LINEAR PROGRAMS

General Linear Program

Canonical Form

n variables $x_1, \cdots x_n$, $n+m$ constraints

$$\max \sum_{i=1}^{n} c_i x_i$$

s.th. $x_i \geq 0 \quad i=1,\cdots n$

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i=1,\cdots,m$$

$$\boxed{\begin{aligned} &\max \ c^T x \\ &\text{s.t.} \ x \geq 0 \\ &\quad Ax \leq b \end{aligned}}$$

$x = \begin{pmatrix} x_1 \\ x_n \end{pmatrix}$ $c = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$

$b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$ $A = (a_{ij})$

Rem: If we don't have the constraints $x_i \geq 0$, we can transform to the canonical form by considering the $2n$ variables $x_i^{\pm}$, setting $x_i = x_i^+ - x_i^-$ and imposing the constraints $x_i^{\pm} \geq 0$

Simplex Algorithm

Feasible Region: Polytop of $x \in \mathbb{R}^n$ s.th. x satisfies all constraints

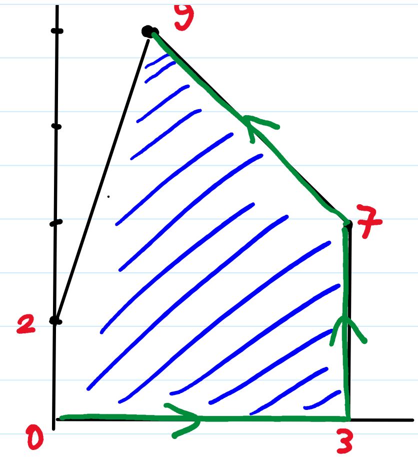
Vertex: Point x in the feasible region such that n of the constraints are tight, i.e., satisfied as equalities

Neighbor of a vertex x: vertex x' which differs in one the defining constraint

Value: $c^T x$

Simplex Algorithm

- Start at vertex x^*
- Find neighbor y^* with maximal value
- If value$(y^*) >$ value(x^*) move to y^*.
- Repeat



Running time

- $O(n^3 \cdot nm)$ per step
- Worst case exponentially many steps

Maximum Flow

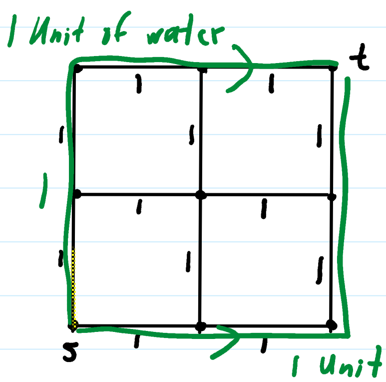
Input: 1) Directed graph $G=(V,E)$

2) "Source" vertex $s \in V$

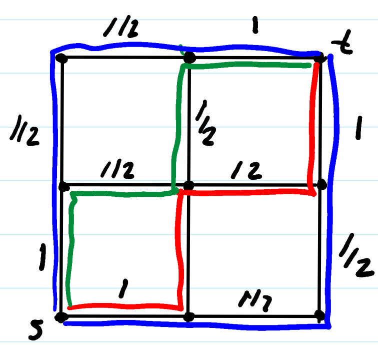
3) "Sink" vertex $t \in V$

4) For each edge $e \in E$ capacity $c_e \in \mathbb{N}$

Goal: Route maximum amount of water from s to t



Flow 2



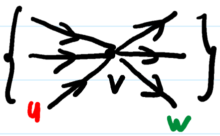
Flow 2

Def.: A flow f is an assignment of a number f_e to each directed edge $e \in E$ such that:

(Nonnegativity) $f_e \geq 0$ for all edges

(Capacity) $f_e \leq c_e$ — " —

(Flow-in = Flow-out) For each vertex $v \neq s, t$

Flow-in  Flow-out

$$\sum_{uv \in E} f_{uv} = \sum_{vw \in E} f_{vw}$$

Maximum Flow Problem

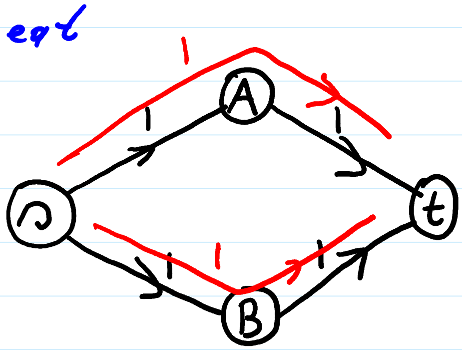
Maximize $\text{size}(f) := \sum_{v: sv \in E} f_{sv}$ (flow from s to t)

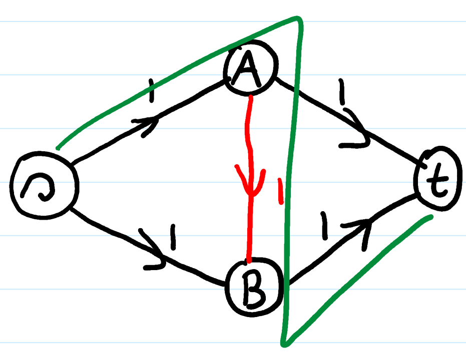
s.th. f is a flow

$\boxed{\text{This is a LP!}}$

Algorithm (first, greedy try)

1) Find a path P from s to t which is not yet saturated
2) Send more flow along P
3) Repeat





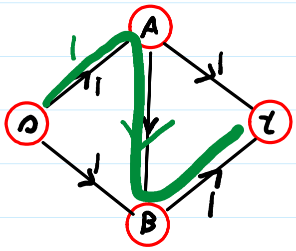
Flow $s \to A \to t$ 1 unit
Flow $s \to B \to t$ 1 unit
Total: 2 units

Flow $s \to A \to B \to t$: 1 unit

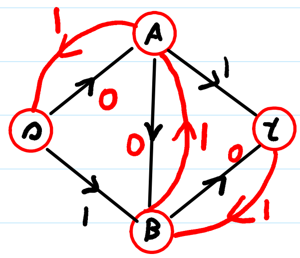
Wrong Answer

Need to undo flow $s \to A \to B \to t$

Algorithm to undo flow

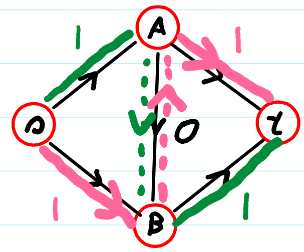


$s \to A \to B \to t$: one unit



"residual graph"

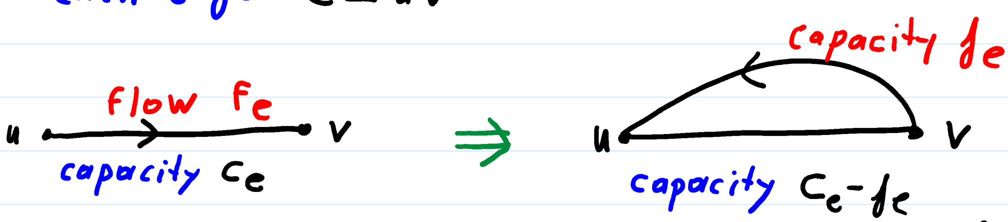
$s \to B \to A \to t$



2 units total

Def: Given a graph G and a flow f on G the residual graph G_f is constructed as follows:

For each edge $e = uv$

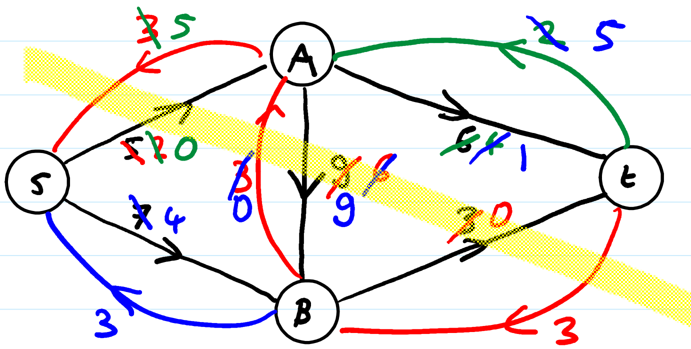


(original graph) (residual graph)

Ford Fulkerson Algorithm

1) Find path P from s to t which is not yet saturated in the residual graph
2) Send more flow along P
3) Repeat

Example:



$s \to A \to B \to t$ Flow 3
$s \to A \to t$ " 2
$s \to B \to A \to t$ Flow 3

} Total 8

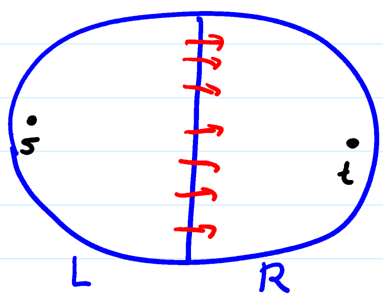
Proof this is opt.

Look at cut

Flow $s \rightsquigarrow t = 8$

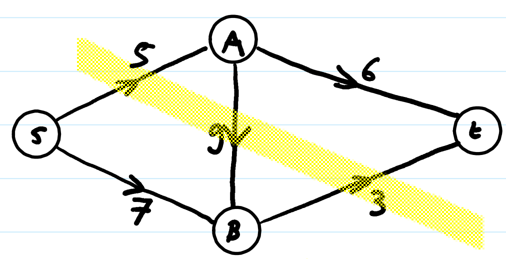
Rem: A path P which is not yet saturated in G_f (i.e., has >0 capacity for all edges in P) is called an augmenting path

Def.: An s-t cut is a partition $V = L \cup R$ of the vertex set such that $s \in L$ and $t \in R$



Def: The capacity of the cut is capacity$(L,R) := \sum_{\substack{u \in L, v \in R \\ uv \in E}} c_{u,v}$

Thm: $\boxed{\text{For any flow } f \text{ and any cut } (L,R) \text{ size}(f) \leq \text{capacity}(L,R)}$



Cor: Max Flow $\leq$ Min Cut

capacity of = 8

Thm: Max Flow = Min Cut

Need to show $\geq$

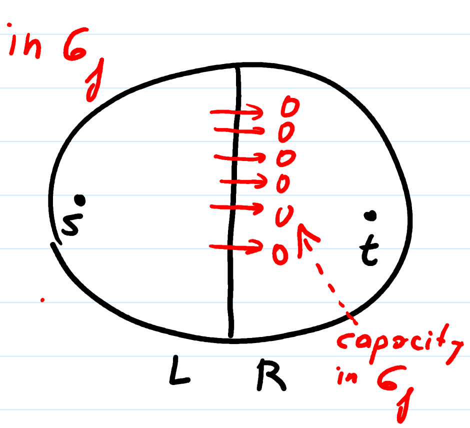
Idea: Run Ford-Fulkerson on G. Let f be the flow it outputs. Prove $\boxed{\text{size}(f) \geq \text{Min Cut}}$

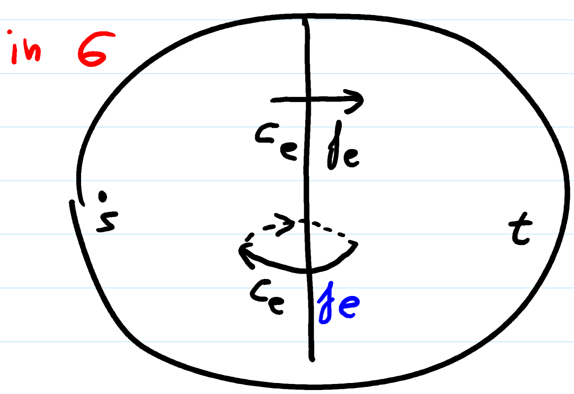
$\Rightarrow$ capacity$(L,R) \geq$ size$(f) \geq$ Min Cut $\quad \forall$ cuts (L,R)

$\Rightarrow$ Min Cut = Max Flow = Output of Ford-Fulkerson

Proof:

- When Ford-F. terminates, there exists no s-t path in residual graph with >0 capacity for all edges
- L = set of vertices reachable from s in G_f
  R everything else $\Rightarrow t \in R$ (since FF terminated)
- We will prove that $\boxed{\text{size}(f) = \text{capacity}(L,R)}$
  - First, note that for all edges uv crossing the cut from L to R ($u \in L, v \in R$), the capacity in G_f must be zero (otherwise, v would be reachable from s, and hence in L)





- This means that the flow f has the following

Property: Assume the $e = uv$ is a directed edge crossing the cut

a) If $u \in L, v \in R$ (LR-edge) $\quad f_{uv} = c_{uv}$

b) If $u \in R, v \in L$ (RL-edge) $\quad f_{uv} = 0$

Pf of a) In G_f, uv has capacity $c_{uv} - f_{uv}$ which is zero, see above. ✓

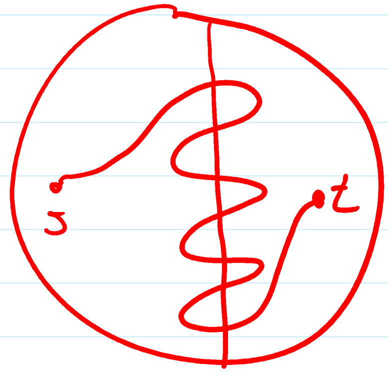
Pf of b) Assume $f_{uv} > 0$. Then the capacity of the residual edge vu would be $f_{uv} > 0$ in G_f. But vu is a LR-edge, and must have residual capacity 0. ✓

- This allows us to complete the proof:

$$\text{size}(f) = \sum_{\substack{u \in L \\ v \in R \\ uv \in E}} \underbrace{f_{u,v}}_{c_{uv}} - \sum_{\substack{u \in R \\ v \in L \\ uv \in E}} \underbrace{f_{u,v}}_{0} = \sum_{\substack{u \in L \\ v \in R \\ uv}} c_{u,v}$$

$$= \text{capacity}(L, R) \qquad \blacksquare$$

Q: Why was it important that $f_{u,v} = 0$ for right to left edges



A: In general, a path could go back and forth, so not all edges from L to R would contribute to the flow from s to t. If $f_{u,v} = 0$ for all $R \to L$ edges, this can't happen

Rem: If the capacities are integer, in each step Ford-Fulkerson updates the capacities by integer amounts, increasing size(f) by an integer amount $\Delta \geq 1$. So in particular, F.-F. uses at most U = max-flow many path

Runtime: $\underbrace{\text{\# of augmenting path}}_{\leq U = \text{max-flow}} \times \underbrace{\text{time to find paths}}_{O(n+m)}$ depth first search