Algorithm Design and Analysis

CSE
565

LECTURES 20

Maximum Flow Applications

- Edge-disjoint paths
- Image segmentation
- Project selection

Extensions to Max Flow

Sofya Raskhodnikova

Exercise

We have been considering flows in graphs where the source has only outgoing edges and the sink has only incoming edges.

- Suppose the source also has incoming edges. How should we define max flow in such a graph?

- Can we reduce this variant of the problem to the one we solved before?

(When the sink has outgoing edges, the solution is ``symmetric''.)

7.6 Disjoint Paths

Application of Max Flow With $C=1$

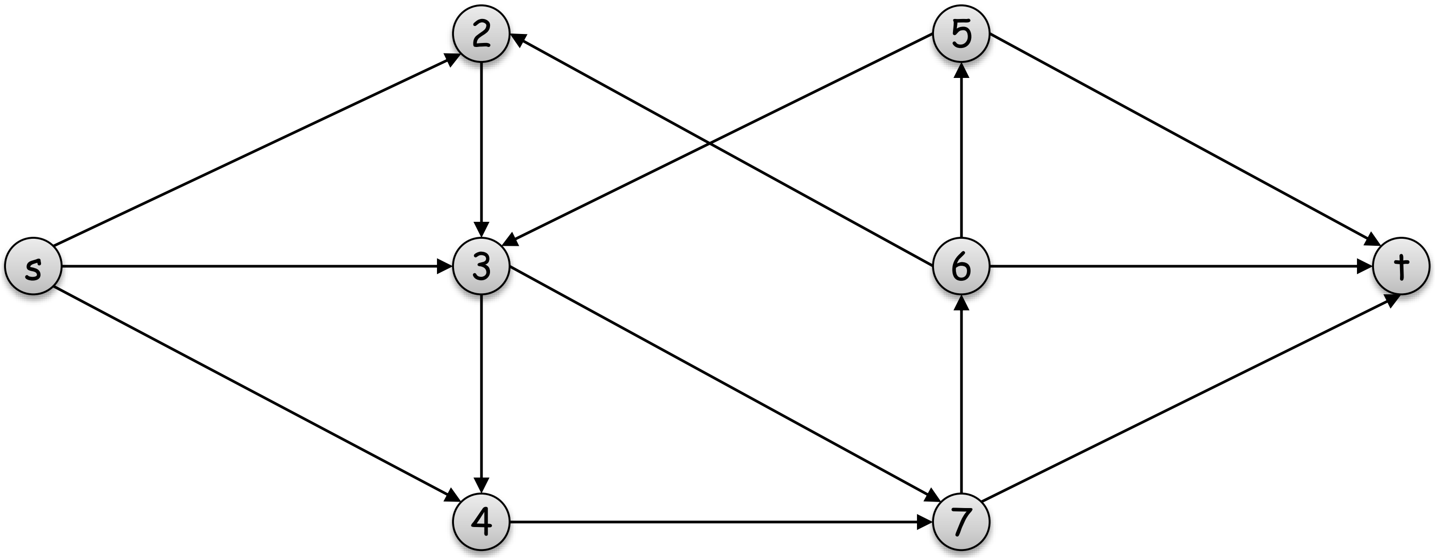
Two problems

Given a network:

- Find edge-disjoint paths
- Find how many edges must be deleted to disconnect the graph

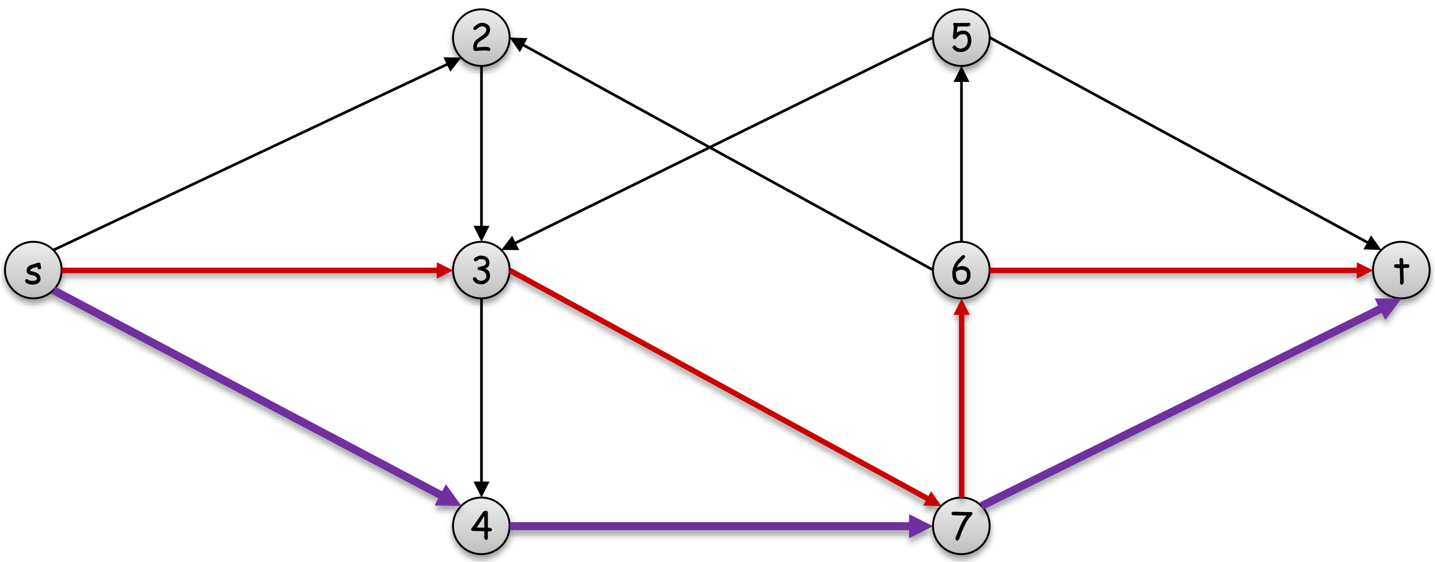
Edge Disjoint Paths

- **Disjoint paths problem.** Given a digraph $G = (V, E)$ and two nodes s and t, find the max number of edge-disjoint s-t paths.
  - Two paths are edge-disjoint if they have no edge in common.
  - In networks: how many packets can I send in parallel?



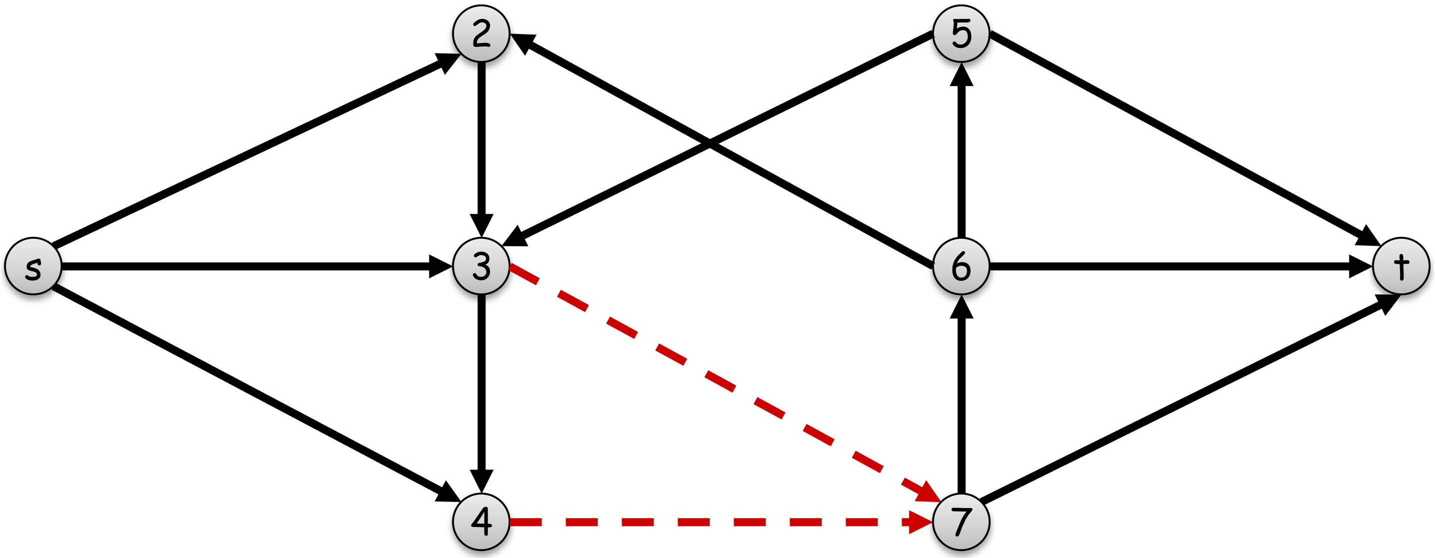
Edge Disjoint Paths

- **Disjoint paths problem.** Given a digraph $G = (V, E)$ and two nodes s and t, find the max number of edge-disjoint s-t paths.
  - Two paths are edge-disjoint if they have no edge in common.
  - In networks: how many packets can I send in parallel?



Network Connectivity

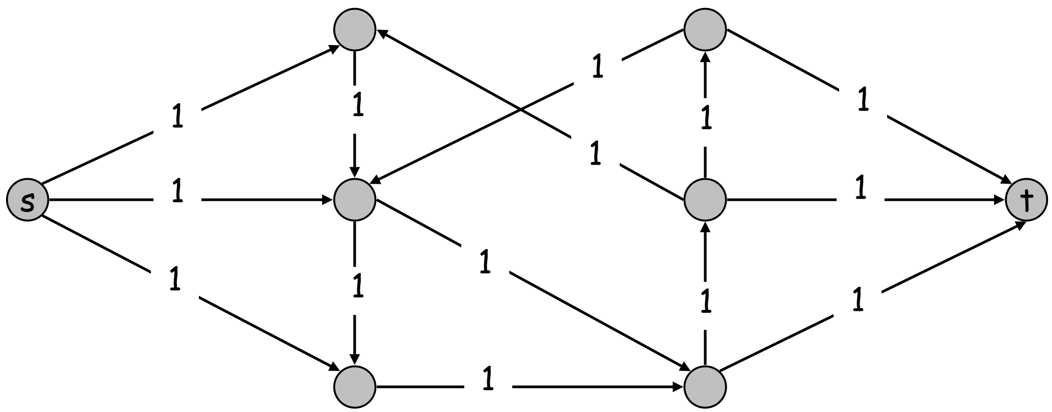
- **Network connectivity problem.** Given a digraph G = (V, E) and two nodes s and t, find min number of edges whose removal disconnects t from s.
  - A set of edges $F \subseteq E$ disconnects t from s if each s-t paths uses at least one edge in F.
    (That is, removing F would make t unreachable from s.)



How is it related to edge-disjoint paths?

Edge Disjoint Paths

Max flow formulation: assign unit capacity to every edge.



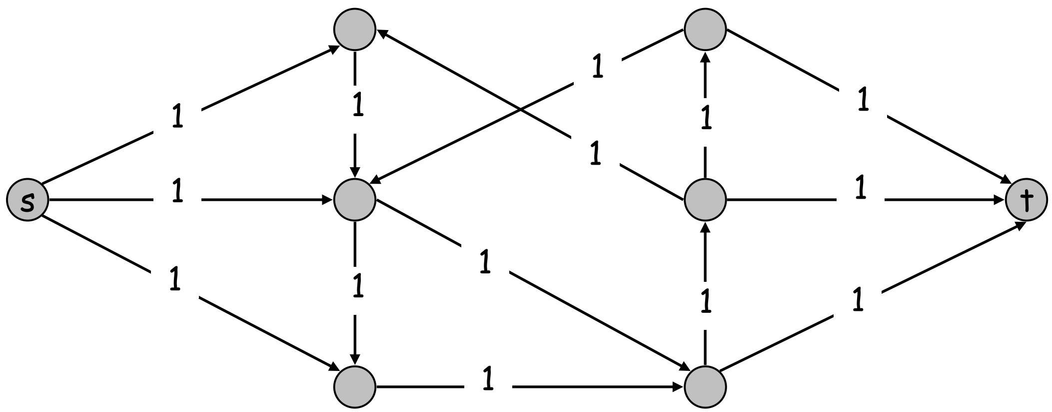
Theorem. Max number edge-disjoint s-t paths equals max flow value.

Proof. $\leq$

- Suppose there are k edge-disjoint paths $P_1, \ldots, P_k$.
- Set f(e) = 1 if e participates in some path P_i ; else set f(e) = 0.
- Since paths are edge-disjoint, f is a flow of value k. ▪

Edge-Disjoint Paths

Max flow formulation: assign unit capacity to every edge.



Theorem. Max number edge-disjoint s-t paths equals max flow value.

Proof. $\geq$

- Suppose max flow value is k.
- Integrality theorem $\Rightarrow$ there exists 0-1 flow f of value k.
- Consider edge (s, u) with f(s, u) = 1.
  - by conservation, there exists an edge (u, v) with f(u, v) = 1
  - continue until reach t, always choosing a new edge
- Produces k (not necessarily simple) edge-disjoint paths. ▪

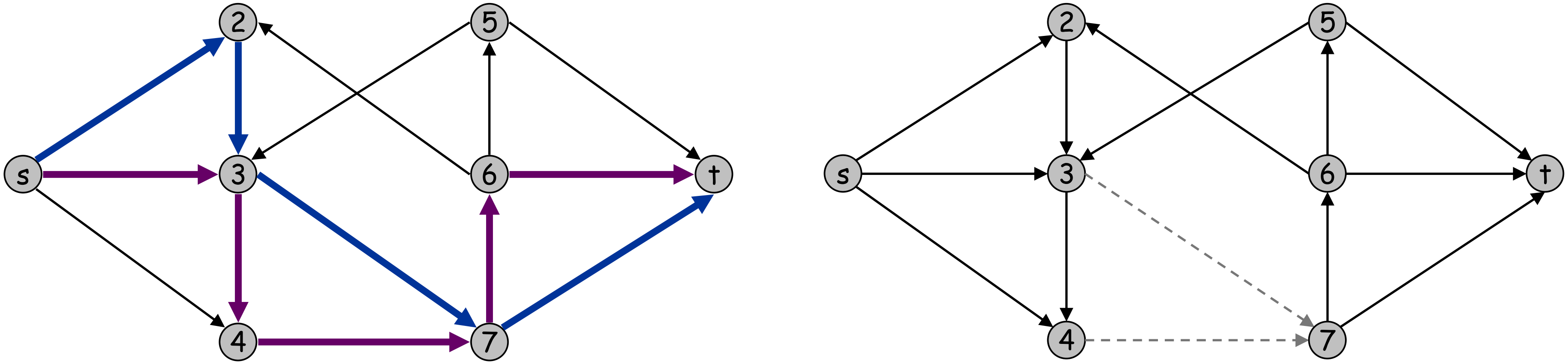
can eliminate cycles to get simple paths if desired

Edge-Disjoint Paths and Network Connectivity

Theorem. [Menger 1927] The max number of edge-disjoint s-t paths is **equal** to the min number of edges whose removal disconnects t from s.

Proof. $\leq$

- Suppose the removal of $F \subseteq E$ disconnects t from s, and $|F| = k$.
- All s-t paths use at least one edge of F. Hence, the number of edge-disjoint paths is at most k. ▪

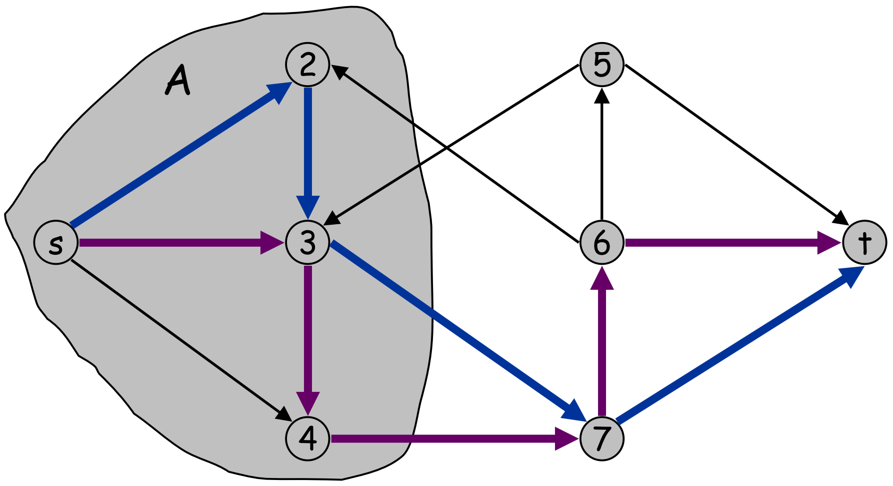


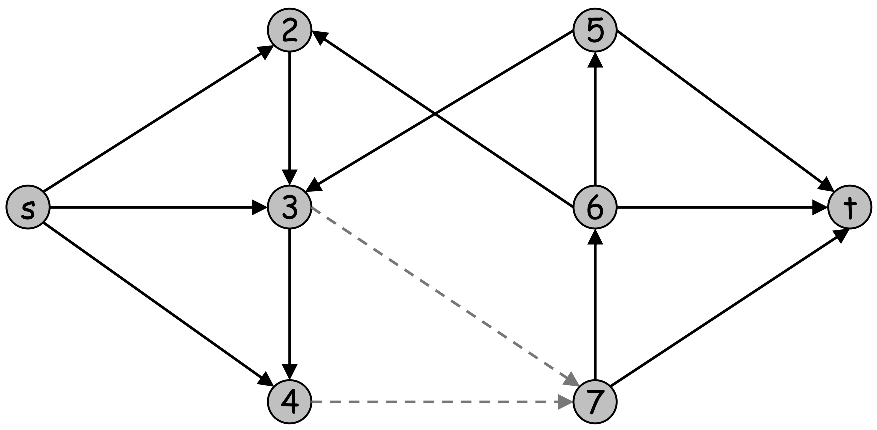
Edge-Disjoint Paths and Network Connectivity

Theorem. [Menger 1927] The max number of edge-disjoint s-t paths is equal to the min number of edges whose removal disconnects t from s.

Pf. $\geq$

- Suppose max number of edge-disjoint paths is k.
- Then max flow value is k.
- Max-flow min-cut $\Rightarrow$ cut (A, B) of capacity k.
- Let F be set of edges going from A to B.
- |F| = k and disconnects t from s. ▪





In-class exercise

- Getting ambulances to accidents.
  - Inputs: T, $d_1, \dots, d_n$ and $s_1, \dots, s_k$ and driving times $t_{i,j}$ for all i, j
    - Accident i needs d_i ambulances ($i \in \{1, \dots, n\}$)
    - Ambulance station j has s_j ambulances ($j \in \{1, \dots, k\}$)
    - Ambulance needs to be within T minutes' drive of accident
  - Give an algorithm to determine if there is a way to assign enough nearby ambulances to each accident
- Pose as a flow problem
  - What is the graph?
  - What are the capacities?
  - How do you solve the original problem once you know the maximum flow?
  - What is the running time of the resulting algorithm?

- Variation: What if T is not given? Can we find the smallest T for which the problem is feasible?