

Bài kỹ hình dạng hình học nào trong mặt phẳng trên cơ sở điểm và đường thẳng cũng có thể biểu diễn bằng phương pháp này. Phương pháp Spaghetti cũng cấp liên kết cơ sở của các hình thể không gian, nhưng không cấp nhiều quan hệ không gian của chúng. Thí dụ, vùng A và B trên Hình 1 là kế nhau, chúng có đường biên chung. Nhưng phương pháp Spaghetti không chỉ ra các điểm chung. Cách biểu diễn này không có hiệu quả với các phép toán phân tích không gian và hệ thống phải lưu trữ nhiều dữ liệu dữ liệu. Phương pháp spaghetti thường được sử dụng vào những ứng dụng bản đồ máy tính đơn giản, không

D: [12, 13, 14, 15, 16, 17, 18, 19, 20, 21]

C: [6, 7, 8, 9, 10, 11, 12, 21, 24, 25]

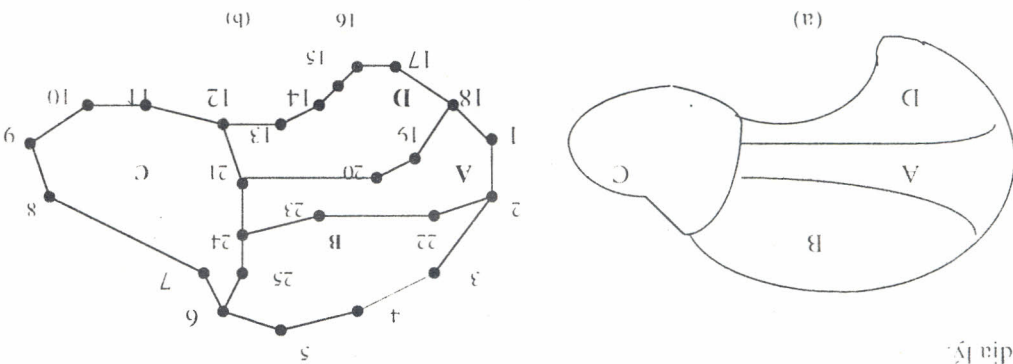
B: [2, 3, 4, 5, 6, 25, 24, 23, 22]

A: [1, 2, 22, 23, 24, 21, 20, 19, 18]

được biểu diễn như sau:

Mỗi hình dạng của vùng địa lý trên đây được biểu diễn bởi các số "điểm" (spaghetti). Các cạnh đã giắc tạo thành đường biên của vùng. Mỗi đa giác là một polyline khép kín. Mỗi polyline được biểu diễn bởi danh sách các điểm trở trên đầu và cuối đoạn thẳng. Theo thí dụ trên Hình 1 thì các vùng A, B, C, D

Hình 1. Chia mặt phẳng thành các vùng địa lý.



Câu trúc dữ liệu "spaghetti" biểu diễn hình dạng phẳng của các điểm, cung và các vùng hình học được biểu diễn như tập các đoạn thẳng. Các cung có thể được biểu diễn bởi xâu các đoạn thẳng như trên Hình 1. Phương pháp này không biểu diễn bất kỳ mối quan hệ nào của hình thể, thí dụ mối quan hệ của các vùng địa lý.

Vấn đề nghiên cứu ở đây dựa trên mô hình véc tơ để biểu diễn các dữ liệu không gian của hệ thống tin địa lý. Các vùng địa lý như các đơn vị hành chính, ao hồ được mô tả bởi các đường bao, các thực thể địa lý như đường quốc lộ, sông ngòi được mô tả bởi các đường thẳng... Hai phương pháp hay được sử dụng nhất để biểu diễn các thực thể địa lý thành các thực thể của cơ sở dữ liệu là "spaghetti" và "topology".

ABSTRACT: This article describes the algorithm on building a relationship among point and line objects in a geographical information system (GIS) in order to make level chain and area objects of vector GIS. It also describes the method of linking those spatial objects and statistical data of the system. This algorithm has been applied by Institute of Information Technology engineers and statistical data of the system. This algorithm has been applied by system to the United Nations Department for Economic and Social Information and Policy Analysis.

I. ĐẶT VẤN ĐỀ

LE QUỐC HÙNG

MỘT PHƯƠNG PHÁP XÂY DỰNG QUAN HỆ THÔNG TIN KHÔNG GIAN TRONG HỆ THỐNG TIN ĐỊA LÝ VEC TƠ PHÂN CẤP.

đòi hỏi thao tác hình học phức tạp và quan hệ cấu trúc. Đồng thời, chúng đặc biệt thuận tiện cho việc biểu diễn đồ họa trên các thiết bị như màn hình, máy in.

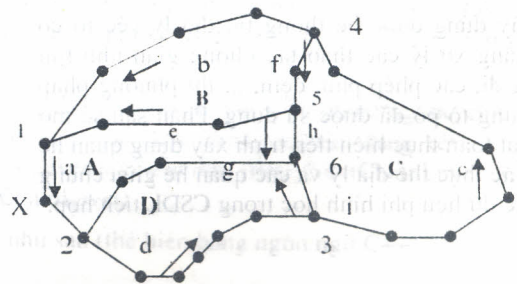
Phương pháp "topology" cho phép biểu diễn tổ pô của các hình thể hay xây dựng nhiều quan hệ tổ pô của các thực thể địa lý trong CSDL. Phương pháp này được hình thành trên mô hình quan hệ thực thể (E-R) của CSDL quan hệ. Từ 1976 Chen đã đề xuất mô hình E-R và chúng được sử dụng rộng rãi để mô hình hóa các hệ thống thông tin. Hệ thống tin địa lý cũng là một loại hình hệ thống thông tin [3], mô hình E-R cũng được sử dụng để mô tả hình dạng của các thực thể không gian. Giả sử ta có các thông tin qui chiếu không gian tới các đơn vị hành chính (thành phố, tỉnh, huyện...), mạng lưới giao thông (đường quốc lộ, đường sông...),... Yêu cầu đặt ra là phải tổ chức chúng sao cho dễ dàng khai thác thông tin của các thực thể và các thuộc tính của chúng, một cách đơn giản nhất là ứng dụng mô hình quan hệ E-R để xây dựng CSDL địa lý và các thao tác trên chúng. Khái niệm "kiểu thực thể" là trừu tượng, dùng để biểu diễn một lớp các đối tượng giống nhau. Trong thí dụ này, chúng có thể là thành phố, tỉnh, huyện, đường quốc lộ... Biểu đồ quan hệ thực thể là công cụ mô tả sự liên kết giữa các kiểu thực thể. Trong thí dụ này, ta có sự liên kết giữa thành phố và đường quốc lộ chạy qua thành phố. Ta gọi sự liên kết này là liên kết "nằm trên". Đó là khái niệm quan hệ trong mô hình E-R. Quan hệ có thuộc tính, thí dụ chiều dài đoạn đường đi trong thành phố, dân số của thành phố... Các kiểu quan hệ bao gồm N-N, N-1 và 1-N. Quan hệ "nằm trên" trong thí dụ này là quan hệ N-N bởi vì có thể có nhiều quốc lộ qua một thành phố và một quốc lộ có thể chạy qua nhiều thành phố.

Trong biểu đồ quan hệ mô tả quan hệ tổ pô có thực thể sơ đẳng như điểm, xâu, nút, vùng và chúng tuân thủ các luật sau đây:

- Mỗi xâu bao gồm một đoạn hay nhiều đoạn thẳng và chỉ có một điểm đầu và một điểm cuối.
- Mỗi vùng được bao bọc bởi một hay nhiều xâu.
- Mỗi xâu chỉ có một vùng ở phía phải và một vùng ở phía trái nó.
- Mỗi vùng phải là vùng phía phải hay phía trái của ít nhất một xâu nào đó.
- Điểm đầu hay điểm cuối là nút nếu chúng là giao của các xâu.
- Mỗi nút là điểm đầu hoặc điểm cuối của ít nhất một xâu hay điểm đầu hoặc điểm cuối của một xâu sẽ thuộc một nút nào đó.

Như vậy, kiểu thực thể "vùng" có thể có các kiểu con "vùng trái", "vùng phải" và chúng tạo thành quan hệ 1-N với các xâu bao chúng ở phía trái và phía phải. Kiểu thực thể "nút" có các kiểu con "điểm đầu", "điểm cuối" và cũng tạo thành quan hệ 1-N với các xâu thuộc các nút đó. Thí dụ trên Hình 2 mô tả hình dạng của Hình 1 trên cơ sở các xâu, nút và các vùng.

ID xâu	Điểm đầu	Điểm cuối	Vùng trái	Vùng phải
a	1	2	A	X
b	4	1	B	X
c	3	4	C	X
d	2	3	D	X
e	5	1	A	B
f	4	5	C	B
g	6	2	D	A
h	5	6	C	A
i	3	6	D	C



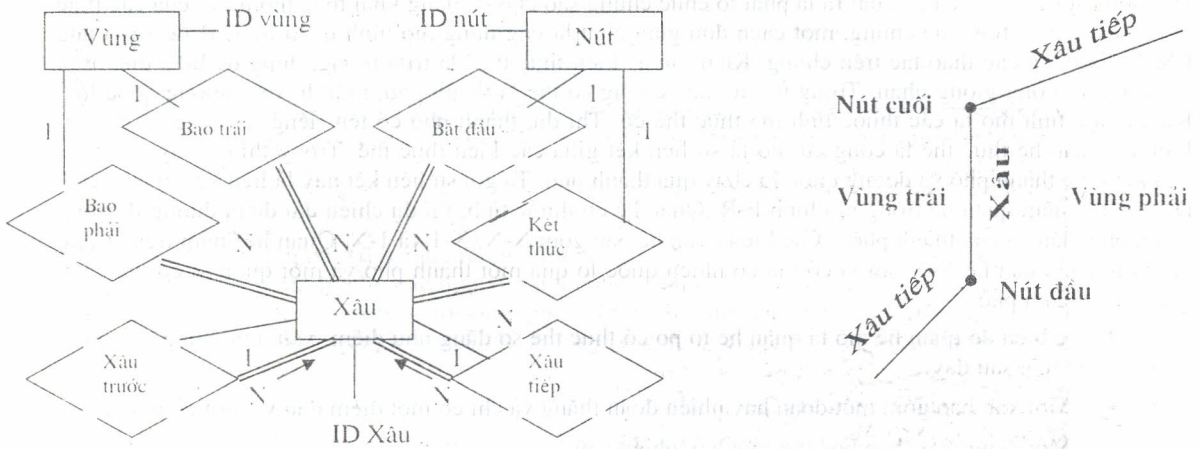
Hình 2: Các thực thể sơ đẳng và các quan hệ tương ứng với hình dạng vùng.

Sơ đồ quan hệ E-R như mô tả trên đã hình thành cơ sở của sơ đồ CSDL quan hệ chuẩn như các thí dụ sau đây:

XAU (XAU_ID, DIEM_DAU, DIEM_CUOI, VUNG_TRAI, VUNG_PHAU)

NUT(NUT_ID, TOA_DO_X, TOA_DO_Y) VUNG(VUNG_ID, TRINH_TU_CAC_XAU)

Mỗi thực thể xâu còn có xâu trước, và xâu sau nó. Đồng thời, chúng còn có hướng "đi tới" hay "đi từ" một nút cụ thể nào đó. Hình 3 sau đây là sơ đồ E-R mở rộng, biểu diễn quan hệ tô pô của các thực thể địa lý trong CSDL.



Hình 3. Biểu đồ E_R mở rộng

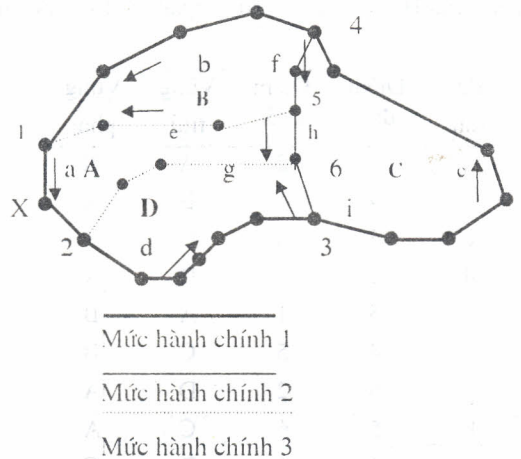
Hệ thống tin địa lý được nghiên cứu là hệ thống thông tin phân cấp. Mỗi thực thể vùng vừa mô tả như trên có thể là một đơn vị hành chính cấp thấp nhất, cấp L. Đơn vị hành chính cấp cao hơn, cấp (L - i), trong đó $i = 1, \dots, L-1$ là tập của các thực thể vùng ở cấp thấp cao hơn nó. Để xây dựng tô pô có thực thể địa lý trong hệ thống tin loại này, thuộc tính "cấp" đã được thêm vào thực thể xâu để có:

XAU(XAU_ID, DIEM_DAU, DIEM_CUOI, VUNG_TRAI, VUNG_PHAU, MUC)

Khi đó Hình 2 sẽ như Hình 4.

Một xâu ở mức hành chính cao hơn cũng sẽ thuộc mức hành chính thấp hơn. Ví dụ: xâu c thuộc mức hành chính 1 cũng sẽ thuộc mức hành chính 2 và mức hành chính 3 trong việc xây dựng vùng địa lý phân cấp. (Mức hành chính 1 là mức hành chính cao nhất)

Để xây dựng được hệ thống tin địa lý véc tơ có khả năng xử lý các thao tác không gian như tìm đường đi, các phép phủ, đệm, ... thì phương pháp xây dựng tô pô đã được sử dụng. Phần sau sẽ mô tả thuật toán thực hiện tiến trình xây dựng quan hệ giữa các thực thể địa lý và các quan hệ giữa chúng với các dữ liệu phi hình học trong CSDL tích hợp.



Hình 4: Bản đồ với các đường biên hành chính.

II. THUẬT TOÁN TẠO VÙNG ĐỊA LÝ PHÂN CẤP TỪ CÁC XÂU PHÂN CẤP.

Thuật toán được xây dựng dựa trên mô hình dữ liệu véc tơ. Trong mô hình này, một xâu được định nghĩa là tập hợp $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ trong đó: (x_i, y_i) là tọa độ các điểm liên tiếp; (x_1, y_1) : *điểm đầu* của xâu; (x_n, y_n) : *điểm cuối* của xâu.

Với bài toán đang xét, mỗi xâu sẽ thuộc một mức hành chính nào đó (hình 4). Ta sẽ dùng cấu trúc sau (bằng ngôn ngữ C++) để mô tả một xâu để thuận lợi cho việc cài đặt thuật toán trên máy tính:

```
typedef struct {
    LONG chain_idx;          /* số thứ tự (chỉ số) của xâu */
    INT level;              /* mức hành chính của xâu */
    DOUBLE data; /* con trỏ tới vùng lưu trữ dữ liệu tọa độ của xâu (x1, y1, ..., xn, yn) */
    LONG first_nextchainlevel; /* chỉ số của xâu là Next ChainLevel của xâu tại điểm đầu. */
    LONG end_nextchainlevel; /* chỉ số của xâu là Next ChainLevel của xâu tại điểm cuối. */
} Chain_st;
```

(NextChainLevel sẽ được giải thích ở phần dưới)

Trong CSDL một bản đồ hành chính sẽ được biểu diễn bởi tập hợp các xâu:

BảnĐồ = { xâu 1, xâu 2, ..., xâu n }

Đầu vào của thuật toán là bản đồ (tập hợp các xâu)

Đầu ra là một bảng mô tả các vùng tạo được:

Vùng 1: xâu 1, xâu 3, ..., xâu k

Vùng 2: xâu 1, xâu 2, ..., xâu j

...

Ta dùng cấu trúc để mô tả một vùng

```
typedef struct {
```

```
    LONG area_dix;          /* số thứ tự (chỉ số) của vùng */
```

```
    INT level;             /* mức hành chính của vùng */
```

```
    DOUBLE* chain_list; /* con trỏ tới vùng dữ liệu lưu trữ danh sách của xâu tạo nên vùng (xâu 1, xâu 3, ..., xâu k) */
```

```
    } Area_st;
```

Mô tả thuật toán

- Gọi hình chữ nhật chứa xâu i (P_i) là hình chữ nhật nhỏ nhất chứa toàn bộ xâu thứ i . $R_i = \{(left_i, top_i); (right_i, bottom_i)\}$ trong đó $(left_i, top_i)$ là tọa độ góc bên trái và $(right_i, bottom_i)$ là tọa độ góc dưới bên phải.
- NextChainLevel của một xâu tại một *nút* là xâu đầu tiên duyệt theo chiều ngược chiều kim đồng hồ có mức hành chính bằng hoặc lớn hơn mức hành chính của xâu đang xét. Như vậy với một xâu ta sẽ có thể có một NextChainLevel tại *điểm đầu* (first_NextChainLevel) và một NextChainLevel tại *điểm cuối* (end_NextChainLevel).
- Một *nút* được mô tả bằng cấu trúc dữ liệu như sau (thể hiện bằng ngôn ngữ C++)

```
typedef struct {
```

```
    INT idx;              /* số thứ tự (chỉ số nút) */
```

```
    INT ref_num;         /* số các xâu liên kết với nút */
```

```
    DOUBLE x,y;         /* tọa độ của nút */
```

```
    LONG chain_list; /* con trỏ tới vùng dữ liệu chứa danh sách các xâu liên kết với nút. */
```

```
    }Node_st;
```

Các bước của thuật toán:

- b1) Sắp xếp lại tọa độ trong từng râu sao cho tọa độ y của *điểm đầu* lớn hơn tọa độ y của *điểm cuối* cùng trong râu.
- b2) Sắp xếp lại thứ tự các râu trong từng mức theo chiều tăng của các Ri từ trái sang phải và từ trên xuống dưới.
- b3) Tạo các *nút* trong bản đồ.
- b4) Sắp xếp các râu liên kết với *nút* theo thứ tự ngược chiều kim đồng hồ theo độ lớn của góc nghiêng (theta) của các râu với *nút*.

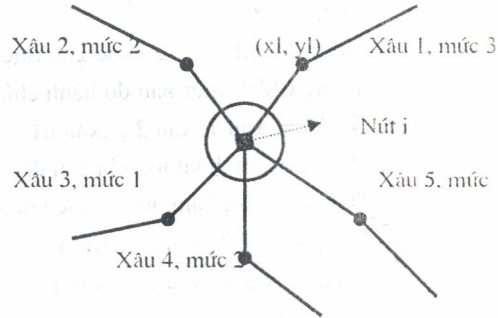
Góc nghiêng của từng râu tại một *nút* được tính như sau (thể hiện theo ngôn ngữ lập trình C):

```
double theta(x,y,x1,y1)
```

```
double x,y; /* tọa độ của nút hay tọa độ điểm đầu hoặc điểm cuối của 1 râu. */
```

```
double x1,y1; /* tọa độ điểm thứ 2 hoặc điểm áp cuối của râu. */
```

```
{
double ax, ay, dx, dy;
double t;
dx = x1-x; ax=fabs(dx);
dy = y1-y; ay=fabs(dy);
if((ax+ay)==0)t=0' else t=dt/(ax+ay);
if(dx<0)t=2.0-t;
else if(dy<0) t=4.0+t;
return(t*90.0);
}
```



Hình 5. Nút và các râu liên kết

- b5) Xác định các râu kế nhau theo mức hành chính tại một *nút* (xác định NextChainLevel của từng râu).

Xét một râu tại một *nút*:

Duyệt theo chiều ngược chiều kim đồng hồ đến khi gặp râu đầu tiên có mức hành chính bằng hoặc lớn hơn mức hành chính của râu đang xét.

- Nếu tìm được và nếu *nút* là *điểm đầu* của râu đang xét thì gán first_nextchainlevel của râu đang xét bằng chỉ số của râu vừa tìm được.
- Nếu tìm được và nếu *nút* là *điểm cuối* của râu đang xét thì gán end_nextchainlevel của râu đang xét bằng chỉ số của râu vừa tìm được.

Ví dụ: Trong Hình 5 tại nút i:

NextChainLevel của râu 1 (mức 3) là râu 2 (mức 2)

NextChainLevel của râu 5 (mức 1) là râu 3 (mức 1)

- b6) Tạo các vùng (polygon) theo từng mức:

Xét tại một mức hành chính M/

b6.1) Lấy một râu thuộc mức M làm râu đầu tiên trong danh sách các râu tạo ra vùng.

b6.2) Tạo vùng phải của râu k như sau:

b6.2.1) Xuất phát từ *điểm cuối* của râu k.

- Chọn *nút* đang xét là *nút* có tọa độ trùng với tọa độ tại *điểm đầu* của râu k.
- Nếu tồn tại first_nextchainlevel của râu k thì cộng thêm râu này vào cuối danh sách các râu tạo ra vùng.
- Nếu không tồn tại first_nextchainlevel thì dừng

b6.2.2) Xét tiếp một trong hai trường hợp sau:

i) Nếu nút đang xét là *điểm đầu* của xâu cuối cùng vừa được chọn thì:

- Chọn nút đang xét là nút có tọa độ trùng với tọa độ tại *điểm cuối* của xâu cuối cùng.
- Nếu tồn tại `end_nextchainlevel` của xâu cuối cùng thì cộng thêm xâu này vào cuối danh sách các xâu tạo ra vùng.
- Nếu không tồn tại `end_nextchainlevel` thì dừng.

ii) Nếu nút đang xét là *điểm cuối* của xâu cuối cùng vừa được chọn thì:

- Chọn nút đang xét là nút có tọa độ trùng với tọa độ tại *điểm đầu* của xâu cuối cùng.
- Nếu tồn tại `first_nextchainlevel` của xâu cuối cùng thì cộng thêm xâu này vào cuối danh sách các xâu tạo ra vùng.
- Nếu không tồn tại `first_nextchainlevel` thì dừng.

b6.2.3) Lặp lại bước b6.2.2) đến khi nào nút đang xét có tọa độ trùng với tọa độ *điểm cuối* của xâu đầu tiên thì dừng.

* Trong các trường hợp dừng:

Nếu danh sách chỉ có một xâu thì nếu *điểm đầu* và *điểm cuối* trùng nhau ta tạo được vùng phải của xâu k; ngược lại nếu nút đang xét có tọa độ trùng với tọa độ *điểm cuối* của xâu đầu tiên thì ta cũng đã tạo được vùng phải của xâu k.

b6.3) Tạo vùng trái của xâu k như sau:

b6.3.1) Xuất phát từ *điểm đầu* của xâu k.

- Chọn nút đang xét là nút có tọa độ trùng với tọa độ tại *điểm cuối* của xâu k.
- Nếu tồn tại `end_nextchainlevel` của xâu k thì cộng thêm xâu này vào cuối danh sách các xâu tạo ra vùng.
- Nếu không tồn tại `end_nextchainlevel` thì dừng.

b6.3.2) Xét tiếp một trong hai trường hợp sau:

i) Nếu nút đang xét là *điểm đầu* của xâu cuối cùng vừa được chọn thì:

- Chọn nút đang xét là nút có tọa độ trùng với tọa độ tại *điểm cuối* của xâu cuối cùng.
- Nếu tồn tại `end_nextchainlevel` của xâu cuối cùng thì cộng thêm xâu này vào cuối danh sách các xâu tạo ra vùng.
- Nếu không tồn tại `end_nextchainlevel` thì dừng.

ii) Nếu nút đang xét là *điểm cuối* của xâu cuối cùng vừa được chọn thì:

- Chọn nút đang xét là nút có tọa độ trùng với tọa độ tại *điểm đầu* của xâu cuối cùng.
- Nếu tồn tại `first_nextchainlevel` của xâu cuối cùng thì cộng thêm xâu này vào cuối danh sách các xâu tạo ra vùng.
- Nếu không tồn tại `first_nextchainlevel` thì dừng.

b6.3.3) Lặp lại bước 6.3.2) đến khi nào nút đang xét có tọa độ trùng với tọa độ *điểm đầu* của xâu đầu tiên thì dừng.

* Trong các trường hợp dừng:

Nếu danh sách chỉ có một xâu thì nếu *điểm đầu* và *điểm cuối* trùng nhau ta tạo được vùng trái với xâu k; ngược lại nếu nút đang xét có tọa độ trùng với tọa độ *điểm đầu* của xâu đầu tiên thì ta cũng đã tạo được vùng trái của xâu k.

Ví dụ: Với bản đồ hành chính như hình 4 ta tạo được các vùng hành chính như sau:

Mức hành chính 1:

Vùng 1 = { xâu b, xâu c, xâu d, xâu a }

Mức hành chính 2:

Vùng 2={ xâu f, xâu h, xâu i, xâu d, xâu a, xâu b}

Vùng 3={ xâu i, xâu h, xâu f, xâu c}

Mức hành chính 3:

Vùng 4={ xâu f, xâu e, xâu b}

Vùng 5={xâu h, xâu g, xâu a, xâu e}

Vùng 6={xâu i, xâu d, xâu g}

Vùng 7={xâu i, xâu h, xâu f, xâu c}

III. CÀI ĐẶT VÀ KẾT LUẬN

Thuật toán trên đã được cài đặt trên máy tính bằng ngôn ngữ lập trình C phục vụ cho việc xây dựng các vùng hành chính phân cấp trong hệ thống thông tin địa lý POPMAP. Thuật toán trên được áp dụng cho lớp bản đồ hành chính- lớp bản đồ chỉ bao gồm các đối tượng địa lý là các vùng (polygon). Trong thực tế một lớp bản đồ thường rất phức tạp, các đối tượng địa lý không chỉ bao gồm các vùng mà còn bao gồm các điểm (point) (như trường học, bệnh viện...), các đường (polyline) (như đường sông, đường quốc lộ, đường sắt...) v.v.; trong trường hợp này thuật toán cần phải được mở rộng để có thể phân biệt được các đối tượng địa lý không phải là vùng.

TÀI LIỆU THAM KHẢO

- [1] Đặng Văn Đức (1996) *Một phương pháp xây dựng hệ thống thông tin tích hợp. Luận án Phó tiến sĩ khoa học kỹ thuật*, 120 trang.
- [2] Michael F. Worboys (1995). *GIS: A Computing Perspective*. Taylor & Francis Inc. Bristol PA 19007, USA. 367 p.
- [3] Robert G. Cromley (1992). *University of Connecticut. Digital Cartography*. Prentice Hall, Englewood Cliffs, New Jersey 07632. 311p.
- [4] United Nations DESIPA (December 1996). *Population Software Notes*. NewsLetter, New York, USA. 21P.

Hà Nội, ngày 08/09/1997.