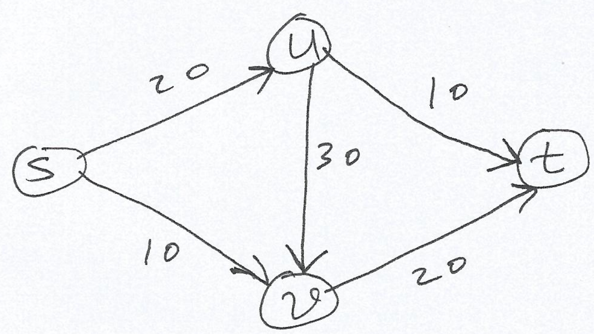
The Maximum-Flow Problem:

A <u>flow network</u> is a directed graph $G=(V,E)$ with the following features:

1. Associated with each edge e is a capacity, which is a nonnegative integer c_e.
2. There is a single source node $s \in V$. No edge enters the source s.
3. There is a single sink node $t \in V$. No edge leaves the sink t.
4. Nodes other than s and t are called internal nodes. There is at least one edge incident to each node.

<u>Example</u>:



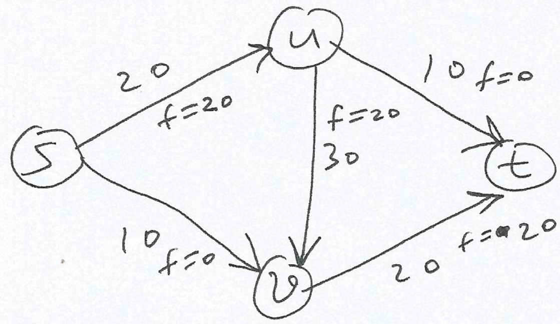
An <u>s-t flow</u> is a function f that maps each edge e to a nonnegative real number, $f: E \to R^+$.

The value $f(e)$ represents the amount of flow carried by edge e. A flow f must satisfy the following two properties:

1. <u>Capacity conditions</u>: For each $e \in E$, we have $0 \le f(e) \le c_e$.
2. <u>Conservation conditions</u>: For each node v other than s and t, we have $\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e)$

s is the producer of flow and t is the consumer of flow

Example:



flow f

The value of a flow f, denoted $v(f)$, is defined to be the amount of flow generated at the source:

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

We define $$f^{out}(v) = \sum_{e \text{ out of } v} f(e)$$

and $$f^{in}(v) = \sum_{e \text{ into } v} f(e)$$

If $S \subseteq V$, we define $$f^{out}(S) = \sum_{e \text{ out of } S} f(e)$$

and $$f^{in}(S) = \sum_{e \text{ into } S} f(e)$$

Conservation condition: $f^{in}(v) = f^{out}(v) \quad \forall v \in V - \{s, t\}$

$$v(f) = f^{out}(s)$$

Example: In the above example, we have:

$$v(f) = f^{out}(s) = f(su) + f(sv) = 20 + 0 = 20$$

$$f^{in}(u) = f(su) = 20$$

$$f^{out}(u) = f(uv) + f(ut) = 20 + 0 = 20$$

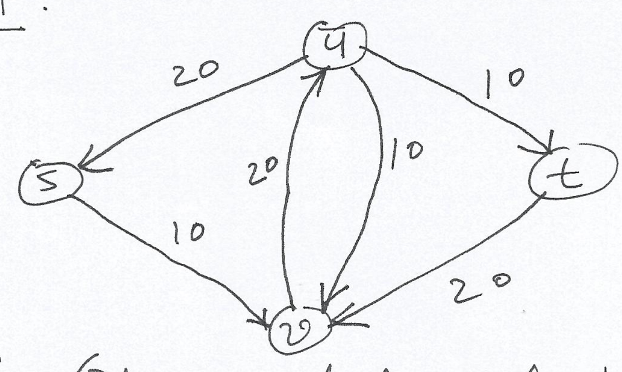
$$\Rightarrow f^{in}(u) = f^{out}(u)$$

The Maximum - Flow Problem is : Given a Flow Network, find a flow of maximum possible value.

The Residual Graph : Given a flow network G, and a flow f on G, we define the residual graph G_f of G with respect to f as follows:

1. The node set of G_f is the same as that of G.
2. For each edge $e = (u, v)$ of G on which $f(e) < c_e$, there are $c_e - f(e)$ "leftover" units of capacity on which we could try pushing flow forward. So we include the edge $e = (u, v)$ in G_f, with a capacity of $c_e - f(e)$. We will call edges included this way forward edges.
3. For each edge $e = (u, v)$ of G on which $f(e) > 0$, there are $f(e)$ units of flow that we can "undo" if we want to, by pushing flow backward. So we include the edge $e' = (v, u)$ in G_f, with a capacity of $f(e)$. Note that e' has the same ends as e, but its direction is reversed; we will call edges included this way backward edges.

Example : G_f :



The capacities in G_f are called residual capacity

Augmenting Paths in a Residual Graph:

Let P be a simple $s-t$ path in G_f (P does not visit any node more than once). We define bottleneck(P, f) to be the minimum residual capacity of any edge on P, with respect to the flow f. Now we define the following operation augment(f, P), which yields a new flow f' in G:

```
augment (f, P)
    Let b = bottleneck (P, f)
    For each edge (u, v) ∈ P
        If e = (u, v) is a forward edge then
            increase f(e) in G by b
        Else ((u, v) is a backward edge, and let
              e = (v, u))
                decrease f(e) in G by b
        Endif
    End for
    Return(f)
```

Example: Consider the path $s-v-u-t$ in G_f.

bottleneck$(P, f) = 10$. $s-v$ in G_f is a forward edge.

$\Rightarrow f'(s-v) = f(s-v) + 10 = 0 + 10 = 10.$

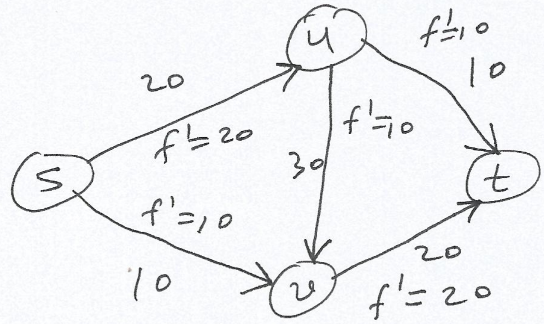
$v-u$ in G_f is a backward edge

$\Rightarrow f'(u-v) = f(u-v) - 10 = 20 - 10 = 10$

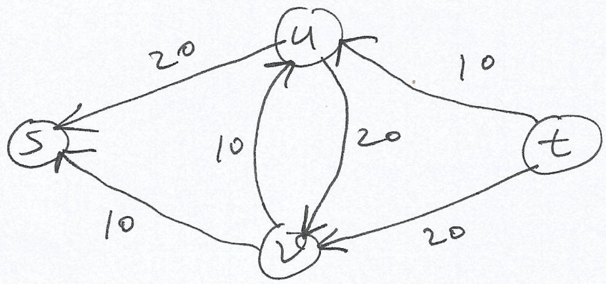
$u-t$ in G_f is a forward edge.

$\Rightarrow f'(u-t) = f(u-t) + 10 = 0 + 10 = 10$

flow f':



$G_{f'}$:



f' is a flow in G: We must verify the capacity and conservation conditions.

① Capacity conditions: Since f' differs from f only on edges of P, we need to check the capacity condition only on these edges. Let (u, v) be an edge of P. By the definition of bottleneck (P, f), it is no longer than the residual capacity of (u, v).

Case A: If $e = (u, v)$ is a forward edge, then its residual capacity is $c_e - f(e) \Rightarrow$

$$\text{bottleneck}(P, f) \leq c_e - f(e)$$

$$0 \leq f(e) \leq f'(e) = f(e) + \text{bottleneck}(P, f)$$

$$\leq f(e) + (c_e - f(e)) = c_e$$

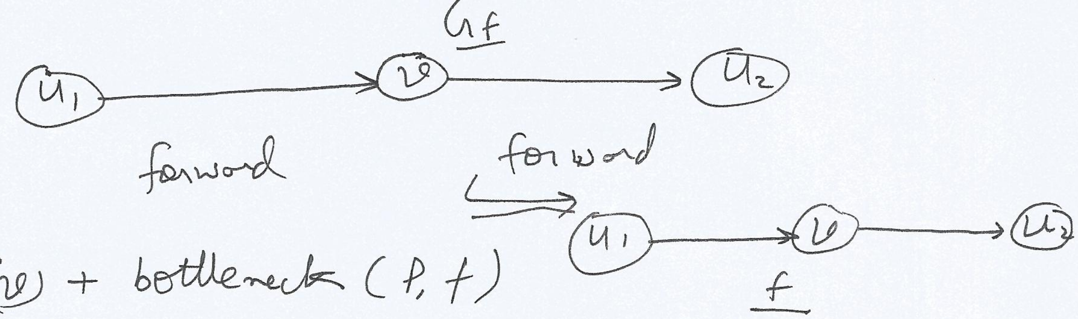
Case B: If (u, v) is a backward edge arising from edge $e = (v, u) \in E$, then its residual capacity is $f(e)$

$\Rightarrow$ bottleneck $\leq f(e)$

$$c_e \geq f(e) \geq f'(e) = f(e) - \text{bottleneck}(P, f) \geq f(e) - f(e) = 0$$

(2) Conservation Conditions: we need to check the conservation condition at each internal node that lies on the path P. Let v be such a node; we can verify that the change in the amount of flow entering v is the same as the change in the amount of flow exiting v; since f satisfied the conservation condition at v, so must f'.

Case A:

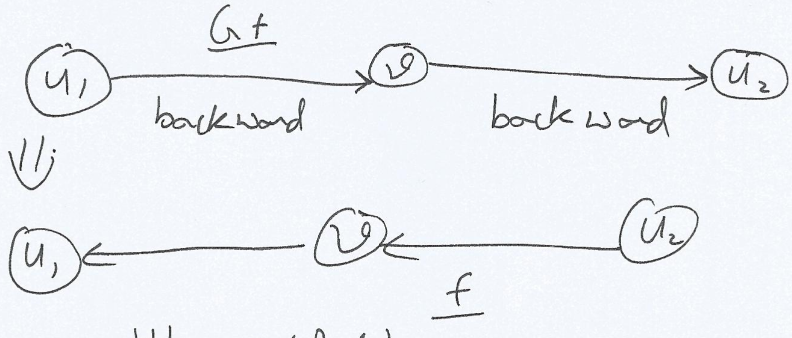


$$f'^{in}(v) = f^{in}(v) + \text{bottleneck}(P, f)$$

$$f'^{out}(v) = f^{out}(v) + \text{bottleneck}(P, f)$$

$$f^{in}(v) = f^{out}(v) \Rightarrow f'^{in}(v) = f'^{out}(v)$$

Case B:

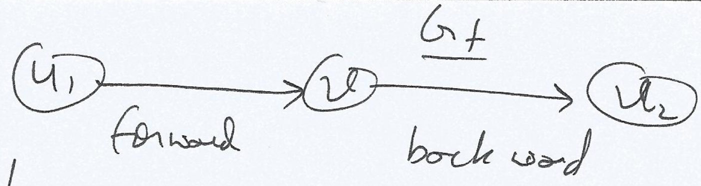


$$f'^{in}(v) = f^{in}(v) - \text{bottleneck}(P, f)$$

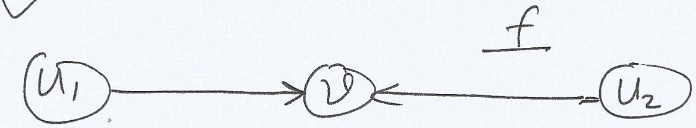
$$f'^{out}(v) = f^{out}(v) - \text{bottleneck}(P, f)$$

$$f^{in}(v) = f^{out}(v) \Rightarrow f'^{in}(v) = f'^{out}(v)$$

Case C:



$\Downarrow$

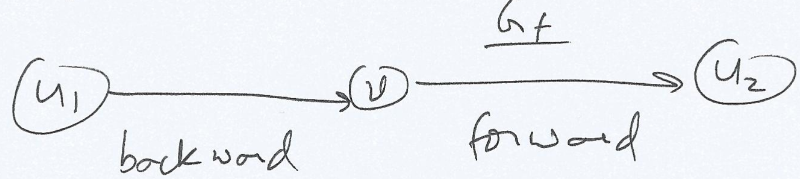


$$f'^{in}(v) = f^{in}(v) + \text{bottleneck}(P, f) - \text{bottleneck}(P, f) = f^{in}(v)$$

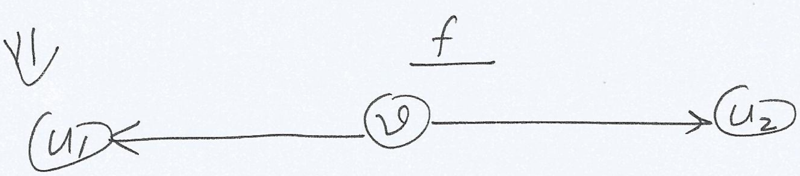
$$f'^{out}(v) = f^{out}(v)$$

$$f^{in}(v) = f^{out}(v) \Rightarrow f'^{in}(v) = f'^{out}(v)$$

Case D:



$\Downarrow$



$$f'^{in}(v) = f^{in}(v)$$

$$f'^{out}(v) = f^{out}(v) + \text{bottleneck}(P, f) - \text{bottleneck}(P, f) = f^{out}(v)$$

$$f^{in}(v) = f^{out}(v) \Rightarrow f'^{in}(v) = f'^{out}(v)$$

Ford-Fulkerson Algorithm

Max-Flow
    Initially $f(e) = 0$ for all e in G
    While there is an s-t path in the residual graph G_f
        Let P be a simple s-t path in G_f
        $f' = \text{augment}(f, P)$
        Update f to be f'
        Update the residual graph G_f to be $G_{f'}$
    Endwhile
    Return f

Complexity of Ford-Fulkerson Algorithm

We make the following observations about the Ford-Fulkerson Algorithm:

1. At every intermediate stage of the Ford-Fulkerson Algorithm, the flow values $\{f(e)\}$ and the residual capacities in G_f are integers.
2. Let f be a flow in G, and let P be a simple s-t path in G_f. Then $v(f') = v(f) + \text{bottleneck}(P, f)$, and since $\text{bottleneck}(P, f) > 0$, we have $v(f') > v(f)$.
3. The Ford-Fulkerson Algorithm terminates in at most C iterations of the while loop, where $C = \sum_{e \text{ out of } s} c_e$
4. One iteration of the while loop in the Ford-Fulkerson Algorithm can be implemented in $O(|E|)$ time

We have assumed that all nodes have at least one incident edge $\Rightarrow |E| \geq |V|/2 \Rightarrow O(|V| + |E|) = O(|E|)$.
The residual graph G_f has at most $2|E|$ edges, since each edge of G gives rise to at most two edges in the residual graph. We will maintain G_f using an adjacency list representation; we will have two linked lists for each node v, one containing the edges entering v, and one containing the edges leaving v. To find an s-t path in G_f, we can use breadth-first search or depth-first search, which runs in $O(|V| + |E|) = O(|E|)$ time. The procedure augment(f, P) takes $O(|V|)$ time, as the path P has at most $|V| - 1$ edges. Given the new flow f', we can build the new residual graph in $O(|E|)$ time: For each edge e of G, we construct the correct forward and backward edges in $G_{f'}$.
$\Rightarrow$ time complexity of Max-Flow is $\boxed{O(|E|C)}$