Date 29/10/24
Page 21

29/10

Hill Climbing Algorithm for N-Queens

Algorithm

i) Initialize a variable N (no of queens)

ii) Consider N×N square board with initialgenerate(), where each queen is placed randomly.

state[i] = j → i^{th} queen (column) is placed in j^{th} row.

iii) cal.attack()
→ The heuristic function h(n) calculates the attacking (collision) from all the 8 directions (initially, attacking = 0)

iv) objective()

generate new state(), based on new neighbour optimize the best state, until attacking = 0

```
func h(state):
    h = 0
    for i in range(len(state)
        for j in range(i+1, len state):
            if abs(state[i] - state[j] ==
                                abs(i - j)
            or state[i] == state[j]
            h += 1
    return h
```

Date __/__/__
Page _____

```
func generatenew():
    best = state
    for i in range(len(state)):
        for j in range(8):
            new = state[:i] + j + state[i+1:]
            if h(new) < h(best):
                best = new
    return best

func hillclimb():
    int cur = random.randint(0,8).
    c_h = h(cur)
    cur = generatenew(cur)
    if h(cur) == 0:
        return cur
    if h(cur) >= c_h:
        cur = randomrestart(1,8).
```

A* search algorithm for N queens

```
func h(state):
    h = 0
    for i in range(len(state)):
        for j in range(i+1, len(state)):
            if abs(state[i] - state[j])
                == abs(i-j):
                state[i] == state[j]
                h++
    return h
```

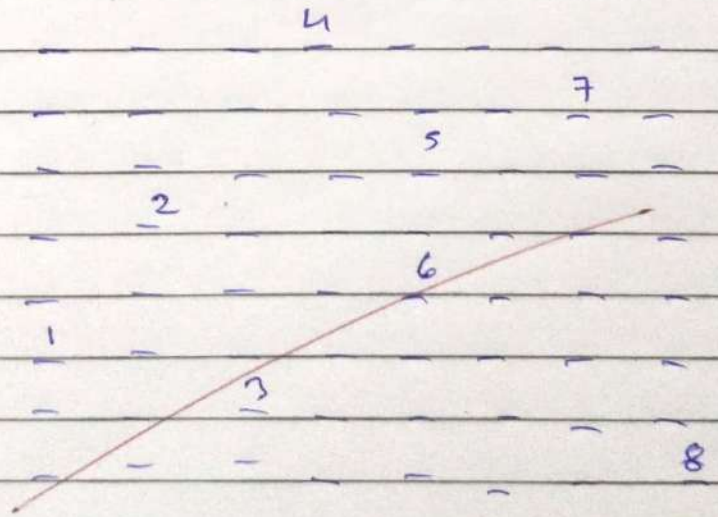
Date __/__/__
Page _____

```
func a star():
    initial []
    h = [], g = 8
    heap.push(h, (heuristic(initial) + g, white h: initial))
    c = heap.pop()
    if h(c[1]) + c[2] == 0:
        return c
    if len(c[1]) == 8:
        continue
    for i in range(1, 9):
        n = c[1] + [i]
        heap.push(q, h(n) + (c[2] - 1, n, g - 1))
```

proceed

Output:

[4 7 5 2 6 1 3 8]



29/10/24