[320] Graph Search

Department of Computer Sciences
University of Wisconsin-Madison

Tracing DFS

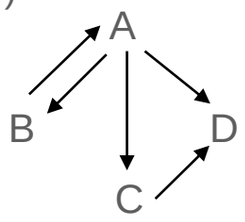
```
def find(self, dst):
    self.graph.visited.add(self)

    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)



Tracing DFS

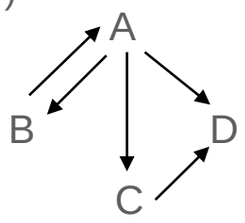
```
def find(self, dst):
    self.graph.visited.add(self)

    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)



A.find(D) =

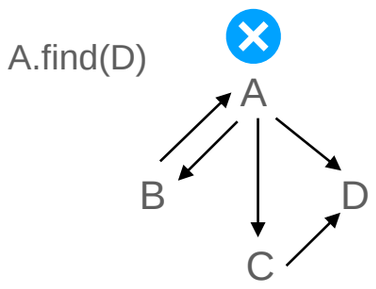
Tracing DFS

```
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) =

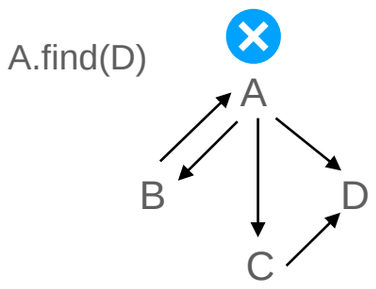
Tracing DFS

```
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) =

B.find(D) =

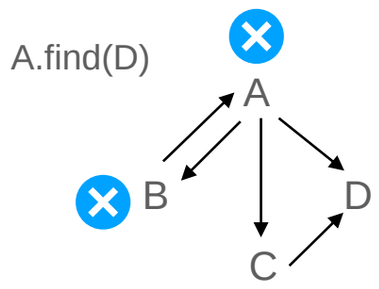
Tracing DFS

```
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) =

B.find(D) =

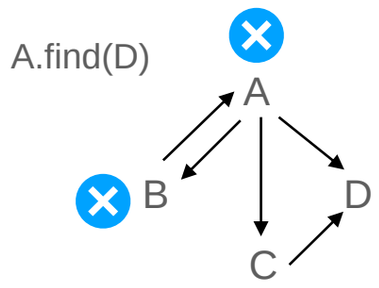
Tracing DFS

```
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) =
B.find(D) = None

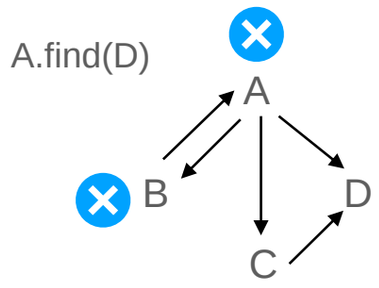
Tracing DFS

```
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) =
B.find(D) = None

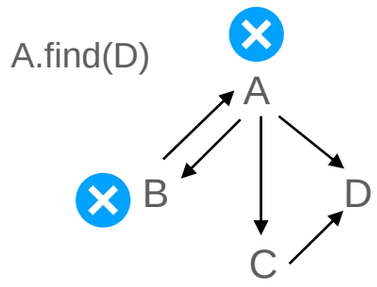
Tracing DFS

```
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) =
B.find(D) = None

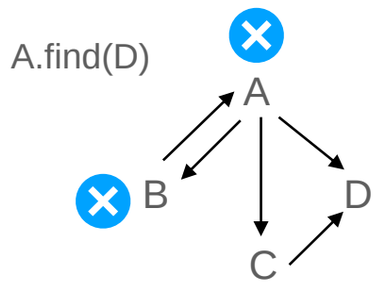
Tracing DFS

```
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) =

B.find(D) = None

C.find(D) =

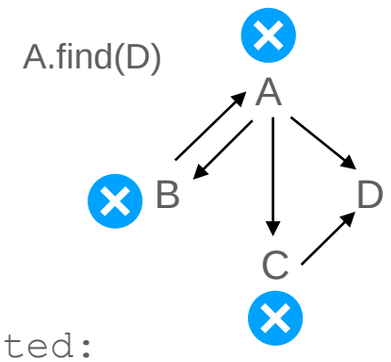
Tracing DFS

```
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) =

B.find(D) = None

C.find(D) =

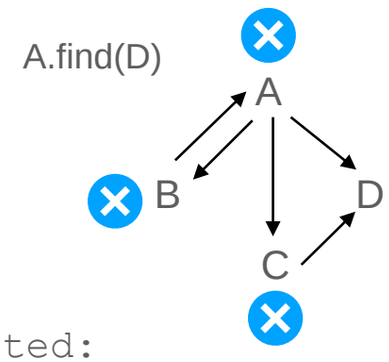
Tracing DFS

```
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) =
B.find(D) = None
C.find(D) =
D.find(D) =

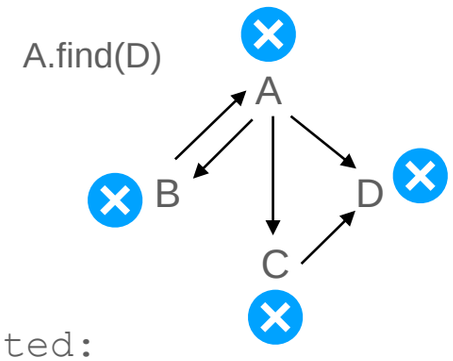
Tracing DFS

```
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) =

B.find(D) = None

C.find(D) =

D.find(D) =

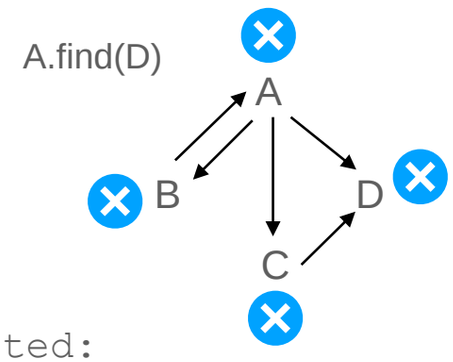
Tracing DFS

```
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) =

B.find(D) = None

C.find(D) =

D.find(D) = (D,)

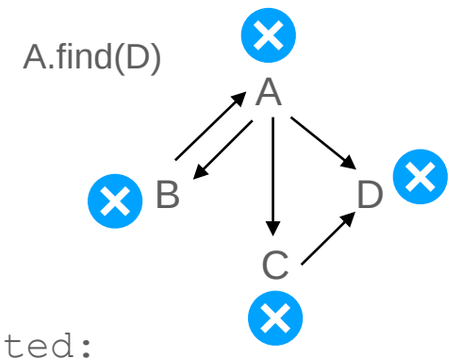
Tracing DFS

```
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) =

B.find(D) = None

C.find(D) = (C, D)

D.find(D) = (D,)

where (C, D) = (C,) + (D,)

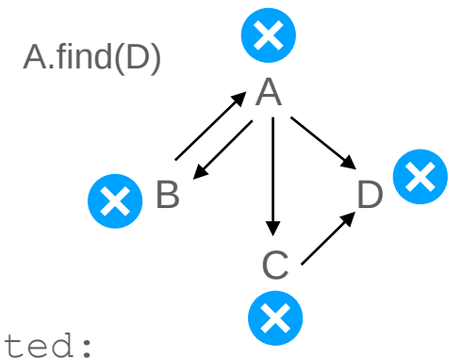
Tracing DFS

```
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) = (A, C, D)

B.find(D) = None

C.find(D) = (C, D)

D.find(D) = (D,)

where (A, C, D) = (A,) + (C, D)