CPSC 319 Tutorial 05
A Review of Sorting Algorithms

Longsheng Zhou

Department of Computer Science
University of Calgary

January 31, 2015

A Tour

Tuesday, Feb.03

1. A Review of Sorting Algorithms
   - Bubble Sort
   - Insertion Sort
   - Merge Sort
   - Quick Sort
2. Some Suggestions for Assignment

Thursday, Feb.05

1. Assignment 1 Working Period

- No pseudocode, No java implementations, No complexity analysis. What are the topics today?
  - **High intuition of these sorting mechanisms.**
  - **Describe them as brief as you can.** [Exercises!]
  - **Pros and Cons**

- **Bubble Sort**: Starting from the beginning of the list, compare every adjacent pair, swap their position if they are not in the right order (the latter one is smaller than the former one).
- Too abstract? Let's dance!

  `https://www.youtube.com/watch?v=lyZQPjUT5B4`

- **Pros:**
  - Simplicity and ease of implementation.
  - Auxiliary Space used is $O(1)$.
- **Cons:**
  - Very inefficient. General complexity is $O(n^2)$. Best case complexity is $O(n)$

- **Insertion Sort**: To sort unordered list of elements, we remove its entries one at a time and then insert each of them into a sorted part (initially empty):.
- Let's dance, again!

  `https://www.youtube.com/watch?v=ROalU379l3U`

- **Pros:**
  - Online.
  - Auxiliary Space used is $O(1)$.
- **Cons:**
  - Inefficient. General complexity is $O(n^2)$. Best case complexity is $O(n)$

- **Merge Sort**: Divide-and-conquer-based sorting involves the following three steps: Divide the array into two (or more) subarrays, then sort each subarray (Conquer) and merge them into one.
- Let's dance, again!

  `https://www.youtube.com/watch?v=XaqR3G_NVoo`

- **Pros:**
  - Faster, $O(nlgn)$.
  - The best choice for sorting a *linked list* (a data structure which will be discussed later).
- **Cons:**
  - At least twice the memory requirements of the other sorts because it is recursive. It requires about a $O(n)$ auxiliary space for its working.

- **Quick Sort**: Divide and conquer-based algorithm which divides a large array into two smaller sub-arrays: the low elements and the high elements. Quicksort can then recursively sort the sub-arrays.
- Let's dance, again!

  `https://www.youtube.com/watch?v=ywWBy6J5gz8`

- **Pros:**
  - In almost every case, time cost is $O(n)$. It is claimed to be faster than mergesort. It is named as top 10 algorithms of the 20th Century.

    `https://www.siam.org/pdf/news/637.pdf`
  - It can be made to use $O(lgn)$ space much better than mergesort..
- **Cons:**
  - Its worst case has a time complexity of $O(n^2)$ which can prove very fatal for large data sets.

Suggestions.

1. Ensure your java program could be compiled successfully.
2. Write some brief comments to make your program understandable. What's *Java Documentation Comments*?
   http://www.tutorialspoint.com/java/java_documentation.htm
3. Do some test yourself before submission. Test What?
   - Command Line Arguments
   - Creation of Input Integer Array
   - Algorithms Implementation
   - Time Piece of Code
   - File Output
   - Code Understandable (documentation, format, etc)

Thank you!

Author: Longsheng Zhou
Address: ICT 609e
Department of Computer Science
University of Calgary
Email: lozhou@ucalgary.ca