Lecture 5

Lecture 5

Previously in Opt 2 …

The maximum-flow problem

- Input
  - $G = (N, E)$, a directed graph
  - The node set N contains:
    - A source node, s
    - A sink node, t
  - u_{ij} = edge capacity; $u_{ij} \geq 0$
- Objective: To maximize net flow into t, subject to constraints:
  - Capacity constraints
  - Flow conservation constraints:

    Net flow out of node i = 0

    (for each node i in N, except s and t)

The maximum-flow problem

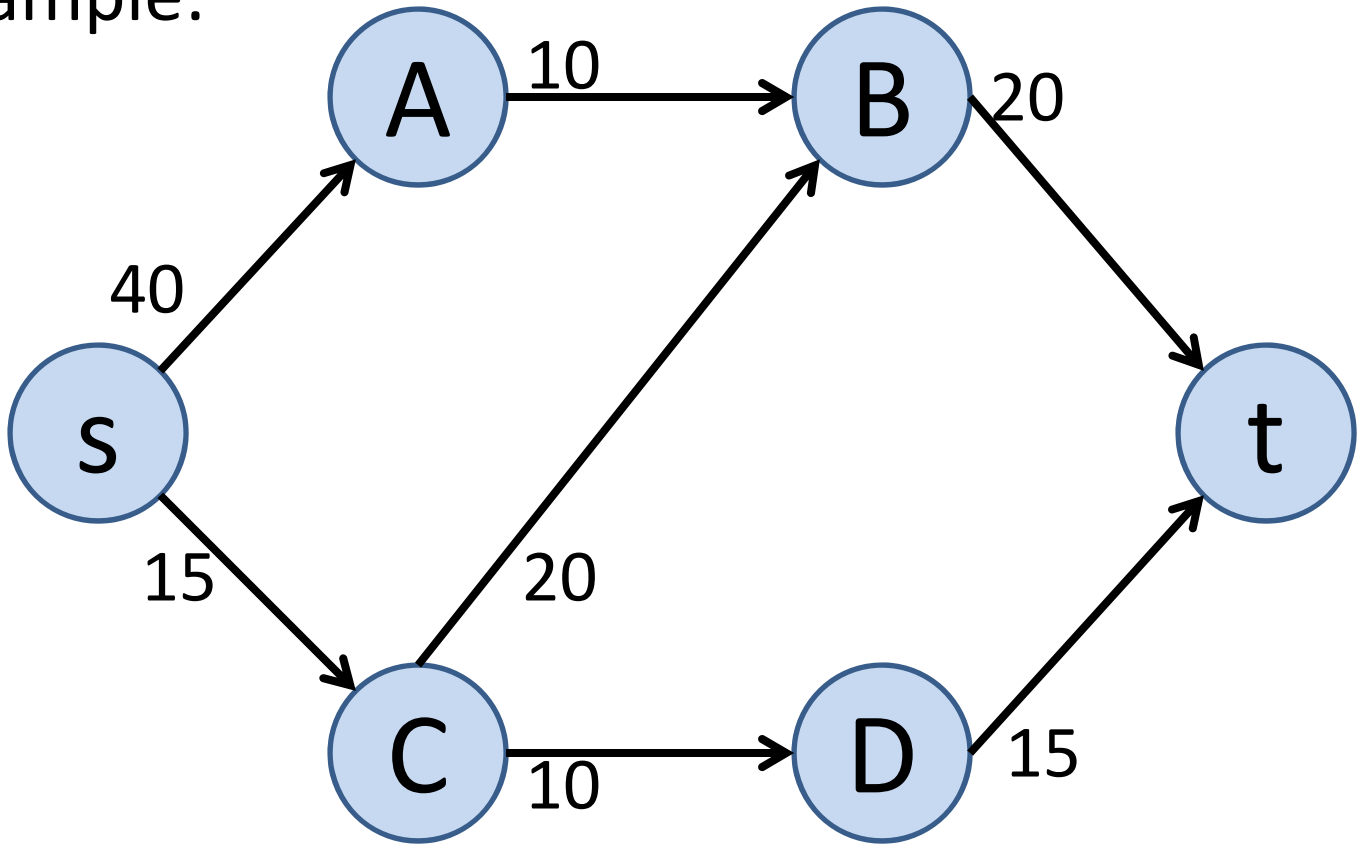
- Input
  - $G = (N, E)$, a directed graph
  - The node set N contains:
    - A source node, s
    - A sink node, t
  - u_{ij} = edge capacity; $u_{ij} \geq 0$
- Objective: To maximize net flow into t, subject to constraints:
  - Capacity constraints
  - Flow conservation constraints:

    Total flow into i = total flow out of i

    (for each node i in N, except s and t)

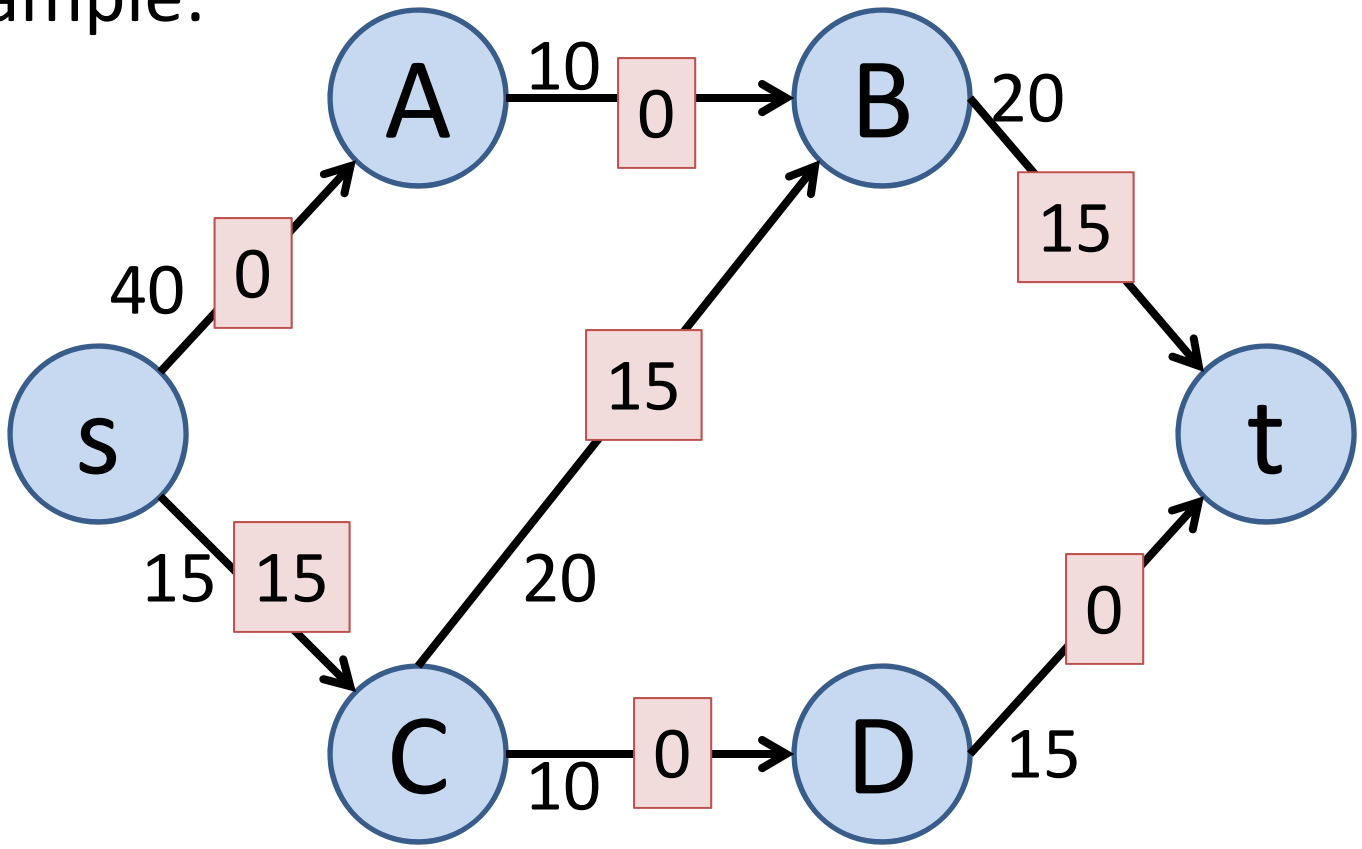
The maximum-flow problem

- Example:



The maximum-flow problem

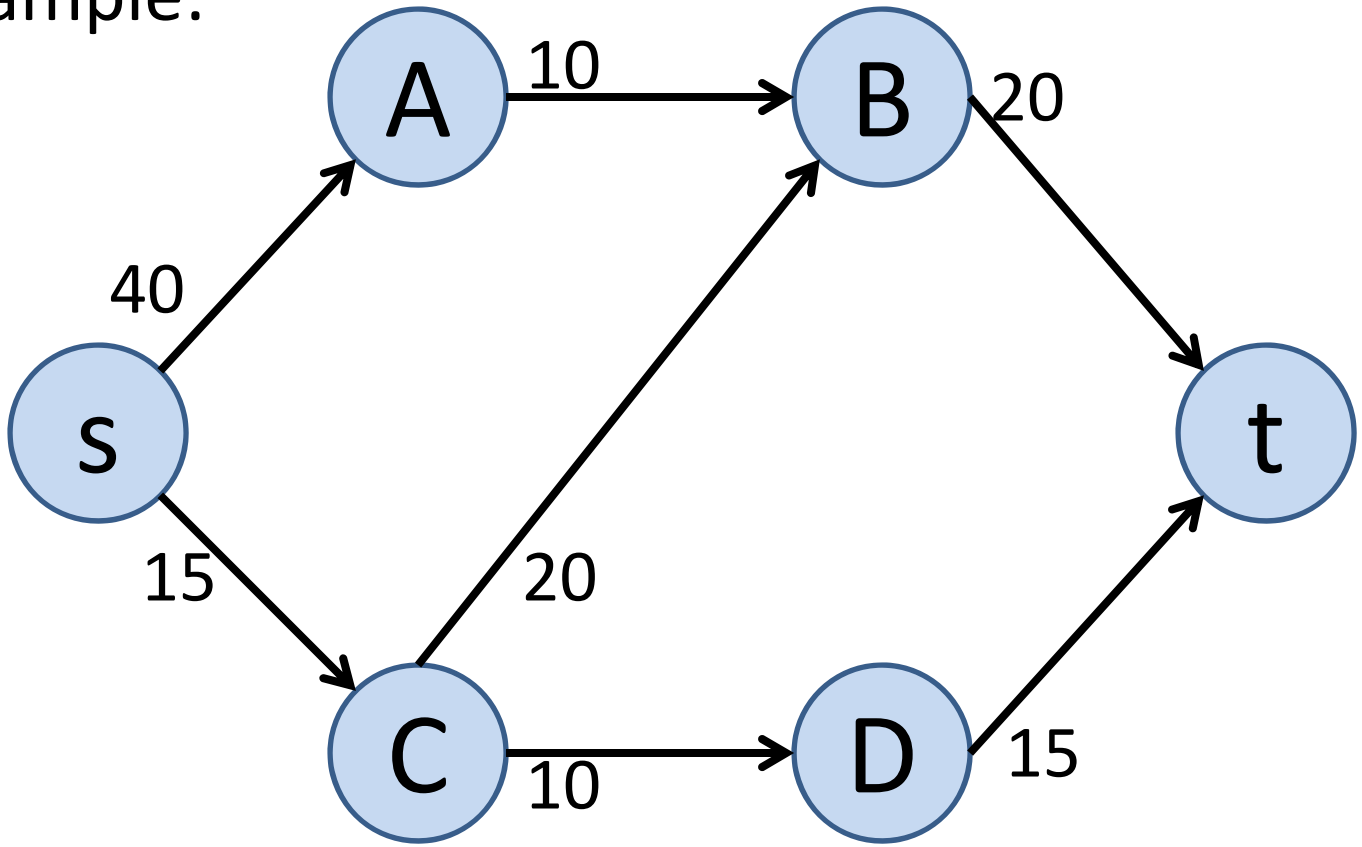
- Example:



Flow value = 15

Cuts and cut capacities

- Example:



Q1 (i>clicker)

Q1: Which of the following pairs of sets (S, T) is a valid cut, and what is its corresponding cut capacity?

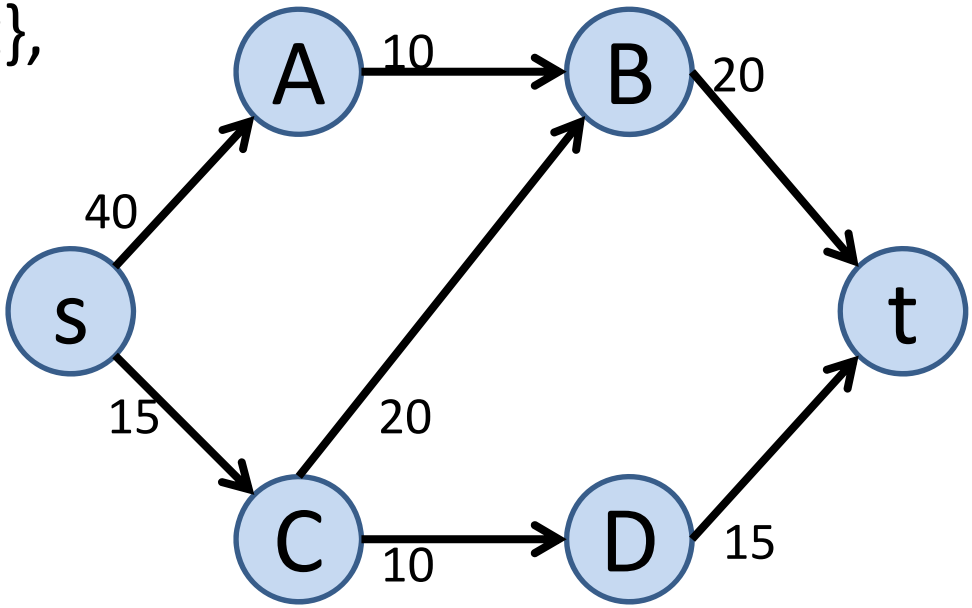
A. S = {t, B, C}, T = {s, A, D}, capacity = 50

B. S = {s, B, C}, T = {A, t}, capacity = 70

C. S = {s, B, C}, T = {A, B, C, t}, capacity = 80

D. S = {s, B, C}, T = {A, D, t}, capacity = 70

E. S = {s, B, C}, T = {A, D, t}, capacity = 80



Q1: Which of the following pairs of sets (S, T) is a valid cut, and what is its corresponding cut capacity?

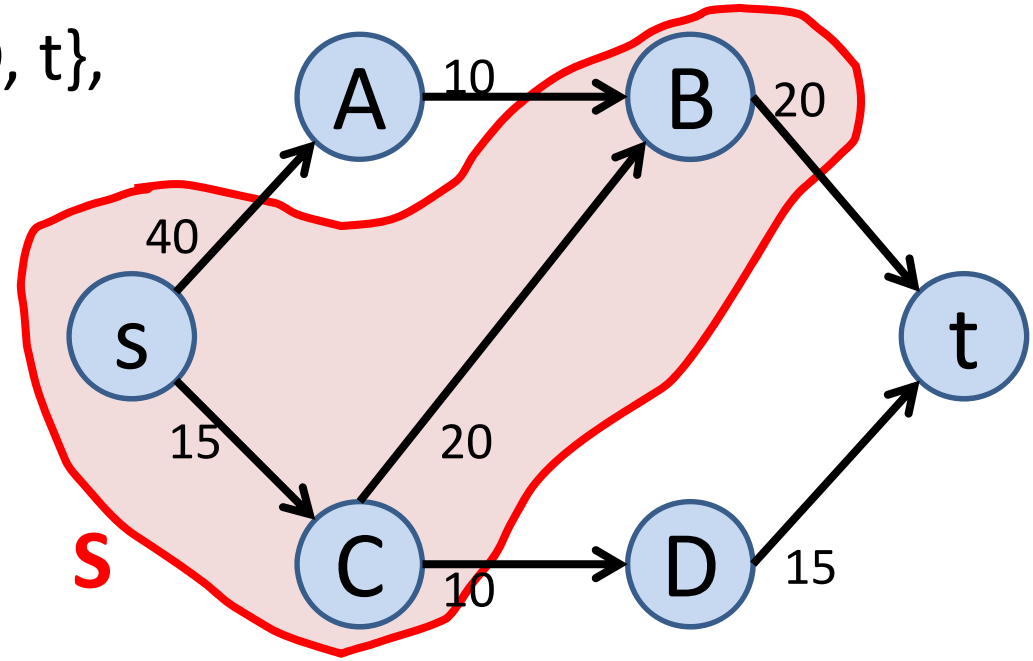
A. S = {t, B, C}, T = {s, A, D}, capacity = 50

B. S = {s, B, C}, T = {A, t}, capacity = 70

C. S = {s, B, C}, T = {A, B, C, t}, capacity = 80

D. S = {s, B, C}, T = {A, D, t}, capacity = 70

E. S = {s, B, C}, T = {A, D, t}, capacity = 80



Q1: Which of the following pairs of sets (S, T) is a valid cut, and what is its corresponding cut capacity?

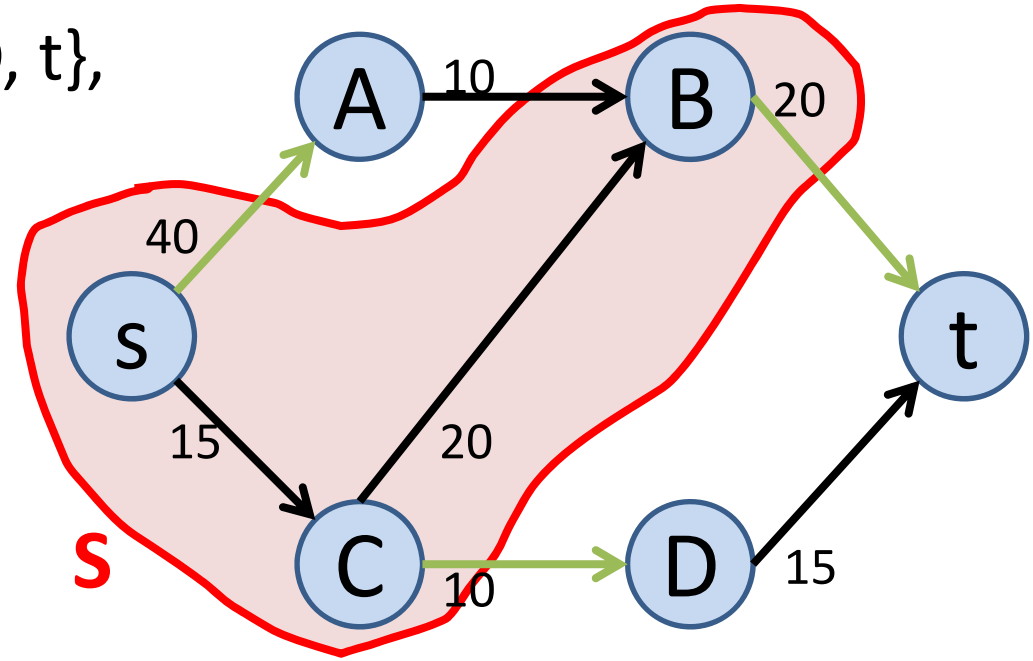
A. S = {t, B, C}, T = {s, A, D}, capacity = 50

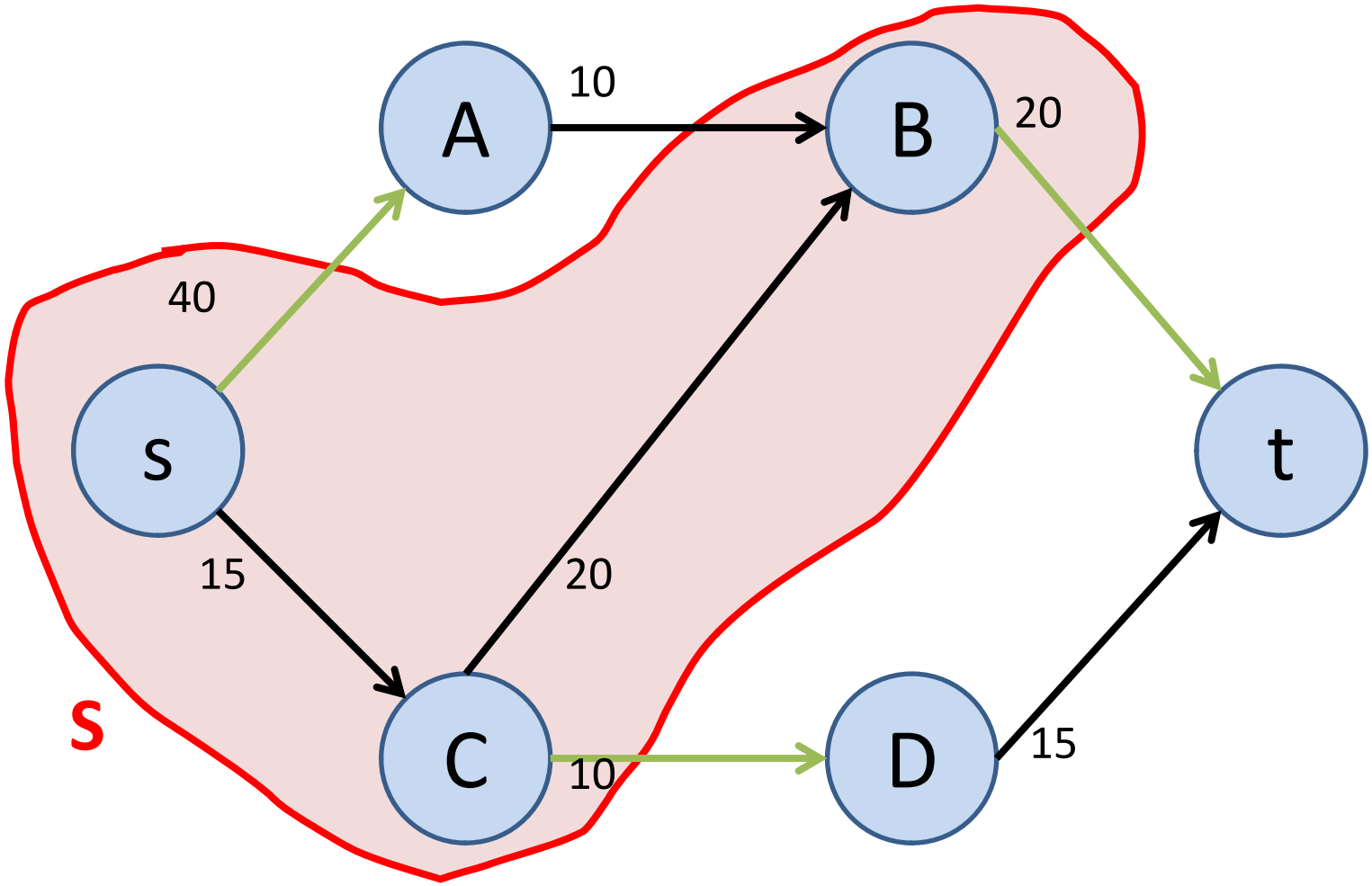
B. S = {s, B, C}, T = {A, t}, capacity = 70

C. E = {s, B, C}, T = {A, B, C, t}, capacity = 80

D. S = {s, B, C}, T = {A, D, t}, capacity = 70

E. S = {s, B, C}, T = {A, D, t}, capacity = 80




A
B
s
t
C
D
10
20
40
15
20
10
15
S

The maximum-flow problem and capacities of cuts

Claim 1

The value of any feasible flow

is less than or equal to the capacity of any cut

Claim 2

The maximum flow value

is less than or equal to the capacity of any cut

Today: Maxflow, continued

More on cuts, flows, and
Ford-Fulkerson's algorithm

The maximum-flow and the minimum-cut problems

Theorem 1

Consider a maxflow problem with input
G = (N, E) and capacities u_{ij} for each (i, j) in E.

If x^* is a feasible flow and (S, T) is a valid cut with the property that

the value of flow of x^* = capacity of the cut (S, T),

then x^* is a maximum flow.

The maximum-flow problem and capacities of cuts

Claim 1

The value of any feasible flow

is less than or equal to the capacity of any cut

Claim 2

The maximum flow value

is less than or equal to the capacity of any cut

The maximum-flow problem and capacities of cuts

Claim 1

The value of any feasible flow

is less than or equal to the capacity of any cut

Claim 2'

The maximum flow value

is less than or equal to the smallest cut capacity

The minimum-cut problem

- Input
  - G = (N, E), a directed graph
  - The node set N contains:
    - A source node, s
    - A sink node, t
  - u_{ij} = edge capacity; $u_{ij} \geq 0$
- Objective: To find a cut with minimum cut capacity.
  - A cut: A partition of N into sets (S, T), where S contains the source and T contains the sink
  - Capacity of the cut (S, T) is

    the sum of capacities of edges that go from S to T:

$$\sum_{\substack{(i,j)\in E, \text{s.t.} \\ i\in S,\, j\in T}} u_{ij}$$

The maximum-flow and the minimum-cut problems

Maxflow

- Input
  - G = (N, E), a directed graph
  - The node set N contains:
    - A source node, s
    - A sink node, t
  - u_{ij} = edge capacity; $u_{ij} \geq 0$
- Objective:
  To maximize net flow into t, subject to constraints:
  - Capacity constraints
  - Flow conservation constraints:

    Net flow out of node i = 0

    (for each node i in N, except s and t)

Mincut

- Input
  - G = (N, E), a directed graph
  - The node set N contains:
    - A source node, s
    - A sink node, t
  - u_{ij} = edge capacity; $u_{ij} \geq 0$
- Objective: To find a cut with minimum cut capacity.
  - A cut: A partition of N into sets (S, T), where S contains the source and T contains the sink
  - Capacity of the cut (S, T) is

    the sum of capacities of edges that go from S to T:

$$\sum_{\substack{(i,j)\in E, \text{s.t.} \\ i\in S,\, j\in T}} u_{ij}$$

The maximum-flow and the minimum-cut problems

Claim 2’

The maximum flow value

is less than or equal to the minimum cut capacity

The maximum-flow and the minimum-cut problems

Theorem 2

The maximum flow value

is equal to the minimum cut capacity

The maxflow-mincut theorem

Theorem 2

The maximum flow value

is equal to the minimum cut capacity

Proof.

…

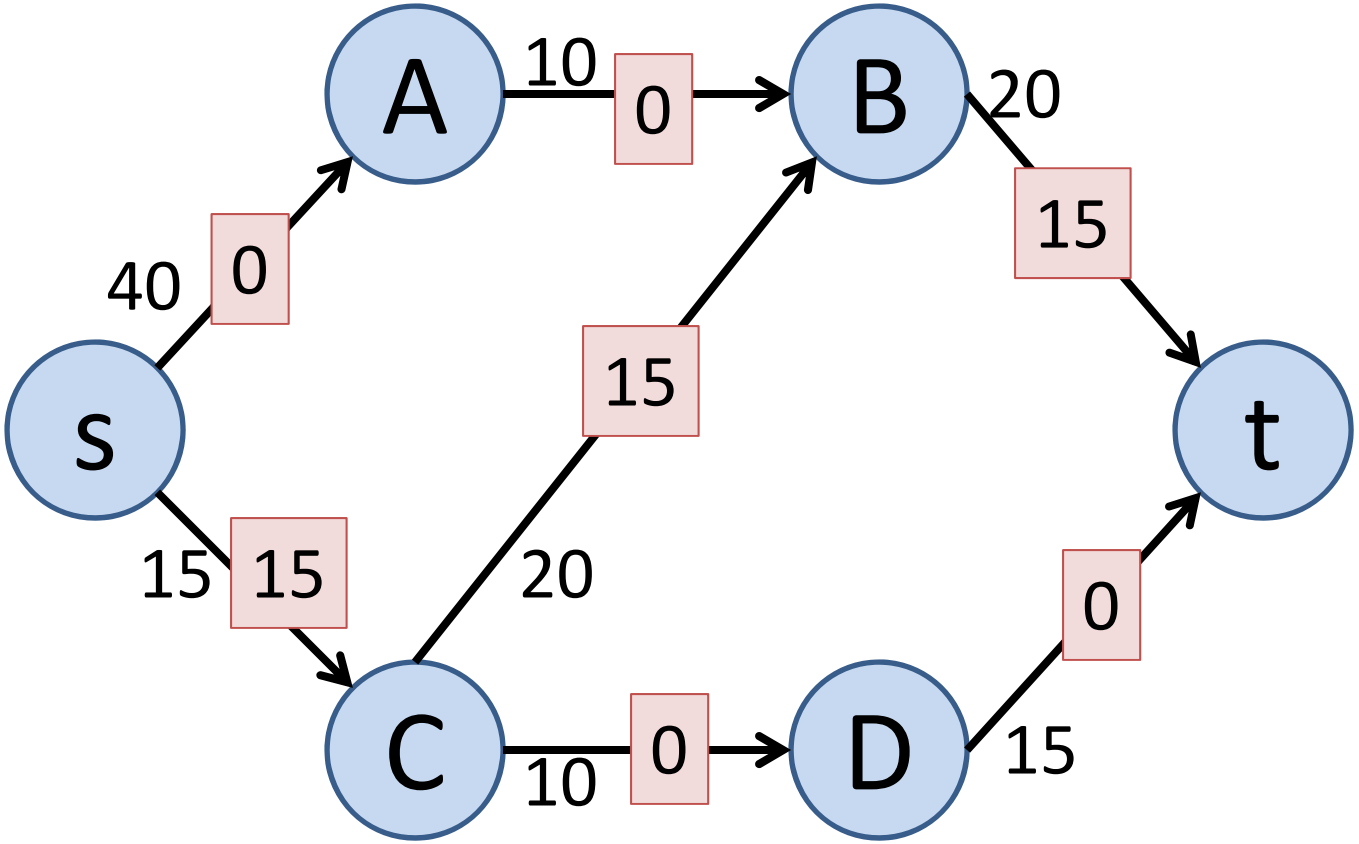
The Ford-Fulkerson method

0. Find an initial feasible solution (flow)
1. Consider the current solution, x.
2. If x is not optimal,
   then find a way to improve the flow.
   Otherwise, if x is optimal, we're done!

The Ford-Fulkerson method

0. Find an initial feasible solution (flow)
1. Consider the current solution, x.
2. If x is not optimal,
   then find a way to improve the flow.
   Otherwise, if x is optimal, we're done!

Example: An initial feasible flow



The Ford-Fulkerson method

0. Find an initial feasible solution (flow)
1. Consider the current solution, x.
2. If x is not optimal,
   then find a way to improve the flow.
   Otherwise, if x is optimal, we're done!

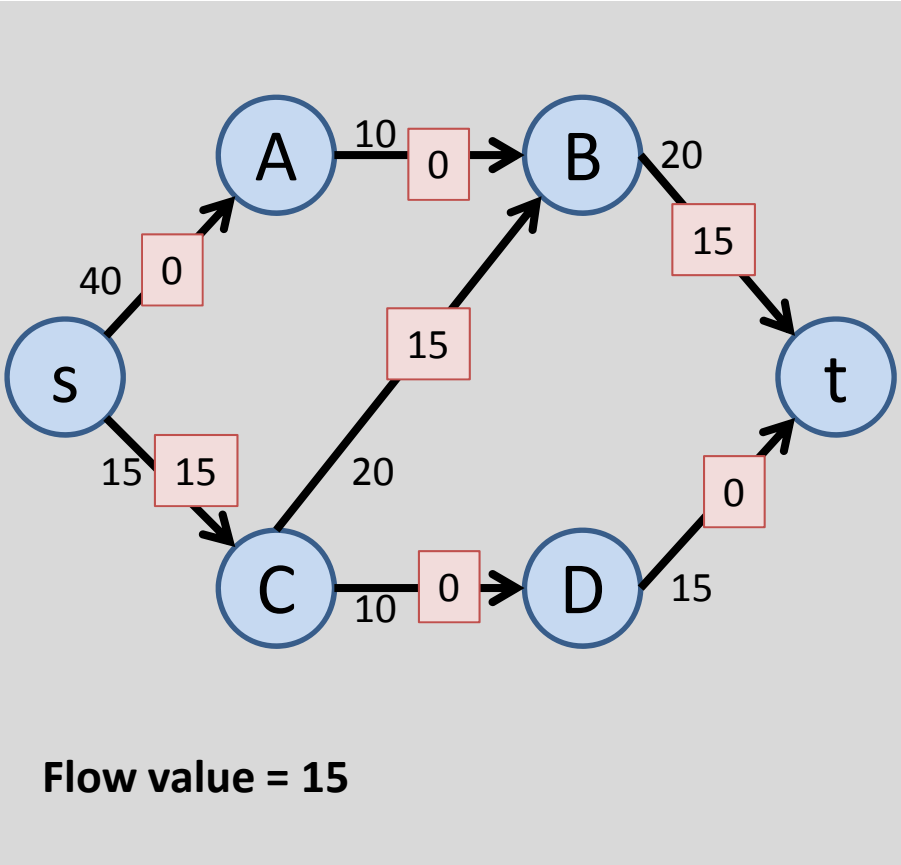
The Ford-Fulkerson method

0. Find an initial feasible solution (flow)
1. Consider the current solution, x.
2. If x is not optimal,
   then find a way to improve the flow.

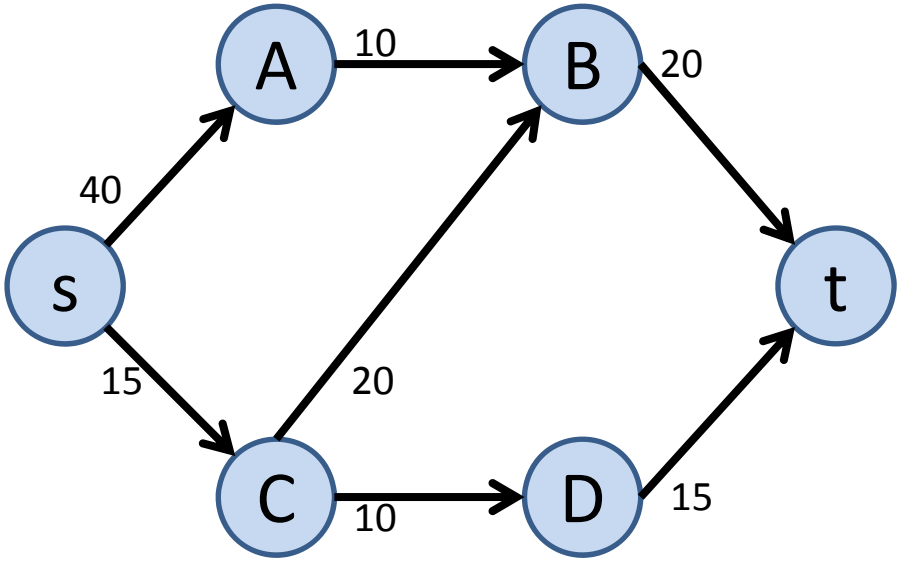
   Otherwise, if x is optimal, we're done!

Finding a way to improve flow

Current solution, $x^{(1)}$

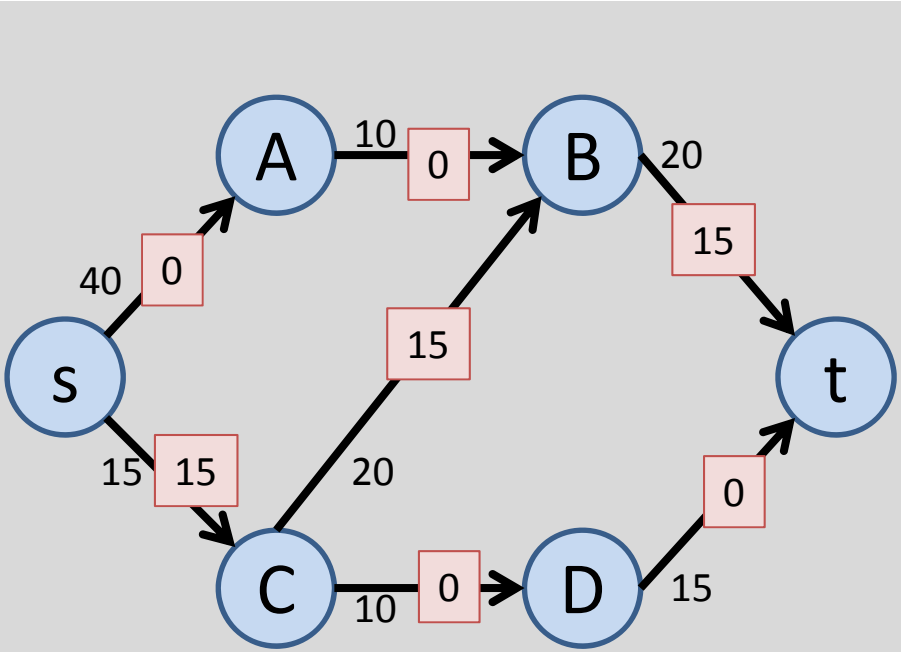


Scratchwork graph for $x^{(1)}$

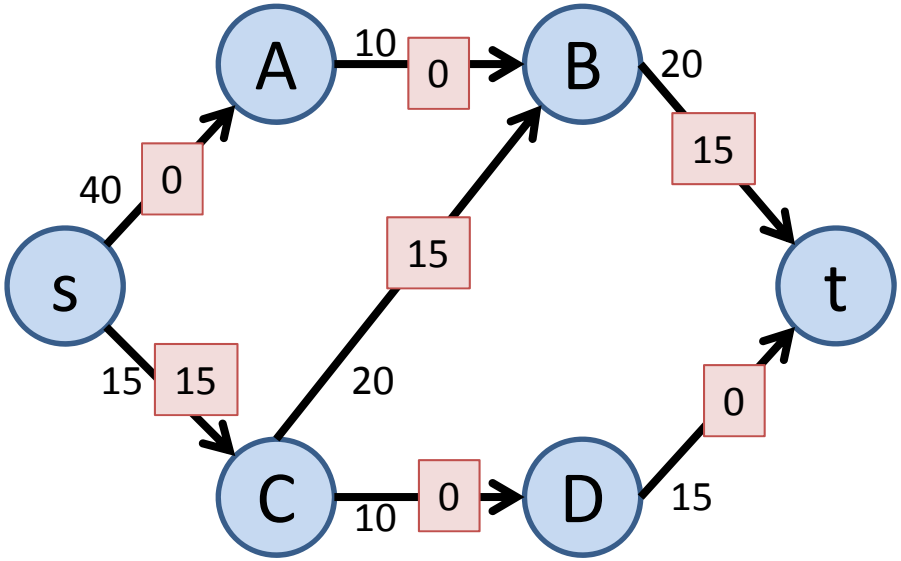


Finding a way to improve flow

Current solution, $x^{(1)}$

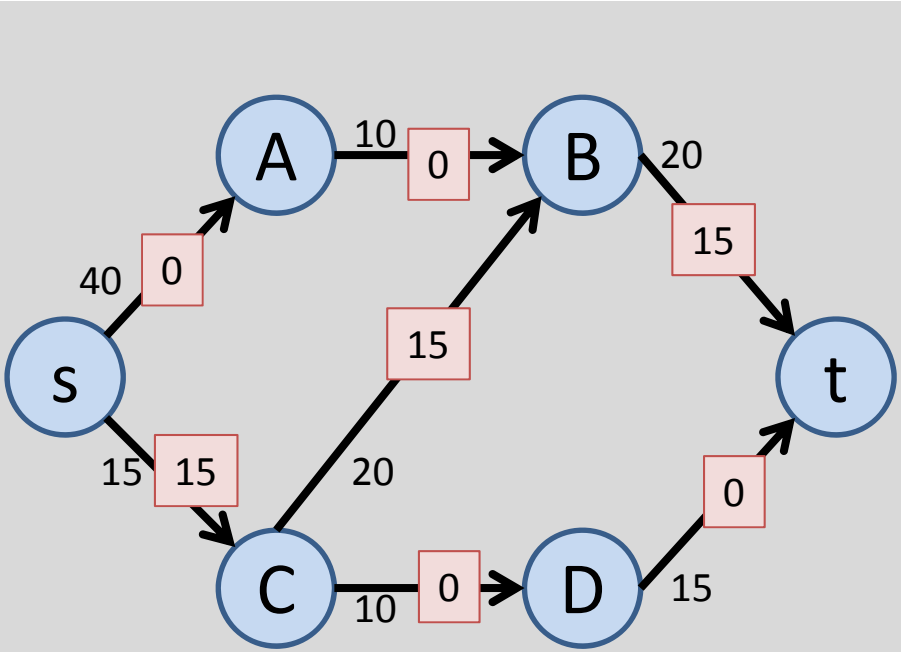


Scratchwork graph for $x^{(1)}$



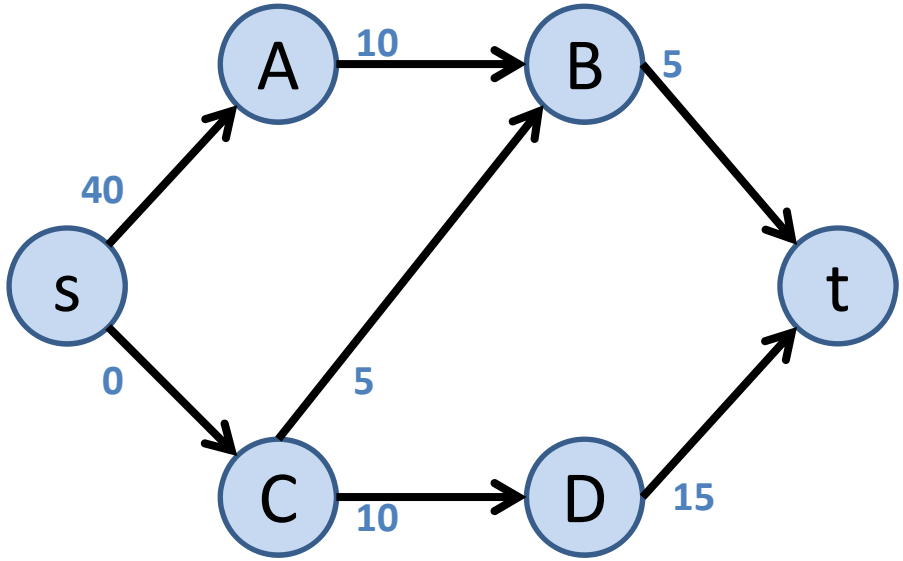
Finding a way to improve flow

Current solution, $x^{(1)}$



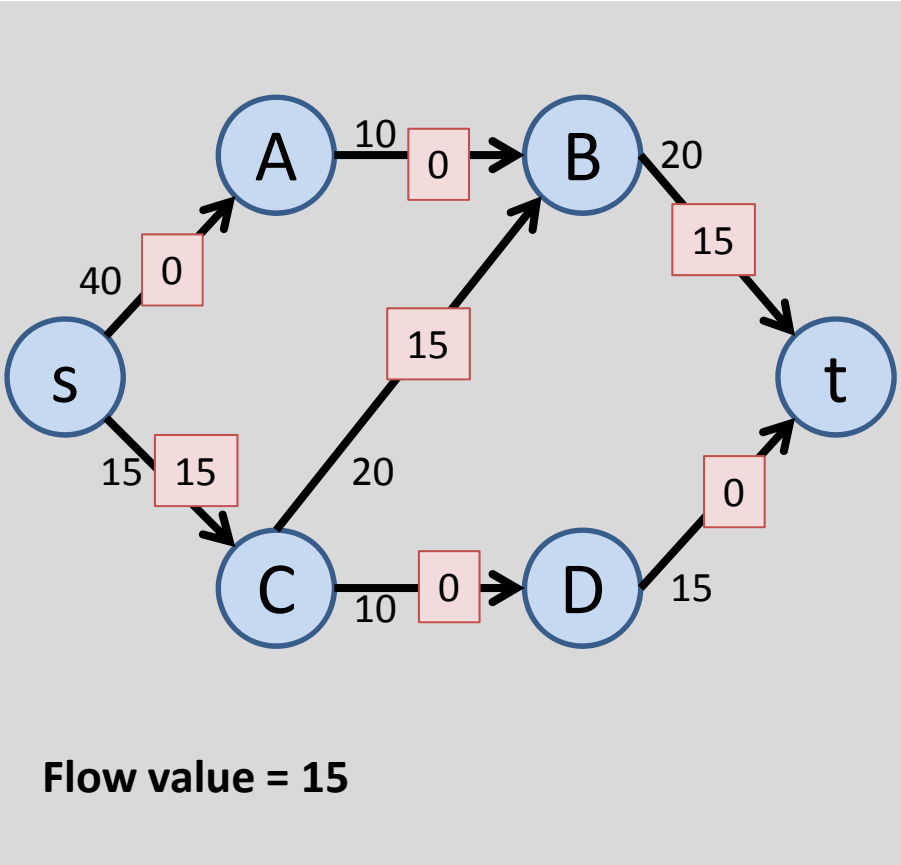
Flow value = 15

Scratchwork graph for $x^{(1)}$

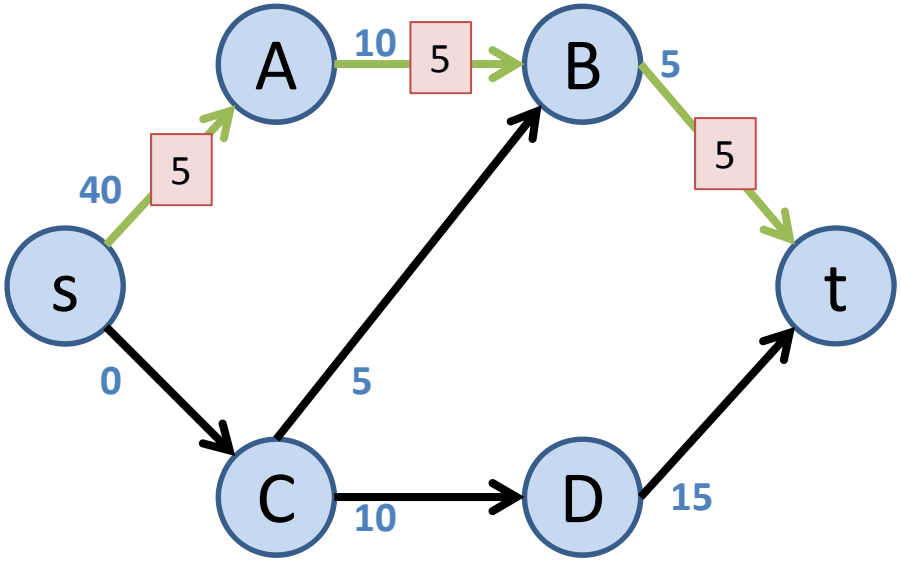


Finding a way to improve flow

Current solution, $x^{(1)}$

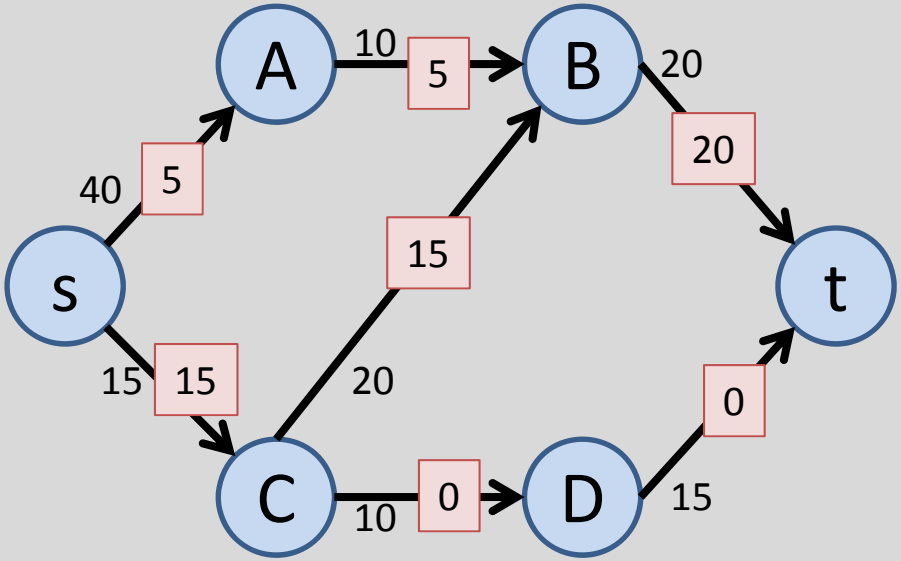


Scratchwork graph for $x^{(1)}$



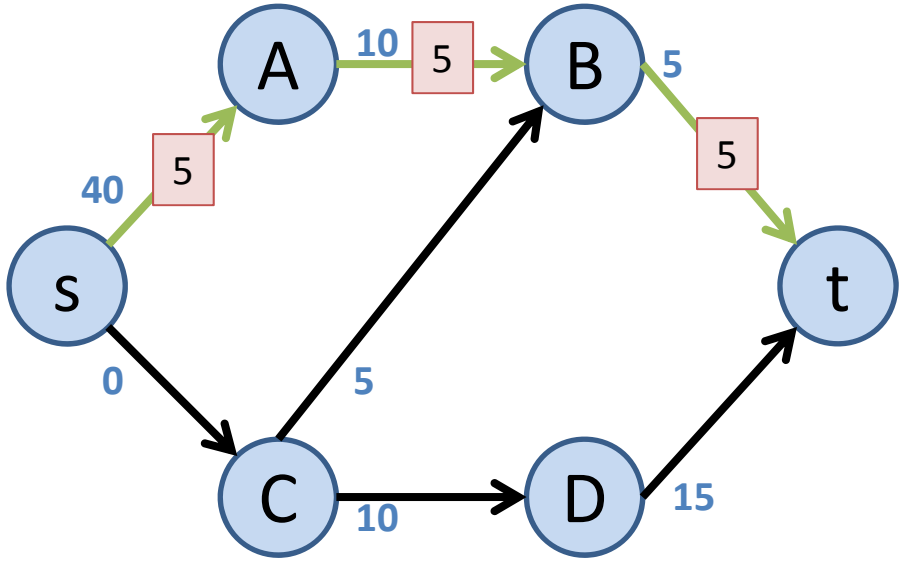
Finding a way to improve flow

Updated solution, $x^{(2)}$



Flow value = 20

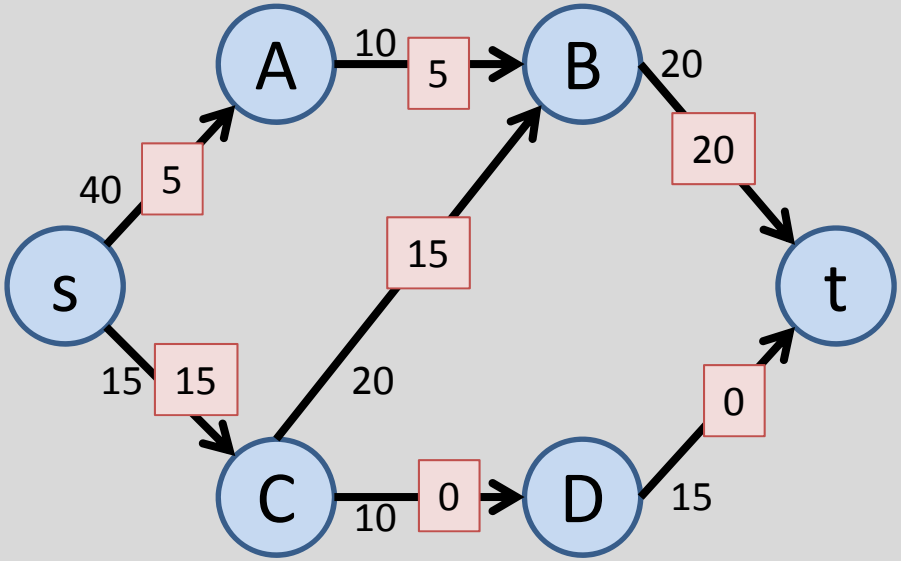
Scratchwork graph for $x^{(1)}$



An augmenting path of capacity 5!

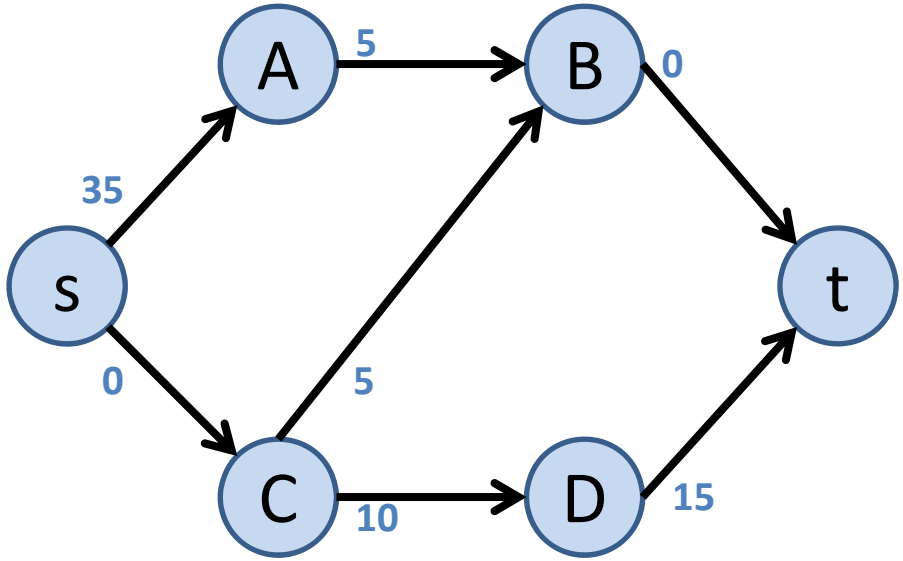
One more iteration to improve flow?

Current solution, $x^{(2)}$



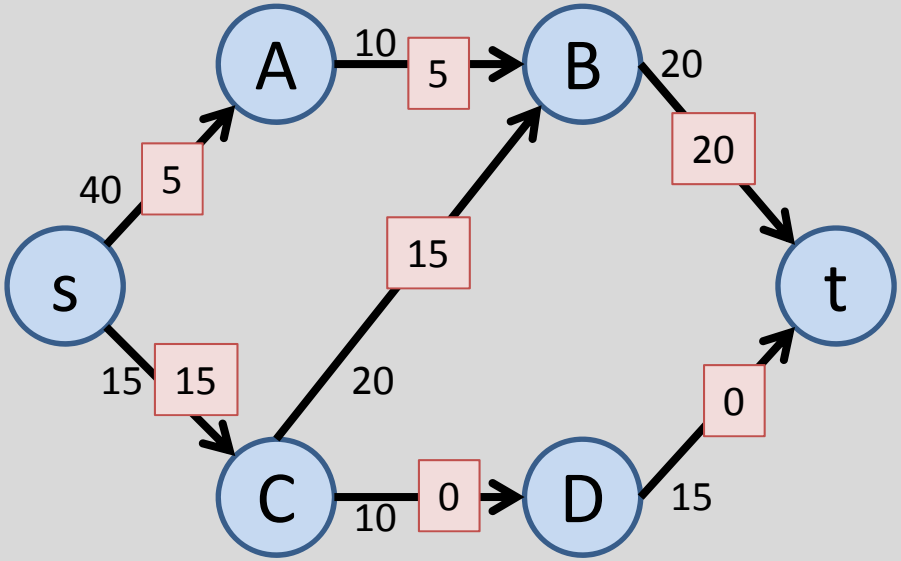
Flow value = 20

Scratchwork graph for $x^{(2)}$



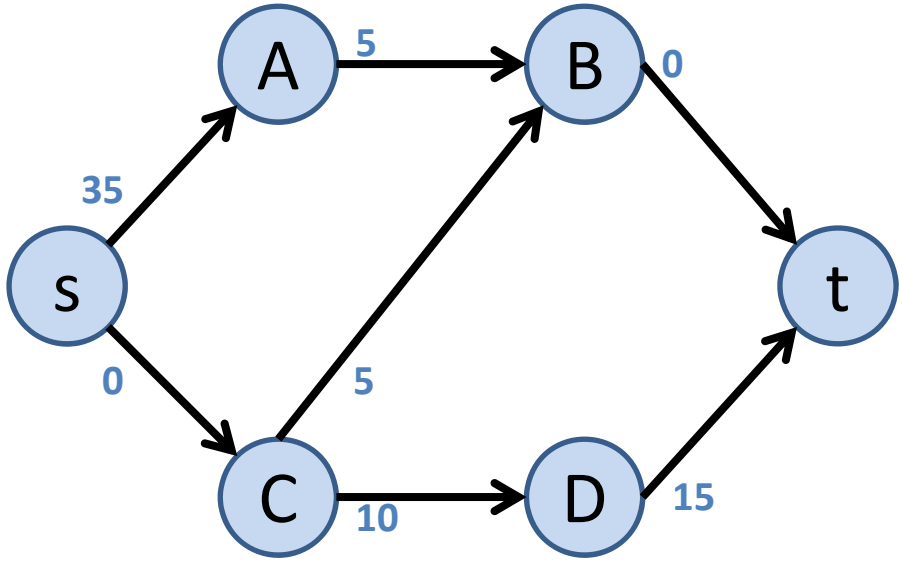
One more iteration to improve flow?

Current solution, $x^{(2)}$



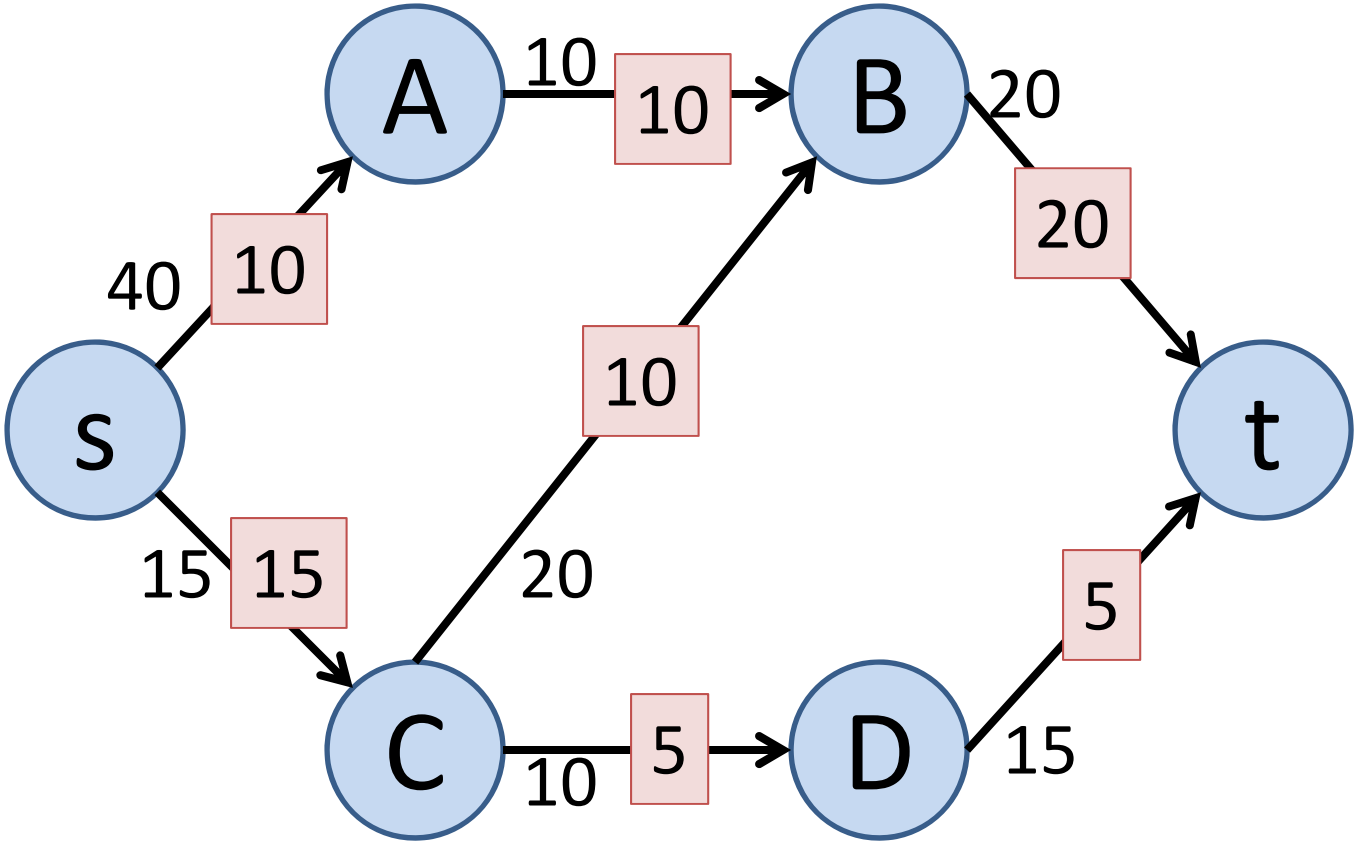
Flow value = 20

Scratchwork graph for $x^{(2)}$



No way to improve the current solution?
Is $x^{(2)}$ an optimal flow?

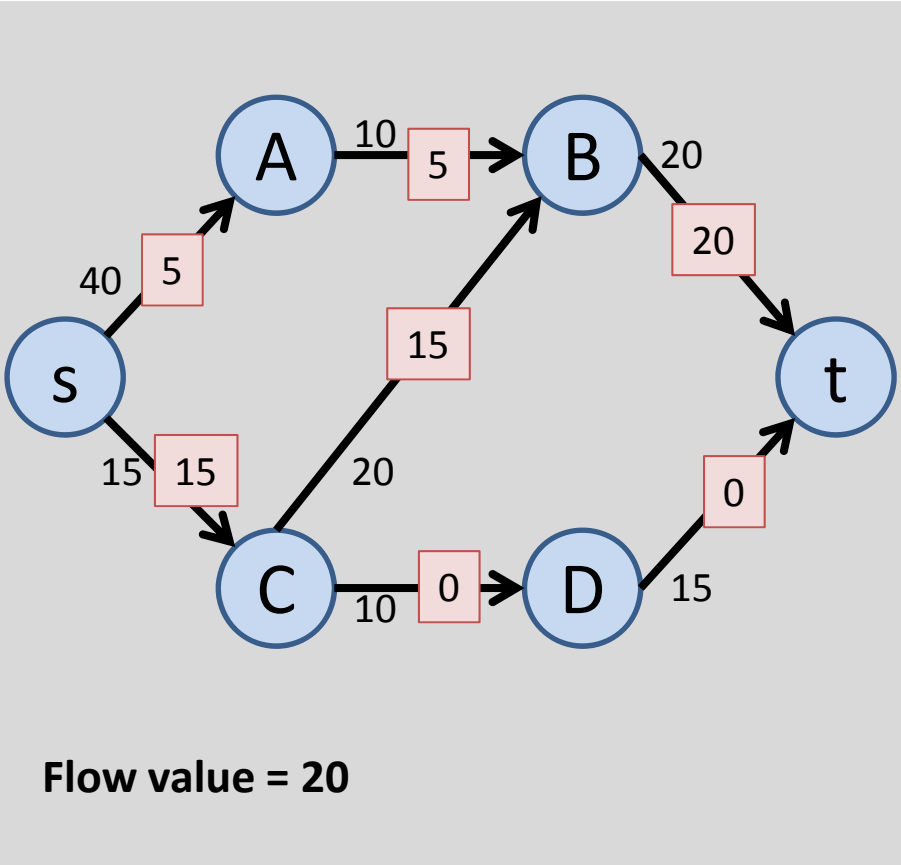
Nope! A better flow:



Flow value = 25

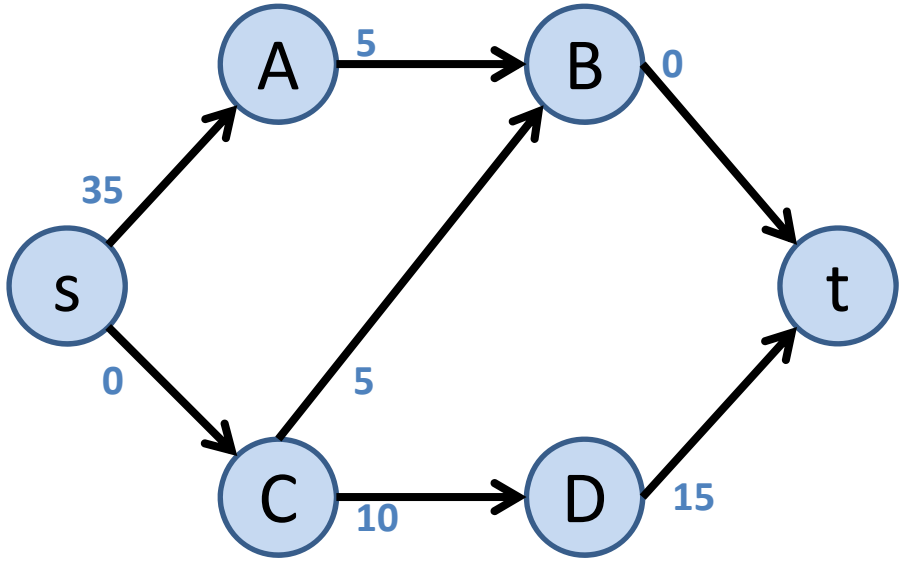
Finding a way to improve flow

Current solution, $x^{(2)}$



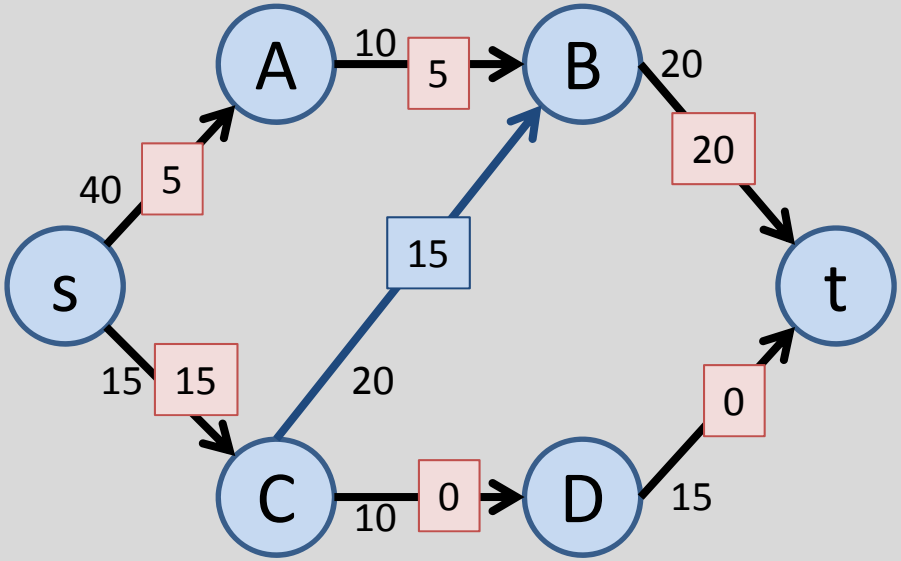
Flow value = 20

Scratchwork graph for $x^{(2)}$

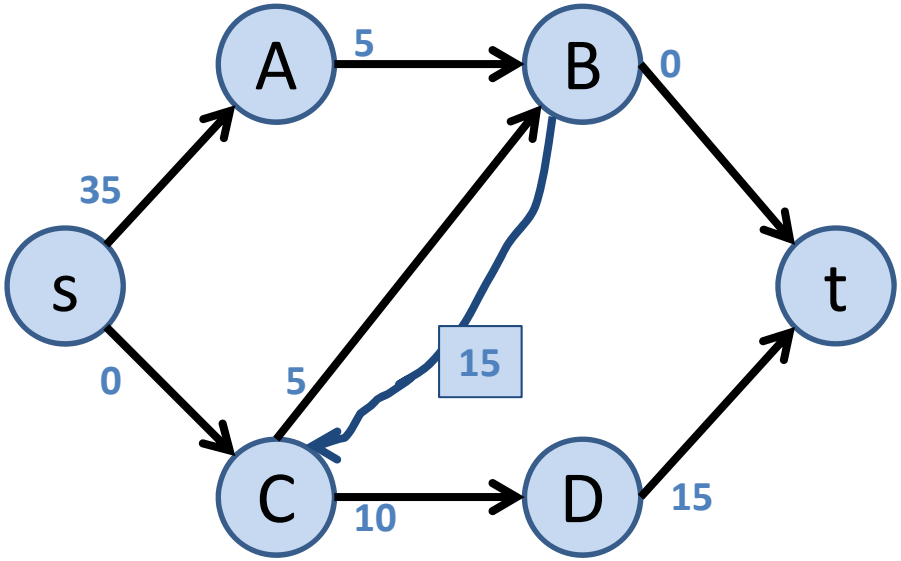


Finding a way to improve flow, v.2! By building a Residual Graph

Current solution, $x^{(2)}$

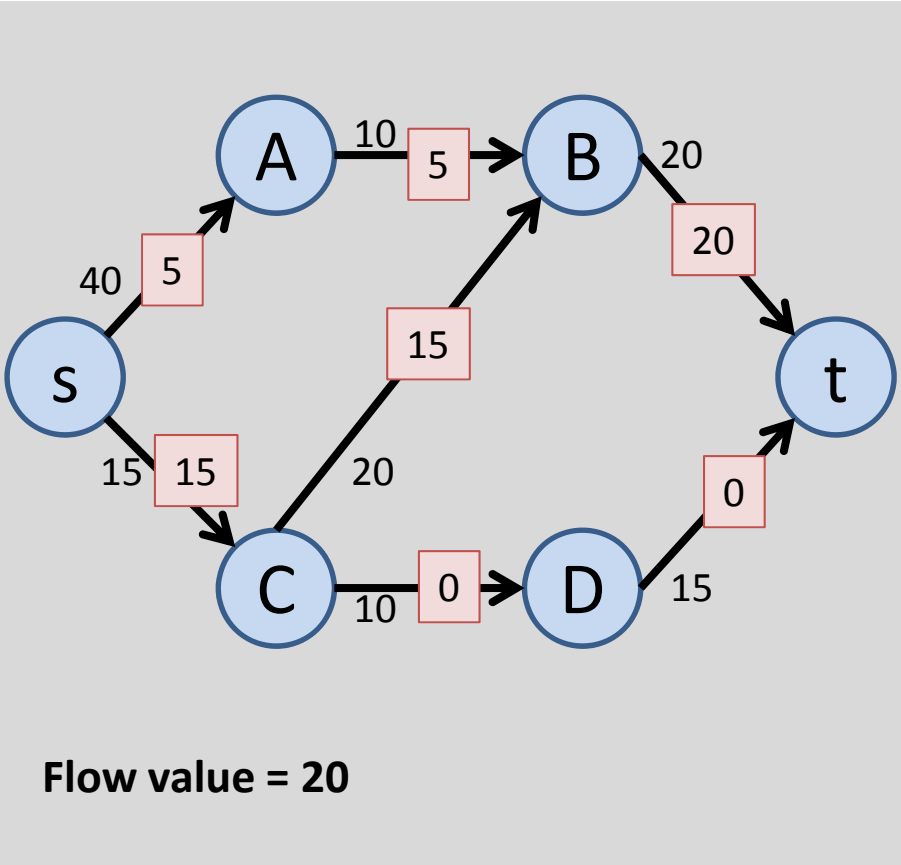


Residual graph for $x^{(2)}$, $G_{x(2)}$



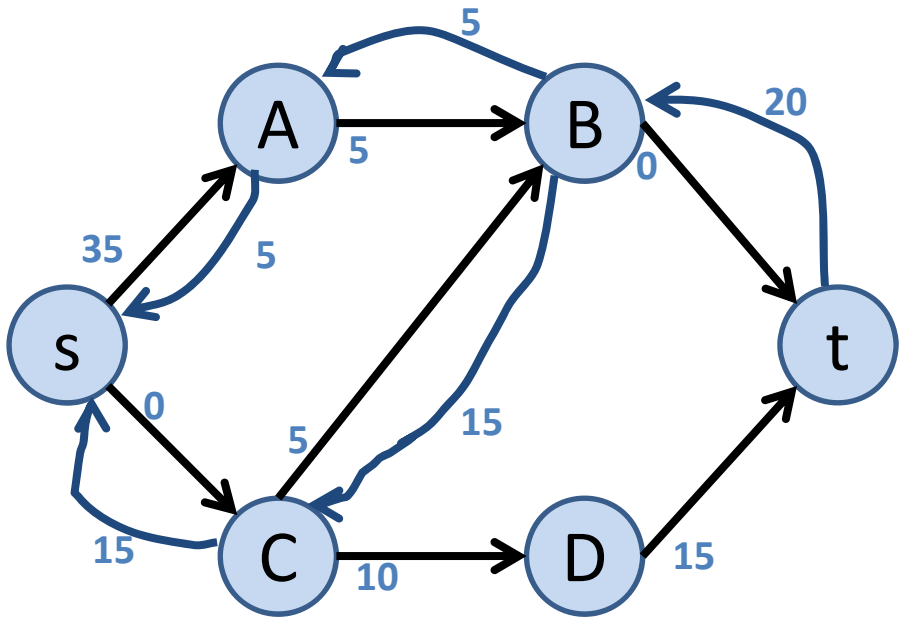
Building a Residual Graph

Current solution, $x^{(2)}$



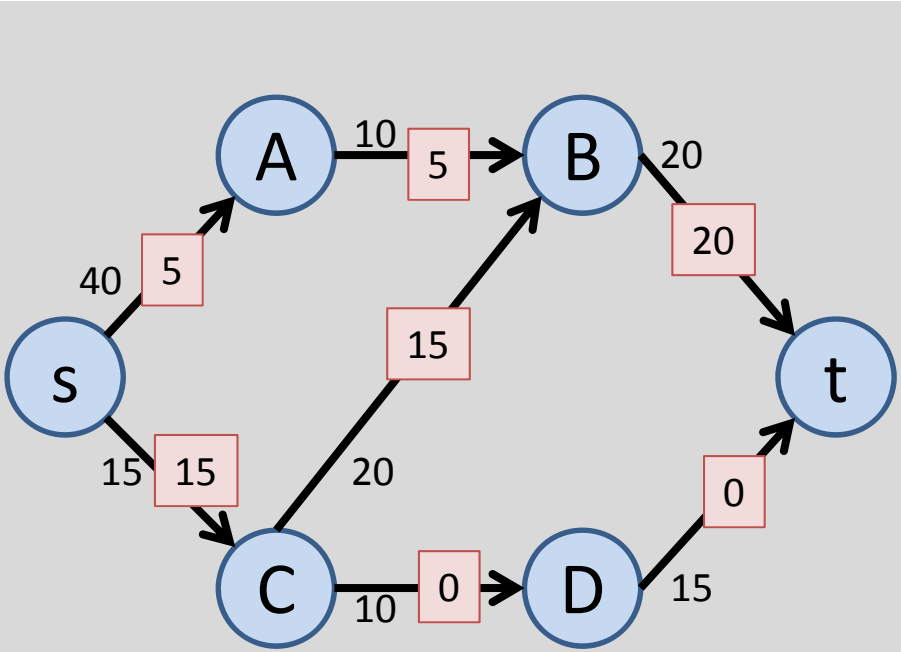
Flow value = 20

Residual graph for $x^{(2)}$, $G_{x(2)}$

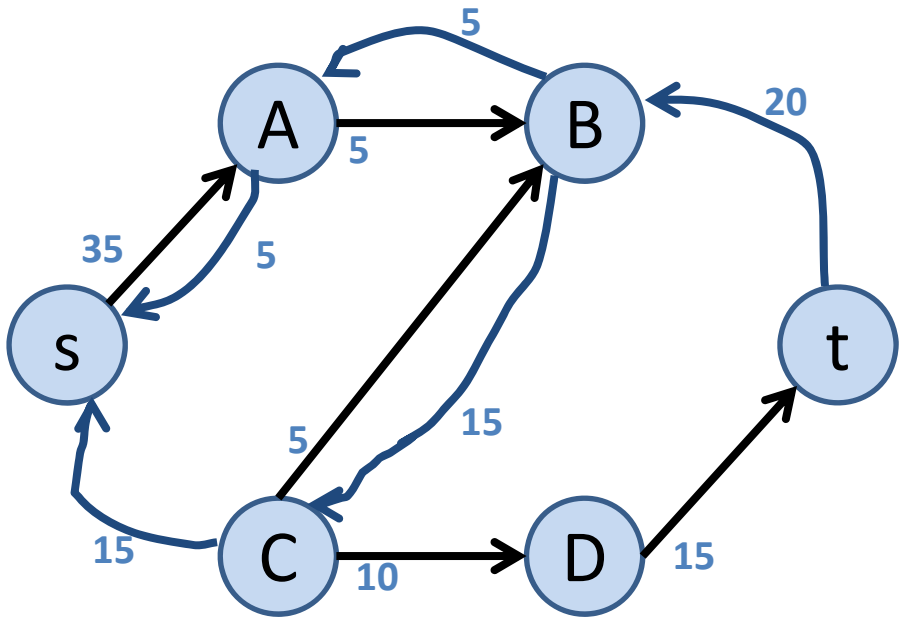


Building a Residual Graph

Current solution, $x^{(2)}$

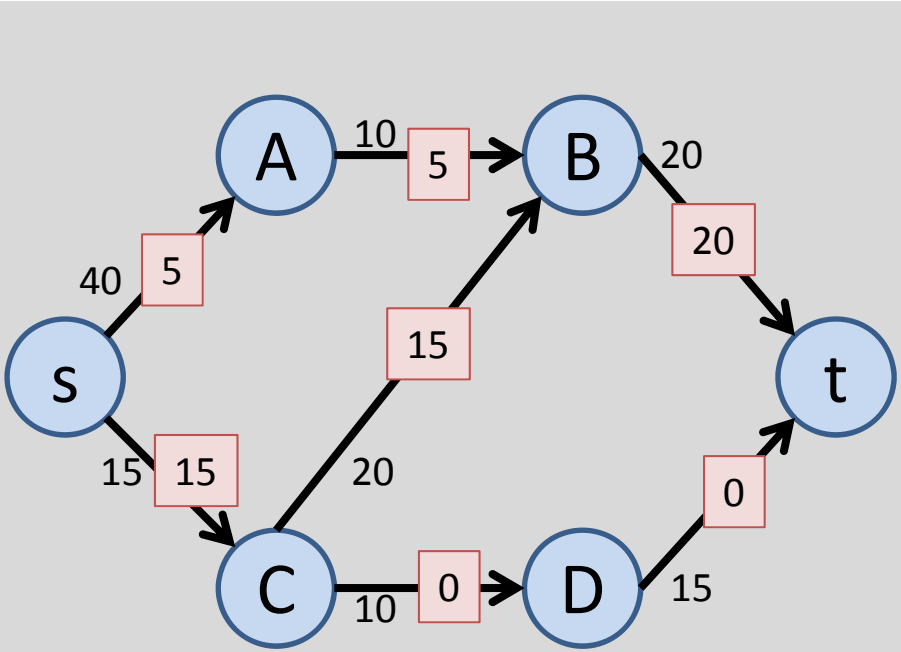


Residual graph for $x^{(2)}$, $G_{x(2)}$



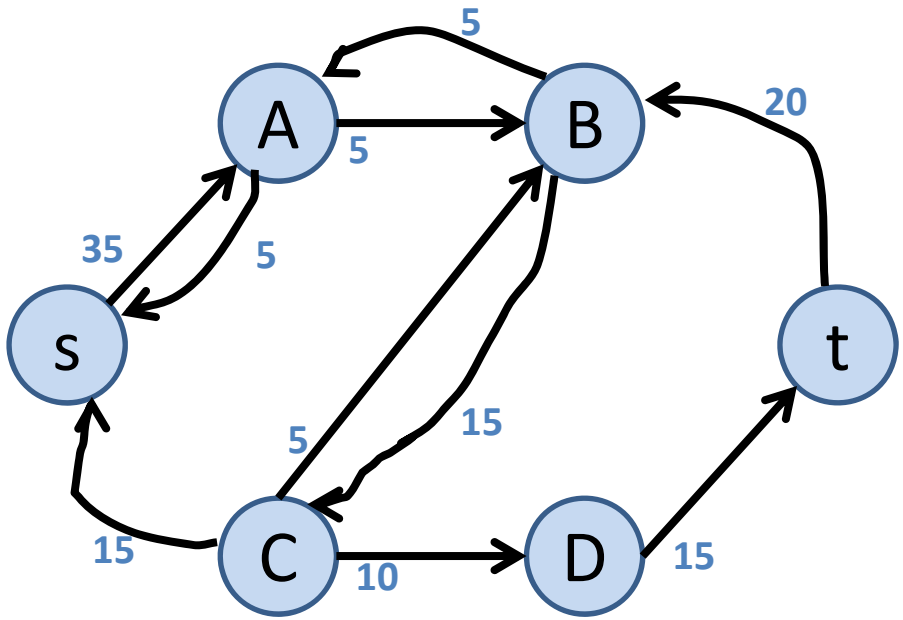
Building a Residual Graph

Current solution, $x^{(2)}$



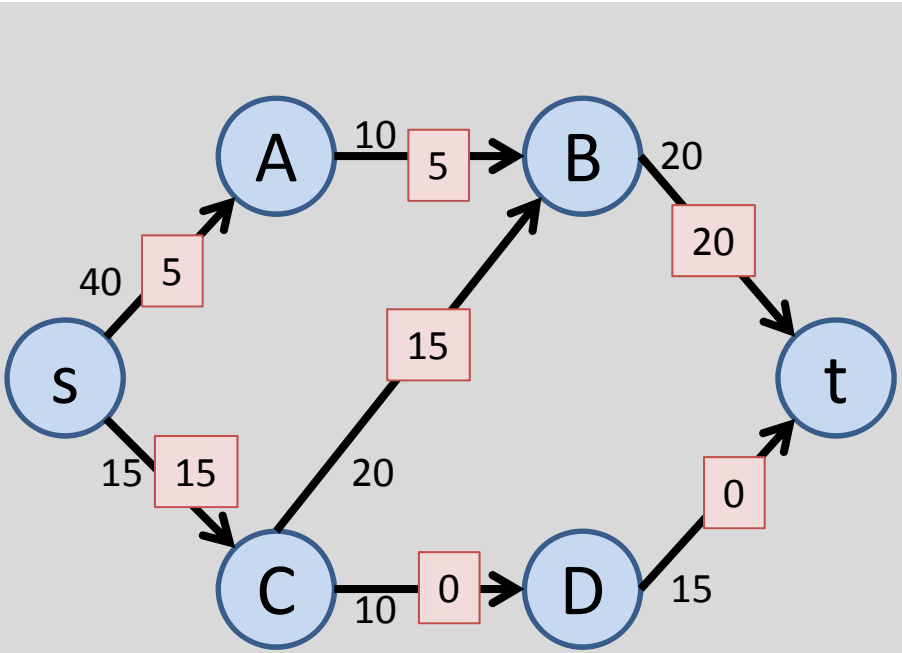
Flow value = 20

Residual graph for $x^{(2)}$, $G_{x(2)}$



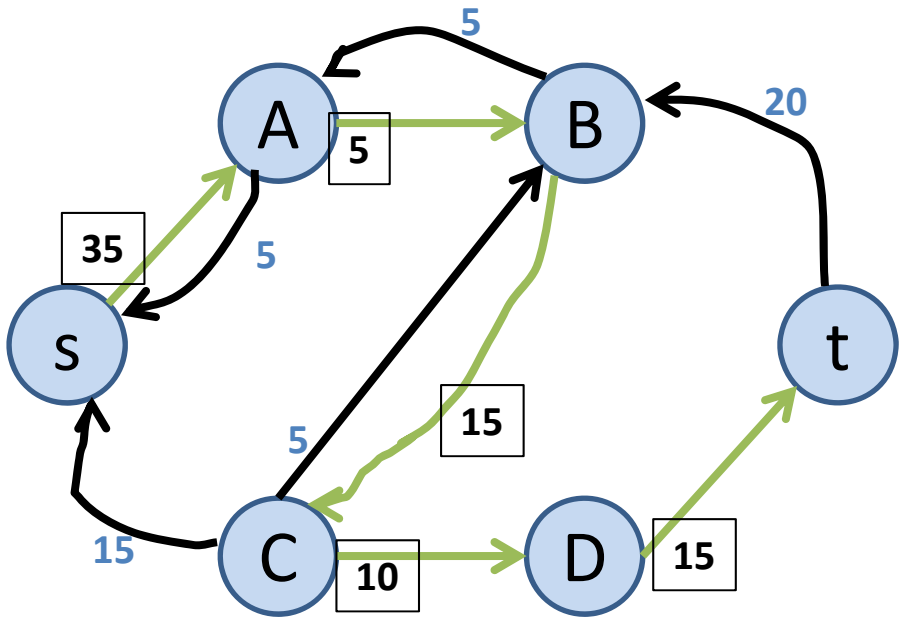
Then, finding an augmenting path in the residual graph

Current solution, $x^{(2)}$



Flow value = 20

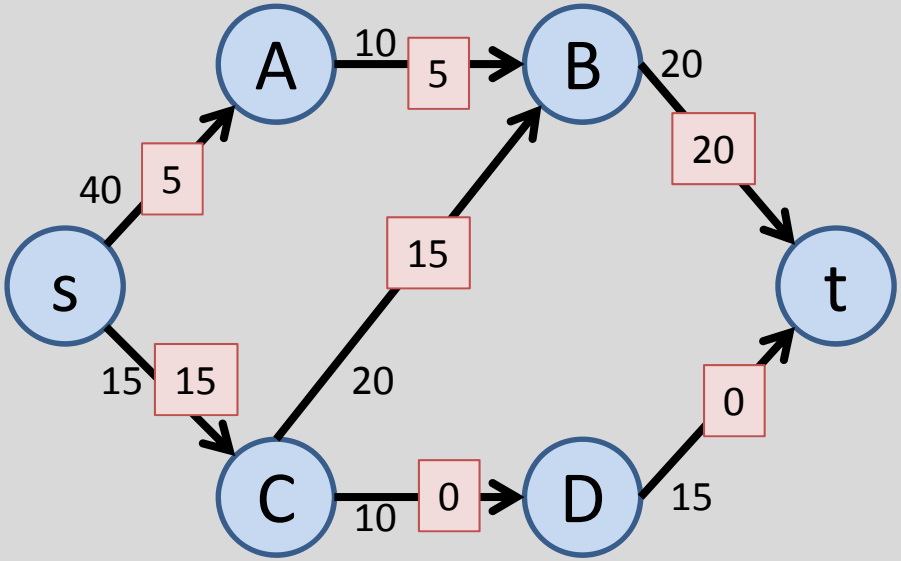
Residual graph for $x^{(2)}$, $G_{x(2)}$



An augmenting path!

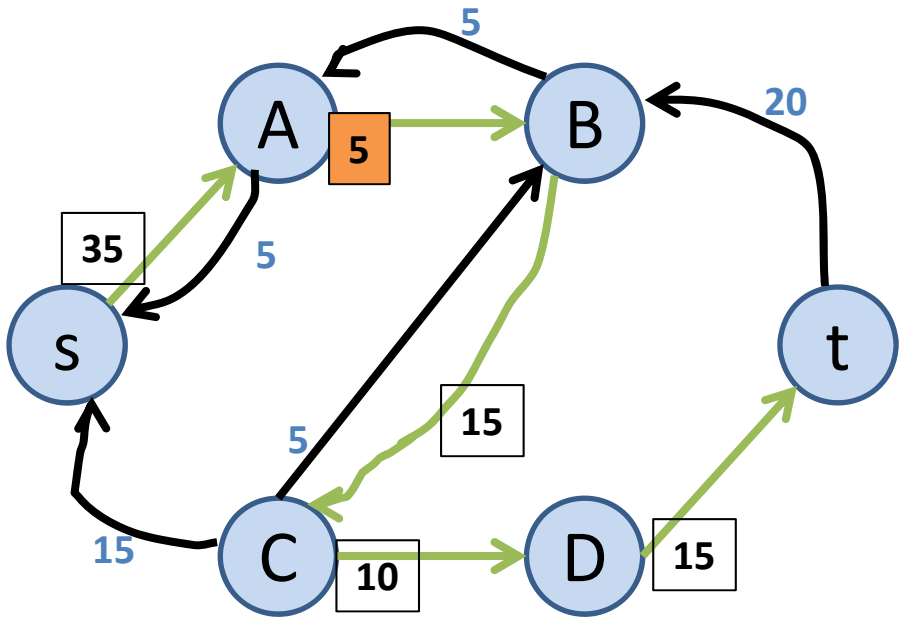
Then, finding an augmenting path in the residual graph

Current solution, $x^{(2)}$



Flow value = 20

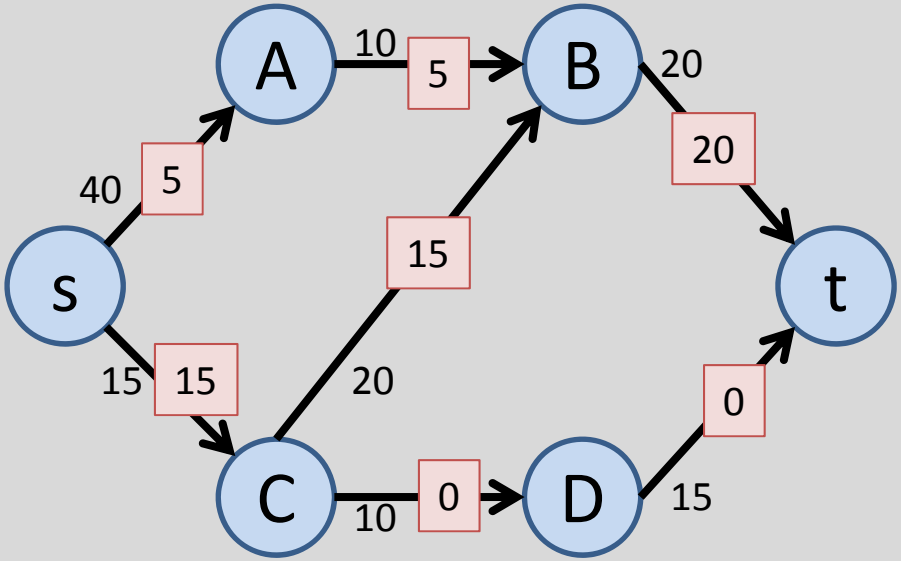
Residual graph for $x^{(2)}$, $G_{x(2)}$



An augmenting path of capacity 5!

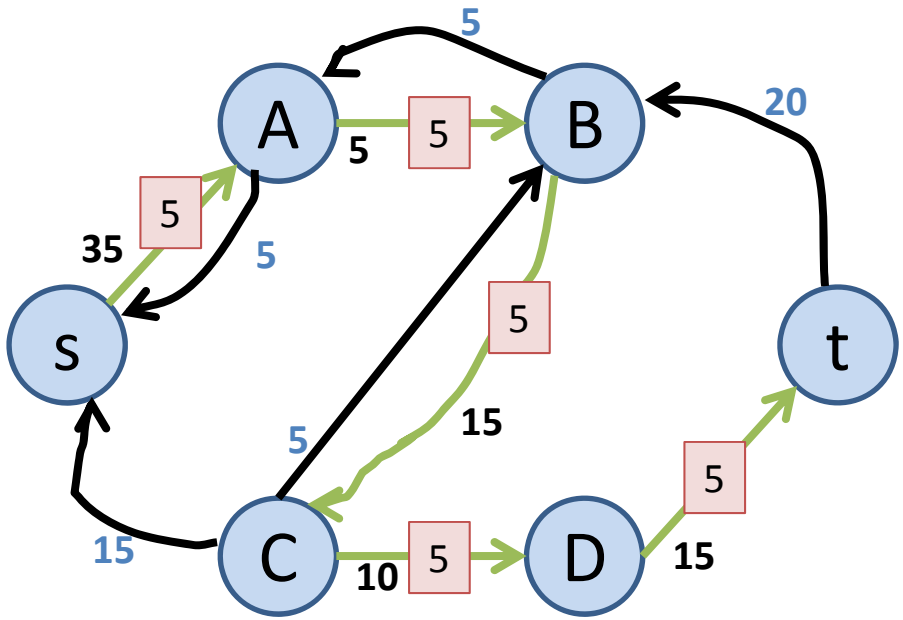
Then, finding an augmenting path in the residual graph

Current solution, $x^{(2)}$



Flow value = 20

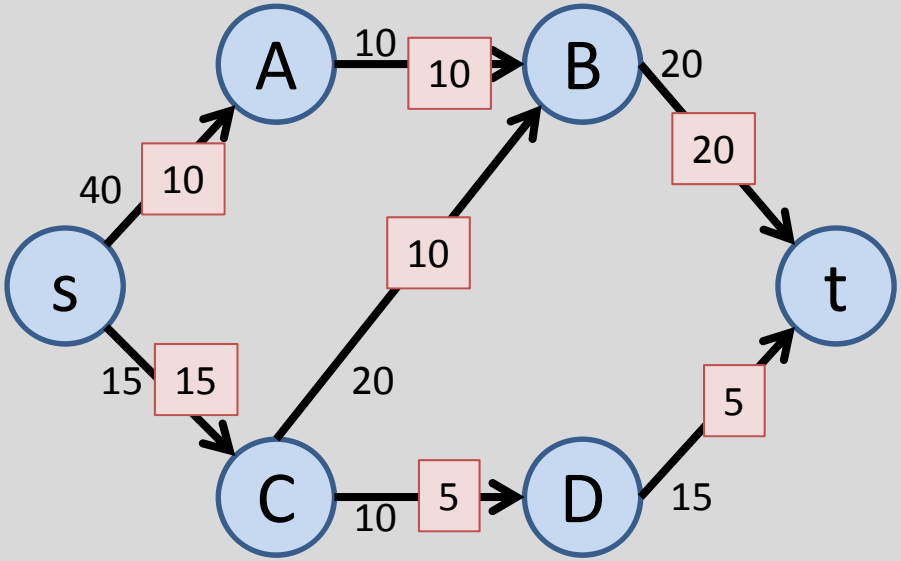
Residual graph for $x^{(2)}$, $G_{x(2)}$



An augmenting path of capacity 5!

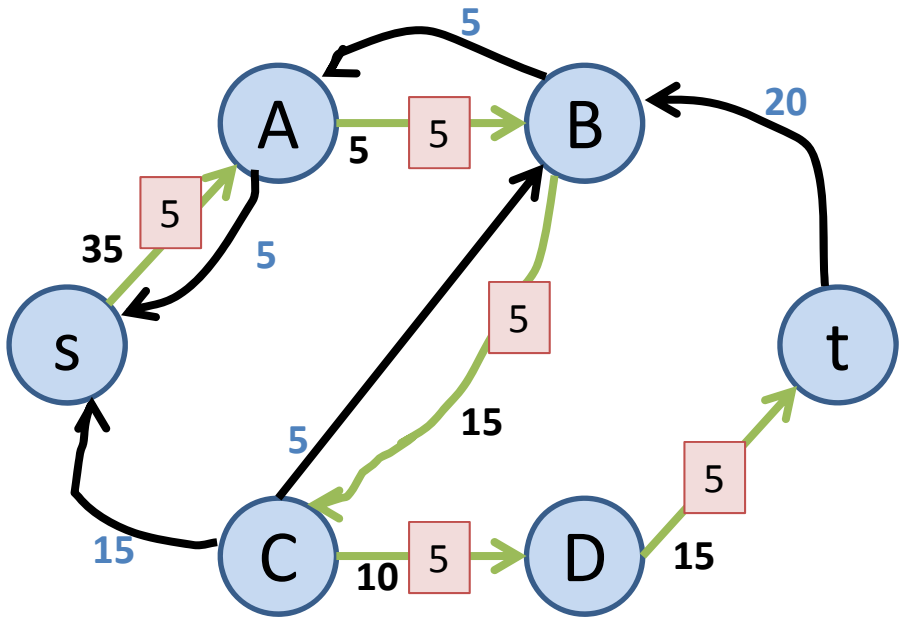
Finally, using the augmenting path to obtain a better solution

Updated solution, $x^{(3)}$



Flow value = 25

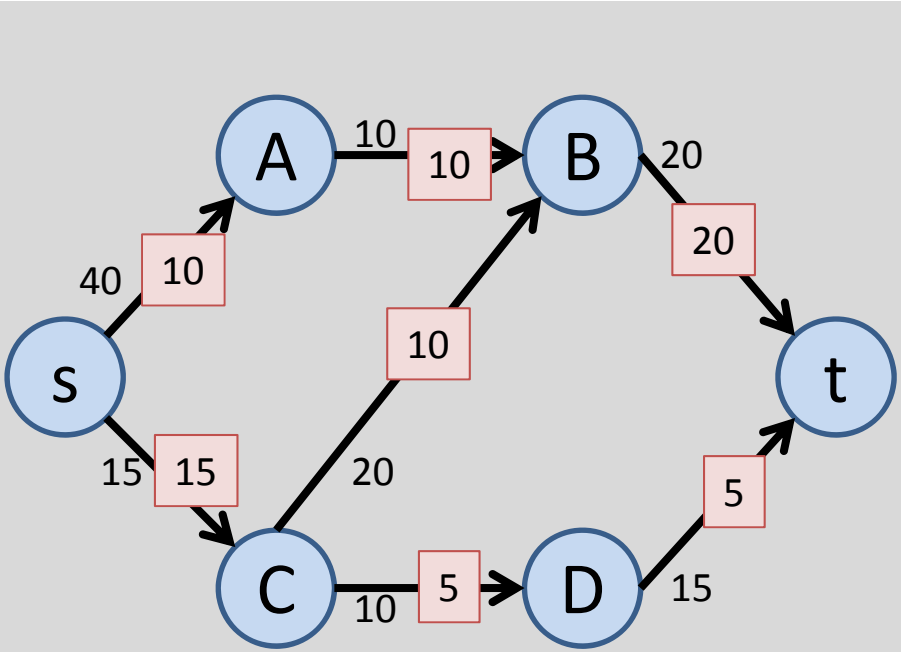
Residual graph for $x^{(2)}$, $G_{x(2)}$



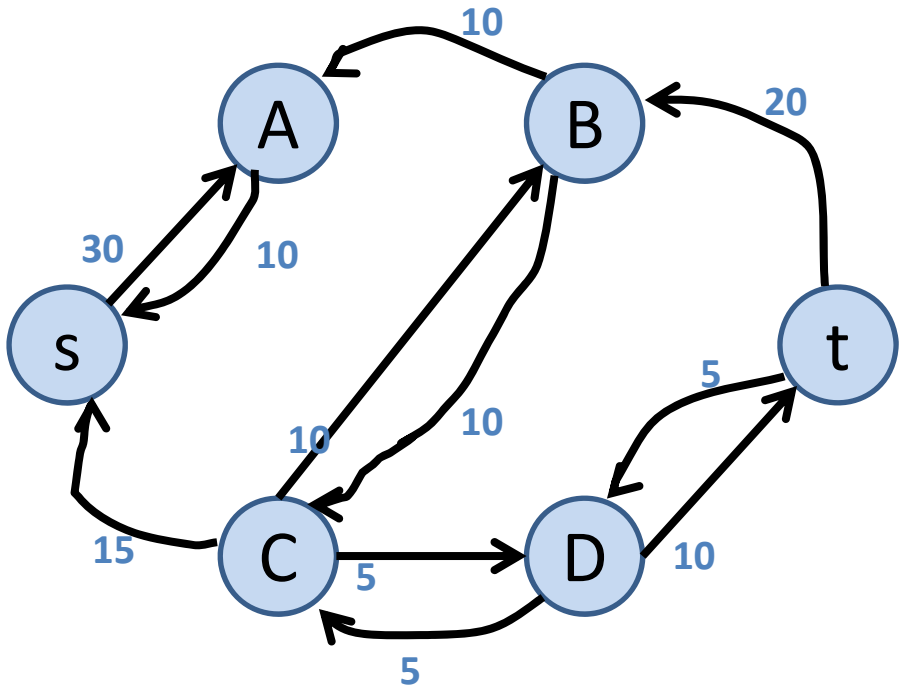
An augmenting path of capacity 5!

To proceed, repeat the steps!

Updated solution, $x^{(3)}$



Residual graph for $x^{(3)}$, $G_{x(3)}$



The Ford-Fulkerson method

0. Find an initial feasible solution (flow)
1. Consider the current solution, x.
2. If x is not optimal,
   then find a way to improve the flow.
   Otherwise, if x is optimal, we're done!

The Ford-Fulkerson method

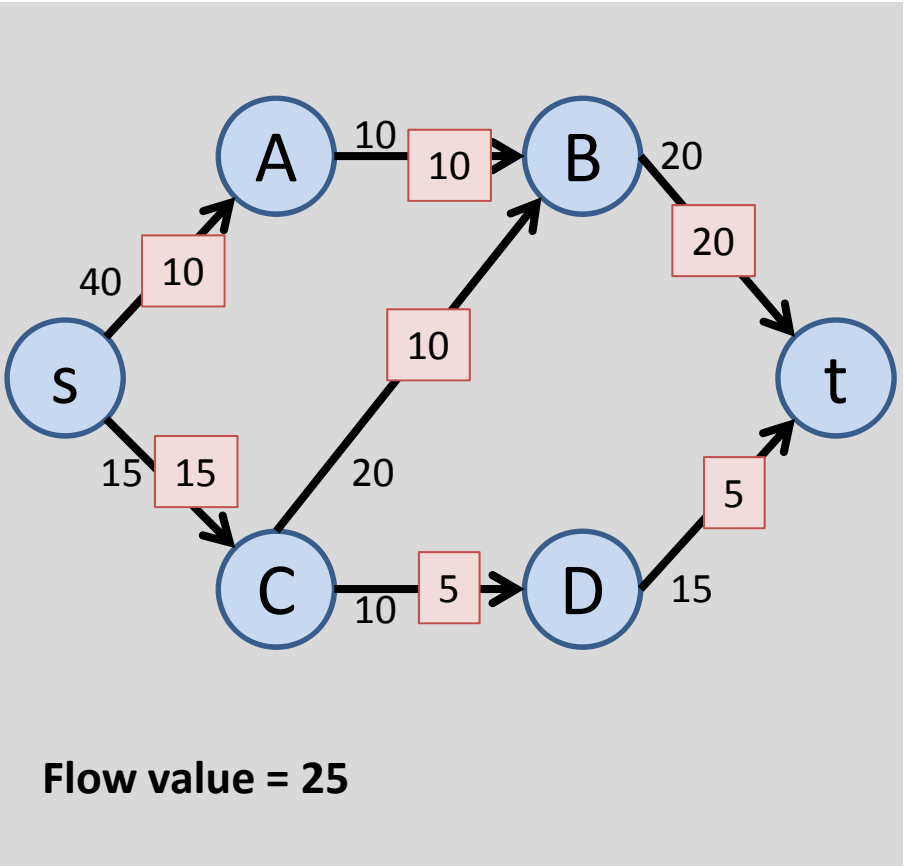
0. Find an initial feasible solution (flow)
1. Consider the current solution, x.
   Construct a residual graph, G_x.
2. Try to find an augmenting path in G_x.
   If there is an augmenting path,
      then use this path to improve the flow.
   If there is no augmenting path,
      then x is optimal. We're done!

The Ford-Fulkerson method

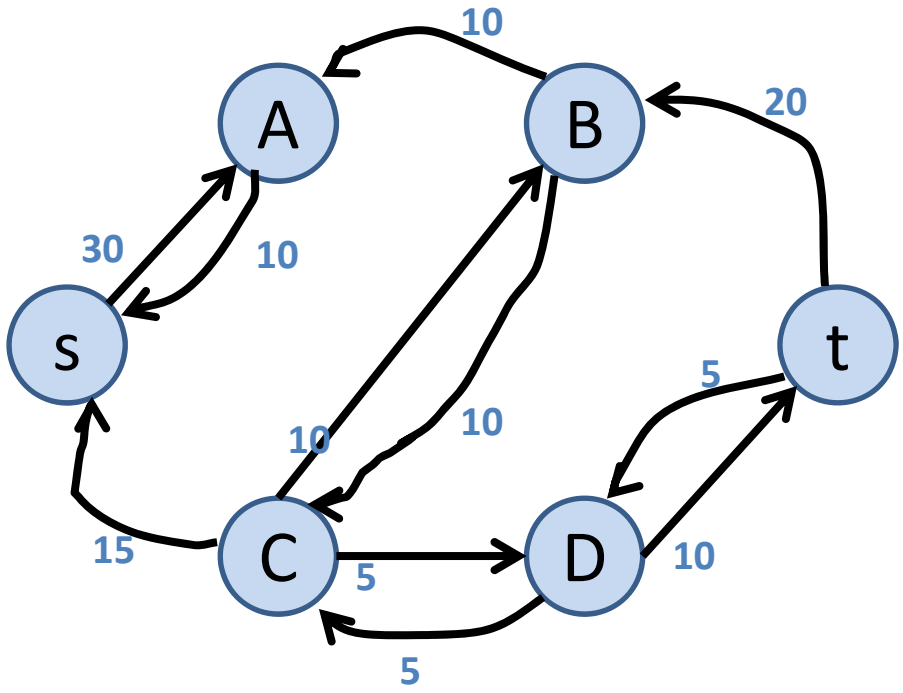
0. Find an initial feasible solution (flow)
1. Consider the current solution, x.
   Construct a residual graph, G_x.
2. Try to find an augmenting path in G_x.
   If there is an augmenting path,
      then use this path to improve the flow.
   If there is no augmenting path,
      then x is optimal. We're done!

No augmenting path = optimal?

Updated solution, $x^{(3)}$

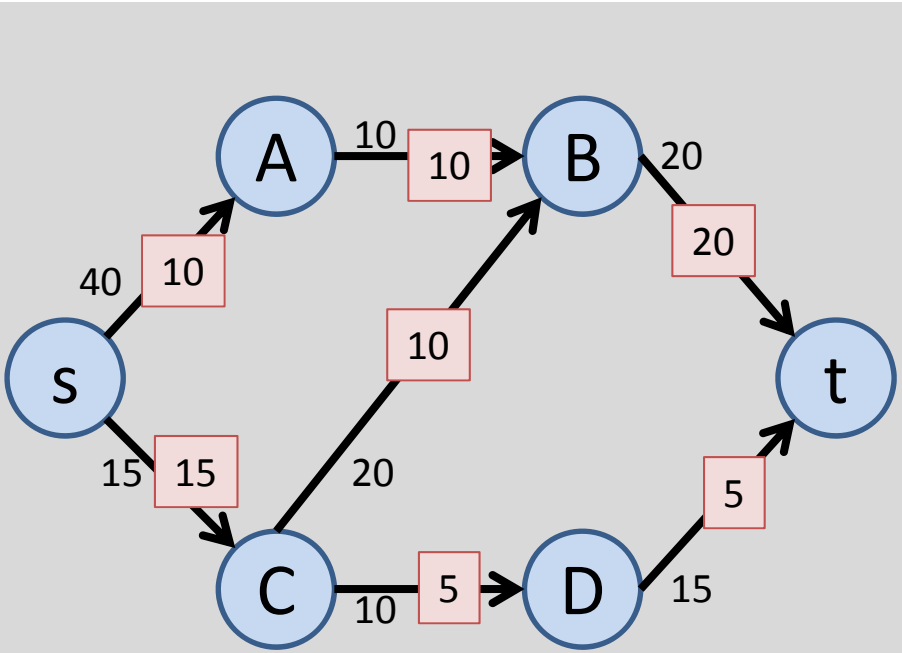


Residual graph for $x^{(3)}$, $G_{x(3)}$



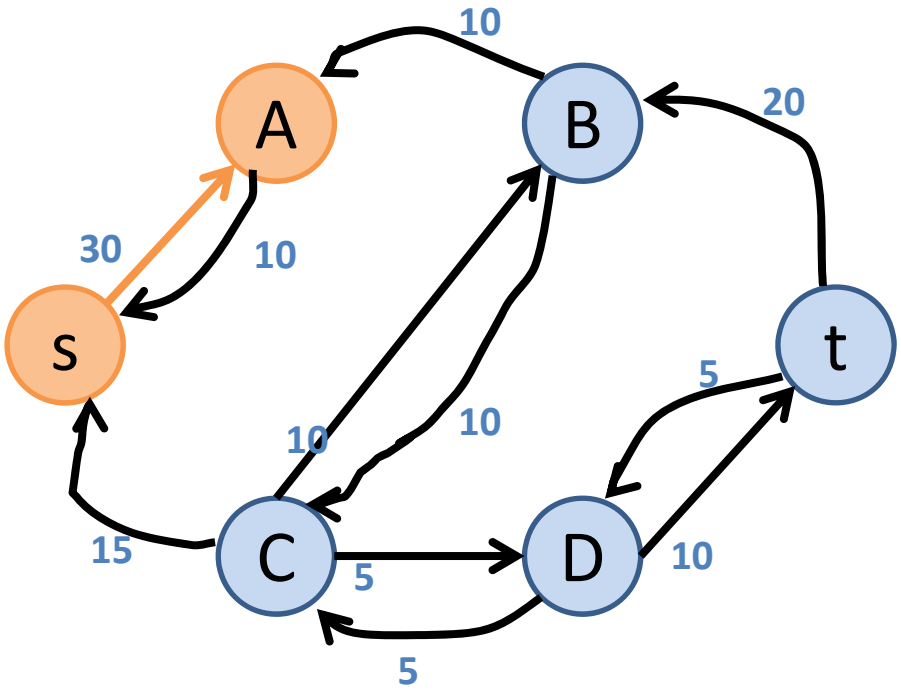
Nodes that are reachable from s in $G_{x(3)}$

Updated solution, $x^{(3)}$



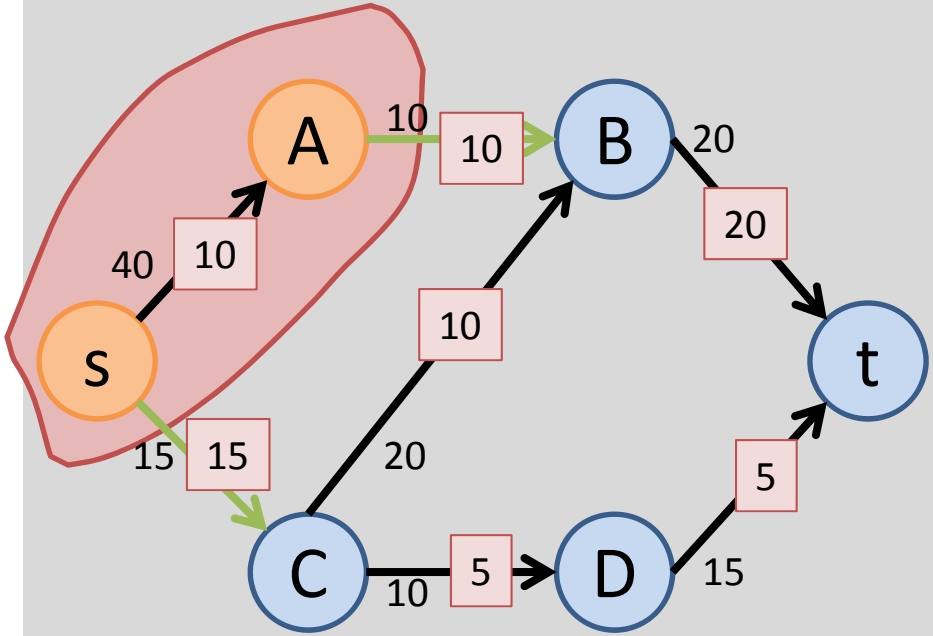
Flow value = 25

Residual graph for $x^{(3)}$, $G_{x(3)}$



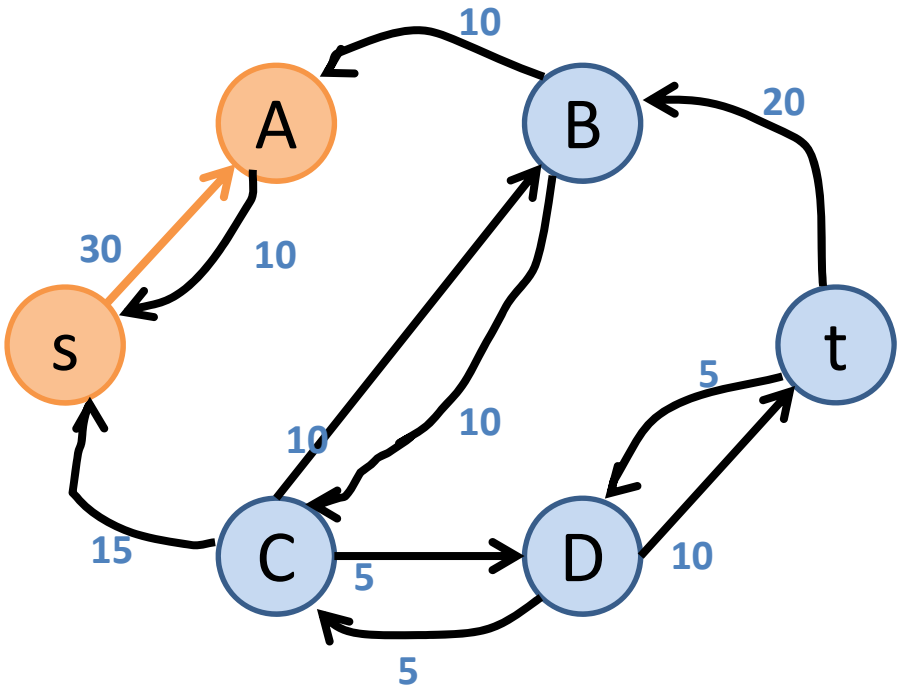
Nodes that are reachable from s in $G_{x(3)}$

Updated solution, $x^{(3)}$



Flow value = 25
Cut capacity = 25

Residual graph for $x^{(3)}$, $G_{x(3)}$



The maximum-flow and the minimum-cut problems

Theorem 1

Consider a maxflow problem with input
G = (N, E) and capacities u_{ij} for each (i, j) in E.

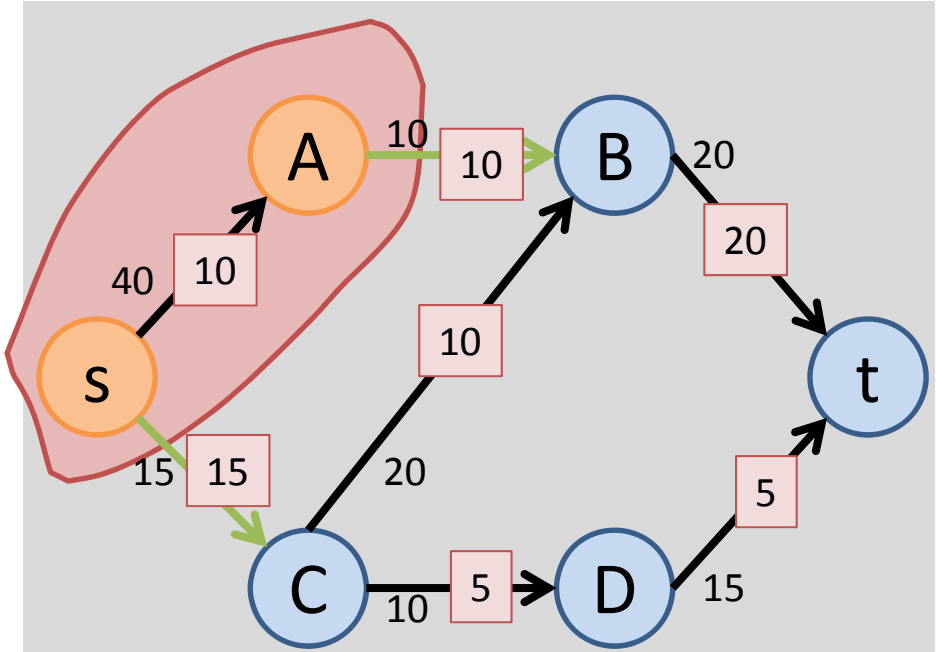
If x^* is a feasible flow and (S, T) is a valid cut with the property that

the value of flow of x^* = capacity of the cut (S, T),

then x^* is a maximum flow.

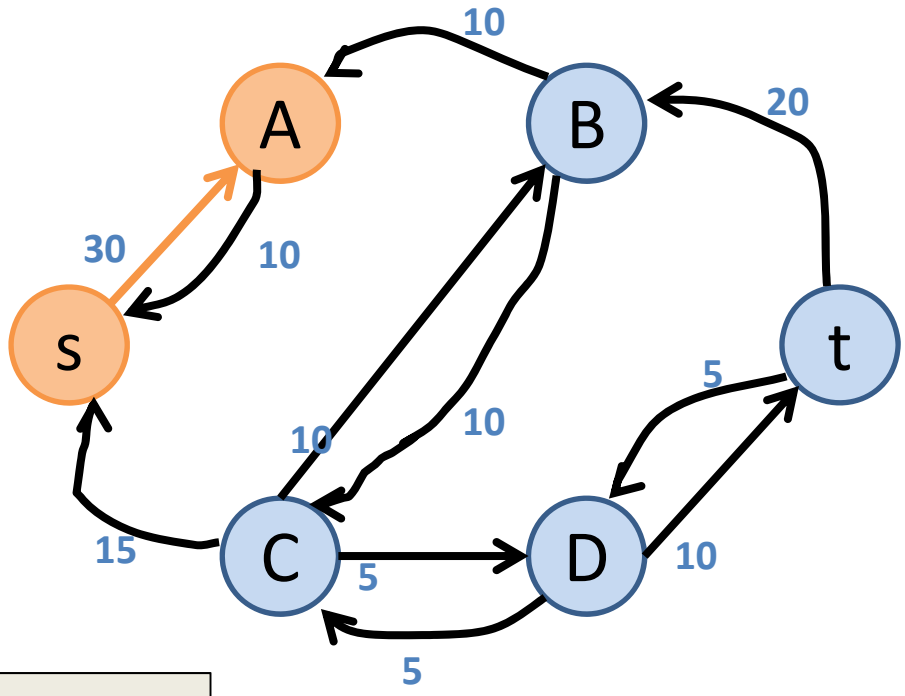
Nodes that are reachable from s in $G_{x(3)}$

Updated solution, $x^{(3)}$



Flow value = 25
Cut capacity = 25

Residual graph for $x^{(3)}$, $G_{x(3)}$



By Theorem 1,
$x^{(3)}$ is an optimal flow!

The Ford-Fulkerson method

0. Find an initial feasible solution (flow)
1. Consider the current solution, x.
   Construct a residual graph, G_x.
2. Try to find an augmenting path in G_x.
   If there is an augmenting path,
   then use this path to improve the flow.
   If there is no augmenting path,
   then x is optimal. We're done! (A cut whose capacity is equal to the flow value is found.)

Q2 (i>clicker)

(after the break)

Q2: Fun and Game!

You win if the number that you choose is closest to half the mean of the numbers chosen by the class.

A. 0

B. 1

C. 2

D. 3

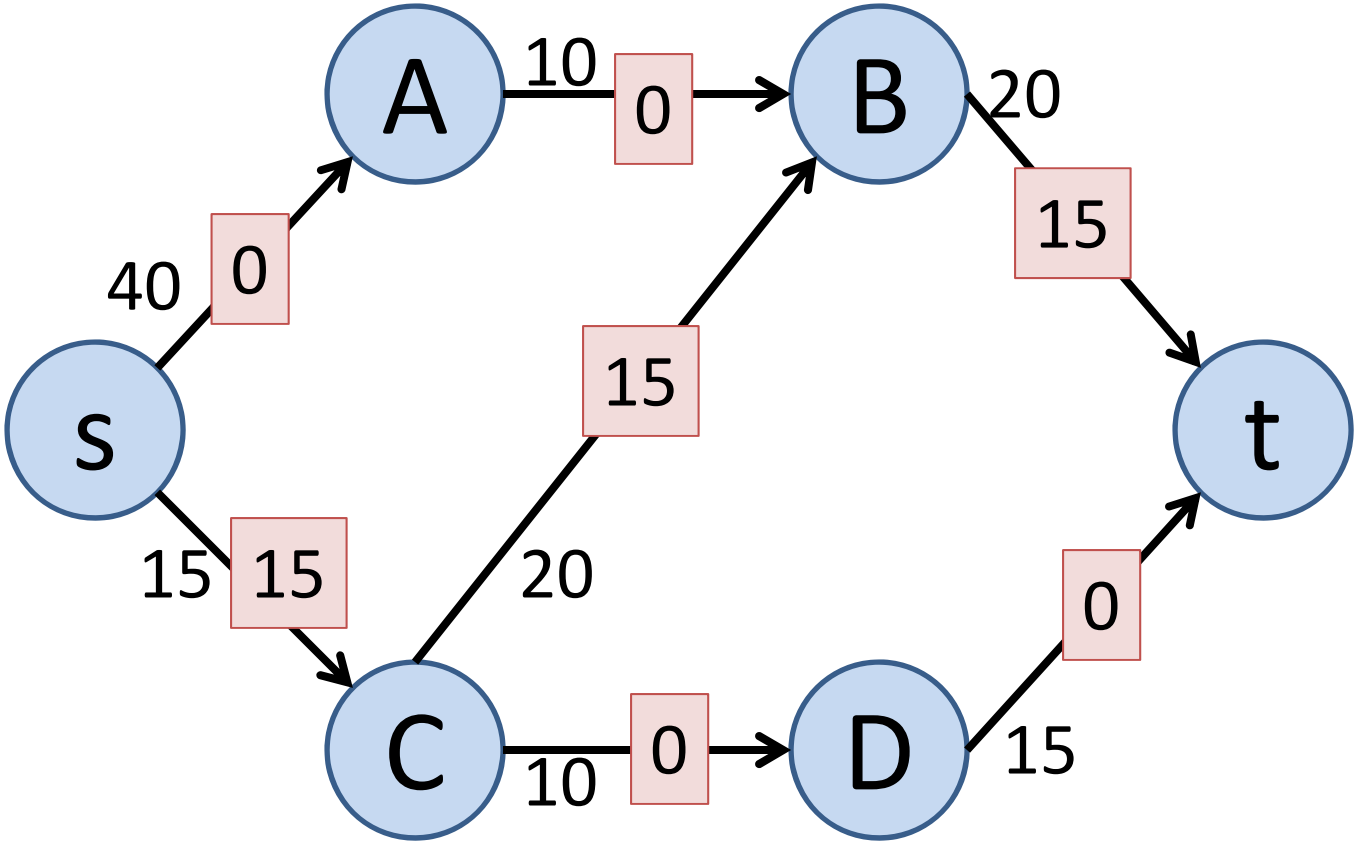
E. 4

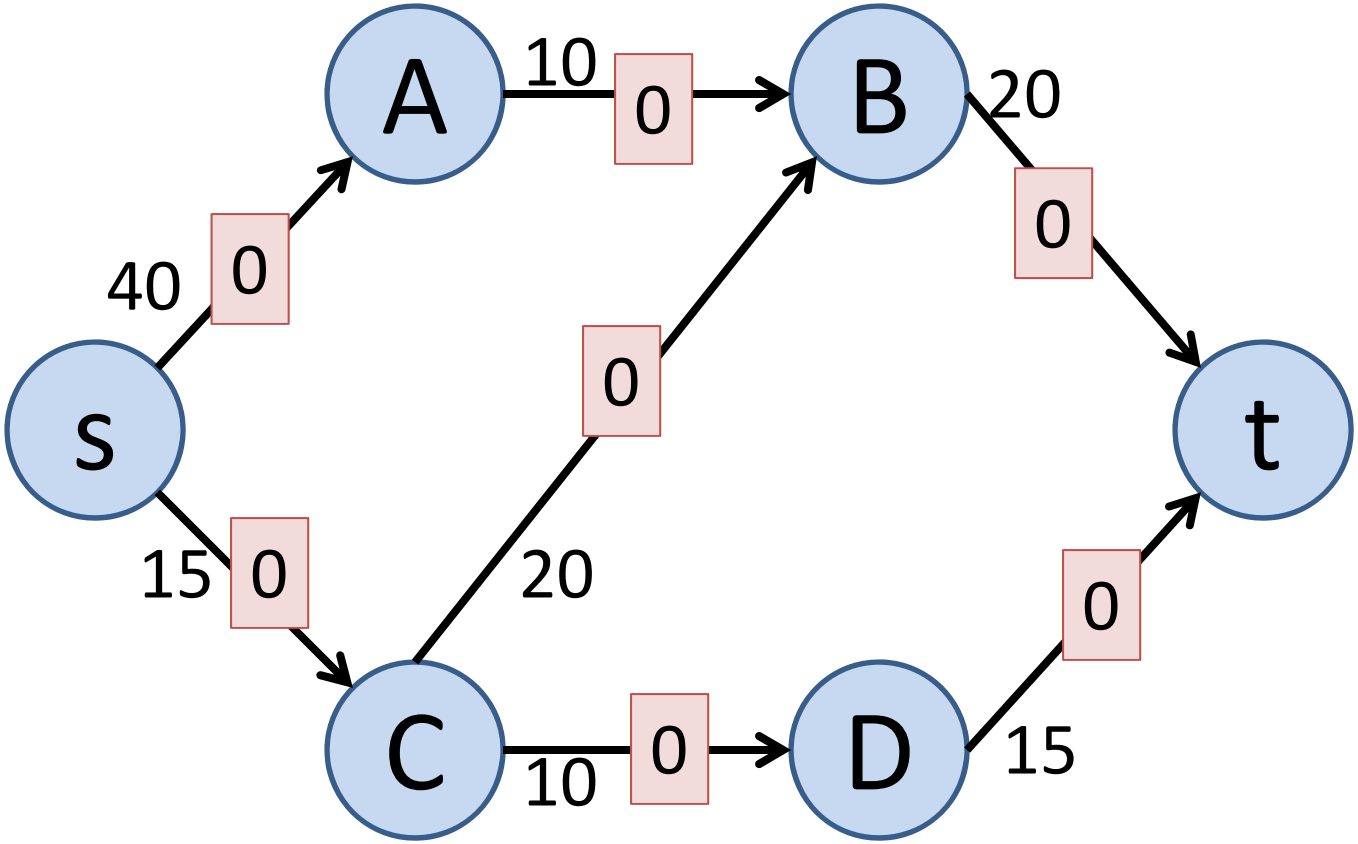
Back to Ford-Fulkerson

Tying loose ends

The Ford-Fulkerson method

0. Find an initial feasible solution (flow)
1. Consider the current solution, x.
   Construct a residual graph, G_x.
2. Try to find an augmenting path in G_x.
   If there is an augmenting path,
   then use this path to improve the flow.
   If there is no augmenting path,
   then x is optimal. We're done! (A cut whose capacity is equal to the flow value is found.)


A
B
C
D
s
t
10
0
20
15
40
0
15
15
20
15
0
10
0
15

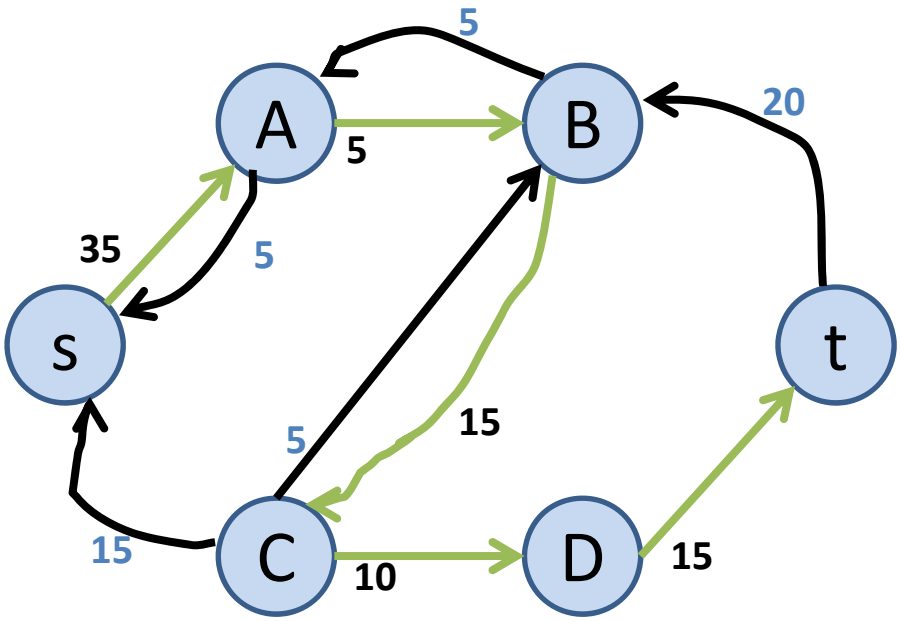

A
B
s
t
C
D
10
0
20
0
40
0
0
15
0
20
0
10
0
15

The Ford-Fulkerson method

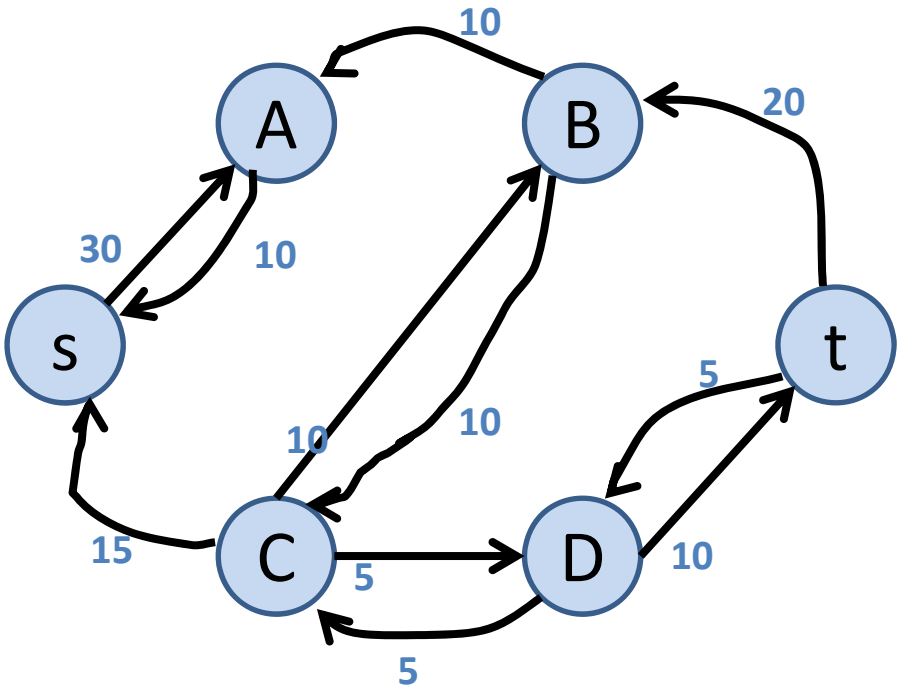
0. Find an initial feasible solution (flow)
1. Consider the current solution, x.
   Construct a residual graph, G_x.
2. Try to find an augmenting path in G_x.
   If there is an augmenting path,
   then use this path to improve the flow.
   If there is no augmenting path,
   then x is optimal. We're done! (A cut whose capacity is equal to the flow value is found.)

Lazy approach:
Obtaining $G_{x(3)}$ directly from $G_{x(2)}$

Residual graph for $x^{(2)}$, $G_{x(2)}$



Residual graph for $x^{(3)}$, $G_{x(3)}$



The Ford-Fulkerson method

0. Find an initial feasible solution (flow)
1. Consider the current solution, x.
   Construct a residual graph, G_x.
2. Try to find an augmenting path in G_x.
   If there is an augmenting path,
   then use this path to improve the flow.
   If there is no augmenting path,
   then x is optimal. We're done! (A cut whose capacity is equal to the flow value is found.)

Finding an augmenting path (Finding nodes reachable from s)

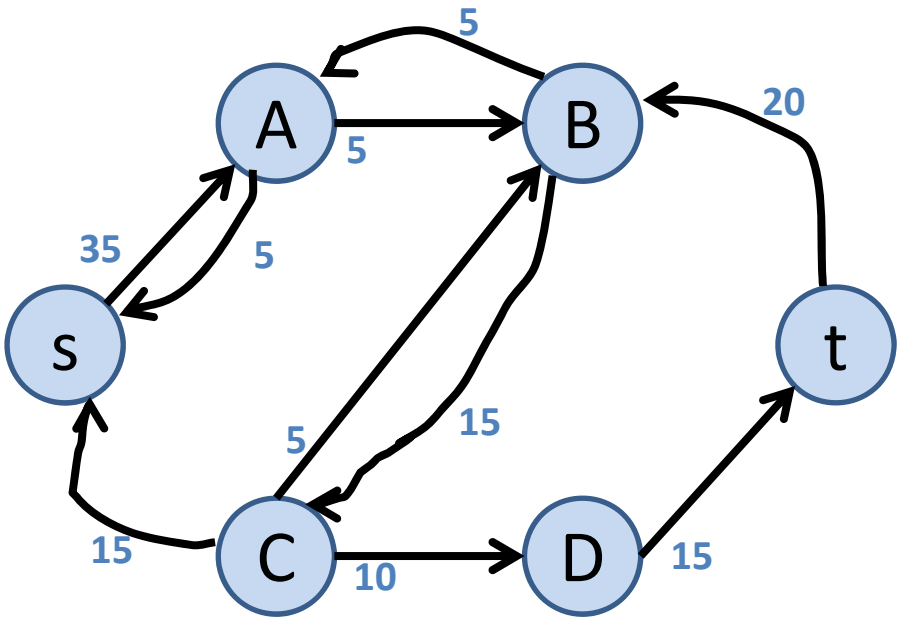
A labeling algorithm:

List of "checked" nodes: L = {s}

While L is not empty:

- Take the first node off the list L, suppose it's node i.
- For each edge (i, j) out of i, if j is unchecked, check it and add j to the list L. Highlight the edge.
- (repeat)

Residual graph for $x^{(2)}$, $G_{x(2)}$



Finding an augmenting path (Finding nodes reachable from s)

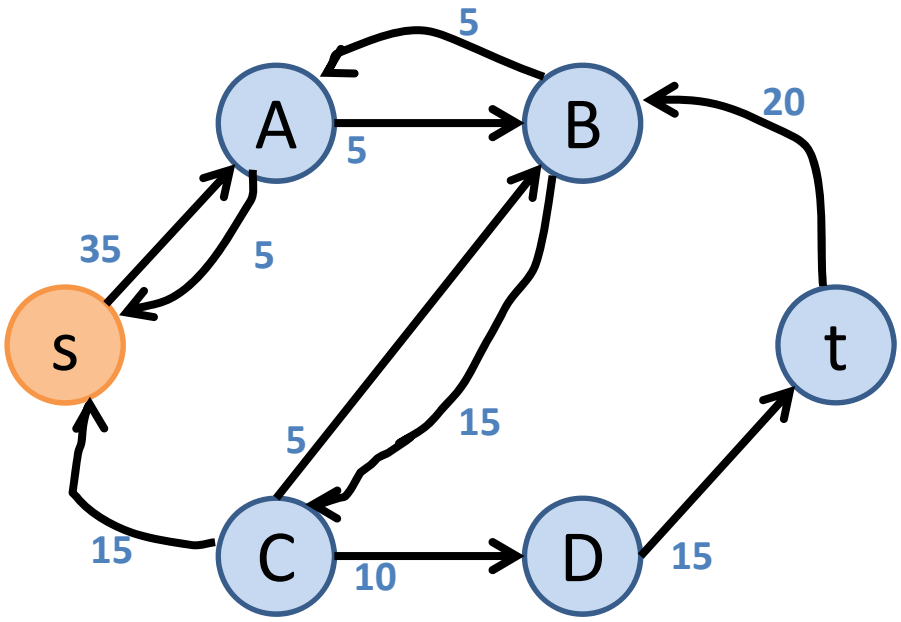
A labeling algorithm:

List of "checked" nodes: L = {s}

While L is not empty:

- Take the first node off the list L, suppose it's node i.
- For each edge (i, j) out of i, if j is unchecked, check it and add j to the list L. Highlight the edge
- (repeat)

Residual graph for $x^{(2)}$, $G_{x(2)}$



L = {s}

Finding an augmenting path (Finding nodes reachable from s)

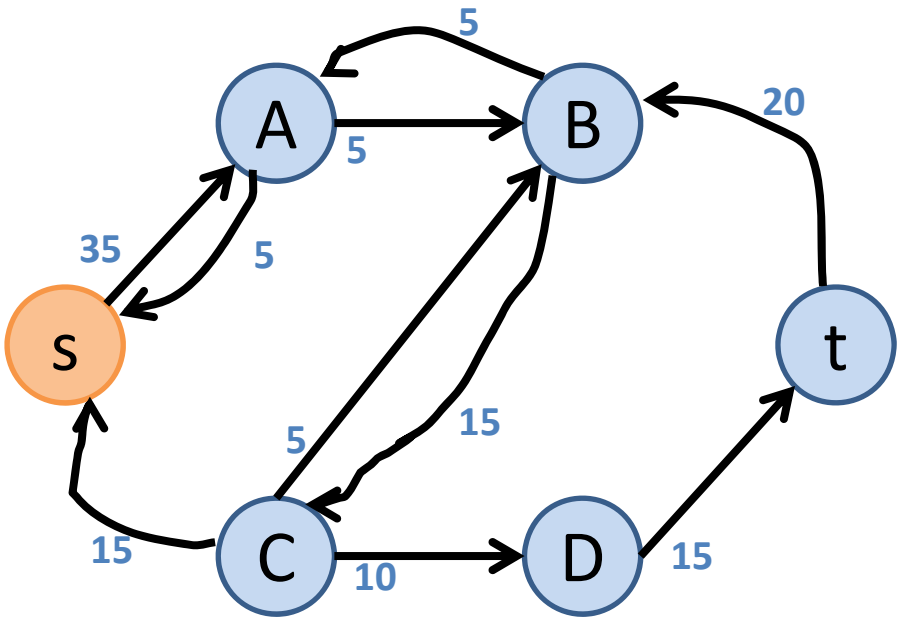
A labeling algorithm:

List of "checked" nodes: L = {s}

While L is not empty:

- Take the first node off the list L, suppose it's node i.
- For each edge (i, j) out of i, if j is unchecked, check it and add j to the list L. Highlight the edge
- (repeat)

Residual graph for $x^{(2)}$, $G_{x(2)}$



L = {}

Current node: s

Finding an augmenting path (Finding nodes reachable from s)

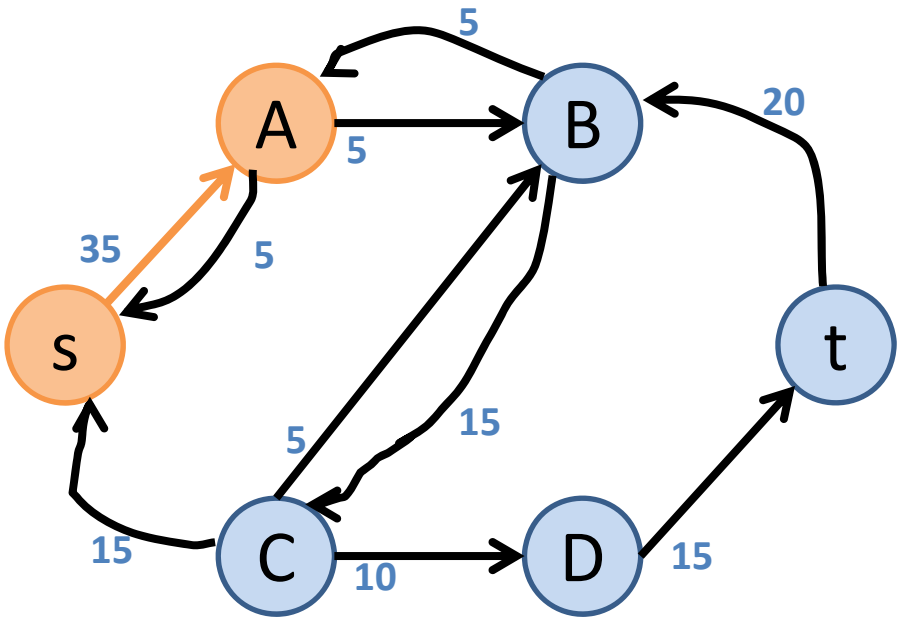
A labeling algorithm:

List of "checked" nodes: L = {s}

While L is not empty:

- Take the first node off the list L, suppose it's node i.
- For each edge (i, j) out of i, if j is unchecked, check it and add j to the list L. Highlight the edge
- (repeat)

Residual graph for $x^{(2)}$, $G_{x(2)}$



L = {A}. Edges: (s, A)

Current node: s

Finding an augmenting path (Finding nodes reachable from s)

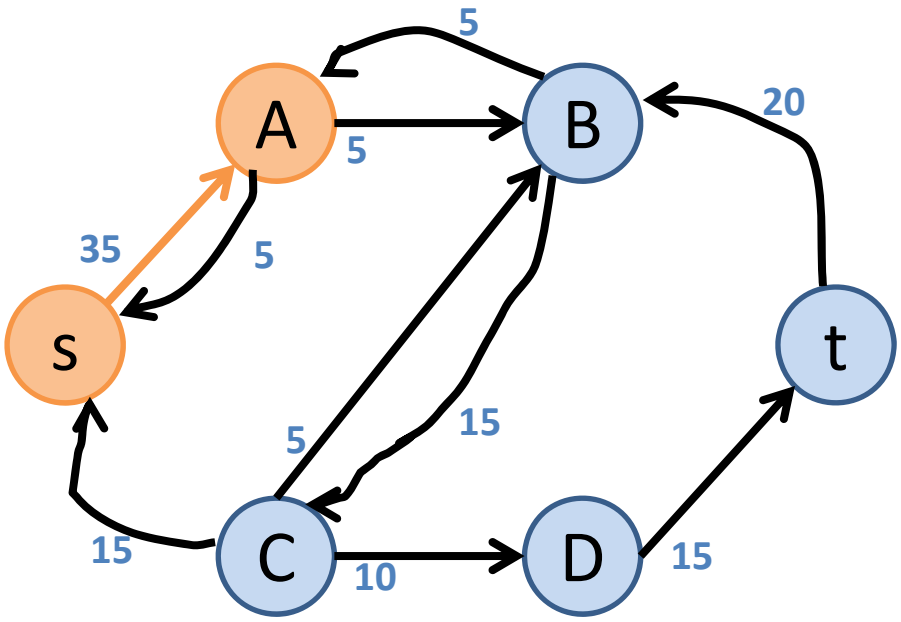
A labeling algorithm:

List of "checked" nodes: L = {s}

While L is not empty:

- Take the first node off the list L, suppose it's node i.
- For each edge (i, j) out of i, if j is unchecked, check it and add j to the list L. Highlight the edge
- (repeat)

Residual graph for $x^{(2)}$, $G_{x(2)}$



L = {}. Edges: (s, A)

Current node: A

Finding an augmenting path (Finding nodes reachable from s)

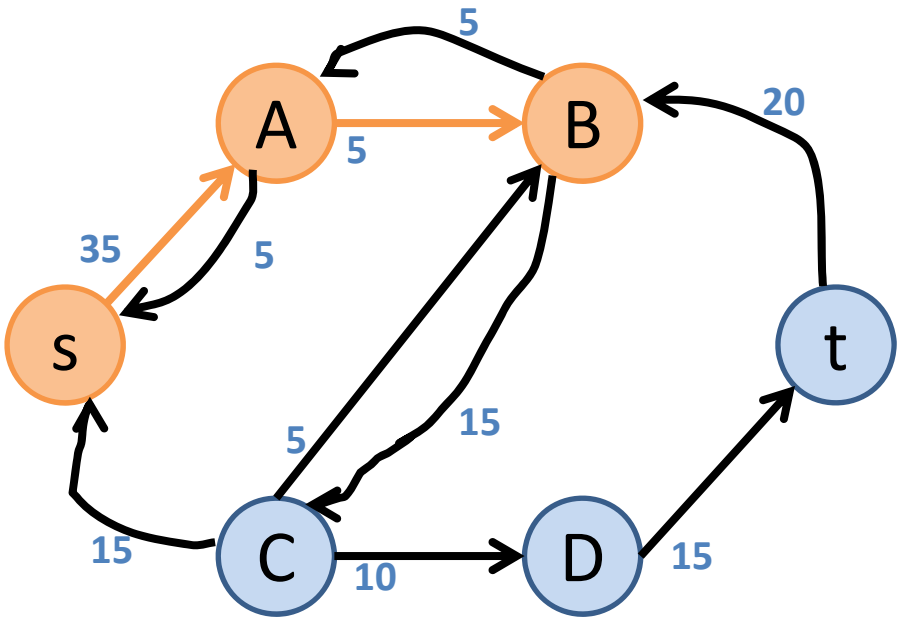
A labeling algorithm:

List of "checked" nodes: L = {s}

While L is not empty:

- Take the first node off the list L, suppose it's node i.
- For each edge (i, j) out of i, if j is unchecked, check it and add j to the list L. Highlight the edge
- (repeat)

Residual graph for $x^{(2)}$, $G_{x(2)}$



L = {B}. Edges: (s, A), (A, B)

Current node: A

Finding an augmenting path (Finding nodes reachable from s)

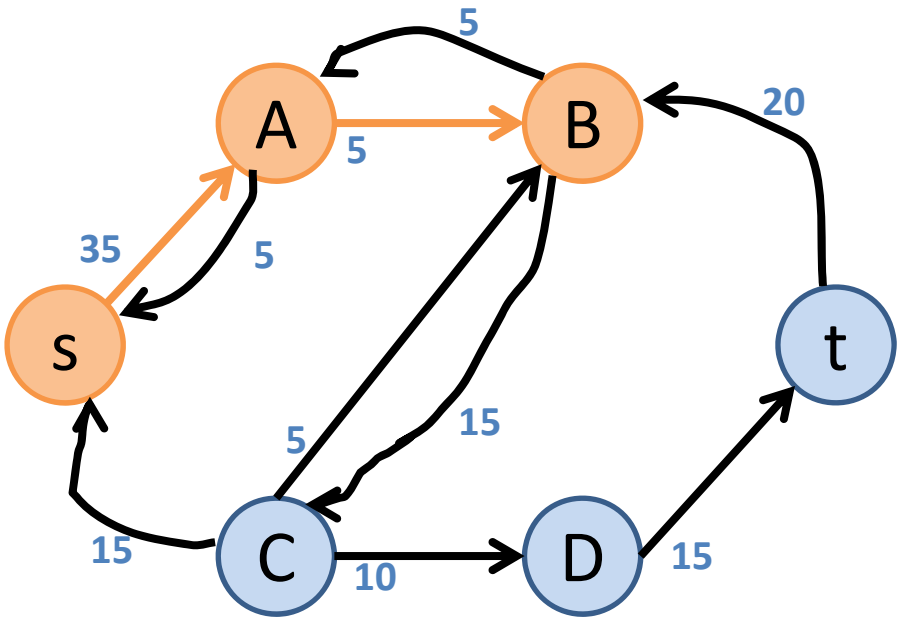
A labeling algorithm:

List of "checked" nodes: L = {s}

While L is not empty:

- Take the first node off the list L, suppose it's node i.
- For each edge (i, j) out of i, if j is unchecked, check it and add j to the list L. Highlight the edge
- (repeat)

Residual graph for $x^{(2)}$, $G_{x(2)}$



L = {}. Edges: (s, A), (A, B)

Current node: B

Finding an augmenting path (Finding nodes reachable from s)

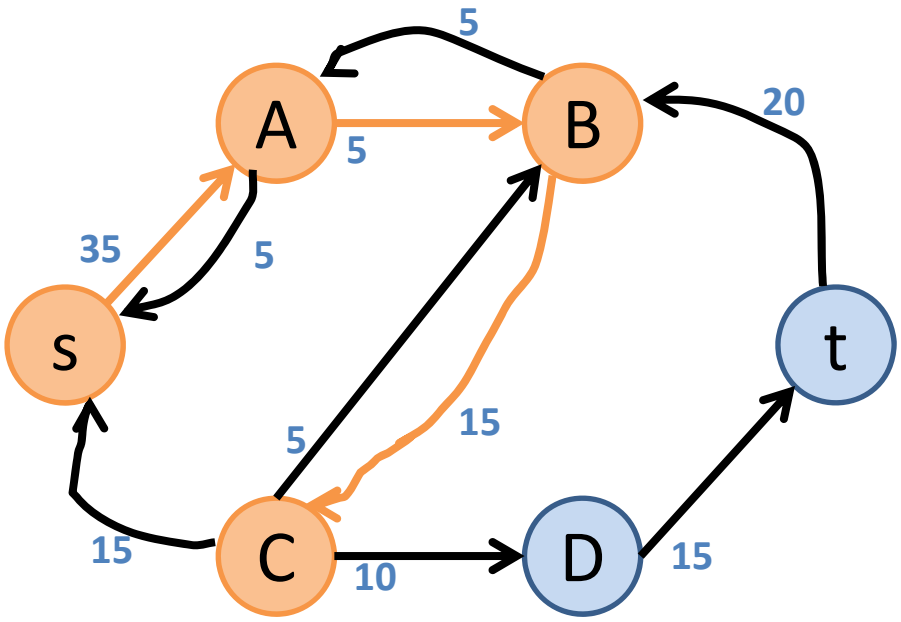
A labeling algorithm:

List of "checked" nodes: L = {s}

While L is not empty:

- Take the first node off the list L, suppose it's node i.
- For each edge (i, j) out of i, if j is unchecked, check it and add j to the list L. Highlight the edge.
- (repeat)

Residual graph for $x^{(2)}$, $G_{x(2)}$



L = {C}. Edges: (s, A), (A, B), (B, C)

Current node: B

Finding an augmenting path (Finding nodes reachable from s)

A labeling algorithm:

List of "checked" nodes: L = {s}

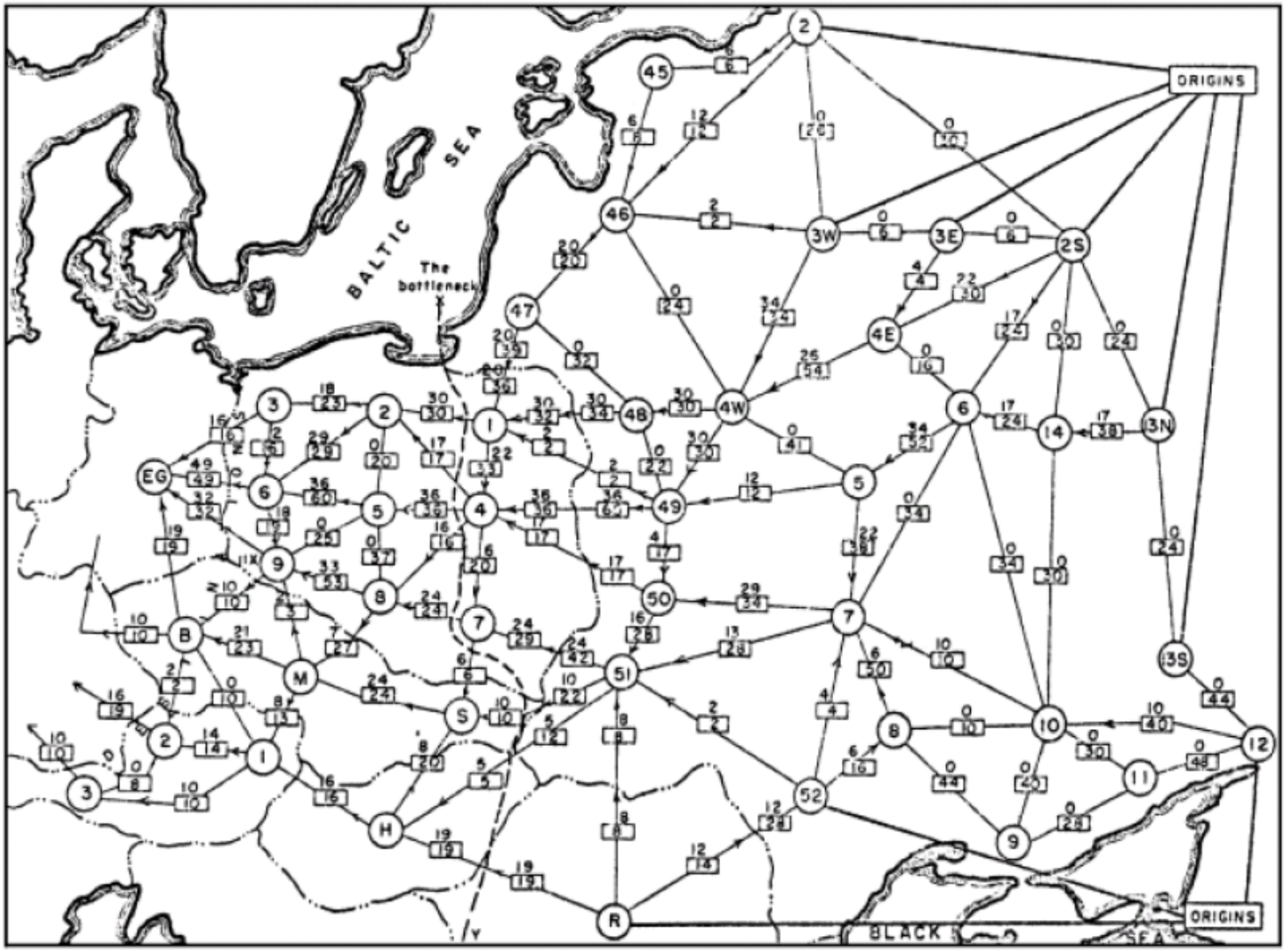
While L is not empty:

- Take the first node off the list L, suppose it's node i.
- For each edge (i, j) out of i, if j is unchecked, check it and add j to the list L. Highlight the edge.
- (repeat)

At the end of the algorithm one of the following happens:

1. The algorithm stops, and node t is not "checked".
   Then, …
2. The algorithm stops, and node t is "checked".
   Then, …

Some history



Harris-Ross, 1955

“

Air power is an effective means of interdicting an enemy's rail system, and such usage is a logical and important mission for this Arm.

As in many military operations, however, the success of interdiction depends largely on how complete, accurate, and timely is the commander's information, particularly concerning the effect of his interdiction-program efforts on the enemy's capability to move men and supplies. This information should be available at the time the results are being achieved.

The present paper describes the fundamentals of a method intended to help the specialist who is engaged in estimating railway capabilities, so that he might more readily accomplish this purpose and thus assist the commander and his staff with greater efficiency than is possible at present.

”