A STUDY ON SOME NAMED GRAPHS TO FIND THE MINIMUM SPANNING TREE (MST) USING GREEDY ALGORITHMS

A Arockiamary and B Preethi

PG and Research Department of Mathematics,
St. Joseph’s College of Arts and Science (Autonomous),
Cuddalore-607001, Tamilnadu, India.

E-mail: arockia68@gmail.com, preethibalaji1996@gmail.com

ABSTRACT: In this paper, Greedy Algorithms such as Kruskal’s, Prim’s, Boruvka’s, Reverse-delete Algorithm were applied on different types of graphs like Mobius-Kantor graph, Durer graph, Golomb graph to find the (MST).

Keywords: Kruskal’s Algorithm, Prim’s Algorithm, Boruvka’s Algorithm, Reverse-delete Algorithm, Mobius-Kantor graph, Durer graph, Golomb graph, Minimum Spanning Tree (MST).

1. Introduction:
A Graph consists of vertices and edges [5]. We use Greedy Algorithms to find the MST. First one is Boruvka’s Algorithm [2] developed by Otakar Boruvka in 1926. Second one is Prim’s Algorithm [2] invented by Vojtech Jarnik in 1930 and rediscovered by Prim in 1957. Kruskal’s Algorithm [3] is the third algorithm which we use commonly. And the Fourth one is Reverse-delete Algorithm. It is the reverse Process of Kruskal’s Algorithm, which is not commonly used.

In section 3, we use Mobius-Kantor graph, it is a graph with 16 vertices and 24 edges. In section 4, we introduce Durer graph, it is a graph with 12 vertices and 18 edges. And in section 5, we discuss about Golomb graph, it is a graph with 10 vertices and 18 edges. Thoughout the paper we use Greedy Algorithms to find the MST for these named graphs. And finally we conclude with the help of Greedy Algorithms, which is the best algorithm to find the MST.

2. Definitions:

2.1. Definition
A graph G is an ordered triple [1] < V, E, ψ > where V- set of vertices, E - set of edges and ψ - incidence function.

2.2. Definition
A closed trail [1] whose source and interior vertices are different is called a cycle.

2.3. Definition
A connected acyclic graph is called a Tree [4].

2.4. Definition
MST is a sub-component of the edges of a connected, undirected edge weighted [4] graph which connects the vertices side by side, with no cycle and with the minimum feasible total edge weight.
2.5. Definition
If there is a path [1] between each pair of vertices then the graph is said to be Connected.

3. Mobius-Kantor Graph:

![Mobius-Kantor Graph](image)

**Figure 3(a)**

3.1. Mobius-Kantor Graph using Kruskal’s Algorithm

Step 1: Arrange the edges in ascending sequence.

| EDGE | A | L | B | H | D | K | C | E | J | D | F | E | G | F | G | H | I | J | P | I | K | P | L |
|------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|      | B | C | C | I | M | B | D | N | A | E | O | F | P | G | H | A | N | O | M | M | L | N | K | O |
| WEIGHT | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 4 | 4 | 5 | 5 | 6 | 7 | 8 | 8 | 8 | 8 | 9 | 9 | 10 | 10 |

Step 2: Pick an edge with the smallest weight that does not form a cycle.

i.e.) connect an edge AB and LC having weight 1.
Step 3: Pick an edge with the next smallest weight that does not form a cycle. i.e.) connect an edge BC, BK, DM and HI having weight 2.

Step 4: Repeat step 2, until all the vertices are connected and a MST is obtained.
Its weight = 1+2+3+4+5+6+7+2+3+2+1+2+3+4+5 = 50

3.2. Mobius-Kantor Graph using Prim’s Algorithm

Step 1: Select any vertex and connect that vertex to the edge having smallest weight.
   i.e. connect an edge AB having weight 1.

![Figure 3.2(a)](image)

Step 2: Now treat the vertices A and B as the root vertex and choose the edge having smallest weight.
   i.e) connect an edge BK or BC having weight 2. Now we connect an edge BK.

![Figure 3.2(b)](image)

Step 3: Repeat step 2, as far as all the vertices are connected and a MST is obtained.

![Figure 3.2(c)](image)
Its weight = 1+2+1+9+2+2+9+3+4+2+8+5+6+4+8 = 66.

3.3. Mobius-Kantor graph using Boruvka’s Algorithm

Step 1: Select the minimum weighted edge in every vertex.

Step 2: Select the weighted edge that connects the graph and that does not form a cycle.

Step 3: Repeat step 2, until we obtain a MST.
It's weight = 1+3+3+2+1+2+3+4+5+2+8+9+8+9+8 = 68.

3.4. Mobius-Kantor graph using Reverse-Delete Algorithm

Step 1: Arrange the edges in descending sequence.

Table 3.4.

| EDGE | P | L | I | K | H | I | J | J | P | G | F | E | G | D | F | C | E | J | B | H | D | K | A | L |
|------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|      |   |   |   | K | O | N | N | A | N | O | M | M | H | G | F | P | E | O | D | N | A | C | I | M | B | B | C |
| WEIGHT | 10 | 10 | 9 | 9 | 8 | 8 | 8 | 8 | 8 | 7 | 6 | 5 | 5 | 4 | 4 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 1 | 1 |

Step 2: Delete the edges with the highest weight and check if deletion does not lead to disconnection. i.e.) delete an edge PK and LO having weight 10.
Step 3: Repeat step 2, until we obtain a MST.

\[
\text{Its weight} = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 2 + 3 + 2 + 1 + 2 + 3 + 4 + 5 = 50.
\]
4. Durer Graph:

![Durer Graph](image)

**Figure 4(a)**

4.1. **Durer Graph using Kruskal’s Algorithm**

Step 1: Arrange the edges in ascending sequence.

| EDGE | A | H | B | I | C | J | F | E | E | E | F | A | G | B | L | I | L | G |
|------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|      | B | J | C | K | D | D | L | D | F | K | A | G | I | H | H | C | J | K |
| WEIGHT | 1 | 1 | 2 | 2 | 3 | 3 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | 9 | 10 | 10 | 10 | 11 |

Step 2: Pick an edge with the smallest weight that does not form a cycle.

i.e.) connect an edge AB and HJ having weight 1.

![Edge AB and HJ](image)

**Figure 4.1(a)**
Step 3: Pick an edge with the next smallest weight that does not form a cycle. i.e.) connect an edge BC and IK having weight 2.

![Figure 4.1(b)](image)

Step 4: Repeat step 2, until all the vertices are connected and a MST is obtained.

![Figure 4.1(c)](image)

\[ \therefore \text{Its weight} = 1+2+3+4+5+3+7+1+3+5+2 = 36. \]

4.2. *Durer Graph using Prim’s Algorithm*

Step 1: Select any vertex and connect that vertex to the edge having smallest weight. i.e.) connect an edge AB having weight 1.
Step 2: Now treat the vertices A and B as the root vertex and choose the edge having smallest weight. i.e. connect an edge BC having weight 2.

Step 3: Repeat step 2, until all the vertices are connected and a MST is obtained.

\[ \therefore \text{Its weight} = 1 + 2 + 3 + 4 + 5 + 2 + 8 + 3 + 1 + 3 + 6 = 38. \]

4.3. Durer graph using Boruvka's Algorithm

Step 1: Select the minimum weighted edge in every vertex.
Step 2: Select the weighted edge that connects the graph and make sure a cycle is not formed.

Step 3: Repeat step 3, until we obtain a MST.
It’s weight = 1+2+3+4+3+2+8+7+1+10+3 = 44.

4.4. Durer graph using Reverse-Delete Algorithm

Step 1: Arrange the edges in descending sequence.

Table 4.4.

| EDGE | G | L | I | L | B | G | A | F | E | E | E | F | J | C | I | B | H | A |
|------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|      | K | J | C | H | H | I | G | A | K | F | D | L | D | D | K | C | J | B |
| WEIGHT | 11 | 10 | 10 | 10 | 9 | 8 | 7 | 6 | 5 | 5 | 4 | 3 | 3 | 3 | 2 | 2 | 1 | 1 |

Step 2: Delete the edges with the highest weight and check if deletion does not lead to disconnection. i.e.) delete an edge GK having weight 11.
Step 3: Repeat step 2, until we obtain a MST.

![Figure 4.4(b)](image)

\[
\therefore \text{Its weight } = 1+2+3+4+5+3+5+2+3+1+7 = 36.
\]

5. Golomb Graph:

![Figure 5(a)](image)

5.1. Golomb Graph using Kruskal’s Algorithm

Step 1: Arrange the edges in ascending sequence.

Table 5.1.

| EDGE | A | C | F | B | C | H | A | I | B | D | H | J | I | J | J | J | E |
|------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|      | B | H | G | C | A | J | D | H | F | E | G | E | J | D | G | D | F | F |
| WEIGHT | 1 | 1 | 1 | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 6 | 6 | 7 | 8 | 8 | 9 | 9 | 10 |
Step 2: Pick an edge with the smallest weight that does not form a cycle.
i.e.) connect an edge AB, CH and FG having weight 1.

Figure 5.1(a)

Step 3: Pick an edge with the next smallest weight and make sure a cycle is not formed.
i.e.) connect an edge BC having weight 2.

Figure 5.1(b)

Step 4: Repeat step 2, until all the vertices are connected and a MST is obtained.
\[
\text{Its weight } = 1 + 2 + 1 + 4 + 3 + 1 + 5 + 5 + 4 = 26.
\]

5.2. Golomb Graph using Prim’s Algorithm

Step 1: Select any vertex and connect that vertex to the edge having smallest weight. i.e.) connect an edge AB having weight 1.

Step 2: Now treat the vertices A and B as the root vertex and choose the edge having smallest weight. i.e.) connect an edge BC having weight 2.
Step 3: Repeat step 2, as far as all the vertices are connected and a MST is obtained.

\[
\text{Its weight} = 1 + 2 + 1 + 3 + 6 + 5 + 8 + 5 + 1 = 32.
\]

5.3. Golomb graph using Boruvka's Algorithm

Step 1: Select the minimum weighted edge in every vertex.
Step 2: Select the weighted edge that connects the graph and make sure a cycle is not formed.

Step 3: Repeat step 2, as far as we obtain a MST.
It's weight = 1+2+1+4+3+6+1+5+4 = 27.

5.4. Golomb graph using Reverse-Delete Algorithm

Step 1: Arrange the edges in descending sequence.

Table 5.4.

| EDGE | E | J | D | I | J | I | J | H | F | D | I | A | J | C | C | C | G | A |
|------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|      | H | F | J | D | G | J | E | G | B | E | H | D | H | A | B | H | F | B |
| WEIGHT | 10 | 9 | 9 | 8 | 8 | 7 | 6 | 6 | 5 | 5 | 4 | 4 | 3 | 3 | 2 | 1 | 1 | 1 |

Step 2: Delete the edges with the highest weight and check if deletion does not lead to disconnection. i.e.) delete an edge EF having weight 10.

Step 3: Repeat step 2, until we obtain a MST.
Its weight = 1+2+1+4+3+1+5+5+4 = 26.

Remark:
Till now there is no MST for Mobius-Kantor graph, Durer graph and Golomb graph using Greedy Algorithms. In this paper, we explain about the Greedy Algorithms to find the MST for these named graphs.

6. Conclusion:
In this paper, we discussed about the Greedy Algorithms such as Kruskal’s, Prim’s, Boruvka’s and Reverse-Delete Algorithm to find the MST for some named graphs. Among these Kruskal’s Algorithm is the best to find the MST.

7. Reference:
[1] Bondy J A and Murty U S R, *Graph theory with Applications*, NORTH-HOLLAND, New York · Amsterdam · Oxford. ISBN 0-444-19451-7, pp(1-25)
[2] Cuneyt F. Bazlamacc, Khalil S. Hindi, *Minimum weight spanning tree algorithms A survey and empirical study*, Computer & Operations Research 28 (2001) 767 – 785, PII: S0305-0548(00)00007-1.
[3] Keijo Ruohonen , *Graph Theory*, Tampere University of Technology 2008, pp (67-70)
[4] Krishnaiyan “KT” Thulasiraman, Subramanian Arumugam, Andreas Brandstadt, Takao Nishizeki,*Handbook of Graph Theory, Combinatorial Optimization, and Algorithms*, CRC Press Taylor & Francis Group, Boca Raton London New York, A CHAPMAN & HALL BOOK, ISBN-13: 978-1-4200-1107-4 (eBook-PDF), pp(21-22)
[5] Rame LIKAJ and Ahmet SHALA, *Application of Graph Search Algorithm Dijkstra to find Optimal Solution for the Problem of Transport*, ANNALS of Faculty Engineering Hunedoara-International Journal of Engineering Tome XV[2017] – Fascicule 4[November], ISSN: 1584-2665 [print; online], ISSN:1584-2673[CD-Rom; online].