Final Exam

10 questions

1 point

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!)

- [ ] 5
- [ ] 2
- [ ] 9
- [ ] 3
- [ ] 4

1 point

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

- 4
- 3
- 1

---

1 point

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

- $\theta(\log n)$
- $\theta(n^2)$
- $\theta(n)$
- $\theta(n \log n)$

---

1 point

4.
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?

- $\Theta(n \log n)$ [expected] and $\Theta(n^2)$ [worst case]
- $\Theta(n \log n)$ [expected] and $\Theta(n \log n)$ [worst case]
- $\Theta(n^2)$ [expected] and $\Theta(n^2)$ [worst case]
- $\Theta(n)$ [expected] and $\Theta(n \log n)$ [worst case]

---

1 point

5.

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.)

- [ ] Never
- [ ] Always
- [ ] Yes if $f(n) \leq g(n)$ for all sufficiently large n
- [ ] Maybe, maybe not (depends on the functions f and g).

---

1 point

6.

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?

- ○ α
- ○ $1 - \alpha$
- ○ $2 - 2\alpha$
- ○ $1 - 2\alpha$

---

1 point

7.

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?

- ○ $\frac{\log \epsilon}{\log(p)}$
- ○

- $\frac{\log(1-p)}{\log \epsilon}$
- $\frac{\log(p)}{\log \epsilon}$
- $\frac{\log \epsilon}{\log(1-p)}$

---

1 point

8.

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?

- $\theta(nk \log k)$
- $\theta(nk \log n)$
- $\theta(n \log k)$
- $\theta(nk^2)$

---

1 point

9.

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 $T(n)$)?

- $\theta(n^{\frac{\log\ 2}{\log\ 7}})$
- $\theta(n^2 \log n)$
- $\theta(n^{\log_2(7)})$
- $\theta(n^2)$

1 point

10.

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?

- The rate at which the total work is growing (per level of recursion).
- The rate at which the number of subproblems is growing (per level of recursion).
- The rate at which the subproblem size is shrinking (per level of recursion).
- The rate at which the work-per-subproblem is shrinking (per level of recursion).

8 questions unanswered

Upgrade to submit