Name: Shashwat Shah

SAP-ID: 60004220126

TY BTECH DIV B, Batch : C22

Aim: Perform uninformed searching techniques like DFID.

Theory:

DFID:

Depth-First Iterative Deepening (DFID) is a search algorithm used in computer science and artificial intelligence to efficiently explore a search space or a tree-like structure, particularly when searching for a target node or state. This algorithm combines the depth-first search strategy with an iterative approach to gradually increase the depth of exploration. The core idea behind DFID is to perform a depth-first search with an initial depth limit and, if the target node is not found within that limit, incrementally increase the depth limit until the target is discovered. This approach offers the best of both worlds: the memory efficiency of depth-first search and the guarantee of finding the shortest path, similar to breadth-first search. It is particularly useful in situations where the search space is unknown or large, and finding the shortest path to a goal is essential.

In practical applications, DFID is employed in scenarios such as puzzle-solving, game-playing, and route-finding problems, where the optimal solution is sought while minimizing memory usage. It provides a balance between memory and time complexity, ensuring that the solution found is the shortest one. By systematically deepening the search, DFID avoids some of the potential pitfalls of pure depth-first search and provides a systematic way to explore complex problem spaces, making it a valuable tool in the toolbox of search and optimization algorithms.

Code:

```
class Node:
    def __init__(self, val=None):
        self.val = val
        self.left = None
        self.right = None


def get_root():
    values = iter([3, 8, 6, 9, None, None, 11, 10, None, None,
                   12, None, None, 7, None, None, 4, 5, None, None, 13, None,
None])

    def tree_recur(itr):
        val = next(itr)
```

```
        if val is not None:
            node = Node(val)
            node.left = tree_recur(itr)
            node.right = tree_recur(itr)
            return node

    return tree_recur(values)

def dfids():
    root = get_root()
    res = float("inf")

    def dfids_search(node, depth, limit):
        if depth <= limit and node is not None:
            val = node.val
            if val == 12:
                nonlocal res
                res = min(res, depth)
            else:
                dfids_search(node.left, depth + 1, limit)
                dfids_search(node.right, depth + 1, limit)

    for limit in range(1,5):
        dfids_search(root, 0, limit)
        if res < float("inf"):
            return res
    return -1

if __name__ == "__main__":
   print("\nShortest Depth: ", dfids())
```

output:

```
Shortest Depth:  4
```

Conclusion:

Thus, we successfully studied Uninformed Searching Algorithms like DFID.