# The Algorithm

We make a new source and call it . Make a directed edge from it to with capacity . We modify the Ford-Fulkerson algorithm a bit for this fair flow problem. First of all, we now have two sinks, and we alternate between the two sinks in finding a source to sink path (which at the same time increments the flow to each sink). Also, instead of incrementing the flow by each time, we now increment it by only 1 each time on one path. This modified algorithm runs for as long as there’s a path from to or there’s a path from to and that the flows to each sink don’t differ by more than 1. If you prefer reading pseudocode:

Fair-Flow

Let f(e)=0 for all e

Let s\_new be a new node

Let edge (s\_new, s) have capacity C1 + C2

Let G1 = graph whose source is s\_new and sink is t1

Let G2 = graph whose source is s\_new and sink is t2

Let f1 = flow going into t1

Let f2 = flow going into t2

Let iteration\_num = 0

while ((G1 has an s\_new -> t1 path) or (G2 has an s\_new -> t2 path)) and (f1 and f2 don't differ by more than 1)

if iteration\_num is even

set up G2

find an s\_new -> t2 path P2 in G2

delta = 1 //instead of what we had before with the min(min ce ...

f(e) = f(e) + delta for forward edges in P2

f(e) = f(e) - delta for backward edges in P2

f2 = f2 + delta

else

set up G1

find an s\_new -> t1 path P1 in G1

delta = 1 //instead of what we had before with the min(min ce ...

f(e) = f(e) + delta for forward edges in P1

f(e) = f(e) - delta for backward edges in P1

f1 = f1 + delta

iteration\_num = iteration\_num + 1

endwhile

# Proof of Correctness

Note that the step where we make a new source and direct an edge from it to with capacity won’t decrease the maximum flow possible because the most flow that could go into a sink is the minimum between and (sum of capacities of all edges leaving the source); is greater than this minimum. This modified algorithm is correct because we

1. take care to never let the difference between the flows going into the two sinks exceed 1 because of an invariant in the while loop and that we take turns incrementing the flow between the two sinks by the smallest unit and
2. under that constraint, still maximize the (fair) flow because we are still running the original algorithm, just more slowly because we now increment the flow each time by the smallest unit possible (integer 1).

# Runtime Analysis

Recall that the runtime complexity of Ford-Fulkerson is with being the number of edges and being the sum of the capacities of all edges leaving the source. This modified algorithm’s new source has capacity , and the time per iteration is doubled, which doesn’t change its time complexity in big-O notation (it’s still ). The runtime is thus .