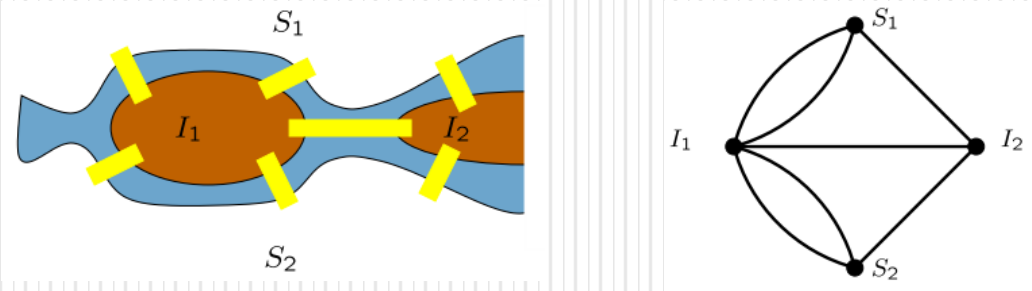
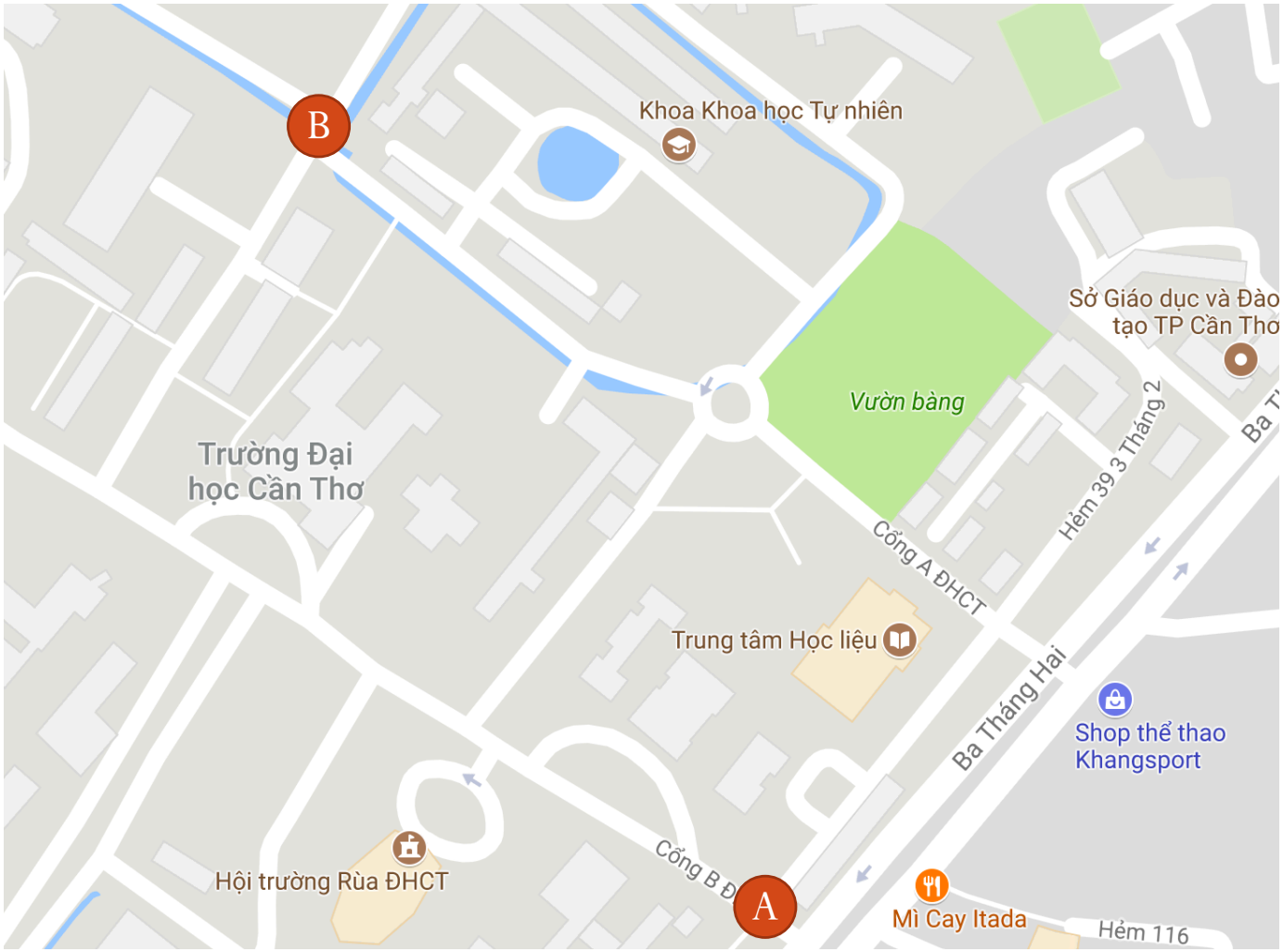
LÝ THUYẾT ĐỒ THỊ
Tìm đường đi ngắn nhất

Phạm Nguyên Khang
BM. Khoa học máy tính, CNTT
pnkhang@cit.ctu.edu.vn



Cần Thơ, 2012

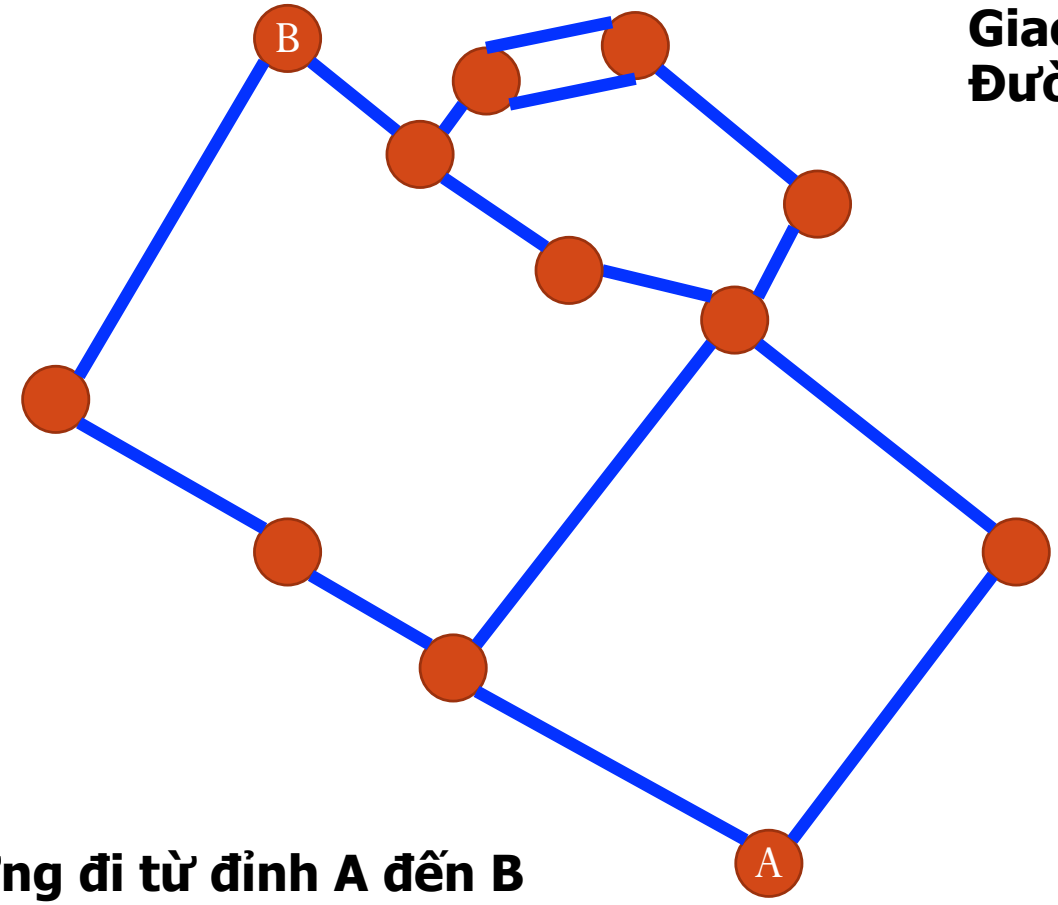
Từ bản đồ đến đồ thị



Từ bản đồ đến đồ thị

Từ bản đồ đến đồ thị

Từ bản đồ đến đồ thị

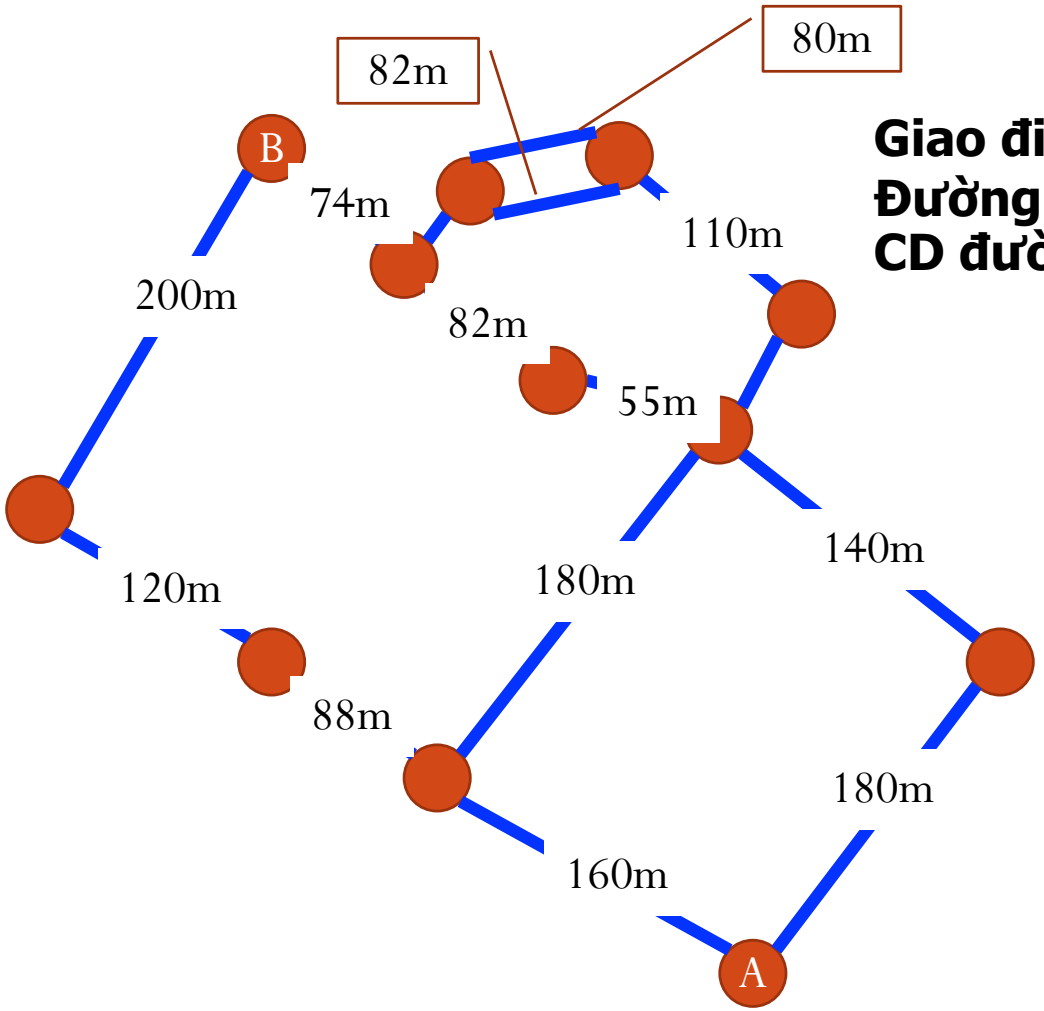


Giao điểm => Đỉnh
Đường => Cung

Tìm đường đi từ đỉnh A đến B

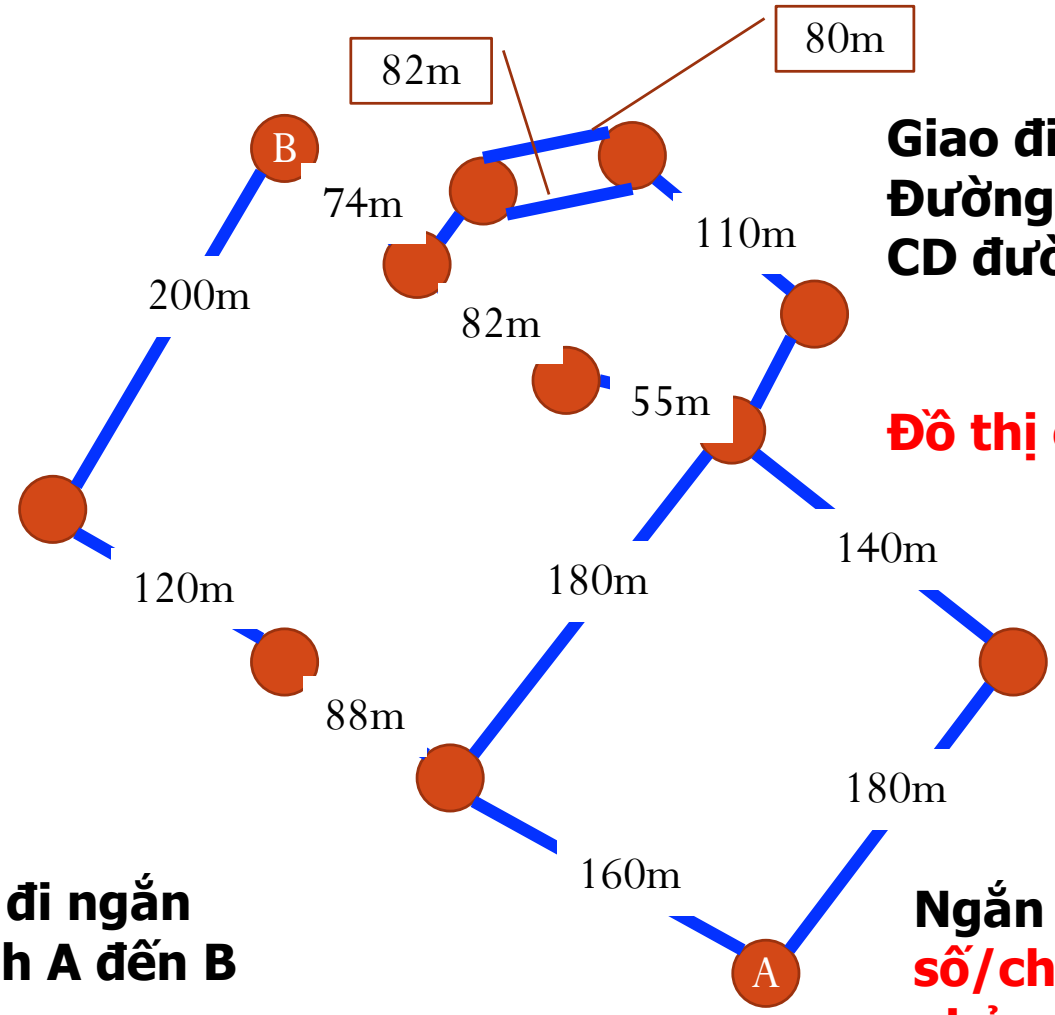
Từ bản đồ đến đồ thị

Từ bản đồ đến đồ thị



Giao điểm => Đỉnh
Đường => Cung
CD đường => ?

Từ bản đồ đến đồ thị



Giao điểm => Đỉnh
Đường => Cung
CD đường => Trọng số/ CD cung

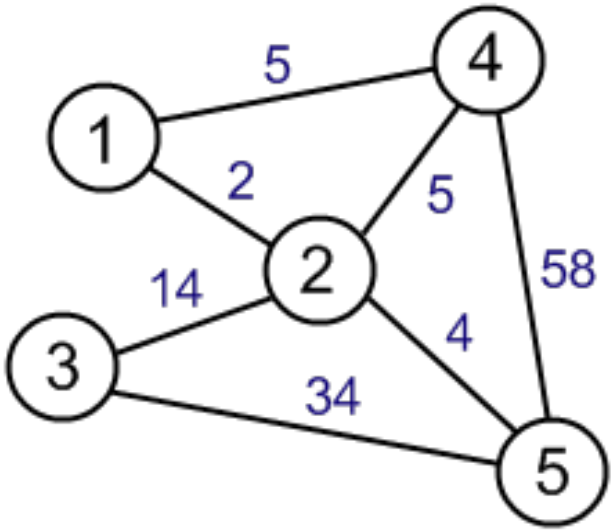
Đồ thị có trọng số

Tìm đường đi ngắn nhất từ đỉnh A đến B

Ngắn nhất: Tổng trọng số/chiều dài các cung nhỏ nhất

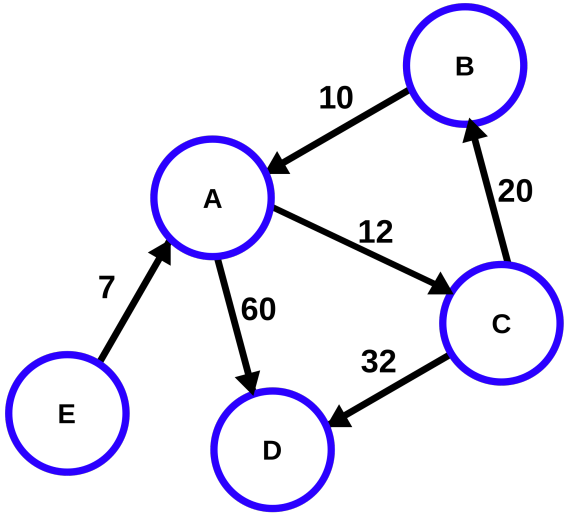
Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
  - Ma trận kề (đỉnh - đỉnh) => **ma trận trọng số**
    - Nếu có (u, v) thì **L[u, v] = trọng số của cung (u, v)**
    - Nếu không có (u, v) thì **L[u, v] = NO_EDGE** (vd: -1)



Biểu diễn đồ thị có trọng số

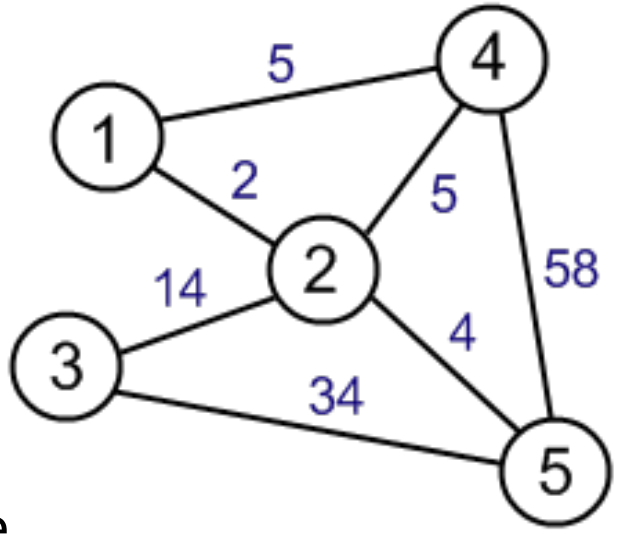
- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
  - Ma trận kề (đỉnh - đỉnh) => **ma trận trọng số**
    - Nếu có (u, v) thì **L[u, v] = trọng số của cung (u, v)**
    - Nếu không có (u, v) thì **L[u, v] = NO_EDGE** (vd: -1)



Biểu diễn đồ thị có trọng số

- Có thể mở rộng các phương pháp biểu diễn đồ thị trước đây để biểu diễn đồ thị có trọng số
  - Danh sách cung
    - Mỗi cung lưu trữ: đỉnh đầu, đỉnh cuối và trọng số

```
typedef struct {
    int u, v;
    int/float/double w;
} Edge;
```



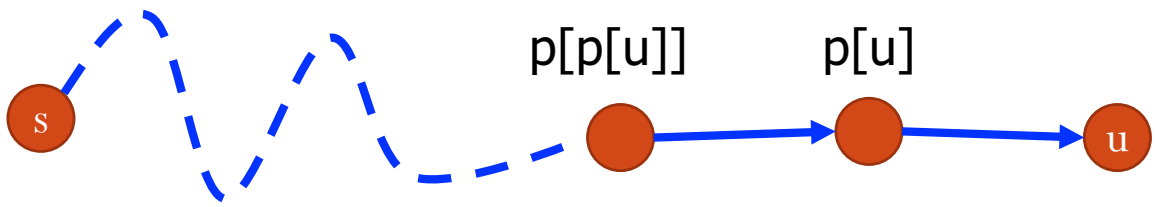
Graph = Danh sách các Edge

Giải thuật Moore-Dijkstra

- Tìm đường đi ngắn nhất từ 1 đỉnh đến các đỉnh khác trên đồ thị có trọng số
- Điều kiện áp dụng:
  - Đồ thị có trọng số không âm
  - Có hướng hoặc vô hướng đều được
- Ý tưởng chính:
  - Khởi tạo đường đi trực tiếp.
  - Lần lượt cập nhật lại đường đi nếu **tìm được đường đi mới tốt hơn đường đi cũ**.

Giải thuật Moore-Dijkstra

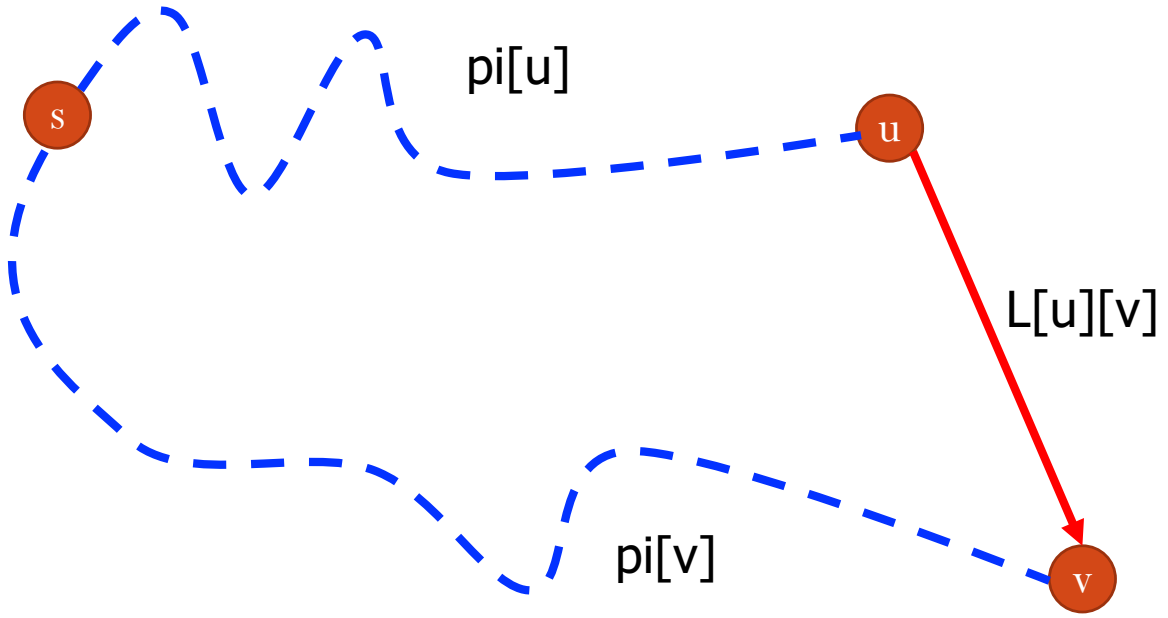
- Gọi đỉnh bắt đầu là s
- Các biến hỗ trợ
  - pi[u]: **chiều dài đường đi ngắn nhất** (tạm thời) từ s đến u.
  - p[u]: **đỉnh trước** đỉnh u trên đường đi ngắn nhất (tạm thời) từ s đến u.
  - mark[u]: đánh dấu đỉnh u (đã tìm được đường đi ngắn nhất đến u)
- Nếu mark[u] == 1 thì pi[u] chứa chiều dài đường đi ngắn nhất từ s đến u và p[u] là đỉnh trước của đỉnh.



Giải thuật Moore-Dijkstra

- Giải thuật
  - Khởi tạo:
    - pi[u] = oo với mọi u
    - p[u] = -1 với mọi u
    - mark[u] = 0 với mọi u
    - pi[s] = 0; đường đi ngắn nhất từ s đến s bằng 0.
  - Lặp n lần
    - Chọn đỉnh **chưa đánh dấu** và có **pi[u] nhỏ nhất**
    - Đánh dấu u
    - Xét các đỉnh kề v (chưa đánh dấu) của u để cập nhật đường đi nếu đường đi qua u rồi đến v tốt hơn đường đi cũ

Giải thuật Moore-Dijkstra



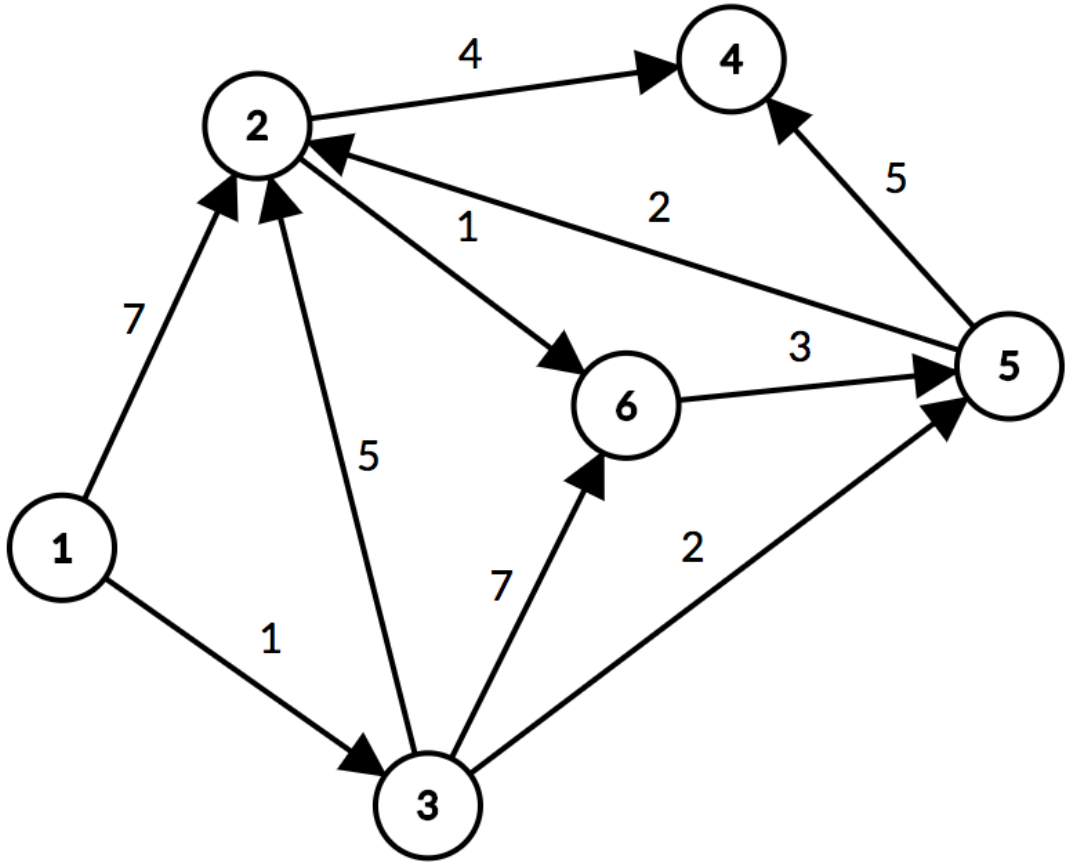
pi[u] + L[u][v] < pi[v]

Giải thuật Moore-Dijkstra

```
void Dijkstra(Graph* G, int s) {
   int u, v, it;
   for (u = 1; u <= G->n; u++) {
      pi[u] = INFINITY;
      mark[u] = 0;
   }
   pi[s] = 0;
   for (it = 1; it < G->n; it++) {
      //1. Tìm u có mark[u] == 0 và có pi[u]nhỏ nhất
      //2. Đánh dấu u đã xét mark[u] = 1;
      //3. Cập nhật pi và p của các đỉnh kề của u (nếu thoả)
      for (v = 1; v <= G->n; v++)
         if (G->L[u][v] != NO_EDGE && mark[v] == 0) {
            if (pi[u] + G->L[u][v] < pi[v]) {
               pi[v] = pi[u] + G->L[u][v];
               p[v] = u;
            }
          }
    }
}
```

Giải thuật Moore-Dijkstra

Tìm đường đi ngắn nhất từ 1 đến các đỉnh khác trên đồ thị có hướng



Giải thuật Moore-Dijkstra

Tìm đường đi ngắn nhất từ A đến các đỉnh khác trên đồ thị vô hướng có ma trận trọng số.

	A	B	C	D	E	F	G	H	K
A		1	1						4
B							9	2	3
C				7					2
D					1	5		4	3
E						2			
F							9	3	
G								5	
H									7
K									