Computer Science and Information Engineering
National Chi Nan University

Chapter 9
Connectivity

§ 9.1 Introduction

§ 9.1 Introduction

- **Def**: $G = (V, E)$ is a graph.

① **vertex connectivity of G between x and y**: ($xy \notin E(G)$)

VC(G, x, y) = min # of vertices whose removal separate x, y.

② **vertex connectivity of G**: ($G \neq K_n$ or let VC(K_n) = $n - 1$)

VC(G) = $\min_{\forall xy \notin E(G)}$ VC(G, x, y).

③ **edge connectivity of G between x and y**:

EC(G, x, y) = min # of edges whose removal separate x, y.

④ **edge connectivity of G**:

EC(G) = $\min_{\forall x,\, y \notin V(G)}$ EC(G, x, y).

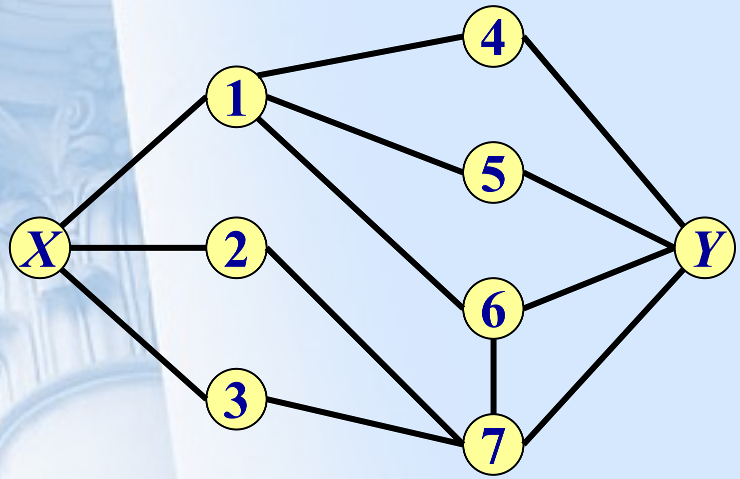
§ 9.1 Introduction

- **Menger's Theorem**:

① **VC(G, x, y) = max # vertex disjoint x-y paths. ($xy \notin E$)**

② **EC(G, x, y) = max # edge disjoint x-y paths.**

- **Ex**:

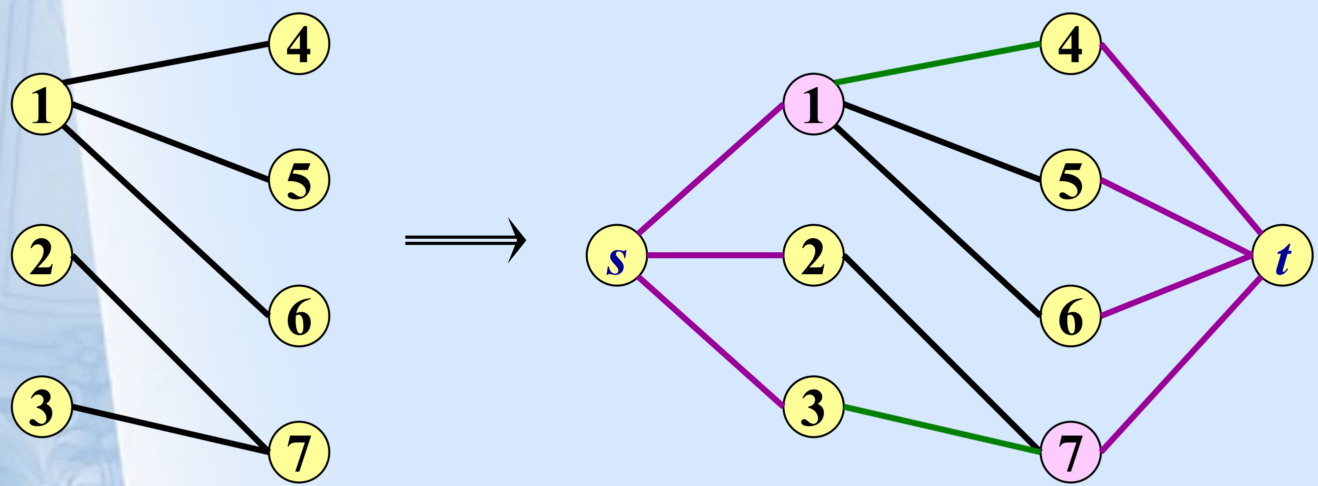


VC(G, x, y) = 2

EC(G, x, y) = 3

§ 9.1 Introduction

- **Ex:**



M^* (and C^*)

$\mathrm{VC}(G, s, t) = M^*$

Computer Science and Information Engineering
National Chi Nan University

Chapter 9
Connectivity

§ 9.2 Network

§ 9.2 Network

- **Def:**

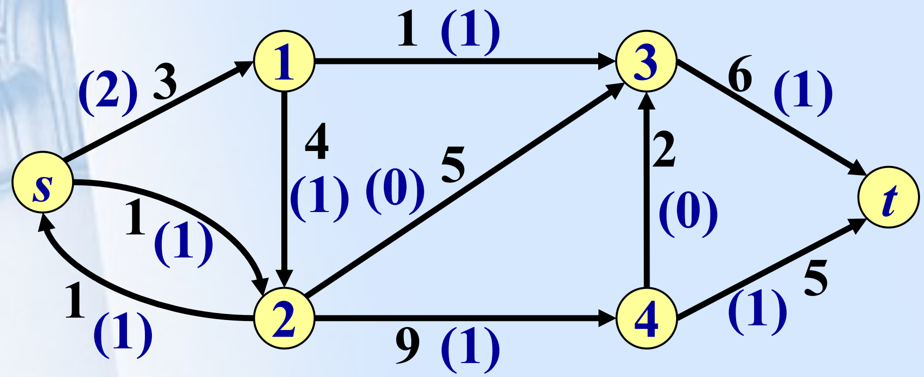
① **A network *N* is a digraph with two special vertices *s* and *t*, and each edge *e* has a capacity, Cap(*e*) ≥ 0.**

② **A flow of *N* is a function *f*: $E(N) \rightarrow R^+ \cup \{0\}$, such that**

(i) $\forall\ e \in E(N)$, $f(e) \leq \text{Cap}(e)$.

(ii) $\forall\ x \notin \{s, t\}$, $\sum_{\forall yx \in E(N)} f(y, x) = \sum_{\forall xz \in E(N)} f(x, z)$.

- **Ex:**



§ 9.2 Network

- **Notation:**

① **value(f)** $= \sum_{sz \in E} f(s, z) - \sum_{ys \in E} f(y, s)$.

② $A \subseteq V(N)$, $B \subseteq V(N)$.

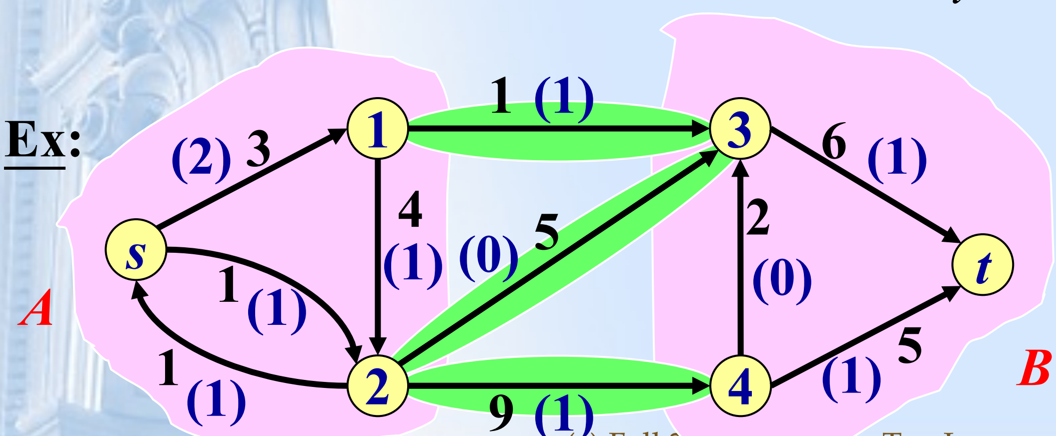
(A, B) $= \{ab \in E(N): a \in A, b \in B\}$. (Note: $A \cap B$ 不一定 $= \varnothing$)

③ **For any function g: $E(N) \to R$.**

We use $g(A, B)$ to denote $\sum_{e \in (A,B)} g(e)$.

So, we can use $f(V(N), \{x\})$ to denote $\sum_{yx \in E} f(y, x)$.

Ex:



value(f) = 2

$f(A, B) = 2$

Cap(A, B) = 15

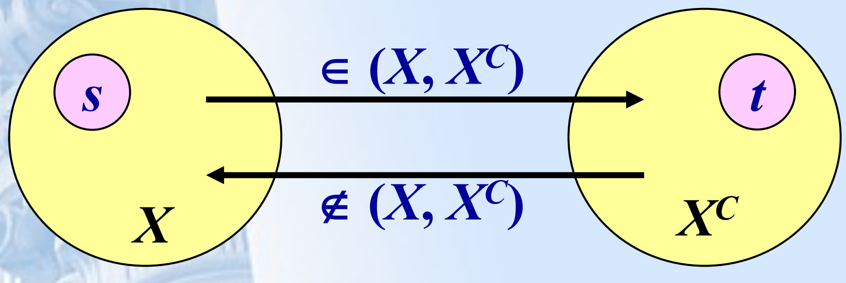
§ 9.2 Network

- **Def: For any network $N = (V, E)$ with s, t and Cap. function on E. An s-t cut B is edge set (X, X^C) where $s \in X$, $t \in X^C$.**

- **Note:**

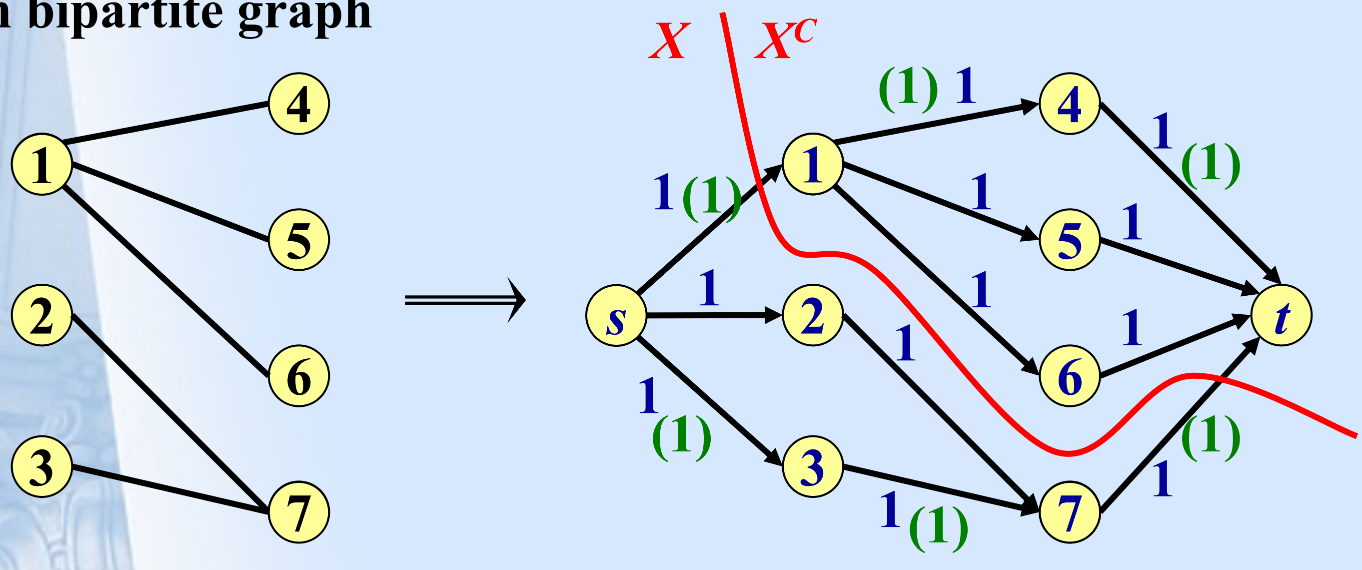
① **$\text{Cap}(X, X^C) = \sum_{e \in (X, X^C)} \text{Cap}(e)$.**

②



§ 9.2 Network

- **<u>Ex</u>: In bipartite graph**



- **<u>目標</u>: 找 $\max_f$ value(f) = $\min_X$ Cap(X, X^C).**

§ 9.2 Network

- 目標: 找 $\max_f$ value(f) = $\min_X$ Cap(X, X^C). (1/3)

  Step 1:

  証 Weakly Duality Inequality: value(f) ≤ Cap(X, X^C), ∀ f, X.

  (∴ $\max_f$ value(f) ≤ $\min_X$ Cap(X, X^C))

  ① Lemma: value(f) = $f(X, X^C) - f(X^C, X)$

  Proof.

$$
\begin{aligned}
\text{value}(f) &= f(s, V) - f(V, s) \\
&= f(s, V) - f(V, s) + \textstyle\sum_{v \in X - \{s\}} (f(v, V) - f(V, v)) \\
&= f(X, V) - f(V, X) \\
&= f(X, X) + f(X, X^C) - (f(X, X) + f(X^C, X)) \\
&= f(X, X^C) - f(X^C, X).
\end{aligned}
$$

§ 9.2 Network

- <u>目標</u>: 找 $\max_f$ value(f) = $\min_X$ Cap(X, X^C). (2/3)

  <u>Step 1</u>:

  証 <u>Weakly Duality Inequality</u>: value(f) ≤ Cap(X, X^C), ∀ f, X.

  (∴ $\max_f$ value(f) ≤ $\min_X$ Cap(X, X^C))

  ② Proof of W.D.I.

$$
\begin{aligned}
&\text{value}(f) - \text{Cap}(X, X^C) \\
&= (f(X, X^C) - f(X^C, X)) - \text{Cap}(X, X^C) \\
&= \underbrace{(f(X, X^C) - \text{Cap}(X, X^C))}_{\le 0} \underbrace{- f(X^C, X)}_{\le 0} \\
&\le 0.
\end{aligned}
$$

§ 9.2 Network

- 目標: 找 $\max_f$ value(f) = $\min_X$ Cap(X, X^C). (3/3)

  Step 2:

  目標: 給一 algorithm, 找到 f^* 及 (X^*, X^{*C}),
  使得 value(f^*) = Cap(X^*, X^{*C}).

  ⇒ Cap(X^*, X^{*C}) = value(f^*) ≤ $\max_f$ value(f) ≤ $\min_X$ Cap(X, X^C)
  (∵ Step 2) (∵ Step 2) (∵ Step 1)
  ≤ Cap(X^*, X^{*C}).
  (∵ Step 2)

  ⇒ all "≤" are "=".

- 結論:

① 找到 max. flow f^*.
② 找到 min. cut (X^*, X^{*C}).
③ $\max_f$ value(f) = $\min_X$ Cap(X, X^C).

§ 9.2 Network

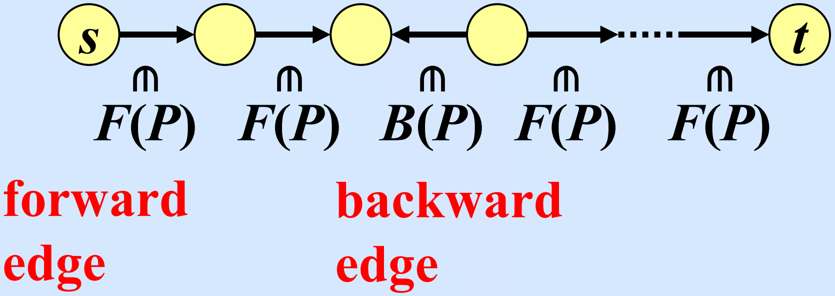
- **<u>Define</u>: Suppose f is an s-t flow in $N = (V, E)$.**

  An *f-augmenting path* is a (not necessary directed) s-t path P s.t.

  (a) if $v_{i-1} \xrightarrow{e_i} v_i$, $e_i \in F(P) \Rightarrow f(e_i) < \text{Cap}(e_i)$.

  (b) if $v_{i-1} \xleftarrow{e_i} v_i$, $e_i \in B(P) \Rightarrow f(e_i) > 0$.

$$P: \underset{\substack{\| \\ s}}{v_0} \overset{e_1}{\text{---}} v_1 \overset{e_2}{\text{---}} v_2 \text{---} \cdots \overset{e_r}{\text{---}} \underset{\substack{\| \\ t}}{v_r}$$



§ 9.2 Network

- **Lemma**: Let $f^*(e) = \begin{cases} f(e) + \varepsilon, & \text{if } e \in F(P) \\ f(e) - \varepsilon, & \text{if } e \in B(P) \\ f(e), & \text{o.w.} \end{cases}$

  where $\varepsilon = \min\{\min_{e \in F(P)} (\text{Cap}(e) - f(e)), \min_{e \in B(P)} f(e)\} > 0$.

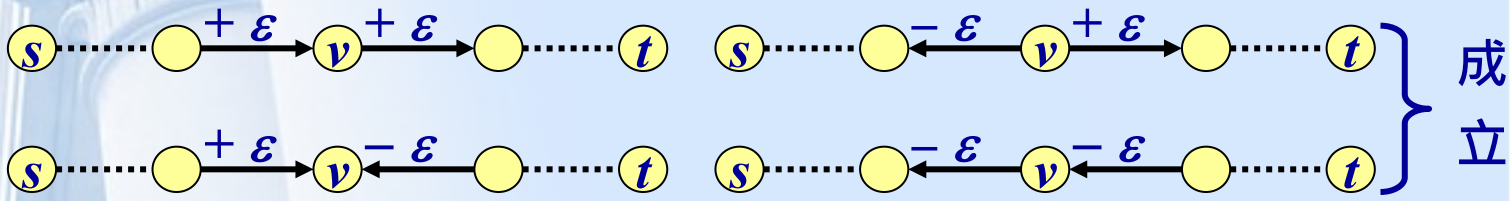
  Then (i) f^* is still a flow of N.

  (ii) value(f^*) = value(f) + ε (> value(f))

  Proof. (1/2)

  (i) $\forall\, e \in E(N)$: $f^*(e) \geq 0$, $f^*(e) \leq \text{Cap}(e)$: easy to check.

  $\forall\, v \in V - \{s, t\}$: $f^*(v, V) = f^*(V, v)$:



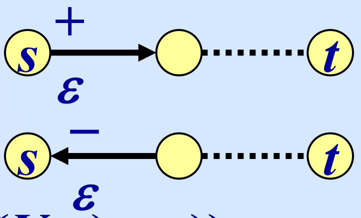
§ 9.2 Network

- **Lemma**: Let $f^*(e) = \begin{cases} f(e) + \varepsilon, & \text{if } e \in F(P) \\ f(e) - \varepsilon, & \text{if } e \in B(P) \\ f(e), & \text{o.w.} \end{cases}$

  where $\varepsilon = \min\{\min_{e \in F(P)} (\mathrm{Cap}(e) - f(e)), \min_{e \in B(P)} f(e)\} > 0$.

  Then (i) f^* is still a flow of N.

  (ii) value(f^*) = value(f) + ε (> value(f))

  Proof. (2/2)

  (ii) value(f^*) = $f^*(s, V) - f^*(V, s)$

  $= f(s, V) + \varepsilon - f(V, s)$ (or $f(s, V) - (f(V, s) - \varepsilon)$)

  = value(f) + ε.

  ∴ ∃ f-augmenting path ⇒ ∃ new flow f^* of larger value.

§ 9.2 Network

- **<u>Algorithm</u>: (Ford and Fulkerson)**

(0) $\forall\, e \in E, f(e) \leftarrow 0$; all vertices are unscanned, no mark;

(1) mark s with "$\varnothing$; ∞"

(2) <u>do while</u> ($\exists$ marked vertex unscanned)

choose a marked vertex x whose mark is "v_x; ε_x";

(3) <u>if</u> $x = t$ <u>then</u>

backtrace from t to get P ; update f to increase valule(f);

erase all mark; <u>go to</u> (1);

(4) $\forall\, xy \in E(N)$ with y unmarked, and $f(xy) < \text{Cap}(xy)$ <u>do</u>

(4.1) mark y with "x; $\min\{\varepsilon_x, \text{Cap}(xy) - f(xy)\}$";

(5) $\forall\, zx \in E(N)$ with z unmarked, and $f(zx) > 0$ <u>do</u>

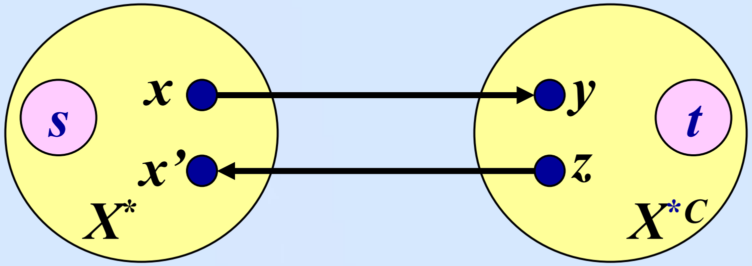
(5.1) mark z with "x; $\min\{\varepsilon_x, f(zx)\}$";

<u>end</u> <u>do</u>;

§ 9.2 Network

- **Claim: When the algorithm STOP,**
  f^* is the final f, X^* is the set of all marked vertices,
  then value(f^*) = Cap(X^*, X^{*C}).

  Proof.



  ∵ when STOP, (3)不會執行

  ∴ $s \in X^*$, $t \notin X^*$.

  $\forall\, x \in X^*$, $y \notin X^*$, $e = xy \in E(N)$:

  ∵ (4.1)不執行 ∴ $f(xy) < \text{Cap}(xy)$ 不成立, i.e. $f(e) = \text{Cap}(e)$.

  $\forall\, x' \in X^*$, $z \notin X^*$, $e = zx \in E(N)$:

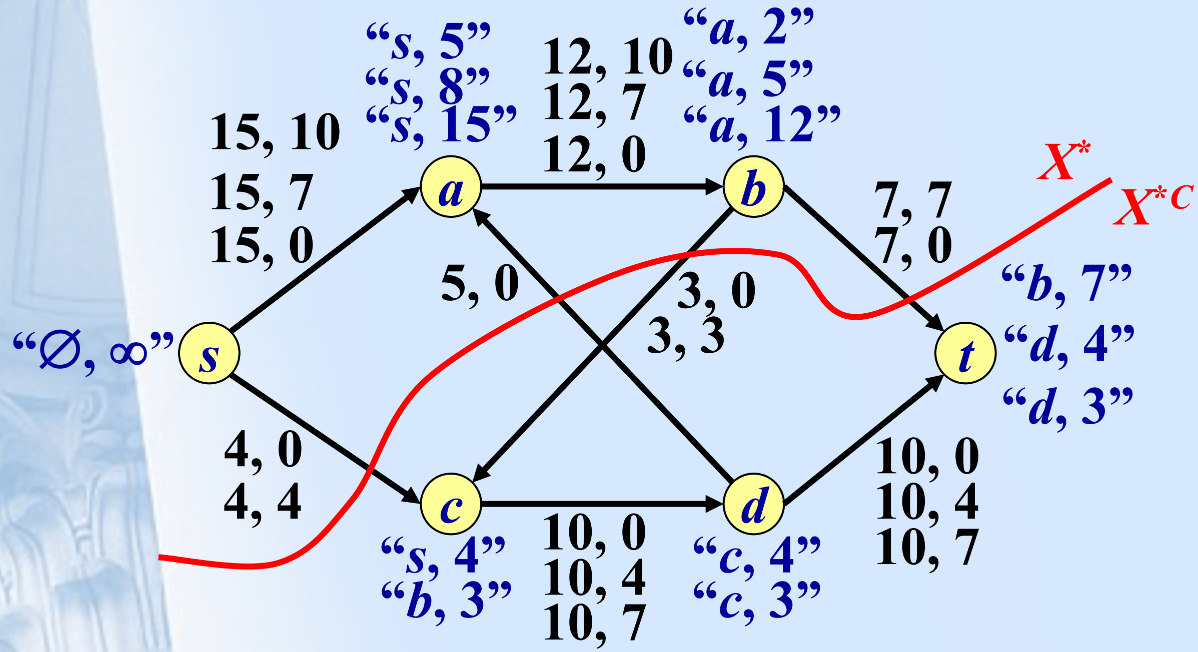
  ∵ (5.1)不執行 ∴ $f(zx) > 0$ 不成立, i.e. $f(zx) = 0$.

  Hence, value(f^*) = $f(X^*, X^{*C}) - f(X^{*C}, X^*)$

  $= \text{Cap}(X^*, X^{*C}) - 0 = \text{Cap}(X^*, X^{*C})$.

§ 9.2 Network

- <u>**Ex**</u>:



§ 9.2 Network

- **<u>Remark</u>:**

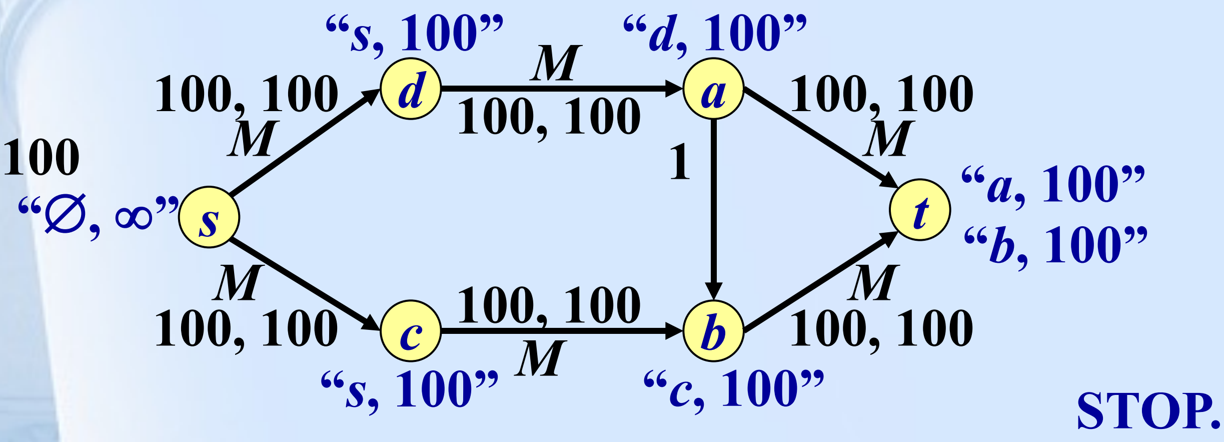
① **If $\forall$ *e* $\in$ *E*(*N*), Cap(*e*) are all in $\mathbb{Q}$, then the algorithm will STOP.**

② **$\exists$ example in which some Cap(*e*) are irrational and algorithm will <u>not</u> stop.**

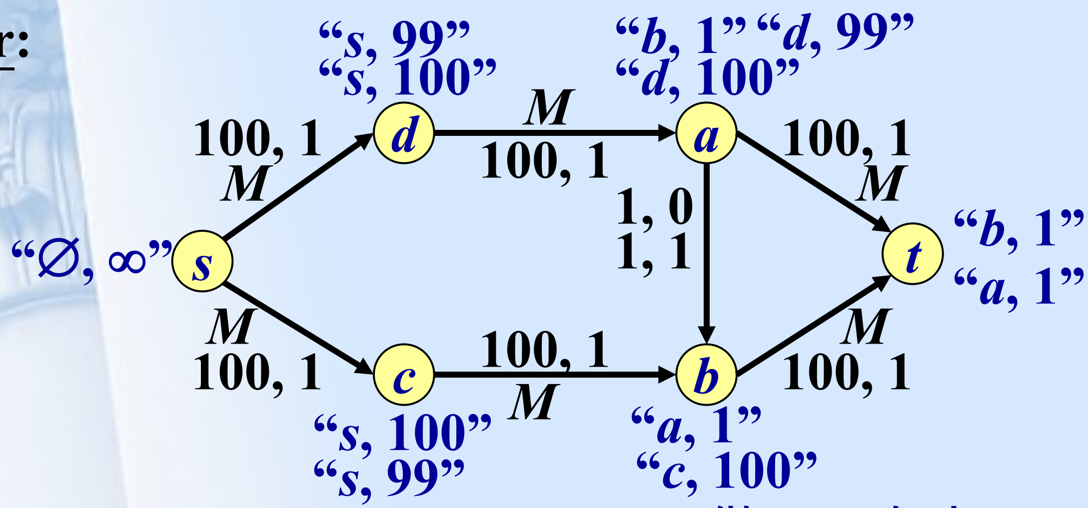
③ **The algorithm may be not efficient even if the capacities are integers.**

§ 9.2 Network

- **Ex:**
  If $M = 100$



Another:



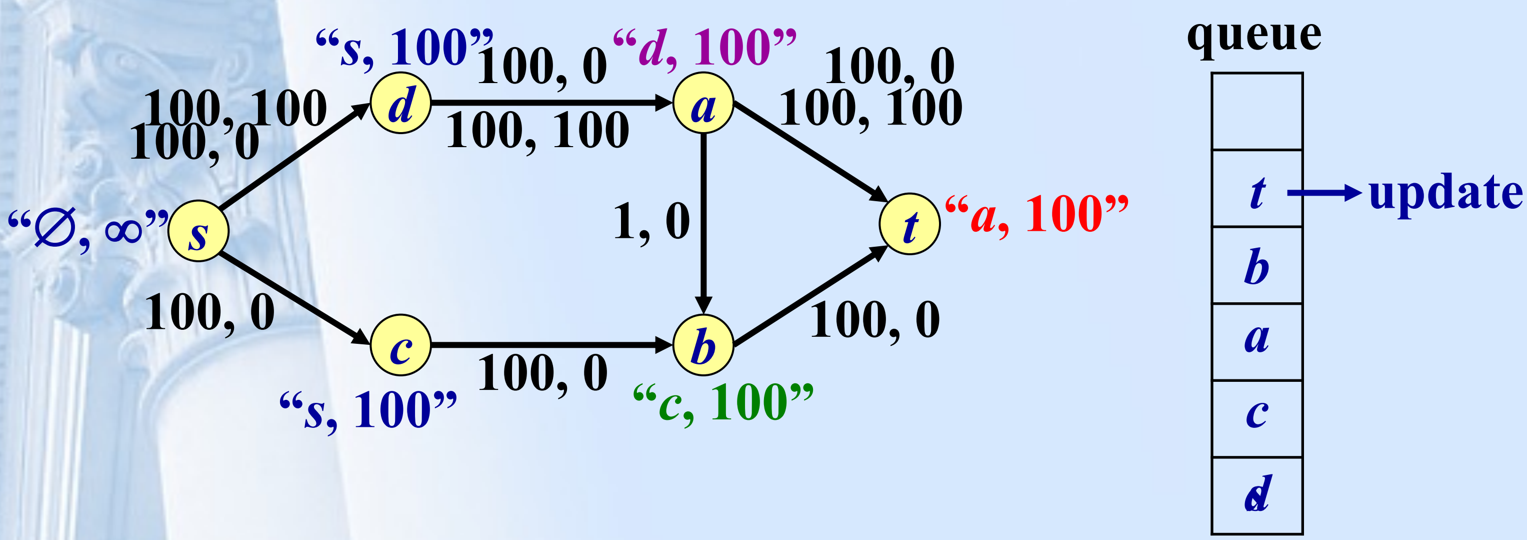
⇒ 做 200 次才 STOP. (2M次)

§ 9.2 Network

- **Remark:**

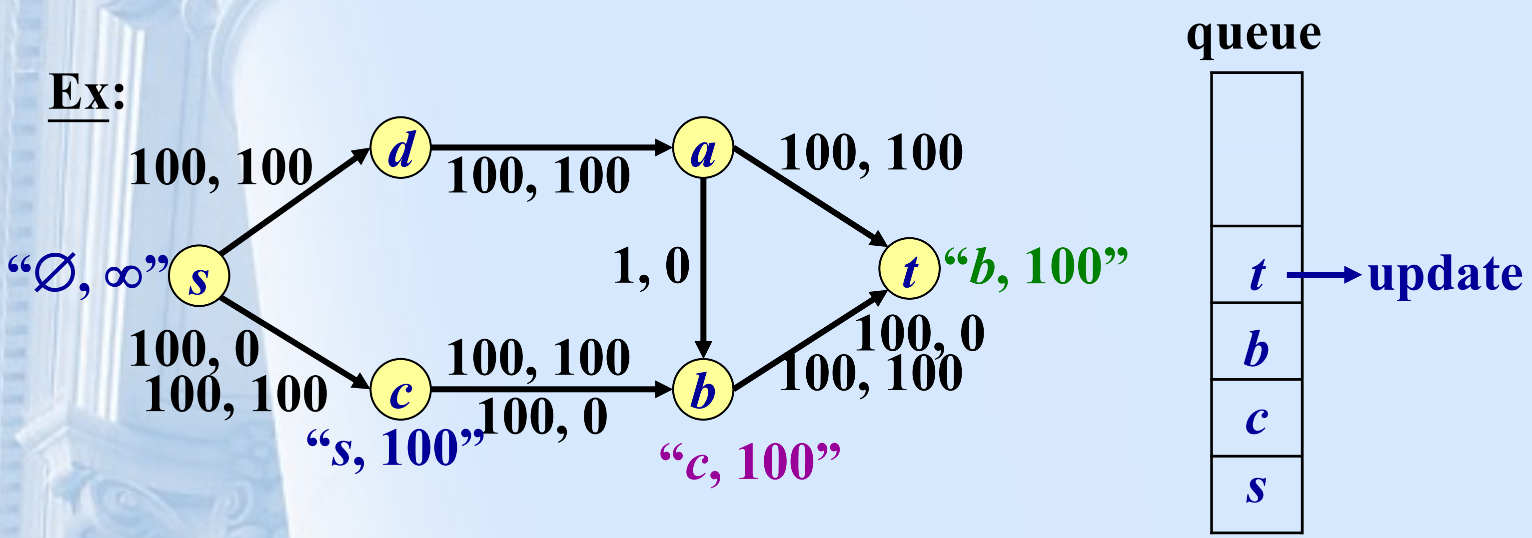
④ **Karp & Edmmonds: Use BFS(FIFO) 選取 *x*,**
則 $O(|V|^3|E|)$ 時間內 Ford & Fulkerson algorithm 會停止.

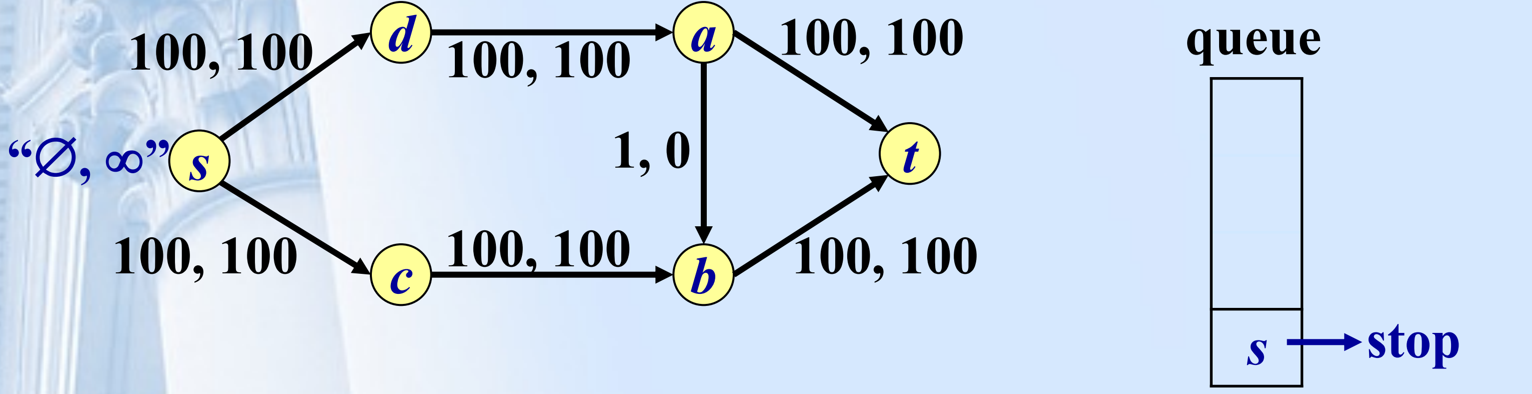
- **Ex:**



§ 9.2 Network

- Ex:





Computer Science and Information Engineering
National Chi Nan University

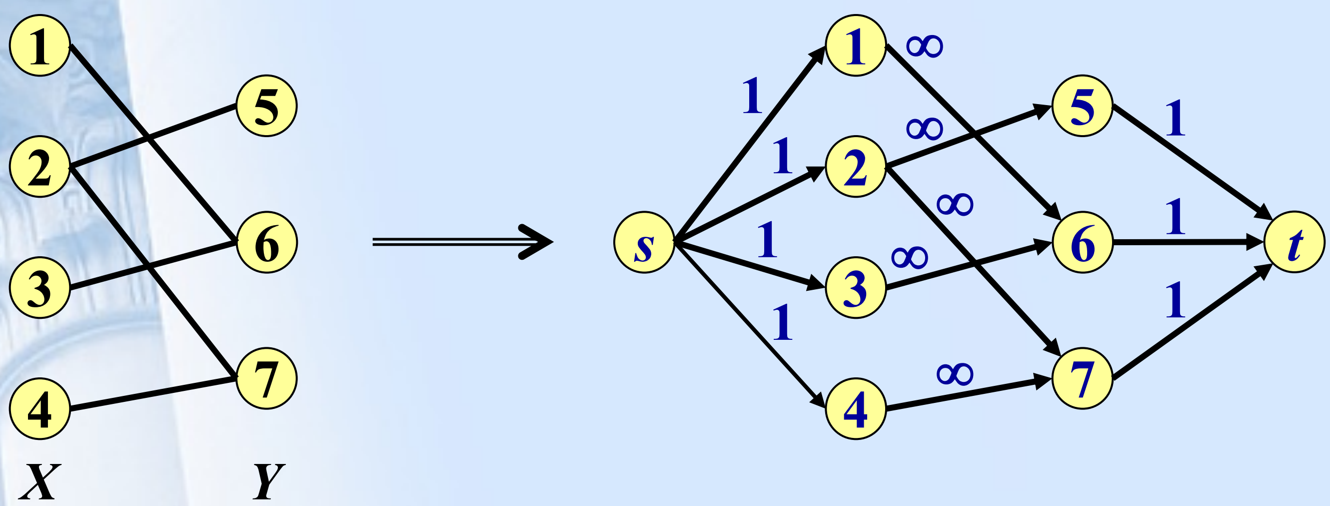
Chapter 9
Connectivity

§ 9.3 Matching and Networks

§ 9.3 Matching and Networks

- **Given an bipartite graph $G = (X, Y, E)$.**
  Let $N = (X \cup Y \cup \{s, t\}, E^*)$ and $\text{Cap}(e) = \begin{cases} \infty, \text{ if } e \in E \\ 1, \text{ o.w.} \end{cases}$
  where $E^* = \{xy: x \in X, y \in Y, xy \in E\} \cup \{sx: x \in X\} \cup \{yt: y \in Y\}$.

- **<u>Ex</u>:**



§ 9.3 Matching and Networks

- <u>Exercise 8 (11/21)</u>: 利用 $\max_{f\text{:flow for } N}$ value(f) = $\min_{S \subseteq N}$ Cap(S, S^C) 証:

  In bipartite graph: $\max_{M\text{:matching for } G} |M| = \min_{C\text{:vertex cover for } G} |C|$.

  $$\left(\begin{array}{l} \max_{M\text{:matching for } G} |M| \\ = \max_{f\text{:flow for } N} \text{value}(f) \text{ (by ①)} \\ = \min_{S \subseteq N} \text{Cap}(S, S^C) \text{ (已知)} \\ = \min_{C\text{:vertex cover for } G} |C| \text{ (by ②)} \end{array}\right)$$

  (You need to prove ①, ②)

  <u>Hint of ①.</u>

  "≤": 令 M^*為 max. matching → 造出一個 flow s.t. value(f^*) = $|M^*|$

  則 $\max_M |M| = |M^*| = \text{value}(f^*) \le \max_f \text{value}(f^*)$.

  "≥": 令 f^*為 max. flow → 造出一個 matching M^* s.t. $|M^*| = \text{value}(f^*)$

  則 $\max_f \text{value}(f) = \text{value}(f^*) = |M^*| \le \max_M |M|$.

Computer Science and Information Engineering
National Chi Nan University

Chapter 9
Connectivity

§ 9.4 *s*-*t* Directed Paths

§ 9.4 *s*-*t* Directed Paths

- **Def**: $N = (V, E)$ is a network with s, t and $\forall\ e$, $\exists$ Cap(e) ≥ 0.
  Let $\mathcal{P} = \{P: P$ is an s-t directed path$\}$.
  Assign every P in $\mathcal{P}$ a non-negative real number a_P
  s.t. $\forall\ e \in E$, Cap(e) $\geq \sum_{\forall P \in \mathcal{P} \text{ s.t. } e \in P} a_P$.

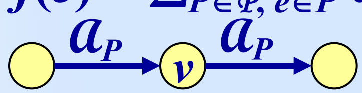
§ 9.4 *s*-*t* Directed Paths

- **Lemma**: $\max_{a} \sum_{P\in\mathcal{P}} a_P \le \max_{f\text{:flow for } N} \text{value}(f)$.

  Proof.

  For $\{a_P\}_{P\in\mathcal{P}}$ s.t. $\sum_{P\in\mathcal{P}} a_P$ is maximum.

  Define flow f by $f(e) = \sum_{P\in\mathcal{P},\, e\in P} a_P$.

  ① f is a flow: (i) 

  (ii) $f(e) \le \text{Cap}(e)$

  ② $\sum_{P\in\mathcal{P}} a_P = \text{value}(f)$: $\sum_{P\in\mathcal{P}} a_P = \sum_{P\in\mathcal{P} \text{ s.t. } sx\in P} a_P - \sum_{P\in\mathcal{P} \text{ s.t. } ys\in P} a_P$

  $= \sum_{sx\in E} f(sx) - \sum_{ys\in E} f(ys)$

  $= \text{value}(f)$.

  $\therefore \max \sum_{P\in\mathcal{P}} a_P = \text{value}(f) \le \max_{f\text{:flow for } N} \text{value}(f)$.

§ 9.4 *s*-*t* Directed Paths

- **Theorem**: $\max_{a} \sum_{P \in \mathcal{P}} a_P = \max_{f\text{:flow for } N} \text{value}(f)$.

  Proof. (1/3)

  When apply Ford and Fulkerson Algorithm;

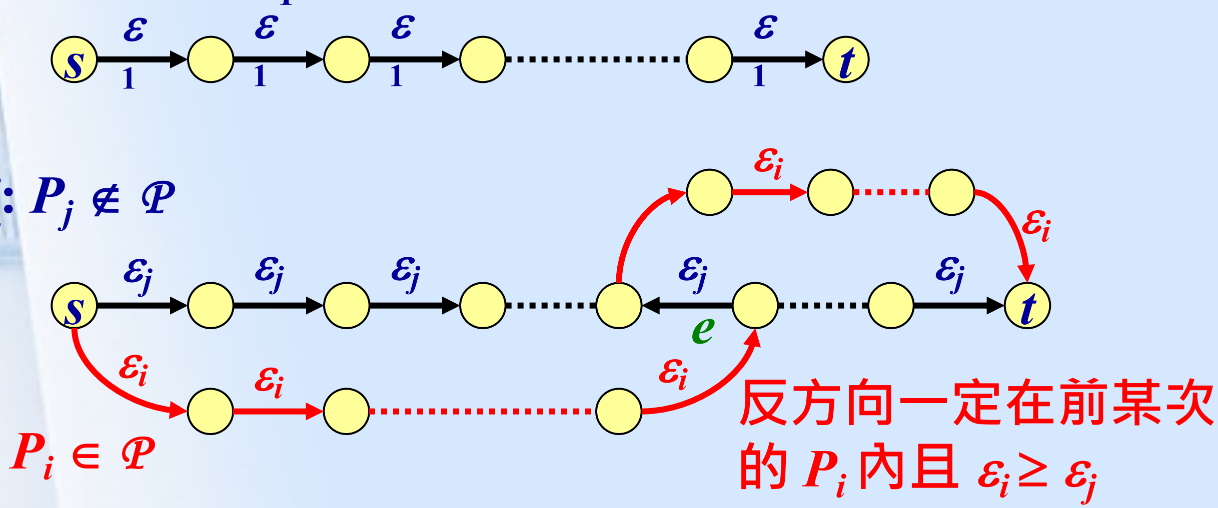
  Define $\{a_P\}_{P \in \mathcal{P}}$ as:

  Initial: $a_P = 0, \forall P \in \mathcal{P}$.

  iteration 1: $P_1 \in \mathcal{P}$: $a_{P_1} \leftarrow \varepsilon_1$;

  ⋮

  iteration *j*: $P_j \notin \mathcal{P}$



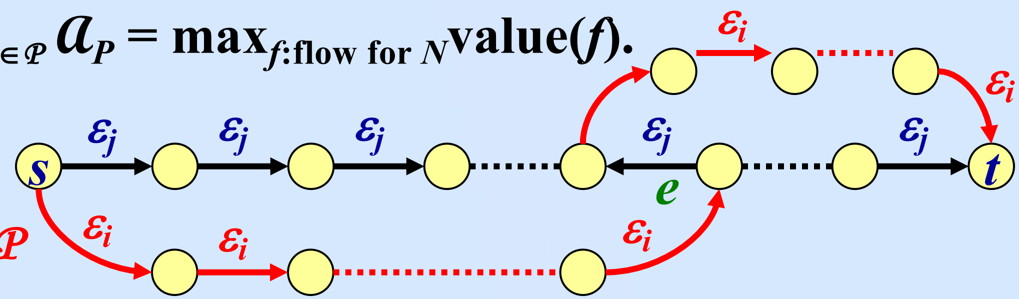
§ 9.4 *s*-*t* Directed Paths

- **Theorem**: $\max_{\alpha} \sum_{P\in\mathcal{P}} \alpha_P = \max_{f\text{:flow for } N} \text{value}(f)$.

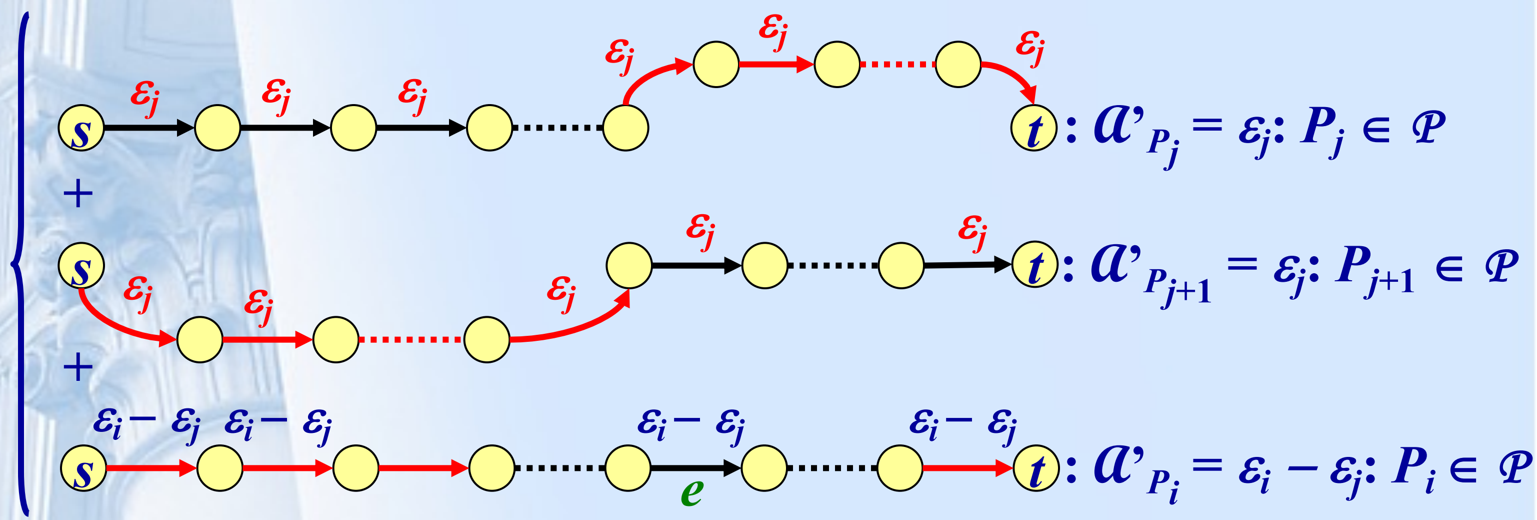
  Proof. (2/3)

  iteration j: $P_j \notin \mathcal{P}$

  $P_i \in \mathcal{P}$



$\Rightarrow$



$\forall$edge $\in P_i \cup P_j$: $f(e) = \sum_{P\in\mathcal{P},\, e\in P} \alpha_P \Rightarrow \forall e \in E$: $f(e) = \sum_{P\in\mathcal{P},\, e\in P} \alpha_P$

§ 9.4 *s*-*t* Directed Paths

- **Theorem**: $\max_{\alpha} \sum_{P\in\mathcal{P}} \alpha_P = \max_{f\text{:flow for } N} \text{value}(f)$.

  Proof. (3/3)

  Initial: $\alpha_P = 0, \forall P \in \mathcal{P}$.

  iteration 1: $P_1 \in \mathcal{P}$: $\alpha_{P_1} \leftarrow \varepsilon_1$;

  ⋮

  iteration *j*: $P_j \notin \mathcal{P}$

  $\forall \text{edge} \in P_i \cup P_j$: $f(e) = \sum_{P\in\mathcal{P},\, e\in P} \alpha_P$

  $\Rightarrow \forall e \in E$: $f(e) = \sum_{P\in\mathcal{P},\, e\in P} \alpha_P$

  $\therefore$ When STOP, we find a $\{\alpha_P^*\}_{P\in\mathcal{P}}$ s.t. $f(e) = \sum_{P\in\mathcal{P},\, e\in P} \alpha_P^*$

  $\max_{\alpha}\sum_{P\in\mathcal{P}} \alpha_P \geq \sum \alpha_P^* = \text{value}(f)$

  $= \max_{f\text{:flow for } N}\text{value}(f) \geq \max_{\alpha} \sum_{P\in\mathcal{P}} \alpha_P$

  $\therefore$ all "≥" are "=".

  $\Rightarrow \max_{\alpha} \sum_{P\in\mathcal{P}} \alpha_P = \max_{f\text{:flow for } N}\text{value}(f)$.

§ 9.4 *s*-*t* Directed Paths

- **<u>Remark</u>: ∀ flow *f*, ∃ $\{a_P\}_{P\in\mathcal{P}}$ s.t. $\sum_{P\in\mathcal{P}} a_P$ = value(*f*).**