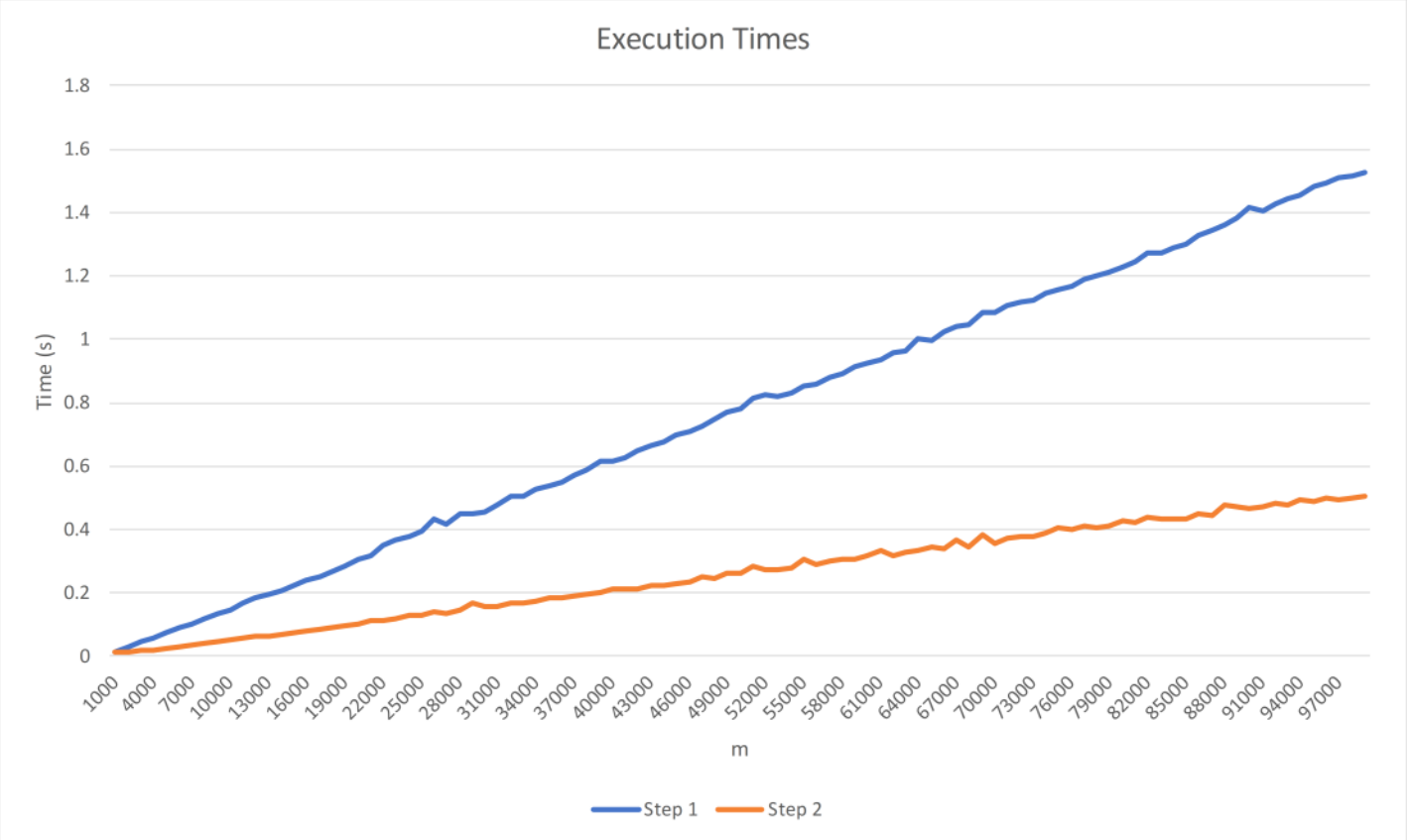
# Question 4

Monday, January 21, 2019 12:25 AM




Windows edition

Windows 10 Home



Windows 10

System

| | |
|---|---|
| Processor: | Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz 3.60 GHz |
| Installed memory (RAM): | 16.0 GB |
| System type: | 64-bit Operating System, x64-based processor |

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void arrTest(int);

FILE *f;

int main() {
    int m;
    f = fopen("memory-frag-output.txt", "w");

    // User input for m
    printf("\nEnter a value for m to go to: ");
    scanf("%d", &m);

    fprintf(f, "m, Step 1, Step 2\n");
    for(int i = 1000; i < m; i += 1000)
        arrTest(i);
/*      arrTest(m); */
}

void arrTest(int m) {
    double time_spent;

    printf("m is %d\t\t", m);
    fprintf(f, "%d", m);

    // Step 1
    clock_t begin_step1 = clock();
    // allocate memory for 3m int arrays for size 800000 each
    int** arrays1 = malloc(sizeof(int*) * 3*m);
    for(int i = 0; i < 3*m; i++)
        arrays1[i] = malloc(sizeof(int[800000]));
    clock_t end_step1 = clock();
    time_spent = (double)(end_step1-begin_step1) / CLOCKS_PER_SEC;
    printf(" | Step 1: %f seconds", time_spent);
    fprintf(f, " %f,", time_spent);

    // Step 2
    clock_t begin_step2 = clock();
    // deallocate memory even numbered arrays from 3m arrays
    for(int i = 0; i < 3*m; i += 2)
        free(arrays1[i]);
    // and allocate memory for m int arrays for size 9000000 each
    int** arrays2 = malloc(sizeof(int*) * m);
    for(int i = 0; i < m; i++)
        arrays2[i] = malloc(sizeof(int[900000]));
    clock_t end_step2 = clock();
    time_spent = (double)(end_step2-begin_step2) / CLOCKS_PER_SEC;
    printf(" | Step 2: %f seconds\n", time_spent);
    fprintf(f, "%f\n", time_spent);

    free(arrays1);
    free(arrays2);
}
```

Test: see how the time fluctuates between the chosen $m$ (array size) value

- Step 1: Allocate memory for a sequence of 3m arrays of size 800,000 elements each
- Step 2: Deallocate even-numbered arrays and allocate m arrays of size 900,000 elements each
- The amount of seconds are for array sizes $m = k1000$ where the integer $k \geq 1$

  $m = \{1000,2000, \dots, 98000,99000\}$

In this test, the time for Step 1 and Step 2 grow linearly showing a direct correlation between array size and how long this scenario will run. At any array size the time seems to stay at the same growth pattern

There was an issue when $m$ was 270000

```
Enter a value for m to go to: 500000
m is 1000               | Step 1: 0.018000 seconds | Step 2: 0.013000 seconds
m is 2000               | Step 1: 0.034000 seconds | Step 2: 0.015000 seconds
m is 3000               | Step 1: 0.050000 seconds | Step 2: 0.019000 seconds
```

...

```
m is 267000             | Step 1: 4.568000 seconds | Step 2: 1.524000 seconds
m is 268000             | Step 1: 4.561000 seconds | Step 2: 1.526000 seconds
m is 269000             | Step 1: 4.586000 seconds | Step 2: 1.531000 seconds
m is 270000
PS Z:\GoogleDrive\Education\College\UofH\AlgorithmsAndDataStructures-COSC3320\Assignments\Assignment1\Question4>
```

This caused the program to crash

Because of this, I decided to only have $m$ be between 1000 and 100000 in my testing