1. Given a flow network \( N = (G = (V, E), c, s, t) \), let \( f_1 \) and \( f_2 \) be two flows in \( N \) (ie, satisfying the three flow properties wrt \( N \)). The flow sum \( f_1 + f_2 \) is the function from \( V \times V \) to \( \mathbb{R} \) defined by:

\[
(f_1 + f_2)(u, v) = f_1(u, v) + f_2(u, v)
\]

for all \( u, v \in V \).

Which of the three flow properties (wrt \( N \)) will \( f_1 + f_2 \) satisfy, and which will it violate?

**Answer:** The three properties are *capacity constraints*, *skew-symmetry*, and *flow conservation*.

Capacity constraints: \( f_1 + f_2 \) might violate the capacity constraints. As an example, consider the network of question 2. Let \( f_1 \) be the flow shown in question 2. Let \( f_2 \) be the flow that ships 4 units along the path \( s \to x \to y \to t \). Then if we add these flows directly as prescribed in this question, we will (for example) define

\[
(f_1 + f_2)(y, t) = f_1(y, t) + f_2(y, t) = 4 + 4 = 8.
\]

This certainly breaks the capacity constraint for \((y, t)\) which is 4.

Skew-symmetry: \( f_1 + f_2 \) will satisfy skew-symmetry. We know \( f_1 \) and \( f_2 \) individually satisfy skew-symmetry, because they are flows. Therefore for any \((u, v)\), we have

\[
(f_1 + f_2)(u, v) = f_1(u, v) + f_2(u, v) = -f_1(v, u) - f_2(v, u) = -(f_1 + f_2)(v, u),
\]

as required (using the defn of \( f_1 + f_2 \) and the skew-symmetry property for \( f_1, f_2 \)).

Flow conservation: \( f_1 + f_2 \) will satisfy flow conservation. Flow conservation for a flow \( f \) states that for all \( u \in V \setminus \{s, t\} \), we have \( \sum_{v \in V} f(u, v) = 0 \). We know this holds individually for \( f_1, f_2 \). Let \( u \in V \setminus \{s, t\} \). Then we can write

\[
\sum_{v \in V} (f_1 + f_2)(u, v) = \sum_{v \in V} (f_1(u, v) + f_2(u, v)) = \sum_{v \in V} f_1(u, v) + \sum_{v \in V} f_2(u, v) = 0 + 0 = 0.
\]

Hence flow conservation holds for \( f_1 + f_2 \).

**tutors:** Use this as an opportunity to point out the difference between this Q and the case when \( f_2 \) is a flow in the residual network (wrt \( f_1 \)) - in that case everything has been set up for the capacity condition to also hold.
Two questions:

(a) Find a pair of subsets $X, Y \subseteq V$ such that $f(X, Y) = -f(V - X, Y)$.
(b) Find a different pair of subsets $X, Y \subseteq V$ such that $f(X, Y) \neq -f(V - X, Y)$.

Answer: The point of this question is to get thinking about flow between *sets of vertices*, by applying Lemma 3 of Lecture slides 10-11. However, it might be good to think about specific examples of (a), (b) first, before looking at the details of what the pattern is.

What we are asking is: when is it the case that

$$f(X, Y) + f(V - X, Y) = 0?$$

Remember from Lemma 3 (part 3) of slides 10-11 that for any two disjoint sets $X', Y' \subseteq V$, and any other set $Z'$, and any flow $f$, we have $f(X', Z') + f(Y', Z') = f(X' \cup Y', Z')$. Observe that for our question, certainly $X$ and $V - X$ are disjoint sets. Hence by Lemma 3 (3), we know

$$f(X, Y) + f(V - X, Y) = f(X \cup (V - X), Y) = f(V, Y).$$

So we are testing whether $f(V, Y) = 0$ for (a), and whether $f(V, Y) \neq 0$ for (b) - once this is satisfied, $X$ can be anything...

To make $f(V, Y) = 0$, we should either take $Y$ such that $Y \cap \{s, t\} = \emptyset$, or $Y \cap \{s, t\} = \{s, t\}$. This can be seen by repeated application of part (3) of Lemma 3 from slides 10-11. To make $f(V, Y) \neq 0$, we should take $Y$ such that $|Y \cap \{s, t\}| = 1$.

Here are some concrete examples of this behaviour:

(a) As a concrete example, let $Y = \{v, x\}$. $X$ can be *any* set, take $X = \{w\}$ as an example. Then $f(X, Y) = -12 + 4 = -8$. Then $f(V - X, Y) = 11 + 8 - 11 = 8$.
(b) As a concrete example, take $Y = \{s\}$. Take $X = \{w\}$ again. Then we have $f(X, Y) = 0$. We have $f(V - X, Y) = -11 - 8 = -19$. 

2.
3. **Question:** execute the Ford-Fulkerson algorithm (using the Edmonds-Karp heuristic) on the Network below:

![Network Diagram]

**Answer:** If we are using the Edmonds-Karp heuristic, then every time we search for an augmenting path, we must choose a shortest augmenting path.

For our given network, we can see that on the first iteration, the path \( p_1 = s \rightarrow v \rightarrow w \rightarrow t \) is a shortest path. We have \( c(p_1) = 12 \). Hence we define the flow \( f_1 = f_{p_1} \) by

\[
f_1(e) = f_{p_1}(e) = \begin{cases} 
12 & \text{for } e = (s, v), (v, w), (w, t) \\
-12 & \text{for } e = (v, s), (w, v), (t, w) \\
0 & \text{otherwise}
\end{cases}
\]

Pictorially, we have

![Residual Network]

The residual network \( N_{f_1} \) is as follows:

![Residual Network]

We now examine \( N_{f_1} \) to find a shortest augmenting path. We find that \( p_2 = s \rightarrow x \rightarrow y \rightarrow t \) is a shortest augmenting path in \( N_{f_1} \), min capacity 4, see above.... We therefore define a new flow \( f_{p_2} \) such that 4 units are shipped along the edges of the path \( p_2 \), and -4 shipped in the backwards direction of \( p_2 \). Then we define the flow \( f_2 = f_1 + f_{p_2} \). Remember to point out this is possible *only* because \( f_1 \) is a flow in \( N \) and \( f_2 \) is a flow in the *residual* network \( N_{f_1} \). Below is the flow \( f_2 = f_1 + f_{p_2} \) in \( N \).
Below is the residual network $N_{f2}$. If we again try the Edmonds-Karp rule for finding an augmenting path of shortest possible length, we find the path $p_3 = s \rightarrow x \rightarrow y \rightarrow w \rightarrow t$ (this is of length 4, but there are no paths of length 3 or less in $N_{f2}$). The min capacity along the path is 7.

We define a new flow $f_{p3}$ in $N_{f2}$ by shipping 7 units along $p3$. Then we define the flow $f_3$ in $N$ as $f_3 = f_2 + f_{p3}$. The flow looks as follows:

We compute the residual network $N_{f3}$, see below for a picture.

By Ford-Fulkerson’s algorithm, we now try for a (shortest) augmenting path in the $N_{f3}$. However, if we examine $N_{f3}$, we see that there is *no* augmenting path from $s$ to $t$ - the set of vertices accessible from $s$ is now $\{s, v, x, y\}$.

Hence we terminate, returning the flow $f_3$, of value 23.
4. **Question:** A well-known problem in graph theory is the problem of computing a *maximum matching* in a *bipartite graph* $G$. Give an algorithm which shows how to solve this problem in terms of the network flow problem.

**Definitions:**

A (undirected) graph $G = (V, E)$ is *bipartite* if we have $V = L \cup R$ for two disjoint sets $L, R$, such that for every edge $e = (u, v)$ exactly one of the vertices $u, v$ lies in $L$, and the other in $R$.

A *matching* in an (undirected) graph $G$ is a collection $M \subseteq E$ of edges, such that for every vertex $v \in V$, $v$ belongs to at most one edge of $M$.

A *maximum matching* is a matching of maximum cardinality (for a specific graph).

**Answer:**

To solve this question, we will design a network, based on the bipartite graph $G$, where a maximum flow in the network corresponds to a maximum matching in $G$.

Define the vertex set $V'$ for our network $N$ to be $V' = L \cup R \cup \{s, t\}$, where $s, t$ are two new distinguished vertices.

Define the (directed) edge set $E'$ as follows:

$$E' = \{(s, u) : u \in L\} \cup \{(u, v) : u \in L, v \in R, (u, v) \in E\} \cup \{(v, t) : v \in R\}.$$  

notice that the middle set in the union above is just the edge set $E$ of the original graph, with all of these edges now directed from $L$ to $R$.

Define the capacities of the network as follows:

$$
c(s, u) = 1 \quad \text{for every } u \in L
$$

$$
c(u, v) = 1 \quad \text{for every } u \in L, v \in R, (u, v) \in E
$$

$$
c(v, t) = 1 \quad \text{for every } v \in R
$$

I now claim that every flow of value $k$ in $N$ corresponds to a matching of cardinality $k$ in $G$. The max flow = maximum matching follows directly from this.

$\implies$ Suppose $f$ is a flow of value $k$ in $N$. We assume without any loss of generality that $f$ is an integral flow (because all capacities are integers).

Recall that in $N$, the vertex $s$ has $|L|$ neighboring edges $(s, u)$. By definition of the value of a flow, $k = \sum_{u \in V} f(s, u) = \sum_{u \in L} f(s, u)$. Therefore exactly $k$ of the $(s, u)$ edges carry 1 unit of flow each (since no $(s, u)$ edge can carry more than 1).

Moreover by Lemma 11 in Lecture slides 13-14, every $(S, T)$ cut in the network must be carrying flow of value $k$. Hence if we take $S = \{s\} \cup L$, then we see there are exactly $k$ $(u, v)$ edges in the network which carry exactly 1 unit of flow from left to right (since no $(u, v)$ edge can carry more than this).

Define $M = \{(u, v) \in E : f(u, v) = 1 \text{ in } N\}$. Certainly $|M| = k$. I now show that $M$ is a matching. For every $u \in L$, the flow conservation property must hold. For this
network, this means that for every \( u \in L \), we require \((\sum_{v \in R} f(u, v)) + f(u, s) = 0\). Therefore if \( f(s, u) = 0 \), we require \( f(u, v) = 0 \) for every \((u, v) \in E\).

If \( f(s, u) = 1 \) (so \( f(u, s) = -1 \)), we require \( f(u, v) = 1 \) for exactly one \((u, v) \in E\) (using our integer assumption). Hence every \( u \in L \) will appear at most once in \( M \). Hence \( M \) is a matching.

⇐ This is easier. Just explain how the matching of \( G \) gets mapped to \( N \) and check flow conservation.