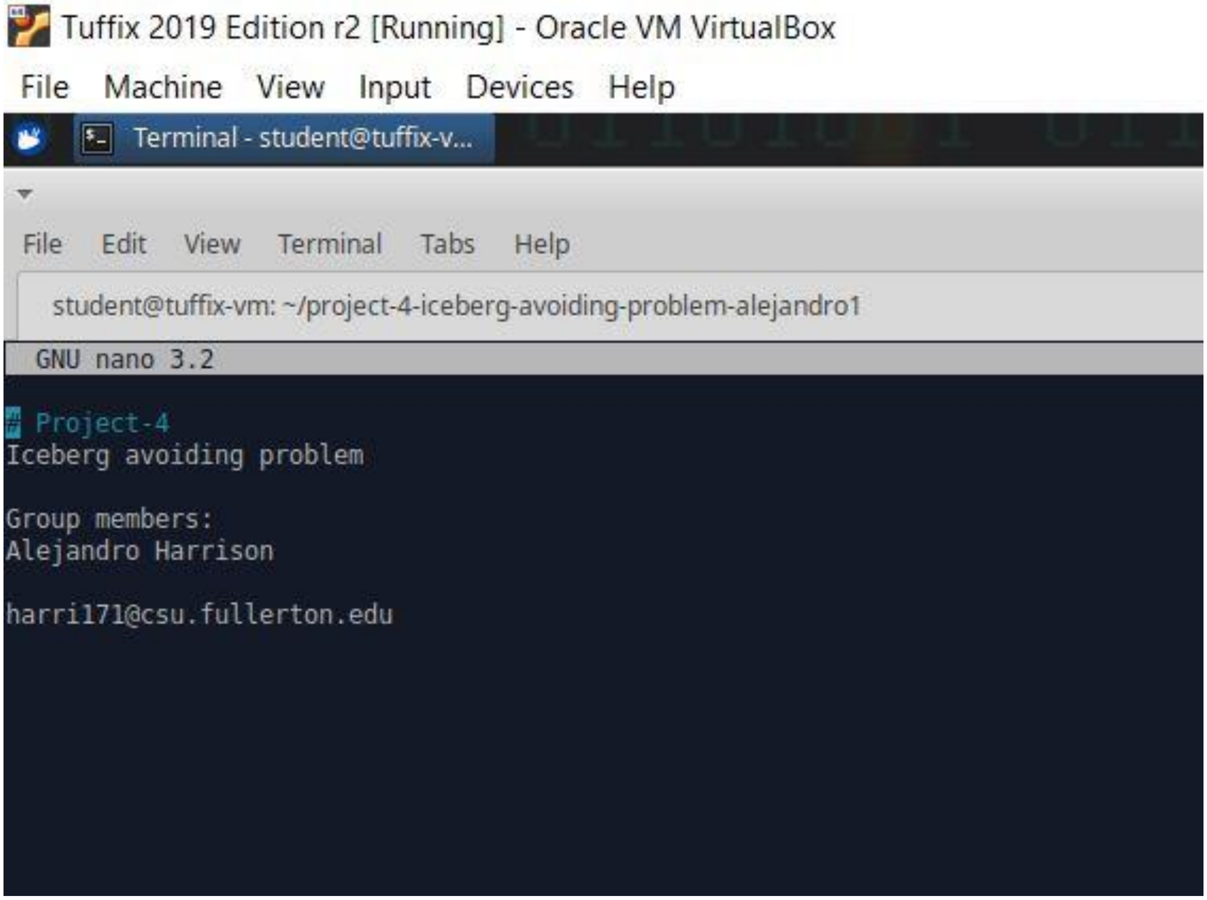
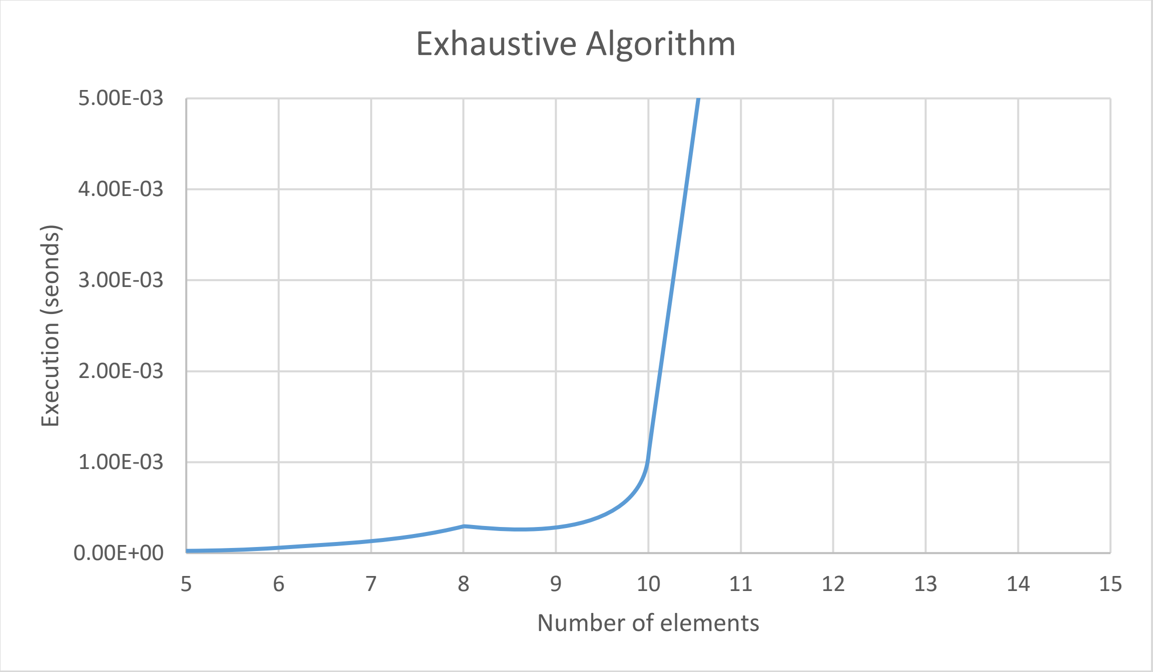
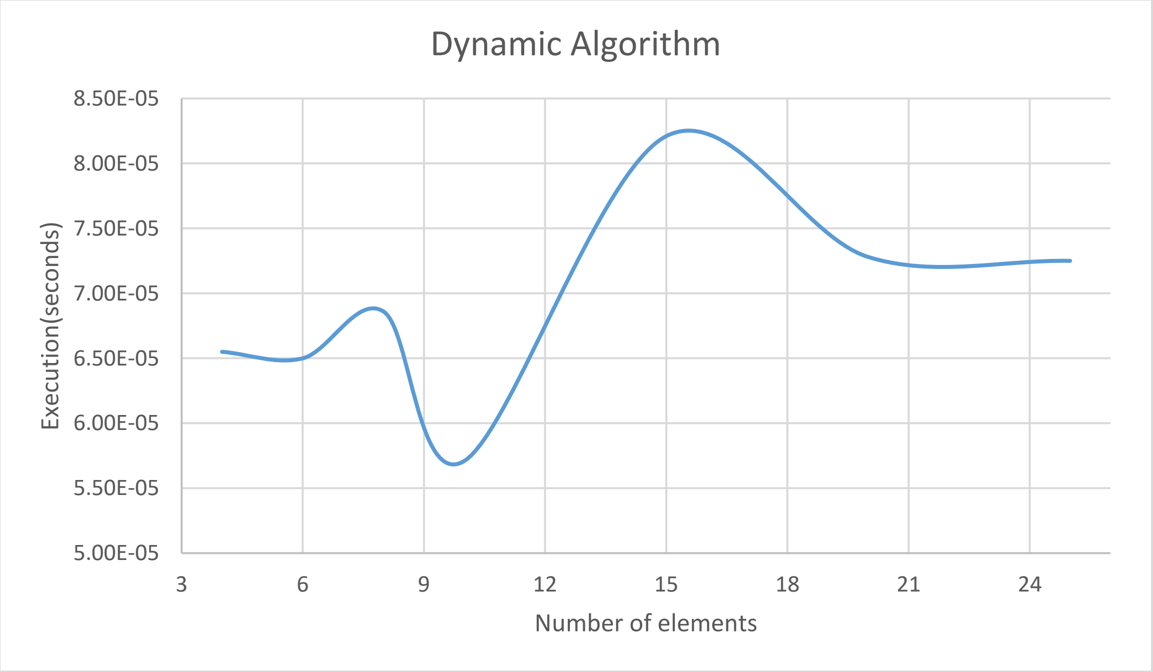

Tuffix 2019 Edition r2 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Terminal - student@tuffix-v...
File Edit View Terminal Tabs Help
student@tuffix-vm: ~/project-4-iceberg-avoiding-problem-alejandro1
GNU nano 3.2

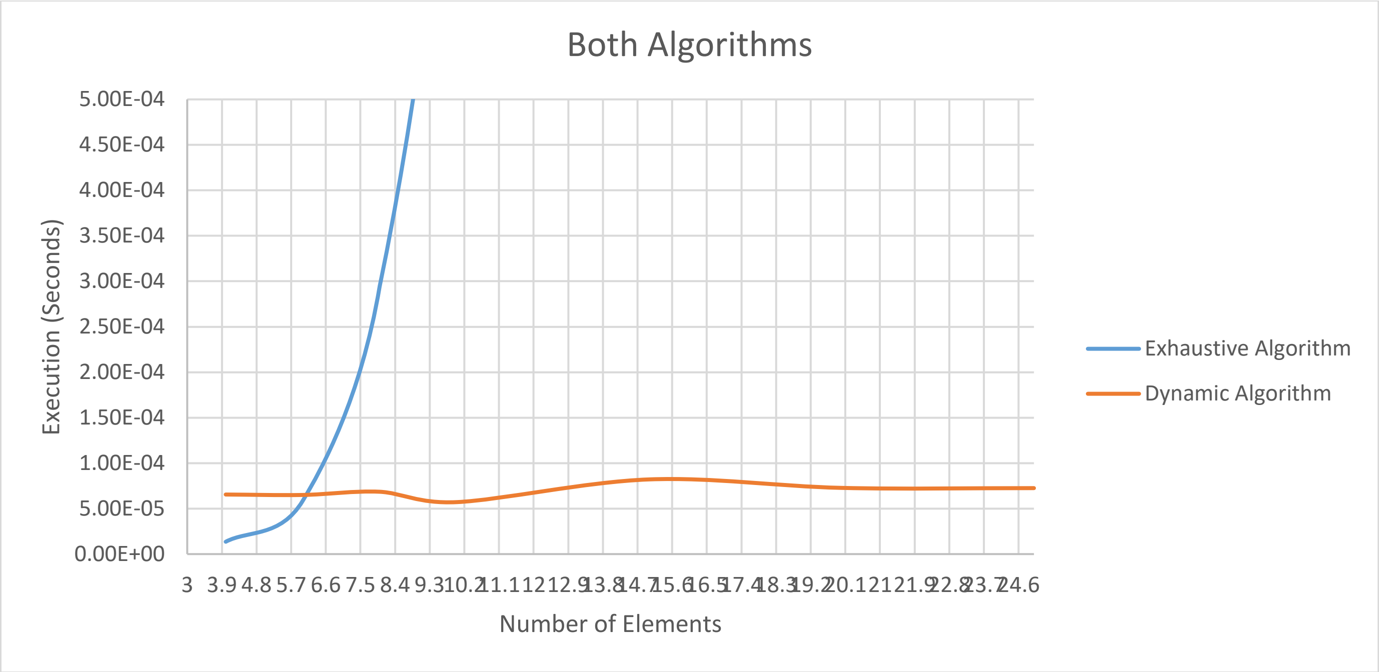
```
# Project-4
Iceberg avoiding problem

Group members:
Alejandro Harrison

harri171@csu.fullerton.edu
```

Alejandro Harrison

Harri171@csu.fullerton.edu


Exhaustive Algorithm
5.00E-03
4.00E-03
3.00E-03
2.00E-03
1.00E-03
0.00E+00
Execution (seonds)
5
6
7
8
9
10
11
12
13
14
15
Number of elements


Dynamic Algorithm
8.50E-05
8.00E-05
7.50E-05
7.00E-05
6.50E-05
6.00E-05
5.50E-05
5.00E-05
Execution(seconds)
3
6
9
12
15
18
21
24
Number of elements



A) It definitely seems like the curves/lines fit the efficiency class. Since the exhaustive algorithm just executes slower and slower the more elements you add, and has that exponential shape. The dynamic algorithm also resembles a shape of the *O*(n^2) efficiency class.

B) The original hypothesis that dynamic algorithms execute more efficiently seems to be correct based on the execution times of the 2 algorithms.

C) The dynamic algorithm was definitely easier to comprehend and implement as compared to the exhaustive. For the exhaustive, the hardest part was understanding the bit & as well as wrapping my head around the fact that each integer could be represented as a binary string, which in turn represented a move set.