#### **Question 1. Comparing Algorithms**

(a) In this lab, for all three algorithms you will

(b) write the pseudo code. (Must follow the notations and conventions used in today’s Lecture)

(c) determine the worst case time complexity.

Problem statement

Find the largest distance between any two even integers in an integer array.

**Algorithm 1.**

Create a new array consisting of even numbers only. Then use nested loops to solve the problem using

the newly created array of even numbers only.

**Algorithm** findLargestDistance(A, n)

**Input** Array A of n integers

**Output** largestDistance between 2 even elements of A

numsEven <- new Array

for i <- 0 to n - 1 do

if(A[i]%2 == 0)

numsEven[x] <- A[i]

largestDistance <- 0

for i <- 0 to n - 2 do

for j <- 1 to n - 1 do

distance <- numsEven[i] - numsEvent[j]

if(numsEven[j] > numsEvent[i])

distance <- numsEven[j] - numsEvent[i]

if(largestDistance < distance)  
 largestDistance <- distance

return largestDistance

**Time complexity:** O(n) + O(n2) -> O(n2)

**Algorithm 2.**

Use a nested loop to solve the problem without creating an extra array.

**Algorithm** findLargestDistance(A, n)

**Input** Array A of n integers

**Output** largestDistance between 2 even elements of A  
  
 largestDistance <- 0

for i <- 0 to n - 2 do  
 if(A[i]%2 != 0)

continue

for j <- 1 to n - 1 do

if(A[j]%2 != 0)

continue

distance <- A[i] - A[j]

if(A[j] > A[i])

distance <- A[j] - A[i]

if(largestDistance < distance)  
 largestDistance <- distance

return largestDistance

**Time complexity:** O(n2)

**Algorithm 3.**

Use one loop. Find max and min of even integers. Compute max – min.

**Algorithm** findLargestDistance(A, n)

**Input** Array A of n integers

**Output** largestDistance between 2 even elements of A

max <- Integer.MIN\_VALUE

min <- Integer.MAX\_VALUE

for i <- 0 to n - 1 do  
 if(A[i]%2 == 0)

if(max < A[i])  
 max <- A[i]

if(min > A[i])  
 min <- A[i]

largestDistance <- max - min

return largestDistance  
  
  
**Time complexity:** O(n)

#### **Question 2.**

Consider the following functions to determine the relationships that exist among the complexity classes they belong: 10, 1, n3 , n1/3 , log(log(n)), n2, n1/2, logn , lognn , nk (k > 3) , n1/k (k > 3), nlogn, ln(n), 2n , 3n , nn , n1/2 logn, n1/3 logn, n!.

|  |  |
| --- | --- |
| 10, 1 | 𝞡(1) |
| log(log(n)) | 𝞡(log(log(n))) |
| log(n), ln(n) | 𝞡(log(n)) |
| n1/2, n1/3, n1/k (k > 3) | 𝞡(n1/k) (0 < k < 1) |
| n1/2 logn, n1/3 logn | 𝞡(n1/klogn) (0 < k < 1) |
| nlogn, lognn | 𝞡(nlog(n)) |
| n2 | 𝞡(n2) |
| n3 | 𝞡(n3) |
| nk (k > 3), nn | 𝞡(nk) |
| 2n | 𝞡(2n) |
| 3n | 𝞡(3n) |
| n! | 𝞡(n!) |