Môn: Lý thuyết đồ thị

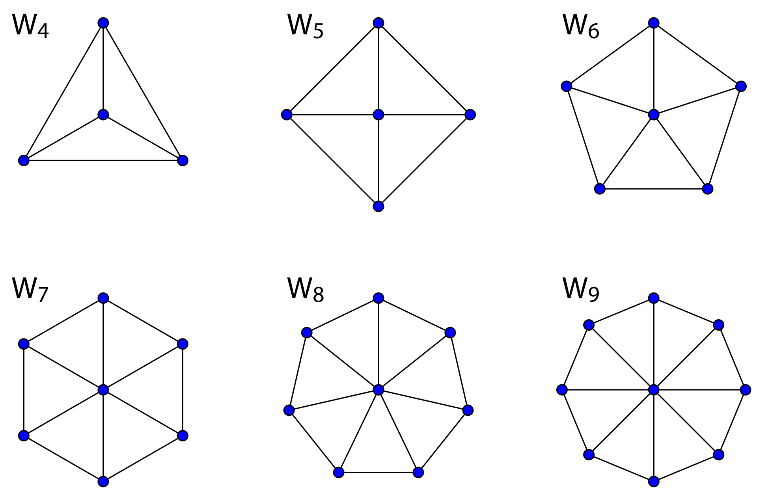
Họ và tên/MSSV: Đặng Đức Trường / 20880108

Họ và tên/MSSV: Trần Thị Tuyết Ngân / 21850013

# ĐỒ ÁN 01

## Phần lý thuyết: Lựa chọn và trình bày tóm tắt các đặc trưng của đồ thị bất kì

* Loại đồ thị lựa chọn: Đồ thị bánh xe (Wheel graph)
* Mô tả tóm tắt: Đồ thị bánh xe là một đơn đồ thị vô hướng, được tạo thành bởi một đồ thị vòng và một đỉnh trung tâm được nối đến tất cả các đỉnh của đồ thị vòng.
* Hình mô tả:



Đồ thị bánh xe có thông số đặc trưng là **Wn**, biểu thị cho tổng số đỉnh có trong đồ thị. Ví dụ: **W6** có nghĩa là đồ thị bánh xe có **n = 6** đỉnh, “**W**” là viết tắt cho từ “**W**heel”.

* Tính đối xứng hình học của đồ thị: Ta đặt điều kiện là tất cả các đỉnh của đồ thị bánh xe cùng nằm trong 1 mặt phẳng, các cạnh nối từ đỉnh tâm ra các đỉnh bên ngoài (cạnh căm xe) và cạnh nối các đỉnh trên vành bánh xe (cạnh vành) là bằng nhau, khi đó ta luôn có một đường thẳng chứa tâm và một đỉnh bất kì là đường đối xứng, chia đồ thị ra làm 2 phần chứa số lượng đỉnh bằng nhau (không bao gồm các đỉnh nằm trên đường đối xứng). Ngoài ra, với đồ thị bánh xe có số đỉnh n lẻ, tâm đồ thị cũng đồng thời là điểm đối xứng, với 1 đỉnh trên vành bánh xe bất kì, ta luôn tìm dc duy nhất 1 đỉnh đối xứng với nó qua tâm của đồ thị bánh xe.
* Tính đối xứng của ma trận kề: Do đồ thị bánh xe là đơn đồ thị, vô hướng nên ma trận kề của đồ thị bánh xe là ma trận đối xứng qua đường chéo chính của ma trận kề.
* Tính phân hoạch của đồ thị: đồ thị đa phân, tồn tại 3 phân hoạch khi số đỉnh n lẻ và 4 phân hoạch khi n chẵn.
* Một số tính chất khác, với đồ thị bánh xe có n đỉnh:
  + Đồ thị bánh xe không chứa chu trình Euler do đồ thị có chứa đỉnh bậc lẻ (theo Định lý Euler về chu trình và đường đi Euler)
  + Tổng số chu trình con trong đồ thị bánh xe là n2 – 3n + 3.
  + Tồn tại 2(n – 1)(n – 2) đường đi Hamilton.

## Phần cài đặt

### Hướng dẫn sử dụng chương trình

* **Bước 1**: đặt các file input của phần cài đặt vào đường dẫn ~\Project01\DataInput\
* **Bước 2**: mở project bằng Visual Studio (khuyến nghị phiên bản 2019)
* **Bước 3**: Chọn “Build -> Build solution” hoặc chạy project ở chế độ Debug, màn hình điều khiển chính xuất hiện

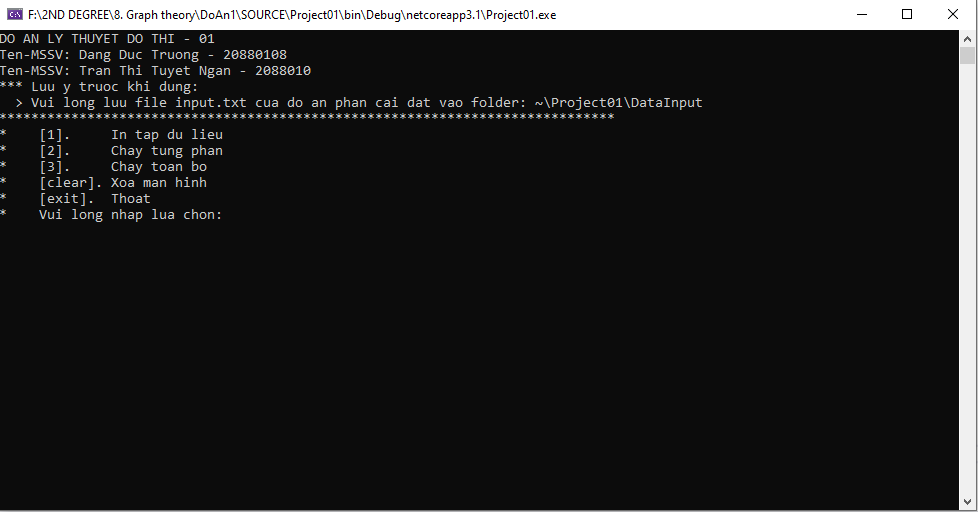


Figure : Giao diện chính

* **Bước 4**: Nhập vào các giá trị chuỗi để tiến hành chạy chương trình:
  + ‘1’: nạp dữ liệu trong tập tin input.txt vào bộ nhớ và in kết quả ra màn hình
  + ‘2’: chạy tác vụ xử lý với 1 đồ thị được chọn trong tập đồ thị đầu vào hợp lệ và in kết quả ra màn hình
  + ‘3’: chạy tác vụ xử lý toàn bộ các đồ thị đầu vào hợp lệ và in kết quả ra màn hình
  + ‘clear’: xóa nội dung kết quả của các lượt chạy trước đang hiển thị trên màn hình console
  + ‘exit’ hoặc các chuỗi ký tự khác bất kì: thoát chương trình

### Cấu trúc tổng quát của chương trình

#### Mô tả khái quát thành phần

|  |  |  |
| --- | --- | --- |
| **STT** | **Tên module** | **Mô tả** |
| 1 | Program.cs | Phần thân chính của chương trình, chứa phương thức main |
| 2 | AdjList.cs | Lớp mô tả đối tượng danh sách kề của đồ thị |
| 3 | Party.cs | Lớp mô tả đối tượng nhóm phân hoạch trong 1 đồ thị |
| 4 | Vertex.cs | Lớp mô tả đối tượng đỉnh của đồ thị |
| 5 | DataIOHelper.cs | Lớp chứa các phương thức hỗ trợ nhập xuất và kiểm tra tính hợp lệ của dữ liệu |
| 6 | GraphSearch.cs | Lớp chứa các phương thức xử lý phân loại đồ thị |
| 7 | Input.txt | File chứa dữ liệu đầu vào |

#### Mô tả chi tiết

|  |  |  |
| --- | --- | --- |
| **Tên module** | **Tên thuộc tính/Phương thức** | **Đặc tả** |
| Program.cs | static void Main | Đây là phân thân chính của chương trình, thực hiện các tác vụ:   * Xuất ra màn hình các thông tin bài làm * Cho phép người dùng nhập các lệnh muốn thực hiện * In ra màn hình kết quả sau khi thực thi |
| AdjList.cs | private List<Vertex> verticesList | Biến lưu danh sách các đỉnh của đồ thị |
|  | private int id; | Biến lưu mã số của đồ thị, mỗi đồ thị trong 1 danh sách tập đồ thị sẽ có 1 mã số tương ứng và là duy nhất. |
|  | public void setId(int id) / public int getId() | Phương thức get và set id của 1 đối tượng AdjList |
|  | public AdjList() / public AdjList(List<Vertex> \_verticesList) | Phương thức constructor của đối tượng AdjList |
|  | public List<Vertex> getVerticesList() | Phương thức lấy toàn bộ danh sách đỉnh hiện có của 1 AdjList |
|  | public void addVertex(Vertex vertex) | Phương thức thêm 1 đỉnh vào danh sách đỉnh của 1 AdjList |
|  | public int getVertexCount() | Phương thức trả về tổng số đỉnh của 1 AdjList |
| Party.cs | private List<Vertex> members; | Danh sách các đỉnh thành phần của 1 phân hoạch |
|  | public Party() / public Party(Vertex vertex) | Phương thức constructor của 1 đối tượng Party |
|  | public void addMember(Vertex vertex) | Phương thức thêm 1 đỉnh vào danh sách thành phần |
|  | public List<Vertex> getMembers() | Phương thức trả về danh sách thành phần của 1 phân hoạch |
| Vertex | private int id; | Biến lưu mã số của 1 đỉnh, mỗi đỉnh trong 1 đồ thị sẽ có 1 mã số tương ứng và là duy nhất. |
|  | private List<int> neighbors; | Danh sách lưu mã số của các đỉnh kề với 1 đỉnh trong 1 đồ thị |
|  | public Vertex() | Phương thức constructor |
|  | public void setId(int id) / public int getId() | Phương thức get và set id của 1 đối tượng Vertex |
|  | public int getDegree() | Phương thức đếm bậc của 1 đỉnh dựa vào số lượng đỉnh trong danh sách đỉnh kề. |
|  | public void addNeighbor(int neighborVertexId) | Phương thức thêm 1 đỉnh vào danh sách đỉnh kề của 1 đỉnh |
|  | public string neighborToString() | Phương thức trả về danh sách đỉnh kề dạng chuỗi |
| DataIOHelper.cs | static List<string> readRawContentFromFile(string filePath) | Phương thức đọc nội dung dạng thô từ file input.txt, mỗi 1 dòng trong file ứng với 1 đối tượng string trong danh sách dữ liệu thô. |
|  | public static List<AdjList> fetchAdjLists(string filePath) | Phương thức chuyển đổi dữ liệu thô dạng List<string> thành List<AdjList>. Phương thức sẽ nạp từng danh sách kề vào bộ nhớ. Nếu xảy ra lỗi ở 1 danh sách kề nào, thì toàn bộ phần danh sách kề bị lỗi và phần tiếp theo sẽ không được nạp vào bộ nhớ. Sẽ xảy ra lỗi trong quá trình nạp dữ liệu nếu: - Số lượng danh sách kề thực tế không khớp với số được khai báo ở đầu file  - Dữ liệu thô có chứa kí tự không phải dạng số nguyên  - Danh sách kề có đỉnh kề trùng nhau hoặc có số đỉnh kề thực tế khác với số được khai báo ở đầu dòng  - Số lượng đỉnh của 1 danh sách kề không khớp với số đỉnh được khai báo ở đầu danh sách kề  Dòng dữ liệu không hợp lệ cũng sẽ được chỉ ra để hỗ trợ người dùng kiểm tra lại. |
|  | public static void printRawData(List<string> rawData) | Phương thức in dữ liệu thô ra màn hình (dùng để kiểm tra) |
|  | public static void printDataSet(List<AdjList> dataset) | Phương thức in các danh sách kề được nạp thành công ra màn hình |
| GraphService.cs | public static int countVertexWithDegree(List<Vertex> verticesList, int degree) | Phương thức đếm số lượng đỉnh có bậc bất kì được truyền vào qua biến int degree trong 1 tập đỉnh. |
|  | public static Vertex getMaxDegreeVertex(List<Vertex> verticesList) | Phương thức trả về đỉnh có bậc lớn nhất trong 1 tập đỉnh |
|  | public static bool isAllVisited(bool[] visitedList) | Phương thức kiểm tra xem 1 danh sách đỉnh đã được duyệt hết hay chưa. Trả về true nếu đã duyệt hết và false nếu còn đỉnh chưa duyệt. |
|  | public static void checkIfGraphIsEmptyGraph(AdjList adjList) | Phương thức kiểm tra và in kết quả xét 1 đồ thị có phải là đồ thị trống (Empty Graph) từ một danh sách kề được nạp vào. Một đồ thị trống sẽ có tập đỉnh với toàn đỉnh bậc 0. Từ đó ta tiến hành đếm số đỉnh bậc 0 và in ra kết quả. |
|  | public static void checkIfGraphIsCycleGraph(AdjList adjList) | Phương thức kiểm tra và in kết quả xét 1 đồ thị có phải là đồ thị vòng (Cycle graph) thông qua một danh sách kề được nạp vào. Một đồ thị vòng sẽ có tập đỉnh với toàn đỉnh bậc 2 và chỉ có 1 thành phần liên thông. Phương thức này sẽ đếm số lượng đỉnh bậc 2 và kiểm tra số lượng thành phần liên thông để trả về kết quả. |
|  | public static void checkIfGraphIsButterflyGraph(AdjList adjList) | Phương thức kiểm tra và in kết quả xét 1 đồ thị có phải là đồ thị hình con bướm (Butterfly graph) thông qua một danh sách kề được nạp vào. Một đồ thị hình con bướm sẽ có tập đỉnh với đúng 4 đỉnh bậc 2 và chỉ có 1 đỉnh bậc 4. Phương thức này sẽ đếm số lượng đỉnh bậc 2 và 4 để trả về kết quả. |
|  | public static void checkIfGraphIsMothGraph(AdjList adjList) | Phương thức kiểm tra và in kết quả xét 1 đồ thị có phải là đồ thị hình con ngài (Moth graph) thông qua một danh sách kề được nạp vào. Một đồ thị hình con ngài sẽ có tập đỉnh với đúng 2 đỉnh bậc 1, 2 đỉnh bậc 2, 1 đỉnh bậc 3 và 1 đỉnh bậc 4. Phương thức này sẽ đếm số lượng đỉnh có bậc tương ứng để trả về kết quả. |
|  | public static void checkIfGraphIsStarGraph(AdjList adjList) | Phương thức kiểm tra và in kết quả xét 1 đồ thị có phải là đồ thị hình sao (Star graph) thông qua một danh sách kề được nạp vào. Một đồ thị hình sao n đỉnh sẽ có tập đỉnh với 1 đỉnh trung tâm có bậc n – 1 và có n – 1 đỉnh bậc 1. Phương thức này sẽ đếm số lượng đỉnh có bậc tương ứng để trả về kết quả. |
|  | public static void checkIfGraphIsWheelGraph(AdjList adjList) | Phương thức kiểm tra và in kết quả xét 1 đồ thị có phải là đồ thị bánh xe (Wheel graph) thông qua một danh sách kề được nạp vào. Một đồ thị bánh xe n đỉnh sẽ có tập đỉnh với 1 đỉnh trung tâm có bậc n – 1, có n – 1 đỉnh bậc 3 và đồng thời n – 1 đỉnh này phải tạo thành 1 chu trình duy nhất. Phương thức này sẽ đếm số lượng đỉnh có bậc tương ứng và xét số chu trình để trả về kết quả. |
|  | public static void checkIfGraphIsBarbellGraph(AdjList adjList) | Phương thức kiểm tra và in kết quả xét 1 đồ thị có phải là đồ thị Barbell (Barbell graph) thông qua một danh sách kề được nạp vào. Một đồ thị Barbell n đỉnh sẽ có tập đỉnh với 2 đỉnh nối có bậc (n / 2), có n – 2 đỉnh bậc (n / 2) - 1. Phương thức này sẽ đếm số lượng đỉnh có bậc tương ứng và xét số chu trình để trả về kết quả. |
|  | public static void checkIfGraphIsFriendshipGraph(AdjList adjList) | Phương thức kiểm tra và in kết quả xét 1 đồ thị có phải là đồ thị tình bạn (Friendship graph) thông qua một danh sách kề được nạp vào. Một đồ thị Tình bạn n (n là số lẻ) đỉnh sẽ có tập đỉnh với n - 1 đỉnh bậc 2, có 1 đỉnh bậc (n - 1). Phương thức này sẽ đếm số lượng đỉnh có bậc tương ứng và xét số chu trình để trả về kết quả. |
|  | public static void graphPartitioning(AdjList adjList) | Phương thức kiểm tra và in ra kết quả xét đồ thị k-phân (k-partite graph, k > 1) dựa trên danh sách kề của 1 đồ thị. Phương thức này sẽ lần lượt xét qua từng đỉnh của tập đỉnh, đối chiếu đỉnh đó với từng nhóm trong phân hoạch. Nếu trong 1 nhóm không có bất kì đỉnh nào kề với đỉnh đang xét thì đỉnh đó sẽ được nạp vào nhóm đó, ngược lại thì xét nhóm kế tiếp. Nếu sau khi xét toàn bộ các nhóm mà vẫn không nạp được đỉnh đang xét vào bất kì nhóm nào thì sẽ tiến hành tạo 1 nhóm mới và thêm đỉnh đang xét vào nhóm mới đó. Tiếp tục lặp lại cho các đỉnh còn lại. Sau cùng, số k chính là số lượng nhóm, và mỗi nhóm này sẽ được lưu danh sách các đỉnh thành phần. |
|  | private static void printPartite(List<Party> partite) | Hàm hỗ trợ in danh sách phân hoạch ra màn hình. |

--- Hết ---