Beginning of 09/09/2024

1 Network Optimization

Definition: Flow

Let $G = (V, E)$ be a directed graph[1] with edge capacities $c(e)$ be given. Pick a **source** vertex $v \in V$ and a **sink** $t \in V$. A **$s-t$ flow** is a function $f : E \to \mathbb{R}$ satisfying

- (Conservation) For each node $v \in V \setminus \{s,t\}$,

  $$\sum_{(u,v)} c(u,v) = \sum_{(v,u)} c(v,u),$$

  i.e., for any non-source, non-sink vertex, the flow *into* into it equals the flow leaving it.

- (Capacity constraints) For each edge (u,v), $0 \leqslant f(u,v) \leqslant c(u,v)$.

We define

$$\partial f(v) := \sum_{e \text{ leaving v}} f(e) - \sum_{e \text{ into v}} f(e).$$

(The conservation constraint then says $\partial f(v) = 0$ for all non-source, non-sink vertices.) The **value** of the flow is defined as $|f| = \partial f(s)$, the amount of flow leaving s, which by conservation equals $-\partial f(s)$, the amount of flow into t.

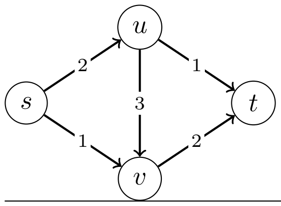
The **max-flow** aims to calculate $\max |f|$, i.e.:

$$\max \partial f(s) \qquad \text{subject to} \qquad \begin{cases} \partial f(v) = 0 & \text{for all } v \neq s,t \\ 0 \leqslant f(e) \leqslant c(e) & \text{for all } e \in E. \end{cases}$$

Naturally, we attempt a greedy-alike algorithm, using viewing a flow as sending various amounts of flow along various $s-t$ paths: while there is a $s-t$ path that we can "stuff" more flow into it, we do so. Put more formally:

Algorithm 0: Max-Flow: DP approach

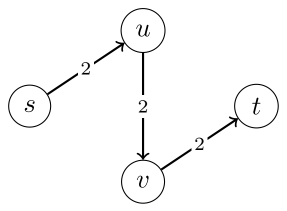
```
// initializations, etc.
while there is a $s-t$ path on which all edges have capacity remaining do
    pick one such path $P$, and put as much flow as possible on it (i.e. min remain capacity)
// return
```



It is clear that this greedy approach returns a valid flow, because nowhere in the algorithm did we break the edge capacity constraints. However, as the example on the left demonstrates, it does not necessarily return the maximum flow. There is only one valid $s \to t$ path: $s \to u \to v \to t$, so we send a 2 units of flow along it. But then there is no path left, even though the following

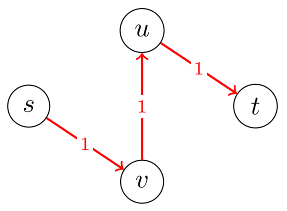
[1] If the graph is directed, (u,v) means the directed edge $u \to v$.

diagrams show that it is possible to construct a flow with $\nu(f) = 3$.



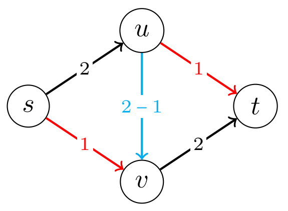
Greedy: $\nu(f) = 2$

+



Augment

=



Max-Flow: $\nu(f) = 3$

The problem in this specific example is that the edges $s \to v$ and $u \to t$ are never used, because greedy considers $u \to v$ as a unidirectional edge. **We can solve this problem by "undoing" flows.** Specifically, we push 1 unit of flow along $s \to v$ because we want to increase $\nu(f)$. But now, v is receiving more flow than its output capacity, so we "undo" 1 unit of flow along $u \to v$. But then u receives 2 units of flow from s and is only currently outputting 1 to v, so we send the other surplus unit of flow to t. In essence, we created an additional, hidden flow through $s \to v \to u \to t$, even though the directed edge $v \to u$ does not exist in the graph.

So far, I haven't been able to find an intuitive explanation of the idea of "undoing flows" using real-life examples. Using the highway example, it simply doesn't make sense if we ask some cars to drive backwards. And I don't think making a one-way road two-way is a proper explanation either, as that essentially makes the directed graph undirected. Any additional feedback would be appreciated. However, one thing for sure is that the resulting flow still satisfies both flow properties.

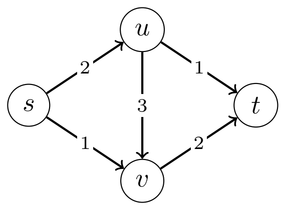
To formalize these, we introduce **residual graphs** and two types of auxiliary edges: **forward** and **backward edges**.

Definition: Residual Graphs

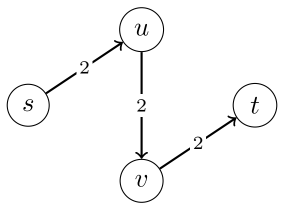
Given a capacity-embedded graph $G = (V, E)$ and an $s - t$ flow f on it, the **residual graph** G_f is defined as follows:

- The nodes are set to be V, identical to G.
- For each edge $e = (u, v)$ of G:
  - If $f(e) < c(e)$, there are $c(e) - f(e)$ "leftover" capacity, so we add a **forward edge** (u, v) with capacity $c(e) - f(e)$ to G_f.
  - If $f(e) > 0$, then there are $f(e)$ amount of flow that we can "undo," so we add a **backward edge** (v, u) [*note the direction*] with capacity $f(e)$ to G_f.

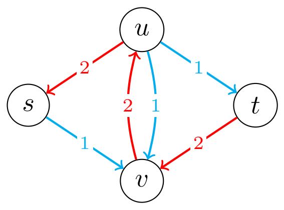
With these definitions, we now look at our simple example again. The residual graph corresponding to the greedy flow f is drawn below, with blue edges representing forward edges, and red edges representing backward edges.



Original graph G



Greedy flow f



Residual graph $G(f)$

Here is an alternate way to explain how we improved our greedy f to Max-Flow. Notice that there is precisely one $s - t$ path remaining in this residual graph $s \to v \to u \to t$, and we see that this path can transmit up to 1 unit of flow. This is precisely how we updated the original graph G: we **augmented** 1 unit of flow along this path found in the residual graph $G(f)$. Specifically, we pay attention to the unidirectional $u \to v$ path in the original graph G: the augmentation process essentially un-sends 1 unit of flow from $u \to v$, since the $v \to u$ path in the residual graph $G(f)$ is an backward edge. Therefore, **when augmenting G using $G(f)$, we need to pay attention to whether each edge is a forward or backward edge in $G(f)$**.