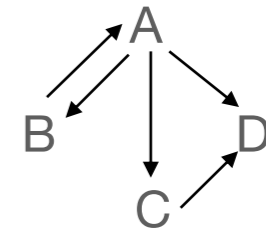# [320] Search Order and Queue Structures

Yiyin Shen

# Tracing DFS

```python
def find(self, dst):
    self.graph.visited.add(self)

    if self == dst:
        return (self.name,)

    for child in self.children:
        if not child in self.graph.visited:
            childpath = child.find(dst)
            if childpath:
                return (self.name,) + childpath

    return None
```
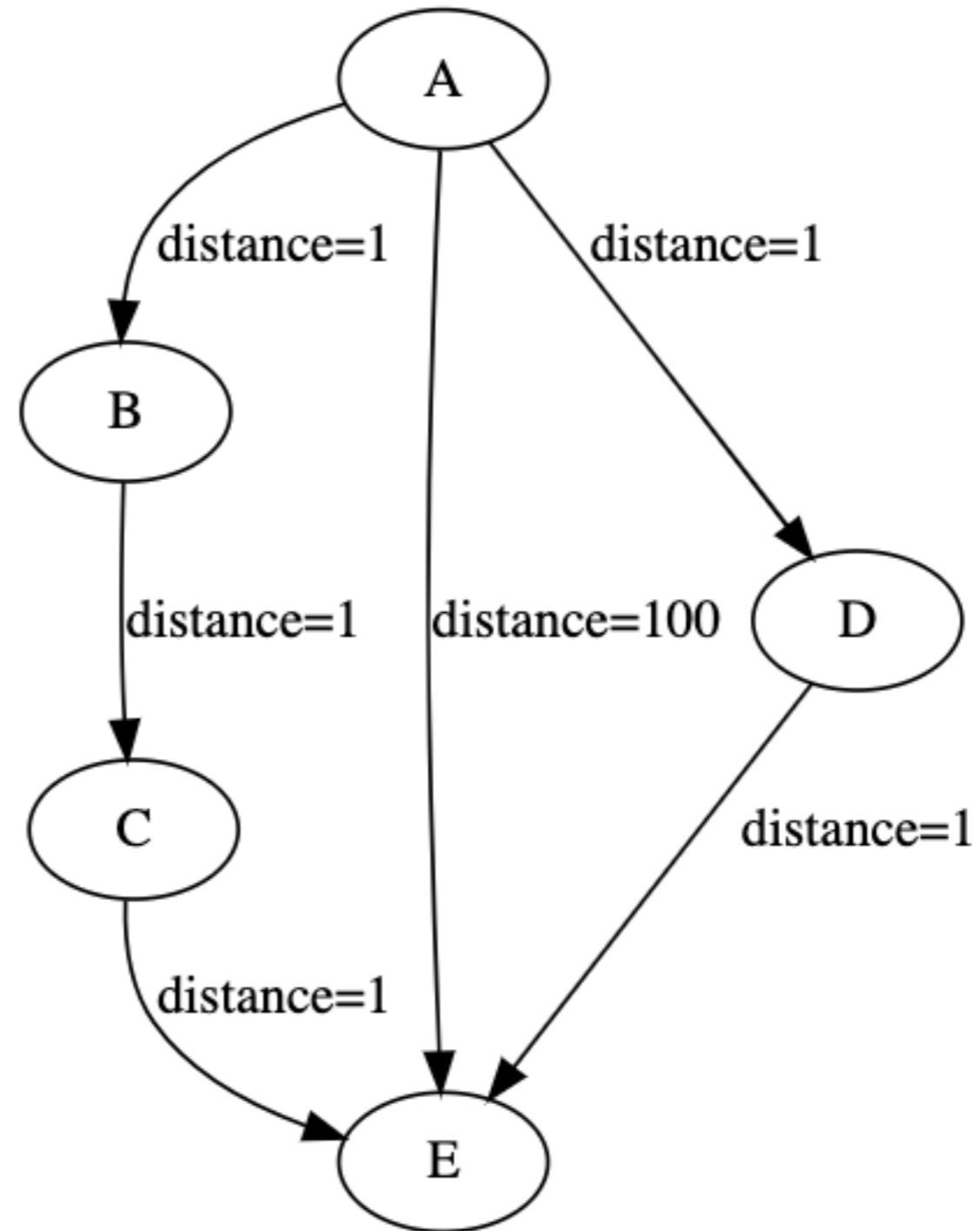
A.find(D)

# Queueing Structures

# Shortest Weighted Path



What path will DFS choose?

What path will BFS choose?
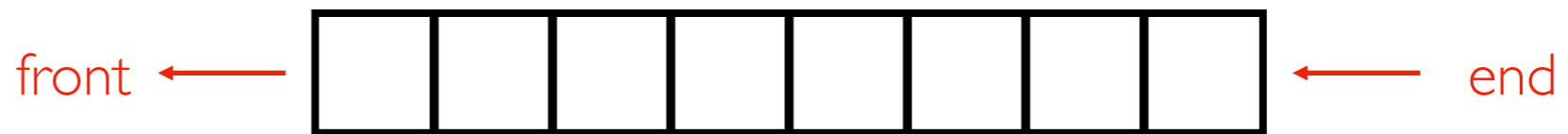
What path would you choose?

A

distance=1                    distance=1
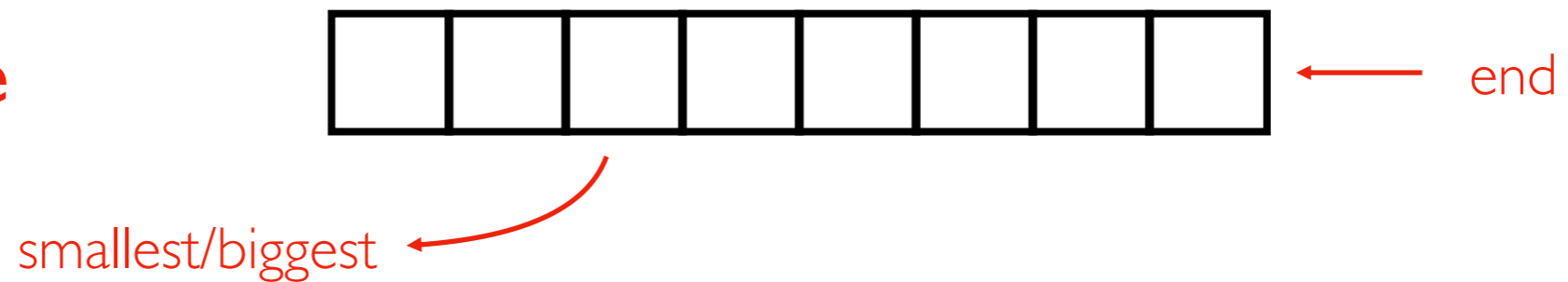
B

distance=1    distance=100    D

C                                      distance=1

distance=1

E

# Work queues ("to do" list)

**Stack**
(FILO)

end
end

**Queue**
(FIFO)

front

end

**Priority Queue**

end

smallest/biggest

# Work queues ("to do" list)

**Stack**
(FILO)

```
L.append(x)
```

← end
→ end

```
x = L.pop(-1)
```

**Queue**
(FIFO)

front ←

← end

```
L.append(x)
```

```
x = L.pop(0)
```

**Priority Queue**

← end

```
L.append(x)
```

smallest/biggest ←

```
L.sort()
x = L.pop(-1)
```

*what operations are slow?*

# Work queues ("to do" list)

**Stack**
(FILO)

```
L.append(x)
```

← end
→ end

```
x = L.pop(-1)
```

**Queue**
(FIFO)

front ←

← end

❗
```
x = L.pop(0)
```

```
L.append(x)
```

**Priority Queue**

← end

```
L.append(x)
```

smallest/biggest ←

❗
```
L.sort()
x = L.pop(-1)
```

*what operations are slow?*