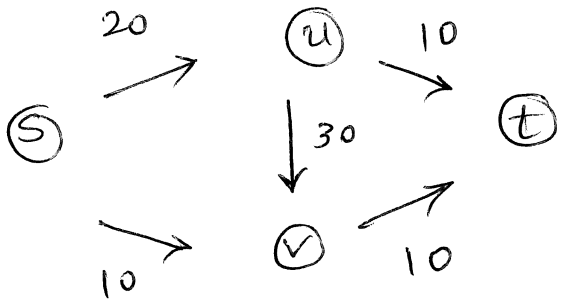
# Maximum Flows

**Def** A flow network consists of

- a directed graph $G(V, E)$
- source node $s \in V$, sink node $t \in V$.
- "Capacity" $c_e > 0$ for every $e \in E$.
- one can assume that $s$ has only outgoing edges and $t$ has only incoming edges.

Example




**Def** A flow on a network $(G(V, E), s, t, \{c_e | e \in E\})$ is a map $f: E \to \mathbb{R}^+$ s.t. ← non-negative reals

capacity constraint

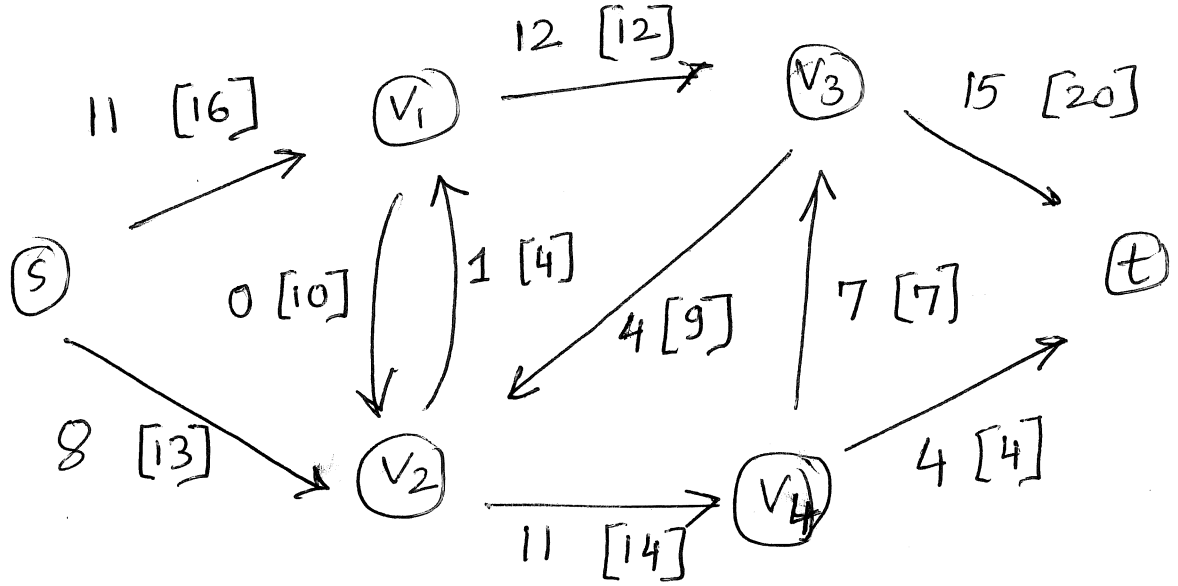
$$\forall e \in E, \quad f(e) \leq c_e$$

conservation constraint $\forall v \in V \setminus \{s, t\}$,

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$$

# Example of a flow

Capacities appear in [brackets].



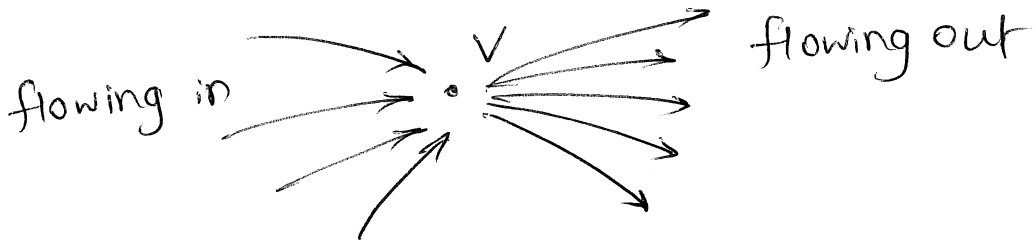

Value of flow = 11 + 8 flow out of s

= 15 + 4 flow into t

= 19

**Def** $\text{Value}(f) = \sum_{e \text{ out of } s} f(e)$.

f is thought of as a "flow" of a fluid from s to t s.t. no edge capacity is exceeded and for all vertices other than s and t, amount of fluid flowing in equals amount of fluid flowing out.




**Problem.** Given a network $(G(V,E), s, t, \{c_e | e \in E\})$, find a flow with maximum value.

**Fact** $\text{value}(f) = \sum_{e \text{ into } t} f(e)$

**Proof** Start by writing

$$\sum_{e \in E} f(e) = \sum_{e \in E} f(e)$$

Every edge leaves one vertex and enters one vertex. So

$$\sum_{v \in V} \sum_{e \text{ into } v} f(e) = \sum_{v \in V} \sum_{e \text{ out of } v} f(e)$$

$$\therefore \sum_{e \text{ into } t} f(e) + \left( \sum_{v \in V \setminus \{s,t\}} \sum_{e \text{ into } v} f(e) \right) =$$

$$\sum_{e \text{ out of } s} f(e) + \left( \sum_{v \in V \setminus \{s,t\}} \sum_{e \text{ out of } v} f(e) \right)$$

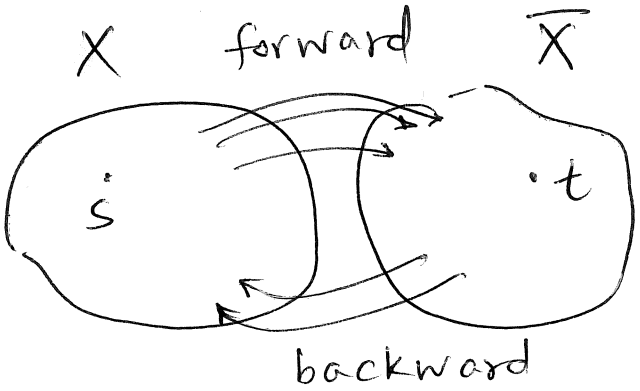
Now these sums are equal for every $v \in V \setminus \{s,t\}$.

Hence $$\sum_{e \text{ into } t} f(e) = \sum_{e \text{ out of } s} f(e) = \text{value}(f).$$

**Def** A cut in graph $(G(V,E), s, t)$ is a partition $V = X \cup \overline{X}$ such that $s \in X$, $t \in \overline{X}$.

**Def** Edges from $X$ to $\overline{X}$ are called forward edges.

" $\overline{X}$ to $X$ " backward ".




**Def** Capacity of a cut $(X, \bar{X})$ is the sum of capacities of all forward edges.

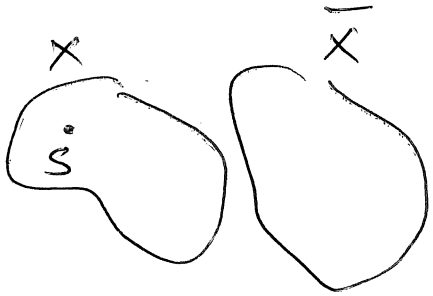
$$cap(X, \bar{X}) = \sum_{\substack{e = (a,b) \\ a \in X,\ b \in \bar{X}}} c_e .$$

**Fact** If $(X, \bar{X})$ is a cut and $f$ is a flow then

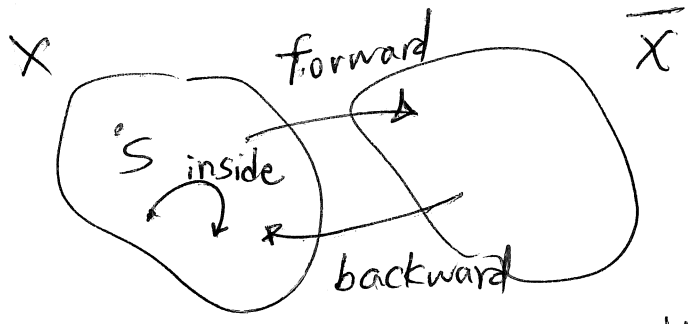
$$value(f) = \sum_{e \text{ is forward edge}} f(e) - \sum_{e \text{ is backward edge}} f(e) .$$

**Proof**

$$value(f) = \sum_{e \text{ out of } s} f(e)$$




$$= \sum_{e \text{ out of } s} f(e) + \sum_{v \in X \setminus \{s\}} \underbrace{\left( \sum_{e \text{ out of } v} f(e) - \sum_{e \text{ into } v} f(e) \right)}_{\text{this is just zero.}}$$




It can be checked that in the above sum

- forward edges appear with coefficient +1.
- backward " −1.
- edges inside $S$ appear once with coefficient +1 and once with coefficient −1 and so cancel each other.

Hence,

$$\text{value}(f) = \sum_{\substack{e=(a,b) \\ a \in X,\ b \in \overline{X}}} f(e) - \sum_{\substack{e=(c,d) \\ c \in \overline{X},\ d \in X}} f(e)$$

$$= \sum_{e \text{ forward}} f(e) - \sum_{e \text{ backward}} f(e)$$

Intuitively, amount of flow from $s$ to $t$ equals amount of flow from $X$ to $\overline{X}$, but one has to subtract the amount of flow

from $\bar{X}$ flowing backward into $X$.

**Lemma.** If $f$ is any flow and $(X, \bar{X})$ is any cut,

$$\text{value}(f) \leq \text{cap}(X, \bar{X}).$$

hence MAX-flow $\leq$ Min-cut-capacity.

**Proof.**

$$\text{value}(f) = \sum_{e \text{ is forward}} f(e) - \sum_{e \text{ is backward}} f(e)$$

$$\leq \sum_{e \text{ is forward}} f(e)$$

$$\leq \sum_{e \text{ is forward}} c_e$$

$$= \text{cap}(X, \bar{X}).$$

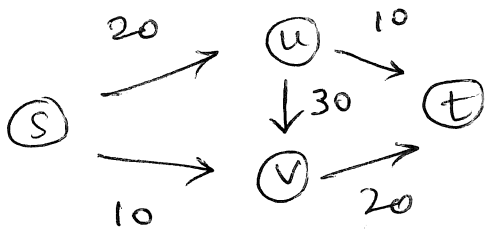
∎

Now we introduce notion of augmenting paths that allow us to improve (i.e. increase) the current flow

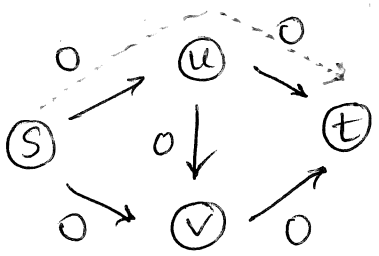
## Augmenting Paths

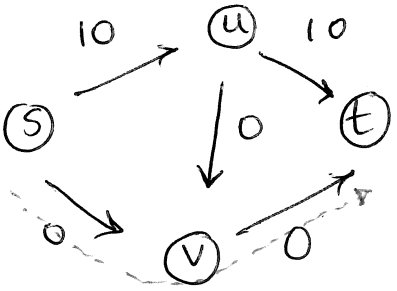
Sending additional flow along a path.

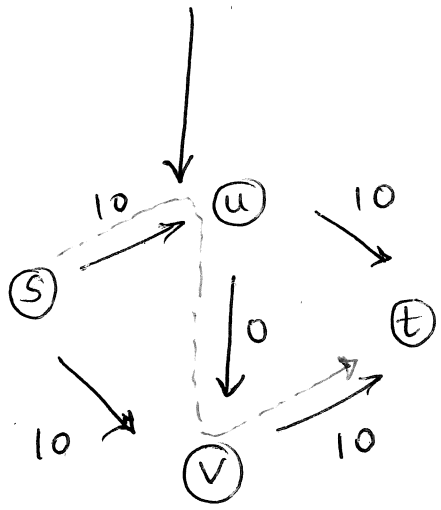
Consider the network with capacities

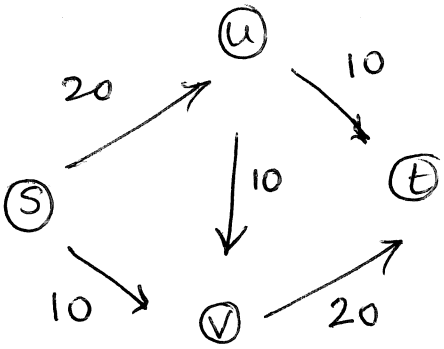



Here is a sequence of augmenting paths that improve the flow starting with zero flow.



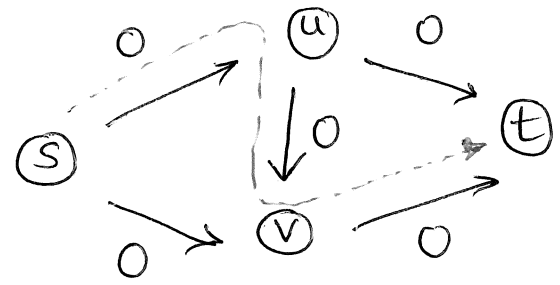



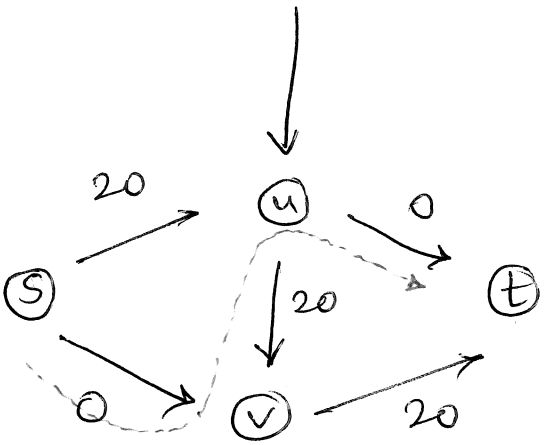



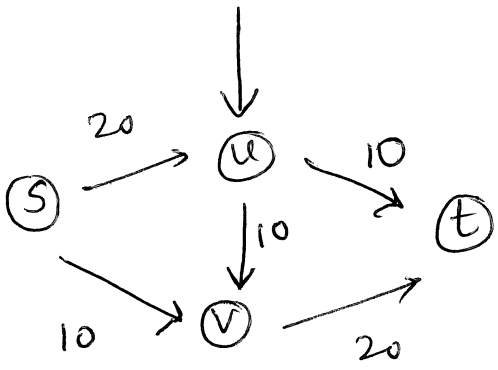




This is the maximum flow.

Here is a different sequence of augmenting paths that uses "push back" of flow.




10 units of flow is pushed back on edge (u, v), reducing flow on that edge to 10.

Residual graphs are convenient tool to find augmenting paths.

# Residual Graph

**Def** Given a network $(G(V,E), s, t, \{c_e \mid e \in E\})$ and flow $f$ on it, the "residual graph" $R_f$ and "residual capacities" $res(\cdot)$ are defined as follows:

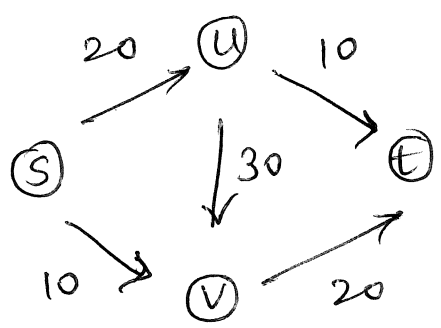
- Vertices of $R_f$ are same as $V$. There are two types of edges, original edges and new edges.
- **Original edges** For any edge $e \in E$, if $f(e) < c_e$ (i.e. the edge is unsaturated) then add $e$ to $R_f$ with residual capacity

$$res(e) = c_e - f(e).$$

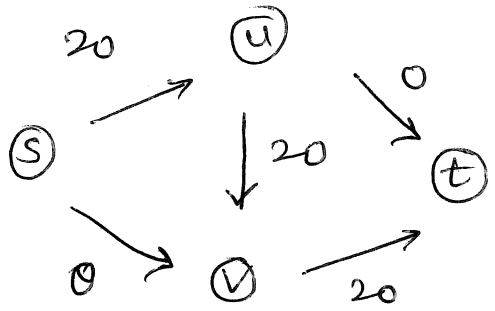
- **New edges** For any edge $e \in E$, if $f(e) > 0$ (i.e. the edge has non-zero flow) then add $e^{reverse}$ (i.e. the reverse of edge $e$) to $R_f$ with residual capacity

$$res(e^{reverse}) = f(e).$$

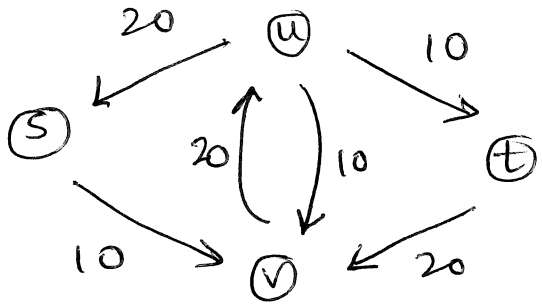
Example




capacities




flow $f$




Residual graph $R_f$

**Lemma** Let  be $s \rightsquigarrow t$ path in the residual graph $R_f$. Let $\alpha = \min_{1 \le i \le k} res(e_i)$. Let $f'$ be new flow obtained by modifying flow $f$ as follows:

- If $e_i$ is original edge, increment flow on it by $\alpha$.

- If $e_i$ is a new edge, then reduce flow on $e_i^{reverse}$ by $\alpha$. (note that $e_i^{reverse}$ will be present in the network).

Then $f'$ is a valid flow and

$$value(f') = value(f) + \alpha.$$

Proof: Exercise. ▨

**Theorem** Let $(G(V,E), s, t, \{c_e \mid e \in E\})$ be a network with flow $f$. Let $R_f$ be the residual graph. Then following are equivalent.

1. $f$ is a maximum flow.
2. $R_f$ has no $s \rightsquigarrow t$ path.
3. There exists a cut $(X, \bar{X})$ s.t.

$$value(f) = cap(X, \bar{X}).$$

Proof

① ⇒ ②: Obvious since if there were a $s \rightsquigarrow t$ path in $R_f$, it could be used to improve flow $f$, which is not possible since $f$ is already a maximum flow.

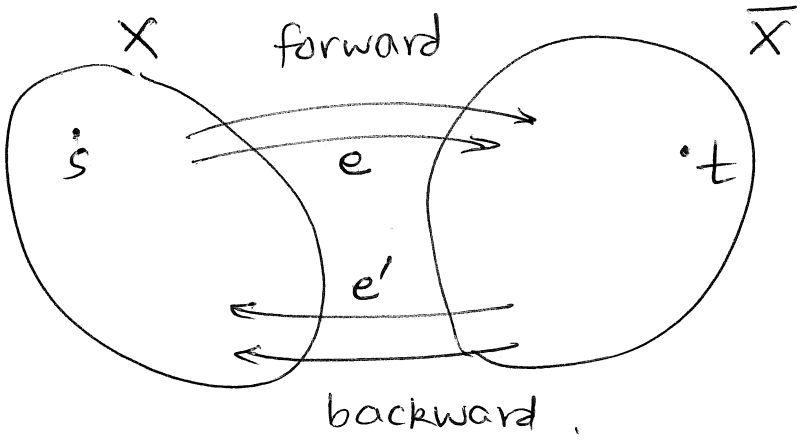
③ ⇒ ①: Obvious since $\text{value}(f) \leq \text{cap}(X, \bar{X})$ for <u>any</u> flow $f$ and <u>any</u> cut $(X, \bar{X})$. Hence if an equality is achieved for <u>some</u> flow $f$ <u>and</u> some cut $(X, \bar{X})$, that flow must be maximum flow and that cut must be minimum cut.

② ⇒ ③: Suppose there is no path from $s$ to $t$ in $R_f$. Let

$X$ = set of all vertices reachable from $s$ in $R_f$.

Thus $t \notin X$, $t \in \bar{X}$.

$G(V, E)$.




Note. There is no edge from $x$ to $\bar{x}$ in $R_f$.

Claim. Every forward edge $e$ is saturated.

Proof. Otherwise $e$ will appear in $R_f$.

Claim. Every backward edge $e'$ has zero flow.

Proof. Otherwise $e'^{\text{reverse}}$ will appear in $R_f$.

Thus

$$\text{value}(f) = \sum_{e \text{ forward}} f(e) - \sum_{e' \text{ backward}} f(e')$$

$$= \sum_{e \text{ forward}} c_e - 0$$

$$= \text{cap}(X, \bar{X}).$$

▨

It follows immediately that

**Theorem:** MAXIMUM FLOW = MINIMUM CUT-CAPACITY.

## Ford-Fulkerson Algorithm

- Assume that all capacities are integers.
  Let $C = \sum_{e \in E} c_e$ be sum of all edge capacities. Clearly max-flow $\leq C$.

- Algorithm
  - start with zero flow.
  - Repeated find augmenting path in the residual graph and use it to improve the flow. Stop if no s→t path in residual graph.

- Running time
  We observe that all flow values and residual capacities remain integral.
  Hence if a flow improves, it improves by at least one unit.

$\therefore$ # iterations $\leq$ max-flow $\leq C$.

Each iteration takes $O(m+n)$ time (to construct the residual graph and to find $s \rightsquigarrow t$ path in it if one exists)

$\therefore$ Running time $= O((m+n) \cdot C)$.

Note: • If the edge-capacities are exponential sized integers, running time could be exponential.

• If the edge-capacities are allowed to be irrational numbers, the algorithm may never terminate, and may converge to a sub-optimal value.

Both these drawbacks are avoided, giving a polynomial time algorithm by

[Dinitz] [Edmond-Karp].

- Always augment along the shortest path (i.e. one with min # edges) in $R_f$.

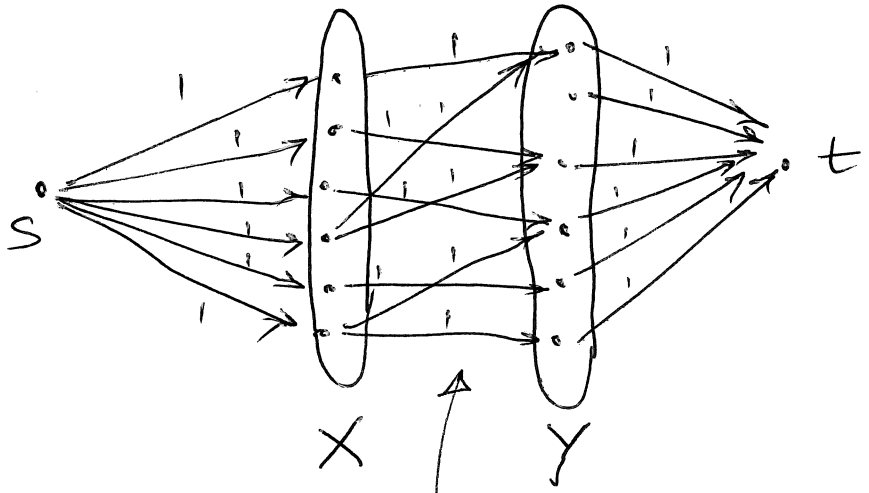
(Given as exercise with clues).

# Application to perfect matchings in bipartite graphs

**Def** Let $G(X, Y, E)$ be a bipartite graph with $|X| = |Y| = n$. A perfect matching is a set of $n$ edges $(x_1, y_1), \dots, (x_n, y_n) \in E$ s.t. $x_1, x_2, \dots, x_n$ are distinct and $y_1, y_2, \dots, y_n$ are distinct.

**Problem** Given $G(X, Y, E)$, decide if it has a perfect matching.

**Algorithm** construct the following network.

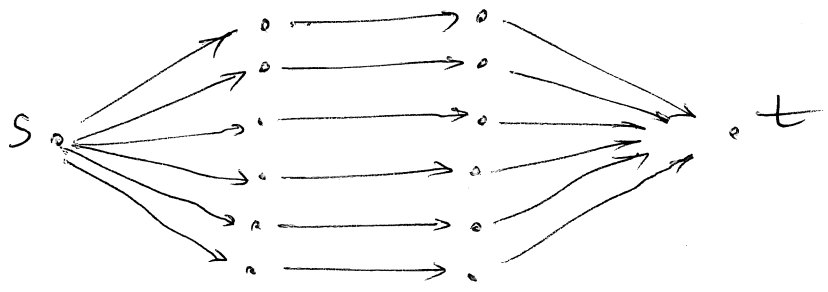



- Given bipartite graph.
- All edge capacities are 1.

- Find maximum flow in the graph/network integral
  (note: since all edge capacities are integers, maximum-flow is guaranteed to be integral).
- There exists perfect matching iff max-flow has value $n$.

Proof: Exercise.

Hint:
- Each edge has either one unit flow or has zero flow.
- This picture will clarify the idea.




Any max-flow must look like this?