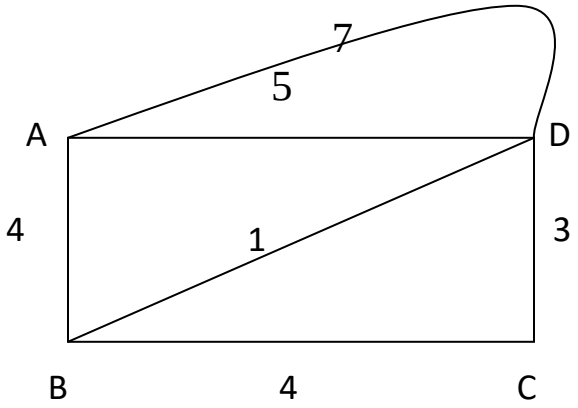
Kruskal's algorithm

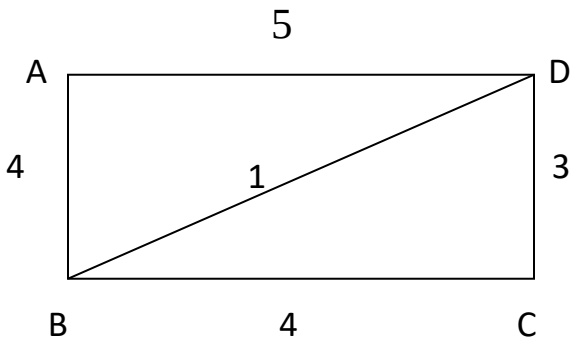
- T form a forest.
- The edge e added to T is always least-weight edge in the graph that connects two distinct trees of T.
- At the end of the algorithm T becomes a single tree.

e.g



Step-1 Remove all loops and parallel edges.

In this step remove all loops and parallel edges from given graph. In case of parallel edges, keep the one which has the least cost associated and remove all others.



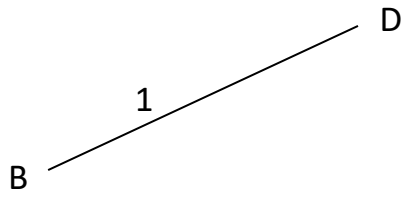
Step-2 Arrange all the edges in their increasing order of weight

The next step to create a set of edges and weight , and arrange them in an ascending order of weightage(cost).

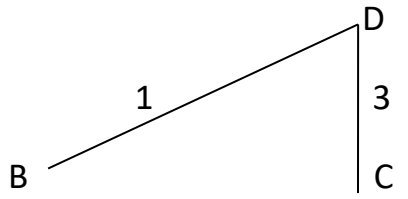
B-D	D-C	B-C	A-B	A-D
1	3	4	4	5

Step-3 Add the edges which Has the least weightage.

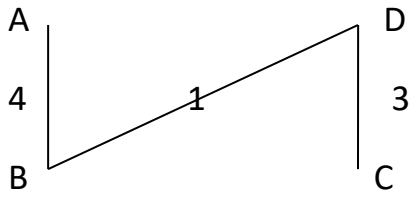
Now add the edge with least weight into the forest which is edge B-D with weight 1.



Now find next edge with minimum weight which is D-C with weight 3 add it to forest. After adding the edge resulting graph may be connect or disconnected Like a forest with multiple trees but at end of the procedure it will become a single tree.



Now in adding next edge with minimum weight there is ambiguity because both edge A-B and B-C have same weight (4) so you can choose any one of them which will not create a cycle in the forest. Hence A-B edge have to added because B-C create cycle in forest/graph.



According to the definition of spanning tree, all the vertices are covered. Now we have minimum cost spanning tree (cost- 8).

```
Procedure kruskal (G, cost).
Begin
        T: forest
        T= Null
        while |T|≤ n-1 & E≠ Null do
        choose an edge (v,w)€E of least weight
        delete (v,w) form E
        If (v,w) does not create a cycle in T
        then
                add (v,w) to T
        else
                discard (v,w);
        endif
        end while.
end
```

Implementation:

Choose the edge with the smallest weight:

- Use min-heap:
    - Get the min & read just the heap takes O (log e).
    - Construct the heap(build heap) takes O (e).

Be sure that the chosen edge does not create a cycle in the so far built forest, T:

- Use union-find: can used to find whether undirected graph contain cycle or not.
    - Once (u,v) is selected.
    - Check if Find (u) ≠ Find (v).(this will check if these two vertices are in same subset if they are then we discard the corresponding edge because it will create cycle in forest T).
    - Implementation of union and find on vertices take O(v) in worst case.

Total time complexity: O(e) + O(e log e) + O(v)

Build heap — read min edges from min heap one by one — Union- find(determine cycle)

Consider the Time complexity of **Kruskal's algorithm O (e log e)** because this process take large time among all the others.