Minimum Spanning Tree Algorithms
CS 375 Final Project

Tim Hung and Samuel David Bravo

Binghamton University

May 3, 2016

Overview

Overview
Prim's Algorithm
Kruskal's Algorithm
Results
Summary

Minimum Spanning Trees
Approach
Problem Statement

Minimum Spanning Trees

A minimum spanning tree connects all the vertices in a graph together into a tree with the lightest weight possible.

Approach

Algorithms:

- ▶ Kruskal's
- ▶ Prim's

Implementations:

- ▶ Adjacency List
- ▶ Adjacency Matrix

Overview
Prim's Algorithm
Kruskal's Algorithm
Results
Summary

Minimum Spanning Trees
Approach
Problem Statement

Problem Statement

How do Prim's and Kruskal's algorithms handle graphs of different densities?

Do they depend on whether the graph is implemented as an adjacency list or an adjacency matrix?

Overview
Prim's Algorithm
Kruskal's Algorithm
Results
Summary

Algorithm
Implementation
Analysis

Prim's Algorithm

Overview
Prim's Algorithm
Kruskal's Algorithm
Results
Summary

Algorithm
Implementation
Analysis

Main Idea

Expand the tree by adding the lightest connecting edge.

Overview
Prim's Algorithm
Kruskal's Algorithm
Results
Summary

Algorithm
Implementation
Analysis

Pseudocode

extemely neat and clear fits in one slide pseudocode.

Implementation Details

Overview
Prim's Algorithm
Kruskal's Algorithm
Results
Summary

Algorithm
Implementation
Analysis

Analysis of Prim's

interesting features, time complexity, why?

Kruskal's Algorithm

Overview
Prim's Algorithm
Kruskal's Algorithm
Results
Summary

Algorithm
Implementation
Analysis

Main Idea

- ▶ Separate vertices into disjoint sets
- ▶ Reorder all edges by smallest weight first
- ▶ Loop through edges, add it to tree if its vertices are disjoint

Overview
Prim's Algorithm
Kruskal's Algorithm
Results
Summary

Algorithm
Implementation
Analysis

Pseudocode

```
EdgeContainer all_edges = graph.sorted_edges()

VerticesSet set = disjoint_set(v.size)
EdgeContainer MST = empty

for (Edge e : all_edges)
    v1 = e.source;
    v2 = e.destination
    if (set.are_vertices_disjoint(v1,v2))
        mst += e;
        set.join(v1,v2)
```

Overview
Prim's Algorithm
Kruskal's Algorithm
Results
Summary

Algorithm
Implementation
Analysis

Key Data Structures

- A vector of all the sorted edges
- A vector for holding edges included to minimum spanning tree
- A vector representing disjoint sets

Overview
Prim's Algorithm
Kruskal's Algorithm
Results
Summary

Algorithm
Implementation
Analysis

Functions

- ReadFromAdjacencyList or ReadFromAdjacencyMatrix
- SortEdgesSmallestFirst, size $|E| \log ||E|$
- CreateDisjointSets, size $|V|$
- JoinSets, size $|V|$
- AreSetsDisjoint, size $O(1)$

Overview
Prim's Algorithm
Kruskal's Algorithm
Results
Summary

Algorithm
Implementation
Analysis

Analysis of Kruskal's

- Only uses graph representation for retrieving edges
- Sorting the edges is of size $|E| \log |E|$
- Meanwhile, looping through edges is of size $|E|$
- The time complexity is carried by the sort, $|E| \log |E|$
- Kruskal's works well with sparse graphs

Results

Demonstration

Overview
Prim's Algorithm
Kruskal's Algorithm
Results
Summary

Demo
Data Set
Results
Limitations

Our Data

Describe the dataset that you used to test the algorithm. How did you generate it? What characteristics does it have, and why? What did you decide to vary in the input set, and why?

Overview
Prim's Algorithm
Kruskal's Algorithm
Results
Summary

Demo
Data Set
Results
Limitations

Results

What did you learn from testing your algorithm?

Overview
Prim's Algorithm
Kruskal's Algorithm
Results
Summary

Demo
Data Set
Results
Limitations

Limitations and Future Work

What limitations does your project currently exhibit? If you had
another month, what could you improve? What additional tests
would you run?

- ► Kruskal's can finish early by checking if there is only 1 set.

Summary

Recap

This is a recap of what we have talked about.

Questions

Thank you.
Any questions?