A comparison on how iterative deepening depth first search behave

Dr Armin Chitizadeh

UNSW SYDNEY

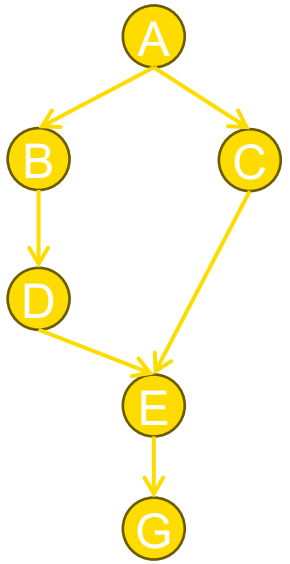
IDDFS should always be implemented with Tree-based DFS

Anytime in this course we use the phrase “Iterative Deepening Depth First Search” or “IDDFS”, we are referring to Tree-based Iterative Deepening Depth First search

Difference of Graph-based IDDFS vs Tree-based IDDFS

In this example, the goal is to find a path from A to G.

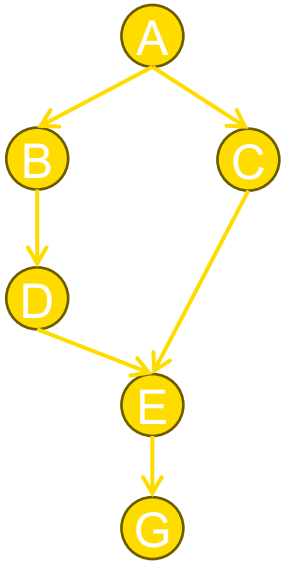
You will see how Graph-based IDDFS will fail on finding the optimal path while Tree-based IDDFS will succeed.



Graph-Based IDDFS

Depth = 1

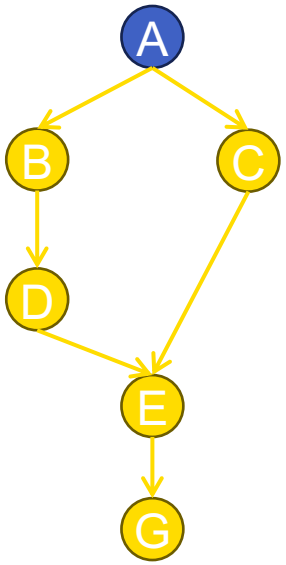
A



visited node: [ ]

Graph-Based IDDFS

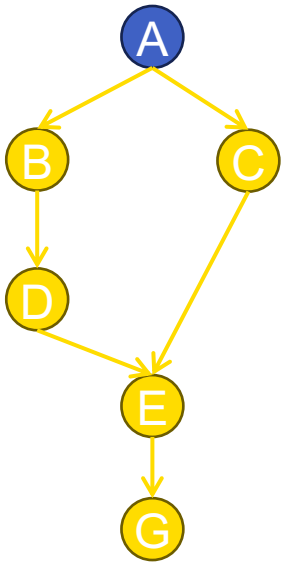
Depth = 1



visited node: [A]

Graph-Based IDDFS

Depth = 1



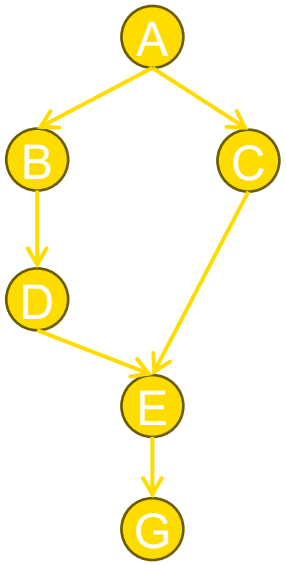
The goal could not be found, so we increase the depth limit

visited node: [A]

Graph-Based IDDFS

Depth = 2

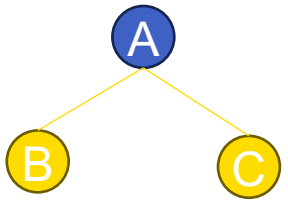
A

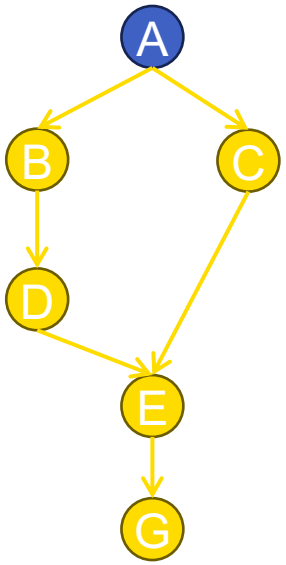


visited node: [ ]

Graph-Based IDDFS

Depth = 2

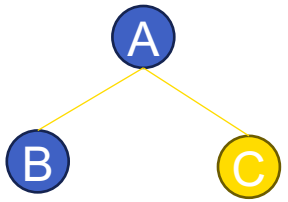


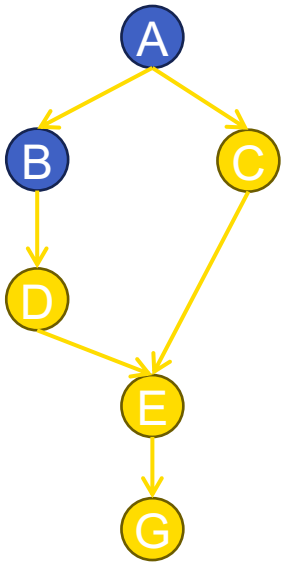


visited node: [A]

Graph-Based IDDFS

Depth = 2

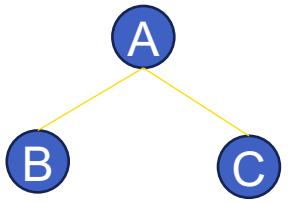


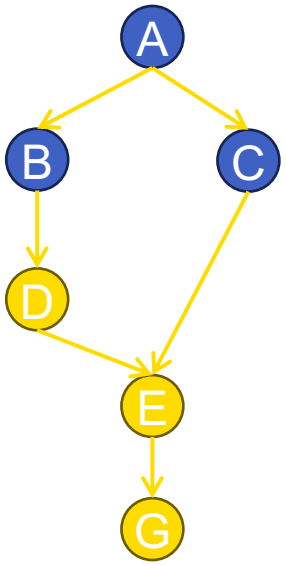


visited node: [A, B]

Graph-Based IDDFS

Depth = 2

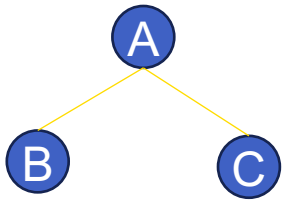


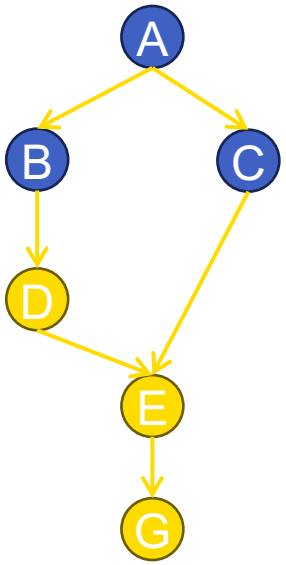


visited node: [A, B, C]

Graph-Based IDDFS

Depth = 2





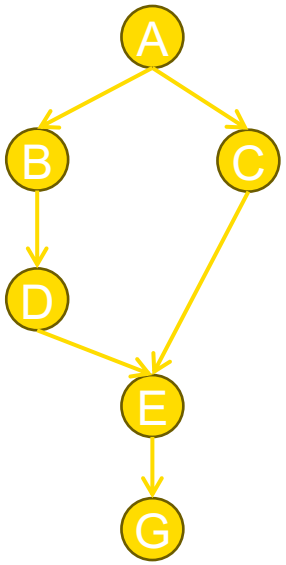
The goal could not be found, so we increase the depth limit

visited node: [A, B, C]

Graph-Based IDDFS

Depth = 3

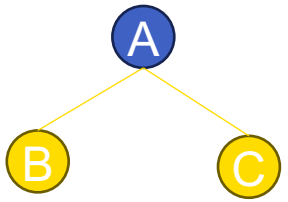
A

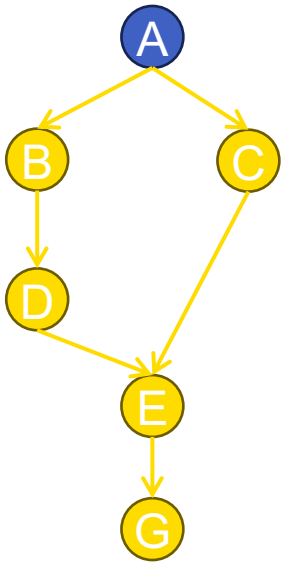


visited node: [ ]

Graph-Based IDDFS

Depth = 3

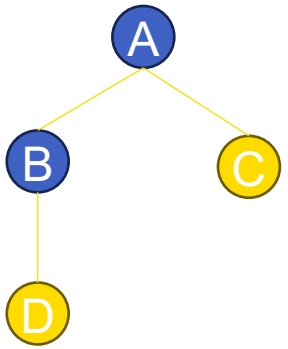


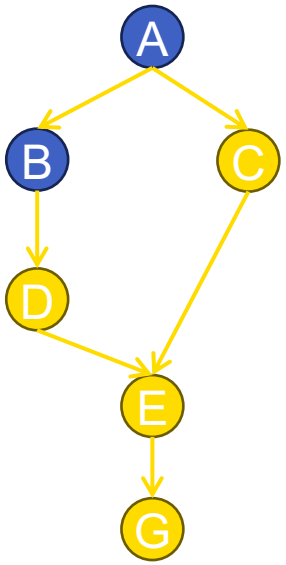


visited node: [A]

Graph-Based IDDFS

Depth = 3

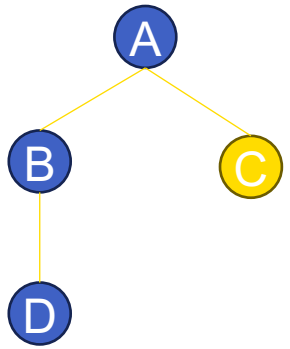


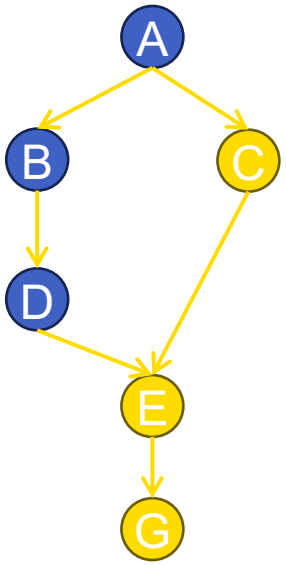


visited node: [A, B]

Graph-Based IDDFS

Depth = 3

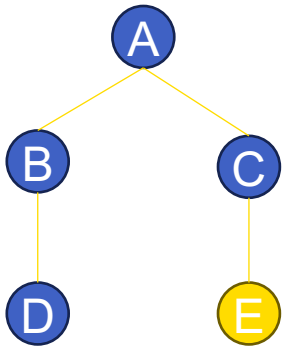


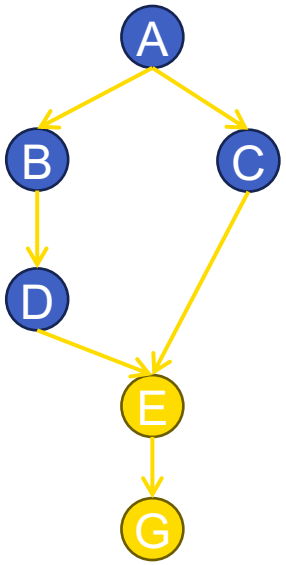


visited node: [A, B, D]

Graph-Based IDDFS

Depth = 3

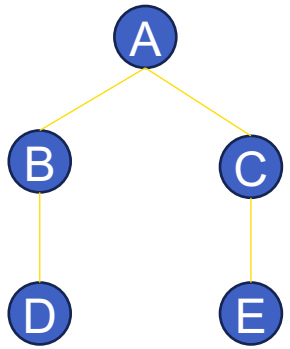


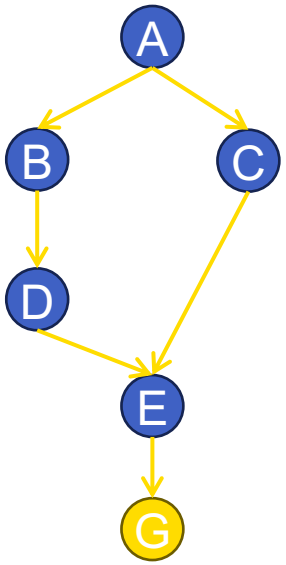


visited node: [A, B, D, C]

Graph-Based IDDFS

Depth = 3

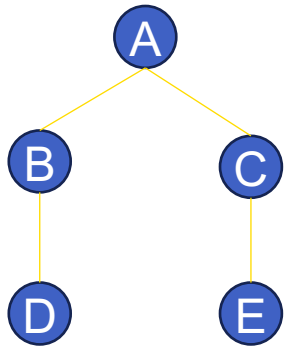


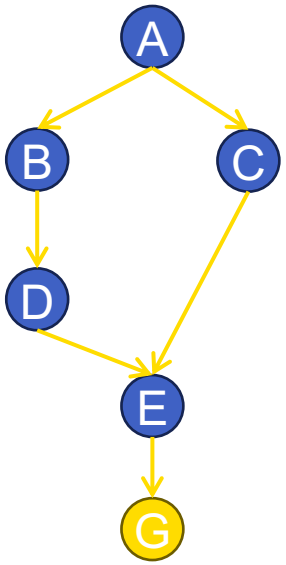


visited node: [A, B, D, C, E]

Graph-Based IDDFS

Depth = 3



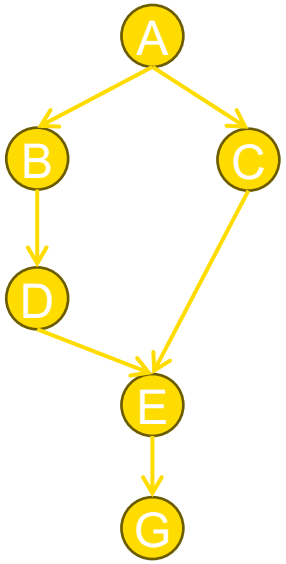


The goal could not be found, so we increase the depth limit

visited node: [A, B, D, C, E]

Graph-Based IDDFS

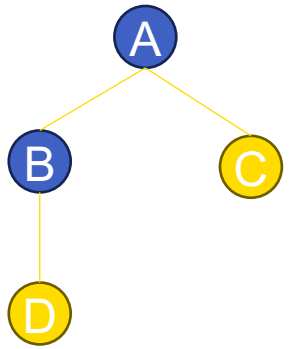
Depth = 4

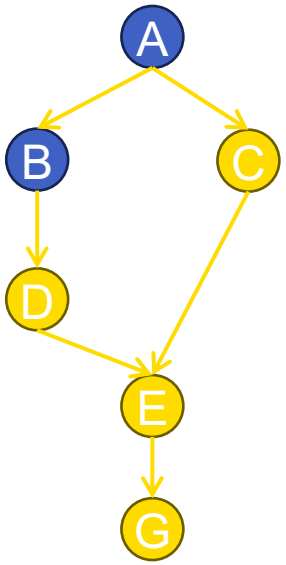


visited node: [ ]

Graph-Based IDDFS

Depth = 4

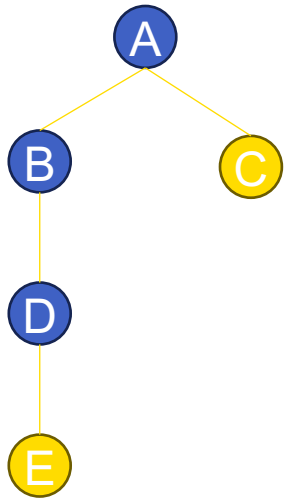


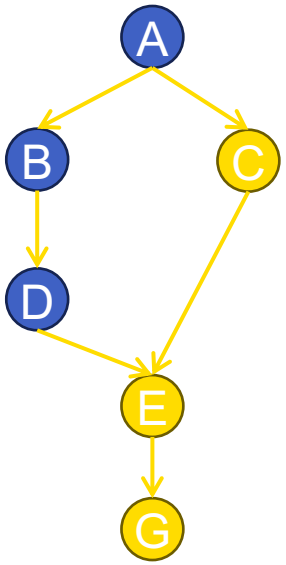


visited node: [A, B]

Graph-Based IDDFS

Depth = 4

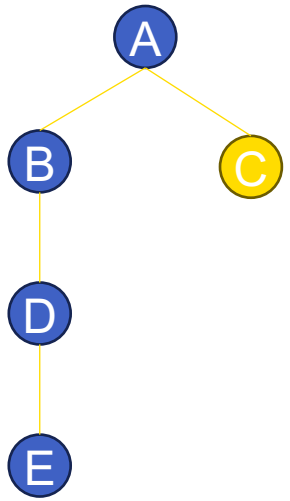


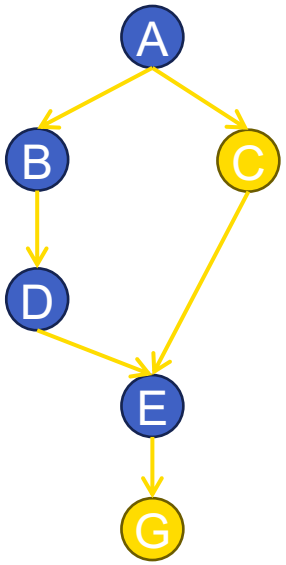


visited node: [A, B, D]

Graph-Based IDDFS

Depth = 4

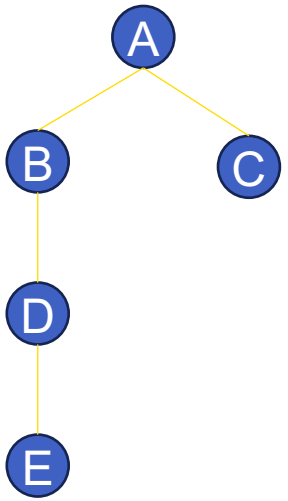


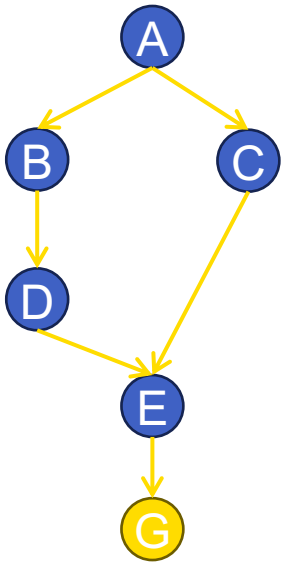


visited node: [A, B, D, E]

Graph-Based IDDFS

Depth = 4

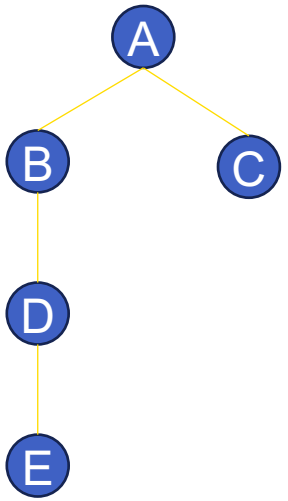




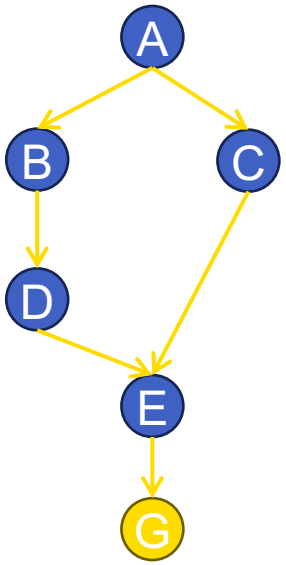
visited node: [A, B, D, E, C]

Graph-Based IDDFS

Depth = 4



See how E can not be generated from C because it is already in the visited node list



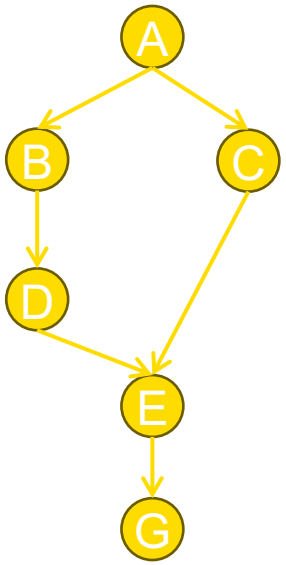
The goal could not be found, so we increase the depth limit

visited node: [A, B, D, E, C]

Graph-Based IDDFS

Depth = 5

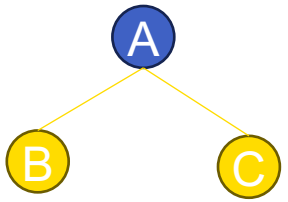
A

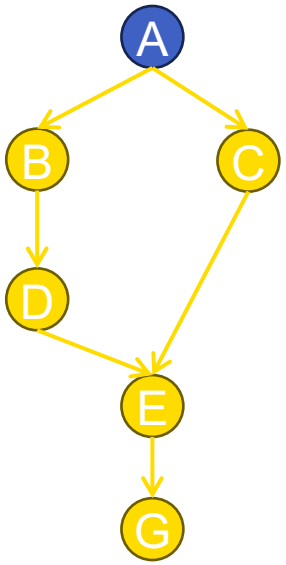


visited node: [ ]

Graph-Based IDDFS

Depth = 5

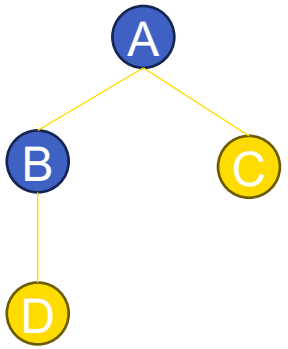


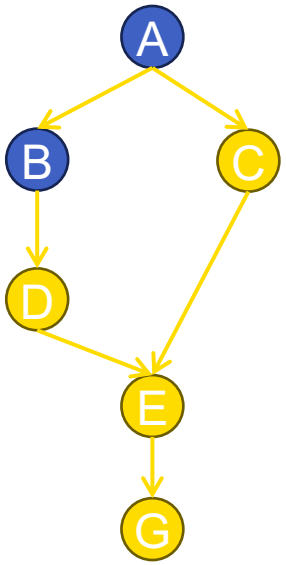


visited node: [A]

Graph-Based IDDFS

Depth = 5

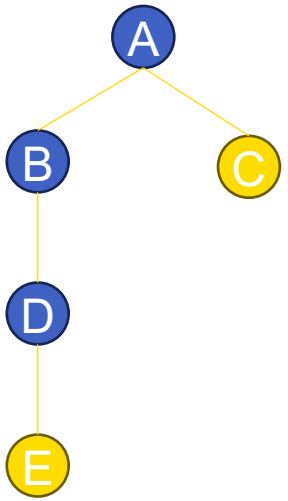


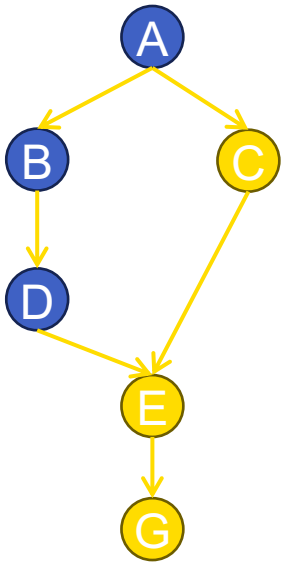


visited node: [A, B]

Graph-Based IDDFS

Depth = 5

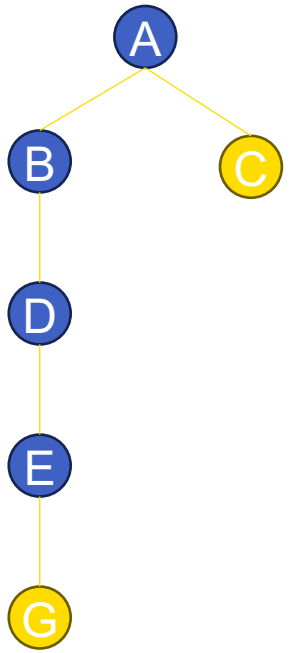


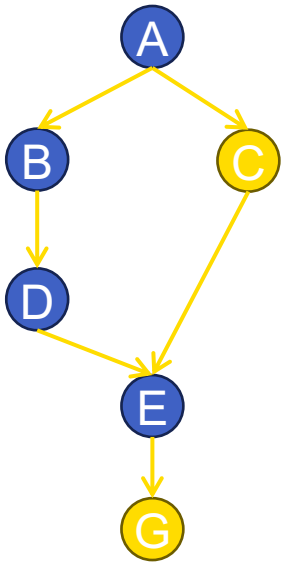


visited node: [A, B, D]

Graph-Based IDDFS

Depth = 5

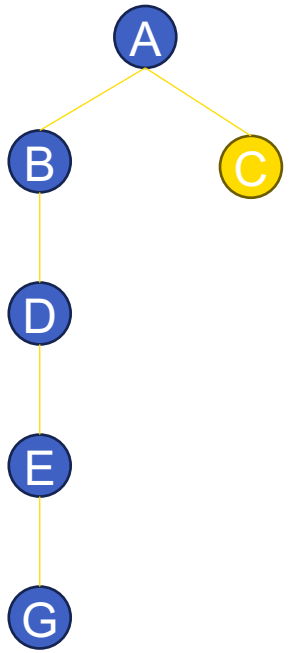


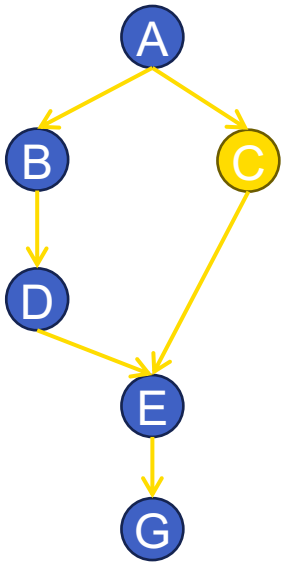


visited node: [A, B, D, E]

Graph-Based IDDFS

Depth = 5

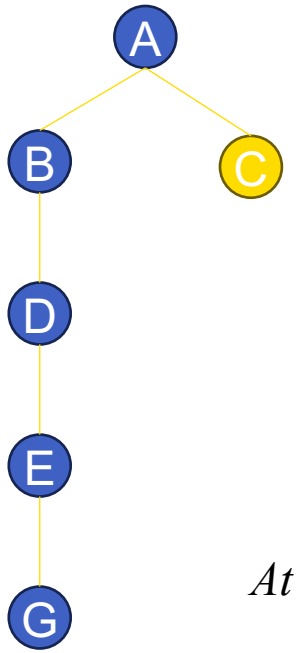


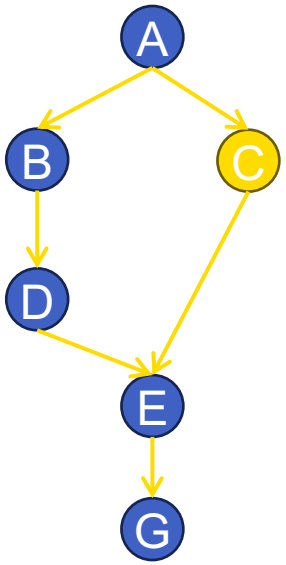


visited node: [A, B, D, E, G]

Graph-Based IDDFS

Depth = 5



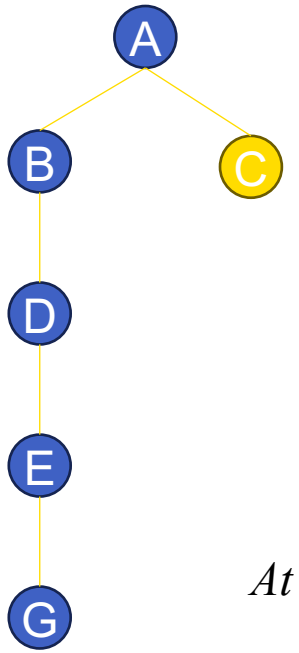


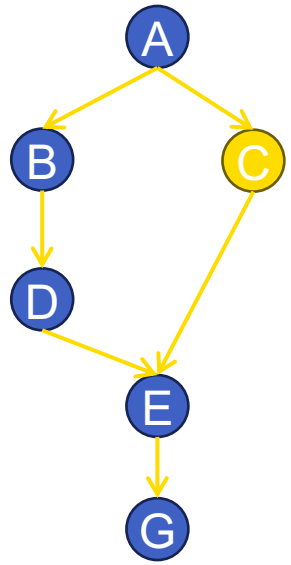
At this stage, the search stops as it expanded node G

visited node: [A, B, D, E, G]

Graph-Based IDDFS

Depth = 5





At this stage, the search stops as it expanded node G

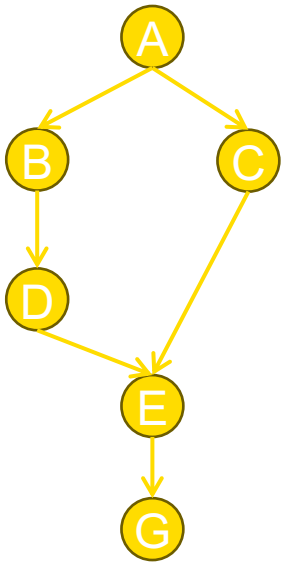
The path that it has discovered is $A \rightarrow B \rightarrow D \rightarrow E \rightarrow G$ which is not the shortest path

visited node: [A, B, D, E, G]

Now we look at how Tree-Based IDDFS can find the shortest path

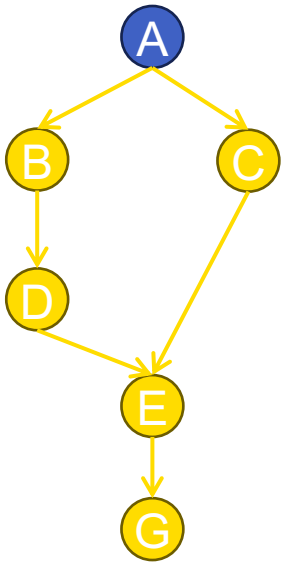
Tree-Based IDDFS

Depth = 1 [] A



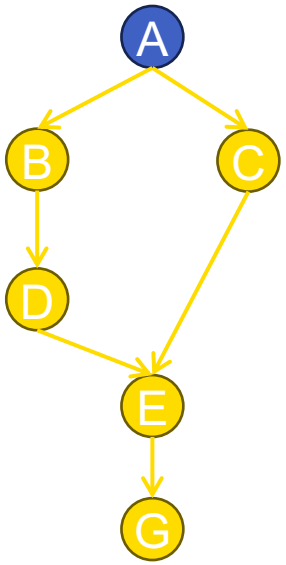
Tree-Based IDDFS

Depth = 1 [] A



Tree-Based IDDFS

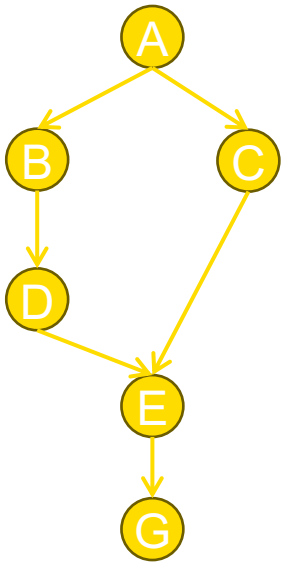
Depth = 1 [] A



The goal could not be found, so we increase the depth limit

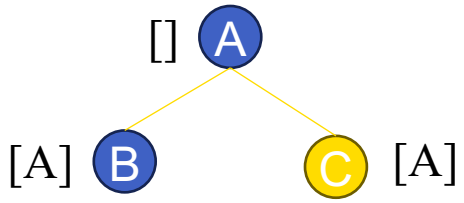
Tree-Based IDDFS

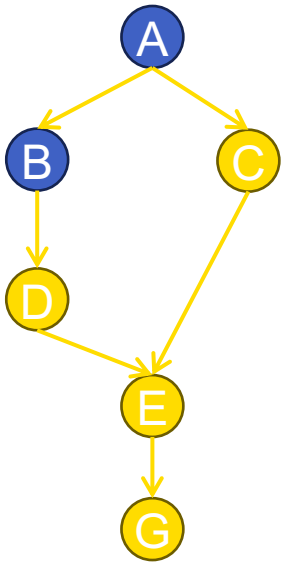
Depth = 2 [] A



Tree-Based IDDFS

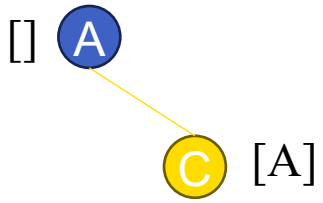
Depth = 2

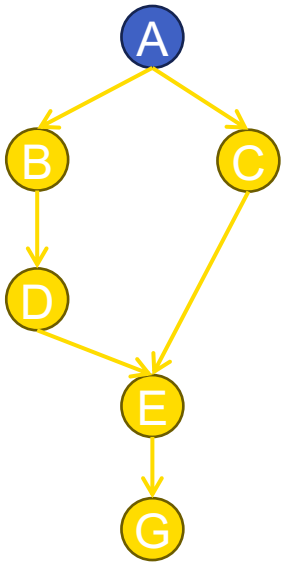




Tree-Based IDDFS

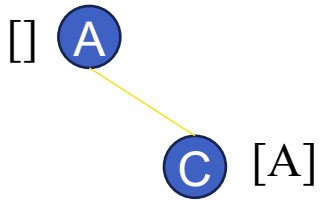
Depth = 2

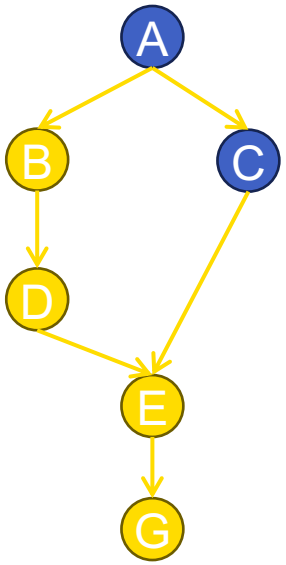




Tree-Based IDDFS

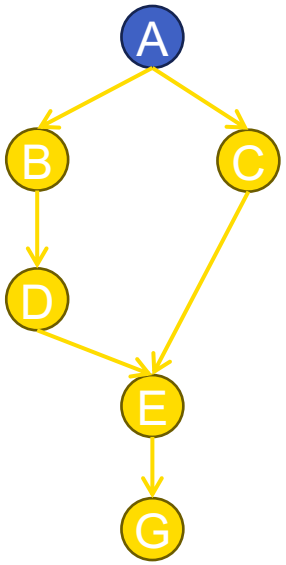
Depth = 2





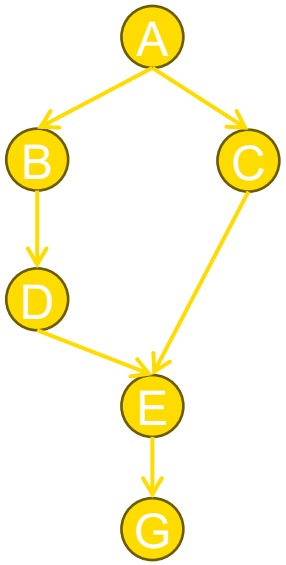
Tree-Based IDDFS

Depth = 2 [] A



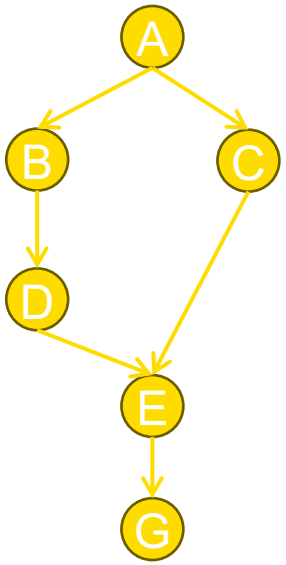
Tree-Based IDDFS

Depth = 2



Tree-Based IDDFS

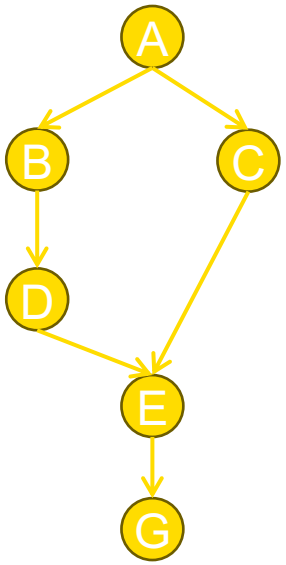
Depth = 2



The goal could not be found, so we increase the depth limit

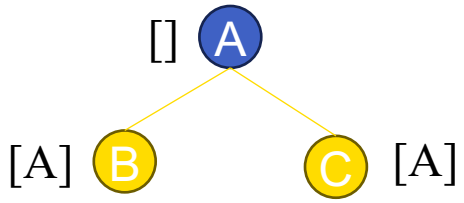
Tree-Based IDDFS

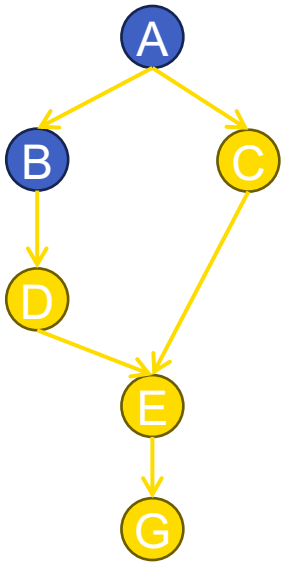
Depth = 3 [] A



Tree-Based IDDFS

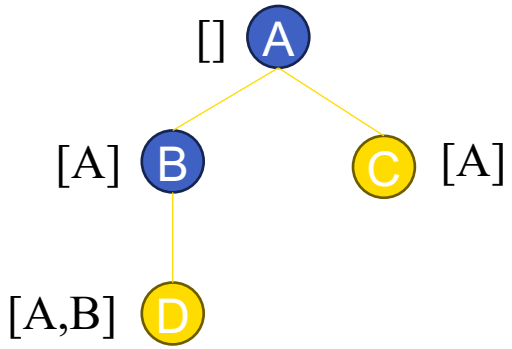
Depth = 3

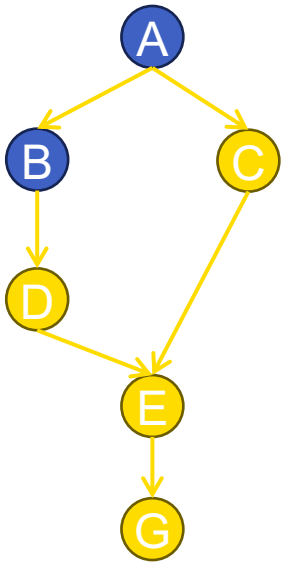




Tree-Based IDDFS

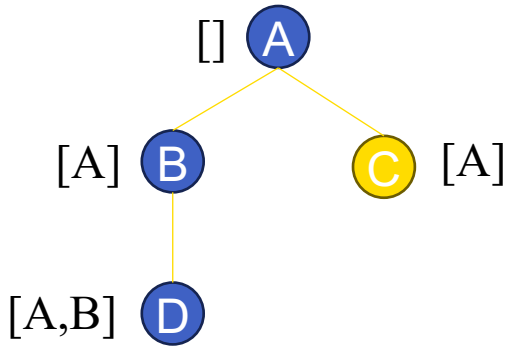
Depth = 3

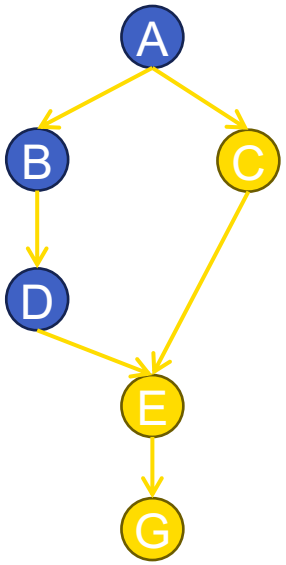




Tree-Based IDDFS

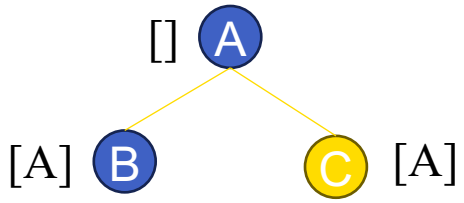
Depth = 3

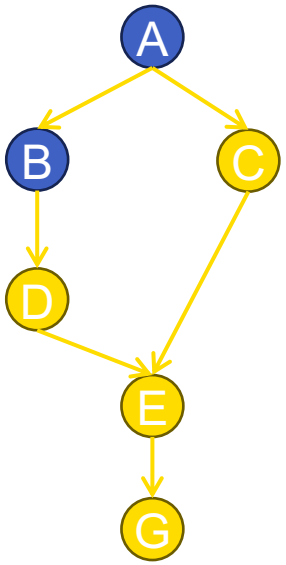




Tree-Based IDDFS

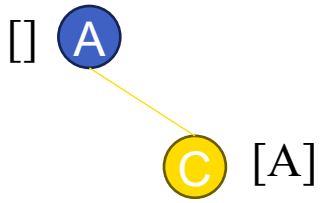
Depth = 3

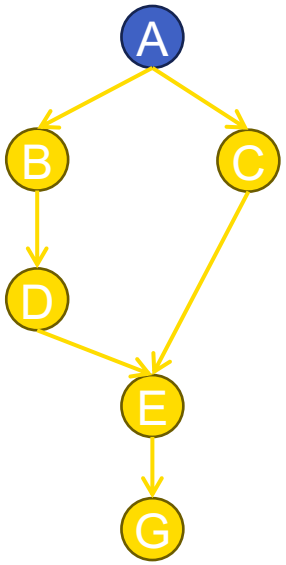




Tree-Based IDDFS

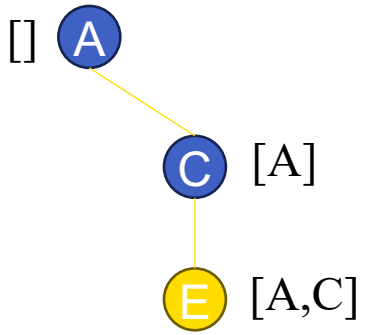
Depth = 3

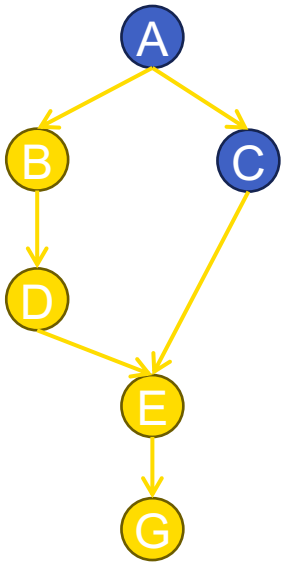




Tree-Based IDDFS

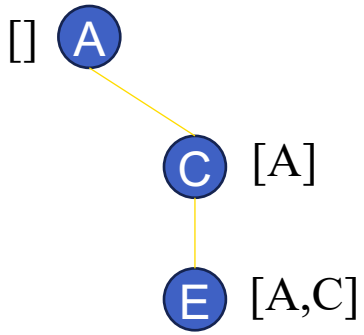
Depth = 3

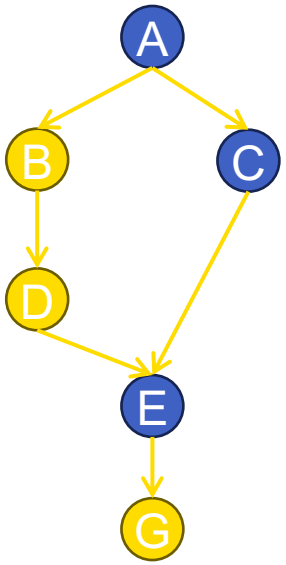




Tree-Based IDDFS

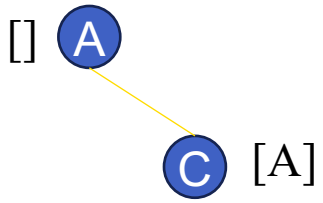
Depth = 3

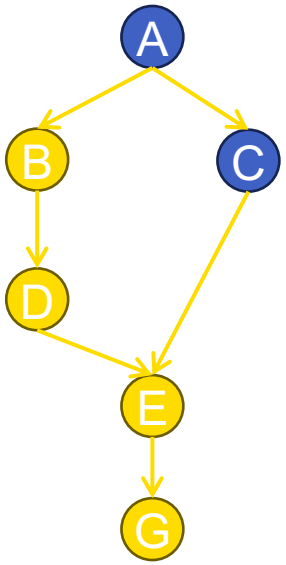




Tree-Based IDDFS

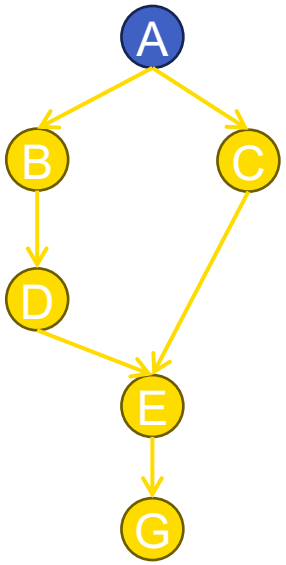
Depth = 3





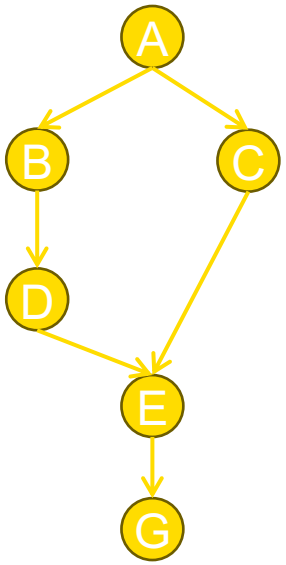
Tree-Based IDDFS

Depth = 3 [] A



Tree-Based IDDFS

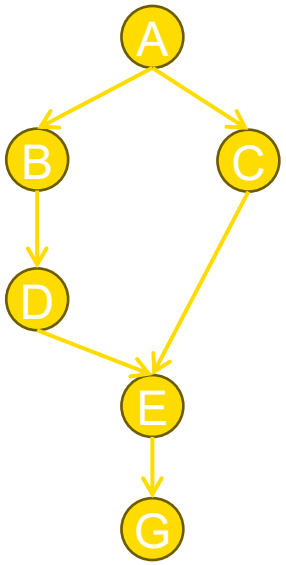
Depth = 3



The goal could not be found, so we increase the depth limit

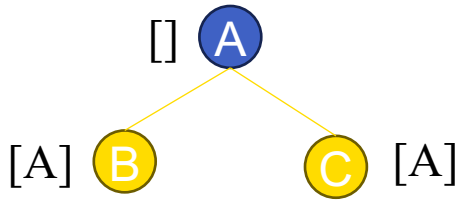
Tree-Based IDDFS

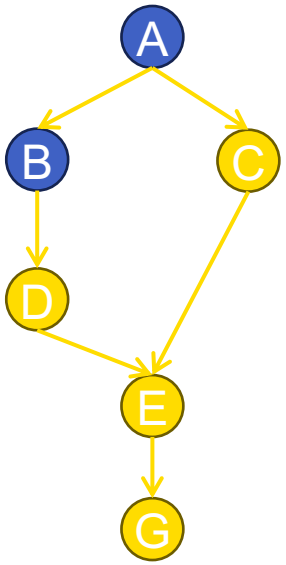
Depth = 4 [] A



Tree-Based IDDFS

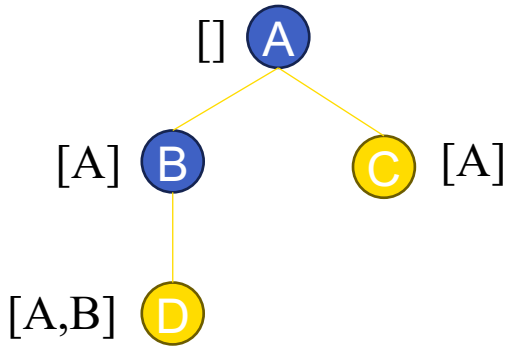
Depth = 4

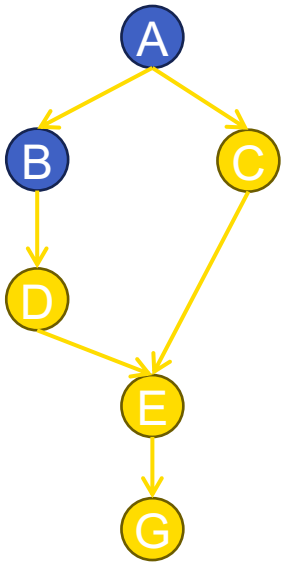




Tree-Based IDDFS

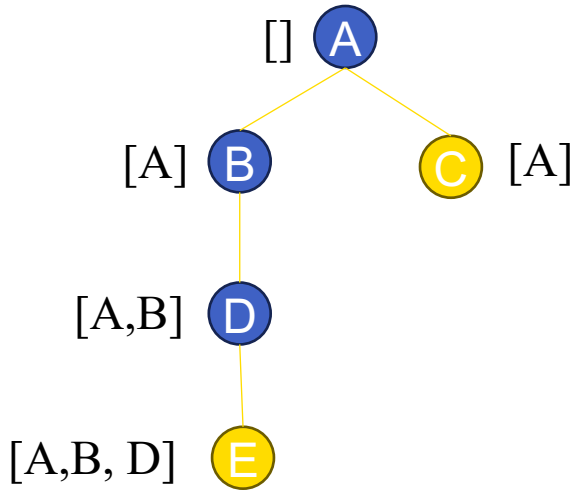
Depth = 4

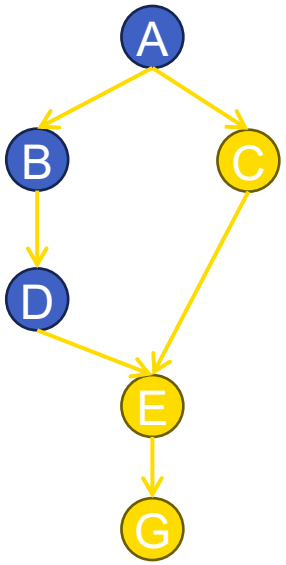




Tree-Based IDDFS

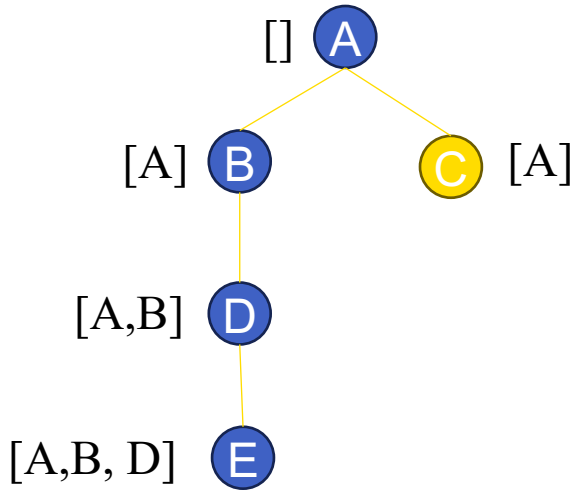
Depth = 4

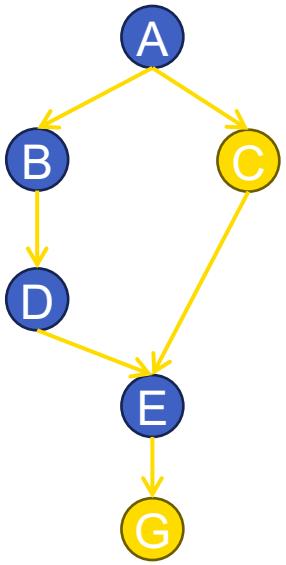




Tree-Based IDDFS

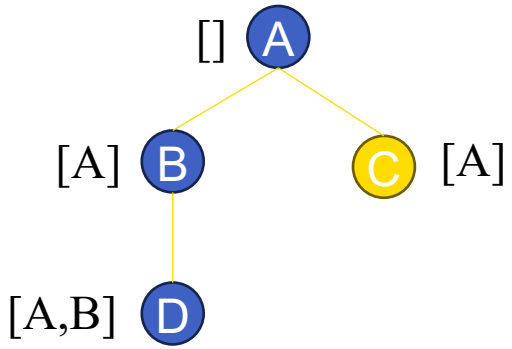
Depth = 4

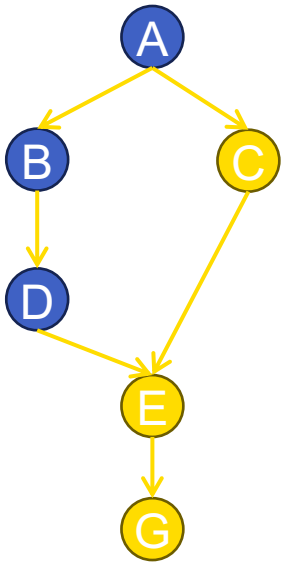




Tree-Based IDDFS

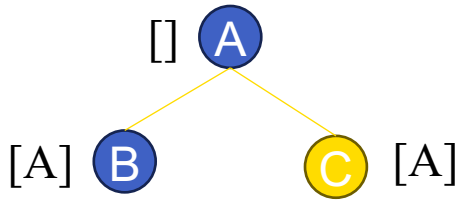
Depth = 4

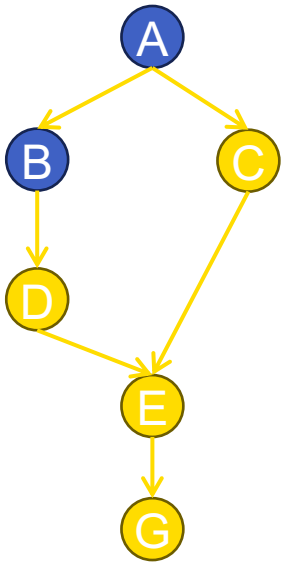




Tree-Based IDDFS

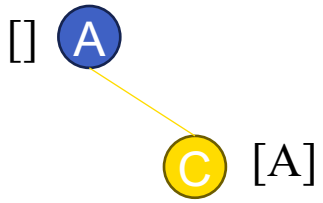
Depth = 4

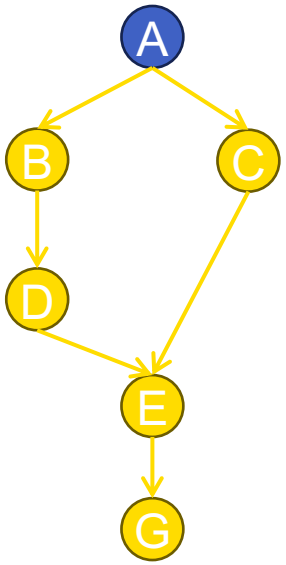




Tree-Based IDDFS

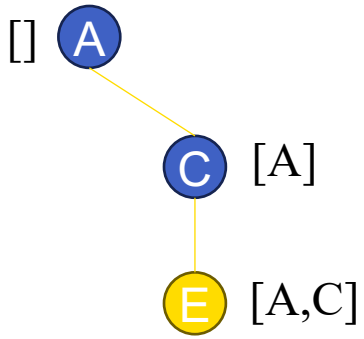
Depth = 4

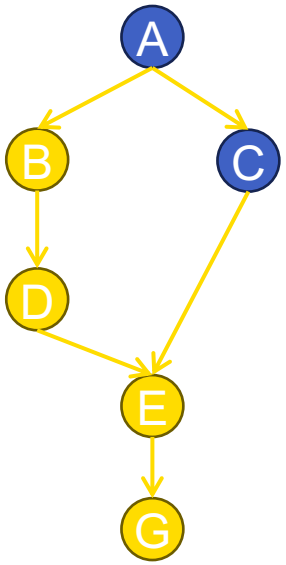




Tree-Based IDDFS

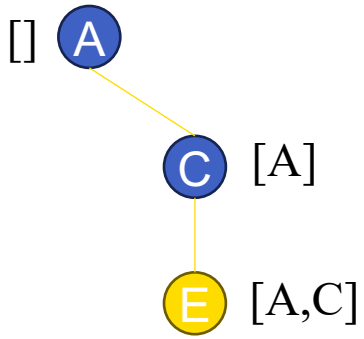
Depth = 4

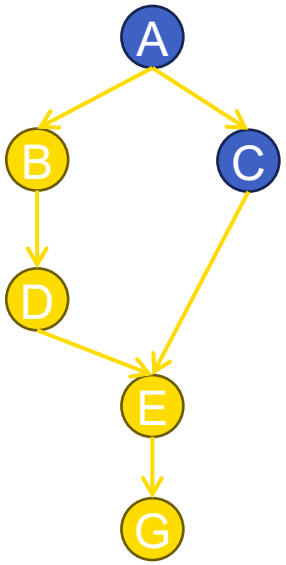




Tree-Based IDDFS

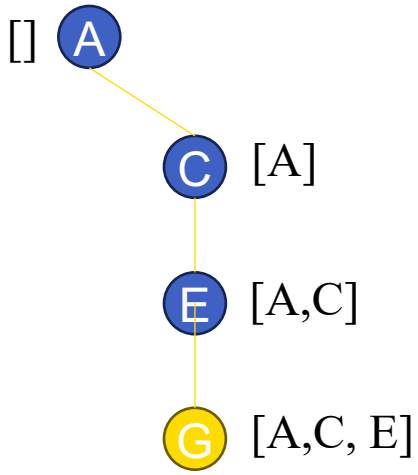
Depth = 4

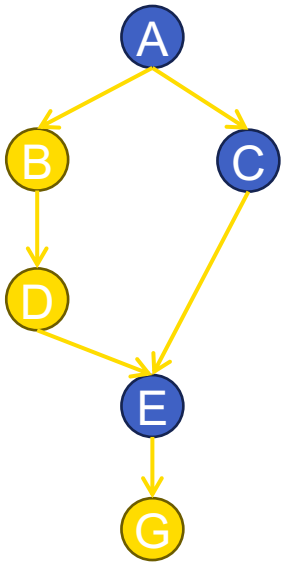




Tree-Based IDDFS

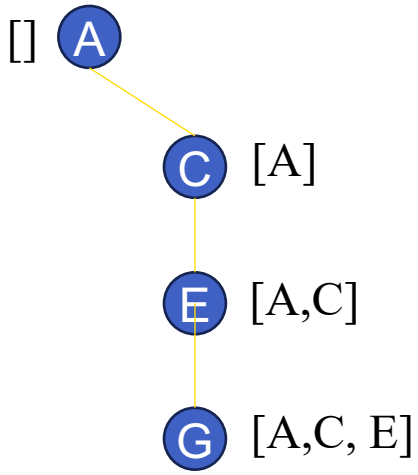
Depth = 4

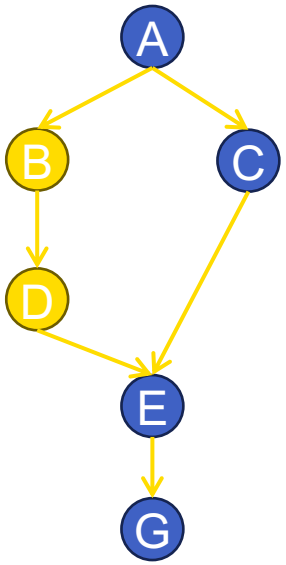




Tree-Based IDDFS

Depth = 4

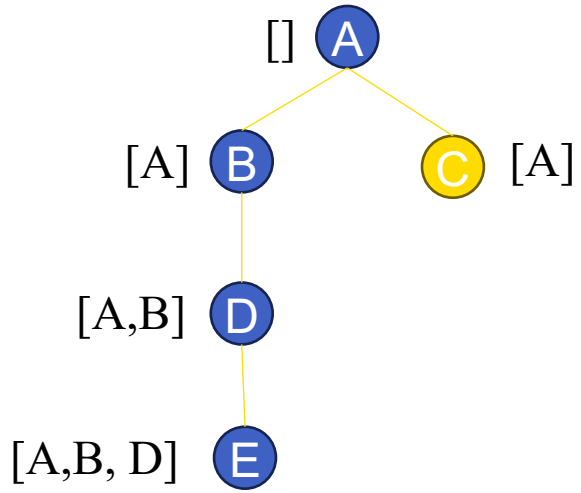




A path to Goal node, G, is found.
It is indeed the shortest path and is A → C → E → G

Further explanation

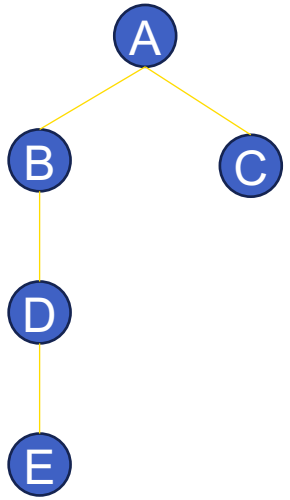
Tree-Based DFS



See how in tree-based DFS search algorithm compares the new nodes only with its ancestors to avoid loop. The ancestor list for each node can be different

Graph-Based DFS

visited node: [A, B, D, E, C]



See how in graph-based DFS search algorithm compares new nodes with a central visited nodes list which is shared among all the nodes.

Feedback

https://forms.office.com/r/NNTtNWh6si