CS201 – HW 2

Name: Eylül Badem

ID: 22003079

Section: 01

Date: 06.04.2022

Complexity of Algorithms

Algorithm 1 (Naive algorithm): This algorithm includes one for loop which operates n times, and this for loop includes two basic operations which have a constant time complexity. Therefore, the time complexity for Algorithm 1 is (1+1)*n = O(n).

Algorithm 2 (Naive algorithm with Cycle shortcut): This algorithm includes the same components as Algorithm 1, with an additional if block inside for loop. Therefore, this algorithm's time complexity is for the worst case is also O(n), and it may differ up to O(1) for the best case.

Algorithm 3 (Recursive Algorithm): This algorithm is a recursive algorithm which calls itself with n/2 in each call. Therefore, the time complexity for this algorithm becomes log(n) with base 2. Since the constant value 2 in the base will be ignored when showing the time complexity with Big-O notation, time complexity can be shown as O(log(n)).

Specifications of the Computer

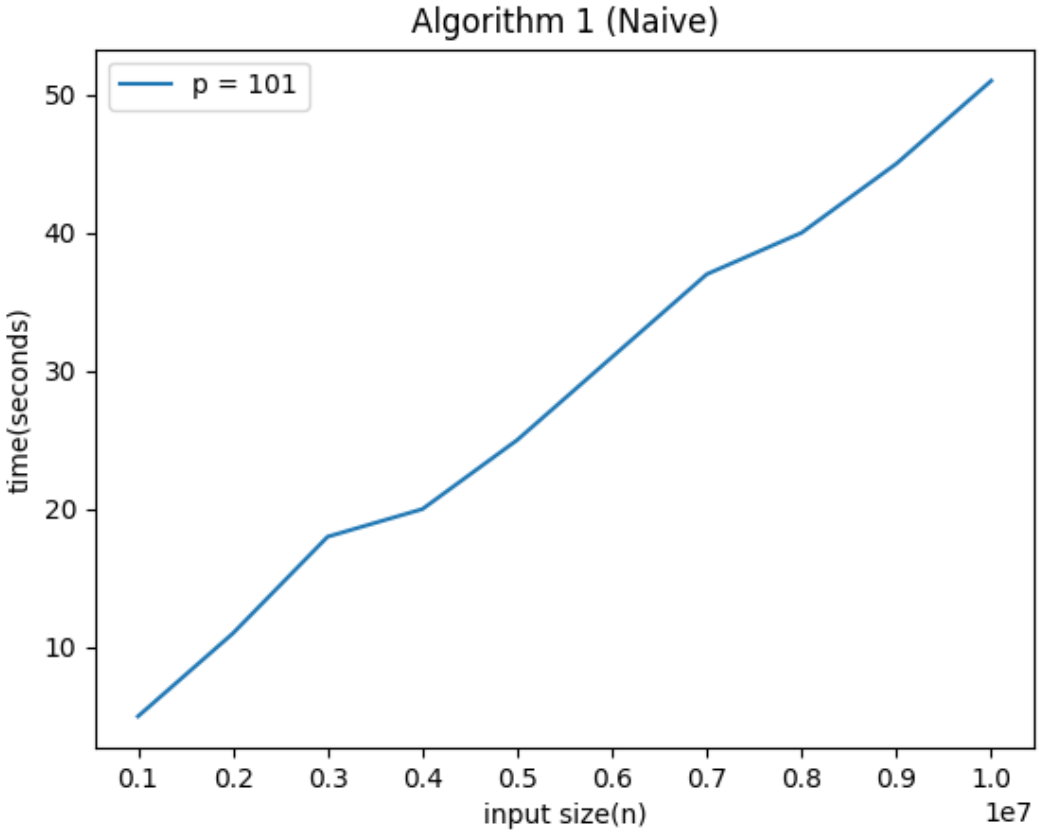
Processor: Intel(R) Core(TM) i7-3517U CPU @ 1.90GHz 2.40 GHz

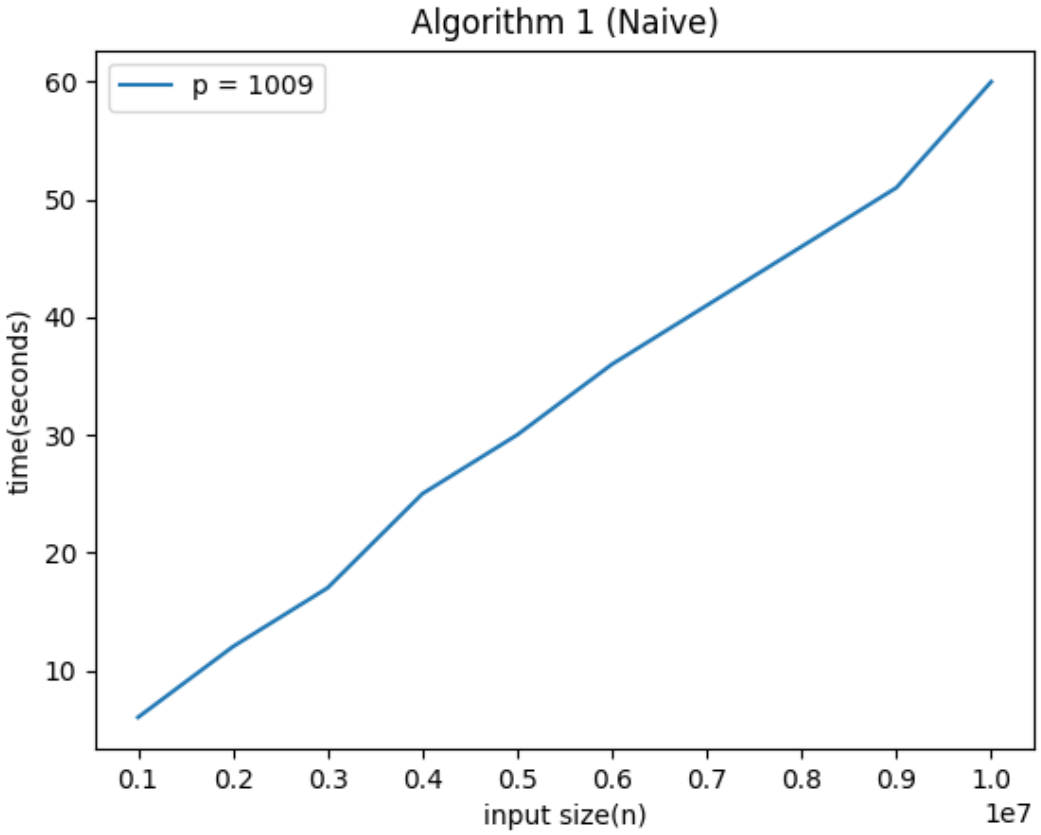
RAM: 4.00 GB RAM

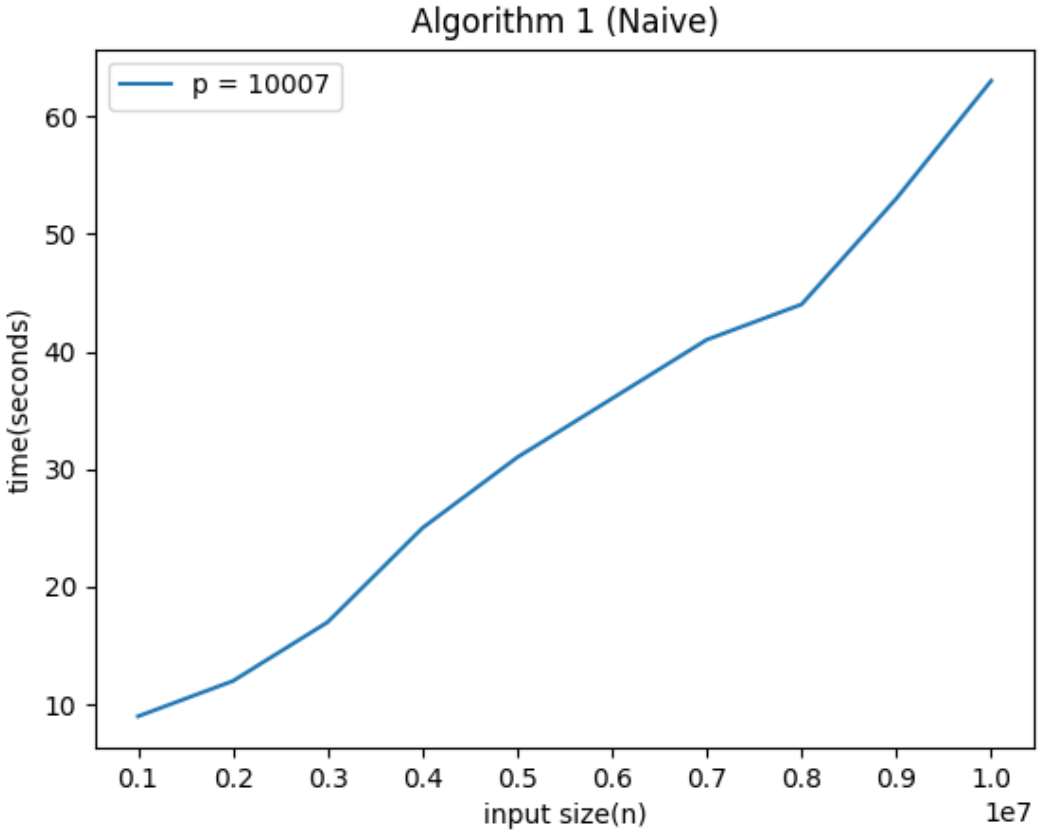
OS: Windows 10 Home Single Language

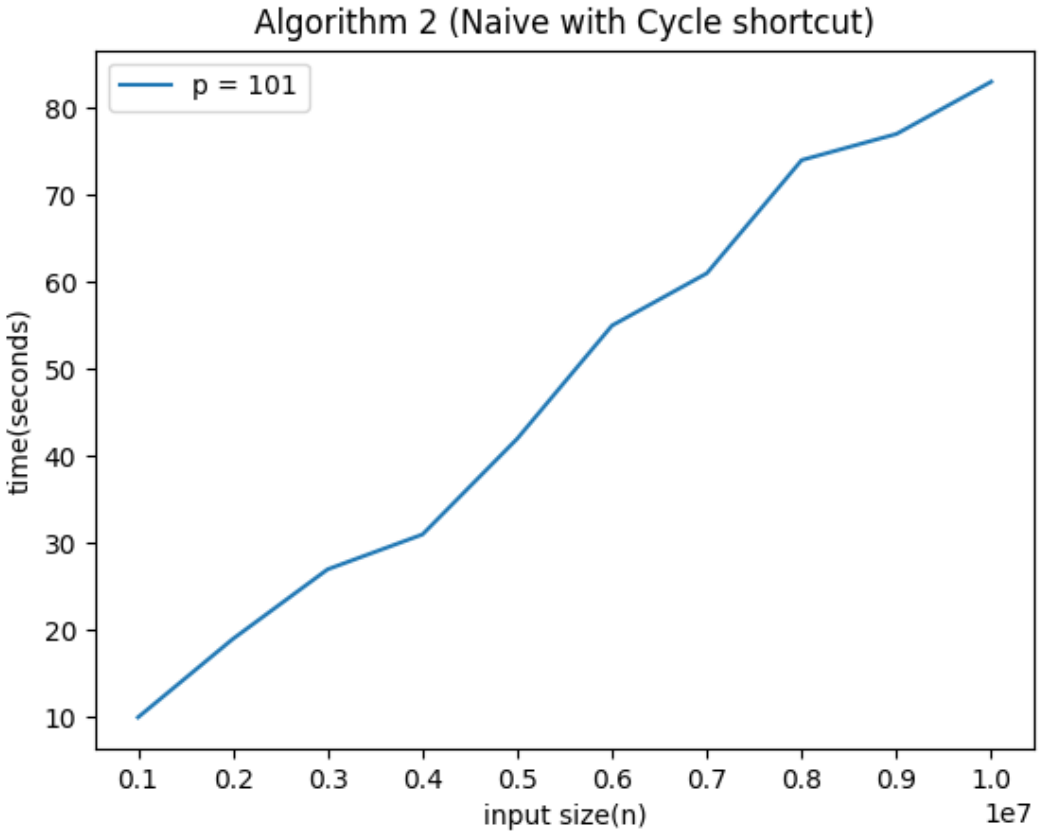
System Type: 64-bit operating system, x64-based processor

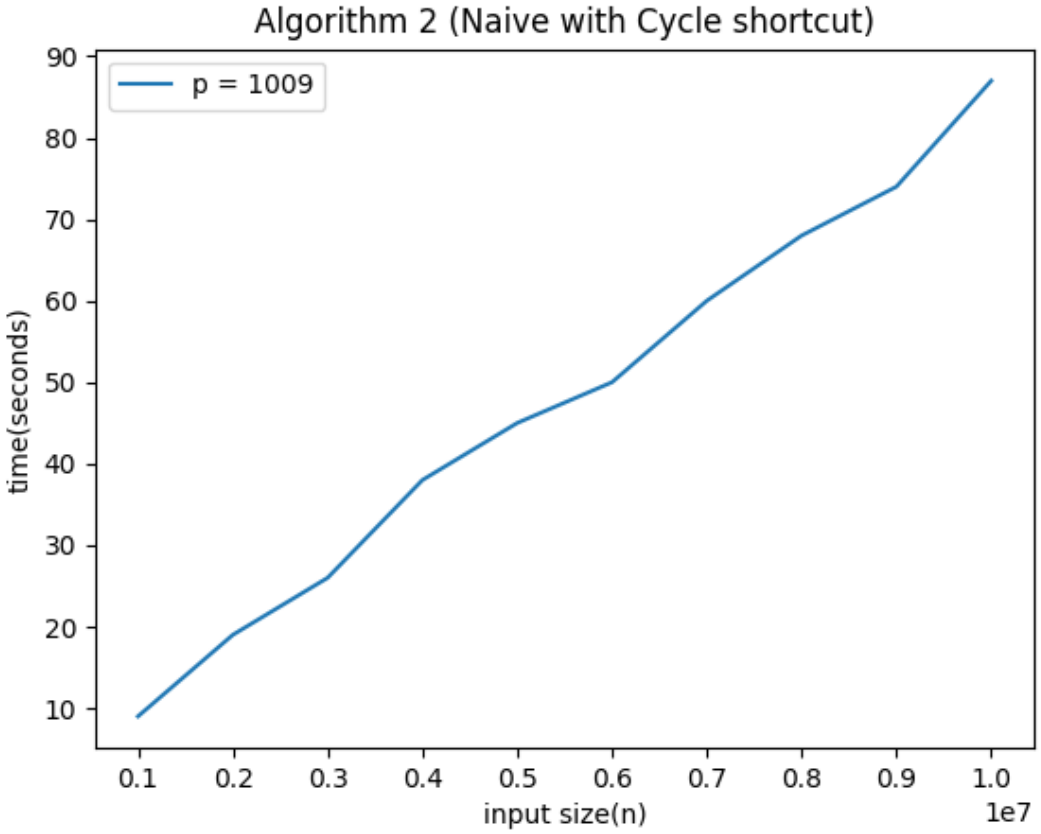
n	Algorithm 1			Algorithm 2			Algrotihm 3		
	p=101	p=1009	p=10007	p=101	p=1009	p=10007	p=101	p=1009	p=10007
10^6	5.67	6.46	9.75	10.97	9.99	10.52	0.00012	0.000154	0.000262
$2*10^6$	11.96	12.49	12.75	19.82	19.35	21.31	0.000124	0.00015	0.000223
$3*10^6$	18.83	17.81	17.37	27.95	26.47	26.02	0.000113	0.000164	0.000209
$4*10^6$	20.18	25.92	25.72	31.32	38.8	34.07	0.000135	0.000171	0.000137
$5*10^6$	25.2	30.24	31.39	42.43	45.31	49.73	0.000187	0.000107	0.000101
$6*10^6$	31.16	36.59	36.05	55.42	50.26	58.58	0.000149	0.000139	0.000158
$7*10^6$	37.93	41.26	41.40	61.5	60.55	61.65	0.00016	0.000114	0.000106
$8*10^6$	40.72	46.12	44.49	74.95	68.38	69.7	0.000167	0.000167	0.000152
$9*10^6$	45.28	51.41	53.84	77.43	74.29	76.01	0.000118	0.00018	0.000103
$10*10^6$	51.33	60.14	63.94	83.43	87.3	99.25	0.000117	0.00016	0.000128

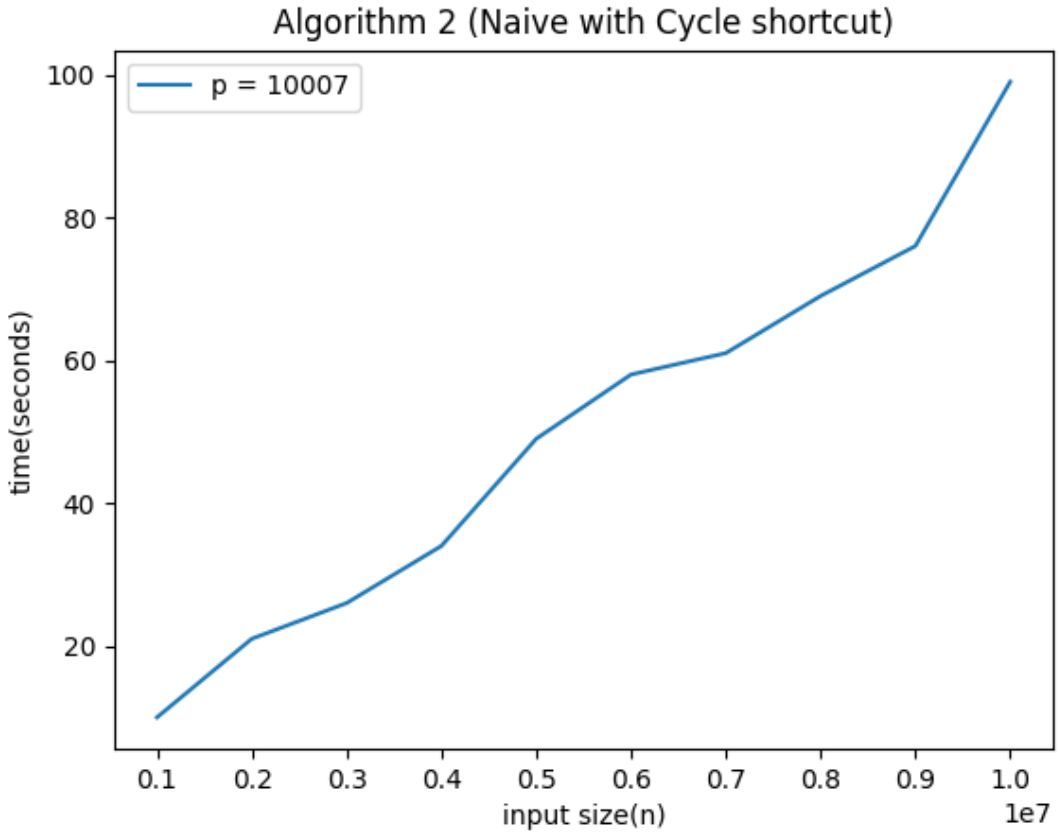

Algorithm 1 (Naive)
p = 101
time(seconds)
50
40
30
20
10
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
input size(n)
1e7

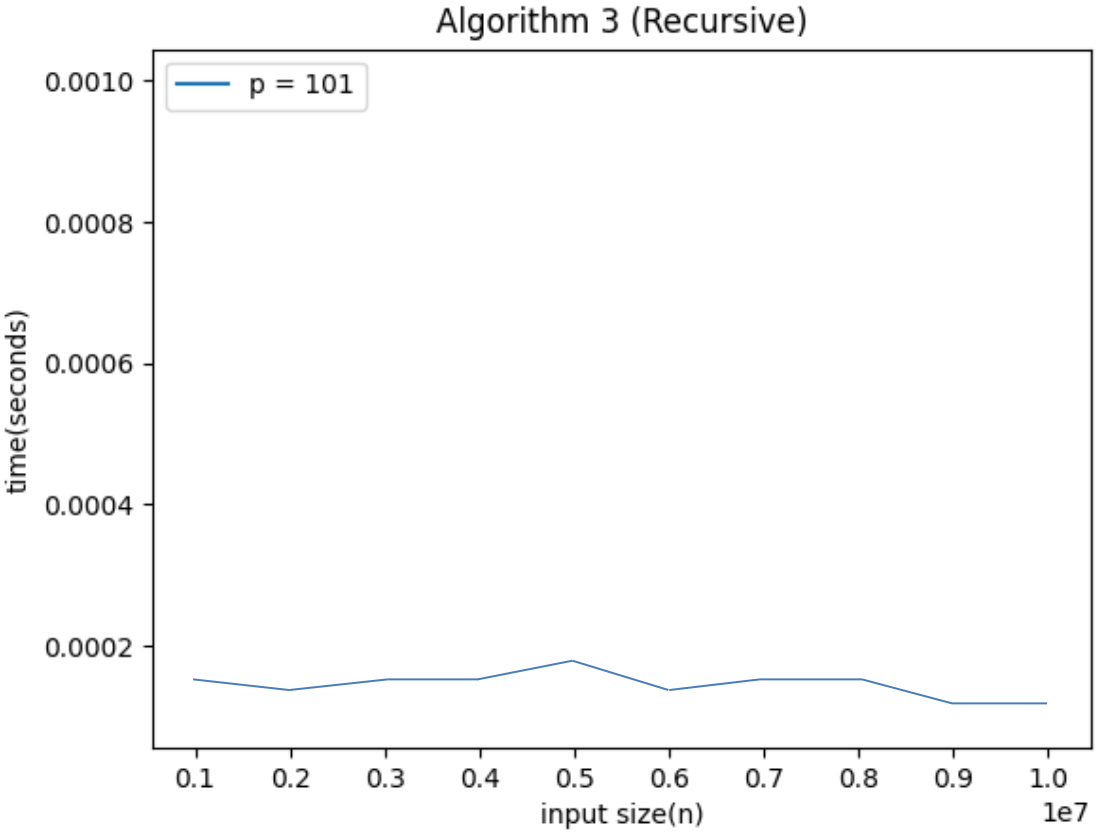

Algorithm 1 (Naive)
p = 1009
time(seconds)
60
50
40
30
20
10
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
input size(n)
1e7

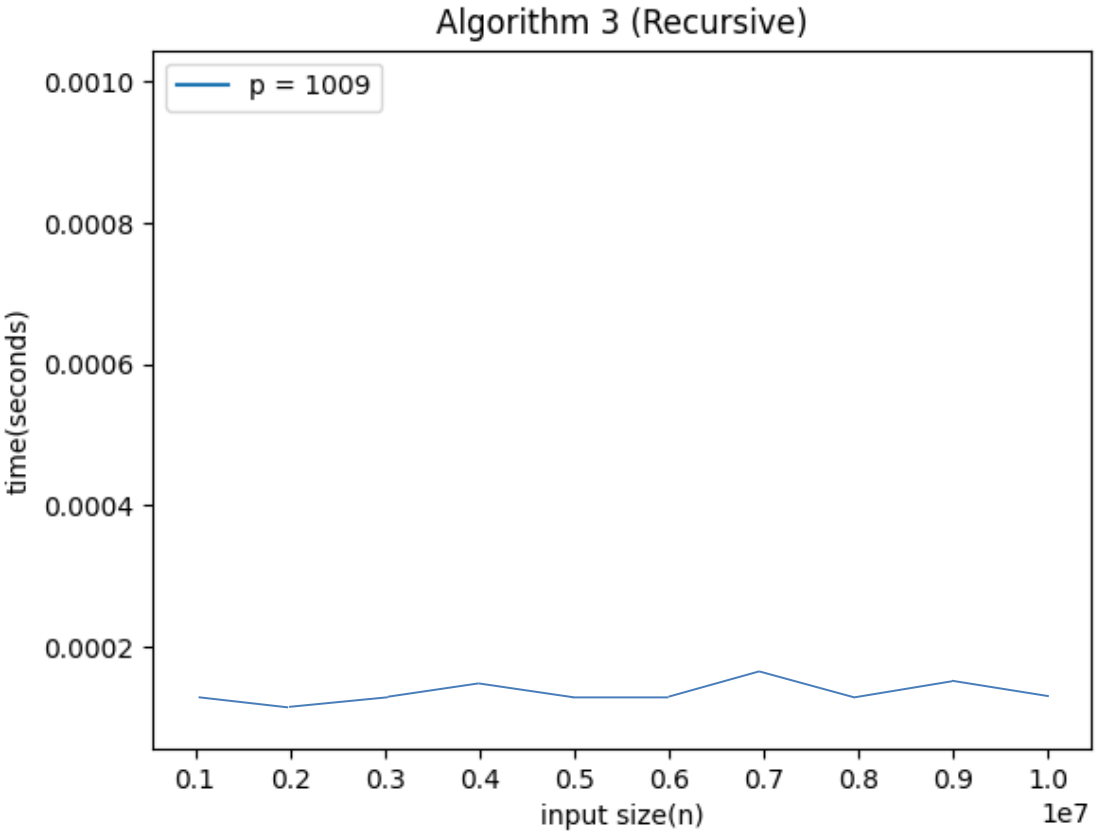

Algorithm 1 (Naive)
p = 10007
time(seconds)
10
20
30
40
50
60
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1e7
input size(n)

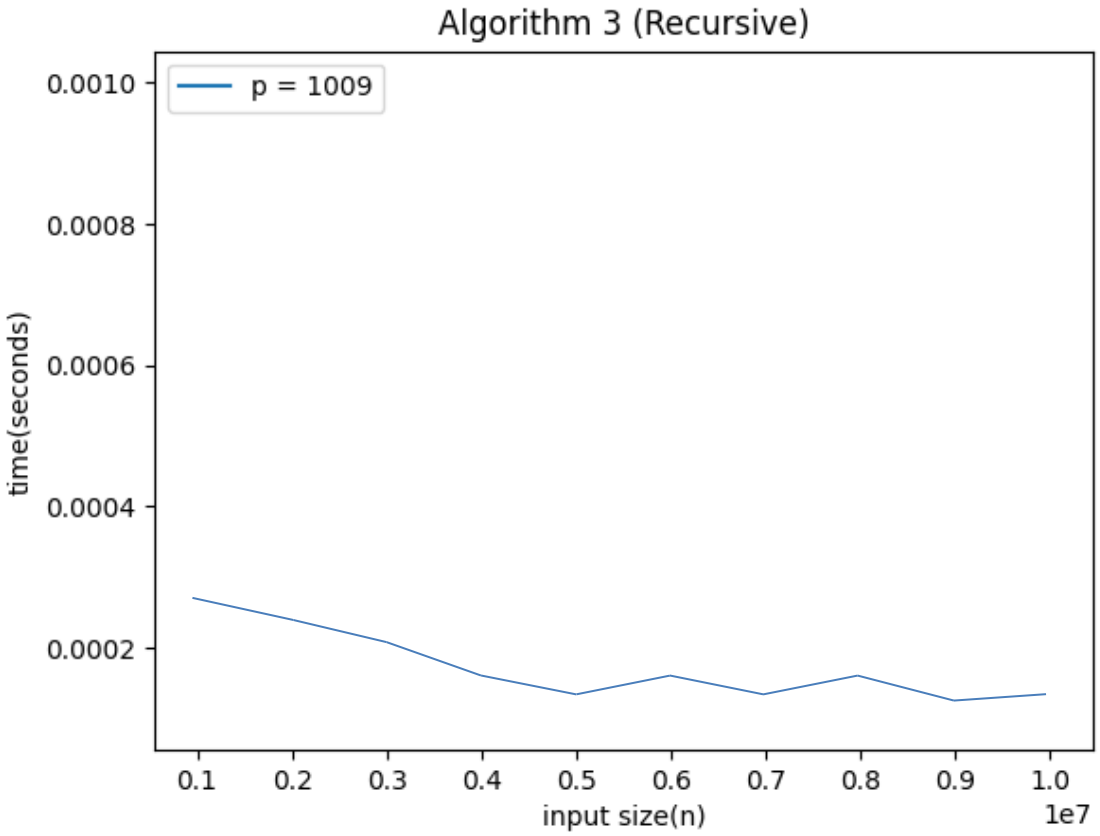

Algorithm 2 (Naive with Cycle shortcut)
p = 101
time(seconds)
10
20
30
40
50
60
70
80
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1e7
input size(n)


Algorithm 2 (Naive with Cycle shortcut)
p = 1009
time(seconds)
input size(n)
1e7


Algorithm 2 (Naive with Cycle shortcut)
p = 10007
time(seconds)
input size(n)
1e7


Algorithm 3 (Recursive)
p = 101
0.0010
0.0008
0.0006
0.0004
0.0002
time(seconds)
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
input size(n)
1e7


Algorithm 3 (Recursive)
p = 1009
0.0010
0.0008
0.0006
0.0004
0.0002
time(seconds)
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
input size(n)
1e7


Algorithm 3 (Recursive)
p = 1009
0.0010
0.0008
0.0006
0.0004
0.0002
time(seconds)
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
1e7
input size(n)