BÀI TOÁN
ĐƯỜNG ĐI NGẮN NHẤT

Phát biểu bài toán

- ***Định nghĩa đường đi có trọng số:***

  Cho đồ thị G = (V, E) là đồ thị có trọng số và trọng số mỗi cạnh e là w(e). Với G' là một đồ thị con của G thì trọng số của G' được định nghĩa là:

$$w(G') = \sum_{e \in G'} w(e)$$

  - Nếu G' là đường đi hay chu trình thì w(G') gọi là độ dài của G'.
  - Nếu G' là một mạch (chu trình có các đỉnh không lặp lại) và w(G') < 0 thì ta gọi G' là *mạch âm*.

Phát biểu bài toán

- Cho đồ thị $G = (V, E)$ là một đồ thị có trọng số và $s, t \in V$. Gọi P là tập hợp tất cả các đường đi từ s tới t. Xét bài toán:

  Tìm $p_0 \in P$ sao cho $p_0 = \min\{w(p): p \in P\}$

- Bài toán này gọi là bài toán đường đi ngắn nhất (shortest path problem) và p_0 gọi là đường đi ngắn nhất (shortest path) từ s đến t.

Các nhận xét

1. Mặc dù bài toán được phát biểu cho đồ thị có hướng có trọng, nhưng các thuật toán sẽ trình bày đều có thể áp dụng cho các đồ thị vô hướng có trọng bằng cách xem mỗi cạnh của đồ thị vô hướng như hai cạnh có cùng trọng lượng nối cùng một cặp đỉnh nhưng có chiều ngược nhau.
2. Khi làm bài toán tìm đường đi ngắn nhất thì chúng ta có thể bỏ bớt đi các cạnh song song và chỉ chừa lại một cạnh có trọng lượng nhỏ nhất trong số các cạnh song song.

Các nhận xét

3. Đối với các khuyên có trọng lượng không âm thì cũng có thể bỏ đi mà không làm ảnh hưởng đến kết quả của bài toán. Đối với các khuyên có trọng lượng âm thì có thể đưa đến bài toán đường đi ngắn nhất không có lời giải.
4. Nếu G có mạch âm q trên đường đi từ u đến v thì đường đi ngắn nhất từ u đến v không tồn tại. Nhận xét số 4 này thực chất là mở rộng của nhận xét 3.

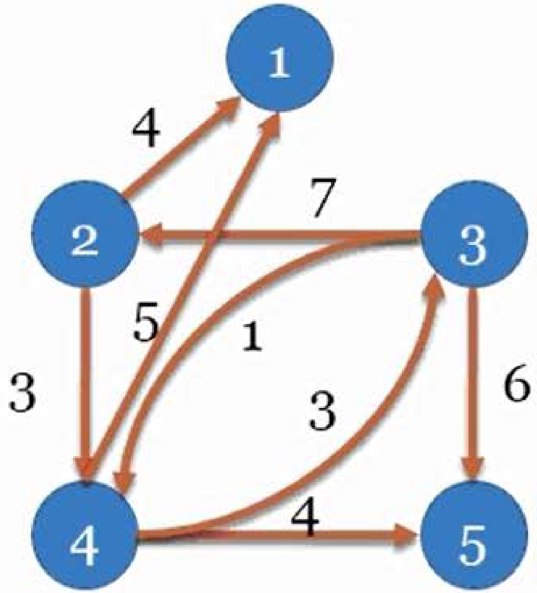
Ma trận khoảng cách

- Từ các nhận xét vừa nêu, có thể xem dữ liệu nhập của bài toán đường đi ngắn nhất là ma trận khoảng cách (distance matrix) D được định nghĩa như sau:
  - Ma trận khoảng cách của G là ma trận D = (D_{ij}) với

$$D_{ij} = \begin{cases} 0 & i = j \\ w(i,j) & (i,j) \in E \\ +\infty & (i,j) \notin E \end{cases}$$

  - Ma trận khoảng cách còn được gọi là ma trận trọng lượng.

Ví dụ về ma trận khoảng cách



	1	2	3	4	5
1	0	$+\infty$	$+\infty$	$+\infty$	$+\infty$
2	4	0	$+\infty$	3	$+\infty$
3	$+\infty$	7	0	1	6
4	5	$+\infty$	3	0	4
5	$+\infty$	$+\infty$	$+\infty$	$+\infty$	0

Nguyên lý Bellman

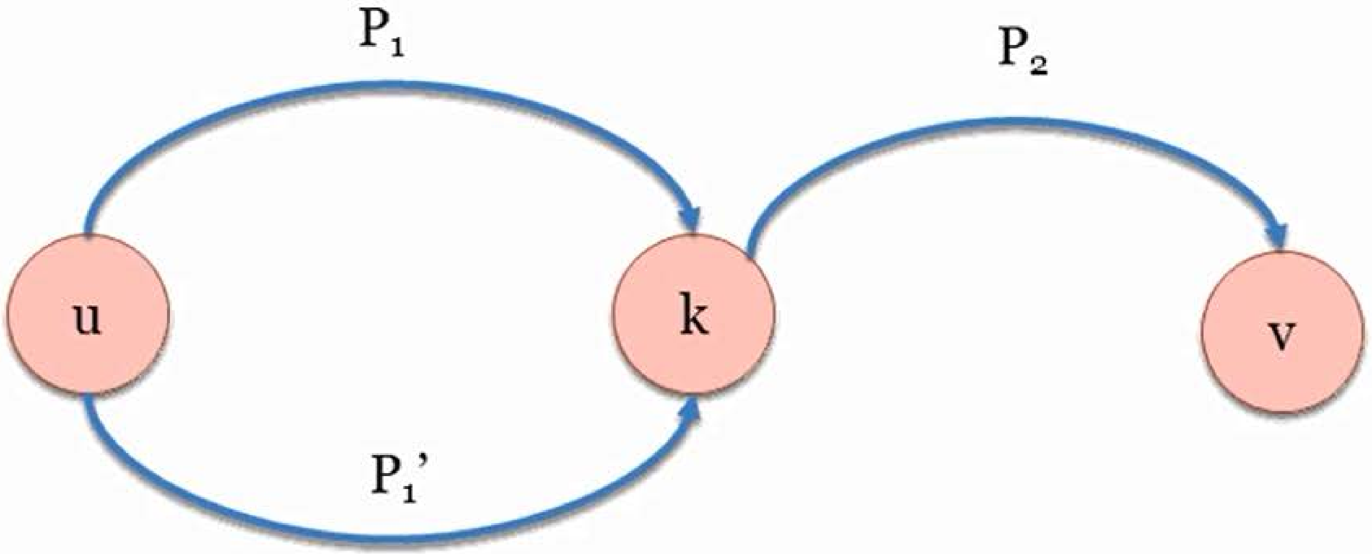
- Hầu hết các thuật toán tìm đường đi ngắn nhất đều đặt cơ sở trên nguyên lý Bellman. Đây là nguyên lý tổng quát cho các bài toán tối ưu hóa rời rạc, được nhà toán học người Mỹ **Richard Ernest Bellman** (1920 - 1984) đưa ra vào năm 1953. Nguyên lý này còn được gọi là nguyên lý quy hoạch động Bellman.

Richard Ernest Bellman
(1920 - 1984)

Nguyên lý Bellman

- Đối với trường hợp bài toán đường đi ngắn nhất thì có thể trình bày nguyên lý này như sau:
  - Giả sử P là đường đi ngắn nhất từ đỉnh u đến đỉnh v và k là một đỉnh nằm trên đường đi P.
  - Giả sử $P = P_1 \oplus P_2$ với P_1 là đường đi con của P từ u đến k và P_2 là đường đi con của P từ k đến v.
  - Nguyên lý Bellman nói rằng P_1 cũng là đường đi ngắn nhất từ u đến k, vì nếu có một đường đi khác là $P_{1'}$ từ u đến k có trọng lượng nhỏ hơn hơn P_1 thì $P_{1'} \oplus P_2$ là đường đi từ u đến v mà có trọng lượng nhỏ hơn P, điều này mâu thuẫn với tính ngắn nhất của P.

Nguyên lý Bellman



$$w(P1') < w(P1) \Rightarrow w(P_1' \oplus P_2) < w(P_1 \oplus P_2) = w(P)$$

Điều kiện tồn tại lời giải

- Gọi P là một đường đi từ u đến v, giả sử P có chứa một mạch μ. Có 2 trường hợp sau đây:
  - Nếu $L(\mu) \geq 0$ thì có thể cải tiến đường đi P bằng cách bỏ đi mạch μ.
  - Nếu $L(\mu) < 0$ thì không tồn tại đường đi ngắn nhất từ đỉnh u đến đỉnh v vì nếu quay vòng tại μ càng nhiều vòng thì trọng lượng đường đi P càng nhỏ đi, tức là $L(P) \to -\infty$.

BÀI TOÁN ĐƯỜNG ĐI NGẮN NHẤT

THUẬT TOÁN DIJKSTRA

Thuật toán Dijkstra

- **Edsger Wybe Dijkstra** /dɛɪkstra/ là một nhà toán học người Hà Lan. Ông đưa ra thuật toán tìm đường đi ngắn nhất – thuật toán mang tên ông – vào năm 1959.

Dijkstra (1930 – 2002)

Thuật toán Dijkstra – Dữ liệu vào/ra

- Xét đồ thị G = (V, E) có trọng giả sử không âm.
  - Dữ liệu nhập cho thuật toán là ma trận trọng lượng D và hai đỉnh u, v cho trước.
  - Dữ liệu xuất là đường đi ngắn nhất từ u đến v.

Thuật toán Dijkstra

- **Bước 1.** Gán T = V và gán các nhãn:

  $L[u] = 0; L[k] = +\infty, \forall k \in V \backslash \{u\};$

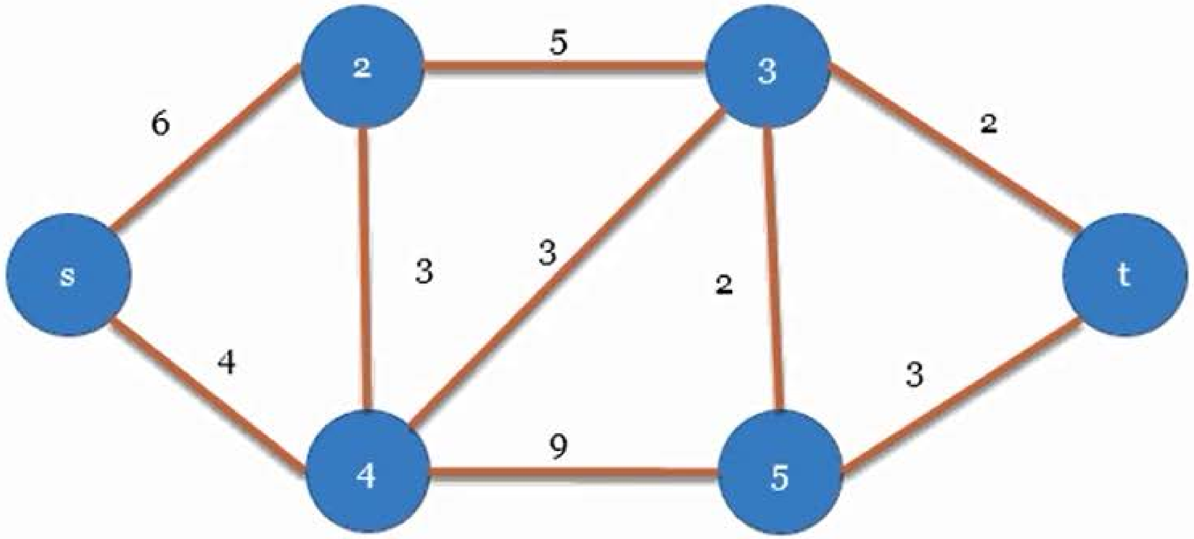
  $Prev[k] = -1, \forall k \in V.$
- **Bước 2.** Nếu $v \notin T$ thì dừng và giá trị L[v] chính là độ dài đường đi ngắn nhất từ u đến v và Prev[v] là đỉnh nằm ngay trước v trên đường đi đó.
- **Bước 3.** Chọn đỉnh $i \in T$ sao cho L[i] nhỏ nhất và gán $T = T \backslash \{i\}$.
- **Bước 4.**
  - Với $\forall k \in T$ và từ đỉnh i (ở bước 3) đến đỉnh k có cạnh nối:

    nếu $L[k] > L[i] + D_{ik}$ thì

    Gán $L[k] = L[i] + D_{ik}$ và $Prev[k] = i$
  - Trở về bước 2.

Ví dụ Dijkstra

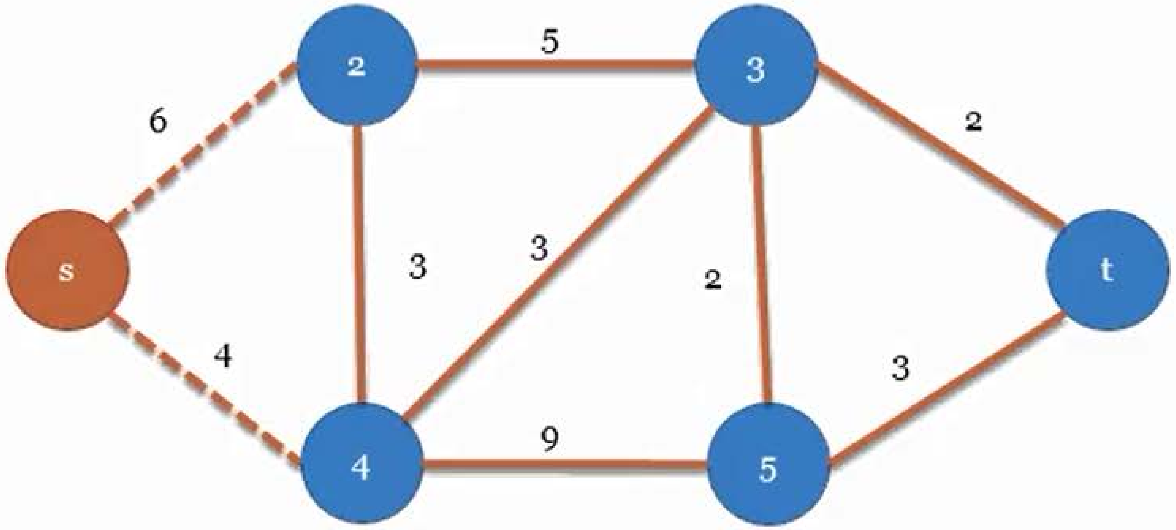
- Ví dụ 1



	s	2	3	4	5	t
T	s	2	3	4	5	t
L	0	+∞	+∞	+∞	+∞	+∞
Prev	-1	-1	-1	-1	-1	-1

Ví dụ Dijkstra

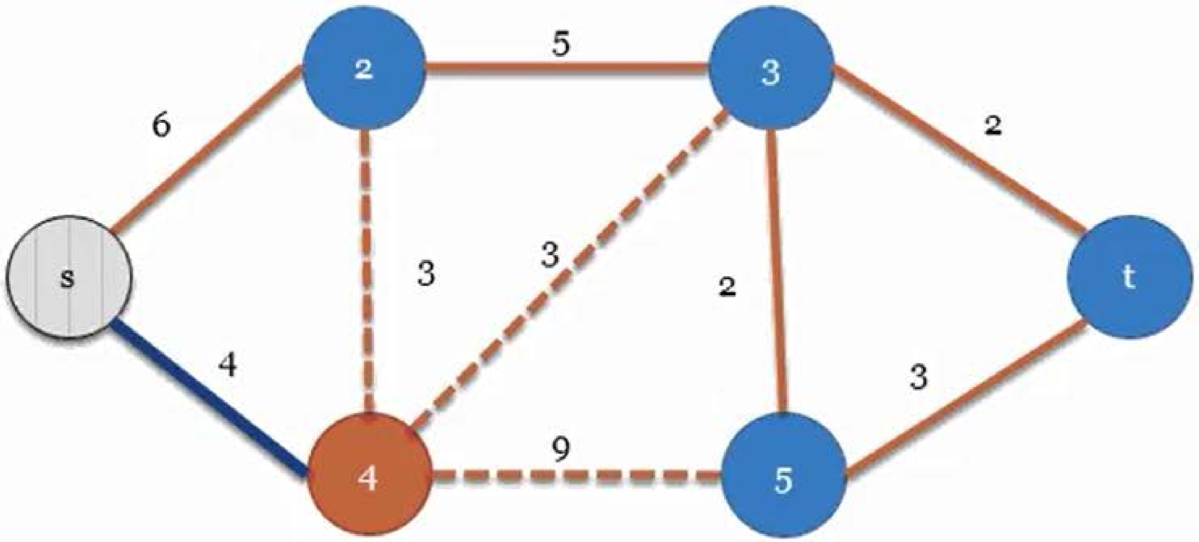
- Ví dụ 1



	s	2	3	4	5	t
T		2	3	4	5	t
L	0	6	$+\infty$	4	$+\infty$	$+\infty$
Prev	-1	s	-1	s	-1	-1

Ví dụ Dijkstra

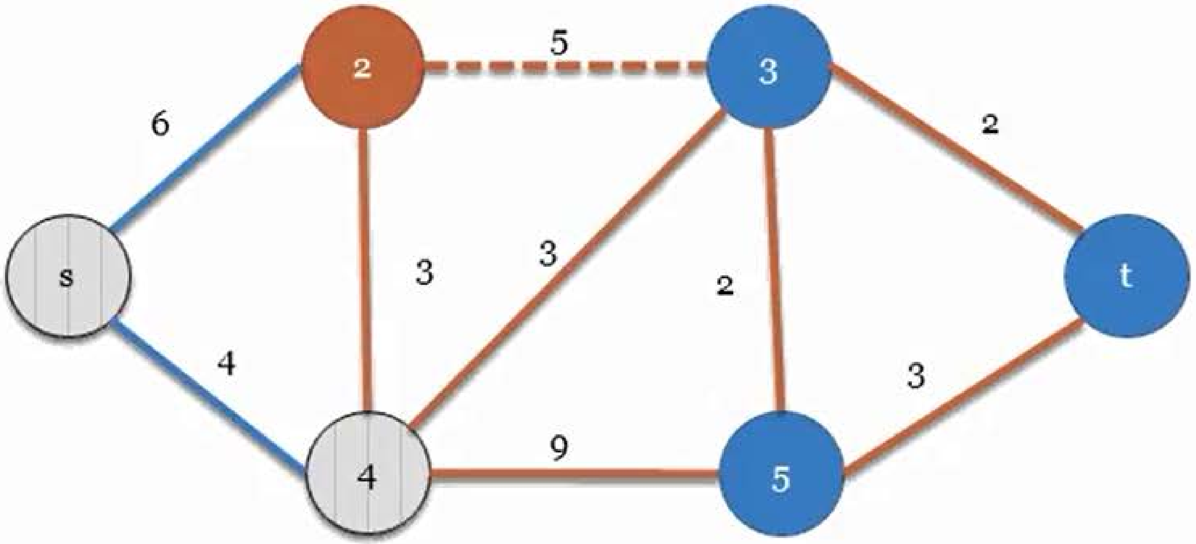
- Ví dụ 1



	s	2	3	4	5	t
T		2	3		5	t
L	0	6	7	4	13	$+\infty$
Prev	-1	s	4	s	4	-1

Ví dụ Dijkstra

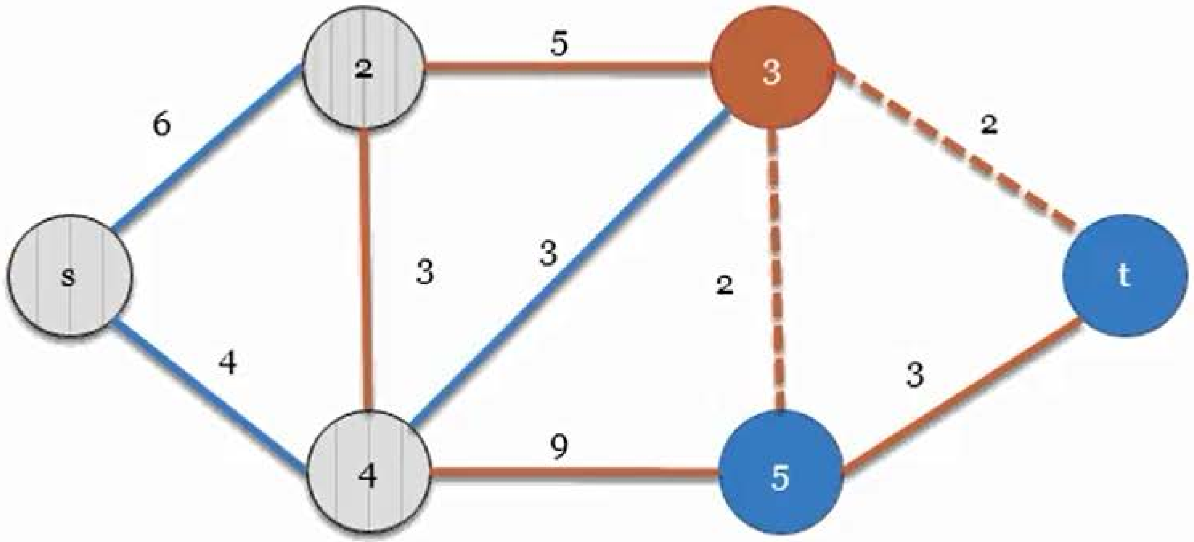
- Ví dụ 1



	s	2	3	4	5	t
T			3		5	t
L	0	6	7	4	13	+∞
Prev	-1	s	4	s	4	-1

Ví dụ Dijkstra

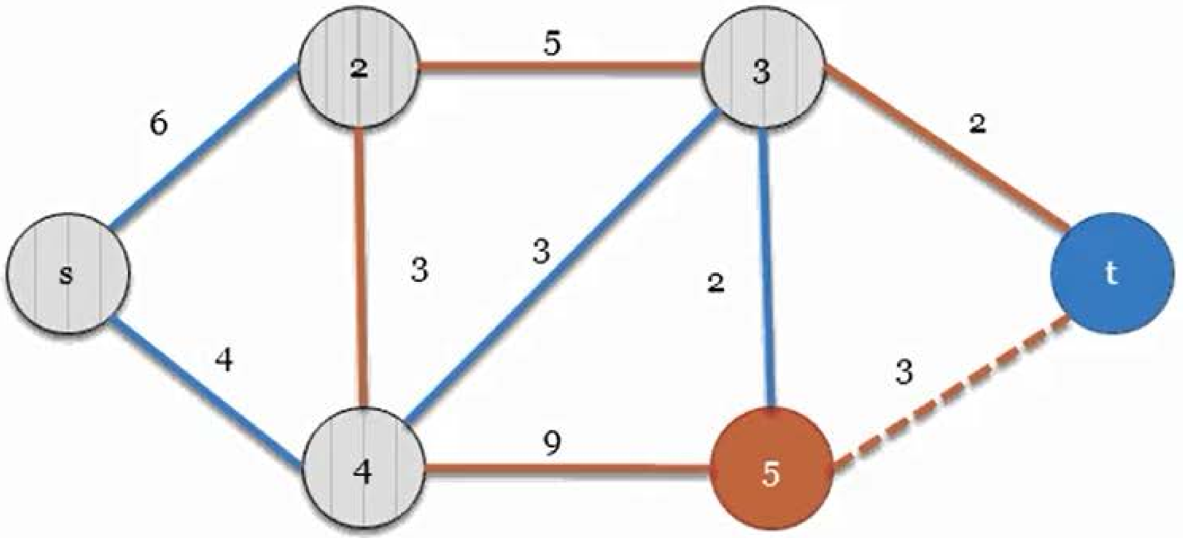
- Ví dụ 1



	s	2	3	4	5	t
T					5	t
L	0	6	7	4	9	9
Prev	-1	s	4	s	3	3.

Ví dụ Dijkstra

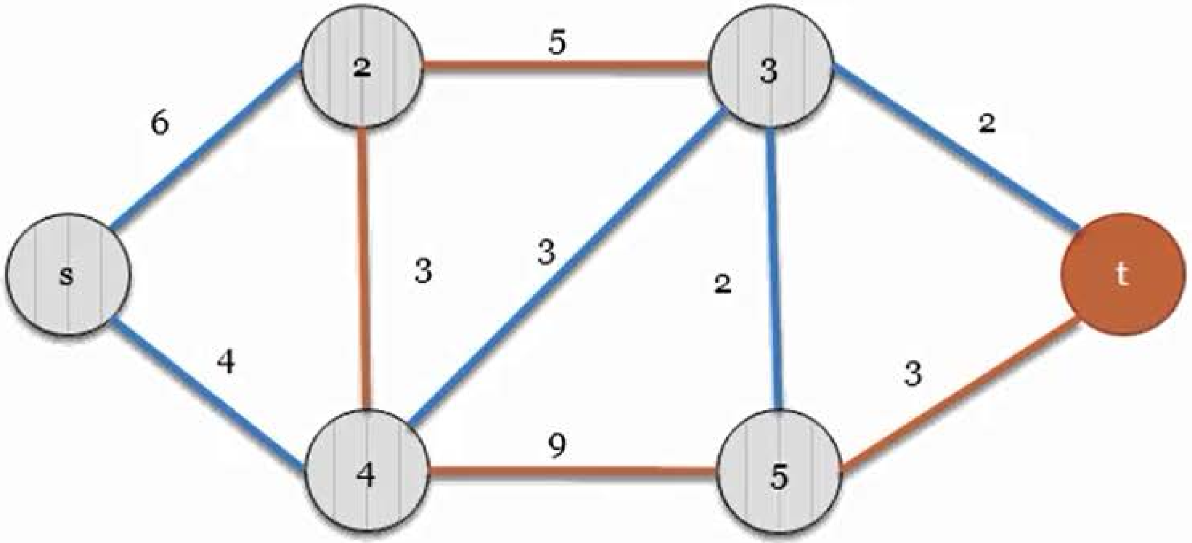
- Ví dụ 1



	s	2	3	4	5	t
T						t
L	0	6	7	4	9	9
Prev	-1	s	4	s	3	3

Ví dụ Dijkstra

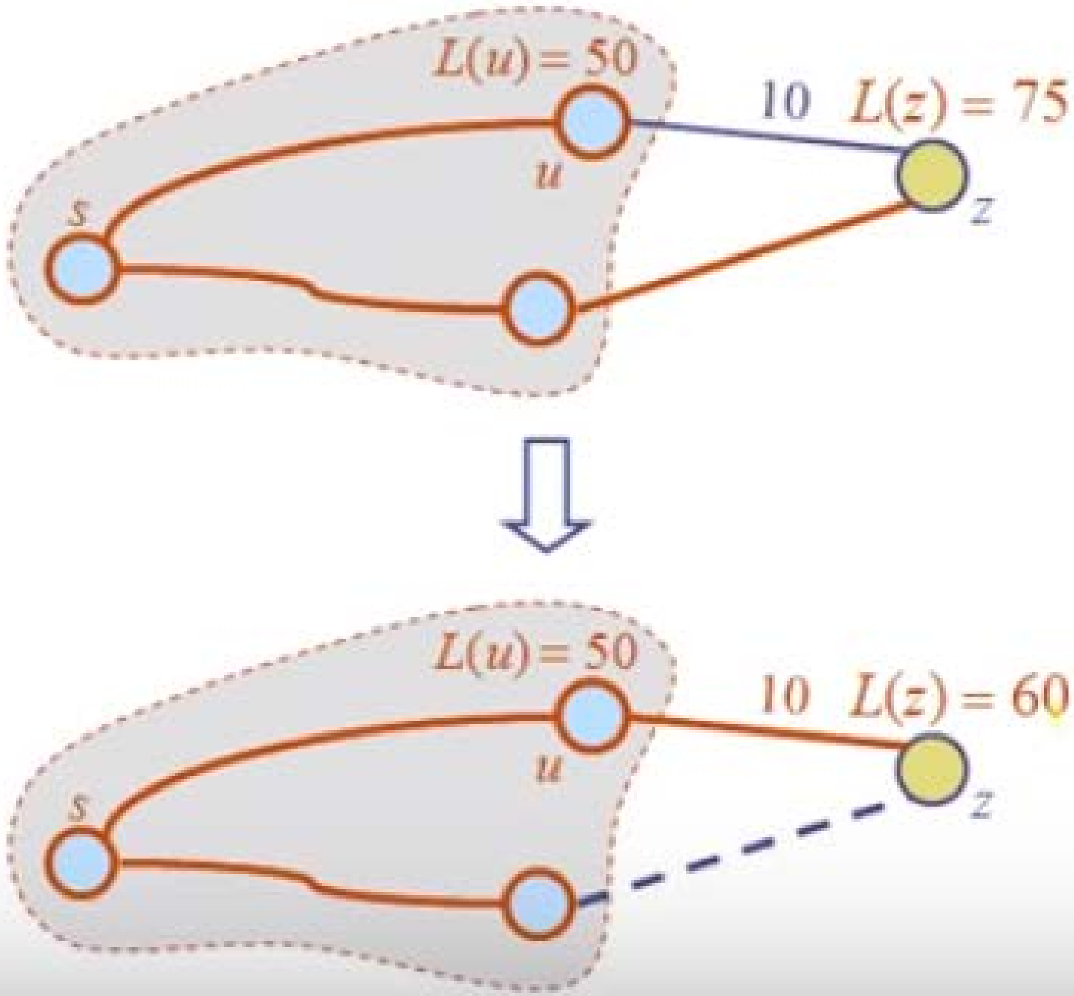
- Ví dụ 1



	s	2	3	4	5	t
T						
L	0	6	7	4	9	**9**
Prev	-1	s	4	s	3	3

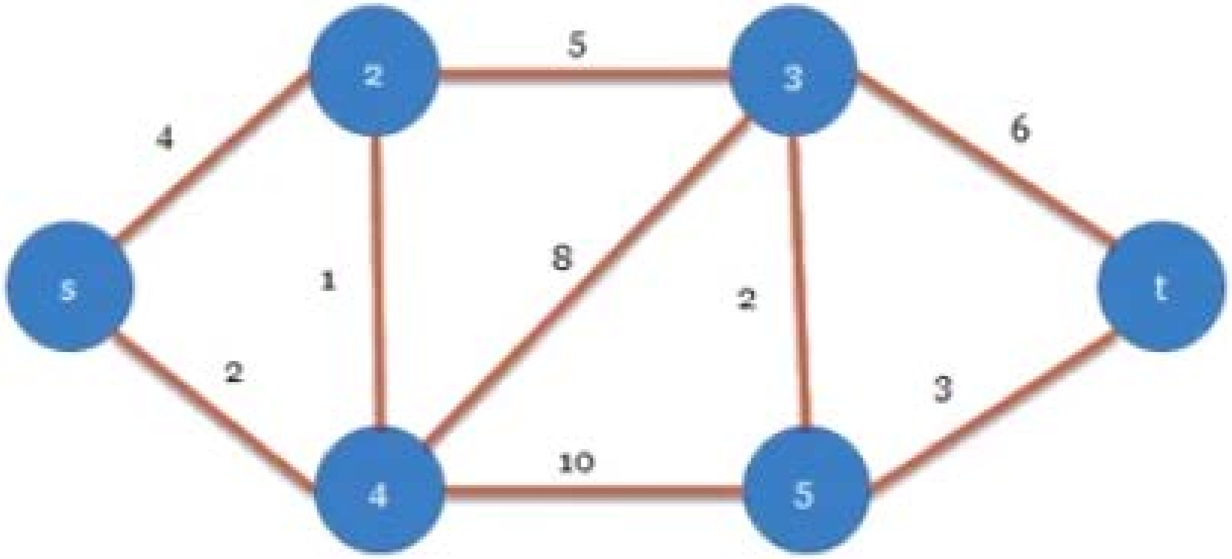
Ví dụ Dijkstra

- Ví dụ 1



Bài tập

- Tìm đường đi ngắn nhất bằng thuật toán Dijkstra từ đỉnh a đến đỉnh z cho đồ thị sau:



BÀI TOÁN ĐƯỜNG ĐI NGẮN NHẤT

THUẬT TOÁN BELLMAN

Lịch sử

- Thuật toán Bellman hay còn gọi là thuật toán Bellman – Ford do hai tác giả Bellman và Ford (Lester Randolph Ford, (1886 –1967)) đưa ra.
- Thuật toán Bellman – Ford có thể tìm được đường đi ngắn nhất trên đồ thị có trọng số âm.

Thuật toán Bellman

- Thuật toán này tìm đường đi ngắn nhất từ một đỉnh của đồ thị đến mỗi đỉnh khác *nếu đồ thị không có mạch âm*.
- Nếu phát hiện đồ thị có mạch âm thì thuật toán dừng. Dữ liệu nhập cho thuật toán là ma trận trọng lượng D.

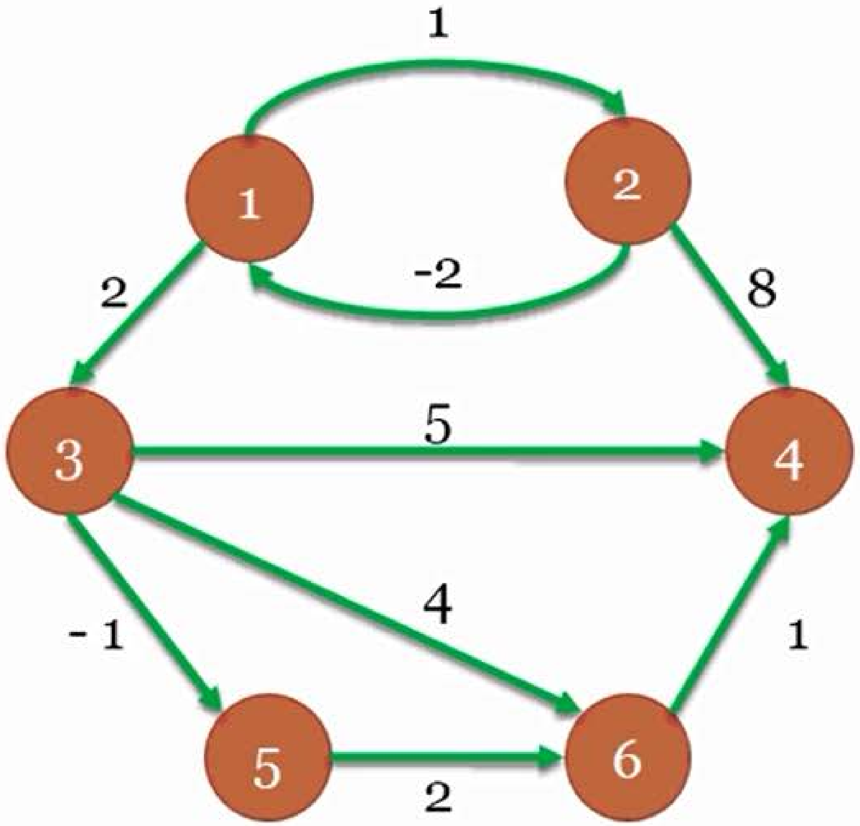
Thuật toán Bellman

- Cho trước đỉnh $x \in V$.
- Bước 1. Khởi tạo:
  - $\pi(0, x) = 0$; $\pi(0, i) = +\infty$, $\forall i \neq x$; $k = 1$.
  - $Prev(i) = i$, $\forall i \in V$
- Bước 2. Với mỗi $i \in V$ ta cập nhật:
  - $\pi(k, i) = min(\{\pi(k-1, i)\} \cup \{\pi(k-1, j) + D_{ji}\})$
  - *Nếu* $\pi(k, i) = \pi(k-1, j) + D_{ji}$: $Prev(i) = j$
- Bước 3.
  - Nếu $\pi(k, i) = \pi(k-1, i)$ với $\forall i \in V$ thì $\pi(k, i)$ chính là độ dài đường đi ngắn từ x đến i.
  - Ngược lại
    - Nếu $k < n$ thì tăng $k = k+1$ và trở lại bước 2;
    - Ngược lại thì dừng vì từ x đi tới được một mạch âm.

Ý nghĩa: Độ dài đường đi ngắn nhất đi qua tối đa k đỉnh

Ví dụ Bellman – Ford

- Xem đồ thị trong hình vẽ, chúng ta sẽ tính toán cho 2 trường hợp:
  - Các đường đi khởi đầu từ đỉnh 1, và
  - Các đường đi khởi đầu từ đỉnh 3.



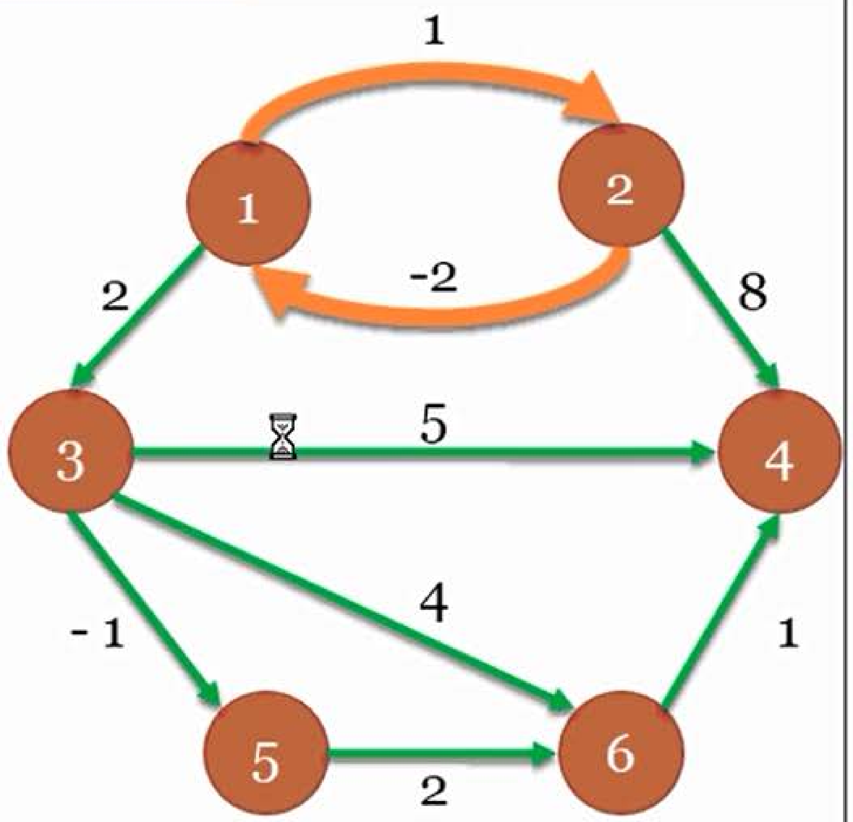
Ví dụ Bellman – Ford

- Khởi đầu từ 1
  - Khởi tạo: $\pi(0, 1) = 0$; $\pi(0, i) = +\infty$, $\forall i \neq x$; $k = 1$

π và k	1	2	3	4	5	6
k=0 và π =	0	+∞	+∞	+∞	+∞	+∞

- Với k = 1, cập nhật $\pi(1, i)$ $\forall i \in V$ theo công thức:
- $\pi(k, i) = min(\{\pi(k\text{-}1, i)\} \cup \{\pi(k\text{-}1, j) + D_{ji}\})$

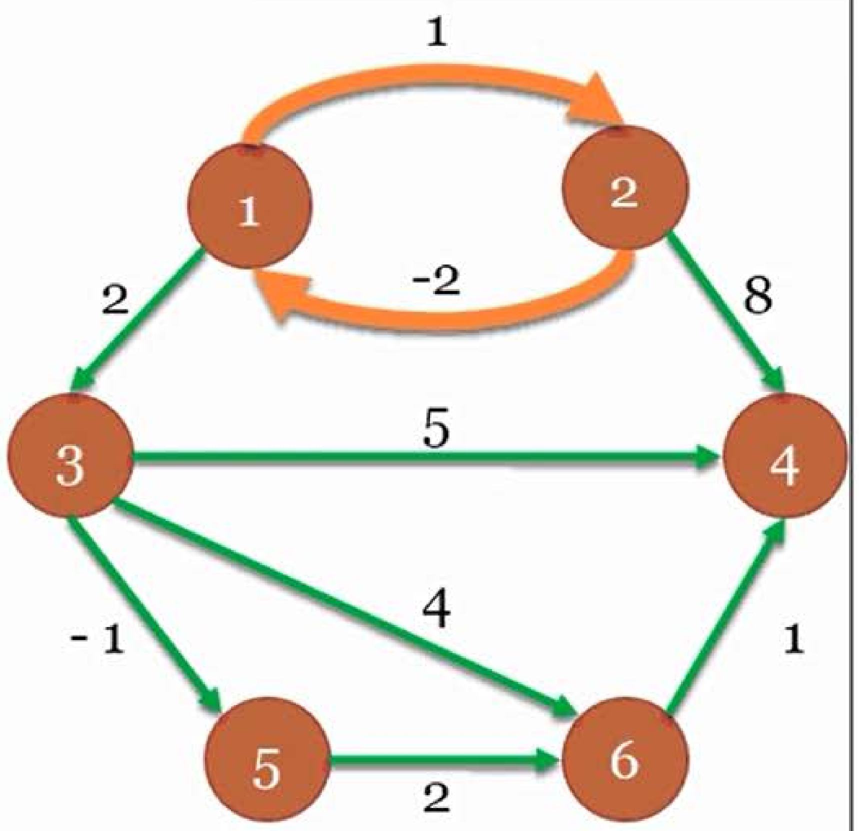
π và k	1	2	3	4	5	6
k=0 và π =	0	+∞	+∞	+∞	+∞	+∞
k=1 và π =	0	1	2	+∞	+∞	+∞



Ví dụ Bellman – Ford

- Với k = 2, cập nhật $\pi(2, i)\ \forall i \in V$ theo công thức:
- $\pi(k, i) = min(\{\pi(k-1, i)\} \cup \{\pi(k-1, j) + D_{ji}\})$

π và k	1	2	3	4	5	6
k=0 và π =	0	+∞	+∞	+∞	+∞	+∞
k=1 và π =	0	1	2	+∞	+∞	+∞
k=2 và π =	-1	1	2	7	1	6

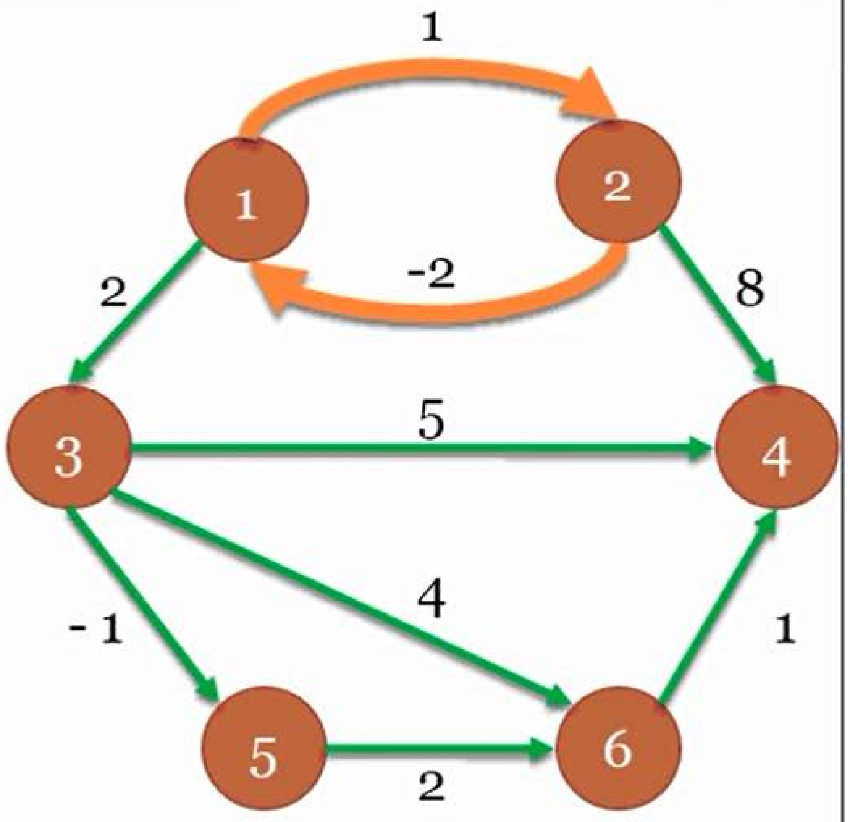


- Với k = 3

π và k	1	2	3	4	5	6
k=0 và π =	0	+∞	+∞	+∞	+∞	+∞
k=1 và π =	0	1	2	+∞	+∞	+∞
k=2 và π =	-1	1	2	7	1	6
k=3 và π =	-1	0	1	7	1	3

Ví dụ Bellman – Ford

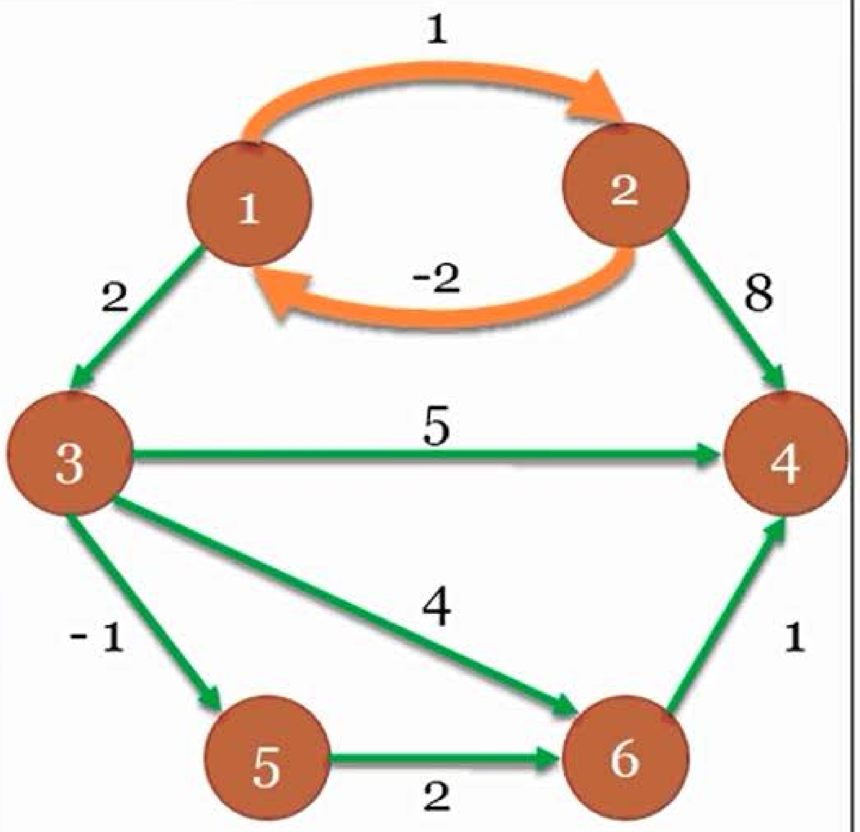
- Với k = 4
- Với k = 5
- Với k = 6



π và k	1	2	3	4	5	6
k=0 và π =	0	+∞	+∞	+∞	+∞	+∞
k=1 và π =	0	1	2	+∞	+∞	+∞
k=2 và π =	-1	1	2	7	1	6
k=3 và π =	-1	0	1	7	1	3
k=4 và π =	-2	0	1	4	0	3
k=5 và π =	-2	-1	0	4	0	2
k=6 và π =	-3	-1	0	3	-1	2

Ví dụ Bellman – Ford

- Trường hợp đường đi khởi đầu từ đỉnh 3, thuật toán dừng và cho biết có đường đi ngắn nhất từ đỉnh 3 đến mỗi đỉnh còn lại hay không. Các số trong ngoặc là các giá trị của đỉnh trước.



π và k	3	1	2	4	5	6
k=0 và π =	0	∞	∞	∞	∞	∞
k=1 và π =	0	∞	∞	5(3)	-1(3)	4(3)
k=2 và π =	0	∞	∞	5(3)	-1(3)	1(5)
k=3 và π =	0	∞	∞	2(6)	-1(3)	1(5)
k=4 và π =	0	∞	∞	2(6)	-1(3)	1(5)

Ví dụ Bellman – Ford

- Dựa vào bảng trên có thể suy ra:
  - Đường đi từ 3 đến 1 hay 2: không có;
  - Đường đi ngắn nhất từ 3 đến 4 (độ dài 2): $4 \leftarrow 6 \leftarrow 5 \leftarrow 3$;
  - Đường đi ngắn nhất từ 3 đến 5 (độ dài -1): $5 \leftarrow 3$;
  - Đường đi ngắn nhất từ 3 đến 6 (độ dài 1): $6 \leftarrow 5 \leftarrow 3$.

π và k	3	1	2	4	5	6
k=0 và π =	0	∞	∞	∞	∞	∞
k=1 và π =	0	∞	∞	5(3)	-1(3)	4(3)
k=2 và π =	0	∞	∞	5(3)	-1(3)	1(5)
k=3 và π =	0	∞	∞	2(6)	-1(3)	1(5)
k=4 và π =	0	∞	∞	2(6)	-1(3)	1(5)

BÀI TOÁN ĐƯỜNG ĐI NGẮN NHẤT
THUẬT TOÁN FLOYD – WARSHALL

Lịch sử

- Thuật toán được nhà toán học người Mỹ, Robert Floyd (1936 – 2001) đưa ra vào năm 1962.
- Bernard Roy (1934 -) cũng đưa ra thuật toán tương tự vào năm 1959. Do vậy, thuật toán còn có tên gọi là Roy – Floyd.
- Stephen Warshall (1935 – 2006) cũng công bố một thuật toán tương tự vào năm 1962.

Robert Floyd

Bernard Roy

Stephen Warshall

Thuật toán Floyd

- Thuật toán Floyd được dùng để tìm ra đường đi ngắn nhất giữa tất cả cặp đỉnh bất kỳ của một đồ thị G với các cạnh có trọng lượng dương.
- Dữ liệu nhập cho thuật toán là ma trận trọng lượng D.

Thuật toán Floyd

- Khởi đầu với ma trận trọng số D.
- Thực hiện n lần lặp trên D. Sau bước lặp thứ k, D[i,j] chứa độ dài đường đi ngắn nhất từ đỉnh i đến đỉnh j mà chỉ đi qua các đỉnh có chỉ số không vượt quá k.
- Vậy trong bước lặp thứ k ta thực hiện theo công thức sau đây:

$$D^{(k)}[i,j] = \min\left(D^{(k-1)}[i,j]\ ,\ D^{(k-1)}[i,k] + D^{(k-1)}[k,j]\right)$$

với k = 1, 2, ... , n.

Cài đặt thuật toán Floyd

```
void Floyd()
{
  for (i = 0; i < n; i++)
      for (j = 0; j < n; j++) {
          d[i][j] = a[i][j];
          p[i][j] = i;
      }
  for (k = 0; k < n; k++)
      for (i = 0; i < n; i++)
          for (j = 0; j<n; j++)
              if (d[i][j] > d[i][k] + d[k][j]) {
                  d[i][j] = d[i][k] + d[k][j];
                  p[i][j] = p[k][j];
              }
}
```

Ứng dụng thuật toán Floyd

- Giả sử G = (V, E) là một đồ thị có trọng số không âm.
- Ký hiệu: *d(x,y) là khoảng cách giữa đỉnh x và đỉnh y trong đồ thị G. Khoảng cách giữa x và y là độ dài đường đi ngắn nhất giữa x và y.*
- Đại lượng $d(a) = max \{d(x, a) \mid x \in V\}$ *được gọi là độ lệch của đỉnh a trong* đồ thị G.

Ứng dụng thuật toán Floyd

- Các khái niệm bán kính, tâm và đường kính của một đồ thị được định nghĩa như sau:

1. *Bán kính R của đồ thị G là độ lệch bé nhất trên các đỉnh:* $R = min\ \{d(a) \mid a \in V\}$.
2. *Tâm của đồ thị G là đỉnh a có độ lệch bé nhất:* $\forall x \in V,\ d(a) \leq d(x)$.
3. *Đường kính của đồ thị là khoảng cách dài nhất giữa các cặp đỉnh trong đồ thị:*

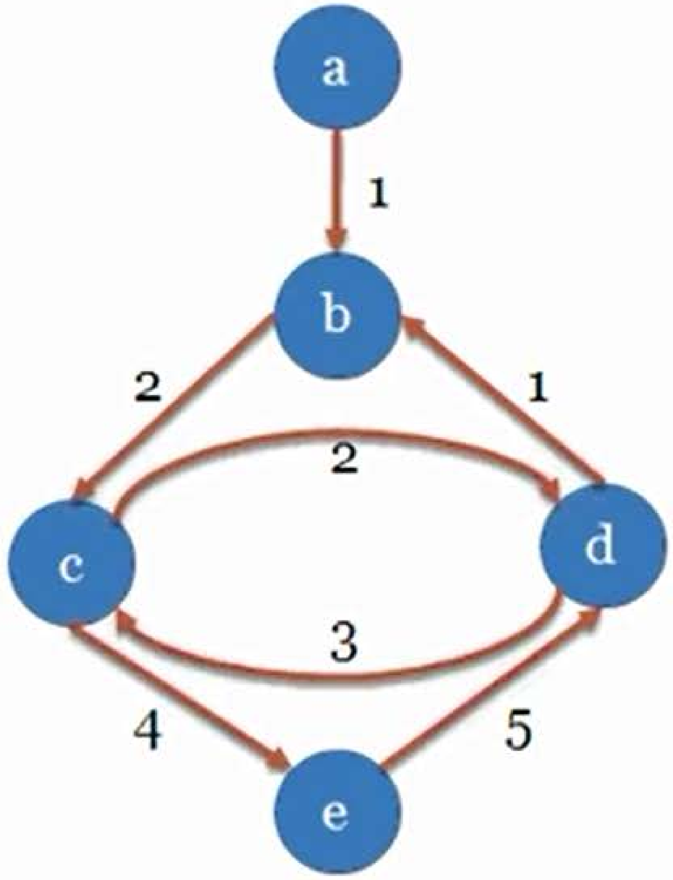
$$d = max\ \{d(x,y) \mid x, y \in V\}.$$

- Ý nghĩa của tâm đồ thị: Dùng để xác định thủ đô của một nước, nút giao thôngquan trọng trong một thành phố, vị trí đặt máy chủ trong một mạng máy tính ...

Ứng dụng thuật toán Floyd

- Dùng thuật toán Floyd để tính ma trận D các khoảng cách của các cặp đỉnh.
- Tìm giá trị lớn nhất trên mỗi cột, cho ta độ lệch của đỉnh tương ứng.
- Tìm đỉnh với độ lệch bé nhất, đó chính là tâm của đồ thị.

Ví dụ



- Độ lệch: a: ∞; b: 6; c: 8; d: 5; e: 7
- Tâm: d,
  bán kính: 5,
  đường kính: ∞.

a	*b*	*c*	*d*	*e*
0	1	3	5	7
∞	0	2	4	6
∞	3	0	2	4
∞	1	3	0	7
∞	6	8	5	0
∞	6	8	**5**	7