Due date: Tuesday, January 31 at 10:30 AM. You will need to submit this via GradeScope. Late problem sets are not accepted.

In CompSci161, your response to each numbered question must be contained within a single piece of paper. Each numbered question must be responded to on a separate page. When you submit to GradeScope, you will need to inform the system which page of your scanned PDF contains the response you want graded. On occasion, we might request you to tag two (or more) parts, even though we know the two parts will be on the same page; when that happens, it usually means we are going to grade the parts separately. Failure to follow these directions will be treated as non-submission for the question(s) affected.

Please review the syllabus and course reference for the expectations of assignments in this class. Remember that problem sets are not online treasure hunts. You are welcome to discuss matters with classmates, but remember the Kenny Loggins rule. Remember that you may not seek help from any source where not all respondents are subject to UC Irvine's academic honesty policy.

1. Suppose you have two disjoint sets A and B with each with n comparable elements. More formally:

   - $n = |A| = |B| \geq 2$
   - $A \cap B = \emptyset$

   You can only perform the following operations on these sets at the costs of their stated time complexities:

   - size(S: Set) $\rightarrow$ Integer
     Description: Returns $|S|$
     Time Complexity: $\Theta(1)$
   - middleElements(S: Set) $\rightarrow$ (Element, Element)
     Description: Returns the two median elements of the set (same element if $|S|$ is odd).
     Time Complexity: $\Theta(1)$
   - discardLeft(S: Set) $\rightarrow$ Set
     Description: Returns a new set where every element less than middleElements(S)[0] is removed
     Time Complexity: $\Theta(1)$
   - discardRight(S: Set) $\rightarrow$ Set
     Description: Returns a new set where every element greater than middleElements(S)[1] is removed
     Time Complexity: $\Theta(1)$
   - bruteForce(A: Set, B: Set) $\rightarrow$ (Element, Element)
     Description: Finds the two median elements of $A \cup B$
     Time Complexity: $\Theta(|A \cup B|)$

   Write or describe an $\mathcal{O}(\log n)$ time algorithm to find the two median elements of $A \cup B$. Do not assume anything about the datatype Element except that it is comparable ($<, =, >$ are valid operations). Briefly justify why your algorithm takes $\mathcal{O}(\log n)$ time.

   There are two additional questions on the next digital page.

2. We've all heard of square root, but what about rectangle root? I define the k-rectangle root r of n to be:

$$(r)(r+k) = n$$

For example, the 3-rectangle root of 130 is 10 because $10(10+3) = 130$.

Write or describe an $\mathcal{O}(\log n)$ time algorithm to find the $\lfloor r \rfloor$, the floored k-rectangle root of n, given that $\lfloor r \rfloor, k, n$ are all natural numbers. Do not use a square root operation as part of your solution.

3. Recently, I was looking back at my career as a professor and I was thinking of some of my former students, each of whom has already graduated and moved onto a great career, as I hope all of you will soon (I will miss you though). I was wondering which pair of students were *both* students concurrently for the longest period of time. To answer this question, I had to gather some information. From the registrar, I was able to get very precise enrollment and graduation dates for each. The registrar, for purposes of this question anyway, defines a student's **enrollment** time as the precise millisecond that the student's intent to register is recorded, and defines the student's **graduation** time as the precise millisecond that the student's diploma is printed. The time from enrollment to graduation is an individual's time as a student. Note that the definitions for enrollment and graduation in this problem mean that no two students have the exact same enrollment time, nor do any two have the exact same graduation time, even if they started the same quarter and ended the same quarter.

Suppose you are given a list of n students, each with the enrollment and graduation times as described above. You may assume the list is sorted by enrollment date if you prefer. Design an efficient **divide and conquer** algorithm to determine which two individuals' time as students has the maximum overlap.

More formally, you want to find the pair of indices $i \neq j$ such that the following value is maximized:

$$\min(A[i].\text{graduation}, A[j].\text{graduation}) - \max(A[i].\text{enrollment}, A[j].\text{enrollment})$$

In a few sentences, justify the correctness of your algorithm. Also, set up and solve a recurrence to justify the running time of your algorithm.

No credit will be given for an algorithm with running time $\Omega(n^2)$.

Do not assume people graduate in the order they register. Do not assume everyone graduates in four years, or 12 quarters, or 45 months, or anything like that.