Kruskal's Algorithm

Indian Institute of Information Technology, Allahabad

Design and Analysis of Algorithms Assignment : 6

Swaraj Bhosle IIT2019024

Ritesh Raj IIT2019025

Utkarsh Garg IIT2019026

***Abstract*—In this paper, we are devising an algorithm to find the Minimum spanning tree by Kruskal's Algorithm. This paper also contains the algorithm's time and space complexity analysis.**

I. Introduction

Given a connected and undirected graph, a spanning tree of that graph is a subgraph that is a tree and connects all the vertices together. A single graph can have many different spanning trees. A minimum spanning tree (MST) or minimum weight spanning tree for a weighted, connected and undirected graph is a spanning tree with weight less than or equal to the weight of every other spanning tree. The weight of a spanning tree is the sum of weights given to each edge of the spanning tree.
A minimum spanning tree has $(V-1)$ edges where V is the number of vertices in the given graph.
Kruskal's algorithm to find the minimum cost spanning tree uses the greedy approach. This algorithm treats the graph as a forest and every node it has as an individual tree. A tree connects to another only and only if, it has the least cost among all available options and does not violate MST properties.

This report further contains -

- Algorithm Description.
- Algorithm And Analysis.
- Time Complexity Analysis
- Conclusion

II. Algorithmic Design

Kruskal's algorithm forms the forest where each edge acts vertex acts as a single tree, the edges are sorted in the increasing order of their costs and included in the set T of selected edges if the edges in the selected tree do not form a cycle and also after the inclusion of edges. Kruskal's algorithm also works on the disconnected graphs.

Steps in kruskal's Algorithm

1) all the edges in the increasing order of their cost.
2) Choose the edge which has the smallest cost and check for the cycle. If the edge forms the cycle with the spanning tree discard that edge else include the edge in the spanning tree.
3) Repeat step 2 until all the vertices are included in the spanning tree.

Algorithm 1: Kruskal Algorithm Pseudo Code

```
Input: undirected graph G = (V, E)
Output: : minimum spanning tree T
Function Main():
    T = 0;
    T (the final spanning tree) is defined to be the empty set;
    For each vertex v of G do
    Make empty set out of v
    Sort the edges of graph 'G' in the increasing order of their weight w
    For each edge (u, v) from the sorted edges do
    If u and v belong to different sets then
    Add (u,v) to T;
    Union (u, v);
    Return T;
```

III. Algorithm Analysis

A. *Time Complexity*

The time complexity of this algorithm is $\mathcal{O}(ElogE)$ or $\mathcal{O}(ElogV)$. Sorting of edges takes $\mathcal{O}(ELogE)$ time. After sorting, we iterate through all edges and apply find-union algorithm. The find and union operations can take atmost $\mathcal{O}(LogV)$ time. So overall complexity is $\mathcal{O}(ELogE + ELogV)$ time. The value of E can be atmost $\mathcal{O}(V2)$, so $\mathcal{O}(LogV)$ are $\mathcal{O}(LogE)$ same. Therefore, overall time complexity is $\mathcal{O}(ElogE)$ or $\mathcal{O}(ElogV)$.

B. *Space complexity*

The space complexity is $\mathcal{O}(E+V)$, since Disjoint Set Data Structure takes $\mathcal{O}(V)$ space to keep track of the roots of all the vertices and another $O(E)$ space to store all edges in sorted manner.

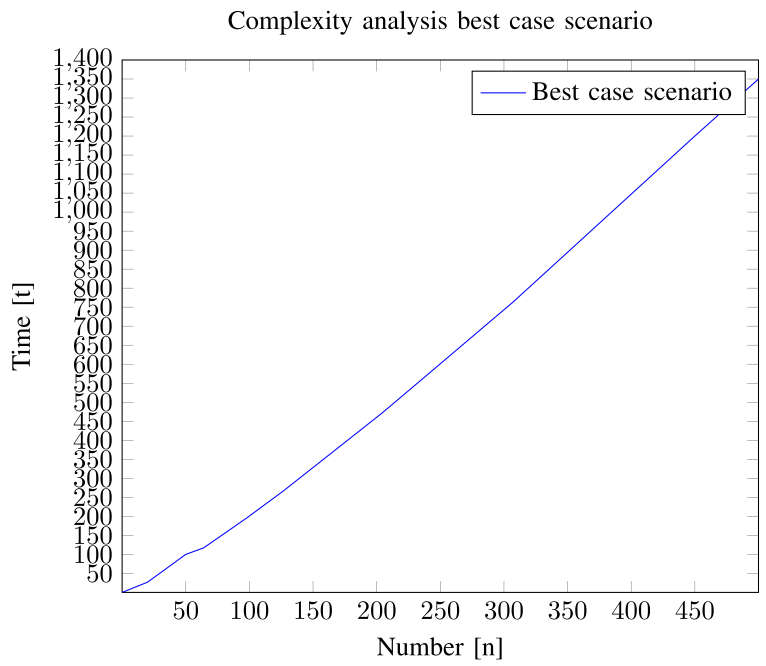
IV. Parallel Algorithm

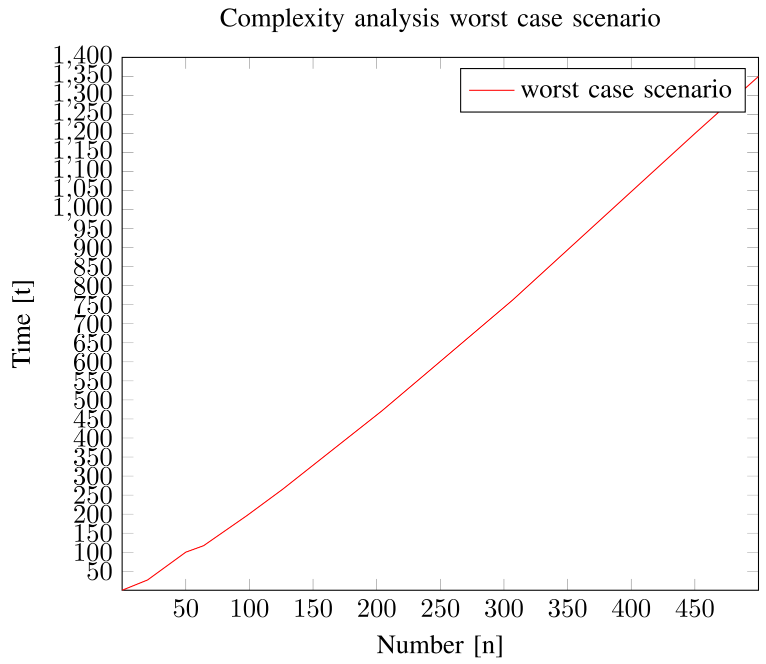
Kruskal's algorithm is inherently sequential and hard to parallelize. It is, however, possible to perform the initial sorting of the edges in parallel or, alternatively, to use a parallel implementation of a binary heap to extract the minimum-weight edge in every iteration.As parallel sorting is possible in time $\mathcal{O}(N)$ on $\mathcal{O}(logN)$ processors, the runtime of Kruskal's algorithm can be reduced to $\mathcal{O}(E\alpha(V))$, where α again is the inverse of the single-valued Ackermann function.

V. Experimental Study

It falls under a class of algorithms called greedy algorithms that find the local optimum in the hopes of finding a global optimum.

Total complexity : $Sortingtime + edgetraversel * (Mergingoperation + findingcomponent)$





VI. Conclusion

In this paper we have discussed about well-known algorithms to construct and find the minimum spanning tree of an undirected and connected graph: Kruskal's algorithm. It is observed that the time complexity of kruskal's algorithm is $\mathcal{O}(ElogV)$ (where E is number of edges and V is number of vertices). The Kruskal's algorithm is simple to implement. kruskal's algorithms is one of the algorithm mostly used for solving MST problem among the other algorithms. We are further looking to reduce the time complexity of MST algorithms.

References

[1] https://en.wikipedia.org/wiki/Kruskal_algorithm
[2] https://cp-algorithms.com/graph/mst_kruskal.html
[3] http://www.cs.rpi.edu/musser/gp/algorithm-concepts/prim-screen.pdf