HUST

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

TRƯỜNG ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

GIẢI THUẬT ĐỊNH TUYẾN

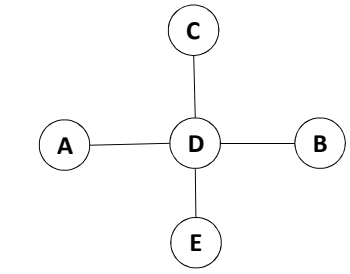
ONE LOVE. ONE FUTURE.

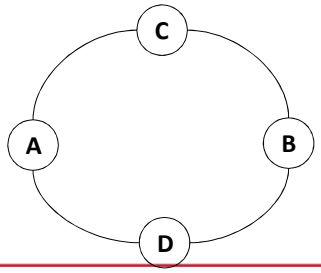
Nội dung

- Định tuyến đơn giản
- Giải thuật tìm cây MST(Cây bao trùm nhỏ nhất)
- Giải thuật tìm cây SPT(Cây đường ngắn nhất)
- Giải thuật Floyd-Warshall

Vai trò của định tuyến trong mạng thông tin

- Mạng sao
  - Đường đi từ A đến B chỉ duy nhất 1 đường
- Mạng ring (Mạng vòng)
- Đường đi từ A đến B chỉ có 02 đường



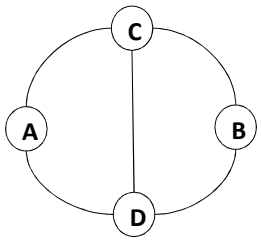


Vai trò của định tuyến trong mạng thông tin (cont.)

- Mạng lưới
- Đường đi từ A đến B có nhiều đường đi

→ Cần phải lựa chọn đường đi:

Đường đi ngắn nhất
Đường đi có giá nhỏ nhất
Đường đi nhanh nhất
Đường đi có xác suất tắc nghẽn thấp nhất
....



Định tuyến đơn giản

- Lan tràn gói
- Bước nhảy ngẫu nhiên
- Khoai tây nóng

Lan tràn gói (flooding) (1)

- Trong phương thức này, mỗi gói đi đến router sẽ được gửi đi trên tất cả các đường ra trừ đường mà nó đi đến.
- Phương thức lan tràn gói này hiển nhiên là tạo ra rất nhiều gói sao chép (duplicate). Trên thực tế, số gói này là không xác định trừ khi thực hiện một số biện pháp để hạn chế quá trình này.

Lan tràn gói (flooding) (2)

- Các biện pháp:
  - Sử dụng bộ đếm bước nhảy trong phần tiêu đề của mỗi gói:
    - Giá trị này sẽ bị giảm đi một tại mỗi bước nhảy. Gói sẽ bị loại bỏ khi bộ đếm đạt giá trị không.
    - Về mặt lý tưởng, bộ đếm bước nhảy sẽ có giá trị ban đầu tương ứng với độ dài từ nguồn đến đích.
    - Nếu như người gửi không biết độ dài của đường đi, nó có thể đặt giá trị ban đầu của bộ đếm cho trường hợp xấu nhất. Khi đó giá trị ban đầu đó sẽ được đặt bằng đường kính của mạng con.

Lan tràn gói (flooding) (3)

- Các biện pháp (cont.):
  - Thêm số thứ tự vào tiêu đề các gói.
    - Mỗi router sẽ cần có một danh sách theo nút nguồn để chỉ ra những số thứ tự từ nguồn đó đã được xem xét.
    - Để tránh danh sách phát triển không giới hạn, mỗi danh sách sẽ tăng lên bởi số đếm *k* để chỉ ra rằng tất cả các số thứ tự đến *k* đã được xem.
    - Khi một gói đi tới, rất dễ dàng có thể kiểm tra được gói là bản sao hay không. Nếu đúng gói là bản sao thì gói này sẽ bị loại bỏ.

Lan tràn gói (flooding) (4)

- Ưu điểm:
  - Lan tràn gói luôn luôn chọn đường ngắn nhất.
  - Có được ưu điểm này là do về phương diện lý thuyết nó chọn tất cả các đường có thể do đó nó sẽ chọn được đường ngắn nhất.
- Nhược điểm:
  - Số lượng gói gửi trong mạng quá nhiều.

Lan tràn gói (flooding) (5)

- Ứng dụng:
  - Trong ứng dụng quân sự, mạng sử dụng phương thức lan tràn gói để giữ cho mạng luôn luôn hoạt động tốt khi đối mặt với quân địch.
  - Dùng để truyền quảng bá.

Lan tràn gói (flooding) (6)

- Ứng dụng (cont.)
  - Trong những ứng dụng cơ sở dữ liệu phân bố, đôi khi cần thiết phải cập nhật tất cả cơ sở dữ liệu. Trong trường hợp đó sử dụng lan tràn gói là cần thiết.
  - Ví dụ sử dụng lan tràn gói để gửi cập nhật bản định tuyến bởi vì cập nhật không dựa trên độ chính xác của bảng định tuyến.

Lan tràn gói (flooding) (7)

- Ứng dụng (cont.)
  - Phương pháp lan tràn gói có thể được dùng như là đơn vị để so sánh phương thức định tuyến khác. Lan tràn gói luôn luôn chọn đường ngắn nhất. Điều đó dẫn đến không có giải thuật nào có thể tìm được đường đi có độ trễ ngắn hơn.

Lan tràn gói (flooding) (8)

- Một biến đổi của phương pháp lan tràn gói là lan tràn gói có chọn lọc. Trong giải thuật này, router chỉ gửi gói đi ra trên các đường mà đi theo hướng đích. Điều đó có nghĩa là không gửi gói đến những đường mà rõ rang nằm trên hướng sai.

Định tuyến bước ngẫu nhiên (random walk)

- Router sẽ chuyển gói đi đến trên một đường đầu ra được chọn một cách ngẫu nhiên.
- Mục tiêu của phương pháp này là các gói lang thang trong mạng cuối cùng cũng đến đích.
- Với phương pháp này giúp cho quá trình cân bằng tải giữa các đường.

Định tuyến bước ngẫu nhiên (random walk) (cont.)

- Ưu điểm:
  - Phương pháp này luôn đảm bảo là gói cuối cùng sẽ đến đích.
  - So với phương pháp trước thì sự nhân rộng gói trong mạng sẽ ít hơn.
- Nhược điểm:
  - Đường từ nguồn đến đích có thể dài hơn đường ngắn nhất. Do đó trễ đường truyền sẽ dài hơn.

Định tuyến bước ngẫu nhiên (random walk) (cont.)

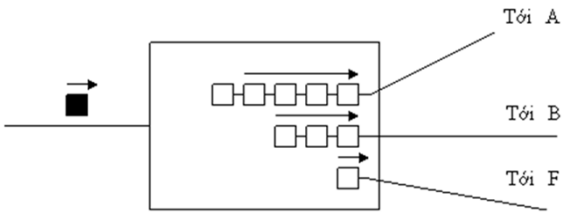
- Ứng dụng: unicast (point to point)

Định tuyến khoai tây nóng (hot potato)

- Router quyết định tuyến đi chỉ dựa vào thông tin bản thân nó lượm lặt được.
- Khi một gói đến một nút, router sẽ cố gắng chuyển gói đó đi càng nhanh càng tốt bằng cách cho nó vào hàng chờ đầu ra ngắn nhất.
- Nói cách khác, khi có gói đi đến router sẽ tính toán số gói được nằm chờ để truyền trên mỗi đường đầu ra. Sau đó nó sẽ gán gói mới vào cuối hàng chờ ngắn nhất mà không quan tâm đến đường đó sẽ đi đâu.

Định tuyến khoai tây nóng (cont.)

- Có ba hàng chờ đầu ra tương ứng với 03 đường ra. Các gói đang xếp hàng trên mỗi đường để chờ được truyền đi.
- Hàng chờ đến *F* là hàng chờ ngắn nhất với chỉ có một gói nằm trên hàng chờ này.
- Giải thuật khoai tây nóng do đó sẽ đặt gói mới đến vào hàng chờ này.



Định tuyến khoai tây nóng(cont.)

- Ưu điểm:
  - Phương pháp này luôn đảm bảo là gói cuối cùng sẽ đến đích.
  - So với phương pháp trước thì đường đi từ nguồn đến đích thường ngắn hơn
- Nhược điểm:
  - So với phương pháp trước thì phức tạp hơn.

Định tuyến khoai tây nóng(cont.)

- Ứng dụng: Unicast

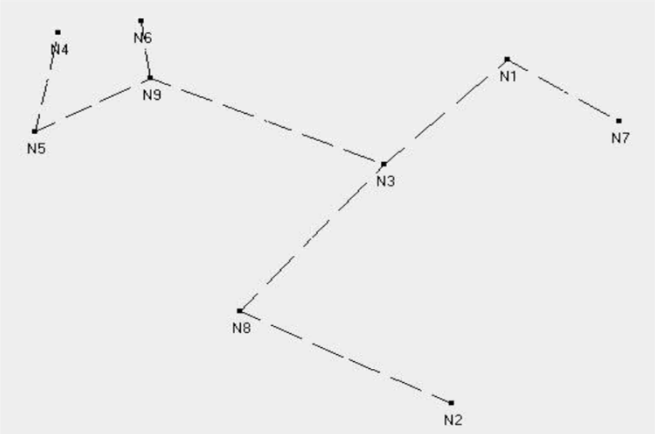
MST (Minimum Spanning Tree)

- Đồ thị con bao trùm bao gồm tất cả các nút của G
- Cây T gọi là cây bao trùm nếu như T là đồ thị con bao trùm của G
- Cây bao trùm nhỏ nhất (MST) là cây bao trùm của G với trọng số nhỏ nhất

Tìm MST

- Hai giải thuật *Kruskal* và *Prim*
- Kruskal tìm MST bằng việc bắt đầu với đồ thị bỏ đi các cạnh
- Prim
  - Bắt đầu bằng việc lựa chọn nút
  - Thêm cạnh có trọng số nhỏ nhất
  - Lặp lại cho đến khi cây được xây dựng

Ví dụ MST



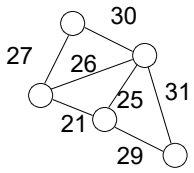
Sử dụng MST

- Truyền quảng bá (broadcast)

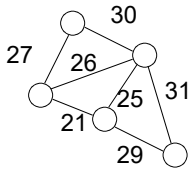
Giải thuật Kruskal (1956)

- 1. Kiểm tra xem đồ thị G có liên thông không. Nếu như không liên thông thì ngừng
- 2. Sắp xếp các cạnh của đồ thị theo thứ tự tăng dần của trọng số.
- 3. Đánh dấu mỗi nút như là thành phần riêng.
- 4. Xem xét mỗi cạnh theo thứ tự sắp xếp
  - Nếu cạnh nối hai thành phần riêng biệt thì thêm cạnh đó vào còn nếu không thì loại bỏ

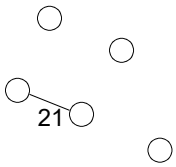
Giải thuật Kruskal cho MST (2)



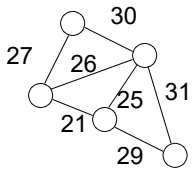
Giải thuật Kruskal cho MST (2)



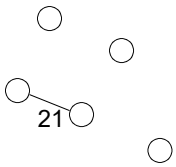
Lần thêm đầu tiên



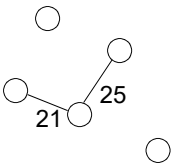
Giải thuật Kruskal cho MST (2)



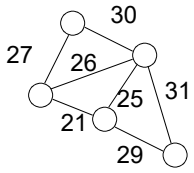
Lần thêm thứ 1



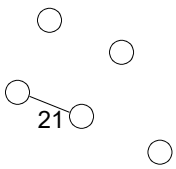
Lần thêm thứ 2



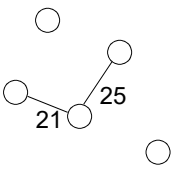
Giải thuật Kruskal cho MST (2)



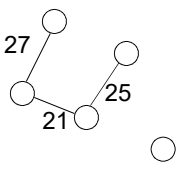
Lần thêm thứ 1



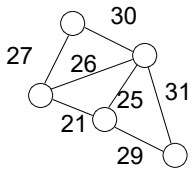
Lần thêm thứ 2



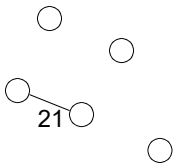
Lần thêm thứ 3



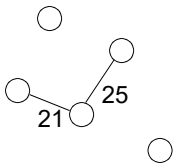
Giải thuật Kruskal cho MST (2)



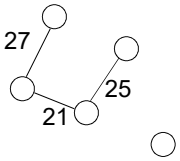
Lần thêm thứ 1



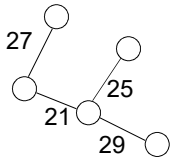
Lần thêm thứ 2



Lần thêm thứ 3

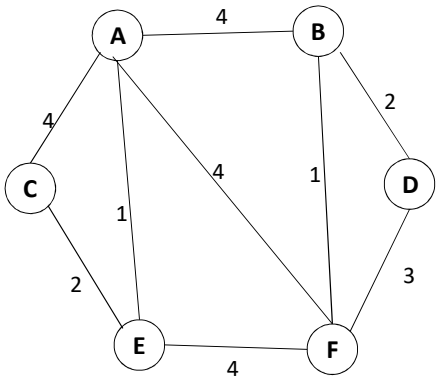


Lần thêm thứ 4



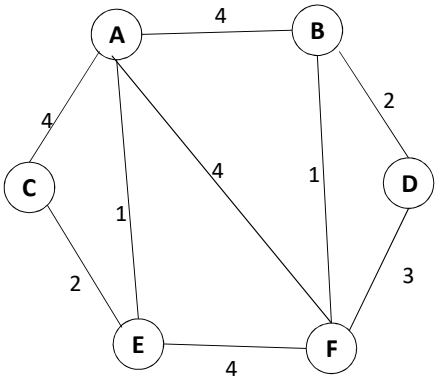
Ví dụ 1 Kruskal(1)

- Tìm cây MST



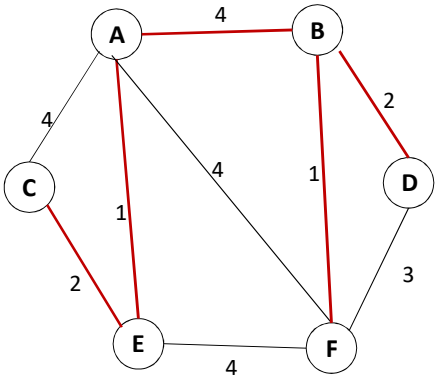
Ví dụ 1 Kruskal(3)

TT	Liên kết	Giá của liên kết	Giải pháp
1	AE	1	
2	BF	1	
3	CE	2	
4	BD	2	
5	DF	3	
6	CA	4	
7	AB	4	
8	EF	4	
9	AF	4	

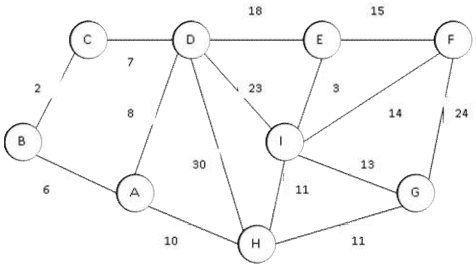


Ví dụ 2 Kruskal(4)

TT	Liên kết	Giá của liên kết	Giải pháp
1	AE	1	Chọn
2	BF	1	Chọn
3	CE	2	Chọn
4	BD	2	Chọn
5	DF	3	Không chọn (tạo vòng)
6	CA	4	Không chọn (tạo vòng)
7	AB	4	Chọn
8	EF	4	Dừng vì đã thêm đủ cạnh
9	AF	4	vào cây

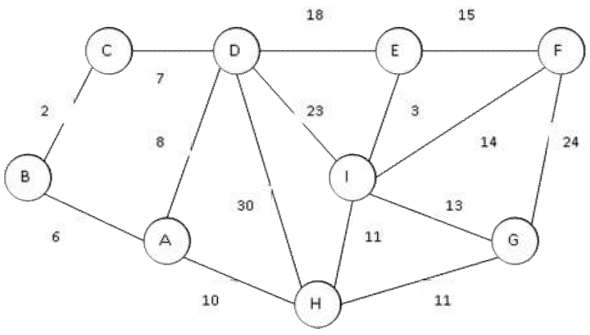


Ví dụ 2 Kruskal(1)



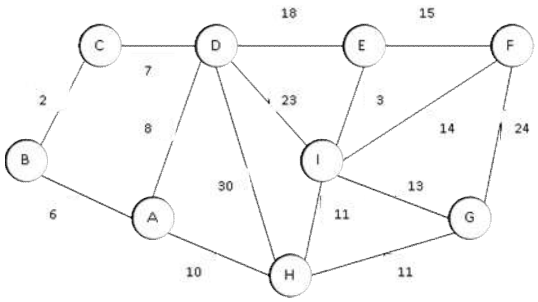
Ví dụ 2 Kruskal(2)

TT	Liên kết	Giá của liên kết	Giải pháp
1	BC	2	
2	IE	3	
3	BA	6	
4	CD	7	
5	AD	8	
6	AH	10	
7	HI	11	
8	HG	11	
9	IG	13	
10	IF	14	
11	EF	15	
12	DE	18	
13	DI	23	
14	GF	24	
15	DH	30	



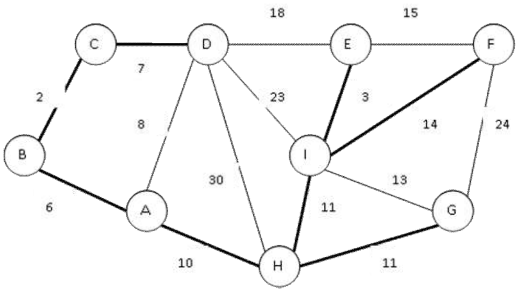
Ví dụ 2 Kruskal(3)

TT	Liên kết	Giá của liên kết	Giải pháp
1	BC	2	Chọn
2	IE	3	Chọn
3	BA	6	Chọn
4	CD	7	Chọn
5	AD	8	Không chọn (tạo vòng)
6	AH	10	Chọn
7	HI	11	Chọn
8	HG	11	Chọn
9	IG	13	Không chọn (tạo vòng)
10	IF	14	Chọn
11	EF	15	Dừng vì đã thêm đủ cạnh
12	DE	18	vào cây
13	DI	23	
14	GF	24	
15	DH	30	



Ví dụ 2 Kruskal(4)

- Tổng độ dài cây MST là :
- L = 2+3+6+7+10+11+11+14 = 64



Giải thuật Prim (1957)

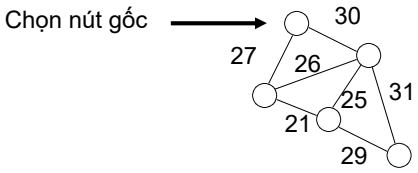
- Đầu vào: đồ thị liên thông có trọng số G=(N,E).
- Đầu ra : Cây bao trùm nhỏ nhất T.

- U = Tập các nút trên MST
- V = Tập tất cả các nút chưa thuộc MST nhưng nó là liền kề những nút thuộc U

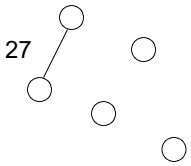
Giải thuật Prim

- 1. Đặt bất kỳ nút nào vào U và cập nhật V
- 2. Tìm cạnh có trọng số nhỏ nhất nối nút thuộc V tới nút thuộc U
- 3. Thêm cạnh đó vào cây và cập nhật U và V
- 4. Lặp lại 2 & 3 cho đến khi tất cả mọi nút đều thuộc cây, | U | = | N |.

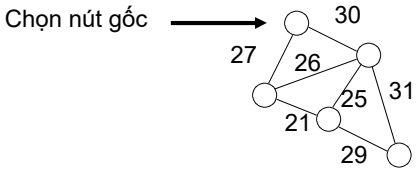
Giải thuật Prim cho MST (2)



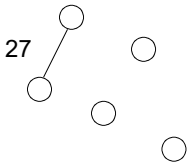
Lần thêm thứ 1



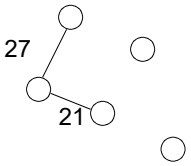
Giải thuật Prim cho MST (2)



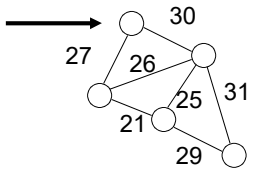
Lần thêm thứ 1



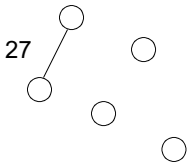
Lần thêm thứ 2



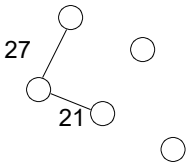
Giải thuật Prim cho MST (2)



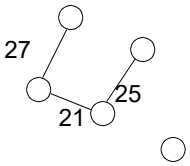
Lần thêm thứ 1



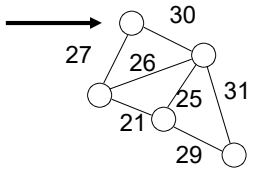
Lần thêm thứ 2



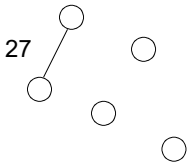
Lần thêm thứ 3



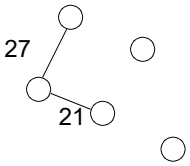
Giải thuật Prim cho MST (2)



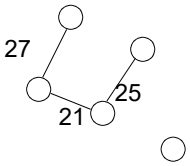
Lần thêm thứ 1



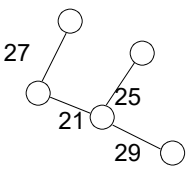
Lần thêm thứ 2



Lần thêm thứ 3

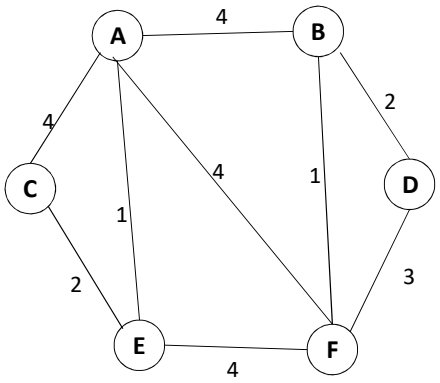


Lần thêm thứ 4



Ví dụ 1 Prim(1)

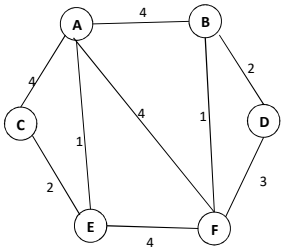
- Tìm cây MST

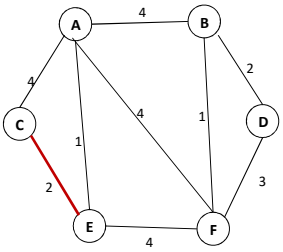


Ví dụ 1 Prim(2)

- U={C},V={E,A}
  - CE=2 là nhỏ nhất.

  → Thêm CE vào cây

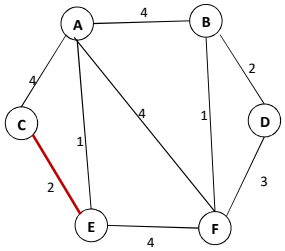


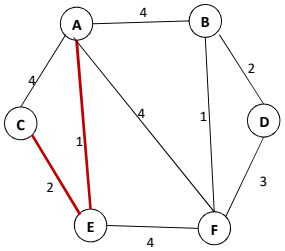


Ví dụ 1 Prim(3)

- U={C,E},V={A,F}
  - EA=1 là nhỏ nhất.

  → Thêm EA vào cây

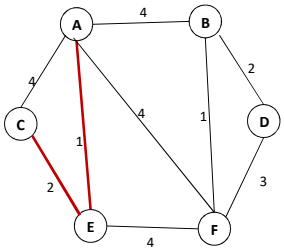


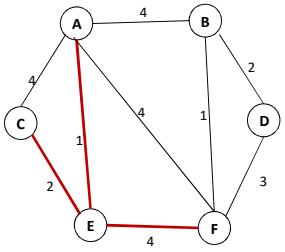


Ví dụ 1 Prim(4)

- U={C,E,A},V={F,B}
  - EF=4 là nhỏ nhất.

  → Thêm EF vào cây

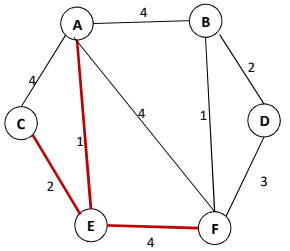


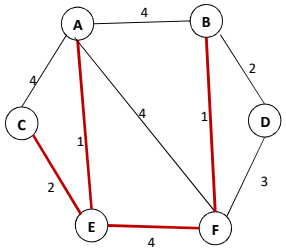


Ví dụ 1 Prim(5)

- U={C,E,A,F},V={B,D}
  - FB=1 là nhỏ nhất.

  → Thêm FB vào cây

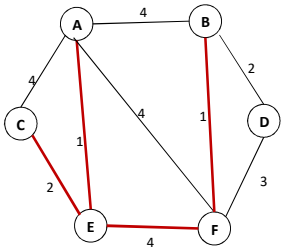


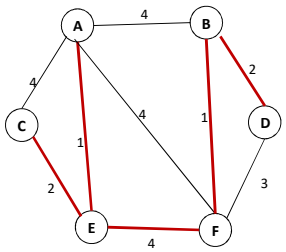


Ví dụ 1 Prim(6)

- U={C,E,A,F,B},V={D}
  - BD=2 là nhỏ nhất.

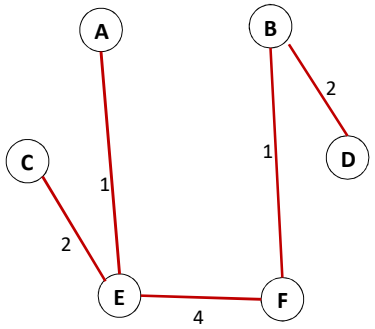
  → Thêm BD vào cây





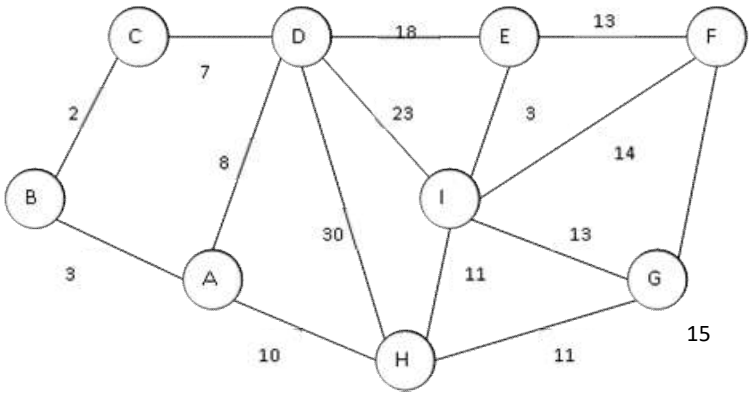
Ví dụ 1 Prim(7)

- $|U| = N$ *nên kết thúc*
- $\sum w = 1 + 2 + 1 + 2 + 4 = 10$



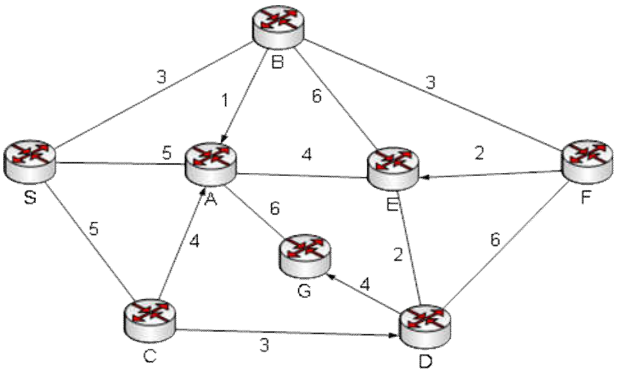
Ví dụ 2 Prim (1)

- Tìm cây MST



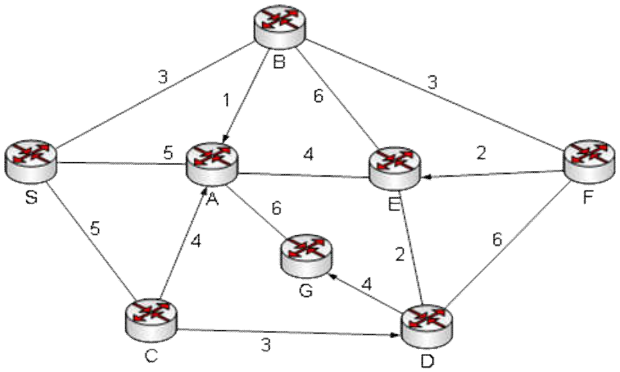
Ví dụ 3 Prim(1)

- Tìm cây đường đi MST từ nút nguồn S đến tất cả các nút còn lại theo thuật toán Prim



Ví dụ 3 Prim(2)

- SB,BA,BF,
- FE,ED,DG,
- SC
- 3+1+3+2+2+4+5=20



Giới hạn của MSTs

- Không có tính dư (dự phòng)
  - Một liên kết hỏng sẽ tách mạng thành hai thành phần không liên kết
  - Vấn đề lớn đối với mạng lớn
- Khi mạng lớn có thể có những đường rất dài

SPT

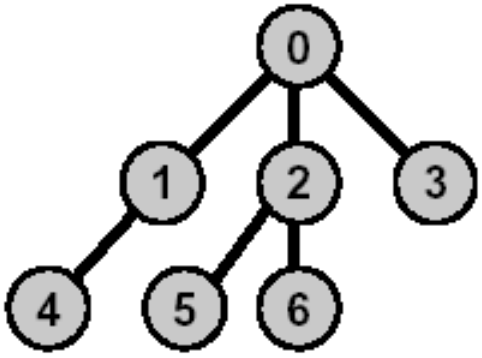
- Cho đồ thị trọng số (G,w), và nút n1 và n2, đường ngắn nhất P từ n1 đến n2 có giá trị $\Sigma_{e\epsilon P} w(e)$ nhỏ nhất
- Cây đường ngắn nhất shortest-path tree (SPT) có gốc tại nút n1 là cây T mà có đường từ n1 đến n2 là ngắn nhất với bất kỳ nút n2 nào
- Chú ý – không giống như MST, SPT có nút gốc - sự lựa chọn gốc khác nhau sẽ cho những cây khác nhau

Hàm "tiền bối"

- Cây T có gốc tại nút gốc có thể biểu đơn giản bởi hàm tiền bối pred: V $\rightarrow$V
- Hàm tiền bối :
  - pred(gốc) = gốc
  - Với mọi nút N luôn tồn tại giá trị n>0 sao cho $pred^n(N)$ = gốc
- Cây được định nghĩa bằng tập các nút V và các cạnh (V,pred(V))

Con cháu

- Cho một cây T và hàm tiền bối, con cháu của nút N là tất cả các nút N* mà $pred^n(N^*) = N$ với giá trị n nào đó > 0

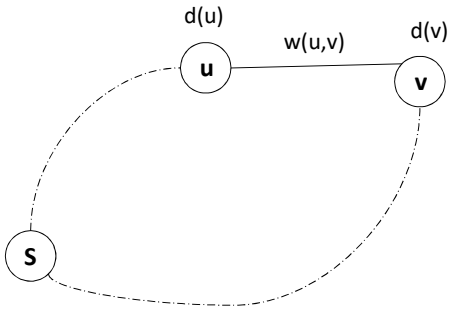


Con cháu(1) = { 4 }
Con cháu (2) = { 5, 6 }
Con cháu (6) = { }

Giải thuật Dijkstra cho SPT(1)

1. Đánh dấu các nút chưa được xét, ấn định nhãn vô cùng
2. Thiết lập nhãn của gốc bằng 0 và thiết lập predecessor(gốc)= nút gốc.
3. Tìm nút u chưa xét có nhãn nhỏ nhất. Đánh dấu là đã xét
   - Xem xét tất cả các nút liền kề v, xem nếu khoảng cách qua u đến nút v < nhãn của nút v
     - Nếu có, cập nhật nhãn, và cập nhật predecessor(v) = u
4. Lặp lại bước 3 cho đến tất cả các nút đã được xét.

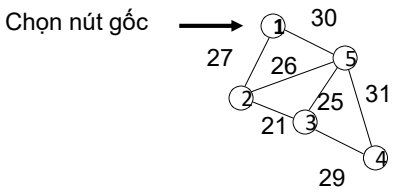
- Nhãn nút u : d(u)
- nếu khoảng cách qua u đến nút v < nhãn của nút v
- →If d(u)+w(u,v)< d(v)



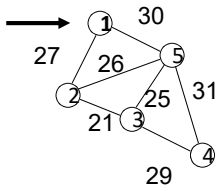
Giải thuật Dijkstra cho SPT(2)

1. $S=\{\}; d(v) = \infty \text{ với } \forall\, v \neq s$
2. $d(s) = 0,\ \pi(s) = s,$
3. Nút u: có d(u) min và $u \notin S$
   - **Với mỗi nút liền kề** v:
     - If d(u)+w(u,v)< d(v) then D(v)= d(u)+w(u,v) và $\pi(v) = u$
4. Lặp lại bước 3 cho đến khi $|S| = N$

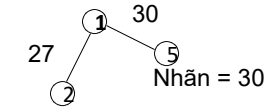
Ví dụ Dijkstra 1



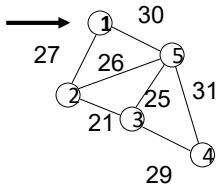
Ví dụ Dijkstra 1



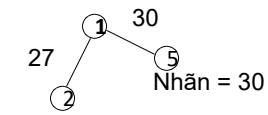
Nút 5 & 2.liền kề với gốc



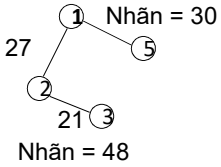
Ví dụ Dijkstra 1



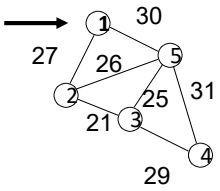
5 & 2 liền kề với gôc



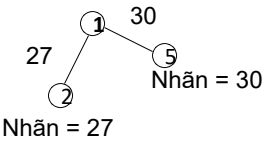
Nút 3 và 5 liền kề với 2



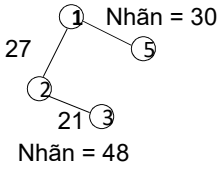
Ví dụ Dijkstra 1



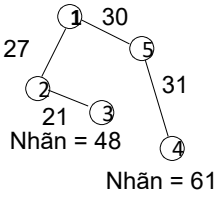
5 & 2 liền kề với gốc



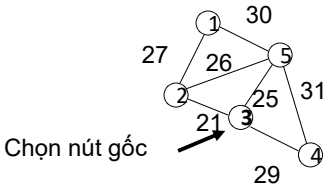
3& 5 liền kề với 2



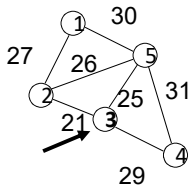
4&3. liền kề với 5



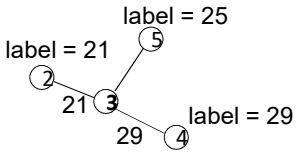
Ví dụ Dijkstra 2



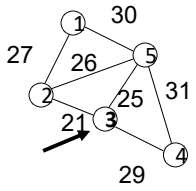
Ví dụ Dijkstra 2



4, 2 & 5 liền kề với nút gốc

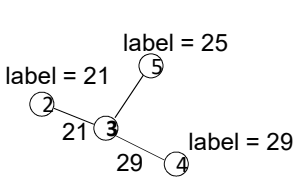


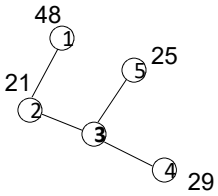
Ví dụ Dijkstra 2



4, 2 & 5 liền kề với nút gốc

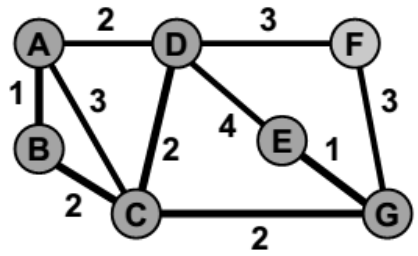
1 ,3& 5. liền kề nút 2





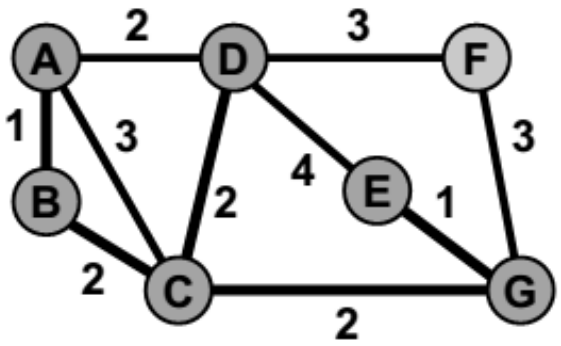
Ví dụ 1 Dijkstra(1)

- Tìm cây đường ngắn nhất với nút nguồn tại D



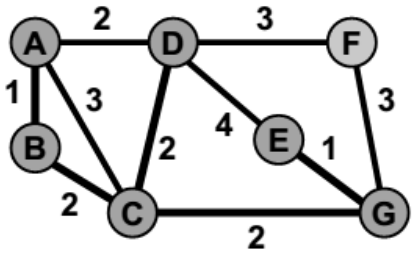
Ví dụ 1 Dijkstra(2)

- S={}; $d(v) = \infty$ với $\forall\, v \neq D$; $d(D) = 0$, $\pi(D) = D, d(v) = \infty$ với $\forall\, v \neq D$,



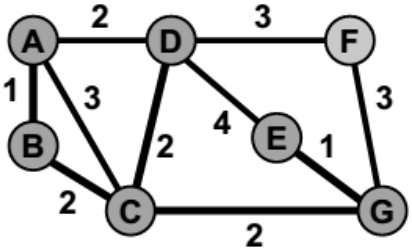
Ví dụ 1 Dijkstra(3)

- *Nút D:* S={D}
- $Vì\ d(D) + w(D,A) = 0 + 2 < \infty\ nên\ d(A) = 2,\ \pi(A) = D$
- $Vì\ d(D) + w(D,C) = 0 + 2 < \infty\ nên\ d(C) = 2,\ \pi(C) = D$
- $Vì\ d(D) + w(D,E) = 0 + 4 < \infty\ nên\ d(E) = 4,\ \pi(E) = D$
- $Vì\ d(D) + w(D,F) = 0 + 3 < \infty\ nên\ d(F) = 3,\ \pi(F) = D$



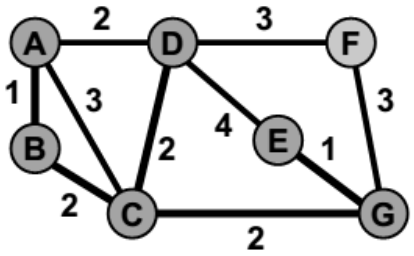
Ví dụ 1 Dijkstra(4)

- *Nút A:* S={D,A}
- $Vì\ d(A) + w(A,B) = 2 + 1 < \infty\ nên\ d(B) = 3,\ \pi(B) = \mathrm{A}$
- $Vì\ d(A) + w(A,C) = 2 + 3 > 2\ \ nên\ d(C) không\ thay$ đổi



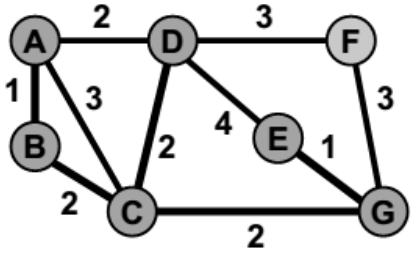
Ví dụ 1 Dijkstra(4)

- *Nút C:* S={D,A,C}
- $Vì\ d(C) + w(C,G) = 2 + 2 < \infty\ nên\ d(G) = 4,\ \pi(G) = C$
- $Vì\ d(C) + w(C,B) = 2 + 2 > 3\ nên\ d(B) không\ thay$ đổi



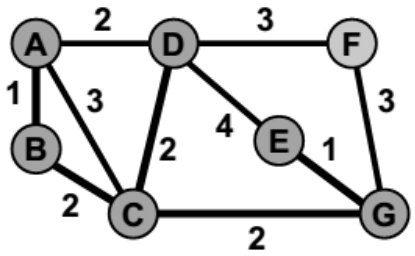
Ví dụ 1 Dijkstra(4)

- *Nút B:* S={D,A,C,B}
- *Nút F:* S={D,A,C,B,F}
- $Vì\ d(F) + w(F,G) = 3 + 3 > 4\ nên\ d(G) không\ thay$ đổi



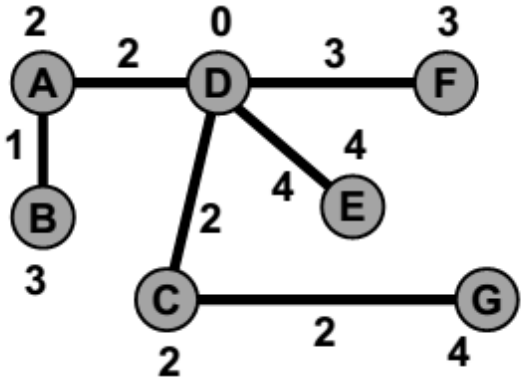
Ví dụ 1 Dijkstra(4)

- *Nút E:* S={D,A,C,B,F,E}
- $Vì\ d(E) + w(E,G) = 4 + 1 > 4\ nên\ d(G) không\ thay\ đổi$
- *Nút G:* $|S| = N \rightarrow Kết\ thúc$



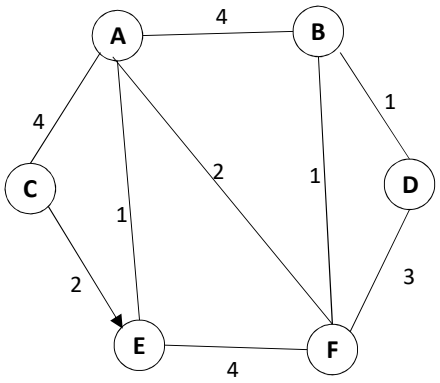
Ví dụ 1 Dijkstra(5)

- $\pi(A) = D$
- $\pi(C) = D$
- $\pi(E) = D$
- $\pi(F) = D$
- $\pi(B) = A$
- $\pi(G) = C$



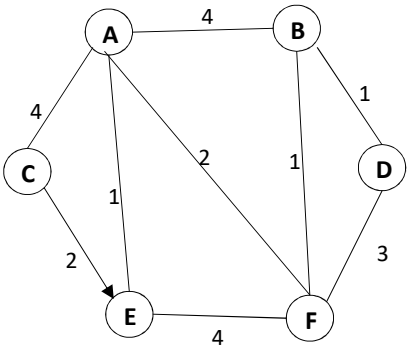
Ví dụ 2 Dijkstra(1)

- Tìm cây đường ngắn nhất với nút nguồn tại A



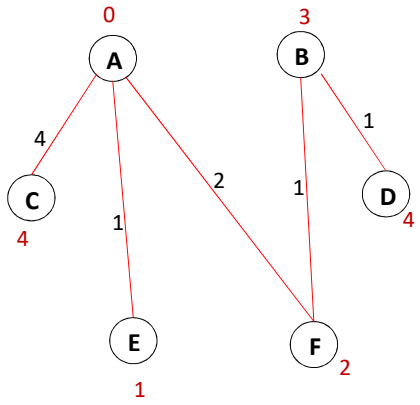
Ví dụ 2 Dijkstra(2)

- S={}; $d(v) = \infty$ $với$ $\forall\ v \neq A$; $d(A) = 0$, $\pi(A) = A, d(v) = \infty$ $với$ $\forall\ v \neq A$,



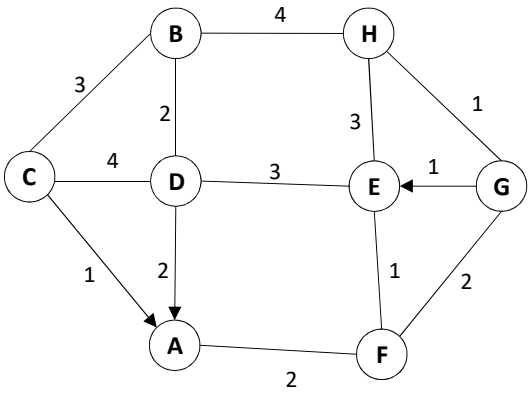
Ví dụ 2 Dijkstra(3)

- Cây đường ngắn nhất

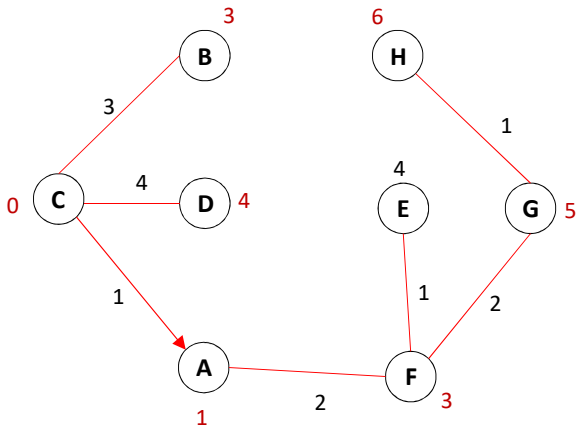


Ví dụ 3 Dijkstra(1)

- Tìm cây đường ngắn nhất với nút nguồn tại C

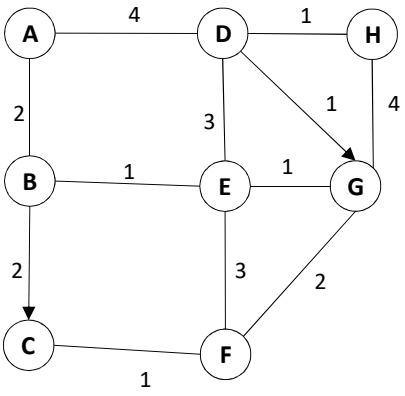


Ví dụ 3 Dijkstra(1)



Ví dụ 4 Dijkstra

- Tìm cây đường ngắn nhất với nút nguồn Tại B



Giải thuật Bellman –Ford (1)

- Thuật toán làm việc với đồ thị có trọng số âm nhưng không có chu trình mà tổng trọng số là âm.
- Thuật toán tìm tất cả đường đi ngắn nhất đến các đỉnh còn lại.
- Sử dụng hàng đợi Q theo nguyên tắc FIFO

Giải thuật Bellman –Ford (2)

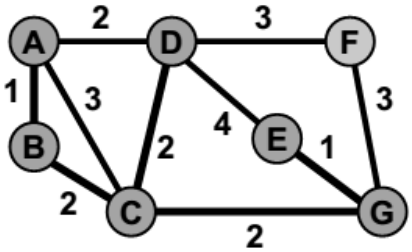
1. Thiết lập nhãn của gốc bằng 0, thiết lập predecessor(gốc)= nút gốc.Thiết lập nhãn các nút khác bằng vô cùng. Hàng đợi Q ban đầu chỉ có mình nút gốc.
2. Lấy u từ hàng đợi Q
   - Xem xét tất cả các nút liền kề v, xem nếu khoảng cách qua u < nhãn
     - Nếu có, cập nhật nhãn, và cập nhật predecessor(v) = u, chèn nút v vào hàng đợi Q nếu như v chưa có mặt trong hàng đợi
3. Lặp lại bước 2 cho đến khi không còn nút trong hàng đợi

Giải thuật Bellman –Ford (3)

1. $d(s) = 0,\ \pi(s) = s, d(v) = \infty$ *với* $\forall\ v \neq s,\ Q =$ s
2. $Q \rightarrow u$
   - Với mỗi nút liền kề v:
     - If d(u)+w(u,v)<d(v) then
     - d(v)= d(u)+w(u,v) và $\pi(v) = u$ *and* if v $\notin Q$ *then* $v \rightarrow$Q
3. Lặp lại bước 2 cho đến khi Q = /

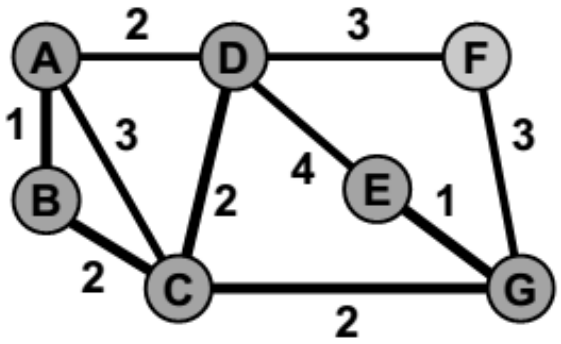
Ví dụ 1 Bellman-Ford(1)

- Tìm cây đường ngắn nhất với nút nguồn tại D



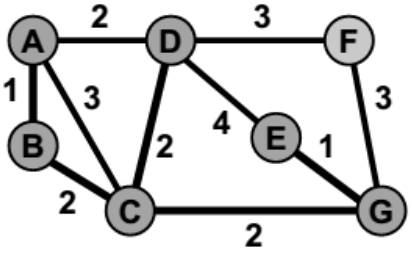
Ví dụ 1 Bellman-Ford(2)

- $d(v) = \infty \; với \; \forall \; v \neq D \; ; \; d(D) = 0, \; \pi(D) = D, d(v) = \infty \; với \; \forall \; v \neq D, \; Q=$ D



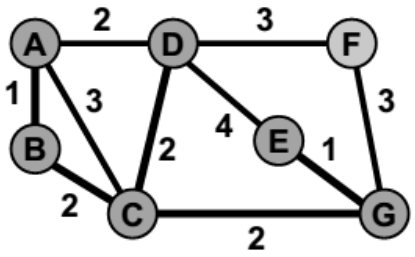
Ví dụ 1 Bellman-Ford(3)

- *Nút D:* Q= /
- $Vì \; d(D) + w(D, A) = 0 + 2 < \infty \; nên \; d(A) = 2, \; \pi(A) = D,$ Q= A
- $Vì \; d(D) + w(D, C) = 0 + 2 < \infty \; nên \; d(C) = 2, \; \pi(C) = D,$ Q= C A
- $Vì \; d(D) + w(D, E) = 0 + 4 < \infty \; nên \; d(E) = 4, \; \pi(E) = D,$ Q= E C A
- $Vì \; d(D) + w(D, F) = 0 + 3 < \infty \; nên \; d(F) = 3, \; \pi(F) = D,$ Q= F E C A



Ví dụ 1 Bellman-Ford(4)

- *Nút A:* Q= F | E | C
- $Vì\ d(A) + w(A,B) = 2 + 1 < \infty\ nên\ d(B) = 3,\ \pi(B) = A,$
  Q = B | F | E | C
- $Vì\ d(A) + w(A,C) = 2 + 3 > 2\ \ nên\ d(C) không\ thay$ đổi



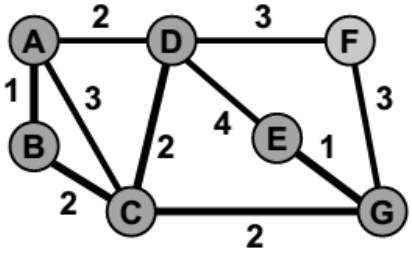
Ví dụ 1 Bellman-Ford (5)

Nút C: Q= B | F | E

$Vì\ d(C) + w(C,G) = 2 + 2 < \infty\ nên\ d(G) = 4,\quad \pi(G) = C$

Q = G | B | F | E

- $Vì\ d(C) + w(C,B) = 2 + 2 > 3\ nên\ d(B) không\ thay$ đổi
- $Vì\ d(C) + w(C,A) = 2 + 3 > 2\ nên\ d(A) không\ thay$ đổi



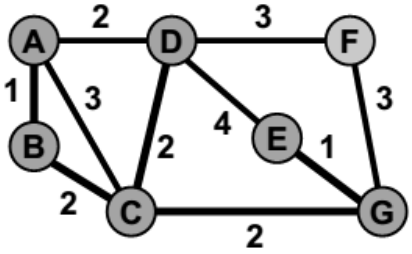
Ví dụ 1 Bellman-Ford (6)

Nút E: Q= | G | B | F |

$Vì\ d(E) + w(E,G) = 4 + 1 > 4\ nên\ d(G)\ không\ thay\ đổi$

Nút F: Q= | G | B |

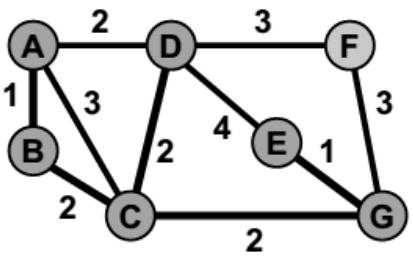
$Vì\ d(F) + w(F,G) = 3 + 3 > 4\ nên\ d(G)\ không\ thay\ đổi$



Ví dụ 1 Bellman-Ford (7)

Nút B: Q= | G |

$Vì\ d(B) + w(B,C) = 3 + 2 > 2\ nên\ d(C)\ không\ thay\ đổi$
$Vì\ d(B) + w(B,A) = 3 + 1 > 2\ nên\ d(A)\ không\ thay\ đổi$

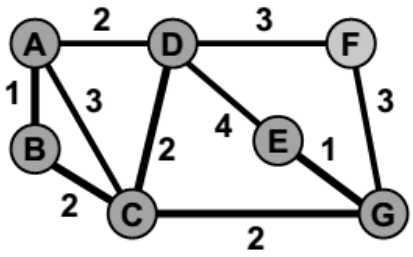


Ví dụ 1 Bellman-Ford (8)

Nút G: Q=/

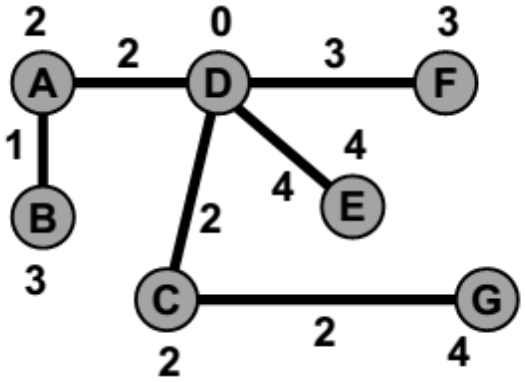
$Vì\ d(G) + w(G,C) = 4 + 2 > 2\ nên\ d(C)\ không\ thay\ đổi$
$Vì\ d(G) + w(G,E) = 4 + 1 > 4\ nên\ d(E)\ không\ thay\ đổi$
$Vì\ d(G) + w(G,F) = 4 + 3 > 3\ nên\ d(F)\ không\ thay\ đổi$

Kết thúc



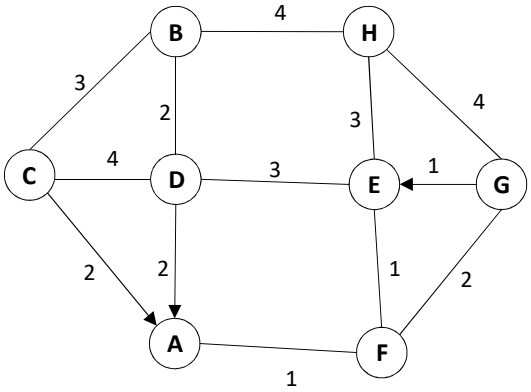
Ví dụ 1 Bellman-Ford(9)

- $\pi(A) = D$
- $\pi(C) = D$
- $\pi(E) = D$
- $\pi(F) = D$
- $\pi(B) = A$
- $\pi(G) = C$



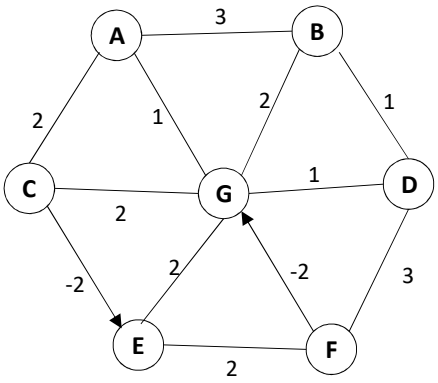
Ví dụ 2 Bellman-Ford(1)

- Tìm cây đường ngắn nhất với nút nguồn tại B



Ví dụ 3 Bellman-Ford(1)

- Tìm cây đường ngắn nhất với nút nguồn tại A



Đặc tính của SPT

- Trong đồ thị đầy đủ, SPT dạng sao
  - Chất lượng hoạt động và độ tin cạy cao
  - Nhưng độ sử dụng đường liên kết thấp→ tiêu tốn

Thuật tóan Floyd-Warshall (1)

- Giải bài toán tìm đường đi ngắn nhất giữa mọi cặp đỉnh trên đồ thị
- Nhiều cách giải: Dùng Dijkstra ….

Thuật tóan Floyd-Warshall (2)

- Định nghĩa: Các đỉnh v_2, ..v_{l-1} là đỉnh trung gian của đường P (v_1,v_2,...,v_l).
- $d_{ij}^{(k)}$ là đường đi ngắn nhất từ i đến j mà có tất cả nút trung gian nằm trong tập [1..k]
- $d_{ij}^{(0)}$ được đặt bằng w(i,j)
- $D^{(k)}$ là ma trận nxn của $[d_{ij}^{(k)}]$.

Thuật tóan Floyd-Warshall (3)

- Đường ngắn nhất không bao giờ đi qua một đỉnh hai lần
- Với đường ngắn nhất từ i đến j với các bất cứ đỉnh trung gian nằm trong tập từ 1..k có hai khả năng:
  - K không phải là một đỉnh nằm trên đường ngắn nhất. Đường ngắn nhất sẽ có độ dài $d_{ij}^{(k-1)}$
  - K là một đỉnh nằm trên đường ngắn nhất. Đường ngắn nhất có độ dài là $d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$

Thuật tóan Floyd-Warshall (4)

- Xét đường ngắn nhất từ i đến j có hai đường con (i,k) và (k,j). Các đường con chỉ có những đỉnh trung gian từ (1..k-1) nên đường ngắn nhất là $d_{ik}^{(k-1)}$ và $d_{kj}^{(k-1)}$.
- Kết hợp hai trường hợp

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

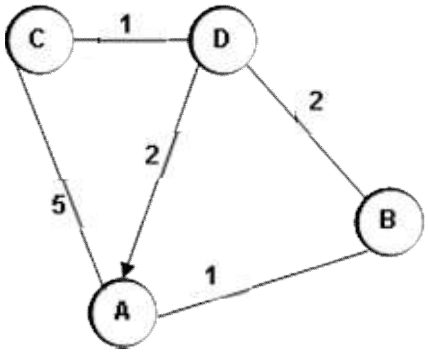
Thuật tóan Floyd-Warshall (5)

1. $D^{(0)} = [w_{ij}]$, pred (i,j)=0
2. Tính $D^{(k)}$ từ $D^{(k-1)}$ sử dụng công thức

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

pred(i,j)= k nếu như đường ngắn nhất đi qua k

Ví dụ Thuật tóan Floyd-Warshall(1)

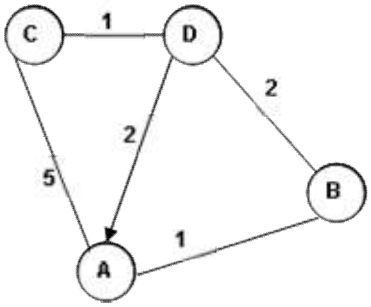


$$D^0 = \begin{bmatrix} 0 & 1 & 5 & \infty \\ 1 & 0 & \infty & 2 \\ 5 & \infty & 0 & 1 \\ 2 & 2 & 1 & 0 \end{bmatrix}$$

Ví dụ Thuật tóan Floyd-Warshall (2)

$$D^0 = \begin{bmatrix} 0 & 1 & 5 & \infty \\ 1 & 0 & \infty & 2 \\ 5 & \infty & 0 & 1 \\ 2 & 2 & 1 & 0 \end{bmatrix}$$

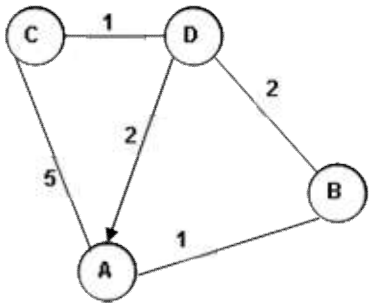
$$D^1 = \begin{bmatrix} 0 & 1 & 5 & \infty \\ 1 & 0 & & \\ 5 & & 0 & \\ 2 & & & 0 \end{bmatrix}$$



$$D^1 = \begin{bmatrix} 0 & 1 & 5 & \infty \\ 1 & 0 & 6 & 2 \\ 5 & 6 & 0 & 1 \\ 2 & 2 & 1 & 0 \end{bmatrix}$$

Ví dụ Thuật tóan Floyd-Warshall (3)

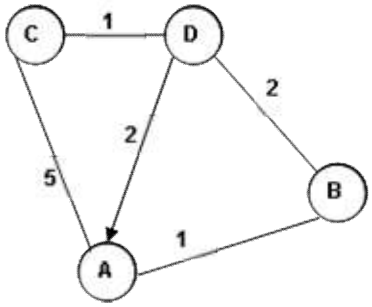
$$D^1 = \begin{bmatrix} 0 & 1 & 5 & \infty \\ 1 & 0 & 6 & 2 \\ 5 & 6 & 0 & 1 \\ 2 & 2 & 1 & 0 \end{bmatrix} \qquad D^2 = \begin{bmatrix} 0 & 1 & & \\ 1 & 0 & 6 & 2 \\ & 6 & 0 & \\ & 2 & & 0 \end{bmatrix}$$



$$D^2 = \begin{bmatrix} 0 & 1 & 5 & 3 \\ 1 & 0 & 6 & 2 \\ 5 & 6 & 0 & 1 \\ 2 & 2 & 1 & 0 \end{bmatrix}$$

Ví dụ Thuật tóan Floyd-Warshall (4)

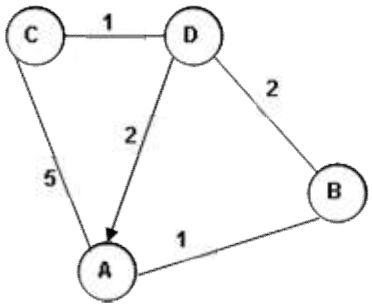
$$D^2 = \begin{bmatrix} 0 & 1 & 5 & 3 \\ 1 & 0 & 6 & 2 \\ 5 & 6 & 0 & 1 \\ 2 & 2 & 1 & 0 \end{bmatrix} \qquad D^3 = \begin{bmatrix} 0 & & 5 & \\ & 0 & 6 & \\ 5 & 6 & 0 & 1 \\ & & 1 & 0 \end{bmatrix}$$



$$D^3 = \begin{bmatrix} 0 & 1 & 5 & 3 \\ 1 & 0 & 6 & 2 \\ 5 & 6 & 0 & 1 \\ 2 & 2 & 1 & 0 \end{bmatrix}$$

Ví dụ Thuật tóan Floyd-Warshall (5)

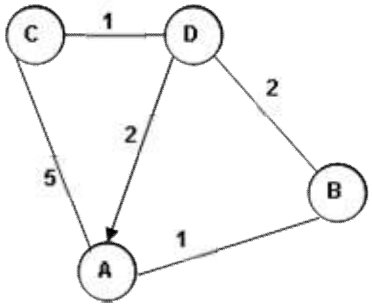
$$D^3 = \begin{bmatrix} 0 & 1 & 5 & 3 \\ 1 & 0 & 6 & 2 \\ 5 & 6 & 0 & 1 \\ 2 & 2 & 1 & 0 \end{bmatrix} \qquad D^4 = \begin{bmatrix} 0 & & & 3 \\ & 0 & & 2 \\ & & 0 & 1 \\ 2 & 2 & 1 & 0 \end{bmatrix}$$



$$D^4 = \begin{bmatrix} 0 & 1 & 4 & 3 \\ 1 & 0 & 3 & 2 \\ 3 & 3 & 0 & 1 \\ 2 & 2 & 1 & 0 \end{bmatrix}$$

Ví dụ Thuật tóan Floyd-Warshall (6)

$$D^1 = \begin{bmatrix} 0 & 1 & 5 & \infty \\ 1 & 0 & 6 & 2 \\ 5 & 6 & 0 & 1 \\ 2 & 2 & 1 & 0 \end{bmatrix} \quad D^2 = \begin{bmatrix} 0 & 1 & 5 & 3 \\ 1 & 0 & 6 & 2 \\ 5 & 6 & 0 & 1 \\ 2 & 2 & 1 & 0 \end{bmatrix} \quad D^4 = \begin{bmatrix} 0 & 1 & 4 & 3 \\ 1 & 0 & 3 & 2 \\ 3 & 3 & 0 & 1 \\ 2 & 2 & 1 & 0 \end{bmatrix}$$



$$\pi = \begin{bmatrix} 0 & 0 & D & B \\ 0 & 0 & D & 0 \\ D & D & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Thuật tóan Floyd-Warshall (6)

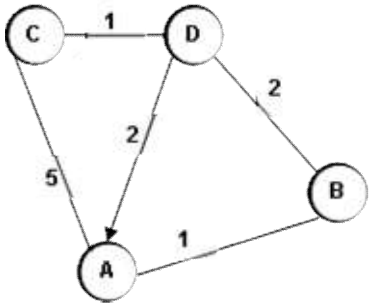
- Sử dụng pred (i,j) để tính đường ngắn nhất

1. Nếu pred (i,j)=0 thì đường ngắn nhất không có đỉnh trung gian, đường ngắn nhất chính là (i,j)

2. Nếu pred (i,j) <>0 thì chia ra thành 2 đường con (i, pred(i,j)) và (pred(i,j),j)

Ví dụ Thuật tóan Floyd-Warshall (7)

$$A \dashrightarrow C \Longrightarrow A \longrightarrow D \text{ và } D \longrightarrow C$$
$$\Longrightarrow A \longrightarrow B \text{ và } B \longrightarrow D \text{ và } DC$$
$$AB, BD \text{ và } DC$$



$$\pi = \begin{bmatrix} 0 & 0 & D & B \\ 0 & 0 & D & 0 \\ D & D & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Pseudcode Floyd-Warshall

```
let dist be a |V| × |V| array of minimum distances initialized to ∞ (infinity)
for each edge (u, v) do
    dist[u][v] ← w(u, v)  // The weight of the edge (u, v)
for each vertex v do
    dist[v][v] ← 0
for k from 1 to |V|
    for i from 1 to |V|
        for j from 1 to |V|
            if dist[i][j] > dist[i][k] + dist[k][j]
                dist[i][j] ← dist[i][k] + dist[k][j]
            end if
```


HUST

hust.edu.vn fb.com/dhbkhn

TRÂN TRỌNG CẢM ƠN!