Linear Program

$$\overbrace{x_1, \dots \quad x_n}^{n \text{ variables}} \in \mathbb{R}^n$$

Variables: $x_1, \dots x_n \in \mathbb{R}^n$

INPUT: $\{a_{ij}, b_j\}$

Constraints:

m constraints

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \leq b_1 \\ a_{21}x_1 + a_{22}x_2 \quad + a_{2n}x_n \leq b_2 \\ \vdots \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 \quad \dots \quad + a_{mn}x_n \leq b_m \end{cases}$$

Maximise $c_1x_1 + c_2x_2 + \dots c_nx_n$

CHANGING FORMS OF LP

$\leq$ Constraints to $\geq$ Constraints

$$\sum a_i x_i \geq b_i \iff -\sum_i a_i x_i \leq -b_i$$

$=$ constraints to $\leq$ Constraints

$$\sum a_i x_i = b_i \iff \left\{ \begin{array}{l} \sum a_i x_i \leq b_i \\ -\left(\sum a_i x_i\right) \leq -b_i \end{array} \right\}$$

Maximization to Minimization

$$\text{Max} \sum a_i x_i \iff \text{Min} \; -\sum a_j x_j$$

WRITING LINEAR PROGRAMS WITH MATRICES

Variables: $x_1, \ldots x_n \in \mathbb{R}^n$

Constraints:

$$\underbrace{\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 \quad + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 \cdots + a_{mn}x_n \end{cases}}_{A \cdot x} \quad \begin{matrix} \leq b_1 \\ \leq b_2 \\ \vdots \\ \leq b_m \end{matrix}$$

$$A \cdot x \leq b$$

Maximise $c_1x_1 + c_2x_2 + \cdots c_nx_n$

$$\| \\ c^T x$$

Standard form:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \qquad \text{for } i = 1 .. m$$

$$x_i \geq 0 \qquad i = 1 \ldots n$$

$$\text{Min} \quad \sum_{i=1}^{n} c_i x_i$$

Maximum Flow

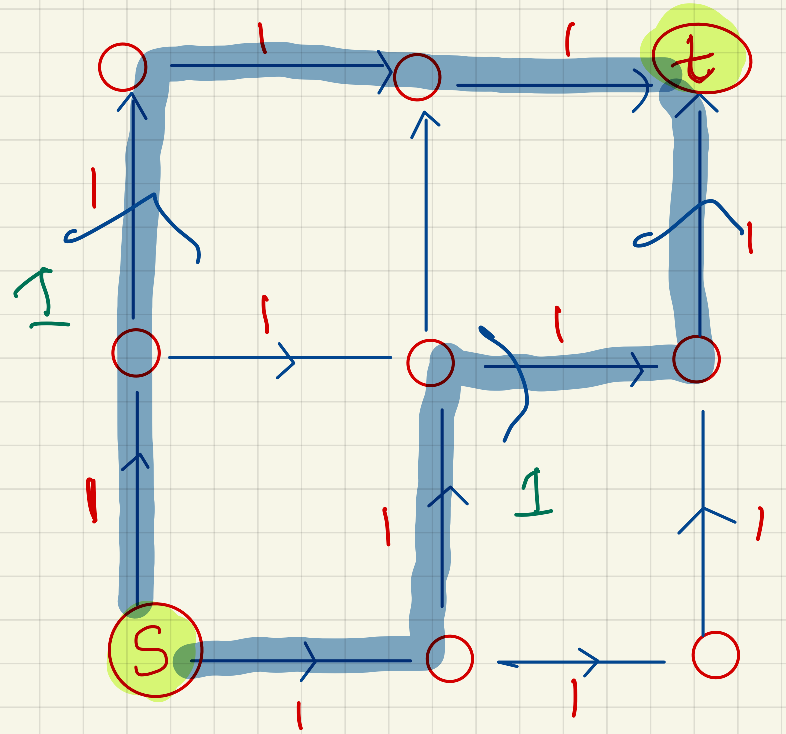
Setup: 1) Directed graph
$G = (V, E)$

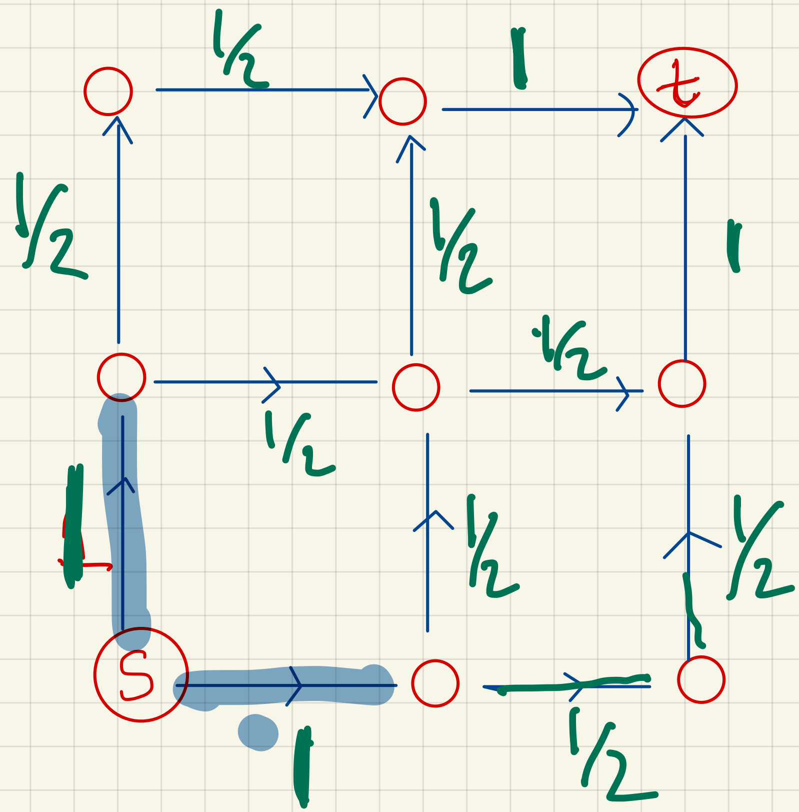
2) "Source" s node

3) "Sink" t node

4) Capacities $c_e \in \mathbb{R}^+$ / $\mathbb{Z}^+$ for each edge e

GOAL: THINK of G as a network of "pipes", what is maximum amount of water one can flow from $s \rightsquigarrow t$?





Definition: (Flow)

A flow is an assignment f_e for each directed edge e.

Capacity: $\forall e,\ f_e \leq c_e$ = capacity of edge e

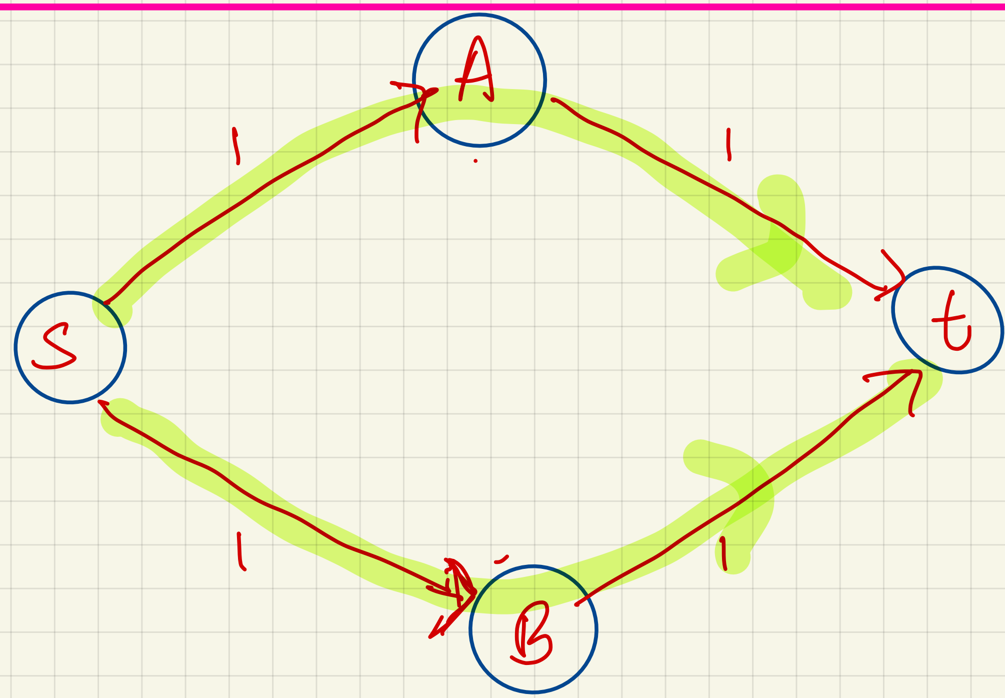
Conservation: $\forall$ node $v \notin$ Source s / Sink t

Total Flow entering = Total flow leaving

$$\sum_{u \to v} f_{uv} = \sum_{v \to w} f_{vw}$$

Maximize : $\sum_{s \to u} f_{su}$

GENERAL IDEA OF AN ALGORITHM TO COMPUTE MAX FLOW



EXAMPLE:

1) SEND 1 unit on S→A→t

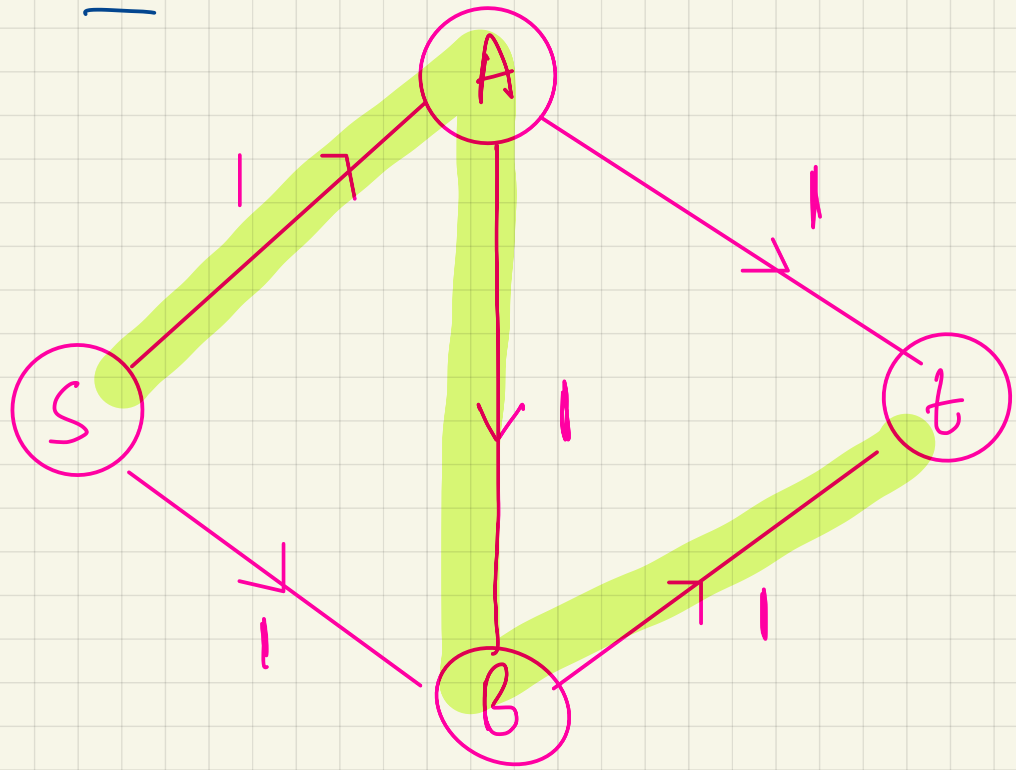
2) SEND 1 unit on S→B→t

GENERAL IDEA

REPEAT: 1) Find a path P from s to t with left-over capacity to send more flow.

2) Add flow along P.

FAILURE OF ALGO



MAX FLOW:

1 unit S→A→t

1 unit S→B→t

Iteration 1:

- Send 1 unit along SABt

Iteration 2:

NO PATH from S⇝t with left over capacity in G!! ALGO TERMINATES!!

ALGO FOR MAX FLOW

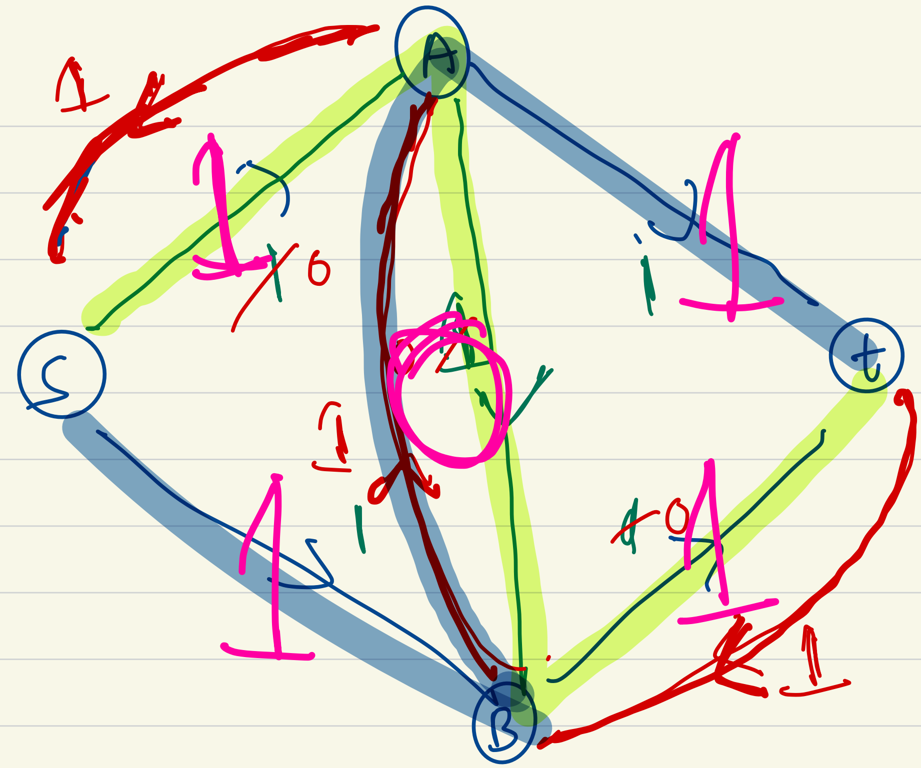
REPEAT:

* FIND A PATH P from s to t with non-zero capacity in RESIDUAL GRAPH

[TERMINATE if NO PATH P exists]

* Add flow along P to the current flow

Residual graph



1 → (1 unit : S → A → B → t)

1-unit S → B → A → t

2 units

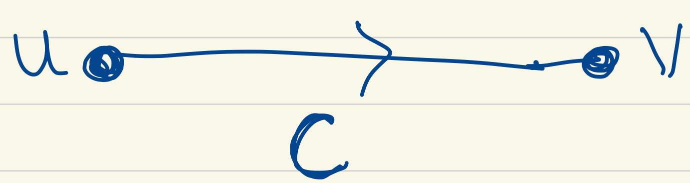
Residual Graph:

<u>Given</u>: * $G = (V, E)$ is a directed graph

* f is some flow on G.

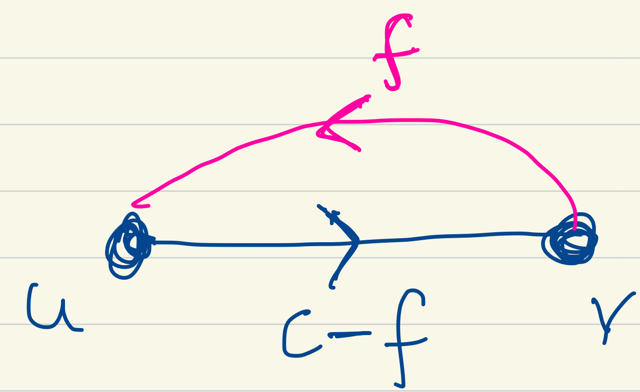
THE RESIDUAL GRAPH G_f on same vertices V and edges

$\forall$ edge u, v with capacity c
flow f



in ORIGINAL GRAPH G

$\Longrightarrow$



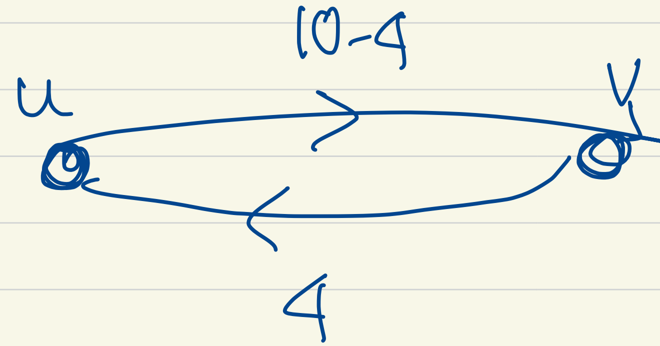
in RESIDUAL GRAPH

EXAMPLE:

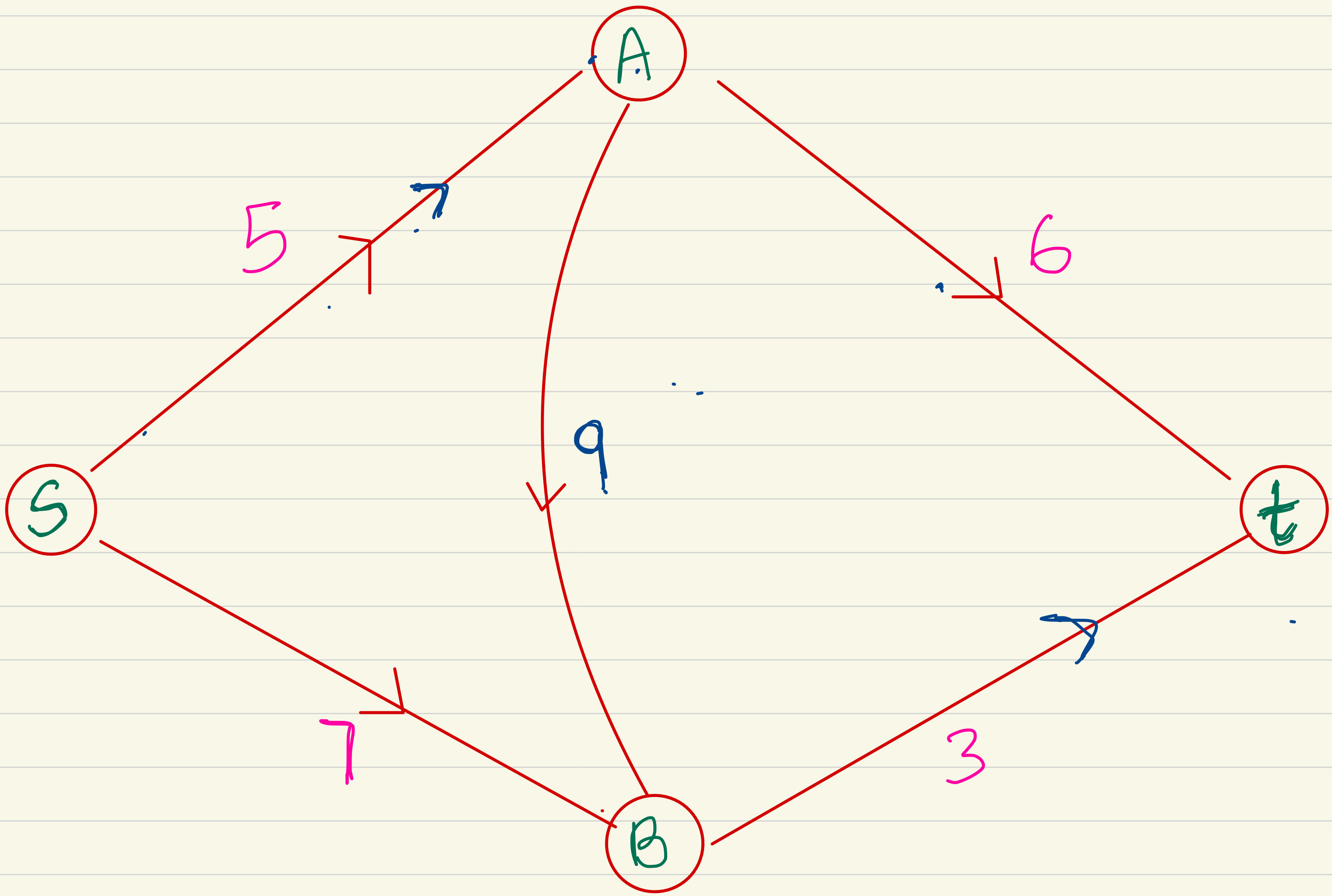
ORIGINAL GRAPH

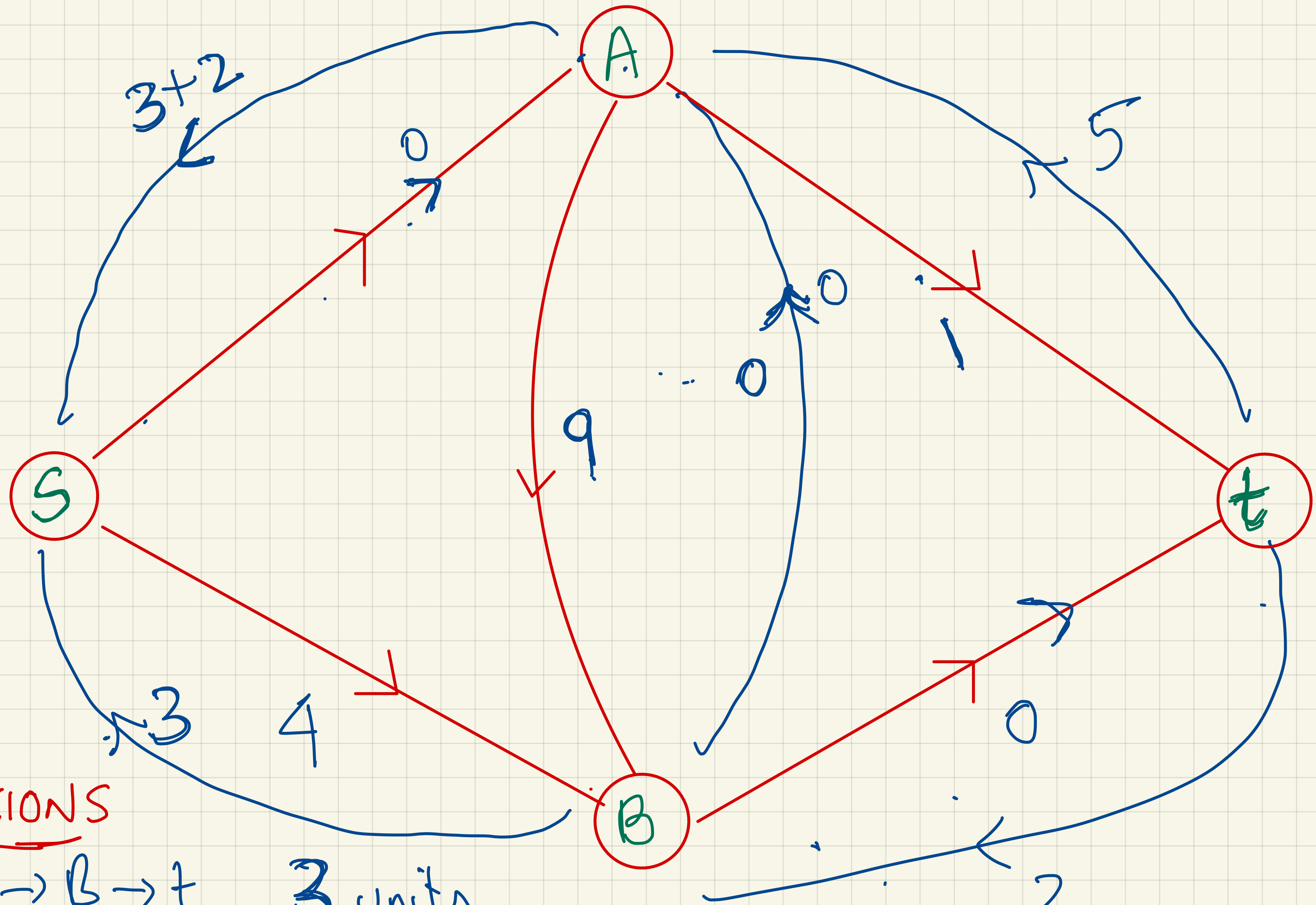
Capacity 10
Flow 4

RESIDUAL GRAPH



EXECUTE MAX FLOW ALGO ON





ITERATIONS

$S \to A \to B \to t$ 3 units

$S \to A \to t$ 2 units

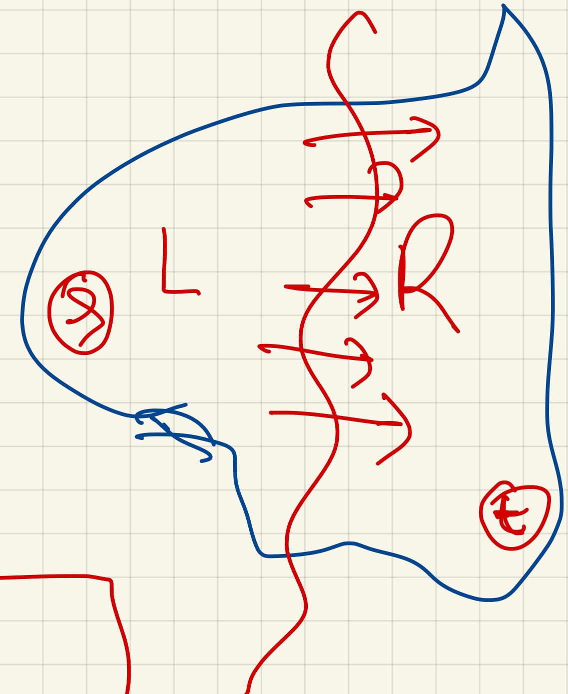
$S \to B \to A \to t$ 3 units

<u>s-t Cut:</u> An s-t Cut is (L, R)

$$L \cup R = V$$

L → left, R → right

$s \in L \quad t \in R$

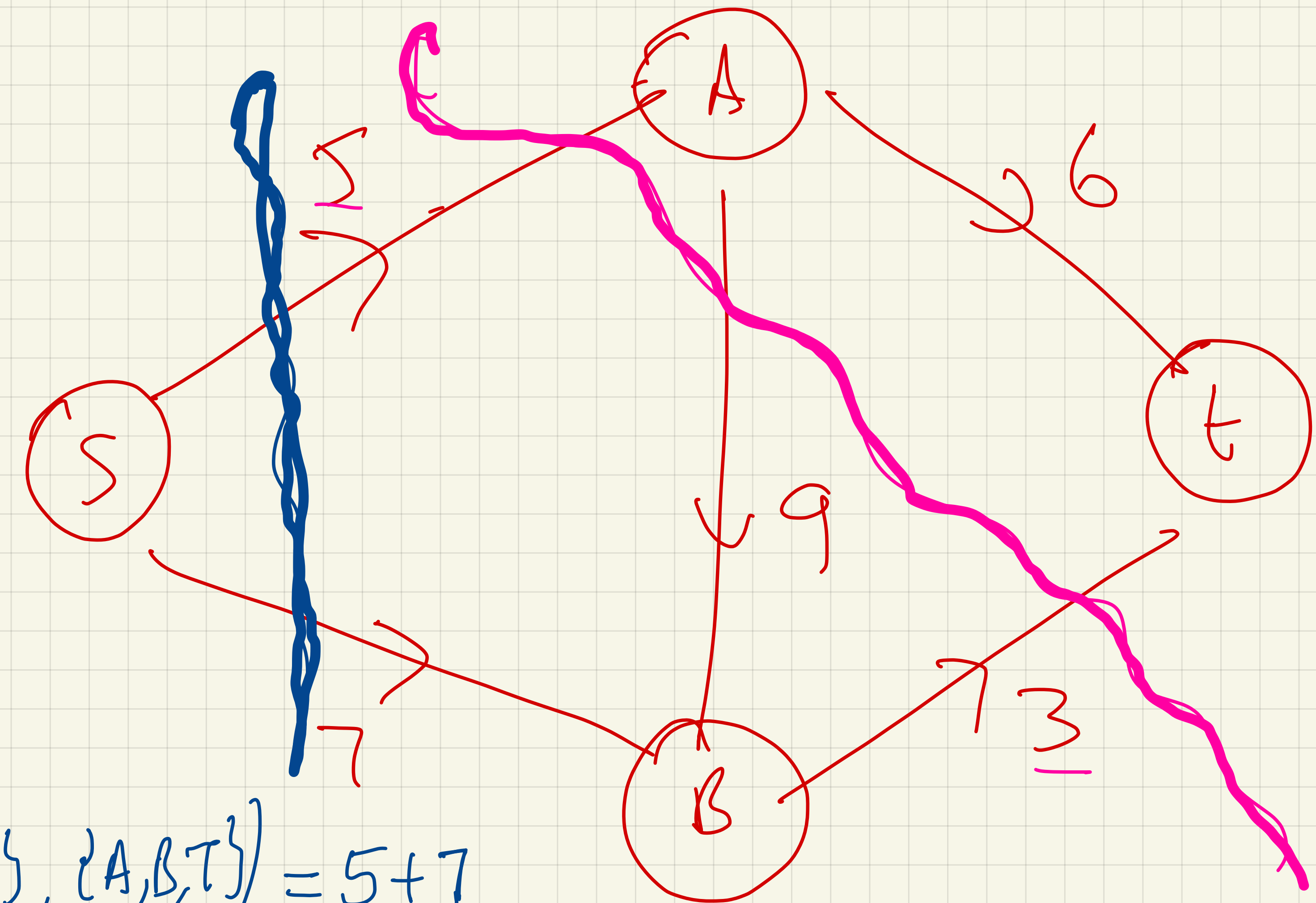


$$\boxed{\text{capacity}(L, R) = \sum_{\substack{u \to v \\ u \in L,\, v \in R}} c_{u \to v}}$$

<u>FACT:</u> For any flow f, and any cut (L, R)

$$\text{size}(f) \leq \text{capacity}(L, R)$$

CUTS:



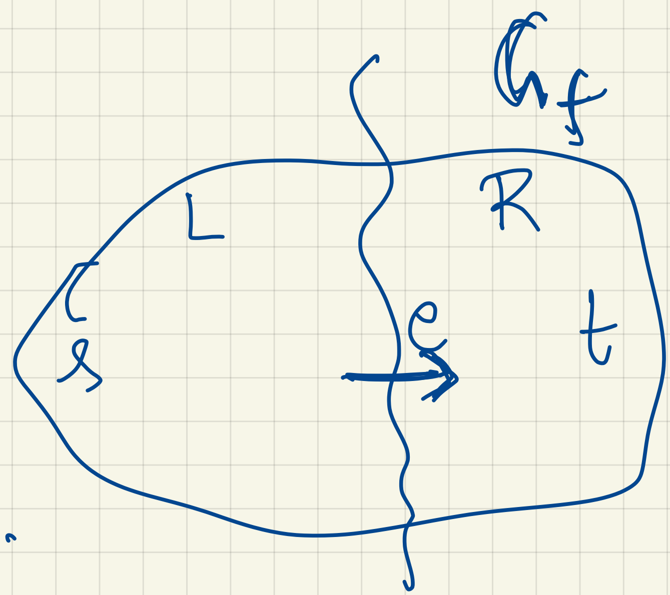
EXAMPLES:

$$\text{Capacity}(\{S\}, \{A,B,T\}) = 5+7 = 12$$

$$\text{Capacity}(\{S,B\}, \{A,t\}) = 5+3 = 8$$

Theorem: In any graph

Maximum s-t flow = Minimum s-t cut



Proof: 1) Run the algo

At termination $\nexists$ no path from $s \rightsquigarrow t$ in residual graph G_f.

L = vertices reachable from s in residual graph

R = remaining $V - L$.

$\nexists$ no residual capacity from L to R.

$\text{Size}(f) = \text{Capacity}(L, R)$

SURPRISE COROLLARY OF MAX FLOW ALGO

Corollary: If all capacities are integers

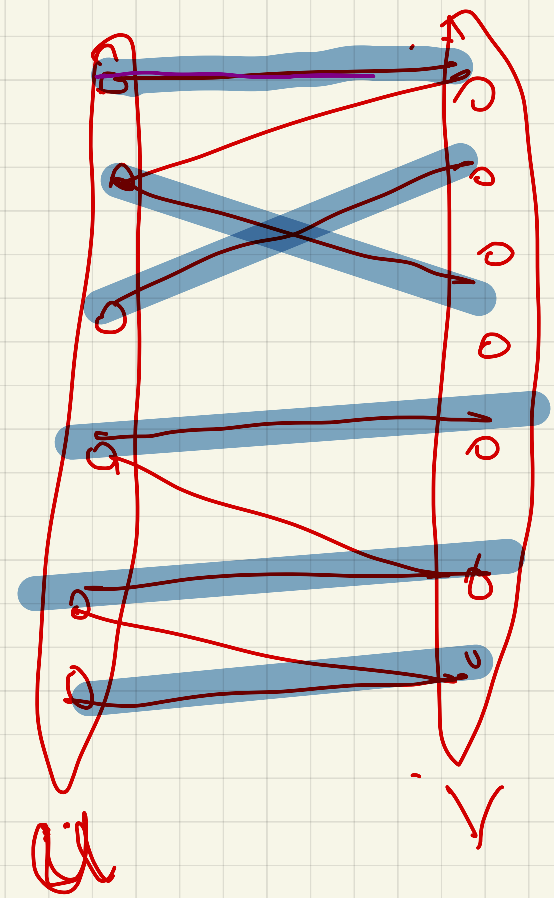
$\Longrightarrow$ Maximum flow is integral
(all flow values are integers)

Proof: If all capacities are integers
$\Rightarrow$ in each iteration, the algo adds an integer amount of flow
$\Rightarrow$ At termination all flow values are integers

Matching:

Input: Bipartite Graph $G = (V, E)$

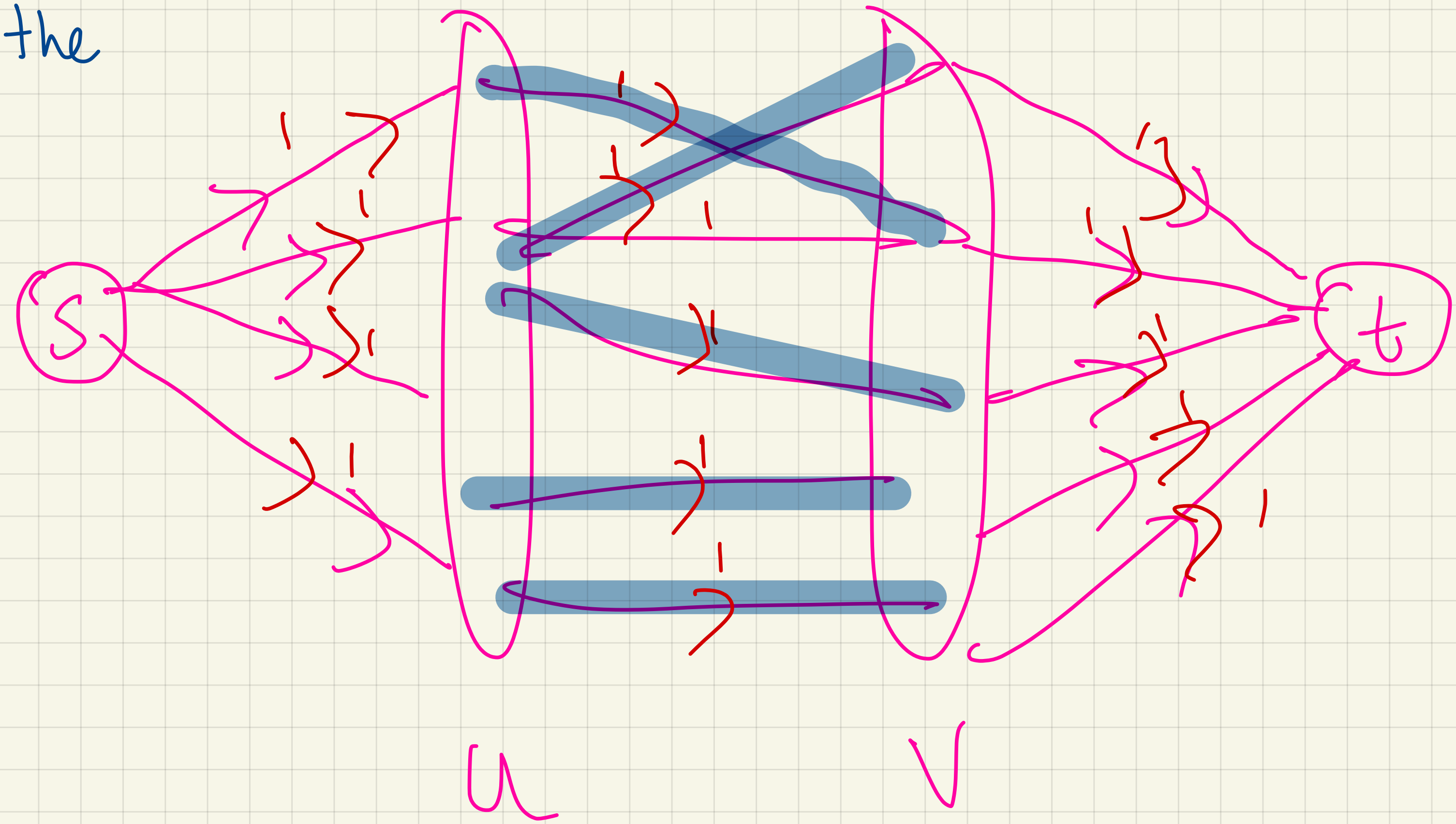
Goal: Find a matching between U and V.



$|U| = |V| = n$

USING MAX FLOW ALGO TO FIND MATCHINGS

Construct the graph ⟹



1) Find maximum flow from $s \longrightarrow t$.

2) If $\exists$ a matching from U to V

$\iff$ Send n units of flow