CSE 21 Spring 2022
Homework 4
Due date: Thurs April 28, 2022 at 11:59pm

---

INSTRUCTIONS

This homework should be done in **groups** of one to four students, without assistance from anyone besides the instructional staff and your group members. Homework must be submitted through Gradescope by a **single representative** of your group and received by **11:59pm** on the due date. There are no exceptions to this rule.

You will be able to look at your scanned work before submitting it. You must **type** your solutions. (hand-drawn diagrams are okay.) Your group representative can resubmit your assignment as many times as you like before the deadline. Only the most recent submission will be graded.

Students should consult their textbook, class notes, lecture slides, podcasts, group members, instructors, TAs, and tutors when they need help with homework. You may ask questions about the homework in office hours, but questions on Piazza should be private, visible only to instructors.
This assignment will be graded for not only the *correctness* of your answers, but on your ability to present your ideas clearly and logically. You should explain or justify, present clearly how you arrived at your conclusions and justify the correctness of your answers with mathematically sound reasoning (unless explicitly told not to). Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to **convince the reader** that your results and methods are sound.
KEY CONCEPTS Sorting, searching, asymptotic notation, time analysis.

---

1. (20 points)

   Consider these two algorithms that find the indices of BOTH the minimum and maximum value in an even length array.

   (a) $MinMax1(\ a[1], \ldots, a[n]$, array of distinct integers with $n$ an even number.)

```
1. $i = 2$
2. $m = 1$
3. $M = 1$
4. **while** $i \leq n$:
5.    **if** $a[i] < a[m]$ then
6.         $m = i$
7.    **if** $a[i] > a[M]$ then
8.         $M = i$
9.    $i = i + 1$
10. **return** $(m, M)$
```

   Show that for a list of size $n$ the number of *comparisons between list elements* is equal to $2n - 2$. (The *comparisons between list elements* occur in the conditions of the if statements (lines 5,7).)

   (b) $MinMax2(\ a[1], \ldots, a[n]$, array of distinct integers with $n$ an even number.)

```
1. $i = 1$
2. $m = 1$
3. $M = 1$
4. **while** $i \leq n$:
5.    **if** $a[i] < a[i+1]$:
6.        **if** $a[i] < a[m]$ then
7.            $m = i$
8.        **if** $a[i+1] > a[M]$ then
9.            $M = i + 1$
10.   **else**:
11.       **if** $a[i+1] < a[m]$ then
12.           $m = i + 1$
13.       **if** $a[i] > a[M]$ then
14.           $M = i$
15.   $i = i + 2$
16. **return** $(m, M)$
```

   Show that for a list of size $n$ the number of *comparisons between list elements* is equal to $3n/2$. (The *comparisons between list elements* occur in the conditions of the if statements (lines 5,6,8,11,13).)

   (c) Consider the sorting algorithm *MinMax1Sort* on a list of distinct integers of an even length array.

   $MinMax1Sort(\ a[1], \ldots, a[n]$, array of distinct integers with $n$ an even number.)

```
1. $i = 1$
2. **while** $i \leq n/2$:
3.    $(m, M) = MinMax1(a[i], \ldots, a[n - i + 1])$
4.    **if** $M == i$:
5.        $M = m$
6.    swap $a[m]$ and $a[i]$
7.    swap $a[M]$ and $a[n - i + 1]$
8.    $i = i + 1$
```

   For a list of size $n$, count the number of *comparisons between list elements* (in terms of $n$.)

(d) Consider the sorting algorithm *MinMax2Sort* on a list of distinct integers of an even length array.

```
$MinMax2Sort($ $a[1], \ldots, a[n]$, array of distinct integers with $n$ an even number.)
1. $i = 1$
2. while $i \leq n/2$:
3.     $(m, M) = MinMax2(a[i], \ldots, a[n-i+1])$
4.     if $M == i$:
5.         $M = m$
6.     swap $a[m]$ and $a[i]$
7.     swap $a[M]$ and $a[n-i+1]$
8.     $i = i + 1$
```

For a list of size $n$, count the number of *comparisons between list elements* (in terms of $n$.)

2. (20 points) Let $f : \mathbb{R} \to \mathbb{R}$ be a continuous increasing function. Then we can use a version of binary search to solve for the "root" of the function $f$ (value $x$ such that $f(x) = 0$.) This algorithm will take the input $f$, starting value $n$ such that $0 < r < n$ where $r$ is the root of the function and an error term $e$.

The function will stop when it finds a value $v$ such that $|r - v| \leq e$.

```
$FunctionSolver($ $f, n, e)$
1. $lo = 0$
2. $hi = n$
3. $v = (hi + lo)/2$
4. while $hi - lo > 2e$:
5.     if $f(v) = 0$:
6.         return $v$
7.     if $f(v) < 0$:
8.         $lo = v$
9.     if $f(v) > 0$:
10.        $hi = v$
11.    $v = (hi + lo)/2$
12. return $v$
```

(a) Trace through the algorithm on the input $FunctionSolver(e^x - 1/x, 2, 0.0625)$.

(b) Trace through the algorithm on the input $FunctionSolver(x^2 - 10, 4, 0.25)$.

(c) If $n = 2^j$ and $e = 2^k$ for some integers $k$ and $j$ such that $k < j$, how many iterations does the algorithm go through and why?

(d) The loop invariant for this algorithm is:

After each iteration, $lo \leq r \leq hi$.

Finish the proof that this loop invariant is true by filling in case 3.

**Basis Step:** Before the while loop begins, $lo \leq r \leq hi$ because it is assumed that $0 \leq r \leq n$.

**Inductive Hypothesis:** Assume that after $t$ iterations, $lo \leq r \leq hi$ for some $t \geq 0$.

**Inductive Step:** (Want to show that after the next iteration, $lo \leq r \leq hi$.

**Case 1:** If $f(v) = 0$ then $lo$ and $hi$ do not change, so by the inductive hypothesis, $lo \leq r \leq hi$.

**Case 2:** If $f(v) < 0$ then $lo = v$. Since the function is increasing, since $f(r) = 0$ and $f(v) < f(r)$, then $v < r$. So if we set $lo = v$, then $lo \leq r$. And by the inductive hypothesis, since we did not change $hi$, $r \leq hi$, so $lo \leq r \leq hi$.

**Case 3:** If $f(v) > 0$. [Fill in the argument for this case.]

Therefore, after the next iteration, $lo \leq r \leq hi$ and by induction, after any number of iterations, $lo \leq r \leq hi$.

(e) Assuming the loop invariant is correct, show that the algorithm is correct. (i.e., show that when the loop terminates, the algorithm will output $v$ such that $|r - v| \leq e$.

3. (15 points) For each algorithm, compute the exact number of times the algorithm prints in terms of $n$ and compute the runtime in terms of $\Theta$, where $n \geq 3$. (We use the convention that if there is a for loop such that the starting value is greater than the ending value, then that loop does not run.)

(a)
```
for $i = 1$ to $n$:
    for $j = i$ to $n - i$:
        print $(i, j)$.
```

(b)
```
for $i = 2$ to $n$:
    for $j = i - 1$ to $n$:
        print $(i, j)$.
```

(c)
```
for $i = 1$ to $n - 1$:
    for $j = i + 1$ to $n$:
        for $k = 1$ to $n$:
            print $(i, j, k)$
```

4. (21 points) State if each statement is true or false and give a justification either way. (you may use any of the methods used in class including the limit rules.)

(3 points each)

(a) $\sqrt{n^5 + 2n^2 + n^8} \in O(n^4)$

(b) $n \in O(n * 2^n/3^n)$

(c) $2 + 4 + 6 + \cdots + 2n \in \Omega(n^2)$

(d) $\binom{2n}{n} \in O(4^n)$

(e) $4^{\log_2(n)} = \Theta(2^{\log_4(n)})$

(f) $\left(\sqrt[3]{n}\right)^2 \in O(n \log n)$

(g) $n! \in \Theta((n-1)!)$