Project 2

Cristian Gomez, Enhui Lin, Alexis Martinez

Department of Computer Science - California State University, Fullerton

Spring 2024 CPSC 335 – 08, 11 - Algorithm Engineering

Kavil Jain

December 14, 2024

1. Graphs:

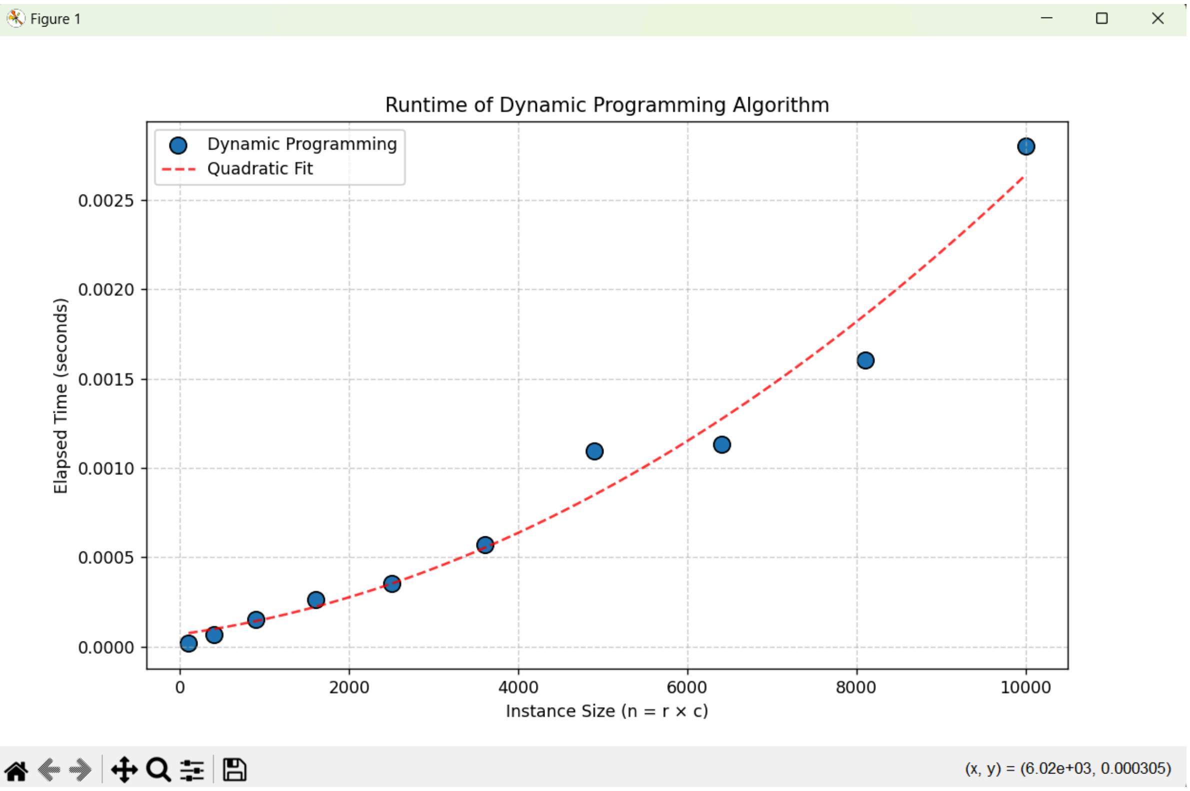
Figure 1
Runtime of Dynamic Programming Algorithm
Dynamic Programming
Quadratic Fit
Elapsed Time (seconds)
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
2000
4000
6000
8000
10000
Instance Size (n = r × c)
(x, y) = (6.02e+03, 0.000305)

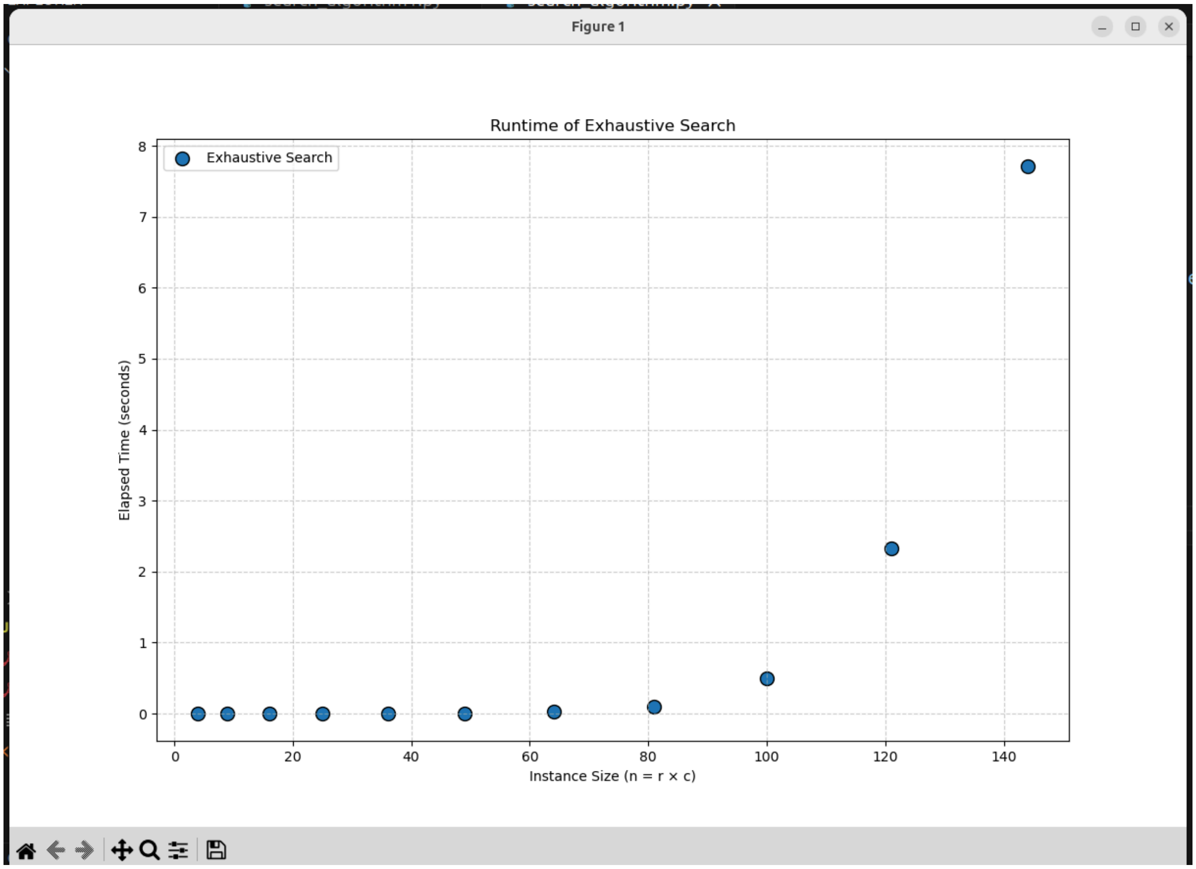
Figure 1
Runtime of Exhaustive Search
Exhaustive Search
Elapsed Time (seconds)
8
7
6
5
4
3
2
1
0
0
20
40
60
80
100
120
140
Instance Size (n = r × c)

2. Empirical analysis:
The data we gathered is strongly consistent with the predicted big-O efficiency classes for both algorithms:

Dynamic Programming (predicted $O(n^2)$):

The graph shows a clear quadratic growth pattern and the red quadratic fit line closely matches the actual data points. The Growth rate increases gradually and smoothly and it can handles large inputs (up to 320×320 grids which we tested) efficiently. The curved shape of the line exactly matches what we expect from quadratic growth

Exhaustive Search (predicted O(n·2^n)):

The graph for Exhaustive Search shows dramatic exponential growth exactly as predicted. Runtime increases explosively with larger inputs and even small increases in grid size cause large jumps in runtime. This method quickly becomes impractical around 12×12 grids and the steep vertical growth at the end is characteristic of exponential algorithms

The empirical evidence strongly supports the theoretical predictions because:

- Each algorithm's runtime behavior matches its predicted complexity class
- The relative performance difference between the algorithms aligns with theoretical expectations
- The shapes of both graphs reflect their respective mathematical growth functions
- The scale difference between the two algorithms (seconds vs milliseconds) is consistent with the huge efficiency gap we expect between exponential and polynomial algorithms

Overall, these results provide clear experimental validation of the theoretical big-O classifications.

3. Conclusion:

I. Is there a noticeable difference in the performance of the two algorithms?
    A. Yes, there is a significant difference in performance between the two algorithms, with dynamic programming operating much faster (in milliseconds) compared to exhaustive search which quickly grows out of hand for the algorithm to handle as shown by our graph and analysis.

II. II. According to your experimental observation, which of the implementations is faster, and by how much?
    A. The dynamic programming implementation is significantly faster, processing a 100×100 grid (n=10,000) in about 0.0028 seconds, while the exhaustive search takes 7.7 seconds just to process a 12×12 grid (n=144); this means dynamic

programming is thousands of times faster when handling larger grids and the performance difference becomes even more pronounced as grid sizes increase.

III. Are your empirical analyses consistent with the predicted big-O efficiency class for each algorithm? Justify your answer.

   A. Yes, the empirical analyses are entirely consistent with the predicted efficiency classes: the dynamic programming algorithm shows clear quadratic growth $O(n^2)$ with a smooth curved line fitting the data points, while the exhaustive search demonstrates dramatic exponential growth $O(n \cdot 2^\wedge n)$ with runtime increasing explosively for larger inputs, exactly matching their theoretical complexity classes.

IV. Is this evidence consistent or inconsistent with hypothesis 1? Justify your answer.

   A. The evidence is consistent with hypothesis 1 ("Exhaustive search algorithms can be implemented, and produce correct outputs") because both algorithms were successfully able to produce correct results for the opponent avoidance problem, with both approaches finding valid paths through the grid while avoiding obstacles.

V. Is this evidence consistent or inconsistent with hypothesis 2? Justify your answer.

   A. The evidence strongly supports hypothesis 2 ("Algorithms with exponential or factorial running times are extremely slow, probably too slow to be practical use") because the exhaustive search algorithm becomes impractically slow even for modest grid sizes, taking over 7 seconds for a 12×12 grid while being unable to handle larger grids in reasonable time, whereas the polynomial-time dynamic programming solution remains efficient even for much larger grids (up to 100x100 grids or more).