Tommy Vu

[tommyuv23@csu.fullerton.edu](mailto:tommyuv23@csu.fullerton.edu)

Project 4 - CPSC 335-01

Professor Bein

# **The Hypothesis**

* This experiment will test the following hypothesis:
* Polynomial-time dynamic programming algorithms are more efficient than exponential-time exhaustive search algorithms that solve the same problem.

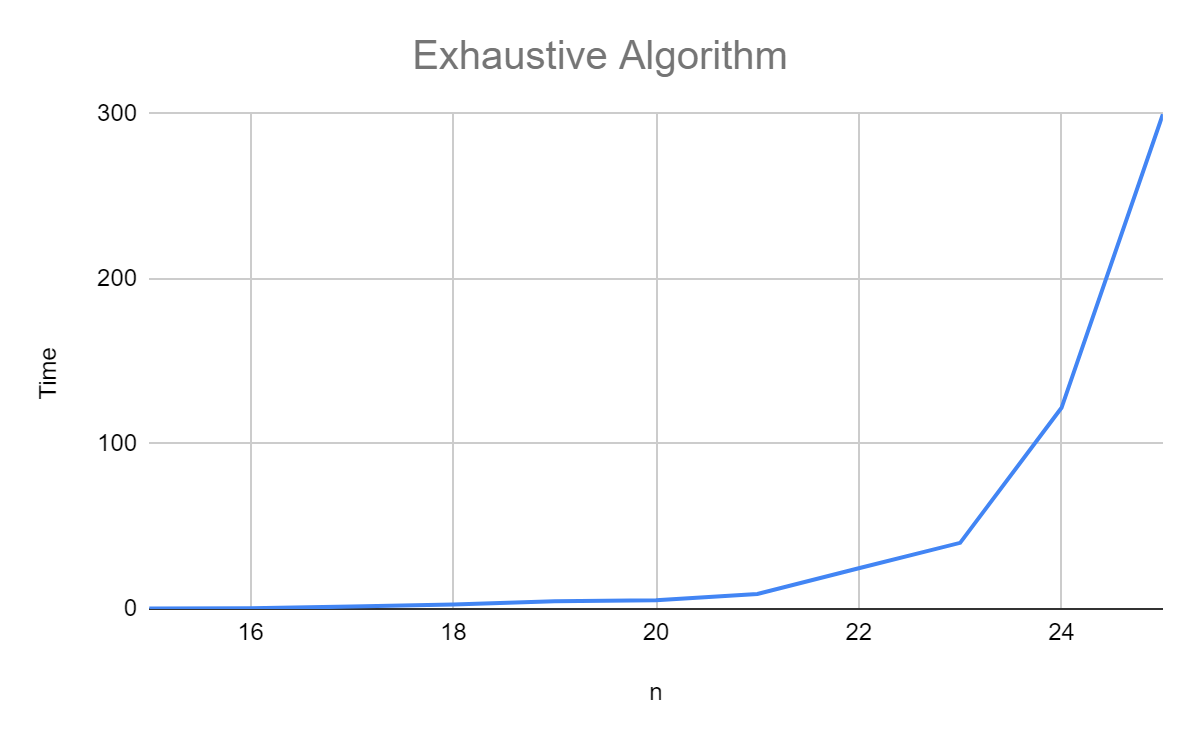
**Exhaustive Algorithm Code**

| path candidate(setting); // 1  bool valid = true; // 1  // add to candidate a path not exceedings <steps> binary values  for (size\_t k = 0; k < steps; k++) { // n - 1 - 0 + 1  int bit = (bits >> k) & 1; // 3  if (bit == 1) { // 1 time unit  if (candidate.is\_step\_valid(STEP\_DIRECTION\_EAST)) { // 1  candidate.add\_step(STEP\_DIRECTION\_EAST); // 1  }  else {  valid = false; // 1  break; // 1  }  }  else {  if (candidate.is\_step\_valid(STEP\_DIRECTION\_SOUTH)) { //1  candidate.add\_step(STEP\_DIRECTION\_SOUTH); // 1  }  else {  valid = false; // 1  break; // 1  }  }  }  if (valid && (candidate.total\_cranes() > best.total\_cranes())) { //2  best = candidate; // 1  }  } |
| --- |

**Exhaustive Mathematical Analysis**

| O(1 + 1 + 2n (1 + n(c + 1 + 1) + 1 + 1 + 1) + 1)  O(3 + 2n (4 + n(c + 2))  O(3 + 2n (4 + nc + 2n))  O(3 + 8n + 2nnc + 4nn)  = O(2n \* n) |
| --- |

**Scatterplot**



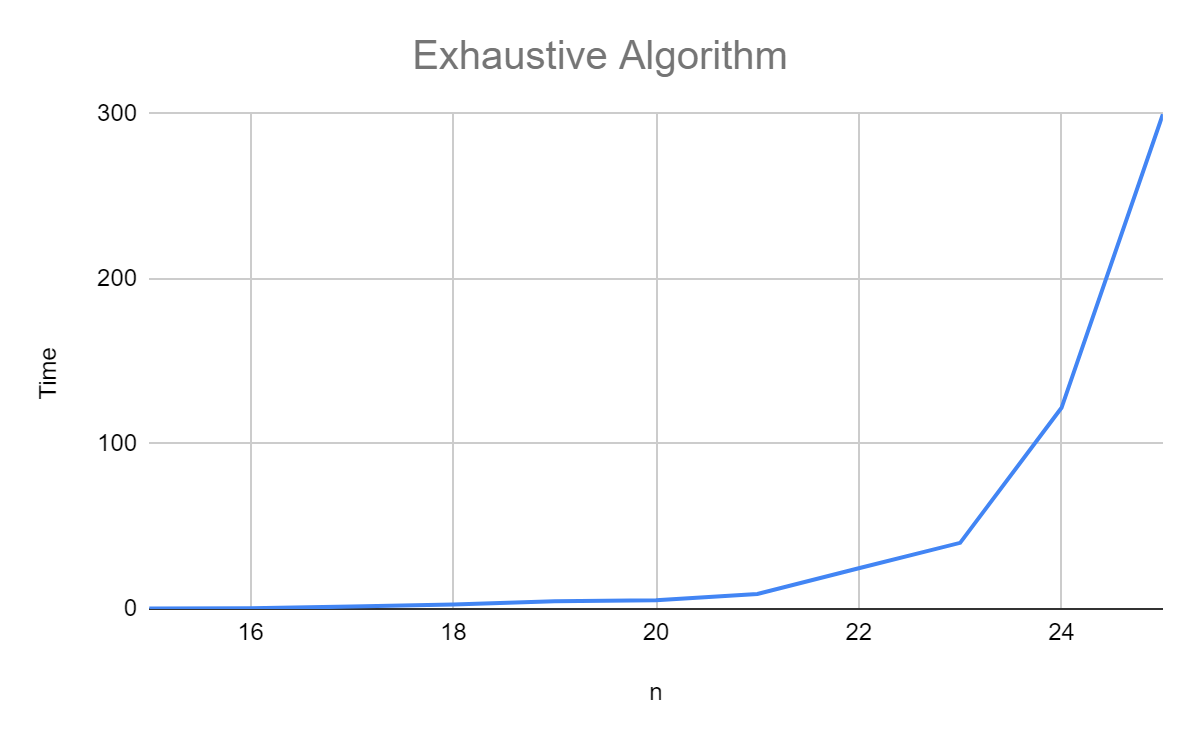
**Dynamic Algorithm Code**

| cell\_type from\_above; // 1  cell\_type from\_left; // 1  if (r > 0 && A[r - 1][c].has\_value()) { //2  from\_above = A[r - 1][c]; // 1  if (from\_above->is\_step\_valid(STEP\_DIRECTION\_SOUTH)) { // 1  from\_above->add\_step(STEP\_DIRECTION\_SOUTH); // 1  }  }  if (c > 0 && A[r][c - 1].has\_value()) { // 2  from\_left = A[r][c - 1]; // 1  if (from\_left->is\_step\_valid(STEP\_DIRECTION\_EAST)) { //1  from\_left->add\_step(STEP\_DIRECTION\_EAST);  }  }  if (from\_above.has\_value() && from\_left.has\_value()) { // 1  if (from\_above->total\_cranes() > from\_left->total\_cranes()) { //1  A[r][c] = from\_above; // 1  }  else {  A[r][c] = from\_left; // 1  }  }  else if (from\_above.has\_value() && !from\_left.has\_value()) { // 1  A[r][c] = from\_above; // 1  }  else if (from\_left.has\_value() && !from\_above.has\_value()) { //1  A[r][c] = from\_left; // 1  }  else { //3  A[r][c] = path(setting); // 1  } |
| --- |

**Dynamic Mathematical Analysis**

| O(1 + n(1 + n(1 + n(n(1 + 1 + 1 + 1 + 1 + 1 + n(1 + 1 + 1 + 1 + 1 + 1 + 1)))))  O(1 + n(1 + n(1 + n(n(6 + n(7))))  O(1 + n(1 + n(1 + 6𝑛2 + 7n3)  O(1 + n(1 + n + 6𝑛3 + 7n4 )  O(1 + n + 𝑛2 +6n4 +7n5 )  = O(𝑛2 ) |
| --- |

**Scatterplot**

****

1. **Is there a noticeable difference in the performance of the two algorithms? Which is faster, and by how much? Does this surprise you?**

There is a clear difference in the performance between the dynamic algorithm and the exhaustive search algorithm. This is no surprise because the exhaustive search has a time complexity of O(2n\*n) which is exponential. This causes it to take way longer for it to execute as compared to the dynamic algorithm.

1. **Are your empirical analyses consistent with your mathematical analyses? Justify your answer.**

Yes, the empirical analysis is consistent with the mathematical analysis because for both algorithms, they match the time complexities.

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

Yes, the evidence is consistent with the hypothesis that polynomial-time dynamic programming algorithms are more efficient than exponential-time exhaustive search algorithms that solve the same problem. This is because the exhaustive algorithm has an exponential time, whereas the dynamic algorithm has a polynomial time complexity. This results in the dynamic program being more efficient than the exhaustive algorithm.