All-Pairs Max-Flow Algorithms and Implementations

Dillan Pribak

December 14, 2022

Introduction

In class, we investigated a few different algorithms for the max flow problem on a single source and sink, such as Ford-Fulkerson and Dinitz's algorithm. A related problem is solving max-flow for every single pair of vertices in a graph. This is the all-pairs max-flow problem. An obvious solution is to call a max-flow algorithm for every pair of vertices, but this accumulates an extra n^2 factor in the complexity. More sophisticated methods use a structure called a Gomory-Hu tree to encode information about all the max flows. Modern algorithms for finding a Gomory-Hu tree run in $O(n^2)$ time which is a signficant improvement over elementary methods.

Implementations

Three different algorithms were implemented: Ford-Fulkerson, Dinitz's algorithm, and the Gomory-Hu algorithm. These implementations were written in Python and used the `networkx` package for its graph data structure. Everything else was implemented from scratch, including other data structures used by the algorithms and various subroutines they call such as BFS, DFS, and blocking flow.

For Ford-Fulkerson and Dinitz's algorithm, a wrapper function was written to run them on all pairs of vertices to solve all-pairs max-flow.

Findings

A benchmark was performed on each of the algorithms. The task was to find all pairs max flow on a complete graph with 20 vertices and unit weight edges. Actual runtimes varied slightly but were approximately as follows:

Algorithm	Runtime (s)
Ford-Fulkerson	~ 0.50
Dinitz	~ 7.00
Gomory-Hu	~ 0.05

The slow runtime of Dinitz's algorithm may be explained by a poor implementation of BFS or an overuse of graph copy operations.