1. Given a flow network $N = (G) = (V, E, c, s, t)$, let $f_1$ and $f_2$ be two flows in $N$ (i.e., 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 $s \rightarrow x \rightarrow y \rightarrow t$. 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.
2. **Question**: we are given

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 $X', Y' \subset V$, 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 $f(V - X, Y) = -11 - 8 = -19$. 
3. **Question:** execute the Ford-Fulkerson algorithm (*using the Edmonds-Karp heuristic*) on the Network below:

![Network Diagram](image)

**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

![Flow Diagram](image)

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

![Residual Network Diagram](image)

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 $N_{f_2}$. If we again try the Edmonds-Karp rule for finding an augmenting path of shortest possible length, we find the path $p_3 = s \to x \to y \to w \to t$ (this is of length 4, but there are no paths of length 3 or less in $N_{f_2}$). The min capacity along the path is 7.

We define a new flow $f_{p_3}$ in $N_{f_2}$ by shipping 7 units along $p_3$. Then we define the flow $f_3$ in $N$ as $f_3 = f_2 + f_{p_3}$. The flow looks as follows:

We compute the residual network $N_{f_3}$, see below for a picture.

By Ford-Fulkerson’s algorithm, we now try for a (shortest) augmenting path in the $N_{f_3}$. However, if we examine $N_{f_3}$, 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:** This question considers the scenario of *dynamically-evolving* capacities in a network. We start with a fixed network $N = (G, c, s, t)$, and compute a maximum flow $f$ for $N$. However at any time in the future we may receive notification that the capacity of a particular arc in the network is being increased by 1 (or decreased by 1). Two questions:

(a) Suppose that the capacity of a single edge $(u, v) \in E$ is increased by 1. Give an $O(|V| + |E|)$-time algorithm which updates $f$ to obtain a max flow $f'$ for the updated network.

(b) Suppose that the capacity of a single edge $(u, v) \in E$ is decreased by 1. Give an $O(|V| + |E|)$-time algorithm which updates $f$ to obtain a max flow $f'$ for the updated network.

**Answer:** (fairly hard)

(a) Suppose that we have a max flow $f$ for the network $N$, and that we are told that a particular edge $(u, v)$ is having its capacity increased by 1.

First observe that this increase cannot *decrease* the max flow of the network - clearly every flow in the original network is a flow in the modified network. The increase in $(u, v)$’s capacity may or may not *increase* the max flow.

Let $N'$ be the modified network (same as $N$ except $(u, v)$ has capacity $c(u, v) + 1$). Consider the flow $f$ in the network $N'$, and let $N'_f$ be the residual network. I claim that $f$ is a max flow in $N'$ if and only if there is *no* augmenting path in $N'_f$.

Hence we have the following algorithm to update the max flow when the capacity of an edge is increased by 1:

1. Take the original max flow $f$ and compute $N'_f$.
2. Search for an augmenting path in $N'_f$.
3. If we find an augmenting path $p$, return $f + f_p$ as the max flow for the update network. OTHERWISE if we find no augmenting path, return $f$.

The algorithm is correct for the following reasons. First if there is no augmenting path in $N'_f$, then by Corollary 14 of lecture slides 13-14, $f$ is a max flow.

Second, if there *is* an augmenting path $p$ in $N'_f$, then

(i) The capacity $c(p)$ of $p$ in $N'_f$ is 1 (if it was greater than this, $f$ could not have been a max flow for $N$).

(ii) Define $f' = f + f_p$. Then $N'_f$ has *no* augmenting path (same reason as for (i)). Hence in the case where $N'_f$ does have an augmenting path $p$, we know that $f' = f + f_p$ is a max flow of $N$.

This algorithm can be executed in the time it takes to construct the residual network $O(|V| + |E|)$, and the time it takes to find *one* augmenting path ($O(|V| + |E|)$ by breadth-first search). Hence it is $O(|V| + |E|)$ in total.
(b) If we decrease the value of \((u, v)\) by 1 from its original value in \(N\), then the max flow of the updated network \(N'\) can be no greater than the max flow of \(N\) (and possibly less).

If it is the case that the max flow \(f\) of \(N\) satisfies \(f(u, v) < c(u, v)\), then \(f\) will also be a flow (and hence the max flow) in \(N'\) (where \((u, v)\) has capacity \(c(u, v) - 1\)). However, if we had \(f(u, v) = c(u, v)\) in \(N\), then the max flow of \(N'\) might have value strictly less than \(|f|\). Here is our algorithm.

1. If we had \(f(u, v) < c(u, v)\) in \(N\), then \(f\) is a (max) flow in \(N'\). Return \(f\).
2. OTHERWISE (if we had \(f(u, v) = c(u, v)\)).
   (i). Find a simple path \(p1\) in \(Nf\) from \(u\) to \(s\).
   (ii). Find a simple path \(p2\) in \(Nf\) from \(t\) to \(v\).
   (iii). Take the path \(p2, (v, u), p1\) in \(Nf\), and route 1 unit of flow from \(t\) to \(s\) along this path. Adding this to \(f\), we get a new flow \(f'\) of value \(|f| - 1\) in \(N\), with \(f'(u, v) < c(u, v)\).
   Hence \(f'\) is a flow in \(N'\) also.
   (iv) Finally perform a search in \(N'_f\) for an augmenting path.
   (v) If we find an augmenting path \(p\) in \(N'_f\), return the flow \(f' + f_p\) (of value \(|f|\)).
   (vi) Otherwise, return \(f'\).

Note that a residual network only keeps edges with strictly +ve value (nb for (iii)).

The running time of the algorithm is \(O(|V| + |E|)\) (same reason as for (a) except we may do 3 breadth-first searches, in (i), (ii), (iv), here).

5. **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 \(M\) of edges, \(M \subseteq E\), 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:

\[
\begin{align*}
    c(s, u) &= 1 & \text{for every } u \in L \\
    c(u, v) &= 1 & \text{for every } u \in L, v \in R, (u, v) \in E \\
    c(v, t) &= 1 & \text{for every } v \in R
\end{align*}
\]

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.

⇒ 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 $M$. We can use a similar argument to show that every $v \in R$ can 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.

Mary Cryan