**221044025**



**Thabo Mthethwa**

**2(a) Heuristic Evaluation Function for A\* Algorithm**

* Heuristic count the number of Misplaced coins
* It Count the number of Even coins in odd positions and visa verse.
* It count the number of any coins that are left in the centre stack ‘E’

**Admissibility**

- This Heuristic is admissible; this is because it never overestimates the cost to reach the Goal

- Misplaced coin requires at least one move to reach it valid position

**Consistency**

* The heuristic is consistent, this is because:

\*)Cost from any State to goal<= Cost from successor state + Cost to successor state

**b) A comparison of the performance of the three search methods in solving problems of differing difficulty**



**1. Depth-First Search (DFS)**

**Performance:**

* 4 coins : DFS typically finds a solution, but not necessarily the optimal one. It explores many paths.
* 6 to 10 coins: DFS struggles, especially with higher coin counts, as it explores many possible states. As the problem size grows, the stack of states increases, leading to poor performance and memory exhaustion.

**2. Breadth-First Search (BFS)**

**Performance:**

* 4 coins: BFS quickly finds the optimal solution, exploring fewer states than DFS. It is more efficient for this size.
* 6 coins: BFS remains effective, but the number of explored states increases considerably.
* 8 and 10 coins: BFS becomes computationally expensive as it explores every state systematically, which leads to significant memory and time consumption.

**3. Greedy Best-First Search**

**Performance:**

* 4 coins: Greedy Best-First finds a solution quickly, often in fewer moves than BFS and DFS.
* 6 coins: The heuristic helps prune unproductive paths, making it faster than BFS but sometimes missing the optimal solution.
* 8 and 10 coins: For larger problems, the heuristic may mislead the search, causing it to explore less optimal paths, though it still performs faster than uninformed methods like BFS and DFS.

**4. A\* Search**

**Performance:**

* 4 coins: A\* performs similarly to Greedy Best-First, finding the optimal solution efficiently.
* 6 coins: A\* is effective, exploring fewer states than BFS while still ensuring optimality.
* 8 and 10 coins: While A\* is slower than Greedy Best-First in some cases, it consistently finds the best solution, making it the most reliable method for larger problems.