Course: Algorithm
Prof. Prem Nair
Student: Binh Van Tran
ID: 986648
Homework: Lab 3

1. **Question 1 –** Write an algorithm such that the best running time is equal to the worst case running time

$$
\begin{aligned}
&\boldsymbol{findMaximum}(A, n)\\
&\qquad \boldsymbol{Input}\ array\ A \text{ of } n \text{ integers}\\
&\qquad \boldsymbol{Output}\ largest\ value\ of\ A\\
&\qquad max\ \leftarrow INT_{MINVALUE}\\
&\qquad \boldsymbol{for}\ i\ \leftarrow 0\ \boldsymbol{to}\ n-1\ \boldsymbol{do}\\
&\qquad\qquad \boldsymbol{if}\ A[i] > max\ \boldsymbol{then}\\
&\qquad\qquad\qquad \text{max} = A[i]\\
&\qquad \boldsymbol{return}\ max
\end{aligned}
$$

The *for…loop* from index *zero* to *n – 1* is always be executed no matter where the maximum value is. So, the time complexity of this algorithm is always *O(n).*

2. **Question 2 –** Order the following functions based on their complexity
2^n, 2^{2n}, 2^{n+1}, 2^{2^n}
2^n is in $O(2^n)$
2^{2n} is in $O(2^n.2^n)$
2^{n+1} is in $O(2.2^n)$
2^{2^n} is in $O(2^{2^n})$
The order should be: 2^n, 2^{n+1}, 2^{2n} and 2^{2^n}

3. **Question 3** – Mention one algorithm you know for each of the time complexities listed
O(1): Find maximum or minimum of a sorted array; Get value from an array by index
O(log*n*): Search a value in sorted array using binary search
O(n): Find maximum or minimum value of an unordered array; Search a value in array
O(nlog*n*): Quick sort, Heap sort, merge sort
$O(n^2)$: Insertion Sort, Selection Sort
$O(n^3)$: Any algorithm with 3 deep nested *for…loop*
$O(2^n)$: Fibonacci, finding subset

4. **Question 4 –** Apply Master Theorem and determine the time complexity of
   - **fib(n)**
     We have T(1) = d, T(n) = T(n – 1) + T(n – 2).

Because this algorithm is not a Divide-And-Conquer one, so we cannot apply the master theorem to calculate the time complexity of T(n)

- **binarySearch**

  We have:

  T(1) = d,

  $T(n) = T(\frac{n}{2}) + c$

  So, a = 1, b = 2, k = 0, c = 1

  $b^k = 2^0 = 1$

  Hence $a = b^k$ => T(n) is in $O(n^k \log n) = O(n^0 \log n) = O(\log n)$

5. **Practice Master theorem**

- <u>Case 1</u>: $a < b^k$

  $T(n) = 3T(n/2) + n^2$

  a = 3, b = 2, k = 2

  $b^k = 4 > a$

  T(n) is in $O(n^2)$

- <u>Case 2</u>: $a = b^k$

  Merge sort:

  T(1) = d,

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

  a = 2, b = 2, k = 1; $b^k = 2^1 = 2 = a$

  T(n) is in $O(n^k \log n) = O(n \log n)$

- <u>Case 3</u>: $a > b^k$

  T(n) = 4T(n/2) + (½)n

  a = 4, b = 2, k = 1, $b^k = 2 < a$

  T(n) is in $O(n^{\log 4})$