{|z

Name: Dawood Iqbal Reg No: L1F21BSCS1080 Teacher Name: Sir Buzdar		Total Marks: 05 Due Date: 21/04/2024 AI Lab F11- (Wednesday)

Problem No.1

Explain the Iterative Deepening A* algorithm (IDA*), draw his graph and write 02 codes of Iterative Deepening A* algorithm (IDA*) in Jupyter Notebook and past the screen shots of these codes.

Note: Students must mention their names in these codes as a comment at the top of code.

Solution:

Iterative Deepening A* (IDA*) is a search algorithm that combines the principles of both Iterative Deepening Depth-First Search (IDDFS) and the A* algorithm. It aims to find the shortest path from a start node to a goal node in a graph while minimizing memory usage.

Initialization:

- ***Set the initial depth limit to a small value.***
- ***Initialize the search with the start node and set the cost of the initial node to 0.***

Depth-Limited Search:

- ***Perform a depth-limited search (DLS) starting from the initial node with the current depth limit.***
- ***DLS is like depth-first search (DFS) but with a depth limit, meaning it explores nodes up to a certain depth.***
- ***During the search, keep track of the minimum cost encountered so far and update it whenever a better solution is found.***

Iterative Deepening:

- ***If the goal node is not found within the current depth limit, increase the depth limit and repeat the search.***
- ***This iterative deepening process continues until the goal node is found or until the entire graph is explored.***

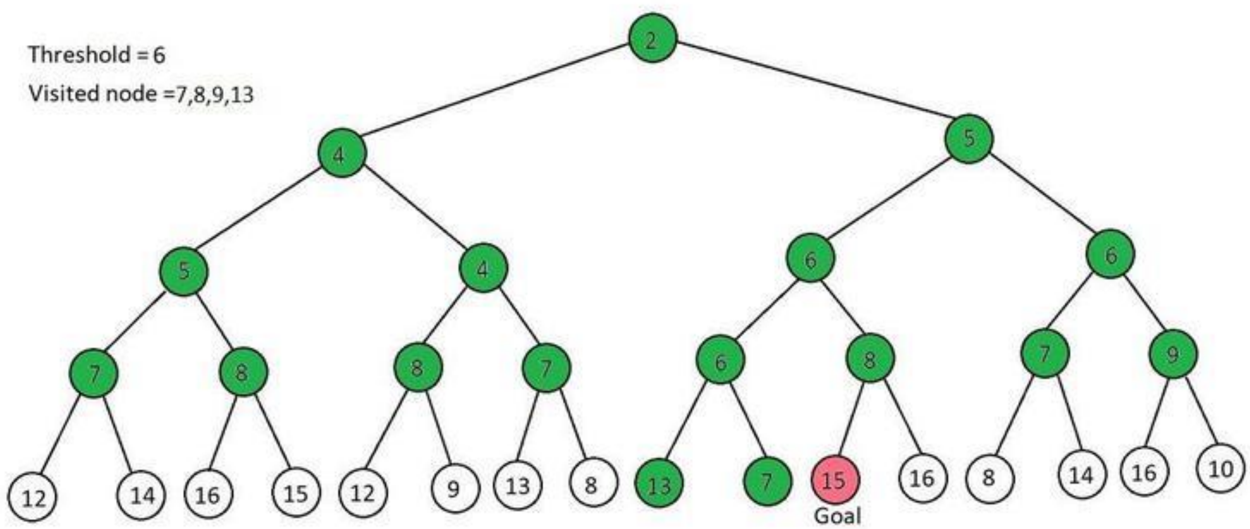
A Heuristic Search:

- ***A* search is used within each depth-limited search iteration to guide the search towards the goal node efficiently.***
- ***A* uses a heuristic function to estimate the cost of reaching the goal from each node.***
- ***The heuristic function helps prioritize exploration of nodes that are likely to lead to the goal more quickly.***

Termination:

- ***Terminate the search when the goal node is found or when the entire graph has been explored.***

Graph:



Code: 1:

CODE: 1 Dawood Iqbal(1080)

```
def heuristic(node):
    return 0


def goal_test(node):
    return node == 5


def successors(node):
    successors = []
    for i in range(node + 1, 6):
        successors.append((i, 1))
    return successors


def cost(node1, node2):
    return 1


def ida_star_search(start_node, heuristic_fn, goal_test_fn, successors_fn, cost_fn):
    threshold = heuristic_fn(start_node)

    while True:
        print("Threshold:", threshold)
        result, new_threshold = depth_limited_search(start_node, 0, threshold, heuristic_fn, goal_test_fn, successors_fn, cost_fn)
        if result == "FOUND":
            return threshold
        if result == "CUTOFF":
            threshold = new_threshold
        if result == float('inf'):
            return None
def depth_limited_search(node, cost, threshold, heuristic_fn, goal_test_fn, successors_fn, cost_fn):
    stack = [(node, cost)]
    min_cost = float('inf')

    while stack:
        node, current_cost = stack.pop()

        estimated_cost = current_cost + heuristic_fn(node)
        if estimated_cost > threshold:
            min_cost = min(min_cost, estimated_cost)
            continue

        if goal_test_fn(node):
            return "FOUND", estimated_cost

        cutoff = False
        for successor, step_cost in successors_fn(node):
            stack.append((successor, current_cost + step_cost))
            if current_cost + step_cost + heuristic_fn(successor) > threshold:
                cutoff = True

        if cutoff:
            min_cost = min(min_cost, current_cost + step_cost + heuristic_fn(successor))

    return "CUTOFF", min_cost

start_node = 0
goal_test = (2,2)
result = ida_star_search(start_node, heuristic, goal_test, successors, cost)
print("Threshold for optimal path found by IDA* algorithm:", result)
```

Code: 2:

CODE: 2 Dawood Iqbal(1080)

```python
def heuristic(node):
    return 0


def goal_test(node):
    return node == 5


def successors(node):
    successors = []
    for i in range(node + 1, 6):
        successors.append((i, 1))
    return successors


def cost(node1, node2):
    return 1


def ida_star_search(start_node, heuristic_fn, goal_test_fn, successors_fn, cost_fn):
    threshold = heuristic_fn(start_node)

    while True:
        print("Threshold:", threshold)
        result, _ = depth_limited_search(start_node, 0, threshold, heuristic_fn, goal_test_fn, successors_fn, cost_fn)
        if result == "FOUND":
            return threshold
        if result == float('inf'):
            return None
        threshold = result
def depth_limited_search(node, cost, threshold, heuristic_fn, goal_test_fn, successors_fn, cost_fn):
    estimated_cost = cost + heuristic_fn(node)

    if estimated_cost > threshold:
        return threshold

    if goal_test_fn(node):
        return "FOUND"

    min_cost = float('inf')

    for successor, step_cost in successors_fn(node):
        new_cost = cost + step_cost
        result = depth_limited_search(successor, new_cost, threshold, heuristic_fn, goal_test_fn, successors_fn, cost_fn)
        if result == "FOUND":
            return "FOUND"
        if result < min_cost:
            min_cost = result

    return min_cost


start_node = 0
result = ida_star_search(start_node, heuristic, goal_test, successors, cost)
print("Threshold for optimal path found by IDA* algorithm:", result)
```

********************************** **The End** **************************************