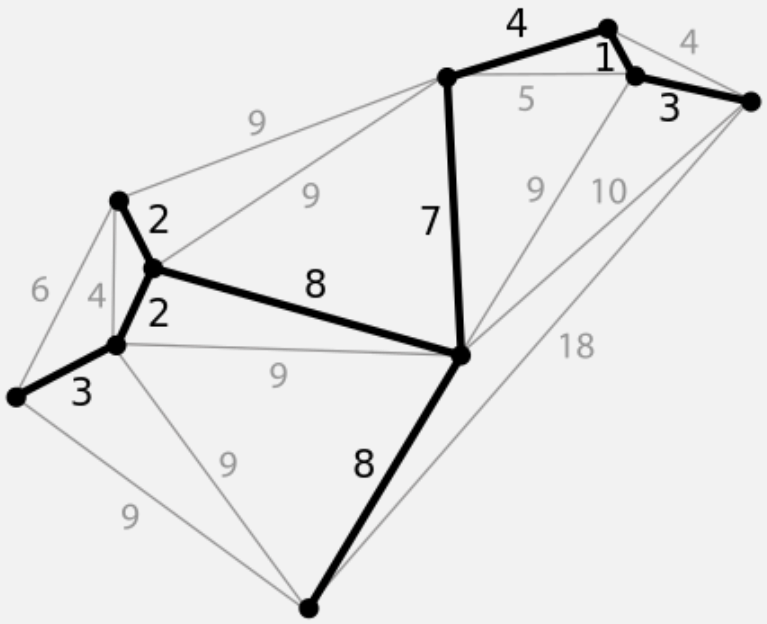
MINIMUM SPANNING TREES

Tian Cilliers | Training Camp 3 | 9-10 March 2019

DEFINITION

- A **minimum spanning tree** (**MST**) is a subset of the edges of a connected, edge-weighted undirected graph that connects all the vertices together, without any cycles and with the minimum possible total edge weight.



APPLICATIONS

- Approximating **Travelling Salesman Problem**, Perfect Matching, Single-Terminal Maximum Flow
- Communications **network design**, electricity grid or transportation system optimization
- Cluster analysis
- Taxonomy
- A lot of other weird stuff I don't understand

DETERMINING THE MST

SAMPLE PROBLEM

- Given some weighted, undirected graph. Determine the minimum sum of edge weights necessary to connect all nodes.

POSSIBLE SOLUTION

- Generate all possible spanning trees (connecting all nodes), calculate the sum of their edge weights, and choose their minimum.

PROBLEMS WITH SOLUTION

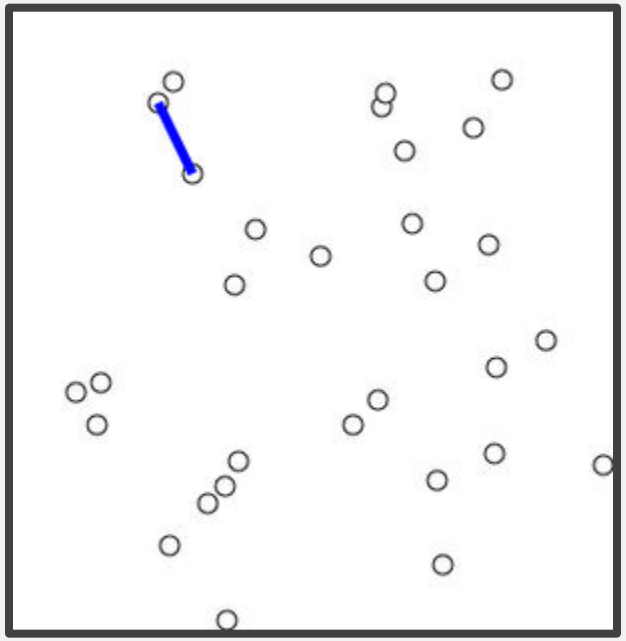
- Obviously, this solution runs in $O(too\ slow)$.
- Is there a faster, greedy way to determine the MST?

PRIM'S ALGORITHM

ALGORITHM

- Start with a **single vertex** from the graph.
- While all vertices has not been processed, do the following:
  - Retrieve the edge with the **smallest weight** connecting an **unprocessed vertex** to the graph.
  - Connect the vertex to the graph with the edge.
  - Process all other edges connected to the added vertex for use later.

VISUALIZATION



IMPLEMENTATION

- Keep a **priority queue** of the vertices considered, sorted by the weight of the edge connecting them.
- Keep a **visited array** to check if a particular node has been visited yet.
- The rest of the algorithm follows logically from here.
- The total runtime is $O(m \log m)$ as well, since all edges are processed, and the efficiency of a priority queue is logarithmic.

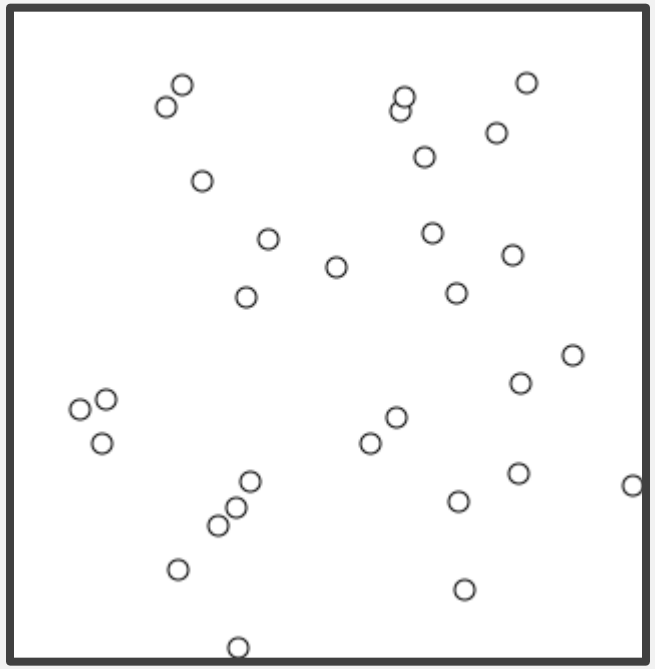
WHY BRUCE IS A GOD

KRUSKAL'S ALGORITHM

ALGORITHM

- Start with a **separate tree for each vertex**, consisting only of that vertex.
- While all edges has not been processed and the graph is not spanning, do the following:
  - Retrieve the **unprocessed edge with the lowest weight.**
  - If it **connects two unconnected trees** in the graph, add it to the graph and join the trees.

VISUALIZATION



IMPLEMENTATION

- Finding minimum unprocessed edge can be done by **sorting edges** at the start.
- Finding if an edge connects two trees can be done using the **Disjoint-Set** data structure.
- The rest of the implementation is trivial and is left as an exercise to the reader.
- The total time complexity is $O(m \log m)$ since it is limited by the time required to sort the array.

EXAMPLE PROBLEM

TRAIL MAINTENANCE (PROBLEM)

- Cows want to maintain certain trails between fields.
- The total length of trails maintained must be minimized.
- They start off with no trails, and after each week they discover a new path.
- Given the trails they discover each week, you need to determine the total distance of the trails after each week.

TRAIL MAINTENANCE (SOLUTION)

- Recalculate the MST after each week using all previous paths (50%)
- Do the same, but only look at the best path between two fields (60%)
- Do the same, but only look at the previous MST and the new path (100%)

QUESTIONS?