BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NÔNG LÂM TP.HỒ CHÍ MINH
KHOA MÔI TRƯỜNG VÀ TÀI NGUYÊN

TIỂU LUẬN TỐT NGHIỆP

ỨNG DỤNG GIS HỖ TRỢ BÀI TOÁN TƯỚI CÂY TRÊN ĐƯỜNG TẠI QUẬN THỦ ĐỨC

Họ và tên: Nguyễn Thị Bảo Xuyên
Ngành: Hệ thống thông tin địa lý
Niên khóa: 2012 - 2016

TP. Hồ Chí Minh, tháng 06 năm 2016

LỜI CẢM ƠN

Trong quá trình làm tiểu luận tốt nghiệp tôi đã nhận được sự giúp đỡ, chỉ bảo tận tình của các cán bộ tại Trung tâm Ứng dụng Hệ thống thông tin Địa lý (Trung tâm HCMGIS) Sở Khoa học và công nghệ TP.HCM và quý thầy cô tại Bộ môn Tài nguyên và GIS – Trường ĐH Nông Lâm TP.HCM và toàn thể các thành viên của lớp DH12GI để tôi có thể hoàn thành tốt nhiệm vụ của mình.

Qua đây, tôi xin gửi lời cám ơn chân thành đến:

- Th.S Khưu Minh Cảnh, công tác tại Trung tâm Ứng dụng Hệ thống thông tin Địa lý – Sở Khoa học và Công nghệ TP.HCM, người đã trực tiếp hướng dẫn, chỉ bảo tận tình, góp ý cho tôi trong suốt qúa trình thực hiện đề tài.
- Quý thầy (cô) Bộ môn Tài nguyên và GIS – Trường ĐH Nông Lâm TP.HCM đặc biệt là thầy PGS.TS. Nguyễn Kim Lợi, Th.S Nguyễn Thị Huyền, đã tận tình giảng dạy và truyền đạt nhiều kiến thức quý báu trong suốt thời gian tôi học tập tại trường.
- Thầy Lê Văn Phận, KS. Nguyễn Duy Liêm, các anh chị tại Bộ môn Tài nguyên và GIS – Trường ĐH Nông Lâm TP.HCM, đã tận tình chỉ bảo, góp ý trong thời gian học tập và thực hiện đề tài.
- Tập thể cán bộ viên chức tại Trung tâm Ứng dụng Hệ thống thông tin Địa lý – Sở Khoa học và Công nghệ TP.HCM, đã tận tình giúp đỡ và tạo điều kiện trong quá trình thực hiện đề tài tiểu luận này.
- Cuối cùng, em gửi lời cảm ơn sâu sắc đến Gia đình và bạn bè luôn động viên giúp đỡ về mọi mặt và tạo điều kiện thuận lợi trong quá trình học tập cũng như trong thời gian làm đề tài.

TP.HCM, tháng 06/2016

Nguyễn Thị Bảo Xuyên

Bộ môn Tài nguyên và GIS

Khoa Môi trường và Tài nguyên

Trường ĐH Nông Lâm TP.HCM

Email:baoxuyennt@gmail.com

TÓM TẮT

Nước có vai trò vô cùng quan trọng đối với con người cũng như bất cứ sinh vật nào trên trái đất. Nước là nguồn tài nguyên vô cùng quý giá nhưng không phải là vô tận. Nước cần cho mọi sự sống và phát triển, nước vừa là môi trường vừa là đầu vào cho các quá sinh sản xuất nông nghiệp, công nghiệp.

Nói cụ thể hơn ở đây là mỗi loại cây trên đường phố thì có nhu cầu về lượng nước khác nhau. Giả sử với lượng nước nhất định thì làm thế nào để giải quyết bài toán tưới cây một cách hiệu quả và tiết kiệm nước nhất có thể mà cây vẫn có thể duy trì sự sống và chi phí đường đi là tối thiểu. Chính vì thế đề tài nghiên cứu "*Ứng dụng GIS hỗ trợ bài toán tưới cây trên đường tại quận Thủ Đức*" đã được thực hiện và hoàn thành tại Trung tâm Ứng dụng Hệ thống thông tin Địa lý (Trung tâm HCMGIS) Sở khoa học và Công nghệ TP.HCM, trong khoảng thời gian từ tháng 03/2016 đến 05/2016.

Kết quả đề tài đạt được của tiểu luận là lập được lịch trình tưới cây và bên cạnh đó hiểu thêm các phương pháp được sử dụng trình bày trong đề tài.

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

CSDL	Cơ sở dữ liệu
GIS (Geographic Information system)	Hệ thống thông tin Địa lý
GDB (Geodatabase)	Cơ sở dữ liệu địa lý
TP.HCM	Thành phố Hồ Chí Minh
HTTTĐL	Hệ thống thông tin địa lý
DBMS (Database Management System)	Hệ quản trị cơ sở dữ liệu
HĐH	Hệ điều hành
P-center	P-trung tâm
Point	Điểm

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH

CHƯƠNG I. MỞ ĐẦU

1.1. Tính cấp thiết của đề tài

Như chúng ta đã biết từ xa xưa đến nay, cuộc sống của loài người luôn gắn bó và không thể tách rời khỏi thiên nhiên. Thế nên, bất kể ai khi đứng trước các yếu tố tạo nên thiên nhiên (nước, cỏ cây hoa lá, núi non…) đều cảm thấy tâm hồn nhẹ nhàng và thư thái, như tìm được chốn yên bình sau khoảng thời gian ồn ào vội vã của cuộc sống. Một trong những tác dụng lớn nhất của cây xanh cho đô thị, đó là nó cải thiện rõ rệt môi trường sống của người dân. Với mật độ dân cư đông, cùng với lượng khí thải từ nhà máy, xe cộ,… tình trạng chung của các khu đô thị chính là môi trường không khí bị ô nhiễm nghiêm trọng. Cây xanh sẽ giúp cải thiện chất lượng không khí bằng cách hấp thu những khí độc như NO_2, CO_2, CO…Theo nhiều nghiên cứu, cây xanh có thể hấp thụ tới 6% các loại khí thải độc. Cây xanh sẽ giúp lọc bớt bụi bẩn, đồng thời thải ra nhiều O_2. Vì vậy có thể xem cây xanh là lá phổi của thành phố. [13]

Trong những năm gần đây do nguồn nước ngày càng trở nên khan hiếm. Theo thạc sĩ Lê Thị Xuân Lan, nguyên Phó Phòng Dự báo Khí tượng Thủy văn Khu vực Nam Bộ, cảnh báo hiện nay tình hình khô hạn đang diễn ra trên diện rộng khắp cả nước, trong đó ở Nam Bộ là rõ rệt nhất. Nguyên nhân do mùa mưa trong năm 2015 thiếu hụt từ 30%-40%, mưa trễ nhưng lại kết thúc quá sớm. Hiện khu vực Nam Bộ, trong đó có TP HCM, đang bước vào thời điểm khô hạn nhất. Dù trong thời gian tới vẫn có mưa trái mùa nhưng xảy ra ít nên không đủ bù cho sự thiếu hụt nói trên. Sự khô hạn này vẫn còn tiếp tục diễn ra vào thời kỳ cao điểm ở các tháng tiếp theo cho đến hết tháng 4/2016. [14]

Bên cạnh đó giá xăng giá dầu ngày một tăng cao. Để chăm sóc "lá phổi" của quận nói riêng và toàn thành phố nói chung và cũng như để duy trì sự sống cho cây xanh. Đặt ra bài toán: Giả định lượng nước tưới không phải vô tận. Và một tuyến đường trồng một loại cây. Một số loại cây có thể 2-3 ngày tưới 1 lần. Một số cây mỗi ngày phải tưới. Lập lịch tưới cây tốt nhất có thể. Tìm phương án tưới sao cho cây vẫn sống mà lượng nước tưới ít nhất mà lược di chuyển trên đường là ít nhất có thể.

Từ những vấn đề cấp thiết trên đề tài "***Ứng dụng GIS hỗ trợ bài toán tưới cây trên đường tại quận Thủ Đức***" được thực hiện nhằm giải quyết bài toán đã đề cập ở trên.

1.2. Mục tiêu của đề tài

❖ **Mục tiêu chung:**

Ứng dụng GIS hỗ trợ bài toán tưới cây trên đường tại quận Thủ Đức.

❖ **Mục tiêu cụ thể:**

- Tưới cây sao cho cây vẫn sống mà lượng nước tưới ít nhất và lược di chuyển trên đường cũng ít nhất có thể.
- Hiểu được một số lý thuyết/ thuật toán; chu trình Halmilton, chu trình Euler; phép phân tích P-center; bài toán đồ thị và bài toán lập lịch.
- Lập lịch tưới cây.

1.3. Giới hạn phạm vi nghiên cứu

- Không gian: Phạm vi nghiên cứu đề tài được thực hiện tại quận Thủ Đức.
- Nội dung: Ứng dụng GIS để hỗ trợ bài toán tưới cây trên đường.
- Công nghệ: Sử dụng phần mềm GIS, ngôn ngữ lập trình Python.

CHƯƠNG 2. TỔNG QUAN KHU VỰC NGHIÊN CỨU

2.1. Tổng quan địa bàn nghiên cứu

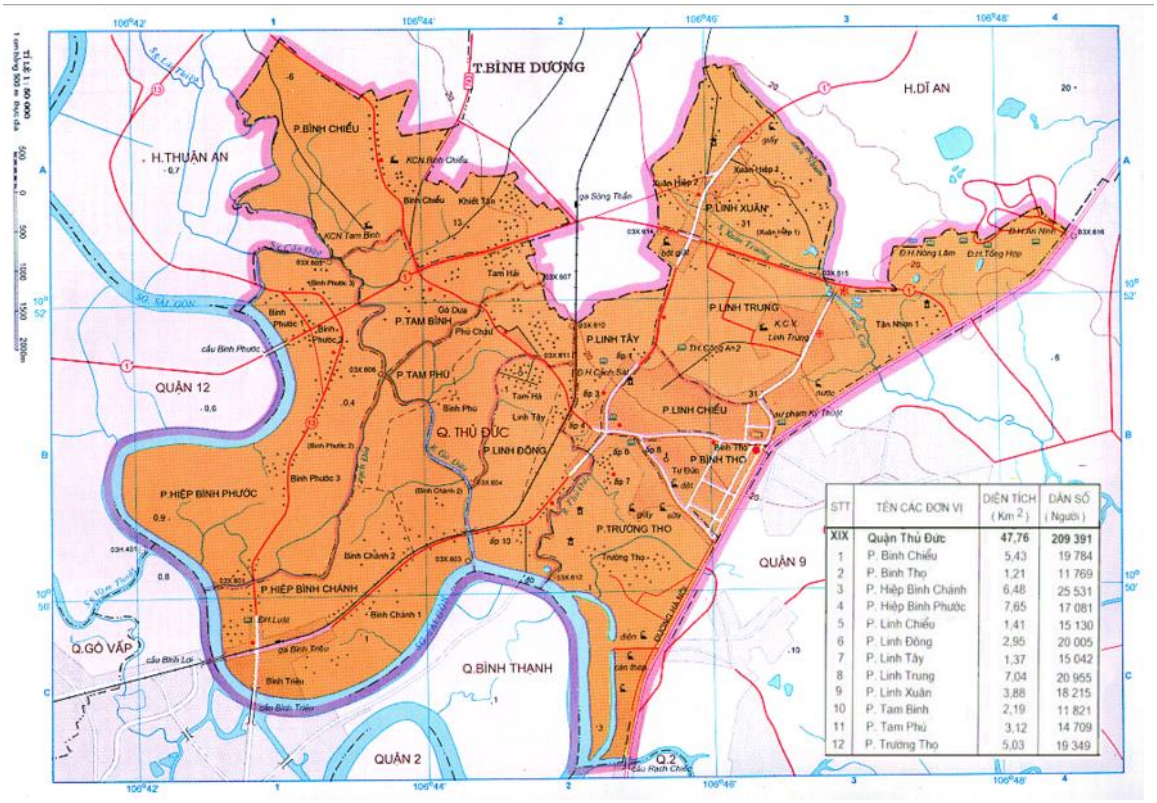
2.1.1. Điều kiện tự nhiên

❖ **Vị trí địa lý**

Thủ Đức sau ngày 30-04-1975 là huyện ngoại thành, nằm ở phía Đông – Bắc thành phố Hồ Chí Minh. Năm 1997, huyện Thủ Đức được phân chia thành 3 quận: quận 2, quận 9 và quận Thủ Đức theo nghị định 03/CP của Chính Phủ ban hành ngày 6-1-1997. Quận Thủ Đức mới có diện tích 47,76 km^2, bao gồm diện tích và dân số của các xã Linh Đông, Linh Trung, Tam Bình, Tam Phú, Hiệp Bình Phước, Hiệp Bình Chánh, thị trấn Thủ Đức, một phần diện tích và nhân khẩu của các xã Hiệp Phú, Tân Phú và Phước Long. Sau khi trở thành quận, các xã đều đổi tên thành phường. Quận Thủ Đức có 12 phường gọi tên theo xã trước đây: Linh Đông, Linh Tây, Linh Chiểu, Linh Trung, Linh Xuân, Hiệp Bình Chánh, Hiệp Bình Phước, Tam Phú, Trường Thọ, Bình Chiểu, Bình Thọ, Tam Bình; dân số tính đến nay là khoảng 250.000 người. [15]

Vốn là một huyện ngoại thành, Thủ Đức không có nhiều công trình hạ tầng kỹ thuật cũng như hạ tầng xã hội. Ba con đường lớn chạy qua huyện Thủ Đức trước kia và quận Thủ Đức ngày nay đều thuộc quốc lộ: xa lộ Hà Nội, quốc lộ 13 và xa lộ vành đai ngoài (là xa lộ Đại Hàn cũ). Bao bọc 3 mặt Thủ Đức là hai con sông lớn, sông Đồng Nai và sông Sài Gòn, rất thuận lợi cho giao thông đường thủy, phục vụ vận chuyển hàng hóa nông sản và thực phẩm của các công ty lớn trên địa bàn. [2]

- Phía Bắc quận Thủ Đức tiếp giáp với huyện Dĩ An – tỉnh Bình Dương.
- Phía Nam giáp sông Sài Gòn, quận 2, quận 9, quận Bình Thạnh.
- Phía Đông giáp quận 9.
- Phía Tây giáp quận 2.



Hình 2.1. Bản đồ quận Thủ Đức

(*Cập nhập 10/06/2007 tại trang: diaoconline.vn*)

❖ **Giao thông**

Không nằm trong trung tâm thành phố nhưng với vị trí chiến lược quan trọng, các tuyế đường giao thông tại Thủ Đức luôn có mật độ giao thông tại quận Thủ Đức luôn có mật độ giao thông cao như Quốc lộ, xa lộ Hà Nội, Quốc lộ 13, Quốc lộ 1K, Võ Văn Ngân, Kha Vạn Cân, Đặng Văn Bi,...

2.1.2. Điều kiện kinh tế xã hội

❖ **Dân số**

Cũng như TP.Hồ Chí Minh, quận Thủ Đức là nơi tập trung một lượng dân cư lớn. Tính đến năm 2009, dân số ở khoảng 43.000 ngàn người. Nguyên nhân dẫn đến việc dân số đông là do sự phát triển của các khu công nghiệp, khu chế xuất (Linh Trung, Linh Xuân,…). Sự phát triển của các trường học đặc biệt là làng ĐH Thủ Đức. Ngoài ra, còn do sự di cư của các quận nội thành chật chội ra vùng ven.

Việc tăng dân số góp phần cung cấp nguồn lao động cho các khu công nghiệp nhưng cũng mang đến nhiều bất cập cho công tác quản lý, y tế, văn hóa, giáo dục, an ninh trật tự.

❖ **Kinh tế**

Do nằm ở ngoại ô nền kinh tế quận Thủ Đức đa dạng với nhiều loại hình kinh tế như sản xuất nông nghiệp, sản xuất công nghiệp, thương mại, dịch vụ.

❖ **Sản xuất nông nghiệp**

Cũng như các huyện ngoại thành khác, Thủ Đức trước ngày giải phóng là "vành đai trắng", là "vùng tự do bắn phá" của Mỹ ngụy, vì thế quá trình khôi phục sản xuất nông nghiệp gặp vô và khó khăn, thậm chí phải chịu hi sinh khi rà phá bom mìn để biến vùng đất hoang hóa trở thành những cánh đồng lúa xanh ngút tầm mắt (chỉ trong 3 năm 1976-1978, Thủ Đức đã khôi phục khoảng 11.000 ha trong 14.000 ha của "vành đai trắng"). Hệ thống thủy lợi nội đồng được xây dựng ngay trong những năm đầu sau ngày 30-4-1975, vừa giải quyết tình trạng nập úng, vừa tăng năng suất các loại cây trồng, đưa cây lúa vào canh tác 2 đến 3 vụ/năm.

❖ **Công nghiệp – Tiểu thủ công nghiệp**

Dù mang tên là huyện, nhưng Thủ Đức lại là vùng đất làm "cầu nối " giữa thành phố Hồ Chí Minh và các tỉnh miền Đông Nam Bộ giàu tiềm năng công nghiệp, do đó ngay trên địa bàn Thủ Đức, dưới chế độ cũ đã hình thành một số cụm công nghiệp và hàng chục nhà máy nằm rải rác trong các khu dân cư. Công ty xi măng Hà Tiên, Công ty Cơ điện, Nhà máy điện có mặt từ rất sớm ở Thủ Đức, là ba trong số hơn 100 nhà máy có quy mô khá do tư bản nước ngoài và tư bản Hoa kiều làm chủ. Cuối năm 1974 và đặc biệt là đầu năm 1975, trước nguy cơ sụp đổ không tránh khỏi của ngụy quyền, giới chủ tư bản công nghiệp đã tháo gỡ máy móc "tùy nghi di tản", gây nên sư xáo trộn rất lớn trong xã hội 25.000 công nhân thất nghiệp sau ngày 30-4-1975 là hậu quả của hành động phá hoại sản xuất ấy.

Sự khôi phục và phát triển nhanh của công nghiệp – tiểu thủ công nghiệp trong những năm đổi mới đã làm tăng tỷ trọng trong tổng giá trị kinh tế của Quận lên 62% - một trong những Quận có tỷ trọng công nghiệp cao nhất thành phố Hồ Chí Minh. Công nghiệp phát triển đã tạo điều kiện củng cố và phát triển giai cấp công nhân. Năm 1997, trước khi tách quận, công nhân công nghiệp Thủ Đức hơn 42.000 người (số liệu thống kê 1996). Thủ Đức có thêm hàng ngàn công nhân xây dựng khi cả ba quận trên địa bàn huyện Thủ Đức trở thành những công trường xây dựng lớn, tấp nập suốt ngày đêm.

Thủ Đức cũng là địa phương hấp dẫn các nhà đầu tư. Khu chế xuất Linh Trung đã được lấp kín; Khu công nghiệp Bình Chiểu cũng được các nhà đầu tư thuê hết mặt bằng xây dựng nhà máy sản xuất. Thủ Đức đang xây dựng khu xuất xuất Linh Trung 2 cho các nhà đầu tư có nhu cầu làm ăn lâu dài trên vùng đất này.

Giá trị tổng sản lượng công nghiệp – tiểu thủ công nghiệp của quận Thủ Đức tăng trưởng nhanh, đặc biệt từ năm tách quận. Năm 1995 giá trị sản lượng của ngành công nghiệp huyện Thủ Đức (bao gồm 3 quận Thủ Đức, quận 2 và quận 9) lá 118 tỉ đồng, đến năm 1997, riêng quận Thủ Đức đã là 248 tỉ đồng. Trong các năm tiếp theo, đặc biệt là từ năm 2000, tỉ lệ tăng trưởng giá trị sản lượng đạt bình quân hơm 50% / năm. Năm 2000 là 529,4 tỉ, năm 2002 là 902,7 tỉ, năm 2003 là 1.119,6 tỉ và năm 2004 đạt 1.444,12 tỉ đồng.

Là địa phương có nền sản xuất công nghiệp – tiểu thủ công nghiệp lâu năm, từ năm 1991 đến nay, nhiều mặt hàng truyền thống của Thủ Đức nhanh chóng có chỗ đứng trong nước là tại thị trường nhiều nước.

❖ **Thương mại – Dịch vụ**

Ngành thương mại Thủ Đức phát triển rất sớm. Ba mươi năm qua, chợ Thủ Đức tuy không lớn – vẫn là trung tâm mua ban tấp nập, có sức hấp dẫn khách hàng trong và ngoài quận.

Cũng như vùng chợ Lớn, Thủ Đức là nơi có một số người Hoa chuyên nghề kinh doanh. Theo một thống kê, trước ngày 30-4-1975 số cơ sở buôn bán, dịch vụ ẩm thực và sạp chợ của giới thương nhân người Hoa trên địa bàn Thủ Đức chiếm khoảng 50%.

Thập niên 90 đánh dấu sự phát triển nhanh và bền vững của hoạt động thương mại trên địa bàn quận Thủ Đức, tốc độ tăng bình quân 30% / năm. Kinh doanh nhà hàng – khách sạn, nhà và biệt thự cho thuê, dịch vụ văn phòng cũng phát triển dù Thủ Đức là vùng ngoại thành. Một hình thức dịch vụ mới đang được triển khai có kết quả là xây và cho thuê dạng nhà phố, biệt thự cạnh các khu vui chơi giải trí và sinh hoạt thể thao.

Trên địa bàn Thủ Đức, ngoài chợ Thủ Đức ở trung tâm thị trấn, còn có 15 "chợ quê" với hơn 5.500 hộ buôn bán, điều đó đã nói lên phần nào quy mô hoạt động thương nghiệp tại đây. Trong quy hoạch chợ của thành phố, quận Thủ Đức đã có chợ đầu mối Tam Bình thay cho chợ đầu mối Cầu Muối – thuộc quận 1.

Một hoạt động có hiệu quả của Thủ Đức là ngoại thương, tăng trưởng đạt bình quân 14% / năm, vừa bảo đảm sản phẩm công – nông nghiệp của quận tham gia thị trường xuất khẩu, vừa thu ngoại tệ để nhập máy móc, nguyên phụ liệu phục vụ sản xuất và nhu yếu phẩm cho thị trường nội địa.

Doanh thu thương mại-dịch vụ: năm 1991 đạt 310 tỉ, năm 1995 đạt 920 tỉ, năm 1997 (tách quận – không tính quận 2 và quận 9) đạt 753 tỉ, năm 2000 đạt 928 tỉ, năm 2001 đạt 1.188 tỉ, năm 2003 đạt 1.746 tỉ và năm 2004 đạt 2.252 tỉ đồng. [15]

CHƯƠNG 3. CƠ SỞ LÝ THUYẾT

3.1. Tổng quan về Hệ thống thông tin Địa lý (GIS)

3.1.1. Lịch sử phát triển

GIS được hình thành từ các ngành khoa học: Địa lý, Bản đồ, Tin hoc và Toán học. Nguồn gốc của GIS là việc tạo các bản đồ chuyên đề, các nhà quy hoạch sử dụng phương pháp chồng lắp bản đồ (Overlay). Việc sử dụng máy tính trong vẽ bản đồ được bắt đầu vào cuối thập niên 50, đầu thập niên 60, từ đây thì khái niệm GIS ra đời nhưng chỉ đến những năm 80 thì GIS mới thực sự phát huy hết khả năng của mình do sự phát triển mạnh mẽ của công nghệ phần cứng và cũng từ thập niên 80 này mà GIS trở nên phổ biến trong các lĩnh vực thương mại, khoa học và quản lý.

Hệ thống thông tin địa lý (GIS) được du nhập vào Việt Nam trong những năm của thập niên 80 thông qua các dự án trong khuôn khổ hợp tác quốc tế. Tuy nhiên, chỉ đến những năm cuối của thập niên 90, GIS mới được nhiều người biết đến và áp dụng rộng rãi tại Việt nam trong các lĩnh vực: Quản lý tài nguyên thiên nhiên, giám sát môi trường, quy hoạch thiết kế cảnh quan đô thị, quản lý cây xanh đô thị,… Hiện nay, nhiều cơ quan nghiên cứu và doanh nghiệp đã và đang ứng dụng công nghệ GIS để giải quyết các bài toán của cơ quan mình.

GIS được sử dụng nhằm xử lý đồng bộ các lớp thông tin không gian (bản đồ) gắn với các thông tin thuộc tín, phục vụ nghiên cứu, quy hoạch và quản lý các hoạt động theo lãnh thổ.

Ngày nay, ở nhiều quốc gia trên thế giới, GIS trở thành công cụ trợ giúp quyết định trong hầu hết các hoạt động kinh tế - xã hội, an ninh, quốc phòng, đối với thảm họa thiên tai v.v… GIS có khả năng trợ giúp cơ quan chính phủ, các nhà quản lý, các doanh nghiệp, các cá nhân v.v… đánh giá được hiện trạng của các quá trình, các thực thể và tích hợp các thông tin được gắn với một bản đồ số nhất quán trên cơ sở tọa độ của các dữ liệu bản đồ đầu vào.

3.1.2. Định nghĩa

GIS được sử dụng trong nhiều lĩnh vực khác nhau như địa lý, kỹ thuật, tin học, tài nguyên môi trường, khoa học xử lý về dữ liệu không gian,.. Sự đa dạng trong các lĩnh vực ứng dụng dẫn đến có rất nhiều định nghĩa về GIS (*Nguyễn Thị Kim Nga 2013*).

Theo Ducker (1979), GIS là một trường hợp đặc biệt của hệ thống thông tin, ở đó cơ sở dữ liệu bao gồm sự quan sát các đặc trưng phân bố không gian, các hoạt động sự kiện có thể được xác định trong khoảng không như đường, điểm, vùng .

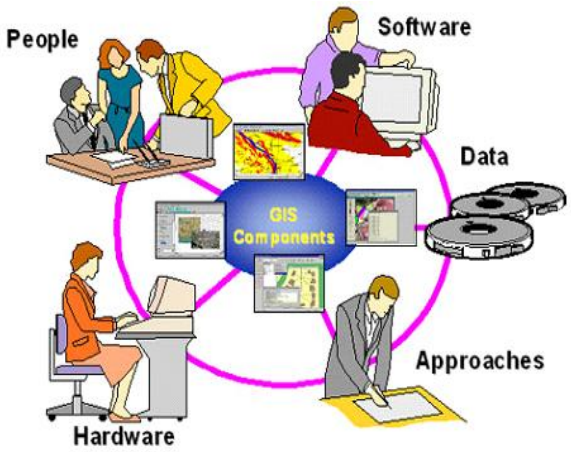
Theo Burrough (1986), GIS là một công cụ mạnh được dùng để lưu trữ và truy vấn, biến đổi và hiển thị dữ liệu không gian từ thế giới thực cho những mục tiêu khác nhau.

Có thể kết luận, Hệ thống thông tin địa lý (Geographic Information system, GIS) được định nghĩa như là một hệ thông thông tin mà nó sử dụng dữ liệu đầu vào, các thao tác thông tích, cơ sở dữ liệu đầu ra liên quan về mặt địa lý không gian (geographically or geospatial), nhằm trợ giúp việc thu nhận, lưu trữ, quản lý, xử lý, phân tích và hiển thị các thông tin không gian từ thế giới thiệu để giải quyết các vấn đề tổng hợp thông tin cho các mục đích của con người đặt ra, chẳng hạn như : để hỗ trợ việc ra các quyết định cho việc quy hoạch (Planning) và quản lý (Managerment), sử dụng đất (Land use), tài nguyên thiên nhiên (Natural resources), môi trường (Enviroment), giao thông (Transportation), dễ dàng cho việc quy hoạch phát triển đô thị và những việc lưu trữ dữ liệu hành chính (*Nguyễn Kim Lợi và cộng tác viên, 2009*).

3.1.3. Thành phần của GIS

Hệ thống thông tin địa lý được tiếp cận dựa trên mô hình thông tin địa lý gồm 5 thành phần sau đây:

- Phần cứng (Hardwave)
- Phần mềm (Softwave)
- Con người (People)
- Số liệu (Data)
- Chính sách và quản lý (Approaches)



Hình 3.1. Các thành phần của GIS

Phần cứng: Sản phẩm được nghiên cứu xây dựng trong đề tài sẽ được cài đặt liên hệ thống thiết bị phần cứng hiện hữu tại các đơn vị sẽ tiếp nhận các kết quả cuối cùng của đề tài. Phần cứng gồm máy tính và các thiết bị ngoại vi để nhập và xuất dữ liệu.

Phần mềm : Những phần mềm cần thiết trong một hệ thống GIS chuyên ngành bao gồm hệ quản trị cơ sở dữ liệu, phần mềm GIS và phần mềm ứng dụng. Phần mềm là tập hợp các câu lệnh, chỉ thị nhằm điều khiển phần cứng của máy tính thực hiện một nhiệm vụ xác định. Phần mềm HTTTĐL có thể là một hoặc tổ hợp các phần mềm máy tính. Một số phần mềm GIS như : MapInfo, ArcInfo, SPANS, WINGIS…

Chính sách và quản lý:

- Hệ thống GIS cần được điều hành bởi một bộ phận quản lý, bộ phận này phải được bổ nhiệm để tổ chức hoạt động hệ thống GIS một cách có hiệu quả, phục vụ người sử dụng thông tin.
- Để hoạt động thành công, hệ thống GIS phải được đặt trong một khuôn khổ tổ chức phù hợp và có những hướng dẫn cần thiết để quản lý, thu thập, lưu trữ và phân tích số liệu, đồng thời có khả năng phát triển được hệ thống GIS theo nhu cầu. Hệ thống GIS cần được điều hành bởi một bộ phận quản lý, bộ phận này được bổ nhiệm để tổ chức hoạt động hệ thống GIS một cách có hiệu quả, nhằm phục vụ người sử dụng thông tin một cách tốt nhất.

Số liệu:

Số liệu được sử dụng trong GIS không chỉ là số liệu địa lý (geo-referenced-data) riêng lẽ mà còn phải được thiết kế trong một CSDL (database).

Những thông tin địa lý bao gồm các dữ kiện về vị trí địa lý, thuộc tính (attributes) của thông tin, mối liên hệ không gian (spatial relationships) giữa các thông tin. Có 2 dạng số liệu được sử dụng trong kỹ thuật GIS là:

1. CSDL bản đồ: là những mô tả hình ảnh bản đồ được số hóa theo một khuôn dạng nhất định mà máy tính hiểu được. Hệ thống thông tin địa lý dùng CSDL này đề xuất ra các bản đồ trên màn hình hoặc ra các thiết bị ngoại vi khác như máy in, máy vẽ.

- Số liệu Vector: được trình bày dưới dạng điểm, đường và vùng, mỗi dạng có liên quan đến một số liệu thuộc tính được lưu trữ trong CSDL.
- Số liệu Raster: được trình bày dưới dạng lưới ô vuông hay ô chữ nhật đều nhau, giá trị được ấn định cho mỗi ô sẽ chỉ định giá trị của thuộc tính. Số liệu của ảnh vệ tinh và số liệu bản dồ được quét (scanned map) là các loại số liệu Raster.

2. Số liệu thuộc tính (Attribute): được trình bày dưới dạng các ký tự hoặc số, hoặc ký hiệu để mô tả các thuộc tính của các thông tin thuộc về địa lý.

Con người: là yếu tố quyết định sự thành công trong quá trình vận hành và khai thác hệ thống thông tin địa lý, do đó việc nâng cao trình độ khoa học kĩ thuật của cán bộ vận hành, khai thác và phát triển hệ thống.

- Nhóm kỹ thuật viên: thao tác trực tiếp các phần mềm để thu thập, tổ chức, hiển thị thông tin.
- Nhóm chuyên viên GIS: sử dụng GIS để đánh giá và thiết kế phân tích các bài toán.
- Nhóm người khai thác sử dụng: những người thuộc chuyên môn khác nhau nhưng cần dùng GIS để giải quyết những vấn đề cụ thể. [7]

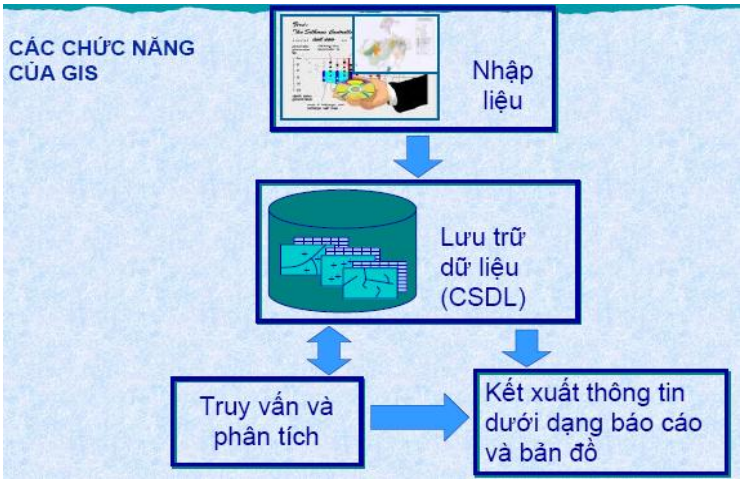
3.1.4. Chức năng của GIS

❖ Một hệ thống phần mềm xử lý HTTĐL yêu cầu tối thiểu phải có ba chức năng sau:

- Tự động hóa bản đồ (Mapping office).
- Quản lý cơ sở dữ liệu (Data base management system - DBMS)
- Xử lý dữ liệu: Đây là chức năng quan trọng để phân biệt một phần mềm HTTĐL với một phần mềm bản đồ khác.

❖ Các chức năng của GIS:

- Tổ chức dữ liệu
- Lưu trữ dữ liệu (CSDL)
- Truy vấn và phân tích dữ liệu
- Hiển thị dữ liệu
- Xuất dữ liệu



Hình 3.2. Các chức năng của GIS

3.1.5. Cấu trúc cơ sở dữ liệu trong GIS

Một cơ sở dữ liệu của hệ thống thông tin địa lý có thể chia ra làm 2 loại cơ bản: số liệu không gian và phi không gian. Mỗi loại có những đặc trưng riêng và chúng khác nhau về yêu cầu và lưu giữ số liệu, hiệu quả, xử lý và hiển thị.

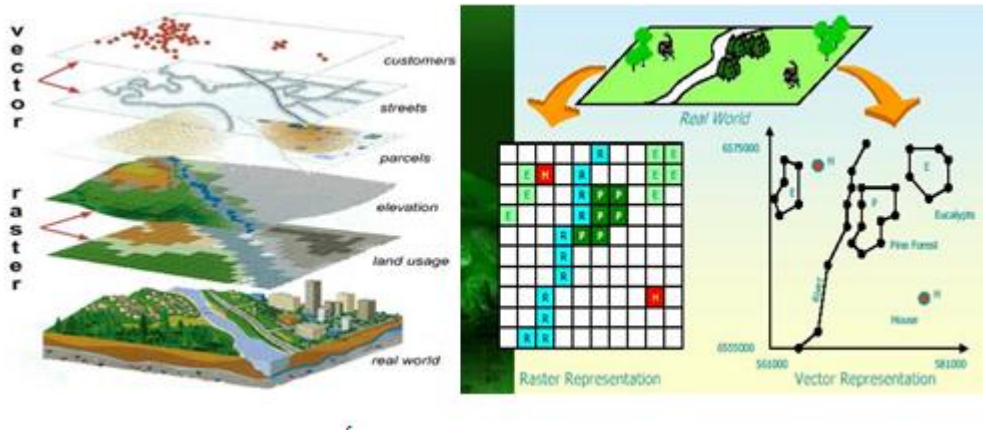
Số liệu không gian là những mô tả của hình ảnh bản đồ số, chúng bao gồm tọa độ, quy luật, các ký hiệu dùng để xác định một hình ảnh bản đồ cụ thể trên từng bản đồ. Hệ thống thông tin địa lý dùng các số hiệu không gian để tạo ra một bản đồ hay hình ảnh bản đồ trên giấy thông qua thiết bị ngoại vi…

Số liệu phi không gian là những diễn tả đặc tính, số lượng, mối quan hệ các hình ảnh bản đồ với vị trí địa lý của chúng. Các số liệu phi không gian được gọi là dữ liệu thuộc tính, chúng liên quan đến vị trí địa lý hoặc các đối tượng không gian và liên kết chặt chẽ với chũng trong hệ thống thông tin địa lý thông qua một cơ chế thống nhất chung.

Dữ liệu không gian:

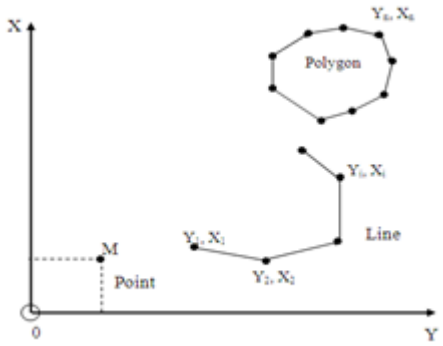
Cơ sở dữ liệu không gian chứa đựng những thông tin định vị của các đối tượng, cho biết vị trí, kích thước, hình dạng, sự phân bố… của các đối tượng. Các đối

tượng không gian được định dạng về 3 loại: đối tượng dạng điểm, dạng đường và dạng vùng. Dữ liệu không gian có hai mô hình lưu trữ: mô hình dữ liệu raster và mô hình dữ liệu vector.



Hình 3.3. Cấu trúc dữ liệu Raster và Vector

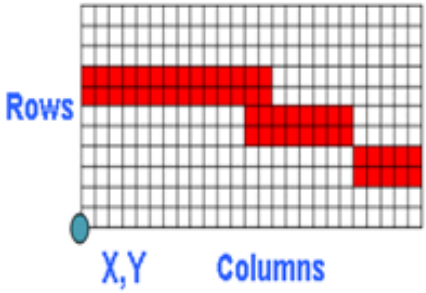
- **Mô hình dữ liệu Vector :** thông tin về điểm, đường, vùng được mã hóa và lưu dưới dạng tập hợp các tọa độ x,y. Đối tượng dạng điểm lưu dưới dạng tọa độ (x,y). Đối tượng dạng đường như đường giao thông, sông suối… được lưu dưới dạng tập hợp các tọa độ điểm x_1y_1, x_2y_2,…, x_ny_n hoặc là một hàm toán học, tính được chiều dài. Đối tượng dạng vùng như khu vực buôn bán, nhà cửa, thủy hệ... được lưu như một vòng khép kín của các điểm tọa độ, tính được chu vi và diện tích vùng.



Hình 3.4. Biểu diễn thông tin điểm, đường, vùng theo cấu trúc vector

- **Mô hình dữ liệu Raster**: Trong cấu trúc dữ liệu Raster, đối tượng được biểu diễn thông qua các ô (cell) hay ô ảnh (pixel) của một lưới các ô. Trong máy tính, các ô lưới này được lưu trữ dưới dạng ma trận trong đó mỗi ô lưới là giao điểm của một hàng và một cột trong ma trận. Điểm được xác định bởi một pixel (giá trị nhỏ nhất trong cấu trúc Raster), đường được xác định bởi một chuỗi các

ô có cùng thuộc tính kề nhau có hướng nào đó, còn vùng được xác định bởi một số các pixel cùng thuộc tính phủ lên trên một diện tích nào đó.



Hình 3.5. Cấu trúc dữ liệu Raster

Mối quan hệ logic giữa vị trí của các đối tượng trong cấu trúc dữ liệu được gọi là TOLOGY. Cấu trúc dữ liệu thuộc topology cung cấp một cách tự động hóa để xử lý việc số hóa, xử lý lỗi; giảm dung lượng lưu trữ dữ liệu.

- **Dữ liệu thuộc tính:**

Cơ sở dữ liệu thuộc tính lưu trữ các số liệu mô tả các đặc trưng, tính chất, … của đối tượng nghiên cứu. Các thông tin này có thể là định tính hay định lượng, được lưu trữ trong máy tính như là tập hợp các con số hay ký tự; ở dạng văn bản và bảng biểu. Thông thường, dữ liệu thuộc tính là các thông tin chi tiết cho đối tượng hoặc các số liệu thống kê cho đối tượng. Các dữ liệu thuộc tính chủ yếu được tổ chức thành các bảng dữ liệu, gồm có các cột dữ liệu (trường dữ liệu): mỗi cột diễn đạt một trong nhiều thuộc tính của đối tượng; và các hàng tương ứng với một bản ghi: gồm toàn bộ nội dung thuộc tính của một đối tượng quản lý. [17]

3.1.6. Tổ chức cơ sở dữ liệu GIS

Cơ sở dữ liệu là một gói dữ liệu được tổ chức dưới dạng các Layer. Các Layer có thể được tạo ra từ nhiều khuôn dạng dữ liệu khác nhau như: Shape files, personal geodatabase, ArcInfo cover datasets, CAD drawings, SDE databases, photo, image. Hiện nay, theo các chuẩn dữ liệu ISO-TC 211 và chuẩn dữ liệu của Bộ Tài nguyên và Môi trường, dữ liệu được tổ chức theo khuôn dạng chuẩn là **GeoDatabase**. [17]

3.1.7. Công cụ Network Analyst trong ArcGis

Network là một hệ thống các yếu tố liên kết với nhau, chẳng hạn như các cạnh (đường) và các nút kết nối (điểm), đại diện cho các tuyến có thể từ một địa điểm khác

Người, tài nguyên và hàng hóa có xu hướng đi du lịch dọc theo mạng: xe hơi và xe tải đi trên những con đường, máy bay bay trên đường bay được xác định trước, dầu

chảy trong đường ống. Bằng mô hình con đường du lịch tiềm năng với một mạng, nó có thể thực hiện các phân tích liên quan đến sự chuyển động của dầu, xe tải, hoặc các đại lý khác nhau trên mạng. Phân tích mạng lưới phổ biến nhất là tìm đường đi ngắn nhất. [19]

ArcGIS Network Analyst cung cấp chức năng phân tích không gian dựa trên hệ thống mạng lưới như tuyến đường, tuyến tàu, định hướng du lịch, cơ sở gần nhất, khu vực dịch vụ và location – allocation. Với ArcGIS Network Analyst, bạn có thể tự động mô hình mạng lưới thực tế, bao gồm đường một chiều, giới hạn rẽ và độ cao, giới hạn tốc độ, và nhiều loại tốc độ khác nhau tùy vào tình hình giao thông. Bạn có thể dễ dàng xây dựng các mạng lưới cho mình từ dữ liệu GIS bằng mô hình dữ liệu mạng lưới phức tạp.

Với ArcGIS Network Analyst, bạn có thể:

- Tìm đường đi ngắn nhất.
- Xác định các tuyến đường hiệu quả nhất cho đội xe phải đi đến nhiều địa điểm.
- Sử dụng khung thời gian để giới hạn thời gian các phương tiện có thể đi đến các địa điểm. Xác định các cơ sở gần nhất.
- Xác định vị trí tối ưu cho các cơ sở bằng phân tích location-allocation.
- Xác định khu vực dịch vụ dựa trên thời gian di chuyển hoặc khoảng cách.
- Tạo ra mạng lưới dựa trên dữ liệu GIS hiện có của bạn.
- Tạo ra một ma trận chi phí đi lại từ mỗi điểm khởi hành cho tất cả các điểm đến. [18]

3.2. Tổng quan về đồ thị

Lý thuyết đồ thị là một lĩnh vực đã có từ lâu và có nhiều ứng dụng hiện đại. Những tư tưởng cơ bản của lý thuyết đồ thị được đề xuất vào những năm đầu của thế kỷ 18 bởi nhà toán học lỗi lạc người Thụy Sỹ Lenhard Eurler. Chính ông là người đã sử dụng đồ thị để giải bài toán nổi tiếng về các cái cầu ở thành phố Konigsberg.

Đồ thị được sử dụng để giải các bài toán trong nhiều lĩnh vực khác nhau. Chẳng hạn, đồ thị có thể sử dụng để xác định các mạch vòng trong vấn đề giải thích mạch điện. Chúng ta có thể phân biệt các hợp chất hóa học hữu cơ khác nhau với cùng công thức phân tử nhưng khác nhau về cấu trúc phân tử nhờ đồ thị. Chúng ta có thể xác định hai máy tính trong mạng có thể trao đổi thông tin được với nhau hay không nhờ mô

hình đồ thị của mạng máy tính. Đồ thị có trọng số trên các cạnh có thể sử dụng để giải các bài toán như: Tìm đường đi ngắn nhất giữa hai thành phố trong mạng giao thông. Chúng ta cũng còn sử dụng đồ thị để giải các bài toán về lập lịch, thời khóa biểu và phân bố tần số cho các trạm phát thanh và truyền hình; giải các bài toán tô màu trên bản đồ: tìm số màu ít nhất để tô các quốc gia sao cho hai quốc gia kề nhau được tô khác nhau… [6]

3.2.1. Định nghĩa đồ thị (Graph)

Đồ thị là một cấu trúc rời rạc bao gồm các đỉnh và các cạnh nối các đỉnh này. Được mô tả hình thức:

$$G = (V, E)$$

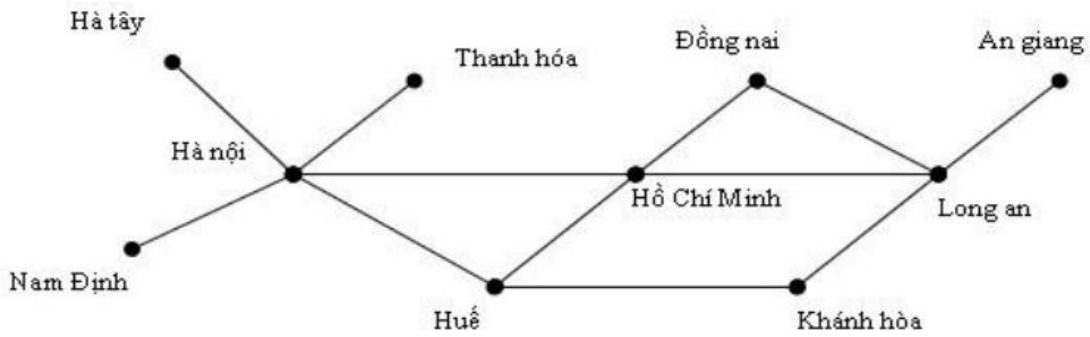
V gọi là tập các **đỉnh** (Vertices) và E gọi là tập các **cạnh** (Edges). Có thể coi E là tập các cặp (u, v) với u và v là hai đỉnh của V.

Một số hình ảnh của đồ thị:

Sơ đồ giao thông | Mạng máy tính

Hình 3.6. Ví dụ về mô hình đồ thị

Chúng ta phân biệt các loại đồ thị khác nhau bởi kiểu và số lượng cạnh nối hai đỉnh nào đó của đồ thị. Để có thể hình dung được tại sao lại cần đến các loại đồ thị khác nhau, chúng ta sẽ nêu ví dụ sử dụng chúng để mô tả một mạng máy tính. Giả sử ta có một mạng gồm các máy tính và các kênh điện thoại (gọi tắt là kênh thoại) nối các máy tính này. Chúng ta có thể biểu diễn các vị trí đặt máy tính bởi các điểm và các kênh thoại nối chúng bởi các đoạn nối, xem hình 3.7.



Hình 3.7. Sơ đồ mạng máy tính

Nhận thấy rằng trong mạng ở hình 3.7, giữa hai máy bất kỳ chỉ có nhiều nhát là một kênh thoại nối chúng, kênh thoại này cho phép liên lạc cả hi chiều và không có máy tính nào lại được nối với chính nó. Sơ đồ mạng máy cho trong hình 3.7 được gọi là đơn vị vô hướng.

❖ **Các khái niệm cơ bản**

Định nghĩa 1: Một đơn đồ thị G = (V, E) gồm một tập khác rỗng V mà các phần tử của nó gọi là các đỉnh và một tập hợp E mà các phần tử của nó gọi là các cạnh, đó là các cặp không có thứ tự của các đỉnh phân biệt.

Định nghĩa 2: Một đa đồ thị G = (V, E) gồm một tập khác rỗng V mà các phần tử của nó gọi là các đỉnh và một họ E mà các phần tử của nó gọi là các cạnh, đó là các cặp không có thứ tự của các đỉnh phân biệt. Hai cạnh được gọi là cạnh bội hay song song nếu chúng cùng tương ứng với một cặp đỉnh.

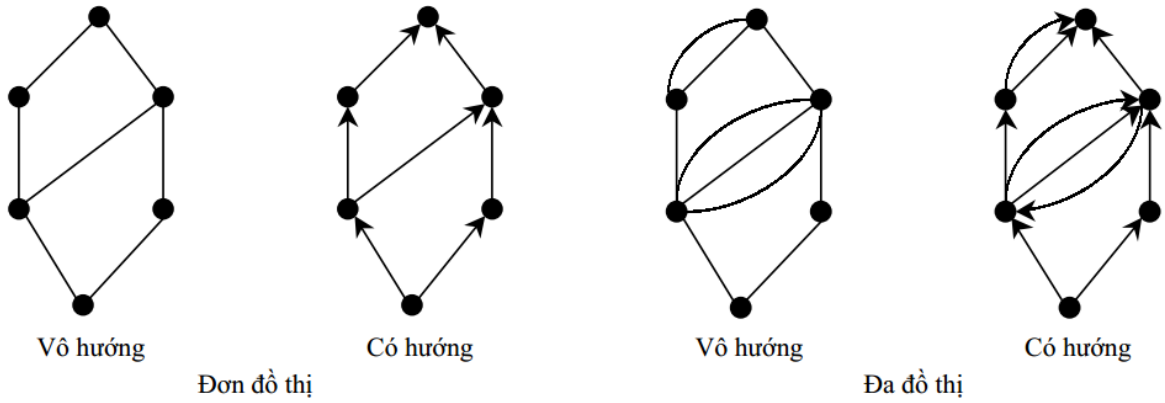
Rõ ràng mỗi đơn đồ thị là đa đồ thị, nhưng không phải đa đồ thị nào cũng là đơn đồ thị.

Định nghĩa 3: Một giả đồ thị G = (V, E) gồm một tập khác rỗng V mà các phần tử của nó gọi là các đỉnh và một họ E mà các phần tử của nó gọi là các cạnh, đó là các cặp không có thứ tự của các đỉnh (không nhất thiết là phân biệt).

Với v∈V, nếu (v,v)∈ E thì ta nói có một khuyên tại đỉnh v.

Tóm lại, giả đồ thị là loại đồ thị vô hướng tổng quát nhất vì nó có thể chứa các khuyên và các cạnh bội. Đa đồ thị là loại đồ thị vô hướng có thể chứa cạnh nhưng không thể có có các khuyên, còn đơn đồ thị là loại đồ thị vô hướng không chứa cạnh bội hoặc các khuyên.

Ví dụ:



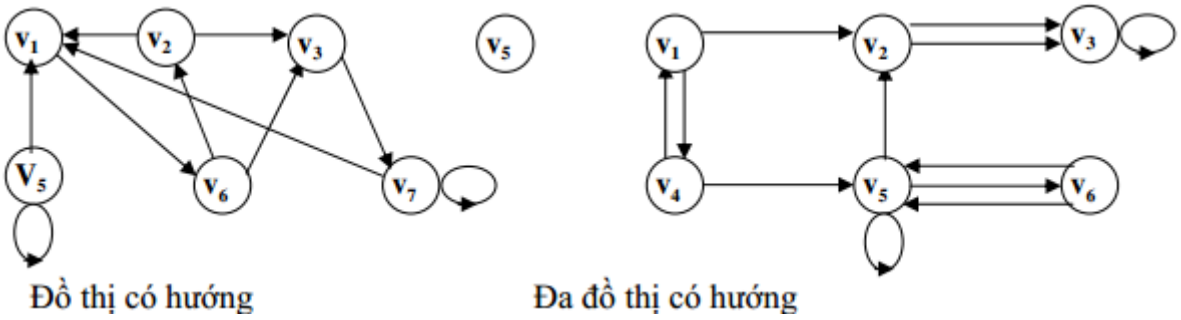
Hình 3.8. Phân loại đồ thị

Định nghĩa 4: Một đồ thị có hướng G = (V, E) gồm một tập khác rỗng V mà các phần tử của nó gọi là các đỉnh và một tập E mà các phần tử của nó gọi là các cung, đó là các cặp có thứ tự của các phần tử thuộc V.

Định nghĩa 5: Một đa đồ thị có hướng G = (V, E) gồm một tập hợp khác rỗng V mà các phần tử của nó gọi là các đỉnh và một họ E mà các phần tử của nó gọi là các cung, đó là các cặp có thứ tự thuộc V.

Đồ thị vô hướng nhận được từ đồ thị có hướng G bằng cách xóa bỏ các chiều mũi tên trên các cung được gọi là đồ thị vô hướng nền của G. [4]

Ví dụ:



3.3. Biểu diễn đồ thị trên máy tính

Để lưu trữ đồ thị và thực hiện các thuật toán khác nhau với đồ thị trên máy tính cần phải tìm 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 của thuật toán. Vì vậy, việc chọn lựa cấu trúc dữ liệu để biểu diễn đồ thị phụ thuộc vào từng tình huống cụ thể (bài toán và thuật toán cụ thể). Trong mục này chúng ta sẽ xét một số phương pháp cơ

bản được sử dụng để biểu diễn đồ thị trên máy tính, đồng thời cũng phân tích một cách ngắn gọn những ưu điểm cũng như nhược điểm của chúng.

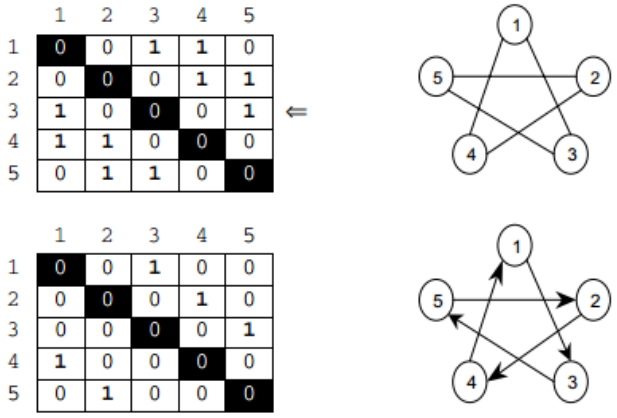
3.3.1. Ma trận liền kề (Ma trận kề)

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 là các đỉnh được đánh số 1, 2, …, n. Khi đó ta có thể biểu diễn đồ thị bằng một ma trận vuông A = [a_{ij}] cấp n. Trong đó :

- a_{ij} = 1 nếu (i, j) ∈ E
- a_{ij} = 0 nếu (i , j) ∉ E
- Quy ước $a_{ii} = 0$ với ∀i ;

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_{ij} mà là ghi số cạnh nối giữ đỉnh i và đỉnh j.

Ví dụ :



Các tính chất của ma trận liền kề:

1. Đối với đồ thị vô hướng G, thì ma trận liền kề tương ứng là ma trận đối xứng (a_{ij}=a_{ji}), điều này không đúng với đồ thị có hướng.
2. Nếu G là đồ thị vô hướng và A là ma trận liền kề tương ứng thì ma trận A :

   Tổng các số trên hàng i = Tổng các số trên cột i = Bậc của đỉnh i = deg(i)
3. Nếu G là đồ thị có hướng và A là ma trận liền 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 liền kề A tương ứng là các phần tử logic. a_{ij} = TRUE nếu (i, j) ∈ E và a_{ij} = FALSE nếu (i, j) ∉ E

❖ Ưu điểm của ma trận liền kề:

- Đơn giản, trựa 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 phép so sánh : $a_{uv} \neq 0$.

❖ Nhược điểm của ma trận liền kề:

- Bất kể số cạnh của đồ thị là nhiều hay ít, ma trận liền kề luôn đòi hỏi n^2 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 kỳcủa đồ thị, nhiều khi ta phải xét tất cảcác đỉnh v khác kề với nó, hoặc xét tất cả các cạnh liên thuộc với nó. Trên ma trận liền kề việc đó được thực hiện bằng cách xét tất cả các đỉnh v và kiểm tra điều kiện $a_{uv} \neq 0$. Như vậy, ngay cả khi đỉnh u là đỉnh cô lập (không kề với đỉnh nào) hoặc đỉnh treo (chỉ kềvới 1 đỉnh) ta cũng buộc phải xét tất cả các đỉnh và kiểm tra điều kiện trên dẫn tới lãng phí thời gian. [4]

3.4 Đồ thị Euler và đồ thị Halmiton

3.4.1 Đồ thị Euler

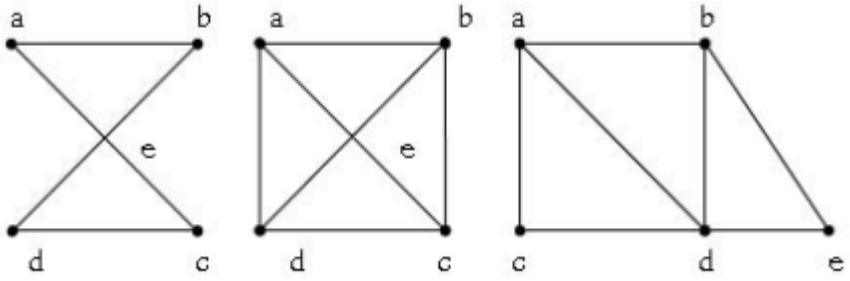
Trong chương này chúng ra sẽ nghiên cứu hai dạng đồ thị đặc biệt là đồ thị Euler và đồ thị Hamilton. Dưới đây, nếu không có giải thích bổ sung, thuật ngữ đồ thị được dùng để chỉ chung đa đồ thị vô hướng và có hướng, và thuật ngữ cạnh sẽ dùng để chỉ chung cạnh của đồ thị vô hướng cũng như cung của đồ thị có hướng.

3.4.1.1. Khái niệm về đường đi và chu trình Euler

Định nghĩa 1: Chu trình đơn trong đồ thị G đi qua mỗi cạnh của nó một lần được gọi là chu trình Euler. Đường đi đơn trong G đi qua mỗi cạnh của nó một lần được gọi là đường đi Euler. Đồ thị được gọi là đồ thị Euler nếu nó có chu trình Euler, và gọi là đồ thị nửa Euler nếu nó có đường đi Euler.

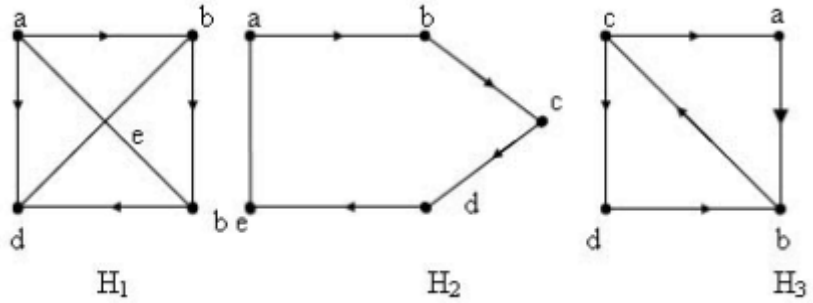
Rõ ràng mọi đồ thị Euler luôn là nửa Euler, nhưng điều ngược lại không luôn đúng.

Thí dụ 1. Đồ thị G_1 trong hình 1 là đồ thị Euler vì nó có chu trình Euler a, e, c, d, e, b, a. Đồ thị G_3 không có chu trình Euler nhưng nó có đường đi Euler a, c, d, e, b, d, a, b, vì thế G_3 là đồ thị cửa Euler. Đồ thị G_2 không có chu trình cũng như đường đi Euler.



Hình 3.9. Đồ thị G1, G2, G3

Thí dụ 2. Đồ thị H_2 trong hình 2 là đồ thị Euler vì nó có chu trình Euler a, b, c, d, e, a. Đồ thị H_3 không có chu trình Euler nhưng nó có đường đi Euler c, a, b, c, d, b vì thế H_3 là đồ thị nửa Euler. Đồ thị H_1 không có chu trình cũng như đường đi Euler.



Hình 3.10. Đồ thị H1, H2, H3

Điều kiện cần và đủ để một đồ thị là một đồ thị Euler tìm ra vào năm 1736 khi ông giải quyết bài toán hóc búa nổi tiếng thế giới thời đó về bảy cái cầu ở thành phố Konigsberg và đây là định lý đầu tiên của lý thuyết đồ thị.

3.4.1.2. Điều kiện tồn tại đường đi hoặc chu trình Euler

Định lý 1 (Euler). Đồ thị vô hướng liên thông G là đồ thị Euler khi và chỉ khi mọi đỉnh của G đều có bậc chẵn.

Để chứng minh định lý trước hết ta chứng minh bổ đề :

Bổ đề. Nếu bậc của mỗi đỉnh của đồ thị G không nhỏ hơn 2 thì G chứa chu trình.

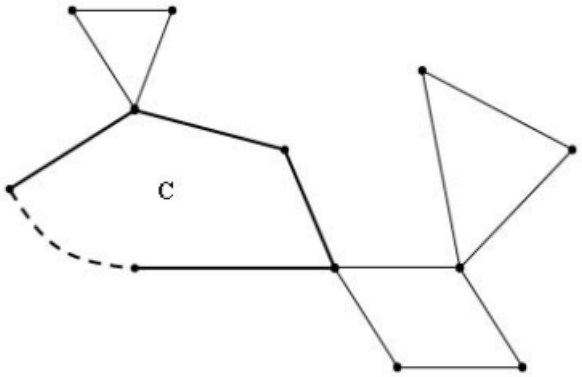
Chứng minh :

Nếu G có cạnh lặp thì khẳng định của bồ đề là hiển nhiên. Vì vậy giả sử G là đơn đồ thị. Gọi v là một đỉnh nào đó của G. Ta sẽ xây dựng theo qui nạp đường đi v, v_1, v_2,… trong đó v_1 là đỉnh kề với v, còn với $i \geq 1$ chọn $v_{i+1} \# v_{i-1}$ (có thể chọn v_{i+1} như vậy vì deg(vi) ≥2). Do tập đỉnh cyar G là hữu hạn, nên sau một số hữu hạn bước ta phải quay lại một đỉnh đã xuất hiện trước đó. Gọi đỉnh đầu tiên như thế là vk. Khi đó, đoạn của đường đi xây dựng nằm hai đỉnh vk là chu trình cần tìm.

Chứng minh định lý :

Cần. Giả sử G là đồ thị Euler tức là tồn tại chu trình Euler P trong G. Khi đó cứ mỗi lần chu trình P đi qua một đỉnh nào đó của G bậc của đỉnh đó tăng lên 2. mặt khác mỗi cạnh của đồ thị xuất hiện trong P đúng một lần , suy ra mỗi đỉnh của đồ thị điều khiển có bậc chẵn.

Đủ. Quy nạp theo số đỉnh và các số cạnh của G. Do G liên thông và deg (v) là số chẵn nên bậc của mỗi đỉnh của nó không nhỏ hơn 2. Từ đó theo bổ đề G phải chứa chu trình C. Nếu C đi qua tất cả các cạnh của G thì nó chính là chu trình Euler. Giả sử C không đi qua tất cả các cạnh của G. Khi đó loại bỏ G tất cả các cạnh thuộc C ta thu một đồ thị mới H vẫn có bậc là chẵn. Theo giả sử qui nạp, trong mỗi thành phần liên thông của H điều tìm được chu trình Euler. Do G là liên thông nên trong mỗi thành phần của H có ít nhất một đỉnh chung với chu trình C. Vì vậy, ta có thể xây dựng chu trình Euler trong G như sau: bắt đầu từ một đỉnh nào đó của chu trình C, đi theo các cạnh của C chừng nào chưa gặp phải đỉnh không cô lập của H. Nếu gặp phải đỉnh như vậy ta sẽ đi theo chu trình Euler của thành phần liên thông của H chứa đỉnh đó. Sau đó lại tiếp tục di theo cạnh của C cho đến khi gặp phải đỉnh không cô lập của H thì lại theo chu trình Euler của thành phần liên thông tương ứng trong H v.v… (Xem hình 3.11). Quá trình sẽ kết thúc khi ta trở về đỉnh xuất phát, tức là thu được chu trình đi qua mỗi cạnh của đồ thị đúng một lần.



Hình 3.11. Minh họa cho chứng minh định lý 1

Hệ quả 2. Đồ thị vô hướng liên thông G là nửa Euler khi và chỉ khi nó có không quá 2 đỉnh bậc lẻ.

Chứng minh: Thực vậy, nếu G có không quá 2 đỉnh bậc lẻ thì số đỉnh bậc lẻ của nó chỉ có thể là 0 hoặc 2. Nếu G không có đỉnh bậc lẻ thì theo định lý 1, nó là đồ thị Euler. Giả sử có 2 đỉnh bậc lẻ là u và v. Gọi H là đồ thị thu được từ G bằng cách thêm vào G một đỉnh mới w và hai cạnh (w,u) và (w,v). Khi đó tất cả các đỉnh của H diều có

bậc chẵn, vì thế theo định lý 1, nó có chu trình Euler C. Xóa bỏ khỏi chu trình này đỉnh w và hai cạnh kề nó ta thu được đường đi Euler trong đồ thị của G. [8]

3.4.2 Đồ thị Halmiton

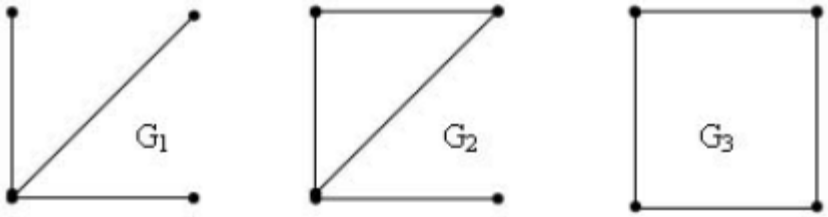
Trong mục này chúng ta xét bài toán tương tự như trong mục trước chỉ khác là ta quan tâm đến đường đi qua các đỉnh của đồ thị, mỗi đỉnh đúng một lần. Sự thay đổi tưởng chừng như là không đáng kể này trên thực tế đã dẫn đến sự phức tạp hóa vấn đề cần giải quyết.

3.4.1.3. Khái niệm về đường đi và chu trình Haminlton

Định nghĩa 2: Đường đi qua tất cả các đỉnh của đồ thị mỗi đỉnh đúng một lần được gọi là đường đi Hamilton. Chu trình bắt đầu từ một đỉnh v nào đó qua tất cả các đỉnh còn lại mỗi đỉnh đúng một lần rồi quay trở về v được gọi là chu trình Hamilton. Đồ thị G được gọi là đồ thị Hamilton nếu có chứa chu trình Hamilton nếu nó có đường đi Hamilton.

Rõ ràng đồ thị Hamilton, nhưng điều ngược lại không còn đúng.

Thí dụ 3. Trong hình 4: G_3 là Hamilton, G_2 là nửa Hamilton còn G_1 không là nửa Hamilton.



Hình 3.12. Đồ thị Hamilton G_3, nửa Hamilton G_2 và G_1.

Cho đến nay việc tìm một tiêu chuẩn nhận biết đồ thị Hamilton vẫn còn là mờ, mặc dù đây là một vấn đề trung tâm của lý thuyết đồ thị. Hơn thế nữa, cho đến nay cũng chưa có thuật toán hiệu quả để kiểm tra một đồ thị có là Hamilton hay không. Các kết quả thu được phần lớn là điều kiện đủ để một đồ thị là đồ thị Hamilton. Phần lớn chúng điều có dạng "nếu G có số cạnh đủ lớn thì G là Hamilton". Một kết quả như vậy được phát biểu trong định lý sau đây.

3.4.2.2. Điều kiện tồn tại đường đi hoặc chu trình Hamilton

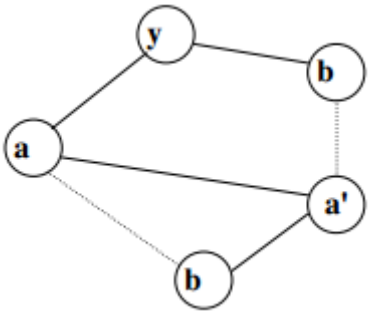
Định lý (Rédei): Nếu G là một đồ thị có hướng đầy đủ thì G là đồ thị nửa Hamilton.

Chứng minh: Giả sử G=(V, E) là đồ thị có hướng đầy đủ và $\alpha=(v_1,v_2, ..., v_{k-1}, v_k)$ là đường đi sơ cấp bất kỳ trong đồ thị G.

- Nếu α đã đi qua tất cả các đỉnh của G thì nó là một đường đi Hamilton của G.
- Nếu trong G còn có đỉnh nằm ngoài α, thì ta có thể bổ sung dần các đỉnh này vào α và cuối cùng nhận được đường đi Hamilton.

Thật vậy, giả sử v là đỉnh tùy ý không nằm trên α

a) Nếu có cung nối v với v_1 thì bổ sung v vào đầu của đường đi α để được $\alpha_1=(v, v_1, v_2,\ldots, v_{k-1}, v_k)$.
b) Nếu tồn tại chỉ số I ($1 \le i \le k-1$) mà từ v_i có cung nối tới v và từ v có cung nối tới v_{i+1} thì ta chen v vào giữa v_i và v_{i+1} để được đường đi sơ cấp $\alpha_2=(v_1, v_2,\ldots, v_i, v, v_{i+1},\ldots, v_k)$.
c) Nếu cả hai khả năng trên đều không xảy ra nghĩa là với mọi I ($1 \le i \le k$) v_i đều có cung đi tới v. Khi đó bổ sung v vào cuối của đường đi $\alpha_3=(v_1, v_2,\ldots, v_{k-1}, v_k, v)$.



Nếu đồ thị G có n đỉnh thì sau n-k bổ sung ta sẽ nhận được đường đi Hamilton.

Định lý (Dirac, 1952): Nếu G là một đơn đồ thị có n đỉnh và mọi đỉnh của G đều có bậc không nhỏ hơn $\frac{n}{2}$ thì G là một đồ thị Hamilton.

Chứng minh: Định lý được chứng minh bằng phản chứng. Giả sử G không có chu trình Hamilton. Ta thêm vào G một số đỉnh mới và nối mỗi đỉnh mới này với mọi đỉnh của G, ta được đồ thị G'. Giả sử k (>0) là số tối thiểu các đỉnh cần thiết để G' chứa một chu trình Hamilton. Như vậy, G' có n+k đỉnh.

Gọi P là chu trình Hamilton ayb …a trong G', trong đó a và b là các đỉnh của G, còn y là một trong các đỉnh mới. Khi đó b không kề với a, vì nếu trái lại thì ta có thể bỏ đỉnh y và được chu trình ab…a, mâu thuẫn với giả thuyết về tính chất nhỏ nhất của k.

Ngoài ra, nếu a' là một đỉnh kề nào đó của a (khác với y) và b' là đỉnh nối tiếp ngay a' trong chu trình P thì b' không thể là đỉnh kề với b vì nếu trái lại thì ta có thể thay P bởi chu trình aa' …bb' …a, trong đó không có y, mâu thuẫn với giả thiết về tính chất nhỏ nhất của k.

Như vậy, với mỗi đỉnh kề với a, ta có một đỉnh không kề với b, tức là số đỉnh không kề với b không thể ít hơn số đỉnh kề với a (số đỉnh kề với a không nhỏ hơn $\frac{n}{2} + k$). Mặt khác, theo giả thiết số đỉnh kề với b cũng không nhỏ hơn $\frac{n}{2} + k$. Vì không có đỉnh nào vừa kề với b lại vừa không kề với b, nên số đỉnh của G' không ít hơn 2 ($\frac{n}{2} + k$)=n+2k, mâu thuẫn với giả thiết là số đỉnh của G' bằng n+k (k>0). Định lý được chứng minh.

Hệ quả: Nếu G là đơn đồ thị có n đỉnh và mọi đỉnh của G đều có bậc không nhỏ hơn $\frac{n-1}{2}$ thì G là đồ thị nửa Hamilton.

Chứng minh: Thêm vào G một đỉnh x và nối x với mọi đỉnh của G thì ta nhận được đơn đồ thị G' có n+1 đỉnh và mỗi đỉnh có bậc không nhỏ hơn $\frac{n+1}{2}$. Do đó trong G' có một chu trình Hamilton. Bỏ x ra khỏi chu trình này, ta nhận được đường đi Hamilton trong G.

Định lý (Ore, 1960): Nếu G là một đơn đồ thị có n đinht và bất kỳ hai đỉnh nào không kề nhau cũng có tổng số bậc không nhỏ hơn n thì G là một đồ thị Hamilton.

Định lý: Nếu G là đồ thị phân đôi với hai tập đỉnh là V_1, V_2 có số đỉnh cùng bằng n (n ≥2) và bậc của mỗi đỉnh lớn hơn $\frac{n}{2}$ thì G là một đồ thị Hamilton. [3]

3.5. Bài toán tô màu và bài toán lập lịch

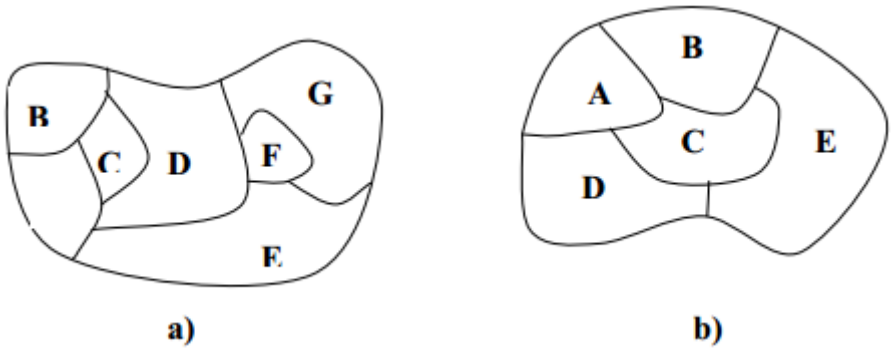
3.5.1. Bài toán tô màu

3.5.1.1. Giới thiệu tô màu đồ thị

Vấn đề liên quan đến tô màu các miền trên bản đồ, ví dụ bản đồ các vùng trên thế giới đã dẫn đến nhiều kết quả trong lý thuyết đồ thị. Khi tô màu bản đồ, ta thường tô 2 miền có chung đường biên giới bằng 2 màu khác nhau. Để đảm bảo điều này, ta có thể sử dụng màu sắc riêng cho mỗi miền. Tuy nhiên, cách làm này là không hiệu quả, và nếu bản đồ có quá nhiều miền, sẽ rất khó để phân biệt giữa các miền có màu sắc gần giống nhau. Do đó, ta nên sử dụng số màu ít nhất có thể được. Nó dẫn đến bài toán xác định số màu ít nhất có thể được. Nó dẫn đến bài toán xác định số màu tối thiểu cần sử dụng để tô màu các miền bản đồ sao cho các miền lân cận luôn khác nhau nhau.

Ví dụ: Bản đồ H1a có thể tô được bằng 4 màu, nhưng không thể tô bằng 3 màu -> Số màu tối thiểu phải là 4.

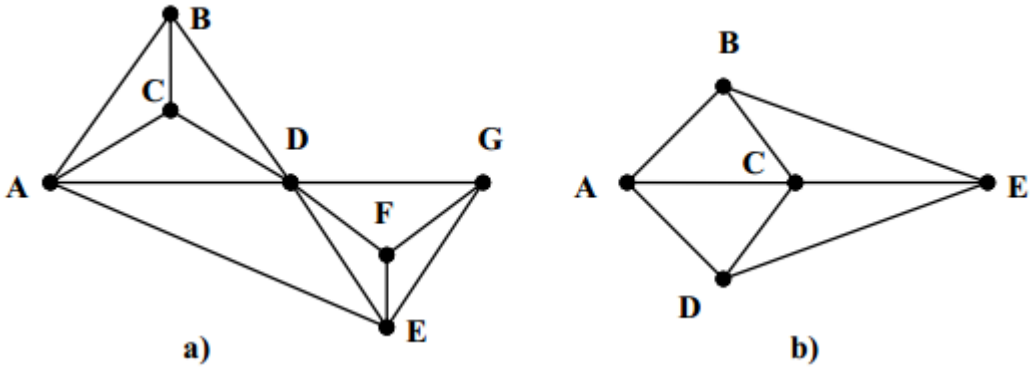
Bản đồ H1b có thể tô bằng 3 màu, nhưng 2 màu là không thể -> Số màu tối thiểu là 3.



H1: 2 bản đồ ví dụ

Mỗi bản đồ trên mặt phẳng có thể biểu diễn bằng đồ thị : Mỗi miền biểu diễn bằng 1 đỉnh ; 2 đỉnh sẽ được nối với nhau khi 2 miền tương ứng có chung đường biên giới. Hai miền chỉ tiếp xúc nhau tại 1 điểm coi như không kề nhau. Đồ thị này được gọi là đồ thị kép của bản đồ. Từ phương pháp xấy dựng đồ thị kép của 1 bản đồ, dễ thấy mỗi bản đồ phẳng sẽ tương ứng với 1 đồ thị kép phẳng. [6]

H2 thể hiện đồ thị phẳng tương ứng của các bản đồ trong H1.



H2: Đồ thị kép của các bản đồ trong H1

Trong Lý thuyết đồ thị, tô màu đồ thị tiếng Anh: *graph coloring*) là trường hợp đặc biệt của gán nhãn đồ thị, mà trong đó mỗi đỉnh hay mỗi cạnh hay mỗi miền của đồ thị có thể được gán bởi một màu hay một tập hợp các màu nào đó. Tô màu đồ thị có thể là:

- **tô màu đỉnh** (tiếng Anh: *vertex coloring*) là gán cho mỗi đỉnh của đồ thị một màu nào đó sao cho không có hai đỉnh nào liền kề lại trùng màu nhau;
- **tô màu cạnh** (tiếng Anh: *edge coloring*) là gán cho mỗi cạnh của đồ thị một màu nào đó sao cho sao cho không có 2 cạnh nào trùng màu;

- **tô màu miền** (tiếng Anh: *face coloring*) là gán cho mỗi miền của đồ thị phẳng một màu sao cho không có 2 miền có chung đường biên lại cùng màu.

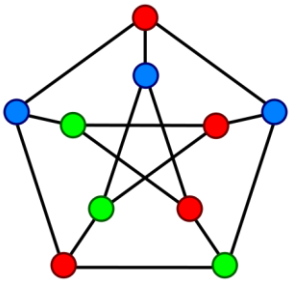
Sắc số (tiếng Anh: *chromatic number*) của một đồ thị là số màu ít nhất để tô các đỉnh. Sắc số của đồ thị G được kí hiệu là $\chi(G)$.

Số màu cạnh (tiếng Anh: *chromatic index*) của một đồ thị là số màu ít nhất dùng để tô các cạnh. Số màu cạnh của đồ thị G được kí hiệu là $\chi'(G)$.

Số màu cạnh của đồ thị G bất kì bằng sắc số của đồ thị đường $L((G))$ của đồ thị đó:

$$\chi'(G) = \chi(L(G)),$$

do đó việc nghiên cứu tô màu cạnh của G tương đương với nghiên cứu tô màu đỉnh của $L(G)$. [21]



Hình 3.13. Đồ thị Petersen có sắc số bằng 3

3.5.1.2. Các khái niệm cơ bản

❖ **Định nghĩa 1:**

Phép tô màu của một đồ thị đơn là một quy tắc tô mỗi đỉnh đồ thị một màu cụ thể sao cho không có 2 đỉnh kề nhau nào được tô cùng màu.

1 đồ thị có thể tô màu bằng các màu khác nhau cho mỗi đỉnh. Tuy nhiên, trong phần lớn của các đồ thị, ta có thể tô bằng số màu ít hơn số đỉnh. Vậy số màu tối thiểu cần sử dụng là bao nhiêu?

❖ **Định nghĩa 2:**

Số màu của một đồ thị G (kí hiệu **c**(G)) là số màu tối thiểu cần sử dụng để tô màu đồ thị này.

Chú ý rằng số màu của 1đồ thị phẳng chính là số màu tối thiểu cần sử dụng để tô màu các miền bản đồ phẳng sao cho không có 2 miền nào kề nhau và được tô cùng màu. Bài toán này đã được nghiên cứu hơn 100 năm, dẫn đến một trong các định lý nổi tiếng nhất của toán học:

❖ **Định lý 4 màu:**

Số màu của 1 đồ thị phẳng không lớn hơn 4.

Giả thuyết 4 màu được đề ra từ những năm 1850. Nó cuối cùng đã được chứng minh bởi 2 nhà toán học Mĩ là Kenneth Appel và Wolfgang Haken năm 1976. Trước đó, nhiều người đã đề ra các cách chứng minh khác nhau của bài toán, nhưng tất cả đều sai và thường mắc phải những lỗi khó phát hiện. Bên cạnh đó là những cố gắng vô ích trong việc phủ định giả thuyết bằng cách chỉ ra bả đồ đòi hỏi nhiều hơn 4 màu.

Có lẽ, sai lầm nổi tiếng nhất là chứng minh được xuất bản năm 1879 bởi một luật sư ở Luân Đôn và một nhà toán học nghiệp dư, Afred Kempe. Các nhà toán học công nhận chứng minh này là đúng cho đến năm 1890, khi Percy Heawood tìm ra lỗi. tuy nhiên, hướng lí luận của Kempe lại trở thành cơ sở cho thành công của Appel và Haken sau này. Chứng minh của họ dựa trên phân tích cẩn thận từng trườn hợp trên máy tính. Họ chỉ ra rằng nếu giả thuyết 4 màu sai, họ phải tìm ra được phản ví dụ của 1 trong khoảng 2000 trường hợp khác nhau. Họ đã sử dụng máy tính chạy trong 1000 tiếng để thực hiện chứng minh của mình. Chứng minh này đã gây ra nhiều tranh cãi khi sử dụng máy tính để thực hiện các thao tác chính. Ví dụ, chương trình máy tính có thể mắc lỗi dẫn đến kết quả sai. Liệu lí lẽ của họ có thật sự là 1 chứng minh khi phụ thuộc những kết qủa không đáng tin cậy của máy tính?

Định lý 4 màu chỉ ứng dụng trên đồ thị phẳng. Đồ thị không phẳng có thể số màu lớn hơn 4.

Hai điều kiện được yêu cầu để xác định số màu của một đồ thị n: Đầu tiên, phải chứng minh đồ thị có thể tô bằng n màu. Việc này có thể thực hiện bằng cách xây dựng, như tô màu. Thứ 2, chúng ta phải chỉ ra rằng đồ thị không thể tô bằng số màu ít hơn n. [5]

3.5.2. Bài toán lập lịch

3.5.2.1. Tìm hiểu chung

Lập lịch có thể được định nghĩa là một bài toán tìm kiếm chuỗi tối ưu để thực hiện một tập các hoạt động chịu tác động của một tập các ràng buộc cần phải được thỏa mãn. Người lập lịch thường cố gắng thử đến mức tối đa sự sử dụng các cá thể, máy móc và tối thiểu thời gian đòi hỏi để hoàn thành toàn bộ quá trình nhằm sắp xếp lịch.Vì thế bài toán lập lịch là một vấn đề rất khó để giải quyết.

Hiện nay có nhiều khả năng để phát triển các kỹ thuật hiện tại để giải quyết bài toán này. Những kỹ thuật đó bao gồm: các tiếp cận Trí tuệ nhân tạo như hệ thống tri thức cơ sở (knowledge-based systems), bài toán thoả mãn ràng buộc, hệ chuyên gia, mạng Nơron và các tiếp cận của các Nghiên cứu hoạt động: lập trình tính toán, lập trình động, tìm kiếm nhánh và đường biên, kỹ thuật mô phỏng, tìm kiếm Tabu và phương pháp nút cổ chai. [5]

3.5.2.2. Các đặc tính của bài toán lập lịch

Tài nguyên: đó là các nguồn dữ liệu đầu vào của bài toán. Các tài nguyên này có thể phục hồi hoặc không.

Tác vụ: được đánh giá qua các tiêu chuẩn thực hiện như thời gian thực hiện, chi phí, mức tiêu thụ nguồn tài nguyên. [5]

Ràng buộc: đây là những điều kiện cần thỏa mãn để bài toán có thể đưa ra lời giải tốt nhất.

Mục tiêu: đánh giá độ tối ưu của lịch trình lời giải của bài toán.

Khi các mục tiêu được thỏa mãn thì các ràng buộc cũng phải được thỏa mãn.

3.6. Giới thiệu sơ lược về ngôn ngữ lập trình Python

3.6.1. Giới thiệu

Python là một ngôn ngữ lập trình thông dịch do Guido van Rossum (Hiện tại Guido van Russum đang làm việc cho Google) tạo ra năm 1990. Python hoàn toàn tạo kiểu động và dùng cơ chế cấp phát bộ nhớ tự động; do vậy nó tương tự như Perl, Ruby, Scheme, Smaltalk và Tcl. Python được phát triển trong một dự án mã mở, do tổ chức phi lợi nhuận Python Softwase Foundation quản lý.

Theo đánh giá của Eric S.Raymond, Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu, như nhận định của chính Guido van Rossum trong bài phỏng vấn ông (Training Java Script & MVC 01/06/2013 (W4)).

Ban đầu, Python được phát triển để chạy trên nền Unix. Nhưng rồi theo thời gian, nó đã "bành trướng" sang mọi HĐH từ MS-DOS đến Mac OS, OS/2, Windows, Linux và các HĐH khác thuộc họ Unix. Mặc dù sự phát triển của Python có sự đóng góp của rất nhiều cá nhân, nhưng Guido van Rossum hiện nay vẫn là tác giả chủ yếu của

Python. Ông giữ vai trò chủ thốt trong mọi việc quyết định hướng phát triển của Python. Hiện tại Python đang được ứng dụng như sau:

- Google sử dụng Python vào web search system.
- YouTube dịch vụ chia sẻ video số 1 thế giới phần lớn viết bằng Python.
- Hệ thống Bit-Torrent P2P là một Python Program.
- Intrel, Cisco, HP, IBM… sử dụng Python để dùng vào quá trình hardware-tesing.
- Pixar hãng phim hoạt hình nổi tiếng sử dụng Python vào việc Production of movie amination.
- NASA sử dụng Python vào scientific programming tasks.
- Open ERD.

Và còn nhiều nữa… [18]

3.6.2. Lịch sử hình thành

❖ Sự phát triển Python đến nay có thể chia làm các giai đoạn:

Python 1 (12/1989): bao gồm các bản phát hành 1.x. Giai đoạn này, kéo dài từ đầu đến cuối thập niên 1990. Từ năm 1990 đến 1995, Guido làm việc tại CWI (Centrum voor Wiskunde en Informatica) – Trung tâm Toán –Tin học tại Amsterdam, Hà Lan). Phiên bản cuối cùng phát hành tại CWI là 1.2.

Vào năm 1995, Guido chuyển sang CNRI (Corporation for National Research Initiatives) ở Reston, Virginia. Tại đây, ông phát hành một số phiên bản khác. Python 1.6 là phiên bản cuối cùng phát hành tại CNRI.

Sau bản phát hành 1.6, Guido rời bỏ CNRI để làm việc với các lập trình viên chuyên viết phần mềm thương mại. Tại đây, ông có ý tưởng sử dụng Python với các phần mềm tuân theo chuẩn GPL. Sau đó, CNRI và FSF (Free Software Foundation – Tổ chức phần mềm tự do) đã cùng nhau hợp tác để làm bản "quyền Python phù hợp với GPL. Cùng năm đó, Guido được nhận giải thưởng FSF vì sự phát triển tự do (Award for the Advancement of Free Software). Phiên bản 1.6.1 ra đời sau đó là phiên bản đầu tiên tuân theo bản quyền GPL. Tuy nhiên, bản này hoàn toàn giống bản 1.6, trừ một số lỗi cần thiết.

Python 2 (16/10/2000): vào năm 2000, Guido và nhóm phát triển Python dời đến BeOpen.com và thành lập BeOpen PythonLabs team. Phiên bản Python 2.0 được phát

hành tại đây. Sau khi phát hành Python 2.0, Guido và các thành viên PythonLabs gia nhập Digital Creations. Python 2.1 ra đời kế thừa từ Python 1.6.1 và Python 2.0. Bản quyền của phiên bản này được đổi thành Python Software Foundation License. Từ thời điểm ày trở đi, Python thuộc sở hữ Python Software Foundation (PSF), một tổ chức phi lợi nhuận được thành lập theo mẫu Apache Software Foundation.

Python 3 (03/12/32008): còn gọi là Python 3000 hoặc Py3K: Dòng 3.x sẽ không hoàn toàn tương thích với dòng 2.x, tuy vậy có công cụ hỗ trợ chuyển đổi từ các phiên bản 2.x sang 3.x. Nguyên tắc chủ đạo để phát triển Python 3.x là "bỏ cách làm việc cũ nhằm hạn chế trùng lặp về mặt chứ năng của Python". Trong PEP (Python Enhancement Proposal) có mô tả chi tiết các thay đổi trong Python. [18]

3.7. Giới thiệu về P-Center

Các vấn đề P-center là để xác định vị trí các cơ sở p trong một mạng lưới để giảm thiểu khoảng cách xa nhất giữa một tập n điểm nhu cầu và các cơ sở p. Vấn đề này là Trung ương đến các lĩnh vực lý thuyết vị trí và hậu cần và đã chịu mở rộng nghiên cứu. Đối với tài liệu tham khảo trong vấn đề p-center [9;10]

Chúng tôi có mô hình mạng như một đồ thị G = (V, E), trong đó V = {v_1, v_2,…, v_n} đỉnh thiết lập với |V| n = và E là cạnh thiết với |E| = m . Chúng tôi giả định rằng nhu cầu điểm trùng với các đỉnh, và hạn chế các vị trí của các cơ sở để các đỉnh. Mỗi đỉnh v_1 có w_1 trọng lượng và các cạnh của đồ thị có trọng lượng tích cực. Để cho d(u,v) là chiều dài của con đường trọng ngắn nhất giữa các đỉnh u và v. Trong P-center là vấn đề chúng ta muốn tìm một tập con X ⊆ V của cardinality p như vậy mà tối đa khoảng cách từ trọng X để tất cả các đỉnh được giảm thiểu. Nói cách khác, chúng ta muốn tìm một tập con X ⊆ V của cardinality p mà F(X) = max w_id (X, v_i) được giảm thiểu.

Chúng tôi gọi một đồ thị G tam giác-miễn phí nếu G không chứa bất kỳ hình tam giác như một đồ thị con gây ra. Đồ thị tam giác-miễn phí là một lớp học của đồ thị được nghiên cứu và đóng vai trò quan trọng trong lý thuyết đồ thị. Nhiều người trong số các thông số lý thuyết đồ thị đối phó với tam giác-miễn phí đồ thị. Để xem một số kết quả trên đồ thị tam giác-miễn phí. Tuy nhiên, xác định các vấn đề vị trí trong đồ thị tam giác-miễn phí mở cửa.

Một trong những câu hỏi liên quan đến vấn đề vị trí là làm thế nào để mở rộng một nontrivially đồ thị G để một đồ thị lớn hơn với các giải pháp tối ưu cùng cho vấn đề p - center. Chúng tôi sẽ trình bày một công trình không tầm thường. Chúng tôi cung cấp một xây dựng trên một đồ thị G mà tạo ra một chuỗi vô hạn $G = G_0 \leq G_1 \leq G_2 \leq \ldots$ của đồ thị có chứa G như vậy mà cho một được tối ưu giải pháp X cho vấn đề p - center trên G, X là một giải pháp tối ưu cho vấn đề p – center trên G_i cho bất kỳ $i \geq 1$. Nếu G có n đỉnh và m cạnh, chúng tôi xây dựng tạo ra một biểu đồ M (G) với đỉnh 2n và 2m cạnh. Hơn nữa, nếu G là tam giác-miễn phí, sau đó M (G) là tam giác-miễn phí. Lưu ý rằng bằng $G_i \leq G_j$ chúng tôi có nghĩa là G_i là một đồ thị con của G_j. Xây dựng này sản xuất lớn và đồ thị lớn hơn (trường hợp) mà có giải pháp tối ưu giống như đồ thị ban đầu G. Điều này cho phép chúng tôi mở rộng một đồ thị với các giải pháp đưa ra tối ưu cho vấn đề p - center để một đồ thị lớn mà không có bất kỳ tính toán hơn nữa để tìm một giải pháp cho vấn đề p -center.

Tất cả các đồ thị xử lý trong báo cáo này là kết nối và vô hướng. Chúng ta nhớ lại rằng mở hàng xóm của một υ đỉnh trong một đồ thị G được ký hiệu là N (v) hoặc N_G (v) G để tham khảo G, do đó N(v) = $u \in V\ uv \in E$. Ngoài ra, lưu ý rằng bằng cách "giải pháp tối ưu", chúng tôi có nghĩa là một giải pháp tốt nhất có thể cho vấn đề của chúng tôi. Vì vậy, nếu X và Y là hai giải pháp tối ưu cho các vấn đề p - center và F là hàm mục tiêu, sau đó F (X) = F(Y). [11]

CHƯƠNG 4. DỮ LIỆU VÀ PHƯƠNG PHÁP NGHIÊN CỨU

4.1. Dữ liệu thu thập

Dữ liệu phục vụ cho nghiên cứu đề tài bao gồm các dữ liệu sau:

- Dữ liệu thô: Lớp giao thông, lớp bản đồ nền.
- Dữ liệu xử lý: Cây

Dữ liệu	Mô tả	Nguồn
Lớp đường giao thông	Dữ liệu dạng vector (polyline), gồm các tuyến đường trong quận	http://download.geofabrik.de/asia/vietnam.html Download OpenStreetMap data for this region: VietNam
Lớp bản đồ nền (hành chính)	Dữ liệu dạng vùng (polygon)	http://www.diva-gis.org/gdata DIVA-GIS
Cây	Dữ liệu dạng điểm (point)	Good Map và quan sát thực tế

Bảng 4.1. Thông tin các lớp dữ liệu

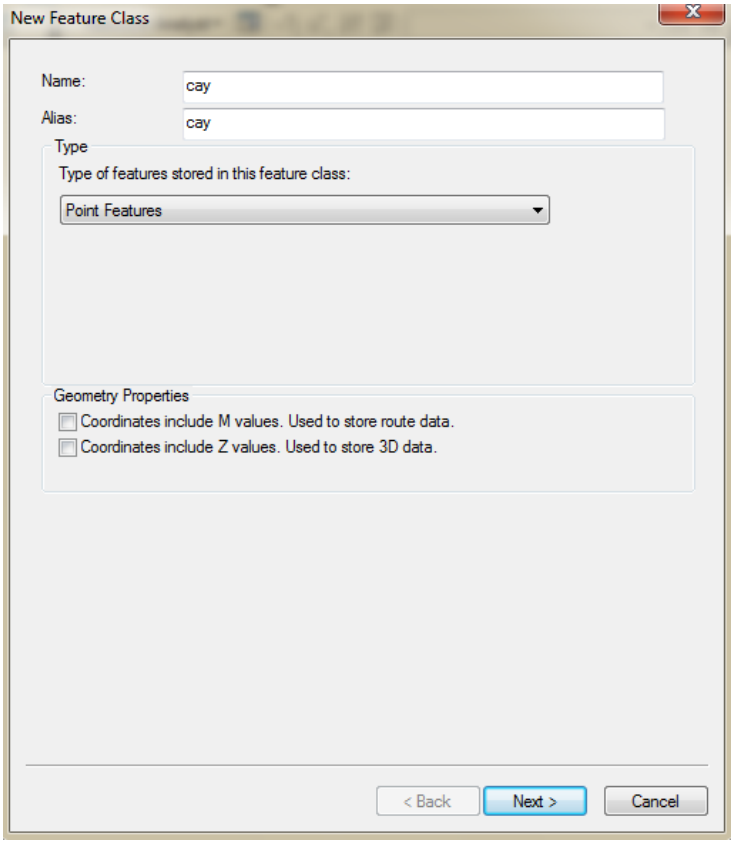
4.2. Phương pháp nghiên cứu

Đề tài sử dụng phương pháp số hóa dữ liệu, tức là chấm điểm các cây với các thuộc tính. Và bài toán trong đề tài dựa trên việc chồng lớp các bản đồ từ phép phân tích P-center để lập được kịch bản lịch tưới cây trên đường cho quận Thủ Đức.

4.2.1. Các công cụ hỗ trợ phân tích không gian trong ArcGIS sử dụng trong nghiên cứu

❖ **Công cụ tạo điểm (cây)**

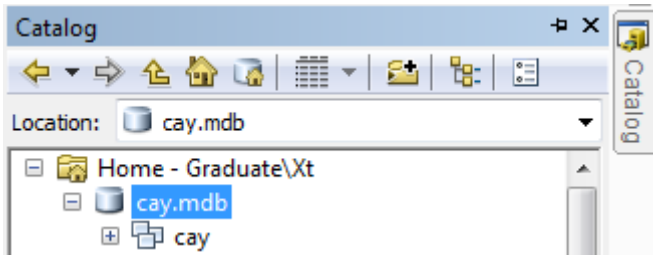
- Để xây dựng một Layer dạng point. Ta chọn Catalog để tạo Geodatabase, sau đó tiếp tục chọn New Feature Class ta sẽ được hình 4.1.



Hình 4.1. Hộp thoại New Feature Class

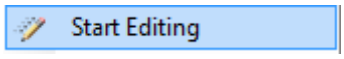
- Sau đó ta chọn hệ quy chiếu cho điểm với hệ quy chiếu là

- Sau đó ta sẽ tạo được như hình

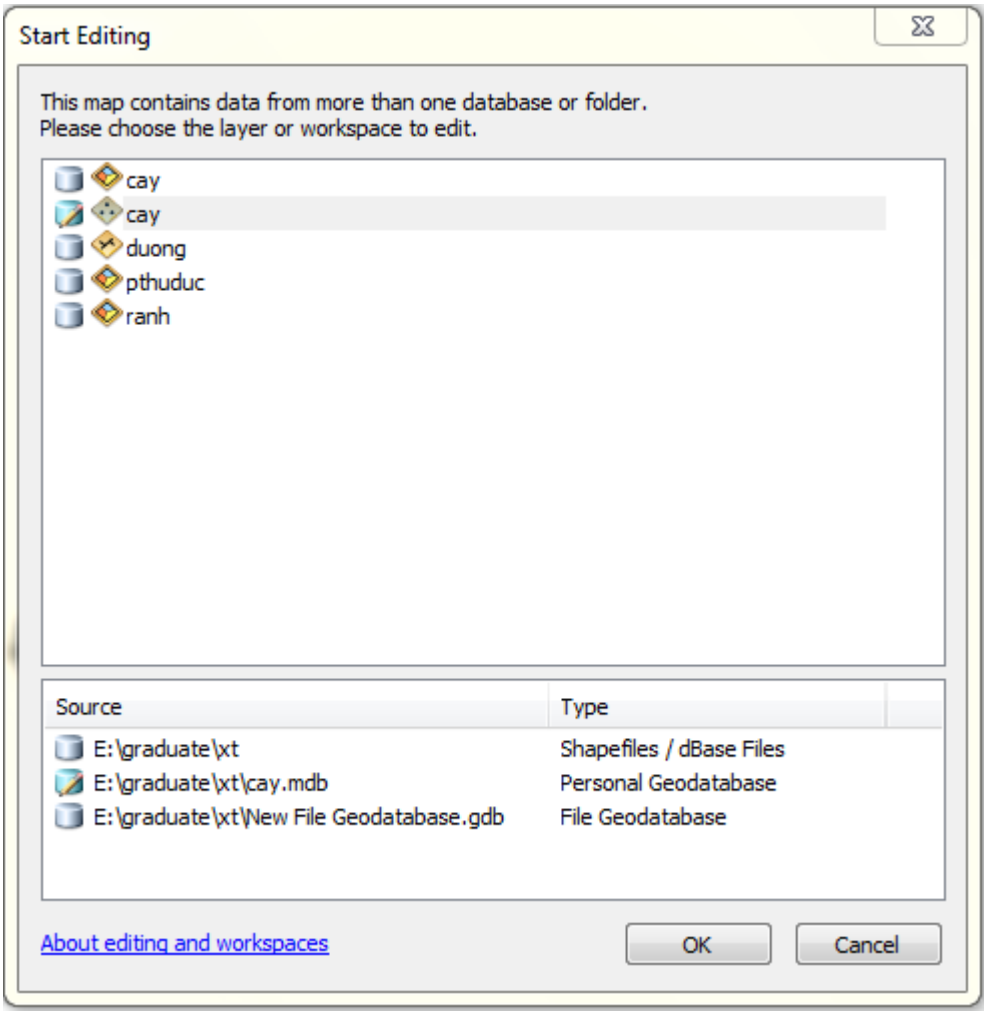


Hình 4.2. Geodatase chứa các điểm của cây

❖ **Công cụ chấm điểm (cây) thuộc tính**

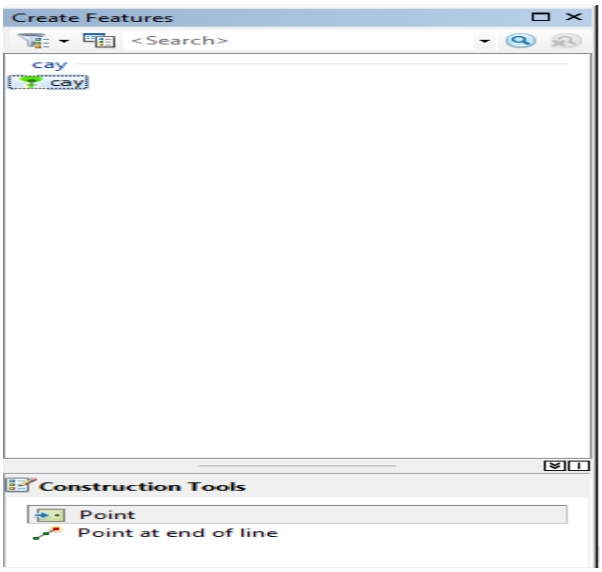
- Chọn chức năng 

(Start Editing) trên thanh Editor
- Trên thanh hộp thoại Strat Editing, hình 4.3



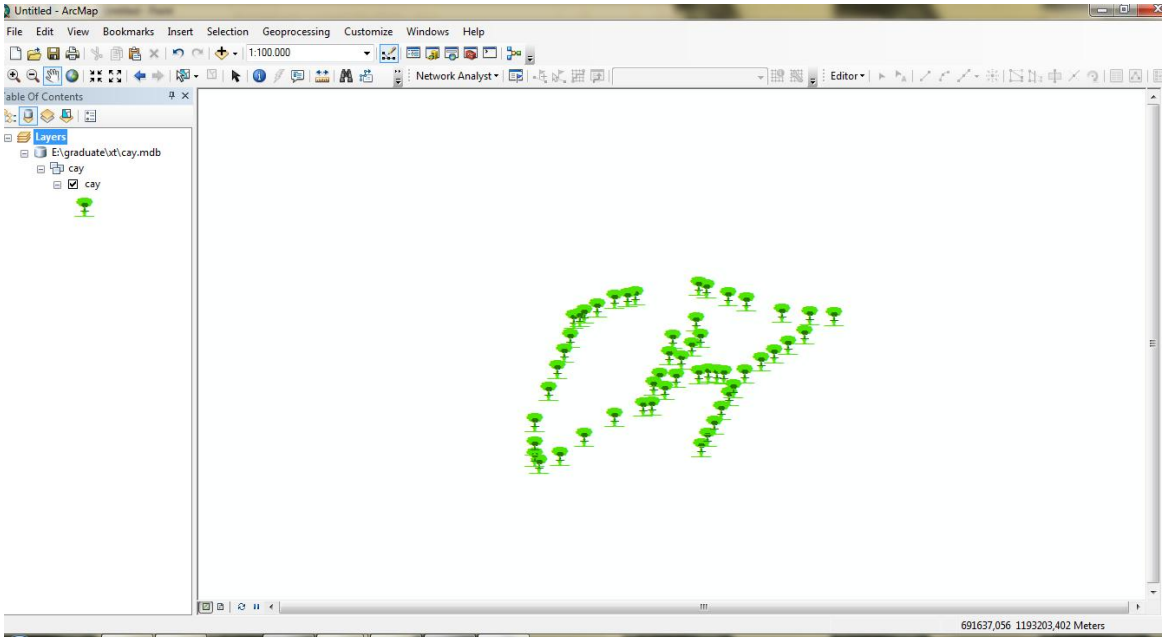
Hình 4.3. Hộp thoại Star Edting

- Nhấn nút OK sẽ xuất hiện hộp thoại hình 4.4 và chọn point để bắt đầu chấm điểm



Hình 4.4. Hộp thoại Create Features

- Sau đó ta được kết quả hình 4.5



Hình 4.5. Kết quả các điểm (cây)

4.2.2. Phương pháp phân tích của phép phân tích P-center đối với từng loại

Giả sử bài toán tưới cây là :

- x cây mỗi 1 ngày tưới 1 lần, mỗi cây cần a lượng nước.
- y cây mỗi 2 ngày tưới 1 lần, mỗi cây cần b lượng nước.
- z cây mỗi 3 ngày tưới 1 lần, mỗi cây cần c lượng nước.
- t cây mỗi 4 ngày tưới 1 lần, mỗi cây cần d lượng nước.

Hiển nhiên, tổng lượng nước cần để tưới sẽ là: ax + by + cz + td = U

Tuy nhiên, giả định chúng ta có một lượng nước W để tưới. Tất nhiên, nếu W > U thì mỗi ngày đều tất cả các cây đều được tưới.

Giả sử, W < U. Khi đó, chúng ta cần tưới cây để mỗi cây đều được tưới đều. Và đặc điểm quan trọng nhất là chi phí di chuyển tưới là thấp nhất, nghĩa là bài toán phải đáp ứng 2 yếu tố:

+ Tưới đủ: tất cả các cây đều tưới đủ (theo hạn)

+ Vùng tưới được chọn không quá to để lãng phí di chuyển làm tăng chi phí di chuyển.

Để tiết kiệm, thêm 1 ràng buộc: các cây mỗi ngày cần tưới đều được gắn vòi tưới nên không cần xe đến tưới. Do đó, bài toán còn các lại là:

Lượng nước có để tưới là: V = W - ax.

Lượng nước V đó để phân bố cho một số cây 2 ngày tưới 1 lần, 3 ngày tưới 1 lần và 4 ngày tưới 1 lần.

Nghĩa là:

Gọi y1, y2 là số cây 2 ngày tưới 1 lần: y1 + y2 = y

Gọi z1, z2, z3 là số cây 3 ngày tưới 1 lần: z1 + z2 + z3 = z

Gọi t1, t2, t3 là số cây 4 ngày tưới 1 lần: t1 + t2 + t3 + t4 = t

Khi đó, ta có các phương trình:

$b.yi + c.zi + d.ti < V$

Khi đó, sẽ có một kịch bản tưới cây.

4.3. Tiến trình thực hiện

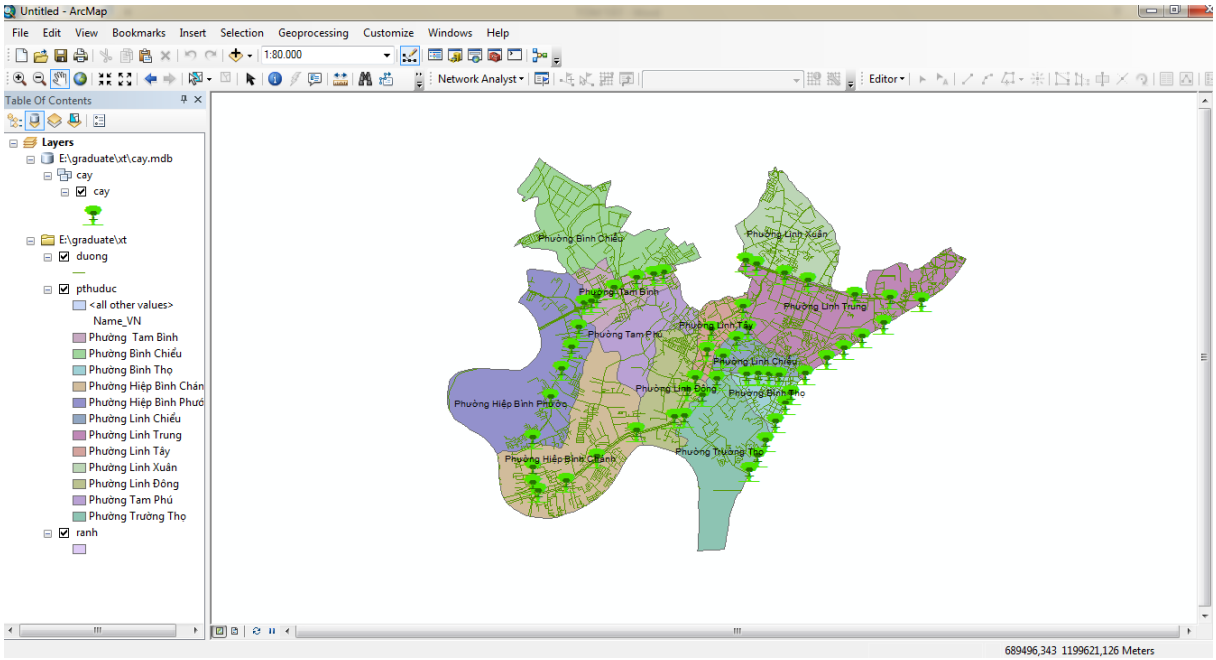
Cụ thể, quá trình nghiên cứu và thực hiện đề tài được tiến hành theo sơ đồ sau:

- Bước 1: Xác định dữ liệu đầu vào (bản đồ, vị trí cây)
- Bước 2: Từ bản đồ mới hình thành nên đồ thị
- Bước 3: Đề xuất phương án lập lịch tưới trên đồ thị đã được hình thành (Sử dụng phép phân tích P-center)
- Bước 4: Kết quả trên mới phân vùng tưới.
- Bước 5: Đưa kết quả phân tích đồ thị thể hiện vào bản đồ.

CHƯƠNG 5. KẾT QUẢ NGHIÊN CỨU

5.1. Kết quả điểm cây của quận Thủ Đức

Với phương pháp số hóa dữ liệu và các dữ liệu của quận, thì ta được bản đồ sau:



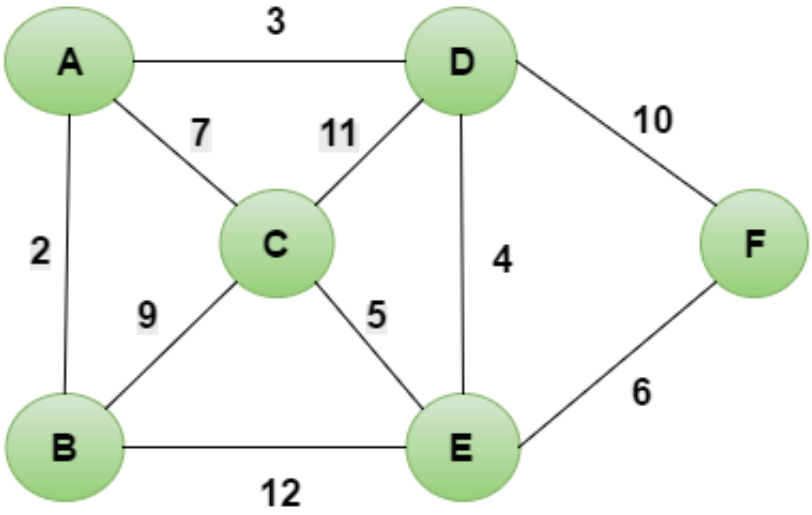
Hình 5.1. Tổng số thuộc tính của cây trên bản đồ

5.2. Lập lịch tưới cây

Đã nêu ở trên phần phương pháp về phép phân tích P-center. Ta có 3 đồ thị và 3 tập tin được suy ra từ đồ thị của các cây tưới 2 ngày/ 1 lần, 3 ngày/ 1 lần và 4 ngày/ 1 lần như sau:

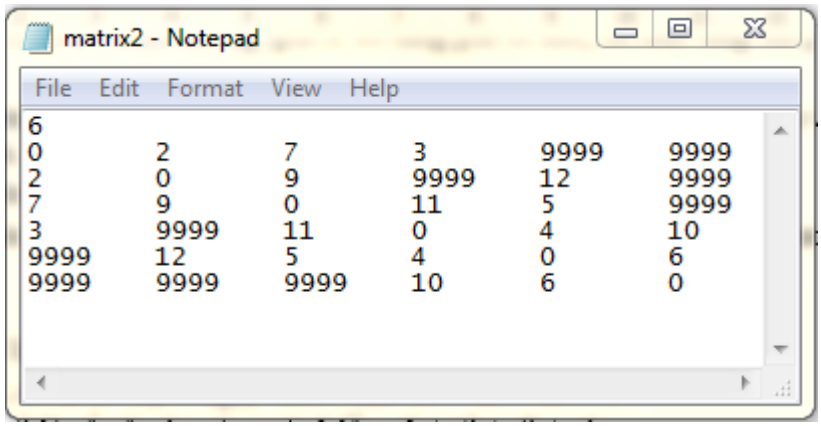
Bước 1: Tạo đồ thị và các text của ma trận

- Đối với các cây tưới 2 ngày/ 1 lần:



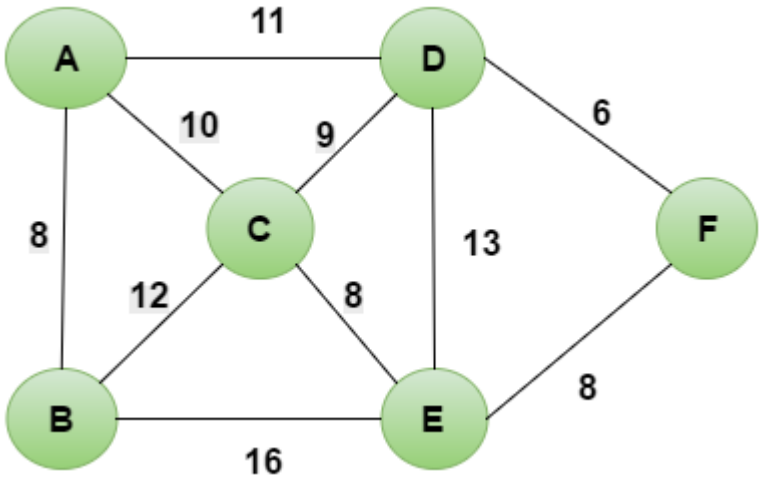
Hình 5.2. Đồ thị của các cây tưới 2 ngày/ 1 lần

Và từ đồ thị suy ra được ma trận kề



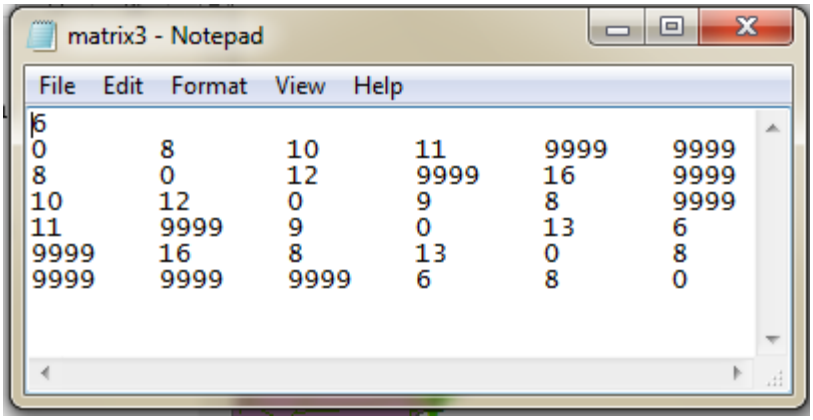
Hình 5.3. Ma trận kề của đồ thị của các cây tưới 2 ngày/ 1 lần

- Đối với các cây tưới 3 ngày/ 1 lần:



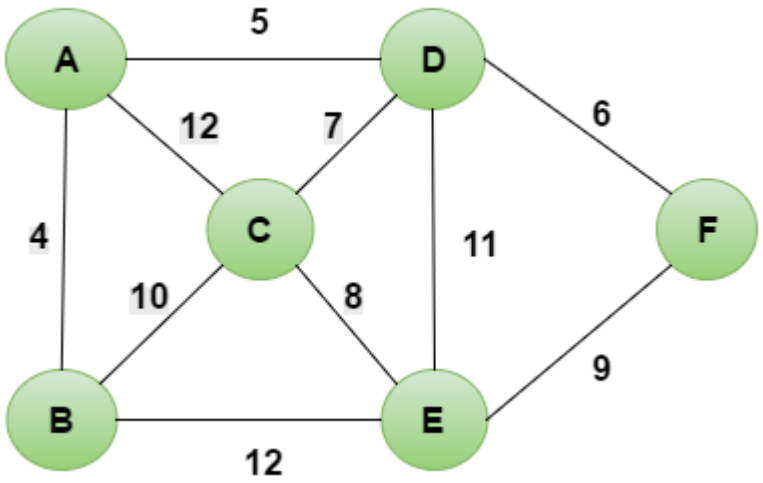
Hình 5.4. Đồ thị của các cây tưới 3 ngày/ 1 lần

Và từ đồ thị suy ra được ma trận kề



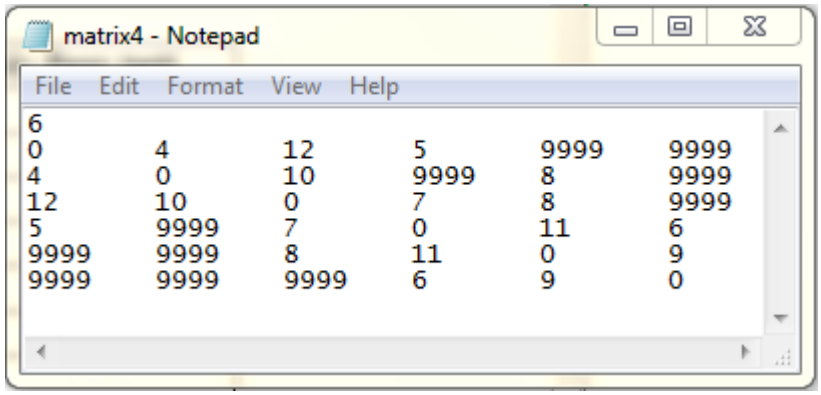
Hình 5.5. Ma trận kề của đồ thị của các cây tưới 3 ngày/ 1 lần

- Đối với các cây tưới 4 ngày / 1 lần



Hình 5.6 Đồ thị của các cây tưới 4 ngày/ 1 lần

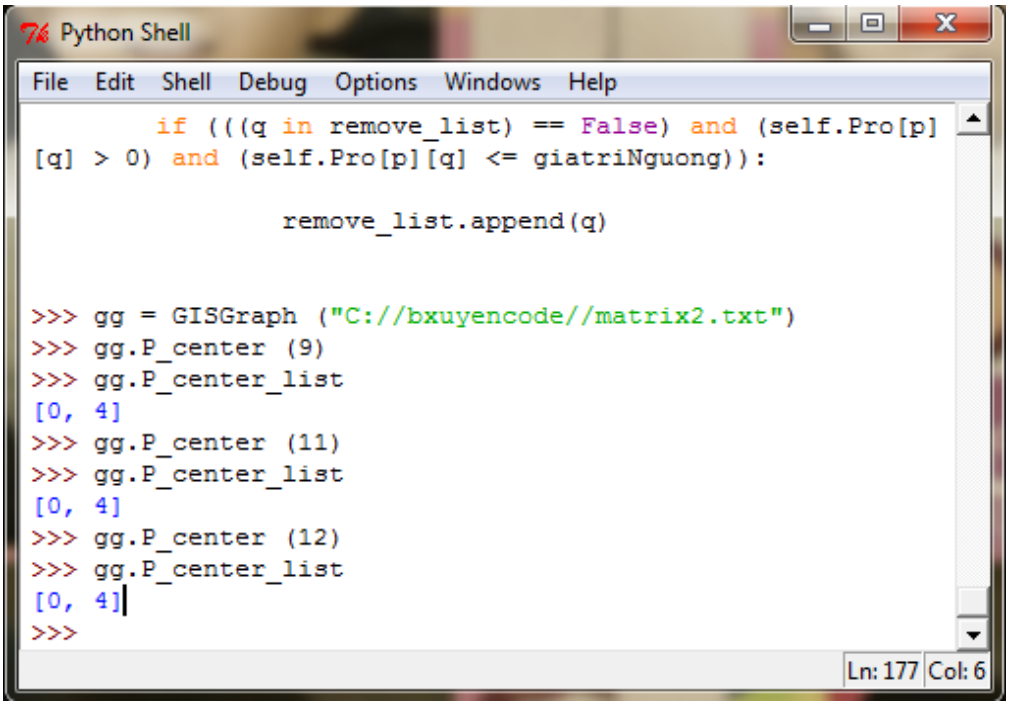
Và từ đồ thị suy ra được ma trận kề



Hình 5.7. Ma trận kề của đồ thị của các cây tưới 4 ngày/ 1 lần

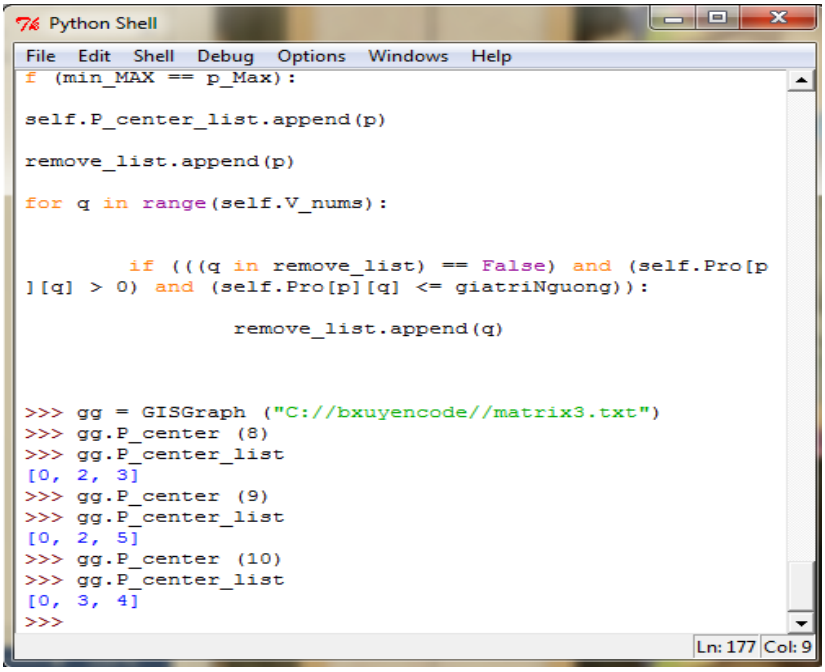
Bước 2: Đưa vào Python để thực hiện bài toán tìm giá trị phân chia đồ thị theo cây

- Với kết quả hình 5.8, ta thấy gg.P_center (9) , (11) và (12). Suy ra được với các giá trị (9), (11), (12) thì đồ thị được chia làm 2 tập đỉnh tương ứng với số các cây 2 ngày tưới / 1 lần.



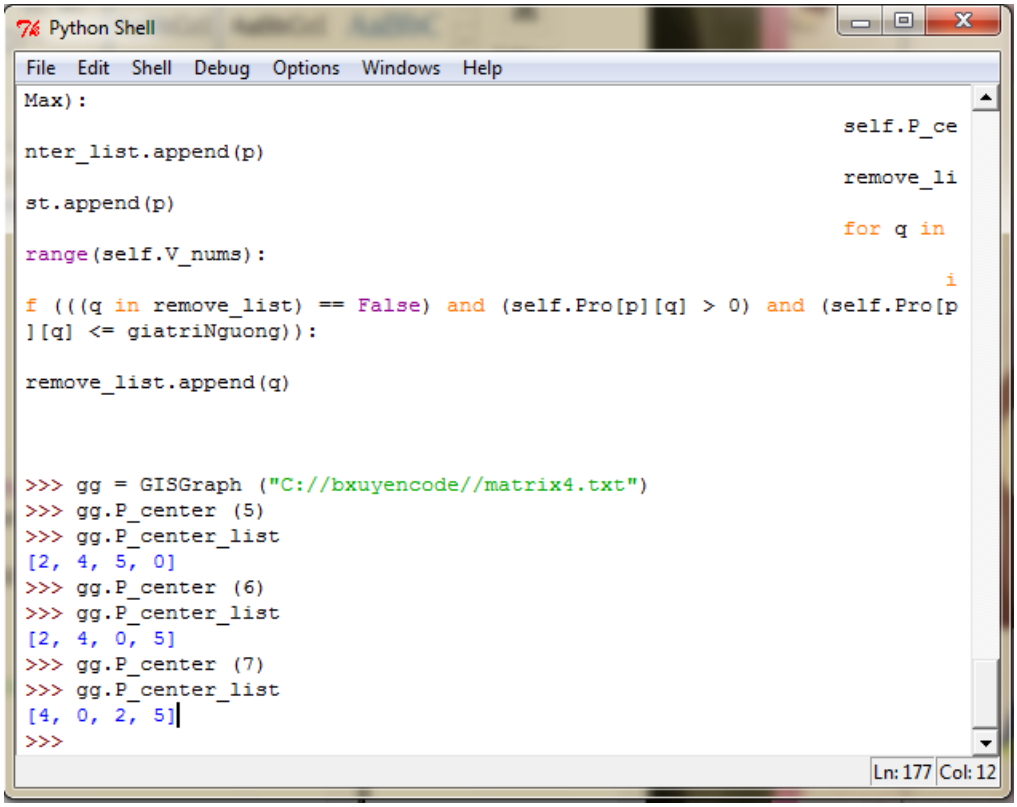
Hình 5.8. Kết quả giá trị cho đồ thị được 2 tập đỉnh

- Tương tự như vậy hình 5.10, gg.P_center (8) (9) và (10), thì giá trị ứng với số các cây 3 ngày tưới / 1 lần



Hình 5.10. Kết quả giá trị cho đồ thị được 3 tập đỉnh

- Và hình 5.11 với gg.P_center (5), (6) và (7), sẽ cho giá trị ứng với số các cây 4 ngày tưới / 1 lần.



Hình 5.11. Kết quả giá trị cho đồ thị được 4 tập đỉnh

Bước 3: Tính toán và tạo kịch bản tưới

Với kết quả ở bước 2, thì ta có thể tùy chọn 1 trong 3 giá trị ở gg.P_center, để tính toán và lập lịch. Đối với bài nghiên cứu này, chọn:

- Giá trị gg.P_center (11) đối với các cây 2 ngày tưới / 1 lần là [0, 4].
- Giá trị gg.P_center (9) đối với các cây 3 ngày tưới / 1 lần là [0, 2, 5].
- Giá trị gg.P_center (5) đối với các cây 4 ngày tưới / 1 lần là [2, 4, 5, 0].

Khi đó, ta sẽ có một kịch bản tưới như sau:

Ngày	y	z	t
1	0	0	2
2	4	2	4
3	0	5	5
4	4	0	0
5	0	2	2
6	4	5	4

7	0	0	5
8	4	2	0
9	0	5	2
10	4	0	4
11	0	2	5
12	4	5	0

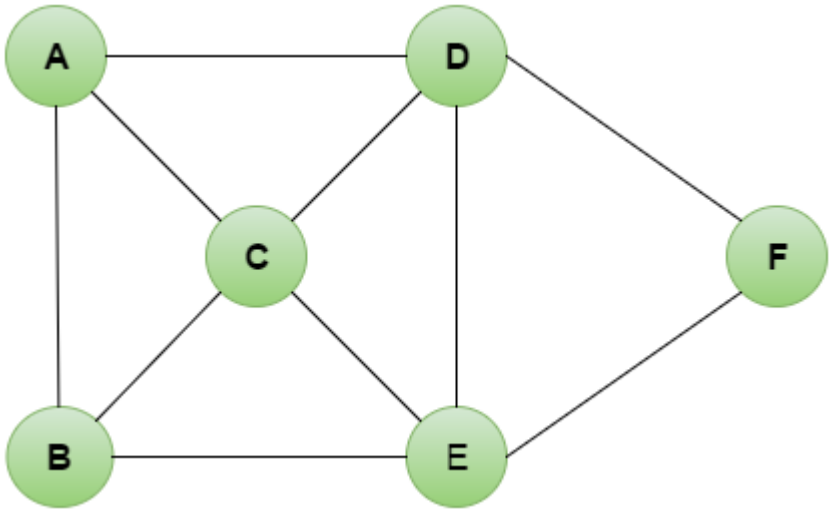
Bảng 5.1. Kết quả lập lịch tưới cây

Trong đó:

- y: là các cây 2 ngày tưới /1 lần.
- z : là các cây 3 ngày tưới /1 lần.
- t : là các cây 4 ngày tưới /1 lần.

➢ Chu kì sẽ lặp lại sau thời gian là bội số chung của 3 loại (2, 3, 4) = 12 ngày

Bước 4: Lấy kết quả của các đồ thị có các tập là 2, 3 và 4 đỉnh. Sau đó chồng lớp lên nhau sẽ được bản đồ tưới cây.



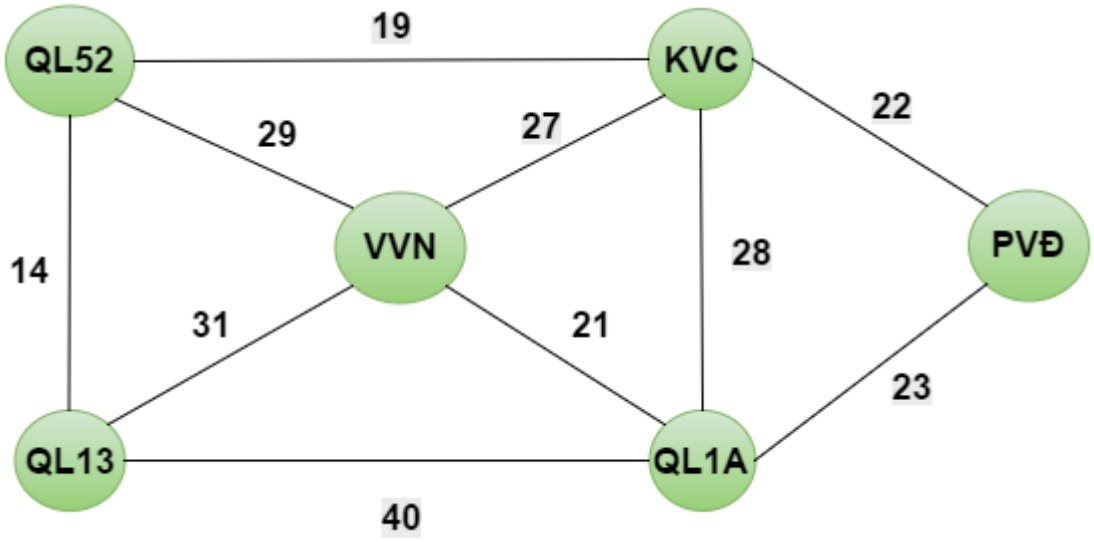
Hình 5.12. Đồ thị đỉnh tương ứng nút giao thông và cạnh là đường

Giả sử quận Thủ Đức có 6 nút giao thông chính

- Đỉnh A ứng với nút giao thông Quốc lộ 52 (QL52).
- Đỉnh B ứng với nút giao thông Quốc lộ 13 (QL13).
- Đỉnh C ứng với nút giao thông Võ Văn Ngân (VVN).
- Đỉnh D ứng với nút giao thông Kha Vạn Cân (KVC).
- Đỉnh E ứng với nút giao thông Quốc lộ 1A (QL1A).

- Đỉnh F ứng với nút giao thông Phạm Văn Đồng (PVĐ).

Ta được kết quả số cây nằm trên các cạnh ứng với các đường đi và các nút giao thông như hình 5.13



Hình 5.13. Kết quả bản đồ tưới cây.

Tóm lại: Với kết quả mới là phân vùng tưới, thì bên cạnh đó phải thực thi các thuật toán tìm đường đi. …Công cụ Netwwork (có hỗ trợ), ngoài ra để xây dựng chương trình độc lập thì có phần lý thuyết về: tìm chu trình Euler, chu trình Hamilton,…

CHƯƠNG 6. KẾT LUẬN VÀ KIẾN NGHỊ

6.1. Kết luận

Đề tài thực hiện đã được những kết quả sau:

- Hiểu được các khái niệm về lý thuyết đồ thị, thể hiện đồ thị, phép phân tích P-center và bài toán lập lịch.
- Cách viết lập trình trên ngôn ngữ Python.
- Biết được mô hình tưới cây theo lý thuyết đồ thị với quy trình lập lịch.
- Giải quyết được bài toán lập lịch cho việc tưới cây với các bố trí trên không gian đồ thị một cách hiệu quả, tiết kiệm chi phí và hợp lý cho công việc.

6.2. Kiến nghị

Do hạn chế về dữ liệu, kiến thức cũng như thời gian nên đề tài nghiên cứu còn nhiều mặt hạn chế như sau:

- Dữ liệu thu thập chưa đạt được kết quả chính xác cao, dữ liệu cây xanh chưa có độ chính xác về tọa độ không gian.
- Các thuật toán còn khá mới mẻ và lạ lẫm để ứng dụng.

Để hoàn thành bài nghiên cứu và ứng dụng một cách hiểu quả hơn cần phát triển các nội dung sau:

- Từ dữ liệu bản đồ hình thành nên đồ thị thì cần bổ sung thêm công cụ làm tự động (trong bài nghiên cứu thì chưa có công cụ nên đã sử dụng một đồ thị tương ứng các cạnh là đường đi với những con số là số lượng cây, còn các đỉnh là các nút giao thông).
- Kết quả phân tích đồ thị thể hiện vào bản đồ cũng cần xây dựng thêm công cụ tự động để hỗ trợ.

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. Nguyễn Kim Lợi và cộng tác viên, 2009. *Hệ thống thông tin Địa lý nâng cao*. Nhà xuất bản Nông nghiệp, Tp.Hồ Chí Minh, trang 5.
2. Trần Thị Mai Xinh, 2012. *Khái quát về quận Thủ Đức và nhà thiếu nhi Thủ Đức*. Báo cáo thực tập, Khoa quản lý văn hóa nghệ thuật, trường Đại Học Văn Hóa TP.Hồ Chí Minh, Việt Nam.
3. Nguyễn Hải Đăng. *Toán ứng dụng*. Trường Cao đẳng Công nghiệp Nam Định, trang 19, trang 39
4. Lê Minh Hoàng. *Lý thuyết đồ thị*. Trường Đại học Sư phạm Hà Nội, trang 4 và trang 6.
5. Hoàng Chính Nghĩa, 2009. *Tìm hiểu giải thuật di truyền ứng dụng giải bài toán lập lịch*. Báo cáo tốt nghiệp, trường Đại học Dân lập Hải Phòng, trang 5.
6. Nguyễn Thanh Hùng, Nguyễn Đức Nghĩa, 2007. *Giáo trình Lý thuyết đồ thị*, NXB Đại học Quốc Gia TPHCM.
7. Đỗ Minh Cảnh, 2014. Ứng dụng GIS hỗ trợ quản lý cây xanh tại trường Đại học Nông Lâm thành phố Hồ Chí Minh. Khóa luận tốt nghiệp, Khoa Môi trường và tài nguyên, trường Đại học Nông Lâm TP.Hồ Chí Minh, Việt Nam.
8. *Bài giảng Lý thuyết đồ thị*, 2009, Hải phòng. Khoa công nghệ thông tin, trường Đại học Hàng Hải.

Tiếng Anh

9. Bespamyatnikh, S., Bhattacharya, B., Keil, M., Kirkpatrick, D., and Segal, M., "*Efficient algorithms for center and Medians interval and circular-arc graph*", *Networks*, 39 (2002) 144-152.
10. Tamir, A., "*Improved complexity bounds for center location problems on networks by using dynamic data structures*", SIAM Journal of Discrete Mathematics, 1 (1988) 377-396.
11. Kariv, O., and Hakimi, S.L., "*An algorithmic approach to network location problem. Part I: the p-center*." SIAM J. Appl. Math., 37 (1979) 513-538.

Link truy cập

12. Đà Thành Xanh , *Tác dụng của cây xanh trong hệ sinh thái đô thị*, cập nhập 20/09/2015. Địa chỉ: <http://khoahoc.tv/tac-dung-cua-cay-xanh-trong-he-sinh-thai-do-thi-60985> (Truy cập: ngày 14 tháng 04 năm 2016).
13. Nhóm phóng viên, *Khô hạn đe dọa TP HCM*, cập nhập 06/01/2016 22:33. Địa chỉ:<http://nld.com.vn/thoi-su-trong-nuoc/kho-han-de-doa-tp-hcm-2016010622300818.htm> (Truy cập: ngày 17 tháng 04 năm 2016).
14. *Tổng quan quận Thủ Đức*. Địa chỉ : <http://www.thuduc.hochiminhcity.gov.vn/pages/gioi-thieu-tong-quat.aspx> (Truy cập: ngày 19 tháng 04 năm 2016)
15. Bản đồ quận Thủ Đức của TP.HCM (cập nhập 10/06/2007 22:00). Địa chỉ: <http://diaoconline.vn/tin-tuc/ban-do-thanh-pho-c20/ban-do-quan-thu-duc-cua-tp-hcm-i324> (Truy cập ngày 20 tháng 04 năm 2016)
16. Nguyễn Thùy Linh, *Cơ sở dữ liệu GIS* (Cập nhập ngày 31 tháng 07năm 2015) Địa chỉ: < http://tracdiapro.com/co-so-du-lieu-gis/> (Truy cập: ngày 01 tháng 05 năm 2016).
17. Giới thiệu sơ lược về Python (Cập nhập 16/07/2013). Địa chỉ: <iscclub.uit.edu.vn/news/?p=328 > (Truy cập: ngày 10 tháng 05 năm 2016).
18. Network Analyst,

Địa chỉ: <http://www.esrivn.com/vi/Products/m1597/index.html> (Truy cập: ngày 12 tháng 05 năm 2016).

19. What is the ArcGIS Netwwork Analyst extension?. Địa chỉ: <http://desktop.arcgis.com/en/arcmap/latest/extensions/network-analyst/what-is-network-analyst-.htm#GUID-81249557-BEB5-49CC-A393-28FAB997C6EE> (Truy cập: ngày 15 tháng 05 năm 2016)
20. Tô màu đồ thị. (Truy cập ngày 16 tháng 05 năm 2016). Địa chỉ: <https://vi.wikipedia.org/wiki/T%C3%B4_m%C3%A0u_%C4%91%E1%BB%93_th%E1%BB%8B>

PHỤ LỤC

Phụ lục 1. Thông tin loại cây phổ biến trên các con đường của quận Thủ Đức

Tên cây	**Xuất xứ**	**Lượng nước tưới**
Sao đen	Lào, Việt Nam, Thái Lan, Việt Nam	Mỗi lần 2-3 lít/m^2
Sao đen	Lào, Việt Nam, Thái Lan, Việt Nam	Mỗi lần 2-3 lít/m^2
Sao đen	Lào, Việt Nam, Thái Lan, Việt Nam	Mỗi lần 2-3 lít/m^2
Sao đen	Lào, Việt Nam, Thái Lan, Việt Nam	Mỗi lần 2-3 lít/m^2
Sao đen	Lào, Việt Nam, Thái Lan, Việt Nam	Mỗi lần 2-3 lít/m^2
Sao đen	Lào, Việt Nam, Thái Lan, Việt Nam	Mỗi lần 2-3 lít/m^2
Sao đen	Lào, Việt Nam, Thái Lan, Việt Nam	Mỗi lần 2-3 lít/m^2
Sao đen	Lào, Việt Nam, Thái Lan, Việt Nam	Mỗi lần 2-3 lít/m^2
Sao đen	Lào, Việt Nam, Thái Lan, Việt Nam	Mỗi lần 2-3 lít/m^2
Hoa sữa	Khu vực châu Á	Mỗi lần 1-2 lít/m^2
Hoa sữa	Khu vực châu Á	Mỗi lần 1-2 lít/m^2
Hoa sữa	Khu vực châu Á	Mỗi lần 1-2 lít/m^2
Hoa sữa	Khu vực châu Á	Mỗi lần 1-2 lít/m^2
Hoa sữa	Khu vực châu Á	Mỗi lần 1-2 lít/m^2
Giáng hương	Mianma, Lào, Thái Lan, Việt Nam và Ấn Độ	Mỗi lần 0,5-1,5 lít/m^2

Giáng hương	Mianma, Lào, Thái Lan, Việt Nam và Ấn Độ	Mỗi lần 0,5-1 lít/m^2
Giáng hương	Mianma, Lào, Thái Lan, Việt Nam và Ấn Độ	Mỗi lần 0,5-1 lít/m^2
Giáng hương	Mianma, Lào, Thái Lan, Việt Nam và Ấn Độ	Mỗi lần 0,5-1 lít/m^2
Giáng hương	Mianma, Lào, Thái Lan, Việt Nam và Ấn Độ	Mỗi lần 0,5-1 lít/m^2
Dầu rái	Đông Nam châu Á	Mỗi lần 1,5-3 lít/m^2
Dầu rái	Đông Nam châu Á	Mỗi lần 1,5-3 lít/m^2
Dầu rái	Đông Nam châu Á	Mỗi lần 1,5-3 lít/m^2
Dầu rái	Đông Nam châu Á	Mỗi lần 1,5-3 lít/m^2
Dầu rái	Đông Nam châu Á	Mỗi lần 1,5-3 lít/m^2
Dầu rái	Đông Nam châu Á	Mỗi lần 1,5-3 lít/m^2
Dầu rái	Đông Nam châu Á	Mỗi lần 1,5-3 lít/m^2
Móng bò	Trung Quốc, Ấn Độ, Mianma	Mỗi lần 0,5-1 lít/m^2
Móng bò	Trung Quốc, Ấn Độ, Mianma	Mỗi lần 0,5-1 lít/m^2
Móng bò	Trung Quốc, Ấn Độ, Mianma	Mỗi lần 0,5-1 lít/m^2
Móng bò	Trung Quốc, Ấn Độ, Mianma	Mỗi lần 0,5-1 lít/m^2
Móng bò	Trung Quốc, Ấn Độ, Mianma	Mỗi lần 0,5-1 lít/m^2
Móng bò	Trung Quốc, Ấn Độ, Mianma	Mỗi lần 0,5-1 lít/m^2
Móng bò	Trung Quốc, Ấn Độ, Mianma	Mỗi lần 0,5-1 lít/m^2
Móng bò	Trung Quốc, Ấn Độ, Mianma	Mỗi lần 0,5-1 lít/m^2
Móng bò	Trung Quốc, Ấn Độ, Mianma	Mỗi lần 0,5-1 lít/m^2
Móng bò	Trung Quốc, Ấn Độ, Mianma	Mỗi lần 0,5-1 lít/m^2
Móng bò	Trung Quốc, Ấn Độ, Mianma	Mỗi lần 0,5-1 lít/m^2
Viết	Châu Á	Mỗi lần 1-1,5 lít/m^2
Viết	Châu Á	Mỗi lần 1-1,5 lít/m^2
Viết	Châu Á	Mỗi lần 1-1,5 lít/m^2
Viết	Châu Á	Mỗi lần 1-1,5 lít/m^2
Viết	Châu Á	Mỗi lần 1-1,5 lít/m^2

Viết	Châu Á	Mỗi lần 1-1,5 lít/m^2
Bàng Đài Loan	Bahamas	Mỗi lần 1-2,5 lít/m^2
Bàng Đài Loan	Bahamas	Mỗi lần 1-2,5 lít/m^2
Bàng Đài Loan	Bahamas	Mỗi lần 1-2,5 lít/m^2
Bàng Đài Loan	Bahamas	Mỗi lần 1-2,5 lít/m^2
Bàng Đài Loan	Bahamas	Mỗi lần 1-2,5 lít/m^2
Bàng Đài Loan	Bahamas	Mỗi lần 1-2,5 lít/m^2
Bàng Đài Loan	Bahamas	Mỗi lần 1-2,5 lít/m^2
Bàng Đài Loan	Bahamas	Mỗi lần 1-2,5 lít/m^2
Bàng Đài Loan	Bahamas	Mỗi lần 1-2,5 lít/m^2

Phụ lục 2. Nội dung một số file trong phần mềm

- **Nội dung file để tìm giá trị cho đồ thị để được 2 tập đỉnh**

```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import os
>>> import sys
>>> import math
>>> class GISGraph:
        V_nums = 0  # so dinh cua do thi (Gph)
        E_set = [] # tap canh cua do thi
        Gph = []  # Ma tran do thi: V_nums x V_nums
        Pro = []  # Ma tran luy thua: V_nums x V_nums
        weight_set = []
        P_center_list = []
        graphFile = "matrix3.txt"
        max_Value = sys.maxint

        def __init__(self, sfilename):
                if sfilename == "":
                        self.graphFile = "matrix3.txt"
                else:
                        self.graphFile = sfilename
                try:
                        f = open(self.graphFile)
                        self.V_nums = int(f.readline())          # doc so dinh cua do
thi:
                        maxtrix_dat = f.readlines()
                except IOError as e:
                        print "Khong doc duoc tap tin"
                except ValueError:
                        print "Khong the doc du lieu so"
                except:
```

```
                print "Khong biet loi gi!"
                raise

        ### Tao ma tran va doc do thi:
        self.Gph = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)]
# ma tran ke V_nums x V_nums
        self.Pro = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)] #
ma tran ke V_nums x V_nums

        for p in range(self.V_nums):
                matrix_matrix = maxtrix_dat[p].split("     ")
                for q in range(self.V_nums):
                        self.Gph[p][q] = int(matrix_matrix[q])
        f.close()

        # Dat gia tri ban dau cho ma tran tich Pro = ma tran Gph
        self.Pro = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)] #
ma tran ke V_nums x V_nums
        for p in range(self.V_nums):
                for q in range(self.V_nums):
                        if (p == q):
                                self.Pro[p][q] = 0
                        else:
                                if self.Gph[p][q] == 9999:
                                        self.Pro[p][q] = sys.maxint
                                else:
                                        self.Pro[p][q] = self.Gph[p][q]


    def HienthiMatranPro(self): # Hien thi ma tran Tinh toan duong di
        for p in range(self.V_nums):
                prn_str = ""
```

```
                for q in range(self.V_nums):
                        if (self.Pro[p][q] == sys.maxint):
                                prn_str = prn_str +  " VC"
                        else:
                                prn_str = prn_str + " " + str(self.Pro[p][q]) + " "
                print prn_str + "\n"

        def HienthiMatranGph(self): # Hien thi ma tran do thi Gph
                for p in range(self.V_nums):
                        prn_str = ""
                        for q in range(self.V_nums):
                                if (self.Gph[p][q] == sys.maxint):
                                        prn_str = prn_str +  " VC "
                                else:
                                        prn_str = prn_str + str(self.Gph[p][q]) + " "
                        print prn_str + "\n"


        # Tim duong di ngan nhat:
        def TinhToanDuongDiNganNhat(self):  # Ham nay tra ve 0 neu khong doi, tra ve 1 neu co it nhat 1 gia tri doi.
                chgVal = 0    # Gia tri thay doi. Neu co thay doi thi chgVal = 1

                # khai bao ma tran tam 1
                ProTemp1 = []
                ProTemp1 = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)]
                for p in range(self.V_nums):
                        for q in range(self.V_nums):
                                ProTemp1[p][q] = self.Pro[p][q]

                # khai bao ma tran tam 2
```

```
            ProTemp2 = []
            ProTemp2 = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)]
            for p in range(self.V_nums):
                for q in range(self.V_nums):
                    ProTemp2[p][q] = self.Pro[p][q]

            # Tinh ma tran luy thua:
            # self.Pro = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)] # khong can thiet vi dinh nghia trong ham init
            for p in range(self.V_nums):
                for q in range(self.V_nums):
                    myMin = sys.maxint
                    for k in range(self.V_nums):
                        if ((ProTemp1[p][k] >= 0) and (ProTemp2[k][q] >= 0)):   # Do thi co huong
                            proVal = ProTemp1[p][k] + ProTemp2[k][q] # Gia tri tinh toan
                            myMin = min(myMin, proVal)
                    if self.Pro[p][q] != myMin:
                        chgVal = 1
                    self.Pro[p][q] = myMin
            return chgVal # Neu co thay doi, thi gia tri se > 0; nguoc lai gia tri tra ve la 0 (i.e. khong doi).

        def P_center(self, giatriNguong):  # tim cac ta^m P-center thoa tu ta^m di den cac dinh khac < giatriNguong
            self.weight_set = [0 for i in range(self.V_nums * (self.V_nums -1) )]  # tap cac trong so, co n*(n-1)
            self.P_center_list = []
            # doc du lieu tu ma tran Pro vao: doc tat ca n(n-1) gia tri (khong doc gia tri duong cheo = 0)
```

```
idx = -1
for p in range(self.V_nums):
    for q in range(self.V_nums):
        if (p != q):     # <-- loai bo duong cheo bang kiem tra nay
            idx = idx + 1
            self.weight_set[idx] = self.Pro[p][q]

self.weight_set.sort()
self.weight_set = list(set(self.weight_set))
# print self.weight_set

if (self.weight_set[0] > giatriNguong): # Neu gia tri nguong qua thap thi moi dinh deu la tam
    self.P_center_list = [0 for i in range(self.V_nums)]       # tap dinh = toan bo tap dinh
    for p in range(self.V_nums):
        self.P_center_list[p] = p
else:
    if (self.weight_set[len(self.weight_set)-1] < giatriNguong):     # Neu giaTriNguong qua qua cao thi lay 1 dinh bat ki:
        self.P_center_list = [0]
    else:    # truong hop nguong nam trong gia tri mang:
        remove_list = []
        #Chon cac dinh co min ket noi > giatriNguong dua vao list:
        for p in range(self.V_nums):
            min_p = sys.maxint # gia tri be nhat cua dinh P
            for q in range(self.V_nums):
                if ((self.Pro[p][q] > 0) and (self.Pro[p][q] < min_p)):
                    min_p = self.Pro[p][q]
            if (min_p > giatriNguong): # khi min gia tri ma van lon hon thi chon dinh do
```

```
                    self.P_center_list.append(p)
                    remove_list.append(p)

            # con lai chon cac dinh co bac cao va dua cac dinh phu thuoc vao trong remove_list:
            while (len(remove_list) < self.V_nums):
                min_MAX = sys.maxint
                p_Max = 0
                for p in range(self.V_nums):
                    if ( (p in remove_list) == False):  # neu p chua bi remove thi xet p chon gia tri thap nhat
                        for q in range(self.V_nums):

                            if (((q in remove_list) == False) and (self.Pro[p][q] > p_Max)):
                                p_Max  = self.Pro[p][q]
                        if (min_MAX > p_Max):
                            min_MAX = p_Max
                p_Max = 0
                for p in range(self.V_nums):
                    if ( (p in remove_list) == False):  # neu p chua bi remove thi xet p chon gia tri thap nhat
                        for q in range(self.V_nums):

                            if (((q in remove_list) == False) and (self.Pro[p][q] > p_Max)):
                                p_Max  = self.Pro[p][q]
                        if (min_MAX == p_Max):
                            self.P_center_list.append(p)
                            remove_list.append(p)
                            for q in range(self.V_nums):
```

```
                                                    if (((q in remove_list)
== False) and (self.Pro[p][q] > 0) and (self.Pro[p][q] <= giatriNguong)):

        remove_list.append(q)


>>> gg = GISGraph ("C://bxuyencode//matrix2.txt")
>>> gg.P_center (9)
>>> gg.P_center_list
[0, 4]
>>> gg.P_center (11)
>>> gg.P_center_list
[0, 4]
>>> gg.P_center (12)
>>> gg.P_center_list
[0, 4]
>>>
```

❖ **Nội dung file để tìm giá trị cho đồ thị để được 3 tập đỉnh**

```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" for more information.
>>> import os
>>> import sys
>>> import math
>>> class GISGraph:
    V_nums = 0  # so dinh cua do thi (Gph)
    E_set = [] # tap canh cua do thi
    Gph = []  # Ma tran do thi: V_nums x V_nums
    Pro = []  # Ma tran luy thua: V_nums x V_nums
    weight_set = []
    P_center_list = []
    graphFile = "matrix3.txt"
```

```
    max_Value = sys.maxint

    def __init__(self, sfilename):
        if sfilename == "":
            self.graphFile = "matrix3.txt"
        else:
            self.graphFile = sfilename
        try:
            f = open(self.graphFile)
            self.V_nums = int(f.readline())          # doc so dinh cua do thi:
            maxtrix_dat = f.readlines()
        except IOError as e:
            print "Khong doc duoc tap tin"
        except ValueError:
            print "Khong the doc du lieu so"
        except:
            print "Khong biet loi gi!"
            raise

        ### Tao ma tran va doc do thi:
        self.Gph = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)] # ma tran ke V_nums x V_nums
        self.Pro = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)] # ma tran ke V_nums x V_nums

        for p in range(self.V_nums):
            matrix_matrix = maxtrix_dat[p].split("    ")
            for q in range(self.V_nums):
                self.Gph[p][q] = int(matrix_matrix[q])
        f.close()
```

```
        # Dat gia tri ban dau cho ma tran tich Pro = ma tran Gph
        self.Pro = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)] #
ma tran ke V_nums x V_nums
        for p in range(self.V_nums):
                for q in range(self.V_nums):
                        if (p == q):
                                self.Pro[p][q] = 0
                        else:
                                if self.Gph[p][q] == 9999:
                                        self.Pro[p][q] = sys.maxint
                                else:
                                        self.Pro[p][q] = self.Gph[p][q]


    def HienthiMatranPro(self): # Hien thi ma tran Tinh toan duong di
        for p in range(self.V_nums):
                prn_str = ""
                for q in range(self.V_nums):
                        if (self.Pro[p][q] == sys.maxint):
                                prn_str = prn_str +  " VC"
                        else:
                                prn_str = prn_str + " " + str(self.Pro[p][q]) + " "
                print prn_str + "\n"

    def HienthiMatranGph(self): # Hien thi ma tran do thi Gph
        for p in range(self.V_nums):
                prn_str = ""
                for q in range(self.V_nums):
                        if (self.Gph[p][q] == sys.maxint):
                                prn_str = prn_str +  " VC "
                        else:
                                prn_str = prn_str + str(self.Gph[p][q]) + " "
```

```
                print prn_str + "\n"


    # Tim duong di ngan nhat:
    def TinhToanDuongDiNganNhat(self):  # Ham nay tra ve 0 neu khong doi, tra ve 1 neu co it nhat 1 gia tri doi.
                chgVal = 0    # Gia tri thay doi. Neu co thay doi thi chgVal = 1

                # khai bao ma tran tam 1
                ProTemp1 = []
                ProTemp1 = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)]
                for p in range(self.V_nums):
                        for q in range(self.V_nums):
                                ProTemp1[p][q] = self.Pro[p][q]

                # khai bao ma tran tam 2
                ProTemp2 = []
                ProTemp2 = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)]
                for p in range(self.V_nums):
                        for q in range(self.V_nums):
                                ProTemp2[p][q] = self.Pro[p][q]

                # Tinh ma tran luy thua:
                # self.Pro = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)] # khong can thiet vi dinh nghia trong ham init
                for p in range(self.V_nums):
                        for q in range(self.V_nums):
                                myMin = sys.maxint
                                for k in range(self.V_nums):
```

```
                                if ((ProTemp1[p][k] >= 0) and (ProTemp2[k][q] >=
0)):   # Do thi co huong
                                        proVal = ProTemp1[p][k] + ProTemp2[k][q]
# Gia tri tinh toan
                                myMin = min(myMin, proVal)
                        if self.Pro[p][q] != myMin:
                                chgVal = 1
                        self.Pro[p][q] = myMin
        return chgVal # Neu co thay doi, thi gia tri se > 0; nguoc lai gia tri tra ve
la 0 (i.e. khong doi).

    def P_center(self, giatriNguong):  # tim cac ta^m P-center thoa tu ta^m di den
cac dinh khac < giatriNguong
        self.weight_set = [0 for i in range(self.V_nums * (self.V_nums -1) )]  #
tap cac trong so, co n*(n-1)
        self.P_center_list = []
        # doc du lieu tu ma tran Pro vao: doc tat ca n(n-1) gia tri (khong doc gia
tri duong cheo = 0)
        idx = -1
        for p in range(self.V_nums):
                for q in range(self.V_nums):
                        if (p != q):     # <-- loai bo duong cheo bang kiem tra nay
                                idx = idx + 1
                                self.weight_set[idx] = self.Pro[p][q]

        self.weight_set.sort()
        self.weight_set = list(set(self.weight_set))
        # print self.weight_set

        if (self.weight_set[0] > giatriNguong): # Neu gia tri nguong qua thap thi
moi dinh deu la tam
```

```
                self.P_center_list = [0 for i in range(self.V_nums)]      # tap dinh
= toan bo tap dinh
                for p in range(self.V_nums):
                        self.P_center_list[p] = p
        else:
                if (self.weight_set[len(self.weight_set)-1] < giatriNguong):    #
Neu giaTriNguong qua qua cao thi lay 1 dinh bat ki:
                        self.P_center_list = [0]
                else:   # truong hop nguong nam trong gia tri mang:
                        remove_list = []
                        #Chon cac dinh co min ket noi > giatriNguong dua vao list:
                        for p in range(self.V_nums):
                                min_p = sys.maxint # gia tri be nhat cua dinh P
                                for q in range(self.V_nums):
                                        if ((self.Pro[p][q] > 0) and (self.Pro[p][q] <
min_p)):
                                                min_p = self.Pro[p][q]
                                if (min_p > giatriNguong): # khi min gia tri ma van
lon hon thi chon dinh do
                                        self.P_center_list.append(p)
                                        remove_list.append(p)

                        # con lai chon cac dinh co bac cao va dua cac dinh phu
thuoc vao trong remove_list:
                        while (len(remove_list) < self.V_nums):
                                min_MAX = sys.maxint
                                p_Max = 0
                                for p in range(self.V_nums):
                                        if ( (p in remove_list) == False):   #   neu   p
chua bi remove thi xet p chon gia tri thap nhat
                                                for q in range(self.V_nums):
```

```
                                        if (((q in remove_list) ==
False) and (self.Pro[p][q] > p_Max)):
                                            p_Max  = self.Pro[p][q]
                                    if (min_MAX > p_Max):
                                        min_MAX = p_Max
                            p_Max = 0
                            for p in range(self.V_nums):
                                if ( (p in remove_list) == False):  #  neu  p
chua bi remove thi xet p chon gia tri thap nhat
                                    for q in range(self.V_nums):

                                        if (((q in remove_list) ==
False) and (self.Pro[p][q] > p_Max)):
                                            p_Max  = self.Pro[p][q]
                                    if (min_MAX == p_Max):
                                        self.P_center_list.append(p)
                                        remove_list.append(p)
                                        for q in range(self.V_nums):

                                            if (((q in remove_list)
== False) and (self.Pro[p][q] > 0) and (self.Pro[p][q] <= giatriNguong)):

    remove_list.append(q)


>>> gg = GISGraph ("C://bxuyencode//matrix3.txt")
>>> gg.P_center (8)
>>> gg.P_center_list
[0, 2, 3]
>>> gg.P_center (9)
>>> gg.P_center_list
[0, 2, 5]
```

```
>>> gg.P_center (10)
>>> gg.P_center_list
[0, 3, 4]
>>>
```

❖ **Nội dung file để tìm giá trị cho đồ thị để được 4 tập đỉnh**

```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import os
>>> import sys
>>> import math
>>> class GISGraph:
        V_nums = 0  # so dinh cua do thi (Gph)
        E_set = [] # tap canh cua do thi
        Gph = []  # Ma tran do thi: V_nums x V_nums
        Pro = []  # Ma tran luy thua: V_nums x V_nums
        weight_set = []
        P_center_list = []
        graphFile = "matrix3.txt"
        max_Value = sys.maxint

        def __init__(self, sfilename):
                if sfilename == "":
                        self.graphFile = "matrix3.txt"
                else:
                        self.graphFile = sfilename
                try:
                        f = open(self.graphFile)
                        self.V_nums = int(f.readline())          # doc so dinh cua do
thi:
                        maxtrix_dat = f.readlines()
                except IOError as e:
                        print "Khong doc duoc tap tin"
```

```
            except ValueError:
                print "Khong the doc du lieu so"
            except:
                print "Khong biet loi gi!"
                raise

            ### Tao ma tran va doc do thi:
            self.Gph = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)]
# ma tran ke V_nums x V_nums
            self.Pro = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)] #
ma tran ke V_nums x V_nums

            for p in range(self.V_nums):
                matrix_matrix = maxtrix_dat[p].split("     ")
                for q in range(self.V_nums):
                    self.Gph[p][q] = int(matrix_matrix[q])
            f.close()

            # Dat gia tri ban dau cho ma tran tich Pro = ma tran Gph
            self.Pro = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)] #
ma tran ke V_nums x V_nums
            for p in range(self.V_nums):
                for q in range(self.V_nums):
                    if (p == q):
                        self.Pro[p][q] = 0
                    else:
                        if self.Gph[p][q] == 9999:
                            self.Pro[p][q] = sys.maxint
                        else:
                            self.Pro[p][q] = self.Gph[p][q]
```

```
    def HienthiMatranPro(self): # Hien thi ma tran Tinh toan duong di
        for p in range(self.V_nums):
            prn_str = ""
            for q in range(self.V_nums):
                if (self.Pro[p][q] == sys.maxint):
                    prn_str = prn_str +  " VC"
                else:
                    prn_str = prn_str + " " + str(self.Pro[p][q]) + " "
            print prn_str + "\n"

    def HienthiMatranGph(self): # Hien thi ma tran do thi Gph
        for p in range(self.V_nums):
            prn_str = ""
            for q in range(self.V_nums):
                if (self.Gph[p][q] == sys.maxint):
                    prn_str = prn_str +  " VC "
                else:
                    prn_str = prn_str + str(self.Gph[p][q]) + " "
            print prn_str + "\n"


    # Tim duong di ngan nhat:
    def TinhToanDuongDiNganNhat(self):  # Ham nay tra ve 0 neu khong doi, tra ve 1 neu co it nhat 1 gia tri doi.
        chgVal = 0    # Gia tri thay doi. Neu co thay doi thi chgVal = 1

        # khai bao ma tran tam 1
        ProTemp1 = []
        ProTemp1 = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)]
        for p in range(self.V_nums):
            for q in range(self.V_nums):
```

```
                        ProTemp1[p][q] = self.Pro[p][q]

                # khai bao ma tran tam 2
                ProTemp2 = []
                ProTemp2 = [[0 for i in range(self.V_nums)] for j in
range(self.V_nums)]
                for p in range(self.V_nums):
                        for q in range(self.V_nums):
                                ProTemp2[p][q] = self.Pro[p][q]

                # Tinh ma tran luy thua:
                # self.Pro = [[0 for i in range(self.V_nums)] for j in range(self.V_nums)]
# khong can thiet vi dinh nghia trong ham init
                for p in range(self.V_nums):
                        for q in range(self.V_nums):
                                myMin = sys.maxint
                                for k in range(self.V_nums):
                                        if ((ProTemp1[p][k] >= 0) and (ProTemp2[k][q] >=
0)):   # Do thi co huong
                                                proVal = ProTemp1[p][k] + ProTemp2[k][q]
# Gia tri tinh toan
                                                myMin = min(myMin, proVal)
                                if self.Pro[p][q] != myMin:
                                        chgVal = 1
                                self.Pro[p][q] = myMin
                return chgVal # Neu co thay doi, thi gia tri se > 0; nguoc lai gia tri tra ve
la 0 (i.e. khong doi).

        def P_center(self, giatriNguong):  # tim cac ta^m P-center thoa tu ta^m di den
cac dinh khac < giatriNguong
                self.weight_set = [0 for i in range(self.V_nums * (self.V_nums -1) )]  #
tap cac trong so, co n*(n-1)
```

```
        self.P_center_list = []
        # doc du lieu tu ma tran Pro vao: doc tat ca n(n-1) gia tri (khong doc gia tri duong cheo = 0)
        idx = -1
        for p in range(self.V_nums):
            for q in range(self.V_nums):
                if (p != q):     # <-- loai bo duong cheo bang kiem tra nay
                    idx = idx + 1
                    self.weight_set[idx] = self.Pro[p][q]

        self.weight_set.sort()
        self.weight_set = list(set(self.weight_set))
        # print self.weight_set

        if (self.weight_set[0] > giatriNguong): # Neu gia tri nguong qua thap thi moi dinh deu la tam
            self.P_center_list = [0 for i in range(self.V_nums)]       # tap dinh = toan bo tap dinh
            for p in range(self.V_nums):
                self.P_center_list[p] = p
        else:
            if (self.weight_set[len(self.weight_set)-1] < giatriNguong):     # Neu giaTriNguong qua qua cao thi lay 1 dinh bat ki:
                self.P_center_list = [0]
            else:    # truong hop nguong nam trong gia tri mang:
                remove_list = []
                #Chon cac dinh co min ket noi > giatriNguong dua vao list:
                for p in range(self.V_nums):
                    min_p = sys.maxint # gia tri be nhat cua dinh P
                    for q in range(self.V_nums):
                        if ((self.Pro[p][q] > 0) and (self.Pro[p][q] < min_p)):
```

```
                    min_p = self.Pro[p][q]
            if (min_p > giatriNguong): # khi min gia tri ma van lon hon thi chon dinh do
                self.P_center_list.append(p)
                remove_list.append(p)

            # con lai chon cac dinh co bac cao va dua cac dinh phu thuoc vao trong remove_list:
            while (len(remove_list) < self.V_nums):
                min_MAX = sys.maxint
                p_Max = 0
                for p in range(self.V_nums):
                    if ( (p in remove_list) == False):  # neu p chua bi remove thi xet p chon gia tri thap nhat
                        for q in range(self.V_nums):

                            if (((q in remove_list) == False) and (self.Pro[p][q] > p_Max)):
                                p_Max  = self.Pro[p][q]
                        if (min_MAX > p_Max):
                            min_MAX = p_Max
                p_Max = 0
                for p in range(self.V_nums):
                    if ( (p in remove_list) == False):  # neu p chua bi remove thi xet p chon gia tri thap nhat
                        for q in range(self.V_nums):

                            if (((q in remove_list) == False) and (self.Pro[p][q] > p_Max)):
                                p_Max  = self.Pro[p][q]
                        if (min_MAX == p_Max):
                            self.P_center_list.append(p)
```

```
                                        remove_list.append(p)
                                        for q in range(self.V_nums):

                                            if (((q in remove_list)
== False) and (self.Pro[p][q] > 0) and (self.Pro[p][q] <= giatriNguong)):

        remove_list.append(q)
```

```
>>> gg = GISGraph ("C://bxuyencode//matrix4.txt")
>>> gg.P_center (5)
>>> gg.P_center_list
[2, 4, 5, 0]
>>> gg.P_center (6)
>>> gg.P_center_list
[2, 4, 0, 5]
>>> gg.P_center (7)
>>> gg.P_center_list
[4, 0, 2, 5]
>>>
```