Writtn #1: Conservation of Flow in Ford–Fulkerson

Goal. We show that Ford–Fulkerson maintains the conservation constraint during augmentation:

$$\forall x \in V \setminus \{s, t\}, \quad f^{\text{in}}(x) = f^{\text{out}}(x).$$

Let (w, x) denote the incoming edge into vertex x and (x, y) the outgoing edge. Four possible combinations arise depending on the edge directions in the residual graph G_r:

Case 1: (w, x) forward and (x, y) forward,
Case 2: (w, x) forward and (x, y) backward,
Case 3: (w, x) backward and (x, y) forward,
Case 4: (w, x) backward and (x, y) backward.

We prove that conservation is preserved for Cases 3 and 4, where x is incident to backward residual edges.

Case 3: (w, x) backward and (x, y) forward

We must show that conservation still holds, i.e. $\forall x \neq s, t$, $f^{\text{in}}(x) = f^{\text{out}}(x)$, where $f^{\text{in}}(x)$ represents the total flow entering x in the *original* flow network.

Because (w, x) is a backward residual edge, it corresponds to the real edge (x, w) in the original network. Sending b units along (w, x) in G_r decreases the existing flow on (x, w) by b, which affects x's outflow but not its inflow. Thus,

$$f^{\text{in}}(x) = \text{old } f^{\text{in}}(x).$$

Next, since (x, y) is a forward residual edge, the algorithm increases the real flow on (x, y) by b. At the same time, the backward traversal along (w, x) reduced x's outgoing flow by b. Hence, the net change in outflow is zero:

$$f^{\text{out}}(x) = \text{old } f^{\text{out}}(x) - b + b = \text{old } f^{\text{out}}(x).$$

Therefore,

$$f^{\text{in}}(x) = f^{\text{out}}(x),$$

and the conservation constraint holds for Case 3.

Case 4: (w, x) backward and (x, y) backward

Here both residual edges are backward. The backward edge (w, x) corresponds to the real edge (x, w) in the original network, so traversing it reduces the existing flow on (x, w) by b, decreasing x's outflow:

$$\text{new } f^{\text{out}}(x) = \text{old } f^{\text{out}}(x) - b.$$

Likewise, (x, y) being backward corresponds to the real edge (y, x) in the original network. Reducing its flow by b decreases the inflow to x:

$$\text{new } f^{\text{in}}(x) = \text{old } f^{\text{in}}(x) - b.$$

Thus,

$$\text{new } f^{\text{in}}(x) = \text{new } f^{\text{out}}(x),$$

and the conservation constraint is preserved for Case 4. $\square$

Python Code

```python
def update(e, reverse, b):
    """ Updates the given edge and the corresponding edge in the
        reverse direction to add the given amount of flow.

        e -- an edge in a flow graph represented as a HalfEdge
        reverse -- the edge that gies between the same vertices as e
                   but in the opposite direction
        b -- a positive number less than or equal to e.residual()
    """
    # cancel flow in opposite-direction first
    c = min(b, reverse.flow())
    if c > 0:
       # decrease reverse flow
       reverse.add_flow(-c)
       # incr reverse and decrease forward residuals
       reverse.add_residual(c)
       e.add_residual(-c)

    # add the remainder to the intended direction
    r = b - c
    if r > 0:
       # ensure we don't increase more than capacity
       safe_add = min(r, e.residual())
       if safe_add > 0:
          # increase forward flow
          e.add_flow(safe_add)
          # decr forward residual and incr reverse residual
          e.add_residual(-safe_add)
          reverse.add_residual(safe_add)
```

Written Problem #2: Correctness of `update()` for Bidirectional Edges

WTS. Show that the `update` function preserves the *capacity constraint* and the new *single-direction flow constraint*.

Definitions.

Capacity constraint. For every directed edge (u, v) in G,

$$0 \leq f(u, v) \leq c(u, v),$$

where $c(u, v)$ is the edge capacity and $f(u, v)$ is the current flow. This ensures that flow on any edge never exceeds its capacity and never becomes negative.

Single-direction flow constraint. Even when both (u, v) and (v, u) exist, flow may go in at most one direction:

$$f(u, v) > 0 \;\Rightarrow\; f(v, u) = 0,$$

or equivalently, $f(u, v)\, f(v, u) = 0$ for all vertex pairs (u, v).

Note. By the Ford–Fulkerson invariant, we may assume that before each call to `update()`, the current flow satisfies the *conservation constraint* at all vertices except the source and sink. We do not need to prove that this property is maintained, since it is ensured by the algorithm and code structure.

Precondition. Before `update()` is called, assume that the current flow satisfies both the capacity and single-direction constraints.

Step 1: Cancelling flow in the opposite direction.

We first set

$$c = \min(b,\ f(v, u)).$$

Because $c \leq f(v, u)$, the operation `reverse.add_flow(`$-c$`)` decreases (never increases) the reverse flow. Hence the new flow satisfies

$$f'(v, u) = f(v, u) - c \geq 0$$

and remains bounded above by $c(v, u)$. Therefore, the capacity constraint is preserved on both directions.

Single-direction constraint. At this point, we only decrease the flow on the reverse edge, and have not yet increased the forward flow. Thus after Step 1, either the reverse flow is unchanged or reduced—possibly to zero—while the forward flow remains the same. It is therefore impossible for both directions to carry positive flow simultaneously.

Step 2: Updating flow in the forward direction.

Let $r = b - c \geq 0$ be the remaining augmentation budget after Step 1, where

$$c = \min\{b,\ f(v,u)\}.$$

Define

$$\text{safe\_add} = \min\{r,\ \text{residual}(u,v)\}, \quad \text{where} \quad \text{residual}(u,v) = c(u,v) - f(u,v).$$

Capacity constraint. We set

$$f'(u,v) = f(u,v) + \text{safe\_add}, \qquad f'(v,u) \text{ unchanged in this substep.}$$

Because $\text{safe\_add} \leq \text{residual}(u,v) = c(u,v) - f(u,v)$,

$$f'(u,v) = f(u,v) + \text{safe\_add} \leq f(u,v) + (c(u,v) - f(u,v)) = c(u,v),$$

and clearly $f'(u,v) \geq 0$. Thus $0 \leq f'(u,v) \leq c(u,v)$, preserving the capacity constraint.

Single-direction flow constraint. We consider two exhaustive cases based on Step 1:

Case A: $f(v,u) \geq b$. Then $c = b$ and $r = b - c = 0$, so $\text{safe\_add} = 0$ and we do not increase $f(u,v)$ at all. After Step 1, $f'(v,u) = f(v,u) - b \geq 0$ and $f'(u,v) = f(u,v)$. If $f'(v,u) > 0$, then $f'(u,v)$ did not increase, so both cannot be positive simultaneously.

Case B: $f(v,u) < b$. Then $c = f(v,u)$, so cancellation makes $f'(v,u) = f(v,u) - c = 0$ after Step 1. Now $r = b - c > 0$, and we may add forward flow by $\text{safe\_add} \leq r$. After the forward update we have $f'(u,v) \geq 0$ and still $f'(v,u) = 0$. Thus only one direction can be positive.

Conclusion. In both cases, after Step 2 at most one of $f'(u,v)$ and $f'(v,u)$ is positive. Therefore, the `update` function preserves both the *capacity* and *single-direction* constraints, assuming the invariants held beforehand.