MONASH University

MONASH
INFORMATION
TECHNOLOGY

FIT2004
Algorithms and Data Structures

Ian Wern Han Lim
lim.wern.han@monash.edu

Referencing materials by
Nathan Companez, Aamir Cheema, Arun Konagurthu and Lloyd Allison

GROUP OF EIGHT AUSTRALIA

Faculty of Information Technology, Monash University

COMMONWEALTH OF AUSTRALIA
Copyright Regulations 1969
This material has been reproduced and communicated to you by or on behalf of Monash University pursuant to Part VB of the Copyright Act 1968 (the Act). The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act. Do not remove this notice

MONASH University

Ready?

Agenda

- Network Flow

Agenda

- Network Flow

- The maximum flow problem
- The residual network
- Path augmentation

Agenda

- Network Flow

- The maximum flow problem
- The residual network
- Path augmentation

Ford-Fulkerson Method

Agenda

- Network Flow

- The maximum flow problem
- The residual network
- Path augmentation

Ford-Fulkerson Method

- Min-cut Max-flow Theorem

Agenda

- Network Flow

- The maximum flow problem
- The residual network
- Path augmentation

Ford-Fulkerson Method

- Min-cut Max-flow Theorem

- Then we have Bipartite Graph

Agenda

- Network Flow

- The maximum flow problem
- The residual network
- Path augmentation

Ford-Fulkerson Method with extra optimization from FIT3155

- Min-cut Max-flow Theorem

- Then we have Bipartite Graph
  - Matching optimally =)

MONASH University

Let us begin…

Flow Network

Transfer of content

MONASH University

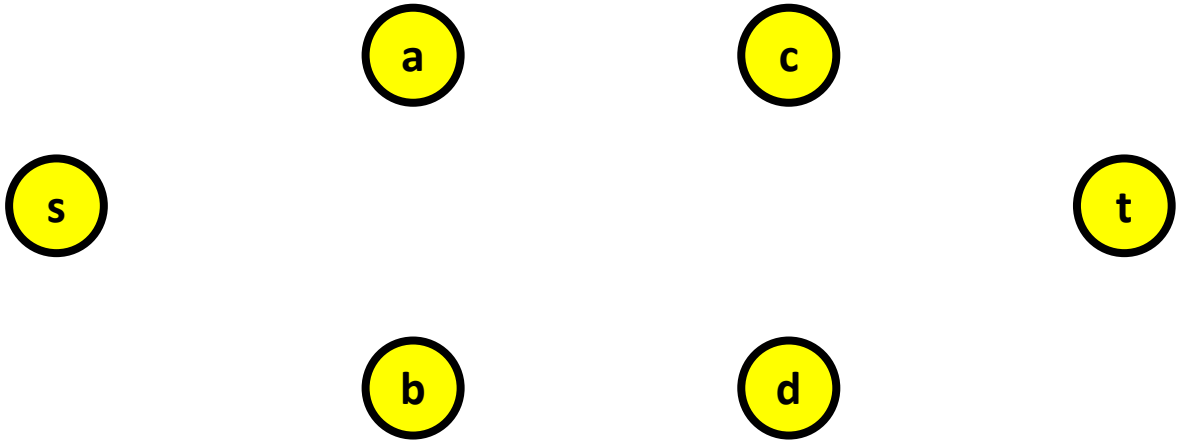
- What is it?

Flow Network

Transfer of content

MONASH University

- What is it?
  - It is a graph
    - With vertices

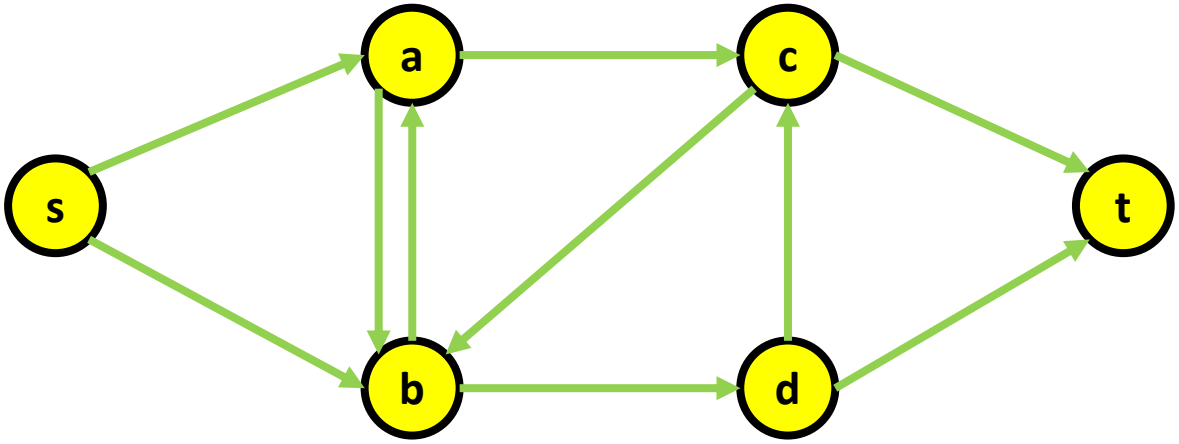


Flow Network

Transfer of content

MONASH University

- What is it?
  - It is a graph
    - With vertices
    - With edges (directed)

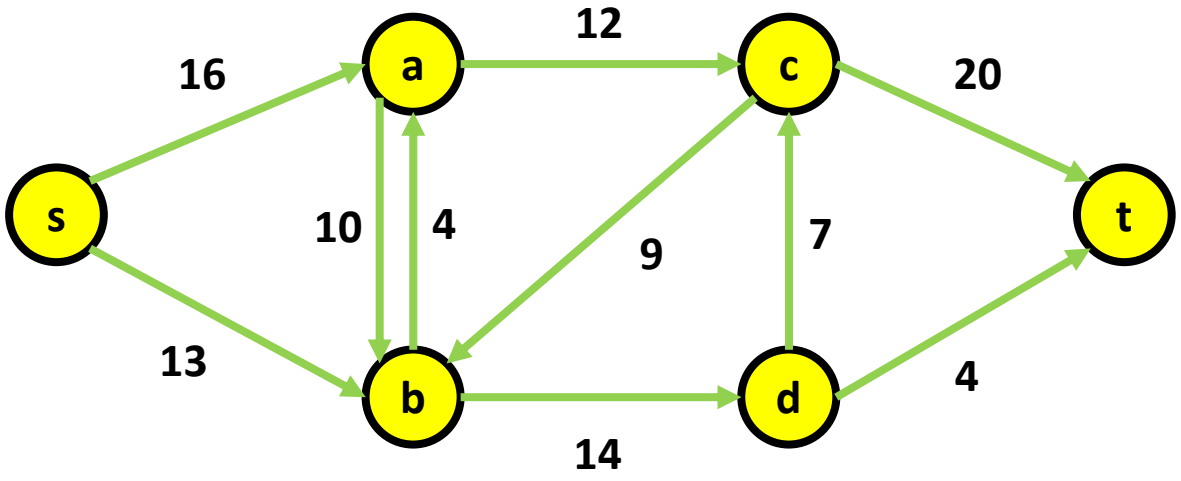


Flow Network

Transfer of content

MONASH University

- What is it?
  - It is a graph
    - With vertices
    - With edges (directed and weighted)

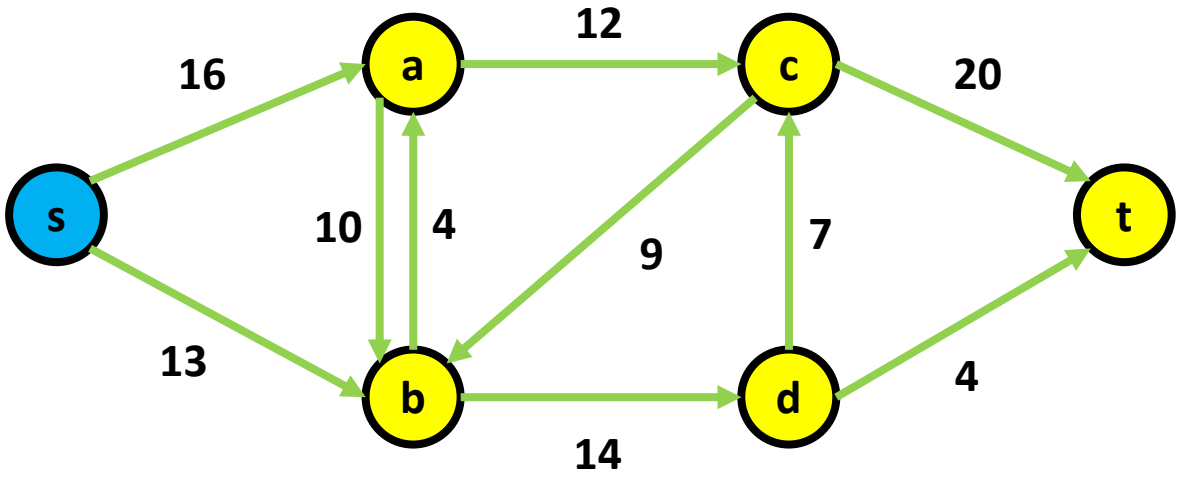


Flow Network

Transfer of content

MONASH University

- What is it?
  - It is a graph
    - With vertices
    - With edges (directed and weighted)
    - A vertex without incoming edges

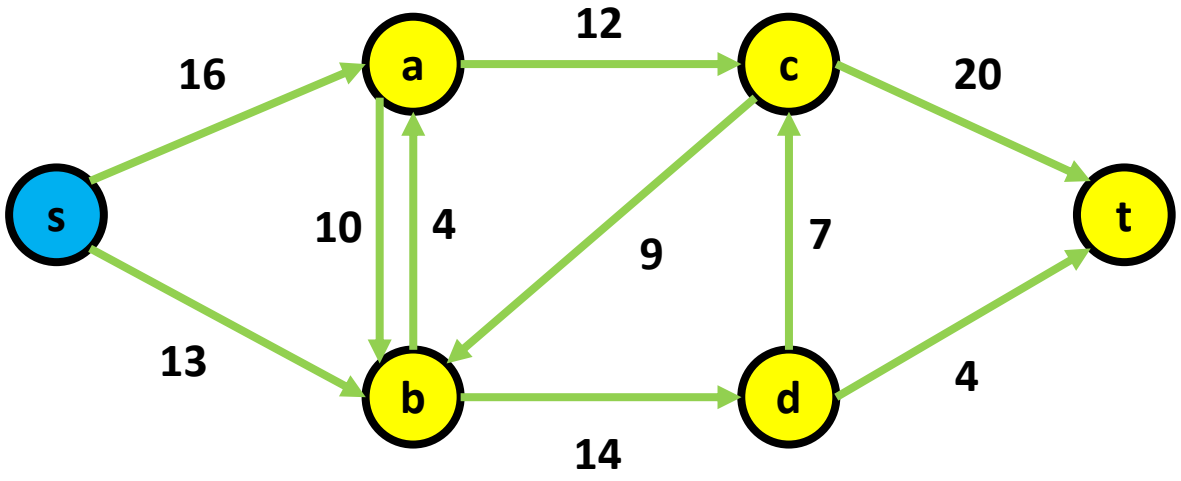


Flow Network

Transfer of content

MONASH University

- What is it?
  - It is a graph
    - With vertices
    - With edges (directed and weighted)
    - A vertex without incoming edges know as the source

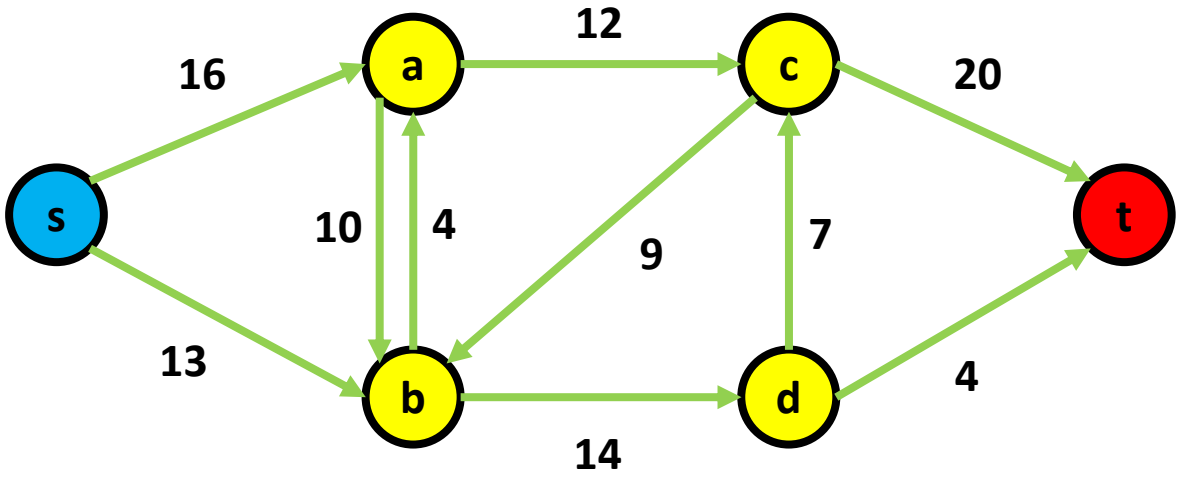


Flow Network

Transfer of content

MONASH University

- What is it?
  - It is a graph
    - With vertices
    - With edges (directed and weighted)
    - A vertex without incoming edges know as the source
    - A vertex without outgoing edges

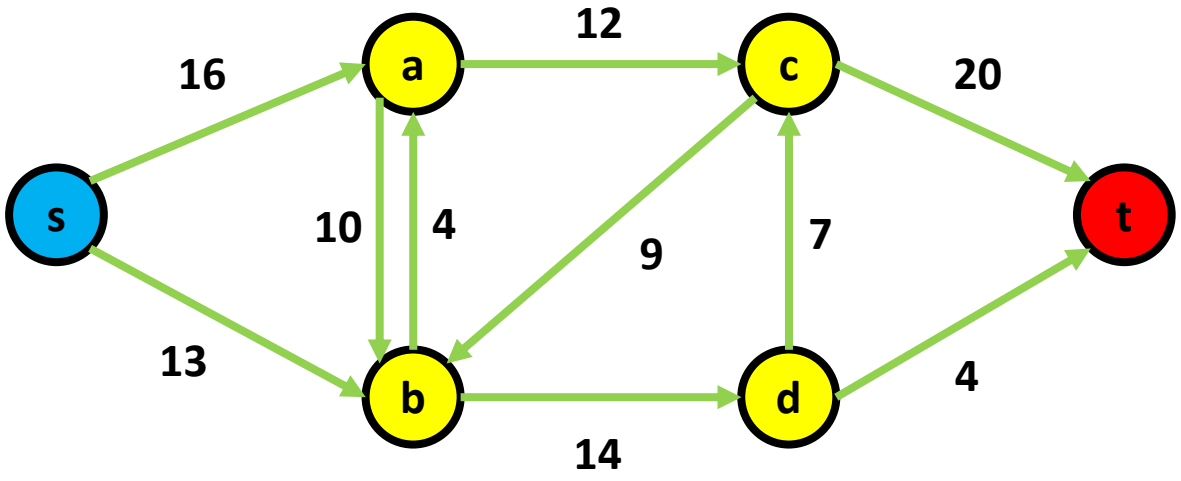


Flow Network

Transfer of content

MONASH University

- What is it?
  - It is a graph
    - With vertices
    - With edges (directed and weighted)
    - A vertex without incoming edges know as the source
    - A vertex without outgoing edges known as the target/ destination

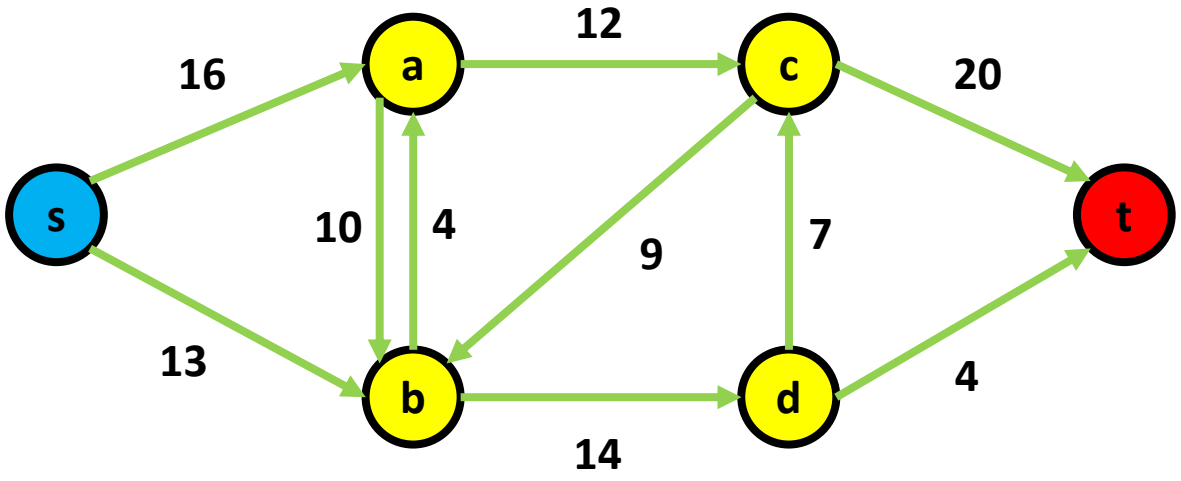


Flow Network

Transfer of content

MONASH University

- What is it?
  - It is a graph
    - With vertices
    - With edges (directed and weighted non-negative known as capacity)
    - A vertex without incoming edges know as the source
    - A vertex without outgoing edges known as the target/ destination

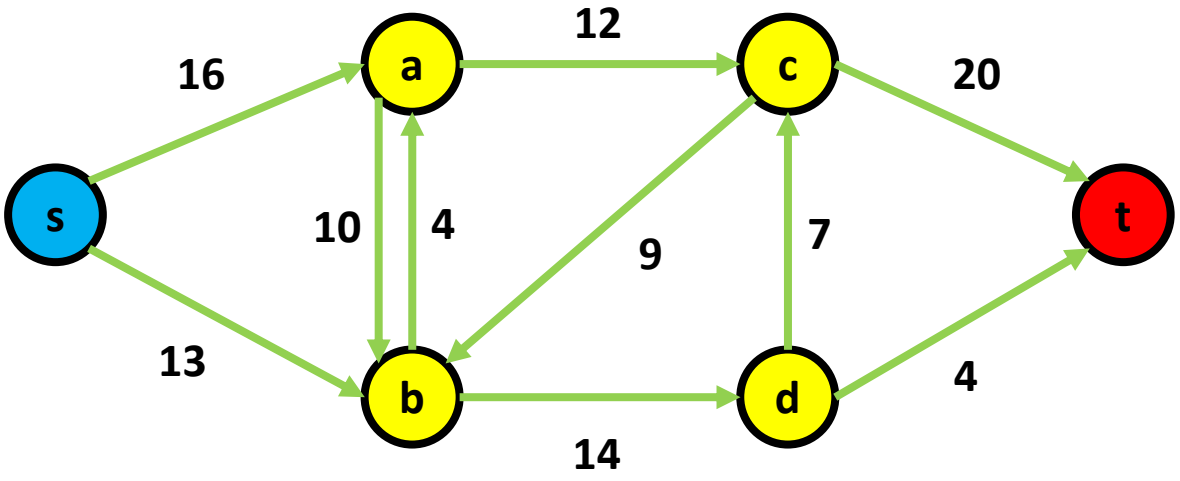


Flow Network

Transfer of content

MONASH University

- What is it?
  - Explore the real world problem of transfer
    - From source
    - To destination

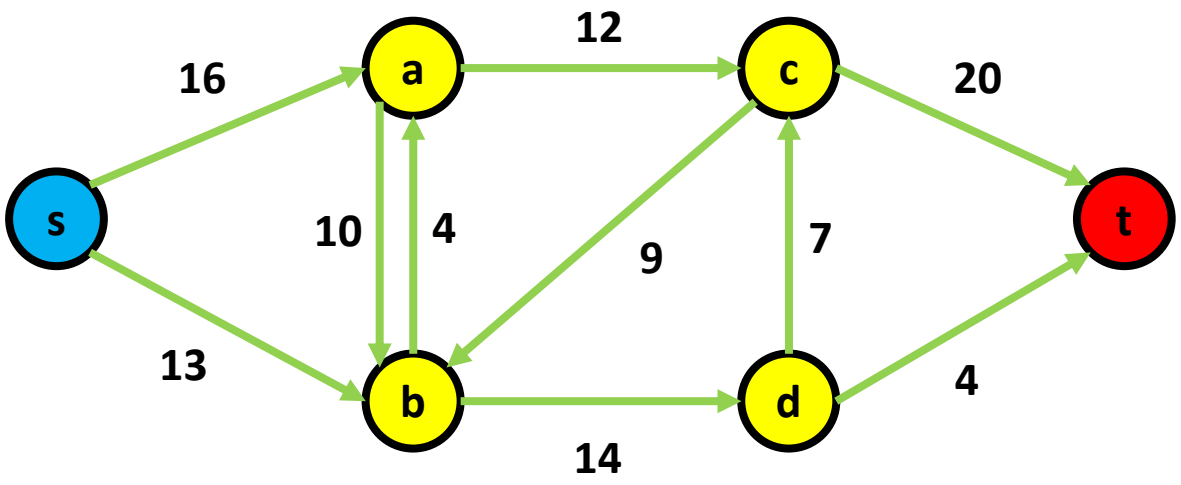


Flow Network

Transfer of content

MONASH University

- What is it?
  - Explore the real world problem of transfer
    - From source
    - To destination
    - Within the capacity (which can be bottlenecks)

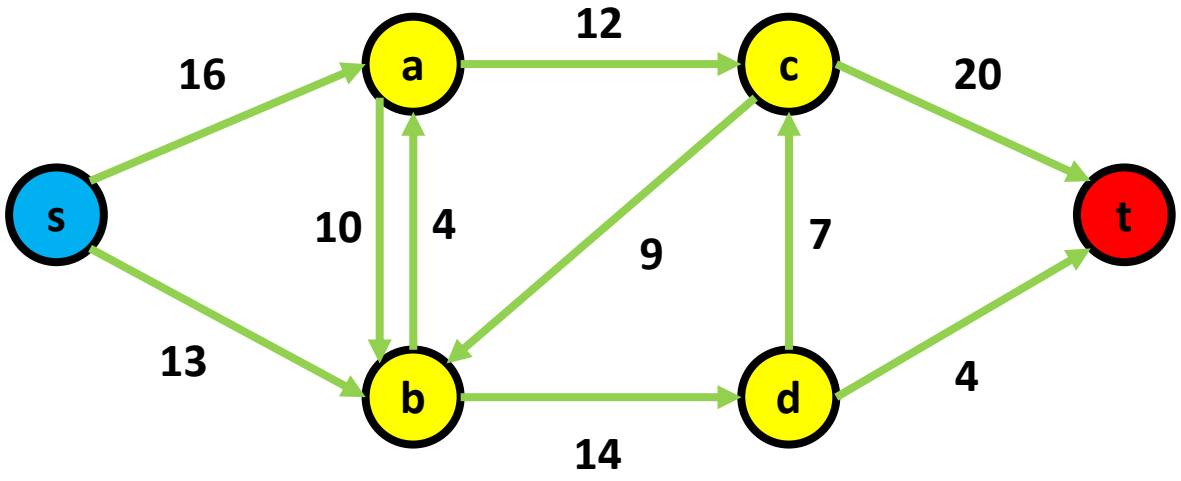


Flow Network

Transfer of content

MONASH University

- What is it?
  - Explore the real world problem of transfer
    - From source
    - To destination
    - Within the capacity (which can be bottlenecks)
    - What is the maximum possible transfer of content?

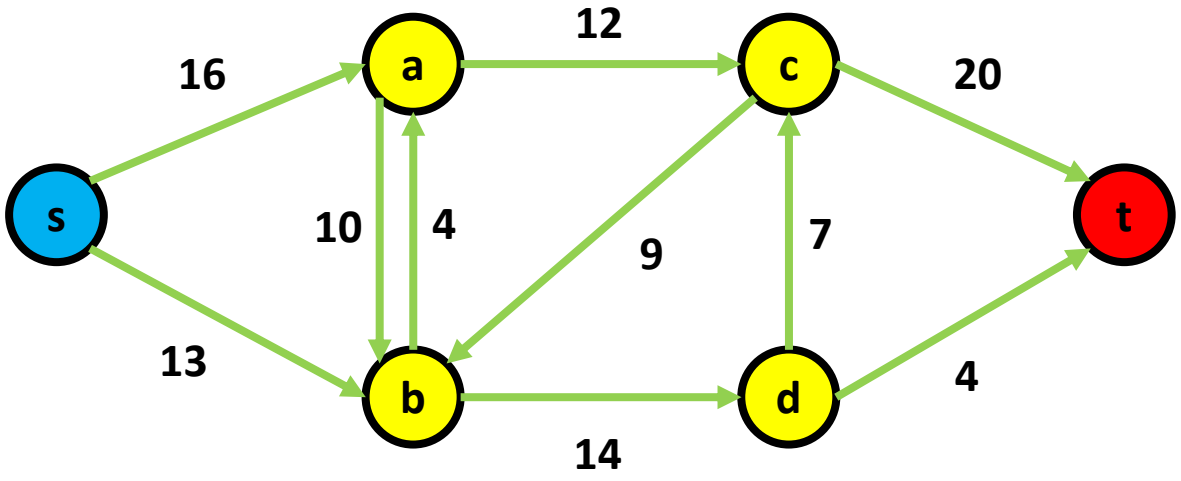


Flow Network

Transfer of content

MONASH University

- What is it?
  - Explore the real world problem of transfer
    - From source
    - To destination
    - Within the capacity (which can be bottlenecks)
    - What is the maximum possible transfer of content? Goal here

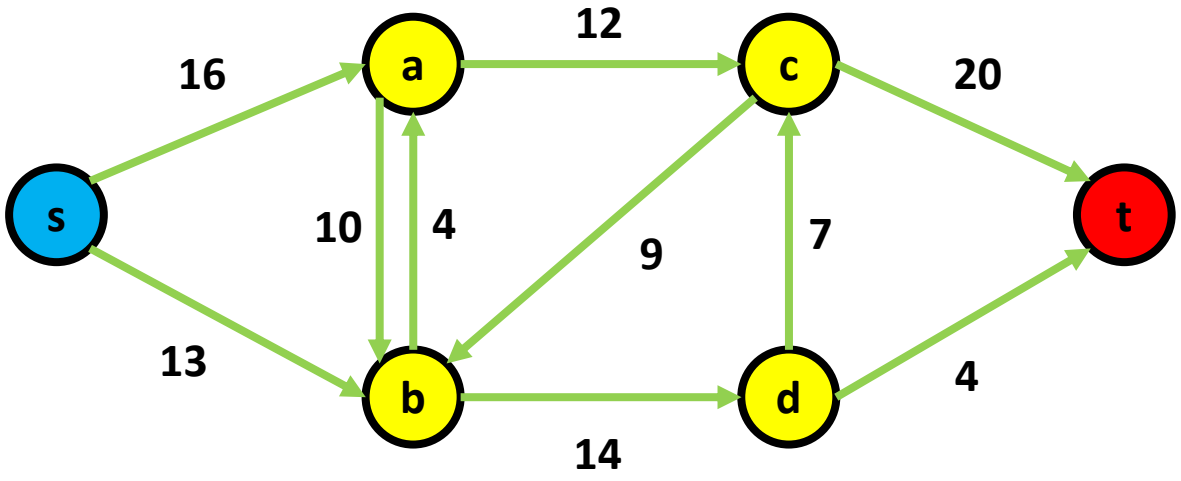


Flow Network

Transfer of content

MONASH University

- We can get details from this graph
  - E_in(b) = edges incoming to b
  - E_out(b) = edges outgoing from b

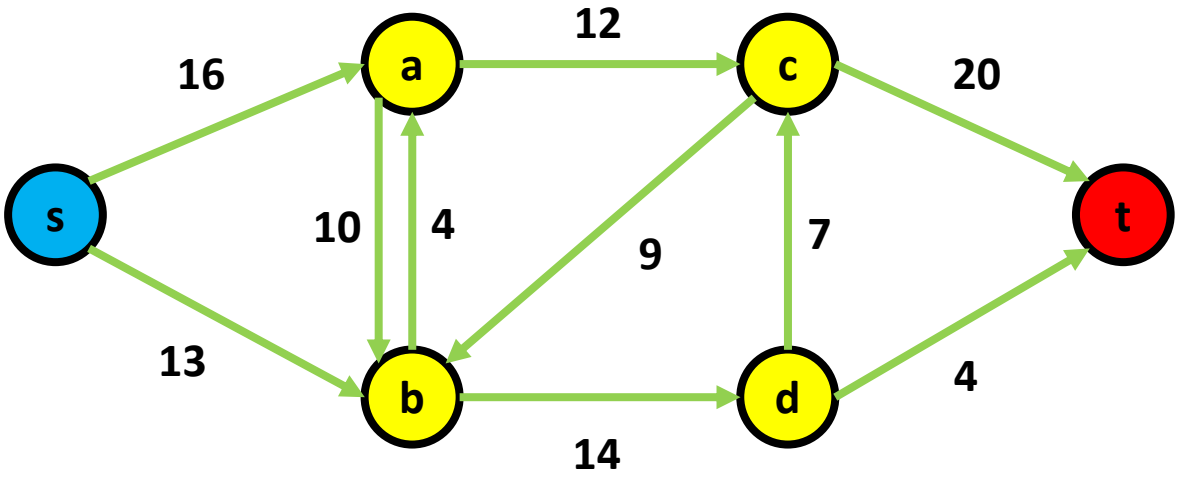


Flow Network

Transfer of content

MONASH University

- We can get details from this graph
  - E_in(b) = edges incoming to b = <s,b,13> <a,b,10> <c,b,9>
  - E_out(b) = edges outgoing from b = <b,a,4> <b,d,14>

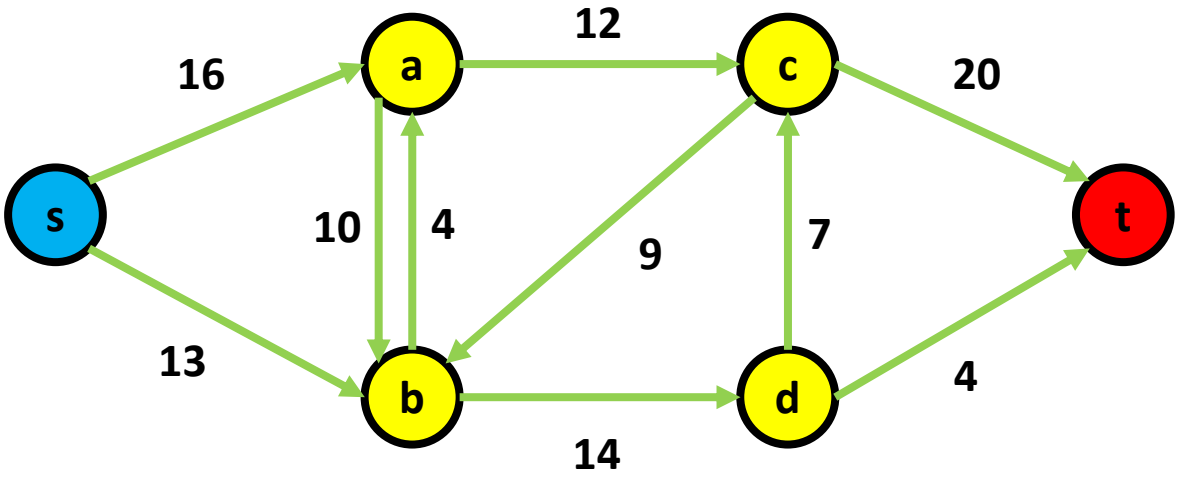


Flow Network

Transfer of content

MONASH University

- Flow network model in real world

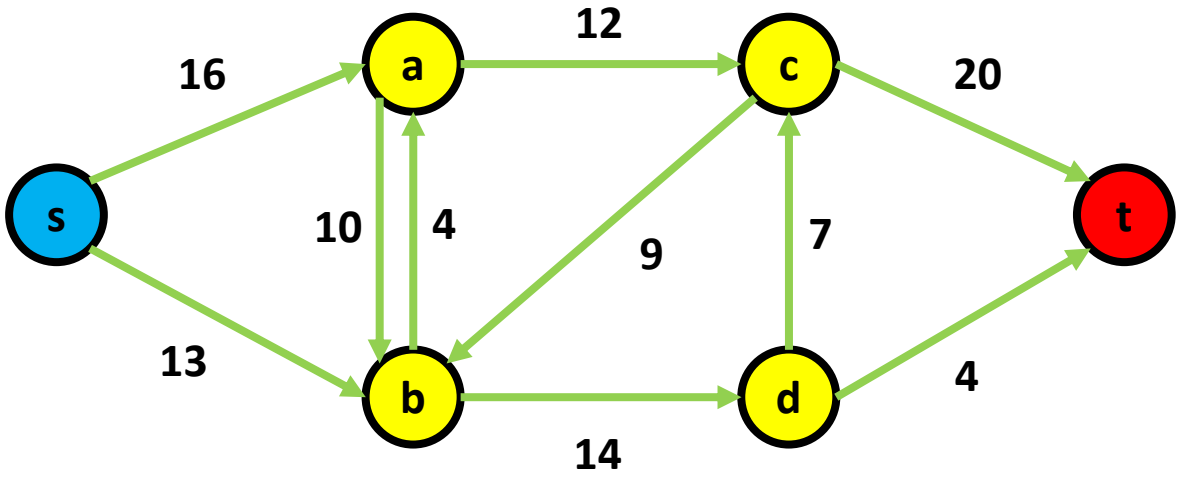


Flow Network

Transfer of content

MONASH University

- Flow network model in real world
  - Water flow through pipes
  - Electric through electrical circuits
  - Information flow through communication network

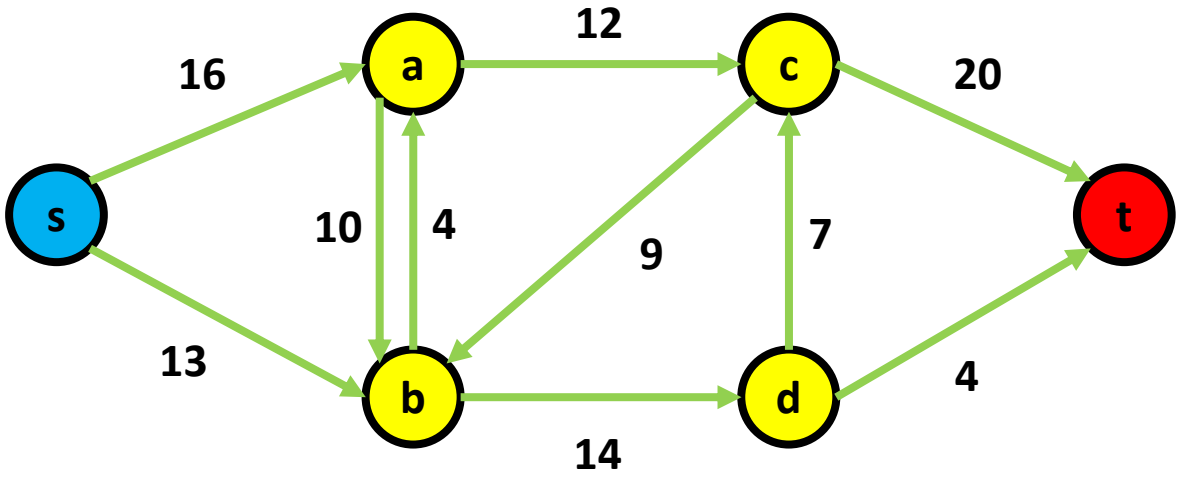


Flow Network

Transfer of content

MONASH University

- Flow network model in real world
  - Water flow through pipes
  - Electric through electrical circuits
  - Information flow through communication network
  - And many more! We can design good networks #engineered

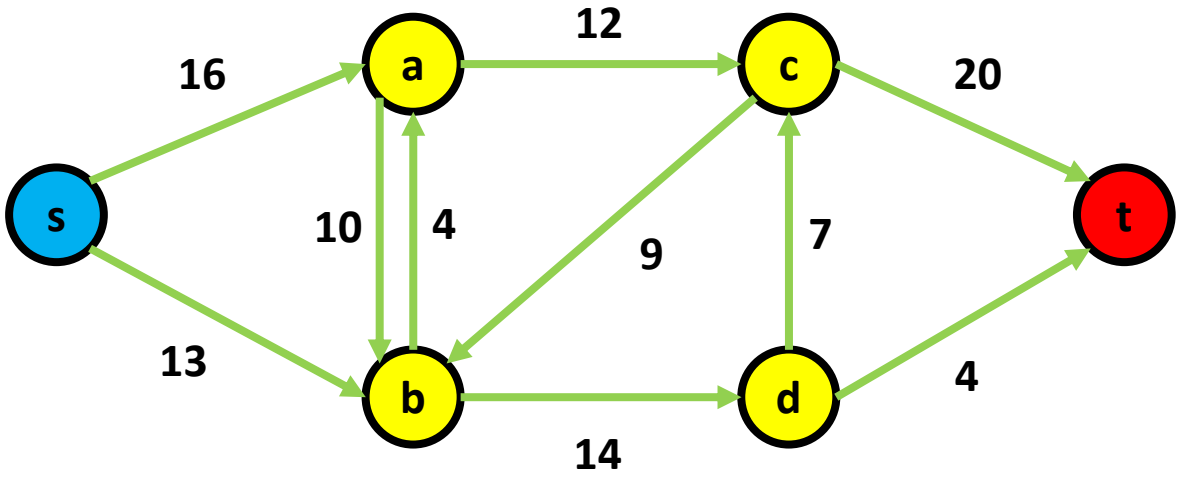


Flow Network

Transfer of content

MONASH University

- Flow network model in real world
  - Water flow through pipes
  - Electric through electrical circuits
  - Information flow through communication network
  - And many more! We can design good networks #engineered
  - Was mainly used in WW2 to disrupt enemy supply lines

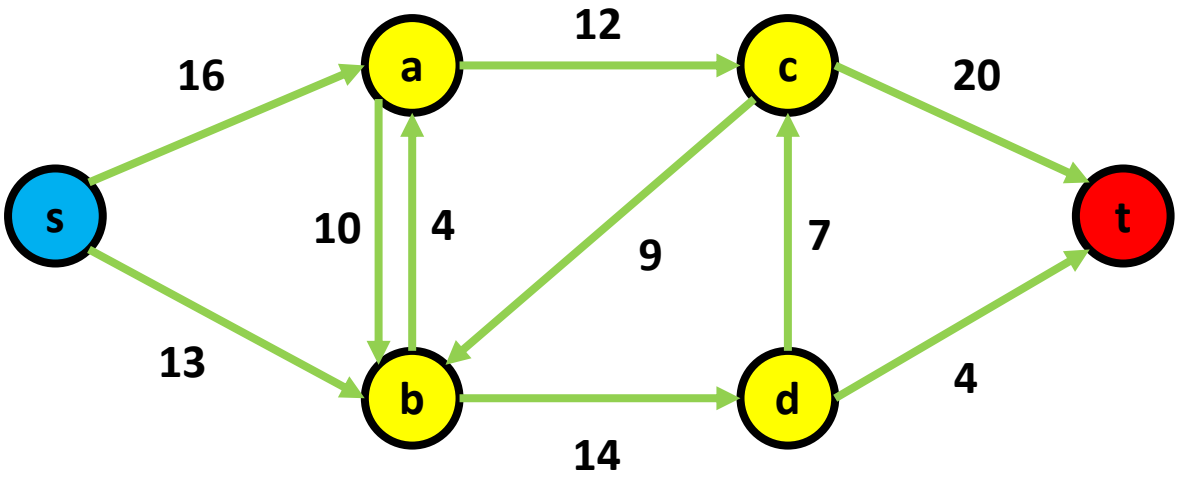


MONASH University

Questions?

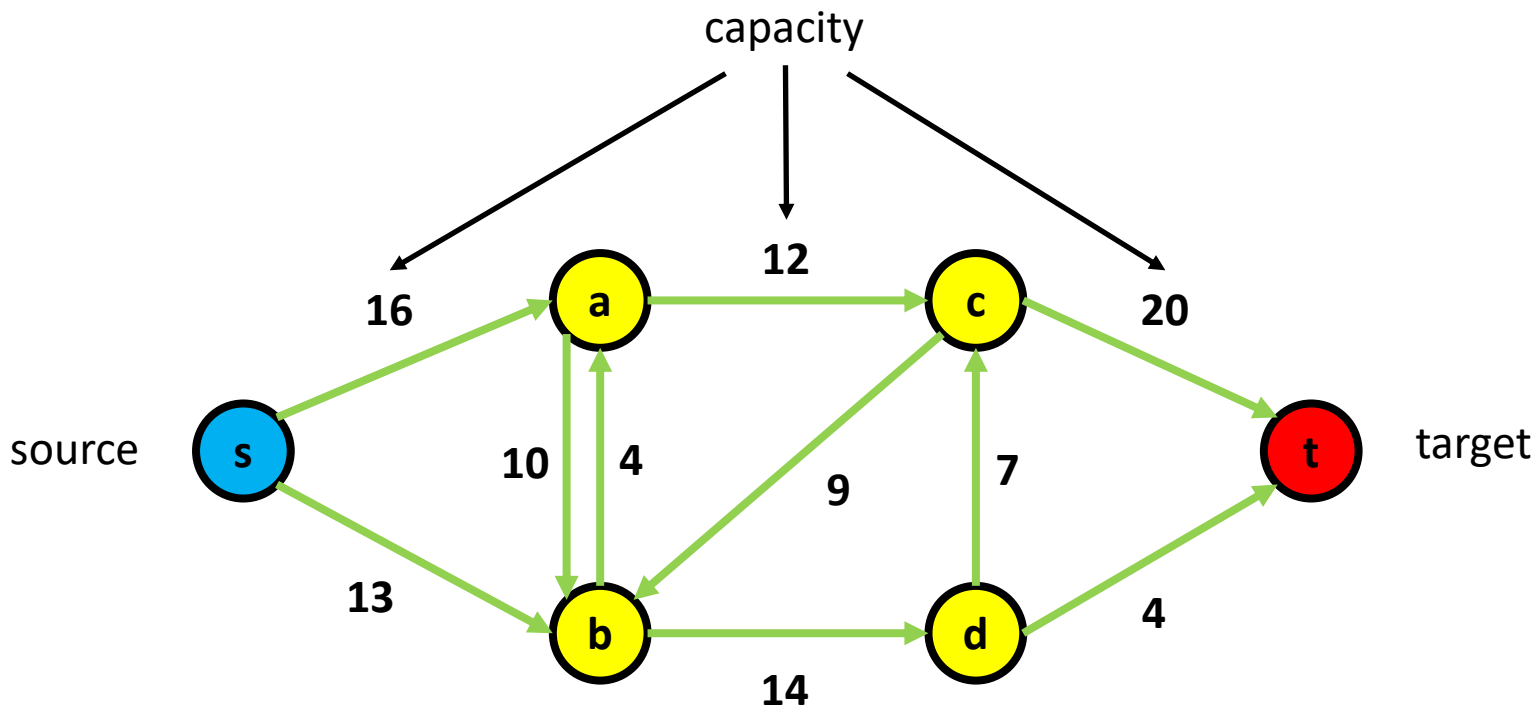
Recap

Of flow network



Recap

Of flow network

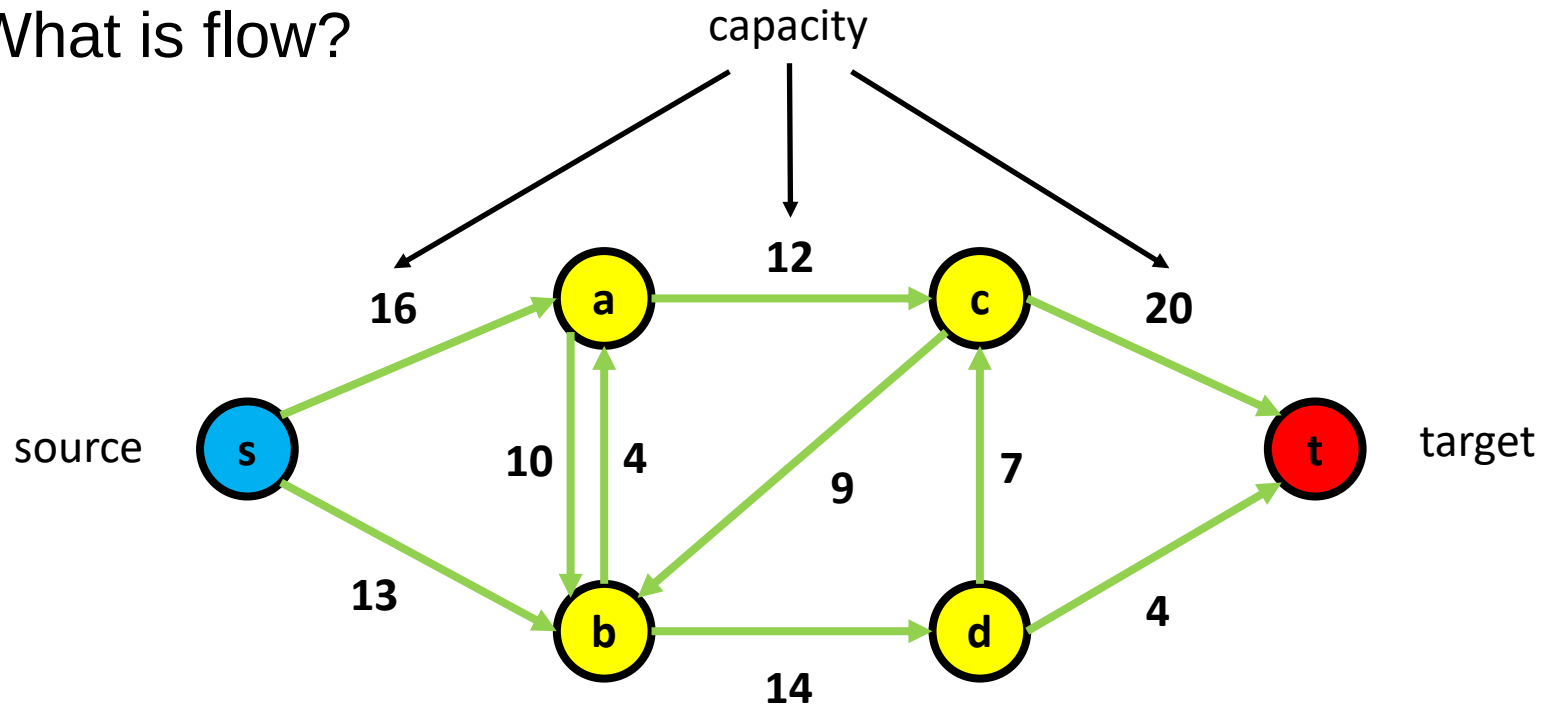


Flow Network

Transfer of content

MONASH University

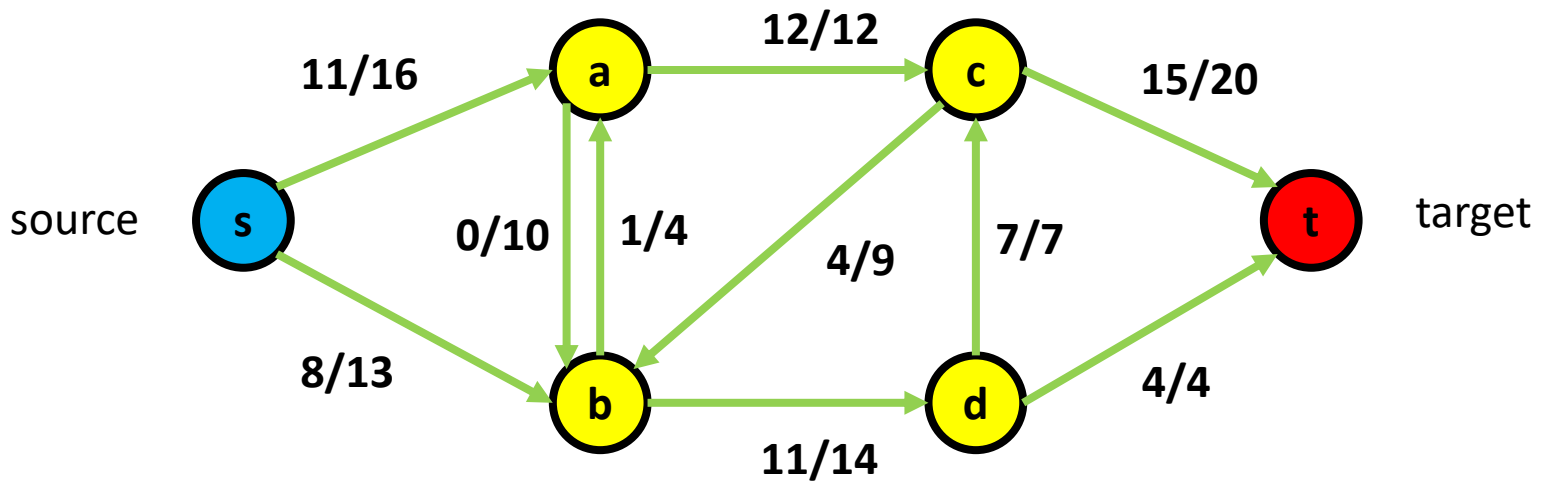
- What is flow?



Flow Network

Transfer of content

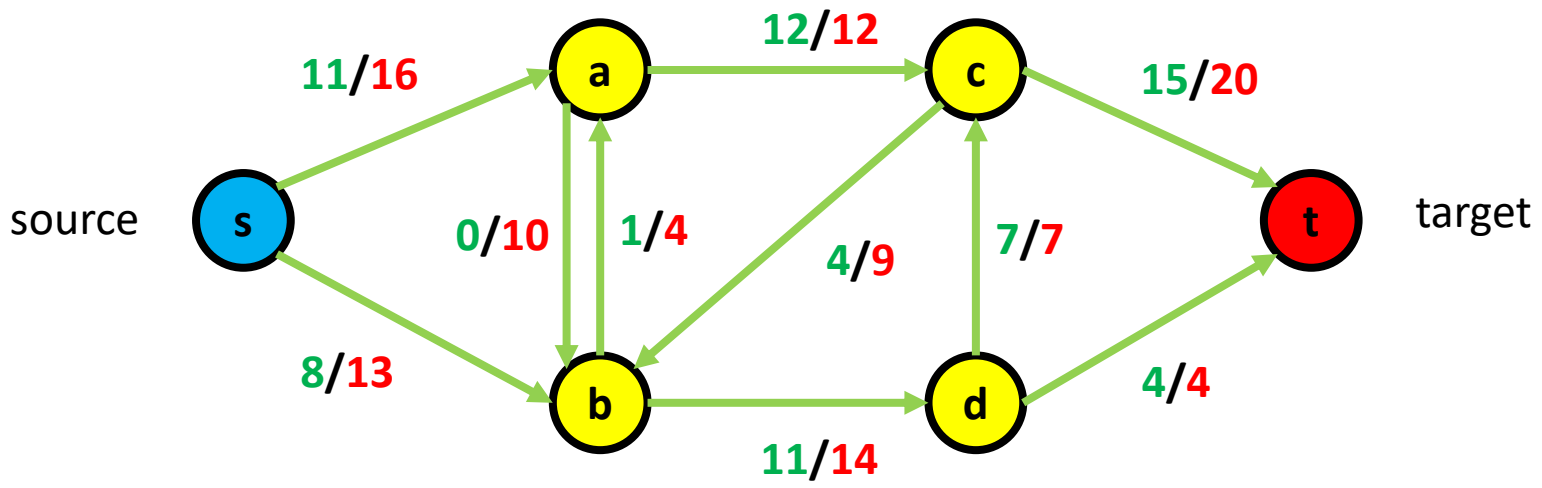
- What is flow?
- What is capacity?



Flow Network

Transfer of content

- What is flow?
- What is capacity?

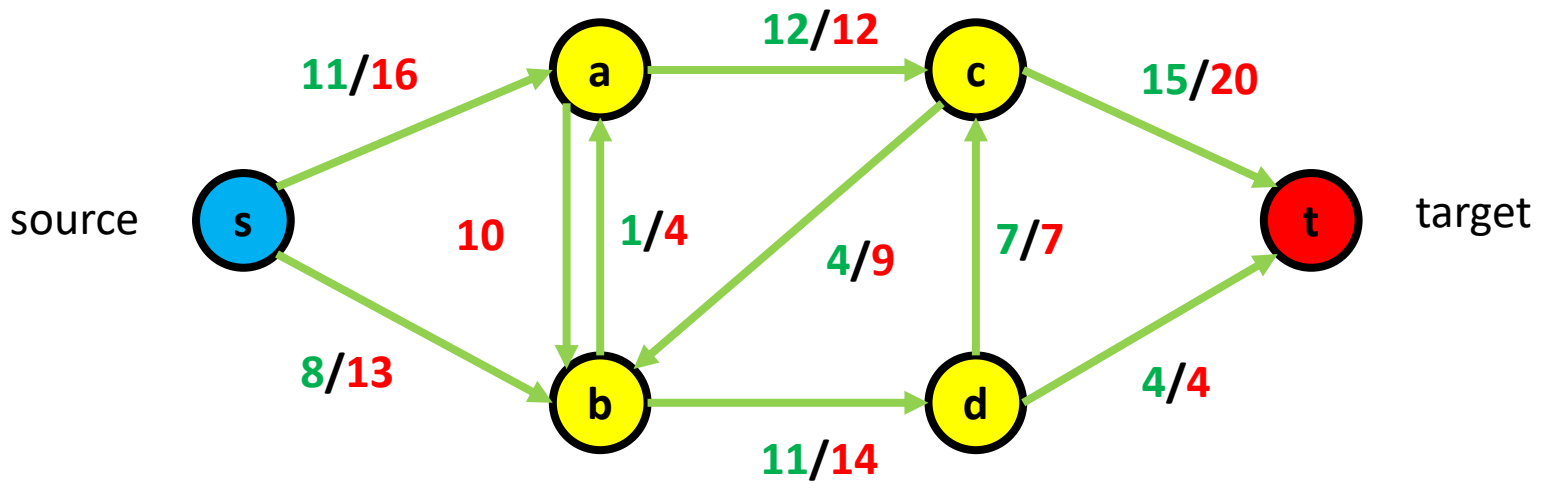


Flow Network

Transfer of content

MONASH University

- What is flow?
  - If there is no flow, you can exclude it
  - It is how much material flowing through each edge
- What is capacity?

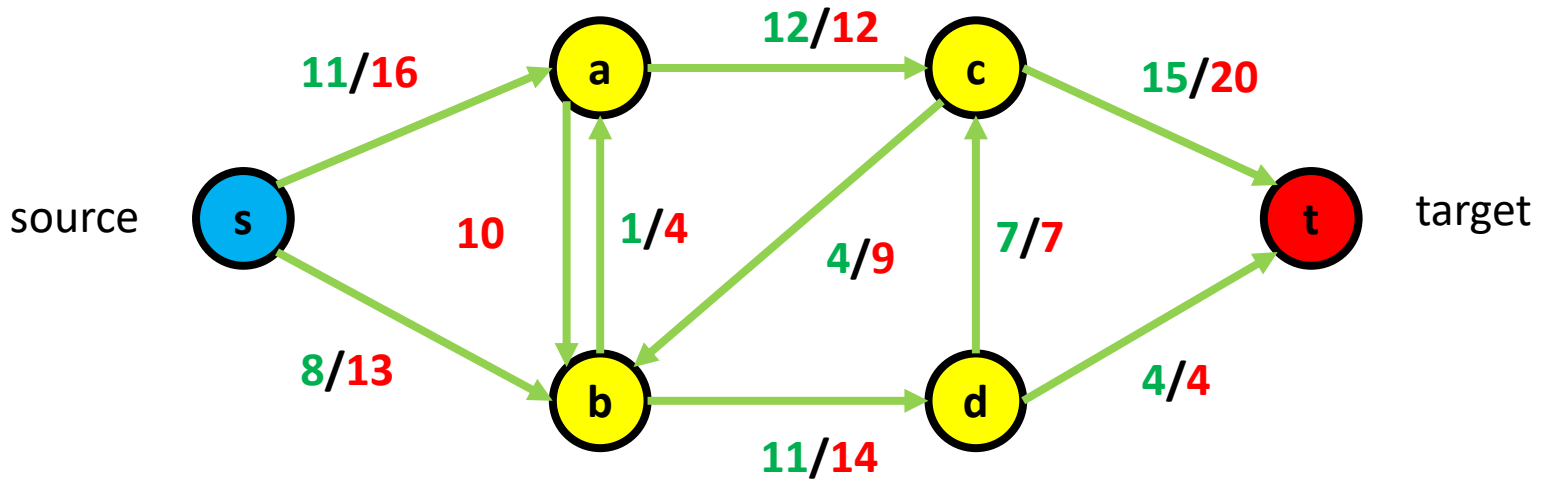


Flow Network

Transfer of content

MONASH University

- Flow constraint property

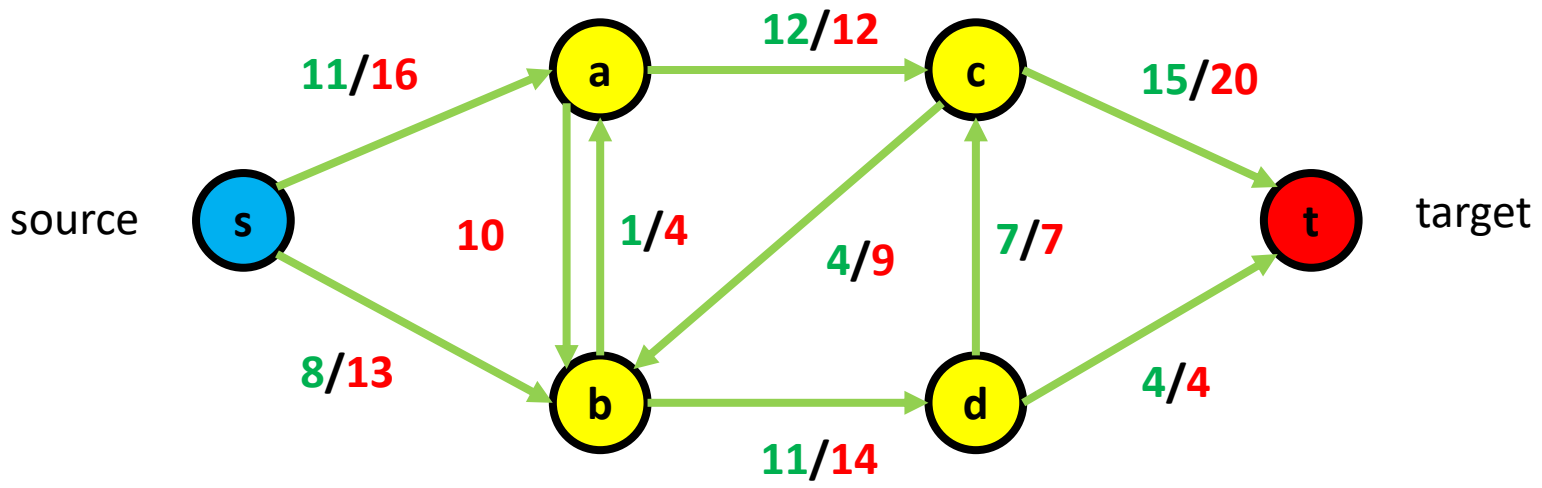


Flow Network

Transfer of content

MONASH University

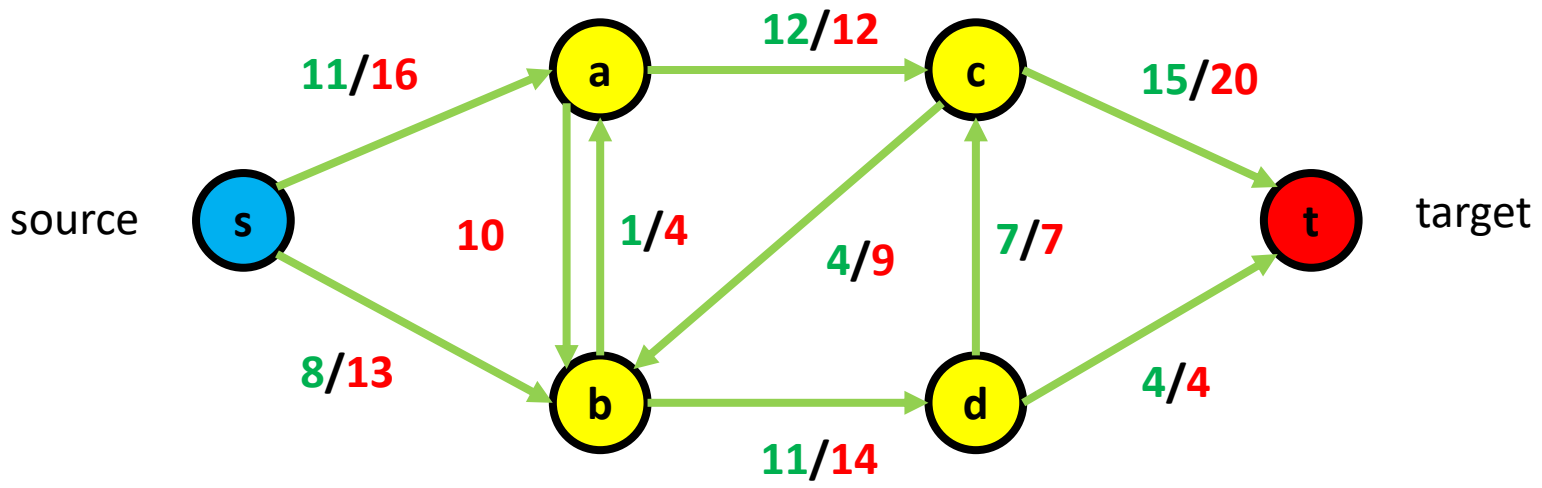
- Flow constraint property
  - For each edge, the flow can't be more than the capacity of the edge



Flow Network

Transfer of content

- Flow constraint property
  - For each edge, the flow can't be more than the capacity of the edge
  - In other words, you can't overload

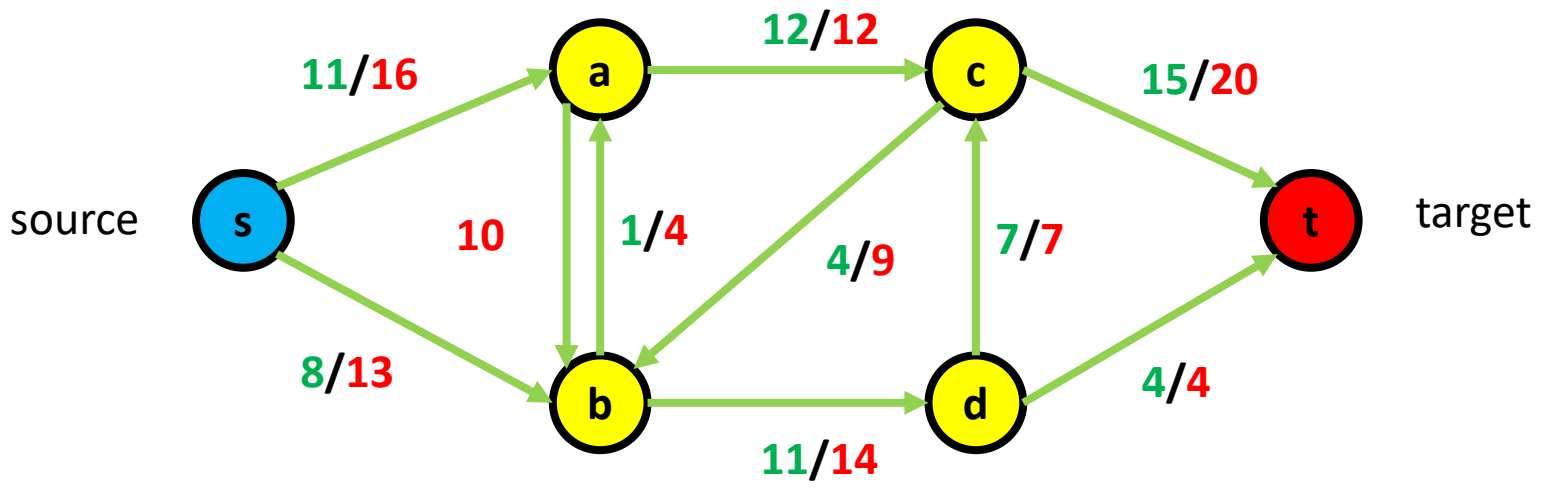


Flow Network

Transfer of content

MONASH University

- Flow conservation property

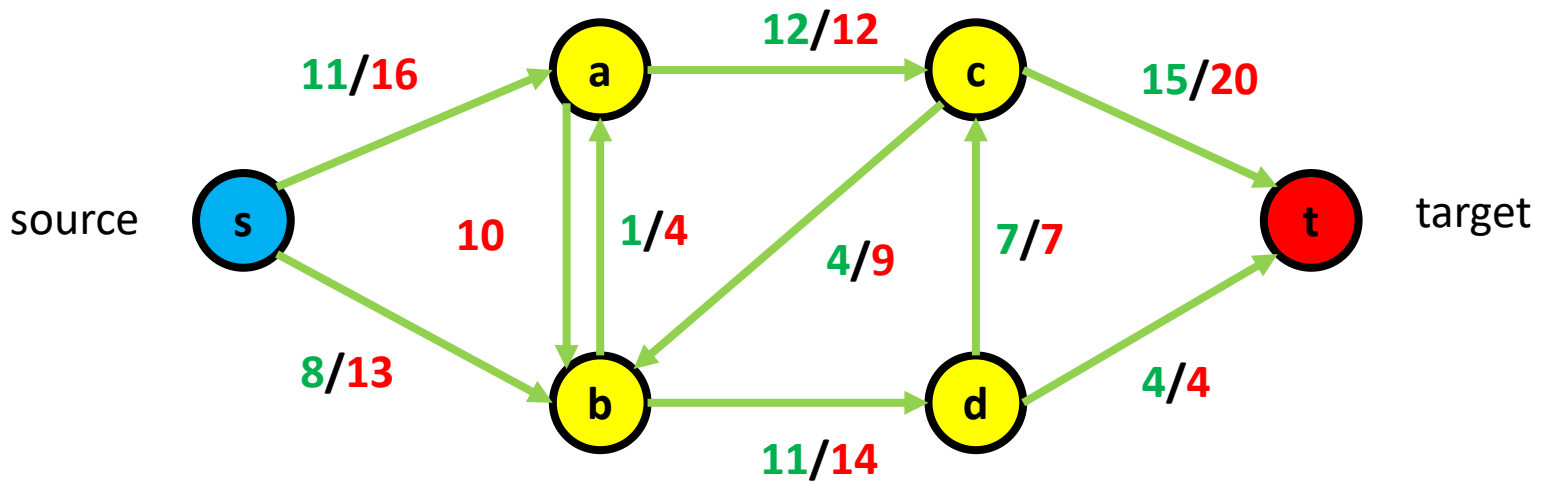


Flow Network

Transfer of content

MONASH University

- Flow conservation property
  - For every vertex in the graph (except source and target)
  - incoming flow == outgoing flow

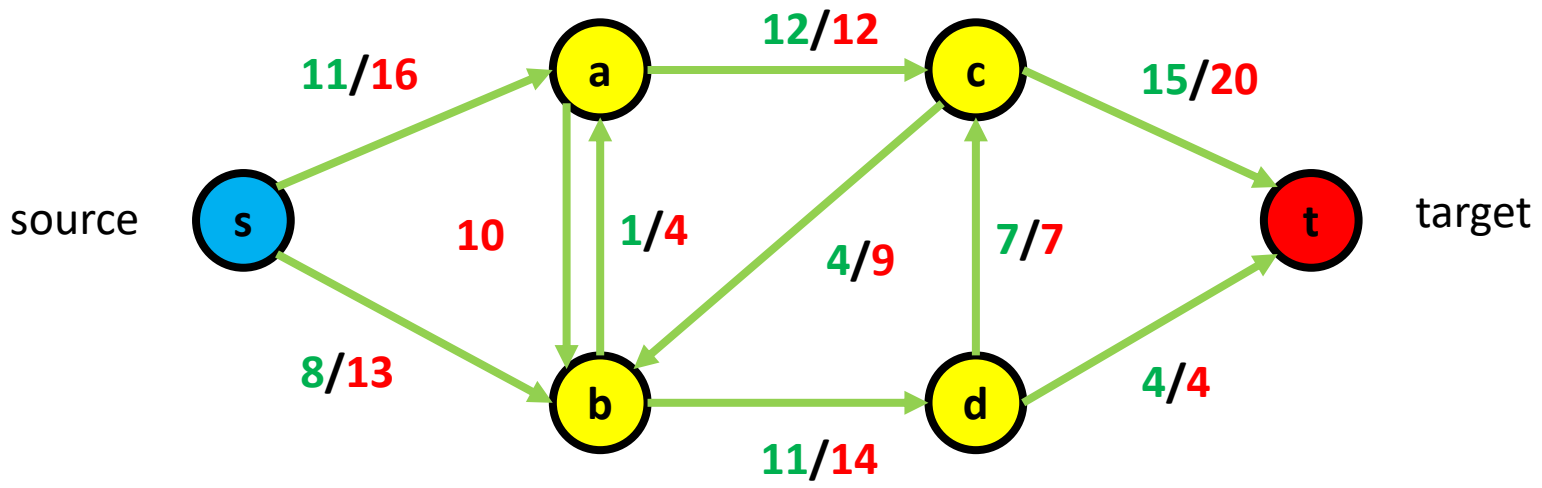


Flow Network

Transfer of content

MONASH University

- Flow conservation property
  - For every vertex in the graph (except source and target)
  - total incoming flow == total outgoing flow

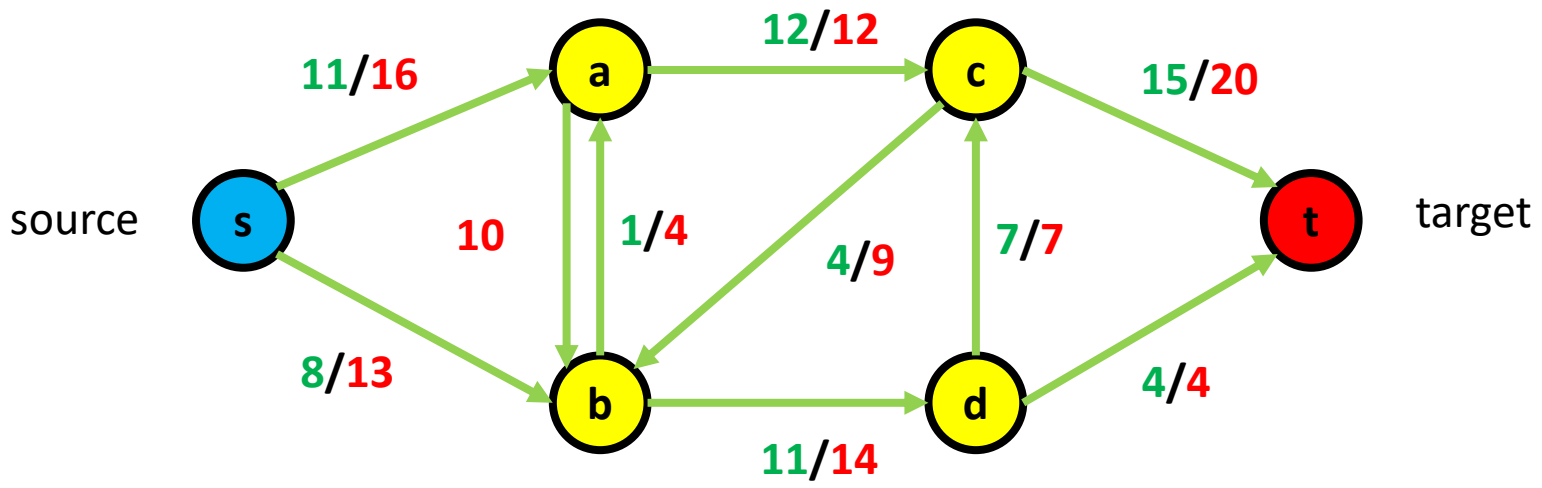


Flow Network

Transfer of content

MONASH University

- Flow conservation property
  - For every vertex in the graph (except source and target)
  - total incoming flow == total outgoing flow
  - What is the total incoming flow to vertex b?

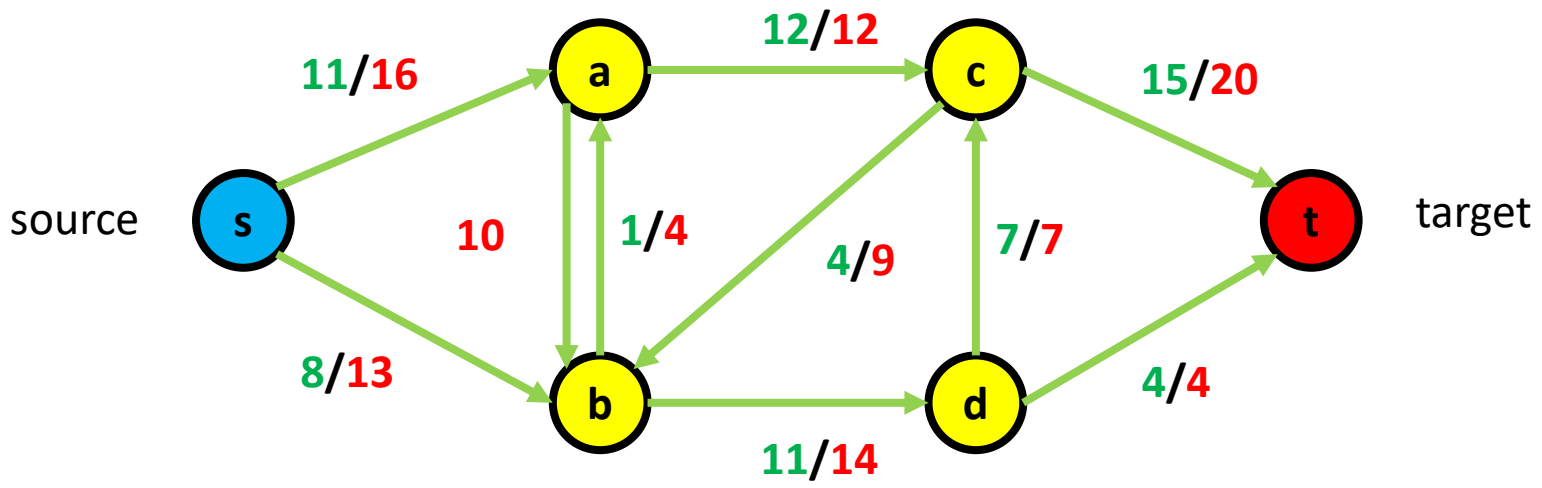


Flow Network

Transfer of content

MONASH University

- Flow conservation property
  - For every vertex in the graph (except source and target)
  - total incoming flow == total outgoing flow
  - What is the total incoming flow to vertex b? 12

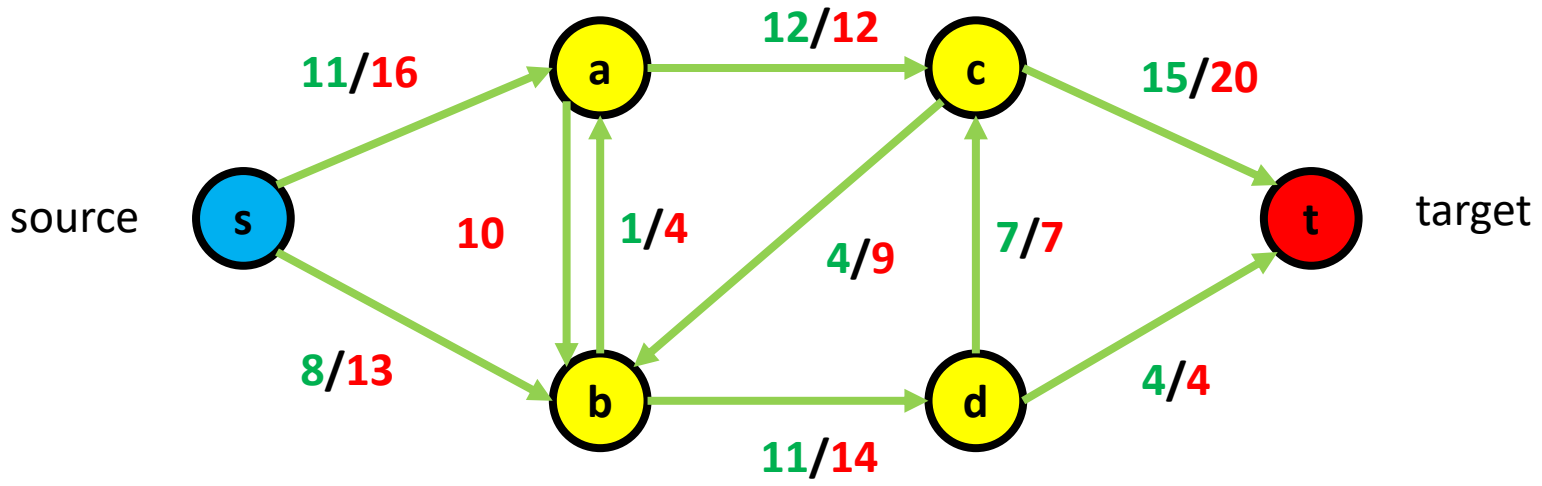


Flow Network

Transfer of content

MONASH University

- Flow conservation property
  - For every vertex in the graph (except source and target)
  - total incoming flow == total outgoing flow
  - What is the total incoming flow to vertex b? 12
  - What is the total outgoing flow to vertex b?

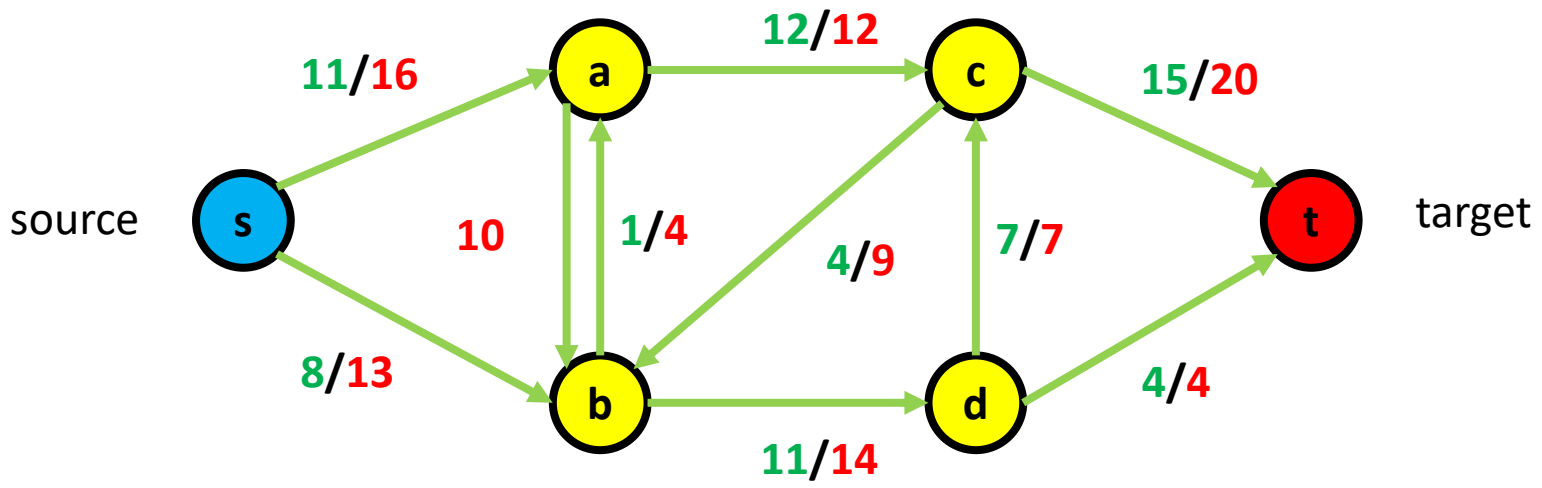


Flow Network

Transfer of content

MONASH University

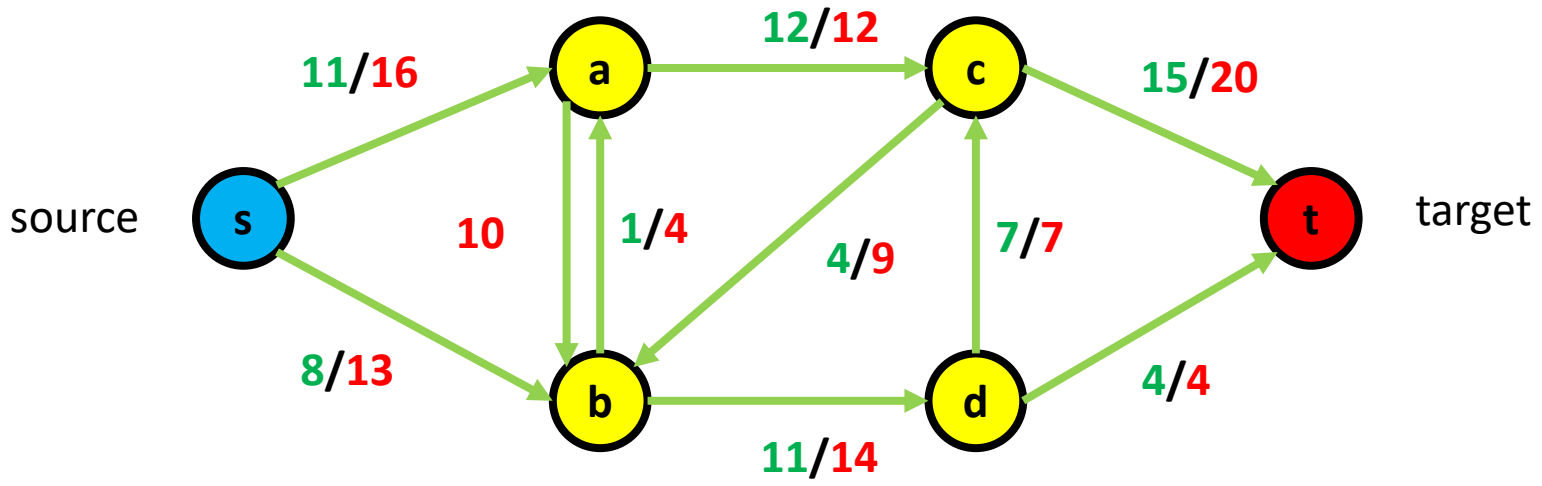
- Flow conservation property
  - For every vertex in the graph (except source and target)
  - total incoming flow == total outgoing flow
  - What is the total incoming flow to vertex b? 12
  - What is the total outgoing flow to vertex b? 12



Flow Network

Transfer of content

- Capacity constraint
- Flow conservation property



MONASH University

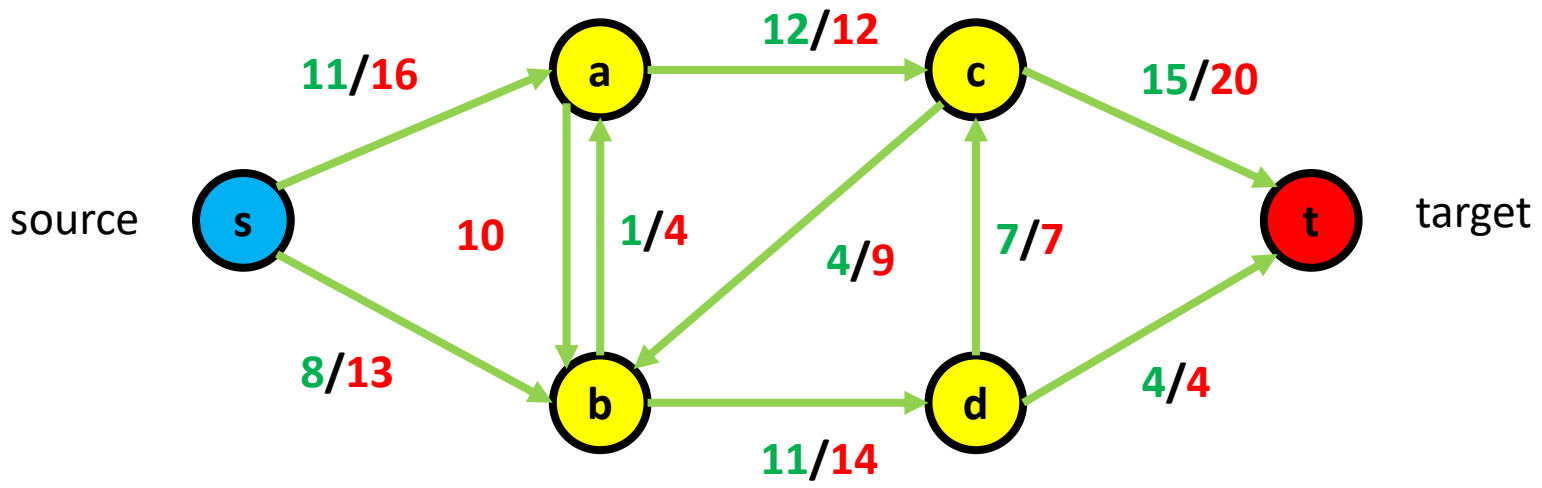
Questions?

Maximum-Flow Problem

Best network?

MONASH University

- What is the flow of the network?

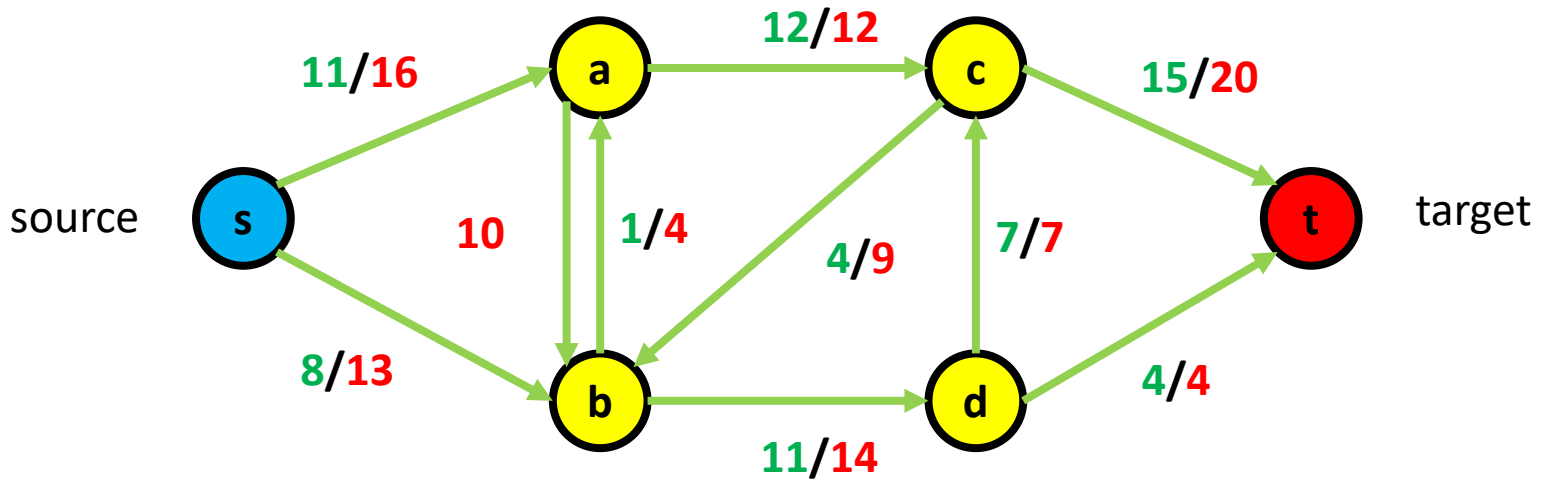


Maximum-Flow Problem

Best network?

MONASH University

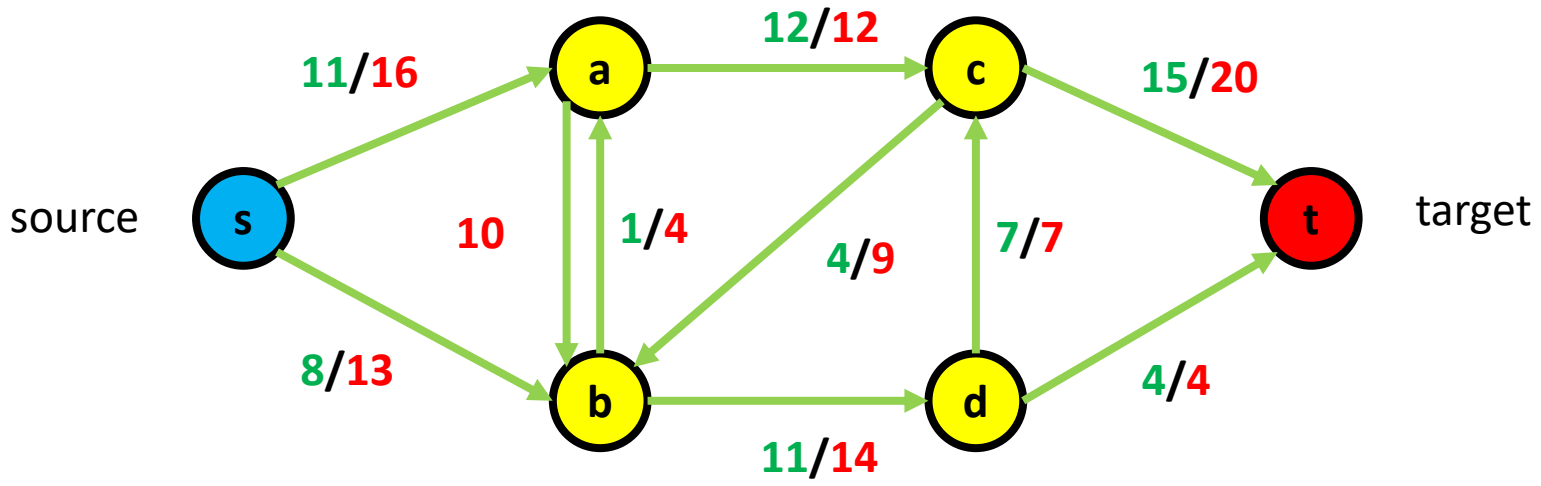
- What is the flow of the network? 19
  - Total flow out of source vertex
  - Total flow into target vertex



Maximum-Flow Problem

Best network?

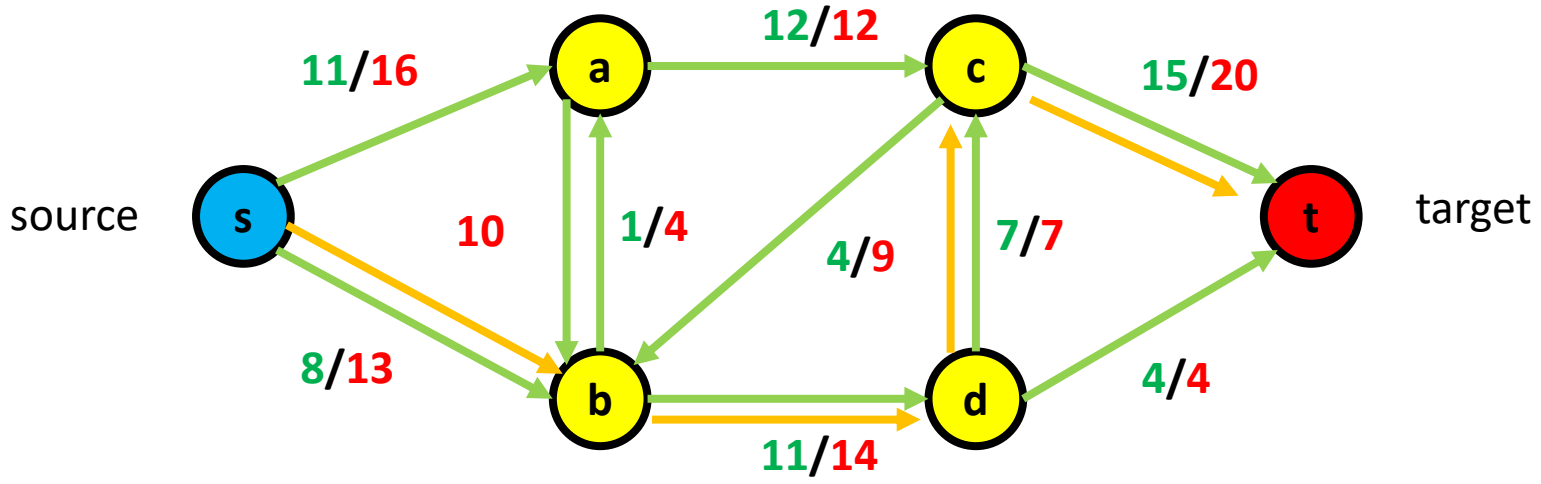
- What is the flow of the network? 19
  - Total flow out of source vertex
  - Total flow into target vertex
- Is this the maximum possible flow for this network?



Maximum-Flow Problem

Best network?

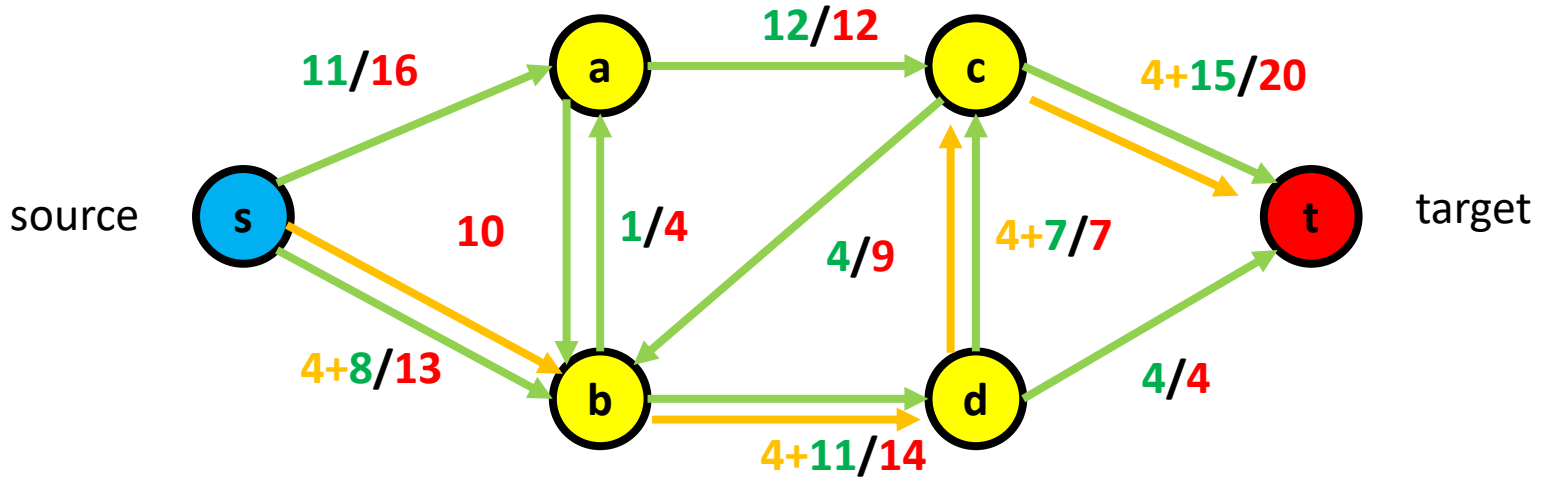
- Is this the maximum possible flow for this network?
  - We can push in 4 more through the following route…



Maximum-Flow Problem

Best network?

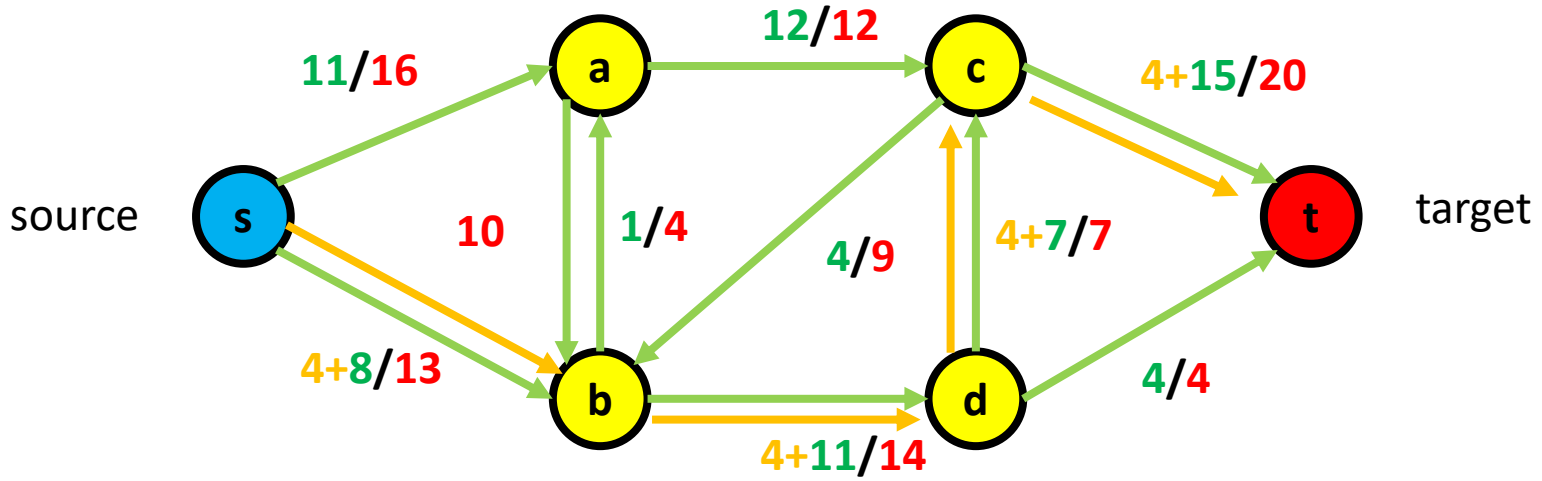
- Is this the maximum possible flow for this network?
  - We can push in 4 more through the following route…



Maximum-Flow Problem

Best network?

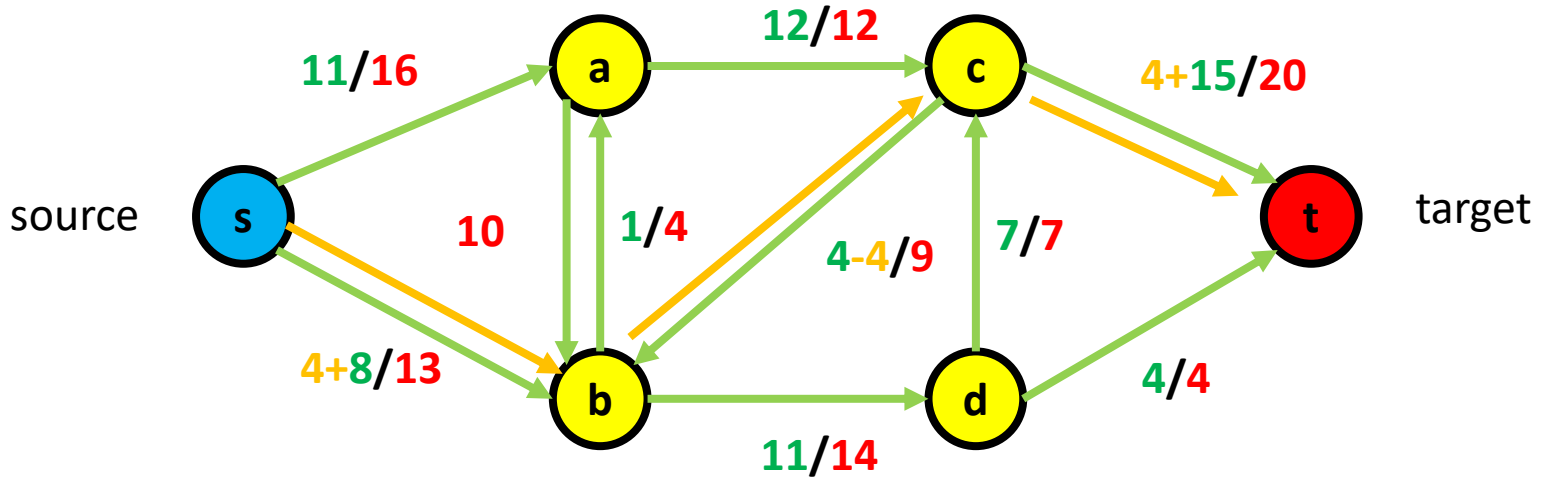
- Is this the maximum possible flow for this network?
  - We can push in 4 more through the following route… we cant! Cause over capacity…



Maximum-Flow Problem

Best network?

- Is this the maximum possible flow for this network?
  - We can push in 4 more through the following route… but we can do this, not accepting the opposite…

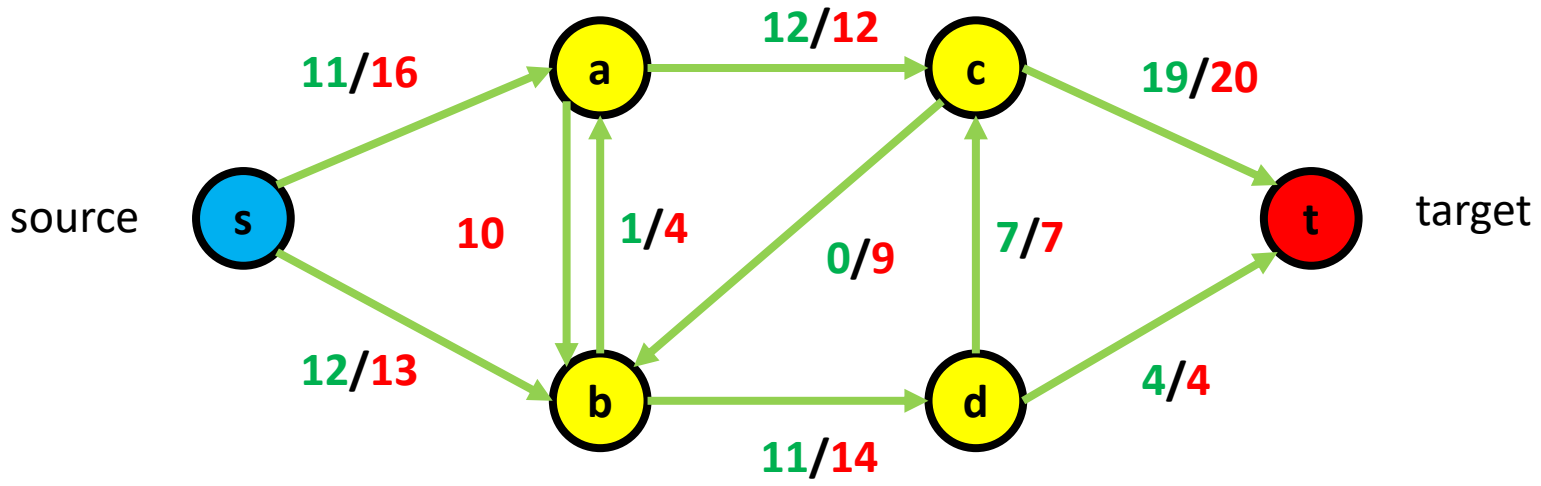


Maximum-Flow Problem

Best network?

MONASH University

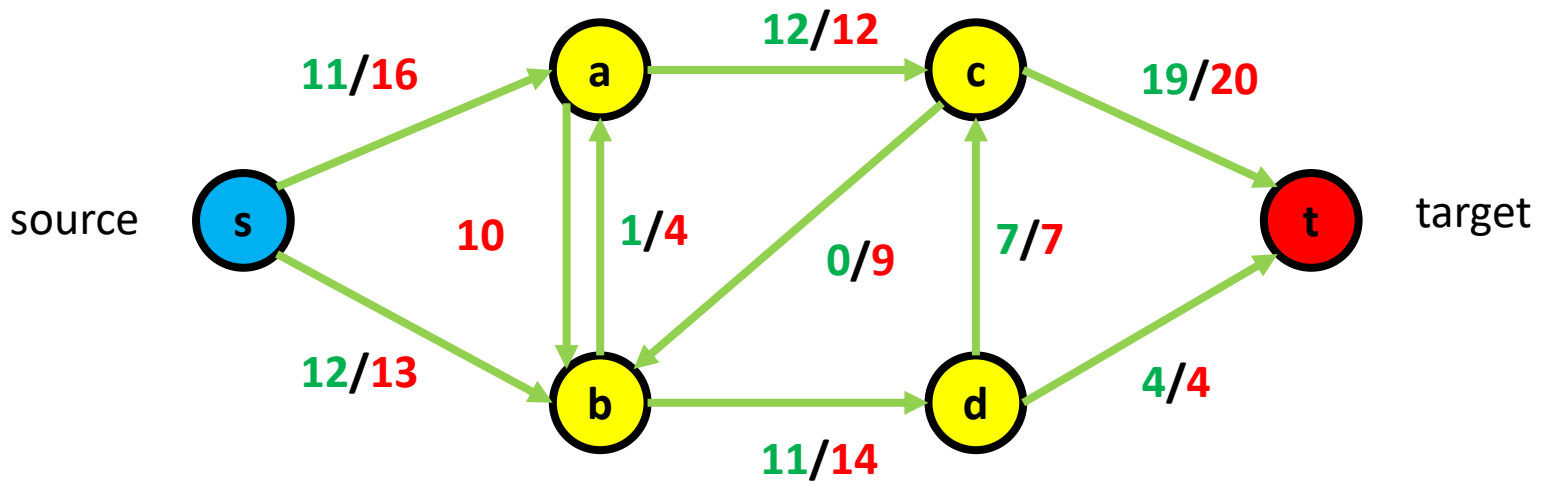
- Is this the maximum possible flow for this network? 23!
  - We can push in 4 more through the following route… but we can do this, not accepting the opposite…



Maximum-Flow Problem

Best network?

- Is this the maximum possible flow for this network? 23!
  - We can push in 4 more through the following route… but we can do this, not accepting the opposite…
- Is this easy to do?
  - No of course, but we in CS to make it easy!



MONASH University

Questions?

Ford-Fulkerson Method

Finding the maximum flow of network

- What we use to find the maximum flow

Ford-Fulkerson Method

Finding the maximum flow of network

- What we use to find the maximum flow

Ford-Fulkerson Method

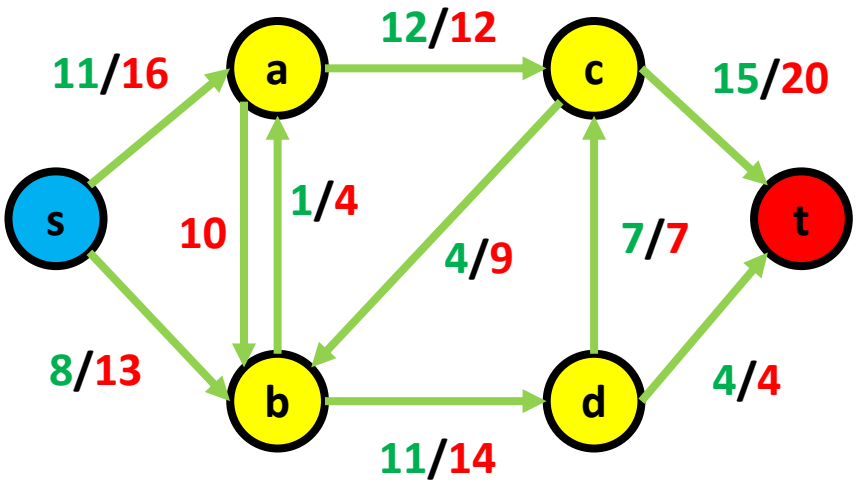
Finding the maximum flow of network

- Residue network first…

Residual network

Another freaking network

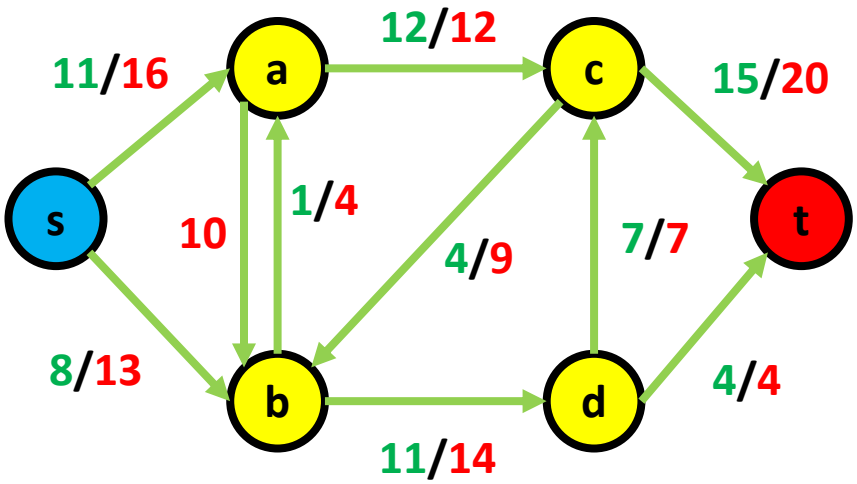
- Consider the following graph (same as earlier)



Residual network

Another freaking network

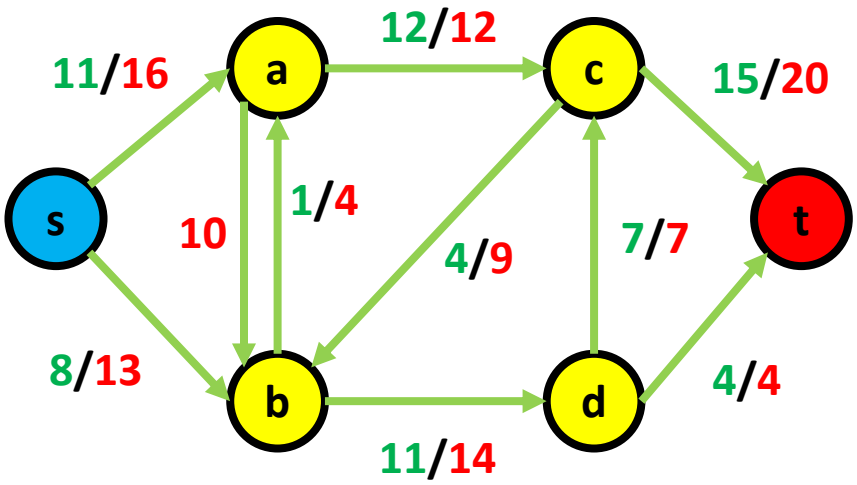
- Consider the following graph (same as earlier)
  - Can you make a residual network?



Residual network

Another freaking network

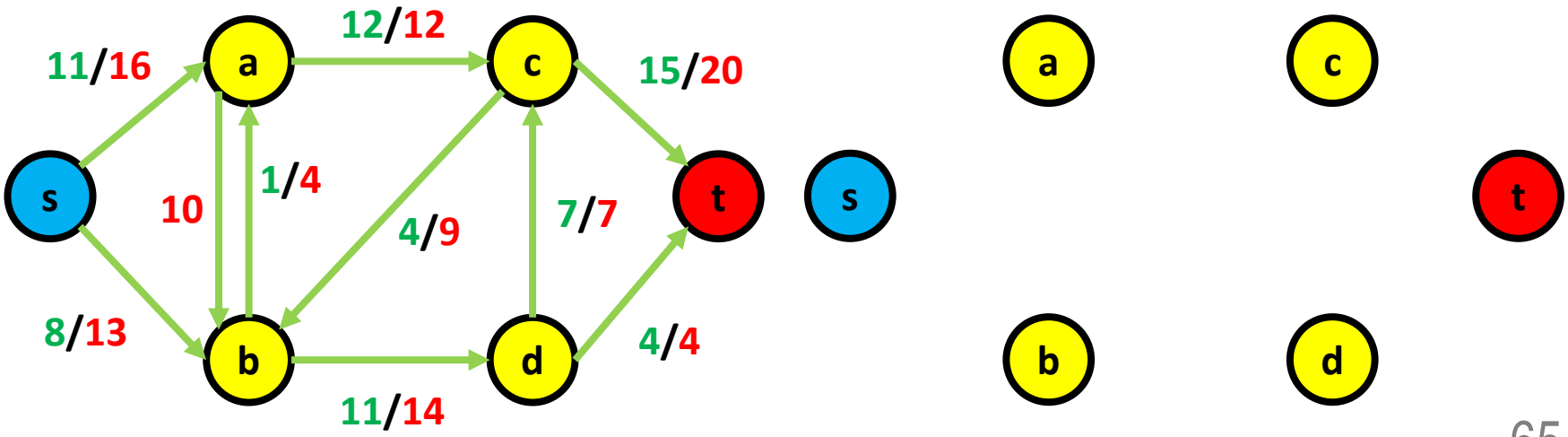
- What is a residual network?



Residual network

Another freaking network

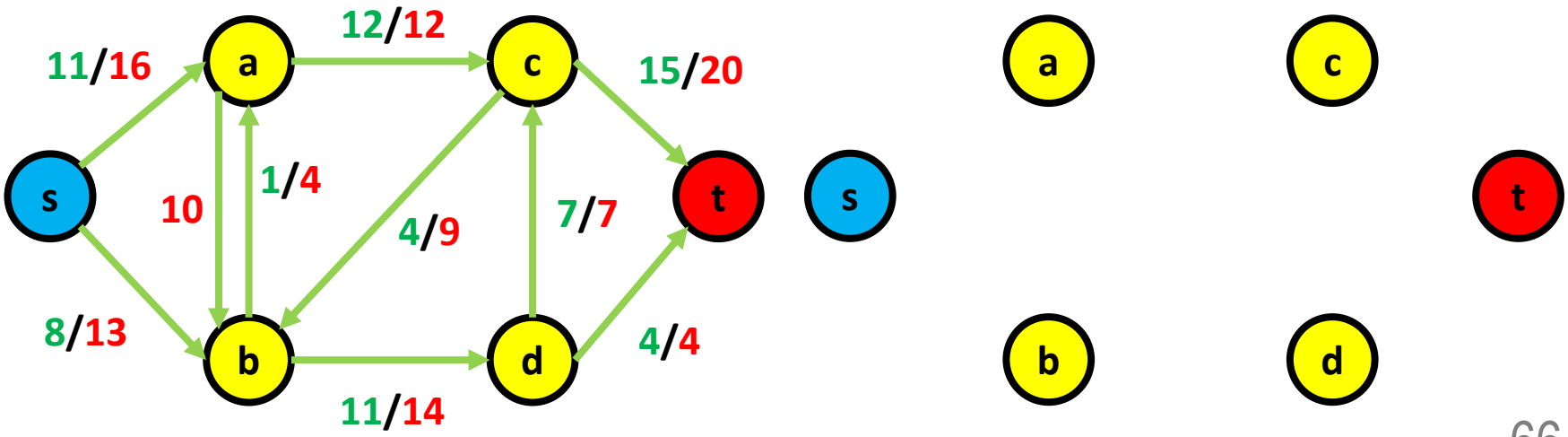
- What is a residual network?
  - Same vertices



Residual network

Another freaking network

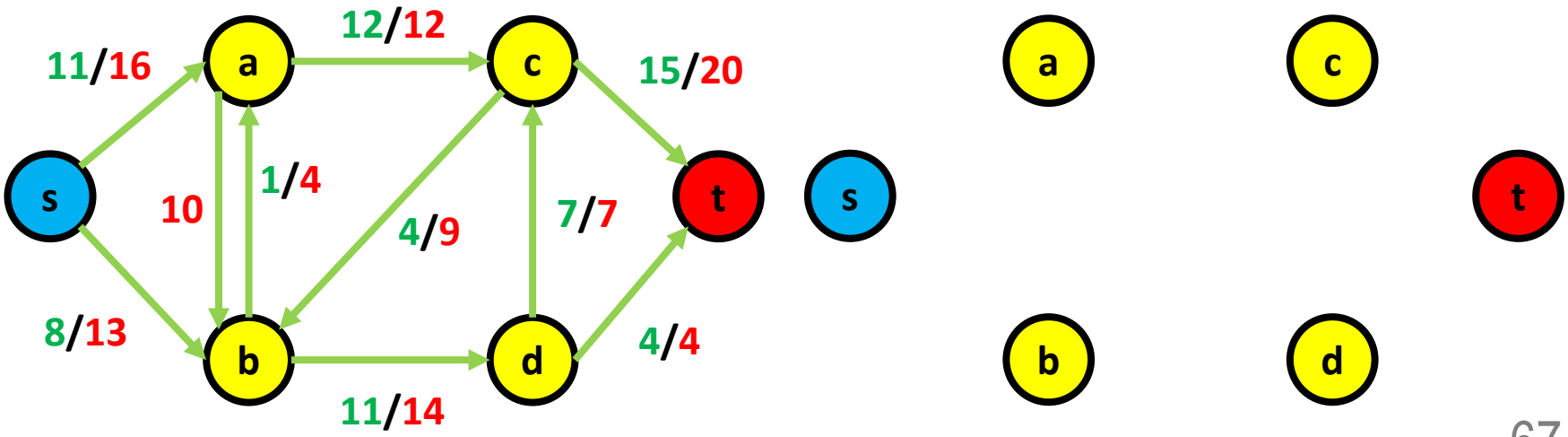
- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge



Residual network

Another freaking network

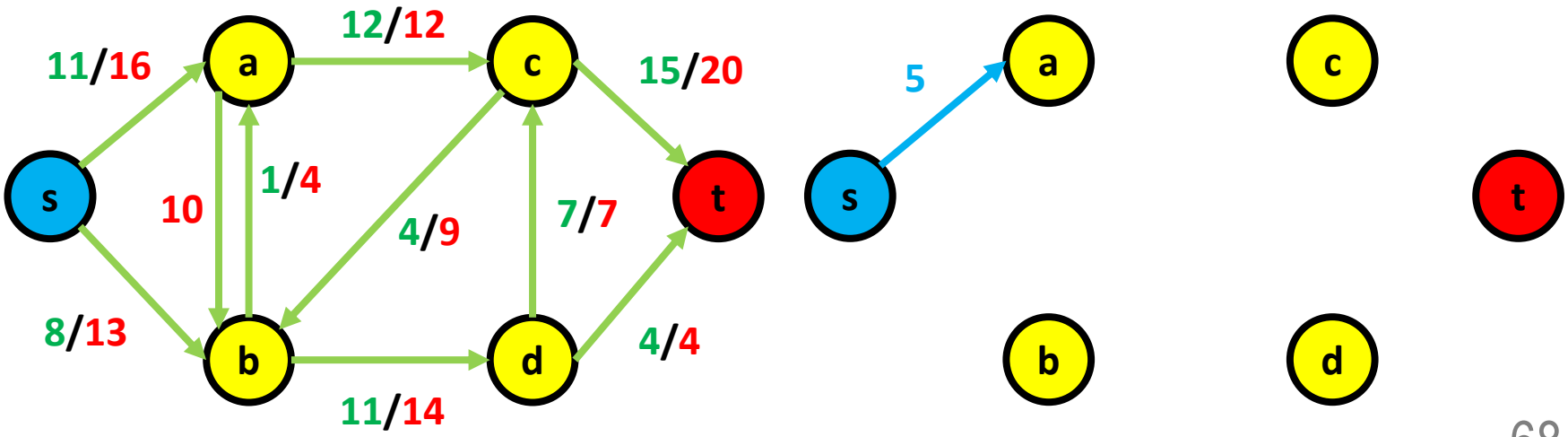
- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity



Residual network

Another freaking network

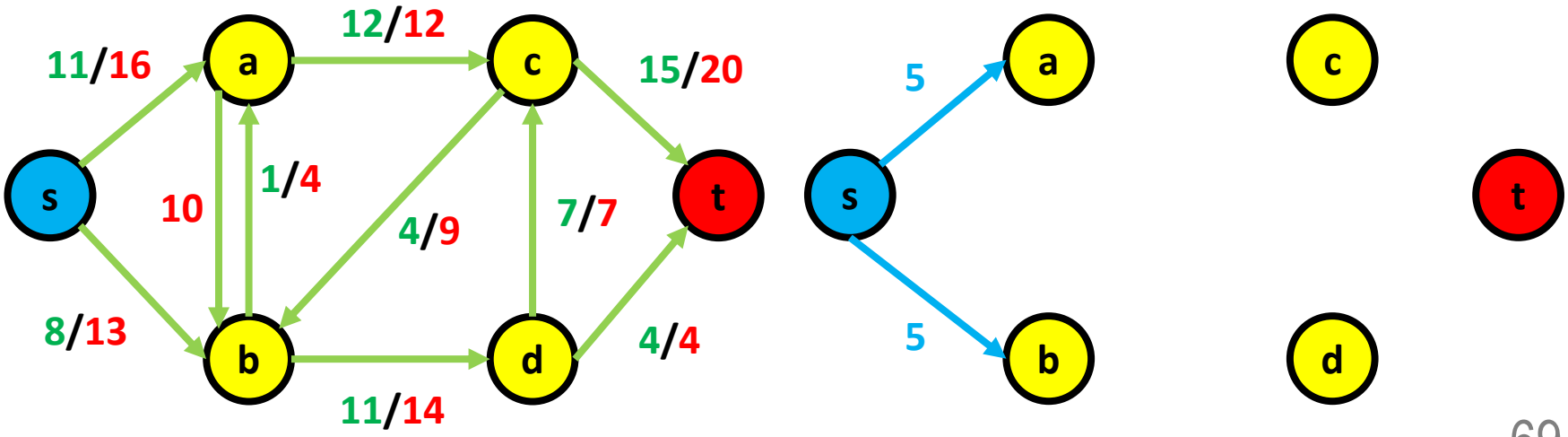
- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity



Residual network

Another freaking network

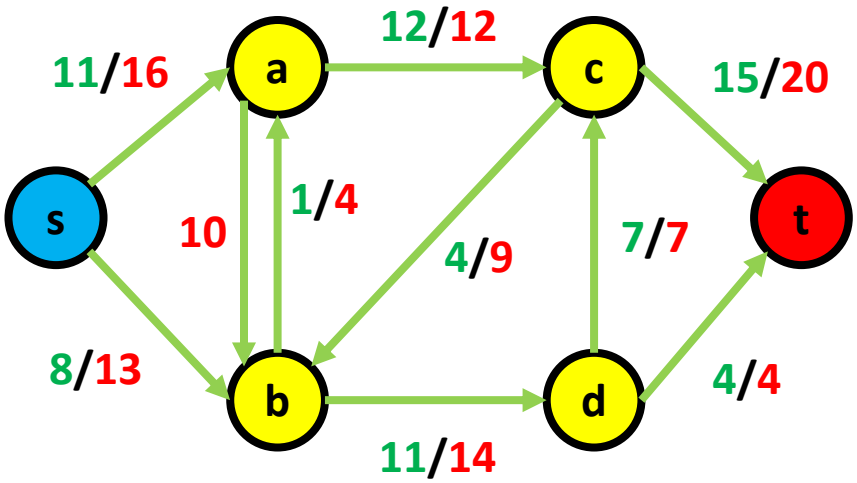
- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity

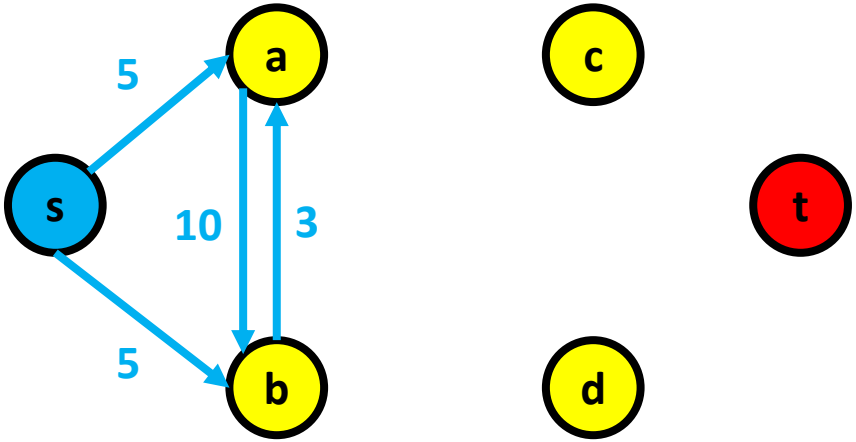


Residual network

Another freaking network

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
    - What about the one between a and b?

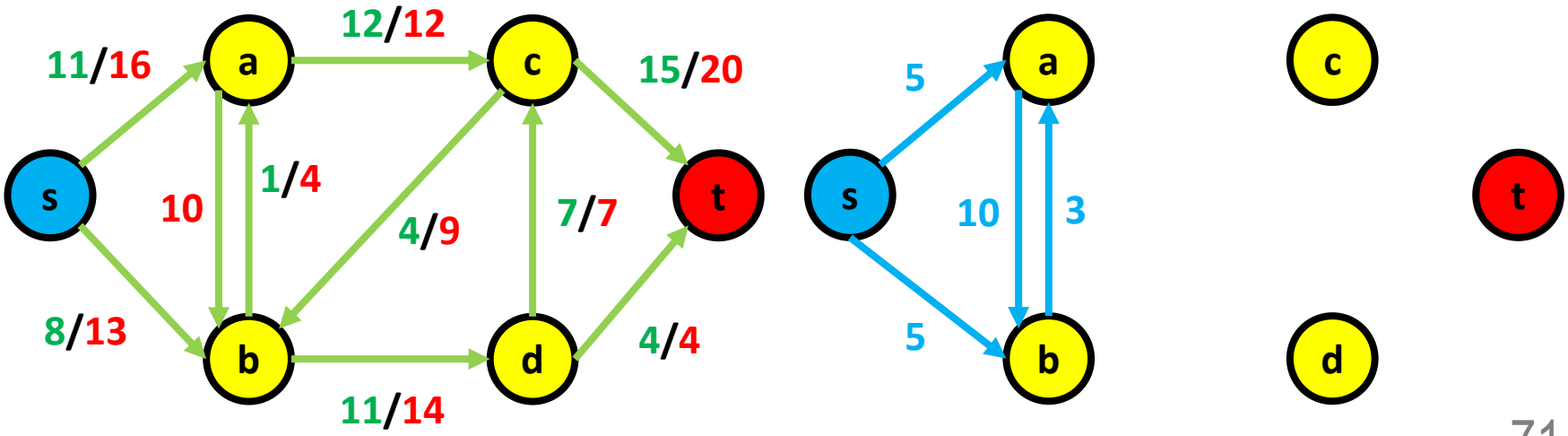




Residual network

Another freaking network

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
    - What about the one between a and b? We will come back to this later…

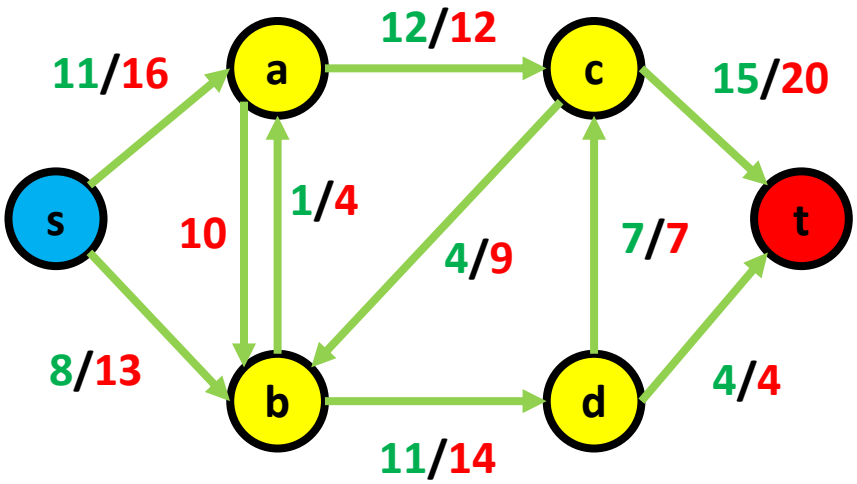


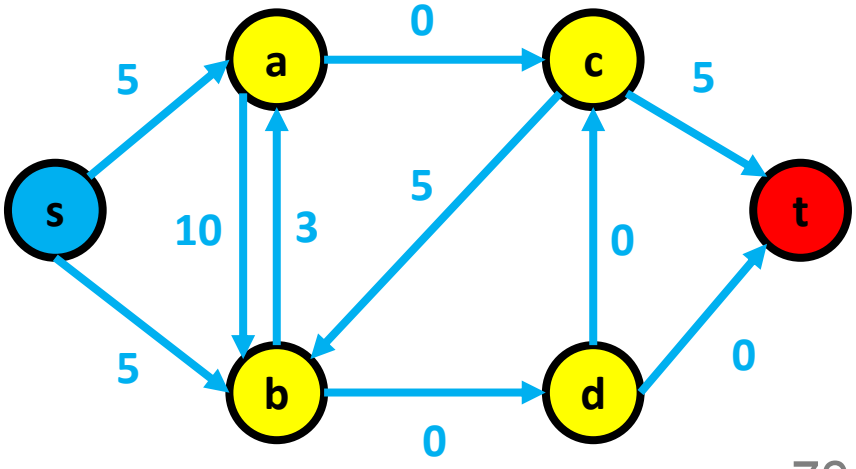
Residual network

Another freaking network

MONASH University

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity



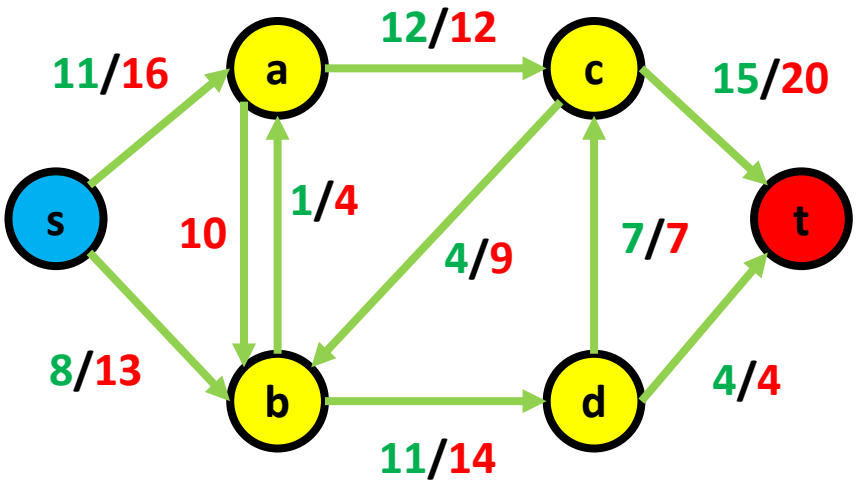


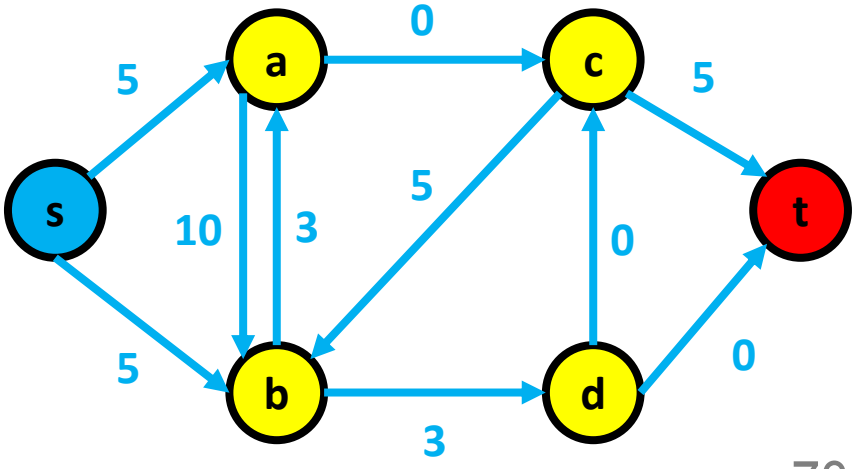
Residual network

Another freaking network

MONASH University

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
    - We can delete the ones with 0

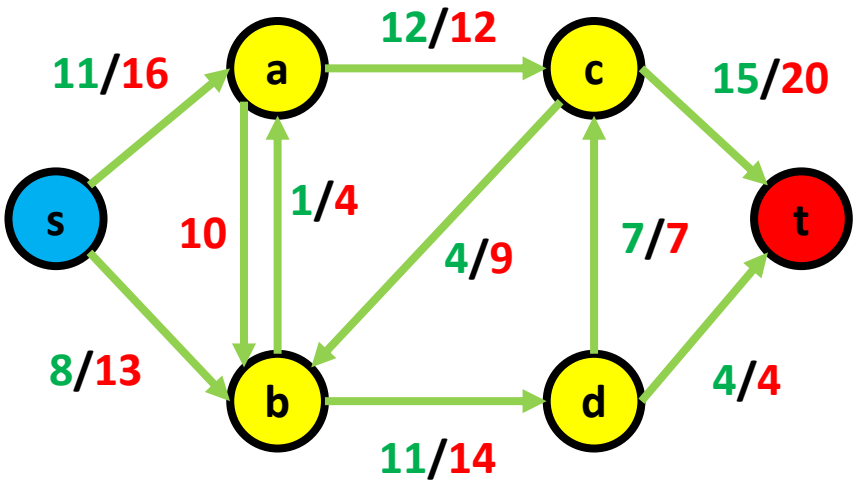


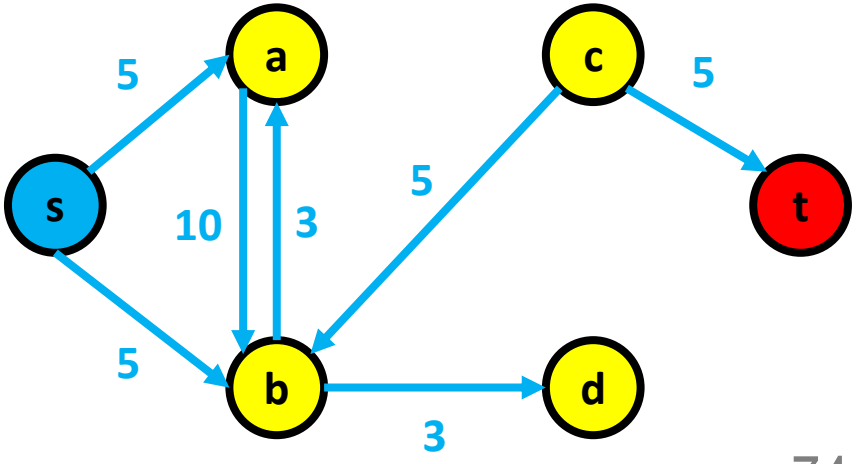


Residual network

Another freaking network

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
    - We can delete the ones with 0

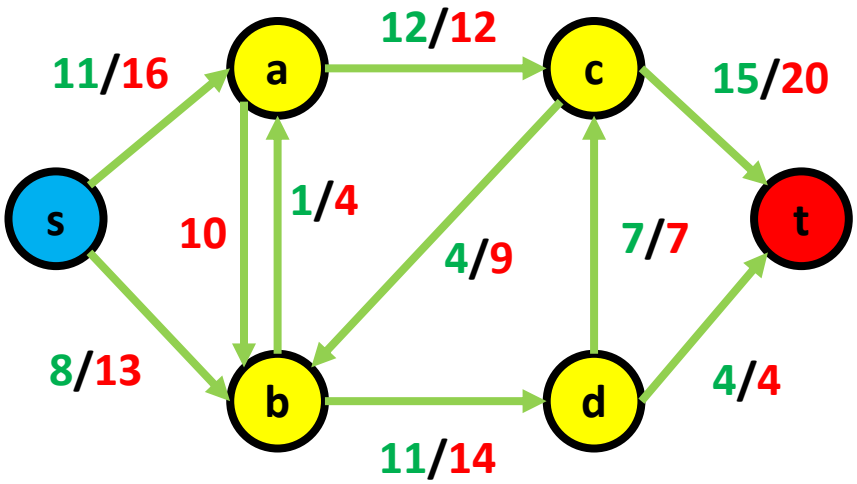


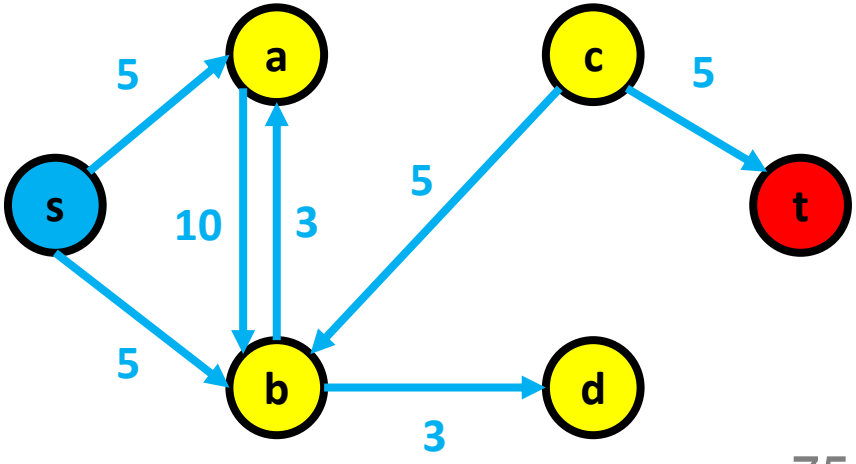


Residual network

Another freaking network

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
  - Backward edge/ reversible edge

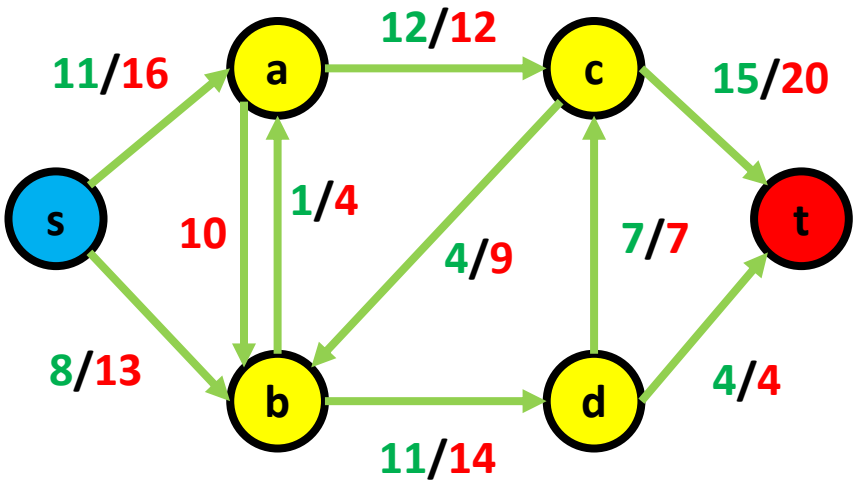


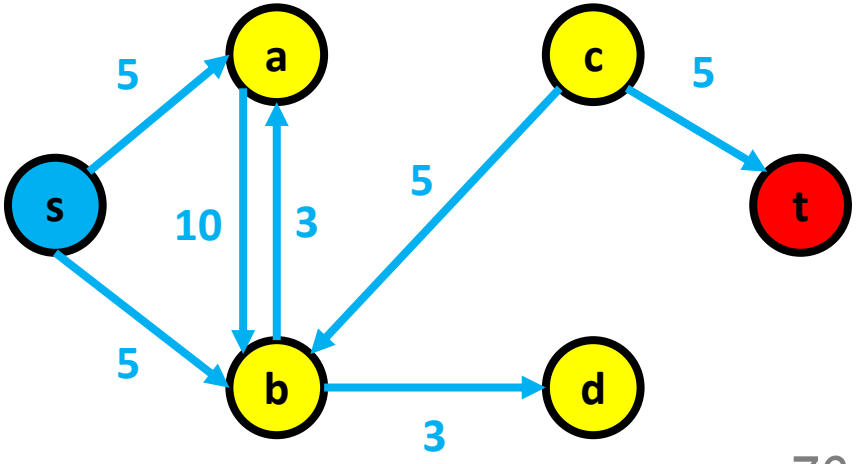


Residual network

Another freaking network

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
  - Backward edge/ reversible edge for flow that can be cancelled
    - Provided they have been allocated

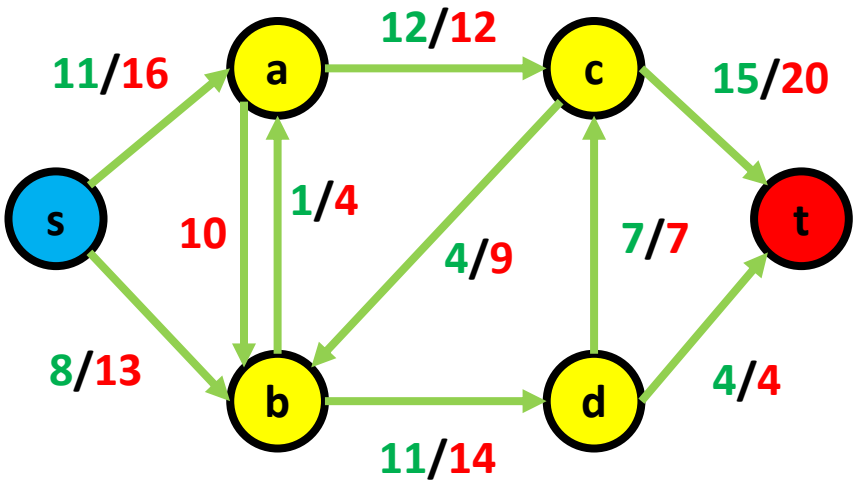


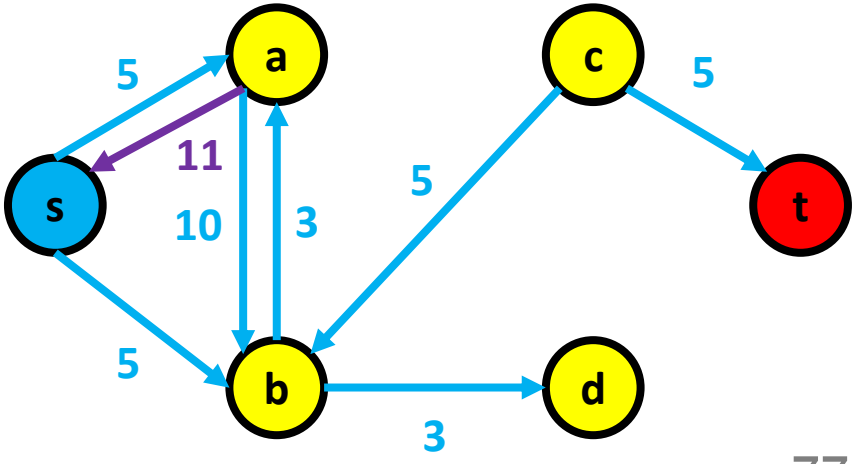


Residual network

Another freaking network

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
  - Backward edge/ reversible edge for flow that can be cancelled
    - Provided they have been allocated

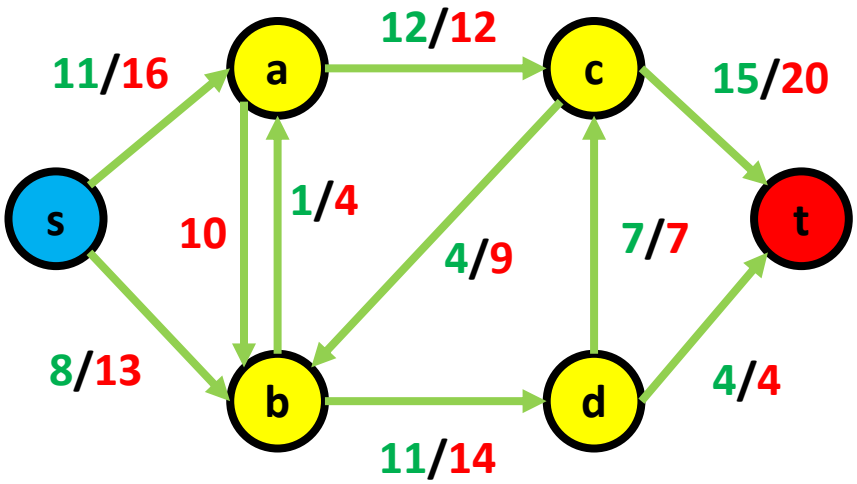


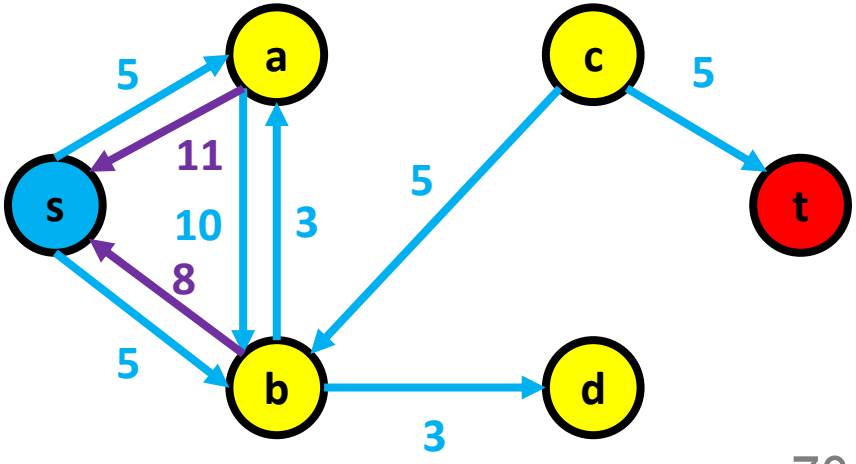


Residual network

Another freaking network

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
  - Backward edge/ reversible edge for flow that can be cancelled
    - Provided they have been allocated

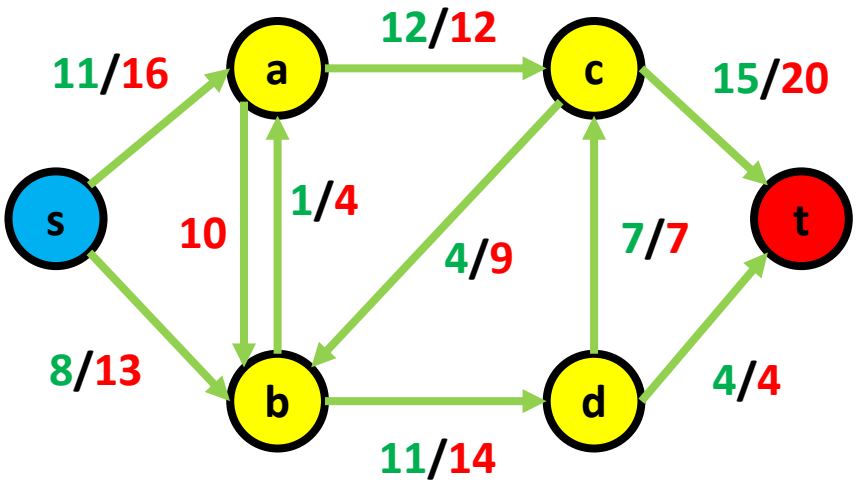


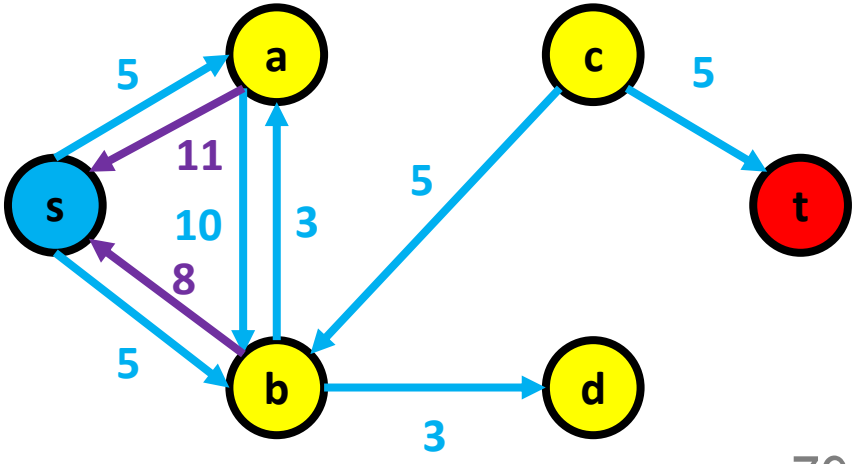


Residual network

Another freaking network

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
  - Backward edge/ reversible edge for flow that can be cancelled
    - What about the one between a and b?

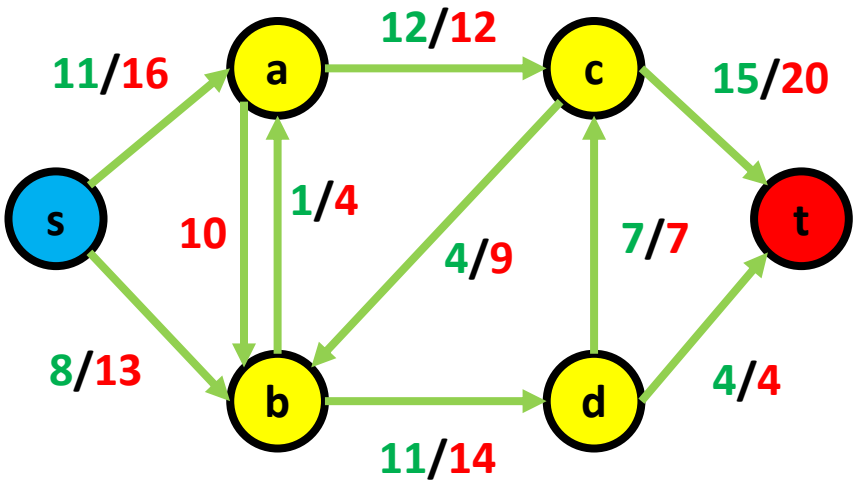


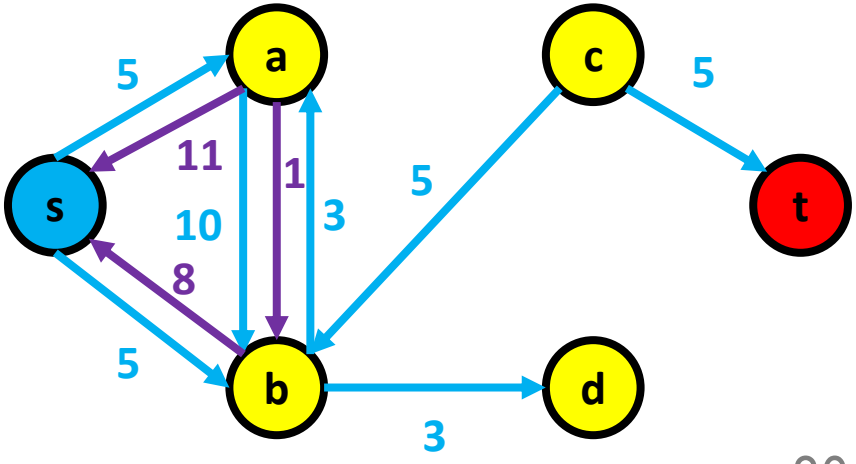


Residual network

Another freaking network

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
  - Backward edge/ reversible edge for flow that can be cancelled
    - What about the one between a and b?

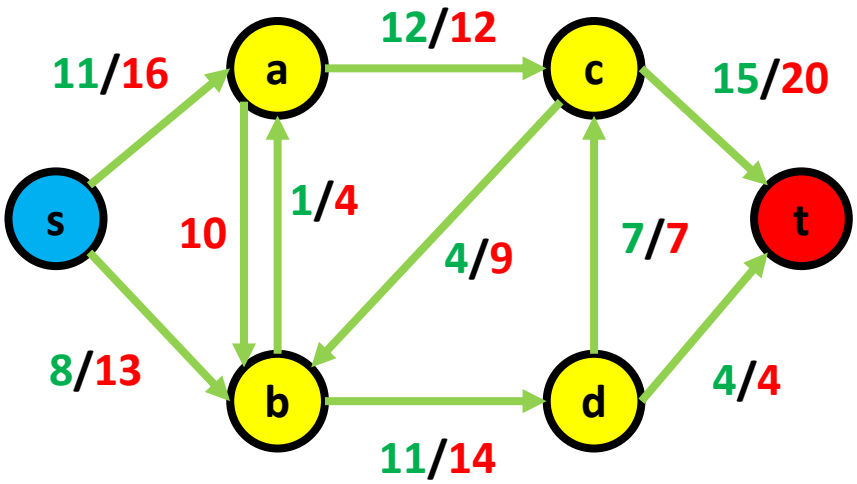


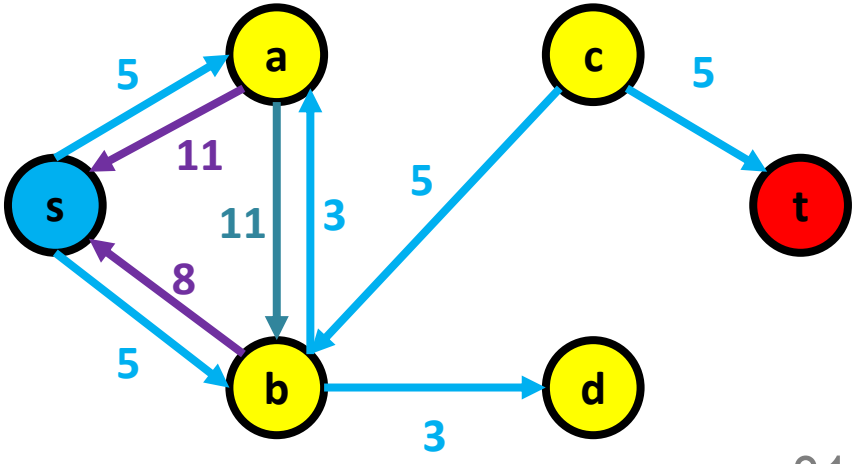


Residual network

Another freaking network

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
  - Backward edge/ reversible edge for flow that can be cancelled
    - What about the one between a and b? We have 2 in the same direction, so we combine both

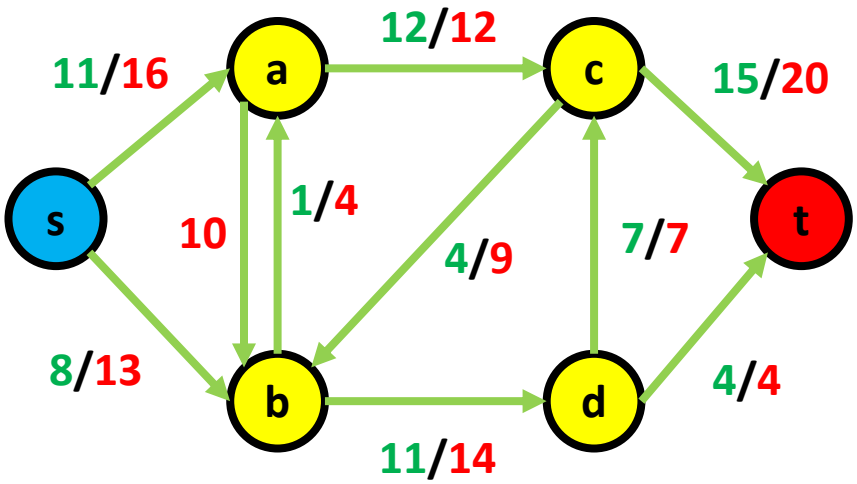


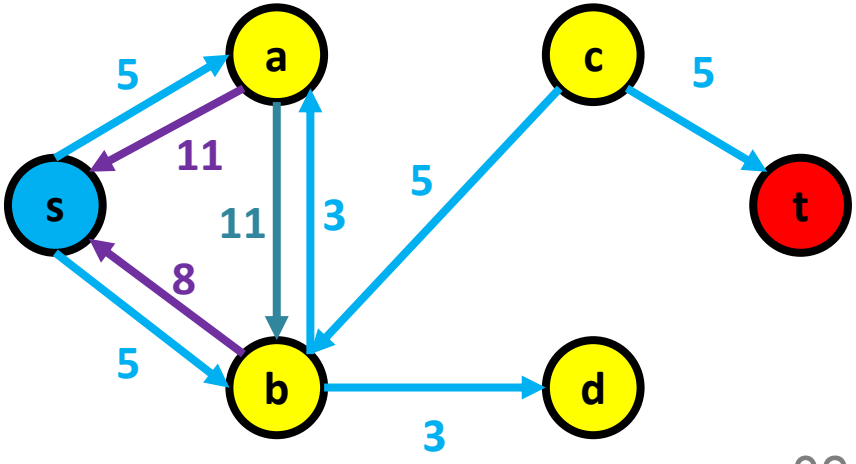


Residual network

Another freaking network

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
  - Backward edge/ reversible edge for flow that can be cancelled
    - What about the one between a and b? We have 2 in the same direction, so we combine both. The other side is 0, so nothing to combine



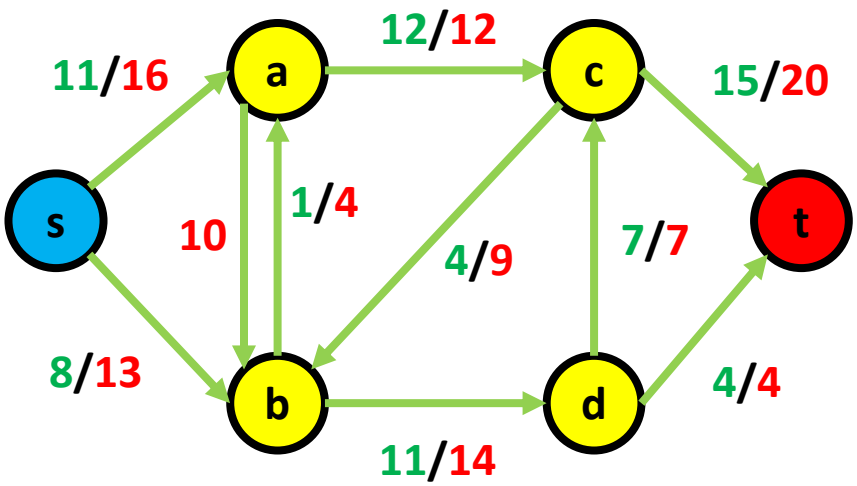


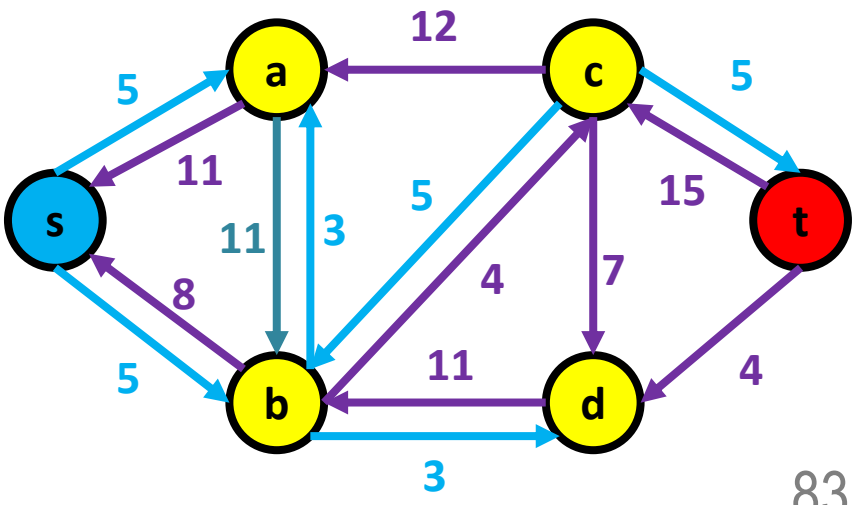
Residual network

Another freaking network

MONASH University

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
  - Backward edge/ reversible edge for flow that can be cancelled
    - And we add for all

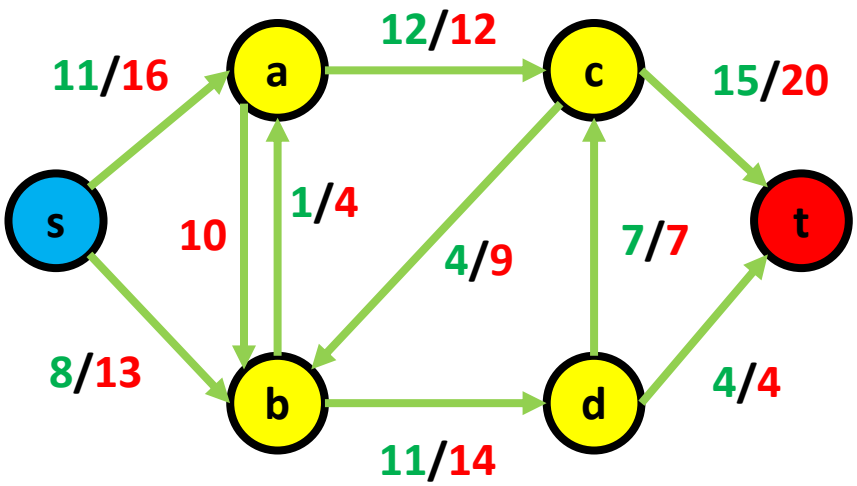


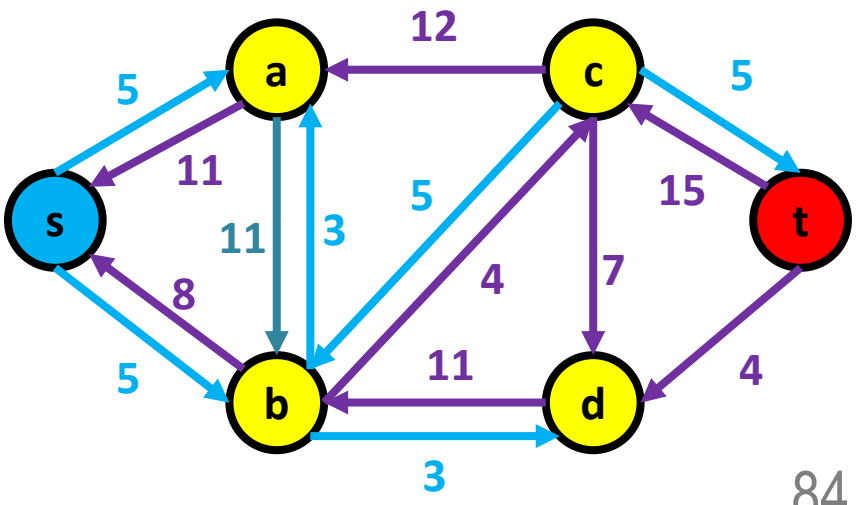


Residual network

Another freaking network

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
  - Backward edge/ reversible edge for flow that can be cancelled

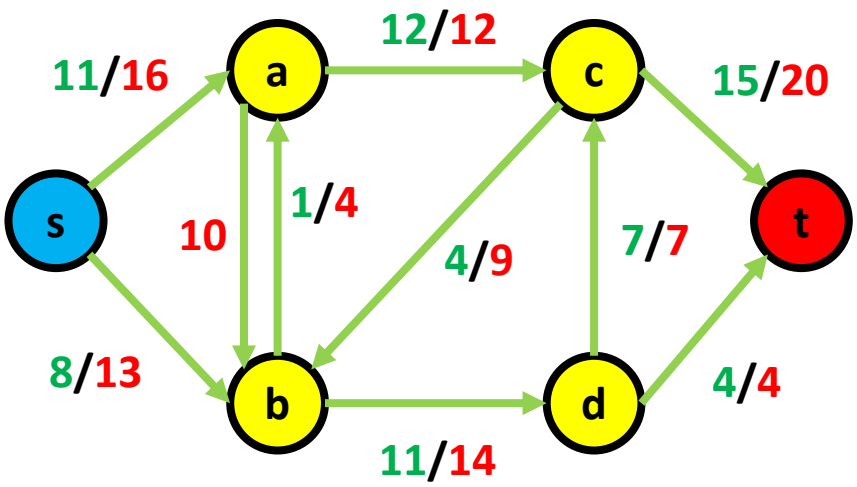


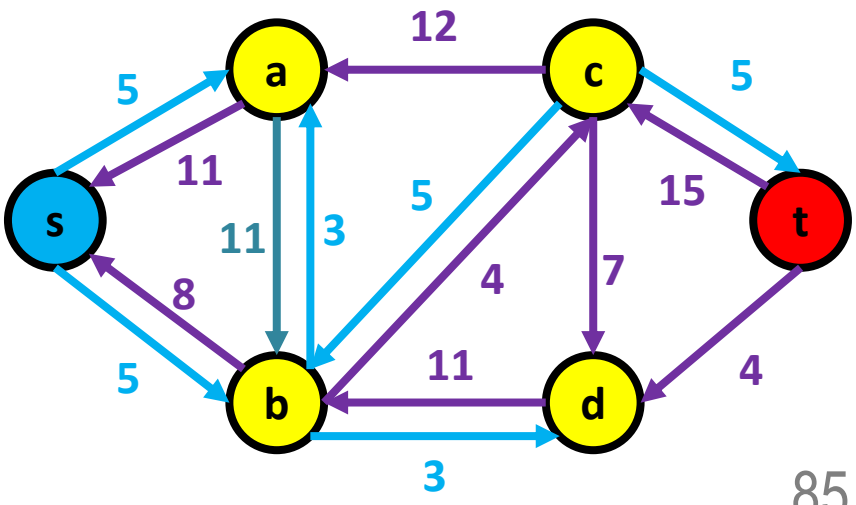


Residual network

Another freaking network

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
  - Backward edge/ reversible edge for flow that can be cancelled
  - Simple graph, so multi edges are merged together

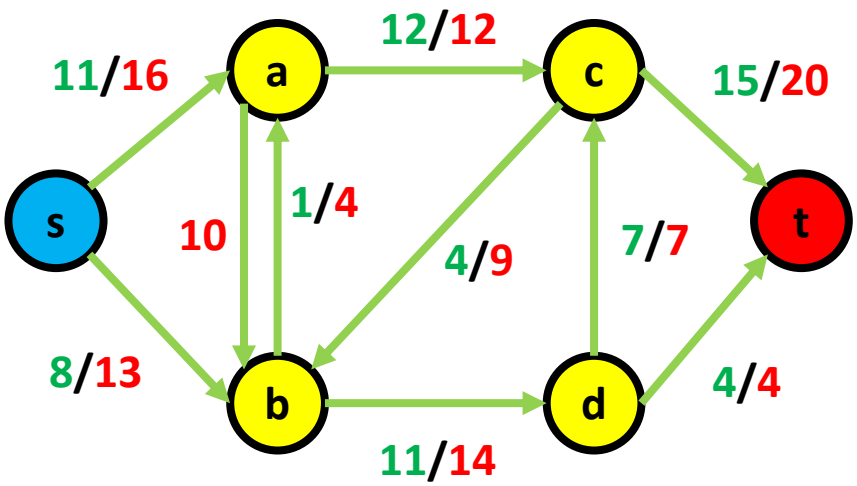


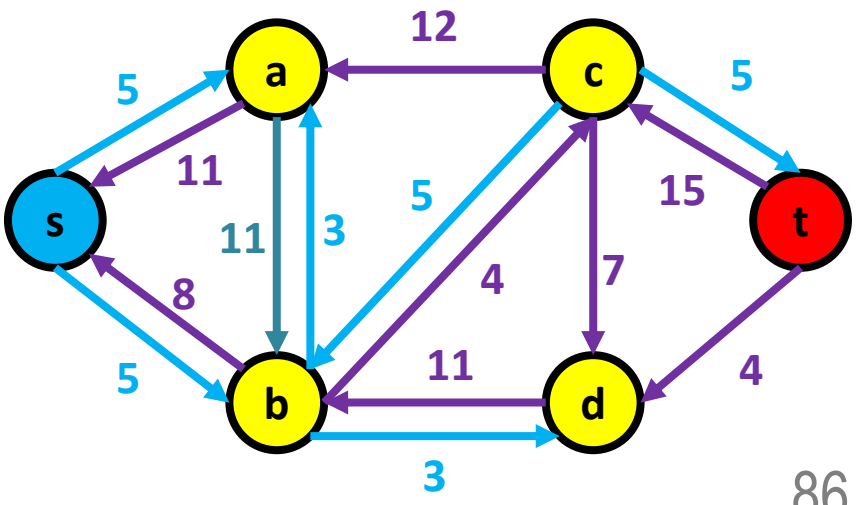


Residual network

Another freaking network

- What is a residual network?
  - Same vertices
  - Forward edge/ residual edge for remaining capacity
  - Backward edge/ reversible edge for flow that can be cancelled
  - Simple graph, so multi edges are merged together
  - Also note, sum of the edge between 2 vertices same as the edge capacity



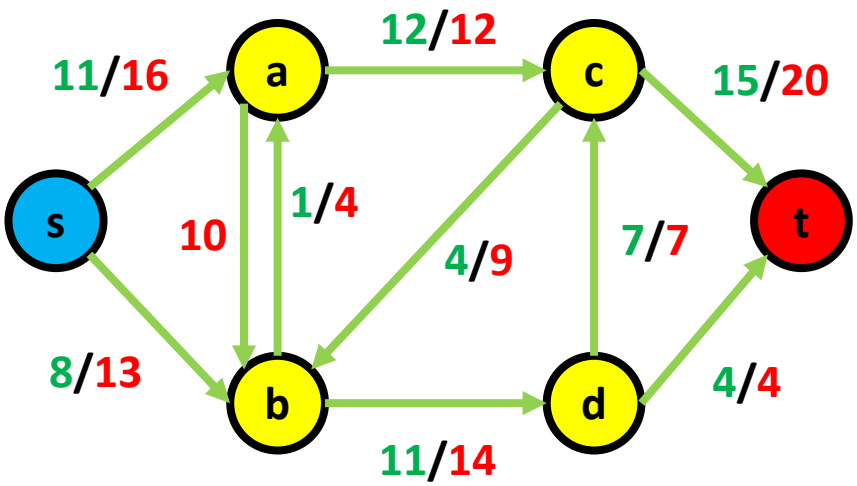


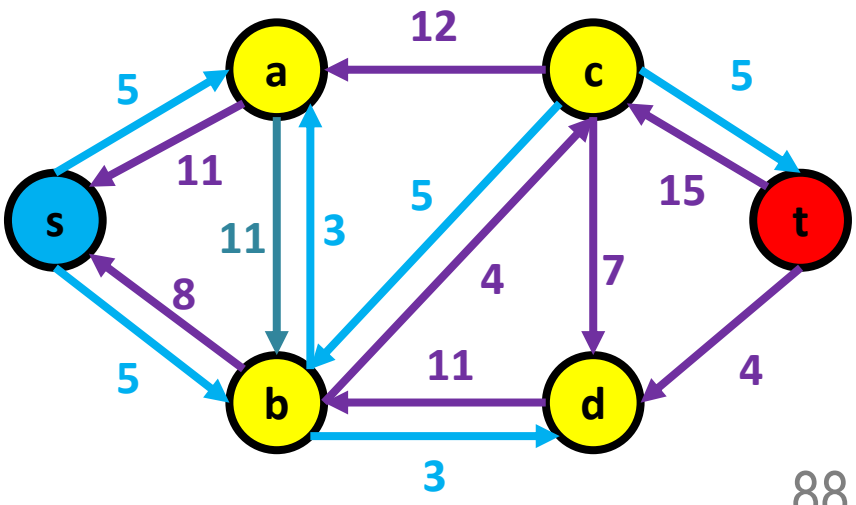
MONASH University

Questions?

Residual network

So what is the purpose?

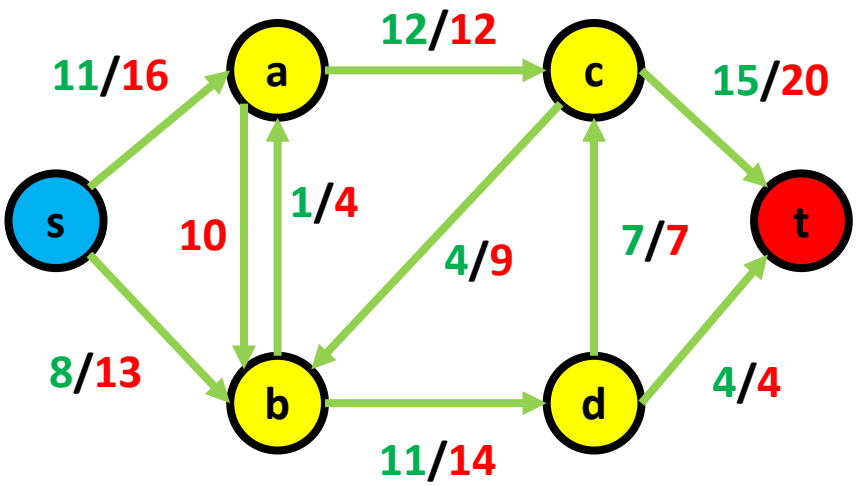


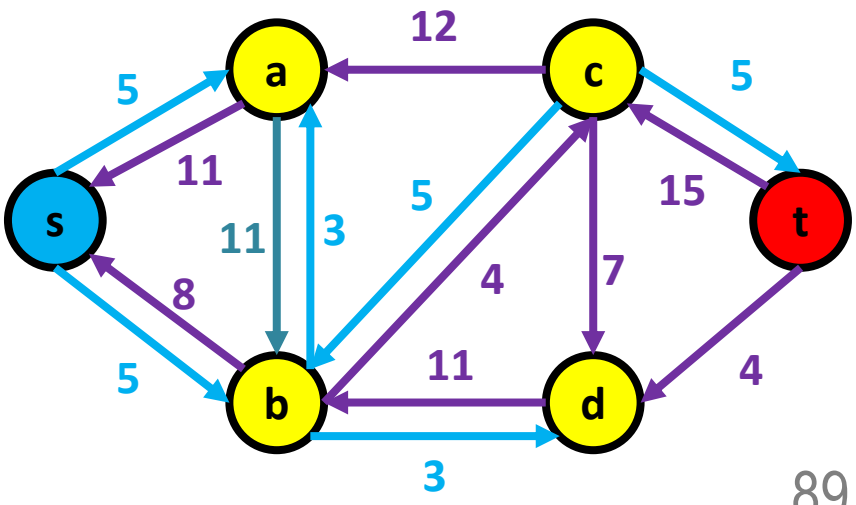


Residual network

Measuring the potential of a network

- Stores the network potential

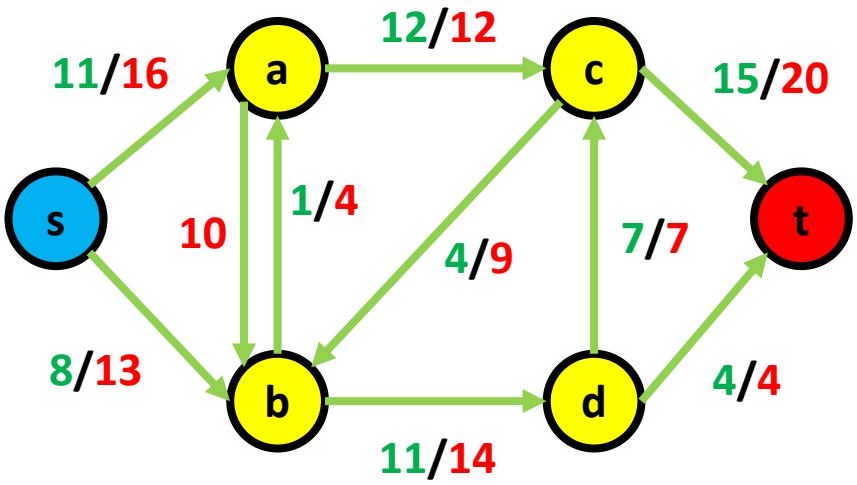


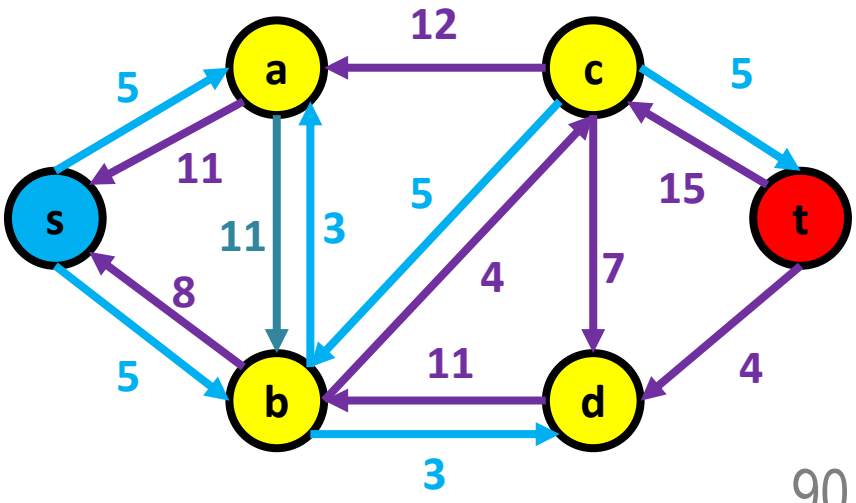


Residual network

Measuring the potential of a network

- Stores the network potential
  - Which we will unleash…

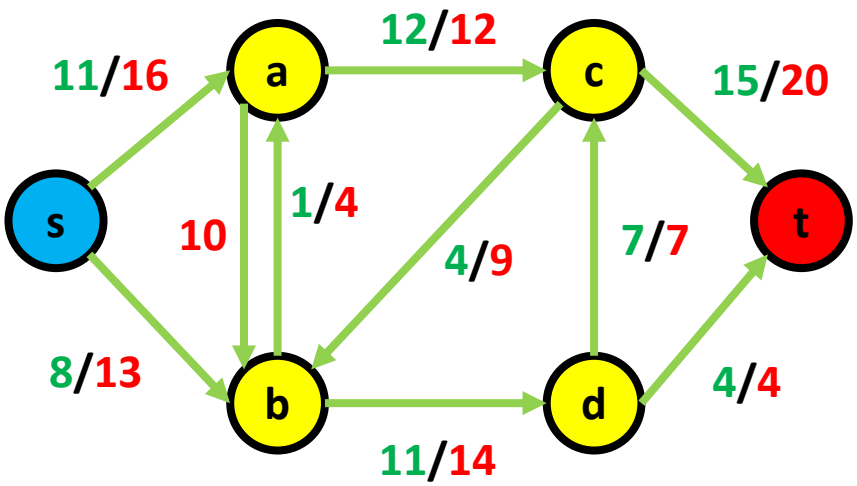


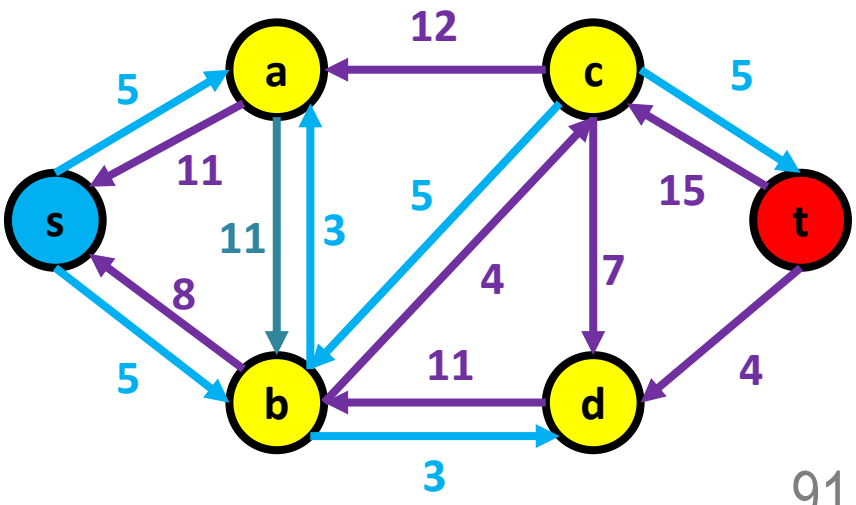


Residual network

Measuring the potential of a network

- Stores the network potential
  - Which we will unleash… via path augmentation!

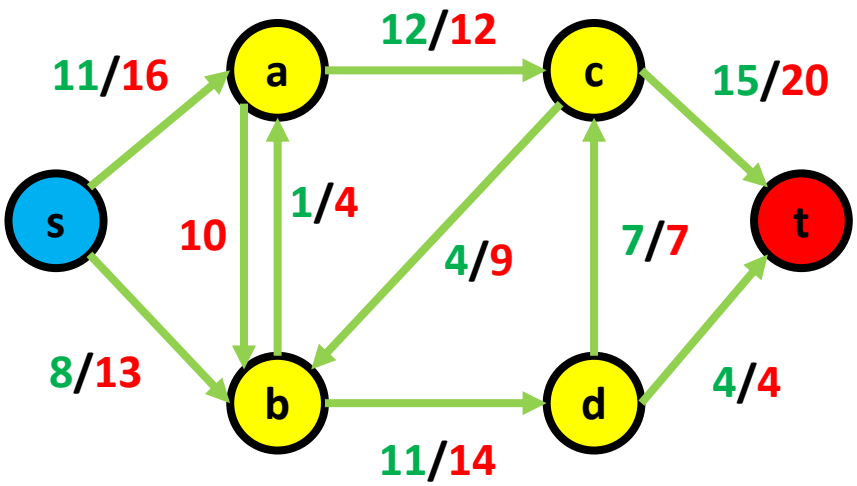


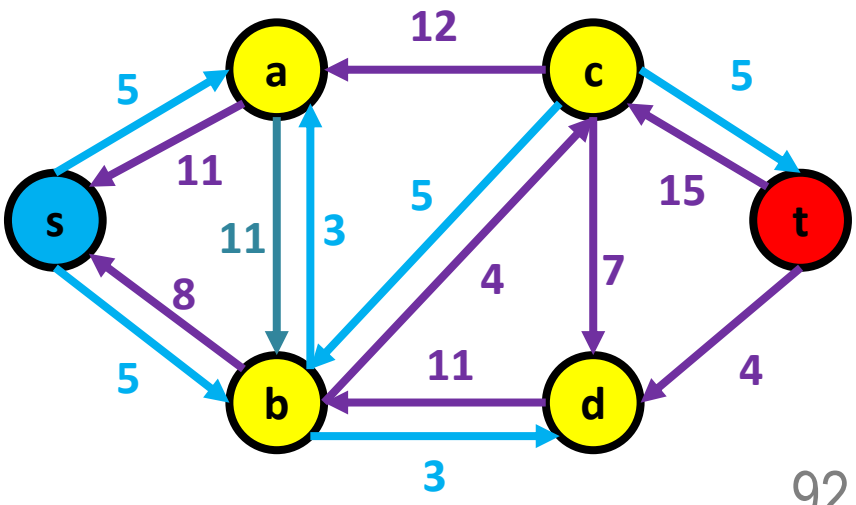


Residual network

Measuring the potential of a network

- So what is path augmentation?

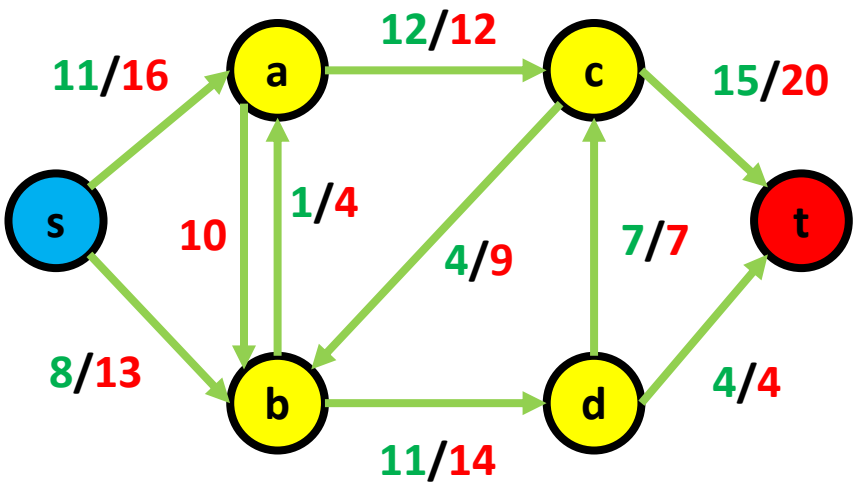


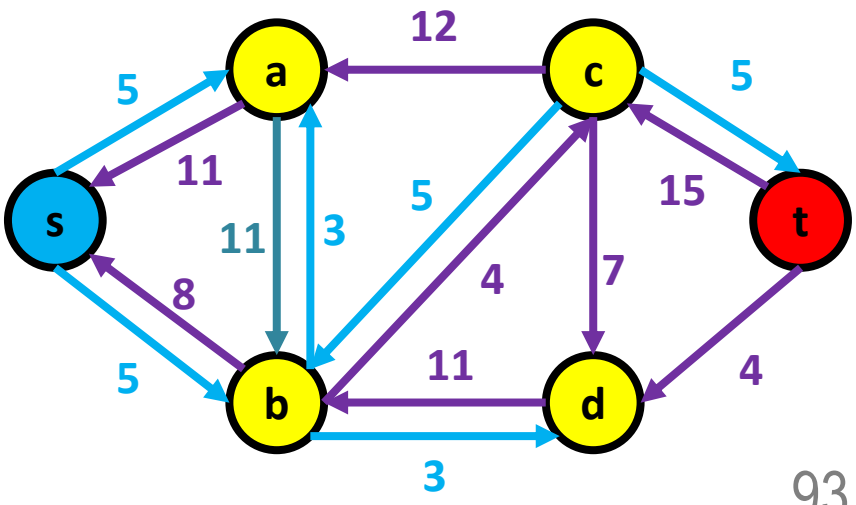


Path Augmentation

Traversal in residual network

- So what is path augmentation?
  - A traversal in the residual network

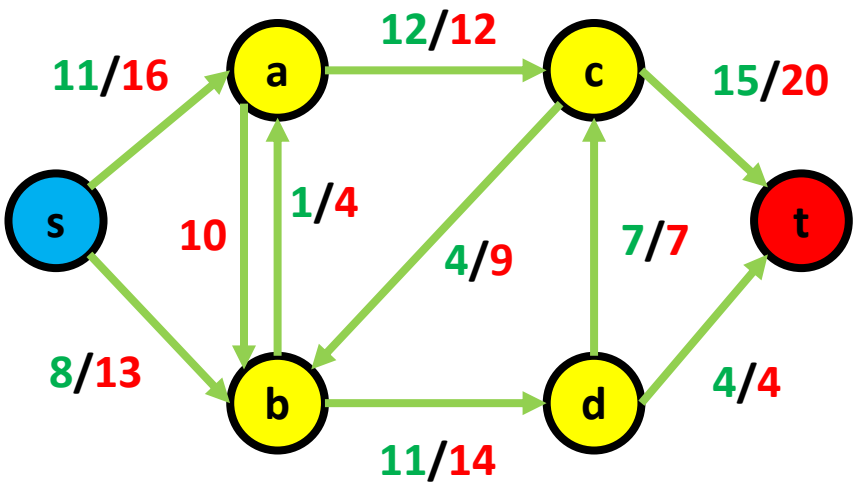


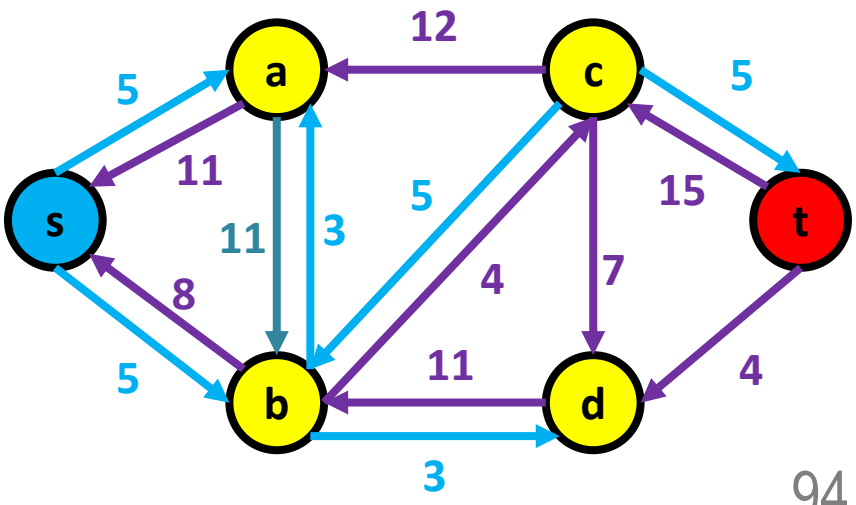


Path Augmentation

Traversal in residual network

- So what is path augmentation?
  - A traversal in the residual network
  - From source, to target



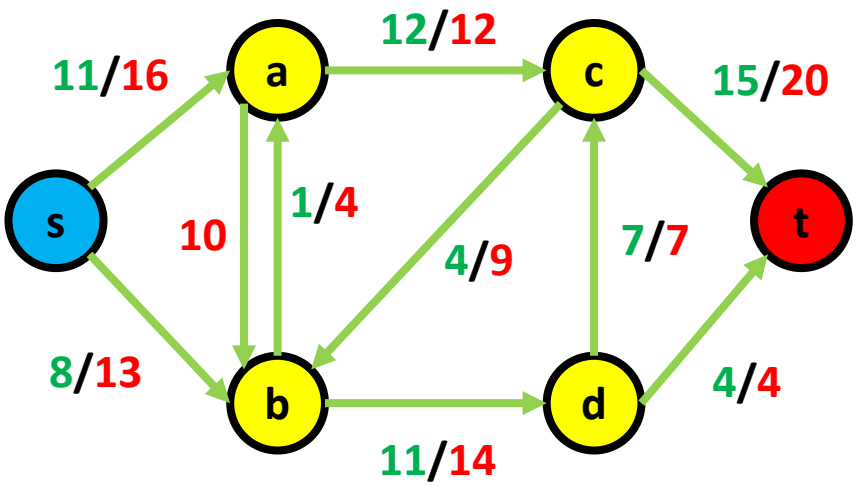


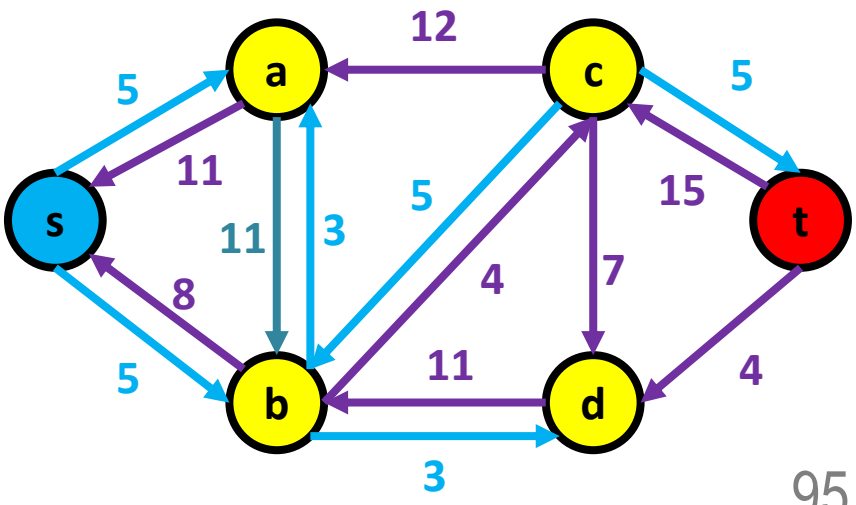
Path Augmentation

Traversal in residual network

MONASH University

- So what is path augmentation?
  - A traversal in the residual network
  - From source, to target
  - Following the edges in the residual network



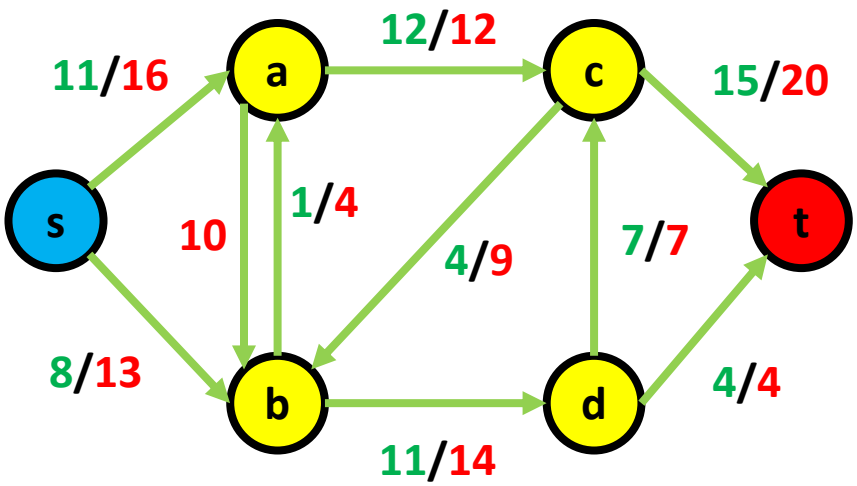


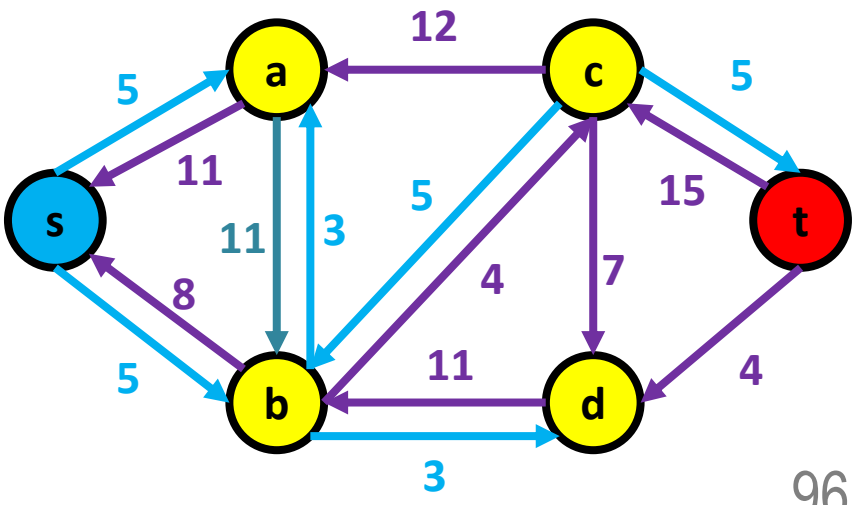
Path Augmentation

Traversal in residual network

MONASH University

- So what is path augmentation?
  - A traversal in the residual network
    - BFS! Or DFS!
  - From source, to target
  - Following the edges in the residual network

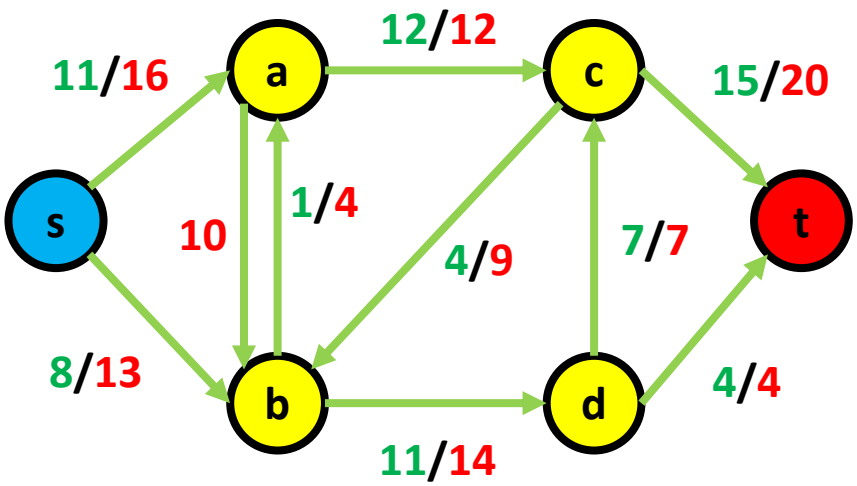


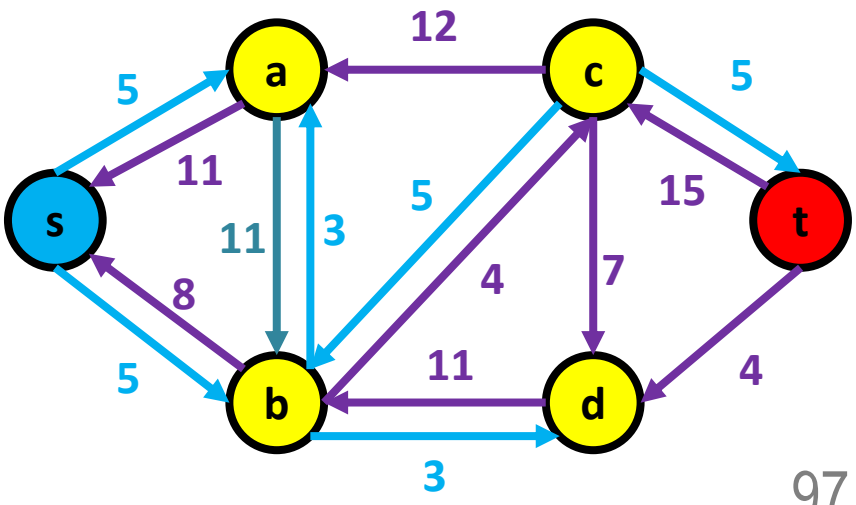


Path Augmentation

Traversal in residual network

- Is there a path here?

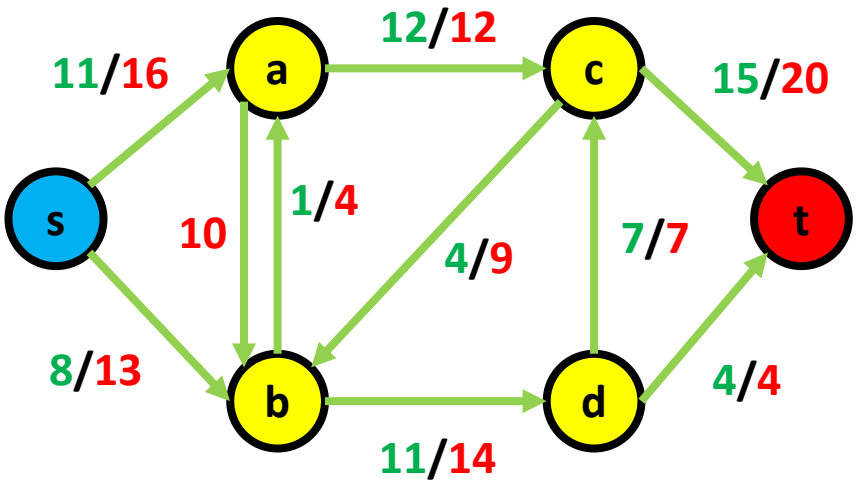


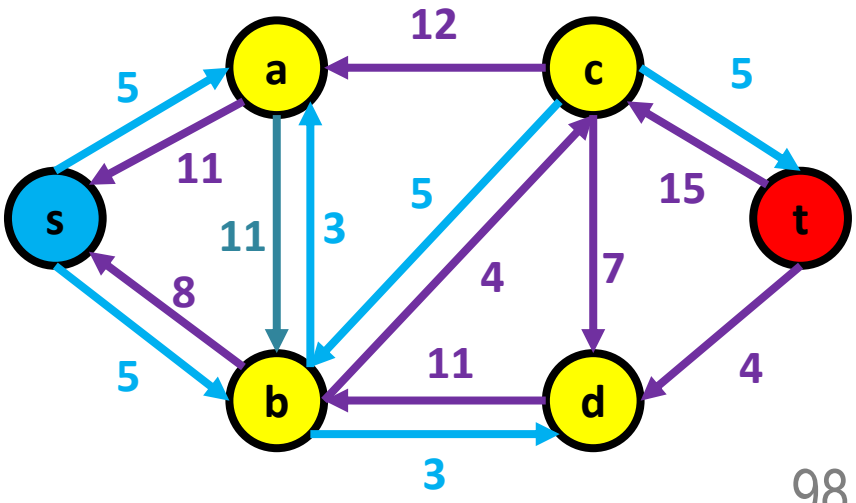


Path Augmentation

Traversal in residual network

- Is there a path here?
  - s -> b -> c -> t

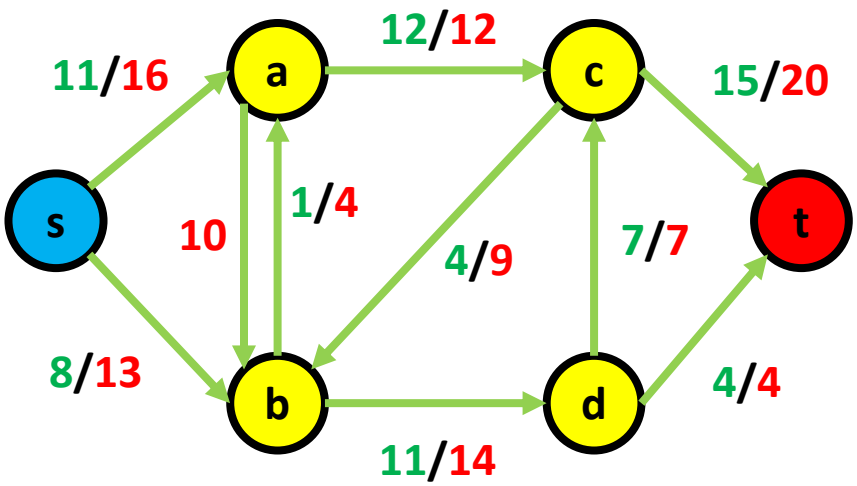


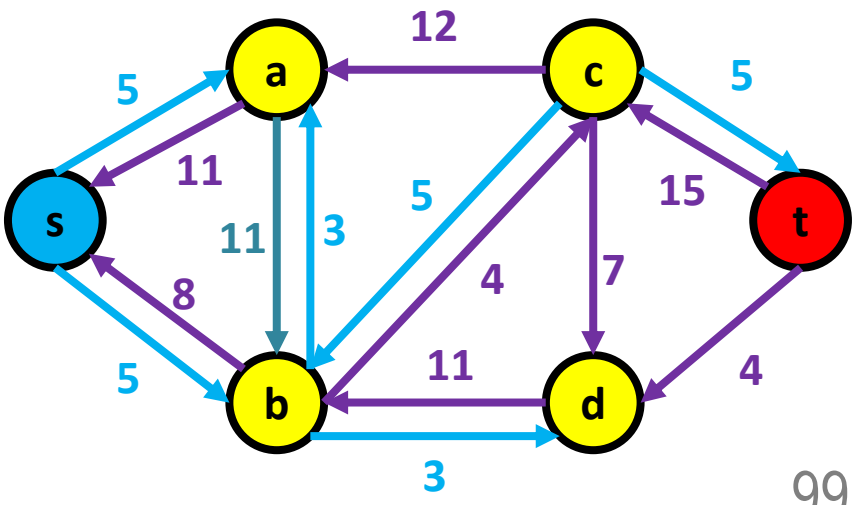


Path Augmentation

Traversal in residual network

- Is there a path here?
  - s -> b -> c -> t
  - s -> a -> b -> c -> t

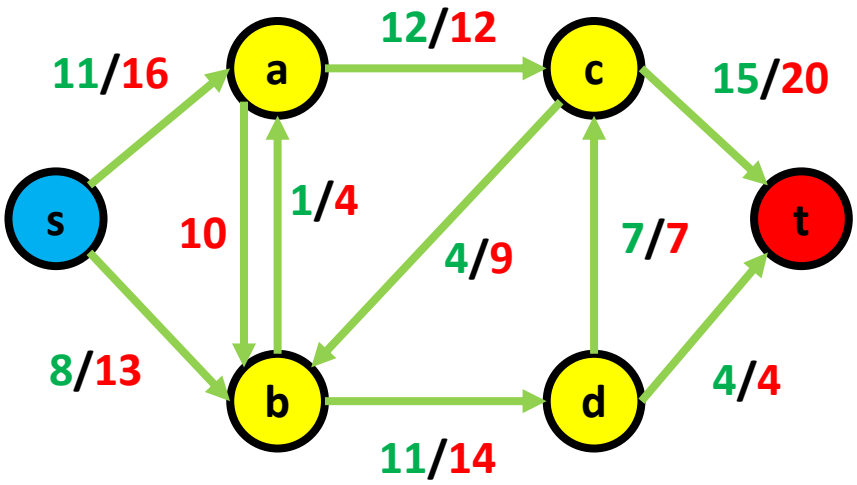


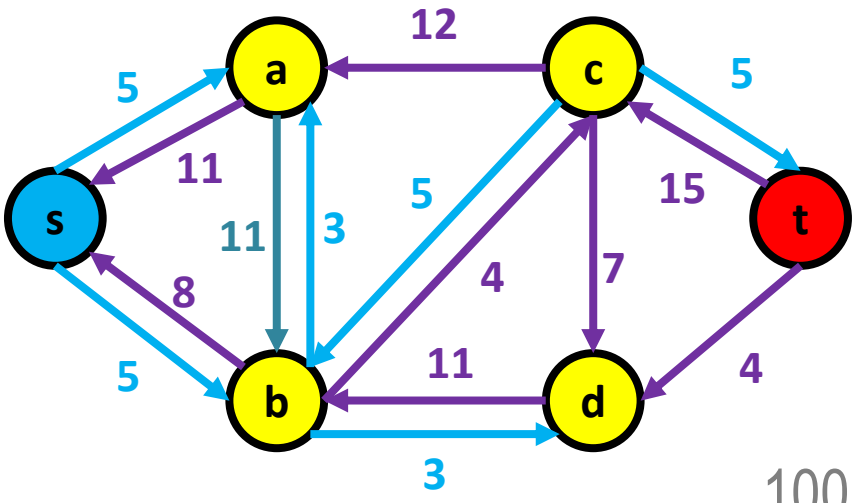


Path Augmentation

Traversal in residual network

- Is there a path here?
  - s -> b -> c -> t… let us look at this one first
  - s -> a -> b -> c -> t

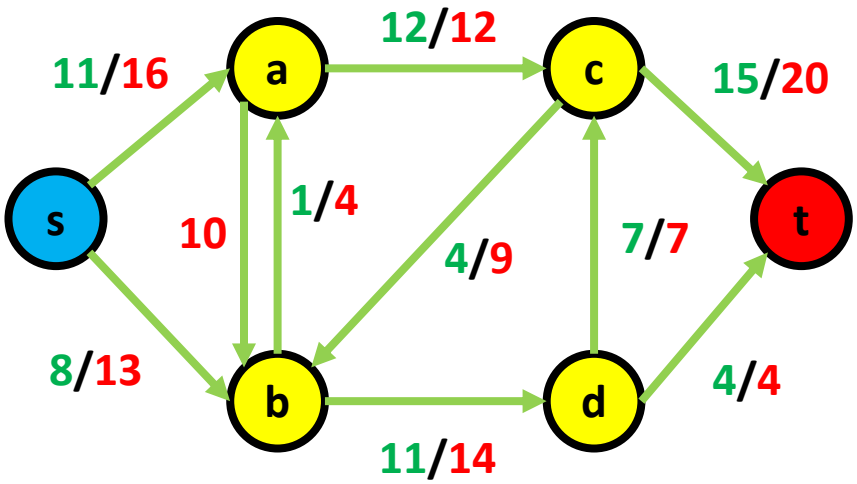


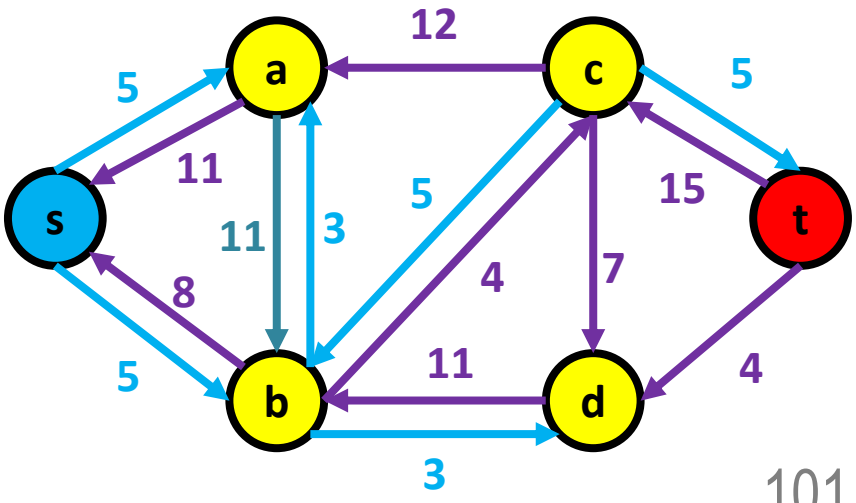


Path Augmentation

Traversal in residual network

- Is there a path here?
  - s -> b -> c -> t… let us look at this one first

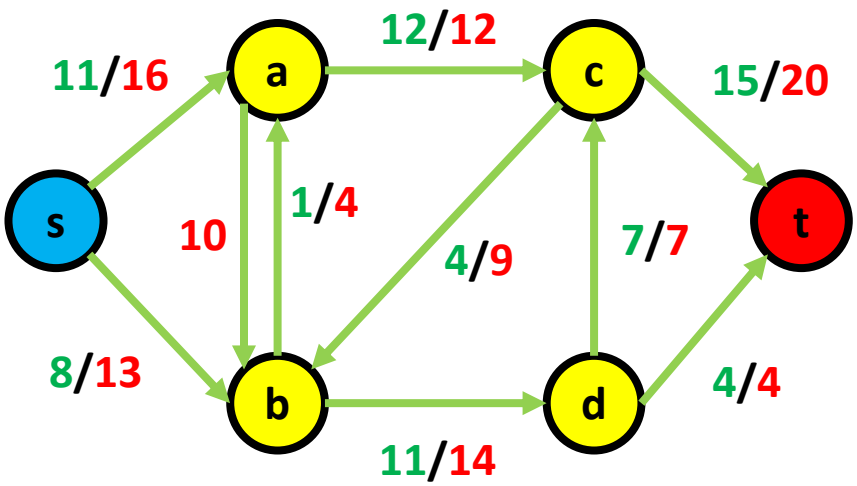


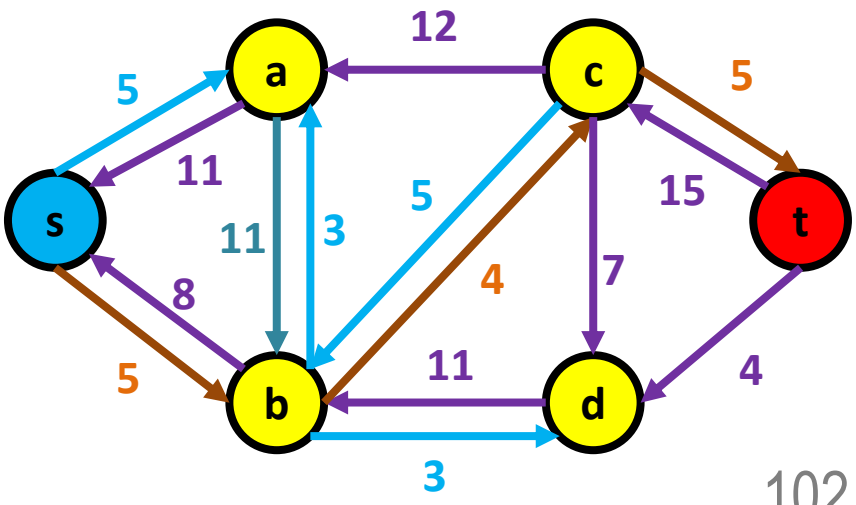


Path Augmentation

Traversal in residual network

- Is there a path here?
  - s -> b -> c -> t… let us look at this one first

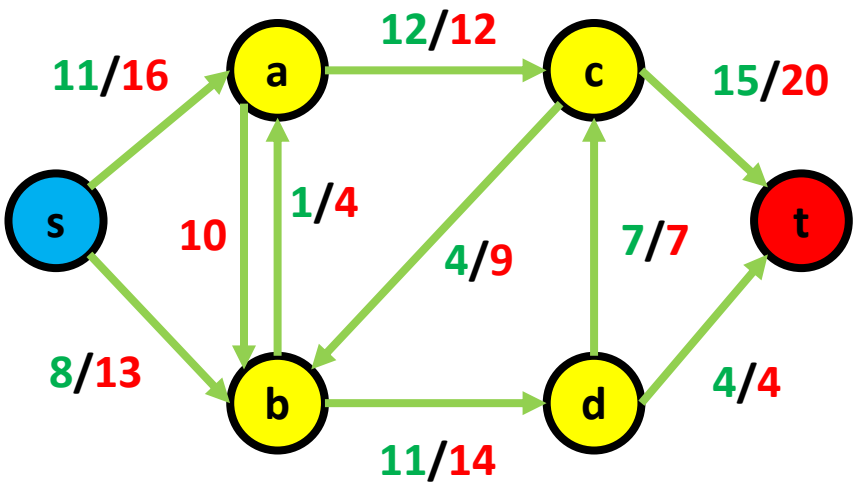


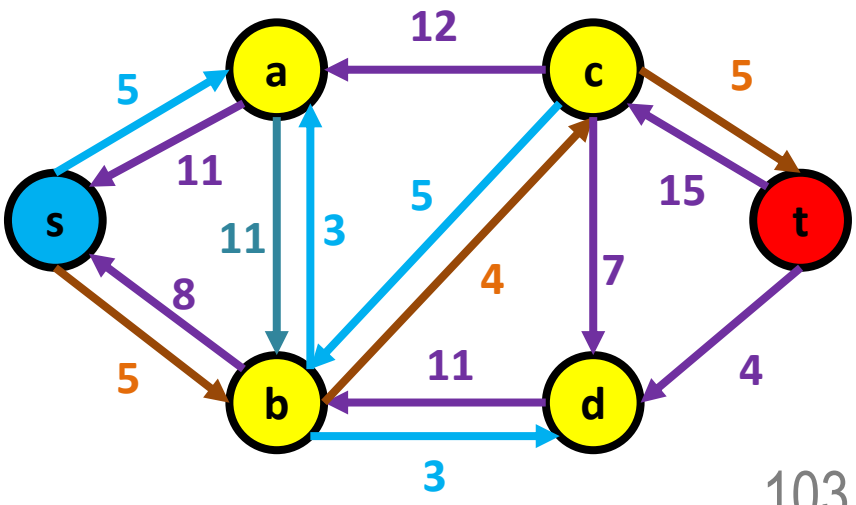


Path Augmentation

Traversal in residual network

- Is there a path here?
  - s -> b -> c -> t… let us look at this one first
  - That the smallest value which is 4
    - This value can flow from source to target

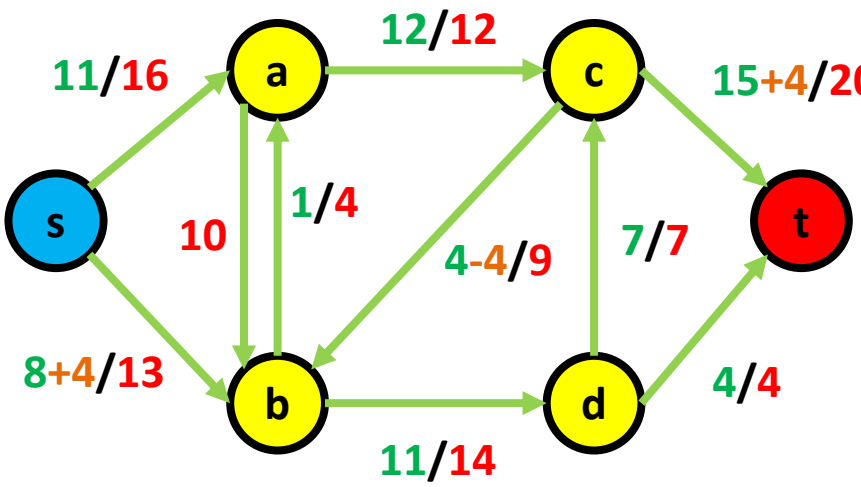


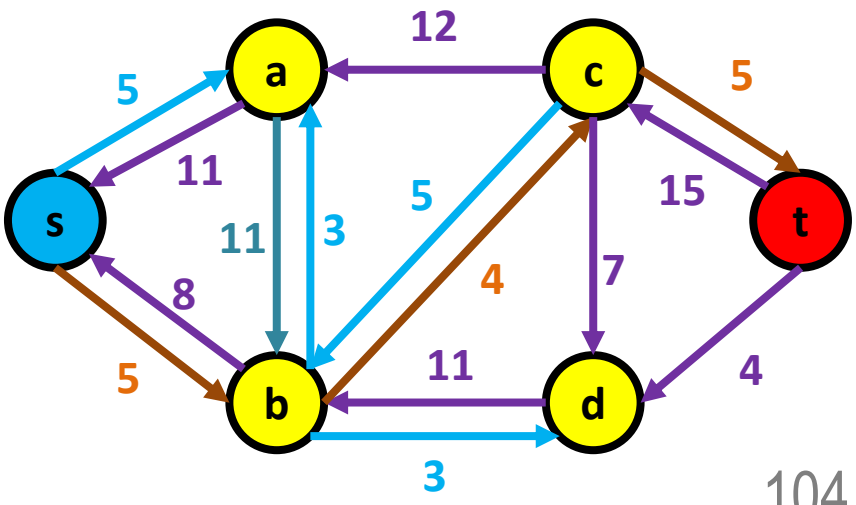


Path Augmentation

Traversal in residual network

- Is there a path here?
  - s -> b -> c -> t… let us look at this one first
  - That the smallest value which is 4
    - This value can flow from source to target

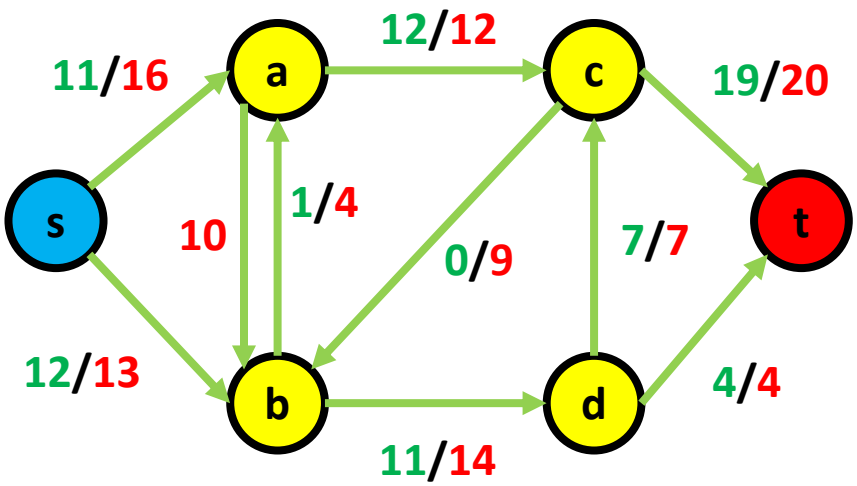


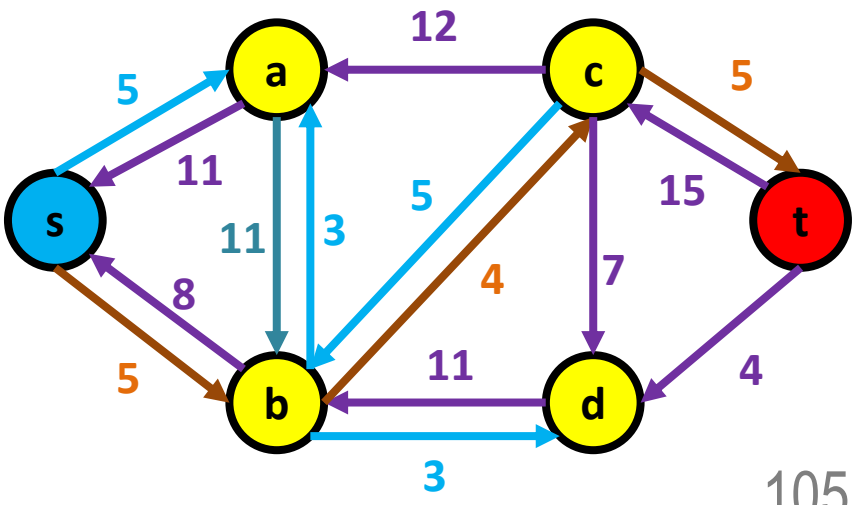


Path Augmentation

Traversal in residual network

- Is there a path here?
  - s -> b -> c -> t… let us look at this one first
  - That the smallest value which is 4
    - This value can flow from source to target

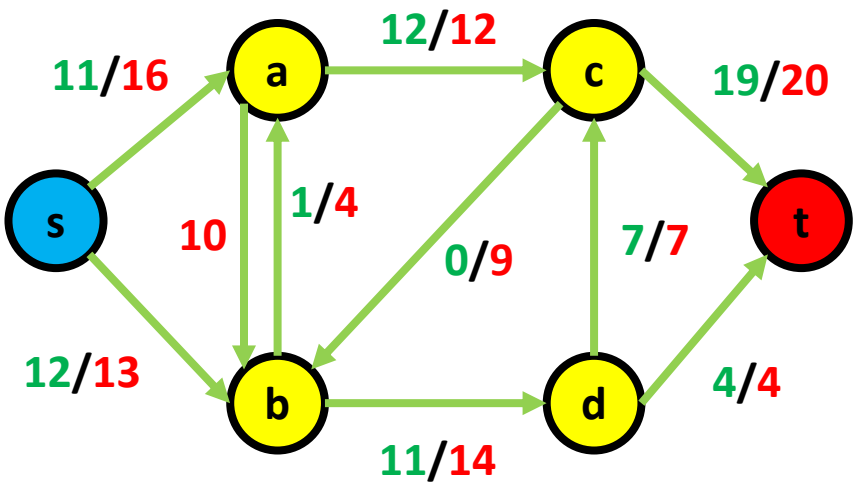


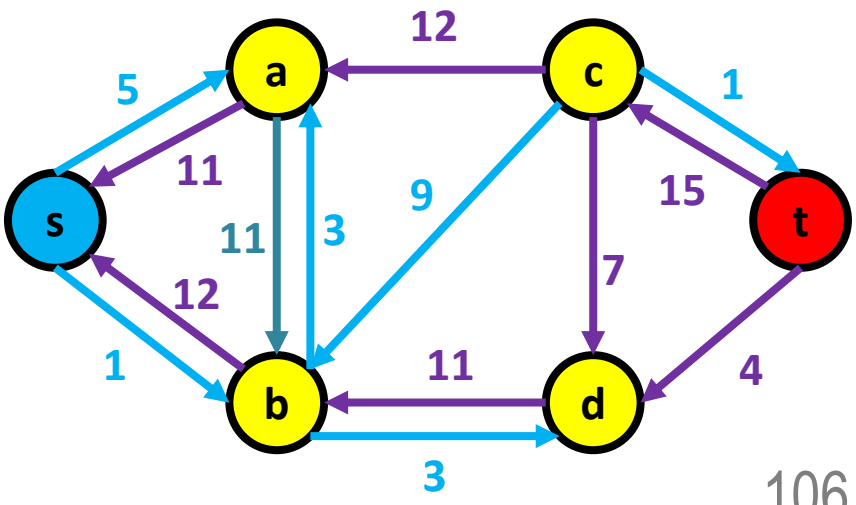


Path Augmentation

Traversal in residual network

- Is there a path here?
  - s -> b -> c -> t… let us look at this one first
  - That the smallest value which is 4
    - This value can flow from source to target

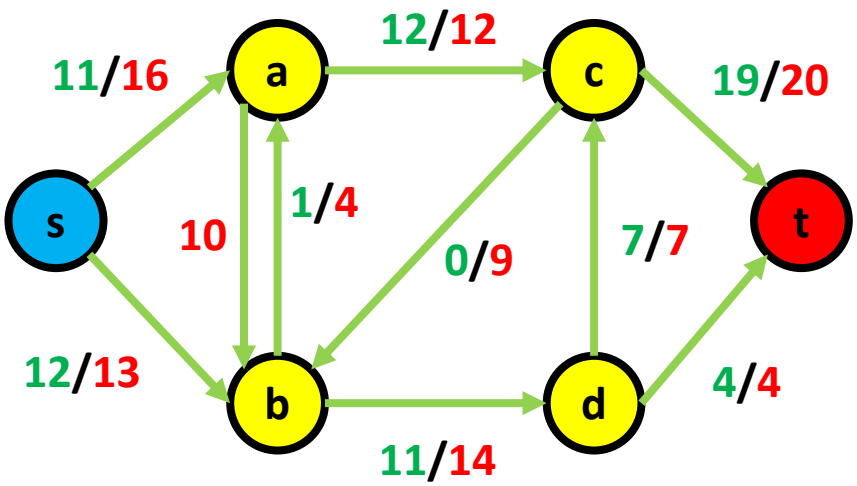


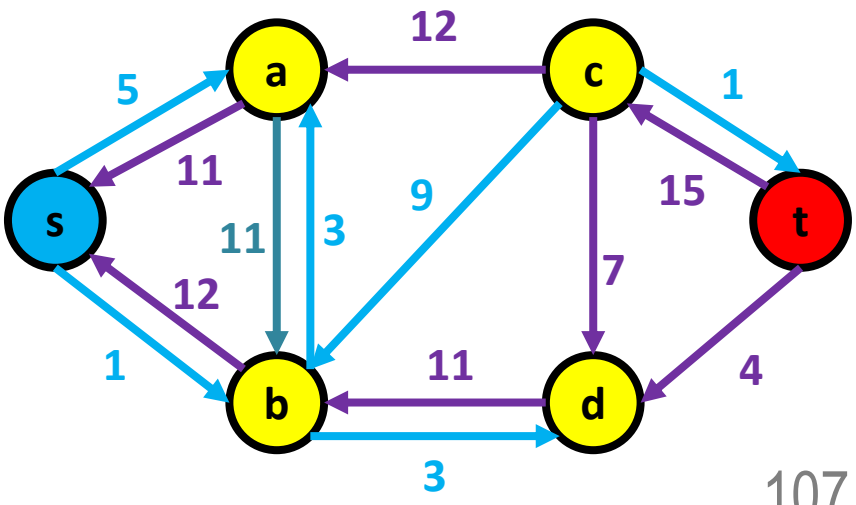


Path Augmentation

Traversal in residual network

- Is there a path here?
  - s -> b -> c -> t… let us look at this one first
  - That the smallest value which is 4
    - This value can flow from source to target
- With this, the flow in the network goes from 19 to 23!





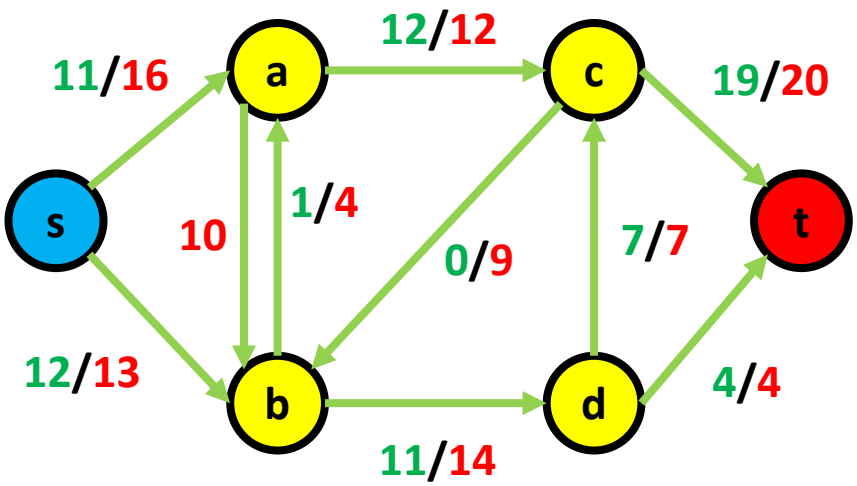
MONASH University

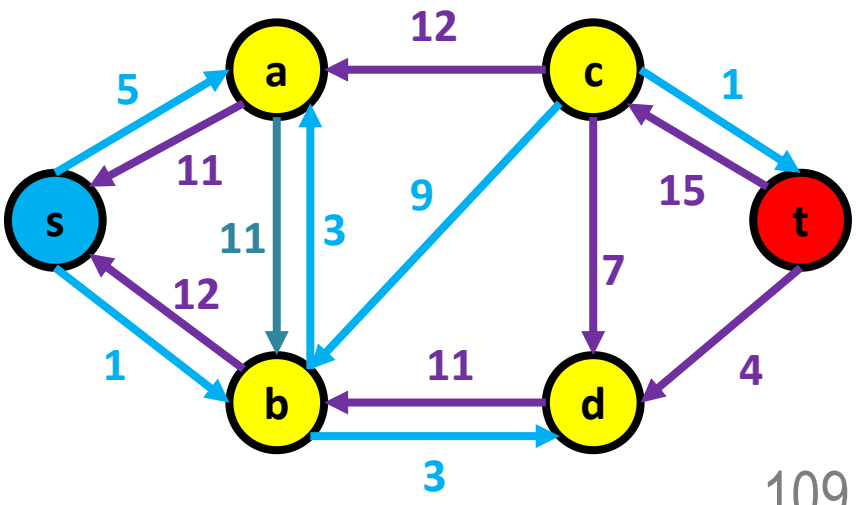
Questions?

Path Augmentation

Traversal in residual network

- Is there another path here?

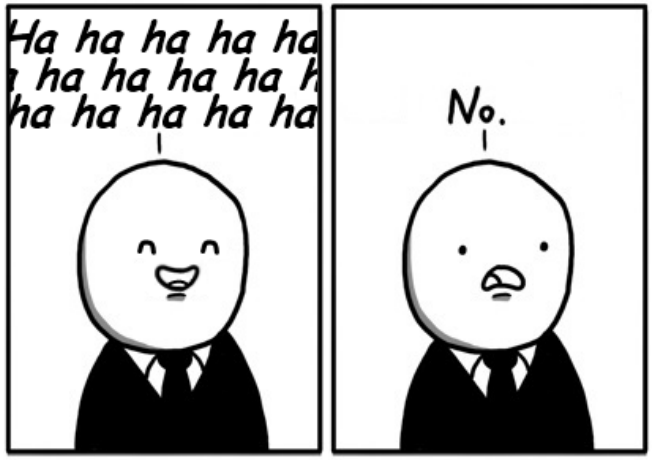


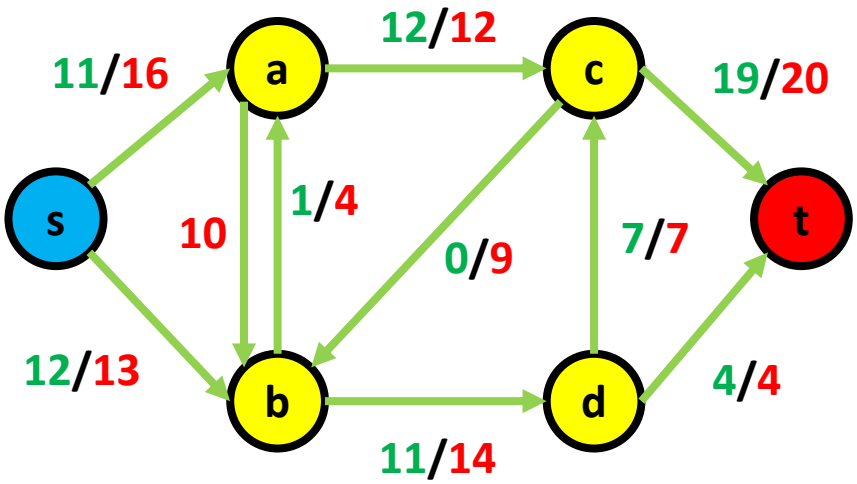


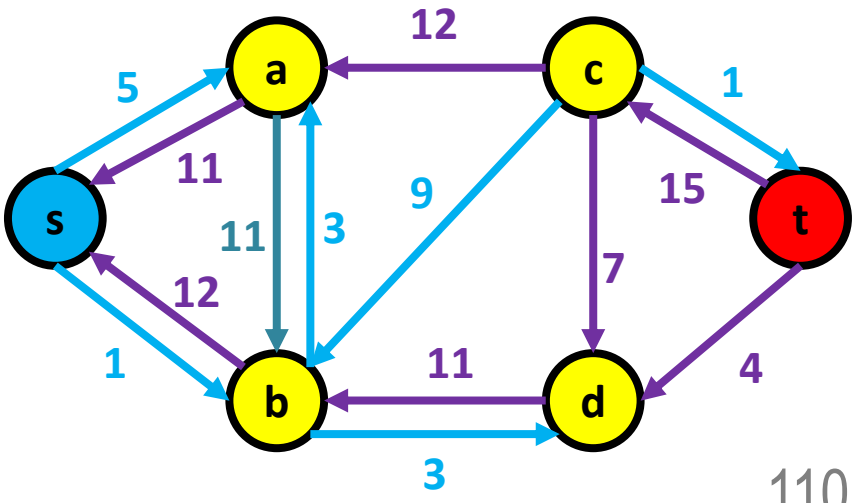
Path Augmentation

Traversal in residual network

- Is there another path here?
  - NO!







MONASH University

Questions?

Ford-Fulkerson Method

Finding the maximum flow

MONASH University

- So we have learnt all the components for the Ford-Fulkerson Method…

Ford-Fulkerson Method

Finding the maximum flow

MONASH University

- So we have learnt all the components for the Ford-Fulkerson Method…
  - Residual network
  - Path augmentation

Ford-Fulkerson Method

Finding the maximum flow

MONASH University

- So we have learnt all the components for the Ford-Fulkerson Method…
  - Residual network
  - Path augmentation
- Now let us look at the algorithm…

MONASH University

Break?

Ford-Fulkerson Method

Finding the maximum flow

MONASH University

- Let break it down slowly…

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```

Ford-Fulkerson Method

Finding the maximum flow

MONASH University

- Let break it down slowly…

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```

See slide 62 to 86

Ford-Fulkerson Method

Finding the maximum flow

MONASH University

- Let break it down slowly…

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```

See slide 88 to 107

Ford-Fulkerson Method

Finding the maximum flow

MONASH University

- Let break it down slowly…

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```

Ends when it is like slide 109 to 110

Ford-Fulkerson Method

Finding the maximum flow

MONASH University

- And that's all…

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```

MONASH University

Questions?

Ford-Fulkerson Method

Finding the maximum flow

MONASH University

- Want a trial run?

Ford-Fulkerson Method

Finding the maximum flow

- Want a trial run?

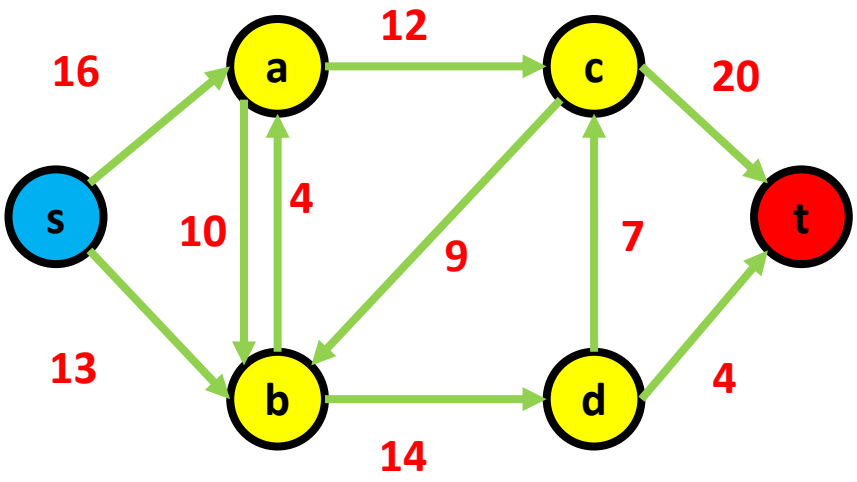
```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?

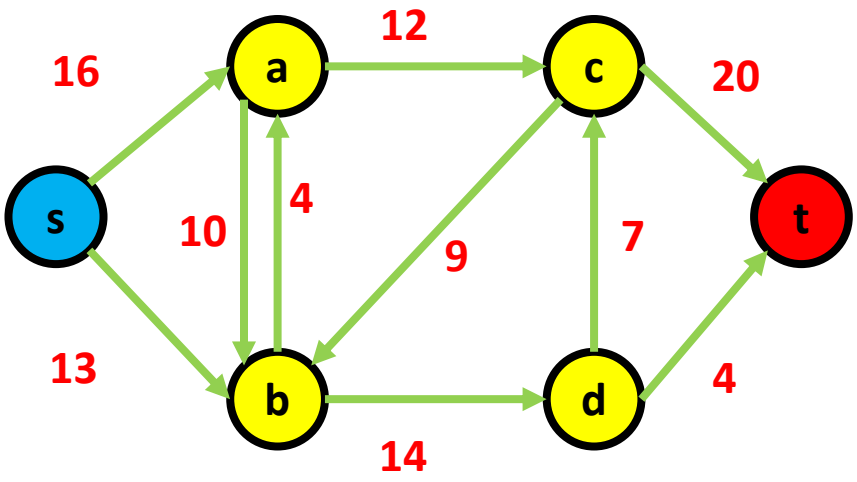


Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 0

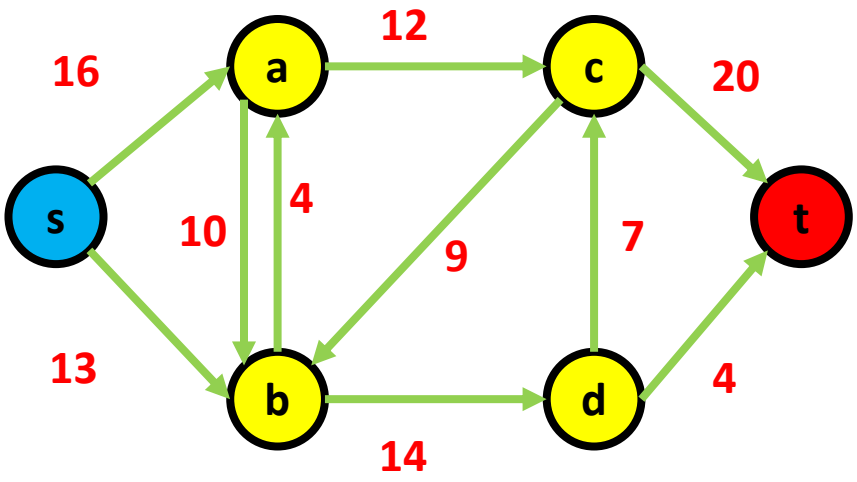


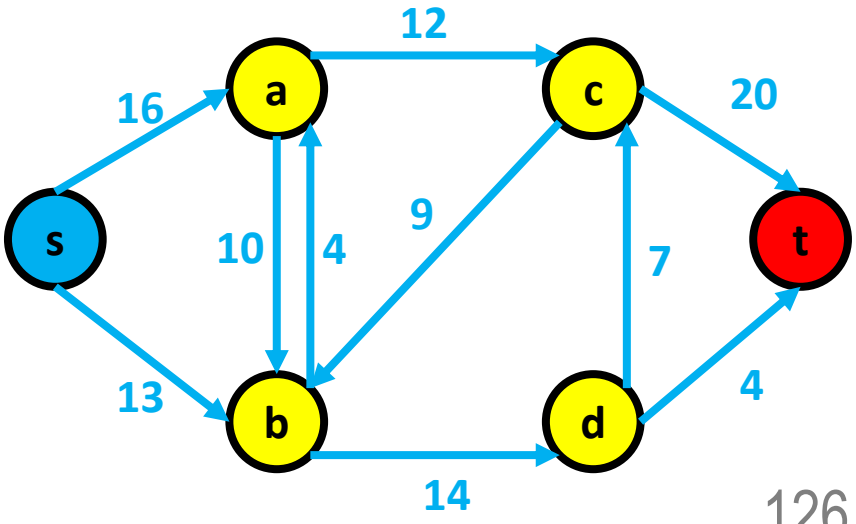
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 0
  - Make residual network



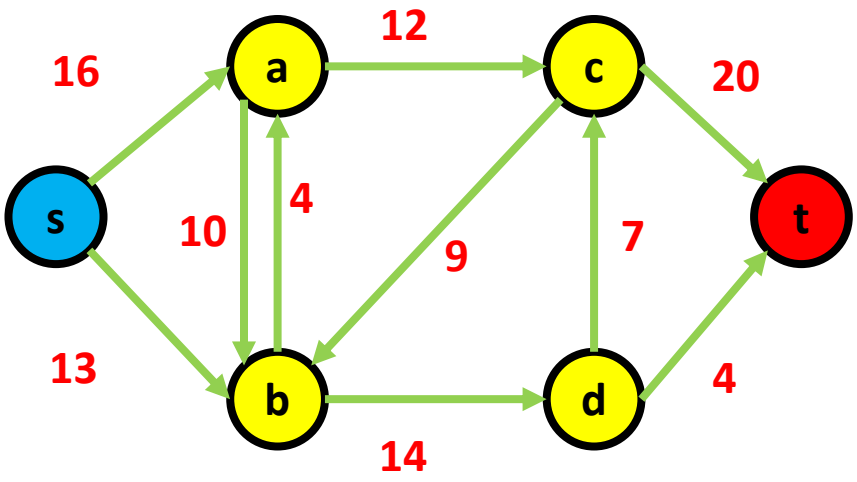


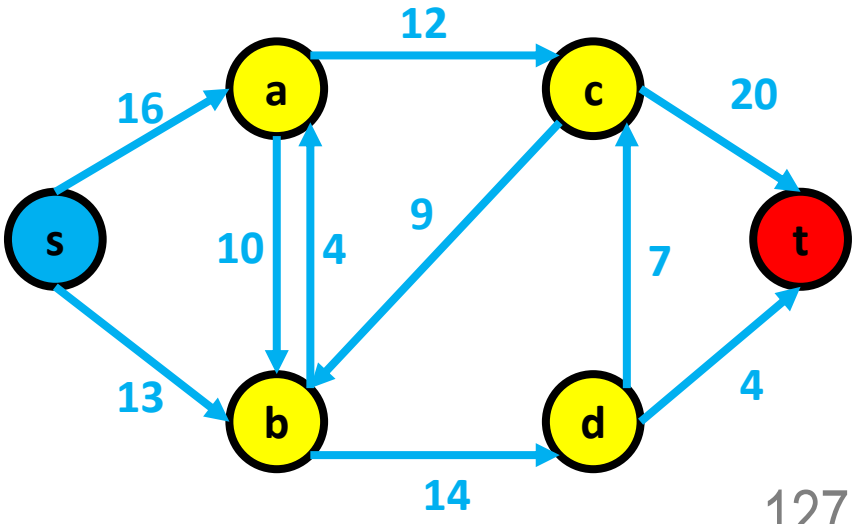
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 0
  - Make residual network
  - Do we have an augmenting path?



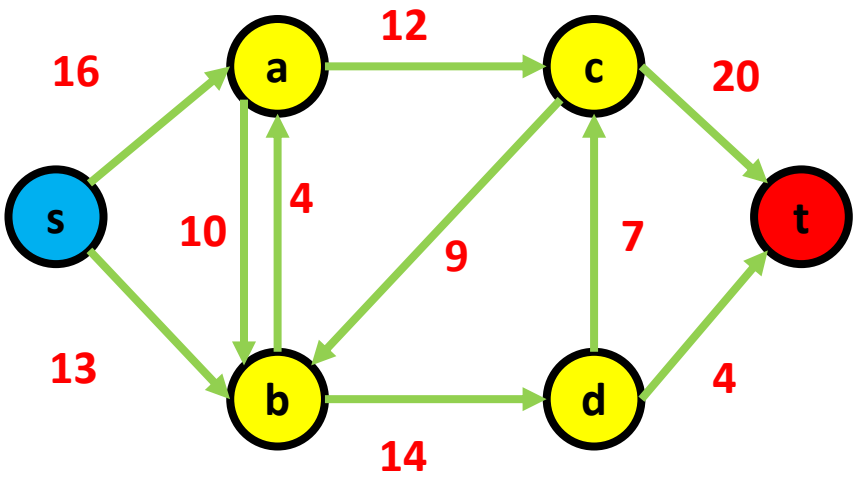


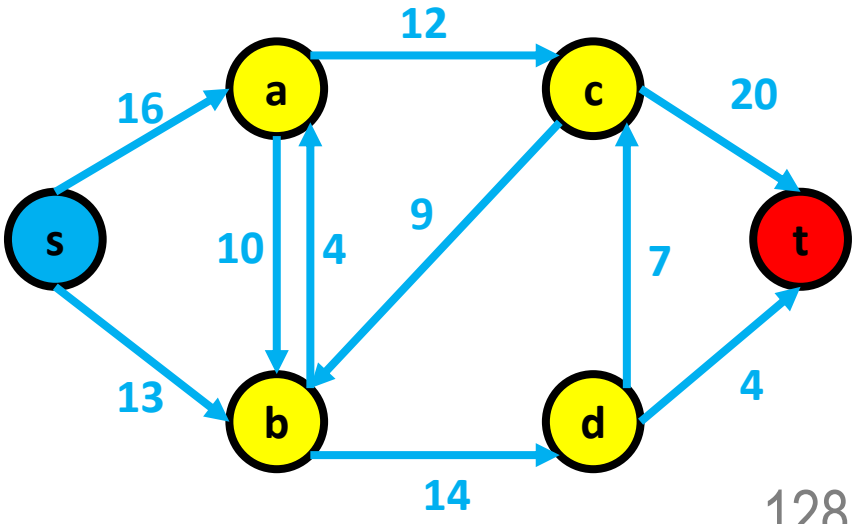
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 0
  - Make residual network
  - Do we have an augmenting path?
    - Yes! What is the path?



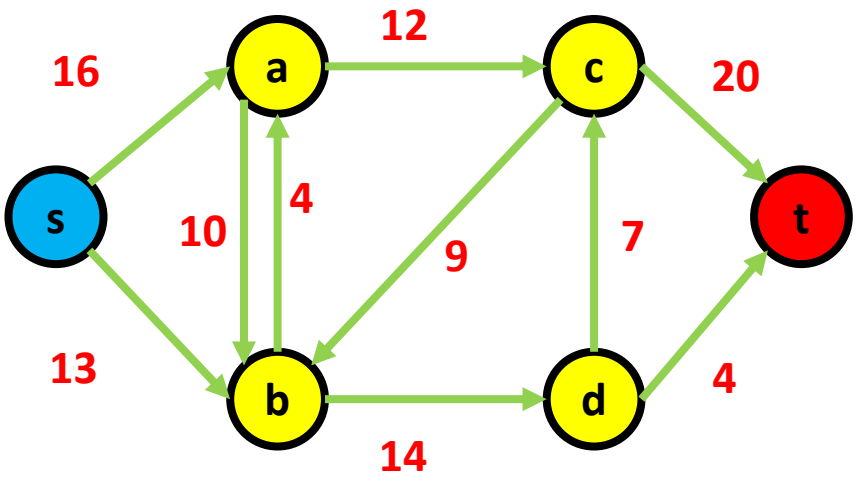


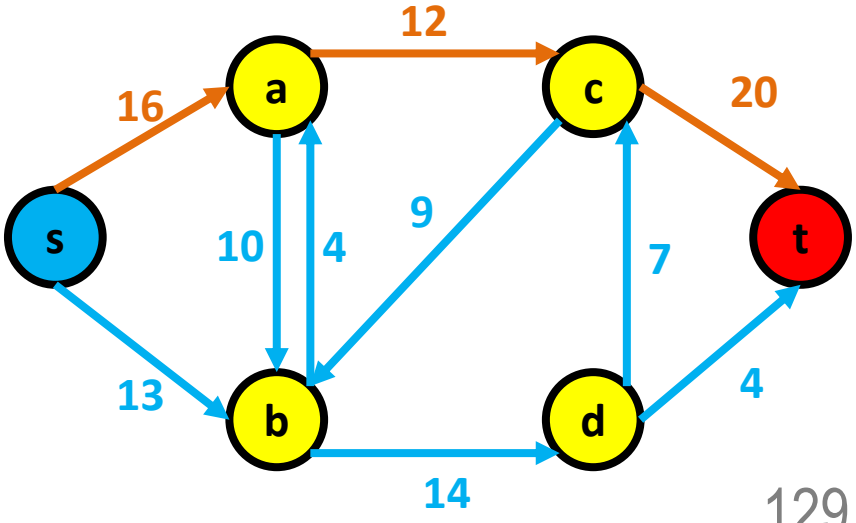
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 0
  - Make residual network
  - Do we have an augmenting path?
    - Yes! What is the path? s -> a -> c -> t



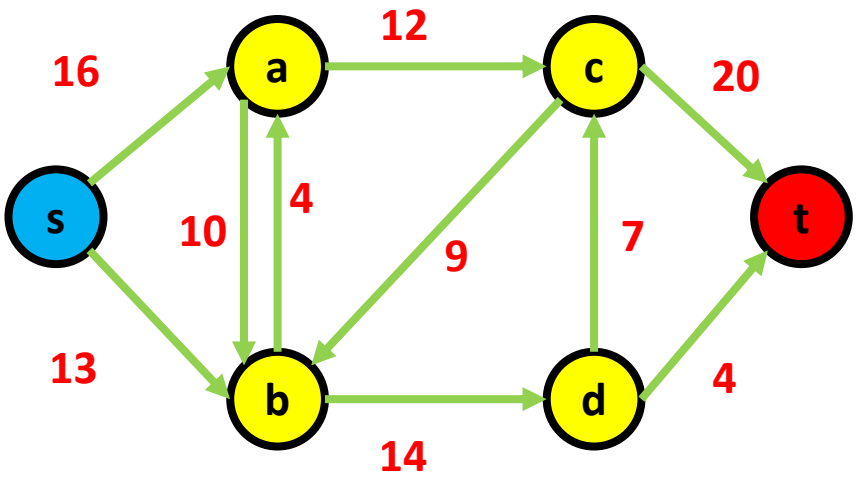


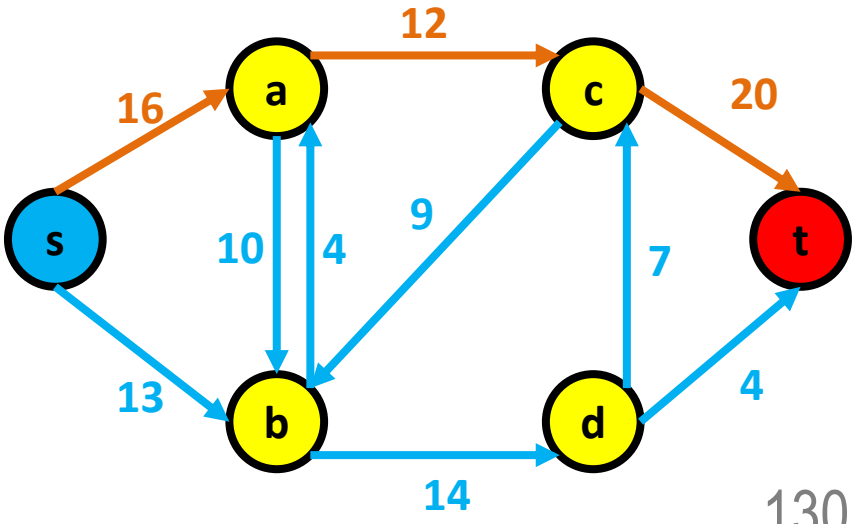
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 0
  - Make residual network
  - Do we have an augmenting path?
    - Yes! What is the path? s -> a -> c -> t
    - Take the smallest value, 12



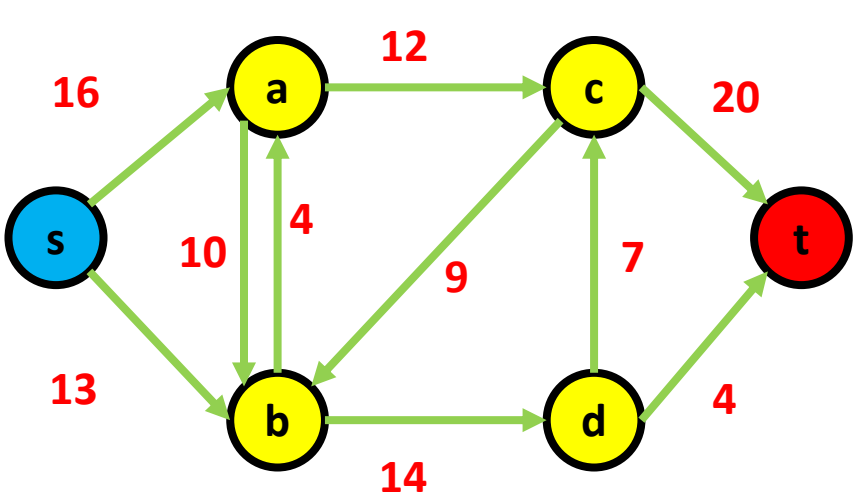


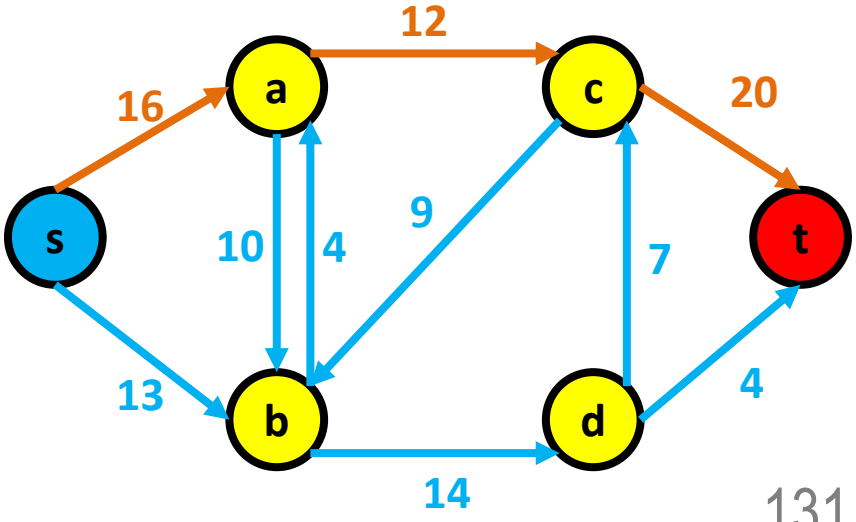
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 0
  - Make residual network
  - Do we have an augmenting path?
    - Yes! What is the path? s -> a -> c -> t
    - Take the smallest value, 12 and update the network



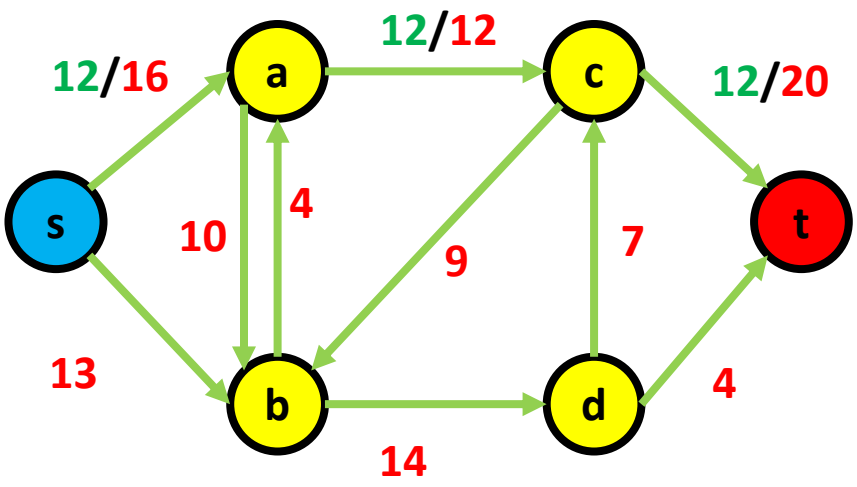


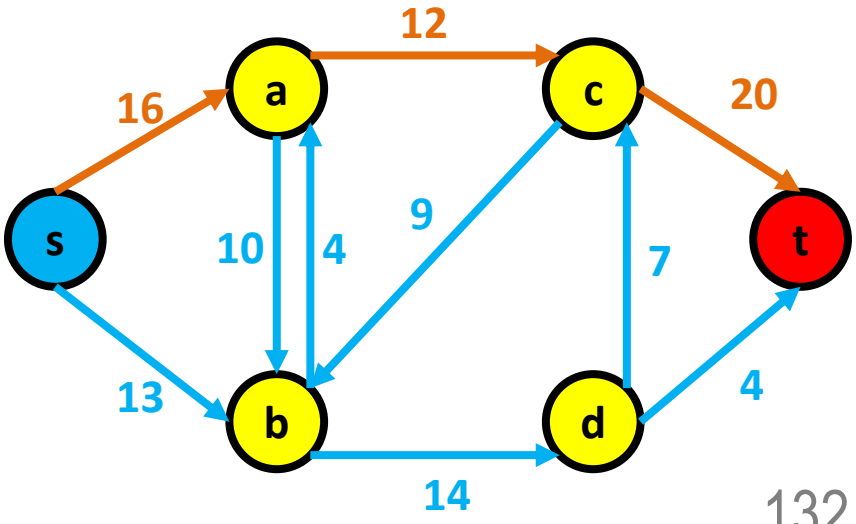
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 0
  - Make residual network
  - Do we have an augmenting path?
    - Yes! What is the path? s -> a -> c -> t
    - Take the smallest value, 12 and update the network



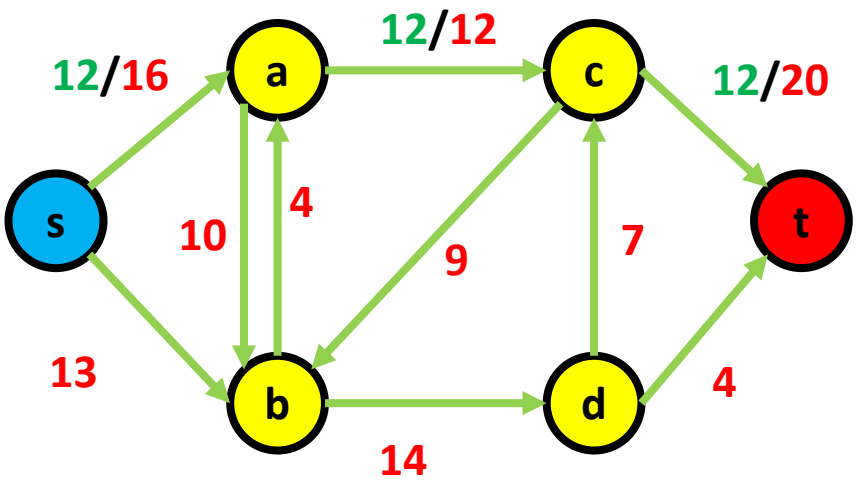


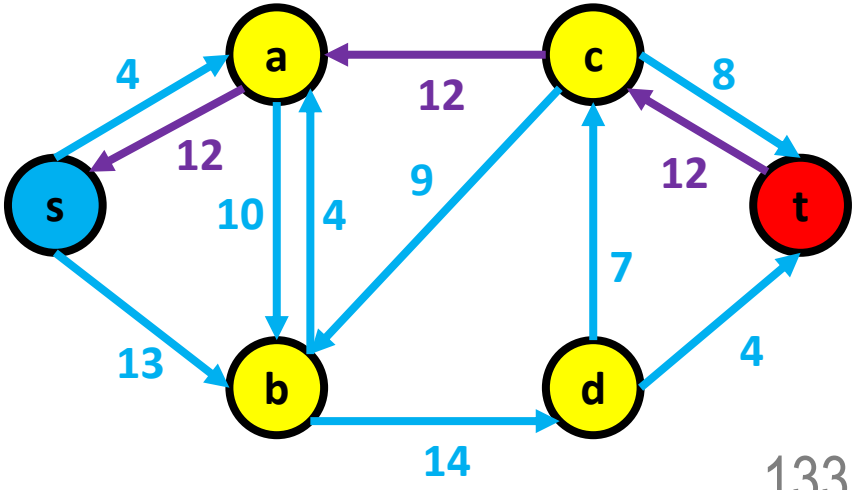
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 0
  - Make residual network
  - Do we have an augmenting path?
    - Yes! What is the path? s -> a -> c -> t
    - Take the smallest value, 12 and update the network
    - Update the residual network as well



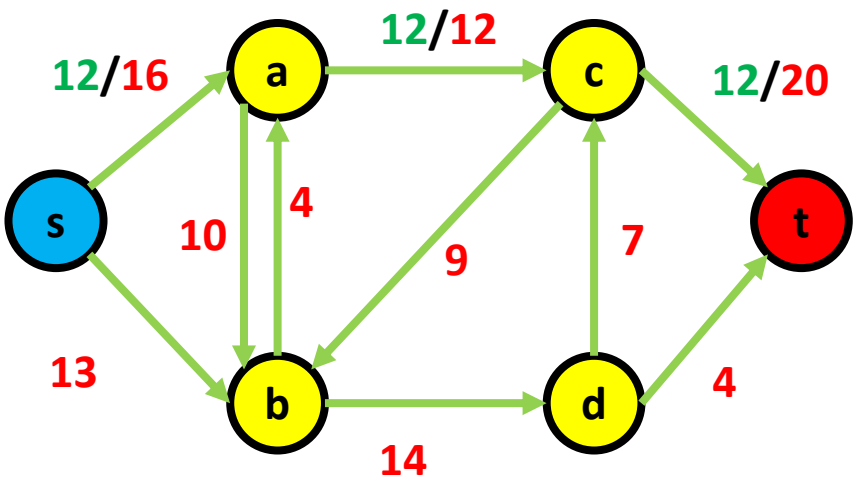


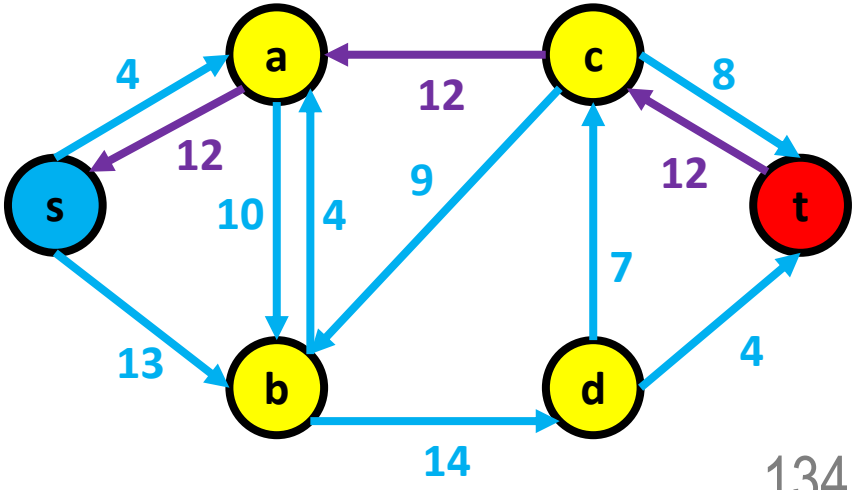
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = **0+12 = 12**
  - Make residual network
  - Do we have an augmenting path?
    - Yes! What is the path? s -> a -> c -> t
    - Take the smallest value, 12 and update the network
    - Update the residual network as well



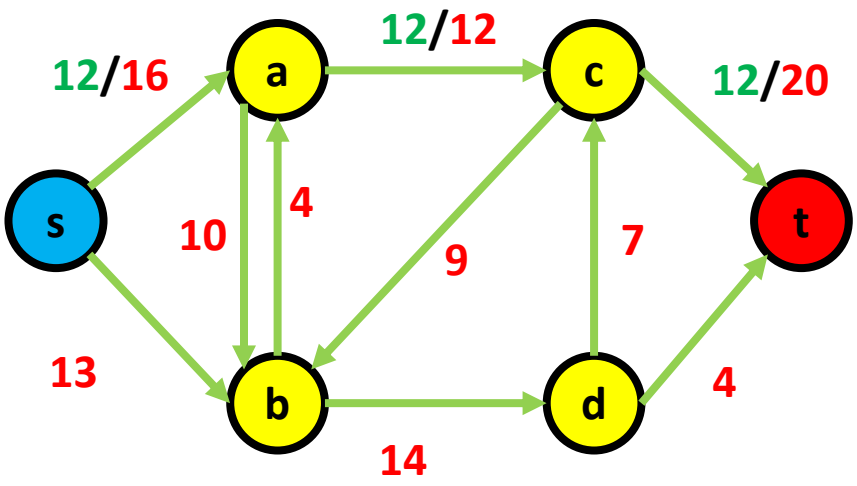


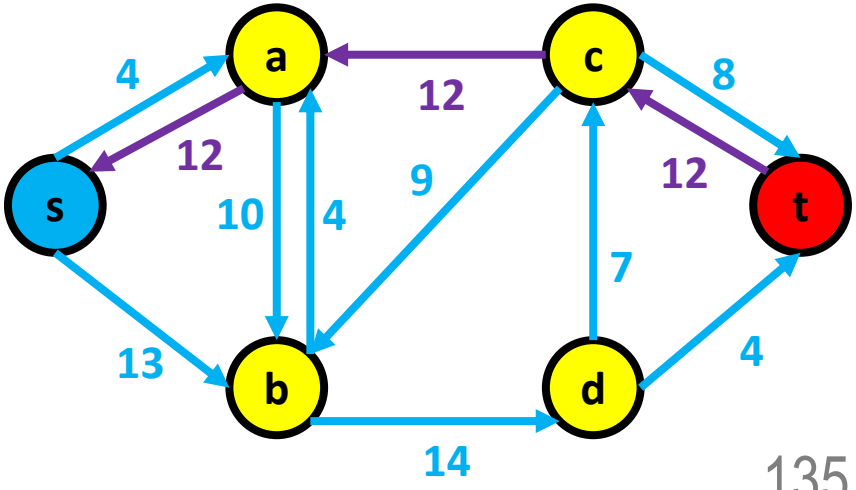
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 12
  - So we just repeat it



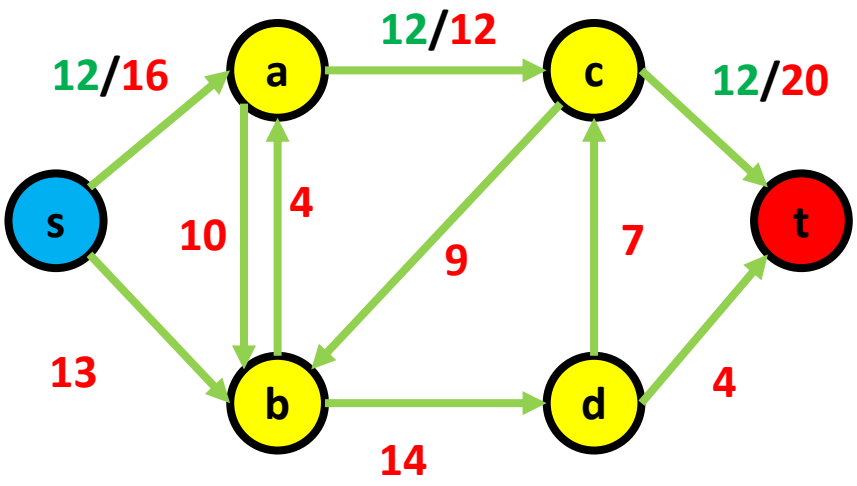


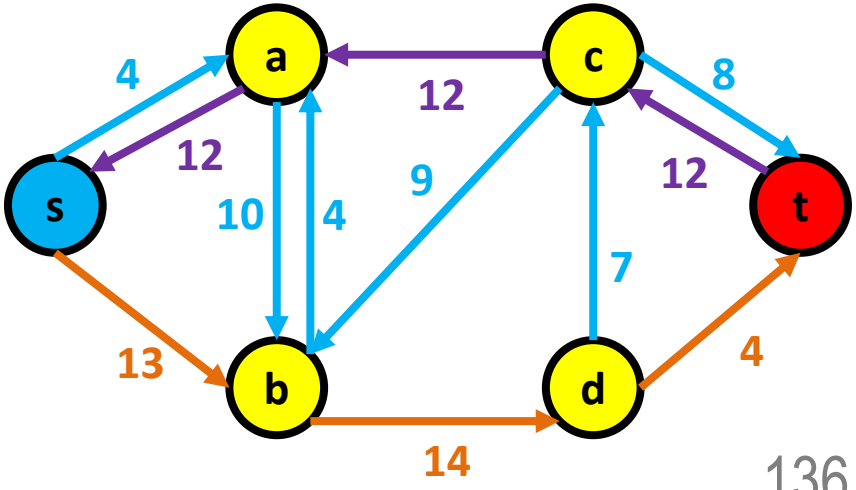
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 12
  - So we just repeat it with path SBDT



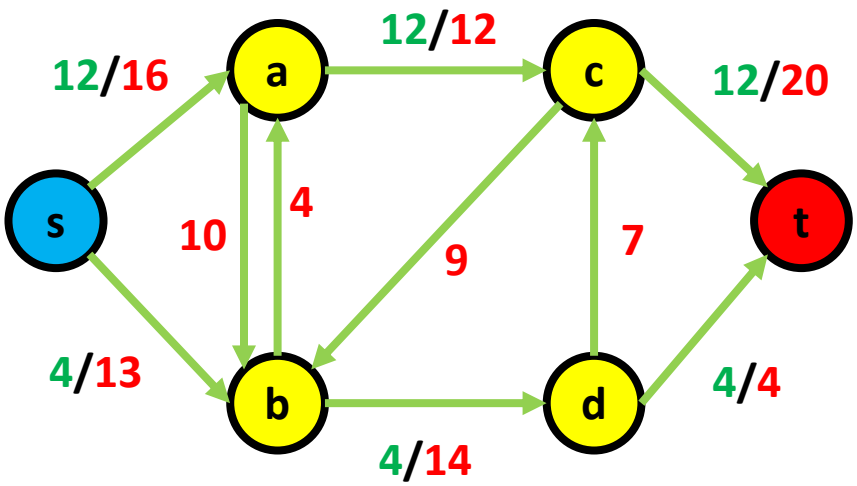


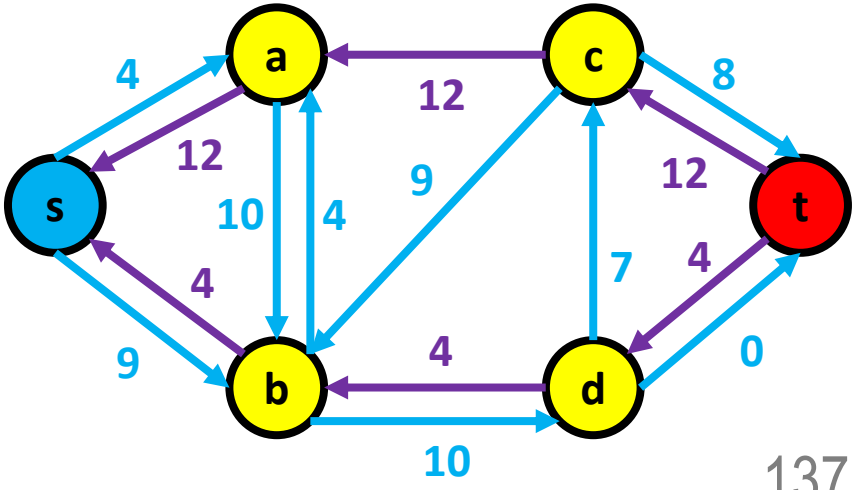
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 12+4 = 16
  - So we just repeat it with path SBDT



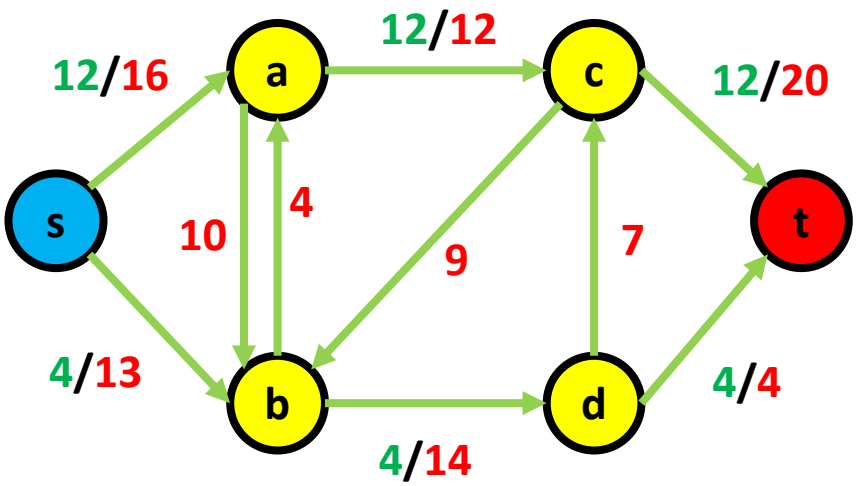


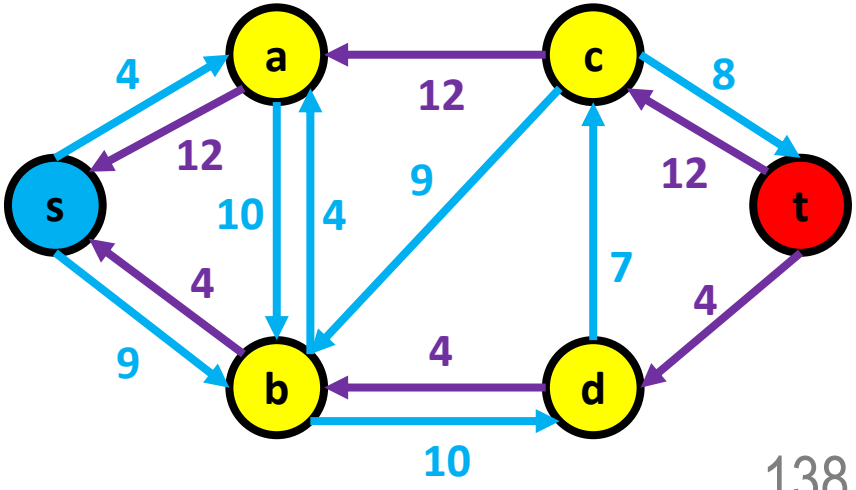
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 16



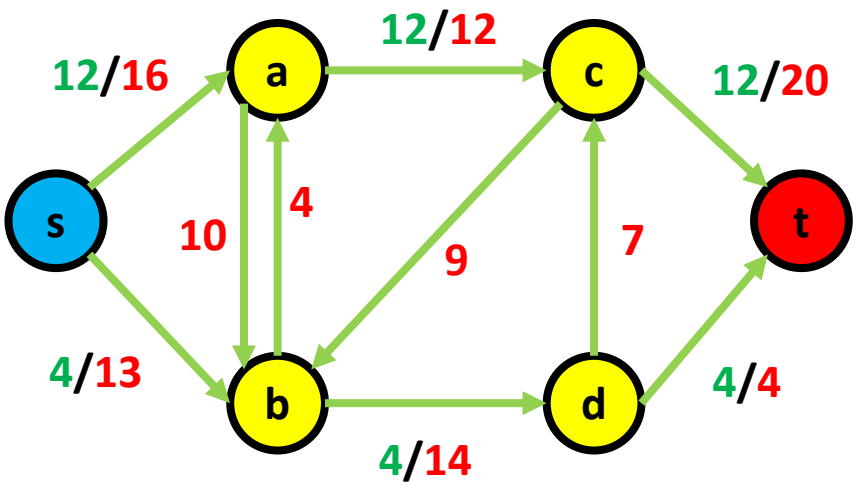


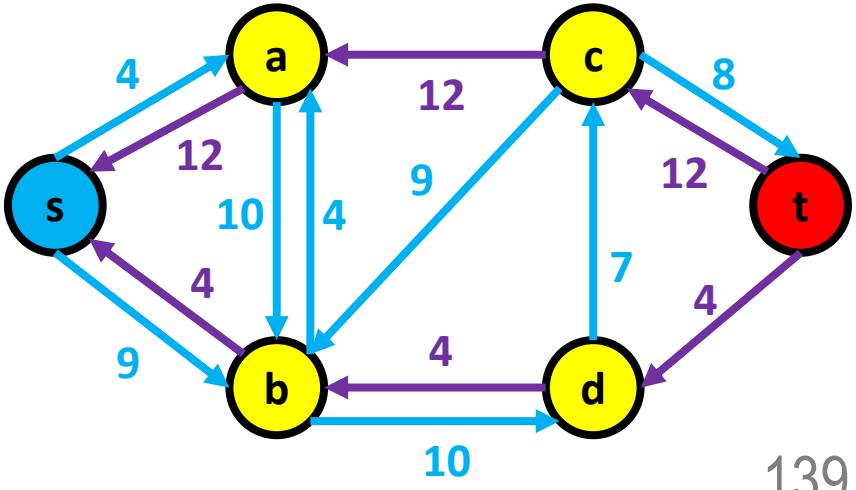
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 16
  - So now we repeat again



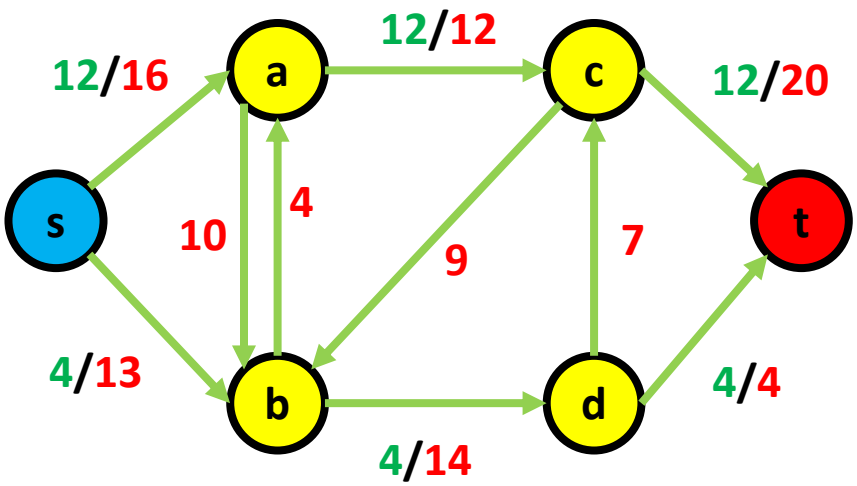


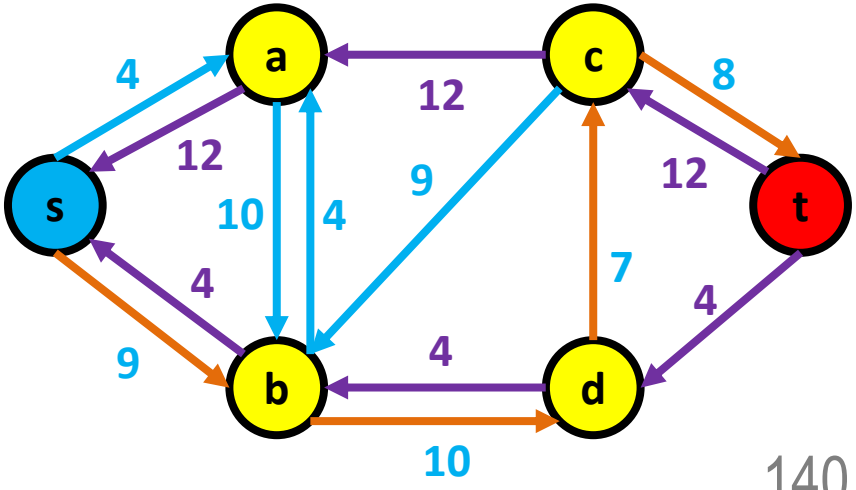
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 16
  - So now we repeat again with path SBDCT



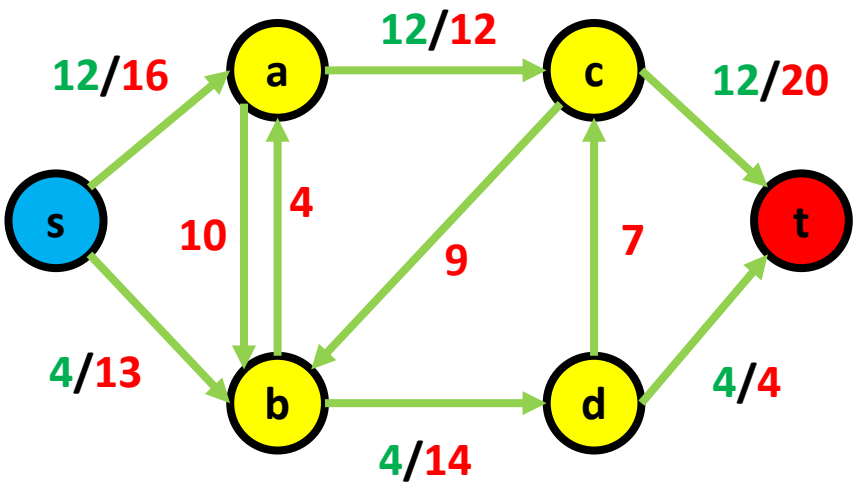


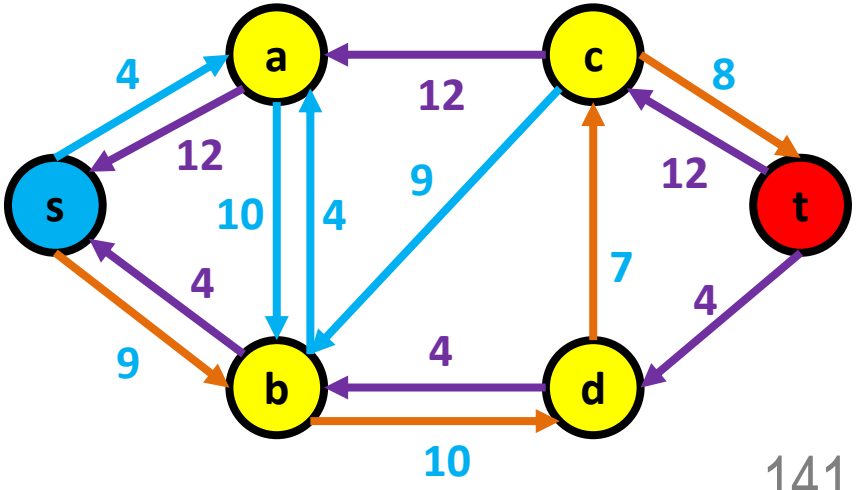
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 16+7
  - So now we repeat again with path SBDCT



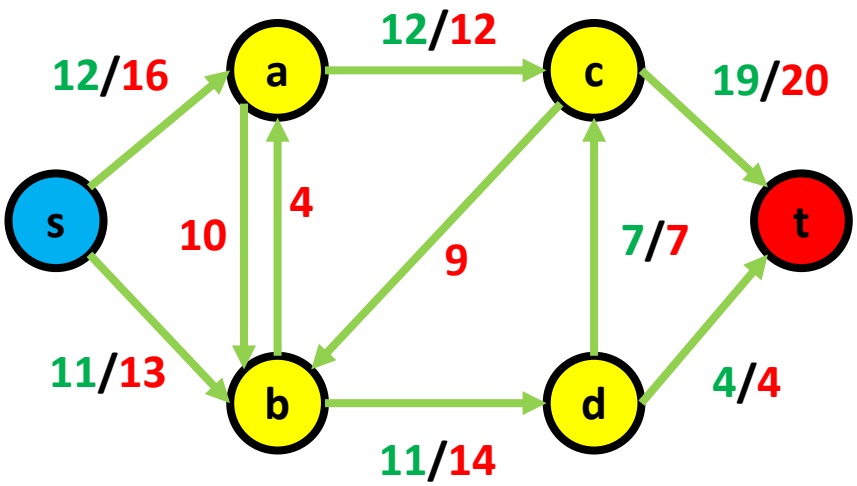


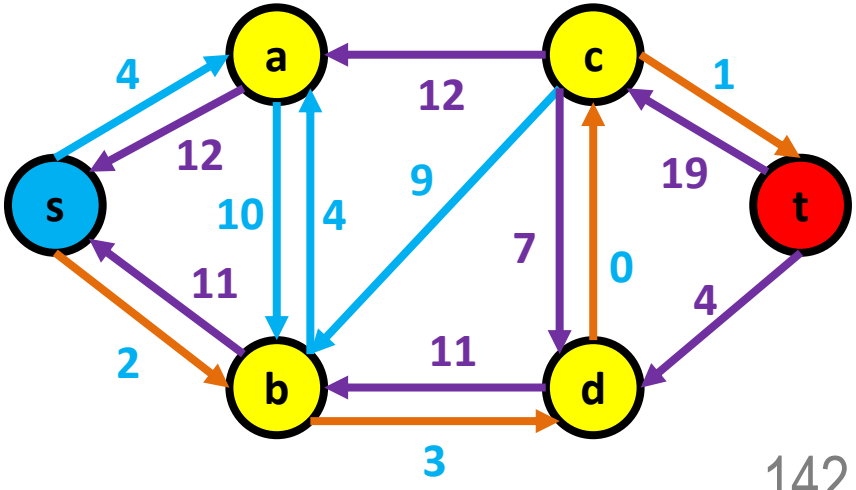
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 16+7
  - So now we repeat again with path SBDCT



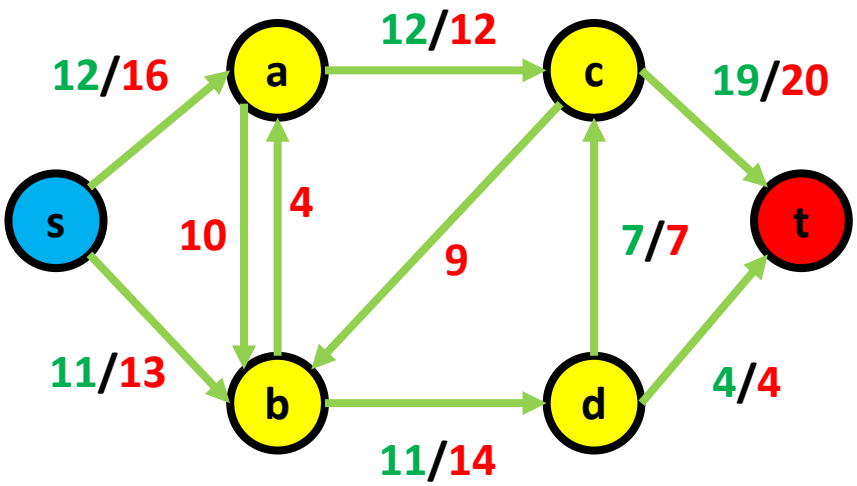


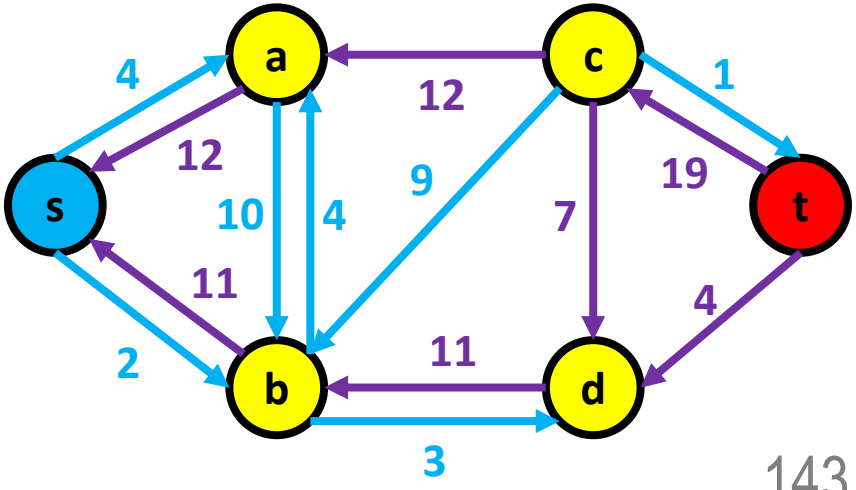
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 23



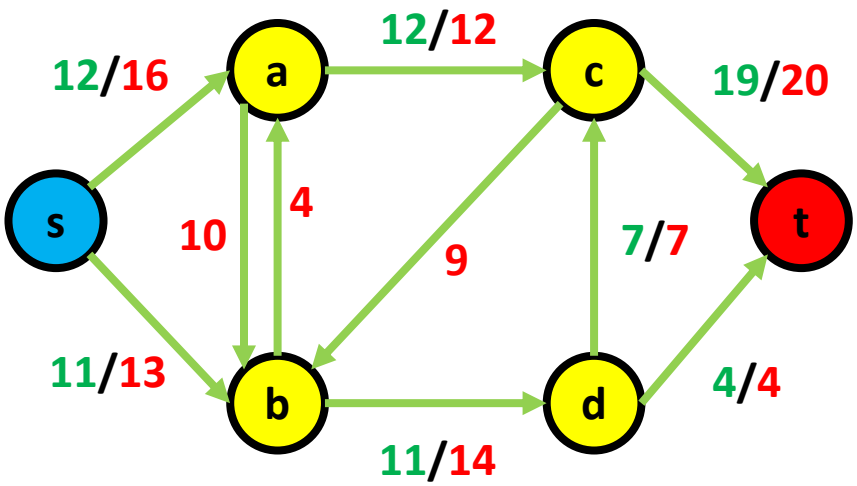


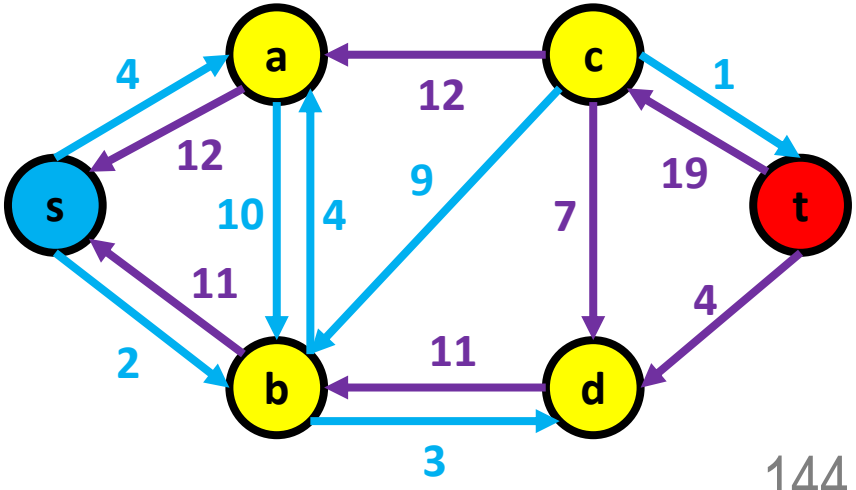
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 23
  - Do we still have a path?



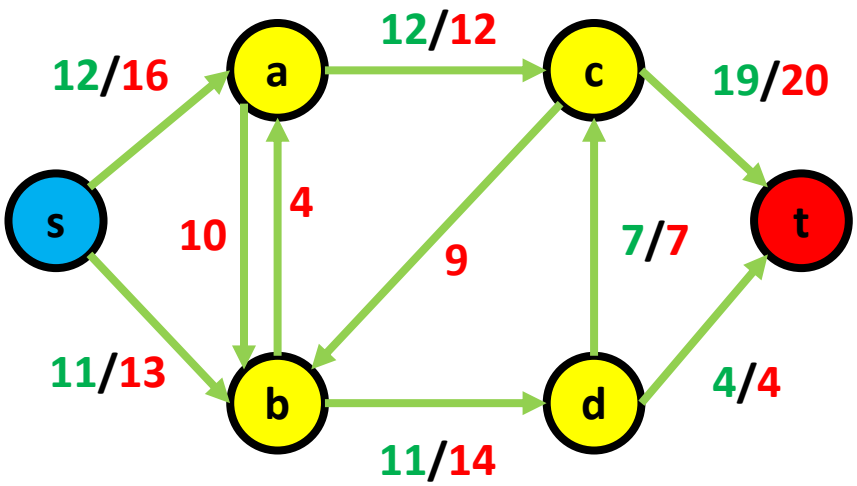


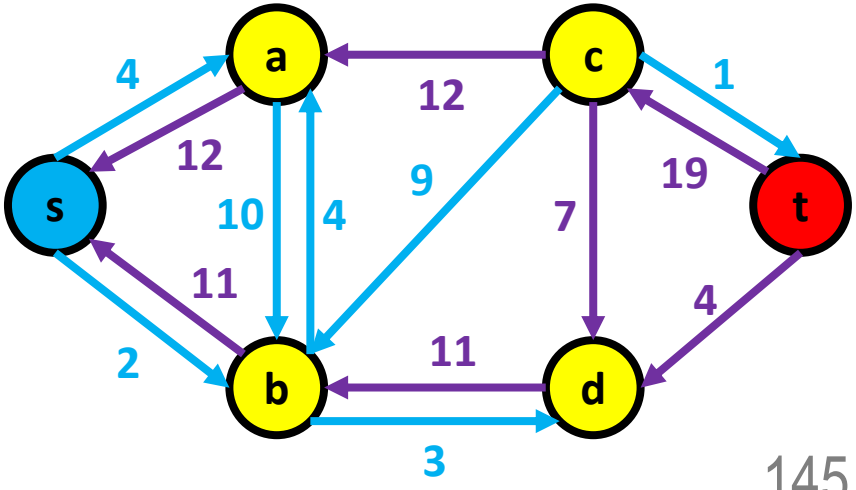
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 23
  - Do we still have a path?
    No more, so we are done!



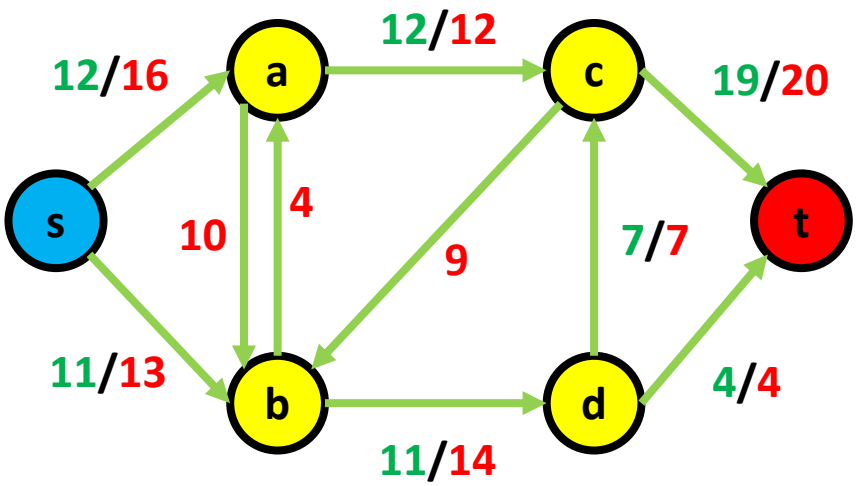


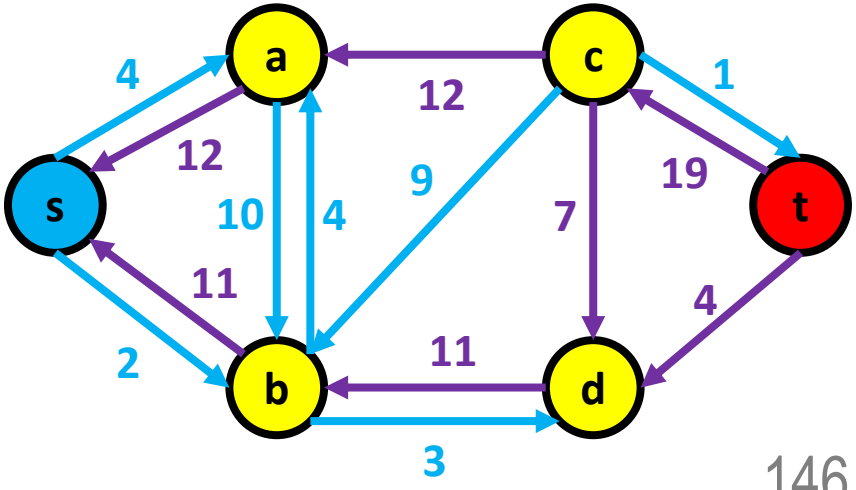
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 23
  - Do we still have a path?
    No more, so we are done!
- Note: The answer in MUA's slide is same value, but different network flow



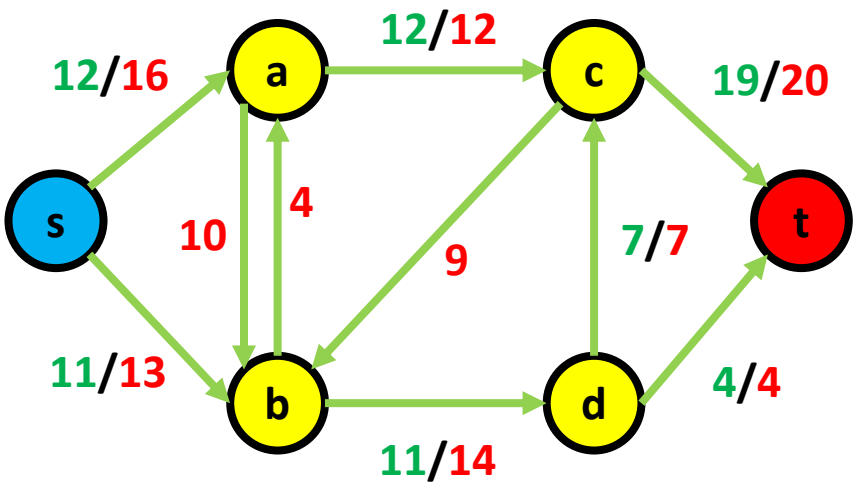


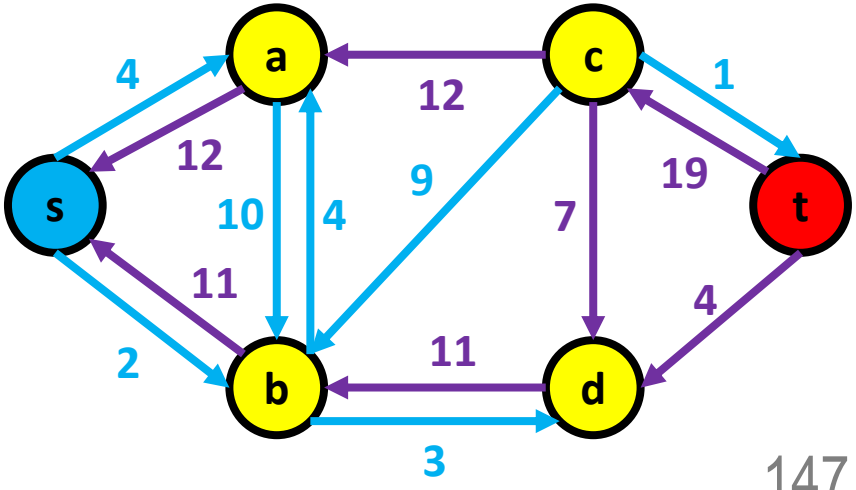
Ford-Fulkerson Method

Finding the maximum flow

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

- Want a trial run?
  - Flow = 23
  - Do we still have a path?
    No more, so we are done!
- Note: The answer in MUA's slide is same value, but different network flow. Thus, answer not unique!





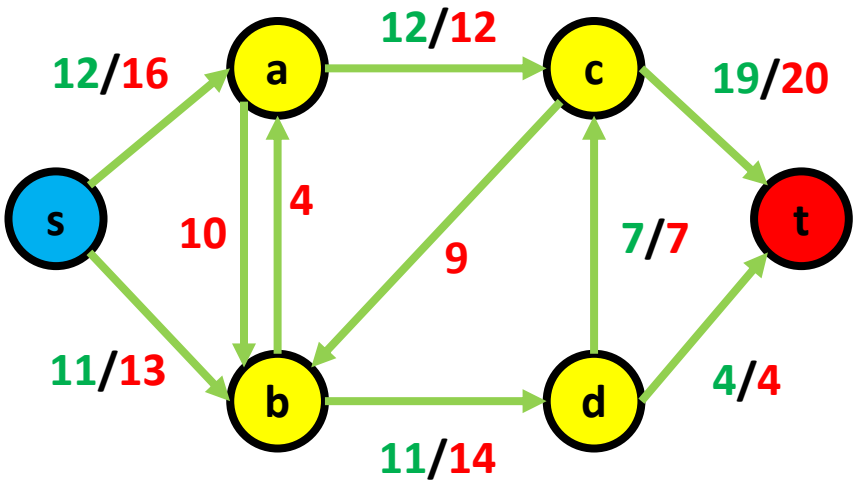
MONASH University

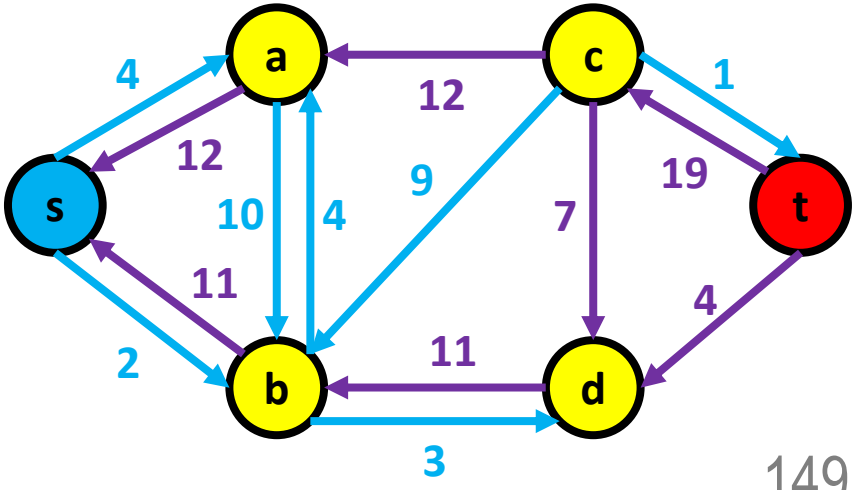
Questions?

Ford-Fulkerson Method

Finding the maximum flow

- Also remember, we maintain all of the properties!

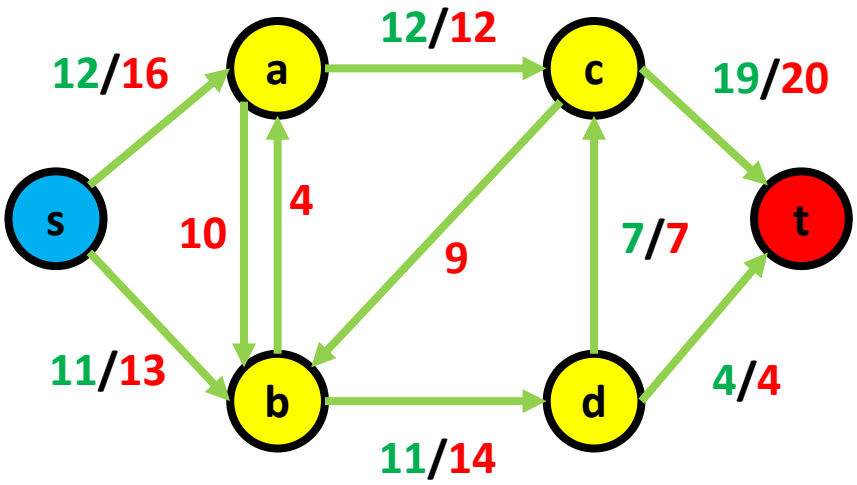


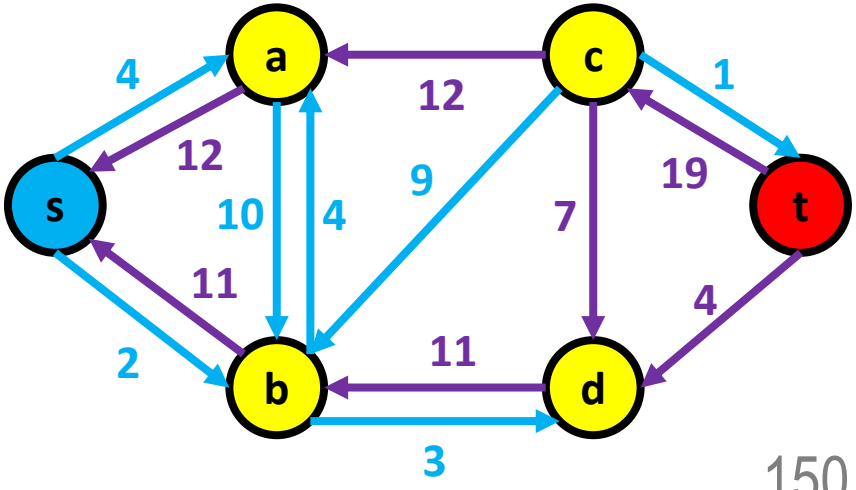


Ford-Fulkerson Method

Finding the maximum flow

- Also remember, we maintain all of the properties!
  - The flow doesn't exceed capacity
  - Incoming to a vertex is the same as outgoing to a vertex



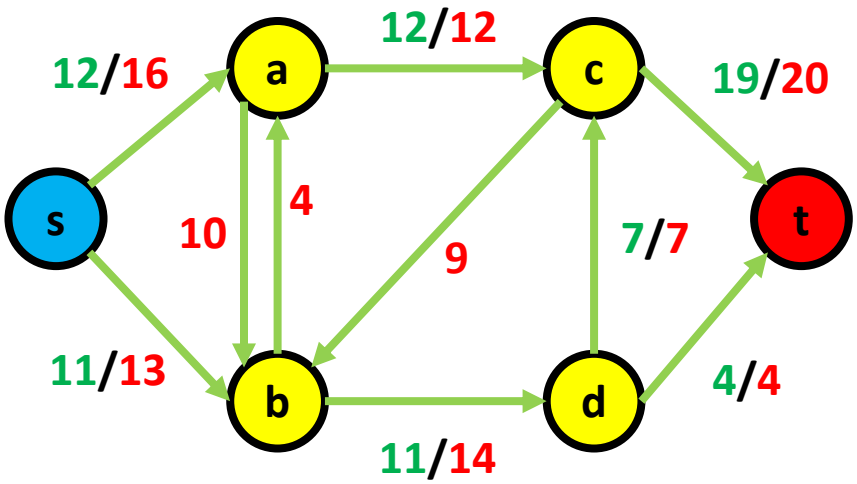


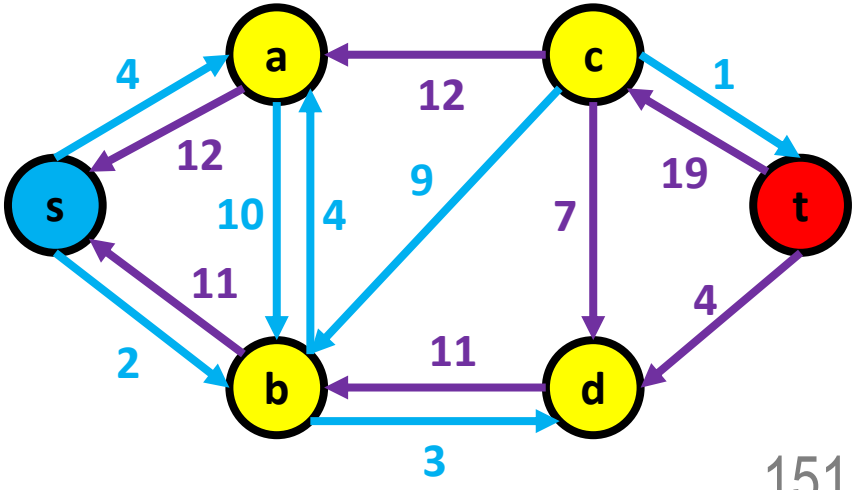
Ford-Fulkerson Method

Finding the maximum flow

MONASH University

- Also remember, we maintain all of the properties!
  - Capacity constraint: The flow doesn't exceed capacity
  - Flow conservation: Incoming to a vertex is the same as outgoing to a vertex





MONASH University

Questions?

Ford-Fulkerson Method

Complexity analysis

MONASH University

- So what is the complexity?

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```

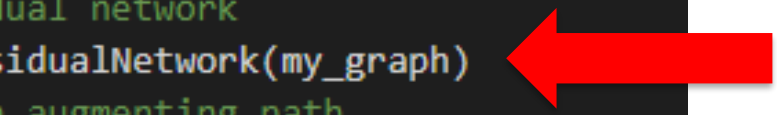
Ford-Fulkerson Method

Complexity analysis

MONASH University

- So what is the complexity?
  - Residual network have a total of 2E edges, thus O(E)

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```



Ford-Fulkerson Method

Complexity analysis

MONASH University

- So what is the complexity?
  - Everything with augmenting path is O(V+E). Why?

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```

Ford-Fulkerson Method

Complexity analysis

MONASH University

- So what is the complexity?
  - Everything with augmenting path is O(V+E). Why? Cause everything is just BFS! Or DFS if you wish to be hipster…

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```

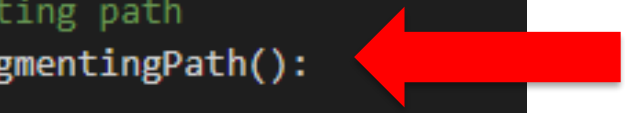
O(V+E)

Ford-Fulkerson Method

Complexity analysis

- So what is the complexity?
  - But how many times do the loop repeat?

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```



Ford-Fulkerson Method

Complexity analysis

MONASH University

- So what is the complexity?
  - But how many times do the loop repeat? Total of O(F) where F is the flow itself.

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```

O(F)

Ford-Fulkerson Method

Complexity analysis

- So what is the complexity?
  - But how many times do the loop repeat? Total of O(F) where F is the flow itself. Why?

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```

O(F)

Ford-Fulkerson Method

Complexity analysis

- So what is the complexity?
  - But how many times do the loop repeat? Total of O(F) where F is the flow itself. Why? Because we increase the flow in each iteration till we can't anymore. Minimum increment is F = F + 1

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```

O(F)

Ford-Fulkerson Method

Complexity analysis

MONASH University

- Total?

```python
def ford_fulkerson(my_graph):
    # initialzie flow
    flow = 0
    # initialize the residual network
    residual_network = ResidualNetwork(my_graph)
    # as long s there is a augmenting path
    while residual_network.has_AugmentingPath():
        # take the path
        path = residual_network.get_AugmentingPath()
        # augment the flow equal to the residual capacity
        flow += path.residual_capacity
        # updating the residual network
        residual_network.augmentFlow(path)
    return flow
```

Ford-Fulkerson Method

Complexity analysis

MONASH University

- Total? O(FV + FE) = O(FE)

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```

Ford-Fulkerson Method

Complexity analysis

- Total? O(FV + FE) = O(FE)
  - This is pseudo-polynomial since F can be very very large…

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```

Ford-Fulkerson Method

Complexity analysis

MONASH University

- Total? O(FV + FE) = O(FE)
  - This is pseudo-polynomial since F can be very very large…
  - But can be proven to be O(VE^2) via Edmonds-Karp (in FIT3155)

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```

Ford-Fulkerson Method

Complexity analysis

- Total? O(FV + FE) = O(FE)
  - This is pseudo-polynomial since F can be very very large…
  - But can be proven to be O(VE^2) via Edmonds-Karp (in FIT3155)

```
1   def ford_fulkerson(my_graph):
2       # initialzie flow
3       flow = 0
4       # initialize the residual network
5       residual_network = ResidualNetwork(my_graph)
6       # as long s there is a augmenting path
7       while residual_network.has_AugmentingPath():
8           # take the path
9           path = residual_network.get_AugmentingPath()
10          # augment the flow equal to the residual capacity
11          flow += path.residual_capacity
12          # updating the residual network
13          residual_network.augmentFlow(path)
14      return flow
```

MONASH University

Questions?

MONASH University

Break?

Ford-Fulkerson Method

Proof of correctness

- But does it work???

Ford-Fulkerson Method

Proof of correctness

- Assume every capacity is integer

Ford-Fulkerson Method

Proof of correctness

- Assume every capacity is integer
- Flow always increase by 1 at each iteration and we know flow is finite

Ford-Fulkerson Method

Proof of correctness

- Assume every capacity is integer
- Flow always increase by 1 at each iteration and we know flow is finite
- Therefore the algorithm do terminate!

Ford-Fulkerson Method

Proof of correctness

- Assume every capacity is integer
- Flow always increase by 1 at each iteration and we know flow is finite
- Therefore the algorithm do terminate!

- But does it terminate with the max flow?

Ford-Fulkerson Method

Proof of correctness

MONASH University

- Assume every capacity is integer
- Flow always increase by 1 at each iteration and we know flow is finite
- Therefore the algorithm do terminate!

- But does it terminate with the max flow?
  - That is why we need the min-cut max-flow theorem to finish our proof

MONASH University

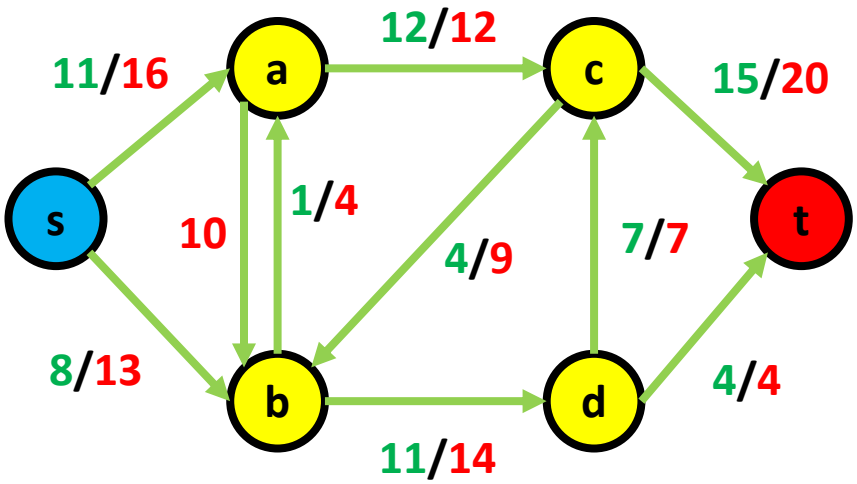
Questions?

MONASH University

- What is a cut?

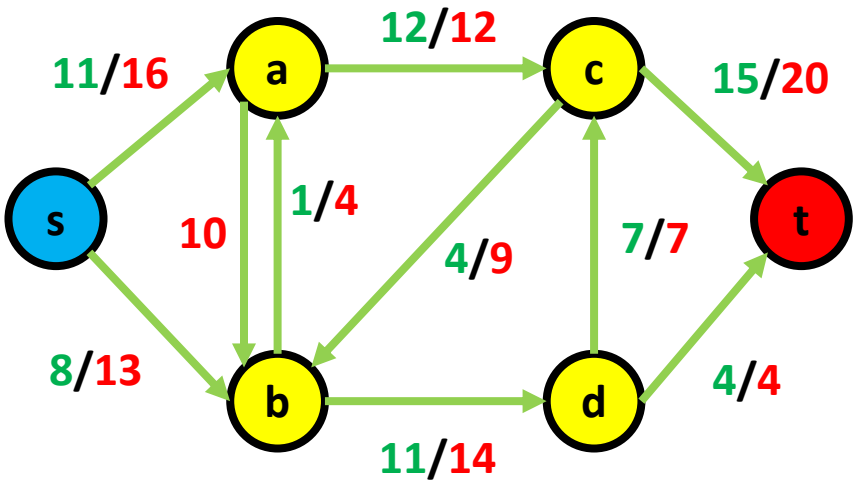
MONASH University

- What is a cut?



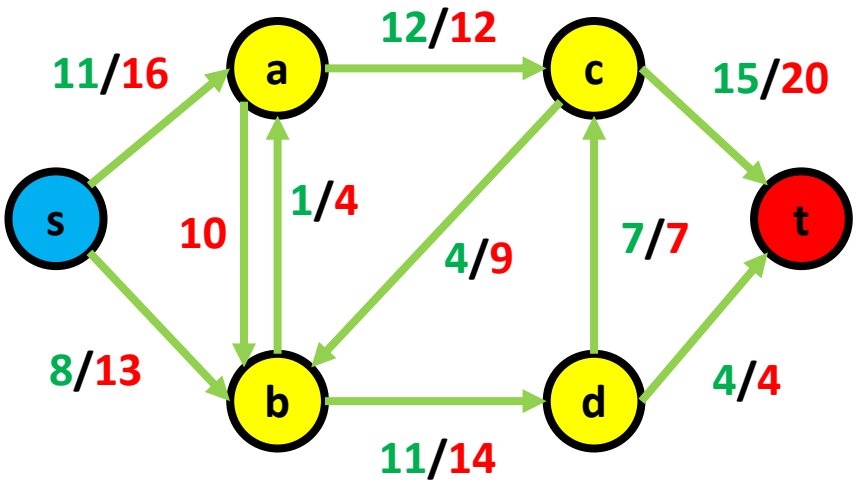
Cut
Flow and Capacity

- What is a cut?
  - A cut (S,T)



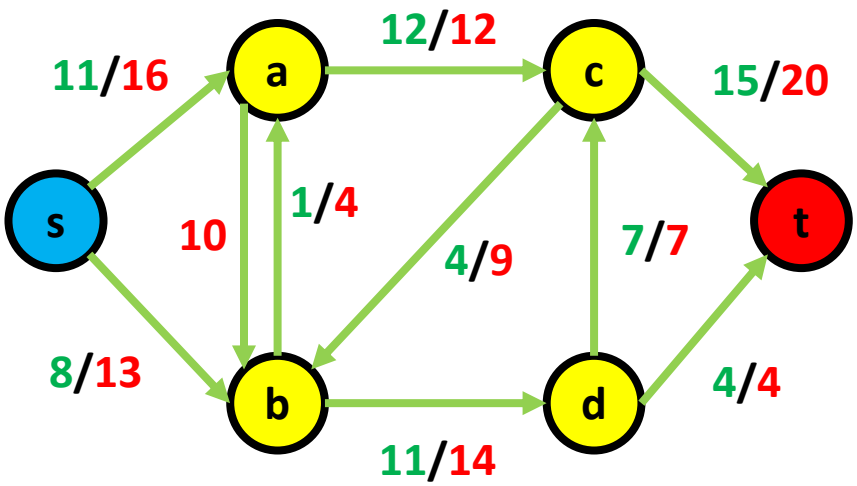
Cut
Flow and Capacity

- What is a cut?
  - A cut (S,T)
    - S must contain vertex s
    - T must contain vertex t



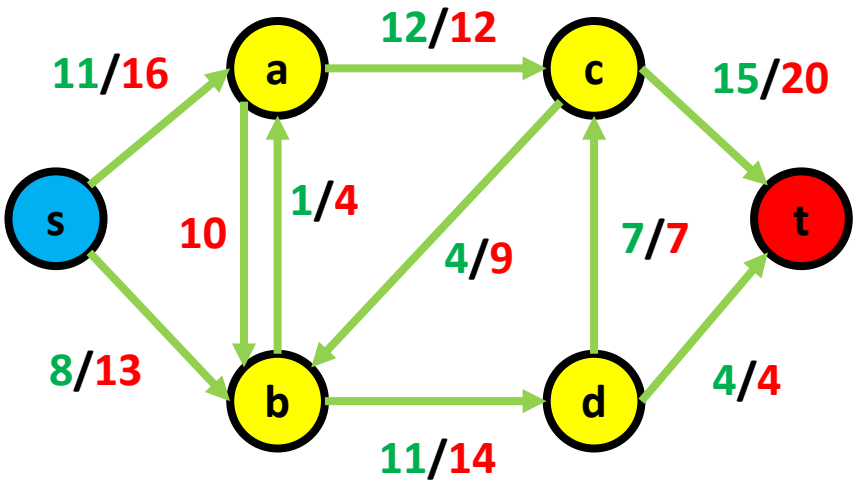
Cut
Flow and Capacity

- What is a cut?
  - A cut (S,T)
    - S must contain vertex s
    - T must contain vertex t



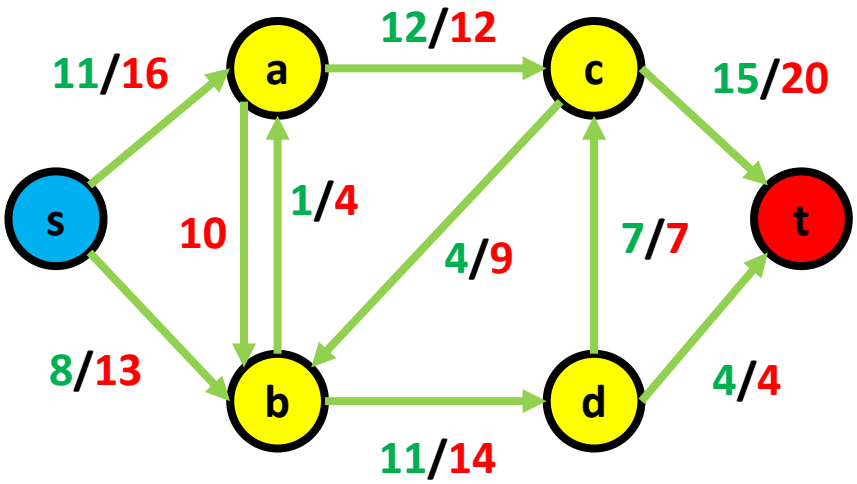
Cut
Flow and Capacity

- What is a cut?
  - A cut (S,T)
    - S must contain vertex s
    - T must contain vertex t
  - Thus we can do the following…



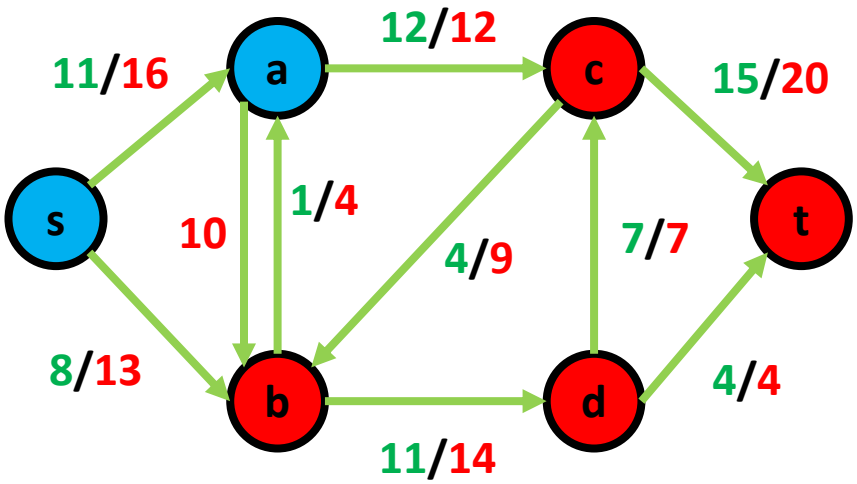
MONASH University

- What is a cut?
  - A cut (S,T)
    - S must contain vertex s. S = (s,a)
    - T must contain vertex t. T = (b,c,d,t)
  - Thus we can do the following…



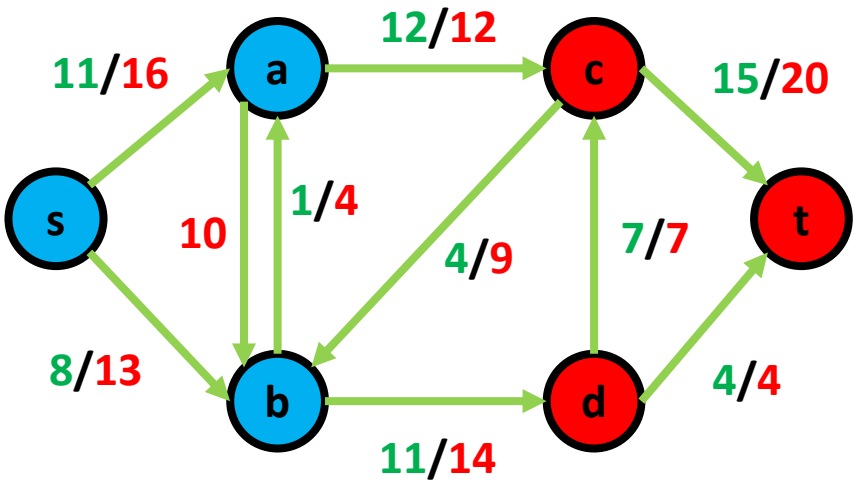
MONASH University

- What is a cut?
  - A cut (S,T)
    - S must contain vertex s. S = (s,a)
    - T must contain vertex t. T = (b,c,d,t)
  - Thus we can do the following…



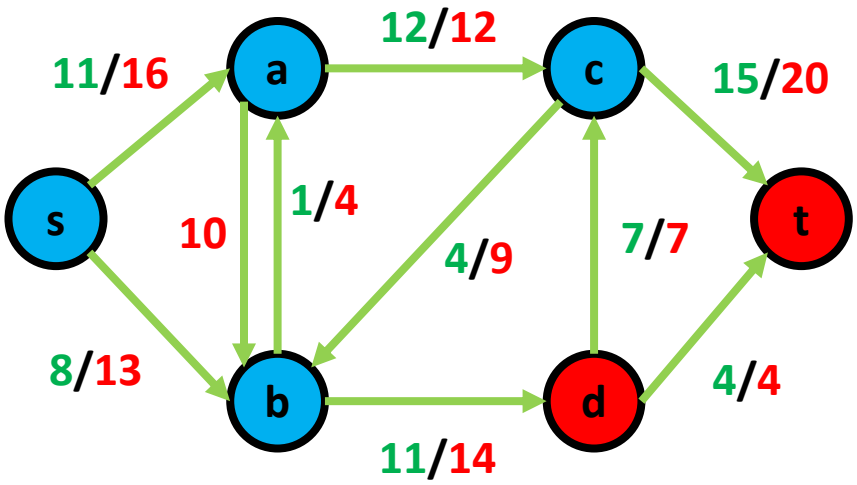
MONASH University

- What is a cut?
  - A cut (S,T)
    - S must contain vertex s. S = (s,a,b)
    - T must contain vertex t. T = (c,d,t)
  - Thus we can do the following…



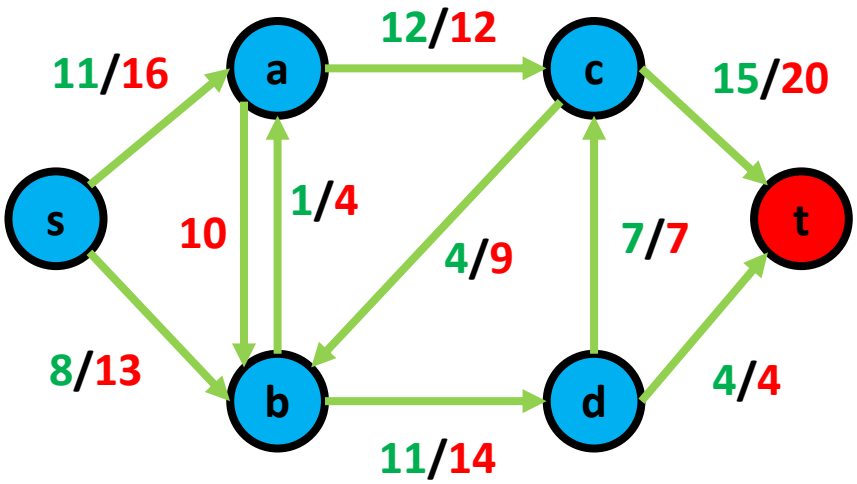
MONASH University

- What is a cut?
  - A cut (S,T)
    - S must contain vertex s. S = (s,a,b,c)
    - T must contain vertex t. T = (d,t)
  - Thus we can do the following…



MONASH University

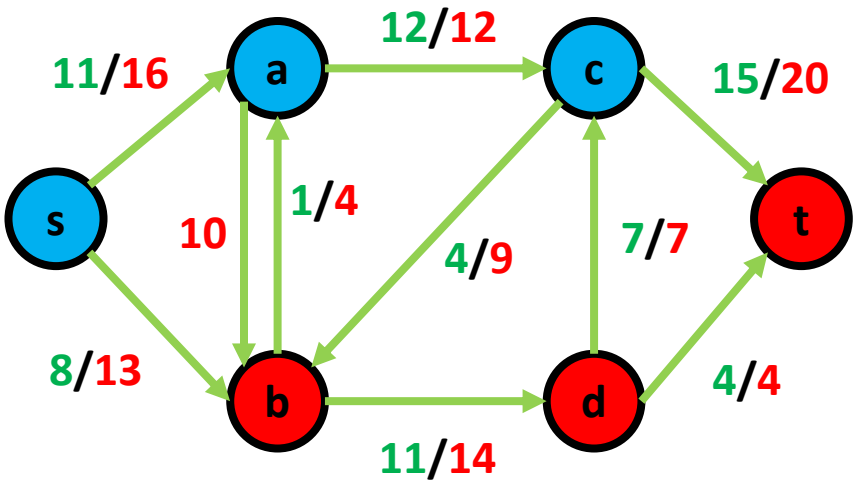
- What is a cut?
  - A cut (S,T)
    - S must contain vertex s. S = (s,a,b,c,d)
    - T must contain vertex t. T = (t)
  - Thus we can do the following…



Cut
Flow and Capacity

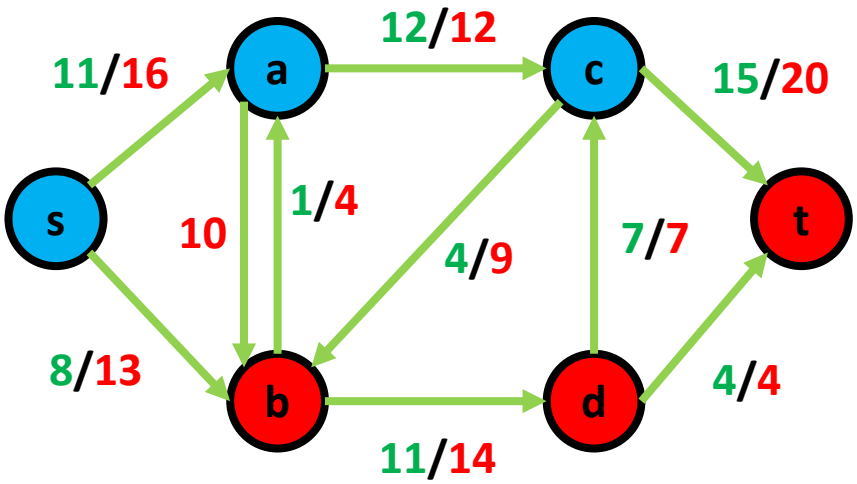
MONASH University

- What is a cut?
  - A cut (S,T)
    - S must contain vertex s. S = (s,a,c)
    - T must contain vertex t. T = (b,d,t)
  - Thus we can do the following…
  - And more!!!



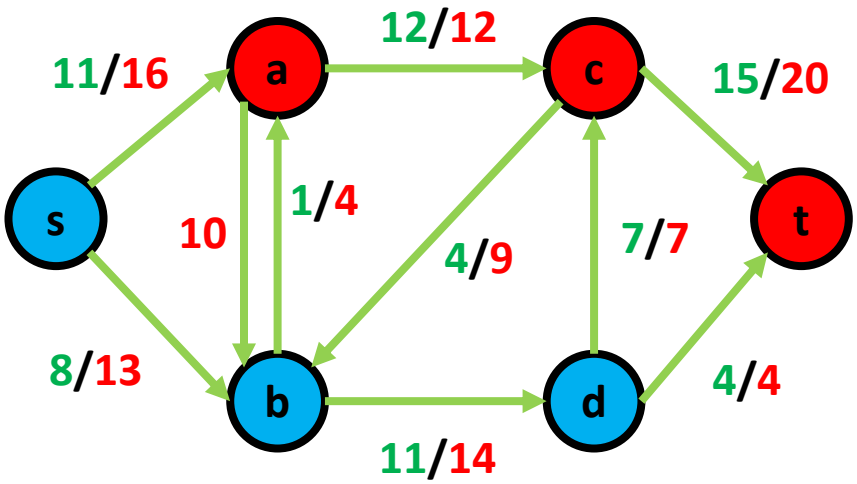
Cut
Flow and Capacity

- What is a cut?
  - A cut (S,T)
    - S must contain vertex s. S = (s,a,c)
    - T must contain vertex t. T = (b,d,t)
  - Thus we can do the following…
  - And more!!! Because the S and T are still connected with path from s and path to t



Cut
Flow and Capacity

- What is a cut?
  - A cut (S,T)
    - S must contain vertex s. S = (s,a,c)
    - T must contain vertex t. T = (b,d,t)
  - Thus we can do the following…
  - And more!!! Because the S and T are still connected with path from s and path to t

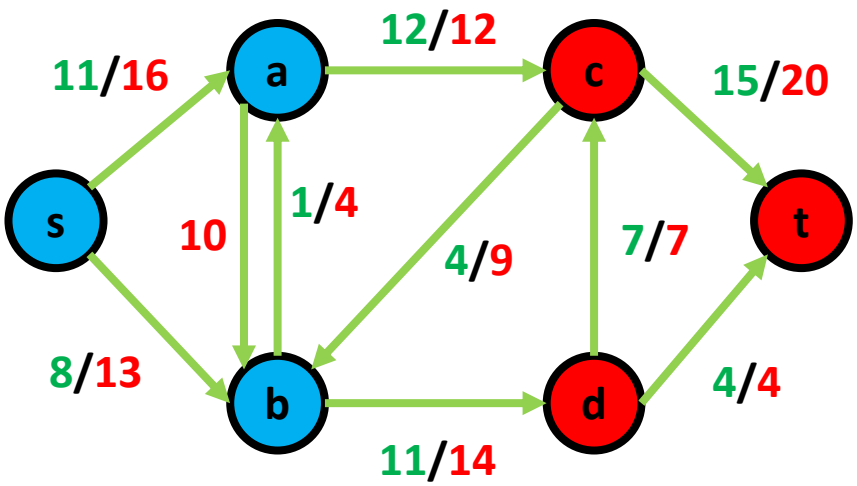


MONASH University

Questions?

Cut
Flow and Capacity

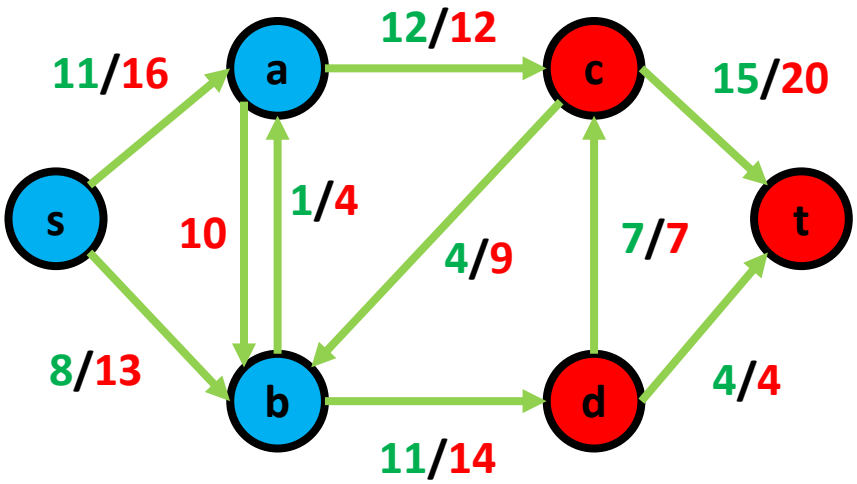
- Let us use this example
  - A cut (S,T)
    - S must contain vertex s. S = (s,a,b)
    - T must contain vertex t. T = (c,d,t)



Cut
Flow and Capacity

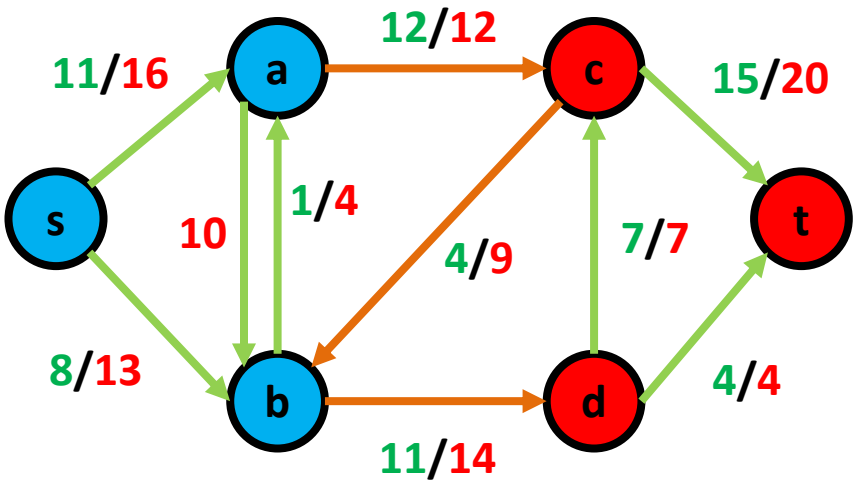
MONASH University

- Let us use this example
  - A cut (S,T)
    - S must contain vertex s. S = (s,a,b)
    - T must contain vertex t. T = (c,d,t)
  - We have edges crossing the cut



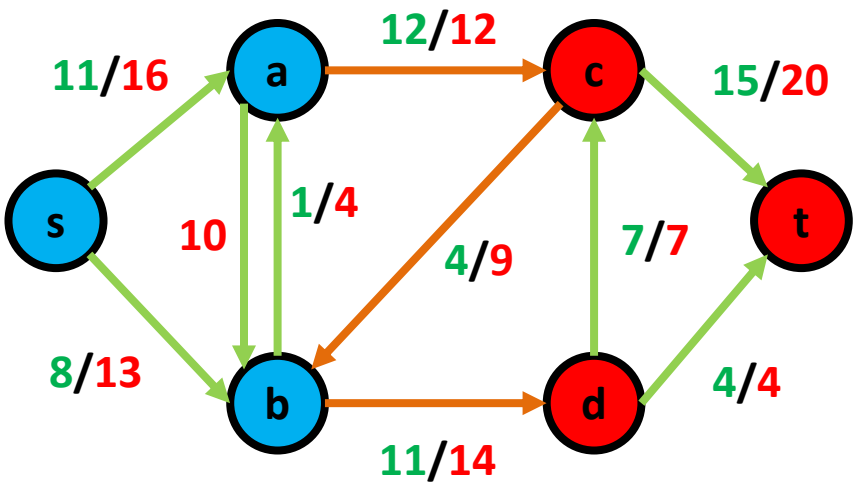
Cut
Flow and Capacity

- Let us use this example
  - A cut (S,T)
    - S must contain vertex s. S = (s,a,b)
    - T must contain vertex t. T = (c,d,t)
  - We have edges crossing the cut



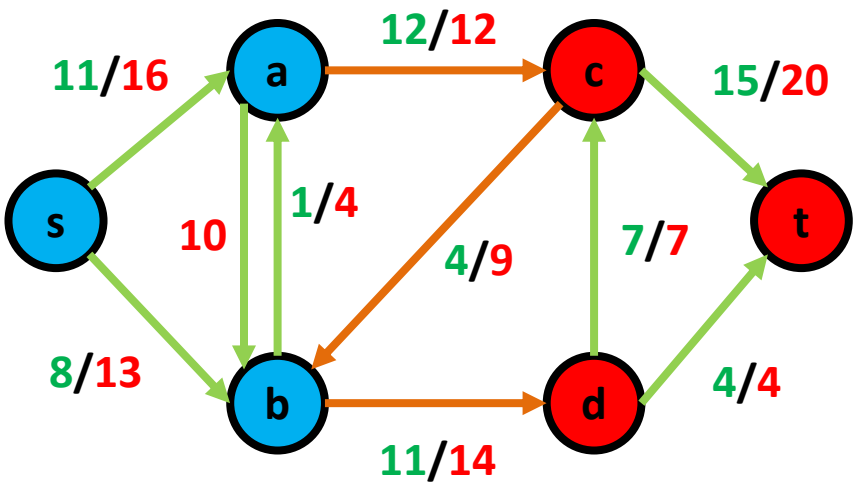
Cut
Flow and Capacity

- Let us use this example
  - A cut (S,T)
    - S must contain vertex s. S = (s,a,b)
    - T must contain vertex t. T = (c,d,t)
  - We have edges crossing the cut
  - Capacity of a cut (S,T) =
  - Flow of a cut (S,T) =

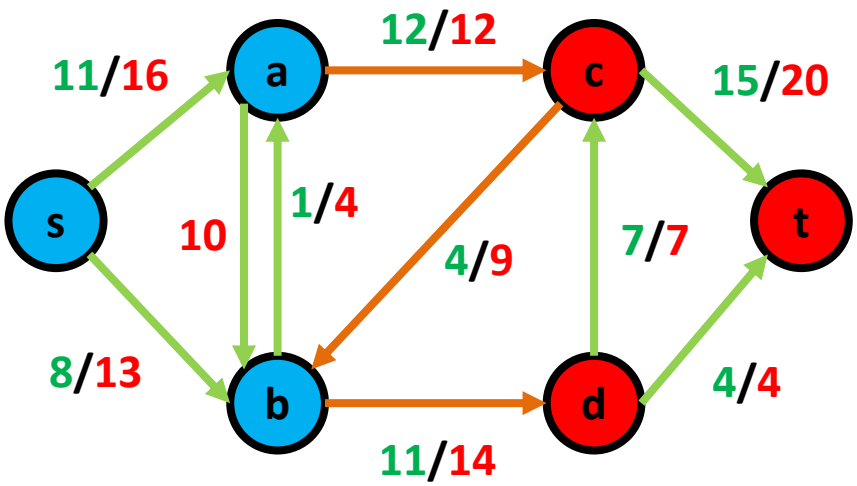


Cut
Flow and Capacity

- Let us use this example
  - A cut (S,T)
    - S must contain vertex s. S = (s,a,b)
    - T must contain vertex t. T = (c,d,t)
  - We have edges crossing the cut
  - Capacity of a cut (S,T) = capacity of outgoing edges
  - Flow of a cut (S,T) =

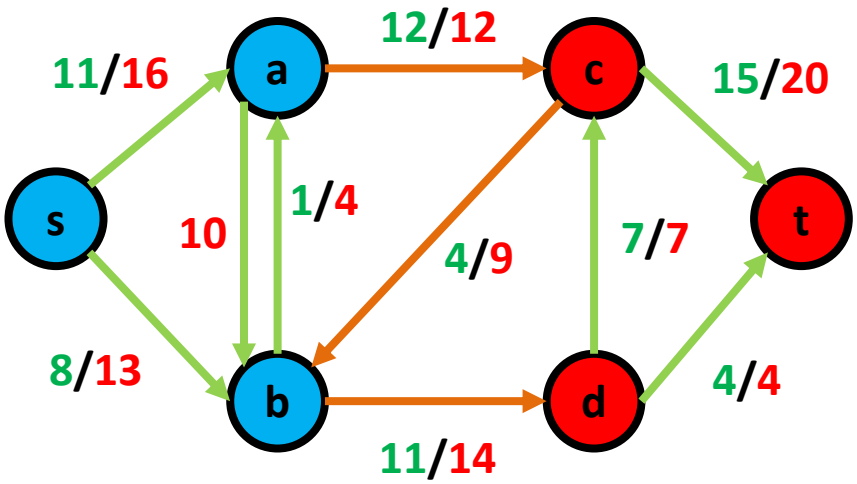


- Let us use this example
  - A cut (S,T)
    - S must contain vertex s. S = (s,a,b)
    - T must contain vertex t. T = (c,d,t)
  - We have edges crossing the cut
  - Capacity of a cut (S,T) = capacity of outgoing edges = 12+14 = 26
  - Flow of a cut (S,T) =



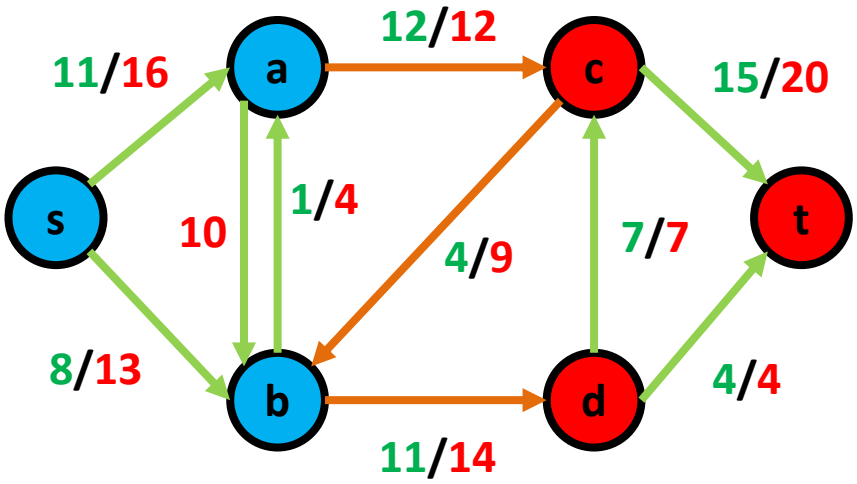
MONASH University

- Let us use this example
  - A cut (S,T)
    - S must contain vertex s. S = (s,a,b)
    - T must contain vertex t. T = (c,d,t)
  - We have edges crossing the cut
  - Capacity of a cut (S,T) = capacity of outgoing edges = 12+14 = 26
  - Flow of a cut (S,T) = flow of outgoing edges – flow of incoming edges



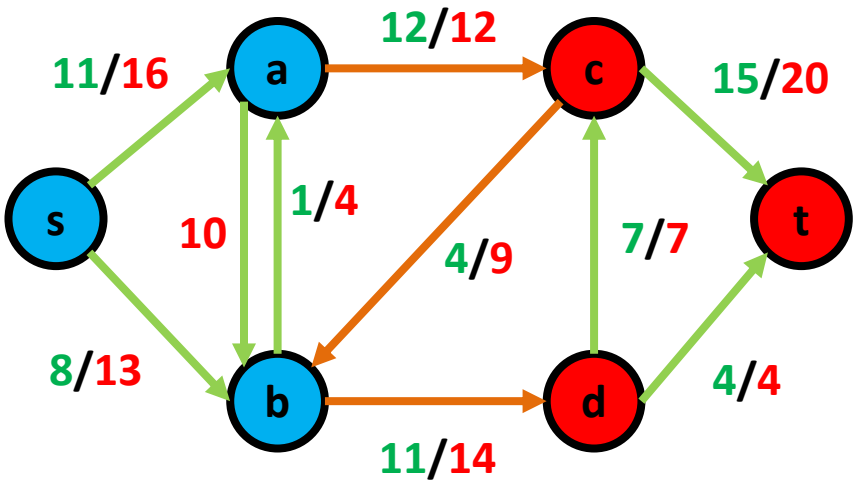
Cut
Flow and Capacity

- Let us use this example
  - A cut (S,T)
  - We have edges crossing the cut
  - Capacity of a cut (S,T) = capacity of outgoing edges = 12+14 = 26
  - Flow of a cut (S,T) = flow of outgoing edges – flow of incoming edges



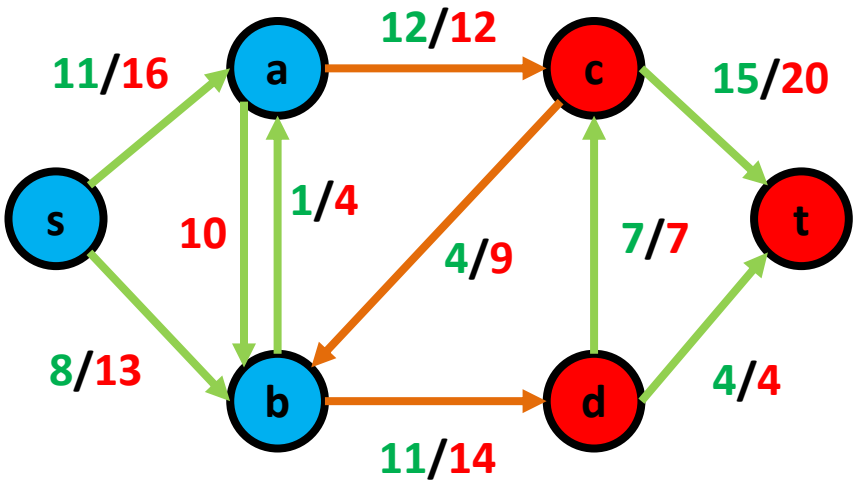
Cut
Flow and Capacity

- Let us use this example
  - A cut (S,T)
  - We have edges crossing the cut
  - Capacity of a cut (S,T) = capacity of outgoing edges = 12+14 = 26
  - Flow of a cut (S,T) = flow of outgoing edges – flow of incoming edges = 12 + 11 – 4



MONASH University

- Let us use this example
  - A cut (S,T)
  - We have edges crossing the cut
  - Capacity of a cut (S,T) = capacity of outgoing edges = 12+14 = 26
  - Flow of a cut (S,T) = flow of outgoing edges – flow of incoming edges = 12 + 11 – 4 = 19

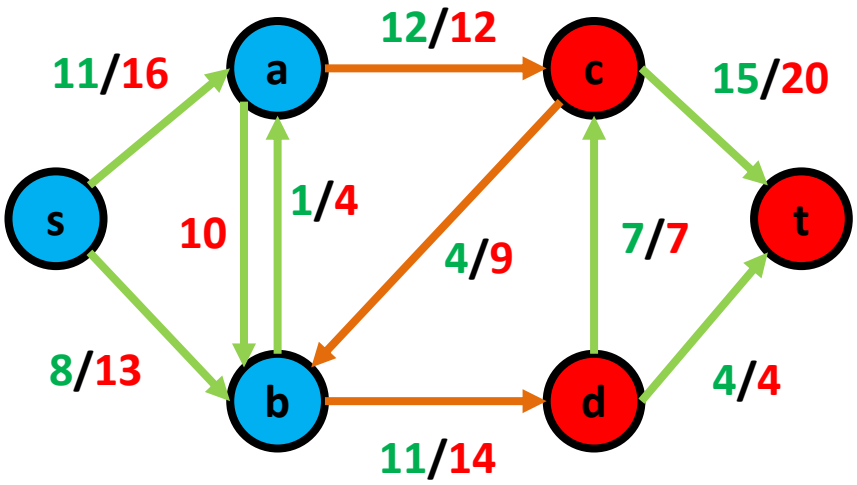


MONASH University

Questions?

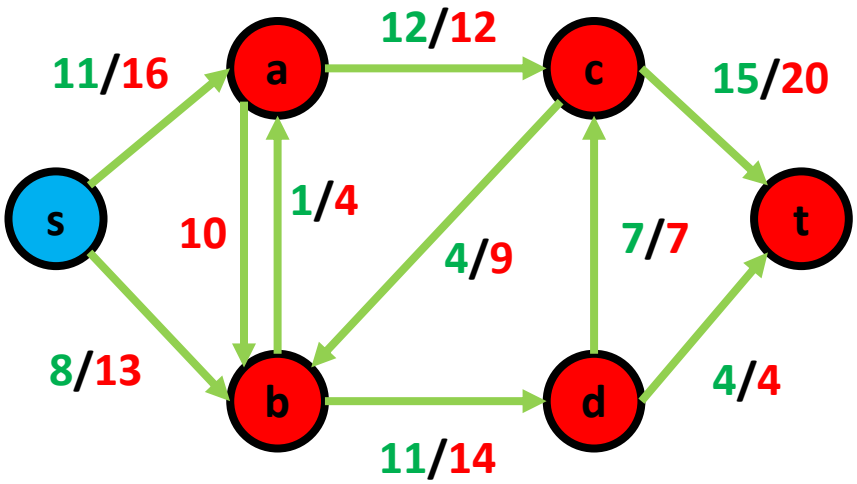
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19



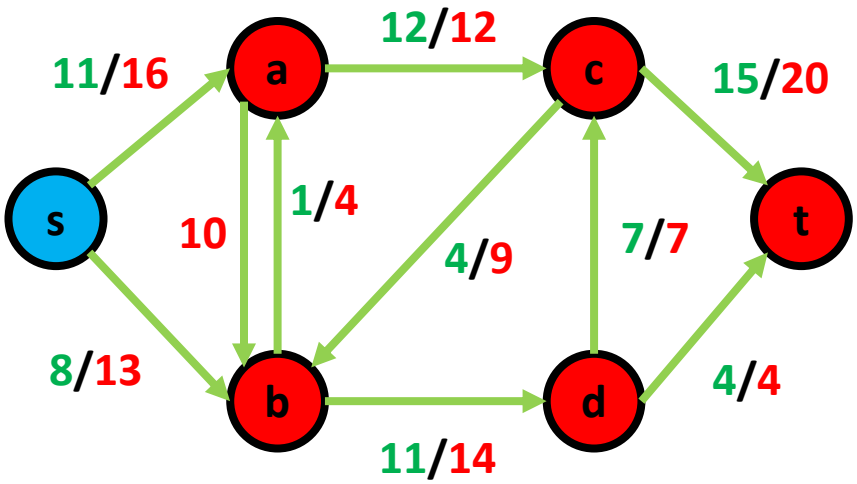
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting?



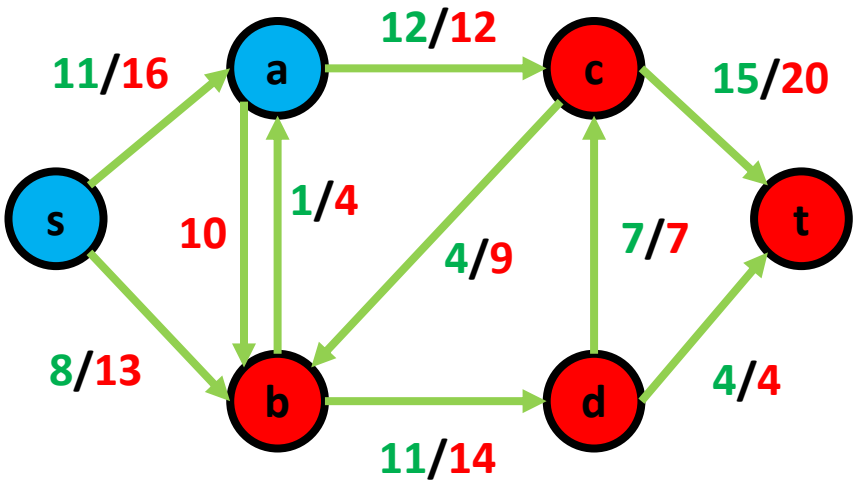
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting?
    - Capacity = 29
    - Flow = 19



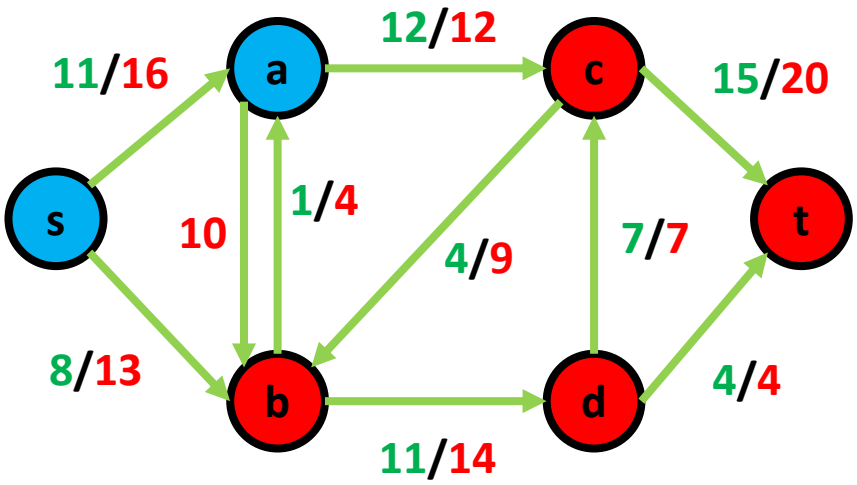
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting?
    - Capacity = ?
    - Flow = ?



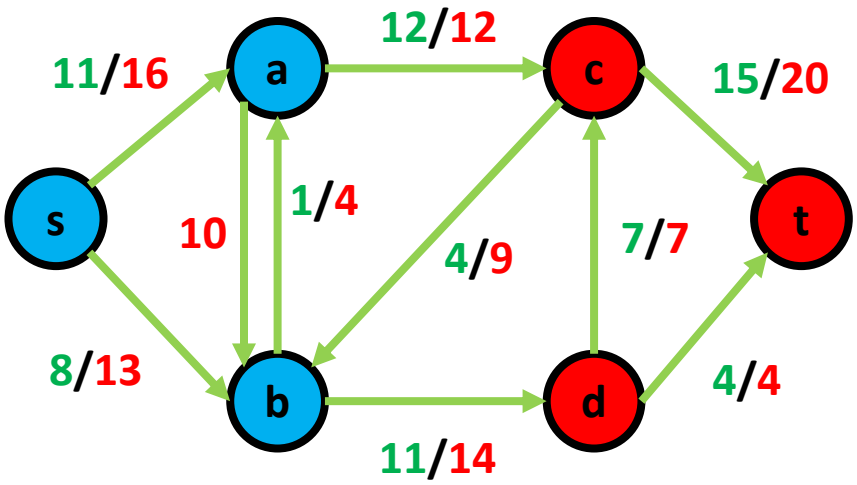
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting?
    - Capacity = 35
    - Flow = 19



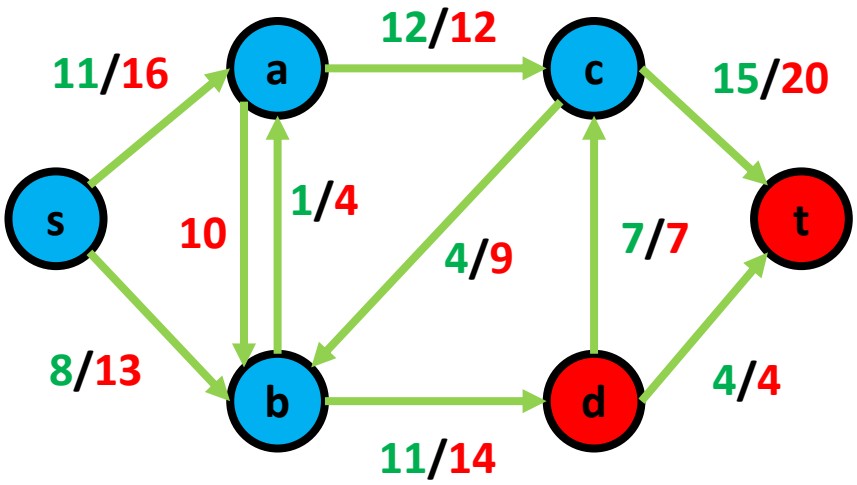
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting?
    - Capacity = 26
    - Flow = 19



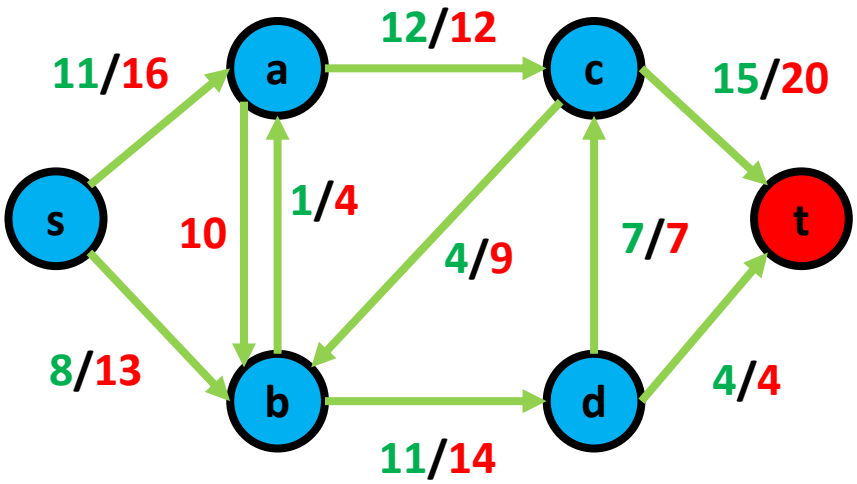
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting?
    - Capacity = 34
    - Flow = 19



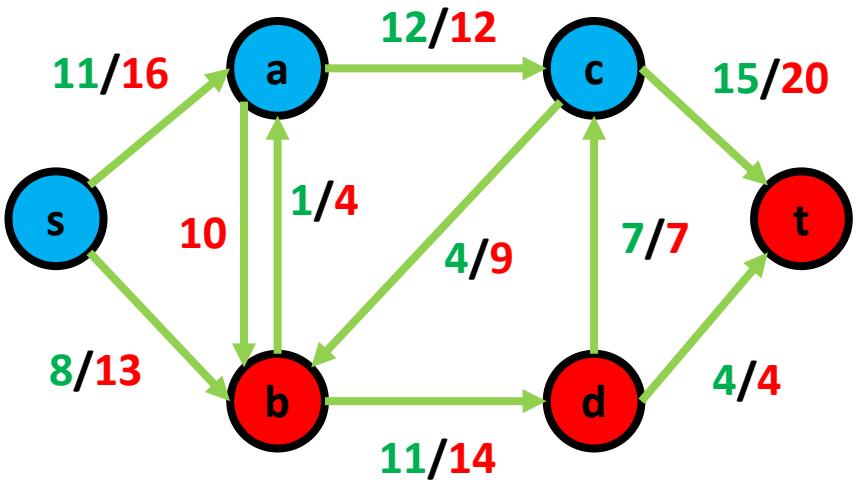
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting?
    - Capacity = 24
    - Flow = 19



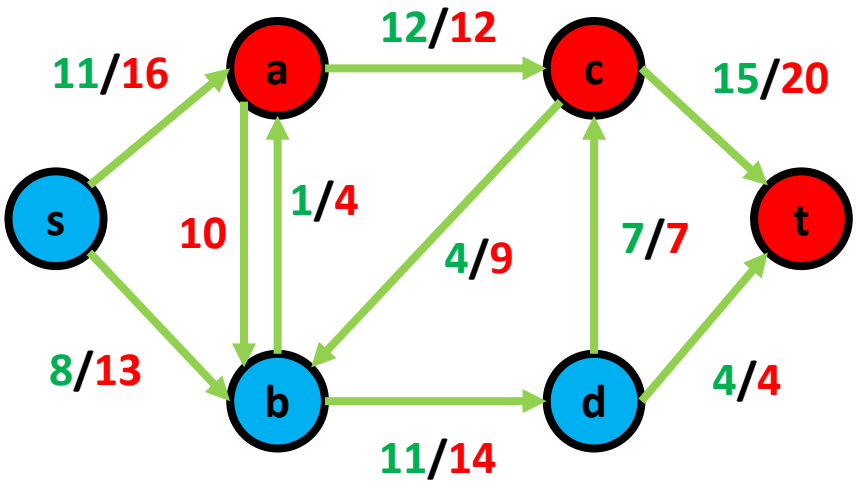
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting?
    - Capacity = 52
    - Flow = 19



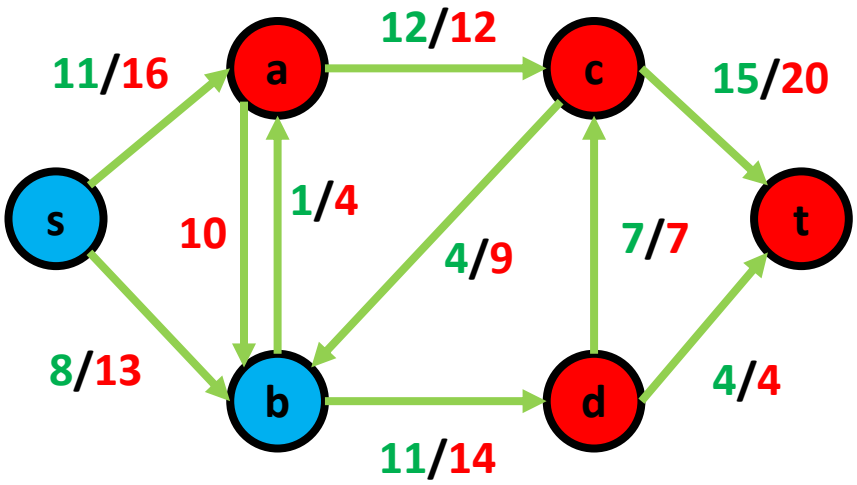
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting?
    - Capacity = 31
    - Flow = 19



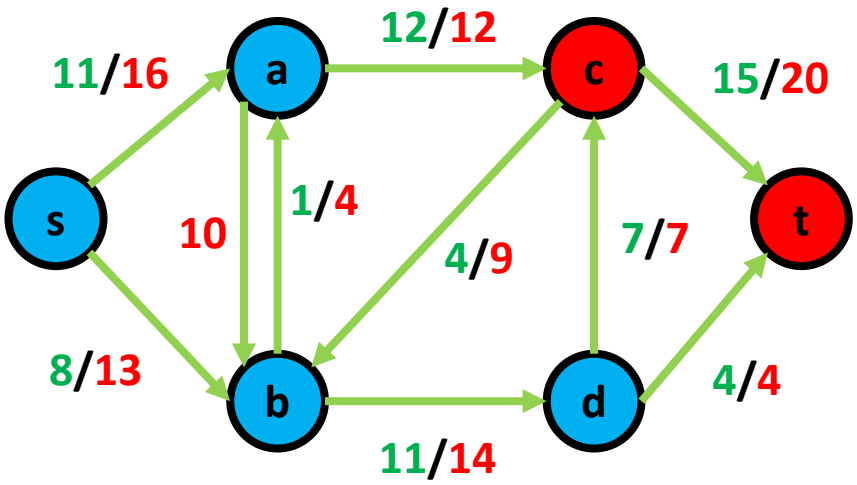
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting?
    - Capacity = 34
    - Flow = 19



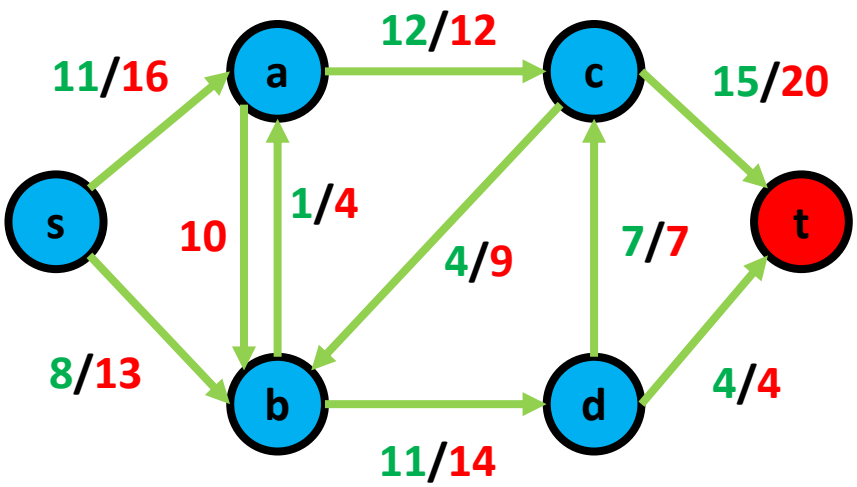
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting?
    - Capacity = 23
    - Flow = 19



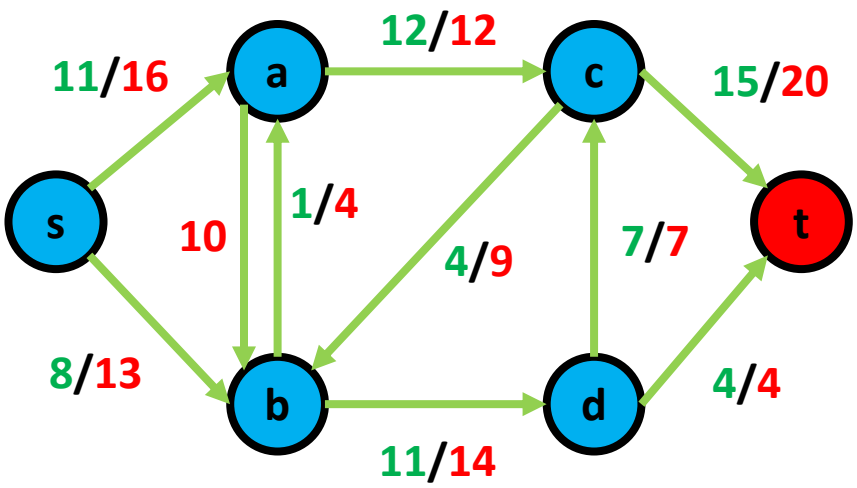
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting?
    - Capacity differs
    - Flow the same



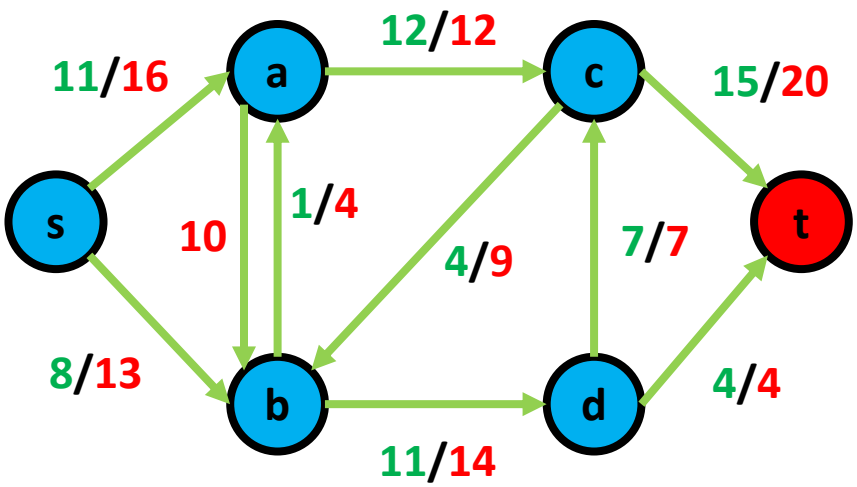
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting?
    - Capacity differs
    - Flow the same
    - But we know flow <= capacity



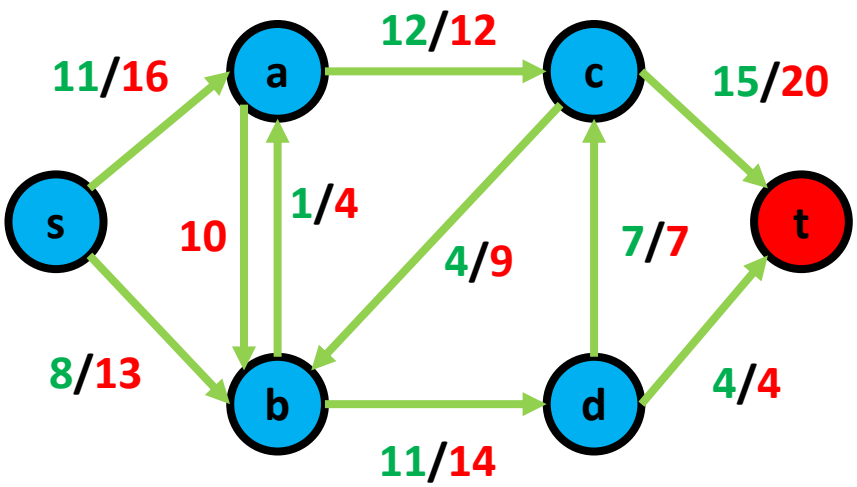
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting?
    - Capacity differs
    - Flow the same
    - But we know flow <= capacity
    - Thus smallest capacity is the upper bound of flow?



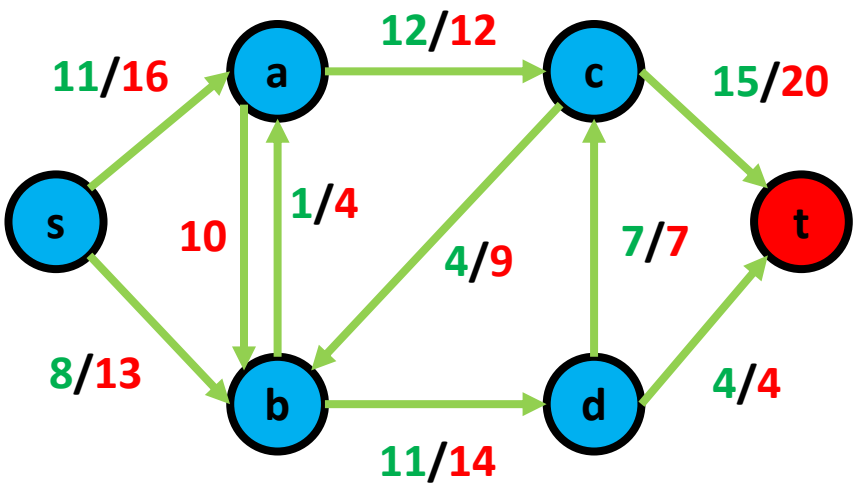
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting? Flow of every cut = flow of network
    - Capacity differs
    - Flow the same
    - But we know flow <= capacity
    - Thus smallest capacity is the upper bound of flow?



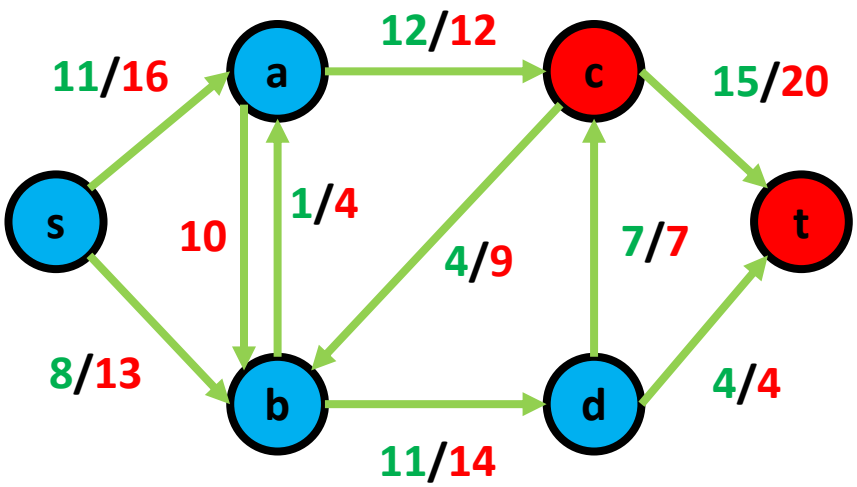
MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting? Flow of every cut = flow of network
    - Capacity differs, smallest is 23!
    - Flow the same
    - But we know flow <= capacity
    - Thus smallest capacity is the upper bound of flow?

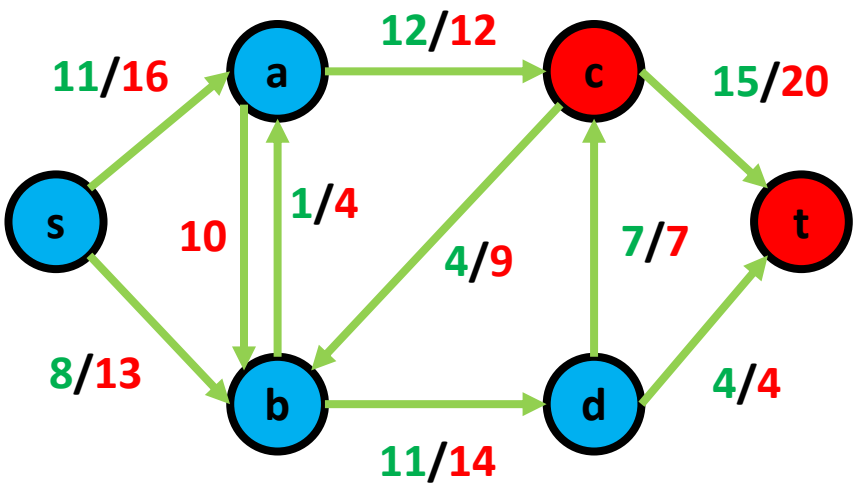


Cut
Flow and Capacity

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting? Flow of every cut = flow of network
    - Capacity differs, smallest is 23!
    - Flow the same
    - But we know flow <= capacity
    - Thus smallest capacity is the upper bound of flow?

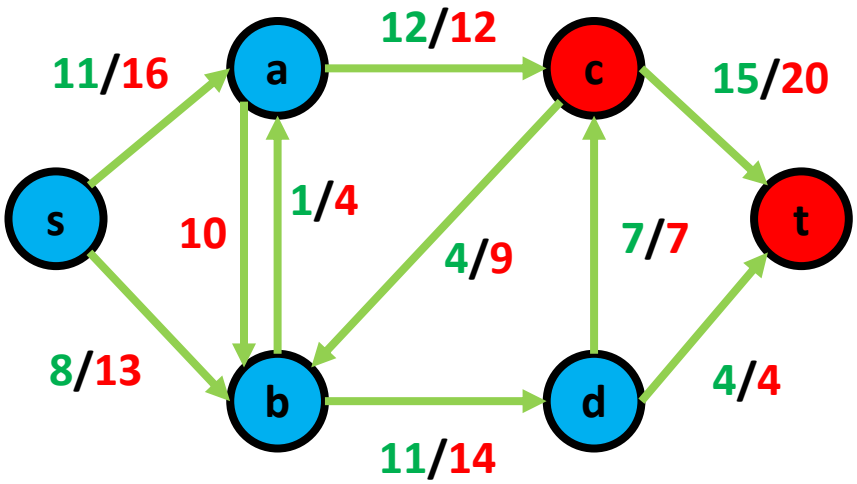


- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting? Flow of every cut = flow of network
    - Capacity differs, smallest is 23!
    - Flow the same, so max flow is 23!
    - But we know flow <= capacity
    - Thus smallest capacity is the upper bound of flow?



MONASH University

- Now we realize something….
  - Remember our earlier cut have capacity of 26 and flow of 19
  - What about other way of cutting? Flow of every cut = flow of network
    - Capacity differs, smallest is 23!
    - Flow the same, so max flow is 23! And we saw earlier when we run Ford-Fulkerson that the maximum flow is 23!
    - But we know flow <= capacity



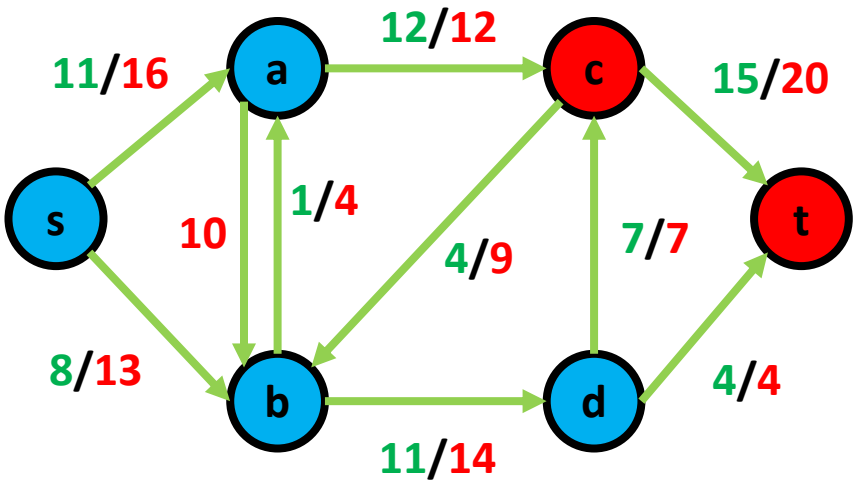
MONASH University

Questions?

Min-Cut Max-Flow Theorem

The key to this…

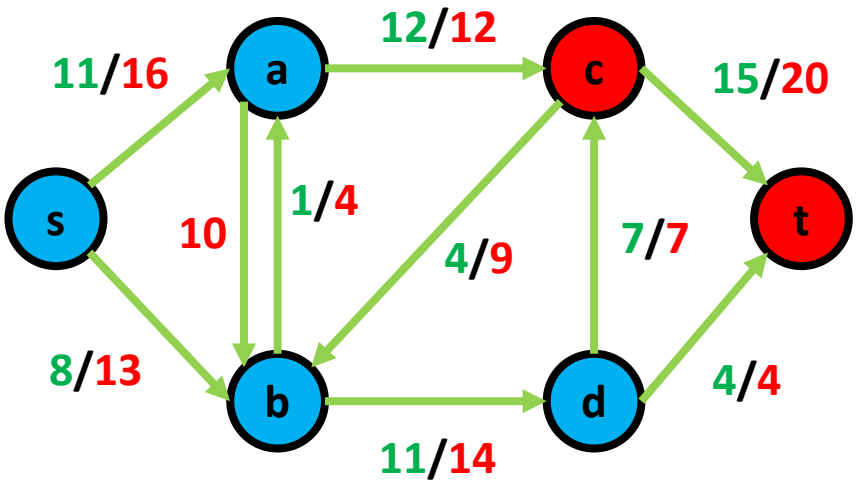
- Minimum capacity of all cut is 23
- Flow of cut <= capacity of cut
- Flow of a cut == flow of a network



Min-Cut Max-Flow Theorem

The key to this…

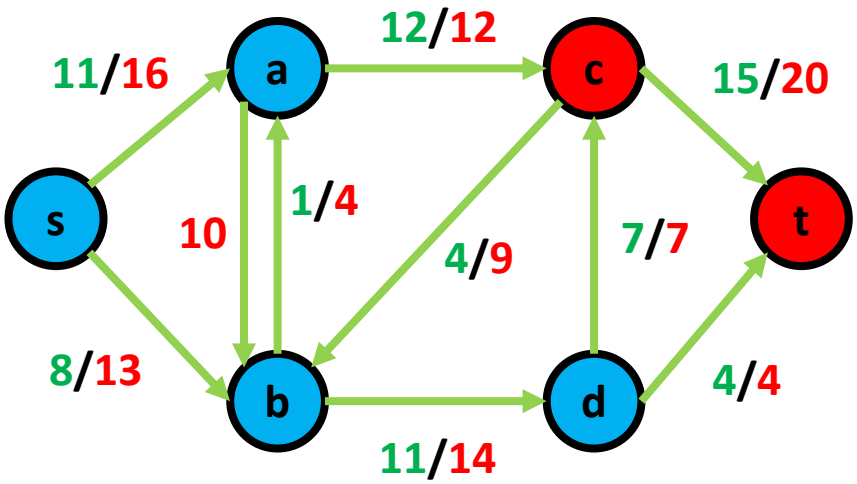
- Minimum capacity of all cut is 23
- Flow of cut <= capacity of cut
- Flow of a cut == flow of a network
- Therefore, capacity of a min-cut = max-flow of a network



Min-Cut Max-Flow Theorem

The key to this…

- Minimum capacity of all cut is 23
- Flow of cut <= capacity of cut
- Flow of a cut == flow of a network
- Therefore, capacity of a min-cut = max-flow of a network



Min-Cut Max-Flow Theorem

The key to this…

- Therefore, capacity of a min-cut = max-flow of a network
- Ford-Fulkerson terminates when there is a cut:

Min-Cut Max-Flow Theorem

The key to this…

MONASH University

- Therefore, capacity of a min-cut = max-flow of a network
- Ford-Fulkerson terminates when there is a cut:
  - Flow on each outgoing edge cut is equal to the capacity of the edge

Min-Cut Max-Flow Theorem

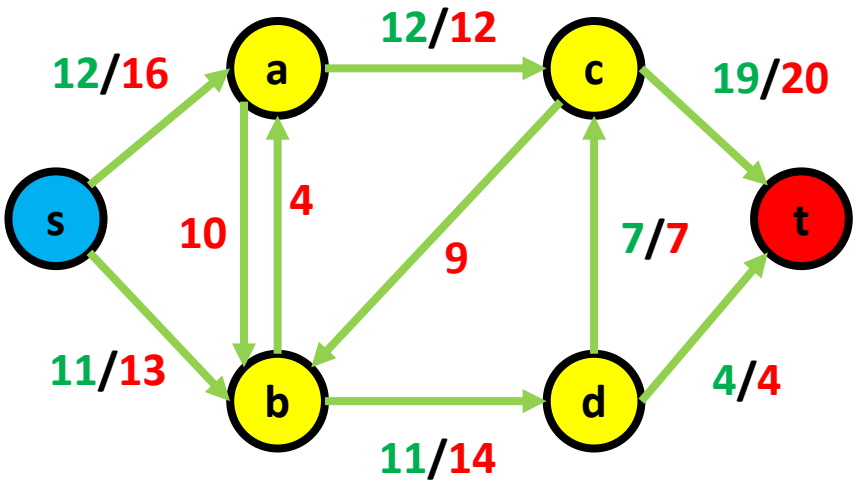
The key to this…

- Therefore, capacity of a min-cut = max-flow of a network
- Ford-Fulkerson terminates when there is a cut:
  - Flow on each outgoing edge cut is equal to the capacity of the edge
  - Flow on each incoming edge to the cut is zero

Min-Cut Max-Flow Theorem

The key to this…

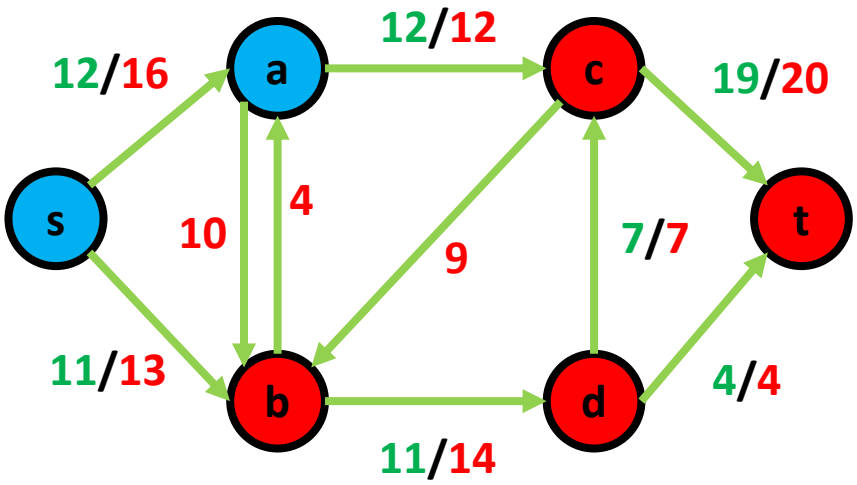
- Therefore, capacity of a min-cut = max-flow of a network
- Ford-Fulkerson terminates when there is a cut:
  - Flow on each outgoing edge cut is equal to the capacity of the edge
  - Flow on each incoming edge to the cut is zero



Min-Cut Max-Flow Theorem

The key to this…

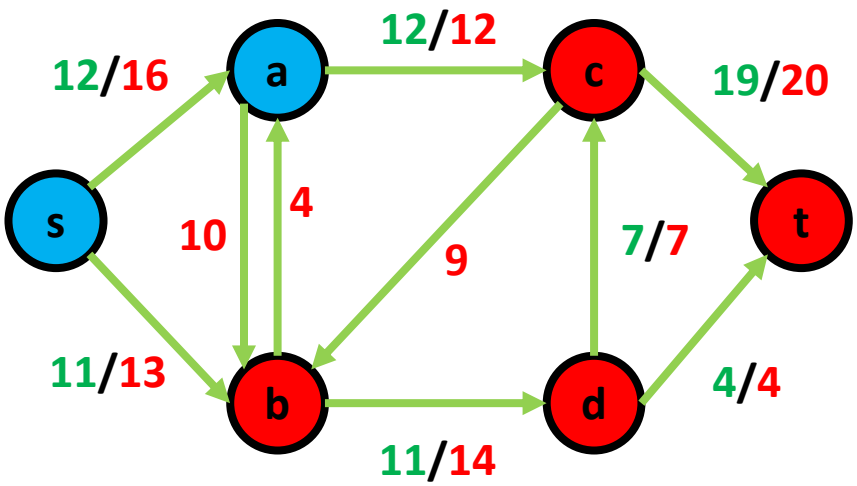
- Therefore, capacity of a min-cut = max-flow of a network
- Ford-Fulkerson terminates when there is a cut:
  - Flow on each outgoing edge cut is equal to the capacity of the edge
  - Flow on each incoming edge to the cut is zero
  - This meets the requirement above right?

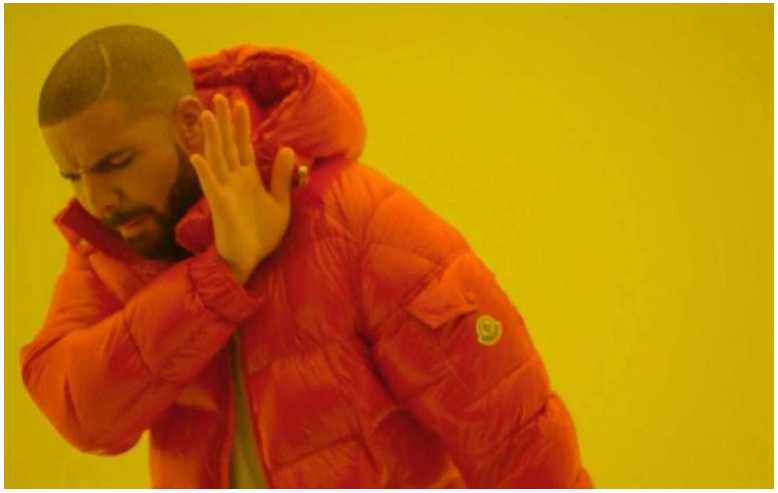


Min-Cut Max-Flow Theorem

The key to this…

- Therefore, capacity of a min-cut = max-flow of a network
- Ford-Fulkerson terminates when there is a cut:
  - Flow on each outgoing edge cut is equal to the capacity of the edge
  - Flow on each incoming edge to the cut is zero
  - This meets the requirement above right? NO!

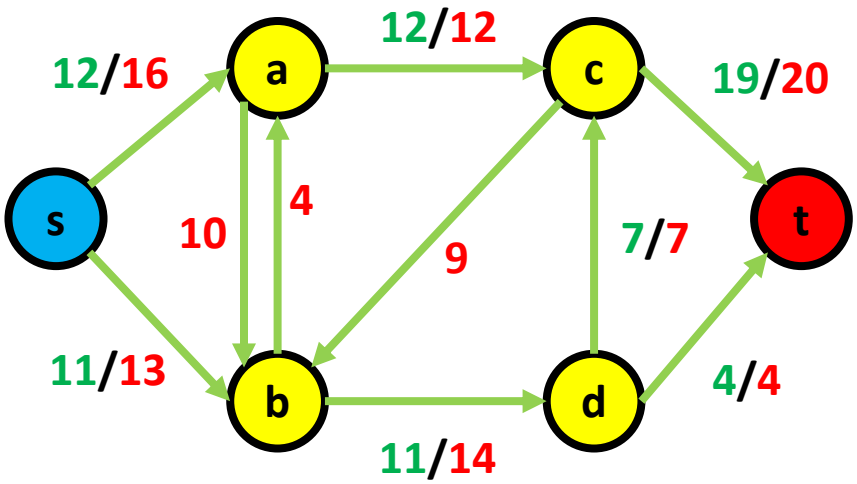




Min-Cut Max-Flow Theorem

The key to this…

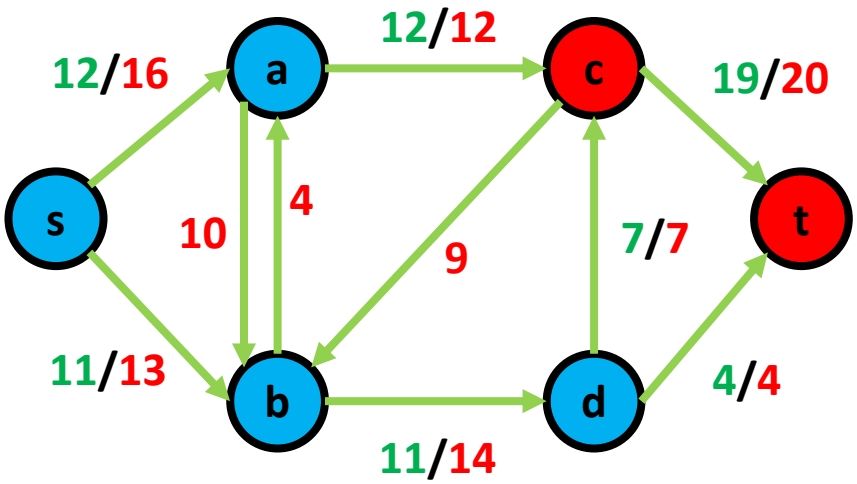
- Therefore, capacity of a min-cut = max-flow of a network
- Ford-Fulkerson terminates when there is a cut:
  - Flow on each outgoing edge cut is equal to the capacity of the edge
  - Flow on each incoming edge to the cut is zero
  - This meets the requirement above right? Which one meet the requirement above?



Min-Cut Max-Flow Theorem

The key to this…

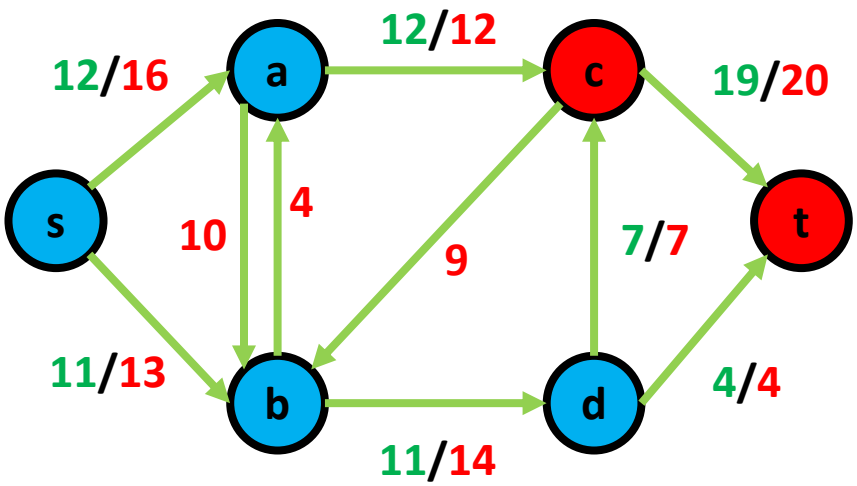
- Therefore, capacity of a min-cut = max-flow of a network
- Ford-Fulkerson terminates when there is a cut:
  - Flow on each outgoing edge cut is equal to the capacity of the edge
  - Flow on each incoming edge to the cut is zero
  - This meets the requirement above right? Which one meet the requirement above? This one!



Min-Cut Max-Flow Theorem

The key to this…

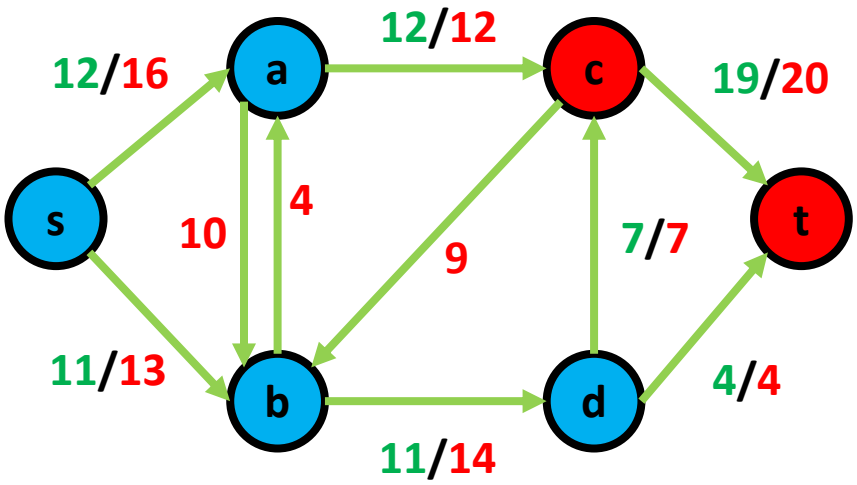
- Therefore, capacity of a min-cut = max-flow of a network
- Ford-Fulkerson terminates when there is a cut:
  - Flow on each outgoing edge cut is equal to the capacity of the edge
  - Flow on each incoming edge to the cut is zero
  - This meets the requirement above right? Which one meet the requirement above? This one! This is called the min-cut



Min-Cut Max-Flow Theorem

The key to this…

- Therefore, capacity of a min-cut = max-flow of a network
- Ford-Fulkerson terminates when there is a cut:
  - Flow on each outgoing edge cut is equal to the capacity of the edge
  - Flow on each incoming edge to the cut is zero
  - This meets the requirement above right? Which one meet the requirement above? This one! This is called the min-cut



Min-cut

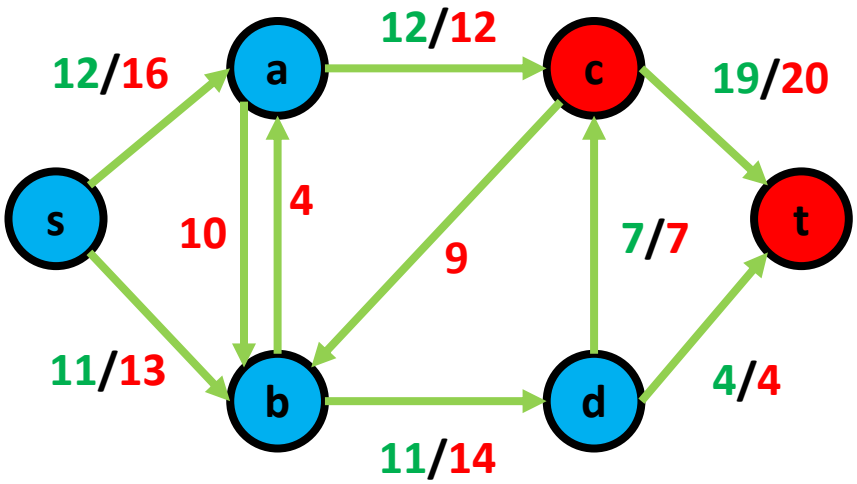
S = (s, a, b, d)

T = (c, t)

Min-Cut Max-Flow Theorem

The key to this…

- Therefore, capacity of a min-cut = max-flow of a network
- Ford-Fulkerson terminates when there is a cut:
  - Flow on each outgoing edge cut is equal to the capacity of the edge
  - Flow on each incoming edge to the cut is zero
  - This meets the requirement above right? Which one meet the requirement above? This one! This is called the min-cut



Go through MUA's slides where his final network differs but still max flow of 23. Can you find such a cut?

MONASH University

Questions?

MONASH University

- We know the following…

MONASH University

- We know the following…
  - Complexity is O(FE) where F is the maximum flow

Optimization

Finding max-flow quicker…

- We know the following…
  - Complexity is O(FE) where F is the maximum flow
  - Each iteration would increase our flow, between 1…F until total flow is F

Optimization

Finding max-flow quicker…

- We know the following…
  - Complexity is O(FE) where F is the maximum flow
  - Each iteration would increase our flow, between 1…F until total flow is F
  - Which is faster?
    - 1,2,3,4,…,F

Optimization

Finding max-flow quicker…

MONASH University

- We know the following…
  - Complexity is O(FE) where F is the maximum flow
  - Each iteration would increase our flow, between 1…F until total flow is F
  - Which is faster?
    - 1,2,3,4,…,F
    - 8,13,41,…,F

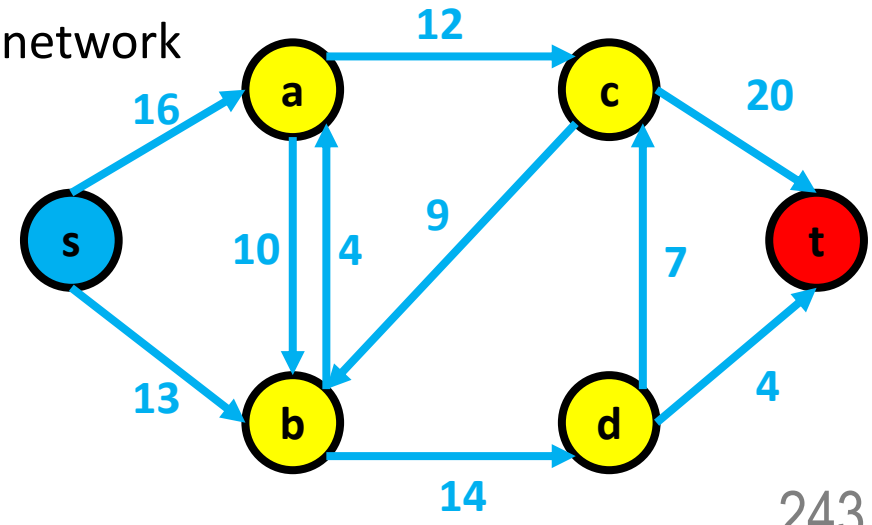
Optimization

Finding max-flow quicker…

MONASH University

- We know the following…
  - Complexity is O(FE) where F is the maximum flow
  - Each iteration would increase our flow, between 1…F until total flow is F
  - Which is faster?
    - 1,2,3,4,…,F
    - 8,13,41,…,F
    - Because we scale quicker!

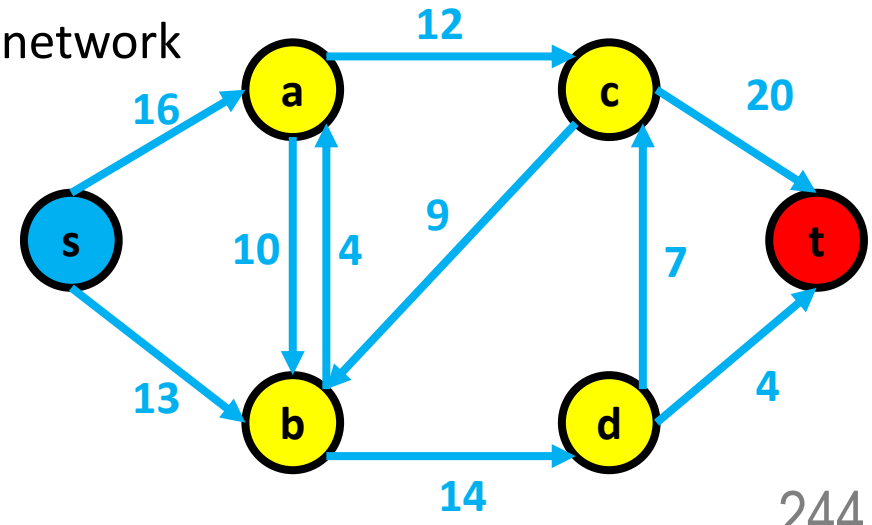
- We know the following…
  - Complexity is O(FE) where F is the maximum flow
  - Each iteration would increase our flow, between 1…F until total flow is F
  - Which is faster?
    - 1,2,3,4,…,F
    - 8,13,41,…,F
    - Because we scale quicker!
  - But this depends on our residual network



Optimization

Finding max-flow quicker…

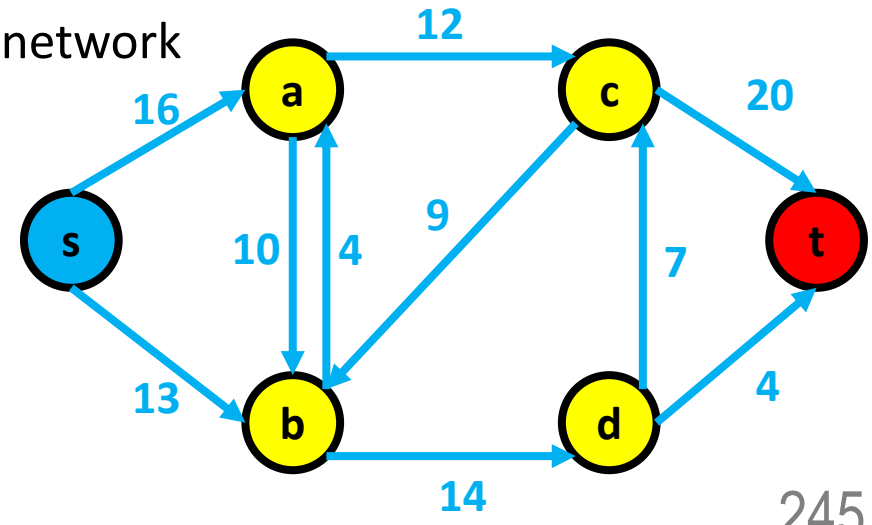
- We know the following…
  - Complexity is O(FE) where F is the maximum flow
  - Each iteration would increase our flow, between 1…F until total flow is F
  - Which is faster?
    - 1,2,3,4,…,F
    - 8,13,41,…,F
    - Because we scale quicker!
  - But this depends on our residual network
    - We used BFS/ DFS
    - Can we choose a better path?



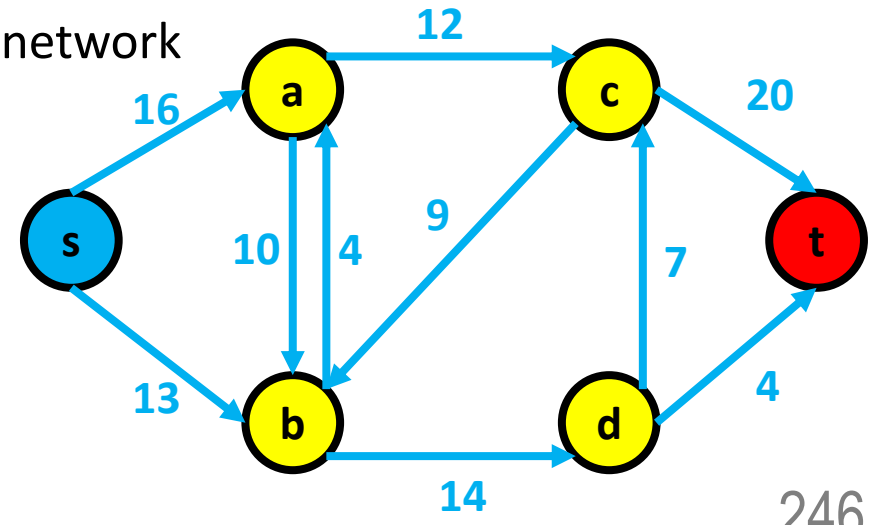
Optimization

Finding max-flow quicker…

- We know the following…
  - Complexity is O(FE) where F is the maximum flow
  - Each iteration would increase our flow, between 1…F until total flow is F
  - Which is faster?
    - 1,2,3,4,…,F
    - 8,13,41,…,F
    - Because we scale quicker!
  - But this depends on our residual network
    - We used BFS/ DFS
    - Can we choose a better path?
      - s -> b -> d -> t?



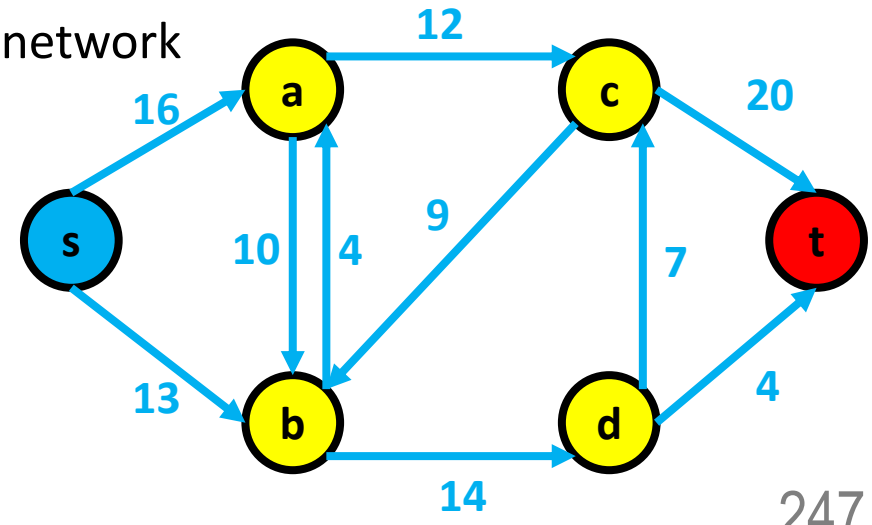
- We know the following…
  - Complexity is O(FE) where F is the maximum flow
  - Each iteration would increase our flow, between 1…F until total flow is F
  - Which is faster?
    - 1,2,3,4,…,F
    - 8,13,41,…,F
    - Because we scale quicker!
  - But this depends on our residual network
    - We used BFS/ DFS
    - Can we choose a better path?
      - s -> b -> d -> t?
      - s -> a -> c -> t?



Optimization

Finding max-flow quicker…

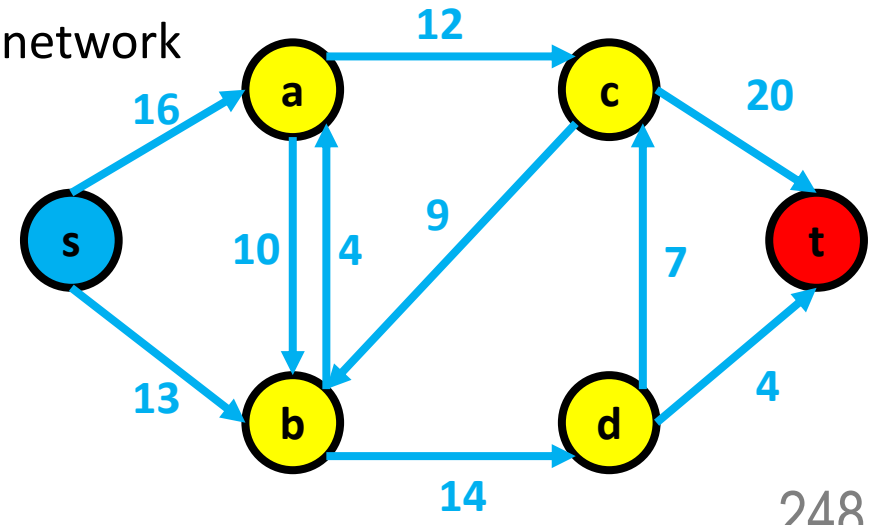
- We know the following…
  - Complexity is O(FE) where F is the maximum flow
  - Each iteration would increase our flow, between 1…F until total flow is F
  - Which is faster?
    - 1,2,3,4,…,F
    - 8,13,41,…,F
    - Because we scale quicker!
  - But this depends on our residual network
    - We used BFS/ DFS
    - Can we choose a better path?
      - s -> b -> d -> t?
      - s -> a -> c -> t?
      - s -> a -> b -> d -> t?



Optimization

Finding max-flow quicker…

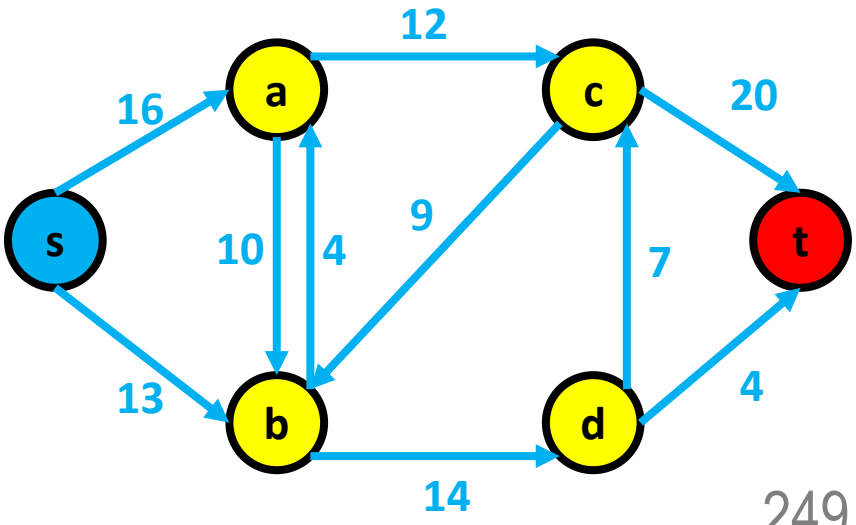
- We know the following…
  - Complexity is O(FE) where F is the maximum flow
  - Each iteration would increase our flow, between 1…F until total flow is F
  - Which is faster?
    - 1,2,3,4,…,F
    - 8,13,41,…,F
    - Because we scale quicker!
  - But this depends on our residual network
    - We used BFS/ DFS
    - Can we choose a better path?
      - s -> b -> d -> t?
      - s -> a -> c -> t?
      - s -> a -> b -> d -> t?



Edmond-Karp

Finding max-flow quicker…

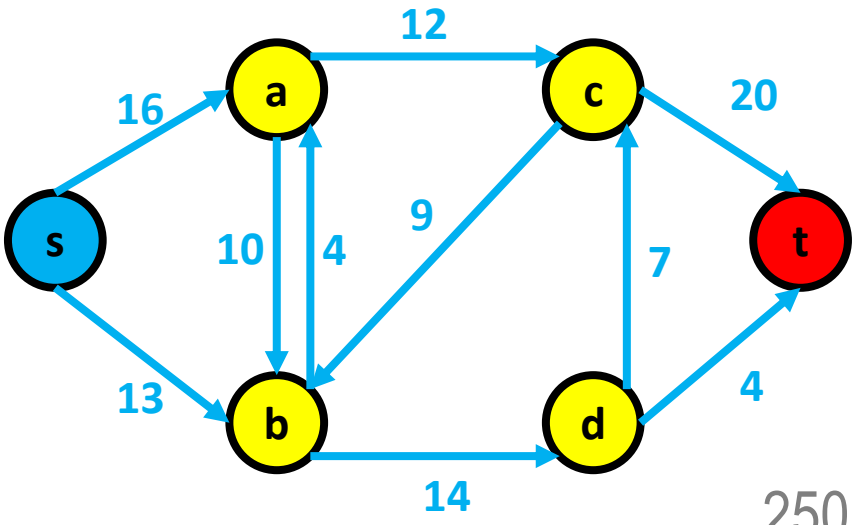
- Consider the following:



Edmond-Karp

Finding max-flow quicker…

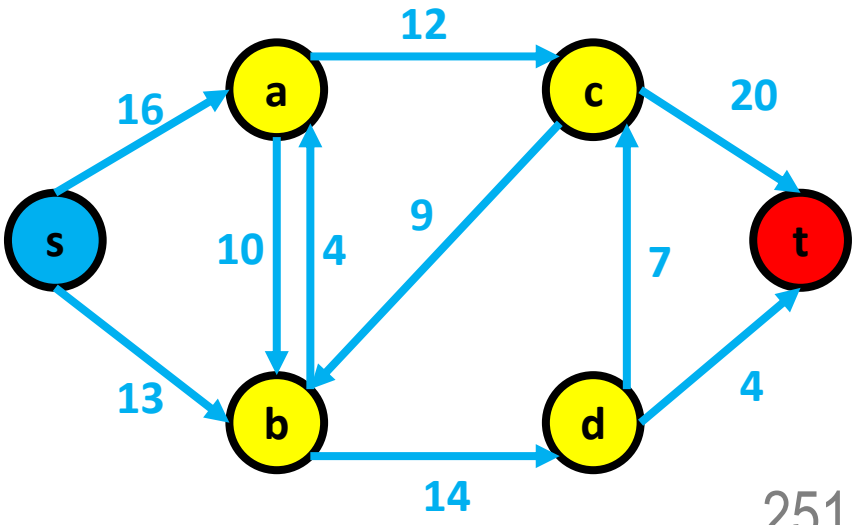
- Consider the following:
  - Dinic/ Edmonds-Karp = choose path with fewest edge
  - Edmonds-Karp = choose path with largest bottle neck



Edmond-Karp

Finding max-flow quicker…

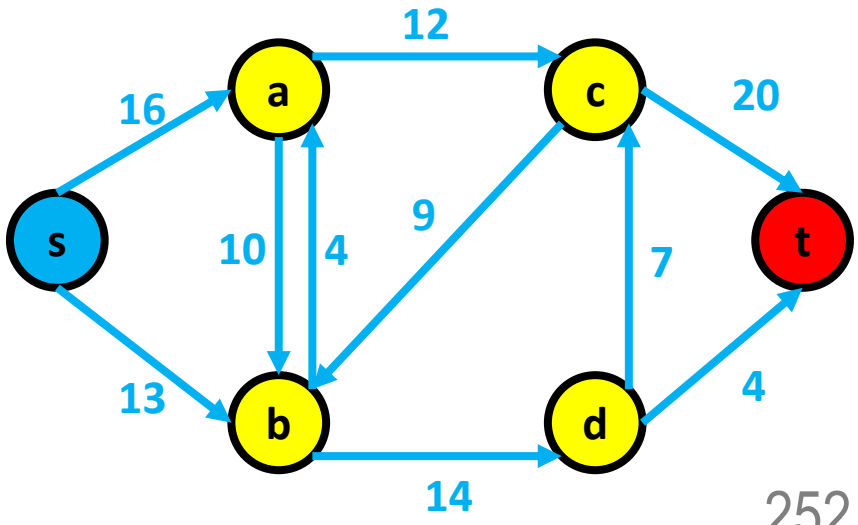
- Consider the following:
  - Dinic/ Edmonds-Karp = choose path with fewest edge
  - Edmonds-Karp = choose path with largest bottle neck
  - How?



Edmond-Karp

Finding max-flow quicker…

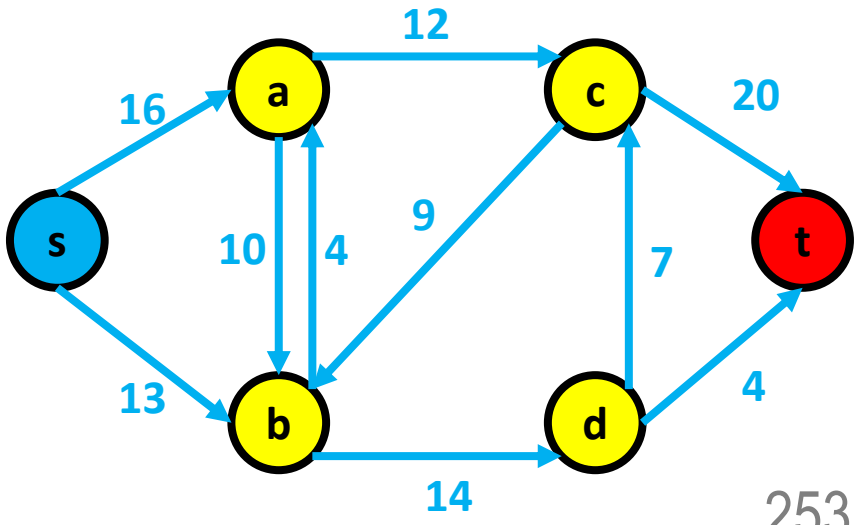
- Consider the following:
  - Dinic/ Edmonds-Karp = choose path with fewest edge
    - Running BFS ensure this isn't it?
  - Edmonds-Karp = choose path with largest bottle neck



Edmond-Karp

Finding max-flow quicker…

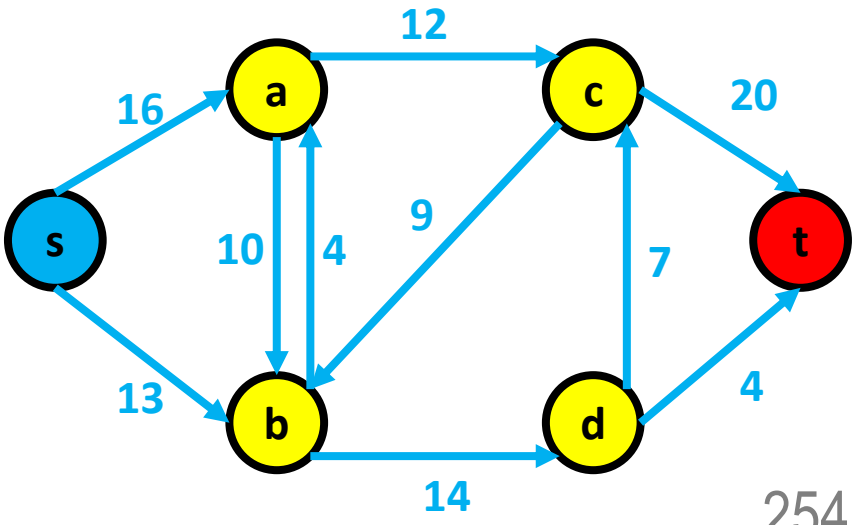
- Consider the following:
  - Dinic/ Edmonds-Karp = choose path with fewest edge
    - Running BFS ensure this isn't it?
    - Less edge, lower chance to get smaller weights
  - Edmonds-Karp = choose path with largest bottle neck



Edmond-Karp

Finding max-flow quicker…

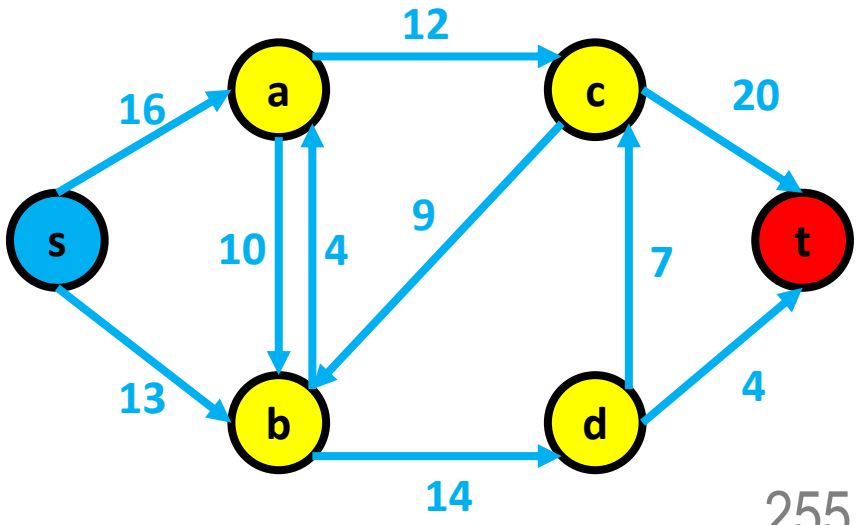
- Consider the following:
  - Dinic/ Edmonds-Karp = choose path with fewest edge
    - Running BFS ensure this isn't it?
    - Less edge, lower chance to get smaller weights
  - Edmonds-Karp = choose path with largest bottle neck
    - Find a maximum spanning tree!



Edmond-Karp

Finding max-flow quicker…

- Consider the following:
  - Dinic/ Edmonds-Karp = choose path with fewest edge
    - Running BFS ensure this isn't it?
    - Less edge, lower chance to get smaller weights
  - Edmonds-Karp = choose path with largest bottle neck
    - Find a maximum spanning tree!
    - Recall we discuss a little in FIT2004 Tutorial 10 about bottleneck…



MONASH University

Questions?

Bipartite Graph

An application of network flow…

MONASH University

- Consider the situation…

Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…

Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - A list of men
  - A list of women

Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - A list of men
  - A list of women
  - Their preferences

Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - A list of men
  - A list of women
  - Their preferences
  - Each match you get, you earn $$$

Bipartite Graph

An application of network flow…

MONASH University

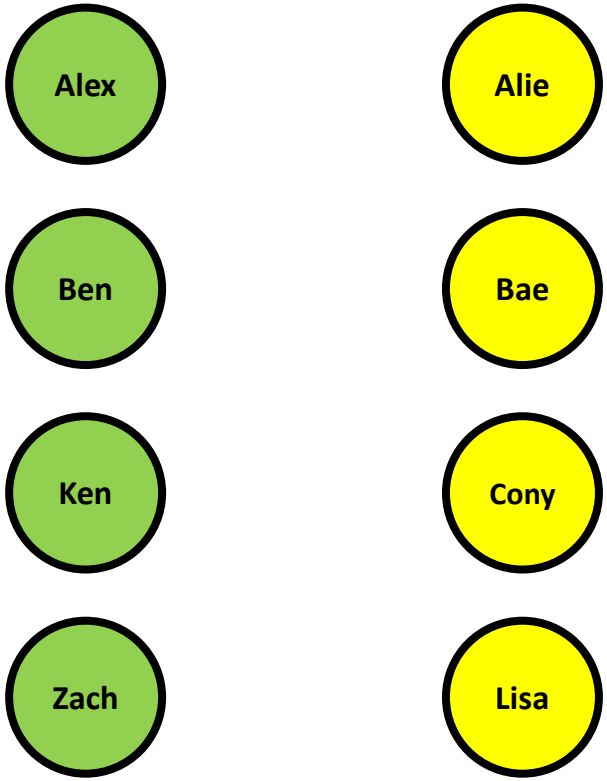
- Arranged marriage agency…
  - A list of men
  - A list of women
  - Their preferences
  - Each match you get, you earn $$$
  - So we want the most matches!

Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…

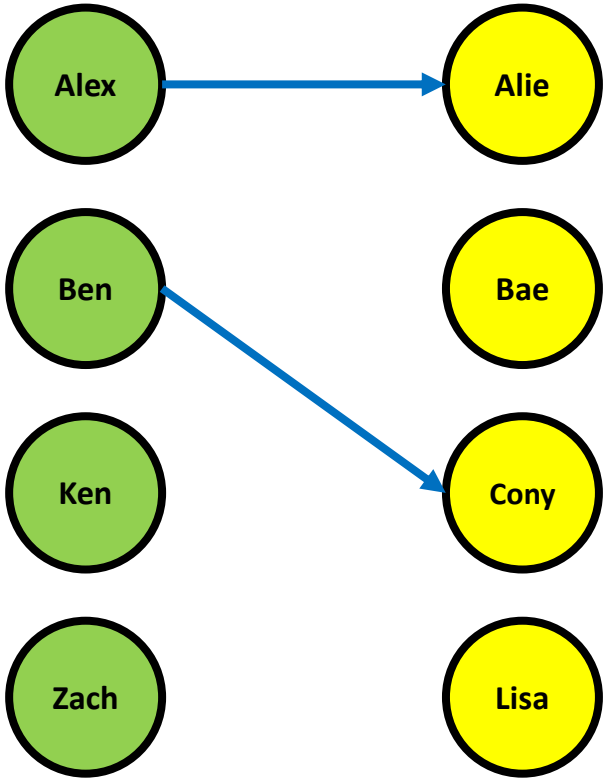


Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - Can you max profit?

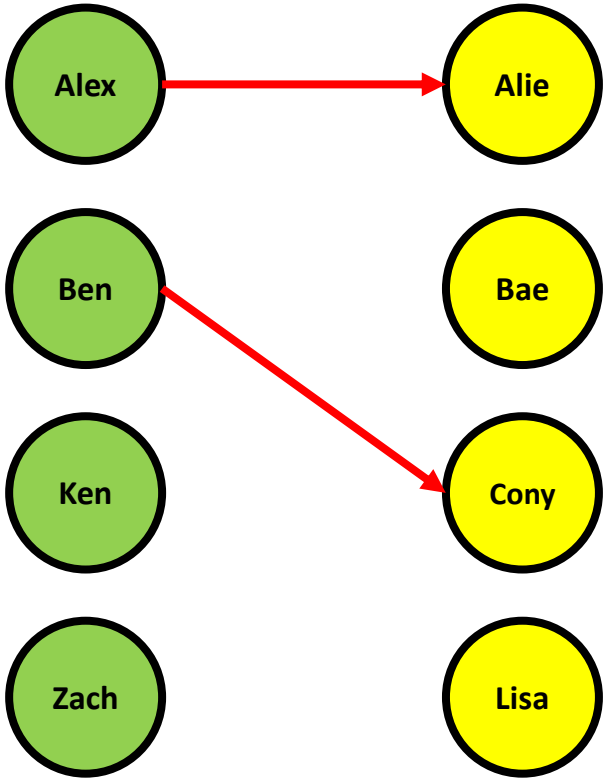


Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - Can you max profit?

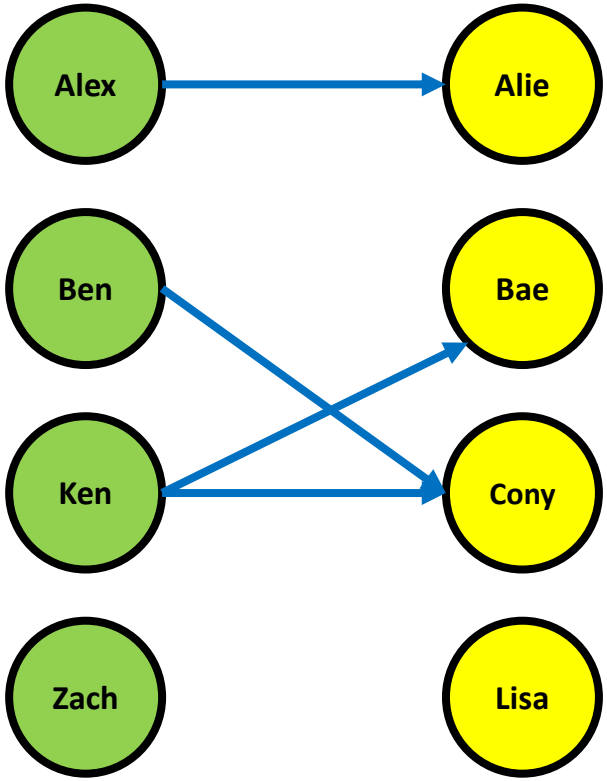


Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - Can you max profit? What about now?

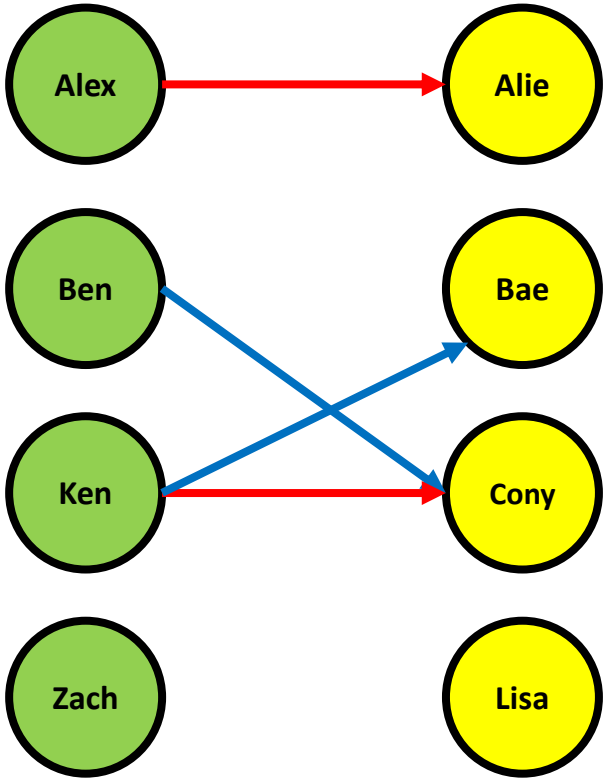


Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - Can you max profit? What about now? Only 2…

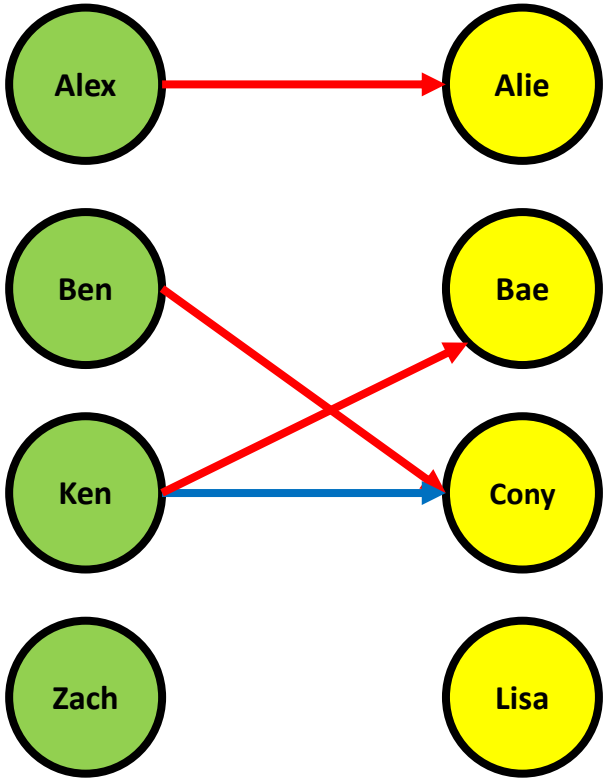


Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - Can you max profit? What about now? We can do better with 3

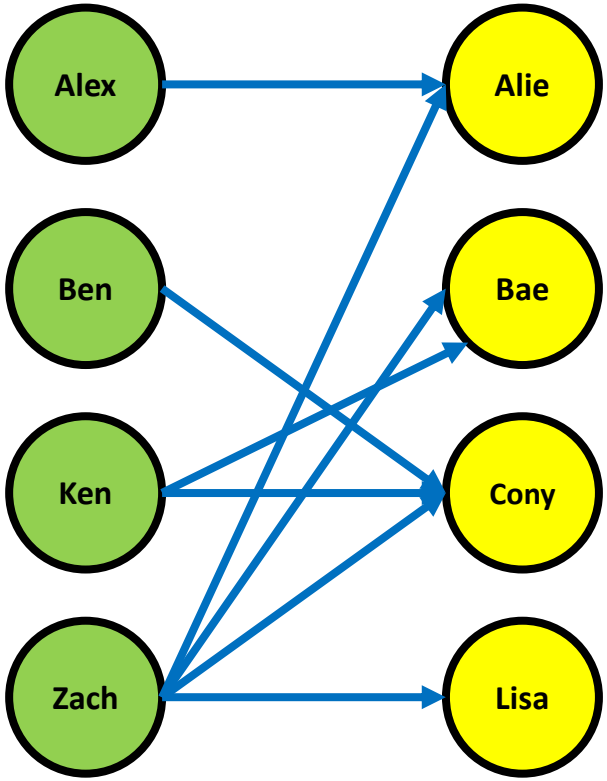


Bipartite Graph

An application of network flow…

MONASH University

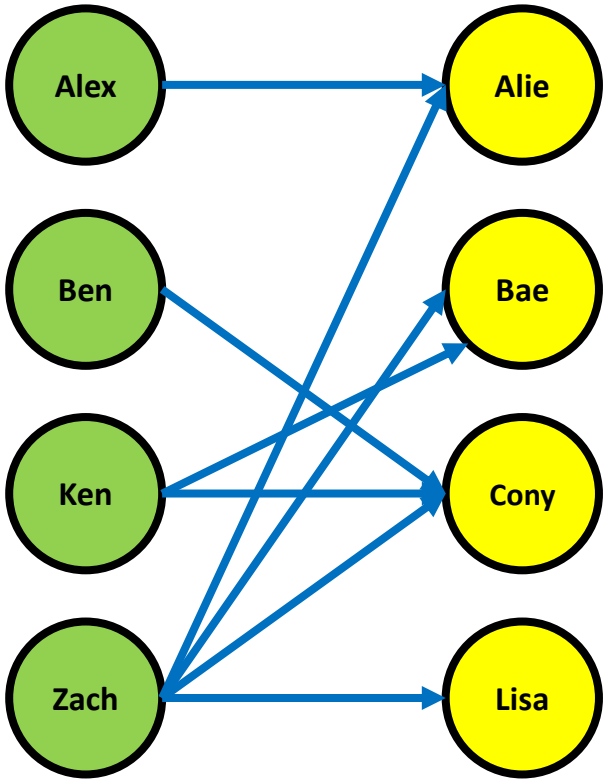
- Arranged marriage agency…
  - What about now???



Bipartite Graph

An application of network flow…

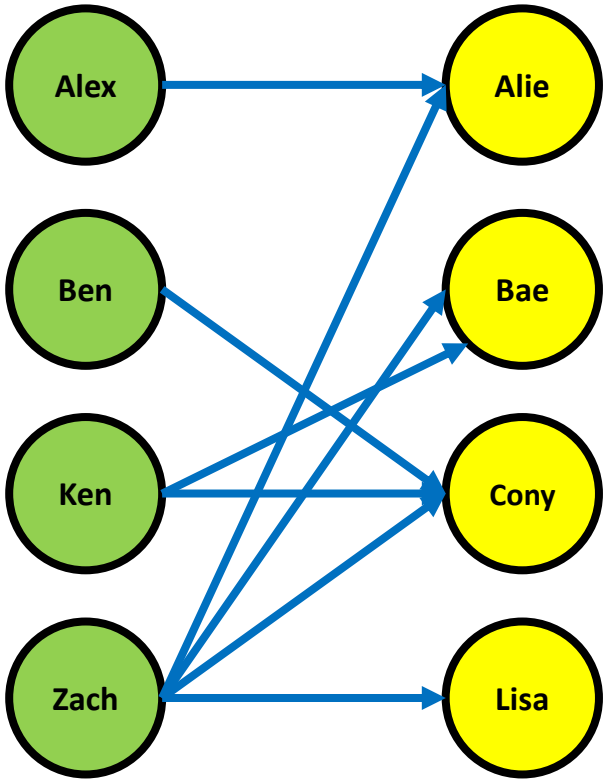
- Arranged marriage agency…
  - What about now???



Bipartite Graph

An application of network flow…

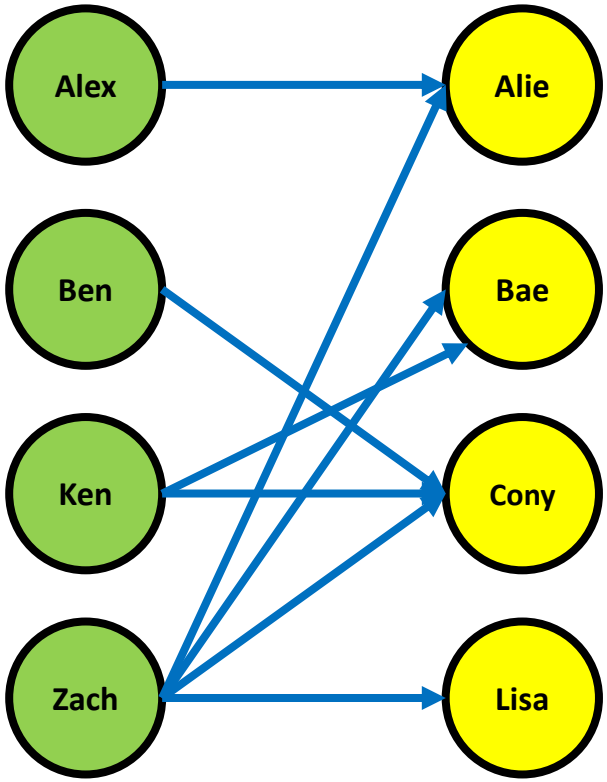
- Arranged marriage agency…
  - What about now???
  - Imagine you run a bigger agency!
    - 100s of eligible bachelor
    - 100s of eligible ladies



Bipartite Graph

An application of network flow…

- Arranged marriage agency…
  - What about now???
  - Imagine you run a bigger agency!
    - 100s of eligible bachelor
    - 100s of eligible ladies
    - Or Tinder!

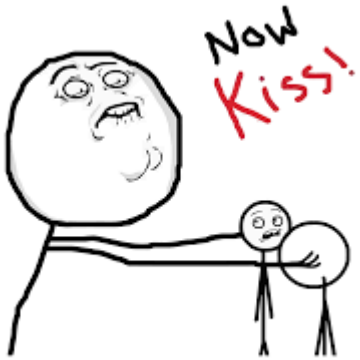


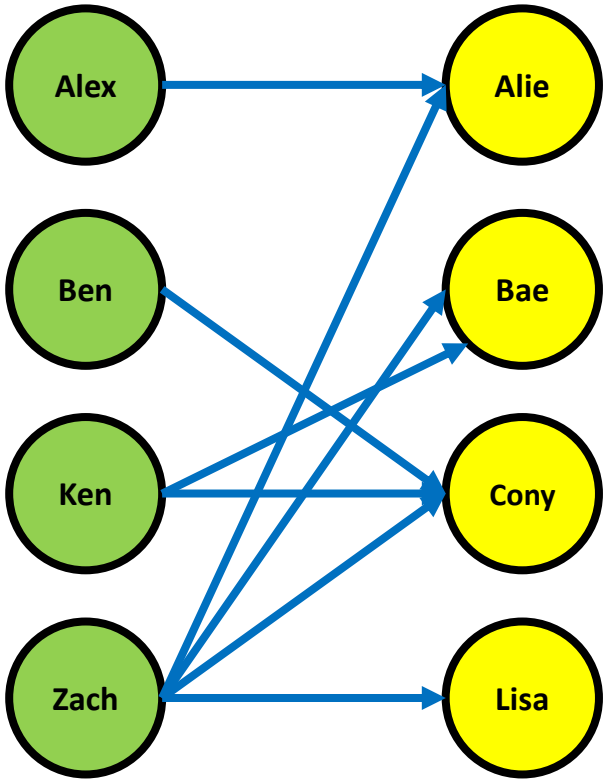
Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - What about now???
  - Imagine you run a bigger agency!
    - 100s of eligible bachelor
    - 100s of eligible ladies
    - Or Tinder!
  - What would you do?

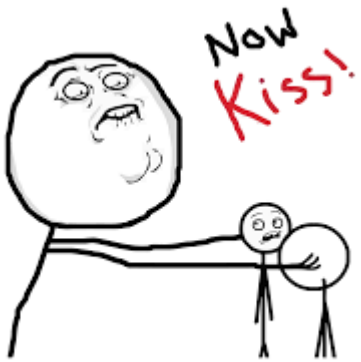


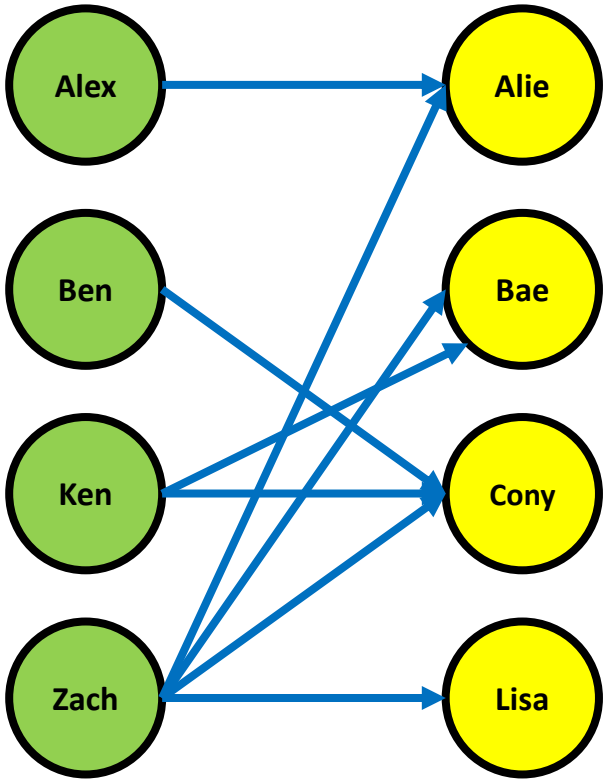


Bipartite Graph

An application of network flow…

- Arranged marriage agency…
  - What about now???
  - Imagine you run a bigger agency!
    - 100s of eligible bachelor
    - 100s of eligible ladies
    - Or Tinder!
  - What would you do? Maximize the flow!



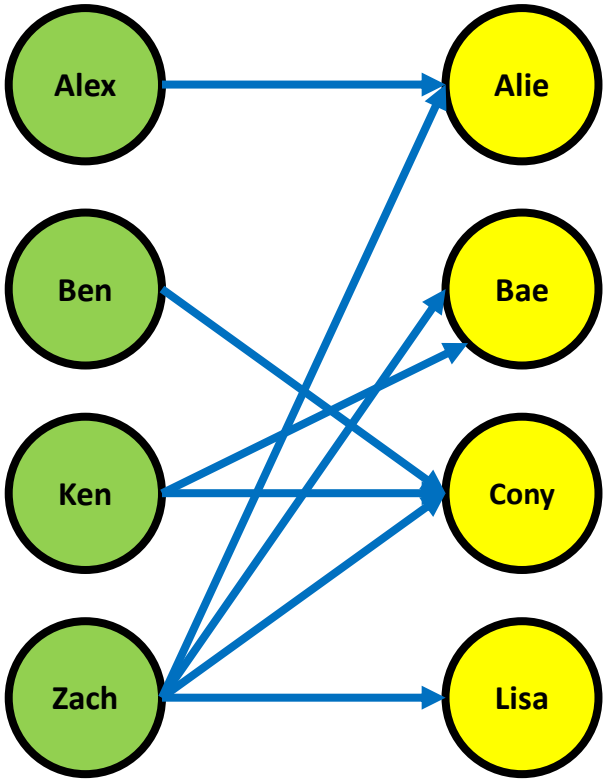


Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - What would you do? Maximize the flow!

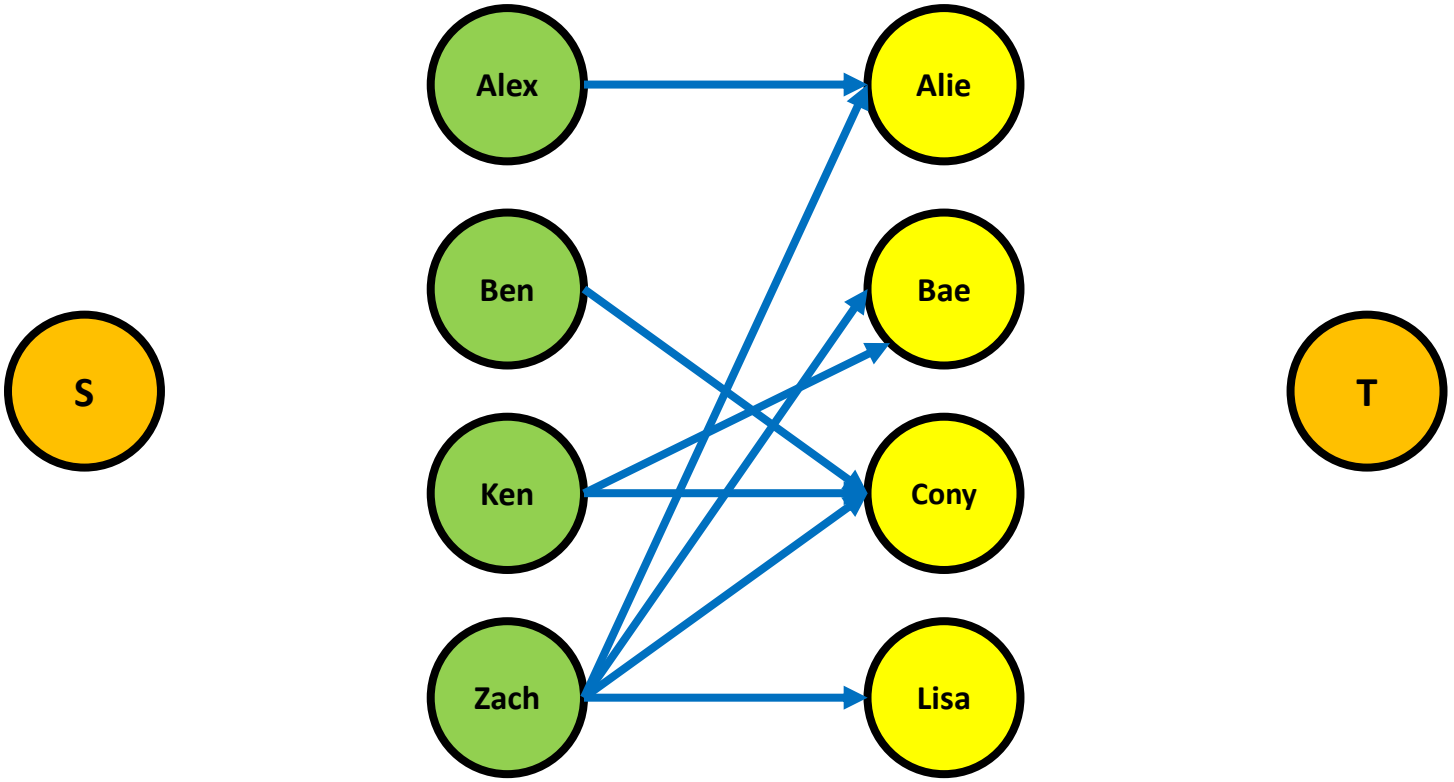


Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - What would you do? Maximize the flow!

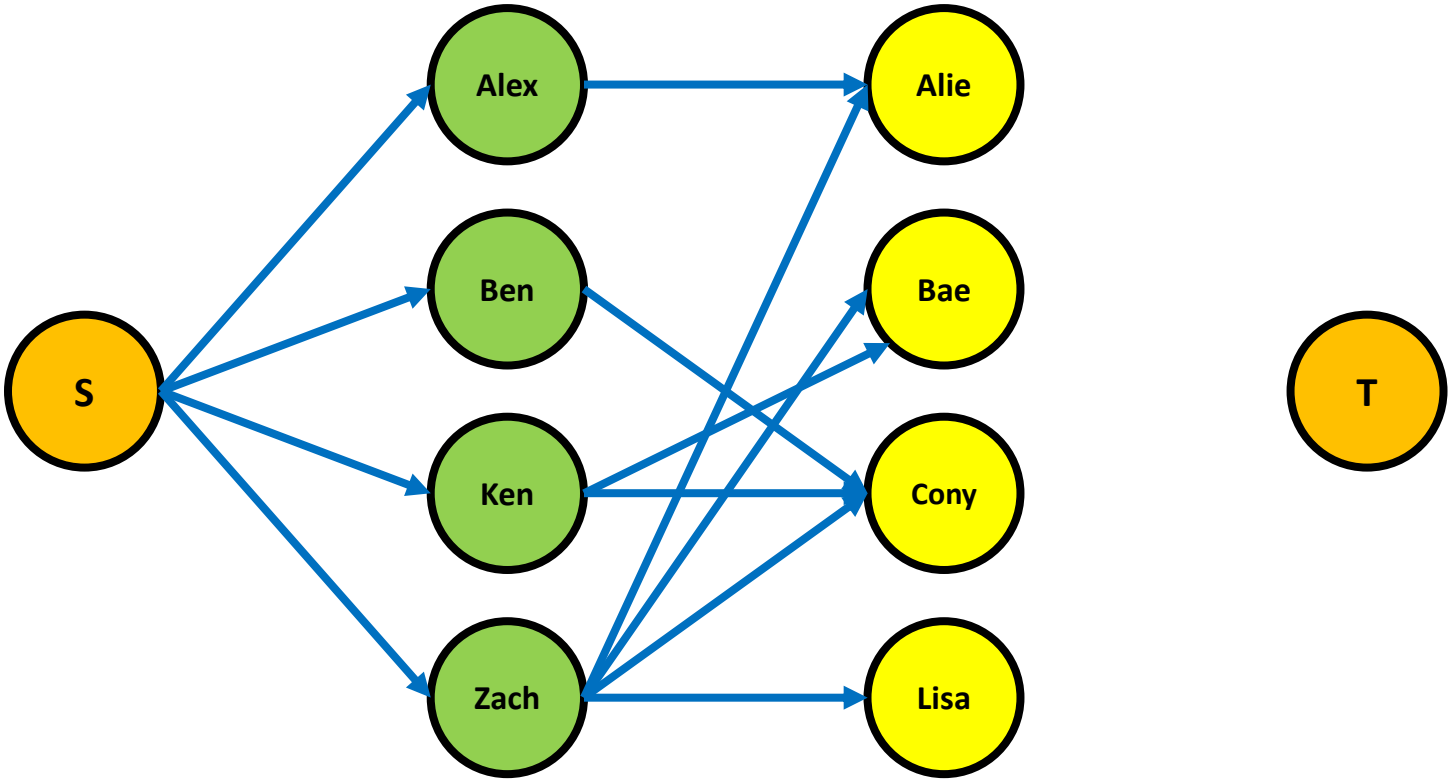


Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - What would you do? Maximize the flow!

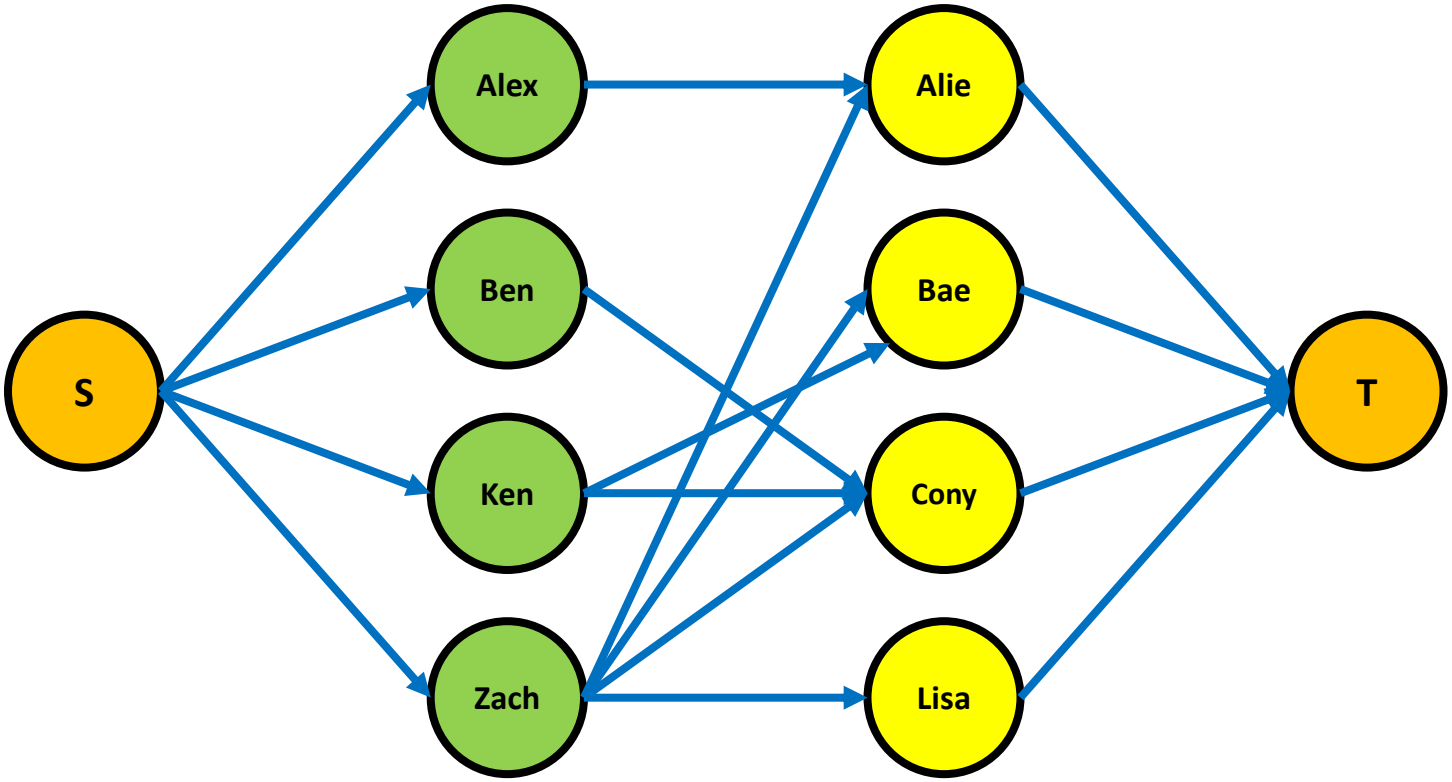


Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - What would you do? Maximize the flow!

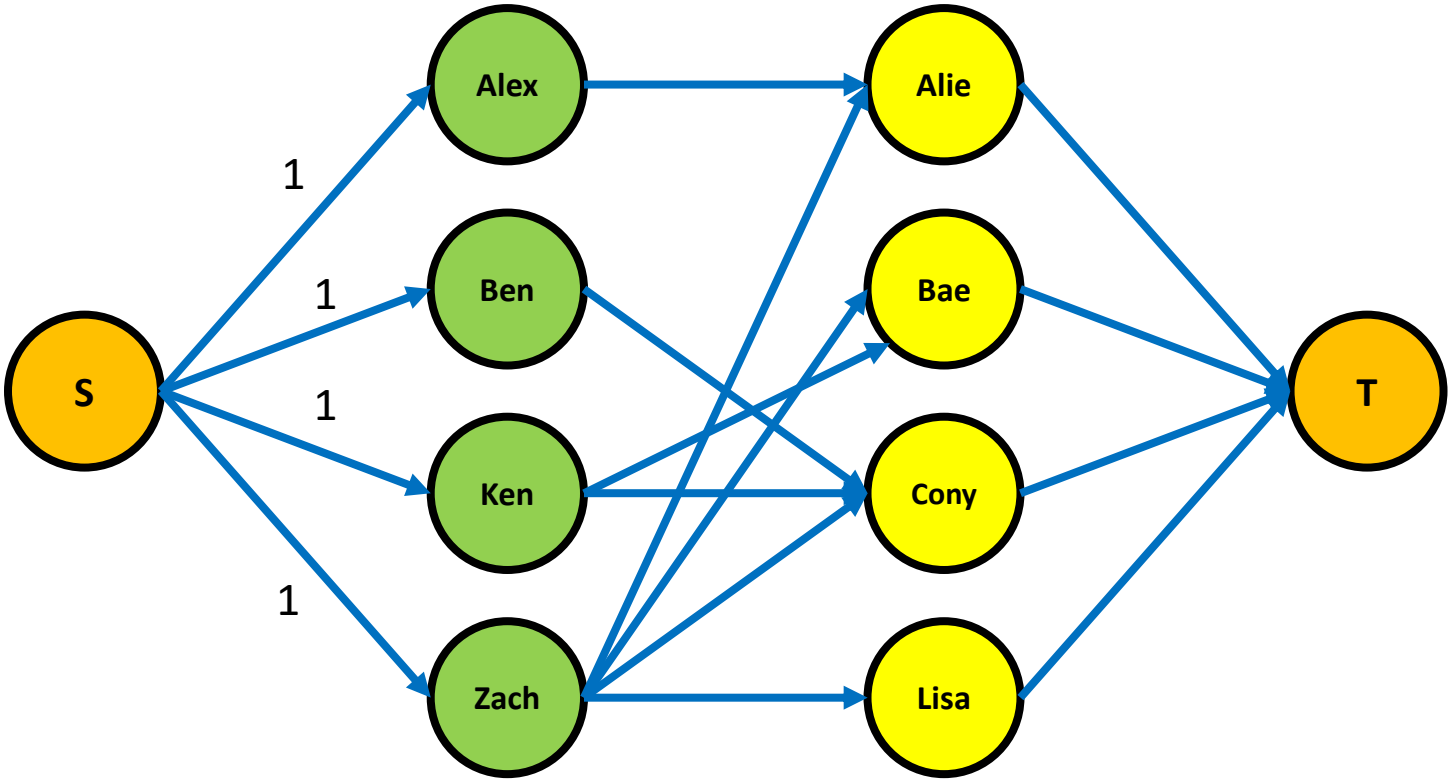


Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - What would you do? Maximize the flow! … I won't finish this #lazy2draw

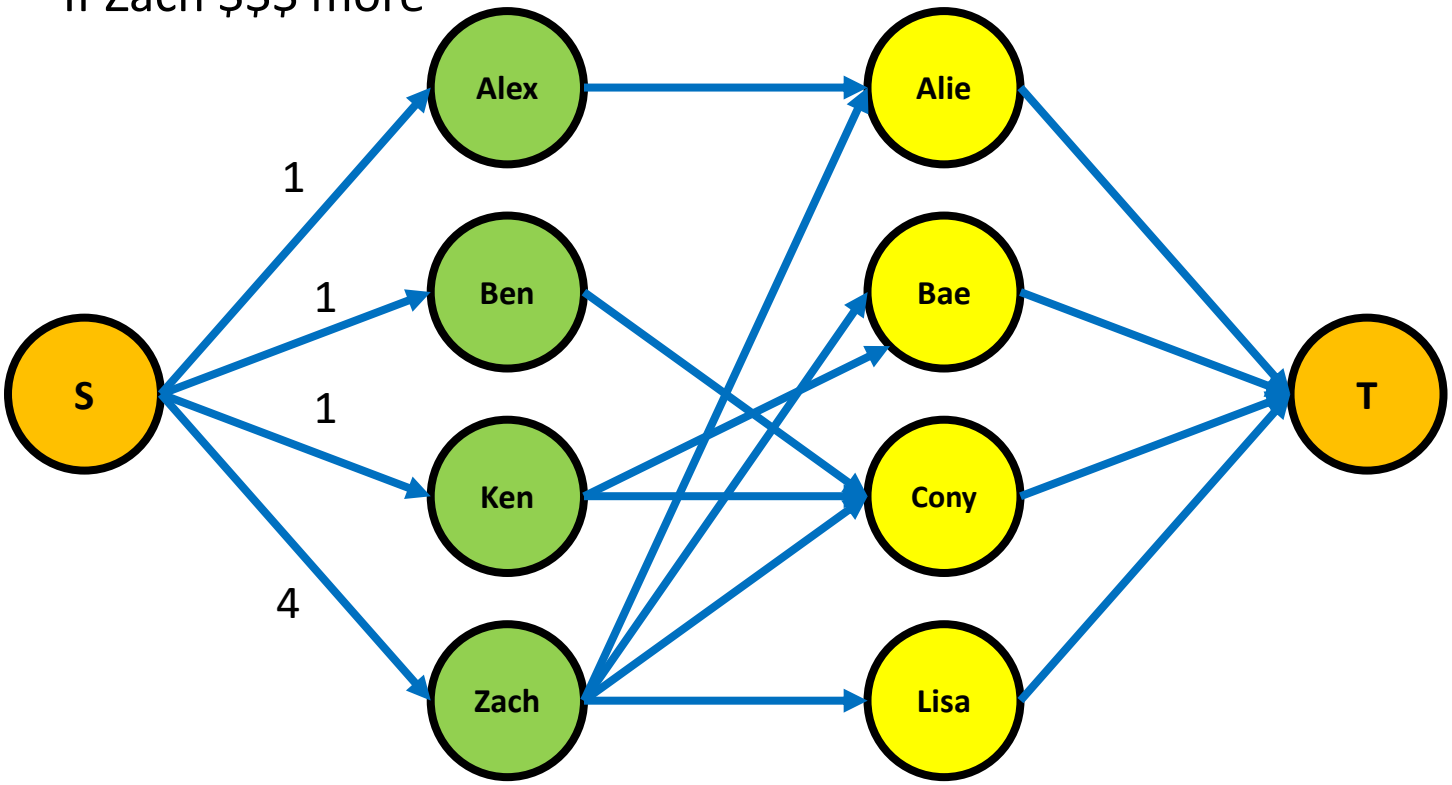


Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - What would you do? Maximize the flow! … I won't finish this #lazy2draw
  - If Zach $$$ more



MONASH University

Questions?

Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - A list of men
  - A list of women
  - Their preferences
  - Each match you get, you earn $$$
  - So we want the most matches!

- Can you think of other applications?

Bipartite Graph

An application of network flow…

MONASH University

- Arranged marriage agency…
  - A list of men
  - A list of women
  - Their preferences
  - Each match you get, you earn $$$
  - So we want the most matches!
- Can you think of other applications?
  - Think of Monash

MONASH University

Questions?

MONASH University

Thank You