ProgTeam Spring Week 2

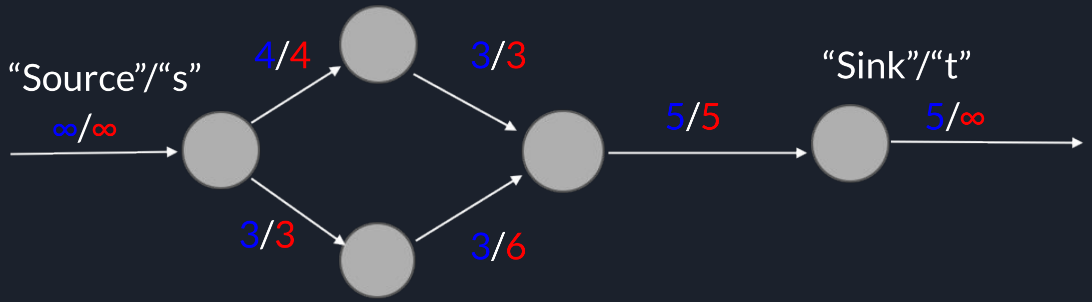
Maximum Flow

Maximum Flow

- Analogy: sending water through pipes
- Each pipe has a capacity
- How much water can flow through the network?
- Flows from source to sink
- Time complexity: Hard to say
  - If all edges have capacity 1, $O(E\sqrt{E})$
  - *Usually* 10^4 edges is "safe"

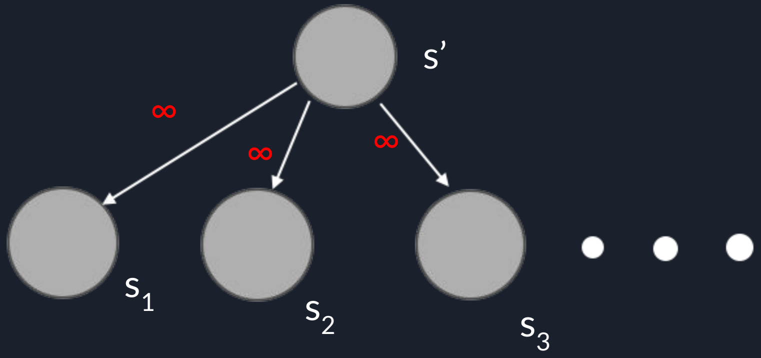
Capacities in red

Flow amount in blue



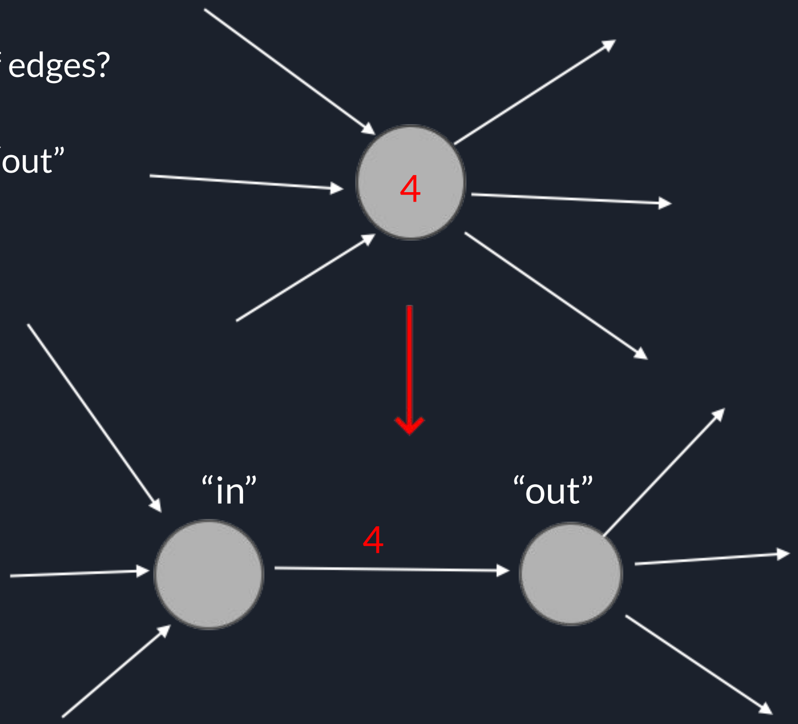
Maximum Flow: Augmentations

- Multiple sources/sink?
  - Instead of rewriting algorithm, add virtual source/sink:



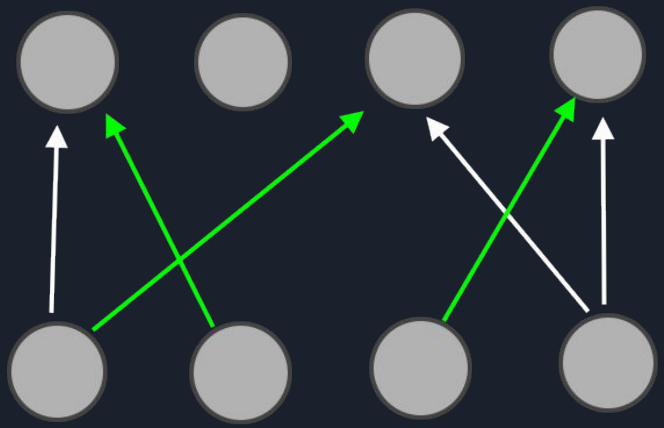
Maximum Flow: Augmentations

- Vertices have capacities instead of edges?
  - Give edges infinite weight
  - Split vertex in two: “in” and “out”



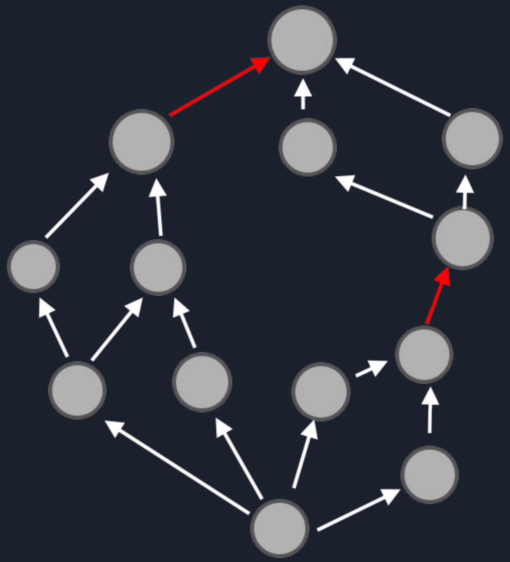
Maximum Flow: Applications

- Maximum matching: maximum number of pairs that can be matched on "bipartite" graph
    - Bipartite = "can be cut in half"



Maximum Flow: Applications

- Minimum cut: minimum number of edges needed to separate two vertices
  - Exactly equal to maximum flow
- Retrieving the cut edges
  - The easy way: remove the edge and check the flow again
  - If it goes down, that edge was part of the cut
  - (More expensive than advanced methods)



Maximum Flow: Applications

- Simulation over time
  - Ex. squares that can hold one person, roads that can let one car through, etc.
- Make into 2D solution: one dimension for graph, one dimension for *time*
- Example: each road can let one car through. Each intersection has K cars. What's the max. number of cars that can pass in T seconds?

Maximum Flow: Applications

- Example: each road can let one car through per minute. Each intersection has K cars. What's the max. number of cars that can pass in T seconds?
- Left = original, right = augmented (repeat pattern T times)
- Blue edges = wait until next minute, have infinite capacity
  White edges = take this road to next vertex, have capacity of one