**Assignment Two**

1. **Pseudocode for Algorithm 1**

**Algorithm** thirdLargest(A, n)

**Input** array A of n integers

**Output** the third largest element of A

# operations

largest Integer.MIN\_VALUE 2

firstIndex 0 1

secondIndex 0 1

**for** i 0 **to** n-1 **do**  n + 1

**if** A[i] > largest **then**  2(n – 1)

largest A[i] 2(n – 1)

firstIndex i n – 1

{increment counter i} 2(n – 1)

largest Integer.MIN\_VALUE 2

**for** i 0 **to** n-1 **do** n + 1

**if** A[i] > largest **AND** i != firstIndex **then** 4(n – 1)

largest A[i] 2(n – 1)

secondIndex i n – 1

{increment counter i} 2(n – 1)

largest Integer.MIN\_VALUE 2

**for** i 0 **to** n-1 **do** n + 1

**if** A[i] > largest **AND** i != firstIndex **AND** i != secondIndex **then** 6(n – 1)

largest A[i] 2(n – 1)

{increment counter i} 2(n – 1)

**return** largest 1

**Pseudocode for Algorithm 2**

**Algorithm** thirdLargest(A, n)

**Input** array A of n integers

**Output** the third largest element of A

# operations

max A[0] 2

preMax A[0] 2

prePreMax A[0] 2

**for** i 0 **to** n-1 **do** n + 1

**if** A[i] > max **then** 2(n – 1)

prePreMax preMax n - 1

preMax max n - 1

max A[i] 2( n – 1)

**else** **if** A[i] > preMax **then** 2(n – 1)

prePreMax preMax n - 1

preMax A[i] 2(n – 1)

**else** **if** A[i] > prePreMax **then** 2(n – 1)

prePreMax A[i] 2(n – 1)

**else**

skip to next iteration 2(n – 1)

{increment counter i} 2(n – 1)

**return** prePreMax 1

1. **Algorithm One** : The worst case time complexity is where = time taken by the fastest primitive operation and

= time taken by the slowest primitive operation

The worst case running time of Algorithm One is .

**Algorithm Two**: The worst case time complexity is

where = time taken by the fastest primitive operation and

= time taken by the slowest primitive operation

The worst case running time of Algorithm Two is .

d. Comparison chart for Algorithm One and Algorithm Two

**Observation and Conclusion**

It can be inferred from the chart drawn above that both Algorithm One and Algorithm Two have worst case running time that is a function of n, for given n inputs. The only difference between these two algorithms is the constant that multiplies each of these functions. The difference caused by the multiplication of the constant can be seen in the chart as Algorithm One is slightly higher than Algorithm Two i.e 30 > 20. In conclusion these two algorithms have the same time complexity of

**Question 2**

|  |  |
| --- | --- |
| 10, 1 |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | ( |
|  |  |