

1. Conduct a small research on the Introspective Sort, which is a built-in sorting algorithm of the `Array.Sort` method that you referred to in task 2.1P. Is it a unique algorithmic solution or a combination of several sorting algorithms? Does this method result in an unstable sort; that is, if two elements are equal, their order might not be preserved? Or does it perform a stable sort, which preserves the order of elements that are equal? What is the time complexity of this sorting algorithm?

`Array.Sort()` is a combination of several sorting algorithms. I think this sorting algorithms is not unstable sort. `Array.Sort()`'s time complexity is $O(n \log n)$.

2. Task 1.1P asked you to develop / provide you with a number of the `Vector<T>` class's methods (and properties), such as *Count*, *Capacity*, *Add*, *IndexOf*, *Insert*, *Clear*, *Contains*, *Remove*, and *RemoveAt*. What is the algorithmic complexity of each of these operations? Does your implementation match the complexity of the corresponding operations offered by Microsoft .Net Framework for its `List<T>` collection?

- a) Count: Algorithmic complexity: best
- b) Capacity: Algorithmic complexity: best
- c) Add: Algorithmic complexity: best
- d) IndexOf: Algorithmic complexity: best
- e) Insert: Algorithmic complexity: worst
- f) Clear: Algorithmic complexity: best
- g) Contains: Algorithmic complexity: best
- h) Remove: Algorithmic complexity: worst
- i) RemoveAt: Algorithmic complexity: worst

I am not implementation match the complexity of the corresponding operations offered by Microsoft .Net Framework for its `List<T>` collection

3. Is it true that $\Theta(n^2)$ algorithm always takes longer to run than $\Theta(\log n)$ an log algorithm? Explain your answer.

No, this is because the $\Theta(n^2)$ algorithm first time is slow to fast. However, the $\Theta(\log n)$ algorithm is fast to slow. Therefore, in the first times, the $\Theta(n^2)$ will slower than $\Theta(\log n)$ in the first time.

4. Answer whether the following statements are right or wrong and explain your answers.

- a) Wrong The n can be 0, then $f(x) = g(x)$
- b) Wrong The n can be 10, then $f(x) = g(x)$
- c) True f grows at least as fast as g between 0 to 10
- d) True f grows slower than g