Học viện Công nghệ Bưu chính Viễn thông
Khoa Công nghệ thông tin 1

Toán rời rạc 2

Bài toán tìm đường đi ngắn nhất

Vũ Hoài Thư

Nội dung

- Phát biểu bài toán tìm đường đi ngắn nhất
- Thuật toán Dijkstra
- Thuật toán Bellman-Ford
- Thuật toán Floyd

Bài toán tìm đường đi ngắn nhất (1/2)

- Khái niệm độ dài đường đi trên đồ thị
  - Xét đồ thị $G =< V, E >$ với tập đỉnh V và tập cạnh E
  - Với mỗi cạnh $(u, v) \in E$, ta đặt tương ứng một số thực $a(u, v)$ được gọi là trọng số của cạnh, $a(u, v) = \infty$ nếu $(u, v) \notin E$
  - Nếu dãy $v_0, v_1, \ldots, v_k$ là một đường đi trên G thì $\sigma_{i=1}^{k} a(v_{i-1}, v_i)$ được gọi là độ dài của đường đi
- Bài toán dạng tổng quát
  - Tìm đường đi (có độ dài) ngắn nhất từ một đỉnh xuất phát $s \in V$ (đỉnh nguồn) đến đỉnh cuối $t \in V$ (đỉnh đích)?
  - Đường đi như vậy được gọi là đường đi ngắn nhất từ s đến t, độ dài của đường đi $d(s, t)$ được gọi là khoảng cách ngắn nhất từ s đến t
  - Nếu không tồn tại đường đi từ s đến t thì độ dài đường đi $d(s, t) = \infty$

Bài toán tìm đường đi ngắn nhất (2/2)

- **Trường hợp 1**: s cố định, t thay đổi
  - Tìm đường đi ngắn nhất từ s đến tất cả các đỉnh còn lại trên đồ thị ?
  - Đối với đồ thị có trọng số không âm, bài toán luôn có lời giải bằng thuật toán Dijkstra
  - Đối với đồ thị có trọng số âm nhưng không tồn tại chu trình âm, bài toán có lời giải bằng thuật toán Bellman-Ford
  - Trong trường hợp đồ thị có chu trình âm, bài toán không có lời giải
- **Trường hợp 2**: s thay đổi và t cũng thay đổi
  - Tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh của đồ thị
  - Đối với đồ thị có trọng số không âm, bài toán được giải quyết bằng cách thực hiện lặp lại n lần thuật toán Dijkstra
  - Đối với đồ thị không có chu trình âm, bài toán có thể giải quyết bằng thuật toán Floyd

Điều kiện để bài toán có lời giải

- Phải tồn tại đường đi từ s tới t
  - Đồ thị vô hướng liên thông hoặc có hướng liên thông mạnh
  - Đồ thị vô hướng, trong đó s và t phải thuộc cùng một thành phần liên thông
  - Đồ thị có hướng, và có đường đi từ s tới t
- Đồ thị không chứa chu trình âm

Nội dung

- Phát biểu bài toán tìm đường đi ngắn nhất
- Thuật toán Dijkstra
- Thuật toán Bellman-Ford
- Thuật toán Floyd

Thuật toán Dijkstra (1/2)

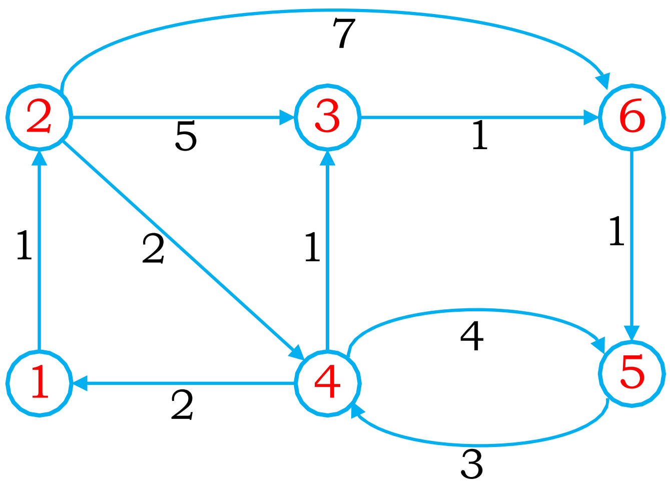
- Mục đích
  - Sử dụng để tìm đường đi ngắn nhất từ một đỉnh s tới các đỉnh còn lại của đồ thị
  - Áp dụng cho đồ thị có hướng với trọng số không âm
- Tư tưởng
  - Gán nhãn tạm thời cho các đỉnh
    - Nhãn của mỗi đỉnh cho biết cận trên của độ dài đường đi ngắn nhất tới đỉnh đó
  - Các nhãn này sẽ được biến đổi (tính lại) nhờ một thủ tục lặp
    - Ở mỗi một bước lặp sẽ có một nhãn tạm thời trở thành nhãn cố định (nhãn đó chính là độ dài đường đi ngắn nhất từ s đến đỉnh đó)

Thuật toán Dijkstra (2/2)

```
Dijkstra (s){
        Bước 1 (Khởi tạo):
        d[s] = 0;   //Gán nhãn của đỉnh s là 0
        T = V\{s}; // T là tập đỉnh có nhãn tạm thời
        for (v ∈ V){ //Sử dụng s gán nhãn cho các đỉnh còn lại
                d[v] = a(s, v);
                truoc[v] = s;
        }
        Bước 2 (Lặp):
        while (T ≠ ∅ ){
                Tìm đỉnh u ∈ T sao cho d[u] = min{ d[z] | z ∈ T};
                T = T\{u}; //cố định nhãn đỉnh u
                for (v ∈ T){ //Sử dụng u, gán nhãn lại cho các đỉnh
                        if (d[v] > d[u] + a(u, v)){
                                d[v] = d[u] + a(u, v); //Gán lại nhãn cho đỉnh v;
                                truoc[v] = u;
                        }
                }
        }
}
```

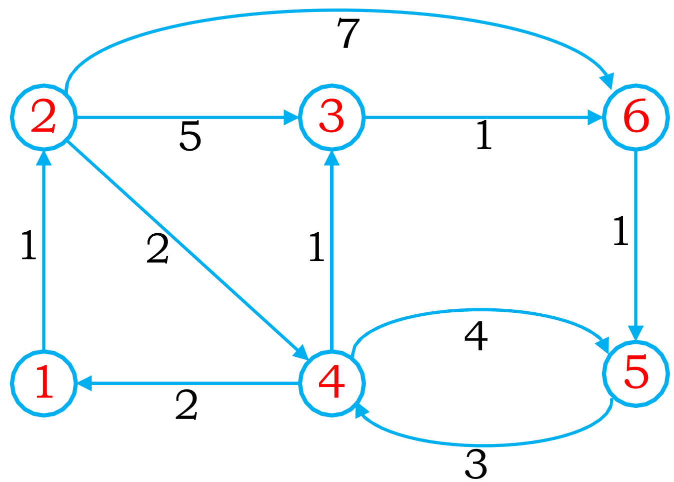
Ví dụ - Dijkstra (1/2)

Áp dụng thuật toán Dijkstra tìm đường đi ngắn nhất từ đỉnh số 1 tới các đỉnh còn lại của đồ thị.



Ví dụ - Dijkstra (2/2)

Áp dụng thuật toán Dijkstra tìm đường đi ngắn nhất từ đỉnh số 1 tới các đỉnh còn lại của đồ thị.



$d[u]$

$truoc[u]$

Bước lặp	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	-	-	-	-	7, 4	5, 3 *
4	-	-	-	-	6, 6 *	-
5						

Nhận xét

- Thuật toán Dijkstra chỉ áp dụng cho đồ thị có hướng và trọng số không âm.
- Ưu điểm:
  - Độ phức tạp về thời gian là tuyến tính (O(|V| +|E| * log (|V|)) nên có thể áp dụng cho các bài toán lớn.
  - Hữu ích cho các bài toán tìm đường đi ngắn nhất, được sử dụng trong bản đồ của google.
- Nhược điểm:
  - Không thể xử lý các trường hợp đồ thị có trọng số âm

Bài tập 1

Cho đơn đồ thị gồm 7 đỉnh được biểu diễn dưới dạng ma trận trọng số sau:

0	25	∞	27	∞	30	∞
25	0	∞	∞	1	∞	15
∞	∞	0	15	3	1	∞
27	∞	15	0	25	∞	∞
∞	1	3	25	0	∞	∞
∞	∞	1	∞	∞	0	1
∞	15	∞	∞	∞	1	0

Áp dụng thuật toán Dijkstra tìm đường đi ngắn nhất từ đỉnh 2 tới đỉnh 6 của đồ thị đã cho, chỉ rõ kết quả trung gian của mỗi bước thực hiện.

Bài tập 2

Áp dụng thuật toán Dijkstra tìm đường đi ngắn nhất từ đỉnh số 2 tới các đỉnh còn lại của đồ thị.

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	∞	2	8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	2	∞	∞	∞	9	∞	∞	∞	∞	∞	∞
3	∞	∞	∞	6	∞	8	1	∞	∞	∞	∞	∞	∞
4	7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
5	∞	∞	1	7	∞	∞	∞	∞	∞	∞	∞	∞	∞
6	∞	∞	∞	∞	1	∞	∞	9	8	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	2	∞	2	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	9	∞	∞	2	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	6	∞	9	8
10	∞	∞	∞	∞	7	6	∞	∞	∞	∞	∞	∞	∞
11	∞	∞	∞	∞	∞	∞	∞	∞	6	7	∞	∞	∞
12	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	2
13	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	7	∞	∞

Nội dung

- Phát biểu bài toán tìm đường đi ngắn nhất
- Thuật toán Dijkstra
- Thuật toán Bellman-Ford
- Thuật toán Floyd

Thuật toán Bellman-Ford (1/2)

- Mục đích
  - Sử dụng để tìm đường đi ngắn nhất từ một đỉnh s tới các đỉnh còn lại của đồ thị
  - Áp dụng cho đồ thị có hướng và không có chu trình âm (có thể có cạnh âm)

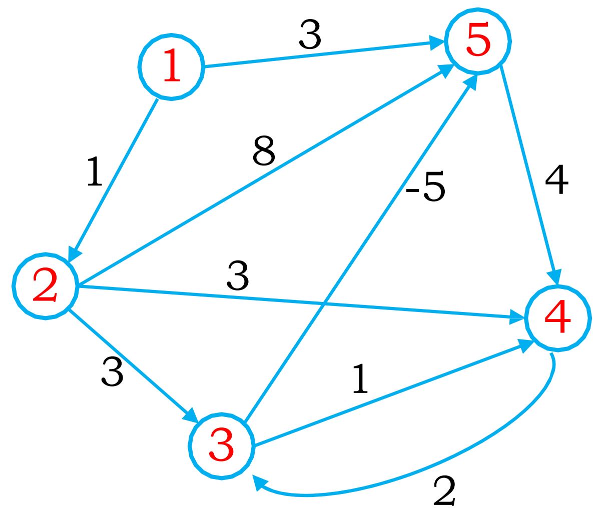
- Tư tưởng
  - Gán nhãn tạm thời cho các đỉnh
    - Nhãn của mỗi đỉnh cho biết cận trên của độ dài đường đi ngắn nhất tới đỉnh đó
  - Các nhãn này sẽ được làm tốt dần (tính lại) nhờ một thủ tục lặp
    - Mỗi khi phát hiện $d[v] > d[u] + a(u, v)$, cập nhật $d[v] = d[u] + a(u, v)$

Thuật toán Bellman-Ford (2/2)

```
Bellman-Ford(s){
        Bước 1 (Khởi tạo):
        for (v ∈ V){ //Sử dụng s gán nhãn cho các đỉnh còn lại
                d[v] = a(s, v);
                truoc[v] = s;
        }
        Bước 2 (Lặp):
        d[s] = 0; k=1
        while (k ≤ n − 2){
                for (v ∈ V\{s}){
                        for (u ∈ V){
                                if (d[v] > d[u] + a(u, v)){
                                        d[v] = d[u] + a(u, v);
                                        truoc[v] = u;
                                }
                        }
                }
        }
}
```

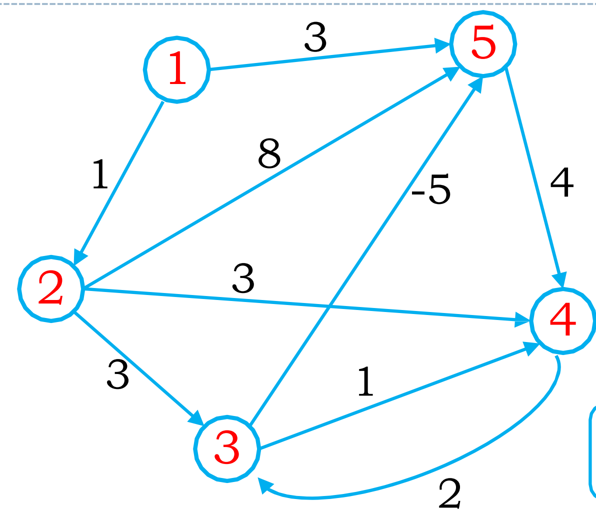
Ví dụ: Bellman-Ford (1/2)

Áp dụng thuật toán Bellman- Ford tìm đường đi ngắn nhất từ đỉnh số 1 tới các đỉnh còn lại của đồ thị.



Ví dụ: Bellman-Ford (2/2)

Áp dụng thuật toán Bellman-Ford tìm đường đi ngắn nhất từ đỉnh số 1 tới các đỉnh còn lại của đồ thị.



$d[u]$

$truoc[u]$

Bước lặp	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5
Khởi tạo	0, 1	1, 1	∞, 1	∞, 1	3, 1
k=1	0, 1	1, 1	4, 2	4, 2	-1, 3
2	0, 1	1, 1	4, 2	3, 5	-1, 3
3	0, 1	1, 1	4, 2	3, 5	-1, 3

Không thay đổi giá trị

Bài tập 3

Áp dụng thuật toán **Bellman-Ford** tìm đường đi ngắn nhất từ đỉnh số 1 tới các đỉnh còn lại của đồ thị.

1	∞	7	∞	9	4	∞	∞	∞	∞
2	∞	∞	3	∞	-4	∞	∞	∞	∞
3	∞	∞	∞	∞	-8	∞	-3	∞	∞
4	∞	∞	∞	∞	∞	∞	∞	-4	∞
5	∞	∞	∞	5	∞	2	∞	3	∞
6	∞	∞	∞	∞	∞	∞	5	∞	2
7	∞	∞	∞	∞	∞	∞	∞	∞	-7
8	∞	∞	∞	∞	∞	-2	∞	∞	-3
9	∞	∞	∞	∞	∞	∞	∞	∞	∞
	1	2	3	4	5	6	7	8	9

Nhận xét

- Thuật toán Bellman-Ford áp dụng được cho trường hợp đồ thị chứa trọng số âm.
- Nhược điểm:
  - Chi phí tính toán lớn (độ phức tạp của thuật toán là $O(n^3)$)
  - Các vòng lặp thường hội tụ về kết quả sớm hơn so với điều kiện dừng của vòng lặp

Nội dung

- Phát biểu bài toán tìm đường đi ngắn nhất
- Thuật toán Dijkstra
- Thuật toán Bellman-Ford
- Thuật toán Floyd

Thuật toán Floyd (1/3)

- Mục đích
  - Sử dụng để tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh của đồ thị
  - Áp dụng cho đồ thị có hướng và không có chu trình âm (có thể có cạnh âm)

- Tư tưởng
  - Thực hiện quá trình lặp
    - Xét từng đỉnh, với tất cả các đường đi (giữa 2 đỉnh bất kỳ), nếu đường đi hiện tại lớn hơn đường đi qua đỉnh đang xét, ta thay lại thành đường đi qua đỉnh này

Thuật toán Floyd (2/3)

```
Floyd( ){
    Bước 1 (Khởi tạo):
    for (i = 1, i ≤ n; i + +){
        for( j = 1, j ≤ n; j + +){ //Xét từng cặp đỉnh
            d[i, j] = a(i, j);
            if(a(i, j)! = ∞) next[i, j] = i;
            else next[i, j] = null;
        }
    }
    Bước 2 (Lặp):
    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];
                    next[i, j] = next[k, j];
                }
            }
        }
    }
}
```

Thuật toán Floyd (3/3)

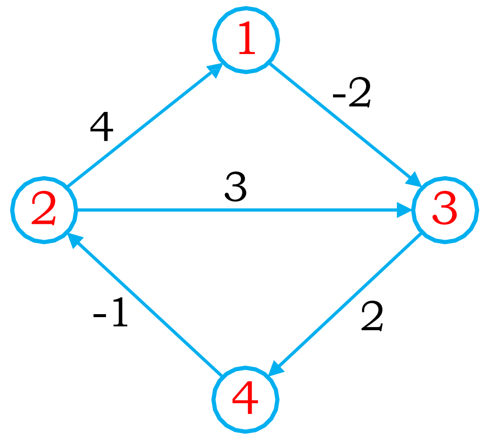
Khôi phục đường đi

```
Reconstruct-Path( u, v){
        if ( next[u][v] == null)
                <Không có đường đi từ u đến v>;
        else{
                path  = [u]; // path bắt đầu từ u
                while(u ≠ v){
                        u = next[u][v];
                        path.append(u);  //đỉnh tiếp theo trên đường đi
                }

                return path;
        }
}
```

Kiểm nghiệm thuật toán

Áp dụng thuật toán Floyd tìm đường đi ngắn nhất giữa tất cả các cặp đỉnh của đồ thị.



Bài tập 4

Cho đơn đồ thị G =< V, E> được biểu diễn dưới dạng ma trận trọng số.

1	0	15	5	20	∞	∞
2	1	0	∞	17	10	∞
3	∞	∞	0	2	∞	50
4	15	1	∞	0	∞	70
5	20	30	∞	10	0	10
6	∞	18	∞	23	20	0
	1	**2**	**3**	**4**	**5**	**6**

Áp dụng thuật toán Floyd, tìm đường đi ngắn nhất giữa các cặp đỉnh (1,2), (1,3), (3,4), (4,2) của đồ thị. Chỉ rõ kết quả tại mỗi bước thực hiện

Nhận xét

- Thuật toán Floy áp dụng được cho trường hợp đồ thị chứa trọng số âm, có thể sử dụng thuật toán để phát hiện chu trình âm trong đồ thị
- Nhược điểm:
  - Chi phí tính toán lớn (độ phức tạp của thuật toán là $O(n^3)$)
  - Áp dụng cho các đồ thị có số đỉnh nhỏ

Tóm tắt

- Bài toán tìm đường đi ngắn nhất trên đồ thị, các dạng của bài toán
- Thuật toán Dijkstra, áp dụng
- Thuật toán Bellman-Ford, áp dụng
- Thuật toán Floyd, áp dụng

Bài tập

- Làm một số bài tập trong giáo trình