BIS LAB 4

Sneha S Bhairappa (1BM23CS333)

CUCKOO SEARCH ALGORITHM:

CODE:

```
import random

# ------------------------ Parameters ------------------------
tasks = [2, 3, 4, 5, 6]       # Each task's duration
num_tasks = len(tasks)
num_nests = 5                 # Number of nests (solutions)
Pa = 0.25                     # Probability of abandoning a nest
MaxGen = 50                   # Maximum generations



# ------------------------ Fitness Function ------------------------
def fitness(schedule):
    """Lower total duration = better fitness."""
    total_time = sum(schedule)
    return -total_time   # Negative because lower time = better fitness



# ------------------------ Generate Random Schedule ------------------------
def random_schedule():
    """Create a random schedule (random order of tasks)."""
    s = tasks[:]
    random.shuffle(s)
    return s



# ------------------------ Levy Flight (Random Neighbor) ------------------------
def levy_flight(schedule):
```

```
    """Generate new schedule by small random changes (swap two tasks)."""
    new_s = schedule[:]
    i, j = random.sample(range(num_tasks), 2)
    new_s[i], new_s[j] = new_s[j], new_s[i]
    return new_s


# ---------- Initialization ----------
nests = [random_schedule() for _ in range(num_nests)]
fitness_values = [fitness(s) for s in nests]


# ---------- Main Loop ----------
for gen in range(MaxGen):
    for i in range(num_nests):
        new_solution = levy_flight(nests[i])
        new_fitness = fitness(new_solution)

        # Randomly choose a nest to compare
        j = random.randint(0, num_nests - 1)
        if new_fitness > fitness_values[j]:
            nests[j] = new_solution
            fitness_values[j] = new_fitness

    # Sort nests by fitness (best to worst)
    sorted_nests = sorted(zip(fitness_values, nests), reverse=True)

    # Abandon a fraction of the worst nests
    num_abandon = int(Pa * num_nests)
    for k in range(num_abandon):
        sorted_nests[-(k + 1)] = (fitness(random_schedule()), random_schedule())
```

```
    # Unzip back into separate lists
    fitness_values, nests = zip(*sorted_nests)
    fitness_values, nests = list(fitness_values), list(nests)



# ------------------------ Result ------------------------
best_index = fitness_values.index(max(fitness_values))
print("Best Schedule:", nests[best_index])
print("Total Time:", -fitness_values[best_index])
```

Output:

```
Best Schedule: [6, 5, 4, 3, 2]
Total Time: 20
```