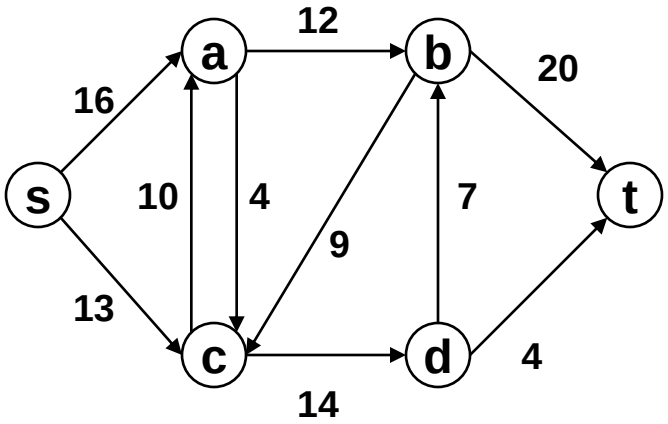
BÀI TOÁN TÌM LUỒNG CỰC ĐẠI TRÊN MẠNG (NETWORK FLOWS)

Nội dung

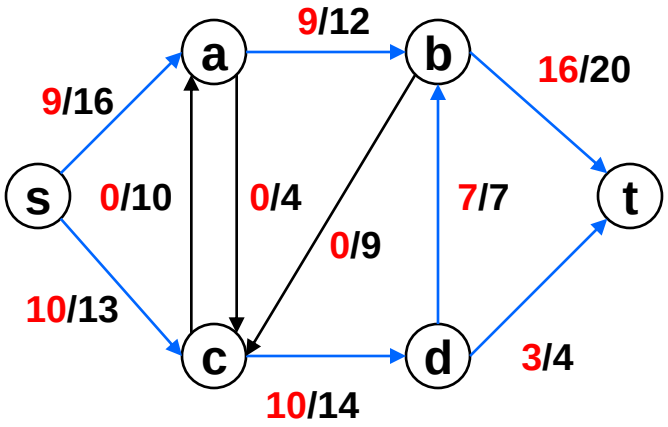
Bài toán vận chuyển dầu

Xét mạng lưới vận chuyển dầu, trong đó s là địa điểm nguồn (là nơi cung cấp dầu), t là địa điểm thu (là nơi nhận dầu để tiêu thụ) và các địa điểm trung gian, mỗi đường ống được gán một số dương cho biết *sức chứa* (lưu lượng dầu tối đa đi qua). Biết rằng tổng lượng dầu đi vào một địa điểm trung gian bằng tổng lượng dầu đi ra từ địa điểm trung gian này, hãy tìm một cách vận chuyển nhiều dầu nhất đến địa điểm t.

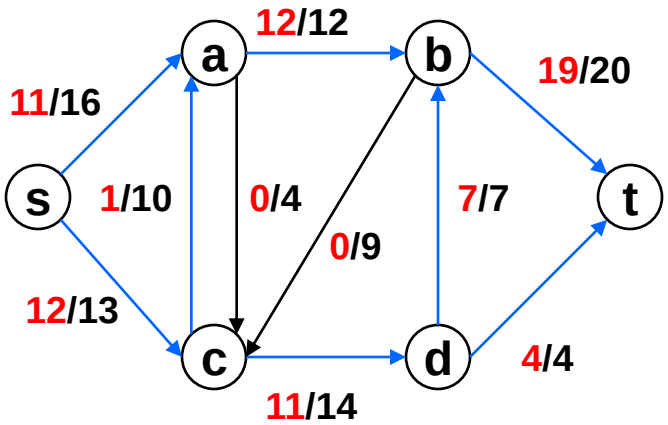


Mạng G

Luồng trên G



Luồng cực đại trên G



Một số khái niệm

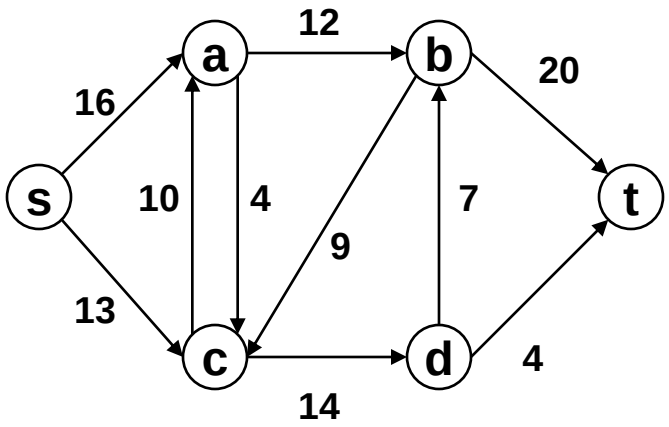
Mạng (network)

❖ Đồ thị có hướng G=(V,E) không có khuyên (cung nối từ đỉnh đến chính nó) với V là tập đỉnh, E là tập cạnh.

❖ Hai đỉnh phân biệt s, t, trong đó s là đỉnh phát (source) và t là đỉnh thu (sink).

❖ Mỗi cung (u,v) có một sức chứa (capacity) $c[u,v] \geq 0$.

❖ Nếu cung (u,v) không tồn tại thì gán c[u,v]=0.



Hãy cho biết sức chứa của các cung

(s,a), (a,s), (s,c), (c,b), (s,b), (b,s)

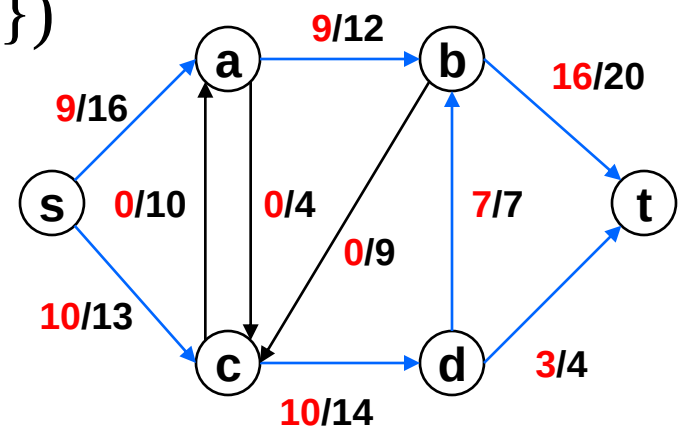
Một số khái niệm

Luồng dương Φ trên mạng G là một phép gán cho mỗi cung e=(u,v) một số thực không âm $\Phi(e) = \Phi[u,v]$ gọi là luồng dương trên cung e, thoả mãn 2 tính chất:

❖Tính chất 1: (Capacity constraint): Luồng trên mỗi cung không được vượt quá sức chứa của cung đó: $0 \le \Phi[u,v] \le c[u,v]$ (Với $\forall$ u, v $\in$ V).

❖Tính chất 2 (Flow conservation): Với mỗi đỉnh v không phải đỉnh phát và cũng không phải đỉnh thu, tổng luồng trên các cung đi vào v bằng tổng luồng trên các cung đi ra khỏi v:

$$\sum_{u \in V} \Phi[u,v] = \sum_{w \in V} \Phi[v,w] \quad (\text{Với } \forall v \in V \setminus \{s, t\})$$



Một số khái niệm

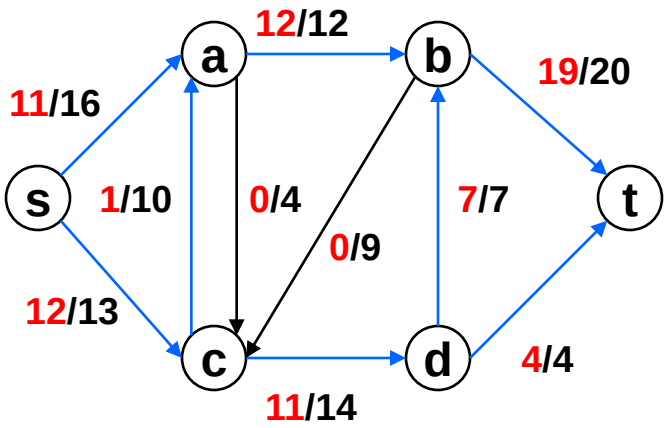
Giá trị luồng dương = tổng luồng trên các cung đi ra khỏi đỉnh phát trừ đi tổng luồng trên các cung đi vào đỉnh phát

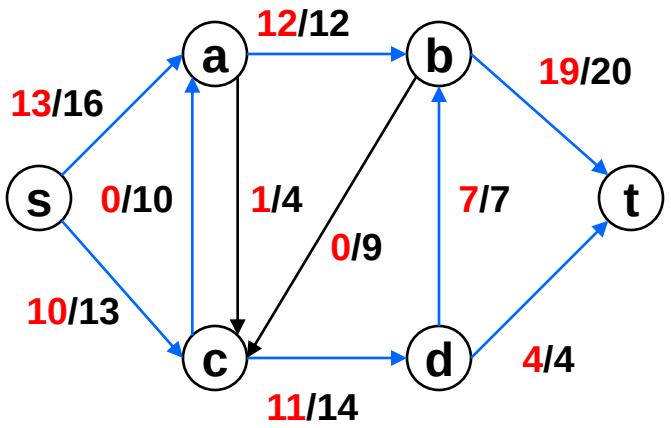
$$|\Phi| = \sum_{u \in V} \Phi[s,u] - \sum_{v \in V} \Phi[v,s]$$

Trường hợp không có cung đi vào đỉnh phát, giá trị luồng dương = tổng luồng trên các cung đi ra khỏi đỉnh phát.

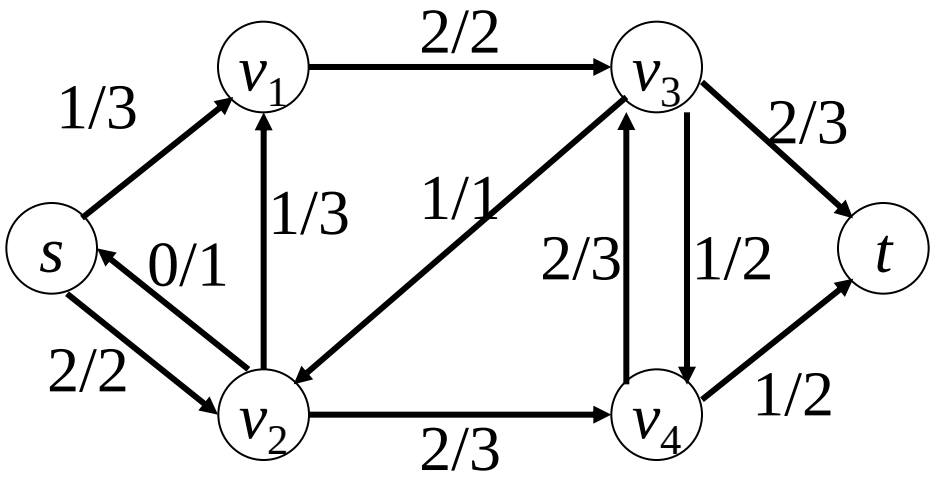
$$|\Phi| = \sum_{u \in V} \Phi[s,u]$$

Luồng dương có giá trị luồng lớn nhất gọi là luồng cực đại.





Một số khái niệm (ví dụ)



$$|\Phi| = \big(\Phi(s, v_1) + \Phi(s, v_2)\big) - \Phi(v_2, s) = (1+2) - 0 = 3$$

$$|\Phi| = \Phi(v_3, t) + \Phi(v_4, t) = 2 + 1 = 3$$

Một số khái niệm

Luồng f trên mạng G là một phép gán cho mỗi cung e=(u,v) một số thực f(e) = f[u,v] gọi là luồng trên cung e, thoả mãn 3 tính chất:

❖Tính chất 1 (Capacity constraint): Luồng trên mỗi cung không được vượt quá sức chứa của cung đó: $f[u,v] \leq c[u,v]$ (Với $\forall$ u, v $\in$ V).

❖Tính chất 2 (Skew symmetry): Với $\forall$ u, v$\in$ V, luồng trên cung (u, v) và luồng trên cung (v, u) có cùng giá trị tuyệt đối nhưng trái dấu nhau: $f[u,v] = -f[v,u]$.

❖Tính chất 3 (Flow conservation): Với mỗi đỉnh u không phải đỉnh phát và cũng không phải đỉnh thu, tổng luồng trên các cung đi ra khỏi u bằng 0: $\sum_{v \in V} f[u,v]=0$

Một số khái niệm

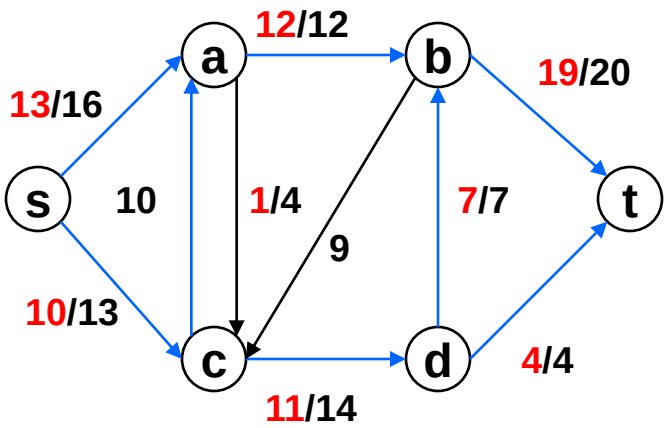
Giá trị luồng = tổng luồng trên các cung đi ra khỏi đỉnh phát

$$|f| = \sum_{u \in V} \Phi[f,u]$$

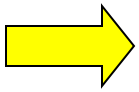
Luồng có giá trị luồng lớn nhất gọi là luồng cực đại.

Nếu có một luồng dương Φ trên G, ta xây dựng luồng f như sau

$$f[u, v] := \Phi[u, v] - \Phi[v, u]$$



Luồng dương Φ

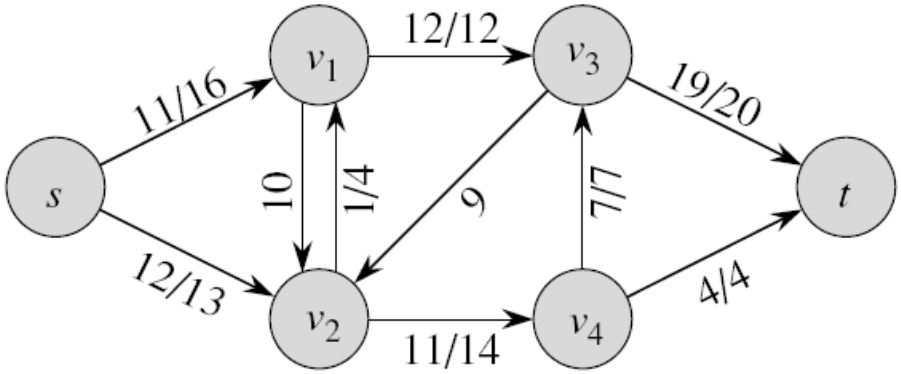


TÌM LUỒNG F?

Quy ước về hình vẽ

Trong các hình vẽ về luồng f, ta quy ước:

- Chỉ vẽ các cung (u,v) có c[u,v]>0
- Chỉ hiển thị các luồng f(u,v)>0 trên cung (u,v)



Ở hình vẽ trên, ta có

$c[v_1,s]=0$; $f[v_1,s]= - f[s,v_1]=-11$

$f[v_1,v_2]=-f[v_2,v_1]=-1$

$c[v_2,s]=0$; $f[v_2,s]=-f[s,v_2]=-12$

……

Một số khái niệm

Lát cắt (X, Y) là một cách phân hoạch tập đỉnh V thành 2 tập khác rỗng rời nhau X và Y, với $X \cap Y = \emptyset$, $X \cup Y = V$.

Lưu lượng lát cắt c(X,Y) xác định bởi công thức:

$$c(X,Y) = \sum_{u \in X,\, v \in Y} c[u,v]$$

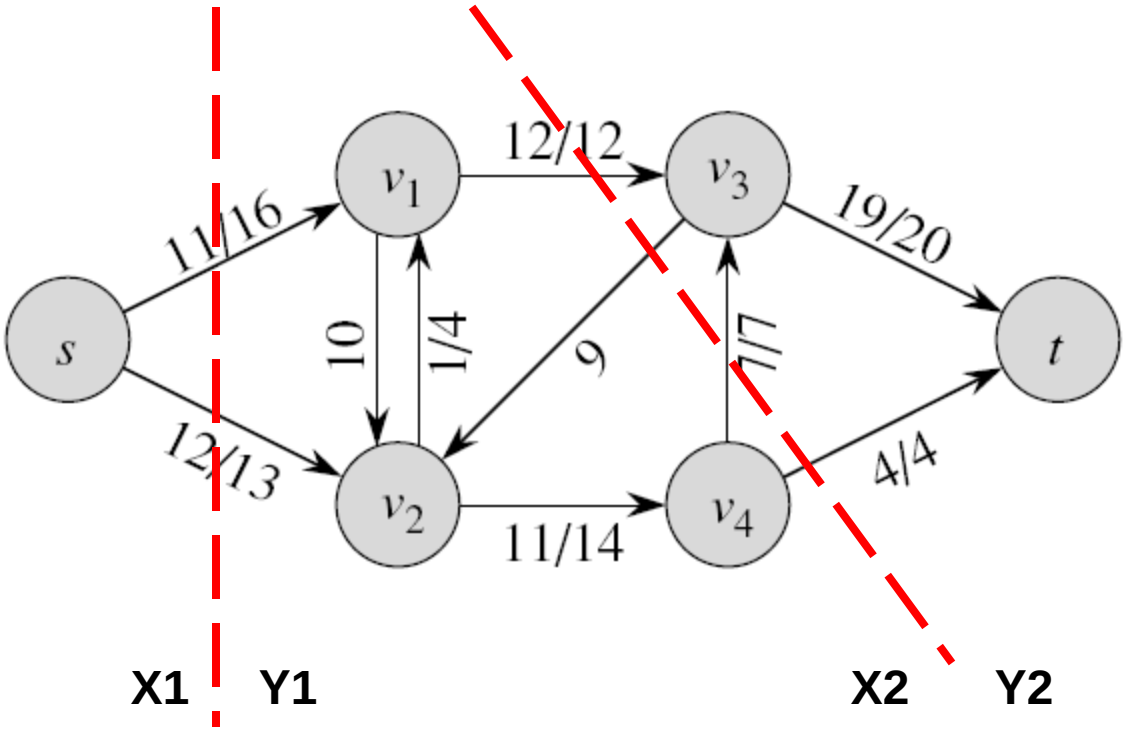
Luồng thông qua lát cắt f(X,Y) xác định bởi công thức:

$$f(X,Y) = \sum_{u \in X,\, v \in Y} f[u,v]$$

Lát cắt có X chứa s và Y chứa t gọi là lát cắt s-t.

Lát cắt s-t có lưu lượng nhỏ nhất (*bằng giá trị luồng cực đại*) gọi là lát cắt s-t hẹp nhất của mạng G.

Luồng cực đại và lát cắt hẹp nhất



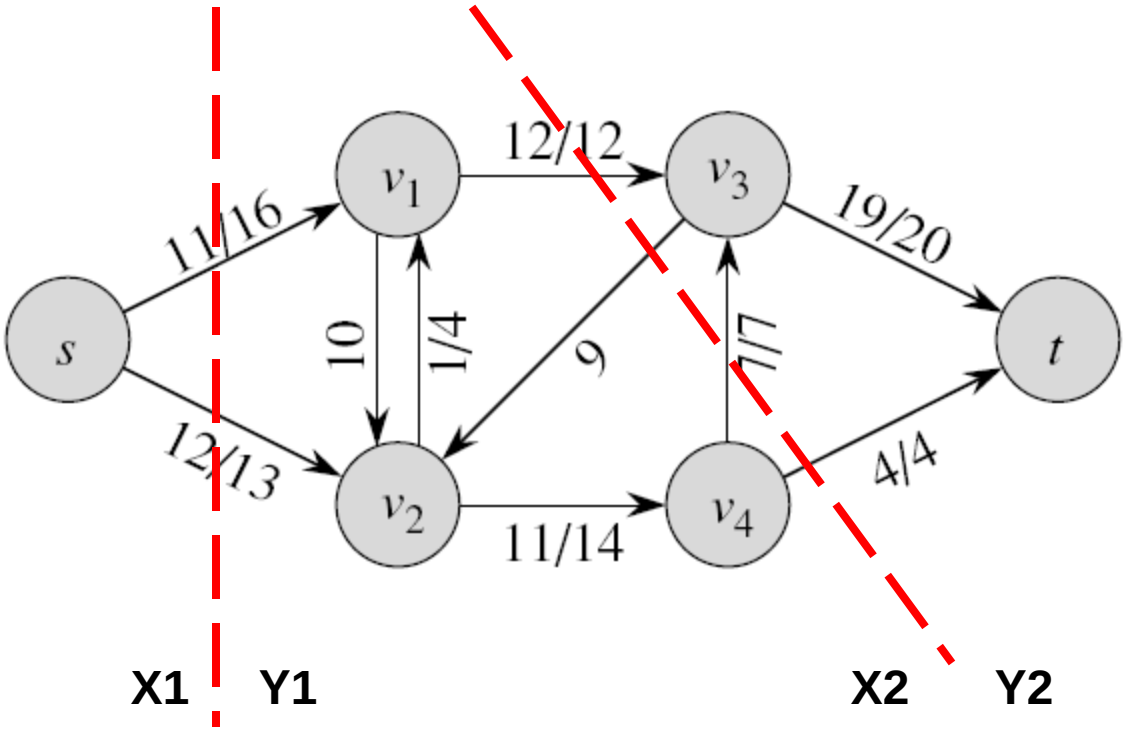
$(X_1, Y_1) = (\{s\}, \{v_1, v_2, v_3, v_4, t\})$

- $c(X_1, Y_1) = 16+13 = 29$
- $f(X_1, Y_1) = 11+12 = 23$

$(X_2, Y_2) = (\{s, v_1, v_2, v_4\}, \{v_3, t\})$

- $c(X_2, Y_2) = 12+7+4 = 23$
- $f(X_2, Y_2) = 12+7+4 = 23$

Luồng cực đại và lát cắt hẹp nhất



$(X_1, Y_1) = (\{s\}, \{v_1, v_2, v_3, v_4, t\})$

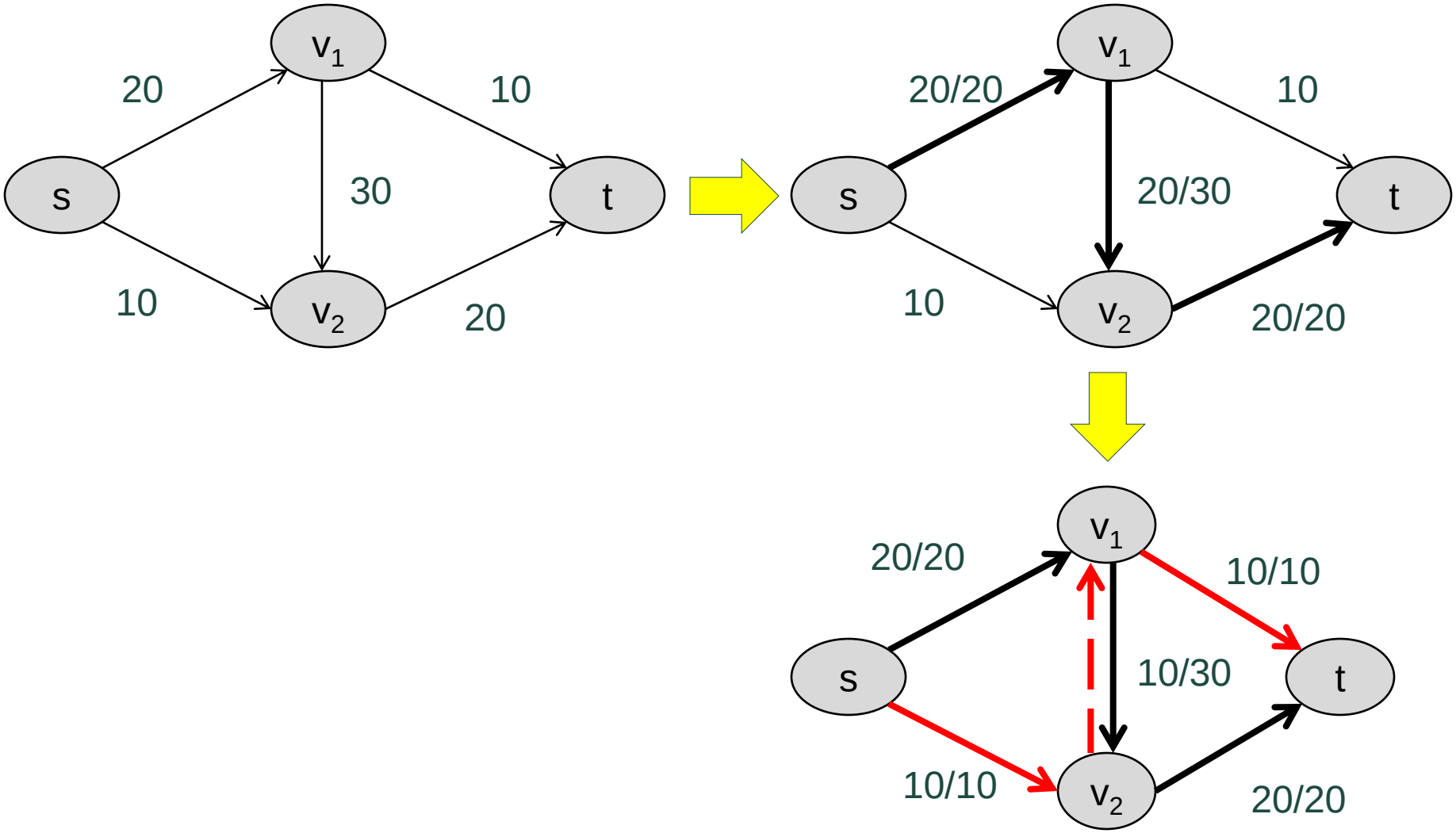
- $c(X_1, Y_1) = 16+13 = 29$
- $f(X_1, Y_1) = 11+12 = 23$

$(X_2, Y_2) = (\{s, v_1, v_2, v_4\}, \{v_3, t\})$

- $c(X_2, Y_2) = 12+7+4 = 23$
- $f(X_2, Y_2) = 12+7+4 = 23$

(X_2, Y_2) là lát cắt hẹp nhất
$c(X_2, Y_2) = f(X_2, Y_2) = 23$

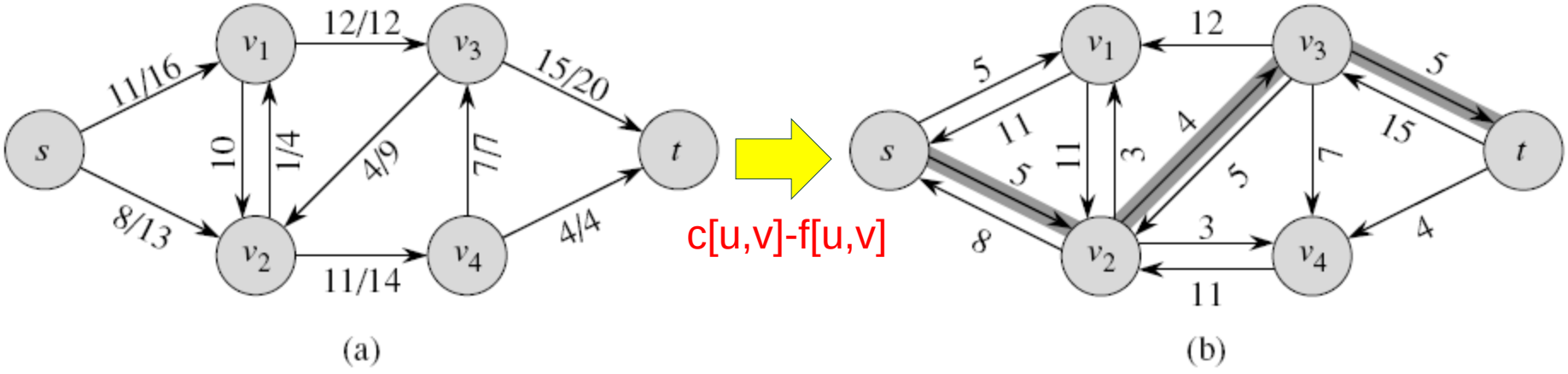
Ý tưởng


v1
20
10
s
30
t
10
v2
20
v1
20/20
10
s
20/30
t
10
v2
20/20
v1
20/20
10/10
s
10/30
t
10/10
v2
20/20

Mạng thặng dư (đồ thị tăng luồng)

Cho mạng G=(V, E) và f là một luồng trong mạng G.
Mạng thặng dư $G_f=(V, E_f)$ bao gồm tất cả các đỉnh của đồ thị G, và các cung được lấy từ G sau đi bỏ đi các cung bão hòa (một cung gọi là bão hoà nếu luồng trên cung đó đúng bằng sức chứa).
Mỗi cung (u,v) thuộc G_f có sức chứa là $c_f[u,v]=c[u,v] – f[u,v]$

Mạng thặng dư được xây dựng như trên còn gọi là đồ thị tăng luồng.



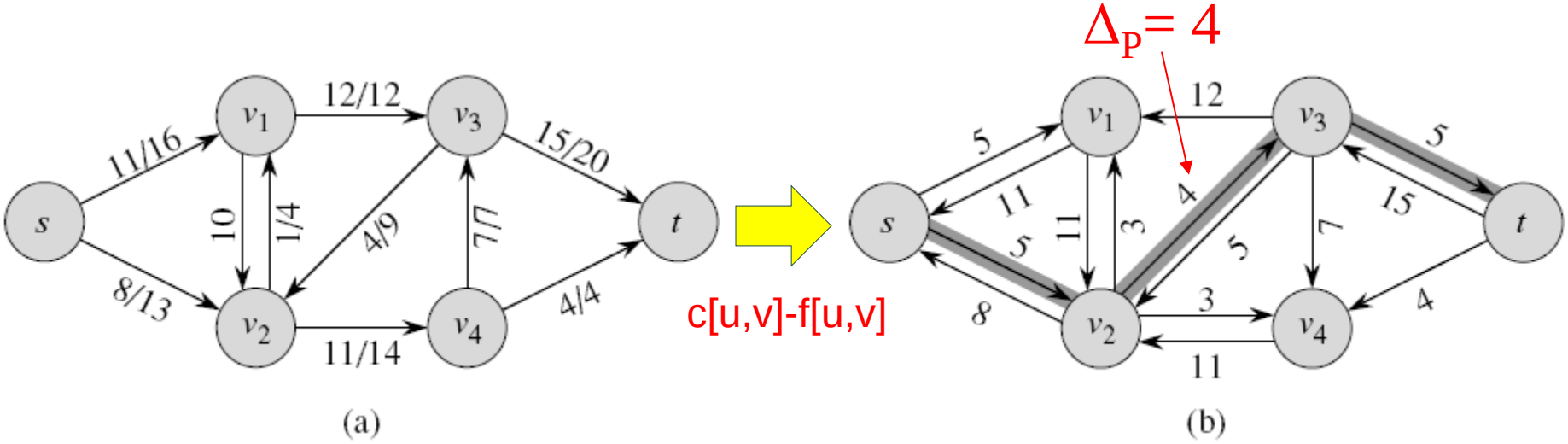
Mạng G

Mạng thặng dư G_f

Đường tăng luồng

Cho mạng G=(V,E) với đỉnh nguồn s, đỉnh thu t và luồng f, một đường đi đơn từ s tới t trên mạng thặng dư G_f gọi là một đường tăng luồng.

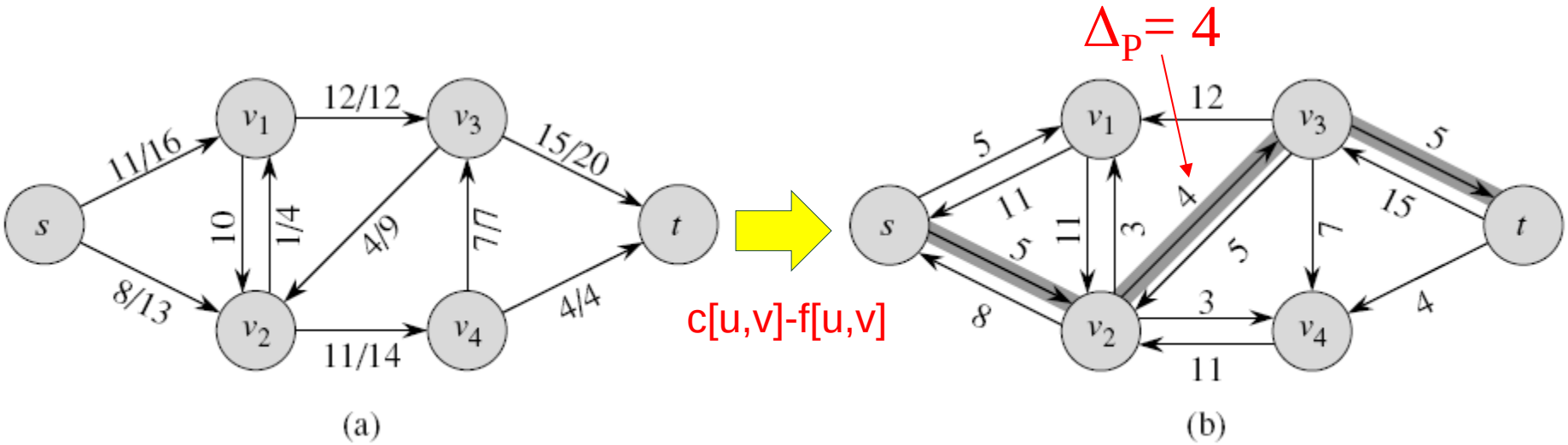
Với một đường tăng luồng P, ta đặt Δ_P là giá trị nhỏ nhất của sức chứa các cung trên P và gọi là giá trị thặng dư của đường P.



Mạng G

Mạng thặng dư G_f

Mạng thặng dư và đường tăng luồng

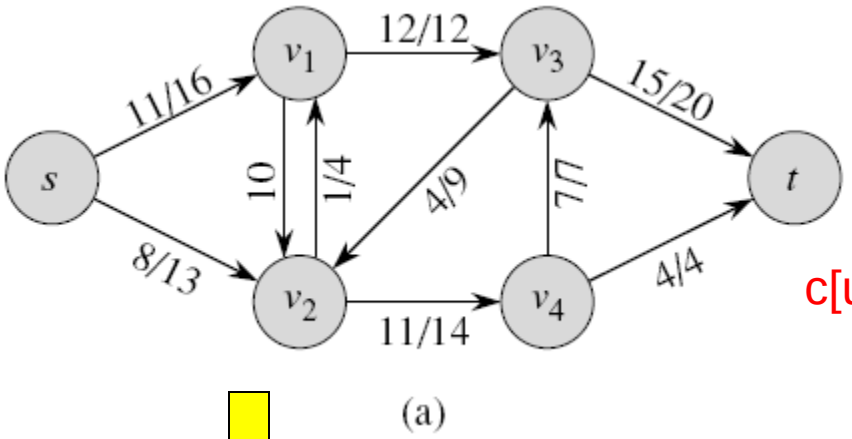


c[u,v]-f[u,v]

Mạng G

Mạng thặng dư G_f

Mạng thặng dư và đường tăng luồng

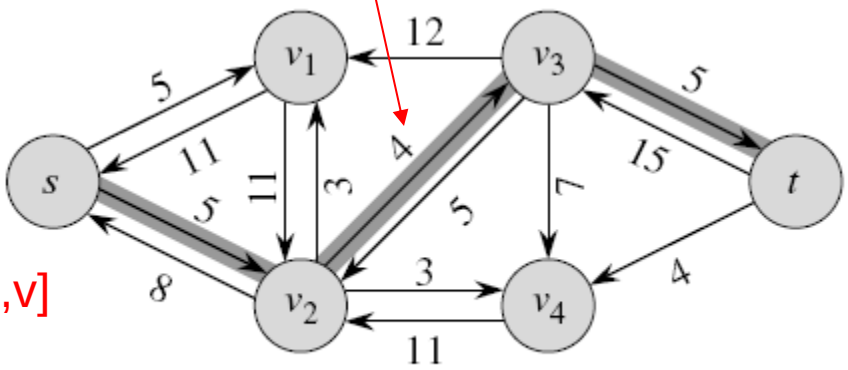


(a)

Mạng G

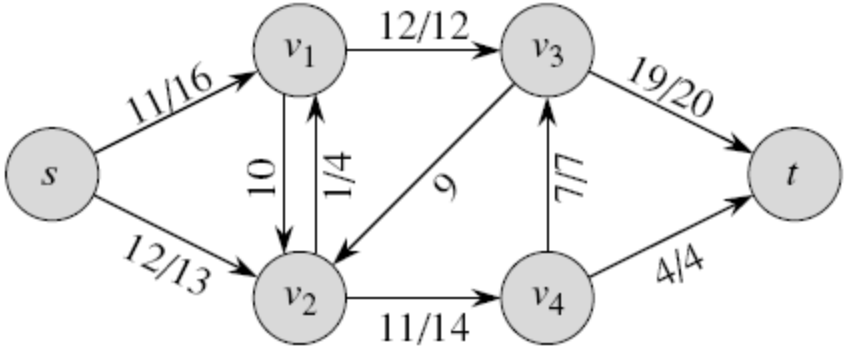
c[u,v]-f[u,v]

$\Delta_P = 4$



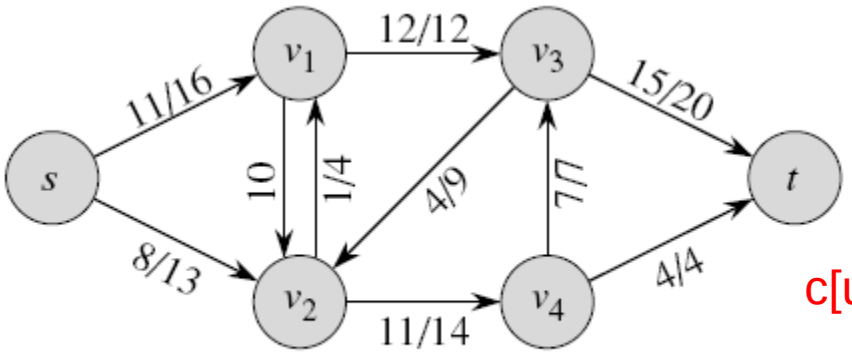
(b)

Mạng thặng dư G_f



(c)

Mạng thặng dư và đường tăng luồng

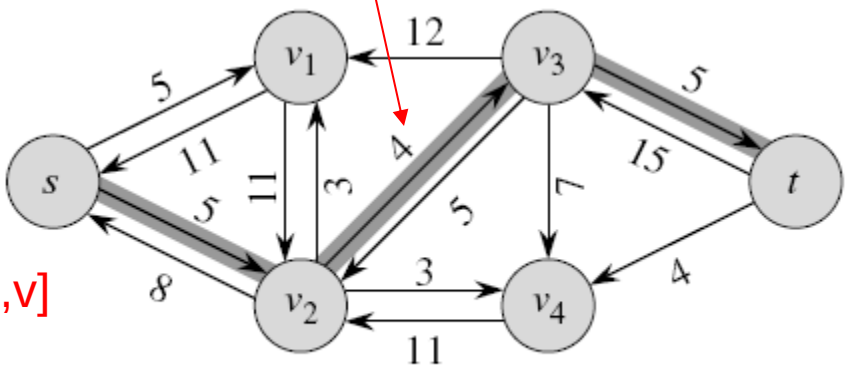


(a)

Mạng G

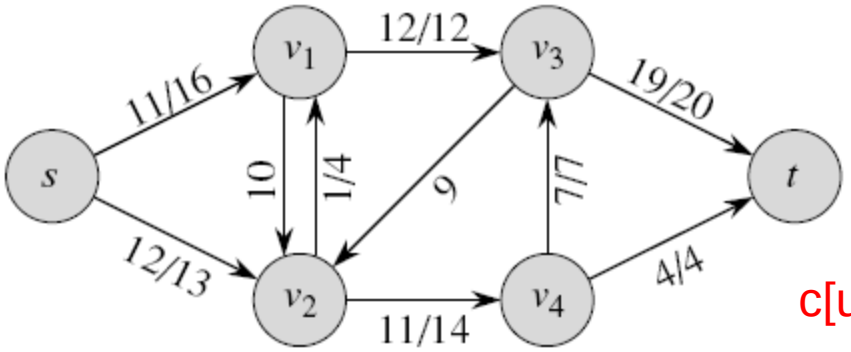
c[u,v]-f[u,v]

$\Delta_P = 4$



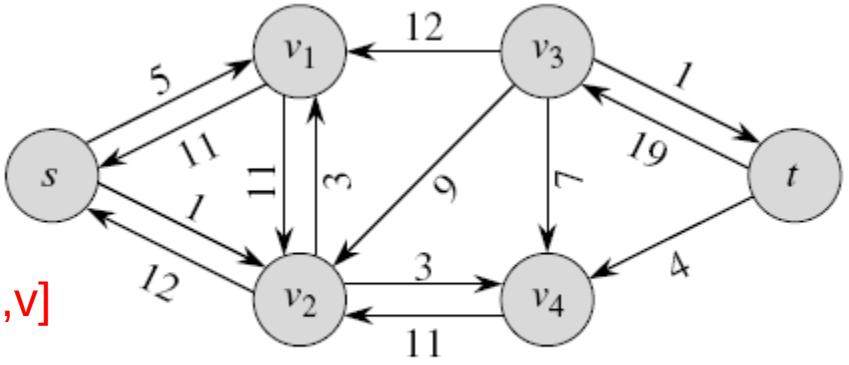
(b)

Mạng thặng dư G_f



(c)

c[u,v]-f[u,v]



(d)

Thuật toán Ford Fulkerson

Thuật toán Ford-Fulkerson tìm luồng cực đại trên mạng G dựa trên cơ chế tăng luồng dọc theo đường tăng luồng.
Bắt đầu từ một luồng f với giá trị luồng trên mọi cung đều bằng 0. Tại mỗi bước lặp thực hiện tìm một *đường tăng luồng* trên mạng thặng dư $G_{f.}$, sau đó thực hiện tăng luồng f dọc theo đường tăng luồng này. Quá trình này lặp lại cho đến khi không tìm thấy đường tăng luồng nào. Khi đó giá trị của luồng f là giá trị luồng cực đại.

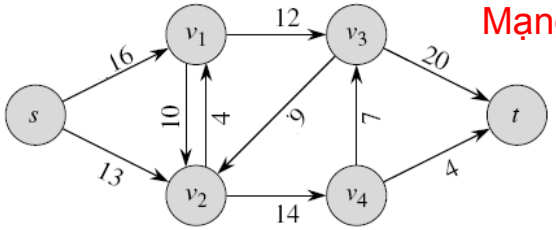
Ý tưởng thuật toán Ford Fulkerson

1. *Khởi tạo một luồng f giá trị 0*
2. **while** *Tìm được đường tăng luồng P trên* G_f
   do *Tăng luồng f dọc theo P*
3. **Output** f

Thuật toán Ford Fulkerson

Thuật toán Ford Fulkerson

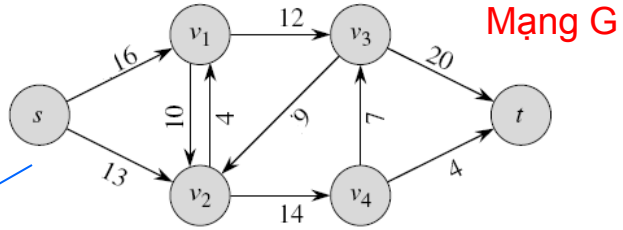
1. **For** *với mọi cung (u,v) trong G* **do** f[u,v]:=0
2. **While** *Tìm được một đường tăng luồng p trên* G_f **do**
   Begin
      Tính Δ_P *là giá trị nhỏ nhất của sức chứa các cung trên p*
      for *với mọi cung (u,v) thuộc đường tăng luồng p trong G* **do**
      Begin
         f[u,v]:=f[u,v] + Δ_P
         f[v,u]:=f[v,u] – Δ_P
      End;
   End;
3. **Output** f



Thuật toán Ford Fulkerson

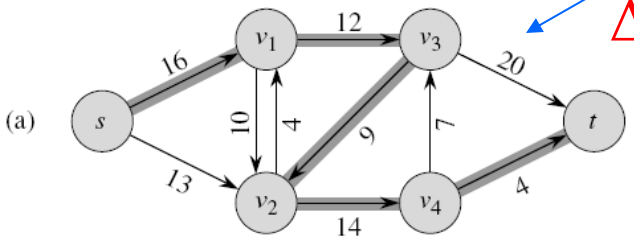
Thuật toán Ford Fulkerson

1. **For** *với mọi cung (u,v) trong G* **do** f[u,v]:=0
2. **While** *Tìm được một đường tăng luồng p trên* G_f **do**
   Begin
      Tính Δ_P *là giá trị nhỏ nhất của sức chứa các cung trên p*
      for *với mọi cung (u,v) thuộc đường tăng luồng p trong G* **do**
      Begin
         f[u,v]:=f[u,v] + Δ_P
         f[v,u]:=f[v,u] – Δ_P
      End;
   End;
3. **Output** f



c[u,v]-f[u,v]

Δ_P= 4

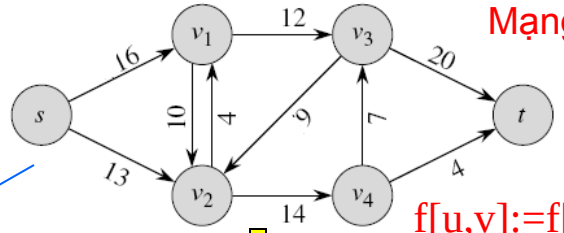


Mạng thặng dư G_f

Thuật toán Ford Fulkerson

Thuật toán Ford Fulkerson

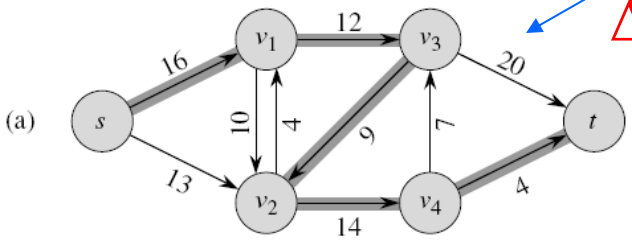
1. **For** *với mọi cung (u,v) trong G* **do** f[u,v]:=0
2. **While** *Tìm được một đường tăng luồng p trên* G_f **do**
   Begin
      Tính Δ_P *là giá trị nhỏ nhất của sức chứa các cung trên p*
      for *với mọi cung (u,v) thuộc đường tăng luồng p trong G* **do**
      Begin
         f[u,v]:=f[u,v] + Δ_P
         f[v,u]:=f[v,u] – Δ_P
      End;
   End;
3. **Output** f



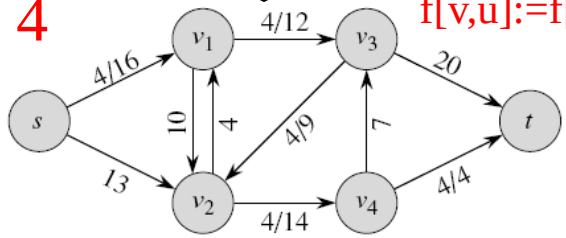
c[u,v]-f[u,v]

Δ_P= 4

f[u,v]:=f[u,v] + 4
f[v,u]:=f[v,u] – 4



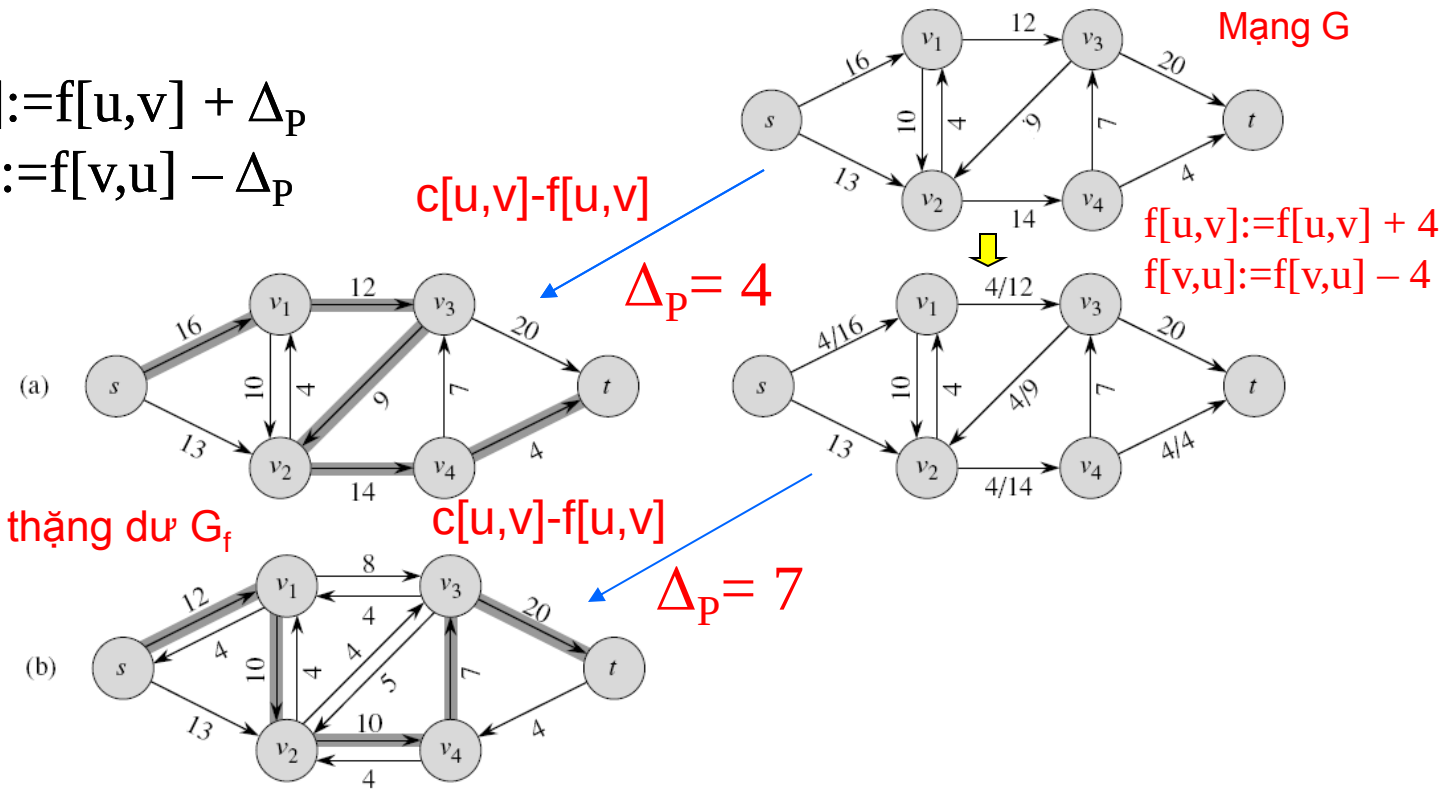
Mạng thặng dư G_f



Thuật toán Ford Fulkerson

Thuật toán Ford Fulkerson

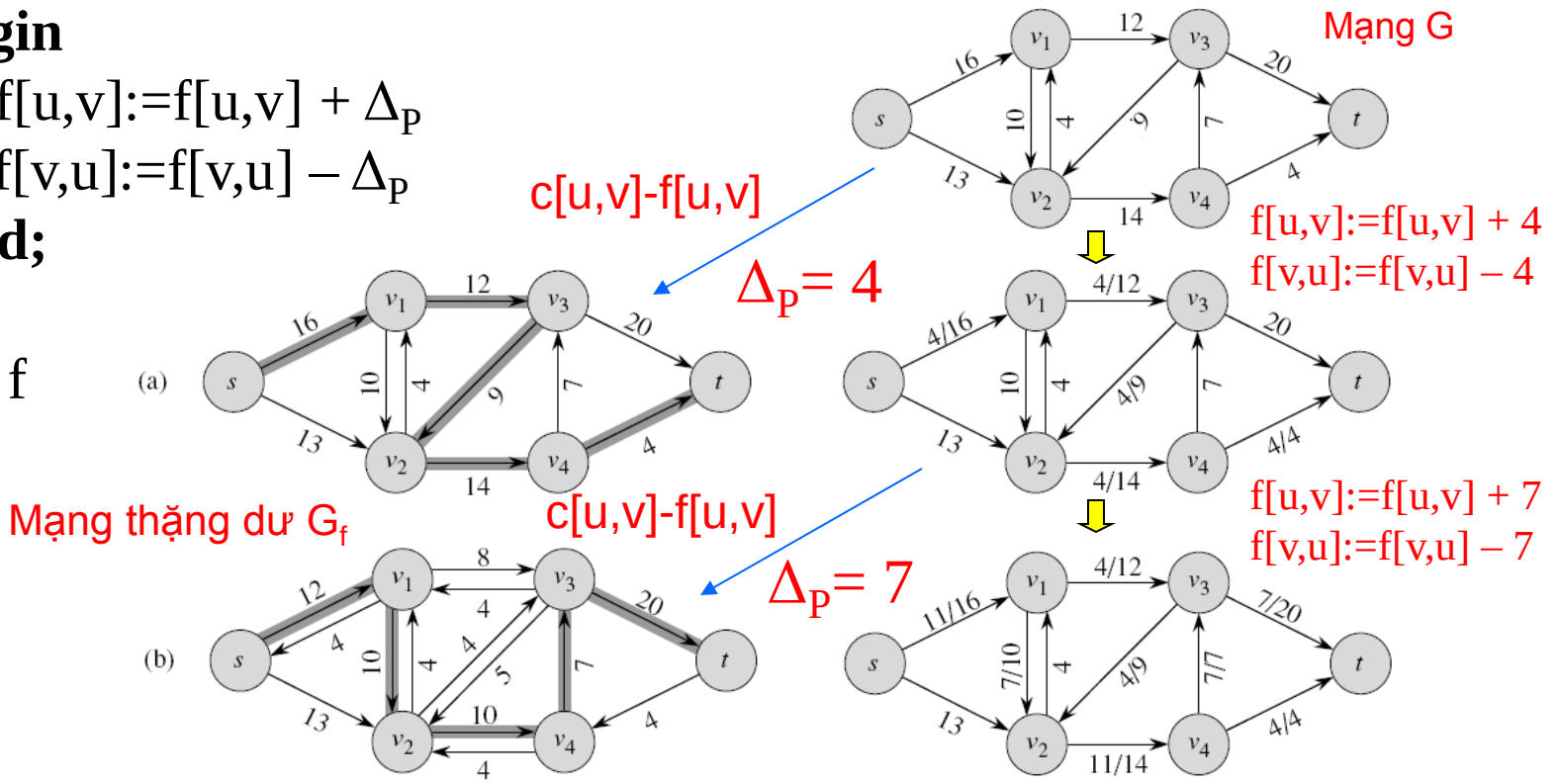
1. **For** *với mọi cung (u,v) trong G* **do** f[u,v]:=0
2. **While** *Tìm được một đường tăng luồng p trên* G_f **do**
   Begin
      Tính Δ_P *là giá trị nhỏ nhất của sức chứa các cung trên p*
      for *với mọi cung (u,v) thuộc đường tăng luồng p trong G* **do**
      Begin
         f[u,v]:=f[u,v] + Δ_P
         f[v,u]:=f[v,u] – Δ_P
      End;
   End;
3. **Output** f



Thuật toán Ford Fulkerson

Thuật toán Ford Fulkerson

1. **For** *với mọi cung (u,v) trong G* **do** f[u,v]:=0
2. **While** *Tìm được một đường tăng luồng p trên* G_f **do**
   Begin
      Tính Δ_P *là giá trị nhỏ nhất của sức chứa các cung trên p*
      for *với mọi cung (u,v) thuộc đường tăng luồng p trong G* **do**
      Begin
         f[u,v]:=f[u,v] + Δ_P
         f[v,u]:=f[v,u] – Δ_P
      End;
   End;
3. **Output** f

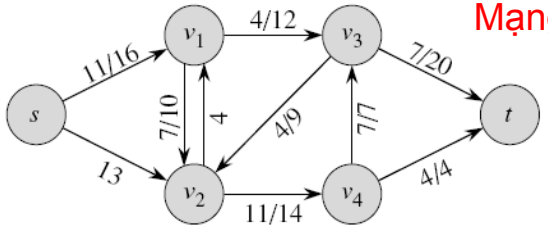


Thuật toán Ford Fulkerson

Thuật toán Ford Fulkerson

1. **For** *với mọi cung (u,v) trong G* **do** f[u,v]:=0
2. **While** *Tìm được một đường tăng luồng p trên* G_f **do**
   Begin
      Tính Δ_P *là giá trị nhỏ nhất của sức chứa các cung trên p*
      for *với mọi cung (u,v) thuộc đường tăng luồng p trong G* **do**
      Begin
         f[u,v]:=f[u,v] + Δ_P
         f[v,u]:=f[v,u] – Δ_P
      End;
   End;
3. **Output** f

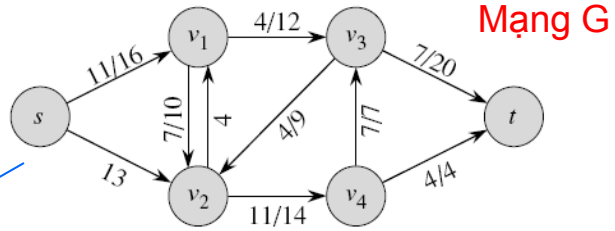
Mạng G



Thuật toán Ford Fulkerson

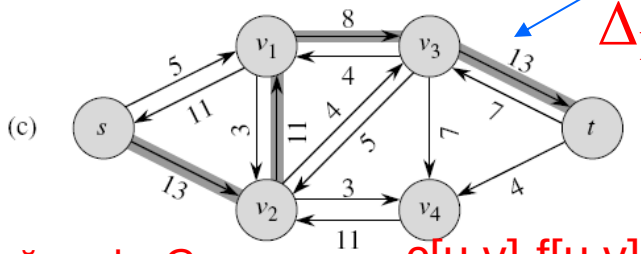
Thuật toán Ford Fulkerson

1. **For** *với mọi cung (u,v) trong G* **do** f[u,v]:=0
2. **While** *Tìm được một đường tăng luồng p trên* G_f **do**
   Begin
      Tính Δ_P *là giá trị nhỏ nhất của sức chứa các cung trên p*
      for *với mọi cung (u,v) thuộc đường tăng luồng p trong G* **do**
      Begin
         f[u,v]:=f[u,v] + Δ_P
         f[v,u]:=f[v,u] – Δ_P
      End;
   End;
3. **Output** f



c[u,v]-f[u,v]

Δ_P= 8



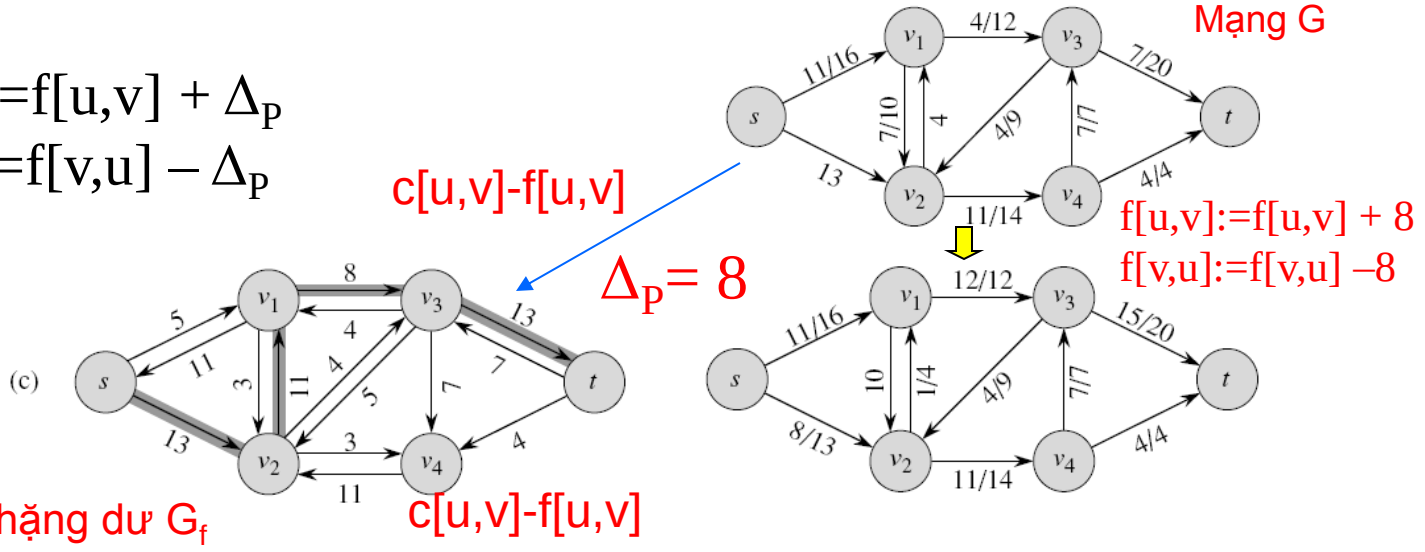
Mạng thặng dư G_f

c[u,v]-f[u,v]

Thuật toán Ford Fulkerson

Thuật toán Ford Fulkerson

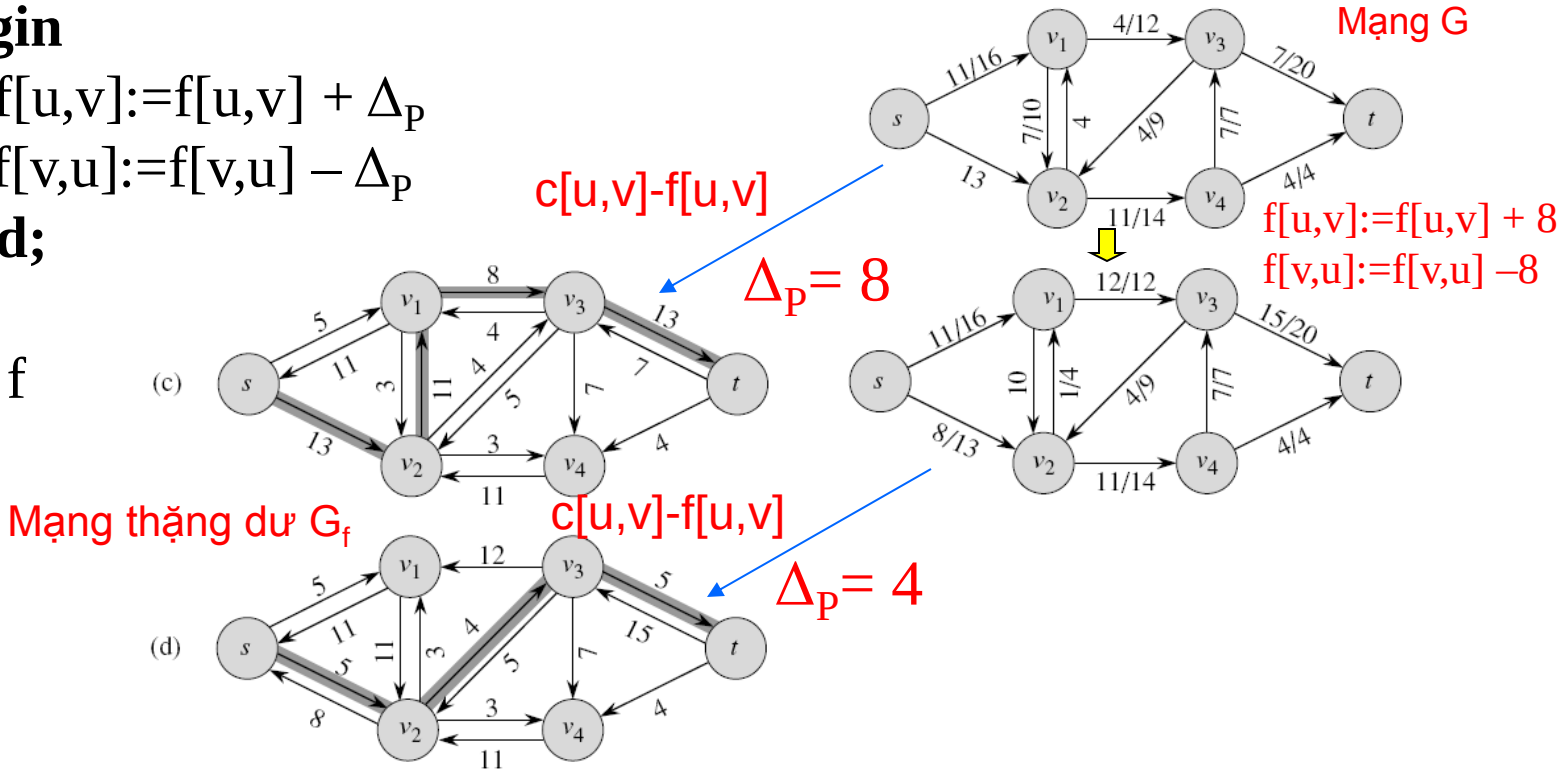
1. **For** *với mọi cung (u,v) trong G* **do** f[u,v]:=0
2. **While** *Tìm được một đường tăng luồng p trên* G_f **do**
   Begin
      Tính Δ_P *là giá trị nhỏ nhất của sức chứa các cung trên p*
      for *với mọi cung (u,v) thuộc đường tăng luồng p trong G* **do**
      Begin
         f[u,v]:=f[u,v] + Δ_P
         f[v,u]:=f[v,u] – Δ_P
      End;
   End;
3. **Output** f



Thuật toán Ford Fulkerson

Thuật toán Ford Fulkerson

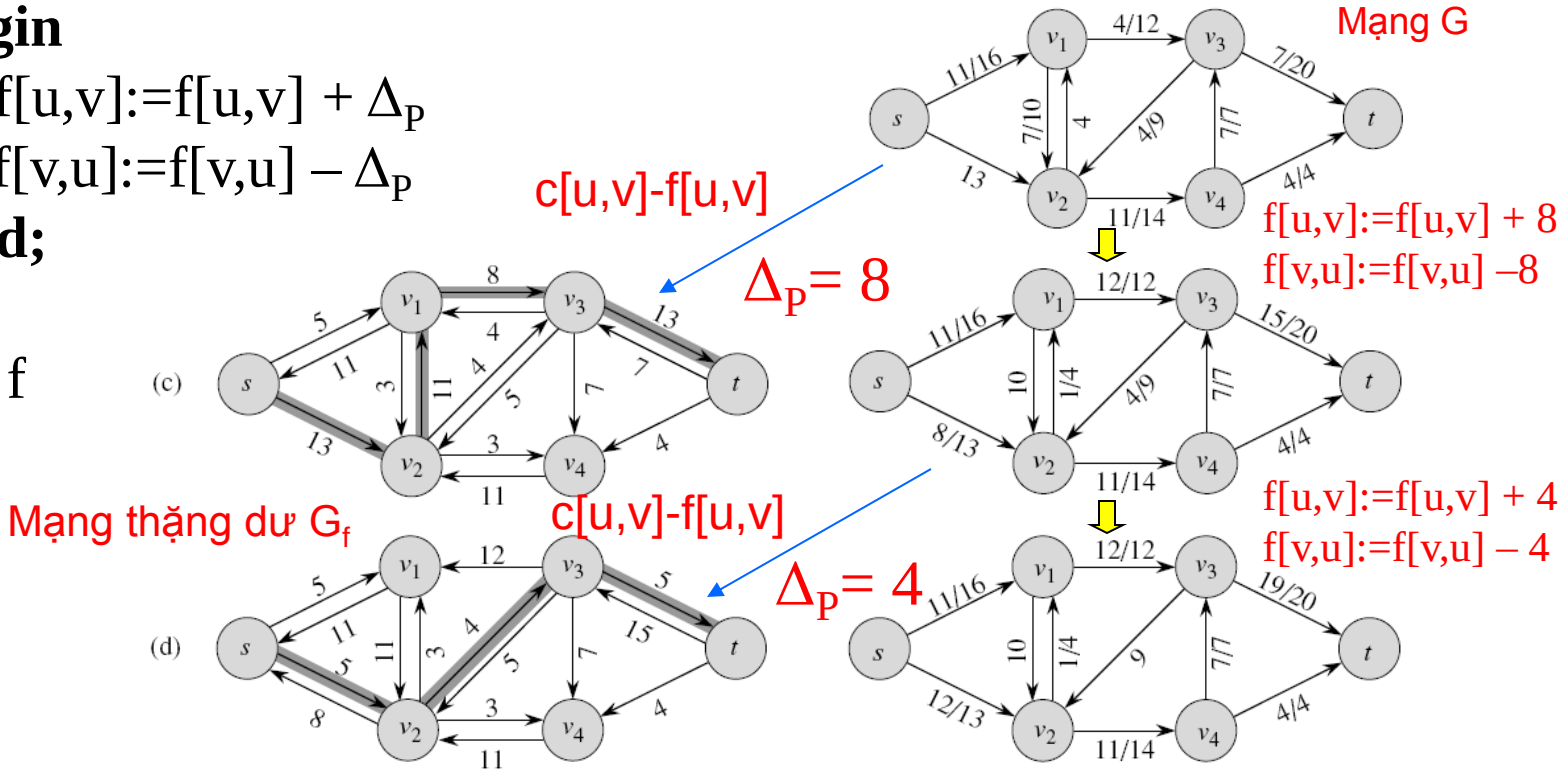
1. **For** *với mọi cung (u,v) trong G* **do** f[u,v]:=0
2. **While** *Tìm được một đường tăng luồng p trên* G_f **do**
   Begin
      Tính Δ_P *là giá trị nhỏ nhất của sức chứa các cung trên p*
      for *với mọi cung (u,v) thuộc đường tăng luồng p trong G* **do**
      Begin
         f[u,v]:=f[u,v] + Δ_P
         f[v,u]:=f[v,u] – Δ_P
      End;
   End;
3. **Output** f



Thuật toán Ford Fulkerson

Thuật toán Ford Fulkerson

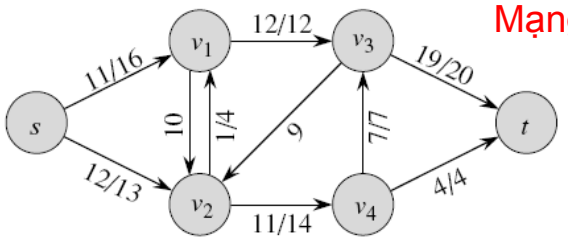
1. **For** *với mọi cung (u,v) trong G* **do** f[u,v]:=0
2. **While** *Tìm được một đường tăng luồng p trên* G_f **do**
   Begin
      Tính Δ_P *là giá trị nhỏ nhất của sức chứa các cung trên p*
      for *với mọi cung (u,v) thuộc đường tăng luồng p trong G* **do**
      Begin
         f[u,v]:=f[u,v] + Δ_P
         f[v,u]:=f[v,u] – Δ_P
      End;
   End;
3. **Output** f



Thuật toán Ford Fulkerson

Thuật toán Ford Fulkerson

1. **For** *với mọi cung (u,v) trong G* **do** f[u,v]:=0
2. **While** *Tìm được một đường tăng luồng p trên* G_f **do**
   Begin
      Tính Δ_P *là giá trị nhỏ nhất của sức chứa các cung trên p*
      for *với mọi cung (u,v) thuộc đường tăng luồng p trong G* **do**
      Begin
         f[u,v]:=f[u,v] + Δ_P
         f[v,u]:=f[v,u] – Δ_P
      End;
   End;
3. **Output** f

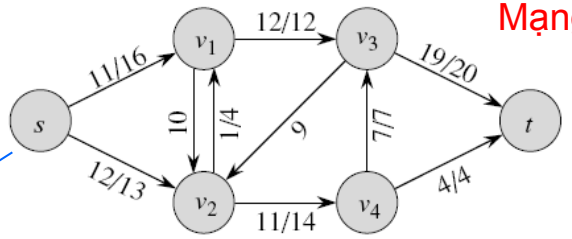


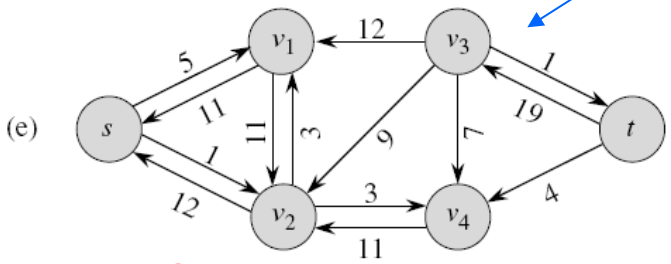
Thuật toán Ford Fulkerson

Thuật toán Ford Fulkerson

1. **For** *với mọi cung (u,v) trong G* **do** f[u,v]:=0
2. **While** *Tìm được một đường tăng luồng p trên* G_f **do**
   Begin
      Tính Δ_P *là giá trị nhỏ nhất của sức chứa các cung trên p*
      for *với mọi cung (u,v) thuộc đường tăng luồng p trong G* **do**
      Begin
         f[u,v]:=f[u,v] + Δ_P
         f[v,u]:=f[v,u] – Δ_P
      End;
   End;
3. **Output** f

Mạng G





Mạng thặng dư G_f

Không tìm thấy đường tăng luồng, thuật toán kết thúc

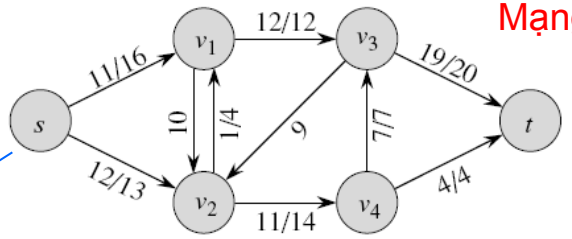
Thuật toán Ford Fulkerson

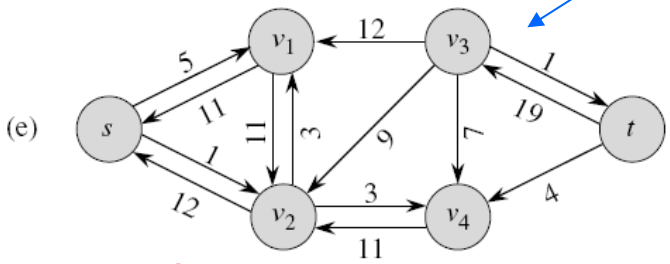
Thuật toán Ford Fulkerson

1. **For** *với mọi cung (u,v) trong G* **do** f[u,v]:=0
2. **While** *Tìm được một đường tăng luồng p trên* G_f **do**
   Begin
      Tính Δ_P *là giá trị nhỏ nhất của sức chứa các cung trên p*
      for *với mọi cung (u,v) thuộc đường tăng luồng p trong G* **do**
      Begin
         f[u,v]:=f[u,v] + Δ_P
         f[v,u]:=f[v,u] – Δ_P
      End;
   End;
3. **Output** f

Dùng BFS để tìm đường tăng luồng từ s đến t

Mạng G





Mạng thặng dư G_f

Không tìm thấy đường tăng luồng, thuật toán kết thúc

Thuật toán Ford Fulkerson

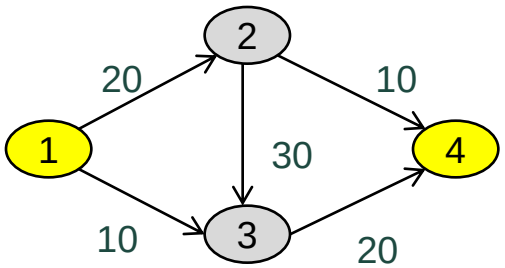
Bài toán

Input: File văn bản MAXFLOW.INP. Trong đó:

❖ Dòng 1: Chứa số đỉnh n (≤ 1000), số cạnh m của đồ thị, đỉnh phát s, đỉnh thu t theo đúng thứ tự cách nhau ít nhất một dấu cách

❖ M dòng tiếp theo, mỗi dòng gồm ba số u, v, c[u, v] cách nhau một dấu cách cho biết có cung (u, v) trong mạng và sức chứa của cung đó là c[u,v] (c[u, v] là số nguyên dương không quá 10^6)

Output: File văn bản MAXFLOW.OUT, ghi luồng trên các cung và giá trị luồng cực đại tìm được



MAXFLOW.INP	MAXFLOW.OUT
4 5 1 4	f[1, 2] = 20
1 2 20	f[1, 3] = 10
1 3 10	f[2, 3] = 10
2 3 30	f[2, 4] = 10
2 4 10	f[3, 4] = 20
3 4 20	Max Flow: 30

Thuật toán Ford Fulkerson

Ta sử dụng một số biến với các ý nghĩa như sau

❖ c[1..n, 1..n]: Ma trận biểu diễn sức chứa của các cung trên mạng

❖ f[1..n, 1..n]: Ma trận biểu diễn luồng trên các cung

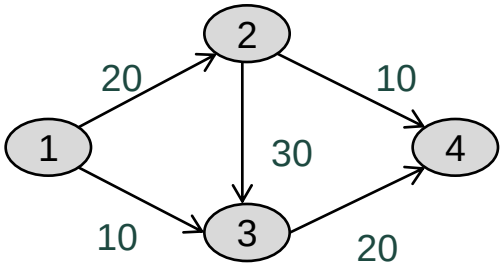
❖ trace[1..n]: Dùng để lưu vết đường tăng luồng, thuật toán tìm đường tăng luồng sẽ sử dụng là thuật toán tìm kiếm theo chiều rộng (BFS)

Ta có thể kiểm tra (u, v) có phải là cung trên mạng thặng dư G_f không bằng điều kiện: c[u,v]>f[u, v]. Nếu (u,v) là cung trên G_f thì sức chứa của nó là c[u,v] - f[u,v].

Thuật toán Ford Fulkerson

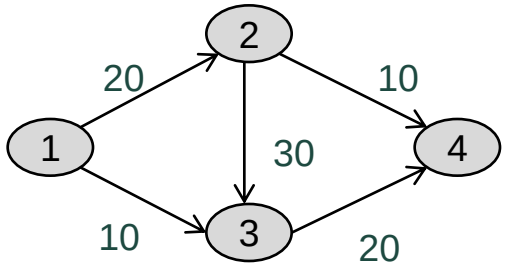
```
{Tìm luồng cực đại bằng thuật toán Ford Fullkerson với kỹ thuật BFS áp dụng ĐƠN ĐỒ THỊ CÓ HƯỚNG
- còn gọi là thuật toán Edmonds-Karp}
{$MODE OBJFPC}
const
  InputFile  = 'MAXFLOW.INP';
  OutputFile = 'MAXFLOW.OUT';
  max = 1000;
type
  TCapacities = array[1..max, 1..max] of Integer;
var
  c: TCapacities;
  f: TCapacities;
  Trace: array[1..max] of Integer;
  n, s, t: Integer;

procedure Enter;
var
  m, i, u, v: Integer;
  fi: Text;
begin
  Assign(fi, InputFile); Reset(fi);
  FillChar(c, SizeOf(c), 0);
  ReadLn(fi, n, m, s, t);
  for i := 1 to m do
    ReadLn(fi, u, v, c[u, v]);
  Close(fi);
end;
```

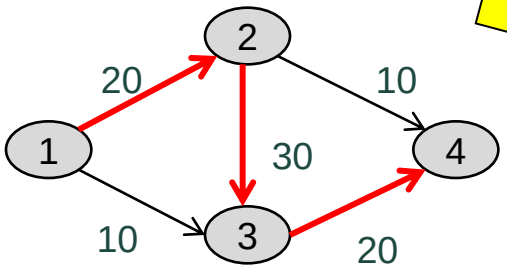


Thuật toán Ford Fulkerson

```
function FindPath: Boolean; {Tìm đường tăng luồng trên Gf bằng BFS, trả về True nếu tìm thấy}
var
  u, v: Integer;
  Queue: array[1..max] of Integer;
  Front, Rear: Integer;
begin
  FillChar(Trace, SizeOf(Trace), 0);
  Front := 1; Rear := 1;
  Queue[1] := s;
  Trace[s] := n + 1; {Trace[v] = 0 <=> v chưa thăm}
  repeat
    u := Queue[Front]; Inc(Front);
    for v := 1 to n do {Xét các đỉnh v chưa thăm kề u trên Gf}
      if (Trace[v] = 0) and (c[u, v] > f[u, v]) then //Xét các cung (u,v) chưa bão hòa, cf=c[u,v]-f[u,v]>0
        begin
          Trace[v] := u;
          if v = t then {Đến được t thì thuật toán dừng}
            begin
              FindPath := True; Exit;
            end;
          Inc(Rear); Queue[Rear] := v;
        end;
  until Front > Rear;
  FindPath := False;
end;
```

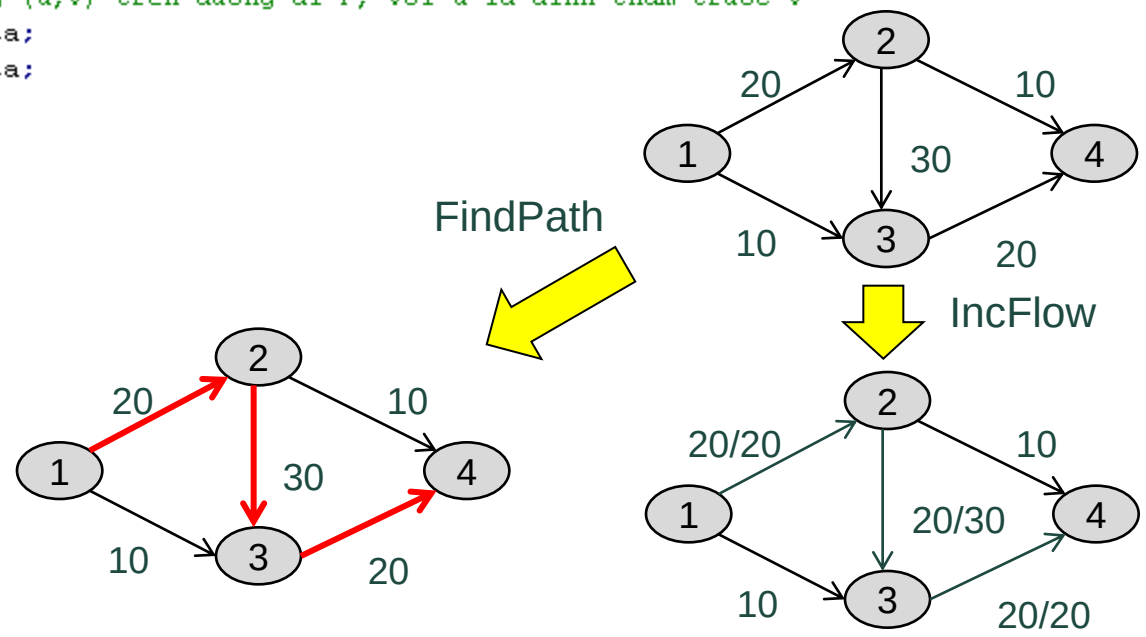


FindPath



Thuật toán Ford Fulkerson

```
procedure IncFlow; {Tăng luồng f dọc đường tăng luồng P}
var
  Delta, u, v: Integer;
begin
  {Tính Delta = ΔP}
  Delta := MaxInt;
  v := t; {Duyệt đường tăng luồng P từ đỉnh t đến đỉnh s}
  repeat
    u := Trace[v]; //Xét cung (u,v) trên đường đi P, với u là đỉnh thăm trước v
    if c[u, v] - f[u, v] < Delta then Delta := c[u, v] - f[u, v]; //Tìm giá trị cf[u,v] nhỏ nhất
    v := u;
  until v = s;
  {Tăng luồng f}
  v := t; {Duyệt đường tăng luồng P từ đỉnh t đến đỉnh s}
  repeat
    u := Trace[v]; //Xét cung (u,v) trên đường đi P, với u là đỉnh thăm trước v
    f[u, v] := f[u, v] + Delta;
    f[v, u] := f[v, u] - Delta;
    v := u;
  until v = s;
end;
```

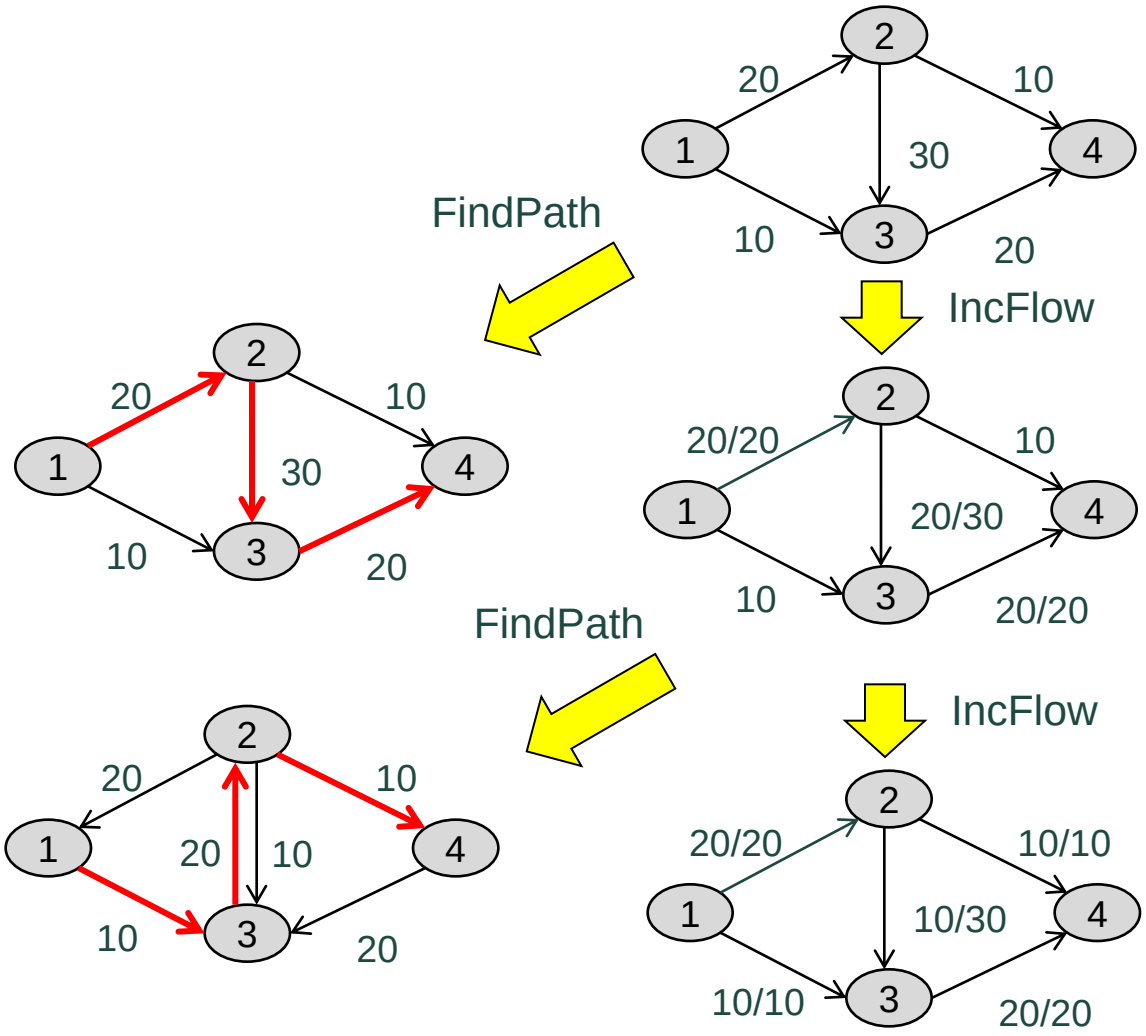


Thuật toán Ford Fulkerson

```
procedure PrintResult;
var
  u, v: Integer;
  m: Integer;
  fo: Text;
begin
  Assign(fo, OutputFile); Rewrite(fo);
  m := 0;
  for u := 1 to n do
    for v := 1 to n do
      if f[u, v] > 0 then {Chi quan tam den nhung cung co luong duong}
        begin
          WriteLn(fo, 'f[', u, ', ', v, '] = ', f[u, v]);
          if u = s then m := m + f[s, v];
        end;
  WriteLn(fo, 'Max Flow: ', m);
  Close(fo);
end;

begin
  Enter;
  FillChar(f, SizeOf(f), 0); //Khoi tao luong 0
  repeat
    if not FindPath then Break; //Thoat neu khong tim thay duong tang luong
    IncFlow; //Thuc hien tang luong f
  until False;
  PrintResult; //In ket qua
end.
```

Thuật toán Ford Fulkerson



Thank You !