# CSE 100: Algorithm Design and Analysis
# Midterm Exam 2

Spring 2019

**Note: Please write down your name on *every* page. 9 pages in total including this cover. Time: 4:30-5:45pm**

| Problem | Points earned | Out of |
|---|---|---|
| 1 | | 14 |
| 2 | | 8 |
| 3 | | 9 |
| 4 | | 4 |
| 5 | | 5 |
| 6 | | 10 |
| 7 | | 10 |
| 8 | | 10 |
| 9 | | 10 |
| 10 | | 10 |
| Sum | | Max 90 |

Name ______________________

1. **[14 points]** For each of the following claims, determine if it is true or false. *No* explanation is needed.

   (a) **[2 points]** Counting-sort is an in-place sorting algorithm.
   **True** **False Sol.** False

   (b) **[2 points]** The average running time of the randomized selection algorithm is $O(n)$ for all inputs.
   **True** **False Sol.** True

   (c) **[2 points]** The running time of Heap-sort is $\Theta(n^2)$.
   **True** **False Sol.** False

   (d) **[2 points]** Suppose we resolve hash table collisions using chaining. Under the simple uniform hashing assumption, an unsuccessful search takes $\Theta(1 + \alpha)$ time in expectation where $\alpha$ is the load factor of the current hash table.
   **True** **False Sol.** True

   (e) **[2 points]** The decision tree of *any* comparison based sorting algorithm has a height $O(n \log n)$.
   **True** **False Sol.** False

   (f) **[2 points]** If we solve $T(n) = T(\frac{9}{10}n) + T(\frac{1}{10}n) + \Theta(n)$, then we obtain $T(n) = \Theta(n)$.
   **True** **False Sol.** False

   (g) **[2 points]** One can sort $n$ integers in the range between 0 and $n^{10}$ in $O(n)$ time.
   **True** **False Sol.** True

2. **[8 points]** In class we analyzed the running time of building a max-heap. For each of the following multiple-choice questions regarding the analysis, choose the correct answer.

   (a) Suppose the binary heap has a height $H$. What is $H$ in terms of $n$ asymptotically?
   $\Theta(n^2)$**;** $\Theta(n)$**;** $\Theta(\log n)$**; or** $\Theta(1)$ **Sol.** $\Theta(n)$

   (b) What is the number of nodes of height $h$ in the binary heap when $h \geq 1$?
   $\Theta(n/2^h)$**;** $\Theta(2^h)$**;** $\Theta(h)$**; or** $\Theta(nh)$ **Sol.** $\Theta(n/2^h)$

   (c) What is the running time of Max-Heapify on a node of height $h$ when $h \geq 1$? You must choose the tightest answer.
   $O(n/2^h)$**;** $O(2^h)$**;** $O(h)$**; or** $O(nh)$ **Sol.** $O(h)$

   (d) What is the running time of building a max-heap? You must choose the tightest answer.
   $O(n^2)$**;** $O(n \log n)$**;** $O(n)$**; or** $O(\log n)$ **Sol.** $O(n)$

3. (a) [**3 points**] Briefly explain the uniform hashing assumption.
   **Sol.** Any given element is equally likely to hash into any of the $m$ slots (independently of where any other element has hashed to).

   (b) [**3 points**] What does stable-sort mean? Name one sorting algorithm that is stable-sort.
   **Sol.** ([**2 points**] Elements with the same key value (keys with the same value) appear in output in the same order as they did in input. [**1 points**] Counting sort, Insertion sort;, Radix sort is also acceptable.

   (c) [**3 points**] Suppose the input is an array of $n$ fractional numbers sampled from $[0, 1)$ uniformly at random. To sort the array in $O(n)$ time in expectation, which sorting algorithm would you use?
   **Sol.** Bucket sort

4. [**4 points**] Let $A[1 \cdots 9] = \langle 3, 2, 9, 0, 7, 5, 4, 8, 6 \rangle$. Illustrate the execution of Randomized-Select(A, 1, 9, 7) with Randomized-Partition replaced with Partition; see the last question for the pseudocode of Partition, which is exactly the same one appearing in the textbook. More precisely, **list all calls** to Randomized-Select$(A, p, r, i)$ with $p, r, i$ specified. Recall that Randomized-Select$(A, p, r, i)$ is supposed to return the $i$th smallest element in $A[p \cdots r]$.
   **Sol.** Randomized-Select(A, 1, 9, 7)
   Randomized-Select(A, 7, 9, 1)
   Randomized-Select(A, 7, 7, 1)

5. **[5 points]** The following is a pseudocode of the naive divide-and-conquer algorithm for matrix multiplication. Here, partitioning a $n$ by $n$ matrix means partitioning it into four $n/2$ by $n/2$ (sub-)matrices.

```
SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)
n = A.rows
let C be a new n × n matrix
if n == 1
    c11 = a11 · b11
else partition A, B, and C as in equations (4.9)
    C11 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B11)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B21)
    C12 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A11, B12)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A12, B22)
    C21 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B11)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B21)
    C22 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A21, B12)
          + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A22, B22)
return C
```

Give the recurrence for the running time of Square-Matrix-Multiply-Recursive and solve it. We let $T(n)$ denote the running time when input matrices are $n$ by $n$. No need to show how you solved it. Just state the final result along with the recurrence.

Recurrence:
$T(n) =$

After solving the recurrence,
$T(n) =$

**Sol.** $T(n) = 8T(n/2) + \Theta(n^2)$. (3 pts)
$T(n) = \Theta(n^3)$. (2 pts)
It is okay to use $O$ in place of $\Theta$.

6. **[10 points]** Solve the following recurrences using the *Master Theorem*. You must state *which case applies and set parameters* (like $a, b, c, \epsilon$). If the theorem is not applicable, just say N/A.

> ***Theorem 4.1 (Master theorem)***
> Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence
>
> $T(n) = aT(n/b) + f(n)$ ,
>
> where we interpret $n/b$ to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:
>
> 1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
> 2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
> 3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large $n$, then $T(n) = \Theta(f(n))$. ■

(a) $T(n) = 4T(n/2) + n^3$.
**Sol.** Case 3. $a = 4$, $b = 2$, $\epsilon$ can be any positive constant $\leq 1$; and $c$ can be any constant such that $1/2 \leq c \leq 1$. Therefore, $T(n) = \Theta(n^3)$.

(b) $T(n) = 9T(n/3) + 10n^2$.
**Sol.** Case 2. $a = 9$ and $b = 3$. Therefore, $T(n) = \Theta(n^2 \log n)$.

7. **[10 points]** Solve the following recurrence using the *recursion tree method*: $T(n) = 3T(n/2) + n$. To get full points, the following quantities must be clear from your solution: the tree depth (be careful with the log base), each subproblem size at depth $d$, the number of nodes at depth $d$, workload per node at depth $d$, (total) workload at depth $d$.

   **Sol.** The tree visualization is omitted. (rough visualization: 1pts)
   For simplicity, let $T(1) = 1$. Tree depth $D = \log_2 n$. (1.5pts)
   Each subproblem size at depth $d$: $n/2^d$.(1.5pts)
   Number of nodes at depth $d$: $3^d$ (1.5pts)
   WL per node at depth $d$: $n/2^d$ (1.5pts)
   WL at depth $d$: $n(3/2)^d$. (1.5pts)
   Final answer: $\sum_{d=0}^{D}(3/2)^d n = \Theta((3/2)^D n) = \Theta(n^{\log_2 3})$ (1.5pts)

8. **[10 points]** Consider a hash table with $m = 10$ slots and using the hash function $h(k) = k$ mod 10. Say we insert (elements of) keys $k = 3, 5, 13, 23, 15, 29$ in this order. Show the final table in the following two cases. In both cases the same hash function $h(k)$ is used.

   (a) **[5 points]** When chaining is used to resolve collisions. Insert the element at the *beginning* of the linked list.

   **Sol.**

   slot 3 has a linked list storing 23, 13, 3 in this order.
   slot 5 has a linked list storing 15, 5, in this order.
   slot 9 has a linked list storing 29. All other slots have NIL.

   Any ordering of elements with the same hash value is acceptable. Rubric: if you has an integer to a wrong slot, -1.

   (b) **[5 points]** When open addressing and linear probing are used to resolve collisions, i.e. $h(k, i) = k + i \bmod 10$.

   **Sol.**

   0: NIL
   1: NIL
   2: NIL
   3: 3
   4: 13
   5: 5
   6: 23
   7: 5
   8: NIL
   9: 29

   Rubric: for each integer in a wrong position, -1.

9. **[10 points]** Give a *pseudocode* of Heap-sort. For simplicity, you can assume that the heap $A$ you're given is *already* a max-heap. You can use the function $\textsc{Max-Heapify}(i)$ as a sub-procedure. Recall that the function $\textsc{Max-Heapify}(i)$ makes the subtree rooted at node $i$ a max-heap, *if both* the left and right subtrees of node $i$ are max-heaps. You can use $A.heapsize$ to denote the current heap size. If you can't give a pseudocode, you can describe the Heap-sort algorithm in words, but you will lose some points.

   **Sol.** See CLRS.

10. **[10 points]** Quicksort implementation. Give a *pseudo-code* of Quicksort$(A, p, r)$ that sorts $A[p...r]$ via quick-sort. You can assume that all elements in $A$ have distinct values. You can use the following helper function, Partition$(A, p, r)$:

```
Partition(A, p, r)
1. x = A[r]
2. i =  p - 1
3. for j = p to r-1
4.    if A[j] <= x
5.       i = i+1
6.       exchange A[i] with A[j]
7. exchange A[i+1] with A[r]
8. return i +1
```

**Sol.** See CLRS.