Instructions (for the real exam)

- **DO NOT OPEN THE EXAM UNTIL YOU ARE INSTRUCTED TO.**
- Answer all of the questions as well as you can. You have three hours.
- The exam is **non-collaborative**; you must complete it on your own. If you have any clarification questions, please ask the course staff (we are outside the exam room). We cannot provide any hints or help.
- This exam is **closed-book**, except for **up to two double-sided sheets of paper** that you have prepared ahead of time. You can have anything you want written on these sheets of paper.
- **Please DO NOT separate pages of your exam**. The course staff is not responsible for finding lost pages, and you may not get credit for a problem if it goes missing.
- There are a few pages of extra paper at the back of the exam in case you run out of room on any problem. If you use them, please clearly indicate on the relevant problem page that you have used them, and please clearly label any work on the extra pages.
- Please do not discuss the exam before solutions are posted. *[Obviously it's okay to discuss this **practice** exam :) But these are the instructions on the main exam.]*

General Advice

- If you get stuck on a question or a part, move on and come back to it later. The questions on this exam have a wide range of difficulty, and you can do well on the exam even if you don't get a few questions.
- Pay attention to the point values. Don't spend too much time on questions that are not worth a lot of points.
- There are **105 + 5** (bonus) total points on this exam. There are **six problems** split across **14 pages**.

Name and SUNet ID (please print clearly):

This page intentionally blank. Please do not write anything you want graded here.

Honor Code

The following is a statement of the Stanford University Honor Code:

1. *The Honor Code is an undertaking of the students, individually and collectively:*

   (1) that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;

   (2) that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

2. *The faculty on its part manifests its confidence in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code.*

3. *While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.*

For my part, I believe that we have upheld our end of the agreement in Item 2. I don't think we are taking unusual or unreasonable precautions that would indicate a lack of confidence in the honor of students, and I believe that the in-person setting avoids temptations ot violate the honor code to the extent practicable.

[signed, Mary Wootters]

Please acknowledge that you have held up your end of the agreement in Item 1:

I have abided by the Honor Code, and in particular the policies listed above, both in letter and in spirit, while taking this exam.

signed, _____

Good Luck!

This page intentionally blank. Please do not write anything you want graded here.

1. **(20 pt.)[Multiple Choice]** For each of the following, select **ALL answers that apply.**

   [**We are expecting:** *For all parts of this problem, just circle all of the answers that apply. No justification is required or will be considered for grading.*]

   (i) Which of the following algorithms runs in (worst-case) $O(n^2)$ time? Select all that apply.

   (A) Karatsuba Multiplication, where n is the number of digits in each number.

   (B) MergeSort, where n is the number of elements we're sorting.

   (C) QuickSort, where n is the number of elements we're sorting, and the chosen pivot is random.

   (D) k-SELECT, where n is the number of elements in the input array, and the chosen pivot is random.

   (E) None of the above.

   (ii) Recall that the formal definition of big-O is:
   "$f(n)$ is $O(g(n))$ if there exist some values $c, n_0 > 0$ such that $f(n) \leq c(g(n))$ for all $n \geq n_0$."
   Which of the following are true about this definition? Select all that apply.

   (A) c and n_0 **cannot** have the same value.

   (B) c and n_0 **must** be integers.

   (C) If $f(n) = g(n)$, then $f(n) = O(g(n))$.

   (D) $g(n)$ **cannot** be $O(1)$.

   (E) None of the above.

   (iii) Let T be a binary search tree storing n elements. Which of the following statements are necessarily true?

   (A) The number of nodes in T is n.

   (B) The number of leaves in T is $\Theta(n)$.

   (C) The height of T is $O(\log n)$

   (D) None of the above.

(iv) Let $\mathcal{A}$ be a comparison-based sorting algorithm for sorting an array of length n, and let T be the corresponding decision tree (as per our proof in Lecture 6). Which of the following are necessarily true about T?

(A) T has at least $n!$ leaves

(B) The depth of T is $O(\log n)$

(C) The depth of T is $\Omega(n \log n)$

(D) Running $\mathcal{A}$ corresponds to a path from the root of T to a leaf

(E) Running $\mathcal{A}$ corresponds to an internal node of T

(F) None of the above.

(v) Imagine we want to sort a list of numbers in **descending** order using RadixSort. Which of the following modifications (made *individually*) should we make to the algorithm shown in class? Select all that apply.

(Note: "MSD" stands for "Most Significant Digit" and "LSD" stands for "Least Significant Digit".)

(A) We bucket the numbers in MSD-to-LSD order instead of LSD-to-MSD.

(B) We no longer need to pad the numbers with leading zeros.

(C) We "unload" each bucket in last-in-first-out order, instead of first-in-first-out.

(D) In each iteration of CountingSort, we traverse the buckets in descending order, starting with the n^{th} bucket and ending with the 0^{th} bucket.

(E) None of the above.

(vi) Let T be a red-black tree that contains n items. Which must be true about T?

(A) T has $O(\log n)$ levels.

(B) T has $\Omega(\log n)$ levels.

(C) T contains $O(\log n)$ black nodes.

(D) Inserting an item into T takes time $O(\log n)$.

(E) None of the above.

2. **(10 pt.) [True or false?]** For each of the parts below, say whether the statement is true or false, and explain why.

   [**We are expecting:** *For each, either* TRUE *or* FALSE. *If it is false, give a counter-example. If it is true, give a short explanation why (a sentence or two). You don't need to give a formal proof, and you do not need to appeal to the formal definition of big-Oh.*]

   (a) If $f(n) = O(g(n))$, then $f(n) = \Omega(g(n))$.

   (b) If $f(n) = 4^{\log_2(n)}$, then $f(n) = \Omega(n)$.

   (c) If $f(n) = 2^{\log_2(\log_2(\log_2(n)))}$, then $f(n) = \Omega(n)$.

   (d) If $f(n) = O(g(n))$, then $2^{f(n)} = O(2^{g(n)})$.

3. **(20 pt.) [Recurrence Relations!]** For each of the following recurrence relations for $T(n)$, give the best big-Oh bound that you can. Simplify your expressions so that they are of the form $O(n^{\text{something}})$. You may ignore floors and ceilings in your answers.

   [**We are expecting:** *For each part, a statement of the form* $T(n) = O(n^{something})$*. No justification is required if your answer is correct, but a short informal justification may be considered for partial credit.*]

   (a) $T(n) = 5T(\lfloor n/3 \rfloor) + O(n^2)$ for $n > 0$, with $T(0) = 1$.

   (b) $T(n) = 10T(\lfloor n/3 \rfloor) + O(n^2)$ for $n > 0$, with $T(0) = 1$.

   (c) $T(n) = T(\lfloor n/5 \rfloor) + T(\lfloor n/10 \rfloor) + n^2$ for $n > 10$ with $T(n) = 1$ for $0 \leq n \leq 10$.

   (d) $T(n) = T(\lfloor \log_2(n) \rfloor) + n^{10}$ for $n > 2$ with $T(n) = 1$ for $0 \leq n \leq 2$.

4. **(15 pt.) [Can it be done?]**

For each of the following tasks, either explain briefly how to do it, or prove that it cannot be done. In all of the cases, running times are to be interpreted using worst-case analysis. We have done the first two for you to give you a sense of what we are looking for.

[**We are expecting:** *For each part, either a description of an algorithm, or a short proof of impossibility. You may appeal to any result/algorithm from class, but you must clearly state how you are using it.*]

(a) **(0 pt.) (Example)** Find the maximum of a (not necessarily sorted) array A in time $O(n \log n)$.

Use `mergeSort` on the array A; then return the last element of what `mergeSort` returns.

(b) **(0 pt.) (Example)** Find the maximum of a (not necessarily sorted) array A in time $O(1)$.

This cannot be done. Since any of the n entries of the array could be the maximum, we need to at least look at every element in the array, which takes time $\Omega(n)$.

(c) Given an array A of length n which contains d-digit integers (base 10) for $d = 3\log_{10}(n)$, sort A in time $O(n)$.

(d) Given an array A of length n which contains arbitrary comparable elements, sort A in time $O(n)$. (Here, "arbitrary comparable elements" means that you cannot interact with the values of the elements other than by comparing them to each other.)

(e) Design a data structure that holds n elements and that supports INSERT/DELETE/SEARCH as well as MAX/MIN, each in time $O(\log n)$.

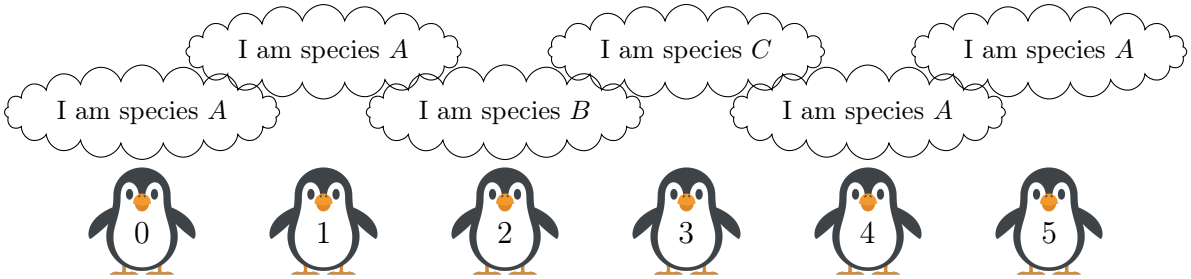
5. **(20 pt.) [Algorithm Design.]**

On an island, there are n penguins of many different species. The differences between the species are very subtle, so without help you can't tell the penguins apart at all. Fortunately, you have an expert with you, and she can tell you whether or not two penguins belong to the same species. More precisely, she can answer queries of the form:

$$\texttt{isTheSame(penguin1, penguin2)} = \begin{cases} \textbf{True} & \text{if } \texttt{penguin1} \text{ and } \texttt{penguin2} \text{ belong to the same species} \\ \textbf{False} & \text{if } \texttt{penguin1} \text{ and } \texttt{penguin2} \text{ belong to different species} \end{cases}$$

The only way you can get any information about the penguins is by running `isTheSame`. You cannot ask them what species they are, or compare them in any other way.

The expert assures you that one species of penguin is in the majority. That is, there are *strictly greater* than $n/2$ penguins of that species. Your goal is to return a single member of that majority species. For example, if the population looked like this:



then species A is in the majority, and your algorithm should return any one of Penguins 0, 1, 4, or 5.

If there is no species with a strict majority, your algorithm may return whatever it wants.

Questions start on next page!

(a) **(10 pt.)**[1] Design a deterministic *divide-and-conquer* algorithm that uses $O(n \log(n))$ calls to `isTheSame` and returns a penguin belonging to the majority species. You may assume that n is a power of 2 if it is helpful.

[**We are expecting:** *Pseudocode (which calls* `isTheSame`*)* ***AND*** *a clear English description of what your algorithm is doing.*]

More parts on next page!

[1]Note: On the actual exam, we will not expect you to do creative algorithm design that requires "aha!" moments in a timed setting (except maybe for bonus points). There will be algorithm design question(s), but they will be variants of things you have seen before in class or on HW, so that if you have studied enough, you won't need (what we consider to be) an unreasonable "aha!" moment. This problem requires an "aha!" moment, so it wouldn't appear on the actual exam — but it is a good chance to get more practice with algorithm design! (Which will help on the real exam, since it will help you remember and recreate the aha! moments from HW and lecture.)

(b) **(5 pt.)** Explain why your algorithm calls `isTheSame` $O(n \log(n))$ times.
[**We are expecting:** *A short justification of the number of calls to* `isTheSame`*. You may invoke the Master Theorem if it applies.*]

(c) **(0 pt.)** [This is not required, but since you're taking a practice test, at this point it would be good practice to formally prove that your algorithm is correct!]

More parts on next page!

(d) **(5 pt.)** Give a *randomized* algorithm that *always* (with probability 1) returns a majority penguin, so that, for any input, the *expected* number of calls to `isTheSame` $O(n)$.

[**We are expecting:** *Pseudocode AND a clear English description, as well as an informal explanation (a few sentences) of why the expected number of calls to* `isTheSame` *is* $O(n)$.]

(e) **(NOT REQUIRED. 5 BONUS pt.)** Can you come up with a deterministic algorithm that is asymptotically better than $O(n \log(n))$? Either give a deterministic algorithm with asymptotically fewer calls to `isTheSame`, or else prove that no such algorithm exists. (Or, if you think that your algorithm from part (a) already does better, just write that :)).

[**We are expecting:** *Nothing. This part is not required. To get the bonus point, you should either: (a) give pseudocode AND a clear English description of what it does, along with a short informal justification of why it uses asymtotically fewer calls to* `isTheSame`*; or (b) a rigorous proof of why you cannot do better.*]

6. **(20 pt.) [Proofs!]** Suppose that $T(n) = T(\lfloor n/2 \rfloor) + T(\lfloor n/4 \rfloor) + n$ for all $n \geq 4$, and $T(0) = T(1) = T(2) = T(3) = 1$. Prove by induction that $T(n) = O(n)$.

   [**We are expecting:** *A rigorous proof by induction. Make sure you clearly state your inductive step, base case, inductive step, and conclusion.*]

This is the end of the exam! You can use this page for extra work on any problem. **Keep this page attached** to the exam packet (whether or not you use it), and if you want extra work on this page to be graded, clearly label which question your extra work is for.

This page is for extra work on any problem. **Keep this page attached** to the exam packet (whether or not you use it), and if you want extra work on this page to be graded, clearly label which question your extra work is for.

This page is for extra work on any problem. **Keep this page attached** to the exam packet (whether or not you use it), and if you want extra work on this page to be graded, clearly label which question your extra work is for.

This page is for extra work on any problem. **Keep this page attached** to the exam packet (whether or not you use it), and if you want extra work on this page to be graded, clearly label which question your extra work is for.