CSE 100: Algorithm Design and Analysis: Midterm Exam 2

Fall 2021

Note: This is a **closed** book examination. You have 75 minutes to answer as many questions as possible. The number in the square bracket at the beginning of each question indicates the number of points given to the question. Write all of your answers directly on this paper; if you take this exam online, you can solve multiple problems on the same page, but you should clearly write down the problem numbers. Please make your answers concise as well as precise. If there is something in the question that you believe is open to interpretation, then please go ahead and interpret, but state your assumptions in your answer. There is no penalty for attempting to answer a question with the wrong answer, so please answer as many questions as you can. Partial credit will be given even if the answer is not fully correct. If you run out of space you can ask for extra papers. Please write down your name on *every* page as some papers may fall off although it rarely happens. 12 pages in total including this cover and the last page which is intentionally left blank (please check!). Any points above 100 will be considered as bonus. Time: 4:30-5:45pm.

For students who take this exam remotely:

1. You must upload your solutions to catcourses by 5:55pm.
2. If you failed uploading your solutions by 5:55pm, you can email them to the instructor, at `sim3@ucmerced.edu`, but it will incur 1 pt penalty for you for *every* minute delay after 5:55pm. (e.g. If the instructor received your solutions at 5:59pm, you will lose 4pts).
3. After the submission, double check if you submitted the right files and if they are readable.

Problem	Points earned	Out of
1		20
2		18
3		10
4		10
5		6
6		10
7		10
8		13
9		5
10 (bonus)		25
Sum		Max 127

Name ______________________

1. **[20 points]** For each of the following claims, determine if it is true or false. *No* explanation is needed.

   (a) **[2 points]** The standard Mergesort (the implementation in the textbook) is an in-place sorting.
   True **False** **Sol.** False

   (b) **[2 points]** The *expected* running time of the randomized quicksort algorithm (that chooses the pivot randomly) is $O(n \log n)$ for all inputs.
   True **False** **Sol.** True

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

   (d) **[2 points]** If we use a hash table, there must be a slot/bucket that has at least α keys assigned to it, where α is the load factor of the current hash table.
   True **False** **Sol.** True

   (e) **[2 points]** A binary search tree of n nodes may have a depth/height as large as $\Omega(n)$.
   True **False** **Sol.** True

   (f) **[2 points]** If $A[1...n]$ is *sorted*, we can find the k th smallest element in $A[1...n]$ in $O(1)$ time.
   True **False** **Sol.** True

   (g) **[2 points]** One can build a max-heap of n elements in $O(n)$ time.
   True **False** **Sol.** True

   (h) **[2 points]** Given a binary search tree of n nodes, one can sort the keys therein in $O(n)$ time.
   True **False** **Sol.** True

   (i) **[2 points]** Consider a comparison based sorting algorithm and its decision tree on n elements. Then, the path from the root to every leaf node in the tree must have $\Omega(n \log n)$ edges.
   True **False** **Sol.** False

   (j) **[2 points]** We can sort n binary numbers of $100 \log_2 n$ bits in $O(n)$ time assuming a basic bit operation takes $O(1)$ time.
   True **False** **Sol.** True

2. [**18 points**] Explain...

   (a) [**3 points**] What is the main drawback of direct-addressing? Please make your answer concise. **Sol.**

   It wastes too much memory. Or it's not memory-efficient. (Typically, the universe of possible keys is super large compared to the set of actual keys, so lots of space is wasted.)

   (b) [**3 points**] What does counting sort mean? **Sol.**

   If an object a appears before object b before sorting and they have the same key value, then a must appear after b after sorting. Or simply, the relative ordering of elements with the same key value must be preserved.

   (c) [**3 points**] We have tuples of dates of the form $\langle xxxx(year), xx(month), xx(date)\rangle$. Which sorting algorithm would you use to sort them? **Sol.**

   Radix sort

(d) **[3 points]** Explain why the decision tree of any sorting algorithm on n elements must have at least $n!$ leaf nodes. **Sol.** There are $n!$ possible orderings (and every ordering must appear in a leaf node to distinguish among them).

(e) **[6 points]** If we sort n elements using randomized quicksort, what is the probability that the maximum element and the minimum element are compared? Explain why. **Sol.** $\frac{2}{n}$. Because it happens if and only if the maximum or the minimum element is chosen as the first pivot.

3. **[10 points]**

   (a) **[5 points]** You're given an array $A[1 \cdots 8] = \langle 2, 6, 5, 4, 1, 2, 4, 3 \rangle$. Run Max-heapify on the root. What is $A[1 \cdots 8]$? **Sol.** 6, 4, 5, 3, 1, 2, 4, 2.

   (b) **[5 points]** You're given an array $A[1 \cdots 7] = \langle 9, 7, 6, 4, 1, 5, 3 \rangle$. If you execute Max-Heap-Insert(A, 8), what is the resulting A? **Sol.** 9, 8, 6, 7, 1, 5, 3, 4

4. **[10 points]** The following is a *variant* of the $O(n)$ time deterministic algorithm for the Selection problem you learned in class. The algorithm consists of the following five steps. To analyze the algorithm's running time, state what is the running time of *each* step and state the *recurrence* for the running time. For example, the first step's run time is $O(n)$. Use $T(n)$ to denote the algorithm's running time on inputs of size n.

   (a) Divide the n elements into groups of size 11 (so we will have $n/11$ groups). Time: $O(n)$.

   (b) **[2 points]** Find the median of each group (more precisely, the aggregate running time over all groups). Time: **Sol.** $O(n)$.

   (c) **[2 points]** Find the median x of the $n/11$ medians by a recursive call to Select. Time: **Sol.** $T(n/11)$.

   (d) **[2 points]** Call Partition with x as the pivot. Time: **Sol.** $O(n)$.

   (e) **[3 points]** Make a recursive call to Select either on the smaller elements or larger elements (depending on where the element we are looking for lies); if the pivot is the element we are done. Time: **Sol.** $T(\frac{8}{11}n)$.

   [1 points] Recurrence for the running time $T(n) =$

   Sol. $T(n) = T(\frac{1}{11}n) + T(\frac{8}{11}n) + O(n)$.

5. **[6 points]** Consider a hash table with $m = 10$ slots. Suppose we use the hash function $h(k) = k$ mod 10. (Yes, we learned that this is a bad hash function, but this problem is just to test if you understand how chaining works.) Say we insert (elements of) keys $k = 7, 9, 29, 4, 15, 19, 5$ in this order. Show the final table when chaining is used to resolve collisions. Insert the element at the *beginning* of the linked list.

   Sol. slot 4 has a linked list storing 4.
   slot 5 has a linked list storing 5, 15 in this order.
   slot 7 has a linked list storing 7.
   slot 9 has a linked list storing 19, 29, 9.
   All other slots have NIL.

   Any ordering of elements with the same hash value is acceptable. Rubric: For each key value assigned to a wrong slot, -1.

6. **[10 points]** Universal Hash Family. Here, we're considering the following scenario. The universe U of all possible key values is super large, compared to n, the number of actual keys we have.

   (a) **[5 points]** If you choose a hash function from all possible functions from U to $\{0, 1, ..., t-1\}$, what goes wrong? **Sol.** See lecture slides.

   (b) **[5 points]** Suppose $t = 2$. Say $h_1(k)$ is defined as the least significant bit of k. e.g. $h_1(00000) = 0$ and $h_1(00101) = 1$. Similarly, $h_2(k)$ is defined as the second least significant bit of k. e.g. $h_2(00010) = 1$ and $h_2(00000) = 0$. Is $\mathcal{H} = \{h_1, h_2\}$ a universal hash family? Explain why. **Sol.** No. Because for two keys x and y with the same last two digits, $h_1(x) = h_1(y)$ and also $h_2(x) = h_2(y)$, meaning that x and y collide with probability 1; if $\mathcal{H}$ were a universal hash family, the probability must be $1/t = 1/2$.

7. **[10 points]** Randomized-Select implementation. The following is the pseudo-code of Randomized-Select(A, p, r, i) from the textbook where missing parts are underlined. Recall that

   - Randomized-Select(A, p, r, i) is supposed to return the ith smallest element in the subarray $A[p...r]$.
   - Randomized-Partition(A, p, r) randomly chooses a pivot from $A[p...r]$ and returns the pivot index q such that all elements in $A[p...q-1]$ are smaller than $A[q]$ and all elements in $A[q+1...r]$ are greater than $A[q]$.

```
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
```

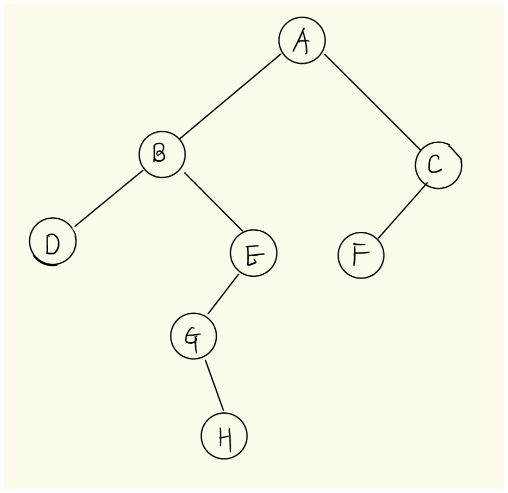
Complete the following pseudocode. You can assume that all elements have distinct values.

```
Randomized-Select(A, p, r, i)
1. if p == r
2.   return _____________________________________________________
3. q = Randomized-Partition(A, p, r)
4. k = q - p + 1
5. if i == k
6.   return _____________________________________________________
7. elseif i < k
8.    return _____________________________________________________

9. else return ___________________________________________________
```

Sol. See CLRS. 2, 2, 3, 3 pts for the four blanks, in this order.

8. **[13 points]** Consider the following binary search tree where nodes are labeled by alphabets. Here the keys are *not* shown in the picture; $a, b, c, \cdots$ are node labels, not keys. Assume that all keys have distinct values.



(a) **[2 points]** What is the node with the min key value? **Sol.** D

(b) **[2 points]** What is the node with the max key value?**Sol.** C

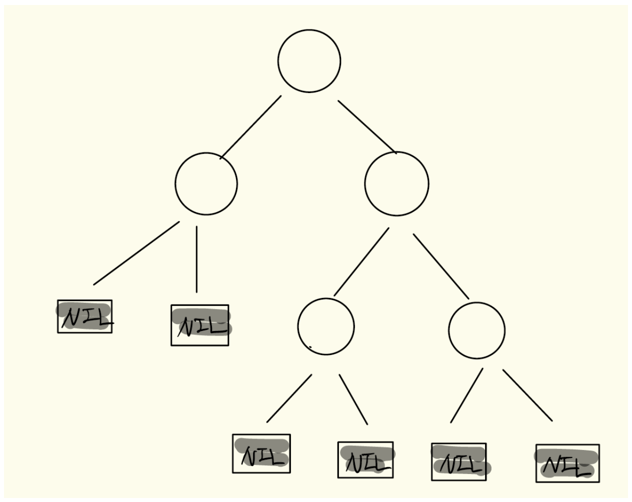
(c) **[2 points]** What is G's successor? **Sol.** H

(d) **[3 points]** Output node labels according to in-order-traversal. **Sol.** D, B, G, H, E, A, F, C

(e) **[4 points]** Delete node B and show the resulting BST.

Sol. See the separate pdf file.

9. **[5 points]** Color each node of the following red-black tree either black or red, to make it a valid red-black tree. You can put B or R in the nodes instead of coloring them. Note that the NIL nodes are already colored black as required.



10. (bonus) [**25 points**] We have n pairs of integers, $p_1 = (x_1, y_1), p_2 = (x_2, y_2), ..., p_n = (x_n, y_n)$ where all entries have integer values between 1 and n. We want to know if there are diagonally adjacent pairs in $O(n)$ time (expected running time is acceptable). We say that two points p_i and p_j are diagonally adjacent if $|x_i - x_j| = 1$ and $|y_i - y_j| = 1$.