Given a directed graph G, with positive capacities on the edges, also given source & target node; maximize the flow.

\[
\text{max flow out of } S = \frac{100 + 10 + 2}{\text{sum of capacities}} = 112
\]

\[
\text{max flow into } T = \frac{10 + 70 + 10}{\text{sum of capacities}} = 90
\]

\[
\text{max flow} \leq \min\{40, 112\} = 40
\]

\[
\text{max flow} \leq \min\left\{\text{capacity of}(S, T) \text{ cuts}\right\}
\]

\[
V = S \cup T
\]
\[ V = S \cup T \]
\[ T = V - S \]
\[ x \in S \]
\[ t \in T \]

\[ (s, t) \text{ cuts} \]

Capacity of an \((s, t)\) cut is:
\[
\sum_{x \in S} \sum_{y \notin T} c(x, y)
\]

Max flow \( \leq \) min \( S-T \) cut in \( G \)

---

**Ford-Fulkerson Max Flow Algorithm**

1) Compute the residual graph
2) Find an \( S-T \) path (any path from \( s \) to \( t \))
3) Add this path to existing flow, using the min capacity edge
Add this path to existing flow, using the min capacity edge.

\[ \text{Flow} = \emptyset \]

Residual

Residual

No more \( s-t \) path. Stop!

Value (flow) = \( \sum \text{flow}(u,v) \)

\( u \neq x \)

\[ S = \{ s \} \]

\[ T = \{ t \} \]
\[ x = 90 + 10 = 100 \leq \text{max flow} \]

1) \( \text{max-flow} \leq \text{min ST cut} \)

In fact, \( \text{max-flow} = \text{min ST cut} \)

Worst-case time for Ford-Fulkerson:

1) \( \text{time to compute the residual graph } G_f \)  
\[ 2m \rightarrow O(\mid E \mid) \text{ or } O(m) \]

2) Finding \( s-t \) path in \( G_f \)

- Do a BFS from \( s \) and check if \( t \) can be reached

\[ O(n + m) \quad n = \mid V \mid \quad m = \mid E \mid \]

Each iteration: \( O(m + n) \)

How many steps in the worst case?
Every time the flow value increases by at least 1,

\[ \Rightarrow \text{total \# of iterations} \leq \text{max capacity edge in } G \]

\[ = C \]

**Total time:** \( C \cdot (m+n) \)

\[ \text{max capacity value} \]

**Pseudo-polynomial!**

\[ PF(G, C) \]

\( b \) bits to represent \( C \)

\[ \Rightarrow \text{value } C = O(2^b) \]

\( O(2^b \cdot (n+m)) \)

Exponential in input size.

**Correction of Ford-Fulkerson (FF)**

1) \( G^f \) does not violate any capacities
2) Any \( s-t \) path we find, respect the conservation of flow
3) Value of the current flow \( l \)
3) Value of the current flow keeps on increasing with each iteration.

$log \text{ will eventually stop with some }$ 

"local" max flow

4) "local" max flow = "global" max flow

Proof: In the final residual graph $G$, there is no $st$ path!

1) all edge $(x,y)$ such that $x \in S$ and $y \in T$

are at full capacity
are at full capacity

\[ \implies \text{flow on that edge} = \text{capacity of that edge in } G \]

2) If there exist any "forward" edge from \( T \) to \( S \) then the flow on that edge is 0

\[ (y, x) \in G, \quad \text{Capacity} (y, x) \]

\[ \implies \text{reservs} \exists \text{edge} (x, y) \in G^f \]

\[ \iff y \text{ can be reached from } T \]

\[ \implies \text{ all "forward" edges are flowing at full capacity} \]

& there is no base flow from \( T + S \)

\[ F F \rightarrow \text{Completion } G^f \]

\[ \text{All nodes reachable from } y \]

\[ (S, T) \] is a certificate!

\[ \max \text{flow across } (S, T) = \] min or possible in \( G \)
ST cut value $= 7$

FF is pseudo-polynomial or $O(b(m+n))$ time

$b$ is # of bits to represent max capacity.

1) try to find "heavy" paths

where the min capacity edge is
\[ \Delta \text{- phase FF} \]

\[ \text{flow} = \phi \]

\[ \Delta = \text{least power of } 2 \text{ such that we do not exceed the max capacity out of } \{x, \ldots, C\} \]

\[ \text{vertex } G \rightarrow G(\Delta) \]

\[ \text{run FF on } G(\Delta) \]

\[ \Delta = \Delta / 2 \]

\[ \text{try to restrict the graph to edge with or less } \Delta \text{ capacity.} \]

\[ \Rightarrow \text{any flow value will increase by or less } \Delta \text{ in each step.} \]

\[ \Delta = \frac{512}{2} \]

\[ \text{max flow} \]

\[ G(\Delta) = \]

\[ \text{max flow} \]

\[ \text{vertex } G \rightarrow G(\Delta) \]

\[ \text{run FF on } G(\Delta) \]

\[ \Delta = \Delta / 2 \]

\[ \text{try to restrict the graph to edge with or less } \Delta \text{ capacity.} \]

\[ \Rightarrow \text{any flow value will increase by or less } \Delta \text{ in each step.} \]
At most \( \log C \) \( \Delta \)-phase \( \checkmark \)

\[
\begin{align*}
\text{FF on } G(\Delta) & \rightarrow \text{how many steps due regular FF do on } G(\Delta) \\
\text{each step } O(m+n) & \rightarrow \text{FF will never exceed } 2m \text{ steps}
\end{align*}
\]

\[
\text{total time: }
\log C \left[ \begin{array}{c}
2m(m+n) \\
\Delta \text{-phase} \\
\Rightarrow G(\Delta)
\end{array} \right] 
\]

\[
O \left( b \cdot \frac{2^2 + mn}{m+n} \right) \quad b = \log C
\]

truly polynomial.

Maximum Bipartite Matching

Bipartite graph: \( X \quad Y \)
Bipartite graph
\[ V = X \cup Y \]
\[ X \cap Y = \emptyset \]
\[ e(a, b) \Rightarrow a \in X, \quad b \in Y \]

Matching
\[ 1 \to 1 \]
mapping from a subset of \( X \)
\[ x \to y \]
subset of \( Y \)
\[ x_1 \to y_1 \]
\[ x_3 \to y_3 \]

Maximum size matching
\[ \# \text{of matched pairs} \]

\[ G = \left[ \text{Bipartite graph} + \{ a, x_i \} + \{ y_i, t \} \right] \]

Find a max flow in \( G \) = max matching in bipartite graph