Topic 7 Discussion 1

What factors would you consider when you are deciding to use a Depth First Search or a Breadth-First Search when implementing topological sort? Create a Loom video in which you offer your explanation. Paste the link to your video here. Your video should not exceed two minutes.

Topological sorting for Directed Acyclic Graph (DAG) is a linear ordering of vertices such that for every directed edge u v, vertex u comes before v in the ordering. Topological Sorting for a graph is not possible if the graph is not a DAG.

For example, a topological sorting of the following graph is "5 4 2 3 1 0".

A topological ordering is possible if and only if the graph has no directed cycles, that is, if it is a directed acyclic graph (DAG). Any DAG has at least one topological ordering, and algorithms are known for constructing a topological ordering of any DAG in linear time.

DFS is typically implemented with LIFO (a stack) - last in first out.

BFS typically implemented with FIFO (a queue ) - first in first out.

DFS is the best option,for the topological order of the nodes because it reflects their depth in the

graph

Depth first search is more memory efficient than breadth first search and can be implemented with

ease.

In DFS implementation of Topological Sort focussed on sink vertices, i.e, vertices with zero out-

going edges, and then at last reversed the order in which we got the sink vertices by using a stack,

which is a Last In First Out data structure.

In BFS implementation of the Topological sort is the opposite: Look for for edges with no

inbound edges.So here we don’t have to reverse the order in which we get the vertices, since we

get the vertices in order of the topological ordering. We use First-In-First-Out data structure Queue in

this implementation

DFS is typically implemented with LIFO (a stack) - last in first out.

BFS typically implemented with FIFO (a queue ) - first in first out.

DFS is the best option,for the topological order of the nodes because it reflects their depth in the

graph

Depth first search is more memory efficient than breadth first search and can be implemented with

ease.

In DFS implementation of Topological Sort focussed on sink vertices, i.e, vertices with zero out-

going edges, and then at last reversed the order in which we got the sink vertices by using a stack,

which is a Last In First Out data structure.

In BFS implementation of the Topological sort is the opposite: Look for for edges with no

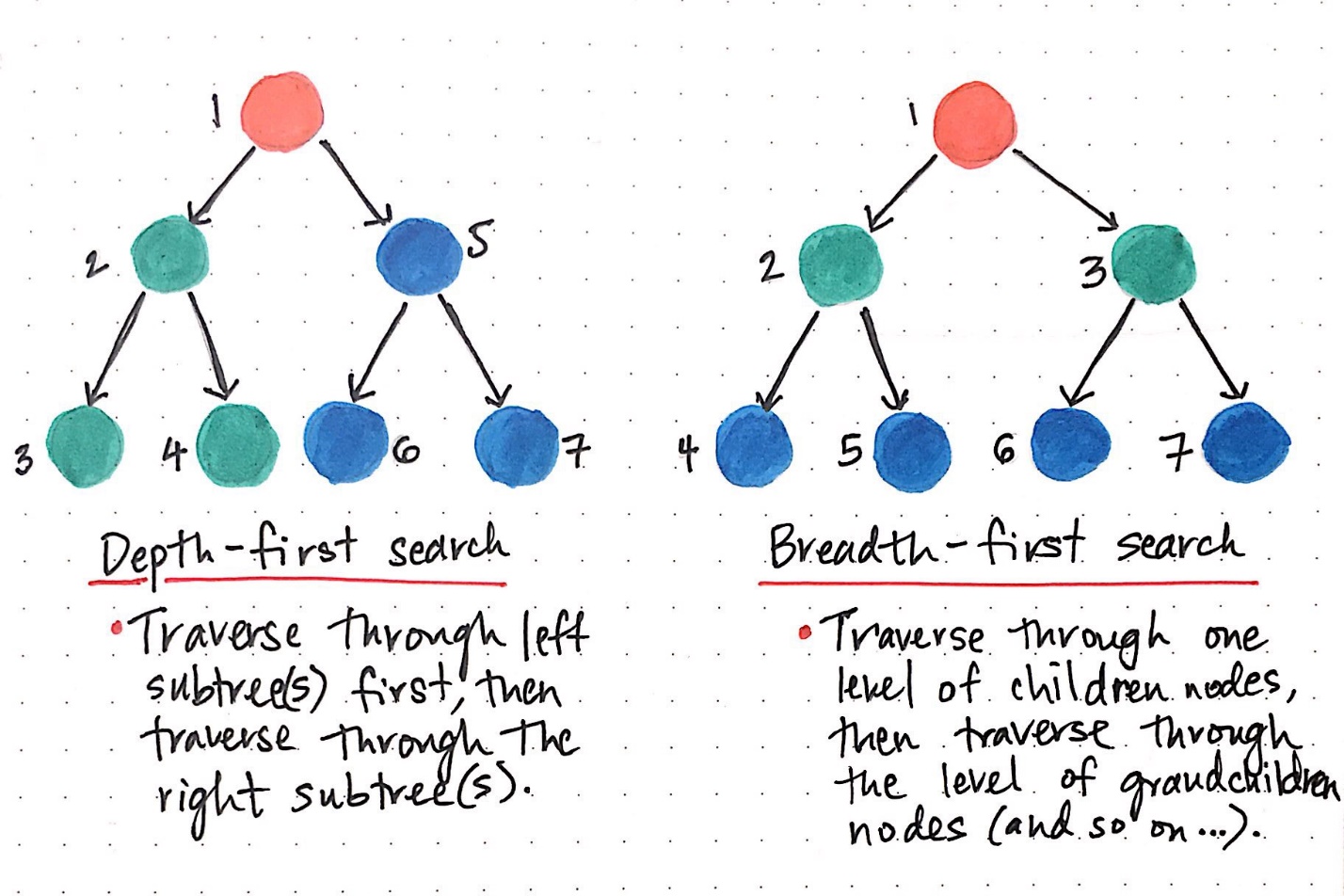
inbound edges.So here we don’t have to reverse the order in which we get the vertices, since we

get the vertices in order of the topological ordering. We use First-In-First-Out data structure Queue in

this implementation

DFS is typically implemented with LIFO (a stack) - last in first out.

BFS typically implemented with FIFO (a queue) - first in first out.



DFS is the best option, for the topological order of the nodes because it reflects their depth in the graph

Depth first search is more memory efficient than breadth first search and can be implemented with ease.

In DFS implementation of Topological Sort focused on sink vertices, i.e., vertices with zero out-going edges, and then at last reversed the order in which we got the sink vertices by using a stack, which is a Last In First Out data structure.

In BFS implementation of the Topological sort is the opposite: L**ook for edges with no inbound edges**. So here we don’t have to reverse the order in which we get the vertices, since we get the vertices in order of the topological ordering. We use First-In-First-Out data structure Queue in this implementation.