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

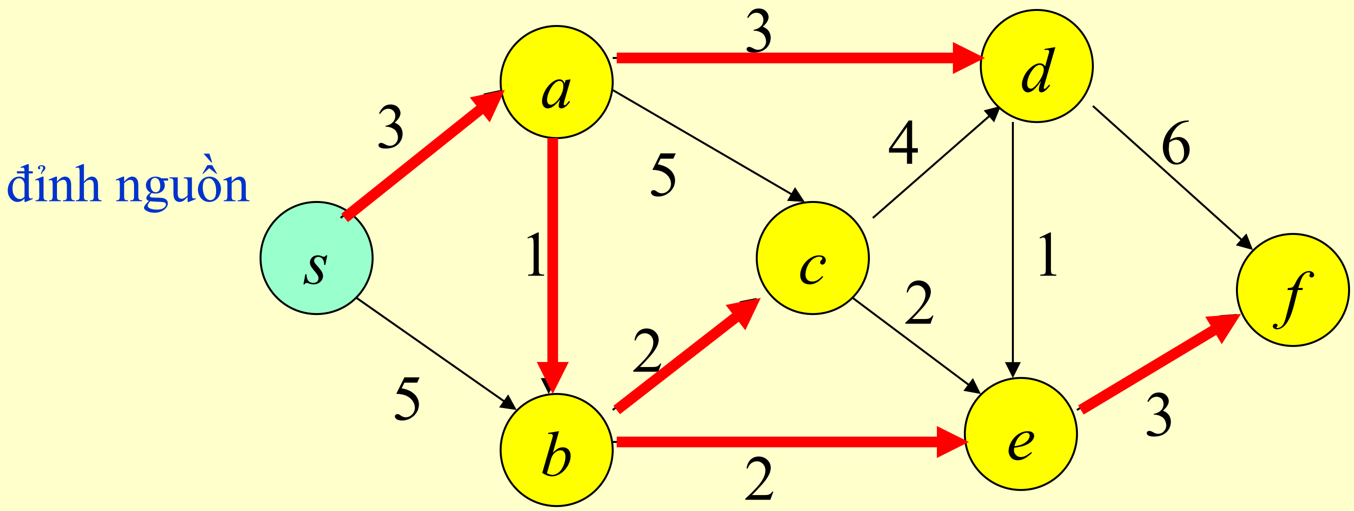
- Cho đơn đồ thị có hướng $G = (V,E)$ với hàm trọng số w: $E \to R$ ($w(e)$ được gọi là độ dài hay trọng số của cạnh e)
- Độ dài của đường đi $P = v_1 \to v_2 \to \ldots \to v_k$ là số

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

- 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.

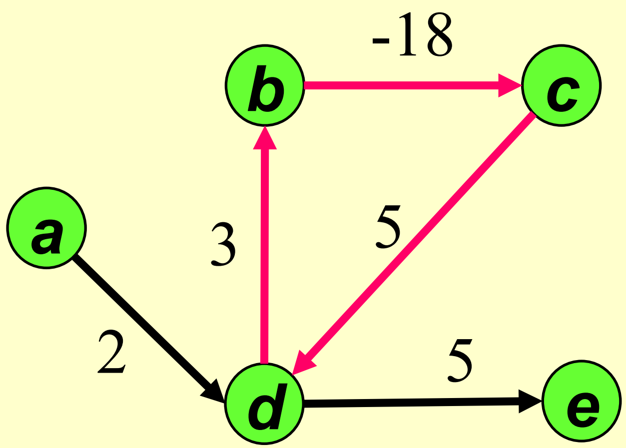
Nhận xét

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

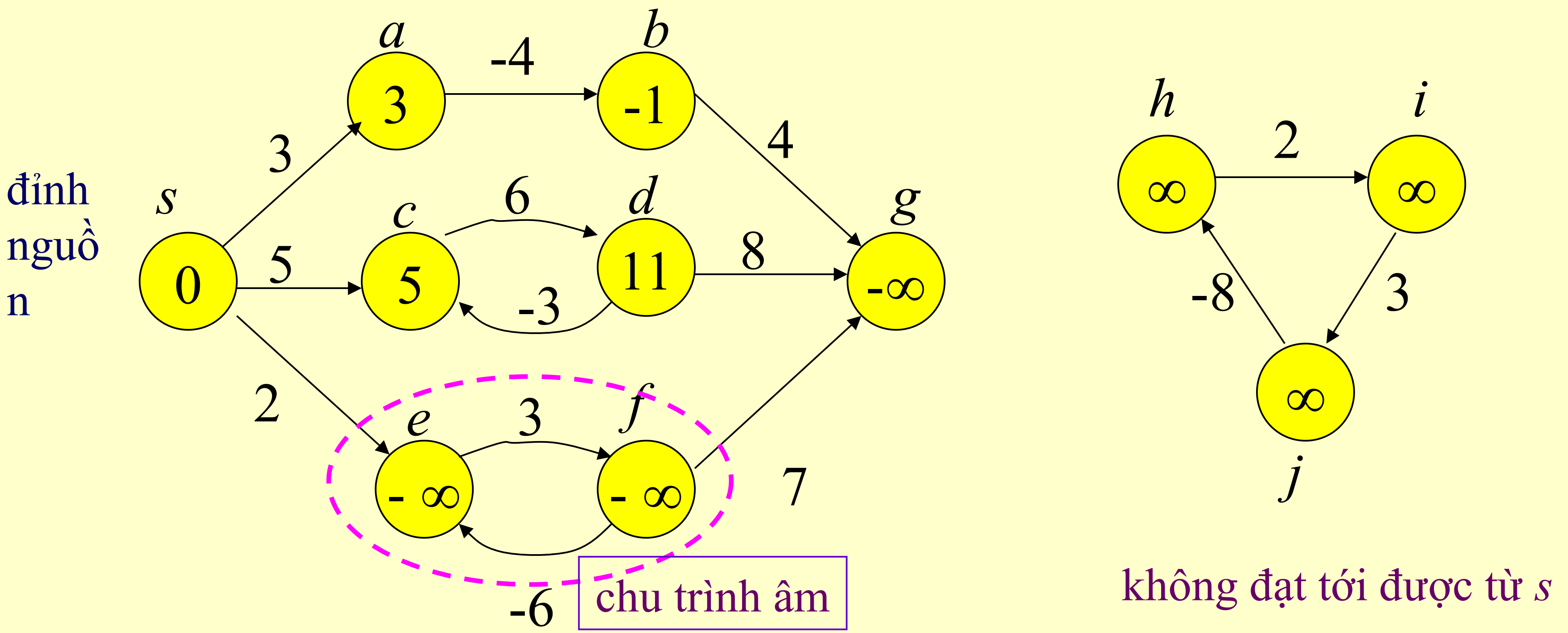
$w(P) = 7\text{-}10\sigma \rightarrow -\infty$, khi $\sigma \rightarrow +\infty$

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 – ∞.



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

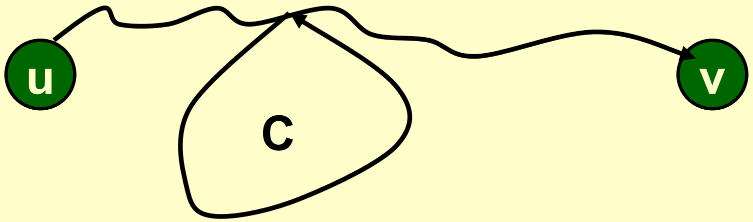
5.5. Đường đi ngắn nhất trong đồ thị không có chu trình

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.*

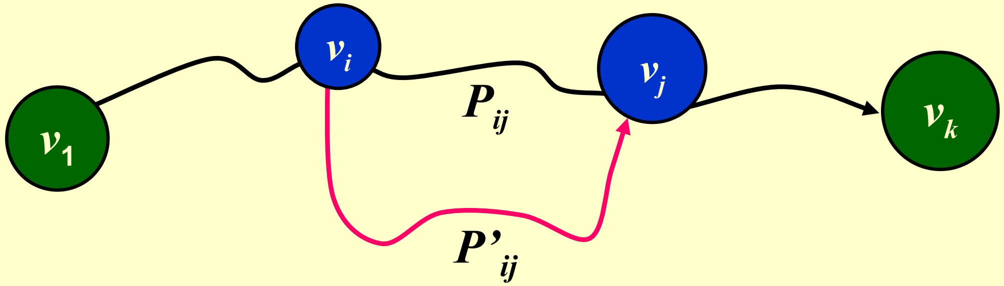
- *Như là hệ quả của tính chất 1*

Các tính chất của ĐĐNN

<u>Tính chất 3:</u> 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) .

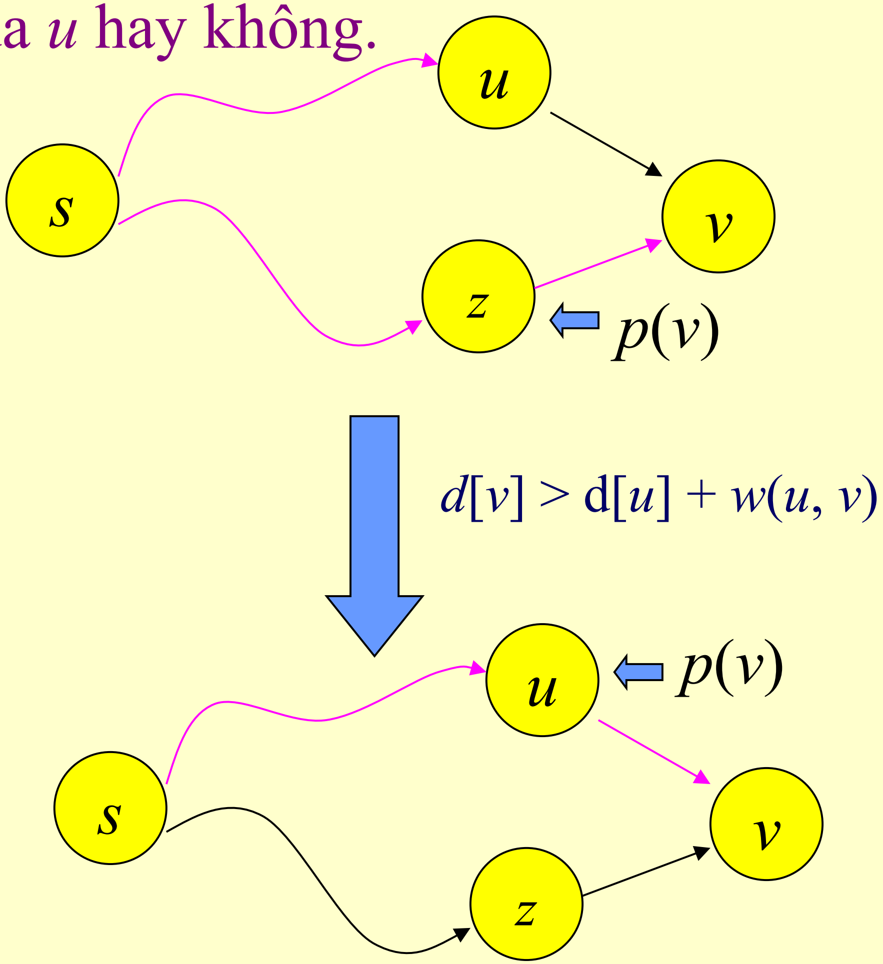
Khởi tạo (Initialization)

```
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 $d[v] \leftarrow d[u] + w(u, v)$
      $p[v] \leftarrow u$



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

- ViÖc cµi ®Æt c¸c thuËt to¸n ®îc thÓ hiÖn nhê ***thñ tôc g¸n nh·n***:
  - Mçi ®Ønh *v* sÏ cã nh·n gåm 2 thµnh phÇn (*d*[*v*], *p*[*v*]). Nh·n sÏ biÕn ®æi trong qu¸ tr×nh thùc hiÖn thuËt to¸n

- NhËn thÊy r»ng ®Ó tÝnh kho¶ng c¸ch tõ *s* ®Õn *t*, ë ®©y, ta ph¶i tÝnh kho¶ng c¸ch tõ *s* ®Õn tÊt c¶ c¸c ®Ønh cßn l¹i cña ®å thÞ.

- HiÖn nay vÉn cha biÕt thuËt to¸n nµo cho phÐp t×m ®®nn nhÊt gi÷a hai ®Ønh lµm viÖc thùc sù hiÖu qu¶ h¬n nh÷ng thuËt to¸n t×m ®®nn tõ mét ®Ønh ®Õn tÊt c¶ c¸c ®Ønh cßn l¹i.

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

- ThuËt to¸n Ford - Bellman t×m ®êng ®i ng¾n nhÊt tõ ®Ønh *s* ®Õn tÊt c¶ c¸c ®Ønh cßn l¹i cña ®å thÞ.
- ThuËt to¸n lµm viÖc trong trêng hîp träng sè cña c¸c cung lµ tuú ý.
- Gi¶ thiÕt r»ng trong ®å thÞ kh«ng cã chu tr×nh ©m.
- **§Çu vµo:** *§å thÞ G=*(*V,E*) *víi n ®Ønh x¸c ®Þnh bëi ma trËn träng sè w*[*u,v*]*, u,v* $\in$ *V, ®Ønh nguån s* $\in$ *V;*
- **§Çu ra:** *Víi mçi v* $\in$ *V*
  - *d*[*v*] = δ(*s, v*);
  - *p*[*v*] - *®Ønh ®i tríc v trong ®®nn tõ s ®Õn v.*

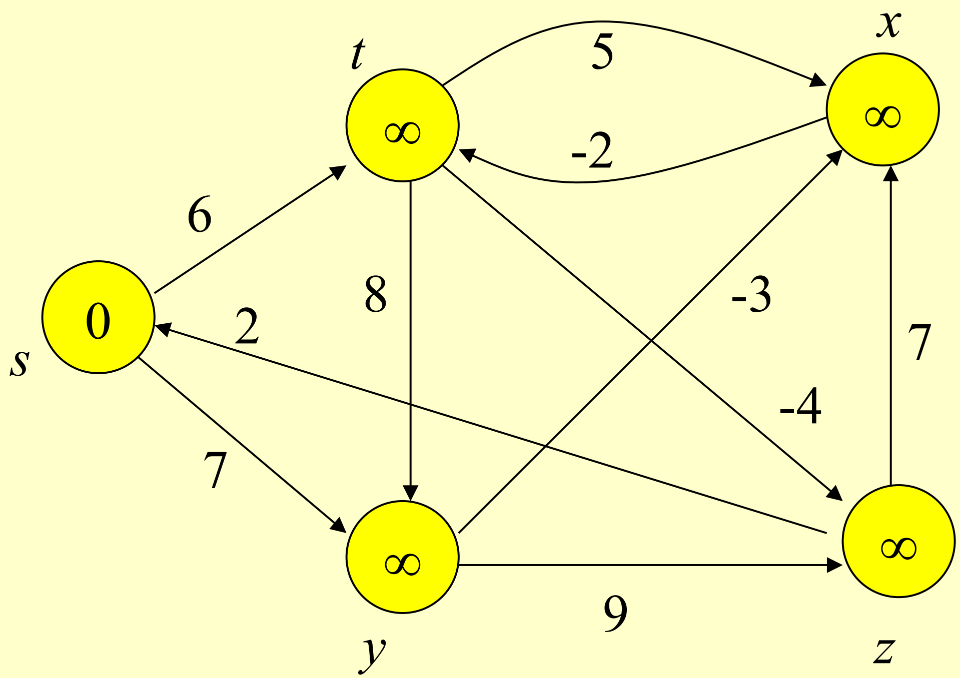
Mô tả thuật toán

```
procedure  Ford_Bellman;
begin
    for   v ∈ V  do  begin  (*   Khëi t¹o  *)
          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

- TÝnh ®óng ®¾n cña thuËt to¸n cã thÓ chøng minh trªn c¬ së nguyªn lý tèi u cña quy ho¹ch ®éng.
- §é phøc t¹p tÝnh to¸n cña thuËt to¸n lµ $O(n^3)$.
- Cã thÓ chÊm døt vßng lÆp theo *k* khi ph¸t hiÖn trong qu¸ tr×nh thùc hiÖn hai vßng lÆp trong kh«ng cã biÕn *d*[*v*] nµo bÞ ®æi gi¸ trÞ. ViÖc nµy cã thÓ x¶y ra ®èi víi $k < n$-2, vµ ®iÒu ®ã lµm t¨ng hiÖu qu¶ cña thuËt to¸n trong viÖc gi¶i c¸c bµi to¸n thùc tÕ.

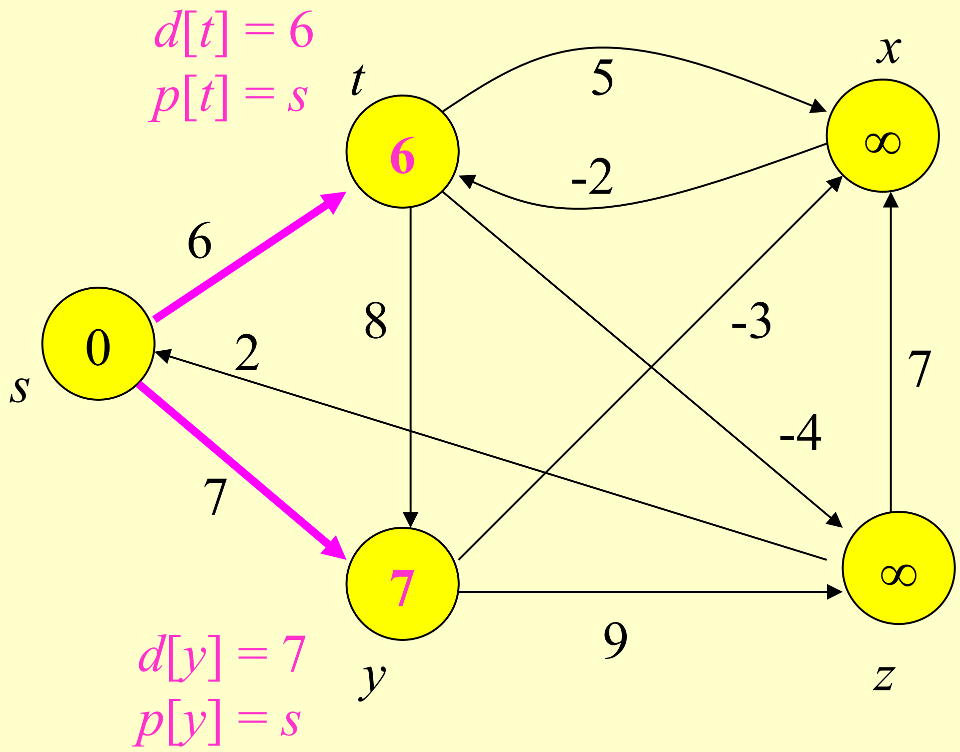
Ví dụ



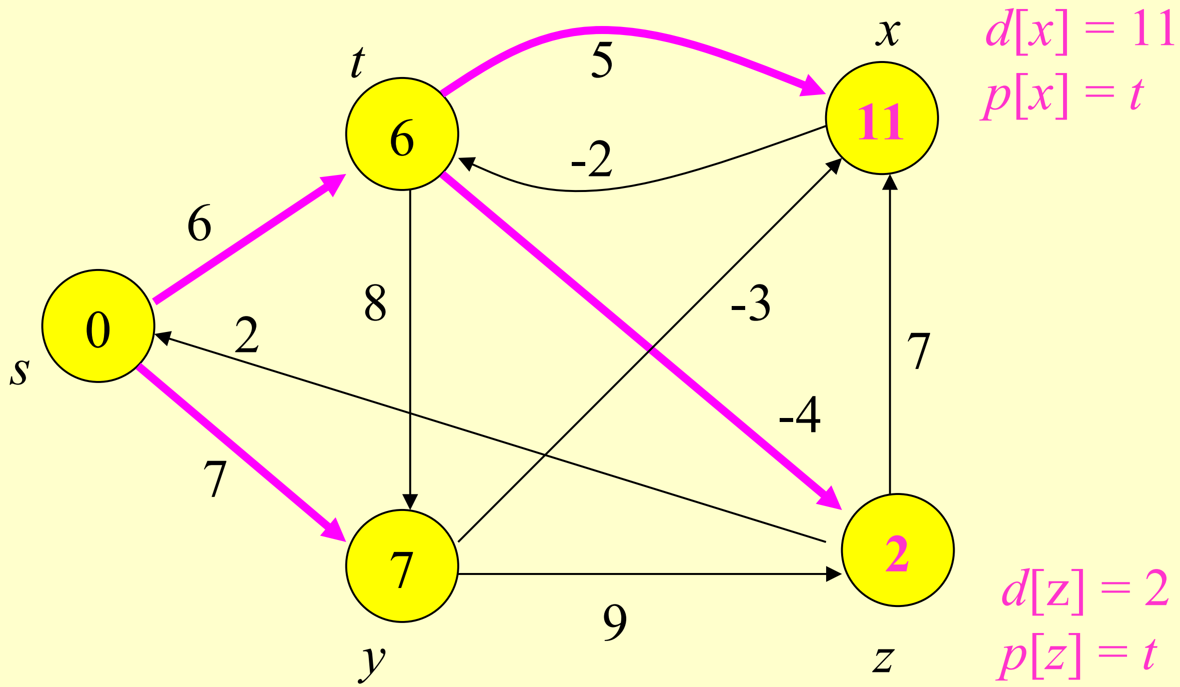
Source: *s*

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

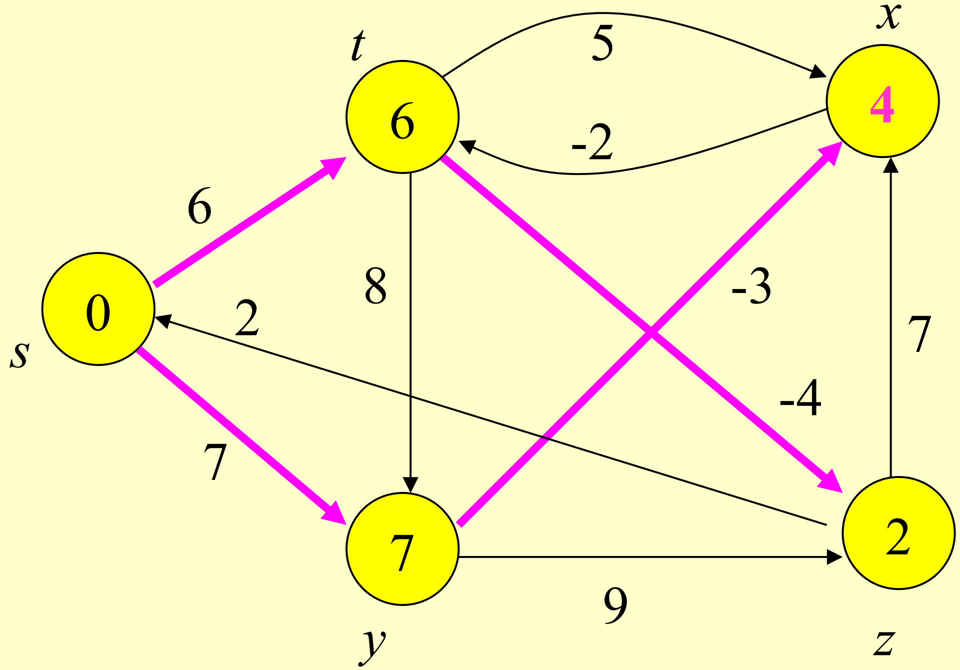


Lần 2



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

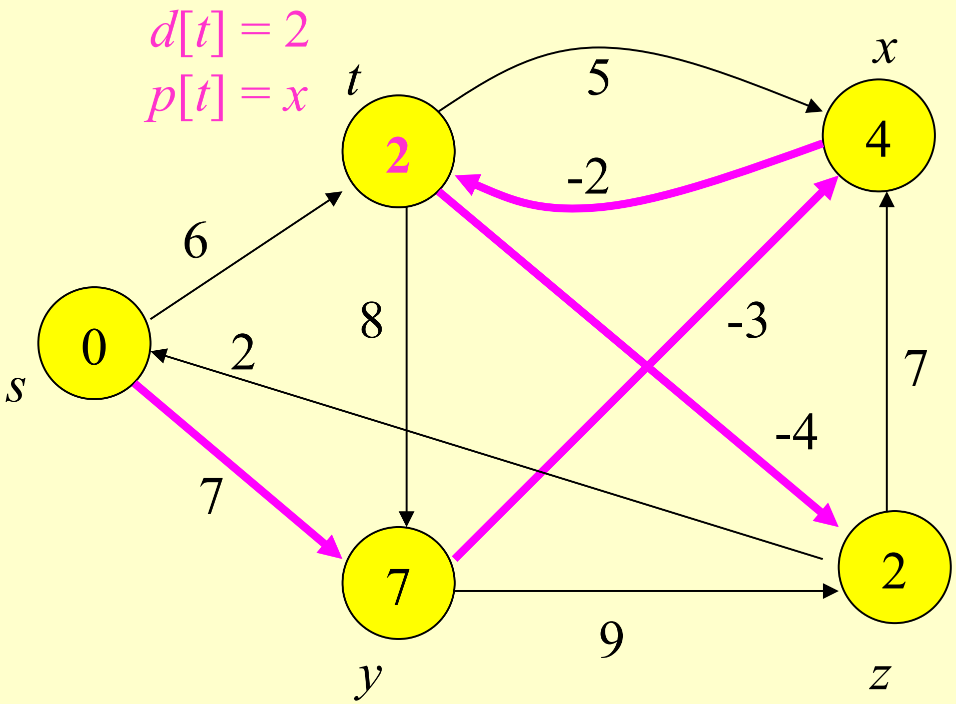
Lần 2 (tiếp)



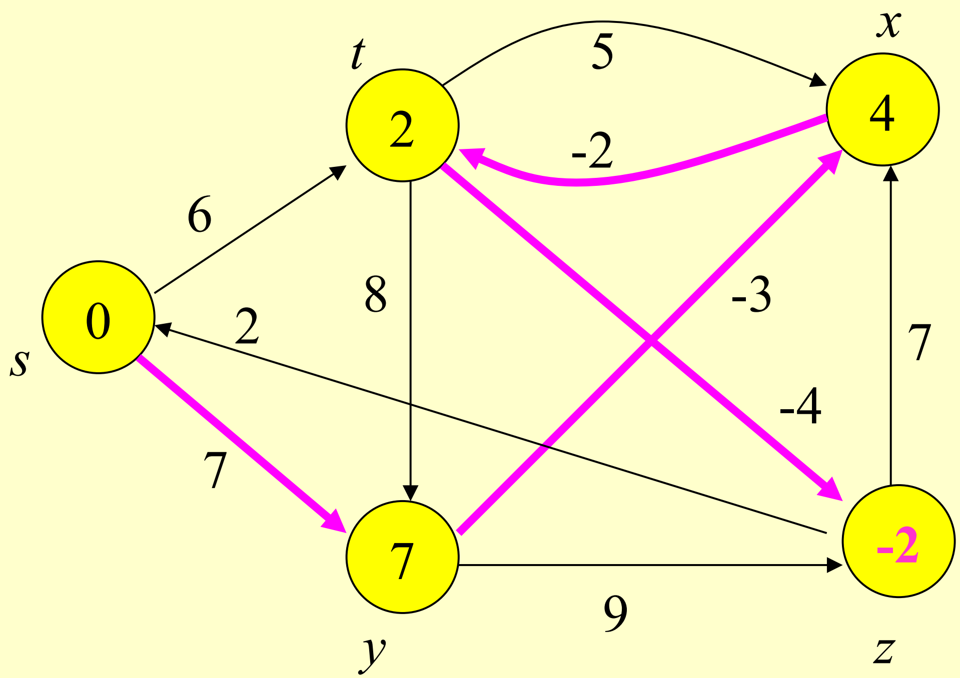
$d[x] = 4$
$p[x] = y$

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

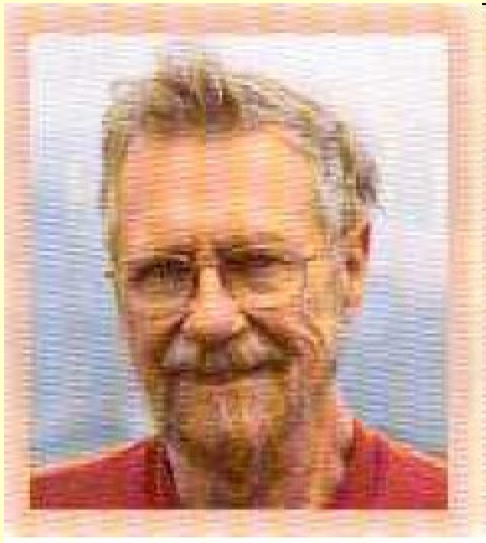
- §èi víi ®å thÞ tha tèt h¬n lµ sö dông danh s¸ch kÒ Ke-(*v*), *v* ∈ *V*, ®Ó biÓu diÔn ®å thÞ, khi ®ã vßng lÆp theo *u* cÇn viÕt l¹i díi d¹ng

```
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;
```

- ThuËt to¸n cã ®é phøc t¹p *O*(*n.m*).

Nội dung

Thuật toán Dijkstra



- Trong trêng hîp träng sè trªn c¸c cung lµ kh«ng ©m, thuËt to¸n do Dijkstra ®Ò nghÞ h÷u hiÖu h¬n rÊt nhiÒu so víi thuËt to¸n Ford-Bellman.
- ThuËt to¸n ®îc x©y dùng dùa trªn thủ tục gán nhãn. Thoạt tiên nh·n của các đỉnh là t¹m thêi. ë mçi mét bíc lÆp cã mét nh·n t¹m thêi trë thµnh nh·n cè ®Þnh. NÕu nh·n cña mét ®Ønh *u* trë thµnh cè ®Þnh th× *d*[*u*] sÏ cho ta ®é dµi cña ®®nn tõ ®Ønh *s* ®Õn *u*. Thuật toán kết thúc khi nhãn của tất cả các đỉnh trở thành cố định.

Edsger W.Dijkstra

(1930-2002)

Thuật toán Dijkstra

- ***§Çu vµo:*** *§å thÞ cã híng G=(V,E) víi n ®Ønh,*

  s $\in$ *V lµ ®Ønh xuÊt ph¸t,*

  w[u,v], u,v $\in$ *V - ma trËn träng sè;*
- ***Gi¶ thiÕt:*** ***w[u,v]*** $\geq$ ***0, u, v*** $\in$ ***V.***
- **§Çu ra:** *Víi mçi v* $\in$ *V*
  - *d*[*v*] = δ(*s*, *v*);
  - *p*[*v*] - *®Ønh ®i tríc v trong ®®nn tõ s ®Õn v.*

Thuật toán Dijkstra

Tập S: Chỉ cần cho chứng minh định lý

```
procedure Dijkstra;
begin
   for  v ∈ V  do  begin      (*   Khëi t¹o  *)
                    d[v] := w[s,v] ;  p[v]:=s;
                 end;
   d[s] := 0;   S := {s};                    (*  S – tËp ®Ønh cã nh·n cè ®Þnh *)
   T := V \ {s};                            (*   T lµ tËp c¸c ®Ønh cã nh·n t¹m thêi *)
   while  T  ≠ ∅  do                          (*     Bíc lÆp   *)
   begin
      T×m ®Ønh  u ∈ T tho¶ m·n  d[u] = min{ d[z] :  z  ∈  T};
      T :=  T \ {u};  S:= S  ∪ {u};      (*  Cè ®Þnh nh·n cña ®Ønh u *)
      for   v ∈ T   do                       (*  G¸n nh·n l¹i cho c¸c ®Ønh 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;
```

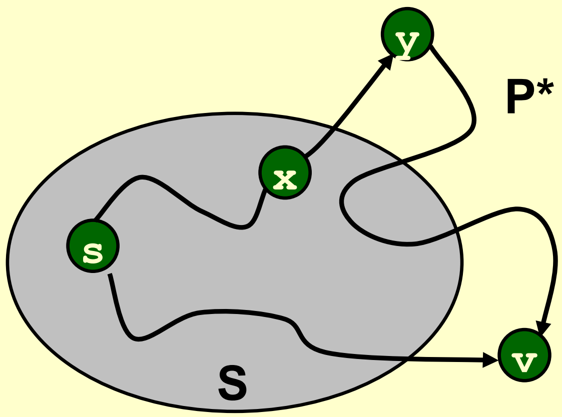
Thuật toán Dijkstra

- **Chó ý:** NÕu chØ cÇn t×m ®êng ®i ng¾n nhÊt tõ *s* ®ªn *t* th× cã thÓ chÊm døt thuËt to¸n khi ®Ønh t trë thµnh cã nh·n cè ®Þnh.

- **§Þnh lý 1.** *ThuËt to¸n Dijkstra t×m ®îc ®êng ®i ng¾n nhÊt tõ ®Ønh s ®Õn tÊt c¶ c¸c ®Ønh cßn l¹i trªn ®å thÞ sau thêi 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 $v \in S$, $d(v) = \delta(s, v)$.

- 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 $d(v) > \delta(s, v)$ giả thiết

$= \delta(s, x) + w(x, y) + \delta(y, v)$ tính chất 3

$\geq \delta(s, x) + w(x, y)$ $\delta(y, v)$ là không âm

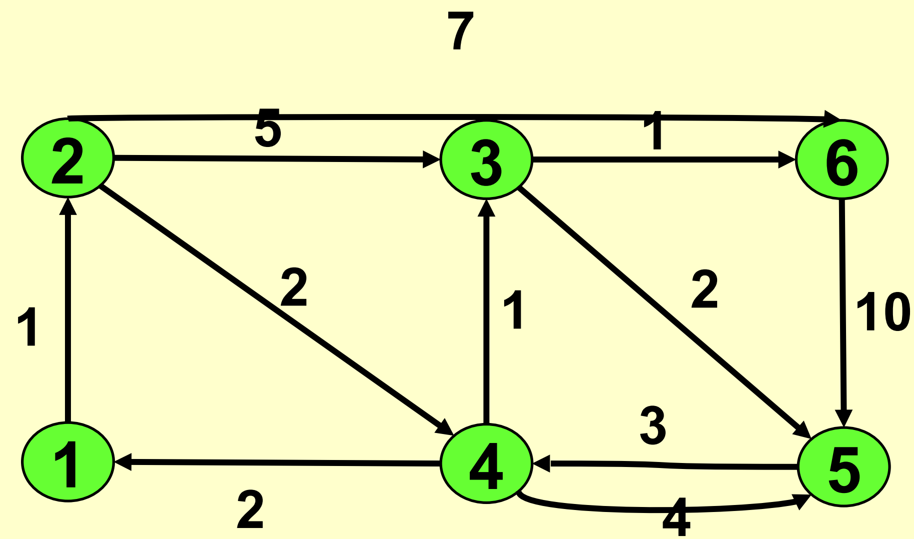
$= d(x) + w(x, y)$ giả thiết quy nạp

$\geq d(y)$ 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

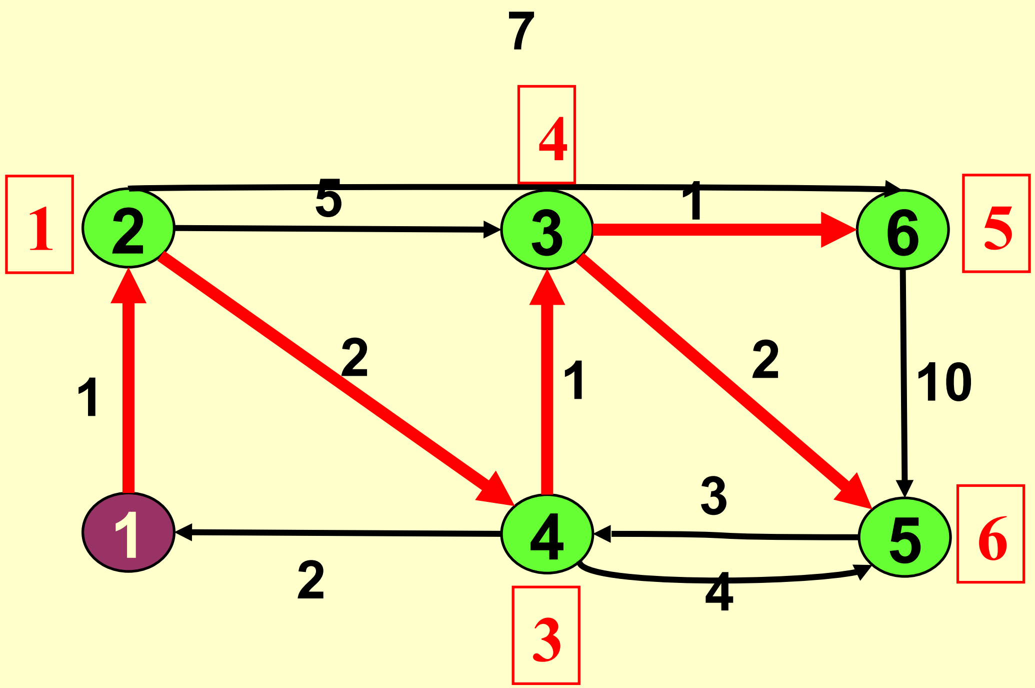


	Đỉnh 1	Đỉnh 2	Đỉnh 3	Đỉnh 4	Đỉnh 5	Đỉnh 6
Khởi tạo	[0, 1]	[1, 1]*	[∞, 1]	[∞, 1]	[∞, 1]	[∞, 1]
1	-	-	[6, 2]	[3, 2]*	[∞, 1]	[8, 2]
2	-	-	[4, 4]*	-	[7, 4]	[8, 2]
3	-	-	-	-	[6, 3]	[5, 3]*
4	-	-	-	-	[6, 3]*	-
5	-	-	-	-	-	-

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

Đường đi trong đồ thị không có chu trình

Shortest Paths In Directed Acyclic Graphs

Đường đi trong đồ thị không có chu trình

- Mét trêng hîp riªng cña bµi to¸n ®êng ®i ng¾n nhÊt gi¶i ®îc nhê thuËt to¸n víi ®é phøc t¹p tÝnh to¸n $O(n^2)$, ®ã lµ bµi to¸n trªn ®å thÞ kh«ng cã chu tr×nh (cßn träng sè trªn c¸c cung cã thÓ lµ c¸c sè thùc tuú ý). KÕt qu¶ sau ®©y lµ c¬ së ®Ó x©y dùng thuËt to¸n nãi trªn:

- **§Þnh lý 2.** *Gi¶ sö G lµ ®å thÞ kh«ng cã chu tr×nh. Khi ®ã c¸c ®Ønh cña nã cã thÓ ®¸nh sè sao cho mçi cung cña ®å thÞ chØ híng tõ ®Ønh cã chØ sè nhá h¬n ®Õn ®Ønh cã chØ sè lín h¬n, nghÜa lµ mçi cung cña nã cã thÓ biÓu diÔn díi d¹ng* $(v[i], v[j])$, *trong ®ã* $i < j$.

Thuật toán đánh số đỉnh

- Tríc hÕt nhËn thÊy r»ng: *Trong ®å thÞ kh«ng cã chu tr×nh bao giê còng t×m ®îc ®Ønh cã b¸n bËc vµo b»ng 0*. Thùc vËy, b¾t ®Çu tõ ®Ønh v_1 nÕu cã cung ®i vµo nã tõ v_2 th× ta l¹i chuyÓn sang xÐt ®Ønh v_2. NÕu cã cung tõ v_3 ®i vµo v_2, th× ta l¹i chuyÓn sang xÐt v_3, ... Do ®å thÞ lµ kh«ng cã chu tr×nh nªn sau mét sè h÷u h¹n lÇn chuyÓn nh vËy ta ph¶i ®i ®Õn ®Ønh kh«ng cã cung ®i vµo.
- ThuËt to¸n ®îc x©y dùng dùa trªn ý tëng rÊt ®¬n gi¶n sau: Tho¹t tiªn, t×m c¸c ®Ønh cã b¸n bËc vµo b»ng 0. Râ rµng ta cã thÓ ®¸nh sè chóng theo mét thø tù tuú ý b¾t ®Çu tõ 1. TiÕp theo, lo¹i bá khái ®å thÞ nh÷ng ®Ønh ®· ®îc ®¸nh sè cïng c¸c cung ®i ra khái chóng, ta thu ®îc ®å thÞ míi còng kh«ng cã chu tr×nh, vµ thñ tôc ®îc lÆp l¹i víi ®å thÞ míi nµy. Qu¸ tr×nh ®ã sÏ ®îc tiÕp tôc cho ®Õn khi tÊt c¶ c¸c ®Ønh cña ®å thÞ ®îc ®¸nh sè.

Thuật toán đánh số đỉnh

- **§Çu vµo:** *§å thÞ cã híng G=(V,E) víi n ®Ønh kh«ng chøa chu tr×nh ®îc cho bëi danh s¸ch kÒ Ke(v), v ∈ V.*
- **§Çu ra:** *Víi mçi ®Ønh v ∈ V chØ sè NR [v] tho¶ m·n: Víi mäi cung (u, v) cña ®å thÞ ta ®Òu cã NR[u] < NR[v].*

Thuật toán đánh số đỉnh

```
procedure  Numbering;
begin
    for   v ∈  V  do   Vao[v] := 0;
    for   u ∈  V  do       (*  TÝnh Vao[v] = b¸n bËc vµo cña 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

- Râ rµng trong bíc khëi t¹o ta ph¶i duyÖt qua tÊt c¶ c¸c cung cña ®å thÞ khi tÝnh b¸n bËc vµo cña c¸c ®Ønh, v× vËy ë ®ã ta tèn cì *O*(*m*) phÐp to¸n, trong ®ã *m* lµ sè cung cña ®å thÞ. TiÕp theo, mçi lÇn ®¸nh sè mét ®Ønh, ®Ó thùc hiÖn viÖc lo¹i bá ®Ønh ®· ®¸nh sè cïng víi c¸c cung ®i ra khái nã, chóng ta l¹i duyÖt qua tÊt c¶ c¸c cung nµy. Suy ra ®Ó ®¸nh sè tÊt c¶ c¸c ®Ønh cña ®å thÞ chóng ta sÏ ph¶i duyÖt qua tÊt c¶ c¸c cung cña ®å thÞ mét lÇn n÷a.
- VËy ®é phøc t¹p cña thuËt to¸n lµ *O*(*m*).

Thuật toán tìm đđnn trên đồ thị không có chu trình

- Do cã thuËt to¸n ®¸nh sè trªn, nªn khi xÐt ®å thÞ kh«ng cã chu tr×nh ta cã thÓ gi¶ thiÕt lµ c¸c ®Ønh cña nã ®îc ®¸nh sè sao cho mçi cung chØ ®i tõ ®Ønh cã chØ sè nhá ®Õn ®Ønh cã chØ sè lín h¬n.

- ***ThuËt to¸n t×m ®êng ®i ng¾n nhÊt tõ ®Ønh nguån v*[1] *®Õn tÊt c¶ c¸c ®Ønh cßn l¹i trªn®å thÞ kh«ng cã chu tr×nh***
- **§Çu vµo:** *§å thÞ G*=(*V, E*)*, trong ®ã V*={ *v*[1]*, v*[2]*, ... , v*[*n*] }.
  §èi víi mçi cung (*v*[*i*], *v*[*j*]) ∈ *E, ta cã i < j.*
  §å thÞ ®îc cho bëi danh s¸ch kÒ Ke(*v*) *, v* ∈ *V.*
- **§Çu ra:** *Kho¶ng c¸ch tõ v*[1] *®Õn tÊt c¶ c¸c ®Ønh cßn l¹i ®îc ghi trong m¶ng d*[*v*[*i*]], *i* = 2, 3, ..., *n*

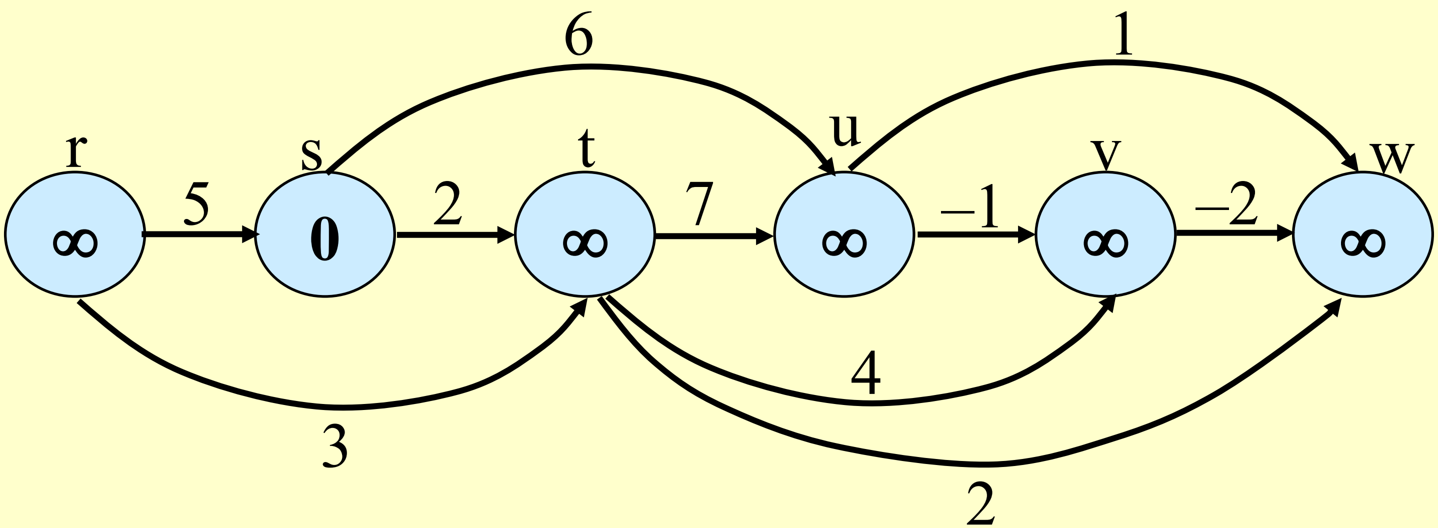
Thuật toán tìm đđnn trên đồ thị không có chu trình

```
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;
```

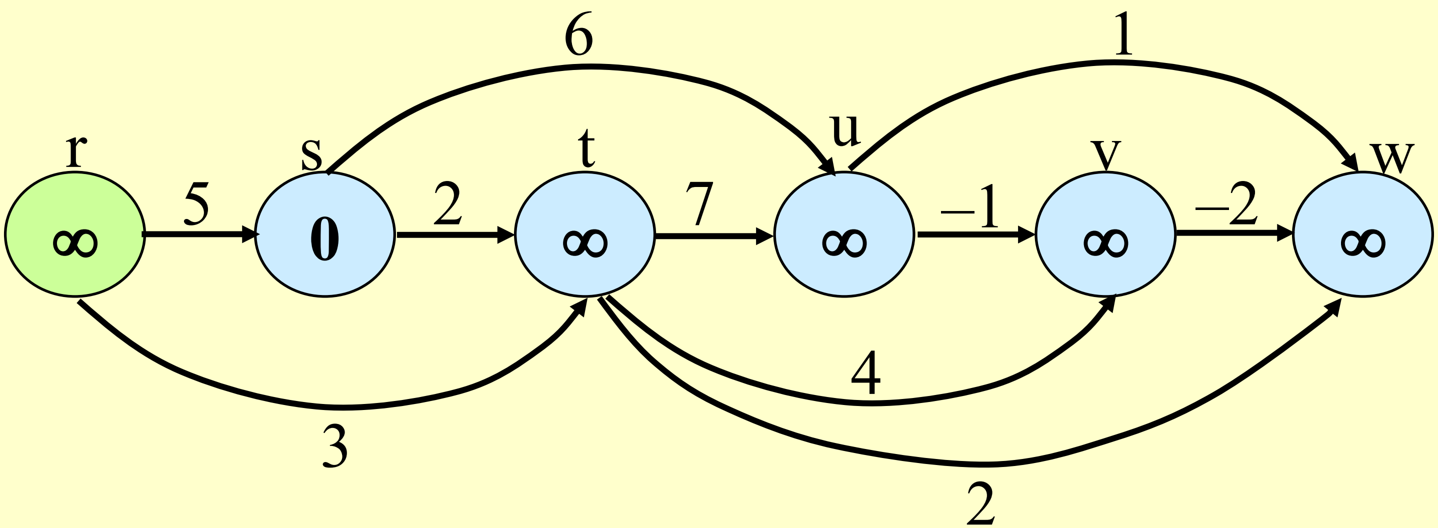
- **§é phøc t¹p tÝnh to¸n cña thuËt to¸n lµ O(*m*), do mçi cung cña ®å thÞ ph¶i xÐt qua ®óng mét lÇn.**

Ví dụ

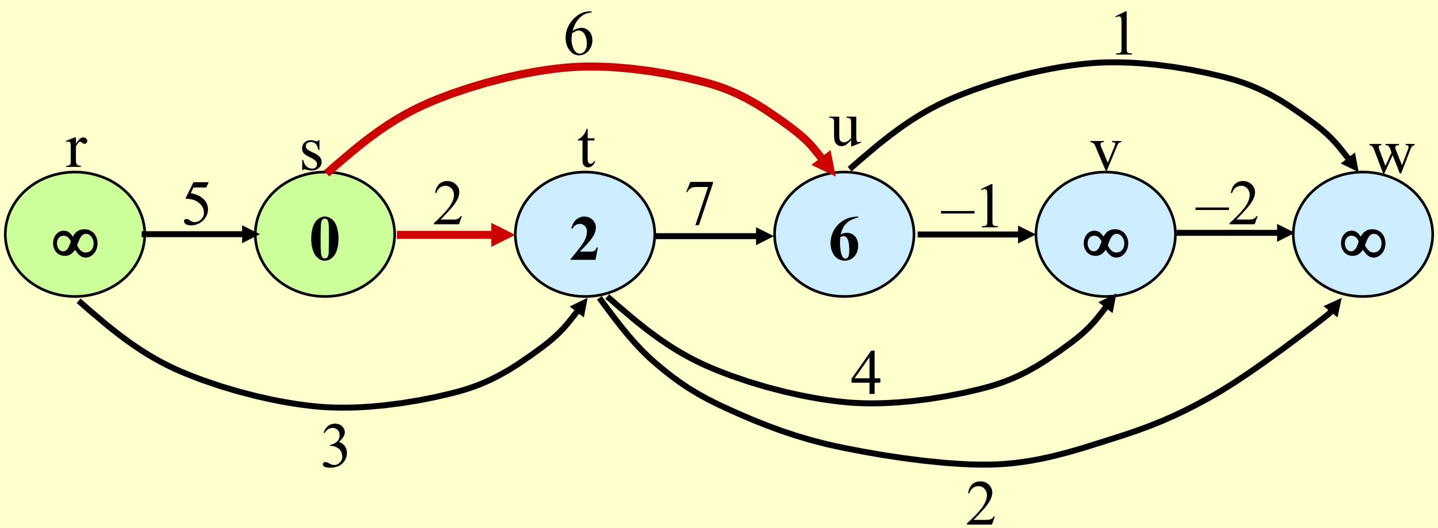
Cần tìm đường đi ngắn nhất từ ***s*** đến tất cả các đỉnh đạt đến được từ nó



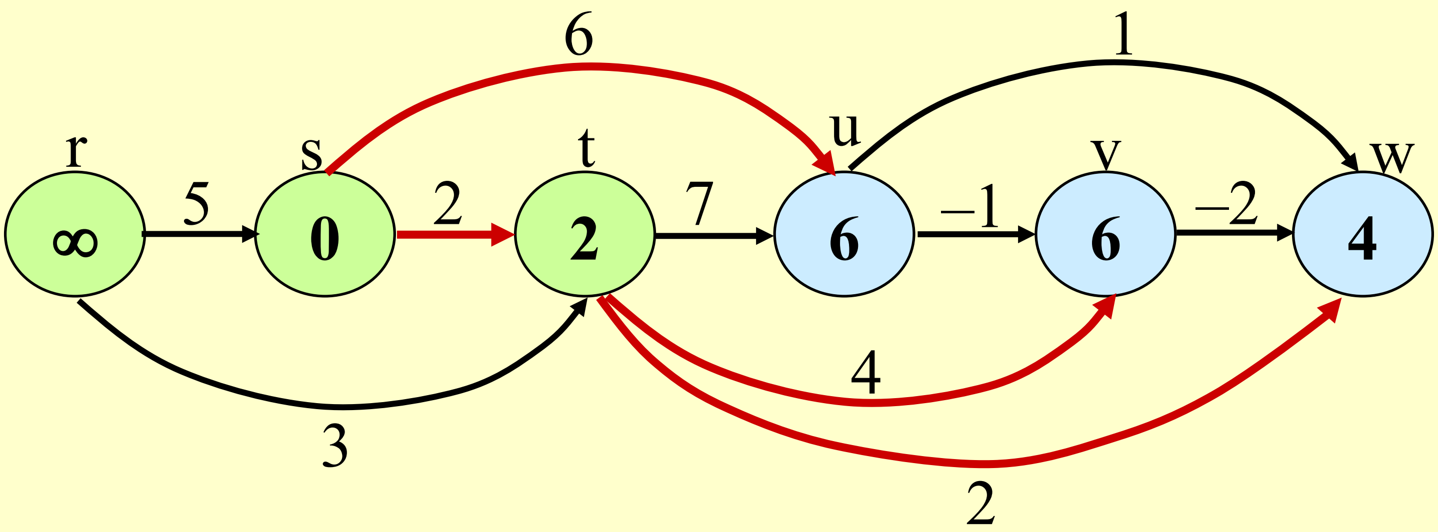
Ví dụ



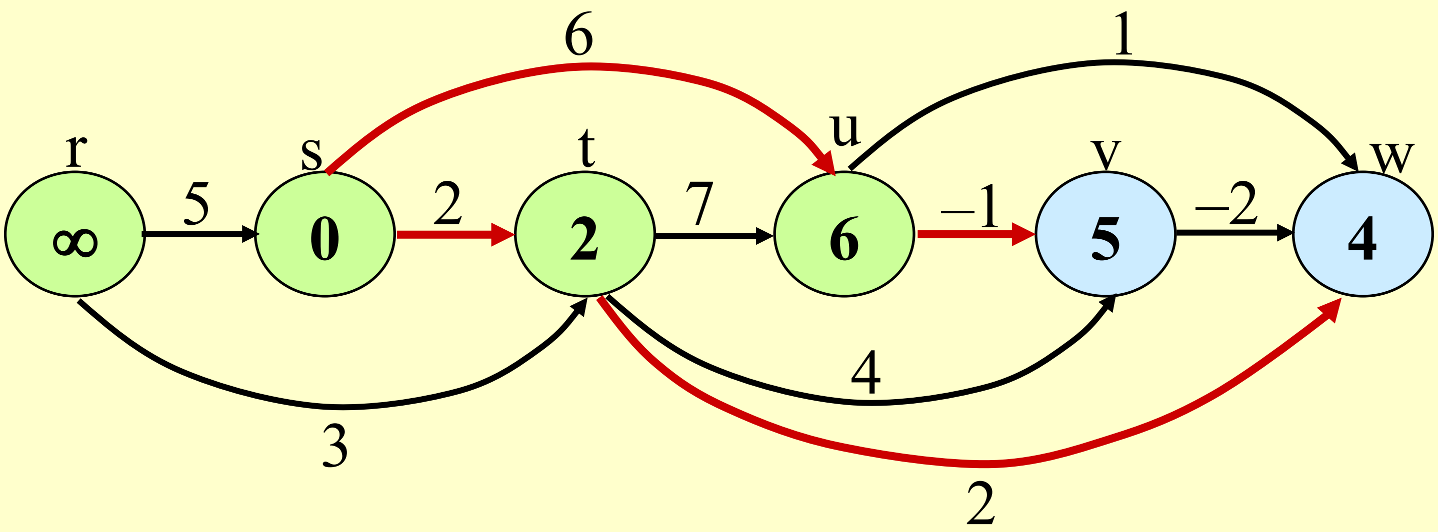
Ví dụ



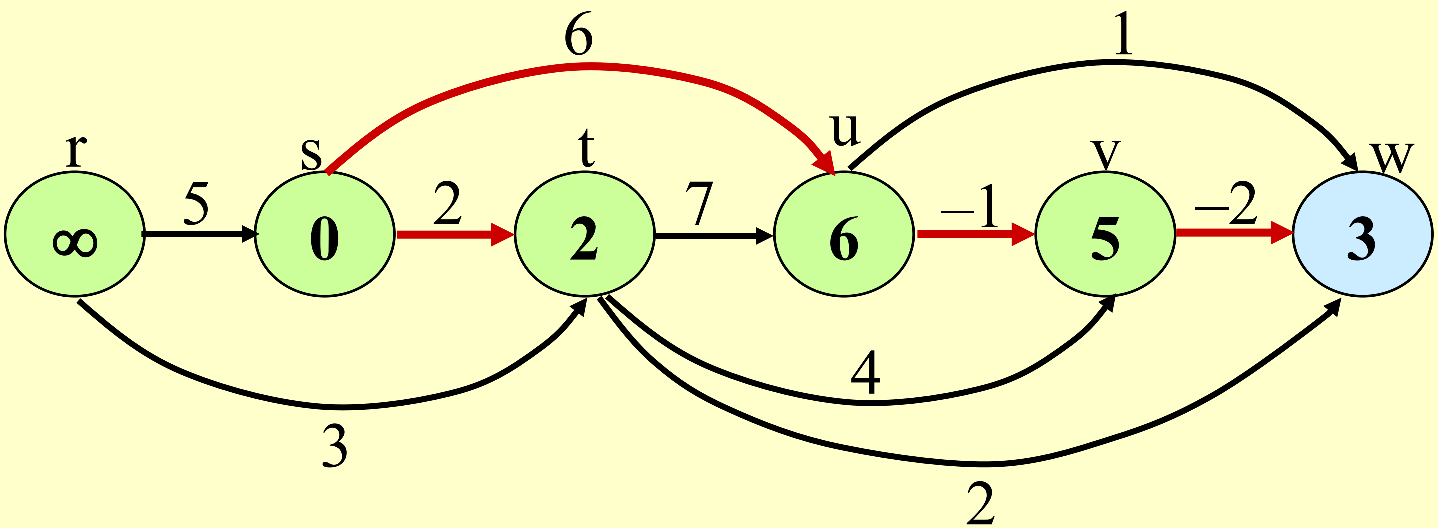
Ví dụ



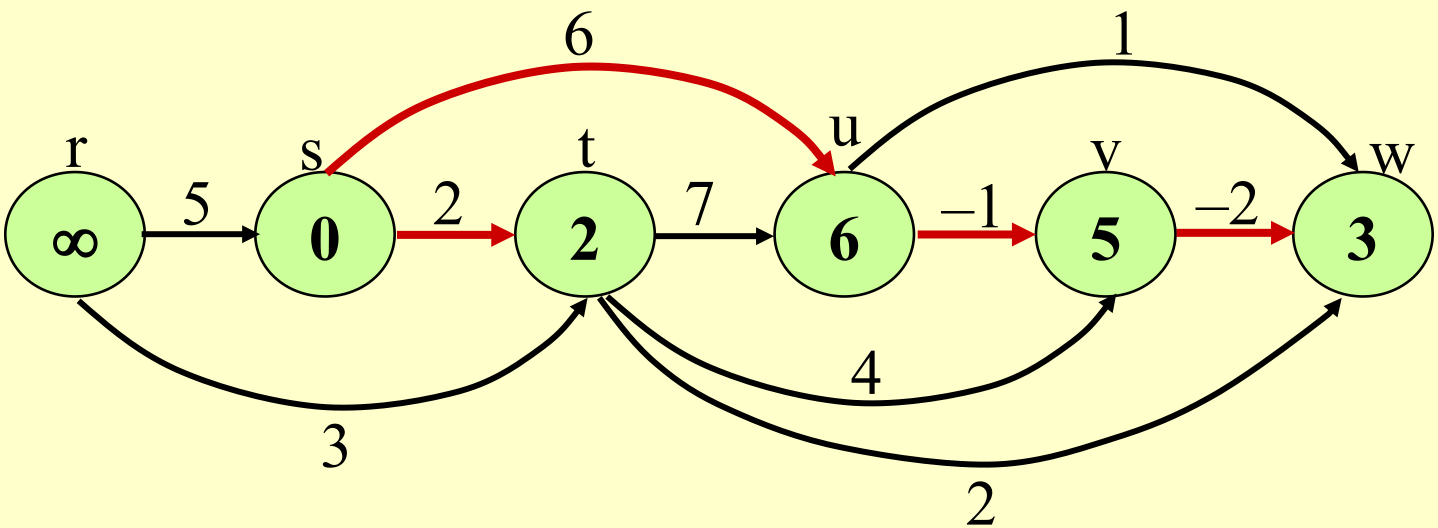
Ví dụ



Ví dụ



Ví dụ



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

Ứng dụng: PERT

- X©y dùng ph¬ng ph¸p gi¶i bµi to¸n ®iÒu khiÓn viÖc thùc hiÖn nh÷ng dù ¸n lín, gäi t¾t lµ PERT (*Project Evaluation and Review Technique*) hay CDM (*Critical path Method*).

- ViÖc thi c«ng mét c«ng tr×nh lín ®îc chia ra lµm *n* c«ng ®o¹n, ®¸nh sè tõ 1 ®Õn *n*. Cã mét sè c«ng ®o¹n mµ viÖc thùc hiÖn nã chØ ®îc tiÕn hµnh sau khi mét sè c«ng ®o¹n nµo ®ã ®· hoµn thµnh. §èi víi mçi c«ng ®o¹n *i* biÕt *t*[*i*] lµ thêi gian cÇn thiÕt ®Ó hoµn thµnh nã (*i* = 1, 2,..., *n*).

Ứng dụng: PERT

- C¸c d÷ liÖu víi *n* = 8 ®îc cho trong b¶ng sau ®©y

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

- **Bµi to¸n PERT:** Gi¶ sö thêi ®iÓm b¾t ®Çu tiÕn hµnh thi c«ng c«ng tr×nh lµ 0. H·y t×m tiÕn ®é thi c«ng c«ng tr×nh (chØ râ mçi c«ng ®o¹n ph¶i ®îc b¾t ®Çu thc hiÖn vµo thêi ®iÓm nµo) ®Ó cho c«ng tr×nh ®îc hoµn thµnh xong trong thêi ®iÓm sím nhÊt cã thÓ ®îc.

- Ta cã thÓ x©y dùng ®å thÞ cã híng *n* ®Ønh biÓu diÔn rµng buéc vÒ tr×nh tù thùc hiÖc c¸c c«ng viÖc nh sau:
  - Mçi ®Ønh cña ®å thÞ t¬ng øng víi mét c«ng viÖc.
  - NÕu c«ng viÖc *i* ph¶i ®îc thùc hiÖn tríc c«ng ®o¹n *j* th× trªn ®å thÞ cã cung (*i,j*), träng sè trªn cung nµy ®îc g¸n b»ng *t*[*i*]

Thuật toán PERT

- Thªm vµo ®å thÞ 2 ®Ønh 0 vµ *n*+1 t¬ng øng víi hai sù kiÖn ®Æc biÖt:
  - ®Ønh sè 0 t¬ng øng víi c«ng ®o¹n *LÔ khëi c«ng*, nã ph¶i ®îc thùc hiÖn tríc tÊt c¶ c¸c c«ng ®o¹n kh¸c, vµ
  - ®Ønh *n*+1 t¬ng øng víi c«ng ®o¹n *C¾t b¨ng kh¸nh thµnh c«ng tr×nh,* nã ph¶i thùc hiÖn sau tÊt c¶ c¸c c«ng ®o¹n,
  - víi *t*[0] = *t*[*n*+1] = 0 (trªn thùc tÕ chØ cÇn nèi ®Ønh 0 víi tÊt c¶ c¸c ®Ønh cã b¸n bËc vµo b»ng 0 vµ nèi tÊt c¶ c¸c ®Ønh cã b¸n bËc ra b»ng 0 víi ®Ønh *n*+1).

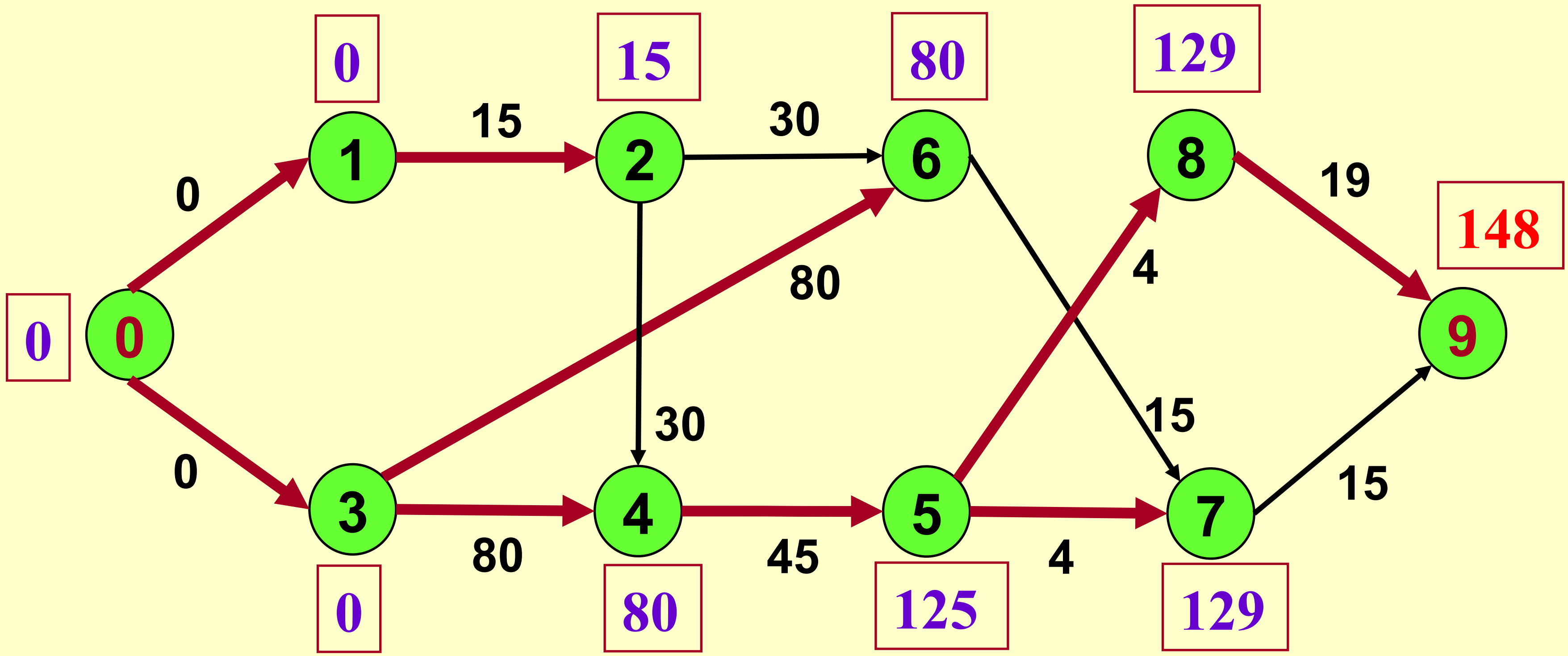
  Gäi ®å thÞ thu ®îc lµ *G*.
- Râ rµng bµi to¸n ®Æt ra dÉn vÒ bµi to¸n t×m ®êng ®i dµi nhÊt tõ ®Ønh 0 ®Õn tÊt c¶ c¸c ®Ønh cßn l¹i trªn ®å thÞ *G*.

Thuật toán PERT

- Do ®å thÞ *G* kh«ng chøa chu tr×nh, nªn ®Ó gi¶i bµi to¸n ®Æt ra cã thÓ ¸p dông thuËt to¸n Critical_Path trong ®ã chØ cÇn ®æi to¸n tö ***min*** thµnh to¸n tö ***max***.
- KÕt thóc thuËt to¸n, ta thu ®îc *d*[*v*] lµ ®é dµi ® êng ®i dµi nhÊt tõ ®Ønh 0 ®Õn ®Ønh *v*.
- Khi ®ã *d*[*v*] cho ta thêi ®iÓm sím nhÊt cã thÓ b¾t ®Çu thùc hiÖn c«ng ®o¹n *v*, nãi riªng *d*[*n*+1] lµ thêi ®iÓm sím nhÊt cã thÓ c¾t b¨ng kh¸nh thµnh, tøc lµ thêi ®iÓm sím nhÊt cã thÓ hoµn thµnh toµn bé c«ng tr×nh.

PERT: Ví dụ minh hoạ

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



Nội dung

ĐƯỜNG ĐI NGẮN NHẤT GIỮA MỌI CẶP ĐỈNH

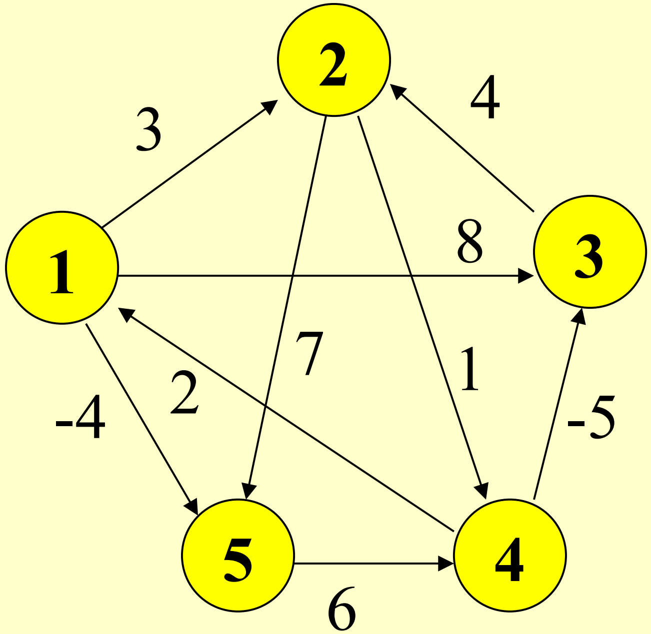
All-Pairs Shortest Paths

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



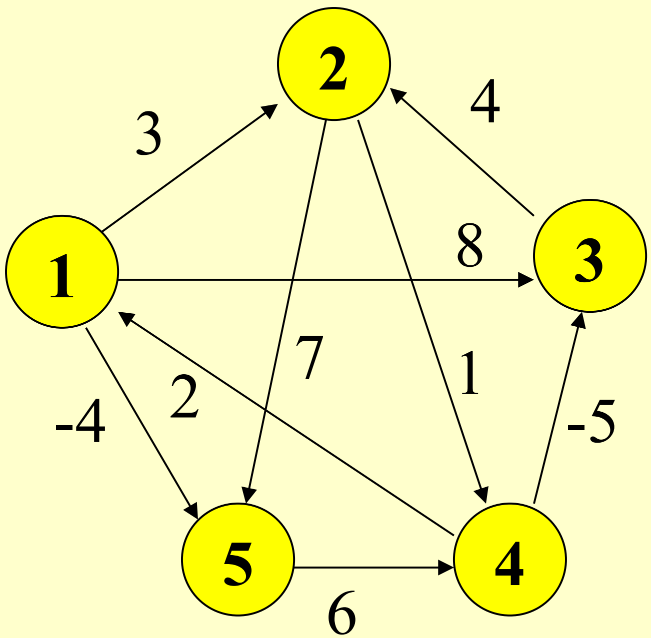
Đầu vào

$n \times n$ 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

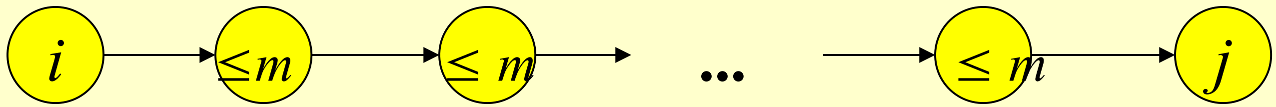
$$\begin{pmatrix} 0 & \mathbf{1} & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & \mathbf{-2} \\ \mathbf{8} & 5 & 1 & 6 & 0 \end{pmatrix}$$

4 - 1- 5

5 - 4 - 1

Thuật toán Floyd-Warshall

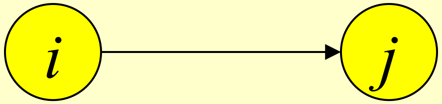
$d_{ij}^{(m)}$ = độ dài đường đi ngắn nhất từ *i* đến *j* sử dụng các đỉnh trung gian trong tập đỉnh { 1, 2, ..., *m* }.



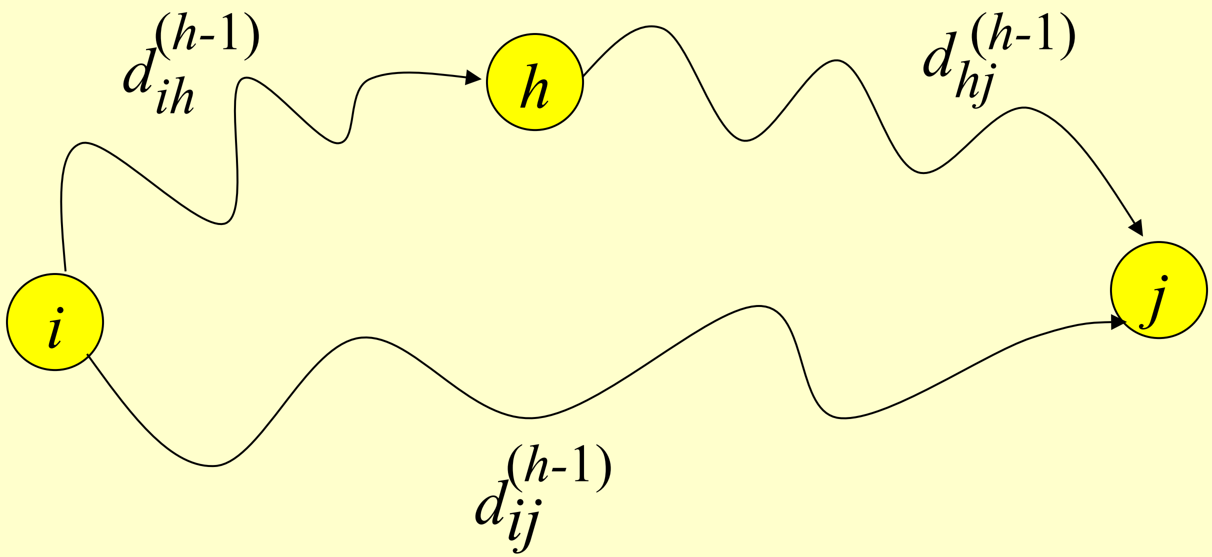
Khi đó độ dài đường đi ngắn nhất từ *i* đến *j* là $d_{ij}^{(n)}$

Công thức đệ qui tính $d^{(h)}$

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



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

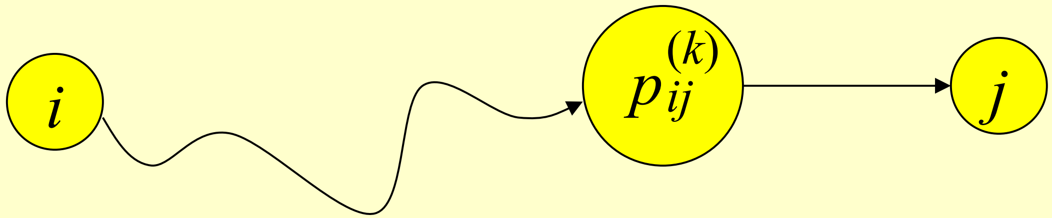
$d_{ij}^{(k)} \leftarrow \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$

return $D^{(n)}$

Thời gian tính **$\Theta(n^3)$** !

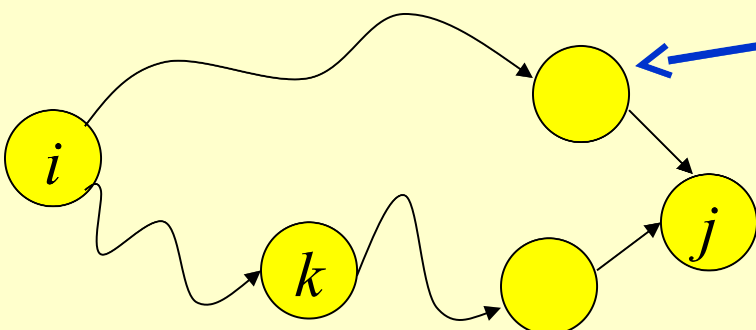
Xây dựng đường đi ngắn nhất

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



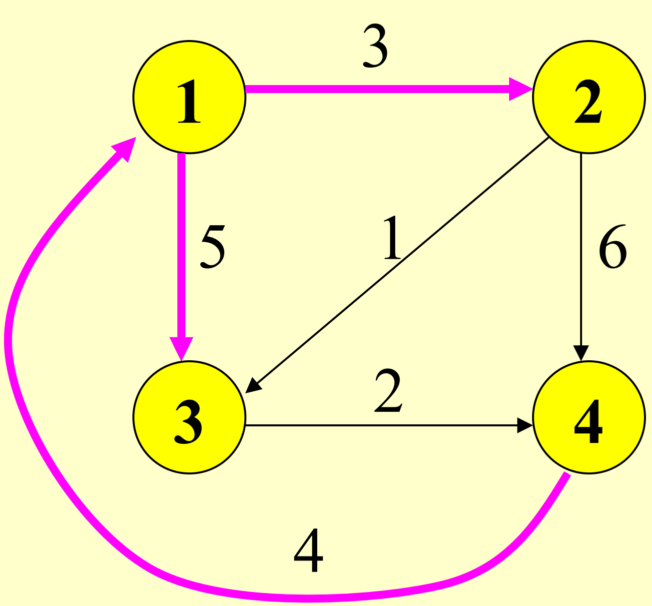
đường đi ngắn nhất từ *i* đến *j* chỉ qua các đỉnh trung gian trong $\{1, 2, \ldots, k\}$.

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



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

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:

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

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

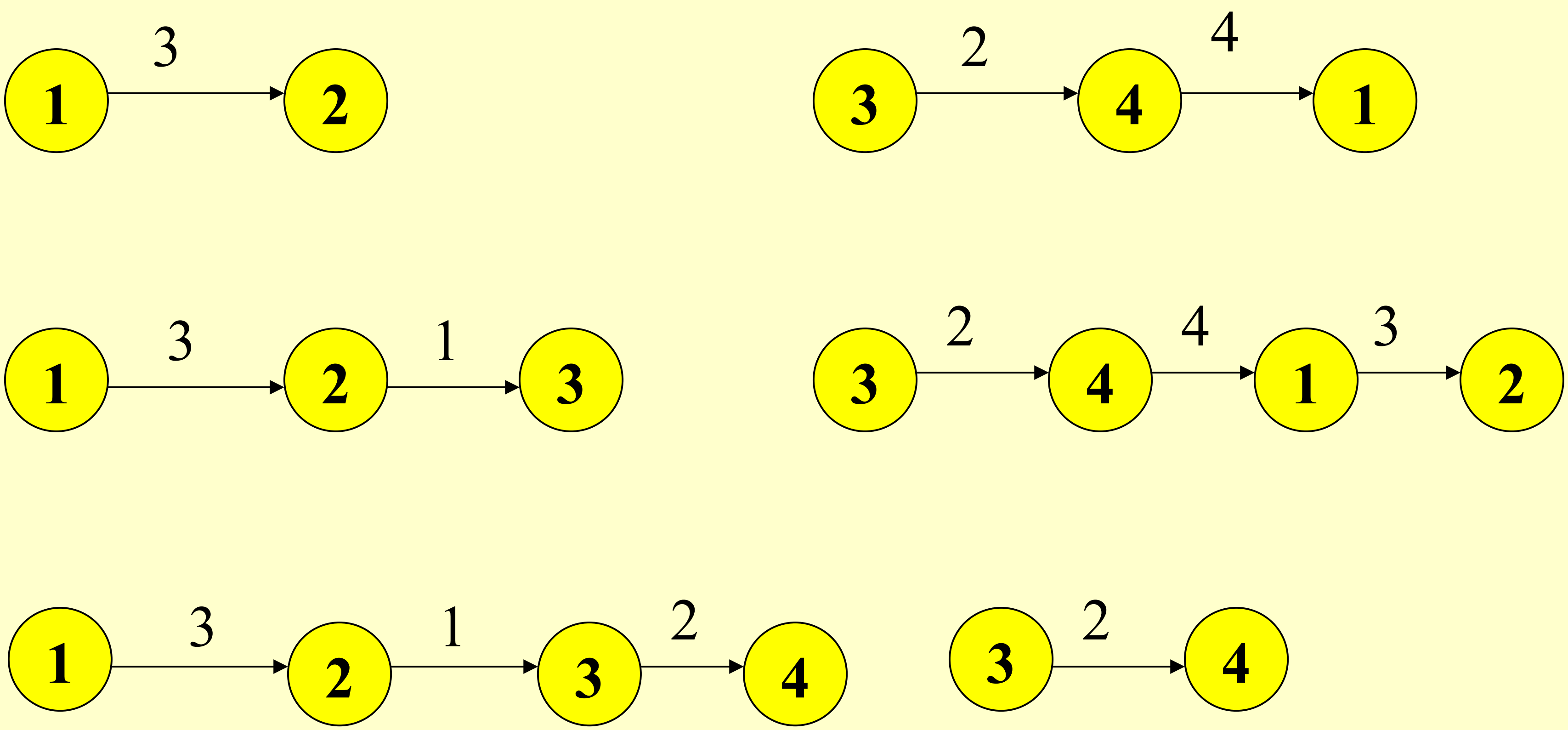
Ví dụ (tiếp)

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

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

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

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

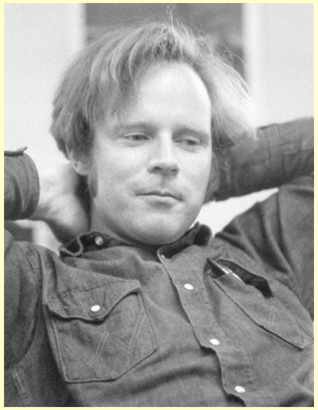
for $j \leftarrow 1$ to n do

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

return D

Thời gian tính $\boldsymbol{\Theta(n^3)}$!

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.

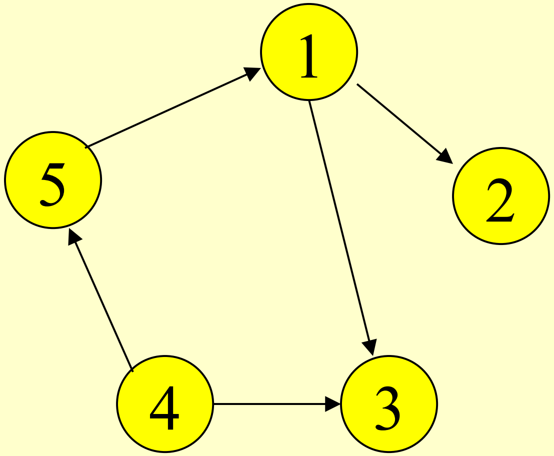
Questions?

Bao đóng truyền ứng
(Transitive Closure)

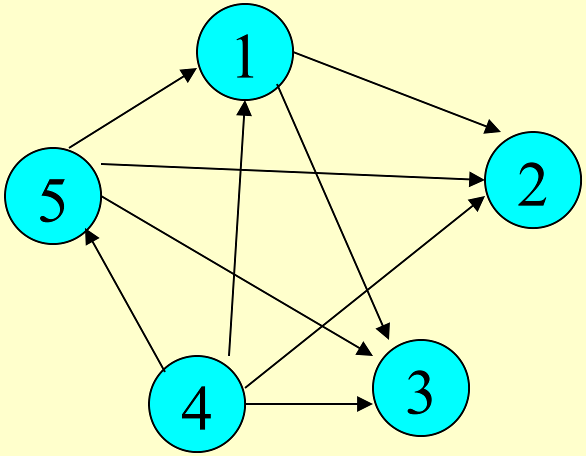
Bao đóng truyền ứng của đồ thị $G = (V, E)$ là $G^* = (V, E^*)$ sao cho

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

G:



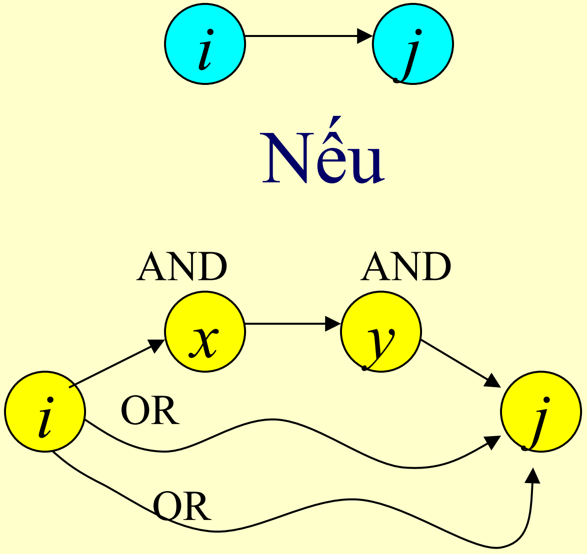
*G**:



Thuật toán Floyd-Warshall

- Ma trận xuất phát là ma trận kề

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



- Thuật toán Floyd-Warshall thay

min ⟹ boolean OR

\+ ⟹ boolean AND

- Thời gian tính $\Theta(n^3)$

Questions?