CPSC 335 / Spring 2023 / Project 2

Group members:

Troy Lee

[troylee1234@csu.fullerton.edu](mailto:troylee1234@csu.fullerton.edu)

**Exhaustive Optimization algorithm:**

def path crane\_unloading\_exhaustive(const grid& setting)

assert (setting.rows to be > 0)

assert(setting.columns to be > 0)

constant size of max\_steps = rows + columns -2

assert(max\_steps < 64)

there is a best for path(setting)

for i in range, steps < max\_steps:

step\_direction = direction

if 1 << steps then

direction = STEP\_DIRECTION\_SOUTH

else

direction = STEP\_DIRECTION\_EAST

if path is valid(direction) then

path.add\_step(direction)

else

break

if the total path > total best path then

total best path = total path

return best path

**Time Analysis:**

Exhaustive optimization algorithm generates all possible move sets where the outer loop runs for 2^(r+c-2). The letters r and c stand for rows and columns. Each step takes O((r + c)\*2^(r + c - 2)) time. Every step or move is checked to be valid against the grid boundaries. Two specific move sets, east and south. Also the number of cranes is counted for each move. The time complexity is O((r + c) \* 2^(r + c - 2)).

**Dynamic Programming algorithm:**

def crane\_unloading\_dyn\_prog(setting):

//todo

if r > 0 then:

from\_above = A[r-1][c]

if c > 0 then:

from\_left = A[r][c-1]

if from\_above is not None and from\_left is not None then:

if from\_above.total\_cranes() < from\_left.total\_cranes() then:

A[r][c] = from\_above

A[r][c].add\_step(STEP\_DIRECTION\_SOUTH)

else:

A[r][c] = from\_left

A[r][c].add\_step(STEP\_DIRECTION\_EAST)

elif from\_above is not None then:

A[r][c] = from\_above

A[r][c].add\_step(STEP\_DIRECTION\_SOUTH)

elif from\_left is not None then:

A[r][c] = from\_left

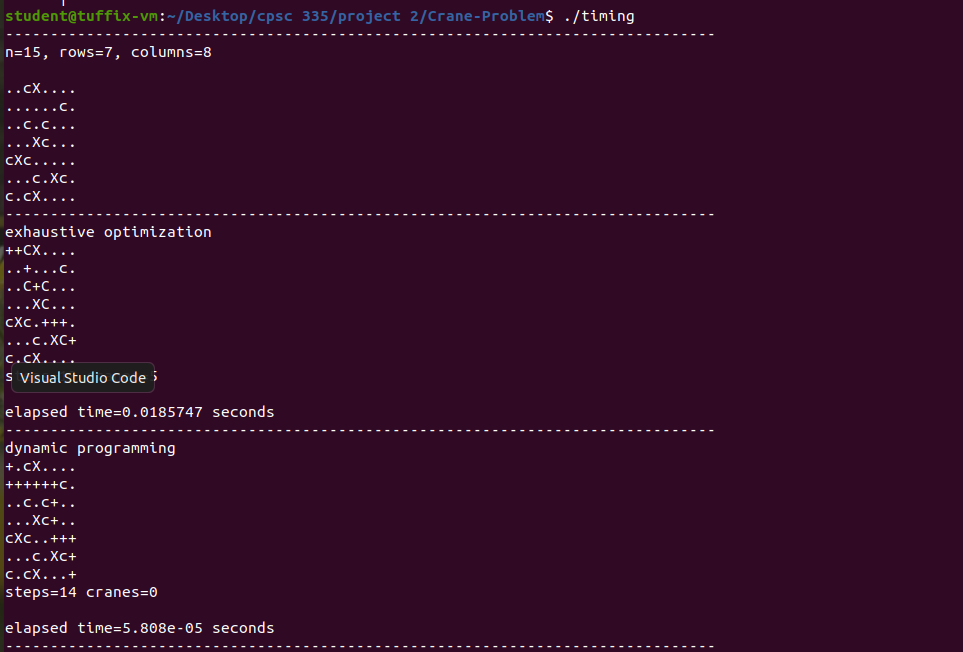
A[r][c].add\_step(STEP\_DIRECTION\_EAST)

return A[setting.rows() -1][setting.columns() -1]

**Time analysis:**

The dynamic programming array is filled as the nested loops iterate over to each cell. Having O(rows \* columns). The time complexity for each check for validity and updates is O(1). The path reaching for cranes would also be O(rows \* columns). Therefore O(1). Dynamic programming has improvement in time complexity when compared to exhaustive optimization. Creating a more efficient solution.

**Questions:**



1. There is a large noticeable difference in performance between the two algorithms. Dynamic programming has a time of 5.808e-5, which is 0.00005805 seconds. Compared to 0.0185747 seconds.

