KRUSKAL'S MST

COP4533 PROJECT

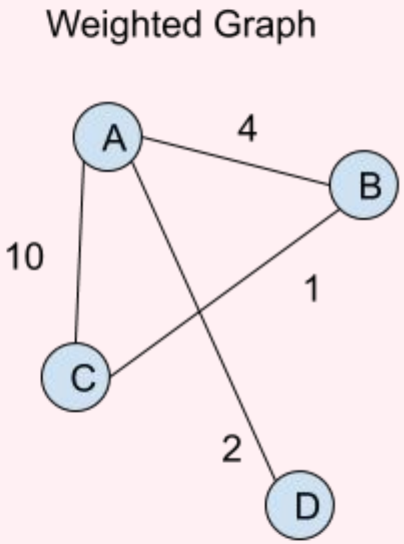
Lydia Chung and Lauren Nunag

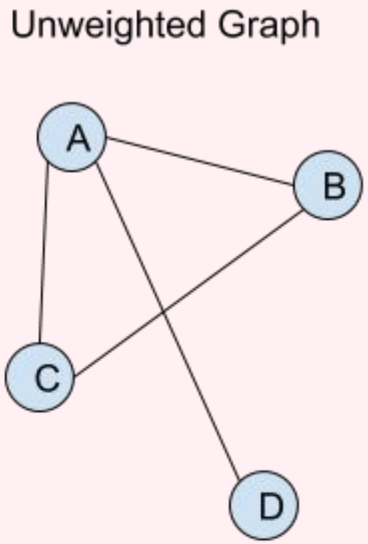
ALGORITHM OVERVIEW

Background

Graph: A structure comprising vertices (nodes) connected by edges (lines).

Weighted Graph: A graph where each edge has an associated weight (e.g., cost, distance, time).



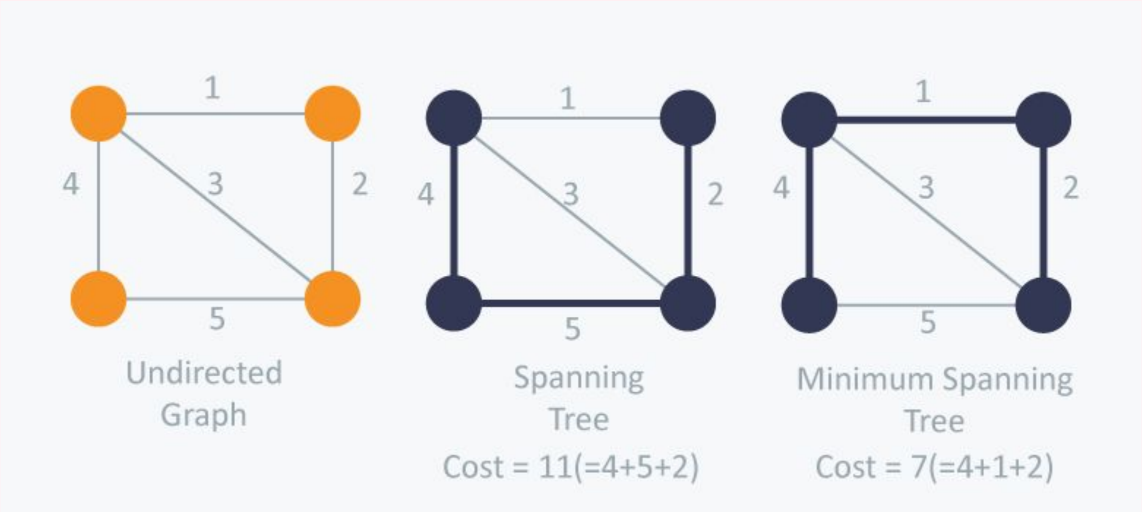


ALGORITHM OVERVIEW

Background

Spanning Tree: A subgraph that connects all vertices with the minimum number of edges and no cycles.

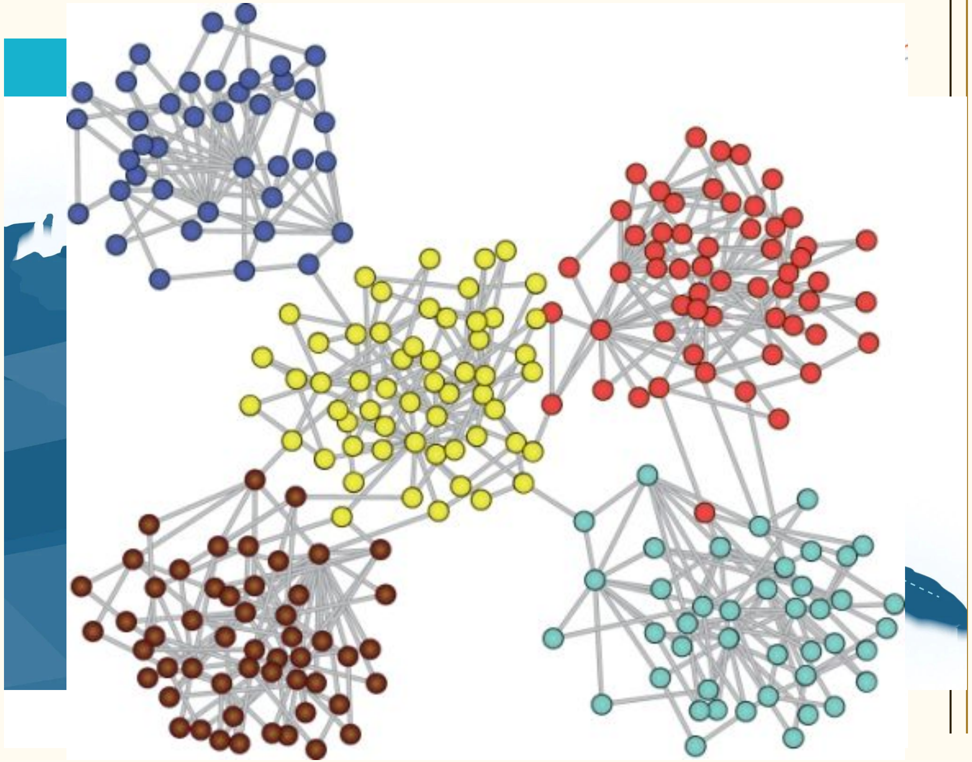
Minimum Spanning Tree: A spanning tree with the lowest possible total edge weight.



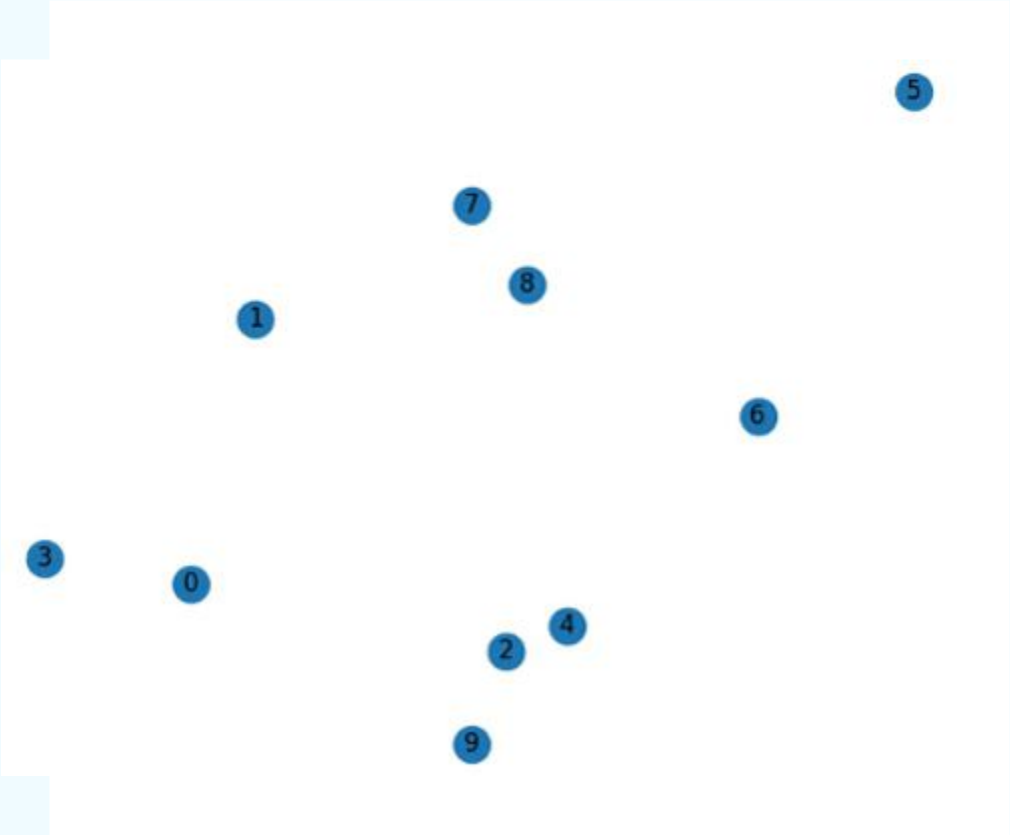
⬆️ This is what Kruskal's Algorithm finds.

Purpose & Real World Examples

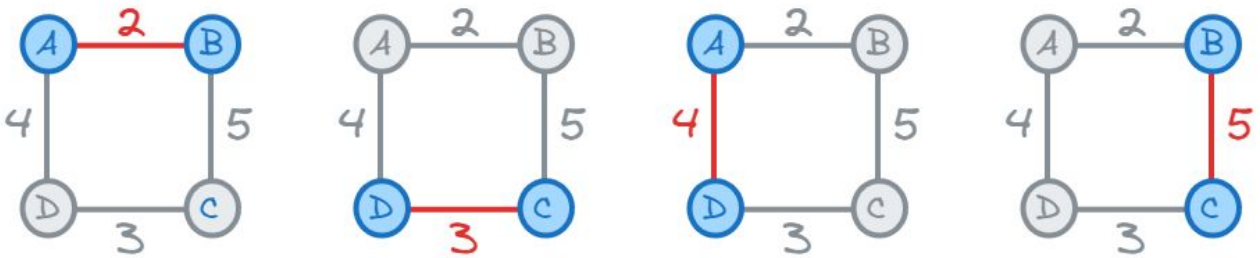
Find the cheapest way to connect all nodes in a network without creating any loops.



How Kruskal’s Algorithm works

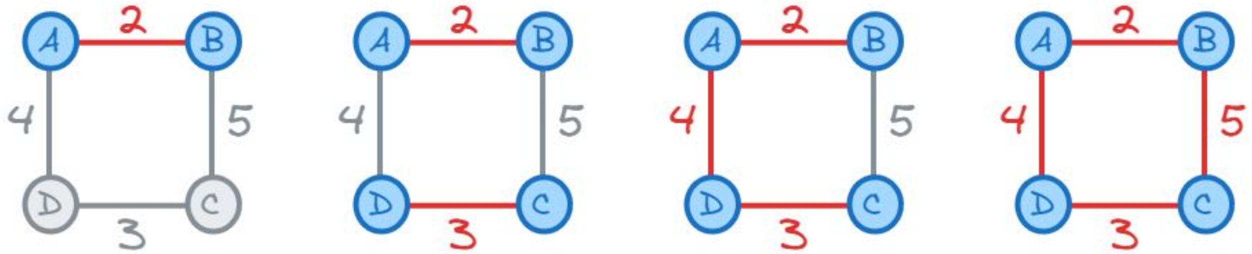


1. Sort Edges by Weight



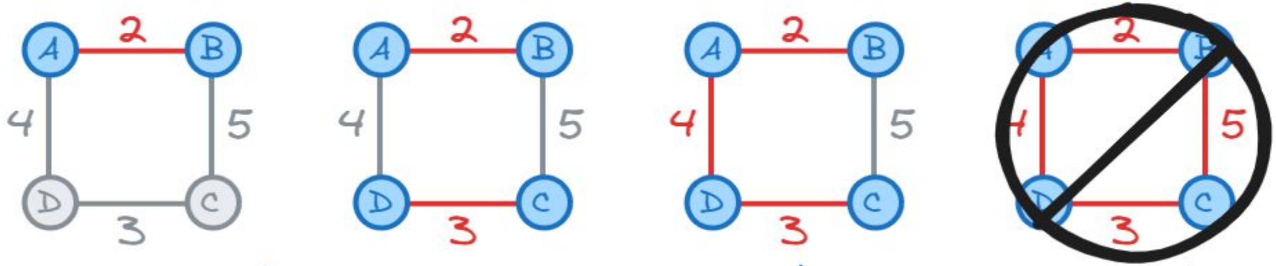
We'll create an array which will hold a arrays of each edge in our graph, then sort the array on edge weight.

2. Add Edges While Avoiding Cycles



We then construct the MST using the edges, checking to ensure we don't create any cycles (loops).

3. Terminate When MST is Complete



When the edges we construct are 1 less than the number of vertices, we have completed the MST!

Implementation Details

Code Review

Challenges Faced and How We Addressed Them:

- Lauren had to learn how Kruskal’s MST worked again
- Lydia had all the DSU code written in Java already, but she had to translate it into C++ for the sake of this implementation

PERFORMANCE ANALYSIS

RUNNING TIME

Background

Kruskal's Algorithm runs in O(E log E) time, where E is the number of edges. Sorting the edges dominates the cost. The Disjoint Set Union (DSU) operations run in nearly constant time with path compression and union by rank.

1

x_1	y_1
10	.012
14	.011
100	.012
1000	.015
10000	.031
15000	.041

2

$y_1 \sim a x_1 \log x_1 + b$

REGRESSION PARAMETERS

$a = 4.64763 \times 10^{-7}$ $b = 0.0121328$

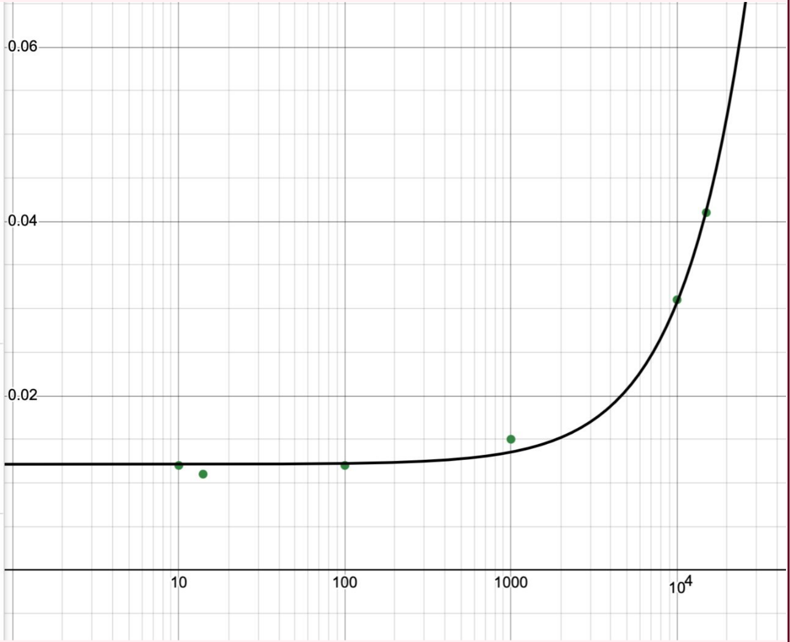
STATISTICS

$R^2 = 0.9954$

RESIDUALS

e_1 plot

3



SPACE COMPLEXITY

Memory Usage

The space complexity is O(V + E) because we store all the edges and use arrays in DSU of size V.

x_2	y_2
20	1049344
23	1049344
150	1196800
1500	1327936
7500	1524544
15000	2016064
25000	2343872

Linear Regression

EQUATION

$y = 50.89289x + 1143475.7$

STATISTICS

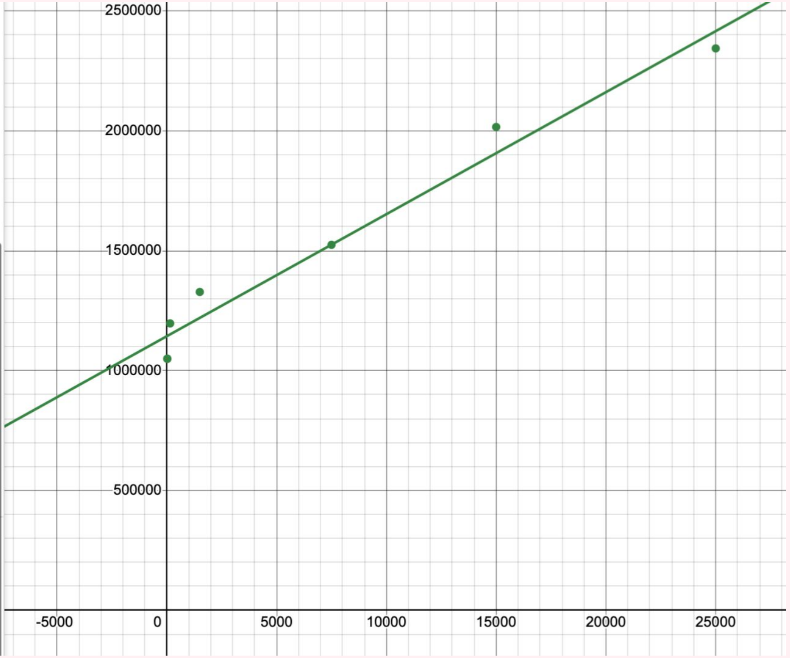
$R^2 = 0.9675$

$r = 0.9836$

RESIDUALS

e_2 plot

powered by desmos



REFLECTION

Strengths, Limitations, Improvements

1

Strengths

Efficient for sparse graphs. Uses sorting and DSU to build MST quickly with low time complexity.

2

Limitations

Slower on dense graphs due to edge sorting. Memory grows with number of edges.

3

Improvements

Added path compression & union by rank to speed up DSU. File I/O used for flexible testing.

THANK

YOU!