ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

TOÁN RỜI RẠC VÀ THUẬT TOÁN

Bài 5
Đồ thị và ứng dụng
(*Graph theory and its applications*)

Nguyễn Thị Hồng Minh
minhnth@gmail.com

Nội dung

1. **Đồ thị và các khái niệm liên quan**
2. **Cài đặt đồ thị**
3. **Đường đi trên đồ thị**
4. **Đường đi ngắn nhất và bài toán người đưa hàng**
5. **Bài tập thực hành**

Chú ý: Hầu hết các hình vẽ trong các bài giảng được sưu tầm từ internet và được trình bày theo quan điểm của giảng viên.

Đồ thị và một số khái niệm liên quan

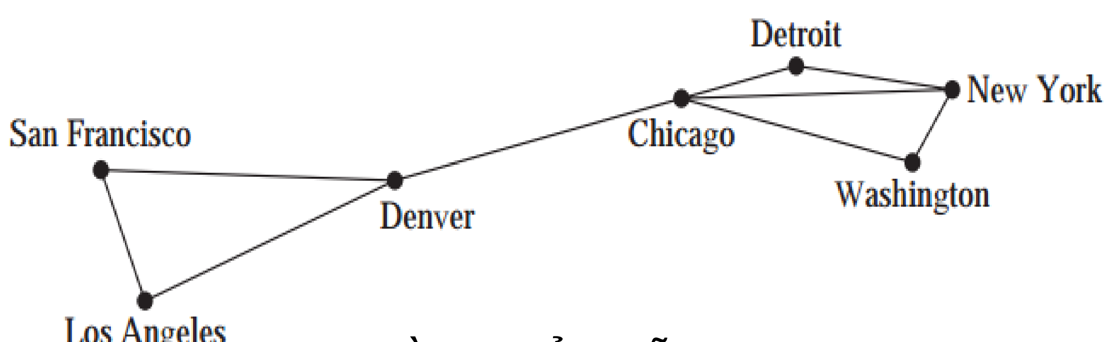
❖ Khái niệm đồ thị

- Là một cấu trúc rời rạc gồm các đỉnh và các cạnh nối các đỉnh đó.
- Mô tả hình thức

  G = (V, E)

  V – tập đỉnh (vertices);

  E – tập cạnh (edges), $E = \{(u,v) \mid u, v \in V\}$

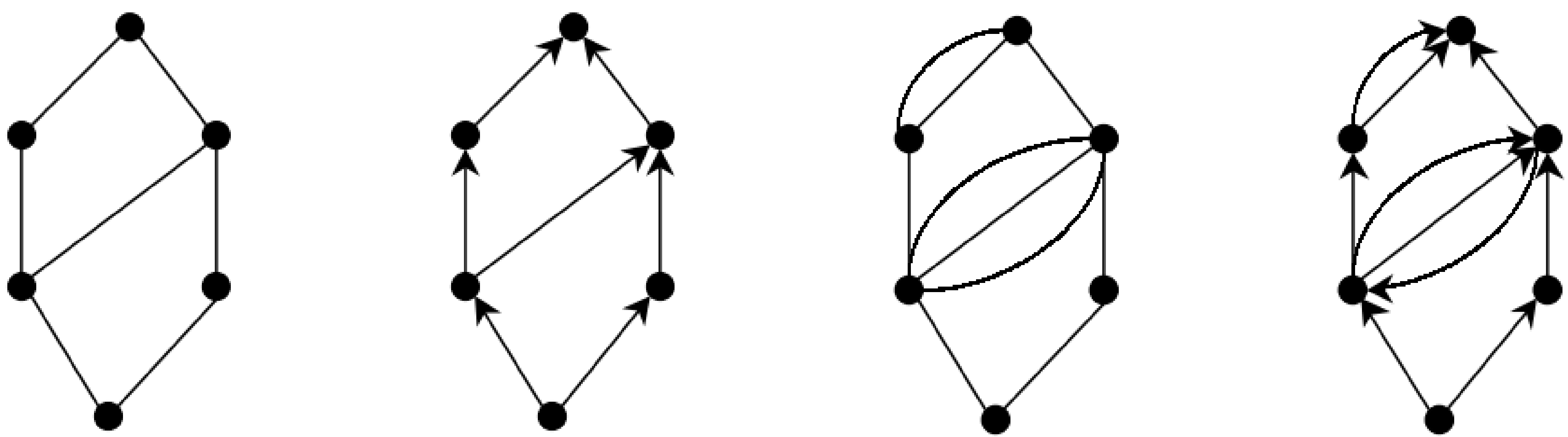


Đồ thị biểu diễn mạng giao thông

Đồ thị và một số khái niệm liên quan

❖ Phân loại đồ thị

- Đơn đồ thị, Đa đồ thị
- Đồ thị vô hướng, đồ thị có hướng



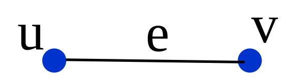
Vô hướng Có hướng Vô hướng Có hướng

Đơn đồ thị Đa đồ thị

Đồ thị và một số khái niệm liên quan

❖ Đô thị vô hướng

G= (V,E),   e = (u,v)



u, v kề nhau (adjacent); e liên thuộc (incident) với u, v

Bậc (degree) của đỉnh bằng số cạnh liên thuộc – *deg*(*u*)

- *Định lí*: Đồ thị G = (V,E) có số cạnh bằng m thì:

$$\sum_{v \in V} \deg(v) = 2m$$

- *Hệ quả*: Số đỉnh bậc lẻ trong đồ thị là số chẵn

Đồ thị và một số khái niệm liên quan

❖ Đô thị có hướng

G= (V,E),   e = (u,v) $u \xrightarrow{e} v$

u tới v; v nối từ u; e là **cung** đi ra từ u; là cung đi vào v

Bán bậc ra ($deg^{+}(v)$) – số cung đi ra từ đỉnh v

Bán bậc vào ($deg^{-}(v)$) – số cung đi vào từ đỉnh v

- *Định lí*: Đồ thị G = (V,E) có số cạnh bằng m thì:

$$\sum_{v\in V} deg^{+}(v) = \sum_{v\in V} deg^{-}(v) = m$$

Đồ thị và một số khái niệm liên quan

- **Một số mô hình đồ thị** (Rosen book – page 644)
  - Mạng xã hội (Social Networks)
  - Mạng kết nối (Communication Networks)
  - Mạng thông tin (Infomation Networks)
  - Ứng dụng thiết kế phần mềm (Software Design Application)
  - Mạng giao thông (Transportation Networks)
  - Mạng sinh học (Biological Networks)
  - Mạng thi đấu (Tournament)

Cài đặt đồ thị

- **Đồ thị có hướng hoặc vô hướng**

  G= (V,E); |V| = n (số đỉnh của đồ thị)

  Đánh số đỉnh 1, 2, ..., n.

Cài đặt đồ thị: Dùng các cấu trúc dữ liệu biểu diễn đồ thị

Tiêu chí:

- Thời gian truy cập thông tin một cạnh (*u*,*v*)
- Thời gian khảo sát tập đỉnh kề với đỉnh *u*
- Độ phức tạp không gian của cấu trúc
- Tính đơn giản trong cài đặt

Cài đặt đồ thị

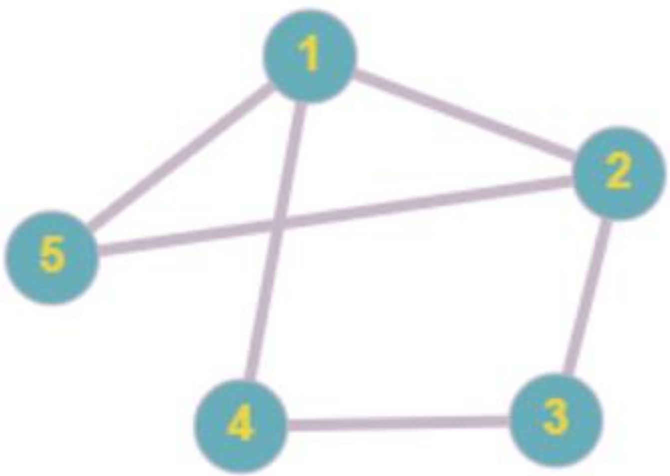
❖ **Cài đặt đồ thị bằng ma trận kề**

Ma trận kề $A = [a_{ij}]_{nxn}$. Trong đó:

$a_{ij} = 1$ nếu $(i, j) \in E$

$a_{ij} = 0$ nếu $(i, j) \notin E$

Quy ước $a_{ii} = 0$ với mọi i;

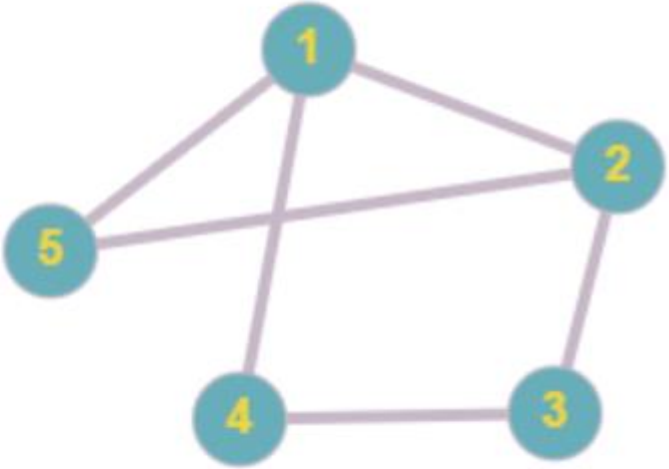


$$A = \begin{bmatrix} 0, & 1, & 0, & 1, & 1 \\ 1, & 0, & 1, & 0, & 1 \\ 0, & 1, & 0, & 1, & 0 \\ 1, & 0, & 1, & 0, & 0 \\ 1, & 1, & 0, & 0, & 0 \end{bmatrix}$$

Cài đặt đồ thị

❖ **Cài đặt đồ thị bằng danh sách cạnh**

Liệt kê các cạnh (u,v)



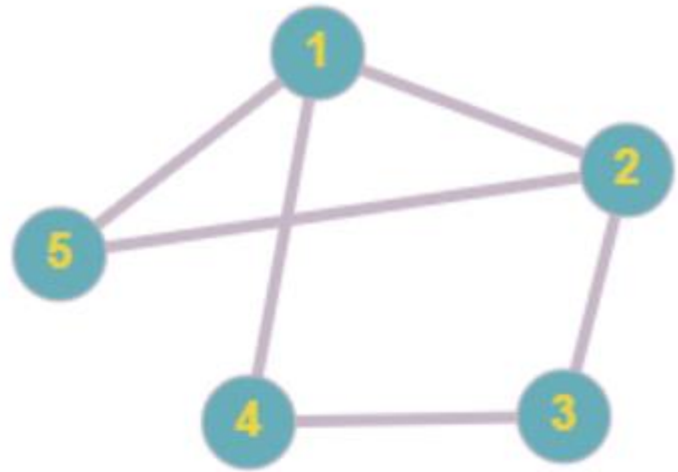
- Sử dụng mảng

1	2	3	4	5	6
(1 , 2)	(1 , 4)	(1 , 5)	(2 , 3)	(2 , 5)	(3 , 4)

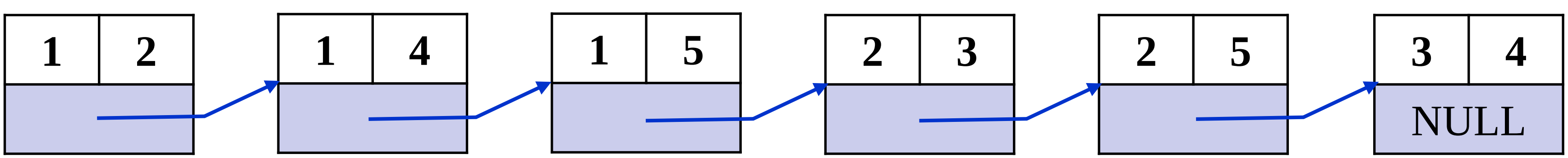
Cài đặt đồ thị

- **Cài đặt đồ thị bằng danh sách cạnh**

Liệt kê các cạnh (u,v)

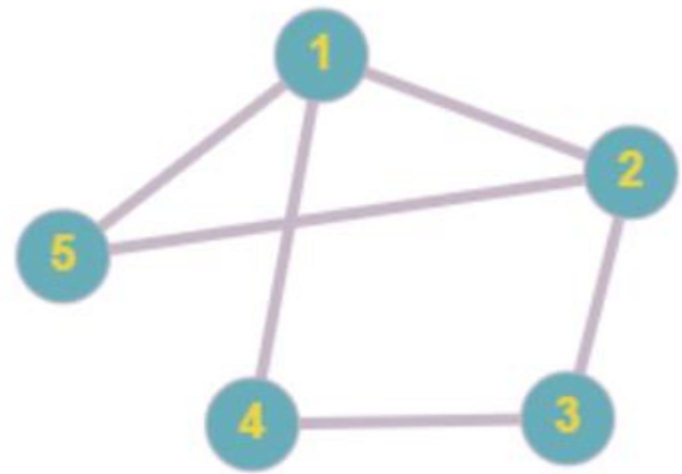


- Danh sách liên kết



Cài đặt đồ thị

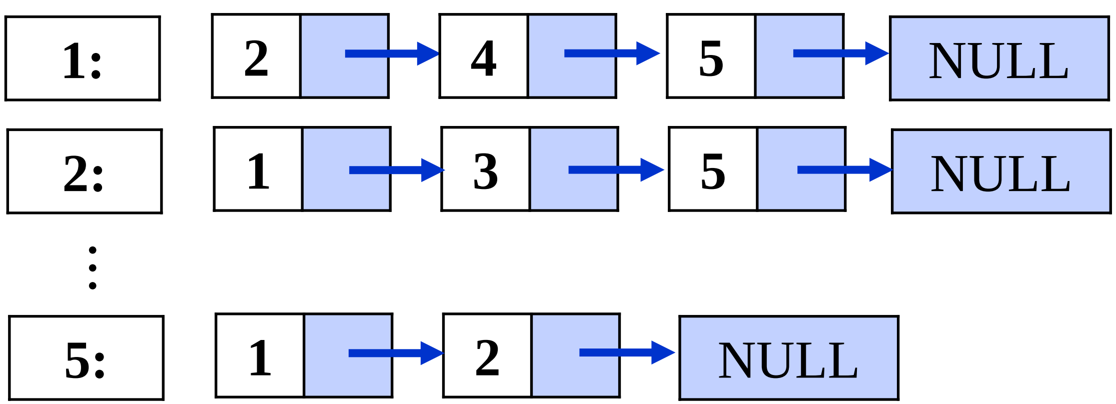
❖ Cài đặt đồ thị bằng danh sách kề



Mảng các danh sách kề đỉnh *i*

(i=1..n)

- Danh sách liên kết



Cài đặt đồ thị

❖ So sánh các cấu trúc cài đặt

Cấu trúc	Ưu điểm	Nhược điểm
Ma trận kề	- Trực quan, dễ cài đặt - Kiểm tra đỉnh kề O(1)	- Không gian lưu trữ $O(n^2)$
Danh sách cạnh	- Không gian lưu trữ O(m) - Duyệt tất cả cạnh (Kruskal)	- Kiểm tra đỉnh kề phải duyệt toàn bộ cạnh.
Danh sách kề	- Tiết kiệm không gian - Dễ duyệt các đỉnh kề đỉnh cho trước - Dễ duyệt các cạnh	- Cài đặt phức tạp

Một số khái niệm khác

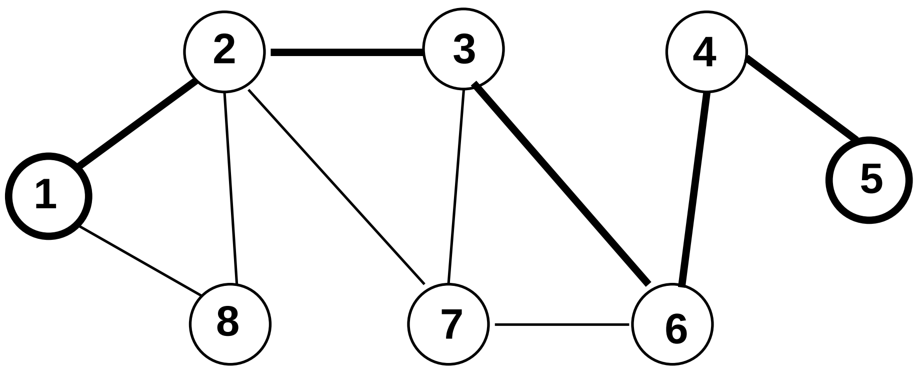
- Đồ thị đơn (Simple Graph)
- Đồ thị liên thông (Conneted Graph)
- Đồ thị hai phía (Bipartite Graph)
- Sắc số (Chromatic number)
- Cây bao trùm tối thiểu (Spanning tree)
- Đường đi trong đồ thị (path)
- Chu trình Euler (Euler Circuit)
- Chu trình Hamilton (Hamiltonian Circuit)

Một số thuật toán

- Đường đi trong đồ thị
- Xác định thành phần liên thông
- Tô màu đồ thị
- Xác định cây bao trùm tối thiểu
- Chu trình Euler
- Chu trình Hamilton
- Tìm luồng cực đại

Đường đi trên đồ thị

❖ Bài toán



- Đỉnh xuất phát: $S=1 \in V$, đỉnh kết thúc $F=5 \in V$
- Đường đi tìm được là dãy $S=1\text{-}2\text{-}3\text{-}6\text{-}4\text{-}5=F$

Đường đi trên đồ thị

- **Sử dụng phương pháp quay lui**
  - Mô tả dữ liệu
    - Đánh chỉ số các đỉnh của đồ thị từ 1..*n*
    - Biểu diễn đồ thị G bằng ma trận kề $M = (m_{ij})$ cỡ *n* x *n*

      m_{ij}= 1 nếu có cạnh nối đỉnh *i* với đỉnh *j*

      = 0 nếu ngược lại
    - Mảng: Daqua[1..*n*] đánh dấu đỉnh *i* đã được đi qua trên đường đi hay chưa?

      Daqua[*i*]= *true* nếu đỉnh *i* đã có trên đường đi

      = *false* nếu ngược lại, đỉnh *i* chưa có trên đường đi

      Khởi tạo: Daqua[1..*n*] = *false*

Đường đi trên đồ thị

❖ Lược đồ thuật toán

```
Try(i) ≡
    for (v=1..n)    //duyệt qua các đỉnh
        if ((m[x_{i-1},v]=1)&(not Daqua[v])) //v chấp nhận được
            x_i = v;
            Daqua[v] = true; //ghi nhận trạng thái đã chọn v
            if (x_i= F) printResult (x_1,x_2,…,x_i);
            else Try(i+1) ;
            Daqua[v] = false; //khôi phục trạng thái chưa chọn v
        endif;
    endfor;
End.
```

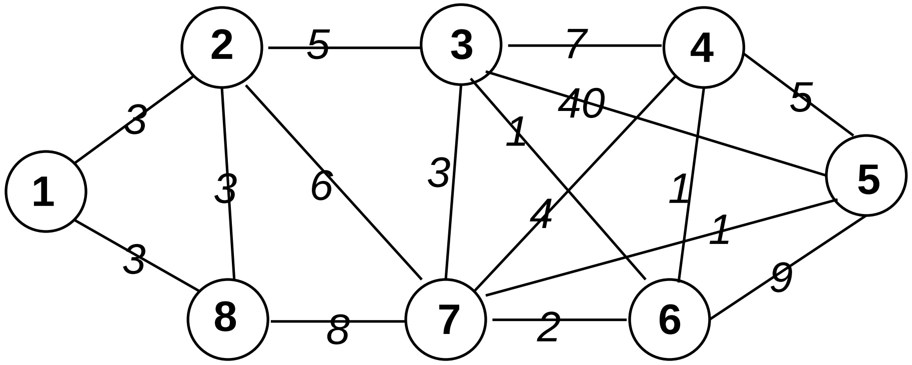
Lời gọi ban đầu: `x_1=S;`

`Try(2);`

Đường đi ngắn nhất trên đồ thị

- **Bài toán người đưa hàng (Travelling Salesman Problem – TPS):**

Một người bán hàng trên hệ thống *n* thành phố. Giữa các thành phố có thể có hoặc không các đường nối, mỗi đường nối có chi phí xác định từ trước. Người bán hàng xuất phát từ một thành phố, đi tới tất cả các thành phố khác và mỗi thành phố đi qua một lần và quay trở lại thành phố ban đầu. Xác định một hành trình sao cho tổng chi phí trên đường đi là nhỏ nhất.



Đường đi ngắn nhất trên đồ thị

- **Phân tích bài toán**

Mạng lưới giao thông giữa các thành phố như một đồ thị có trọng số $G=(V,E)$

- o Mỗi thành phố là một nút của đồ thị (đánh số 1,..,n)
- o Mỗi đường đi giữa các thành phố là một cạnh nối giữa các nút của đồ thị, có thể có hướng hoặc vô hướng, trên đó có ghi trọng số là chi phí đường đi

  Các cặp cạnh không có đường đi trọng số là ∞
- o Đỉnh xuất phát ≡ kết thúc: $S \in V$
- o Đường đi tìm được là dãy $S=x_1,x_2,\ldots,x_n,x_1=S$ với $x_i \in V$, $(x_i,x_{i+1}) \in E$, có tổng chi phí nhỏ nhất

⇒ Sinh các dãy hoán vị 1..n và tính dãy có chi phí nhỏ nhất:

- Quay lui
- Nhánh cận

Đường đi ngắn nhất trên đồ thị

- **Sử dụng phương pháp nhánh cận**
  - Chi phí tốt nhất đã tìm được (*BestCost*). Ban đầu *BestCost*= $+\infty$
  - Tại mỗi bước chọn x_i : Chi phí đường đi từ x_1 đến x_{i-1} là C
    - Với mỗi khả năng v, tính chi phí $C_1 = C$ + chi phí từ x_{i-1} tới v
    - Nếu C_1 xấu hơn (lớn hơn) *BestCost* hoặc không có khả năng nào chấp nhận được cho x_i thì lùi lại bước trước để xác định lại thành phần x_{i-1}.
    - Nếu C_1 tốt hơn (nhỏ hơn) *BestCost* thì chấp nhận x_i theo khả năng v. Tiếp tục xác định x_{i+1},…
    - Đến khi gặp nghiệm ($i=n+1$ & $x_i=S$):
      - Cập nhật đường đi tốt nhất hiện tại đã tìm được
      - Cập nhật giá trị *BestCost* mới: *BestCost* = C_1
  - Kết thúc tìm kiếm nếu *BestCost*= $+\infty$ => không có đường đi

Đường đi ngắn nhất trên đồ thị

- **Sử dụng phương pháp nhánh cận**
  - Mô tả dữ liệu
    - Đánh chỉ số các đỉnh của đồ thị từ 1..*n*
    - Biểu diễn đồ thị G bằng ma trận kề $M = (c_{ij})$ cỡ n x n

      c_{ij} = *cost* nếu có cạnh nối đỉnh *i* với đỉnh *j* với chi phí *cost*

      = ∞ nếu không có đường đi
    - Mảng: Daqua[1..*n*] đánh dấu đỉnh *i* đã được đi qua trên đường đi hay chưa?

      Daqua[*i*]= *true* nếu đỉnh *i* đã có trên đường đi

      = *false* nếu ngược lại, đỉnh *i* chưa có trên đường đi

      Khởi tạo: Daqua[1..*n*] = *false*

Đường đi ngắn nhất trên đồ thị

- **Lược đồ thuật toán**

```
Try(i,C)≡  //Sinh thành phần thứ i của cấu hình với chi phí hiện thời C
    for (v = 1..n)
      if (c[x_{i-1},v]<∞)&(not Daqua[v])) )
          C1 = C + c[x_{i-1},v];
          if (C1 < BestCost)  //Tốt hơn chi phí tốt nhất hiện có
              x_i = v ;
              Daqua[v] = true;
              if (i = n+1)&(x_i = S)
                  <Ghi nhận nghiệm x_1,x_2…x_{n+1}>;
                  BestCost = C1;  //Cập nhật chi phí tốt nhất
              else if (i<=n)
                   Try(i+1,C1);  //sinh thành phần tiếp theo với chi phí hiện thời C1
              endif;
              Daqua[v] = false;
          endif;
      endif;
End.
```

Đường đi ngắn nhất trên đồ thị

- **Sử dụng phương pháp tham lam (thuật toán Dijkstra)**

Rosen-Ed7 book page 712

procedure *Dijkstra*(G: weighted connected simple graph, with
    all weights positive)
{G has vertices $a = v_0, v_1, \ldots, v_n = z$ and lengths $w(v_i, v_j)$
    where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in G}
for $i := 1$ **to** n
    $L(v_i) := \infty$
$L(a) := 0$
$S := \emptyset$
{the labels are now initialized so that the label of a is 0 and all
    other labels are ∞, and S is the empty set}
while $z \notin S$
    $u :=$ a vertex not in S with $L(u)$ minimal
    $S := S \cup \{u\}$
    for all vertices v not in S
        if $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
        {this adds a vertex to S with minimal label and updates the
        labels of vertices not in S}
return $L(z)$ {$L(z) =$ length of a shortest path from a to z}

Đường đi ngắn nhất trên đồ thị

- **Một số thuật toán tìm đường đi**

Bài tập thực hành

- **Phát triển ý tưởng bài toán sử dụng thuật toán tìm đường đi**
  - Phát biểu bài toán
  - Mô hình bài toán
  - Phương pháp giải và lược đồ thuật toán
  - Gợi ý:
    - Hệ thống phân phối hàng của chuỗi siêu thị Vinmart
    - Hệ thống cấp tiền cho các cây ATM của ngân hàng BIDV
    - Hệ thống đưa đón học sinh cho trường phổ thông dân lập Nguyễn Siêu
    - ...