Muhammet Ali Topcu – 2020400147
Abdullah Enes Güleş – 2021400135

Table of Contents

Muhammet Ali Topcu – 2020400147
Abdullah Enes Güleş – 2021400135

THEORETICAL ANALYSIS

Basic operation is the comparison marked as (1)

Analyze B(n)

B(n) = $\sum_{i=0}^{n-1} 1$ = n
So, for the best case, there exists n basic operations.
B(n) ∈ θ(n)

Analyze W(n)

W(n) = $\sum_{i=0}^{n-1} 1$ = n
So, for the worst case, there exists n basic operations.
W(n) ∈ θ(n)

Analyze A(n)

A(n) = $\sum_{i=0}^{n-1} 1$ = n
So, for the average case, there exists n basic operations.
A(n) ∈ θ(n)

Basic operations are the two assignments marked as (2)

Analyze B(n)

For the best case, if all of the elements in the array are 2, (i.e. [2,2,2,…]), then operation (2) will never executes. So, it will execute 0 basic operations.

Then, B(n) ∈ O(1)

Analyze W(n)

For the worst case, the array shouldn't include any 2 element in it, so that the count of basic operations is maximized. In the first if block and the second else if block, for constructs works in the same way. So, we need an array that consists of 0s and 1s, no matter how many of them are 0 or 1.

W(n) = $\sum_{i=0}^{n-1} \sum_{j=i}^{n-1} 1 = \sum_{i=0}^{n-1} \sum_{m=i}^{n-1} 1 = \sum_{i=0}^{n-1} n - i = \frac{n(n+1)}{2}$

So, W(n) ∈ θ(n^2)

Analyze A(n)

A(n) = E[T] = E[T_1] + E[T_2] where

T_1= number of times first assignment executed (i.e. first if statement is true)

T_2= number of times second assignment executed (i.e. else if statement is true)

Let X denotes the array's ith element, i.e. X = arr[i]

E[T_1] = $\sum_{i=0}^{n-1}$ E[T_1 | X = 0] * P(X = 0)

E[T_1 | X = 0] = $\sum_{j=i}^{n-1} 1$ = n – i

$P(X = 0) = \frac{1}{3}$

Then, $\sum_{i=0}^{n-1} E[T_1 \mid X = 0] * P(X = 0) = \sum_{i=0}^{n-1} (n - i) * \frac{1}{3} = \frac{1}{3} * \frac{n(n+1)}{2} = \frac{n(n+1)}{6}$

$E[T_2] = \sum_{i=0}^{n-1} E[T_2 \mid X = 1] * P(X = 1)$

$E[T_2 \mid X = 1] = \sum_{m=i}^{n-1} 1 = n - i$

$P(X = 0) = \frac{1}{3}$

Then, $\sum_{i=0}^{n-1} E[T_1 \mid X = 0] * P(X = 0) = \sum_{i=0}^{n-1} (n - i) * \frac{1}{3} = \frac{1}{3} * \frac{n(n+1)}{2} = \frac{n(n+1)}{6}$

$A(n) = E[T] = E[T_1] + E[T_2] = \frac{n(n+1)}{3}$.

$A(n) \in \theta(n^2)$

Basic operations are the two comparisons marked as (3)

Analyze B(n)

For the best case, if all of the elements in the array are 1, (i.e. [1,1,1,…]), then operation (3) will never executes. So, it will execute 0 basic operations.

Then, $B(n) \in O(1)$

Analyze W(n)

For the worst case analysis, since else if block doesn't contain any (3) basic operation, we don't need to calculate it.

Assume $n = 3^k$ $(k > 0)$. So $k = log_3 n$.

For the first if block, for one pass of the for loop, there exists

$\sum_{j=i}^{n-1} \sum_{t=0}^{k+1} 1 = \sum_{j=i}^{n-1} (k+2) = \sum_{j=i}^{n-1} (log_3 n + 2)$

$= (n - i)(log_3 n + 2)$ basic operations.

For the else block, for one pass of the for loop, there exists

$\sum_{p=0}^{n} 1 = n+1$ basic operations.

We need to find a formula for i as a function of n.

$(n - i)\ (log_3 n + 2) = n+1$

$n log_3 n + 2n - i log_3 n - 2i = n + 1$

$n log_3 n + n - i log_3 n - 2i = 1$

$i = n - \frac{n+1}{log_3 n+2}$

For lower values of i, $(n - i)(log_3 n + 2) > (n+1)$. Therefore, we must fill the array initially with 0s. At the point where $i \geq n - \frac{n+1}{log_3 n+2}$, we need to start to fill the remaining part of the array with 2s.

So, $W(n) = \sum_{i=0}^{\lfloor n - \frac{n+1}{log_3 n+2} \rfloor} (n - i)(log_3 n + 2) + \sum_{i=\lfloor n - \frac{n+1}{log_3 n+2} \rfloor + 1}^{n} (n + 1)$

Assume $\left\lfloor n - \frac{n+1}{log_3 n+2} \right\rfloor = n - \frac{n+1}{log_3 n+2}$

Then, $W(n) = \sum_{i=0}^{n - \frac{n+1}{log_3 n+2}} (n - i)(log_3 n + 2) + \sum_{i=n - \frac{n+1}{log_3 n+2} + 1}^{n} (n + 1)$

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

$$(log_3 n + 2) * \left(n - \frac{n+1}{log_3 n+2} + 1\right) * \left(n - \frac{n+1}{log_3 n+2}\right) * \left(\frac{1}{2}\right)$$

$$= (log_3 n + 2) * \left(n - \frac{n+1}{log_3 n+2} + 1\right) \left[n - \frac{1}{2}n + \frac{1}{2}\frac{n+1}{log_3 n+2}\right]$$

$$= [(n + 1)(log_3 n + 2) - (n + 1)] * \frac{1}{2}\left(n + \frac{n+1}{log_3 n+2}\right)$$

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

So, $W(n) = [(n + 1)(log_3 n + 2) - (n + 1)] * \frac{1}{2}\left(n + \frac{n+1}{log_3 n+2}\right)$

$$+ (n+1)\left(\frac{n+1}{log_3 n+2} - 1\right)$$

$$= (n+1) * \left[\frac{1}{2}(log_3 n + 1)\left(n + \frac{n+1}{log_3 n+2}\right) + \left(\frac{n+1}{log_3 n+2} - 1\right)\right]$$

$$= (n+1) * \left[\frac{1}{2}\left(n log_3 n + n + \frac{(n+1)(log_3 n+1)}{log_3 n+2}\right) + \left(\frac{n+1}{log_3 n+2} - 1\right)\right]$$

$$= \frac{1}{2}(n + 1)(2n + n log_3 n)$$

$= \frac{1}{2}\ (n^2 log_3 n + 2n^2 + n log_3 n + 2n)$

W(n) ∈ θ($n^2 log_3 n$)

Analyze A(n)

A(n) = E[T] = E[T_1] + E[T_2] where

T_1= number of times first comparison executed (i.e. "while k > 0 do" statement)

T_2= number of times second comparison executed (i.e. "while p < n do" statement)

Assume n = 3^k (k > 0). So k = $log_3 n$.

Let X denotes the array's ith element, i.e. X = arr[i]

E[T_1] = $\sum_{i=0}^{n-1}$ E[T_1 | X = 0] * P(X = 0)

E[T_1 | X = 0] = $\sum_{j=i}^{n-1}$ $\sum_{t=0}^{k+1} 1$ = $\sum_{j=i}^{n-1}$ (k+2) = $\sum_{j=i}^{n-1}$ ($log_3 n$ + 2)

= (n - i)($log_3 n + 2$)

P(X = 0) = $\frac{1}{3}$

Then, $\sum_{i=0}^{n-1}$ E[T_1 | X = 0] * P(X = 0) = $\sum_{i=0}^{n-1}$ (n - i)($log_3 n + 2$)* $\frac{1}{3}$

= $\frac{1}{3}$ *($log_3 n + 2$) * $\frac{n(n+1)}{2}$ = $\frac{1}{3}$ *($log_3 n + 2$) * $\frac{(n^2+n)}{2}$

E[T_2] = $\sum_{i=0}^{n-1}$ E[T_2 | X = 2] * P(X = 2)

E[T_2 | X = 2] = $\sum_{p=0}^{n}$ 1 = n+1

P(X = 0) = $\frac{1}{3}$

Then, $\sum_{i=0}^{n-1}$ E[T_2 | X = 2] * P(X = 2) = $\sum_{i=0}^{n-1}$ (n+1) * $\frac{1}{3}$ = $\frac{n(n+1)}{3}$ = $\frac{(n^2+n)}{3}$

A(n) = E[T] = E[T_1] + E[T_2] = $\frac{1}{3}$ *[($log_3 n + 2$) * $\frac{(n^2+n)}{2}$ + ($n^2 + n$)]

= $\frac{(n^2+n)}{3}$ *[$\frac{(log_3 n+2)}{2} + 1$]

A(n) ∈ θ($n^2 log_3 n$)

Basic operations are the three assignments marked as (4)

Before starting the analysis case by case, we can first calculate the count of basic operations (4) executed in each block.

For the first if block:

Assume n = 3^k (k > 0). So k = $log_3 n$.

$\sum_{j=i}^{n-1} \sum_{t=0}^{k} 1 = \sum_{j=i}^{n-1}$ (k+1) = $\sum_{j=i}^{n-1} (log_3 n + 1)$

= (n - i)$(log_3 n + 1)$

For the else if block:

$\sum_{m=i}^{n-1} \sum_{l=m}^{n-1} \sum_{t=1}^{n} \sum_{z=1}^{\lfloor \frac{n}{t} \rfloor} 1$.

Assume for simplicity that $\left\lceil \frac{n}{t} \right\rceil = \frac{n}{t}$

Then, $\sum_{m=i}^{n-1} \sum_{l=m}^{n-1} \sum_{t=1}^{n} \sum_{z=1}^{\frac{n}{t}} 1 = \sum_{m=i}^{n-1} \sum_{l=m}^{n-1} \sum_{t=1}^{n} \frac{n}{t} = \sum_{m=i}^{n-1} \sum_{l=m}^{n-1} n * H(n)$.

Assume H(n) = logn.

Then, $\sum_{m=i}^{n-1} \sum_{l=m}^{n-1} n * H(n) = \sum_{m=i}^{n-1} \sum_{l=m}^{n-1} n * logn = \sum_{m=i}^{n-1}$ (n − m) $* n * logn$

$= n * logn * [n * (n - i) - \sum_{m=i}^{n-1} \mathrm{m}] = n * logn * (n - i)[\frac{n-i+1}{2}]$

For the else block:

$\sum_{p=0}^{n} \sum_{j=0}^{p^2-1} 1 = \sum_{p=0}^{n} p^2 = \frac{n(n+1)(2n+1)}{6}$

Analyze B(n)

If we simply compare the inner blocks

Comparing if and else if blocks:

(n – i)$log_3 n < n * logn * (n - i)[\frac{n-i+1}{2}]$

1 < n*$[\frac{n-i+1}{2}]$, which is always satisfied. So, if block has lower order compared to else if block.

In addition, for the smallest i, i.e. i=0,

if block executes n*logn

else if block executes n*logn*n*$\frac{n+1}{2}$

else block executes $\frac{n(n+1)(2n+1)}{6}$

It is clear that in all (i,n) cases, first block executes the smallest time. So, we should fill the array with 0s for the best case.

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

So, B(n) ∈ θ$(n^2 log_3 n)$

Analyze W(n)

Since in the best case, we found that the array will be full of 0s. So, in the worst case, we shouldn't add 0 to the array. We need to find an array with 1s and 2s that maximizes the execution time.

Comparing the else if block and else block:

For smaller values of i, let's say i=0,

Check whether $\frac{n(n+1)(2n+1)}{6} < n * logn * (n - i) \left[\frac{n-i+1}{2}\right]$ true

$\frac{(2n+1)}{3}$ < n*log(n)

As n increases, right-handside increases faster. Therefore, initially, we should fill the array with 1s. At some point, since the right-handside includes "-i", we need to start filling with 2s.

To find that point, we need to solve,

$\frac{n(n+1)(2n+1)}{6} = n * logn * (n - i) \left[\frac{n-i+1}{2}\right]$

for i.

After calculations,

i = (3 log(n) + 6 n log(n) + $\sqrt[2]{3}$

($\sqrt[2]{(4 \log(x) + 12 x \log(x) + 8 x\text{^}2 \log(x) + 3 \log\text{^}2(x))}$)/(6 log(x))

i= $\frac{3logn + 6nlogn - \sqrt[2]{3}\left(\sqrt[2]{(4logn + 12nlogn + 8n^2 logn + 3\,(logn)^2)}\right)}{6logn}$

So, W(n)

$= \sum_{i=0}^{\left\lfloor \frac{3logn + 6nlogn - \sqrt[2]{3}\left(\sqrt[2]{(4logn + 12nlogn + 8n^2 logn + 3\,(logn)^2)}\right)}{6logn} \right\rfloor} n * logn * (n - i)\left[\frac{n-i+1}{2}\right]$

$$+\sum_{i=\lfloor \frac{3logn + 6nlogn - \sqrt[2]{3}\left(\sqrt[2]{(4logn + 12nlogn + 8n^2logn + 3\,(logn)^2)}\right)}{6logn} \rfloor + 1}^{n} \frac{n(n+1)(2n+1)}{6}$$

W(n) ∈ θ(n^4log$n$)

Analyze A(n)

A(n) = E[T] = E[T_1] + E[T_2] + E[T_3] where

T_1= number of times first assignment executed

T_2= number of times second assignment executed

T_3= number of times third assignment executed

Assume n = 3^k (k > 0). So k = $log_3 n$.

Let X denotes the array's ith element, i.e. X = arr[i]

E[T_1] = $\sum_{i=0}^{n-1}$ E[T_1 | X = 0] * P(X = 0)

E[T_1 | X = 0] = (n - i)($log_3 n + 1$) (calculated above)

P(X = 0) = $\frac{1}{3}$

Then, $\sum_{i=0}^{n-1}$ E[T_1 | X = 0] * P(X = 0) = $\sum_{i=0}^{n-1}$ (n - i)($log_3 n + 1$)* $\frac{1}{3}$

= $\frac{1}{3}$ *($log_3 n + 1$) * $\frac{n(n+1)}{2}$ = $\frac{1}{3}$ *($log_3 n + 1$) * $\frac{(n^2+n)}{2}$

E[T_2] = $\sum_{i=0}^{n-1}$ E[T_2 | X = 1] * P(X = 1)

E[T_2 | X = 1] = $n * logn * (n - i)\left[\frac{n-i+1}{2}\right]$ (calculated above)

P(X = 1) = $\frac{1}{3}$

Then, $\sum_{i=0}^{n-1}$ E[T_1 | X = 1] * P(X = 1) = $\sum_{i=0}^{n-1}$ $\frac{1}{3} * n * logn * (n - i)\left[\frac{n-i+1}{2}\right]$

= $\sum_{i=0}^{n-1}$ $\frac{1}{3} * n * logn * \left[\frac{n^2-2ni+n+i^2-i}{2}\right]$

= $\frac{1}{3}$ * [$logn$*($\frac{n^4+n^3}{2}$) + $\frac{logn}{2}$*($\sum_{i=0}^{n-1}$ $-2n^2 i + i^2 n - ni$)]

= $\frac{1}{3}$ * [$logn$*($\frac{n^4+n^3}{2}$) + $\frac{logn}{2}$*($\frac{(-2n^2-n)n(n-1)}{2}$)+ $\frac{n(n-1)n(2n-1)}{6}$]

= $\frac{1}{3}$ * [$logn$*($\frac{n^4+n^3}{2} - \frac{2n^4-n^3-n^2}{4} + \frac{2n^4-3n^3+n^2}{12}$)]

$= \frac{1}{3} * [\frac{logn}{12} * (2n^4 + 6n^3 + 4n^2)]$

$E[T_3] = \sum_{i=0}^{n-1} E[T_3 \mid X = 2] * P(X = 2)$

$E[T_3 \mid X = 2] = \frac{2n^3+3n^2+n}{6}$

$P(X = 2) = \frac{1}{3}$

Then, $\sum_{i=0}^{n-1} E[T_2 \mid X = 2] * P(X = 2) = \sum_{i=0}^{n-1} \frac{1}{3} * \frac{2n^3+3n^2+n}{6} = \frac{2n^4+3n^3+n^2}{18}$

$A(n) = E[T] = E[T_1] + E[T_2] + E[T_3] = \frac{1}{3} * [(log_3 n + 1) * \frac{(n^2+n)}{2} + \frac{logn}{12} * (2n^4 + 6n^3 + 4n^2) + \frac{2n^3+3n^2+n}{6}]$

$A(n) \in \theta(n^4 \log n)$

IDENTIFICATION OF BASIC OPERATION(S)

Since the performance of an algorithm should be measured in terms of a basic operation, rather than all types of operations. The basic operation must be the most important and recurrent operation that contributes the most to the total execution time. Therefore, the correct basic operation should be (4).

REAL EXECUTION

Best Case

N Size	Time Elapsed
1	2.500000000016378e-06
5	4.900000000002125e-06
10	1.2900000000010126e-05
20	4.4600000000005746e-05
30	0.0001501000000000141
40	0.0002234999999999876
50	0.0003272000000000002747
60	0.0004685999999999302
70	0.0007291999999998744
80	0.0008219999999998784
90	0.0012341000000004598
100	0.0015672000000002129

110	0.0018228999999987394
120	0.002126799999999207
130	0.0026034999999993147
140	0.0028883999999962384
150	0.003322400000001835

Worst Case

N Size	Time Elapsed
1	1.700000000021129e-06
5	5.990000000000162e-05
10	0.0005741999999999969
20	0.008635500000000018
30	0.04529960000000002
40	0.1353661
50	0.3347411
60	0.6896737999999999
70	1.2693172
80	2.1751240000000003
90	3.4983033
100	5.4087327
110	7.999686999999998
120	11.283970700000001
130	15.583498900000002
140	21.266430800000002
150	28.4524962

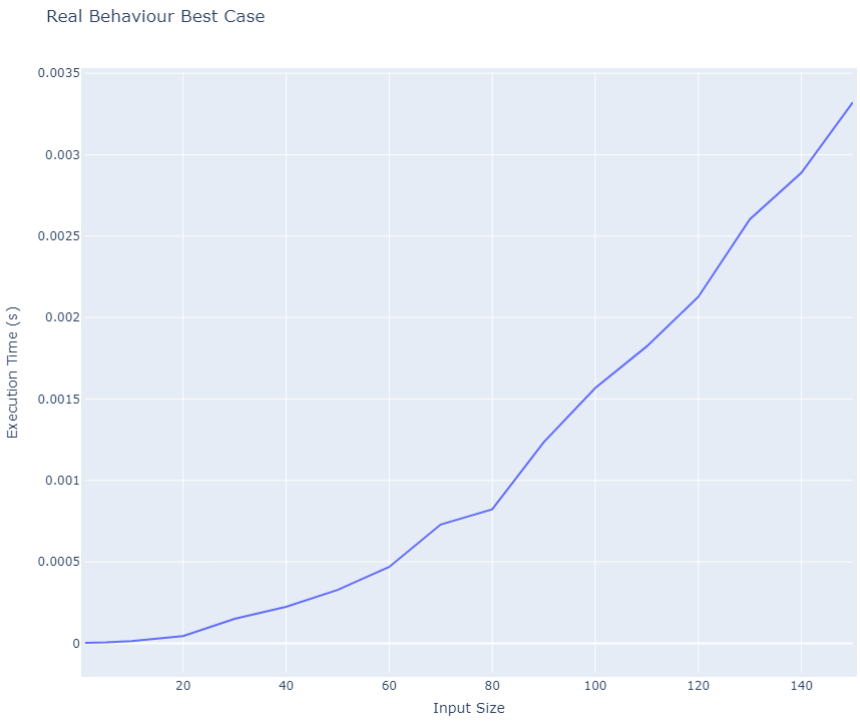
Average Case

N Size	Time Elapsed
1	3.600000000006376e-06
5	1.8900000000005023e-05
10	0.0003170100000000009
20	0.003883719999999999
30	0.017984730000000004
40	0.051483330000000015
50	0.13484727
60	0.25211634999999993
70	0.46691844000000016
80	0.8610922799999992
90	1.459513570000001
100	2.1710280399999986
110	3.07327377
120	4.3023081
130	6.393240019999996
140	7.695921159999989
150	10.922374860000009

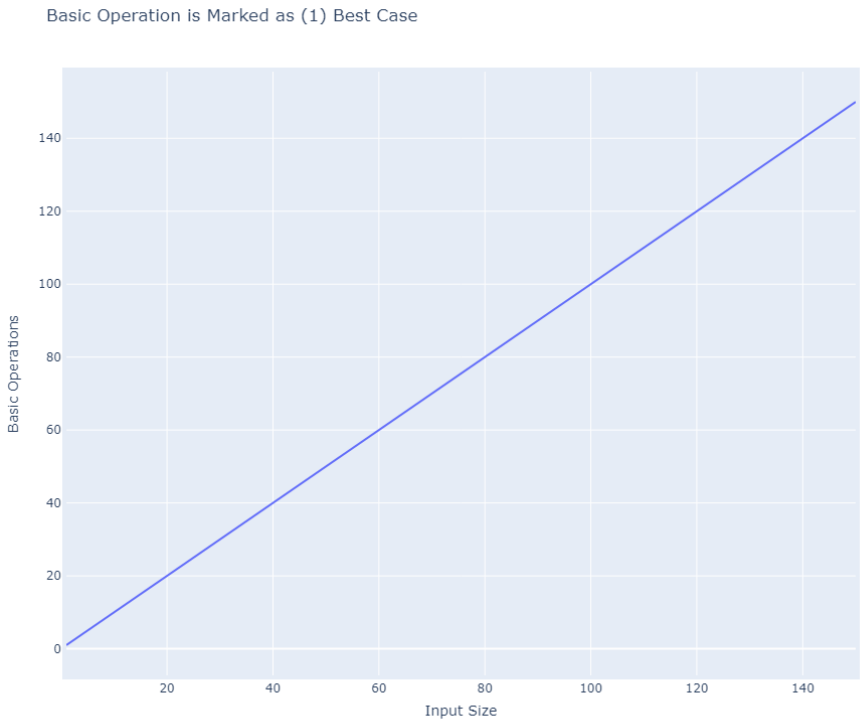
COMPARISON

Best Case

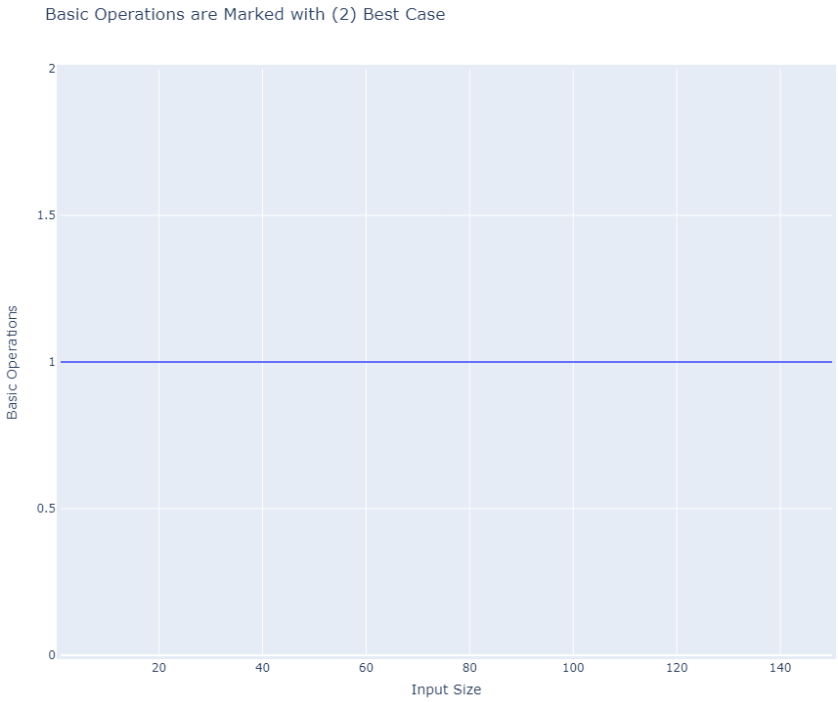
Graph of the real execution time of the algorithm



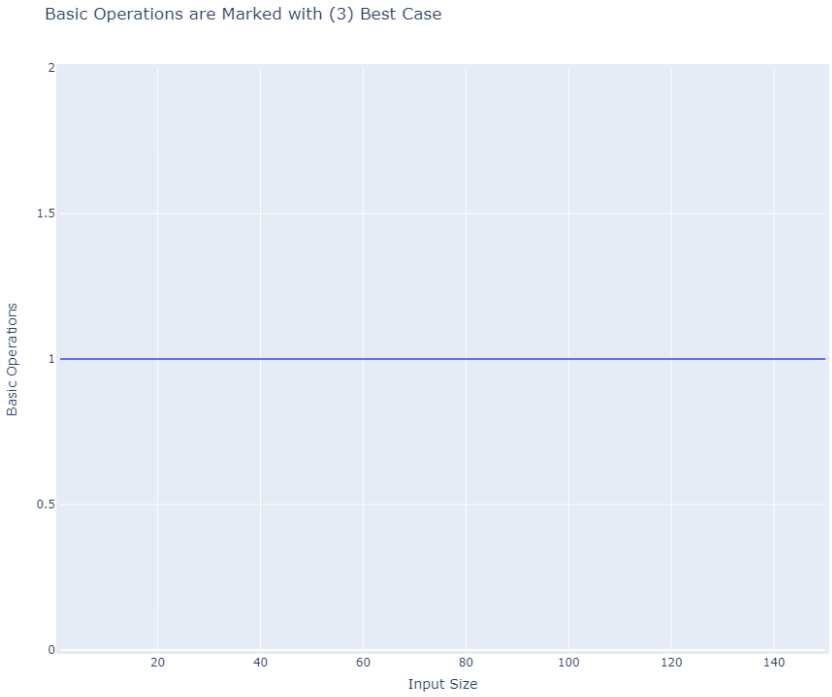
Graph of the theoretical analysis when basic operation is the operation marked as (1)



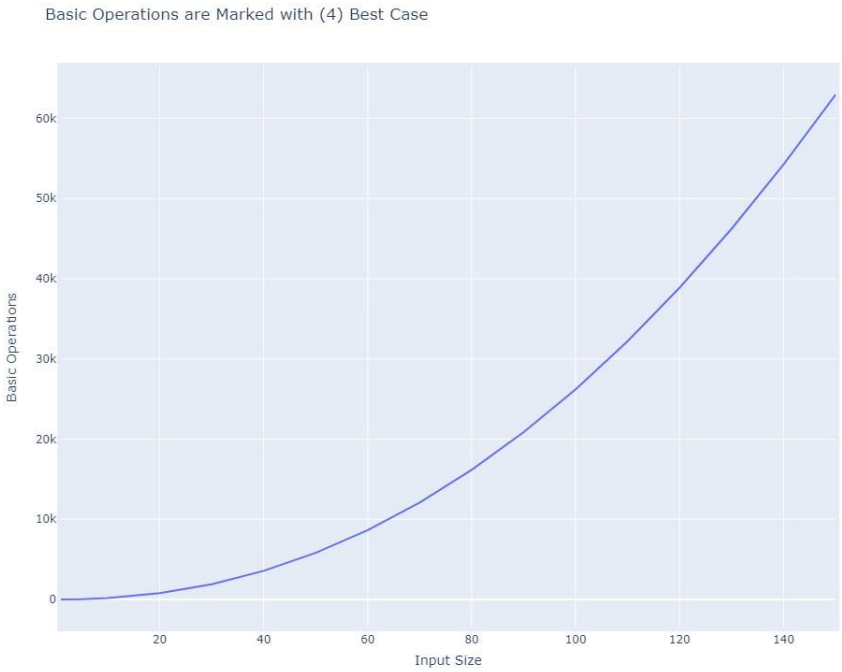
Graph of the theoretical analysis when basic operation is the operation marked as (2)



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)

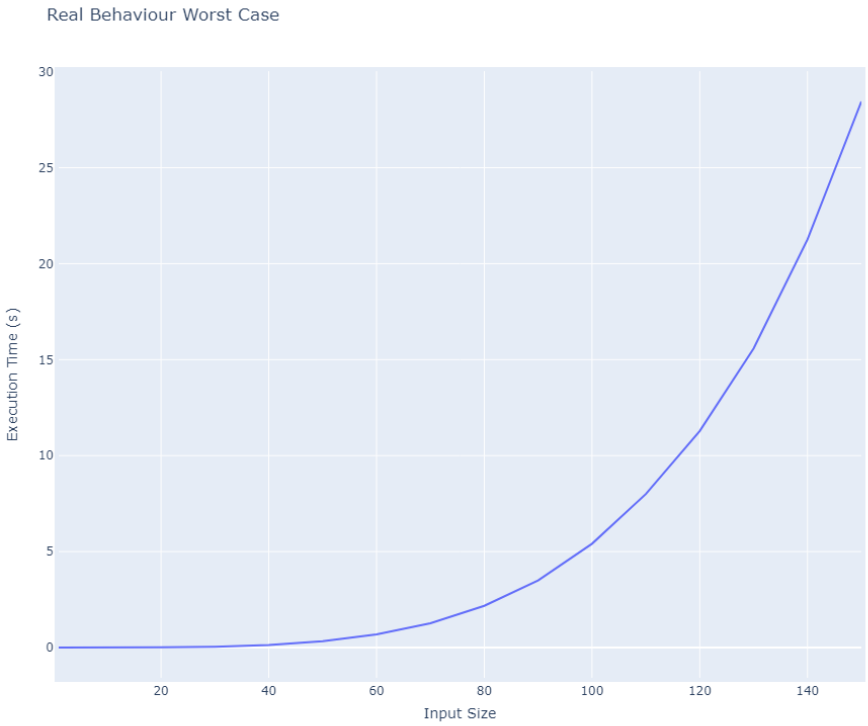


Comments

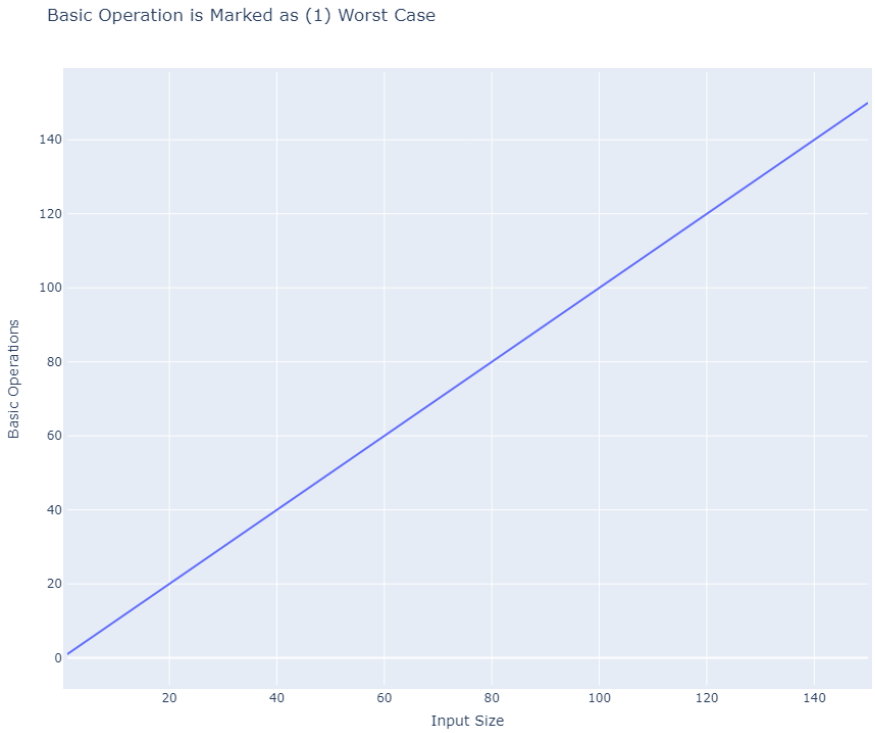
By looking at the graphs above, we can say that choosing the basic operation wisely can give us more realistic information about the actual running time of the algorithm. If we were to choose the basic operation as (2) or (3), our theoretical analysis would mislead us. Choosing (1) as the basic operation is better for best case when compared with (2) or (3). However, choosing (4) is the best basic operation choice since it is the operation which contributes most to the total execution time and which is the most recurrent operation in the algorithm.

Worst Case

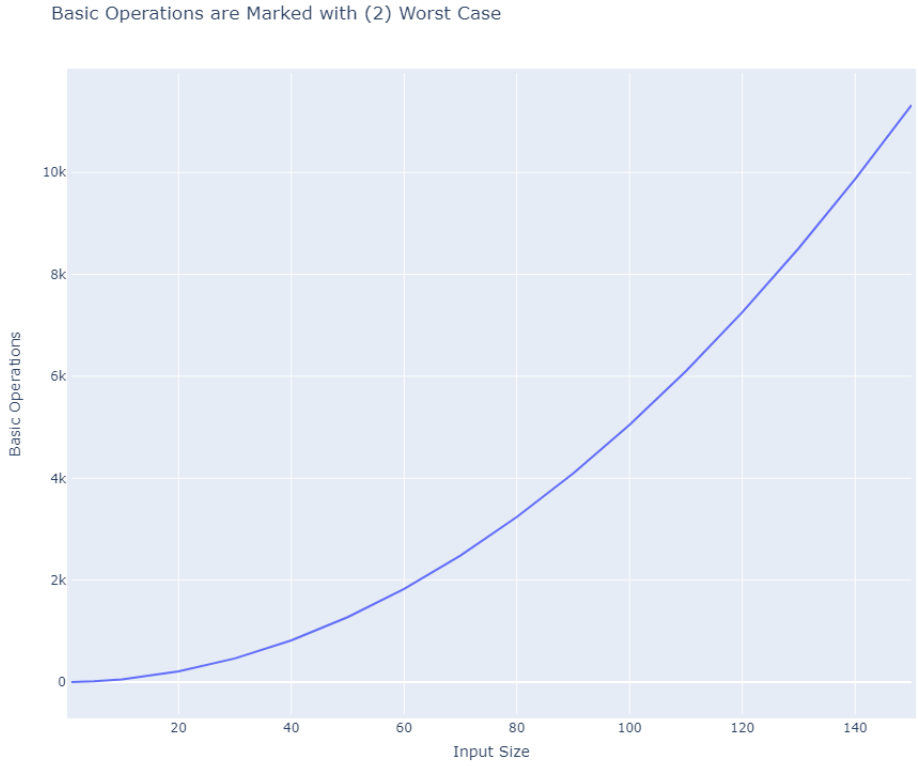
Graph of the real execution time of the algorithm



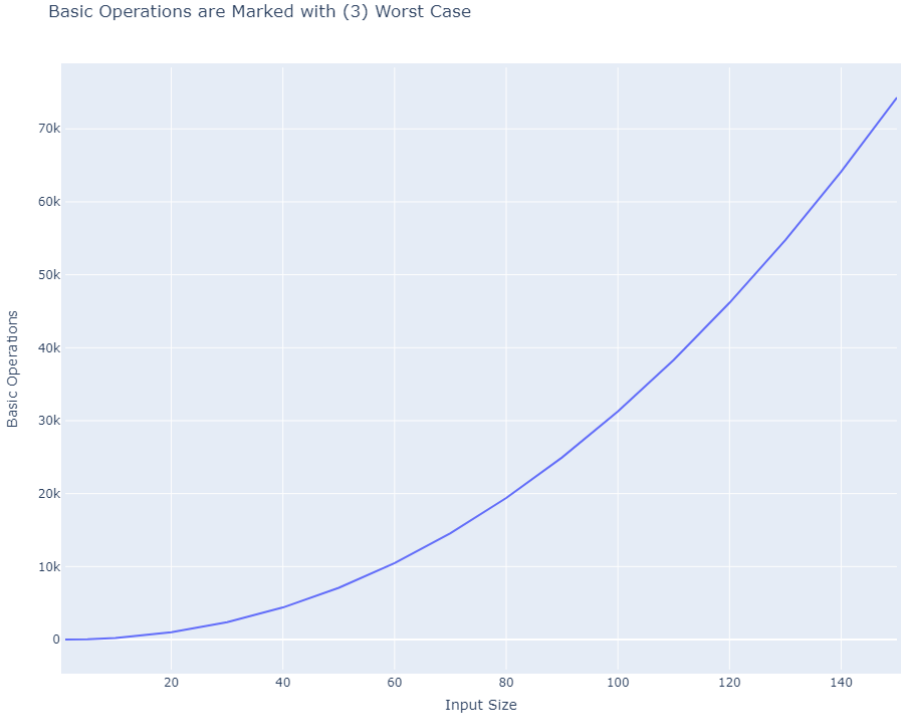
Graph of the theoretical analysis when basic operation is the operation marked as (1)



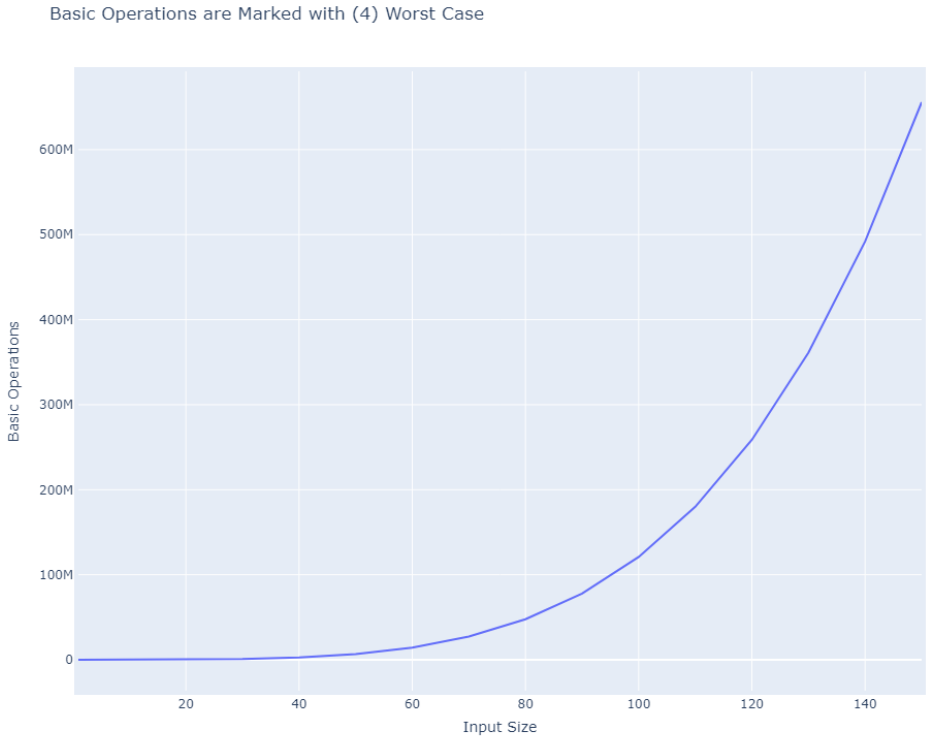
Graph of the theoretical analysis when basic operation is the operation marked as (2)



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)

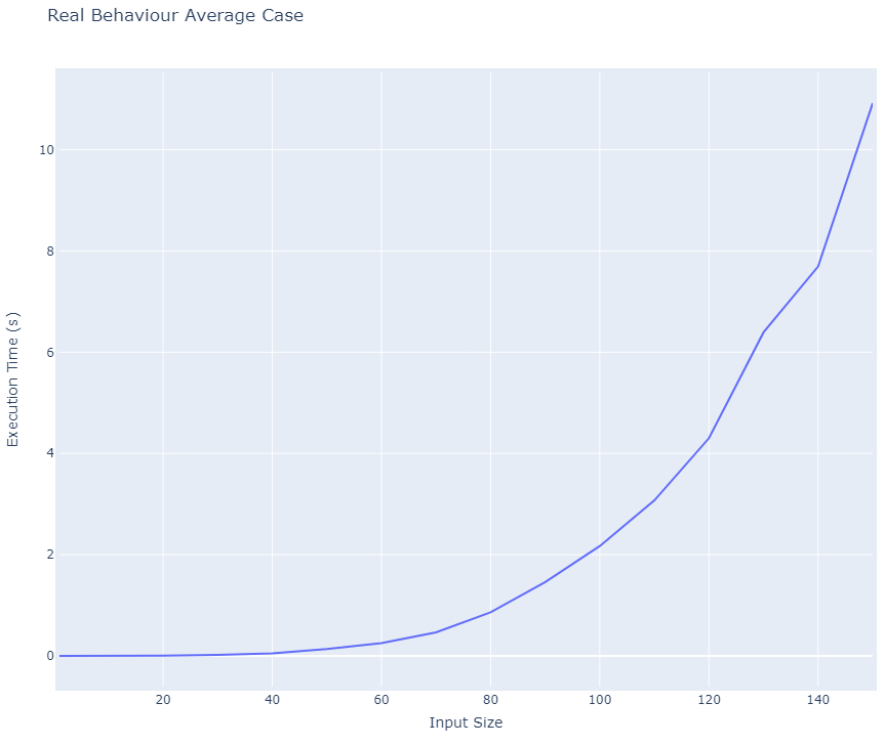


Comments

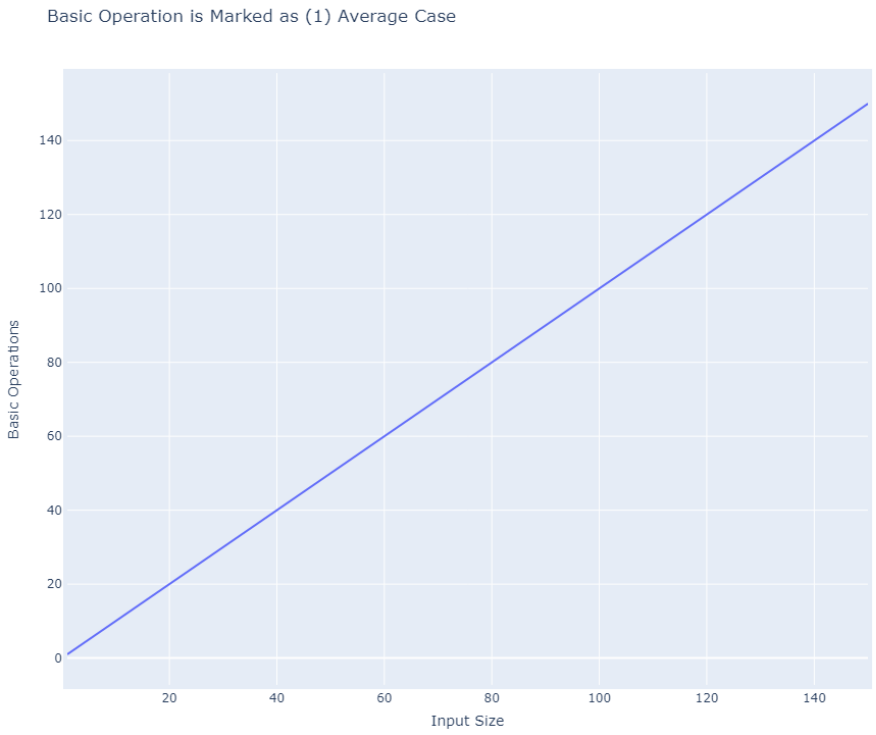
By looking at the graphs above, and comparing them with the best-case analysis graphs, we can say that worst-case analysis gives more realistic information about the real execution time of the algorithm than the best-case analysis. Actually, this depends on the algorithm executed. Since in this algorithm, for the best-case analysis, there were some basic operations that never executes, best-case analysis misled us. So, worst-case analysis is better to estimate the real execution time of this algorithm. In addition, as we move from basic operation (1) to (4), our analysis's graph approaches to the real execution graph of the algorithm. As a result, we can say that choosing (4) as basic operation was a correct choice for the algorithm.

Average Case

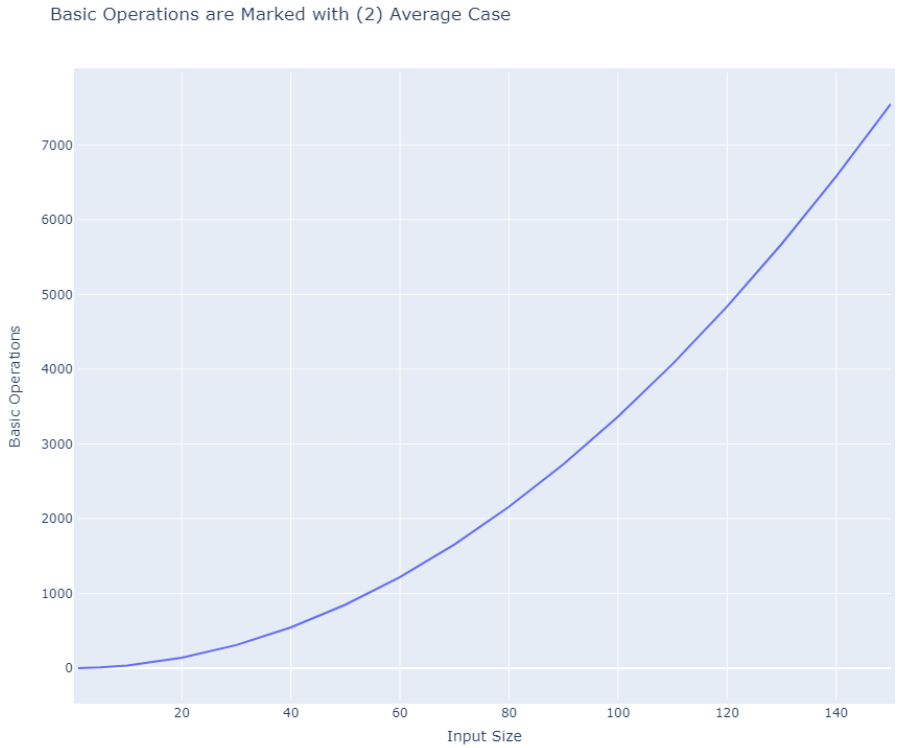
Graph of the real execution time of the algorithm



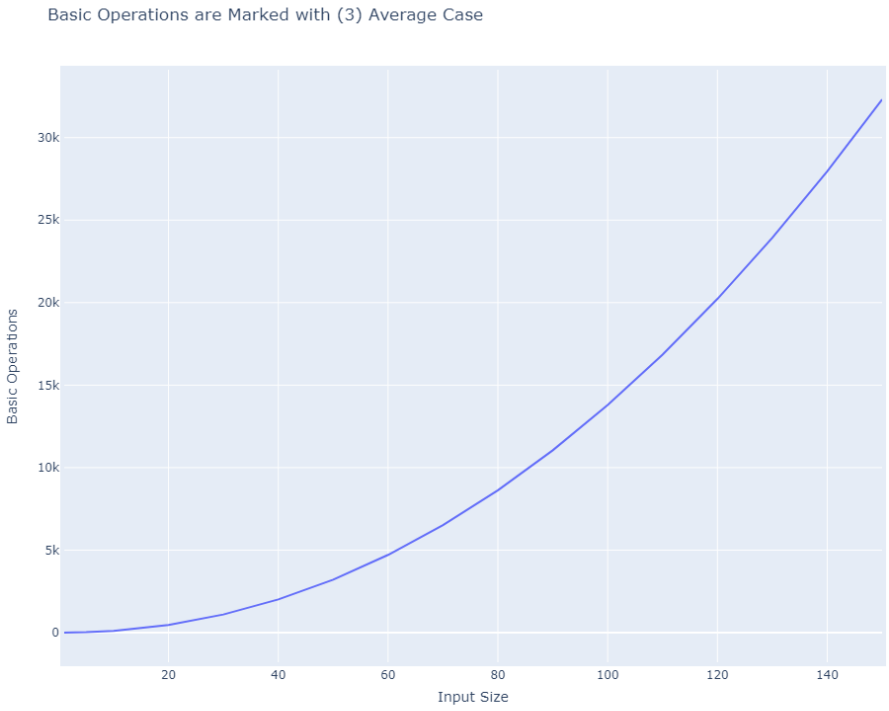
Graph of the theoretical analysis when basic operation is the operation marked as (1)



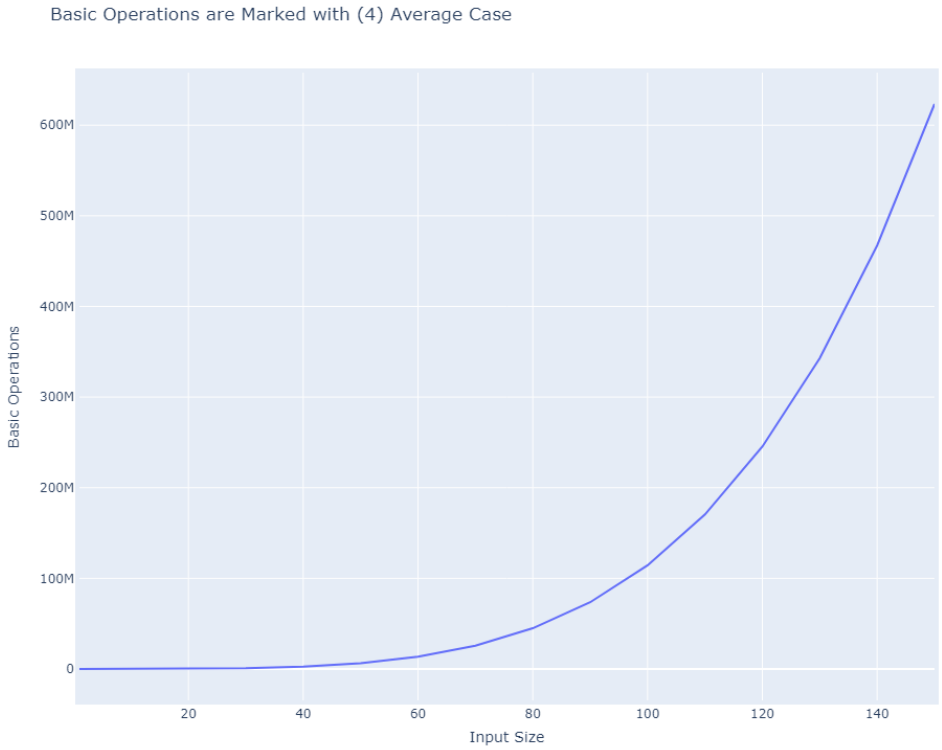
Graph of the theoretical analysis when basic operation is the operation marked as (2)



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

By looking at the graphs above, and comparing them with the best-case and worst-case analysis graphs, we can say that average-case analysis is very similar to worst-case analysis. Since it averages the execution of each block of the algorithm, it doesn't mislead us as was the case with the best-case analysis. In addition, as we move from basic operation (1) to (4), our analysis's graph approaches to the real execution graph of the algorithm. As a result, we can say that choosing (4) as basic operation was a correct choice for the algorithm.