# The Algorithm

1. Add a new node
2. Add a new node
3. Add a directed edge with capacity
4. Add a directed edge with capacity
5. Run Ford-Fulkerson on this new graph with source and sink
6. Let equal the maximum flow from that
7. Run Ford-Fulkerson on this new graph with source and sink
8. Let equal the maximum flow from that
9. add a sink node
10. direct one edge from each of the two old sinks to
11. recall that
12. let (e.g. when , when )
13. let
14. make the capacities of each of the two new directed edges we made in step 3
15. run the Ford-Fulkerson algorithm on this new graph with source (the old source) and sink .
16. Flow on edges and are the fair flows to and , respectively.

# Proof of Correctness

Observe that the most flow that could go into a sink is the minimum between the sum of the capacities of edges that go out from the source () and the sum of the capacities of edges that go into said sink (). When trying to make the flow fair, we must consider what conditions would make it ­*un*fair. The flow would be unfair when and differ by more than 1 *and* when the graph allows for and to differ by more than 1 also. To solve this dilemma, we would have to restrict and . We restrict them to the three-way minimum between them two and the ceiling of half of . Restricting them to the minimum of and is obvious, and the constraint is to satisfy the runtime requirement, which we’ll talk about later. This constraint is implemented by setting up a new sink node and making each of the two capacities of the directed edges that come from the two old sinks . The algorithm is thus correct because the flows to each of the two sinks aren’t more than