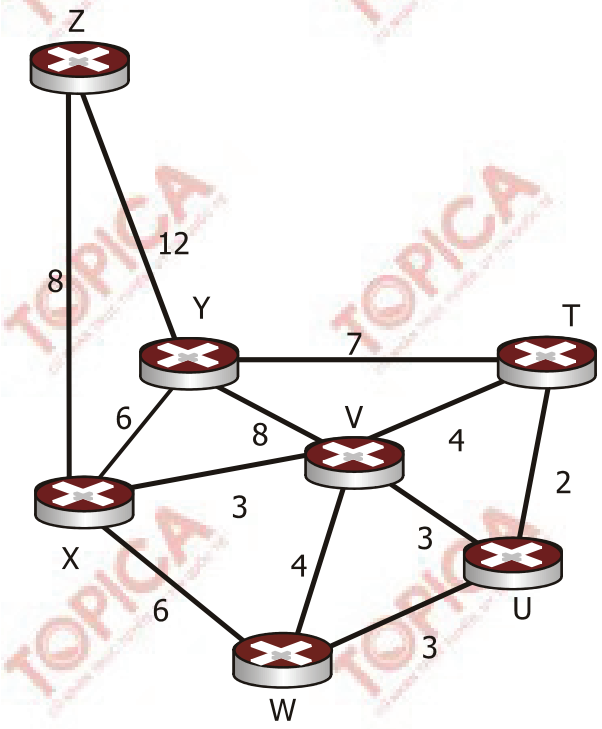
BÀI 6: CÁC THUẬT TOÁN TÌM KIẾM TRÊN ĐỒ THỊ VÀ MỘT SỐ ỨNG DỤNG



Nội dung

- Giới thiệu bài toán tìm kiếm trên đồ thị
- Thuật toán tìm kiếm theo chiều rộng
- Thuật toán tìm kiếm theo chiều sâu
- Một số ứng dụng

Giới thiệu

Bài toán tìm kiếm trên đồ thị được phát biểu chung cho cả đồ thị có hướng hay vô hướng với ý nghĩa một cạnh vô hướng xem như đi theo chiều nào cũng được. Có thể nói, trong việc giải quyết nhiều bài toán ứng dụng trên đồ thị, thao tác tìm kiếm được dùng như là một trong những thao tác cơ bản, đến mức vai trò của nó trong ứng dụng đồ thị cũng giống như vai trò của các phép cộng, trừ, ... trong tính toán số học.

Mục tiêu

Sau khi học bài này, các bạn có thể:

- Hiểu thế nào là bài toán tìm kiếm trên đồ thị.
- Sử dụng các thuật toán:
  - Tìm kiếm theo chiều rộng
  - Tìm kiếm theo chiều sâu

  vào việc giải quyết bài toán tìm kiếm trên đồ thị.
- Minh họa một số kết quả và ứng dụng của bài toán tìm kiếm trên đồ thị qua việc nghiên cứu một số ứng dụng cụ thể.

Thời lượng

- 6 tiết

TÌNH HUỐNG DẪN NHẬP

Tình huống: Truyền tin

Một lớp gồm N học viên, mỗi học viên cho biết những bạn mà học viên đó có thể liên lạc được (chú ý liên lạc này là liên lạc một chiều, ví dụ : Bạn An có thể gửi tin tới Bạn Vinh nhưng Bạn Vinh thì chưa chắc đã có thể gửi tin tới Bạn An).

Thầy chủ nhiệm đang có một thông tin rất quan trọng cần thông báo tới tất cả các học viên của lớp (tin này phải được truyền trực tiếp). Để tiết kiệm thời gian, thầy chỉ nhắn tin tới 1 số học viên rồi sau đó nhờ các học viên này nhắn lại cho tất cả các bạn mà các học viên đó có thể liên lạc được, và cứ lần lượt như thế làm sao cho tất cả các học viên trong lớp đều nhận được tin .

Câu hỏi

Có phương án nào giúp thầy chủ nhiệm với một số ít nhất các học viên mà thầy chủ nhiệm cần nhắn?

6.1. Phát biểu bài toán tìm kiếm trên đồ thị

Cho trước một đồ thị và một đỉnh s cho trước, cần duyệt tất cả các đỉnh, có đường đi từ s đến nó, sao cho mỗi đỉnh được duyệt đúng một lần. Thao tác vừa nói, được gọi là thao tác *tìm kiếm các đỉnh trên đồ thị bắt đầu từ đỉnh s*.

Có thể nói, trong rất nhiều lời giải của các ứng dụng trên đồ thị, thao tác tìm kiếm được dùng như là một trong những thao tác cơ bản, đến mức vai trò của nó trong ứng dụng đồ thị cũng giống như vai trò của các phép cộng, trừ, ... trong tính toán số học.

Chú ý rằng, trong phát biểu vừa nêu, hai điều kiện cơ bản của thao tác tìm kiếm là *không bỏ sót* và *không trùng lặp* còn thứ tự tìm kiếm là không được tính đến. Chúng có thể khác nhau tùy các chiến lược tìm kiếm khác nhau trong những mục tiêu ứng dụng khác nhau.

Bài toán tìm kiếm trên đồ thị được phát biểu chung cho cả đồ thị có hướng hay vô hướng với ý nghĩa một cạnh vô hướng xem như đi theo chiều nào cũng được. Ngoài ra, việc có nhiều cạnh nối cùng một cặp đỉnh cũng không có gì khác chỉ có một cạnh nối cặp đỉnh này, vì thế trong bài toán tìm kiếm, một đa đồ thị cũng giống như một đơn đồ thị.

Thao tác tìm kiếm các đỉnh trên đồ thị bắt đầu từ đỉnh s còn được gọi là thao tác *duyệt các đỉnh* hay *đi thăm các đỉnh bắt đầu từ đỉnh s*. Các đỉnh được tìm kiếm được gọi là các đỉnh *được duyệt* hay *được thăm từ đỉnh s*.

Có hai thuật toán cơ bản thực hiện việc tìm kiếm các đỉnh trên đồ thị bắt đầu từ một đỉnh là *tìm kiếm theo chiều rộng* và *tìm kiếm theo chiều sâu* được trình bày trong các mục dưới đây.

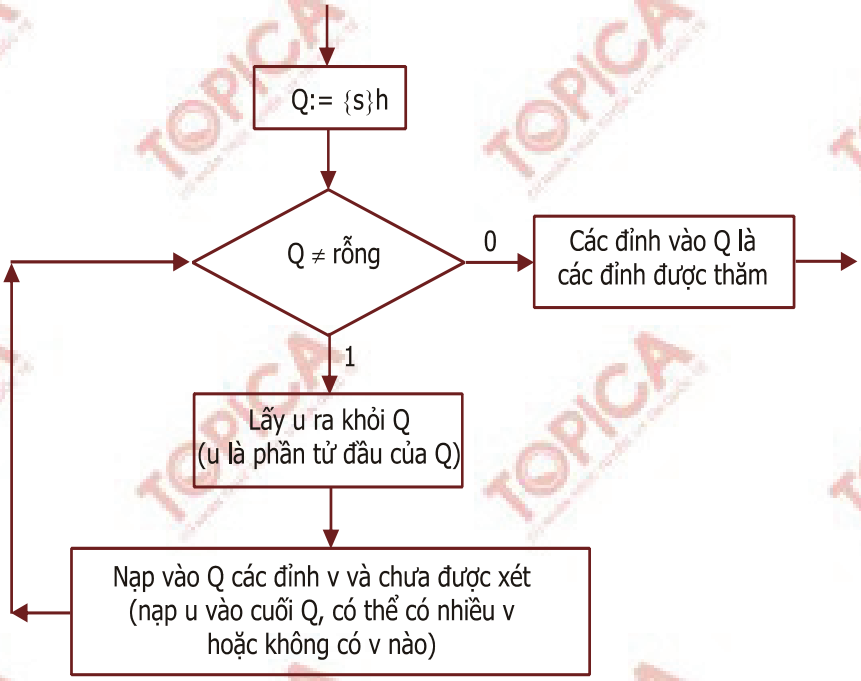
6.2. Tìm kiếm theo chiều rộng

Việc tìm kiếm theo chiều rộng bắt đầu từ đỉnh s, được tiến hành đồng thời theo mọi hướng và phát triển theo từng mức: "gần" s thăm trước, "xa" s thăm sau. Để đo độ gần, xa của các đỉnh được thăm so với s, ta đánh "mức" của các đỉnh như sau. Đỉnh s là đỉnh có mức 0 (đỉnh xuất phát). Những đỉnh kề với s được đánh mức 1. Những đỉnh kề với đỉnh mức 1 mà chưa được xét được đánh mức 2, ...

Chú ý: mỗi đỉnh chỉ được đánh mức đúng một lần vì thế quá trình đánh mức sẽ kết thúc, khi đó các đỉnh đã được đánh mức là những đỉnh được thăm, trong đó những đỉnh có mức thấp hơn được thăm trước, những đỉnh có mức cao hơn được thăm sau, những đỉnh cùng mức xem như được thăm đồng thời với ý nghĩa theo thứ tự nào cũng được. Từ quy tắc đánh mức ta cũng thấy rằng, mức của đỉnh được thăm là độ dài (tính theo số cạnh) của đường đi ngắn nhất từ đỉnh s đến đỉnh đó. Về trực giác, ta có thể hình dung rằng, việc tìm kiếm các đỉnh theo chiều rộng được tiến hành theo các vòng tròn đồng tâm (với tâm điểm là đỉnh xuất phát) với bán kính tăng dần (các đỉnh trên cùng một vòng tròn là những đỉnh cùng mức) cho đến khi không tăng được nữa, nó giống như việc nhỏ một giọt dầu trên mặt nước, vết dầu sẽ loang rộng ra xung quanh. Cũng chính vì vậy, tên gọi của thuật toán này là *tìm kiếm theo chiều rộng* và nó cũng có một tên gọi khác là *vết dầu loang*.

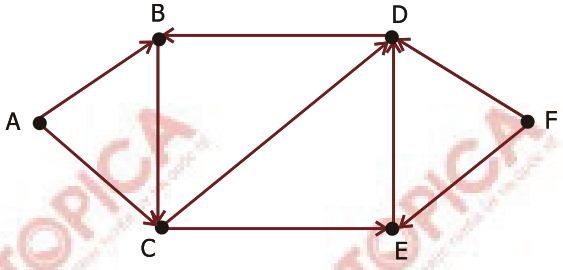
Để tổ chức việc tìm kiếm theo chiều rộng, các đỉnh được xét thăm cần được lưu trữ dần trong một danh sách, trong đó đỉnh được thăm được lấy ra khỏi danh sách (xem

như đã xét), đồng thời các đỉnh kề với đỉnh này và chưa được xét thăm, sẽ được nạp vào danh sách. Vì việc đi thăm phải tuân theo thứ tự "gần" trước, "xa" sau nên các đỉnh được lấy ra phải ở một đầu của danh sách (gọi là *đầu* danh sách), còn các đỉnh được nạp vào phải ở đầu kia của danh sách (gọi là *cuối* danh sách). Một danh sách hoạt động theo cơ chế vào trước, ra trước như vậy được gọi là một *hàng đợi* (queue – xem trong các giáo trình về cấu trúc dữ liệu và giải thuật). Rõ ràng lúc ban đầu, hàng đợi cần chứa một phần tử là đỉnh xuất phát. Quá trình tìm kiếm theo chiều rộng xuất phát từ đỉnh s, nhờ hàng đợi Q, có thể mô tả qua sơ đồ khối sau:



Để cài đặt, hàng đợi Q có thể tổ chức như một mảng (đánh số từ 1) với số phần tử tối đa là số đỉnh và giá trị khởi động là $Q_1 = s$. Để kiểm soát các vị trí đầu, cuối của Q, có thể dùng hai biến nguyên *dau*, *cuoi* với giá trị khởi động là 1. Phần tử u được lấy ra là Q_{dau} và mỗi khi lấy ra cần tăng biến *dau* lên một đơn vị. Mỗi khi nạp v vào Q cần tăng biến *cuoi* lên một đơn vị và gán Q_{cuoi} bằng v. Để kiểm soát một đỉnh đã được xét thăm hay chưa, cần tổ chức một mảng lôgic B có các chỉ số chạy trên tập đỉnh. Ban đầu B được khởi gán B_u := TRUE với mọi đỉnh u, trừ B_s := FALSE (đỉnh xuất phát được thăm đầu tiên, các đỉnh còn lại chưa được thăm). Điều kiện nạp v vào Q khi đó sẽ là *v kề với u và* B_v. Mỗi khi nạp v vào Q cần gán lại B_v := FALSE (đỉnh v đã được xét thăm) để đảm bảo không xét lại v lần thứ hai. Dấu hiệu Q rỗng (cũng là dấu hiệu kết thúc vòng lặp) là các biến *dau*, *cuoi* lần đầu tiên thỏa mãn bất đẳng thức *dau* > *cuoi*. Khi kết thúc, các đỉnh được thăm từ s theo thứ tự sẽ là $Q_1, Q_2, ..., Q_{cuoi}$.

Ví dụ. Cho đồ thị (có hướng) dưới đây:



Hãy đi thăm các đỉnh bắt đầu từ A.

Giải. Bảng dưới đây cho biết trạng thái của hàng đợi Q, đỉnh lấy ra, đỉnh đưa vào trong từng bước theo thuật toán đã nêu:

Hàng đợi Q	Đỉnh lấy ra	Đỉnh đưa vào
A	A	B, C
B, C	B	
C	C	D, E
D, E	D	F
E, F	E	
F	F	
rỗng		

Kết quả cho thấy các đỉnh được thăm bắt đầu từ A theo thứ tự là A, B, C, D, E, F. Có thể giải lại thí dụ này với các đỉnh xuất phát khác nhau, chẳng hạn nếu xuất phát từ C, các đỉnh đều được thăm ngoại trừ đỉnh A (D, E có mức 1, B, F có mức 2), còn nếu xuất phát từ F thì không có đỉnh nào được thăm cả, ngoại trừ chính nó.

Chú ý

- Các đỉnh được thăm theo thứ tự mức: mức thấp thăm trước, mức cao thăm sau. Trong thí dụ trên, A (đỉnh xuất phát) có mức 0 được thăm đầu tiên, tiếp theo là các đỉnh B, C có mức 1, sau đó là các đỉnh D, E có mức 2 và cuối cùng là đỉnh F có mức 3.
- Các đỉnh cùng mức được thăm theo thứ tự nào cũng được. Như vậy, việc thăm theo chiều rộng chỉ xác định thứ tự theo mức các đỉnh, còn thứ tự theo các đỉnh cùng mức là không duy nhất, tùy theo từng cài đặt xử lý thứ tự này. Trong thí dụ trên, những đỉnh cùng mức được xử lý theo thứ tự từ điển của tên đỉnh, nếu xử lý theo thứ tự ngược từ điển, trình tự đi thăm sẽ là A, C, B, E, D, F (vẫn đảm bảo thứ tự theo mức).
- Mức cao nhất đạt được là độ dài của đường đi ngắn nhất (tính theo số cạnh) từ đỉnh xuất phát đến đỉnh "xa" nó nhất (những đỉnh "xa" đỉnh xuất phát nhất là những đỉnh đạt được mức cao nhất này). Trong thí dụ trên, đỉnh F có mức 3 là đỉnh "xa" A nhất, và 3 là độ dài đường đi ngắn nhất từ A đến F.

6.3. Tìm kiếm theo chiều sâu

Quá trình tìm kiếm theo chiều sâu được tiến hành theo một hướng: từ đỉnh xuất phát s (xem như đã thăm), chọn một đỉnh u bất kỳ kề với s để thăm u, sau đó từ đỉnh u, chọn một đỉnh v bất kỳ kề với u và chưa được thăm để thăm v, ... Quá trình đi thăm được phát triển theo một hướng như vậy cho đến khi không phát triển được nữa (chú ý không thăm đỉnh đã thăm rồi), khi đó cần lùi lại đỉnh trước để phát triển theo hướng khác. Chính vì việc tìm kiếm chỉ phát triển theo một hướng cho đến tận cùng như vậy, nên thuật toán này có tên gọi là *tìm kiếm theo chiều sâu*.

Để tiến hành tìm kiếm theo chiều sâu, các đỉnh được thăm cần được lưu trữ vào một danh sách sao cho đỉnh được nạp sau cùng sẽ được lùi ra đầu tiên. Với cơ chế vào sau, ra trước như vậy, danh sách chỉ có một đầu, vừa là nơi lấy ra vừa là nơi nạp vào các phần tử. Một danh sách như thế được gọi là một *ngăn xếp* (stack – xem các giáo trình về cấu trúc dữ liệu và giải thuật).

Thuật toán tìm kiếm theo chiều sâu dễ dàng được phát biểu một cách đệ quy (giống như thuật toán quay lui), vì thế tiện hơn cả là dùng một ngôn ngữ đệ quy để cài đặt (chẳng hạn C hay Pascal). Với cách này ta có thể lợi dụng cơ chế ngăn xếp của lời gọi đệ quy mà không cần phải tổ chức ngăn xếp một cách tường minh.

Giả sử các đỉnh của đồ thị được đánh số bằng các số nguyên. Việc tìm kiếm theo chiều sâu có thể thực hiện nhờ thủ tục đệ quy (mô phỏng Pascal) dưới đây:

```
PROCEDURE VISIT (u: INTEGER);
{đi thăm các đỉnh theo chiều sâu bắt đầu từ đỉnh u}
VAR v: INTEGER;
BEGIN
    (đánh dấu đã thăm u);
    FOR (v kề với u và v chưa được thăm) DO VISIT(v);
END;
```

Trong chương trình chính, sau khi khởi gán các đỉnh đều chưa được thăm, thủ tục VISIT được khởi động bằng lời gọi VISIT(s) với s là đỉnh xuất phát.

Trở lại thí dụ đã xét trong mục trên, với đỉnh xuất phát là A (xem A như số nguyên). Lời gọi VISIT(A) sẽ đánh dấu thăm A và duyệt các đỉnh kề với A và chưa được thăm, sau đó chọn một trong các đỉnh này để thăm tiếp. Việc chọn đỉnh nào trước tùy thuộc thứ tự duyệt trong vòng FOR. Để xác định, giả sử vòng FOR duyệt các đỉnh theo thứ tự từ điển, khi đó VISIT(B) được gọi, ... cứ như vậy, các lời gọi đệ quy tiếp theo sẽ là VISIT(C), VISIT(D), VISIT(F). Đến đây, không có đỉnh nào thỏa mãn điều kiện được thăm, lời gọi VISIT(F) kết thúc và quá trình lùi lại cho đến VISIT(C) và trong thủ tục này VISIT(E) được gọi, sau đó quá trình lùi dần ra đến VISIT(A) và kết thúc. Cuối cùng ta nhận được kết quả đi thăm theo thứ tự là A, B, C, D, F, E.

Như vậy, thứ tự đi thăm theo chiều sâu là không xác định, nó tùy thuộc vào thứ tự duyệt các đỉnh thỏa mãn điều kiện được thăm trong từng bước tìm kiếm, vì thế nó không thỏa mãn điều kiện "gần" thăm trước, "xa" thăm sau như trong tìm kiếm theo chiều rộng. Bạn đọc có thể cài một thứ tự duyệt ngẫu nhiên trong vòng FOR của thủ tục VISIT để thấy rằng, mỗi lần chạy lại chương trình (với cùng một bộ dữ liệu), thứ tự các đỉnh được thăm là khác nhau. Cuối cùng cần chú ý rằng, dù thứ tự đi thăm là khác nhau, việc tìm kiếm theo chiều rộng hay theo chiều sâu vẫn đảm bảo điều kiện các đỉnh được thăm là *không bỏ sót* và *không trùng lặp*.

Việc viết một chương trình cụ thể minh họa các thuật toán tìm kiếm theo chiều rộng và theo chiều sâu được dành cho bạn đọc như bài tập.

6.4. Một số ứng dụng của các thuật toán tìm kiếm

Thuật toán tìm kiếm được dùng trong rất nhiều ứng dụng của đồ thị. Dưới đây ta chỉ xét những ứng dụng đơn giản nhất mà kết quả của chúng dường như là hệ quả trực tiếp của thao tác tìm kiếm.

6.4.1. Xác định tính liên thông của đồ thị

Điều kiện liên thông của đồ thị thường là một yêu cầu tất yếu trong nhiều ứng dụng, chẳng hạn một mạng giao thông hay mạng thông tin nếu không liên thông thì xem như bị hỏng, cần sửa chữa. Vì thế, việc kiểm tra một đồ thị có liên thông hay không là một thao tác cần thiết trong nhiều ứng dụng khác nhau của đồ thị. Dưới đây ta xét một tình huống đơn giản (nhưng cũng là cơ bản) là xác định tính liên thông của một đồ thị vô hướng với nội dung cụ thể như sau: "cho trước một đồ thị vô hướng, hỏi rằng nó có liên thông hay không?".

Để trả lời bài toán, xuất phát từ một đỉnh tùy ý, ta bắt đầu thao tác tìm kiếm từ đỉnh này (có thể chọn một trong hai thuật toán tìm kiếm đã nêu). Khi kết thúc tìm kiếm, xảy ra hai tình huống: nếu tất cả các đỉnh của đồ thị đều được thăm thì đồ thị đã cho là liên thông, nếu có một đỉnh nào đó không được thăm thì đồ thị đã cho là không liên thông. Như vậy, câu trả lời của bài toán xem như một hệ quả trực tiếp của thao tác tìm kiếm.

Để kiểm tra xem có phải tất cả các đỉnh của đồ thị có được thăm hay không, ta chỉ cần thêm một thao tác nhỏ trong quá trình tìm kiếm, đó là dùng một biến đếm để đếm số đỉnh được thăm. Khi kết thúc tìm kiếm, câu trả lời của bài toán sẽ phụ thuộc vào việc so sánh giá trị của biến đếm này với số đỉnh của đồ thị: nếu giá trị biến đếm bằng số đỉnh thì đồ thị là liên thông, nếu trái lại thì đồ thị là không liên thông.

Trong trường hợp đồ thị là không liên thông, kết quả tìm kiếm sẽ xác định một thành phần liên thông chứa đỉnh xuất phát. Bằng cách lặp lại thao tác tìm kiếm với đỉnh xuất phát khác, không thuộc thành phần liên thông vừa tìm, ta nhận được thành phần liên thông thứ hai, ..., cứ như vậy ta giải quyết được bài toán tổng quát hơn là *xác định các thành phần liên thông của một đồ thị vô hướng bất kỳ*.

Ví dụ. Xác định các thành phần liên thông của đồ thị vô hướng cho bởi ma trận kề dưới đây (chỉ ghi nửa trên vì ma trận là đối xứng):

	A	B	C	D	E	F
A	0	0	0	1	0	1
B		0	0	0	0	0
C			0	0	1	0
D				0	0	1
E					0	0
F						0

Giải. Xuất phát từ A, kết quả tìm kiếm được A, D, F. Xuất phát từ B, kết quả tìm kiếm được B. Xuất phát từ C, kết quả tìm kiếm được C, E. Đến đây, tất cả các đỉnh của đồ thị đều đã được xét. Kết quả nhận được đồ thị đã cho gồm 3 thành phần liên thông là {A, D, F}, {B}, {C, E}.

Đối với đồ thị có hướng, để xác định tính liên thông mạnh, ta lặp lại thao tác đã trình bày với các đỉnh xuất phát lần lượt là các đỉnh của đồ thị. Nếu với mọi đỉnh xuất phát, tất cả cá đỉnh của đồ thị đều được thăm thì đồ thị đang xét là liên thông mạnh, trái lại, đồ thị này là không liên thông mạnh. Để xác định tính liên thông có cách làm hiệu quả hơn đòi hỏi 2 lần DFS:

- Lần 1: DFS trên G
- Lần 2: DFS trên G^T (là đồ thị thu được từ G bằng việc đảo ngược hướng trên tất cả các cung)

6.4.2. Tìm đường đi

Việc tìm đường đi từ đỉnh này đến đỉnh kia của đồ thị là một thao tác thường gặp trong nhiều ứng dụng. Trong mục này, ta giải quyết bài toán tìm đường đi với nội dung như sau:

"Cho một đồ thị (có hướng hoặc vô hướng) và hai đỉnh s, t ($s \neq t$). Hỏi rằng có đường đi từ s đến t hay không? Trong trường hợp tồn tại, hãy xác định một đường đi từ s đến t đi qua ít cạnh nhất".

Bài toán vừa phát biểu bao hàm hai nội dung:

1) Thuộc bài toán tồn tại.
2) Thuộc bài toán tối ưu.

Gợi ý:

Lời giải của bài toán như sau: xuất phát từ s, tiến hành thao tác tìm kiếm. Trong quá trình tìm kiếm, để ý hai tình huống: nếu có một lúc nào đó đỉnh t được thăm, thì khẳng định có đường đi từ s đến t, trái lại, nếu kết thúc tìm kiếm mà đỉnh t vẫn chưa được thăm thì khẳng định không có đường đi từ s đến t.

Trong trường hợp có đường đi từ s đến t, nếu thuật toán tìm kiếm được thực hiện *theo chiều rộng* thì đường đi tìm được sẽ là đường đi ngắn nhất (theo số cạnh) từ s đến t (xem lại nội dung thuật toán tìm kiếm theo chiều rộng, mục 6.2).

Chú ý: Trong thao tác tìm kiếm, ta chỉ quan tâm đến trạng thái các đỉnh đã được thăm hay chưa mà không quan tâm đến "vết" của các đường đi, vì thế để xác định các đường đi này, trong lúc tìm kiếm ta phải lưu lại "vết" của chúng bằng một thủ thuật rất hay được dùng trong việc giải các bài toán tối ưu trên đồ thị, đó là kỹ thuật "gán nhãn" cho các đỉnh ("nhãn" của đỉnh là các thông tin trung gian, phụ thuộc vào từng đỉnh, phục vụ cho việc tìm kiếm, ý nghĩa cụ thể của chúng tùy thuộc vào từng bài toán, ta sẽ thấy rõ hơn điều này trong bài 8). Ở đây, "nhãn" của đỉnh v ghi nhận đỉnh ngay trước v trên đường đi từ s đến v. Giá trị của chúng được xác định lồng vào trong từng bước của sơ đồ mô tả thuật toán tìm kiếm theo chiều rộng (xem mục 6.2) như sau:

- Đầu tiên, nhãn của mọi đỉnh được khởi gán là rỗng (với ý nghĩa là chúng chưa được thăm).
- Khi khởi gán hàng đợi Q gồm đỉnh xuất phát s, ta khởi gán nhãn của s là chính s (xem như s đã được thăm).
- Trong vòng lặp xử lý hàng đợi Q, mỗi khi lấy u ra, ta kết nạp các đỉnh v vào Q với điều kiện v kề với u và nhãn của v là rỗng (chưa được xét thăm), sau đó gán nhãn của các đỉnh v này bằng u.
- Khi kết thúc tìm kiếm, các đỉnh được thăm là những đỉnh đã được gán nhãn (nghĩa là có giá trị khác rỗng). Nhờ những nhãn này, ta xác định được các đường đi ngắn nhất từ s đến mọi đỉnh t đã được gán nhãn theo kỹ thuật "lần ngược" như sau: $t \leftarrow u_1 = \text{nhãn}(t) \leftarrow u_2 = \text{nhãn}(u_1) \leftarrow \ldots \leftarrow s = \text{nhãn}(u_k)$.

Ví dụ. Tìm đường đi ít cạnh nhất từ đỉnh A đến tất cả các đỉnh được thăm trong thí dụ mục 6.2.

Giải. Ta chỉ cần thêm vào các thao tác gán nhãn cho các đỉnh trong bảng tìm kiếm ở thí dụ đã xét (nhãn của mỗi đỉnh được viết trong cặp ngoặc vuông):

Hàng đợi Q	đỉnh lấy ra	đỉnh đưa vào
A [A]	A	B [A], C [A]
B, C	B	
C	C	D [C], E [C]
D, E	D	F [D]
E, F	E	
F	F	
rỗng		

Trong ví dụ này, tất cả các đỉnh đều được thăm và ta nhận được kết quả các đường đi ít cạnh nhất từ A đến các đỉnh này theo kỹ thuật "lần ngược":

Từ A đến B: B ← A (1 cạnh)
Từ A đến C: C ← A (1 cạnh)
Từ A đến D: D ← C ← A (2 cạnh)
Từ A đến E: E ← C ← A (2 cạnh)
Từ A đến F: F ← D ← C ← A (3 cạnh)

Chú ý:

- Nếu chỉ cần tìm một đường đi từ đỉnh xuất phát đến một đỉnh nào đó thì quá trình có thể dừng khi lần đầu tiên đỉnh này được gán nhãn mà không cần xử lý hết hàng đợi.
- Nếu thực hiện tìm kiếm theo chiều sâu, ta chỉ xác định được một đường đi nào đó từ đỉnh xuất phát đến đỉnh kết thúc mà không đảm bảo được đó là đường đi ngắn nhất.

Bài toán tìm đường đi ngắn nhất (theo nghĩa số cạnh) có nhiều mô hình ứng dụng trên thực tế. Chẳng hạn, xét bài toán dưới đây của một mạng hàng không:

"Có một mạng hàng không (chẳng hạn Vietnam Airlines), bao gồm một số sân bay và một số tuyến bay nối các sân bay này. Một hành khách muốn đi từ sân bay s đến sân bay t.

Câu hỏi đặt ra cho hệ thống mạng:

1) Khách có thể đi được không? (kể cả chấp nhận dừng lại một số sân bay trung gian để đổi tuyến).
2) Trong trường hợp đi được, khách cần đi như thế nào để số lần đổi tuyến là ít nhất?"

Bài toán được dẫn về việc tìm đường đi ít cạnh nhất từ s đến t trên đồ thị mô tả mạng.

Một bài toán khác được phát biểu trên một mạng truyền tin:

"Có một mạng truyền tin bao gồm một số trạm và một số cạnh nối các trạm này. Một trạm có thể truyền tin đến một trạm khác theo một đường truyền tin bao gồm một hay nhiều cạnh nối. Giả sử độ thất thoát thông tin trên mỗi cạnh nối là như nhau. Hỏi phải truyền tin từ trạm s đến trạm t theo đường nào để độ thất thoát thông tin là ít nhất?"

Rõ ràng bài toán được dẫn về việc tìm đường đi ít cạnh nhất từ s đến t trên đồ thị mô tả mạng. Bạn đọc có thể tìm thêm nhiều thí dụ như vậy trên thực tế.

Trong Bài 8, ta sẽ đề cập đến bài toán tìm đường đi ngắn nhất theo trọng số mà bài toán tìm đường đi ngắn nhất theo nghĩa ít cạnh nhất trong mục này, được xem như một trường hợp riêng.

6.4.3. Liệt kê (đếm) các đường đi

Trong những trường hợp tổng quát nhất của bài toán tìm đường đi, người ta vẫn chưa có một giải pháp hữu hiệu nào ngoài việc phải lựa chọn trong toàn bộ (nghĩa là phải liệt kê tất cả các đường đi có thể có). Bài toán liệt kê các đường đi có thể phát biểu như sau:

"Cho một đồ thị (có hướng hoặc vô hướng) và hai đỉnh s, t ($s \neq t$). Hãy liệt kê (và đếm) tất cả các đường đi đơn từ s đến t".

Đây là bài toán thuộc lĩnh vực liệt kê tổ hợp, trong đó mỗi đường đi đơn từ s đến t được xem như một cấu hình bao gồm một dãy đỉnh liên tiếp kề nhau (không được lặp) $x_1, x_2, ..., x_k$ với x_1 là đỉnh xuất phát s và x_k là đỉnh kết thúc t.

Để giải quyết bài toán, ta dùng mô hình quay lui đã trình bày trong Bài 3 mà thuật toán tìm kiếm theo chiều sâu được xem như một biến thể của nó.

Ví dụ. Liệt kê tất cả các đường đi đơn từ đỉnh A đến đỉnh F trong thí dụ mục 6.2.

Giải. Bằng quay lui (với trình tự duyệt các đỉnh theo thứ tự từ điển tên các đỉnh), ta nhận được 6 đường đi đơn từ A đến F như sau:

1) A → B → C → D → F
2) A → B → C → E → D → F
3) A → B → C → E → F
4) A → C → D → F
5) A → C → E → D → F
6) A → C → E → F

Bài tập 10 trong phần bài tập dưới đây cho thấy số các đường đi đơn nối giữa hai đỉnh của đồ thị nói chung là rất lớn. Điều này cho thấy các bài toán phải liệt kê tất cả các đường đi là không khả thi.

TÓM LƯỢC CUỐI BÀI

Sau khi học xong bài học này, các bạn cần ghi nhớ các vấn đề sau:

- Yêu cầu của bài toán tìm kiếm trên đồ thị.
- Các thuật toán: tìm kiếm theo chiều rộng (sử dụng hàng đợi); tìm kiếm theo chiều sâu (sử dụng ngăn xếp hoặc đệ quy).
- Một số ứng dụng cụ thể như: xác định tính liên thông; tìm, liệt kê các đường đi.

BÀI TẬP

Các bài tập dưới đây đều là những bài lập trình nhằm minh họa các thuật toán tìm kiếm và những ứng dụng của chúng. Dữ liệu nhập vào có thể tổ chức từ bàn phím hoặc từ file.

1. Viết một chương trình, nhập một đồ thị dưới dạng ma trận kề và một đỉnh xuất phát, sau đó đưa ra kết quả tìm kiếm bắt đầu từ đỉnh này. Yêu cầu chương trình cho phép hai lựa chọn: tìm kiếm theo chiều rộng và tìm kiếm theo chiều sâu.

2. Viết một chương trình, nhập một đồ thị vô hướng dưới dạng danh sách cạnh, sau đó dùng các thuật toán tìm kiếm (theo chiều rộng hoặc theo chiều sâu), xác định các thành phần liên thông của đồ thị này.

3. Viết một chương trình, nhập một đồ thị vô hướng dưới dạng danh sách cạnh, sau đó dùng các thuật toán tìm kiếm (theo chiều rộng hoặc theo chiều sâu), xác định các cạnh là cầu của đồ thị đã cho.

4. Một mạng hàng không bao gồm một số các sân bay và một số tuyến bay nối các sân bay này. Viết một chương trình nhập thông tin của mạng dưới dạng ma trận kề, sau đó cứ mỗi lần nhập sân bay xuất phát s và sân bay kết thúc t, chương trình lại đưa ra hành trình bay từ s đến t (nếu có) sao cho số lần phải chuyển tuyến là ít nhất.

5. Xét một lớp học sinh, trong đó mỗi học sinh trong lớp giữ địa chỉ của một số học sinh khác. Giả thiết rằng, một người khi nhận được một tin nào đó, đều gửi tin này cho tất cả các bạn mà mình biết địa chỉ. Viết một chương trình nhập thông tin giữ địa chỉ dưới dạng danh sách kề, sau đó xác định người cần gửi tin để số người trong lớp nhận được tin này là nhiều nhất.

6. Viết một chương trình, nhập một đồ thị dưới dạng ma trận kề, và hai đỉnh $s \neq t$, sau đó liệt kê (và đếm) tất cả các đường đi đơn từ s đến t.

7. Một mạng giao thông gồm một số thành phố và một số đoạn đường (hai chiều) nối các thành phố này. Một khách du lịch đi từ thành phố s đến thành phố t. Trên đường đi khách muốn đi qua để tham quan những thành phố khác, càng nhiều càng tốt (mỗi thành phố chỉ đi qua một lần). Viết chương trình nhập thông tin của mạng dưới dạng ma trận kề (đối xứng), các thành phố s và t ($s \neq t$), sau đó đưa ra hành trình tốt nhất thỏa mãn yêu cầu của khách.

8. Trên bàn cờ Vua kích thước $n \times n$, người ta đánh số các cột từ 1 đến n theo chiều từ trái sang phải và đánh số các dòng từ 1 đến n theo chiều từ trên xuống dưới. Mỗi ô của bàn cờ được xác định bởi tọa độ (x, y) trong đó x là chỉ số cột và y là chỉ số dòng, $1 \leq x, y \leq n$. Một quân Mã đứng ở ô (u, v) muốn di chuyển đến ô (s, t). Viết chương trình nhập kích thước n, (u, v) – ô xuất phát, (s, t) – ô cần đến, sau đó đưa ra một hành trình của Mã dưới dạng một dãy các ô kế tiếp nhau trên hành trình này (bắt đầu từ ô xuất phát và kết thúc tại ô cần đến) sao cho số bước đi của Mã trên hành trình là ít nhất (chú ý có thể có nhiều hành trình, chỉ cần đưa ra một). Thí dụ với n = 8, (u, v) = (3, 2), (s, t) = (7, 8), một hành trình tìm được là (3, 2), (4, 4), (6, 5), (5, 7), (7, 8) với số bước đi bằng 4.

9. Cho một bảng ô vuông gồm $m \times n$ ô, ô nằm trên dòng i, cột j được gọi là ô (i, j), $1 \leq i \leq m$, $1 \leq j \leq n$. Trong mỗi ô (i, j) người ta ghi một số 0 hoặc 1 (gọi là giá trị của ô) với ý nghĩa ô có

giá trị 0 là cấm vào, còn ô có giá trị 1 là cho phép vào. Từ một ô cho phép vào chỉ có thể bước sang những ô cho phép vào khác có chung cạnh với nó (theo 4 phía). Viết chương trình, nhập các giá trị của bảng dưới dạng một ma trận m dòng, n cột, sau đó tìm một con đường ngắn nhất (ít bước nhất) nếu có, đi từ một ô mép trái bảng (nằm ở cột 1) đến một ô mép phải bảng (nằm ở cột n). Chương trình thông báo hoặc không tồn tại đường đi này, hoặc đưa ra dãy các ô kế tiếp theo thứ tự trên đường đi. Ví dụ với bảng:

1	0	1	1
0	1	1	0
1	1	0	1
1	1	0	1
0	1	1	1

một đường đi tìm được là (4, 1), (4, 2), (5, 2), (5, 3), (5, 4) (chú ý có thể có nhiều đường đi ngắn nhất, chỉ cần tìm một).

Bài tập dưới đây là một bài toán đếm các đường đi, kết quả của nó cho thấy việc duyệt tất cả các đường đi là một bài toán có độ phức tạp không khả thi.

10. Gọi S_n là số các đường đi đơn giữa hai đỉnh bất kỳ khác nhau của đồ thị đầy đủ K_n.

a) Tìm một công thức truy hồi cho S_n, từ đó tính S_9.

b) Khử công thức truy hồi tìm được để nhận được một công thức tường minh tính S_n.

c) Từ công thức tường minh này, chứng minh S_n là một vô cùng lớn tương đương với $e(n-2)!$ khi $n \to \infty$.

d) Dùng chương trình viết trong bài tập 6 (với dữ liệu vào là đồ thị đầy đủ) để kiểm tra các kết quả tính S_n trong bài tập này.

CÂU HỎI THƯỜNG GẶP

1. Phân biệt sự khác nhau của định nghĩa liên thông trong đồ thị vô hướng và đồ thị có hướng?
2. Phân biệt thuật toán tìm kiếm theo chiều rộng và tìm kiếm theo chiều sâu?
3. Hai điều kiện cơ bản của thuật toán tìm kiếm là gì?