LAB 6.

1.Write a program to implement Simulated Annealing Algorithm

Code:

import mlrose\_hiive as mlrose

import numpy as np

def queens\_max(position):

no\_attack\_on\_j = 0

queen\_not\_attacking = 0

for i in range(len(position) - 1):

no\_attack\_on\_j = 0

for j in range(i + 1, len(position)):

if (position[j] != position[i]) and (position[j] != position[i] + (j - i)) and

(position[j] != position[i] - (j - i)):

no\_attack\_on\_j += 1

if (no\_attack\_on\_j == len(position) - 1 - i):

queen\_not\_attacking += 1

if (queen\_not\_attacking == 7):

queen\_not\_attacking += 1

return queen\_not\_attacking

objective = mlrose.CustomFitness(queens\_max)

problem = mlrose.DiscreteOpt(length=8, fitness\_fn=objective, maximize=True,

max\_val=8)

T = mlrose.ExpDecay()

initial\_position = np.array([4, 6, 1, 5, 2, 0, 3, 7])

best\_position, best\_objective,fitness\_curve=

mlrose.simulated\_annealing(problem=problem, schedule=T,

max\_attempts=500,init\_state=initial\_position)

print('The best position found is:', best\_position)

print('The number of queens that are not attacking each other is:',

best\_objective)

Output:

2.Tower Of Hanoi

Code:

import mlrose\_hiive as mlrose

import numpy as np

def hanoi\_fitness(state):

correct\_disks = 0

destination\_peg = 2

for i in range(len(state)):

if state[i] == destination\_peg:

correct\_disks += 1

else:

break

return correct\_disks

fitness\_fn = mlrose.CustomFitness(hanoi\_fitness)

problem = mlrose.DiscreteOpt(length=3, fitness\_fn=fitness\_fn, maximize=True,

max\_val=3)

schedule = mlrose.ExpDecay()

initial\_state = np.array([0, 0, 0])

best\_state, best\_fitness, fitness\_curve = mlrose.simulated\_annealing(problem,

schedule=schedule, max\_attempts=1000, init\_state=initial\_state)

print("Best state (final configuration):", best\_state)

print("Number of correct disks on destination peg:", best\_fitness)

def print\_hanoi\_solution(state):

print("\nTower of Hanoi Configuration:")

pegs = {0: [], 1: [], 2: []}

for disk, peg in enumerate(state):

pegs[peg].append(disk)

for peg in pegs:

print(f"Peg {peg}: {pegs[peg]}")

print\_hanoi\_solution(best\_state)