

Iterative Improvement Algorithms

10.1-10.4

Greedy Algorithm approach recap

The greedy strategy, considered in the preceding chapter, constructs a solution to an optimization problem piece by piece, always adding a locally optimal piece to a partially constructed solution.

Iterative improvement algorithm intro

In this chapter, we discuss a different approach to designing algorithms for optimization problems.

It starts with some feasible solution (a solution that satisfies all the constraints of the problem) and proceeds to improve it by repeated applications of some simple step.

Iterative improvement algorithm intro

In this chapter, we discuss a different approach to designing algorithms for optimization problems.

It starts with some feasible solution (a solution that satisfies all the constraints of the problem) and proceeds to improve it by repeated applications of some simple step.

This step typically involves a small, localized change yielding a feasible solution with an improved value of the objective function. When no such change improves the value of the objective function, the algorithm returns the last feasible solution as optimal and stops.

Obstacles for iterative improvement algorithms

First, we need an initial feasible solution.

For some problems, we can always start with a trivial solution or use an approximate solution obtained by some other (e.g., greedy) algorithm. But for others, finding an initial solution may require as much effort as solving the problem after a feasible solution has been identified.

Obstacles for iterative improvement algorithms

Second, it is not always clear what changes should be allowed in a feasible solution so that we can check efficiently whether the current solution is locally optimal and, if not, replace it with a better one.

Obstacles for iterative improvement algorithms

Third—and this is the most fundamental difficulty—is an issue of local versus global extremum (maximum or minimum). Think about the problem of finding the highest point in a hilly area with no map on a foggy day. A logical thing to do would be to start walking “up the hill” from the point you are at until it becomes impossible to do so because no direction would lead up. You will have reached a local highest point, but because of a limited feasibility, there will be no simple way to tell whether the point is the highest (global maximum you are after) in the entire area.

Problems that can be solved by iterative improvement

- Section 10.1 talks about the simplex method, the classic algorithm for linear programming.
- Section 10.2 talks about the important problem of maximizing the amount of flow that can be sent through a network with links of limited capacities.
(special case of linear programming)
- Section 10.3 deals with the problem of maximizing the number of matched pairs.
- Section 10.4 is concerned with the matching stability

Simplex Method

- We already talked about linear programming in section 6.6
- I gave an example of manufacturing different amounts of each product in order to maximize profit
- the general problem of optimizing a linear function of several variables subject to a set of linear constraints:

Simplex Method

Can someone give a review of simplex method?

Simplex Method recap

This example is in your textbook:

The set of points defined by inequality
 $x + y \leq 4$ comprises the points **on**

and below the line $x + y = 4$,

and the set of points defined by

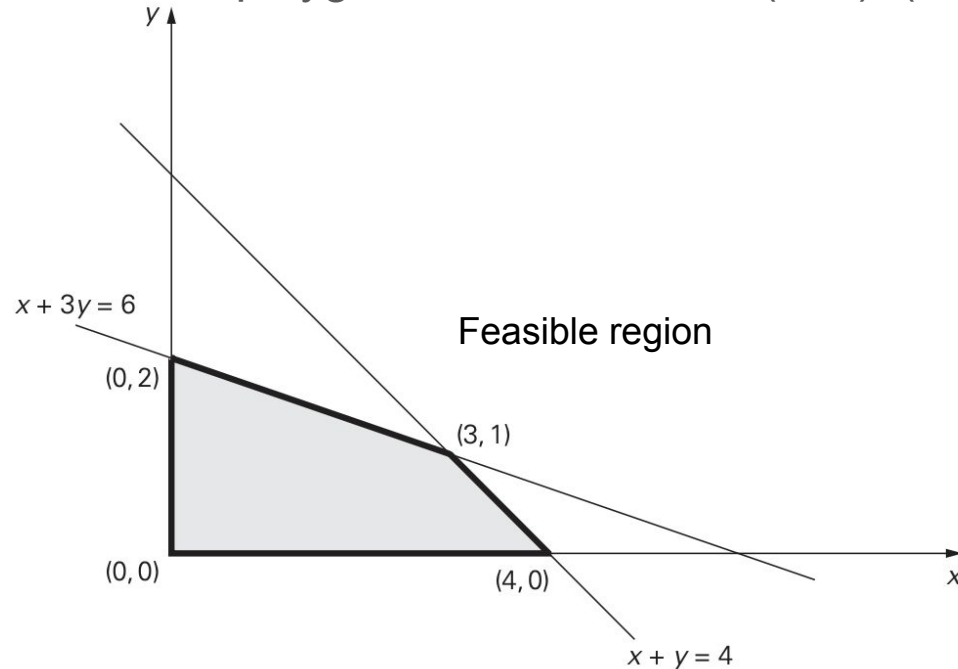
inequality $x + 3y \leq 6$ comprises

the points **on and below the line $x + 3y = 6$.**

$$\begin{array}{ll}\text{maximize} & 3x + 5y \\ \text{subject to} & x + y \leq 4 \\ & x + 3y \leq 6 \\ & x \geq 0, \quad y \geq 0.\end{array}$$

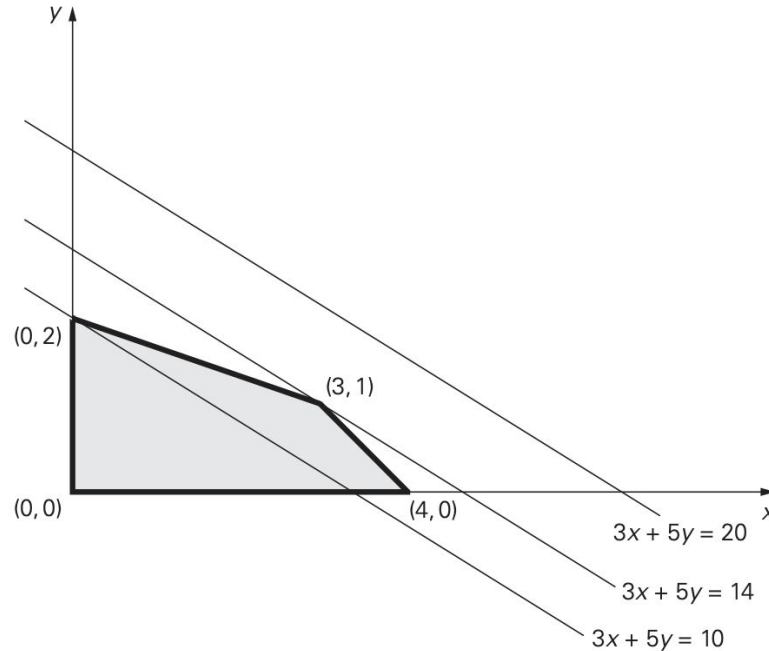
Simplex Method

When we draw the plane of feasible solutions and include the non-negativity constraint, we get the convex polygon with the vertices $(0, 0)$, $(4, 0)$, $(0, 2)$, and $(3, 1)$.



Simplex Method

Our task is to find an optimal solution, a point in the feasible region with the largest value of the objective function $z = 3x + 5y$.



Simplex Method

When I explained my personal story with linear programming, I told you all that it was no coincidence that the optimal solution was found at a vertex of my feasible region.

Simplex Method

When I explained my personal story with linear programming, I told you all that it was no coincidence that the optimal solution was found at a vertex of my feasible region.

Let's add to our elementary knowledge of simplex method and look at a couple exceptions to this:

Simplex Method

When I explained my personal story with linear programming, I told you all that it was no coincidence that the optimal solution was found at a vertex of my feasible region.

Let's add to our elementary knowledge of simplex method and look at a couple exceptions to this:

1- if the constraints include two contradictory requirements, such as $x + y \leq 1$ and $x + y \geq 2$

2- if the problem's feasible region is unbounded

Simplex Method: unbounded

Let's reverse our inequalities from the previous example

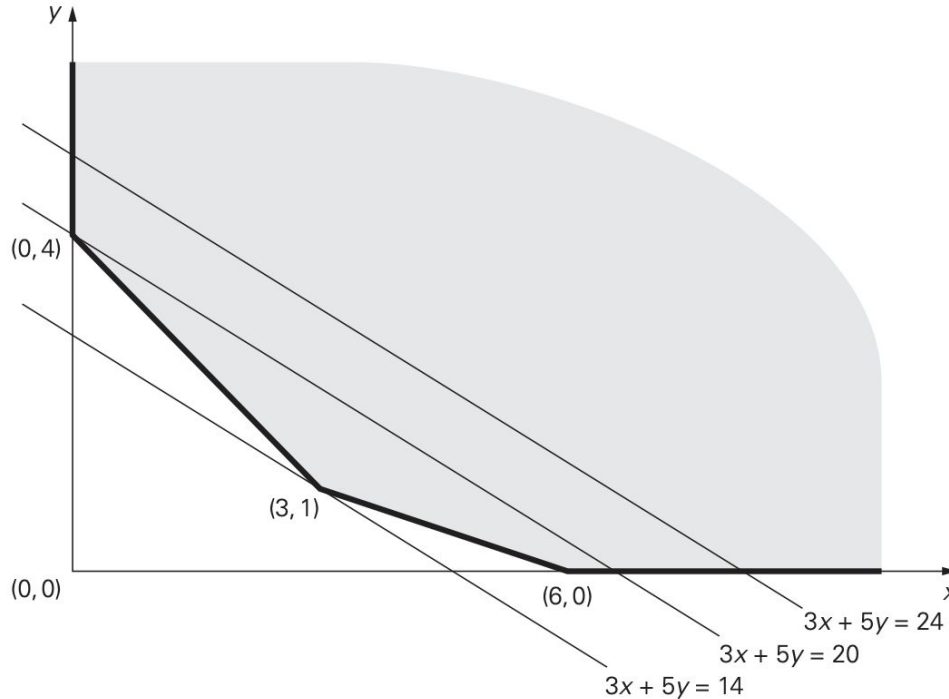
Simplex Method: unbounded

Let's reverse our inequalities from the previous example

So now we have constraints of $x + y \geq 4$ and $x + 3y \geq 6$

Simplex Method: unbounded

When we look at the feasible region, it is now unbounded



Note on simplex method

We can minimize an unbounded problem just as we can maximize an objective function

Extreme Point Theorem

Any linear programming problem with a nonempty bounded feasible region has an optimal solution; moreover, an optimal solution can always be found at an extreme point of the problem's feasible region.

Extreme Point Theorem

Any linear programming problem with a nonempty bounded feasible region has an optimal solution; moreover, an optimal solution can always be found at an extreme point of the problem's feasible region.

In principle, we can solve such a problem by computing the value of the objective function at each extreme point and selecting the one with the best value.

Extreme Point Theorem obstacles

1- The first lies in the need for a mechanism for generating the extreme points of the feasible region.

2- The second obstacle lies in the number of extreme points a typical feasible region has. This can grow exponentially as the size of the problem increases

Testing extreme points in feasible region

The simplex method is the famous algorithm used for testing a number of extreme points before finding an optimal solution

Testing extreme points in feasible region

The simplex method is the famous algorithm used for testing a number of extreme points before finding an optimal solution

- 1- Start by identifying an extreme point of the feasible region.

Testing extreme points in feasible region

The simplex method is the famous algorithm used for testing a number of extreme points before finding an optimal solution

- 1- Start by identifying an extreme point of the feasible region.
- 2- Then check whether one can get an improved value of the objective function by going to an adjacent extreme point.

Testing extreme points in feasible region

The simplex method is the famous algorithm used for testing a number of extreme points before finding an optimal solution

- 1- Start by identifying an extreme point of the feasible region.
- 2- Then check whether one can get an improved value of the objective function by going to an adjacent extreme point. If it is not the case, the current point is optimal—stop; if it is the case, proceed to an adjacent extreme point with an improved value of the objective function.

Testing extreme points in feasible region

The simplex method is the famous algorithm used for testing a number of extreme points before finding an optimal solution

- 1- Start by identifying an extreme point of the feasible region.
- 2- Then check whether one can get an improved value of the objective function by going to an adjacent extreme point. If it is not the case, the current point is optimal—stop; if it is the case, proceed to an adjacent extreme point with an improved value of the objective function.
- 3- After a finite number of steps, the algorithm will either reach an extreme point where an optimal solution occurs or determine that no optimal solution exists.

Standard form

To implement the simplex method, the problem needs to be in standard form

- It must be a maximization problem.
- All the constraints (except the nonnegativity constraints) must be in the form of linear equations with nonnegative right-hand sides.
- All the variables must be required to be nonnegative.

Slack variables

We talked about slack variables when we discussed linear programming in chapter 6

Please read the part of 10.1 that goes through the linear algebra of slack variables- super fascinating!

If a constraint is given as an inequality, it can be replaced by an equivalent equation by adding a slack variable

Let's do an example together

$$\textit{Minimize} : -z = -8x_1 - 10x_2 - 7x_3$$

$$\textit{s.t.} : x_1 + 3x_2 + 2x_3 \leq 10$$

$$-x_1 - 5x_2 - x_3 \geq -8$$

$$x_1, x_2, x_3 \geq 0$$

Simplex method- example

What do we do first?

Simplex method- example

What do we do first?

Standard form!

The 3 requirements for standard form can always be satisfied by transforming any given linear program using basic algebra and substitution.

Transform a minimization model into a maximization model

Multiply both the left and the right sides of the objective function by -1.

$$-1 \times (-z = -8x_1 - 10x_2 - 7x_3)$$

$$z = 8x_1 + 10x_2 + 7x_3$$

$$\textit{Maximize} : z = 8x_1 + 10x_2 + 7x_3$$

Transform constraints from \geq inequality to \leq inequality

By multiplying by -1 on both sides, the inequality can be changed to less-than-or-equal-to

$$-1 \times (-x_1 - 5x_2 - x_3 \geq -8)$$

$$x_1 + 5x_2 + x_3 \leq 8$$

Determine slack variables

Slack variables are additional variables that are introduced into the linear constraints of a linear program to transform them from inequality constraints to equality constraints.

If the model is in standard form, the slack variables will always have a +1 coefficient.

Slack variables are needed in the constraints to transform them into solvable equalities with one definite answer.

Determine slack variables

$$x_1 + 3x_2 + 2x_3 + \mathbf{s_1} = 10$$

$$x_1 + 5x_2 + x_3 + \mathbf{s_2} = 8$$

$$x_1, x_2, x_3, \mathbf{s_1}, \mathbf{s_2} \geq 0$$

Now we have:

$$\textit{Maximize} : z = 8x_1 + 10x_2 + 7x_3$$

$$\textit{s.t.} : x_1 + 3x_2 + 2x_3 + s_1 = 10$$

$$x_1 + 5x_2 + x_3 + s_2 = 8$$

Setting up the Tableau

A Simplex tableau is used to perform row operations on the linear programming model as well as to check a solution for optimality. The tableau consists of the coefficient corresponding to the linear constraint variables and the coefficients of the objective function.

Setting up the Tableau

In the tableau below, the bolded top row of the tableau states what each column represents. The following two rows represent the linear constraint variable coefficients from the linear programming model, and the last row represents the objective function variable coefficients.

x1	x2	x3	s1	s2	z	b
1	3	2	1	0	0	10
1	5	1	0	1	0	8
-8	-10	-7	0	0	1	0

Check optimality

The optimal solution of a maximization linear programming model are the values assigned to the variables in the objective function to give the largest zeta value.

The optimal solution would exist on the corner points of the graph of the entire model.

To check optimality using the tableau, all values in the last row must contain values greater than or equal to zero. If a value is less than zero, it means that variable has not reached its optimal value.

As seen in the previous tableau, three negative values exists in the bottom row indicating that this solution is not optimal.

Identify pivot table

If a tableau is not optimal, the next step is to identify the pivot variable to base a new tableau on.

The pivot variable can be identified by looking at the bottom row of the tableau and the indicator.

Assuming that the solution is not optimal, pick the smallest negative value in the bottom row. One of the values lying in the column of this value will be the pivot variable.

To find the indicator, divide the beta values of the linear constraints by their corresponding values from the column containing the possible pivot variable. The intersection of the row with the smallest non-negative indicator and the smallest negative value in the bottom row will become the pivot variable

Identify pivot table

In the example shown below, -10 is the smallest negative in the last row. This will designate the x_2 column to contain the pivot variable. Solving for the indicator gives us a value of $10/3$ for the first constraint, and a value of $8/5$ for the second constraint. Due to $8/5$ being the smallest non-negative indicator, the pivot value will be in the second row and have a value of 5

x_1	x_2	x_3	s_1	s_2	z	b	Indicator
1	3	2	1	0	0	10	$10/3$
1	5	1	0	1	0	8	$8/5$
-8	-10	-7	0	0	1	0	

↑
Smallest Value

Create the new Tableau

Now that the new pivot variable has been identified, the new tableau can be created to optimize the variable and find the new possible optimal solution.

To optimize the pivot variable, it will need to be transformed into a unit value (value of 1). To transform the value, multiply the row containing the pivot variable by the reciprocal of the pivot value. In the example below, the pivot variable is originally 5, so multiply the entire row by $1/5$

x1	x2	x3	s1	s2	z	b
1/5	①	1/5	0	1/5	0	8/5

← Pivot row

Create the new Tableau

After the unit value has been determined, the other values in the column containing the unit value will become zero. This is because the x_2 in the second constraint is being optimized, which requires x_2 in the other equations to be zero.

x_1	x_2	x_3	s_1	s_2	z	b	
	0						
$1/5$	①	$1/5$	0	$1/5$	0	$8/5$	← Pivot row
	0						

↑
Pivot Column

Create the new Tableau

In order to keep the tableau equivalent, the other variables not contained in the pivot column or pivot row must be calculated by using the new pivot values. For each new value, multiply the negative of the value in the old pivot column by the value in the new pivot row that corresponds to the value being calculated. Then add this to the old value from the old tableau to produce the new value for the new tableau.

New tableau value = (Negative value in old tableau pivot column) x (value in new tableau pivot row) + (Old tableau value)

Create the new Tableau

Old Tableau:

x1	x2	x3	s1	s2	z	b
1	3	2	1	0	0	10
1	5	1	0	1	0	8
-8	-10	-7	0	0	1	0

Old pivot column

New Tableau:

x1	x2	x3	s1	s2	z	b
2/5	0	7/5	1	-3/5	0	26/5
1/5	1	1/5	0	1/5	0	8/5
-6	0	-5	0	2	1	16

New pivot row

Example to clarify

To find the s2 value in row 1:

New tableau value = (Negative value in old tableau pivot column) * (value in new tableau pivot row) + (Old tableau value)

$$\text{New tableau value} = (-3) * \left(\frac{1}{5}\right) + 0 = -\frac{3}{5}$$


Check optimality

The optimal solution of a maximization linear programming model are the values assigned to the variables in the objective function to give the largest zeta value. Optimality will need to be checked after each new tableau to see if a new pivot variable needs to be identified. A solution is considered optimal if all values in the bottom row are greater than or equal to zero. If all values are greater than or equal to zero, the solution is considered optimal and you are done. If negative values exist, the solution is still not optimal and a new pivot point will need to be determined.

Identify new pivot variable

If the solution has been identified as not optimal, a new pivot variable will need to be determined. The pivot variable can be identified by the intersection of the row with the smallest non-negative indicator and the smallest negative value in the bottom row.

x1	x2	x3	s1	s2	z	b	Indicator
2/5	0	7/5	1	-3/5	0	26/5	$(26/5) / (2/5) = 13$
1/5	1	1	0	1/5	0	8/5	$(8/5) / (1/5) = 8$
-6	0	-5	0	2	1	0	



Smallest Value

Create new tableau

After the new pivot variable has been identified, a new tableau will need to be created. Make the pivot variable 1 by multiplying the row containing the pivot variable by the reciprocal of the pivot value. In the tableau below, the pivot value was $1/5$, so everything is multiplied by 5.

x1	x2	x3	s1	s2	z	b
①	5	1	0	1	0	8

Create new tableau

Next, make the other values in the column of the pivot variable zero. This is done by taking the negative of the old value in the pivot column and multiplying it by the new value in the pivot row. That value is then added to the old value that is being replaced.

x1	x2	x3	s1	s2	z	b
0	-2	1	1	-1	0	2
①	5	1	0	1	0	8
0	30	1	0	8	1	64

Check optimality

Using the new tableau, check for optimality. As mentioned, an optimal solution appears when all values in the bottom row are greater than or equal to zero. If all values are greater than or equal to zero, optimality has been reached. If negative values still exist, repeat identifying a pivot and creating a tableau until an optimal solution is obtained.

Identify optimal values

Once the tableau is proven optimal the optimal values can be identified. These can be found by distinguishing the basic and non-basic variables. A basic variable can be classified to have a single 1 value in its column and the rest be all zeros. If a variable does not meet this criteria, it is considered non-basic. If a variable is non-basic it means the optimal solution of that variable is zero. If a variable is basic, the row that contains the 1 value will correspond to the beta value. The beta value will represent the optimal solution for the given variable.

Identify optimal values

x1	x2	x3	s1	s2	z	b
0	-2	1	1	-1	0	2
1	5	1	0	1	0	8
0	30	1	0	8	1	64

Basic variables: x_1 , s_1 , z

Non-basic variables: x_2 , x_3 , s_2

Identify optimal values - explained

For the variable x_1 , the 1 is found in the second row. This shows that the optimal x_1 value is found in the second row of the beta values, which is 8.

Variable s_1 has a 1 value in the first row, showing the optimal value to be 2 from the beta column. Due to s_1 being a slack variable, it is not actually included in the optimal solution since the variable is not contained in the objective function.

The zeta variable has a 1 in the last row. This shows that the maximum objective value will be 64 from the beta column.

Identify optimal values

The final solution shows each of the variables having values of:

$$x_1 = 8$$

$$s_1 = 2$$

$$x_2 = 0$$

$$s_2 = 0$$

$$x_3 = 0$$

$$z = 64$$

The maximum optimal value is 64 and found at (8, 0, 0) of the objective function.

Conclusion to simplex method

The Simplex method is an approach for determining the optimal value of a linear program by hand. The method produces an optimal solution to satisfy the given constraints and produce a maximum zeta value. To use the Simplex method, a given linear programming model needs to be in standard form, where slack variables can then be introduced. Using the tableau and pivot variables, an optimal solution can be reached. From the example worked throughout this document, it can be determined that the optimal objective value is 64 and can be found when $x_1=8$, $x_2=0$, and $x_3=0$.

Homework problem!

Please complete exercise 10.1 #2

The maximum flow problem

We will now look at the problem of maximizing the flow of a material through a transportation network (pipeline system, communication system, electrical distribution system, and so on). We will assume that the transportation network in question can be represented by a connected weighted digraph with n vertices numbered from 1 to n and a set of edges E , with the following properties:

The maximum flow problem

We will now look at the problem of maximizing the flow of a material through a transportation network (pipeline system, communication system, electrical distribution system, and so on). We will assume that the transportation network in question can be represented by a connected weighted digraph with n vertices numbered from 1 to n and a set of edges E , with the following properties:

1. It contains exactly one vertex with no entering edges; this vertex is called the source and assumed to be numbered 1.
2. It contains exactly one vertex with no leaving edges; this vertex is called the sink and assumed to be numbered n .
3. The weight u_{ij} of each directed edge (i, j) is a positive integer, called the edge capacity. (This number represents the upper bound on the amount of the material that can be sent from i to j through a link represented by this edge.)

The maximum flow problem

Maximum flow problems involve finding a feasible flow through a single-source, single-sink flow network that is maximum.

Each edge is labeled with capacity, the maximum amount of stuff that it can carry. The goal is to figure out how much stuff can be pushed from the vertex s (source) to the vertex t (sink).

The maximum flow problem

<https://www.youtube.com/watch?v=MD4Cw8nUSis>

Ford-Fulkerson algorithm

The following is simple idea of Ford-Fulkerson algorithm:

- 1) Start with initial flow as 0.
- 2) While there is a augmenting path from source to sink.
 Add this path-flow to flow.
- 3) Return flow.

Time complexity of Ford-Fulkerson

Time complexity of the above algorithm is $O(\text{max_flow} * E)$.

We run a loop while there is an augmenting path. In worst case, we may add 1 unit flow in every iteration. Therefore the time complexity becomes $O(\text{max_flow} * E)$.

Ford-Fulkerson algorithm

<https://www.youtube.com/watch?v=TI90tNtKvxs>

Ford-Fulkerson algorithm- example

Given a flow network G with source s and sink t , the maximum flow problem is an optimization problem to find a flow of maximum value from s to t .

Flow network $G=(V, E)$, is essentially just a directed graph where each edge has a nonnegative flow capacity.

Ford-Fulkerson algorithm- example

The Ford Fulkerson method, also known as 'augmenting path algorithm' is an effective approach to solve the maximum flow problem. The Ford Fulkerson method depends on two main concepts:

1. Residual Network
2. Augmenting paths

Residual Network

It is a graph, that has the same vertices as the original network with one or two edges for each edge in the original network.

And it indicates the additional possible flow through the network.

For each edge, we'll calculate the additional flow as follows.

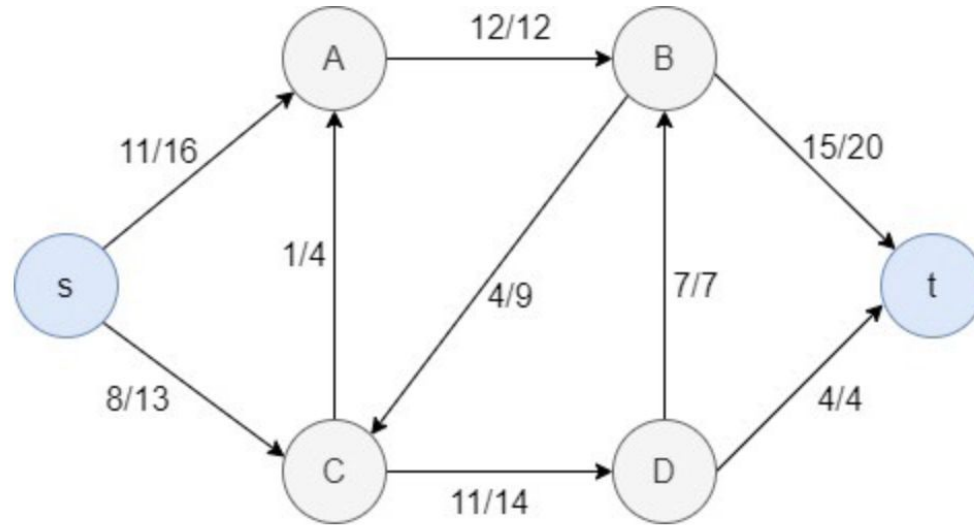
Residual Network

Additional flow = edge's capacity — the flow of the edge

So, to create a residual network we will take our original network and will update the capacity of each edge with the additional flow corresponding to that edge. Then we will also add reverse edges to indicate the amount of flow currently going across the edges of the original network.

Ford-Fulkerson algorithm- example

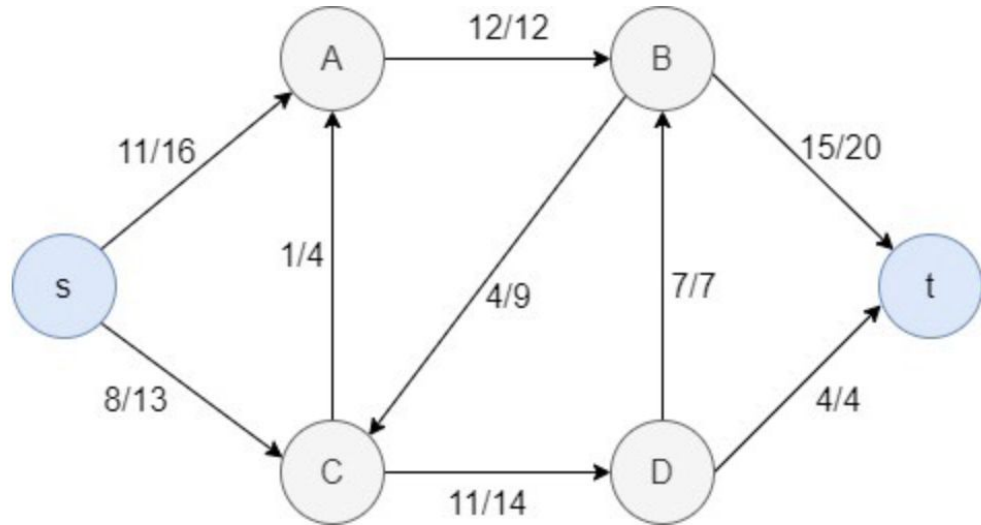
To make this more clear, let's look at the following example.



Flow network

Ford-Fulkerson algorithm- example

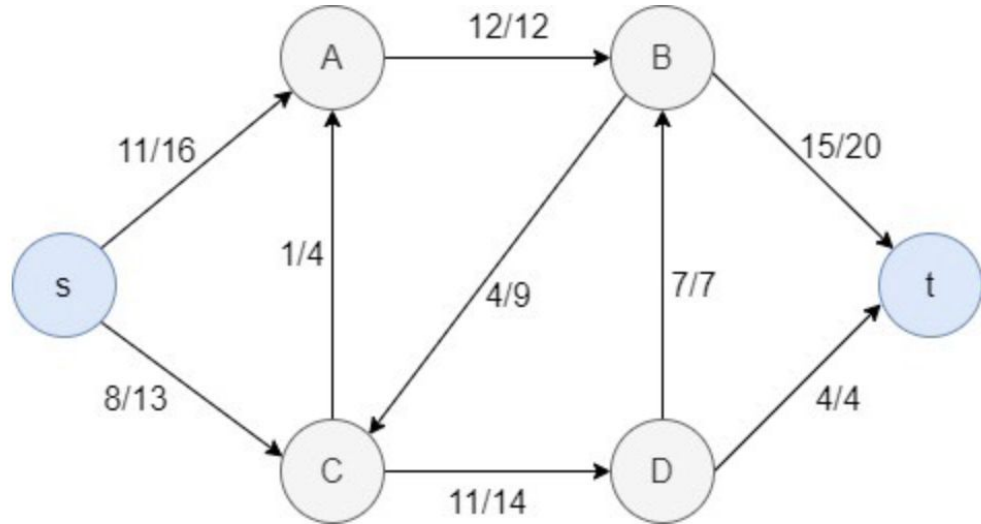
In the example, for each edge, the first value represents the flow value and the second stands for the capacity of the edge.



Flow network

Ford-Fulkerson algorithm- example

Let's consider the edge $C \rightarrow D$. Here the flow is 11 and the capacity of the edge is 14. So, if we calculate the additional flow for this edge, we get $14 - 11 = 3$. So in the residual network, we can update the capacity of the $C \rightarrow D$ edge as 3, and then add a reverse edge with value 11 (flow value of the original network) between C and D.



Flow network

Ford-Fulkerson algorithm- example

Let's consider the edge C->D. Here the flow is 11

and the capacity of the edge is 14. So, if we

calculate the additional flow for this edge, we get

$14 - 11 = 3$. So in the residual network, we can

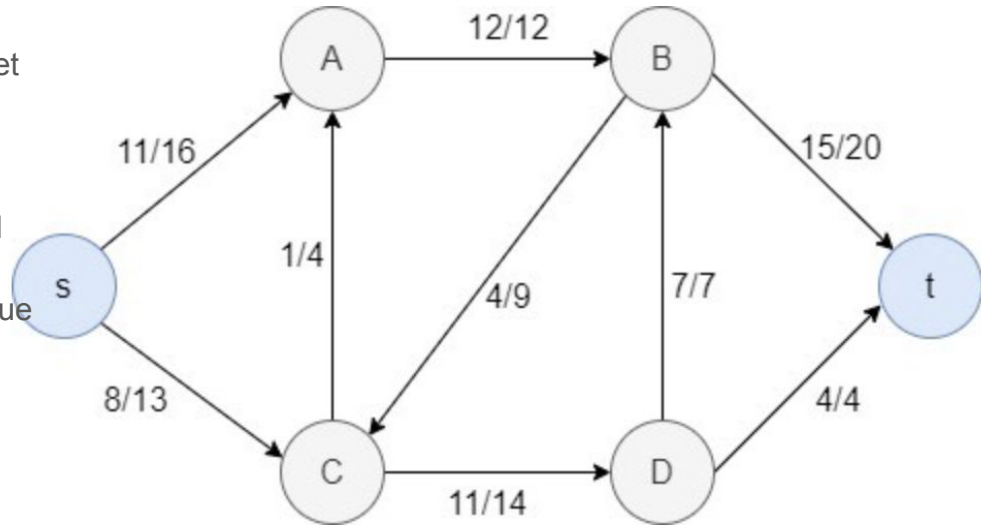
update the capacity of the C->D edge as 3, and

then add a reverse edge with value 11 (flow value

of the original network) between C and D.

Take a moment and do this for the rest of the

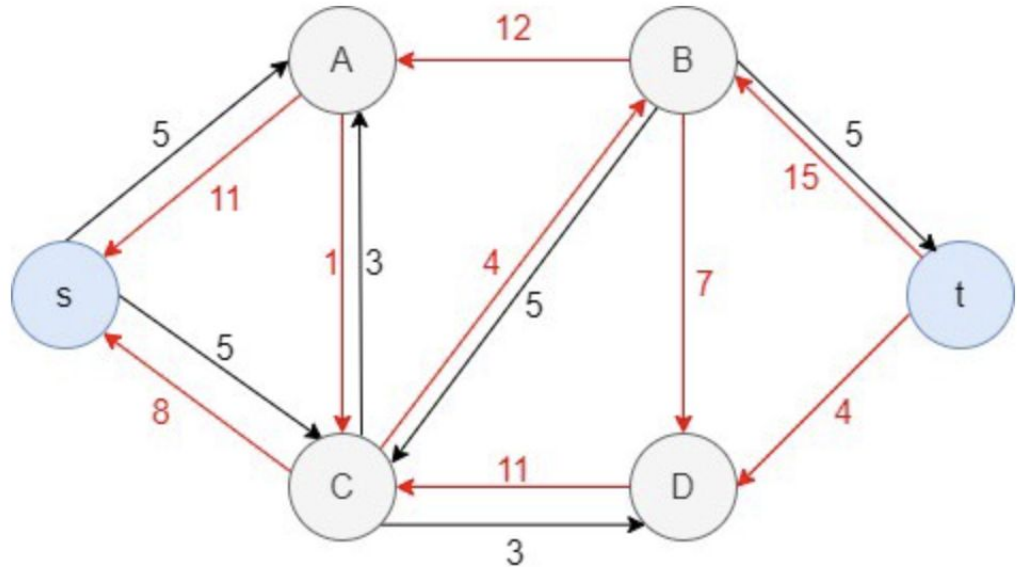
edges



Flow network

Ford-Fulkerson algorithm- example

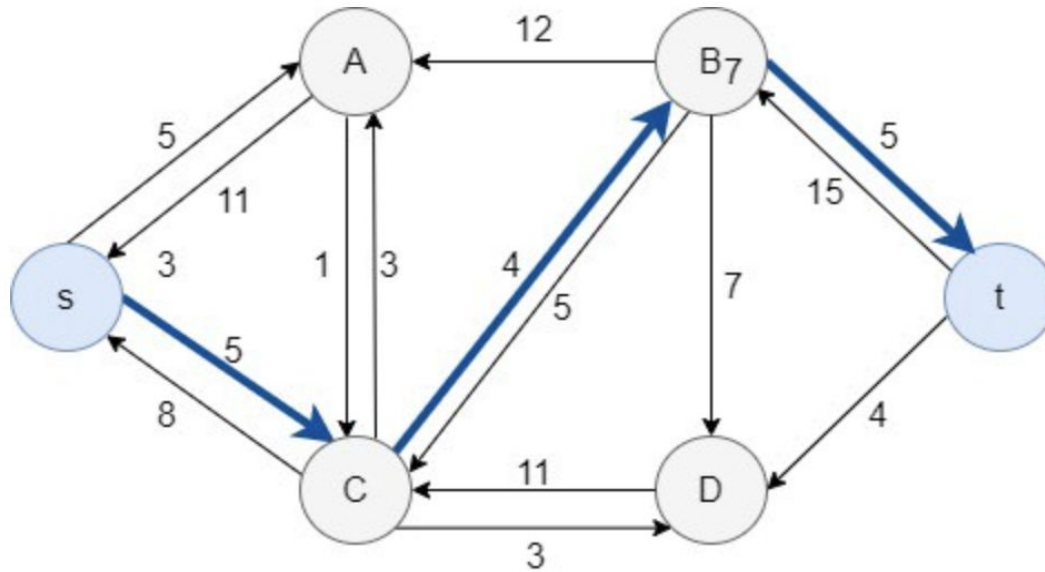
Likewise, we can repeat this for all the edges, and here's what we get as the residual network.



Residual Network (Reverse edges are marked in red)

Augmenting path

Given a flow network $G=(V, E)$, the **augmenting path** is a simple path from s to t in the corresponding residual graph of the flow network.



Residual network with an augmenting path

Ford-Fulkerson algorithm- example

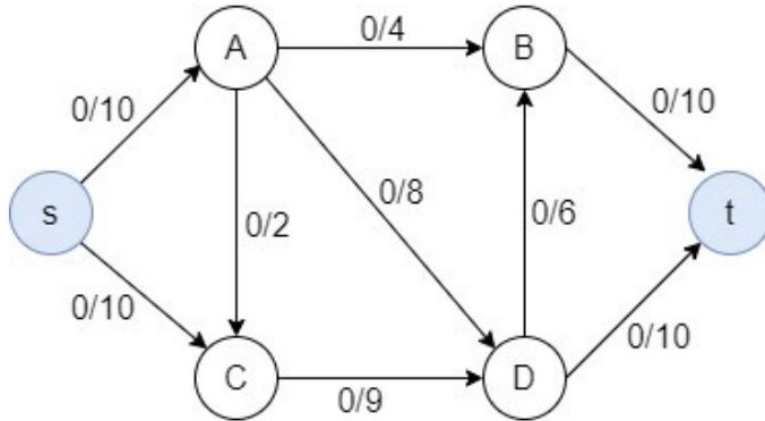
Now we have the basic knowledge to understand the Ford Fulkerson method.

Ford-Fulkerson Method

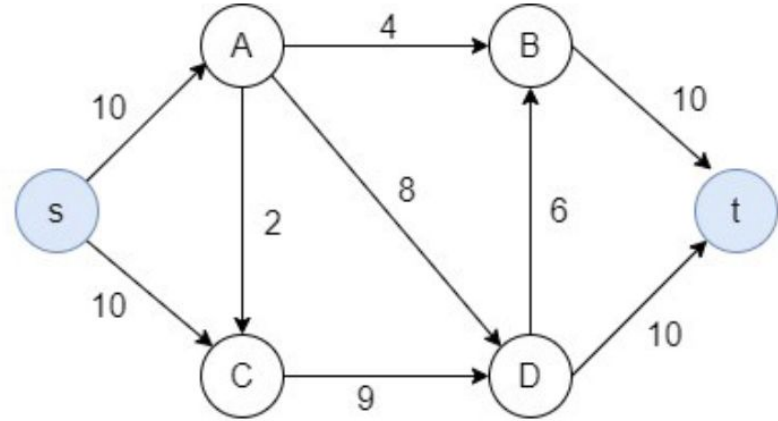
- (1) Initially, set the flow ' f ' of every edge to 0
- (2) While there exists an augmenting path ' p ' in the residual network
 augment the flow ' f ' along ' p '
- (3) return flow ' f '

Ford-Fulkerson algorithm- example

Step 1: set the flow of every edge to 0



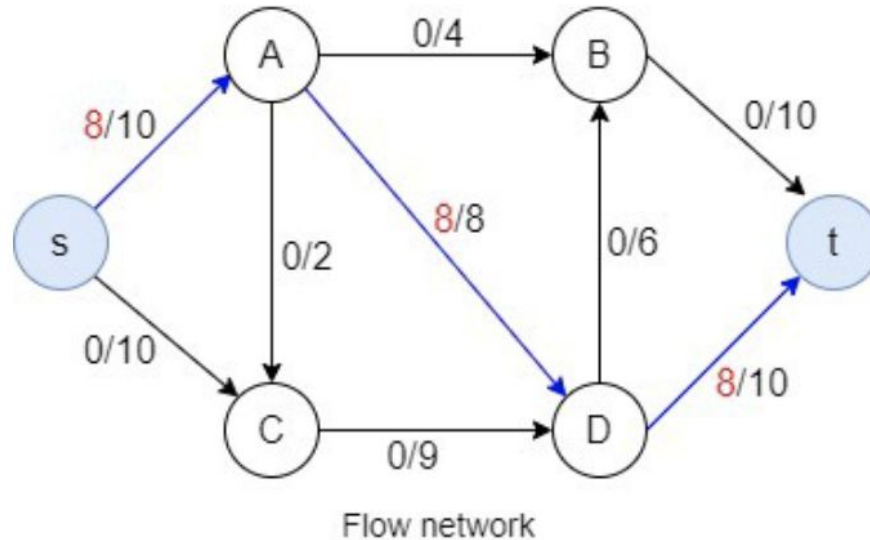
Flow network



Residual network

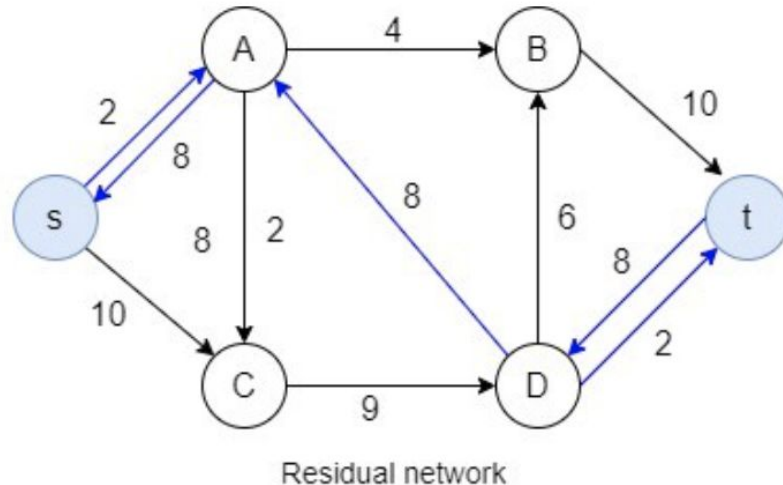
Ford-Fulkerson algorithm- example

Step 2: Now, find an augmenting path in the residual network. Here, I select the path $s \rightarrow A \rightarrow D \rightarrow t$. Then we have to identify the bottleneck capacity (i.e. maximum flow for that path) for the selected path.



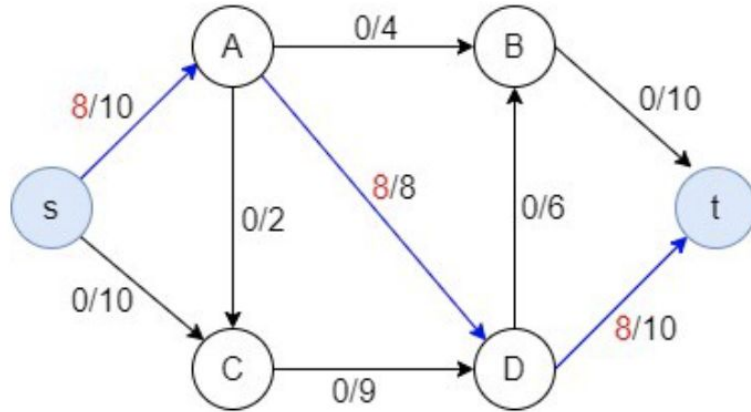
Ford-Fulkerson algorithm- example

Step 2: Now, find an augmenting path in the residual network. Here, I select the path $s \rightarrow A \rightarrow D \rightarrow t$. Then we have to identify the bottleneck capacity (i.e. maximum flow for that path) for the selected path. As you can see, along this path, the bottleneck capacity is 8. Now without violating the capacity constraint, update the flow values of the edges in the augmenting path. Then you will get the following flow network and the residual network.

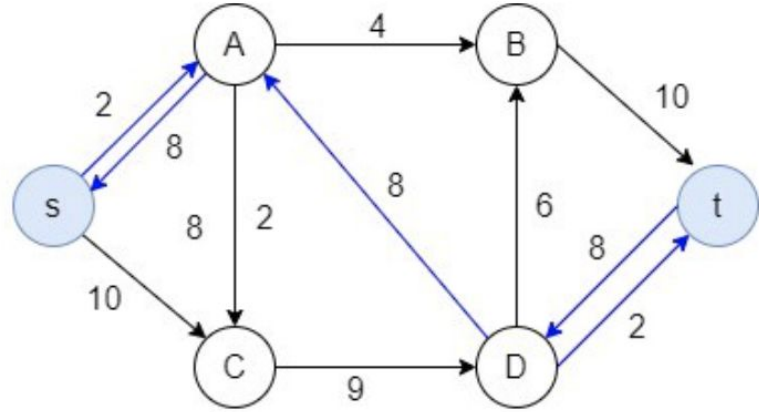


Ford-Fulkerson algorithm- example

Step 3: Then I select the augmenting path $s \rightarrow C \rightarrow D \rightarrow t$. Now what is the bottleneck capacity?



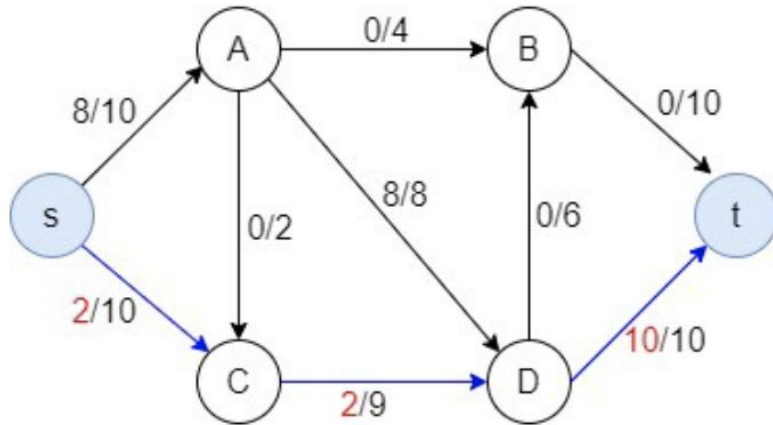
Flow network



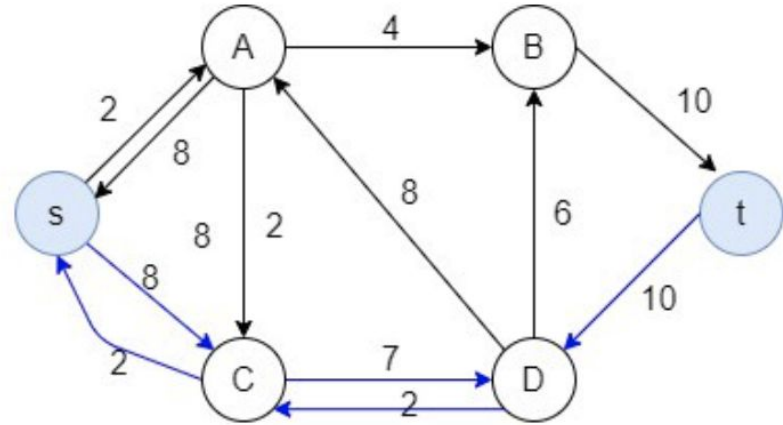
Residual network

Ford-Fulkerson algorithm- example

Step 3: Then I select the augmenting path $s \rightarrow C \rightarrow D \rightarrow t$. Now the bottleneck capacity is 2.



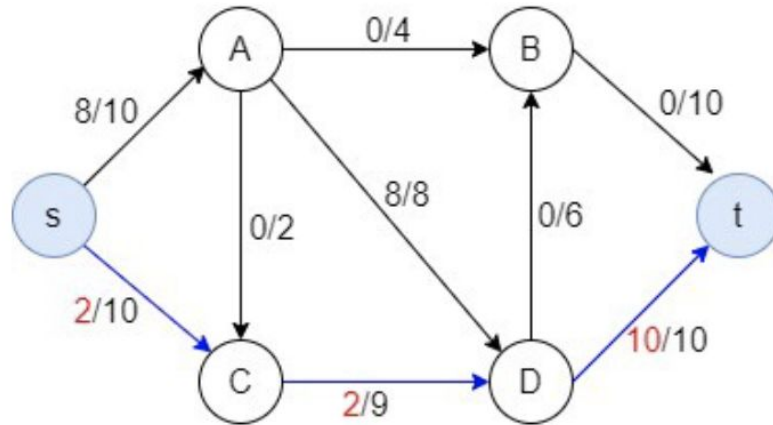
Flow network



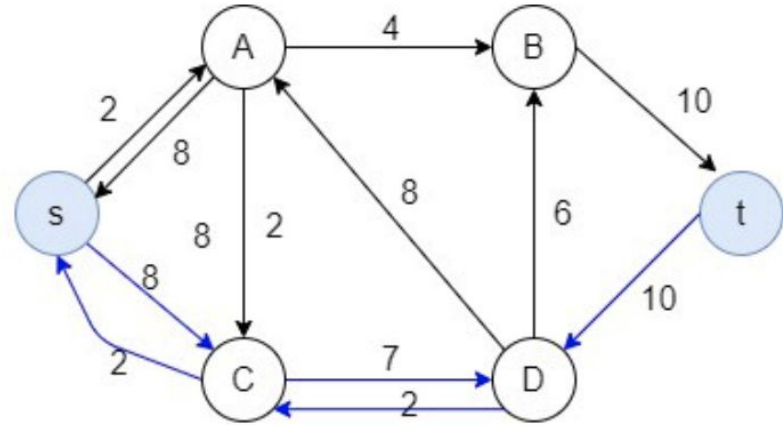
Residual network

Ford-Fulkerson algorithm- example

Step 4: The augmenting path $s \rightarrow A \rightarrow B \rightarrow t$, what is the bottleneck capacity?



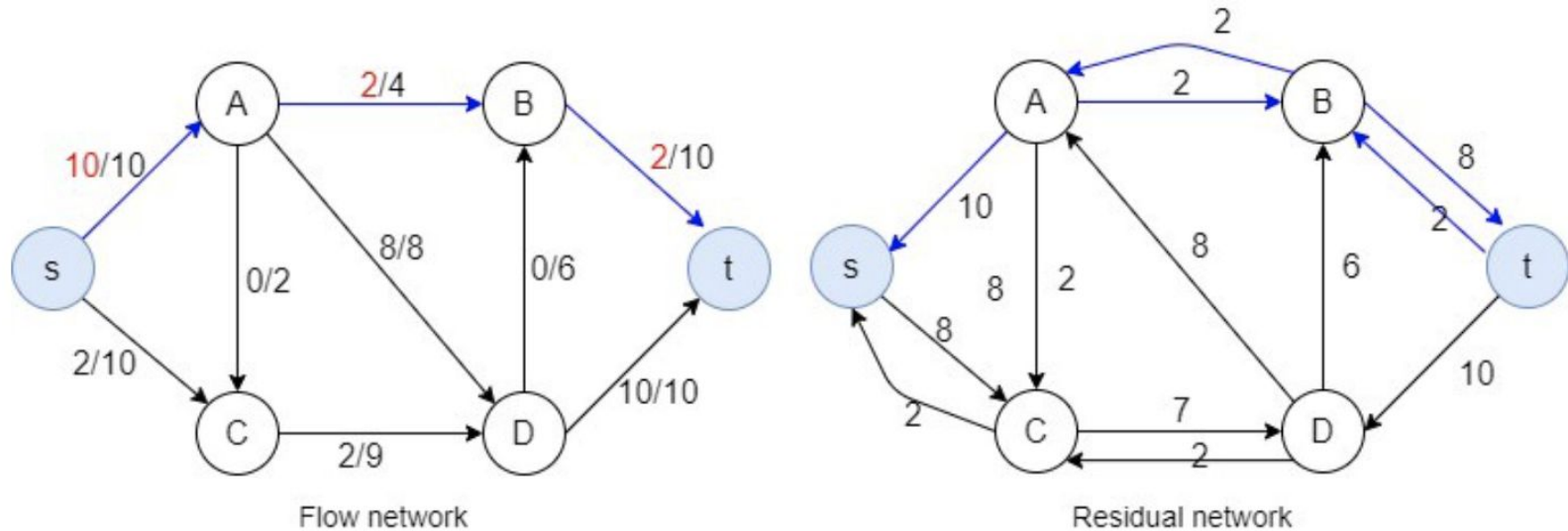
Flow network



Residual network

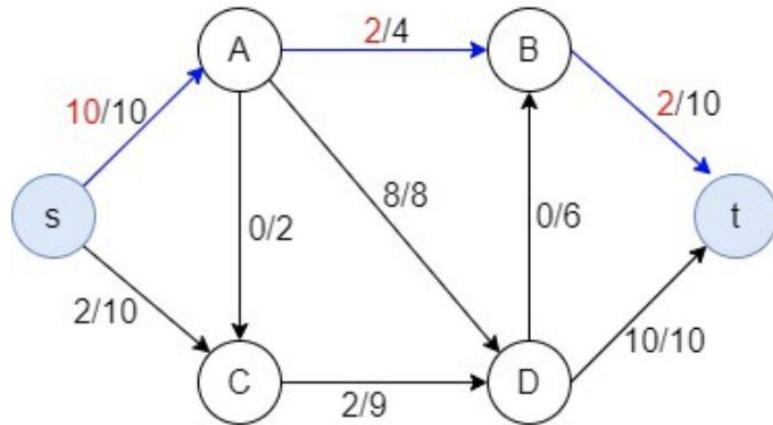
Ford-Fulkerson algorithm- example

Step 4: The augmenting path $s \rightarrow A \rightarrow B \rightarrow t$, the bottleneck capacity is 2.

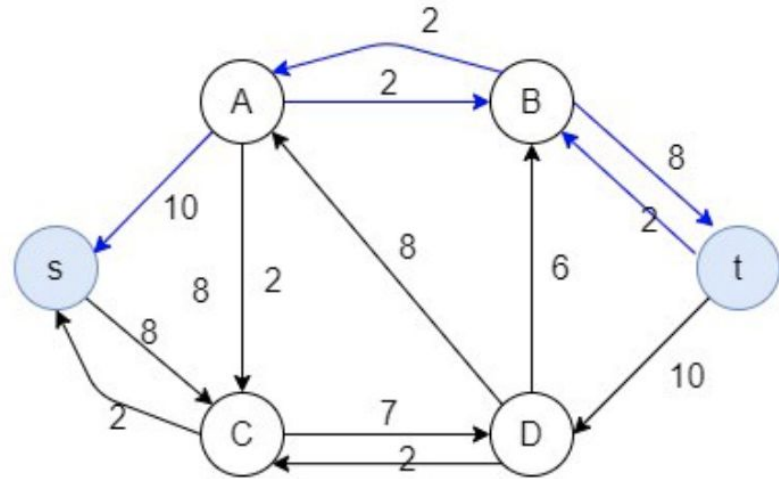


Ford-Fulkerson algorithm- example

Step 5: augmenting path $s \rightarrow C \rightarrow D \rightarrow B \rightarrow t$, what is the bottleneck capacity?



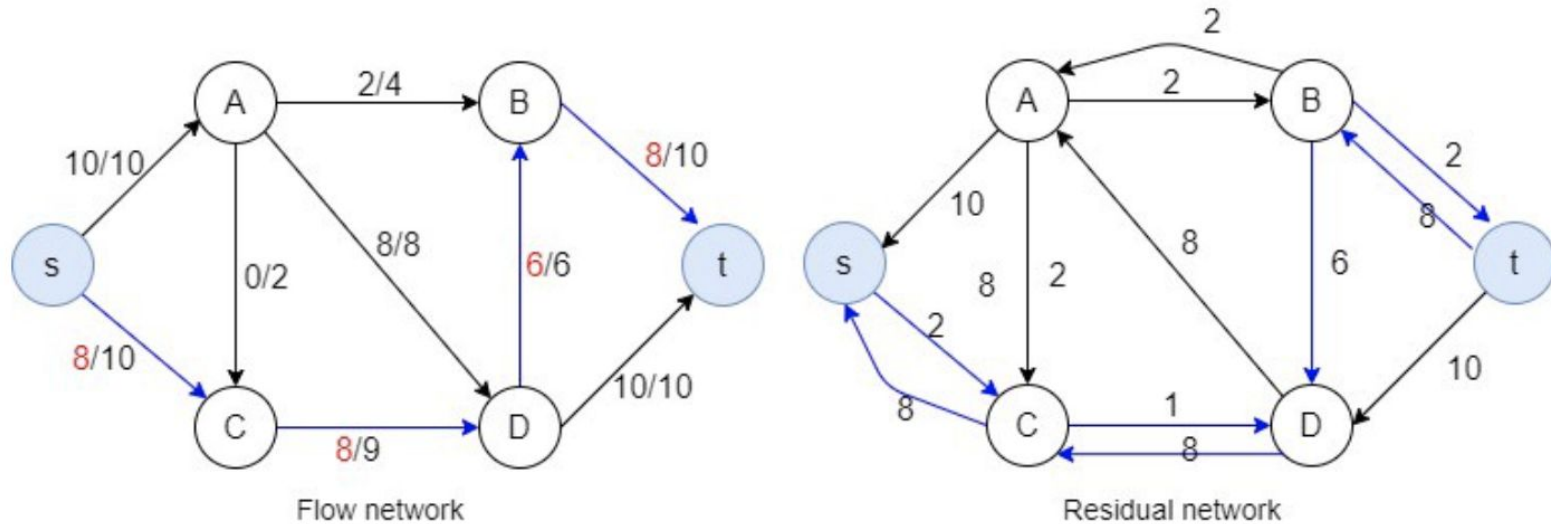
Flow network



Residual network

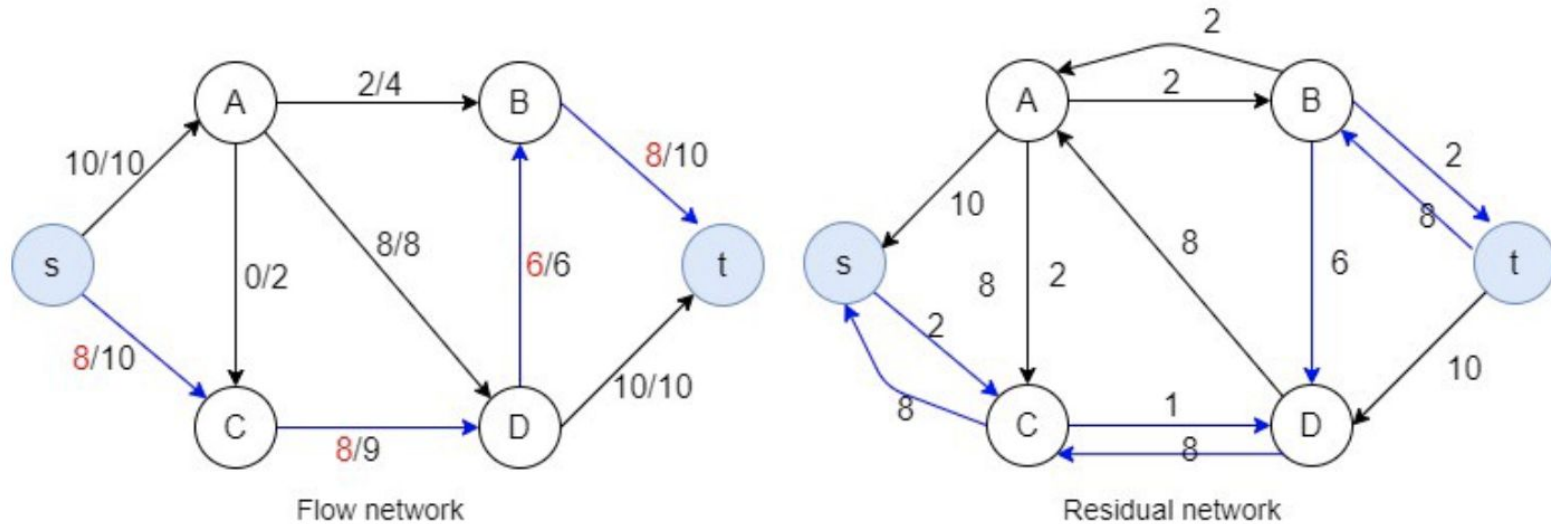
Ford-Fulkerson algorithm- example

Step 5: augmenting path $s \rightarrow C \rightarrow D \rightarrow B \rightarrow t$, the bottleneck capacity is 6.



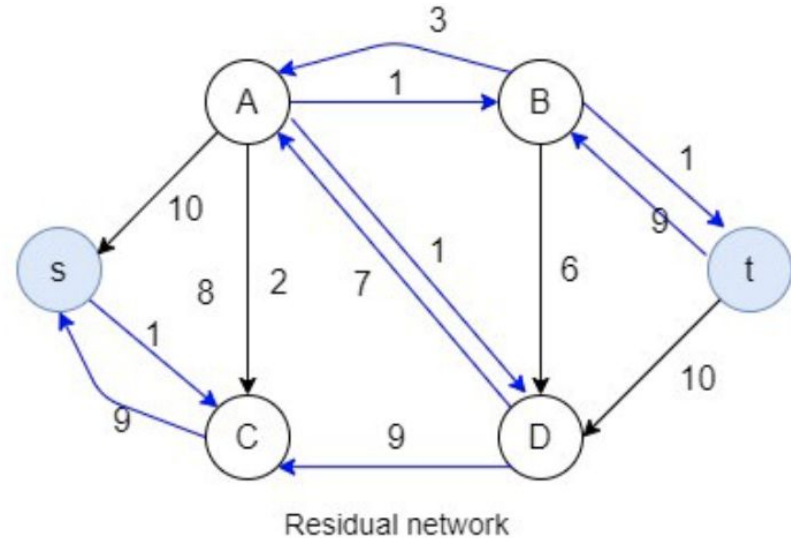
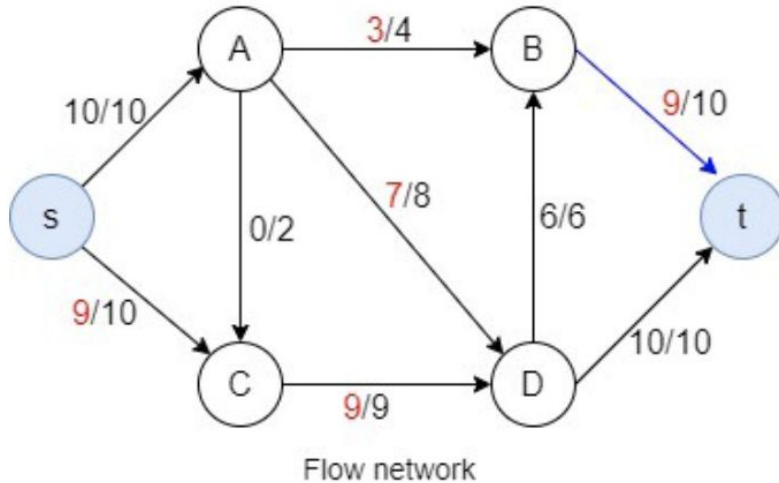
Ford-Fulkerson algorithm- example

Step 6: augmenting path $s \rightarrow C \rightarrow D \rightarrow A \rightarrow B \rightarrow t$, what is the bottleneck capacity?



Ford-Fulkerson algorithm- example

Step 6: augmenting path $s \rightarrow C \rightarrow D \rightarrow A \rightarrow B \rightarrow t$, bottleneck capacity is 1.



Ford-Fulkerson algorithm- example

Look! Now there are no paths left from the s to t in the residual graph. So, there is no possibility to add flow. This means the Ford Fulkerson method is complete and we are ready to find the maximum flow.

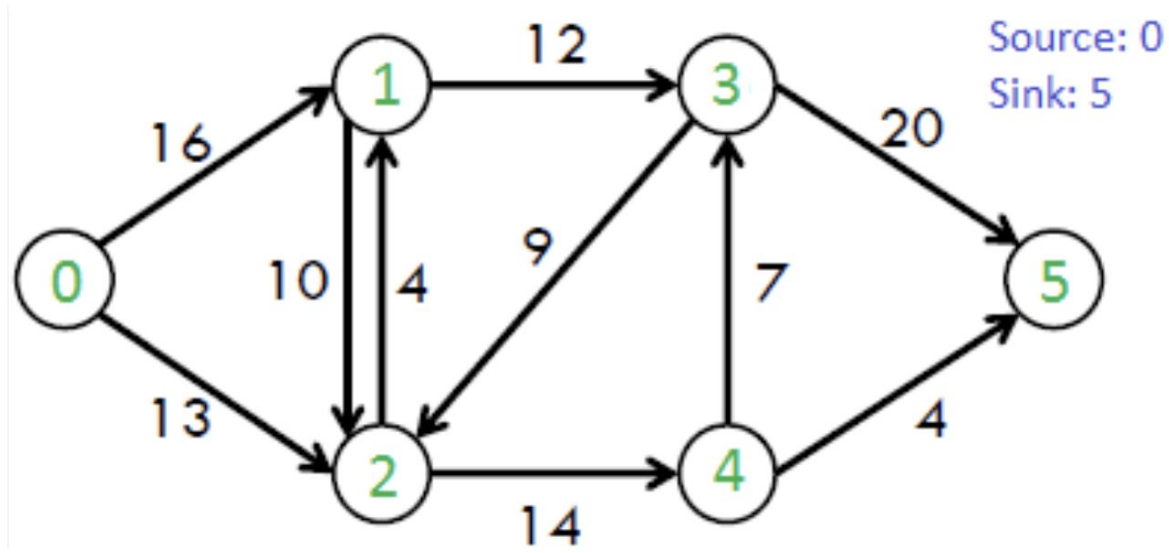
Ford-Fulkerson algorithm- example

Look! Now there are no paths left from the s to t in the residual graph. So, there is no possibility to add flow. This means the Ford Fulkerson method is complete and we are ready to find the maximum flow.

Since the maximum flow is equal to the flow coming out of the source, in this example, the maximum flow is $10+9 = 19$.

Code

Python script is for this diagram



Homework problems!

Please complete exercise 10.2 #7