BÀI TOÁN TÌM ĐƯỜNG ĐI NGẮN NHẤT

Toán rời rạc 2

Nội dung

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

Phát biểu bài toán tìm đường đi ngắn nhất

Phát biểu bài toán

- Xét đồ thị G=<V, E>:
  - Với mỗi cạnh (u, v)$\in$E, ta đặt tương ứng với nó một số thực A[u][v] được gọi là trọng số của cạnh.
  - Ta sẽ đặt A[u,v]=∞ nếu (u, v)$\notin$E. Nếu dãy v0, v1, . . . , vk là một đường đi trên G thì độ dài của đường đi của nó là.

$$\sum_{i=1}^{p} A[v_{i-1}, v_i]$$

- Bài toán dạng tổng quát:
  - Tìm đường đi 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 (trong trường hợp tổng quát d(s,t) có thể âm).
  - Nếu như không tồn tại đường đi từ s đến t thì độ dài đường đi d(s,t)=∞.

Một số thể hiện cụ thể của bài toán

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

Thuật toán Dijkstra

Mô tả thuật toán

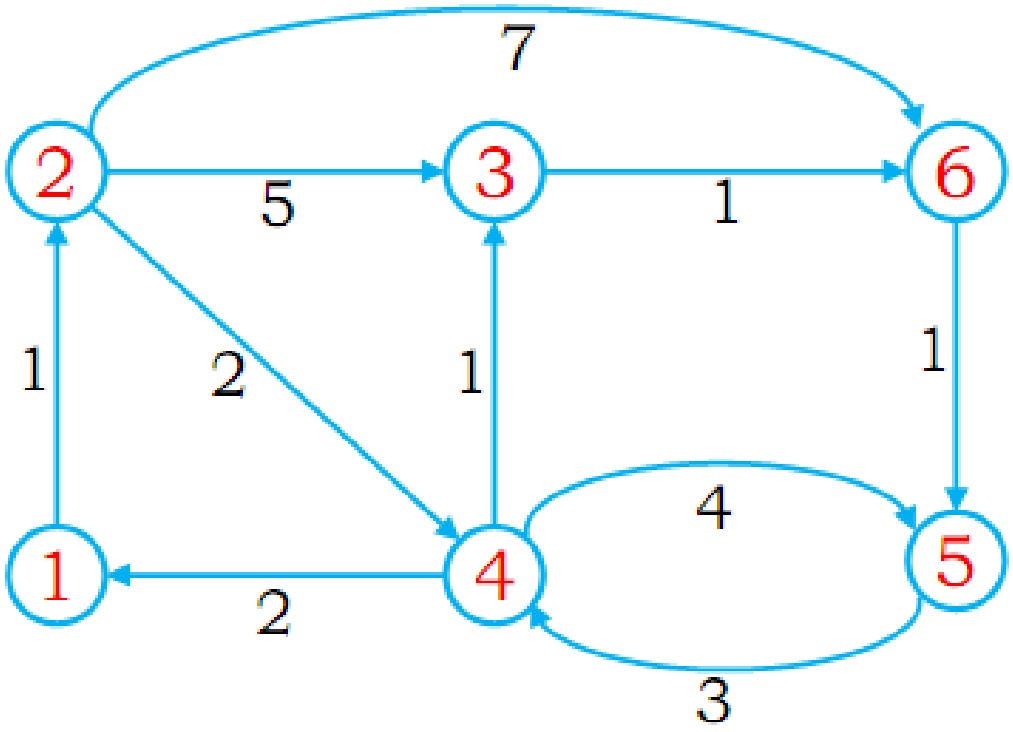
- Mục đích:
  - Sử dụng để tìm đường đi ngắn nhất từ một đỉnh s tới các đỉnh còn lại của đồ thị
  - Áp dụng cho đồ thị có hướng với trọng số không âm.
- Tư tưởng:
  - Gán nhãn tạm thời cho các đỉnh
  - Nhãn của mỗi đỉnh cho biết cận trên của độ dài đường đi ngắn nhất tới đỉnh đó
  - Các nhãn này sẽ được 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

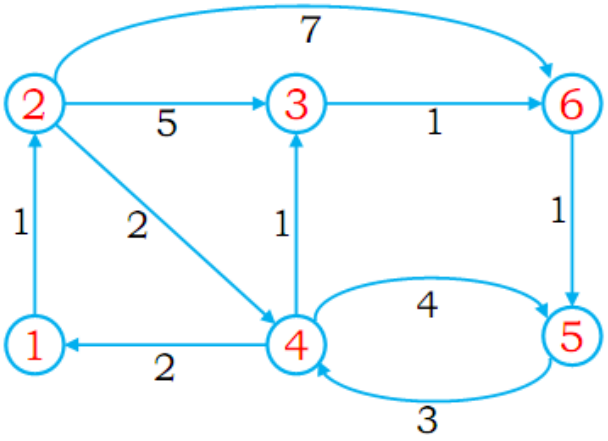
Thuật toán Dijkstra (s): //s∈ V là một đỉnh bất kỳ của G = <V,E>
Begin
Bước 1 (Khởi tạo):
d[s]=0; *//Gán nhãn của đỉnh s là 0*
T = V\{s}; *// T là tập đỉnh có nhãn tạm thời*
for each v∈ V do { *//Sử dụng s gán nhãn cho các đỉnh còn lại*
d[v] = A[s,v];
truoc[v]=s;
endfor;
Bước 2 (Lặp):
while (T≠ ∅) do {
Tìm đỉnh u∈T sao cho d[u] = min { d[z] | z∈T};
T= T\{u}*; //cố định nhãn đỉnh u*
for each v∈ T do { //Sử dụng u, gán nhãn lại cho các đỉnh
if (d[v] > d[u] + A[u, v]) then {
d[v] = d[u] + A[u, v]; //Gán lại nhãn cho đỉnh v;
truoc[v] = u;
endif;
endfor;
endwhlie;
Bước 3 (Trả lại kết quả):
Return (d[s], truoc[s]);
End.

Ví dụ 1- 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 1 - Dijkstra (2/2)



$d[u]$

$truoc[u]$

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						

Ví dụ 2 Dijkstra (1/3)

- Áp dụng thuật toán Dijkstra tìm đường đi ngắn nhất từ đỉnh số 1 tới các đỉnh còn lại của đồ thị được biểu diễn dưới dạng ma trận trọng số như hình bên.

∞	2	8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	2	∞	∞	∞	9	∞	∞	∞	∞	∞	∞
∞	∞	∞	6	∞	8	1	∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	1	7	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	1	∞	∞	9	8	∞	∞	∞	∞
∞	∞	∞	∞	∞	2	∞	2	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	9	∞	∞	2	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	6	∞	9	8
∞	∞	∞	∞	7	6	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	6	7	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	2
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	7	∞	∞

Ví dụ 2 Dijkstra (2/3)

Các bước thực hiện thuật toán Dijkstra tại s =1

Bước	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6	Đỉnh 7	Đỉnh 8	Đỉnh 9	Đỉnh 10	Đỉnh 11	Đỉnh 12	Đỉnh 13
1	<0,1>	<2,1>	<8,1>	<∞,1>	<∞,1>	<∞,1>	<∞,1>	<∞,1>	<∞,1>	<∞,1>	<∞,1>	<∞,1>	<∞,1>
2	*	<2,1>	<4,2>	<∞,1>	<∞,1>	<∞,1>	<11,2>	<∞,1>	<∞,1>	<∞,1>	<∞,1>	<∞,1>	<∞,1>
3	*	*	<4,2>	<10,3>	<∞,1>	<12,3>	<5,3>	<∞,1>	<∞,1>	<∞,1>	<∞,1>	<∞,1>	<∞,1>
4	*	*	*	<10,3>	<∞,1>	<7, 7>	<5,3>	<7, 7>	<∞,1>	<∞,1>	<∞,1>	<∞,1>	<∞,1>
5	*	*	*	<10,3>	<8,6>	<7, 7>	*	<7,7>	<15,6>	<∞,1>	<∞,1>	<∞,1>	<∞,1>
6	*	*	*	<10,3>	<8,6>	*	*	<7,7>	<15,6>	<∞,1>	<∞,1>	<9,8>	<∞,1>
7	*	*	*	<10,3>	<8,6>	*	*	*	<15,6>	<∞,1>	<∞,1>	<9,8>	<∞,1>
8	*	*	*	<10,3>	*	*	*	*	<15,6>	<∞,1>	<∞,1>	<9,8>	<11,12>
9	*	*	*	<10,3>	*	*	*	*	<15,6>	<∞,1>	<∞,1>	*	<11,12>
10	*	*	*	*	*	*	*	*	<15,6>	<∞,1>	<18,13>	*	<11,12>
11	*	*	*	*	*	*	*	*	<15,6>	<21,9>	<18,13>	*	*
12	*	*	*	*	*	*	*	*	*	<21,9>	<18,13>	*	*
13	*	*	*	*	*	*	*	*	*	<21,9>	*	*	*

Ví dụ 2 Dijkstra (3/3)

- Kết quả:
  - Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 2: 2. Đường đi: 1-2.
  - Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 3: 4. Đường đi: 1-2-3.
  - Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 4: 10. Đường đi: 1-2-3-4. Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 5: 8. Đường đi: 1-2-3-7-6-5.
  - Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 6: 7. Đường đi: 1-2-3-7-6.
  - Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 7: 5. Đường đi: 1-2-3-7.
  - Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 8: 7. Đường đi: 1-2-3-7-8.
  - Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 9: 15. Đường đi: 1-2-3-7-6-9.
  - Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 10: 21. Đường đi: 1-2-3-7-6-9-10.
  - Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 11: 18. Đường đi: 1-2-3-7-8-12-13-11.
  - Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 12: 18. Đường đi: 1-2-3-7-8-12.
  - Đường đi ngắn nhất từ đỉnh 1 đến đỉnh 13: 11. Đường đi: 1-2-3-7-8-12-13.

Cài đặt thuật toán Dijkstra

- Xem code minh họa.

Thuật toán Bellman-Ford

Mô tả thuật toán

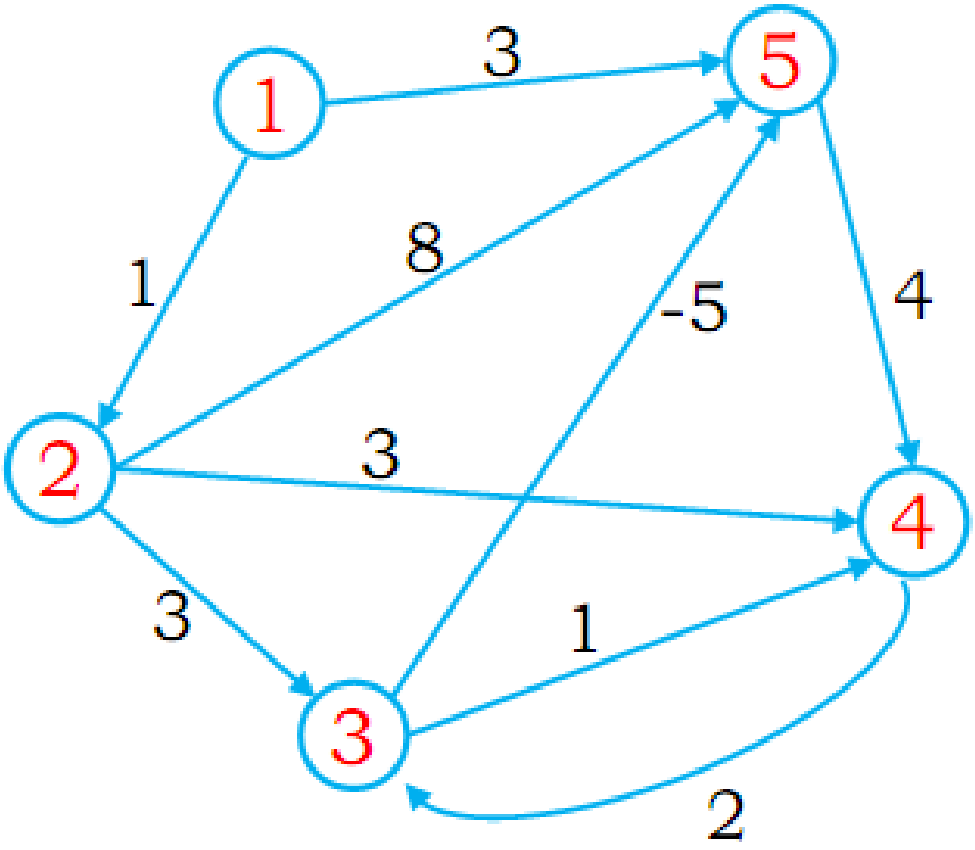
- Mục đích
  - Sử dụng để tìm đường đi ngắn nhất từ một đỉnh s tới các đỉnh còn lại của đồ thị
  - Áp dụng cho đồ thị có hướng và không có chu trình âm (có thể có cạnh âm)
- Tư tưởng
  - Gán nhãn tạm thời cho các đỉnh
  - Nhãn của mỗi đỉnh cho biết cận trên của độ dài đường đi ngắn nhất tới đỉnh đó
  - 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 đ[v]= d[u]+A[u][v].

Thuật toán Bellman-Ford

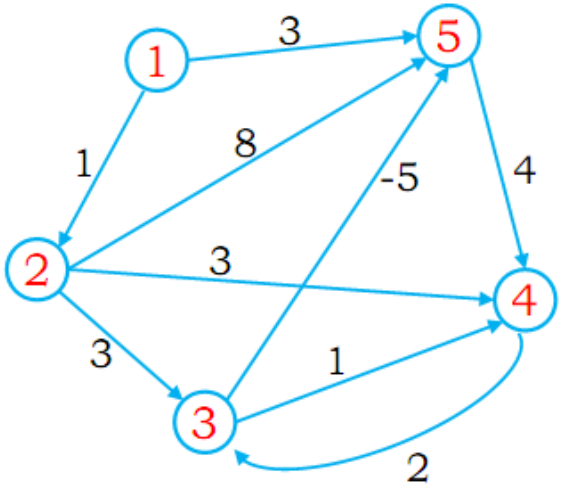
Thuật toán Bellman-Ford (s): *//s ∈ V là đỉnh bất kỳ của đồ thị*
Begin:
Bước 1 (Khởi tạo):
for v∈V do { *//Sử dụng s gán nhãn cho các đỉnh v∈V*
D[v] = A[s][v];
Truoc[v] = s;
}
Bước 2 (Lặp) :
D[s] = 0; K=1;
while (K<=N-2) { //N-2 vòng lặp
for v∈V\{s} do { *//Lấy mỗi đỉnh v∈V\s*
for u∈V do { *//Gán nhãn cho v*
if (D[v] > D[u] + A[u][v]) {
D[v]= D[u] + A[u][v];
Truoc[v] = u;
endif;
endfor;
endfor;
endwlie;
Bước 3 (Trả lại kết quả):
Return(D[v], Truoc[v]: v∈U);
End.

Ví dụ 1: 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 1: Bellman-Ford (2/2)



$d[u]$

$truoc[u]$

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

Ví dụ 2 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 được biểu diễn dưới dạng ma trận trọng số như hình bên.

$$A = \begin{vmatrix} \infty & 1 & \infty & \infty & 3 \\ \infty & \infty & 3 & 3 & 8 \\ \infty & \infty & \infty & 1 & -5 \\ \infty & \infty & 2 & \infty & \infty \\ \infty & \infty & \infty & 4 & \infty \end{vmatrix}$$

Ví dụ 2 Bellman-Ford (2/2)

Kết quả kiểm nghiệm theo thuật toán Bellman-Ford

K=?	D[1], Truoc[1]	D[2], Truoc[2]	D[3], Truoc[3]	D[4], Truoc[4]	D[5], Truoc[5]
	<0,1>	<1,1>	<∞,1>	<∞,1>	<3,1>
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>

Cài đặt thuật toán Bellman-Ford

- Xem code minh họa.

Thuật toán Floyd

Mô tả thuật toán

- 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

Thuật toán Floy:
Begin:

Bước 1 (Khởi tạo):

```
for (i=1; i≤ n; i++) {
    for (j =1; j≤ n; j++) {
        d[i,j] = a[i, j];
        p[i,j] = i;
    }
}
```

Bước 2 (lặp) :

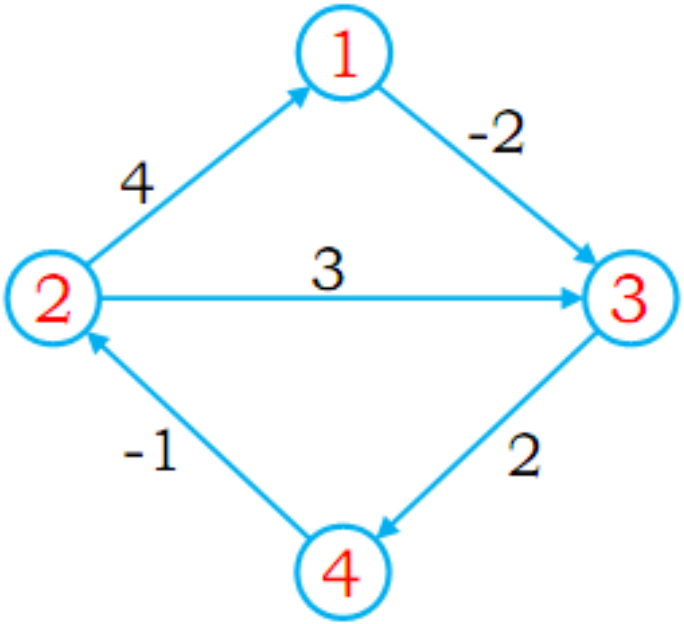
```
for (k=1; k≤ n; k++) {
    for (i=1; i≤ n; i++){
        for (j =1; j≤ n; j++) {
            if (d[i,j] > d[i, k] + d[k, j]) {
                d[i, j] = d[i, k] + d[k, j];
                p[i,j] = p[k, j];
            }
        }
    }
}
}
```

Bước 3 (Trả lại kết quả):

```
Return (p([i,j], d[i,j]: i, j ∈V);
```

Kiểm nghiệm thuật toán

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



Cài đặt thuật toán Floyd

- Xem code minh họa.

Bài tập

- Làm các bài tập 1, 5, 6 trong Tài liệu giảng dạy môn Toán rời rạc 2.