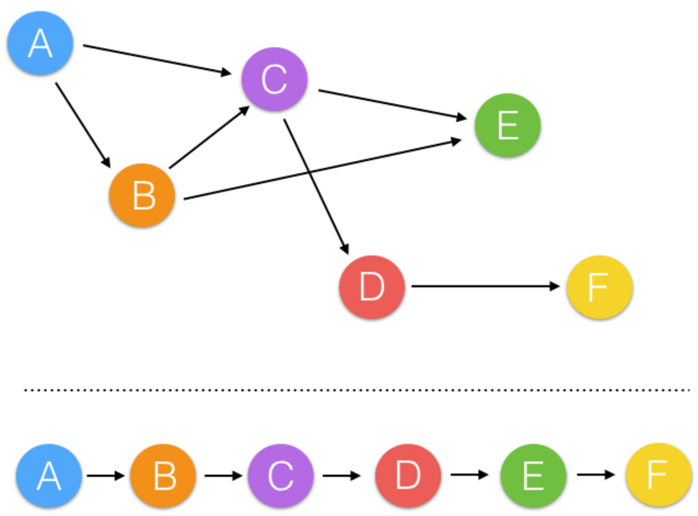
Forward propagation

`MiniFlow` has two methods to help you define and then run values through your graphs: `topological_sort()`and `forward_pass()`.



An example of topological sorting

In order to define your network, you'll need to define the order of operations for your nodes. Given that the input to some node depends on the outputs of others, you need to flatten the graph in such a way where all the input dependencies for each node are resolved before trying to run its calculation. This is a technique called a topological sort.

The `topological_sort()`function implements topological sorting using Kahn's Algorithm. The details of this method are not important, the result is; `topological_sort()`returns a sorted list of nodes in which all of the calculations can run in series. `topological_sort()`takes in a `feed_dict`, which is how we initially set a value for an `Input` node. The `feed_dict` is represented by the Python dictionary data structure. Here's an example use case:

```
# Define 2 `Input` nodes.
```

```
x, y = Input(), Input()

# Define an `Add` node, the two above`Input` nodes being the input.
add = Add(x, y)

# The value of `x` and `y` will be set to 10 and 20 respectively.
feed_dict = {x: 10, y: 20}

# Sort the nodes with topological sort.
sorted_nodes = topological_sort(feed_dict=feed_dict)
```

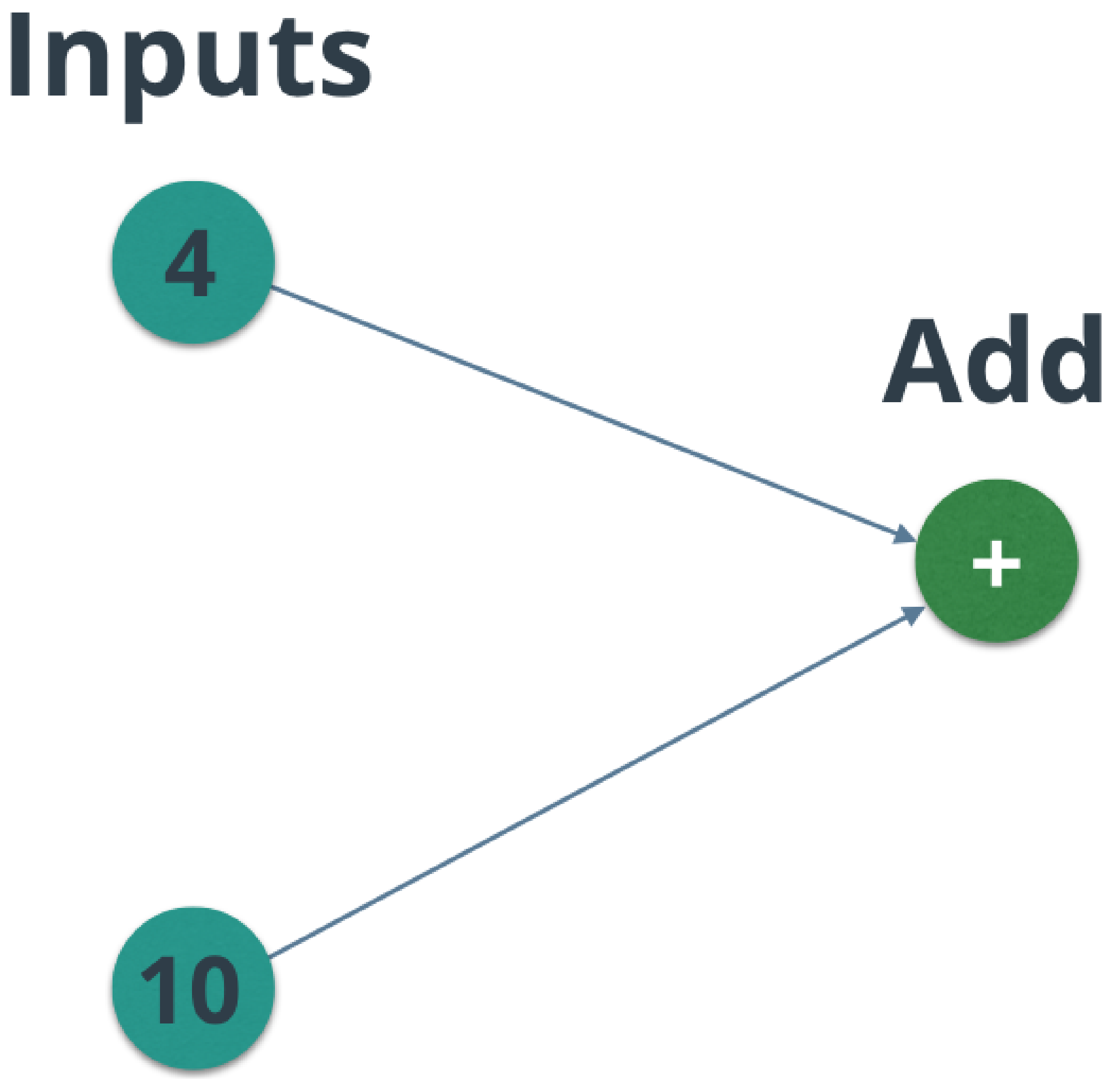
(You can find the source code for `topological_sort()`in miniflow.py in the programming quiz below.)

The other method at your disposal is `forward_pass()`, which actually runs the network and outputs a value.

```
def forward_pass(output_node, sorted_nodes):
    """
    Performs a forward pass through a list of sorted nodes.

    Arguments:

        `output_node`: The output node of the graph (no outgoing edges).
        `sorted_nodes`: a topologically sorted list of nodes.

    Returns the output node's value
    """

    for n in sorted_nodes:
        n.forward()

    return output_node.value
```

Quiz 1 - Passing Values Forward

Create and run this graph!



The graph you'll run in this quiz. The node values may change, though!

Setup

Review `nn.py` and `miniflow.py`.

The neural network architecture is already there for you in nn.py. It's your job to finish `MiniFlow` to make it work.

For this quiz, I want you to:

1. Open `nn.py`below. **You don't need to change anything.** I just want you to see how `MiniFlow`works.
2. Open `miniflow.py`. **Finish the `forward`method on the `Add` class. All that's required to pass this quiz is a correct implementation of `forward`.**
3. Test your network by hitting "Test Run!" When the output looks right, hit "Submit!"

(You'll find the solution on the next page.)