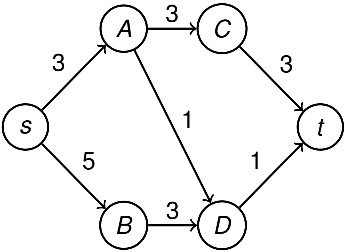
Lecture Pattern Analysis

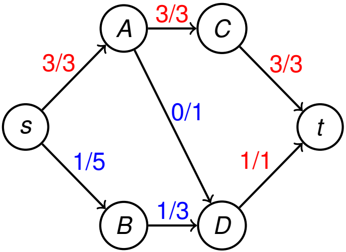
Part 23: Recap: Max Flow and Min Cut

Christian Riess
IT Security Infrastructures Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
July 2, 2021

Overview

- Max flow is a combinatorial standard problem, solved in polynomial time
- Given: graph with positive edge weights, source node, sink node
- Task: Interpret edge weights as tube diameters, and determine the maximum possible throughput ("flow") of water from source *s* to sink *t* per time unit:

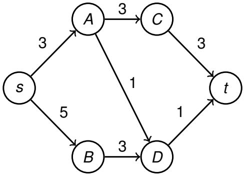




- The minimum cut task seeks the smallest sum of edges to disconnect *s* and *t*.
- Max flow and min cut are identical: a min cut is easily found, e.g., by selecting the red edges until there is no *s*-*t* path left

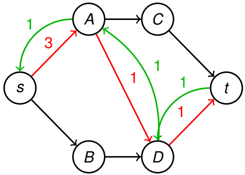
Ford Fulkerson in a Nutshell (1/2)

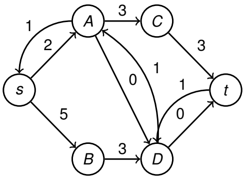
- Max flow algorithm by Ford and Fulkerson is probably most well-known:
  1. Greedily search shortest path
  2. Max out the flow capacity along that path, reduce edge weights
  3. Introduce backward edges to undo greedy dead ends, goto 1) if *s*-*t* path left



Shortest path: 5

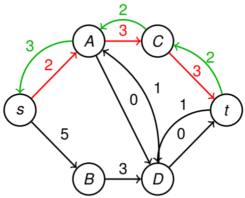
Path flow: 1





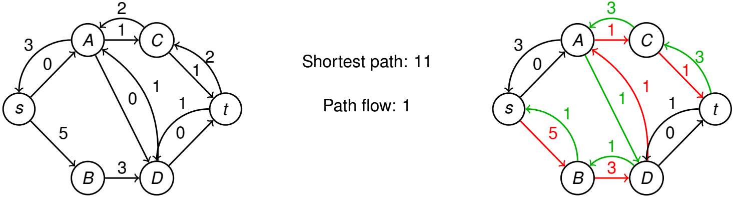
Shortest path: 8

Path flow: 2



Ford Fulkerson in a Nutshell (1/2)

- The third shortest path uses a back link, and completes the max flow algorithm:



- The total flow is $1 + 2 + 1 = 4$, with pipe usage as shown on slide 1
- The minimum cut includes the edge $D \rightarrow t$ and any one of the edges $(s \rightarrow A, A \rightarrow C, C \rightarrow t)$
- Hence, the four sets of edges for equivalent minimum cuts are
  - $(C \rightarrow t, D \rightarrow t)$,
  - $(A \rightarrow C, D \rightarrow t)$, and
  - $(s \rightarrow A, A \rightarrow D, D \rightarrow t)$, where $A \rightarrow D$ is a “backward edge” that does not count