TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO MÔN TOÁN TỔ HỢP VÀ ĐỒ THỊ**

**BÀI TẬP LỚN**

*Người thực hiện*: **HUỲNH NGỌC TIẾN – 51702194**

**HUỲNH VĂN HOÀI – 51702105**

**ĐOÀN TUẤN KIỆT - 51702125**

Lớp **: 17050201**

Khoá  **: 21**

**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**



**BÁO CÁO MÔN TOÁN TỔ HỢP VÀ ĐỒ THỊ**

**BÀI TẬP LỚN**

*Người thực hiện*: **HUỲNH NGỌC TIẾN – 51702194**

**HUỲNH VĂN HOÀI – 51702105**

**ĐOÀN TUẤN KIỆT - 51702125**

Lớp **: 17050201**

Khoá  **: 21**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

LỜI CẢM ƠN

Xin cảm ơn giảng viên đã cung cấp kiến thức và tạo điều kiện để em có thể hoàn thành bài báo cáo cũng như củng cố kiến thức cho bản thân để có thể áp dụng

# 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; 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.

**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.

*TP. Hồ Chí Minh, ngày 22 tháng 02 năm 2019*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Huỳnh Ngọc Tiến*

*Huỳnh Văn Hoài*

*Đoàn Tuấn Kiệt*

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)

**MỤC LỤC**

[**ĐỒ ÁN ĐƯỢC HOÀN THÀNH ii**](#_Toc4359366)

[**PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii**](#_Toc4359367)

[**MỤC LỤC 1**](#_Toc4359368)

[**CHƯƠNG I: INTRODUCTION 2**](#_Toc4359369)

[**CHƯƠNG II: STATE OF THE ART 3**](#_Toc4359370)

[**CHƯƠNG III: APPROACH 4**](#_Toc4359371)

[**CHƯƠNG IV: EXPERIMENT AND RESULT 5**](#_Toc4359372)

[**CÂU 1. MAXIMUM NETWORK FLOW 5**](#_Toc4359373)

[**4.1 Code 5**](#_Toc4359374)

[**4.2 Kết Quả 11**](#_Toc4359375)

[**CHƯƠNG V: CONCLUSION 12**](#_Toc4359376)

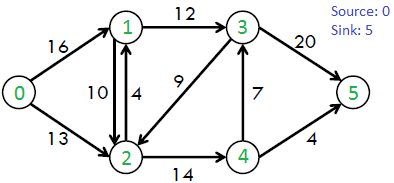
# CHƯƠNG I: INTRODUCTION

# CHƯƠNG II: STATE OF THE ART

# CHƯƠNG III: APPROACH

# CHƯƠNG IV: EXPERIMENT AND RESULT

## CÂU 1. MAXIMUM NETWORK FLOW



* 1. Code
     1. **Maximum Network Flow**

import java.util.\*;

import java.lang.\*;

import java.io.\*;

public class MaximumFlow

{

public static final int S = 6; //S la so luong dinh

//tra ve gia tri la true neu di tu s den t co trong do thi

boolean BFS(int Graph[][],int s,int t,int parent[])

{

boolean visited[] = new boolean[S];

for(int i=0; i<S; ++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<S; v++)

{

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

{

queue.add(v);

parent[v] = u;

visited[v] = true;

}

}

}

return (visited[t] == true);

}

public int maxFlow(int graph[][],int s,int t)

{

int u, v;

int rGraph[][] = new int[S][S];

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

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

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

int parent[] = new int[S];

int maxFlow = 0;

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

{

int pathFlow = Integer.MAX\_VALUE;

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

{

u = parent[v];

pathFlow = Math.min(pathFlow, rGraph[u][v]);

}

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

{

u = parent[v];

rGraph[u][v] -= pathFlow;

rGraph[v][u] += pathFlow;

}

maxFlow += pathFlow;

}

return maxFlow;

}

private static void DFS(int[][]rGraph,int s,boolean[] visited) {

visited[s] = true;

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

if (rGraph[s][i] > 0 && !visited[i]) {

DFS(rGraph, i, visited);

}

}

}

public void minCut(int[][] graph, int s, int t) {

int u,v;

int[][] rGraph = new int[S][S];

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

for (int j = 0; j < S; j++) {

rGraph[i][j] = graph[i][j];

}

}

int[] parent = new int[S];

while (BFS(rGraph, s, t, parent)) {

int pathFlow = Integer.MAX\_VALUE;

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

u = parent[v];

pathFlow = Math.min(pathFlow, rGraph[u][v]);

}

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

u = parent[v];

rGraph[u][v] = rGraph[u][v] - pathFlow;

rGraph[v][u] = rGraph[v][u] + pathFlow;

}

}

boolean[] isVisited = new boolean[S];

DFS(rGraph, s, isVisited);

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

for (int j = 0; j < S; j++) {

if (graph[i][j] > 0 && isVisited[i] && !isVisited[j]) {

System.out.println(i + " - " + j);

}

}

}

}

public static void main (String[] args)

{

// Tạo biểu đồ

int graph[][] = { {0, 16, 13, 0, 0, 0},

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

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

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

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

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

MaximumFlow n = new MaximumFlow();

System.out.println("Maximum Flow: " + n.maxFlow(graph, 0, 5)); //giả sử đi từ đỉnh 0 đến đỉnh 5

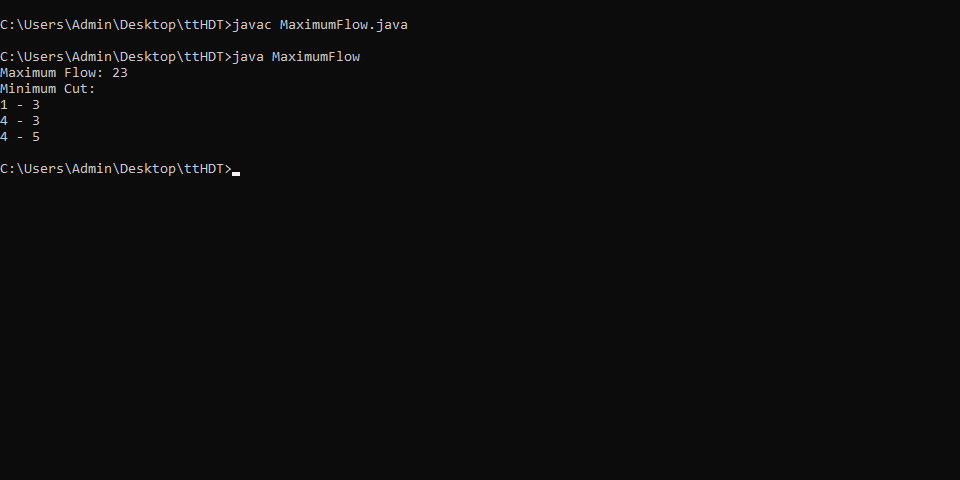
System.out.println("Minimum Cut:");

n.minCut(graph,0,5);

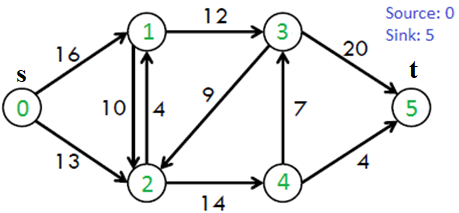
}

}

**Kết quả** Maximum Network Flow



* + 1. ShortestPath



**Source Code:**

import java.util.\*;

class ShortestPath {

static final int V = 6;

int minDistance(int dist[], Boolean set[])

{

//Tao gia tri nho nhat

int min = Integer.MAX\_VALUE, minIndex = -1;

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

if (set[v] == false && dist[v] <= min) {

min = dist[v];

minIndex = v;

}

return minIndex;

}

void printSolution(int dist[])

{

System.out.println("Khoang cach tu s toi t la: "+dist[V-1] );

}

void dijkstra(int graph[][], int src)

{

int dist[] = new int[V];

Boolean set[] = new Boolean[V];

//Khoi tao khoang cach la INFI va set[i] la false

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

dist[i] = Integer.MAX\_VALUE;

set[i] = false;

}

//Khoang cach tu source toi source = 0

dist[src] = 0;

//Tim duong di ngan nhat cho cac dinh

for (int count = 0; count < V - 1; count++) {

int u = minDistance(dist, set);

set[u] = true;

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

if (!set[v] && graph[u][v] != 0 &&

dist[u] != Integer.MAX\_VALUE && dist[u] + graph[u][v] < dist[v])

dist[v] = dist[u] + graph[u][v];

}

printSolution(dist);

}

public static void main(String[] args)

{

int graph[][] = { {0, 16, 13, 0, 0, 0},

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

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

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

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

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

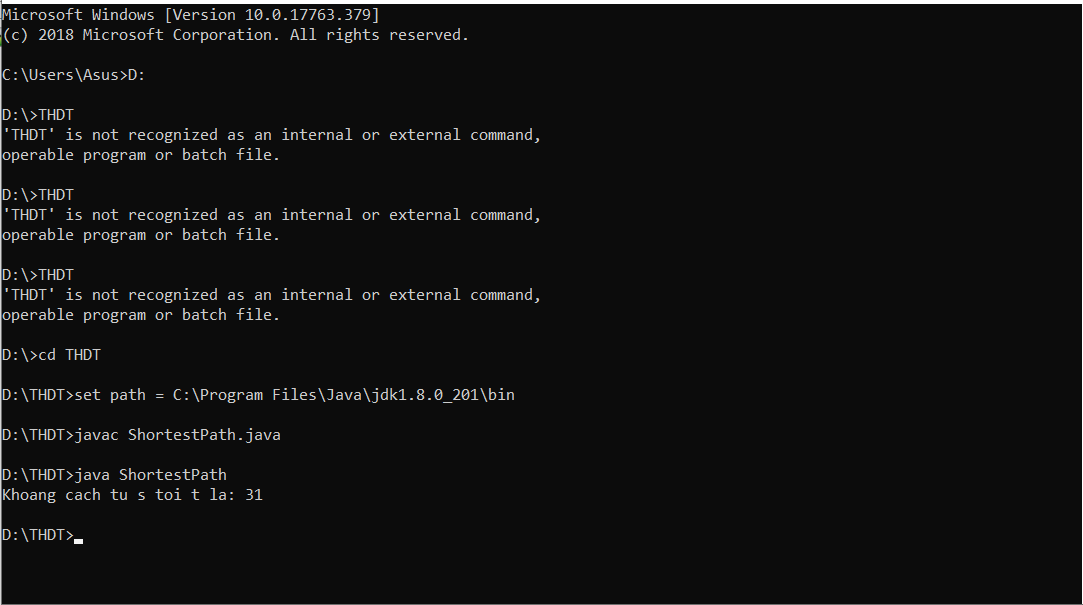
ShortestPath t = new ShortestPath();

t.dijkstra(graph, 0);

}

}

**Kết quả ShortestPath**



# CHƯƠNG V: CONCLUSION