Quiz 4

The quiz concerns the run-time of sorting and search algorithms.

1. Sorting algorithms.

(a) **INSERTION-SORT**: sorts an array in place. What is the run-time of INSERTION-SORT?
   - Best ... (choose from $O(lgn), O(n), O(nlgn)$)
   - Worst-case ... (choose from $O(lgn), O(n), O(nlgn), O(n^2)$)

(b) **MERGE-SORT** (randomized): is a divide-and-conquer sorting algorithm. What is the run-time of MERGE-SORT?
   - ... (choose from $O(lgn), O(n), O(nlgn)$)

(c) **HEAPSORT**
   - i. **HEAPIFY(A[i])**: is a subroutine used to maintain the heap property of an array $A[i]$, assuming that $LEFT(i)$ and $RIGHT(i)$ are heaps. What is the run-time of HEAPIFY(A[i])?
     - ... (choose from $O(lgn), O(n), O(nlgn)$)
   - ii. **BUILD-HEAP(A)**: converts an array $A$ into heap. What is the run-time of BUILD-HEAP(A)?
     - ... (choose from $O(lgn), O(n), O(nlgn)$)
   - iii. **HEAPSORT(A)**: uses the above as subroutines to sort the array $A$ into heap. What is the run-time of HEAPSORT(A)?
     - ... (choose from $O(lgn), O(n), O(nlgn)$)

(d) **QUICKSORT(A)**: uses the divide-and-conquer paradigm to sort an array $A$.
   - i. **PARTITION(A, p, r)** is used as a subroutine to first partition the array. What is the run-time of PARTITION(A, p, r)?
     - ... (choose from $\Theta(lgn), \Theta(n), \Theta(nlgn)$)
   - ii. Performance of **QUICKSORT**. What is the run-time of QUICKSORT with:
     - A. Worst case partitioning? ... (Choose from $\Theta(n), \Theta(nlgn), \Theta(n^2)$).
     - B. Best case partitioning? ... (Choose from $\Theta(n), \Theta(nlgn), \Theta(n^2)$).
     - C. Average case partitioning? ... (Choose from $\Theta(n), \Theta(nlgn), \Theta(n^2)$).
     - D. **RANDOMIZED-PARTITION A(p, q, r)**? ... (Choose from $\Theta(n), \Theta(nlgn), \Theta(n^2)$).

(e) **Sorting in linear time**
   - i. **COUNTING-SORT(A, B, C, k)**: assumes that the elements of $A$ are in the range: 1, 2, ..., $k$. It uses two other arrays $B[1..n]$ and $C[1..k]$. What is the run-time of COUNTING-SORT(A, B, C, k) when $k = O(n)$?
     - ... (choose from $O(lgn), O(n), O(nlgn)$)
   - ii. **RADIX-SORT(A, d)**: sorts $d$-digit numbers. What is the run-time of RADIX-SORT(A, d)?
     - ... (choose from $O(lgn), O(n), O(nlgn)$)
   - iii. **BUCKET-SORT(A)**: sorts by dividing the interval (0, 1] into $n$ equal sized subintervals (buckets) and then distributing the numbers of $A$ uniformly into these. What is the run-time of BUCKET-SORT(A)?
     - ... (choose from $O(lgn), O(n), O(nlgn)$)

2. Search algorithms.

(a) **Selecting in random linear time**. **RANDOMIZED-SELECT(A)** uses a modified version of **RANDOMIZED-PARTITION(A)**. What is the run-time of **RANDOMIZED-SELECT(A)**?
   - ... (choose from $O(n), O(nlgn)$)

(b) **Selecting in worst-case linear time**. **SELECT(A)** uses a deterministic partitioning routine **PARTITION(A)** (from QuickSort). What is the worst-case run-time of **SELECT(A)**?
   - ... (choose from $O(n), O(nlgn)$)