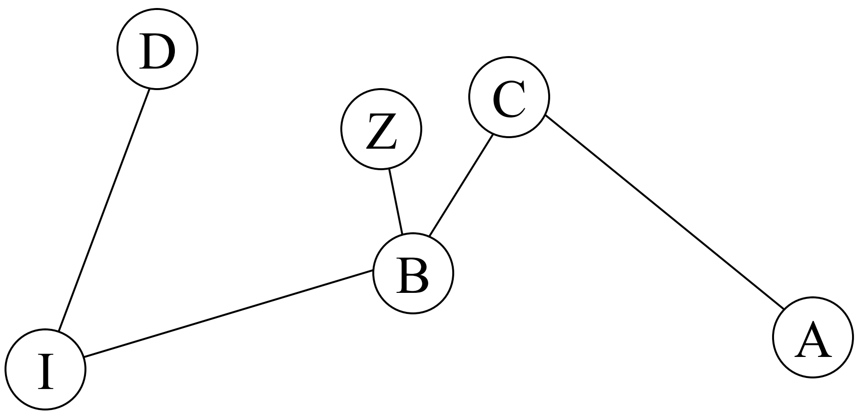
Lý thuyết đồ thị

Nội dung

- Giới thiệu lý thuyết đồ thị
- Cây bao trùm nhỏ nhất
- Cây đường ngắn nhất
- Tour

Đồ thị

- Đồ thị bao gồm một tập các đỉnh (hay nút) V và tập các cạnh E hoặc cung A



Đồ thị (2)

- Cạnh: Cặp đỉnh không sắp xếp thứ tự
- Cung: Cặp đỉnh có sắp xếp thứ tự
- Đồ thị vô hướng: đồ thị có chứa cạnh
- Đồ thị có hướng: đồ thị có chứa cung

Một số định nghĩa

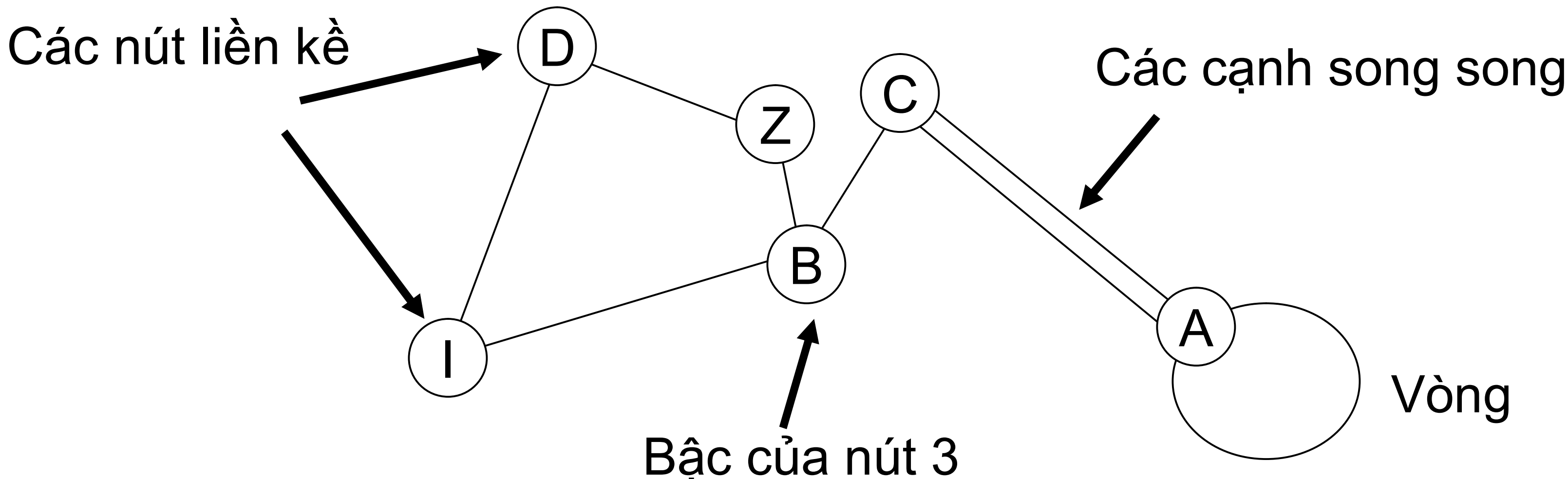
- Các điểm cuối: Tập của một hoặc hai đỉnh của một cạnh
- Vòng (loop): cạnh mà điểm cuối là giống nhau. Còn được gọi là tự lặp
- Cạnh song song: Tập hợp của hai hay nhiều cạnh có cùng hai điểm cuối. Còn được gọi đa cạnh
- Một đồ thị đơn giản là đồ thị không có vòng hay cạnh song song
- Bậc của một đỉnh là số cạnh trong đồ thị có nút đó là một điểm cuối.
- Hai nút gọi là liền kề nếu như có cạnh có nó là điểm cuối.

Một số định nghĩa cho đồ thị (2)

- Đường giữa hai đỉnh v_1 và v_n là tập những cạnh (e_1, e_2, …, e_{n-1}) có e_i và e_{i+1} có cùng điểm cuối và v_1 là điểm cuối của e_1 và v_n là điểm cuối của e_n
- Chu trình là một đường có ít nhất một cạnh từ một đỉnh tới chính bản thân nó.
- Đồ thị liên thông (connected) là đồ thị luôn tồn tại một đường giữa hai nút bất kỳ

S E T

Ví dụ một đồ thị



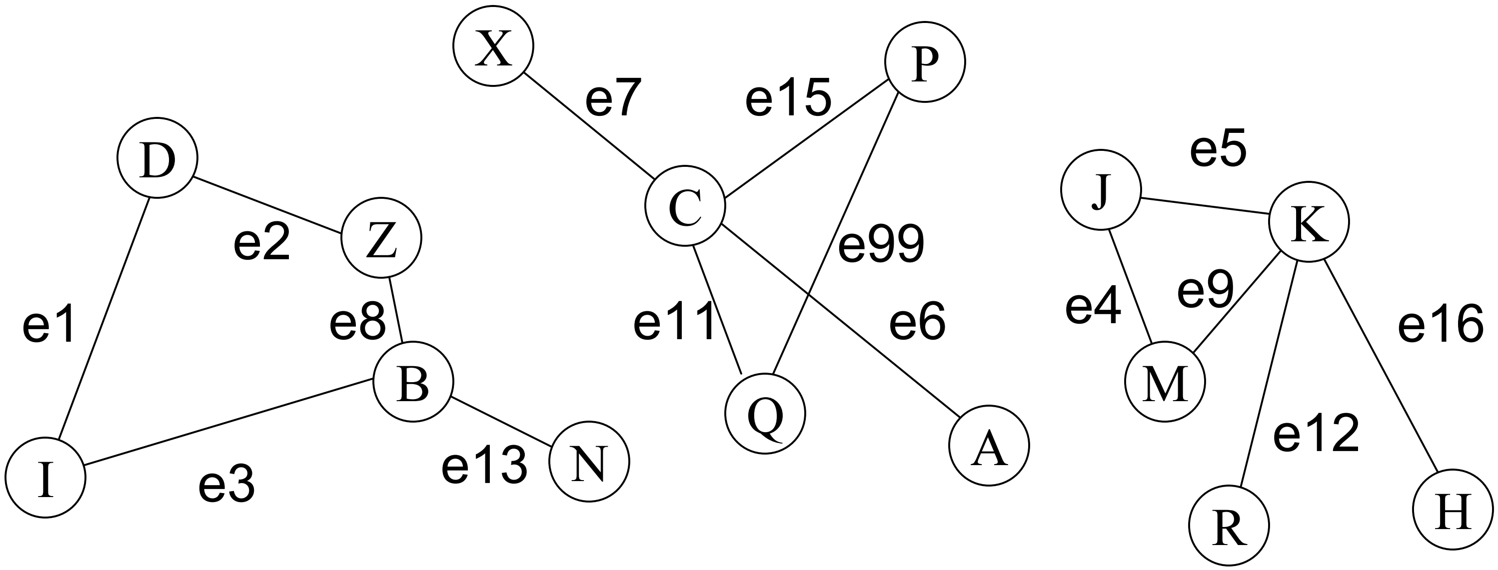
(DZ), (ZB), (BI), (ID) là chu trình

Đồ thị là liên thông

Định nghĩa (3)

- Đồ thị con G* của một đồ thị G với các đỉnh V và các cạnh E có cặp (V*, E*) với
  - V* là tập con của V
  - E* là tập con của E
  - Nếu như 1 cạnh thuộc E* thì cả hai điêm cuối của nó phải thuộc to V*
- Một thành phần (component) của một đồ thị là một đồ thị con liên thông cực đại

Ví dụ đồ thị



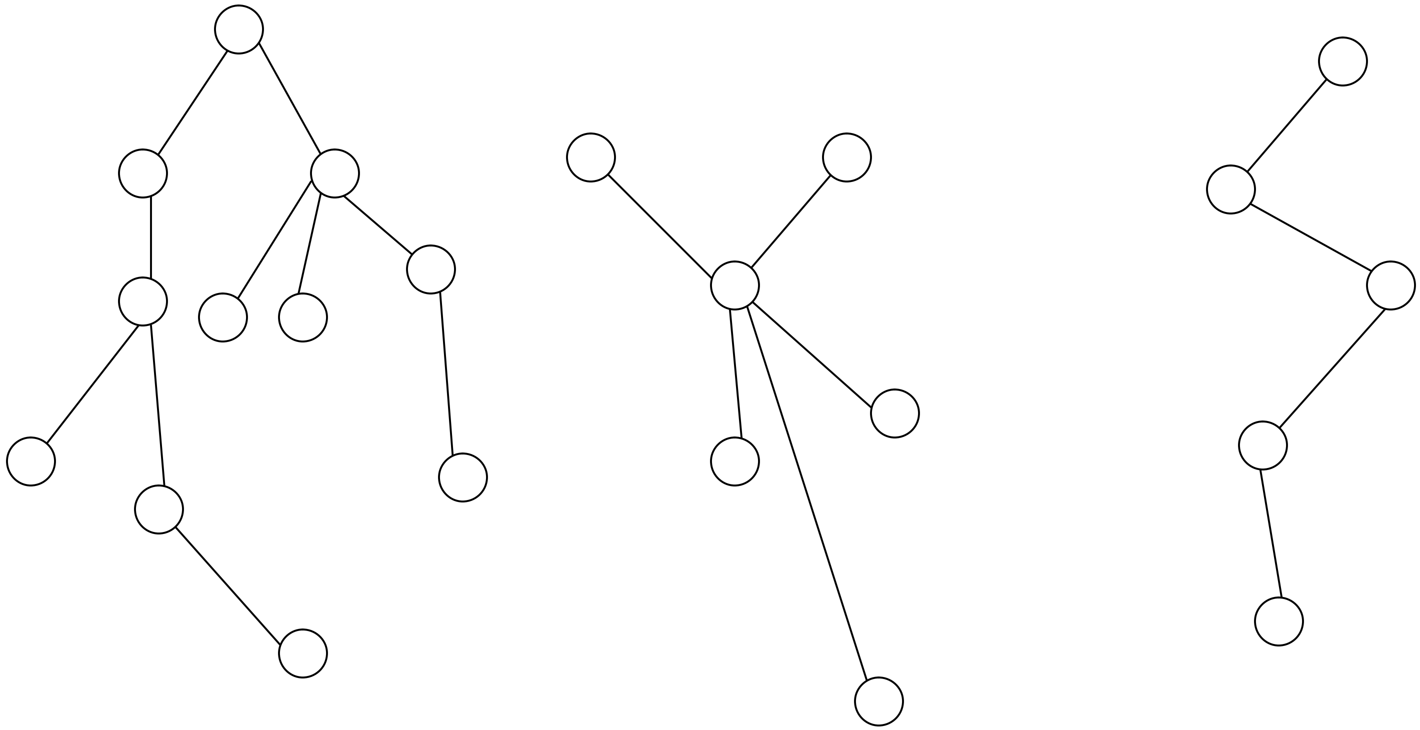
((D, Z, B, M, J), (e2, e8, e4)) là đồ thị con

Nhưng nó không phải là thành phần

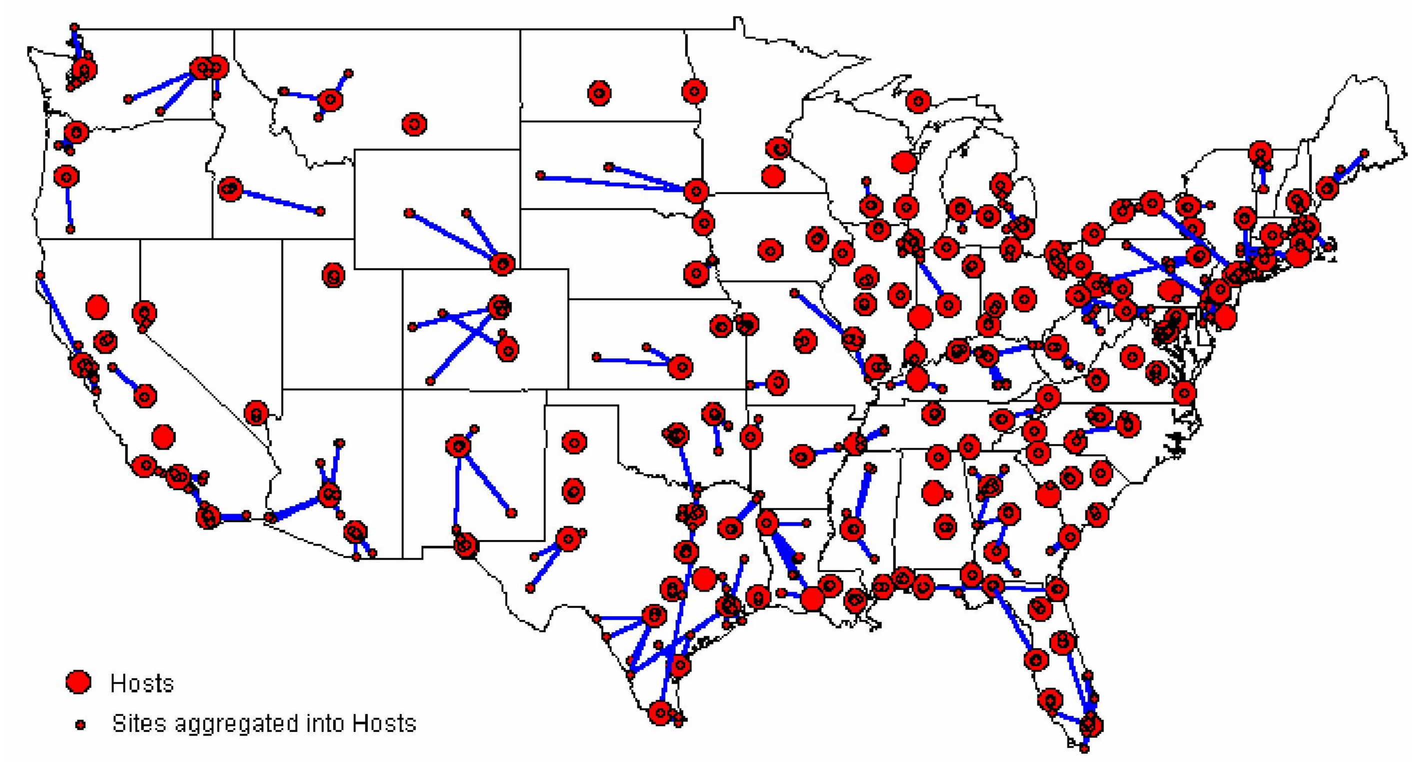
Định nghiã (4)

- Cây là độ thị liên thông đơn giản không có có xích
- Sao là cây mà có duy nhất một nút có bậc lớn hơn 1
- Xích (<u>chain)</u> là cây không có nút nào có bậc lớn hơn 2
- Định nghĩa N(G) = số lượng nút trong G

Cây, Sao, xích

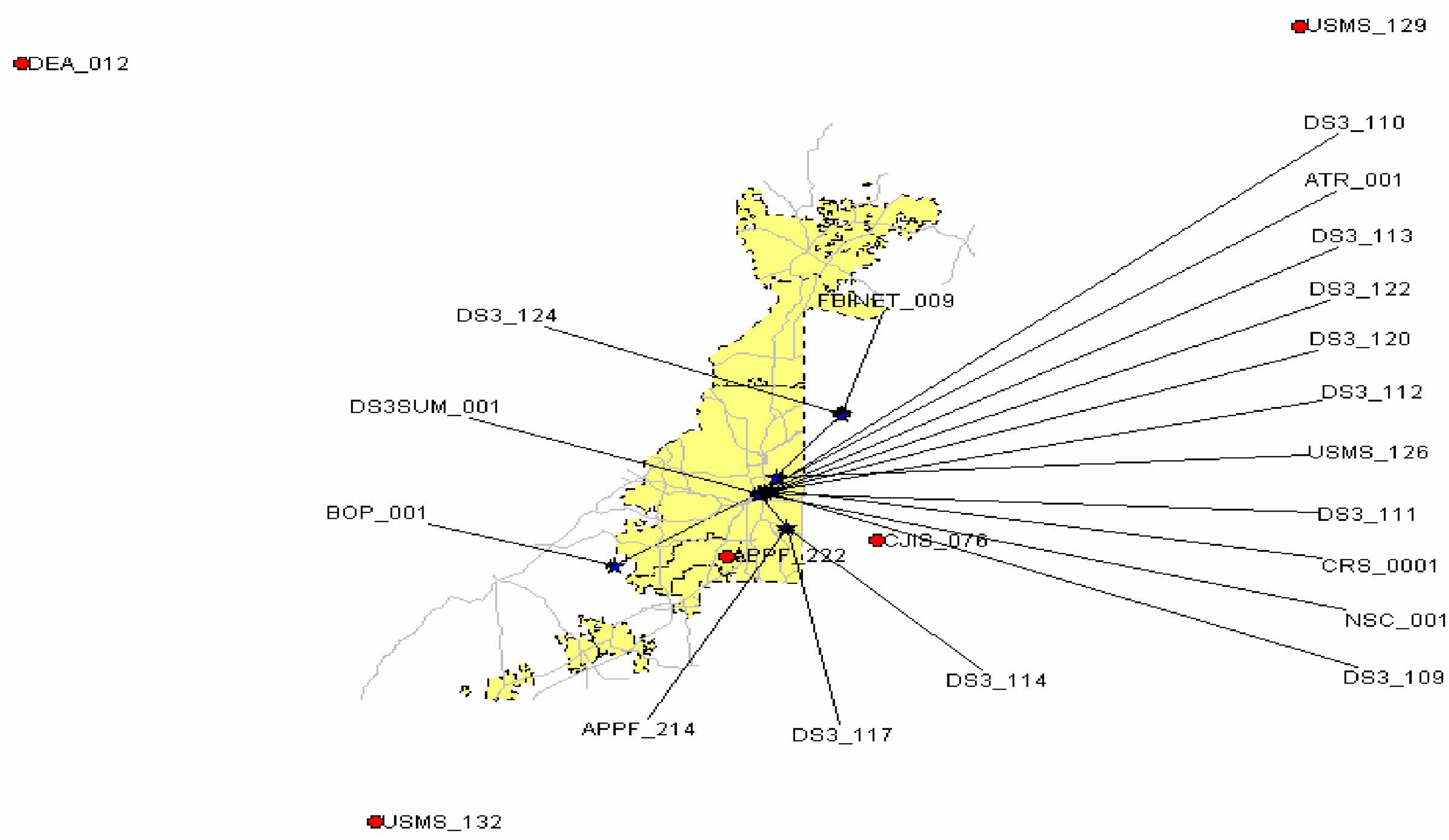


Mạng thực tế (không kể phần Backbone)



Vendor A Cost: $1.159M
Vendor B Cost: $1.213M
164 Hosts

Chi tiết mạng thực tế (Atlanta, GA)



● Stand Alone

★ Hosts

Đồ thị có trọng số

- Đồ thị trọng số là đồ thị G mà mỗi cạnh có một trọng số w(e)
  - Được biểu diễn bằng (G, w)
  - Thường thì w(e) > 0
  - Trọng số của đồ thị con G* là tổng trọng số của của các cạnh trong G*
- Mạng thực tế là các đồ thị có trọng số
  - Trọng số có thể là giá thành, trễ hoặc thông số khác

S E T

Biểu diễn đồ thị

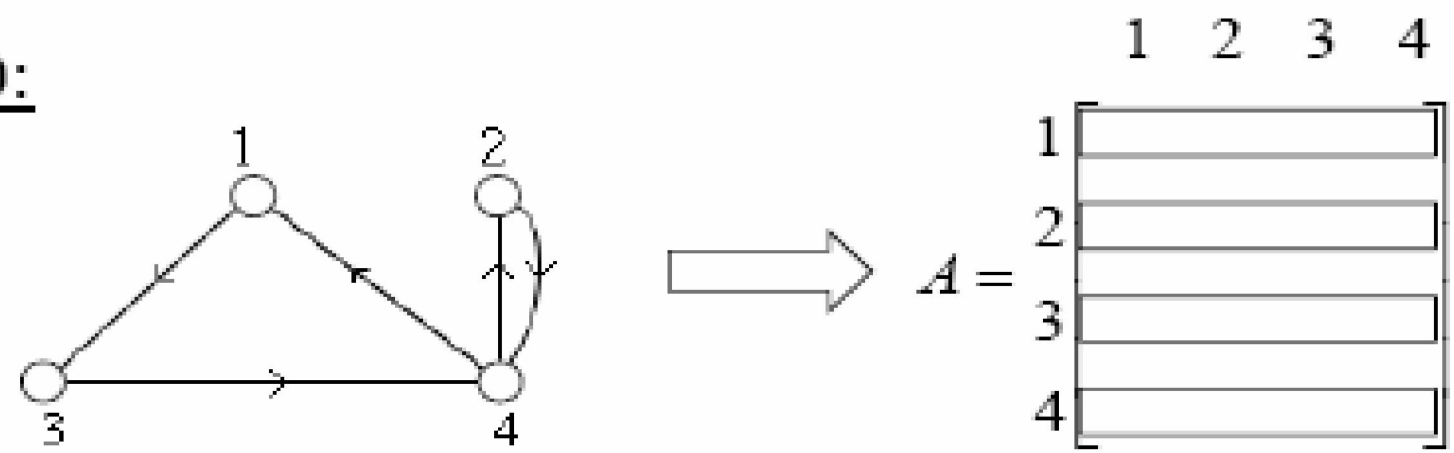
- Ma trận liền kề
- Ma trận liên thuộc
- Danh sách cạnh
- Danh sách liền kề

Ma trận liền kề

- Cho đồ thị G = <V, E>, với V = {v_1, v_2, …, v_n}. Ma trận kề biểu diễn G là một ma trận vuông A, kích thước nxn, được xác định như sau:

$$A_{ij} = \begin{cases} 1, & (v_i, v_j) \in E \\ 0, & (v_i, v_j) \notin E \end{cases}$$

VD:



Ma trận liên thuộc

- Cho đồ thị có hướng G = <V, E>, với $V = \{v_1, v_2, \ldots, v_n\}$, $E = \{e_1, e_2, \ldots, e_m\}$. Ma trận liên thuộc đỉnh – cạnh biểu diễn G là một ma trận A, kích thước nxm, được xác định như sau:

$$A_{ij} = \begin{cases} 1 & v_i \text{ là đỉnh đầu của cạnh } e_j \\ -1 & v_i \text{ là đỉnh cuối của cạnh } e_j \\ 0 & v_i \text{ không là đỉnh đầu, đỉnh cuối của cạnh } e_j \end{cases}$$

<u>Ví dụ:</u>

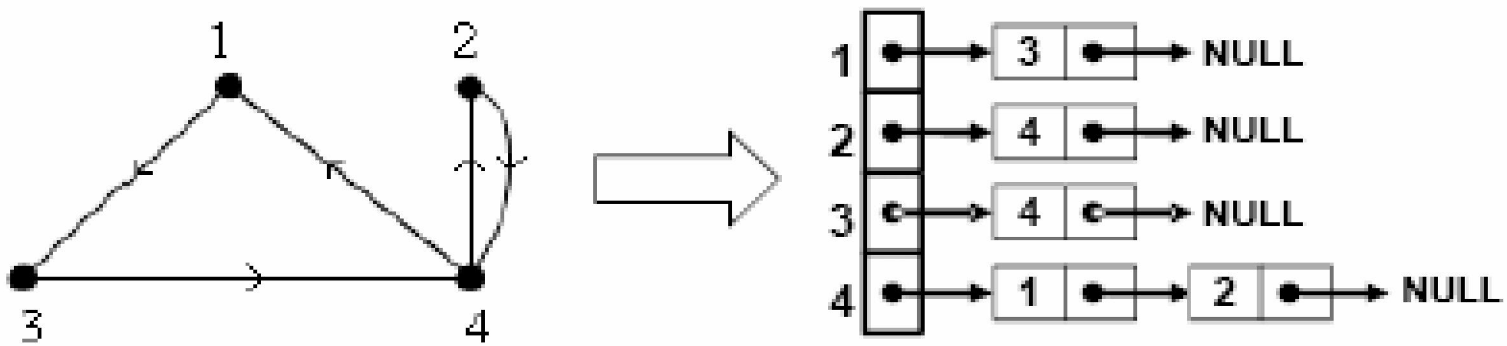
Danh sách cạnh

- Cho đồ thị G=<V,E> có m cạnh. Danh sách cạnh của G sẽ bao gồm hai mảng 1 chiều có kích thước m
- Mảng đầu sẽ lưu các đỉnh đầu của cạnh
- Mảng cuối sẽ lưu các đỉnh cuối của cạnh

Danh sách kề

- Cho đồ thị G = <V,E> có n đỉnh. Đồ thị G có thể được biểu diễn bằng n danh sách liên kết. Mỗi danh sách liên kết thứ i sẽ biểu diễn các đỉnh kề với đỉnh v_i

VD:



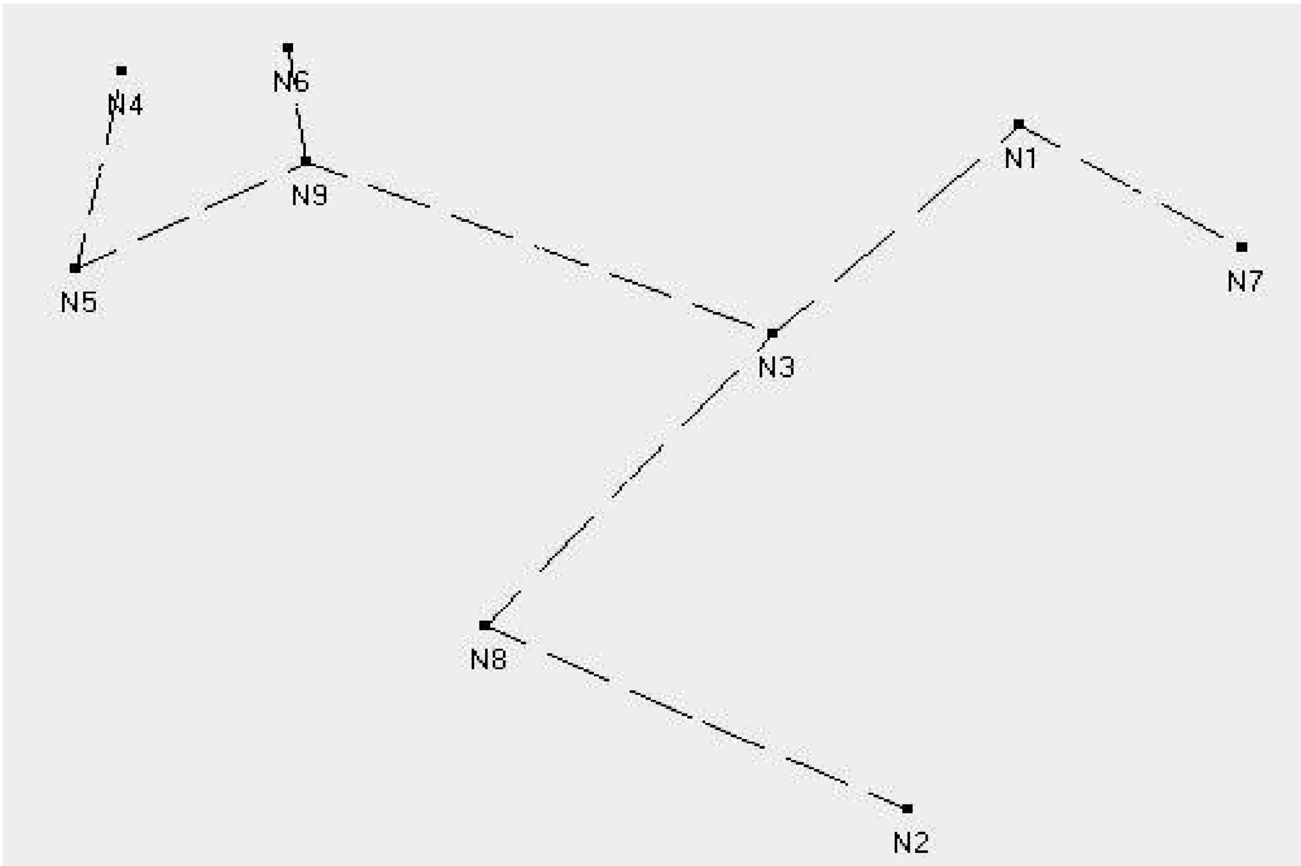
MST (Minimum Spanning Tree)

- 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

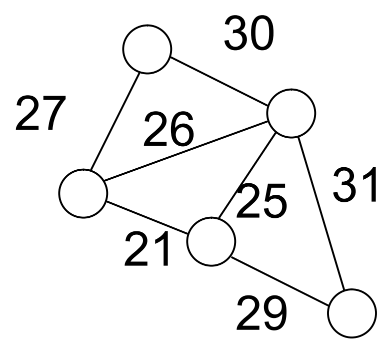
- Bài toán thiết kế nhỏ - vài nút
- Các liên kết có độ tin cạy cao với thời gian không làm việc thấp
  - Hoặc mạng có thể xử lý được sự không tin cạy
- Nút 'v' tin cạy
- Khi số nút tăng lên thì độ tin cạy giảm xuống (theo hàm số mũ)

S E T

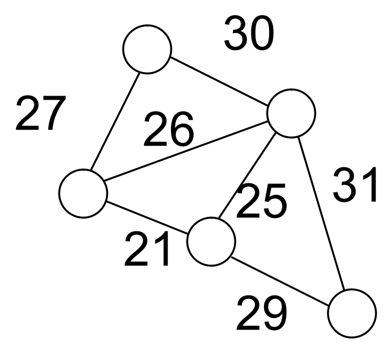
Giải thuật Kruskal (1956)

- 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

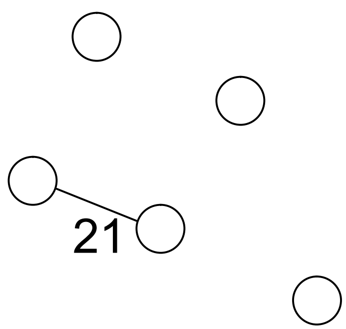
Giải thuật Kruskal cho MST (2)



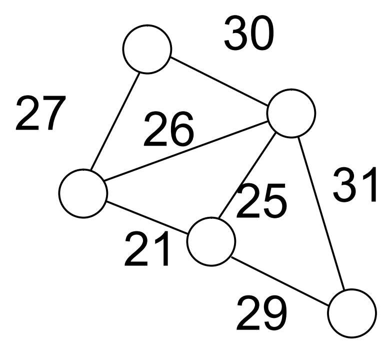
Giải thuật Kruskal cho MST (2)



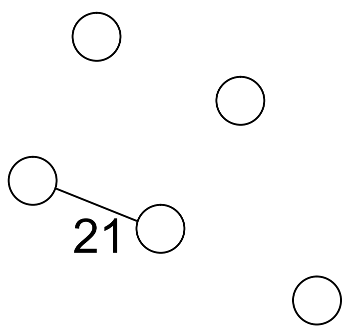
Lần thêm đầu tiên



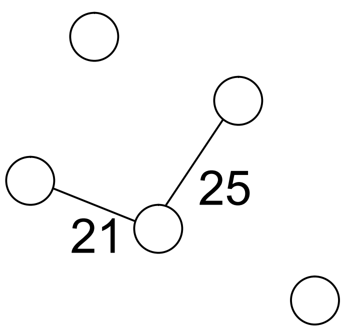
Giải thuật Kruskal cho MST (2)



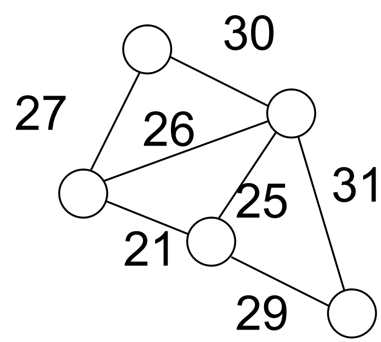
Lần thêm thứ 1



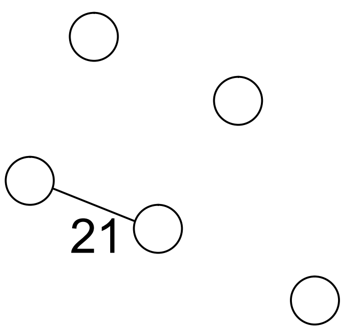
Lần thêm thứ 2



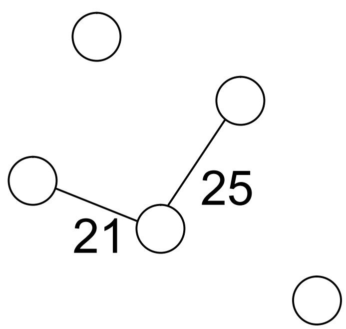
Giải thuật Kruskal cho MST (2)



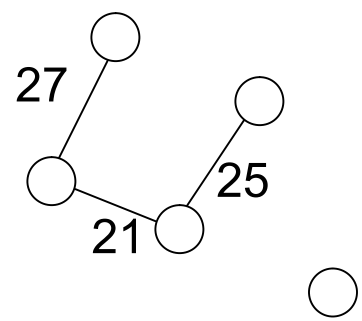
Lần thêm thứ 1



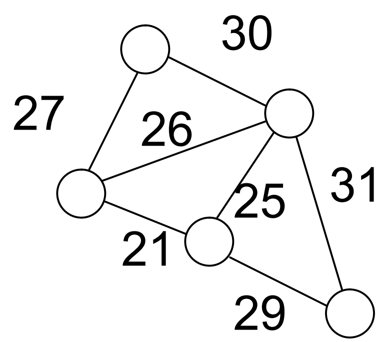
Lần thêm thứ 2



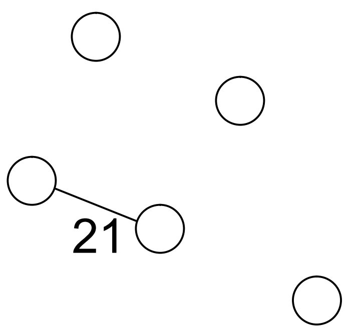
Lần thêm thứ 3



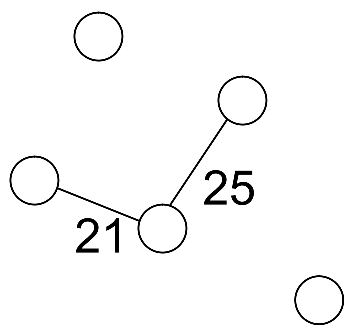
Giải thuật Kruskal cho MST (2)



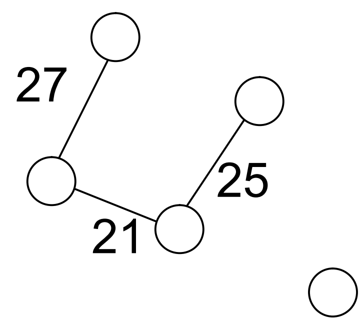
Lần thêm thứ 1



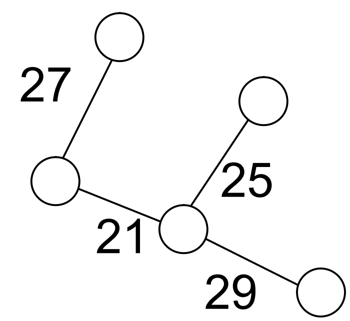
Lần thêm thứ 2



Lần thêm thứ 3



Lần thêm thứ 4



Giải thuật Prim (1957)

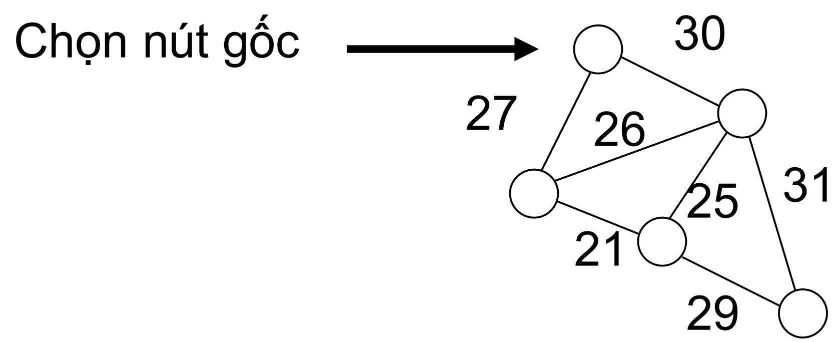
- Đầu vào: đồ thị liên thông có trọng số G=(N,E).
- Đầu ra : Cây bao trùm nhỏ nhất T.

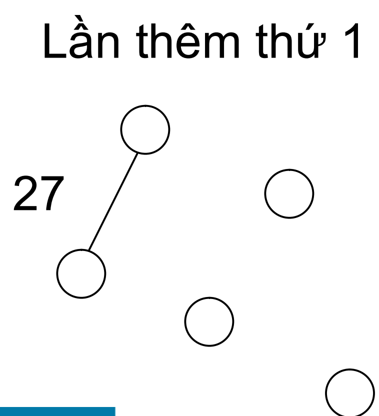
- U = Tập các nút trên MST
- V = Tập tất cả các nút chưa thuộc MST nhưng nó là liền kề những nút thuộc U

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 |.

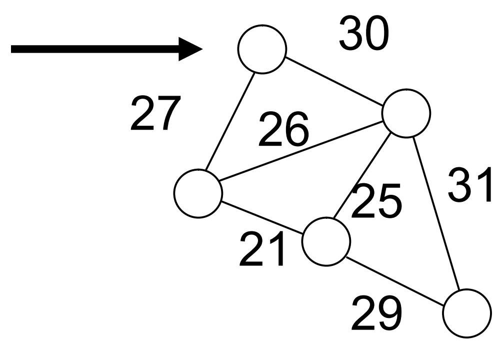
Giải thuật Prim cho MST (2)



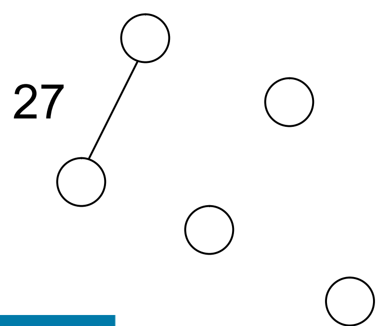


Giải thuật Prim cho MST (2)

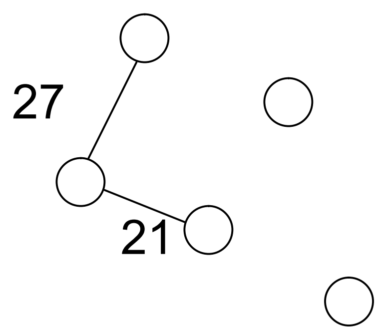
Chọn nút gốc



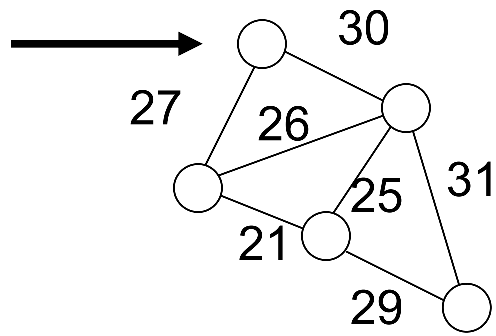
Lần thêm thứ 1



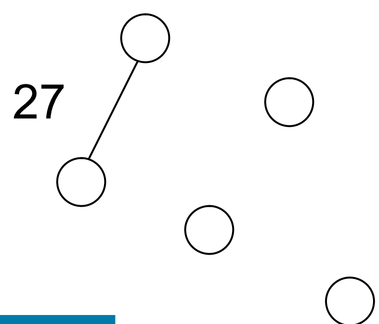
Lần thêm thứ 2



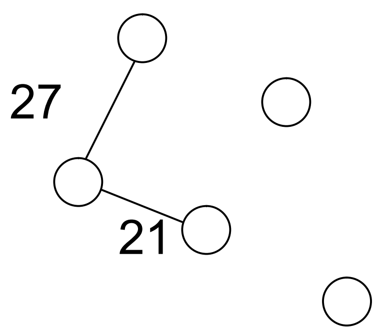
Giải thuật Prim cho MST (2)



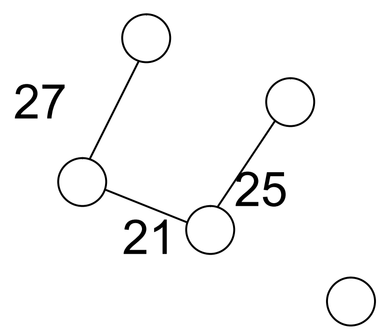
Lần thêm thứ 1



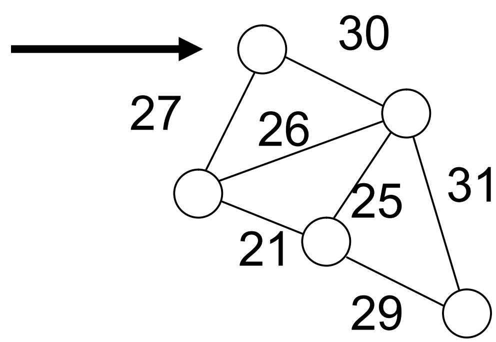
Lần thêm thứ 2



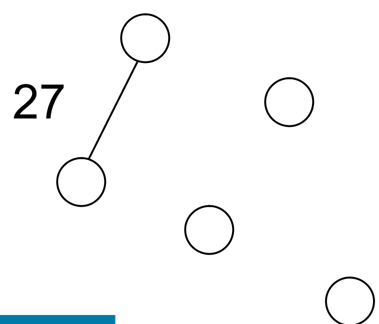
Lần thêm thứ 3



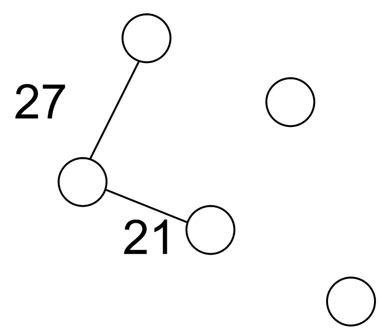
Giải thuật Prim cho MST (2)



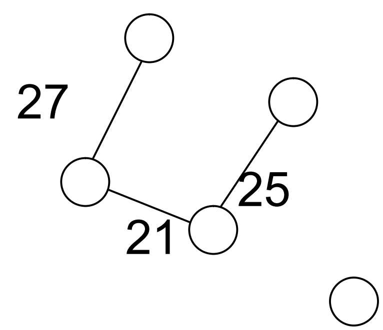
Lần thêm thứ 1



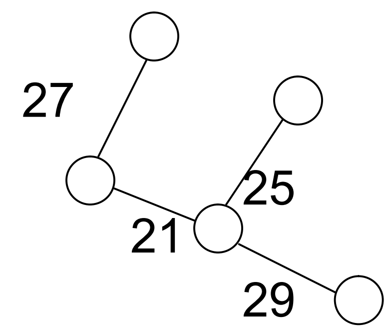
Lần thêm thứ 2



Lần thêm thứ 3



Lần thêm thứ 4



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

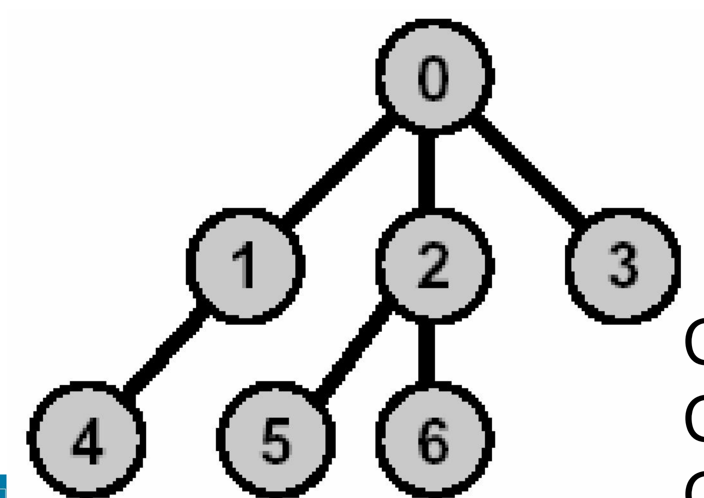
- Cho đồ thị trọng số (G,w), và nút n1 và n2, đường ngắn nhất P từ n1 đến n2 có giá trị $\Sigma_{e \varepsilon P} w(e)$ nhỏ nhất
- Cây đường ngắn nhất shortest-path tree (SPT) có gốc tại nút n1 là cây T mà có đường từ n1 đến n2 là ngắn nhất với bất kỳ nút n2 nào
- Chú ý – không giống như MST, SPT có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



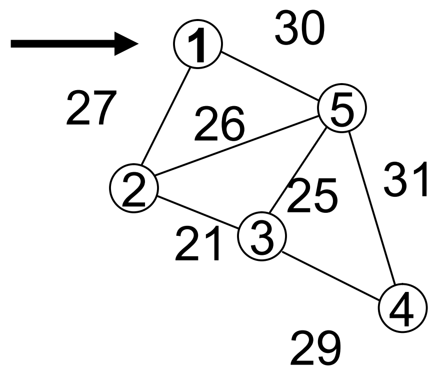
Con cháu(1) = { 4 }
Con cháu (2) = { 5, 6 }
Con cháu (6) = { }

Giải thuật Dijkstra cho SPT

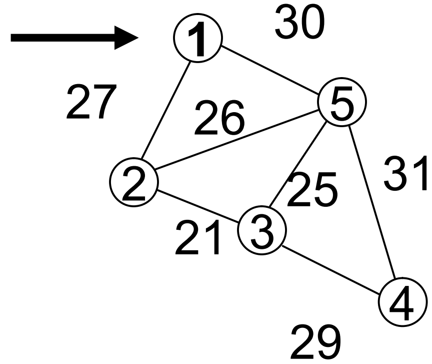
1. Đánh dấu các nút chưa được xét, ấn định nhãn vô cùng
2. Thiết lập nhãn của gốc bằng 0 và thiết lập predecessor(gốc)= nút gốc.
3. Lặp lại
    1. Tìm nút n có nhãn nhỏ nhất Đánh dấu là đã xét
    2. Xem xét tất cả các nút liền kề m, xem nếu khoảng cách qua n < nhãn
        1. Nếu có, cập nhật nhãn,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

Ví dụ Dijkstra 1

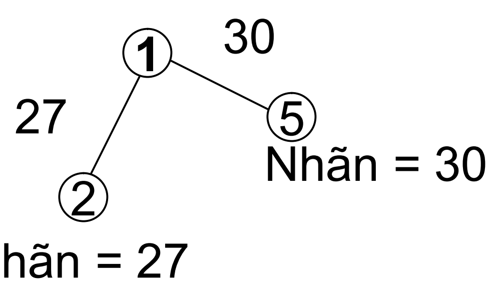
Chọn nút gốc



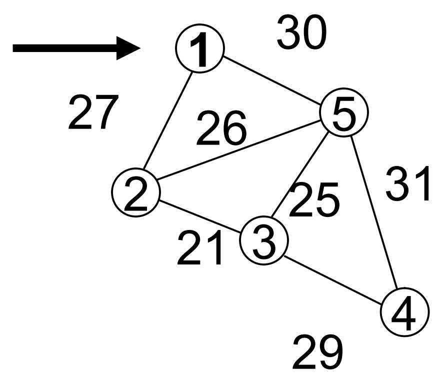
Ví dụ Dijkstra 1



Nút 5 & 2.liền kề với gốc

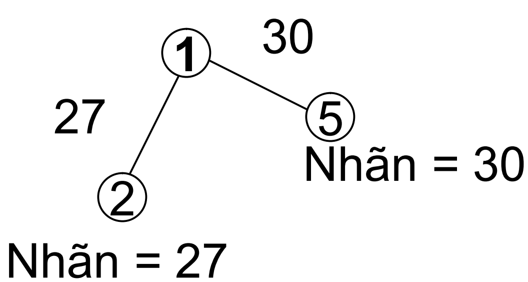


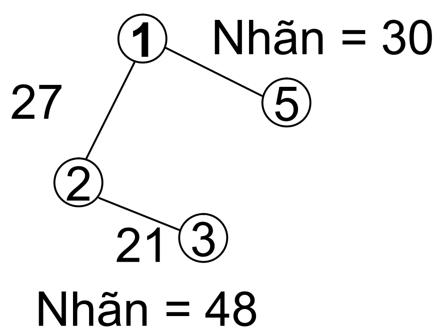
Ví dụ Dijkstra 1



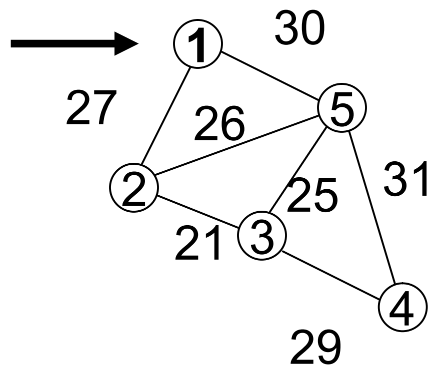
5 & 2 liền kề với gôc

Nút 3 và 5 liền kề với 2





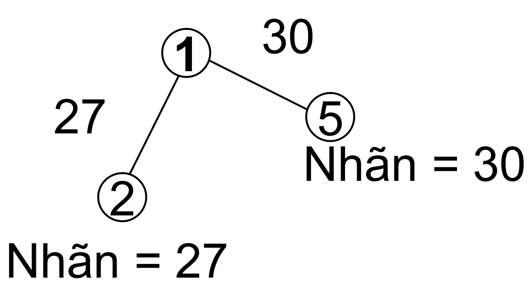
Ví dụ Dijkstra 1

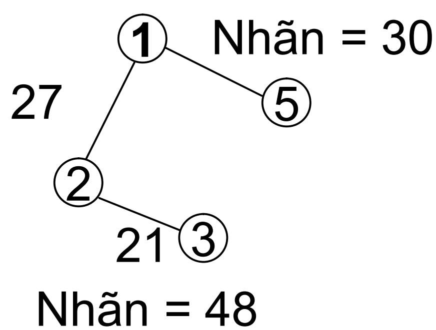


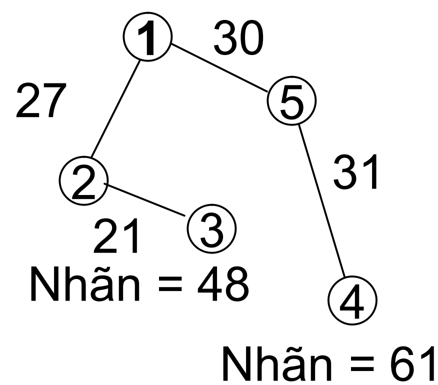
5 & 2 liền kề với gốc

3& 5 liền kề với 2

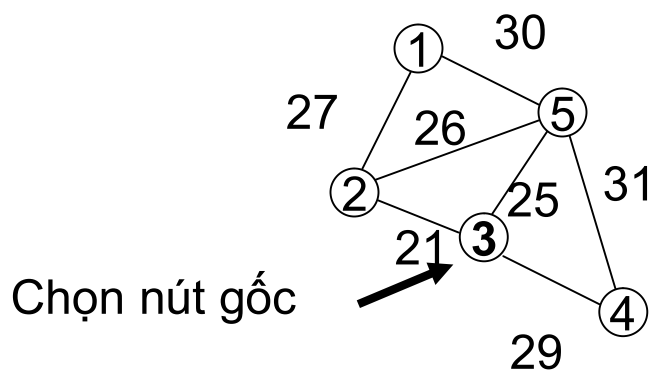
4&3. liền kề với 5



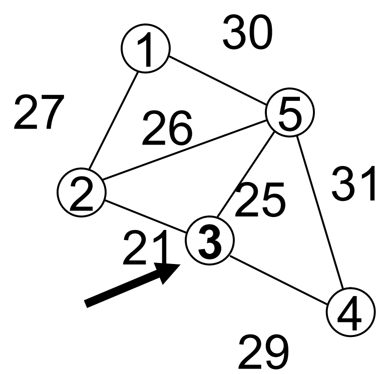




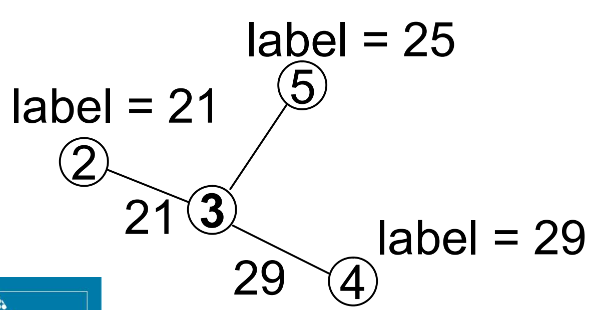
Ví dụ Dijkstra 2



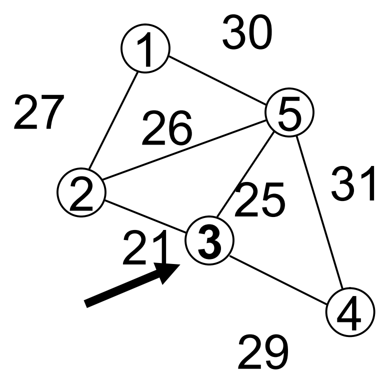
Ví dụ Dijkstra 2



4, 2 & 5 liền kề với nút gốc

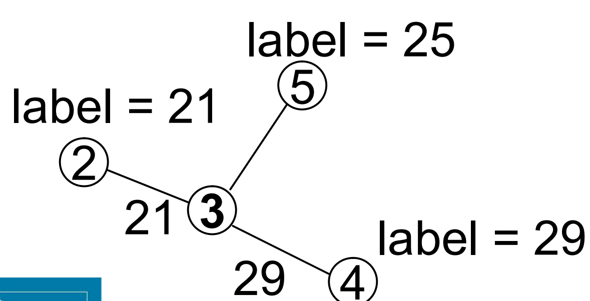


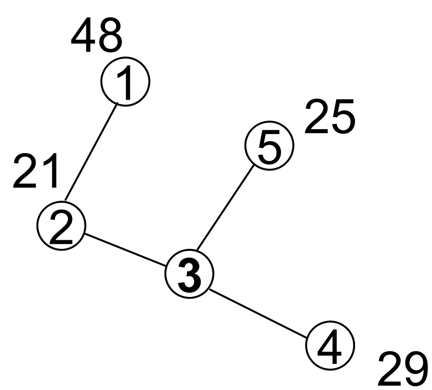
Ví dụ Dijkstra 2



4, 2 & 5 liền kề với nút gốc

1 ,3& 5. liền kề nút 2





Đặc tính của SPT

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

Cây Prim-Dijkstra

- Nhãn Prim
  $= \min_{neighbors}$dist(nút, hàng xóm)
- Nhãn Dijkstra
  $= \min_{neighbors}$[dist(gốc, hàng xóm) + dist(hàng xóm, node)]
- Nhãn Prim-Dijkstra =
  $= \min_{neighbors}[\alpha*$dist(gốc, hàng xóm) + dist(hàng xóm, nút)]
- Trong đó α là tham số có thể được chọn trong khoảng 0 và 1

Giải thuật Bellman –Ford (1)

- Thuật toán trên làm việc với đồ thị có trọng số không âm,hoặc 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. 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. Hàng đợi Q ban đầu chỉ có mình nút gốc.
3. Lặp lại cho đến khi không còn nút trong hàng đợi
    1. Lấy u từ hàng đợi Q
    2. Xem xét tất cả các nút liền kề v, xem nếu khoảng cách qua u < nhãn
        1. 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

Ví dụ giải thuật Bellman -Ford

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

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)

Tours

- Thiết kế cây nhiều khi không tin cậy
- Tour thêm một liên kết vào để tăng độ tin cạy
- Tour là một tập các nút (v1, v2, …, vn) có n cạnh và mỗi nút có bậc là 2 và đồ thị là liên thông

Tours (2)

- Đưa đến bài toán đường đi người bán hàng
  - Cho một tập các nút (v1, v2, …, vn) và hàm khoảng cách d(vi,vj) giữa các nút, tìm tour sao cho $\Sigma d(vt_i,t_{i+1})$ là tối thiểu
- Đây là bài toán khó