Report

CSCI203 A3

Asjad Athick

4970512

mama158@uowmail.edu.au

Introduction

Graphs are an essential data structure for the representation of data in mathematics and computer science. A minimum spanning tree is a tree that is:

A vertex (plural: vertices) is a point on the graph (node)

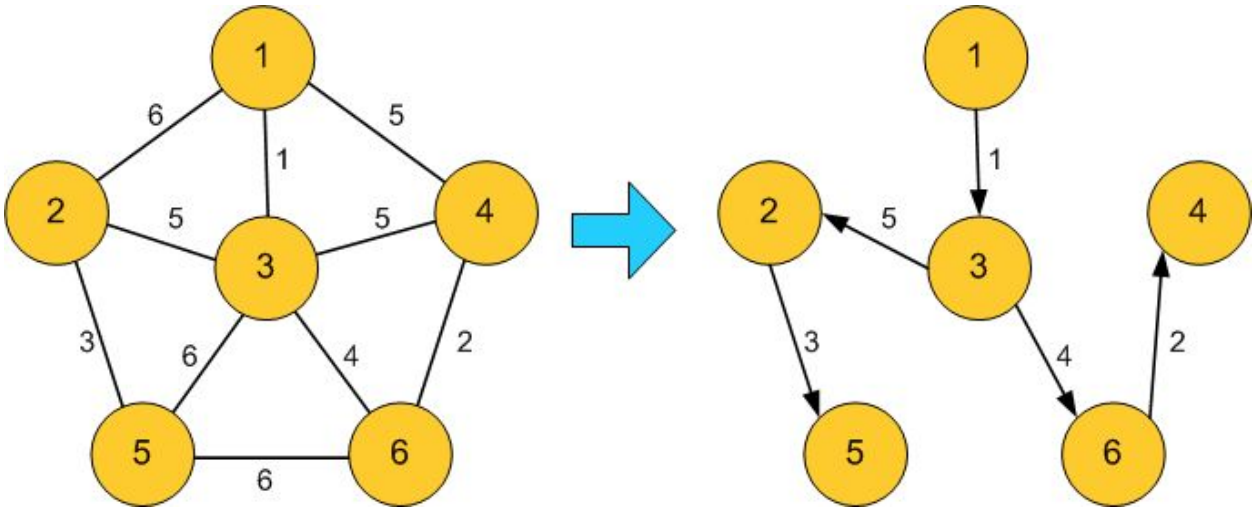
An edge connects 2 vertices

- Connected (there is at least one path between each pair of vertices)
- Acyclic (no loops in the graph)
- Includes all vertices of the graph
- And has the lowest weight/cost (minimum)

By these points, a minimum spanning tree will always have (N-1) edges, where N is the number of vertices

Edge weight is the cost associated with using the edge in an application.

[1]



Source: http://www.xatlantis.ch/images/education/graph2_example.png

Applications of Minimum Spanning Trees

MSTs can be used in various applications, where data points can be represented as nodes, and the dataset can be represented as a graph.

- Network Design

Crucial in designing digital or other kinds of networks, where nodes/vertices need to be connected to each other in the most efficient way possible. Each connection might have a cost or delay associated with it, and the network would need to operate as efficiently as possible. The creation of MSTs in such cases to determine the most optimal network is required.

- Travelling salesman / NP Complete Problems analysis

Determining paths across a graph to cover all edges, in the shorted time possible. This can be applied to other kinds of NP complete problems too.

- Cluster Analysis

Algorithms for detecting clusters with irregular boundaries, without assuming spherical shape based clustering structure of data. The MSTs are used to produce the structure of point clusters in the Euclidean space.

[3]

- Image processing

Used for the fast and efficient segmentation of large image datasets.

[4]

[2]

Background and Purpose of Experiment

This experiment looks at the use of Minimum Spanning Trees in selecting the best route accross a computer network for packet delivery. The route with the smallest cost / delay needs to be determined when routing packets to reduce latency in internet connections. The cost/delay is represented by the weight. The weighted graph describing the relationships between the nodes/vertices of the network are described by an adjacency matrix.

This problem is an MST problem by nature, as graphs directly represent the data. The devices on the network being nodes, and the edges being links in between the devices.

The best route on the network is therefore the minimum spanning tree of the graph.

Methods

Sample Adjacency Matrix

The script provided was used to generate the adjacency matrix, which is passed in as input for the program.

First line is the number of vertices

```
5

0.0 6.4 8.2 6.7 3.5

6.4 0.0 6.1 5.9 8.5

8.2 6.1 0.0 7.0 6.8

6.7 5.9 7.0 0.0 2.9

3.5 8.5 6.8 2.9 0.0
```

Brute Force Algorithm

The brute force approach looks through all possible candidates for a spanning tree from the list of edges in the graph. It then checks each candidate/permutation if it contains all vertices, and is connected, and meets the requirements to be a spanning tree. When a spanning tree is found, the weight is calculated. The spanning tree with the minimum weight is then chosen as the minimum spanning tree.

Pseudocode

```
numEdges = size - 1

for i = 0 to 2^edges  //all possible permutations of edges

      IF numberOf1Bits(i) == numEdges  //permutations with required
num //of edges

            IF containsAllVertices(i)

                  IF connected(i)

                        listOfWeights[ ] = getWeight(i)

                  END IF

            END IF

      END IF

next i

Find min(listOfWeights);
```

Prim's Algorithm on Adjacency Matrix

Prim's algorithm works on the adjacency matrix by keeping track of rows of the vertices visited in the adjacency matrix, and then choosing the lowest weight edge from those vertices to visit next.

Pseudocode

```
mark first row as visited
while (not visited all rows)
      find lowest weighted edge from from visited vertices
      visit vertex with lowest weighted edge
      mark row of the vertex as visited
end while
```

Result Analysis

Time Complexity

The brute-force approach has an extremely inefficient time complexity, as it has multiple layers of n loops, coupled with the outer loop which iterates through all candidates with 4 bits from the 10 bit list of edges. (10 Choose 4).

Prim's algorithm running on the adjacency matrix has a time complexity of

N^2 (where n = size of matrix)

due to the outer loop iterating through all the elements in the array to find the minimum weight edge.

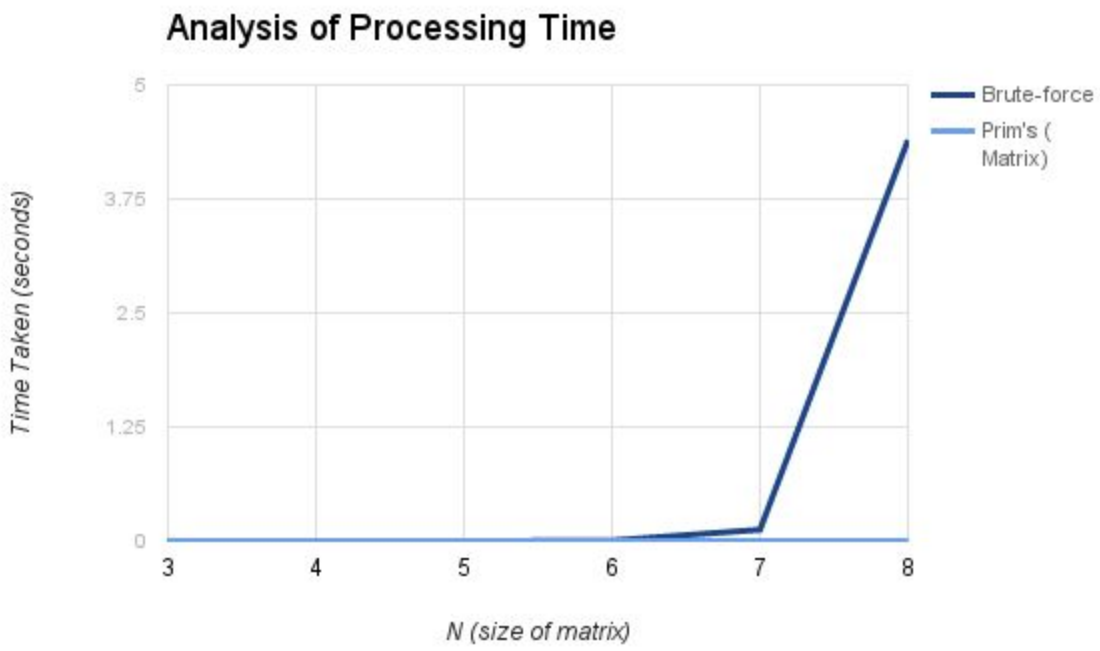
Space Complexity

Both algorithms have a space complexity of N^2, as it stores the adjacency matrix in memory, of size N*N (where N = size of matrix)

Both algorithms were operating on a NxN adjacency matrix, represented as a 2D array.

Time Taken

Prim's algorithm increases linearly with time, whereas brute-force increases almost with an increasing gradient (almost exponential)



The algorithms were tested for N >=3 and N<=8. For N < 3, the timing measured was too unreliable to use, and for N>8, the brute force algorithm did not return results in time.

Conclusion

Prim's algorithm is the acceptable method of finding minimum spanning trees, to be able to work for a large range of matrix sizes. It is not feasible or practical to use the brute-force algorithm because of it's complexity, and thereby the processing time required to complete the task.

The report concludes that brute-force methods are not a practical way of determining the minimum spanning tree of a graph for the aforementioned reasons.

References

[1]"Minimum Spanning Tree", 2016. [Online]. Available: http://algs4.cs.princeton.edu/lectures/43MinimumSpanningTrees.pdf. [Accessed: 30- Sep- 2016].

[2]"Geometry in Action: Minimum Spanning Trees", *Ics.uci.edu*, 2016. [Online]. Available: http://www.ics.uci.edu/~eppstein/gina/mst.html. [Accessed: 30- Sep- 2016].

[3]"Image Processing & Communications Challenges 6", *Google Books*, 2016. [Online]. Available: https://books.google.com.au/books?id=KVpbBAAAQBAJ&pg=PA47&lpg=PA47&dq=mst+in+image+processing&source=bl&ots=zv3RSso-_N&sig=zzpCFlC81eEmdipw14HfWPpvQXE&hl=en&sa=X&ved=0ahUKEwic4ZXbr7bPAhUE64MKHfPJCtkQ6AEIKTAC#v=onepage&q=mst%20in%20image%20processing&f=false. [Accessed: 30- Sep- 2016].

[4]"Bit Twiddling Hacks", *Graphics.stanford.edu*, 2016. [Online]. Available: https://graphics.stanford.edu/~seander/bithacks.html#CountBitsSetParallel. [Accessed: 30- Sep- 2016].

[5]M. Lecture 16: Greedy Algorithms, "Lecture 16: Greedy Algorithms, Minimum Spanning Trees | Video Lectures | Introduction to Algorithms (SMA 5503) | Electrical Engineering and Computer Science | MIT OpenCourseWare", *Ocw.mit.edu*, 2016. [Online]. Available: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-introduction-to-algorithms-sma-5503-fall-2005/video-lectures/lecture-16-greedy-algorithms-minimum-spanning-trees/. [Accessed: 30- Sep- 2016].

[6]"Spanning Trees", *University of Alabama*, 2016. [Online]. Available: http://www4.ncsu.edu/~zjorgen/ictai06.pdf. [Accessed: 30- Sep- 2016].