Network Flows

\[ G = (V, E) \]

\[ f \leq c(A, B) \]

\[ f \leq 2 \]

\[ \max \text{ flow} = c(A', B') \]

\[ b' = \{ b \}, h' = \{ x, y, z \} \]

\[ \text{capacities are integers} \]

**Goal:** maximize "flow" from \( x \) to \( t \)

1) for each edge \( e \), find a integer

\[ c(e) \geq f(e) \geq 0 \]

2) flow has to be conserved at each node

\[ f^\text{in} (v) = \sum_{e = (x, v)} f(e) \]

\[ f^\text{out} (v) = \sum_{e = (v, u)} f(e) \]

\[ \text{incoming flow at } v \]

\[ \text{outgoing flow at } v \]
\[ f^{\text{out}}(v) = \sum_{e = (v, y)} f(e) \]

\[ f^{\text{in}}(v) = f^{\text{out}}(v) \quad \forall v \in V \text{ except for } s \text{ and } t \]

\[ f^{\text{out}}(s) = f^{\text{in}}(t) \]

3. Value of the flow:
\[ f^{\text{out}}(s) = f^{\text{in}}(t) \leftarrow \text{maximizing task} \]

Max-flow value = min-cut in the graph

If a flow matches the capacity of the minimum cut, then the flow is maximum.

\[ V, \text{ a cut is } A, B = V - A \]

such that \( x \in A, t \in B \)

Capacity of cut
\[ c(A, B) = \sum_{e = (x, y)} c(x, y) \]

\( x \in A \quad y \in B \)
\[ f_{\text{flow}} \leq c(A, B) \quad \forall \text{ cuts} \]

\[ \Rightarrow \quad f_{\text{flow}} \leq \text{minimum cut in } G \]

\[ \text{Max-flow in } G = \text{min-cut in } G \]

\[ f = 20 \]

Keep track of residual capacities:
\[ c(e) - f(e) \]

\[ e = (y, u) \quad c(e) = 20 \quad f(e) = 20 \]

Negative path

Capacity of s-t path is bounded by the least capacity of any edge on that path

Residual (temp) graph
Stopping criteria:
If no $A$-$T$ path exists in the residual graph

Ford-Fulkerson Algorithm

1) Find $A$-$T$ paths in the residual graph (also called augmenting paths)

Also called "virtual" edge (includes a new type of backward edge)

$G_f$ : residual graph
$e = (x,y)$
\[ f(e) > 0 \]
Ford - Fulkerson

\[ f(e) = 0 \quad \forall e \in E \]

While there is an s-t path in \( G_f \) (residual graph)

\[ \# \exists \Gamma \left( \text{P: s-t path} \right) \quad \text{Residual} \]
# of Iterations

\[
\text{while } \quad \begin{align*}
P &: \text{s-t path} \\
c(P) &: \text{capacity of } P \\
(\text{in each capacity edge in } P) \\
\text{Augment the flow using } P \& c(P) \\
\text{Update } G
\end{align*}
\]

\[G^f:\quad \text{twice as many edges } \quad 2|E| \]

\[\Rightarrow \text{run a BFS on } G^f \text{ to find some } s-t \text{ path} \\
O(1V + 2|E|) \]

\[= O(1V + |E|)
\]

Each iteration therefore takes \(O(1V + |E|)\) time.

How many iterations?

\[\Rightarrow \text{any edge in } G^f \text{ has to have capacity } \geq 1
\]

Each iteration will increase the flow by at least 1 unit.

\[
\text{any flow } f \leq \sum \frac{c(\gamma, x)}{e = (\gamma, x)} = C
\]
Any flow \( + \leq \sum_{e=(y,x)} c_x = \sum_{e=(y,x)} c_x \)

In the worst case \( O(C) \) iterations

\[ O(C) \times O(|V|+|E|) \]

= \( O(C \cdot (|V|+|E|)) \) \( \leftarrow \) total cost

\( \leftarrow \) pseudo-polynomial algorithm

because it depends on the value \( C \) by \( C \) bits to store true value

\[ O \left( 2^m \cdot (|V|+|E|) \right) \text{ where } m = \log_2 C \]

\[ \text{Exponential in size } \] (size of \( \log \) bits) to store \( C \)

\( \Delta \) scaling \( \widehat{FF} \)

1) try high capacity path first
\[ \Delta = \text{largest power of 2 that is less than } C \]
\[ \Delta = 2^m \leq C \]

\[ \text{while } \Delta \geq 1 \]
\[ \text{G}(\Delta): \text{ keep only those edges that have capacity at least } \Delta \]
\[ \text{run FF on } G(\Delta) \]
\[ \Delta = \Delta/2 \]

\[ \frac{O(\log C)}{\text{outer loop}} \times \frac{2|E|}{\text{while loop}} \times \frac{|V| + |E|}{\text{cost per } FF \text{ iteration on } G(\Delta)} \]

\[ O\left( \log C \left( \frac{|E|^2 + |E| \cdot |V|}{\log C} \right) \right) \]
\[ = O\left( \log C \cdot |E|^2 \right) \]
\[ = O\left( m \cdot |E|^2 \right) \]
\[ = \frac{1}{\log_2 C} \]

Note: the convention

\[ G(\Delta) \downarrow \Delta \uparrow \Delta/2 \]

\[ G(\Delta) \]

\[ G(\Delta/2) \]
C = \sum_{e} \psi(x) = 1050 \quad \Delta = 512

G(\Delta) =

G(\Delta)_{f}
Next $\Delta = \Delta/2 = \frac{812}{2} = 257$, and so on.