Chương 5

BÀI TOÁN ĐƯỜNG ĐI NGẮN NHẤT

Nội dung

5.1. Bài toán đường đi ngắn nhất (ĐĐNN)

5.2. Tính chất của ĐĐNN, Giảm cận trên

5.3. Thuật toán Bellman-Ford

5.4. Thuật toán Dijkstra

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

5.6. Thuật toán Floyd-Warshal

5.1. Bài toán đường đi ngắn nhất

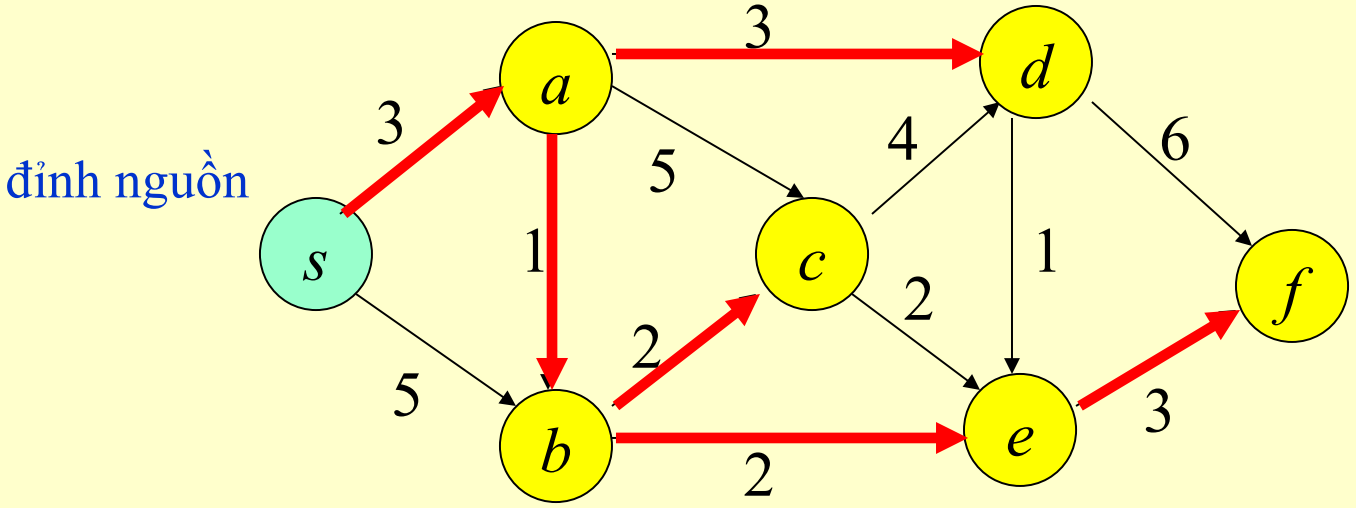
- Cho đơn đồ thị có hướng $G = (V,E)$ với hàm trọng số w: $E \to R$ ($w(e)$ được gọi là độ dài hay trọng số của cạnh e)
- Độ dài của đường đi $P = v_1 \to v_2 \to \ldots \to v_k$ là số

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

- 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 $\delta(u,v)$.

Ví dụ

Cho đồ thị có trọng số $G = (V, E)$, và đỉnh nguồn $s \in V$, hãy tìm đường đi ngắn nhất từ s đến mỗi đỉnh còn lại.



	s	a	b	c	d	e	f
weight	0	3	4	6	6	6	9
path	s	s,a	s,a,b	s,a,b,c	s,a,d	s,a,b,e	s,a,b,e,f

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.

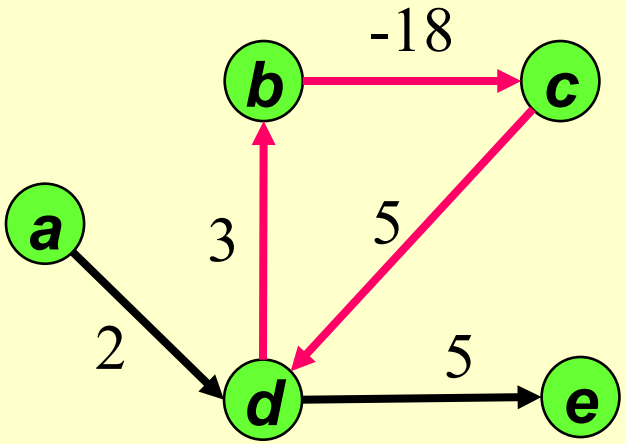
Nhận xét

- Các bài toán được xếp theo thứ tự từ đơn giản đến phức tạp

- Hễ có thuật toán hiệu quả để giải một trong ba bài toán thì thuật toán đó cũng có thể sử dụng để giải hai bài toán còn lại

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 $\sigma \rightarrow +\infty$

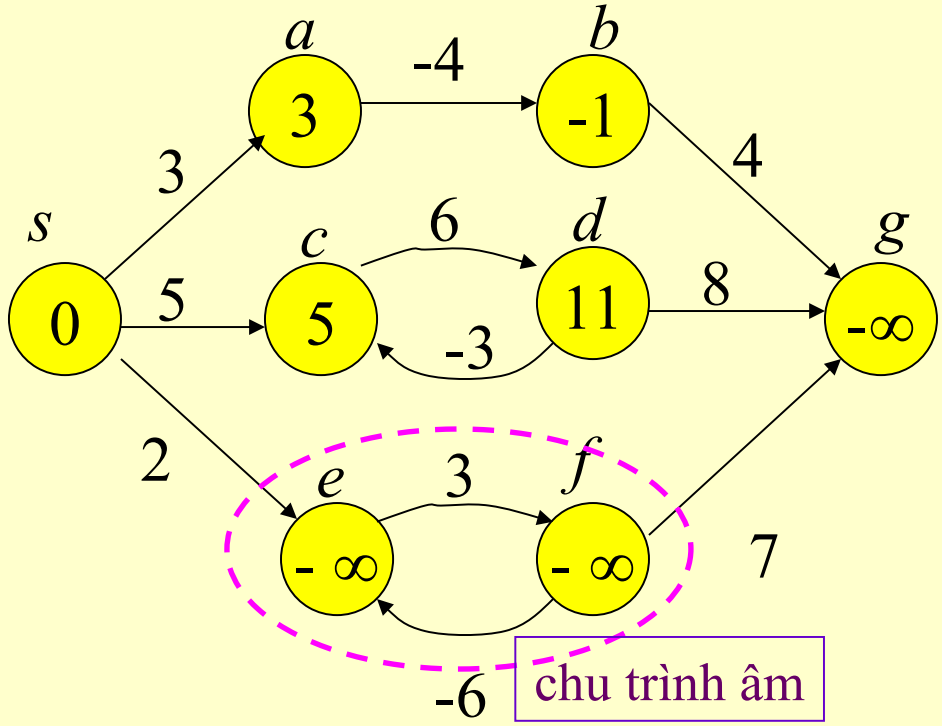
Giả thiết:

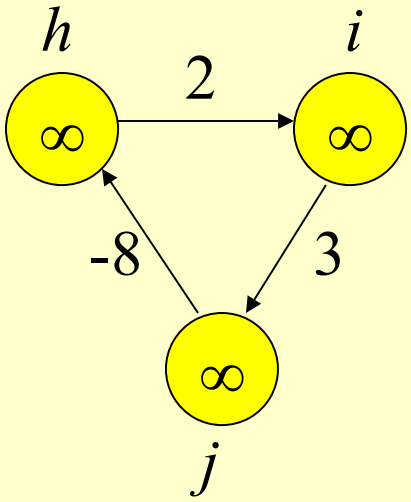
Đồ thị không chứa chu trình độ dài âm (gọi tắt là chu trình âm)

Trọng số âm

Độ dài của đường đi ngắn nhất có thể là ∞ hoặc – ∞.

đỉnh nguồn





không đạt tới được từ *s*

5.1. Bài toán đường đi ngắn nhất (ĐĐNN)

5.2. Tính chất của ĐĐNN, Giảm cận trên

5.3. Thuật toán Bellman-Ford

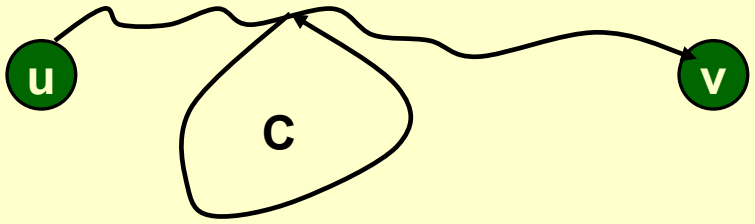
5.4. Thuật toán Dijkstra

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

5.6. Thuật toán Floyd-Warshal

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.



w(C) ≥ 0

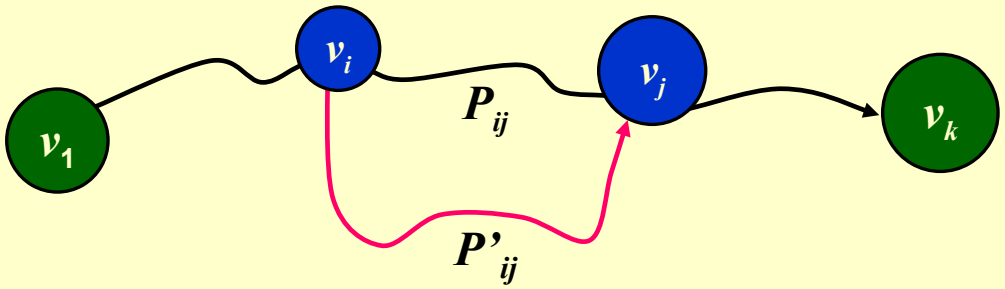
- ***Tính chất* 2.** *Mọi đường đi ngắn nhất trong đồ thị G đều đi qua không quá n-1 cạnh, trong đó n là số đỉnh.*
  - *Như là hệ quả của tính chất 1*

Các tính chất của ĐĐNN

Tính chất 3: Giả sử $P = \langle v_1, v_2, \ldots, v_k \rangle$ là đđnn từ v_1 đến v_k. Khi đó, $P_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$ là đđnn từ v_i đến v_j, với $1 \le i \le j \le k$.

(Bằng lời: Mọi đoạn đường con của đường đi ngắn nhất đều là đường đi ngắn nhất)

CM. Phản chứng. Nếu P_{ij} không là đđnn từ v_i đến v_j, thì tìm được P'_{ij} là đường đi từ v_i đến v_j thoả mãn $w(P'_{ij}) < w(P_{ij})$. Khi đó gọi P' là đường đi thu được từ P bởi việc thay đoạn P_{ij} bởi P'_{ij}, ta có $w(P') < w(P)$?!



Các tính chất của ĐĐNN

Ký hiệu: $\delta(u, v)$ = độ dài đđnn từ u đến v (gọi là khoảng cách từ u đến v)

Hệ quả: Giả sử P là đđnn từ s tới v, trong đó $P = s \xrightarrow{p'} u \rightarrow v$. Khi đó $\delta(s, v) = \delta(s, u) + w(u, v)$.

Tính chất 4: Giả sử $s \in V$. Đối với mỗi cạnh $(u,v) \in E$, ta có $\delta(s, v) \leq \delta(s, u) + w(u,v)$.

Đường đi ngắn nhất xuất phát từ một đỉnh
Single-Source Shortest Paths

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

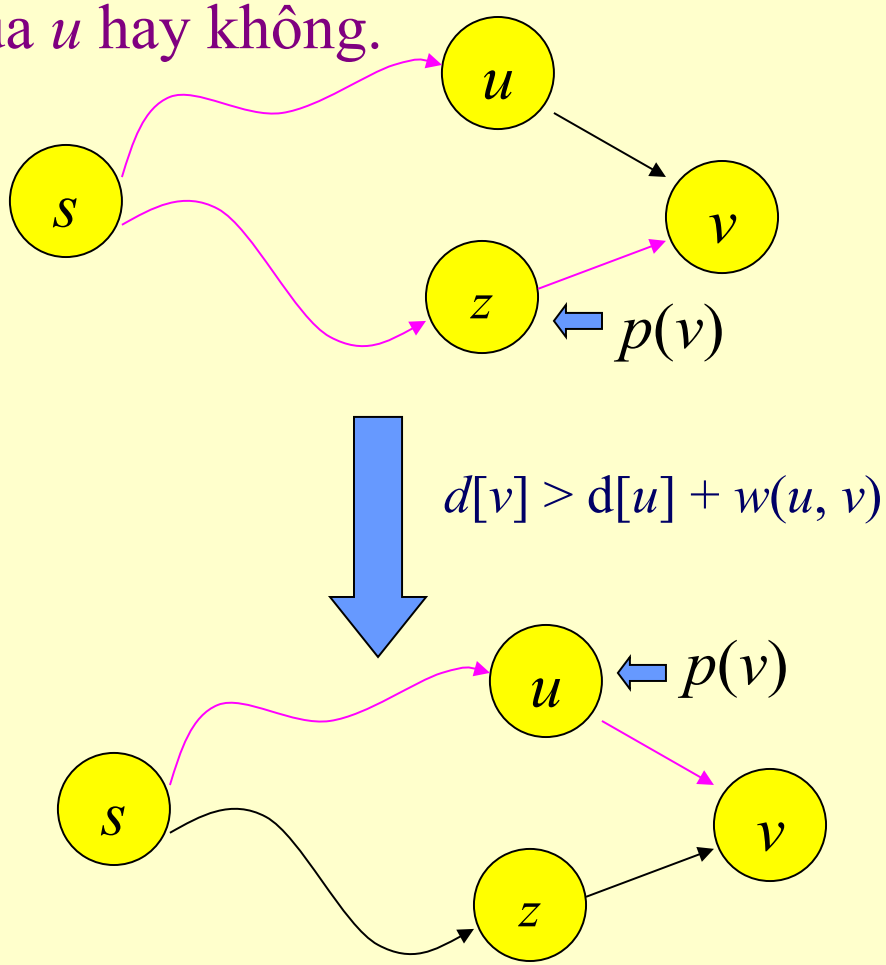
Khởi tạo (Initialization)

```
for v ∈ V(G)
      do d[v] ← ∞
         p[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]  + w(u, v)
        then d[v] ← d[u] + w(u, v)
             p[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

.

Nhận xét chung

- Việc cài đặt các thuật toán đợc thể hiện nhờ ***thủ tục gán nhãn***:
  - Mỗi đỉnh v sẽ có nhãn gồm 2 thành phần $(d[v], p[v])$. Nhãn sẽ biến đổi trong quá trình thực hiện thuật toán

- Nhận thấy rằng để tính khoảng cách từ s đến t, ở đây, ta phải tính khoảng cách từ s đến tất cả các đỉnh còn lại của đồ thị.

- Hiện nay vẫn cha biết thuật toán nào cho phép tìm đđnn nhất giữa hai đỉnh làm việc thực sự hiệu quả hơn những thuật toán tìm đđnn từ một đỉnh đến tất cả các đỉnh còn lại.

Nội dung

5.1. Bài toán đường đi ngắn nhất (ĐĐNN)

5.2. Tính chất của ĐĐNN, Giảm cận trên

5.3. Thuật toán Bellman-Ford

5.4. Thuật toán Dijkstra

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

5.6. Thuật toán Floyd-Warshal

Thuật toán Ford-Bellman

Richard Bellman
1920-1984

Lester R. Ford, Jr.
1927~

Thuật toán Ford-Bellman

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

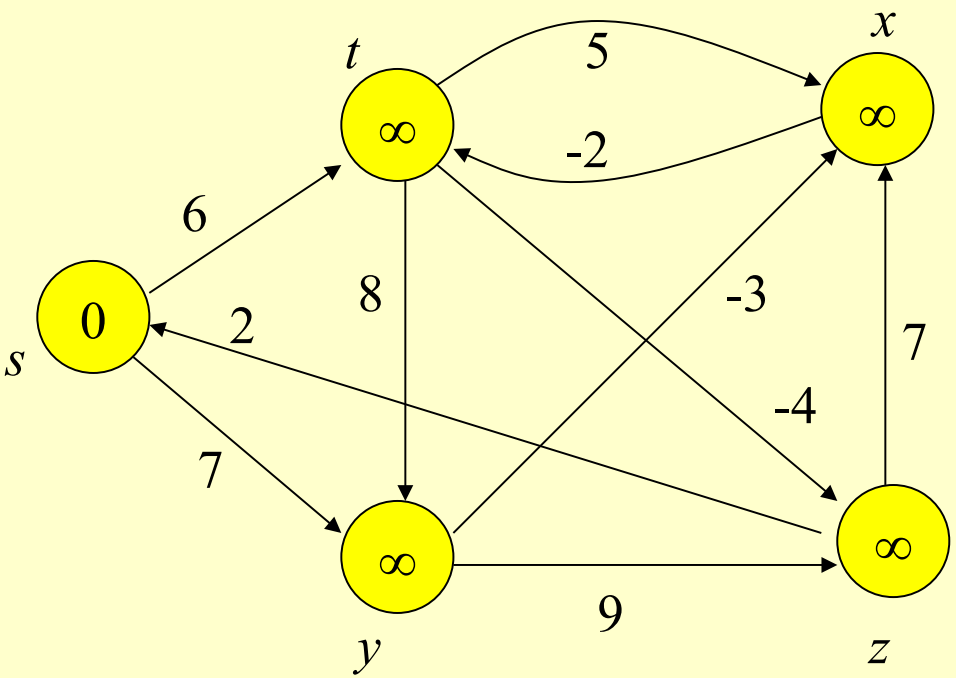
Mô tả thuật toán

```
procedure  Ford_Bellman;
begin
    for   v ∈ V  do  begin  (*   Khởi tạo  *)
          d[v] := w[s,v] ; p[v]:=s;
    end;
    d[s]:=0; p[s]:=s;
    for  k := 1  to  n-2  do        (* n = |V|  *)
          for   v ∈ V \ {s}   do
                for   u ∈ V do
                      if   d[v] > d[u] + w[u,v]   then
                      begin  d[v] := d[u] + w[u,v] ;
                             p[v] := u ;
                      end;
end;
```

Nhận xét

- Tính đúng đắn của thuật toán có thể chứng minh trên cơ sở nguyên lý tối u của quy hoạch động.
- Độ phức tạp tính toán của thuật toán là $O(n^3)$.
- Có thể chấm dứt vòng lặp theo k khi phát hiện trong quá trình thực hiện hai vòng lặp trong không có biến $d[v]$ nào bị đổi giá trị. Việc này có thể xảy ra đối với $k < n$-2, và điều đó làm tăng hiệu quả của thuật toán trong việc giải các bài toán thực tế.

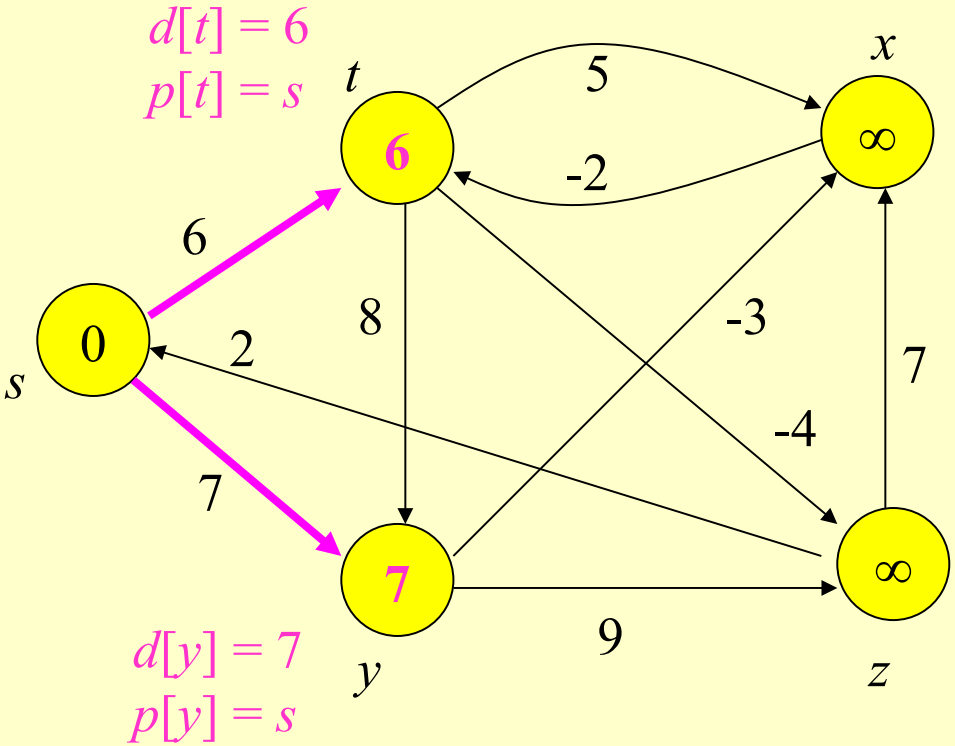
Ví dụ



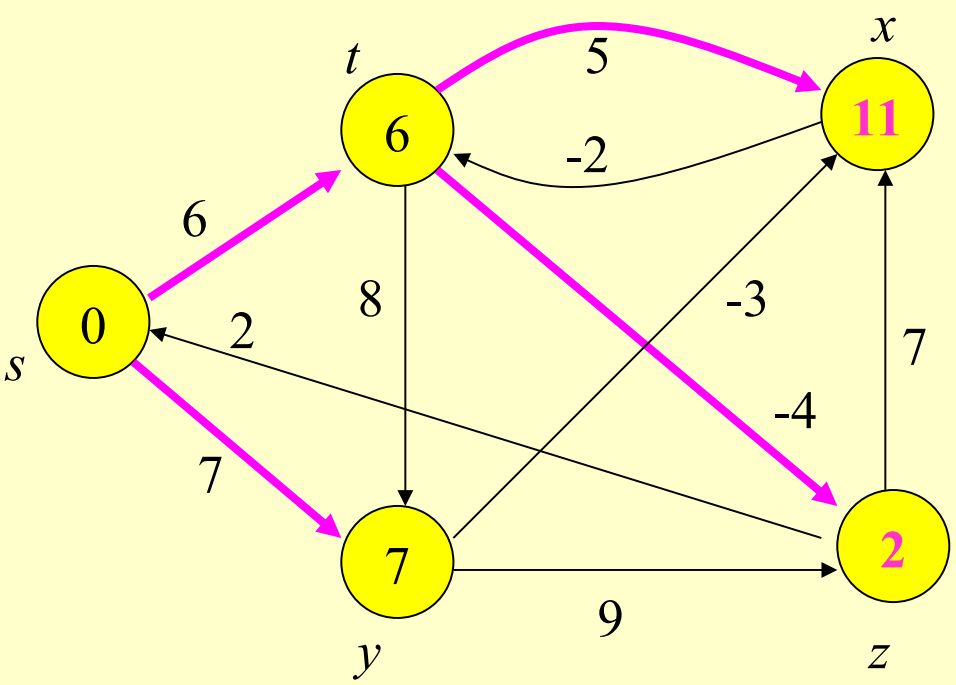
Source: *s*

Trình tự duyệt cạnh để giảm cận: (t, x), (t, y), (t, z), (x, t), (y, x), (y, z), (z, x), (z, s), (s, t), (s, y)

Lần 1



Lần 2

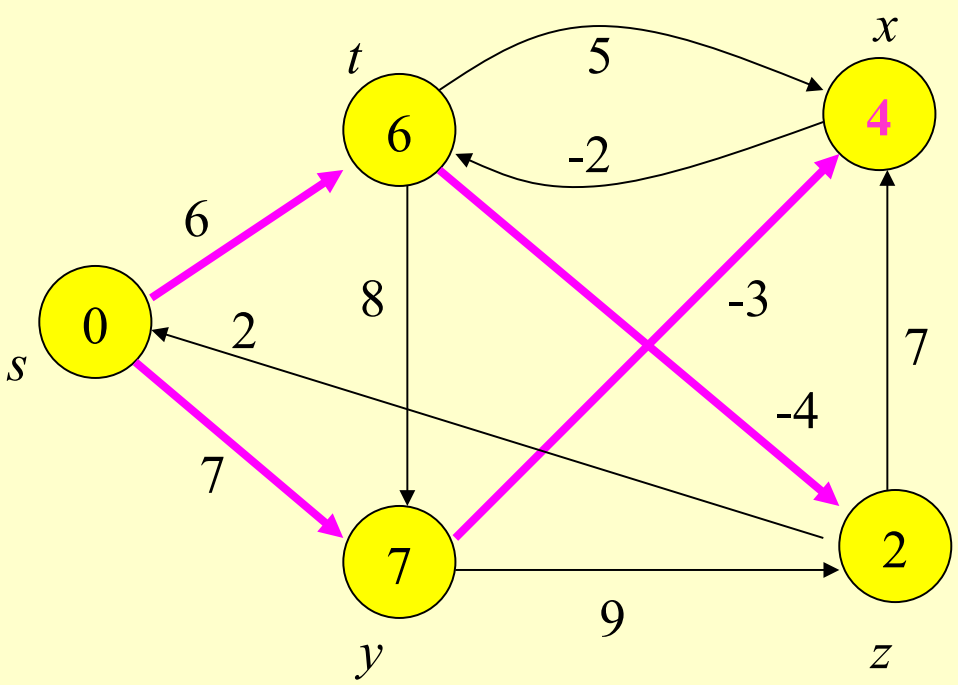


$d[x] = 11$
$p[x] = t$

$d[z] = 2$
$p[z] = t$

Relax (t, x), (t, y), (t, z), (x, t).

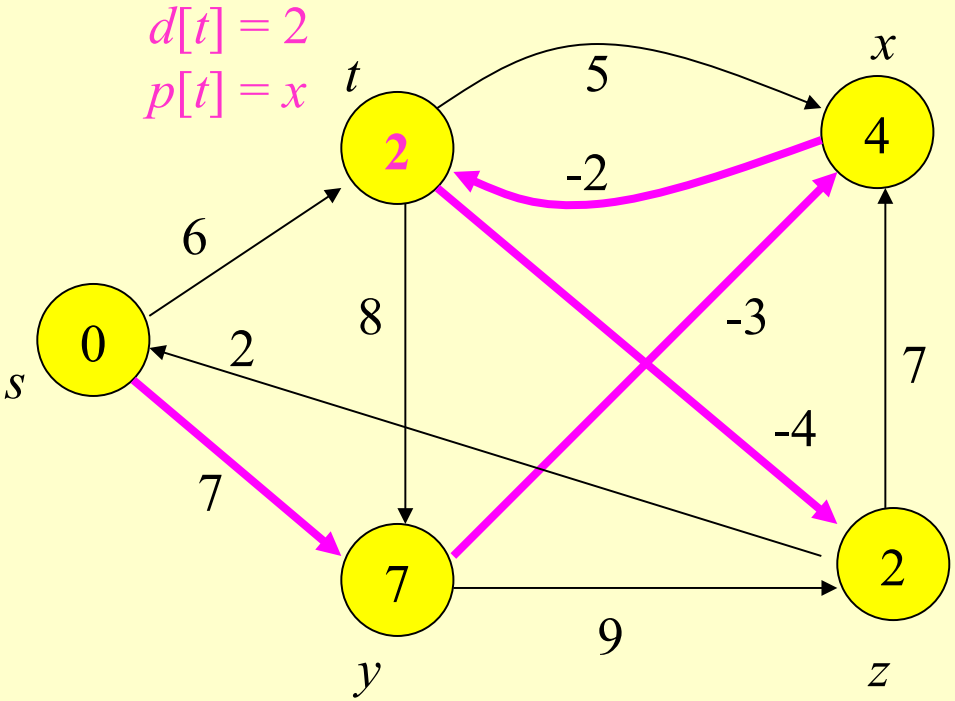
Lần 2 (tiếp)



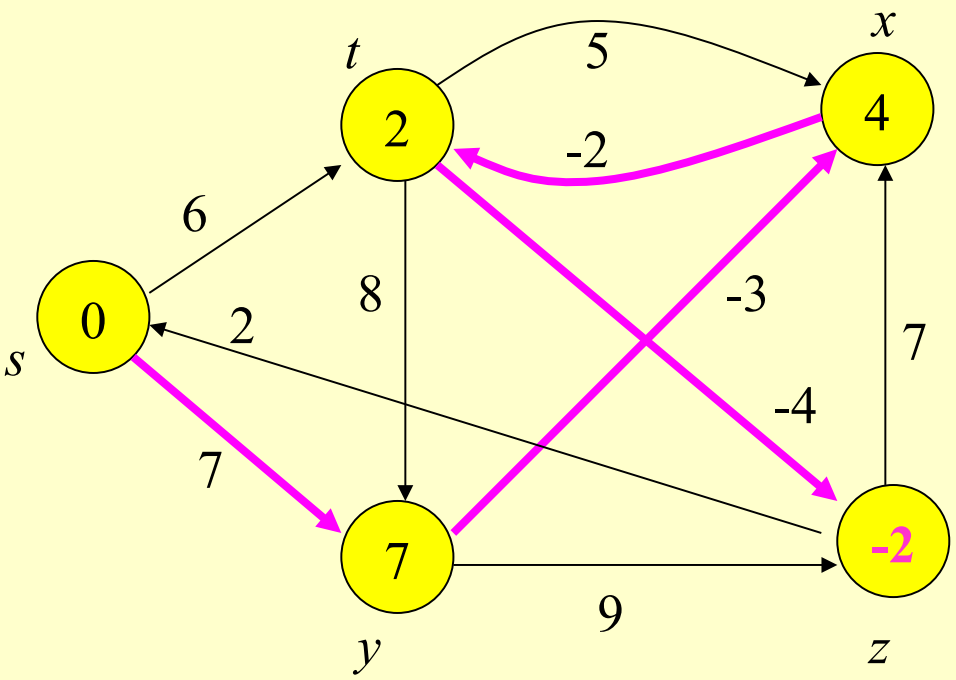
$d[x] = 4$
$p[x] = y$

Relax (y, x), (y, z), (z, x), (z, s), (z, s), (s, t), (s, y).

Lần 3



Lần 4



$d[z] = -2$

$p[z] = t$ (không đổi)

Nhận xét

- Đối với đồ thị tha tốt hơn là sử dụng danh sách kề Ke-(v), $v \in V$, để biểu diễn đồ thị, khi đó vòng lặp theo u cần viết lại dới dạng

```
for   u ∈ Ke-(v)  do
    if   d[v] > d[u] + w[u,v]   then
    begin
        d[v] := d[u] + w[u,v] ;
        p[v] := u ;
    end;
```

- Thuật toán có độ phức tạp $O(n.m)$.

Nội dung

5.1. Bài toán đường đi ngắn nhất (ĐĐNN)

5.2. Tính chất của ĐĐNN, Giảm cận trên

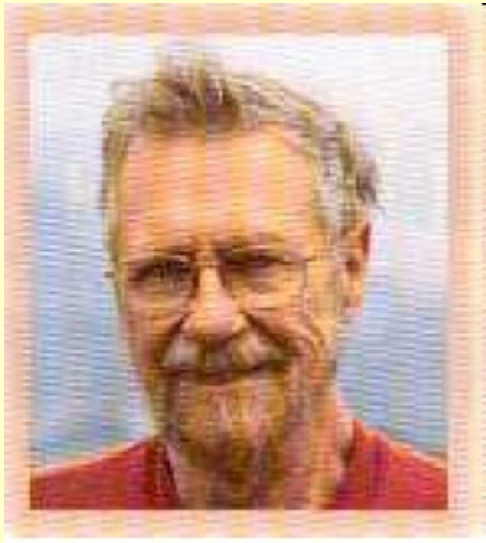
5.3. Thuật toán Bellman-Ford

5.4. Thuật toán Dijkstra

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

5.6. Thuật toán Floyd-Warshal

Thuật toán Dijkstra



- Trong trờng 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.
- Thuật toán đợc xây dựng dựa trên thủ tục gán nhãn. Thoạt tiên nhãn của các đỉnh là tạm thời. Ở mỗi một bớc lặp có một nhãn tạm thời trở thành nhãn cố định. Nếu nhãn của một đỉnh u trở thành cố định thì $d[u]$ sẽ cho ta độ dài của đđnn từ đỉnh s đến u. Thuật toán kết thúc khi nhãn của tất cả các đỉnh trở thành cố định.

Edsger W.Dijkstra

(1930-2002)

Thuật toán Dijkstra

- ***Đầu vào:*** *Đồ thị có hướng* $G=(V,E)$ *với* n *đỉnh,*
  $s \in V$ *là đỉnh xuất phát,*
  $w[u,v]$, $u,v \in V$ *- ma trận trọng số;*
- ***Giả thiết:*** $\boldsymbol{w[u,v] \geq 0, \quad u, v \in V.}$
- **Đầu ra:** *Với mỗi* $v \in V$
  - $d[v] = \delta(s, v)$;
  - $p[v]$ - *đỉnh đi trước v trong đđnn từ s đến v.*

Thuật toán Dijkstra

Tập S: Chỉ cần cho chứng minh định lý

```
procedure Dijkstra;
begin
    for   v ∈ V  do  begin     (*   Khởi tạo  *)
                        d[v] := w[s,v] ;  p[v]:=s;
                     end;
    d[s] := 0;   S := {s};                  (*  S — tập đỉnh có nhãn cố định *)
    T := V \ {s};                           (*   T là tập các đỉnh có nhãn tạm thời   *)
    while  T  ≠ ∅  do                       (*     Bớc lặp   *)
    begin
        Tìm đỉnh  u ∈ T thoả mãn  d[u] = min{ d[z] :  z  ∈  T};
        T :=  T \ {u};  S:= S  ∪ {u};       (*  Cố định nhãn của đỉnh u *)
        for   v ∈ T   do                    (*  Gán nhãn lại cho các đỉnh trong T *)
            if   d[v] > d[u] + w[u,v]   then begin
                    d[v] := d[u] + w[u,v] ;  p[v] := u ;
            end;
    end;
end;
```

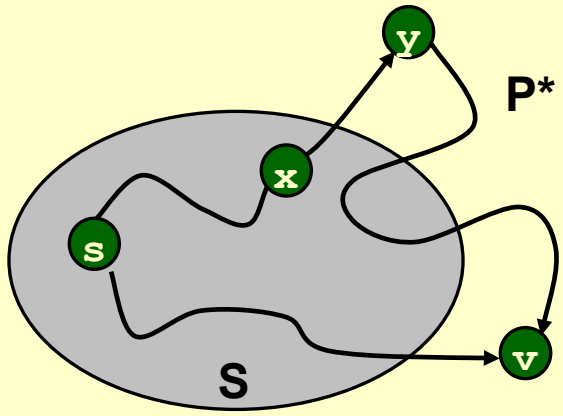
Thuật toán Dijkstra

- **Chú ý:** Nếu chỉ cần tìm đờng 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 đợc đờng đi ngắn nhất từ đỉnh s đến tất cả các đỉnh còn lại trên đồ thị sau thời gian* $O(n^2)$.

- CM: Rõ ràng thời gian tính là $O(n^2)$

Chứng minh tính đúng đắn của Thuật toán Dijkstra



❖Ta sẽ CM với mỗi v ∈ S, d(v) = δ(*s, v*).

- Qui nạp theo |S|.
- Cơ sở qui nạp: Với |S| = 1, rõ ràng là đúng.

- Chuyển qui nạp:
  - giả sử thuật toán Dijkstra bổ sung v vào S
  - d(v) là độ dài của một đường đi từ s đến v
  - nếu d(v) không là độ dài đđnn từ s đến v, thì gọi P* là đđnn từ s đến v
  - P* phải sử dụng cạnh ra khỏi S, chẳng hạn (x, y)
  - khi đó d(v) > δ(s, v) giả thiết

    = δ(s, x) + w(x, y) + δ(y, v) tính chất 3

    ≥ δ(s, x) + w(x, y) δ(y, v) là không âm

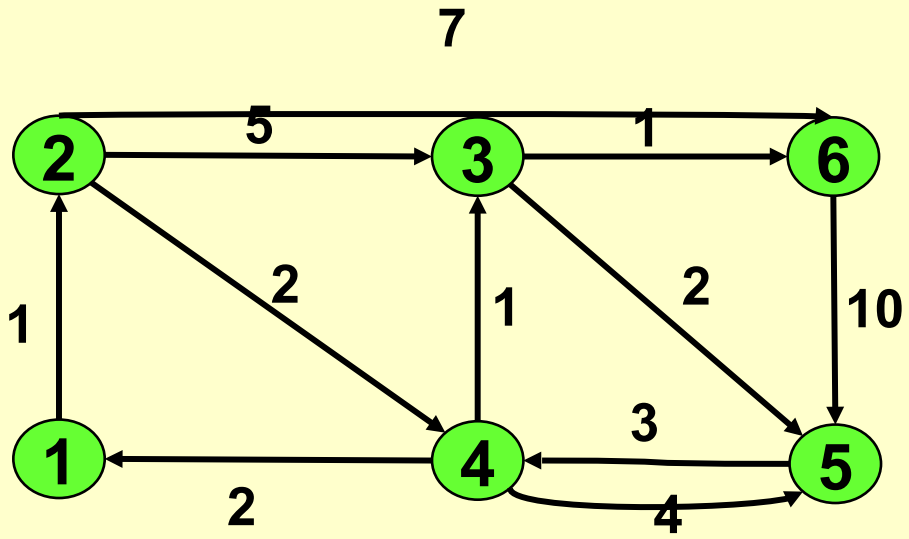
    = d(x) + w(x, y) giả thiết quy nạp

    ≥ d(y) theo thuật toán

    vì thế thuật toán Dijkstra phải chọn y thay vì chọn v ?!

Ví dụ

Tìm đường đi ngắn nhất từ đỉnh 1 đến tất cả các đỉnh còn lại

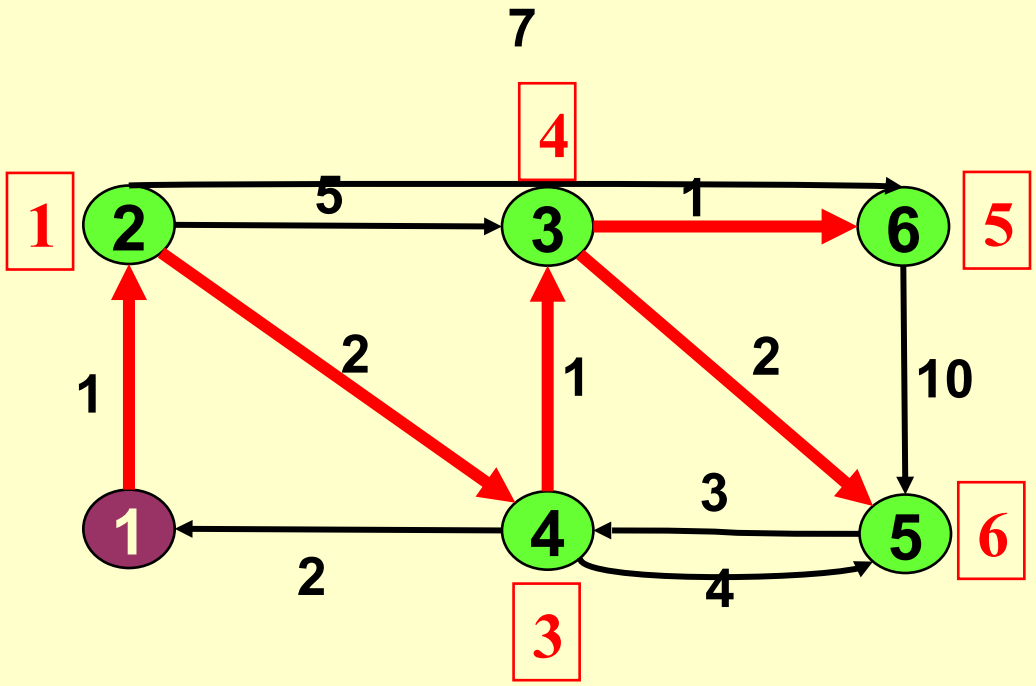


	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
Khởi tạo	[0, 1]	[1, 1]*	[∞, 1]	[∞, 1]	[∞, 1]	[∞, 1]
1	-	-	[6, 2]	[3, 2]*	[∞, 1]	[8, 2]
2	-	-	[4, 4]*	-	[7, 4]	[8, 2]
3	-	-	-	-	[6, 3]	[5, 3]*
4	-	-	-	-	[6, 3]*	-
5	-	-	-	-	-	-

Cây đường đi ngắn nhất

- Tập cạnh $\{(p(v), v): v \in V \setminus \{s\}\}$ tạo thành cây có gốc tại đỉnh nguồn s được gọi là cây đđnn xuất phát từ đỉnh s.

- Các cạnh màu đỏ tạo thành cây đđnn xuất phát từ đỉnh 1
- Số màu đỏ viết bên cạnh mỗi đỉnh là độ dài đường đi ngắn nhất từ 1 đến nó.



Nội dung

5.1. Bài toán đường đi ngắn nhất (ĐĐNN)

5.2. Tính chất của ĐĐNN, Giảm cận trên

5.3. Thuật toán Bellman-Ford

5.4. Thuật toán Dijkstra

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

5.6. Thuật toán Floyd-Warshal

Đường đi trong đồ thị không có chu trình

Shortest Paths In Directed Acyclic Graphs

Đường đi trong đồ thị không có chu trình

- Một trờng hợp riêng của bài toán đờng đi ngắn nhất giải đợc 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 hớng từ đỉnh có chỉ số nhỏ hơn đến đỉnh có chỉ số lớn hơn, nghĩa là mỗi cung của nó có thể biểu diễn dới dạng* $(v[i], v[j])$, *trong đó* $i < j$.

Thuật toán đánh số đỉnh

- Trớc hết nhận thấy rằng: *Trong đồ thị không có chu trình bao giờ cũng tìm đợc đỉnh có bán bậc vào bằng 0*. Thực vậy, bắt đầu từ đỉnh v_1 nếu có cung đi vào nó từ v_2 thì ta lại chuyển sang xét đỉnh v_2. Nếu có cung từ v_3 đi vào v_2, thì ta lại chuyển sang xét v_3, ... Do đồ thị là không có chu trình nên sau một số hữu hạn lần chuyển nh vậy ta phải đi đến đỉnh không có cung đi vào.
- Thuật toán đợc xây dựng dựa trên ý tởng rất đơn giản sau: Thoạt tiên, tìm các đỉnh có bán bậc vào bằng 0. Rõ ràng ta có thể đánh số chúng theo một thứ tự tuỳ ý bắt đầu từ 1. Tiếp theo, loại bỏ khỏi đồ thị những đỉnh đã đợc đánh số cùng các cung đi ra khỏi chúng, ta thu đợc đồ thị mới cũng không có chu trình, và thủ tục đợc lặp lại với đồ thị mới này. Quá trình đó sẽ đợc tiếp tục cho đến khi tất cả các đỉnh của đồ thị đợc đánh số.

Thuật toán đánh số đỉnh

- **Đầu vào:** *Đồ thị có hướng $G=(V,E)$ với n đỉnh không chứa chu trình đợc 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

```
procedure  Numbering;
begin
     for   v ∈  V  do   Vao[v] := 0;
    for   u ∈  V  do       (*  Tính Vao[v] = bán bậc vào của v  *)
         for   v ∈ Ke(u)  do   Vao[v] := Vao[v] + 1 ;
      QUEUE := ⊘ ;
      for   v ∈  V  do
            if   Vao[v] = 0  then  QUEUE ⇐  v ;
       num := 0;
       while  QUEUE  ≠ ⊘   do
       begin
            u  ⇐  QUEUE ;  num := num + 1 ;   NR[u] := num ;
            for   v ∈ Ke(u)  do  begin
                                    Vao[v] := Vao[v] - 1 ;
                                     if   Vao[v] = 0  then  QUEUE ⇐  v ;
                                end;
       end;
end;
```

Thuật toán đánh số đỉnh

- Rõ ràng trong bớc 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 $O(m)$.

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 đợc đánh số sao cho mỗi cung chỉ đi từ đỉnh có chỉ số nhỏ đến đỉnh có chỉ số lớn hơn.

- ***Thuật toán tìm đờng đi ngắn nhất từ đỉnh nguồn v[1] đến tất cả các đỉnh còn lại trênđồ 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 đợc cho bởi danh sách kề $Ke(v)$, $v \in V$.
- **Đầu ra:** *Khoảng cách từ* $v[1]$ *đến tất cả các đỉnh còn lại đợc ghi trong mảng* $d[v[i]]$, $i = 2, 3, \ldots, n$

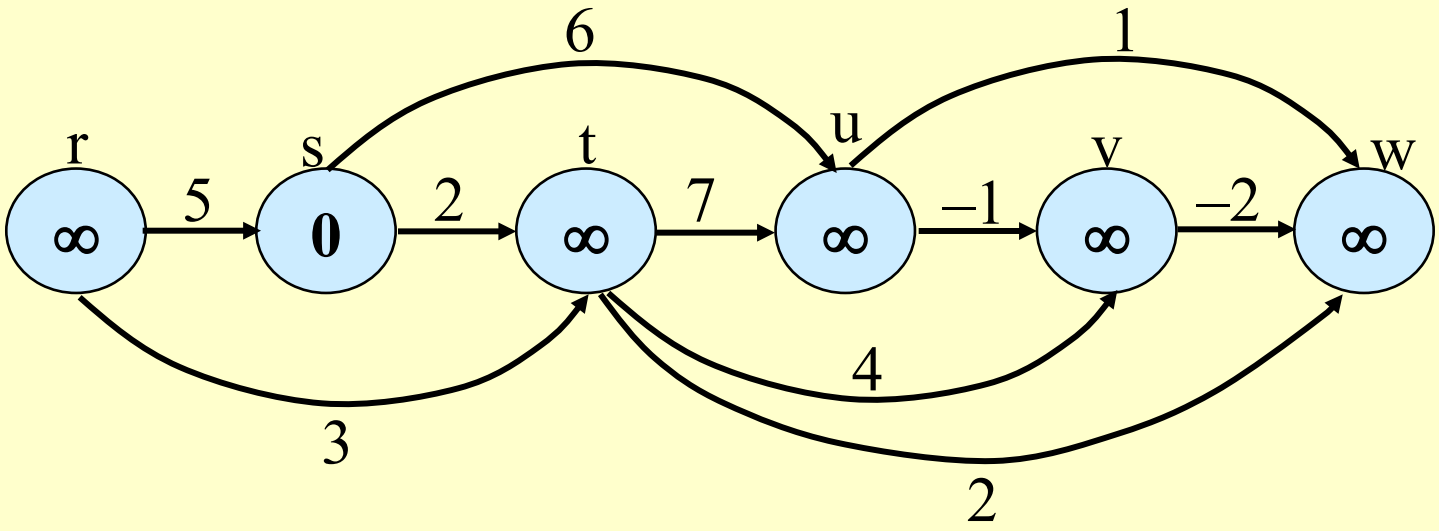
Thuật toán tìm đđnn trên đồ thị không có chu trình

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

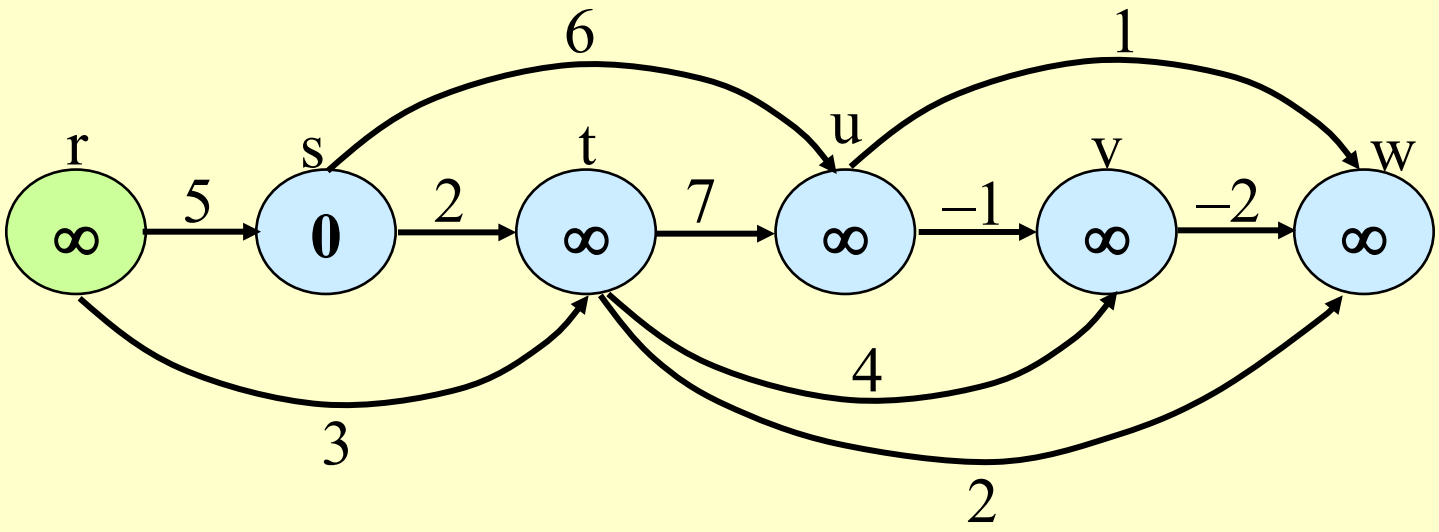
- **Độ phức tạp tính toán của thuật toán là O(*m*), do mỗi cung của đồ thị phải xét qua đúng một lần.**

Ví dụ

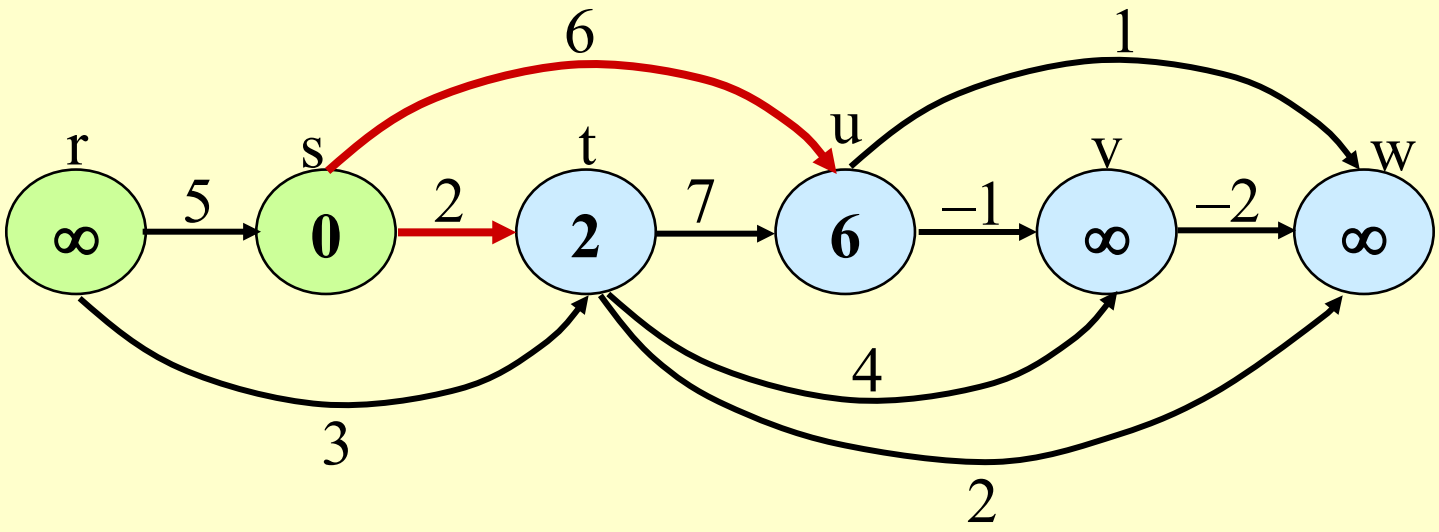
Cần tìm đường đi ngắn nhất từ ***s*** đến tất cả các đỉnh đạt đến được từ nó



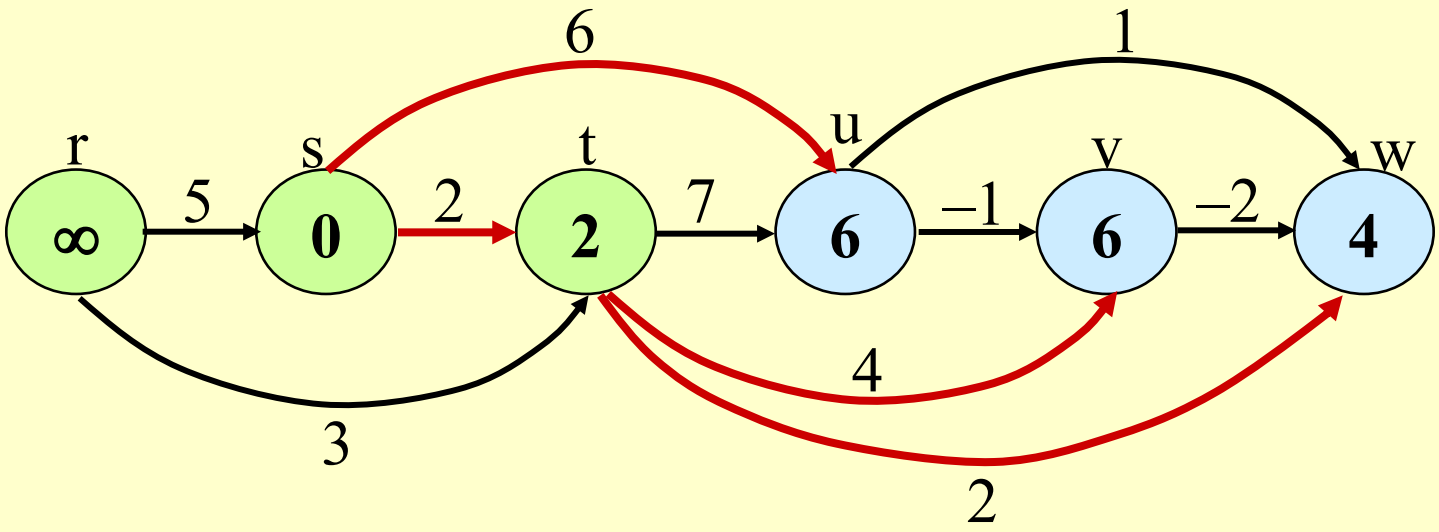
Ví dụ



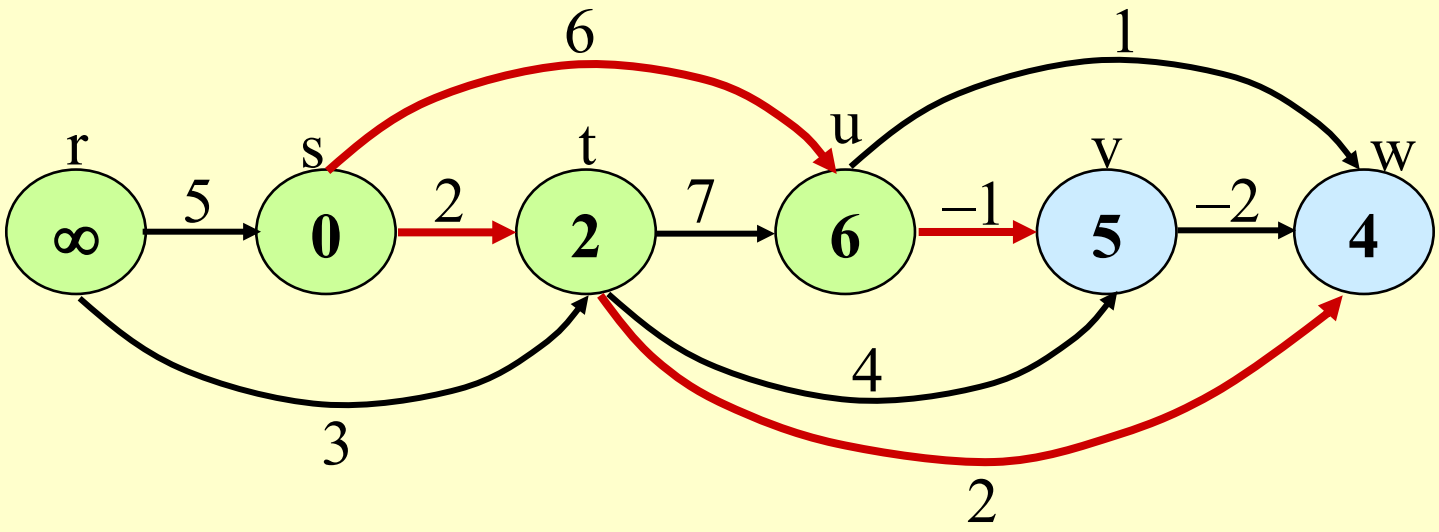
Ví dụ



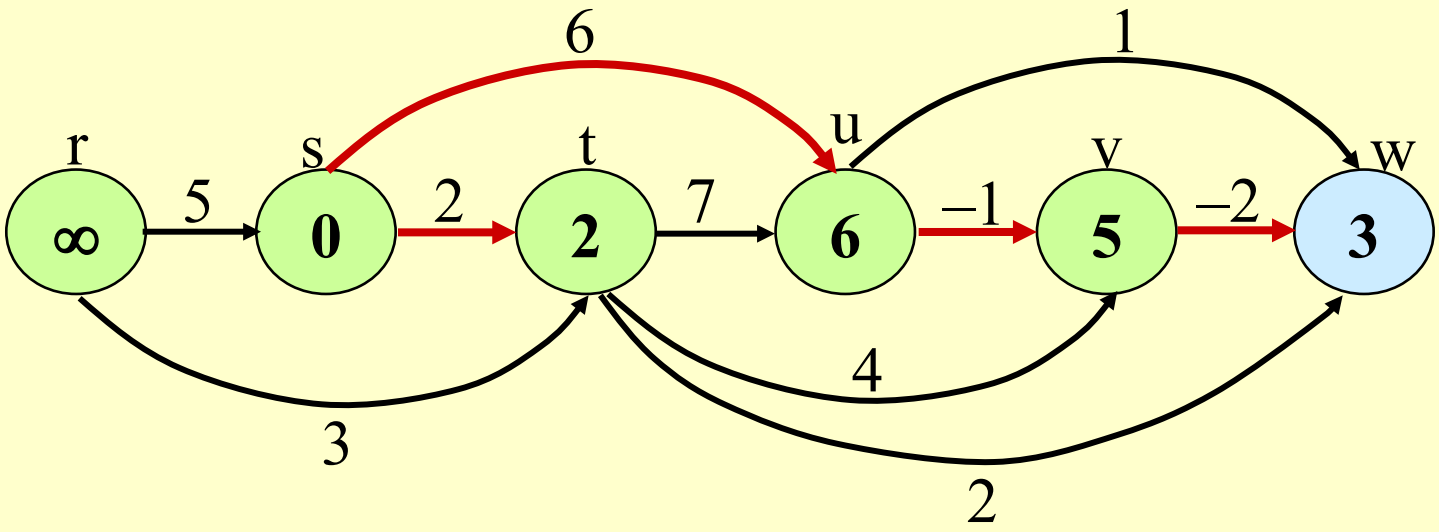
Ví dụ



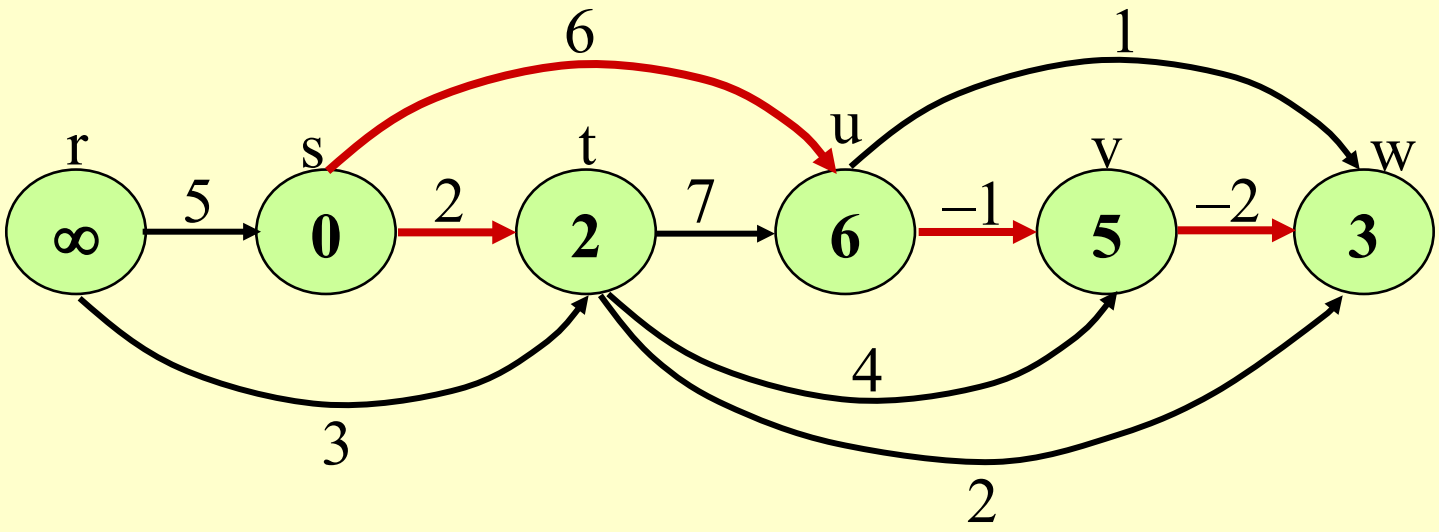
Ví dụ



Ví dụ



Ví dụ



Kết quả: Cây đường đi ngắn nhất từ *s* thể hiện bởi các cung màu đỏ

Ứng dụng: PERT

- Xây dựng phơng 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 đợc chia ra làm n công đoạn, đánh số từ 1 đến n. Có một số công đoạn mà việc thực hiện nó chỉ đợc 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 ($i = 1, 2,..., n$).

Ứng dụng: PERT

- Các dữ liệu với $n = 8$ đợc 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

Ứng dụng: PERT

- **Bài toán PERT:** Giả sử 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 đợc bắt đầu thc hiện vào thời điểm nào) để cho công trình đợc hoàn thành xong trong thời điểm sớm nhất có thể đợc.

- Ta có thể xây dựng đồ thị có hớng n đỉnh biểu diễn ràng buộc về trình tự thực hiệc các công việc nh sau:
  - Mỗi đỉnh của đồ thị tơng ứng với một công việc.
  - Nếu công việc i phải đợc thực hiện trớc công đoạn j thì trên đồ thị có cung (i,j), trọng số trên cung này đợc gán bằng $t[i]$

Thuật toán PERT

- Thêm vào đồ thị 2 đỉnh 0 và n+1 tơng ứng với hai sự kiện đặc biệt:
  - đỉnh số 0 tơng ứng với công đoạn *Lễ khởi công*, nó phải đợc thực hiện trớc tất cả các công đoạn khác, và
  - đỉnh n+1 tơng ứng với công đoạn *Cắt băng khánh thành công trình*, nó phải thực hiện sau tất cả các công đoạn,
  - với $t[0] = t[n+1] = 0$ (trên thực tế chỉ cần nối đỉnh 0 với tất cả các đỉnh có bán bậc vào bằng 0 và nối tất cả các đỉnh có bán bậc ra bằng 0 với đỉnh n+1).

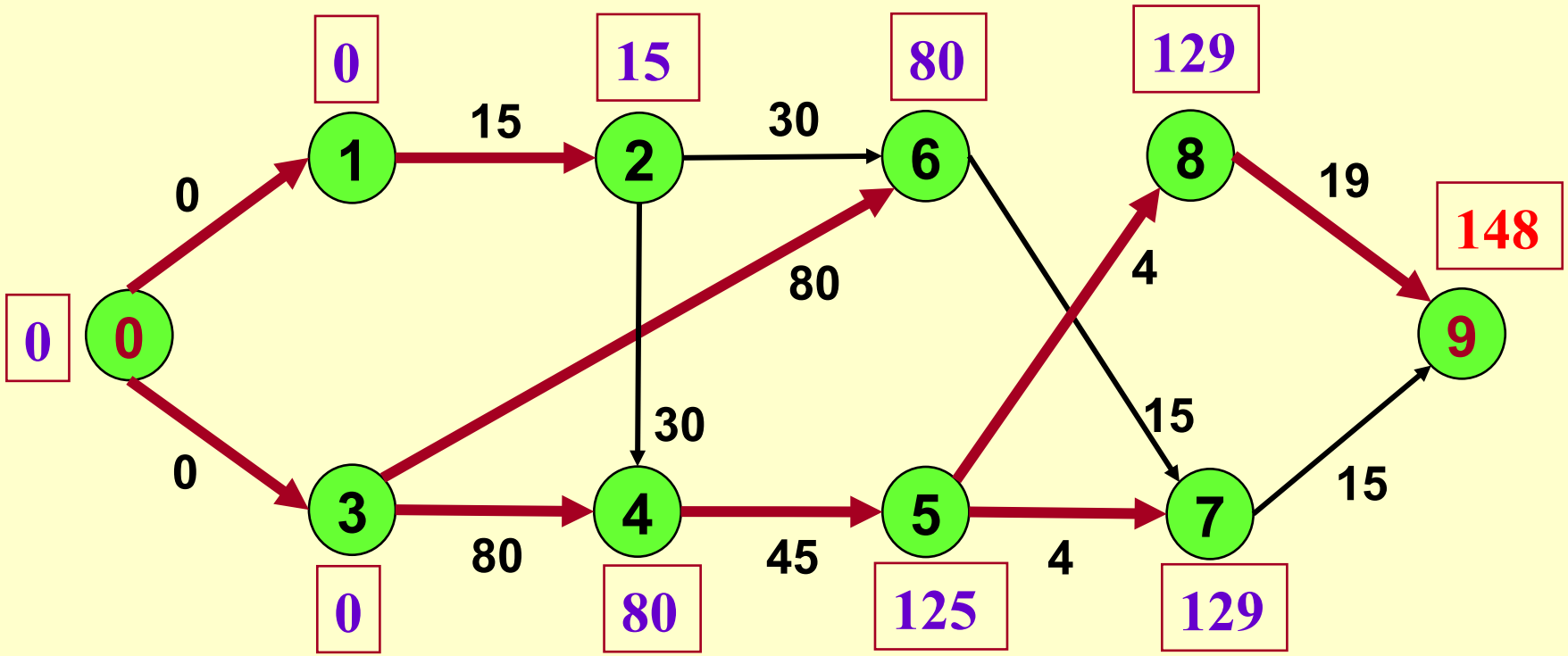
  Gọi đồ thị thu đợc là G.
- Rõ ràng bài toán đặt ra dẫn về bài toán tìm đờng đi dài nhất từ đỉnh 0 đến tất cả các đỉnh còn lại trên đồ thị G.

Thuật toán PERT

- Do đồ thị G không chứa chu trình, nên để giải bài toán đặt ra có thể áp dụng thuật toán Critical_Path trong đó chỉ cần đổi toán tử ***min*** thành toán tử ***max***.
- Kết thúc thuật toán, ta thu đợc $d[v]$ là độ dài đờng đi dài nhất từ đỉnh 0 đến đỉnh v.
- Khi đó $d[v]$ cho ta thời điểm sớm nhất có thể bắt đầu thực hiện công đoạn v, nói riêng $d[n+1]$ là thời điểm sớm nhất có thể cắt băng khánh thành, tức là thời điểm sớm nhất có thể hoàn thành toàn bộ công trình.

PERT: Ví dụ minh hoạ

- Qui bài toán PERT về tìm đường đi dài nhất trên đồ thị không có chu trình



Nội dung

5.1. Bài toán đường đi ngắn nhất (ĐĐNN)

5.2. Tính chất của ĐĐNN, Giảm cận trên

5.3. Thuật toán Bellman-Ford

5.4. Thuật toán Dijkstra

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

5.6. Thuật toán Floyd-Warshal

ĐƯỜNG ĐI NGẮN NHẤT GIỮA MỌI CẶP ĐỈNH

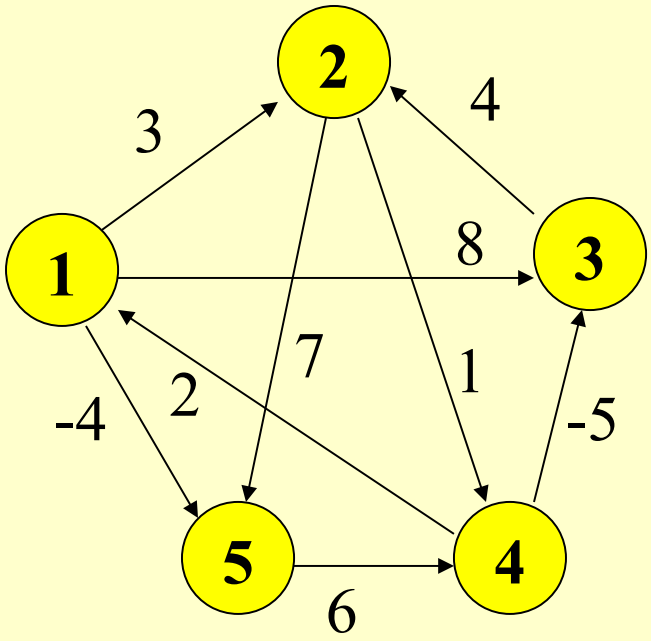
All-Pairs Shortest Paths

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.**

Ví dụ



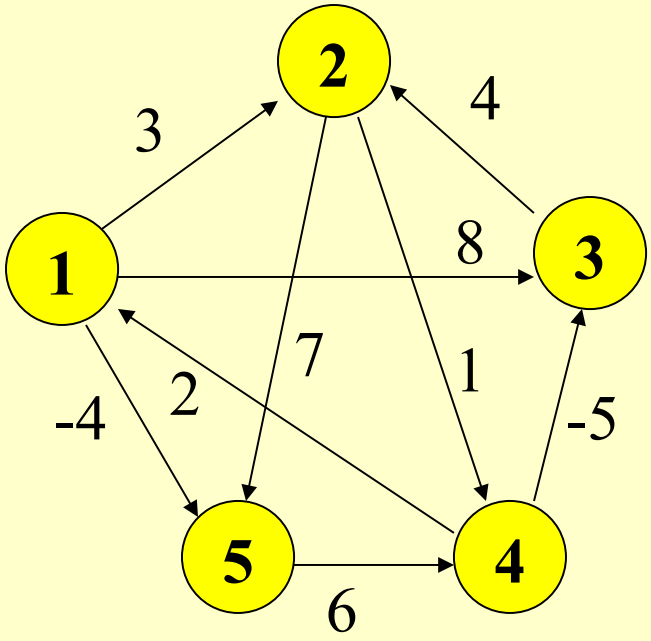
Đầu vào

$n \times n$ ma trận $W = (w_{ij})$ với

$$w_{ij} = \begin{cases} 0 & \text{nếu } i = j \\ w(i,j) & \text{nếu } i \neq j \ \& \ (i,j) \in E \\ \infty & \text{còn lại} \end{cases}$$

$$\begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix}$$

Tiếp



Đầu ra

Đường đi: 1- 5 - 4 - 3 - 2

= – 4 + 6 – 5 + 4

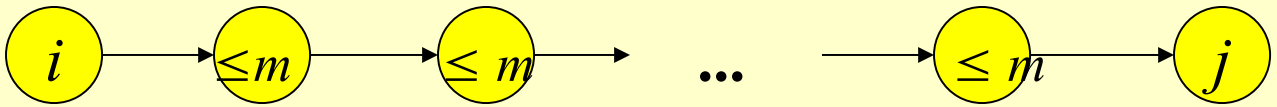
0	**1**	-3	2	-4
3	0	-4	1	-1
7	4	0	5	3
2	-1	-5	0	**-2**
8	5	1	6	0

4 - 1- 5

5 - 4 - 1

Thuật toán Floyd-Warshall

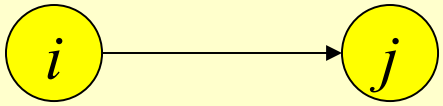
$d_{ij}^{(m)}$ = độ dài đường đi ngắn nhất từ i đến j sử dụng các đỉnh trung gian trong tập đỉnh { 1, 2, …, m }.



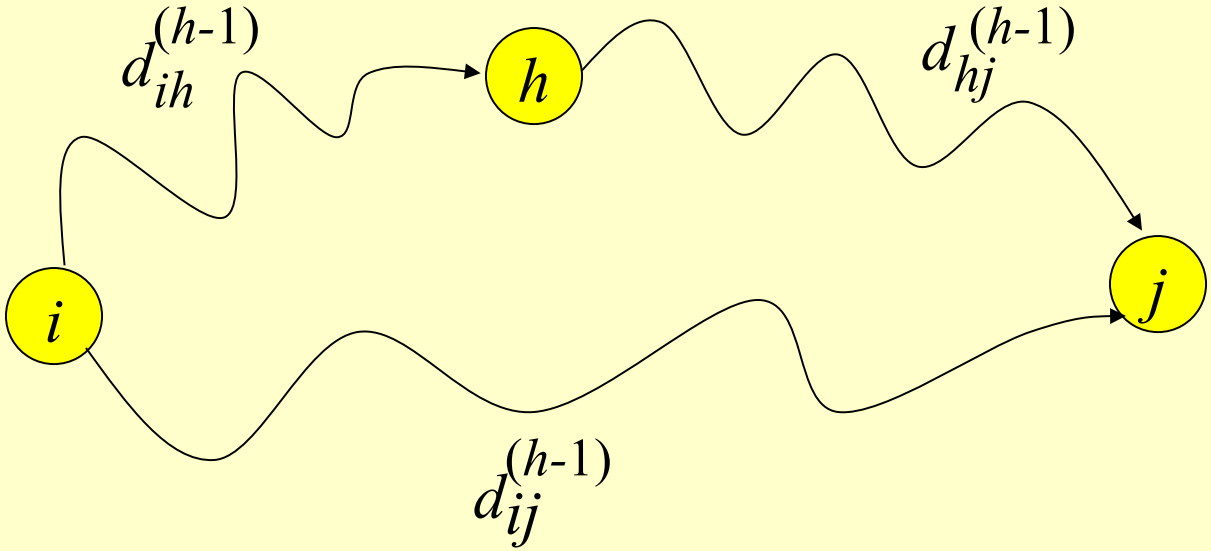
Khi đó độ dài đường đi ngắn nhất từ i đến j là $d_{ij}^{(n)}$

Công thức đệ qui tính $d^{(h)}$

- $d_{ij}^{(0)} = w_{ij}$



- $d_{ij}^{(h)} = \min (d_{ij}^{(h-1)}, d_{ih}^{(h-1)} + d_{hj}^{(h-1)})$ nếu $h \geq 1$



Thuật toán Floyd-Warshall

Floyd-Warshall(n, W)

$D^{(0)} \leftarrow W$

for $k \leftarrow 1$ to n do

  for $i \leftarrow 1$ to n do

    for $j \leftarrow 1$ to n do

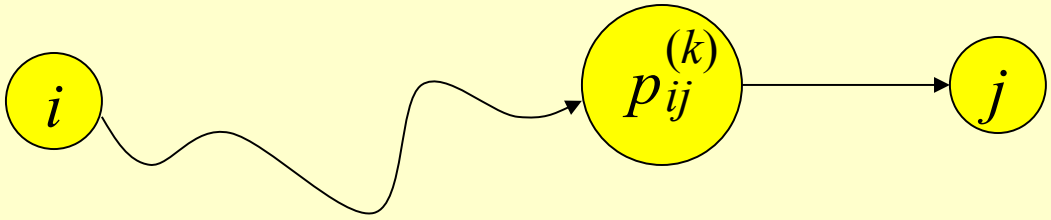
      $d_{ij}^{(k)} \leftarrow \min (d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$

return $D^{(n)}$

Thời gian tính $\Theta(n^3)$!

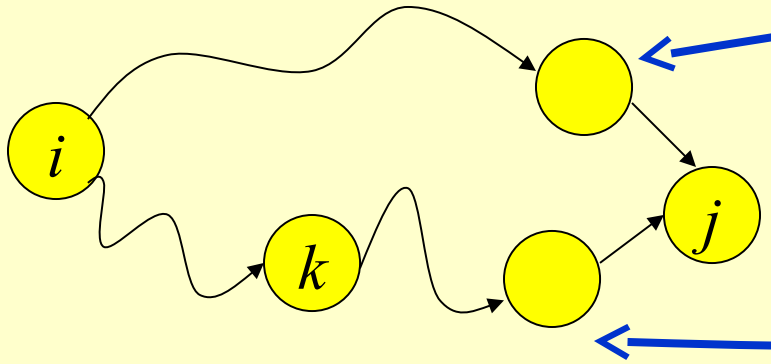
Xây dựng đường đi ngắn nhất

Predecessor matrix $P^{(k)} = (p_{ij}^{(k)})$:



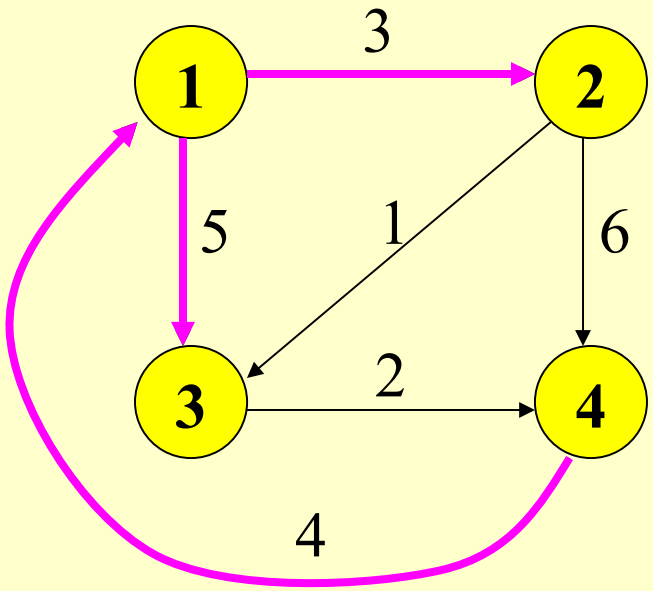
đường đi ngắn nhất từ i đến j chỉ qua các đỉnh trung gian trong $\{1, 2, \ldots, k\}$.

$$p_{ij}^{(0)} = \begin{cases} i, & \text{nếu } (i,j) \in E \\ \text{NIL}, & \text{nếu } (i,j) \notin E \end{cases}$$



$$p_{ij}^{(k)} = \begin{cases} p_{ij}^{(k-1)} & \text{nếu } d_{ij}^{(k-1)} \le d_{ik}^{(k-1)} + d_{kj}^{(k-1)} \\ p_{kj}^{(k-1)} & \text{trái lại} \end{cases}$$

Ví dụ



$$D^{(0)} \begin{pmatrix} 0 & 3 & 5 & \infty \\ \infty & 0 & 1 & 6 \\ \infty & \infty & 0 & 2 \\ 4 & \infty & \infty & 0 \end{pmatrix}$$

$$P^{(0)} \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} \\ \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & \text{NIL} & \text{NIL} & 3 \\ 4 & \text{NIL} & \text{NIL} & \text{NIL} \end{pmatrix}$$

Có thể sử dụng 1 là đỉnh trung gian:

$$D^{(1)} \begin{pmatrix} 0 & 3 & 5 & \infty \\ \infty & 0 & 1 & 6 \\ \infty & \infty & 0 & 2 \\ 4 & 7 & 9 & 0 \end{pmatrix}$$

$$P^{(1)} \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} \\ \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & \text{NIL} & \text{NIL} & 3 \\ 4 & 1 & 1 & \text{NIL} \end{pmatrix}$$

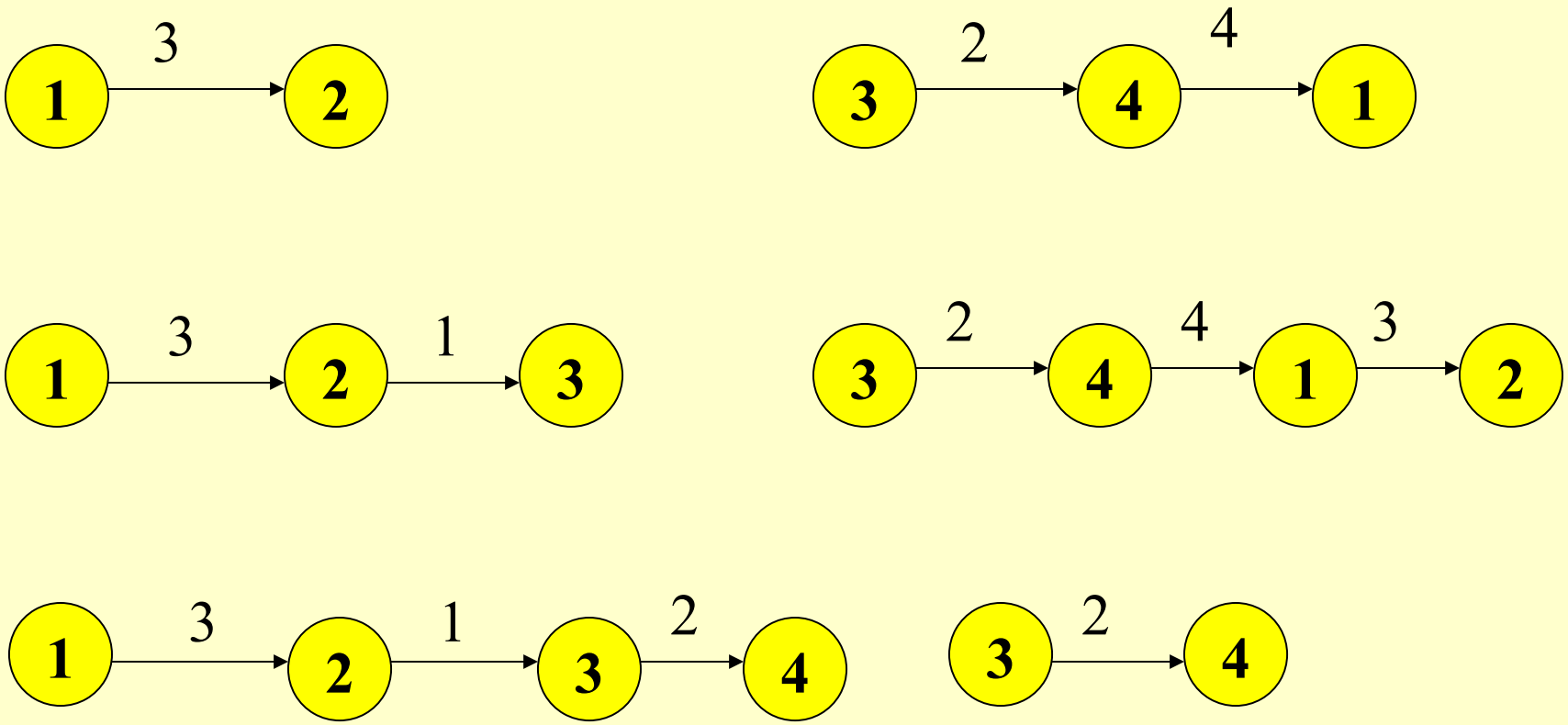
Ví dụ (tiếp)

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 4 & 9 \\ \infty & 0 & 1 & 6 \\ \infty & \infty & 0 & 2 \\ 4 & 7 & 8 & 0 \end{pmatrix} \qquad P^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 2 & 2 \\ \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & \text{NIL} & \text{NIL} & 3 \\ 4 & 1 & 2 & \text{NIL} \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 4 & 6 \\ \infty & 0 & 1 & 3 \\ \infty & \infty & 0 & 2 \\ 4 & 7 & 8 & 0 \end{pmatrix} \qquad P^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 2 & 3 \\ \text{NIL} & \text{NIL} & 2 & 3 \\ \text{NIL} & \text{NIL} & \text{NIL} & 3 \\ 4 & 1 & 2 & \text{NIL} \end{pmatrix}$$

$$D^{(4)} = \begin{pmatrix} 0 & 3 & 4 & 6 \\ 7 & 0 & 1 & 3 \\ 6 & 9 & 0 & 2 \\ 4 & 7 & 8 & 0 \end{pmatrix} \qquad P^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 2 & 3 \\ 4 & \text{NIL} & 2 & 3 \\ 4 & 1 & \text{NIL} & 3 \\ 4 & 1 & 2 & \text{NIL} \end{pmatrix}$$

Ví dụ (tiếp)



Thuật toán Floyd-Warshall

Floyd-Warshall(n, W)

$D \leftarrow W$

for $k \leftarrow 1$ to n do

for $i \leftarrow 1$ to n do

for $j \leftarrow 1$ to n do

$d_{ij} \leftarrow \min (d_{ij}, d_{ik} + d_{kj})$

return D

Thời gian tính $\Theta(n^3)$!

Robert W. Floyd, 1936-2001

- Born in New York, Floyd finished school at age 14. At the University of Chicago, he received a Bachelor's degree in liberal arts in 1953 (when still only 17) and a second Bachelor's degree in physics in 1958.
- Becoming a computer operator in the early 1960s, he began publishing many noteworthy papers and was appointed an associate professor at Carnegie Mellon University by the time he was 27 and became a full professor at Stanford University six years later. He obtained this position without a Ph.D.
- Turing Award, 1978.

Stephen Warshall

- 1935 – 2006
- Proving the correctness of the transitive closure algorithm for boolean circuit.
  - (Wikipedia) There is an interesting anecdote about his proof that the transitive closure algorithm, now known as Warshall's algorithm, is correct. He and a colleague at Technical Operations bet a bottle of rum on who first could determine whether this algorithm always works. Warshall came up with his proof overnight, winning the bet and the rum, which he shared with the loser of the bet. Because Warshall did not like sitting at a desk, he did much of his creative work in unconventional places such as on a sailboat in the Indian Ocean or in a Greek lemon orchard.

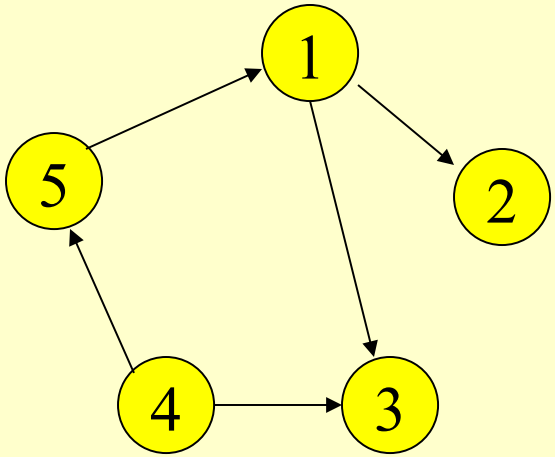
Questions?

Bao đóng truyền ứng
(Transitive Closure)

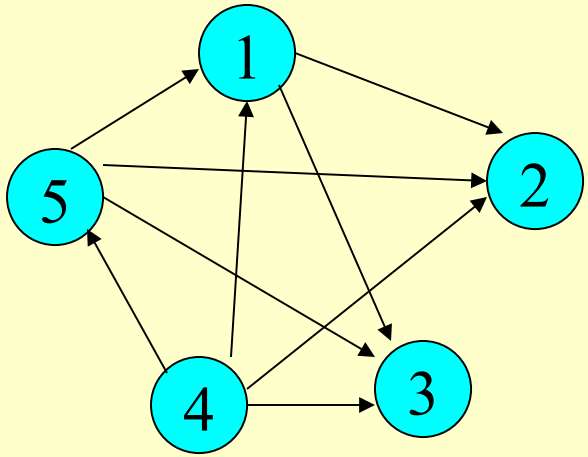
Bao đóng truyền ứng của đồ thị $G = (V, E)$ là $G^* = (V, E^*)$ sao cho

$(i, j) \in E^*$ iff có đường đi từ i đến j trên G.

G:



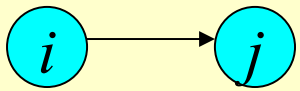
G^*:



Thuật toán Floyd-Warshall

- Ma trận xuất phát là ma trận kề

$$a(i,j) = \begin{cases} 1 & \text{nếu } i = j \text{ hoặc có cạnh nối 2 đỉnh } i \text{ và } j \\ 0 & \text{trái lại} \end{cases}$$

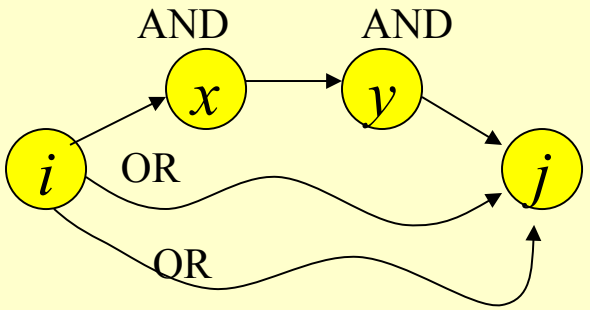


Nếu

- Thuật toán Floyd-Warshall thay

min ⟶ boolean OR

\+ ⟶ boolean AND



- Thời gian tính $\Theta(n^3)$

Questions?