(A) The growth function of the first algorithm (alg1) is “4n^2 + 2” and its complexity is “O(n^2).” (B) Alg2 shuffles the sets of numbers into different positions. Alg1 arranges the sets given from least to greatest. It also seems if you use the two algorithms during the same execution, one will mess with the other. If alg1 comes first, numbers from both are ordered from least to greatest. If vice versa, alg2 randomizes the numbers, then alg1 orders them.

(C1) Alg1’s growth function suggests that there would be 402 operations performed with an array size of 10, though the number of operations returned by the function was 55. The numbers returned from array sizes of 100 and 1000 are also tabled below.

|  |  |  |
| --- | --- | --- |
| Alg1 | | |
| n | T(n) = 4n^2 + 2 | Value Returned |
| 10 | 402 | 55 |
| 100 | 40,002 | 5,050 |
| 1000 | 4,000,002 | 500,500 |

The numbers of operations performed are significantly smaller than what the growth function suggests. Also, judging by the growth function and the outputs as well, the function itself is linear. (C2) This must the opposite for alg2 because the values returned seem proportionate to the input (array size) and also do not have a set pattern across them, as shown below.

|  |  |
| --- | --- |
| Alg2 | |
| N | Value Returned (Multiple Trials; Estimated Ranges) |
| 100 | 1,660 – 1,800 |
| 1,000 | 22,220 – 24,480 |
| 10,000 | 272,650 – 286,860 |
| 20,000 | 581,410 – 624,980 |

Of all the BigO notations, O(log(n)) seems to fit best because of how huge the differences between input and output are, especially as the input . Also, I had to rule out the linear complexities because, the algorithm’s outputs (return values) don’t show a linear pattern between them like O(n) or O(n^2).

|  |  |  |
| --- | --- | --- |
| C3 | Values Returned (Separate Arrays; Multi. Trials; Est. Ranges) | |
| Input (n) | Alg1 | Alg2 |
| 10 | 55 | 100 – 125 |
| 100 | 5,050 | 1,630 – 1,895 |
| 1,000 | 500,500 | 22,125 – 23,220 |
| 10,000 | 50,005,000 | 273,520 – 298,390 |

For smaller much arrays, I would prefer to use the first algorithm (Alg1) since the second (Alg2) would take longer, regardless of precision. Once the array sizes become larger, Alg2 would be used because even though the number of operations varies, even the (approx.) maximum amount of operations used is still less than the set amount that Alg1 takes.