1. The Big-Oh runtime for the brute force n-queens algorithm is O(n!)
   1. The brute force algorithm takes 1.374 seconds when n=8
   2. The backtracking algorithm only takes .0003 seconds when n=8
   3. This makes the backtracking solution approximately 458 times faster for n=8
2. Backtracking is not always faster than brute force. When n is small, they take the same amount of time. However, as n grows the backtracking method is significantly faster. Here are the running times:

Bruteforce (2)

Running time: 0.0 ms

Backtracking (2)

Running time: 0.0 ms

Bruteforce (3)

Running time: 0.0 ms

Backtracking (3)

Running time: 0.0 ms

Bruteforce (4)

Running time: 0.0 ms

Backtracking (4)

Running time: 0.0 ms

Bruteforce (5)

Running time: 0.0 ms

Backtracking (5)

Running time: 0.0 ms

Bruteforce (6)

Running time: 13.999700546264648 ms

Backtracking (6)

Running time: 0.9868144989013672 ms

Bruteforce (7)

Running time: 39.99066352844238 ms

Backtracking (7)

Running time: 0.0 ms

Bruteforce (8)

Running time: 1383.9564323425293 ms

Backtracking (8)

Running time: 2.997875213623047 ms

1. I will show that a square at position (i, j) is diagonal to a square at (x, y) if an only if

or I will use the following definition of a diagonal square. Two squares (i, j) and (x, y) are diagonal if one of the following cases is true:

and

and

and

and

I will show that each definition resolves to either or

1. For the first definition I will subtract the second formula from the first
2. For the second definition I will add both formulas together
3. For the third equation I will add both formulas together
4. For the fourth and final definition I will subtract the second formula from the first
5. The Big-Oh run time for the fast\_fib or iterative Fibonacci algorithm is O(n).
   1. The fastest of the three Fibonacci algorithms implored is by far the matrix algorithm when referring to Big-Oh run times.
      1. Matrix Fibonacci n = 4 Running time: 0.0 ms
      2. Fast Fibonacci n = 4 Running time: 0.0 ms
      3. Recursive Fibonacci n = 4 Running time: 0.0 ms
      4. Matrix Fibonacci n = 32 Running time: 0.0 ms
      5. Fast Fibonacci = 32 Running time: 0.0 ms
      6. Recursive Fibonacci n = 32 Running time: 416.5472984313965 ms
      7. Matrix Fibonacci n = 65536 Running time: 7.00068473815918 ms
      8. Fast Fibonacci n = 65536 Running time: 51.44095420837402 ms
      9. Recursive Fibonacci n = 65536 Running time: N/A
   2. All algorithms ran at the same speed when n = 4
   3. At n = 32 the Matrix and Fast Fibonacci algorithms were the fastest
   4. At n = 65536 the Matrix Fibonacci algorithm was the fastest
   5. The fastest Big-Oh runtime was the Matrix Fibonacci algorithm with a runtime of O(log n). Big-Oh represents the worst scenario for each algorithm relative to the size of the input. The second fastest was the Fast Fibonacci algorithm with a runtime of O(n). The slowest algorithm was the Recursive Fibonacci algorithm with a runtime of O(2^n) which is on the slower side of Big-Oh runtimes.