BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

NGUYỄN VĂN NHÂN

ỨNG DỤNG ĐỒ THỊ EULER TỐI ƯU HÓA BÀI TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT

LUẬN VĂN THẠC SĨ

Chuyên ngành: Công Nghệ Thông Tin

Mã số ngành: 60480201

TP. HỒ CHÍ MINH, 17 tháng 10 năm 2015

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

NGUYỄN VĂN NHÂN

ỨNG DỤNG ĐỒ THỊ EULER TỐI ƯU HÓA BÀI TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT

LUẬN VĂN THẠC SĨ

Chuyên ngành: Công Nghệ Thông Tin

Mã số ngành: 60480201

CÁN BỘ HƯỚNG DẪN KHOA HỌC: PGD TSKH NGUYỄN XUÂN HUY

TP. HỒ CHÍ MINH, 17 tháng 10 năm 2015

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

Cán bộ hướng dẫn khoa học: **PGS TSKH NGUYỄN XUÂN HUY**

Luận văn Thạc sĩ được bảo vệ tại Trường Đại học Công nghệ TP. HCM ngày 17 tháng 10 năm 2015.

Thành phần Hội đồng đánh giá Luận văn Thạc sĩ gồm:

TT	Họ và Tên	Chức danh Hội đồng
1		Chủ tịch
2		Phản biện 1
3		Phản biện 2
4		Ủy viên
5		Ủy viên, Thư ký

Xác nhận của Chủ tịch Hội đồng đánh giá Luận văn sau khi Luận văn đã sửa chữa (nếu có).

Chủ tịch Hội đồng đánh giá LV

TRƯỜNG ĐH CÔNG NGHỆ TP. HCM
PHÒNG QLKH – ĐTSĐH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

TP. HCM, ngày..... tháng 10 năm 2015

NHIỆM VỤ LUẬN VĂN THẠC SĨ

Họ tên học viên : Nguyễn Văn Nhân Giới tính : Nam.
Ngày, tháng, năm sinh : 04 / 08 / 1980 Nơi sinh : Tây Ninh.
Chuyên ngành : Công Nghệ Thông Tin MSHV : 1341860047

I - Tên đề tài: ỨNG DỤNG ĐỒ THỊ EULER TỐI ƯU HÓA BÀI TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT

II- Nhiệm vụ và nội dung:

- Tìm hiểu lĩnh vực Lý thuyết đồ thị, một số khái niệm cơ bản.
- Tìm hiểu các thuật toán tìm kiếm tối ưu trên đồ thị.
- Tìm hiểu đồ thị Euler, các biến thể và ứng dụng liên quan.
- Nghiên cứu ứng dụng đồ thị Euler tối ưu cho bài toán tìm đường đi ngắn nhất trên đồ thị.
- Cài đặt thử nghiệm ứng dụng cho bài toán đề xuất.

III - Ngày giao nhiệm vụ: 03/04/2014

IV- Ngày hoàn thành nhiệm vụ: 31/08/2015

V- Cán bộ hướng dẫn: PGS TSKH NGUYỄN XUÂN HUY

CÁN BỘ HƯỚNG DẪN
(Họ tên và chữ ký)

KHOA QUẢN LÝ CHUYÊN NGÀNH
(Họ tên và chữ ký)

NGUYỄN XUÂN HUY

LỜI CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi. Các số liệu, kết quả đánh giá, nhận xét và các đề xuất cải tiến mới nêu trong Luận văn là trung thực và chưa từng được ai công bố trong bất kỳ công trình nào khác.

Tôi xin cam đoan rằng mọi sự giúp đỡ cho việc thực hiện Luận văn này cũng như các trích dẫn hay tài liệu học thuật tham khảo đã được cảm ơn đến tác giả hay ghi rõ ràng nguồn gốc thông tin trích dẫn trong Luận văn.

Học viên thực hiện Luận văn

NGUYỄN VĂN NHÂN

LỜI CẢM ƠN

Trước hết, cho tôi được gửi lời cảm ơn đến sự hướng dẫn và giúp đỡ tận tình của PGS TSKH Nguyễn Xuân Huy.

Xin cảm ơn TS. Võ Đình Bảy, TS. Cao Tùng Anh, TS. Bùi Đức Minh, các Thầy/Cô tại trường Đại học Công nghệ Thành phố Hồ Chi Minh cùng các đồng nghiệp tại Trung tâm Công nghệ thông tin Ngân hàng Xây Dựng đã sát cánh cùng tôi và cung cấp cho tôi những kiến thức quí báu trong suốt thời gian học tập và nghiên cứu thực hiện luận văn.

Tôi cũng xin gởi lời cảm ơn đến gia đình, bạn bè và những người thân đã luôn quan tâm và giúp đỡ tôi trong suốt thời gian học tập và nghiên cứu hoàn thành luận văn này.

Luận văn không thể tránh khỏi những sai sót, rất mong nhận được ý kiến đóng góp của mọi người để luận văn được hoàn thiện hơn.

Tôi xin chân thành cảm ơn.

TP. Hồ Chí Minh, tháng 10 năm 2015

NGUYỄN VĂN NHÂN

TÓM TẮT

Bài toán phân công một xe thực hiện hành trình công việc đi qua tất cả các con đường cho trước như: thu gom rác thải, tưới nước cây xanh, tuần tra giao thông, chuyển phát thư từ ... Đề bài yêu cầu nơi xe xuất phát và quay về là tại Công sở, tổng chiều dài đường đi của xe là ngắn nhất có thể.

Khi đó, bản đồ đường đi sẽ được mô hình hoá bằng đồ thị vô hướng liên thông, có cạnh biểu diễn các con đường, trọng số là chiều dài của con đường và đỉnh là các điểm giao lộ, khi đó bài toán thực hiện tìm đường đi ngắn nhất trên đồ thị được mô phỏng.

Trong trường hợp tốt nhất, hành trình mà xe đi qua mỗi con đường đúng một lần là hành trình ngắn nhất, đây thực chất là chu trình Euler. Khi đó, đồ thị đầu vào chắc chắn phải thoả định lý Euler là tất cả các đỉnh đều có bậc chẵn. Chỉ cần áp dụng thuật toán duyệt đồ thị Euler để in ra đường đi ngắn nhất.

Trong thực tế, các giao lộ thường có bậc là lẻ như: ngã ba, ngã năm... Vì vậy, đồ thị đầu vào sẽ không thoả định lý Euler. Trong trường hợp này, một số con đường sẽ phải đi lại hai lần. Vấn đề là chúng ta cần lựa chọn được con đường nào nên đi lại hai lần để tổng chiều dài của cả chu trình là ngắn nhất có thể. Việc con đường nào được chọn để đi lại hai lần được mô hình hoá là cạnh đó trên đồ thị sẽ được vẽ hai nét.

Sự thật là các đỉnh có bậc lẻ trên đồ thị luôn là số chẵn. Vậy nên nếu có nhiều hơn 2 đỉnh bậc lẻ thì chúng ta phải tìm cách vẽ thêm nét nối các đỉnh bậc lẻ này để đỉnh này trở thành đỉnh có bậc chẵn. Khi đó, đồ thị sẽ thoả mãn định lý Euler là tất cả các đỉnh đều có bậc chẵn, chỉ cần áp dụng thuật toán Fleury để duyệt đồ thị và in ra chu trình Euler.

Bước quan trọng nhất của thuật toán là tìm các gặp ghép tối ưu giữa các đỉnh bậc lẻ sao cho tổng chiều dài là ngắn nhất có thể. Luận văn đề xuất 02 giải pháp để tìm bộ ghép tối ưu có tổng chi phí nhỏ nhất là giải thuật Tham lam và giải thuật FindMinMatch, phân tích và đánh giá ưu và nhược điểm của 02 giải thuật này để đưa ra kiến nghị tuỳ chọn áp dụng cho mục đích khác nhau với từng trường hợp cụ thể.

ABSTRACT

The shortest path may be used to solve many problem in the real life. For example, path of garbage truck, watering car, checking traffic, mail delivering… in the local map, with start and finish is the same place.

The map can be represeted by a connected graph, where edges represent streets and vertices - crossroads between them. With this modeling the problem can be formulated to searching a path on the graph which go through all edges at least once such that the total length is minimal.

In the best case, the path goes through all edges exactly once, it mean all vertices have even degree then we know that there exists an Euler path, and finding the solution of our problem amounts to giving such an Euler path in the graph

Map in the real life can have vertices with odd degrees. This means that it doesn't have an Euler path, so finding a solution of our problem amounts to finding which edges will be followed multi times. To modeling this we will build an extension of our graph by duplicating edges such that the extended graph does have all his vertices of even degree. Our goal will be to minimize the cost of the duplicated edges.

We known a graph can only have an even number of odd degree vertices. Indeed the sum of degrees over all vertices of the graph is equal to twice the number of edges, which is then an even number, giving a contradiction if we suppose that the number of vertices of odd degree is odd, we use Fleury algorithm to resolve that.

Special step of the algorithm is finding perfect matching of minimum cost. The thesis proposed two solutions to matching with minimum cost are FindMinMatch algorithm and Greedy algorithm, we analyze and evaluate all the strong point and weakpoint of these algorithms to make recommendations preferences apply different purposes for each case.

MỤC LỤC

LỜI CAM ĐOAN i
LỜI CẢM ƠN ii
TÓM TẮT iii
ABSTRACT iv
MỤC LỤC v
DANH MỤC CÁC TỪ VIẾT TẮT vii
DANH MỤC CÁC BẢNG viii
DANH MỤC CÁC HÌNH ix
LỜI MỞ ĐẦU 1
Chương 1. ĐẠI CƯƠNG VỀ LÝ THUYẾT ĐỒ THỊ 3
1.1. Đồ thị và các khái niệm liên quan [3] 3
1.1.1. Định nghĩa đồ thị 3
1.1.2. Đồ thị vô hướng, đồ thị có hướng 4
1.1.3. Bậc của đồ thị 5
1.1.4. Một số dạng đồ thị đặc biệt 6
1.1.5.1. Đồ thị đầy đủ 6
1.1.5.2. Đồ thị vòng 6
1.1.5.3. Đồ thị bánh xe 7
1.1.5.4. Đồ thị lập phương 7
1.1.5.5. Đồ thị hai phía 8
1.1.5.6. Đồ thị phẳng 9
1.2. Biểu diễn đồ thị trên máy tính [3] 10
1.2.1. Ma trận kề, ma trận trọng số 10
1.2.2. Danh sách cạnh (cung) 13
1.2.3. Danh sách kề 15
1.3. Chu trình Euler, Đường đi Euler và Đồ thị Euler [3] 16
1.3.1. Khái niệm Đường đi, Chu trình, tính Liên thông trên Đồ thị 16
1.3.2. Khái niệm Chu trình Euler, Đường đi Euler và Đồ thị Euler 18
1.3.3. Thuật toán Fleury tìm chu trình Euler 19
1.4. Một số thuật toán trên Đồ thị 21
1.4.1. Thuật toán Floyed tìm đường đi ngắn nhất giữa mọi cặp đỉnh trên đồ thị 21
1.4.2. Giải thuật Tham lam 24
1.4.3. Tìm bộ ghép trên đồ thị 27
1.4.3.1. Giới thiệu chung 27

1.4.3.2. Bài toán tìm cặp ghép cực đại với tổng trọng số nhỏ nhất 27
1.4.3.3. Bài toán tìm bộ ghép cực đại trọng số nhỏ nhất trên trên đồ thị đầy đủ 28
1.4.3.4. Bài toán Người phát thư Trung Hoa 32
Chương 2. ỨNG DỤNG ĐỒ THỊ EULER TỐI ƯU HÓA BÀI TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT 34
2.1. Phân tích bài toán "Thanh tra giao thông" của tác giả Nguyễn Tam Hùng. 35
2.1.1. Phát biểu bài toán 35
2.1.2. Hướng giải bài toán theo tác giả Nguyễn Tam Hùng 35
2.1.3. Nhận xét về bài toán "Thanh tra giao thông" của tác giả Nguyễn Tam Hùng 38
2.2. Đề xuất bài toán "Phân công xe đi thu gom rác thải" tạ Quận 4 39
2.2.1. Đặt vấn đề 39
2.2.2. Ý tưởng chính của thuật toán 40
2.2.3. Hướng giải quyết bài toán 40
2.2.4. Ứng dụng giải bài toán phân công việc thực tế tại Quận 4 42
Chương 3. ĐÁNH GIÁ 47
3.1. Độ phức tạp của các thuật toán được sử dụng trong bài toán 47
3.2. Đánh giá giải pháp dùng giải thuật Tham lam so với giải thuật FindMinMatch 48
3.2.1. Giải thuật Tham lam 48
3.2.2. Giải thuật FindMinMatch 48
3.2.3. So sánh hiệu quả của 02 giải thuật trên đối với "bài toán phân công xe đi thu gom rác thải" tại Quận 4 48
KẾT LUẬN 50
HƯỚNG PHÁT TRIỂN CỦA LUẬN VĂN 50
TÀI LIỆU THAM KHẢO 52
PHỤ LỤC

DANH MỤC CÁC TỪ VIẾT TẮT

Ký hiệu, viết tắt	Ý nghĩa tiếng Việt	Ý nghĩa tiếng anh
G	Đồ thị	Graph
V	Đỉnh của đồ thị	Vertices
E	Cạnh của đồ thị	Edges
Deg	Bậc của đỉnh trên đồ thị	Degree
GPS	Định vị toàn cầu	Global Positioning System
∞	Vô cực (giá trị không giới hạn)	Infinity (without any limit)
Euler	Nhà toán học và vật lý học Leonhard Euler người Thuỵ Sĩ	Leonhard Euler was a pioneering Swiss mathematician and physicist
Floyd	Thuật toán Floyd-Warshall	Floyd-Warshall algorithm
$\in$	Thuộc, nếu $a \in A$: ta nói a là phần tử con của tập A	Member, if A is a set and a is one of the objects of A, this is denoted $a \in A$
$\notin$	Không thuộc	Not member
>; $\geq$	Lớn hơn; lớn hơn hoặc bằng	Greater than; is greater than or equal to
<; $\leq$	Nhỏ hơn; nhỏ hơn hoặc bằng	Less than; is less than or equal to
{, }	Tập	Set of
$\cap$	Phép giao	Intersected with
$\cup$	Phép hợp	The union of ... or ...
#	Khác	Is incomparable to
$\emptyset$	Tập rỗng	Empty set
| |	Tập tử của tập	Memmer of set
|...|	Ma trận	Matrix
$\rightarrow$	Kéo theo	if ... then
O( )	Ô lớn (độ phức tạp tính toán)	Big O (complexity theory)

DANH MỤC CÁC BẢNG

Bảng 1.1. Biểu diễn đồ thị vô hướng không trọng số bằng ma trận 11
Bảng 1.2. Biểu diễn đồ thị có hướng có trọng số bằng ma trận 12
Bảng 1.3. Biểu diễn đồ thị vô hướng có trọng số bằng ma trận 13
Bảng 1.4. Biểu diễn đồ thị không trọng số bằng danh sách cạnh cung 14
Bảng 1.5. Biểu diễn đồ thị có trọng số bằng danh sách cạnh có trọng số 14
Bảng 1.6. Biểu diễn đồ thị bằng danh sách kề biểu diễn đồ thị.....15
Bảng 1.7. Biểu diễn đồ thị cần tìm bằng thuật toán Floyd bằng ma trận 22
Bảng 1.8. Các bước tìm đường đi ngắn nhất giữa mọi cặp đỉnh bằng Floyd 22
Bảng 1.9. Ma trận biểu diễn đồ thị cần tìm cặp ghép bằng giải thuật Tham lam. ..25
Bảng 1.10. Các bước tìm bộ ghép có trọng số min bằng giải thuật Tham lam 26
Bảng 1.11. Ma trận biểu diễn đồ thị đầy đủ vô hướng có trọng số 29
Bảng 1.12. Ma trận kết quả bộ ghép cực đại có trọng số cực tiểu.....32
Bảng 2.1. Số cạnh nối thêm giữa các cặp đỉnh bậc lẻ 36
Bảng 2.2. Cách chọn cặp đỉnh bậc lẻ và số cạnh nối thêm.....37
Bảng 2.3. Chu trình Euler tìm được với đồ thị G_T.....38
Bảng 2.4. Ma trận đồ thị ban đầu.....42
Bảng 2.5. Ma trận các đỉnh có bậc là lẻ.....43
Bảng 2.6. Ma trận đường đi ngắt nhất giữa các đỉnh bậc lẻ 43
Bảng 2.7. Ma trận chiều dài ngắn nhất giữa các đỉnh có bậc lẻ 44
Bảng 2.8. Ma trận cặp ghép tối ưu có trọng số nhỏ nhất.....44
Bảng 2.9. Ma trận bậc chẵn có được sau khi thêm các cặp ghép 45
Bảng 3.1. So sánh giữa hai giải thuật Tham lam và FindMinMatch về thời gian chạy và giá trị min tìm được.....48

DANH MỤC CÁC HÌNH

Hình 1.1. Các mô hình đồ thị....3
Hình 1.2. Đồ thị hữu hạn....4
Hình 1.3. Các dạng đồ thị....5
Hình 1.4. Bậc của đồ thị vô hướng G....6
Hình 1.5. Đồ thị đầy đủ....6
Hình 1.6. Đồ thị vòng C_1, C_2, C_3....7
Hình 1.7. Đồ thị bánh xe W_1, W_2, W_3....7
Hình 1.8. Đồ thị lập phương Q_1, Q_2, Q_3....8
Hình 1.9. Đồ thị hai phía K_{23}, K_{33}, K_{34}....9
Hình 1.10. Đồ thị phẳng K....9
Hình 1.11. Đồ thị vô hướng không trọng số G....11
Hình 1.12. Đồ thị G có hướng, có trọng số....12
Hình 1.13. Đồ thị vô hướng G có trọng số....13
Hình 1.14. Đồ thị vô hướng không trọng số G....14
Hình 1.15. Đồ thị vô hướng có trọng số G....14
Hình 1.16. Đồ thị vô hướng không trọng số và có trọng số....15
Hình 1.17. Đường đi trên đồ thị....17
Hình 1.18. Đồ thị không liên thông....18
Hình 1.19. Đồ thị có chu trình Euler....19
Hình 1.20. Đồ thị liên thông và có các đỉnh bậc chẵn....20
Hình 1.21. Duyệt đồ thị theo thuật toán Fluery....20
Hình 1.22. Đồ thị Floyd....22
Hình 1.23. Đồ thị đầy đủ có trọng số....29
Hình 1.24. Trường hợp chọn cặp ghép đầu tiên là A-B....30
Hình 1.25. Trường hợp chọn cặp ghép đầu tiên là A-C....31
Hình 1.26. Đồ thị bộ ghép tìm được....32
Hình 1.27. Đồ thị không thoả Euler....34
Hình 1.28. Đồ thị thoả Euler sau khi thêm cạnh....34
Hình 2.1. Đồ thị hành trình thanh tra giao thông....35
Hình 2.2. Đồ thị GT có được khi thêm cạnh (các nét đứt là các cạnh nối thêm)....38
Hình 2.3. Bản đồ giao thông tại quận 4....39

Hình 2.4. Đồ thị mô hình hóa bản đồ quận 4. 40
Hình 2.5. Ví dụ các bước giải bài toán tìm đường đi ngắn nhất 42
Hình 2.6. Mô hình đồ thị Euler sau khi thêm một số con đường đi 2 lần 46
Hình 2.7. Mô hình đường Euler sau khi duyệt bằng Fleury 46

LỜI MỞ ĐẦU

Khái niệm lý thuyết đồ thị được biết đến từ những năm 1736 bởi nhà toán học lừng danh Leonhard Euler, ông đã sử dụng khái niệm lý thuyết đồ thị để đưa ra hướng giải quyết cho bài toán tìm đường đi qua bảy cây cầu ở thành phố Konigsberg. Từ đây, việc dùng lý thuyết đồ thị đã phổ biến hơn, nó giúp cho nhiều nhà toán học mô phỏng và giải quyết rất nhiều bài toán lớn trên thế giới.

Tìm đường đi ngắn nhất là một trong những bài toán kinh điển sử dụng lý thuyết đồ thị để mô phỏng và triển khai giải thuật. Bài toán có tính ứng dụng thực tiễn rất cao, đặc biệt là trong xã hội phát triển ngày nay có rất nhiều ứng dụng được đưa ra theo chủ đề như: hướng dẫn đường đi tự động, ứng dụng truyền dẫn tín hiệu mạng máy tính, đường đi của tín hiệu định vị toàn cầu (gps)…. Ngày nay, nhiều nhà toán học vẫn không ngừng nghiên cứu, để tìm giải pháp tối ưu hơn để giải quyết cho bài toán này.

Chu trình đường đi ngắn nhất qua tất cả các cạnh trên đồ thị liên thông được biết đến là chu trình Euler, được công bố bởi nhà toán học cùng tên. Bài toán sau này có nhiều biến thể được đưa ra bởi nhiều vấn đề hóc búa hơn trong xã hội thực tiễn. Biến thể đầu tiên được nhà toán học Trung Hoa - Quản Mai Cốc đưa ra vào năm 1962 và được Alan Goldman của Cục Tiêu chuẩn quốc gia Hoa Kỳ đặt tên là bài toán *"Người đưa thư Trung Hoa – Chinese postman problem"*. Năm 1973 định lý Good-man-Hedetniemi ra đời nhằm đưa ra một giải pháp để giải bài toán kinh điển này, các biến thể sau nữa như bài toán *"Người quét rác New York - New York Street Sweeper problem"* ….

Phạm vi đề tài đưa ra yêu cầu tìm giải pháp để giải bài toán tìm đường đi ngắn nhất dựa trên đồ thị không thoả chu trình Euler và đề xuất phương pháp giải quyết khác có thể áp dụng trong thực tiễn, kiến thức chủ yếu dựa trên các công trình đã nghiên cứu của nhiều nhà toán học khác với chủ đề tối giản độ phức tạp tính toán.

Một số công trình liên quan đến đồ thị Euler được công gần đây như: Luận

văn Thạc sỹ ngành Khoa học Máy tính của tác giả Nguyễn Tam Hùng, thực hiện vào tháng 5 năm 2014 tại trường Đại học Thái Nguyên với đề tài "*Các thuật toán về đường đi và chu trình Euler và ứng dụng*". Trong đó, tác giả đã áp dụng chu trình Euler giải bài toán tìm đường đi ngắn nhất qua tất cả con đường cho trước dựa trên đồ thị vô hướng, không có trọng số. Với đồ thị đầu vào không thoả định lý Euler, tác giả đã đề xuất áp dụng giải thuật "Tham lam" để tìm bộ ghép tối ưu nhằm đưa đồ thị ban đầu về dạng thoả định lý Euler để giải bài toán.

Ngoài nước có bài báo của các đồng tác giả Gauvain Chaste, Aurélien Ooms và Robin Walravens, công bố ngày 01 tháng 7 năm 2014 với đề tài "*Chinese postman problem",* bài báo này đã đưa một ra hướng giải quyết khác cho bài toán tìm chu trình đường đi ngắn nhất qua tất cả con đường cho trước trên đồ thị vô hướng, có trọng số, với đồ thị đầu vào là một biến thể của đồ Euler. Tác giả đề xuất áp dụng thuật toán "Bông hoa" (Blossom) để tìm bộ ghép tối ưu có tổng trọng số nhỏ nhất nhằm đưa đồ thị ban đầu về dạng thoả định lý Euler để giải bài toán.

Trong đề tài này, ngoài việc tìm hiểu các kiến thức và các công trình nghiên cứu liên quan đã được công bố, đề tài cũng đề xuất hướng giải quyết khác cho bài toán tương tự dựa trên đồ thị vô hướng, có trọng số, đồ thị đầu vào không thoả định lý Euler. Trong đó, áp dụng 02 thuật toán để tìm bộ ghép tối ưu có tổng trọng số nhỏ nhất là giải thuật "Tham lam" và giải thuật "tìm bộ ghép tối ưu có tổng trọng số nhỏ nhất trên đồ thị đầy đủ" nhằm đưa đồ thị ban đầu về dạng thoả định lý Euler để giải bài toán.

Chương 1. ĐẠI CƯƠNG VỀ LÝ THUYẾT ĐỒ THỊ

Lý thuyết đồ thị đã được khoa học phát triển từ rất lâu nhưng lại có rất nhiều ứng dụng hiện đại ngày nay vẫn sử dụng. Đặc biệt trong khoảng vài mươi năm trở lại đây, cùng với sự ra đời của máy tính điện tử và sự phát triển nhanh chóng của tin học, lý thuyết đồ thị càng được quan tâm đến nhiều hơn. Ví dụ như trong kiến trúc mạng máy tính, tổ chức và tìm kiếm dữ liệu trên mạng, chỉ dẫn đường đi ...

Trong chương 1 sẽ trình bày những khái niệm tổng quan cơ bản về lý thuyết đồ thị, cách biểu diễn đồ thị trên máy tính như: định nghĩa một đồ thị, bậc của đồ thị, tính liên thông của đồ thị, đường đi, chu trình của đồ thị.

1.1. Đồ thị và các khái niệm liên quan [3]

1.1.1. Định nghĩa đồ thị

Chúng ta thường xuyên nhìn thấy hoặc đã sử dụng bản đồ giao thông của một thành phố, sơ đồ tổ chức một cơ quan, sơ đồ khối tính toán của một thuật toán, sơ đồ mạng máy tính..., đó chính là những ví dụ cụ thể về đồ thị.

Ví dụ 1.1. Một số dạng của đồ thị trong thực tế

Mạng máy tính Sơ đồ giao thông Mạng nơron

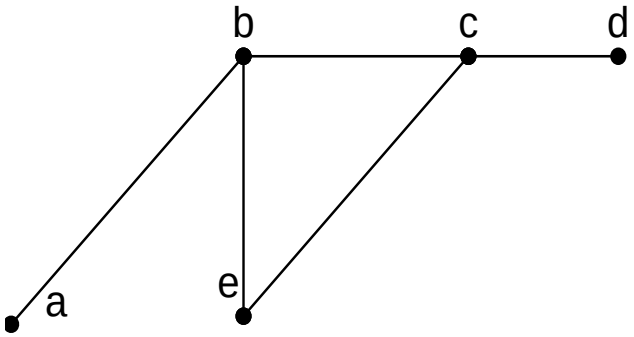
Hình 1.1. Các mô hình đồ thị

Đồ thị (Graph, ký hiệu G): là một cấu trúc rời rạc gồm các đỉnh và các cạnh nối các đỉnh đó. Được mô tả:

G = (V, E), trong đó:

V gọi là tập các *đỉnh* (vertices) và E gọi là tập các *cạnh* (edges). Có thể coi E là tập các cặp (u, v) với u và v là hai đỉnh của V.

Ví dụ 1.2. Xét đồ thị vô hướng bên dưới:



Hình 1.2. Đồ thị hữu hạn

Đồ thị G cho ở hình 1.2 với tập các đỉnh V = {a, b, c, d, e} và tập các cạnh E = {(a, b), (b, c), (b, e), (c, e), (c, d)}.

Nếu (a, b) là một cạnh của đồ thị thì ta nói rằng đỉnh b kề với đỉnh a và cả hai đỉnh a và b kề với cạnh (a, b).

Trong đồ thị ở ví dụ hình 1.2 thì hai đỉnh a và c kề với đỉnh b, ba đỉnh a, c và e kề với đỉnh b. Do vậy, ta có thể định nghĩa đồ thị bằng ánh xạ kề như sau:

Về bản chất, đồ thị là một tập hợp các đối tượng được biểu diễn bằng các đỉnh và giữa các đối tượng này có một quan hệ nhị nguyên biểu diễn bằng các cạnh.

Cặp đỉnh (x, y) $\in$ E không sắp thứ tự được gọi là *cạnh vô hướng*, còn nếu nó có sắp thứ tự thì được gọi là *cạnh có hướng*.

1.1.2. Đồ thị vô hướng, đồ thị có hướng

Đồ thị vô hướng là đồ thị chỉ chứa các cạnh vô hướng (không phân biệt hướng).

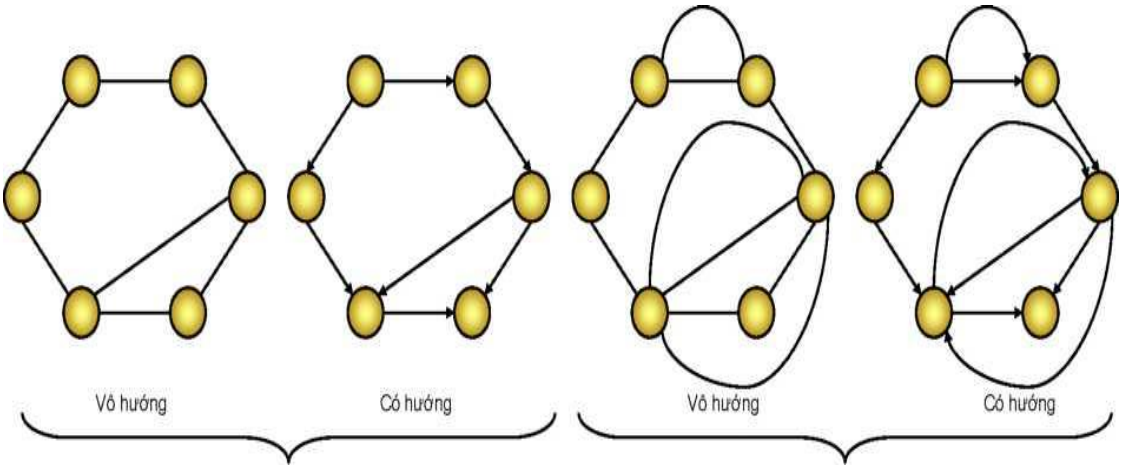
Đồ thị có hướng là đồ thị có chứa các cạnh có hướng.

Hiển nhiên, mỗi đồ thị vô hướng có thể biểu diễn bằng một đồ thị có hướng bằng cách thay mỗi cạnh vô hướng bằng hai cạnh có hướng tương ứng.

Đơn đồ thị: là đồ thị mà mỗi cặp đỉnh được nối với nhau bởi không quá một cạnh (thường gọi tắt là đồ thị).

Đa đồ thị: là đồ thị có những cặp đỉnh được nối với nhau nhiều hơn một cạnh.

Ví dụ 1.3. Một số dạng đồ thị hữu hạn



Đơn đồ thị Đa đồ thị

Hình 1.3. Các dạng đồ thị

Ta có thể biểu diễn hình học cho đồ thị như sau: Trên mặt phẳng, biểu diễn đỉnh bằng các vòng tròn nhỏ, biểu diễn cạnh vô hướng bằng đoạn thẳng, biểu diễn cạnh có hướng bằng mũi tên nối hai đỉnh của đồ thị.

1.1.3. Bậc của đồ thị

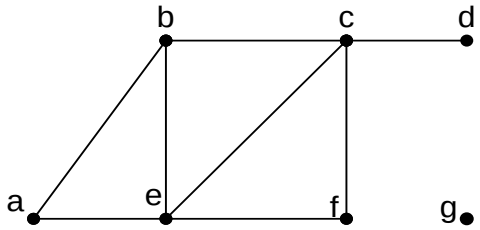
Bậc của đỉnh $v \in V$ là tổng số cạnh liên thuộc với nó và ký hiệu là deg(v).

Nếu đỉnh có khuyên thì mỗi khuyên được tính là 2 khi tính bậc, như vậy:

$$\deg(v) = (\text{số cạnh liên thuộc}) + (2 * \text{Số khuyên})$$

Đỉnh cô lập: trong đồ thị đơn, đỉnh cô lập là đỉnh có bậc bằng 0.

Đỉnh treo: là đỉnh có bậc bằng 1.



Hình 1.4. Bậc của đồ thị vô hướng G.

Ví dụ 1.4. Xét đồ thị trong hình 1.4 trên, ta có:

- deg(a) = deg(f) = 2, deg(b) = 3, deg(c) = deg(e) = 4, deg(d) = 1, deg(g) = 0.

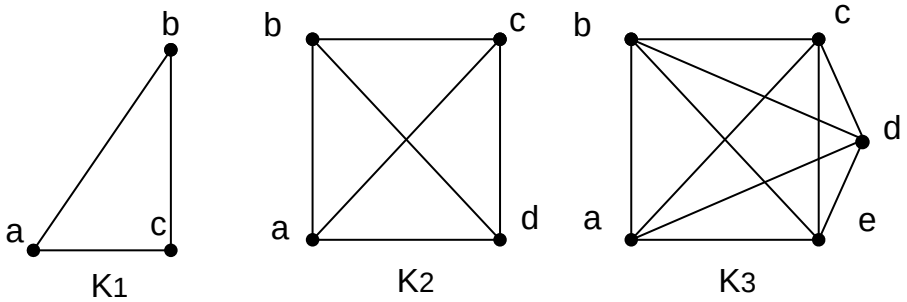
- Đỉnh g là *đỉnh cô lập*, đỉnh d là *đỉnh treo*.

1.1.4. Một số dạng đồ thị đặc biệt

1.1.5.1. Đồ thị đầy đủ

Đồ thị đầy đủ k đỉnh, ký hiệu bởi K_k, là đơn đồ thị vô hướng mà giữa hai đỉnh bất kỳ của nó luôn có cạnh nối.

Ví dụ 1.5. Các đồ thị đầy đủ K_1, K_2, K_3 cho trong hình 1.5



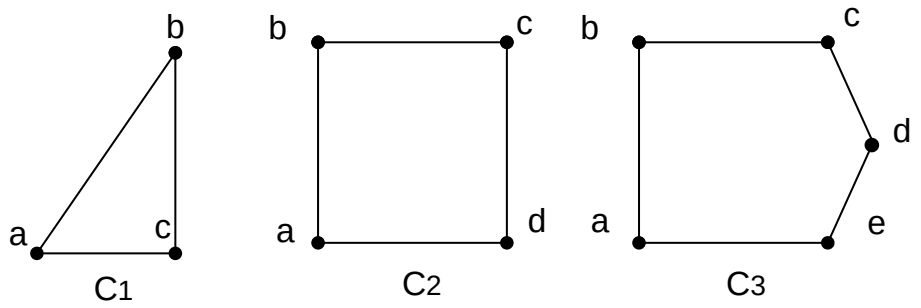
Hình 1.5. Đồ thị đầy đủ

Đồ thị đầy đủ K_k có tất cả k(k-1)/2 cạnh, nó là đơn đồ thị có nhiều cạnh nhất.

1.1.5.2. Đồ thị vòng

Đồ thị vòng C_k, $k \geq 3$, gồm k đỉnh v_1, v_2,....v_k và các cạnh (v_1, v_2), (v_2, v_3)... (v_{k-1}, v_k), (v_k, v_1).

Ví dụ 1.6. Đồ thị vòng C_1, C_2, C_3 như hình 1.6

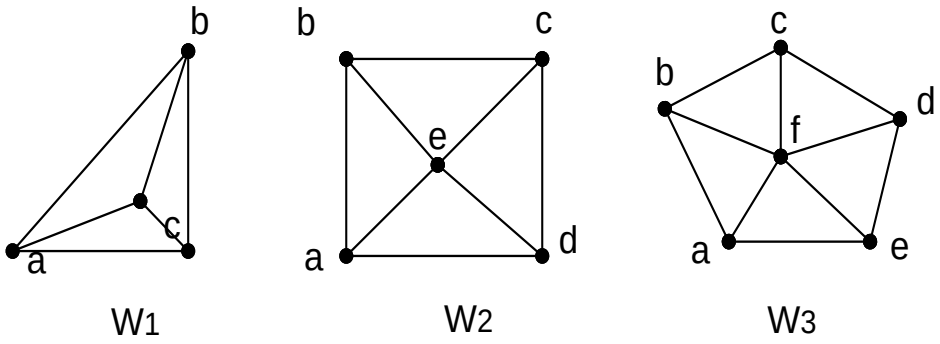


Hình 1.6. Đồ thị vòng C_1, C_2, C_3

1.1.5.3. Đồ thị bánh xe

Đồ thị bánh xe W_k thu được từ C_k bằng cách bổ sung vào một đỉnh mới nối với tất cả các đỉnh của C_k

Ví dụ 1.7. Đồ thị bánh xe W_1, W_2, W_3 như hình 1.7

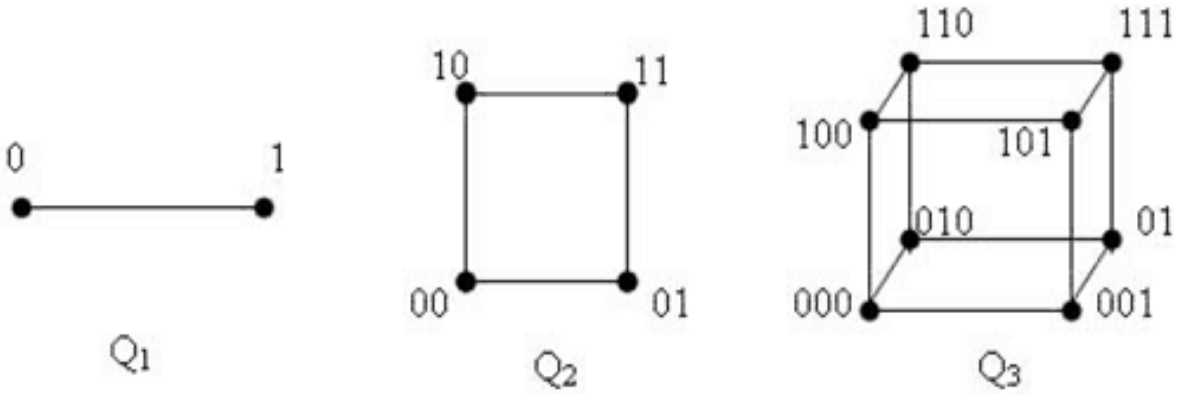


Hình 1.7. Đồ thị bánh xe W_1, W_2, W_3

1.1.5.4. Đồ thị lập phương

Đồ thị lập phương k đỉnh Q_k là đồ thị với các đỉnh biểu diễn 2k xâu nhị phân độ dài k. Hai đỉnh của nó gọi là kề nhau nếu như hai xâu nhị phân tương ứng chỉ khác nhau 1 bit.

Ví dụ 1.8. Đồ thị vòng Q_1, Q_2, Q_3 như hình 1.8



Hình 1.8. Đồ thị lập phương Q_1, Q_2, Q_3

1.1.5.5. Đồ thị hai phía

Đơn đồ thị G=(V,E) được gọi là *đồ thị hai phía* nếu như tập đỉnh V của nó có thể phân hoạch thành hai tập X và Y sao cho mỗi cạnh của đồ thị chỉ nối một đỉnh nào đó trong X với một đỉnh nào đó trong Y. Khi đó ta sẽ sử dụng ký hiệu G=(X∪Y, E) để chỉ đồ thị hai phía với tập đỉnh X∪Y.

Định lý: Đơn đồ thị là đồ thị hai phía khi và chỉ khi nó không chứa chu trình độ dài lẻ.

Để kiểm tra xem một đồ thị liên thông có phải là hai phía hay không ta có thể áp dụng thủ tục sau:

- Cho v là một đỉnh bất kỳ của đồ thị.

- Đặt X={v}, còn Y là tập các đỉnh kề của v. Khi đó các đỉnh kề của các đỉnh trong Y phải thuộc vào X. Ký hiệu tập các đỉnh như vậy là T.

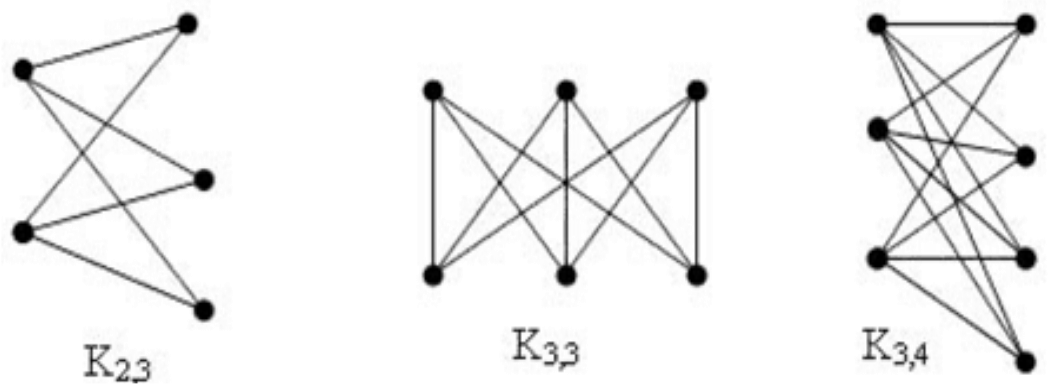
- Vì thế nếu phát hiện T∩Y # ∅ thì đồ thị không phải là hai phía, kết thúc.

- Ngược lại, đặt X=X∩T.

- Tiếp tục xét như vậy đối với T’ là tập các đỉnh kề của T,...

Đồ thị hai phía G=(X∪Y, E) với |X|= m, |Y| = n được gọi là *đồ thị hai phía đầy đủ* và ký hiệu là $K_{2,3}$, $K_{3,3}$, $K_{3,4}$.

Ví dụ 1.9. Các đồ thị 2 phía $K_{2,3}$, $K_{3,3}$, $K_{3,4}$ như hình 1.9

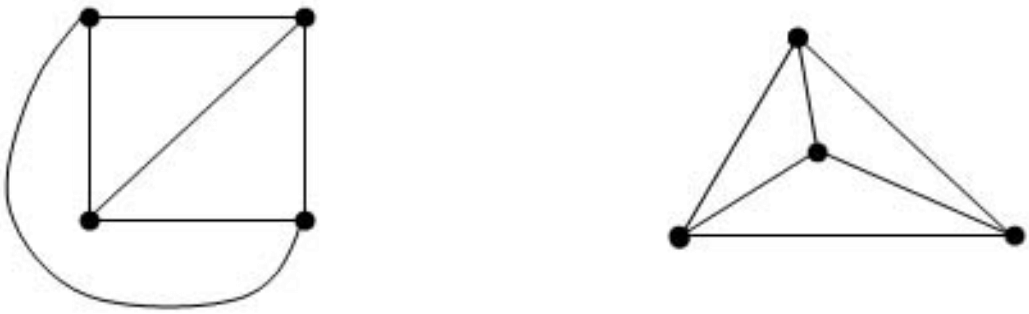


Hình 1.9. Đồ thị hai phía K_{23}, K_{33}, K_{34}

1.1.5.6. Đồ thị phẳng

Đồ thị được gọi là *đồ thị phẳng* nếu ta có thể vẽ nó trên mặt phẳng sao cho các cạnh của nó không cắt nhau ngoài ở đỉnh. Cách vẽ như vậy sẽ được gọi là biểu diễn phẳng của đồ thị.z

Ví dụ 1.10. Đồ thị K như hình 1.10 bên dưới là đồ thị phẳng, vì ta có thể vẽ nó trên mặt phẳng sao cho các cạnh của nó không cắt nhau ngoài ở đỉnh.



Hình 1.10. Đồ thị phẳng K

Một điều đáng lưu ý: Nếu đồ thị là phẳng thì ta luôn có thể vẽ nó trên mặt phẳng với các cạnh nối là các đoạn thẳng không cắt nhau ngoài ở đỉnh (ví dụ xem cách vẽ K trong hình 1.5.6.1).

1.2. Biểu diễn đồ thị trên máy tính [3]

1.2.1. Ma trận kề, ma trận trọng số

Để lưu trữ đồ thị và thực hiện các thuật toán khác nhau, ta cần phải biểu diễn đồ thị trên máy tính, đồng thời sử dụng những cấu trúc dữ liệu thích hợp để mô tả đồ thị.

Việc chọn cấu trúc dữ liệu nào để biểu diễn đồ thị có tác động rất lớn đến hiệu quả thuật toán. Vì vậy, lựa chọn cấu trúc dữ liệu thích hợp biểu diễn đồ thị sẽ phụ thuộc vào từng bài toán cụ thể.

Giả sử G = (V, E) là một đơn đồ thị có số đỉnh (ký hiệu |V|) là n, không mất tính tổng quát có thể coi các đỉnh được đánh số 1, 2, …, n.

Khi đó ta có thể biểu diễn đồ thị bằng một ma trận vuông A = [a[i, j]] cấp n, trong đó:

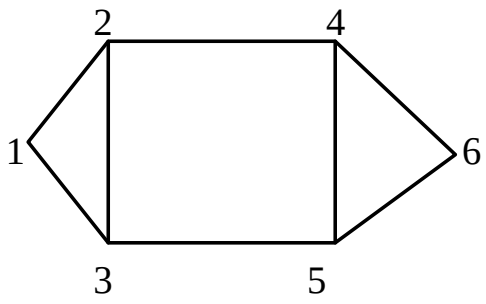
+ a[i, j] = 1 nếu (i, j) $\in$ E

+ a[i, j] = 0 nếu (i, j) $\notin$ E

Với $\forall$i, giá trị của a[i, i] có thể đặt tuỳ theo mục đích, thông thường nên đặt bằng 0.

Đối với đa đồ thị thì việc biểu diễn cũng tương tự trên, chỉ có điều nếu như (i, j) là cạnh thì không phải ta ghi số 1 vào vị trí a[i, j] mà là ghi số cạnh nối giữa đỉnh i và đỉnh j.

Ví dụ 1.11. Biểu diễn đồ thị trong hình 1.11 dưới đây bằng ma trận kề.



Hình 1.11. Đồ thị vô hướng không trọng số G

Bảng 1.1. Biểu diễn đồ thị vô hướng không trọng số bằng ma trận

	1	2	3	4	5	6
1	0	1	1	0	0	0
2	1	0	1	1	0	0
3	1	1	0	0	1	0
4	0	1	0	0	1	1
5	0	0	1	1	0	1
6	0	0	0	1	1	0

Các tính chất của ma trận kề

- Đối với đồ thị vô hướng G, thì *ma trận kề* tương ứng là *ma trận đối xứng* (a[i, j] = a[j, i]), điều này không đúng với đồ thị có hướng.

- Nếu G là đồ thị vô hướng và A là ma trận kề tương ứng thì trên ma trận A:

  + Tổng các số trên hàng i = Tổng các số trên cột i = Bậc của đỉnh i = deg(i)

- Nếu G là đồ thị có hướng và A là ma trận kề tương ứng thì trên ma trận A:

  + Tổng các số trên hàng i = Bán bậc ra của đỉnh i = deg+(i)

  + Tổng các số trên cột i = Bán bậc vào của đỉnh i = deg-(i)

- Trong trường hợp G là đơn đồ thị, ta có thể biểu diễn ma trận kề A tương ứng là các phần tử logic:

  + a[i, j] = TRUE nếu (i, j) $\in$ E và a[i, j] = FALSE nếu (i, j) $\notin$ E

Ưu điểm của ma trận kề:

+ Đơn giản, trực quan, dễ cài đặt trên máy tính

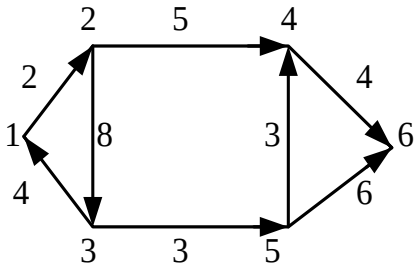
+ Để kiểm tra xem hai đỉnh (u, v) của đồ thị có kề nhau hay không, ta chỉ việc kiểm tra bằng một phép so sánh: a[u, v] $\neq$ 0.

Nhược điểm của ma trận kề:

+ Bất kể số cạnh của đồ thị là nhiều hay ít, ma trận kề luôn luôn đòi hỏi n2 ô nhớ để lưu các phần tử ma trận, điều đó gây lãng phí bộ nhớ dẫn tới việc không thể biểu diễn được đồ thị với số đỉnh lớn.

+ Với một đỉnh u bất kỳ của đồ thị, nhiều khi ta phải xét tất cả các đỉnh v khác kề với nó, hoặc xét tất cả các cạnh liên thuộc với nó. Trên ma trận kề việc đó được thực hiện bằng cách xét tất cả các đỉnh v và kiểm tra điều kiện $a[u, v] \neq 0$. Như vậy, ngay cả khi đỉnh u là đỉnh cô lập (không kề với đỉnh nào) hoặc đỉnh treo (chỉ kề với 1 đỉnh) ta cũng buộc phải xét tất cả các đỉnh và kiểm tra điều kiện trên dẫn tới lãng phí thời gian.

Ví dụ 1.12. Tìm ma trận kề của đồ thị có hướng, c1o trọng số như hình 1.12

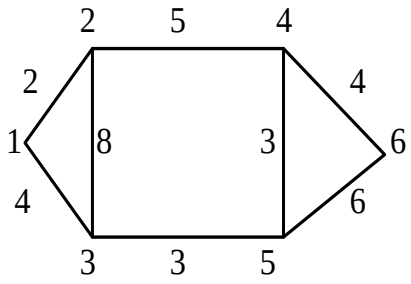


Hình 1.12. Đồ thị G có hướng, có trọng số

Bảng 1.2. Biểu diễn đồ thị có hướng có trọng số bằng ma trận

1	2	3	4	5	6
0	2	0	0	0	0
0	0	8	5	0	0
4	0	0	5	1	0
0	0	0	0	0	4
0	0	0	3	0	6
0	0	0	0	0	0

Ví dụ 1.13. Tìm ma trận kề của đồ thị có trọng số như hình 1.13



Hình 1.13. Đồ thị vô hướng G có trọng số

Bảng 1.3. Biểu diễn đồ thị vô hướng có trọng số bằng ma trận

	1	2	3	4	5	6
1	0	2	4	0	0	0
2	2	0	8	5	0	0
3	4	8	0	5	1	0
4	0	5	0	0	3	4
5	0	0	3	3	0	6
6	0	0	0	4	6	0

1.2.2. Danh sách cạnh (cung)

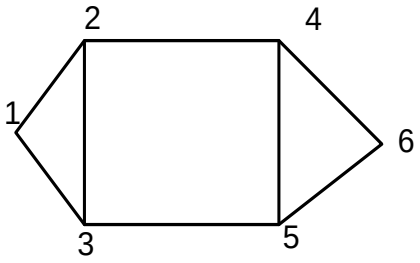
Trong trường hợp đồ thị thưa (đồ thị có số cạnh $m \leq 6n$), người ta thường biểu diễn đồ thị dưới dạng *danh sách cạnh*.

Trong phép biểu diễn này, chúng ta sẽ lưu trữ danh sách tất cả các cạnh (cung) của đồ thị vô hướng (có hướng). Mỗi cạnh (cung) e (x, y) được tương ứng với hai biến dau[e], cuoi[e].

Như vậy, để lưu trữ đồ thị, ta cần 2m đơn vị bộ nhớ, đây là **ưu điểm** của việc lưu trữ bằng danh sách kế.

Nhược điểm lớn nhất của phương pháp này là để nhận biết những cạnh nào kề với cạnh nào chúng ta cần m phép so sánh trong khi duyệt qua tất cả m cạnh (cung) của đồ thị. Nếu là đồ thị có trọng số, ta cần thêm m đơn vị bộ nhớ để lưu trữ trọng số của các cạnh.

Ví dụ 1.14. Biểu diễn đồ thị bằng danh sách cạnh như hình 1.14

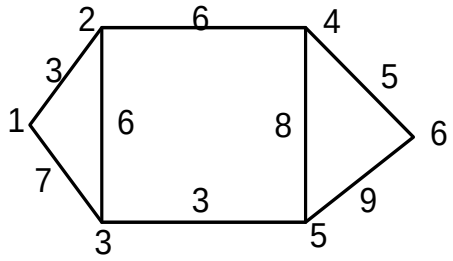


Hình 1.14. Đồ thị vô hướng không trọng số G

Bảng 1.4. Biểu diễn đồ thị không trọng số bằng danh sách cạnh cung

Dau	Cuoi
1	2
1	3
2	3
2	4
3	5
4	5
4	6
5	6

Ví dụ 1.15. Biểu diễn đồ thị có trọng số bằng danh sách cạnh như hình 1.15



Hình 1.15. Đồ thị vô hướng có trọng số G

Bảng 1.5. Biểu diễn đồ thị có trọng số bằng danh sách cạnh có trọng số

Dau	Cuoi	Trongso
1	2	3
1	3	7

2	3	6
2	4	6
3	5	3
4	5	8
4	6	5
5	6	9

1.2.3. Danh sách kề

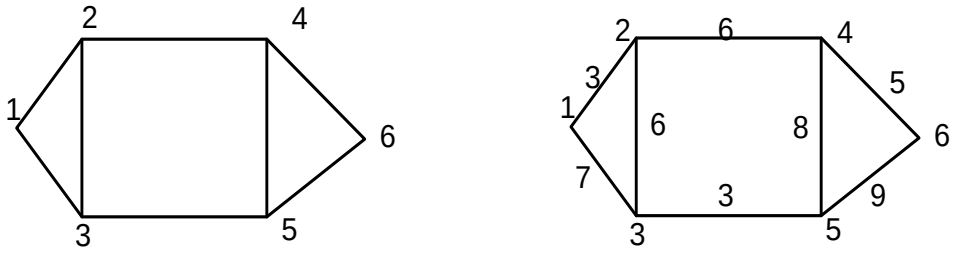
Trong rất nhiều ứng dụng, cách biểu diễn đồ thị dưới dạng *danh sách kề* thường được sử dụng. Trong biểu diễn này, với mỗi đỉnh v của đồ thị chúng ta lưu trữ danh sách các đỉnh kề với nó mà ta ký hiệu là Ke(v), nghĩa là

$$Ke(v) = \{ u \in V: (u, v) \in E\},$$

Với cách biểu diễn này, mỗi đỉnh i của đồ thị, ta làm tương ứng với một danh sách tất cả các đỉnh kề với nó và được ký hiệu là List(i).

Để biểu diễn List(i), ta có thể dùng các kiểu dữ liệu kiểu tập hợp, mảng hoặc danh sách liên kết.

Ví dụ 1.16. Biểu diễn danh sách kề của đồ thị vô hướng không trọng số và có trọng số như hình 1.16



Hình 1.16. Đồ thị vô hướng không trọng số và có trọng số

Bảng 1.6. Biểu diễn đồ thị bằng danh sách kề biểu diễn đồ thị

Đỉnh	List(i)		
2	1	3	4
3	1	2	5
4	2	5	6
5	3	4	6
6	4	5	

Đỉnh	List(i)		
2	4	5	
3	4		
5	1		

1.3. Chu trình Euler, Đường đi Euler và Đồ thị Euler [3]

1.3.1. Khái niệm Đường đi, Chu trình, tính Liên thông trên Đồ thị

Đường đi có độ dài k từ đỉnh u đến đỉnh v trên đồ thị vô hướng G=<V,E> là dãy:

$$x_0, x_1, \ldots, x_{n-1}, x_k$$

Trong đó:

+ k là số nguyên dương,

+ $x_0 = u$, $x_k = v$, $(x_i, x_{i+1}) \in E$,

+ $i = 0, 1, 2, \ldots, k-1$.

Đường đi như trên còn có thể biểu diễn thành dãy các cạnh:

$$(x_0, x_1), (x_1,x_2), \ldots, (x_{k-1}, x_k).$$

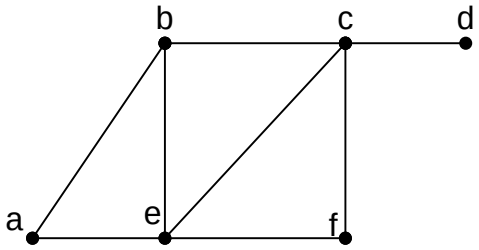
Đỉnh u là *đỉnh đầu*, đỉnh v là *đỉnh cuối* của đường đi.

Chu trình: là đường đi có đỉnh đầu trùng với đỉnh cuối (u = v).

Đường đi đơn: là đường đi mà không có cạnh nào lặp lại.

Chu trình đơn: là chu trình mà không có cạnh nào lặp lại.

Ví dụ 1.17. Tìm các đường đi, chu trình trong đồ thị vô hướng như hình 1.17



Hình 1.17. Đường đi trên đồ thị

+ a, b, c, d là đường đi đơn độ dài 3.

+ a, b, f, không là đường đi vì (b, f) không phải là cạnh của đồ thị.

+ Dãy a, b, c, f, e, a là chu trình độ dài 5.

+ Đường đi a, b, c, f, e, b, c có độ dài 6 không phải là đường đi đơn vì cạnh (b, c) có mặt hai lần.

Khái niệm đường đi và chu trình trên đồ thị có hướng được định nghĩa hoàn toàn tương tự, chỉ có điều khác biệt duy nhất là ta phải chú ý tới các cung của đồ thị

Đồ thị liên thông: Nếu giữa hai điểm bất kỳ của một đồ thị đều có thể thiết lập một đường đi từ đỉnh này đến đỉnh kia thì đồ thị được coi là đồ thị liên thông; nếu không, đồ thị được coi là không liên thông.

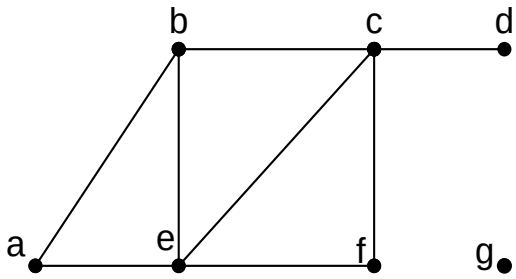
Một đồ thị được coi là hoàn toàn không liên thông nếu không có đường đi giữa hai đỉnh bất kỳ trong đồ thị. Đây chỉ là một cái tên khác để miêu tả một đồ thị rỗng hoặc một tập độc lập.

Một đồ thị có hướng được coi là *liên thông mạnh* nếu từ mỗi đỉnh đều đến được mọi đỉnh khác.

Ngược lại, đồ thị có hướng được coi là *liên thông yếu* nếu đồ thị vô hướng nền tảng của nó là đồ thị liên thông.

Ví dụ 1.18. Kiểm tra tính liên thông của đồ thị hình 1.18

Đồ thị này không liên thông vì có đỉnh g là đỉnh cô lập nên không có đường đi từ đình bất kỳ đến đỉnh g.



Hình 1.18. Đồ thị không liên thông

Chu trình sơ cấp: là chu trình không đi qua một đỉnh quá 1 lần (hay đi qua mỗi đỉnh đúng 1 lần).

1.3.2. Khái niệm Chu trình Euler, Đường đi Euler và Đồ thị Euler

Cho đồ thị vô hướng G=(V,E).

Chu trình Euler là chu trình đi qua mọi cạnh và mọi đỉnh của đồ thị, mỗi cạnh không đi quá 1 lần.

Đường đi Euler là đường đi qua mọi cạnh và mọi đỉnh của đồ thị, mỗi cạnh không đi quá 1 lần.

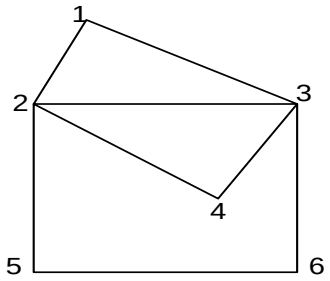
Cho đồ thị có hướng G=(V,E).

Chu trình có hướng Euler là chu trình có hướng đi qua mọi cung và mọi đỉnh đồ thị, mỗi cung không đi quá 1 lần.

Đường đi có hướng Euler là đường đi có hướng đi qua mọi cung và mọi đỉnh đồ thị, mỗi cung không đi quá 1 lần.

Đồ thị chứa chu trình Euler gọi là *Đồ thị Euler*.

Ví dụ 1.19. Đồ thị hình 1.19 có chu trình Euler (1, 2, 4, 3, 6, 5, 2, 3, 1)



Hình 1.19. Đồ thị có chu trình Euler

Điều kiện cần và đủ:

Định lý 1 (Định lý Euler): Đồ thị G có chu trình Euler khi và chỉ khi G liên thông và mọi đỉnh có bậc chẵn khác 0.

Định lý 2 Cho đồ thị G có *k* đỉnh bậc lẻ. Khi đó số đường đi tối thiểu phủ G là *k/2*. Đồ thị luôn có số đỉnh bậc lẻ là chẵn, *k=2n*.

1.3.3. Thuật toán Fleury tìm chu trình Euler

Đầu vào. Đồ thị G ≠ ∅, không có đỉnh cô lập.

Đầu ra. Chu trình Euler C của G, hoặc kết luận G không có chu trình Euler.

Phương pháp.

(1) Chọn đỉnh xuất phát bất kỳ v_0. Đặt $v_1 := v_0$, C := (v_0). H := G.

(2) Nếu H = ∅, thì kết luận *C là chu trình Euler*, kết thúc.

Ngược lại sang bước (3).

(3) Chọn cạnh đi tiếp:

- Trường hợp đỉnh v_1 là đỉnh treo: Tồn tại duy nhất đỉnh v_2 kề v_1.

Chọn cạnh (v_1, v_2). Sang bước (4).

- Trường hợp đỉnh v_1 không là đỉnh treo:

Nếu mọi cạnh liên thuộc v_1 là cầu, thì *không có chu trình Euler*, kết thúc.

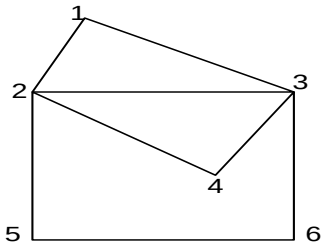
- Ngược lại, chọn cạnh (v_1, v_2) bất kỳ không phải là cầu trong H. Thêm vào đường đi C đỉnh v_2. Sang bước (4).

(4) Xoá cạnh vừa đi qua, và xoá đỉnh cô lập:

Loại khỏi H cạnh (v_1, v_2). Nếu H có đỉnh cô lập, thì loại chúng khỏi H.

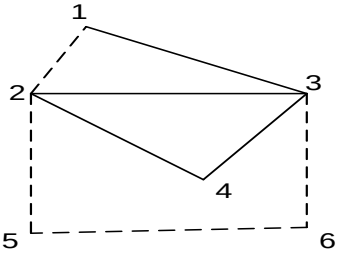
Đặt v_1 := v_2. Sang bước (2).

Ví dụ 1.20. Cho G là đồ thị hình sau



Hình 1.20. Đồ thị liên thông và có các đỉnh bậc chẵn.

Nếu xuất phát từ đỉnh 1, có hai cách đi tiếp: hoặc sang 2 hoặc sang 3, giả sử ta sẽ sang 2 và xoá cạnh (1, 2) vừa đi qua. Từ 2 có ba cách để đi, là sang 3 sang 4 hoặc sang 5, giả sử ta sẽ sang 5 và xoá cạnh (2, 5) vừa đi qua. Từ đỉnh 5, chỉ có 1 cách đi là sang đỉnh 6, nên cho dù (5, 6) là cầu ta cũng phải đi sau đó xoá luôn cạnh (5, 6). Từ đỉnh 6, chỉ có 1 cách đi là sang đỉnh 3, nên cho dù (6, 3) là cầu ta cũng phải đi sau đó xoá luôn cạnh (6, 3). Đến đây, các cạnh còn lại của đồ thị có thể vẽ như Hình bằng nét liền, các cạnh đã bị xoá được vẽ bằng nét đứt.



Hình 1.21. Duyệt đồ thị theo thuật toán Fluery

Bây giờ đang đứng ở đỉnh 6 thì ta có 3 cách đi tiếp: sang 1, sang 2 hoặc sang 4. Vì (3, 1) là cầu nên ta sẽ không đi theo cạnh (3, 1) mà sẽ đi (3, 4) hoặc (3, 2). Nếu đi theo (3, 4) và cứ tiếp tục đi như vậy, ta sẽ được chu trình Euler là <1, 2, 5, 6, 3, 4, 2, 3, 1>. Còn đi theo (3, 2) sẽ tìm được chu trình Euler là: <1, 2, 5, 6, 3, 2, 4, 3, 1>.

1.4. Một số thuật toán trên Đồ thị

1.4.1. Thuật toán Floyed tìm đường đi ngắn nhất giữa mọi cặp đỉnh trên đồ thị

Thuật giải tìm độ dài đường đi ngắn nhất giữa mọi cặp đỉnh trong đồ thị có hướng liên thông có trọng số (không bắt buộc ≥ 0).

Đầu vào. Đồ thị liên thông G=(V,E), V= {1, 2,..., n}, có trọng số w(i,j) với mọi cung (i,j).

Đầu ra. Ma trận D=[d(i,j)], trong đó d(i,j) là chiều dài đường đi ngắn nhất từ i đến j với mọi cặp (i,j).

Phương pháp:

(1) Bước khởi tạo: Ký hiệu D_0 là ma trận xuất phát
$D_0 = [d_0(i,j)]$

Trong đó: $d_0(i,j) = w(i,j)$ nếu tồn tại cung (i,j) và $d_0(i,j) = +\infty$ nếu không tồn tại cung (i,j) (đặc biệt nếu không có khuyên tại i thì $d_0(i,i) = +\infty$).

Gán k:=0.

(2) Kiểm tra kết thúc: Nếu k = n, kết thúc.
$D = D_n$ là ma trận độ dài đường đi ngắn nhất.
Ngược lại tăng k lên 1 đơn vị (k:=k+1) và sang (3).

(3) Tính ma trận D_k theo D_{k-1}:
Với mọi cặp (i,j), i=1..n, j=1..n thực hiện:

Nếu $d_{k-1}(i,j) > d_{k-1}(i,k) + d_{k-1}(k,j)$ thì đặt

$d_k(i,j) := d_{k-1}(i,k) + d_{k-1}(k,j)$

ngược lại đặt

$d_k(i,j) := d_{k-1}(i,j)$

Quay lại bước (2).

- ***Định lý 1*:** Thuật toán Floyd là đúng.

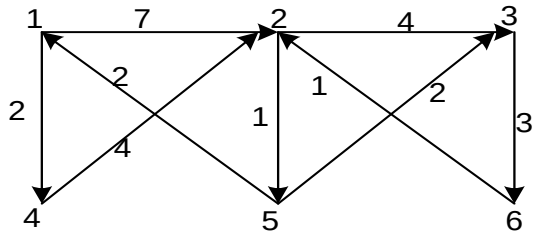
- ***Hệ quả:***

(i) Nếu ma trận kết quả của thuật toán Floyd có phần tử hữu hạn trên đường chéo i = i thì đồ thị chứa chu trình.

(ii) Nếu ma trận kết quả chứa phần tử +∞ ngoài đường chéo i = i thì đồ thị không liên thông mạnh.

Ghi chú: Từ hệ quả trên ta có thể sử dụng thuật toán Floyd, với w(i, j) = 1 nếu tồn tại cung (i, j) và w(i, j)= +∞ nếu không tồn tại cung (i, j), để xác định xem đồ thị có chu trình hay có liên thông hay không.

Ví dụ 1.21. Xét đồ thị có hướng hình 1.22



Hình 1.22. Đồ thị Floyd

Áp dụng thuật toán Floyd ta có:

Ma trận khoảng cách xuất phát D_0 là (các ô trống là ∞)

Bảng 1.7. Biểu diễn đồ thị cần tìm bằng thuật toán Floyd bằng ma trận

D_0 =

Đỉnh	**1**	**2**	**3**	**4**	**5**	**6**
1		7		2		
2			4		1	
3						3
4		4				
5	2		2			
6		1				

- Từ ma trận D_0, theo thuật toán, ta xây dựng các ma trận tiếp theo như sau (các ô gạch dưới có giá trị thay đổi)

Bảng 1.8. Các bước tìm đường đi ngắn nhất giữa mọi cặp đỉnh bằng Floyd

Đỉnh	**1**	**2**	**3**	**4**	**5**	**6**
1		7		2		
2			4		1	

D_1 =

3						3
4		4				
5	2	**9**	2	**4**		
6		1				

D_2 =

Đỉnh	**1**	**2**	**3**	**4**	**5**	**6**
1		7	**11**	2	**8**	
2			4		1	
3						3
4		4	**8**		5	
5	2	9	2	4	**10**	
6		1	**5**		**2**	

D_3 =

Đỉnh	**1**	**2**	**3**	**4**	**5**	**6**
1		7	11	2	8	**14**
2			4		1	**7**
3						3
4		4	8		5	**11**
5	2	9	2	4	10	**5**
6		1	5		2	**8**

D_4 =

Đỉnh	**1**	**2**	**3**	**4**	**5**	**6**
1		**6**	**10**	2	**7**	**13**
2			4		1	7
3						3
4		4	8		5	11
5	2	**8**	2	4	**9**	5
6		1	5		2	8

D_5 =

Đỉnh	**1**	**2**	**3**	**4**	**5**	**6**
1	**9**	6	**9**	2	7	**12**
2	**3**	**9**	**3**	**5**	1	**6**
3						3
4	**7**	4	**7**	**9**	5	**10**
5	2	8	2	4	9	5
6	**4**	1	**4**	**6**	2	**7**

D_6 =

Đỉnh	1	2	3	4	5	6
1	9	6	9	2	7	12
2	3	**7**	3	5	1	6
3	**7**	**4**	**7**	**9**	**5**	3
4	7	4	7	9	5	10
5	2	**6**	2	4	**7**	5
6	4	1	4	6	2	7

- Cuối cùng, D_6 là ma trận khoảng cách ngắn nhất giữa các đỉnh. Theo hệ quả ta thấy đồ thị liên thông mạnh và chứa chu trình

1.4.2. Giải thuật Tham lam

Khái niệm:

Giải thuật Tham lam là một thuật toán giải quyết bài toán theo kiểu tìm kiếm lựa chọn tối ưu cục bộ ở mỗi bước đi với hy vọng tìm được tối ưu toàn cục**.**

Tại mỗi bước lựa chọn, thuật toán sẽ "chọn kết quả tốt nhất" được xác định bởi hàm "chọn giá trị tốt nhất" (có thể là giá trị Max, Min, hoặc theo 1 ý nghĩa nào đó tùy vào bài toán cụ thể) và nếu có thể "kết quả chọn" sẽ trở thành nghiệm của bài toán thì "chọn" luôn; nếu không nó sẽ bỏ đi và không xem xét lại.

Đặc trưng:

Có thể lựa chọn giải pháp nào được cho là tốt nhất ở thời điểm hiện tại và sau đó giải bài toán con nảy sinh từ việc thực hiện lựa chọn vừa rồi. Lựa chọn của thuật toán tham lam có thể phụ thuộc vào các lựa chọn trước đó. Nhưng nó không thể phụ thuộc vào một lựa chọn nào trong tương lai hay phụ thuộc vào lời giải của các bài toán con. Thuật toán tiến triển theo kiểu thực hiện các chọn lựa theo một vòng lặp, cùng lúc đó thu nhỏ bài toán đã cho về một bài toán con nhỏ hơn.

Đặc trưng tham lam của phương pháp thể hiện bởi: trong mỗi bước việc xử lý sẽ tuân theo một sự lựa chọn trước, không kể đến tình trạng không tốt có thể xảy ra khi thực hiện lựa chọn lúc đầu.

Phương pháp tham lam thường được áp dụng cho bài toán tối ưu các bước tính toán về độ phức tạp và chấp nhập kết quả tìm được theo dạng chấp nhận được tùy vào yêu cầu của bài toán.

Mô hình giải thuật:

Đầu vào: Ma trận A cần tìm

Đầu ra: Bộ giá trị x của tập S cần tìm

Phương pháp:

Chọn S từ tập A.
Tính chất tham lam của thuật toán định hướng bởi hàm Chọn
Khởi tạo: S = ∅
Trong khi A ≠ ∅
Chọn phần tử tốt nhất của A gán vào x : x = Chọn (A)
Cập nhật các đối tượng để chọn: A = A –{x}
Nếu S ∪ {x} thỏa mãn yêu cầu của bài toán thì

Cập nhật lời giải: S = S ∪ {x}

Ví dụ 1.21. Cho ma trận của đồ thị 2 phía gồm 4 phần tử mỗi phía như hình bên dưới. Tìm bộ ghép có trọng số nhỏ nhất.

Bảng 1.9. Ma trận biểu diễn đồ thị cần tìm cặp ghép bằng giải thuật Tham lam.

	1	**2**	**3**	**4**
A	2	5	1	6
B	8	7	6	4
C	6	9	3	5
D	5	3	2	7

- Xây dựng hàm tìm Bộ ghép như sau:

```
Procedure Boghep_Thamlam
 Begin
 S :=Ø  ; //S là bộ ghép cần tìm
 A : Ma trận đầu vào
   While(A≠0) //A là ma trận đầu vào
   Begin
```

```
    For i in 0..3;
         For j in 0..3;
         If A[i,i] là min;
              Begin
                   Chọn(A(i,j)); //Chọn bộ ghép
                   Xóa(A[n,j]); //xóa cột dòng j
                   Xóa(A[i,m]); //xóa dòng i
              End If;
  End while;
  XuatBoghep();
End;
```

- Duyệt ma trận: chọn giá trị nhỏ nhất và xóa đi dòng cột liên quan, giá trị gạch chân là được chọn tương ứng từng bước như sau:

Bảng 1.10. Các bước tìm bộ ghép có trọng số min bằng giải thuật Tham lam

Bước 1

	1	**2**	**3**	**4**
A			**1**	
B	8	7		4
C	6	9		5
D	5	3		7

Bước 2

	1	**2**	**3**	**4**
A			**1**	
B	8			4
C	6			5
D		**3**		

Bước 3

	1	**2**	**3**	**4**
A			**1**	
B				**4**
C	6			
D		**3**		

Bước 4

	1	**2**	**3**	**4**
A			**1**	
B				**4**
C	**6**			
D		**3**		

Ta được bộ ghép {(A,3),(D,2),(B,4),(C,1)} có trọng số bằng 1+3+4+6 = 14

1.4.3. Tìm bộ ghép trên đồ thị

1.4.3.1. Giới thiệu chung

Định nghĩa: Cho đồ thị vô hướng G = (V, E) trong đó V là tập đỉnh, E là tập cạnh, một cặp ghép M trên G là một tập con của E thoả mãn không có hai cạnh nào chung đỉnh đầu mút.

Có hai bài toán quan trọng trong vấn đề về cặp ghép :

Tìm bộ ghép M có |M| lớn nhất. Bài toán này gọi là bài toán tìm cặp ghép cực đại.

Với mỗi cạnh (u, v) $\in$ E ta cho tương ứng một số thực W(u, v) gọi là trọng số cạnh (u, v). Hãy tìm cặp ghép cực đại với tổng trọng số M nhỏ nhất.

$$\sum W(u, v) \rightarrow \min$$

$$(u, v) \in M$$

1.4.3.2. Bài toán tìm cặp ghép cực đại với tổng trọng số nhỏ nhất

Bài toán cặp ghép cực đại với tông trọng số nhỏ nhất, ta luôn có thể coi đồ thị hai phía là đầy đủ, Thật vây, ta chỉ cần thay cạnh không có bằng một cạnh có trọng số vô cùng lớn so với các trọng số còn lại. Do đó, từ lúc này ta chỉ nói đến đồ thị đầy đủ.

Bài toán: Cho G = (X, Y) trong đó |X| = |Y|, với mỗi x $\in$ X, y $\in$ Y ta cho tương ứng với một số nguyên w(x, y) gọi là trọng số cạnh. Tìm cặp ghép đầy đủ |M| = |X| sao cho $\sum w(x, y) \rightarrow \min$

$$(x, y) \in M$$

Một trong những mô hình thực tế của bài toán là bài toán phân công: Có N thợ và N việc, thợ thứ i làm công việc j mất chi phí w(i, j). Hãy tìm cách phân công mỗi thợ làm một việc, sao cho tổng chi phí thực hiện là nhỏ nhất.

1.4.3.3. Bài toán tìm bộ ghép cực đại trọng số nhỏ nhất trên trên đồ thị đầy đủ

Đầu vào: Ma trận đồ thị đầy đủ có trọng số

Đầu ra: Ma trận cặp ghép có tổng trọng số nhỏ nhất

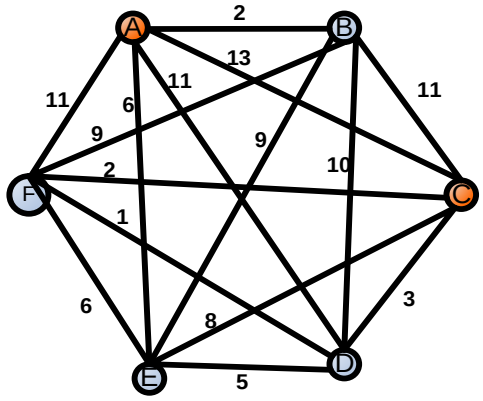
Phương pháp: Lần lượt ghép tất cả đỉnh của A với mỗi đỉnh B, đổi đỉnh sẽ ghép chính xác với một đỉnh khác và không có đỉnh nào được ghép với đỉnh khác nhiều hơn hai lần. Cần tìm tổng cặp ghép có trọng số là nhỏ nhất.

Hàm tìm bộ ghép tối ưu có trọng số nhỏ nhất trên đồ thị đầy đủ *(theo Dominic Battré)*

```
Procedure FindMinMatch
A: đồ thị đầu vào
N: số phần tử ma trận đầu vào
ketQua: mảng các cặp ghép
danhDau: mảng lưu đỉnh đã duyệt, khởi tạo giá trị 0
gtMin: lưu giá trị tối ưu
gtCur: giá trị hiện tại, khởi tạo bằng 0
Level: lưu mức đỉnh đã ghép cặp , khởi tạo bằng 0
LastAss: lưu đỉnh đang xét, khởi tạo =0
Begin
For nextfree in lastAss+1 … N
  If danhDau[nextfree] = 0
    Break
For i in nextfree+1 … N
 begin
  If danhDau[i] = 0
   Begin
    Ghép căp nextfree với i
    Cập nhật gtrị i từ mảng danhdau bằng gtrị nextfree
     Cộng giá trị đường di vào biến gtCur
     If giá trị gtCur < giá trị gtMin
       Begin
         If level+1 < N/2
            Gọi lại đệ quy FindMinMatch với level+1;
         Else
```

```
            Begin
              Cập nhật gtMin = gtCur
              For t in 0 … N
                Lưu cặp ghép hiện tại vào mảng KetQua
            End Else
          End If
        Else
         loại bỏ cặp [i][nextfree] từ mảng MinMatch
         Loại bỏ các đánh dấu
            End If
          End For
End
```

Ví dụ 1.22. Cho đồ thị đầy đủ vô hướng có trọng số như hình bên dưới. Tìm bộ ghép cực đại có tổng trọng số nhỏ nhất



Hình 1.23. Đồ thị đầy đủ có trọng số

Đồ thị trên được biểu diễn bằng ma trận vuông có trọng số

Bảng 1.11. Ma trận biểu diễn đồ thị đầy đủ vô hướng có trọng số

	A	B	C	D	E	F
A	0	2	13	11	6	11
B	2	0	11	10	9	9
C	13	11	0	3	8	2
D	11	10	5	0	5	1
E	6	9	8	5	0	6
F	11	9	2	1	6	0

Thuật toán chạy trình tự các bước như sau:

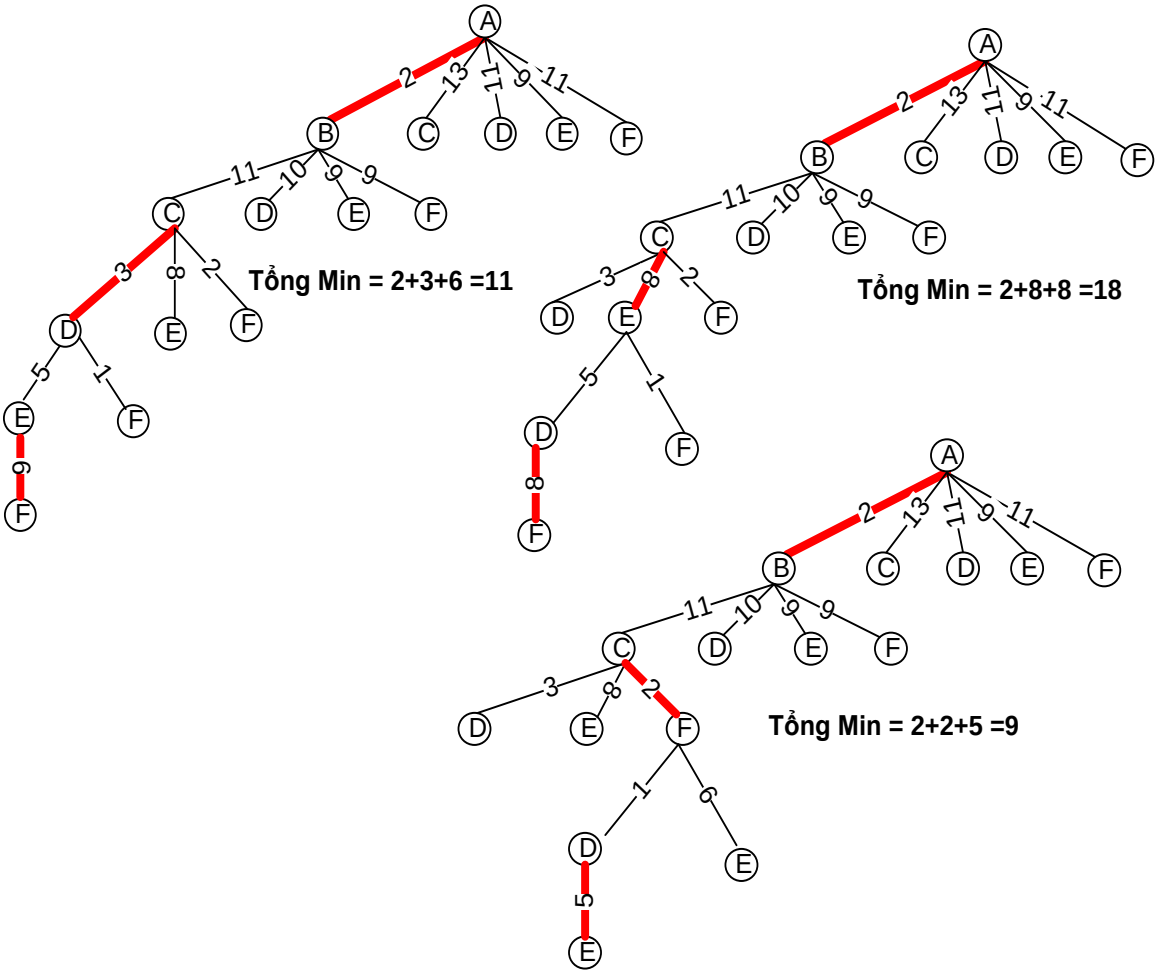
- Khi chọn A-B là cặp ghép đầu tiên thì tai có 3 cách chọn tiếp theo:

  + Nếu chọn C-D thì bước tiếp sau chỉ còn cách chọn duy nhất là E-F, có tổng trọng số là 2 + 3 + 6 = 11

  + Nếu chọn C-E thì bước tiếp sau chỉ còn cách chọn duy nhất là D-F, có tổng trọng số là 2 + 8 + 8 = 18

  + Nếu chọn C-F thì bước tiếp sau chỉ còn cách chọn duy nhất là D-E, có tổng trọng số là 2 + 2 + 5 = 9

  + Ta thấy kết quả nhỏ nhất là 9, nên tạm thời lưu kết quả gtMin = 9.



Hình 1.24. Trường hợp chọn cặp ghép đầu tiên là A-B

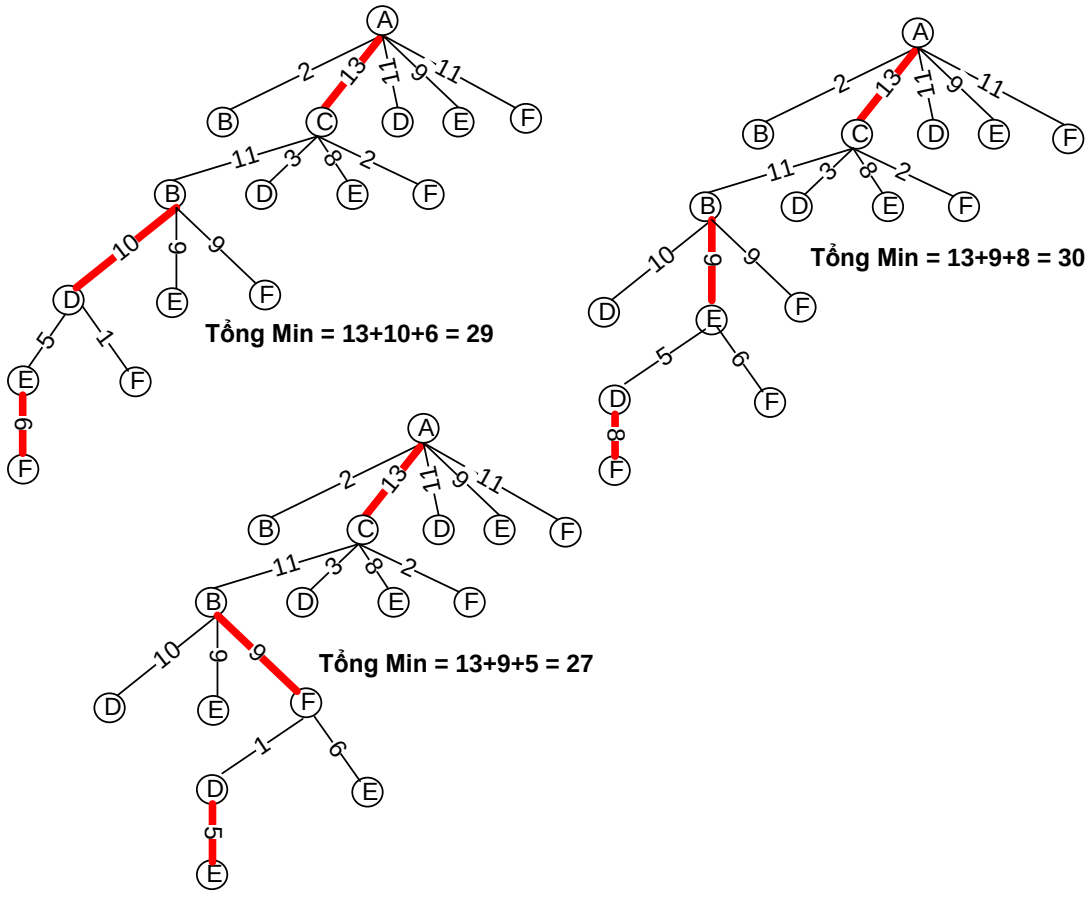
- Khi chọn A-C là cặp ghép đầu tiên thì tai có 3 cách chọn tiếp theo:

+ Nếu chọn B-D thì bước tiếp sau chỉ còn cách chọn duy nhất là A-F, có tổng trọng số là 13 + 10 + 6 = 29

+ Nếu chọn B-E thì bước tiếp sau chỉ còn cách chọn duy nhất là D-F, có tổng trọng số là 13 + 8 + 8 = 30

+ Nếu chọn B-F thì bước tiếp sau chỉ còn cách chọn duy nhất là D-E, có tổng trọng số là 13 + 9 + 5 = 27

+ Giá trị nhỏ nhất là 27, lớn hơn giá trị gtMin hiện tại nên không được chọn.



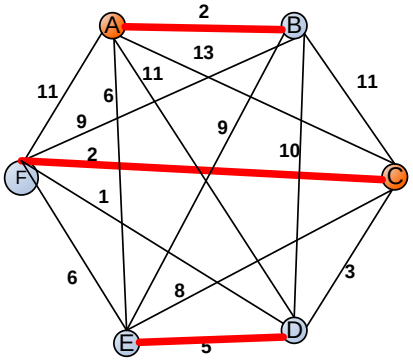
Hình 1.25. Trường hợp chọn cặp ghép đầu tiên là A-C

Tương tự cho các bước chọn gặp ghép khác nhau từng đôi một tiếp theo như A-D, A-E, AF và thực hiện các bước tiếp theo cho đến hết tất cả các trường hợp có thể.

Sau khi ghép cặp và so sánh tìm gia trị có tổng trọng số nhỏ nhất, kết quả tìm được bộ ghép gồm 03 cặp ghép {(2,3), (7,11), (8,9)} có tổng trọng số bằng 9

Bảng 1.12. Ma trận kết quả bộ ghép cực đại có trọng số cực tiểu

	A	B	C	D	E	F
A	0	**2**	13	11	6	11
B	**2**	0	11	10	9	9
C	13	11	0	3	8	**2**
D	11	10	5	0	**5**	1
E	6	9	8	**5**	0	6
F	11	9	**2**	1	6	0



Hình 1.26. Đồ thị bộ ghép tìm được

1.4.3.4. Bài toán Người phát thư Trung Hoa

Phát biểu bài toán:

Một nhân viên đi từ Sở Bưu điện, qua một số đường phố để phát thư, rồi quay về Sở. Người ấy phải đi qua các đường theo trình tự nào để đường đi là ngắn nhất?

Cho đồ thị liên thông G. Một chu trình qua mọi cạnh của G gọi là một hành trình trong G. Trong các hành trình đó, hãy tìm hành trình ngắn nhất, tức là qua ít cạnh nhất.

Rõ ràng rằng nếu G là đồ thị Euler (mọi đỉnh đều có bậc chẵn) thì chu trình Euler trong G (qua mỗi cạnh của G đúng một lần) là hành trình ngắn nhất cần tìm.

Ta xét trường hợp G có một số đỉnh bậc lẻ. Khi đó, mọi hành trình trong G phải đi qua ít nhất hai lần một số cạnh nào đó. Dễ thấy rằng một hành trình T qua

một cạnh (u,v) nào đó quá hai lần thì không phải là hành trình ngắn nhất trong G. Vì vậy, ta chỉ cần xét hành trình sẽ đi qua hai lần một số cạnh nào đó của G.

Ta quy ước xem mỗi hành trình T trong G là một hành trình trong đồ thị Euler G_T có được từ G bằng cách vẽ thêm một cạnh song song đối với những cạnh mà T đi qua hai lần. Bài toán đặt ra được đưa về bài toán sau:

Trong các đồ thị Euler G_T, tìm đồ thị có số cạnh ít nhất (khi đó chu trình Euler trong đồ thị này là hành trình ngắn nhất).

Định lý (Gooodman và Hedetniemi, 1973)

Nếu G là một đồ thị liên thông có q cạnh thì hành trình ngắn nhất trong G có chiều dài q + m(G), trong đó:

m(G) là số cạnh mà hành trình đi qua 2 lần.

Gọi $V_0(G)$ là tập hợp các đỉnh bậc lẻ (2k đỉnh) của G. Ta phân 2k phần tử của G thành k cặp, mỗi tập hợp k cặp gọi là một phân hoạch cặp của $V_0(G)$.

Với mỗi cặp đỉnh *u, v* trong một phân hoạch P_i của $V_0(G)$, ta xét khoảng cách giữa 2 đỉnh đó (chính bằng độ dài đường đi ngắn nhất nhận *u,v* làm 2 đầu mút), ký hiệu là *d(u,v)*. Tính khoảng cách của k cặp đỉnh, rồi cộng lại ta được tổng $d(P_i)$

+ Số *m(G)* chính là số nhỏ nhất trong các tổng $d(P_i)$

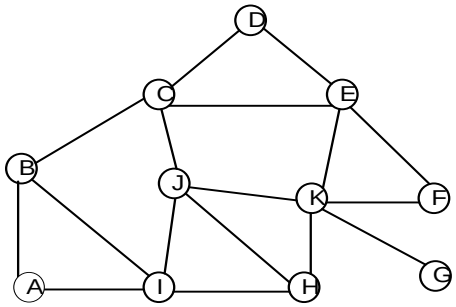
+ $m(G) = \min d(P_i)$.

Giải bài toán Người phát Trung Hoa thư theo đồ thị sau:

Cho đồ thị *G* gồm 1 đỉnh: A, B, C, D, E, F, G, H, I, J, K với bậc tương ứng là 2,3,4,2,4,2,1,3,4,4,5 như trong hình vẽ.

Ta sử dụng định lý Goodman-Hedetniemi để tìm hành trình ngắn nhất trong *G*.

Số cạnh của *G*: q=17.



Hình 1.27. Đồ thị không thoả Euler

Tập hợp các đỉnh bậc lẻ $V_O(G)=\{B, G, H, K\}$ và tập hợp các phân hoạch cặp là $P=\{P_1, P_2, P_3\}$, trong đó:

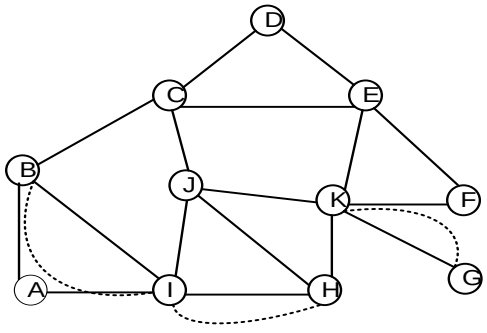
$P_1 = \{(B, G), (H, K)\} \rightarrow d(P_1) = d(B, G)+d(H, K) = 4+1 = 5,$

$P_2 = \{(B, H), (G, K)\} \rightarrow d(P_2) = d(B, H)+d(G, K) = 2+1 = 3,$

$P_3 = \{(B, K), (G, H)\} \rightarrow d(P_3) = d(B, K)+d(G, H) = 3+2 = 5.$

$m(G) = \min(d(P_1), d(P_2), d(P_3)) = 3.$

Do đó G_T có được từ G bằng cách thêm vào 3 cạnh: (B, I), (I, H), (G, K) và G_T là đồ thị Euler. Vậy hành trình ngắn nhất cần tìm là đi theo chu trình Euler trong G_T: A, B, C, D, E, F, K, G, K, E, C, J, K, H, J, I, H, I, B, I, A.



Hình 1.28. Đồ thị thoả Euler sau khi thêm cạnh

Chương 2. ỨNG DỤNG ĐỒ THỊ EULER TỐI ƯU HÓA BÀI TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT

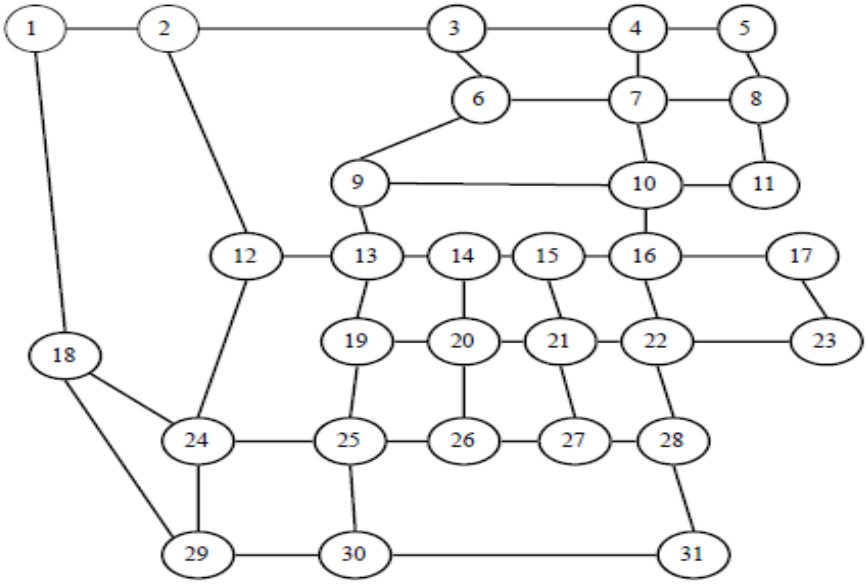
2.1. Phân tích bài toán "Thanh tra giao thông" của tác giả Nguyễn Tam Hùng

(Theo Luận văn Thạc sỹ được thực hiện năm 2014 tại trường Đại học Thái Nguyên, tên đề tài: "*Các thuật toán về đường đi và chu trình euler và ứng dụng*" của tác giả Nguyễn Tam Hùng.

2.1.1. Phát biểu bài toán

Một đội tuần tra giao thông xuất phát từ trụ sở chính, họ phải tuần tra trên tất cả các con đường của một khu vực đội quản lý rồi quay trở về trụ sở.

Bài toán được đưa về dạng tìm chu trình Euler với đồ thị G không trọng số là khu vực quản lý, với các cạnh là những con đường, các đỉnh là các nút giao thông, đỉnh xuất phát là trụ sở chính. Trong các hành trình đó, hãy tìm hành trình ngắn nhất, tức là qua ít cạnh nhất.



Hình 2.1. Đồ thị hành trình thanh tra giao thông

2.1.2. Hướng giải bài toán theo tác giả Nguyễn Tam Hùng

Nếu G là một đồ thị Euler (tất cả các đỉnh đều bậc chẵn) thì việc tìm hành trình của đội thanh tra chính là tìm 1 chu trình Euler. Nhưng ta nhận thấy đồ thị G có 31 đỉnh, trong đó :

- Tập đỉnh bậc chẵn V_0={1, 5, 7, 10, 11, 13, 16, 17, 20, 21, 22, 23, 24, 25, 31} (15 đỉnh)

- Tập đỉnh bậc lẻ V_1={2, 3, 4, 6, 8, 9, 12, 14, 15, 18, 19, 26, 27, 28, 29, 30} (16 đỉnh)

Khi đó một hành trình phải đi qua ít nhất 2 lần một số cạnh nào đó. Ta quy ước hành trình T trong đồ thị Euler G_T có được bằng cách vẽ thêm một cạnh song song với mỗi cạnh mà hành trình T đi qua 2 lần là chu trình Euler cần phải tìm. Trước hết để có được đồ thị G_T là đồ thị Euler, ta phải đưa tập đỉnh bậc lẻ V_1 thành đỉnh có bậc chẵn.

Công việc cần làm là phân hoạch các đỉnh trong tập V_1 thành từng cặp sao cho tổng số cạnh thêm vào là ít nhất (số đường đi ngắn nhất). Tính đường đi ngắn nhất giữa các cặp đỉnh lẻ : Tập hợp V_1 có tất cả 16 đỉnh số cặp sẽ là tổ hợp chập 2 của 16 bằng 120. Trong đó có chỉ số cặp đỉnh và Dmin là số cạnh phải nối giữa hai đỉnh.

Bảng 2.1. Số cạnh nối thêm giữa các cặp đỉnh bậc lẻ

Cặp đỉnh	2, 3	2, 4	2, 6	2, 8	2, 9	2, 12	2, 14	2, 15	2, 18	2, 19
D_{min}	1	2	2	4	3	1	3	4	2	3
Cặp đỉnh	2, 26	2, 27	2, 28	2, 29	2, 30	3, 4	3, 6	3, 8	3, 9	3, 12
D_{min}	4	5	6	3	4	1	1	3	2	2
Cặp đỉnh	3, 14	3, 15	3, 18	3, 19	3, 26	3, 27	3, 28	3, 29	3, 30	4, 6
D_{min}	4	5	3	4	5	6	6	4	5	2
Cặp đỉnh	4, 8	4, 9	4, 12	4, 14	4, 15	4, 18	4, 19	4, 26	4, 27	4, 28
D_{min}	2	3	3	5	4	4	5	6	6	5
Cặp đỉnh	4, 29	4, 30	6, 8	6, 9	6, 12	6, 14	6, 15	6, 18	6, 19	6, 26
D_{min}	5	6	2	1	3	3	4	4	3	5
Cặp đỉnh	6, 27	6, 28	6, 29	6, 30	8, 9	8, 12	8, 14	8, 15	8, 18	8, 19
D_{min}	6	5	5	5	3	5	5	4	6	5
Cặp đỉnh	8, 26	8, 27	8, 28	8, 29	8, 30	9, 12	9, 14	9, 15	9, 18	9, 19
D_{min}	7	6	5	7	7	2	2	3	4	2

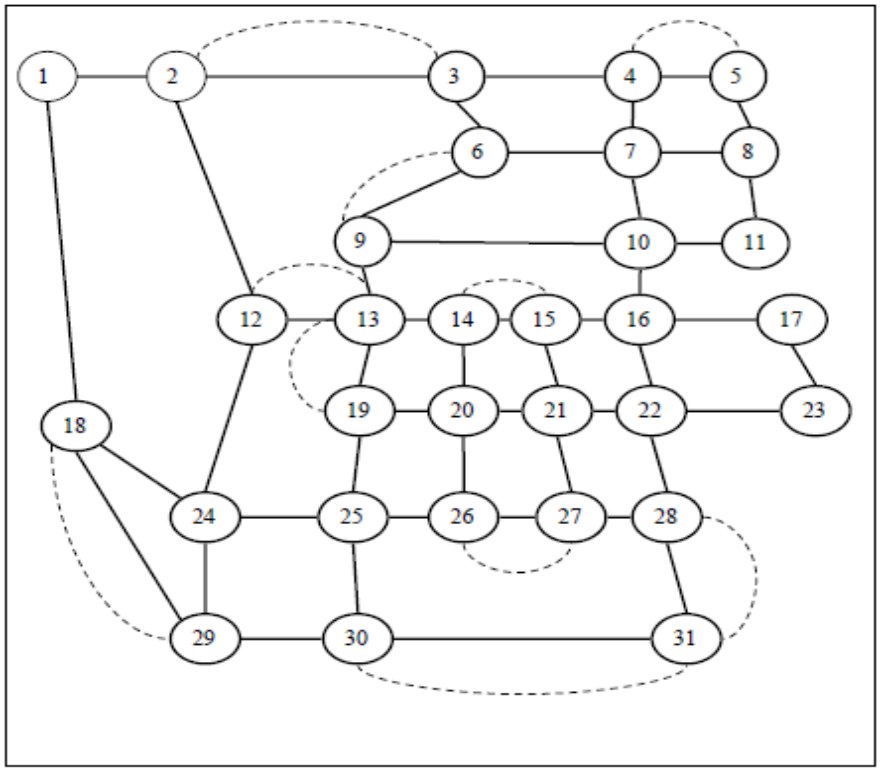
Cặp đỉnh	9, 26	9, 27	9, 28	9, 29	9, 30	12, 14	12, 15	12, 18	12, 19	12, 26
D_{min}	4	5	4	4	4	2	3	2	2	3
Cặp đỉnh	12, 27	12, 28	12, 29	12, 30	14, 15	14, 18	14, 19	14, 26	14, 27	14, 28
D_{min}	4	5	2	3	1	4	2	2	3	4
Cặp đỉnh	14, 29	14, 30	15, 18	15, 19	15, 26	15, 27	15, 28	15, 29	15, 30	18, 19
D_{min}	4	4	5	3	3	2	3	5	5	3
Cặp đỉnh	18, 26	18, 27	18, 28	18, 29	18, 30	19, 26	19, 27	19, 28	19, 29	19, 30
D_{min}	3	4	4	1	2	2	3	4	3	2
Cặp đỉnh	26, 27	26, 28	26, 29	26, 30	27, 28	27, 29	27, 30	28, 29	28, 30	29, 30
D_{min}	1	2	3	2	1	4	3	3	2	1

Sử dụng giải thuật Tham lam, lần lượt chọn các cặp đỉnh sẽ nối có giá trị cạnh thêm là nhỏ nhất, chọn cho đến khi đủ tất cả các đỉnh bậc lẻ của đồ thị như sau:

Bảng 2.2. Cách chọn cặp đỉnh bậc lẻ và số cạnh nối thêm

STT	Cặp đỉnh chọn	D_{min}	Cạnh nối thêm
1	2, 3	1	(2, 3)
2	6, 9	1	(6, 9)
3	14 15	1	(14, 15)
4	18 29	1	(18, 29)
5	26 27	1	(26, 27)
6	4 8	2	(4, 5) ; (5, 8)
7	12 19	2	(12, 13) ; (13, 19)
8	28 30	2	(28, 31) ; (31, 30)

Đồ thị G_T sau khi thêm các cạnh:



Hình 2.2. Đồ thị G_T có được khi thêm cạnh (các nét đứt là các cạnh nối thêm)

Khi đó đồ thị G_T là một đồ thị Euler (tất cả các đỉnh đều bậc chẵn), áp dụng thuật toán *Hierholzer* với đỉnh xuất phát là đỉnh 3 (do trụ sở là nằm tại đỉnh 3)

Bảng 2.3. Chu trình Euler tìm được với đồ thị G_T

Số đỉnh: 31	Số cạnh: 59
Số đỉnh bậc lẻ là: 0	
Có chu trình Euler: 3 2 12 13 19 25 30 31 28 31 30 29 18 29 24 25 26 27 28 22 23 17 16 22 21 27 26 20 21 15 14 20 19 13 14 15 16 10 11 8 5 4 7 10 9 6 9 13 12 24 18 1 2 3 6 7 8 5 4 3	

2.1.3. Nhận xét về bài toán "Thanh tra giao thông" của tác giả Nguyễn Tam Hùng

Bài toán mô hình hóa bản đồ bằng đồ thị vô hướng, không trọng số, việc này nếu áp dụng trong thực tế chưa thật sự khả thi, vì các hành trình đi qua thì mỗi con

đường là có chiều dài là khác nhau, việc đi qua 1 con đường này có thể có chiều dài xa hơn việc đi qua 2 hoặc 3 con đường khác cộng lại.

Để có thể áp dụng được trong thực tiễn, cần mô hình hóa bản đồ đường đi bằng đồ thị có trọng số.

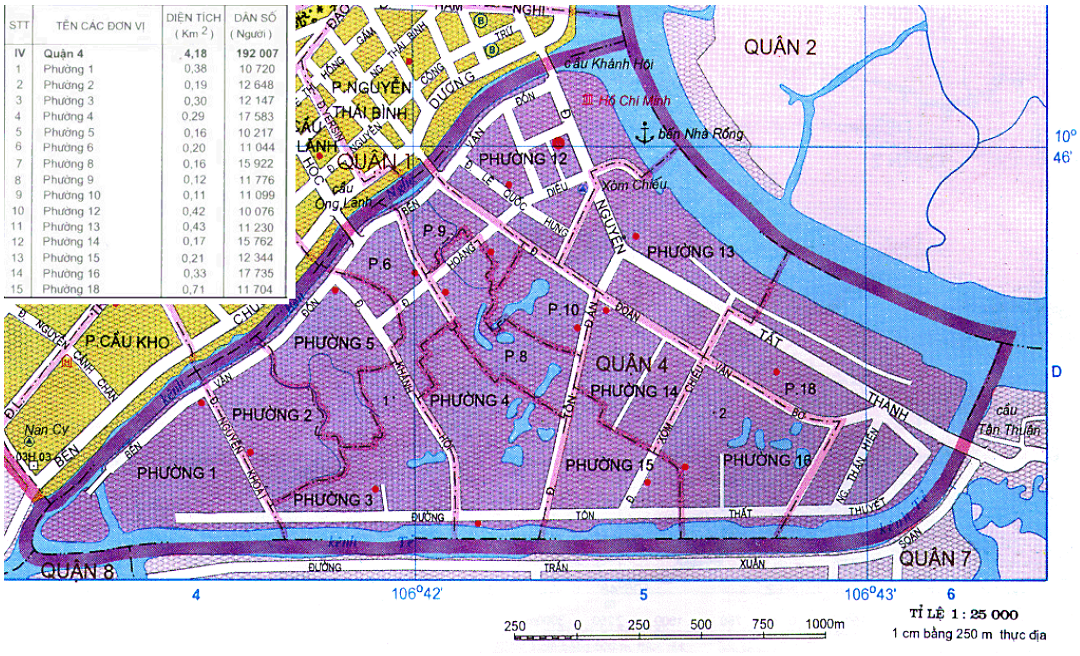
2.2. Đề xuất bài toán "Phân công xe đi thu gom rác thải" tạ Quận 4

2.2.1. Đặt vấn đề

Công ty Cây xanh Quận 4 cần phân công 1 xe thực hiện gom rác thải dọc theo các tuyến đường chính của quận. Rác thải này được các xe gom rác cá nhân tập kết rác đã gom từ các con hẻm ra các tuyến đường chính. Mỗi buổi sáng xe gom rác xuất phát từ Cơ quan, thực hiện đi qua các con đường để thu gom rác và cuối giờ chiều lại quay về Cơ quan hoàn thành công việc của ngày.

Yêu cầu của đề bài đòi hỏi xe phải đi qua các con đường và quay về lại cơ quan. Vì vậy để tiết kiệm chi phí đi lại, bài toán yêu cầu phải vẽ ra được hướng dẫn đường đi cho xe chạy, sao cho chi phí tối ưu nhất.

Bản đồ quận 4 (*Theo trang web Thành Phố Hồ Chí Minh*)

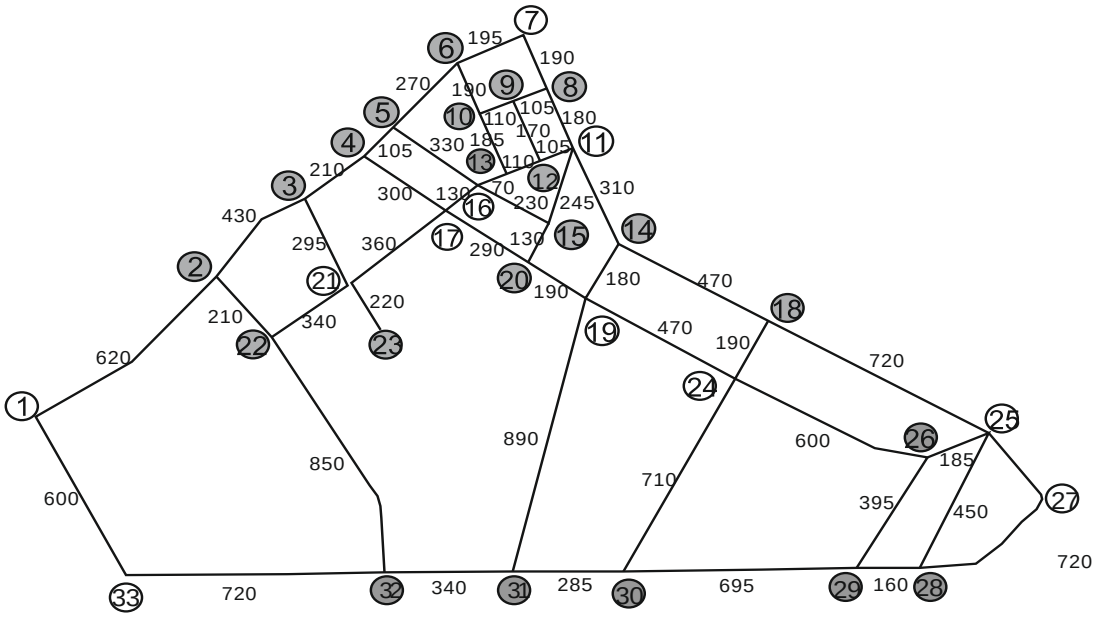


Hình 2.3. Bản đồ giao thông tại quận 4

Bản đồ được trừu tượng hóa bằng ma trận liên thông vô hướng, có trọng số để biểu diễn các con đường như sau:

+ Đỉnh là các giao lộ.

+ Cạnh là các con đường với chiều dài đã biết trước (*chiều dài được ước lượng thực tế lấy từ trang web www.diadiem.com*).



Hình 2.4. Đồ thị mô hình hóa bản đồ quận 4.

2.2.2. Ý tưởng chính của thuật toán

- Mô hình hóa bản đồ thành ma trận đường đi có trọng số.
- Trường hợp tốt nhất thì xe chỉ đi qua mỗi con đường đúng 1 lần duy nhất. Ở trường hợp này thì đồ thị đã thỏa mãn định lý Euler (các đỉnh trên đồ thị đều có bậc chẵn).
- Thực tế thì đồ thị của đề bài là không thỏa định lý Euler (có 22 đỉnh có bậc là lẻ). Chúng ta cần đưa đồ thị về dạng sao cho phải thỏa mãn định lý Euler.
- Thực hiện bằng cách vẽ thêm các cạnh vào đồ thị để các đỉnh đều có bậc chẵn (thỏa định lý Euler). Vấn đề cần giải quyết là nên chọn cạnh nào để vẽ thêm vào đồ thị ban đầu sao cho tổng chiều dài con đường phải đi là ngắn nhất.

2.2.3. Hướng giải quyết bài toán

Nhận xét về số lượng các đỉnh có bậc lẻ:

Một đồ thị chỉ có thể có một số chẵn các đỉnh có bậc lẻ. Vì tổng bậc của tất cả các đỉnh trên đồ thị bằng hai lần số cạnh. Vì vậy tổng số các đỉnh lẻ trên đồ thị luôn luôn là số chẵn.

Các bước thực hiện bài toán:

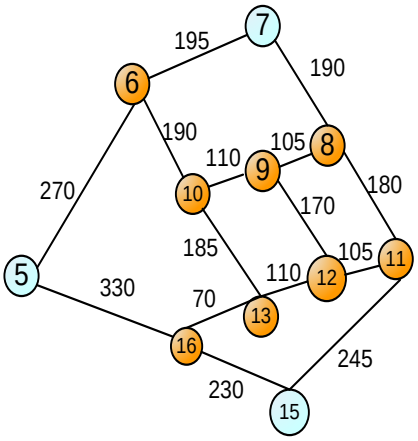
- **Bước 1:** duyệt ma trận đồ thị đầu vào, tìm tất cả các đỉnh có bậc lẻ. Với mỗi đỉnh có bậc lẻ này, tìm đường đi ngắn nhất giữa mọi cặp đỉnh giữa chúng với nhau: -> áp dụng thuật toán floyed tìm đường đi ngắn nhất giữa mọi cặp đỉnh trên đồ thị.
- **Bước 2:** Từ các đỉnh có bậc lẻ đã tìm được ở bước 1, vẽ lại đồ thị mới là đồ thị đầy đủ (mỗi đỉnh nối đến tất cả các đỉnh còn lại), trọng số của mỗi cạnh trên đồ thị đầy đủ này là giá trị đường đi ngắn nhất đã tìm được ở bước 1.
- **Bước 3***:* Tìm bộ ghép cực đại có trọng số cực tiểu trên đồ thị đầy đủ bằng giải thuật Tham lam hoặc giải thuật "tìm bộ ghép tối ưu cực đại có trọng số cực tiểu trên đồ thị đầy đủ".

Thêm các cạnh tìm được này vào ma trận ban đầu, các cạnh được thêm vào theo đường đi Floyed đã tìm được, đưa đồ thị về dạng thỏa tất cả các đỉnh đều có bậc chẵn.

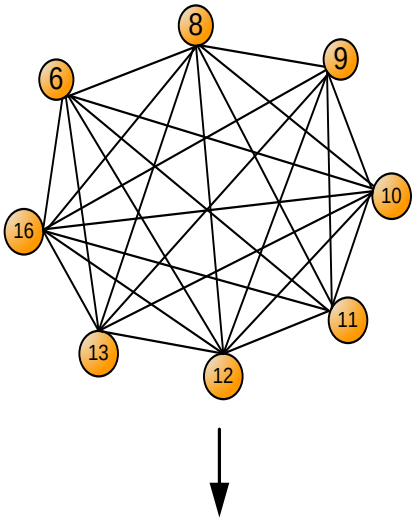
- **Bước 4***:* Dùng thuật toán Fleury duyệt chu trình Euler trên đồ thị mới này và in ra kết quả.

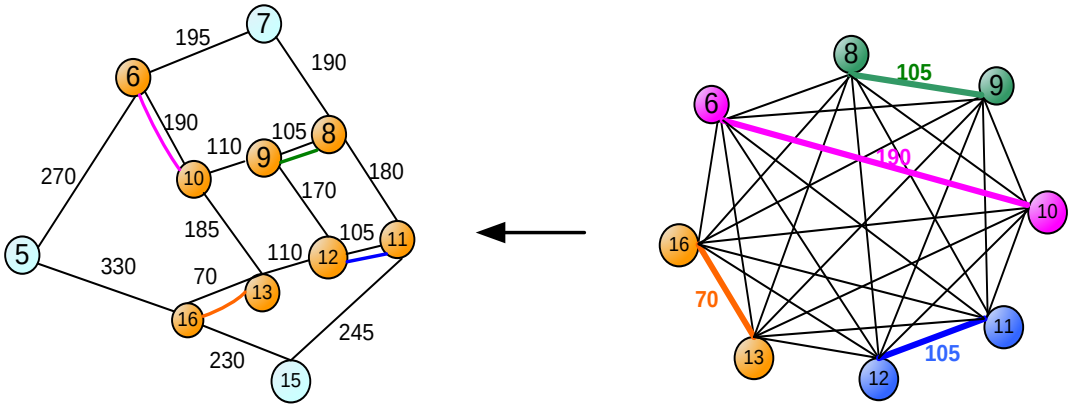
Ví dụ 2.1. Lấy một phần bản đồ Quận 4 gồm 11 đỉnh, mô hình hóa thành đồ thị gồm 11 đỉnh để tìm đường đi ngắn nhất.

Từ đồ thị ban đầu -> tìm các đỉnh có bậc lẻ -> tìm đường đi ngắn nhất giữa tất cả các đỉnh này



Vẽ lại đồ thị đầy đủ với các trọng số đã tìm được ở B1





Nối thêm các cạnh tìm được ở bước 3 vào đồ thị ban đầu để có các đỉnh toàn bậc chẵn, dùng thuật toán Fleury duyệt và in ra chu trình Euler

Tìm bộ cặp ghép có tổng trọng số là nhỏ nhất (có 4 cặp ghép)

In ra đường đi: 5->6->7->8->9->8->11->12->9->10->6->10->13->12->11->15->16->13->16->5

Hình 2.5. Ví dụ các bước giải bài toán tìm đường đi ngắn nhất

2.2.4. Ứng dụng giải bài toán phân công việc thực tế tại Quận 4

Mô hình hóa bản đồ các con đường cần đi qua ở Quận 4 bằng ma trận trọng số tương ứng. Tiếp theo sẽ thực hiện các bước sau:

Bước 1: Từ ma trận đồ thị ban đầu -> tìm các đỉnh có bậc lẻ và tìm đường đi ngắn nhất giữa mọi cặp đỉnh có bậc lẻ với nhau:

- Biểu diễn ma trận đồ thị ban đầu (33 đỉnh)

Bảng 2.4. Ma trận đồ thị ban đầu

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	0	620	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	600
2	620	0	430	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	210	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3	-1	430	0	210	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	295	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
4	-1	-1	210	0	105	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	300	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
5	-1	-1	-1	105	0	270	-1	-1	-1	-1	-1	-1	-1	-1	-1	330	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	270	0	195	-1	-1	190	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
7	-1	-1	-1	-1	-1	195	0	190	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
8	-1	-1	-1	-1	-1	-1	190	0	105	-1	180	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
9	-1	-1	-1	-1	-1	-1	-1	105	0	110	-1	170	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
10	-1	-1	-1	-1	-1	190	-1	-1	110	0	-1	-1	185	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
11	-1	-1	-1	-1	-1	-1	-1	180	-1	-1	0	105	-1	310	245	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
12	-1	-1	-1	-1	-1	-1	-1	-1	170	-1	105	0	110	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
13	-1	-1	-1	-1	-1	-1	-1	-1	-1	185	-1	110	0	-1	-1	70	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
14	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	310	-1	-1	0	-1	-1	-1	470	180	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
15	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	245	-1	-1	-1	0	230	-1	-1	-1	130	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
16	-1	-1	-1	-1	330	-1	-1	-1	-1	-1	-1	-1	70	-1	230	0	130	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
17	-1	-1	-1	300	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	130	0	-1	-1	290	360	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
18	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	470	-1	-1	-1	0	-1	-1	-1	-1	-1	190	720	-1	-1	-1	-1	-1	-1	-1	-1
19	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	180	-1	-1	-1	-1	0	190	-1	-1	-1	470	-1	-1	-1	-1	-1	-1	890	-1	-1

20	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	130	-1	290	-1	190	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
21	-1	-1	295	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	360	-1	-1	-1	0	340	220	-1	-1	-1	-1	-1	-1	-1	-1
22	-1	210	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	340	0	-1	-1	-1	-1	-1	-1	-1	-1	850	-1
23	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	220	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
24	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	190	470	-1	-1	-1	-1	-1	0	-1	600	-1	-1	-1	710	-1	-1	-1
25	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	720	-1	-1	-1	-1	-1	-1	0	185	270	450	-1	-1	-1	-1	-1
26	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	600	185	0	-1	-1	395	-1	-1	-1	-1
27	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	270	-1	0	470	-1	-1	-1	-1	-1
28	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	450	-1	470	0	160	-1	-1	-1	-1
29	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	395	-1	160	0	695	-1	-1	-1
30	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	710	-1	-1	-1	-1	695	0	285	-1	-1
31	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	890	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	285	0	340	-1
32	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	850	-1	-1	-1	-1	-1	-1	-1	-1	340	0	720
33	600	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	720	0

- Tìm ma trận các đỉnh bậc lẻ (22 đỉnh)

Bảng 2.5. Ma trận các đỉnh có bậc là lẻ

	2	3	4	5	6	8	9	10	12	13	14	15	18	20	22	23	26	28	29	30	31	32
2	0	430	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	210	-1	-1	-1	-1	-1	-1	-1
3	430	0	210	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
4	-1	210	0	105	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
5	-1	-1	105	0	270	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	270	0	-1	-1	190	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
8	-1	-1	-1	-1	-1	0	105	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
9	-1	-1	-1	-1	-1	105	0	110	170	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
10	-1	-1	-1	-1	190	-1	110	0	-1	185	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
12	-1	-1	-1	-1	-1	-1	170	-1	0	110	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
13	-1	-1	-1	-1	-1	-1	-1	185	110	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
14	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	470	-1	-1	-1	-1	-1	-1	-1	-1	-1
15	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	130	-1	-1	-1	-1	-1	-1	-1	-1
18	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	470	-1	0	-1	-1	-1	-1	-1	-1	-1	-1	-1
20	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	130	-1	0	-1	-1	-1	-1	-1	-1	-1	-1
22	210	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1	850
23	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	-1	-1	-1	-1	-1
26	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	-1	395	-1	-1	-1
28	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0	160	-1	-1	-1
29	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	395	160	0	695	-1	-1
30	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	695	0	285	-1
31	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	285	0	340
32	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	850	-1	-1	-1	-1	-1	340	0

- Tìm ma trận đường đi ngắn nhất giữa các đỉnh (33 đỉnh) theo thuật toán Floyd

Bảng 2.6. Ma trận đường đi ngắt nhất giữa các đỉnh bậc lẻ

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	1	1	2	3	4	5	6	7	10	6	22	22	22	22	22	22	22	22	22	22	22	2	22	22	33	33	33	33	33	33	33	33	1
2	2	2	2	3	4	5	6	7	10	6	22	22	22	22	22	22	22	22	22	22	22	2	22	22	26	24	26	32	32	32	32	22	1
3	2	3	3	3	4	5	6	7	10	6	17	17	17	20	17	17	4	20	20	17	3	21	21	20	26	24	26	29	26	32	32	22	2
4	3	3	4	4	4	5	6	7	10	6	17	17	17	20	17	17	4	20	20	17	3	21	21	20	26	24	26	29	26	31	20	22	3
5	4	4	4	5	5	5	6	7	10	6	16	16	16	16	16	5	4	16	20	16	4	21	21	20	18	24	25	29	26	31	20	22	4
6	5	5	5	5	6	6	6	7	10	6	8	10	10	11	16	13	16	14	20	16	5	21	21	20	18	24	25	25	26	31	20	22	5
7	6	6	6	6	6	7	7	7	8	6	8	9	10	11	11	13	16	14	14	15	6	21	21	19	18	24	25	25	26	31	19	31	6
8	7	7	7	7	7	7	8	8	8	9	8	9	12	11	11	13	16	14	14	15	17	21	21	19	18	24	25	25	26	31	19	31	7

9	10	10	10	10	10	10	8	9	9	9	12	9	12	12	12	13	16	14	14	15	17	21	21	19	18	24	25	25	26	31	19	31	10
10	6	6	6	6	6	10	6	9	10	10	12	9	10	12	16	13	16	14	20	16	17	21	21	20	18	24	25	25	26	31	20	22	6
11	22	22	17	17	16	8	8	11	12	12	11	11	12	11	11	13	16	14	14	15	17	21	21	19	18	24	25	25	26	31	19	31	32
12	22	22	17	17	16	10	9	9	12	9	12	12	12	11	11	13	16	14	14	15	17	21	21	19	18	24	25	25	26	31	19	31	22
13	22	22	17	17	16	10	10	12	12	13	12	13	13	12	16	13	16	14	20	16	17	21	21	20	18	24	25	25	26	31	20	22	22
14	22	22	20	20	16	11	11	11	12	12	14	11	12	14	20	13	20	14	14	19	20	21	21	19	18	24	25	25	26	31	19	31	32
15	22	22	17	17	16	16	11	11	12	16	15	11	16	20	15	15	16	20	20	15	17	21	21	20	26	24	26	29	26	31	20	31	32
16	22	22	17	17	16	13	13	13	13	13	13	13	16	13	16	16	16	14	20	15	17	21	21	20	18	24	25	29	26	31	20	22	22
17	22	22	4	17	4	16	16	16	16	16	16	16	16	20	16	17	17	20	20	17	17	21	21	20	26	24	26	29	26	31	20	22	22
18	22	22	20	20	16	14	14	14	14	14	14	14	14	18	20	14	20	18	14	19	20	21	21	18	18	24	25	25	26	24	30	31	32
19	22	22	20	20	20	20	14	14	14	20	14	14	20	19	20	20	20	14	19	19	20	21	21	19	26	24	26	29	26	31	19	31	32
20	22	22	17	17	16	16	15	15	15	16	15	15	16	19	20	15	20	19	20	20	17	21	21	19	26	24	26	29	26	31	19	31	32
21	22	22	21	3	4	5	6	17	17	17	17	17	17	20	17	17	21	20	20	17	21	21	21	20	26	24	26	29	26	32	32	22	22
22	2	22	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	22	22	21	21	26	24	26	32	32	32	32	22	2
23	22	22	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	21	23	21	23	21	26	24	26	29	26	32	32	22	22
24	22	22	20	20	20	20	19	19	19	20	19	19	20	19	20	20	20	24	24	19	20	21	21	24	26	24	26	29	26	24	30	31	32
25	33	26	26	26	18	18	18	18	18	18	18	18	18	18	26	18	26	25	26	26	26	26	26	26	25	25	25	25	26	29	30	31	32
26	33	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	26	26	26	25	29	26	29	30	31	32
27	33	26	26	26	25	25	25	25	25	25	25	25	25	25	26	25	26	25	26	26	26	26	26	26	27	25	27	27	28	29	30	31	32
28	33	32	29	29	29	25	25	25	25	25	25	25	25	25	29	29	29	25	29	29	29	32	29	29	28	29	28	28	28	29	30	31	32
29	33	32	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	32	26	26	26	29	28	29	29	29	30	31	32
30	33	32	32	31	31	31	31	31	31	31	31	31	31	31	31	31	31	24	31	31	32	32	32	30	29	29	29	29	30	30	30	31	32
31	33	32	32	20	20	20	19	19	19	20	19	19	20	19	20	20	20	30	31	19	32	32	32	30	30	30	30	30	30	31	31	31	32
32	33	22	22	22	22	22	31	31	31	22	31	31	22	31	31	22	22	31	31	31	22	32	22	31	31	31	31	31	31	31	32	32	32
33	33	1	2	3	4	5	6	7	10	6	32	22	22	32	32	22	22	32	32	32	22	2	22	32	32	32	32	32	32	32	32	33	33

Bước 2: Tìm chiều dài đường đi ngắn nhất của đường đi giữa các đỉnh bậc lẻ

Bảng 2.7. Ma trận chiều dài ngắn nhất giữa các đỉnh có bậc lẻ

	2	3	4	5	6	8	9	10	12	13	14	15	18	20	22	23	26	28	29	30	31	32
2	0	430	640	745	1015	1400	1315	1205	1220	1110	1570	1270	2040	1200	210	770	2460	2540	2380	1685	1400	1060
3	430	0	210	315	585	970	885	775	820	710	1170	870	1640	800	635	515	2060	2615	2455	2110	1825	1485
4	640	210	0	105	375	760	675	565	610	500	960	660	1430	590	845	725	1850	2405	2245	1955	1670	1695
5	745	315	105	0	270	655	570	460	510	400	925	560	1395	690	950	830	1950	2505	2345	2055	1770	1800
6	1015	585	375	270	0	385	300	190	470	375	875	675	1345	805	1220	1100	2065	2515	2460	2170	1885	2070
8	1400	970	760	655	385	0	105	215	275	385	490	425	960	555	1285	1165	1740	2130	2135	1845	1560	1900
9	1315	885	675	570	300	105	0	110	170	280	585	520	1055	650	1180	1060	1835	2225	2230	1940	1655	1995
10	1205	775	565	460	190	215	110	0	280	185	695	485	1165	615	1085	965	1875	2335	2270	1980	1695	1935
12	1220	820	610	510	470	275	170	280	0	110	415	350	885	480	1010	890	1665	2055	2060	1770	1485	1825
13	1110	710	500	400	375	385	280	185	110	0	525	300	995	430	900	780	1690	2165	2085	1795	1510	1750
14	1570	1170	960	925	875	490	585	695	415	525	0	500	470	370	1360	1240	1250	1640	1645	1355	1070	1410
15	1270	870	660	560	675	425	520	485	350	300	500	0	970	130	1060	940	1390	1945	1785	1495	1210	1550
18	2040	1640	1430	1395	1345	960	1055	1165	885	995	470	970	0	840	1830	1710	790	1170	1185	900	1185	1525
20	1200	800	590	690	805	555	650	615	480	430	370	130	840	0	990	870	1260	18 15	1655	1365	1080	1420
22	210	635	845	950	1220	1285	1180	1085	1010	900	1360	1060	1830	990	0	560	2250	2330	2170	1475	1190	850
23	770	515	725	830	1100	1165	1060	965	890	780	1240	940	1710	870	560	0	2130	2685	2525	2035	1750	1410
26	2460	2060	1850	1950	2065	1740	1835	1875	1665	1690	1250	1390	790	1260	2250	2130	0	555	395	1090	1375	1715
28	2540	2615	2405	2505	2515	2130	2225	2335	2055	2165	1640	1945	1170	1815	2330	2685	555	0	160	855	1140	1480
29	2380	2455	2245	2345	2460	2135	2230	2270	2060	2085	1645	1785	1185	1655	2170	2525	395	160	0	695	980	1320
30	1685	2110	1955	2055	2170	1845	1940	1980	1770	1795	1355	1495	900	1365	1475	2035	1090	855	695	0	285	625
31	1400	1825	1670	1770	1885	1560	1655	1695	1485	1510	1070	1210	1185	1080	1190	1750	1375	1140	980	285	0	340
32	1060	1485	1695	1800	2070	1900	1995	1935	1825	1750	1410	1550	1525	1420	850	1410	1715	1480	1320	625	340	0

Bước 3: Ma trận cặp ghép có tổng chiều dài nhỏ nhất (có 11 cặp ghép)

Bảng 2.8. Ma trận cặp ghép tối ưu có trọng số nhỏ nhất

Số cặp ghép: 11
Tổng chiều dài: 3425

Đỉnh đầu	Đỉnh cuối	Chiều dài
2	22	210
3	23	515
4	5	105
6	10	190
8	9	105
12	13	110
14	18	470
15	20	130
26	28	555
29	30	695
31	32	340

- Ma trận bậc chẵn có được sau khi thêm các cặp ghép tìm được vào ma trận ban đầu:

Bảng 2.9. Ma trận bậc chẵn có được sau khi thêm các cặp ghép

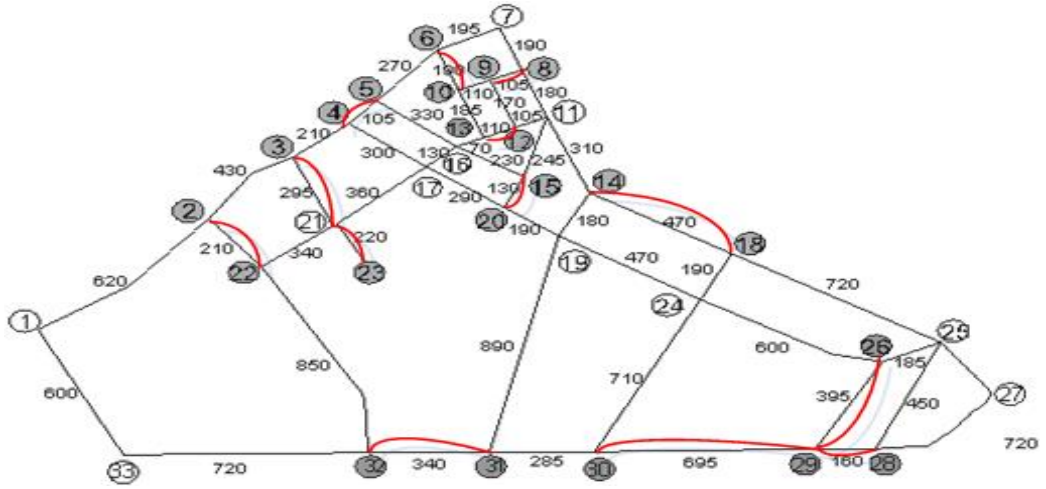
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0
3	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0
4	0	0	1	0	2	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	2	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	1	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	1	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	2	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	2	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	1	0	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	1	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
22	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
23	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	2	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	2	0	1	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	2	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2	0	1
33	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Bước 4: Duyệt ma trận tìm được ở bước 3 theo thuật toán Fleury, in ra chu trình đường đi Euler là kết quả cần tìm.

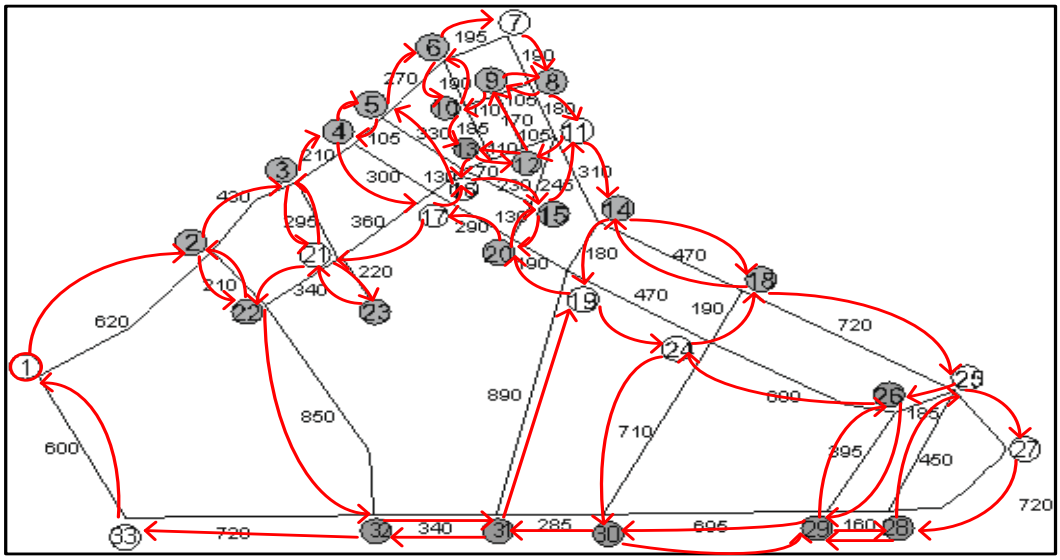
- Duyệt Ma trận và in ra chu trình đường đi cần tìm:

1 → 2 → 3 → 4 → 5 → 4 → 17 → 16 → 5 → 6 → 7 → 8 → 9 → 8 → 11 → 12 → 9 → 10 → 6 → 10 → 13 → 12 → 13 → 16 → 15 → 11 → 14 → 18 → 14 → 19 → 20 → 15 → 20 → 17 → 21 → 3 → 21 → 23 → 21 → 22 → 2 → 22 → 32 → 31 → 19 → 24 → 18 → 25 → 26 → 24 → 30 → 29 → 26 → 29 → 28 → 25 → 27 → 28 → 29 → 30 → 31 → 32 → 33 → 1

+ Số đỉnh chu trình đi qua: 33, số cạnh chu trình đi qua: 63



Hình 2.6. Mô hình đồ thị Euler sau khi thêm một số con đường đi 2 lần



Hình 2.7. Mô hình đường Euler sau khi duyệt bằng Fleury

Chương 3. ĐÁNH GIÁ

3.1. Độ phức tạp của các thuật toán được sử dụng trong bài toán

Giả sử bài toán cho đồ thị G(E, V) có n đỉnh, trong đó có K đỉnh có bậc là lẻ, chúng ta xác định độ phức tạp tính toán như sau:

Ở bước 1 và bước 2: Dùng thuật toán Floyd tìm đường đi ngắn nhất giữa mọi cặp đỉnh trên đồ thị ban đầu. Thuật toán Floyd sử dụng 03 vòng for lồng nhau để truy vét tất cả các trường hợp tìm đường đi để so sánh các trọng số với giá trị min nên có độ phức tạp tính toán được ước lượng là $O(n)^3$.

Ở bước 3: Thuật toán đề xuất 02 thuật toán khác nhau là giải thuật Tham lam và giải thuật tìm bộ ghép cực đại có trọng số nhỏ nhất trên đồ thị đầy đủ (Find-MinMatch), độ phức tạp tính toán của hai giải thuật này được ước lượng như sau:

+ Giải thuật Tham lam sử dụng 2 vòng lặp for lồng nhau để duyệt ma trận, nên có độ phức tạp tính toán được ước lượng là $O(k)^2$.

+ Giải thuật FindMinMatch sử dụng 3 vòng lặp for lồng nhau để tính toán các trường hợp, trong vòng lặp for con có gọi lại đệ quy trong điều kiện số đỉnh ghép <= $\frac{k}{2}$, độ phức tạp tính toán ước lượng là $O(k)^3$ $(\frac{k}{2})!$

Ở bước 4: Thuật toán Fleury sử dụng 3 vòng lặp for lồng nhau để duyệt ma trận bậc chẵn và so sánh giá trị lưu trung gian để tìm đường đi nên có độ phức tạp tính toán được ước lượng là $O(n)^3$.

Thời gian xử lý của toàn bài toán phụ thuộc rất lớn vào quyết định lựa chọn giải thuật xử lý ở bước 3 của giải pháp. Vì vậy cần đi sâu vào phân tích, đánh giá ưu nhược điểm của hai giải pháp này.

3.2. Đánh giá giải pháp dùng giải thuật Tham lam so với giải thuật Find-MinMatch

3.2.1. Giải thuật Tham lam

Ưu điểm: Chỉ sử dụng 2 vòng lặp for lồng nhau thực hiện duyệt tất cả các đỉnh để tìm giá trị nhỏ nhất nên có độ phức tạp tính toán là $O(n)^2$

Nhược điểm: Đối với bài toán có dữ liệu đầu vào lớn, giải thuật Tham lam không cho ra được giá trị tối ưu nhất, do việc lựa chọn giá trị tối ưu tại bước đã thực hiện trước đó có thể làm mất đi các giá trị tối ưu hơn ở bước tiếp theo của bài toán con phát sinh sau khi đã chọn giá trị tốt nhất ở bước xử lý trước đó.

3.2.2. Giải thuật FindMinMatch

Ưu điểm: Thuật toán tìm ra được giá trị tối ưu nhất theo đề bài.

*Nhược điểm:*Thuật toán dùng phương pháp quét cạn tất cả mọi cặp đỉnh có thể, cần duyệt qua $\frac{k}{2}$ đỉnh lẻ để tìm $\frac{k}{2}$ cặp ghép, và gọi lại $(\frac{k}{2})$-1 lần, nên có độ phức tạp tính toán được ước lượng là $O(\frac{k}{2})!$, thời gian xử là khá chậm đối với các bài toán có dữ liệu đầu vào lớn gồm nhiều đỉnh có bậc lẻ thì thời gian xử lý càng chậm.

3.2.3. So sánh hiệu quả của 02 giải thuật trên đối với "bài toán phân công xe đi thu gom rác thải" tại Quận 4

- Đồ thị được xây dựng để mô hình hóa bản đồ giao thông quận 4 là ma trận vuông gồm 33 đỉnh, trong đó có 22 đỉnh có bậc lẻ.

- Máy vi tính dùng kiểm thử thời gian xử lý cho bài toán này là: Laptop Dell Inspiron N4010, Ram 4GB, CPU Core i5 2.53Ghz.

Bảng 3.1. So sánh giữa hai giải thuật Tham lam và FindMinMatch về thời gian chạy và giá trị min tìm được

Thông tin Test	Giải thuật Tham Lam	FindMinMatch
Thời gian chạy (giây)	1.023	1.457
Giá trị Min tìm được (mét)	3995	3425

Nhận xét:

+ **Thời gian xử lý**: 2 giải thuật là khá tương đồng, không chênh lệch nhiều về thời gian xử lý vì số lượng đỉnh có bậc lẻ ở đầu bài là không lớn (chỉ 22 đỉnh).

+ **Giá trị min tìm được**: so với giải thuật Tham lam thì giải thuật FindMinMatch tìm được giá trị tối ưu hơn rất nhiều:

+ Tổng chiều dài đường đi xe phải thực hiện trên mỗi hành trình theo đồ thị của đề bài là 19.900 mét.

+ Mỗi chu trình đường đi giảm được nếu dùng giải thuật FindMinMatch so với giải thuật Tham lam: 3995 mét (theo thuật toán Tham lam) – 3425 mét (theo thuật toán FindMinMatch) = **570 mét**.

KẾT LUẬN

Đề tài đã đề xuất thêm hướng giải quyết cho bài toán tìm chu trình đường đi ngắn nhất qua tất cả các cạnh trên đồ thị vô hướng, có trọng số.

Bài toán tìm đường đi ngắn nhất trong đề tài có sử dụng giải pháp tìm cặp ghép tối ưu với điều kiện đề bài yêu cầu tổng chiều dài đường đi phải là nhỏ nhất có thể. Thuật toán ở bước xử lý này có ảnh hưởng rất lớn đến tốc độ xử lý cũng như giá trị tối ưu tìm được chung của cả bài toán là hai giải pháp Tham lam và FindMinMatch.

Tuỳ vào nhu cầu thực tế mà có thể áp dụng một trong hai giải thuật này để thực thi cho bài toán. Đối với các bài toán có chiều dài của các cạnh trên đồ thị không chênh lệch nhau nhiều như: mạch điện, ô cờ… cũng như chi phí cho đường đi không lớn thì có thể sử dụng giải thuật Tham lam. Ngược lại, đối với các bài toán có chiều dài của cạnh trên đồ thị chênh lệch nhau nhiều như: lộ trình đường đi giao thông, tín hiệu GPS… hoặc chi phí cho đường đi lớn thì việc áp dụng giải Thuật FindMinMatch sẽ mang lại hiệu quả tiết kiệm cao về chi phí.

Đối với các bài toán có ma trận dữ liệu đầu vào dưới 30 đỉnh bậc có lẻ thì thuật toán FindMinMatch có thể xử lý tốt và cho giá trị đường đi là tối ưu nhất có thể.

Ngoài ra, giải pháp xử lý cho bài toán chung có thể được tối ưu hơn nữa, nếu sử dụng kết hợp giữa 2 giải thuật này sẽ tiết giảm thời gian tính toán cho bài toán lớn cần tìm kết quả tối ưu nhất như sau:

- Dùng giải thuật Tham lam tìm ra kết quả Min để tham chiếu.
- Dùng thuật toán FindMinMatch lấy đầu vào chặn trên là kết quả Min tìm được của giải thuật Tham lam, để khống chế bớt các bước xử lý tìm kiếm bằng cách so sánh nếu giá trị tổng Min cần tìm phải nhỏ hơn giá trị Min của giải thuật Tham lam đưa ra thì sẽ thực hiện tiếp việc tính toán các bước tiếp theo.

HƯỚNG PHÁT TRIỂN CỦA LUẬN VĂN

Trong thực tiễn thì các con đường giao thông luôn có phân biệt về hướng đi là đường một chiều hoặc đường hai chiều. Vì vậy, để áp dụng triệt để vào thực tiễn, cần xây dựng giải pháp xử lý cho bài toán có đồ thị đầu vào dạng có hướng và có trọng số.

Đề tài tập trung vào giả thuyết của đầu bài chỉ phần công 1 xe thực hiện hành trình qua tất cả các con đường trong một khu vực làm việc cho trước nào đó. Tuy nhiên, trong thực tế, công việc này không chỉ phân công cho một xe thực hiện mà có thể phân công cho một nhóm xe thực tùy ý. Vì vậy đề tài cần được nghiên cứu phát triển nâng cấp để đưa ra giải pháp xử lý cho bài toán phân công lịch trình đường đi ngắn nhất qua tất cả các con đường cho số lượng K xe tuỳ ý.

TÀI LIỆU THAM KHẢO

[1] Gauvain Chaste, Aurélien Ooms and Robin Walravens (2014), *Chinese postman problem*, [online], viewed 10 May 2015, from: <http://aureooms.wolkom.net /in-fof521/pdf/report.pdf>.

[2] Kenneth H. Rosen (Phạm Văn Thiều và Đặng Hữu Thịnh dịch, 2003), *Toán rời rạc ứng dụng trong tin học*, NXB Khoa học và Kỹ thuật, Hà Nội.

[3] Lê Minh Hoàng (1999-2002), *Giải thuật và lập trình*, NXB Hà Nội, Hà Nội.

[4] Nguyễn Tam Hùng (2014), *Các thuật toán về đường đi và chu trình Euler và ứng dụng*, Luận văn Thạc sỹ Khoa học máy tính, Đại học Thái Nguyên.

[5] Nguyễn Xuân Huy (2011), *Sáng tạo trong thuật toán và lập trình,* NXB Thông tin và truyền thông, Hà Nội.

PHỤ LỤC

1. Tổng hợp dữ liệu và kết quả test cho cả 2 phương án dùng giải thuật Tham lam và FindMinMatch

Thực hiện test chương trình của hai thuật toán bằng các đồ thị khác nhau, trên máy Laptop Dell Inspiron N4010, Core i5 2.53Ghz, Ram 4GB.

Kết quả về thời gian chạy và giá trị Min tìm được như bảng bên dưới.

Các đồ thị kiểm thử được tăng dần về số lượng đỉnh để tăng thời gian tính toán.

Test 1: đồ thị gồm 12 đỉnh, có 4 đỉnh bậc lẻ

+ Ma trận đầu vào (12x12.txt)

```
12
1  2   3   4   5   6   7   8  9 10 11 12
0  20  12  15  18 -1  -1  -1 -1 -1 -1 -1
20  0 -1  15 -1  15 -1  -1  -1 -1 -1 -1
12 -1   0   8 -1  -1  16  28 -1 -1 -1 -1
15 15   8   0 -1  -1  -1  -1 -1 -1 -1 -1
18 -1  -1  -1  0 -1  15 -1 -1 -1 -1 -1
-1 15  -1  -1  -1   0 -1  -1 -1 -1 30 -1
-1 -1  16  -1  15  -1   0 -1 -1 25 -1 -1
-1 -1  28  -1  -1  -1  -1  0 21 -1 -1 -1
-1 -1  -1  -1  -1  -1  -1 21  0 -1 -1 17
-1 -1  -1  -1  -1  -1  25 -1 -1  0 -1 27
-1 -1  -1  -1  -1  30 -1 -1 -1 -1  0 31
-1 -1  -1  -1  -1  -1  -1 -1 17 27 31  0
```

+ Thời gian chạy

Đồ thị đầu vào	Thông tin Test	Tham lam	FindMinMatch
Đồ thị test 1 (12 đỉnh: có 4 đỉnh bậc lẻ)	Thời gian chạy (giây)	0,725	0,665
	Giá trị Min tìm được	67	67

```
//load len do thi ban dau
char *tenfile = "C:\\Users\\admin\\Desktop\\Test\\Timduongdi_ThamLam\\Vidu\\12x12.txt";
LoadGraph(tenfile, doThiBanDau, n1);
```

```
C:\Users\admin\Desktop\Test\Timduongdi_ThamLam\NguoiDuaThu.exe
so Canh: 18
Chay hoan tat, de xem ket qua, vui long mo file: 4_chuTrinhthucte_ThamLam.txt
--------------------------------
Process exited after 0.7256 seconds with return value 0
Press any key to continue . . .
```

```
//Load len do thi ban dau
char *tenfile = "C:\\Users\\admin\\Desktop\\Test\\Timduongdi_FindMinMatch\\Vidu\\12x12.txt";
LoadGraph(tenfile, doThiBanDau, n1);
```

```
C:\Users\admin\Desktop\Test\Timduongdi_FindMinMatch\NguoiDuaThu.exe
so Canh: 18
Chay hoan tat, de xem ket qua, vui long mo file: 4_chuTrinhthucte_FindMinMatch.t
xt

--------------------------------
Process exited after 0.6655 seconds with return value 0
Press any key to continue . . .
```

Test 2: đồ thị gồm 35 đỉnh, có 20 đỉnh bậc lẻ

+ Ma trận đầu vào (35x35.txt)

```
35
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
0 -1 -1 -1 -1 33 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 0 -1 -1 -1 -1 -1 -1 -1 -1 12 33 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1
-1 -1 0 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 123 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 -1 -1 -1 213 -1 -1 -1 -1 -1 -1 -1 32 -1
-1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 21 -1 -1 -1
33 -1 21 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 76 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 -1 -1 -1 67 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 55 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 332 -1 -1 -1 -1 -1 21 -1 -1 432 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1
-1 12 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 33 -1 43 -1 -1 -1 -1 -1 332 -1 0 -1 213 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 54 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 213 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 44 -1 44 -1 -1 -1 -1 -1 43
-1 -1 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 543 -1 -1 -1 -1 -1 -1
-1 -1 32 -1 -1 -1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 -1
-1 -1 -1 65 -1 76 -1 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 22 -1 -1 -1 -1 -1 76 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 47 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 432 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 47 -1 -1 -1
-1 -1 123 -1 -1 -1 -1 -1 -1 -1 432 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 75 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 67 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 213 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 22 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 44 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 65 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 44 -1 -1 543 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 76 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 0 -1 -1 -1 95 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 47 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 32 -1 -1
-1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 76 47 -1 -1 -1 -1 -1 -1 -1 -1 76 -1 -1 0 -1 -1 -1
-1 32 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 0 -1 -1
-1 -1 -1 32 -1 -1 -1 -1 55 -1 -1 54 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 75 -1 -1 -1 -1 -1 -1 95 -1 -1 -1 0 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0
```

+ Thời gian chạy

Đồ thị đầu vào	Thông tin Test	Tham lam	FindMinMatch
Đồ thị test 2 (35 đỉnh: có 20 đỉnh bậc lẻ)	Thời gian chạy (giây)	0,774	4,802
	Giá trị Min tìm được	2162	1882

```
//Load len do thi ban dau
char *tenfile = "C:\\Users\\admin\\Desktop\\Test\\Timduongdi_ThamLam\\Vidu\\35x35.txt";
LoadGraph(tenfile, doThiBanDau, n1);
```

```
C:\Users\admin\Desktop\Test\Timduongdi_ThamLam\NguoiDuaThu.exe
so Canh: 52
Chay hoan tat, de xem ket qua, vui long mo file: 4_chuTrinhthucte_ThamLam.txt

--------------------------------
Process exited after 0.7049 seconds with return value 0
Press any key to continue . . .
```

```
//Load len do thi ban dau
char *tenfile = "C:\\Users\\admin\\Desktop\\Test\\Timduongdi_FindMinMatch\\Vidu\\35x35.txt";
LoadGraph(tenfile, doThiBanDau, n1);
```

```
C:\Users\admin\Desktop\Test\Timduongdi_FindMinMatch\NguoiDuaThu.exe
so Canh: 52
Chay hoan tat, de xem ket qua, vui long mo file: 4_chuTrinhthucte_FindMinMatch.t
xt

--------------------------------
Process exited after 4.802 seconds with return value 0
Press any key to continue . . .
```

- **Test 3**: đồ thị gồm 40 đỉnh, có 22 đỉnh bậc lẻ

+ Ma trận đầu vào (40x40.txt)

```
40
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40
0 -1 -1 -1 -1 33 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 0 -1 -1 -1 -1 -1 -1 -1 -1 12 33 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1
-1 -1 0 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 123 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 21
-1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 -1 -1 -1 213 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 0 -1 -1 23 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1
33 -1 21 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 76 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1
-1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 -1 -1 -1 67 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 23 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 55 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 332 -1 -1 -1 -1 -1 21 -1 -1 432 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 31 -1 -1
-1 12 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 432 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 33 -1 43 -1 -1 -1 -1 -1 332 -1 0 -1 213 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 54 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 77 -1 -1 -1 -1
-1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 213 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 44 -1 44 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 543 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 32 -1 -1 -1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 -1 -1 -1 65 -1 -1
-1 -1 -1 65 -1 76 -1 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 22 -1 -1 -1 -1 -1 76 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 47 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 432 -1 -1 -1 -1 -1 21 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 45 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 47 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 123 -1 -1 -1 -1 -1 -1 -1 432 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 75 -1 -1 -1 -1 -1 54
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 76 -1 -1 -1
-1 -1 -1 -1 -1 -1 67 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 213 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 22 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 44 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 78 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 65 -1 -1 -1 -1 -1 -1 -1 -1 87 -1
-1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 44 -1 -1 543 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 76 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 0 -1 -1 -1 95 -1 -1 -1 -1 67 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 47 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 32 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 76 47 -1 -1 -1 -1 -1 -1 -1 -1 76 -1 -1 0 -1 -1 -1 -1 55 -1 -1 -1
-1 32 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 0 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 32 -1 -1 -1 -1 55 -1 -1 54 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 75 -1 -1 -1 -1 -1 -1 95 -1 -1 -1 0 -1 -1 -1 -1 112 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 77 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1
-1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 76 -1 -1 78 -1 -1 -1 -1 55 -1 -1 -1 -1 0 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 31 -1 -1 -1 -1 65 -1 -1 65 -1 -1 45 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 87 -1 67 -1 -1 -1 112 -1 -1 -1 -1 0 -1
-1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 54 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0
```

+ Thời gian chạy

Đồ thị đầu vào	Thông tin Test	Tham lam	FindMinMatch
Đồ thị test 3 (40 đỉnh: có 22 đỉnh bậc lẻ)	Thời gian chạy (giây)	0,901	12,84
	Giá trị Min tìm được	908	715

```
//Load len do thi ban dau
char *tenfile = "C:\\Users\\admin\\Desktop\\Test\\Timduongdi_ThamLam\\Vidu\\40x40.txt";
LoadGraph(tenfile, doThiBanDau, n1);
```

```
C:\Users\admin\Desktop\Test\Timduongdi_ThamLam\NguoiDuaThu.exe
so Canh: 69
Chay hoan tat, de xem ket qua, vui long mo file: 4_chuTrinhthucte_ThamLam.txt

--------------------------------
Process exited after 0.901 seconds with return value 0
Press any key to continue . . .
```

```
//Load len do thi ban dau
char *tenfile = "C:\\Users\\admin\\Desktop\\Test\\Timduongdi_FindMinMatch\\Vidu\\40x40.txt";
LoadGraph(tenfile, doThiBanDau, n1);
```

```
C:\Users\admin\Desktop\Test\Timduongdi_FindMinMatch\NguoiDuaThu.exe
so Canh: 69
Chay hoan tat, de xem ket qua, vui long mo file: 4_chuTrinhthucte_FindMinMatch.t
xt

--------------------------------
Process exited after 12.84 seconds with return value 0
Press any key to continue . . .
```

- **Test 4**: đồ thị gồm 45 đỉnh, có 24 đỉnh bậc lẻ

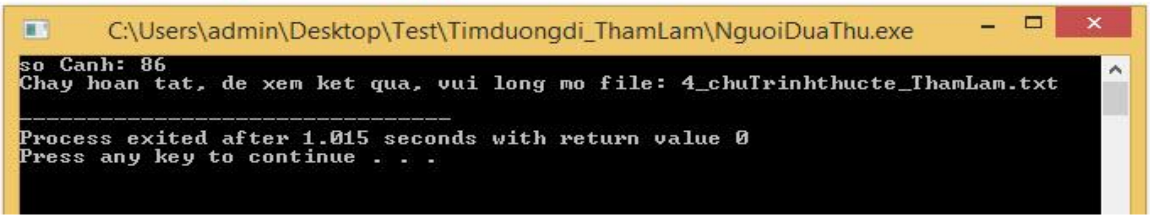
+ Ma trận đầu vào (45x45.txt)

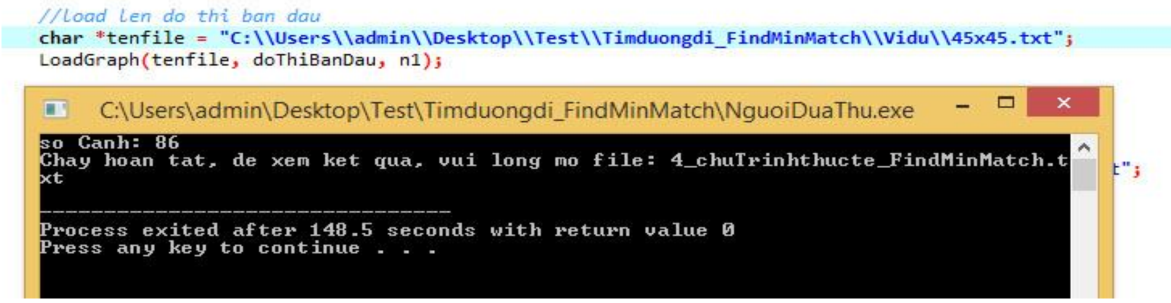
```
45
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45
0 -1 -1 -1 -1 33 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 0 -1 -1 -1 -1 -1 -1 -1 -1 12 33 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32
-1 -1 0 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 123 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 21 -1 -1 -1 -1 -1
-1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 -1 -1 -1 213 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 0 -1 -1 23 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1
33 -1 21 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 76 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 43 -1 -1
-1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 -1 -1 -1 67 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 23 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 55 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 332 -1 -1 -1 -1 -1 21 -1 -1 432 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 31 -1 -1 -1 -1 -1 -1 54
-1 12 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 432 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 33 -1 43 -1 -1 -1 -1 -1 332 -1 0 -1 213 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 54 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 77 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 213 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 44 -1 44 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 55 -1
-1 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 543 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 22
-1 -1 32 -1 -1 -1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 65 -1 76 -1 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 22 -1 -1 -1 -1 -1 76 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 47 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 432 -1 -1 -1 -1 -1 21 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 45 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 47 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 123 -1 -1 -1 -1 -1 -1 -1 432 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 75 -1 -1 -1 -1 -1 54 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 76 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 67 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 755 -1 -1 -1 -1
-1 -1 -1 213 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 22 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 4 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 44 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 78 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 65 -1 -1 -1 -1 -1 -1 -1 -1 87 -1 98 -1 -1 -1 -1
-1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 44 -1 -1 543 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 76 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 331
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 0 -1 -1 -1 95 -1 -1 -1 -1 67 -1 -1 331 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 47 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 32 -1 -1 -1 -1 -1 -1 -1 87 -1 -1 -1 -1
-1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 76 47 -1 -1 -1 -1 -1 -1 -1 -1 76 -1 -1 0 -1 -1 -1 -1 55 -1 -1 -1 -1 -1 -1 -1 -1
-1 32 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 32 -1 -1 -1 -1 55 -1 -1 54 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 75 -1 -1 -1 -1 -1 -1 95 -1 -1 -1 0 -1 -1 -1 -1 112 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 12 321 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 77 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 76 -1 -1 78 -1 -1 -1 -1 55 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 87 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 31 -1 -1 -1 -1 65 -1 -1 65 -1 -1 45 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 87 -1 67 -1 -1 -1 112 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1
-1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 54 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 54 -1 -1
-1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 755 4 -1 98 -1 -1 87 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 331 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1
-1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 12 -1 87 -1 -1 54 -1 -1 0 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 55 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 321 -1 -1 -1 -1 -1 -1 -1 -1 0 -1
-1 32 -1 -1 -1 -1 -1 -1 -1 54 -1 -1 -1 -1 -1 -1 22 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 331 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0
```

\+ Thời gian chạy

Đồ thị đầu vào	Thông tin Test	Tham lam	FindMinMatch
Đồ thị test 4 (45 đỉnh: có 24 đỉnh bậc lẻ)	Thời gian chạy (giây)	1,015	148,5
	Giá trị Min tìm được	1601	1151

```
//load len do thi ban dau
char *tenfile = "C:\\Users\\admin\\Desktop\\Test\\Timduongdi_ThamLam\\Vidu\\45x45.txt";
LoadGraph(tenfile, doThiBanDau, n1);
```





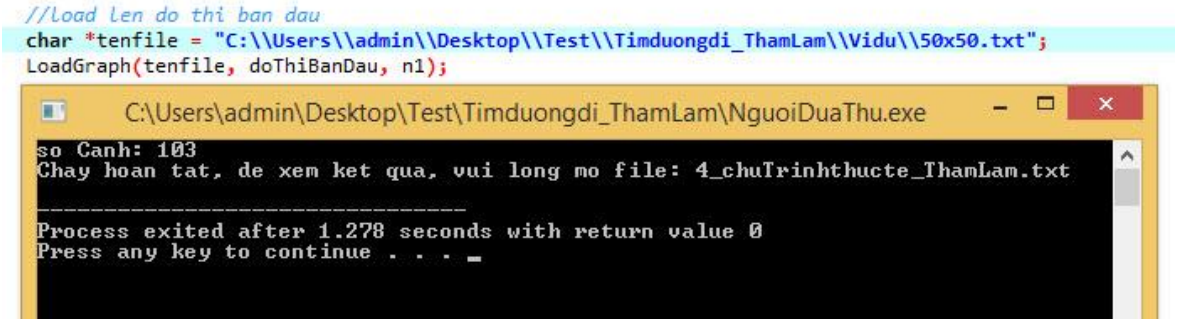
- **Test 5**: đồ thị gồm 50 đỉnh: có 26 đỉnh bậc lẻ

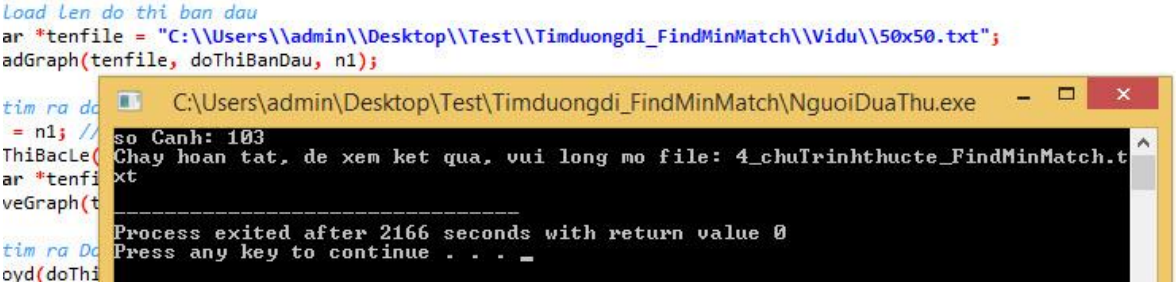
+ Ma trận đầu vào (50x50.txt)

```
50
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
0 -1 -1 -1 -1 33 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 0 -1 -1 -1 -1 -1 -1 -1 -1 12 33 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1
-1 -1 0 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 123 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 324 -1 -1
-1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 -1 -1 -1 213 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1
33 -1 21 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 76 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 -1 -1 -1 67 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 54 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 55 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 332 -1 -1 -1 -1 -1 21 -1 -1 432 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 31 -1 -1 -1 -1 -1 -1 54 -1 -1 -1 -1 -1
-1 12 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 432 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 33 -1 43 -1 -1 -1 -1 -1 332 -1 0 -1 213 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 54 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 77 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 12 -1 -1 -1
-1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 213 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 44 -1 44 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 55 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 543 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 22 -1 -1 -1 -1 -1
-1 -1 32 -1 -1 -1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 65 -1 76 -1 -1 -1 -1 -1 -1 34 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 22 -1 -1 -1 -1 -1 76 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 47 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 34 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 432 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 45 -1 -1 -1 -1 -1 -1 -1 34 -1 56 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 47 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 123 -1 -1 -1 -1 -1 -1 -1 432 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 75 -1 -1 -1 -1 -1 54 -1 -1 -1 -1 -1 21 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 76 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 67 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 755 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 213 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 22 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 4 -1 -1 -1 -1 12 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 44 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 78 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 244 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 65 -1 -1 -1 -1 -1 -1 -1 -1 87 -1 98 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 44 -1 -1 543 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 76 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 331 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 0 -1 -1 -1 95 -1 -1 -1 -1 67 -1 -1 331 -1 -1 -1 -1 32 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 47 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 32 -1 -1 -1 -1 -1 -1 -1 87 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 76 47 -1 -1 -1 -1 -1 -1 -1 -1 76 -1 -1 0 -1 -1 -1 -1 55 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 32 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 123 -1
-1 -1 -1 32 -1 -1 -1 -1 55 -1 -1 54 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 75 -1 -1 -1 -1 -1 -1 95 -1 -1 -1 0 -1 -1 -1 -1 112 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 12 321 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 77 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 76 -1 -1 78 -1 -1 -1 -1 55 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 87 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 31 -1 -1 -1 -1 65 -1 -1 65 -1 -1 45 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 333 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 87 -1 67 -1 -1 -1 112 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 21 -1 -1 -1 -1
-1 -1 21 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 54 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 54 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 755 4 -1 98 -1 -1 87 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 221 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 331 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 43 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 12 -1 87 -1 -1 54 -1 -1 0 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 55 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 321 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 321 -1 -1 65 -1
-1 32 -1 -1 -1 -1 -1 -1 -1 54 -1 -1 -1 -1 -1 -1 22 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 331 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 54 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 34 -1 21 -1 -1 12 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 21 -1 -1 -1 -1 321 -1 0 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 12 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 32 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 54
-1 -1 324 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 34 56 -1 -1 -1 -1 -1 244 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 333 -1 -1 221 -1 -1 -1 -1 -1 -1 0 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 123 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 65 -1 -1 -1 -1 0 6
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 54 -1 6 0
```

+ Thời gian chạy

Đồ thị đầu vào	Thông tin Test	Tham lam	FindMinMatch
Đồ thị test 5 (50 đỉnh: có 26 đỉnh bậc lẻ)	Thời gian chạy (giây)	1,278	2166
	Giá trị Min tìm được	1612	1151





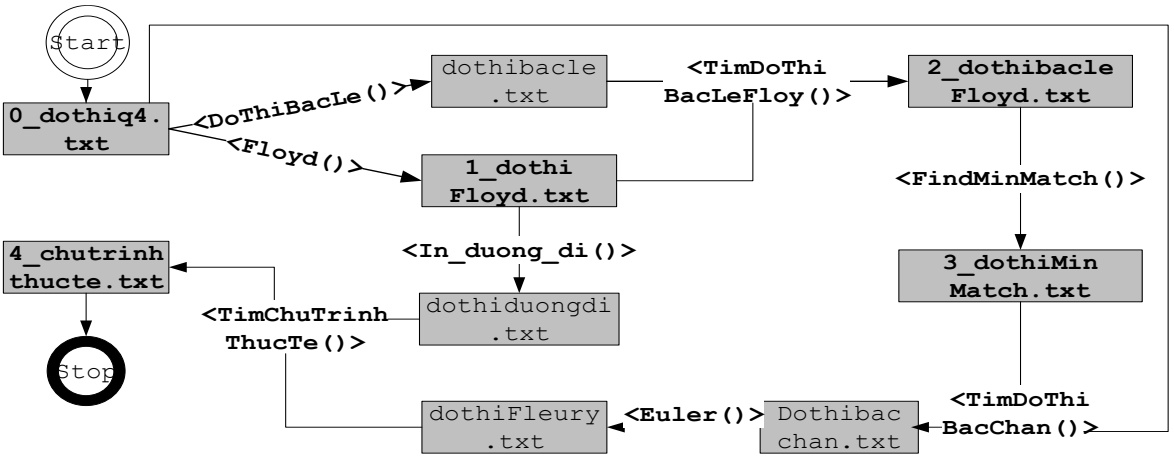
2. Code Chương trình Chính mô phỏng thuật toán bằng ngôn ngữ C++

```
#include<iostream>
#include<stdlib.h>
#include<fstream>
#define MAX 100
#define TRUE 1
const int lim = 32767; //max de so sanh
using namespace std;

typedef int Dothi[MAX][MAX];
typedef int mangKq[3][MAX];
#define TRUE 1
#define FALSE 0

int main()
{ Dothi doThiBanDau, doThiBacLe, doThiFloyd, doThiBacLeFloyd,
doThiDuongDi, doThiBacChan, doThiFluery, minMatch;
```

```
     mangKq mangCapGhep, mangCapGhepCopy;
     int n1, n2, n;
     Init(doThiBanDau, MAX);
     Init(doThiBacLe, MAX);
     Init(doThiFloyd, MAX);
     Init(doThiDuongDi, MAX);
     Init(doThiBacChan, MAX);
     Init(doThiFluery, MAX);
     Init(minMatch, MAX);
     int kq[MAX];
     int canh[MAX];
     int min = lim;
     for (int j = 0; j < MAX; j++)
          canh[j] = 0;

//load len do thi ban dau
     char *tenfile = "\\0_dothiq4.txt";
     LoadGraph(tenfile, doThiBanDau, n1);
//tim ra do thi bac le
     n2 = n1; //Sau nay n2<n1; n2 la so dinh co bac le
     DoThiBacLe(doThiBanDau, doThiBacLe, n2);
     char *tenfile2 = "\\dothibacle.txt";
     SaveGraph(tenfile2, doThiBacLe, n2);
//tim ra DoThiFloyd va DoThiDuongDi de tim duong di sau nay
     Floyd(doThiBanDau, doThiFloyd, doThiDuongDi, n1);
     tenfile2 = "\\1_dothiFloyd.txt";
     SaveGraph(tenfile2, doThiFloyd, n1);
     in_duong_di(doThiFloyd, doThiDuongDi, n1);
//khoi tao doThiBacLeFloyd
     CopyDoThi(doThiBacLe, doThiBacLeFloyd, n2);
//them gia tri cho doThiBacLeFloyd trong DoThiFloyd
     TimDoThiBacLeFloyd(doThiBacLeFloyd, doThiFloyd, n2);
     tenfile2 = "\\2_dothibacleFloyd.txt";
     SaveGraph(tenfile2, doThiBacLeFloyd, n2);
//tim cap ghep bang tham lam: kq trong mangCapGhep2
     mangKq mangCapGhep2;
     int t = ThamLam(doThiBacLeFloyd, mangCapGhep2, n2); // t
la gtri min cua tong cac cap ghep = tham lam
     XuatMangCapGhep(mangCapGhep2, n2, t);
     //tim cap ghep bang min match: kq trong mangcapghep
     //cap ghep trong mang kq(mang 1 chieu, luu dinh cap)
     FindMinMatch(doThiBacLeFloyd, n2, kq, canh, min, 0, 0,
0);  // neu muon cai thien thuat toan minmatch thi co the thay
min = t ben tren
     XuatMang(doThiBacLeFloyd, kq, n2, min, mangCapGhep);
//chuyen mang kq thanh mangCapghep, dong thoi ghi ket qua xuong
file
```

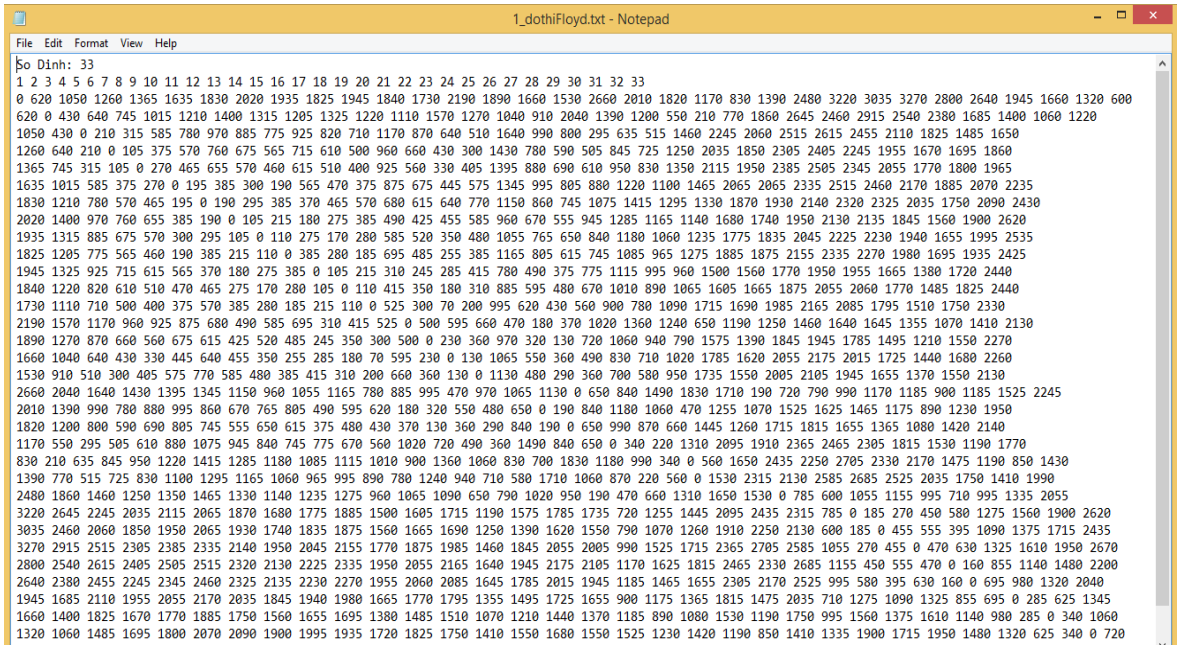
```
//tao ra doThiBacChan = do thi ban dau + mangCapGhep  // neu muon dung tham lam thay cho min match thi thay mangCapGhep = mangCapGhep2 , thay luon cho ham TimChuTrinhThucTe()
     CopyMangCapGhep(mangCapGhep,    mangCapGhepCopy,    n1);
//lay mangCapGhepCopy de su dung, luu lai mangCapGhep
     TimDoThiBacChan(doThiBacChan, doThiBanDau, mangCapGhep-
Copy, n1);
     tenfile2 = "\\dothibacchan.txt";
     SaveGraph(tenfile2, doThiBacChan, n1);
//tim chu trinh
     CopyDoThi(doThiBacChan, doThiFluery, n1);
     tenfile2 = "\\doThiFluery.txt";
     SaveGraph(tenfile2, doThiFluery, n1);
     DemCanh(doThiBacChan, n1);

     Euler(doThiBacChan, doThiFluery, n1);
     cout << "\nChay hoan tat, de xem ket qua, vui long mo
file: 4_chuTrinhthucte.txt";
//chu trinh thuc te: ket hop chu trinh, do thi duong di, mang cap ghep
//kiem tra chu trinh, duong nao co trong mang cap ghep -> ghi lai chu trinh ve them tu mang duong di(cua thuat toan Floyd)
     TimChuTrinhThucTe(mangCapGhep, doThiDuongDi, n1);
     XuatChuTrinhThucTe(n1);
     tenfile2 = "\\doThiDuongdiFloyd.txt";
     SaveGraph(tenfile2, doThiDuongDi, n1);
getchar();
return 0;
}
```

3. Mô hình quy trình xử lý của thuật toán

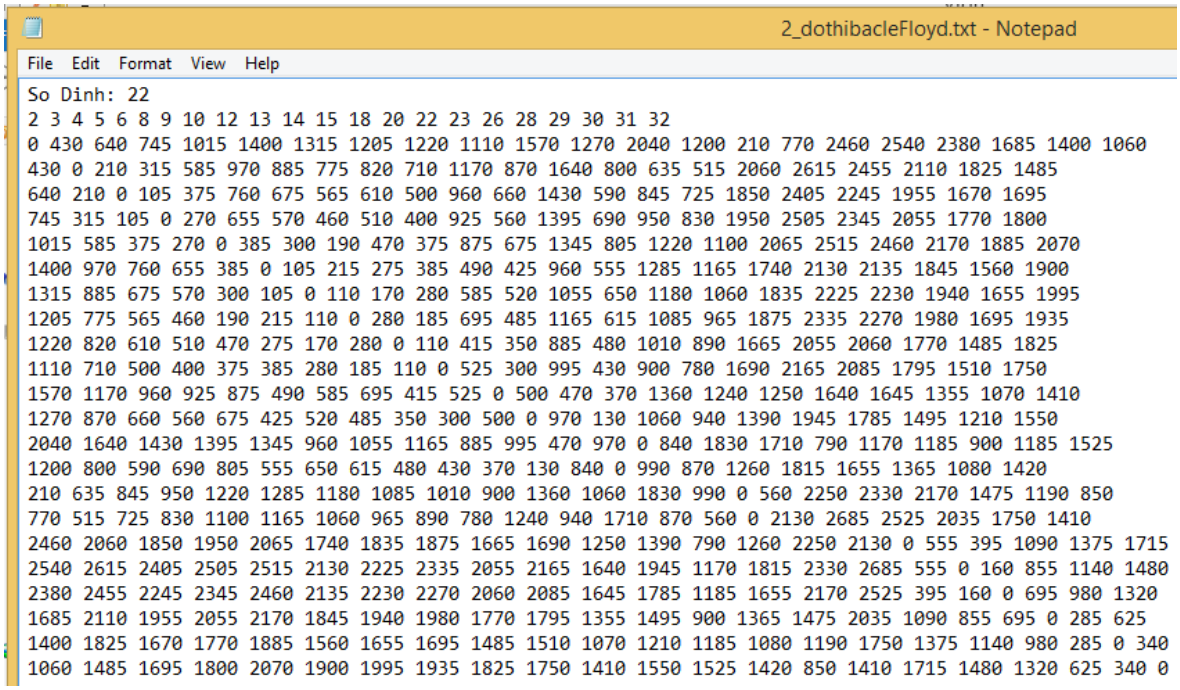


4. Kết quả file xuất ra sau khi chạy chương trình

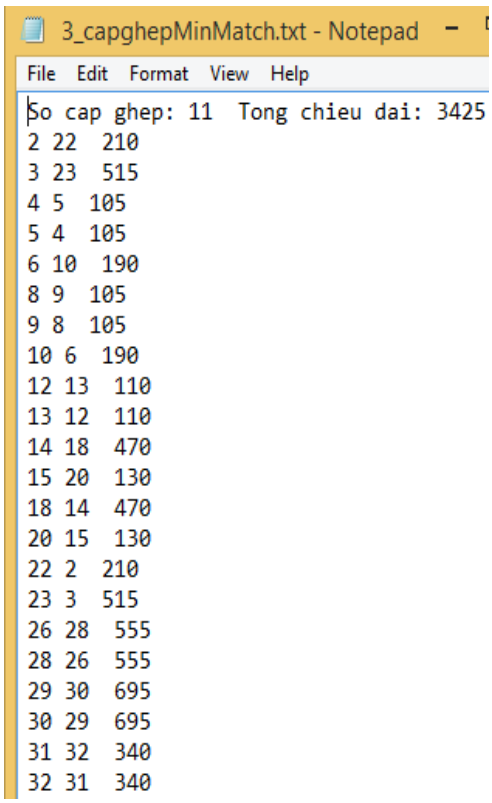
- File 1: 1_dothiFloyd.txt

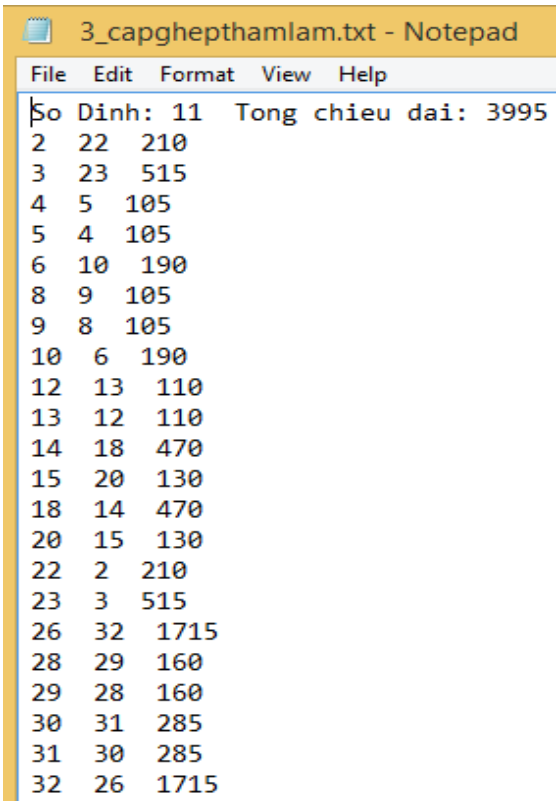


- File 2: 2_dothibacleFloyd.txt

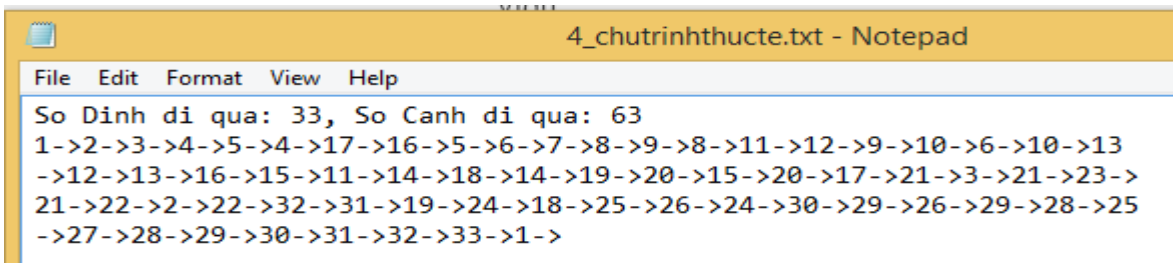


- File 3: 3_capghepMinMatch.txt- File 4: 3_capghepthamlam.txt

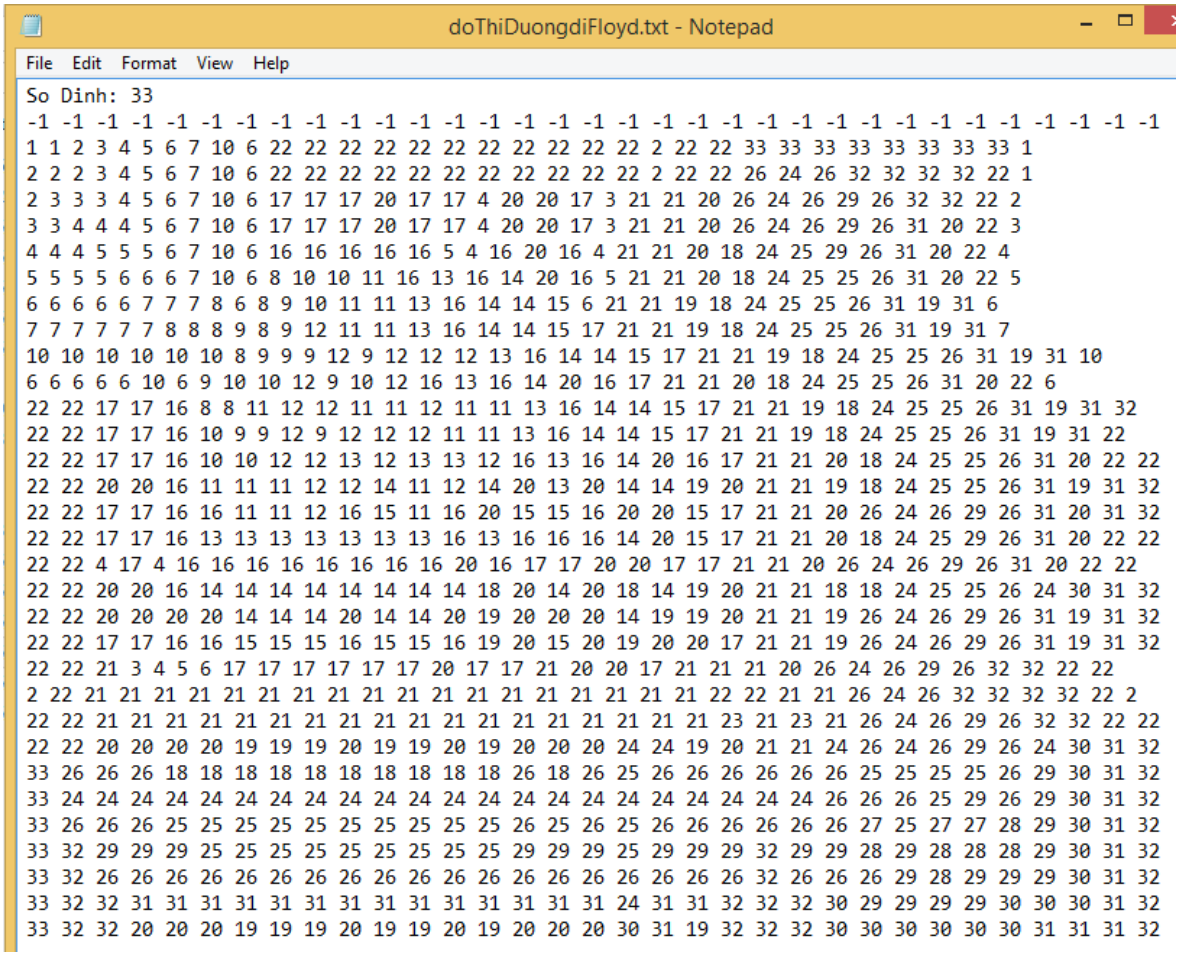




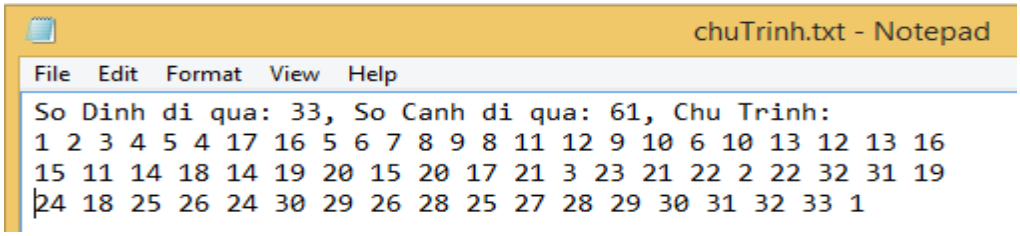
- File 5: 4_chutrinhthucte.txt



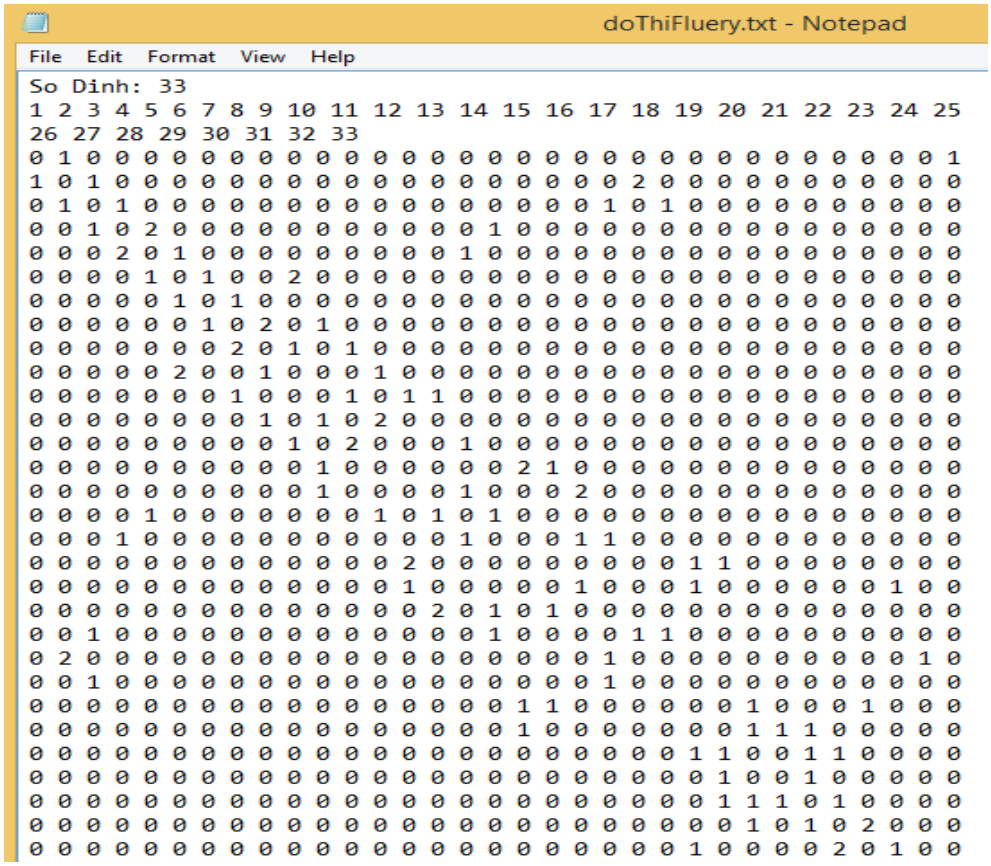
- File 6: doThiDuongdiFloyd.txt



- File 7: chutrinh.txt



- File 8: doThiFluery.txt

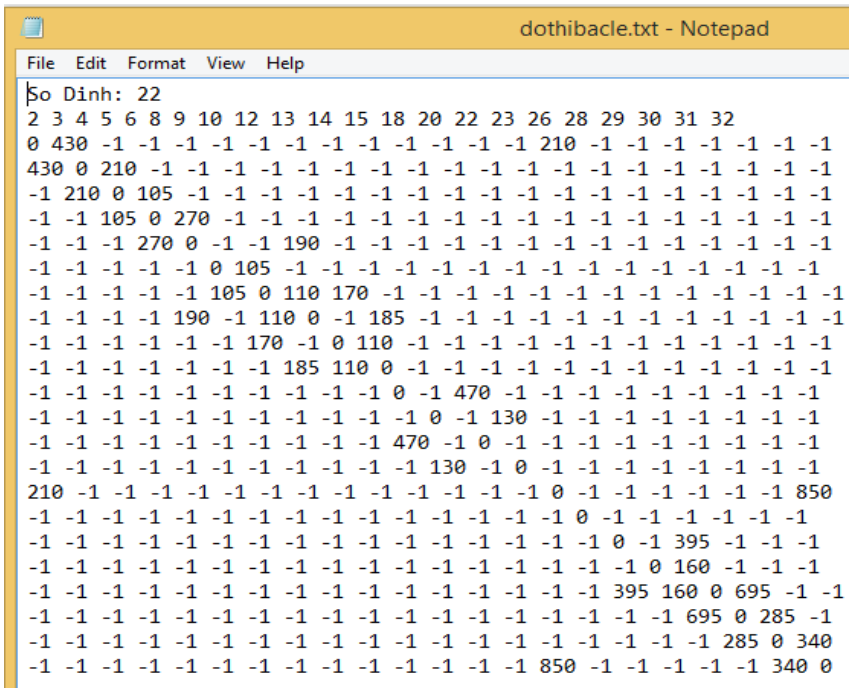


doThiFluery.txt - Notepad

File Edit Format View Help

```
So Dinh: 33
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 33
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0
0 0 1 0 2 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 2 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 2 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 2 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 2 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 2 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 2 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 2 0 1 0 0
```

- File 9:dothibacle.txt



dothibacle.txt - Notepad

File Edit Format View Help

```
So Dinh: 22
2 3 4 5 6 8 9 10 12 13 14 15 18 20 22 23 26 28 29 30 31 32
0 430 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 210 -1 -1 -1 -1 -1 -1 -1
430 0 210 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 210 0 105 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 105 0 270 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 270 0 -1 -1 190 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 0 105 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 105 0 110 170 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 190 -1 110 0 -1 185 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 170 -1 0 110 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 185 110 0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 470 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 130 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 470 -1 0 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 130 -1 0 -1 -1 -1 -1 -1 -1 -1 -1
210 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1 850
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 -1 395 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 160 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 395 160 0 695 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 695 0 285 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 285 0 340
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 850 -1 -1 -1 -1 -1 340 0
```