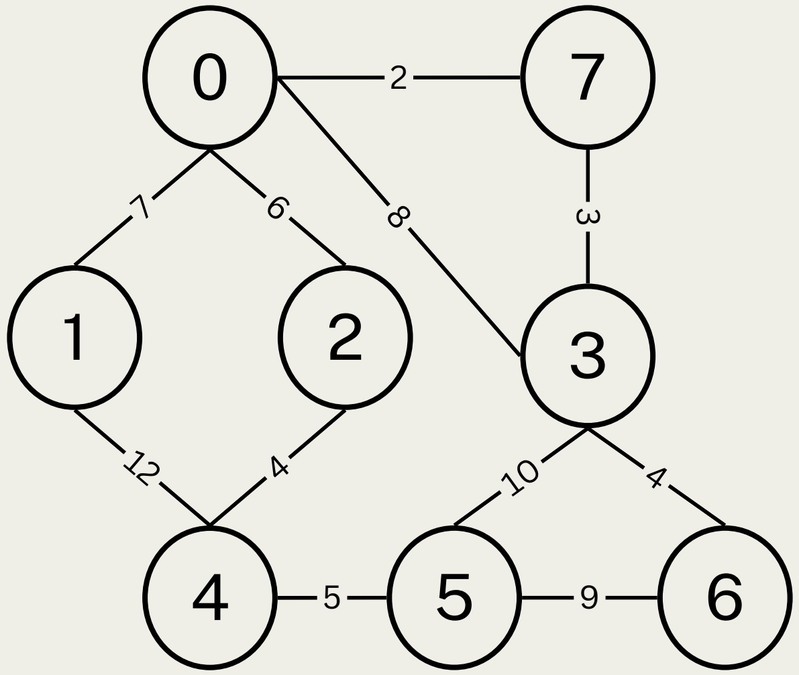
PA 6 HELP: ILLUSTRATING THE ALGORITHM AND ITS COMPLICATIONS

CPSC-350

Dr. EEL

Let’s run Kruskal’s on this graph:

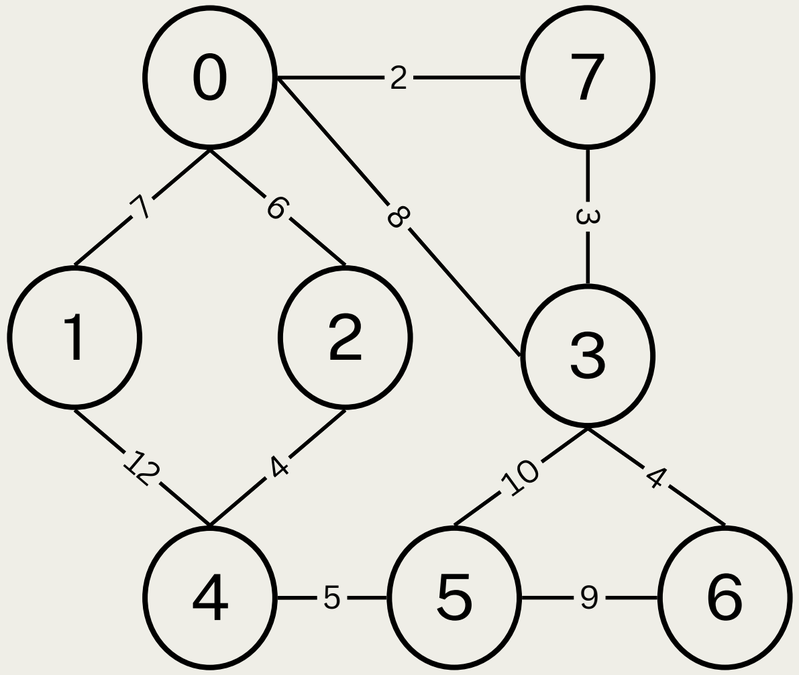
graph:



MST:

Let’s run Kruskal’s on this graph:

graph:

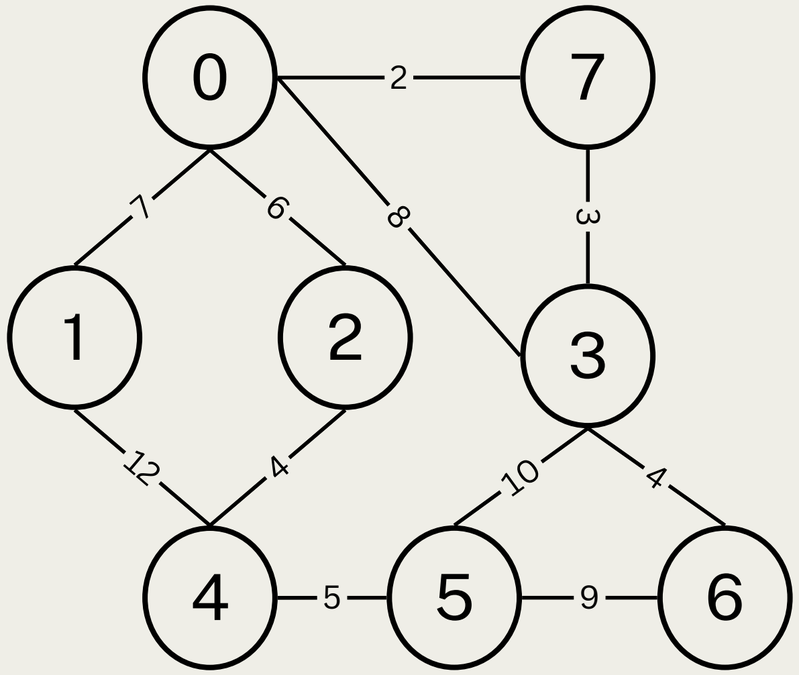


MST:

First, create a min priority queue with all of the edges. It is up to you to decide how you want to represent the edges in this p-queue. I will represent each edge with a tuple of (node1, node2, weight), note that this is an undirected graph.

Let’s run Kruskal’s on this graph:

graph:



MST:

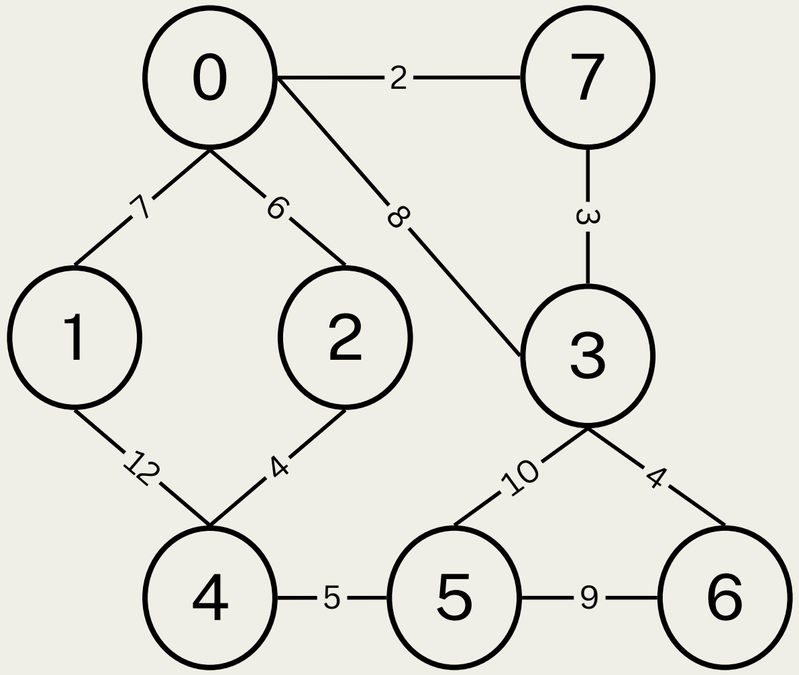
First, create a min priority queue with all of the edges. It is up to you to decide how you want to represent the edges in this p-queue. I will represent each edge with a tuple of (node1, node2, weight), note that this is an undirected graph.

p-queue:

[(0,7,2), (3,7,3), (2,4,4), (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

Let’s run Kruskal’s on this graph:

graph:



MST:

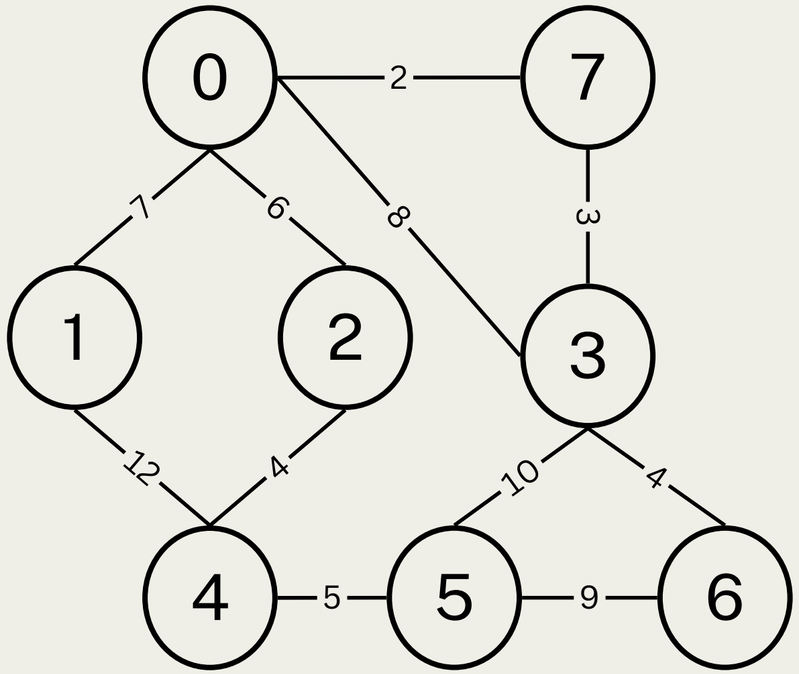
Now, we will remove the edges from the p-queue and decide whether they should be added to the MST or not (do they take us to a node we haven’t been to yet?)

p-queue:

[(0,7,2), (3,7,3), (2,4,4), (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

Let’s run Kruskal’s on this graph:

graph:



MST:

Now, we will remove the edges from the p-queue and decide whether they should be added to the MST or not (do they take us to a node we haven’t been to yet?)

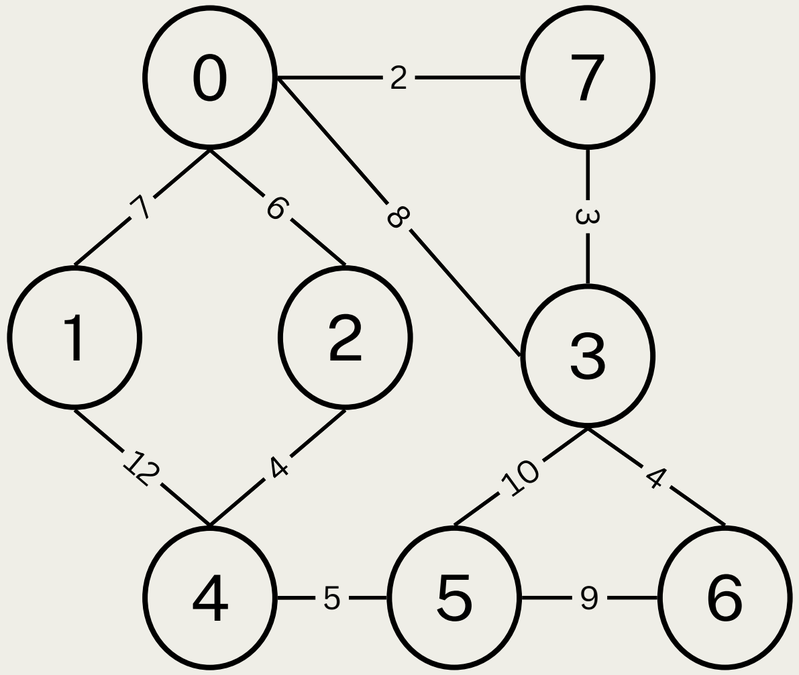
Notice how we will need to keep track of nodes we’ve visited in order to answer this question. I will keep track of nodes I have been to with a set.

p-queue:

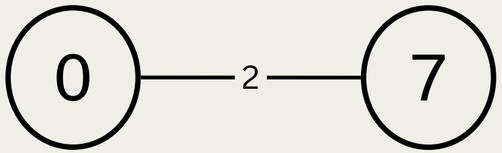
[(0,7,2), (3,7,3), (2,4,4), (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

Let’s run Kruskal’s on this graph:

graph:



MST:



We remove the first edge and we add it to the MST and to the set since we haven’t visited either 0 or 2 (we haven’t visited any nodes yet)

p-queue:

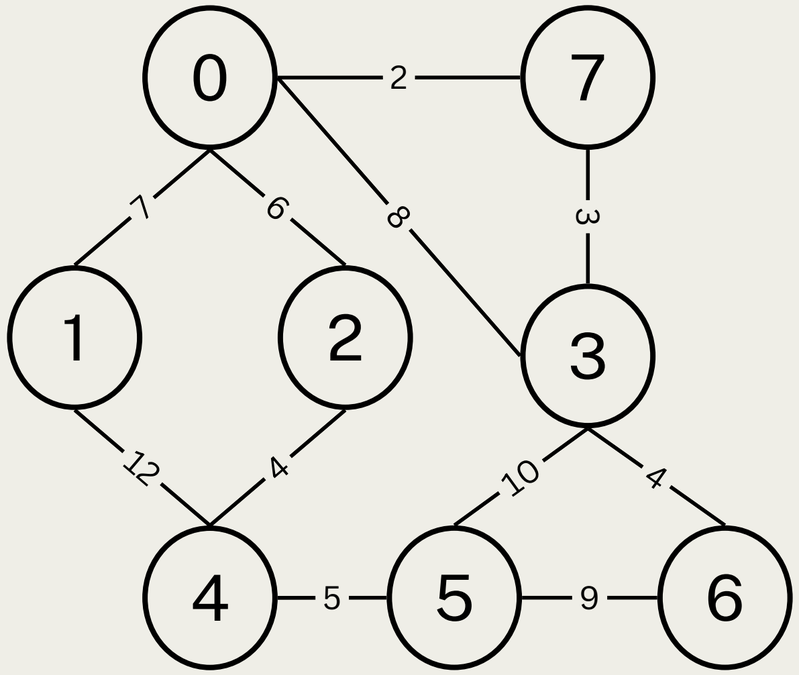
[~~(0,7,2)~~, (3,7,3), (2,4,4), (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

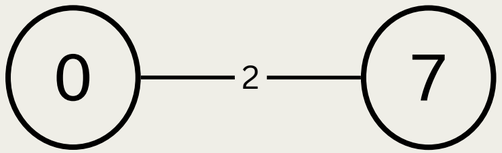
[(0,7,

Let’s run Kruskal’s on this graph:

graph:



MST:



Now, we take out the next edge and check whether adding this edge to the MST will help us connect to any new node. If yes, then add it to MST if not, then skip it.

p-queue:

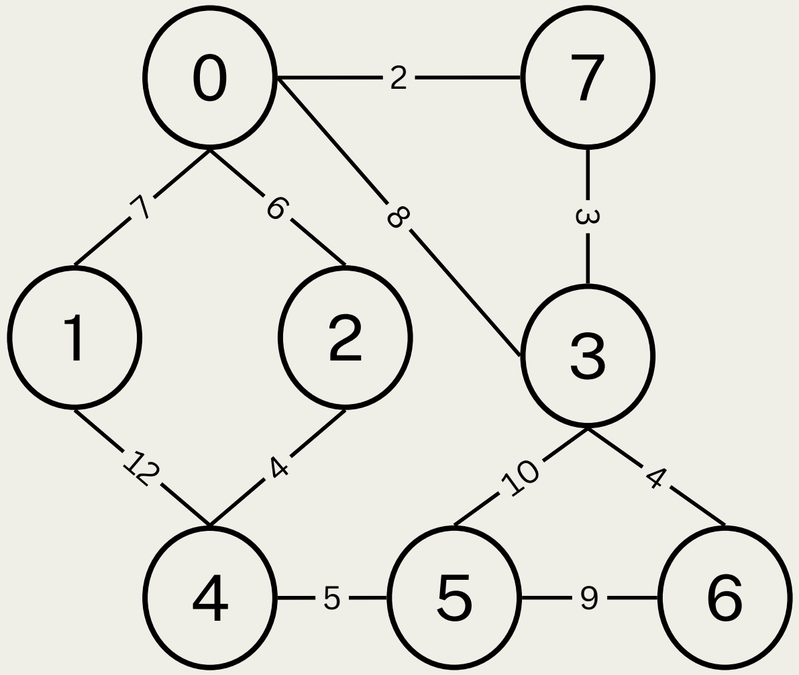
[~~(0,7,2)~~, (3,7,3), (2,4,4), (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

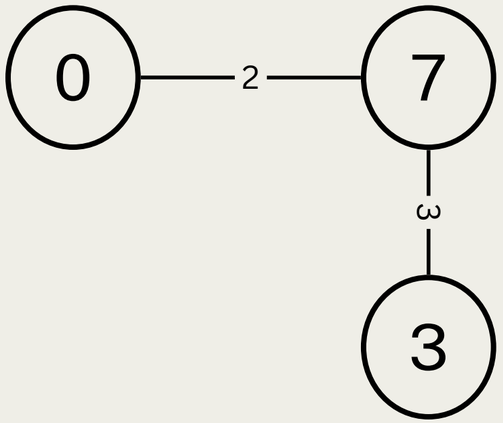
[(0,7,

Let’s run Kruskal’s on this graph:

graph:



MST:



When checking the edge (3,7,3), we note that we have already visited node 7 but we haven’t been to node 3 yet so we add this edge to the MST and node 3 to the visited set.

p-queue:

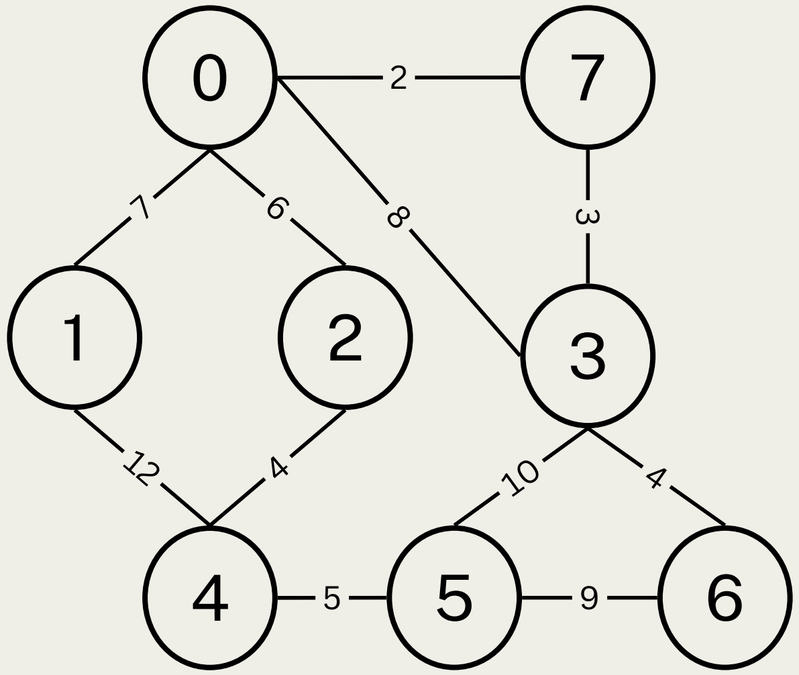
[~~(0,7,2)~~, ~~(3,7,3)~~, (2,4,4), (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

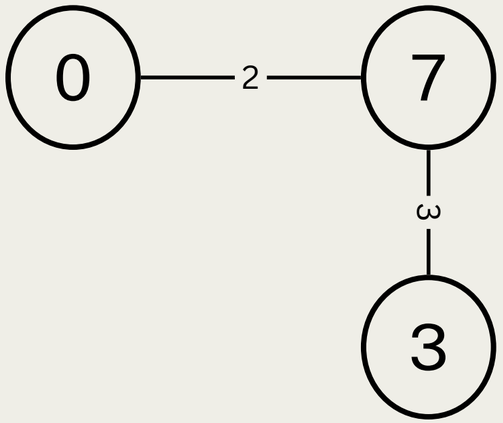
[(0, 7, 3

Let’s run Kruskal’s on this graph:

graph:



MST:



Now, onto the next edge. (2,4,4) we want to add this edge because we have not been to either 2 nor 4.

p-queue:

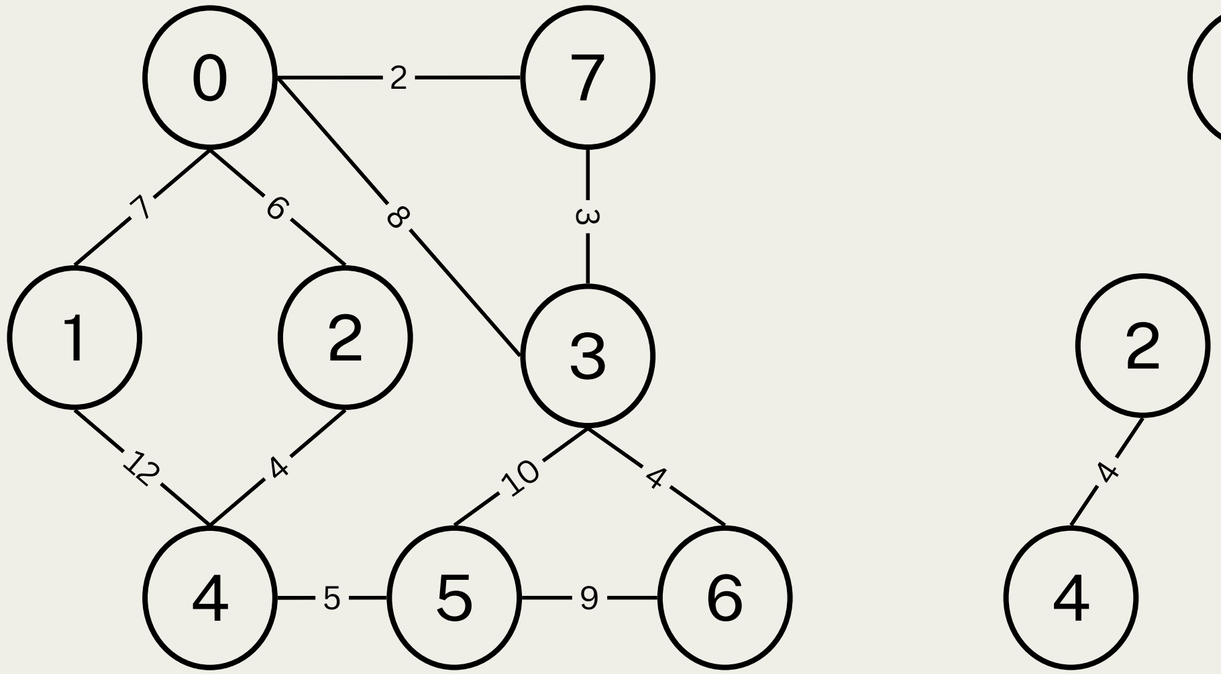
[~~(0,7,2)~~, ~~(3,7,3)~~, (2,4,4), (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

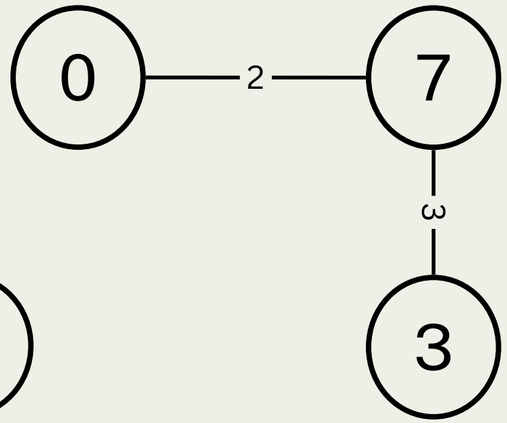
[(0, 7, 3

Let’s run Kruskal’s on this graph:

graph:



MST:



Now, onto the next edge. (2,4,4) we want to add this edge because we have not been to either 2 nor 4.

This is where the problem begins,,, but we will continue this process as we’ve done so far to illustrate the issue.

p-queue:

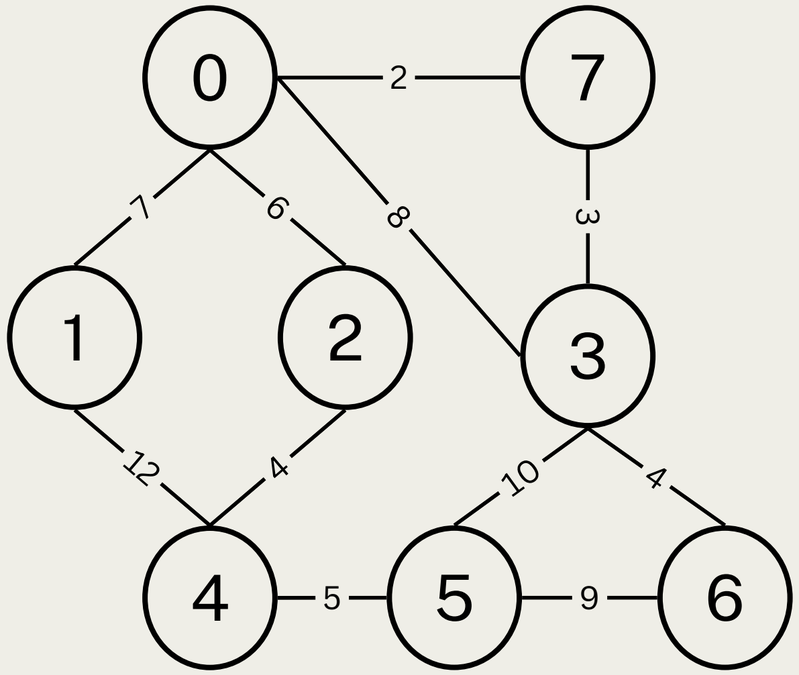
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

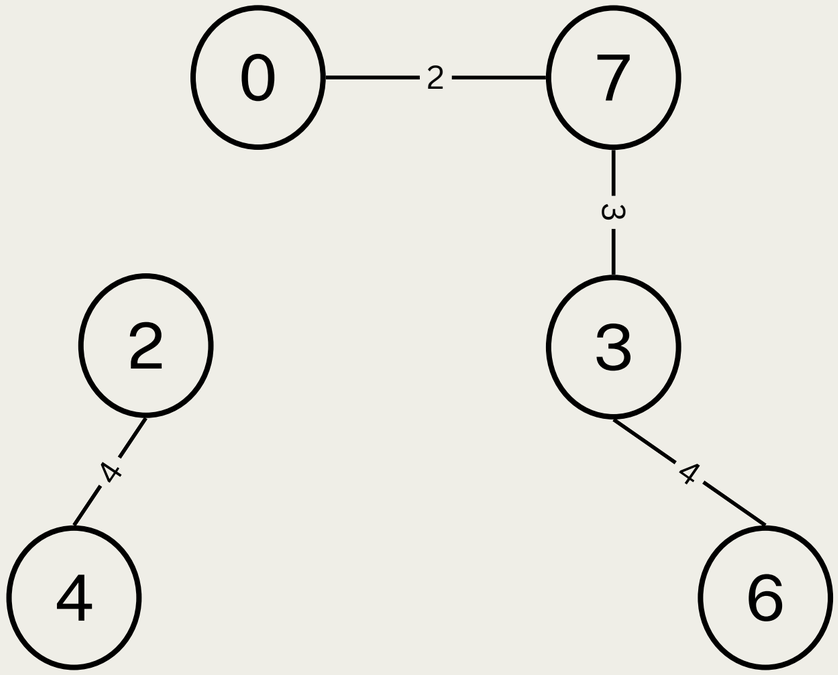
[(0, 7, 3, 2, 4,

Let’s run Kruskal’s on this graph:

graph:



MST:



We check the next edge. (3,6,4) takes us to 6 so we add it.

p-queue:

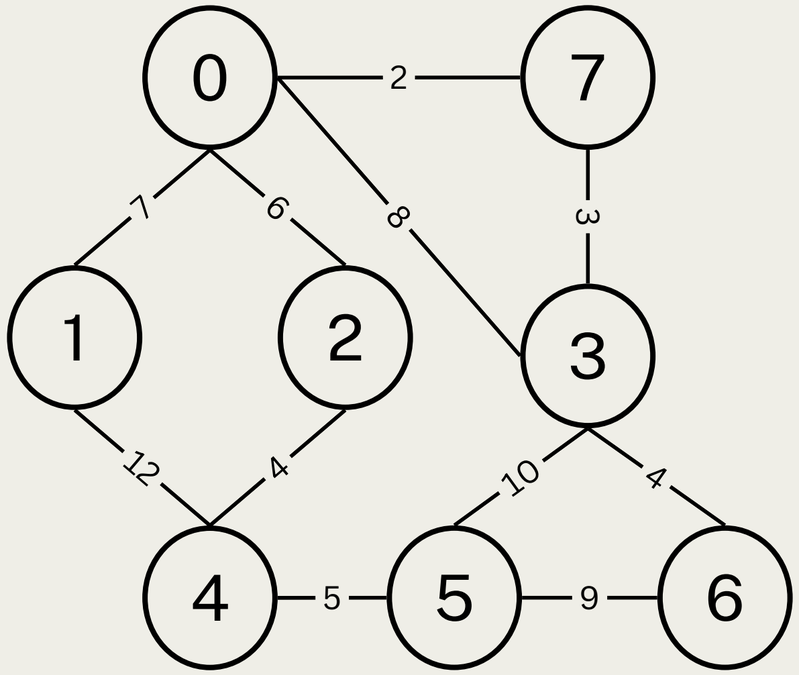
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

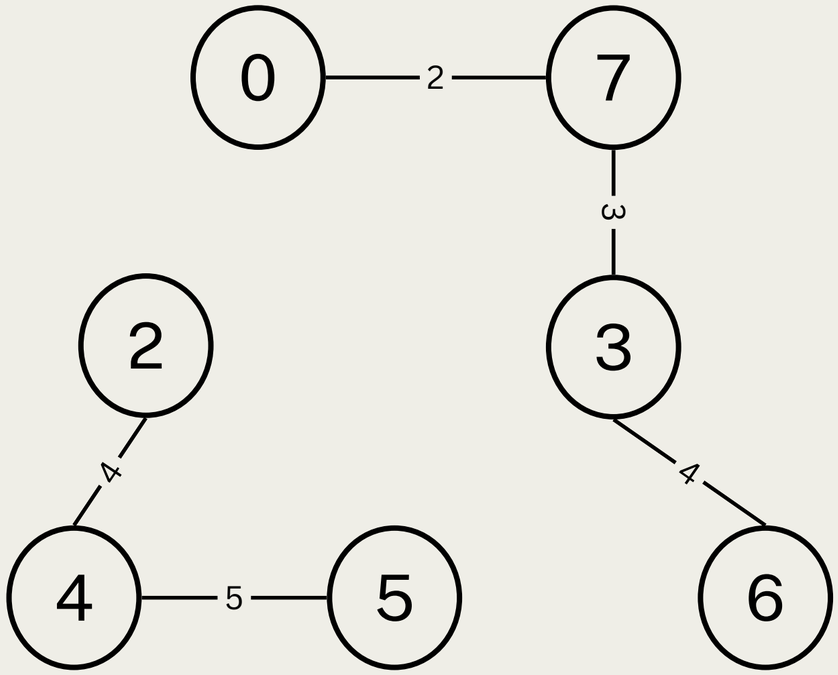
[(0, 7, 3, 2, 4, 6,

Let’s run Kruskal’s on this graph:

graph:



MST:



now the next edge, (4,5,5) takes us to 5 so we also add it.

p-queue:

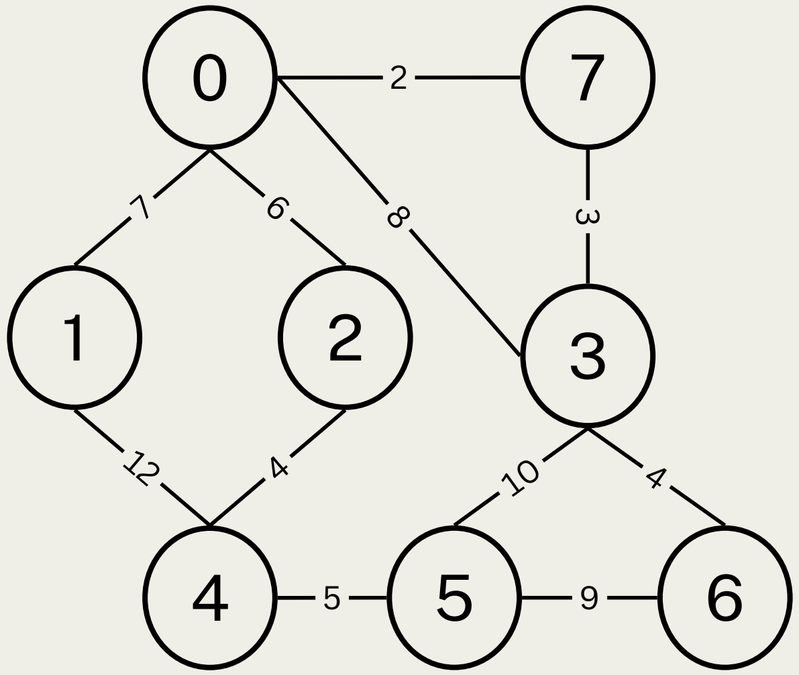
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

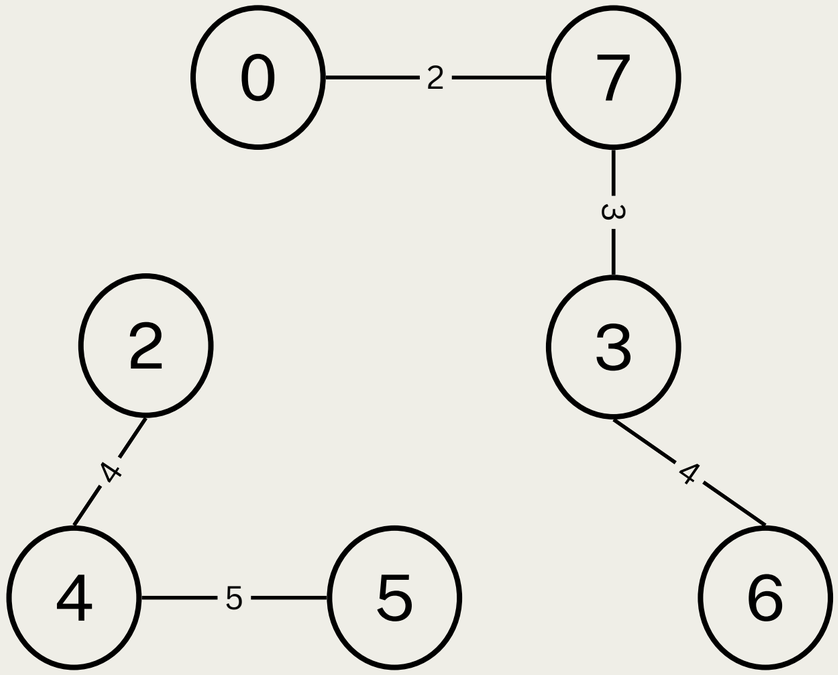
[(0, 7, 3, 2, 4, 6, 5,

Let’s run Kruskal’s on this graph:

graph:



MST:



We now get to (0,2,6) Do we add it????
Well, in our visited we already hace node 0 and node 2.

p-queue:

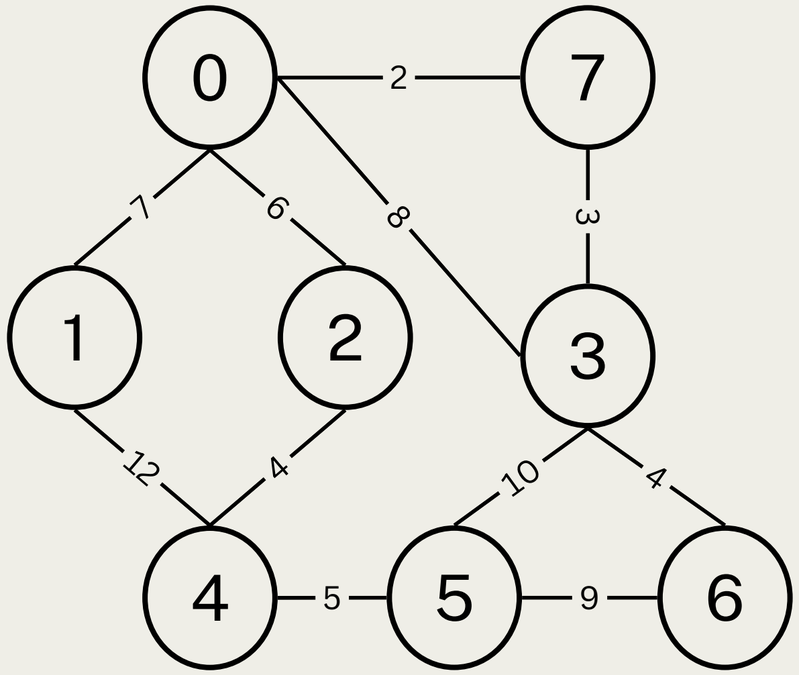
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~,
(0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

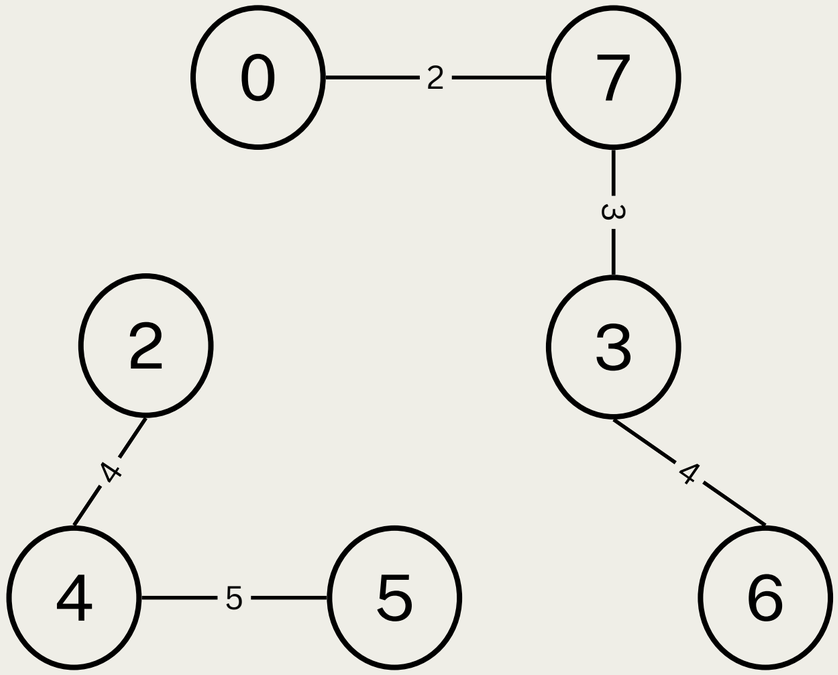
[(0, 7, 3, 2, 4, 6, 5,

Let’s run Kruskal’s on this graph:

graph:



MST:



We now get to (0,2,6) Do we add it????
Well, in our visited we already have node 0 and node 2
So, let’s assume (wrongly) that we can skip this edge since it doesn’t take us anywhere new.

p-queue:

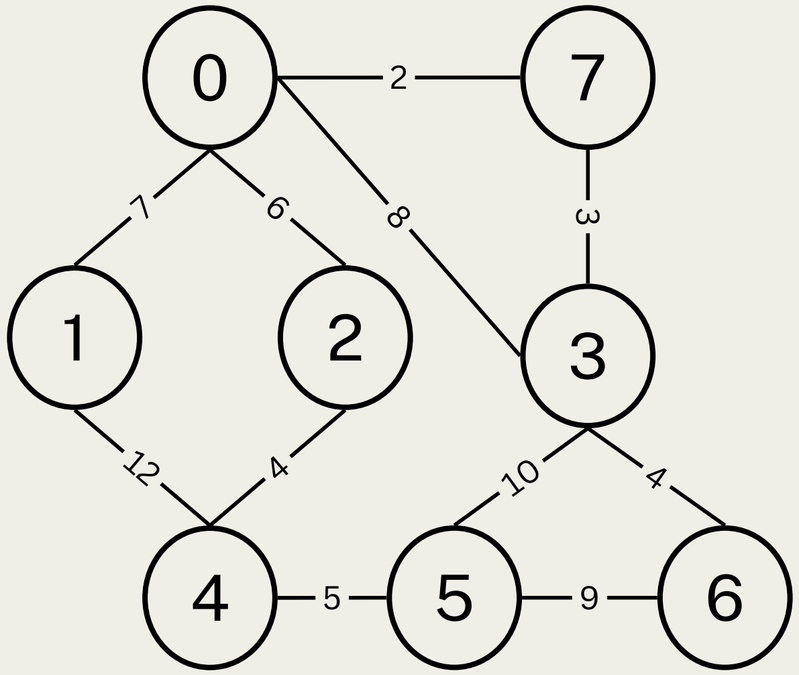
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~,
(0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

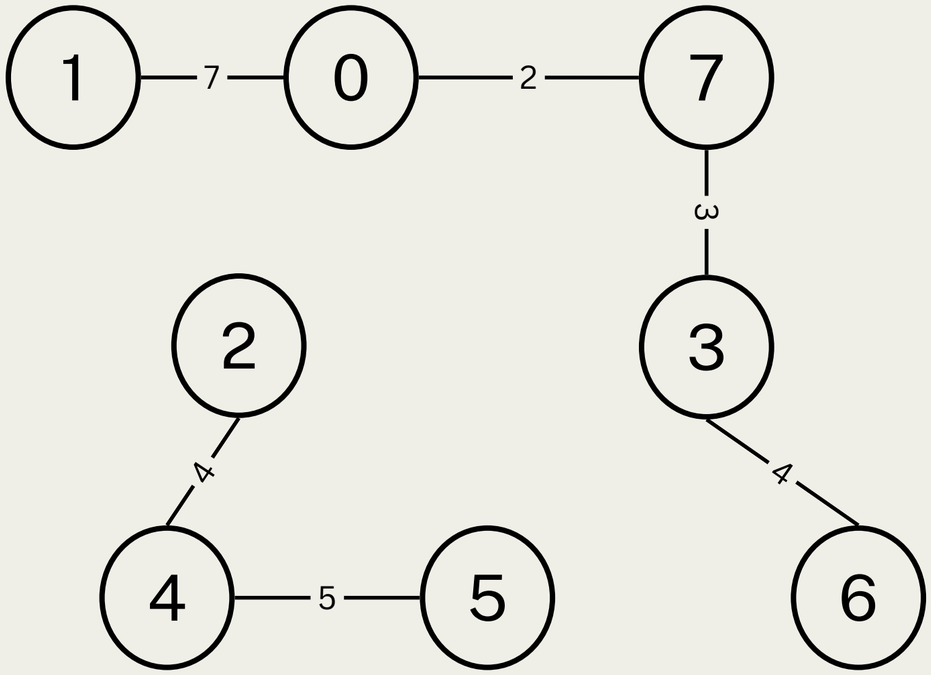
[(0, 7, 3, 2, 4, 6, 5,

Let’s run Kruskal’s on this graph:

graph:



MST:



The next edge is (0,1,7) We haven’t been to 1 yet so we will add this edge

p-queue:

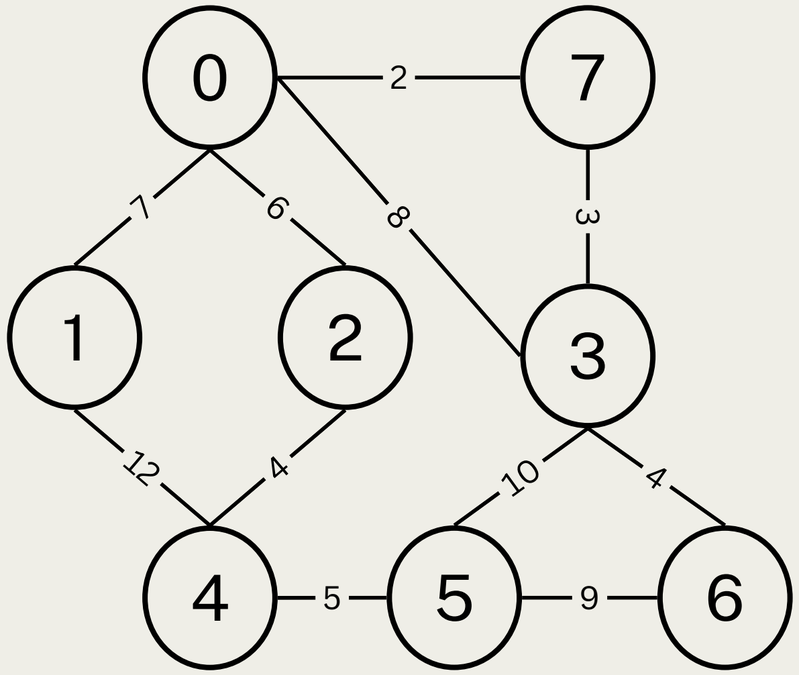
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~, ~~(0,1,7)~~, (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

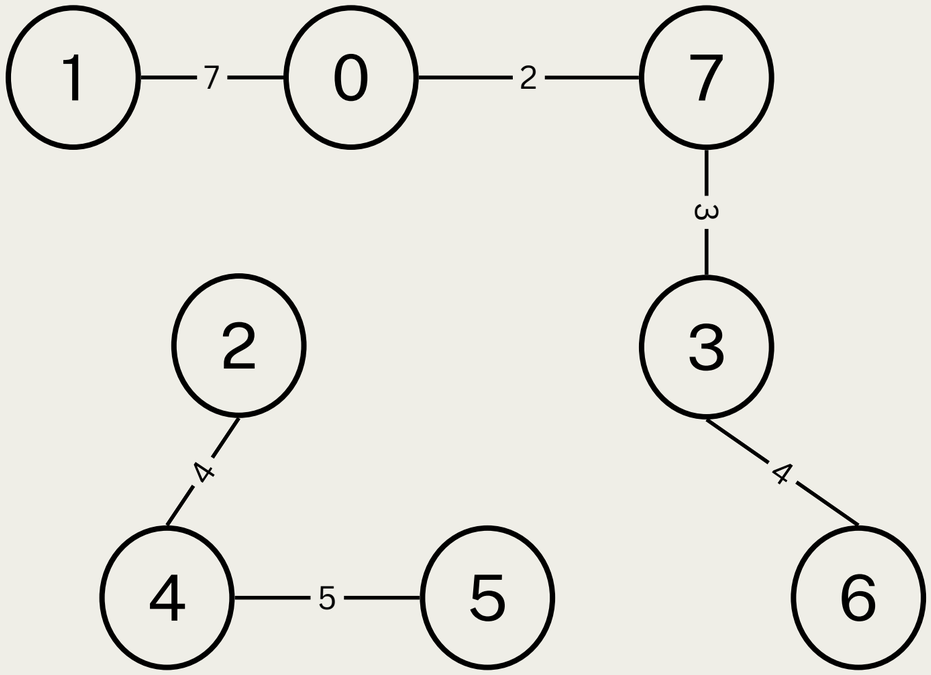
[(0, 7, 3, 2, 4, 6, 5, 1

Let’s run Kruskal’s on this graph:

graph:



MST:



The next edge is (0,3,8) We have been to both 0 and 3 so we do NOT add this edge.

p-queue:

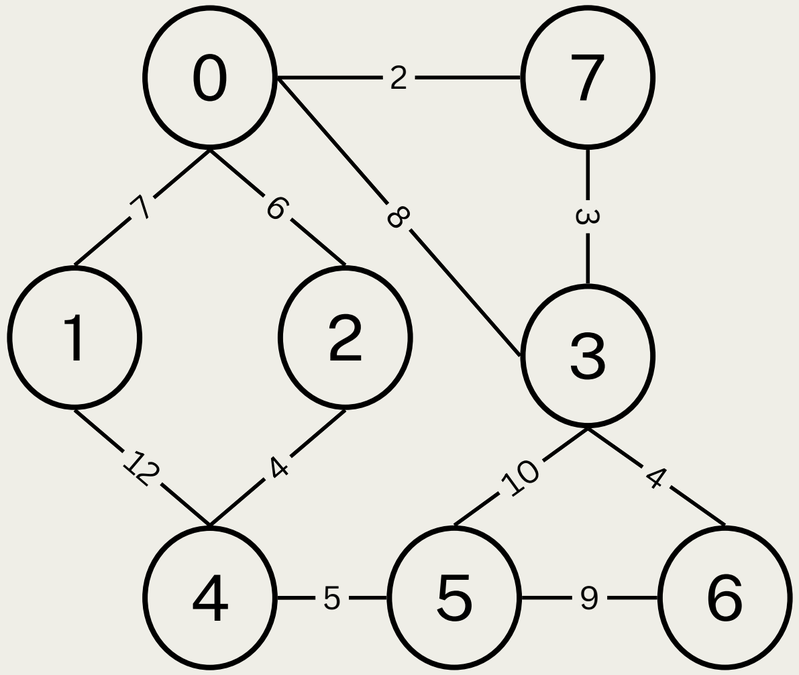
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~, ~~(0,1,7)~~, ~~(0,3,8)~~, (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

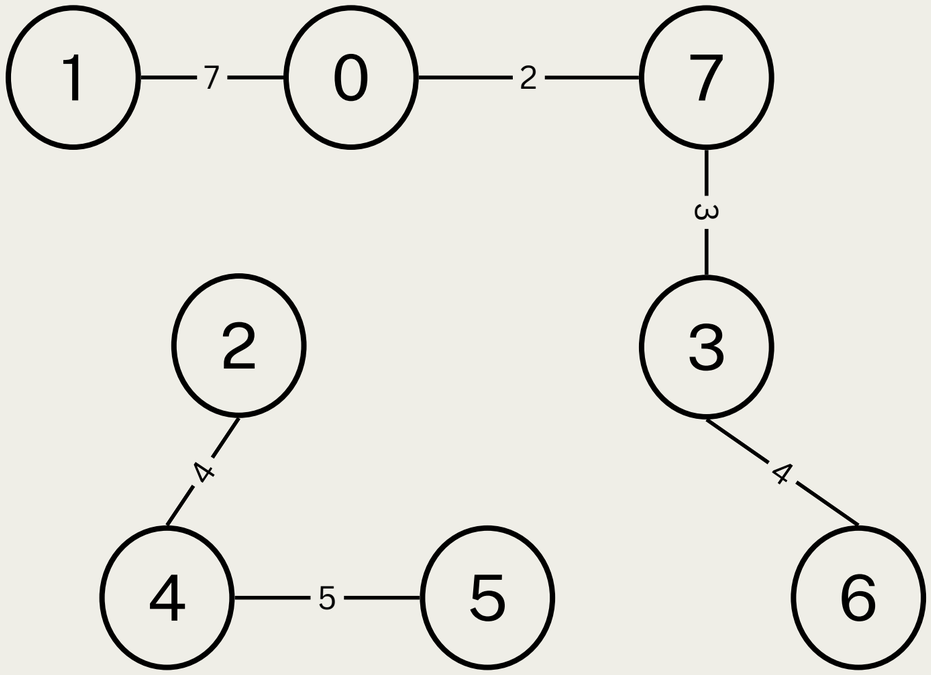
[(0, 7, 3, 2, 4, 6, 5, 1,

Let’s run Kruskal’s on this graph:

graph:



MST:



The next edge is (5,6,9) We have been to both 5 and 6 so we do NOT add this edge.

p-queue:

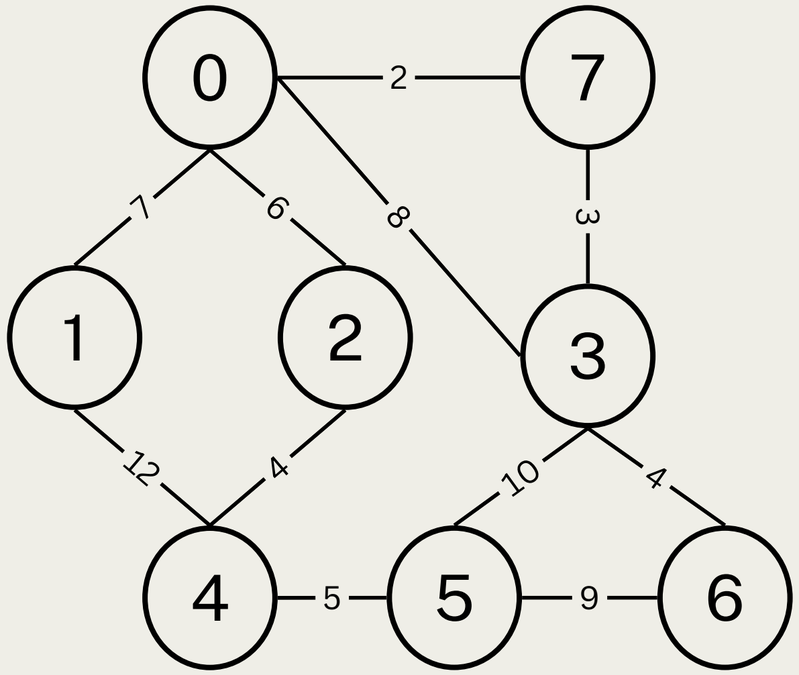
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~, ~~(0,1,7)~~, ~~(0,3,8)~~, ~~(5,6,9)~~, (3,5,10), (1,4,12)]

nodes visited:

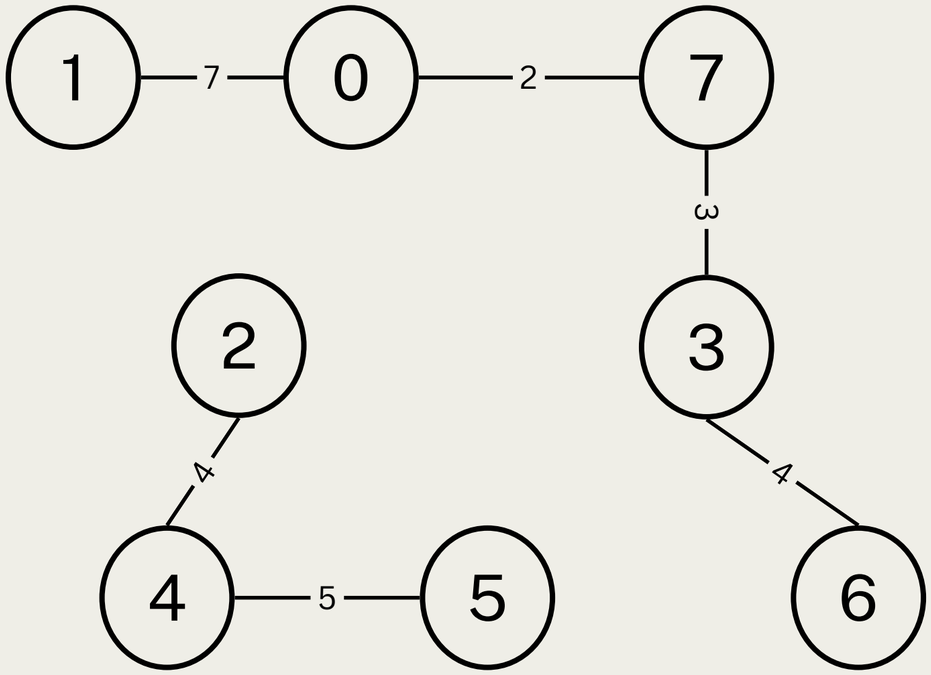
[(0, 7, 3, 2, 4, 6, 5, 1,

Let’s run Kruskal’s on this graph:

graph:



MST:



The next edge is (3,5,10) We have been to both 3 and 5 so we do NOT add this edge.

p-queue:

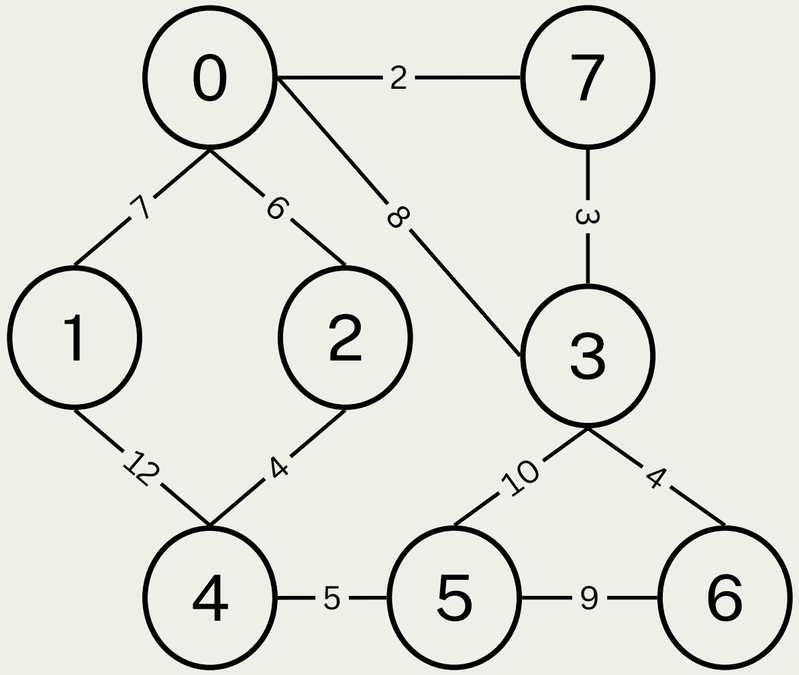
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~, ~~(0,1,7)~~, ~~(0,3,8)~~, ~~(5,6,9)~~, ~~(3,5,10)~~, (1,4,12)]

nodes visited:

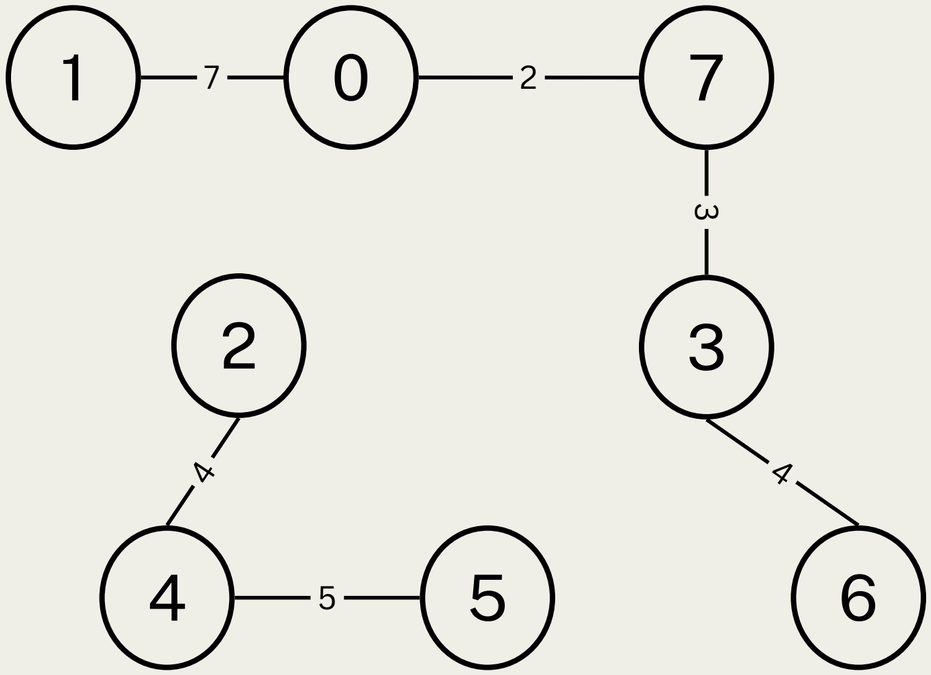
[(0, 7, 3, 2, 4, 6, 5, 1,

Let’s run Kruskal’s on this graph:

graph:



MST:



The next edge is (1,4,12) We have been to both 1 and 4 so we do NOT add this edge.

p-queue:

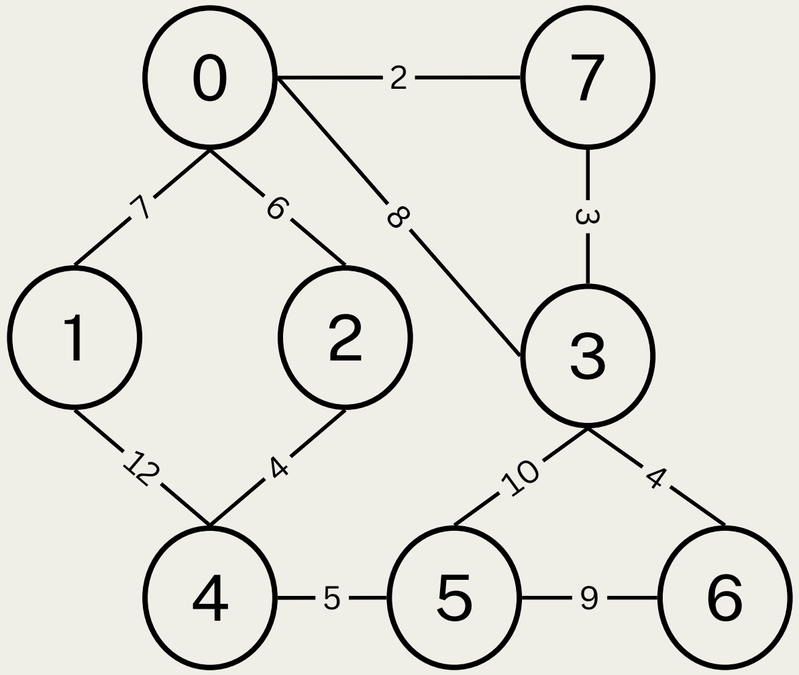
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~, ~~(0,1,7)~~, ~~(0,3,8)~~, ~~(5,6,9)~~, ~~(3,5,10)~~, ~~(1,4,12)~~]

nodes visited:

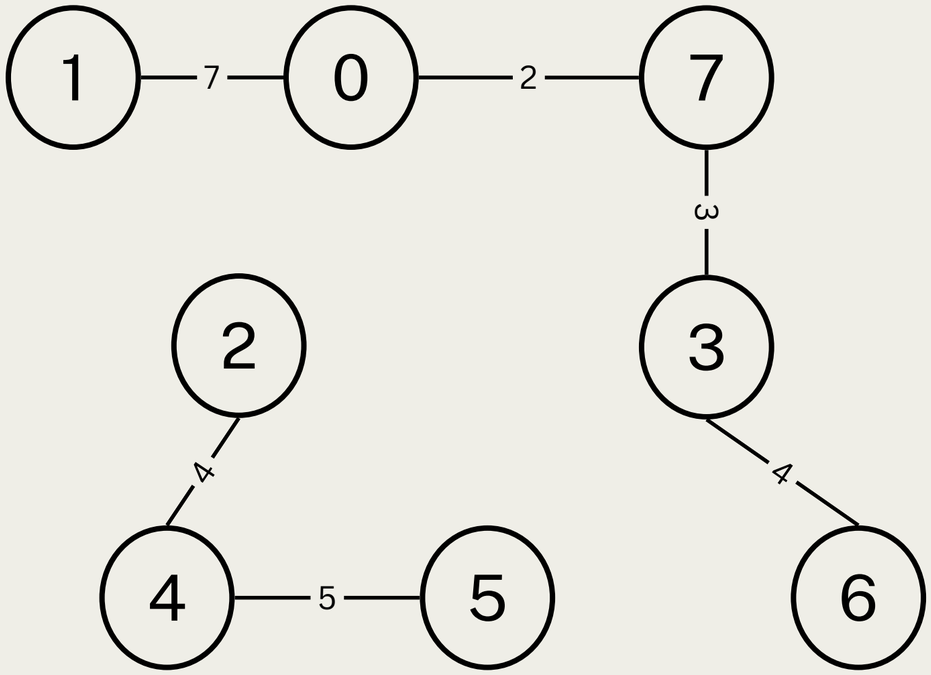
[(0, 7, 3, 2, 4, 6, 5, 1,

Let’s run Kruskal’s on this graph:

graph:



MST:



Our priority queue is now empty but... what do you notice about our MST?

p-queue:

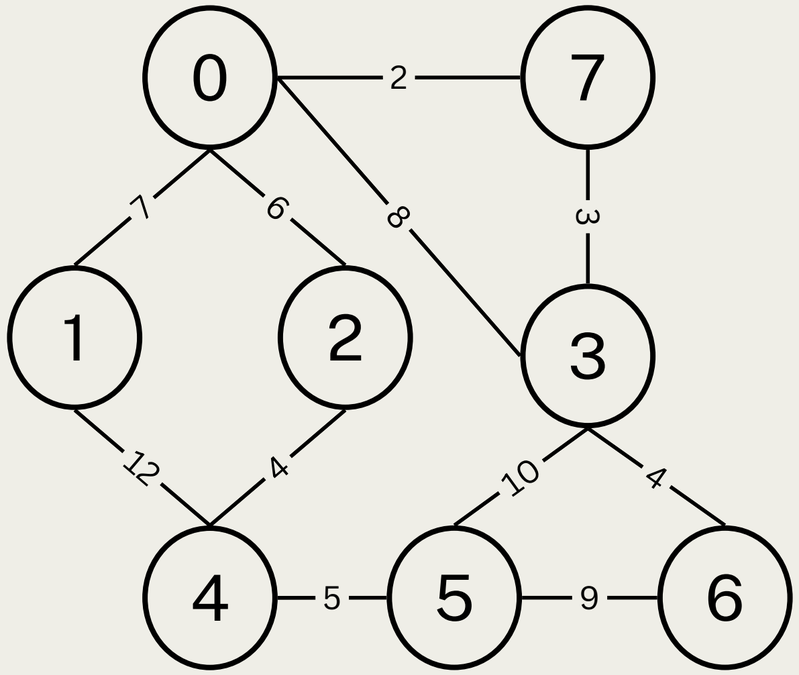
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~, ~~(0,1,7)~~, ~~(0,3,8)~~, ~~(5,6,9)~~, ~~(3,5,10)~~, ~~(1,4,12)~~]

nodes visited:

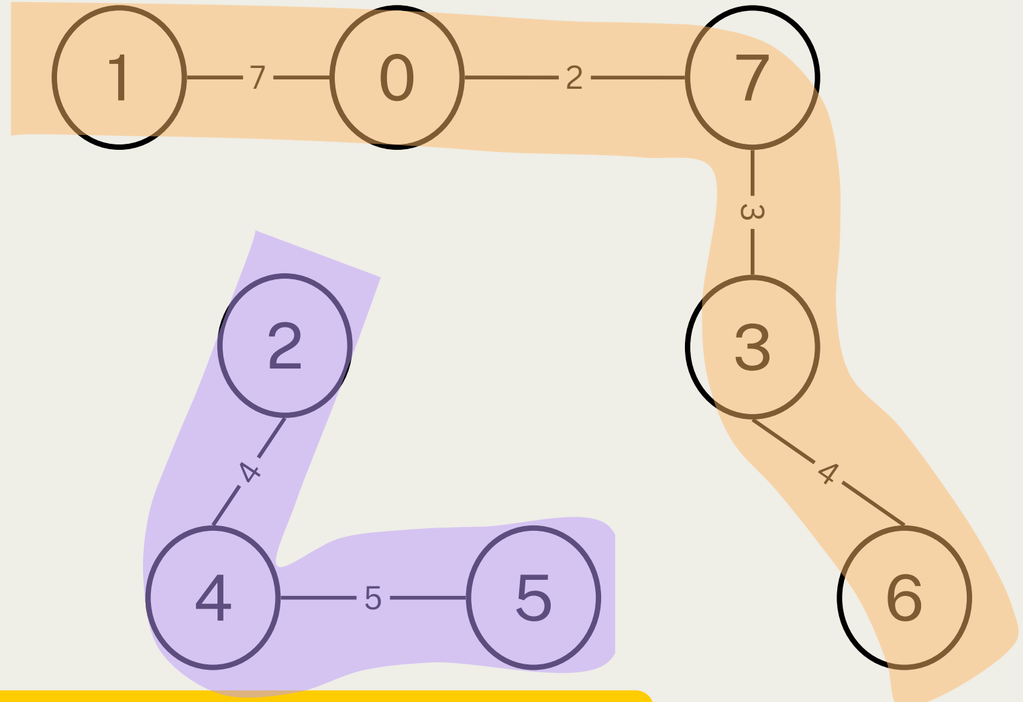
[(0, 7, 3, 2, 4, 6, 5, 1,

Let's run Kruskal's on this graph:

graph:



MST:



Our priority queue is now empty but... what do you notice about our MST? THERE ARE MULTIPLE, DISCONNECTED COMPONENTS! We don't want this as the whole point of an MST is to tell us the cheapest way to connect all nodes of a graph together

p-queue:

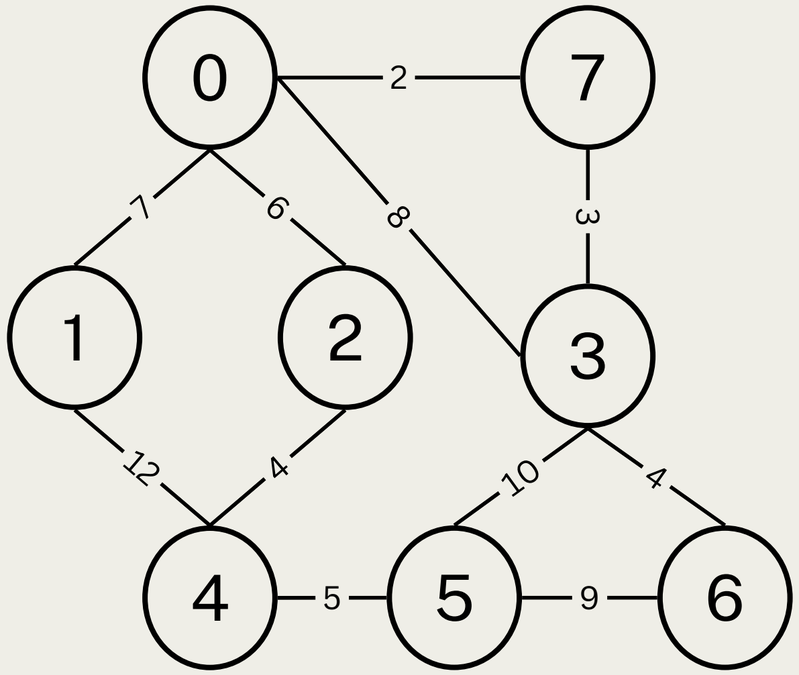
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~, (0,1,7), ~~(0,3,8)~~, ~~(5,6,9)~~, ~~(3,5,10)~~, ~~(1,4,12)~~]

nodes visited:

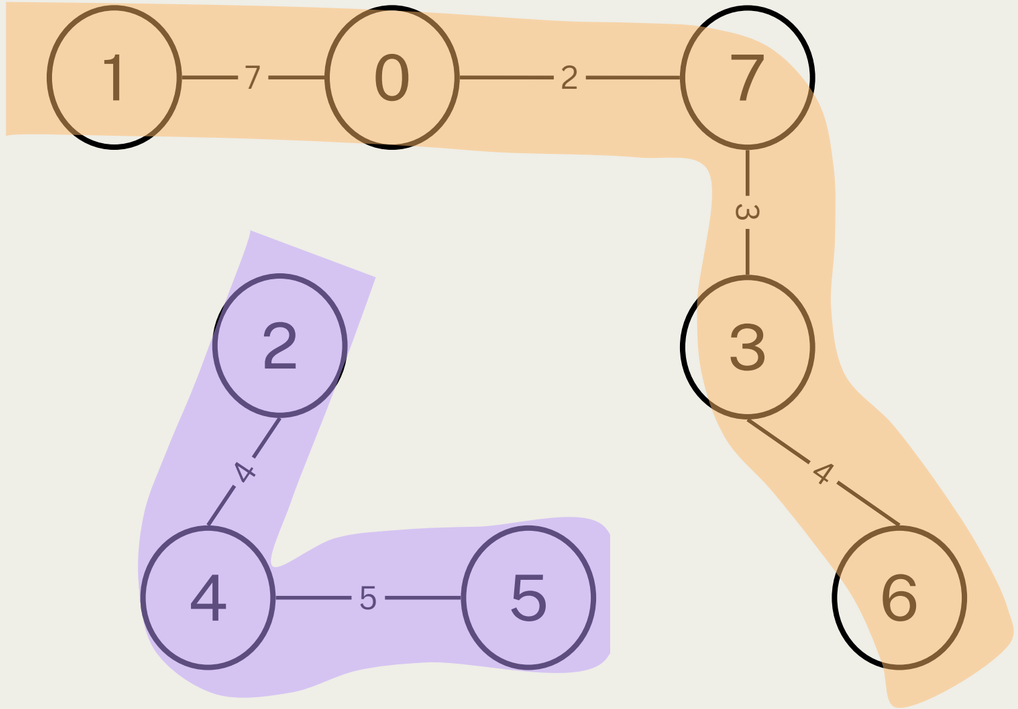
[(0, 7, 3, 2, 4, 6, 5, 1,

Let’s run Kruskal’s on this graph:

graph:



MST:



In order to fix and avoid this issue we must keep track of unconnected components. Let’s do this the right way now!

p-queue:

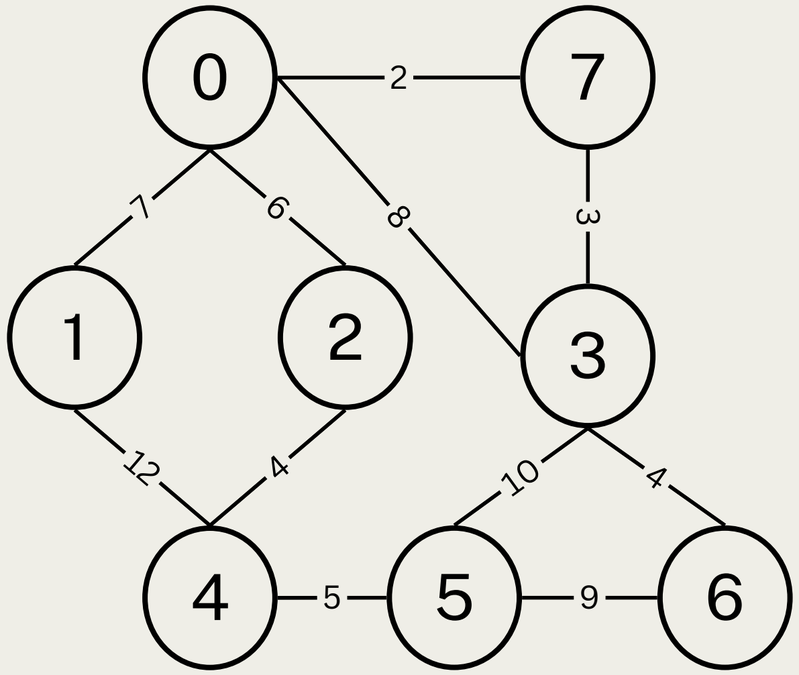
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~, ~~(0,1,7)~~, ~~(0,3,8)~~, ~~(5,6,9)~~, ~~(3,5,10)~~, ~~(1,4,12)~~]

nodes visited:

[(0, 7, 3, 2, 4, 6, 5, 1,

Let’s run Kruskal’s on this graph:

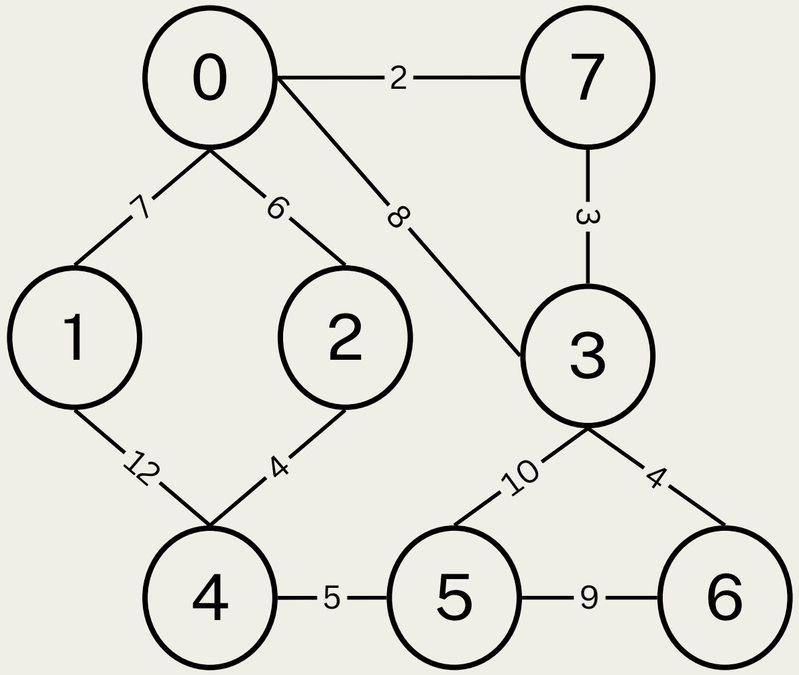
graph:



MST:

Let’s run Kruskal’s on this graph:

graph:

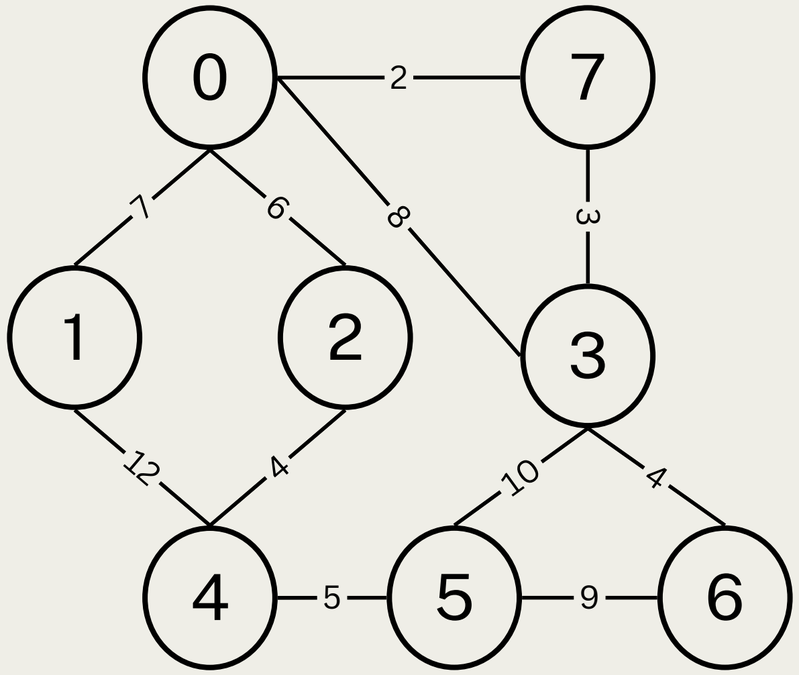


MST:

First, create a min priority queue with all of the edges. It is up to you to decide how you want to represent the edges in this p-queue. I will represent each edge with a tuple of (node1, node2, weight), note that this is an undirected graph.

Let’s run Kruskal’s on this graph:

graph:



MST:

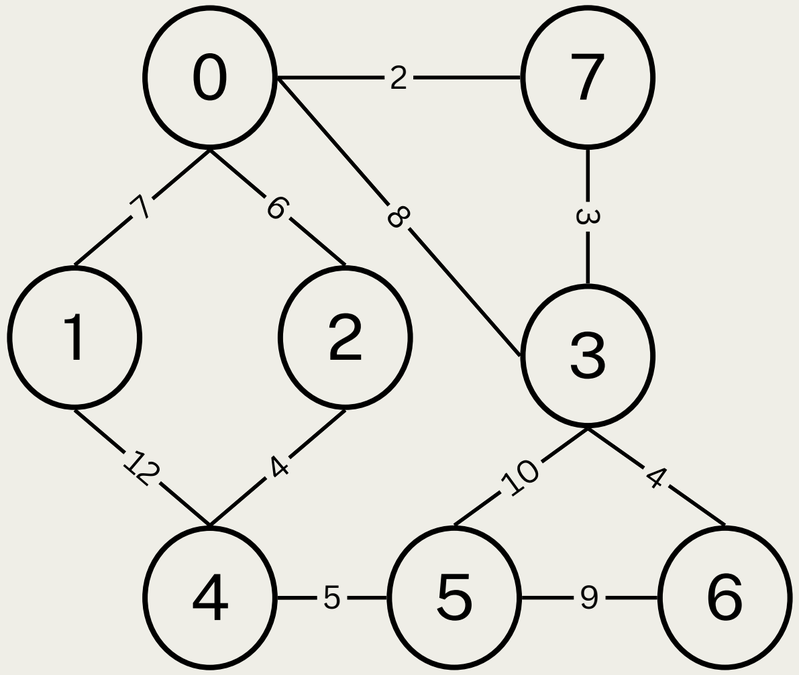
First, create a min priority queue with all of the edges. It is up to you to decide how you want to represent the edges in this p-queue. I will represent each edge with a tuple of (node1, node2, weight), note that this is an undirected graph.

p-queue:

[(0,7,2), (3,7,3), (2,4,4), (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

Let’s run Kruskal’s on this graph:

graph:



MST:

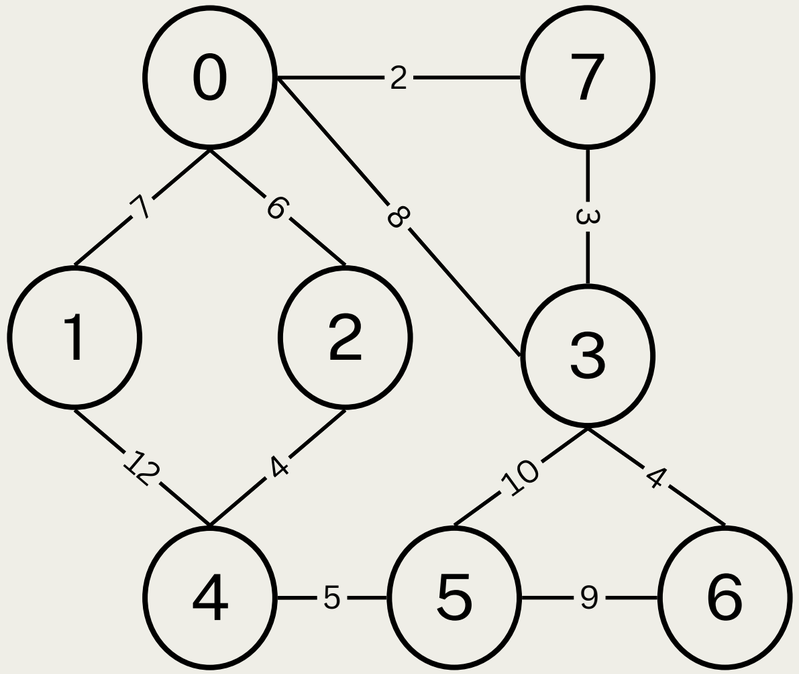
Now, we will remove the edges from the p-queue and decide whether they should be added to the MST or not (do they take us to a node we haven’t been to yet?)

p-queue:

[(0,7,2), (3,7,3), (2,4,4), (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

Let’s run Kruskal’s on this graph:

graph:



MST:

Now, we will remove the edges from the p-queue and decide whether they should be added to the MST or not (do they take us to a node we haven’t been to yet?)

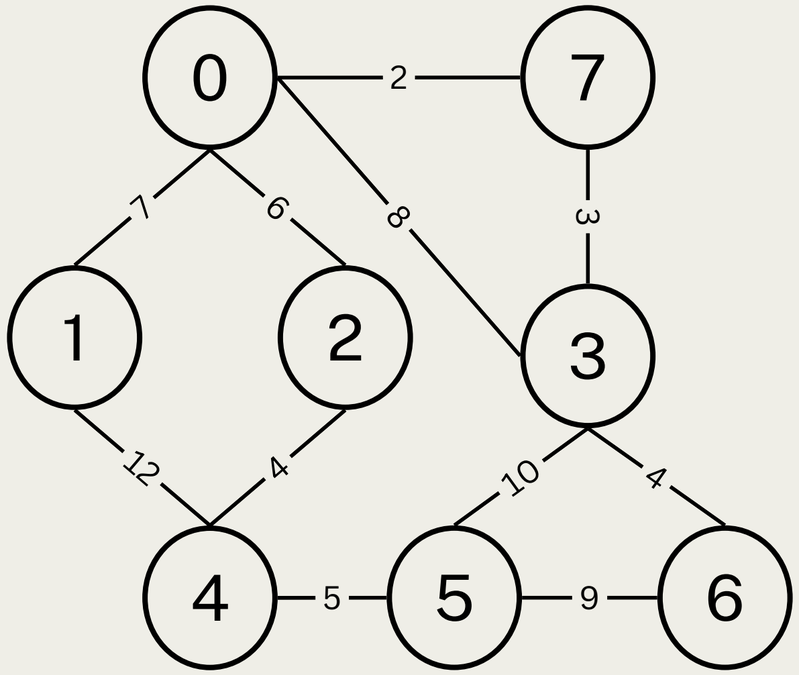
Notice how we will need to keep track of nodes we’ve visited in order to answer this question. I will keep track of nodes I have been to with a set.

p-queue:

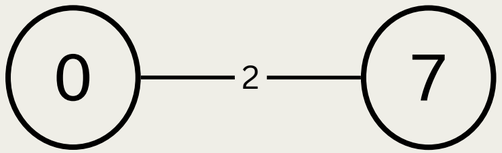
[(0,7,2), (3,7,3), (2,4,4), (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

Let’s run Kruskal’s on this graph:

graph:



MST:



We remove the first edge and we add it to the MST and to the set since we haven’t visited either 0 or 2 (we haven’t visited any nodes yet)
Now the difference is that we are going to have multiple sets!

p-queue:

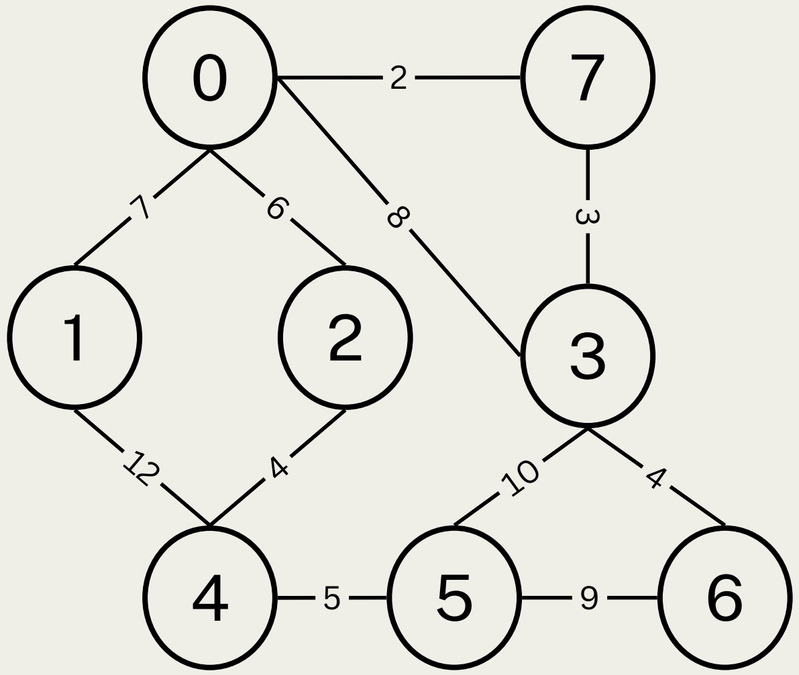
[~~(0,7,2)~~, (3,7,3), (2,4,4), (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

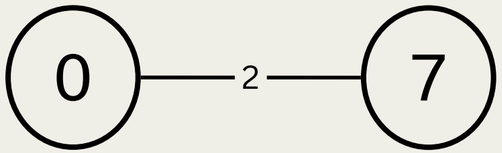
[(0,7,

Let’s run Kruskal’s on this graph:

graph:



MST:



Now, we take out the next edge and check whether adding this edge to the MST will help us connect to any new node. If yes, then add it to MST if not, then skip it.

p-queue:

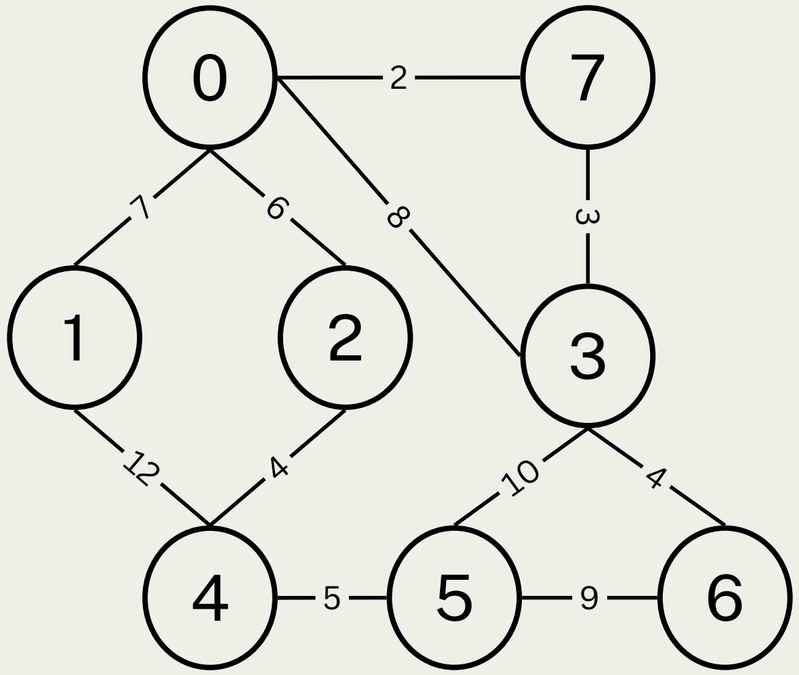
[~~(0,7,2)~~, (3,7,3), (2,4,4), (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

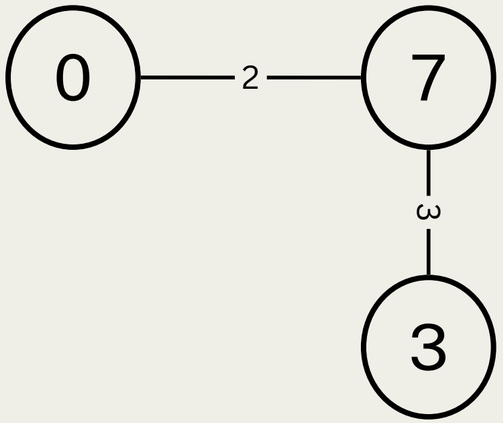
[(0,7,

Let’s run Kruskal’s on this graph:

graph:



MST:



When checking the edge (3,7,3), we note that we have already visited node 7 but we haven’t been to node 3 yet so we add this edge to the MST and node 3 to the visited set.

p-queue:

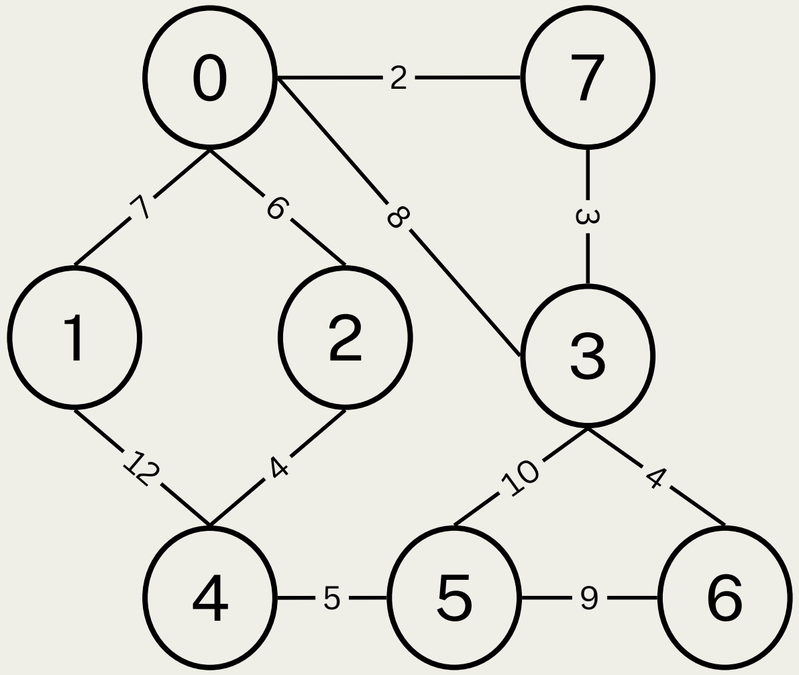
[~~(0,7,2)~~, ~~(3,7,3)~~, (2,4,4), (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

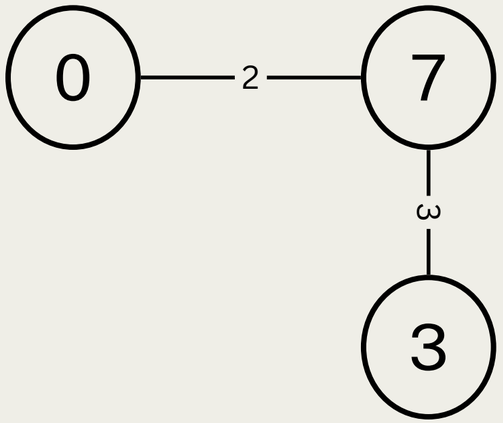
[(0, 7, 3

Let’s run Kruskal’s on this graph:

graph:



MST:



Now, onto the next edge. (2,4,4) we want to add this edge because we have not been to either 2 nor 4.

p-queue:

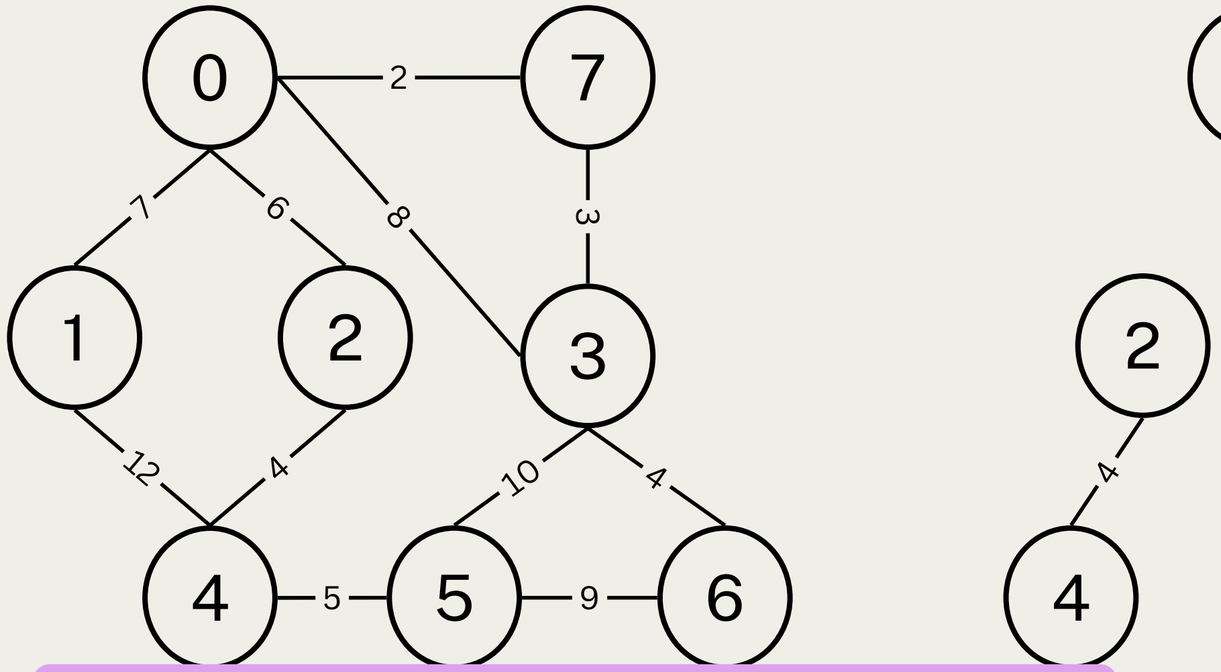
[~~(0,7,2)~~, ~~(3,7,3)~~, (2,4,4), (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

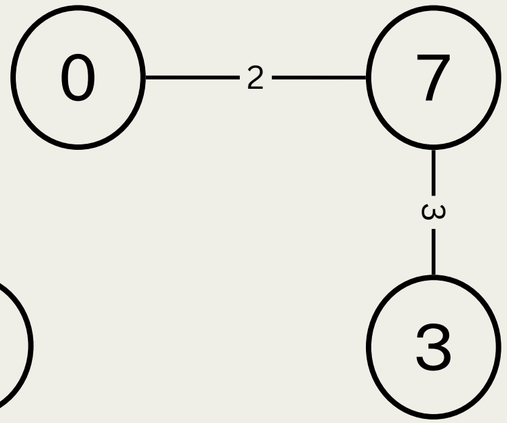
[(0, 7, 3

Let’s run Kruskal’s on this graph:

graph:



MST:



Now, onto the next edge. (2,4,4) we want to add this edge because we have not been to either 2 nor 4.

This is where the problem was before. Instead of adding 2 and 4 to the same set as our other visited nodes, we will add them to their own set to signify we have two “islands”

p-queue:

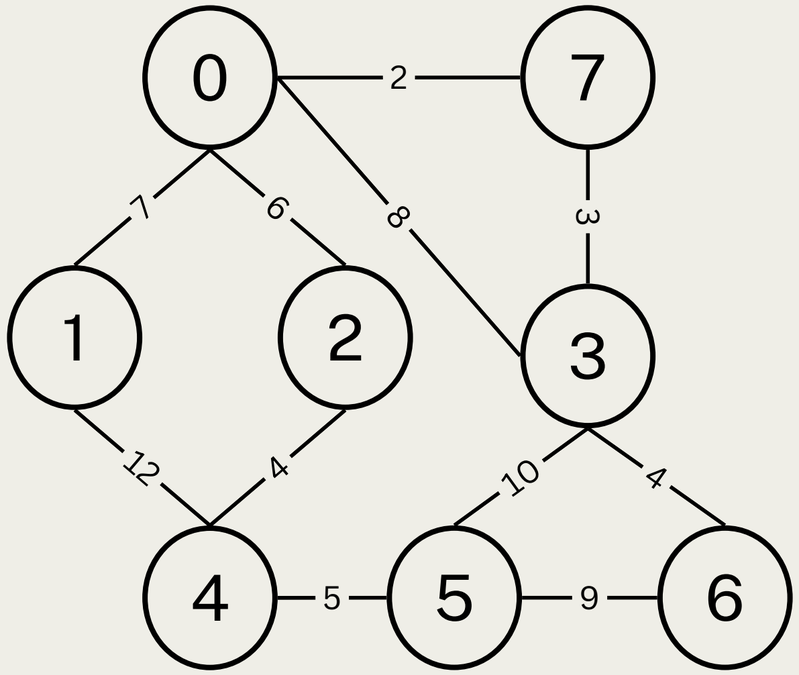
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

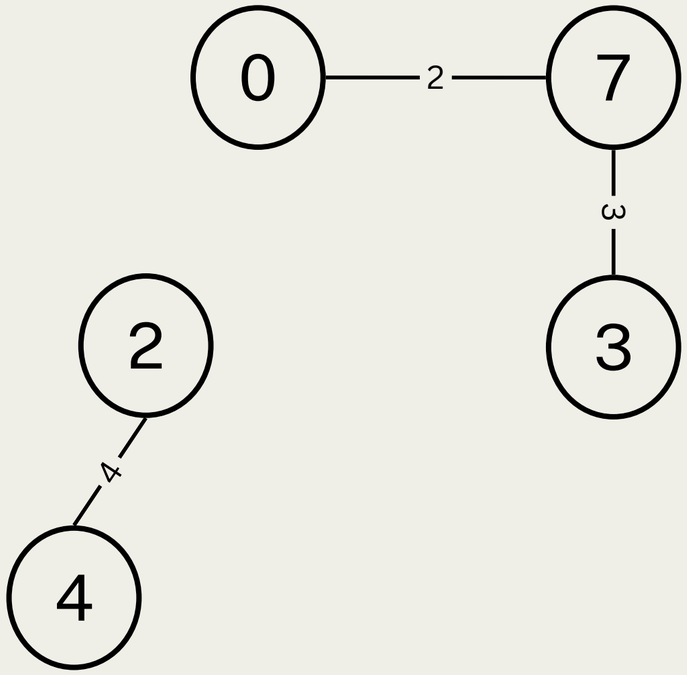
[(0, 7, 3,
(2, 4,

Let’s run Kruskal’s on this graph:

graph:



MST:



we continue the process...

p-queue:

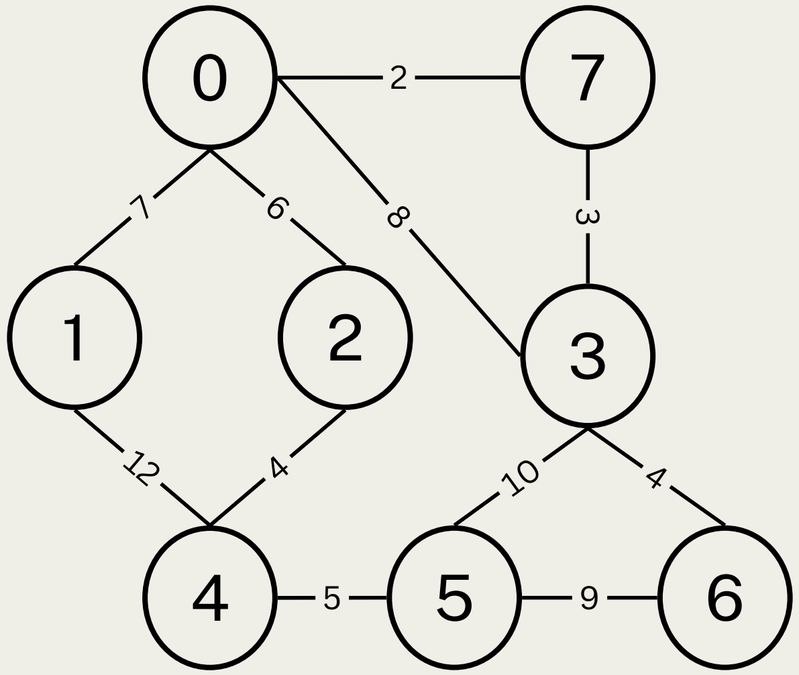
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, (3,6,4), (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

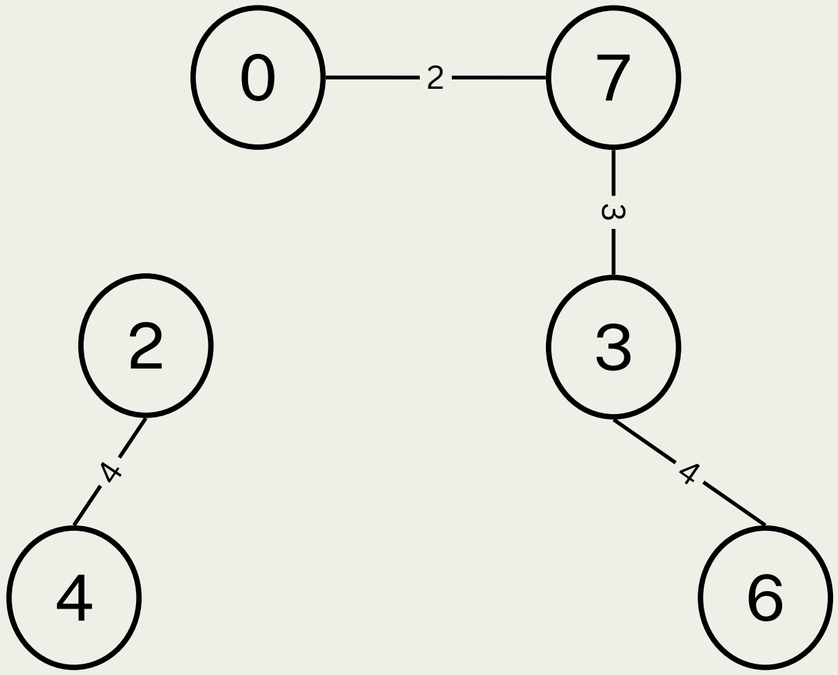
[(0, 7, 3,
(2, 4,

Let’s run Kruskal’s on this graph:

graph:



MST:



We check the next edge. (3,6,4) takes us to 6 so we add it.
We add node 6 to the set of visited nodes that is connected to 3.

p-queue:

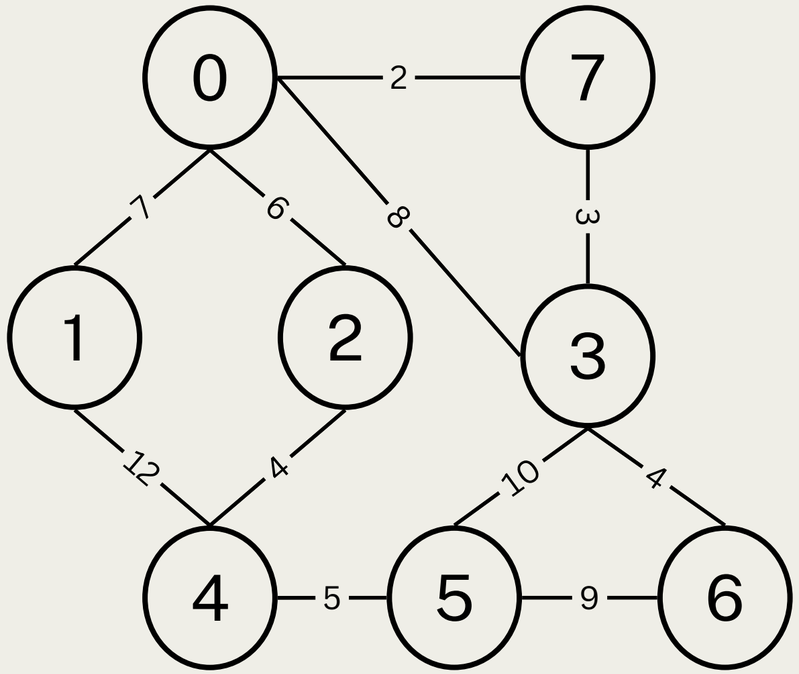
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, (4,5,5), (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

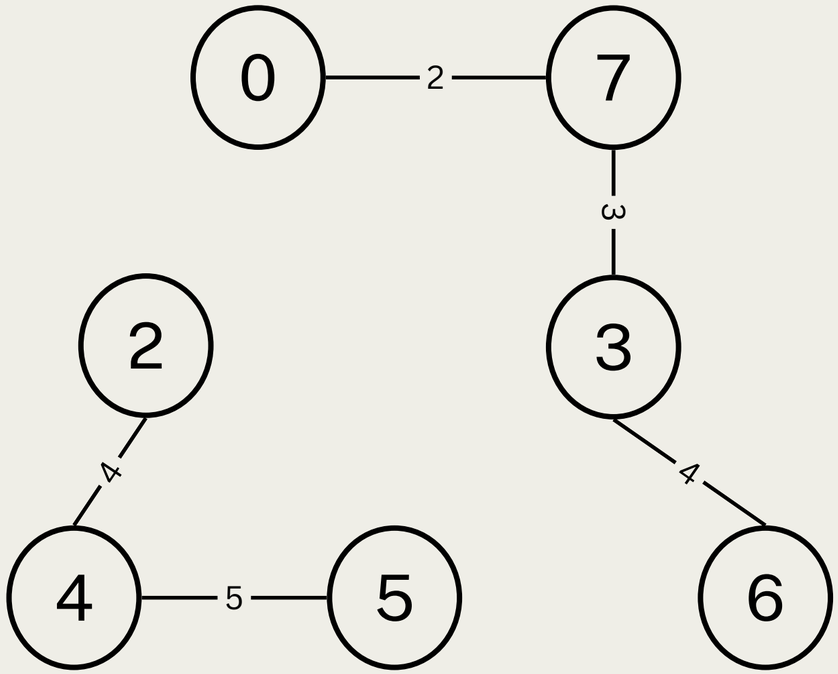
[(0, 7, 3, 6
(2, 4,

Let’s run Kruskal’s on this graph:

graph:



MST:



now the next edge, (4,5,5) takes us to 5 so we also add it. We add node 5 to the set that is connected to 4.

p-queue:

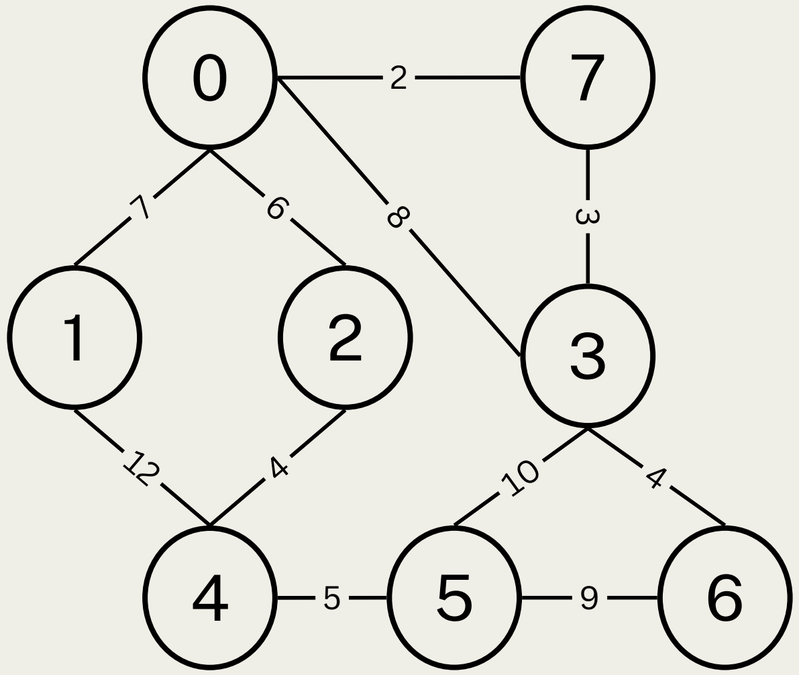
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, (0,2,6), (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

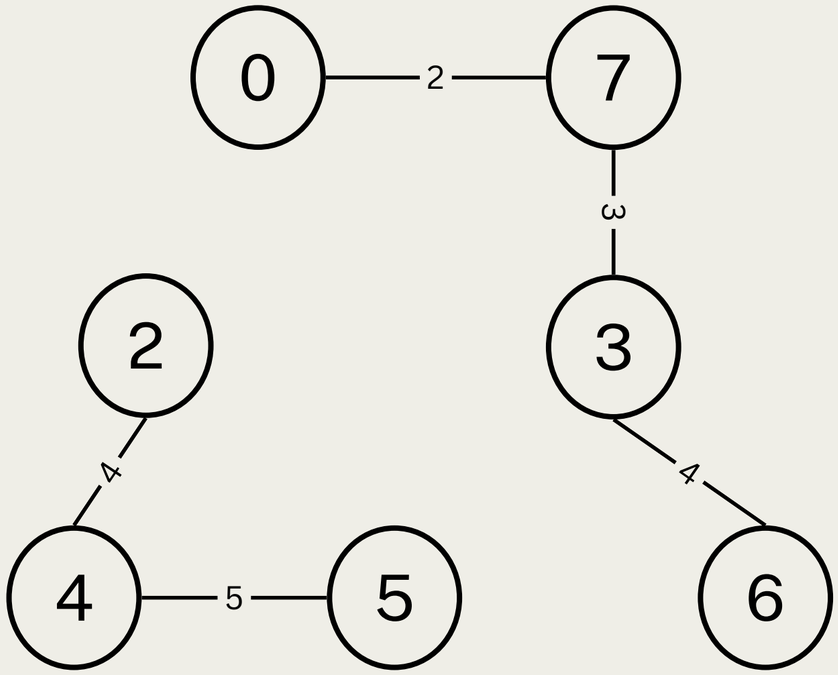
[(0, 7, 3, 6
(2, 4, 5

Let’s run Kruskal’s on this graph:

graph:



MST:



We now get to (0,2,6) Do we add it????
Well, in our visited we already hace node 0 and node 6.

p-queue:

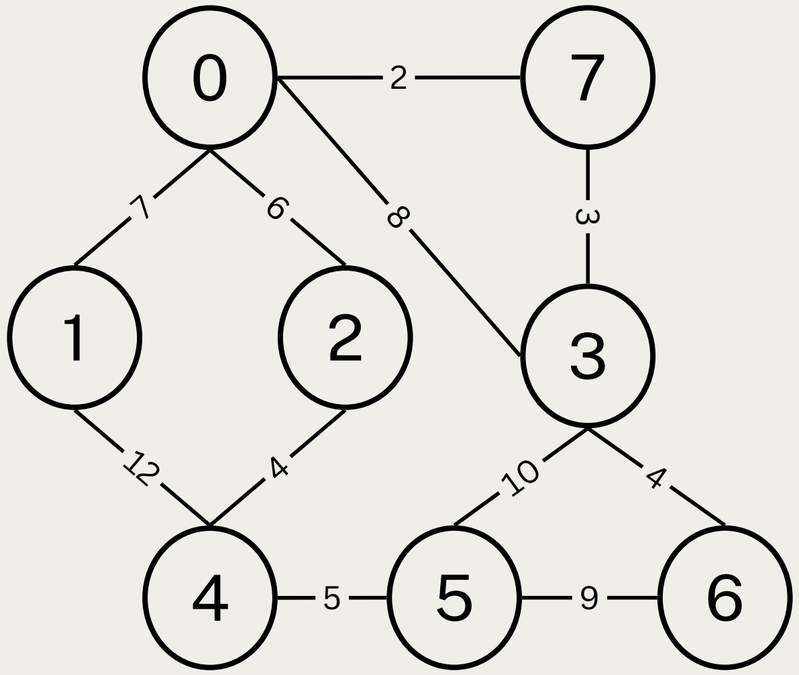
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~,
(0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

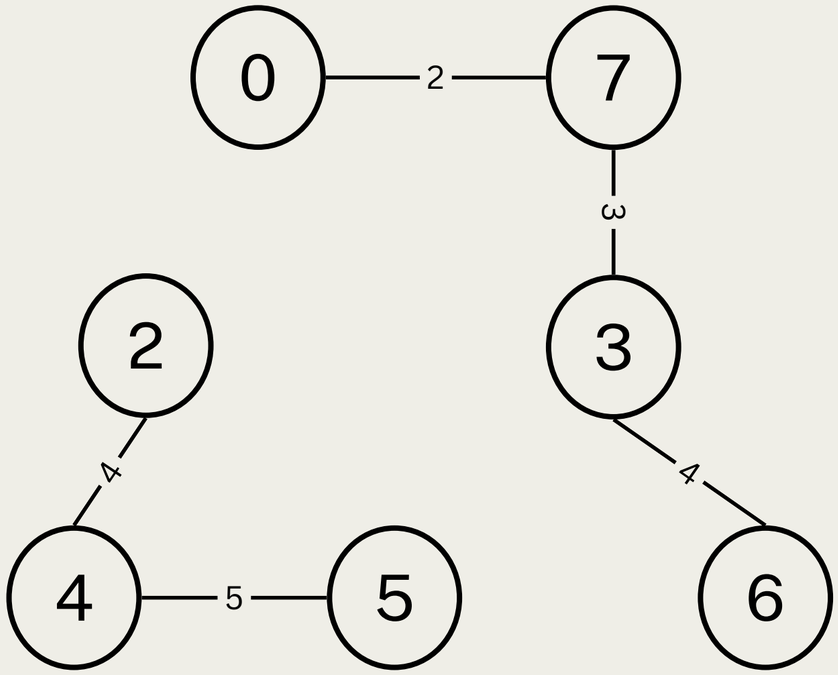
[(0, 7, 3, 6
(2, 4, 5

Let's run Kruskal's on this graph:

graph:



MST:



We now get to (0,2,6) Do we add it????
Well, in our visited we already have node 0 and node 2.
However, notice how 0 is in the first set and 2 is in the second.

p-queue:

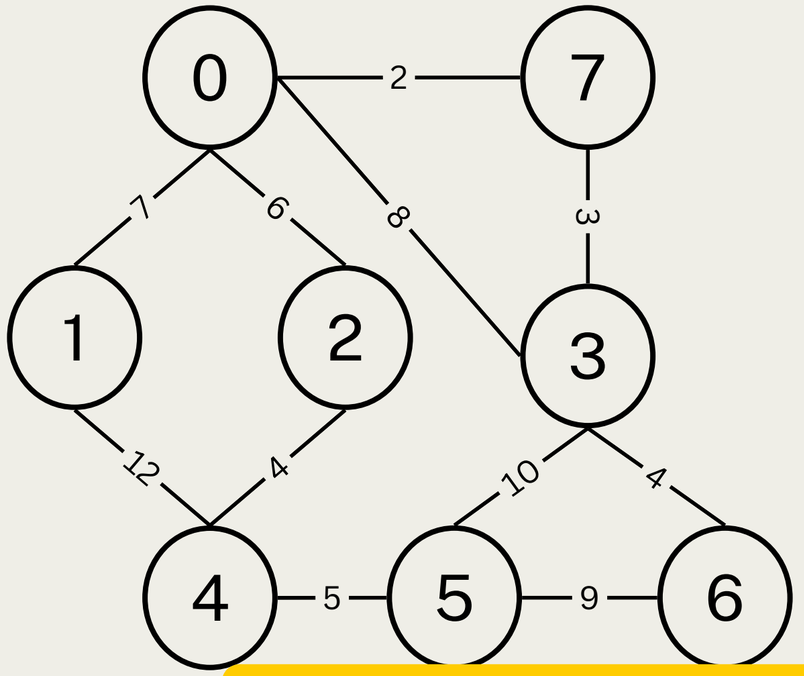
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~,
(0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

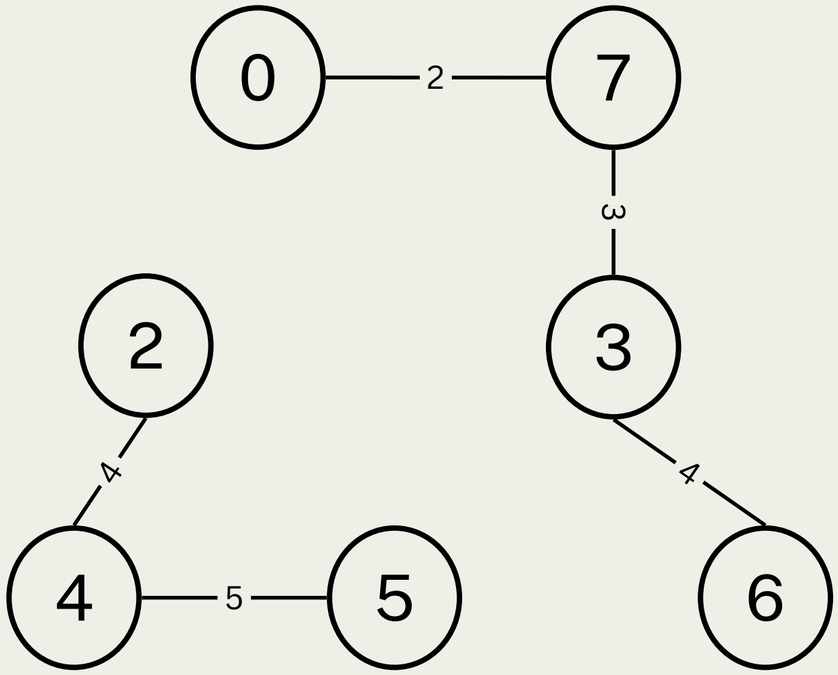
[(0, 7, 3, 6
(2, 4, 5

Let’s run Kruskal’s on this graph:

graph:



MST:



We now get to (0,2,6) Do we add it????
Well, in our visited we already have node 0 and node 2.
However, notice how 0 is in the first set and 2 is in the second.
This let’s us know that we SHOULD add this edge to the MST to ensure that we connect the two components/ islands.

p-queue:

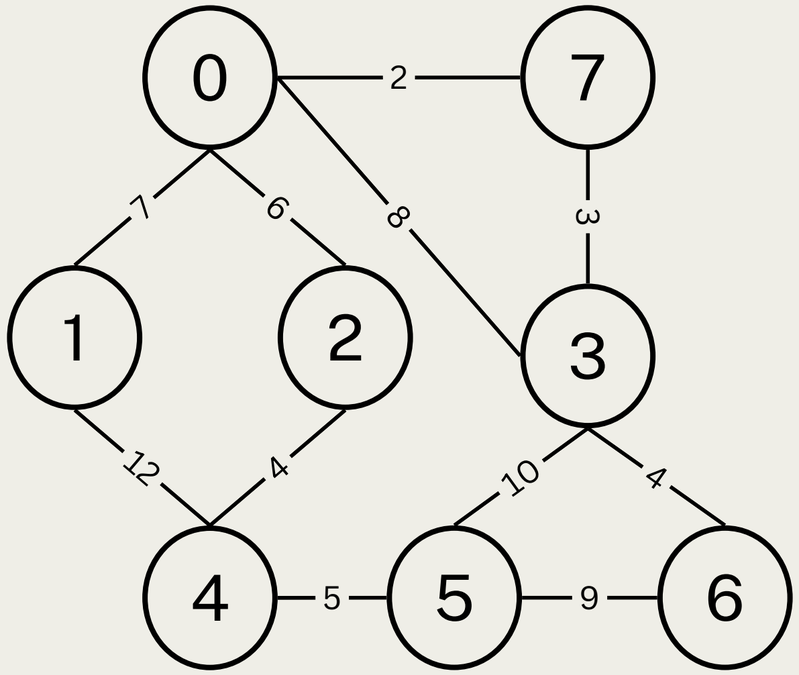
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~,
(0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

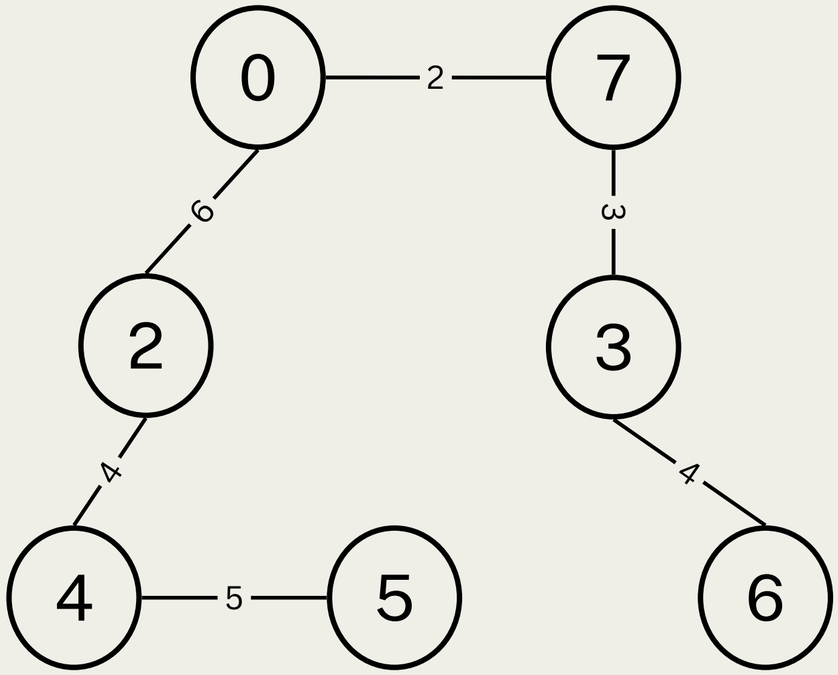
[(0, 7, 3, 6
(2, 4, 5

Let’s run Kruskal’s on this graph:

graph:



MST:



What do we do about our sets?

p-queue:

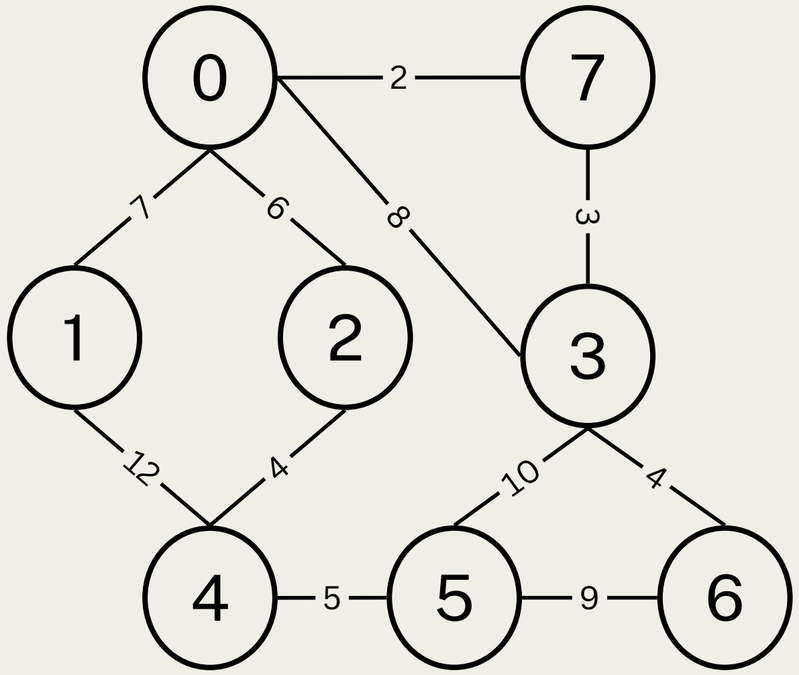
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~,
(0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

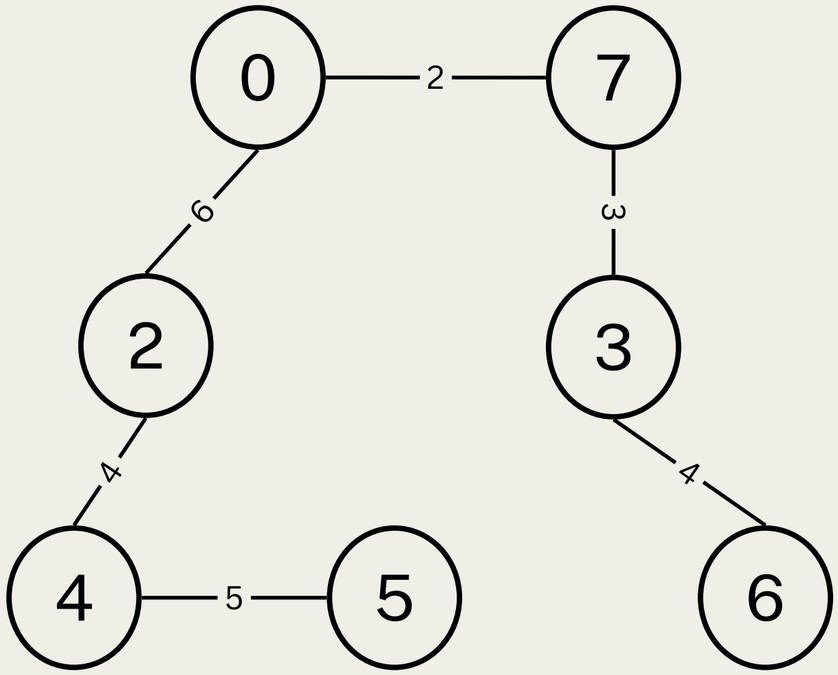
[(0, 7, 3, 6
(2, 4, 5

Let’s run Kruskal’s on this graph:

graph:



MST:



We combine them!!

p-queue:

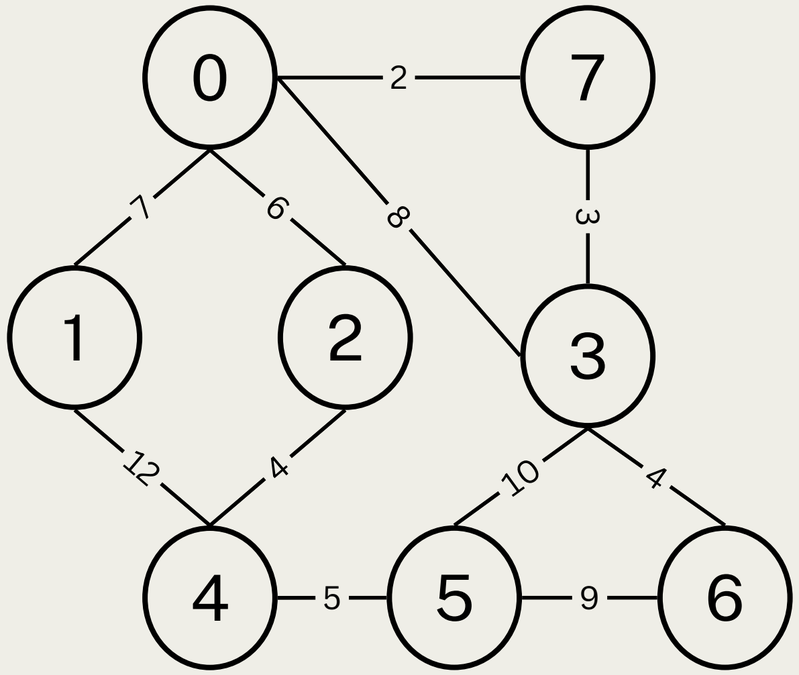
[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~, (0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

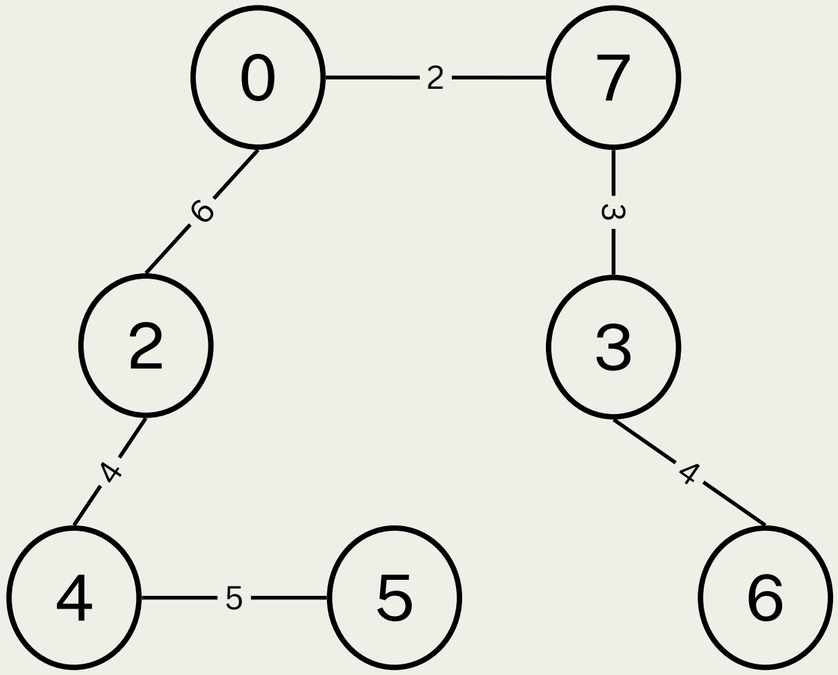
[(0, 7, 3, 6, 2, 4, 5

Let’s run Kruskal’s on this graph:

graph:



MST:



We can now continue the process in this way...

p-queue:

[~~(0,7,2)~~, ~~(3,7,3)~~, ~~(2,4,4)~~, ~~(3,6,4)~~, ~~(4,5,5)~~, ~~(0,2,6)~~,
(0,1,7), (0,3,8), (5,6,9), (3,5,10), (1,4,12)]

nodes visited:

[(0, 7, 3, 6, 2, 4, 5