What is your course level?

- ○ Undergraduate
- ○ Masters
- ○ Doctoral
- ○ __________ Other

What is your major?

- ○ Computer Science
- ○ Computer or Electrical Engineering
- ○ Mathematics
- ○ __________ Other

Please list any courses related to algorithms you have taken (including those in progress):

Please list any courses related to algorithms you have been a TA for (including those in progress):

FF-Rate Understanding

Rate your understanding of the Ford Fulkerson algorithm to compute the maximum flow on a graph. Check the option that you feel applies best to you.

- ○ I have never heard of this algorithm.
- ○ I have heard of the algorithm, but do not know it.
- ○ I know the algorithm, but do not understand it.
- ○ I know the algorithm and understand it a little bit.
- ○ I know the algorithm and have a fair understanding of it.
- ○ I understand the algorithm completely.

Euclidean-Rate Understanding

Rate your understanding of the Euclidean Algorithm to compute the greatest common divisor of two integers. Check the option that you feel applies best to you.

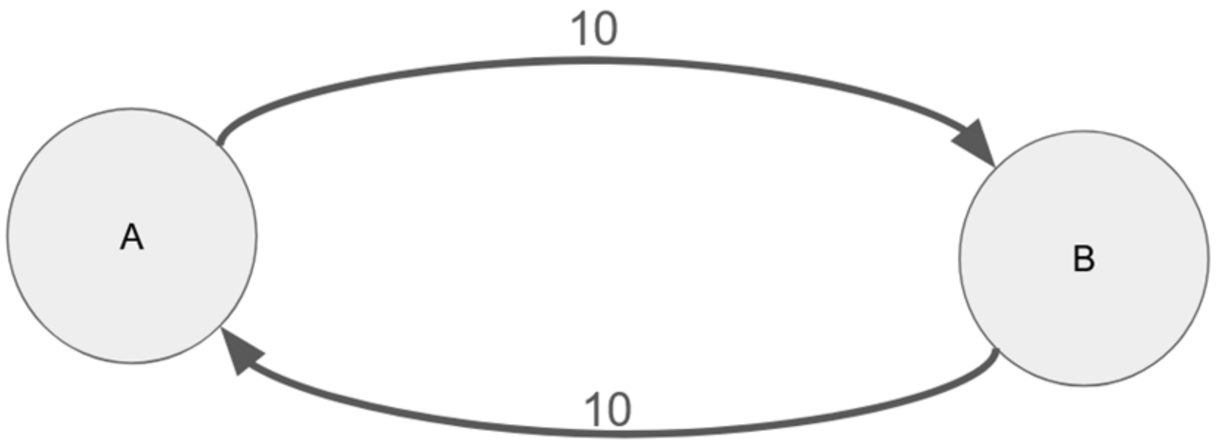
- ○ I have never heard of this algorithm.
- ○ I have heard of the algorithm, but do not know it.
- ○ I know the algorithm, but do not understand it.
- ○ I know the algorithm and understand it a little bit.
- ○ I know the algorithm and have a fair understanding of it.
- ○ I understand the algorithm completely.

Flow V1

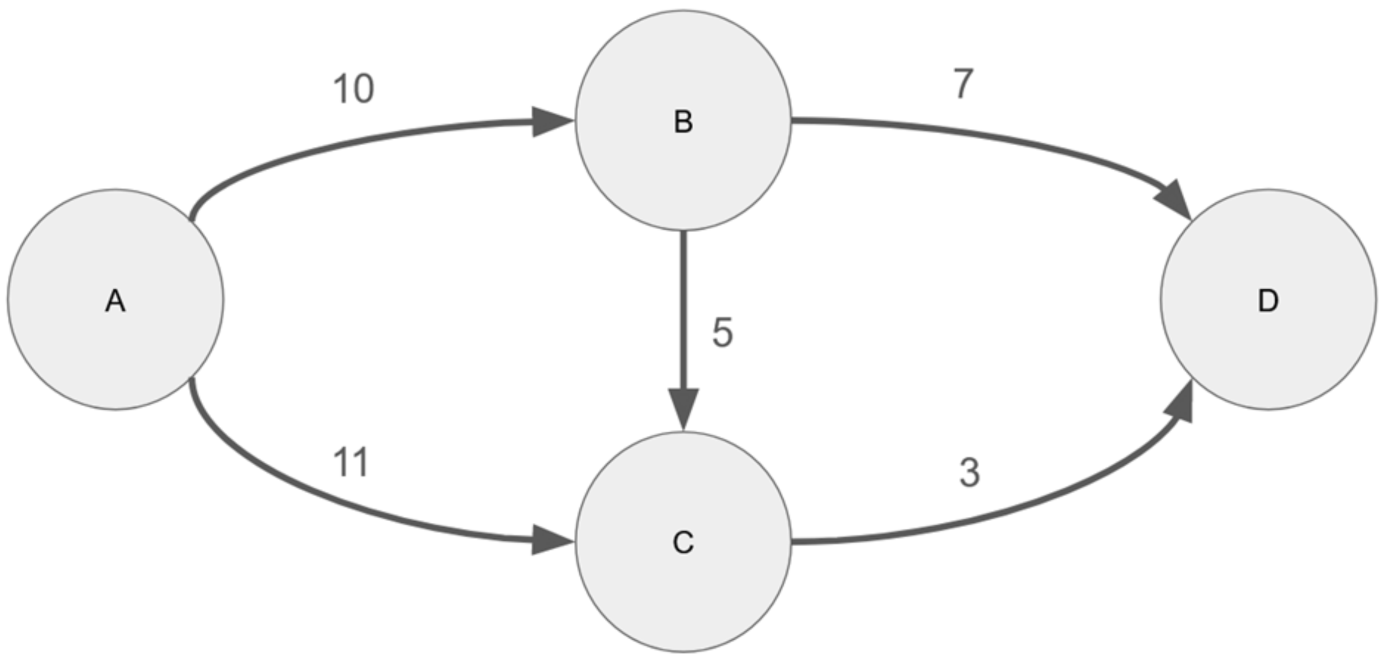
This section of the survey will test your ability to answer various problems related to the Ford Fulkerson algorithm for computing the maximum flow between two vertices. Please answer the questions to the best of your ability. You are free to skip any questions.

For the following two questions, compute the maximum flow using the Ford Fulkerson Algorithm.

Compute the maximum flow between A and B. List each augmenting path and the flow along the path at each step.



Compute the maximum flow between A and D. List each augmenting path and the flow along the path at each step.


A
B
C
D
10
11
7
5
3

The following is an implementation of breadth-first search in Python with one line missing. Write a condition to replace the highlighted text. If you are not familiar with Python syntax, you may use your best guess.

```python
def BFS(G, s, t):
    visited = [False]*G.shape[0]
    visited[s] = True
    q = [s]
    parents = [-1]*G.shape[0]
    while len(q) > 0:
        current = q[0]
        q = q[1:]
        for index, capacity in enumerate(G[current]):
            if <CONDITION>:
                q.append(index)
                visited[index] = True
                parents[index] = current
                if index == t:
                    return parents
    #If a path from s to t was not found, return None
    return None
```

Give an example of a graph where the Ford Fulkerson algorithm computes exactly six augmenting flows before terminating. Write the example as a list of edges and capacities.

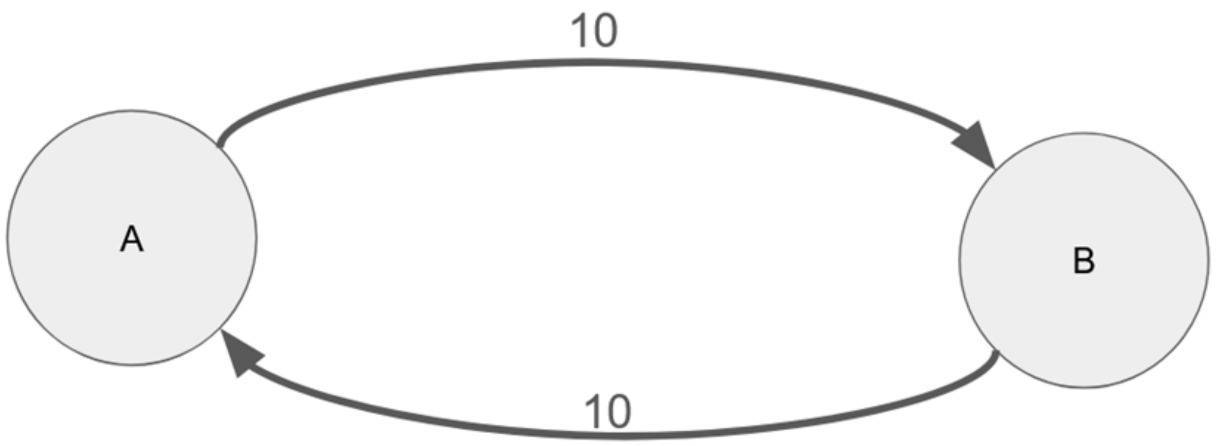
Consider an implementation of Ford Fulkerson which uses a breadth-first search to find the augmenting path. Give an example which illustrates why the algorithm needs to track residual capacity of reverse edges.

Flow V2

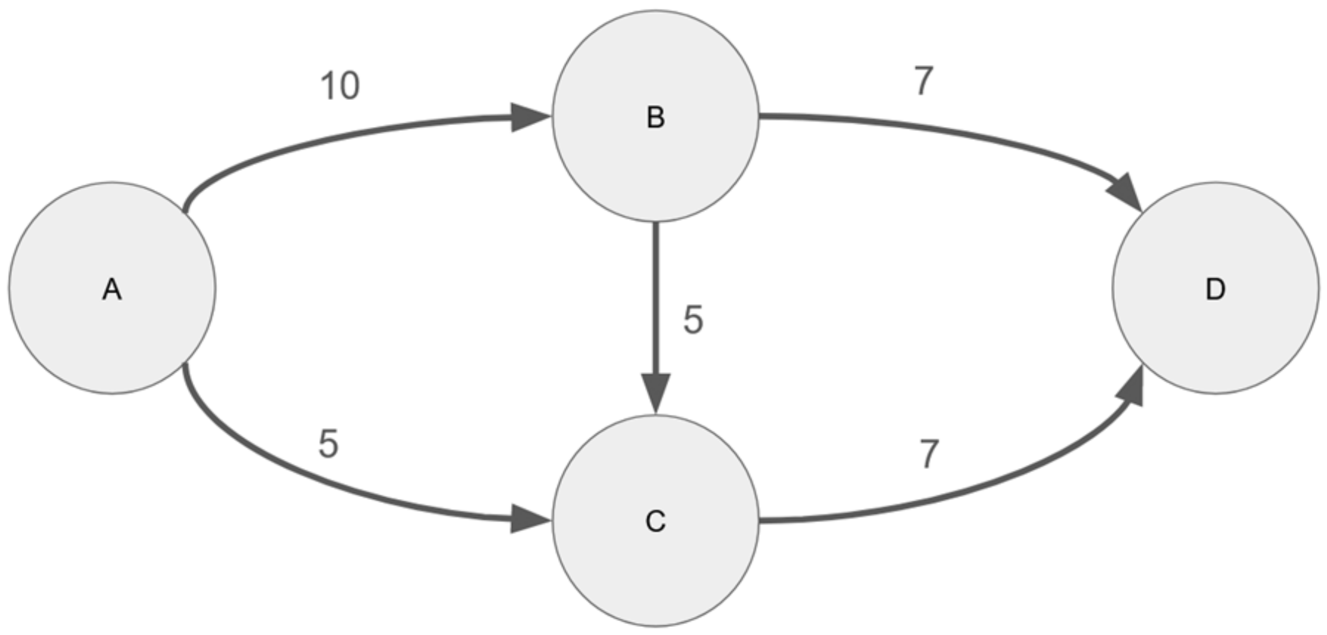
This section of the survey will test your ability to answer various problems related to the Ford Fulkerson algorithm for computing the maximum flow between two vertices. Please answer the questions to the best of your ability. You are free to skip any questions.

For the following two questions, compute the maximum flow using the Ford Fulkerson Algorithm.

Compute the Maximum Flow between A and B. List each augmenting path and the flow along the path at each step.



Compute the Maximum Flow between A and D. List each augmenting path and the flow along the path at each step.



The following is an implementation of Ford Fulkerson in Python with two lines missing. Write code to replace the highlighted text. If you are not familiar with Python syntax, you may use your best guess

```
def FF(G,s,t):
    parents = BFS(G, s, t)
    max_flow = 0
    while parents is not None:
        flow_to_send = np.inf
        current = t
        while current != s:
            <INSERT CODE HERE>

        current = t
        while current != s:
            G[parents[current], current] -= flow_to_send
            G[current, parents[current]] += flow_to_send
            current = parents[current]
        max_flow +=flow_to_send

        parents = BFS(G, s, t)
    return max_flow
```

You are a civil engineer at a city planning commission. Your boss has asked you whether it is possible to calculate the maximum volume of traffic that can be routed between two destinations in the city. Describe the Ford Fulkerson algorithm, but since he is a busy man, do not bore him with the details.

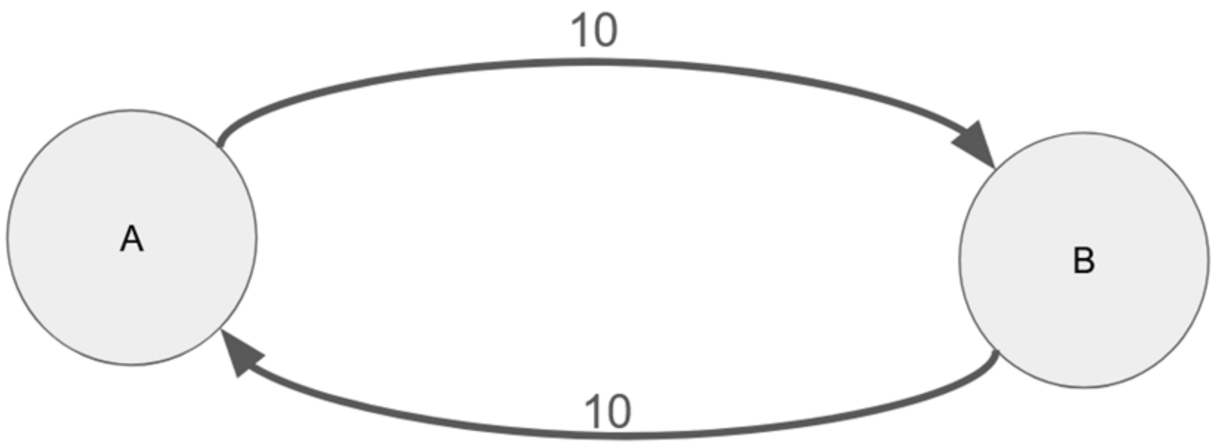
Suppose you are an event planner, and you need to determine the maximum amount of traffic that can be routed from the airport to one of two event venues. In other words, you want to compute the maximum flow between a source *s* and two sinks *t1* and *t2.* How would you implement this?

Flow V3

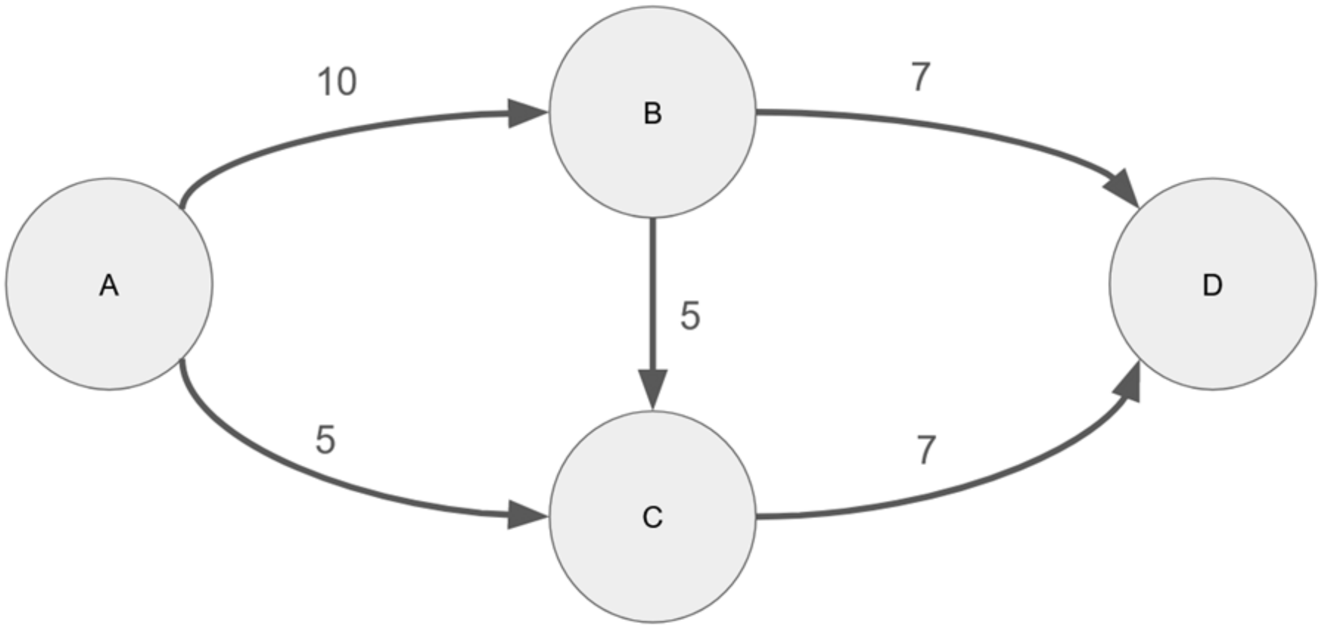
This section of the survey will test your ability to answer various problems related to the Ford Fulkerson algorithm for computing the maximum flow between two vertices. Please answer the questions to the best of your ability. You are free to skip any questions.

For the following two questions, compute the maximum flow using the Ford Fulkerson Algorithm.

Compute the Maximum Flow between A and B. List each augmenting path and the flow along the path at each step.



Compute the Maximum Flow between A and D. List each augmenting path and the flow along the path at each step.



The following is an implementation of Ford Fulkerson in Python with three lines missing. Write code to replace the highlighted text. If you are not familiar with Python syntax, you may use your best guess.

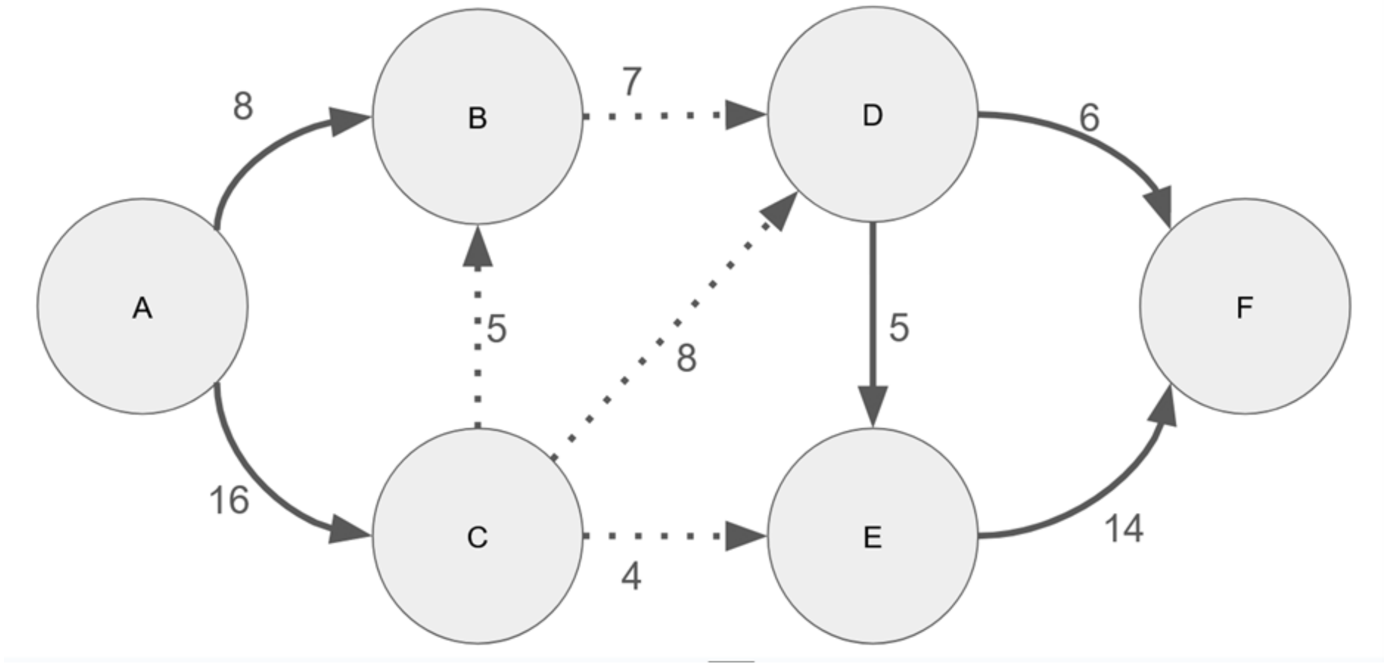
```
def FF(G,s,t):
    parents = BFS(G, s, t)
    max_flow = 0
    while parents is not None:
        flow_to_send = np.inf
        current = t
        while current != s:
            flow_to_send = min([flow_to_send, G[parents[current], current]])
            current = parents[current]

        current = t
        while current != s:
            <INSERT CODE HERE>
        max_flow +=flow_to_send

        parents = BFS(G, s, t)
    return max_flow
```

You are assisting a student in an algorithms course. He is struggling to understand the Ford-Fulkerson algorithm for computing MAXFLOW. Describe it to him, prioritizing conveying intuition.

An *s-t cut* is a set of edges which, when removed, divide vertex *s* and vertex *t* into separate components. For example, the dotted edges in the graph below are an A-F cut with capacity 24.



It turns out that the maximum *s-t* flow on a graph with capacities is equal to the minimum capacity of any *s*-*t* cut. Describe your intuition for why this might be true.

Feedback

Please provide any feedback related to this survey. Were there any questions that were ambiguous or confusing?

Euclidean V1

This section of the survey will test your ability to answer various problems related to the Euclidean algorithm. Please answer the questions to the best of your ability. You are free to skip any questions.

For the next two questions, compute the greatest common divisor (GCD) of the two integers.

Compute GCD(24, 15), and show each step of the algorithm.

Compute GCD(462, 946), and show each step of the algorithm.

Using a programming language you are familiar with, write code for a function gcd(a,b), which computes the greatest common divisor of two integers a and b using the Euclidean algorithm.

You are instructing a student in an algebra course. The student is familiar with basic mathematical operations such as addition, subtraction, multiplication and division. However, she struggles with algebraic equations and division with remainders. Explain the Euclidean algorithm to her, prioritizing conveying intuition.

Suppose gcd(a,b) = x and gcd(b,c) = y. Does gcd(a,b,c) = gcd(x,y)? Explain your reasoning.

Euclidean V2

This section of the survey will test your ability to answer various problems related to the Euclidean algorithm. Please answer the questions to the best of your ability. You are free to skip any questions.

For the next two questions, compute the greatest common divisor (GCD) of the two integers.

Compute GCD(24, 15), and show each step of the algorithm.

Compute GCD(4088, 1241), and show each step of the algorithm.

Using a programming language you are familiar with, write code for a function gcd(a,b), which computes the greatest common divisor of two integers a and b using the Euclidean algorithm.

Consider computing GCD(55, x) for an input x. Give an integer 0<x<55 that requires the greatest number of recursive steps to compute GCD(55, x). Describe how you chose this number.

Determine whether the following statement is true. If not, provide a counterexample.

For any two positive, nonzero integers a,b, the equation sa + tb = GCD(a,b) has exactly one solution where s and t are integers.

Euclidean V3

This section of the survey will test your ability to answer various problems related to the Euclidean algorithm. Please answer the questions to the best of your ability. You are free to skip any questions.

For the next two questions, compute the greatest common divisor (GCD) of the two integers. Describe each step of the algorithm and all intermediate computations.

Compute GCD(24, 15), and show each step of the algorithm.

Compute GCD(1008, 468), and show each step of the algorithm.

Using a programming language you are familiar with, write code for a function gcd(a,b), which computes the greatest common divisor of two integers a and b using the Euclidean algorithm.

You are speaking with a mathematics student who understands modular arithmetic. Explain the proof that the Euclidean algorithm finds the greatest common divisor of two numbers.

The Fibonacci numbers are a sequence F(n) where F(0) = 0, F(1) = 1, and F(n) = F(n-1) + F(n-2) for any n>= 2. Consecutive Fibonacci numbers can be thought of as 'worst case' inputs for the Euclidean algorithm. Can you explain why?