Table of Contents

1 THEORETICAL ANALYSIS

1.1 Basic operation is the comparison marked as (1)

Analyze B(n)

Let Tn be set of all inputs of size n and let τ(X) be number of basic operations for input X ∈ Tn.

For all X ∈ Tn, there is $\sum_{i=0}^{n-1} 1$ comparison operation. Because X[i]=0 is a common operation for all inputs and it is iterated $\sum_{i=0}^{n-1} 1$ times. Therefore τ(X) is same for all X ∈ Tn and all X ∈ Tn is the best case input for this algorithm.

$$B(n) = \sum_{i=0}^{n-1} 1 = n \in \Theta(n)$$

Analyze W(n)

Let Tn be set of all inputs of size n and let τ(X) be number of basic operations for input X ∈ Tn.

For all X ∈ Tn, there is $\sum_{i=0}^{n-1} 1$ comparison operation. Because X[i]=0 is a common operation for all inputs and it is iterated $\sum_{i=0}^{n-1} 1$ times. Therefore τ(X) is same for all X ∈ Tn and all X ∈ Tn is the worst case input for this algorithm.

$$W(n) = \sum_{i=0}^{n-1} 1 = n \in \Theta(n)$$

Analyze A(n)

Let Tn be set of all inputs of size n and let τ(X) be number of basic operations for input X ∈ Tn.

For all X ∈ Tn, there is $\sum_{i=0}^{n-1} 1$ comparison operation. Because X[i]=0 is a common operation for all inputs and it is iterated $\sum_{i=0}^{n-1} 1$ times. Therefore τ(X) is same for all X ∈ Tn and all X ∈ Tn is the average case input for this algorithm.

$$A(n) = \sum_{i=0}^{n-1} 1 = n \in \Theta(n)$$

1.2 Basic operations are the two loop incrementations marked as (2)

Analyze B(n)

Let Tn be set of all inputs of size n and let τ(X) be number of basic operations for input X ∈ Tn.

Regardless of the X[i] is being 0 or 1, in each iteration of the outer for loop, the algorithm will do "(2)" numbered operation (n-i) times until the outer for loop finishes. This gives us $\sum_{i=0}^{n-1}(n-i)$ operations for all X ∈ Tn. Therefore τ(X) is same for all X ∈ Tn and all X ∈ Tn is the best case input for this algorithm.

$$B(n) = \sum_{i=0}^{n-1}(n-i) = n + (n-1) + \dots + 1 = \frac{n(n+1)}{2} = \frac{n^2+n}{2} \in \Theta(n^2)$$

Analyze W(n)

Let Tn be set of all inputs of size n and let τ(X) be number of basic operations for input X ∈ Tn.

Regardless of the X[i] is being 0 or 1, in each iteration of the outer for loop, the algorithm will do "(2)" numbered operation (n-i) times until the outer for loop finishes. This gives us $\sum_{i=0}^{n-1}(n-i)$ operations for all X ∈ Tn. Therefore τ(X) is same for all X ∈ Tn and all X ∈ Tn is the worst case input for this algorithm.

$$W(n) = \sum_{i=0}^{n-1}(n-i) = n + (n-1) + \dots + 1 = \frac{n(n+1)}{2} = \frac{n^2+n}{2} \in \Theta(n^2)$$

Analyze A(n)

Let Tn be set of all inputs of size n and let τ(X) be number of basic operations for input X ∈ Tn.

Regardless of the X[i] is being 0 or 1, in each iteration of the outer for loop, the algorithm will do "(2)" numbered operation (n-i) times until the outer for loop finishes. This gives us $\sum_{i=0}^{n-1}(n-i)$ operations for all X ∈ Tn. Therefore τ(X) is same for all X ∈ Tn and all X ∈ Tn is the average case input for this algorithm. Also we know that B(n) = W(n). Therefore A(n) = W(n) = B(n).

$$A(n) = \sum_{i=0}^{n-1}(n-i) = n + (n-1) + \dots + 1 = \frac{n(n+1)}{2} = \frac{n^2+n}{2} \in \Theta(n^2)$$

1.3 Basic operation is the assignment marked as (3)

Analyze B(n)

At each iteration i, if X[i]=1 then 0 basic operations. So the best case input is,

Best case input: X[i]=1, for all i ∈ Z and 0<=i<=n-1

$$B(n) = \sum_{i=0}^{n-1} 0 = 0 \in \Theta(1)$$

Analyze W(n)

At each iteration i, if X[i]=0 then algorithm executes “(3)” numbered operation for each member of the input array X. So the worst case input is,

$Worst\ case\ input$:X[i]=0, for all i ∈ Z and 0<=i<=n-1

W(n) = $\sum_{i=0}^{n-1}\sum_{j=i}^{n-1}\sum_{z=1}^{\lfloor\frac{logn}{log2}\rfloor+1} 1$ = $\sum_{i=0}^{n-1}\sum_{j=i}^{n-1}(\lfloor log_2 n\rfloor + 1)$

$= \sum_{i=0}^{n-1}((n - i) * (\lfloor log_2 n\rfloor + 1)) = (\lfloor log_2 n\rfloor + 1) * \sum_{i=0}^{n-1}(n - i) = (\lfloor log_2 n\rfloor + 1) * (\frac{n^2+n}{2}) = (n^2\lfloor log_2 n\rfloor + n^2 + n\lfloor log_2 n\rfloor + n)/2$

We have $(n^2\lfloor log_2 n\rfloor)$ $as\ highest\ order\ term\ and\ we\ also\ know\ that\ floor\ function\ is\ \Theta(1).\ Therefore,$

$W(n) \in \Theta(n^2 logn)$

Analyze A(n)

Probability of X[i]=0 is 1/3, and X[i]=1 is 2/3 for all i ∈ Z and 0<=i<=n-1

$$A(n) = \sum_{i=0}^{n-1}(\frac{1}{3} * \sum_{j=i}^{n-1}\sum_{z=1}^{\lfloor\frac{logn}{log2}\rfloor+1} 1 + \frac{2}{3} * 0)$$

$= \sum_{i=0}^{n-1}(\frac{1}{3}(n - i) * (\lfloor log_2 n\rfloor + 1) + \frac{2}{3} * 0) = \frac{1}{3}(\lfloor log_2 n\rfloor + 1)\sum_{i=0}^{n-1}(n - i)$

$= (\lfloor log_2 n\rfloor + 1) * (\frac{n^2+n}{6}) = (n^2\lfloor log_2 n\rfloor + n^2 + n\lfloor log_2 n\rfloor + n)/6$

We have $(n^2\lfloor log_2 n\rfloor)$ $as\ highest\ order\ term\ and\ we\ also\ know\ that\ floor\ function\ is\ \Theta(1).\ Therefore,$

$A(n) \in \Theta(n^2 logn)$

1.4 Basic operations are the two assignments marked as (4)

Analyze B(n)

Basic operation (2) is in 2 places in the algorithm. First one is in the "if part" (the first occurrence of basic operation), and the second one is in the "else part"(the second occurrence of the basic operation). We know the complexity of "if part" of the algorithm already. It is $\Theta(n^2 logn)$. The complexity of "else part" of the algorithm is greater than n^3 because even if it does not enter the while loop it has n^3complexity already. Therefore if X[i] = 0 for all elements of input array X then we have a best case input.

$Best\ case\ input: X[i] = 0,\ for\ all\ i \in Z\ and\ 0 <= i <= n$-1

B(n) = $\sum_{i=0}^{n-1}\sum_{j=i}^{n-1}\sum_{z=1}^{\lfloor\frac{logn}{log2}\rfloor+1} 1$ = $\sum_{i=0}^{n-1}\sum_{j=i}^{n-1}(\lfloor log_2 n\rfloor + 1)$

$= \sum_{i=0}^{n-1}((n - i) * (\lfloor log_2 n\rfloor + 1)) = (\lfloor log_2 n\rfloor + 1) * \sum_{i=0}^{n-1}(n - i) = (\lfloor log_2 n\rfloor + 1) * (\frac{n^2+n}{2}) = (n^2\lfloor log_2 n\rfloor + n^2 + n\lfloor log_2 n\rfloor + n)/2$

We have $(n^2\lfloor log_2 n\rfloor)$ *as highest order term and we also know that floor function is* $\Theta(1)$. *Therefore,*

$B(n) \in \Theta(n^2 logn)$

Analyze W(n)

We proved above that the complexity of the "else part" (second occurrence of the basic operation) of the algorithm is higher than the complexity of "if part" (first occurrence of the basic operation) of the algorithm. Therefore if X[i] = 1 for all elements of input array X then we have a worst case input.

$Worst\ case\ input: X[i] = 1,\ for\ all\ i \in Z\ and\ 0 <= i <= n$-1

$$W(n) = \sum_{i=0}^{n-1}\sum_{j=i}^{n-1}\sum_{t=1}^{n}\sum_{z=1}^{\lceil n/t\rceil} 1 = \sum_{i=0}^{n-1}\sum_{j=i}^{n-1}\sum_{t=1}^{n}\lceil n/t\rceil$$

$= \sum_{i=0}^{n-1}((n - i) * \sum_{t=1}^{n}\lceil n/t\rceil) = \sum_{t=1}^{n}\lceil n/t\rceil * \sum_{i=0}^{n-1}(n - i)$

For asymptotic analysis of W(n):

W(n) <= $\sum_{t=1}^{n}(n/t + 1) * \sum_{i=0}^{n-1}(n - i)$ = (n+n* $\sum_{t=1}^{n}(1/t)$) * $\sum_{i=0}^{n-1}(n - i)$

If we use integral method to examine $\sum_{t=1}^{n}(1/t)$,

$\int_{1}^{n+1}\frac{1}{t}\,dt <= \sum_{t=1}^{n}(1/t)$ <= 1 + $\int_{1}^{n}\frac{1}{t}\,dt$

ln(n+1) <= $\sum_{t=1}^{n}(1/t)$ <= 1 + ln(n),

(n+n* $\sum_{t=1}^{n}(1/t)$) * $\sum_{i=0}^{n-1}(n - i)$ <= (n+n*$(1 + (ln(n))$ * $\sum_{i=0}^{n-1}(n - i)$ = (n + n + n*ln(n)) * $\sum_{i=0}^{n-1}(n - i)$

= (2n + n*ln(n)) * $(\frac{n^2+n}{2})$ = $(n^3 * ln(n) + n^2 * ln(n) + 2n^3 + 2n^2)$ / 2

Therefore,

$W(n) = \sum_{i=0}^{n-1}((n - i) * \sum_{t=1}^{n}\lceil n/t\rceil) = \sum_{t=1}^{n}\lceil n/t\rceil * \sum_{i=0}^{n-1}(n - i) \in O(n^3 logn)$

$W(n) = \sum_{t=1}^{n} \lceil n/t \rceil * \sum_{i=0}^{n-1} (n - i) \Rightarrow \sum_{t=1}^{n} n/t * \sum_{i=0}^{n-1} (n - i) = n * \sum_{t=1}^{n} 1/t * \sum_{i=0}^{n-1} (n - i)$

If we use integral method to examine $\sum_{t=1}^{n} (1/t)$,

$\int_{1}^{n+1} \frac{1}{t}\, dt <= \sum_{t=1}^{n} (1/t) <= 1 + \int_{1}^{n} \frac{1}{t}\, dt$

ln(n+1) <= $\sum_{t=1}^{n} (1/t)$ <= 1 + ln(n),

$n * \sum_{t=1}^{n} 1/t * \sum_{i=0}^{n-1} (n - i)$ => n*ln(n+1)* $\sum_{i=0}^{n-1} (n - i)$ = n*ln(n+1)*($\frac{n^2+n}{2}$)

= (n^3* ln(n+1) + n^2* ln(n+1)) / 2

Therefore,

W(n) ∈ $\Omega(n^3 logn)$

We also know that W(n) ∈ $O(n^3 logn)$

This implies that W(n) ∈ $\Theta(n^3 logn)$

Analyze A(n)

Probability of X[i]=0 is 1/3, and X[i]=1 is 2/3 for all i ∈ Z and 0<=i<=n-1

$$A(n) = \sum_{i=0}^{n-1} \left(\frac{1}{3} * \left(\sum_{j=i}^{n-1} \sum_{z=1}^{\lfloor \frac{logn}{log2} \rfloor + 1} 1\right) + \frac{2}{3} * \left(\sum_{j=i}^{n-1} \sum_{t=1}^{n} \sum_{z=1}^{\lceil n/t \rceil} 1\right)\right)$$

$$= \sum_{i=0}^{n-1} \left(\frac{1}{3}(n - i)(\lfloor log_2 n \rfloor + 1) + \frac{2}{3}(n - i) * \sum_{t=1}^{n} \lceil n/t \rceil \right)$$

For asymptotic analysis of A(n):

A(n) => $\sum_{i=0}^{n-1} \left(\frac{1}{3}(n - i)(\lfloor log_2 n \rfloor + 1) + \frac{2}{3}(n - i) * \sum_{t=1}^{n} n/t \right) =$

$\sum_{i=0}^{n-1} \left(\frac{1}{3}(n - i)(\lfloor log_2 n \rfloor + 1) + \frac{2}{3}(n - i) * (n) \sum_{t=1}^{n} 1/t \right)$

If we use integral method to examine $\sum_{t=1}^{n} (1/t)$,

$\int_{1}^{n+1} \frac{1}{t}\, dt$ <= $\sum_{t=1}^{n} (1/t)$ <= $1 + \int_{1}^{n} \frac{1}{t}\, dt$

ln(n+1) <= $\sum_{t=1}^{n} (1/t)$ <= 1 + ln(n)

$\sum_{i=0}^{n-1} \left(\frac{1}{3}(n-i)(\lfloor log_2 n \rfloor + 1) + \frac{2}{3}(n-i) * (n) \sum_{t=1}^{n} 1/t \right)$ =>
$\sum_{i=0}^{n-1} \left(\frac{1}{3}(n-i)(\lfloor log_2 n \rfloor + 1) + \frac{2}{3}(n-i) * (n) * ln(n+1) \right)$

=
$\sum_{i=0}^{n-1} (n-i)\left(\frac{1}{3}(\lfloor log_2 n \rfloor + 1) + \frac{2}{3}(n * ln(n+1)) \right) = \left(\frac{1}{3}(\lfloor log_2 n \rfloor + 1) + \frac{2}{3}(n * ln(n+1)) \right) * \sum_{i=0}^{n-1}(n-i)$

= $\left(\frac{1}{3}(\lfloor log_2 n \rfloor + 1) + \frac{2}{3}(n * ln(n+1)) \right) * \left(\frac{n^2+n}{2} \right)$

= $(n^2 * \lfloor log_2 n \rfloor + n * \lfloor log_2 n \rfloor + n^2 + n + 2 * n^3 *$ ln(n+1) + $2 * n^2 * ln(n+1)) / 6$

Therefore,

$A(n) = \sum_{i=0}^{n-1} \left(\frac{1}{3}(n-i)(\lfloor log_2 n \rfloor + 1) + \frac{2}{3}(n-i) * \sum_{t=1}^{n} \lceil n/t \rceil \right) \in \Omega(n^3 logn)$

We know that for any algorithm A(n) <= W(n) and we also know that W(n) ∈ $O(n^3 logn)$.

Therefore,

$A(n) \in O(n^3 logn)$

We know that A(n) ∈ $O(n^3 logn)$ and A(n) ∈ $\Omega(n^3 logn)$, this implies that,

A(n) ∈ $\Theta(n^3 logn)$

2 IDENTIFICATION OF BASIC OPERATION(S)

Here, state clearly which operation(s) in the algorithm must be the basic operation(s). Also, you should provide a simple explanation about why you have decided on the basic operation you choose. (1-3 sentences)
Answer:

The operations marked as (4) must be the basic operations. Because in any version of this algorithm – rather this be for, while, recursion etc. version – this repetitive incrementation operation will be there. In short words, operations marked as (4) are characteristic and repetitive operations in this algorithm

3 REAL EXECUTION

3.1 Best Case

N Size	Time Elapsed (in seconds)
1	0.00000286102294921875
10	0.0000181198120117 1875
50	0.0005769729614257812
100	0.00279188156127929 7
200	0.012072086334228516
300	0.03457212448120117
400	0.06287527084350586
500	0.10094308853149414
600	0.1611940860748291
700	0.21352696418762207

3.2 Worst Case

N Size	Time Elapsed (in seconds)
1	0.0000021457672119140625
10	0.00015091896057128906
50	0.02344822883605957
100	0.20537400245666504
200	1.8039209842681885
300	7.507683038711548
400	19.063247203826904
500	38.98806095123291
600	69.85111021995544
700	114.84602427482605

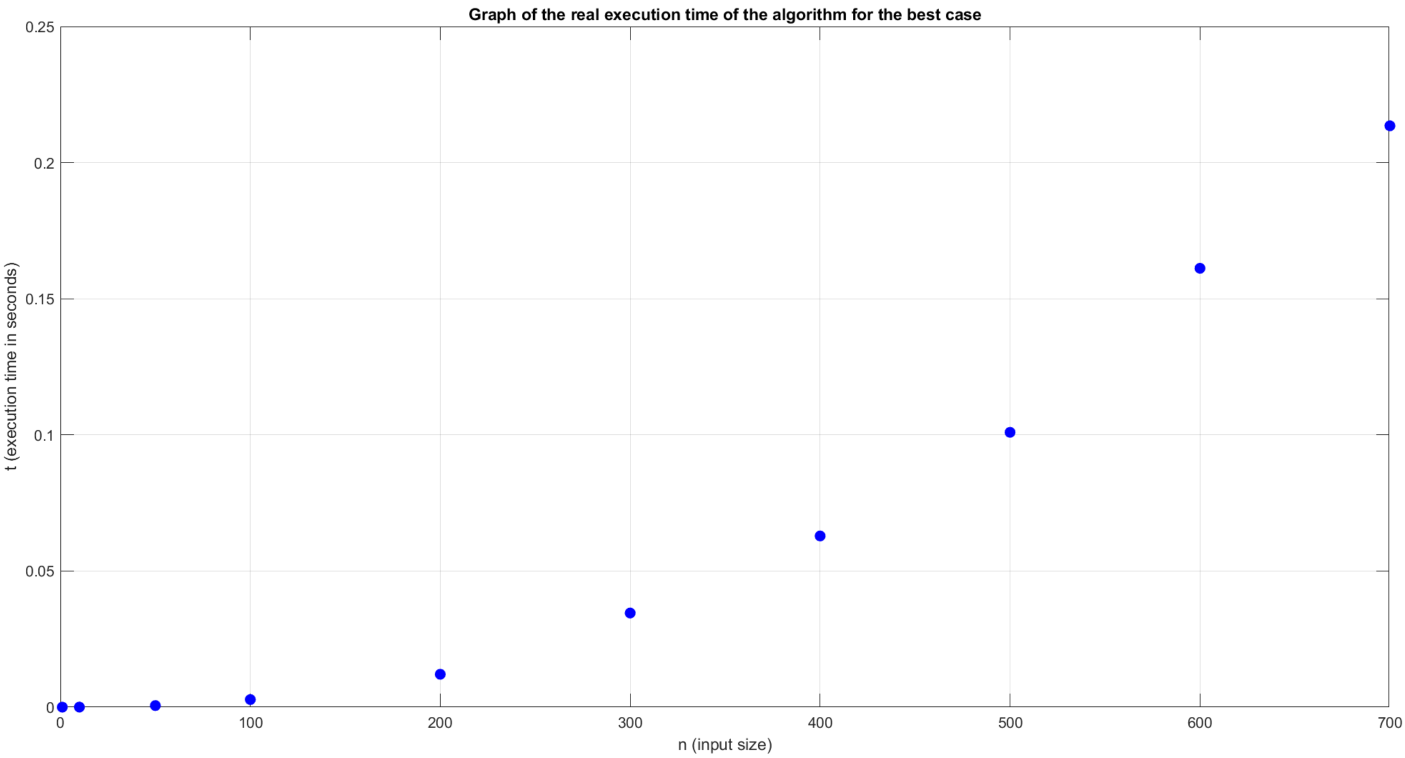
3.3 Average Case

N Size	Time Elapsed (in seconds)
1	0.0000019073486328125
10	0.00009799003601074219
50	0.01652693748474121
100	0.13457417488098145
200	1.1241791248321533
300	4.727046728134155
400	12.808381080627441
500	25.074081897735596
600	47.93078088760376
700	76.08589911460876

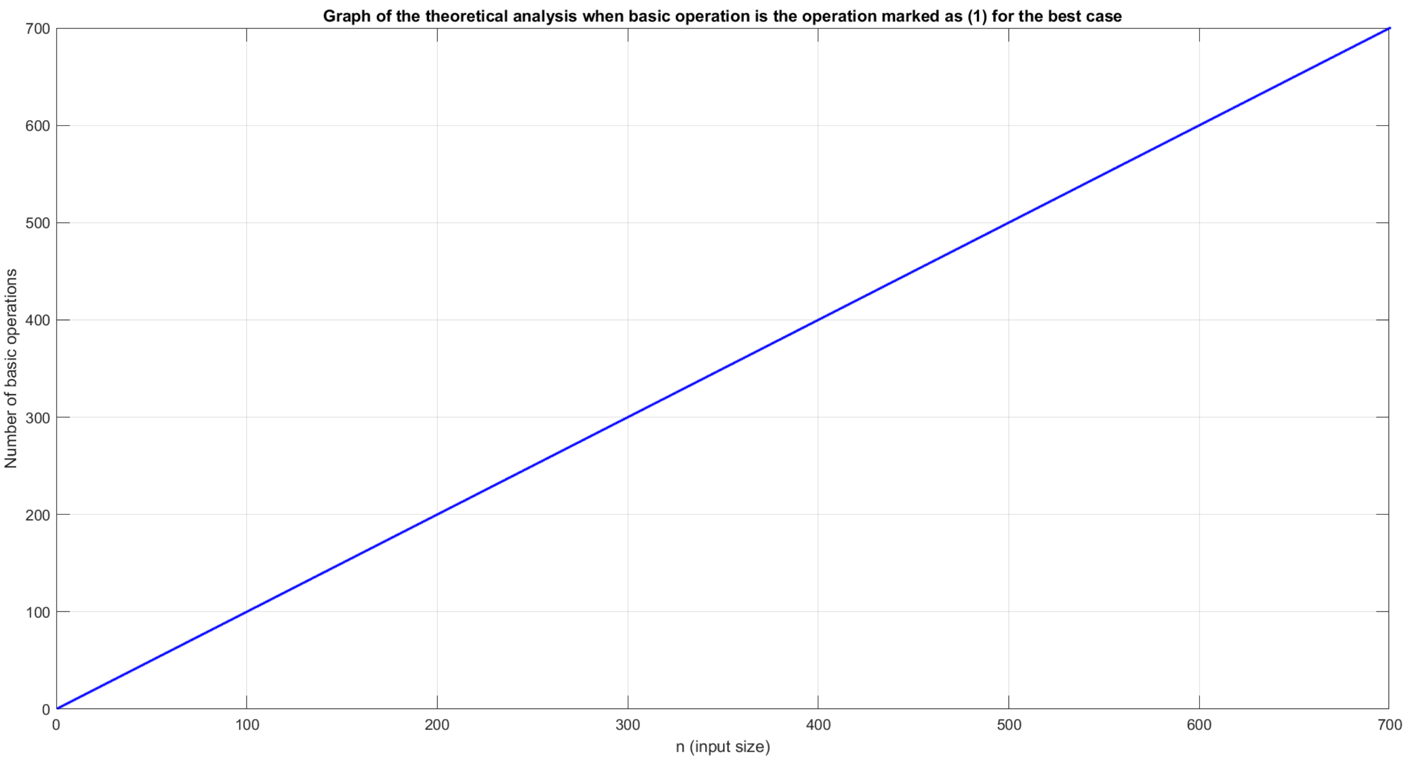
4 COMPARISON

4.1 Best Case

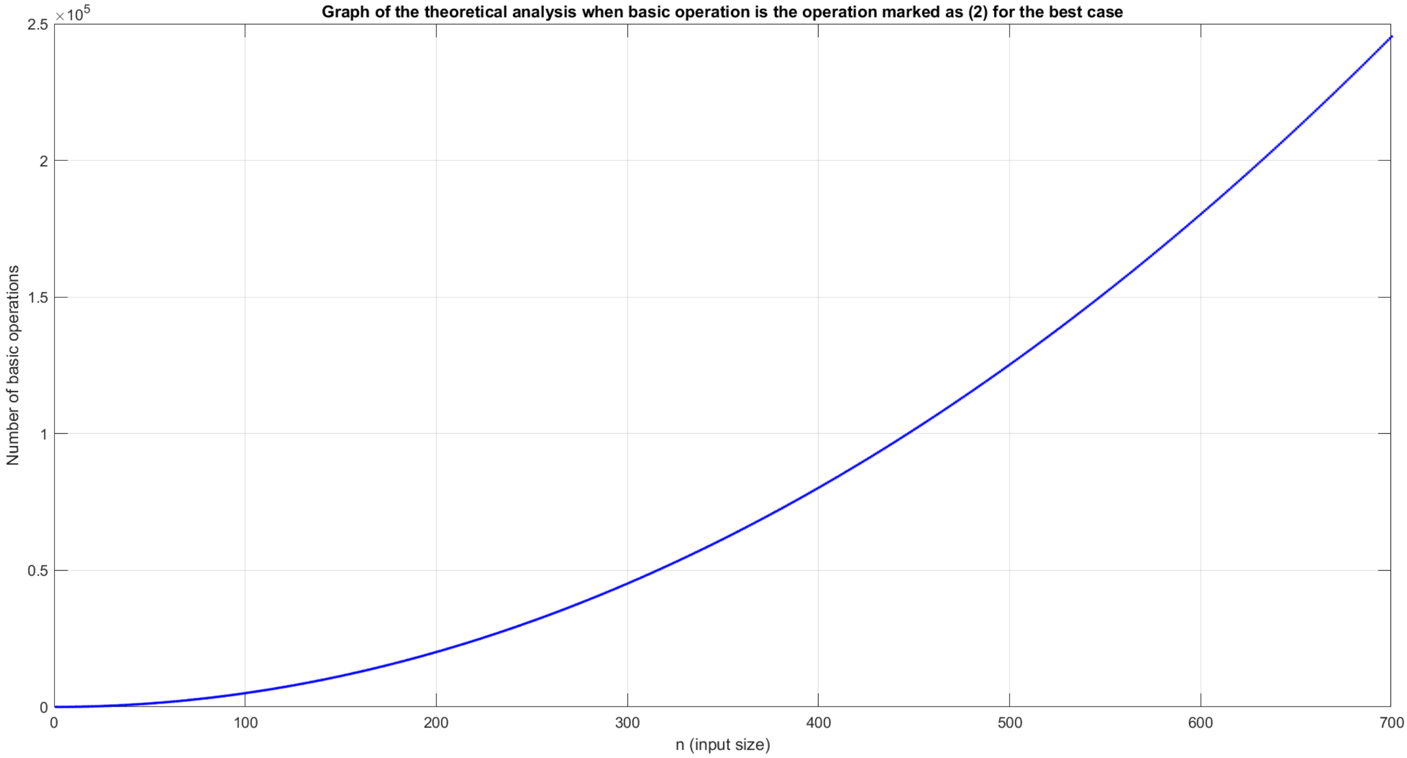
Graph of the real execution time of the algorithm



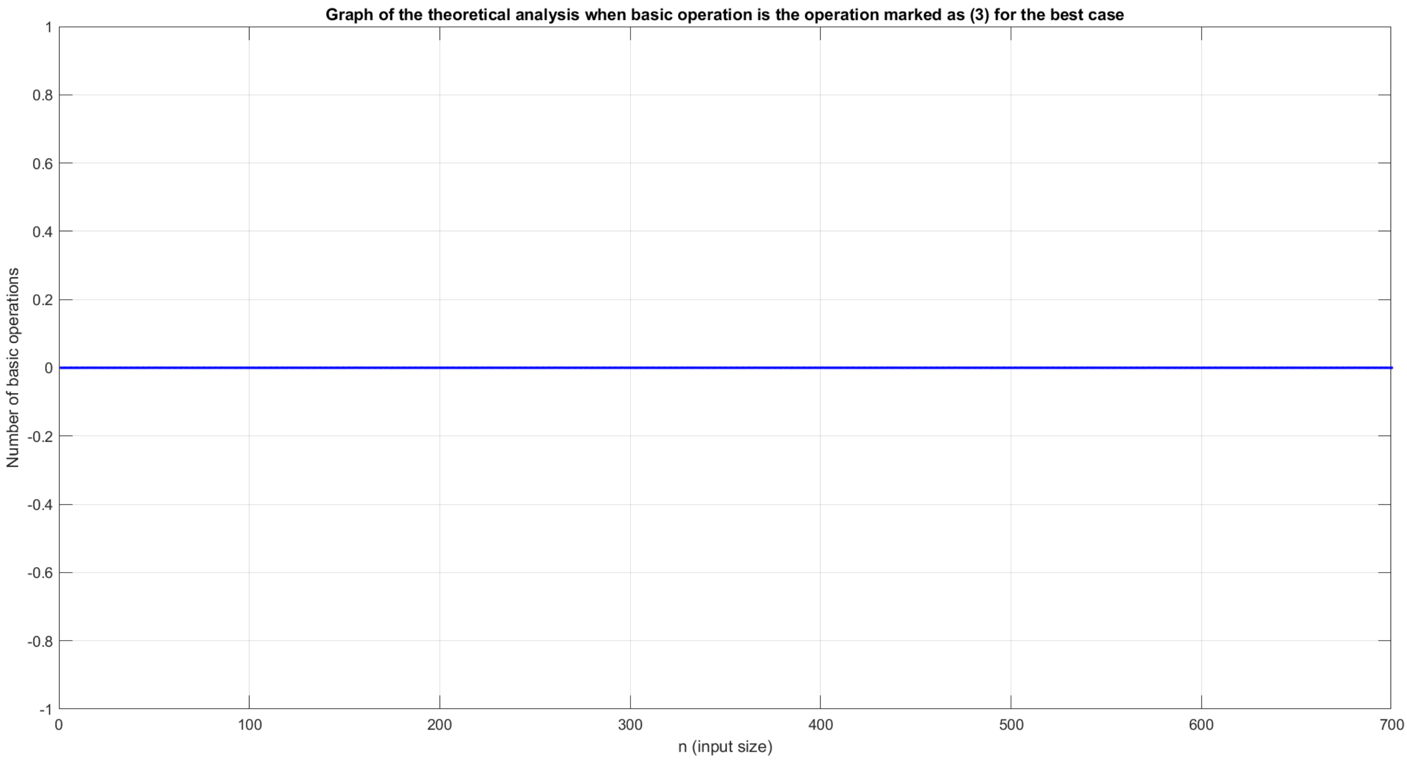
Graph of the theoretical analysis when basic operation is the operation marked as (1)



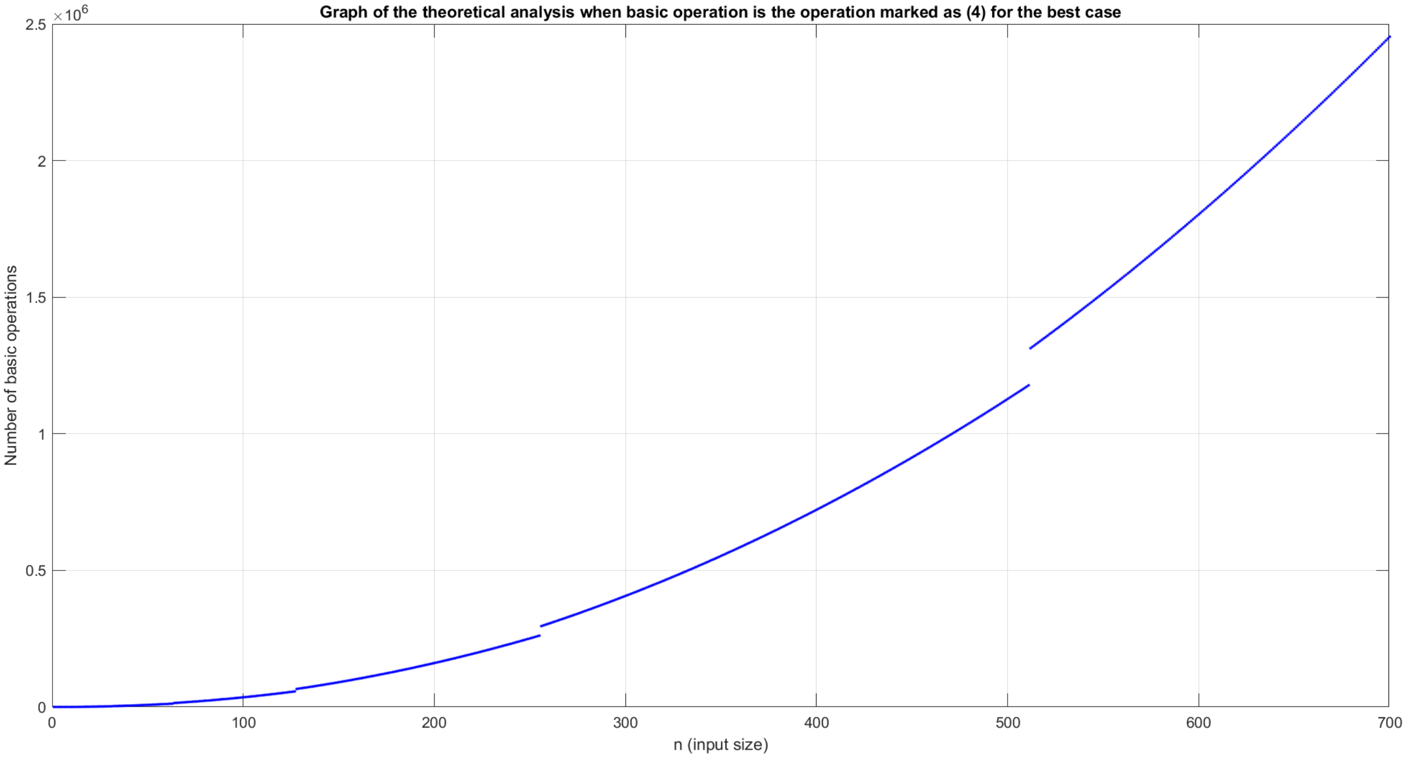
Graph of the theoretical analysis when basic operation is the operation marked as (2)



Graph of the theoretical analysis when basic operation is the operation marked as (3)



Graph of the theoretical analysis when basic operation is the operation marked as (4)



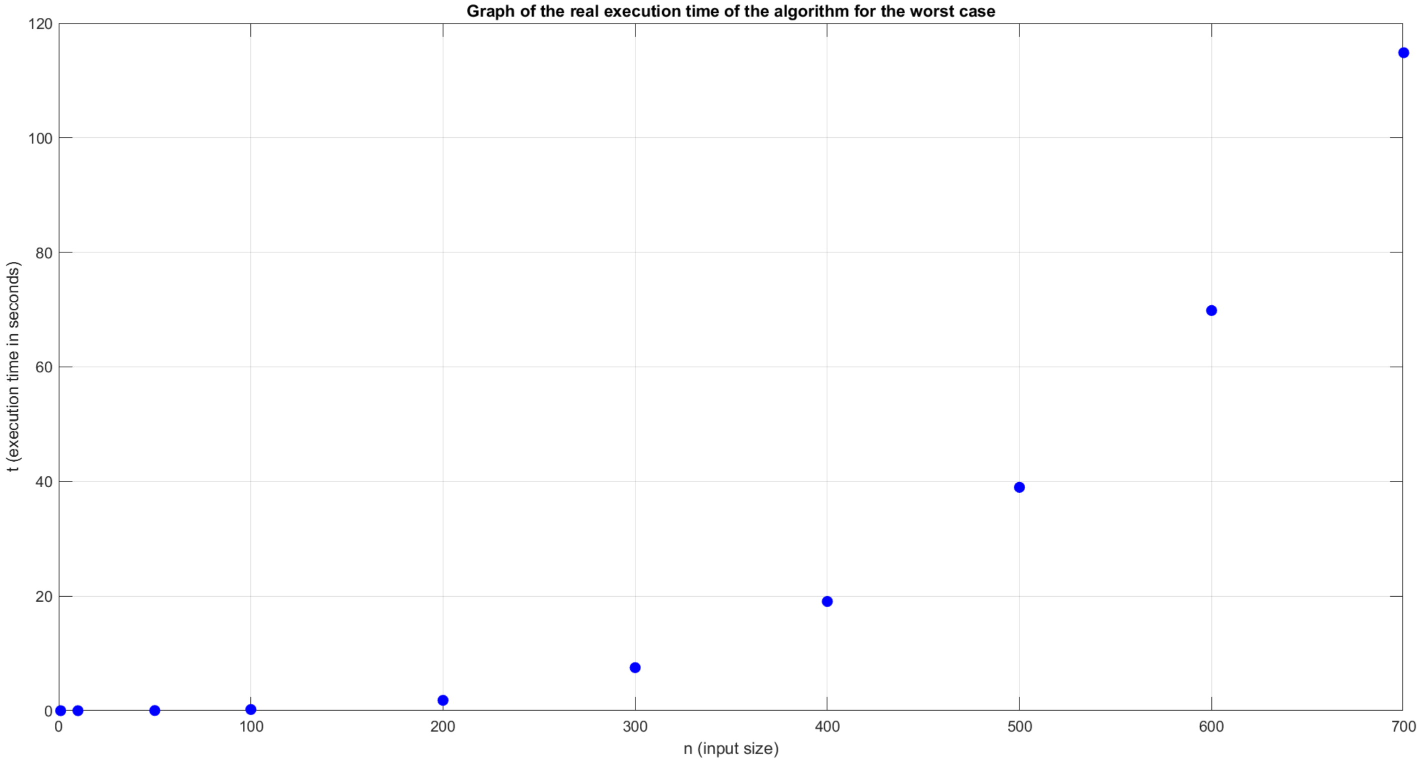
Comments

For the best case, the real execution time mostly resembles to theoretical graph (4). Because the operation marked as (4) is the basic operation of this algorithm. The operation (4) is characterizing this algorithm. Also from the similarity of graph (4) and reel execution time graph we can clearly see choosing (4) as a basic operation is the most accurate choice.

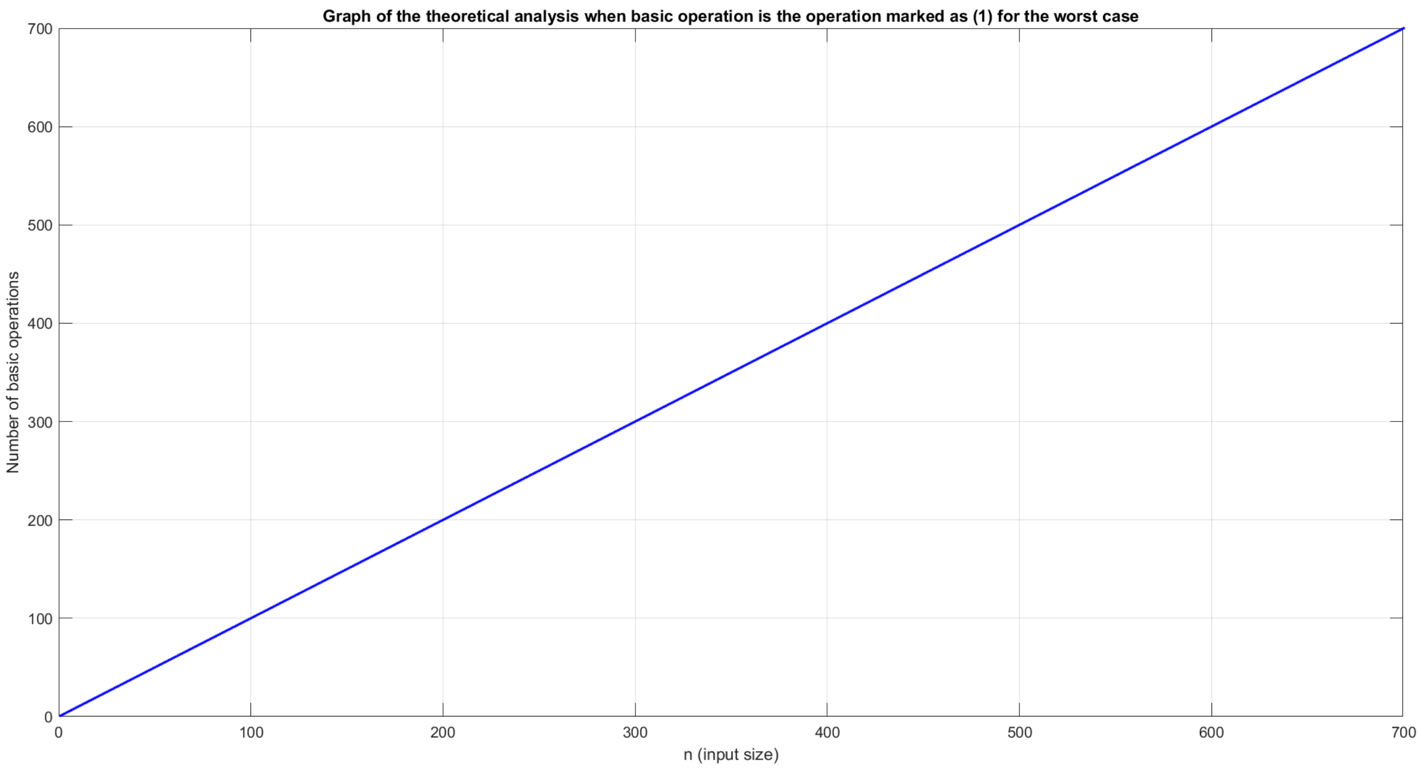
Graphs (1) and (3) diverge dramatically from the real execution time graph. Graph (2) is diverging also but not as much as graphs(1) and graphs(2) because complexity of graph(4) is $\Theta(n^2 logn)$ and complexities of graphs (2) (1) and (3) are $\Theta(n^2)$, $\Theta(n)$ and $\Theta(1)$ $respectively.$

4.2 Worst Case

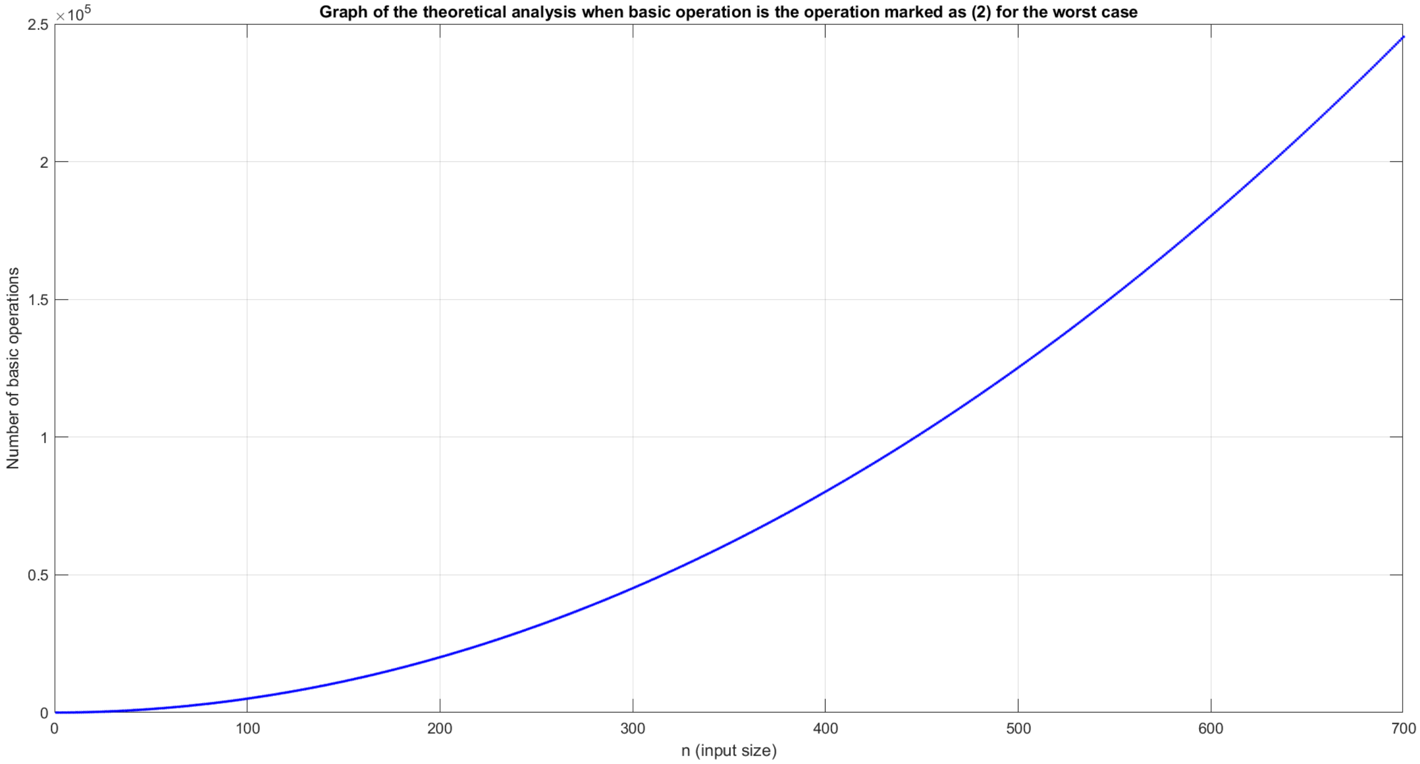
Graph of the real execution time of the algorithm



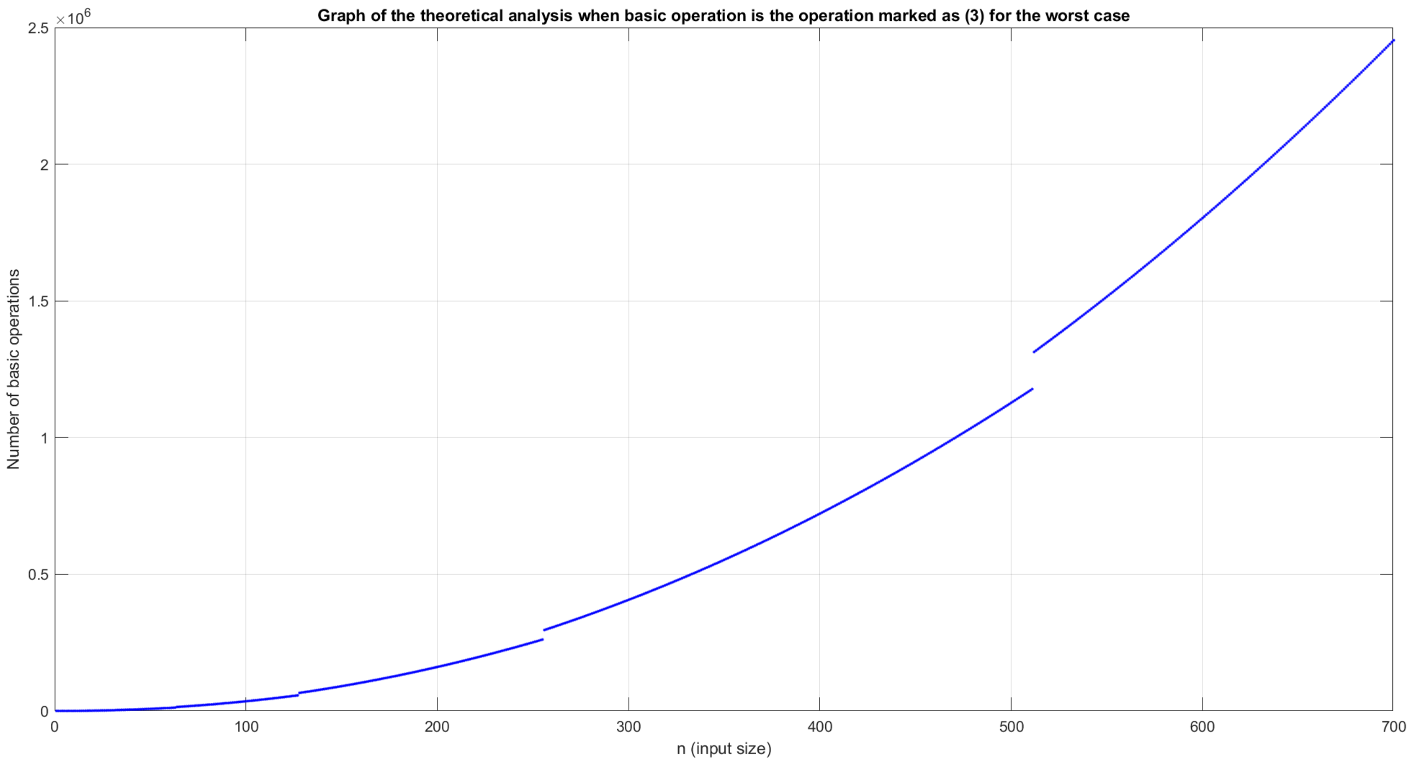
Graph of the theoretical analysis when basic operation is the operation marked as (1)



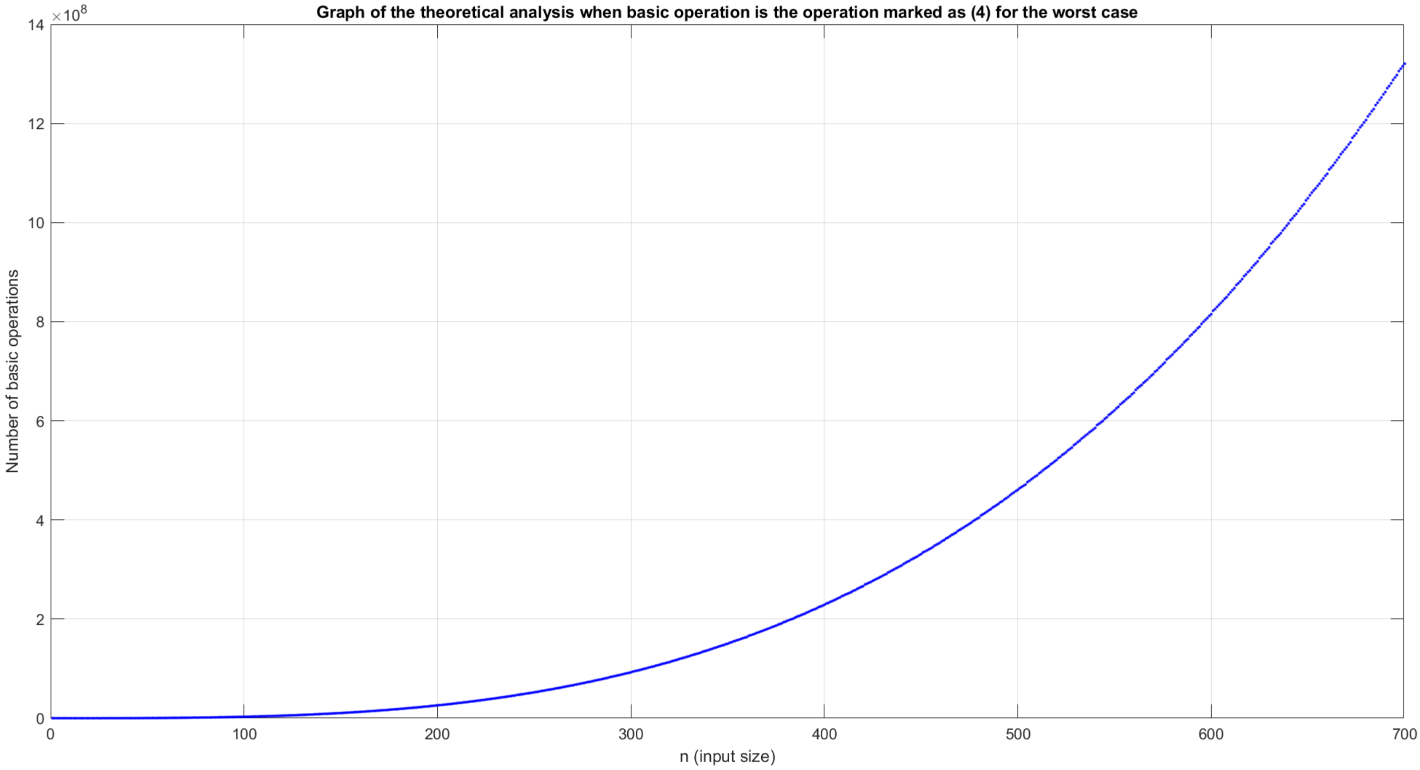
Graph of the theoretical analysis when basic operation is the operation marked as (2)



Graph of the theoretical analysis when basic operation is the operation marked as (3)



Graph of the theoretical analysis when basic operation is the operation marked as (4)



Comments

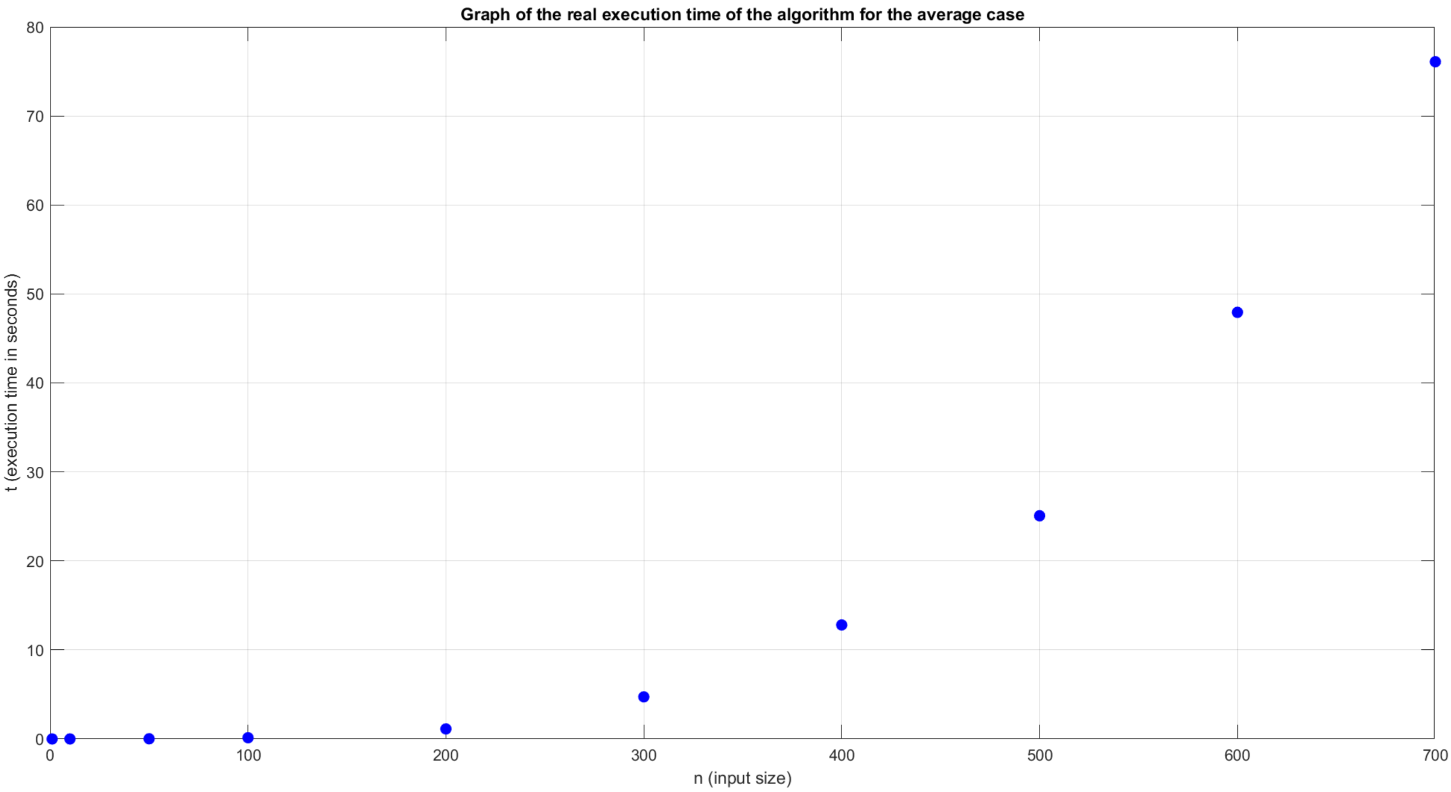
For the worst case, the real execution time mostly resembles to theoretical graph (4). Because the operation marked as (4) is the basic operation of this algorithm. The operation (4) is characterizing this algorithm. Also from the similarity of graph (4) and reel execution time graph we can clearly see choosing (4) as a basic operation is the most accurate choice.

Graphs (1) diverge dramatically from the real execution time graph. Graph (2) is diverging less than graph(1) but more than graph(3). Because complexity of the graph(4) is $\Theta(n^3 \log n)$ and complexities of graphs (1) (2) and (3) are $\Theta(n)$,

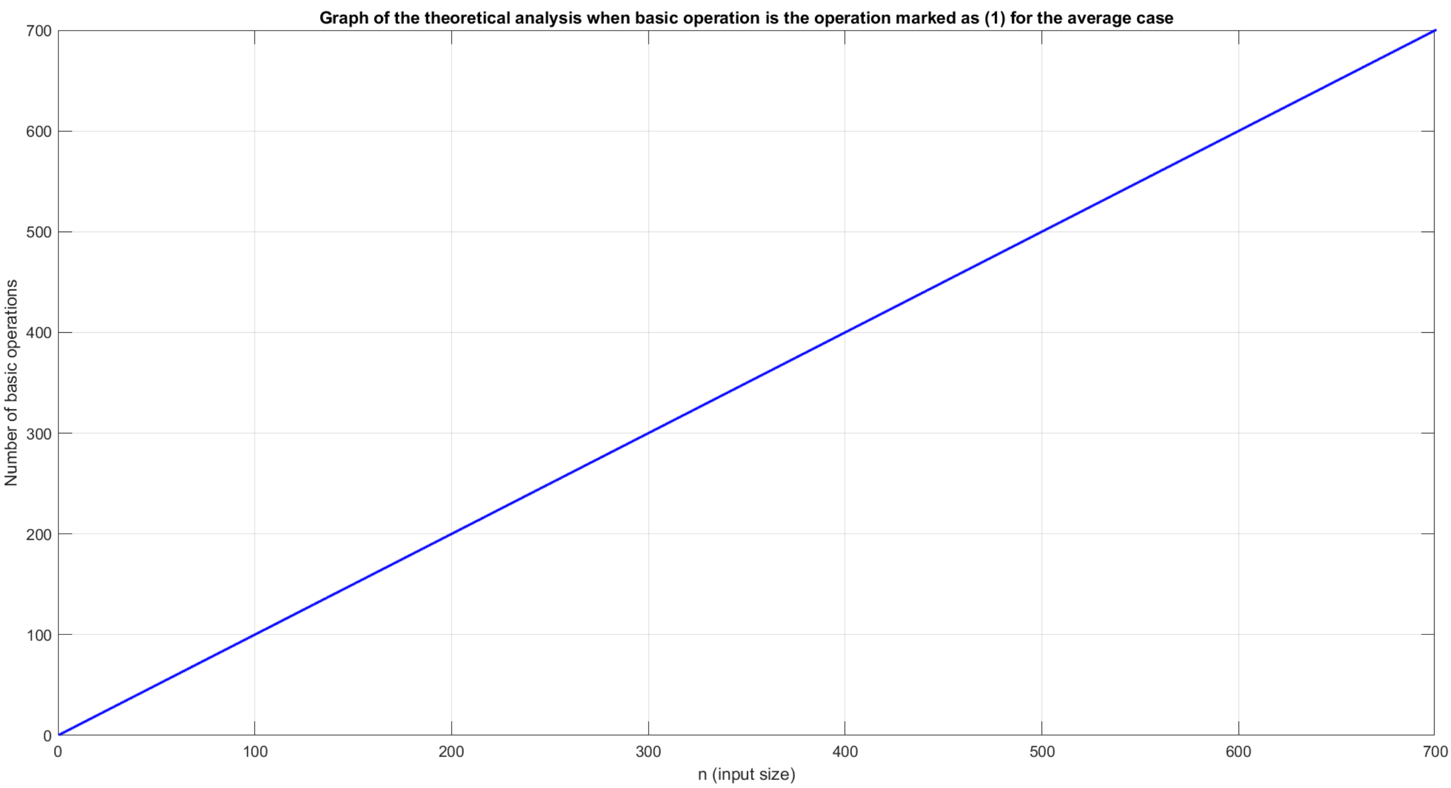
$\Theta(n^2)$ *and* $\Theta(n^2 \log n)$ *respectively.*

4.3 Average Case

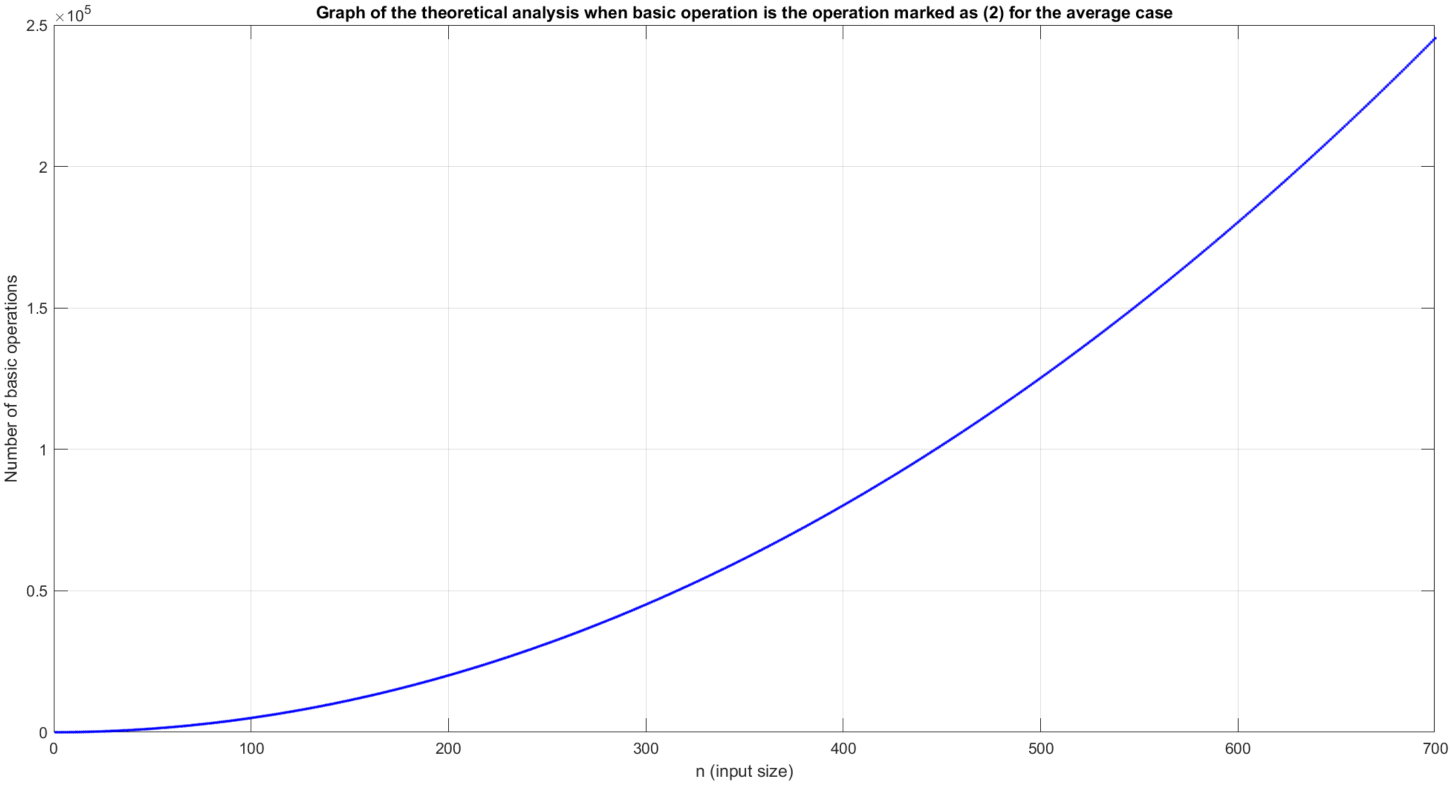
Graph of the real execution time of the algorithm



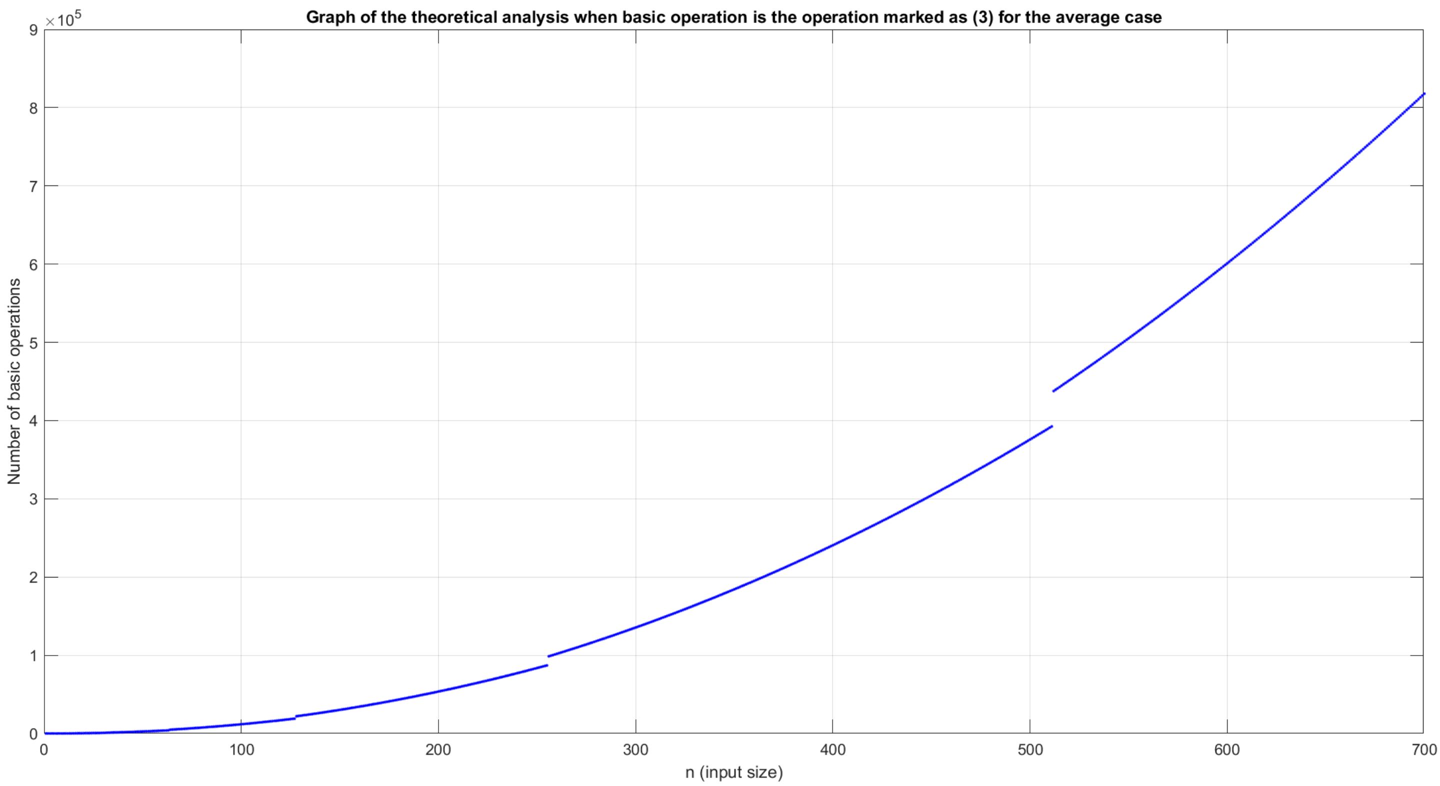
Graph of the theoretical analysis when basic operation is the operation marked as (1)



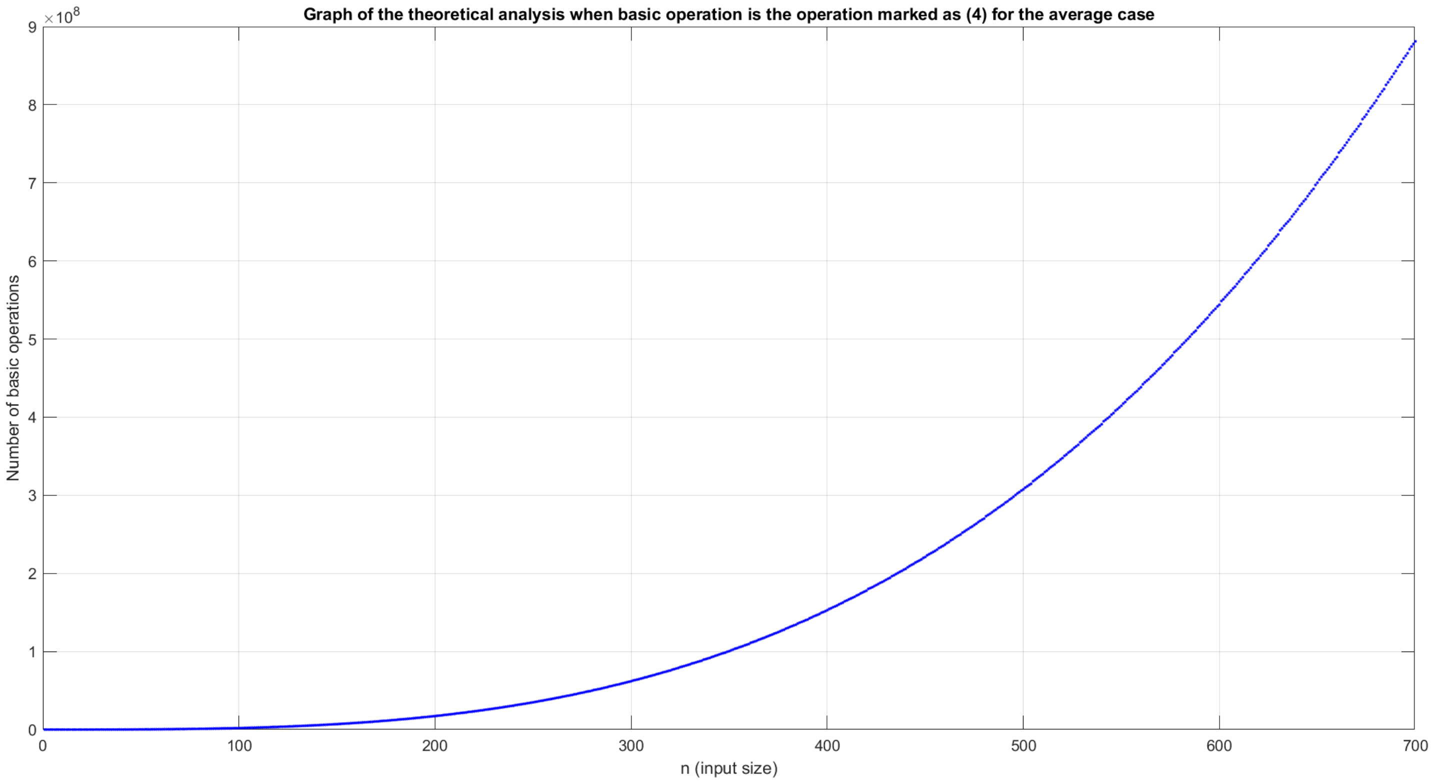
Graph of the theoretical analysis when basic operation is the operation marked as (2)



Graph of the theoretical analysis when basic operation is the operation marked as (3)



Graph of the theoretical analysis when basic operation is the operation marked as (4)



Comments

For the average case, the real execution time mostly resembles to theoretical graph (4). Because the operation marked as (4) is the basic operation of this algorithm. The operation (4) is characterizing this algorithm. Also from the similarity of graph (4) and reel execution time graph we can clearly see choosing (4) as a basic operation is the most accurate choice.

Graphs (1) diverge dramatically from the real execution time graph. Graph (2) is diverging less than graph(1) but more than graph(3). Because complexity of the graph(4) is $\Theta(n^3 logn)$ and complexities of graphs (1) (2) and (3) are $\Theta(n)$,

$\Theta(n^2)$ and $\Theta(n^2 logn)$ $respectively.$