TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN MÔN TOÁN TỔ HỢP VÀ ĐỒ THỊ**

**Combinatorial Optimization**

*Người hướng dẫn*: **THẦY LÊ ĐÌNH THẬN**

*Người thực hiện*: **TRỊNH THẾ DUY– 51603083**

**LÊ TRÍ DŨNG – 51603065**

**PHƯƠNG LẬP PHÁT - 51403176**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN MÔN TOÁN TỔ HỢP VÀ ĐỒ THỊ**

**Combinatorial Optimization**

*Người hướng dẫn*: **THẦY LÊ ĐÌNH THẬN**

*Người thực hiện*: **TRỊNH THẾ DUY– 51603083**

**LÊ TRÍ DŨNG – 51603065**

**PHƯƠNG LẬP PHÁT - 51403176**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của Thầy Lê Đình Thận;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Trịnh Thế Duy*

*Lê Trí Dũng*

*Phương Lập Phát*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Ba thuật toán: Maximum Network Flow, Shortest Path và Minimum Spanning Tree .

CATEGORY

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN 4](#_Toc7949007)

[TÓM TẮT 5](#_Toc7949008)

[CATEGORY 1](#_Toc7949009)

[I.INTRODUCTION 2](#_Toc7949010)

[1.1 The 3 Algorithims 2](#_Toc7949011)

[1.1.1 Maximum Network Flow 2](#_Toc7949012)

[1.1.2 Shortest Path 3](#_Toc7949013)

[1.1.3 Minimum Spanning Tree 3](#_Toc7949014)

[II.STATE OF THE ART 3](#_Toc7949015)

[2.1 Maximum Network Flow 3](#_Toc7949016)

[2.2 Shortest Path 3](#_Toc7949017)

[2.3 Minimum Spanning Tree 4](#_Toc7949018)

[III.APPROACH 4](#_Toc7949019)

[3.1 Maximum NetWork Flow 4](#_Toc7949020)

[3.2 Shotest Path 4](#_Toc7949021)

[3.3 Minimum Spanning Tree 4](#_Toc7949022)

[3.4 Input 4](#_Toc7949023)

[IV.EXPEREMENTS AND RESULTS 5](#_Toc7949024)

[4.1 Maximum Network Flow 5](#_Toc7949025)

[4.2 Shortest Path 5](#_Toc7949026)

[4.3 Minimum Spanning Tree 5](#_Toc7949027)

[V.CONCLUSION 5](#_Toc7949028)

[REFERENCES OF DOCUMENT 6](#_Toc7949029)

I.INTRODUCTION

Combinatorial Optimaztion is about finding the optimal path/object from a finite set of paths/objects.

We will be using 3 different algorithms to solve problems about optimal pathing of a given graph.

1.1 The 3 Algorithims

1.1.1 Maximum Network Flow

1.1.2 Shortest Path

In graph theory, the problem of finding the shortest path is to solve the problem of finding the shortest path between two vertices in a graph so that the path weight is the shortest.

The problem of finding the shortest path between crossroads can be considered a special case of finding the shortest path on a graph, where we can see crossroads as vertices and edges as path from this crossroad to another, weight of edges is the distance from one crossroad to another.

The shortest path problem can apply on graphs whether undirected, directed or mixed.

1.1.3 Minimum Spanning Tree

A minimum spanning tree or minimum weight spanning tree is a subset of edges in a connected, edge weighted undirected graph in which all the verticles connects together containng no cycles and with the smallest number of total edge weight.

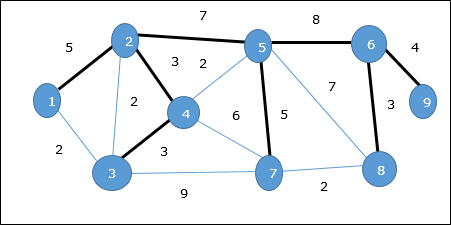


Image 1.3: Minimum Spanning Tree

Source:” <https://www.tutorialspoint.com/design_and_analysis_of_algorithms/images/minimum_spinning_tree.jpg>”

II.STATE OF THE ART

2.1 Maximum Network Flow

2.2 Shortest Path

There are many algorithm to solve the problem but the most important are:

* Dijkstra’s algorithm
* Bellman-Ford algorithm
* A\* search algorithm
* Floyd-Warshall algorithm
* Johnson’s algorithm
* Viterbi algorithm

The most common algorithm to use for solving the shortest path problem is Dijkstra’s algorithm.

2.3 Minimum Spanning Tree

As of right now there are many ways proposed by different people, some state-of-the-art algorithm are:

* Kruksal’s (Serial distributed)
* Prim’s (Serial centralized)
* Boruvska’s (Parallel distributed)
* Fredman & Tarjan’s
* Gabow, Galil & Spencer’s
* Karger, Klein & Tarjan’s
* Frederickson’s

I will be using the Prim’s algorithm.

III.APPROACH

3.1 Maximum NetWork Flow

3.2 Shotest Path

Dijkstra’s algorithm given source as initial node. We maintain two sets, one set contains vertices included in shortest path tree, other set includes vertices not yet included in shortest path tree. At every step of the algorithm, we find a vertex which is in the other set (set of not yet included) and has a minimum distance from the source.

3.3 Minimum Spanning Tree

First we will start with an empty spanning tree while maintaining two sets of vertices, vertices that are already included in the MST and vertices that are not yet inclued, at every step we will find a cut between the two sets and pick the minimum weight edge then include that vertex to the MST set.

3.4 Input

A graph will be inputted from a text file named graph.txt, then the required input for each algorithm ( number of vertices and source point ) input will differ between 3 algorithms. The Main.java file will be our main file to execute the algorithms.

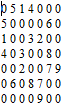
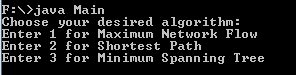


Image 3.1: Format for the graph.txt file



IV.EXPEREMENTS AND RESULTS

4.1 Maximum Network Flow

4.2 Shortest Path

**4.3 Minimum Spanning Tree**

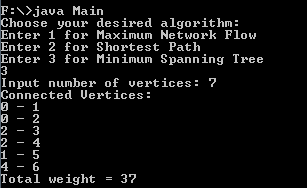
A graph and number of vertices of that graph is the input of my program.

Image 4.3: Result of MST.

V.CONCLUSION

Through our group research for the assignment we have also learned about new algorithm as well as useful information about the subject of Combinatorial Optimization.

REFERENCES OF DOCUMENT

1. <https://en.wikipedia.org/wiki/Combinatorial_optimization>
2. <https://en.wikipedia.org/wiki/Minimum_spanning_tree>
3. <https://en.wikipedia.org/wiki/Prim%27s_algorithm>