A graph is a set of vertices $V$ and a set of edges $(u,v) \in E$ where $u,v \in V$. How can we print out all of the vertices in a tree?
Searching a tree

We could do any of the traversals we saw before: pre-order, in-order, post-order

A flash from the past

Stack:
- LIFO
- Add to the back
- Remove from the back

Queue:
- FIFO
- Add to the back
- Remove from the front

Searching a tree

treeBFS( start )
q = new Queue()
q.add(start)
treeSearch(q)

treeDFS( start )
s = new Stack()
s.add(start)
treeSearch(s)

treeSearch( toVisit )
while toVisit.empty()
v = toVisit.remove()
// visit v, e.g., print it out
for c in v.getChildren()
toVisit.add(c)

Tree BFS

treeBFS( start )
q = new Queue()
q.add(start)
treeSearch(q)

q:
Visited:

A

B
D
E

C
F
G
Tree BFS

treeBFS(start)
q = new Queue()
q.add(start)
treeSearch(q)

q: A
Visited:

9

toVisit-queue: A
printed:

10

What order will the nodes get printed out?
Assume children are traversed left to right.

11
toVisit-queue: A
printed:

12

13

14

15
Tree BFS

treeSearch( toVisit )
  while !toVisit.empty()
    v = toVisit.remove()  // visit v, e.g., print it out
    for c in v.getChildren()
      toVisit.add(c)

toVisit-queue: B D E
  printed: A

Tree BFS

treeSearch( toVisit )
  while !toVisit.empty()
    v = toVisit.remove()  // visit v, e.g., print it out
    for c in v.getChildren()
      toVisit.add(c)

toVisit-queue: B D E  
  printed: A
Tree BFS

treeSearch( toVisit )
while toVisit.empty()
  v = toVisit.remove() // visit v, e.g., print it out
  for c in v.getChildren() toVisit.add(c)

print: A B

Tree BFS

treeSearch( toVisit )
while toVisit.empty()
  v = toVisit.remove() // visit v, e.g., print it out
  for c in v.getChildren() toVisit.add(c)

print: A B

Tree BFS

treeSearch( toVisit )
while toVisit.empty()
  v = toVisit.remove() // visit v, e.g., print it out
  for c in v.getChildren() toVisit.add(c)

print: A B

Tree BFS

treeSearch( toVisit )
while toVisit.empty()
  v = toVisit.remove() // visit v, e.g., print it out
  for c in v.getChildren() toVisit.add(c)

print: A B
Tree BFS

```python
def treeSearch(toVisit):
    while not toVisit.empty():
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

`toVisit-queue: E C F`
`printed: A B D`

---

Tree BFS

```python
def treeSearch(toVisit):
    while not toVisit.empty():
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

`toVisit-queue: E C F`
`printed: A B D`

---

Tree BFS

```python
def treeSearch(toVisit):
    while not toVisit.empty():
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

`toVisit-queue: E C F`
`printed: A B D`

---

Tree BFS

```python
def treeSearch(toVisit):
    while not toVisit.empty():
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)
```

`toVisit-queue: C F`
`printed: A B D E`
treeSearch( toVisit )
    while toVisit.empty()
        v = toVisit.remove()
        // visit v, e.g., print it out
        for c in v.getChildren()
            toVisit.add(c)

How are we exploring the vertices?

orElse: all vertices a given number of edges from the start/root
Tree BFS = Tree breadth first search

treeSearch( toVisit )
while !toVisit.empty()
    v = toVisit.remove()
    // visit v, e.g., print it out
    for c in v.getChildren()
        toVisit.add(c)

Frontier: all vertices a given number of edges from the start/root

toVisit-queue: C F G
printed: A B D E

Tree BFS

treeSearch( toVisit )
while !toVisit.empty()
    v = toVisit.remove()
    // visit v, e.g., print it out
    for c in v.getChildren()
        toVisit.add(c)

toVisit-queue: C F G
printed: A B D E

Tree BFS

treeSearch( toVisit )
while !toVisit.empty()
    v = toVisit.remove()
    // visit v, e.g., print it out
    for c in v.getChildren()
        toVisit.add(c)

toVisit-queue: C F G
printed: A B D E C

Tree BFS

treeSearch( toVisit )
while !toVisit.empty()
    v = toVisit.remove()
    // visit v, e.g., print it out
    for c in v.getChildren()
        toVisit.add(c)

print: A B D E C F G
Tree DFS

\[
\text{treeDFS( start )}
\]
\[
\begin{align*}
 & s = \text{new Stack()} \\
 & s.\text{add( start )} \\
 & \text{treeSearch( s )}
\end{align*}
\]

s:
printed:

What order will the nodes get printed out?
Assume children are traversed left to right.

33

Tree DFS

\[
\text{treeSearch( toVisit )}
\]
\[
\begin{align*}
 & \text{while toVisit.\text{empty}()} \\
 & \quad v = \text{toVisit.\text{remove}()} \\
 & \quad // \text{visit v, e.g., print it out} \\
 & \quad \text{for c in v.\text{getChildren}()} \\
 & \quad \quad \text{toVisit.\text{add}(c)}
\end{align*}
\]

toVisit-stack: A
printed:

35

Tree DFS

\[
\text{treeDFS( start )}
\]
\[
\begin{align*}
 & s = \text{new Stack()} \\
 & s.\text{add( start )} \\
 & \text{treeSearch( s )}
\end{align*}
\]

s: A
printed:

34

Tree DFS

\[
\text{treeSearch( toVisit )}
\]
\[
\begin{align*}
 & \text{while toVisit.\text{empty}()} \\
 & \quad v = \text{toVisit.\text{remove}()} \\
 & \quad // \text{visit v, e.g., print it out} \\
 & \quad \text{for c in v.\text{getChildren}()} \\
 & \quad \quad \text{toVisit.\text{add}(c)}
\end{align*}
\]

toVisit-stack: A
printed:

36
Tree DFS

treeSearch(toVisit)
  while toVisit.empty()
    v = toVisit.remove()
    // visit v, e.g., print it out
    for c in v.getChildren()
      toVisit.add(c)

toVisit-stack: B D E
  printed: A

treeSearch(toVisit)
  while toVisit.empty()
    v = toVisit.remove()
    // visit v, e.g., print it out
    for c in v.getChildren()
      toVisit.add(c)

toVisit-stack: B D E
  printed: A
Tree DFS

treeSearch( toVisit )
while toVisit.empty()
  v = toVisit.remove()
  // visit v, e.g., print it out
  for c in v.getChildren()
    toVisit.add(c)

toVisit-stack: B D
printed: A E

41

Tree DFS

treeSearch( toVisit )
while toVisit.empty()
  v = toVisit.remove()
  // visit v, e.g., print it out
  for c in v.getChildren()
    toVisit.add(c)

toVisit-stack: B D
printed: A E

42

Tree DFS

treeSearch( toVisit )
while toVisit.empty()
  v = toVisit.remove()
  // visit v, e.g., print it out
  for c in v.getChildren()
    toVisit.add(c)

toVisit-stack: B D G
printed: A E

43

Tree DFS

treeSearch( toVisit )
while toVisit.empty()
  v = toVisit.remove()
  // visit v, e.g., print it out
  for c in v.getChildren()
    toVisit.add(c)

toVisit-stack: B D G
printed: A E

44
Tree DFS

treeSearch( toVisit )
while toVisit.empty()
    v = toVisit.remove()
    // visit v, e.g., print it out
    for c in v.getChildren()
        toVisit.add(c)

toVisit-stack: B D
printed: A E G

treeSearch( toVisit )
while toVisit.empty()
    v = toVisit.remove()
    // visit v, e.g., print it out
    for c in v.getChildren()
        toVisit.add(c)

toVisit-stack: B D
printed: A E G

Tree DFS

treeSearch( toVisit )
while toVisit.empty()
    v = toVisit.remove()
    // visit v, e.g., print it out
    for c in v.getChildren()
        toVisit.add(c)

toVisit-stack: B D
printed: A E G

Tree DFS

treeSearch( toVisit )
while toVisit.empty()
    v = toVisit.remove()
    // visit v, e.g., print it out
    for c in v.getChildren()
        toVisit.add(c)

toVisit-stack: B
printed: A E G D
Tree DFS

```
treeSearch(toVisit)
  while !toVisit.empty()
    v = toVisit.remove()
    // visit v, e.g., print it out
    for c in v.getChildren()
      toVisit.add(c)
```

```
toVisit-stack: B 
printed: A E G D
```

How are we exploring the vertices?

```
treeSearch(toVisit)
  while !toVisit.empty()
    v = toVisit.remove()
    // visit v, e.g., print it out
    for c in v.getChildren()
      toVisit.add(c)
```

```
toVisit-stack: B 
printed: A E G D
```

50

Frontier: go as far down one branch as possible, working right to left

```
treeSearch(toVisit)
  while !toVisit.empty()
    v = toVisit.remove()
    // visit v, e.g., print it out
    for c in v.getChildren()
      toVisit.add(c)
```

```
toVisit-stack: 
printed: A E G D B
```

52
Tree DFS

treeSearch( toVisit )
while toVisit.empty()
    v = toVisit.remove()
    // visit v, e.g., print it out
    for c in v.getChildren()
        toVisit.add(c)

toVisit-stack: C F
printed: A E G D B

Run-time of graph algorithms

A graph is a set of vertices $V$ and a set of edges $(u,v) \in E$ where $u,v \in V$

When we analyze graph algorithms, the run-time often includes both the number of vertices AND the number of edges:
- $|V|$ = number of vertices
- $|E|$ = number of edges

Sometimes, in big-O notation, we’ll use just $V$ and $E$ to represent these to simplify notation
What is the big-O run-time of `treeSearch`?
Assume all of the stack/queue operations are constant.
How many times do we visit each vertex?
How many times do we traverse each edge (via the for loop)?

What algorithm is this?
What order will the vertices get printed out?
Assume children are traversed left to right.
What algorithm is this?

```python
def search(v):
    # visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```

Visited: A B

What algorithm is this?

```python
def search(v):
    # visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```

Visited: A B C

Now where?

What algorithm is this?

```python
def search(v):
    # visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```

Visited: A B C

What algorithm is this?

```python
def search(v):
    # visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```

Visited: A B C F
What algorithm is this?

```
search(v)
    // visit v, e.g., print it out
    for c in v.getChildren()
        search(c)
```

Visited: A B C F D

Depth first search!

Any difference between this version and the stack version?

Visited: A B C F D
What algorithm is this?

```python
search(v)
// visit v, e.g., print it out
for c in v.getChildren()
    search(c)
```

Visited: A B C F D

Any difference between this version and the stack version?

Traverses in the other direction (left to right in this case).

treeDFS used a stack.
Is there a stack here?

The run-time stack keeping track of recursive calls!
What algorithm is this?

```java
search( v )
// visit v, e.g., print it out
for c in v.getChildren()
    search(c)
```

Visited: A B C F D E G

DFS versions

```java
treeDFS( start )
s = new Stack()
s.add(start)
treeSearch(s)
```

```java
treeRecursiveDFS( v )
// visit v, e.g., print it out
for c in v.getChildren()
    treeRecursiveDFS(c)
```

treeSearch on graphs

```java
treeSearch( toVisit )
while !toVisit.empty()
    v = toVisit.remove()
    // visit v, e.g., print it out
    for c in v.getChildren()
        toVisit.add(c)
```

What would happen if we ran `treeSearch` on this graph?

Won't end!

How can we fix this?

Keep track of the vertices that we've visited.
Searching on graphs

**BFS**

```
graphBFS(start)
q = new Queue()
q.add(start)
treeSearch(q)
```

```
toVisit-queue: A
visited:
```

**DFS**

```
graphDFS(start)
s = new Stack()
s.add(start)
treeSearch(s)
```

```
toVisit-queue: A
visited: What order will the nodes get printed out? Assume edges are traversed alphabetically.
```
```python
graphSearch(toVisit)
while !toVisit.empty():
    v = toVisit.remove()
    if !visited[v]:
        visited[v] = true
        for c in v.getAdjacent():
            if !visited[c]:
                toVisit.add(c)
```

```python
graphSearch(toVisit)
while !toVisit.empty():
    v = toVisit.remove()
    if !visited[v]:
        visited[v] = true
        for c in v.getAdjacent():
            if !visited[c]:
                toVisit.add(c)
```

toVisit-queue: A
visited:

BFS

BFS

BFS

BFS

81

82

83

84
graphSearch(toVisit)
while !toVisit.empty()
    v = toVisit.remove()
    if !visited[v]
        visited[v] = true
        for c in v.getAdjacent()
            if !visited[c]
                toVisit.add(c)

Notice that we do add E again to toVisit since we haven’t visited it yet.
BFS

`graphSearch(toVisit)
while !toVisit.empty()
    v = toVisit.remove()
    visited[v] = true
    for c in v.getAdjacent()
        if !visited[c]
            toVisit.add(c)`

toVisit-queue: D E C E F

visited: A B

We add E again to toVisit since we still haven't visited it yet.
BFS

graphSearch(toVisit)
while toVisit.empty()
    v = toVisit.remove()
    if visited[v]
        visited[v] = true
        for c in v.getAdjacent()
            if visited[c]
                toVisit.add(c)

visited: A B D

toVisit-queue: E C E F E

BFS

graphSearch(toVisit)
while toVisit.empty()
    v = toVisit.remove()
    if visited[v]
        visited[v] = true
        for c in v.getAdjacent()
            if visited[c]
                toVisit.add(c)

toVisit-queue: E C E F E
visited: A B D E

No adjacent vertices that haven't been visited
BFS

graphSearch(toVisit)
while !toVisit.empty()
    v = toVisit.remove()
    visited[v] = true
    for c in v.getAdjacent()
        if !visited[c]
            toVisit.add(c)

toVisit-queue: E C E F E
visited: A B D E

Frontier: all vertices a given number of edges from the start/root

E has already been visited
BFS

graphSearch( toVisit )
while !toVisit.empty()
    v = toVisit.remove()
    if !visited[v]
        visited[v] = true
        for c in v.getAdjacent()
            if !visited[c]
                toVisit.add(c)

toVisit-queue: E F E
visited: A B D E C

E has already been visited

BFS

graphSearch( toVisit )
while !toVisit.empty()
    v = toVisit.remove()
    if visited[v]
        visited[v] = true
        for c in v.getAdjacent()
            if !visited[c]
                toVisit.add(c)

toVisit-queue: E G
visited: A B D E C F
BFS

toVisit-queue:
visited: A B D E C F G