CHƯƠNG 4. BÀI TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT

4.1 Đường đi ngắn nhất xuất phát từ 1 đỉnh

4.1.1 Thuật toán Dijkstra

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]= max nếu không có cạnh nối i với j; Đỉnh s;

Output: Độ dài d[v] đường đi từ s đến v và pr[v] là đỉnh trước v trên đường đi từ s đến v.

2) Mô tả thuật toán

Khởi tạo: d[v]= a[s][v]; pr[v]= s; vs[v]= 0;

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

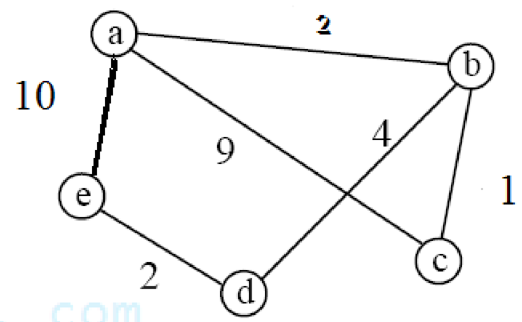
(3) Đặt vs[u]= 1.

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

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

(5) Xuất d[v] và pr[v].

3) Kiểm nghiệm thuật toán

Ví dụ: Cho đồ thị có trọng số G như hình bên. Tìm đường đi ngắn nhất xuất phát từ đỉnh a.	

Giải: Lần lượt có:

i	a	B	c	d	e
1	0; 0	2; a	9; a	m; a	10; a
2		2; a	3; b	6; b	10; a
3			3; b	6; b	10; a
4				6; b	8; d
5					8; d

Kết quả: Độ dài đường đi ngắn nhất từ a đến b là 2: a → b.
Độ dài đường đi ngắn nhất từ a đến c là 3: a → b → c.
Độ dài đường đi ngắn nhất từ a đến d là 6: a → b → d.
Độ dài đường đi ngắn nhất từ a đến e là 8: a → b → d → e.

Ghi chú: Trong thực tế thường sử dụng giải thuật trên vào bài toán sau:

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 d[t] đường đi từ s đến t và đường đi từ s đến t.

Giải thuật:

Khởi tạo: d[i]= a[s, i]; pr[i]= s; vs[i]= 0;

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

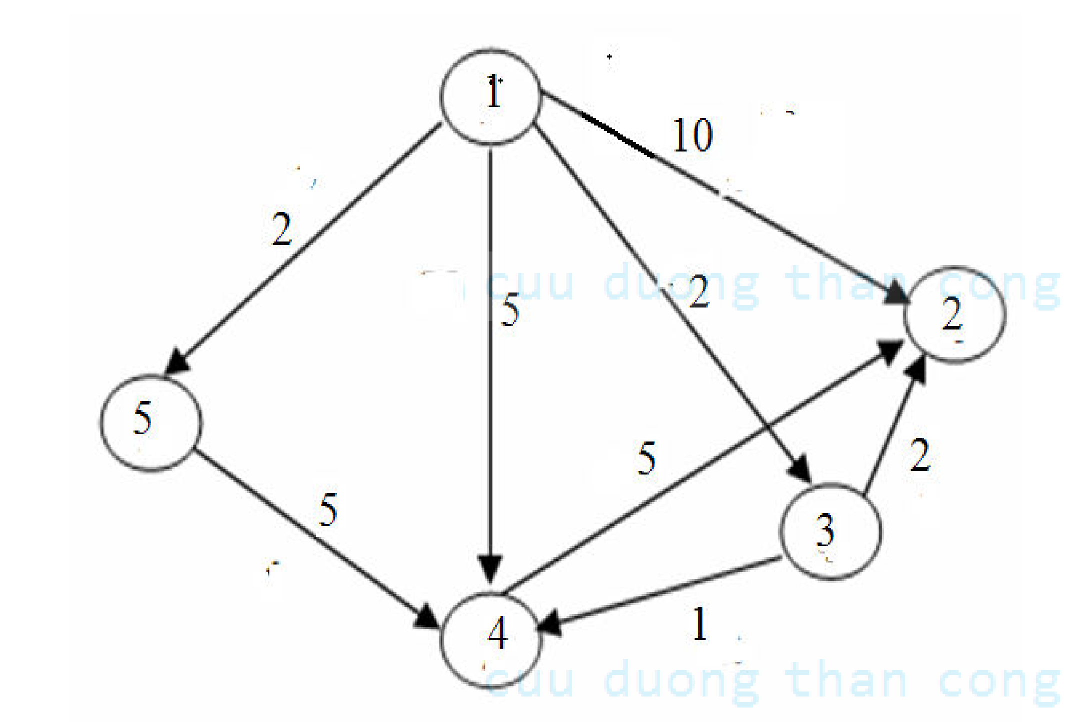
(3) Đặt vs[u]= 1. Nếu u= t thì chuyển sang (5); ngược lại chuyển sang (4);

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

pr[i]= u; d[i]= d[u] + a[u, i]; 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.

Ví dụ. Cho đồ thị G = (V,E), với V = {1, 2, 3, 4, 5} biểu diễn bởi hình vẽ sau, tìm đường đi ngắn nhất từ đỉnh 1 đến đỉnh 2



Kết quả: đường đi ngắn nhất là $1 \rightarrow 3 \rightarrow 2$ và độ dài là 4.

4) Độ phức tạp tính toán

Giải thuật Dijkstra có độ phức tạp $O(n^2)$.

4.1.2 Thuật toán Bellman-Ford

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 từ s đến v và pr[v] là đỉnh trước v trên đường đi từ s đến v.

2) Mô tả thuật toán

Khởi tạo: d[v]= a[s][v]; pr[v]= s; vs[v]= 0;

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

(2) Thực hiện n-2 lần lặp:

(2.1) Với mọi đỉnh $v \in V \setminus \{s\}$ thực hiện

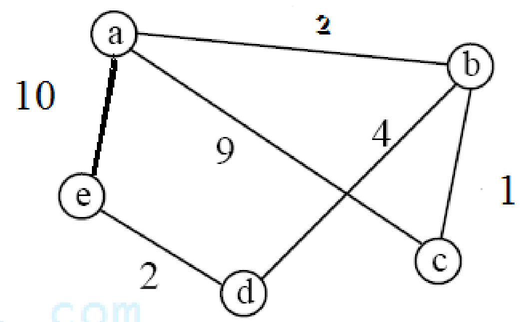
(2.2) Với mọi đỉnh $u \in V$ thực hiện

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

pr[v]= u; d[v]= d[u] + a[u][i];

(3) Xuất d[v] và pr[v].

3) Kiểm nghiệm thuật toán

Ví dụ: Cho đồ thị có trọng số G như hình bên. Tìm đường đi ngắn nhất xuất phát từ đỉnh a.	

Giải: Lần lượt có:

k	a	B	c	d	e
0	0; 0	2; a	9; a	∞; a	10; a
1		2; a	3; b	6; b	8; a
2		2; a	3; b	6; b	8; a
3		2; a	3; b	6; b	8; a
4		2; a	3; b	6; b	8; a

Kết quả: Độ dài đường đi ngắn nhất từ a đến b là 2: a → b.
Độ dài đường đi ngắn nhất từ a đến c là 3: a → b → c.
Độ dài đường đi ngắn nhất từ a đến d là 6: a → b → d.
Độ dài đường đi ngắn nhất từ a đến e là 8: a → b → d → e.

4) Độ phức tạp tính toán

Thuật toán Bellman-ford có độ phức tạp $O(n^3)$.

4.2 Đường đi ngắn nhất giữa các cặp đỉnh

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]= max nếu không có cạnh nối i với j;

Output: Độ dài d[i][j] đường đi từ i đến j và pr[i][j] là đỉnh trước j trên đường đi từ i đến j.

2) Giải thuật Floyd

- Khởi tạo: d[i][j]= a[i][j]; pr[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:

  pr[i][j]= k; d[i][j]= d[i][k] + d[k][j];
- Xuất d[i][j] và pr[i][j].

3) Độ phức tạp tính toán

Thuật toán floyd có độ phức tạp $O(n^3)$.