Final Exam

Back to Week 7

40/40 points earned (100%)

Quiz passed!

---

2 / 2 points

1.
Recall the Partition subroutine that we used in both QuickSort and RSelect. Suppose that the following array has just been partitioned around some pivot element: 3, 1, 2, 4, 5, 8, 7, 6, 9

Which of these elements could have been the pivot element? (Hint: Check all that apply, there could be more than one possibility!)

---

2 / 2 points

2.
Here is an array of ten integers: 5 3 8 9 1 7 0 2 6 4

Suppose we run MergeSort on this array. What is the number in the 7th position of the partially sorted array after the outermost two recursive calls have completed (i.e., just before the very last Merge step)? (When we say "7th" position, we're counting positions starting at 1; for example, the input array has a "0" in its 7th position.)

---

2 / 2 points

3.
What is the asymptotic worst-case running time of MergeSort, as a function of the input array length n?

✔ 2 / 2 points

4.
Consider a directed graph $G = (V, E)$ with non-negative edge lengths and two distinct vertices s and t of V. Let P denote a shortest path from s to t in G. If we add 10 to the length of every edge in the graph, then: [Check all that apply.]

✔ 2 / 2 points

5.
What is the running time of depth-first search, as a function of n and m, if the input graph $G = (V, E)$ is represented by an adjacency matrix (i.e., NOT an adjacency list), where as usual $n = |V|$ and $m = |E|$?

✔ 2 / 2 points

6.
What is the asymptotic running time of the Insert and Extract-Min operations, respectively, for a heap with n objects?

✔ 2 / 2 points

7.
On adding one extra edge to a directed graph G, the number of strongly connected components...?

✔ 2 / 2 points

8.
What is the asymptotic running time of Randomized QuickSort on arrays of length n, in expectation (over the choice of random pivots) and in the worst case, respectively?

✔ 2 / 2 points

points

9.

Let f and g be two increasing functions, defined on the natural numbers, with $f(1), g(1) \geq 1$. Assume that $f(n) = O(g(n))$. Is $2^{f(n)} = O(2^{g(n)})$? (Multiple answers may be correct, check all that apply.)

---

2 / 2 points

10.

Let $0 < \alpha < .5$ be some constant. Consider running the Partition subroutine on an array with no duplicate elements and with the pivot element chosen uniformly at random (as in QuickSort and RSelect). What is the probability that, after partitioning, both subarrays (elements to the left of the pivot, and elements to the right of the pivot) have size at least α times that of the original array?

---

2 / 2 points

11.

Which of the following statements hold? (As usual n and m denote the number of vertices and edges, respectively, of a graph.) [Check all that apply.]

---

2 / 2 points

12.

When does a directed graph have a unique topological ordering?

---

2 / 2 points

13.

Suppose that a randomized algorithm succeeds (e.g., correctly computes the minimum cut of a graph) with probability p (with $0 < p < 1$). Let ϵ be a small positive number (less than 1).

How many independent times do you need to run the algorithm to ensure that, with probability at least $1 - \epsilon$, at least one trial succeeds?

✔ 2 / 2 points

14.
Suppose you implement the operations Insert and Extract-Min using a *sorted* array (from biggest to smallest). What is the worst-case running time of Insert and Extract-Min, respectively? (Assume that you have a large enough array to accommodate the Insertions that you face.)

✔ 2 / 2 points

15.
Which of the following patterns in a computer program suggests that a heap data structure could provide a significant speed-up (check all that apply)?

✔ 2 / 2 points

16.
Which of the following patterns in a computer program suggests that a hash table could provide a significant speed-up (check all that apply)?

✔ 2 / 2 points

17.
Which of the following statements about Dijkstra's shortest-path algorithm are true for input graphs that might have some negative edge lengths? [Check all that apply.]

✔ 2 / 2 points

18.
Suppose you are given k sorted arrays, each with n elements, and you want to combine them into a single array of kn elements. Consider the following approach. Divide the k arrays into $k/2$ pairs of arrays, and use the Merge subroutine taught in the MergeSort lectures to combine each pair. Now you are left with $k/2$ sorted arrays, each with $2n$ elements. Repeat this approach until you have a single sorted array with kn elements. What is the running time of this procedure, as a function of k and n?

2 / 2
points

19.

Running time of Strassen's matrix multiplication algorithm: Suppose that the running time of an algorithm is governed by the recurrence $T(n) = 7 * T(n/2) + n^2$. What's the overall asymptotic running time (i.e., the value of $T(n)$)?

2 / 2
points

20.

Recall the Master Method and its three parameters a, b, d. Which of the following is the best interpretation of b^d, in the context of divide-and-conquer algorithms?