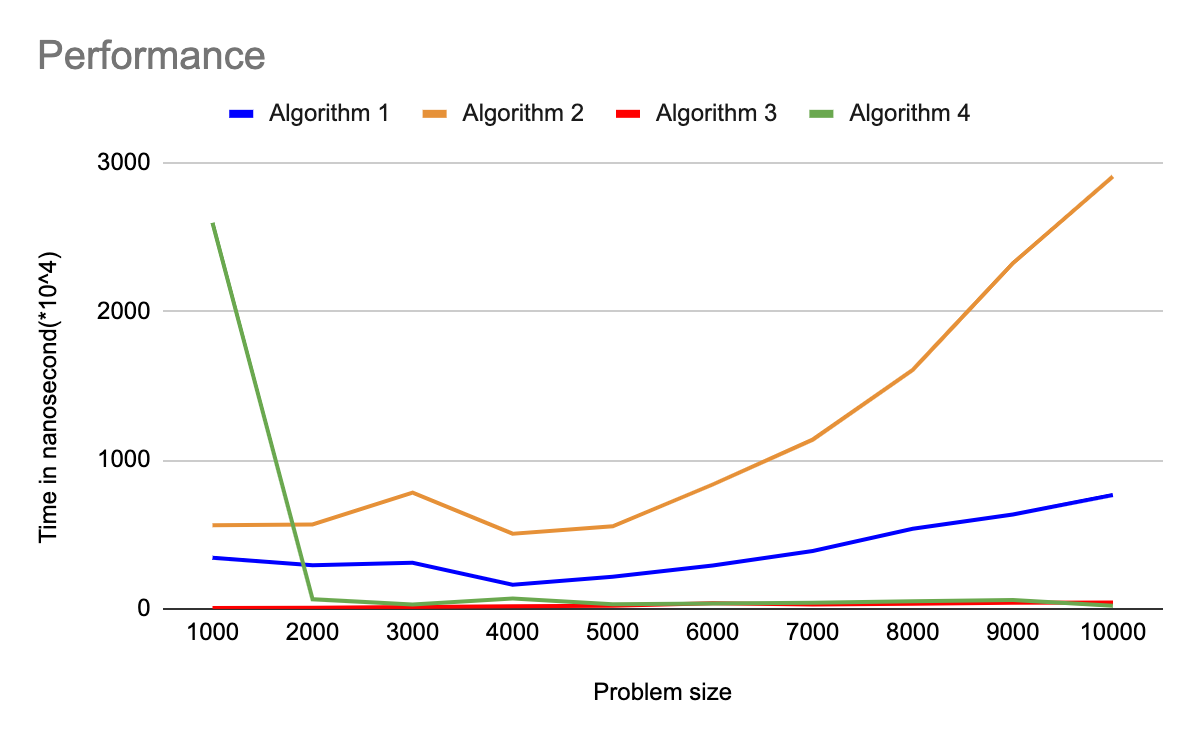
Question 1. Comparing Algorithms

We used the System.nanoTime() method in java to get time needed to solve the problem.

| Problem sizes | Algorithm 1 (\*10^4) | Algorithm 2  (\*10^4) | Algorithm 3  (\*10^4) | Algorithm 4  (\*10^4) |
| --- | --- | --- | --- | --- |
| 1000 | 344.5167 | 561.6750 | 6.2083 | 2595.5708 |
| 2000 | 293.0583 | 567.3042 | 8.5250 | 64.5583 |
| 3000 | 309.5666 | 781.2083 | 12.9458 | 28.8833 |
| 4000 | 162.2875 | 505.5500 | 17.1500 | 69.9584 |
| 5000 | 216.1250 | 555.1584 | 21.5416 | 31.3375 |
| 6000 | 291.0250 | 836.1833 | 38.2084 | 35.5542 |
| 7000 | 388.5958 | 1137.6250 | 30.2125 | 40.8667 |
| 8000 | 539.3792 | 1606.3417 | 34.3292 | 51.3958 |
| 9000 | 633.9291 | 2323.4333 | 41.0000 | 59.1334 |
| 10000 | 765.8583 | 2905.7542 | 42.2250 | 21.9166 |



The above graph shows the time required to find the max distance between two even numbers on different array sizes of 1000, 2000, …. 10000 integers using four different algorithms.

With Algorithm 1, the time to solve the problem rises subtly as the array size increases.

With Algorithm 2, it is similar to Algorithm 1. But when the array size increases, the time difference to take is significantly obvious while Algorithm 1 is under 1000ns.

With Algorithm 3, the time required to solve the problem remains steady even if the size of the arrays increases.

With Algorithm 4, the time for problem size 1000 takes a little bit long if it’s compared with other sizes. After that, it decreases steadily when the array size increases.

So, Algorithm 2(Use a nested loop to solve the problem without creating an extra array) is the worst of the four algorithms in terms of running time.

Algorithm 3(Use one loop. Find max and min of even integers. Compute max – min) is the best because it takes the shortest running time for different problem sizes among four algorithms.

Algorithm 4(Use Streams to find the max and min. Compute max – min) could be the second best because it decreases running time when array size increases although in the first 1000 problem size takes a little bit long.

Algorithm 1(Create a new array consisting of even numbers only. Then use nested loops to solve the problem using the newly created array of even numbers only) would be the third best algorithm for that problem.

Question 2. Proof by Induction

F(n) = F(n-1) + F(n-2).

F(1)=1

F(2)=1

F(3)=2

F(4)=3

Step 1. Prove base case(s).

Let n = 5.

Then LHS = F(5)=F(4)+F(3)=3+2=5 > (4/3)^5=20/3=4.21=RHS

Let n = 6.

Then LHS = F(6)=F(5)+F(4)=5+4=9 > (4/3)^6=24/3=7=RHS

Hence the base cases are proved.

Step 2. State induction hypothesis.

Assume the result is true for n = k,

F(k)=F(k-1) + F(k-2) > (4/3)^k

Step 3. Induction step.

Prove the result for n = k+1.

LHS = F((k+1)-1)+F((k+1)-2)

= F(k)+F(k-1)

//Using induction hypothesis

F(k)>(4/3)^k

F(k-1)>(4/3)^(k-1)

= (4/3)^k+(4/3)^(k-1)

=(4/3)^k[1+(4/3)^(-1)]

=(4/3)^k[1+(3/4)]

=(4/3)^k\*7/4 < (4/3)^k\*(4/3)=(4/3)^(k+1) = RHS