Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual
Obscure Binary Search Trees [12.5%]

One alternative data structure that can be used is the ‘Skip List’. A skip list is a data structure that is similar to a linked list. It is used so that we are able to quickly search a particular element in the structure. It consists of a base list that contains all elements which are also connected with layers of lists that connect to one another, in which each layer maintains a linked hierarchy of subsequences, each layer skipping over fewer elements of the list, starting from the bottom to the topmost layer. When implemented correctly, it performs better than other data structures such as red black trees when we want to have rapid insertions as there is no need for balancing, meaning no rotations or reallocations needed.

Knight’s Travails [12.5%]

To know the simplest way the knight can move to its destination, we can use BFS which uses the Queue data structure to search the minimum amount of steps that can be taken for the knight to reach the destination as this problem is using an undirected graph where the weight to each movement taken is equal which is 1.

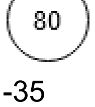
How it works is that first, we have to declare the chessboard, the position of both the knight and its destination, and the different moves the knight can make. A knight can move in a total of 8 different ways as shown in the picture. The total possible moves a knight can make depends on its location as it cannot go out of the bounds of the chessboard.

After that, we may implement BFS by first inputting the starting location of the knight to the queue and calculate the amount of locations it can move to at that time. The starting point will be now marked as visited and a ‘cost’ will be assigned to it to know the amount of steps taken to get to that point (in this case it is 0 since it is the starting point), and the different locations the horse could move to will be now moved to the queue while the starting point will be removed from the queue. The ‘cost’ of a location can be calculated by taking the ‘cost’ of the previous location and increasing it by one. The algorithm continues by doing the exact same thing to the positions in the queue and it will only stop when the queue is empty or in this case, we can stop it if the destination is already reached, and if that is the case then the cost to reach the destination will be displayed. If a location is already visited, it won’t be added to the queue again and will be printed instead to know where the knight is.

Tree Simulation [16%]

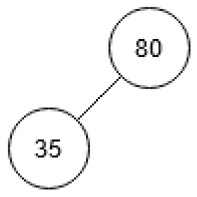
AVL Tree Simulation:
a.)
Insert:

-80

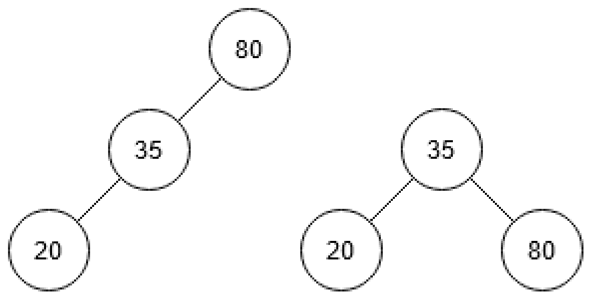


-35

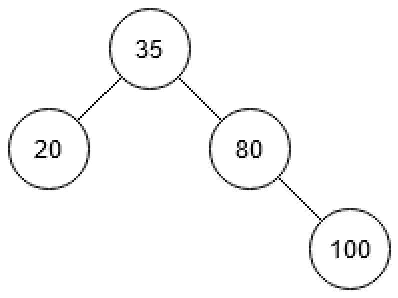
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual



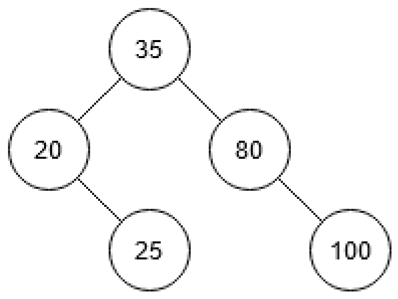
-20



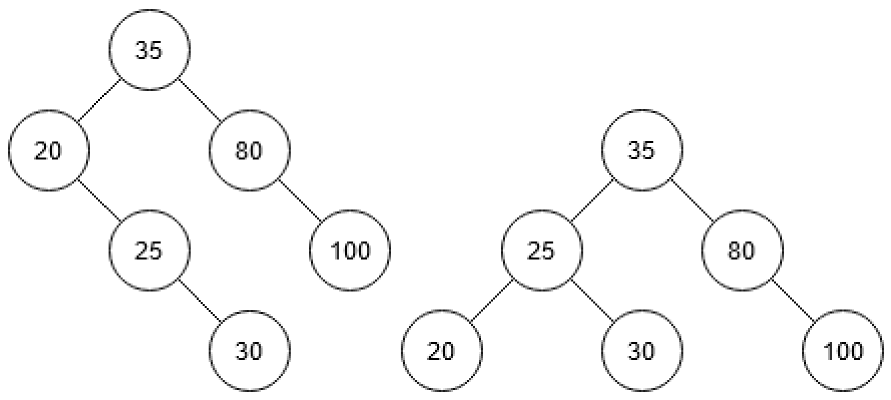
-100



-25

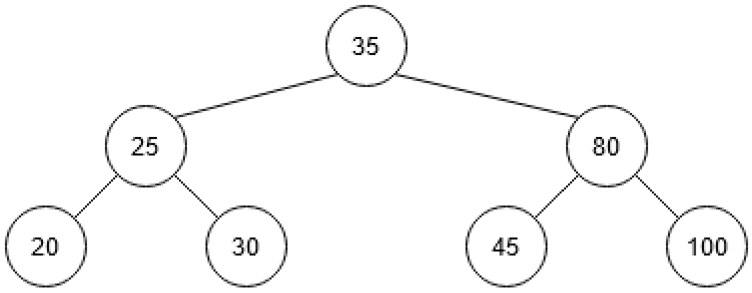


-30

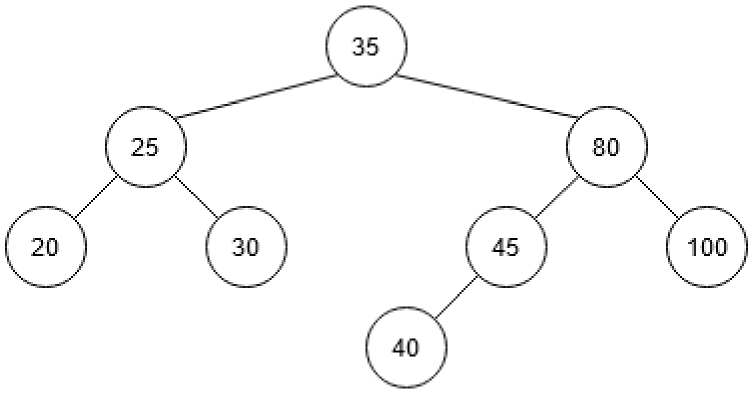


-45

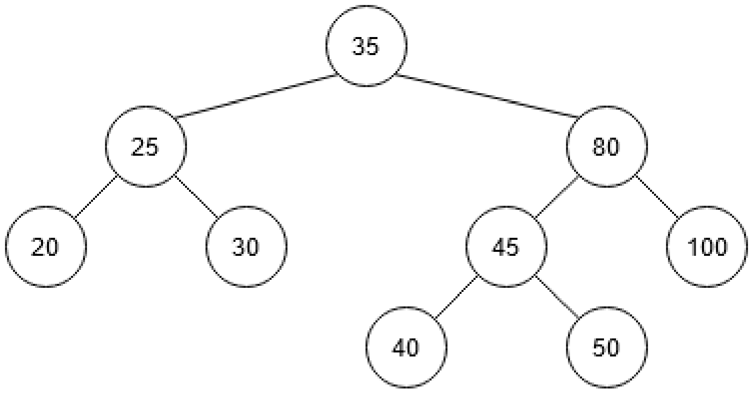
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual



-40

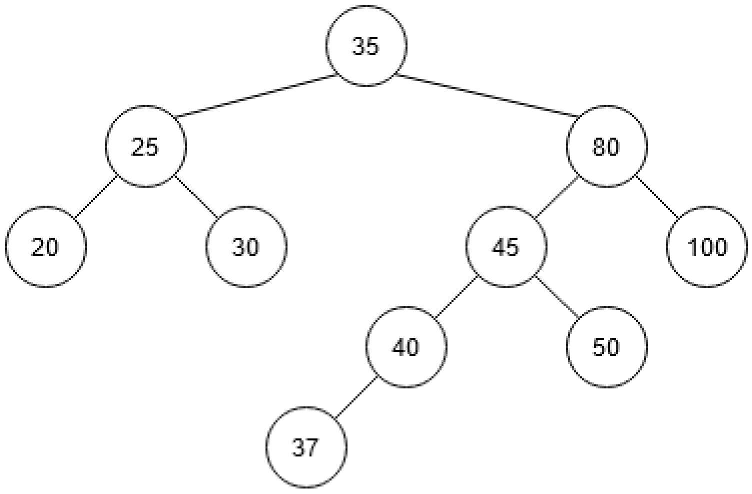


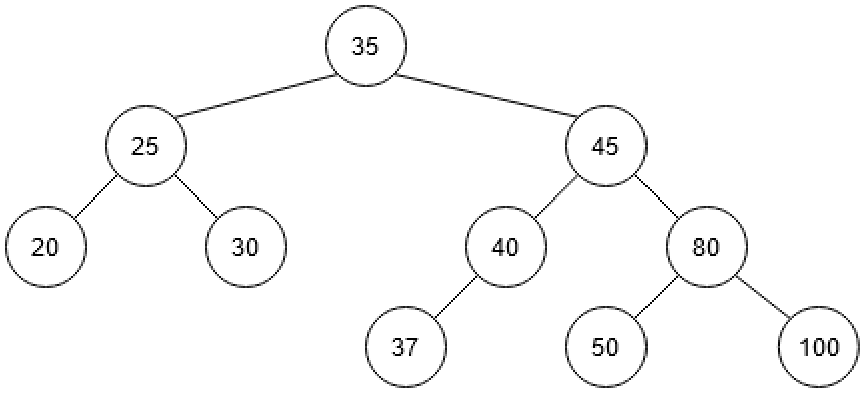
-50



-37

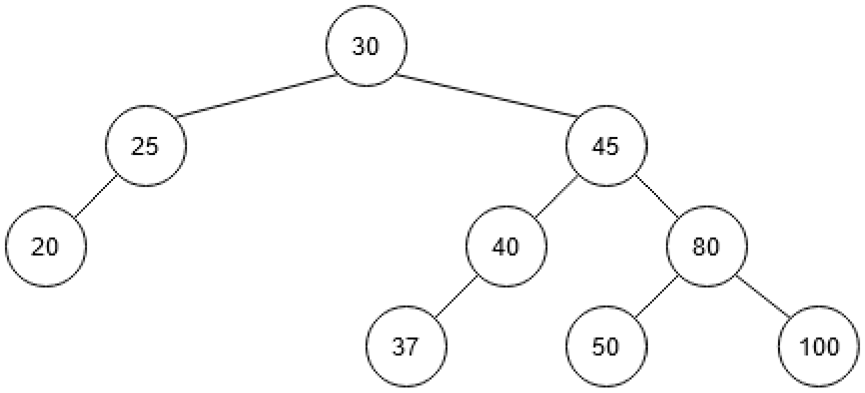
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual





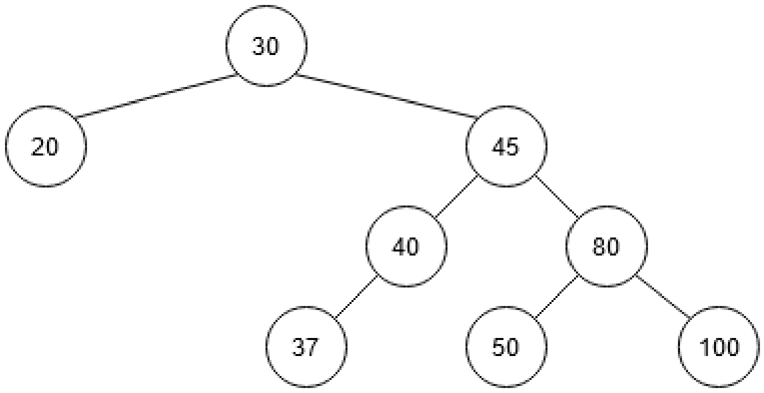
Remove:

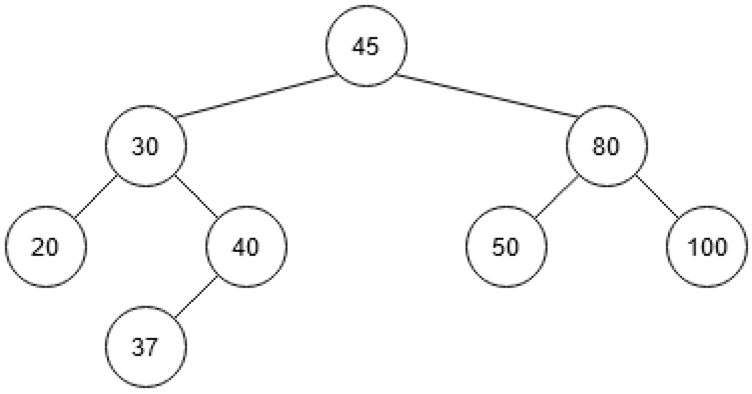
-35



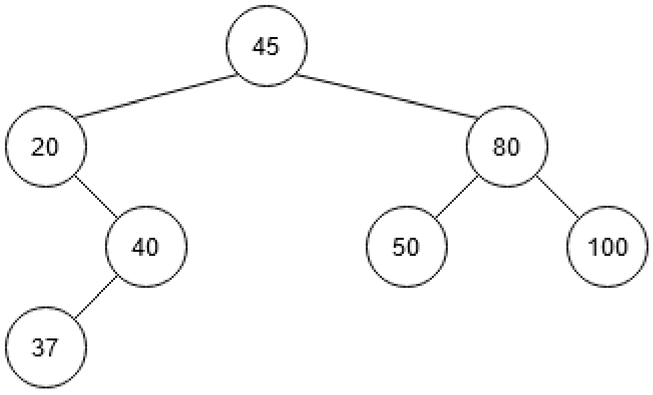
-25

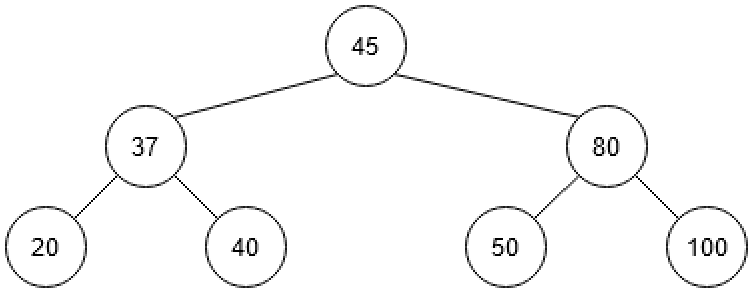
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual





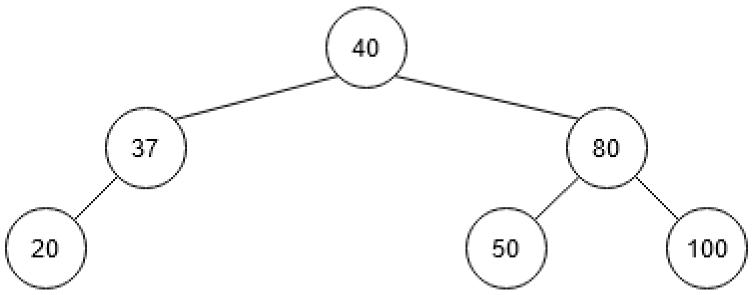
-30



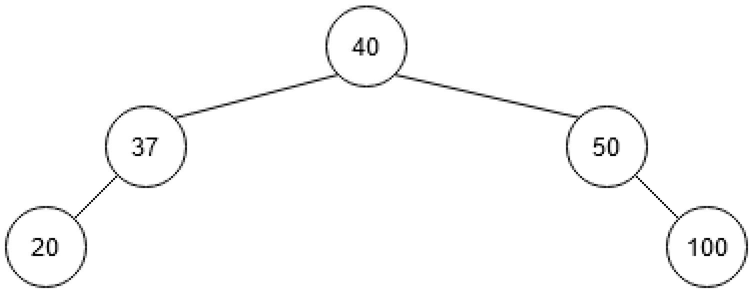


Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

-45

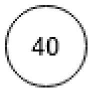


-80

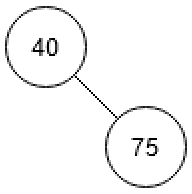


b.)
Insert:

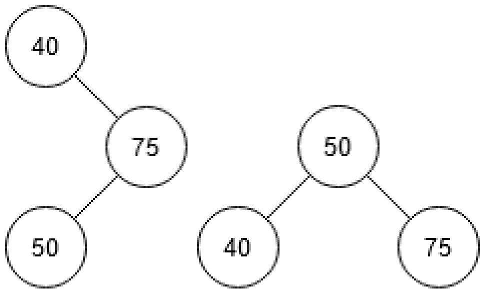
-40



-75

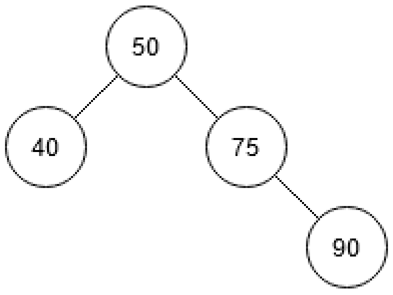


-50

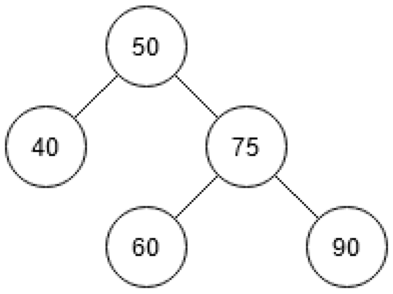


Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

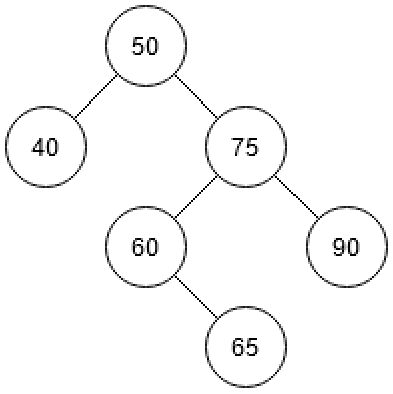
-90

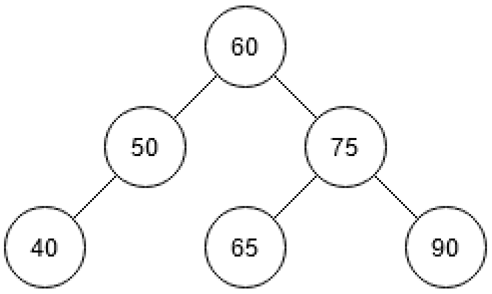


-60

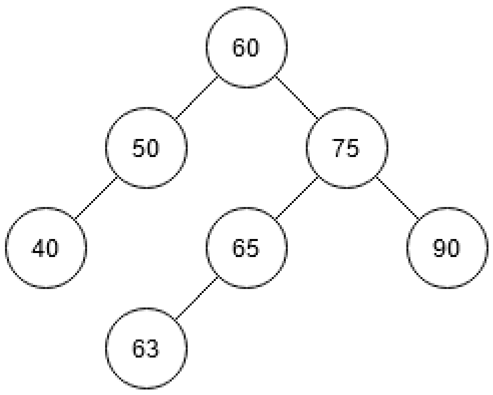


-65



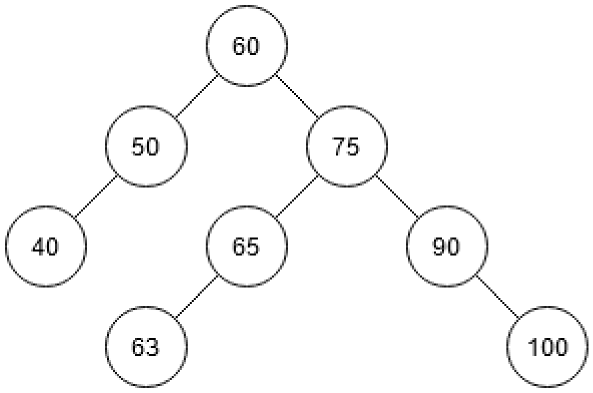


-63

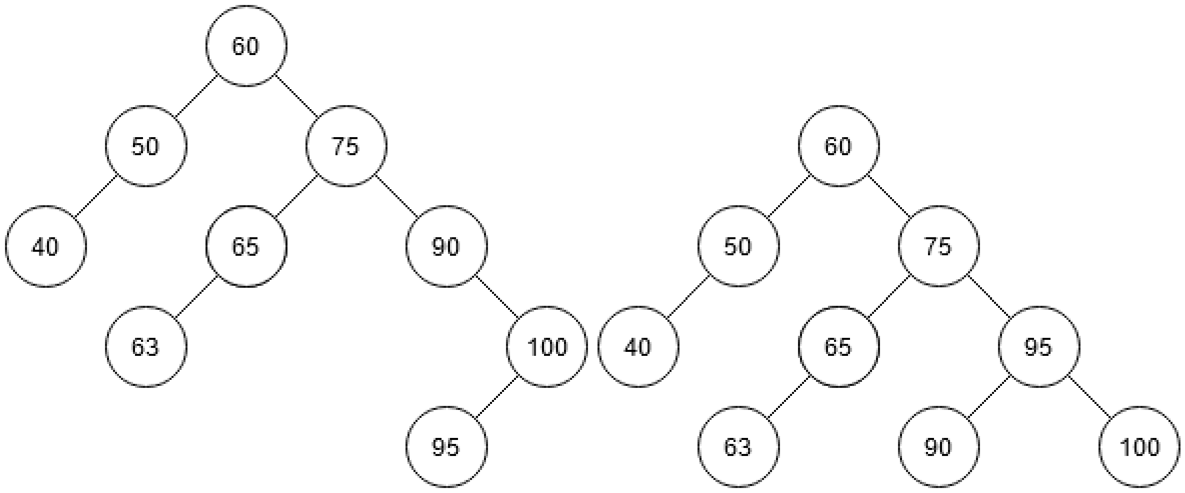


-100

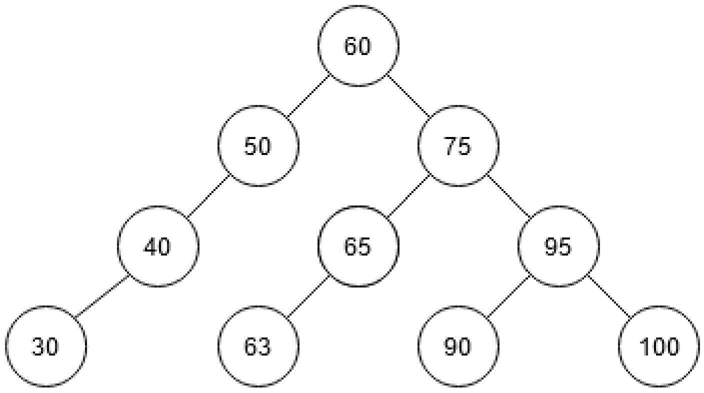
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

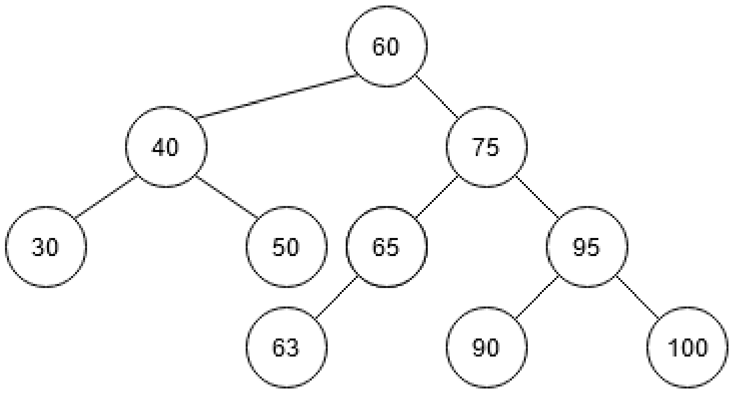


-95



-30

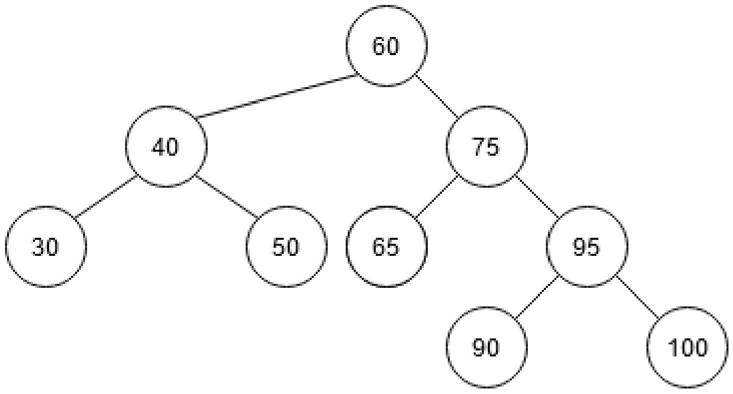




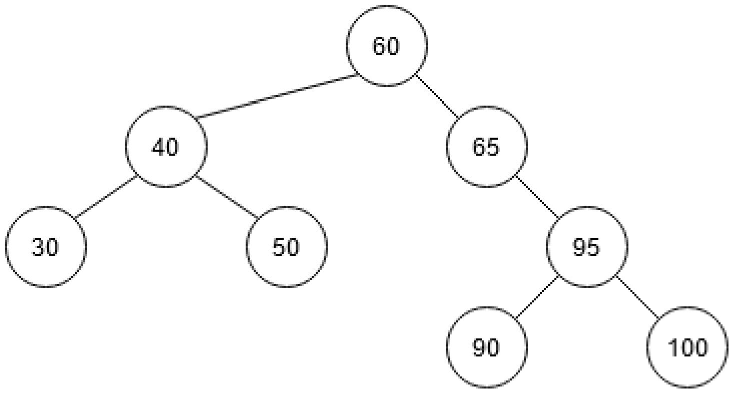
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

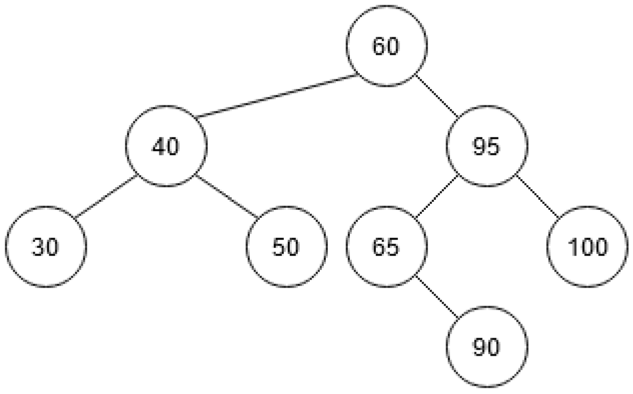
Remove:

-63

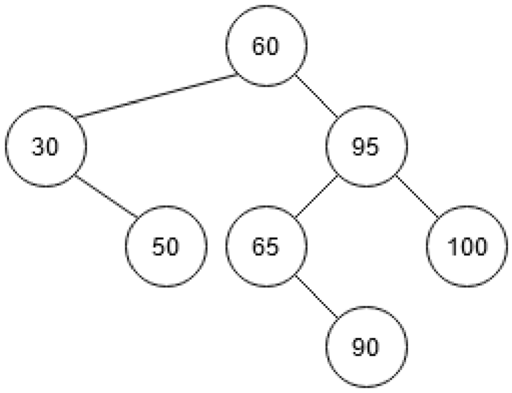


-75



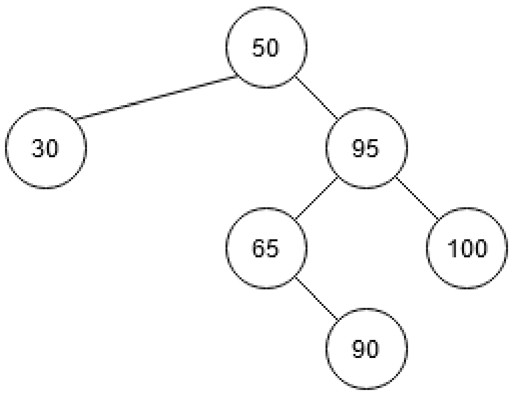


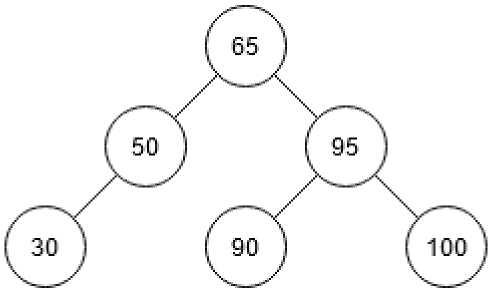
-40



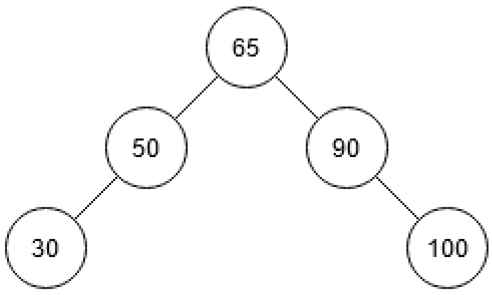
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

-60





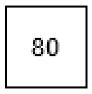
-95



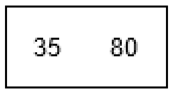
2-3 Tree / B-Tree

a.)
Insert:

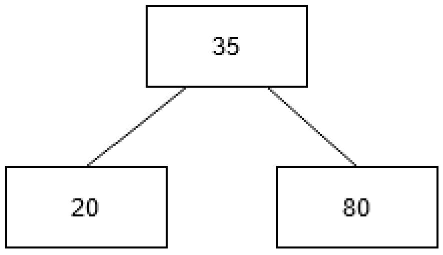
-80



-35

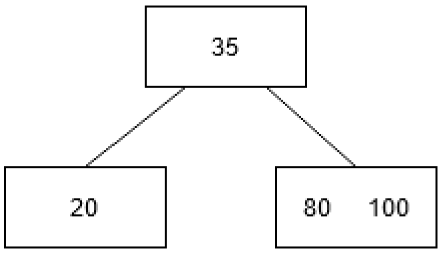


-20

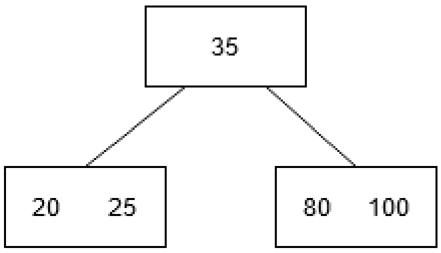


Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

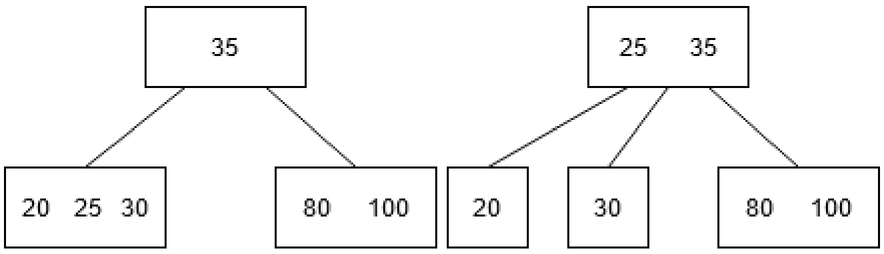
-100



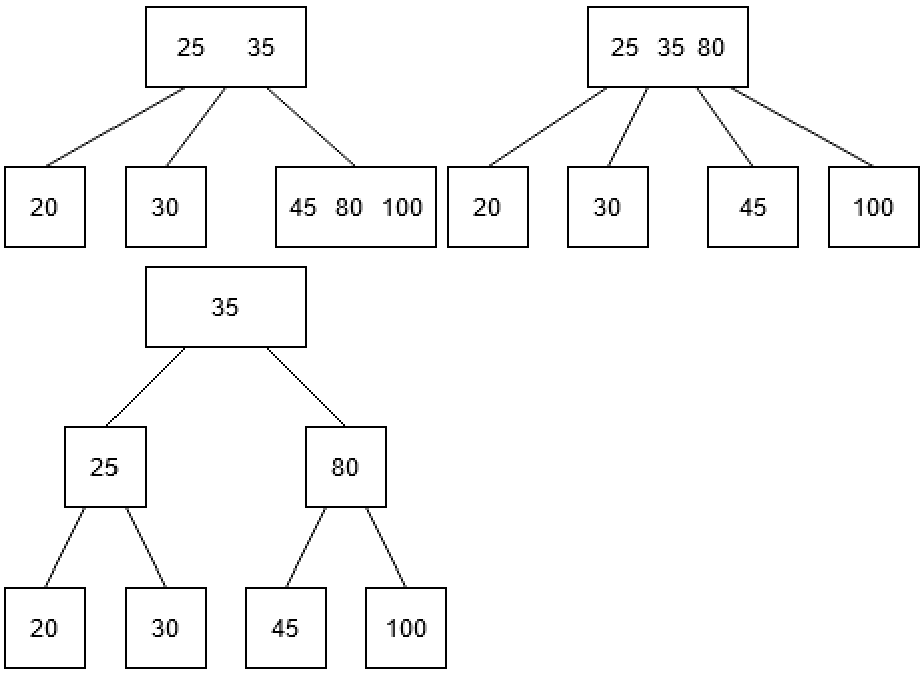
-25



-30

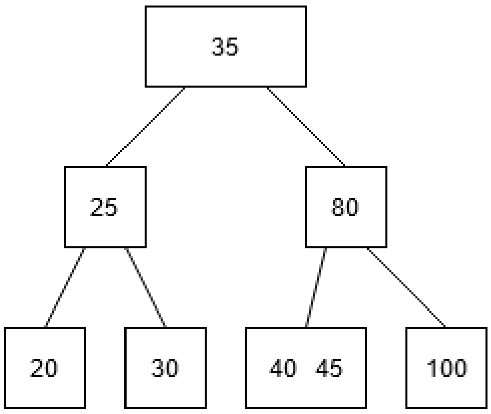


-45

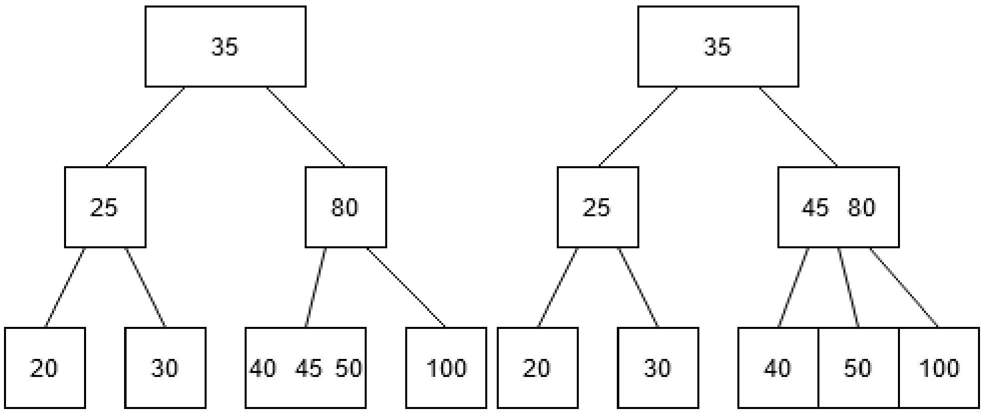


Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

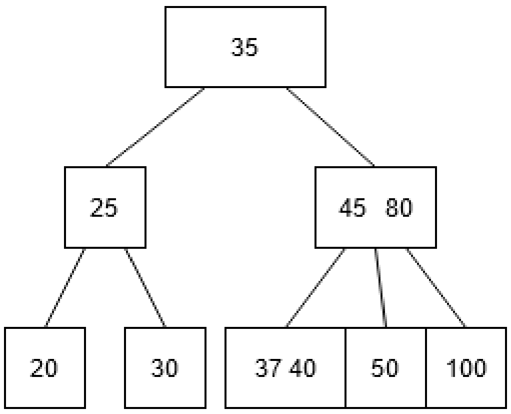
-40



-50



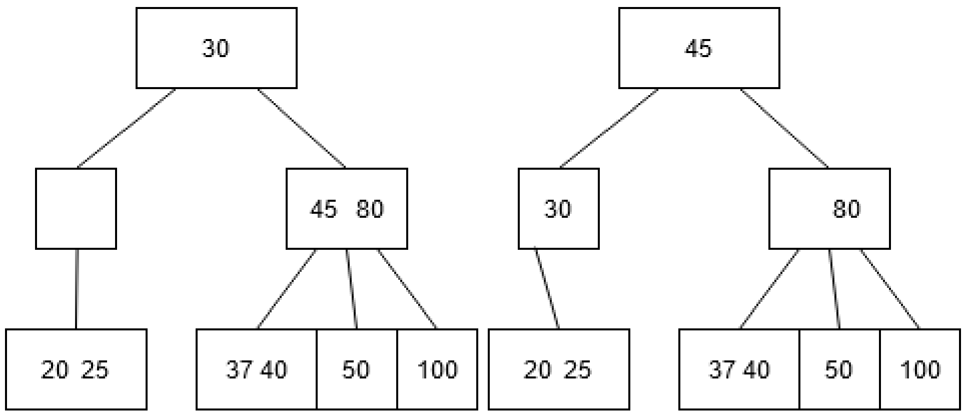
-37



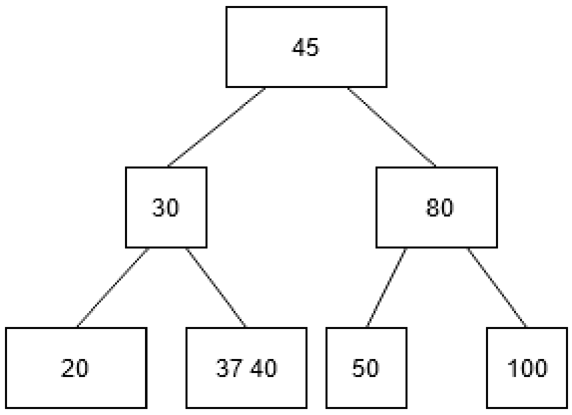
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

Remove:

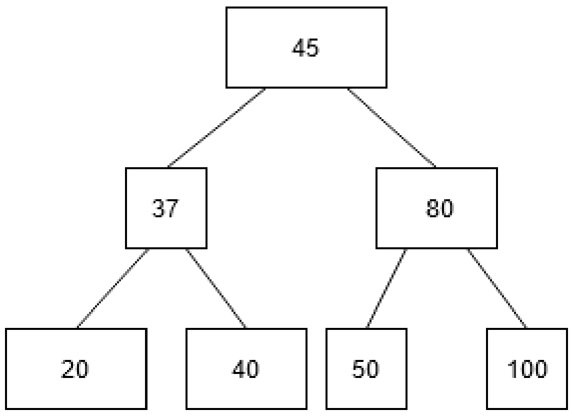
-35



-25

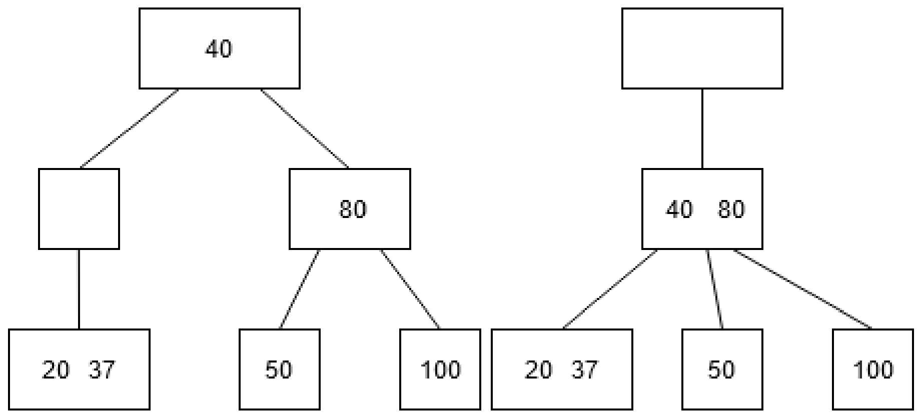


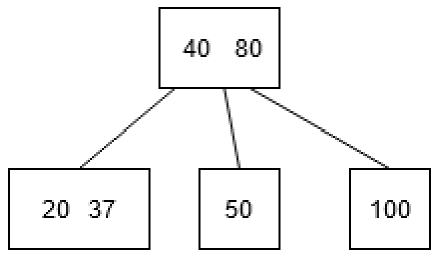
-30



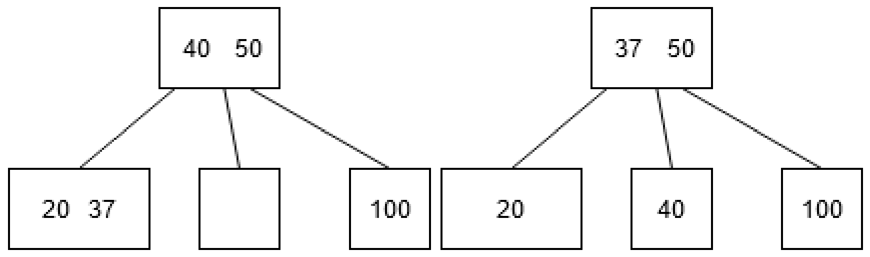
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

-45



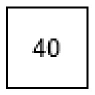


-80

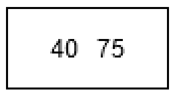


b.)
Insert:

-40

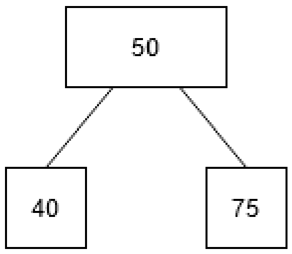


-75

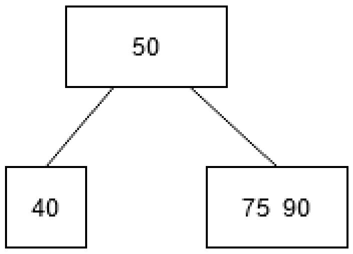


-50

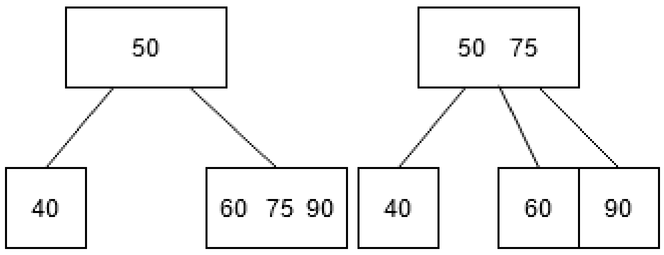
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual



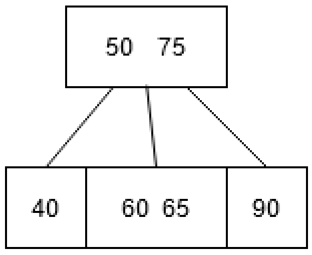
-90



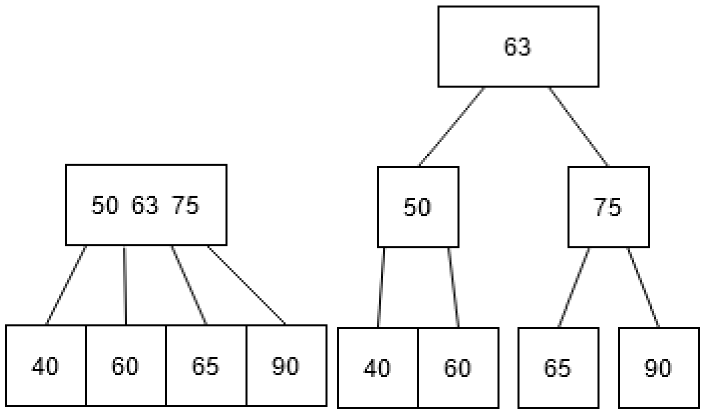
-60



-65

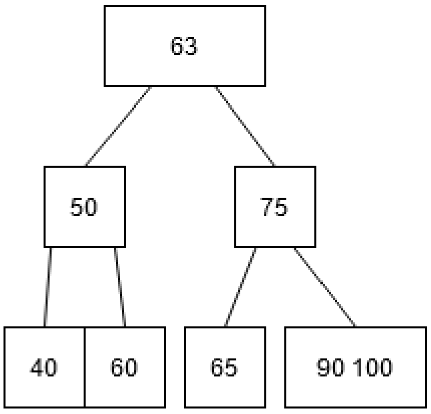


-63

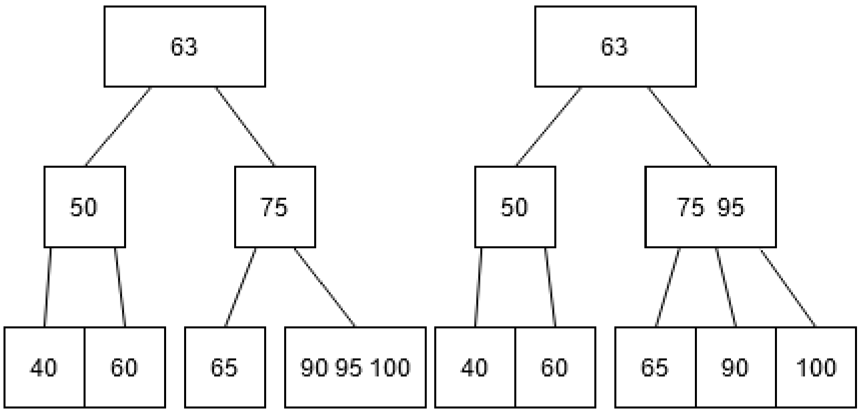


-100

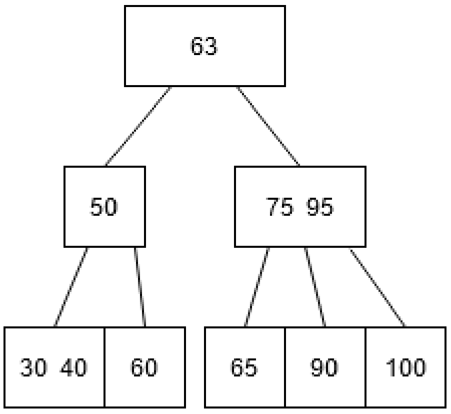
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual



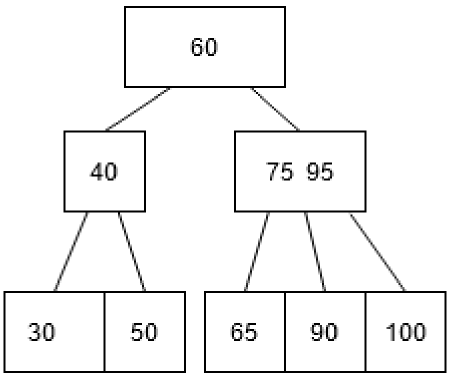
-95



-30

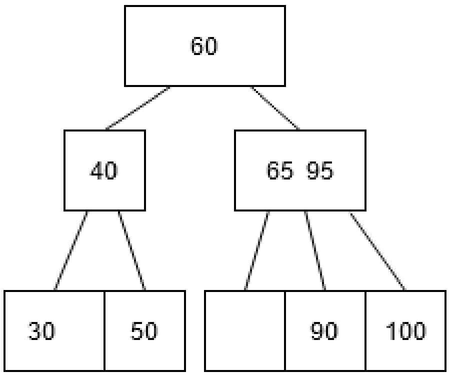


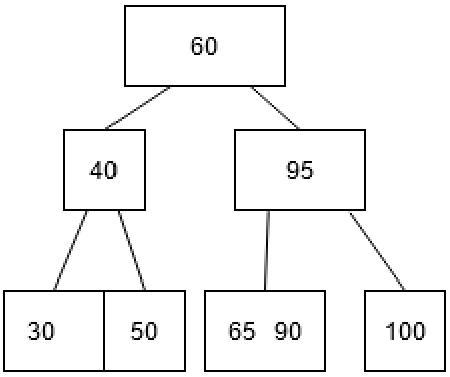
Remove:
-63



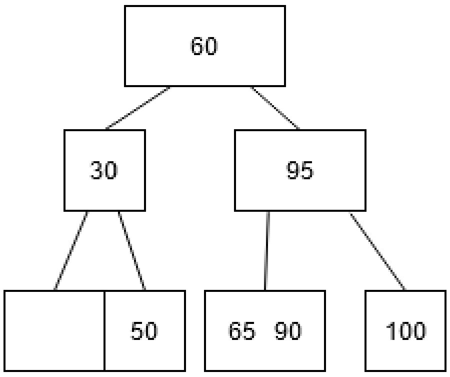
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

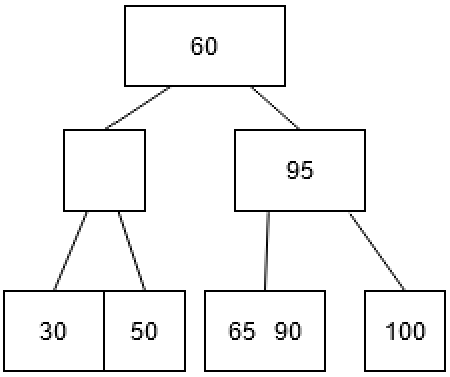
-75

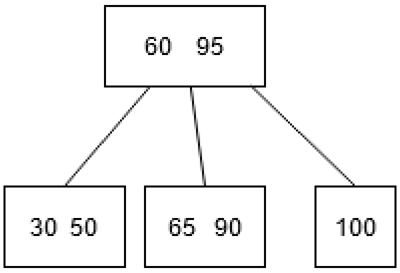




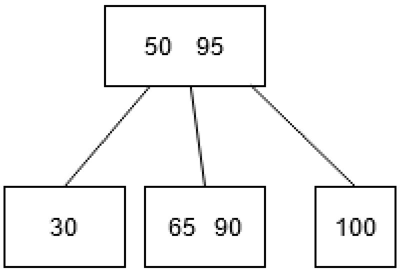
-40



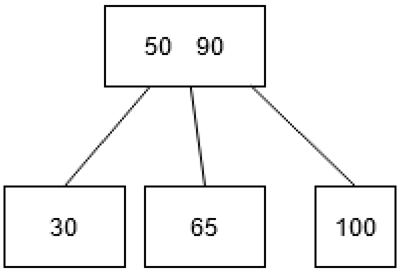




-60



-95

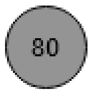


Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

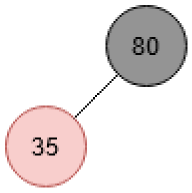
Red Black Tree:

a.)
Insert:

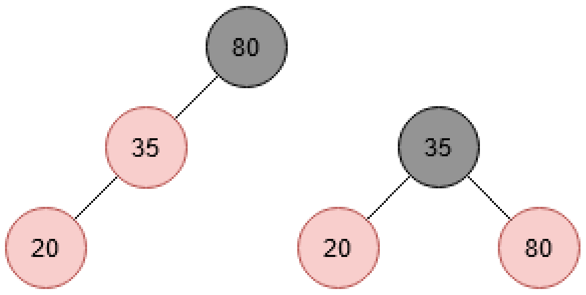
-80



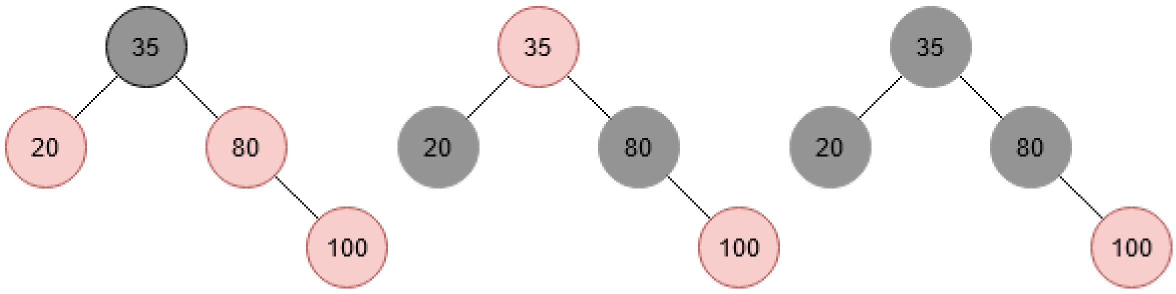
-35



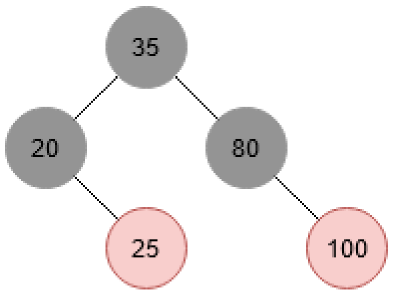
-20



-100

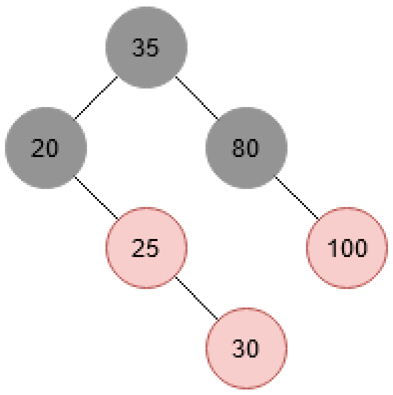


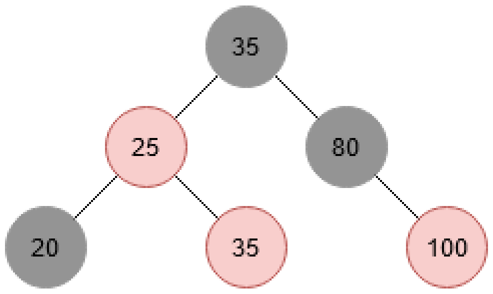
-25

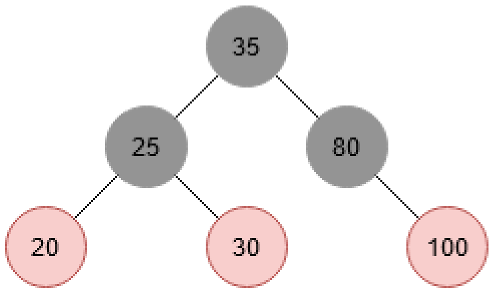


Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

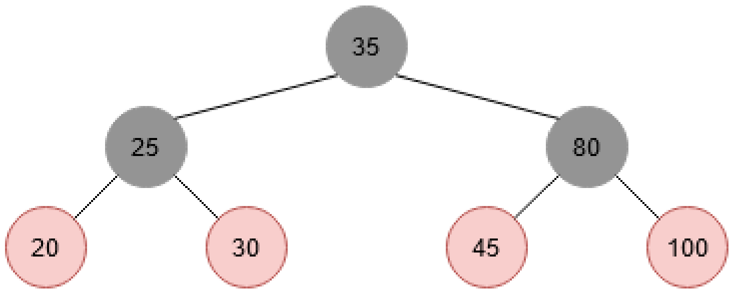
-30





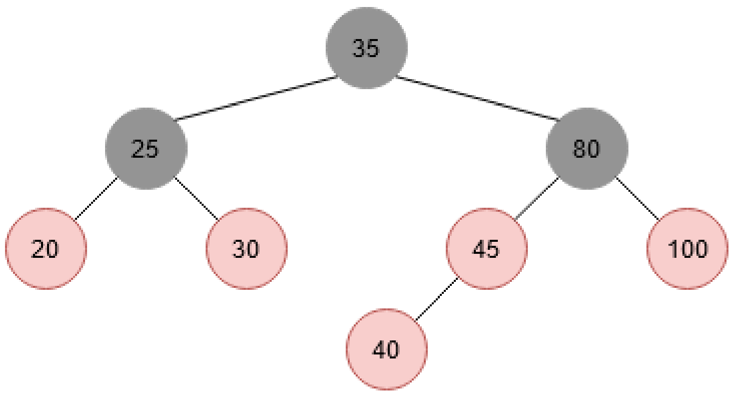


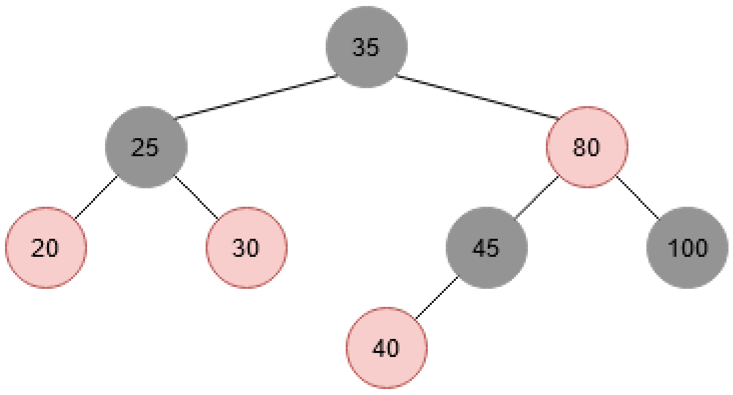
-45



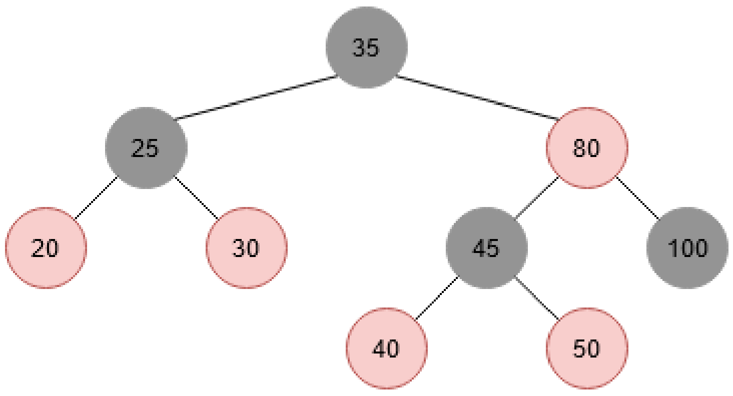
-40

Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual



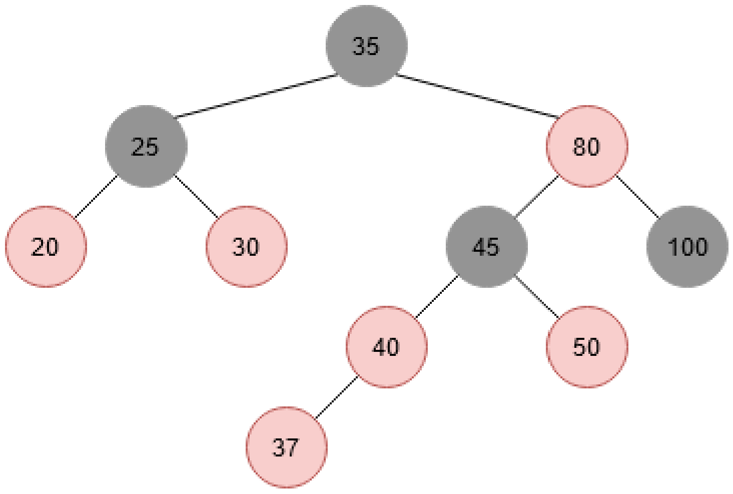


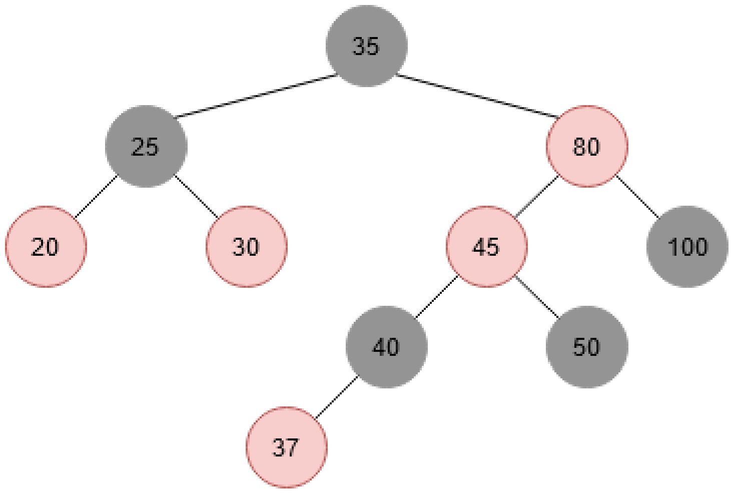
-50

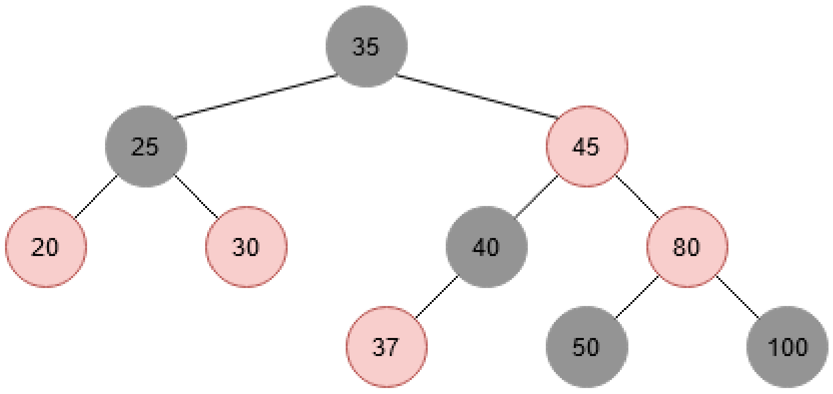


-37

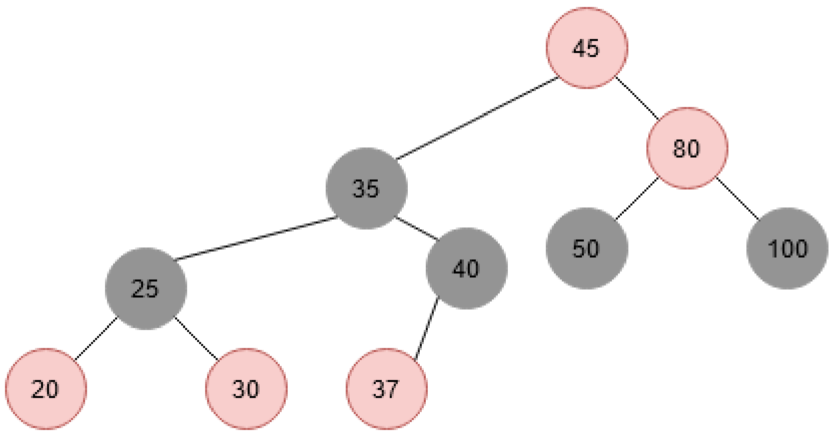
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

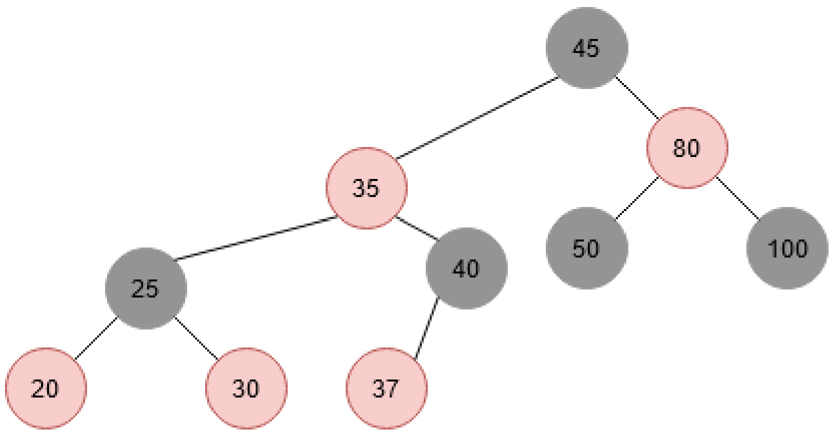






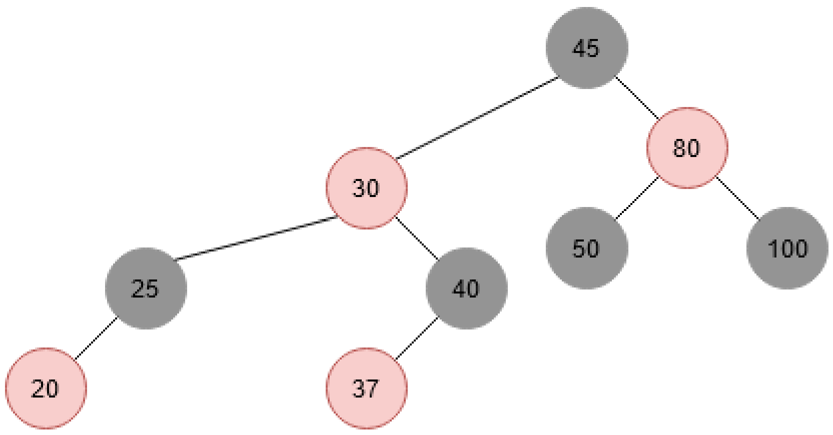
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual





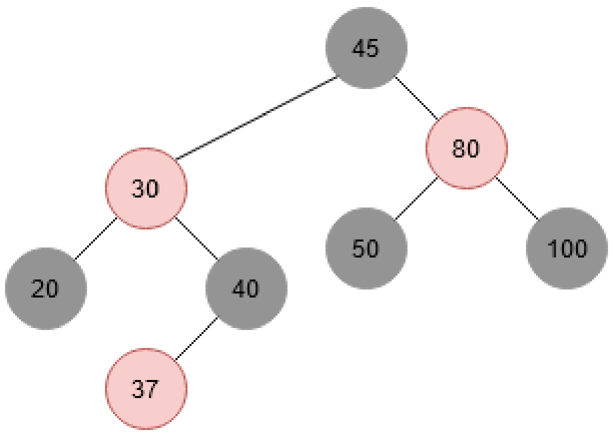
Remove:

-35

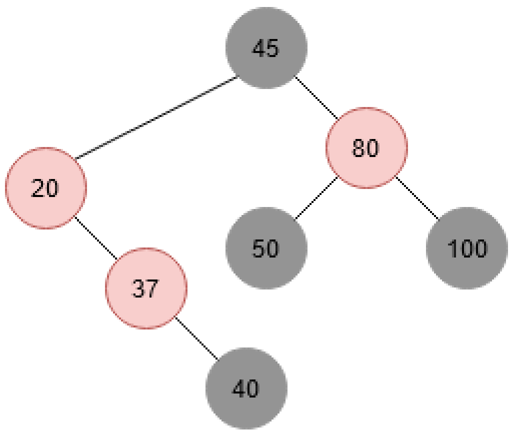


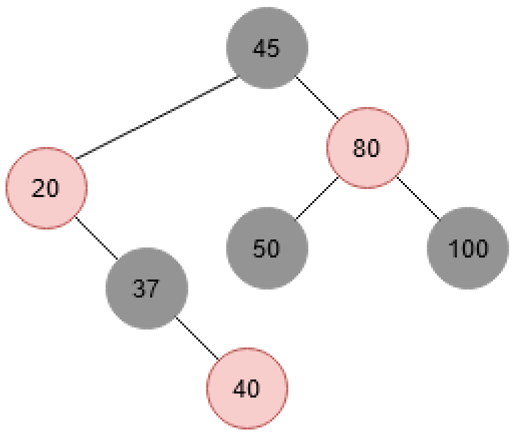
-25

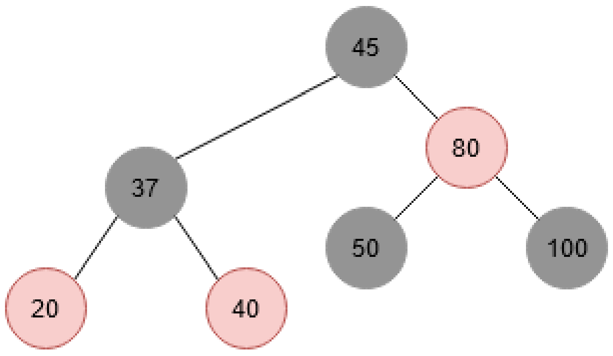
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

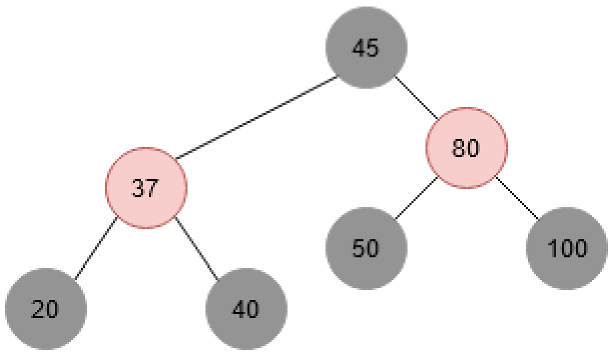


-30



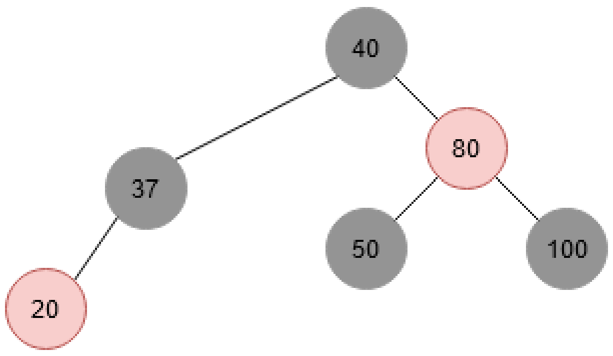




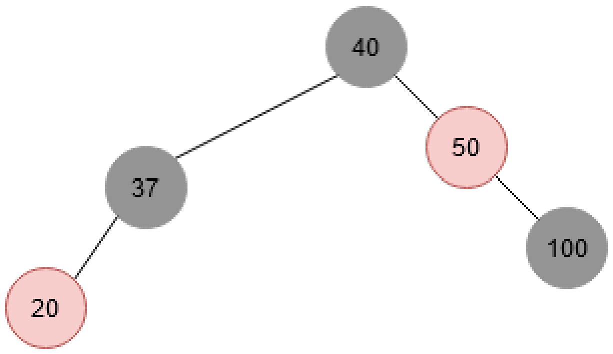


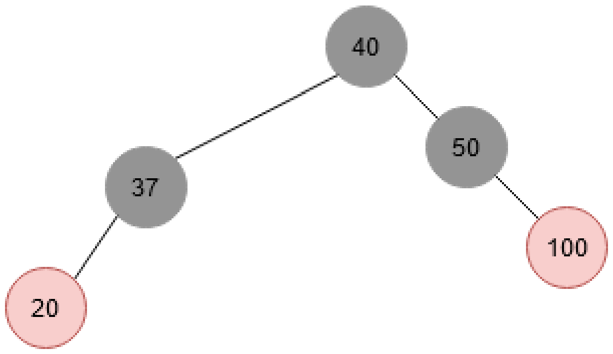
-45

Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual



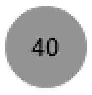
-80



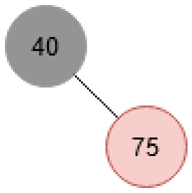


b.)
Insert:

-40

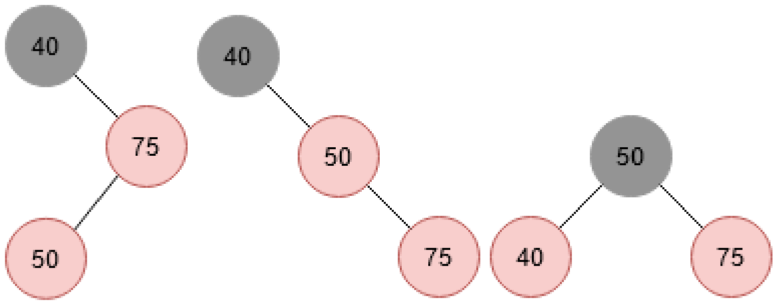


-75

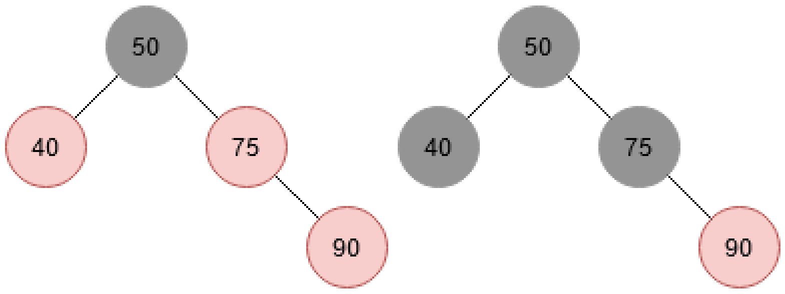


-50

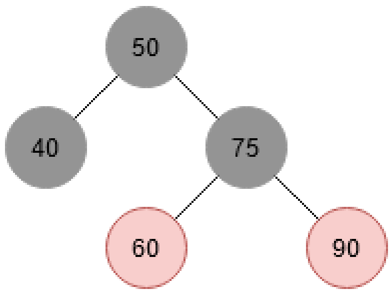
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual



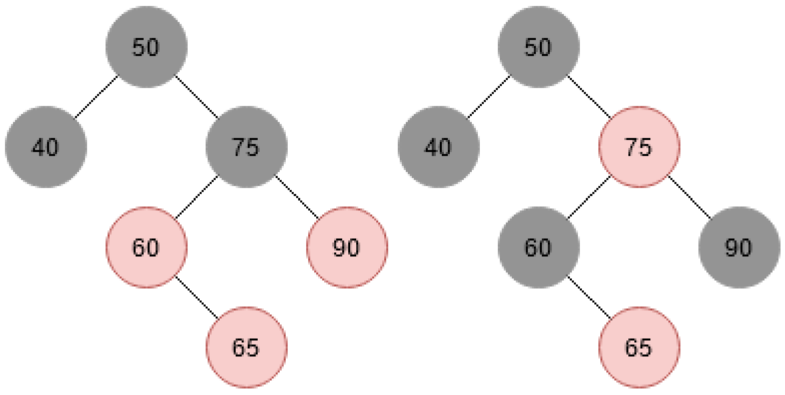
-90



-60

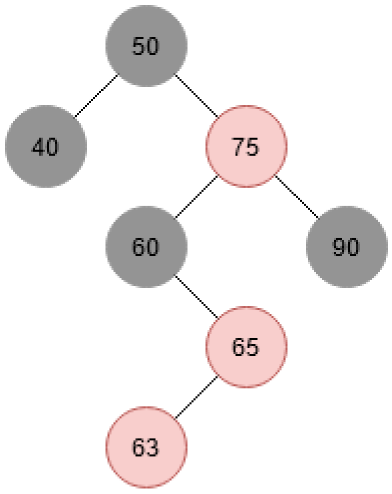


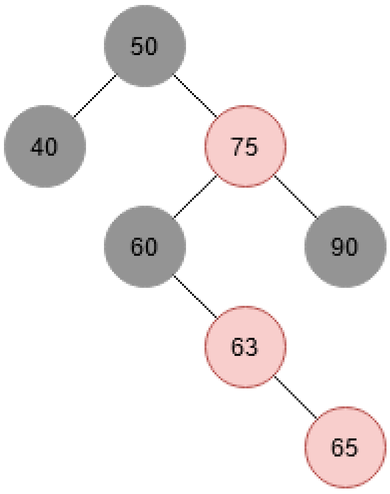
-65

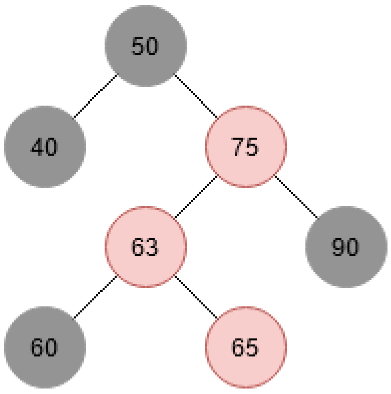


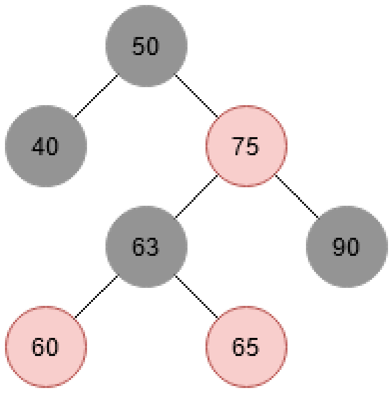
-63

Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

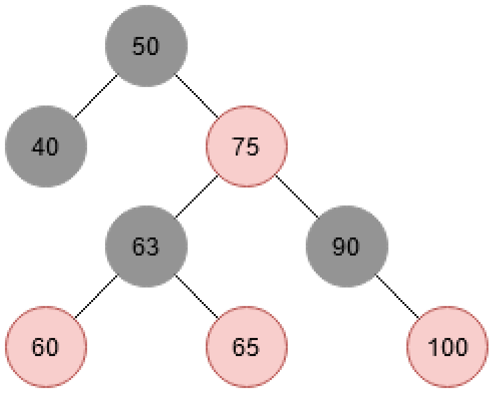






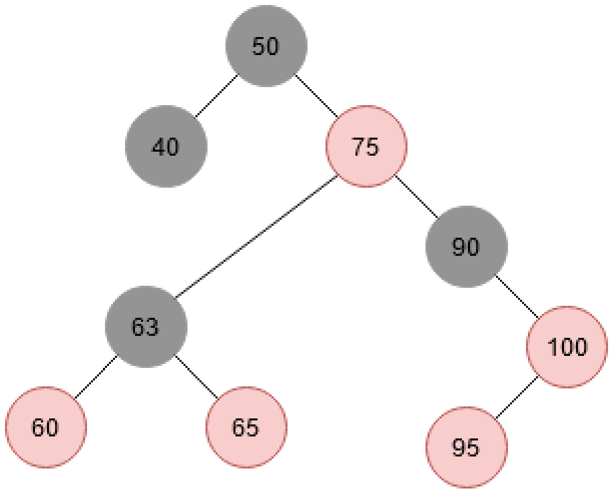


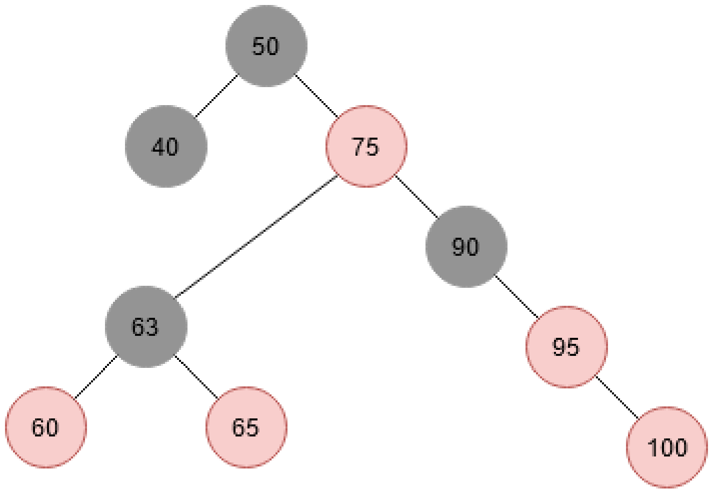
-100

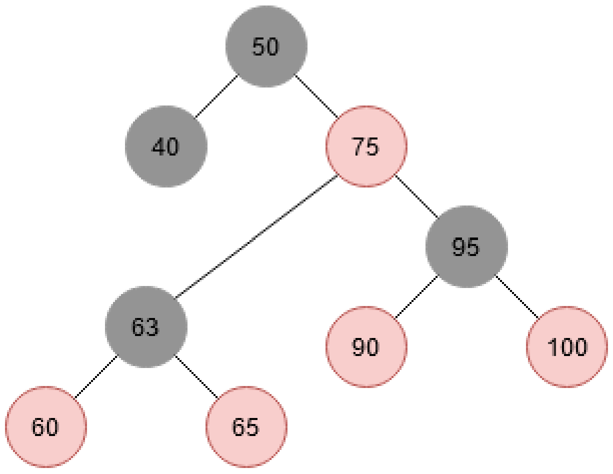


-95

Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

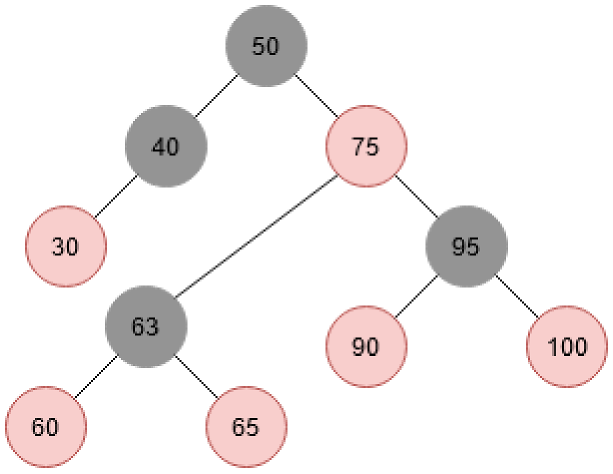




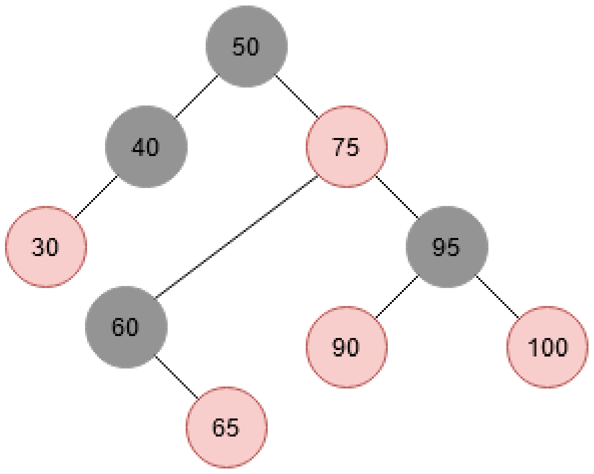


-30

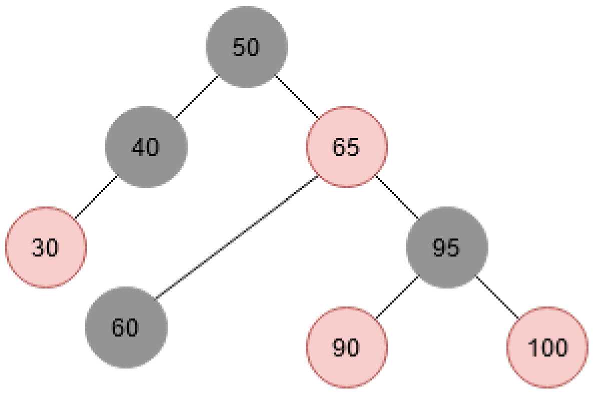
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual



Remove:
-63

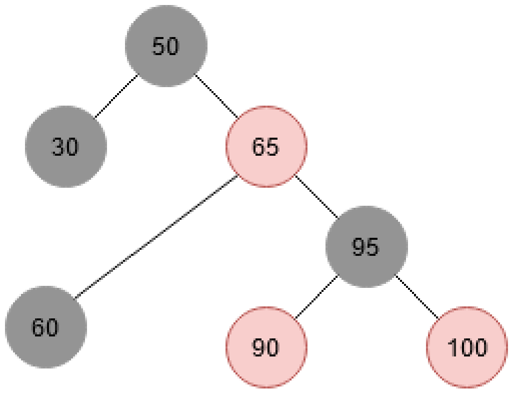


-75

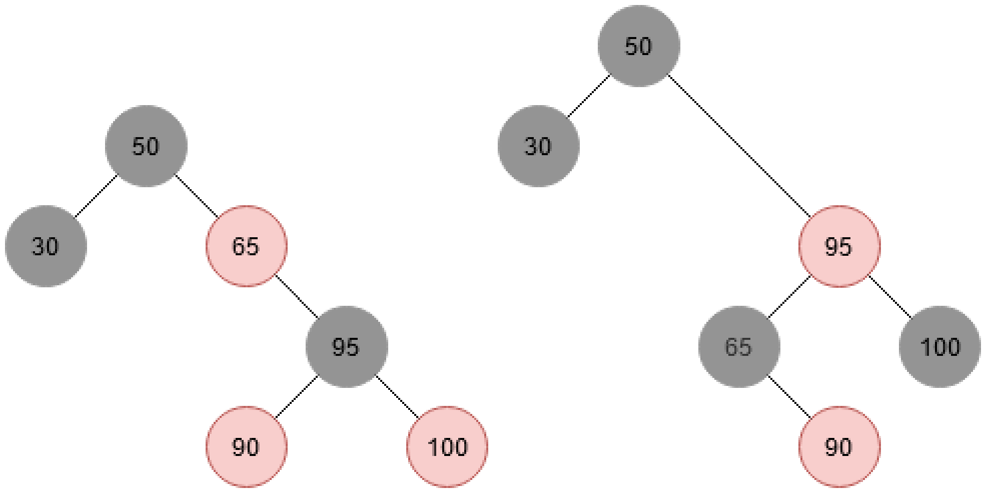


-40

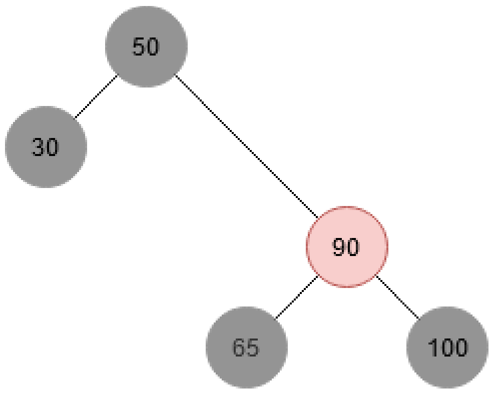
Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual



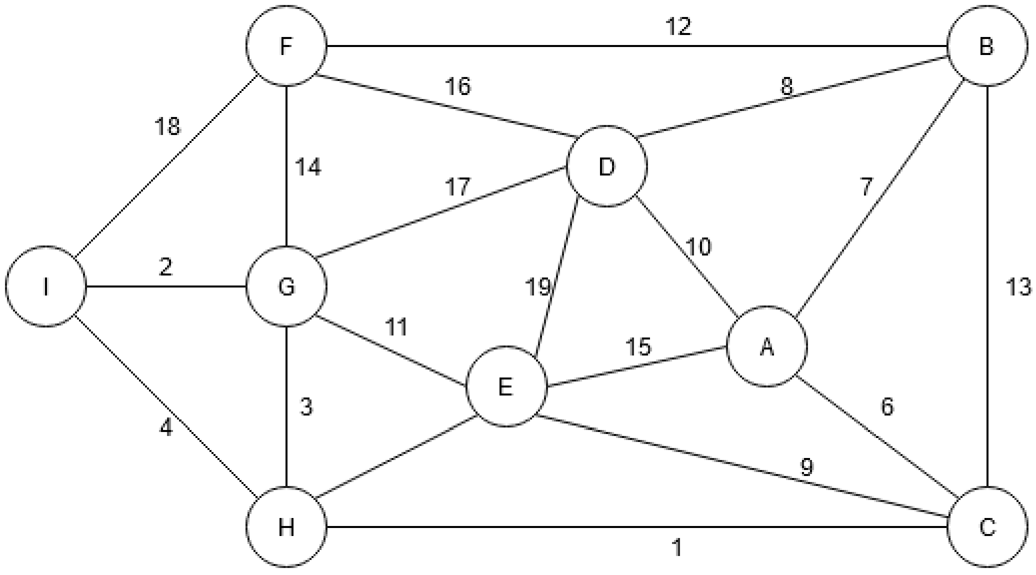
-60



-95

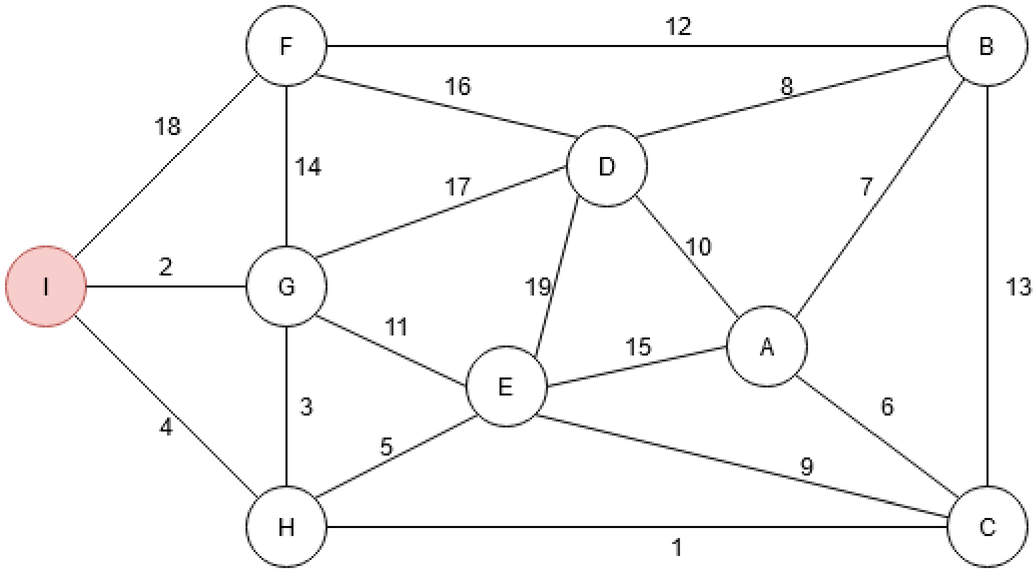


Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual
Disjointed Set & Graphs [9%]



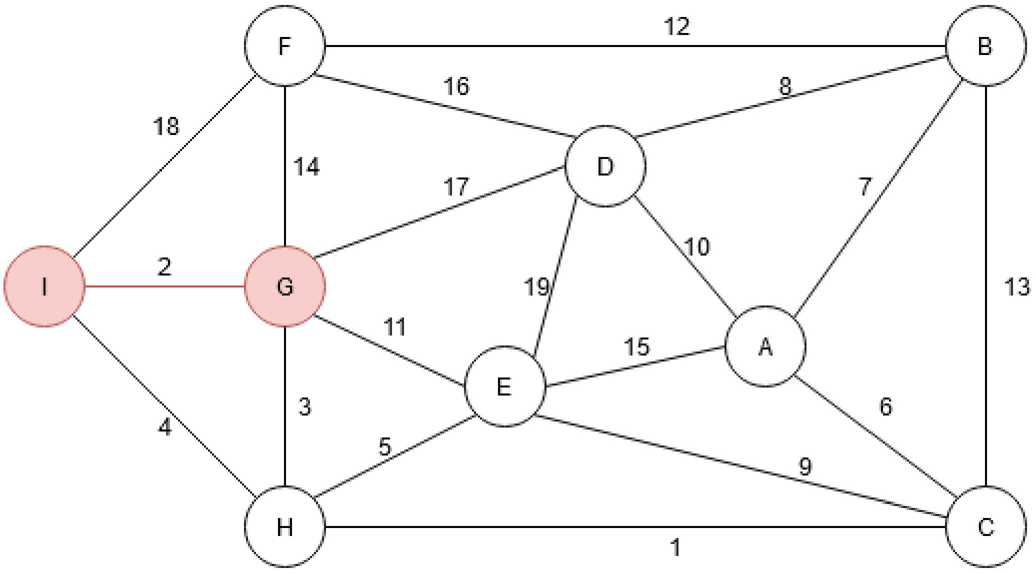
Find minimum spanning tree:

a.) Prim

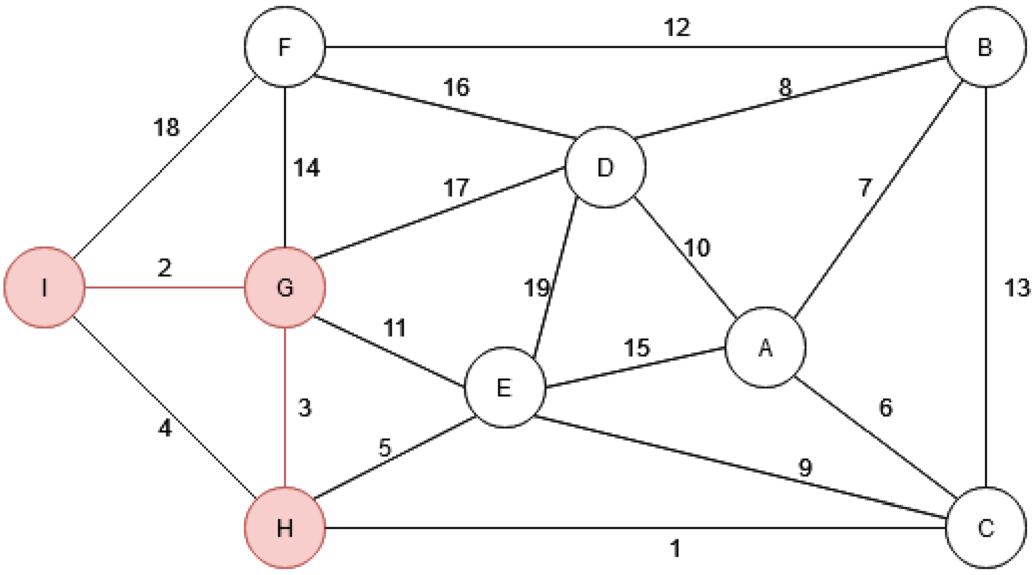


Visited={I}

Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

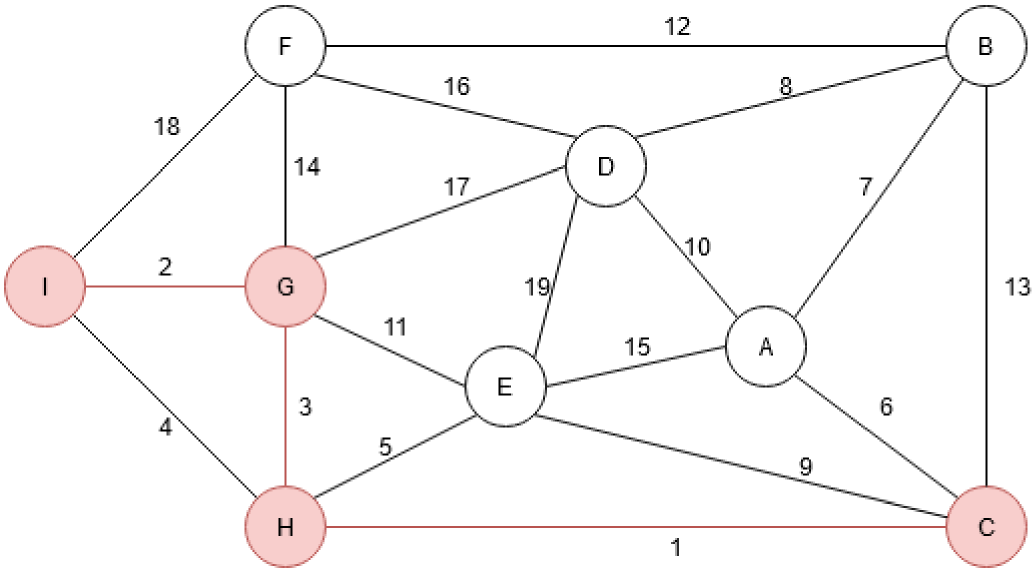


Visited={I, G}

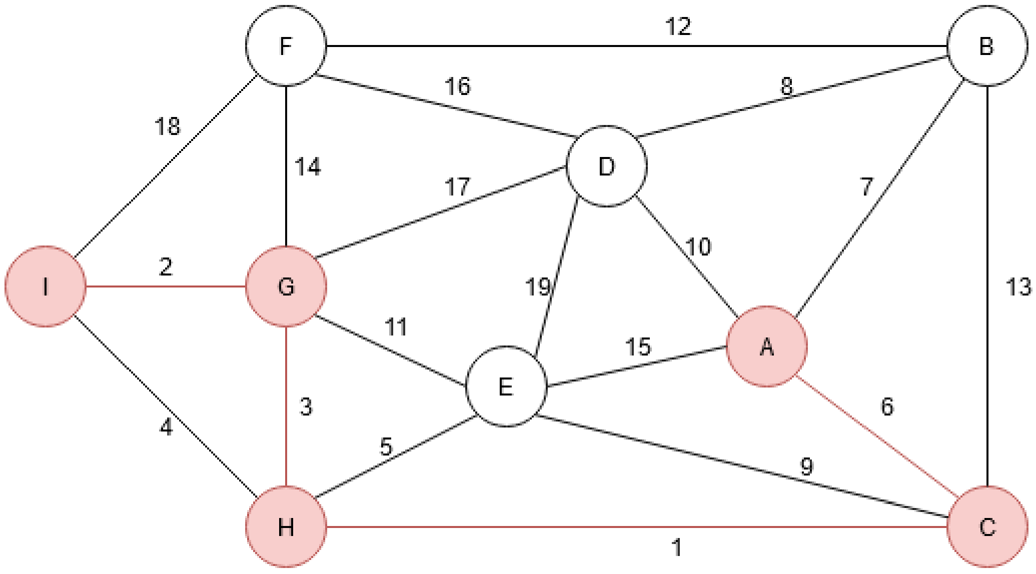


Visited={I, G, H}

Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

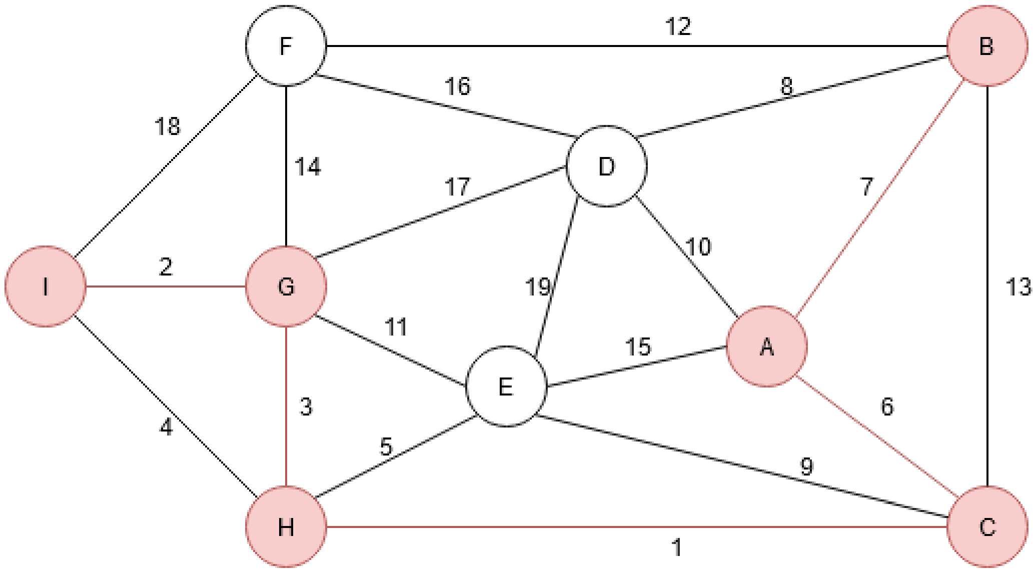


Visited={I, G, H, C}

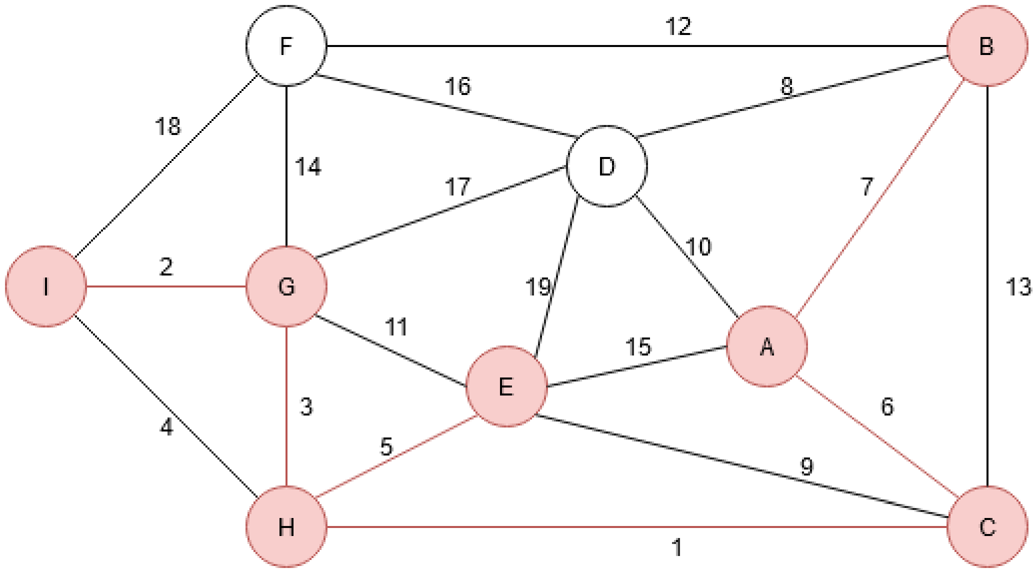


Visited={I, G, H, C, A}

Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

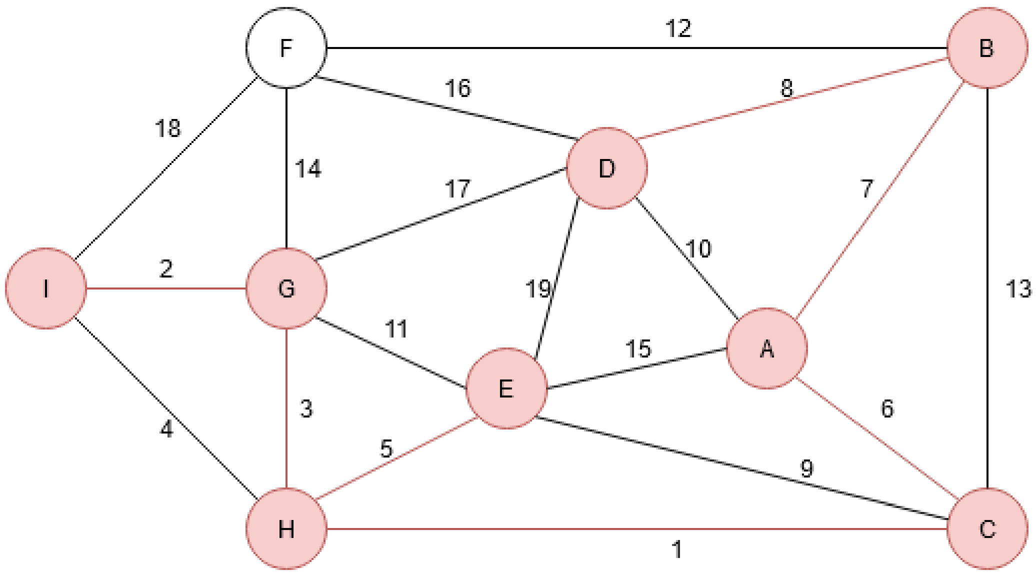


Visited={I, G, H, C, A, B}

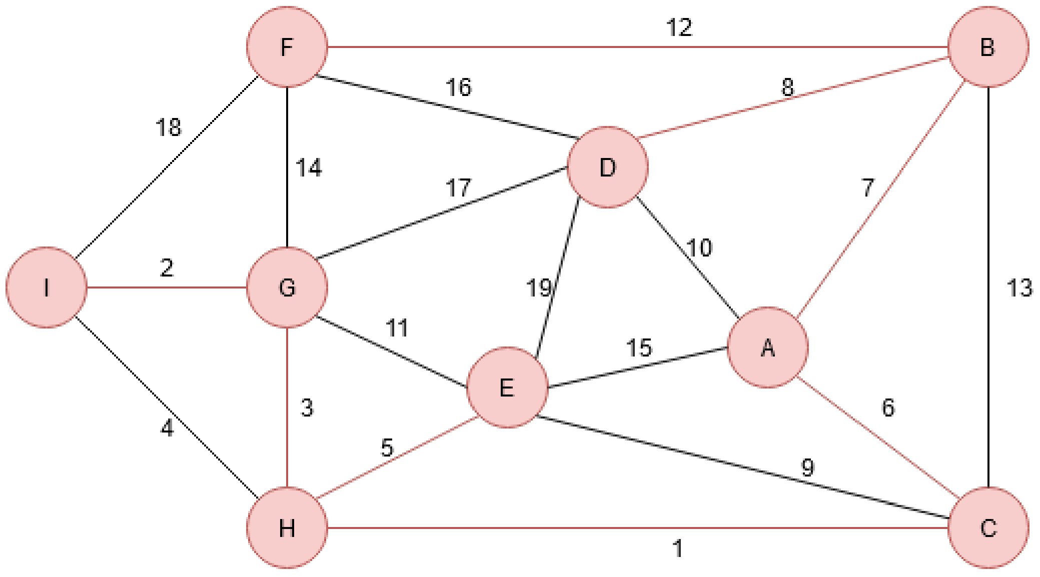


Visited={I, G, H, C, A, B, E}

Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual



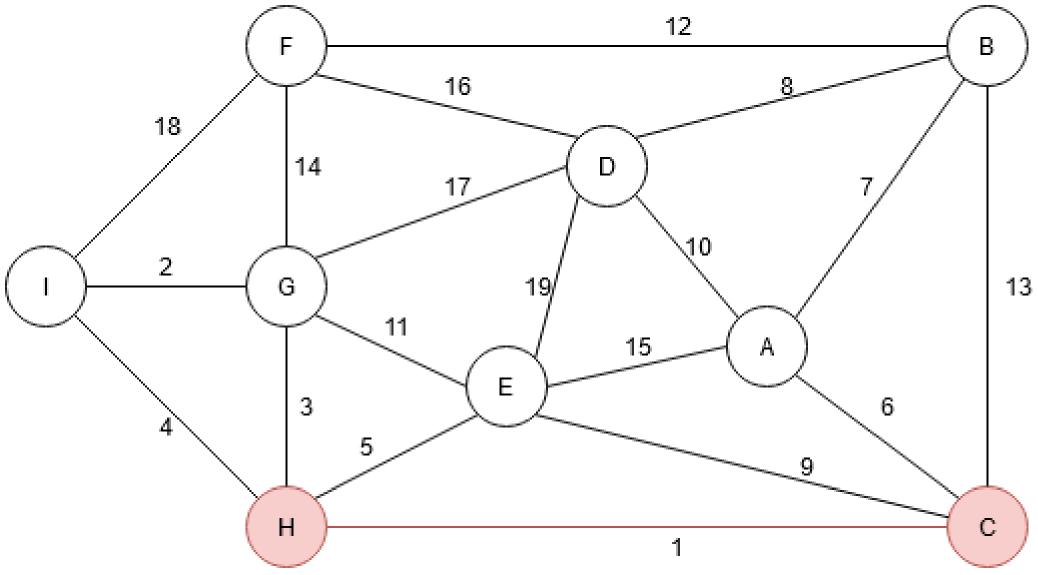
Visited={I, G, H, C, A, B, E, D}



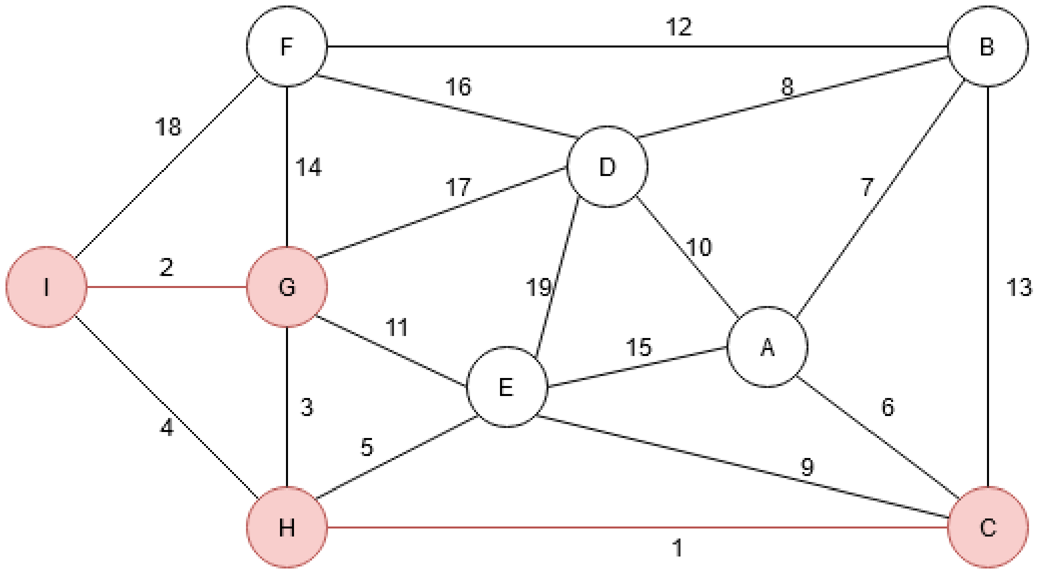
Visited={I, G, H, C, A, B, E, D, F}

b.) Kruskal

Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

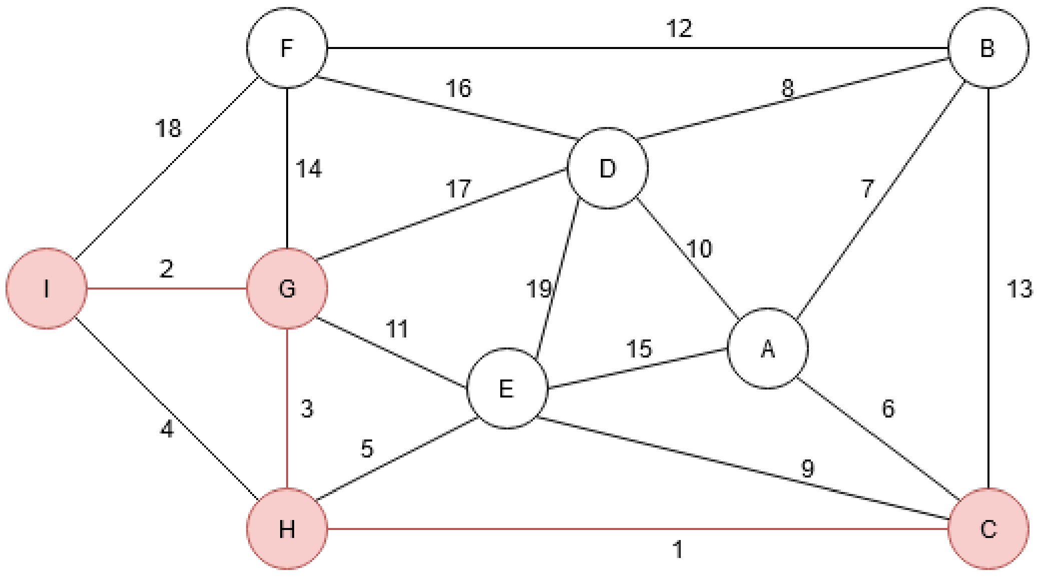


Visited: {H, C}

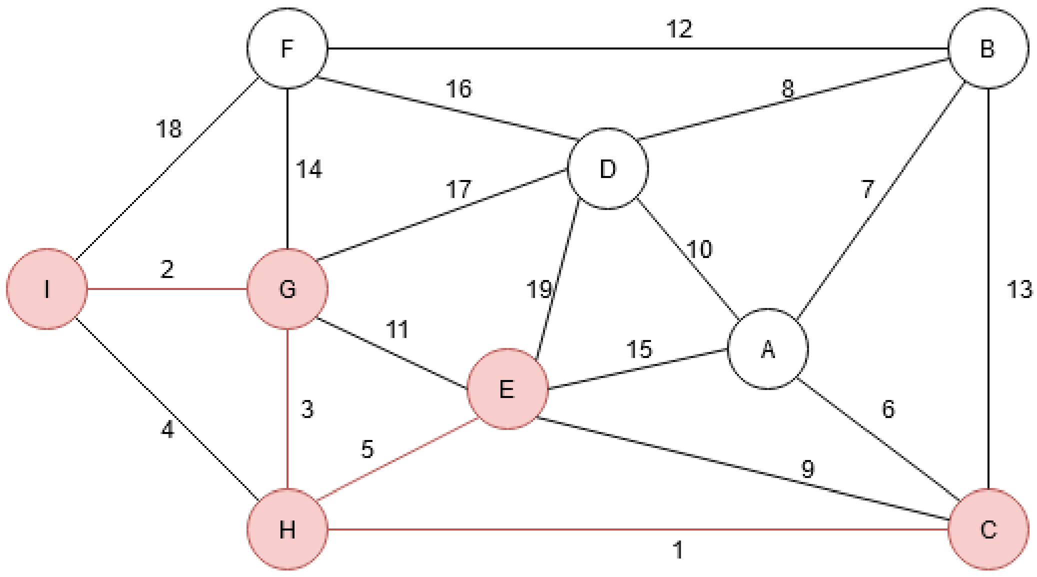


Visited: {H, C, I, G}

Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

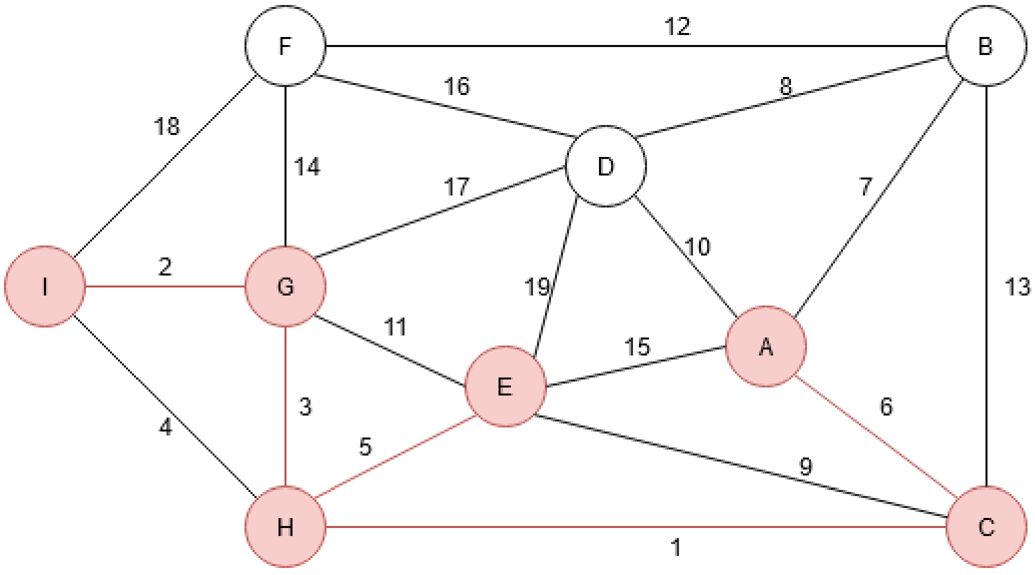


Visited: {H, C, I, G}

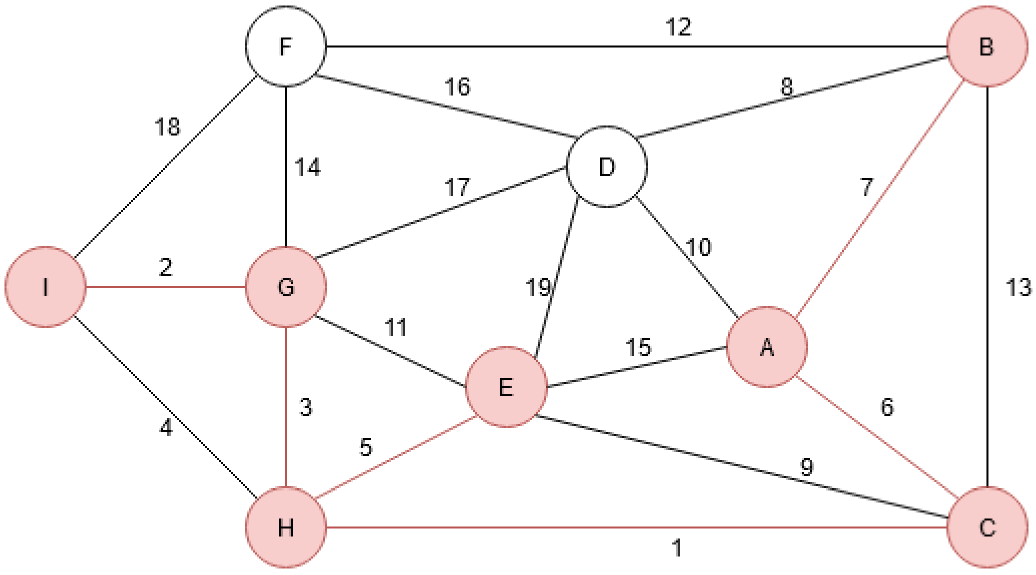


Visited: {H, C, I, G, E}

Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

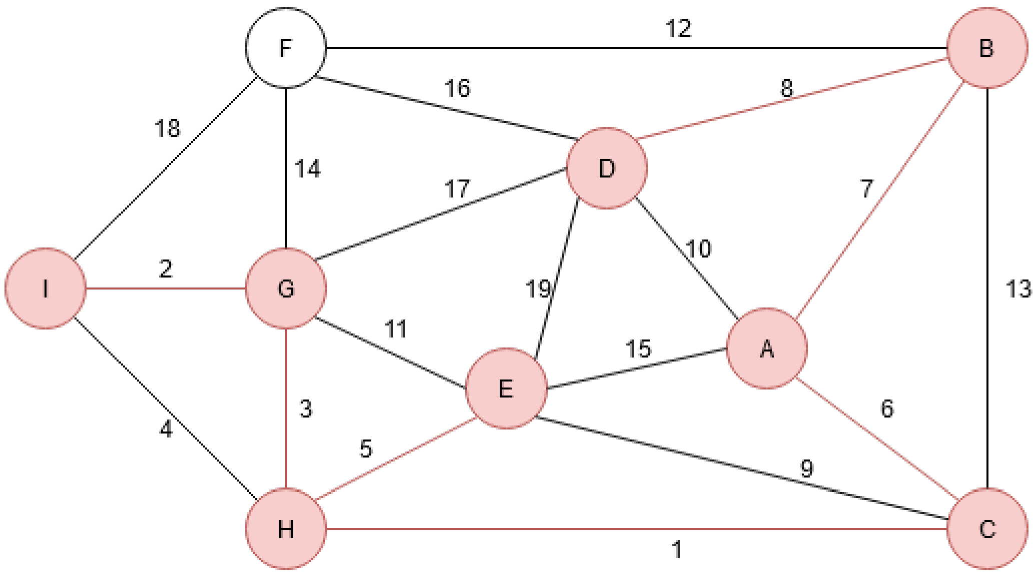


Visited: {H, C, I, G, E, A}

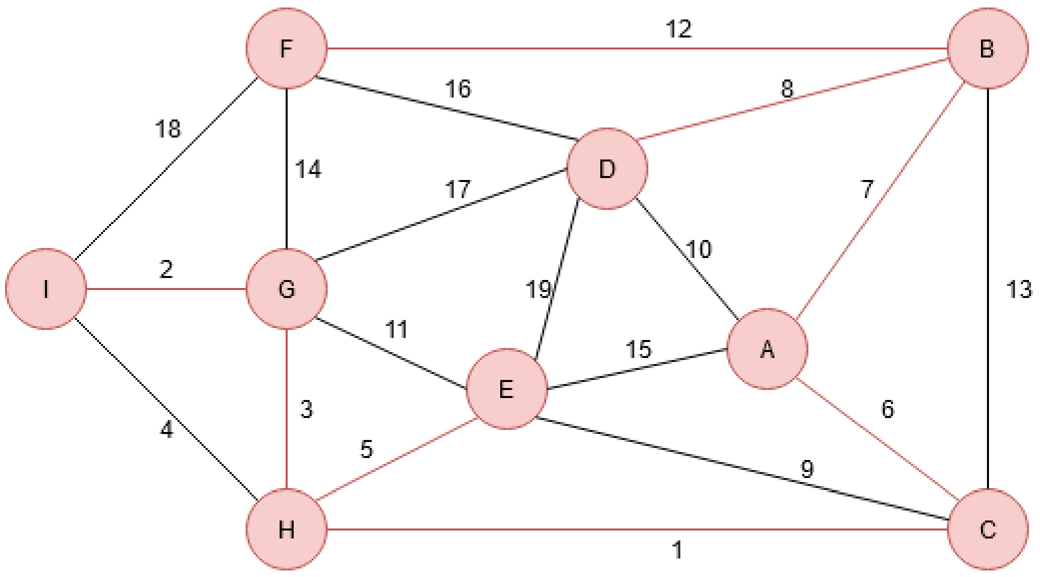


Visited: {H, C, I, G, E, A, B}

Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual



Visited: {H, C, I, G, E, A, B}

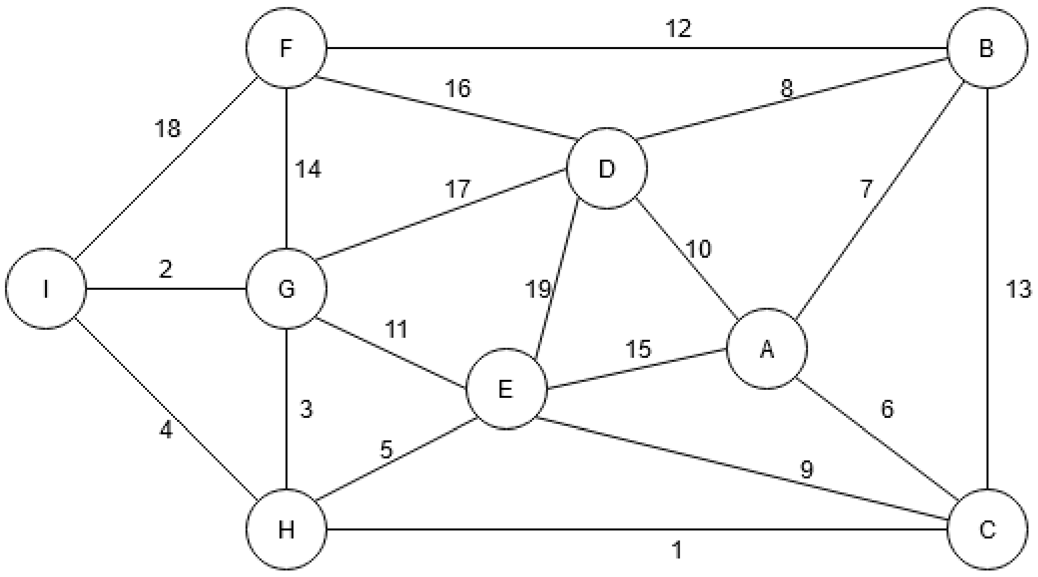


Visited: {H, C, I, G, E, A, B, F}

Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

Shortest path using Djikstra's algorithm from:

a.) I to A
b.) F to C



a.) I to A

	I	F	G	H	D	E	A	B	C
I	0-I	18-I	2-I	4-I	∞	∞	∞	∞	∞
F	X	18-I	2-I	4-I	34-F	∞	∞	30-F	∞
G	X	16-G	X	4-I	19-G	13-G	∞	30-F	∞
H	X	16-G	X	X	19-G	9-H	∞	30-F	5-H
D	X	16-G	X	X	19-G	9-H	29-D	27-D	X
E	X	16-G	X	X	19-G	X	24-E	27-D	X
A	X	X	X	X	19-G	X	24-E	27-D	X
B	X	X	X	X	X	X	X	27-D	X
C	X	X	X	X	X	X	X	18-C	X

Shortest path: I-H-E-A

Sathya Narendra Atmajati Satoto
Group: Kick_Kepin
Final Project Individual

B.) F to C

	F	I	G	D	B	H	E	A	C
F	0-F	18-F	14-F	16-F	12-F	∞	∞	∞	∞
I	X	18-F	14-F	16-F	12-F	∞	∞	∞	∞
G	X	16-G	14-F	16-F	X	17-G	25-G	∞	∞
D	X	16-G	X	16-F	X	17-G	25-G	26-D	∞
B	X	X	X	16-F	X	17-G	25-G	19-B	25-B
H	X	X	X	X	X	17-G	22-H	19-B	18-B
E	X	X	X	X	X	X	22-H	19-B	18-B
A	X	X	X	X	X	X	22-H	19-B	X
C	X	X	X	X	X	X	22-H	X	X

Shortest path: F-B-C