Algorithms Midterm Exam (Spring 2023)

Total：110%

Name: ______________________

Student ID #: ________________

Question	**Score**
1 (35%)	
2 (15%)	
3 (15%)	
4 (10%)	
5 (15%)	
6 (20%)	
Total	

1. **[Simple questions: 35%]** Complete the following questions with simple answers.
   (a) (5%) Suggest a best sorting algorithm when *in-place* as well as the *best* and the *average*-case performance are the top concerns.

   (b) (5%) Compare the two sorting algorithms merge sort and heapsort where both perform the best in the worst-case scenarios by giving pros and cons of the two algorithms.

   (c) (5%) Is $\Omega(f(n)) + O(f(n)) = \Theta(f(n))$ correct? Explain your answer.

(d) (5%) Is $O(n^2) + \Omega(n^3 \ln n) = \Omega(n^3 \ln n)$ correct? Explain your answer.

(e) (5%) Describe how the quadratic probing can be better than linear probing.

(f) (5%) A sequence has been stored into a hash table following the order $k_1, k_2, \dots, k_n$, now searching for the first element k_1 or the last element k_n takes the fastest time. Discuss the answer when the chaining strategy is used.

(g) (5%) Following the previous question, but for open addressing strategy.

2. **[Sorting: 15%]** Answer two questions regarding to the *stable* property of sorting.
   (a) (8%) Show how to detect a sorting algorithm is stable or not. That is, given an unknown sorting algorithm, can you help to write a short script to detect whether the output of the sorting algorithm produces stable outcomes.

(b) (7%) Below is an implementation of the heapsort algorithm. Can you identify the line(s) where executing the line(s) can lead to non-stable sorting result, and why?

MAX-HEAPIFY(*A*, *i*)

```
l → LEFT(i)
r → RIGHT(i)
if   l ≤ heap-size[A] and A[l] > A[i]
then   largest   ← l
else   largest   ← i
if   r ≤ heap-size[A] and A[r] > A[largest]
then   largest ← r
if   largest ≠ i
   then   SWAP(A[i], A[largest])
          MAX-HEAPIFY(A, largest)
```

BUIDE-MAX-HEAP(*A*)

```
heap-size[A] ← length[A]
for i ← ⌊length[A]/2⌋ downto 1
    do MAX-HEAPIFY(A, i)
```

HEAPSORT(*A*)

```
BUIDE-MAX-HEAP(A)
for i ← length[A] downto 2
    do SWAP(A[1], A[i])
           heap-size[A] ← heap-size[A] – 1
           MAX-HEAPIFY(A,1)
```

3. **[Sorting: 15%]** Finish the following two tasks in $O(n)$ time.

(a) (7%) Select the five smallest elements from an unknown sequence.

(b) (8%) Sort n elements where the elements are in the range of 0 to $n^3 - 1$.

4. **[Quicksort: 10%]** The following is the quicksort implementation that we discussed in class.

```
QUICKSORT(A, p, r)
1   if  p < r
2       then q ← PARTITION(A, p, r)
3            QUICKSORT(A, p, q-1)
4            QUICKSORT(A, q+1, r)
```

```
PARTITION(A, p, r)
1   x ← A[r]
2   i ← p − 1
3   for j ← p to r − 1
4       do if A[j] ≤ x
5           then i ← i + 1
6                SWAP(A[i], A[j])
7   SWAP(A[i + 1], A[r])
8   return  i + 1
```

Different from the previous implementation, we have now QUICKSORT(*A, q, r*) rather than QUICKSORT(*A, q*+1*, r*) on line 4.

```
QUICKSORT(A, p, r)
1   if  p < r
2       then q ← PARTITION(A, p, r)
3            QUICKSORT(A, p, q-1)
4*           QUICKSORT(A, q, r)
```

Can you comment on this modification from the correctness and efficiency point of view?

5. **[Recurrence equations: 15%]** Solve the following recurrence equations.

(a) (5%) $T(n) = T(n-1) + \lg(n/2)$

(b) (5%) $T(n) = 4T\left(\frac{n}{5}\right) + n\lg n + n^2$

(c) (5%) $T(n) = 3T\left(\frac{n}{2} + 5\right) + n^2$

6. **[Hashing : 20%]** Answer the following questions related to hashing.

   (a) (7%) Demonstrate what happens when we insert the keys 17, 16, 10, 9, 8, 1, 2, 3, 15, 14, 13, 12 into a hash table of size 13 with collision resolved by double hashing. The hash functions are defined by $h_1(k) = k \bmod 13$ and $h_2(k) = 1 + (k \bmod 7)$. You do not need to finish all but up to the moment when you just finish the insertion with the second collision.

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	

(b) (7%) If the function $h_1(k)$ is substituted by $h_1(k) = k \bmod 14$, what will happen? Try to discuss as many problems as possible.

(c) (6%) Describe any defect for the quadratic probing designed by $h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod 24$, where $c_1 = 3$ and $c_2 = 6$.