

Pseudo code:

```
def IDS(G, start, max_depth):
    path = [start]
    for depth_bound in max_depth:
        result = DBS(start, depth_bound, path)
        if result is found:
            return path
        if result not found:
            continue
    return None if the thing have no solution

def DBS(node, remaining_depth, path):
    if the current node is prime:
        return the path leading to that (solution is founded)

    if remaining depth == 0:
        return depth bound reached can't go further down

    for each child in node:
        if child not in path (Detect cycle):
            add child into the path
            result = DBS(child, goal, remaining_depth - 1, new_path)
            popping the path to backtrack
            if result is found:
                return result

    return None
```

Explanation:

So we search through every node at depth limit provided (0 ongoing till max depth) using depth-first search. If we go through all nodes without max depth being the current limit depth provided and don't find a prime, it means we didn't search deep enough. So we increase the allowed search depth by 1 and research the whole graph again, but this time allowing one more level deeper. We repeat this process until a prime number is found. If we detect a cycle during any path, we abandon that path and move on to the next one

Time complexity:

Best case it's $O(1)$ due to the fact that the prime number could be the 1st value, the start node in which case it's going to be found out and return immediately

Worst case it's $O(b^d)$, b is the number of children at each node and d is the level depth, because if it's the worst case, then every level is going to be checked until the last one and for each level they will need to recheck the already checked part of the previous level.