CS 101 Fall 2020 - Quiz 4-A

10/13/2020 - 20 Minutes **Name:** **ID number:**

Problem 1(4×2pts): **True or False:** For each statement, write "T" if this statement is correct; write "F" otherwise. Please **write your answers in the box below**.

Statement (1)	Statement (2)	Statement (3)	Statement (4)
F	F	T	T

(1) Merge sort requires $O(1)$ extra space complexity.

(2) In quicksort (sort in ascending order), if we randomly select the pivot, after the first partition operation, the smallest element of the array can be anywhere.

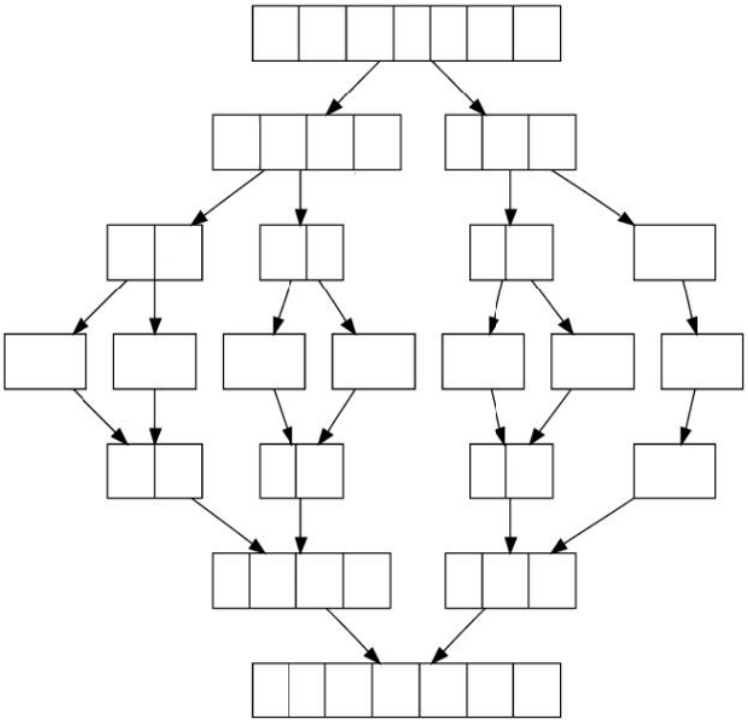
(3) The average and the worst time complexity of mergesort are both $O(nlog(n))$.

(4) By applying the partition step of quicksort on an **unsorted** array repeatedly, we can get the $k-th$ biggest number of that array with an **average** time complexity of $O(n)$. (k is an arbitrary number)

Problem 2(5pts):

Consider this array: $4, 5, 2, 6, 1, 3, 7$.

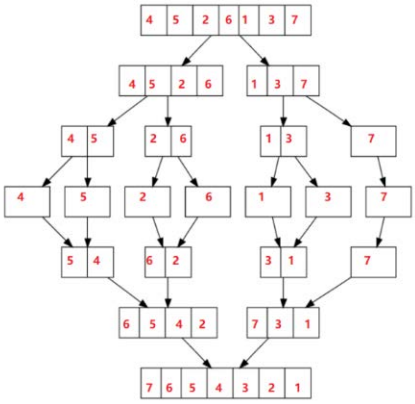
(1)(3.5pts) Use **mergesort** to sort this array in ascending order. Show your process in the following figure.



(2)(1.5pts) How many inversions are there in the array? _____

CS 101 Fall 2020 - Quiz 4-A
10/13/2020 - 20 Minutes **Name:** **ID number:**

Solution: 1. As follows. 2. 12



Problem 3(3×1pts):
Tom wants to sort his favorite colors in ascending order using quicksort. The original array is:

$$red, cyan, yellow, gray, green, black, blue, white$$

After the first partitioning step, it becomes: ("red" is chosen as pivot)

$$white, cyan, yellow, gray, red, black, green, blue$$

Known that **NO** elements are equal, we can infer that: (Fill the blanks with ">", "<", or "?" if given information is insufficient to judge)

(a) red _____ blue (b) yellow _____ gray (c) green _____ cyan
Solution: 1. < 2. ? 3. >

Problem 4(4pts): Prove that: When performing quicksort, if the array is **equally** divided into two parts, the time complexity for quicksort would be $O(nlog(n))$.
Solution:

(1) $T(n) = 2T(n/2) + O(n)$
(2) $= 2(2T(n/4) + O(n/2)) + O(n)$
(3) $= 4T(n/4) + 2O(n/2) + O(n)$
(4) $= 4T(n/4) + 2O(n)$
(5) $= ...$
(6) $= O(nlog(n))$