1. Comparing algorithms

Algo1: time complexity: O(n^2), space complexity O(n). Used extra array and nested loops. Speed is faster than Algo2 because, we do not check 'divide by 2' condition on each element during nested loop traversal, instead we check once on first loop and saved result to extra array, then loop on new array to find max difference. Algorithm gave up extra 'n' space complexity to reach achieve better runtime performance.

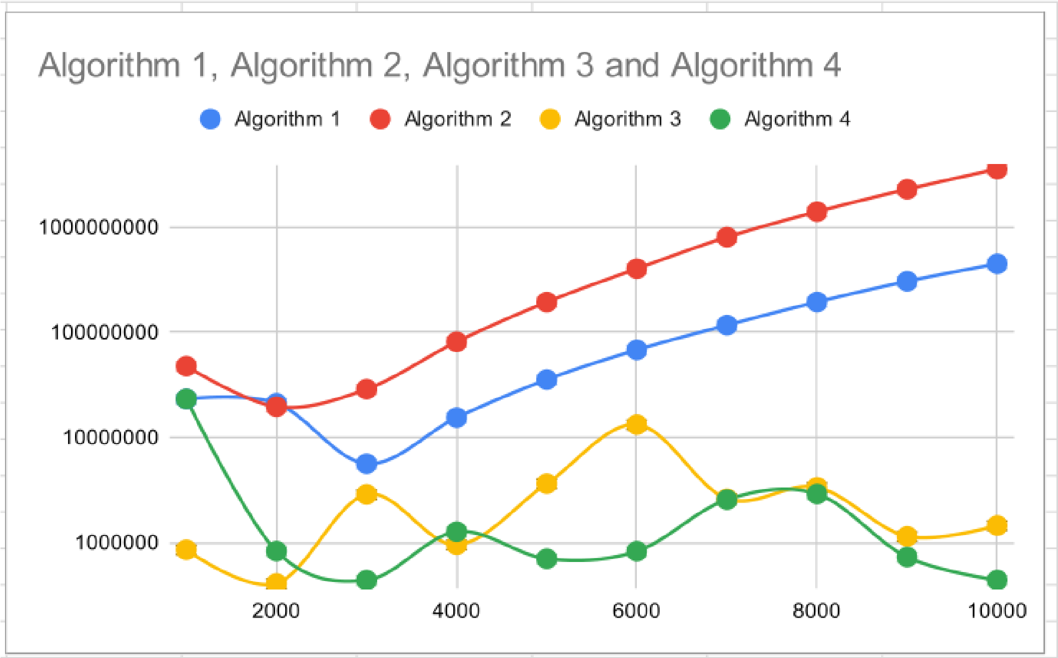
Algo2: time complexity: O(n^2), space complexity O(1). It is slower than Algo1, because it checks 'divided by two' conditions on each element of nested loops traversal. But we have less space complexity than Algo1.

Algo3: time complexity: O(n), space complexity O(1). It is much faster since we use only a single loop. Also we did not use extra space for this algorithm.

Algo4: time complexity: O(n), space complexity O(1). It is identical time and space complexity to Algo3. But we could see slight differences at different inputs.

X axis shows size of array we execute our algorithm on.

Y axis shows time our program took in nanoseconds.



2. Induction proof of F(n) > (4/3)^n for n > 4.

2.1. Base case:

$$let\ n = 5, LHS: f(5) = 5 > \left(\frac{4}{3}\right)^5 \approx 4.21 = RHS$$

$$let\ n = 6, LHS: f(6) = 8 > \left(\frac{4}{3}\right)^5 \approx 5.61 = RHS$$

2.2. State induction hypothesis:

Assume the result is true for n = k

$$let\ n = k, LHS: f(k) > \left(\frac{4}{3}\right)^k$$

$$let\ n = k-1, LHS: f(k-1) > \left(\frac{4}{3}\right)^{k-1}$$

2.3. Induction.

Prove the result for n = k+1.

$$LHS = \ f(k+1) = f(k-1) + f(k)$$

$$= \left(\frac{4}{3}\right)^{k-1} + \left(\frac{4}{3}\right)^k = \left(\frac{4}{3}\right)^{k-1}\left(1 + \frac{4}{3}\right)$$

$$= \left(\frac{4}{3}\right)^{k-1}\left(\frac{7}{3}\right) > \left(\frac{4}{3}\right)^{k-1}\left(\frac{4}{3}\right)^2 = RHS$$

Proved, hence $\left(\frac{7}{3}\right) > \left(\frac{4}{3}\right)^2$ is true.