CS 161
Summer 2017

Problem Set 1
Due: July 7, 2017, 11:59 p.m.

---

1. **What do you want from this course?** (1 point) What skills do you hope to learn, what topics do you think we'll cover that will stick with you five, or ten years down the road? Write *at least three sentences* describing how you expect to benefit from taking this class. The reason for this question is twofold. First, it will help us understand what you want. Second, keep your answer to this question in mind throughout the quarter as motivation if the going gets tough!

2. **New friends.** (5 points) Each of n users spends some time on a social media site. For each $i = 1, \ldots, n$, user i enters the site at time a_i and leaves at time $b_i \geq a_i$. You are interested in the question: how many distinct pairs of users are ever on the site at the same time? (Here, the pair (i, j) is the same as the pair (j, i)).

   Example: Suppose there are 5 users with the following entering and leaving times:

User	Enter time	Leave time
1	1	4
2	2	5
3	7	8
4	9	10
5	6	10

   Then, the number of distinct pairs of users who are on the site at the same time is three: these pairs are $(1, 2)$, $(4, 5)$, $(3, 5)$. (Drawing the intervals on a number line may make this easier to see).

   (a) (1 point) Given input $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$ as above in no particular order (i.e., not sorted in any way), describe a straightforward algorithm that takes $\Theta(n^2)$-time to compute the number of pairs of users who are ever on the site at the same time, and explain why it takes $\Theta(n^2)$-time.

   [We are expecting pseudocode and a brief justification for its runtime.]

   (b) (4 points) Give an $\Theta(n \log(n))$-time algorithm to do the same task and analyze its running time. (**Hint:** consider sorting relevant events by time).

   [We are expecting pseudocode and a brief justification for its runtime.]

3. **Proof of correctness.** (6 points) Consider the following algorithm that is supposed to sort a list of integers. Provide a proof that this algorithm is correct.

   [We are expecting a rigorous proof.]

Algorithm 1: `sort`

```
Input: Unsorted list A = [a_1, ..., a_n] of n items
Output: Sorted list A' = [a'_1, ..., a'_n] of n items
for i = 0, i < n − 1, i++ do
    // Find the minimum element
    min_index = i
    for j = i + 1, j < n, j++ do
        if A[j] < A[min_index] then
            min_index = j
    // Swap the minimum element with the first element
    swap(A, i, min_index)
return A
```

4. **n-naught not needed.** (3 points) Suppose that $T(n) = O(n^d)$, and that $T(n)$ is never equal to ∞. Prove rigorously that there exists a c so that $0 \leq T(n) \leq c \cdot n^d$ for all $n \geq 1$. That is, the definition of $O(\cdot)$ holds with $n_0 = 1$.

   [We are expecting a brief, but convincing proof.]

5. **Fun with recurrences.** (6 points)

   Solve the following recurrence relations; i.e. express each one as $T(n) = O(f(n))$ for the tightest possible function $f(n)$, and give a short justification. Be aware that some parts might be slightly more involved than others. Unless otherwise stated, assume $T(1) = 1$.

   [To see the level of detail expected, we have worked out the first one for you.]

   (z) $T(n) = 6T(n/6) + 1$. We apply the master theorem with $a = b = 6$ and with $d = 0$. We have $a > b^d$, and so the running time is $O(n^{\log_6(6)}) = O(n)$.

   (a) (0.5 points) $T(n) = 2T(n/2) + 3n$

   (b) (0.5 points) $T(n) = 3T(n/4) + \sqrt{n}$

   (c) (0.5 points) $T(n) = 7T(n/2) + \Theta(n^3)$

   (d) (2 points) $T(n) = 4T(n/2) + n^2 \log n$

   (e) (0.5 points) $T(n) = 2T(n/3) + n^c$, where $c \geq 1$ is a constant (that is, it doesn't depend on n).

   (f) (2 points) $T(n) = 2T(\sqrt{n}) + 1$, where $T(2) = 1$

6. **Why groups of 5?** (6 points) In the `select_k` algorithm from class, in order to find a pivot, we decompose our array of length n into $m = \lceil n/5 \rceil$ blocks of at most length 5. Why 5? In this question, we explore this decision.

   (a) (0.5 points) Prove that the recursive call inside of `select_k_with_groups_of_3` is on an array of size at most $2n/3 + 2$. Unlike in lecture, here include the values within the same group as the median of medians as elements that are guaranteed to be greater or less than it.

   [We are expecting a brief, but convincing algebraic proof.]

   (b) (0.5 points) Write a recurrence relation for the runtime of `select_k_with_groups_of_3`.

   [We are expecting a one-line answer with your recurrence relation.]

   (c) (2 points) Is `select_k_with_groups_of_3` $O(n)$? Justify your answer.

   [We are expecting a convincing argument, such as analyzing a tree, unraveling the recurrence relation to get a summation, or attempting the substitution method.]

   (d) (0.5 points) Prove that the recursive call inside of `select_k_with_groups_of_7` is on an array of size at most $5n/7 + 4$. Make the same assumptions as in part (a).

   [We are expecting the same as part (a).]

   (e) (0.5 points) Write a recurrence relation for the runtime of `select_k_with_groups_of_7`.

   [We are expecting the same as part (b).]

   (f) (2 points) Is `select_k_with_groups_of_7` $O(n)$? Justify your answer.

   [We are expecting the same as part (c).]

7. **k-order statistic of compressed dataset.** (9 points) Suppose you have a collection of data points where there are a huge number of copies of each data point. For example, you might have a data set:

```
A, B, C, B, A, C, B, A,
A, B, A, C, A, A, C, B,
D, A, A, A, D, B, E, A,
A, F, A, B, A, A, A, B
```

   Here, there are 32 data points, but there are only six distinct values. Rather than storing the data as above, suppose that you store the data as a list of tuples of the form $(value, frequency)$. For example, the above data might be stored as

```
(A, 16), (B, 8), (C, 4), (D, 2), (E, 1), (F, 1)
```

   That is, there are 16 `A`'s, eight `B`'s, four `C`'s, two `D`'s, one `E`, and one `F`.

Suppose that you have a data set represented as m of these tuples, where the tuples are stored in no particular order. Design an $O(m)$-time algorithm for computing the kth order statistic of the data set. Note that m is the number of tuples rather than the total number of elements n in the uncompressed data set, so your algorithm's runtime should not depend asymptotically on n.

(a) (4 points) Describe your algorithm.
[We are expecting pseudocode.]

(b) (4 points) Provide a proof that this algorithm is correct.
[We are expecting a rigorous proof.]

(c) (1 point) Prove that your algorithm runs in $O(m)$.
[We are expecting a brief justification.]

8. **How is the course so far?** (0.5 points extra credit)

Please complete the poll at `https://goo.gl/forms/GzB9KO7j2NyddV3x2`.