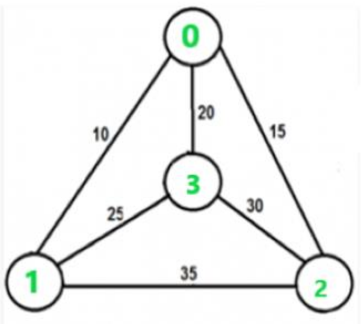
SAMPLE OUTPUT AND TIME COMPLEXITY

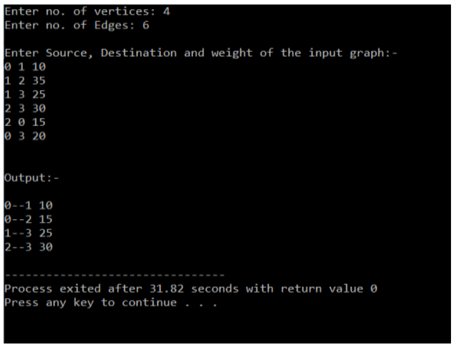
<u>CHEAPEST LINK ALGORITHM</u>

SAMPLE OUTPUT 1:

INPUT GRAPH

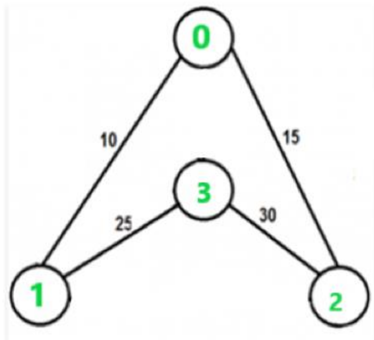


PROGRAM OUTPUT



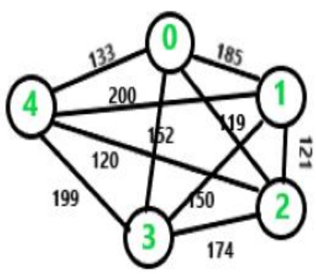
```
Enter no. of vertices: 4
Enter no. of Edges: 6

Enter Source, Destination and weight of the input graph:-
0 1 10
1 2 35
1 3 25
2 3 30
2 0 15
0 3 20


Output:-

0--1 10
0--2 15
1--3 25
2--3 30

--------------------------------
Process exited after 31.82 seconds with return value 0
Press any key to continue . . .
```

OUTPUT GRAPH

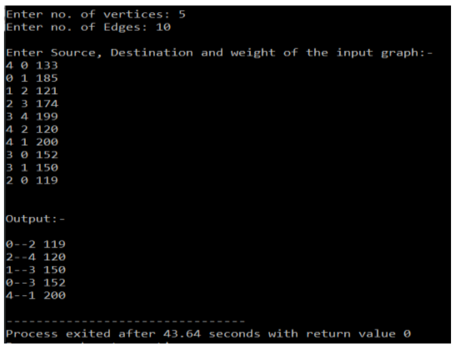


SAMPLE OUTPUT 2:

INPUT GRAPH

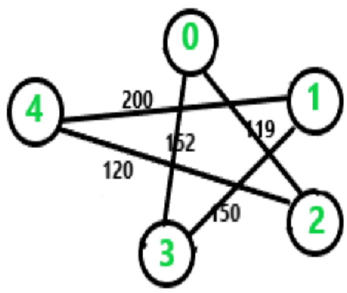


PROGRAM OUTPUT



```
Enter no. of vertices: 5
Enter no. of Edges: 10

Enter Source, Destination and weight of the input graph:-
4 0 133
0 1 185
1 2 121
2 3 174
3 4 199
4 2 120
4 1 200
3 0 152
3 1 150
2 0 119


Output:-

0--2 119
2--4 120
1--3 150
0--3 152
4--1 200

--------------------------------
Process exited after 43.64 seconds with return value 0
```

OUTPUT GRAPH



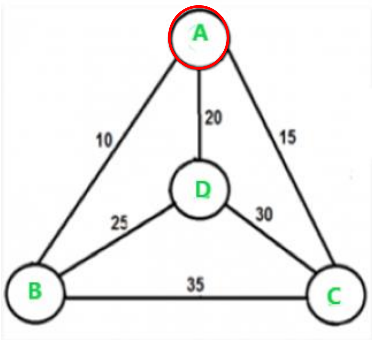
Time Complexity:

To traverse through all edges, it takes O(E) time. For union find algorithm to detect cycle, it takes around O(log n) time. Overall it takes, O(E)+O(log n). So we can say that Overall time complexity is O(E).

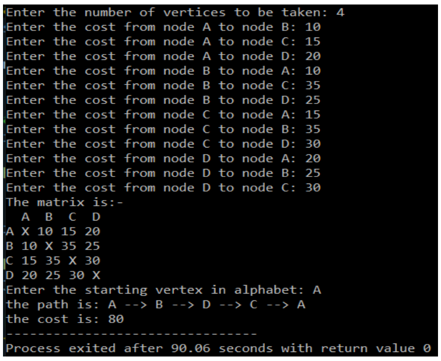
NEAREST NEIGHBOUR ALGORITHM

SAMPLE OUTPUT 1:

INPUT GRAPH

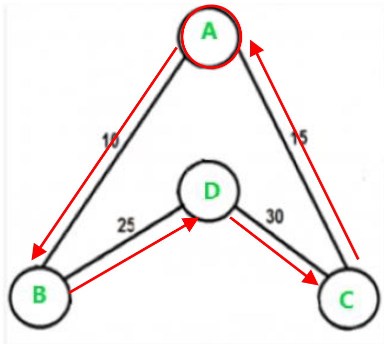


PROGRAM OUTPUT



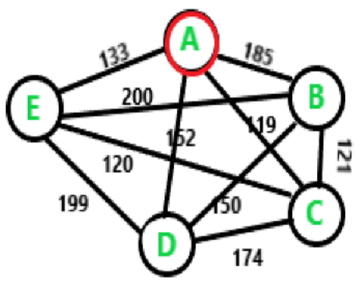
```
Enter the number of vertices to be taken: 4
Enter the cost from node A to node B: 10
Enter the cost from node A to node C: 15
Enter the cost from node A to node D: 20
Enter the cost from node B to node A: 10
Enter the cost from node B to node C: 35
Enter the cost from node B to node D: 25
Enter the cost from node C to node A: 15
Enter the cost from node C to node B: 35
Enter the cost from node C to node D: 30
Enter the cost from node D to node A: 20
Enter the cost from node D to node B: 25
Enter the cost from node D to node C: 30
The matrix is:-
  A  B  C  D
A X 10 15 20
B 10 X 35 25
C 15 35 X 30
D 20 25 30 X
Enter the starting vertex in alphabet: A
the path is: A --> B --> D --> C --> A
the cost is: 80
--------------------------------
Process exited after 90.06 seconds with return value 0
```

OUTPUT GRAPH

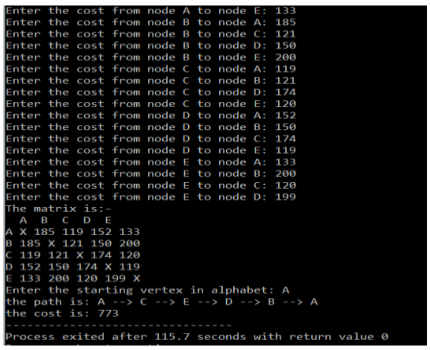


SAMPLE OUTPUT 2:

INPUT GRAPH

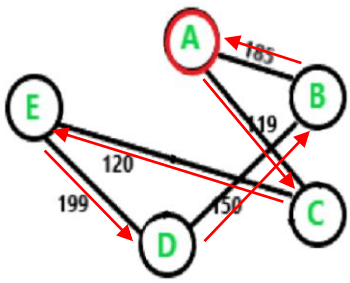


PROGRAM OUTPUT



```
Enter the cost from node A to node E: 133
Enter the cost from node B to node A: 185
Enter the cost from node B to node C: 121
Enter the cost from node B to node D: 150
Enter the cost from node B to node E: 200
Enter the cost from node C to node A: 119
Enter the cost from node C to node B: 121
Enter the cost from node C to node D: 174
Enter the cost from node C to node E: 120
Enter the cost from node D to node A: 152
Enter the cost from node D to node B: 150
Enter the cost from node D to node C: 174
Enter the cost from node D to node E: 119
Enter the cost from node E to node A: 133
Enter the cost from node E to node B: 200
Enter the cost from node E to node C: 120
Enter the cost from node E to node D: 199
The matrix is:-
  A  B  C  D  E
A X 185 119 152 133
B 185 X 121 150 200
C 119 121 X 174 120
D 152 150 174 X 119
E 133 200 120 199 X
Enter the starting vertex in alphabet: A
the path is: A --> C --> E --> D --> B --> A
the cost is: 773
--------------------------------
Process exited after 115.7 seconds with return value 0
```

OUTPUT GRAPH



Time Complexity:

To Create and work on adjacency matrix, it takes around $O(n^2)$. And for all other Operations, it takes $O(n)$ time. So, Overall, this algorithm takes $O(n^2)$.

COMPARISONS BETWEEN CHEAPEST LINK AND NEAREST NEIGHBOUR ALGORITHM

- **Execution Time**: We can have a look on the execution time of both the algorithms using two different graph. For 1st Graph, Using *Cheapest link algorithm*, it takes **31.82 sec**. while for the same graph using *Nearest Neighbour*, takes around **90.06 sec**. For 2nd Graph, Using *Cheapest link algorithm*, it takes **43.64 sec**. while for the same graph using *Nearest Neighbour*, takes around **115.7 sec**.

- **Time Complexity**: Time complexity for cheapest link algorithm is $O(E)$, where E refers to number of edges and time complexity for Nearest Neighbour algorithm is $O(n^2)$, where n refers to number of vertices.

- **Cost of Travel:** For the 1st graph, travelling cost is same for both algorithms. But for 2nd graph, travelling cost using Nearest Neighbour is $773 whereas using Cheapest Link algorithm, cost is $741. Hence, we can see that travelling cost using cheapest link algorithm is less than Nearest neighbour approach.