**Lab W1D2**

**Algorithm 1**

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

avoids <= []

fMax <= -1

fIndex <= -1

for i <= 0 to {get size} do

if data[i] > fMax

fMax <= data[i]

fIndex <= i

avoids <= fIndex

sMax <= -1

sIndex <= -1

for i <= 0 to {get size} do

if i in avoids

continue

if data[i] > sMax

sMax <= data[i]

sIndex <= i

avoids <= sIndex

tMax <= -1

for i <= 0 to {get size} do

if i in avoids

continue

if data[i] > tMax

tMax <= data[i]

print fMax sMax tMax

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

1 + 1 + 1 +

(2 + n) + (2n) + (2n) + n + n +

1 +

1 + 1 +

(2 + n) + nk + (2n) + (2n) + n + n +

1 +

1 +

(2 + n) + k + (2n) + (2n) + n +

1

Total: 20n + kn = n^2

**Algorithm 2**

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

max <= -1

preMax <= -1

prePreMax <= -1

for i <= 0 to n do

if n[i] > max

prePreMax <= preMax

preMax <= max

max <= n[i]

else if n[i] > preMax

prePreMax <= preMax

preMax <= n[i]

else if n[i] > prePreMax

prePreMax <= n[i]

print max preMax prePreMax

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

1 + 1 + 1 +

(2 + n) + n + (n + n + n) + n + (n + n) + n + n +

1

Total: 10n

A graph of a line

Description automatically generated

**Question 2**

1. O(1): 10, 1, 3

2. O(log(log n)): log(log n)

3. O(log n): log n

4. O(n^(1/k)) for k > 3: n^(1/k) for k > 3

5. O(n^(1/3)): n^(1/3)

6. O(n^(1/2)): n^(1/2)

7. O(n^(1/3) \* log n): n^(1/3) \* log n

8. O(n^(1/2) \* log n): n^(1/2) \* log n

9. O(n): n

10. O(n^2): n^2

11. O(n^k) for k > 3: n^k for k > 3

12. O(n \* log n): n \* log n

13. O(ln n): ln n

14. O(2^n): 2^n

15. O(3^n): 3^n

16. O(n!): n!

17. O(n^n): n^n