

Chương 6

Các bài toán tối ưu trên đồ thị

- ❖ Bài toán tìm đường đi ngắn nhất
- ❖ Bài toán tìm cây khung nhỏ nhất
- ❖ Bài toán tìm luồng cực đại

6.1. Tìm đường đi ngắn nhất

- ❖ Thuật toán Dijkstra
- ❖ Thuật toán Floyd

6.1.1. Thuật toán Dijkstra

Thuật toán Dijkstra để tìm đường đi ngắn nhất từ đỉnh **a** đến đỉnh **z** trên đồ thị có trọng số $G=(V,E,W)$ bao gồm việc đánh nhãn cho các đỉnh. Ở mỗi bước, một số đỉnh có nhãn cố định và một số đỉnh có nhãn tạm thời. Khi đỉnh **z** có nhãn cố định D_z thì D_z là độ dài đường đi ngắn nhất.

6.1.1. Thuật toán Dijkstra

Bước 1 $T=V$; $D_a = 0$; $D_i = \infty$, $v_i \neq a$.

Bước 2 Lặp cho đến khi $z \notin T$:

- Lấy ra khỏi T đỉnh v_i có D_i nhỏ nhất
- Đánh nhãn lại cho mọi v_j còn ở T và v_j kề v_i theo công thức:

$$D_j = \min\{D_j, D_i + W_{ij}\}$$

6.1.1. Thuật toán Dijkstra

Bảng trình bày có các cột như sau:

T: Tập đỉnh có nhãn tạm thời.

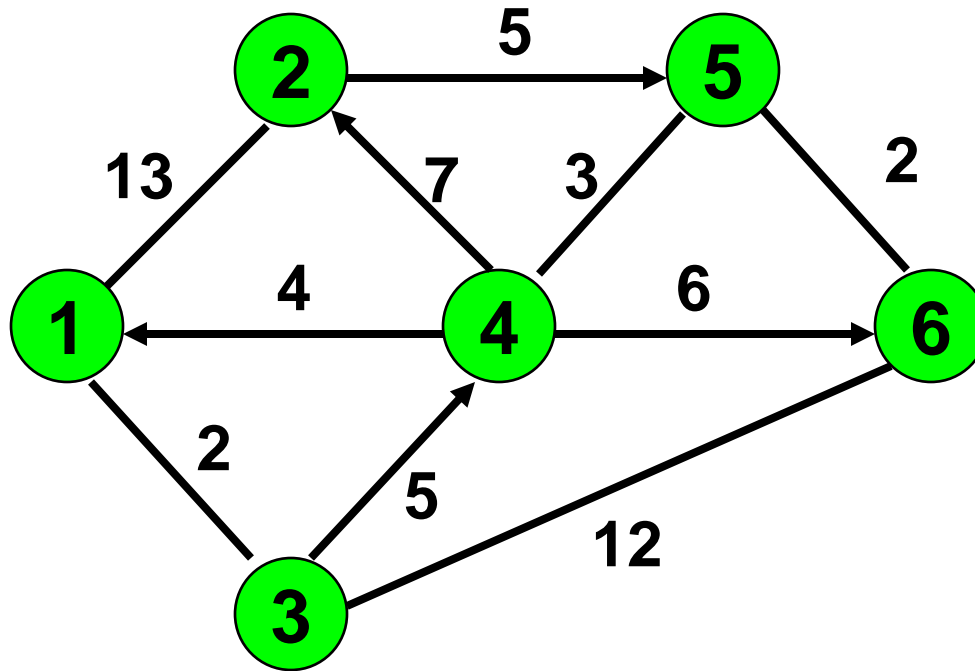
v_i : Đỉnh lấy ra khỏi T ở mỗi bước.

D_j : Độ dài đường đi ngắn nhất $a \rightarrow v_j$.

6.1.1. Thuật toán Dijkstra

Ví dụ. Cho đồ thị $G=(V,E,W)$

Tìm đường đi ngắn nhất từ v_1 đến v_6 .



6.1.1. Thuật toán Dijkstra

T	V_i	D_1	D_2	D_3	D_4	D_5	D_6
{1..6}	-	0	∞	∞	∞	∞	∞

6.1.1. Thuật toán Dijkstra

T	V_i	D_1	D_2	D_3	D_4	D_5	D_6
{1..6}	-	0	∞	∞	∞	∞	∞
{2..6}	v_1	0	13	2	∞	∞	∞

6.1.1. Thuật toán Dijkstra

T	V_i	D_1	D_2	D_3	D_4	D_5	D_6
{1..6}	-	0	∞	∞	∞	∞	∞
{2..6}	v_1	0	13	2	∞	∞	∞

6.1.1. Thuật toán Dijkstra

T	V_i	D_1	D_2	D_3	D_4	D_5	D_6
{1..6}	-	0	∞	∞	∞	∞	∞
{2..6}	v_1	0	13	2	∞	∞	∞
{2, 4..6}	v_3	-	13	2	7	∞	14

6.1.1. Thuật toán Dijkstra

T	V_i	D_1	D_2	D_3	D_4	D_5	D_6
{1..6}	-	0	∞	∞	∞	∞	∞
{2..6}	v_1	0	13	2	∞	∞	∞
{2, 4..6}	v_3	-	13	2	7	∞	14

6.1.1. Thuật toán Dijkstra

T	V_i	D_1	D_2	D_3	D_4	D_5	D_6
{1..6}	-	0	∞	∞	∞	∞	∞
{2..6}	v_1	0	13	2	∞	∞	∞
{2, 4..6}	v_3	-	13	2	7	∞	14
{2,5,6}	v_4	-	13	-	7	10	13

6.1.1. Thuật toán Dijkstra

T	V_i	D_1	D_2	D_3	D_4	D_5	D_6
{1..6}	-	0	∞	∞	∞	∞	∞
{2..6}	v_1	0	13	2	∞	∞	∞
{2, 4..6}	v_3	-	13	2	7	∞	14
{2,5,6}	v_4	-	13	-	7	10	13

6.1.1. Thuật toán Dijkstra

T	V_i	D_1	D_2	D_3	D_4	D_5	D_6
{1..6}	-	0	∞	∞	∞	∞	∞
{2..6}	v_1	0	13	2	∞	∞	∞
{2, 4..6}	v_3	-	13	2	7	∞	14
{2,5,6}	v_4	-	13	-	7	10	13
{2, 6}	v_5	-	13	-	-	10	12

6.1.1. Thuật toán Dijkstra

T	V_i	D_1	D_2	D_3	D_4	D_5	D_6
{1..6}	-	0	∞	∞	∞	∞	∞
{2..6}	v_1	0	13	2	∞	∞	∞
{2, 4..6}	v_3	-	13	2	7	∞	14
{2,5,6}	v_4	-	13	-	7	10	13
{2, 6}	v_5	-	13	-	-	10	12

6.1.1. Thuật toán Dijkstra

T	V_i	D_1	D_2	D_3	D_4	D_5	D_6
{1..6}	-	0	∞	∞	∞	∞	∞
{2..6}	v_1	0	13	2	∞	∞	∞
{2, 4..6}	v_3	-	13	2	7	∞	14
{2,5,6}	v_4	-	13	-	7	10	13
{2, 6}	v_5	-	13	-	-	10	12
{2}	v_6	-	13	-	-	-	12

6.1.1. Thuật toán Dijkstra

$v_6 \notin T$, thuật toán dừng. Để lấy đường đi ngắn nhất, dựa vào bảng trên, lần ngược từ v_6 đến v_1 có được đường đi ngắn nhất cần tìm:

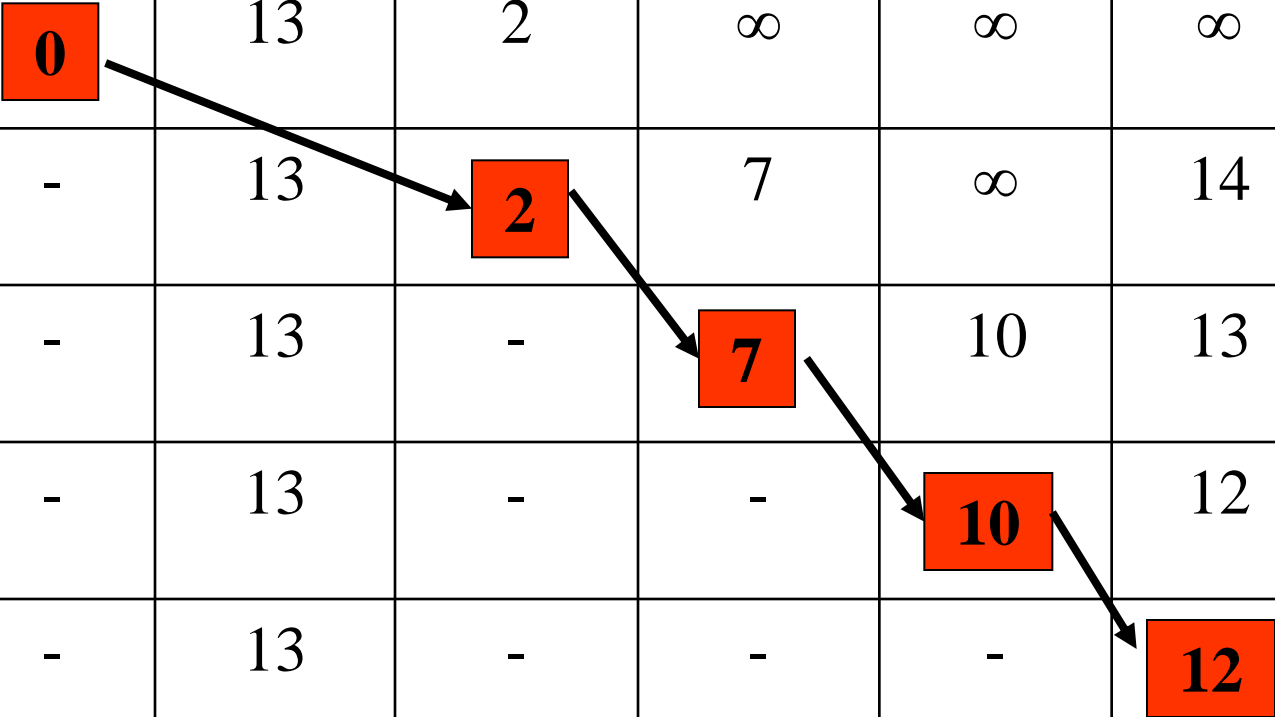
$$P = v_6 \leftarrow v_5 \leftarrow v_4 \leftarrow v_3 \leftarrow v_1.$$

Độ dài của đường đi ngắn nhất là

$$D_6 = 12$$

6.1.1. Thuật toán Dijkstra

T	V_i	D_1	D_2	D_3	D_4	D_5	D_6
{1..6}	-	0	∞	∞	∞	∞	∞
{2..6}	v_1	0	13	2	∞	∞	∞
{2, 4..6}	v_3	-	13	2	7	∞	14
{2,5,6}	v_4	-	13	-	7	10	13
{2, 6}	v_5	-	13	-	-	10	12
{2}	v_6	-	13	-	-	-	12



6.1.1. Thuật toán Dijkstra

Nhận xét

Để lấy đường đi ngắn nhất từ **a** đến mọi đỉnh, thay vòng lặp “Lặp cho đến khi $z \notin T$ ” bởi “Lặp cho đến khi $T = \emptyset$ ”.

Với bảng trên, thêm một bước nữa có được đường đi ngắn nhất từ v_1 đến mọi đỉnh.

6.1.1. Thuật toán Dijkstra

Nhận xét

Cũng có thể đánh cho mỗi đỉnh v_j một cặp nhãn $[D_j, v_i]$ với:

D_j là độ dài đường đi ngắn nhất $a \rightarrow v_j$.

v_i là đỉnh trước v_j trên đường đi ngắn nhất.

Nhãn thứ hai để lấy đường đi ngắn nhất.

Cách này gần với cài đặt chương trình.

Với ví dụ trên. Có bảng sau:

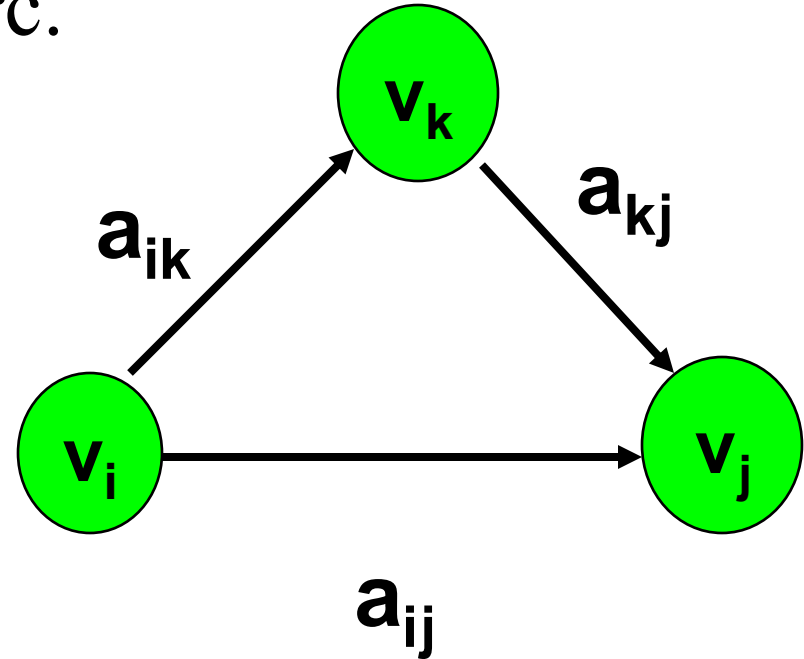
6.1.1. Thuật toán Dijkstra

T	v_1	v_2	v_3	v_4	v_5	v_6
{1..6}	$[0, v_1]$	$[\infty, v_1]$	$[\infty, v_1]$	$[\infty, v_1]$	$[\infty, v_1]$	$[\infty, v_1]$
{2..6}	*	$[13, v_1]$	$[2, v_1]$	$[\infty, v_1]$	$[\infty, v_1]$	$[\infty, v_1]$
{2, 4..6}	-	$[13, v_1]$	*	$[7, v_3]$	$[\infty, v_1]$	$[14, v_3]$
{2, 5, 6}	-	$[13, v_1]$	-	*	$[10, v_4]$	$[13, v_4]$
{2, 6}	-	$[13, v_1]$	-	-	*	$[12, v_5]$
{2}	-	$[13, v_1]$	-	-	-	*

6.1.2. Thuật toán Floyd

Để tìm đường đi ngắn nhất giữa mọi cặp đỉnh và lưu độ dài trong chính ma trận trọng số $A=(a_{ij})_{n \times n}$.

Thuật toán thực hiện n bước.



Bước k đặt $a_{ij} = \min\{ a_{ij}, a_{ik} + a_{kj} \}$

6.1.2. Thuật toán Floyd

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


6.2. Tìm cây khung nhỏ nhất

- ❖ Các khái niệm
- ❖ Các định lý
- ❖ Thuật toán Prim
- ❖ Thuật toán Kruskal

6.2.1. Các khái niệm

- *Cây* là đồ thị vô hướng liên thông không chu trình.
- Cho đồ thị vô hướng $G = (V, E)$, *cây khung* T của đồ thị G là đồ thị con chứa tất cả các đỉnh của G và T là một cây.
- Cho đồ thị vô hướng $G = (V, E, W)$, *cây khung nhỏ nhất* của đồ thị G là cây khung có trọng số nhỏ nhất trong tất cả các cây khung của G .

6.2.2. Các định lý

Định lý 1. Giả sử $T=(V,E)$ là đồ thị vô hướng n đỉnh. Khi đó các mệnh đề sau tương đương:

- (1) T là cây;
- (2) T không chứa chu trình và có $n-1$ cạnh;
- (3) T liên thông và có $n-1$ cạnh;

Định lý 2.

G có cây khung khi và chỉ khi G liên thông.

6.2.3. Thuật toán Prim

Bước 1 $T = \{v\}$; v bất kỳ

Bước 2 Lặp $n-1$ lần:

- Tìm đỉnh v có cạnh e nối T với $w(e)$ nhỏ nhất.
- Đưa v và e vào T .

6.2.3. Thuật toán Prim

Ví dụ. Cho đồ thị có ma trận trọng số sau

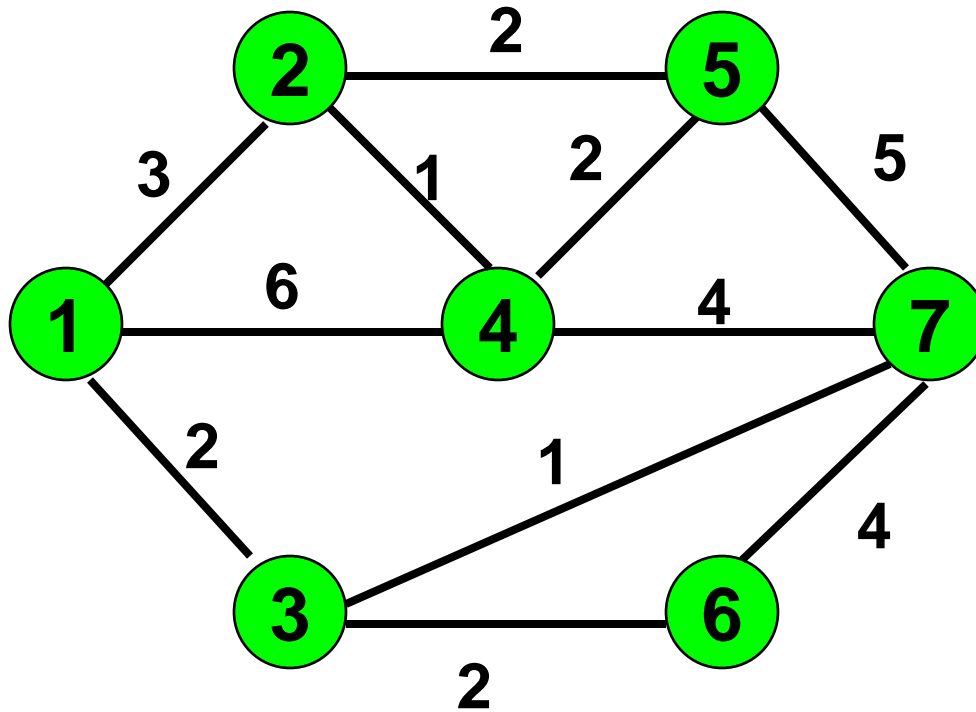
	v1	v2	v3	v4	v5	v6
v1	0	33	17	∞	∞	∞
v2	33	0	18	20	∞	∞
v3	17	18	0	16	4	∞
v4	∞	20	16	0	9	8
v5	∞	∞	4	9	0	14
v6	∞	∞	∞	8	14	0

6.2.3. Thuật toán Prim

E_T	v1	v2	v3	v4	v5	v6
-	*	[33,v1]	[17,v1]	$[\infty, v1]$	$[\infty, v1]$	$[\infty, v1]$
(1,3)	-	[18,v3]	*	[16,v3]	[4,v3]	$[\infty, v1]$
(3,5)	-	[18,v3]	-	[9,v5]	*	[14,v5]
(5,4)	-	[18,v3]	-	*	-	[8,v4]
(4,6)	-	[18,v3]	-	-	-	*
(3,2)	-	*	-	-	-	-

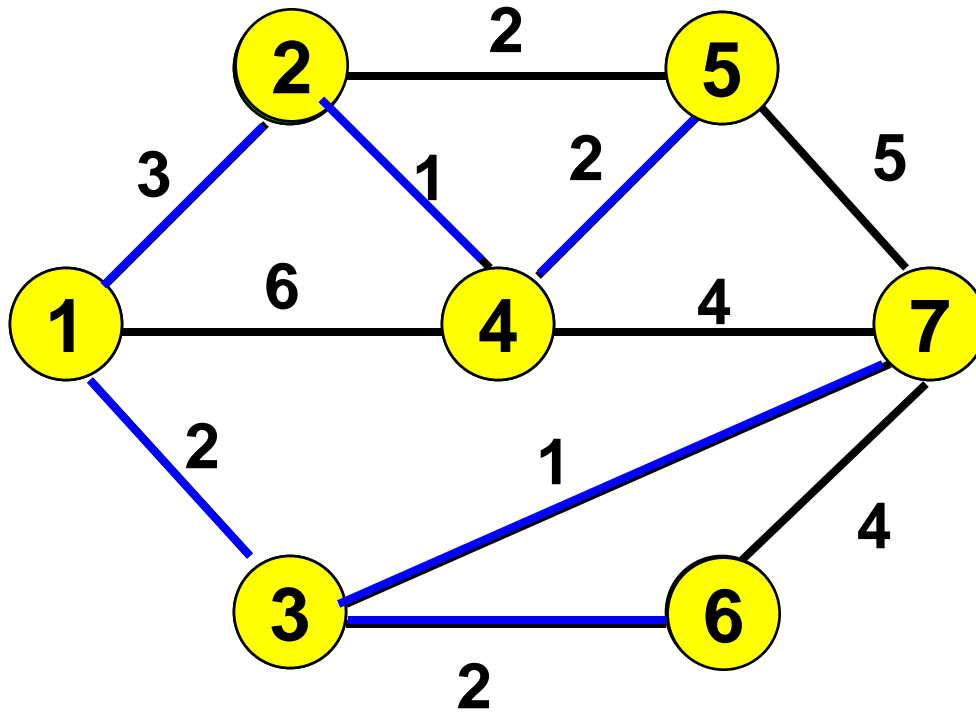
6.2.3. Thuật toán Prim

Cũng có thể trình bày trực quan. *Ví dụ.*



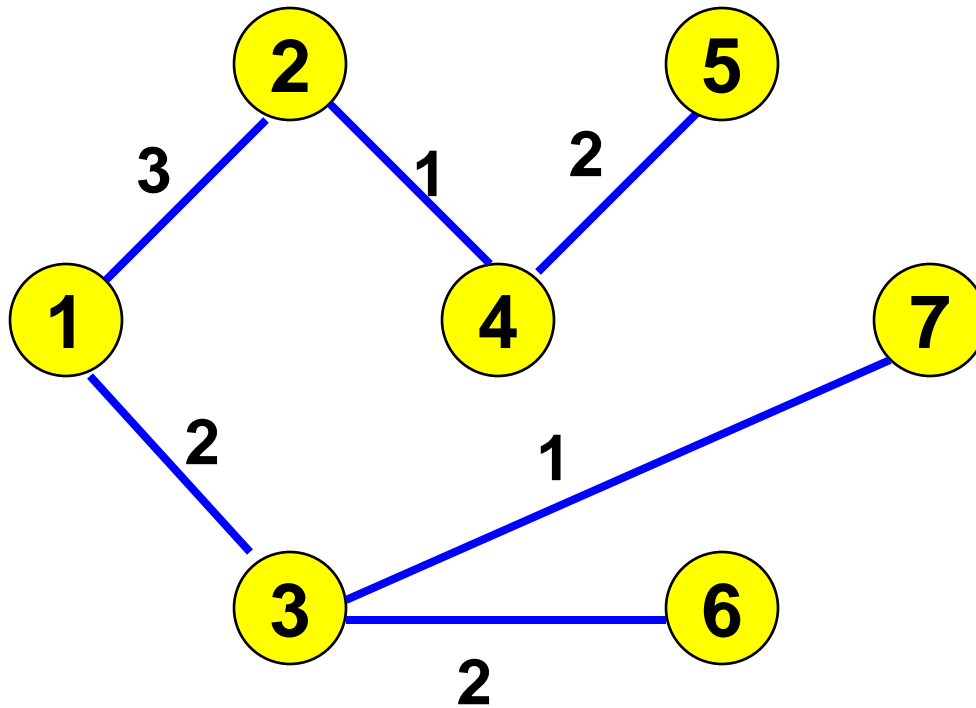
6.2.3. Thuật toán Prim

Cũng có thể trình bày trực quan. *Ví dụ.*



6.2.3. Thuật toán Prim

Cây khung nhỏ nhất T với $W(T)=11$



6.2.4. Thuật toán Kruskal

Bước 1 $T=V$; T không có cạnh

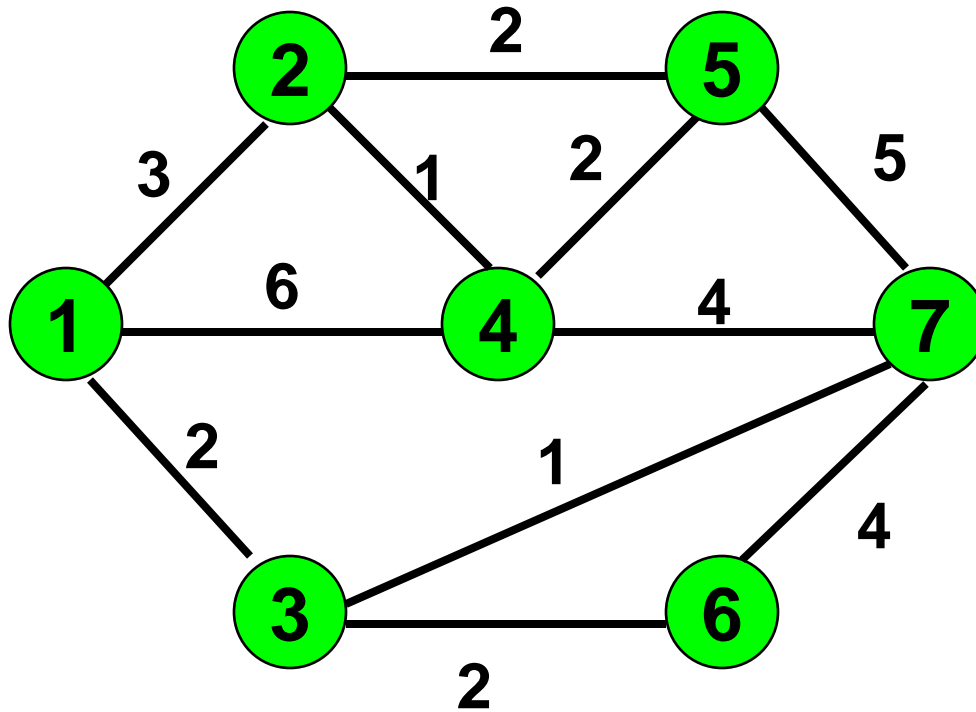
Bước 2 Lặp $n-1$ lần:

- Tìm cạnh e có trọng số nhỏ nhất và đưa vào T không tạo chu trình.
- Đưa e vào T .

Ban đầu sắp xếp cạnh có trọng số tăng dần

6.2.4. Thuật toán Kruskal

Ví dụ.



6.2.4. Thuật toán Kruskal

Sắp xếp các cạnh tăng dần trọng số

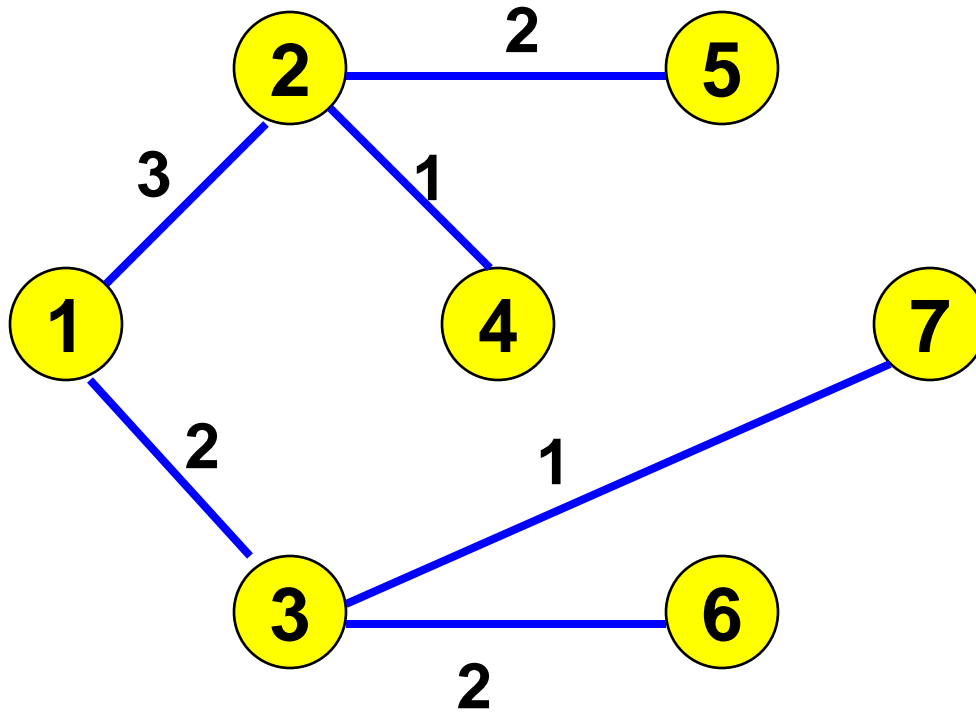
e	(2,4)	(3,7)	(1,3)	(2,5)	(3,6)	(4,5)	(1,2)	(4,7)	(6,7)	(5,7)	(1,4)
w_e	1	1	2	2	2	2	3	4	4	5	6
e_T	1	2	3	4	5	-	6	-	-	-	-

Cây T tìm gồm 6 cạnh sau:

(2,4), (3,7), (1,3), (2,5), (3,6), (1,2).

6.2.4. Thuật toán Kruskal

Cây khung nhỏ nhất T với $W(T)=11$



6.1+2.

Tìm đường đi ngắn nhất

Tìm cây khung nhỏ nhất

DOTHI.CPP

6.3. Tìm luồng cực đại

- ❖ Các khái niệm
- ❖ Thuật toán Ford-Fulkerson

6.3.1. Các khái niệm

- *Mạng* là đồ thị có hướng, có trọng số $G=(V,E,C)$
- G liên thông yếu (bỏ hướng thì liên thông).
- Có duy nhất một đỉnh s không có cung vào gọi là *đỉnh phát* và duy nhất một đỉnh t không có cung ra gọi là *đỉnh thu*.
- Mỗi cung (i,j) được gán một số $c_{ij} \geq 0$ gọi là *khả năng thông qua* của cung (i,j) .

6.3.1. Các khái niệm

- *Luồng* $F=(f_{ij})$ trên mạng $G=(V,E,C)$ là việc gán cho mỗi cung (i,j) một số f_{ij} thoả mãn:
 - Mọi cung (i,j) có: $0 \leq f_{ij} \leq c_{ij}$
 - Mọi đỉnh v_i khác *s* và *t* có tổng luồng vào và ra bằng nhau.

Từ đó có: Tổng luồng ra khỏi *s* bằng tổng luồng vào *t* gọi là *giá trị của luồng*, ký hiệu v_F .

Luồng cực đại trên mạng G là luồng có giá trị lớn nhất trong tất cả các luồng trên G .

6.3.2. Thuật toán Ford-Fulkerson

Bước 1 $F=0$; //khởi đầu luồng 0, mọi (i,j) có: $f_{ij}=0$

Bước 2 Lặp cho đến khi hết đường tăng luồng:

- Tìm đường tăng luồng P từ s đến t , với lượng tăng luồng ∂ .
- Tăng luồng dọc theo P một lượng ∂ .

6.3.2. Thuật toán Ford-Fulkerson

Đường tăng luồng P tìm được có dạng

$P: s \rightarrow \dots \rightarrow \mathbf{i} \rightarrow \mathbf{j} \rightarrow \dots \rightarrow t$ thì (i,j) là cung thuận.

$P: s \rightarrow \dots \rightarrow \mathbf{i} \leftarrow \mathbf{j} \rightarrow \dots \rightarrow t$ thì (j,i) là cung nghịch.

6.3.2. Thuật toán Ford-Fulkerson

Bước tìm đường tăng luồng P có thể dùng cách đánh nhãn như sau:

- Đặt nhãn s là ∞ .
- Lặp cho đến khi t có nhãn ∞ : khi đỉnh v_i vừa có nhãn thì đánh nhãn cho mọi v_j kề v_i nếu thỏa mãn một trong hai trường hợp sau:

6.3.2. Thuật toán Ford-Fulkerson

- a) Nếu có cung (i,j) và $c_{ij} - f_{ij} > 0$ thì đặt $\partial_j = \min\{\partial_i, c_{ij} - f_{ij}\}$, nạp cung thuận (i,j) vào P.
- b) Nếu có cung (j,i) và $f_{ji} > 0$ thì đặt $\partial_j = \min\{\partial_i, f_{ji}\}$, nạp cung nghịch (j,i) vào P.

Khi t có nhãn ∂t thì lượng tăng luồng $\partial = \partial t$. Tăng luồng xong xóa nhãn.

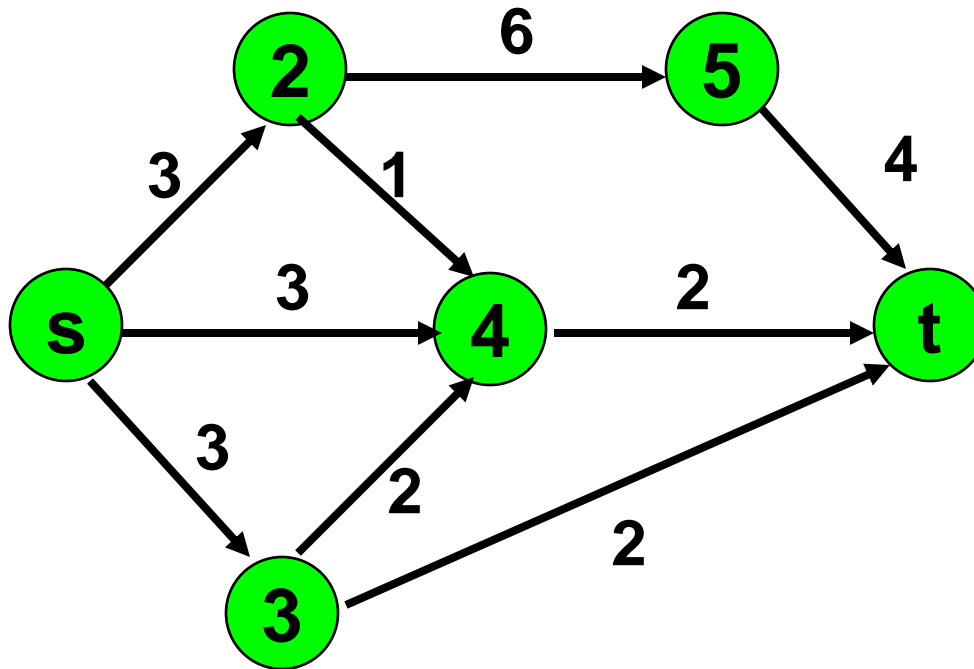
6.3.2. Thuật toán Ford-Fulkerson

Tăng luồng dọc theo P một lượng ∂ theo công thức:

$F_{ij}' = F_{ij} + \partial$	nếu (i,j) là cung thuận
$F_{ji}' = F_{ji} - \partial$	nếu (i,j) là cung nghịch
F_{ij}	nếu (i,j) ngoài P

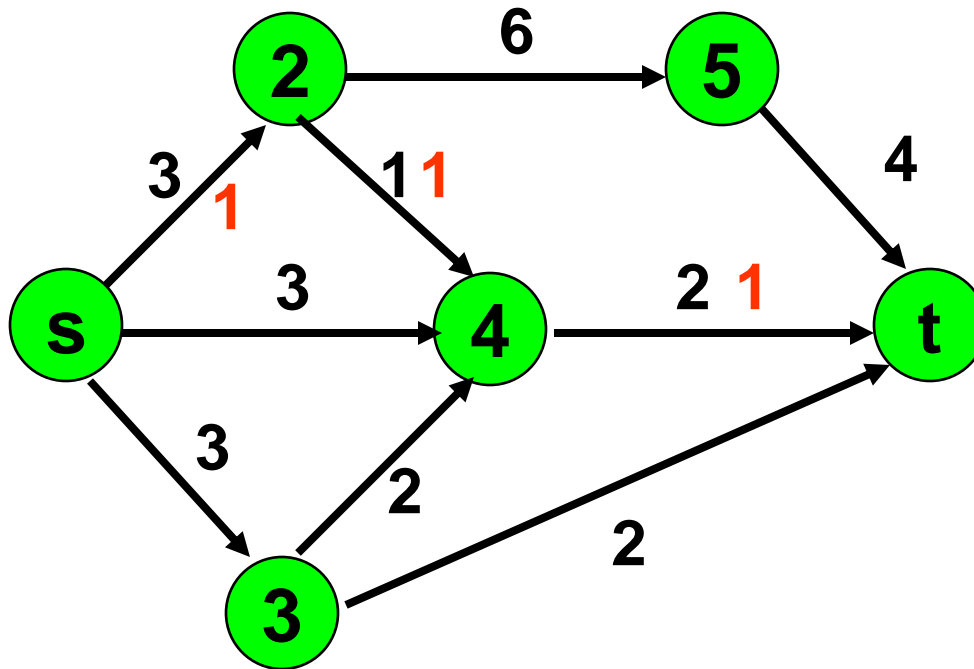
6.3.2. Thuật toán Ford-Fulkerson

Ví dụ. Cho mạng $G=(V,E,C)$



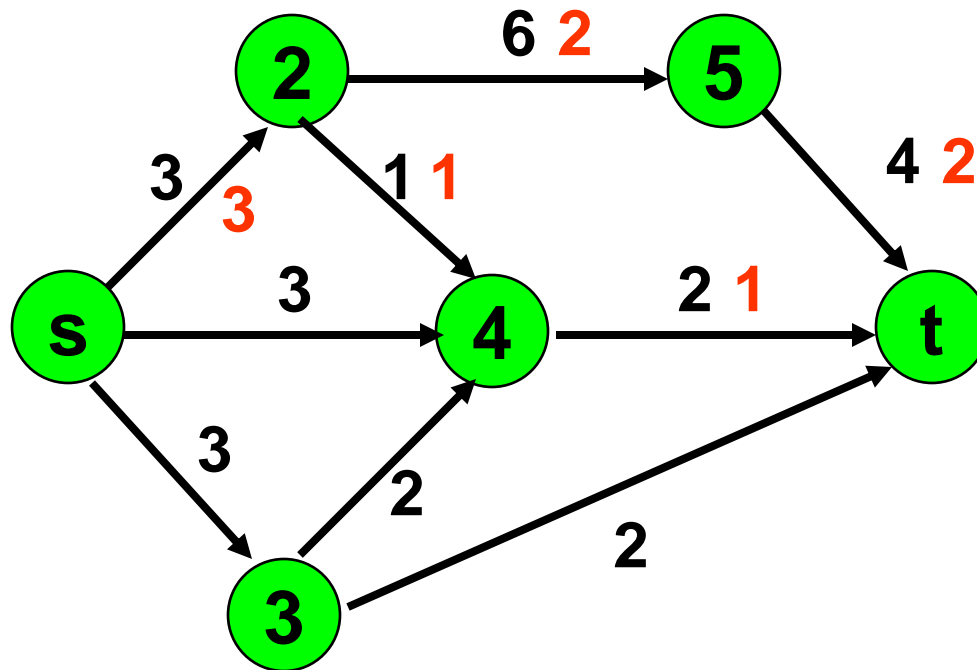
6.3.2. Thuật toán Ford-Fulkerson

$P_1: s \rightarrow 2 \rightarrow 4 \rightarrow t, \partial_1 = 1.$



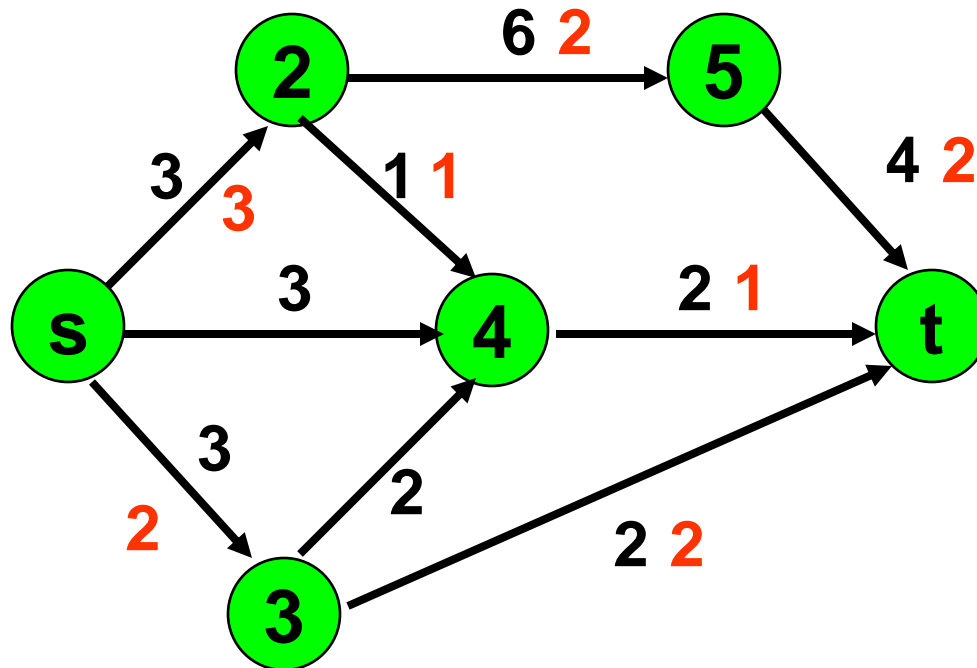
6.3.2. Thuật toán Ford-Fulkerson

$P_2: s \rightarrow 2 \rightarrow 5 \rightarrow t, \partial_2 = 2.$



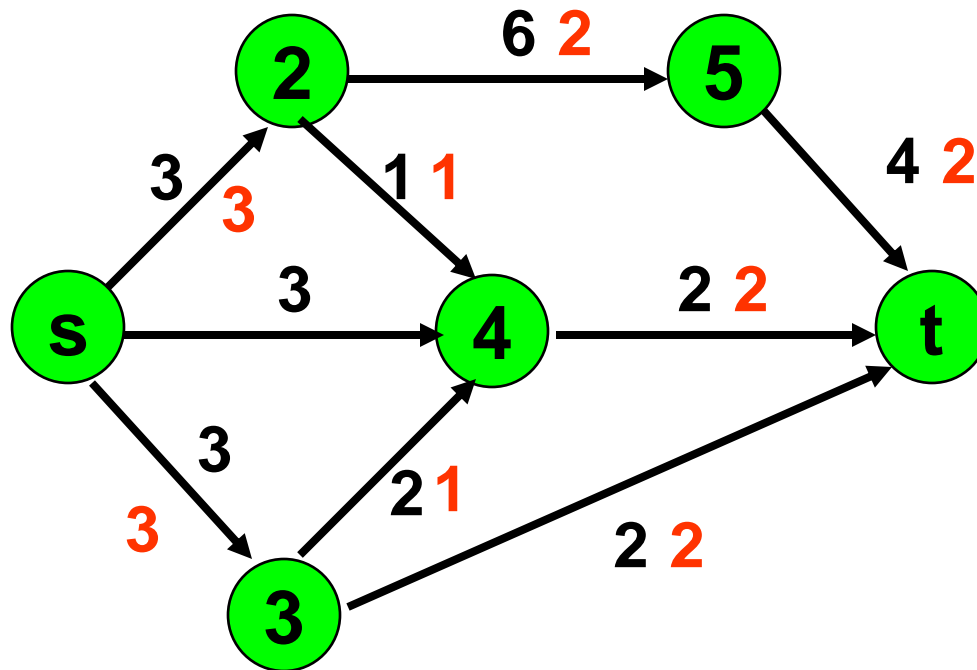
6.3.2. Thuật toán Ford-Fulkerson

$P_3: s \rightarrow 3 \rightarrow t, \partial_3 = 2.$



6.3.2. Thuật toán Ford-Fulkerson

$P_4: s \rightarrow 3 \rightarrow 4 \rightarrow t, \partial_4 = 1.$



6.3.2. Thuật toán Ford-Fulkerson

$P_5: s \rightarrow 4 \leftarrow 2 \rightarrow 5 \rightarrow t, \partial_5 = 1.$

Hết đường tăng luồng, $F_{\max} = 7$. Lát cắt cực tiểu $V_1 = \{s, 3, 4\}, V_2 = \{t, 2, 5\}$.

