SONER KUYAR – 2021400297
YUNUS KAĞAN AYDIN – 2021400123

# Table of Contents

SONER KUYAR – 2021400297
YUNUS KAĞAN AYDIN – 2021400123

## THEORETICAL ANALYSIS

### Basic operation is the comparison marked as (1)

*Analyze B(n):* Basic operation (if arr[i] = 0 then) is the comparison with time complexity 1. It will be executed each time in a loop.

So, B(n) = n*1 = n ∈ $\theta(n)$

*Analyze W(n):* As we said before no matter what the array's element is, it will execute n times again.

So, W(n) = n*1 = n ∈ $\theta(n)$

*Analyze A(n) :* A(n) = $\sum_{i=B(n)}^{W(n)} i * p$ due to our W(n) and B(n) is equal to each other sum of probabilities p = 1.

So , A(n) = n*1 = n ∈ $\theta(n)$

### Basic operations are the two assignments marked as (2)

*Analyze B(n):* If we choose (2) as basic operation, it will only execute when the arr[i] = 0 or 1. Best case is for all arr[i] = 2, it will never execute the (2) operation.

So, B(n) = 0*1 ∈ $\theta(1)$

*Analyze W(n):* (2) executes at two branch and inner loops goes through same range:

(for m ← i to n − 1 do).

If arr[i] != 2, code will execute inside one of the branches at each iteration of the outer loop.

Outer loop executes n times , inner loop executes i times which is the value of outer loop.

$\sum_{i=0}^{n-1}(n-i) = n + n-1 + \cdots + 1 = \frac{n(n+1)}{2} = \frac{n^2+n}{2} \in \theta(n^2)$

*Analyze A(n):* There are three possibilities arr[i] to be. We can take the average time complexity by the expected value theorem. Our expected value will be the basic operation count.

$\sum_{i=0}^{n-1} p_k I = p_k \sum_{i=0}^{n-1} p_k * I \ , \ k \in \{0,1,2\}$

- $p_k$ is the probability of arr[i] = k, where $p_1 = p_2 = p_3 = \frac{1}{3}$
- $I$: expected value of the basic operation inside the loop.

For arr[i] = 0 : $I = \sum_{j=i}^{n-1} p_k * 1$

For arr[i] = 1 : $I = \sum_{j=i}^{n-1} p_k * 1$

For arr[i] = 2 : $I = p_k * 0$

A(n) = 2* $\sum_{i=0}^{n-1}\sum_{j=i}^{n-1} p_k * 1 = 2 * p_k \sum_{i=0}^{n-1}\sum_{j=i}^{n-1} 1$

A(n) = $\frac{2}{3}$* $\sum_{i=0}^{n-1}\sum_{j=i}^{n-1} 1 = \frac{2}{3}\sum_{i=0}^{n-1}(n-i) = \frac{n(n+1)}{3} = \frac{n^2+n}{3} \in \theta(n^2)$

SONER KUYAR – 2021400297
YUNUS KAĞAN AYDIN – 2021400123

Basic operations are the two comparisons marked as (3)

*Analyze B(n):* If we choose (3) as basic operation, it will only execute when the arr[i] = 0 or 2. Best case is for all arr[i] = 1, it will never execute the (3) operation.
So, B(n) = 0*1 $\in \theta(1)$

*Analyze W(n):* In each outer for loop algorithm executes only if arr[i] = 0 or 2.
We can assess each case separately:

- arr[i] = 0 :
  Execution time depends on two elements, one for loop from i to n-1 and a while loop.
  If we start from while loop:
  k <- n (starts with n value) and the loop terminates when k $\leq 0$.
  If we assume that n = $3^b$ loop terminates in $\log_3 n + 2$ steps. b$\geq 0$
  Outer loop executes n-i times.
  So, for each i from 0 to n-1. (3) executes $(\mathrm{n} - \mathrm{i}) * (\log_3 n + 2)$ times.
- arr[i] = 2:

  It basically goes through n + 1 times in a while loop for each i from 0 to n -1.
  So, for each i from 0 to n-1. (3) executes $n$ times.

It's clearly seen that $(\mathrm{n} - \mathrm{i}) * (\log_3 n + 2)$ is equal to n somewhere in the array, after that point execution for arr[i] = 2 will be greater than the arr[i] = 1.
Let's find this index with respect to n.
$(\mathrm{n} - \mathrm{i}) * (\lfloor \log_3 n \rfloor + 2) = n + 1$ (index where the executions of cases are equal)
If we solve this equation for i:
$i = n - \frac{n+1}{2 + \lfloor \log_3 n \rfloor}$ due to its an index, we can round this i value. $i = n - \left\lfloor \frac{n+1}{2 + \log_3 n} \right\rfloor$
If we subtract the execution time after i index from the total execution, we can find the time complexity for the first part $T_1$.

$$\sum_{i=0}^{n-1}(n-i) * (\log_3 n + 2) = \sum_{i=0}^{n-\left\lfloor \frac{n+1}{2+ \log_3 n} \right\rfloor - 1}(n-i) * (\log_3 n + 2) + \sum_{i=n-\left\lfloor \frac{n+1}{2+ \log_3 n} \right\rfloor}^{n-1}(n-i) * (\log_3 n + 2)$$

$$T_1 = \sum_{i=0}^{n-1}(n-i) * (\log_3 n + 2) - \sum_{i=n-\left\lfloor \frac{n+1}{2+ \lfloor \log_3 n \rfloor} \right\rfloor}^{n-1}(n-i) * (\log_3 n + 2)$$

$$T_1 = (\log_3 n + 2)\left(\frac{n^2+n}{2} - \sum_{i=n-\left\lfloor \frac{n+1}{2+ \lfloor \log_3 n \rfloor} \right\rfloor}^{n-1}(n-i)\right)$$

$$T_1 = (\log_3 n + 2)\frac{1}{2}(n^2 + n - \left(\left\lfloor \frac{n+1}{2+ \lfloor \log_3 n \rfloor} \right\rfloor + 1\right) * \left(\left\lfloor \frac{n+1}{2+ \lfloor \log_3 n \rfloor} \right\rfloor\right)$$

$$T_1 = \frac{1}{2}\left(n^2 \log_3 n + n log_3 n - \left\lfloor \frac{(n+1)^2}{2 + \lfloor \log_3 n \rfloor} \right\rfloor - n - 1\right) + n^2 + n - \left\lfloor \frac{n+1}{2 + \lfloor \log_3 n \rfloor} \right\rfloor^2 - \left\lfloor \frac{n+1}{2 + \lfloor \log_3 n \rfloor} \right\rfloor$$

We can say $T_2$ the time where the else part(arr[i] = 2) started to execute it will execute rest from the n execution.

$$T_2 = \left(\left\lfloor \frac{n+1}{2 + \log_3 n} \right\rfloor\right) * n = n * \left\lfloor \frac{n+1}{2 + \lfloor \log_3 n \rfloor} \right\rfloor$$

SONER KUYAR – 2021400297
YUNUS KAĞAN AYDIN – 2021400123

W(n) = $T_1 + T_2$
W(n) = $\frac{1}{2}\left(n^2 \log_3 n + nlog_3 n - \left\lfloor\frac{(n+1)^2}{2+ \lfloor\log_3 n\rfloor}\right\rfloor - n - 1\right) + n^2 + n - \left\lfloor\frac{n+1}{2+ \lfloor\log_3 n\rfloor}\right\rfloor^2 - \left\lfloor\frac{n+1}{2+ \lfloor\log_3 n\rfloor}\right\rfloor + n * \left\lfloor\frac{n+1}{2+ \lfloor\log_3 n\rfloor}\right\rfloor$
If we make an asymptotic analysis all the other terms are lower growth rate than $n^2 \log_3 n$
So, W(n) ∈ $\theta(n^2 logn)$

*Analyze A(n):*
We can use our findings from the W(n). There will be no operation for arr[i] = 1.
So, we will take the expected time complexity of first and third condition.

- arr[i] = 0:
  Probability of entering first loop for each I is $p_1 = \frac{1}{3}$.Execution time depends on two elements, one for loop from i to n-1 and a while loop.
  If we start from while loop:
    k <- n (starts with n value) and the loop terminates when k ≤ 0.
    If we assume that n = $3^b$ terminates in $\log_3 n + 2$ steps. b≥ 0
  Outer loop executes n-i times.
  So, for each i from 0 to n-1. (3) executes $(n - i) * (\log_3 n + 1)$ times.
  $\sum_{i=0}^{n-1}(n - i) * (\log_3 n + 2) * p_1$
  $T_0 = \frac{n*(n+1)}{2} * (\log_3 n + 2) * \frac{1}{3}$
  $T_0 = \frac{n^2 \log_3 n + 2n^2 + 2n + nlog_3 n}{6}$

- arr[i] = 1: Again, it is zero because it will never execute if it enters here.
  $T_1 = 0$
- arr[i] = 2:

  Probability of entering second loop for each i is $p_2 = \frac{1}{3}$. It basically goes through n + 1 times in a while loop for each i from 0 to n -1.
  So, for each i from 0 to n-1. (3) executes $n + 1$ times.
  $T_2 = \sum_{i=0}^{n-1}(n + 1) * p_2 = \frac{n^2+n}{3}$

So, A(n) = $T_0 + T_1 + T_2$
A(n) = $\frac{n^2 \log_3 n + 4n^2 + 4n + nlog_3 n}{6}$ highest term is the $n^2 \log_3 n$
So, A(n) ∈ $\theta(n^2 log\, n)$

Basic operations are the three assignments marked as (4)
We have 3 different conditions when filling an array with full of same elements.

arr[i] = 0: $\sum_{i=0}^{n-1}\sum_{j=i}^{n-1}(\log_3 n + 1) = \frac{n*(n+1)*\log(3n)}{log9} \in \theta(n^2 \log n)$

arr[i] = 1: $\sum_{i=0}^{n-1}\sum_{m=i}^{n-1}\sum_{l=m}^{n-1}\sum_{t=1}^{n}\left(\frac{n-1}{t} + 1\right) = \frac{1}{6} * n * (n^2 + 3n + 2) * \left((n - 1) * H_n + n\right)$
∈ $\theta(n^4 \log n)$
* $H_n$ will execute similar as logn.

SONER KUYAR – 2021400297
YUNUS KAĞAN AYDIN – 2021400123

arr[i] = 2: $\sum_{i=0}^{n-1}\sum_{p=0}^{n}\sum_{j=0}^{p^2-1} 1 = \frac{1}{6} * n^2 * (n+1) * (2n+1) \in \theta(n^4)$

*Analyze B(n)*: All elements must be 0 from the operations given as above,

B(n) = $\sum_{i=0}^{n-1}\sum_{j=i}^{n-1} (log_3 n + 1) = \frac{n*(n+1)*log(3n)}{log9} \in \theta(n^2 log\, n)$

*Analyze W(n):* All elements must be 1 from the operations given as above,

W(n) = $\sum_{i=0}^{n-1}\sum_{m=i}^{n-1}\sum_{l=m}^{n-1}\sum_{t=1}^{n}\left(\frac{n-1}{t}+1\right) = \frac{1}{6} * n * (n^2+3n+2) * \left((n-1) * H_n + n\right)$
$\in \theta(n^4 \log n)$

*Analyze A(n):* In average case, these three situations will appear in $\frac{1}{3}$ probability each.

A(n) = $T_0 + T_1 + T_2$

$p_0 = \frac{1}{3}, p_1 = \frac{1}{3}, p_2 = \frac{1}{3}$

$T_0 = \sum_{i=0}^{n-1} p_0 * \sum_{j=i}^{n-1} (\log_3 n + 1) = \frac{1}{3} * \frac{n*(n+1)*\log(3n)}{log9}$

$T_1 = \sum_{i=0}^{n-1} p_1 * \sum_{m=i}^{n-1}\sum_{l=m}^{n-1}\sum_{t=1}^{n}\left(\frac{n-1}{t}+1\right) = \frac{1}{3} * \frac{1}{6} * n * (n^2+3n+2) * \left((n-1) * H_n + n\right)$

$T_2 = \sum_{i=0}^{n-1} p_2 * \sum_{p=0}^{n}\sum_{j=0}^{p^2-1} 1 = \frac{1}{3} * \frac{1}{6} * n^2 * (n+1) * (2n+1)$

A(n) = $\frac{1}{3} * \frac{n*(n+1)*\log(3n)}{log9} + \frac{1}{3} * \frac{1}{6} * n * (n^2+3n+2) * \left((n-1) * H_n + n\right) + \frac{1}{3} * \frac{1}{6} * n^2 *$
$(n+1) * (2n+1) \in \theta(n^4 \log n)$

## IDENTIFICATION OF BASIC OPERATION(S)

We selected basic operation (4) because operation (4) has the complexity of the most inner loop. It characterizes the algorithm because it executes in each possible case: (arr[i] = 0, arr[i] = 1, arr[i] = 2). Also, it has highest complexity so we should consider while finding characterizing complexity of the algorithm.

## REAL EXECUTION

### Best Case

| N Size | Time Elapsed (s) |
|---|---|
| **1** | 0.000004 |
| **5** | 0.000005 |
| **10** | 0.000016 |
| **20** | 0.000053 |
| **30** | 0.000152 |
| **40** | 0.000247 |
| **50** | 0.000383 |
| **60** | 0.000564 |

| | |
|---|---|
| **70** | 0.000759 |
| **80** | 0.000961 |
| **90** | 0.001538 |
| **100** | 0.001893 |
| **110** | 0.002161 |
| **120** | 0.002555 |
| **130** | 0.003009 |
| **140** | 0.003483 |
| **150** | 0.004020 |

### Worst Case

| N Size | Time Elapsed (s) |
|---|---|
| **1** | 0.000003 |
| **5** | 0.000059 |
| **10** | 0.000733 |
| **20** | 0.010014 |
| **30** | 0.049520 |
| **40** | 0.154938 |
| **50** | 0.438406 |
| **60** | 0.785263 |
| **70** | 1.461040 |
| **80** | 2.498144 |
| **90** | 4.248906 |
| **100** | 6.171375 |
| **110** | 9.035313 |
| **120** | 12.849510 |
| **130** | 17.867899 |
| **140** | 24.049748 |
| **150** | 31.999491 |

### Average Case

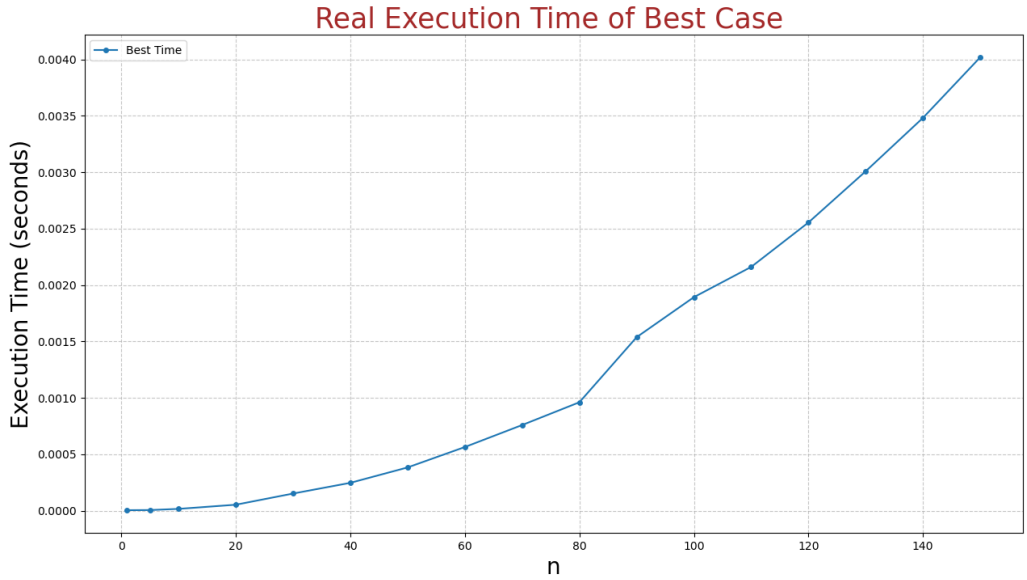
| N Size | Time Elapsed (s) |
|---|---|
| **1** | 0.000002 |
| **5** | 0.000017 |
| **10** | 0.000134 |
| **20** | 0.001807 |
| **30** | 0.010003 |
| **40** | 0.030351 |
| **50** | 0.073329 |
| **60** | 0.163482 |
| **70** | 0.299136 |
| **80** | 0.519155 |
| **90** | 0.835262 |
| **100** | 1.237465 |
| **110** | 1.800697 |

| | |
|---|---|
| **120** | 2.573088 |
| **130** | 3.571887 |
| **140** | 4.859178 |
| **150** | 6.421979 |

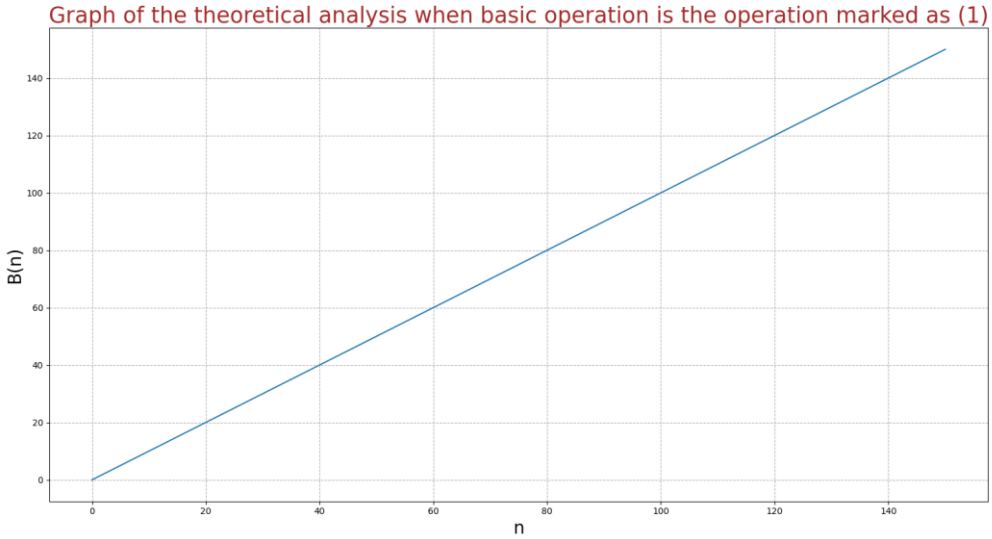
## COMPARISON

### Best Case

*Graph of the real execution time of the algorithm*

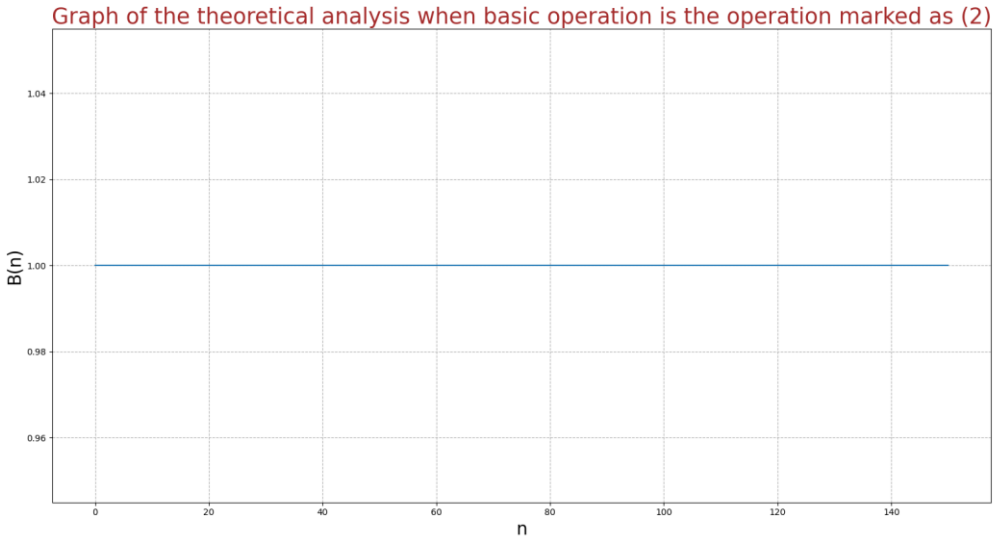



*Graph of the theoretical analysis when basic operation is the operation marked as (1):*

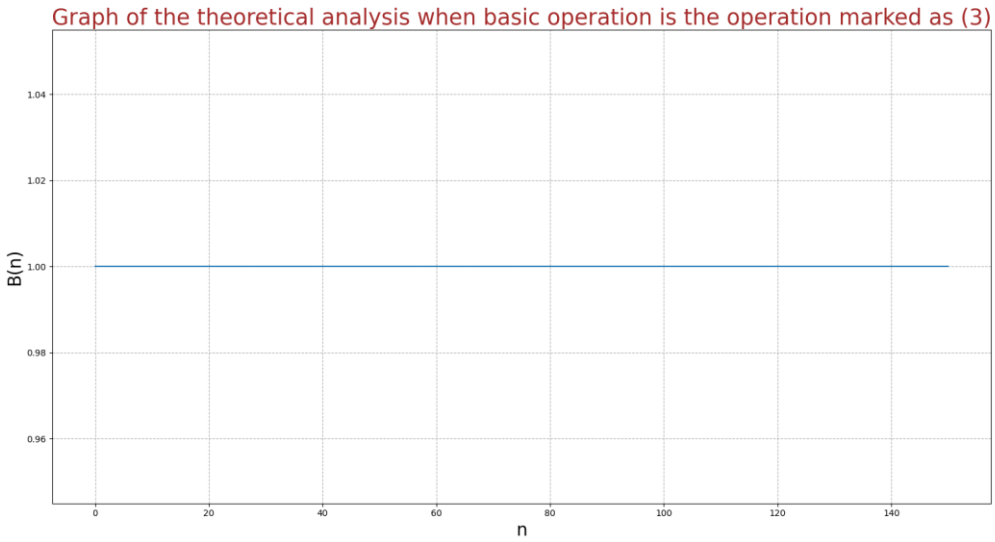


SONER KUYAR – 2021400297
YUNUS KAĞAN AYDIN – 2021400123

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

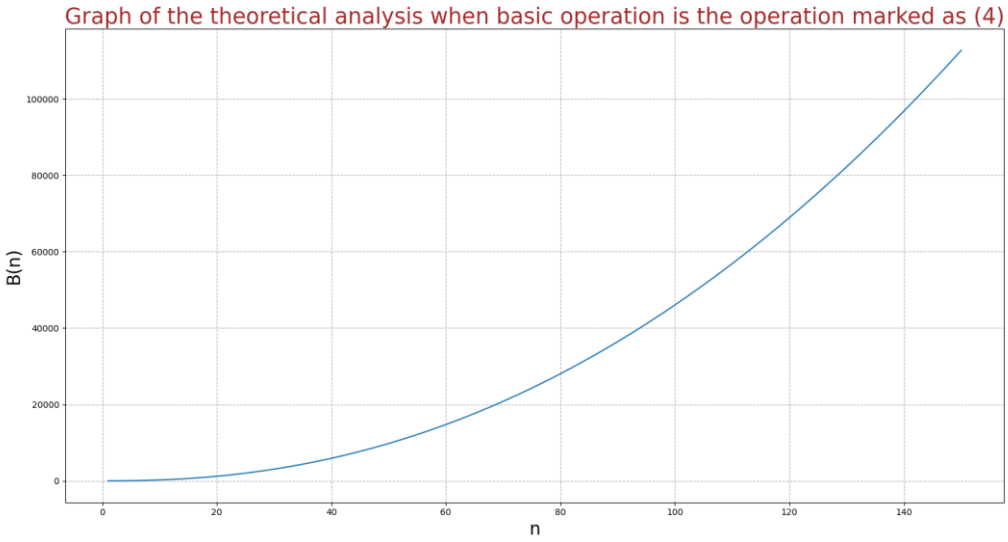



*Graph of the theoretical analysis when basic operation is the operation marked as (3):*

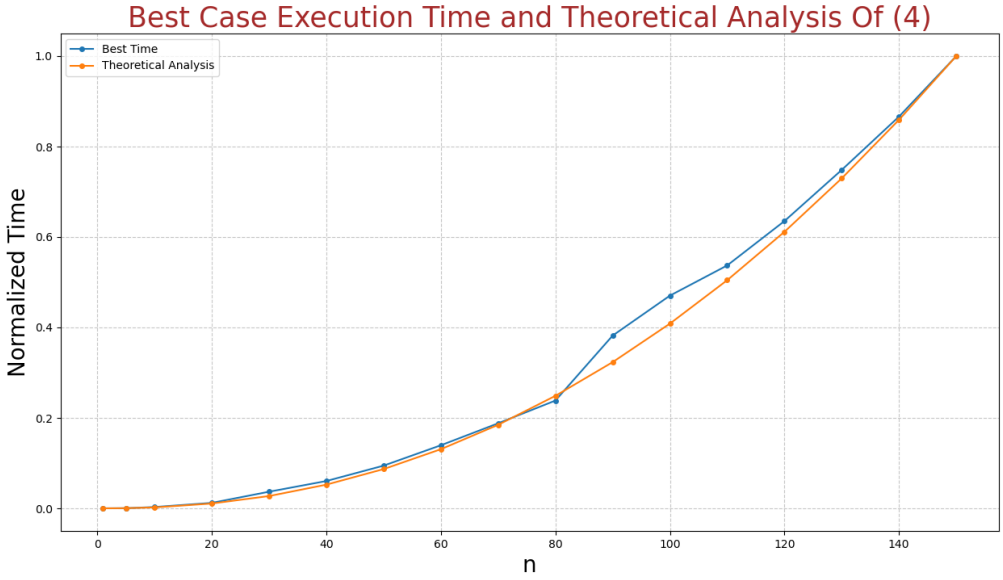


SONER KUYAR – 2021400297
YUNUS KAĞAN AYDIN – 2021400123

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




*Comments*

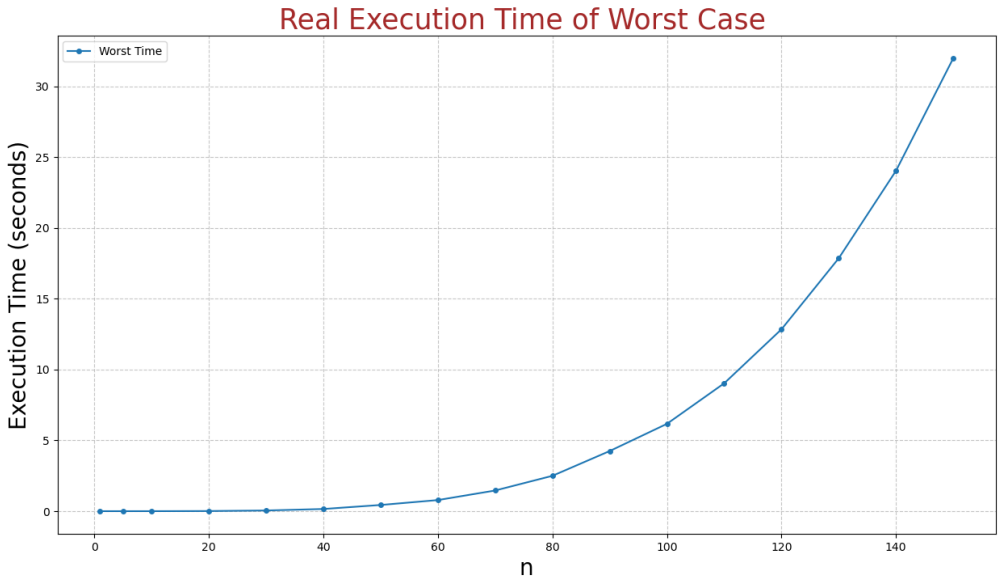



We have normalized the best execution time values in order to compare with theoretical analysis of (4). It can be clearly seen that these two graphs match with each other. With this result, we proved that execution (4) is the basic operation of this algorithm.

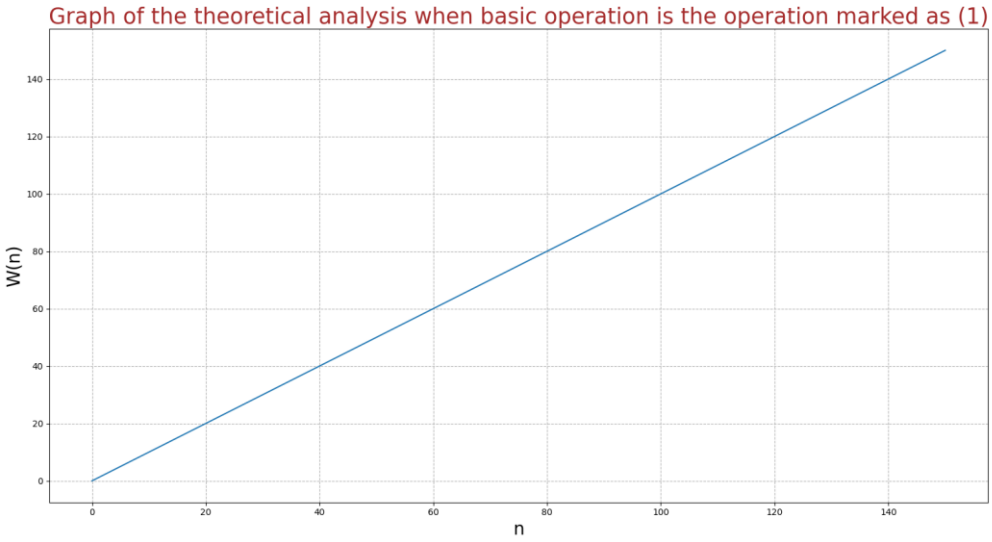
SONER KUYAR – 2021400297
YUNUS KAĞAN AYDIN – 2021400123

## Worst Case

*Graph of the real execution time of the algorithm*

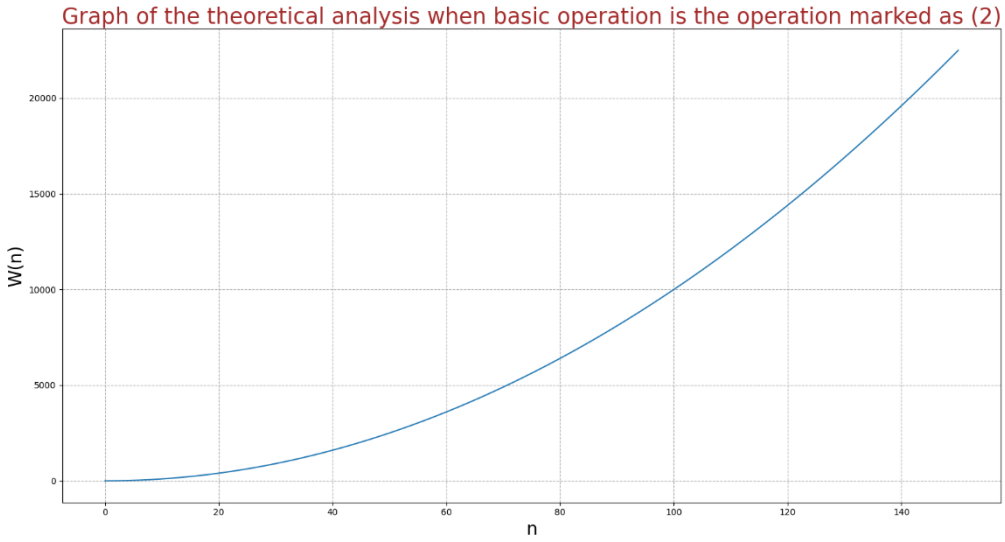



*Graph of the theoretical analysis when basic operation is the operation marked as (1)*

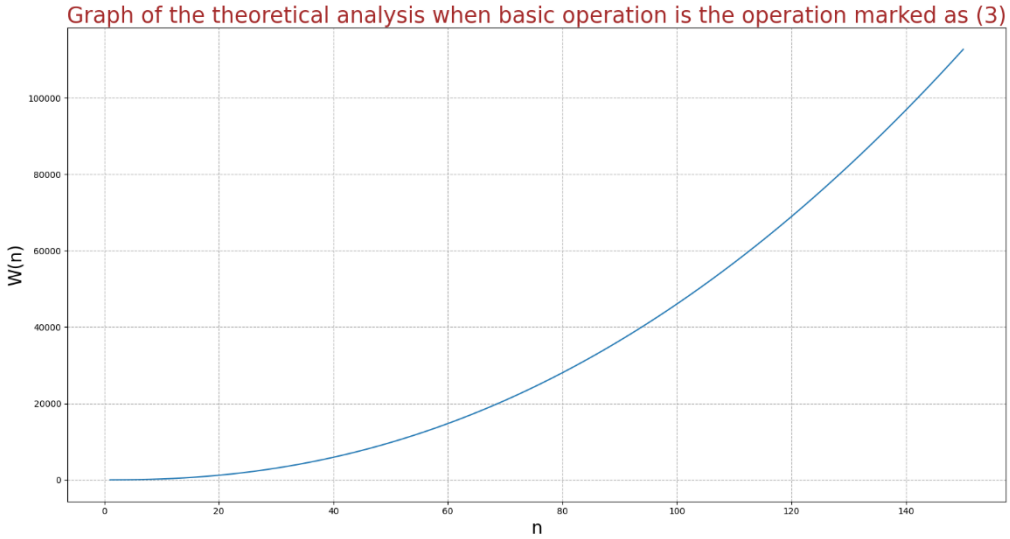


SONER KUYAR – 2021400297
YUNUS KAĞAN AYDIN – 2021400123

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

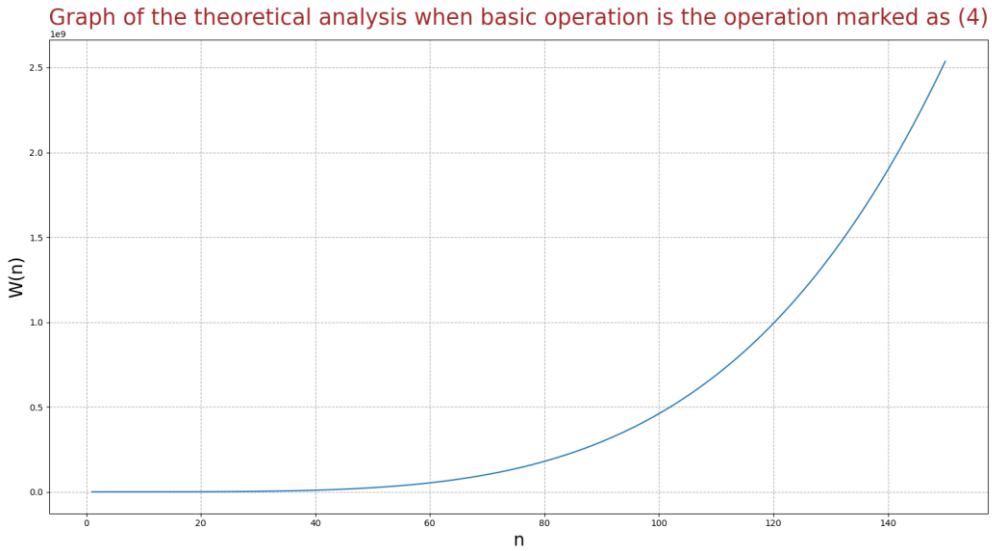



*Graph of the theoretical analysis when basic operation is the operation marked as (3):*

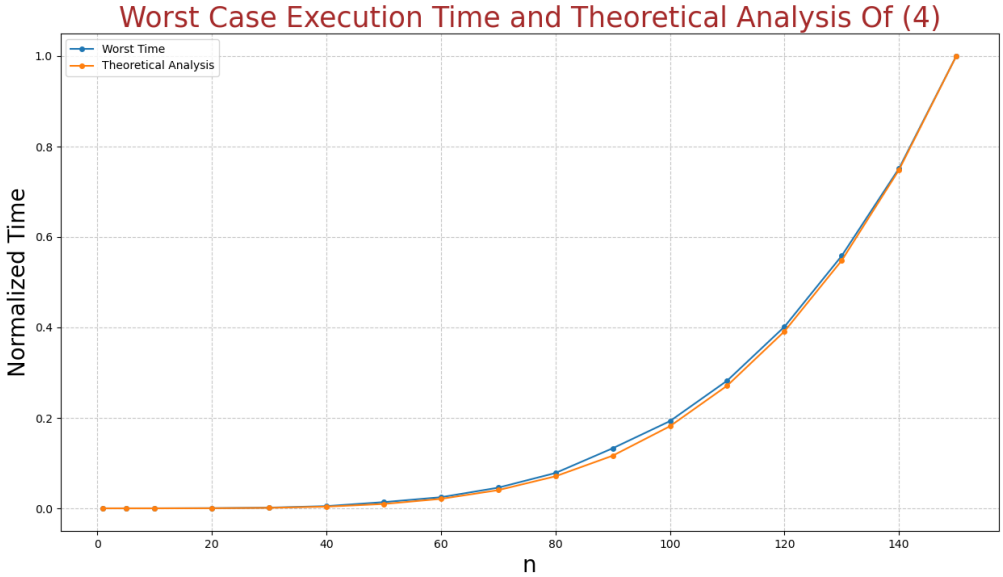


SONER KUYAR – 2021400297
YUNUS KAĞAN AYDIN – 2021400123

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




*Comments*

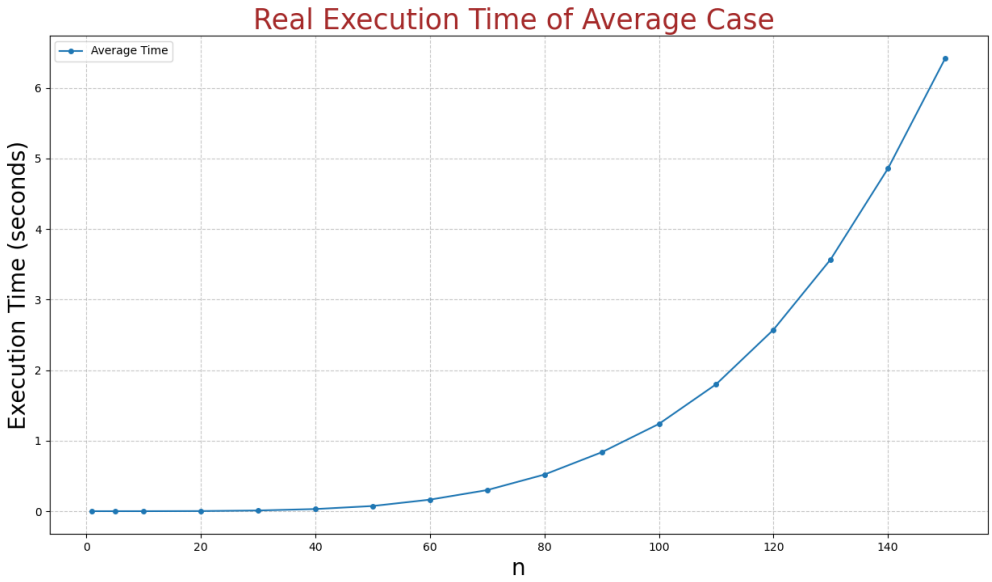



We have normalized the worst execution time graph in order to compare with theoretical analysis of (4). It can be clearly seen that these two graphs match perfectly with each other. Because, exact analysis of worst time and asymptotic approaching are close to each other.

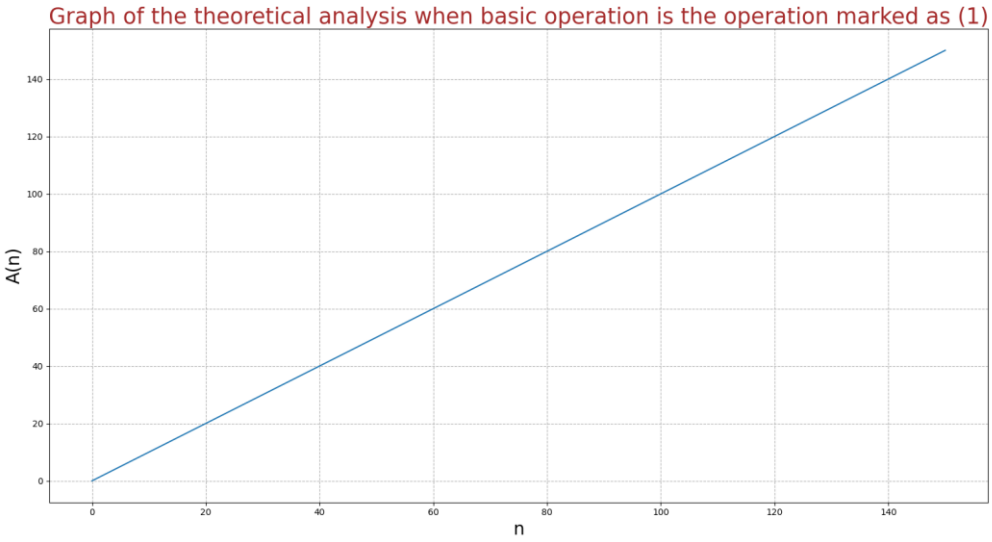
SONER KUYAR – 2021400297
YUNUS KAĞAN AYDIN – 2021400123

## Average Case

*Graph of the real execution time of the algorithm*

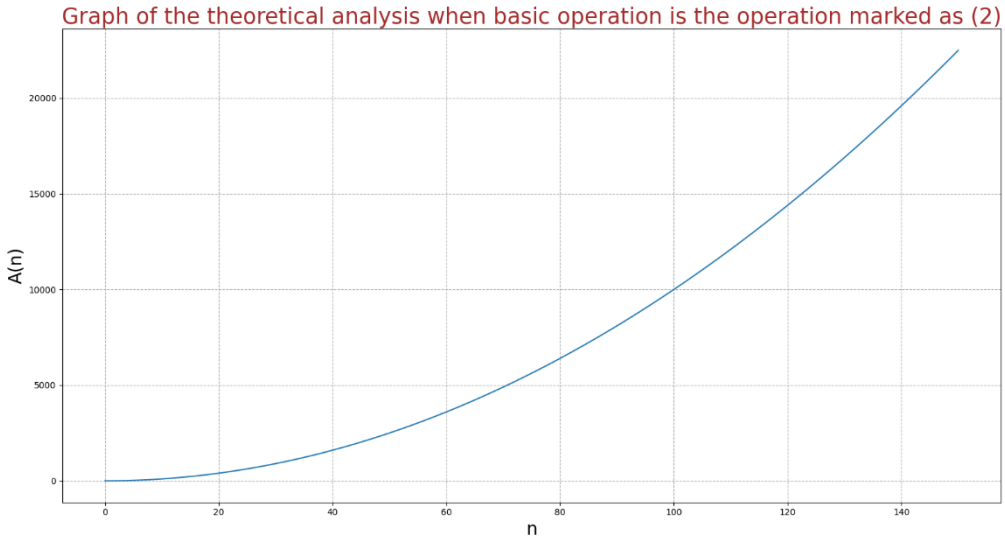



*Graph of the theoretical analysis when basic operation is the operation marked as (1)*

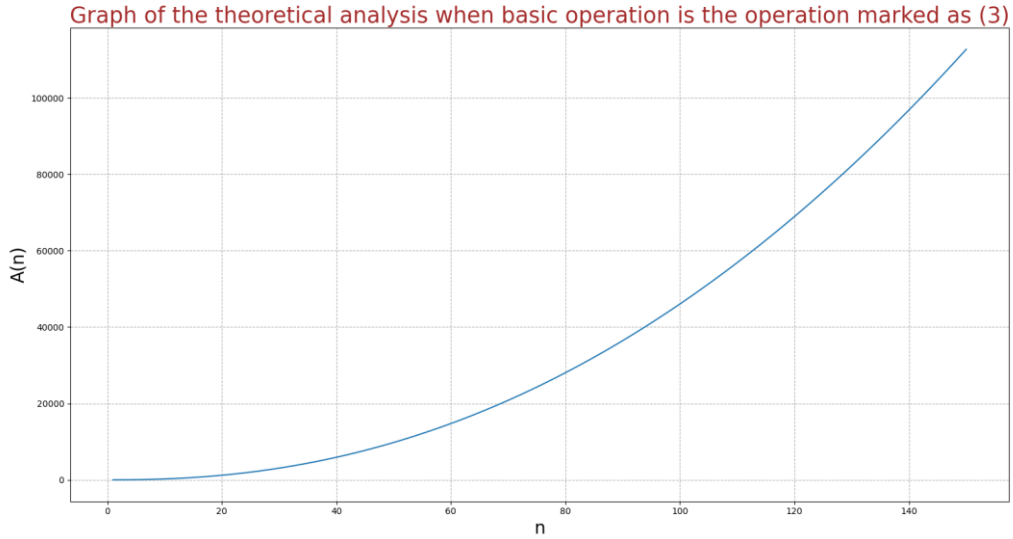


SONER KUYAR – 2021400297
YUNUS KAĞAN AYDIN – 2021400123

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

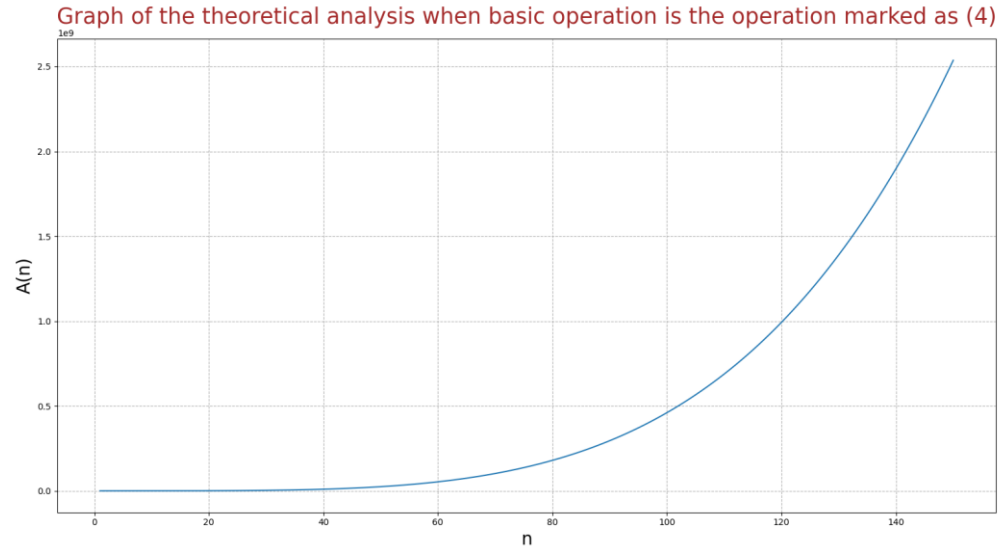



*Graph of the theoretical analysis when basic operation is the operation marked as (3):*

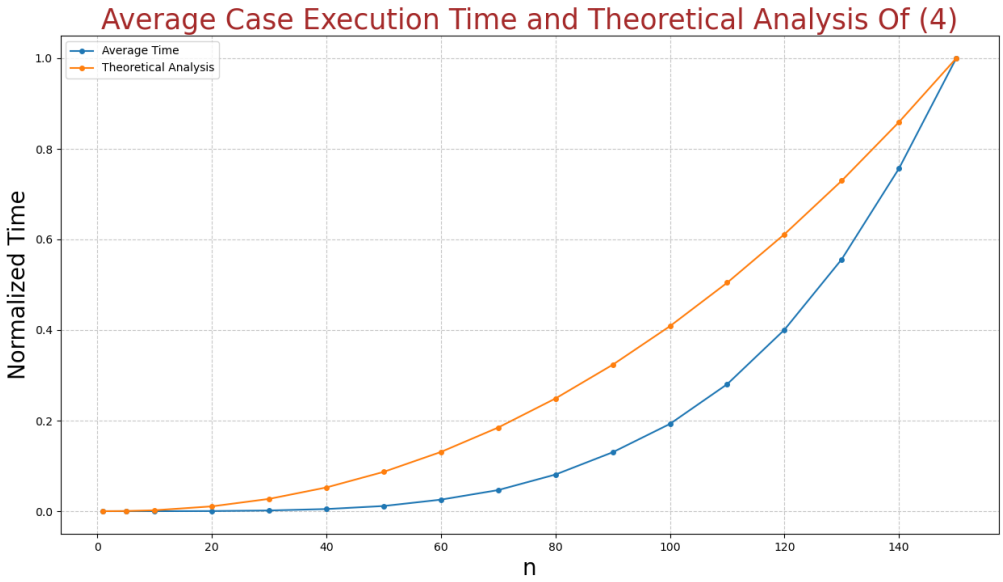


SONER KUYAR – 2021400297
YUNUS KAĞAN AYDIN – 2021400123

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




*Comments*




We have normalized the average execution time values in order to compare with theoretical analysis of (4). The reason for the gap between these values comes from the difference between exact average case analysis and theoretical average case analysis. When n reaches higher values these two graphs will come close to each other. So we can use asymptotic approximation while analyzing this algorithm at higher values of n.