CSP: N-Queens

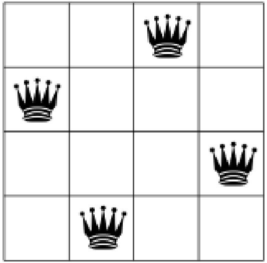
Team 3

AI Class – Project 2

February 2026

Problem Description

- Ensure exactly one queen per row and column.
- Avoid conflicts along diagonals.
- Apply MRV and LCV heuristics for efficient search.
- Use AC-3 algorithm for constraint propagation to reduce possibilities and speed up solving.



Algorithms

CSP Backtracking

For smaller boards (n <= 50)

Minimum Remaining Values (MRV)

Selects the row with the fewest remaining valid columns.

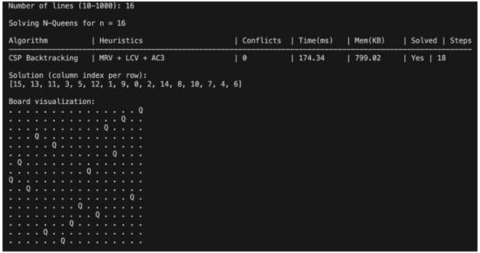
Least Constraining Value (LCV)

Orders columns that minimize conflicts with other queens.

AC-3 (Arc Consistency)

Propagates constraints to reduce domains before trying values.

N = 11



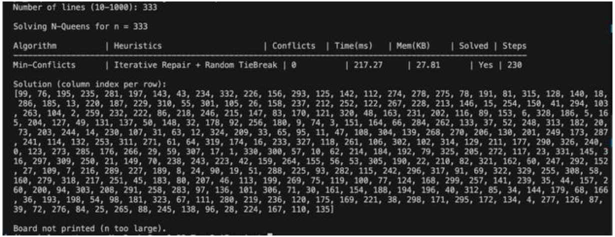
```
Number of lines (10-1000): 16

Solving N-Queens for n = 16

Algorithm          | Heuristics                    | Conflicts  | Time(ms)   | Mem(KB)    | Solved | Steps
---------------------------------------------------------------------------------------------------------
CSP Backtracking   | MRV + LCV + AC3               | 0          | 174.34     | 799.02     | Yes | 18

Solution (column index per row):
[15, 13, 11, 3, 5, 12, 1, 9, 0, 2, 14, 8, 10, 7, 4, 6]

Board visualization:
. . . . . . . . . . . . . . . Q
. . . . . . . . . . . . . Q . .
. . . . . . . . . . . Q . . . .
. . . Q . . . . . . . . . . . .
. . . . . Q . . . . . . . . . .
. . . . . . . . . . . . Q . . .
. Q . . . . . . . . . . . . . .
. . . . . . . . . Q . . . . . .
Q . . . . . . . . . . . . . . .
. . Q . . . . . . . . . . . . .
. . . . . . . . . . . . . . Q .
. . . . . . . . Q . . . . . . .
. . . . . . . . . . Q . . . . .
. . . . . . . Q . . . . . . . .
. . . . Q . . . . . . . . . . .
. . . . . . Q . . . . . . . . .
```

Algorithms

Min-Conflicts (Iterative Repair)

- For larger boards (n > 50)
  - Start with a random assignment of queens.
  - Identify rows where queens are in conflict.
  - Move conflicted queens to positions that minimize conflicts.
  - Repeat up to a maximum number of steps

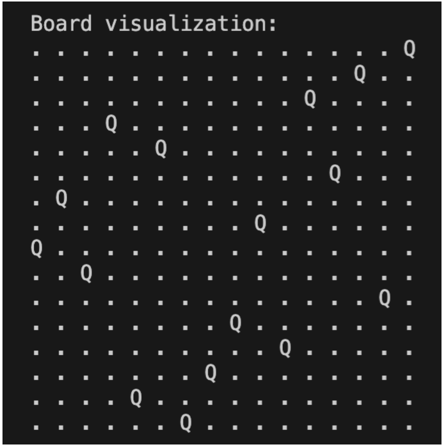
N = 333



```
Number of lines (10-1000): 333

Solving N-Queens for n = 333

Algorithm          | Heuristics                      | Conflicts  | Time(ms)   | Mem(KB)    | Solved | Steps
------------------------------------------------------------------------------------------------------------
Min-Conflicts      | Iterative Repair + Random TieBreak | 0          | 217.27     | 27.81      | Yes | 230

Solution (column index per row):
[99, 76, 195, 235, 281, 197, 143, 43, 234, 332, 226, 156, 293, 125, 142, 112, 274, 278, 275, 78, 191, 81, 315, 128, 140, 18,
 286, 185, 13, 220, 187, 229, 310, 55, 301, 105, 26, 158, 237, 212, 252, 122, 267, 228, 213, 146, 15, 254, 150, 41, 294, 103
, 263, 104, 2, 259, 232, 222, 86, 218, 246, 215, 147, 83, 170, 121, 320, 48, 163, 231, 202, 116, 89, 153, 6, 328, 186, 5, 16
5, 204, 127, 49, 131, 137, 50, 148, 32, 178, 92, 256, 180, 9, 74, 3, 151, 164, 66, 284, 262, 133, 37, 52, 248, 313, 182, 20,
 73, 203, 244, 14, 230, 107, 31, 63, 12, 324, 209, 33, 65, 95, 11, 47, 108, 304, 139, 268, 270, 206, 130, 201, 249, 173, 287
, 241, 114, 132, 253, 311, 271, 61, 64, 319, 174, 16, 233, 327, 118, 261, 106, 302, 102, 314, 129, 211, 177, 290, 326, 240,
0, 123, 273, 285, 176, 266, 29, 59, 307, 17, 1, 330, 300, 57, 10, 62, 214, 184, 192, 79, 325, 205, 272, 117, 23, 331, 145, 3
16, 297, 309, 250, 21, 149, 70, 238, 243, 223, 42, 159, 264, 155, 56, 53, 305, 190, 22, 210, 82, 321, 162, 60, 247, 292, 152
, 27, 109, 7, 216, 289, 227, 189, 8, 24, 90, 19, 51, 288, 225, 93, 282, 115, 242, 296, 317, 91, 69, 322, 329, 255, 308, 58,
160, 279, 318, 217, 251, 45, 183, 80, 207, 46, 113, 199, 269, 75, 119, 100, 77, 124, 168, 299, 257, 141, 239, 35, 44, 157, 2
60, 200, 94, 303, 208, 291, 258, 283, 97, 136, 101, 306, 71, 30, 161, 154, 188, 194, 196, 40, 312, 85, 34, 144, 179, 68, 166
, 36, 193, 198, 54, 98, 181, 323, 67, 111, 280, 219, 236, 120, 175, 169, 221, 38, 298, 171, 295, 172, 134, 4, 277, 126, 87,
39, 72, 276, 84, 25, 265, 88, 245, 138, 96, 28, 224, 167, 110, 135]

Board not printed (n too large).
```

Implementation Notes

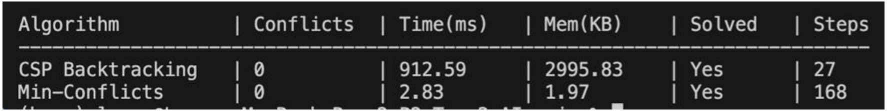
- Initial Board Input
- Conflict Counting
- Memory & Performance Tracking
- Board Visualization



```
Board visualization:
. . . . . . . . . . . . . . . Q
. . . . . . . . . . . . . Q . .
. . . . . . . . . . . Q . . . .
. . . Q . . . . . . . . . . . .
. . . . . Q . . . . . . . . . .
. . . . . . . . . . . . Q . . .
. Q . . . . . . . . . . . . . .
. . . . . . . . . Q . . . . . .
Q . . . . . . . . . . . . . . .
. . Q . . . . . . . . . . . . .
. . . . . . . . . . . . . . Q .
. . . . . . . . Q . . . . . . .
. . . . . . . . . . Q . . . . .
. . . . . . . Q . . . . . . . .
. . . . Q . . . . . . . . . . .
. . . . . . Q . . . . . . . . .
```

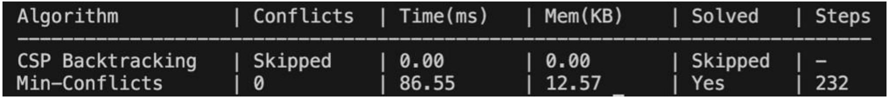
Testing and Validation

Example result table for n = 23:



```
Algorithm           | Conflicts   | Time(ms)    | Mem(KB)     | Solved     | Steps
------------------------------------------------------------------------------------
CSP Backtracking    | 0           | 912.59      | 2995.83     | Yes        | 27
Min-Conflicts       | 0           | 2.83        | 1.97        | Yes        | 168
```

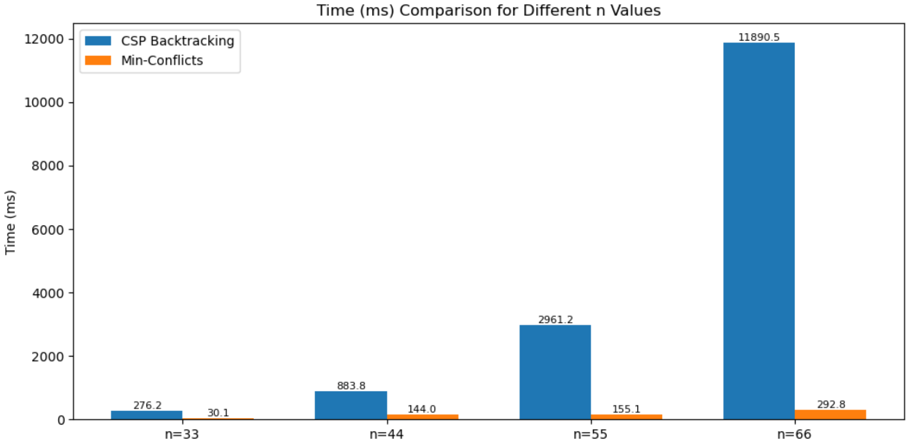
Example result table for n = 200:



```
Algorithm           | Conflicts   | Time(ms)    | Mem(KB)     | Solved     | Steps
------------------------------------------------------------------------------------
CSP Backtracking    | Skipped     | 0.00        | 0.00        | Skipped    | -
Min-Conflicts       | 0           | 86.55       | 12.57       | Yes        | 232
```

Testing and Validation

4 different test cases in one graph



Conclusion: Observations and Notes

CSP Backtracking is slower on large boards due to exponential growth in possibilities.

Min-Conflicts performs well even for n = 1000 but may not guarantee a solution if max steps are reached.

Memory usage scales linearly with n for both algorithms.

AC-3 significantly reduces the search space for small boards.

Thank You!