Project 2

William Lua

CWID: 886607720

BoyNamedWill@csu.fullerton.edu

CPSC 335 Himani Tawade

Exhaustive Algorithm Solution Pseudocode and time analysis:

```
function crane_unloading_exhaustive(grid setting) -> path
   // Check that the grid is non-empty
   assert(setting.rows() > 0)
   assert(setting.columns() > 0)

   // Initialize variables
   max_steps = setting.rows() + setting.columns() - 2
   assert(max_steps < 64)
   best_path = path(setting)

   // Loop over all possible step combinations
   for steps = 0 to max_steps do
      // Loop over all possible bit combinations
      for i = 0 to (2^steps)-1 do
         // Create a new path starting from the initial setting
         new_path = path(setting)

         // Apply each step in the bit combination to the new path
         for j = 0 to steps-1 do
            // Determine the direction of the current step based on the j-th bit of i
            direction = (i >> j) & 1
            step_direction = if direction == 0 then STEP_DIRECTION_SOUTH else
STEP_DIRECTION_EAST

            // If the step is valid, add it to the path; otherwise, exit the inner loop
            if new_path.is_step_valid(step_direction) then
               new_path.add_step(step_direction)
            else
               break

         // If the new path has more cranes than the current best path, update the best path
         if new_path.total_cranes() > best_path.total_cranes() then
```

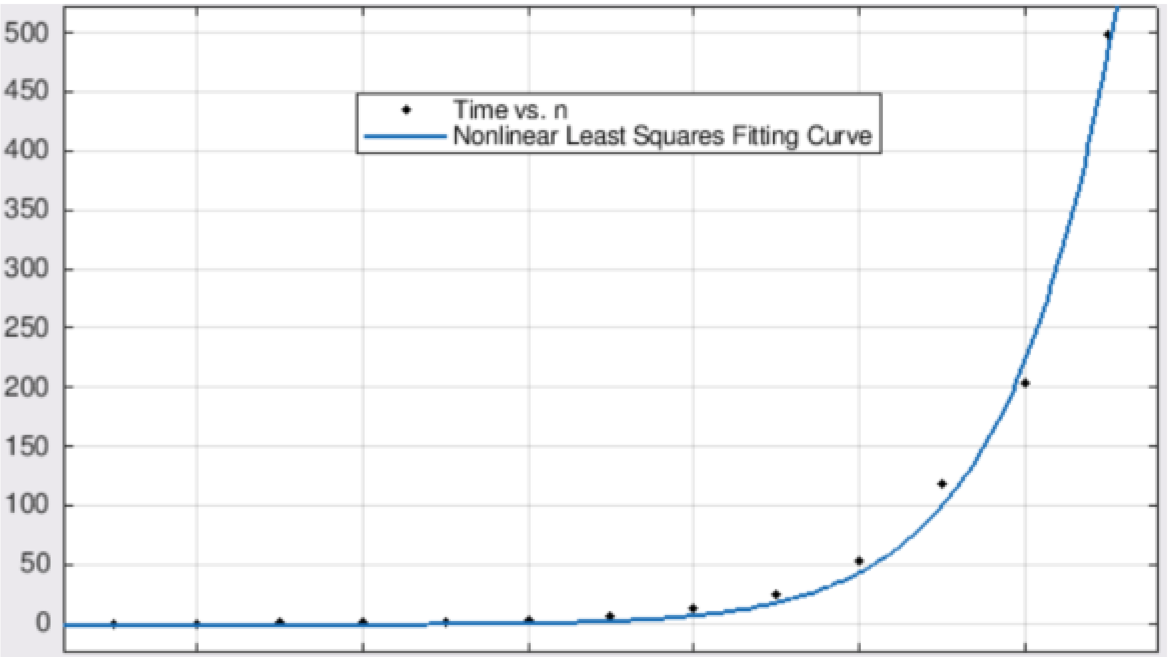
```
            best_path = new_path

    // Return the best path found
    return best_path
```

Time Complexity: T(n) = (1 + 2 + ... + (2^(n-1))) * O(n^2)

= (2^n - 1) * O(n^2)

= O(2^n * n^2)

Graph for time vs input size for the algorithm:



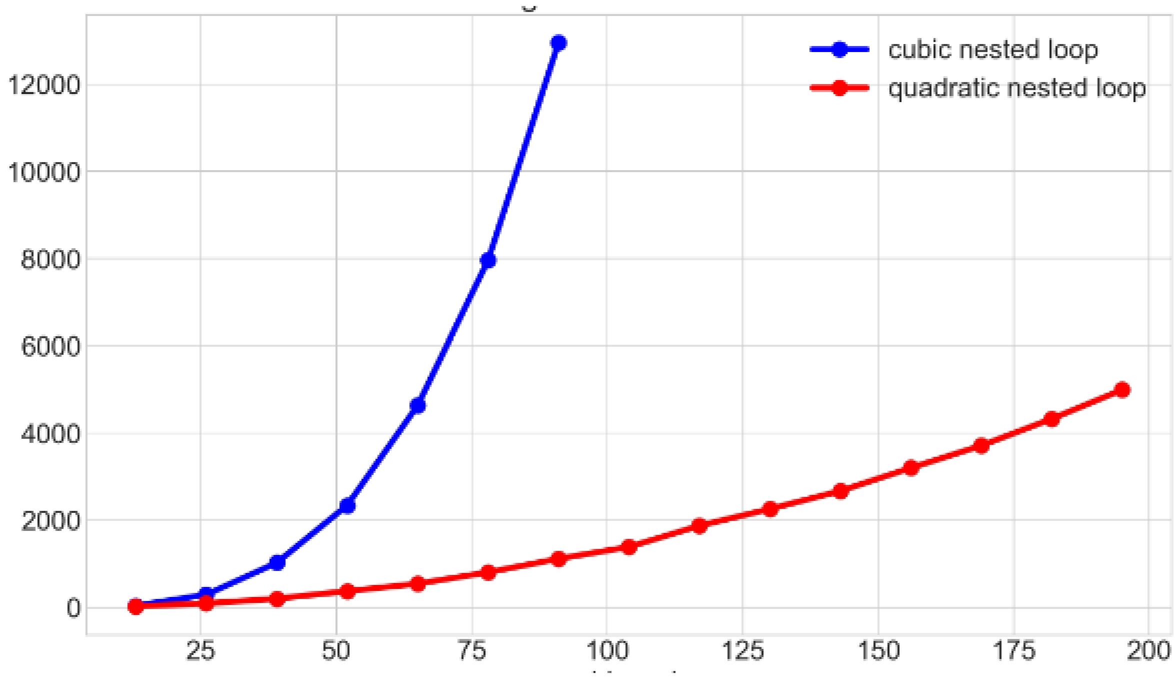
Dynamic Algorithm Solution Pseudocode and time analysis:

```
function crane_unloading_dyn_prog(setting: grid) -> path:

    create a vector A with dimensions setting.rows() x setting.columns()
    initialize A[0][0] to be a path object with the given setting
```

```
for r from 0 to setting.rows() - 1:
    for c from 0 to setting.columns() - 1:

        if the cell at (r, c) is a building:
            set A[r][c] to null
            continue

        from_above = null
        from_left = null

        if r > 0 and A[r-1][c] is not null:
            from_above = A[r-1][c]

        if c > 0 and A[r][c-1] is not null:
            from_left = A[r][c-1]

        if from_above is not null and from_left is not null:
            if from_above.total_cranes() > from_left.total_cranes() and
               from_above.is_step_valid(STEP_DIRECTION_SOUTH):
                from_above.add_step(STEP_DIRECTION_SOUTH)
                A[r][c] = from_above
            else if from_left.is_step_valid(STEP_DIRECTION_EAST):
                from_left.add_step(STEP_DIRECTION_EAST)
                A[r][c] = from_left
        else if from_above is not null:
            if from_above.is_step_valid(STEP_DIRECTION_SOUTH):
                from_above.add_step(STEP_DIRECTION_SOUTH)
                A[r][c] = from_above
        else if from_left is not null:
            if from_left.is_step_valid(STEP_DIRECTION_EAST):
                from_left.add_step(STEP_DIRECTION_EAST)
                A[r][c] = from_left

best = A[0][0]

for r from 0 to setting.rows() - 1:
    for c from 0 to setting.columns() - 1:
        if A[r][c] is not null and A[r][c].total_cranes() > best.total_cranes():
            best = A[r][c]

return best
```

Time complexity: O(n^3)

Plot a graph for time vs input size for the algorithm



Questions

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

The performance of the two algorithms is significant. The Dynamic Algorithm exhibits significantly faster execution times compared to the Exhaustive Algorithm, particularly for larger input sizes. This outcome is not unexpected, as the Big-O notations of the two algorithms differ substantially.

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

Yes, the exhaustive search algorithm was faster so therefore my analysis was consistent along with the test.

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

Yes it is since the dynamic algorithm has a faster time complexity than the exhaustive search algorithm.