22-10-2024 WEEK - 05

Implementing Hill climbing search algorithm to solve N-Queens Problem.

ALGORITHM / PSUEDOCODE -

ALGORITHM / FSUEDOCODE -	classnate
	Data 22110124
WEEK-4	Page 13
solve N-Dugger	nbing search algorithm to
solve N-Queens probl	em,
ALGORITHM PECEDOC	
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	DDE :
step 1: Initialization	
	by representing position
of the 4 queen.	- 09 representing post non
SKP2: affine hinch	
define cost carrie	Lation and neighbour
state functions	and ray resour
Step 3: Hill climbing	g function
initialize current s	rate to initial state and
calculate cost	The national area
step 4: I terative se	arch
	and calculate cost
o of best relambours	has lower-cost man cumuntstate
update ausient wa	He to best neighbour and
cost	or 10 ber heighborn and
5	
step 5: execution	
call hill dingting	function with intial state
and print equet.	
	A sub-track of the sub-

CODE-

import random

```
def calculate cost(state):
  """Calculate the cost for a given state, which is the number of pairs of queens attacking each
other."""
  attacking pairs = 0
  n = len(state)
  for i in range(n):
     for j in range(i + 1, n):
       if state[i] == state[j] or abs(state[i] - state[j]) == abs(i - j):
          attacking pairs += 1
  return attacking pairs
def get neighbors(state):
  """Generate all possible neighboring states by swapping two queens' row positions."""
  neighbors = []
  n = len(state)
  for i in range(n):
     for j in range(i + 1, n):
       neighbor = state[:]
       neighbor[i], neighbor[j] = neighbor[j], neighbor[i]
       neighbors.append(neighbor)
  return neighbors
def hill climbing(initial state):
  """Perform the hill climbing search algorithm and store the state space tree."""
  current state = initial state
  current cost = calculate cost(current state)
  state tree = {tuple(current state): current cost} # Dictionary to store state and cost
  step = 0
  while True:
     print(f"\nStep {step}:")
     neighbors = get neighbors(current state)
     print("Neighbors:")
     for neighbor in neighbors:
       cost = calculate cost(neighbor)
       print(f" {neighbor}: Cost = {cost}")
```

```
state tree[tuple(neighbor)] = cost # Store neighbor state and cost
     best neighbor = None
     best cost = current cost
     for neighbor in neighbors:
       cost = calculate cost(neighbor)
       if cost < best cost:
          best cost = cost
          best neighbor = neighbor
     if best cost >= current cost:
       # No better neighbor found, return current state
       print(f"\nNo better neighbor found. Final state reached.")
       return current_state, current_cost, state_tree
     current state = best neighbor
     current cost = best cost
     step += 1
initial state = [3, 1, 2, 0]
final state, final cost, state space tree = hill climbing(initial state)
print("\nInitial state:", initial state)
print("Final state:", final state)
print("Final cost (attacking pairs):", final cost)
#print("\nState Space Tree (State: Cost):")
#for state, cost in state space tree.items():
# print(f"{list(state)}: {cost}")
```

OUTPUT (Snapshot)-

```
Step 0:
Neighbors:
  [1, 3, 2, 0]: Cost = 1
  [2, 1, 3, 0]: Cost = 1
  [0, 1, 2, 3]: Cost = 6
 [3, 2, 1, 0]: Cost = 6
 [3, 0, 2, 1]: Cost = 1
  [3, 1, 0, 2]: Cost = 1
Step 1:
Neighbors:
 [3, 1, 2, 0]: Cost = 2
  [2, 3, 1, 0]: Cost = 2
  [0, 3, 2, 1]: Cost = 4
  [1, 2, 3, 0]: Cost = 4
  [1, 0, 2, 3]: Cost = 2
  [1, 3, 0, 2]: Cost = 0
Step 2:
Neighbors:
  [3, 1, 0, 2]: Cost = 1
 [0, 3, 1, 2]: Cost = 1
 [2, 3, 0, 1]: Cost = 4
  [1, 0, 3, 2]: Cost = 4
  [1, 2, 0, 3]: Cost = 1
  [1, 3, 2, 0]: Cost = 1
No better neighbor found. Final state reached.
```

```
Initial state: [3, 1, 2, 0]
Final state: [1, 3, 0, 2]
Final cost (attacking pairs): 0
```

STATE SPACE TREE (Snapshot) -

STATE SPACE TREE 3 1' 2 Canquin prinsipani avios Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q	
R 2002003027 I MATIGODIA	
R 300000000 PMHT19000A	
R 300000000 PMHT19001A	
Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q Q	
a cananadat (as	
Proposed in the proposed of the proposed of	
	-
1320 2130 3021 310	21
	134
	Q
	1×
0 0 0 0 0	
cost=1 0 1 2 3 cost=6 cost=18 2 1 0 cost	2 1
Q In MANAGER DE Q E SAN	
1) see Querra de descripto Querran	456
calculated to a day of the same	13
	1
COSH=6 NOCON SICOSH=6'	
con production and amountain promise	1
and the same and the same and	
of pert of shooting and thought a stag to	9-4