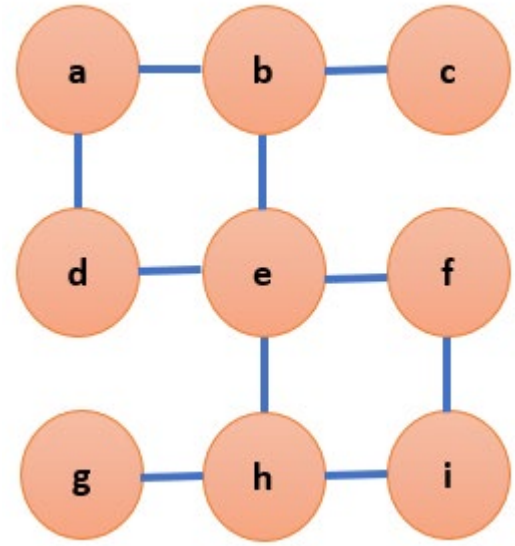
Coursework 1 for COMP0070 Algorithmics (2020)

Name: Lai Pong Chung

Student ID: 13039995

Q1

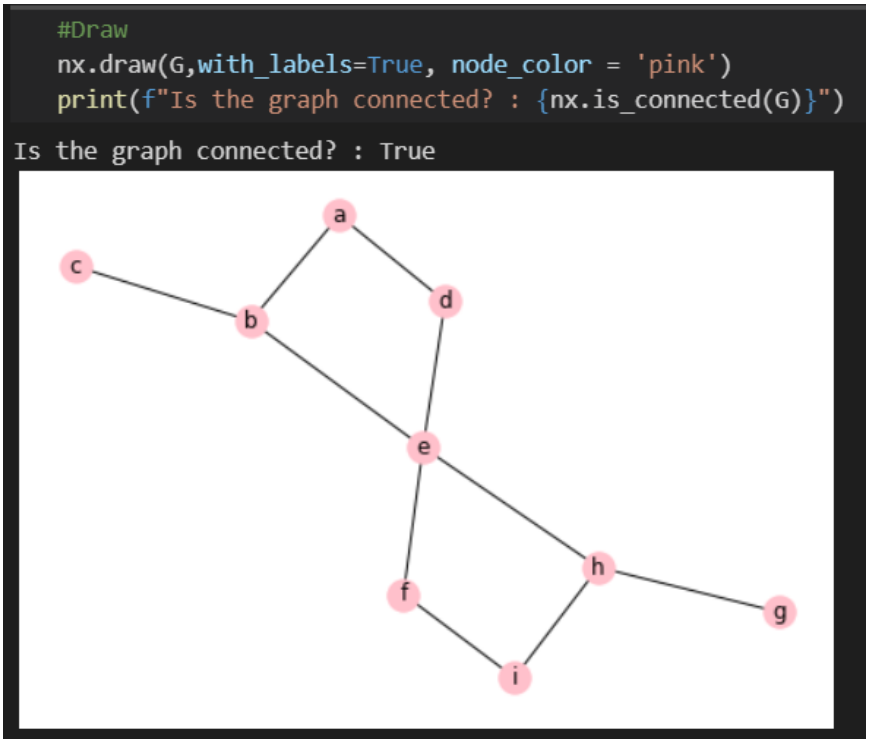
i) The nodes are represented by the orange orb and the edges are represented by the blue edges



iii) The graph is **connected** because there exists a path between all pairs of nodes (i.e. a path exists from any point to any other point in the graph).

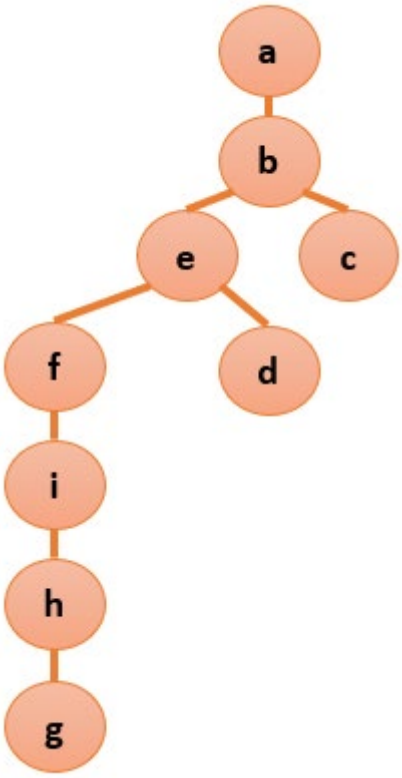
```
#Draw
nx.draw(G,with_labels=True, node_color = 'pink')
print(f"Is the graph connected? : {nx.is_connected(G)}")
```

```
Is the graph connected? : True
```



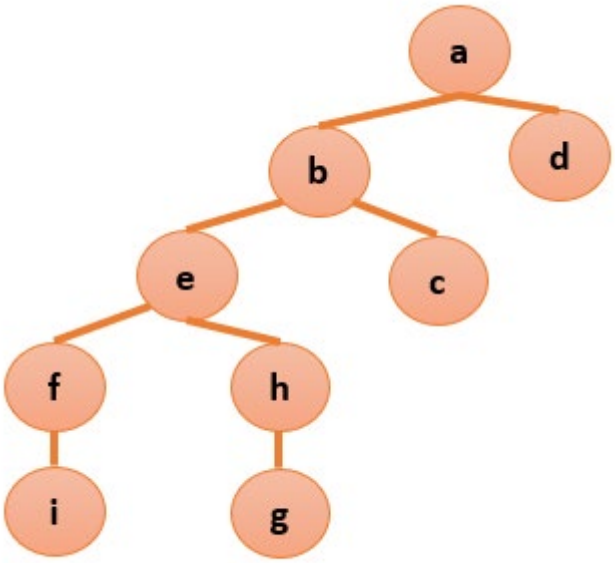
iv) a depth-first search tree starting at a

Operation	Stack
Push a	a
Push b	b,a
Push c	c,b,a
Pop c (c has no unvisited adjacent node)	b,a
Push e	e,b,a
Push f	f,e,b,a
Push i	i,f,e,b,a
Push h	h,i,f,e,b,a
Push g	g,h,i,f,e,b,a
Pop g	h,i,f,e,b,a
Pop h	i,f,e,b,a
Pop i	f,e,b,a
Pop f	e,b,a
Push d	d,e,b,a
Pop d	e,b,a
Pop e,	b,a
Pop b,	a
Pop a	empty



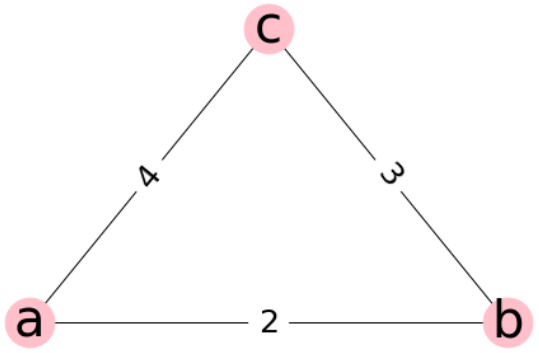
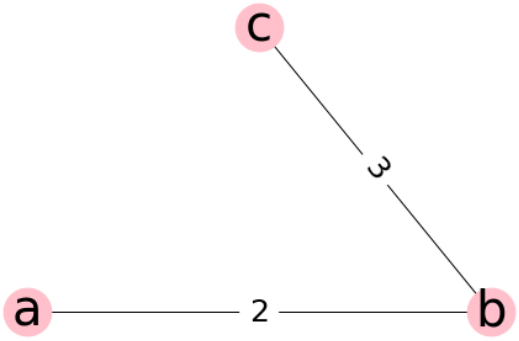
v) e a breadth-first search tree starting at a

Operation	Queue
Enqueue a	a
Dequeue a	Empty
Enqueue b	b
Enqueue d	b,d
Dequeue b	D
Enqueue c	d,c
Enqueue e	d,c,e
Dequeue d	c,e
Dequeue c	e,
Dequeue e	Empty
Enqueue f	f,
Enqueue h	f,h
Dequeue f	H
Enqueue i	h, i
Dequeue H	i
Enqueue g	i, g
dequeue i	g
Dequeue g	empty



Q2

When the Minimum spanning tree is built, for most algorithm the tree is constructed from choosing the edges from the graph with lowest cost. Thus, it will guarantee that the total cost of all the edges to build the minimum spanning tree is lowest. Sometimes, these edges are the shortest path for some of the nodes in the graph. As shown in the example below, (A, B) cost 2 and (B,C) cost 3 which is the minimum cost fir these pairs of nodes.

Original Graph	Minimum Spanning Tree
	

While the overall cost of traversing the minimum spanning tree is 5 but for nodes (A, C), the edge between them is removed. To travel from A to C, we need to take the long way around. Therefore, the cost is 5 (minimum spanning tree) instead of 4 (original graph). Thus, the minimum spanning tree only represent the minimum overall cost for traversing all nodes, but it does not guarantee a minimum cost for travelling between each pair of nodes.

Q3

The following part is finding a minimum tree using Floyd's algorithm. Basically, it will test all possible route between any 2 nodes in a directed graph node "i" and node "j" with node "k" in between. The algorithm will update the adjacency matrix when it discovers a path with lower cost.

The following is a step by step walkthrough of the Floyds algorithm. The adjacency matrix is represented in an array "D" where each element in the array represents the corresponds to element in adjacency matrix as shown below.

inf is infinity where no direct path exists between two nodes.

	a	b	c	d
a	0	1	∞	∞
b	∞	0	1	∞
c	∞	∞	0	1
d	1	∞	∞	0

```
[[ 0.  1. inf inf]
 [inf  0.  1. inf]
 [inf inf  0.  1.]
 [ 1.  2. inf  0.]]
```

Answer for Q3

	a	b	c	d
a	0	1	2	3
b	3	0	1	2
c	2	3	0	1
d	1	2	3	0

Q3 step by step calculation

```
*********starting floyd's algorithm on the adjacency matrix**************
k = a, i = a, j = a
Path Cost:
   original path (D[i][j]) from 'a' to 'a': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: 0.0
-----------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = a, i = a, j = b
Path Cost:
   original path (D[i][j]) from 'a' to 'b': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: 1.0
-----------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = a, i = a, j = c
Path Cost:
   original path (D[i][j]) from 'a' to 'c': inf
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: inf
-----------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
```

```
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = a, i = a, j = d
Path Cost:
   original path (D[i][j]) from 'a' to 'd': inf
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = a, i = b, j = a
Path Cost:
   original path (D[i][j]) from 'b' to 'a': inf
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = a, i = b, j = b
Path Cost:
   original path (D[i][j]) from 'b' to 'b': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = a, i = b, j = c
Path Cost:
   original path (D[i][j]) from 'b' to 'c': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = a, i = b, j = d
Path Cost:
   original path (D[i][j]) from 'b' to 'd': inf
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = a, i = c, j = a
Path Cost:
   original path (D[i][j]) from 'c' to 'a': inf
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = a, i = c, j = b
Path Cost:
   original path (D[i][j]) from 'c' to 'b': inf
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
```

```
k = a, i = c, j = c
Path Cost:
   original path (D[i][j]) from 'c' to 'c': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = a, i = c, j = d
Path Cost:
   original path (D[i][j]) from 'c' to 'd': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = a, i = d, j = a
Path Cost:
   original path (D[i][j]) from 'd' to 'a': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: 1.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = a, i = d, j = b
Path Cost:
   original path (D[i][j]) from 'd' to 'b': inf
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: 2.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition met, updating matrix
[[ 0.  1. inf inf]
 [inf  0.  1. inf]
 [inf inf  0.  1.]
 [ 1.  2. inf  0.]]
======================Next Step=========================
k = a, i = d, j = c
Path Cost:
   original path (D[i][j]) from 'd' to 'c': inf
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = a, i = d, j = d
Path Cost:
   original path (D[i][j]) from 'd' to 'd': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'a' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = b, i = a, j = a
Path Cost:
   original path (D[i][j]) from 'a' to 'a': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
```

```
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = b, i = a, j = b
Path Cost:
   original path (D[i][j]) from 'a' to 'b': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: 1.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = b, i = a, j = c
Path Cost:
   original path (D[i][j]) from 'a' to 'c': inf
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: 2.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition met, updating matrix
[[ 0.  1.  2. inf]
 [inf  0.  1. inf]
 [inf inf  0.  1.]
 [ 1.  2. inf  0.]]
=======================Next Step==========================
k = b, i = a, j = d
Path Cost:
   original path (D[i][j]) from 'a' to 'd': inf
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = b, i = b, j = a
Path Cost:
   original path (D[i][j]) from 'b' to 'a': inf
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = b, i = b, j = b
Path Cost:
   original path (D[i][j]) from 'b' to 'b': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: 0.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = b, i = b, j = c
Path Cost:
   original path (D[i][j]) from 'b' to 'c': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: 1.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = b, i = b, j = d
Path Cost:
   original path (D[i][j]) from 'b' to 'd': inf
```

```
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = b, i = c, j = a
Path Cost:
   original path (D[i][j]) from 'c' to 'a': inf
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = b, i = c, j = b
Path Cost:
   original path (D[i][j]) from 'c' to 'b': inf
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = b, i = c, j = c
Path Cost:
   original path (D[i][j]) from 'c' to 'c': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = b, i = c, j = d
Path Cost:
   original path (D[i][j]) from 'c' to 'd': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = b, i = d, j = a
Path Cost:
   original path (D[i][j]) from 'd' to 'a': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = b, i = d, j = b
Path Cost:
   original path (D[i][j]) from 'd' to 'b': 2.0
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: 2.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = b, i = d, j = c
Path Cost:
   original path (D[i][j]) from 'd' to 'c': inf
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: 3.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
```

```
Condition met, updating matrix
[[ 0.  1.  2. inf]
 [inf  0.  1. inf]
 [inf inf  0.  1.]
 [ 1.  2.  3.  0.]]
=======================Next Step==========================
k = b, i = d, j = d
Path Cost:
   original path (D[i][j]) from 'd' to 'd': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'b' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = c, i = a, j = a
Path Cost:
   original path (D[i][j]) from 'a' to 'a': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = c, i = a, j = b
Path Cost:
   original path (D[i][j]) from 'a' to 'b': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = c, i = a, j = c
Path Cost:
   original path (D[i][j]) from 'a' to 'c': 2.0
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: 2.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
=======================Next Step==========================
k = c, i = a, j = d
Path Cost:
   original path (D[i][j]) from 'a' to 'd': inf
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: 3.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition met, updating matrix
[[ 0.  1.  2.  3.]
 [inf  0.  1. inf]
 [inf inf  0.  1.]
 [ 1.  2.  3.  0.]]
=======================Next Step==========================
k = c, i = b, j = a
Path Cost:
   original path (D[i][j]) from 'b' to 'a': inf
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
```

```
======================Next Step==========================
k = c, i = b, j = b
Path Cost:
   original path (D[i][j]) from 'b' to 'b': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = c, i = b, j = c
Path Cost:
   original path (D[i][j]) from 'b' to 'c': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: 1.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = c, i = b, j = d
Path Cost:
   original path (D[i][j]) from 'b' to 'd': inf
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: 2.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition met, updating matrix
[[ 0.  1.  2.  3.]
 [inf  0.  1.  2.]
 [inf inf  0.  1.]
 [ 1.  2.  3.  0.]]
======================Next Step==========================
k = c, i = c, j = a
Path Cost:
   original path (D[i][j]) from 'c' to 'a': inf
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = c, i = c, j = b
Path Cost:
   original path (D[i][j]) from 'c' to 'b': inf
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = c, i = c, j = c
Path Cost:
   original path (D[i][j]) from 'c' to 'c': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: 0.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = c, i = c, j = d
Path Cost:
   original path (D[i][j]) from 'c' to 'd': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: 1.0
```

```
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = c, i = d, j = a
Path Cost:
   original path (D[i][j]) from 'd' to 'a': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = c, i = d, j = b
Path Cost:
   original path (D[i][j]) from 'd' to 'b': 2.0
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: inf
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = c, i = d, j = c
Path Cost:
   original path (D[i][j]) from 'd' to 'c': 3.0
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: 3.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = c, i = d, j = d
Path Cost:
   original path (D[i][j]) from 'd' to 'd': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'c' in between: 4.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = d, i = a, j = a
Path Cost:
   original path (D[i][j]) from 'a' to 'a': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 4.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = d, i = a, j = b
Path Cost:
   original path (D[i][j]) from 'a' to 'b': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 5.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = d, i = a, j = c
Path Cost:
   original path (D[i][j]) from 'a' to 'c': 2.0
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 6.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
```

```
======================Next Step=========================
k = d, i = a, j = d
Path Cost:
   original path (D[i][j]) from 'a' to 'd': 3.0
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 3.0
-------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = d, i = b, j = a
Path Cost:
   original path (D[i][j]) from 'b' to 'a': inf
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 3.0
-------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition met, updating matrix
[[ 0.  1.  2.  3.]
 [ 3.  0.  1.  2.]
 [inf inf  0.  1.]
 [ 1.  2.  3.  0.]]
======================Next Step=========================
k = d, i = b, j = b
Path Cost:
   original path (D[i][j]) from 'b' to 'b': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 4.0
-------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = d, i = b, j = c
Path Cost:
   original path (D[i][j]) from 'b' to 'c': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 5.0
-------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = d, i = b, j = d
Path Cost:
   original path (D[i][j]) from 'b' to 'd': 2.0
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 2.0
-------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
k = d, i = c, j = a
Path Cost:
   original path (D[i][j]) from 'c' to 'a': inf
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 2.0
-------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition met, updating matrix
[[ 0.  1.  2.  3.]
 [ 3.  0.  1.  2.]
 [ 2. inf  0.  1.]
 [ 1.  2.  3.  0.]]
======================Next Step=========================
```

```
k = d, i = c, j = b
Path Cost:
   original path (D[i][j]) from 'c' to 'b': inf
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 3.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition met, updating matrix
[[0. 1. 2. 3.]
 [3. 0. 1. 2.]
 [2. 3. 0. 1.]
 [1. 2. 3. 0.]]
======================Next Step==========================
k = d, i = c, j = c
Path Cost:
   original path (D[i][j]) from 'c' to 'c': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 4.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = d, i = c, j = d
Path Cost:
   original path (D[i][j]) from 'c' to 'd': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 1.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = d, i = d, j = a
Path Cost:
   original path (D[i][j]) from 'd' to 'a': 1.0
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 1.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = d, i = d, j = b
Path Cost:
   original path (D[i][j]) from 'd' to 'b': 2.0
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 2.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = d, i = d, j = c
Path Cost:
   original path (D[i][j]) from 'd' to 'c': 3.0
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 3.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
Condition failed, the new route doesn't cost less than the original route.
======================Next Step==========================
k = d, i = d, j = d
Path Cost:
   original path (D[i][j]) from 'd' to 'd': 0.0
   new path (D[i][k] + D[k][j]) with nodes 'd' in between: 0.0
------------------------Checking Condition: D[i][j] > D[i][k] + D[k][j]-------------------------
```

```
Condition failed, the new route doesn't cost less than the original route.
======================Next Step=========================
final
[[0. 1. 2. 3.]
 [3. 0. 1. 2.]
 [2. 3. 0. 1.]
 [1. 2. 3. 0.]]
****************************End**************************************
```

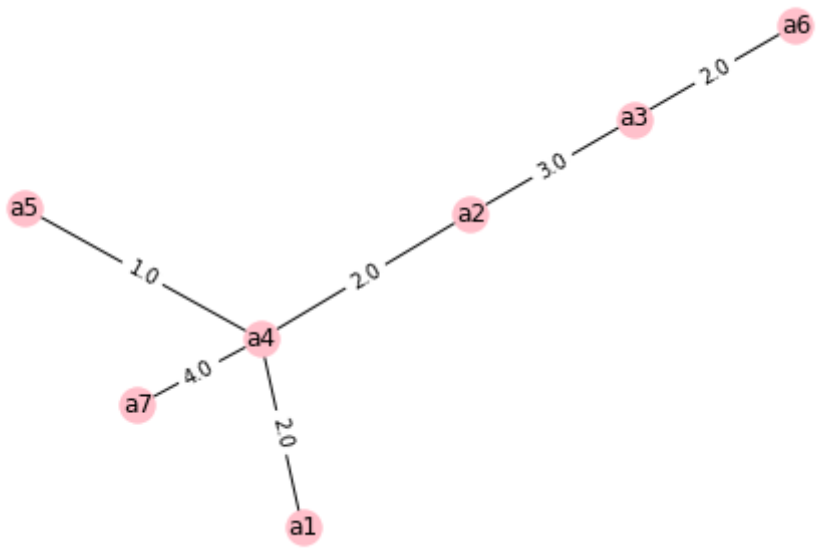
Q4

The prims algorithm functions by starting from any node in a graph. Second, a Mindist array is populated by the cost of travelling to nodes with a directly connected to the starting node with an edge while the Nearest array record the nearest node to get to the corresponding node from the current graph. Initially, it will be populated by the starting node. After that, the algorithm adds the node which cost the least to get to i.e. the minimum in the Mindist Array. After that, the Mindist and Nearest array is updated using the new information.

Arbitrarily, I started Prim's algorithm starting from node "a1"

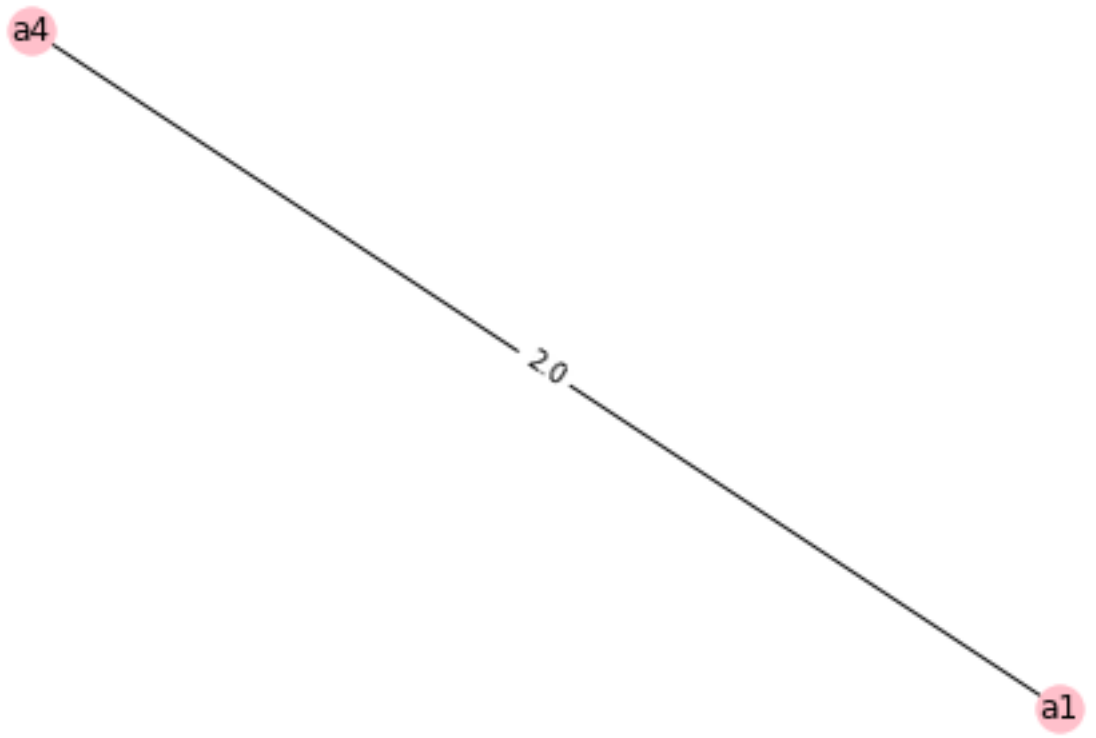
The value of the MinDist Array is the cost of adding new nodes to the graph. The value is extracted from the graph provided where the cost is based on the wight of the edges and the value corresponds to the cost of travelling to nodes [a1, a2, a3, a4, a5, a6, a7] respectively with -1 indicating that the node is already added to the minimum spanning tree and inf indicating that no direct path exist in between and the Nearest Array is populated be the label of the nodes as described in the graph provided.

Answer for Q4

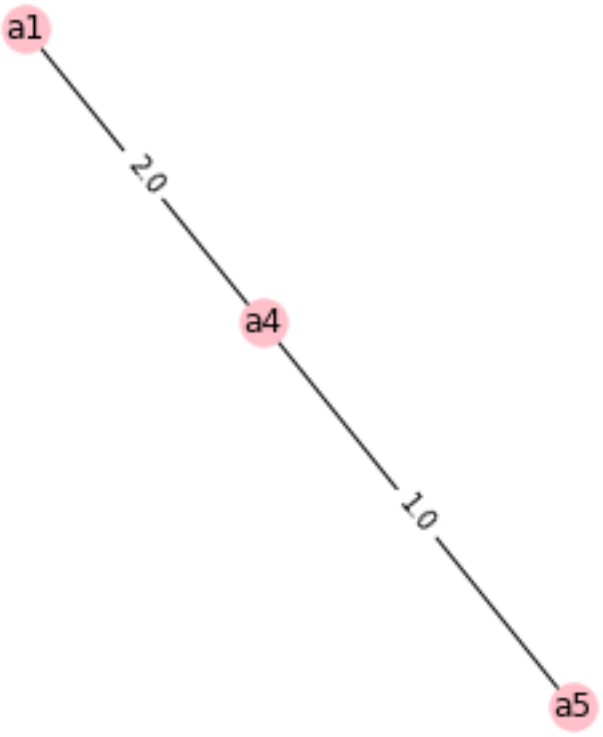


Step By Step Calculation

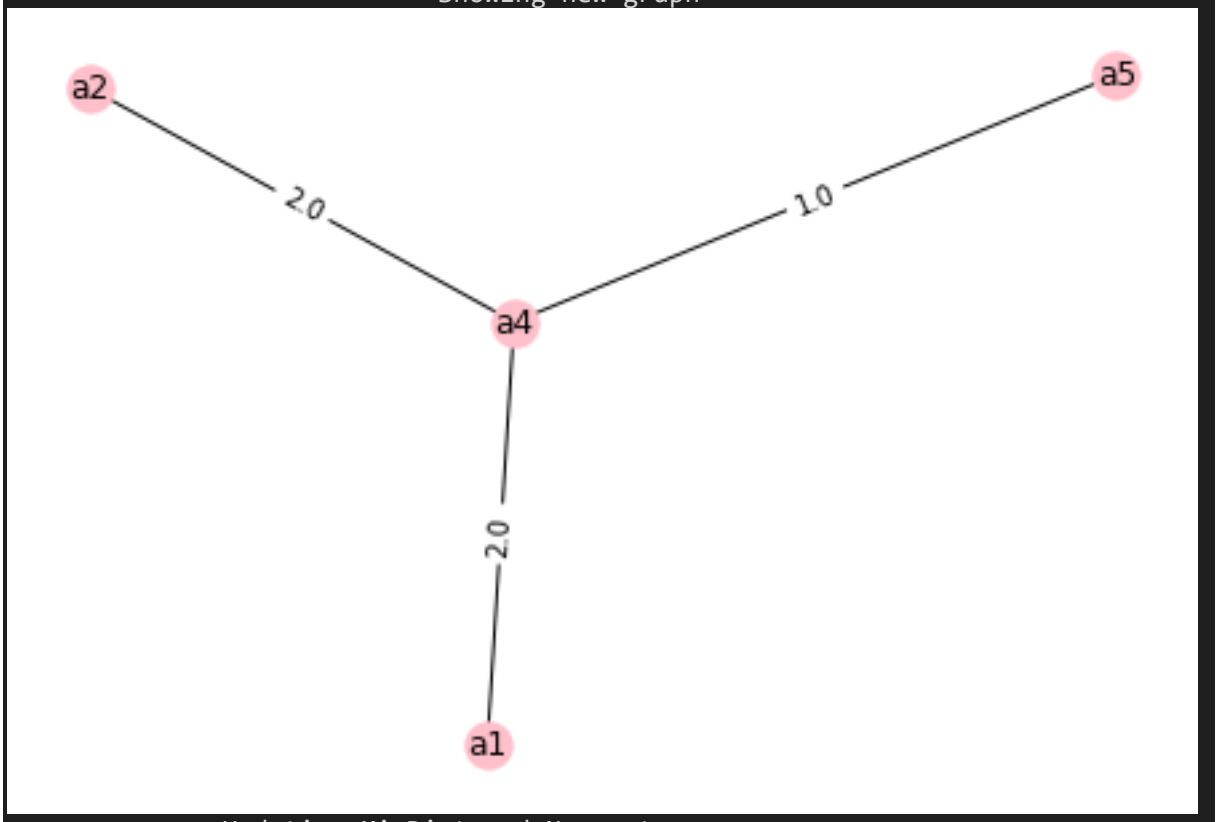
```
****************starting Prim's algorithm**********************
============Showing initial MinDist and Nearest Array============
MinDist Array: [-1. 8. inf 2. inf inf inf]
Nearest Array: ['a1', 'a1', 'a1', 'a1', 'a1', 'a1', 'a1']
------------------Finding Minimum in MinDist Array-----------------------
the Minimum is: 2.0 on node 'a4' from node 'a1', adding edge...
----------------------------Showing new graph----------------------------
```



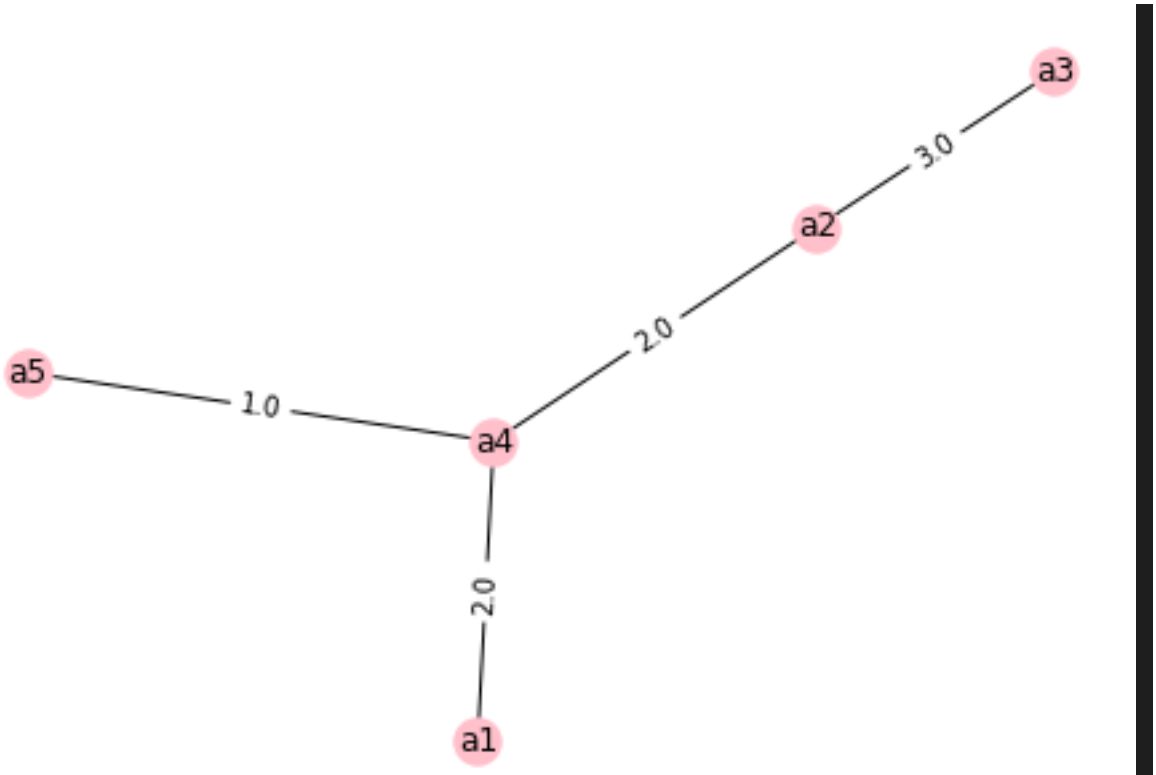

```
--------------Updating MinDist and Nearest-------------------------
MinDist Array: [-1. 2. inf -1. 1. inf 4.]
Nearest Array: ['a1', 'a4', 'a1', 'a1', 'a4', 'a1', 'a4']
======================Next Cycle==================================
------------------Finding Minimum in MinDist Array-----------------------
the Minimum is: 1.0 on node 'a5' from node 'a4', adding edge...
----------------------------Showing new graph----------------------------
```



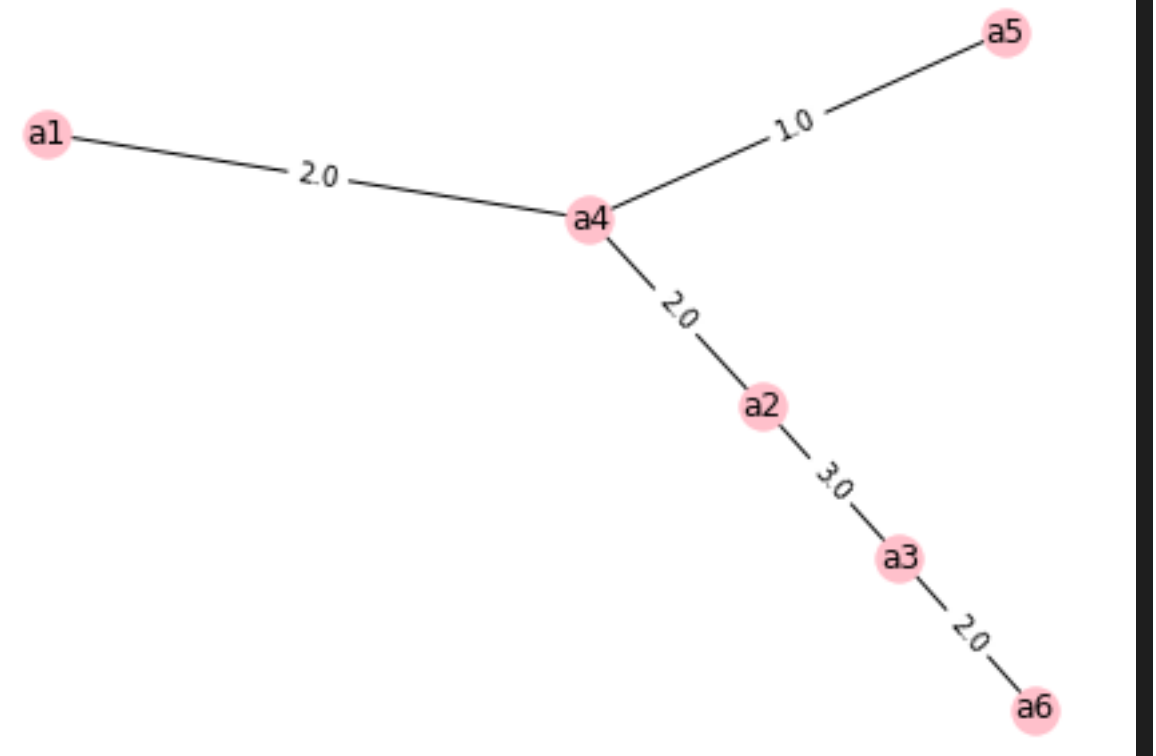
```
-------------Updating MinDist and Nearest-----------------------
MinDist Array: [-1. 2. 9. -1. -1. 7. 4.]
Nearest Array: ['a1', 'a4', 'a5', 'a1', 'a4', 'a5', 'a4']
======================Next Cycle==================================
-----------------Finding Minimum in MinDist Array-----------------------
the Minimum is: 2.0 on node 'a2' from node 'a4', adding edge...
---------------------------Showing new graph----------------------------
```



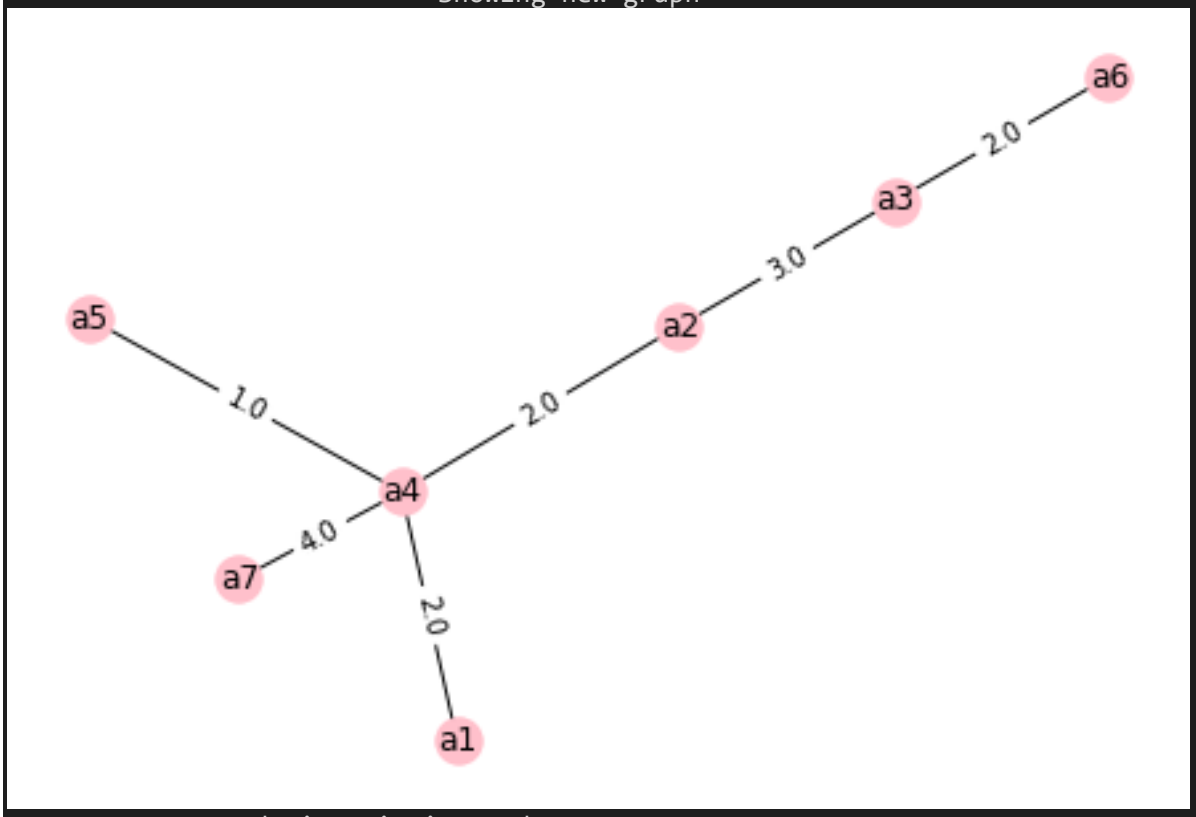

```
-------------Updating MinDist and Nearest-----------------------
MinDist Array: [-1. -1. 3. -1. -1. 7. 4.]
Nearest Array: ['a1', 'a4', 'a2', 'a1', 'a4', 'a5', 'a4']
======================Next Cycle==================================
-----------------Finding Minimum in MinDist Array-----------------------
the Minimum is: 3.0 on node 'a3' from node 'a2', adding edge...
---------------------------Showing new graph----------------------------
```




```
--------------Updating MinDist and Nearest-------------------------
MinDist Array: [-1. -1. -1. -1. -1. 2. 4.]
Nearest Array: ['a1', 'a4', 'a2', 'a1', 'a4', 'a3', 'a4']
======================Next Cycle=====================================
------------------Finding Minimum in MinDist Array-----------------------
the Minimum is: 2.0 on node 'a6' from node 'a3', adding edge...
----------------------------Showing new graph-----------------------------
```



```
-------------Updating MinDist and Nearest-----------------------
MinDist Array: [-1. -1. -1. -1. -1. -1. 4.]
Nearest Array: ['a1', 'a4', 'a2', 'a1', 'a4', 'a3', 'a4']
======================Next Cycle=====================================
-----------------Finding Minimum in MinDist Array-----------------------
the Minimum is: 4.0 on node 'a7' from node 'a4', adding edge...
---------------------------Showing new graph----------------------------
```




```
-------------Updating MinDist and Nearest-----------------------
MinDist Array: [-1. -1. -1. -1. -1. -1. -1.]
Nearest Array: ['a1', 'a4', 'a2', 'a1', 'a4', 'a3', 'a4']
======================Next Cycle=====================================
******************Finish********************
```