Analytical report

Summary

The table below summarizes, for each dataset, vertices, edges, algorithm, total cost, operation count, and execution time based on your generated outputs.

Data case	Vertices	Edges	Algorithm	Total cost	Operations	Time (ms)
output_1.json	5	9	Kruskal	10	34	0.26
output_2.json	6	10	Kruskal	14	41	0.21
output_3.json	7	11	Kruskal	20	48	0.15
output_4.json	8	13	Kruskal	23	63	0.24
output_5.json	9	15	Kruskal	24	73	0.19

Each minimum spanning tree has exactly

$$N - 1$$

edges and achieves the minimum possible total cost for a connected, weighted, undirected graph, which is consistent with the MST definition and the assignment's expectations.

Prim vs Kruskal

- Kruskal sorts all edges by weight and adds them if they do not form a cycle, relying on a Disjoint Set Union (Union-Find) structure, which typically yields complexity
  $$O(M \log M)$$
  and works naturally with edge-list inputs.
- Prim grows a single tree from an arbitrary start vertex using a priority queue over frontier edges, reaching
  $$O(M \log N)$$
  with a binary heap and often performing well on dense graphs stored as adjacency structures.
- In practice, Kruskal is convenient for sparse graphs and when the input is already a list of edges, while Prim is convenient for dense graphs and adjacency-based storage with efficient decrease-key behavior.
- Both algorithms must return identical MST total cost on the same connected graph, although the selected edge sets may differ due to tie-breaking on equal weights, as required by the assignment.

Conclusions

- Prefer Kruskal for sparse graphs and edge-list representations, as the global sort coupled with efficient Union-Find tends to be simple to implement and fast when
$$M$$
is close to
$$N$$
.
- Prefer Prim for dense graphs and adjacency-list/matrix representations, where maintaining a frontier with a binary heap can be advantageous as
$$M$$
approaches
$$N^2$$
.
- The measured operation counts increase monotonically across your five cases as graph size grows, which aligns with expectations from sorting more edges and performing more find/union operations in Kruskal.

Link to GitHub Repo