**W1D3**

1. **Question 1**

**Algorithm** findMax(A, start, end)

**Input** array **A** of **n** intergers

**Output** maximum element of **A**

**If** start = end **then**

**return** A[start]

**If** start+1 = end **then**

**return** max(A[start], A[end])

  mid <- (start + end)/2

  a <- findMax(A, start, mid)

  b <- findMax(A, mid+1, end)

**return** max(a,b)

**Using master theorem**

T(1) = 1

T(n) = 2T(n/2) + c

Where a = 2, b = 2, k = 0

It is a>b^k

T(n)    = θ(n^log2)    =θ(n^1)         =θ(n)

1. **Question 2** Order them based on their complexity

2^n , 2^(2n) = 4^n, 2^(n + 1), 2^( 2^n ) = 4^n

So 2^n, 2^2n, 2^( 2^n ), 2^(n + 1)

1. **Question 3**

**Answer:**

O(1) : Accessing a value with key form HashMap

O(log n): Binary Search

O(n): Linear Search

O(n log n): Merge Sort

O(n^2): Bubble Sort

O(n^3): Matrix Multiplication

O(2^n): Finding the nth term of Fibonacci using recursive

1. **Question 4**

**F(n) = F(n – 1) + F(n - 2), so b is not a constant, so we can’t apply Master formula**

1. **Question 5**

T(n) = 2T(n/2) + c

Let n = 64= 2^6

T(64) = 2T(32) + c

2T(32) = 2^2T(16) + 2c

2^2T(16) = 2^3T(8) + 2^2c

2^3T(8) = 2^4T(4) + 2^3c

2^4T(4) = 2^5T(2) +2^4 c

2^5T(2)= 2^6T(1) + 2^5c

So T(64) = 2^6 + 2^5c

               = 2^6 + c(1 + 2 + … + 2^5)

               = 2^6 + 2^ 6– 1

T(n)        = n + c(n –1) is O(n)