**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**ĐẠI HỌC UEH**

**KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH**

🖎🕮✍



**ĐỒ ÁN MÔN HỌC**

**ĐỀ TÀI**

**ĐỒ THỊ VÀ ỨNG DỤNG TRONG QUẢN LÝ LỘ TRÌNH ĐƯỜNG ĐI**

**Học Phần: Cấu Trúc Dữ Liệu & Giải Thuật**

**Danh Sách Nhóm**:

1. NGUYỄN NGỌC BẢO
2. LỒ SÌN DẬU
3. ĐOÀN TRẦN BÁ ĐẠT
4. NGUYỄN HOÀN THIỆN

**Chuyên Ngành**: CÔNG NGHỆ PHẦN MỀM

**Khóa**: K46

**Giảng Viên**: TS. Đặng Ngọc Hoàng Thành

**Tp. Hồ Chí Minh, Ngày 19 tháng 12 năm 2021**

# MỤC LỤC

[MỤC LỤC 2](#_Toc90583825)

[CHƯƠNG 1. ĐỒ THỊ 3](#_Toc90583826)

[1.1. Các Khái Niệm Liên Quan 3](#_Toc90583827)

[1.2. Cấu Trúc và Cài Đặt Đồ Thị 3](#_Toc90583828)

[1.3. Các Thuật Toán Trên Đồ Thị 4](#_Toc90583829)

[a) Thuật Toán Tìm kiếm theo chiều sâu - Depth-First Search 4](#_Toc90583830)

[b) Thuật Toán Tìm kiếm theo chiều rộng - Breadth First Search 4](#_Toc90583831)

[c) Thuật Toán Tìm đường đi ngắn nhất 4](#_Toc90583832)

[CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ LỚP 5](#_Toc90583833)

[2.1. Phân Tích Bài Toán Quản Lý Lộ Trình Đường Đi 5](#_Toc90583834)

[2.2. Sơ Đồ Lớp 5](#_Toc90583835)

[2.3. Cài Đặt Lớp 5](#_Toc90583836)

[CHƯƠNG 3. THIẾT KẾ GIAO DIỆN 6](#_Toc90583837)

[3.1. Giao Diện Menu Chính 6](#_Toc90583838)

[3.2. Chi Tiết Chức Năng 6](#_Toc90583839)

[CHƯƠNG 4. THẢO LUẬN & ĐÁNH GIÁ 7](#_Toc90583840)

[5.1. Các Kết Quả Nhận Được 7](#_Toc90583841)

[5.2. Một Số Tồn Tại 7](#_Toc90583842)

[5.1. Hướng Phát Triển 7](#_Toc90583843)

[PHỤ LỤC 8](#_Toc90583844)

[TÀI LIỆU THAM KHẢO 9](#_Toc90583845)

# CHƯƠNG 1. ĐỒ THỊ

## 1.1. Các Khái Niệm Liên Quan

* Đồ thị (Graph) là một dạng cấu trúc dữ liệu được cấu tạo từ tập các đỉnh V (vertex) và tập các cạnh E (edge): G=(V, E).
* Hai đỉnh được nối với nhau tạo thành một cặp đỉnh (pair). Mỗi cặp đỉnh như vậy tạo thành một cạnh. Một cạnh được tạo từ một đỉnh duy nhất gọi là khuyên. Nếu tồn tại hai cạnh trên cùng một cặp điểm, thì cặp cạnh đó gọi là cặp cạnh song song (hoặc cạnh bội).
* Một đồ thì mà trong đó các cặp đỉnh là các bộ sắp thứ tự được gọi Đồ thị có hướng (Digraph). Ngược lại, được gọi là Đồ thị vô hướng.
* Nếu mỗi cạnh của đồ thị được gán với một trọng số (weight), thì đồ thị đó được gọi là Đồ thị có trọng số (hoặc là mạng lưới network).
* Đơn đồ thị là đồ thị (có hướng hoặc vô hướng) KHÔNG chứa khuyên hoặc cạnh bội.
* Để biểu diễn đồ thị: biển diễn hình học, biểu diễn bằng ma trận liền kề.

## 1.2. Cấu Trúc và Cài Đặt Đồ Thị

|  |  |
| --- | --- |
| **Lớp Đỉnh Vertex**  public class **Vertex**  {  public bool wasVisited;  public string label;  public Vertex(string label)  {  this.label = label;  wasVisited = false;  }  } | |
| public class **Graph**  {  private const int NUM\_VERTICES = 20;  private Vertex[] vertices;  private int[,] adjMatrix;  int numVerts;  public Graph()  {  vertices = new **Vertex**[NUM\_VERTICES];  adjMatrix = new int[NUM\_VERTICES, NUM\_VERTICES];  numVerts = 0;  for (int j = 0; j < NUM\_VERTICES; j++)  for (int k = 0; k < NUMVERTICES; k++)  adjMatrix[j, k] = 0;  } | public void AddVertex(string label)  {  vertices[numVerts] = new **Vertex**(label);  numVerts++;  }  public void AddEdge(int start, int eend)  {  adjMatrix[start, eend] = 1;  adjMatrix[eend, start] = 1;  }  public void ShowVertex(int v)  {  Console.Write(vertices[v].label + " ");  }  } |

## 1.3. Các Thuật Toán Trên Đồ Thị

### a) Thuật Toán Tìm kiếm theo chiều sâu - Depth-First Search

|  |  |
| --- | --- |
| private int GetAdjUnvisitedVertex(int v)  {  for (int j = 0; j <= numVertices - 1; j++)  if((adjMatrix(v,j)==1) &&  (vertices[j].WasVisited==false))  return j;  return -1;  }  public void DepthFirstSearch()  {  vertices[0].WasVisited = true;  ShowVertex(0);  Stack gStack = new Stack(); gStack.Push(0);  int v; | while (gStack.Count > 0)  {  v = GetAdjUnvisitedVertex(gStack.Peek());  if (v == -1)  gStack.Pop();  else  {  vertices[v].WasVisited = true;  ShowVertex(v);  gStack.Push(v);  }  }  for (int j = 0; j <= numVertices - 1; j++)  vertices[j].WasVisited = false;  } |

### b) Thuật Toán Tìm kiếm theo chiều rộng - Breadth First Search

|  |  |
| --- | --- |
| public void BreadthFirstSearch()  {  Queue gQueue = new Queue();  vertices[0].WasVisited = true;  ShowVertex(0);  gQueue.EnQueue(0);  int vert1, vert2;  while (gQueue.Count > 0)  {  vert1 = gQueue.Dequeue();  vert2 = GetAdjUnvisitedVertex(vert1); | while (vert2! = -1)  {  vertices[vert2].WasVisited = true;  ShowVertex(vert2);  gQueue.Enqueue(vert2);  vert2 = GetAdjUnvisitedVertex(vert1);  }  }  for (int i = 0; i <= numVertices - 1; i++)  vertices[index].WasVisited = false;  } |

### c) Thuật Toán Tìm đường đi ngắn nhất

|  |  |
| --- | --- |
| public void **Path**()  {  int startTree = 0;  vertexList[startTree].isInTree = true;  nTree = 1;  for (int j = 0; j <= nVerts; j++)  {  int tempDist = adjMat[startTree, j];  sPath[j] = new DistOriginal(startTree, tempDist);  }  while (nTree < nVerts)  {  int indexMin = GetMin(); | int minDist = sPath[indexMin].distance;  currentVert = indexMin;  startToCurrent = sPath[indexMin].distance;  vertexList[currentVert].isInTree = true;  nTree++;  AdjustShortPath();  }  DisplayPaths();  nTree = 0;  for (int j = 0; j <= nVerts - 1; j++)  vertexList[j].isInTree = false;  } |

# CHƯƠNG 2. PHÂN TÍCH VÀ THIẾT KẾ LỚP

## 2.1. Phân Tích Bài Toán Quản Lý Lộ Trình Đường Đi

## 2.2. Sơ Đồ Lớp

## 2.3. Cài Đặt Lớp

# CHƯƠNG 3. THIẾT KẾ GIAO DIỆN

## 3.1. Giao Diện Menu Chính

## 3.2. Chi Tiết Chức Năng

# CHƯƠNG 4. THẢO LUẬN & ĐÁNH GIÁ

## 5.1. Các Kết Quả Nhận Được

## 5.2. Một Số Tồn Tại

## 5.1. Hướng Phát Triển

# PHỤ LỤC

- Đưa Link Toàn Bộ Mã Nguồn Lên GitHub và Đưa Link Vào Đây

- Viết Hướng Dẫn Cách Cài Đặt Để Chạy

- Phân Công Công Việc

|  |  |
| --- | --- |
| **Thành Viên** | **Nhiệm Vụ** |
|  |  |
|  |  |
|  |  |
|  |  |

**Lưu ý**: nhiệm vụ cần phân công rõ ràng, ko có kiểu là “cùng nhau” làm cái này !

# TÀI LIỆU THAM KHẢO

1. xyz

2. xyz