Name: **Aqsa Junaid**

Roll No.: **221004**

TIME COMPLEXITIES OF GRAPH ALGORITHMS

1. BFS:

Time Complexity is : **O(B^d),** where B is the branch factor and d is the depth of the shallowest goal node.

Space Complexity: **O(B^d)**

A tree with branch factor B and all leaves are of depth d - there are exactly 1 + B + B^2 + B^3 + ... + B^d = B^(d) nodes in the tree.

Proof:

At level 1: There is 1 node (the root node).

At level 2: There can be at most b nodes (children of the root).

At level 3: Each node at level 2 can have at most b children, so there can be at most b^2 nodes at level 3.

At level d: Each node at level d−1 can have at most b children, so there can be at most b^d−1 nodes at level d.

Total nodes: Adding up all the nodes at each level, we get 1+b+b^2+...+b^d−1. This is a geometric series, and its sum can be expressed as = **1+b+b^2+...+b^d−1= O(b^d)**

2. DFS:

Time Complexity: O(B^m), where B is the branch factor and m is the maximum depth of the tree.

Space Complexity: O(B^m),

search proceeds immediately to the deepest level of the search tree, where the nodes have no successors. The search then "backs up" to the next deepest node that still has unexpanded successors.

At each level: In DFS, at each level of the tree, the algorithm explores one branch fully before moving to the next branch. Therefore, at each level, the algorithm explores b nodes (assuming each node has b children).

Maximum depth: The maximum depth m of the tree is the longest path from the root node to any leaf node.

Total nodes: The total number of nodes explored by the DFS algorithm is the sum of nodes explored at each level. Since the algorithm explores b nodes at each level, and there are m levels in the longest path, the total number of nodes explored is bm.

Asymptotic complexity: As B and m are both positive integers, we can conclude that the time complexity of DFS **is O(B^m).**

UNIFORM COST SEARCH (UCS):

Time Complexity: O(b^(1+⌊C∗/ϵ⌋))

Space Complexity : O(b^(1+⌊C∗/ϵ⌋))

In UCS, the algorithm explores the node with the lowest path cost g(n) from the initial state to n among all nodes in the frontier. Here, g(n) is the cost of the path from the initial state to node n.

Let's denote $C*$ as the cost of the optimal solution, and ϵ as a lower bound on the cost of each action, with $\epsilon > 0$.

Analysis of Time Complexity:

The UCS algorithm explores nodes based on their path cost, and the path cost can vary from 0 (initial state) to $C*$ (optimal solution). Let's denote d as the depth of the shallowest goal node. In the worst case, UCS may explore all nodes up to the optimal solution's cost. Therefore, the number of nodes explored can be approximated by:

Number of nodes explored ≈ $O(b1+\lfloor C*/\epsilon \rfloor)$

This approximation comes from the fact that for each level, the number of nodes explored is approximately b^k, where k is the cost of the path from the initial state to that level, which can be at most ⌊C∗/ϵ⌋.