Team Contest
September 27–30, 2017

Rules

You have been assigned a team of 4–5 students to solve the following problems and compete with other teams to present solutions. The team contest will have seven sessions, each with roughly 20 students. In each session, teams will take turns selecting a member who has not presented yet and selecting a problem that has not been presented yet. When it is your turn to present a problem:

- Read your chosen problem aloud to the class
- You may draw a diagram on the whiteboard, or copy out some equations or brief pseudocode, but you may not write any text. You must now put away any notes you have.
- As soon as you start speaking, you will have 3 minutes to present the solution to your problem. As usual in this course, you must justify what you say. An algorithm without a proof will be worth few points.
- Our course staff will then award you points, out of 10.
- Go back to your seat and enjoy the rest of the show!

The team contest will count as one regular homework. Half the points are for showing up, because watching your peers present is a valuable experience. The remaining half of the points will be the total score your team earns (multiplied by 1.25 if you are on a team of 4).

Problems

1. Prove via divide-and-conquer that, given a $2^k \times 2^k$ board of squares with the top right square removed, you can exactly cover the entire board with L-shaped pieces (each covering 3 squares).

2. Heartwarming – Florida man discovers how to quickly sort pancakes! Given a stack of pancakes, he can stick his spatula in between any two pancakes, and flip the stack of pancakes above the spatula upside-down. Find an algorithm to sort $n$ pancakes from largest to smallest, using at most $2n$ flips.

3. Given $c$ different types of coins with positive integer denominations $d_1, \ldots, d_c$, find an $O(cn)$ algorithm to find the smallest number of coins needed to make $n$ cents. You may assume that there is at least one way to make each value with the coins.

4. Given $c$ different coins with positive integer denominations $d_1, \ldots, d_c$, where you only have one of each coin, find an $O(cn)$ algorithm to determine if it is possible to make $n$ cents.

5. Given $c$ different coins worth a total of $n$ cents, with positive integer denominations $d_1, \ldots, d_c$, and weights $w_1, \ldots, w_c$ (that might not be integers!), suppose you can carry at most $b$ total
weight in your bag. Find an $O(cn)$ algorithm to find the most valuable subset of coins that you can carry in your bag.

6. Matlab has a function \texttt{unique} that, given $n$ numbers, returns a list of the unique elements, namely, it returns the same values as are in the input but with no repetitions. Find an $O(n \log n)$ algorithm for this task.

7. Prove that if a value $v$ appears in a list $L$ that is sorted in ascending order, then the following \texttt{Binary-Search} algorithm will find it:

\begin{verbatim}
Binary-Search(L)
1 low = 1
2 high = LENGTH(L)
3 loop
4   mid = \left\lfloor \frac{low + high}{2} \right\rfloor
5   if L[mid] > v then high = mid - 1
6 elseif L[mid] < v then low = mid + 1
7 else return mid
\end{verbatim}

8. Prove the correctness of the \texttt{Quicksort} algorithm:

\begin{verbatim}
Quicksort(list)
1 Let n be the total length of the list
2 if n \leq 1
3   return
4 else
5   Choose a random element $x$ from list
6   Let $c$ be the number of elements $< x$, and let $d$ be the number of elements $\leq x$ in the list
7   Reorder the list so that numbers $< x$ are to the left of numbers equal to $x$, which are to the left of the other numbers
8   Quicksort(list(1...c))
9   Quicksort(list(d+1...n))
\end{verbatim}

9. Radix Sort I. We refer to the $d$-th bit of a binary number by zero-indexing from the least significant bit, e.g., $d = 0$ is referring to the rightmost bit whereas $d = 3$ is referring to the 4th bit from the right. Prove the correctness of the following sort algorithm when it is run with input the list of binary numbers and the maximum number of bits of all numbers in the list:

\begin{verbatim}
Radix-Sort-1(list, d)
1 Reorder the list so that numbers with $d$-th bit 0 come before numbers with $d$-th bit 1
2 if $d > 0$
3   Let $c$ be the number of elements in the list with $d$-th bit 0
4   Let $n$ be the total length of the list
5   Radix-Sort-1(list(1...c), d - 1)
6   Radix-Sort-1(list(c+1...n), d - 1)
\end{verbatim}
10. Radix Sort II. A sorting procedure is called “stable” if it preserves the order of elements that are indistinguishable to the comparison function. We refer to the \( d \)-th bit of a binary number by zero-indexing from the least significant bit (as in Radix Sort I, above). Prove the correctness of the following sorting algorithm:

\[
\text{Radix-Sort-2}(\text{list})
\]
\[
\begin{align*}
1 & \text{ for } i = 0 \text{ to maximum number of bits of all numbers in list} \\
2 & \text{ Perform a stable sort on the } i\text{-th bits of the numbers }
\end{align*}
\]

11. An Old Friend From High School has decided to take her vocal expertise and pursue her dream of becoming a pop superstar. On her first tour, she knows that if she performs a concert on day \( i \) she will make \( d(i) \) dollars. While she would like to perform for all \( n \) days of her tour, she must give her voice a rest, and she can never sing for \( k \) days in a row. Design an \( O(kn) \) algorithm for the Old Friend From High School to compute the best days to play on her tour.

12. Not to be outdone by Old Friend From High School, Angry Little League Coach decides to pursue his dreams of being a country superstar. On his tour across the globe, he knows that if he plays a concert on day \( i \) he will make \( d(i) \) dollars. He also wants time to play T-ball at each stop on his tour, so he wants a break from performing for at least \( k \) days after every time he sings. Design an \( O(kn) \) algorithm for the Angry Little League Coach to compute the best days to perform on his tour.

13. Prove that the following algorithm always terminates and correctly sorts its input \( L \):

\[
\text{Mysterious-Sort}(L)
\]
\[
\begin{align*}
1 & \text{ loop} \\
2 & \text{ if } L \text{ is sorted then return } L \\
3 & \text{ else} \\
4 & \text{ Pick an arbitrary index } i \text{ such that } L[i] > L[i + 1]. \\
5 & \text{ Swap } L[i] \text{ and } L[i + 1]
\end{align*}
\]

14. *Turn down for what*: After completing his original mission of obliterating casual dining chains, Millenial Killing Applebee’s leaves suburbia, and he’s begun his (rather futile) career as a DJ. He is DJing his first big party, and he needs to figure out which songs to play. Each song \( i \) from his collection of \( n \) songs has a speed \( s_i \) and a loudness \( \ell_i \). In order to keep the party lit, each song needs to be both faster and louder than the previous song. Design an algorithm that, in \( O(n^2) \) time will find the longest possible sequence of songs he can play.

15. Given \( k \) sorted lists, each of length \( n \), design an algorithm to merge them into a single sorted list in time \( O(kn \log k) \).

16. In a seminar at Brown, 25 students are paying attention and 25 students are looking at memes on their phones. They all sit around a circular table (50 people total). An attendee is “memed” if the people to her left and right are both looking at memes (whether or not she looks at memes). Show that the following algorithm will always find a memed attendee:

\[
\text{Find-Memed-Attendee(AttendeesAtTable)}
\]
\[
\begin{align*}
1 & \text{ for each person } x \in \text{AttendeesAtTable} \\
2 & \text{ if the person to } x\text{’s left and the person to } x\text{’s right are both looking at memes} \\
3 & \text{ return } x
\end{align*}
\]
17. **Dermatologists hate her!** Local Area Mom discovered natural remedies to common skin care problems, and she now has a thriving business. In order to get her products to her loyal customers, she must fit her remedies into boxes for shipping. She has $n_1$ Detoxifying Cleansers that each weigh $w_1$, $n_2$ Revitalizing Herbs that each weigh $w_2$, and $n_3$ Soothing Teas that each weigh $w_3$, where $0 \leq w_1, w_2, w_3 \leq 1$. She wants to ship them using as few boxes as possible (to be as eco-friendly as possible), given that each box can hold total weight at most 1. Find an algorithm to compute the number of boxes she needs, in time $O(n_1 n_2 n_3)$.

18. You start at the top left corner of an $n \times n$ square of numbers, which can be positive or negative, and must end at the bottom right. In each move you can go left, right, or down (but never up!), and you can never revisit a square you have already visited. Find an $O(n^2)$ algorithm to find the maximum sum of numbers you can visit.

19. A list $L$ that is not sorted will have some pairs of indices $i < j$ such that $L[i] > L[j]$. Find an $O(n \log n)$ algorithm to count the number of such pairs.

20. Given two sorted lists of numbers of total length $n$, find an $O(\log n)$ time algorithm that will return the median of the entire set of $n$ numbers in the two lists. You may assume that $n$ is odd.

21. Given the algorithm from homework 1 to find the longest common subsequence between two strings, design an improved algorithm that uses only $O(n \log n)$ memory, while still using time $O(n^2)$. Your algorithm must compute the common subsequence, not just its length. You may assume what you showed in homework 1—that the algorithm from the homework correctly fills out a table whose $(i, j)$th entry represents the longest common subsequence between the first $i$ letters of the first string and the first $j$ letters of the second.