Coursera BETA

Iryna Cherniavska    **Preferences**    All Courses    About    Contact Us    Logout

**Stanford University**
Design and Analysis of Algorithms I

**Tim Roughgarden**
Associate Professor of Computer Science

Home

Syllabus

Video Lectures

Problem Sets

Programming Questions

Theory Problems

Discussion Forums

Course Logistics

Help with Subtitles

# Feedback — Final Exam

## You achieved a score of 22.00 out of 30.00

## Question 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: There could be more than one possibility!)

| Your Answer | | Score | Explanation |
|---|---|---|---|
| ☐ 3 | ✔ | 0.20 | Because there is a "1" and "2" to the right of it, "3" could not have been the pivot. |
| ☑ 5 | ✔ | 0.20 | All numbers to the left of it are less than it; all numbers to the right of it are bigger than it; so it could have been the pivot. |
| ☐ 2 | ✔ | 0.20 | Because there is a "3" to the left of it, "2" could not have been the pivot. |
| ☑ 4 | ✔ | 0.20 | All numbers to the left of it are less than it; all numbers to the right of it are bigger than it; so it could have been the pivot. |
| ☑ 9 | ✔ | 0.20 | All numbers to the left of it are less than it; all numbers to the right of it are bigger than it; so it could have been the pivot. |
| Total | | 1.00 / 1.00 | |

## Question 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.)

| Your Answer | | Score | Explanation |
|---|---|---|---|
| ◉ 2 | ✔ | 1.00 | The array at this point of the algorithm is 1 3 5 8 9 0 2 4 6 7. |
| Total | | 1.00 / 1.00 | |

## Question 3

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

| Your Answer | Score | Explanation |
|---|---|---|
| ◉ $\theta(n \log n)$ ✔ | 1.00 | As discussed in lecture. |
| Total | 1.00 / 1.00 | |

## Question 4

Consider a directed graph $G = (V, E)$ with non-negative edge lengths and two 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:

| Your Answer | Score | Explanation |
|---|---|---|
| ☐ $P$ definitely remains a shortest $s - t$ path. ✔ | 0.25 | As discussed in lecture, adding a constant to every edge can change the shortest path. |
| ☐ $P$ definitely does not remain a shortest $s - t$ path. ✔ | 0.25 | If you get lucky (e.g., $P$ has only one edge) then $P$ might remain a shortest path. |
| ☑ $P$ might or might not remain a shortest $s - t$ path (depending on the graph). ✔ | 0.25 | As discussed in lecture, adding a constant to every edge can change the shortest path. (But it might not, if you get lucky.) |
| ☑ If $P$ has only one edge, then $P$ definitely remains a shortest $s - t$ path. ✔ | 0.25 | Every path $P$ goes up in length by at least as much as $P$ does, so $P$ remains the shortest. |
| Total | 1.00 / 1.00 | |

## Question 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 $m = |E|$ ?

| Your Answer | Score | Explanation |
|---|---|---|
| ◉ $\theta(n^2)$ ✔ | 1.00 | For the lower bound, in the worst case you have to look at every entry twice. For the upper bound, one easy approach is construct the adjacency list from the adjacency matrix (this is easy to do with a scan over the matrix) and then use the subroutine from the video lectures. |
| Total | 1.00 / 1.00 | |

## Question 6

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

| Your Answer | Score | Explanation |
|---|---|---|
| ◉ $\Theta(\log n)$ and $\Theta(\log n)$ ✔ | 1.00 | |
| Total | 1.00 / 1.00 | |

## Question 7

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

| Your Answer | | Score | Explanation |
|---|---|---|---|
| ◉ ...might or might not remain the same (depending on the graph). | ✔ | 1.00 | That's correct! |
| Total | | 1.00 / 1.00 | |

## Question 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?

| Your Answer | | Score | Explanation |
|---|---|---|---|
| ◉ $\Theta(n)$ [expected] and $\Theta(n \log n)$ [worst case] | ✖ | 0.00 | The expected case was shown to be $O(n \log n)$ in lecture. In the worst case, you might repeatedly pick the minumum remaining element as the pivot. |
| Total | | 0.00 / 1.00 | |

## Question 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.)

| Your Answer | | Score | Explanation |
|---|---|---|---|
| ☑ Maybe, maybe not (depends on the functions $f$ and $g$). | ✔ | 0.25 | For a positive example, take $f(n) = g(n) = n$. For a negative example, take $f(n) = 2n$ and $g(n) = n$. |
| ☑ Yes if $f(n) \leq g(n)$ for all sufficiently large $n$ | ✔ | 0.25 | Take $c = 1$ and $n_0$ sufficiently large. |
| ☐ Always | ✔ | 0.25 | Not if $f(n) = 2n$ and $g(n) = n$. |
| ☐ Never | ✔ | 0.25 | Yes if $f(n) = g(n) = n$. |
| Total | | 1.00 / 1.00 | |

## Question 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 $\alpha$ times that of the original array?

| Your Answer | | Score | Explanation |
|---|---|---|---|
| ◉ $1 - 2\alpha$ | ✔ | 1.00 | That's correct! |

| Total | 1.00 / 1.00 |
|-------|-------------|

## Question 11

Which of the following statements hold? (As usual $n$ and $m$ denote the number of vertices and edges, respectively, of a graph.)

| Your Answer | | Score | Explanation |
|-------------|---|-------|-------------|
| ☑ Breadth-first search can be used to compute shortest paths in $O(m+n)$ time (when every edge has unit length). | ✔ | 0.50 | As covered in lecture. |
| ☑ Breadth-first search can be used to compute the connected components of an undirected graph in $O(m+n)$ time. | ✔ | 0.50 | As covered in lecture. |
| ☑ Depth-first search can be used to compute the strongly connected components of a directed graph in $O(m+n)$ time. | ✔ | 0.50 | As covered in lecture. |
| ☑ Depth-first search can be used to compute a topological ordering of a directed acyclic graph in $O(m+n)$ time. | ✔ | 0.50 | As covered in lecture. |
| Total | | 2.00 / 2.00 | |

## Question 12

When does a directed graph have a unique topological ordering?

| Your Answer | | Score | Explanation |
|-------------|---|-------|-------------|
| ◉ Whenever it is directed acyclic | ✖ | 0.00 | It might have many topological orderings. |
| Total | | 0.00 / 2.00 | |

## Question 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 $\epsilon$ 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?

| Your Answer | | Score | Explanation |
|-------------|---|-------|-------------|
| ◉ $\dfrac{\log(p)}{\log \epsilon}$ | ✖ | 0.00 | The failure probability after $i$ independent trials is $(1-p)^i$. To get the correct answer, set this equal to the desired failure probability $\epsilon$, take logarithms (the base doesn't matter) and solve for $i$. |
| Total | | 0.00 / 2.00 | |

## Question 14

Suppose you implement the operations Insert and Extract-Min using a *sorted* array (e.g., 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.)

| Your Answer | Score | Explanation |
|---|---|---|
| ⦿ $\Theta(n)$ and $\Theta(1)$ ✔ | 2.00 | As in Homework #5. To keep the array sorted you need to do $\Theta(n)$ worst-case work on an Insert. |
| Total | 2.00 / 2.00 | |

## Question 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)?

| Your Answer | Score | Explanation |
|---|---|---|
| ☑ Repeated minimum computations ✔ | 0.50 | E.g., in Dijkstra's shortest-path algorithm or in HeapSort. |
| ☑ Repeated maximum computations ✔ | 0.50 | Heaps are just as useful for these as for minimum computations (e.g., just store the negative of each key, so the Extract Min winds up extracting the max). |
| ☐ Repeated lookups ✔ | 0.50 | Heaps do not generally support fast lookups (unless you happen to be looking for the minimum). |
| ☐ None of these ✔ | 0.50 | Useful in, e.g., in Dijkstra's shortest-path algorithm or in HeapSort. |
| Total | 2.00 / 2.00 | |

## Question 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)?

| Your Answer | Score | Explanation |
|---|---|---|
| ☐ Repeated minimum computations ✔ | 0.50 | Hash tables don't support fast queries that reference the ordering of the keys (just lookups). |
| ☐ Repeated maximum computations ✔ | 0.50 | Hash tables don't support fast queries that reference the ordering of the keys (just lookups). |
| ☑ Repeated lookups ✔ | 0.50 | The raison d'etre of a hash table. |
| ☐ None of these ✔ | 0.50 | Hash tables are super-useful for repeated lookups. |
| Total | 2.00 / 2.00 | |

## Question 17

Which of the following statements about Dijkstra's shortest-path algorithm are true for input graphs that might have some negative edge lengths?

| Your Answer | Score | Explanation |
|---|---|---|
| ☐ It is guaranteed to terminate. | ✖ 0.00 | |
| ☑ It may or may not terminate (depending on the graph). | ✖ 0.00 | |
| ☐ It is guaranteed to correctly compute shortest-path distances (from a given source vertex to all other vertices). | ✔ 0.50 | We gave a counterexample in lecture. |
| ☑ It may or may not correctly compute shortest-path distances (from a given source vertex to all other vertices), depending on the graph. | ✔ 0.50 | |
| Total | 1.00 / 2.00 | |

## Question 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 $n$?

| Your Answer | Score | Explanation |
|---|---|---|
| ⊙ $\theta(nk \log n)$ | ✖ 0.00 | There are $\Theta(\log k)$ iterations (you terminate once you've divided $k$ by two enough times to get to 1), and each iteration takes $\Theta(nk)$ time. |
| Total | 0.00 / 2.00 | |

## Question 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 $T(n)$)?

| Your Answer | Score | Explanation |
|---|---|---|
| ⊙ $\theta(n^{\log_2(7)})$ | ✔ 2.00 | That's correct! |
| Total | 2.00 / 2.00 | |

## Question 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?

| Your Answer | Score | Explanation |
|---|---|---|
| ⊙ The rate at which the work-per-subproblem is shrinking (per level of recursion). | ✔ 2.00 | That's correct! |
| Total | 2.00 / | |

2.00