Algorithm-1

Step	Cost of each execution	Total # of times executed
1	1	1
2	1	Q-P+2
3	1	$\frac{Q^2+3Q}{2}$
4	1	$\frac{Q^2+Q}{2}$
5	1	$\sum_{i=1}^{Q}\frac{i^2+3i}{2}$
6	6	$\sum_{i=1}^{Q}\frac{i^2+i}{2}$
7	7	$\frac{Q^2+Q}{2}$
8	1	1

Multiply col.1 with col.2, add across rows and simplify

$T_1(n) = 1 + Q - P + 2 + \frac{Q^2+3Q}{2} + \frac{Q^2+Q}{2} + \sum_{i=1}^{Q}\frac{i^2+3i}{2} + 6 * \sum_{i=1}^{Q}\frac{i^2+i}{2} + 7 * \frac{Q^2+Q}{2} + 1$

$$= n^2 + 3n + 4 + \sum_{i=1}^{Q}(2i + i^2)$$

$$= n^2 + 3n + 4 + n(n+1) + \frac{n(n+1)(2n+1)}{2}$$

$$= \frac{n^3}{3} + \frac{5n^2}{2} + \frac{25n}{6} + 4 = O(n^3)$$

Algorithm-2

Step	Cost of each execution	Total # of times executed
1	1	1
2	1	Q-P+2
3	1	Q-P+1
4	1	$\frac{Q^2+3Q}{2}$
5	6	$\frac{Q^2+Q}{2}$
6	7	$\frac{Q^2+Q}{2}$
7	1	1

Multiply col.1 with col.2, add across rows and simplify

$T_2(n) = 1 + Q - P + 2 + Q - P + 1 + \frac{Q^2+3Q}{2} + 6 * \frac{Q^2+Q}{2} + 7 * \frac{Q^2+Q}{2} + 1 = 7Q^2 + 10Q - 2P + 5$

$$= 7n^2 + 5 = O(n^2)$$

Algorithm-3

Step	Cost of each execution	Total # of times executed in any single recursive call
1	4	1
2	4	1
Steps executed when the input is a base case:1or2		
First recurrence relation: T(n=1 or n=0) =4		
3		1
4	1	1

5	1	M-L+2
6	6	M-L+1
7	7	M-L+1
8	1	1
9	1	U-M+1
10	6	U-M
11	7	U-M
12	3	1
13	T(n/2)	(cost excluding the recursive call)
14	T(n/2)	(cost excluding the recursive call)
15	5	1
Steps executed when input is NOT a base case: all steps		
Second recurrence relation: T(n>1) = T(n/2) +T(n/2)+4		
Simplified second recurrence relation (ignore the constant term): T(n>1) = 2T(n/2)+O(1)		

Solve the two recurrence relations using any method (recommended method is the Recursion Tree). Show your work below:

```
Level 0                  T(n)
                      /          \
Level 1          T(n/2)          T(n/2)
               /       \        /        \
Level 2    T(n/4)   T(n/4)   T(n/4)   T(n/4)
```

$T_3(n) = 2T(n/2) + 4$

$= T(2T(n/4)+4)+4$

$T(n) = c + 2c + 4c + \ldots\ldots + nc$

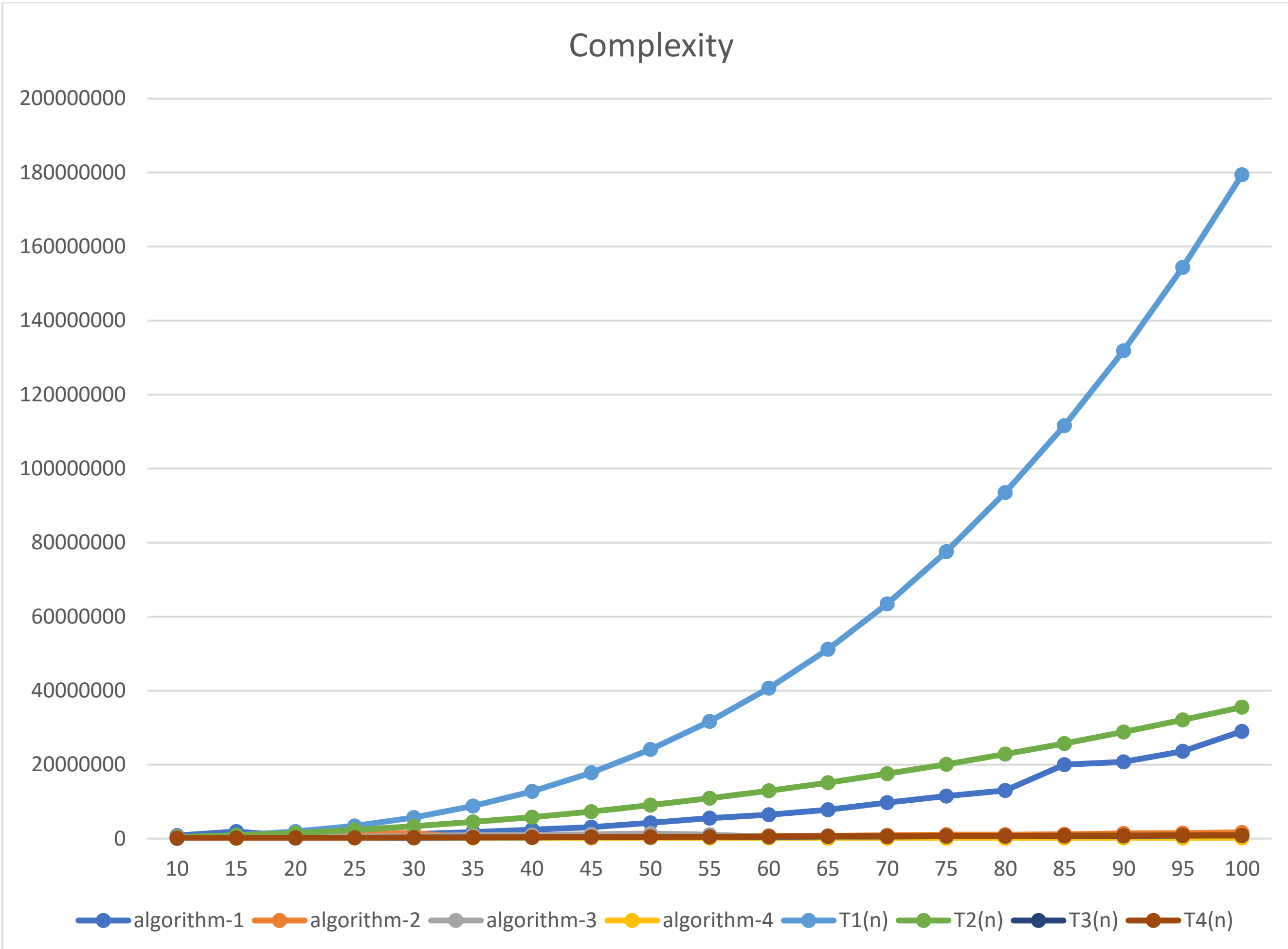
$=O(n)$

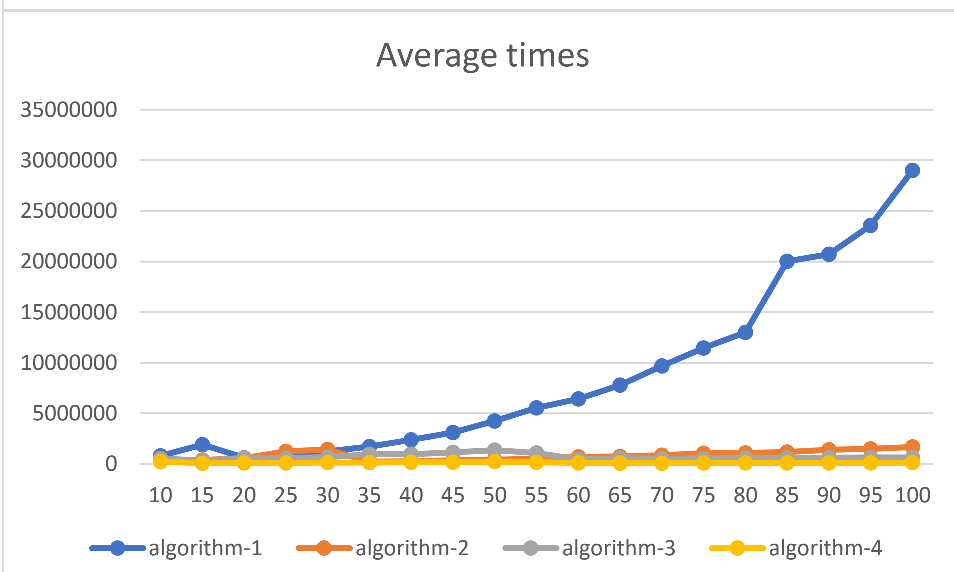
Algorithm-4

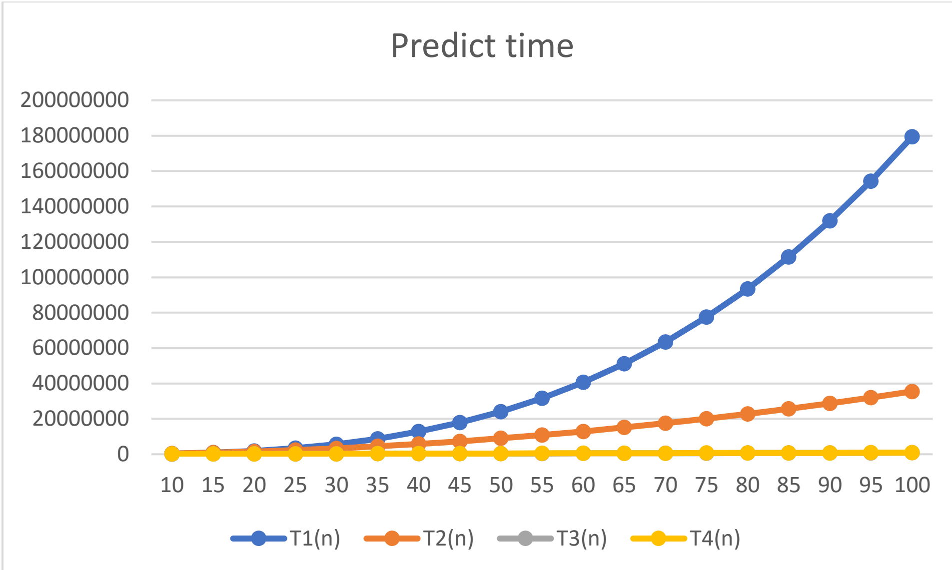
Step	Cost of each execution	Total # of times executed
1	1	1
2	1	1
3	1	Q-P+2
4	10	Q-P+1
5	7	Q-P+1
6	1	1

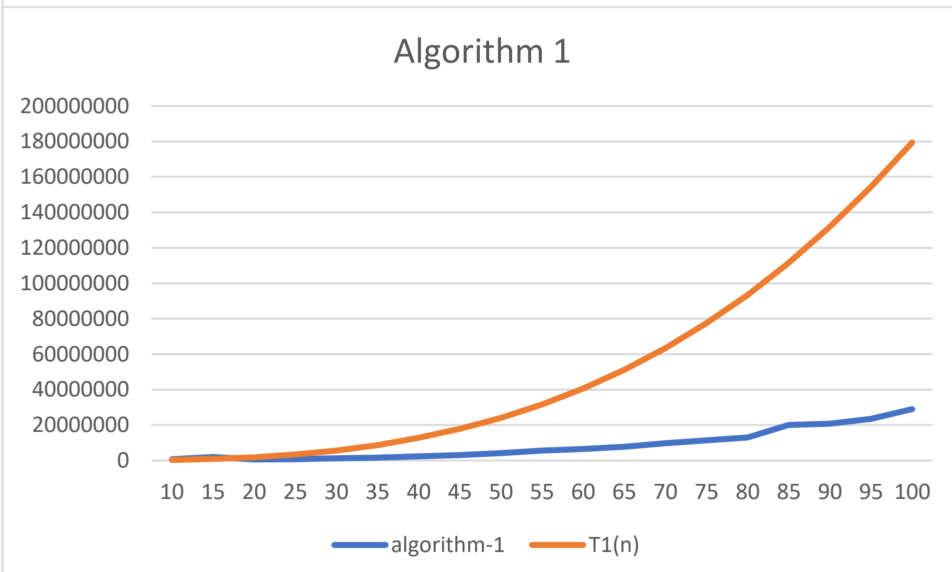
Multiply col.1 with col.2, add across rows and simplify

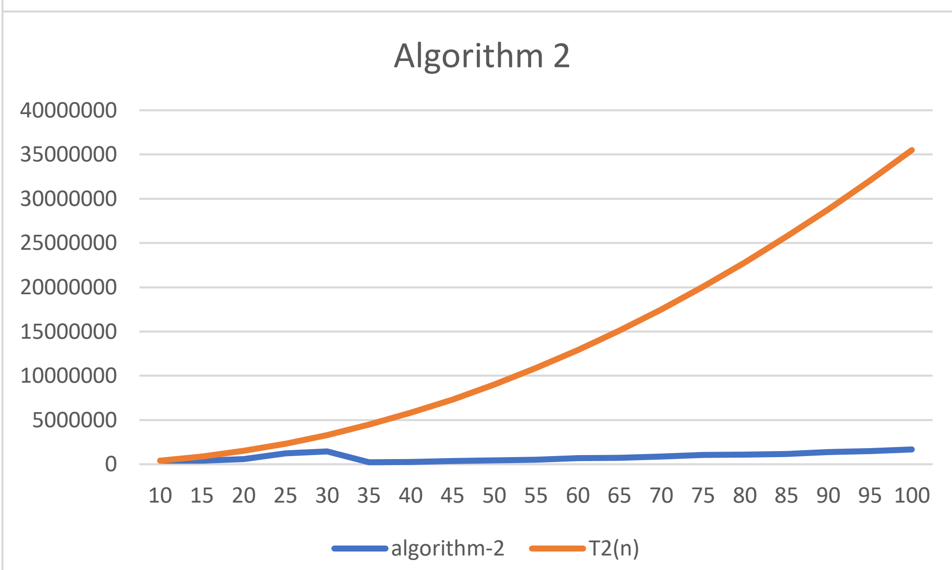
$T_4(n) = 1 + 1 + Q - P + 2 + 10 * (Q - P + 1) + 7 * (Q - P + 1) + 1 = 18Q - 18P + 22 = O(n)$

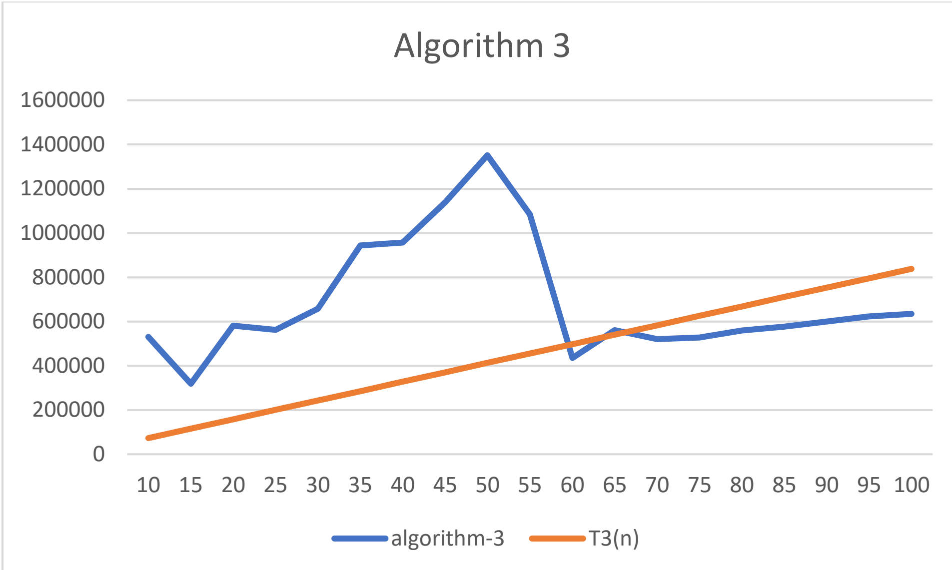

Complexity
200000000
180000000
160000000
140000000
120000000
100000000
80000000
60000000
40000000
20000000
0
10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
algorithm-1 algorithm-2 algorithm-3 algorithm-4 T1(n) T2(n) T3(n) T4(n)

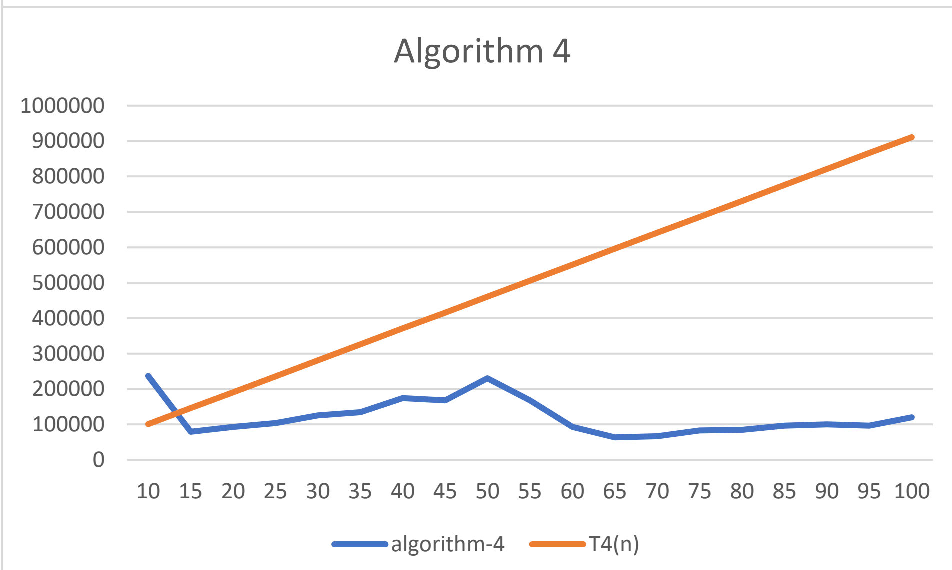

Average times
35000000
30000000
25000000
20000000
15000000
10000000
5000000
0
10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
algorithm-1 algorithm-2 algorithm-3 algorithm-4


Predict time
200000000
180000000
160000000
140000000
120000000
100000000
80000000
60000000
40000000
20000000
0
10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
T1(n)
T2(n)
T3(n)
T4(n)


Algorithm 1
200000000
180000000
160000000
140000000
120000000
100000000
80000000
60000000
40000000
20000000
0
10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
algorithm-1
T1(n)


Algorithm 2
40000000
35000000
30000000
25000000
20000000
15000000
10000000
5000000
0
10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
algorithm-2
T2(n)





In conclusion, the predicted time complexity usually larger than the real average time. The predictions fit approximate the average time cost. But the difference between prediction and real cost is too large, especially in algorithm 1 and algorithm 2. Therefore, the result is not very reliable and need more accuracy.