CS 381, Practice Midterm Problems, Spring 2021

This booklet contains previous exam problems. It does not correspond to the length of a 2-hour exam. We suggest you to work out 3-4 problems at a time without consulting any resources/study group members. Cheat sheet is okay.

No solutions will be posted for the problems in this booklet. For questions, come to office hours or ask on Piazza.

Students taking the online version of the exam should practice entering answers as well as uploading a file representing the answer to a problem.

Printing the exam, writing the answers and upload or emailing the resulting file is not an option if you are taking the online exam.

1. For each of the following three pairs of functions $f(n)$ and $g(n)$ determine whether or not $f(n) = O(g(n))$. All logs are base two. Show your work for each pair.
   (a) $f(n) = 2^{\log(2^{2n})}, g(n) = 2^{2^n}$
   (b) $f(n) = n^2 \log^{19} n$, $g(n) = n^{1.7} \log^{20} n$
   (c) $f(n) = 2^{n+1}$, $g(n) = n^{n-1}$

2. Order the following functions according to their asymptotic growth rate. Indicate which functions belong to the same complexity class.
   $4n \log n$, $2^n \log n$, $n^2 + 8n \log n$, 2^n, $(n+4)(n-6)$, $n!$, n^{2^6}, $\sqrt{n}$, $2^{n/2}$, 2^{n^6}, $(n-2)!$

3. How many times is function F called in the code segment below? Give a tight bound in terms of n, preferably using big-Θ notation, and a brief explanation.

```
for i = 1 to n do
    m = 1
    for j = 1 to i do
        m = m * j^2
    end for
    k = 2
    while k < m do
        F(k)
        k = k^2
    end while
end for
```

4. Assume n is an even positive integer. How many inversions (in big-Θ notation) are there in the permutation

$$n^2, (n-1)^2, (n-2)^2, \ldots, (\frac{n}{2})^2, 1^2, 2^2, 3^2, \ldots, (\frac{n}{2}-1)^2.$$

Explain in at most three lines.

5. Which of the following statements is true? Explain your answer.
   (a) $n^3 = O(n^4/16)$
   (b) $64\sqrt{n} + \frac{18n}{\log n} + 32n \log\log n = O(n)$
   (c) $n^3 = \Omega((n \log n)^3)$
   (d) $(\frac{n}{2})! = \Theta((\frac{n}{4})!)$

6. Given is an array A of size n containing integers in arbitrary order. The integers are between 0 and $2n$. Describe and analyze an efficient algorithm to determine whether there exist two indices i and j, $i \neq j$, with $A[i] = A[j]$.

7. A is an array of size n with elements in arbitrary order. Describe an efficient algorithm to determine whether a given element x is the 5-th smallest element in A?

8. Given an array A that contains n integers, assume that you have access to a special function F that returns the maximum of up to $\lceil n^{1/4} \rceil$ numbers in constant time. Here is how F works: it takes two inputs, array indices i and j with $i \leq j$. Then, $F(i, j)$ returns the maximum element of $\{A[i], A[i+1], \ldots, A[j]\}$. $F(i, j)$ takes constant time if $j - i \leq \lceil n^{1/4} \rceil$; otherwise, F does **not** work.

   Using this function F, describe an efficient algorithm to find the maximum entry in A. Describe, using big-Θ notation, the number of times that the function F is called. To get full credit, the running time of your algorithm should be asymptotically **better** than $\mathcal{O}(n)$. (**Reminder**: You do not need to construct F. For this problem simply assume that it exists.)

9. Suppose that $T(n) = 7T(n/2) + T(n/5) + n^3$ with $T(1), T(2), T(3), T(4) \leq 10$.
   Use induction to prove that $T(n) \leq 10n^3$ for every $n \geq 1$.

10. Consider the recurrence relation $T(n) = 3T(n-1) + 2$ with $T(1) = 1$.
    Use induction to prove that $T(n) = O(3^n)$ for every $n \geq 1$.

11. Assume A is an array of size n containing integers in arbitrary order, A_s is an array of size n containing integers in sorted order, and A_h is an array of size n containing integers arranged in a min-heap. Give the running times (in big-O notation) for the specified operations on a given element x. Give a brief explanation for each entry in the table.

	determine whether x is in the array	determine whether x occurs at least $3n/4$ times	determine whether x is smaller than the smallest element in the array
A (not sorted)			
A_s (sorted)			
A_h (min-heap)			

12. You are given $\sqrt{n}$ lists, each containing $\sqrt{n}$ elements in sorted order. How fast can one merge the $\sqrt{n}$ sorted lists into one sorted sequence? State the time bound and explain your answer.

13. Describe a linear time algorithm for finding the sum of the $2 \log n$ smallest elements in an unsorted array of size n.

14. Design a linear-time algorithm that on input an array A containing n distinct integers in arbitrary order, and integers i, j satisfying $0 < i < j \leq n$, outputs all the elements of the array with ranks from i to j. For example, given $A = [41, 20, 10, 34, 58, 98, 64, 72]$, and $i = 3, j = 5$, the numbers output should be 34, 41, 58, since the 3rd smallest element element is 34, and the 5th smallest element is 58. Briefly argue your algorithm's correctness and running time.

15. The Nonadjacent Pair Selection (NPS) problem is defined as follows. Given is an array A of size n containing positive integers in arbitrary order. You need to select adjacent pairs of array locations so that (a) the sum of the elements in the selected pairs is a maximum and (b) no two array elements in different pairs are adjacent. A single array location cannot be selected.

    For example, for A = [3, 5, 1, 10, 4], selecting 3+5 and 10+4 achieves a total of 22. Any other feasible solutions select only one pair. Selecting 5 and 10+4 is not allowed.

    (a) State and explain DP recurrences solving the NPS problem.
    Hint: Consider two quantities: OPT(i,N), the value of the optimum solution on elements A[1] to $A[i]$ when $A[i]$ is **not** included in a pair; OPT(i,R) the value of the optimum solution on elements A[1] to $A[i]$ when $A[i]$ is the **right element** in a pair.

    (b) State and explain the time complexity for computing the recurrences given above.

    (c) Using the OPT values computed, what is the maximum value (i.e., sum elements in all selected pairs) for array A.