**Program 8**

# Write a Program to Implement Monkey Banana Problem using Python.

# https://i1.wp.com/miro.medium.com/v2/resize:fit:700/1*IjI7hyFivUg6YUJyLO4dZQ.png?ssl=1

**Description:**

The **Monkey Banana Problem** is a classic problem in Artificial Intelligence and involves a monkey trying to get a banana hanging from a ceiling. The problem can be solved using a simple form of problem-solving techniques such as search algorithms.

**Problem Description:**

* There is a monkey and a banana hanging from the ceiling.
* The monkey is on the ground and needs to reach the banana.
* The monkey has a box, which it can push or pull. By standing on the box, it can reach the banana.
* The monkey can perform actions like moving, pushing the box, and climbing the box.

To solve the problem, the goal is to design an intelligent agent (in this case, the monkey) that can figure out how to get the banana by using these actions.

**Steps to Implement:**

1. **State Representation**: Represent the state of the environment, such as the position of the monkey, the position of the box, and whether the monkey has the banana.
2. **Actions**: Define the actions that the monkey can take (e.g., move left, move right, climb, push box, etc.).
3. **Goal**: The goal is to get the banana by either climbing on the box or reaching the banana if the box is under it.
4. **Search**: A search algorithm (like BFS, DFS, or A\* algorithm) can be used to find the solution.

**Step – by -Step:**

1. **State Class**: Represents the state of the environment. It keeps track of the monkey's position, the box's position, and whether the monkey has the banana.
2. **Action Class**: Defines the possible actions the monkey can take (move left, move right, climb, push box, and get banana).
3. **MonkeyBananaProblem Class**:
   * Contains the initial state, possible actions, and transitions.
   * The search() method implements a depth-first search (DFS) to find a sequence of actions that lead to the goal (the monkey having the banana).
4. **Goal State**: The goal is reached when the monkey has the banana.
5. **Search Algorithm**: A basic DFS is used to find a solution, but you can replace it with BFS or A\* for better performance or optimality.

Let's implement the **Monkey Banana Problem** using a simple **search-based approach** in Python.

**SOURCE CODE :**

def monkey\_banana\_problem():

# Initial state

initial\_state = ('Far-Box', 'Box-Not-Under-Banana', 'Off-Box', 'Empty') # (Monkey's Location, Monkey's Position on Box, Box's Location, Monkey's Status)

print(f"\n Initial state is {initial\_state}")

goal\_state = ('Near-Box', 'Box-Under-Banana', 'On-Box', 'Holding') # The goal state when the monkey has the banana

# Possible actions and their effects

actions = {

"Move to Box": lambda state: ('Near-Box', state[1], state[2], state[3]) if state[0] != 'Near-Box' else None,

"Push Box under Banana": lambda state: ('Near-Box', 'Box-Under-Banana', state[2], state[3]) if state[0] == 'Near-Box' and state[1] != 'Box-Under-Banana' else None,

"Climb Box": lambda state: ('Near-Box', 'Box-Under-Banana', 'On-Box', state[3]) if state[0] == 'Near-Box' and state[1] == 'Box-Under-Banana' and state[2] != 'On-Box' else None,

"Grasp Banana": lambda state: ('Near-Box', 'Box-Under-Banana', 'On-Box', 'Holding') if state[0] == 'Near-Box' and state[1] == 'Box-Under-Banana' and state[2] == 'On-Box' and state[3] !='Holding' else None

}

# BFS to explore states

from collections import deque

dq = deque([(initial\_state, [])]) # Each element is (current\_state, actions\_taken)

visited = set()

while dq:

current\_state, actions\_taken = dq.popleft()

# Check if we've reached the goal

if current\_state == goal\_state:

print("\nSolution Found!")

print("Actions to achieve goal:")

for action in actions\_taken:

print(action)

print(f"Final State: {current\_state}")

return

# Mark the current state as visited

if current\_state in visited:

continue

visited.add(current\_state)

# Try all possible actions

for action\_name, action\_func in actions.items():

next\_state = action\_func(current\_state)

if next\_state and (next\_state not in visited):

dq.append((next\_state, actions\_taken + [f"Action: {action\_name}, Resulting State: {next\_state}"]))

print("No solution found.")

# Run the program

monkey\_banana\_problem()

**OUTPUT :**

Initial state is ('Far-Box', 'Box-Not-Under-Banana', 'Off-Box', 'Empty')

Solution Found!

Actions to achieve goal:

Action: Move to Box, Resulting State: ('Near-Box', 'Box-Not-Under-Banana', 'Off-Box', 'Empty')

Action: Push Box under Banana, Resulting State: ('Near-Box', 'Box-Under-Banana', 'Off-Box', 'Empty')

Action: Climb Box, Resulting State: ('Near-Box', 'Box-Under-Banana', 'On-Box', 'Empty')

Action: Grasp Banana, Resulting State: ('Near-Box', 'Box-Under-Banana', 'On-Box', 'Holding')

Final State: ('Near-Box', 'Box-Under-Banana', 'On-Box', 'Holding')