Nội dung

- Chương 1: Các khái niệm cơ bản
- Chương 2: Biểu diễn đồ thị
- Chương 3: Các thuật toán tìm kiếm trên đồ thị
- Chương 4: Đồ thị Euler và đồ thị Hamilton
- Chương 5: Cây và cây khung của đồ thị
- **Chương 6: Bài toán đường đi ngắn nhất**

Chương 6

BÀI TOÁN ĐƯỜNG ĐI NGẮN NHẤT

Nội dung

6.1. Bài toán đường đi ngắn nhất (ĐĐNN)

6.2. Tính chất của đường đi ngắn nhất

6.3. Tìm đường đi ngắn nhất - Thuật toán Dijkstra

6.1 Bài toán đường đi ngắn nhất

- Xét đồ thị $G=<V, E>$; trong đó $|V| = n$, $|E| = m$. Với mỗi cạnh (u, v)∈E, ta đặt tương ứng với nó một số thực *w[u,v]* được gọi là trọng số của cạnh, đặt $w[u,v]=\infty$ nếu (u, v)∉E.
- Độ dài của đường đi $P = v_1 \to v_2 \to \ldots \to v_k$ là số

$$w(P) = \sum_{i=1}^{k-1} w(v_i, v_{i+1})$$

- Đường đi ngắn nhất từ đỉnh *u* đến đỉnh *v* là đường đi có độ dài ngắn nhất trong số các đường đi nối *u* với *v*.
- Độ dài của đường đi ngắn nhất từ *u* đến *v* còn được gọi là khoảng cách từ *u* tới *v* và ký hiệu là d(*u*,*v*).

Các dạng bài toán đường đi ngắn nhất

1. **Bài toán một nguồn một đích:** Cho hai đỉnh s và t, cần tìm đường đi ngắn nhất từ s đến t.
2. **Bài toán một nguồn nhiều đích:** Cho s là đỉnh nguồn, cần tìm đường đi ngắn nhất từ s đến tất cả các đỉnh còn lại.
3. **Bài toán mọi cặp:** Tìm đường đi ngắn nhất giữa mọi cặp đỉnh của đồ thị.

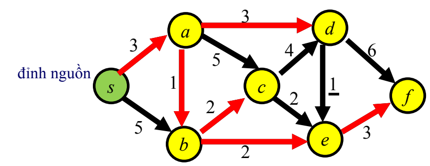
- Đường đi ngắn nhất theo số cạnh (dùng thuật toán BFS).

Nhận xét:

- Các bài toán trên được xếp theo thứ tự từ đơn giản đến phức tạp
- Nếu có thuật toán hiệu quả để giải một trong ba bài toán thì thuật toán đó cũng có thể sử dụng để giải hai bài toán còn lại.

Ví dụ

Cho đồ thị có trọng số $G = (V, E)$, và đỉnh nguồn $s \in V$, hãy tìm đường đi ngắn nhất từ s đến mỗi đỉnh còn lại.



	s	a	b	c	d	e	f
weight	0	3	4	6	6	6	9
path	s	s,a	s,a,b	s,a,b,c	s,a,d	s,a,b,e	s,a,b,e,f

Các ứng dụng thực tế

- Giao thông (Transportation)
- Truyền tin trên mạng (Network routing) (cần hướng các gói tin đến đích trên mạng theo đường nào?)
- Truyền thông (Telecommunications)
- Speech interpretation (best interpretation of a spoken sentence)
- Điều khiển robot (Robot path planning)
- Medical imaging
- Giải các bài toán phức tạp hơn trên mạng
- ...

6.2 Các tính chất của ĐĐNN

- ***Tính chất* 1.** *Đường đi ngắn nhất luôn có thể tìm trong số các đường đi đơn.*
- ***Tính chất* 2.** *Mọi đường đi ngắn nhất trong đồ thị G đều đi qua không quá n-1 cạnh, trong đó n là số đỉnh.*
- ***Tính chất 3:*** Giả sử $P = \langle v_1, v_2, \ldots, v_k \rangle$ là đđnn từ v_1 đến v_k. Khi đó, $P_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ là đđnn từ v_i đến v_j, với $1 \le i \le j \le k$.

  Hay có thể phát biểu:

  Mọi đoạn đường con của đường đi ngắn nhất đều là đường đi ngắn nhất.

6.2 Các tính chất của ĐĐNN

Ký hiệu: d(u, v) = độ dài đđnn từ u đến v (gọi là khoảng cách từ u đến v)

Hệ quả: Giả sử P là đđnn từ s tới v, trong đó $P = s \xrightarrow{p'} u \rightarrow v$.
Khi đó d(s, v) = d(s, u) + $w(u, v)$.

Tính chất 4: Giả sử $s \in V$. Đối với mỗi cạnh $(u,v) \in E$, ta có
$$d(s, v) \le d(s, u) + w(u,v).$$

6.3 ĐĐNN XUẤT PHÁT TỪ MỘT ĐỈNH
Single-Source Shortest Paths

- Xét đỉnh bắt đầu là s. Các thuật toán tìm đường đi ngắn nhất xuất phát từ s sử dụng hai mảng:

  - ***$d(v)$* - độ dài đường đi từ *s* đến *v* ngắn nhất hiện biết**
  - ***truoc*(v) - Lưu đỉnh đi trước *v* trong đường đi ngắn nhất**
    (lưu vết đường đi, dùng để truy lại đường đi từ s) .

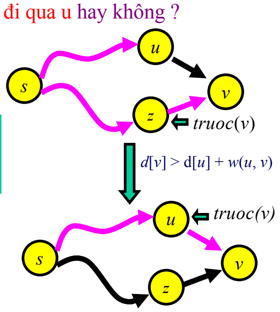
- Khởi tạo ban đầu (Initialization):

  for $v \in$ V(G) do
  { $d[v] \leftarrow \infty$; $truoc[v] \leftarrow 0$; }
  $d[s] \leftarrow 0$;

Tìm đường đi tốt hơn

Sử dụng cạnh (u, v) để kiểm tra xem đường đi đến v đã tìm được có thể làm ngắn hơn nhờ đi qua u hay không ?

if $d[v] > d[u] + w(u, v)$
then $d[v] \leftarrow d[u] + w(u, v)$
$truoc[v] \leftarrow u$



Tìm đường đi tốt hơn

- Các thuật toán thực hiện ***gán nhãn*** cho mỗi đỉnh:
  - ✓Mỗi đỉnh *v* có nhãn gồm 2 thành phần: ***(d[v], truoc[v]).***
  - ✓Nhãn sẽ biến đổi trong quá trình thực hiện thuật toán
- Chú ý:
  - Để tính khoảng cách từ s đến t ta phải tính khoảng cách từ s đến tất cả các đỉnh còn lại của đồ thị.
  - Hiện nay vẫn chưa biết thuật toán nào cho phép tìm đđnn giữa hai đỉnh làm việc thực sự hiệu quả hơn những thuật toán tìm đđnn từ một đỉnh đến tất cả các đỉnh còn lại.

Thuật toán Dijkstra

Edsger W.Dijkstra

(1930-2002)

- ❑ Trong trường hợp trọng số trên các cung là không âm, thuật toán do Dijkstra đề nghị là thuật toán hiệu quả để tìm đường đi ngắn nhất từ 1 đỉnh đến các đỉnh còn lại.
- ❑ Thuật toán thực hiện gán nhãn cho các đỉnh:
  - ✓ Ban đầu, nhãn của các đỉnh là tạm thời.
  - ✓ Ở mỗi một bước lặp, có một đỉnh được chọn để chuyển nhãn tạm thời thành nhãn cố định.
  - ✓ Nếu nhãn của một đỉnh *u* trở thành cố định thì $d[u]$ sẽ là độ dài của đđnn từ đỉnh *s* đến *u*.
  - ✓ Thuật toán kết thúc khi tất cả các đỉnh có nhãn cố định.

Thuật toán Dijkstra

Đầu vào: *Đồ thị có hướng G=(V,E) với n đỉnh,*

s ∈ V là đỉnh xuất phát,

w[u,v], u,v ∈ V - ma trận trọng số;

Giả thiết: ***w[u,v] ≥ 0, u, v ∈ V.***

Đầu ra: *Với mỗi v ∈ V*

d[v] = d(s, v) - độ dài đường đi ngắn nhất từ s đến v

truoc[v] - đỉnh đi trước đỉnh v trong đđnn từ s đến v.

Thuật toán Dijkstra

Tập S: Chỉ dùng để dễ mô tả thuật toán

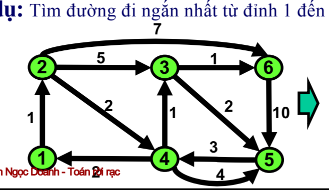
```
procedure Dijkstra;
begin
    for  v ∈ V  do  begin     (*  Khëi t¹o  *)
                d[v] := w[s,v] ;  truoc[v]:=s;
             end;
    d[s] := 0;   S := {s};            (*  S -  tËp ®Ønh cã nh·n cè ®Þnh *)
    T := V \ {s};                     (*   T lµ tËp c¸c ®Ønh cã nh·n t¹m thêi   *)
    while  T  ≠ ∅  do                  (* Bước lặp *)
    begin
        T×m ®Ønh  u ∈ T tho¶ m·n  d[u] = min{ d[z] :  z  ∈  T};
        T :=  T \ {u};  S:= S  ∪ {u};     (*  Cè ®Þnh nh·n cña ®Ønh u *)
        for   v ∈ T   do                  (*  G¸n nh·n l¹i cho c¸c ®Ønh trong T *)
            if   d[v] > d[u] + w[u,v]   then
                begin
                    d[v] := d[u] + w[u,v] ;  truoc[v] := u ;
                end;
    end;
end;
```


Thuật toán Dijkstra

- Kết thúc thuật toán trả về mảng d[] và mảng truoc[], mỗi mảng gồm *n* phần tử (tương ứng với *n* đỉnh).

  Đường đi ngắn nhất từ đỉnh s được lấy ra từ 2 mảng này.
- **Chú ý:**

  Nếu chỉ cần tìm đường đi ngắn nhất từ *s* đến *t* thì có thể dừng thuật toán khi đỉnh t trở thành có nhãn cố định.
- **Ví dụ:** Tìm đường đi ngắn nhất từ đỉnh 1 đến các đỉnh của đồ thị



W=

	1	2	3	4	5	6
1	0	1	∞	∞	∞	∞
2	∞	0	5	2	∞	7
3	∞	∞	0	∞	2	1
4	2	∞	1	0	4	∞
5	∞	∞	∞	3	0	∞
6	∞	∞	∞	∞	10	0

Ví dụ

Sửa nhãn:

```
if  d[v] > d[u] + w[u,v]   then
    begin
        d[v] := d[u] + w[u,v] ;
        truoc[v] := u ;
    end;
```

W=

	1	2	3	4	5	6
1	0	1	∞	∞	∞	∞
2	∞	0	5	2	∞	7
3	∞	∞	0	∞	2	1
4	2	∞	1	0	4	∞
5	∞	∞	∞	3	0	∞
6	∞	∞	∞	∞	10	0

	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
Khởi tạo	[0, 1]	[1, 1]*	[∞, 1]	[∞, 1]	[∞, 1]	[∞, 1]
1						
2						
3						
4						
5						

Ví dụ

Sửa nhãn:

```
if  d[v] > d[u] + w[u,v]   then
    begin
        d[v] := d[u] + w[u,v] ;
        truoc[v] := u ;
    end;
```

W=

	1	2	3	4	5	6
1	0	1	∞	∞	∞	∞
2	∞	0	5	2	∞	7
3	∞	∞	0	∞	2	1
4	2	∞	1	0	4	∞
5	∞	∞	∞	3	0	∞
6	∞	∞	∞	∞	10	0

	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
Khởi tạo	[0, 1]	[1, 1]*	[∞, 1]	[∞, 1]	[∞, 1]	[∞, 1]
1	-	-	[6, 2]	[3, 2]*	[∞, 1]	[8, 2]
2						
3						
4						
5						

Ví dụ

Sửa nhãn:

```
if  d[v] > d[u] + w[u,v]   then
    begin
        d[v] := d[u] + w[u,v] ;
        truoc[v] := u ;
    end;
```

W=

	1	2	3	4	5	6
1	0	1	∞	∞	∞	∞
2	∞	0	5	2	∞	7
3	∞	∞	0	∞	2	1
4	2	∞	1	0	4	∞
5	∞	∞	∞	3	0	∞
6	∞	∞	∞	∞	10	0

	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
Khởi tạo	[0, 1]	[1, 1]*	[∞, 1]	[∞, 1]	[∞, 1]	[∞, 1]
1	-	-	[6, 2]	[3, 2]*	[∞, 1]	[8, 2]
2	-	-	[4, 4]*	-	[7, 4]	[8, 2]
3						
4						
5						

Ví dụ

Sửa nhãn:

```
if  d[v] > d[u] + w[u,v]   then
    begin
        d[v] := d[u] + w[u,v] ;
        truoc[v] := u ;
    end;
```

W=

	1	2	3	4	5	6
1	0	1	∞	∞	∞	∞
2	∞	0	5	2	∞	7
3	∞	∞	0	∞	2	1
4	2	∞	1	0	4	∞
5	∞	∞	∞	3	0	∞
6	∞	∞	∞	∞	10	0

	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
Khởi tạo	[0, 1]	[1, 1]*	[∞, 1]	[∞, 1]	[∞, 1]	[∞, 1]
1	-	-	[6, 2]	[3, 2]*	[∞, 1]	[8, 2]
2	-	-	[4, 4]*	-	[7, 4]	[8, 2]
3	-	-	-	-	[6, 3]	[5, 3]*
4						
5						

Ví dụ

Sửa nhãn:
```
if  d[v] > d[u] + w[u,v]   then
    begin
        d[v] := d[u] + w[u,v] ;
        truoc[v] := u ;
    end;
```

W=

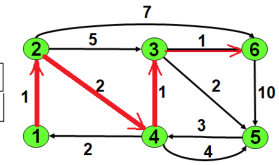
	1	2	3	4	5	6
1	0	1	∞	∞	∞	∞
2	∞	0	5	2	∞	7
3	∞	∞	0	∞	2	1
4	2	∞	1	0	4	∞
5	∞	∞	∞	3	0	∞
6	∞	∞	∞	∞	10	0

	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
Khởi tạo	[0, 1]	[1, 1]*	[∞, 1]	[∞, 1]	[∞, 1]	[∞, 1]
1	-	-	[6, 2]	[3, 2]*	[∞, 1]	[8, 2]
2	-	-	[4, 4]*	-	[7, 4]	[8, 2]
3	-	-	-	-	[6, 3]	[5, 3]*
4	-	-	-	-	[6, 3]*	-
5						

Ví dụ (tiếp)

Kết thúc, thuật toán trả về 2 mảng:

v	1	2	3	4	5	6
d[v]	0	1	4	3	6	5
truoc[v]	1	1	4	2	3	3



Ví dụ, đường đi ngắn nhất từ đỉnh s=1 đến đỉnh t=6 là:
- ✓ Độ dài d[t]=d[6]=5
- ✓ Đường đi:

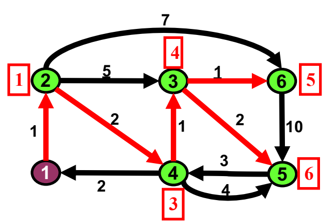
  t= 6← truoc[6]=3 ← truoc[3]=4 ← truoc[4]=2 ← truoc[2]=1=s

Cây đường đi ngắn nhất

- Tập cạnh { (*truoc*(*v*), *v*): *v*∈*V* \ {s} } tạo thành cây có gốc tại đỉnh nguồn *s* được gọi là cây đđnn xuất phát từ đỉnh *s*.

✓ Các cạnh màu đỏ tạo thành cây đđnn xuất phát từ đỉnh 1

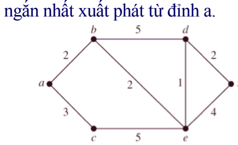
✓ Số màu đỏ viết bên cạnh mỗi đỉnh là độ dài đường đi ngắn nhất từ 1 đến nó.



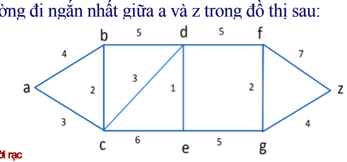
- Từ cây đđnn dễ dàng suy ra đường đi ngắn nhất từ đỉnh 1 đến các đỉnh còn lại.

Bài tập

1. Tìm đường đi ngắn nhất từ đỉnh a đến các đỉnh còn lại trong đồ thị. Vẽ cây đường đi ngắn nhất xuất phát từ đỉnh a.



2. Tìm đường đi ngắn nhất giữa a và z trong đồ thị sau:



Bài tập

3. Tìm đường đi ngắn nhất từ đỉnh 1 đến đỉnh 7 trong đồ thị cho bởi ma trận trọng số sau.

$$\begin{vmatrix} 00 & 11 & 65 & 17 & 65 & 65 & 65 \\ 65 & 00 & 12 & 65 & 65 & 10 & 16 \\ 65 & 65 & 00 & 13 & 14 & 65 & 19 \\ 65 & 65 & 65 & 00 & 65 & 65 & 18 \\ 65 & 65 & 65 & 65 & 00 & 65 & 15 \\ 65 & 13 & 18 & 65 & 65 & 00 & 10 \\ 65 & 65 & 65 & 65 & 65 & 65 & 00 \end{vmatrix}$$

Nguyễn Ngọc Doanh - Toán rời rạc