Student Name : Rakyan Satrya Adhikara

Student ID : 219548135

Task 2.1P (Case complexity (Pass))

Question 1:

1. Best, worst, average case (number of operations)
   1. Case I
      * Best : 2
      * Worst :
      * Average :
   2. Case II
      * Best :
      * Worst :
      * Average :
   3. Case III
      * Best :
      * Worst :
      * Average :
   4. Case IV
      * Best :
      * Worst :
      * Average :
   5. Case V
      * Best :
      * Worst :
      * Average :
   6. Case VI
      * Best :
      * Worst :
      * Average :
2. Best, worst and average case (big-):
   1. Case I
      * Best :
      * Worst :
      * Average :
   2. Case II
      * Best :
      * Worst :
      * Average :
   3. Case III
      * Best :
      * Worst :
      * Average :
   4. Case IV
      * Best :
      * Worst :
      * Average :
   5. Case V
      * Best :
      * Worst :
      * Average :
   6. Case VI
      * Best :
      * Worst :
      * Average :
3. Overall performance in Big‐O notation:
   1. Case I :
   2. Case II :
   3. Case III :
   4. Case IV :
   5. Case V :
   6. Case VI :
4. Overall performance  in Big‐ Ω notation:
   1. Case I :
   2. Case II :
   3. Case III :
   4. Case IV :
   5. Case V :
   6. Case VI :
5. Overall performance in Big‐Θ notation:
   1. Case I :
   2. Case II :
   3. Case III : Not possible
   4. Case IV : Not possible
   5. Case V :
   6. Case VI :
6. Asymptotic Notations:
   1. Case I :
   2. Case II :
   3. Case III :
   4. Case IV :
   5. Case V :
   6. Case VI :

Question 2:

The reasons why the most commonly use asymptotic notation use is frequently Big-O:

* It is common to use Big-O simply because the Big-O notation is the upper bound (worst case). Compared to the Big- is the tightest bound, this information is far more valuable than the Big- (lower bound), as it gives an understanding of how poorly an algorithm will perform with respect to its growth rate. Thus, Big-O will be much more effective to analyze the performance of the Case.
* There are many cases which the big- is unknown

Question 3:

Assume that you mean always in the sense of changeable values of n, that is, a sorting algorithm in which the array that is fed in is sized in various kinds. Then there will never be a **that not** always take longer to run than a . For instance:

The statement above is true. On the other hand, at small values of n, the algorithm will clearly run faster, such as for arrays of a small size.

However, if you mean that this comparison is to be made as n approaches infinity then yes, a will always take longer to run than a .

Question 4:

* True
  + The Big-O notation for
* False
  + The Big-O notation for is ()
* False
  + The Big-O notation for
  + The Big- Ω notation for it is
  + Hence, it must be ()
* True
  + The Big- notation for is