TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**JAVA**

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN/ĐỒ ÁN CUỐI KÌ MÔN TOÁN TỔ HỢP VÀ ĐỒ THỊ**

**Combinatorial Optimization**

****

*Người thực hiện*: **HUỲNH GIA THIỆN – 51702186**

**LÊ HOÀNG PHÚC - 51702161**

Lớp **: 17050201**

Khoá  **: 21**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**JAVA**

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN/ĐỒ ÁN CUỐI KÌ MÔN TOÁN TỔ HỢP VÀ ĐỒ THỊ**

**Combinatorial Optimization**

Người thực hiện: **HUỲNH GIA THIỆN - 51702186**

**LÊ HOÀNG PHÚC 51702161**

Lớp **: 17050201**

Khoá  **: 21**



**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

TÓM TẮT

Trình bày tóm tắt vấn đề nghiên cứu, các hướng tiếp cận, cách giải quyết vấn đề và một số kết quả đạt được, những phát hiện cơ bản trong vòng 1 -2 trang.

MỤC LỤC

[TÓM TẮT i](#_Toc4353605)

[MỤC LỤC 1](#_Toc4353606)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 3](#_Toc4353607)

[I.INTRODUCTION 4](#_Toc4353608)

[1.1 Khái niệm 4](#_Toc4353609)

[1.1.1 Maximum Network Flow 4](#_Toc4353610)

[1.1.2 Shortest Path 4](#_Toc4353611)

[1.1.3 Minimum Spanning Tree 5](#_Toc4353613)

[1.2 Lịch sử 5](#_Toc4353614)

[1.2.1 Maximum Network Flow 5](#_Toc4353615)

[1.2.2 Shortest Path 5](#_Toc4353616)

[1.2.3 Minimum Spanning Tree 5](#_Toc4353617)

[II.STATE OF THE ART 6](#_Toc4353618)

[III.APPROACH 6](#_Toc4353619)

[IV.EXPEREMENTS AND RESULTS 6](#_Toc4353620)

[V.CONCLUSION 6](#_Toc4353621)

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

**CÁC KÝ HIỆU**

*f Tần số của dòng điện và điện áp (Hz)*

*p Mật độ điện tích khối (C/m3)*

**CÁC CHỮ VIẾT TẮT**

CSTD Công suất tác dụng

MF Máy phát điện

BER Tỷ lệ bít lỗi

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 2.1: Kiến trúc FTP 2](#_Toc387689394)

**DANH MỤC BẢNG**

[Bảng 3.1 Ví dụ cho chèn bảng 2](#_Toc387689363)

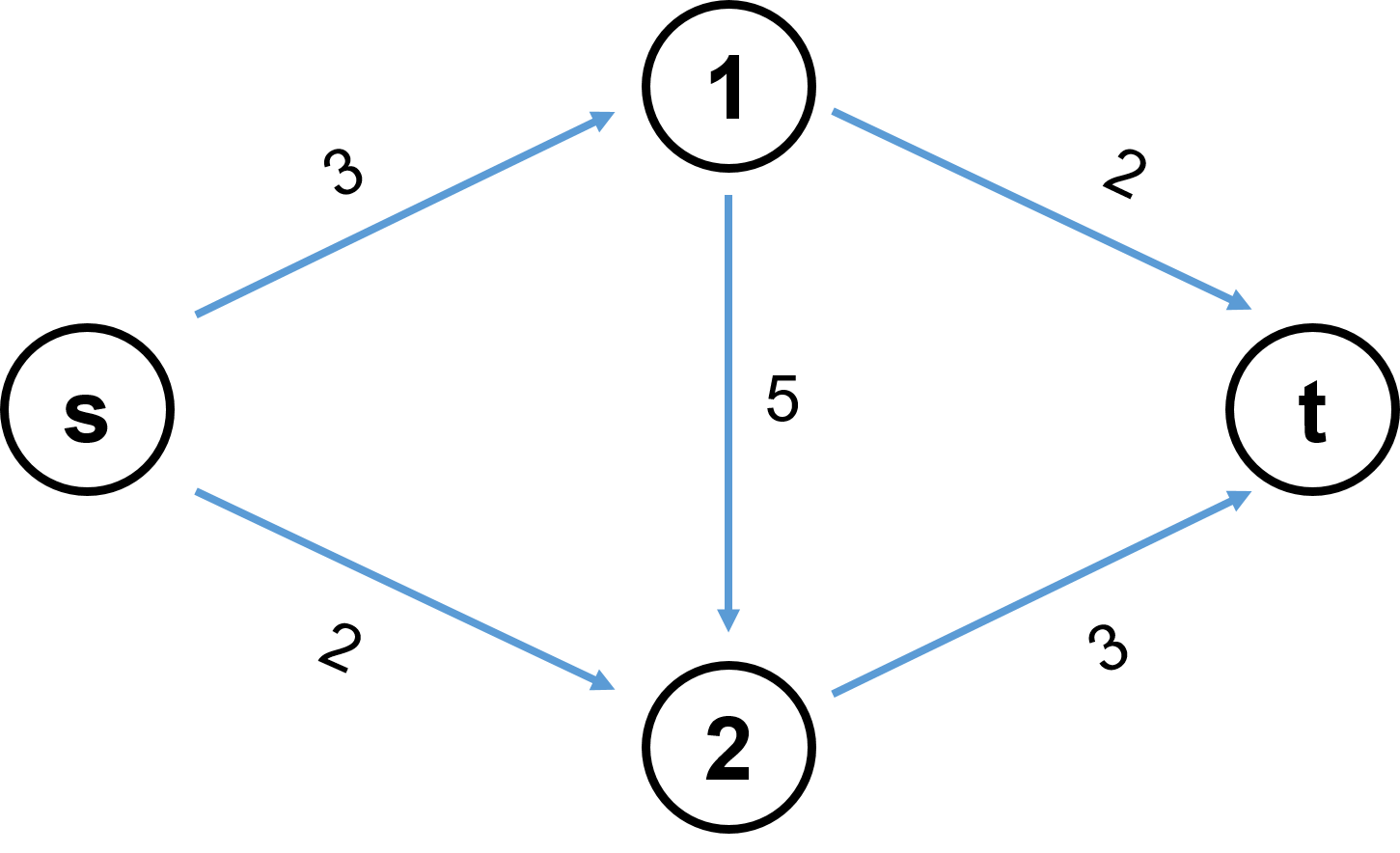
I.INTRODUCTION

Trong hoạt động nghiên cứu, Combinatorial Optimization là một chủ đề bao gồm tìm kiếm một phương pháp tối ưu từ một tập hợp nhiều cách thực thi hữu hạn. Trong nhiều vấn đề như vậy, tìm kiếm toàn diện không thể thực hiện được. Nó hoạt động trên miền của các vấn đề tối ưu hóa đó, trong đó tập hợp các giải pháp khả thi là rời rạc hoặc biến hóa của rời rạc, với mục tiêu là tìm ra giải pháp tốt nhất.

Combinatorial Optimization là một tập hợp con của mathematical optimization có liên quan đến nghiên cứu hoạt động, lý thuyết thuật toán và lý thuyết phức tạp tính toán. Nó có các ứng dụng quan trọng trong một số lĩnh vực, bao gồm trí tuệ nhân tạo, học máy, lý thuyết đấu giá và công nghệ phần mềm

1.1 Khái niệm

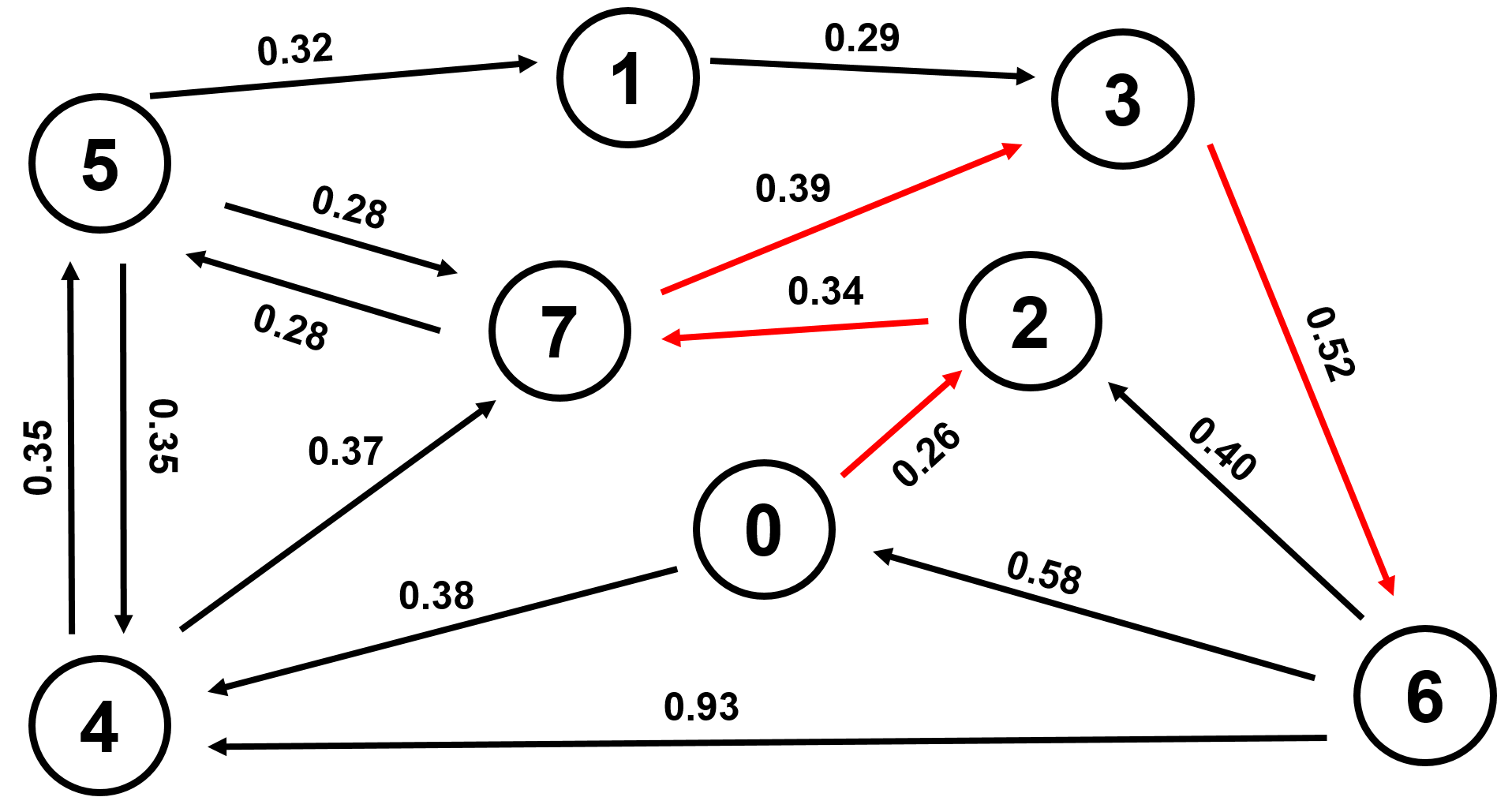
1.1.1 Maximum Network Flow

Maximum flow problems involve liên quan đến việc tìm kiếm một luồng khả thi thông qua một mạng lưu lượng đơn, đó là một luồng có lưu lượng tối đa.

1.1.2 Shortest Path

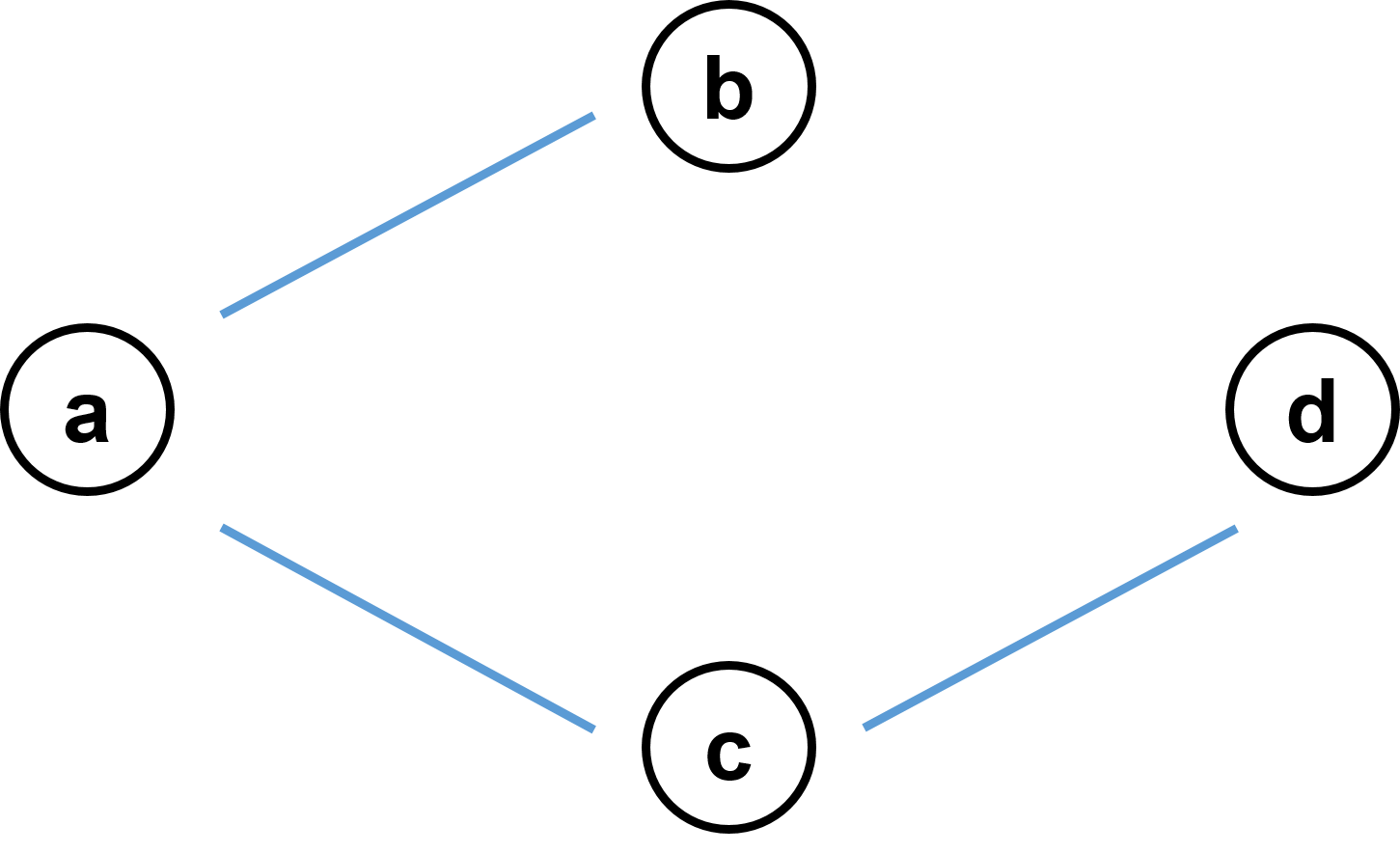
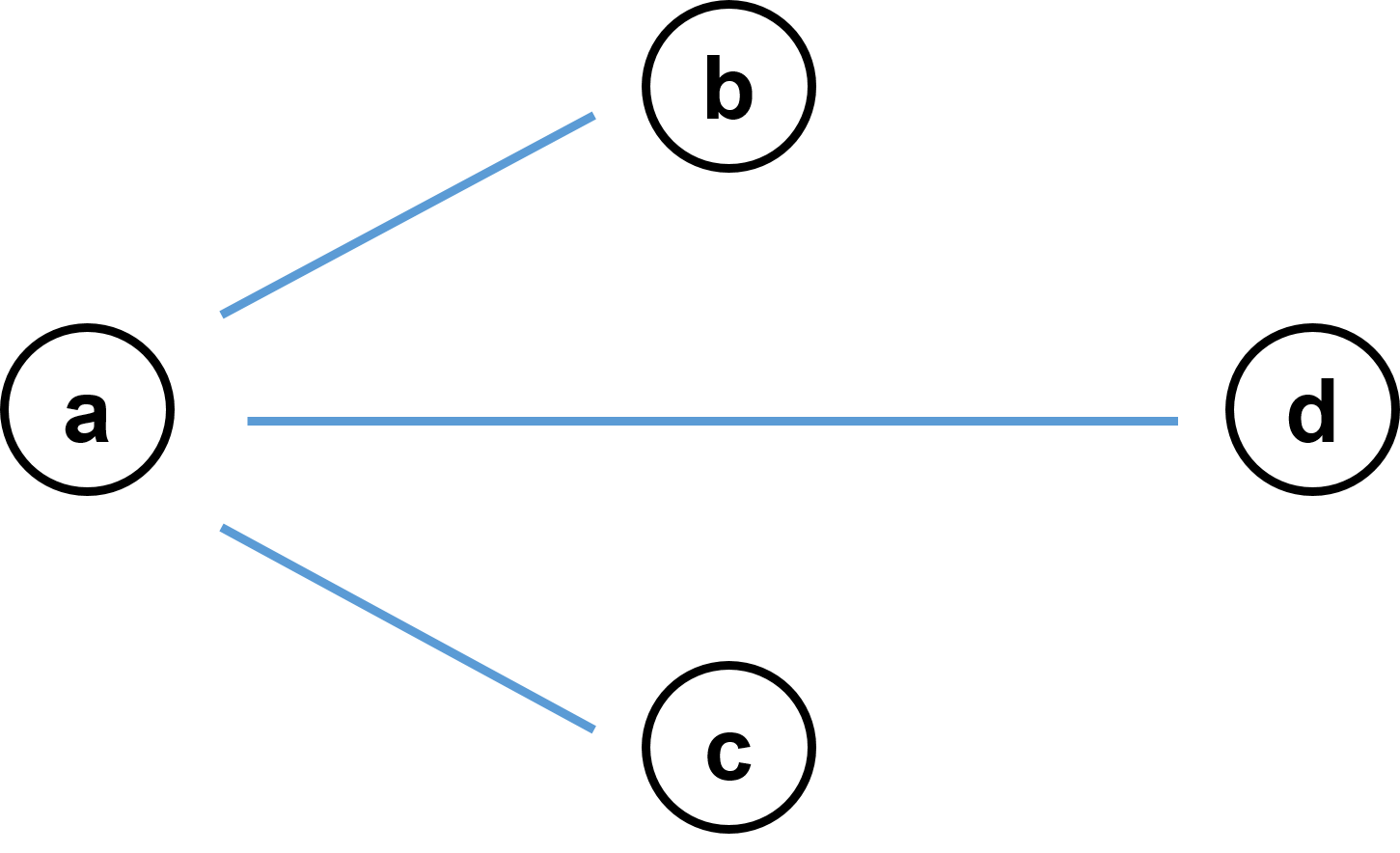
The shortest path là vấn đề tìm đường đi giữa hai đỉnh (hoặc nút) trong biểu đồ sao cho tổng trọng số của các cạnh cấu thành của nó là thấp nhất.

Shorter Path có thể được mô hình hóa như một trường hợp đặc biệt của vấn đề đường đi ngắn nhất trong biểu đồ, trong đó các đỉnh tương ứng với giao lộ và các cạnh tương ứng với các đoạn đường, mỗi trọng số theo chiều dài của phân khúc.



1.1.3 Minimum Spanning Tree

Minimum Spanning Tree là tập hợp con của các cạnh của đồ thị vô hướng có trọng số kết nối tất cả các đỉnh với nhau, không có chu kỳ và với tổng trọng lượng tối thiểu có thể có.



1.2 Lịch sử

1.2.1 Maximum Network Flow

Maximum flow problem được T. E. Harris and F. S. Ross lần đầu tiên báo cáo vào năm 1945 nhằm mô hình hóa tình trạng giao thông đường sắt ở Liên Xô.Nó là một phần của Maximum NetWork Flow.Và sau đó 10 năm, Lester R. Ford, Jr. và Delbert R. Fulkerson đã dựa vào mô hình đó tạo ra thuật toán đầu tiên, thuật toán mang tên Ford–Fulkerson.

1.2.2 Shortest Path

Shortest Path là một mô hình giải quyết vấn đề hữu ích rộng rãi trong việc điều khiển robot, lập bản đồ kết cấu, sắp chữ trong TEX,…Thuật toán Shortest Path đầu tiên được báo cáo bởi Shimbel vào năm 1954.

1.2.3 Minimum Spanning Tree

Thuật toán Minimum Spanning Tree được phát hiện đầu tiên bởi nhà khoa học Séc Otakar Borůvka vào năm 1926,nhằm kiếm vùng phủ sóng hiệu quả của Moravia bằng điện.

Thuật toán Minimum Spanning Tree nhanh nhất hiện nay được phát triển bởi Bernard Chazelle. Thuật toán dựa trên heap mềm, hàng đợi ưu tiên gần đúng

II.STATE OF THE ART

III.APPROACH

IV.EXPEREMENTS AND RESULTS

4.1 Maximum Network Flow

4.1.1 Code

// Java program for implementation of Ford Fulkerson algorithm

import java.util.\*;

import java.lang.\*;

import java.io.\*;

import java.util.LinkedList;

class MaximumNetworkFlow

{

static final int V = 6;

private int maxFlowvalue;

private int[][] graph;

/\*\*

\* @return the maxFlowvalue

\*/

public int getMaxFlow\_value() {

return maxFlowvalue;

}

private boolean bfs(int rGraph[][], int s, int t, int parent[])

{

boolean visited[] = new boolean[V];

for(int i=0; i<V; i++)

visited[i]=false;

LinkedList<Integer> queue = new LinkedList<Integer>();

queue.add(s);

visited[s] = true;

parent[s]=-1;

while (queue.size()!=0)

{

int u = queue.poll();

for (int v=0; v<V; v++)

{

if (visited[v]==false && rGraph[u][v] > 0)

{

queue.add(v);

parent[v] = u;

visited[v] = true;

}

}

}

return (visited[t] == true);

}

public void fordFulkerson(int graph[][], int V)

{

this.graph = graph;

int s = 0, t = V - 1;

int u, v;

int rGraph[][] = new int[V][V];

for (u = 0; u < V; u++)

for (v = 0; v < V; v++)

rGraph[u][v] = graph[u][v];

int parent[] = new int[V];

int max\_flow = 0;

while (bfs(rGraph, s, t, parent))

{

int path\_flow = Integer.MAX\_VALUE;

for (v=t; v!=s; v=parent[v])

{

u = parent[v];

path\_flow = Math.min(path\_flow, rGraph[u][v]);

}

for (v=t; v != s; v=parent[v])

{

u = parent[v];

rGraph[u][v] -= path\_flow;

rGraph[v][u] += path\_flow;

}

max\_flow += path\_flow;

}

maxFlowvalue = max\_flow;

}

public String flowofMax()

{

String flow = "0";

boolean visited[] = new boolean[V];

for(int i=0; i<V; i++)

visited[i]=false;

int current = 0;

visited[current] = true;

while(current < V - 1)

{

int max = graph[current][0];

int v = 0;

for(int i = 0; i < V;i++){

if(graph[current][i] > max && visited[i] == false){

max = graph[current][i];

v = i;

}

}

current = v;

visited[current] = true;

flow += " -> " + current;

}

return flow;

}

public static void main (String[] args) throws java.lang.Exception

{

//call run() before call another method

int graph[][] =new int[][] { {0, 16, 18, 0, 0, 0},

{0, 0, 10, 12, 0, 0},

{0, 4, 0, 0, 14, 0},

{0, 0, 9, 8, 0, 20},

{0, 2, 0, 7, 0, 4},

{0, 0, 0, 0, 0, 0}

};

MaximumNetworkFlow m = new MaximumNetworkFlow();

m.fordFulkerson(graph, V);

System.out.println("The maximum possible flow is "

+m.getMaxFlow\_value());

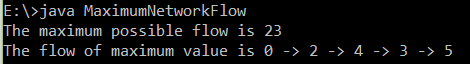
System.out.println("The flow of maximum value is "

+m.flowofMax());

}

}

4.1.2 Result



4.2 Shortest Path

4.2.1 Code

4.2.2 Result

4.3 Minimum Spanning Tree

4.3.1 Code

4.3.2 Result

V.CONCLUSION

**TÀI LIỆU THAM KHẢO**

1. <https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/>

2.