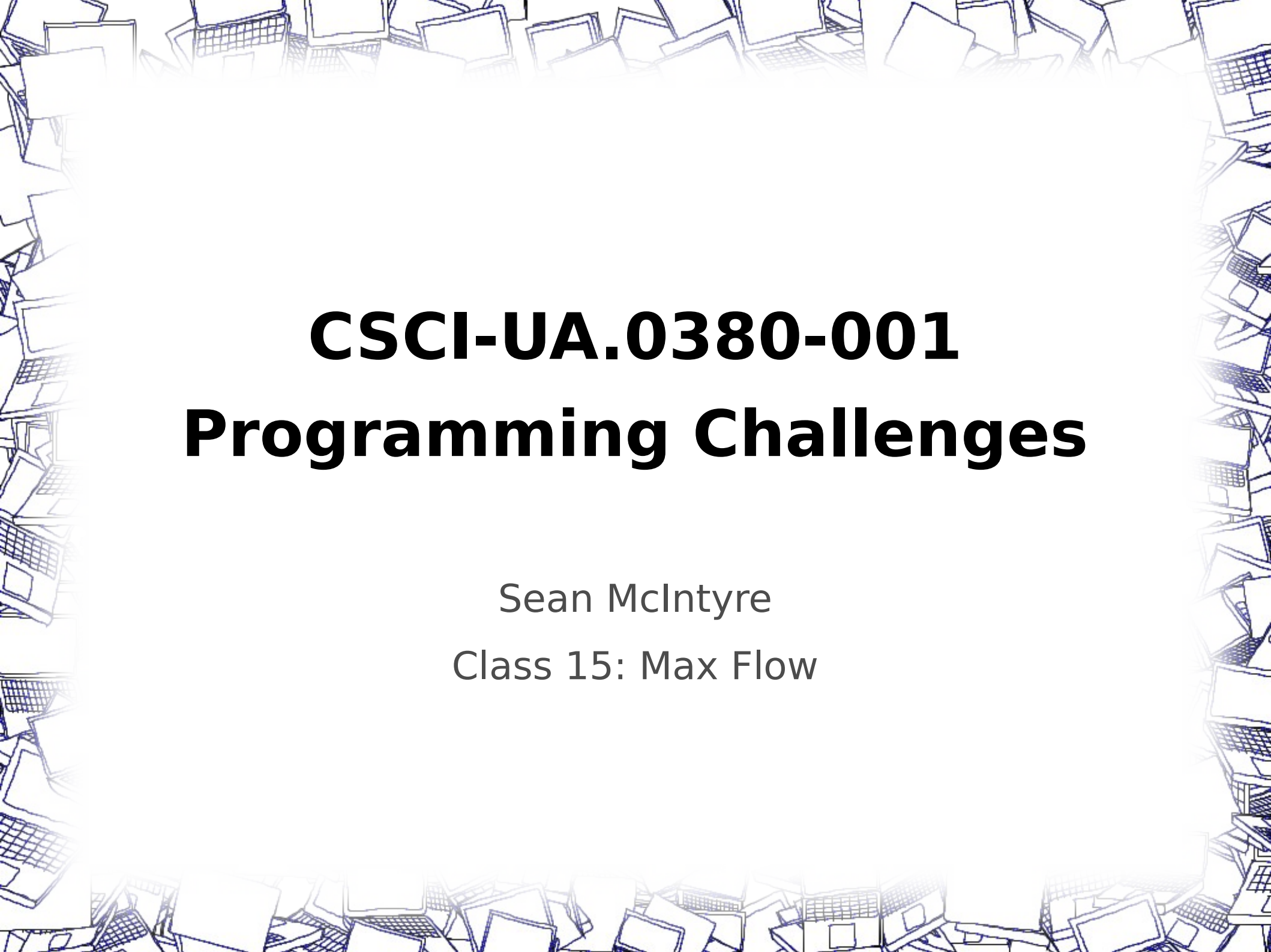


CSCI-UA.0380-001
Programming Challenges

Sean McIntyre

Class 15: Max Flow

Exercise 1

- Example data:

Application label

times to run application

- A4 01234;
  Q1 5;
  P4 56789;

Computers that can run the application

- A4 01234;
  Q1 5;
  P5 56789;

Exercise 1

- Problem take-aways
  - A number of batch jobs are to be run on a number of computers
  - Batch jobs take a day
    - No multitasking on the computer
  - Certain computers are set up to run certain batch jobs
  - Two or more of the same batch may be run
  - What is a possible allocation of jobs → computers so that all jobs run in one day?

Max Flow

- Exercise 1 is a classic max flow problem
- Given a “network” graph
  - Connected, weighted, directed
  - Edges act as pipes
    - Weight is the capacity of the pipe
  - Vertices act as splitting points of the pipes
  - Two special nodes
    - Source node s
    - Sink node t

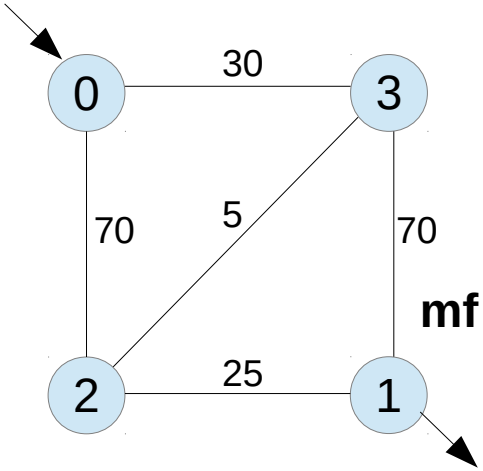
Max Flow

- Given a “network” graph, what is the maximum flow from the source to the sink?
  - How much water can travel through the pipes without bursting?
- Two methods we'll talk about today to solve this problem
  - Ford-Fulkerson and Edmonds-Karp

Max Flow

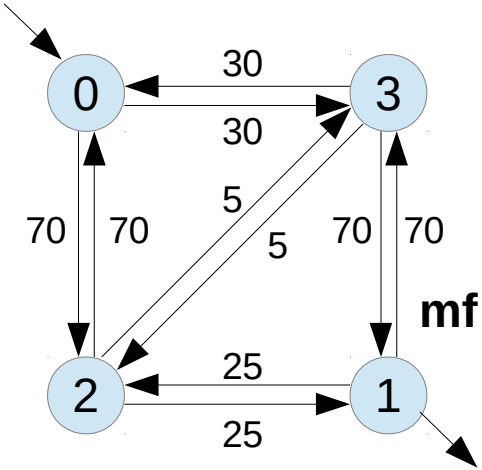
- Ford-Fulkerson
  - Send a flow down a path *p* whenever there exists an augmenting path *p* from *s* to *t*
    - An augmenting path is any path that has capacity
  - Find augmenting paths by using DFS

Max Flow



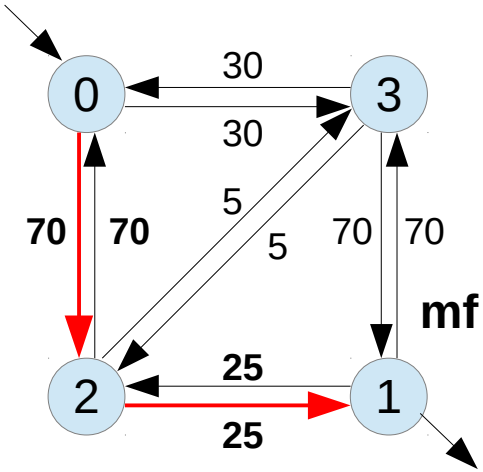
mf = 0

Take the original graph, make it directed



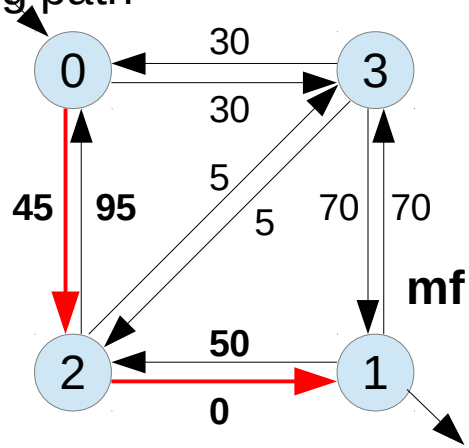
mf = 0

Find an augmenting path



mf = 0

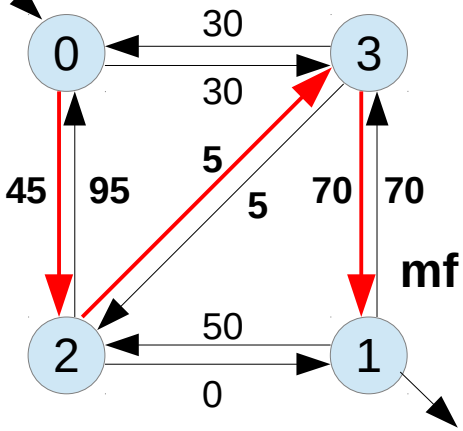
Reduce capacity on path



mf = 25

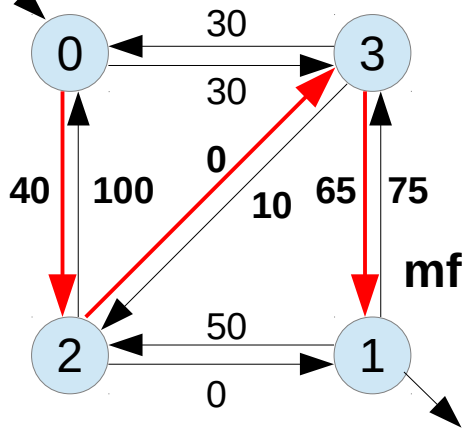
Max Flow

Find an augmenting path



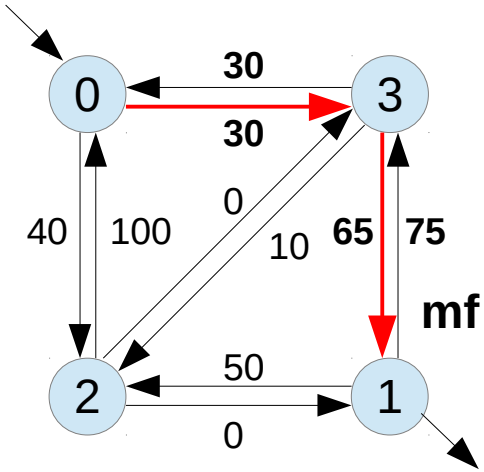
Reduce capacity on path

mf = 25



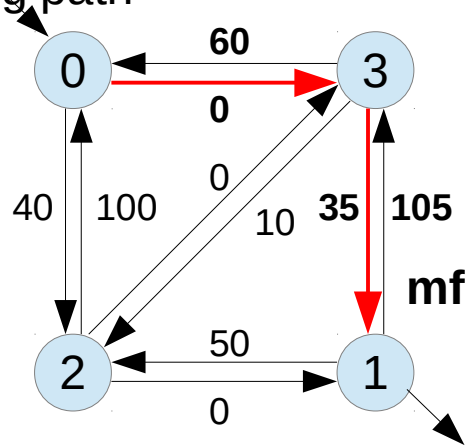
mf = 30

Find an augmenting path



Reduce capacity on path

mf = ~~30~~



mf = 60

Done!

Max Flow

- Ford-Fulkerson algorithm
  1) $mf \leftarrow 0$
  2) while (exists an augmenting path p from s to t)
     1) Send flow along $p = s \rightarrow \ldots \rightarrow i \rightarrow j \rightarrow \ldots \rightarrow t$
     2) Find f, the minimum edge weight along p
     3) Decrease weight of forward edges $i \rightarrow j$ by f
     4) Increase weight of backward edges $j \rightarrow i$ by f
     5) $mf \leftarrow mf + f$
  3) Output mf

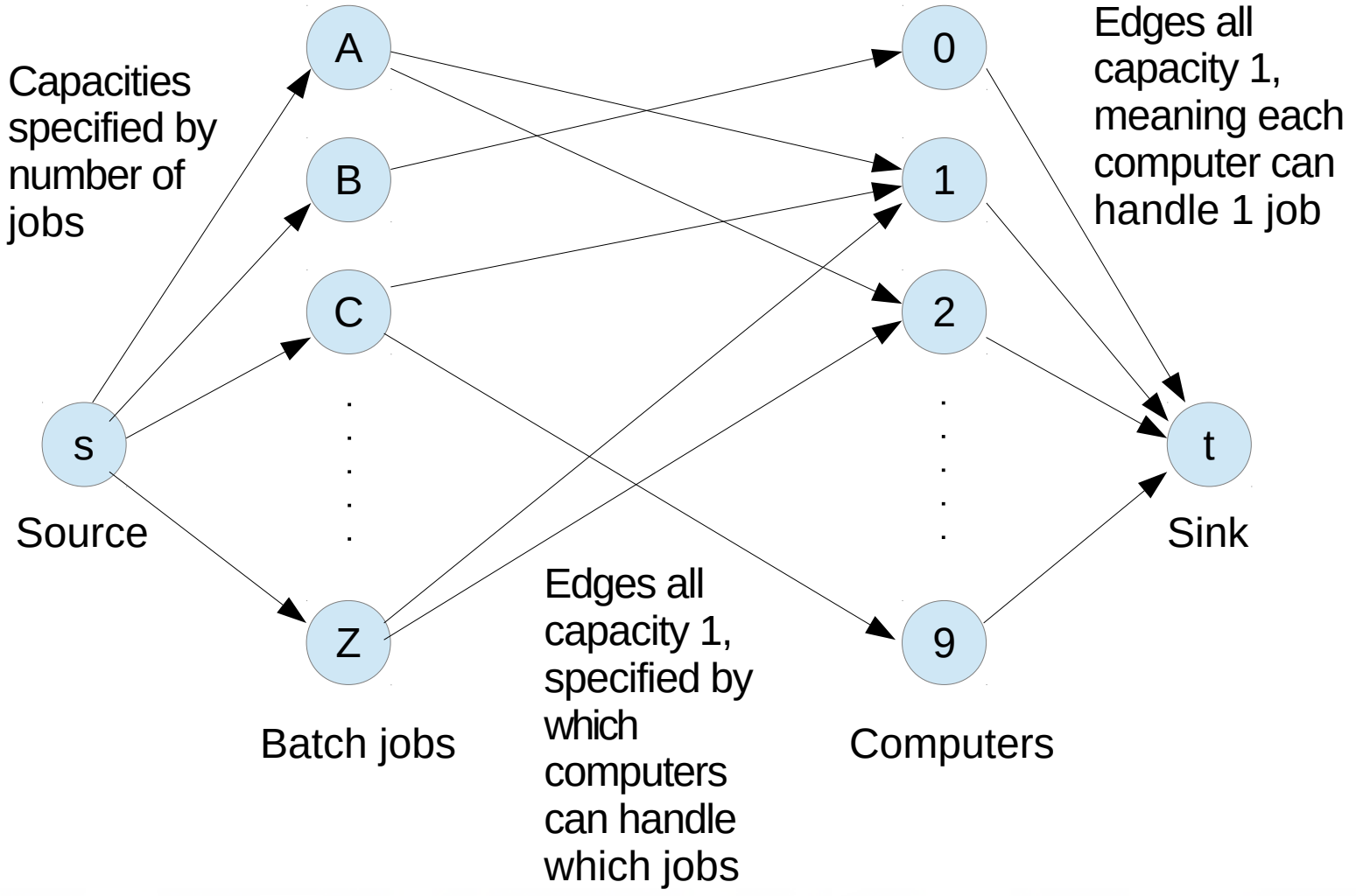
Max Flow

- Ford-Fulkerson algorithm
  - Use DFS to find an augmenting path
  - Runtime: O(E *mf*)
    - *mf* is the maximum flow in the graph
  - So if the solution is large, the runtime could be large!

Back to exercise 1

- How to apply max flow to this problem
  - Treat each application as a vertex
  - Treat each computer as a vertex
  - Draw an edge from each application and computer of capacity 1
  - Create a source node connecting to each application of capacity # batch jobs
  - Create a sink node connecting from all computers of capacity 1

Back to exercise 1



Back to exercise 1

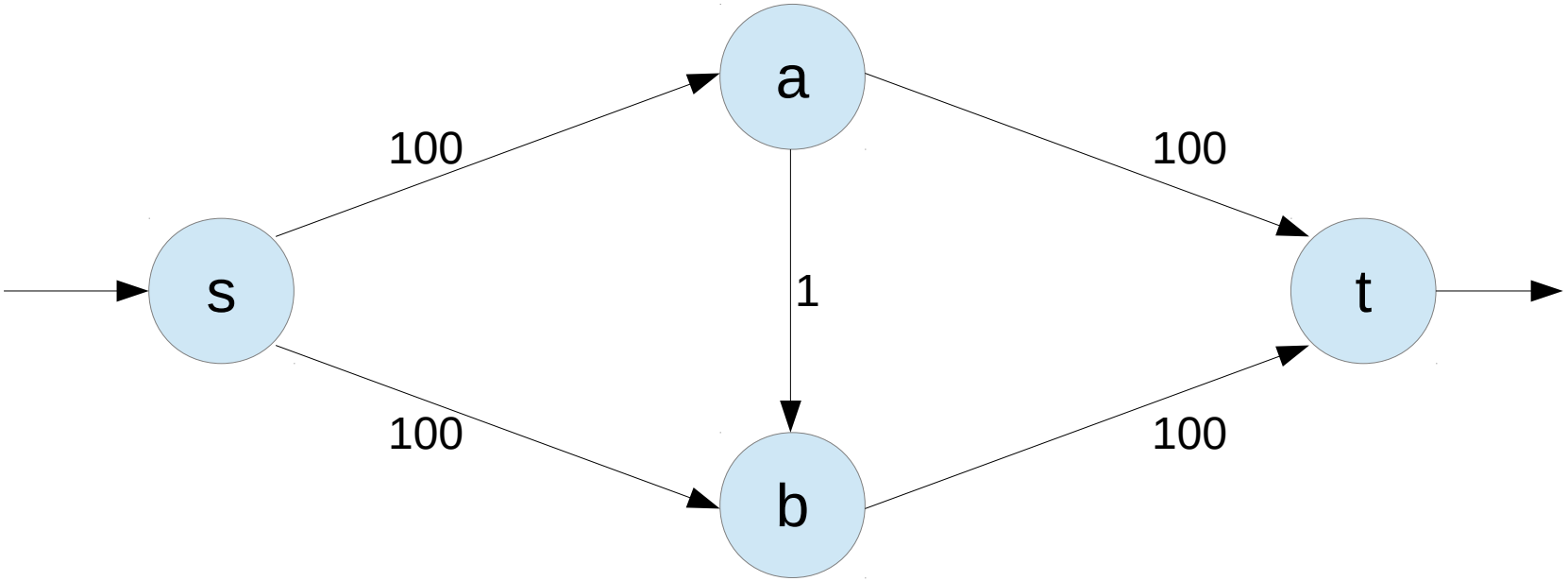
- Reconstructing the solution
  - Look at the residual graph (edge weights from the graph after running max flow)
  - Where edge weights are 0, there is a matching

Back to exercise 1

- Runtime:
  - Ford-Fulkerson
    - Resultant value is at most 10
    - Edges:
      - Between source and applications (26)
      - Between applications and computers (26 * 10)
      - Between computers and sink (10)
      - Total 296
    - O(E * mf), seems very reasonable

Max Flow

- Ford-Fulkerson algorithm degenerate case



Max Flow

- Edmonds-Karp algorithm
  - Use BFS to find an augmenting path
  - Runtime: $O(VE^2)$
    - Provable that after O(VE) iterations, all augmenting paths are exhausted
  - Does not run into the same problem as the Ford-Fulkerson degenerate case

Back to exercise 1

- Runtime:
  - Edmonds-Karp algorithm, $O(VE^2)$
    - Number of edges: 296
    - Number of vertices:
      - Source and sink (2)
      - Applications (26)
      - Computers (10)
      - Total 38
    - Also seems reasonable

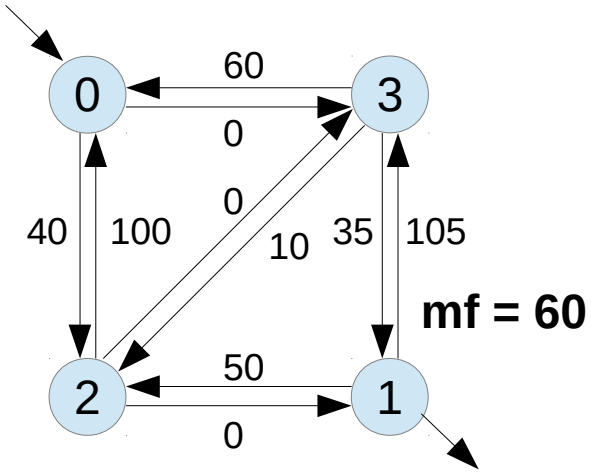
Max Flow

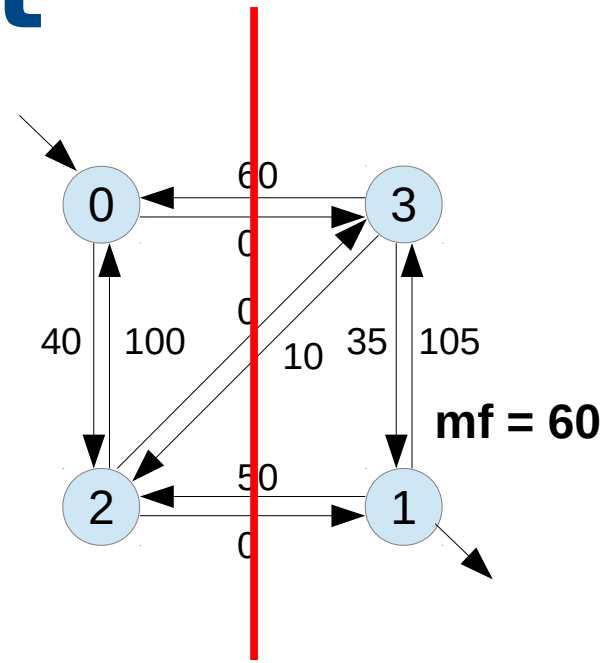
- The challenge of max flow
  - These problems come down to identifying the problem type, building the graph, and running Ford-Fulkerson or Edmonds-Karp
    - Algorithm choice depends on the constraints of the problem
  - Reconstructing the solution can be done for a “matching”
- Now try exercises 2, 3, 4

Max Flow

- Dinic's algorithm
  - Another technique for solving max flow discussed in the book (chapter 9)
  - Runtime is $O(V^2E)$

Min Cut





A consequence of computing the maximum flow is computing the **minimum cut**

Min cut: The smallest cost (or cut set) for removing edges so that the graph is split into two disconnected components

Flooding fields

- Problem gist
  - Up to 50 cows scattered on a 100x100 field
  - Field contains square plots that are of different elevation
  - Fields flood every hour to a new level 0-100 over 24 hours
  - Cows can move every hour to an adjacent plots whose elevations are higher than level
  - Only one cow can be on one plot at the same time

Max flow readings

- Section 4.6