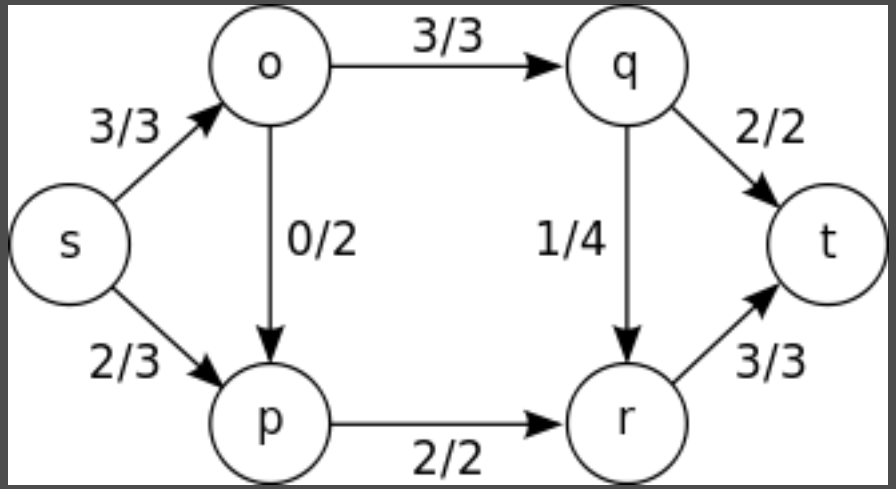
Advanced Computer Contest Preparation

Lecture 29

MAXIMUM FLOW

Flow Network

- A directed, edge-weighted graph
- Edge weights represent capacity of edges
- Each edge can receive a **flow** (like a pipe)

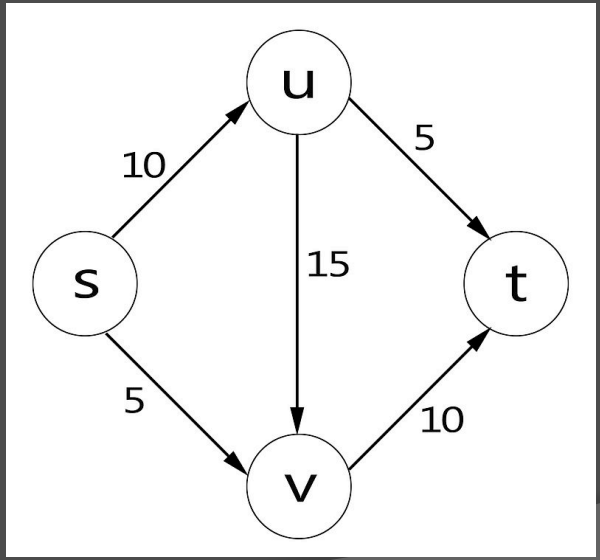


Flow Network – Constraints

- Flow on an edge cannot exceed its capacity
- The net flow from node u to v must be the opposite of the net flow from v to u
  - $flow(u \rightarrow v) = -flow(v \rightarrow u)$
- All flows coming into a node must go out (total flow from any node to its neighbors is 0) except:
  - Source (start) node s can have infinite flow going out
  - Sink (end) node t can have infinite flow coming in

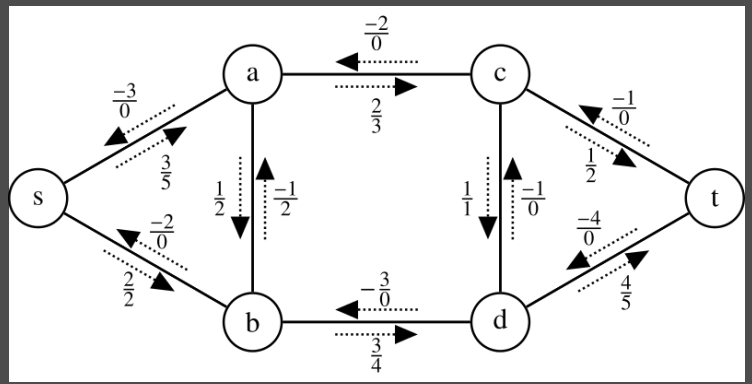
Maximum Flow Problem

- Given a flow network, what is the maximum amount of flow that can be sent on this network?

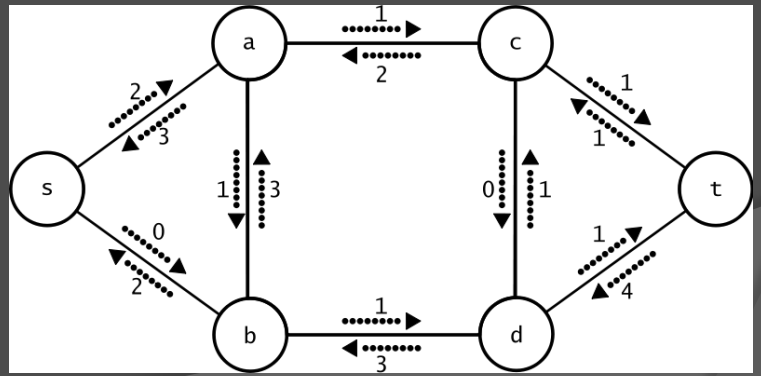


Ford-Fulkerson Method

- Uses the **residual graph** of the network flow to compute the maximum flow
  - For each pair of nodes $(u \rightarrow v)$, whether there is an edge or not, the residual capacity $res(u \rightarrow v)$ is $cap(u \rightarrow v) - flow(u \rightarrow v)$



f/c denotes flow/capacity

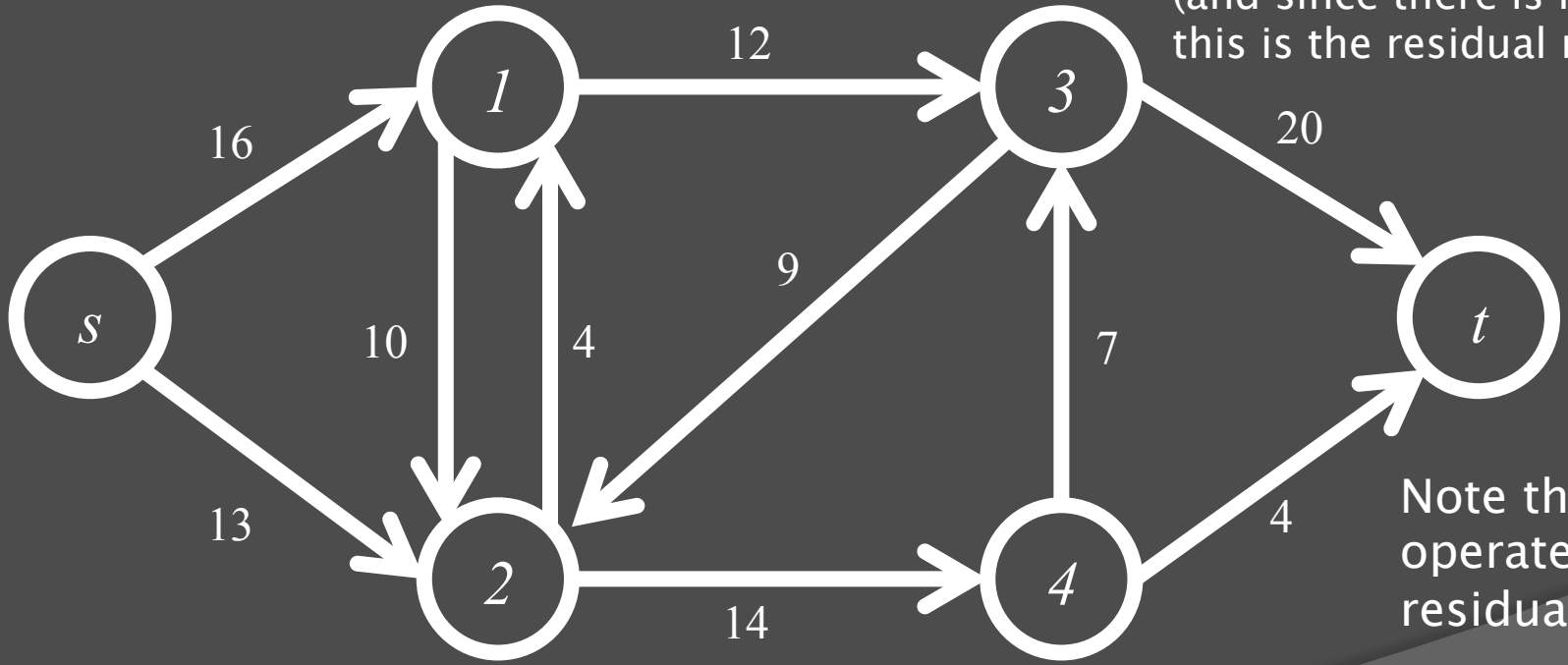


Residual graph of the network flow

Ford-Fulkerson Method

- While there is a path of non-zero edges from source s to sink t in the residual graph:
  - Find such a path (**augmenting path**)
  - Find the minimum value of the residual capacity on this path
  - Increase the flow of each edge by this amount
  - Decrease the flow of each opposite edge
    - Keep the flow network valid
    - The flow might be “returned” later
  - Normally, we operate on the residual graph only

Ford-Fulkerson Diagram



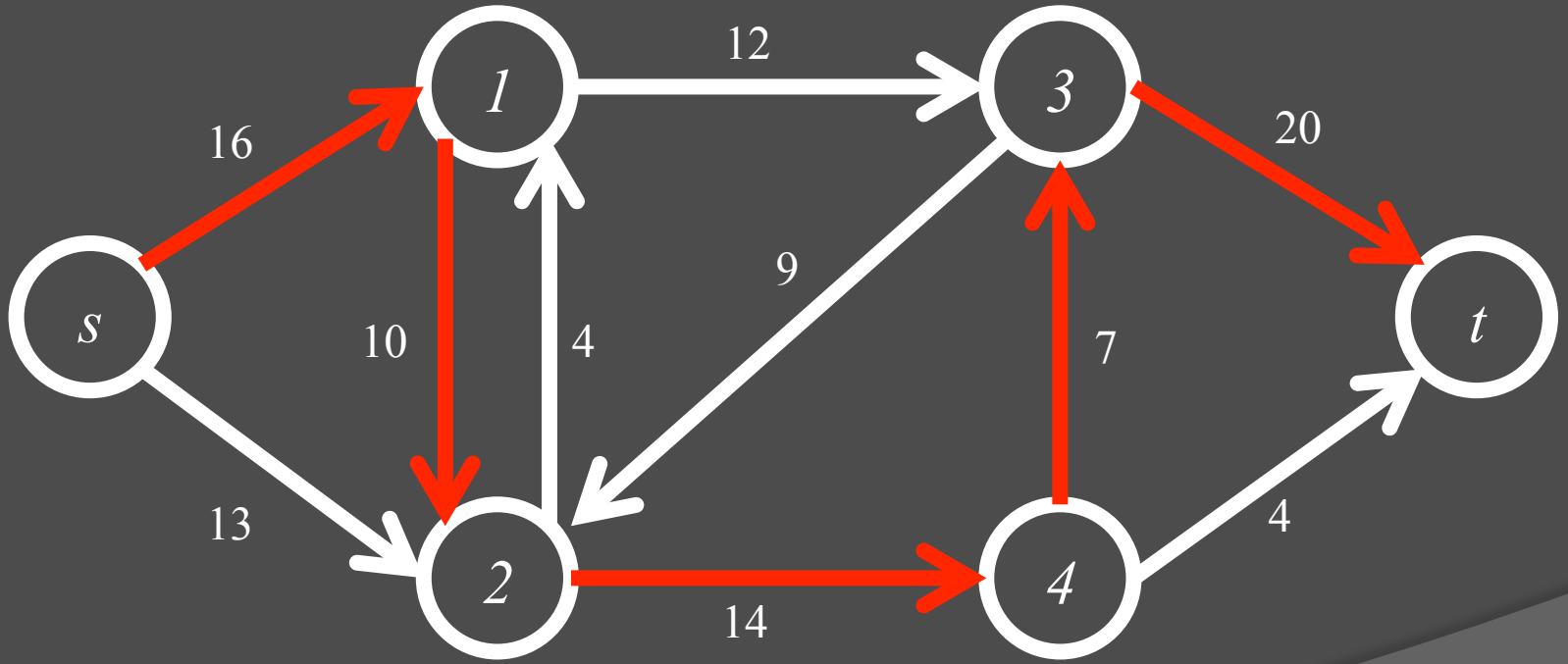
The initial flow network (and since there is no flow, this is the residual network).

Note that we operate on the residual graph.

Flow: 0

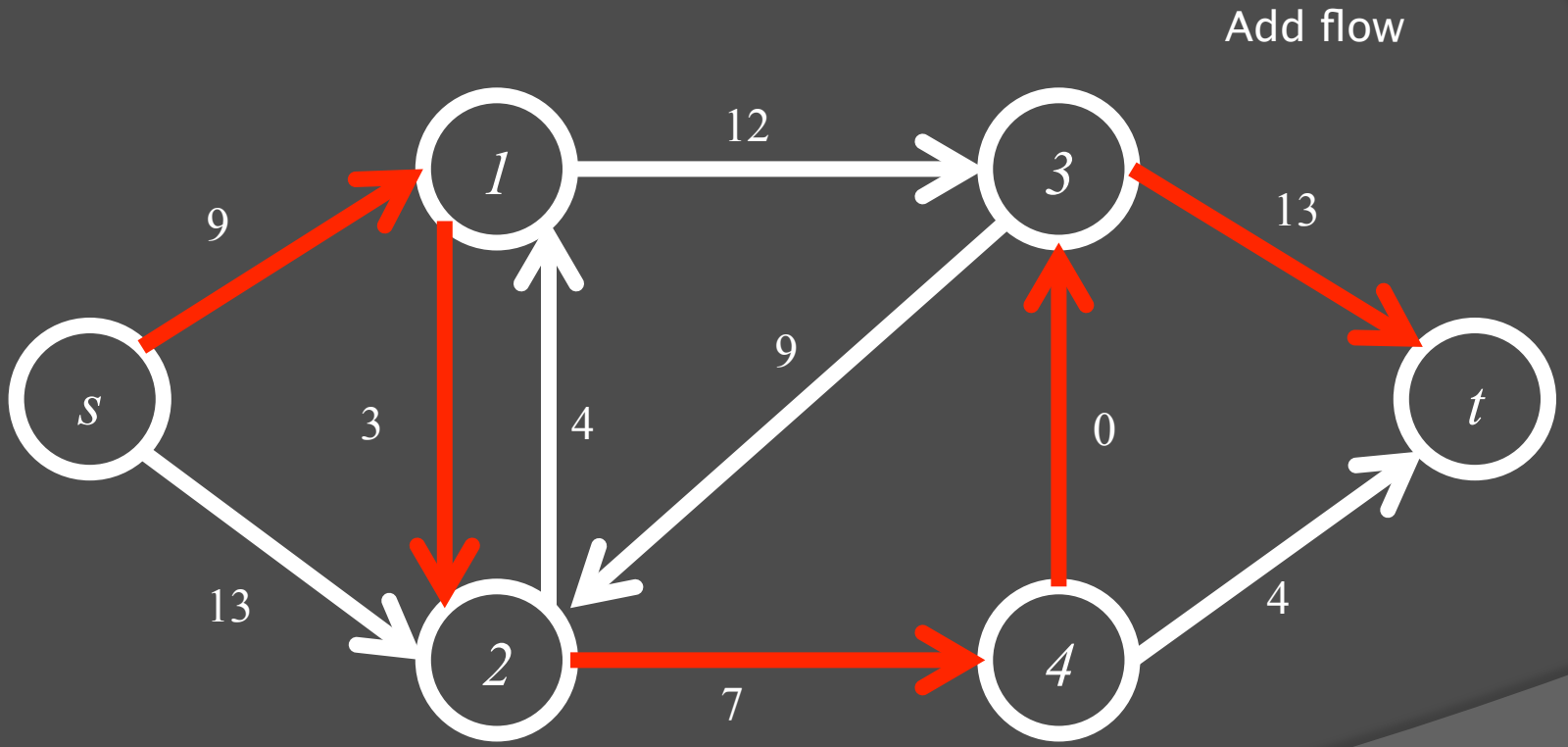
Ford-Fulkerson Diagram

Find an s–t path



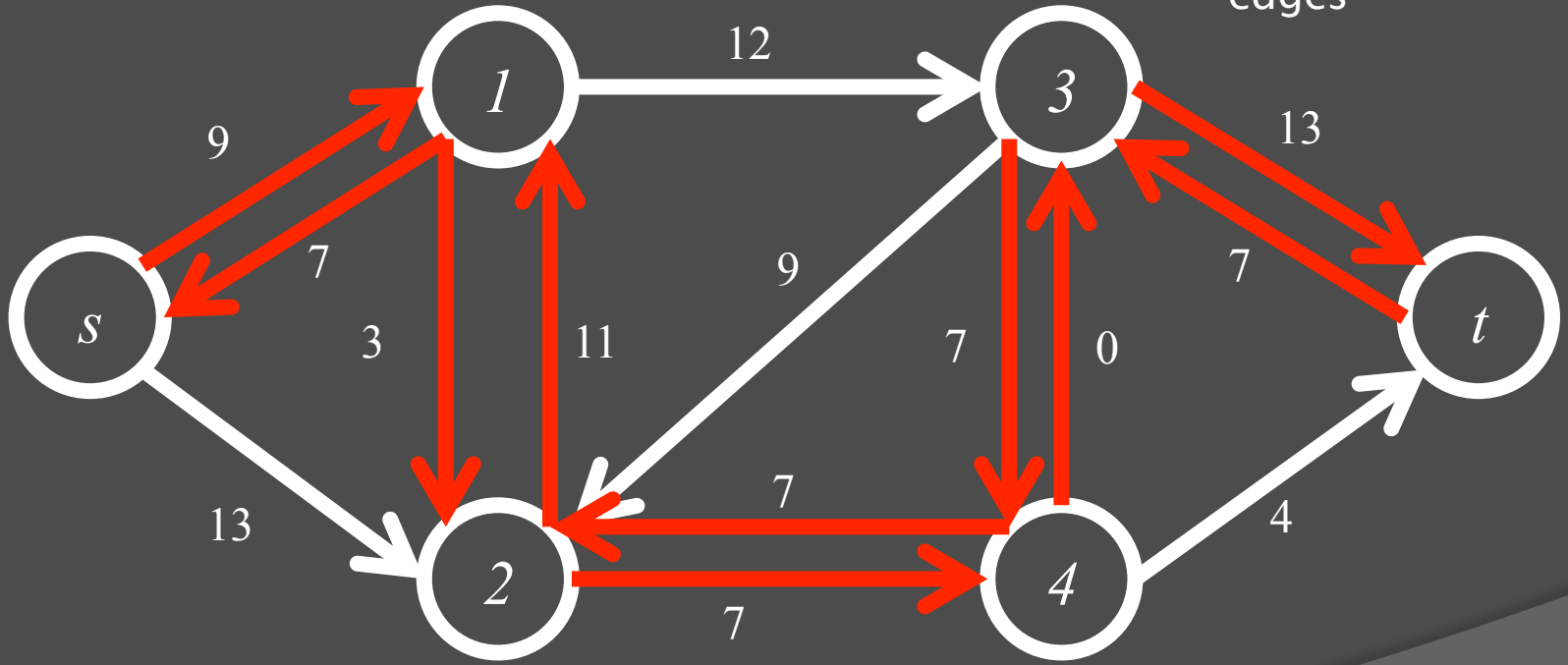
Flow: 0

Ford-Fulkerson Diagram



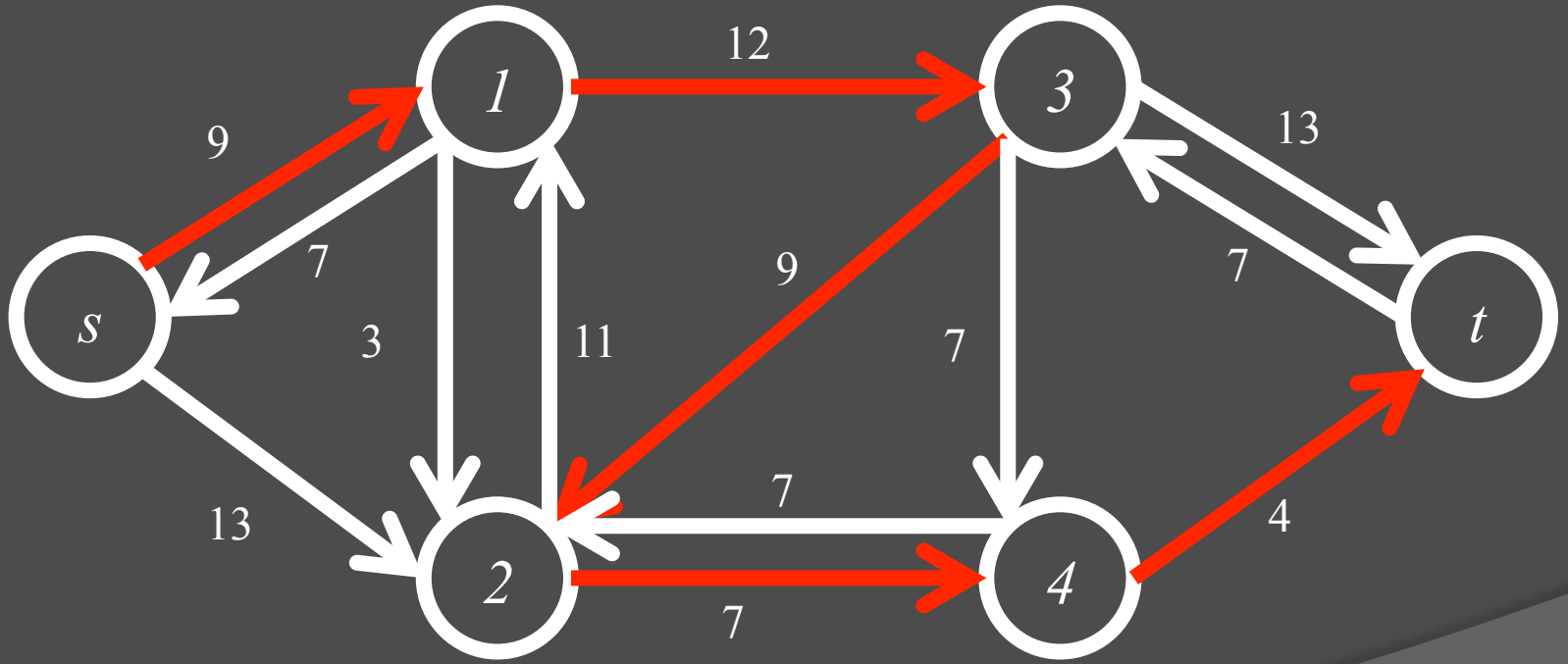
Ford-Fulkerson Diagram

Take flow from reverse edges



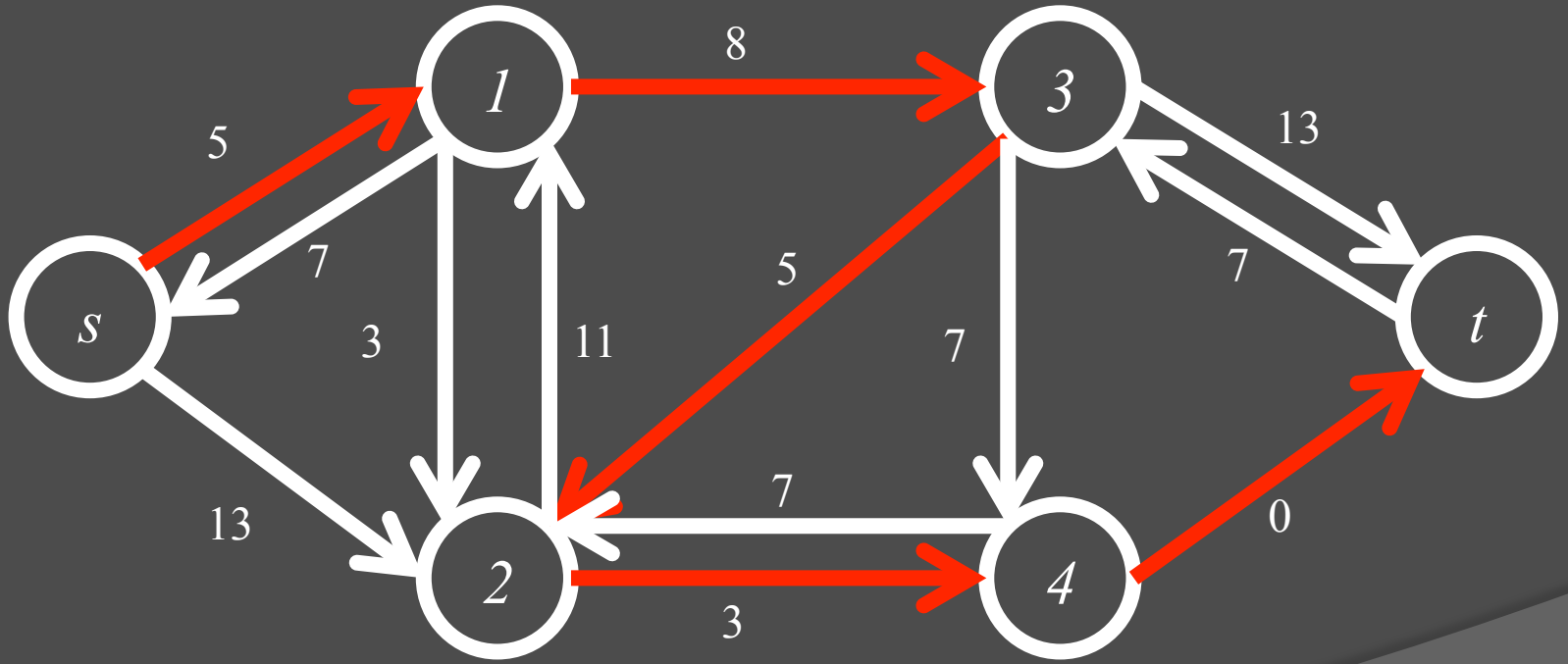
Flow: 7

Ford-Fulkerson Diagram



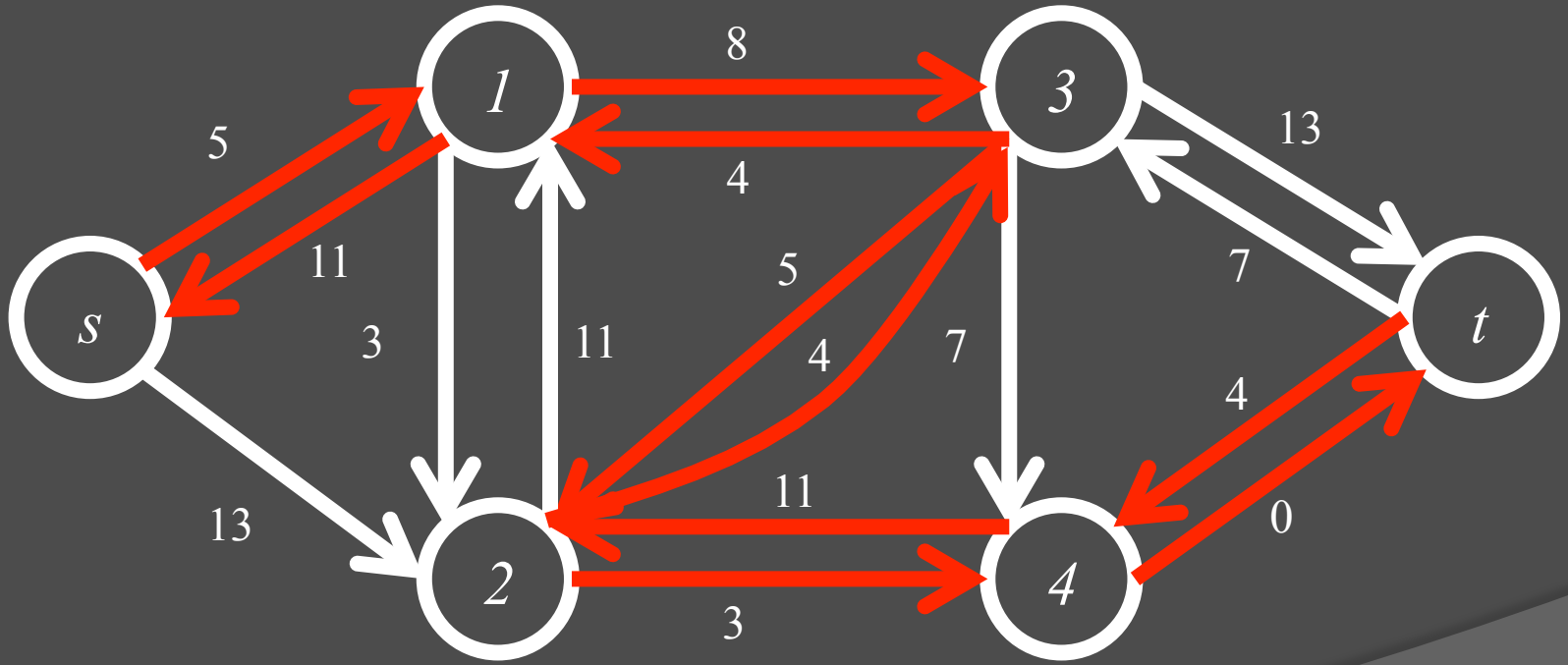
Flow: 7

Ford-Fulkerson Diagram



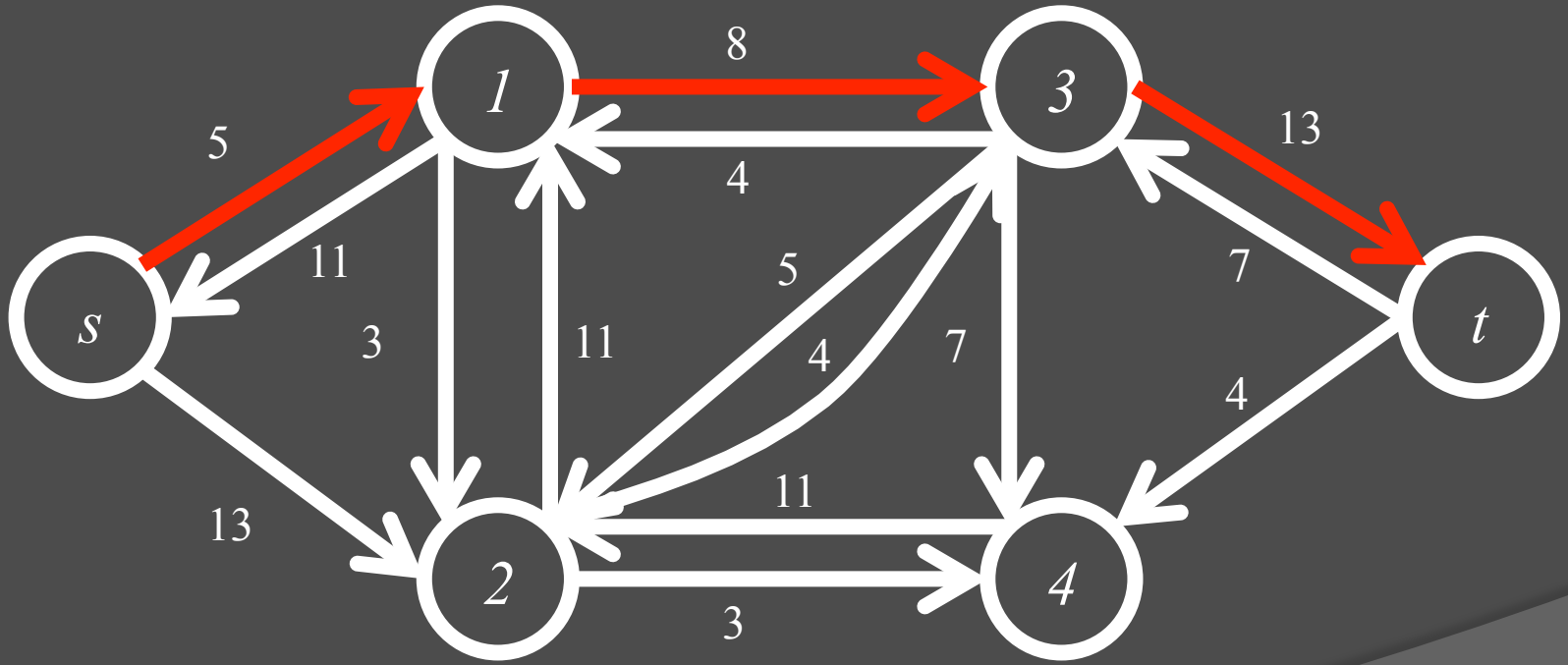
Flow: 11

Ford-Fulkerson Diagram



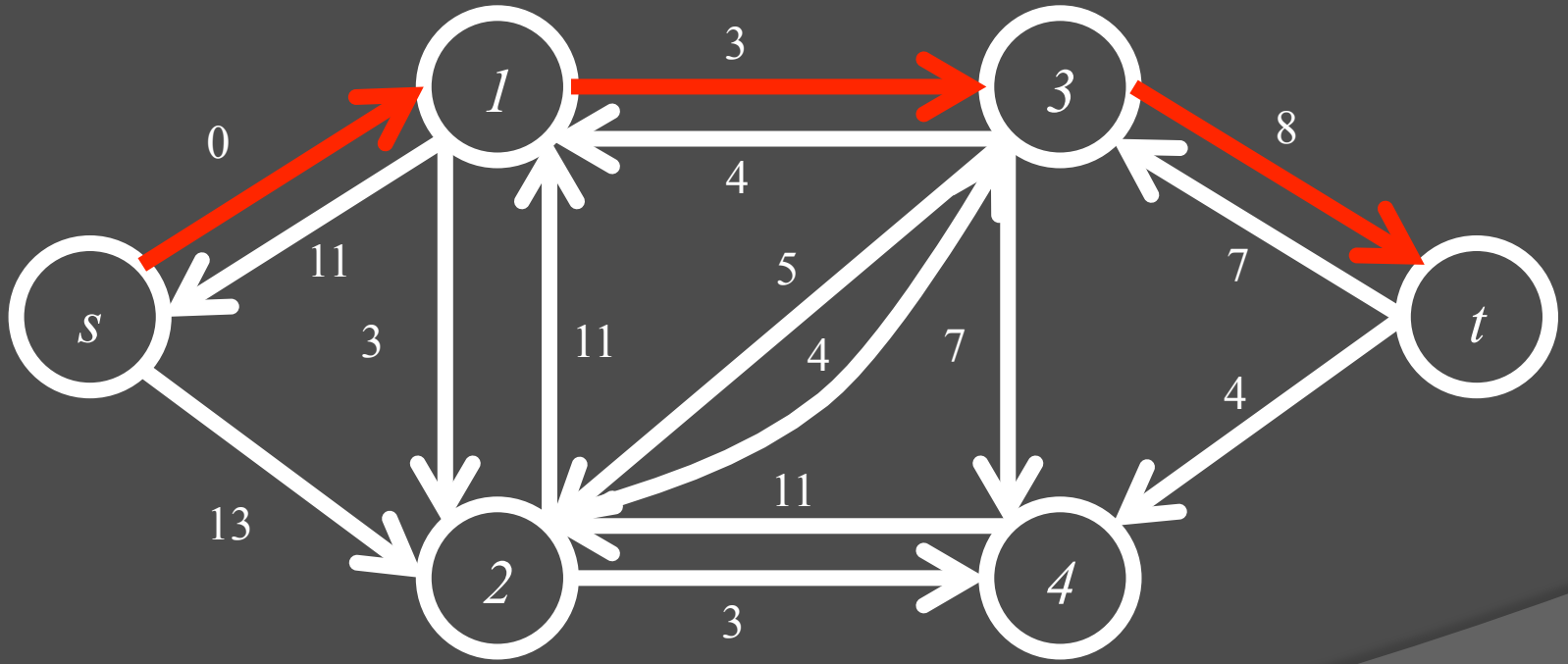
Flow: 11

Ford-Fulkerson Diagram



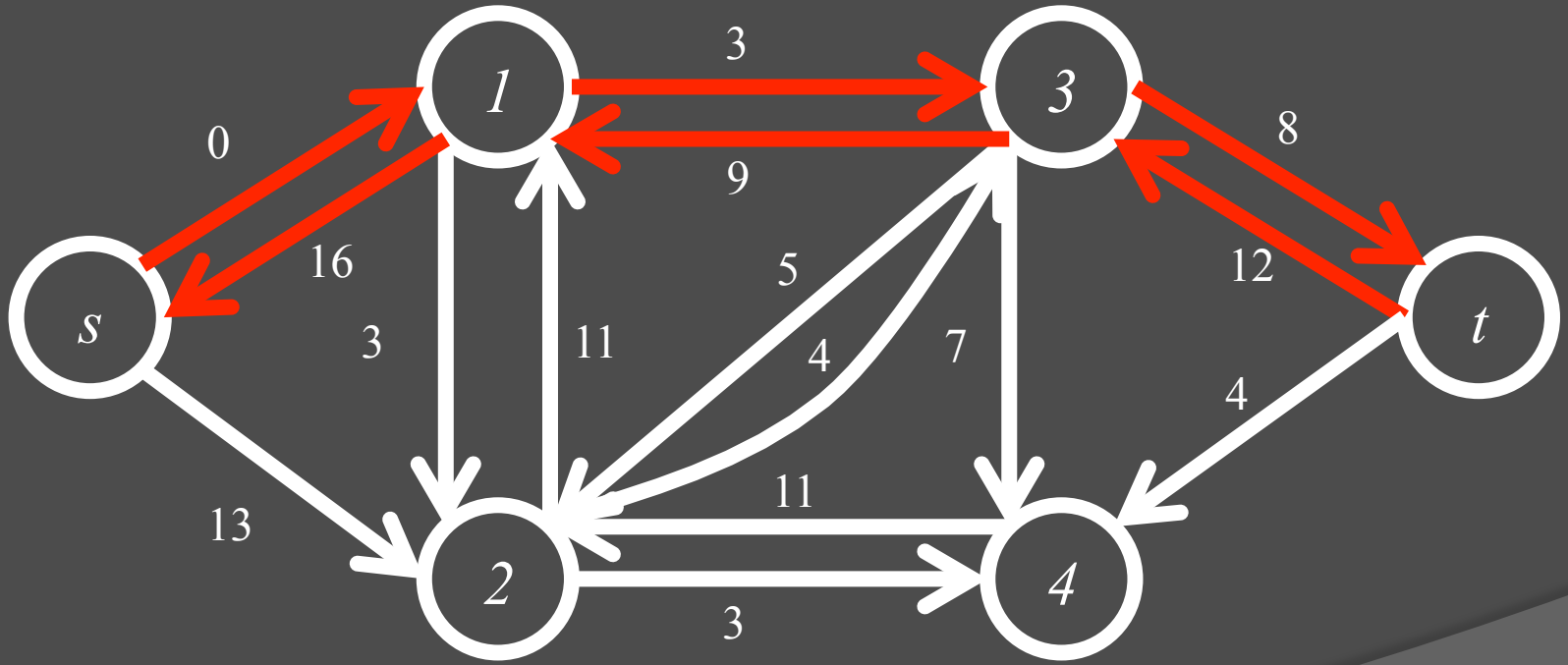
Flow: 11

Ford-Fulkerson Diagram



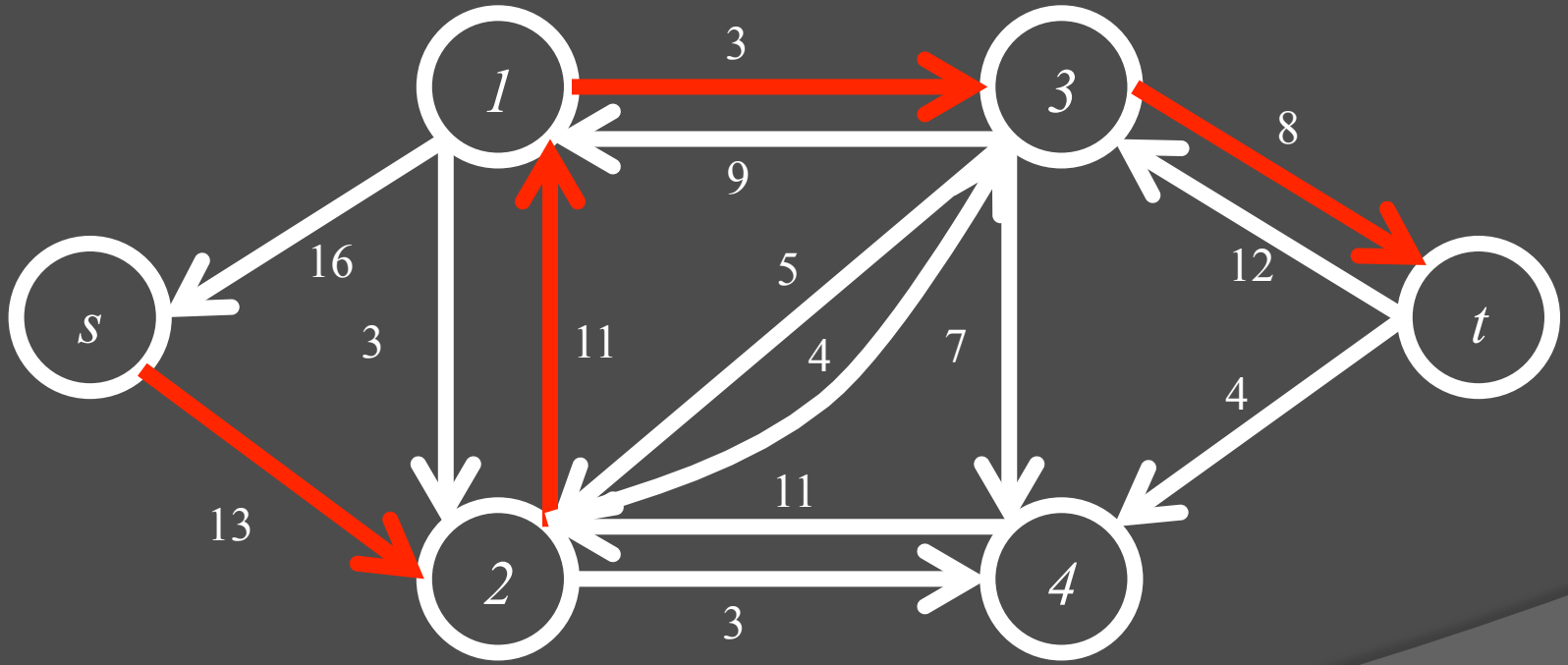
Flow: 16

Ford-Fulkerson Diagram



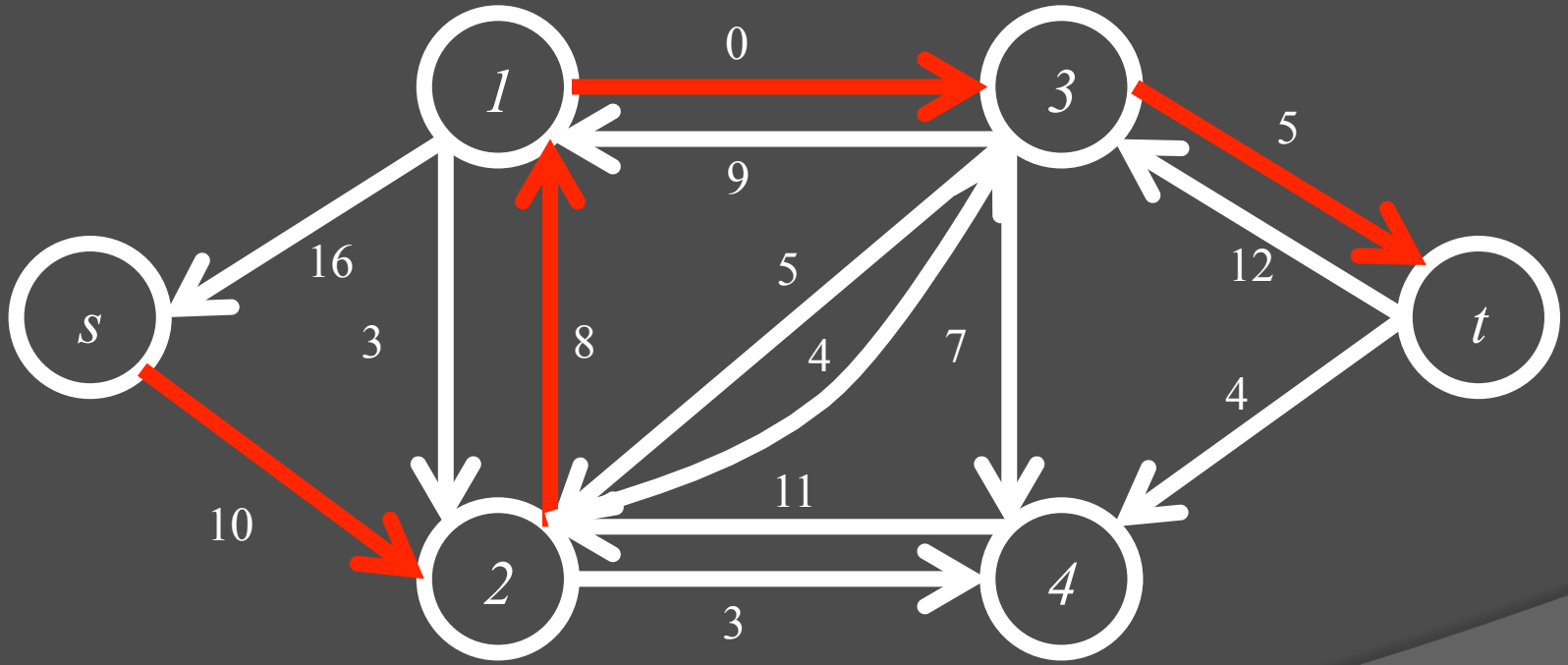
Flow: 16

Ford-Fulkerson Diagram



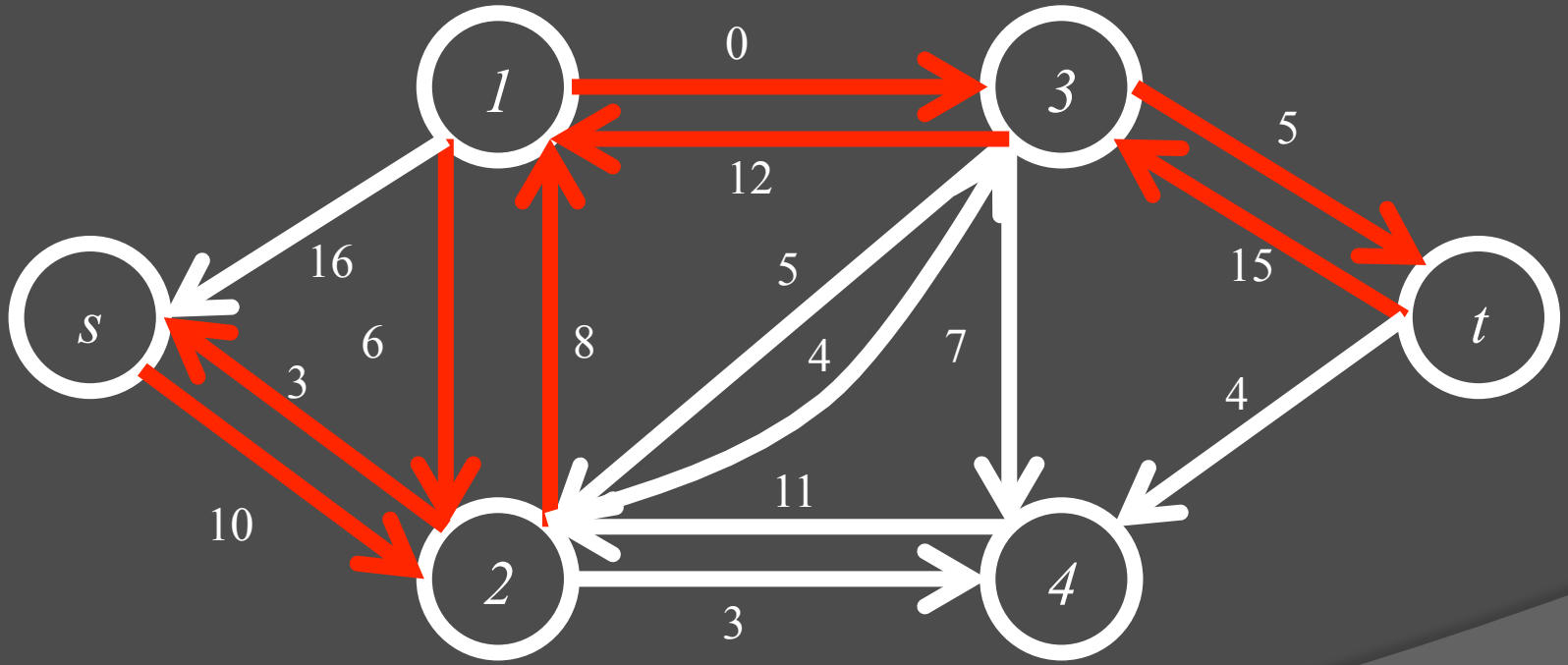
Flow: 16

Ford-Fulkerson Diagram



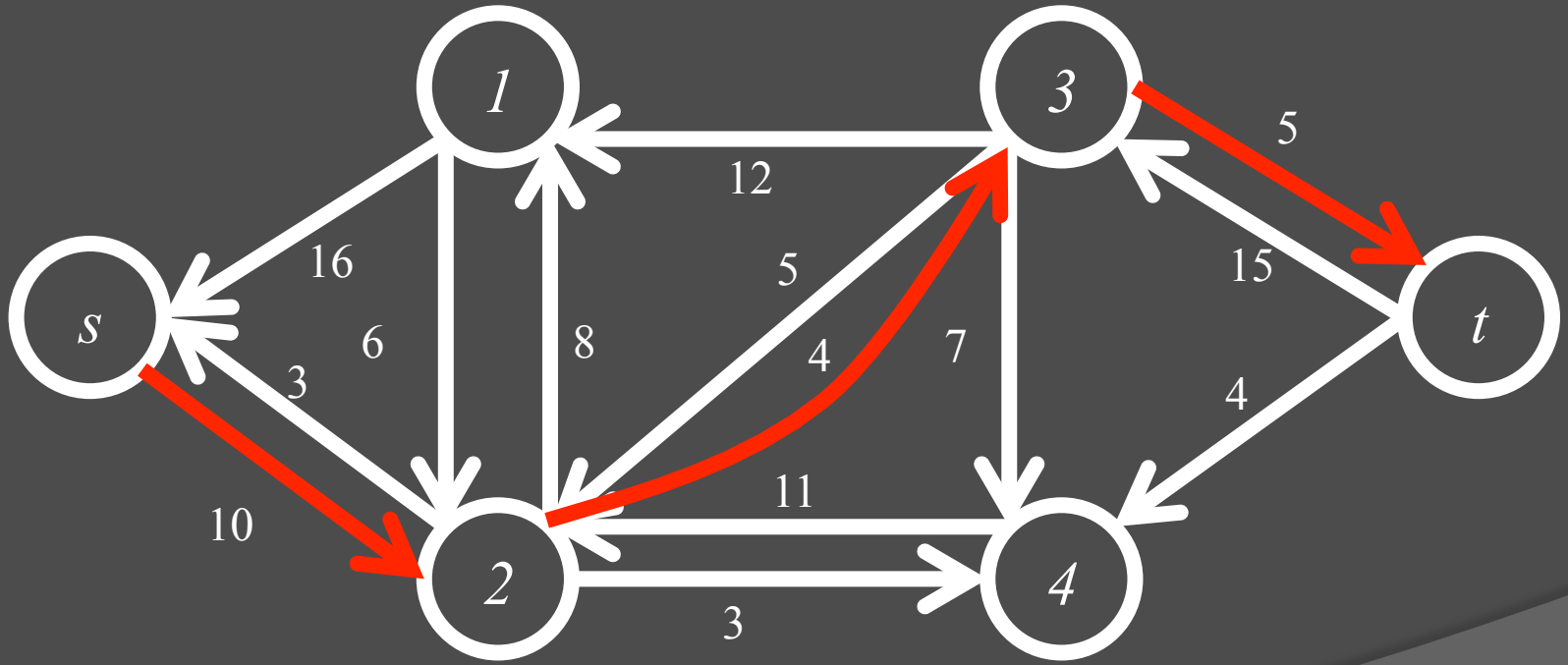
Flow: 19

Ford-Fulkerson Diagram



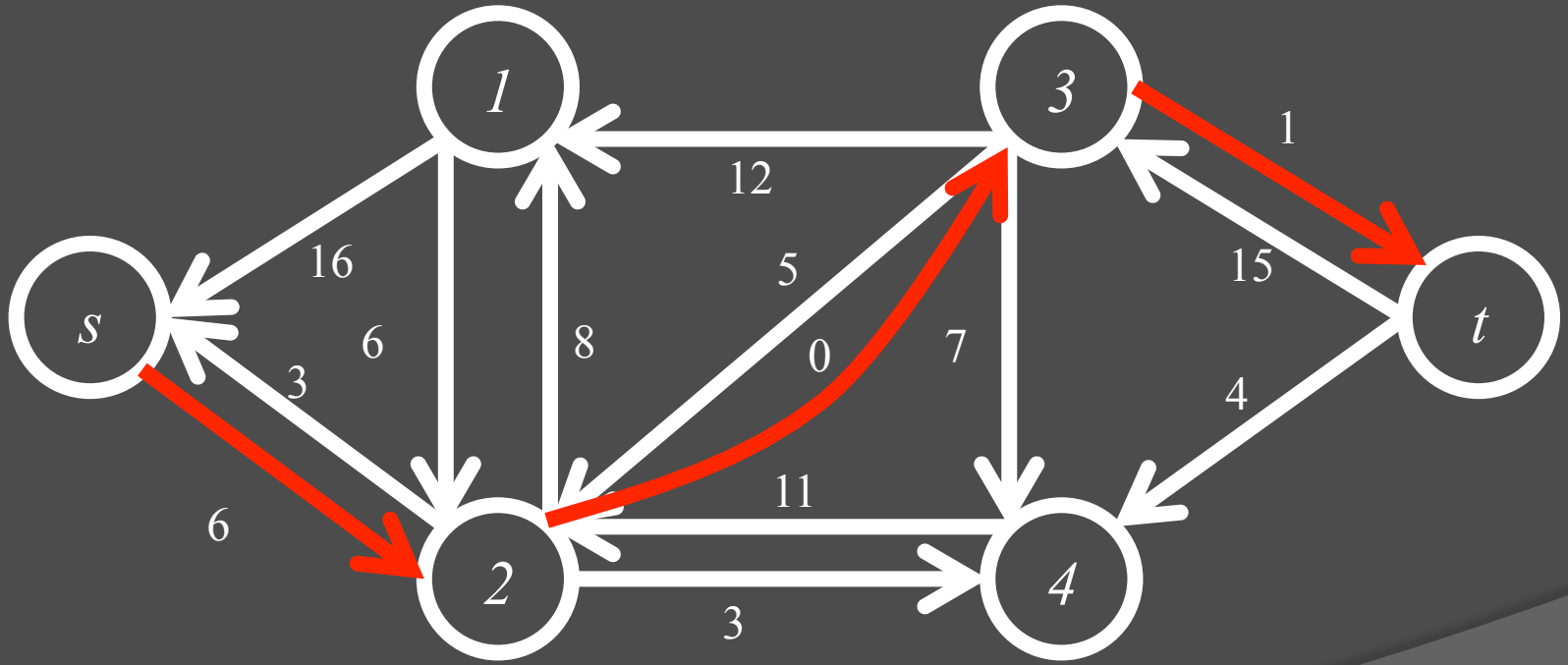
Flow: 19

Ford-Fulkerson Diagram



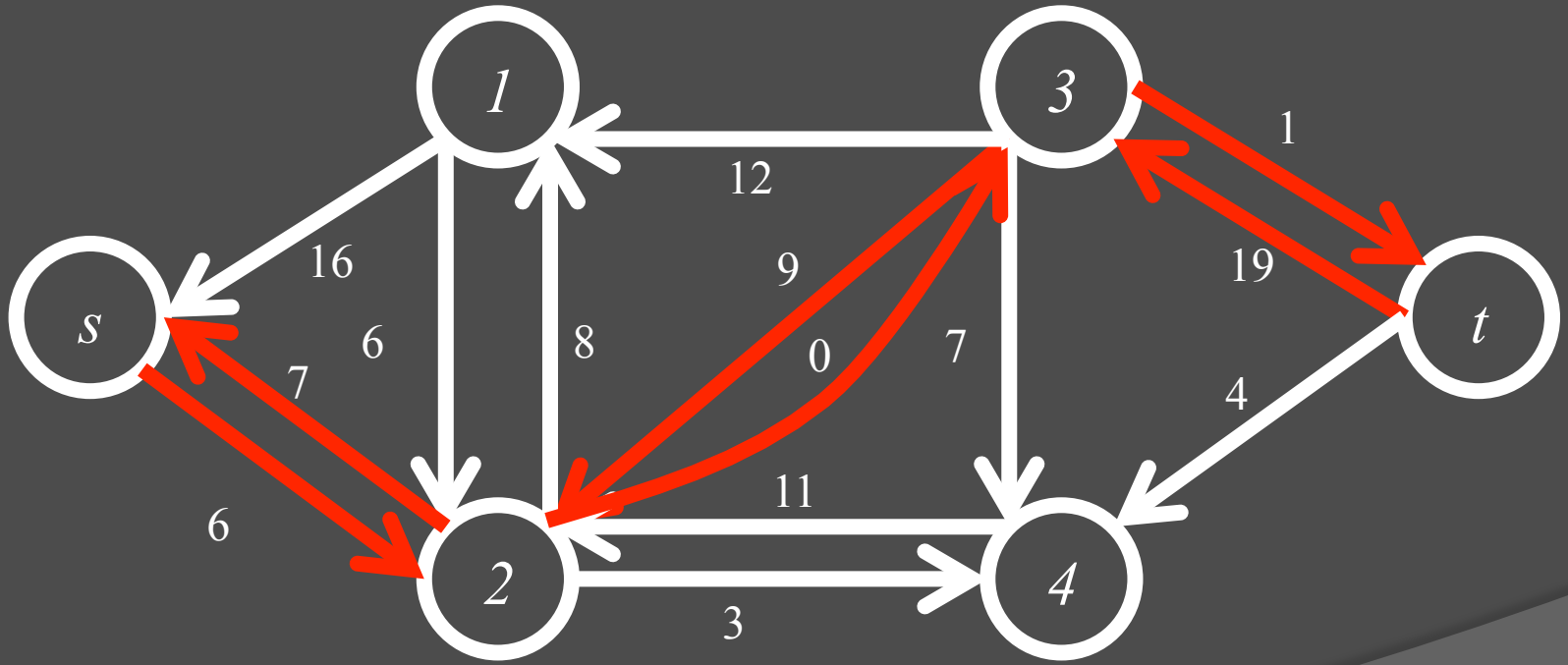
Flow: 19

Ford-Fulkerson Diagram



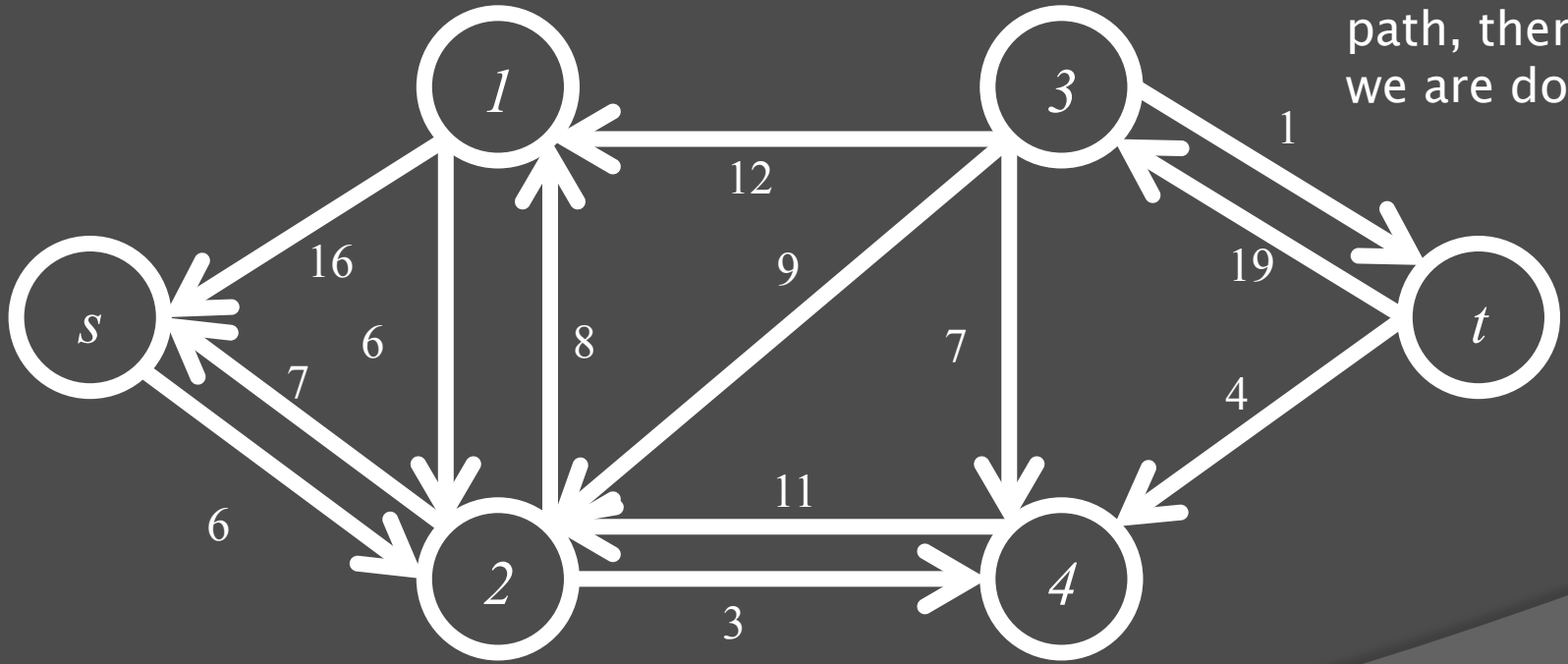
Flow: 23

Ford-Fulkerson Diagram



Flow: 23

Ford-Fulkerson Diagram

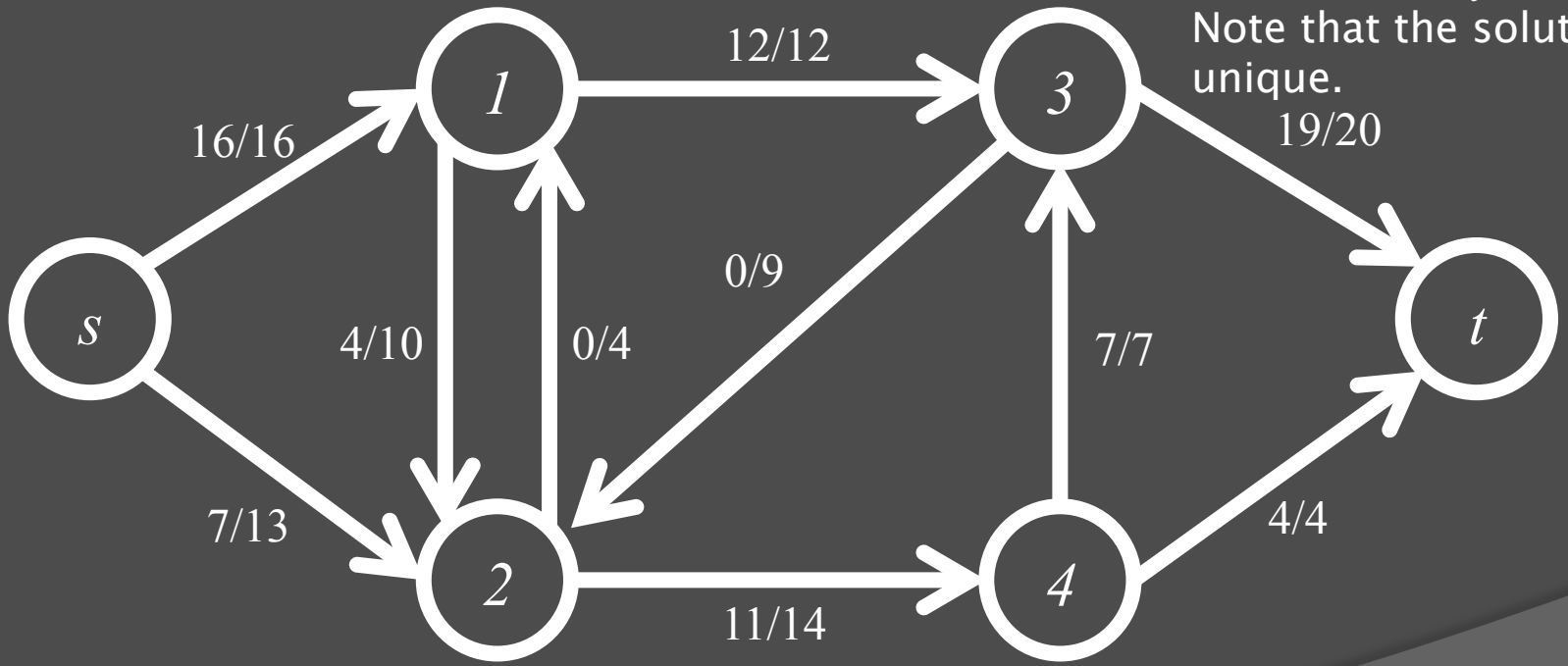


No more s-t path, therefore, we are done

Flow: 23

Ford-Fulkerson Diagram

The final network. Each edge is in the form *f/c* (flow/cap). Note that the solution is not unique.

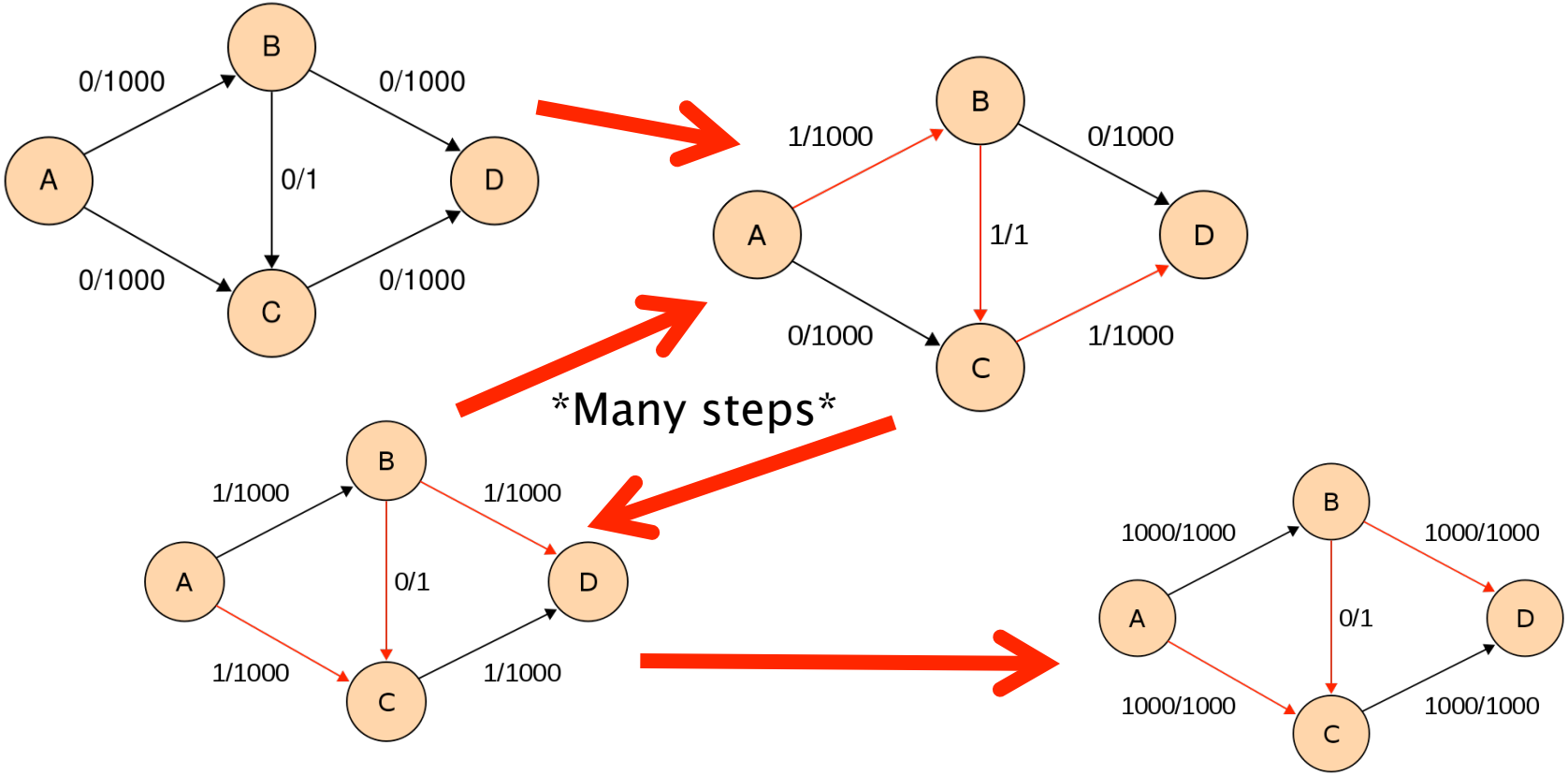


Flow: 23

Ford-Fulkerson Analysis

- Using a DFS to implement the Ford-Fulkerson method: $O(E)$ per loop run
- If all capacities are integers, the flow is increased by at least 1 each time
- Therefore on networks with integer weights, the total runtime is $O(Ef)$, where f is the maximum flow
- Ford-Fulkerson might never terminate if f is irrational

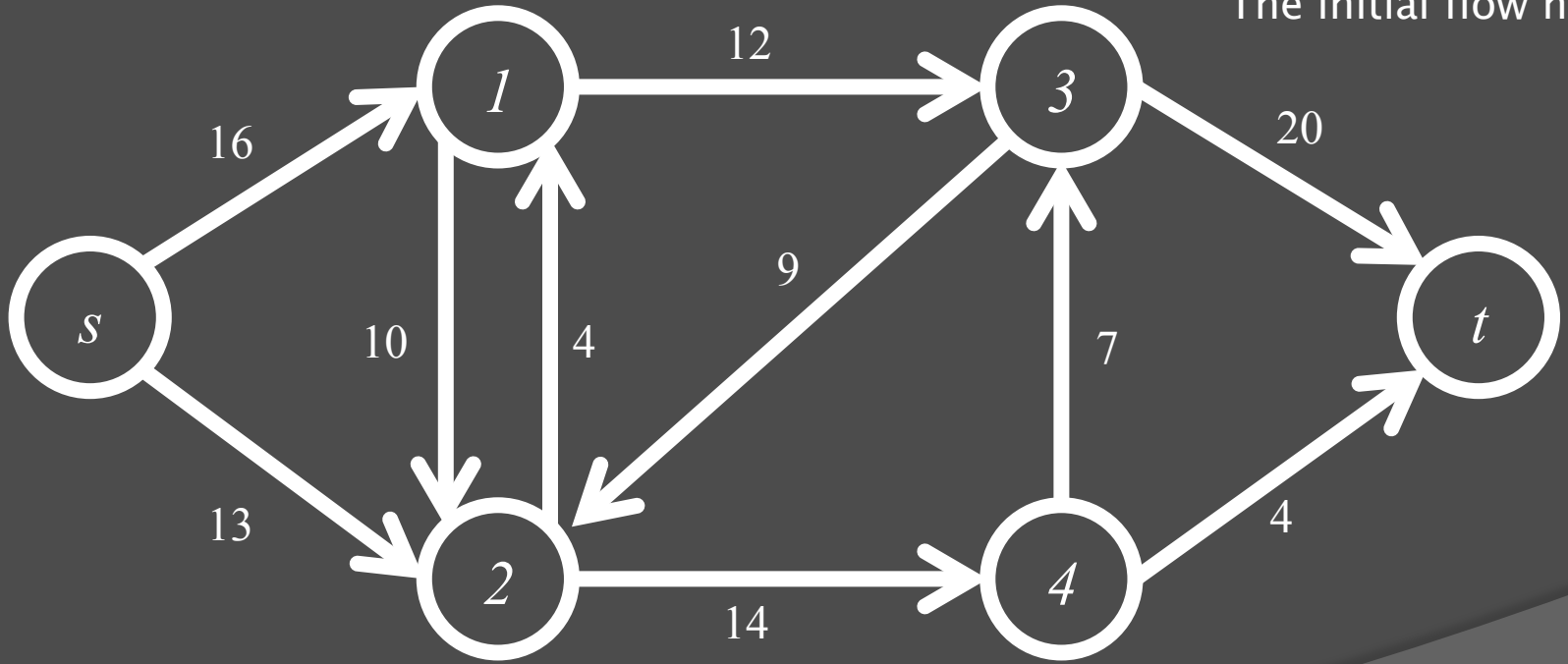
Ford-Fulkerson Worst Case



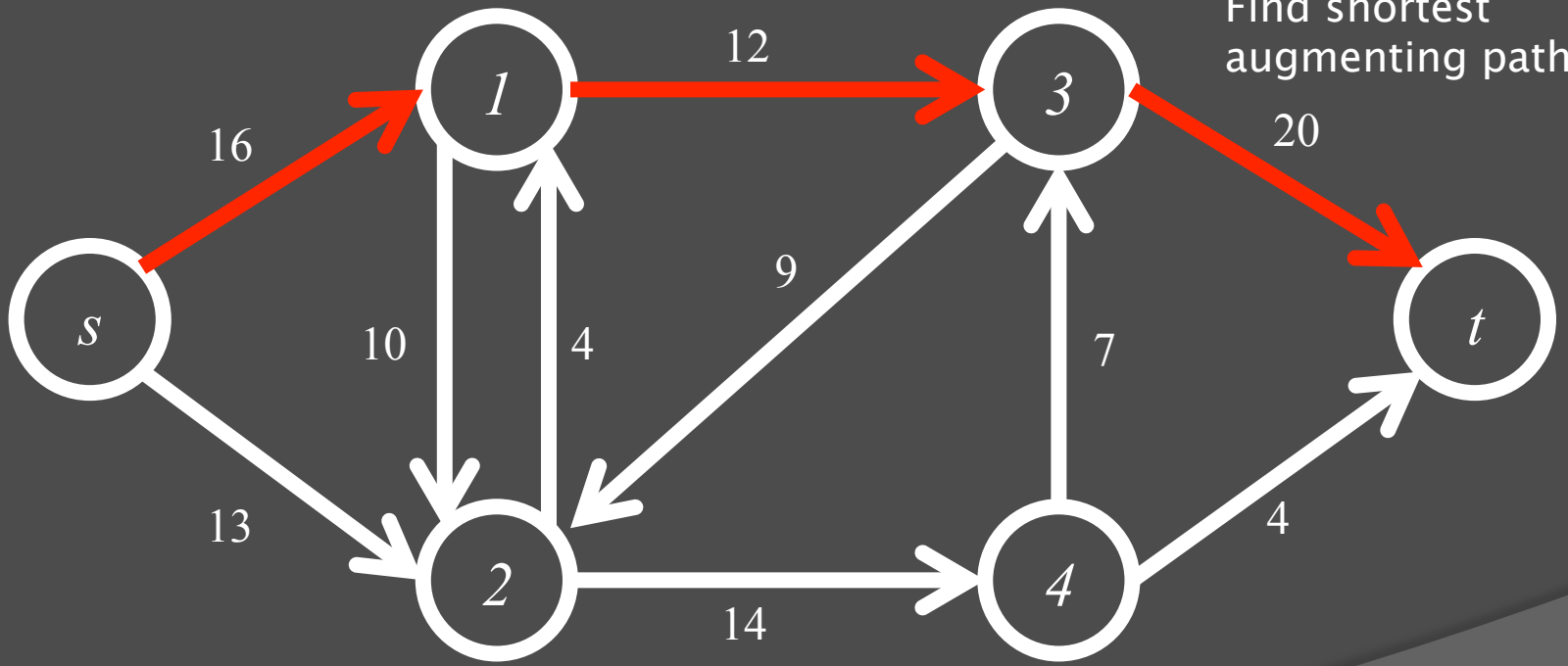
Edmonds-Karp Algorithm

- A variant of the Ford-Fulkerson method that is guaranteed to terminate
- Runtime is independent of the maximum flow value
- Instead of a DFS, we use a BFS instead
- Otherwise, Edmonds-Karp is exactly the same as Ford-Fulkerson

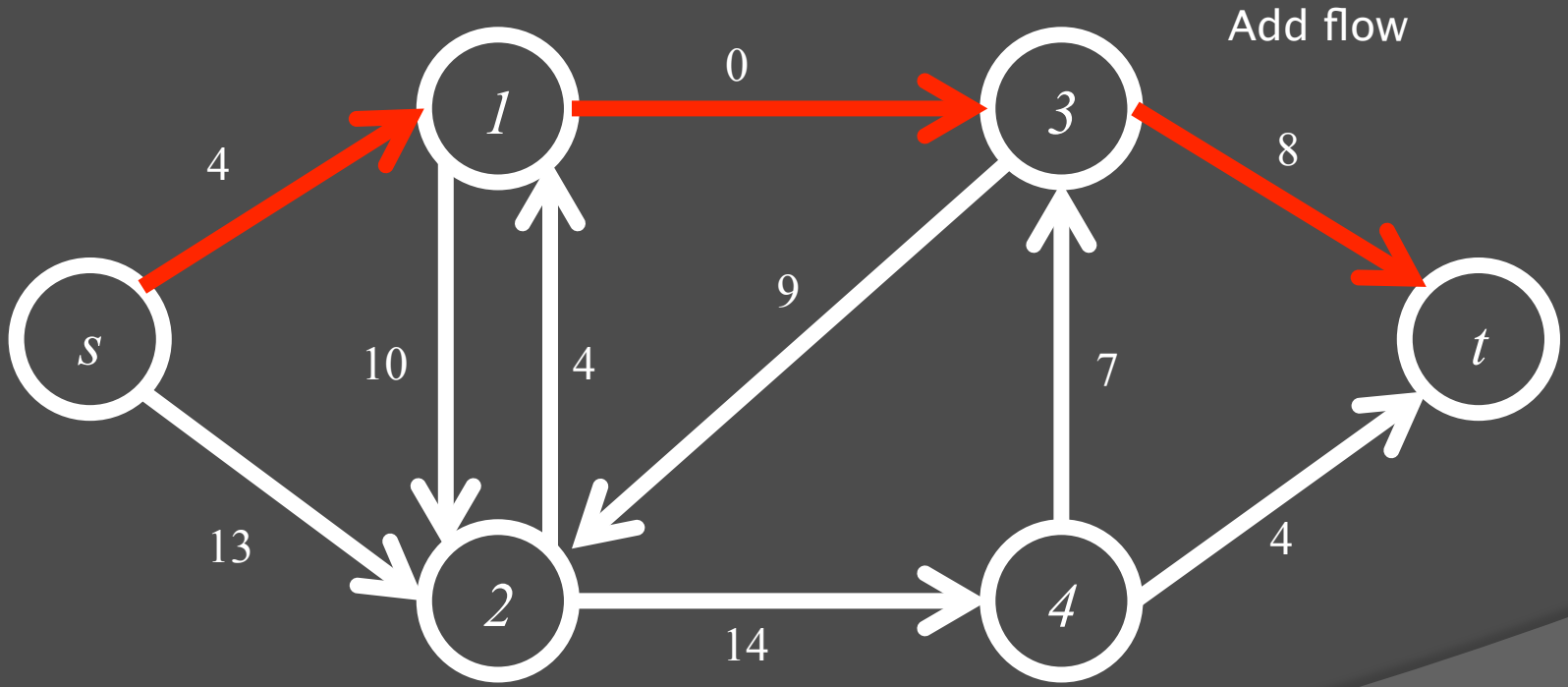
Edmonds-Karp Diagram



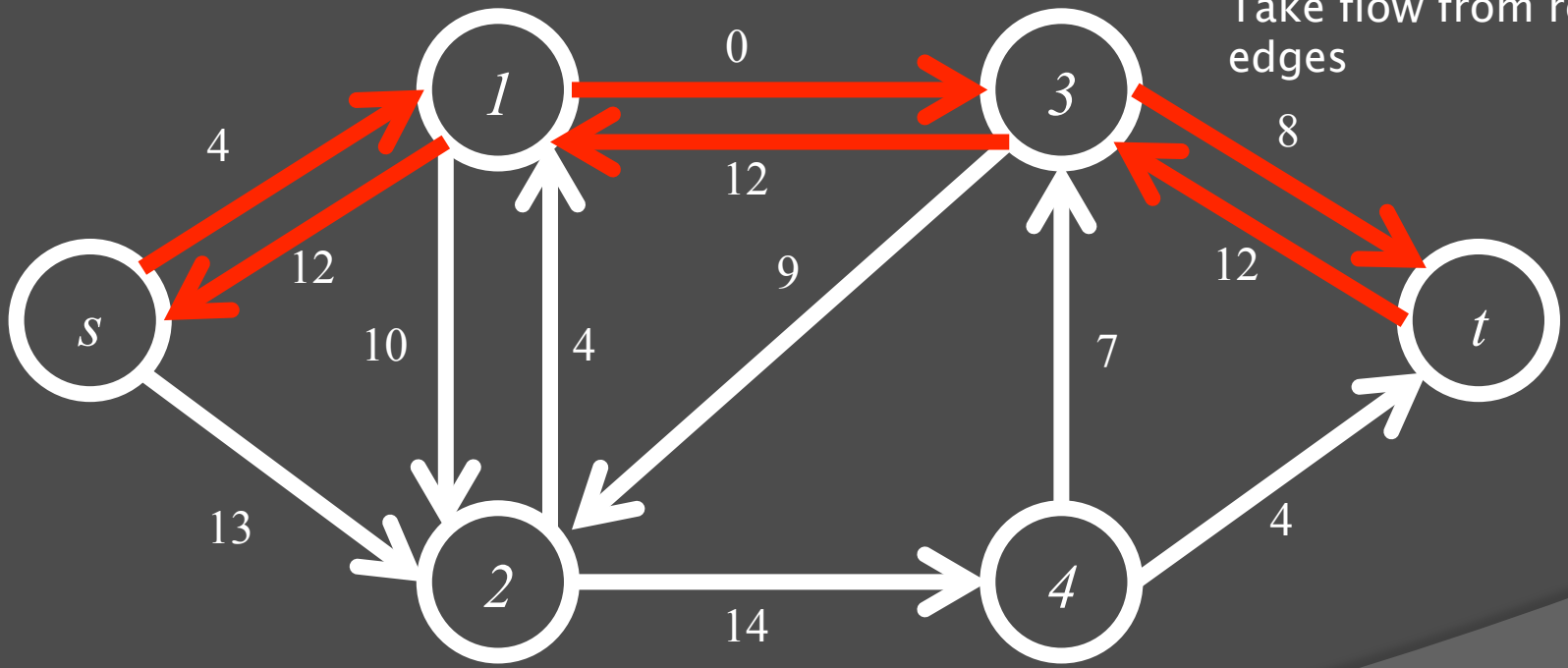
Edmonds-Karp Diagram



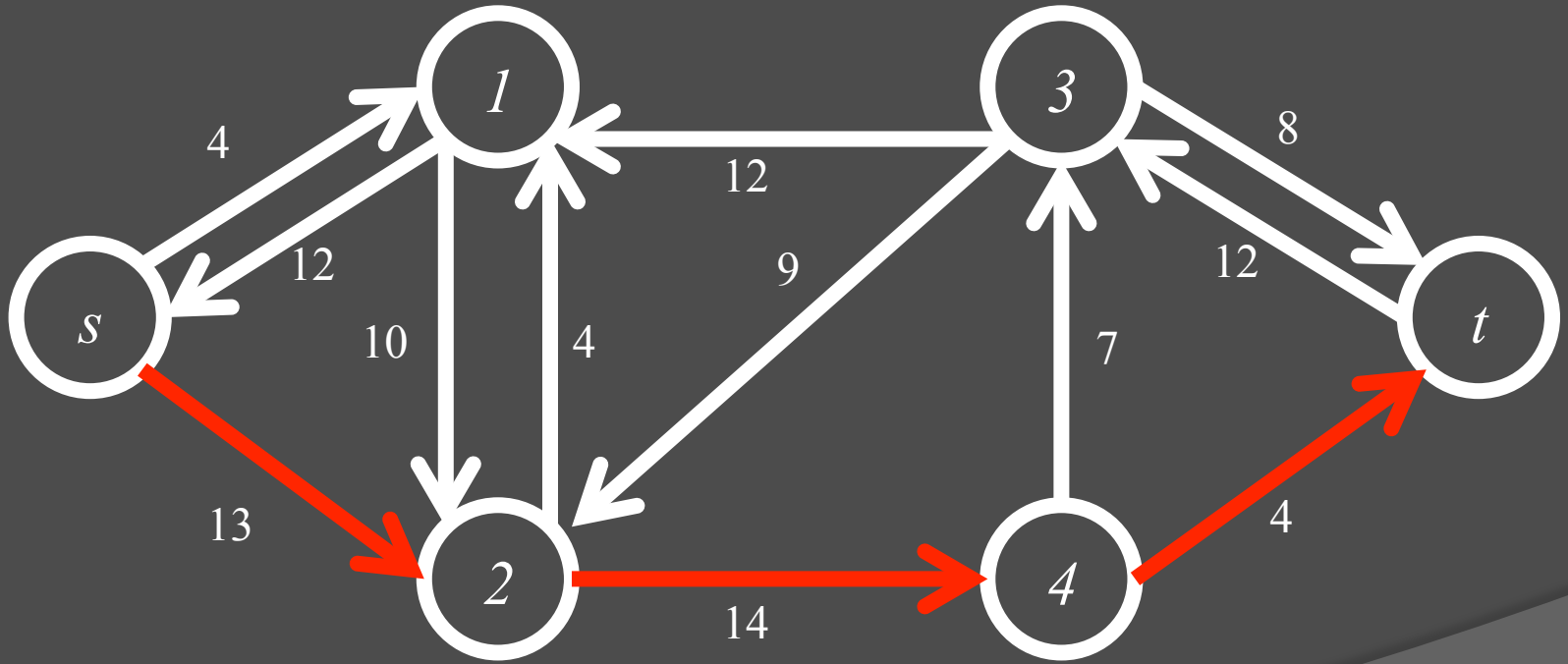
Edmonds-Karp Diagram



Edmonds-Karp Diagram

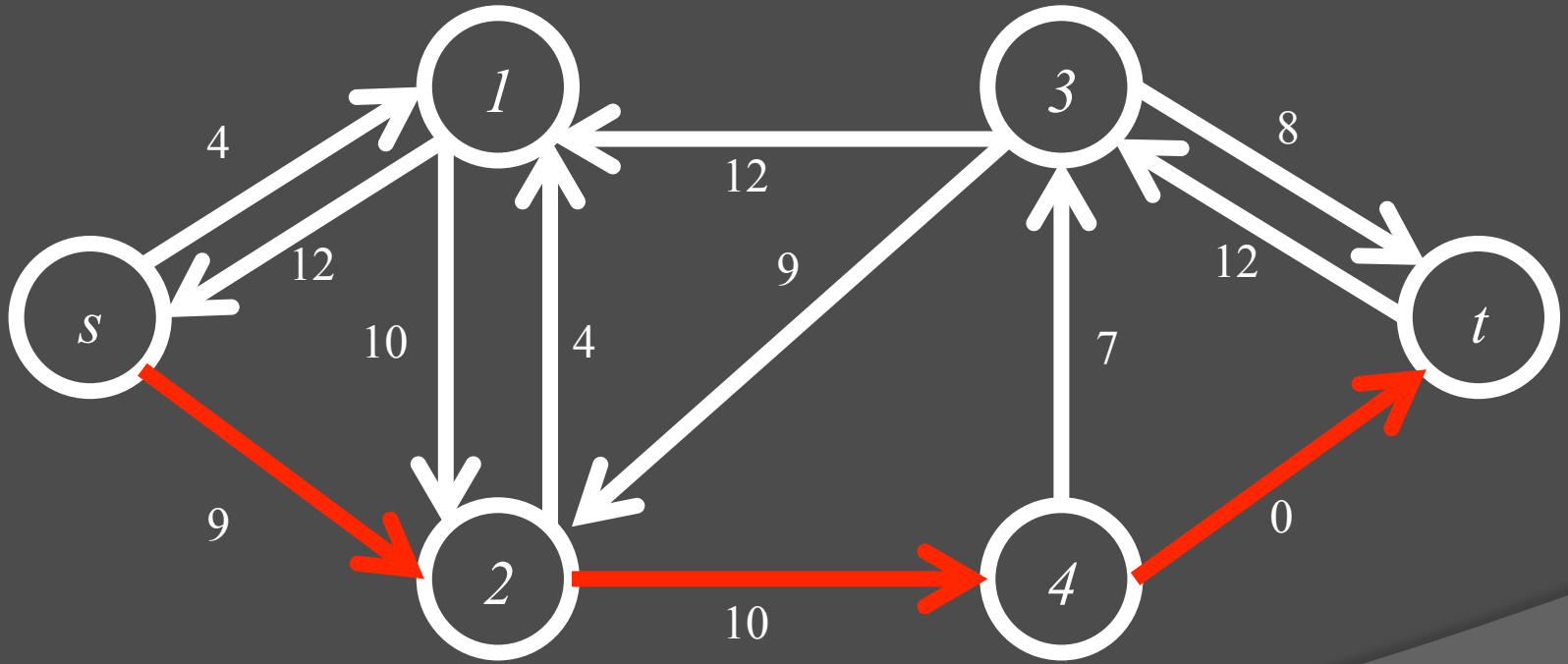


Edmonds-Karp Diagram



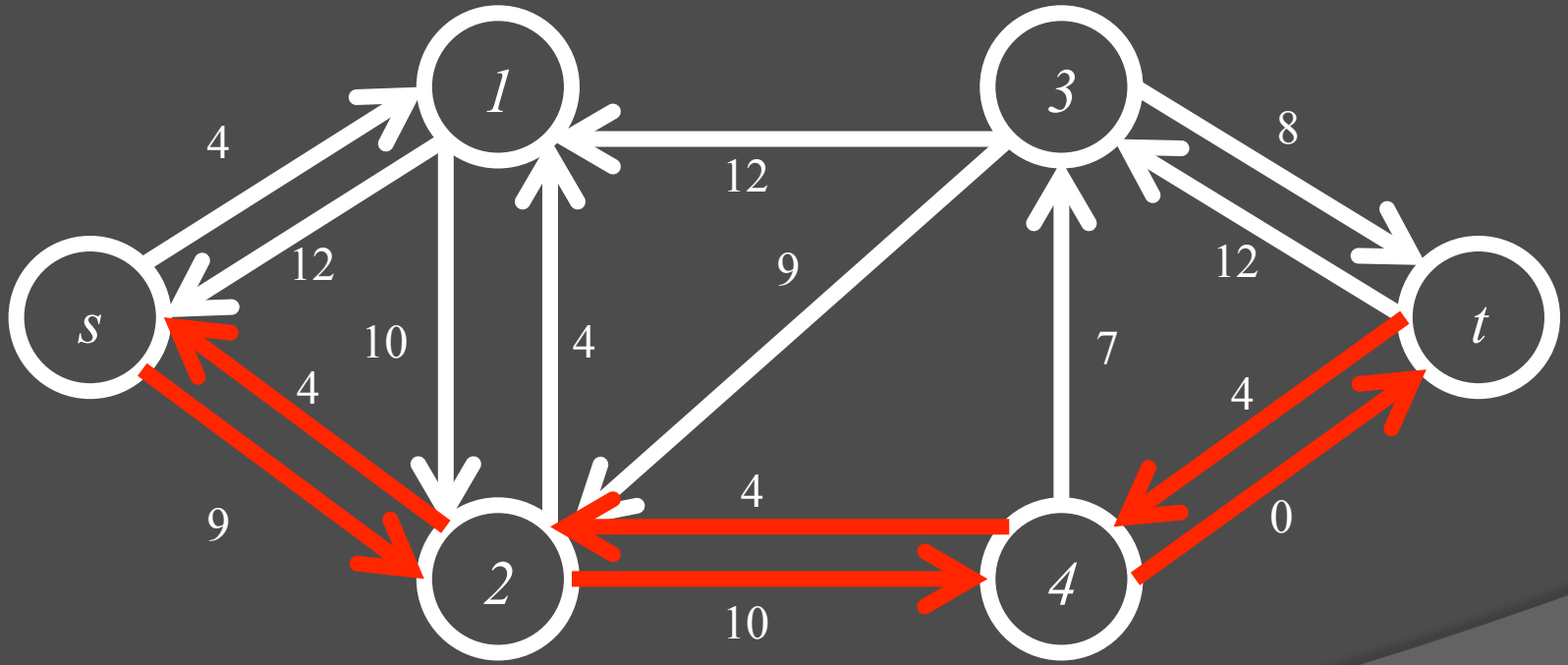
Flow: 12

Edmonds-Karp Diagram



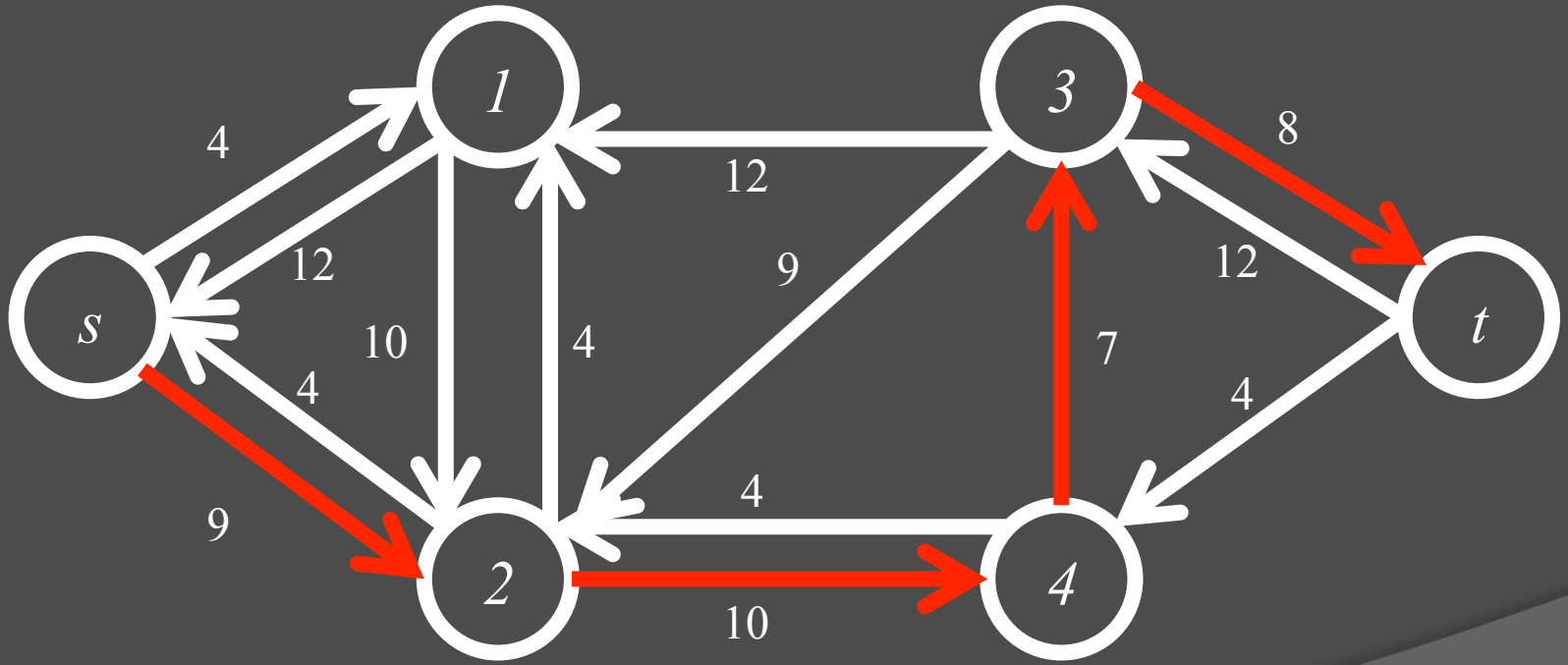
Flow: 16

Edmonds-Karp Diagram



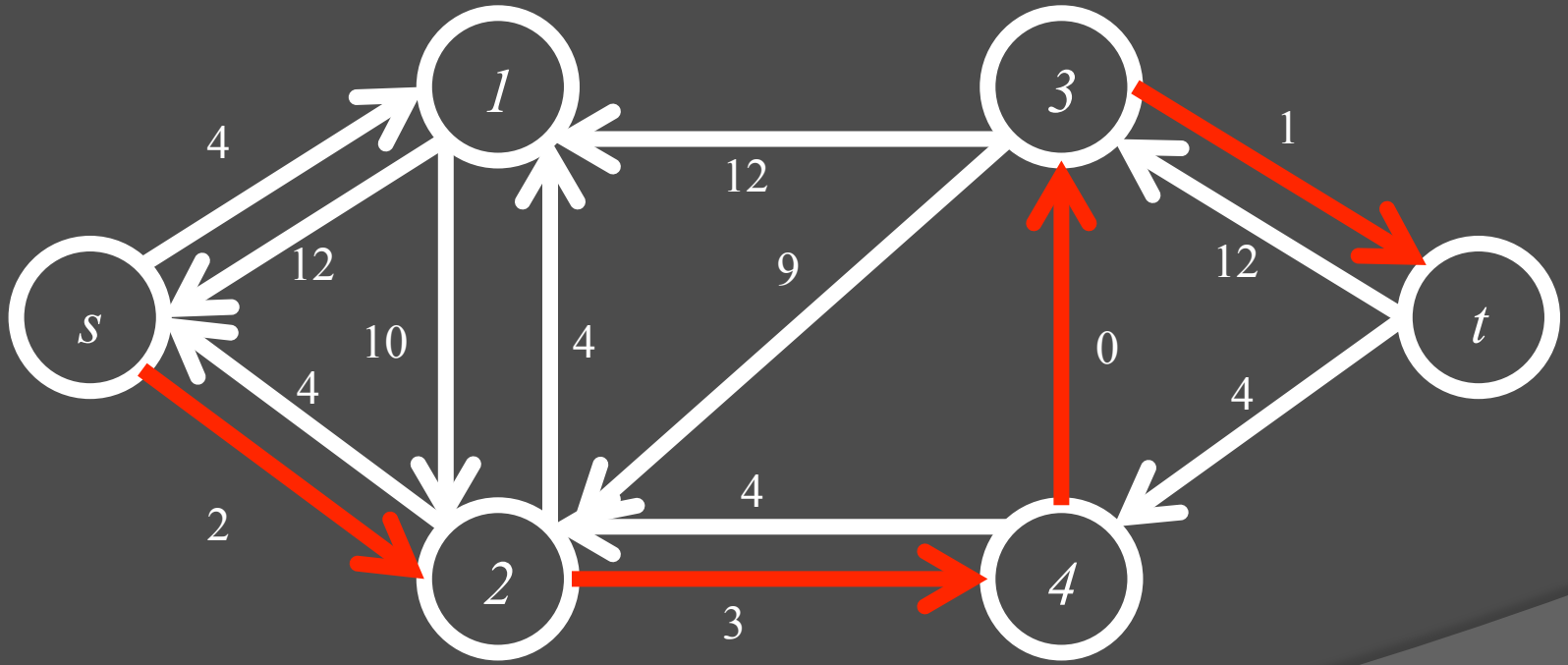
Flow: 16

Edmonds-Karp Diagram



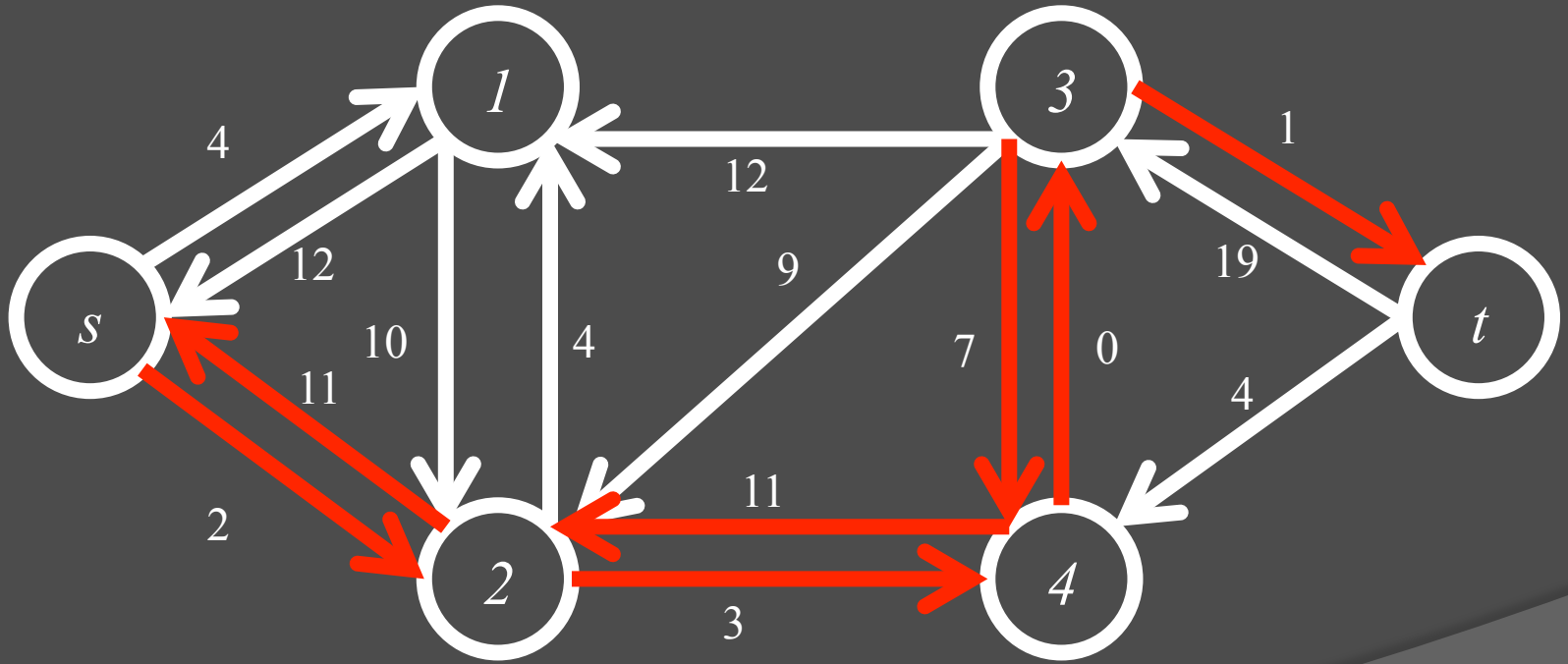
Flow: 16

Edmonds-Karp Diagram



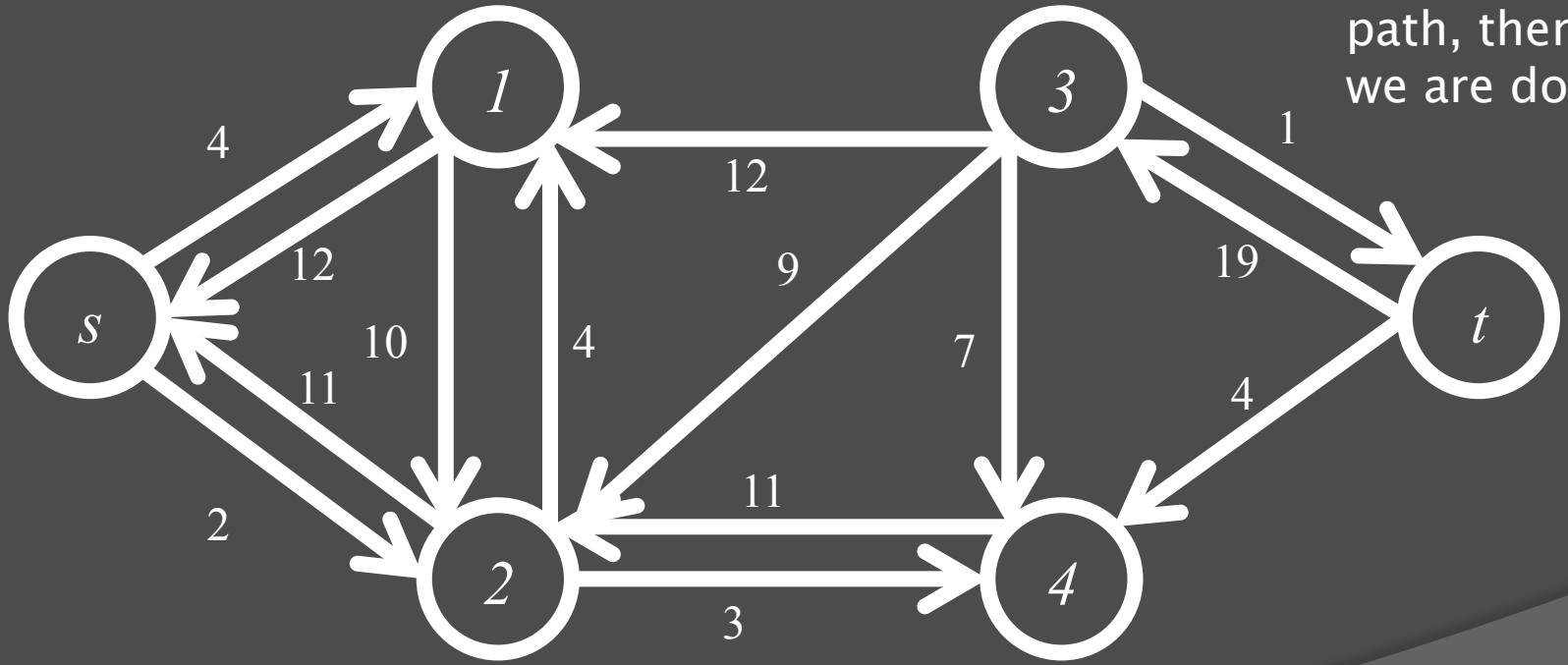
Flow: 23

Edmonds-Karp Diagram



Flow: 23

Edmonds-Karp Diagram

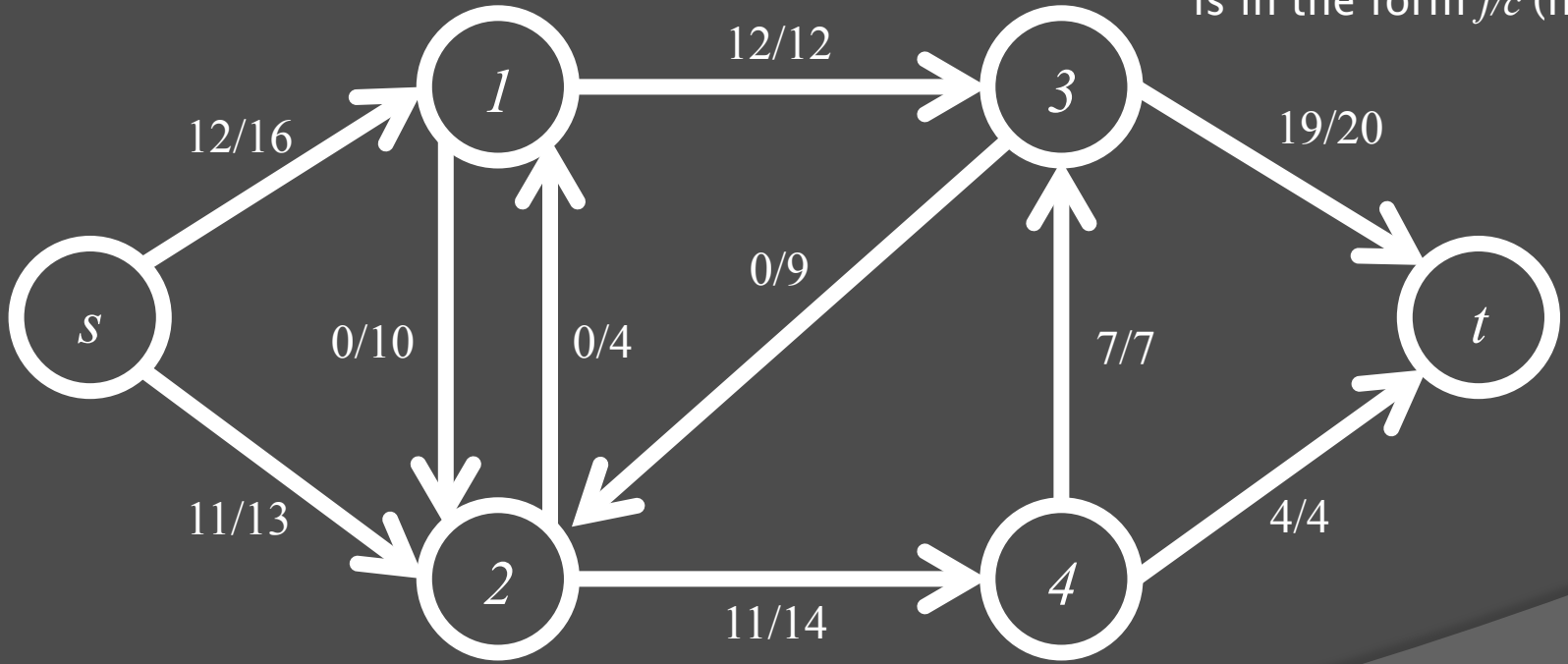


No more s-t path, therefore, we are done

Flow: 23

Edmonds-Karp Diagram

The final network. Each edge is in the form f/c (flow/cap).



Flow: 23

Edmonds-Karp Analysis

- Each loop runs in $O(E)$ time (BFS)
- Every time we run the BFS, at least 1 edge's flow in the network is equal to its capacity (**saturated edge**)
- The distance from the source to this edge must be greater than the last time it was saturated
- The maximum length of a path is V
- At worst case, each edge is saturated $O(V)$ times
- Therefore, we need to the BFS run at most $O(VE)$ times
- Final runtime: $O(VE^2)$

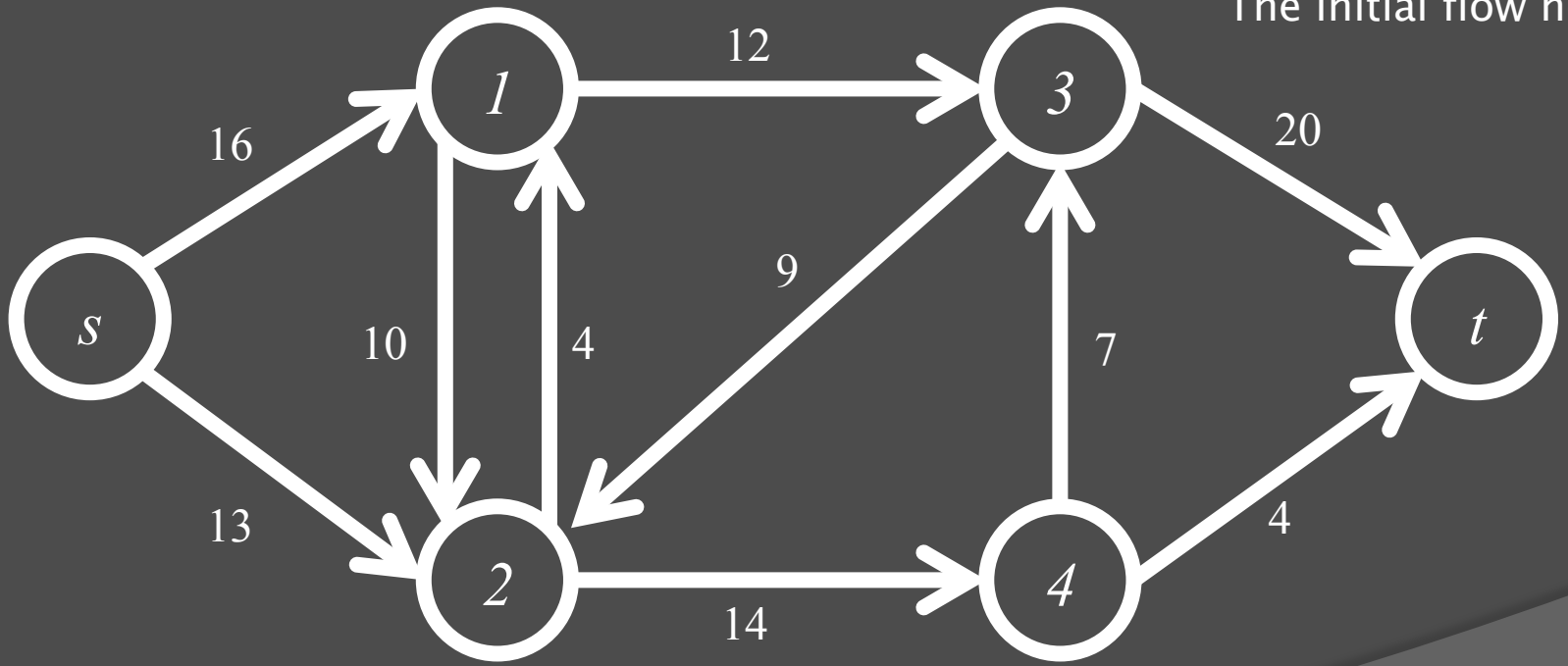
Dinic's Algorithm

- Uses residual graph, **level graph**, and **blocking flows** to compute the maximum flow
- A node's level is the shortest distance from the source to that node (use BFS)
- **Level graph** is the graph containing only edges *(u,v)* where *level(u) = level(v)-1*
  - Levels of augmenting path nodes should be *0, 1, 2, 3,...*
- A **blocking flow** is a flow such that no more flow can be added to a network

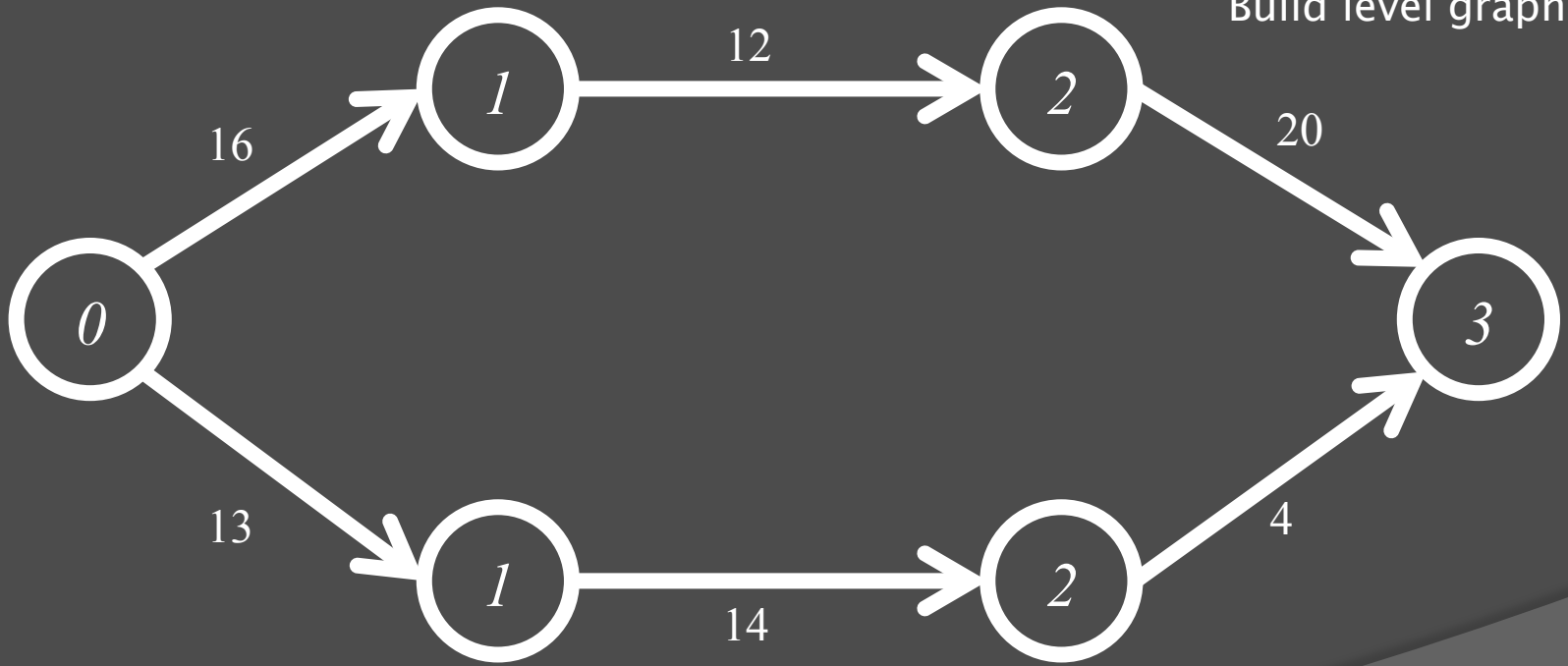
Dinic's Algorithm

- While true:
  - Construct the level graph (BFS)
  - If sink's level is infinite, terminate
  - Create a blocking flow on the level graph using DFS:
    - If the current node is the sink:
      - Find the earliest edge with minimal residual capacity
      - Increase the flow of all edges by this amount
      - Continue to run the DFS from the node just before this edge
    - Always go to a node with a level one greater than the current node

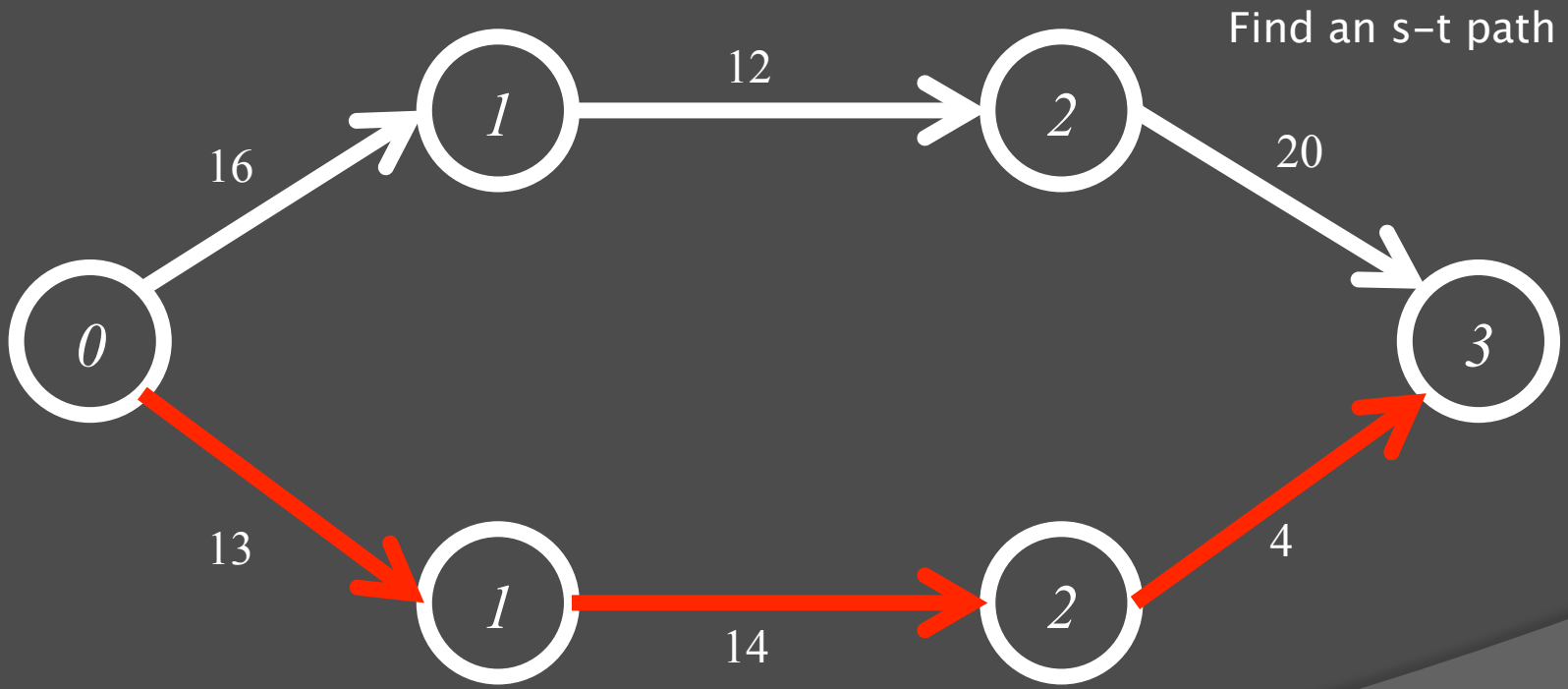
Dinic Diagram



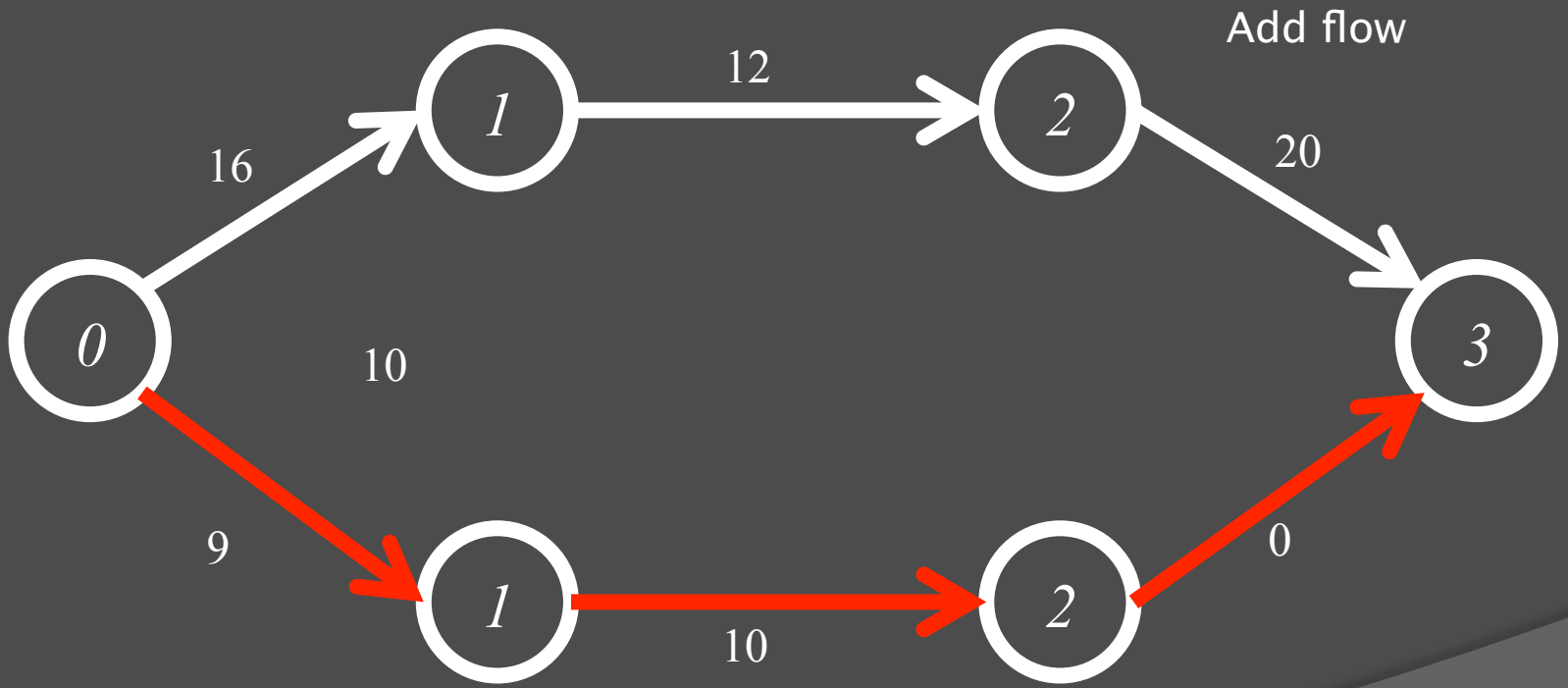
Dinic Diagram



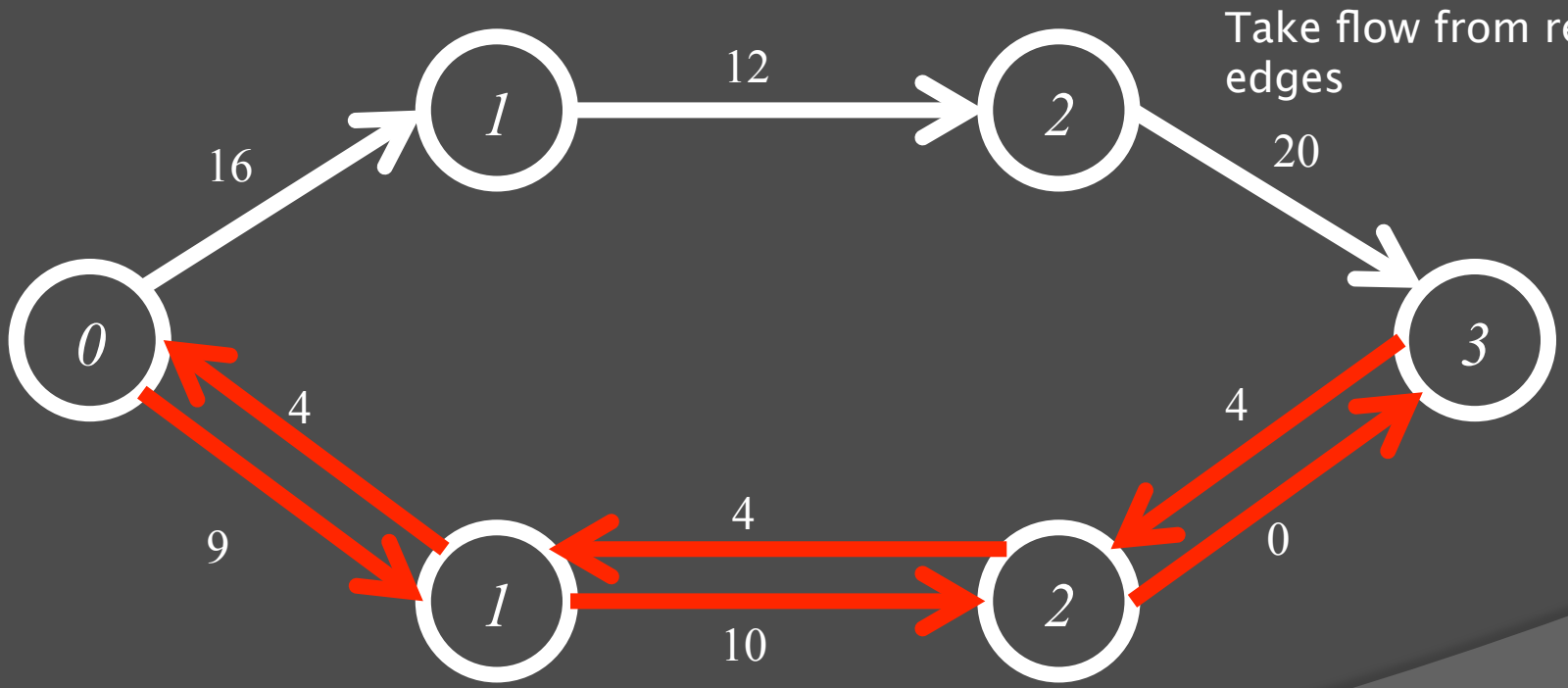
Dinic Diagram



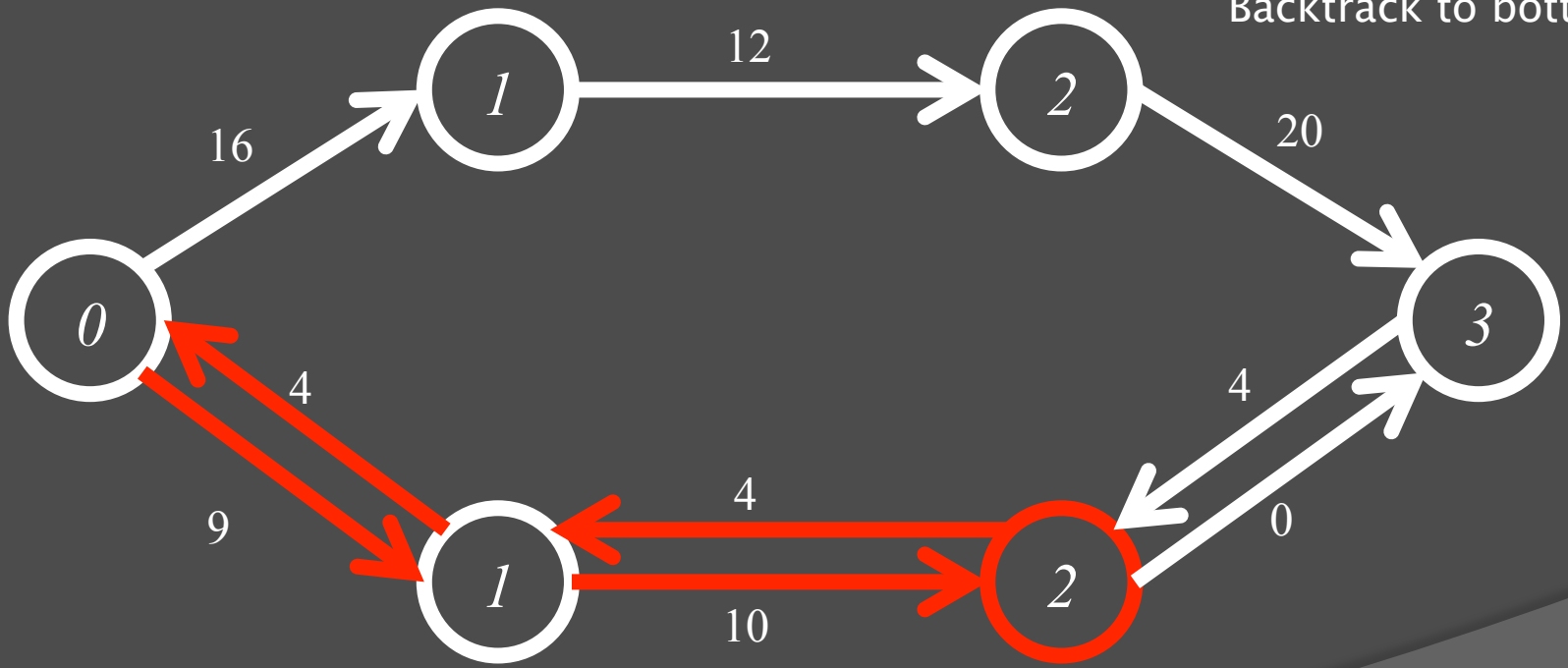
Dinic Diagram



Dinic Diagram

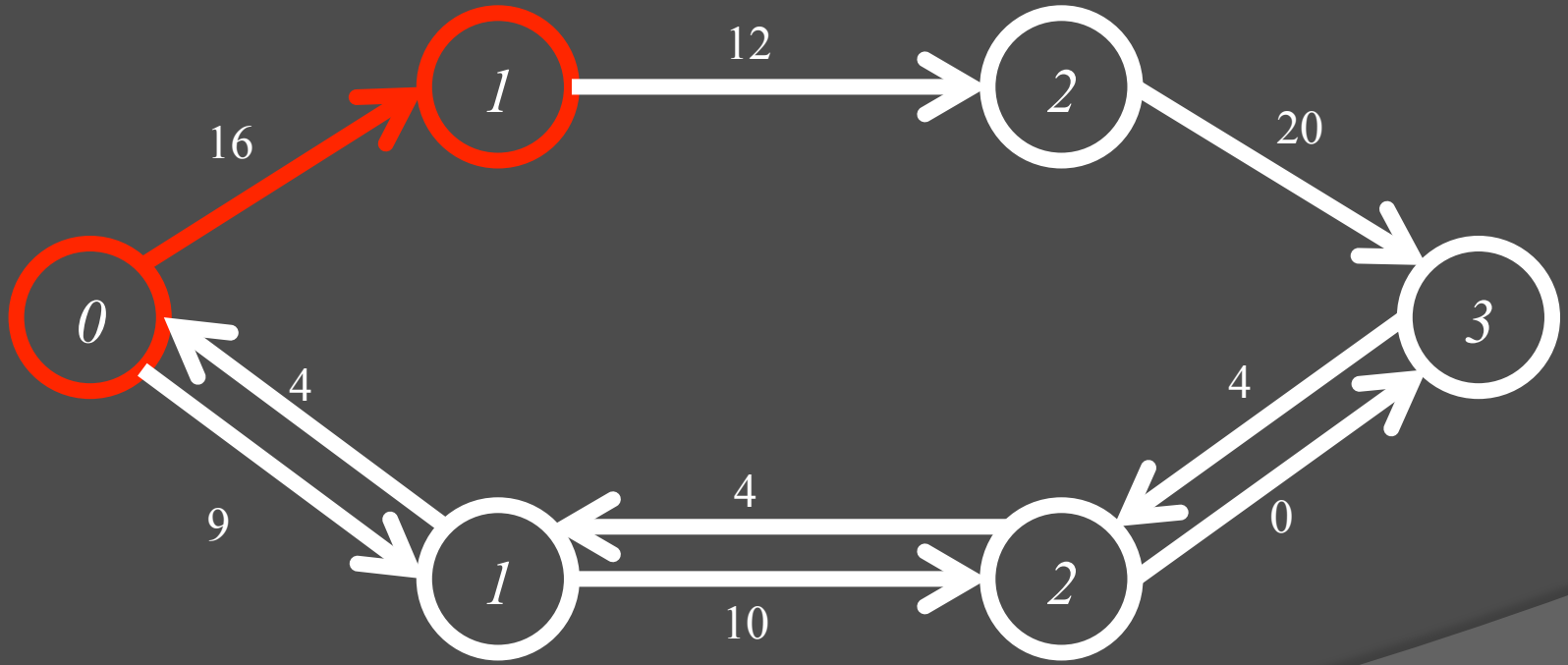


Dinic Diagram



Dinic Diagram

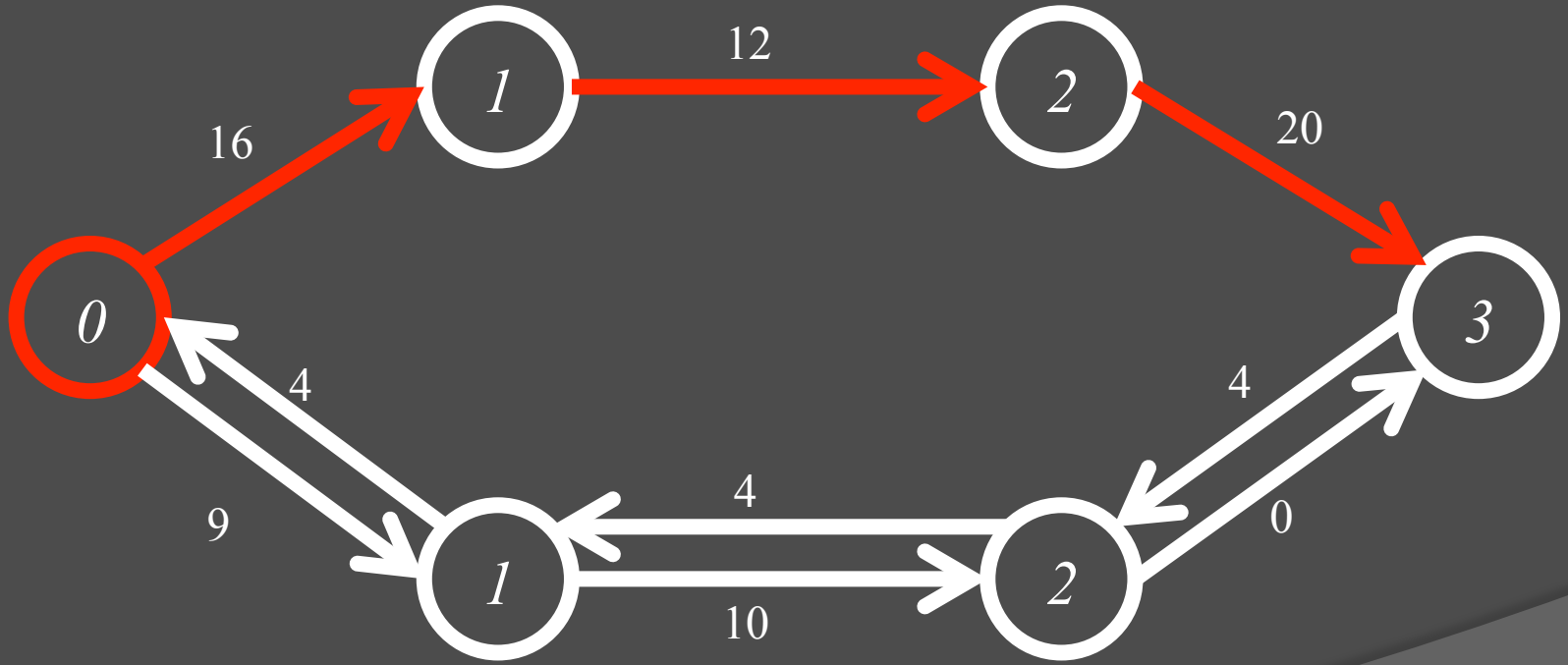
No path to node with level 3, backtrack until we can continue



Flow: 4

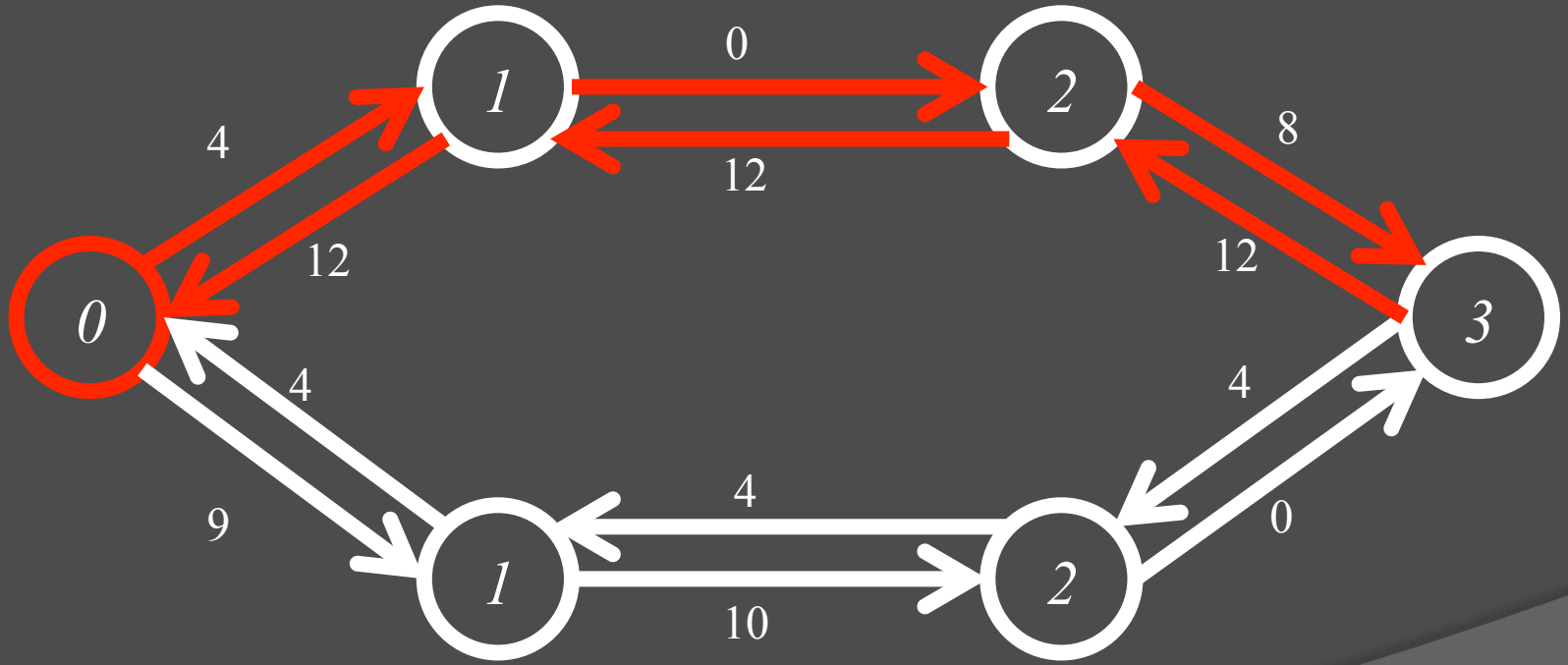
Dinic Diagram

Find an s-t path



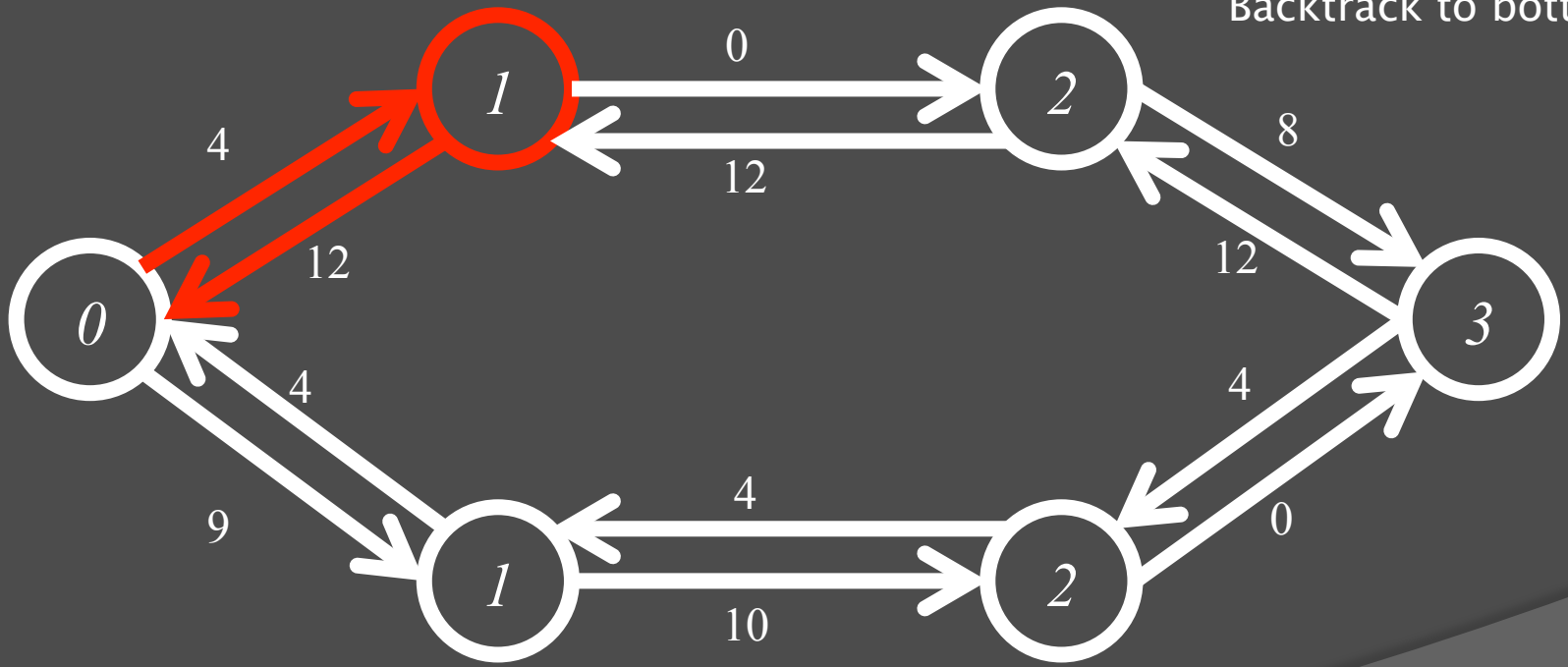
Flow: 4

Dinic Diagram

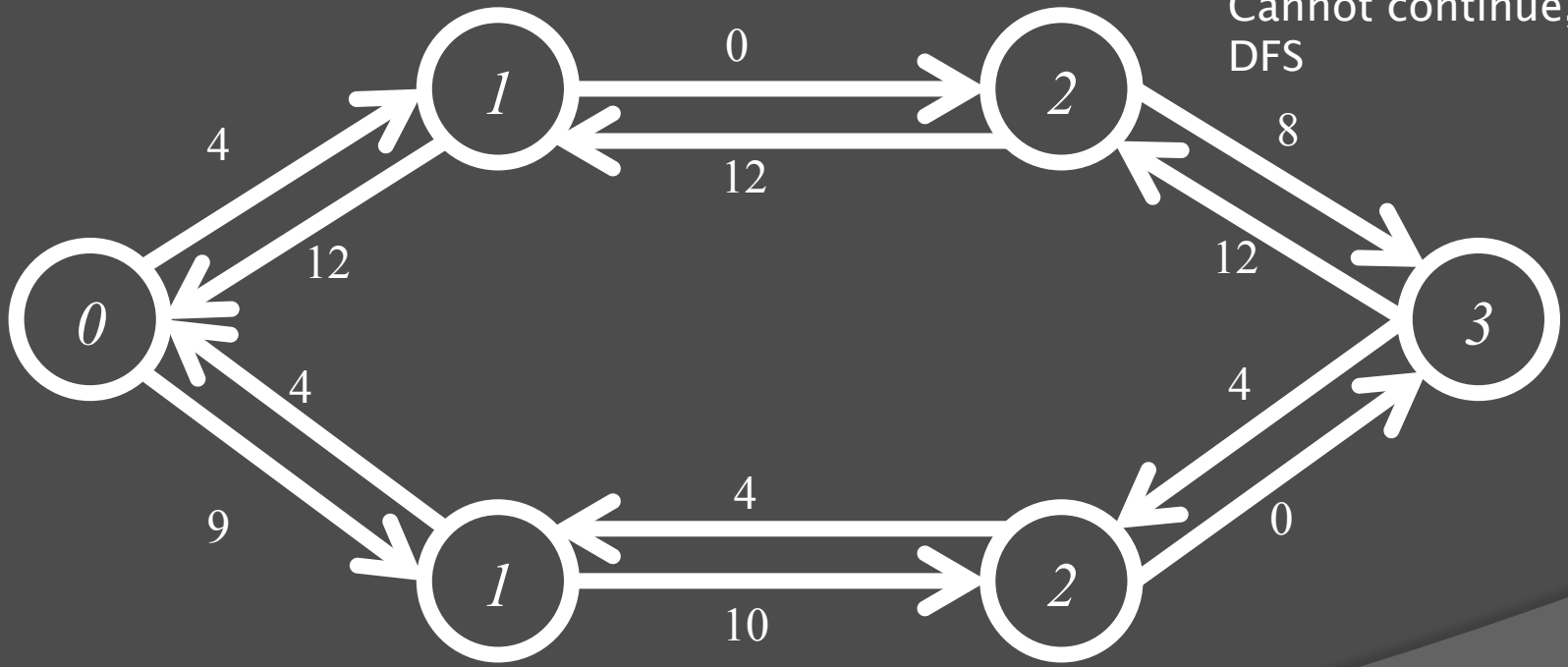


Flow: 16

Dinic Diagram



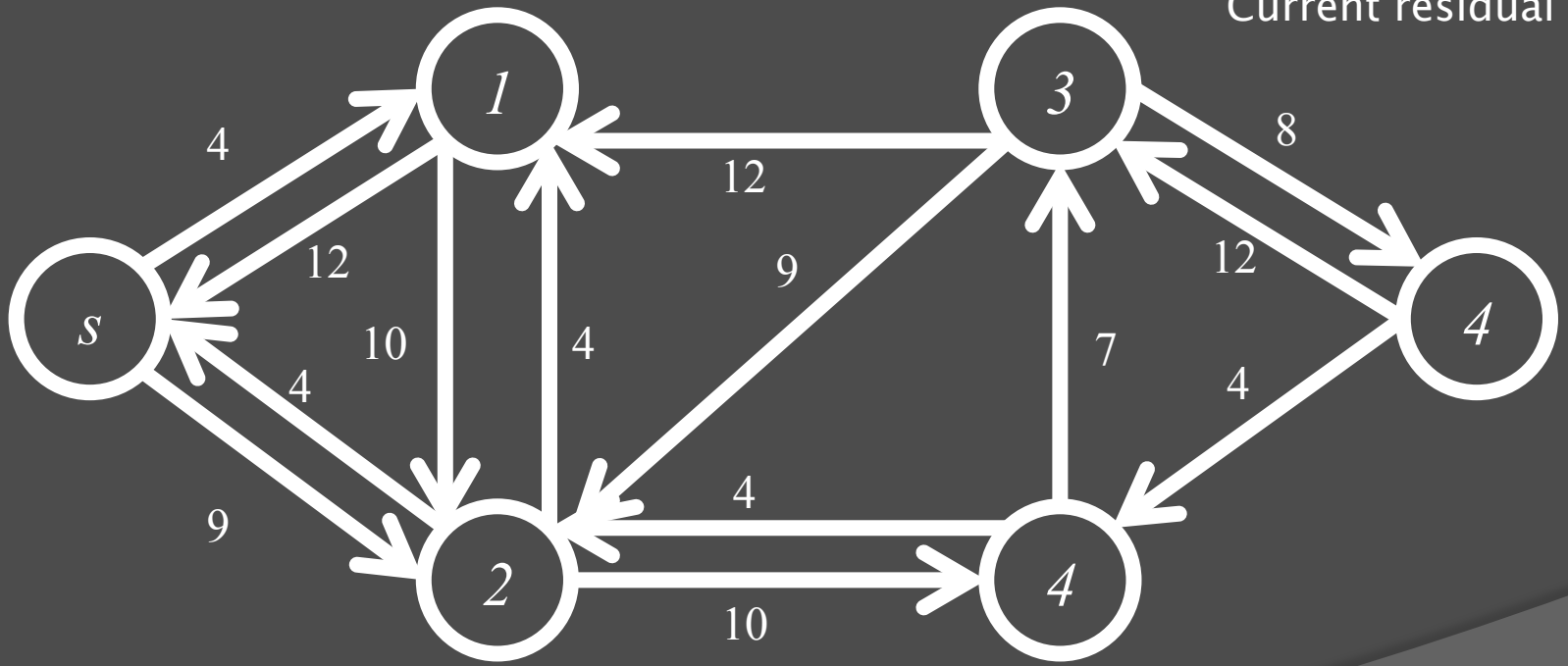
Dinic Diagram



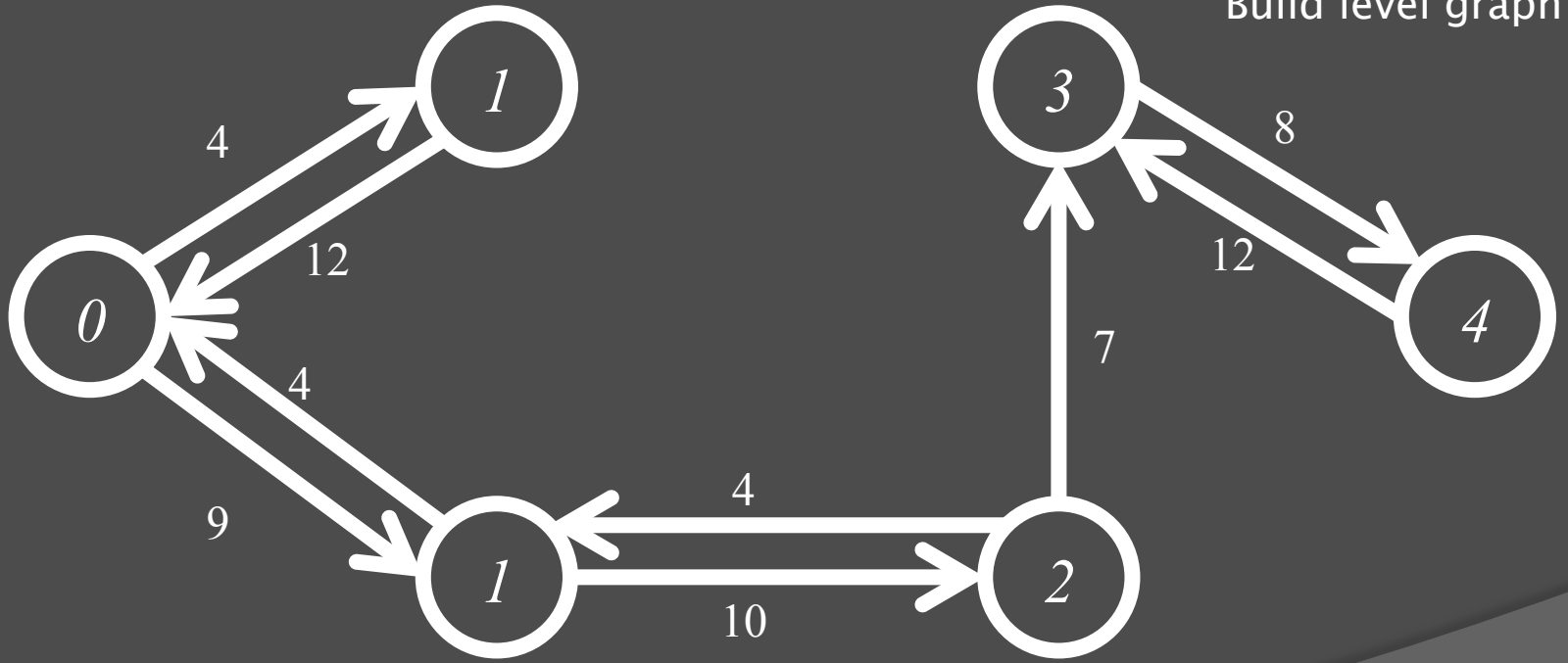
Cannot continue, exit DFS

Flow: 16

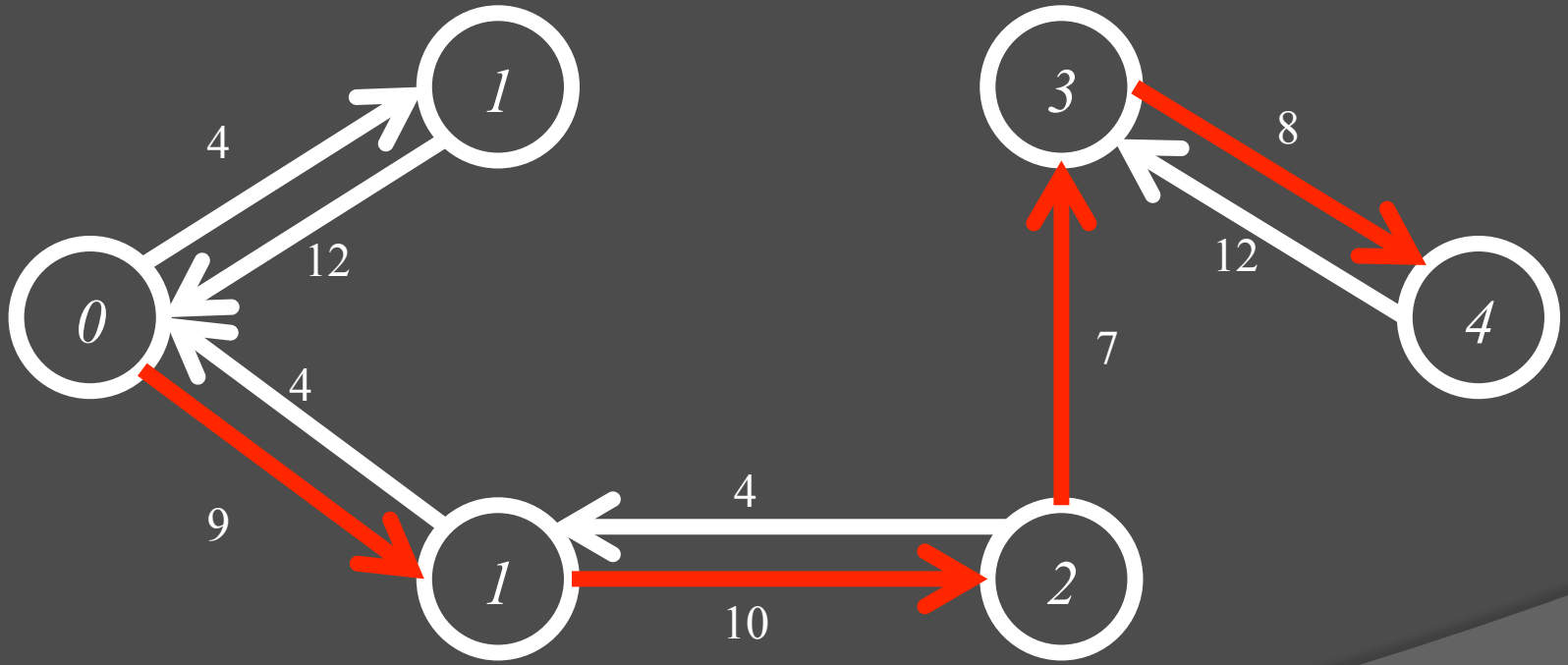
Dinic Diagram



Dinic Diagram

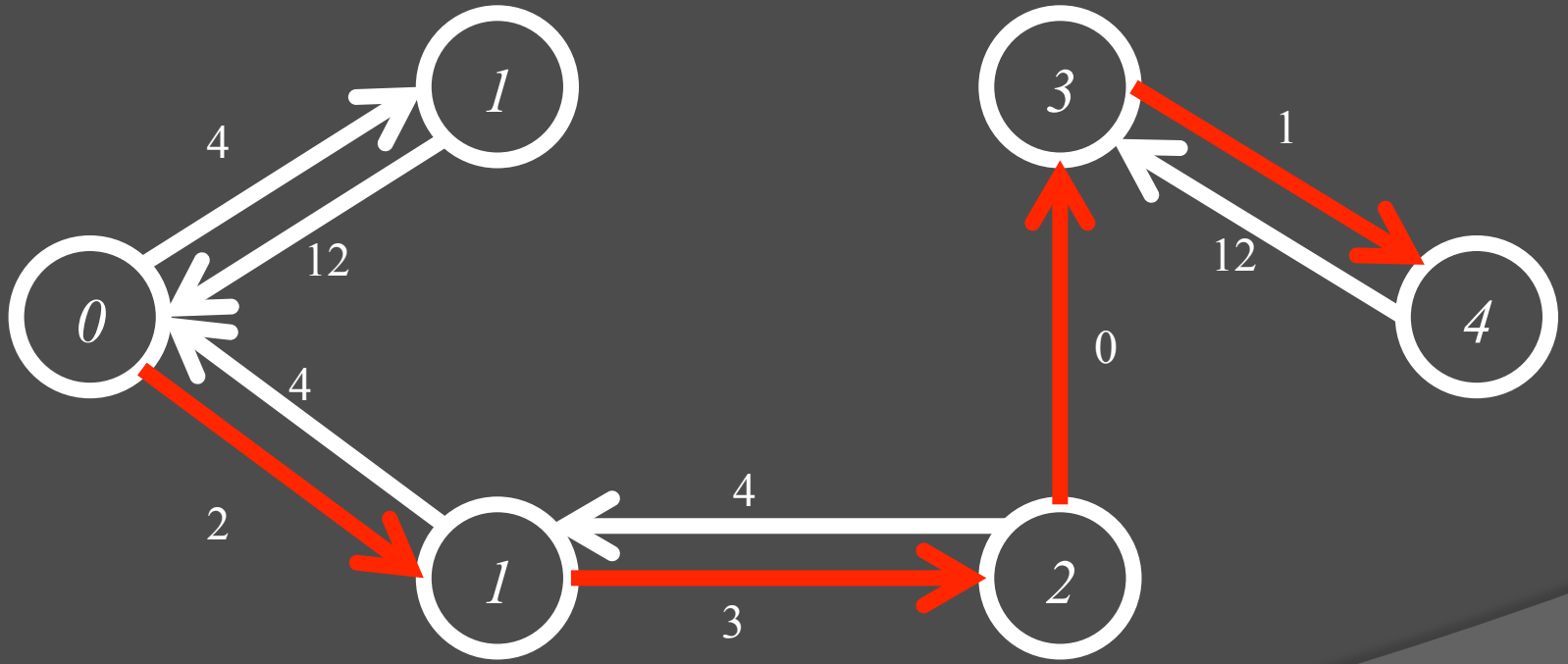


Dinic Diagram



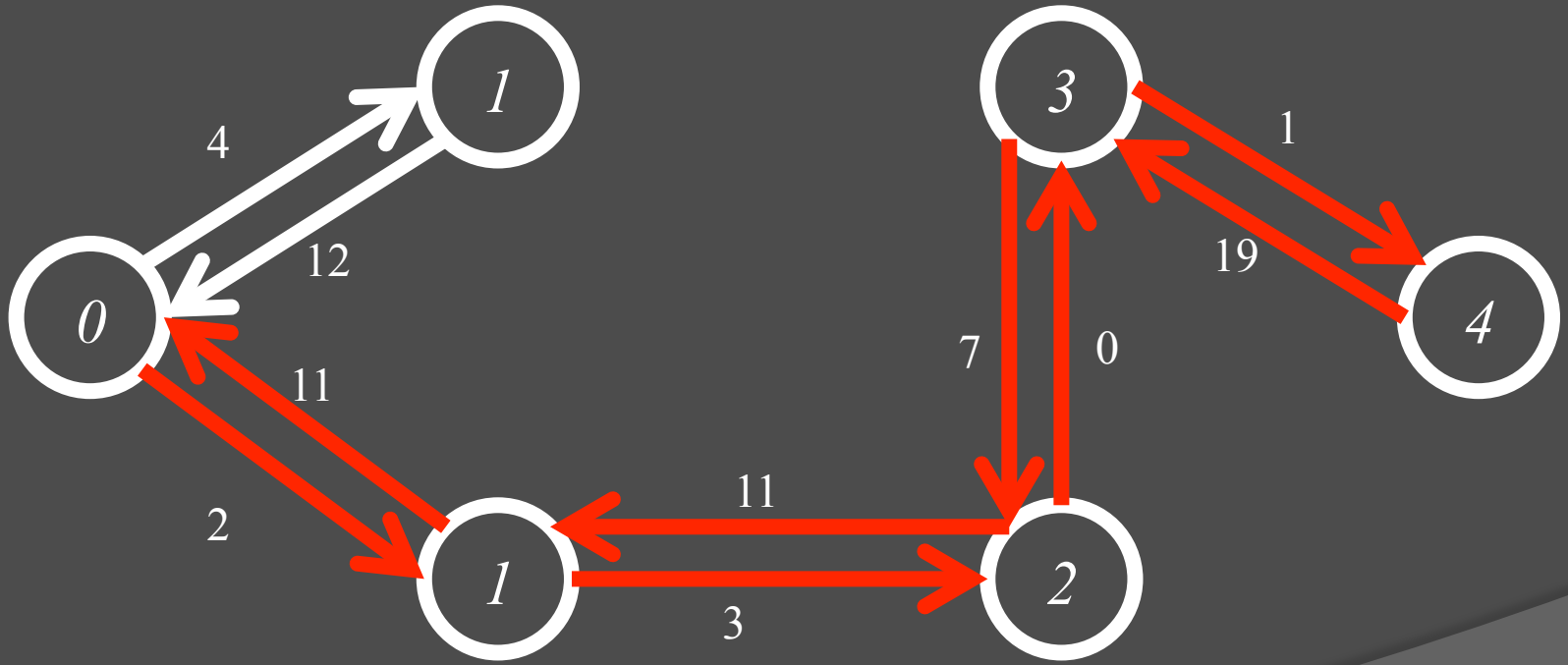
Flow: 16

Dinic Diagram



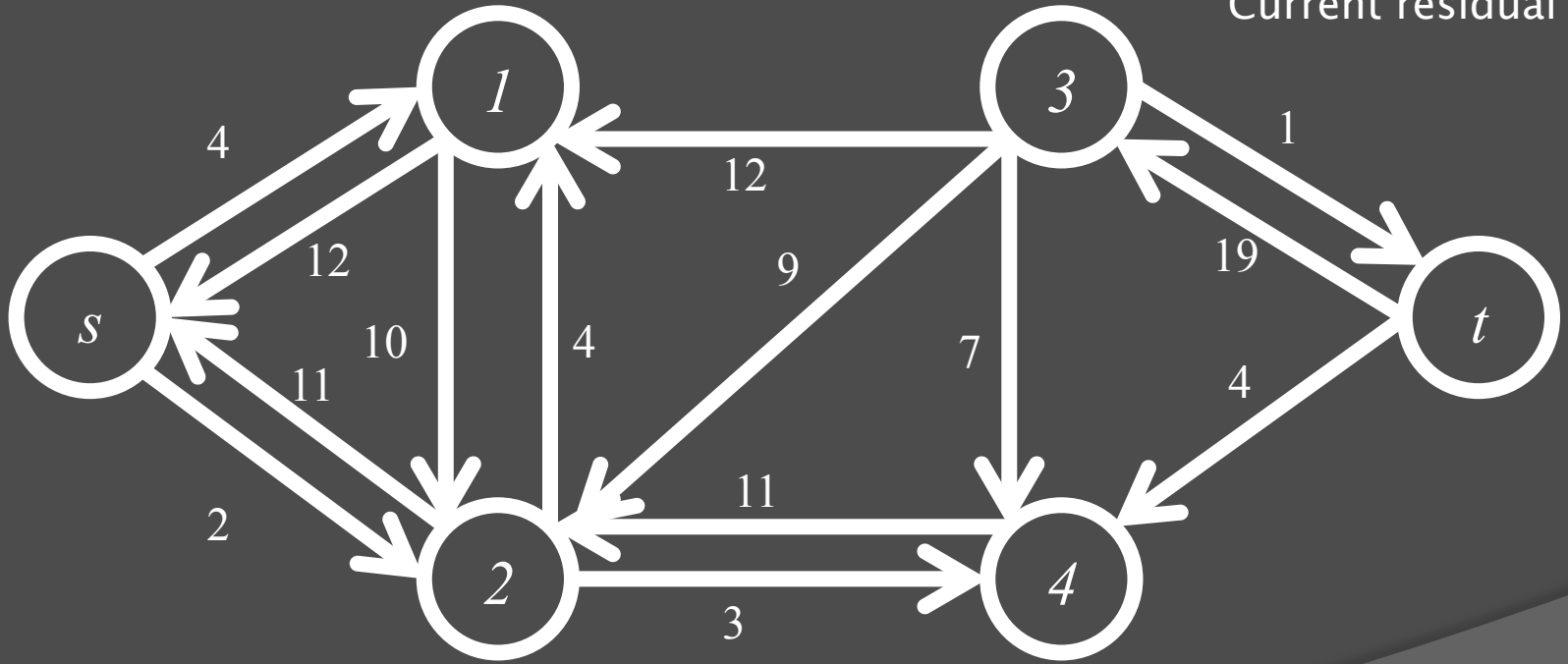
Flow: 23

Dinic Diagram

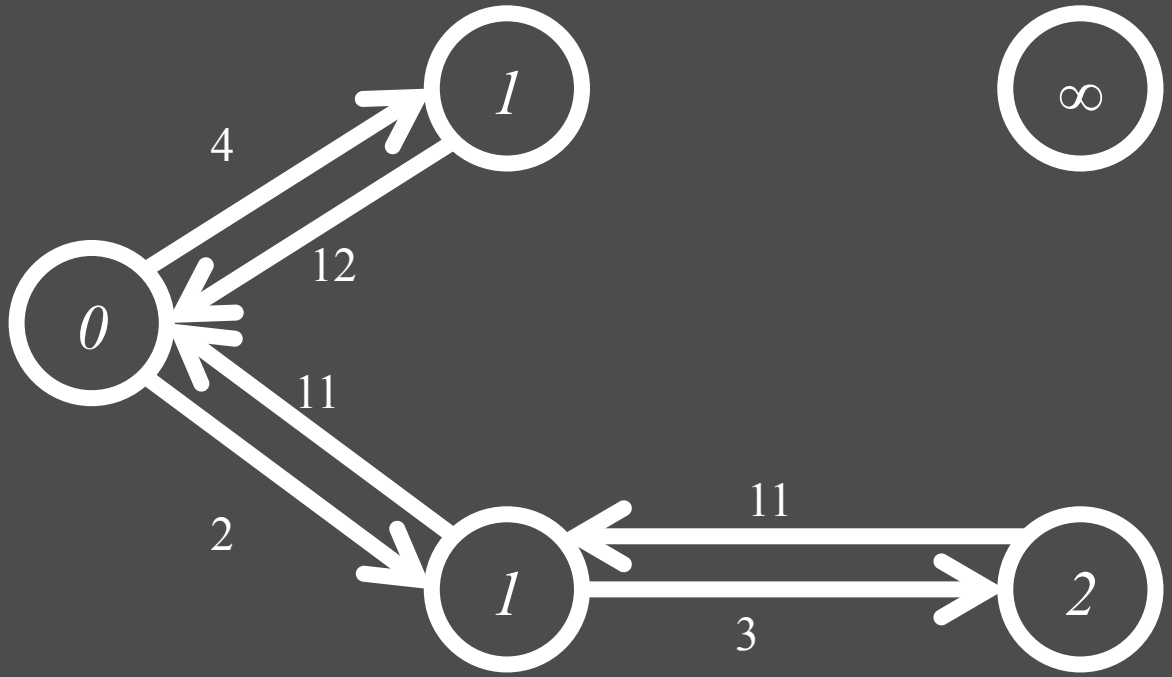


Flow: 23

Dinic Diagram



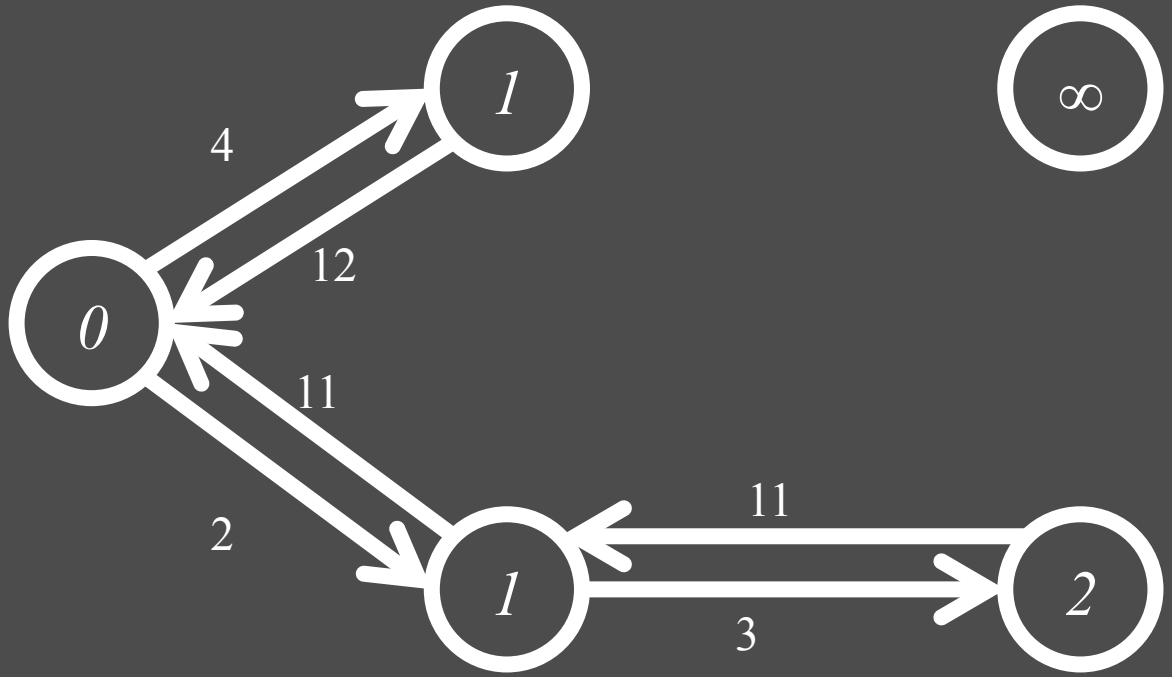
Dinic Diagram



Build level graph

Flow: 23

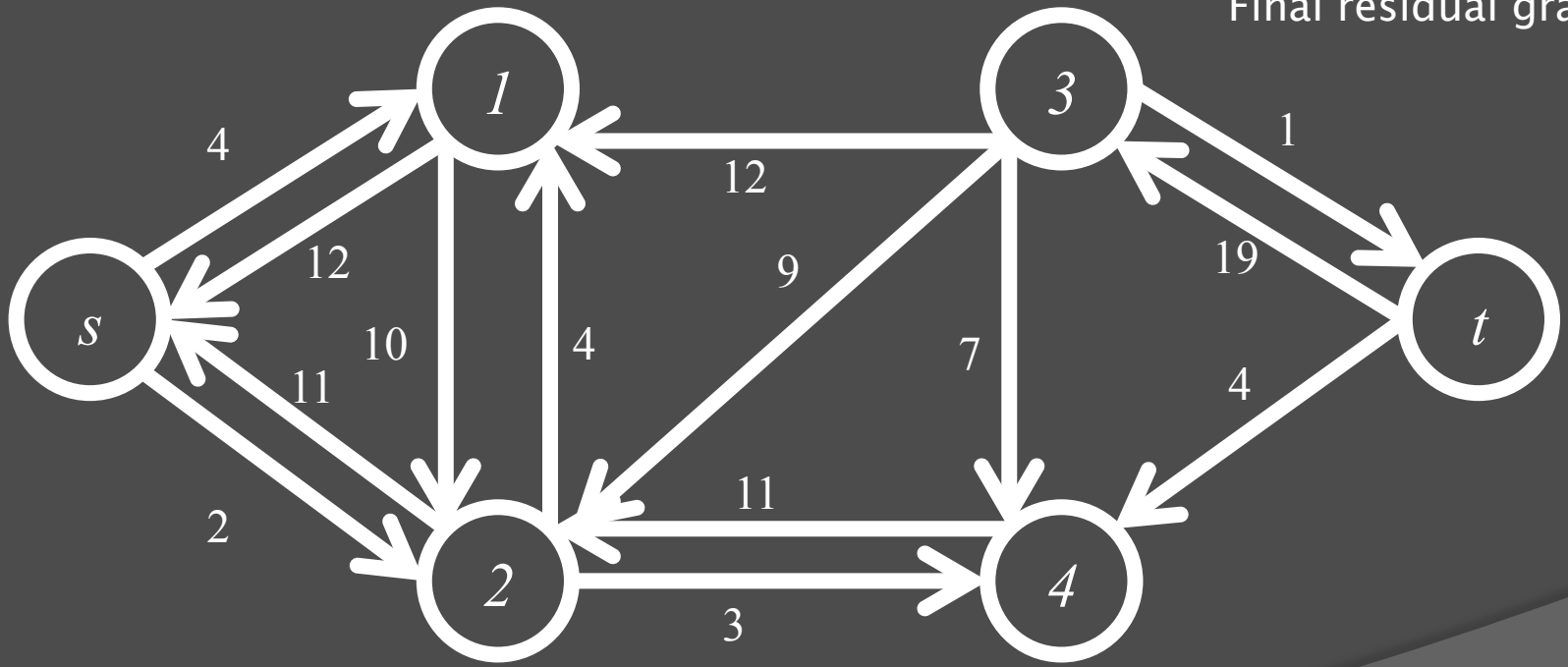
Dinic Diagram



No path from s to t, terminate

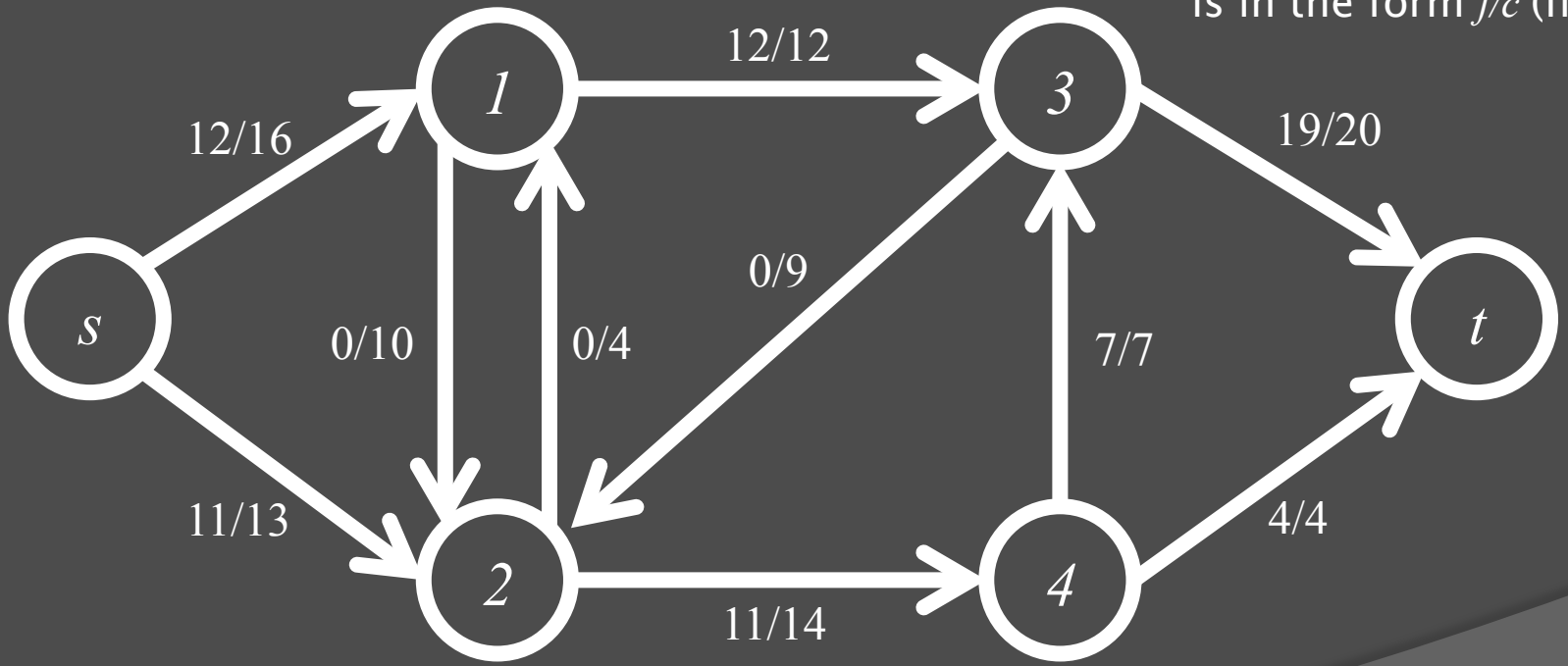
Flow: 23

Dinic Diagram



Dinic Diagram

The final network. Each edge is in the form f/c (flow/cap).



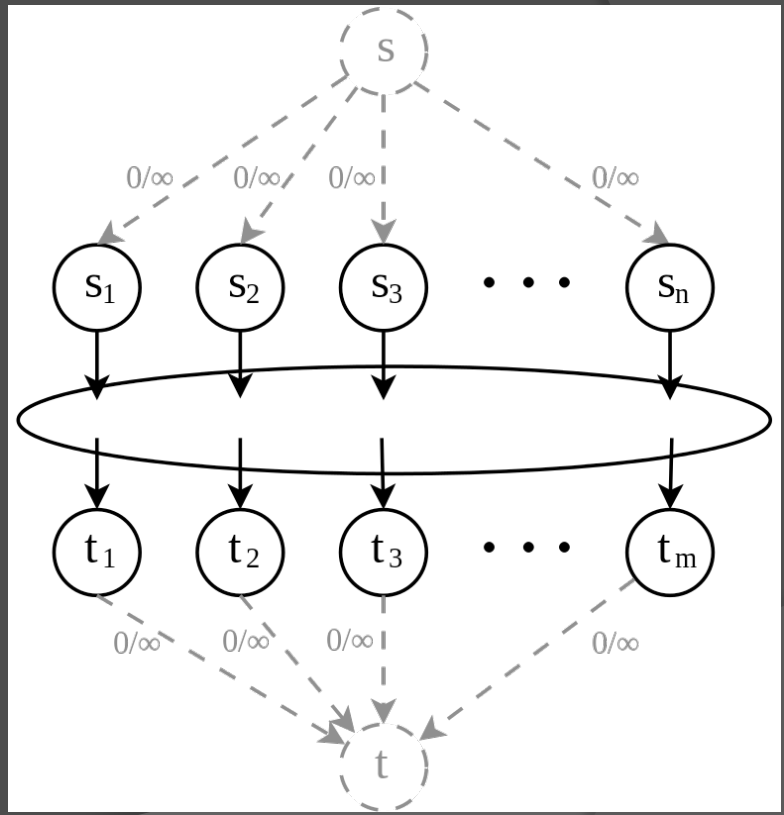
Flow: 23

Dinic Analysis

- BFS to construct level graph runs in $O(E)$ time
- DFS to find blocking flow takes $O(VE)$ time
  - Up to $O(E)$ edges become saturated, each time, up to $O(V)$ edges must have their flow increased
- The entire loop is run at most $O(V)$ times
  - The number of levels in the level graph increases by at least 1 each time; the maximum is V
- Final time complexity: $O(V^2E)$

Multi-source Multi-sink

- Create a **consolidated source** that connects to each source with an edge with infinite capacity
  - Use this new source as the single source
- Create a **consolidated sink** where each sink connects to this new sink with edges of infinite capacity
  - Use this new sink as the single sink

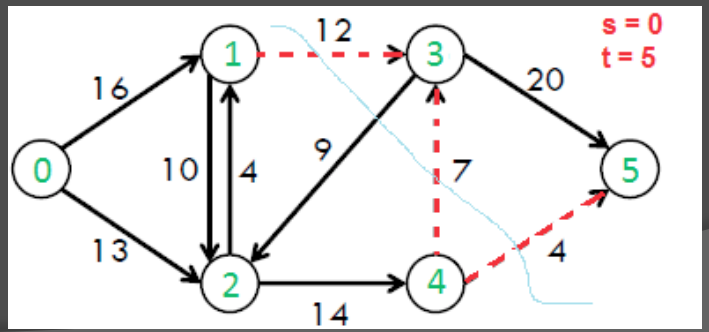


Applications

- Maximum flow of a flow network
- Minimum cut of a graph
- Maximum bipartite matching (next week)

Minimum Cut

- We are given a flow network
- Remove some edges such that there is no longer any path from s to t
- Minimize the sum of the weights of the removed edges
- **Max-flow min-cut theorem**
  - The minimum cut of a flow network is equal to the maximum flow of the network



THANK YOU!