(/add_quest

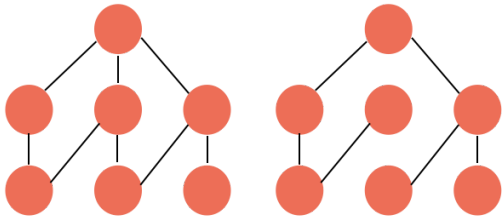
Minimum Spanning Tree (/course/algorithms-minimum-spanning-tree/)

Prim's Algorithm (/course/algorithms-prims-algorithm/)

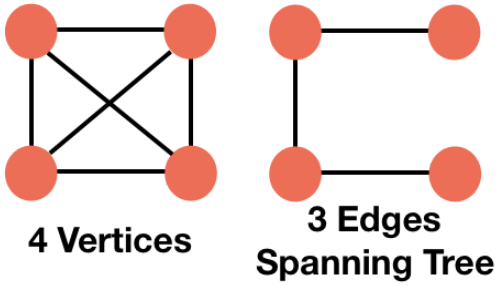
Minimum Spanning Tree | Kruskal's Algorithm

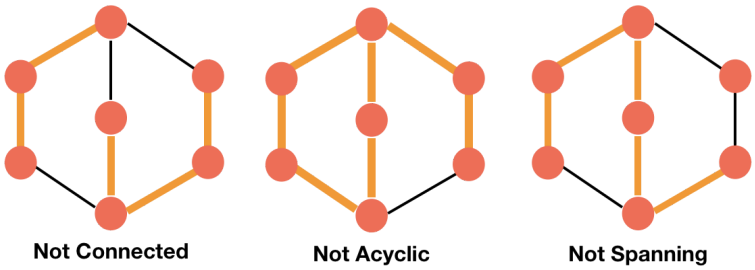
We have seen graphs and we know about trees (or you can read about them in Trees in Computer Science (https://www.codesdope.com/blog/article/trees-in-computer-science/) post). Thus, we can say that a tree is also a special type of connected graph which has no cycles and self-loops.

A **spanning tree** is a subgraph of a graph G which is connected and acyclic (since it is a tree) but has one more property that it includes all the vertices of the graph G (or spans all the vertices of the graph) with minimum possible edges.

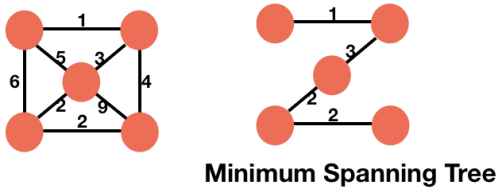


We can connect n vertices with a minimum of n-1 edges, so a spanning tree with n vertices has exactly n-1 edges.





A **minimum spanning tree** or **MST** is a spanning tree of an **undirected** and **weighted** graph such that the total weight of all the edges in the tree is minimum.

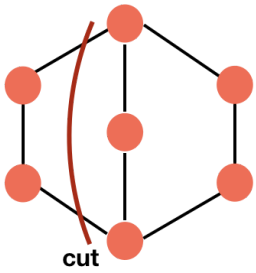


A minimum spanning tree is used in many practical applications. For example, think of providing electricity to n houses. To do this, we need to connect all these houses with wires. Suppose the wires are going along the roads of the locality, so the length of the roads are weights to the graph containing houses as vertices and roads as edges. Now, the minimum spanning tree would provide us the path to connect all these houses with the least length of wires and thus, minimizing the cost.

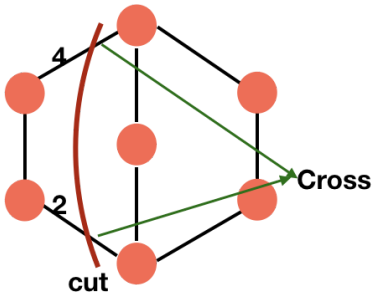
This is also needed in electric circuits, cluster analysis, network design, etc.

We are going to learn two algorithms - Kruskal's algorithm and Prim's algorithm in this course. But let's first go through some terms which are frequently used.

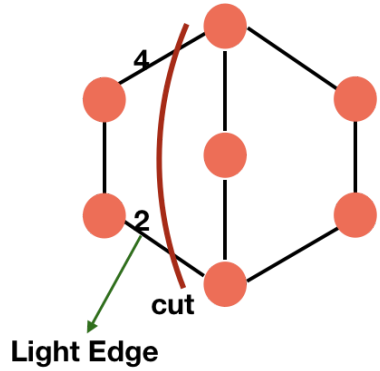
Cut → A cut is a partition in a graph which divides a graph into non-empty sets.



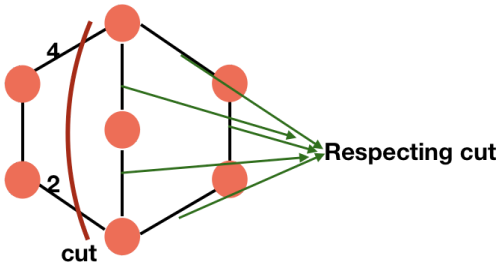
Cross → A edge is said to cross the cut if one of its ends is in the one partition and the other end is in another partition.



Light Edge → Among all the edges crossing a cut, the light edge is the one with the minimum weight.



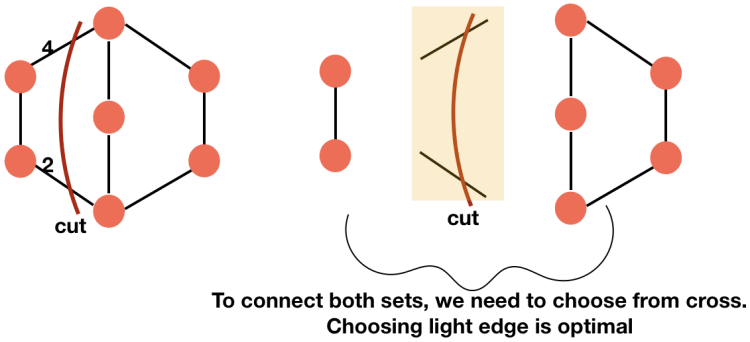
Respect → A cut respects a set of edges if no edge from the set crosses it.



Cut Property

Among all the crossing edges of a cut, the light edge is in the minimum spanning tree (or MST).

We can easily verify this result. Among the two sets of the partitions given to us, there must be an edge from all the crossing edges to connect both sets so that we can span all the edges and choosing the light edge will serve the purpose of a minimum spanning tree.



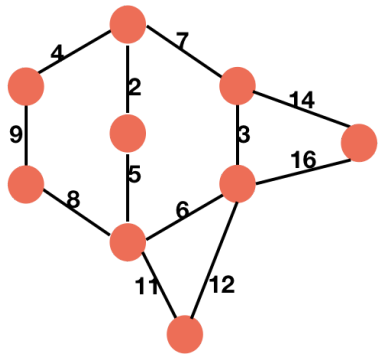
To connect both sets, we need to choose from cross.
Choosing light edge is optimal

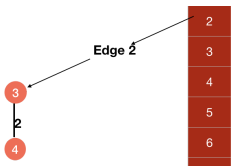
Let's start learning the Kruskal's algorithm to get the minimum spanning tree from a graph. We will learn Prim's algorithm in the next chapter.

Kruskal's Algorithm

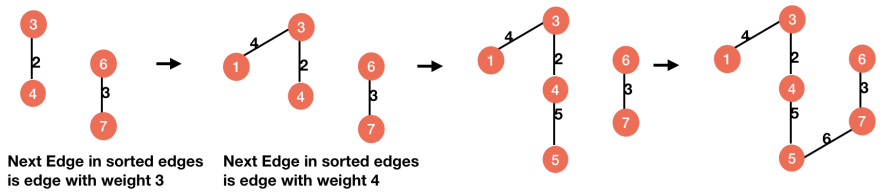
In Kruskal's algorithm, we greedily choose the edge with minimum weight (greedy technique) such that no cycle is formed. Thus, we first sort the edges with their weights and then iterate over these and choose edges from it such that no cycle is formed.

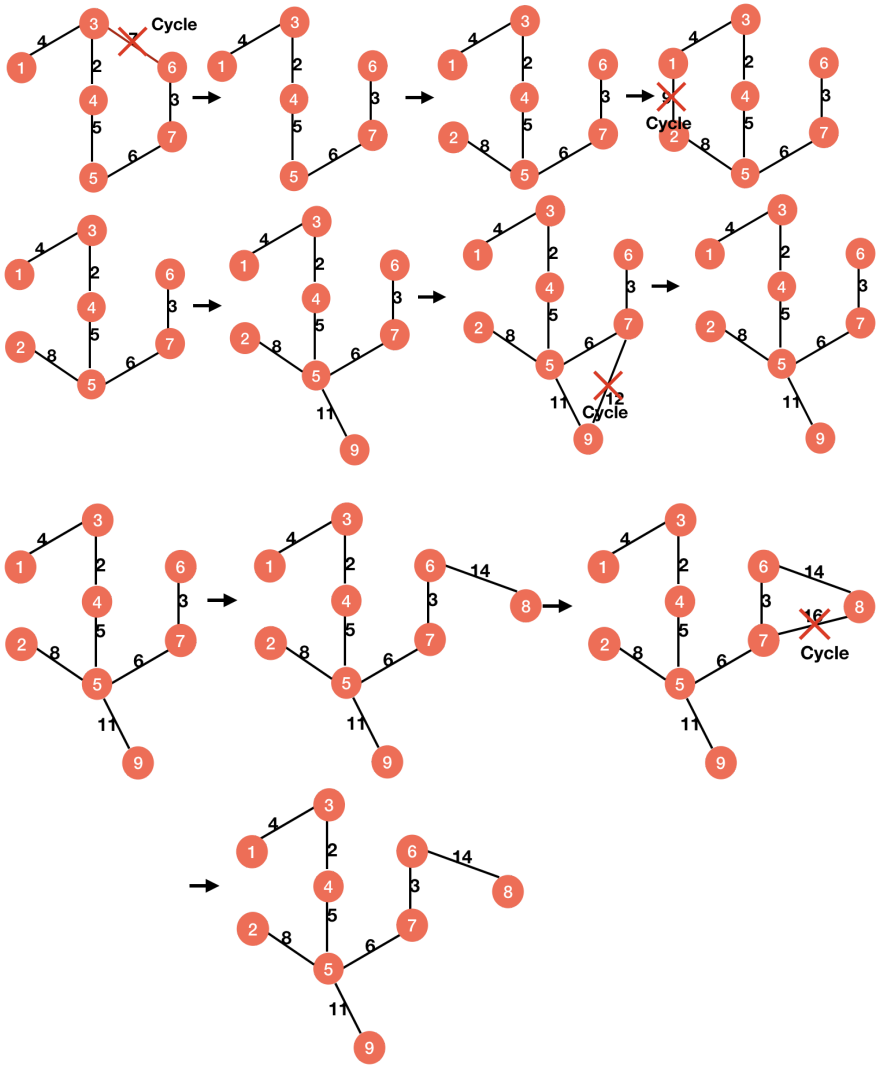
Let's take an example to understand this procedure.

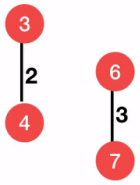


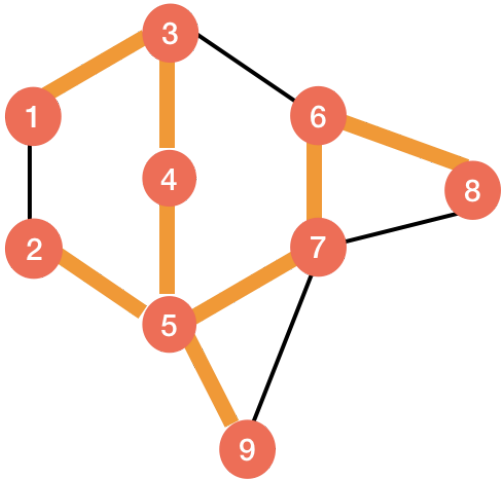


Sorted Edges







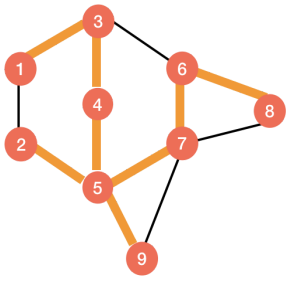


Correctness of Kruskal's Algorithm

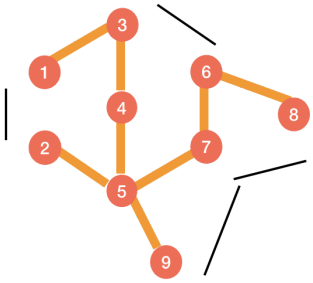
Let's first check if the Kruskal's algorithm is giving a spanning tree or not. The tree we are getting is acyclic because in the entire algorithm, we are avoiding cycles.

The tree is also spanning all the vertices. Let's assume that the tree is not spanning all the edges and a vertex v is not included in the tree. Our algorithm sorts and checks all the edges, so it must have checked the incident edges of the vertex v also. Now, the only reason we couldn't pick the vertex v would be that the vertex v was forming a cycle which means that the vertex v is already in the tree and thus the tree spans all the vertices.

The last property we need to check to prove that the algorithm yields a spanning tree is that the tree we are getting is connected.



Connected

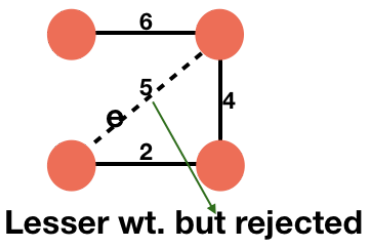


Any of these black edges will form a cycle.
It means nodes are already connected

The graph we are provided with is a connected one, so the trees we are getting must be connected in the graph by some edge(s). Taking any edge which is not in the tree will form a cycle in the tree. It means the nodes are already connected. We can't choose any new edge from the graph which is not in the tree which will connect two unconnected nodes. It means the tree is already connected.

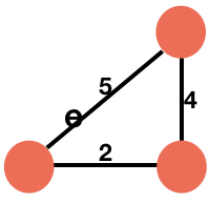
Since the tree doesn't have any cycle, it spans over all vertices and is connected, so it is a spanning tree. Now, we just have to prove that this is an optimal one to prove that it is a minimal spanning tree.

Let T be the spanning tree we got from our algorithm and T^* is the minimum spanning tree. If T and T^* are not equal, then there must be an edge in the tree T^* from the edges which we have rejected while forming T (because they were forming loops) because of their lesser weights.



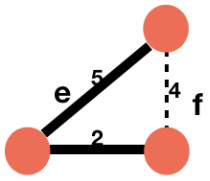
Lesser wt. but rejected

Let say this edge is e, and if we have taken this edge we would have formed a cycle. Also because our algorithm takes the edges with fewer weights first, so all the edges of this cycle must be less than the weight of e.



2 and 4 are less than e

Let's say f is some edge of this cycle which is not in T^* because T^* has the edge e and doesn't for any cycle (as it is MST). Also the edge f is appearing before the edge e in the cycle, so its weight must be less than that of e (our algorithm adds edges with less weight first).



5 and 2 are in T^*

Let's consider a tree T_2 which includes all the edges of the tree T except the edge f and includes e instead of it. Thus, T_2 is closer to the tree T^* because T^* also has the edge e instead of f.

The weight of the tree T $\leq$ T_2 because the weight of f is less than the weight of e.

We can repeat this process with the tree T_2 to get a tree T_3 such that, the weight of $T_2 \leq T_3$. Similarly, we can do for other trees such that,

$$wt(T) \leq W(T_2) \leq wt(T_3) \ldots \leq wt(T^*)$$

Since T^* is a minimum spanning tree, this condition will only be possible if all these weights are equal. Thus, $wt(T) = wt(T^*)$. So, T is a minimum spanning tree.

Now we know how Kruskal's algorithm works and we have also seen the proof of its correctness. Let's look at the code to implement the algorithm in a program.

Code for Kruskal's Algorithm

We can implement the Kruskal's algorithms in many different ways. We need to start by sorting all the edges according to their edges and then we just have to iterate over them and construct our MST by checking for cycles in MST before including them.

```
KRUSKAL(G)
  MST = NULL

  sort the edges acc. to their weights

  for edge in G.E
    if end of edge disconnected in MST // checking for cycle
      Include edge in MST

  return MST
```

> The present is theirs; the future, for which I really worked, is mine.

- Nikola Tesla

PREV (/course/algorithms-dfs/) (/course/algorithms-prims-algorithm/) NEXT

New Questions

setting up an ide for mac.. **- Cpp**

(/discussion/setting-up-an-ide-for-mac)

Please fill the blanks and help me am stuck **- Java**

(/discussion/fill-the-blanks-and-help-me-am-stuck)

Time complexity of the Python Function **- Python**

(/discussion/time-complexity-of-the-python-function)

How I Can find The Best Target Coupons & Latest Deals Offers? **- Other**

(/discussion/how-i-can-find-the-best-target-coupons-latest-deal)

To Calculate salaries **- Java**

(/discussion/to-calculate-salaries)

How to write a c++ program that will declares a two dimensional array say Char My element [4][5], prompt the user to initialize and display the elemen **- C**

(/discussion/how-to-write-a-c-program-that-will-declares-a-two-)

What is the difference between a local variable and an instance variable? **- Java**

(/discussion/what-is-the-difference-between-a-local-variable-an)

Ask Yours

(/add_question/)