15.1 Network Flow

Consider a directed edge-weighted graph G where each edge $e \in E(G)$ has a weight designated as a **capacity** $c(e)$. We also have a designated **source vertex** s and **sink vertex** t. Such a graph is called a **flow network**.

A **flow** $f(e)$ on a flow network G assigns a value to each $e \in E(G)$. For each $v \in V(G)$ we have $f^-(v)$ as the sum of flows from incoming edges on v and $f^+(v)$ as the sum of flows on outgoing edges. For non-source and non-sink vertices, a flow is **feasible** if is satisfies constraints:

1. $\forall e \in E(G) : 0 \leq f(e) \leq c(e)$
2. $\forall v \in V(G), v \neq s, t : f^+(v) = f^-(v)$.

The **value** $\text{val}(f)$ of a flow f is the net flow into the sink, $f^-(t) - f^+(t)$. A **maximum flow** is a feasible flow where $\text{val}(f)$ is maximum.

When f is a feasible flow in a network, a f**-augmenting path** is a source-to-sink path P where for each $e \in P$:

1. if P follows e in a forward direction, then $f(e) < c(e)$
2. if P follows e in a backward direction, then $f(e) > 0$

Define $\epsilon(e) = c(e) - f(e)$ when e is forward on P and $\epsilon(e) = f(e)$ when e is backward on P. The **tolerance** of P is $\min_{e \in E(P)} \epsilon(e)$.

If P is an f-augmenting path with tolerance z, then changing flow by $+z$ on forward edges in P and $-z$ on backward edges in P produces a new feasible flow $\text{val}(f') = \text{val}(f) + z$.

In a flow network, a **source-sink cut** $[S, T]$ consists of the edges between a **source set** S and **sink set** T, where S and T partition the nodes and $s \in S, t \in T$. The **capacity** of the cut $[S, T]$, $\text{cap}(S, T)$ is the total capacities of the edges of $[S, T]$, with the net flow from S to T equal to $\text{val}(f)$ and $\text{val}(f) \leq \text{cap}(S, T)$. Among all possible $[S, T]$ cuts, the one with the lowest $\text{cap}(S, T)$ gives us a bound on our maximum flow. The **Max-flow Min-cut Theorem** states the duality between the maximum flow and **minimum cut** problems; specifically, the maximum value of a feasible flow equals the minimum capacity of a source-sink cut.

```
procedure EDMONDS-KARP(Flow Network G(V, E⁺, E⁻C, s, t))
                    ▷ C = edge capacities, s = source vertex, t = sink vertex
    for all e ∈ E(G) do
        F(e) ← 0                                    ▷ Initialize flows to zero
    do                        ▷ Do iterative BFS searches for f-augmenting paths
        for all v ∈ V(G) do
            parent(v) ← −1
        Q ← s, Q_n ← ∅
        while Q ≠ ∅ do
            for all v ∈ Q do
                for all u ∈ N⁺(v) ∪ N⁻(v) : parent(u) = −1 do
                    e ← (v, u)
                    if (F(e) < C(e) and u ∈ N⁺(v)) or (F(e) > 0 and u ∈ N⁻(v)) then
                        parent(u) = v, Q_n ← u
            swap(Q, Q_n), Q_n ← ∅
        if parent(t) = −1 then                      ▷ Did we find path to sink?
            foundpath ← false
        else
            foundpath ← true, tol ← ∞, v ← t
            while v ≠ s do                      ▷ First determine tolerance tol
                u ← parent(v), e ← (u, v)
                if e ∈ E⁺(G) then
                    tol ← min(tol, C(e) − F(e))
                else
                    tol ← min(tol, F(e))
            v ← t
            while v ≠ s do                   ▷ Now use tolerance to update flows
                u ← parent(v), e ← (u, v)
                if e ∈ E⁺(G) then
                    F(e) ← F(e) + tol
                else
                    F(e) ← F(e) − tol
    while foundPath = true
    return (F⁻(t) − F⁺(t))
```

15.2 Max Flow – Edmonds-Karp Algorithm

The general iterative algorithm for identifying f-augmenting paths to incrementally increase the flow in a network is called the **Ford-Fulkerson Algorithm**. When we use BFS to find the shortest augmenting path, we have the **Edmonds-Karp Algorithm**, defined above. Should we wish to find a min cut instead, we can use the set of vertices visited by our BFS before termination as our S source set, unvisited vertices as the T sink set, with the edges between them $[S, T]$ as our minimum cut.