1. **Question 1: Comparing Algorithms. Problem: Find the THIRD largest in an array.**
2. **Write the pseudo code. (Must follow the notations and conventions used in today’s Lecture)**

***Algorithm 1:***

Algorithm findThirdLargest(A, n) # operations

firstLargest 🡨 0 1

for i 🡨 0 to n – 1 do 1 + n

if A[i] > firstLargest then 2(n – 1)

firstLargest 🡨 A[i] 2(n - 1)

{ increment counter i } 2(n – 1)

secondLargest 🡨 0 1

for i 🡨 0 to n – 1 do 1 + n

if A[i] > secondLargest and A[i] <= firstLargest then 4(n – 1)

secondLargest 🡨 A[i] 2(n – 1)  
 { increment counter i } 2(n – 1)

thirdLargest 🡨 0 1

for i 🡨 0 to n – 1 do 1 + n

if A[i] > thirdLargest and A[i] <= secondLargest then 4(n – 1)

thirdLargest 🡨 A[i] 2(n - 1)

{ increment counter i } 2(n - 1)  
  
return thirdLargest 1

**Total: 27n – 15**

Complexity time: O(n)

***Algorithm 2:***

Algorithm findThirdLargest(A, n) # operations

firstLargest 🡨 0 1

secondLargest 🡨 0 1

thirdLargest 🡨 0 1

for i 🡨 0 to n – 1 do 1 + n

if A[i] >= firstLargest then 2(n – 1)

thirdLargest 🡨 secondLargest n - 1

secondLargest 🡨 firstLargest n – 1

thirdLargest 🡨 A[i] 2(n – 1)

else if A[i] >= secondLargest then 2(n – 1)

thirdLargest 🡨 secondLargest n - 1

secondLargest 🡨 A[i] 2(n – 1)

else if A[i] >= thirdLargest then 2(n – 1)

thirdLargest 🡨 A[i] 2(n – 1)

{ increment counter i } 2(n – 1)

return thirdLargest; 1

**Total: 18n - 11**

Complexity time: O(n)

***Algorithm 3:***

Algorithm findThirdLargest(A, n) # operations

for i 🡨 0 to n – 1 do 1 + n

for j = i + 1 to n – 1 do 2 + n - 1

if A[i] > A[j] 3(n – 2)(n – 1)

temp 🡨 A[j] 2(n – 2)(n – 1)

A[j] 🡨 A[i] 3(n – 2)(n – 1)

A[i] 🡨 temp 2(n – 2)(n – 1)

{ increment counter j } 2(n – 2)(n – 1)

{ increment counter i } 2(n – 1)

return A[n - 3] 3

**Total: 12n^2 + 6n + (constant)**

Complexity time: O(n^2)

Algorithm 2 -> Algorithm 1 -> Algorithm 3

1. **Determine the worst-case time complexity by counting as in Slide 15 Lesson 2**

Algorithm 1: Worst-case time complexity **27n – 15**

Algorithm 2: Worst-case time complexity **18n - 11**

Algorithm 3: Worst-case time complexity **12n^2 + 6n + c (constant)**

1. **Perform an empirical time comparison by implementing using Java, similar to what you did in W1D1.**

See the source code in package: **lesson2.lab.question1**

1. **Question 2:**

|  |  |
| --- | --- |
| 1, 10 | O(1) |
| log(log(n)), log(n), ln(n) | 0(log(n)) |
| n | O(n) |
| n^(1/2), n^(1/3) | O(n^k) (0 < k < 1) |
| n^(1/2)log(n), n^(1/3)log(n) | O(n^k\*log(n)) (0 < k < 1) |
| nlog(n) | O(nlog(n)) |
| n^2, n^k, n^n | O(n^k) (k > 1) |
| 2^n, 3^n | O(2^n) |
| n! | O(n!) |