Algorithm Design

- Here we first pad the sequence to make it of perfect power of 2, which will be helpful in computing complete binary tree.
- Next compute a prefix sum as explained in class with OP being (A+B)%MAX.
- **Prefix sum:**We compute 2 trees
  - *B* -> Data Flow up
    - We start with the data at the leaves of the tree and compute the parent at each level.
    - Where the root is the result of OP on the child nodes.
  - *C*-> Data Flow down
    - Here we start by copying the root of tree *B* computed above to the
    - This is then used in next level to compute its children.
    - if the node index is 0 we copy corresponding value from tree *B*
    - if the node index is odd we copy value from previous level
    - if the node index is even we perform operartion on C[h+1][(i/2-1)] OP B[h][i]
    - We get the prefix array at the leaves of tree C
- Sorting:
  - Now we use bitonic sorting to sort the prefix sum array computed above at leaves of tree C.
  - As we do swaping in prefix sum array, we also perform the swapping on the original array.
  - In the end we bring the data from device to host using memcpy.

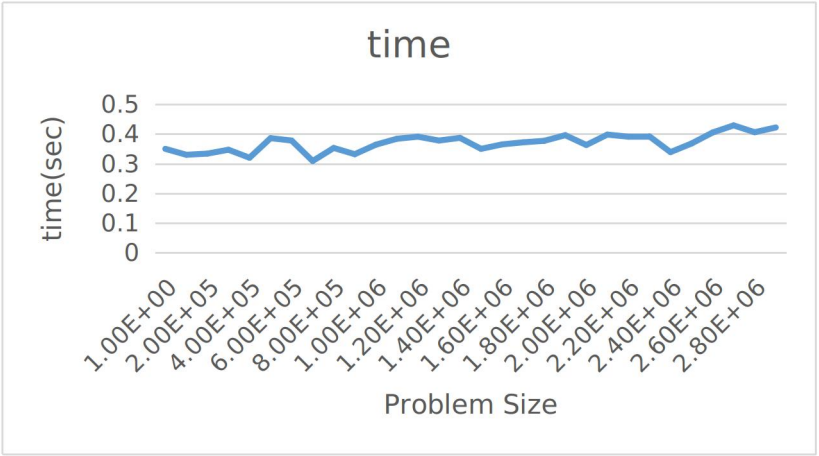
Timing Analysis
We do timing analysis using tables below.

- We see that when the problem size is very small, the time remains constant, which can be because of large initial setup time(0.3 sec).
- We see on very large size problem execution time becomes nearly linear with time.
- On doubling data size at every step, we see that initially when problem size is very small, execution time is nearly constant, and it becomes linear with size at the very high values, same is validated with log time graph which becomes linear in the end. in this table also in the last 5 entries we see that time doubles with size doubeling at each step.

From above we infer that initially when size is small, most of the time goes in setting up the problem, i,e, we get the GPU, copy data etc, so it remains constant. At larger problem sizes we see time increasing with problem size.

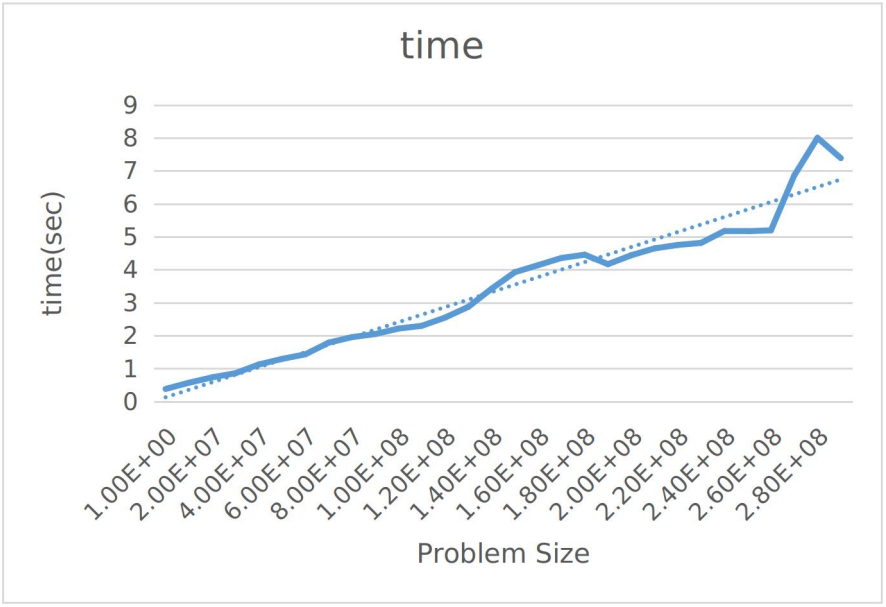
On Small Size data- nearly constant

size	time
1	0.35
100001	0.33
200001	0.334
300001	0.347
400001	0.32
500001	0.386
600001	0.378
700001	0.309
800001	0.353
900001	0.332
1000001	0.364
1100001	0.384
1200001	0.391
1300001	0.378
1400001	0.387
1500001	0.35
1600001	0.365
1700001	0.372
1800001	0.377
1900001	0.396
2000001	0.363
2100001	0.398
2200001	0.391
2300001	0.392
2400001	0.339
2500001	0.368
2600001	0.405
2700001	0.429
2800001	0.406
2900001	0.422



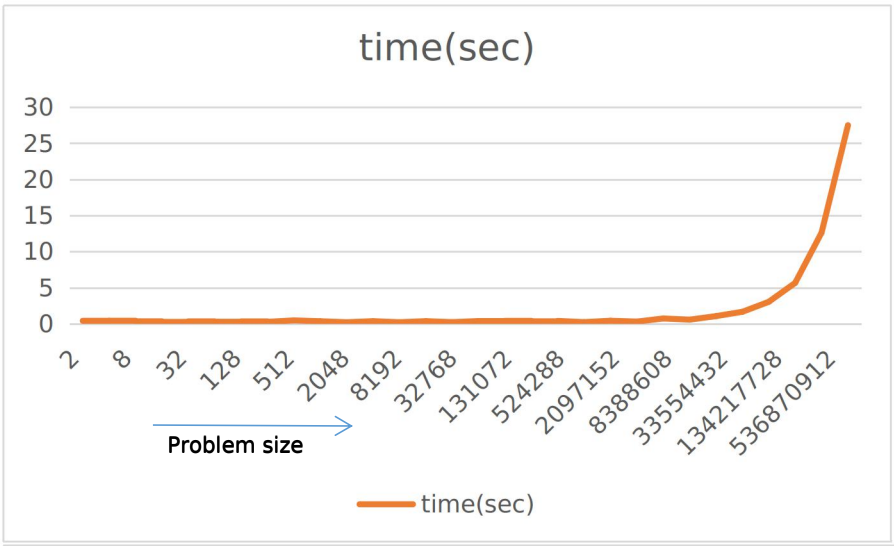
On Large Data Sizes - Linear

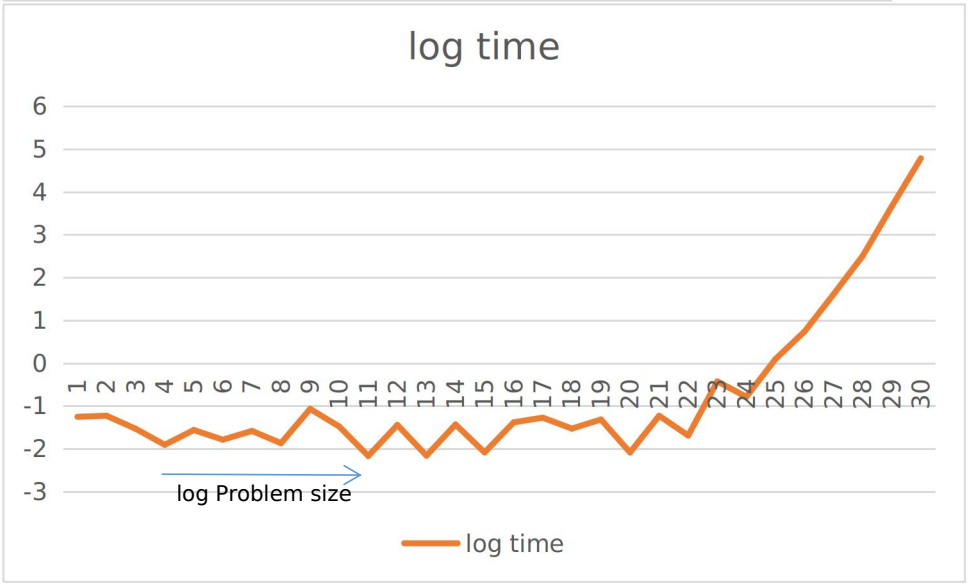
size	time
1	0.381
10000001	0.57
20000001	0.733
30000001	0.859
40000001	1.124
50000001	1.293
60000001	1.432
70000001	1.786
80000001	1.954
90000001	2.048
100000001	2.214
110000001	2.298
120000001	2.547
130000001	2.873
140000001	3.423
150000001	3.926
160000001	4.139
170000001	4.357
180000001	4.457
190000001	4.169
200000001	4.44
210000001	4.649
220000001	4.753
230000001	4.816
240000001	5.175
250000001	5.17
260000001	5.197
270000001	6.861
280000001	8.005
290000001	7.389



On exponential scale

size	time	$\log_2$ size	$\log_2$ time
2	0.42	1	-1.251538767
4	0.427	2	-1.227692025
8	0.346	3	-1.531156057
16	0.267	4	-1.905088353
32	0.339	5	-1.560642822
64	0.29	6	-1.785875195
128	0.334	7	-1.582079992
256	0.274	8	-1.867752202
512	0.477	9	-1.067938829
1024	0.359	10	-1.477944251
2048	0.223	11	-2.164884385
4096	0.368	12	-1.442222329
8192	0.224	13	-2.158429363
16384	0.371	14	-1.430508908
32768	0.236	15	-2.083141235
65536	0.384	16	-1.380821784
131072	0.414	17	-1.272297327
262144	0.347	18	-1.526992432
524288	0.402	19	-1.314732593
1048576	0.236	20	-2.083141235
2097152	0.427	21	-1.227692025
4194304	0.31	22	-1.689659879
8388608	0.744	23	-0.426625474
16777216	0.58	24	-0.785875195
33554432	1.069	25	0.096261853
67108864	1.669	26	0.738983955
134217728	3.054	27	1.610700062
268435456	5.676	28	2.504874589
536870912	12.643	29	3.66026693
1073741824	27.51	30	4.781884235


time(sec)
30
25
20
15
10
5
0
2
8
32
128
512
2048
8192
32768
131072
524288
2097152
8388608
33554432
134217728
536870912
Problem size
time(sec)


log time
6
5
4
3
2
1
0
-1
-2
-3
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
log Problem size
log time