TOÁN RỜI RẠC 2

CHƯƠNG 4

Giảng viên: Vũ Văn Thỏa

CHƯƠNG 4:
BÀI TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT

- Khái niệm
- Đường đi ngắn nhất xuất phát từ 1 đỉnh
- Đường đi ngắn nhất giữa các cặp đỉnh

4.1 Khái niệm

- Độ dài đường đi trên đồ thị có trọng số

Cho G= (V, E) là một đồ thị có trọng số (có thể vô hướng hoặc có hướng).

Xét đường đi (P) từ đỉnh u tới đỉnh v $\in$ G gồm dãy các đỉnh $x_0, x_1, \ldots, x_k$, trong đó $x_0 = u$, $x_k = v$ và $(x_{i-1}, x_i) \in E$, $1 \le i \le k$.

Độ dài đường đi (P) là tổng các trọng số của tất cả các cạnh $(x_{i-1}, x_i) \in E$, $1 \le i \le k$.

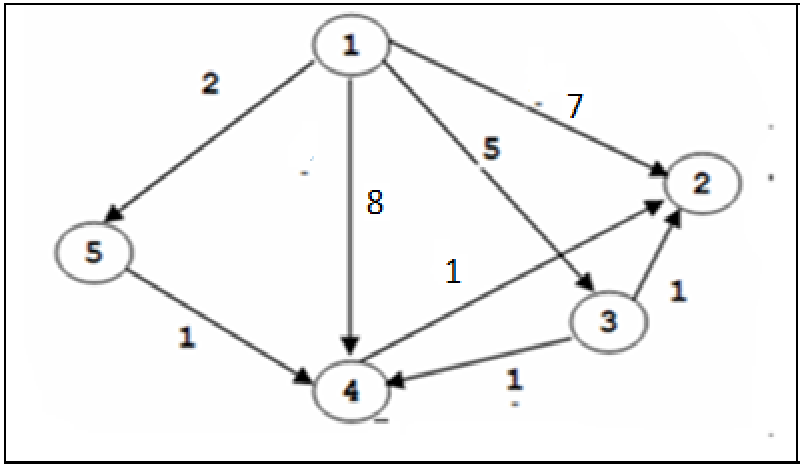
Đường đi ngắn nhất

- Đường đi ngắn nhất từ đỉnh u đến đỉnh v trên đồ thị G là đường đi có độ dài nhỏ nhất.
- **Bài toán tìm đường đi ngắn nhất**:

(1) Tìm đường đi ngắn nhất xuất phát từ đỉnh s đến các đỉnh còn lại;

(2) Tìm đường đi ngắn nhất giữa các cặp đỉnh (u, v)

Ví dụ 1: Độ dài đường đi



	1	2	3	4	5
1	0	7	5	8	2
2	∞	0	∞	∞	∞
3	∞	1	0	1	∞
4	∞	1	∞	0	∞
5	∞	∞	∞	1	0

Đường đi từ đỉnh 1 đên đỉnh 4

(1): Đường đi 1→3→4 có độ dài 5+1= 6;

(2): Đường đi 1→4 có độ dài 8;

(3): Đường đi 1→5→4 có độ dài 2+1= 3;

⇒ Đường đi ngắn nhất từ 1 đến 4 là 1→5→4

4.2 Đường đi ngắn nhất xuất phát từ 1 đỉnh

- Thuật toán Dijkstra
- Thuật toán Bellman-Ford

4.2.1 Thuật toán Dijkstra

- Đặt bài toán
- Mô tả thuật toán
- Kiểm nghiệm thuật toán
- Cài đặt thuật toán
- Đánh giá thuật toán

1) Đặt bài toán:

Input: Đồ thị G gồm n đỉnh cho bởi ma trận trọng số a[][] với các phần tử ≥ 0, trong đó

a[i][j]= c_{ij} nếu cạnh nối i với j có trọng số c_{ij};

a[i][j]= ∞ nếu không có cạnh nối i với j;

Đỉnh s $\in$ G;

Output: Độ dài d[v] đường đi ngắn nhất từ s đến v và e[v] là đỉnh của cạnh (e[v], v) thuộc đường đi từ s đến v;

2) Mô tả thuật toán

Khởi tạo: $\forall v \in G$: d[v]= a[s][v]; e[v]= s; vs[v]= 0;

(1) Bắt đầu tìm kiếm từ s: d[s]= 0; e[s]= 0; vs[s]= 1;

(2) Tìm đỉnh u sao cho d[u]= min{d[i] | vs[i] = 0};

Nếu không tìm được thì chuyển sang (5). Nếu tìm được thì sang (3).

(3) Đặt vs[u]= 1;

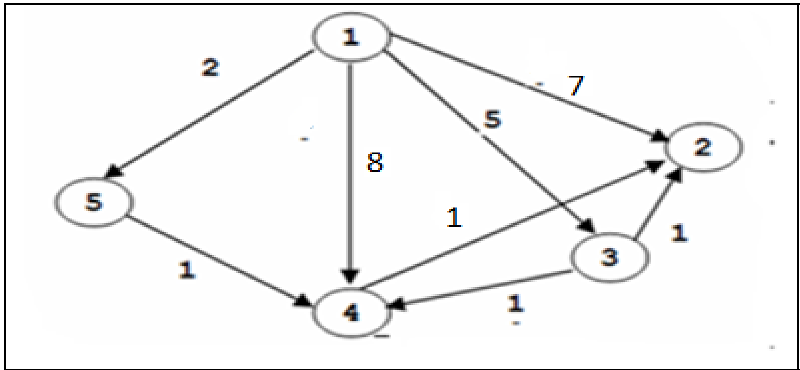
(4) Đối với tất cả v $\in$ G thỏa mãn (vs[v]= 0) & (d[v]> d[u] + a[u][v]) thì thay thế:

e[v]= u; d[v]= d[u] + a[u][v]; và quay lại (2).

(5) Xuất d[v] và đường đi từ s đến v.

3) Kiểm nghiệm thuật toán

Ví dụ 2: Tìm đường đi ngắn nhất từ đỉnh s= 1 trong đồ thị G:

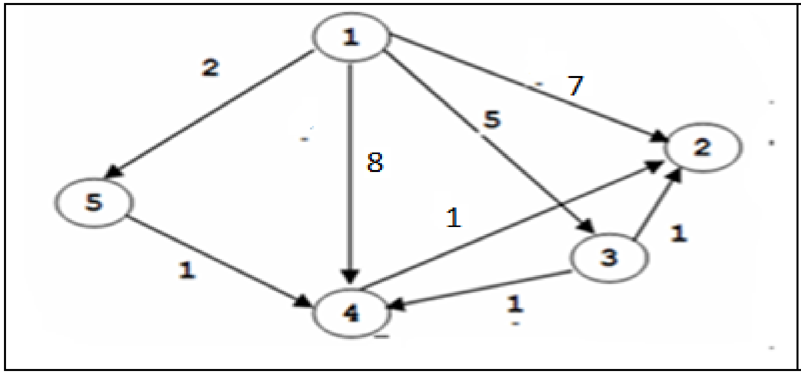


	1	2	3	4	5
1	**0**	7	5	8	2
2	∞	**0**	∞	∞	∞
3	∞	1	**0**	1	∞
4	∞	1	∞	**0**	∞
5	∞	∞	∞	1	**0**

<u>**Giải**</u>

Có n= 5; s= 1

Ví dụ 2: Đường đi ngắn nhất từ s= 1



	1	2	3	4	5
1	0	7	5	8	2
2	∞	0	∞	∞	∞
3	∞	1	0	1	∞
4	∞	1	∞	0	∞
5	∞	∞	∞	1	0

Lập bảng:

Bước	d[1]|e[1]	d[2]|e[2]	d[3]|e[3]	d[4]|e[4]	d[5]|e[5]	Đỉnh được gán nhãn
1	0 | 0	7 | 1	5 | 1	8 | 1	2 | 1	1
2		7 | 1	5 | 1	3 | 5	2 | 1	5
3		4 | 4	5 | 1	3 | 5		4
4		4 | 4	5 | 1			2
5			5 | 1			3

Ví dụ 2: Đường đi ngắn nhất từ s= 1

Bước	d[1]|e[1]	d[2]|e[2]	d[3]|e[3]	d[4]|e[4]	d[5]|e[5]	Đỉnh được gán nhãn
1	0 | O	7 | 1	5 | 1	8 | 1	2 | 1	1
2		7 | 1	5 | 1	3 | 5	2 | 1	5
3		4 | 4	5 | 1	3 | 5		4
4		4 | 4	5 | 1			2
5			5 | 1			3

Kết luận:

Đường đi từ 1 đến 2: 2←4←5 ←1; Độ dài d[2]= 4

Đường đi từ 1 đến 3: 3←1; Độ dài d[3]= 5

Đường đi từ 1 đến 4: 4←5 ←1; Độ dài d[4]= 3

Đường đi từ 1 đến 5: 5 ←1; Độ dài d[5]= 2

4) Cài đặt thuật toán

```
int n, s, a[100][100], d[100], e[100], vs[100];
void Dijkstra(int s){int u, v;
 for (v=1; v<= n; v++){d[v]= a[s][v]; e[v]=s;}
    d[s]= 0; e[s] = 0; vs[s]= 1;
while (1){int u= 0, min= 32767;
 for (v=1; v<= n; v++) if (vs[v]==0 && d[v]< min){
         u= v; min= d[v];}
      if (u== 0) return; vs[u]= 1;
 for (v=1; v<=n; v++)
  if (vs[v]== 0 && d[v]> d[u]+a[u][v]) {d[v]= d[u] + a[u][v]; e[v] = u; }
  }
}
```

5) Đánh giá thuật toán

- Thuật toán Dijkstra có độ phức tạp $O(n^2)$, n là số đỉnh của đồ thị G.
- Trong trường hợp cài đặt tối ưu, thuật toán Dijkstra có độ phức tạp O(nlogn), n là số đỉnh của đồ thị G.

Ghi chú:

Trong thực tế thường sử dụng giải thuật trên vào bài toán tìm đường đi ngắn nhất từ đỉnh s đến t:

- **Input**: Đồ thị G gồm n đỉnh cho bởi ma trận trọng số a[][] với các phần tử ≥ 0, trong đó a[i][j]= max nếu không có cạnh nối i với j; Hai đỉnh s và t;
- **Output**: Độ dài ngắn nhất d[t] và đường đi từ s đến t.

Thuật toán tìm đường đi ngắn nhất từ đỉnh s đến đỉnh t

Khởi tạo: với tất cả v ∈ G, d[v]= a[s][v]; e[v]= s; vs[v]= 0;

(1) Bắt đầu tìm kiếm từ s: d[s]= 0; e[s]= 0; vs[s]= 1;

(2) Tìm đỉnh u sao cho d[u]= min{d[v] | vs[v] = 0}. Nếu không tìm được thì chuyển sang (5). Nếu tìm được thì sang (3).

(3) Đặt vs[u]= 1. Nếu u= t thì chuyển sang (5); ngược lại chuyển sang (4);

(4) Đối với tất cả v ∈ G thỏa mãn (vs[v]= 0) & (d[v]> d[u] + a[u][v]) thì thay thế:

e[v]= u; d[v]= d[u] + a[u][v]; và quay lại (2);

(5) Nếu d[t] < max thì xuất d[t] và đường đi từ s đến t; nếu ngược lại xuất không có đường đi từ s đến t.

4.2.2 Thuật toán Bellman-Ford

- Đặt bài toán
- Mô tả thuật toán
- Kiểm nghiệm thuật toán
- Cài đặt thuật toán
- Đánh giá thuật toán

1) Đặt bài toán

- **Input**: Đồ thị G gồm n đỉnh cho bởi ma trận trọng số a[][] không chứa chu trình âm, trong đó a[i][j]= max nếu không có cạnh nối i với j; Đỉnh s;
- **Output**: Độ dài d[v] đường đi ngắn nhất từ s đến v và e[v] là đỉnh đầu cạnh (e[v], v) trên đường đi từ s đến v.

Nhận xét:

- Nếu G chứa chu trình âm thì sẽ tồn tại đỉnh v sao cho d[v] $\rightarrow$ -$\propto$ khi thực hiện liên tiếp các phép duyệt các đỉnh theo chu trình âm.
- Nếu G không chứa chu trình âm thì với mọi đỉnh v đều có d[v] > -$\propto$.

2) Mô tả thuật toán

Khởi tạo: d[v]= a[s][v]; e[v]= s;

(1) d[s] = 0, e[s] = 0; ok= 0;

(2) Thực hiện n-1 lần lặp:

(2.1) ok= 1;

(2.2) Với mọi đỉnh v $\in$ V thực hiện:

Với mọi đỉnh u $\in$ V thực hiện:

Nếu (d[v]> d[u] + a[u][v]) thì thay thế:

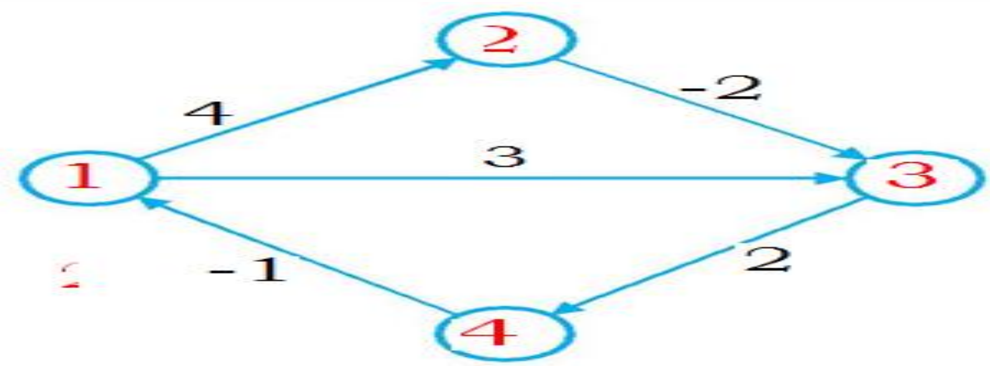
{ e[v]= u; d[v]= d[u] + a[u][v]; ok= 0; }

(2.3) Nếu (ok= 1) chuyển (3);

(3) Nếu (ok= 1) Xuất d[v] và e[v];

Nếu (ok= 0) Xuất thông báo G chứa chu trình âm;

Ví dụ 3: Tìm đường đi ngắn nhất từ s= 1 trên G:



Giải

G có số đỉnh n= 4 và không chứa chu trình âm.

Đỉnh xuất phát s= 1.

Cạnh nối đến 1: 4→1, trọng số -1; Cạnh nối đến 2: 1→2, trọng số 4; Cạnh nối đến 3: 1→3, trọng số 3; 2→3, trọng số -2;

Cạnh nối đến 4: 3→4, trọng số 2.

Lập bảng:

- Cạnh nối đến 1: 4→1, trọng số -1;
- Cạnh nối đến 2: 1→2, trọng số 4;
- Cạnh nối đến 3: 1→3, trọng số 3; 2→3, trọng số -2;
- Cạnh nối đến 4: 3→4, trọng số 2.

Bước	d[1] | e[1]	d[2] | e[2]	d[3] | e[3]	d[4] | e[4]	ok?
Khởi tạo	0 | 0	4 | 1	3 | 1	∞ | 1	0
1	0 | 0	4 | 1	2 | 2	4 | 3	0
2	0 | 0	4 | 1	2 | 2	4 | 3	1

Kết luận:

Bước	d[1] | e[1]	d[2] | e[2]	d[3] | e[3]	d[4] | e[4]	ok?
Khởi tạo	0 | 0	4 | 1	3 | 1	∞ | 1	0
1	0 | 0	4 | 1	2 | 2	4 | 3	0
2	0 | 0	4 | 1	2 | 2	4 | 3	1

- Đường đi từ 1 đến 2: $2 \leftarrow 1$, độ dài d[2]= 4
- Đường đi từ 1 đến 3: $3 \leftarrow 2 \leftarrow 1$, độ dài d[3]= 2
- Đường đi từ 1 đến 4: $4 \leftarrow 3 \leftarrow 2 \leftarrow 1$, độ dài d[4]= 4

4) Cài đặt thuật toán

```
int n, s, a[100][100], d[100], e[100];
int BellmanFord(int s){int dem, u, v;
    for (v=1; v<= n; v++){d[v]= a[s][v]; e[v]=s;}
    d[s]= 0; e[s] = 0; int ok= 0;
      for (dem=1; dem<= n-1; dem++){ int ok= 1;
         for (v=1; v<=n; v++)
             for (u=1; u<=n; u++)
                   if (d[v] > d[u] + a[u][v]) {
                         d[v] = d[u] + a[u][v]; e[v] = u;  ok= 0;}
        if (ok== 1) return(1);}
          return(0);   }
```

5) Đánh giá thuật toán

- Thuật toán Bellman-ford có độ phức tạp $O(n^3)$, với n là số đỉnh của đồ thị G.

4.3 Đường đi ngắn nhất giữa các cặp đỉnh

- Đặt bài toán
- Thuật toán Floyd

4.3.1 Đặt bài toán

- **Input**: Đồ thị G gồm n đỉnh cho bởi ma trận trọng số a[][] không chứa chu trình âm, trong đó a[i][j]= max nếu không có cạnh nối đỉnh i với j;
- **Output**: Độ dài ngắn nhất d[i][j] của đường đi từ i đến j và e[i][j] là đỉnh đầu cạnh (e[i][j], j) trên đường đi từ i đến j.

Phương pháp giải

- Có thể sử dụng Dijstra(s) với 1≤ s ≤ n; Độ phức tạp tính toán là $O(n^3)$.
- Có thể sử dụng BellmanFord(s) với 1≤ s ≤ n; Độ phức tạp tính toán là $O(n^4)$.
- Thường sử dụng thuật toán Floyd.

4.3.2 Thuật toán Floyd

- Mô tả thuật toán
- Kiểm nghiệm thuật toán
- Cài đặt thuật toán
- Đánh giá thuật toán

1) Mô tả thuật toán

- Khởi tạo: d[i][j]= a[i][j]; e[i][j]= i;
- Với mọi k $\in$ G, i $\in$ G, j $\in$ G sao cho

  (d[i][j]> d[i][k] + d[k][j]) thì thay thế:

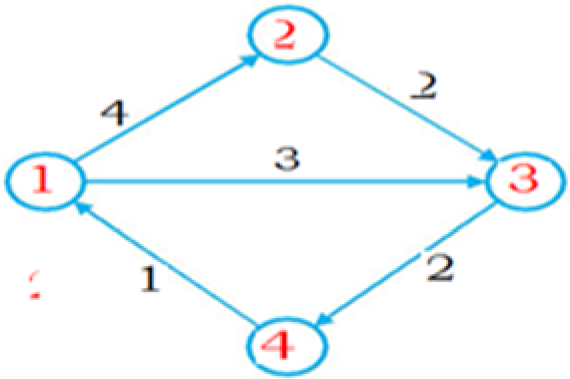
  e[i][j]= k; d[i][j]= d[i][k] + d[k][j];
- Xuất kết quả:

  + Nếu có đỉnh u mà d[u][u] < 0 thì xuất thông báo G chứa chu trình âm;

  + Ngược lại xuất d[i][j] và e[i][j].

2) Kiểm nghiệm thuật toán

Ví dụ 4: Cho G



	1	2	3	4
1	0	4	3	∞
2	∞	0	2	∞
3	∞	∞	0	2
4	1	∞	∞	0

Tìm đường ngắn nhất đi giữa các cặp đỉnh sử dụng Floyd.

Lập bảng:

Khởi tao:

	1	2	3	4
1	0|1	4|1	3|1	∞|1
2	∞|2	0|2	2|2	∞|2
3	∞|3	∞|3	0|3	2|3
4	1|4	∞|4	∞|4	0|4

k= 1:

	1	2	3	4
1	0|1	4|1	3|1	∞|1
2	∞|2	0|2	2|2	∞|2
3	∞|3	∞|3	0|3	2|3
4	1|4	5|1	4|1	0|4

k= 2:

	1	2	3	4
1	0|1	4|1	3|1	∞|1
2	∞|2	0|2	2|2	∞|2
3	∞|3	∞|3	0|3	2|3
4	1|4	5|1	4|1	0|4

k= 3:

	1	2	3	4
1	0|1	4|1	3|1	5|3
2	∞|2	0|2	2|2	4|3
3	∞|3	∞|3	0|3	2|3
4	1|4	5|1	4|1	0|4

k= 4:

	1	2	3	4
1	0|1	4|1	3|1	5|3
2	5|4	0|2	2|2	4|3
3	3|4	7|4	0|3	2|3
4	1|4	5|1	4|1	0|4

Kết luận:

k= 4:

	1	2	3	4
1	0|1	4|1	3|1	5|3
2	5|4	0|2	2|2	4|3
3	3|4	7|4	0|3	2|3
4	1|4	5|1	4|1	0|4

Đường đi từ 1 đến 2: 2←1; d[1][2]= 4 Đường đi từ 1 đến 3: 3←1; d[1][3]= 3 Đường đi từ 1 đến 4: 4←3←1; d[1][4]= 5	Đường đi từ 2 đến 1: 1←4←3←2; d[2][1]= 5 Đường đi từ 2 đến 3: 3←2; d[2][3]= 2 Đường đi từ 2 đến 4: 4←3←2; d[2][4]= 4
Đường đi từ 3 đến 1: 1←4←3; d[3][1]= 3 Đường đi từ 3 đến 2: 2←4←3; d[3][2]= 7 Đường đi từ 3 đến 4: 4←3; d[3][4]= 2	Đường đi từ 4 đến 1: 1←4; d[4][1]= 1 Đường đi từ 4 đến 2: 2←1←4; d[4][2]= 5 Đường đi từ 4 đến 3: 3←1←4; d[4][3]= 4

3) Cài đặt thuật toán

```
int n, a[100][100], d[100][100], e[100][100];
int Floyd(){int i, j, k;
    for (i=1;i<= n; i++)
        for (j=1; j<= n; j++){d[i][j]= a[i][j]; e[i][j]= i;}
    for (k=1; k<= n; k++)
        for (i=1; i<=n; i++)
            for (j=1; j<=n; j++)
                if (d[i][j] > d[i][k] + d[k][j]) {
                    d[i][j] = d[i][k] + d[k][j];  e[i][j] = k; }
    for (i= 1; i<= n; i++) if (d[i][i] < 0) return(0);
     return(1);
 }
```

4) Đánh giá thuật toán

Thuật toán floyd có độ phức tạp $O(n^3)$, với n là số đỉnh của G

Ghi chú

- Floyd có thể dùng để phát hiện chu trình âm:

Sau khi chạy thuật toán, nếu D[u][u] < 0 thì có chu trình âm đi qua đỉnh u.

Tổng kết chương 4

- **Về lý thuyết**:

- Khái niệm đường đi và đường đi ngắn nhất
- Thuật toán Dijkstra
- Thuật toán Bellman-Ford
- Thuật toán Floyd

- **Về các dạng bài tập**

- Viết chương trình mô tả thuật toán.
- Kiểm nghiệm các thuật toán.

Thảo luận


THANK YOU