18 Negative Cycles, Network Flow

- Finding negative cycles
- Network flow

18.1 Negative Cycles

Bellman-Ford will find a negative cycle if one exists, provided

- there is a path from the cycle to t
- it runs long enough to see the cycle.

Lemma 18.1
For any digraph G, there exists a digraph G' with one more vertex than G such that G' has a path from a negative cycle to g iff G has a negative cycle.

Lemma 18.2
A digraph has no negative-weight cycles iff $\text{opt}(n-1, v) = \text{opt}(n, v)$ for all $v \in V$.

Proof. If there are no negative-weight cycles, this follows from the correctness of Bellman-Ford.

Conversely, if $\text{opt}(n-1, v) = \text{opt}(n, v)$ for all $v \in V$, by induction, we have $\text{opt}(n-1, v) = \text{opt}(i, v)$ for all $v \in V$ and all $i \geq n-1$.
However, if there were a negative cycle, then $\text{opt}(i, v)$ could become arbitrarily small. So, there is no negative cycle. □

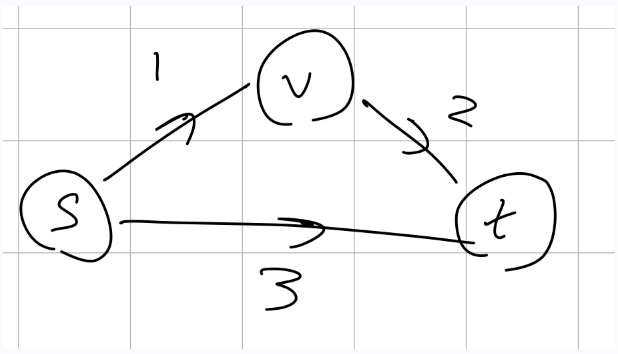
Essentially, we run the bellman ford algorithm for an additional iteration, and compare the last two rows of the opt table.

18.2 Network Flow

Definition 18.3
A **flow network** is a digraph $G = (V, E)$ with

- A **source** vertex $s \in V$ with $\text{indeg}(s) = 0$
- A **sink** vertex $t \in V$ with $\text{outdeg}(t) = 0$
- a capacity $c_e \geq 0$ for all $e \in E$, we assume all c_e's are integers.

Example 18.4
An example of a flow network:



Definition 18.5

A **flow** is a function $f : E \to \mathbb{R}^+$ satisfying

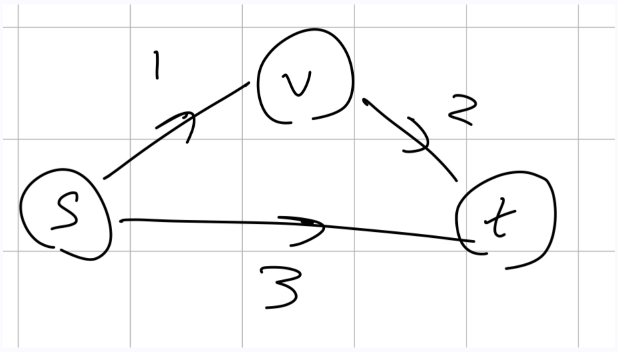
1. Capacity condtions: $0 \leq f(e) \leq c_e$ for all $e \in E$
2. Conservation conditions: $f^{in}(v) = f^{out}(v)$ for all $v \in V \setminus \{s, t\}$ where

$$f^{in}(v) = \sum_{e \text{ into } v} f(e)$$

$$f^{out}(v) = \sum_{e \text{ out of } v} f(e)$$

Example 18.6

From the previous example, we had



To specify a flow, we specify the flow one very edge. The flow on the edge with weight 1 can be at most 1, and so by the conservation condition, the flow on the edge with weight 2 can be at most 1 as well, even though the capacity is higher.

The flow on the edge with weight 3 can be anything not exceeding 3.

Definition 18.7

The **value** of a flow is $v(f) = f^{out}(s)$.

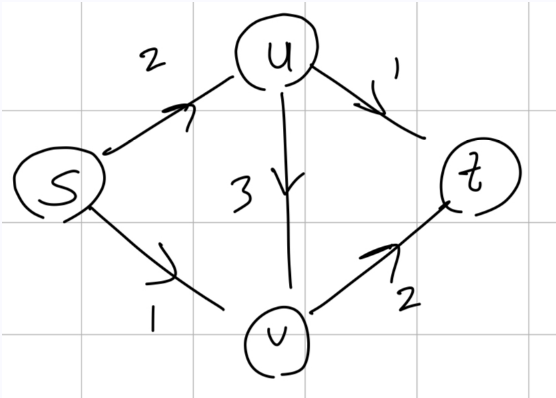
Notice also that $f^{out}(s) = f^{in}(t)$.

Our goal is to find a flow of maximum value.

18.2.1 Toward an Algorithm

Consider a greedy approach: find an $s - t$ path and send as much flow along it as possible.

Example 18.8
Take the following flow network:



On the path from $s \to u \to v \to t$, the best flow we could achieve along this path is 2. Then if we set the flow of the other edges to be 0, then the value of this flow would be 2.

There is a flow of value 3, but we can not achieve it by using this greedy strategy of just looking for a path that can achieve more flow.

To get around this issue, we consider flows in **residual networks**.

Given a flow network and a flow f, their **residual flow network** is as follows:

- The vertices are those of the original flow network
- For each edge e with $f(e) < c_e$, include edge e with capacity $c_e - f(e)$
- For each edge $e = (u, v)$ with $f(e) > 0$, include edge (v, u) with capacity $f(e)$