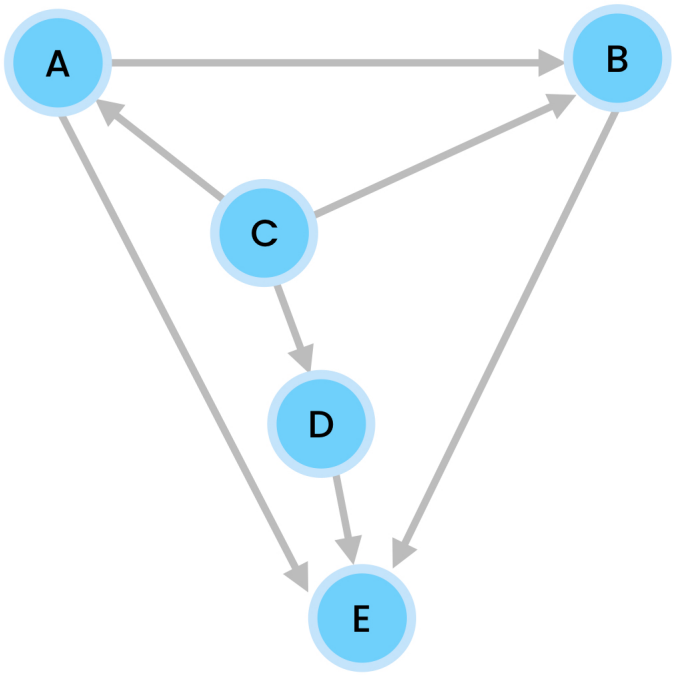
Traversal Algorithms

Given the following graph, what would be the output of breadth-first and depth-first searches starting from C?

At each step, when deciding where to go next, pick the node based on alphabetical order. For example, if you can choose between A, B and D as the next step, go to A.

Solution: next page



Depth-first Traversal

Answer: [C, A, B, E, D]

We start at the C.

Output: [C]

C has three neighbours: A, B and D. Since we've assumed that these nodes are inserted in alphabetical order, we should go to A next.

Output: [C, A]

A has two neighbours: B and E. Based on our assumption, we should go to B next.

Output: [C, A, B]

B has only one neighbour, so we go to E next.

Output: [C, A, B, E]

We can't go any deeper since E has no neighbours, so we go back to B.

We've visited all the neighbours of B, so we go back to A.

A has one more neighbour (E) but we've visited that node, so we go back to C.

Once again, C has three neighbours: A, B and D. We've visited the first two, so now we should go to D.

Output: [C, A, B, E, D]

Breadth-first Traversal

Answer: [C, A, B, D, E]

We start by adding C in a queue.

Queue: [C]

We remove C from the queue and visit it.

Output: [C]

Queue: []

Now we should add its unvisited neighbours to the queue (in alphabetical order).

Output: [C]

Queue: [A, B, D]

We remove A from the queue and visit it.

Output: [C, A]

Queue: [B, D]

Then we add the unvisited neighbours of A to the queue.

Output: [C, A]

Queue: [B, D, B, E]

We remove B from the queue and visit it.

Output: [C, A, B]

Queue: [D, B, E]

Again, we add the unvisited neighbours of B to the queue.

Output: [C, A, B]

Queue: [D, B, E, E]

Now, we remove D from the queue and visit it.

Output: [C, A, B, D]

Queue: [B, E, E]

We add the unvisited neighbours of D to the queue.

Output: [C, A, B, D]

Queue: [B, E, E, E]

We remove B from the queue but we've already visited it.

Output: [C, A, B, D]

Queue: [E, E, E]

We remove E from the queue and visit it.

Output: [C, A, B, D, E]

Queue: [E, E]

E doesn't have any neighbours, so we continue polling the queue.

We remove E from the queue but we've already visited it.

Output: [C, A, B, D, E]

Queue: [E]

We repeat.

Output: [C, A, B, D, E]

Queue: []

The queue is empty and we're done.