LÝ THUYẾT ĐỒ THỊ

Graph Theory

Nội dung

1. Bài toán đường đi ngắn nhất (ĐĐNN)
2. Giả thiết cơ bản
3. Tính chất của ĐĐNN
4. Đường đi ngắn nhất xuất phát từ 1 đỉnh
5. Đường đi ngắn nhất trong đồ thị không có chu trình
6. Thuật toán Floyd-Warshal

1. Bài toán đường đi ngắn nhất

- Cho đơn đồ thị có hướng $G = (V,E)$ với hàm trọng số c: $E \to R$ ($c(e)$ được gọi là độ dài hay trọng số của cạnh e)
- Độ dài của đường đi $P = v_1 \to v_2 \to \ldots \to v_k$ là số

$$w(P) = \sum_{i=1}^{k-1} c(v_i, v_{i+1})$$

- Đường đi ngắn nhất từ đỉnh u đến đỉnh v là đường đi có độ dài ngắn nhất trong số các đường đi nối u với v.
- Độ dài của đường đi ngắn nhất từ u đến v còn được gọi là khoảng cách từ u tới v và ký hiệu là $\delta(u,v)$.

Các ứng dụng thực tế

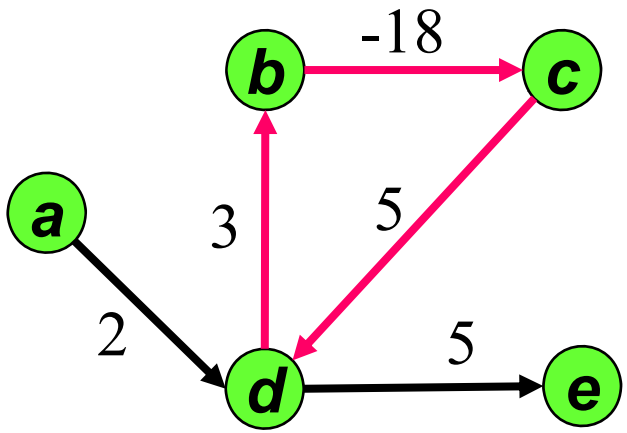
- Giao thông (Transportation)
- Truyền tin trên mạng (Network routing) (cần hướng các gói tin đến đích trên mạng theo đường nào?)
- Truyền thông (Telecommunications)
- Speech interpretation (best interpretation of a spoken sentence)
- Điều khiển robot (Robot path planning)
- Medical imaging
- Giải các bài toán phức tạp hơn trên mạng
- ...

Các dạng bài toán ĐĐNN

1. **Bài toán một nguồn một đích:** Cho hai đỉnh *s* và *t*, cần tìm đường đi ngắn nhất từ *s* đến *t*.
2. **Bài toán một nguồn nhiều đích:** Cho *s* là đỉnh nguồn, cần tìm đường đi ngắn nhất từ s đến tất cả các đỉnh còn lại.
3. **Bài toán mọi cặp:** Tìm đường đi ngắn nhất giữa mọi cặp đỉnh của đồ thị.
   - Đường đi ngắn nhất theo số cạnh - BFS.

2. Giả thiết cơ bản

- Nếu đồ thị có chu trình âm thì độ dài đường đi giữa hai đỉnh nào đó có thể làm nhỏ tuỳ ý:



Xét đường đi từ a đến e:

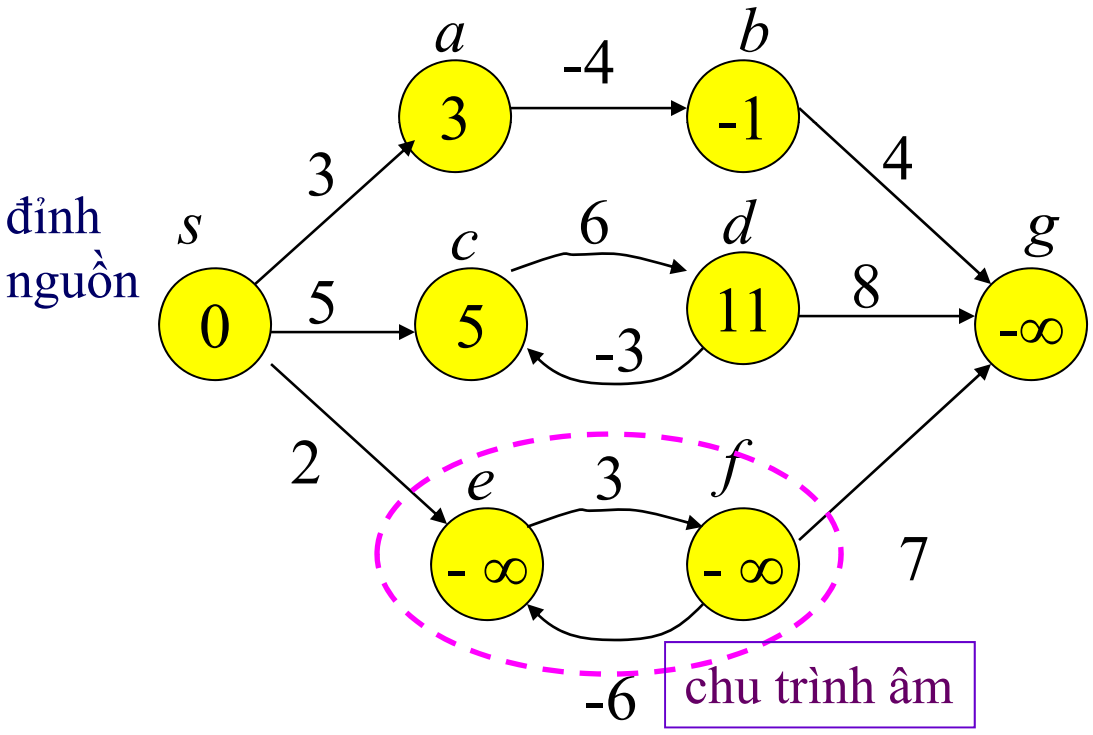
$P: a \rightarrow \sigma(d \rightarrow b \rightarrow c \rightarrow d) \rightarrow e$

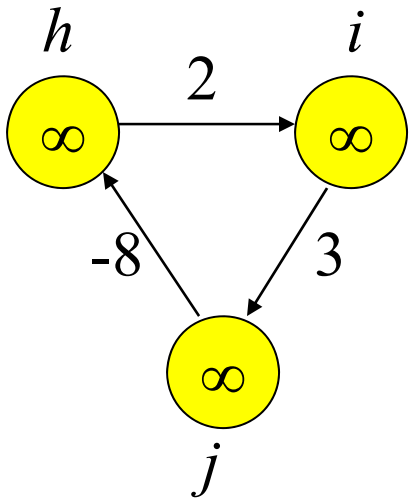
$w(P) = 7\text{-}10\sigma \rightarrow -\infty$, khi $\sigma \rightarrow +\infty$

Giả thiết: Đồ thị không chứa chu trình độ dài âm (gọi tắt là chu trình âm)

Trọng số âm

Độ dài của đường đi ngắn nhất có thể là ∞ hoặc $-\infty$.

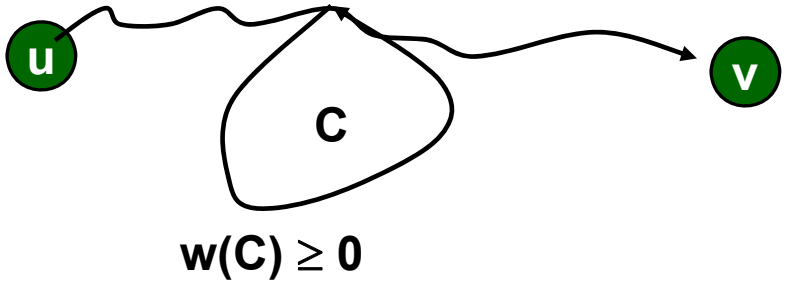




không đạt tới được từ *s*

3. Các tính chất của ĐĐNN

- ***Tính chất* 1.** *Đường đi ngắn nhất luôn có thể tìm trong số các đường đi đơn.*
  - CM: Bởi vì việc loại bỏ chu trình độ dài không âm khỏi đường đi không làm tăng độ dài của nó.



- ***Tính chất* 2.** *Mọi đường đi ngắn nhất trong đồ thị G đều đi qua không quá n-1 cạnh, trong đó n là số đỉnh.*
  - *Như là hệ quả của tính chất 1*

Nội dung

1. Bài toán đường đi ngắn nhất (ĐĐNN)
2. Giả thiết cơ bản
3. Tính chất của ĐĐNN
4. **Đường đi ngắn nhất xuất phát từ 1 đỉnh**
   (Single-Source Shortest Paths)
   - Thuật toán Ford - Bellman
   - Thuật toán Dijkstra
5. Đường đi ngắn nhất trong đồ thị không có chu trình
6. Thuật toán Floyd-Warshal

Biểu diễn đường đi ngắn nhất

Các thuật toán tìm đường đi ngắn nhất làm việc với hai mảng:

- $d(v)$ = độ dài đường đi từ s đến v ngắn nhất hiện biết
  (cận trên cho độ dài đường đi ngắn nhất thực sự).

- $truoc(v)$ = đỉnh đi trước v trong đường đi nói trên
  (sẽ sử dụng để truy ngược đường đi từ s đến v) .

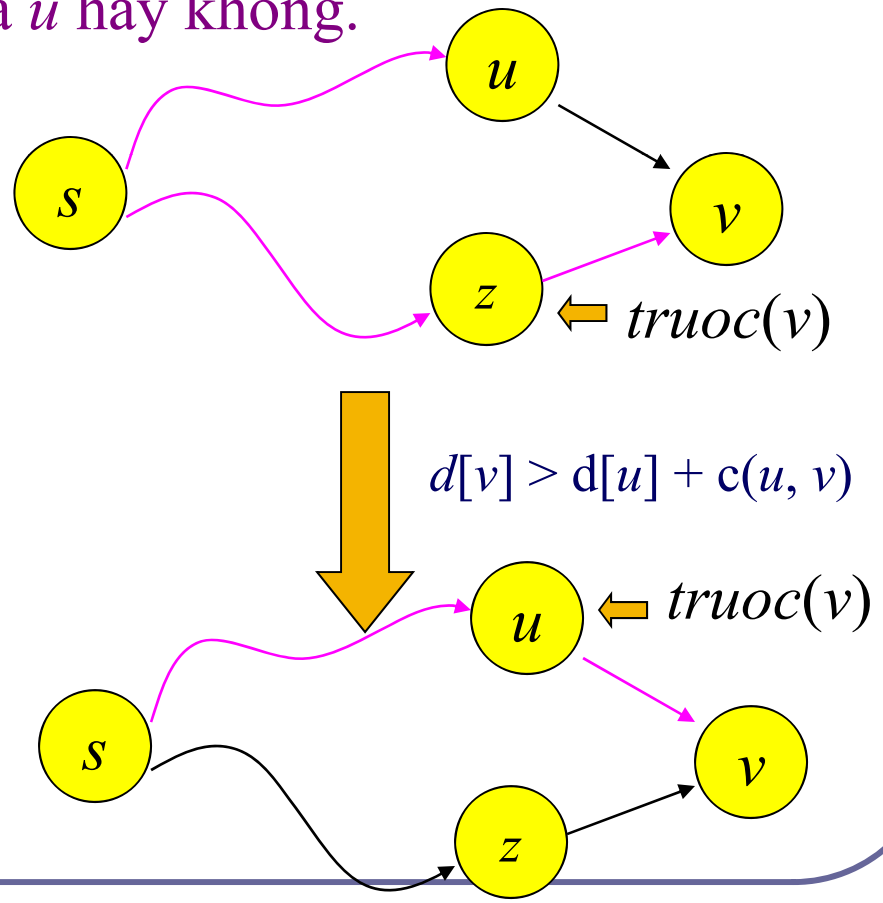
Khởi tạo (Initialization)

```
for v ∈ V(G) do
        d[v] ← ∞
        truoc[v] ← NIL
d[s] ← 0
```

Giảm cận trên (Relaxation)

Sử dụng cạnh (u, v) để kiểm tra xem đường đi đến v đã tìm được có thể làm ngắn hơn nhờ đi qua u hay không.

```
Relax(u, v)
    if d[v] > d[u]  + c(u, v)
        then d[v] ← d[u] + c(u, v)
                truoc[v] ← u
```



Các thuật toán tìm đđnn khác nhau ở số lần dùng các cạnh và trình tự duyệt chúng để thực hiện giảm cận

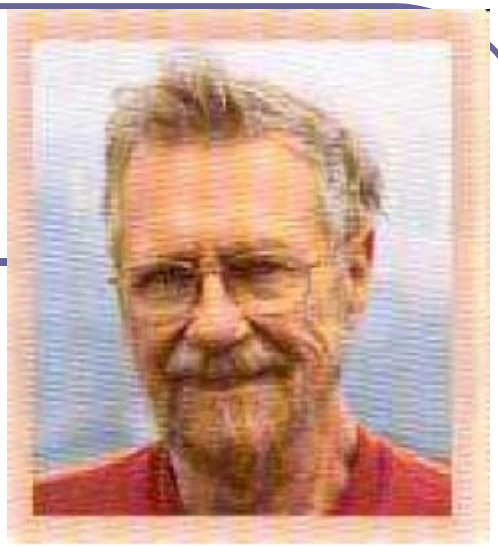
.

4.1 Thuật toán Ford-Bellman

- Thuật toán Ford - Bellman tìm đường đi ngắn nhất từ đỉnh *s* đến tất cả các đỉnh còn lại của đồ thị.
- Thuật toán làm việc trong trường hợp trọng số của các cung là tuỳ ý.
- Giả thiết rằng trong đồ thị không có chu trình âm.
- **Đầu vào:** *Đồ thị* $G=(V,E)$
  - *gồm n đỉnh xác định bởi ma trận trọng số* $c[u,v]$ với $u,v \in V$;
  - *Đỉnh nguồn* $s \in V$
- **Đầu ra:** *Với mỗi* $v \in V$
  - $d[v]$ = *độ dài đường đi ngắn nhất từ s đến v*;
  - *truoc*$[v]$ - *đỉnh đi trước v trong đđnn từ s đến v*.

4.2. Thuật toán Dijkstra

- Trong trường hợp trọng số trên các cung là không âm, thuật toán do Dijkstra đề nghị hữu hiệu hơn rất nhiều so với thuật toán Ford-Bellman.



Edsger W.Dijkstra

(1930-2002)

- **Đầu vào:** *Đồ thị* $G=(V,E)$
  - *ma trận trọng số* $c[u,v] \geq 0$ với $u,v \in V$;
  - *Đỉnh nguồn* $s \in V$
  - Giả thiết rằng trong đồ thị không có chu tr×nh âm.
- **Đầu ra:** *Với mỗi* $v \in V$
  - $d[v]$ = *độ dài đường đi ngắn nhất từ s đến v*;
  - $truoc[v]$ - *đỉnh đi trước v trong đđnn từ s đến v.*

Thuật toán Dijkstra

- **Chú ý:** Nếu chỉ cần tìm đường đi ngắn nhất từ *s* đên *t* thì có thể chấm dứt thuật toán khi đỉnh t trở thành có nhãn cố định.

- **Định lý 1.** *Thuật toán Dijkstra tìm được đường đi ngắn nhất từ đỉnh s đến tất cả các đỉnh còn lại trên đồ thị sau thời gian* $O(n^2)$.

Nội dung

1. Bài toán đường đi ngắn nhất (ĐĐNN)
2. Giả thiết cơ bản
3. Tính chất của ĐĐNN
4. Đường đi ngắn nhất xuất phát từ 1 đỉnh
   - Thuật toán Ford - Bellman
   - Thuật toán Dijkstra
5. Đường đi ngắn nhất trong đồ thị không có chu trình
6. Đường đi ngắn nhất giữa mọi cặp đỉnh

5. Đường đi ngắn nhất trong đồ thị không có chu trình

5.1. Thuật toán đánh số đỉnh

5.2. Thuật toán tìm đđnn trên đồ thị không có chu trình

5.3. Ứng dụng PERT

5. Đường đi ngắn nhất trong đồ thị không có chu trình

- Một trường hợp riêng của bài toán đường đi ngắn nhất giải được nhờ thuật toán với độ phức tạp tính toán O(n^2), đó là bài toán trên đồ thị không có chu trình (còn trọng số trên các cung có thể là các số thực tuỳ ý). Kết quả sau đây là cơ sở để xây dựng thuật toán nói trên:

- **Định lý 2.** *Giả sử G là đồ thị không có chu trình. Khi đó các đỉnh của nó có thể đánh số sao cho mỗi cung của đồ thị chỉ hướng từ đỉnh có chỉ số nhỏ hơn đến đỉnh có chỉ số lớn hơn, nghĩa là mỗi cung của nó có thể biểu diễn dưới dạng* ($v[i]$, $v[j]$), *trong đó* $i < j$.

Thuật toán đánh số đỉnh

- **Đầu vào:** *Đồ thị có hướng* $G=(V,E)$ *với n đỉnh không chứa chu trình được cho bởi danh sách kề* $Ke(v)$, $v \in V$.
- **Đầu ra:** *Với mỗi đỉnh* $v \in V$ *chỉ số* $NR[v]$ *thoả mãn: Với mọi cung* (u, v) *của đồ thị ta đều có* $NR[u] < NR[v]$.

Thuật toán đánh số đỉnh

- Rõ ràng trong bước khởi tạo ta phải duyệt qua tất cả các cung của đồ thị khi tính bán bậc vào của các đỉnh, vì vậy ở đó ta tốn cỡ $O(m)$ phép toán, trong đó m là số cung của đồ thị. Tiếp theo, mỗi lần đánh số một đỉnh, để thực hiện việc loại bỏ đỉnh đã đánh số cùng với các cung đi ra khỏi nó, chúng ta lại duyệt qua tất cả các cung này. Suy ra để đánh số tất cả các đỉnh của đồ thị chúng ta sẽ phải duyệt qua tất cả các cung của đồ thị một lần nữa.
- Vậy độ phức tạp của thuật toán là $O(m)$.

5.2. Thuật toán tìm đđnn trên đồ thị không có chu trình

- Do có thuật toán đánh số trên, nên khi xét đồ thị không có chu trình ta có thể giả thiết là các đỉnh của nó được đánh số sao cho mỗi cung chỉ đi từ đỉnh có chỉ số nhỏ đến đỉnh có chỉ số lớn hơn.
- ***Thuật toán tìm đường đi ngắn nhất từ đỉnh nguồn v[1] đến tất cả các đỉnh còn lại trên đồ thị không có chu trình***
- **Đầu vào:** *Đồ thị* $G=(V, E)$, *trong đó* $V=\{ v[1], v[2], \ldots, v[n] \}$.
  Đối với mỗi cung $(v[i], v[j]) \in E$, *ta có* $i < j$.
  Đồ thị được cho bởi danh sách kề $Ke(v)$, $v \in V$.
- **Đầu ra:** *Khoảng cách từ* $v[1]$ *đến tất cả các đỉnh còn lại được ghi trong mảng* $d[v[i]]$, $i = 2, 3, \ldots, n$

Thuật toán tìm đđnn trên đồ thị không có chu trình

```
procedure   Critical_Path;
begin
    d[v[1]] := 0;
    for   j:=2 to n do d[v[j]] :=∞;
    for   v[j] ∈ Ke[v[1]] do
          d[v[j]] := c(v[1], v[j]) ;
     for   j:= 2  to  n  do
          for  v ∈ Ke[v[j]]  do
                d[v] :=  min ( d[v],  d[v[j]] + c(v[j], v) ) ;
end;
```

- **Độ phức tạp tính toán của thuật toán là O(*m*), do mỗi cung của đồ thị phải xét qua đúng một lần.**

5.3. Ứng dụng: PERT

- Xây dựng phương pháp giải bài toán điều khiển việc thực hiện những dự án lớn, gọi tắt là PERT (*Project Evaluation and Review Technique*) hay CDM (*Critical path Method*).
- Việc thi công một công trình lớn được chia ra làm n công đoạn, đánh số từ 1 đến n. Có một số công đoạn mà việc thực hiện nó chỉ được tiến hành sau khi một số công đoạn nào đó đã hoàn thành. Đối với mỗi công đoạn i biết $t[i]$ là thời gian cần thiết để hoàn thành nó ($i = 1, 2,..., n$).
- Gọi thời điểm bắt đầu tiến hành thi công công trình là 0. Hãy tìm tiến độ thi công công trình (chỉ rõ mỗi công đoạn phải được bắt đầu thực hiện vào thời điểm nào) để cho công trình được hoàn thành xong trong thời điểm sớm nhất có thể được.

Ví dụ:

- Các dữ liệu với $n = 8$ được cho trong bảng sau đây

Công đoạn	t[i]	Các công đoạn phải hoàn thành trước nó
1	15	Không có
2	30	1
3	80	Không có
4	45	2, 3
5	124	4
6	15	2, 3
7	15	5, 6
8	19	5

1. Bài toán đường đi ngắn nhất (ĐĐNN)
2. Giả thiết cơ bản
3. Tính chất của ĐĐNN
4. Đường đi ngắn nhất xuất phát từ 1 đỉnh
   - Thuật toán Ford - Bellman
   - Thuật toán Dijkstra
5. Đường đi ngắn nhất trong đồ thị không có chu trình
6. **Đường đi ngắn nhất giữa mọi cặp đỉnh**
   (All-Pairs Shortest Paths)

Đường đi ngắn nhất giữa mọi cặp đỉnh

Bài toán Cho đồ thị $G = (V, E)$, với trọng số trên cạnh e là $w(e)$, đối với mỗi cặp đỉnh u, v trong V, tìm đường đi ngắn nhất từ u đến v.

- Đầu vào: *ma trận trọng số*.
- Đầu ra *ma trận*: phần tử ở dòng u cột v là độ dài đường đi ngắn nhất từ u đến v.
- Cho phép có trọng số âm
- **Giả thiết: Đồ thị không có chu trình âm.**