Depth-First Search (DFS) and Breadth-First Search (BFS)

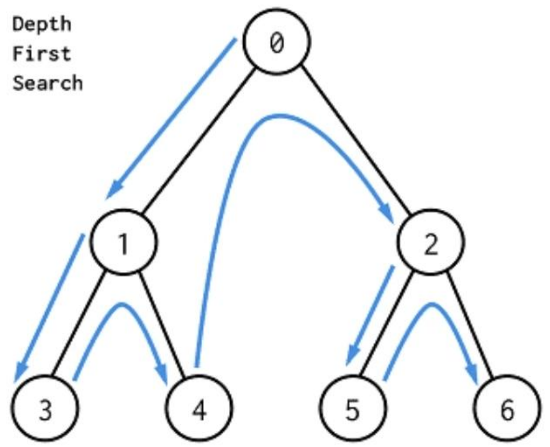
CPE2303 Design and Analysis of Algorithms

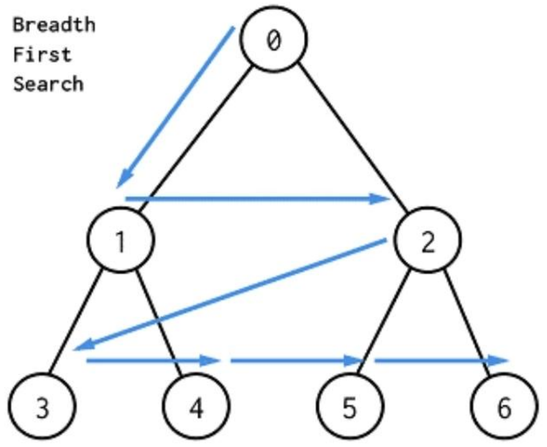
Dr.Pongrapee Kaewsaiha

Computer Engineering, EIT SSRU

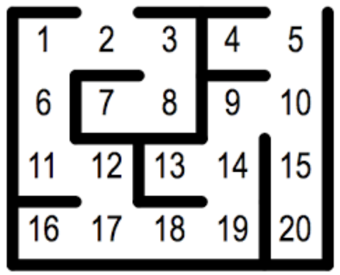
Depth-First Search & Breadth-First Search

Depth-first search (DFS) and breadth-first search (BFS) systematically process all vertices and edges of a graph. These algorithms are very useful for many applications involving graphs in artificial intelligence and operations research. In addition, they are indispensable for efficient investigation of fundamental properties of graphs such as connectivity and cycle presence.

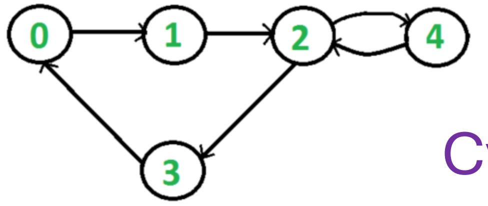




Source: dev.to



Maze solving

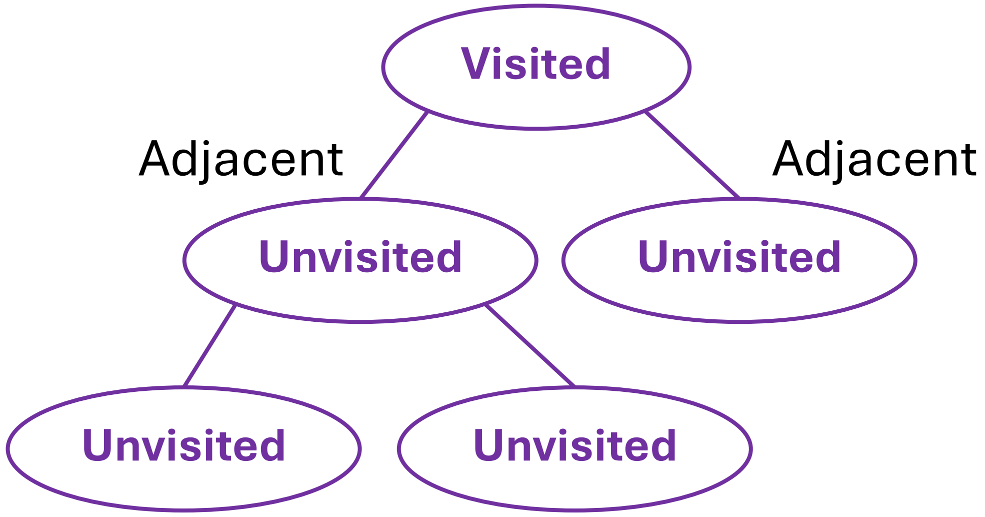


Cycle detection

Depth-First Search (DFS)

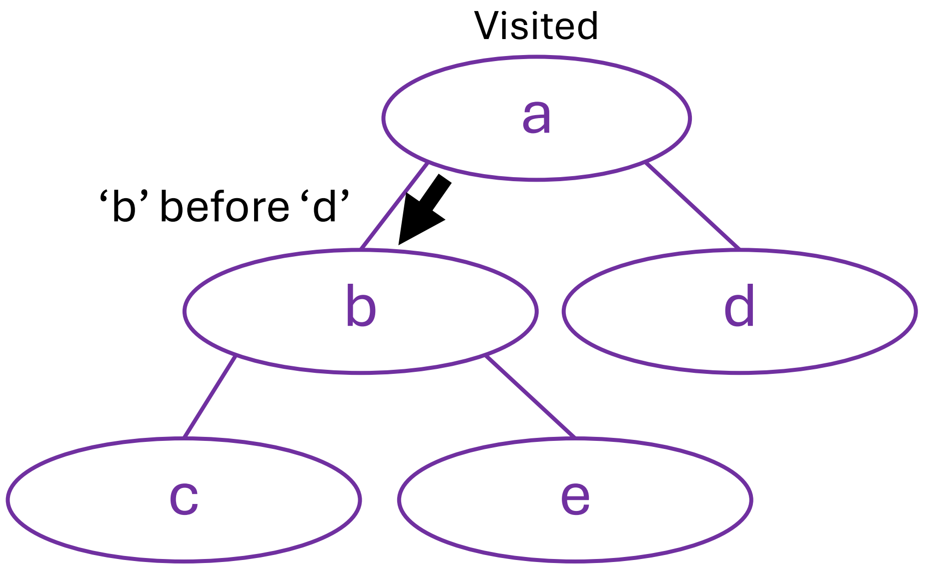
Depth-first search starts a graph's traversal at an arbitrary vertex by marking it as **visited**.

On each iteration, the algorithm proceeds to an **unvisited** vertex that is **adjacent** to the one it is currently in.



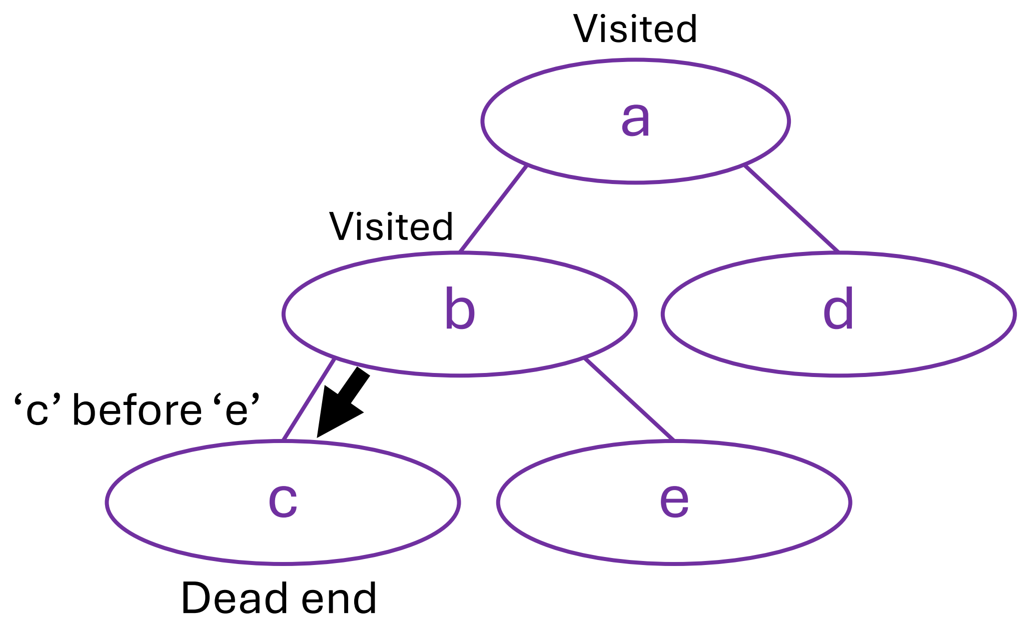
Depth-First Search (DFS)

If there are multiple adjacent vertices the next vertex is chosen based on the data structure representing the graph *(e.g., by the alphabetical order of the vertices)*.



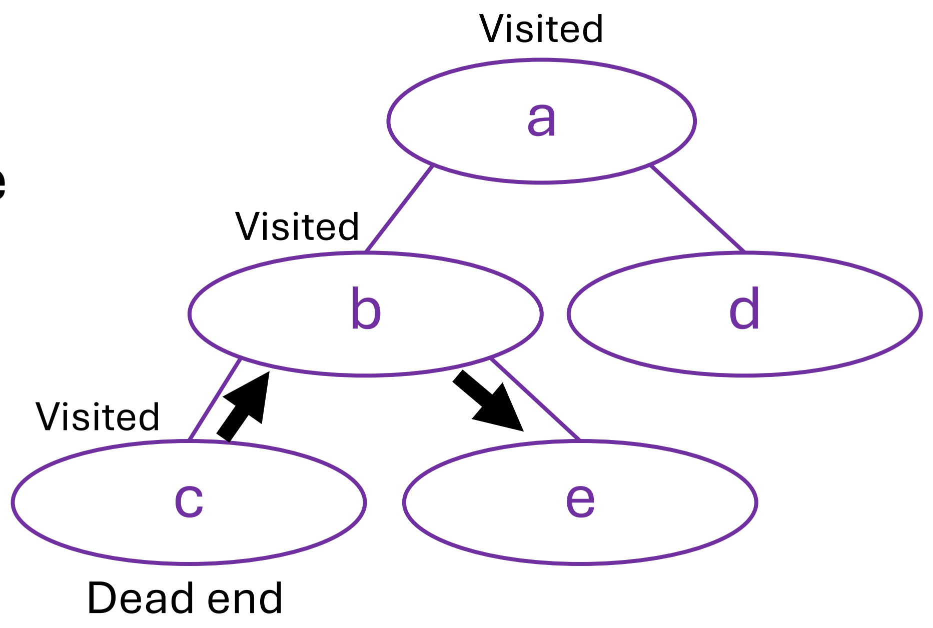
Depth-First Search (DFS)

This process continues until a **dead end** or when the current vertex has no adjacent unvisited vertices.



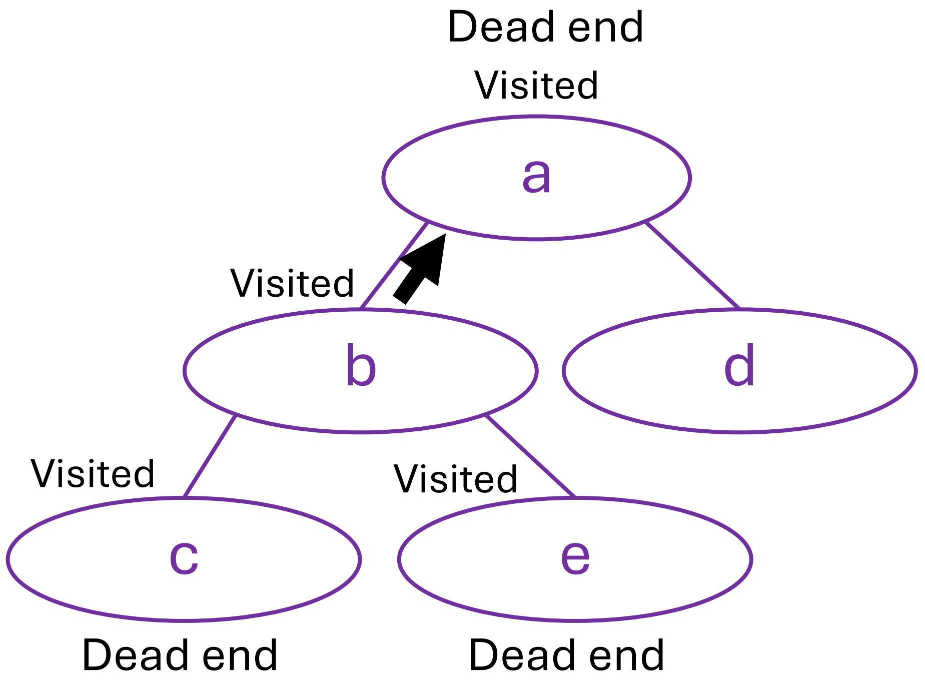
Depth-First Search (DFS)

At a dead end, the algorithm backs up one edge to the vertex it came from and tries to continue visiting unvisited vertices from there.



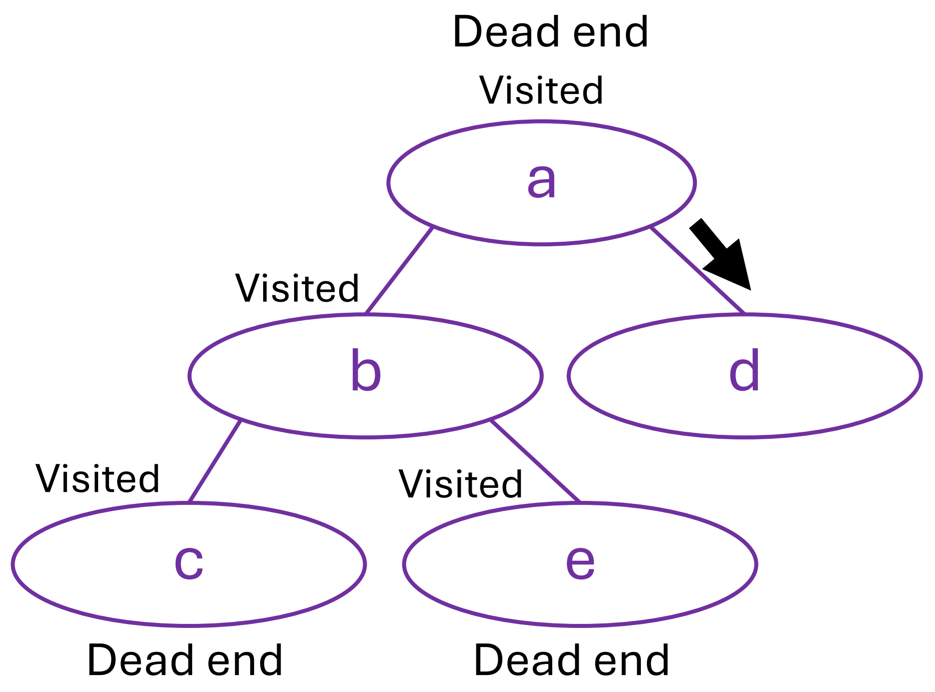
Depth-First Search (DFS)

The algorithm eventually halts after backing up to the starting vertex, with the latter being a dead end.



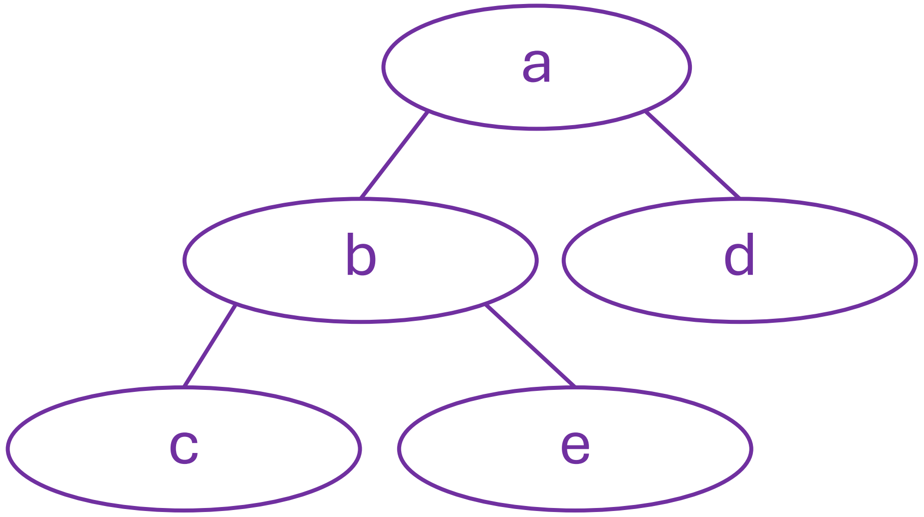
Depth-First Search (DFS)

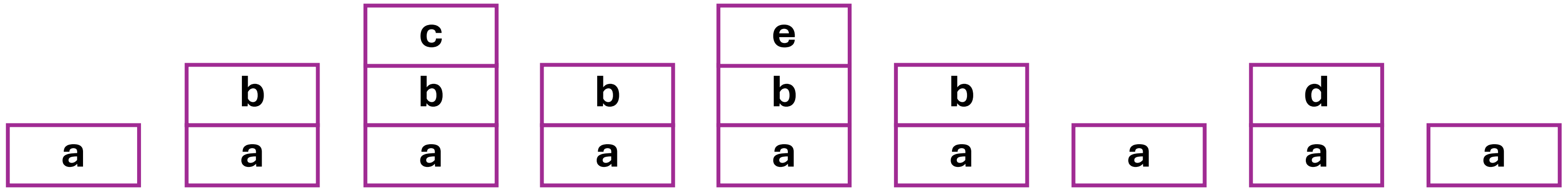
If unvisited vertices still remain, the depth-first search must be restarted at any one of them.



Depth-First Search (DFS)

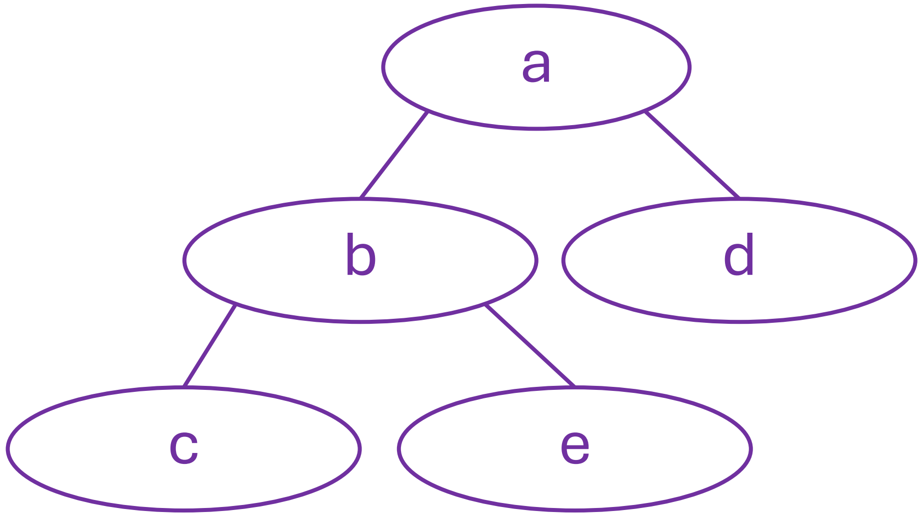
It is convenient to use a **stack** to trace the operation of depth-first search. We push a vertex into the stack when the vertex is reached for the first time, and we pop a vertex off the stack when it becomes a dead end.

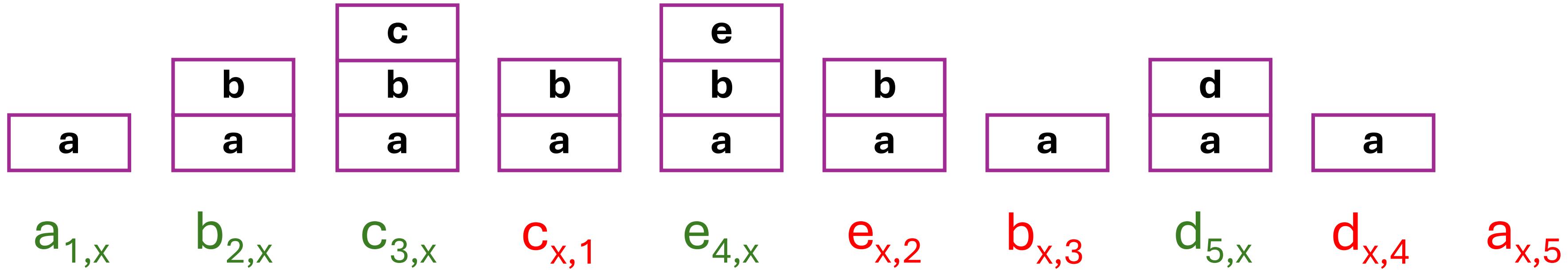




Depth-First Search (DFS)

In the traversal's stack, the first subscript number indicates the order in which a vertex is visited *(i.e., pushed onto the stack)*. The second one indicates the order in which it becomes a dead-end *(i.e., popped off the stack)*.





$a_{1,x}$ $b_{2,x}$ $c_{3,x}$ $c_{x,1}$ $e_{4,x}$ $e_{x,2}$ $b_{x,3}$ $d_{5,x}$ $d_{x,4}$ $a_{x,5}$

$a_{1,5}$

$b_{2,3}$

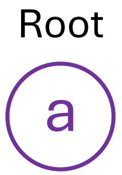
$c_{3,1}$

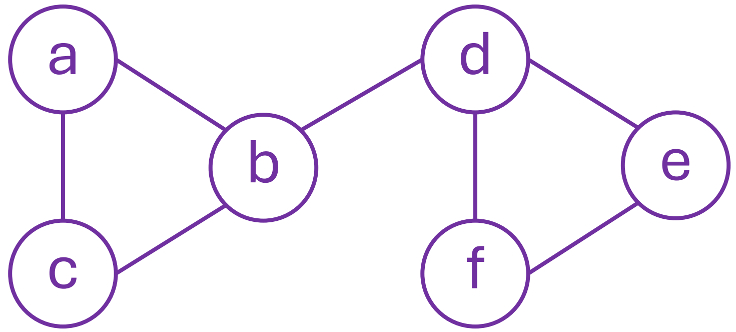
$e_{4,2}$

$d_{5,4}$

Depth-First Search (DFS)

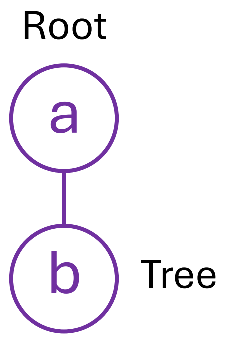
We can also construct a depth-first search **forest.** The starting vertex of the traversal serves as the **root** of the first tree in such a forest.

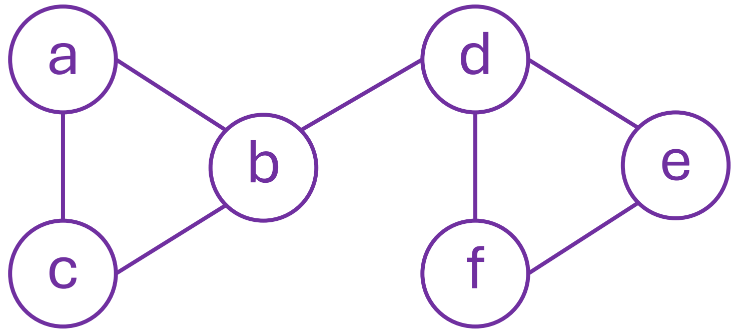




Depth-First Search (DFS)

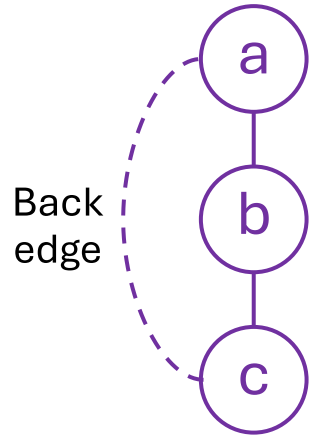
Whenever a new unvisited vertex is reached for the first time, it is attached as a child to the vertex from which it is being reached. Such an edge is called a **tree** edge.

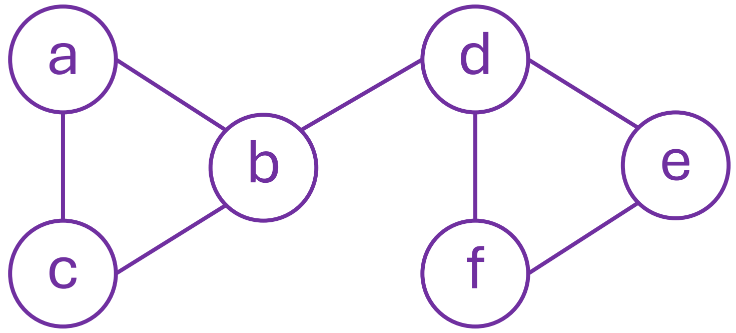




Depth-First Search (DFS)

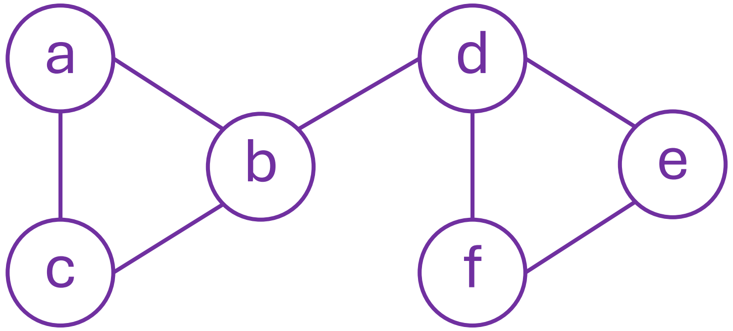
When there exists an edge leading to a previously visited vertex other than its immediate predecessor (i.e., its parent in the tree). Such an edge is called a **back edge** because it connects a vertex to its ancestor, other than the parent.

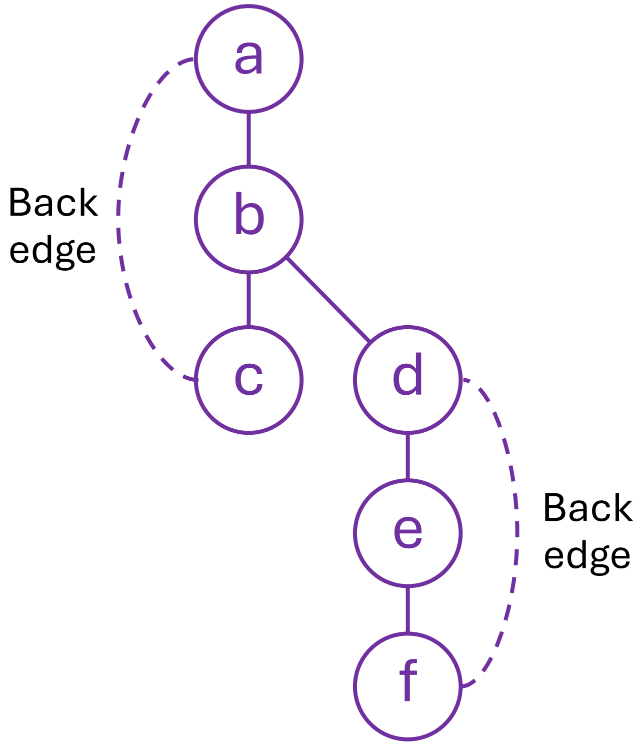




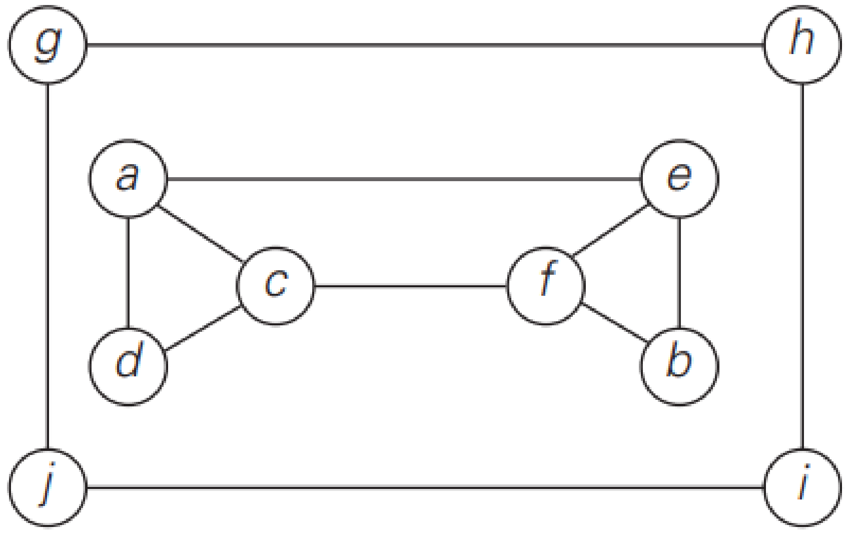
Depth-First Search (DFS)

When there exists an edge leading to a previously visited vertex other than its immediate predecessor (i.e., its parent in the tree). Such an edge is called a **back edge** because it connects a vertex to its ancestor, other than the parent.





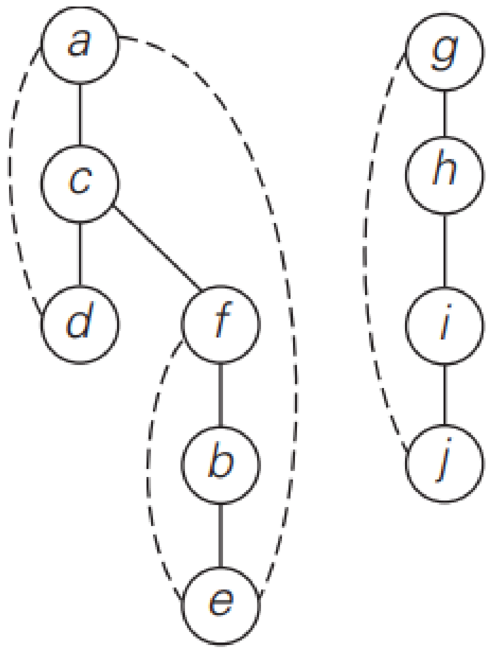
Example: DFS



$e_{6,\,2}$
$b_{5,\,3}$
$f_{4,\,4}$

$d_{3,\,1}$
$c_{2,\,5}$
$a_{1,\,6}$

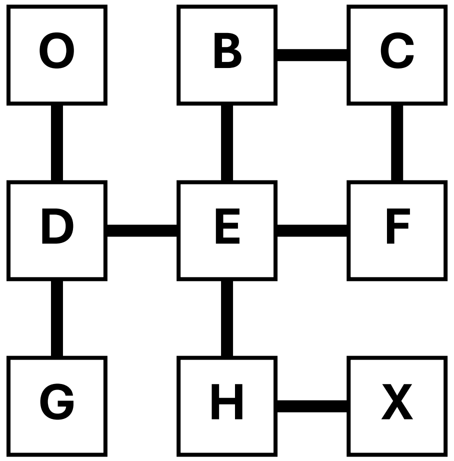
$j_{10,7}$
$i_{9,\,8}$
$h_{8,\,9}$
$g_{7,10}$



Rule: Alphabetical order

Exercise 6.1

Draw a depth-first search forest and a traversal's stack of this maze problem.



Rules:

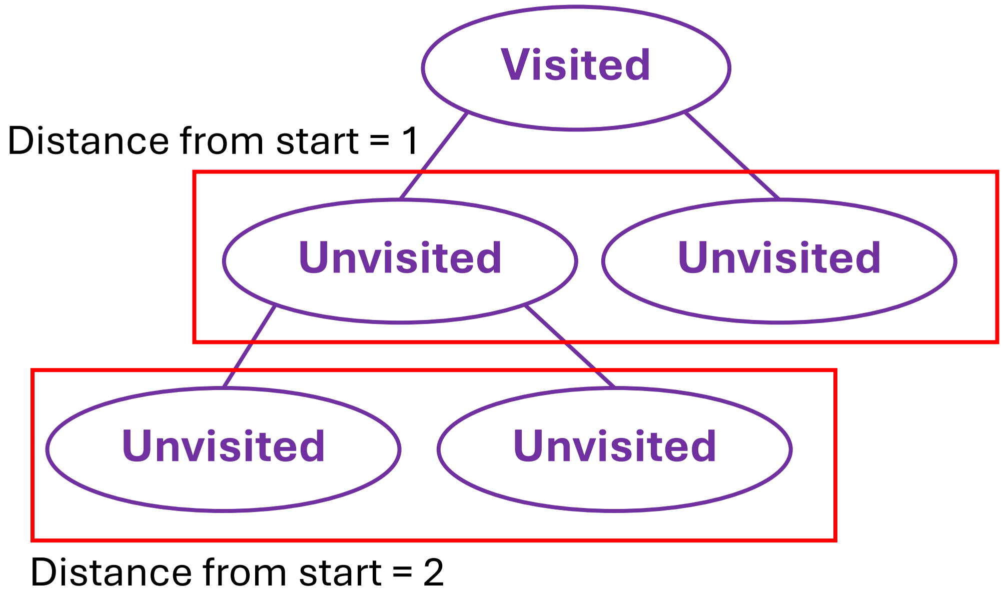
Enter = O, Exit = X

Priority: Down, Right, Left, Up

Breadth-First Search (BFS)

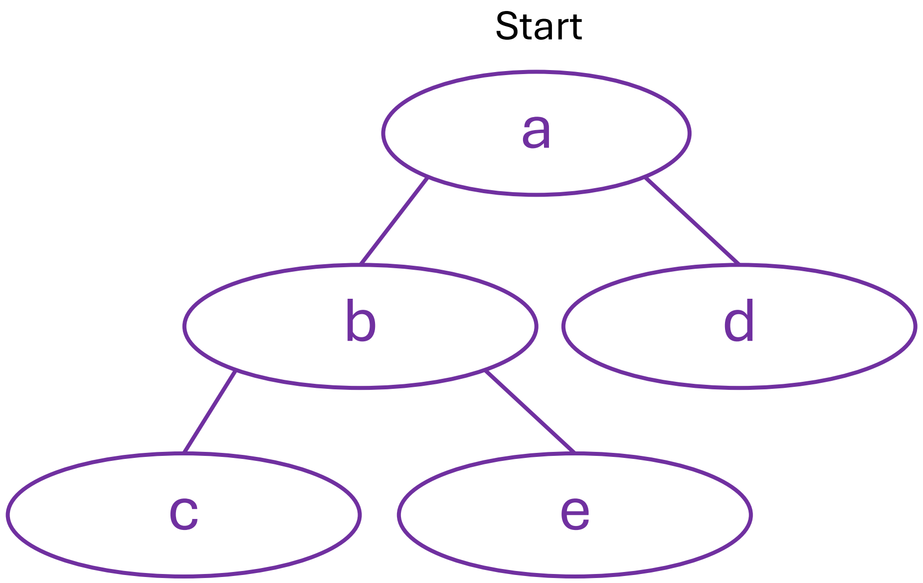
Visit all the vertices that are adjacent to a starting vertex, then all unvisited vertices two edges apart from it, and so on, until all the vertices in the same connected component as the starting vertex are visited.

If there still remain unvisited vertices, the algorithm has to be restarted at an arbitrary vertex of another connected component of the graph.



Breadth-First Search (BFS)

It is convenient to use a **queue** to trace the operation of breadth-first search. The queue is initialized with the traversal’s starting vertex, which is marked as visited.

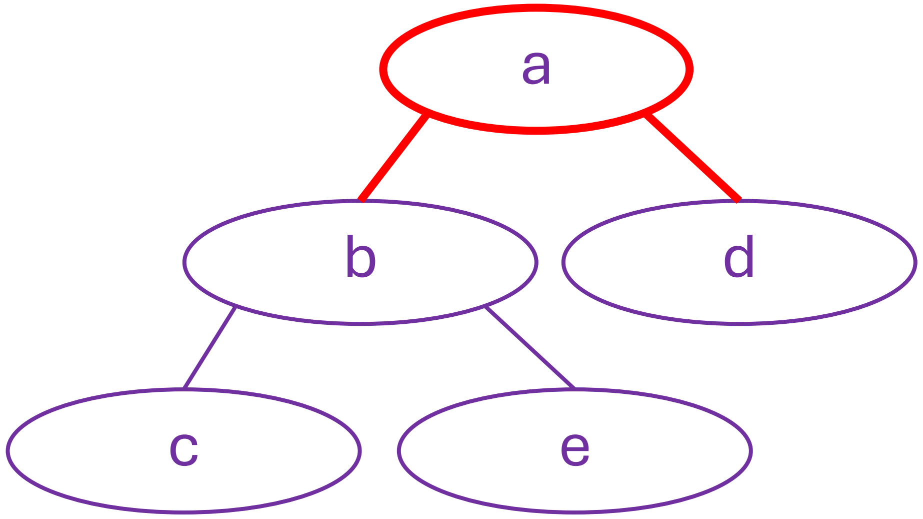


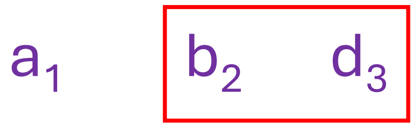
a_1

Start

Breadth-First Search (BFS)

On each iteration, the algorithm identifies all unvisited vertices that are adjacent to the front vertex, marks them as visited, and adds them to the queue.



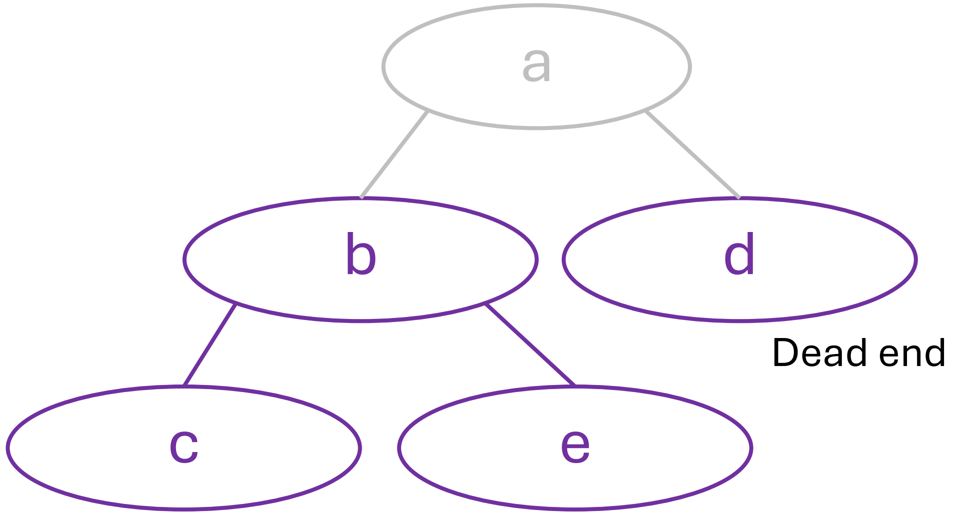


Start

Distance from start = 1

Breadth-First Search (BFS)

After that, the front vertex is removed from the queue.

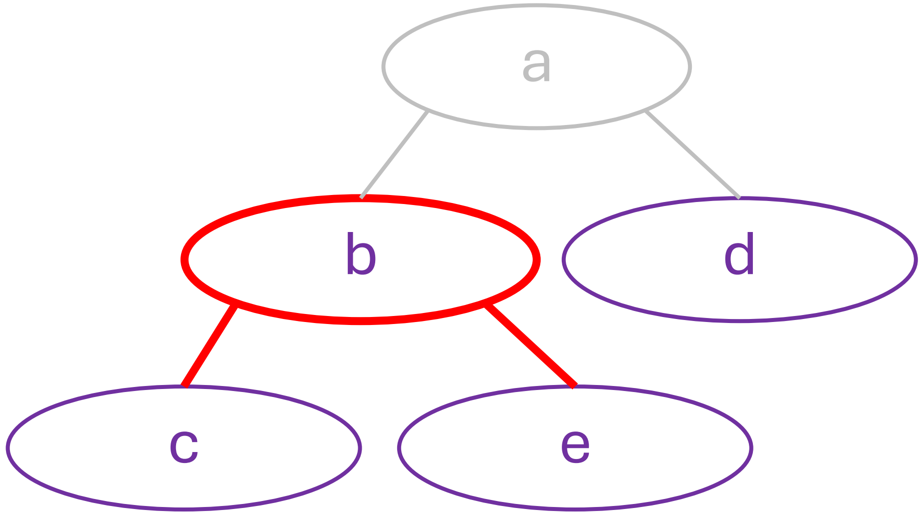


a_1 b_2 d_3

Dead end

Breadth-First Search (BFS)

The search continues until all the vertices in the same connected component as the starting vertex are visited.



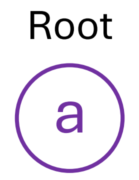
a_1 b_2 d_3 $\boxed{c_4 \quad e_5}$

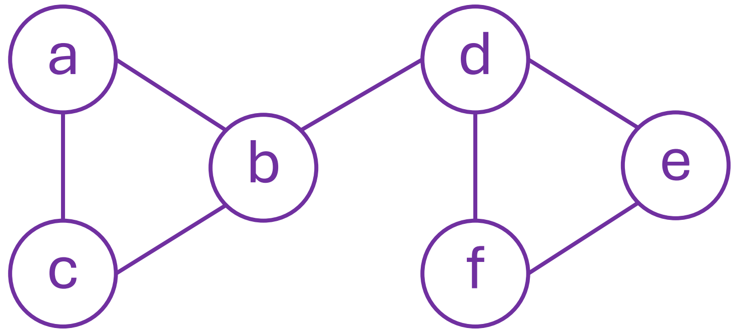
Dead end

Distance from b = 1

Breadth-First Search (BFS)

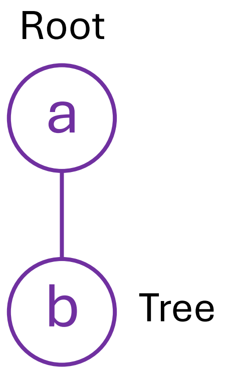
We can also construct a breadth-first search **forest**. The starting vertex of the traversal serves as the **root** of the first tree in such a forest.

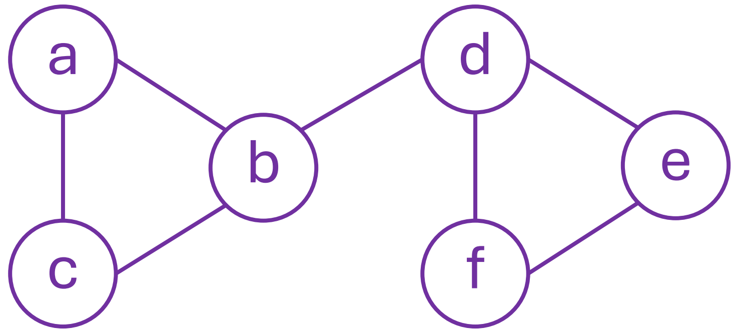




Breadth-First Search (BFS)

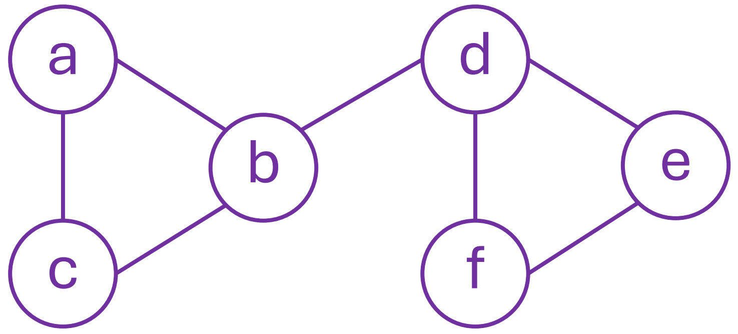
Whenever a new unvisited vertex is reached for the first time, it is attached as a child to the vertex from which it is being reached. Such an edge is called a **tree** edge.

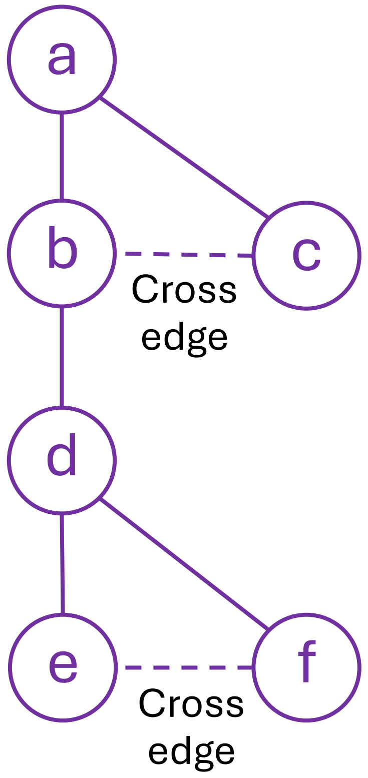




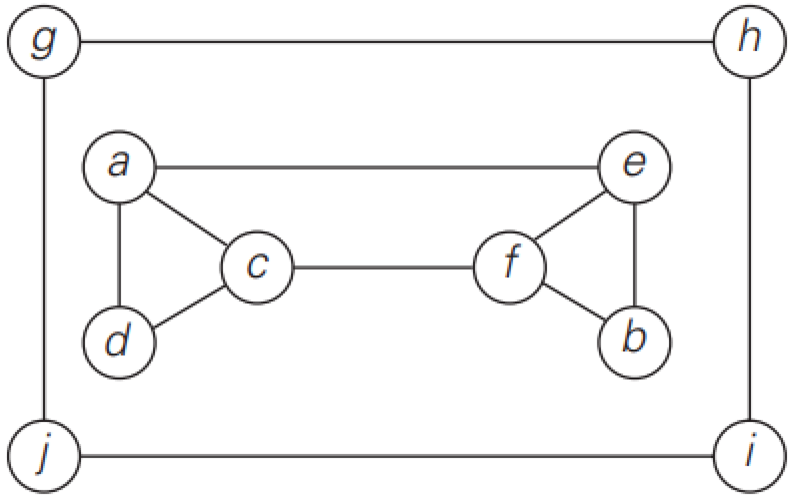
Breadth-First Search (BFS)

If an edge leading to a previously visited vertex other than its immediate predecessor (i.e., its parent in the tree) is encountered, the edge is noted as a **cross edge**.

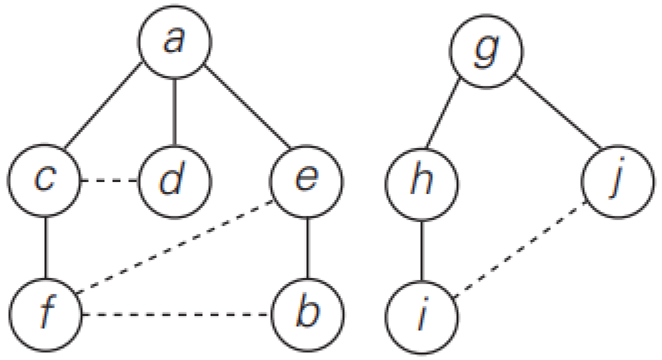




Example: BFS



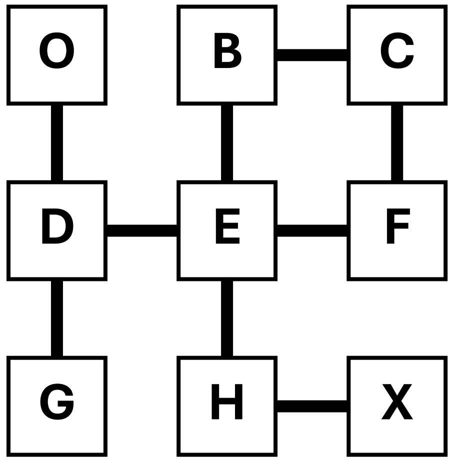
a_1 c_2 d_3 e_4 f_5 b_6
g_7 h_8 j_9 i_{10}



Rule: Alphabetical order

Exercise 6.2

Draw a breadth-first search forest and a traversal's queue of this maze problem.



Rules:

Enter = O, Exit = X

Priority: Down, Right, Left, Up

References

Levitin, A. (2012b). *Introduction to the design & analysis of algorithms* (3rd ed.). Pearson.