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

Bộ môn KHMT, Khoa CNTT1

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

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

$T = V\backslash\{s\};$ // T là tập đỉnh có nhãn tạm thời

for $(v \in V)\{$ //Sử dụng s gán nhãn cho các đỉnh còn lại

$d[v] = a(s, v);$

$truoc[v] = s;$

$\}$

Bước 2 (Lặp):

while $(T \neq \emptyset)\{$

Tìm đỉnh $u \in T$ sao cho $d[u] = \min\{ d[z] \mid z \in T\};$

$T = T\backslash\{u\};$ //cố định nhãn đỉnh u

for $(v \in T)\{$ //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;

$truoc[v] = u;$

$\}$

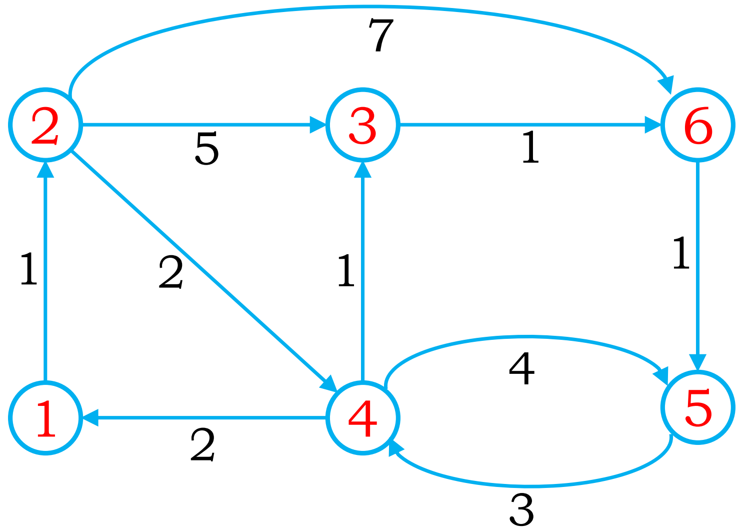
$\}$

$\}$

$\}$

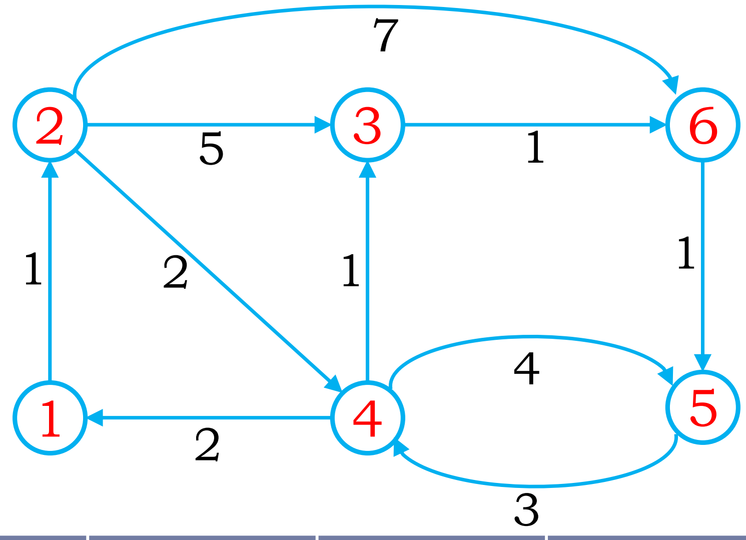
Ví dụ - Dijkstra (1/2)

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						

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 $(v \in V)\{$ //Sử dụng s gán nhãn cho các đỉnh còn lại

$d[v] = a(s, v);$

$truoc[v] = s;$

$\}$

Bước 2 (Lặp):

$d[s] = 0;$

for $(k = 1; k \leq n - 1; k++)\{$

for $(v \in V \backslash \{s\})\{$

for $(u \in V)\{$

if $(d[v] > d[u] + a(u, v))\{$

$d[v] = d[u] + a(u, v);$

$truoc[v] = u;$

$\}$

$\}$

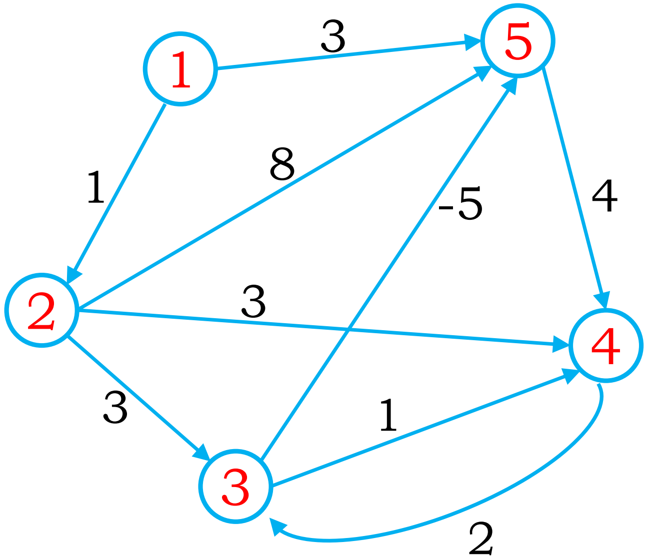
$\}$

$\}$

$\}$

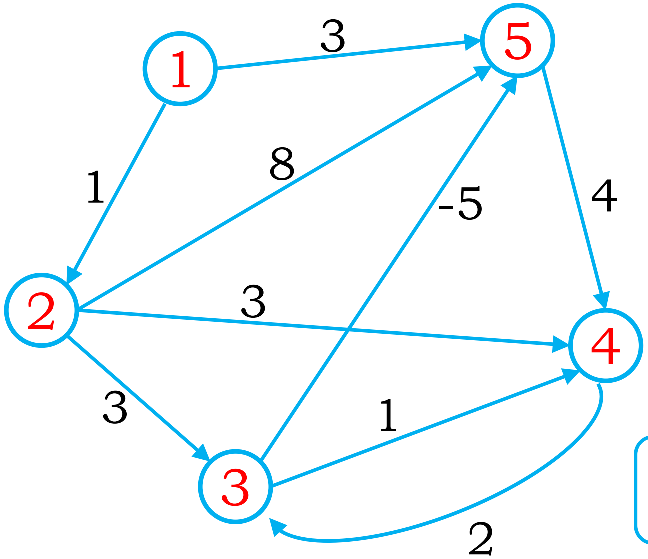
Ví dụ: Bellman-Ford (1/2)

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

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] = j;
            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[i, k];
                }
            }
        }
    }
}
```

Thuật toán Floyd (3/3)

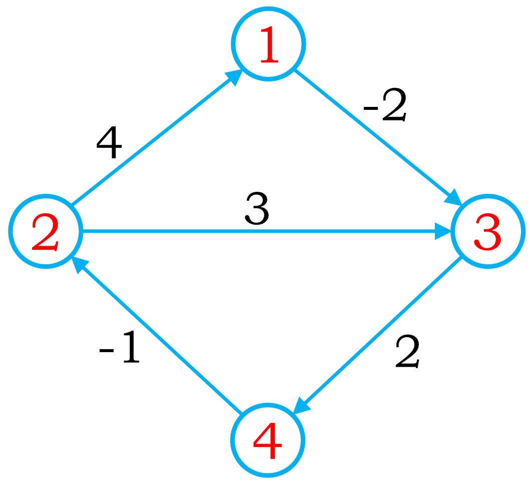
Khôi phục đường đi

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($u \neq v$){
            $u = next[u][v]$;
            $path.append(u)$; //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.



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