

# 2022-2023 SPRING

**CS201 – HOMEWORK 2**

Section: 1

Full Name: Tolga Han Arslan

Student ID: 22003061

**Data for algorithm 1: O(N3)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Size (N)** | **Time (ms)** | **Input Size (N)** | **Time (ms)** |
| 1 | 0.00001 | 1500 | 1999.36920 |
| 10 | 0.00071 | 1750 | 3171.84170 |
| 50 | 0.08060 | 2000 | 4731.73410 |
| 120 | 1.03030 | 2500 | 9227.48090 |
| 150 | 1.99360 | 3000 | 16041.09100 |
| 200 | 5.02340 | 4000 | 37775.36290 |
| 500 | 74.83630 | 5000 | 74125.29030 |
| 750 | 251.71280 | 6000 | 127531.42170 |
| 1000 | 595.88170 | 7500 | 249637.75130 |
| 1250 | 1159.32690 | 10000 | 589872.53170 |

**Data plot:**

*Figure 1: Theoretical and Experimental growth rates of algorithm 1*

* Theoretical divided by 1800000

**Data for algorithm 2: O(N2)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Size (N)** | **Time (ms)** | **Input Size (N)** | **Time (ms)** |
| 1 | 0.00001 | 5000 | 42.80580 |
| 50 | 0.00499 | 7500 | 97.32160 |
| 300 | 0.15952 | 10000 | 172.87590 |
| 600 | 0.97380 | 12500 | 271.12330 |
| 750 | 1.02970 | 15000 | 390.31650 |
| 1000 | 2.03160 | 20000 | 694.15990 |
| 1500 | 3.99170 | 25000 | 1084.51200 |
| 2000 | 6.98540 | 50000 | 4343.72640 |
| 3000 | 15.03610 | 75000 | 9775.99000 |
| 4000 | 27.72390 | 100000 | 17406.51510 |

**Data plot:**

*Figure 2: Theoretical and Experimental growth rates of algorithm 2*

* Theoretical divided by 500000

**Data for algorithm 3: O(N logN)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Size (N)** | **Time (ms)** | **Input Size (N)** | **Time (ms)** |
| 1 | 0.00001 | 1000000 | 131.90160 |
| 100 | 0.00378 | 1250000 | 165.74760 |
| 1000 | 0.06084 | 1500000 | 201.49920 |
| 10000 | 0.99770 | 2000000 | 270.27730 |
| 50000 | 5.95180 | 5000000 | 696.43890 |
| 100000 | 12.08200 | 7500000 | 1058.51500 |
| 150000 | 18.80360 | 10000000 | 1426.38580 |
| 200000 | 24.61560 | 20000000 | 2927.88520 |
| 500000 | 64.52990 | 50000000 | 7546.83940 |
| 750000 | 98.57570 | 100000000 | 15522.64550 |

**Data plot:**

*Figure 3: Theoretical and Experimental growth rates of algorithm 3*

* Theoretical divided by 50000

**Data for algorithm 4: O(N)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Input Size (N)** | **Time (ms)** | **Input Size (N)** | **Time (ms)** |
| 1 | 0.00001 | 2500000 | 7.89080 |
| 100 | 0.00034 | 5000000 | 17.01920 |
| 1000 | 0.00342 | 7500000 | 25.80930 |
| 10000 | 0.03664 | 10000000 | 33.90940 |
| 50000 | 0.16510 | 15000000 | 49.86690 |
| 100000 | 0.33536 | 25000000 | 83.12290 |
| 250000 | 0.82637 | 50000000 | 165.91660 |
| 500000 | 1.68324 | 100000000 | 330.24920 |
| 1000000 | 2.99200 | 500000000 | 1654.93110 |
| 1500000 | 4.98660 | 1000000000 | 3656.65850 |

**Data plot:**

*Figure 4: Theoretical and Experimental growth rates of algorithm 4*

* Theoretical divided by 270000

**All algorithms comparison: (Experimental values)**

*Figure 5: Comparison between all four algorithms for input sizes less than 1000*

**Computer Specifications:**

Processor: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz

RAM: 16,0 GB (available: 15,8 GB)

System: 64-bit operating system, x64 based-processor

Operating System: Windows 10 Pro

SSD: 512 GB

**Discussion:**

Four algorithms for the maximum subsequence sum problem are tested on the above computer. Their execution times in miliseconds are recorded with using different input values (size of the array). For small inputs, execution time cannot be measured precisely since the code code segment in the document rounds execution times of much less than 1 milliseconds to 0 milliseconds. Hence, to measure the execution times for small inputs, the algorithm is called a constant amount of time with using a loop and the is divided by the constant in order to get more precise results. Also, in order to plot theoretical growth rates, time complexity functions are divided by constants to get a more clear image between theoretical and experimental curves.

For algorithm 1, theoretical analysis shows that its time complexity is O(N3). In the data table , input size of 1 to 10000 is used since the growth rate of this algorithm makes it immeasurable to determine higher input sizes. Then, data values from the table and expected growth rate are plotted simultanously in figure 1. Theoretical growth rate is plotted as a cubic function and experimental growth rate in the plot shows that if input size is getting larger, the resulting curve converges to the theoretical one.

For algorithm 2, theoretical analysis shows that its time complexity is O(N2). In the data table, input size of 1 to 100000 is used since this algorithm grows slower than algorithm 1 and allows to measure higher input sizes. Hence, data values from the table and expected growth rate are plotted simultanously in figure 2. Theoretical growth rate is plotted as a quadratic function this time and experimental results shows that for larger input sizes, the resulting curve converges to the theoretical one too.

Algorithm 3 is a involves a recursive approach to the maximum subsequence sum problem and theoretical analysis shows that its time complexity is O(N logN). This growth rate allows even much higher input sizes since it is almost linear because of the fact that logarithm function grows much slowly than other functions. So, input size of 1 to 100000000 is used fort his algorithm in the data table and similarly, experimental and theoretical growth rates are plotted in figure 3. This plot shows that the growth rate of this algorithm is almost a linear one and experimental growth is converging to theoretical one in higher input sizes too. Lastly, algorithm 4 provides a linear solution for the maximum subsequence sum problem and hence its time complexity is O(N). Same as algorithm 3, this growth rate allows to measure much higher input sizes than cubic and quadratic growth rates and therefore, input size of 1 to 1000000000 is used. Theoretical growth rate is plotted as a linear function and the plot shows that for high input sizes, experimental curve converges to theoretical curve in figure 4.

Finally, experimental results of all four algorithms are compared with input sizes of less than 1000 in figure 5. Low input sizes are used in this plot to get a clear image of the efficiency of algorithms. It shows that the algorithm 4 is the most efficient one and algorithm 1 is the least efficient one in terms of execution times. Recursive solution (algorithm 3) is almost like linear solution (algorithm 4) and can be a good solution to the maximum subsequence sum problem if there is no linear solution like algorithm 4.

In conclusion the time complexity and growth rate of the algorithms can be arranged as:

O(N3) > O(N2) > O(N logN) > O(N)

And execution time efficiency of algorithms:

Algorithm 4 > Algorithm 3 > Algorithm 2 > Algorithm 1