AI

Assignment 2

Using Local Search and Constraint Satisfaction Algorithms to Solve 8-Queens

Ashraqat Sheta (13)
Fatma Ibrahim (42)
Marina Zakaria (47)

1. Hill Climbing Algorithm

```
function HILL-CLIMBING( problem) returns a state that is a local maximum
   inputs: problem, a problem
   local variables: current, a node
                    neighbor, a node
   current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
      neighbor ← a highest-valued successor of current
      if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
      current ← neighbor
   end
```

- Simple, general idea:
  - Start wherever
  - Repeat: move to the best neighboring state
  - If no neighbors better than current, quit

Random-restart hill climbing overcomes local maxima—trivially complete

hillCliming()

- Read initial state from file
- Make sure that each col has only one queen if not then move queens diagonally until reach that state
- Start move queens vertically and calculate heuristic function in each state whenever get lower heuristic update current state with that state
- If got stuck then start from randomly state

Data Structures used

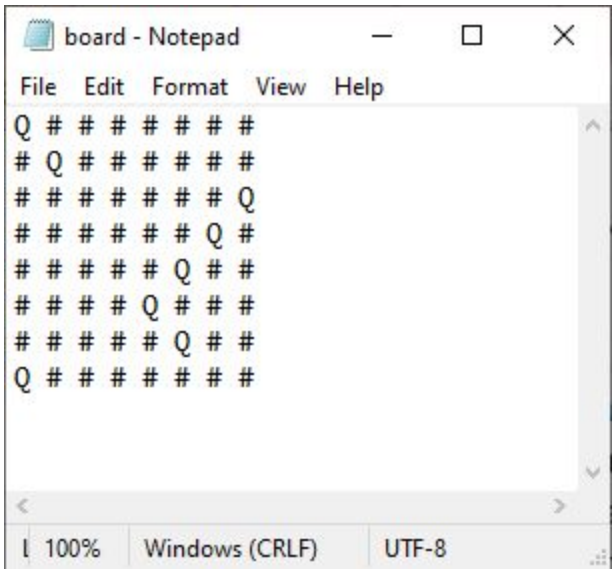
- 2d char array
- Arraylist

Assumptions:

- Max restart 1000 iterations

Sample run:

- Input :



- Output

```
Best board found this iteration:
# # # # # Q # #
# Q # # # # # #
# # # # Q # # #
Q # # # # # # #
# # # # # # Q #
# # # Q # # # #
# # # # # # # Q
# # Q # # # # #
h = 0
# pairs of queens attacking each other: 0

h = 0
Done in 18 restarts.
Execution time is 0.24100 seconds
Cost = 1216
Number of expanded Nodes = 152
```

(a) Pseudo code for each algorithm mentioned.

Beam Search algorithm(k, Node start)

1) Create an empty PriorityQueue

PriorityQueue pq;

2) Insert "start" in pq.

pq.insert(start)

3) Until PriorityQueue is empty

u = PriorityQueue.DeleteMin

If u is the goal

Exit

Else

Foreach neighbor v of u

If v "Unvisited"

Mark v "Visited"

pq.insert(v)

if(K<sizeof(pq))

delete last element in priority queue

Mark u "Examined"

End procedure

(b) Data Structures used in each algorithm.

Priority Queue-LinkedList

(c) Assumptions made for each algorithm.

Max Iteration to fail =1000

start from the initial board.

move one queen horizontally, vertically or diagonally in one step

given k (beam width) as input

(d) One Sample run showing the output to the sample input file given for each algorithm. You should show the requirements for each algorithm printed to the console.

Best K=15 ooooh!!

K=1

```
h = 0
the beam search algorithm reach to the goal !!
Execution time is 932.0 seconds
Final Board:
 # # # # # Q # #
 # Q # # # # # #
 # # # # # # Q #
 Q # # # # # # #
 # # # Q # # # #
 # # # # # # # Q
 # # # # Q # # #
 # # Q # # # # #
The cost to the final board: 241
Number of expanded nodes: 241
```

K=5

```
h = 0
the beam search algorithm reach to the goal !!
Execution time is 328.0 seconds
Final Board:
 # # Q # # # # #
 # # # Q # # # #
 # # # # # # # Q
 # # # # Q # # #
 # Q # # # # # #
 # # # # # Q # #
 Q # # # # # # #
 # # # # # # Q #
The cost to the final board: 107
Number of expanded nodes: 193
```

K=100 with change position

```
h = 0
the beam search algorithm reach to the goal !!
Execution time is 61.0 seconds
Final Board:
 # # Q # # # # #
 # # # # # Q # #
 # # # Q # # # #
 Q # # # # # # #
 # # # # Q # # #
 # # # # # # # Q
 # Q # # # # # #
 # # # # # # Q #
The cost to the final board: 45
Number of expanded nodes: 120
```

K=100

```
# # # # # # # #
the beam search algorithm reach to the goal !!
Execution time is 133.0 seconds
Final Board:
 # # # Q # # # #
 # Q # # # # # #
 # # # # # # Q #
 # # # # Q # # #
 Q # # # # # # #
 # # # # # # # Q
 # # # # # Q # #
 # # Q # # # # #
The cost to the final board: 42
Number of expanded nodes: 147
```

K=1000

```
h = 0
the beam search algorithm reach to the goal !!
Execution time is 175.0 seconds
Final Board:
 # # Q # # # # #
 # # # Q # # # #
 # # # # Q # # #
 Q # # # # # # #
 # # # # # # # Q
 # # # # # Q # #
 # Q # # # # # #
 # # # # # # Q #
The cost to the final board: 40
Number of expanded nodes: 168
```

K=64

```
h = 0
the beam search algorithm reach to the goal !!
Execution time is 62.0 seconds
Final Board:
 # # Q # # # # #
 # # # # # Q # #
 # # # Q # # # #
 Q # # # # # # #
 # # # # Q # # #
 # # # # # # # Q
 # Q # # # # # #
 # # # # # # Q #
The cost to the final board: 45
Number of expanded nodes: 120
```

K=40

```
h = 0
the beam search algorithm reach to the goal !!
Execution time is 49.0 seconds
Final Board:
 # # Q # # # # #
 # # # # # Q # #
 # # # Q # # # #
 Q # # # # # # #
 # # # # Q # # #
 # # # # # # # Q
 # Q # # # # # #
 # # # # # # Q #
The cost to the final board: 45
Number of expanded nodes: 109
```

K=30

```
h = 0
the beam search algorithm reach to the goal !!
Execution time is 34.0 seconds
Final Board:
 # # Q # # # # #
 # # # # # Q # #
 # # # Q # # # #
 Q # # # # # # #
 # # # # Q # # #
 # # # # # # # Q
 # Q # # # # # #
 # # # # # # Q #
The cost to the final board: 45
Number of expanded nodes: 97
```

K=20

```
h = 0
the beam search algorithm reach to the goal !!
Execution time is 33.0 seconds
Final Board:
 # # Q # # # # #
 # # # # # Q # #
 # # # Q # # # #
 Q # # # # # # #
 # # # # Q # # #
 # # # # # # # Q
 # Q # # # # # #
 # # # # # # Q #
The cost to the final board: 45
Number of expanded nodes: 97
```

K=15

```
h = 0
the beam search algorithm reach to the goal !!
Execution time is 17.0 seconds
Final Board:
 # # Q # # # # #
 # # # Q # # # #
 # # # # Q # # #
 Q # # # # # # #
 # # # # # Q # #
 # # # # # # # Q
 # Q # # # # # #
 # # # # # # Q #
The cost to the final board: 34
Number of expanded nodes: 82
```

K=14

```
h = 0
the beam search algorithm reach to the goal !!
Execution time is 52.0 seconds
Final Board:
 # # Q # # # # #
 # # # Q # # # #
 # # # # Q # # #
 Q # # # # # # #
 # # # # # Q # #
 # # # # # # # Q
 # Q # # # # # #
 # # # # # # Q #
The cost to the final board: 51
Number of expanded nodes: 110
```

K=10

```
h = 0
the beam search algorithm reach to the goal !!
Execution time is 52.0 seconds
Final Board:
 # # Q # # # # #
 # # # Q # # # #
 # # # # Q # # #
 Q # # # # # # #
 # # # # # Q # #
 # # # # # # # Q
 # Q # # # # # #
 # # # # # # Q #
The cost to the final board: 51
Number of expanded nodes: 110
```

(e) How to run your code from terminal and any required dependencies

```
ıblic class Test {

   public static void main(String[] args) {
       BeamSearch b=new BeamSearch();
       b.RunAlgorithm(1);

   }
```

The k=1 as an input

Genetic Algorithm

(a) Pseudo code for each algorithm mentioned.

```
1)generate random population

2)while(not solved)

        Calculate fitness

        Select parents

        Do cross over

        Do mutation

        Get next generation
```

(b) Data Structures used in each algorithm.

Arraylist-2d array character

(c) Assumptions made for each algorithm.

none

(d) One Sample run showing the output to the sample input file given for each algorithm. You should show the requirements for each algorithm printed to the console.

(e) How to run your code from terminal and any required dependencies.

Constraint Satisfaction Problem (CSP)

(a) Pseudo code for each algorithm mentioned.

```
eleminate row attacks

2) eleminate col attacks

3) while not solved

      Backtrack

      Check constraints
```

(b) Data Structures used in each algorithm.

Arraylist-2d array character

(c) Assumptions made for each algorithm.

none

(d) One Sample run showing the output to the sample input file given for each algorithm. You should show the requirements for each algorithm printed to the console.

(e) How to run your code from terminal and any required dependencies.