Algorithms Midterm Exam (Spring 2022)

Total：120%

Name: _____

Student ID #: _____

Question	Score
1 (45%)	
2 (20%)	
3 (10%)	
4 (15%)	
5 (10%)	
6 (20%)	
Total	

1. **[Simple questions: 45%]** Complete the following questions with simple answers.
   (a) (5%) Choose the best sorting method from the set of quicksort, insertion sort, selection sort and mergesort, if the input is given incremental. Briefly explain your answer.

   (b) (5%) To sort a set of numbers, name a strategy to provide a *stable* result if there is only quicksort available.

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

(d) (5%) To store the customer data for a bank, what could be the best data structure? Different from the typical case, let us assume that the data will not be changed very often. That is, there is no much insertions and deletions.

(e) (10%) Use your own words to explain the following piece of proof,

$$n! \leq \ell \leq 2^h$$
$$h \geq \lg(n!) = \Omega(n \lg n) \text{ ,}$$

which is used to prove the lower bound of the worst case for comparison sort.

(f) (10%) On double hashing, compare two approaches: (1) the first hash function is chosen to be a multiplication method and the second hash function remains to be a division method and (2) the first hash function remains to be a division method and the second hash function is chosen to be a multiplication method. Which one makes sense more and why?

(g) (5%) What is the best and the worst part of this course so far? Do you find the online videos useful for your study?

2. **[Recurrence equations: 20%]** Solve the recurrence equations in (a) and (b). You can assume the integer arguments for all cases. The answer should be an asymptotically tight solution. You are welcome to apply the master theorem when there is a need. After that, answer the question in (c)

   (a) (5%) $T(n) = \sqrt{5}T(n/5) + n + n^2$

   (b) (5%) $T(n) = 2T(n/3 - 5) + n^2/\lg n$

(c) (10%) Suppose we apply the Master theorem to of the following two recurrence equations

$T(n) = aT\left(\frac{n}{b}\right) + n^k$ and $T(n) = aT\left(\frac{n}{b}\right) + n^{2k}$,

and obtain the same result. What conclusion you can draw over here?

3. **[Quicksort: 15%]** We have the following code for the quicksort algorithm implementation which combines insertion sort and quicksort. (Note that this implementation is different from the version we discussed in class and it does not use the median-3 strategy to find pivots.)

```
ISS2-QUICKSORT(A, p, r, M)
1  if   r – p + 1 ≥ M
2     then  q ← PARTITION(A, p, r)
3           ISS2-QUICKSORT(A, p, q – 1)
4           ISS2-QUICKSORT(A, q + 1 ,r)
5  else if  p < r
6     then  INSERTION-SORT(A, p, r)
```

(a) If someone chooses a big M such as $M = n/4$ where n is the number of inputs, what could be the time complexity for this quicksort implementation? Discuss both the best and worst cases.

(b) If someone chooses a small M such as $M = 2$, what could the time complexity again?

4. **[Heap: 10%]** We have the following codes for heap manipulation. Answer the questions after that.

MAX-HEAPIFY(A, i)

```
 1  l → LEFT(i)
 2  r → RIGHT(i)
 3  if   l ≤ heap-size[A] and A[l] > A[i]
 4  then   largest  ← l
 5  else   largest  ← i
 6  if   r ≤ heap-size[A] and A[r] > A[largest]
 7  then   largest ← r
 8  if   largest ≠ i
 9     then   SWAP(A[i], A[largest])
10          MAX-HEAPIFY(A, largest)
```

BUIDE-MAX-HEAP(A)

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

HEAP-INCREASE-KEY(A, i, key)

```
1  if key < A[i]
2      then error "new key is smaller than current key"
3  A[i] ← key
4  while i > 1 and A[Parent(i)] < A[i]
5      do SWAP(A[i], A[Parent(i)])
6           i ← Parent(i)
```

(a) (5%) If an array $(a_1, a_2, \dots, a_n)$ is a max-heap, now we add one more element to make the array to be $(a_1, a_2, \dots, a_{n+1})$, how can we ensure the new array is also a max-heap?

(b) (5%) What is the time complexity of (a)?

5. **[Counting sort : 10%]** In the counting sort implementation, if the code in line 4 is wrongly put by

   for $i \leftarrow$ length[A]

   what will happen?

```
COUNTING-SORT(A, B)
1   for j ← 0 to m      do C[j] ← 0
2   for i ← 1 to length[A]      INCR(C[A[i]])
3   for j ← 1 to m      do C[j] = C[j-1]+C[j]
4   for i ← length[A] downto 1
5                   do B[C[A[i]]] ← A[i]
6                 DESC(C[A[i]])
```

6. **[Hashing : 20%]** Answer the following questions related to hashing.
   (a) (8%) Demonstrate what happens when we insert the keys 10, 9, 8, 1, 2, 3, 15, 14, 13, 12 into a hash table of size 11 with collision resolved by both linear probing and double hashing. The hash functions are defined by $h_1(k) = k \bmod 11$ and $h_2(k) = 1 + (k \bmod 7)$.
   (b) (5%) How many probes can we save on average, if we shift from linear probing to double hashing?
   (c) (7%) What may go wrong if both hash functions share the same divisor such as $h_2(k) = 1 + (k \bmod 11)$?