# Empirical Analysis of the Johnson Trotter and Grey Code Algorithms

CS330 - Connor Deakin - 12/14/17

---

## Johnson Trotter

---

### Problem Statement

The Johnson Trotter algorithm aims to create every permutation of a set by only switching the position of two elements within the set during each iteration. Each iteration of the algorithm is guaranteed to produce a unique permutation of the set until the last permutation is reached.

Here, we will test the Johnson Trotter algorithm to confirm that is does indeed visit every permutation once and only once.

### Experiment Setup

To confirm this, we analyze the amount of processing time required to visit every permutation with sets of size (N). Each set size (N) tested is in the range [2, 12]. For each N, 10 tests are performed and the runtime from the tests are averaged to get an accurate portrayal of the algorithm's actual run time.

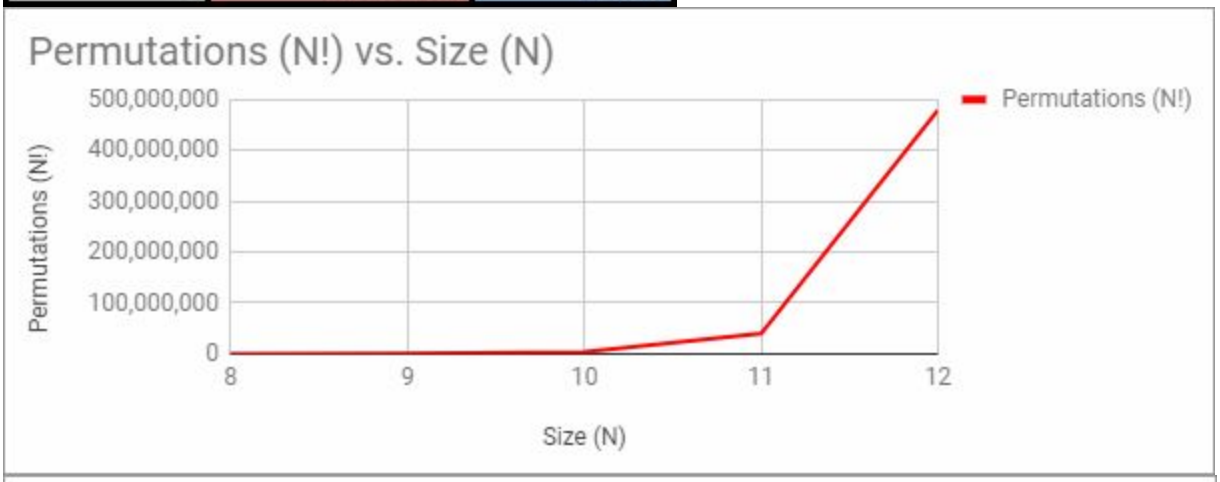
### Results

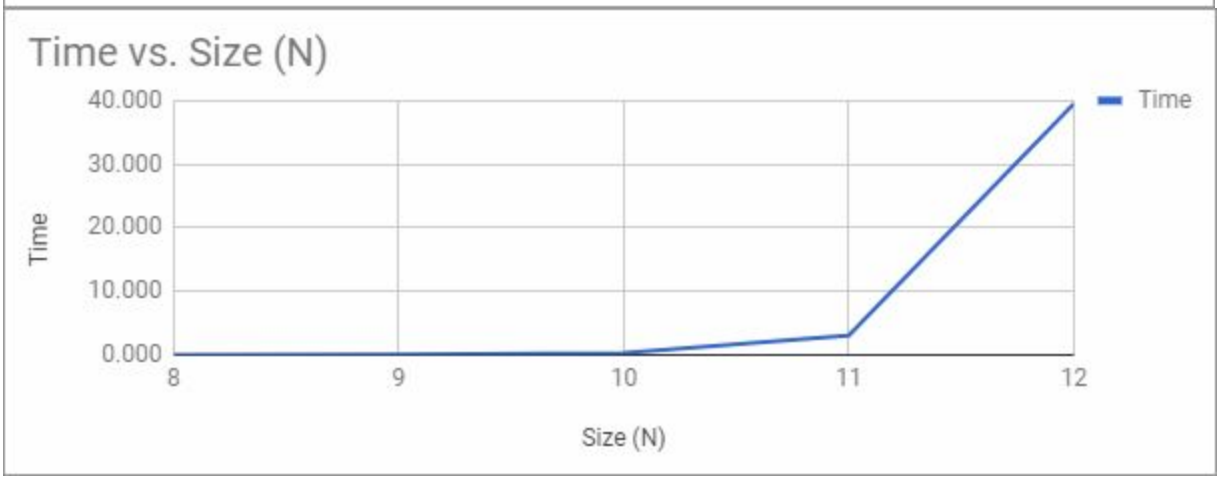
The table and graphs for the results are displayed on the next page (2). The Permutations vs. Size (PvS) graph shows the growth rate of the number of permutations as the size of the set increases. The Time vs. Size (TvS) graph shows the growth rate of the Johnson Trotter algorithm runtime as the size of the set increases.

### Conclusion

Notice that PvS is actually a graph of N! vs. N. The Johnson Trotter algorithm runtime follows the same trend as the N! vs. N graph. This supports the original hypothesis; the Johnson Trotter algorithm visits each permutation in a set's N! permutations once and only once.

| Johnson Trotter | | |
|---|---|---|
| Size (N) | Permutations (N!) | Time |
| 2 | 2 | 0.003 |
| 3 | 6 | 0.000 |
| 4 | 24 | 0.009 |
| 5 | 120 | 0.009 |
| 6 | 720 | 0.000 |
| 7 | 5,040 | 0.003 |
| 8 | 40,320 | 0.000 |
| 9 | 362,880 | 0.033 |
| 10 | 3,628,800 | 0.253 |
| 11 | 39,916,800 | 3.040 |
| 12 | 479,001,600 | 39.508 |



# Grey Code

---

## Problem Statement

The goal of the Grey Code algorithm is to visit every single subset that exists within a set once and only once. This is done by starting with the empty set and adding or removing only one element from our current subset during each iteration. This is done until the Grey Code algorithm creates every possible subset.

Our goal is to test the Grey Code algorithm to make sure that it visits every subset once and only once.

## Experiment Setup

To do this, we will analyze the amount of processing time required to visit every unique subset within a set of size (N). The sizes of the sets tested will be in the range of [1, 30]. For every N, the Grey Code algorithm will run 5 times and the average runtime from those five tests will represent the runtime of the Grey Code algorithm for that set size.

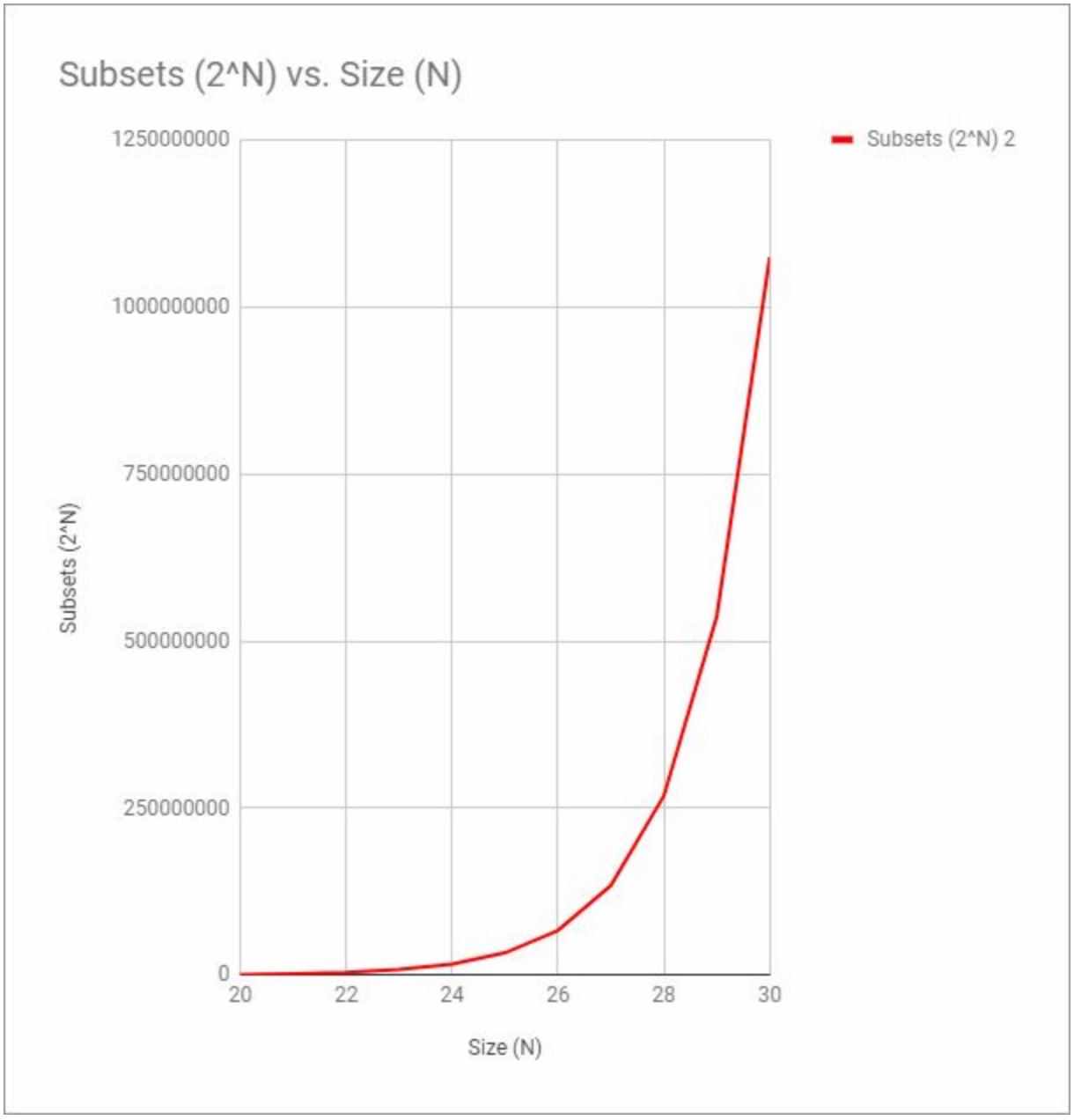
## Results

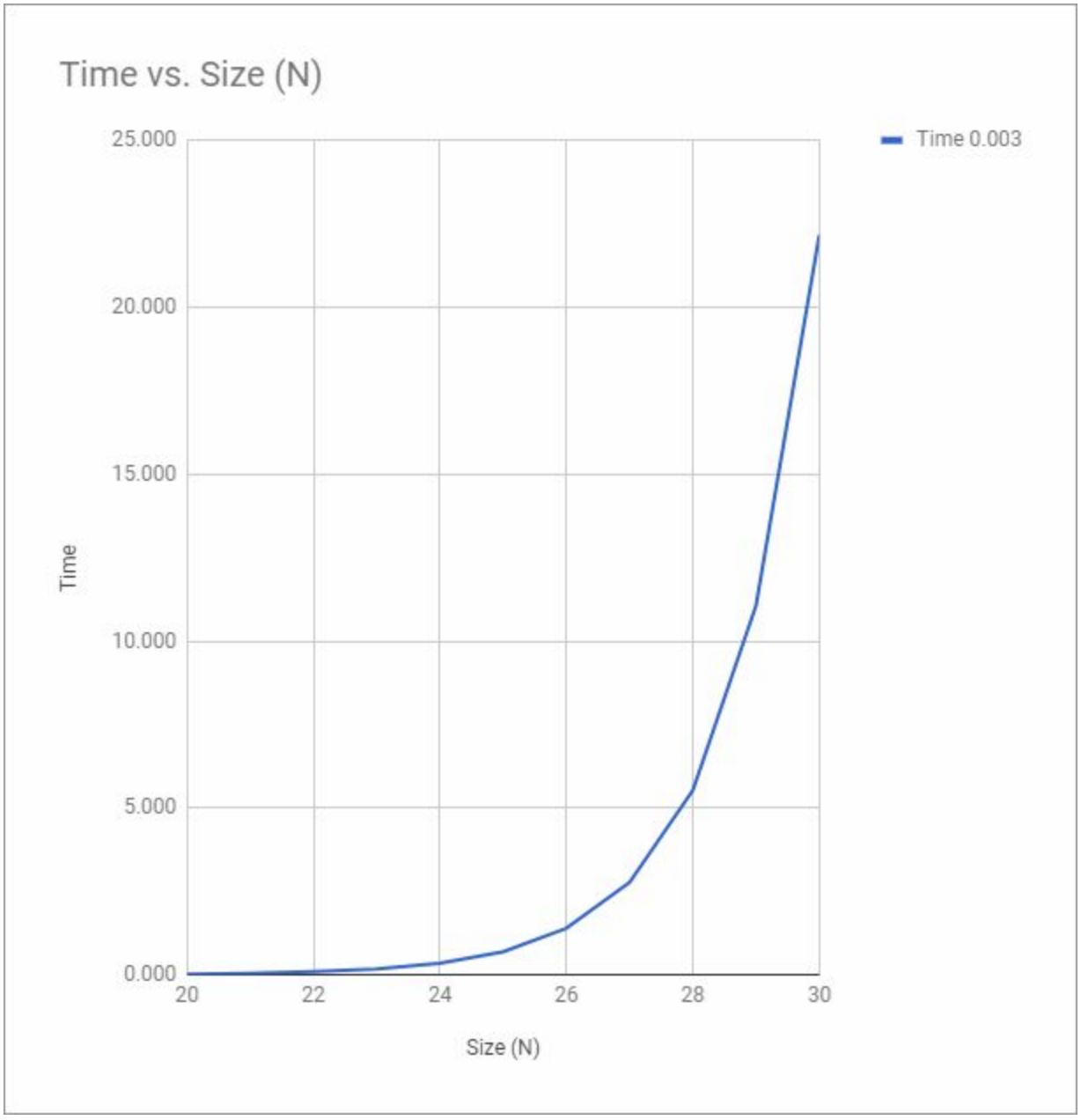
The table and graphs from the results of the tests can be found on the next couple of pages (4-6). The first graph, Subset vs. Size (SvS), shows the growth rate in the number of subsets against the size of the set. The second graph, Time vs. Size (TvS), shows the growth rate of the Grey Code algorithm runtime as the size of the set increases.

## Conclusion

SvS displays a growth rate of 2^N. TvS follows this same growth rate curve. The similar growth rates in these two graphs demonstrate that the Grey Code algorithm visits every subset within a set's 2^N subsets once and only once.

| Gray Code | | |
|---|---|---|
| Size (N) | Subsets (2^N) | Time |
| 1 | 2 | 0.003 |
| 2 | 4 | 0.000 |
| 3 | 8 | 0.000 |
| 4 | 16 | 0.000 |
| 5 | 32 | 0.003 |
| 6 | 64 | 0.000 |
| 7 | 128 | 0.006 |
| 8 | 256 | 0.000 |
| 9 | 512 | 0.000 |
| 10 | 1024 | 0.000 |
| 11 | 2048 | 0.003 |
| 12 | 4096 | 0.000 |
| 13 | 8192 | 0.003 |
| 14 | 16384 | 0.003 |
| 15 | 32768 | 0.003 |
| 16 | 65536 | 0.003 |
| 17 | 131072 | 0.009 |
| 18 | 262144 | 0.009 |
| 19 | 524288 | 0.009 |
| 20 | 1048576 | 0.025 |
| 21 | 2097152 | 0.046 |
| 22 | 4194304 | 0.093 |
| 23 | 8388608 | 0.171 |
| 24 | 16777216 | 0.343 |
| 25 | 33554432 | 0.690 |
| 26 | 67108864 | 1.400 |
| 27 | 134217728 | 2.771 |
| 28 | 268435456 | 5.524 |
| 29 | 536870912 | 11.071 |
| 30 | 1073741824 | 22.165 |


Subsets (2^N) vs. Size (N)
Subsets (2^N) 2
1250000000
1000000000
750000000
500000000
250000000
0
Subsets (2^N)
20
22
24
26
28
30
Size (N)


Time vs. Size (N)
Time 0.003
25.000
20.000
15.000
10.000
5.000
0.000
Time
20
22
24
26
28
30
Size (N)