

Flow network  $(D, c, s, t)$ \_\_\_\_\_

directed graph  $D = (V, E)$ ;  $|V| = n, |E| = m,$

capacity  $c : V \times V \rightarrow \mathbb{R}^+ \cup \{0\},$

such that  $c(uv) = 0$  if  $uv \notin E$

source  $s$ , sink  $t$

Notation: ordered pair  $(u, v)$  is denoted by  $uv$ .

Assumption: for every vertex  $v \in V$  there is an  $st$ -path through  $v$ . Thus  $m \geq n - 1$ .

$f : V \times V \rightarrow \mathbb{R}$  is a **flow** on the network  $(D, c, s, t)$  if it satisfies

1. **Capacity constraint:**

$$f(uv) \leq c(uv) \text{ for every } u, v \in V$$

2. **Skew symmetry:**

$$f(uv) = -f(vu) \text{ for every } u, v \in V$$

3. **Flow conservation:**

$$\sum_{v \in V} f(uv) = 0 \text{ for every } u \in V \setminus \{s, t\}$$

## The MaxFlow problem

---

value of flow  $f$ :  $|f| := \sum_{v \in V} f(sv)$ .

maximum flow of a network: a flow whose value is maximum over all flows of the network

### The Problem

Given a flow network  $(D, c, s, t)$ , **find a maximum flow**.

### Remark

*Existence* of a maximum flow is not self-evident. Follows by a (simple) compactness argument or from the *MaxFlow-MinCut Theorem* (see below).

## Further notation and basic properties\_\_\_\_\_

$$f(X, Y) := \sum_{x \in X} \sum_{y \in Y} f(xy).$$

**Examples** 1. Flow conservation property:

$$f(\{u\}, V) = 0 \text{ for every } u \in V \setminus \{s, t\}.$$

$$2. \text{ Value of flow } f: |f| = f(\{s\}, V)$$

**Lemma** Let  $f$  be a skew-symmetric function. Then

$$(i) f(X, X) = 0 \text{ for all } X \subseteq V$$

$$(ii) f(X, Y) = -f(Y, X) \text{ for all } X, Y \subseteq V$$

$$(iii) f(X \cup Y, Z) = f(X, Z) + f(Y, Z) \text{ and}$$

$$f(Z, X \cup Y) = f(Z, X) + f(Z, Y)$$

$$\text{for all } X, Y, Z \subseteq V \text{ with } X \cap Y = \emptyset$$

## Cuts in Flow Networks

---

**cut**  $(S, T)$  of a flow network  $(D, c, s, t)$  is a partition of  $V$  into  $S$  and  $T = V \setminus S$  such that  $s \in S$  and  $t \in T$

**capacity** of cut  $(S, T)$ :  $c(S, T) = \sum_{x \in S} \sum_{y \in T} c(xy)$

**minimum cut** of a network: one whose capacity is minimum over all (finitely many) cuts of the network.

**Lemma**  $f(S, T) = |f|$  for any cut  $(S, T)$ .

*Proof.*

$$\begin{aligned} |f| &= f(s, V) = f(s, V) + \sum_{w \in S \setminus \{s\}} \underbrace{f(w, V)}_{=0} \\ &= f(S, V) = f(S, T) + \underbrace{f(S, S)}_{=0} = f(S, T). \end{aligned}$$

**Special Case**  $|f| = f(V, t)$

**Corollary** For any flow  $f$  and any cut  $(S, T)$

$$|f| \leq c(S, T).$$

## Residual network

---

residual network  $(D_f, c_f, s, t)$

residual capacity  $c_f(uv) = c(uv) - f(uv) \geq 0$

residual digraph  $D_f = (V, E_f)$

where  $E_f = \{uv \in V \times V : c_f(uv) > 0\}$

**Remark** Edges in  $E_f$  are either edges in  $E$  or their reversals. Hence  $|E_f| \leq 2m$ .

**Lemma** Let  $f$  be a flow in the flow network  $D$  and let  $f'$  be a flow in the residual flow network  $D_f$ . Then  $f + f'$  is a flow in  $D$  with value  $|f| + |f'|$ .

## Augmenting paths

---

A *directed*  $st$ -path  $P$  in  $D_f$  is called an **augmenting path**.

residual capacity of  $P$ :

$$c_f(P) = \min\{c_f(uv) : uv \text{ is on } P\}$$

Define

$$f_P(uv) := \begin{cases} c_f(P) & \text{if } uv \text{ is on } P \\ -c_f(P) & \text{if } vu \text{ is on } P \\ 0 & \text{otherwise} \end{cases}$$

**Claim**  $f_P$  is a flow in  $D_f$ .

**Corollary**

$f + f_P$  is a flow in  $D$  with value larger than  $|f|$ .

## MaxFlow-MinCut Theorem\_\_\_\_\_

**Theorem** The following are equivalent.

- (i)  $f$  is a maximum flow
- (ii)  $D_f$  contains no augmenting paths.
- (iii)  $|f| = c(S, T)$  for some cut  $(S, T)$ .

*Proof.*

(i) $\Rightarrow$ (ii): By preceding corollary.

(iii) $\Rightarrow$ (i): Since value of any flow  $\leq$  capacity of any cut.

(ii) $\Rightarrow$ (iii): Set  $S := \{\text{vertices reachable from } s \text{ in } D_f\}$ . Assume no augmenting path. Then  $t \in T = V \setminus S$ . For all  $u \in S$  and  $v \in T$ ,  $c_f(uv) = 0$  (else  $v \in S$ ). Thus,  $f(S, T) = c(S, T)$ .

## Ford-Fulkerson Method

---

**Initialization**  $f \equiv 0$

WHILE there exists an augmenting path  $P$

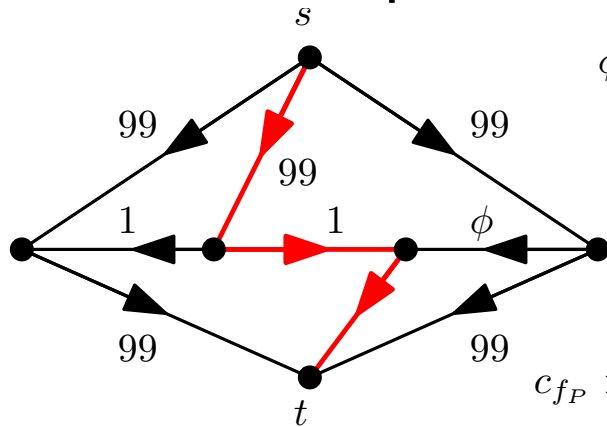
    DO augment flow  $f$  along  $P$

**return**  $f$

Running times:

- Basic (careless) Ford-Fulkerson: might not even terminate, flow value might not converge to maximum;  
when capacities are integers, it terminates in time  $O(m |f^*|)$ , where  $f^*$  is a maximum flow.
- Edmonds-Karp: chooses a *shortest* augmenting path; runs in  $O(nm^2)$

# Zwicky's Example



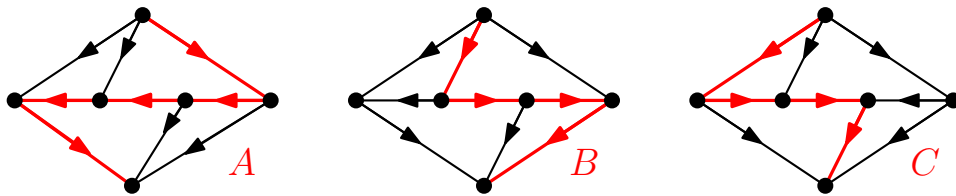
$$\phi = \frac{\sqrt{5}-1}{2} \approx 0.618, \quad 1 - \phi = \phi^2$$

augmenting path  $P$

$$|f_P| = 1$$

$$c_{f_P} \text{ for horizontal edges} = (1 = \phi^0, 0, \phi^1)$$

Assume residual cap's of horizontal edges =  $(\phi^{k-1}, 0, \phi^k)$



- (1) Augment along  $A$  by  $\phi^k$ ; new horiz. resid. cap. =  $(\phi^{k+1}, \phi^k, 0)$
- (2) Augment along  $B$  by  $\phi^k$ ; new horiz. resid. cap. =  $(\phi^{k+1}, 0, \phi^k)$
- (3) Augment along  $A$  by  $\phi^{k+1}$ ; new horiz. resid. cap. =  $(0, \phi^{k+1}, \phi^{k+2})$
- (4) Augment along  $C$  by  $\phi^{k+1}$ ; new horiz. resid. cap. =  $(\phi^{k+1}, 0, \phi^{k+2})$

Value of flow converges to  $1 + 2 \sum_{n=1}^{\infty} \phi^n = 4 + \sqrt{5} < 7$ .

## Preflow

---

$f$  is a **preflow** if the following hold.

1. Capacity constraint:

$$f(uv) \leq c(uv) \text{ for every } u, v \in V$$

2. Skew symmetry:

$$f(uv) = -f(vu) \text{ for every } u, v \in V$$

3. **Relaxed “flow conservation”**:

$$f(V, u) = \sum_{v \in V} f(vu) \geq 0 \text{ for every } u \in V \setminus \{s\}$$

**excess** flow into  $u$ :  $e(u) := f(V, u)$

vertex  $u \in V \setminus \{s, t\}$  is **overflowing** if  $e(u) > 0$

**Lemma** For any overflowing vertex  $u$  there is a  $us$ -path in the residual network  $D_f$ .

In particular, there is a residual outgoing edge from  $u$ .

## Height function

---

$h : V \rightarrow \mathbb{N}$  is a **height function** for a preflow  $f$  if

- $h(s) = n$ ,
- $h(t) = 0$ ,
- $h(u) \leq h(v) + 1$  for every residual edge  $uv \in E_f$

**Lemma** If  $f$  is a preflow which has a height function then there is no augmenting path in the residual network  $D_f$ .

**Corollary** If  $f$  is a flow which has a height function, then  $f$  is a maximum flow.

**Lemma** If  $f$  is a preflow with a height function  $h$ , then for any overflowing vertex  $u$  we have  $h(u) \leq 2n - 1$ .

## Initialization

---

Every flow network  $(D, c, s, t)$  has a preflow with a height function:

```
INITIALIZE-PREFLOW( $D, c, s, t$ )
FOR each pair  $uv \in V \times V$ 
  DO  $f(uv) := 0$ 
FOR each vertex  $u \in N^+(s)$ 
  DO  $f(su) := c(su)$ 
      $f(us) := -c(su)$ 
FOR each vertex  $u \in V$ 
  DO  $h(u) := 0$ 
 $h(s) := n$ 
```

**Claim** INITIALIZE-PREFLOW outputs a preflow  $f$  of  $(D, c, s, t)$  with a height function  $h$ .

The GENERIC-PUSH-RELABEL algorithm maintains a preflow with a height function while performing a series of basic operations (PUSHES and RELABELS) and eventually outputting a flow with a height function.

## The PUSH operation

---

$\text{PUSH}(u, v)$  is applicable if

- $u$  is overflowing,
- $c_f(uv) > 0$  (that is,  $uv \in E_f$ ), and
- $h(u) = h(v) + 1$

**Action:**  $d_f(uv) := \min\{e(u), c_f(uv)\}$  amount of flow is “pushed from  $u$  to  $v$ ”:

$$f(uv) := f(uv) + d_f(uv)$$

$$f(vu) := -f(uv)$$

**Remark:** Preflow changes, height function does not.

**Lemma** Old  $h$  is still height function for new preflow  $f$ .

Two types of PUSHes:

1. **saturating push:** if  $d_f(uv) = c_f(uv)$ .

After a saturating push  $uv$  becomes “saturated”, i.e.,  $c_f(uv)$  becomes 0.

2. **nonsaturating push:** if  $d_f(uv) = e(u) < c_f(uv)$ .

After a nonsaturating push  $u$  is no longer overflowing.

## The RELABEL operation

---

RELABEL( $u$ ) applies if

- $u$  is overflowing and
- $h(u) \leq h(v)$  for all residual edges  $uv \in E_f$ .

**Action:** Define new height for  $u$

$$h(u) := 1 + \min\{h(v) : uv \in E_f\}$$

**Remark:** Minimum is well-defined, because  $u$  is overflowing, so *there is* an outgoing residual edge.

**Remark:** Height function changes, preflow does not.

**Lemma** The updated  $h$  is again height function for the old preflow  $f$ .

**Remark:**  $s$  and  $t$  cannot be relabeled

## The GENERIC-PUSH-RELABEL Algorithm

INITIALIZE-PREFLOW( $D, c, s, t$ )

WHILE there exists an applicable push or relabel operation

    DO select an applicable push or relabel operation  
        and perform it

**Lemma** The algorithm maintains a preflow  $f$  and a height function  $h$  for  $f$  throughout.

**Observation** The height  $h(u)$  of a vertex never decreases.

## Correctness of the push-relabel method\_\_\_\_\_

**Theorem** If `GENERIC-PUSH-RELABEL` algorithm terminates then the preflow it computes is a maximum flow of the network  $D$ .

*Proof.* This follows from the corollary below and the fact that a flow with a height function is maximum.

**Lemma** If  $u$  is an overflowing vertex then either a push or a relabel operation applies to it.

*Proof.* If  $u$  is overflowing then there is an outgoing residual edge  $uv \in E_f$ . Either  $h(u) = h(v) + 1$  for some such  $v$ , in which case we can `PUSH( $u, v$ )`, or  $h(u) \leq h(v)$  for all  $uv \in E_f$ , in which case we can `RELABEL( $u$ )`.

**Corollary** At termination  $f$  is a flow.

## Termination and running time analysis\_\_\_\_\_

**Lemma** (Bound on RELABEL operations) The number of relabel operations is at most  $2n - 1$  per vertex and at most  $2n^2$  overall.

*Proof:* Any time during execution  $h(u) \leq 2n - 1$  for each vertex  $u \in V$ .

**Lemma** (Bound on saturating PUSHes) The number of saturating pushes is at most  $2nm$ .

*Proof:* Between two saturating pushes from  $u$  to  $v$  the height of  $v$  increases by at least 2.

**Lemma** (Bound on non-saturating PUSHes) The number of non-saturating pushes is at most  $4n^2(n + m)$ .

*Proof:* Estimate the change of the potential function  $\Phi = \sum_{v \in V, e(v) > 0} h(v)$  during the three basic operations.

**Theorem** The number of basic operations for the GENERIC-PUSH-RELABEL algorithm is at most  $O(n^2m)$ .

**Corollary** There is an implementation of the GENERIC-PUSH-RELABEL algorithm which runs in  $O(n^2m)$  on any flow network.

## Admissable edges and admissable digraph

$uv$  is an **admissable edge** if

- $c_f(uv) > 0$  and
- $h(u) = h(v) + 1$

**Admissable digraph:**  $D_{f,h} = (V, E_{f,h})$ , where  $E_{f,h}$  is the set of admissable edges.

**Lemma** The admissable digraph is acyclic.

*Proof.* Height decreases along edges.

**Observation** If  $u$  is overflowing and  $uv$  is an admissable edge then  $\text{PUSH}(u, v)$  applies.

$\text{PUSH}(u, v)$  does not create any new admissable edges, but it may cause  $uv$  to become inadmissable.

**Lemma** If  $u$  is overflowing and there are no admissable edges leaving  $u$ , then  $\text{RELABEL}(u)$  applies.

After  $\text{RELABEL}(u)$  there is at least one admissable edge leaving  $u$  and there are no admissable edges entering  $u$ .

## Notation

---

$D$  is given by (non-cyclic) neighbor lists:

$N(u)$  is the **neighbor list** of  $u$

$v$  is on  $N(u)$  if  $uv$  or  $vu \in E$

$head(N(u))$  points to the first vertex in  $N(u)$

$next-neighbor(v)$  points to the vertex following  $v$  in  $N(u)$

$next-neighbor(v) = \mathbf{NIL}$  if  $v$  is the last vertex of  $N(u)$

$current(u)$  points to the neighbor of  $u$  currently under consideration. Initially  $current(u)$  points to  $head(N(u))$ .

## Recall – Basic operations

---

PUSH( $u, v$ ) applies if

$u$  is overflowing and  $uv \in E_{f,h}$ . Then

$$d_f(uv) := \min\{e(u), c_f(uv)\}$$

$$f(uv) := f(uv) + d_f(uv)$$

$$f(vu) := -f(uv)$$

$$e(u) := e(u) - d_f(uv)$$

$$e(v) := e(v) + d_f(uv)$$

RELABEL( $u$ ) applies if  $u$  is overflowing and  $uv \notin E_{f,h}$  for all  $v \in V$ . Then

$$h(u) := 1 + \min\{h(v) : uv \in E_f\}$$

## Discharging a vertex

---

DISCHARGE( $u$ )

```
1  WHILE  $e(u) > 0$ 
2      DO  $v := \text{current}(u)$ 
3          IF  $v = \text{NIL}$ 
4              THEN RELABEL( $u$ )
5                   $\text{current}(u) := \text{head}(N(u))$ 
6          ELSEIF  $c_f(u, v) > 0$  and  $h(u) = h(v) + 1$ 
7              THEN PUSH( $u, v$ )
8          ELSE  $\text{current}(u) := \text{next-neighbor}(v)$ 
```

**Lemma** (Algorithm DISCHARGE is well-defined)

When DISCHARGE calls PUSH( $u, v$ ) then a push operation applies to  $uv$ .

When DISCHARGE calls RELABEL( $u$ ) then a relabel operation applies to  $u$ .

## The RELABEL-TO-FRONT( $D, c, s, t$ ) algorithm

```
1   $f := \text{INITIALIZE-PREFLOW}(D, c, s, t)$ 
2   $L := \text{any order of } V \setminus \{s, t\}$ 
3  FOR each  $u \in V \setminus \{s, t\}$ 
4      DO  $\text{current}(u) := \text{head}(N(u))$ 
5   $u := \text{head}(L)$ 
6  WHILE  $u \neq \text{NIL}$ 
7      DO  $\text{old-height} := h(u)$ 
8          DISCHARGE( $u$ )
9          IF  $h(u) > \text{old-height}$ 
10             THEN move  $u$  to the front of the list  $L$ 
11              $u := \text{next}_L(u)$ 
```

### **Theorem** (Correctness)

The RELABEL-TO-FRONT algorithm is an implementation of the GENERIC-PUSH-RELABEL algorithm.

*Proof.* At each test in line 6 of the algorithm the list  $L$  is a topological sort of the vertices (except  $s, t$ ) of the admissible digraph  $D_{f,h}$  and no vertex in the list before  $u$  has excess flow.

## Running time analysis

---

**Theorem** The running time of RELABEL-TO-FRONT on any flow network is  $O(n^3)$

*Proof*

“Phase”: time between two relabel operations.

There are at most  $O(n^2)$  relabel operations.

Hence there are at most  $O(n^2)$  phases.

Each phase consists of  $\leq n$  calls to DISCHARGE (in each phase, we go through  $L$  at most once).

Hence, the total time of the WHILE loop excluding the work DISCHARGE does is  $O(n^3)$ .

## Time spent during DISCHARGE\_\_\_\_\_

The  $O(n^2)$  relabel operations can be done in  $O(nm)$   
(Homework)

Updating  $current(u)$  in line 8 of DISCHARGE occurs  $O(\deg(u))$  times each time a vertex  $u$  is relabeled and  $O(n \deg(u))$  times over for the vertex. All together the time is  $O(nm)$  (Handshaking Lemma).

The overall number of saturating pushes is  $O(nm)$

There is at most one non-saturating push per call to DISCHARGE

Hence the number of nonsaturating pushes is at most  $O(n^3)$ .