Section 5:
HW6 and Interfaces

SLIDES ADAPTED FROM ALEX MARIAKAKIS,

WITH MATERIAL FROM KRYSTA YOUSOUFIAN, MIKE ERNST, KELLEN DONOHUE

How is Hmwrk 5 going?

Agenda

BFS

Interfaces

Parsing Marvel Data

Reminders:

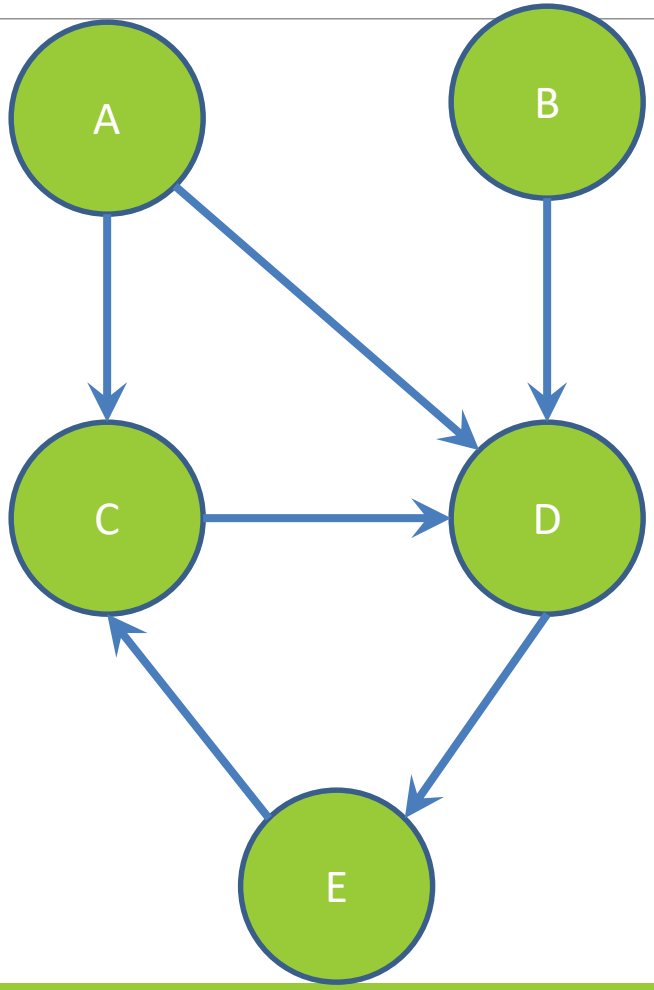
Expensive CheckReps == BAD

(at least when assignments are turned in, but can be useful for finding hard-to-discover problems – so need to be able to control expensive checks)

Debug flags == GOOD

(or enums to indicate depth of debug)

Graphs



Can I reach B from A?

Breadth-First Search (BFS)

Often used for discovering connectivity

Calculates the shortest path *if and only if* all edges have same positive or no weight

Depth-first search (DFS) is commonly mentioned with BFS

- BFS looks “wide”, DFS looks “deep”
- Can also be used for discovery, but not the shortest path

BFS Pseudocode

```
public boolean find(Node start, Node end) {
      put start node in a queue
      while (queue is not empty) {
            pop node N off queue
            if (N is goal)
                  return true;
            else {
                  for each node O that is child of N
                        push O onto queue
            }
      }
      return false;
}
```

Breadth-First Search

START:

Q: <A>

Pop: A, Q: <>

Q: <B, C>

Pop: B, Q: <C>

Q: <C>

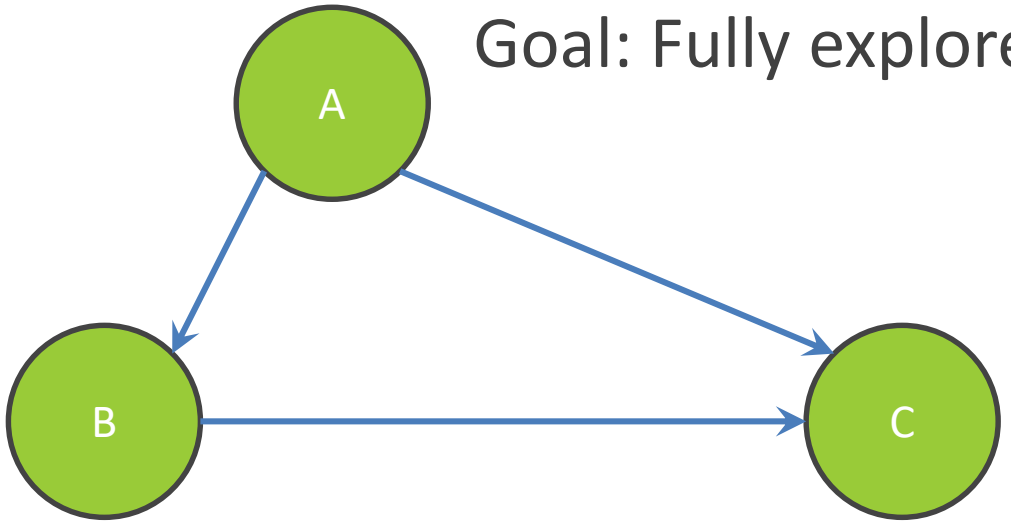
Pop: C, Q: <C>

Q: <>

DONE

Starting at A

Goal: Fully explore



Breadth-First Search with Cycle

START:

Q: <A>

Pop: A, Q: <>

Q: <B>

Pop: B, Q: <>

Q: <C>

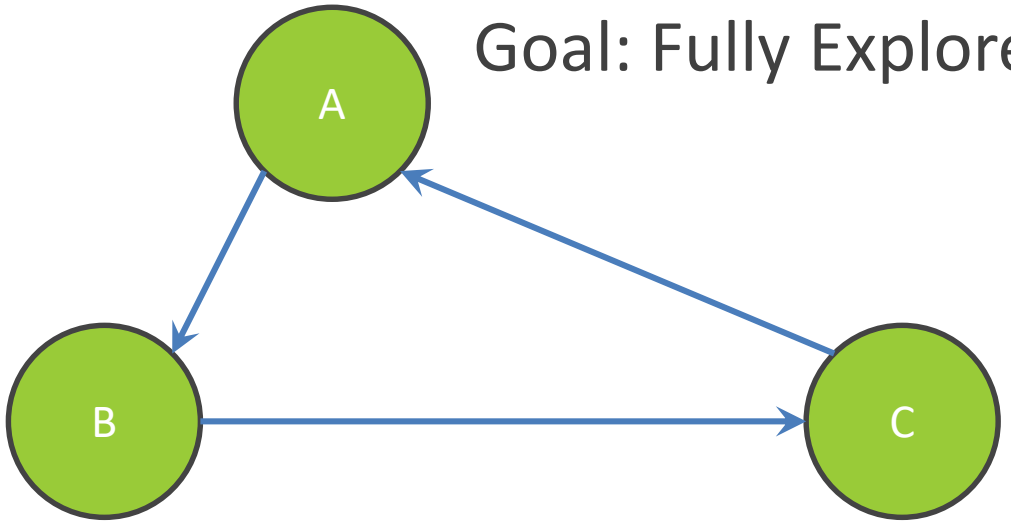
Pop: C, Q: <>

Q: <A>

NEVER DONE

Starting at A

Goal: Fully Explore



BFS Pseudocode

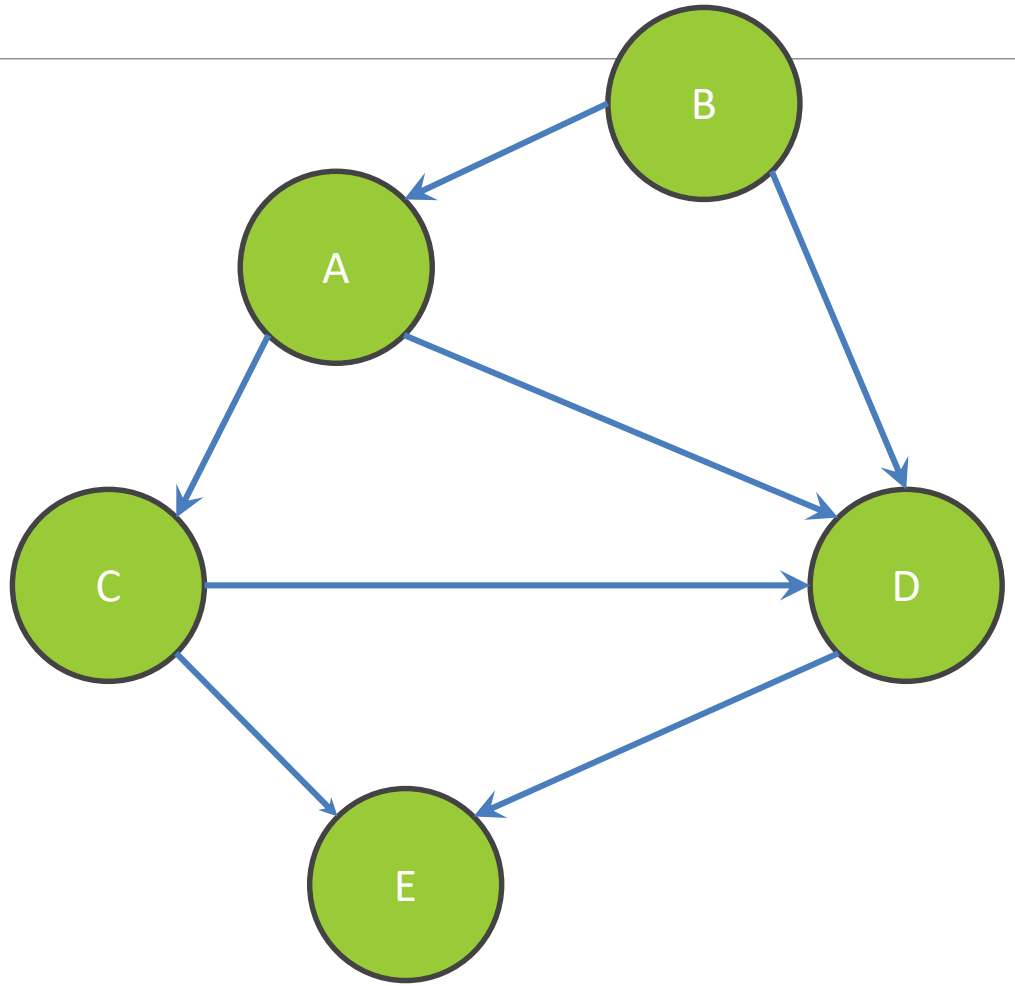
```
public boolean find(Node start, Node end) {
      put start node in a queue
      while (queue is not empty) {
            pop node N off queue
            mark node N as visited

            if (N is goal)
                  return true;
            else {
                  for each node O that is child of N
                        if O is not marked visited
                              push O onto queue
            }
      }
      return false;
}
```

Mark the node as visited!

Breadth-First Search

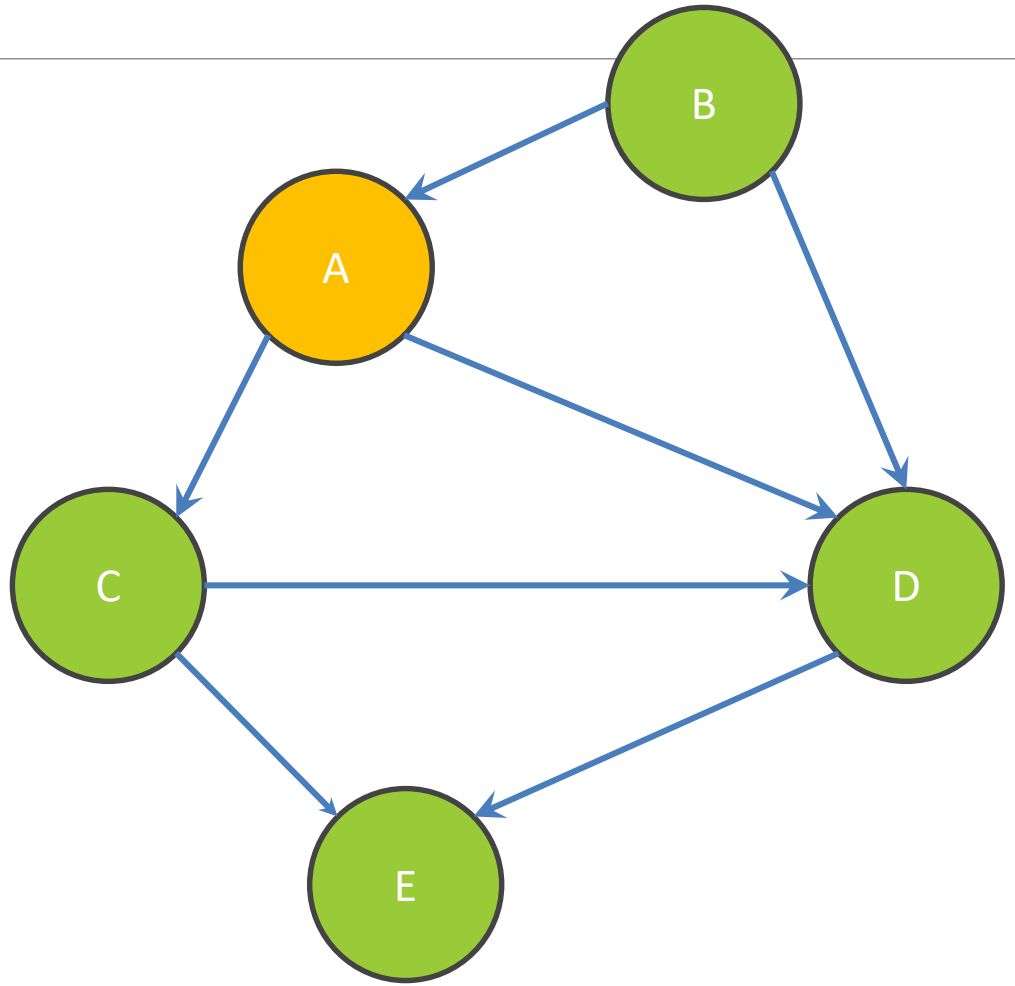
Q: <>



Breadth-First Search

Q: <>

Q: <A>

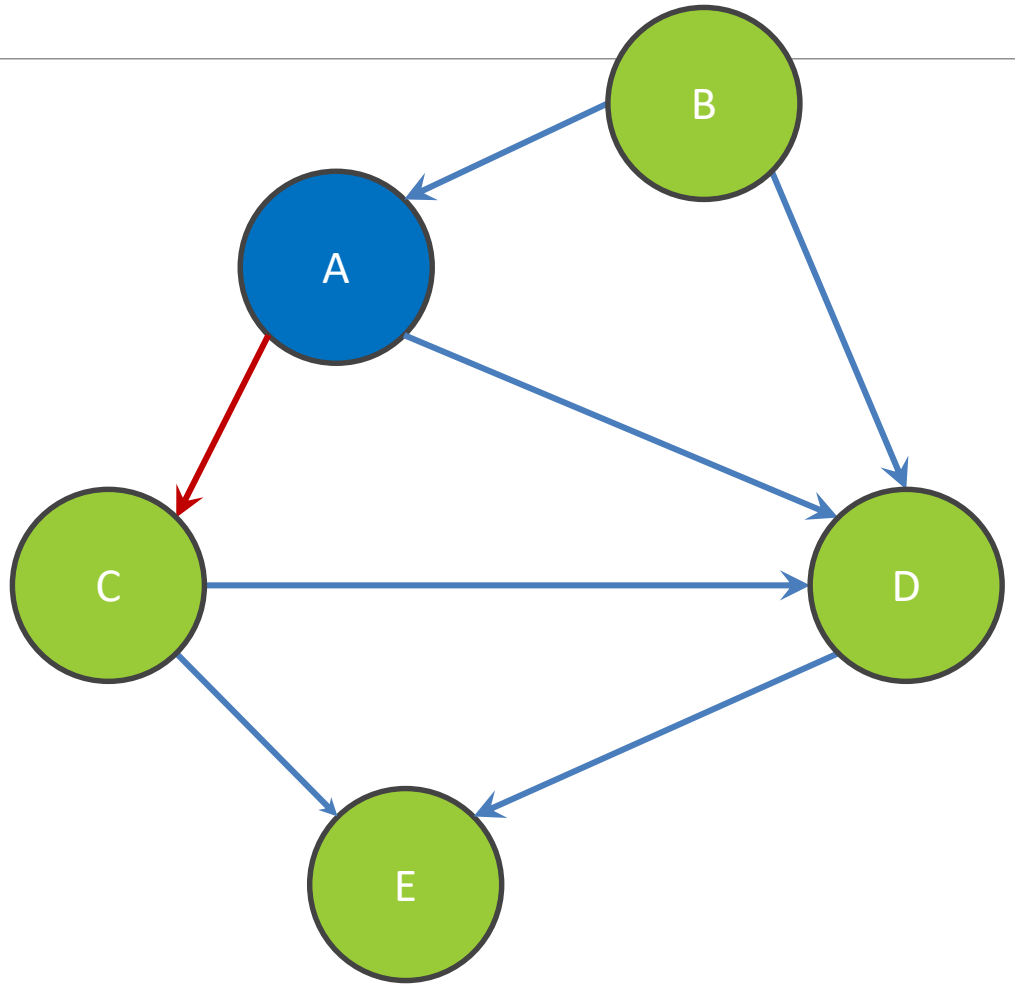


Breadth-First Search

Q: <>

Q: <A>

Q: <>



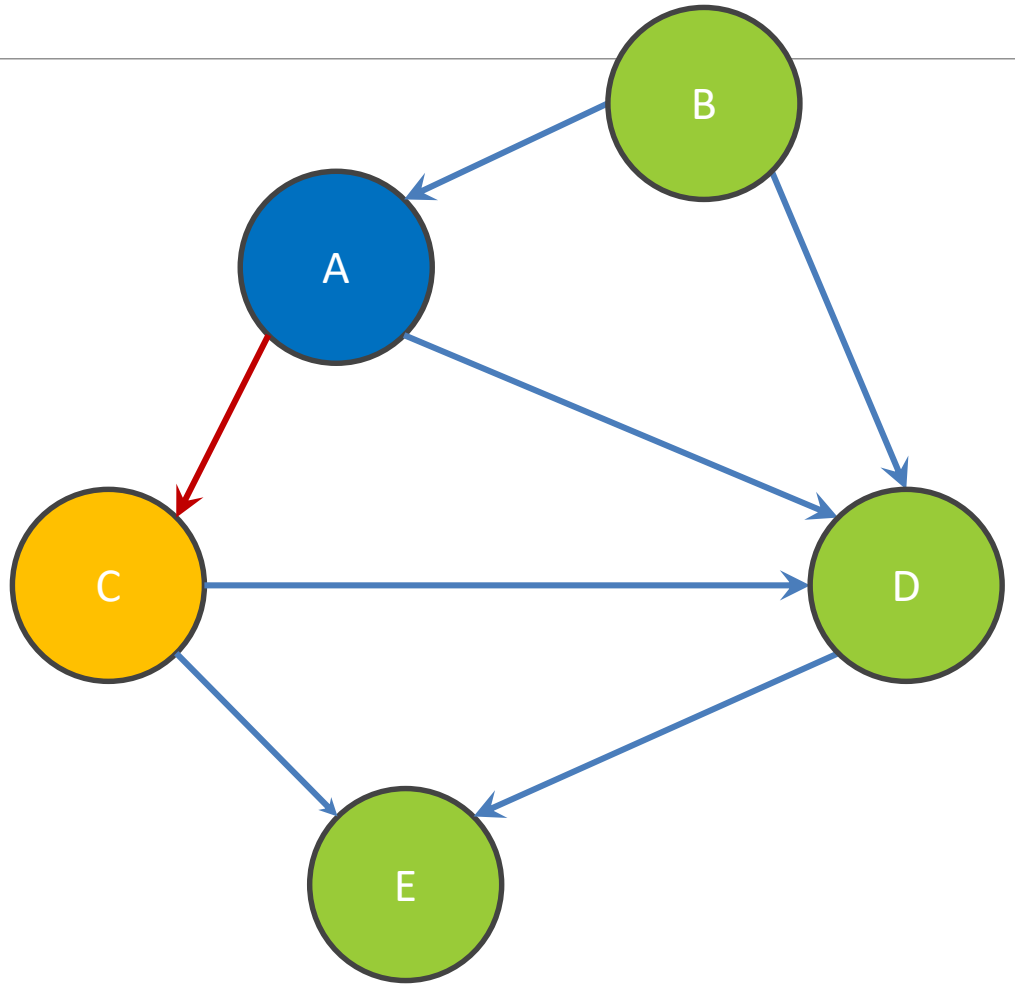
Breadth-First Search

Q: <>

Q: <A>

Q: <>

Q: <C>



Breadth-First Search

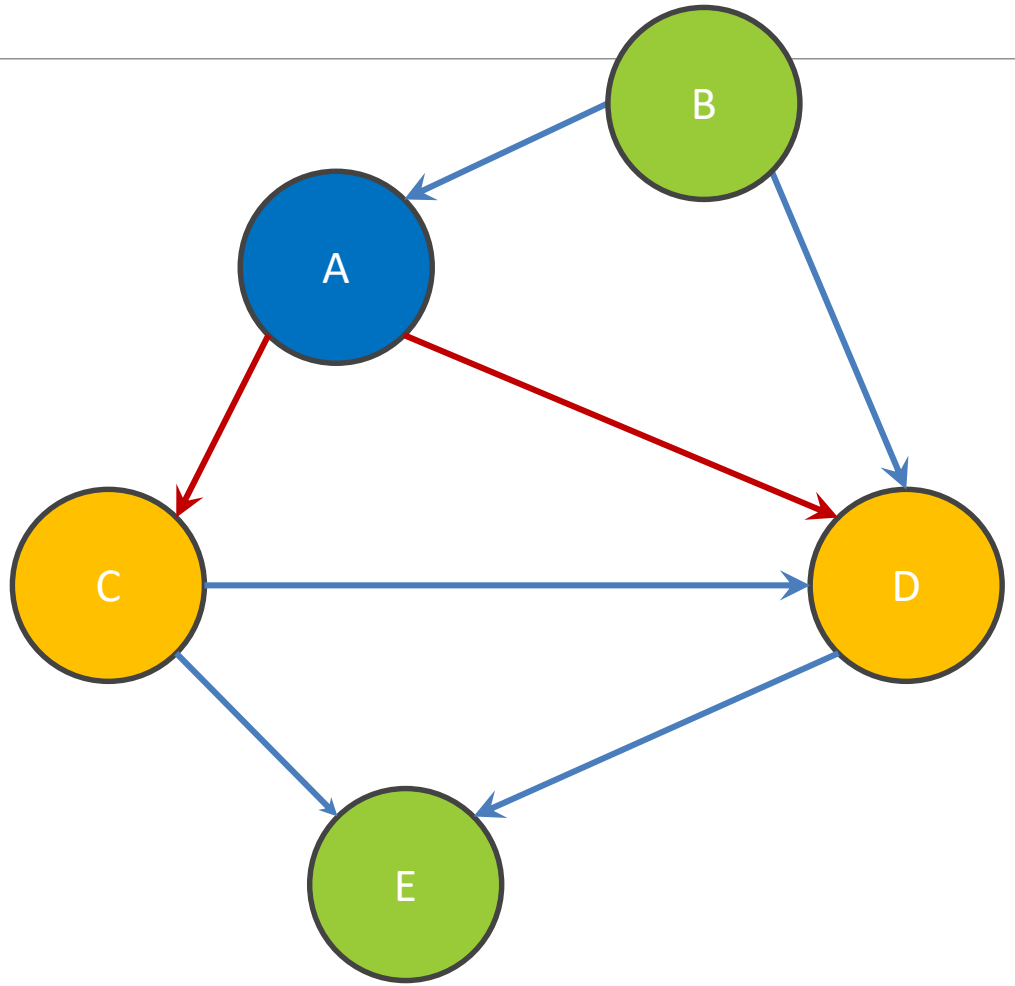
Q: <>

Q: <A>

Q: <>

Q: <C>

Q: <C ,D>



Breadth-First Search

Q: <>

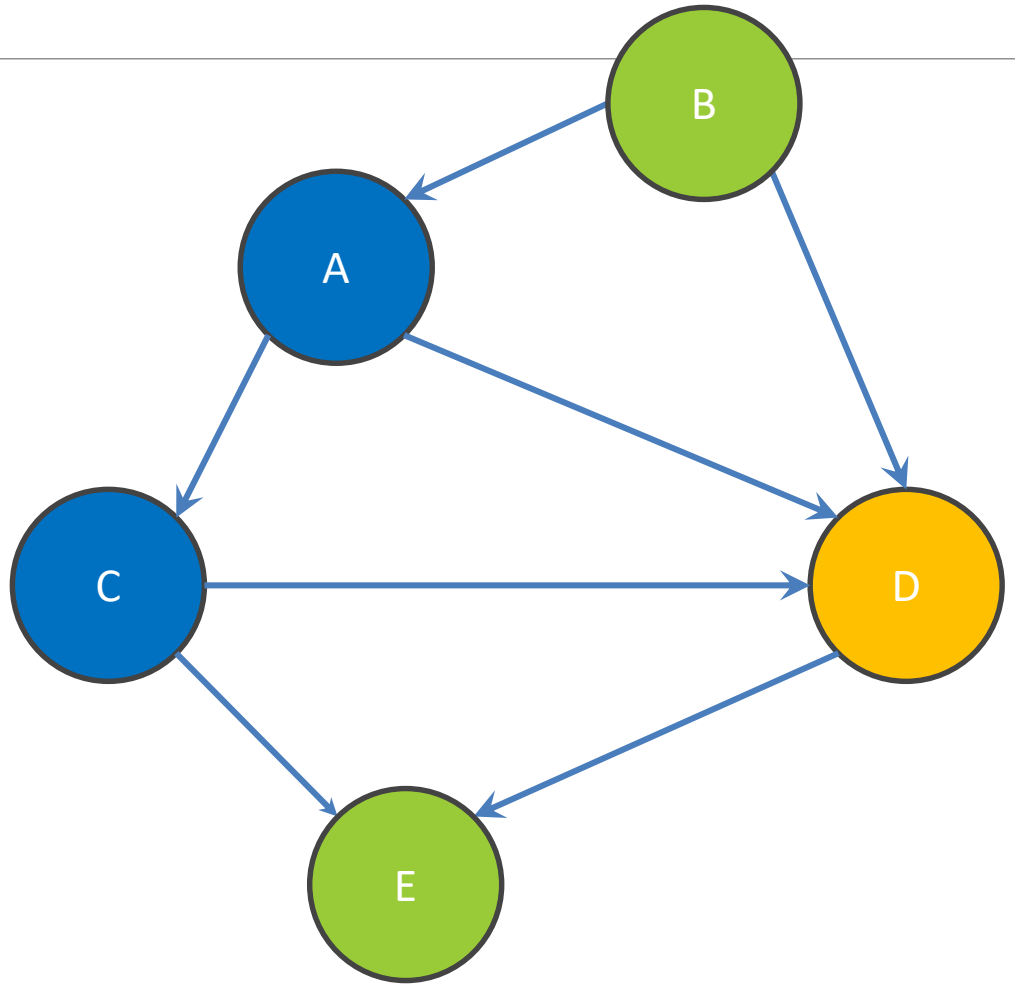
Q: <A>

Q: <>

Q: <C>

Q: <C ,D>

Q: <D>



Breadth-First Search

Q: <>

Q: <A>

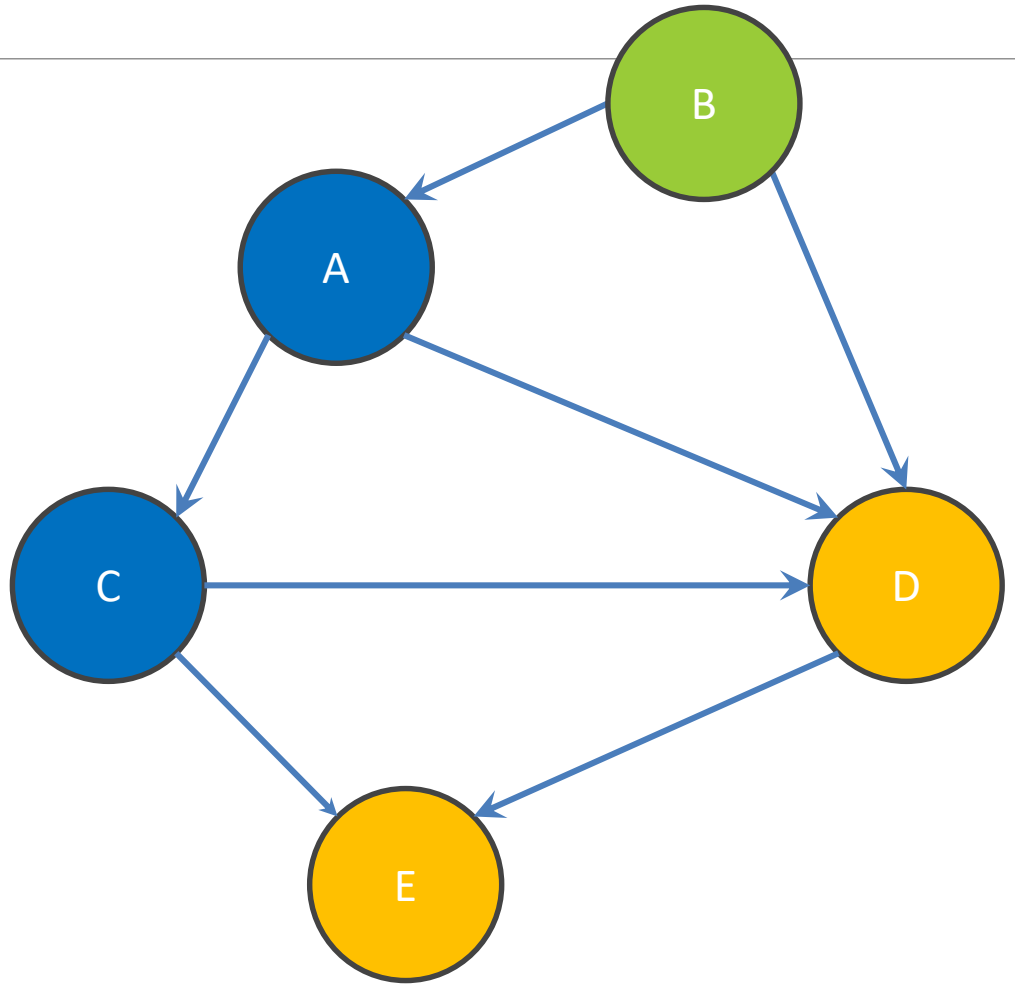
Q: <>

Q: <C>

Q: <C ,D>

Q: <D>

Q: <D, E>



Breadth-First Search

Q: <>

Q: <A>

Q: <>

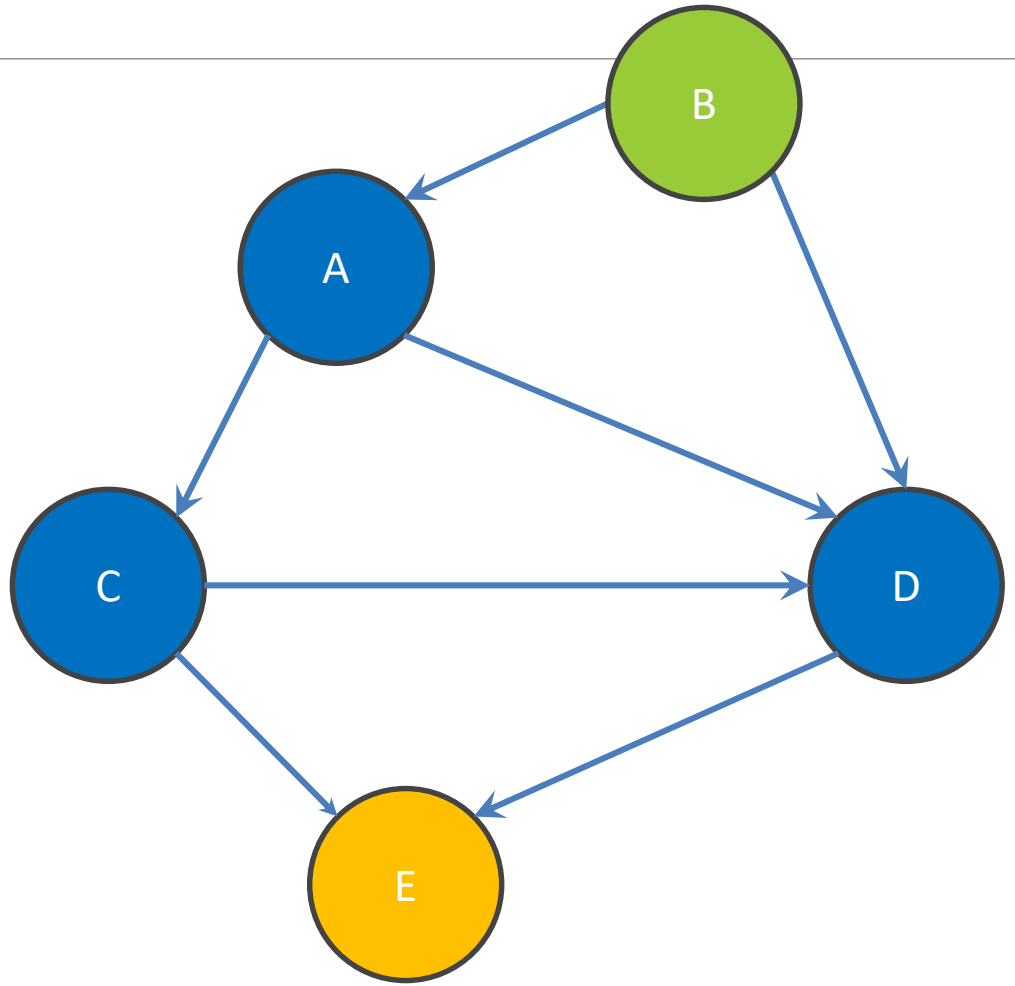
Q: <C>

Q: <C ,D>

Q: <D>

Q: <D, E>

Q: <E>



Breadth-First Search

Q: <>

Q: <A>

Q: <>

Q: <C>

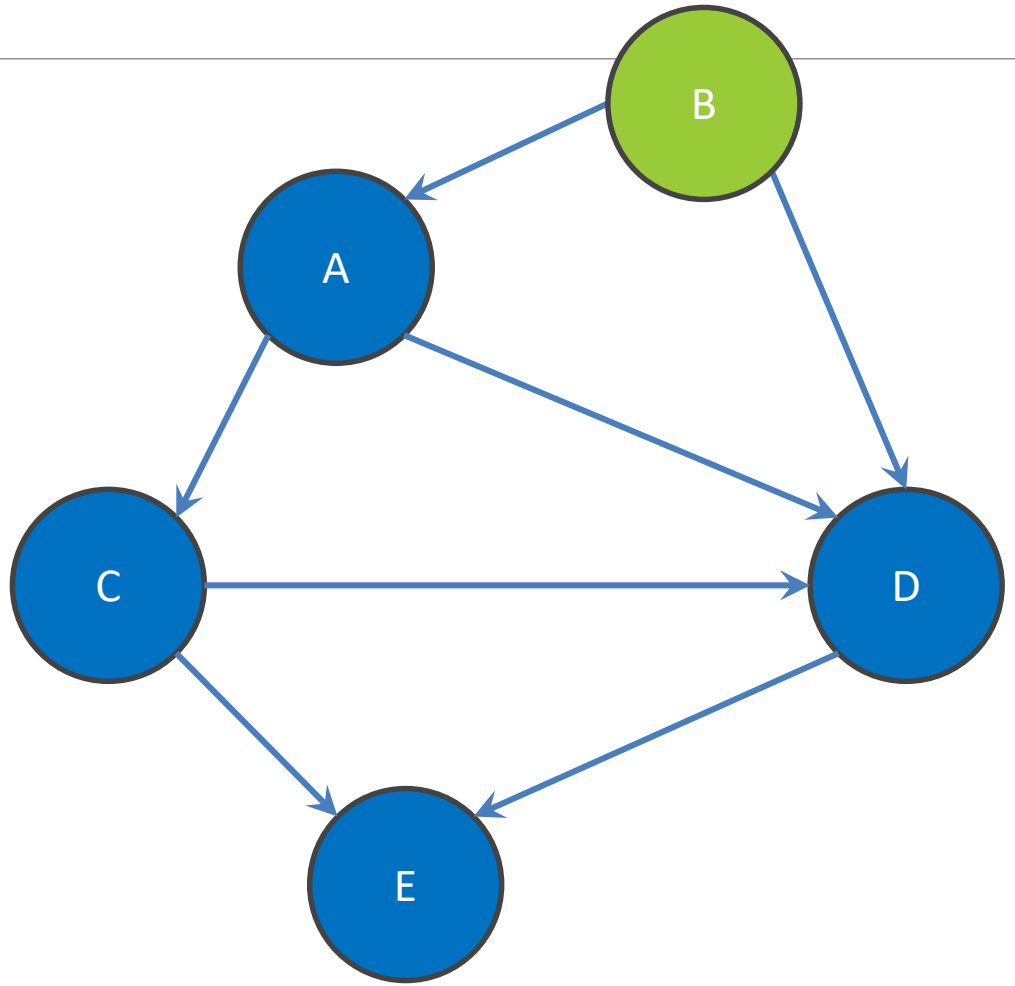
Q: <C ,D>

Q: <D>

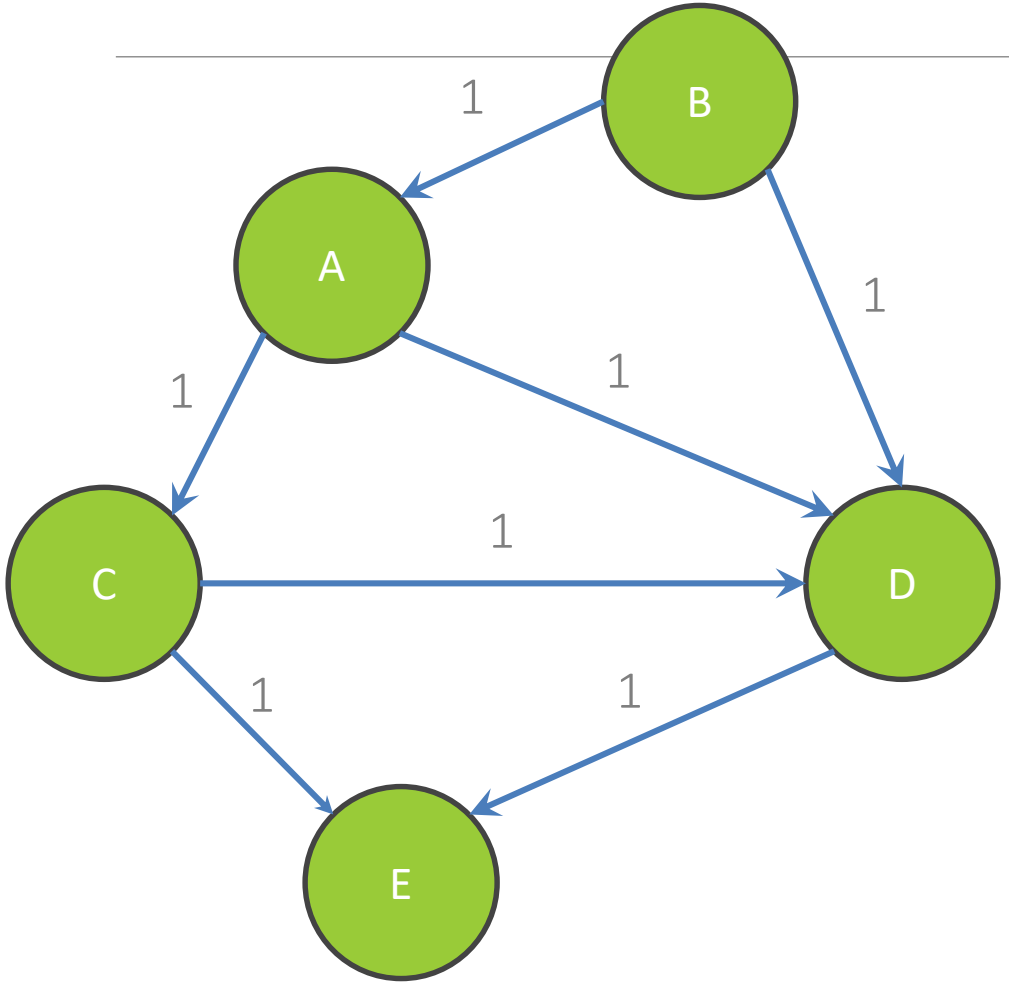
Q: <D, E>

Q: <E>

DONE



Shortest Paths with BFS

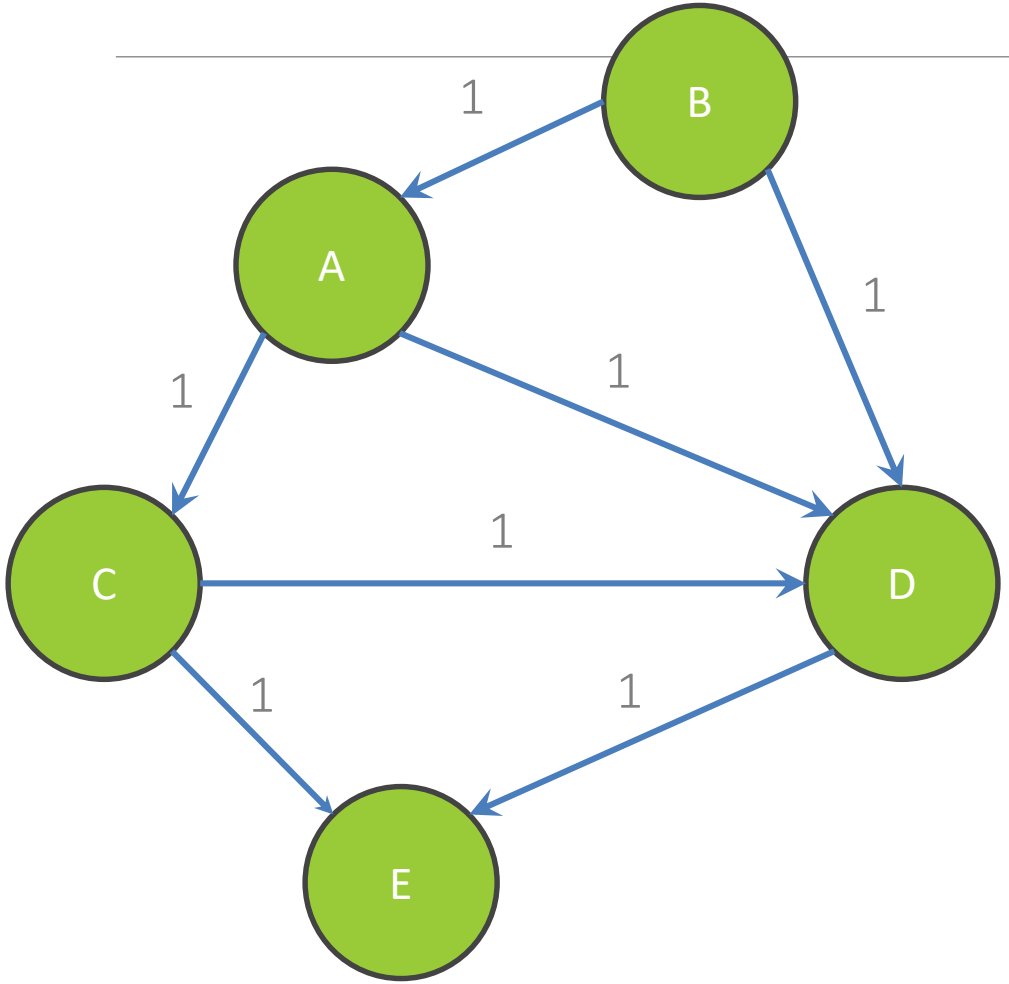


From Node B

Destination	Path	Cost
A	<B,A>	1
B	<B>	0
C	<B,A,C>	2
D		
E		

Shortest path to D? to E?
What are the costs?

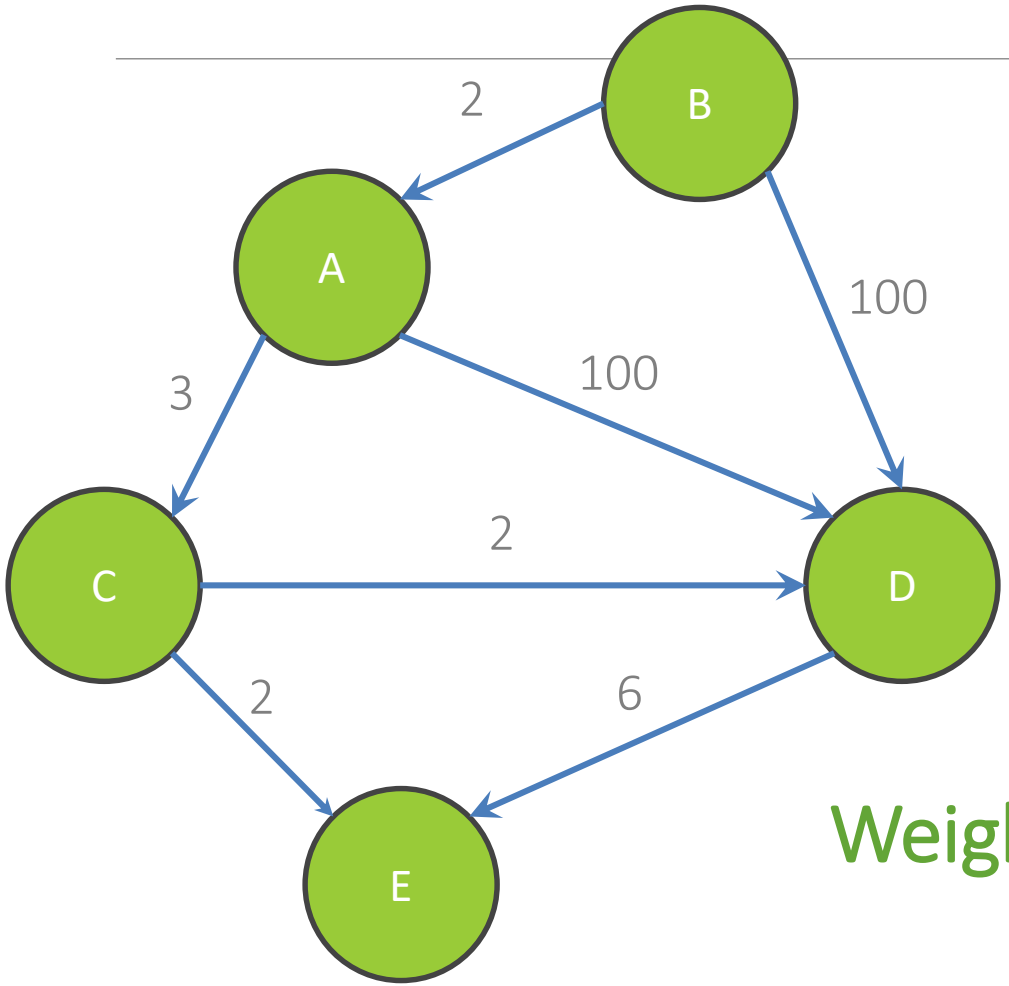
Shortest Paths with BFS



From Node B

Destination	Path	Cost
A	<B,A>	1
B	<B>	0
C	<B,A,C>	2
D	<B,D>	1
E	<B,D,E>	2

Shortest Paths with Weights

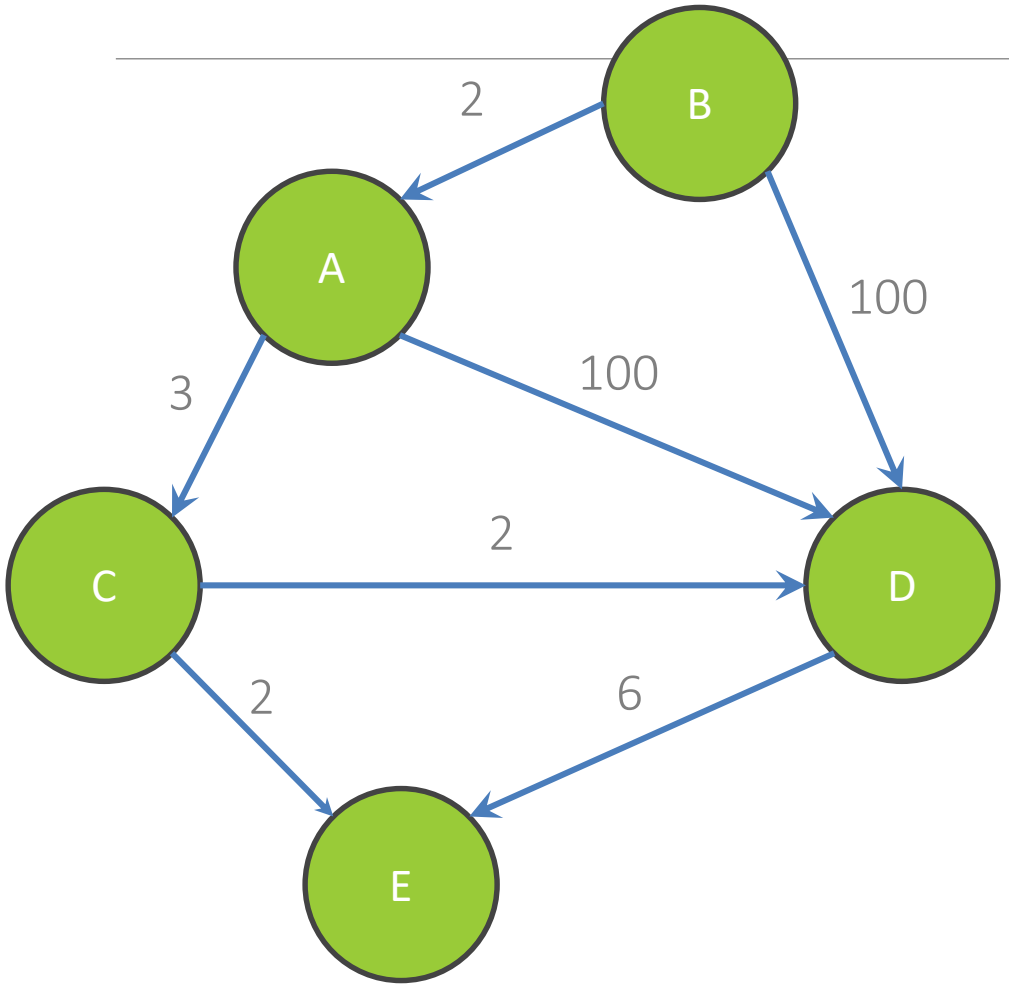


From Node B

Destination	Path	Cost
A	<B,A>	2
B	<B>	0
C	<B,A,C>	5
D		
E		

Weights are not the same!
Are the paths?

Shortest Paths with Weights



From Node B

Destination	Path	Cost
A	<B,A>	2
B	<B>	0
C	<B,A,C>	5
D	<B,A,C,D>	7
E	<B,A,C,E>	7

Interfaces

Classes, Interfaces, and Types

The fundamental unit of programming in Java is a class

Classes can extend other classes and implement interfaces

Interfaces can extend other interfaces

Classes, Objects, and Java

Everything is an instance of a class

- Defines data and methods

Every class extends exactly one other class

- Object if no explicit superclass
- Inherits superclass fields
- (You can make a ladder of “extends”)

Every class also defines a type

- Foo defines type Foo
- Foo inherits all inherited types

Interfaces

Pure type declaration

```
public interface Comparable {
        int compareTo(Object other);
}
```

Can contain:

- Method specifications (implicitly `public abstract`)
- Named constants (implicitly `public final static`)

Does not contain implementation!*

Cannot create instances of interfaces

*Java 8 does allow a form of “default” implementations in interfaces, but we will not use that, at least for now. So for us, for now, interfaces are pure specifications.

Implementing Interfaces

A class can implement one or more interfaces

```
class Kitten implements Pettable, Huggable
```

The implementing class and its instances have the interface type(s) as well as the class type(s)

The class must provide or inherit an implementation of all methods defined by the interface(s)

- Not true for abstract classes

Using Interface Types

An interface defines a type, so we can declare variables and parameters of that type

A variable with an interface type can refer to an object of any class implementing that type

```
List<String> x = new ArrayList<String>();
void sort(List aList) {…}
```

Guidelines for Interfaces

Provide interfaces for significant types and abstractions

Write code using interface types like Map instead of HashMap and TreeMap wherever possible

- Allows code to work with different implementations later on

Both interfaces and classes are appropriate in various circumstances

Demo

Parsing the Marvel data