DTS203TC
Design and Analysis of Algorithms

Lecture 15: Maximum Flow

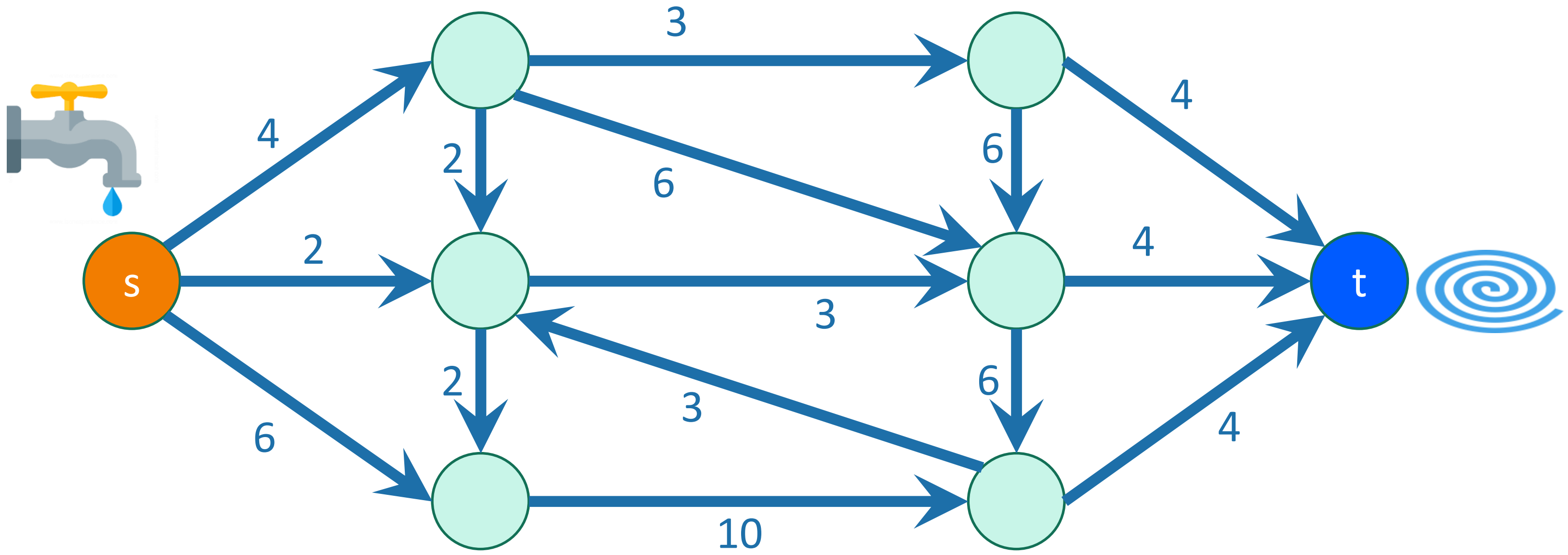
Dr. Qi Chen
School of AI and Advanced Computing

Learning Outcome

- Flow networks
- Maximum Flow
- Ford-Fulkerson Method
- Minimum cut
- Maximum bipartite matching

Flow Network

- Flow Network:
  - Graphs are directed
  - Edges have “capacities” (weights)
  - Two distinguishes vertices, namely
    - “source” vertex s has only outgoing edges
    - “sink” vertex t has only incoming edges

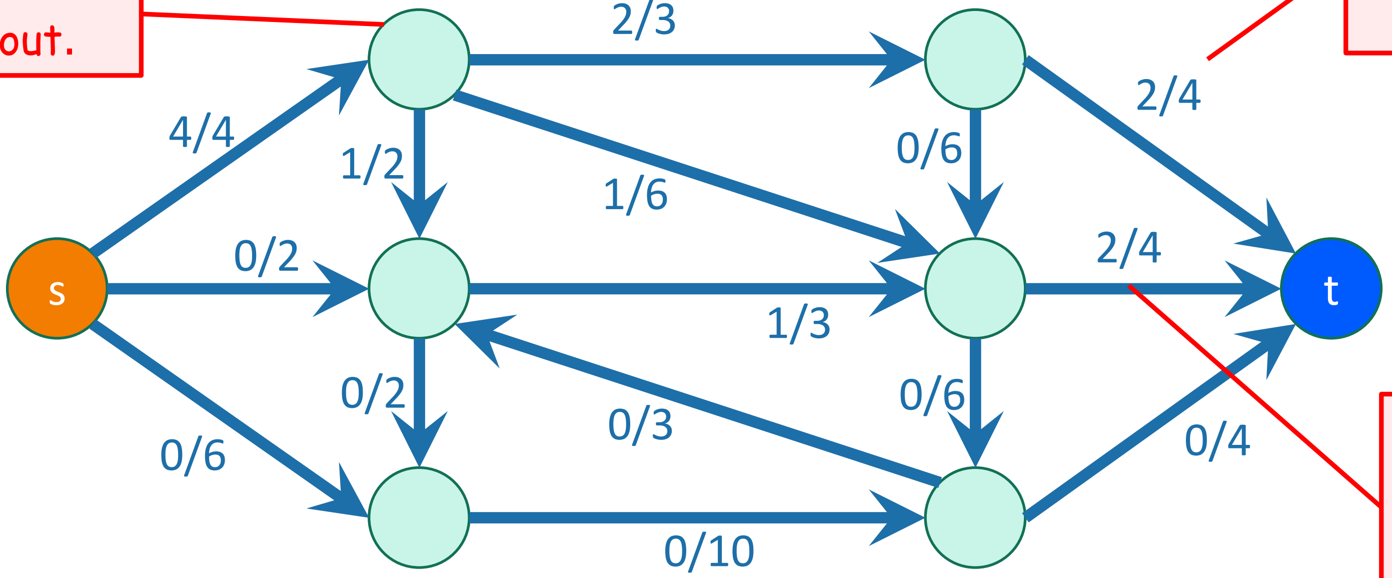


Flow

- In addition to a capacity, each edge has a flow.
- The flow on an edge must be less that its capacity.
- At each vertex (except s and t), the incoming flows must equal the outgoing flows.

4 units in,
1+1+2 = 4 units out.

Flow / Capacity
The slash notation merely separates the flow and capacity; it does not indicate division.



Think of this flow as meaning **"send 2 units of stuff along this edge."**

Flow

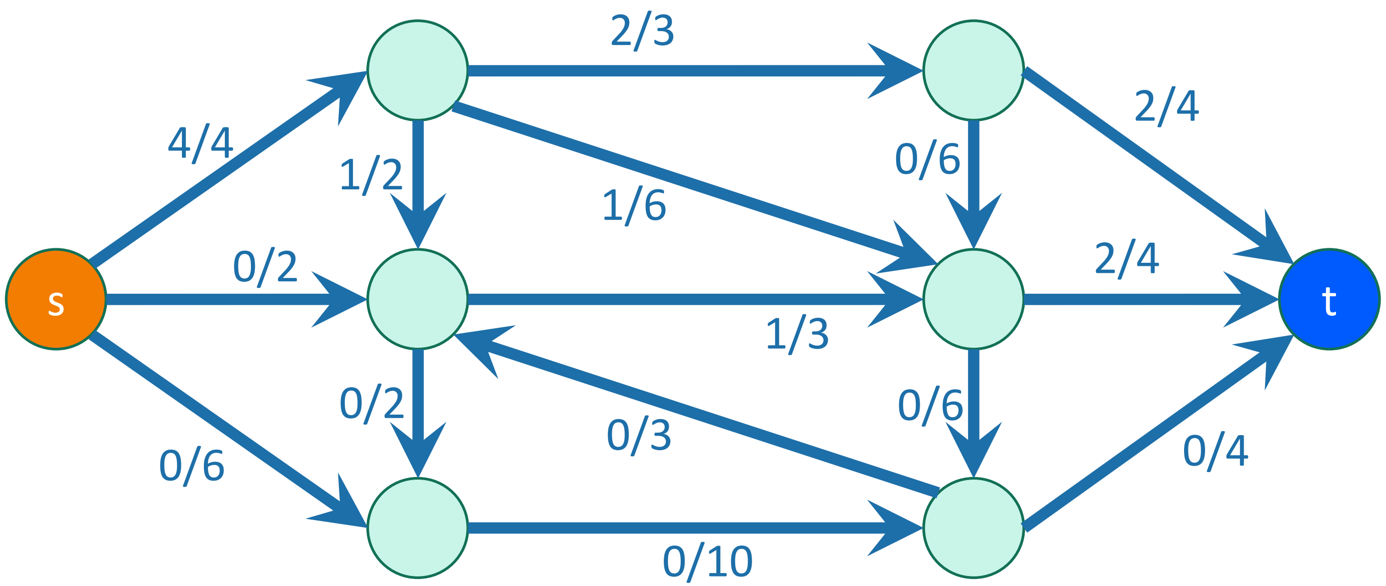
- The value of a flow is:
  - The amount of stuff coming out of s
  - The amount of stuff flowing into t

$$|f| = \sum_{w \epsilon out(s)} flow(s,w) = \sum_{u \epsilon in(t)} flow(u,t)$$

- out(s) is the set of vertices w such that there is an edge from s to w
- in(t) is the set of vertices u such that there is an edge from u to t

Flow

- What is the value of this flow?

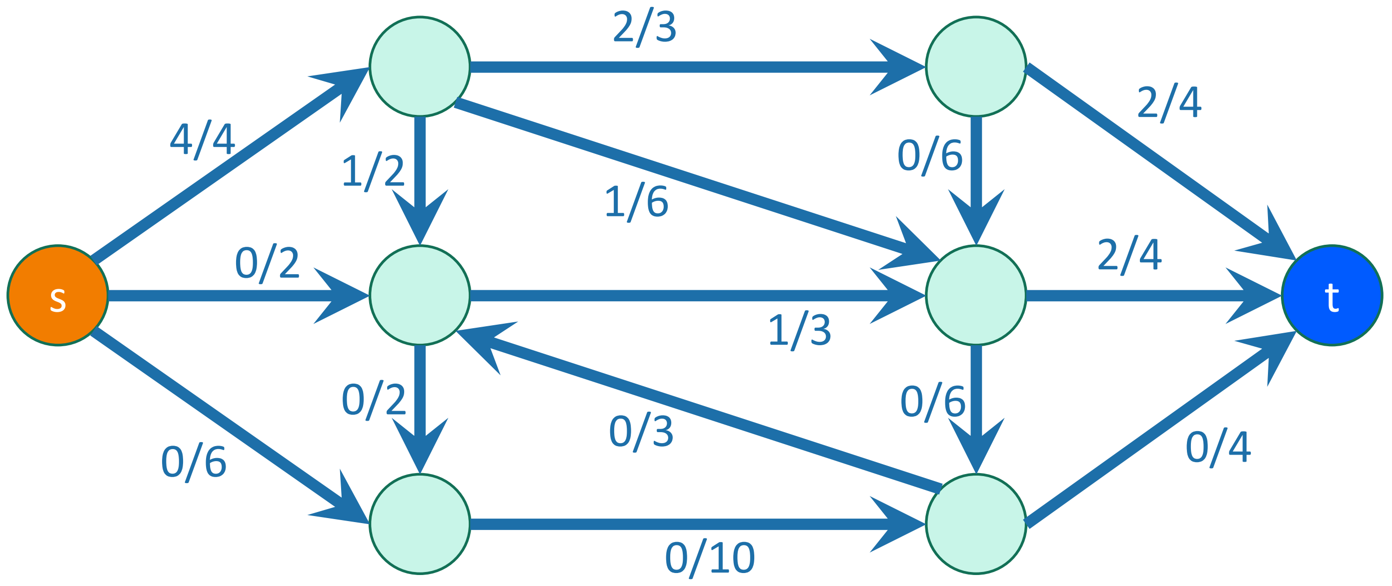


The value of this flow is 4.

Maximum Flow Problem

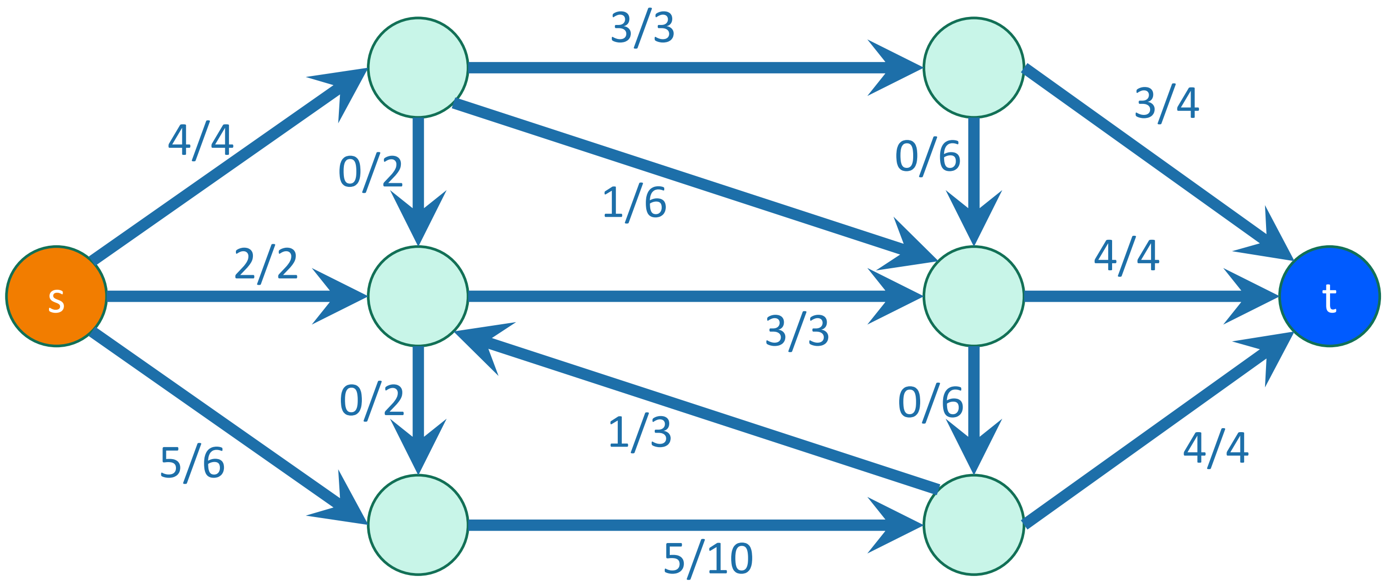
- "Given a network N, find a flow f of maximum value."

- This example is NOT maximum flow. Why?
  - Not utilizing the capacities very well.

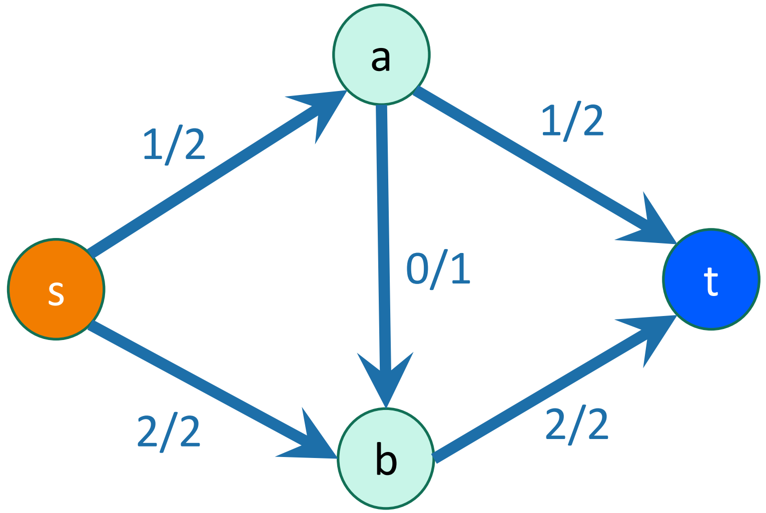


Maximum Flow

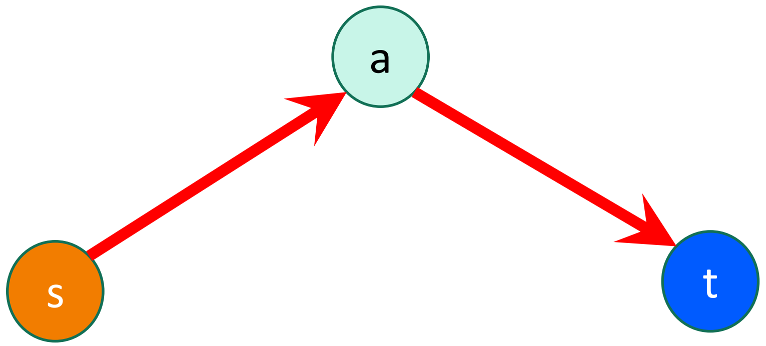
- This one is maximal; it has value 11.



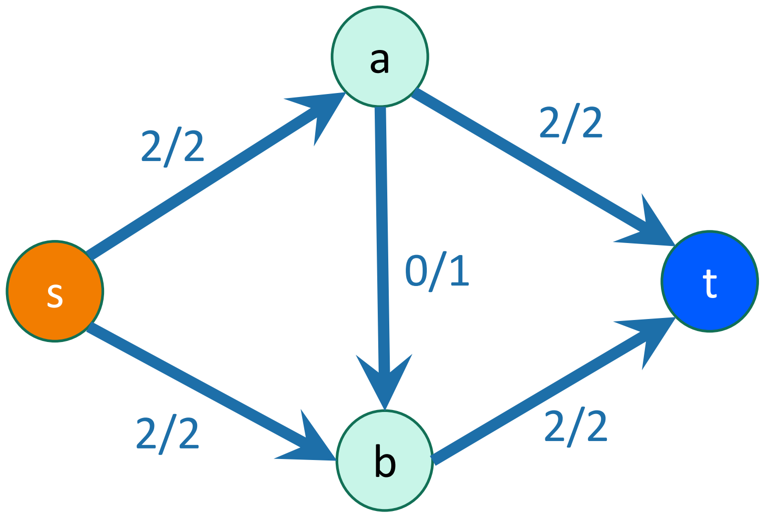
Augmenting Path



A network with a flow of value 3



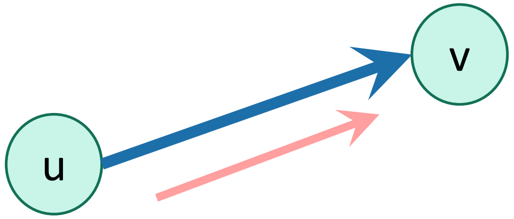
Augmenting path



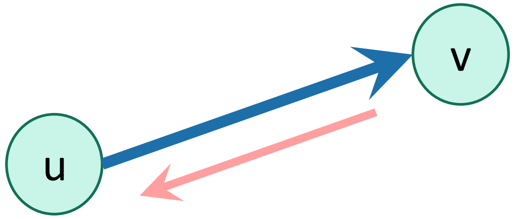
The flow value is increased to 4

Augmenting Path

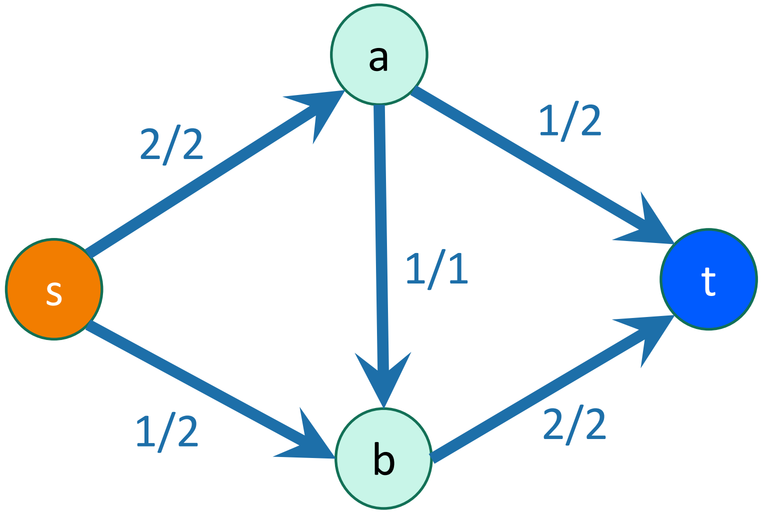
- Forward Edges
  Flow(u,v) < capacity(u,v)
  Flow can be increased



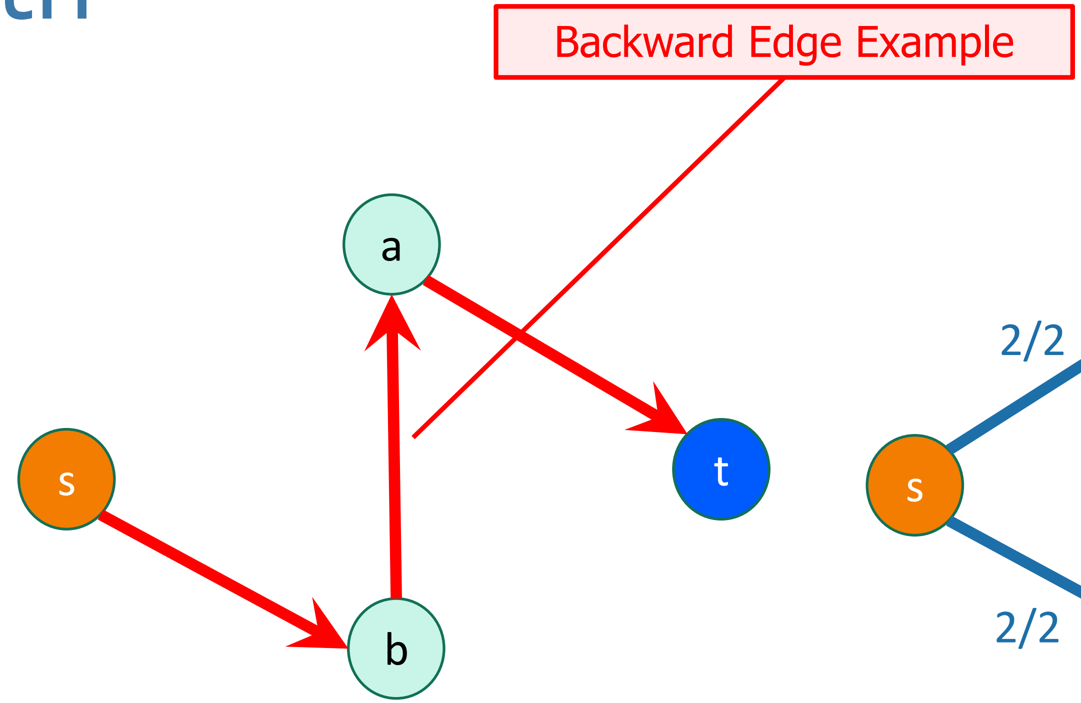
- Backward Edges
  Flow(u,v) > 0
  Flow can be decreased



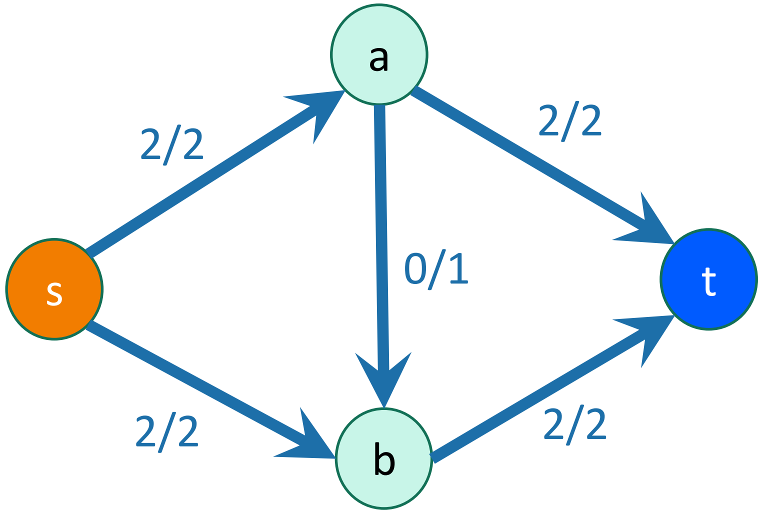
Augmenting Path



A network with a flow of value 3



Augmenting path



The flow value is increased to 4

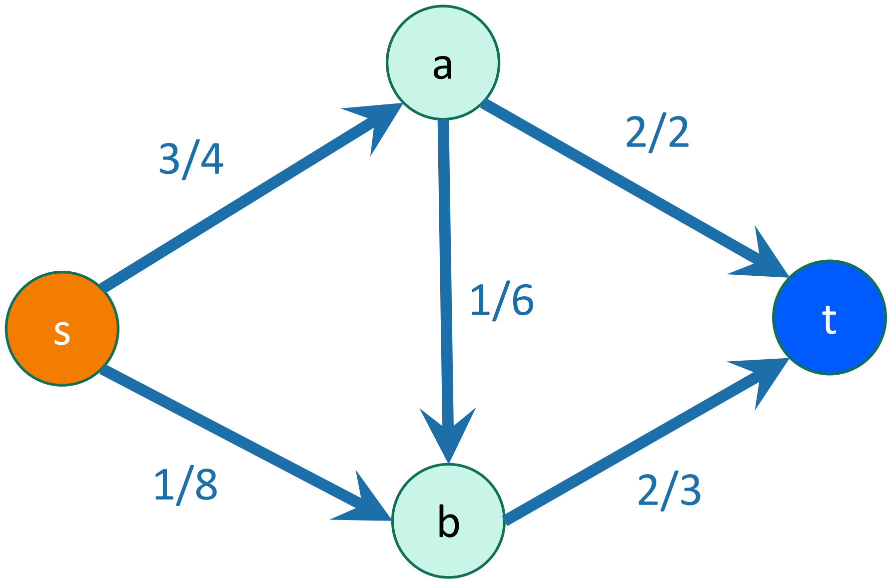
Maximum Flow Theorem

- A flow has maximum value if and only if it has no augmenting path.

Ford-Fulkerson Algorithm

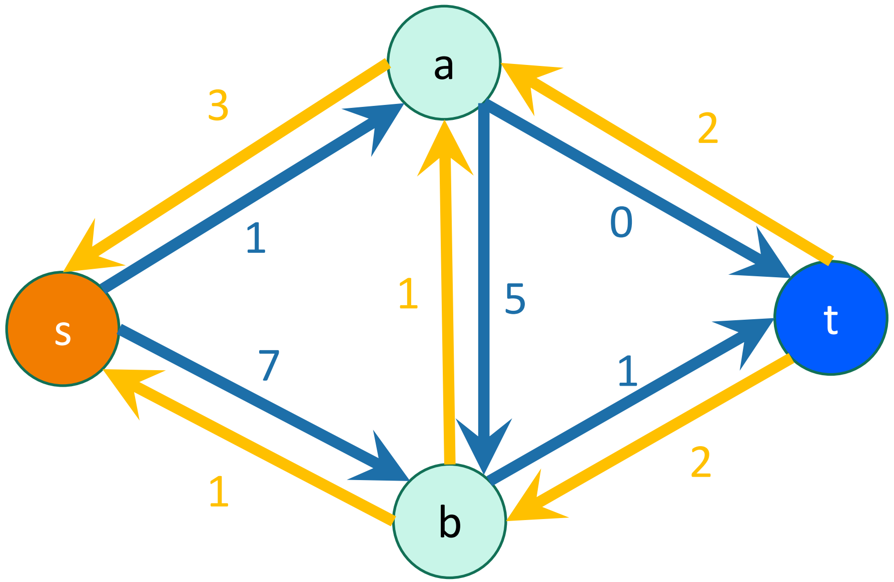
- Outline of algorithm:
  - Start with zero flow
  - We will maintain a "**residual network**" G_f
  - if augmenting paths exist in G_f, then find augmenting path and increase flow
  - Continue until there are no augmenting paths left.

Residual network



$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v) & if \;\; (u,v) \in E \\ f(v,u) & if \; (v,u) \in E \\ 0 & else \end{cases}$$

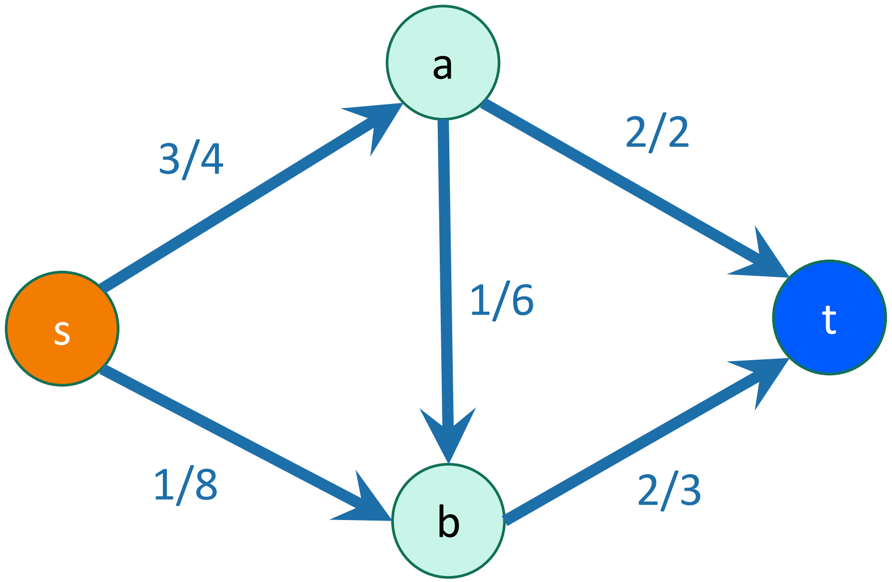
- $f(u,v)$ is the flow on edge (u,v).
- $c(u,v)$ is the capacity on edge (u,v)



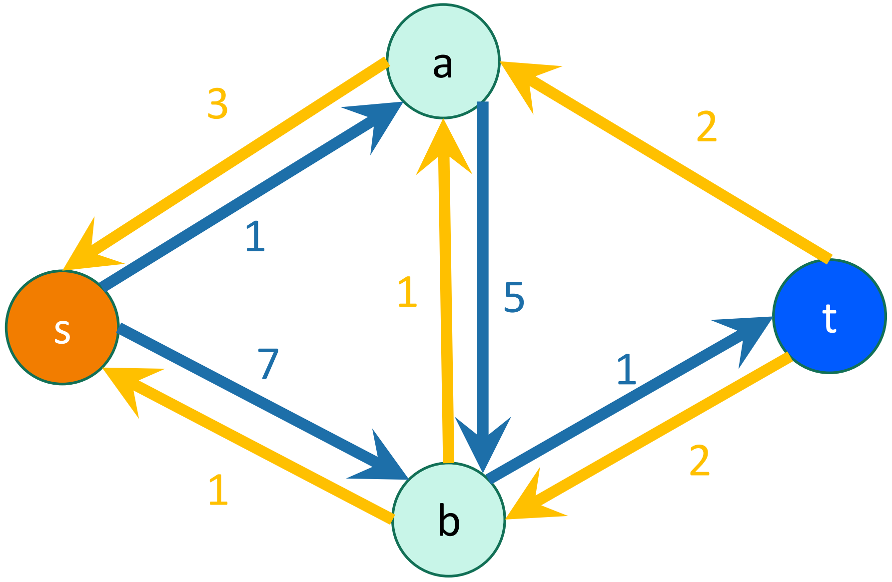
graph G_f

Create a new **residual network** from this flow:

Residual network



Forward edges are the amount that's left.
Backwards edges are the amount that's been used.
Edges with capacity $c_f(u, v)$=0 are removed

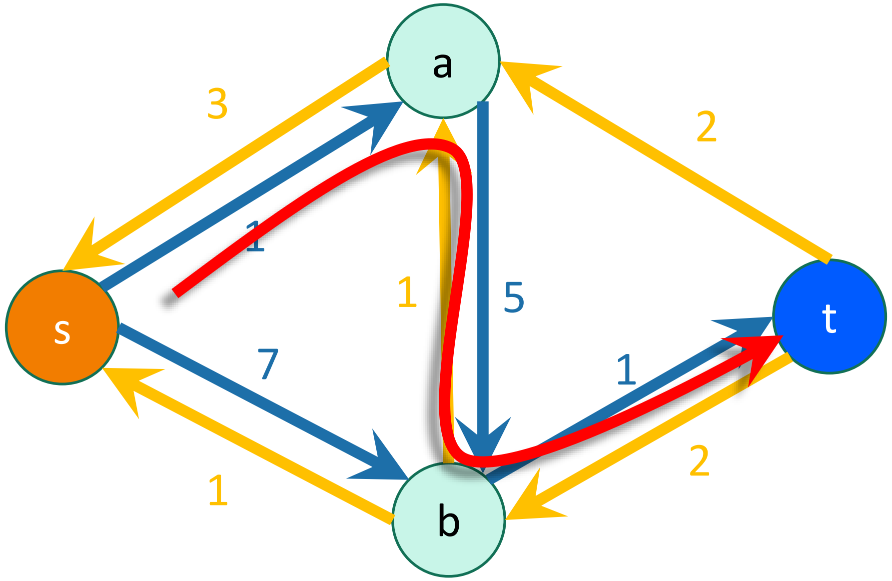


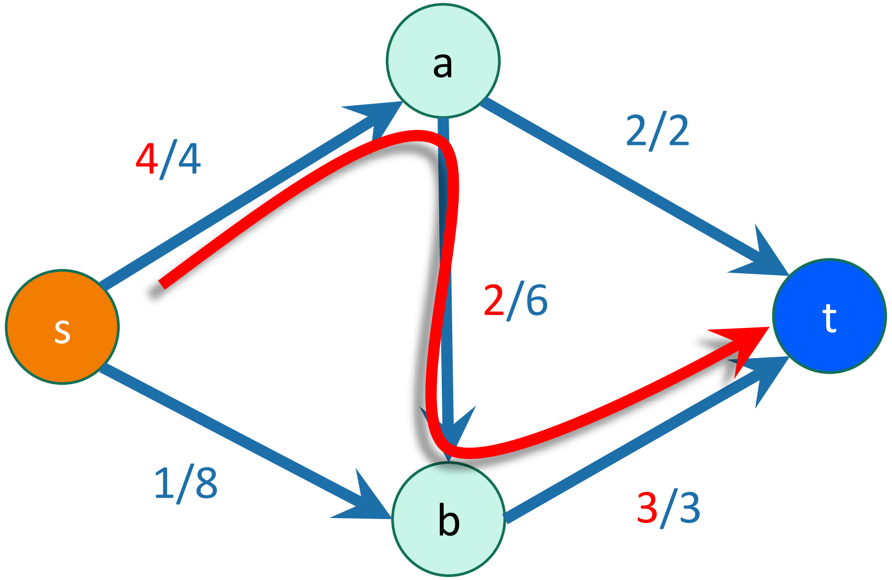
graph G_f

Create a new **residual network** from this flow:

Residual network

- t is not reachable from s in G_f $\Leftrightarrow$ f is a max flow.

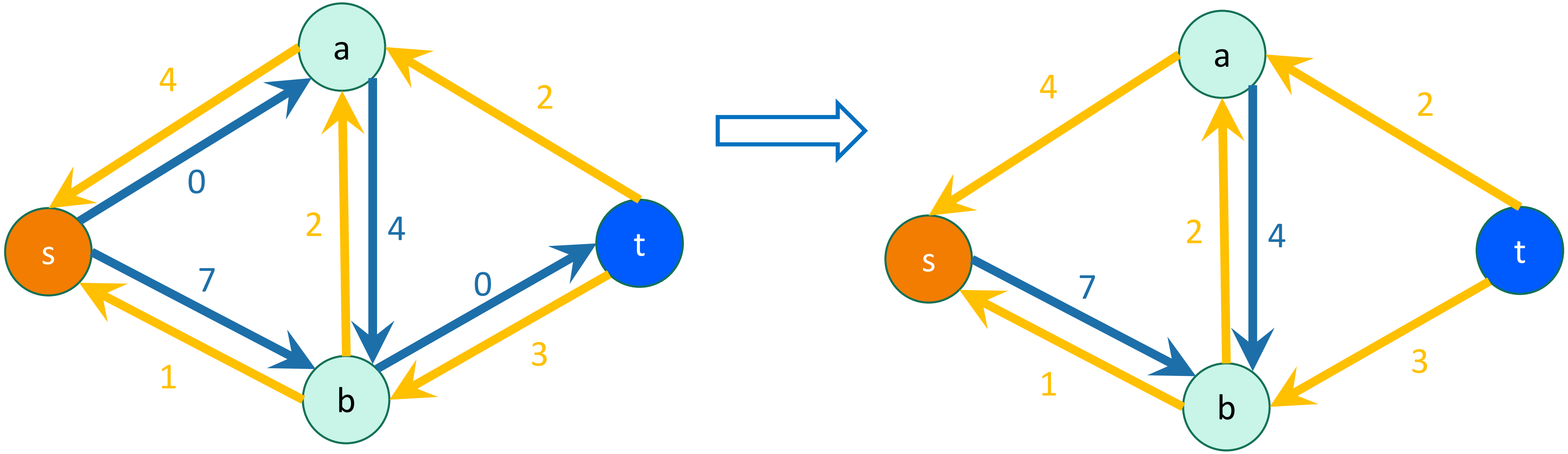




s is reachable from t in this example, so not a max flow.

Residual network

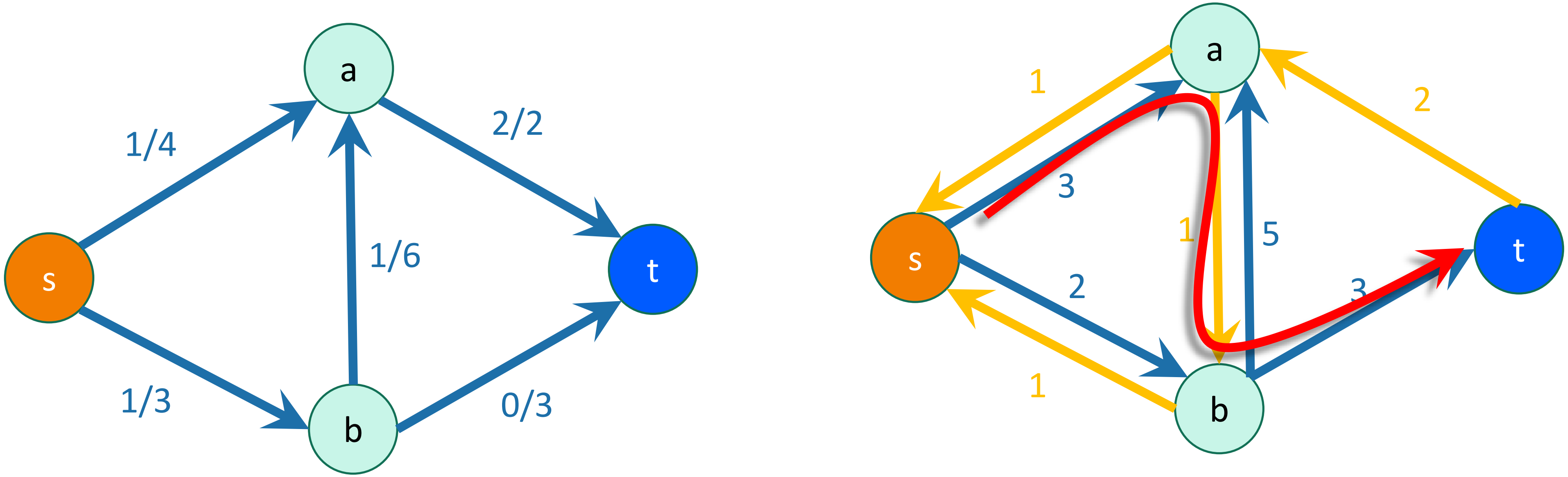
- Now update the residual network



Now we can’t reach t from s.
f is a max flow.

Residual Network

- Maybe there are backward edges in the path.

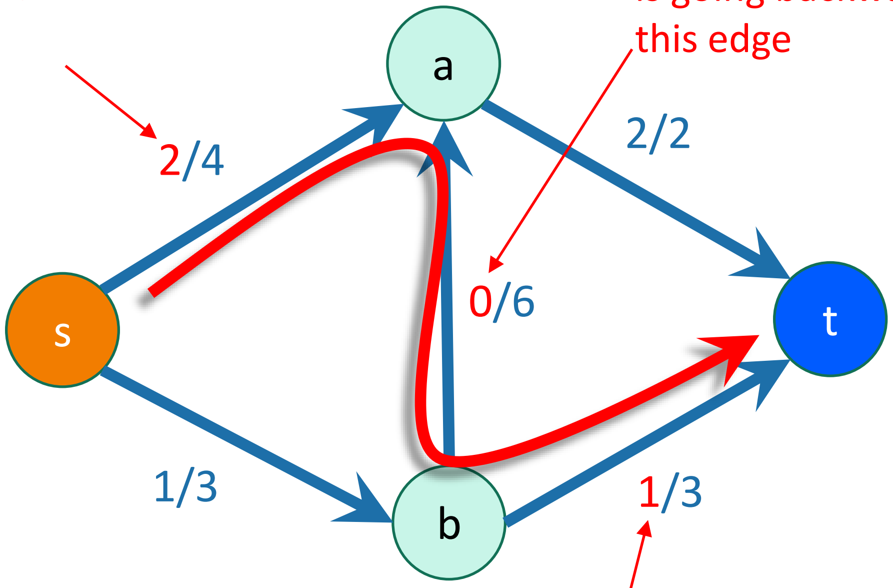


Residual Network

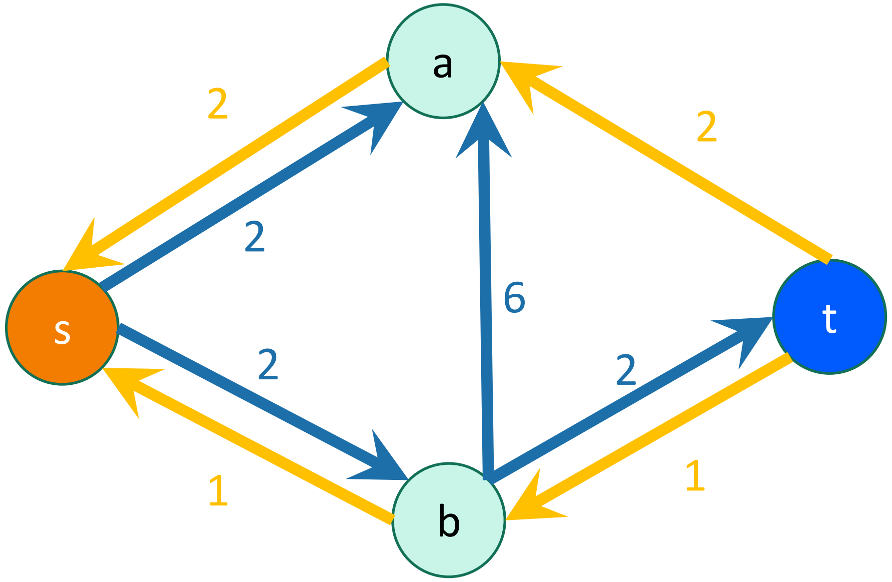
- Maybe there are backward edges in the path.

We will add flow here

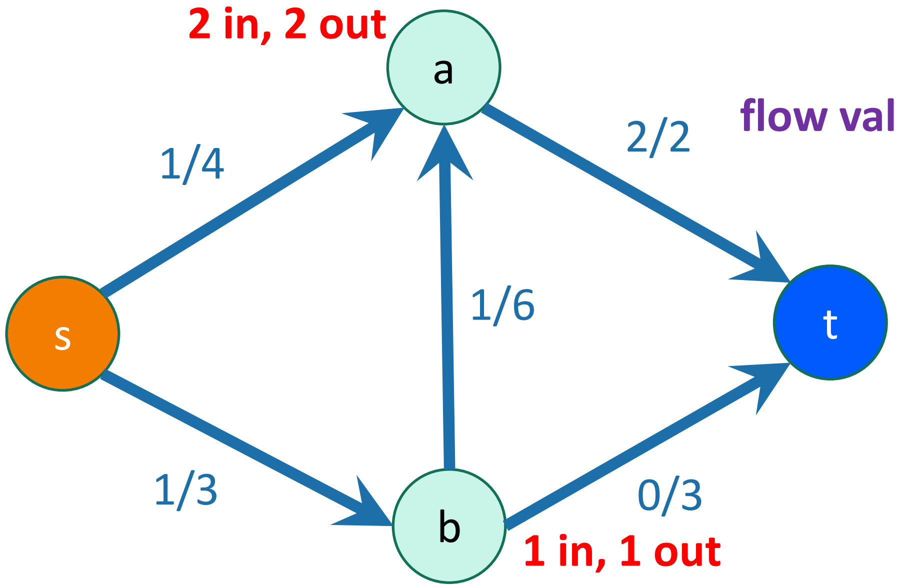
We will remove flow here, since our augmenting path is going backwards along this edge



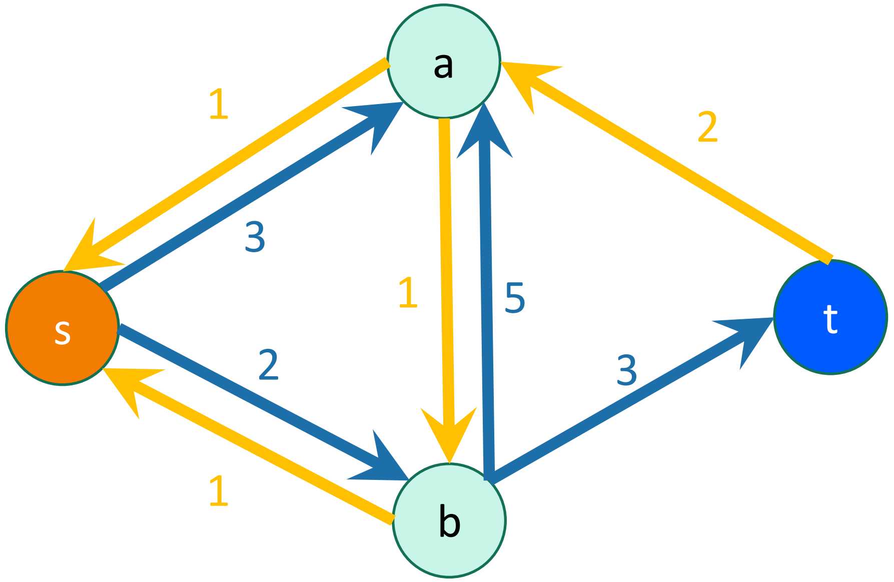
We will add flow here



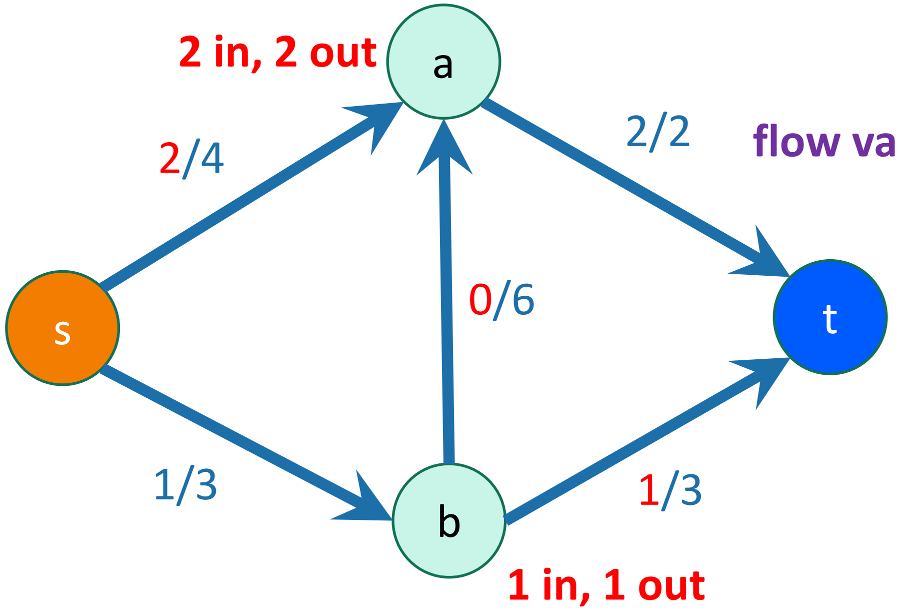
before



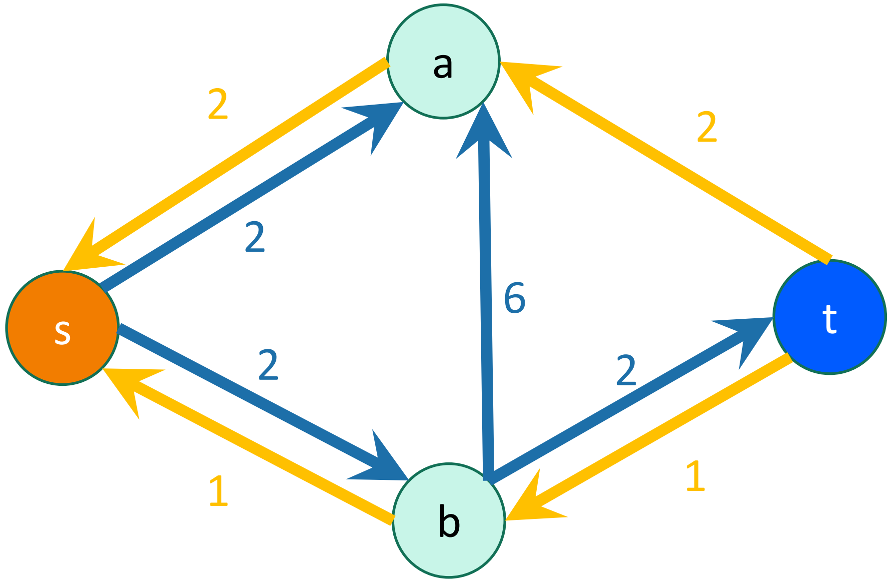
flow value is 2



After



flow value is 3



Increase flow

increaseFlow(path P in G_f , flow f):

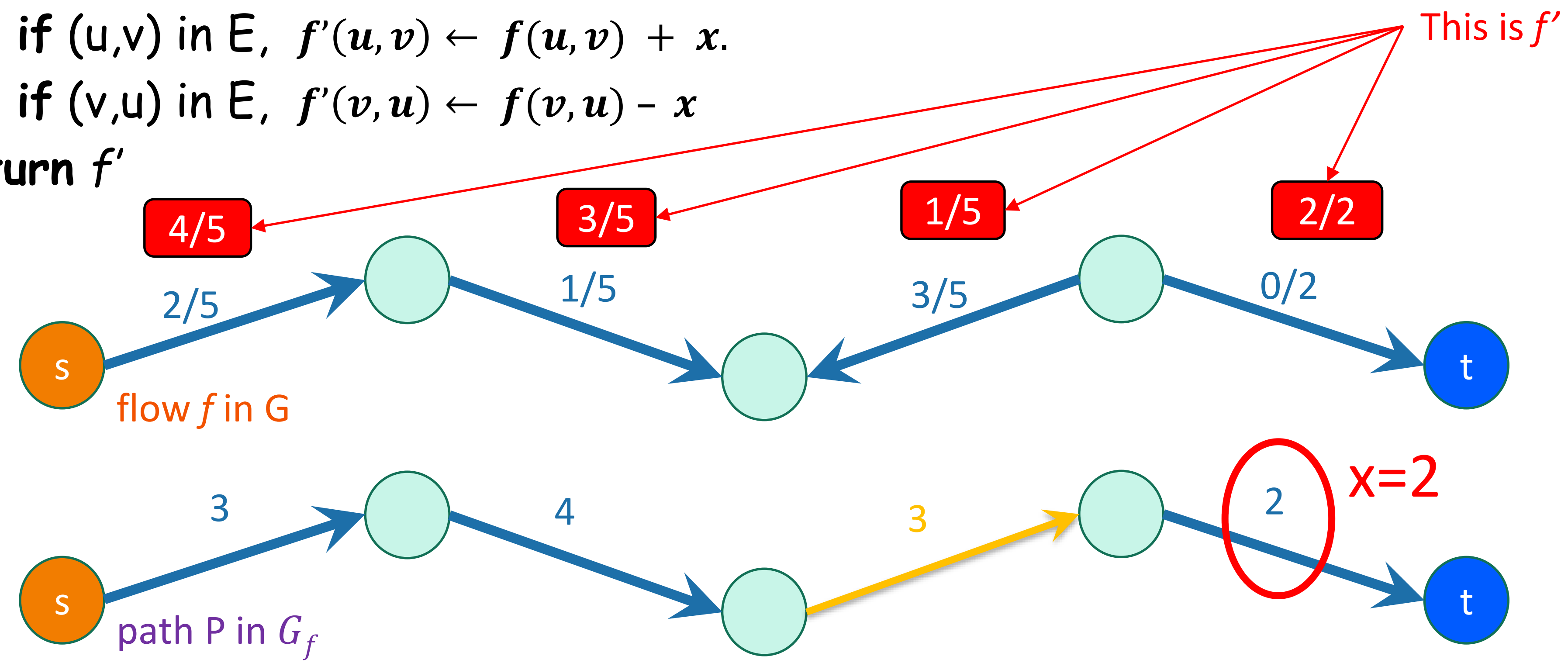
x = min weight on any edge in P

for (u,v) in P:

if (u,v) in E, $f'(u,v) \leftarrow f(u,v) + x$.

if (v,u) in E, $f'(v,u) \leftarrow f(v,u) - x$

return f'



Ford-Fulkerson

Ford-Fulkerson(G):

$f \leftarrow$ all zero flow.

$G_f \leftarrow G$

while t is reachable from s in G_f

Find a path P from s to t in G_f // eg, use BFS

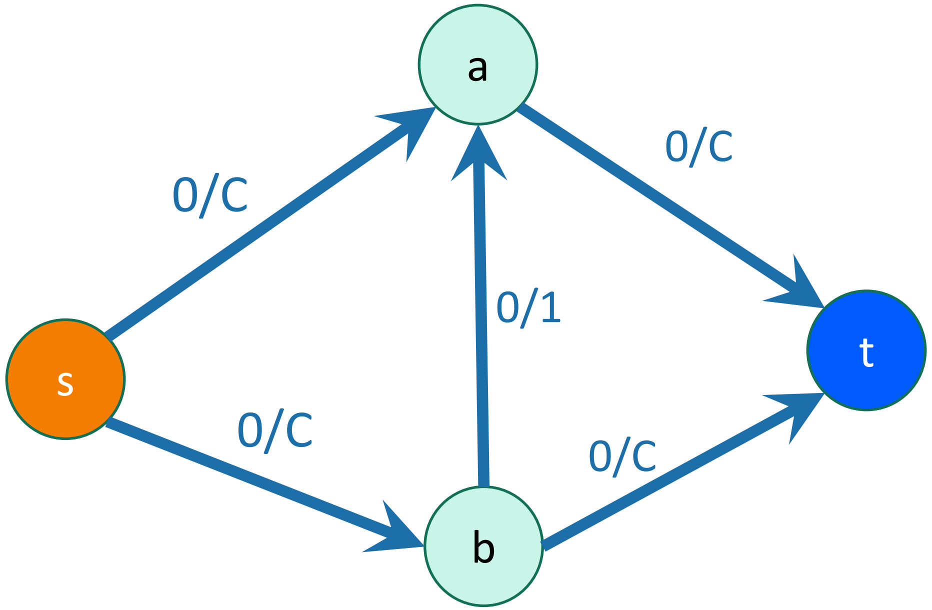
$f \leftarrow$ **increaseFlow**(P,f)

update G_f

return f

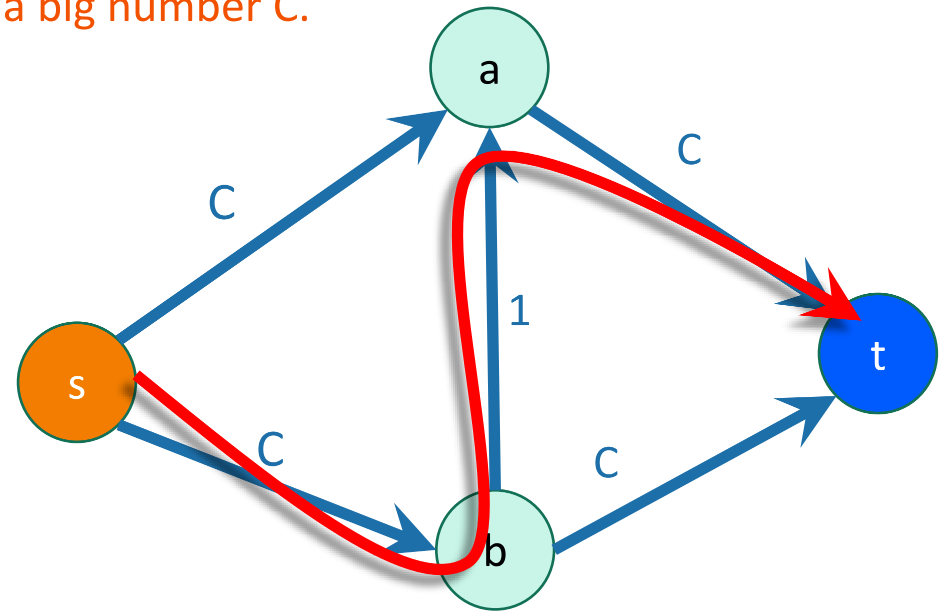
Running Time?

Suppose we just picked paths arbitrarily.



graph G

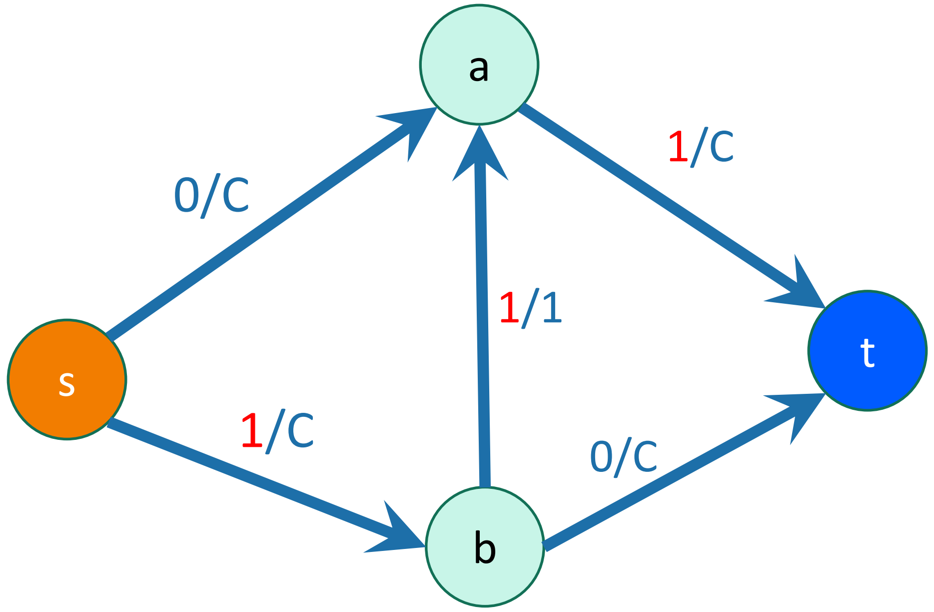
Choose a big number C.



graph G_f

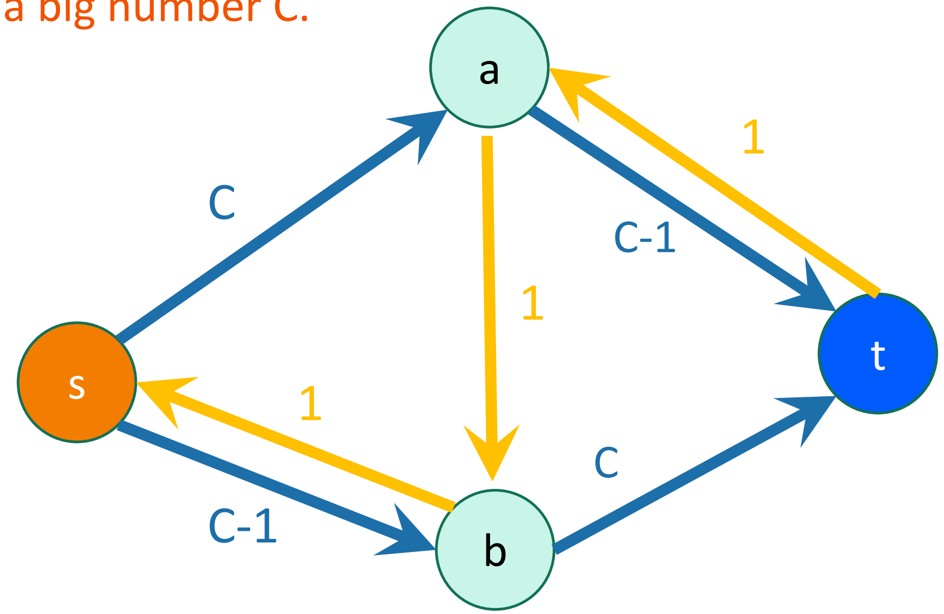
Running Time?

Suppose we just picked paths arbitrarily.



graph G

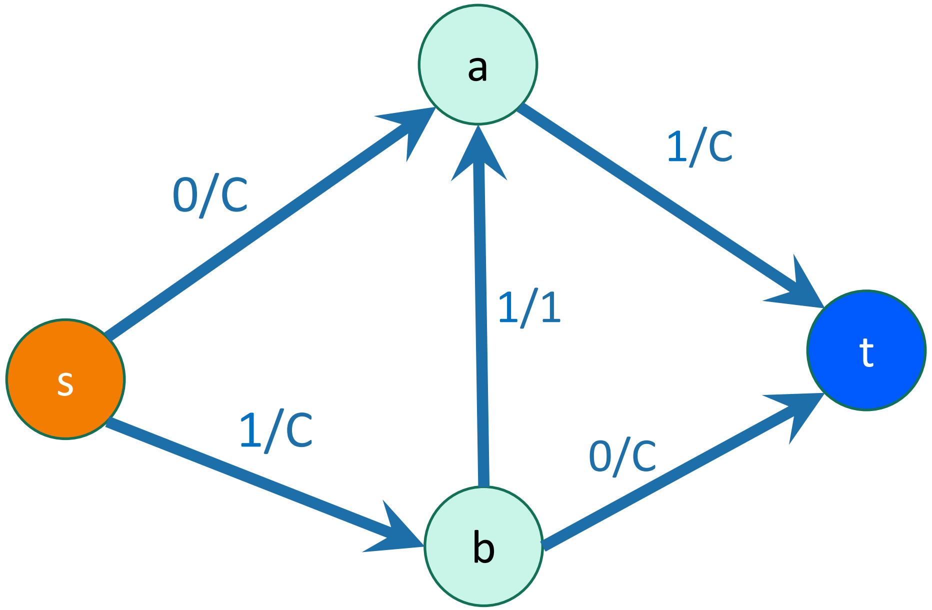
Choose a big number C.



graph G_f

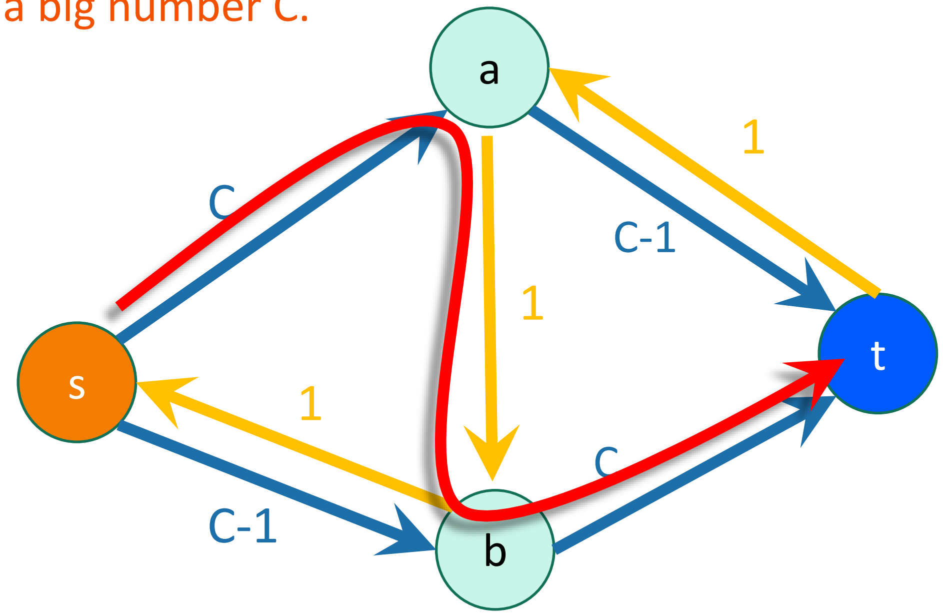
Running Time?

Suppose we just picked paths arbitrarily.



graph G

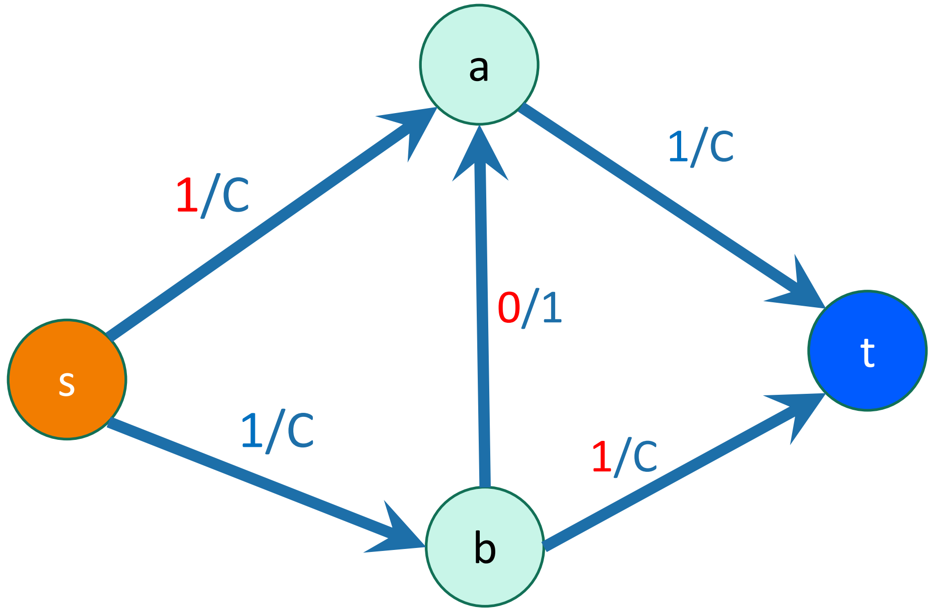
Choose a big number C.



graph G_f

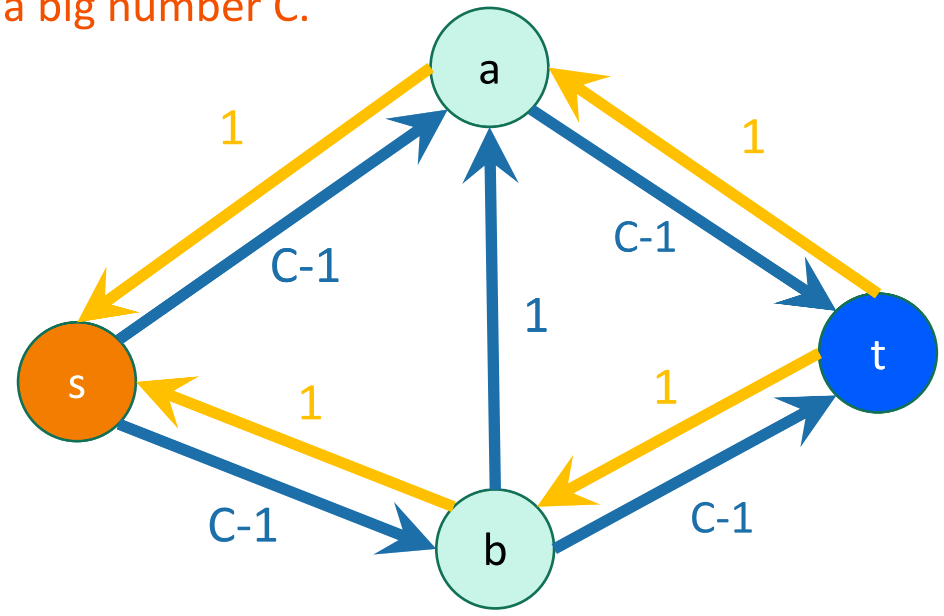
Running Time?

Suppose we just picked paths arbitrarily.



graph G

Choose a big number C.

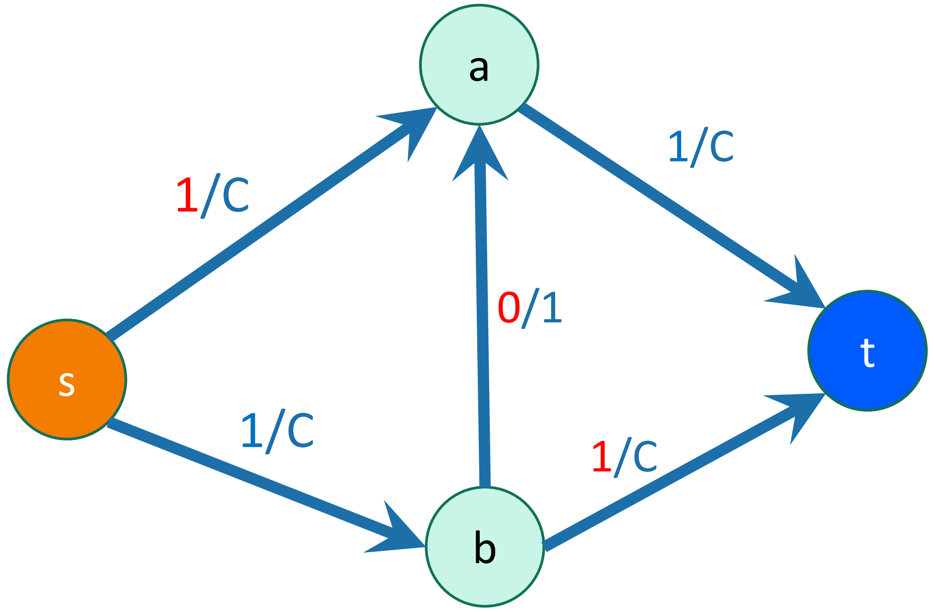


graph G_f

Running Time?

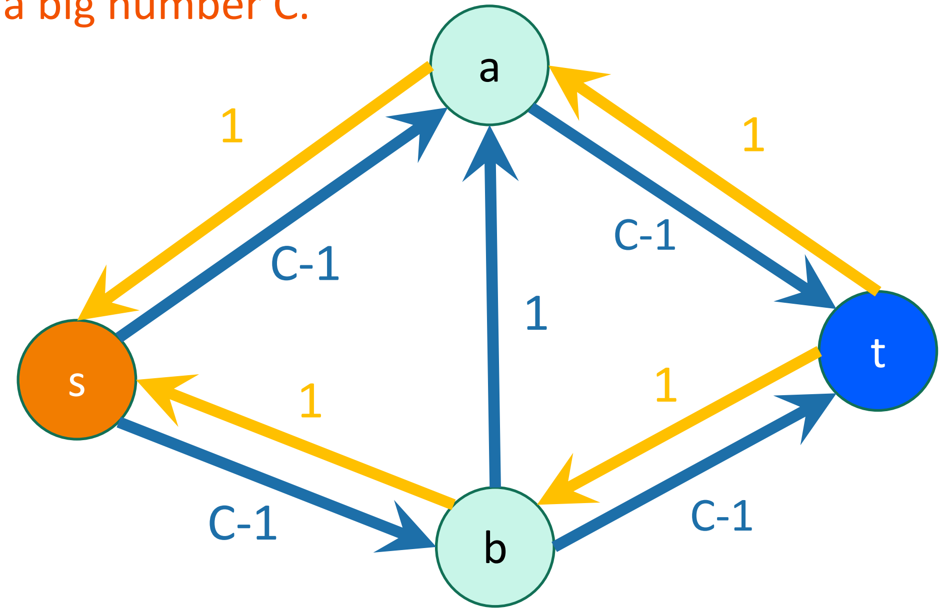
Suppose we just picked paths arbitrarily.

This will go on for O(C) steps, adding flow along (b,a) and then subtracting it again.



graph G

Choose a big number C.



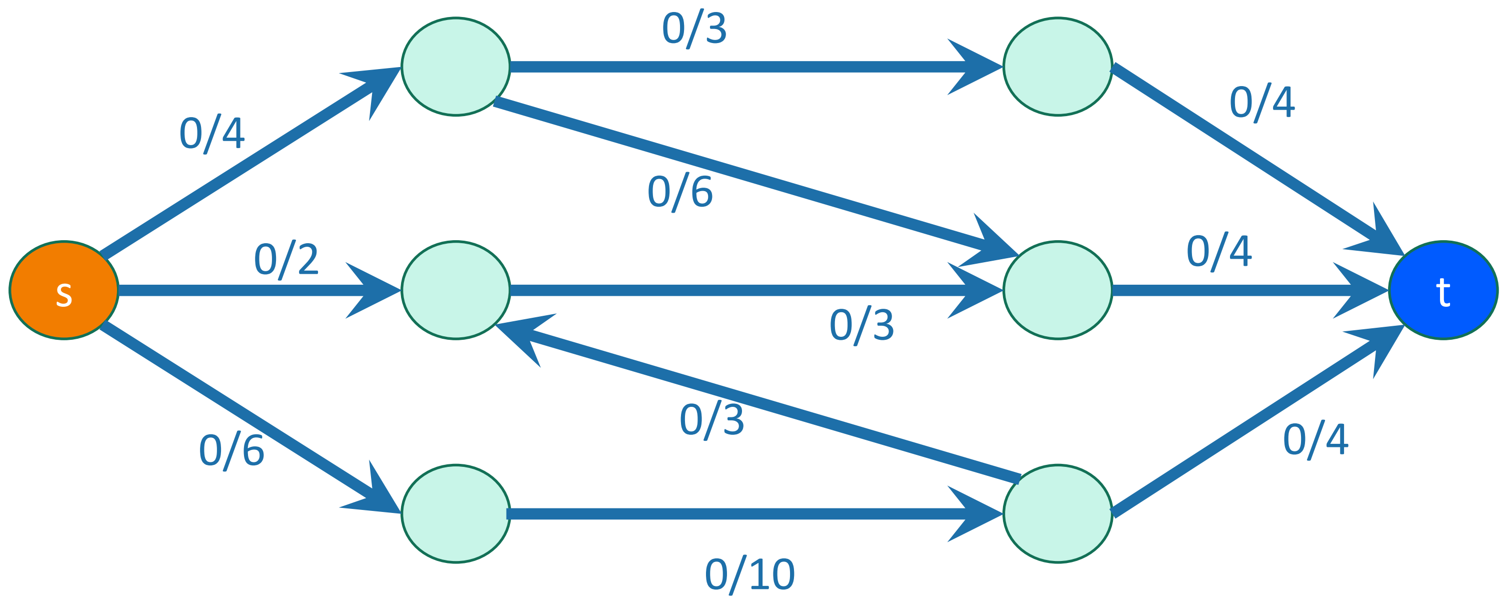
graph G_f

Time Complexity

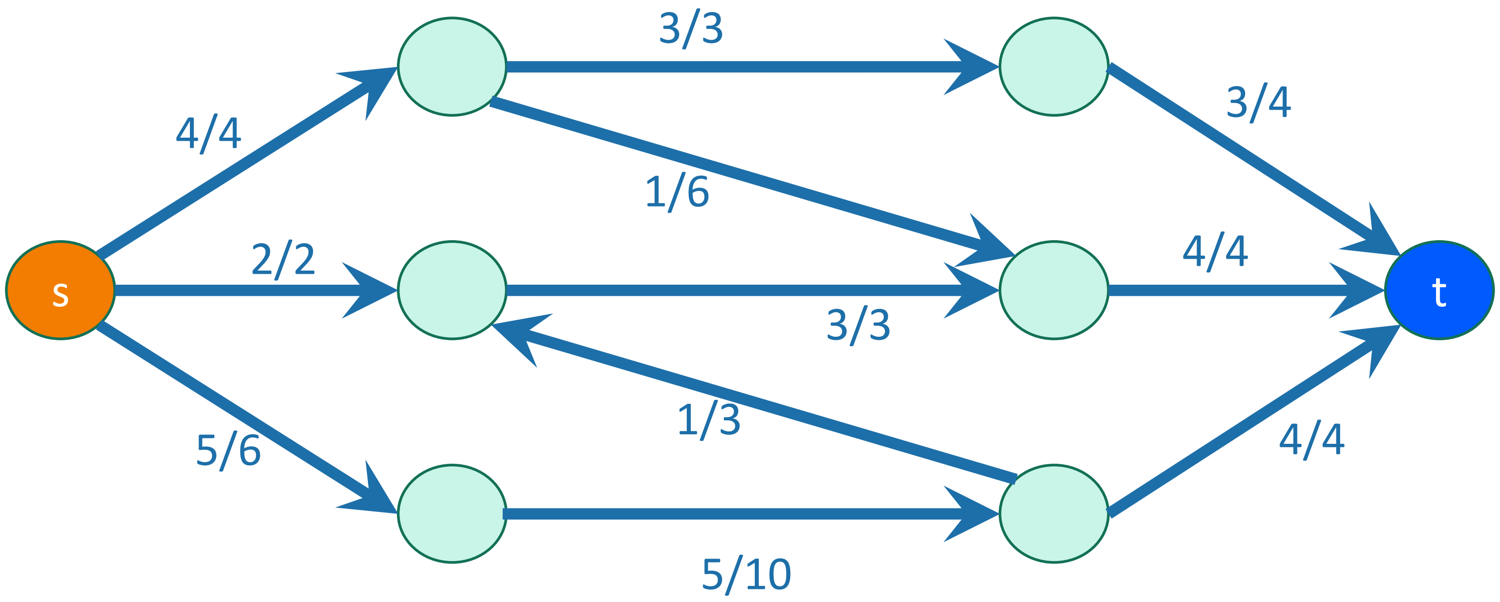
- Ford-Fulkerson algorithm: $O(Cm)$
  - C is the maximum flow value

- Doing Ford-Fulkerson with BFS is called the Edmonds-Karp algorithm. The Edmonds-Karp algorithm runs in $O(nm^2)$.
  - Please refer to the textbook for proof.

Exercise

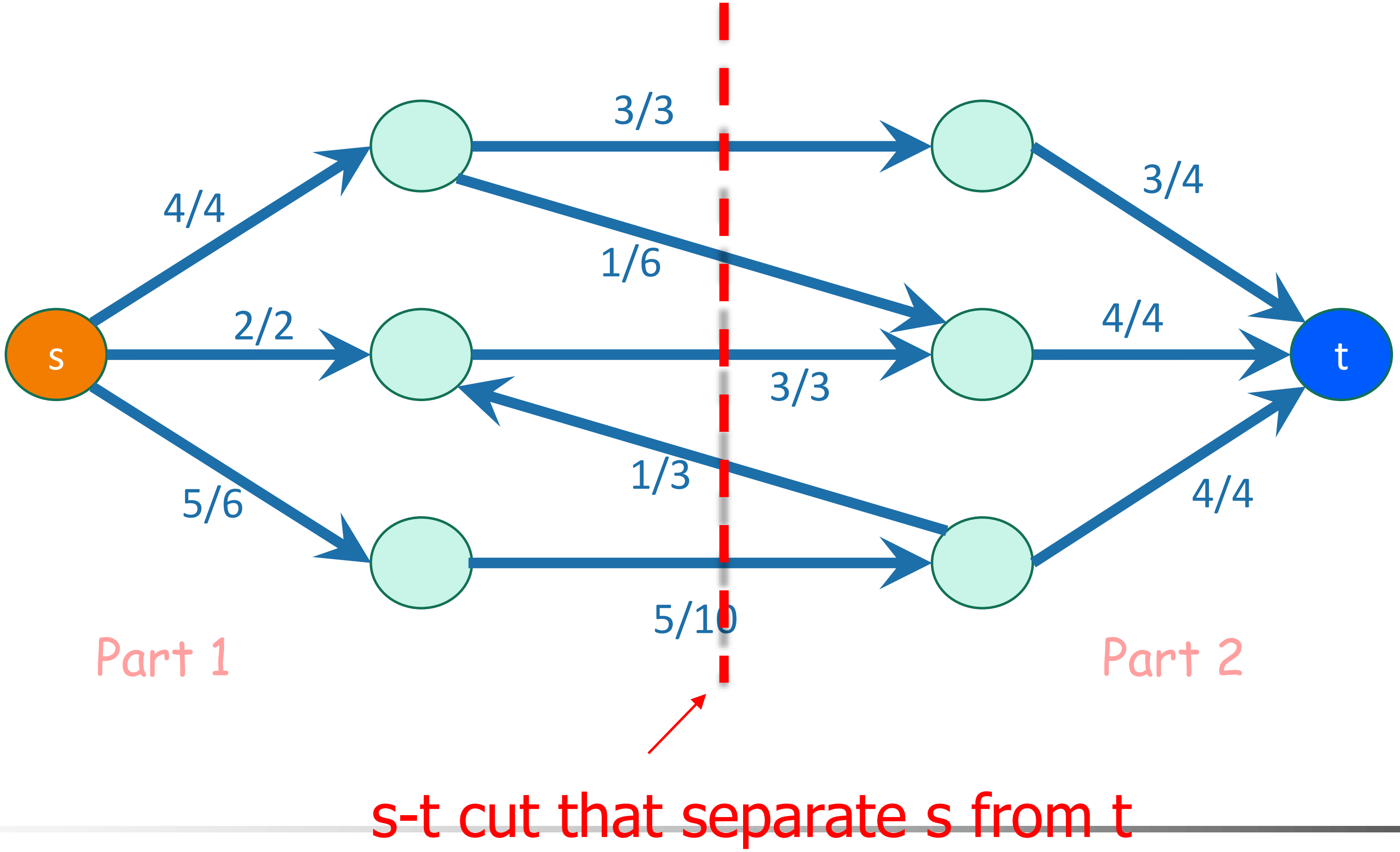


Exercise



What is a Cut?

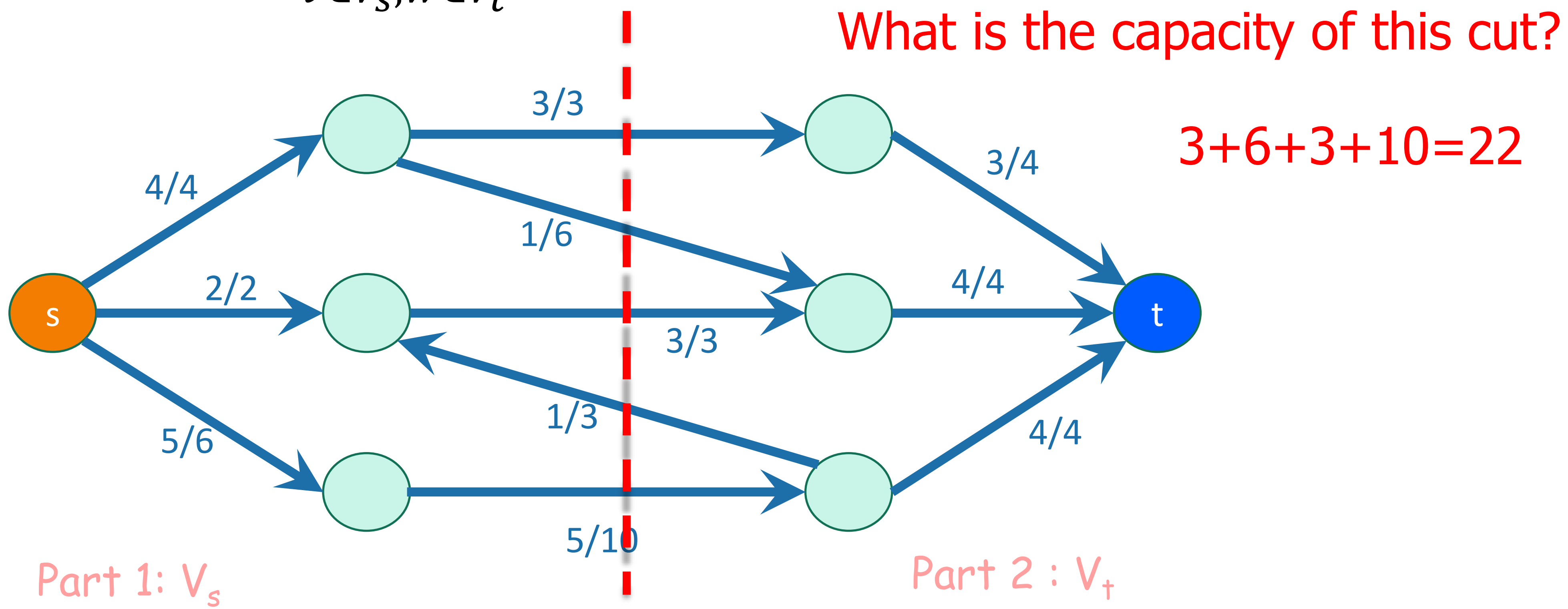
- A cut is a partition of the vertices into two nonempty parts.



Cut

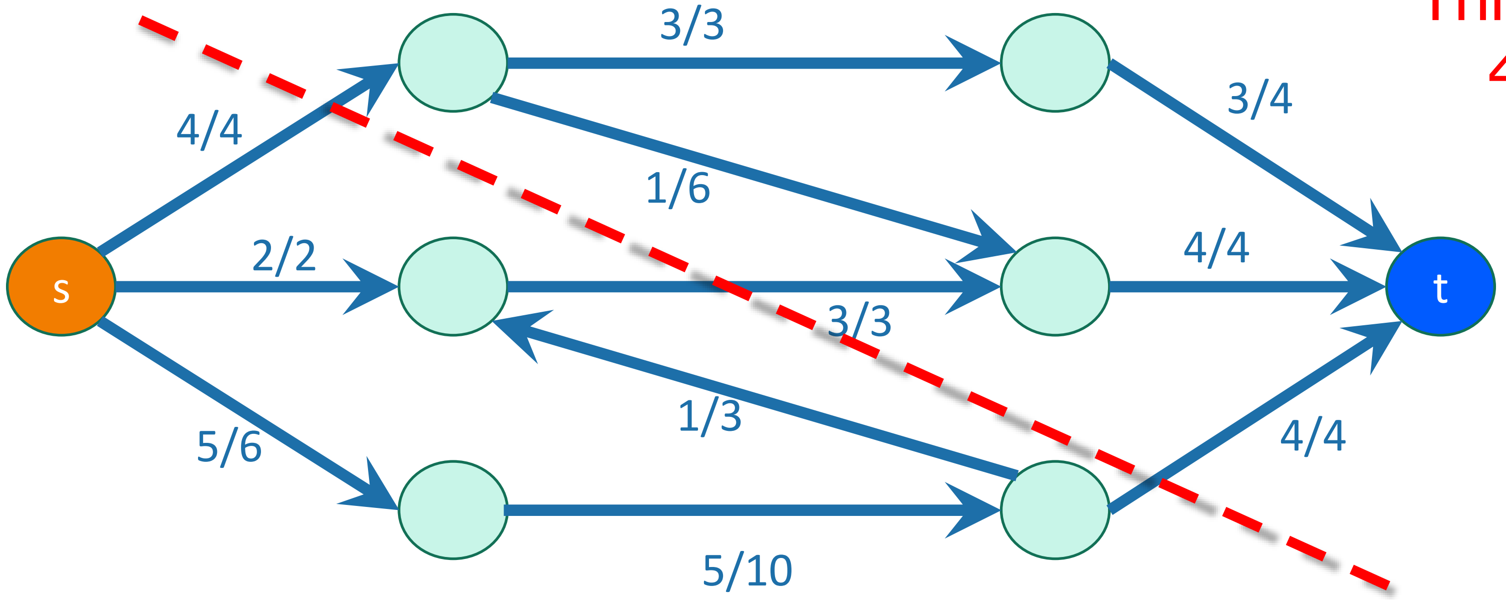
- Capacity of a Cut:

$$\sum_{v \in V_s, w \in V_t} capacity(v, w)$$



Minimum cut

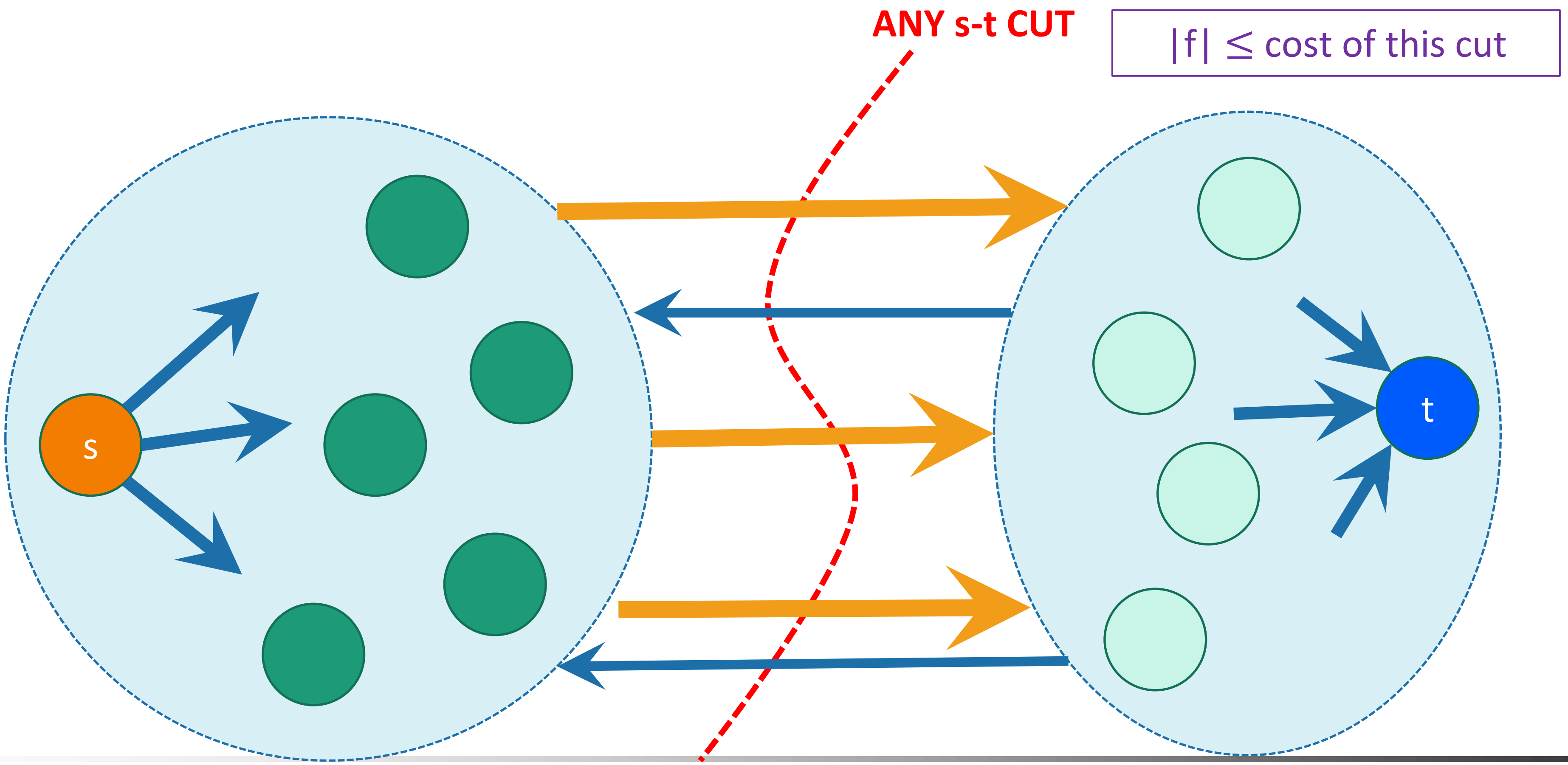
- Minimum cut has minimum capacity.
- Question: how to find a minimum cut?



This cut has cost:
4+3+4 = 11

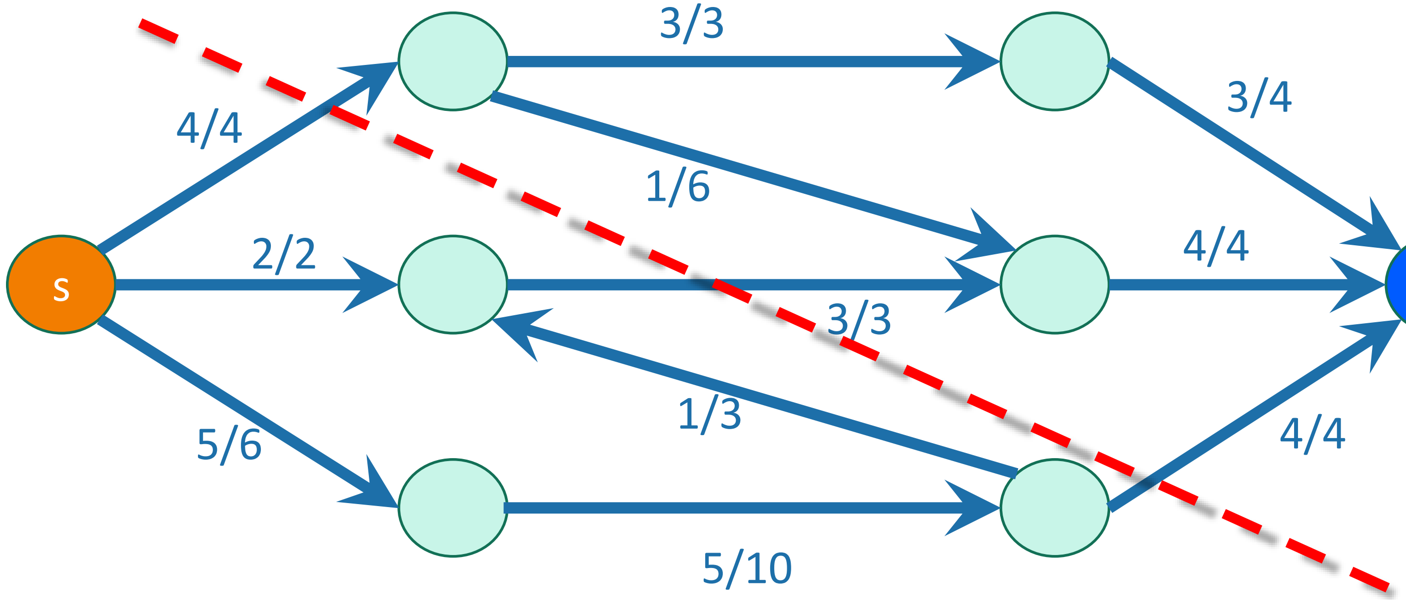
Maximum Flow Minimum Cut theorem

Value of maximum flow = capacity of minimum cut



Maximum Flow Minimum Cut theorem

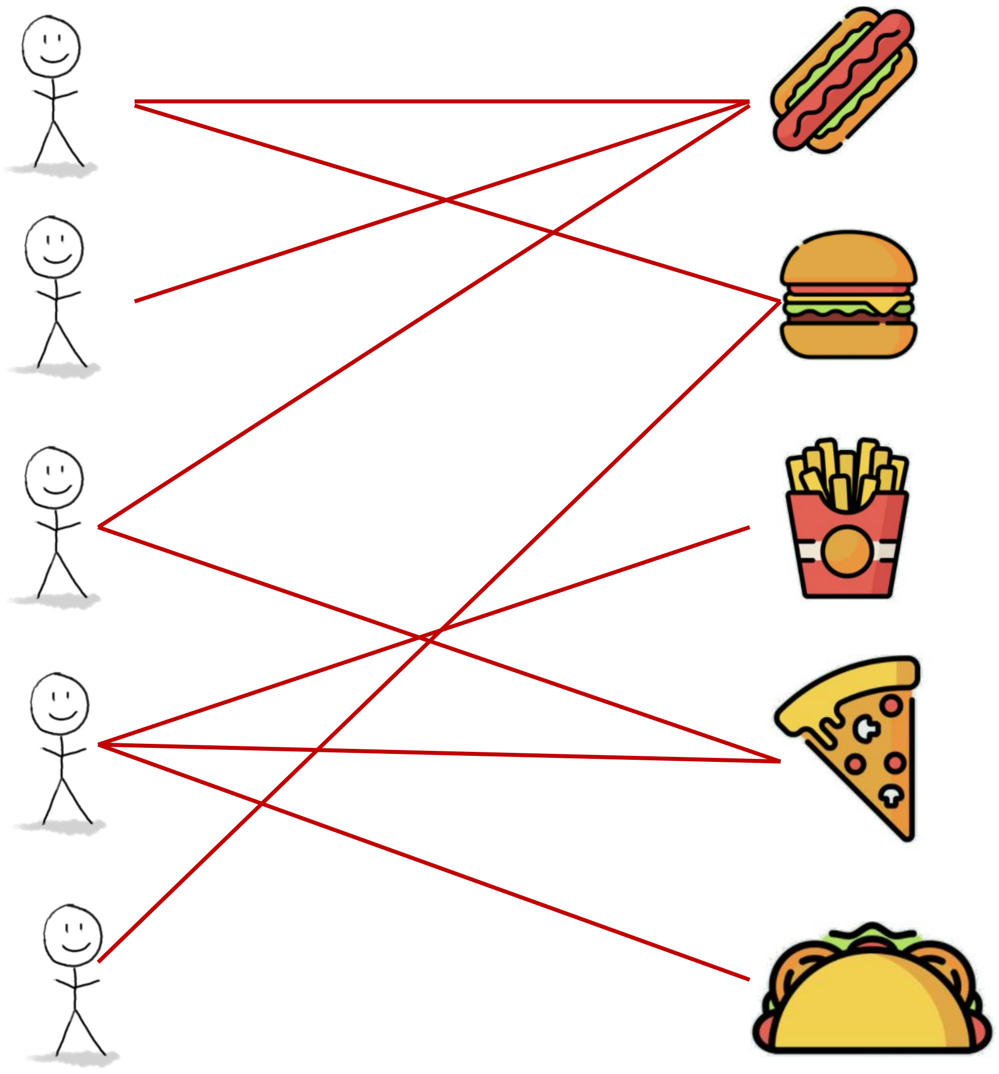
- Minimum cut cost: 11
- Maximum flow: 11



Maximum Bipartite Matching Example

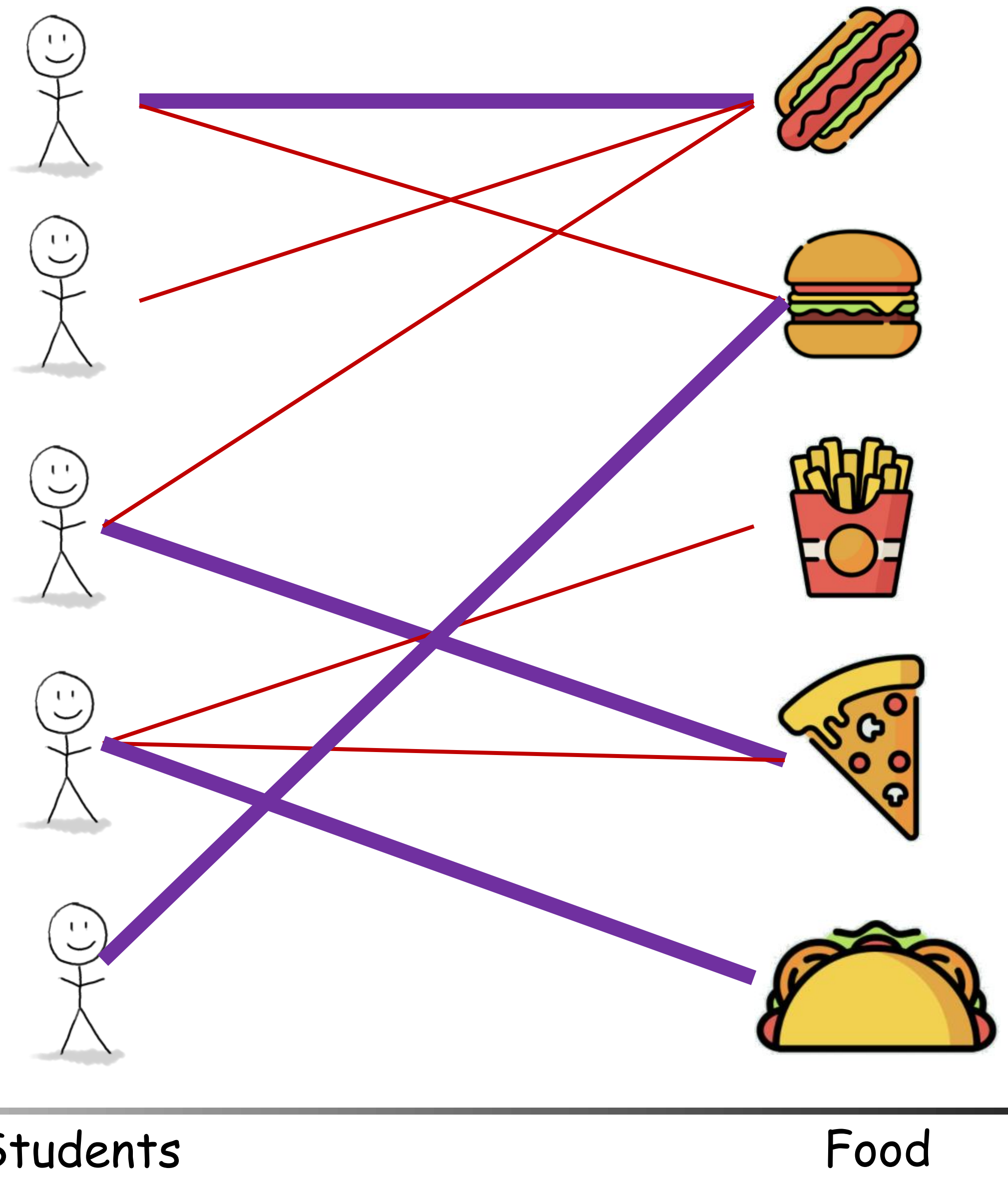
- Different students want different types of food

- How can we make as many students as possible happy?



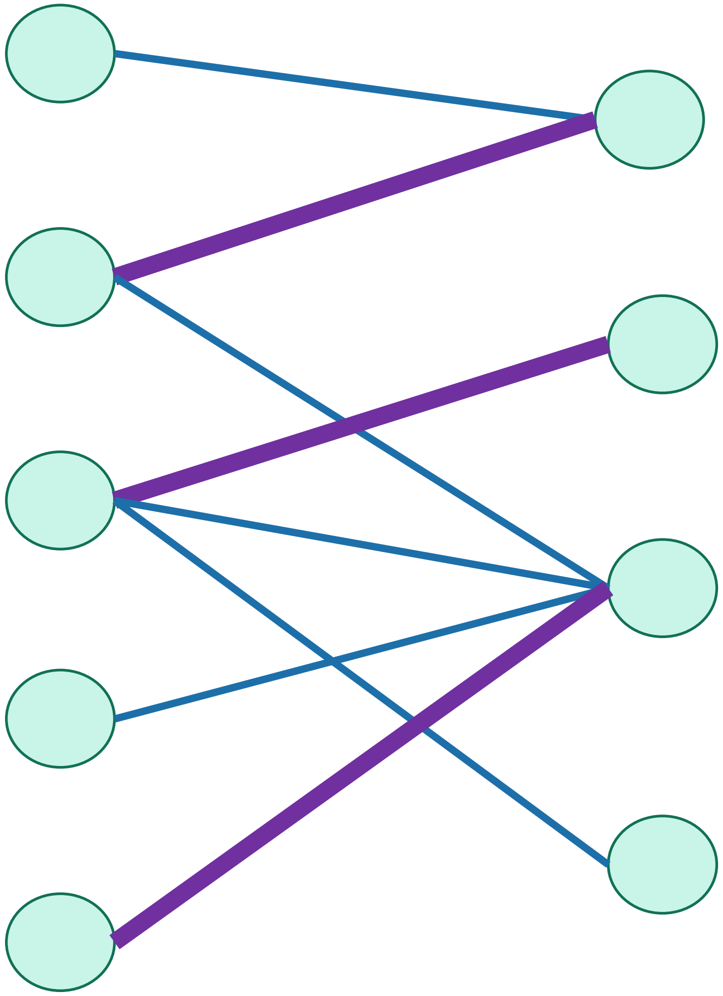
Maximum Bipartite Matching Example

- Different students want different types of food
- How can we make as many students as possible happy?



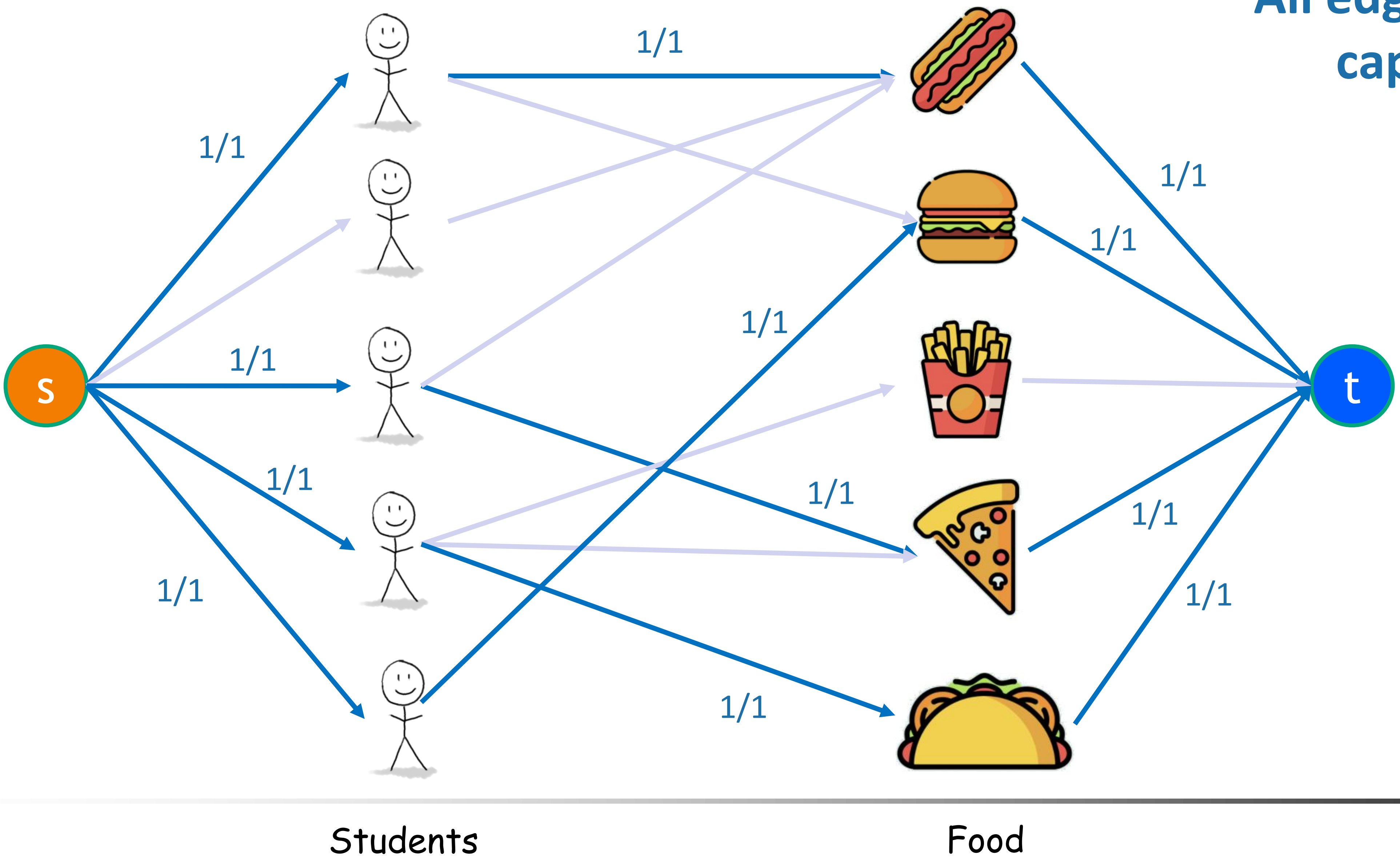
Maximum Bipartite Matching

- An undirected graph G is a bipartite graph if V = L ∪ R and all edges are between L and R (no edges within L or within R).
- A matching is a subset of edges M ∈ E s.t. no two edges share a vertex.
- A maximum matching is a matching with maximum cardinality.



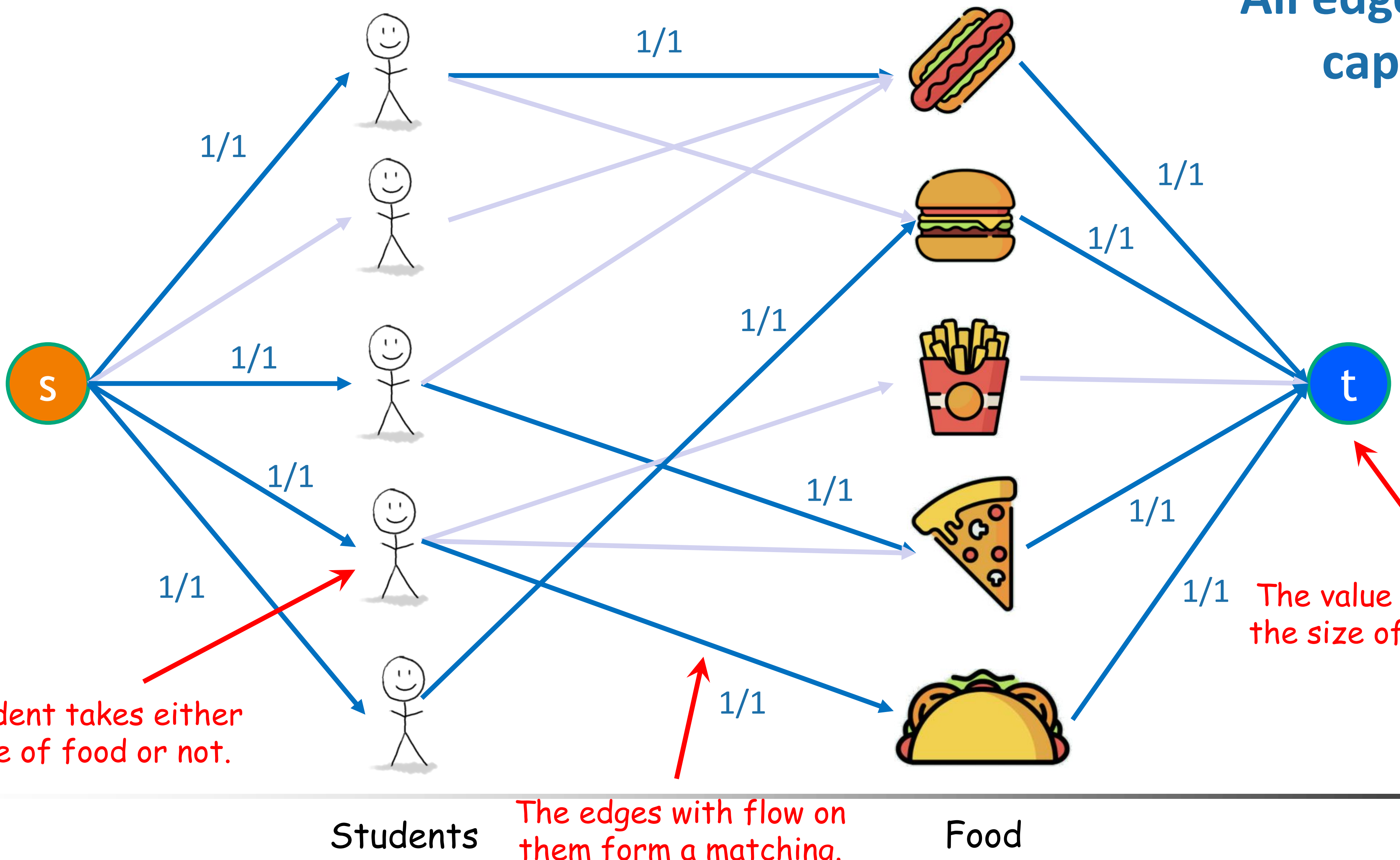
Solution via max flow

All edges have capacity 1.



Solution via max flow

All edges have capacity 1.

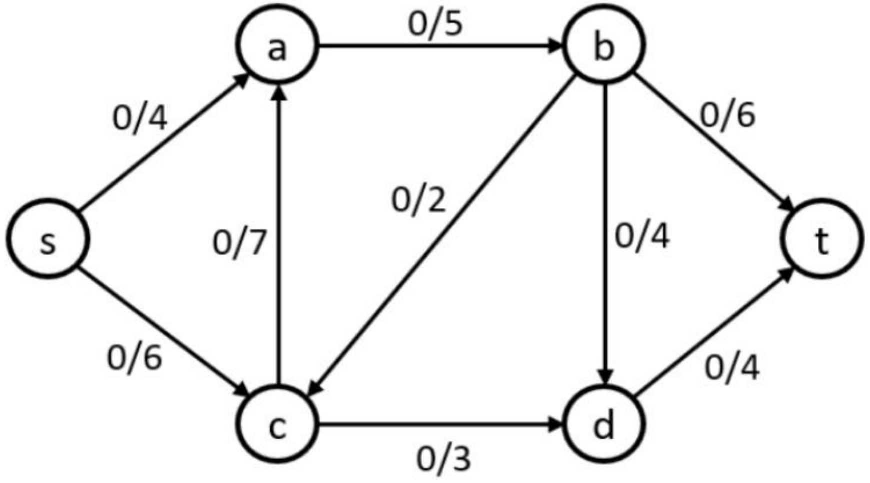


One student takes either one type of food or not.

The edges with flow on them form a matching.

The value of the flow is the size of the matching

Exercise



i. Use Ford-Fulkerson method to find the maximum flow of the network. For each step of the algorithm, report the augmenting path by listing its vertices, its residual capacity, and the resulting augmented flow on the network. **[9 marks]**

ii. Show the flow network that gives the maximum flow and draw the minimum cut. **[4 marks]**

Exercise

- Implement Ford-Fulkerson algorithm with Python
- Optional: Maximum Bipartite Matching

Learning Outcome

- Flow networks
- Maximum Flow
- Ford-Fulkerson Method
- Minimum cut
- Maximum bipartite matching