# All-Pairs Max-Flow Algorithms and Implementations

Dillan Pribak

December 14, 2022

# Naive Approach

Run a max flow algorithm on all vertex pairs.
Let $G(V, E, c)$ be a directed graph with integral capacities.
$m = |E|$
$n = |V|$
$f =$ max flow in G
**All-Pairs Ford-Fulkerson:** $O(fm \cdot n^2) = O(fmn^2)$
**All-Pairs Dinitz:** $O(mn^2 \cdot n^2) = O(mn^4)$

# Implementation Details: Overview

**My work:**

- Functions for building and updating residuals.
- DFS, BFS, blocking flow algorithms.

Written in Python.

Used `networkx` package for graph data structure.

Implementations are not fully optimized.

# Implementation Details: Ford-Fulkerson

**procedure** $\textsc{Ford-Fulkerson}(s, t, G = (V, E))$
  $f \leftarrow 0$
  $G_f \leftarrow G$
  **while** there is a path $P$ from $s$ to $t$ in $G_f$ **do**
    $f' \leftarrow$ maximum flow along $P$
    Update residual $G_f$ accordingly
    $f \leftarrow f + f'$

## Custom DFS:

- Returns list of edges in path from $s$ to $t$.
- Ignores zero-weight edges.
- Return path if $t$ found, otherwise return "no path found".

# Implementation Details: Dinitz

**procedure** $\textsc{Dinitz}(s, t, G = (V, E))$
    $f \leftarrow 0$
    **while** $f$ is not a max flow **do**
        Let $G_f$ be the residual
        Let $E'$ be edges in all shortest paths from $s$ to $t$
        $f' \leftarrow$ any blocking flow in $G' = (V, E')$
        $f \leftarrow f + f'$

## Blocking Flow

- Find path from $s$ to $t$ (DFS).
- Push maximum flow through that path.
- Continue until no paths to $t$.

# Implementation Details: Dinitz

**Custom BFS:**

- Each iteration expands search out by one.
- Once $t$ is found, finish that level to check for other paths.
- Maintains a parent list for backtracking where vertices can have multiple parents.
- Returns all edges in shortest paths from $s$ to $t$.

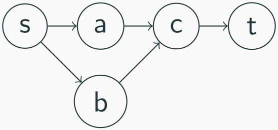


**Figure 1:** Node $c$ has two parents in all shortest paths.

# Better Approach: Gomory-Hu Tree

## Definition

A Gomory-Hu Tree is a tree on the vertices of $G$ where the minimum edge weight in the path between two vertices is the max-flow between those vertices.

**Interpretation:** A tree where the edges are all the bottlenecks.

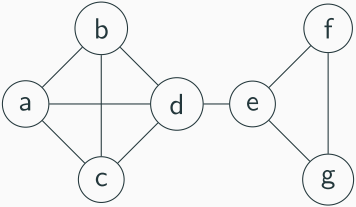


**Figure 2:** Passage from left side to right side bottlenecked by $(e, f)$ edge.

# Better Approach: Gomory-Hu Tree

**Example:**

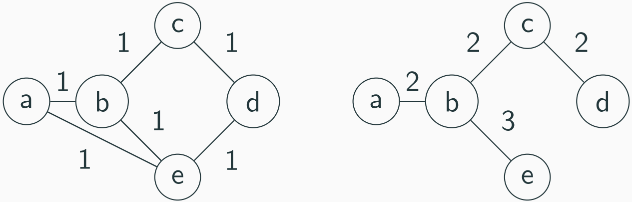


**Figure 3:** A flow network (left) and its Gomory-Hu tree (right)

# Finding Gomory-Hu Tree

**Algorithm [Gomory, Hu 1961]**

Start with all vertices in a big pot. Split into smaller pots via min-cuts. Continue until all pots have one node.

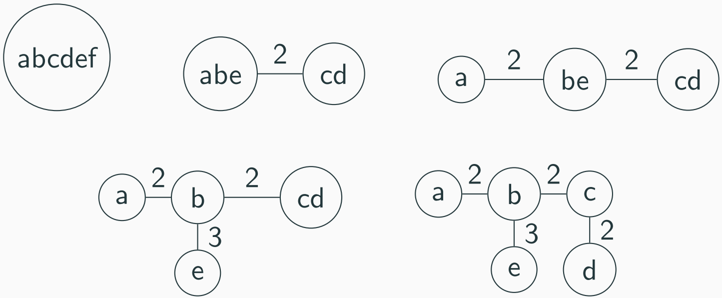


**Figure 4:** Progression of Gomory-Hu algorithm on graph from figure 3

# Gomory-Hu Algorithm Analysis

## Generating Tree

- Creates $(n-1)$ splits $\implies$ makes $(n-1)$ calls to max-flow.
- Better than naive method! (Makes $O(n^2)$ calls to max-flow).

## Accessing Tree

- A flow value can be retrieved efficiently using a shortest path algorithm.
- To get constant access time, convert into a table in $O(n^2)$ time using a good APSP algorithm.

**Note:** Only works on undirected graphs!

# Implementation Details: Gomory-Hu Algorithm

- Built data structure for handling grouped vertices.
- Lots of things going on with splitting and contracting vertices.
- Implemented algorithm to find min cuts.

# Modern Results in Gomory-Hu Trees

**Result [Abboud et. al., 2022]**

A Gomory-Hu tree for a graph can be found in $O(n^2)$ time.

- The new algorithm uses tree packing to find the min cuts.
- The original Gomory-Hu algorithm was state of the art for general graphs for over 60 years.
- Faster solutions have been known for special cases.

github.com/dmpribak/apmf