Table of Contents

[THEORETICAL ANALYSIS 1](#_Toc120636788)

[Basic operation is the comparison marked as (1) 1](#_Toc120636789)

[Basic operations are the three assignments marked as (2) 2](#_Toc120636790)

[Basic operation is two assignments marked as (3) 3](#_Toc120636791)

[Basic operations are the two loop incrementations marked as (4) 5](#_Toc120636792)

[Basic operation is the assignment marked as (5) 7](#_Toc120636793)

[IDENTIFICATION OF BASIC OPERATION(S) 8](#_Toc120636794)

[REAL EXECUTION 8](#_Toc120636795)

[Best Case 8](#_Toc120636796)

[Worst Case 8](#_Toc120636797)

[Average Case 9](#_Toc120636798)

[COMPARISON 9](#_Toc120636799)

[Best Case 9](#_Toc120636800)

[Worst Case 13](#_Toc120636801)

[Average Case 16](#_Toc120636802)

## THEORETICAL ANALYSIS

* Step 1: Input Size

The input for each analysis is an array length of n.

* Step 2: Basic Operation for (1)

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

#### Analyze B(n)

* Step 3: Count

(1) is executed exactly n times regardless of the input.

* Step 4: Closed-Form

* Step 5: Asymptotic Notation

#### Analyze W(n)

* Step 3: Count

(1) is executed exactly n times regardless of the input.

* Step 4: Closed-Form
* Step 5: Asymptotic Notation

#### Analyze A(n)

* Step 3: Count

(1) is executed exactly n times regardless of the input.

* Step 4: Closed-Form
* Step 5: Asymptotic Notation
* Step 2: Basic Operation for (2)

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

#### Analyze B(n)

* Step 3: Count
* Step 4: Closed-Form
* Step 5: Asymptotic Notation

#### Analyze W(n)

* Step 3: Count
* Step 4: Closed-Form
* Step 5: Asymptotic Notation

#### Analyze A(n)

* Step 3: Count
* Step 4: Closed-Form
* Step 5: Asymptotic Notation
* Step 2: Basic Operation for (3)

### Basic operation is two assignments marked as (3)

#### Analyze B(n)

* Step 3: Count

When the input array does not contain any 0, (5) is never executed.

* Step 4: Closed-Form
* Step 5: Asymptotic Notation

#### Analyze W(n)

* Step 3: Count

When the input array is full of 2’s:

* Step 4: Closed-Form
* Step 5: Asymptotic Notation

#### Analyze A(n)

* Step 3: Count
* Step 4: Closed-Form
* Step 5: Asymptotic Notation
* Step 2: Basic Operation for (4)

### Basic operations are the two loop incrementations marked as (4)

#### Analyze B(n)

* Step 3: Count

When the input array is full of 0’s, assignments marked as (4) are never executed:

* Step 4: Closed-Form
* Step 5: Asymptotic Notation

#### Analyze W(n)

When the input array is full of 2’s:

* Step 3: Count
* Step 4: Closed-Form
* Step 5: Asymptotic Notation

#### Analyze A(n)

* Step 3: Count
* Step 4: Closed-Form
* Step 5: Asymptotic Notation
* Step 2: Basic Operation for (5)

### Basic operation is the assignment marked as (5)

#### Analyze B(n)

* Step 3: Count

When the input array does not contain any 0, (5) is never executed.

* Step 4: Closed-Form
* Step 5: Asymptotic Notation

#### Analyze W(n)

* Step 3: Count

When the input array is full of 0’s:

* Step 4: Closed-Form
* Step 5: Asymptotic Notation

#### Analyze A(n)

* Step 3: Count
* Step 4: Closed-Form
* Step 5: Asymptotic Notation

## 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)*

Three assignments in (2) are basic operations. Because at least one of these operations always contributes to an algorithm's total running time. Also, second assignment in (2) is typically the most time-consuming operation in the algorithm’s innermost loop by being inside of 3 for loops and 1 while loop.

## REAL EXECUTION

### Best Case

|  |  |
| --- | --- |
| N Size | Time Elapsed |
| 1 | 0.00000476837158203125 |
| 5 | 0.000010967254638671875 |
| 10 | 0.00003814697265625 |
| 25 | 0.0002429485321044922 |
| 50 | 0.0011188983917236328 |
| 75 | 0.002679109573364258 |
| 100 | 0.004364013671875 |
| 150 | 0.011707067489624023 |
| 200 | 0.02629995346069336 |
| 250 | 0.036992788314819336 |

### Worst Case

|  |  |
| --- | --- |
| N Size | Time Elapsed |
| 1 | 0.0000011920928955078125 |
| 5 | 0.000041961669921875 |
| 10 | 0.00040078163146972656 |
| 25 | 0.013779878616333008 |
| 50 | 0.18341803550720215 |
| 75 | 0.9667437076568604 |
| 100 | 2.958270788192749 |
| 150 | 15.30362319946289 |
| 200 | 48.0098979473114 |
| 250 | 121.85824203491211 |

### Average Case

|  |  |
| --- | --- |
| N Size | Time Elapsed |
| 1 | 0.000002384185791015625 |
| 5 | 0.00003377596537272135 |
| 10 | 0.00031336148579915363 |
| 25 | 0.008852005004882812 |
| 50 | 0.09887814521789551 |
| 75 | 0.4282924334208171 |
| 100 | 1.199751853942871 |
| 150 | 6.5252476533253985 |
| 200 | 19.19698127110799 |
| 250 | 41.276151180267334 |

## 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)

#### 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)

#### Graph of the theoretical analysis when basic operation is the operation marked as (5)

#### Comments

For Best Case, we have 1 real measurement and 5 theoretical analyzes. To reach the real values, we coded the algorithm in Python, recorded the time value for 10 different input sizes and graphed it. Likewise, in the theoretical part, we found the count of basic operations specified for 10 different input sizes. To find the count, we used the equations we found with our analysis and converted the data into 5 different graphs.

We can state that the number of real basic operations in this algorithm will be proportional to the running speed of the algorithm. Therefore, we can interpret the time graph and the number of basic operations graph on the same plane. We see the constant 0 value in the 3rd, 4th and 5th graphs, and the 1st graph is quite far from the real result. So as a result, basic operations marked as (2) seems to be the choice of basic operations that gives the closest result to reality.

When we evaluate these discourses, for the Best Case we can see that real basic operations create the most similar graph to the graph created by real time in our calculations.

### 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)

#### Graph of the theoretical analysis when basic operation is the operation marked as (5)

#### Comments

For Worst Case, we have 1 real measurement and 5 theoretical analyzes. To reach the real values, we coded the algorithm in Python, recorded the time value for 10 different input sizes and graphed it. Likewise, in the theoretical part, we found the count of basic operations specified for 10 different input sizes. To find the count, we used the equations we found with our analysis and converted the data into 5 different graphs.

We can state that the number of real basic operations in this algorithm will be proportional to the running speed of the algorithm. Therefore, we can interpret the time graph and the number of basic operations graph on the same plane. When we look at the graphs this time, we see that although the 1st graph is quite far from the real graph, the 2nd, 3rd, 4th and 5th are closer to each other. We can attribute this to the fact that the basic operations in the last 4 graphs take place in the inner loops compared to the 1st graph. However, when we examine the last 4 graphs carefully, basic operations marked as (2) seems to be the choice of basic operations that gives the closest result to reality.

When we evaluate these discourses, for the Worst Case we can see that real basic operations create the most similar graph to the graph created by real time in our calculations.

### 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)

#### Graph of the theoretical analysis when basic operation is the operation marked as (5)

#### Comments

For Average Case, we have 1 real measurement and 5 theoretical analyzes. To reach the real values, we coded the algorithm in Python, recorded the time value for 10 different input sizes and graphed it. Likewise, in the theoretical part, we found the count of basic operations specified for 10 different input sizes. To find the count, we used the equations we found with our analysis and converted the data into 5 different graphs.

We can state that the number of real basic operations in this algorithm will be proportional to the running speed of the algorithm. Therefore, we can interpret the time graph and the number of basic operations graph on the same plane. When we look at the graphs this time, we see that although the 1st graph is quite far from the real graph, the 2nd, 3rd, 4th and 5th are closer to each other. We can attribute this to the fact that the basic operations in the last 4 graphs take place in the inner loops compared to the 1st graph. However, when we examine the last 4 graphs carefully, basic operations marked as (2) seems to be the choice of basic operations that gives the closest result to reality.

When we evaluate these discourses, for the Average Case we can see that real basic operations create the most similar graph to the graph created by real time in our calculations.