MA2000: Combinatorial Optimization

Madhab Barman and Nachiketa Mishra

Indian Institute of Information Technology, Design & Manufacturing, Kancheepuram

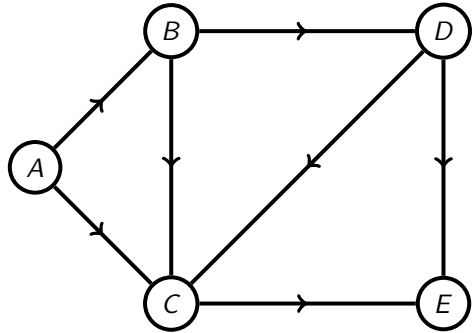
Flows

Lecture notes:

- Lecture note 1
- Lecture note 2

Applications:

- Water/oil/fluids through pipes
- Electrical current through circuit
- Train along rail network
- Data through computer network



A digraph is short for directed graph, and it is a diagram composed of points called vertices (nodes) and arrows called arcs going from a vertex to a vertex.

Definition (Digraph)

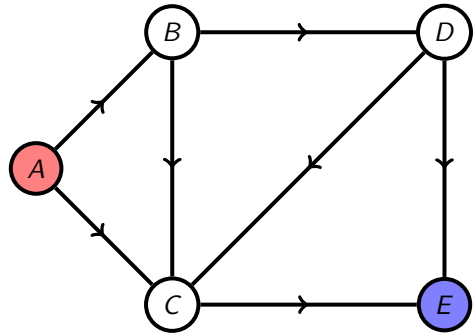
A digraph or directed graph is an ordered pair of sets $G = (\mathcal{V}, E)$, where $\mathcal{V}$ is a set of vertices and $\mathcal{E}$ is a set of ordered pairs (called arcs, edges) of vertices of $\mathcal{V}$.

Flow

- We send things through a digraph G from a source node/vertex to a sink/target node/vertex.
- From this time on, we denote source as $s = A$ and sink node $t = E$

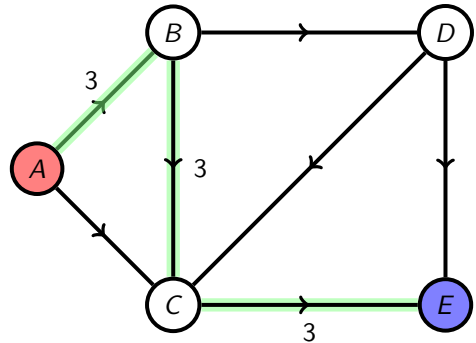
Assumptions

- G is directed and simple (no loops, no multiple edges)
- Source s has edges going out, but not in; Sink t has edges going in, but not out.
- If $(x \to y)$ is an edge, then $(y \to x)$ isn't.
- *These simplifications will be removed later.*



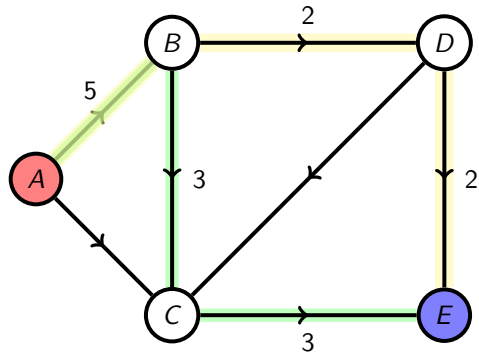
Flow

- Send 3 units from source to sink via ABCE path



Flow

- Send 3 units from source to sink via ABCE path
- Also, send 2 units from source to sink via ABDE path
- Both paths contribute to the flow on edge AB
- We consider **static flows** (time independent); 5 units/sec of AB, 2 units/sec on BD, etc.
- A static *st*-flow means that a static flow from a source s to a sink t



Flow function

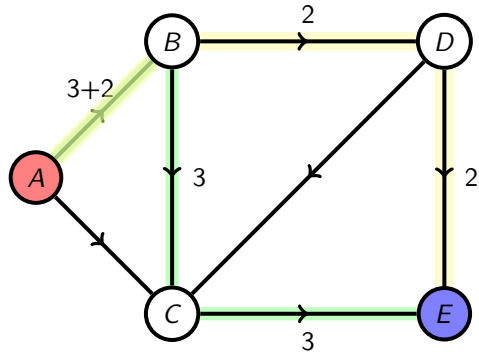
Definition (Flow)

A flow from $s \to t$ is a function $f : \mathcal{E} \to \mathcal{R}^{\geq 0}$ from the edges to nonnegative reals, such that from all vertices x except s and t,

$$\sum_w f(w \to x) = \sum_y f(x \to y)$$

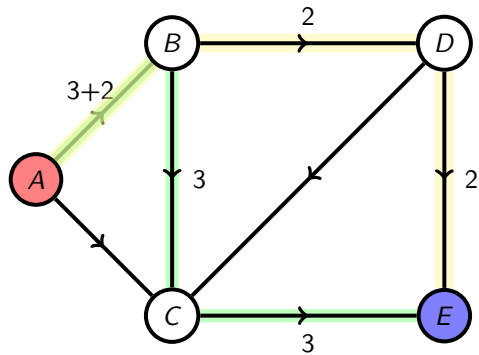
flow into x = flow out of x

- This equation is called **Kirchhoff's Law**
- at $B : 5 = 2 + 3$
- at $C : 3 = 3$
- at $D : 2 = 2$
- $f(x \to y) = 0$ if not shown or if $(x \to y)$ is not an edge!



Flow function

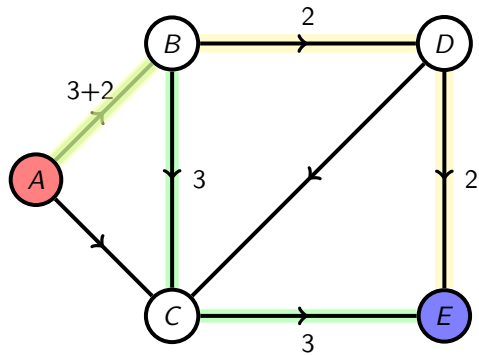
- We started with a flow $f(x \to y) = 0$ everywhere
- We added green path ABCE:
  - increase in & out flow at B and C by 3, keeping them balanced
  - Adds 3 to out-flow at source A or in-flow at sink E, so they are not balanced.
- yellow path ABDE:
  - increase in & out flow at B and D by 2, keeping them balanced
  - Adds 2 to out-flow at source A or in-flow at sink E, so they are not balanced.



Value of a flow

- The value of a flow at source A is

$$|f| \text{ or } v(f) = \sum_y f(x \to y) - \sum_w f(w \to x)$$
$$= \text{total net flow out of } s$$
$$= 5 - 0 = 5$$



- The value of a flow at sink E is

$$|f| \text{ or } v(f) = \sum_w f(w \to x) - \sum_y f(x \to y)$$
$$= (3 + 2) - 0 = 5$$

  - Usually, there are no edges into source or out from sink, so the subtracted sum is 0.
  - However, this formula allows for such edges (in which case the subtracted sums might not be zero)

Value of a flow

- Let $\delta f(x)$ denote the total net flow out of any vertex x:

$$\delta f(x) = \sum_y f(x \to y) - \sum_w f(w \to x) = \text{out flow at } x - \text{in flow at } x$$

- The conservation constraint implies: $\delta f(x) = 0$ for every vertex x except s and t, so

$$\sum_x \delta f(x) = \delta f(s) + \delta f(t).$$

- Any flow that leaves one vertex must enter another vertex, we must have

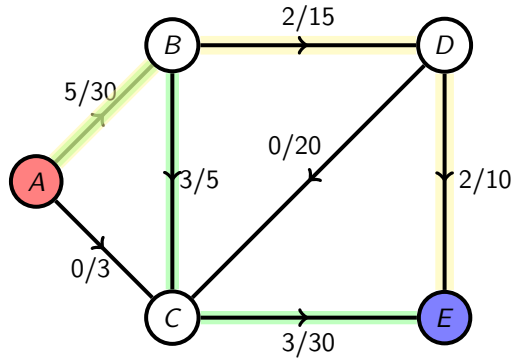
$$\sum_x \delta f(x) = 0 \implies |f| = \delta f(s) = -\delta f(t)$$

Capacities

- Each edge has an upper limit on its flow, called the capacity

Example

- A 2 inch steel water pipe has a maximum flow of 45 gallons/minute
- Wi-Fi download speed capped at 25 Mbps
- Cell phone capped at 5 Mbp
- *all may vary by company/equipment/etc.*



- **Note:** 2/10 presents the flow is 2, out of capacity 10.
- It is not division giving 0.2
- It is not fraction also; can't replace 2/10 by 1/5.

Capacities

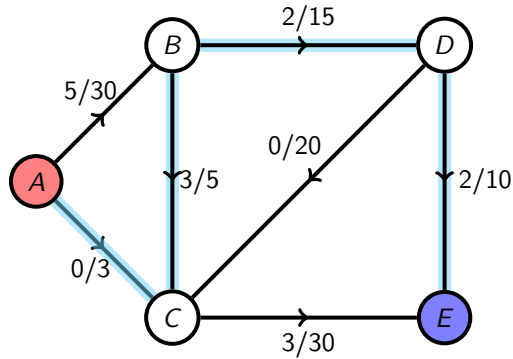
- For each edge:

$$f(x \to y) = \text{ flow on edge } (x \to y)$$
$$c(x \to y) = \text{ capacity on edge } (x \to y)$$

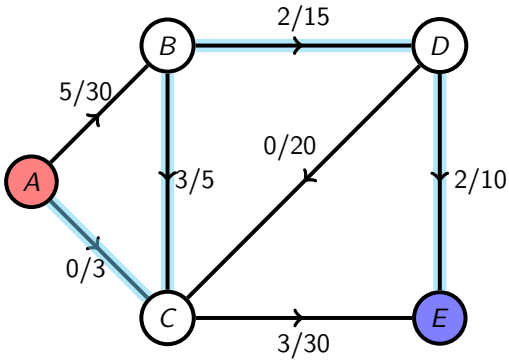
- We require, $0 \le f(x \to y) \le c(x \to y)$ for every edge.
- **Goal:** Find a flow maximizing $v(f)$, subject to the capabilities.
- **Strategy:** Keep finding augmented paths from source to sink, on which we can add flow. Two paths so far are examples.

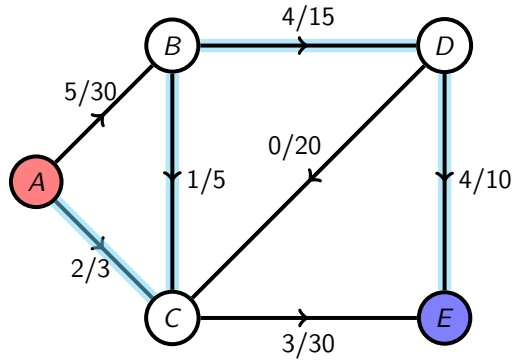
Paths with backwards edges

- We allow paths with backward edges
- Consider path ACBDE; where CB is traversed backward; other nodes are forwards.
- We will send 2 units along the cyan path



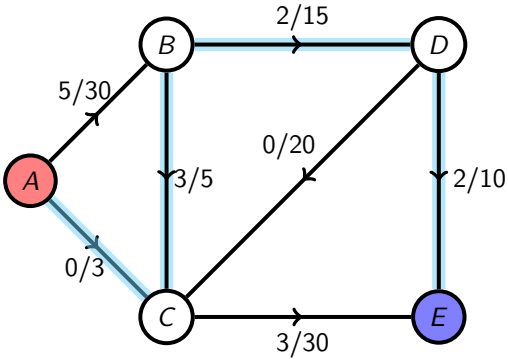
Paths with backwards edges

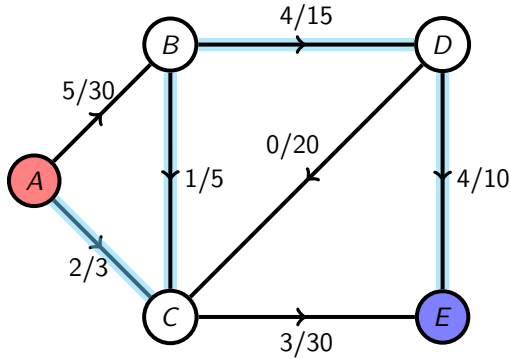




- Add 2 to the flow on each forward edge
- Subtract 2 from the flow on each reverse edge
- This keeps the internal nodes balanced and increases $v(f)$ by 2.

Paths with backwards edges



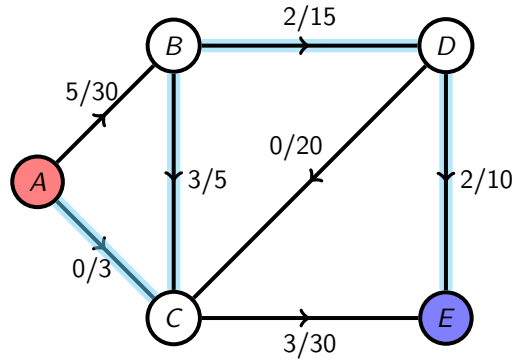


- Nothing actually goes backward! Instead, some paths are rerouted and spliced together differently.
- 2 units on $A \to B \to C \to E$ are redirected to $A \to B \to D \to E$. So, BC and CE go down by 2, while BD and DE go up by 2.
- The 2 units just lost on CE are replaced by 2 new units from AC.

Residual Capacities

- The **residual capacity of an edge** is

$$c_f(u \to v) = \begin{cases} c(u \to v) - f(u \to v), & \text{if } (u \to v) \in \mathcal{E} \\ f(v \to u), & \text{if } (u \to v) \in \mathcal{E} \\ 0, & \text{if } (u \to v) \notin \mathcal{E} \end{cases}$$



- Here $c_f(A \to C) = 3, c_f(C \to B) = 3, c_f(B \to D) = 13, c_f(D \to E) = 8$, etc.
- Residual capacities along ACBDE:
  $c_f(A \to C) = 3, c_f(C \to B) = 3, c_f(B \to D) = 13, c_f(D \to E) = 8.$

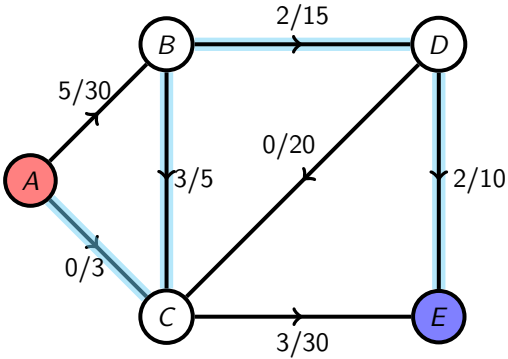
Residual Capacities

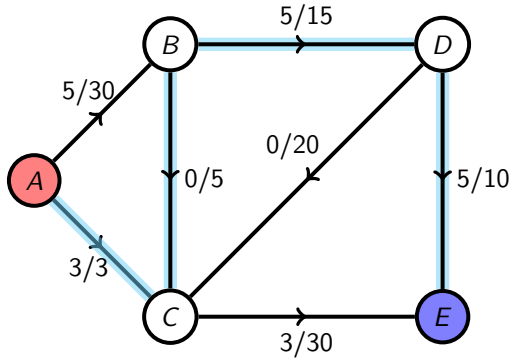
- Let ϵ be the minimum of these:

$$\epsilon = \min\left\{c_f(u \to v) : \text{ along path such that} (u \to v) \in \mathcal{E}\right\}$$

- Here, in the last example, we have $\epsilon = \min\{3, 3, 13, 8\} = 3$
- This is the maximum that we can increase the flow along the path
- i.e. we have to pick a number that works for all edges on the path.
- If a path has $\epsilon > 0$, it is called augmented path.

Residual capacities





- Use this augmenting path to increase the flow by $\epsilon = 3$
- Along the path, add ϵ to the flow for each forwards edge, and subtract it for each reverse edge.

Ford-Fulkerson Algorithm

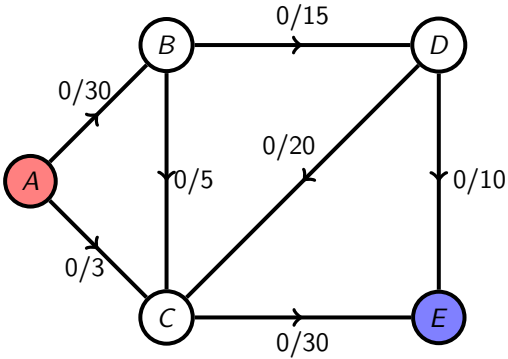
Ford-Fulkerson Algorithm

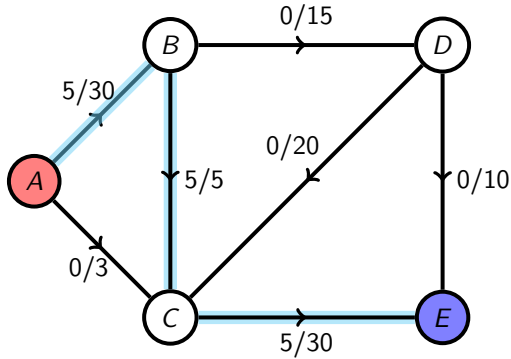
Given a digraph G, source s, sink t, and edge capacities $c(x, y)$

1. Initialize $f(x \to y) = 0$ for all vertices $x \to y$
2. While there is any augmented path $(s \to t)$:
   - 2.1 Increase f by ϵ along forwards edges of the path
   - 2.2 Decrease f by ϵ along reverse edges.

- The above theorem, don't say the specific way to find paths.
-

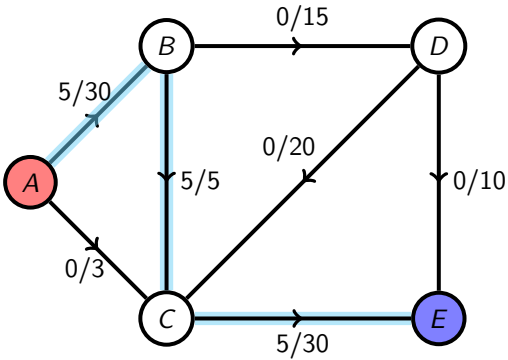
Ford-Fulkerson Algorithm

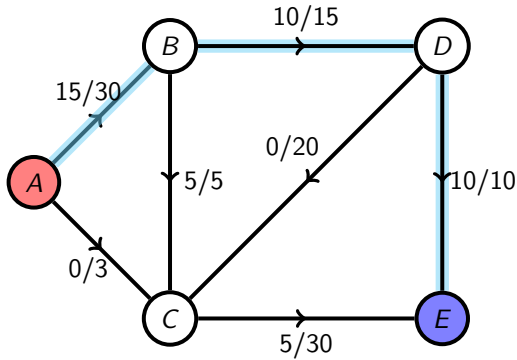




- Start over with the same graph
- We send $\epsilon = 5$ along $ABCE$

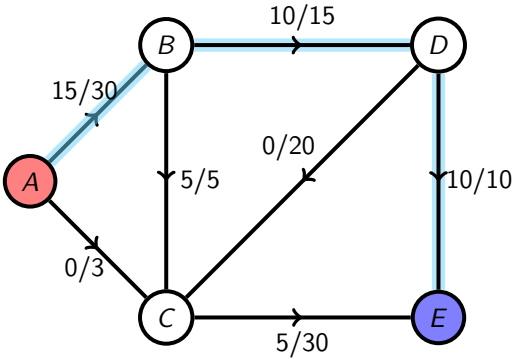
Ford-Fulkerson Algorithm

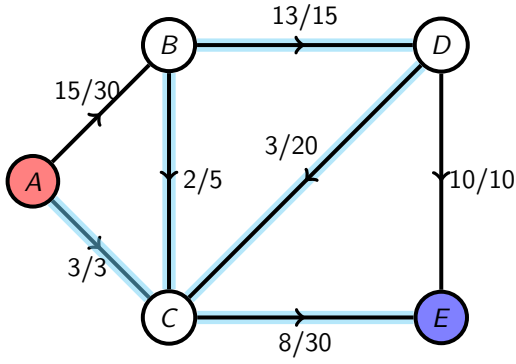




- Send $\epsilon = 10$ on ABDE

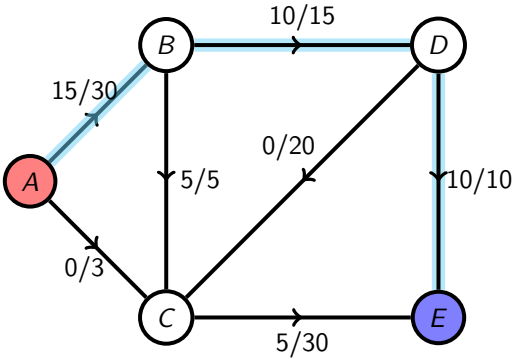
Ford-Fulkerson Algorithm

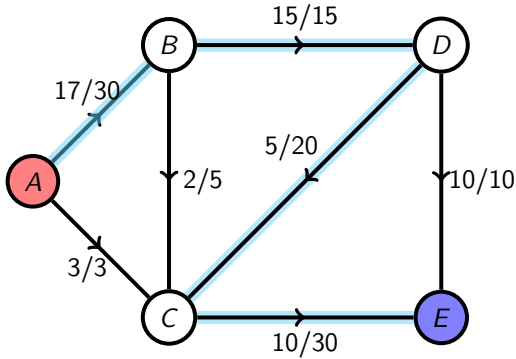




- Send $\epsilon = 3$ on ACBDCE

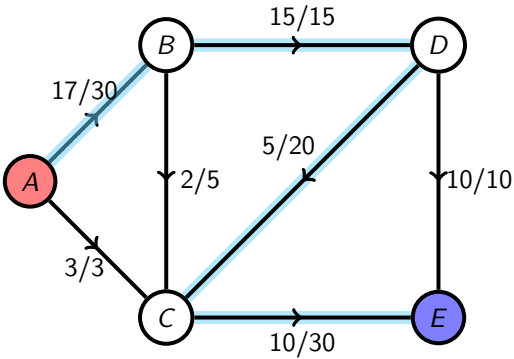
Ford-Fulkerson Algorithm

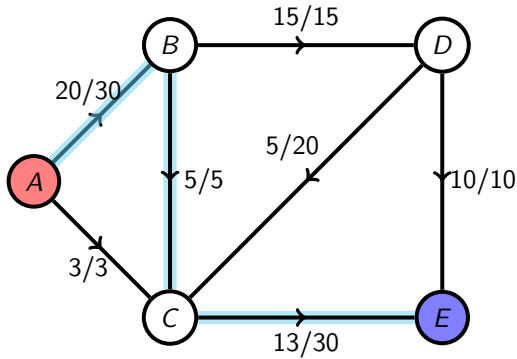




- Send $\epsilon = 2$ on ABDCE

Ford-Fulkerson Algorithm

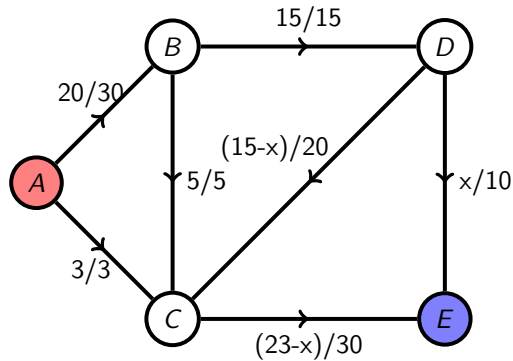




- Send $\epsilon = 3$ on ABCE
- We are stuck!
- We got $v(f) = |f| = 23$
- Could a different choice of paths have lead to a higher flow value?

Ford-Fulkerson Algorithm

- We will see that $v(f) = 23$ is the maximum possible; however there are multiple flows achieving this maximum
- For any real number $x \in [0, 10]$, gives a flow with $v(f) = 23$
- If you use integer capacities, the FF algorithm (alg. based on it) only produces flows with integer values on each edge, even thogh there may be non-integer solutions.

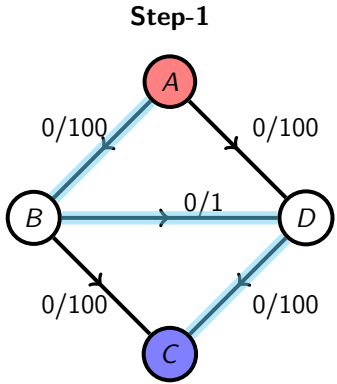


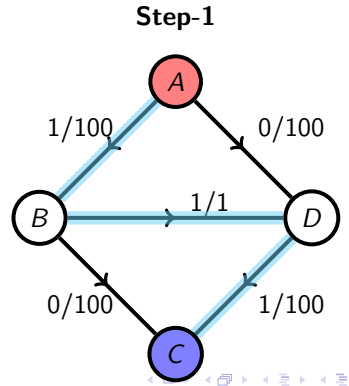
  - With integer, FF alg. must terminate.
  - With real number capacities, there examples where certain choices of paths lead to an infinite loop that may note even converge to the correct answer; yet, if paths are chosen carefully, it will terminate.

Bad news:

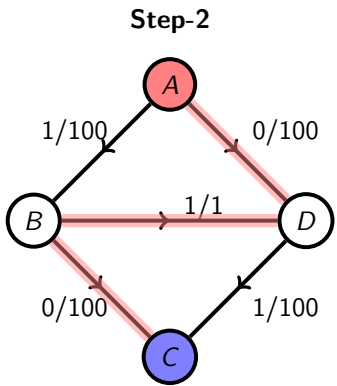
Even when edge capacities are integers, number of augmenting Bad news. Even when edge capacities are integers, number of augmenting paths could be equal to the value of the maxflow.

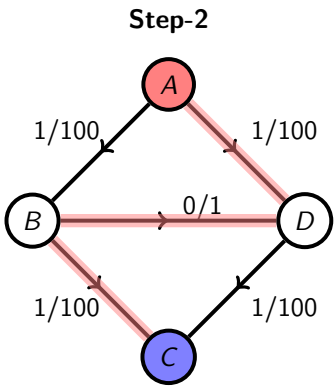
Cyan path is ABDC and Red path is ADBC



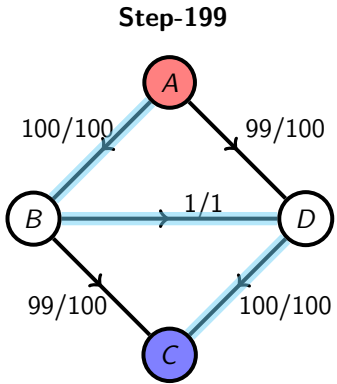


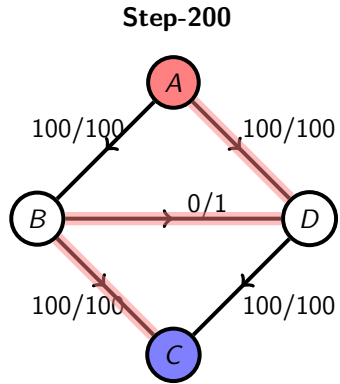
Bad news:


Step-2
A
B
C
D
1/100
0/100
1/1
0/100
1/100


Step-2
A
B
C
D
1/100
1/100
0/1
1/100
1/100

Bad news:


Step-199
A
100/100
99/100
B
1/1
D
99/100
100/100
C


Step-200
A
100/100
100/100
B
0/1
D
100/100
100/100
C

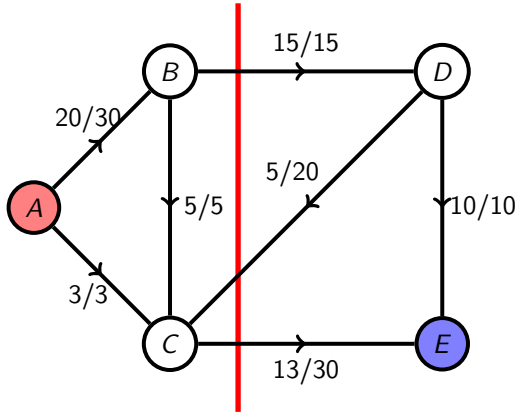
Cuts

- A cut is a partition (S, T) of the vertices with $s \in S, t \in T$:

$$S \cup T = \mathcal{V} \text{ and } S \cap T = \phi.$$

- So, $T = S \backslash \mathcal{V} = \bar{S}$
- The capacity of a cut is the sum of capacities of edges $x \to y$ with $x \in S, y \in T$:

$$c(S, T) = \sum_{x \in S} \sum_{y \in T} c(x \to y).$$



- Here red line is cut with $S = \{A, B, C\}$ and $T = \{D, E\}$.
- Edges crossing from $S \to T$ are BD and CE
- The capacity of this cut is $c(S, T) = c(B \to D) + c(C \to E) = 15 + 30 = 45$
- This sum does not use edges from $T \to S$: DC; nor edges with both ends is S (AB, AC, BC) or both in T (DE).

Cuts

- In principle, with $n \geq 2$ vertices, there are $2^n - 2$ possible cuts.
- In practice, for line cuts we actually use, you can often draw a line/curve separating S on one side and T on the other.
  - The side with the source s is S and the side with the sink t is T.

Lemma

For any flow f and any cut (S, T):

$$|f| = \sum_{x \in S} \sum_{y \in T} f(x \to y) - \sum_{w \in T} \sum_{x \in S} f(w \to x)$$

Proof.

$$\begin{aligned} |f| &= \delta f(s) && \text{By definition} \\ &= \sum_{x \in S} \delta f(x) && \text{Conservation constraint} \\ &= \sum_{x \in S} \Big[\sum_{y} f(x \to y) - \sum_{w} f(w \to x) \Big] && \text{By defn of } \delta \end{aligned}$$

Cuts

$$
\begin{aligned}
|f| &= \sum_{x \in S} \Big[\sum_{y} f(x \to y) - \sum_{w} f(w \to x) \Big] && \text{By defn of } \delta \\
&= \sum_{x \in S} \sum_{y} f(x \to y) - \sum_{x \in S} \sum_{w} f(w \to x) \\
&= \sum_{x \in S} \sum_{y \notin S} f(x \to y) - \sum_{x \in S} \sum_{w \notin S} f(w \to x) && \text{Removing edges from } S \text{ to } S \\
&= \sum_{x \in S} \sum_{y \in T} f(x \to y) - \sum_{x \in S} \sum_{w \in T} f(w \to x) && \text{Definition of cut}
\end{aligned}
$$

Computing value of a flow using a cut

- Here cut: $S = \{A, B, C\}$ and $T = \{D, E\}$

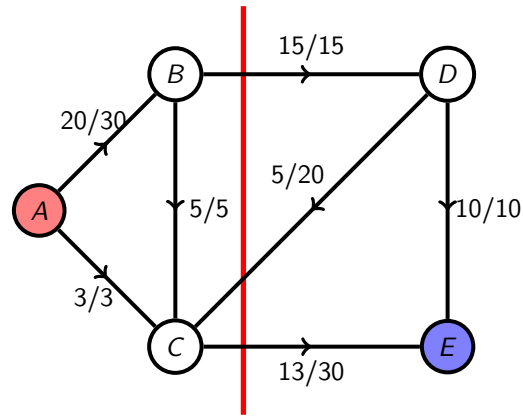
$$|f| = \sum_{x \in S} \Big[\sum_{y} f(x \to y) - \sum_{w} f(w \to x) \Big]$$

At $x = A$: $\quad |f|_A = f(A,B) + f(A,C) - 0$

At $x = B$: $\quad |f|_B = f(B,C) + f(B,D) - f(A,B)$

At $x = C$: $\quad |f|_C = f(C,E) - f(A,C) - f(B,C) - f(D,C)$

$$\begin{aligned} |f| &= |f|_A + |f|_B + |f|_C \\ &= f(B,D) + f(C,E) - f(D,C) \\ &= \sum_{x \in S} \sum_{y \in T} f(x \to y) - \sum_{x \in S} \sum_{w \in T} f(w \to x) \end{aligned}$$



Comparing formulas for value of a flow

$$
\begin{aligned}
|f| &= \sum_{y} f(x \to y) - \sum_{w} f(w \to x) && \text{(using source)} \\
&= \sum_{w} f(w \to x) - \sum_{y} f(x \to y) && \text{(using sink)} \\
&= \sum_{x \in S} \sum_{y \in T} f(x \to y) - \sum_{x \in S} \sum_{w \in T} f(w \to x) && \text{(using cut)}
\end{aligned}
$$

- First two formulas can be obtained by using third one:
  - **Source:** $S = \{s\}$, and $T = \mathcal{V} \backslash S$
  - **Sink:** $S = \mathcal{V} \backslash S$, and $T = \{s\}$.

Relation between $|f|$ and c

Lemma (Maxflow - mincut)

Let f be any feasible (s,t)-flow, and let (S,T) be any (s,t)-cut. The value of f is at most the capacity of (S,T). Moreover, $|f| = c(S,T)$ if and only if f saturates every edge from S to T and avoids every edge from T to S.

Proof. we know, finding value of a flow by using a cut (S,T)

$$\begin{aligned} |f| &= \sum_{x\in S}\sum_{y\in T} f(x\to y) - \sum_{x\in S}\sum_{w\in T} f(w\to x) \\ &\le \sum_{x\in S}\sum_{y\in T} f(x\to y) \qquad (\text{since } f(w\to x)\ge 0) \\ &\le \sum_{x\in S}\sum_{y\in T} c(x\to y) \qquad (\text{since } f(x\to y)\le c(x\to y)) \end{aligned}$$

- Maximize the left-hand side over all flows, and minimize the right-hand side over all cuts:

$$\max\{|f| : \text{ flows } f\} \le \min\{: \text{ cuts } (S,T)\}$$

Maxflow - mincut

- We will show the last inequality is equal. For this:
- Set $S = \{s\}$
- Iteratively add vertices connected to S with residual capacity > 0, i.e.,
  - **Forwards:** If $f(x \to y) < c(x \to y)$ for some $x \in S$ and $y \notin S$, then add y to S
  - **Backwards:** $f(w \to x) > 0$ for some $w \notin S$ and $x \in S$, then add w to S
  - Keep going till we can't add any more vertices this way.
- After constructing this, check if $t \in S$:
  - If $t \in S$, the there is an augmented path from s to t, we could increase the flow. Thus it is not maximum flow.
  - If $t \notin S$, the FF algorithm terminates. Is it maxflow?
- We will us the cut (S, T), where $T = \mathcal{V} \backslash S$.
- No more vertices can be added to S:
  - **FWD:** If $(x \to y) \in \mathcal{E}$ with $x \in S$ and $x \in T$, then $f(x \to y) = c(x \to y)$.
  - **REV:** If $(w \to x) \in \mathcal{E}$ with $x \in S$ and $w \in T$, then $f(x \to y) = 0$.

  Otherwise, we could have added more vertices to S.

Maxflow-mincut

- Flow value:

$$\begin{aligned} |f| &= \sum_{x \in S} \sum_{y \in T} f(x \to y) - \sum_{x \in S} \sum_{w \in T} f(w \to x) \\ &= \sum_{x \in S} \sum_{y \in T} c(x \to y) - 0 \\ &= c(S, T) \end{aligned}$$

- There is no gap between f and $c(S, T)$.
- So, this is simultaneously the maximum flow and the minimum cut.

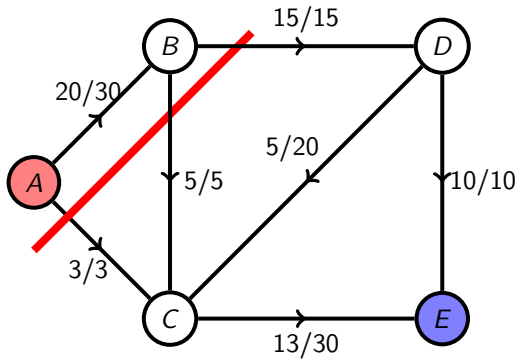
Theorem (Maxflow-Mincut Theorem)

The maximum flow value equals to the minimum cut capacity.

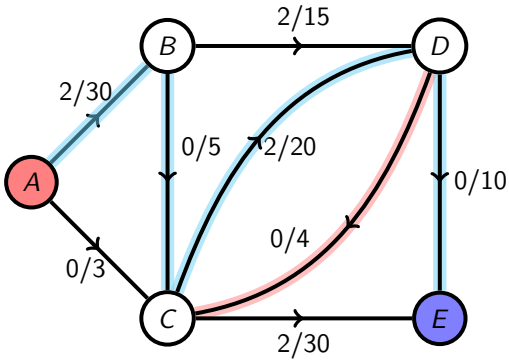
Maxflow-Mincut Theorem

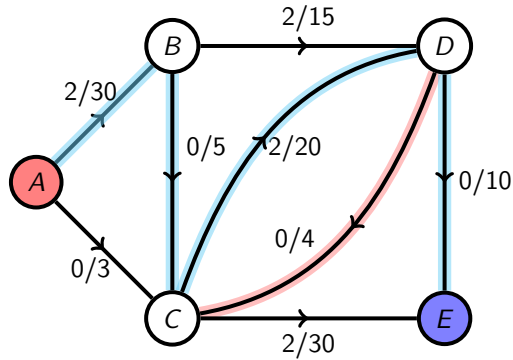
- Flow we found with Ford-Fulkerson has value $|f| = 23$
- Cut $S = \{A, B\}$ and $T = \{D, E, F\}$ has capacity

$$\begin{aligned} c(S,T) &= c(B,D) + c(B,C) + c(A,C) \\ &= 15 + 5 + 3 = 23 \end{aligned}$$



Allowing edges both ways

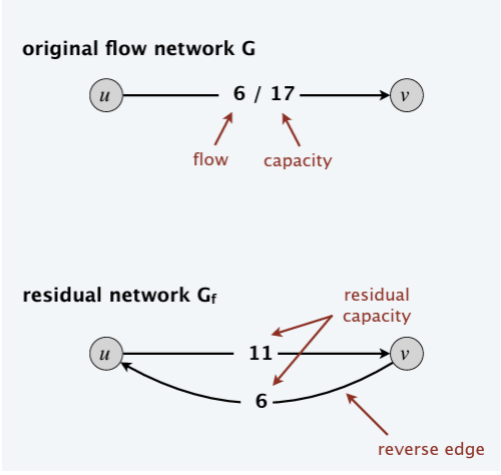




- For path $ABCDE$ there are edges both ways $C \to D$ and $D \to C$
- Adding 5 units of flow here requires handling CD specially.
- Decrease DC (red) by 2 and increase CD (cyan) by 3.
- **Residual capacity formula for allowing edges both ways:**

$$c_f(u \to v) = \underbrace{c(u \to v) - f(u \to v)}_{\text{remaining fwd capacity}} + \underbrace{f(v \to u)}_{\text{amount cancelable in reverse}}$$

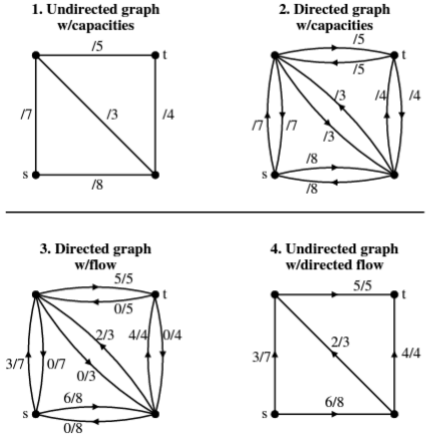
Residual Network



Allowing edges both ways

- One example can be seen at https://www.javatpoint.com/daa-ford-fulkerson-algorithm.

In case of directed graph:



1. **Input:** Undirected graph; source and sink; edge capacities.
2. Each edge → two directed edges.
3. Find directed flow.
4. **Output:** Directed flow on original undirected edges.