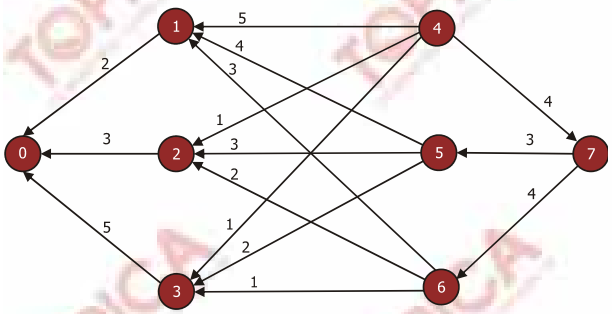
BÀI 8: MỘT SỐ BÀI TOÁN TỐI ƯU TRÊN ĐỒ THỊ TRỌNG SỐ



Nội dung

- Định nghĩa đồ thị có trọng số
- Bài toán tìm cây khung nhỏ nhất
- Bài toán tìm đường đi ngắn nhất

Giới thiệu

Một trong những ứng dụng quan trọng của lý thuyết đồ thị là nó cung cấp một số mô hình tối ưu có nhiều ý nghĩa thực tế. Trên những mô hình này, người ta đã nghiên cứu khá nhiều những thuật toán hiệu quả và cài đặt thuận tiện, được ứng dụng trong nhiều lĩnh vực hiện đại.

Do thời lượng có hạn, trong bài này, chúng tôi chỉ giới thiệu được hai mô hình tối ưu tiêu biểu của lý thuyết đồ thị là bài toán tìm cây khung nhỏ nhất và bài toán tìm đường đi ngắn nhất. Một số mô hình tối ưu nổi tiếng khác như bài toán ghép cặp, bài toán luồng cực đại, ..., chúng tôi hy vọng sẽ được giới thiệu trong những dịp thuận tiện hơn

Thời lượng học

- 6 tiết

Mục tiêu

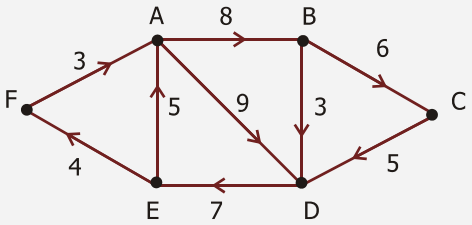
Sau khi học bài này, các bạn có thể:

- Hiểu rõ thế nào là đồ thị có trọng số.
- Biểu diễn được đồ thị có trọng số trên máy tính.
- Sử dụng được thuật toán Kruskal, thuật toán Prim để tìm cây khung nhỏ nhất.
- Sử dụng được thuật toán Dijkstra để tìm đường đi ngắn nhất.

TÌNH HUỐNG DẪN NHẬP

Tình huống: Chọn đường bay

Một mạng hàng không gồm 6 sân bay đặt tại 6 thành phố A, B, C, D, E, F và một số tuyến bay giữa các sân bay cùng với giá vé tương ứng:



Một hành khách muốn đi từ thành phố F đến thành phố C.

Câu hỏi

Hãy giúp vị hành khách đó chọn hành trình có tổng chi phí (chỉ tính giá vé) là nhỏ nhất.

Một trong những ứng dụng quan trọng của lý thuyết đồ thị là nó cung cấp một số mô hình tối ưu có nhiều ý nghĩa thực tế. Trên những mô hình này, người ta đã nghiên cứu khá nhiều những thuật toán hiệu quả và cài đặt thuận tiện, được ứng dụng trong nhiều lĩnh vực hiện đại.

Do thời lượng có hạn, trong bài này, chúng tôi chỉ giới thiệu được hai mô hình tối ưu tiêu biểu của lý thuyết đồ thị là bài toán tìm cây khung nhỏ nhất và bài toán tìm đường đi ngắn nhất. Một số mô hình tối ưu nổi tiếng khác như bài toán ghép cặp, bài toán luồng cực đại, ..., chúng tôi hy vọng sẽ được giới thiệu trong những dịp thuận tiện hơn.

8.1. Đồ thị có trọng số

8.1.1. Định nghĩa

Các mô hình tối ưu phát biểu trên đồ thị thường được xây dựng trên những tổ hợp nào đó gồm những cạnh của đồ thị. Vì thế để đánh giá độ "tốt" của những cấu hình này, Cần phải "lượng hóa" các cạnh của đồ thị tùy theo mục tiêu của bài toán. Một cách thường làm là ứng mỗi cạnh e của đồ thị với một giá trị số W(e), được gọi là *trọng số* của cạnh e. Một đồ thị mà mỗi cạnh của nó đều được gán trọng số, được gọi là *đồ thị có trọng số*. Như vậy, trọng số của đồ thị có thể xem như một hàm W, xác định trên tập cạnh và lấy giá trị trên tập số. Một cách hình thức, đồ thị có trọng số G được xem như một bộ ba (V, E, W), trong đó V là tập đỉnh, E là tập cạnh và W là trọng số. Tùy từng trường hợp mà trọng số của cạnh e được ký hiệu là W(e) hay W(u, v) với (u, v) là cặp đỉnh biểu diễn e. Chú ý rằng W(u, v) chỉ xác định tại những cặp (u, v) là một cạnh của G, ngoài ra nếu (u, v) là cạnh vô hướng thì W(u, v) = W(v, u) còn nếu (u, v) là cạnh có hướng thì có thể không có đẳng thức này. Trên hình vẽ biểu diễn đồ thị, trọng số của cạnh thường được viết bằng một con số nhỏ trên cạnh đó.

Tùy theo nội dung bài toán, trọng số của cạnh mang những ý nghĩa khác nhau. Chẳng hạn, trên đồ thị biểu diễn một mạng hàng không, trong đó mỗi đỉnh biểu diễn một sân bay, mỗi cạnh biểu diễn một tuyến bay, trọng số của cạnh có thể là thời gian bay, độ dài đường bay hay giá vé của tuyến bay tương ứng ...

Từ trọng số của cạnh, người ta xây dựng trọng số của cấu hình cần tìm (hàm mục tiêu của bài toán). Trọng số này thường được xác định bằng một công thức hoặc bằng một thủ tục tính toán, sao cho từ trọng số các cạnh thuộc cấu hình, ta hoàn toàn xác định được trọng số của cấu hình đó.

Khi đó, bài toán tối ưu trên mô hình đồ thị thường được phát biểu một cách hình thức như sau:

Tìm cấu hình X (gồm một số cạnh của đồ thị thỏa mãn những điều kiện nào đó) sao cho trọng số của X, ký hiệu W(X), đạt được nhỏ nhất (bài toán tìm min) nếu trọng số mang ý nghĩa chi phí, hoặc lớn nhất (bài toán tìm max) nếu trọng số mang ý nghĩa lợi nhuận.

8.1.2. Các cách biểu diễn đồ thị có trọng số trong máy tính

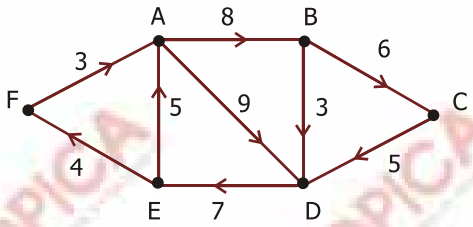
Các cách biểu diễn đồ thị có trọng số trong máy tính được mở rộng từ các cách biểu diễn đồ thị nói chung, trong đó, ngoài các thông tin về đỉnh, cạnh, cần thêm các thông tin về trọng số. Dưới đây là ba cách biểu diễn cơ bản xuất phát từ các cách biểu diễn ma trận kề, danh sách cạnh, danh sách kề của đồ thị.

- **Ma trận trọng số.**

  Ma trận trọng số là mở rộng tự nhiên của ma trận kề. Nếu như phần tử dòng i cột j của ma trận kề được xác định là 1 hay 0, tùy thuộc vào đỉnh j có kề với đỉnh i hay không thì phần tử này của ma trận trọng số chính là W(i, j).

  Chú ý: ma trận trọng số W xác định không đầy đủ, chỉ tại những vị trí (i, j) là cạnh, W(i, j) mới xác định. Trong những cài đặt, người ta thường bổ sung vào những vị trí không xác định, một giá trị đặc biệt mang tính quy ước để ma trận W là đầy đủ mà không ảnh hưởng đến tính toán chung. Rõ ràng một đồ thị có trọng số được xác định từ ma trận trọng số của nó và ngược lại. Giống như ma trận kề, nếu đồ thị là vô hướng thì ma trận trọng số của nó là đối xứng.

  Ví dụ, đồ thị có hướng dưới đây mô tả một mạng hàng không gồm 6 sân bay A, B, C, D, E, F và một số tuyến bay cùng với giá vé tương ứng (đóng vai trò trọng số):



có ma trận trọng số là (những chỗ để trống là những chỗ không xác định):

	A	B	C	D	E	F
A		8		9		
B			6	3		
C				5		
D					7	
E	5					4
F	3					

Chú ý: Ma trận này chỉ xác định tại 9 vị trí (ứng với 9 cạnh) và không đối xứng. Để bổ sung những chỗ không xác định trong những cài đặt mà vẫn đảm bảo ý nghĩa của bài toán, ta có thể xem những giá trị trên đường chéo chính bằng 0 (với ý nghĩa không tốn tiền mua vé) còn những giá trị khác ta xem bằng cực lớn (với ý nghĩa không thể mua vé được).

- **Danh sách cạnh – trọng số.**

  Danh sách cạnh thông thường được mô tả mỗi phần tử là một cặp đỉnh (có thứ tự nếu đồ thị là có hướng). Bây giờ mỗi phần tử của danh sách cạnh, ta bổ sung thêm một trường thứ ba ghi nhận trọng số cạnh tương ứng. Khi đó ta nhận được danh sách cạnh – trọng số của đồ thị đã cho. Một đồ thị có trọng số được hoàn toàn xác định khi biết danh sách cạnh-trọng số của nó và ngược lại.

  Chẳng hạn, trong thí dụ trên, danh sách cạnh – trọng số gồm 9 phần tử, mỗi phần tử gồm ba trường, hai trường đầu ghi cặp đỉnh tạo thành cạnh (theo thứ tự), trường cuối ghi trọng số của cạnh này: {AB8, AD9, BC6, BD3, CD5, DE7, EA5, EF4, FA3}.

- **Danh sách kề – trọng số.**

  Danh sách kề của mỗi đỉnh bao gồm các đỉnh kề với đỉnh đó. Bây giờ ta thêm vào mỗi phần tử của danh sách này, trọng số của cạnh tương ứng. Khi đó ta nhận được danh sách kề – trọng số của đỉnh đang xét. Tập hợp các danh sách kề-trọng số của tất cả các đỉnh của đồ thị được gọi là danh sách kề-trọng số của đồ thị. Một đồ thị có trọng số được hoàn toàn xác định khi biết danh sách kề-trọng số của nó và ngược lại.

  Chẳng hạn, danh sách kề-trọng số của đồ thị trong thí dụ trên gồm:

  - Danh sách kề-trọng số đỉnh A: {B8, D9}
  - Danh sách kề-trọng số đỉnh B: {C6, D3}
  - Danh sách kề-trọng số đỉnh C: {D5}
  - Danh sách kề-trọng số đỉnh D: {E7}
  - Danh sách kề-trọng số đỉnh E: {A5, F4}
  - Danh sách kề-trọng số đỉnh F: {A3}

Giống như danh sách kề, các danh sách này thường được cài đặt bằng cấp phát động dưới dạng một danh sách liên kết. Điều này giúp cho việc xử lý đồ thị được mềm dẻo và linh hoạt, phù hợp với những bài toán liên quan đến những đồ thị luôn thay đổi.

8.2. Bài toán tìm cây khung nhỏ nhất

8.2.1. Phát biểu bài toán

Cho một đồ thị vô hướng liên thông có trọng số:

$$G = (V, E, W)$$

trong đó V là tập đỉnh, E là tập cạnh và W là trọng số.

Với mỗi cây khung T của G, ta định nghĩa trọng số của T là W(T) xác định bởi tổng trọng số các cạnh thuộc T:

$$W(T) = \sum_{e \in T} W(e)$$

Bài toán đặt ra là tìm cây khung T của G sao cho W(T) là nhỏ nhất. Cây khung tìm được, được gọi là *cây khung nhỏ nhất* của đồ thị đã cho.

Trong bài toán tìm cây khung nhỏ nhất, một cách hình ảnh, các trọng số còn được gọi là các "độ đo" (độ đo của cạnh, độ đo của cây khung). Như vậy, cây khung nhỏ nhất là cây khung có độ đo ngắn nhất.

Thông thường, trọng số của cạnh đánh giá chi phí của cạnh đó (thời gian, tiền bạc, ...). Với ý nghĩa này, cây khung nhỏ nhất cho ta một mô hình xây dựng một mạng liên thông có chi phí rẻ nhất. Để thí dụ, ta xét bài toán xây dựng mạng đường sắt dưới đây: "Có n thành phố (đánh số từ 1 đến n). Cần xây dựng một hệ thống đường sắt nối các thành phố này sao cho từ một thành phố ta có thể đến mọi thành phố khác bằng phương tiện xe hỏa. Biết chi phí xây dựng tuyến đường nối thành phố i với thành phố j là W(i, j), $1 \le i, j \le n$. Hãy tìm một phương án xây dựng hệ thống thỏa mãn yêu cầu đã nêu với chi phí là rẻ nhất". Lời giải của bài toán là xây dựng cây khung nhỏ nhất của đồ thị đầy đủ n đỉnh (mỗi đỉnh là một thành phố) với trọng số cạnh (i, j) là W(i, j).

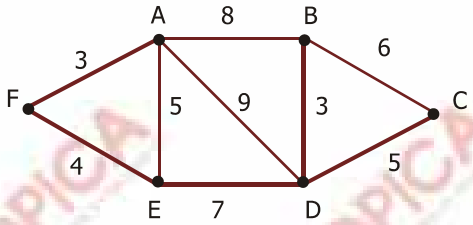
Một giải pháp tầm thường để tìm cây khung nhỏ nhất là xây dựng tất cả các cây khung của đồ thị đang xét rồi sau đó so sánh các độ đo của chúng. Tuy nhiên điều này là

không khả thi vì số lượng các cây khung là quá lớn (xem Định lý Cayley, mục 7.4.3). Dưới đây, ta xét hai thuật toán hiệu quả hơn nhiều, thường dùng để tìm cây khung ngắn nhất.

8.2.2. Thuật toán Kruskal

Thuật toán do Joshep Kruskal đưa ra năm 1956 mặc dù các ý tưởng cơ bản đã được biết sớm hơn nhiều. Thuật toán dựa vào việc kết nạp dần các cạnh vào cây khung sao cho cạnh mới kết nạp không tạo thành một chu trình đối với những cạnh đã được nạp. Để cây khung là ngắn nhất, tại mỗi bước kết nạp, Kruskal chọn cạnh có độ đo nhỏ nhất trong những cạnh chưa được xét. Quá trình kết thúc khi kết nạp được n−1 cạnh (n là số đỉnh của đồ thị). Mặc dù mang ý tưởng tham lam (tối ưu trong từng bước) nhưng thuật toán Kruskal cho lời giải đúng.

Ví dụ, xét đồ thị vô hướng có trọng số:



Sắp xếp danh sách cạnh của đồ thị theo chiều tăng dần của độ đo: AF, BD, EF, AE, CD, BC, ED, AB, AD. Lần lượt chọn từng cạnh theo thứ tự đã sắp để nạp vào cây khung sao cho cạnh được nạp không được tạo thành chu trình với những cạnh đã thuộc cây khung (nếu gặp cạnh không thỏa mãn thì bỏ qua để chọn cạnh tiếp theo). Khi được 5 cạnh (đồ thị có 6 đỉnh) thì ta nhận được cây khung nhỏ nhất. Kết quả nhận được gồm các cạnh: AF, BD, EF, (bỏ qua AE), CD, (bỏ qua BC) và ED (các cạnh tô đậm) với độ đo tổng cộng là 3 + 3 + 4 + 5 + 7 = 22.

Thuật toán Kruskal có ưu điểm là trực giác, dễ thực hiện thủ công đặc biệt đối với đồ thị ít cạnh. Tuy nhiên vấn đề cài đặt có phức tạp đôi chút trong việc kiểm tra cạnh được xét kết nạp có tạo thành chu trình với các cạnh đã kết nạp rồi hay không. Việc chứng minh tính đúng đắn và cài đặt cụ thể các bạn có thể tham khảo [2].

8.2.3. Thuật toán Prim

Thuật toán được đưa ra bởi Joshep Prim vào năm 1957. Ý tưởng của thuật toán cũng giống như Kruskal, là kết nạp dần các cạnh vào cây khung, tuy nhiên khác với Kruskal, cạnh được nạp là cạnh phải có một đỉnh nào đó đã thuộc cây khung còn đỉnh kia thì không phải và dĩ nhiên nó cũng là cạnh có độ đo nhỏ nhất trong những cạnh thỏa mãn điều kiện này. Điều kiện của cạnh được nạp trong thuật toán Prim đảm bảo cạnh này không tạo thành chu trình với bất cứ những cạnh nào đã nạp do vậy không cần kiểm tra việc tạo thành chu trình trong quá trình nạp cạnh như trong thuật toán Kruskal.

Để cài đặt thuật toán Prim, người ta dùng kỹ thuật gán nhãn cho mỗi đỉnh nhằm lưu trữ các thông tin trung gian giúp cho việc xác định tập cạnh được chọn. Nhãn của mỗi đỉnh u gồm hai thành phần, ký hiệu là a(u) và b(u), trong đó a(u) ghi nhận đỉnh sẽ nối với u để tạo thành cạnh còn b(u) ghi nhận độ đo của cạnh này. Ban đầu, nhãn các đỉnh

sẽ được khởi tạo từ một đỉnh đầu tiên nào đó được chọn vào cây khung, sau đó chúng sẽ được điều chỉnh mỗi khi xong một bước kết nạp, sao cho từ các nhãn này, ta xác định được đỉnh sẽ được nạp ở bước kế tiếp. Đỉnh đã được kết nạp vào cây khung dĩ nhiên không được xét tiếp để đảm bảo không có chu trình. Khi các đỉnh được xét hết cũng là lúc nhận được cây khung cần tìm.

Với kỹ thuật gán nhãn (có thể dùng mảng để lưu trữ các nhãn này), thuật toán Prim dễ dàng được thực hiện bằng một chương trình. Để giúp các bạn đọc thuận tiện trong việc triển khai các ứng dụng, chúng tôi giới thiệu chi tiết nội dung này.

Giả thiết đồ thị đang xét có n đỉnh và có ma trận trọng số là W (chú ý W là đối xứng). Để khỏi phải loại trừ những chỗ không xác định trong khi phát biểu thuật toán, ta bổ sung ma trận W cho đầy đủ, trong đó những giá trị trên đường chéo chính bằng 0 và những giá trị tại những chỗ không có cạnh bằng cực lớn (điều này không ảnh hưởng gì đến tính toán chung vì cây khung nhỏ nhất không bao giờ đi qua những cạnh này). Khi đó, các bước của thuật toán Prim có thể phát biểu cụ thể như sau:

- **Bước 1. Khởi tạo:** chọn đỉnh xuất phát s (tùy ý), gán nhãn cho mọi đỉnh u theo công thức: a(u) := s; b(u) := W(s, u). Cây khung T lúc ban đầu chưa có cạnh nào. Tập đỉnh còn lại U bao gồm mọi đỉnh ngoại trừ s (s là đỉnh đầu tiên vào cây khung).
- **Bước 2. Kiểm tra điều kiện dừng:** Nếu U rỗng (tất cả các đỉnh đã vào cây khung) thì kết thúc, T là cây khung nhỏ nhất cần tìm. Trái lại sang bước 3.
- **Bước 3. Kết nạp:** chọn u* trong U sao cho b(u*) đạt nhỏ nhất trong tập {b(u), u thuộc U}, sau đó kết nạp cạnh (a(u*), u*) vào T và loại u* ra khỏi U (u* đã vào cây khung).
- **Bước 4. Sửa nhãn:** với mọi đỉnh u thuộc U, thực hiện quy tắc sửa nhãn sau đây: nếu u thỏa mãn b(u) > W(u^*, u) thì sửa nhãn u theo công thức: a(u) := u^*, b(u) := W(u^*, u). Sau đó quay về bước 2 để thực hiện lần lặp mới.

Các bước của thuật toán có thể mô tả trong đoạn chương trình (mô phỏng Pascal):

```
(Khởi tạo);
WHILE (U khác rỗng) DO
BEGIN
      (Kết nạp);
      (Sửa nhãn);
END;
(ghi nhận cây khung ngắn nhất);
```

Để kiểm tra tính đúng đắn của thuật toán Prim, bạn đọc có thể tham khảo [3].

Quay lại thí dụ trên nhưng đồ thị được cho bởi ma trận trọng số W (chỉ viết một nửa vì đối xứng):

	A	B	C	D	E	F
A	0	8	∞	9	5	3
B		0	6	3	∞	∞
C			0	5	∞	∞
D				0	7	∞
E					0	4
F						0

Trong đó W đã được bổ sung đầy đủ (ký hiệu ∞ chỉ giá trị cực lớn).

Chọn đỉnh đầu tiên vào cây khung là A. Kết quả tính toán theo từng bước của thuật toán được ghi vào bảng sau (mỗi cột của bảng ứng với một đỉnh được xét, mỗi dòng của bảng ứng với các nhãn của đỉnh trong từng bước tính, đỉnh nào được chọn có dấu *, đỉnh nào bị loại (không tham gia xét tiếp) các dòng sau bỏ trắng)

B	C	D	E	F
A, 8	A, ∞	A, 9	A, 5	A, 3*
A, 8	A, ∞	A, 9	F, 4*	
A, 8	A, ∞	E, 7*		
D, 3*	D, 5			
	D, 5*			

Kết quả nhận được cây khung nhỏ nhất gồm các cạnh AF, FE, ED, DC và DB với độ đo bằng 3 + 4 + 7 + 3 + 5 = 22.

Bạn đọc có thể làm lại thí dụ trên bằng một đỉnh xuất phát khác để so sánh kết quả.

Dễ dàng thấy rằng, nếu số đỉnh của đồ thị là n thì vòng lặp thực hiện đúng n − 1 lần. Mỗi lần lặp, các khối *Kết nạp* và *Sửa nhãn* đều phải duyệt U, vì thế độ phức tạp của thuật toán Prim cài đặt theo mô hình trên, có cỡ $O(n^2)$.

Chú ý

- Cây khung nhỏ nhất được tìm là không duy nhất ngay cả khi dùng Kruskal hay Prim vì có nhiều khả năng chọn cạnh (hay đỉnh) được kết nạp.
- Khi dùng Kruskal hay Prim, không cần kiểm tra trước tính liên thông của đồ thị (để đảm bảo sự tồn tại của cây khung). Nếu duyệt hết danh sách cạnh mà không kết nạp đủ n−1 cạnh (Kruskal) hay phải kết nạp cạnh có độ đo ∞ (Prim) vào cây khung thì đồ thị đang xét là không liên thông.
- Không có giả thiết gì về dấu của trọng số, vì thế thuật toán Kruskal hay Prim cũng được dùng để tìm cây khung lớn nhất (bằng cách đổi dấu lại trọng số).

8.3. Bài toán tìm đường đi ngắn nhất

8.3.1. Phát biểu bài toán

Cho một đồ thị (có hướng hoặc vô hướng), có trọng số:

$$G = (V, E, W)$$

trong đó V là tập đỉnh, E là tập cạnh và W là trọng số.

Trọng số của đường đi P từ đỉnh s đến đỉnh t, ký hiệu W(P), được định nghĩa là tổng trọng số các cạnh thuộc P:

$$W(P) = \sum_{e \in P} W(e).$$

Đường đi P được gọi là *đường đi ngắn nhất* từ đỉnh s đến đỉnh t nếu trọng số của nó là nhỏ nhất.

Bài toán đặt ra là từ đỉnh xuất phát s cho trước, hãy tìm tất cả các đường đi ngắn nhất từ đỉnh này đến các đỉnh còn lại.

Để xác định, các đường đi xét trong bài toán này đều được giả thiết là đơn, nghĩa là trên đường đi không có đỉnh lặp lại.

Trong bài toán tìm đường đi ngắn nhất, các trọng số còn được gọi là "độ dài" (độ dài của cạnh, độ dài của đường đi) mặc dù về ý nghĩa thực tế chúng có thể là tiền bạc, thời gian, ... chứ không nhất thiết là khoảng cách. Với ý nghĩa "chi phí" như vậy, đường đi ngắn nhất mô tả đường đi tiết kiệm nhất (rẻ nhất, nhanh nhất, ...)

Trong trường hợp tổng quát, ta có thể giải bài toán này theo cách tầm thường là duyệt tất cả các đường đi đơn từ s đến t rồi so sánh độ dài của chúng. Tuy nhiên cách này không khả thi vì nói chung số đường đi đơn giữa hai đỉnh là rất lớn (xem bài tập 10 trong Bài 6). Hiện nay người ta đã có một số thuật toán hiệu quả giải bài toán đường đi ngắn nhất trong một số trường hợp riêng, có nhiều ứng dụng thực tế.

Dưới đây, chúng tôi giới thiệu một thuật toán nổi tiếng của nhà toán học người Hà lan E.Dijkstra đề xuất năm 1959 giải bài toán đường đi ngắn nhất trong trường hợp trọng số không âm (phù hợp với ý nghĩa của trọng số là chi phí).

8.3.2. Thuật toán Dijkstra

Giả sử đồ thị được xét có ma trận trọng số W (chú ý W có thể không đối xứng) và s là một đỉnh cho trước. Giả thiết W được bổ sung đầy đủ theo như cách thức đã trình bày trong mục 8.2.3. Thuật toán Dijkstra xây dựng dần các đường ngắn nhất từ s đến các đỉnh còn lại. Để lưu trữ các thông tin về những đường đi này, Dijkstra cũng dùng kỹ thuật gán nhãn cho mỗi đỉnh, trong đó nhãn của đỉnh u nhằm mô tả một đường đi nào đó từ s đến u. Nhãn này gồm hai thành phần a(u) và b(u), trong đó a(u) ghi nhận đỉnh trước u trên đường đi và b(u) ghi nhận độ dài của đường đi này. Nhãn của mỗi đỉnh u trong quá trình xây dựng có hai trạng thái: được gọi là *cố định* nếu đường đi đang xét là đường đi ngắn nhất từ s đến u và được gọi là *tạm thời* nếu trái lại. Lúc ban đầu mọi đỉnh, trừ đỉnh xuất phát s, đều là tạm thời. Sau mỗi lần lặp, các nhãn tạm thời đều được điều chỉnh theo hướng "tốt" lên và xuất hiện một đỉnh có nhãn tạm thời trở thành cố định, khi ấy đường đi ngắn nhất từ s đến đỉnh này được tìm thấy. Khi tất cả các đỉnh đều trở thành cố định thì bài toán tìm đường đi ngắn nhất từ s đến mọi đỉnh còn lại được giải quyết xong. Nội dung chi tiết của thuật toán Dijkstra được phát biểu như sau:

- **Bước 1. Khởi tạo:** xuất phát từ đỉnh s, gán nhãn cho mọi đỉnh u theo công thức a(u) := s; b(u) := W(s, u) (chú ý W có thể không đối xứng). Tập U các đỉnh có nhãn tạm thời bằng toàn bộ tập đỉnh trừ đỉnh s. Đỉnh s là đỉnh cố định đầu tiên và được ghi nhận bằng phép gán p := s (p là một biến thuộc kiểu đỉnh của đồ thị, nó xác định đỉnh được cố định).
- **Bước 2. Kiểm tra điều kiện dừng:** nếu U rỗng thì kết thúc, tất cả các đường đi ngắn nhất từ s đều được tìm xong. Trái lại sang bước 3.
- **Bước 3. Sửa nhãn:** với mọi đỉnh u thuộc U, nếu u thỏa mãn b(u) > b(p) + W(p, u) thì sửa nhãn u theo công thức a(u) := p; b(u) := b(p) + W(p, u). Đây thực chất là việc điều chỉnh đường đi: nếu đường đi đang xét từ s đến u dài hơn độ dài đường đi (ngắn nhất) từ s đến p cộng với độ dài cạnh từ p đến u thì phải thay đường đi cũ bằng đường đi này.
- **Bước 4. Cố định nhãn:** tìm p thuộc U thỏa mãn b(p) là nhỏ nhất trên tập {b(u), u thuộc U} (p là đỉnh cố định mới) và loại p ra khỏi U. Quay lại bước 2.

Sau khi kết thúc, thông tin của các đường đi ngắn nhất từ s đều được phản ánh trong các nhãn (đều là cố định) của các đỉnh: độ dài của đường đi ngắn nhất từ s đến t được cho bởi b(t), còn các đỉnh trên đường đi này được xác định bằng cách lần ngược theo thành phần thứ nhất của nhãn: $t \leftarrow u_1 = a(t) \leftarrow u_2 = a(u_1) \leftarrow ... \leftarrow s = a(u_k)$.

Các bước của thuật toán được cho bởi đoạn chương trình (mô phỏng Pascal) sau:

```
(Khởi tạo);
WHILE (U khác rỗng) DO
BEGIN
        (Sửa nhãn);
        (Cố định nhãn);
END;
(ghi nhận các đường ngắn nhất);
```

Để chứng minh tính đúng đắn của thuật toán Dijkstra, bạn đọc có thể tham khảo [2].

Ví dụ. Tìm đường đi ngắn nhất từ đỉnh A đến mọi đỉnh còn lại trên đồ thị có hướng với ma trận trọng số W (chú ý không đối xứng):

	A	B	C	D	E	F
A	0	1	∞	∞	∞	∞
B	∞	0	5	2	∞	7
C	∞	∞	0	∞	∞	1
D	2	∞	1	0	4	∞
E	∞	∞	∞	3	0	∞
F	∞	∞	∞	∞	1	0

Giải. Lập bảng (giống như trong thuật toán Prim) gồm mỗi cột ứng với một đỉnh tạm thời, mỗi dòng ứng với các nhãn của đỉnh tạm thời trong từng bước tính, đỉnh nào được cố định có dấu *, đỉnh bị loại (không tham gia xét tiếp) các dòng sau bỏ trắng:

B	C	D	E	F
A, 1*	A, ∞	A, ∞	A, ∞	A, ∞
	B, 6	B, 3*	A, ∞	B, 8
	D, 4*		D, 7	B, 8
			D, 7	C, 5*
			F, 6*	

Kết quả nhận được các đường đi ngắn nhất từ A đến các đỉnh khác là:

- Từ A đến B: B ← A (độ dài 1)
- Từ A đến C: C ← D ← B ← A (độ dài 4)
- Từ A đễn D: D ← B← A (độ dài 3)
- Từ A đến E: E ← F ← C ← D ← B ← A (độ dài 6)
- Từ A đến F: F ← C ← D ← B ← A (độ dài 5)

Giống như Prim, độ phức tạp của thuật toán Dijkstra cài đặt theo mô hình trên có cỡ $O(n^2)$, trong đó n là số đỉnh của đồ thị.

Chú ý

- Bài toán tìm đường đi ngắn nhất theo nghĩa ít cạnh nhất là trường hợp riêng của bài toán đang xét khi đặt trọng số của mỗi cạnh là 1.
- Nếu chỉ cần tìm một đường đi ngắn nhất từ s đến t thì có thể dừng khi t được cố định mà không cần làm hết bảng.
- Nếu đường đi ngắn nhất từ s đến t có chứa cạnh độ dài vô cùng thì đường đi này không tồn tại.
- Thuật toán Dijkstra chỉ đúng khi các trọng số là không âm vì thế không thể dùng nó để tìm đường đi dài nhất.

Cho đến nay, chưa có một thuật toán hữu hiệu nào giải quyết bài toán tìm đường đi ngắn nhất trong trường hợp trọng số tổng quát ngoài giải pháp tầm thường là duyệt toàn bộ. Tuy nhiên, hiện nay người ta đã đạt khá nhiều kết quả cho việc tìm hiểu và phát triển các thuật toán hiệu quả giải bài toán này trong một số trường hợp riêng có nhiều ứng dụng. Chẳng hạn trong trường hợp đồ thị có hướng không có chu trình, người ta có thuật toán tương tự như Dijkstra với độ phức tạp cỡ $O(n^2)$ giải bài toán tìm đường đi ngắn nhất với trọng số có dấu bất kỳ nghĩa là giải được bài toán tìm đường đi dài nhất. Kết quả này có một ứng dụng quan trọng trong việc giải bài toán điều khiển thực hiện các dự án lớn (PERT − Project Evalutation and Review Technique). Về những vấn đề này, bạn đọc có thể tham khảo trong [2] hay trong một số tài liệu chuyên sâu khác.

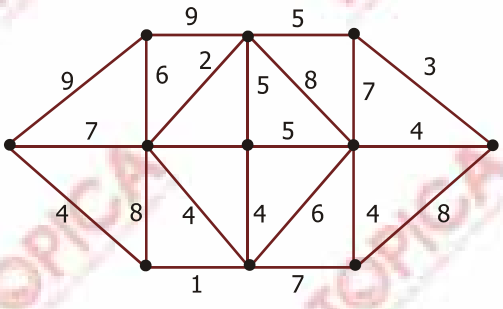
TÓM LƯỢC CUỐI BÀI

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

- Cách biểu diễn đồ thị có trọng số trên máy tính
- Tìm cây khung nhỏ nhất bằng thuật toán Kruskal, thuật toán Prim.
- Tìm đường đi ngắn nhất bằng thuật toán Dijkstra.

BÀI TẬP

1. Tìm cây khung nhỏ nhất theo thuật toán Kruskal của đồ thị có trọng số cho bởi hình vẽ dưới đây. Tô đậm những cạnh của cây khung đã tìm.



2. a) Tìm cây khung nhỏ nhất theo thuật toán Prim bằng cách lập bảng tính giá trị của các nhãn theo từng bước (như ví dụ trong bài) của đồ thị có ma trận trọng số (đối xứng) dưới đây:

	A	B	C	D	E	F	G
A	0	14	54	57	52	87	85
B	14	0	13	28	78	6	68
C	54	13	0	64	90	58	80
D	57	28	64	0	64	91	65
E	52	78	90	64	0	34	15
F	87	6	58	91	34	0	93
G	85	68	80	65	15	93	0

0

b) Viết một chương trình minh họa thuật toán Prim trong việc tìm cây khung nhỏ nhất. Chạy chương trình với thí dụ trên để kiểm tra kết quả.

3. Một mạng máy tính hoạt động liên thông, trong đó có một số máy đã được nối với nhau (hai chiều). Hàng tháng, người ta phải trả một số tiền thuê bao trên những cạnh truyền đã nối. Giả sử biết T(i, j) là số tiền thuê bao cạnh truyền nối máy i với máy j (với mọi cặp máy i, j đã nối). Sau một thời gian, người ta nhận thấy có một số cạnh truyền thừa, nghĩa là bỏ chúng đi, mạng vẫn liên thông.

a) Hãy tìm một giải pháp bỏ bớt một số cạnh truyền mà vẫn đảm bảo mạng liên thông sao cho số tiền phải thuê bao hàng tháng giảm được nhiều nhất.

b) Minh họa giải pháp đã tìm trên thí dụ cụ thể với bảng thuê bao hàng tháng T (đối xứng), các máy được chỉ số bằng các chữ cái A, B, ..., chỗ nào không nối máy thì để trống:

	A	B	C	D	E	F	G
A		4					3
B	4			12			
C				11	3		4
D		12	11		9	10	
E			3	9			8
F				10			7
G	3		4		8	7	

c) Viết một chương trình minh họa giải pháp đã tìm. Chạy chương trình với ví dụ trên.

4. a) Tìm đường đi ngắn nhất từ A đến các đỉnh còn lại theo thuật toán Dijkstra bằng cách lập bảng tính các giá trị của các nhãn theo từng bước (như thí dụ trong bài) của đồ thị có ma trận trọng số (không đối xứng) dưới đây:

	A	B	C	D	E	F	G
A		11		17	12		
B			12			10	16
C				13	14		19
D		15				17	18
E			16	11			15
F		13	18				10
G							

trong đó những chỗ để trống là những chỗ không có cạnh.

b) Viết một chương trình minh họa thuật toán Dijkstra trong việc tìm đường đi ngắn nhất với trường hợp các trọng số không âm. Chạy chương trình với thí dụ trên để kiểm tra kết quả.

5. Cho một mạng truyền tin gồm một số trạm và một số cạnh nối (hai chiều) các trạm này. Với mỗi cạnh nối trạm i với trạm j, người ta cho biết P(i, j) là độ tin cậy khi truyền tin của cạnh đó (P(i, j) được định nghĩa là một số thực nằm giữa 0 và 1 với ý nghĩa càng gần 0 thì độ tin cậy càng kém, càng gần 1 thì độ tin cậy càng cao). Một đường truyền tin từ trạm s đến trạm t là một dãy các cạnh nối kế tiếp nhau bắt đầu từ s và kết thúc tại t. Người ta định nghĩa độ tin cậy của một đường truyền tin là tích các độ tin cậy của các cạnh thuộc đường truyền tin đó.

a) Chứng minh rằng bài toán tìm đường truyền tin có độ tin cậy cao nhất từ s đến t được dẫn về bài toán tìm đường đi ngắn nhất từ s đến t.

b) Viết các bước của thuật toán Dijkstra giải bài toán đã cho.

c) Viết một chương trình minh họa các bước đã nêu trong câu b.

CÂU HỎI THƯỜNG GẶP

1. Phân tích ưu nhược điểm của phương pháp biểu diễn đồ thị trong máy tính bằng ma trận kề?

2. Phân tích ưu nhược điểm của phương pháp biểu diễn đồ thị trong máy tính bằng danh sách cạnh?

3. Phân tích ưu nhược điểm của phương pháp biểu diễn đồ thị trong máy tính bằng danh sách kề?

4. Trong đồ thị vô hướng liên thông, cây khung có phải là duy nhất không?

5. Hiểu thế nào là đồ thị có trọng số?