CS 344
Fall 2022

Problem Set 1
Due: October 10, 2022, 11:59 p.m.

Name: (Replace me with your name) **NetID**: (Replace me with your NetID)

Honor Code: Students may discuss homework problems with peers. However, each student must write down their solutions independently to show they understand the solution well enough to reconstruct it by themselves. Students should clearly mention the names of the other students who offered discussions. We check all submissions for plagiarism. We take the honor code seriously and expect students to do the same.

Instruction for Submission: This homework has a total of 100 points, it will be rescaled to 10 points as the eventual score.

We encourage you to use LaTex to write your answer, because it's particularly suitable to type equations and it's frequently used for writing academic papers. We have provided the homework1.tex file for you, you can write your answer to each question in this .tex file directly after the corresponding question, and then compile the .tex file into a PDF file for submission. As a quick introduction to LaTex, please see this *Learn LaTeX in 30 minutes* [1]. Compiling a .tex file into a PDF file needs installing the LaTex software on your laptop. It's free and open source. But if you don't want to install the software, you can just use this website `https://www.overleaf.com/`, which is also free of charge. You can just create an empty project in this website and upload the homework1.zip, and then the website will compile the PDF for you. You can also directly edit your answers on the website and instantly compile your file.

You can also use Microsoft Word or other software if you don't want to use LaTex. If so, please clearly number your answers so that we know which answer corresponds to which question. You also need to save your answers as a PDF file for submission.

Finally, please submit your PDF file only. You should name your PDF file as "Firstname-Lastname-NetID.pdf".

Late Policy: The homework is due on 10/10 (Monday) at 11:59pm. We will release the solutions of the homework on Canvas on 10/14 (Friday) 11:59pm. If your homework is submitted to Canvas before 10/10 11:59pm, there will no late penalty. If you submit to Canvas after 10/10 11:59pm and before 10/14 11:59pm (i.e., before we release the solution), your score will be penalized by 0.9^k, where k is the number of days of late submission. For example, if you submitted on 10/13, and your original score is 80, then your final score will be $80 * 0.9^3 = 58.32$ for $13 - 10 = 3$ days of late submission. If you submit to Canvas after 10/14 11:59pm (i.e., after we release the solution), then you will earn no score for the homework.

Make best use of picture in Latex: If you think some part of your answer is too difficult to type using Latex, you may write that part on paper and scan it as a picture, and then insert that part into Latex as a picture, and finally Latex will compile the picture into the final PDF

[1] `https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes`

output. This will make things easier. Regarding how to insert pictures in Latex, you may refer to the Figure 1 in the following, which is an example of inserting pictures in Latex.

Discussion Peers (People with whom you discussed ideas used in your answers if any):

I acknowledge and accept the Honor Code. Please type your initials below:
Signed: (Replace me with your initials)

---

1. **Big-O notation.** We have learnt big-O notation to compare the growth rates of functions, this exercise helps you to better understand its definition and properties.

   (a) (14 points) Suppose n is the input size, we have the following commonly seen functions in complexity analysis: $f_1(n) = 1, f_2(n) = \log n, f_3(n) = n, f_4(n) = n \log n, f_5(n) = n^2, f_6(n) = 2^n, f_7(n) = n!, f_8(n) = n^n$. Intuitively, the growth rate of the functions satisfy $1 < \log n < n < n \log n < n^2 < 2^n < n! < n^n$. Prove this is true.

   [**Hint**: You are expected to prove the following asymptotics by using the definition of big-O notation: $1 = O(\log n), \log n = O(n), n = O(n \log n), n \log n = O(n^2), n^2 = O(2^n), 2^n = O(n!), n! = O(n^n)$. **Note**: Chap 3.2 of our textbook provides some math facts in case you need.]

   Answer: my answer to the question.

   (b) (3 points) Let $f, g : N \to R^+$, prove that $O(f(n) + g(n)) = O(\max\{f(n), g(n)\})$.

   [**Hint**: The key is $\max\{f(n), g(n)\} \leq f(n) + g(n) \leq 2 \cdot \max\{f(n), g(n)\}$. **Note**: Proving this will help you to understand why we can leave out the insignificant parts in big-O notation and only keep the dominate part, e.g., $O(n^2 + n \log n + n) = O(n^2)$.]

   Answer:

   (c) (3 points) Let $f, g : N \to R^+$, prove that $\Omega(f(n) + g(n)) = \Omega(\max\{f(n), g(n)\})$.

   [**Note**: Proving this will help you understand that we can also leave out the insignificant parts in big-Ω notation and the result is still a lower bound, e.g., $\Omega(n^2 + n \log n + n) = \Omega(n^2)$.]

   Answer:

2. **Proof of correctness.** (10 points) We have the following algorithm that sorts a list of integers to descending order. Prove that this algorithm is correct. [**Hint:** You are expected to use mathematical induction to provide a rigorous proof.]

   Answer:

Algorithm 1: `Sort a list in descending order`

```
Input: Unsorted list $A = [a_1, \ldots, a_n]$ of $n$ items
Output: Sorted list $A' = [a'_1, \ldots, a'_n]$ of $n$ items in descending order
for $i = 0$, $i < n - 1$, $i$++ do
    // Find the maximum element
    max_index = $i$
    for $j = i + 1$, $j < n$, $j$++ do
        if $A[j] > A[\text{min\_index}]$ then
            max_index = $j$
    // Swap the maximum element with the first element
    swap($A$, $i$, max_index)
return $A$
```

3. **Practice the recursion tree.** (10 points) We have already had a recurrence relation of an algorithm, which is $T(n) = 4T(n/2) + 2n$. Solve this recurrence relation, i.e. express it as $T(n) = O(f(n))$, by using the recursion tree method. [**Note**: If you find it difficult to draw a picture using Latex directly, you can draw it using Microsoft PowerPoint and then save it as a picture file (.png or .jpg), or you can draw on a piece of paper by hand, and take a picture of it using your phone. Then, you can insert the picture into your latex code and compile it into the final PDF submission. The following code shows an example of how to insert figures in latex code, as shown in Figure 1.]

Figure 1: An example showing how to insert figures in latex code.

Answer:

4. **Practice with the iteration method.** We have already had a recurrence relation of an algorithm, which is $T(n) = 4T(n/2) + n \log n$. We know $T(1) \leq c$.

   (a) (5 points) Solve this recurrence relation, i.e., express it as $T(n) = O(f(n))$, by using the iteration method.

   Answer:

   (b) (5 points) Prove, by using mathematical induction, that the iteration rule you have observed in 4(a) is correct and you have solved the recurrence relation correctly. [**Hint**: You can write out the general form of $T(n)$ at the iteration step t, and prove that this form is correct for any iteration step t by using mathematical induction. Then by finding out the eventual number of t and substituting it into your general form of $T(n)$, you get the $O(\cdot)$ notation of $T(n)$.]

Answer:

5. **Practice with the Master Theorem**. Solve the following recurrence relations; i.e. express each one as $T(n) = O(f(n))$ for the tightest possible function $f(n)$ using the Master Theorem, and give a short justification. 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) (5 points) $T(n) = 4T(n/4) + \sqrt{n}$

   Answer:

   (b) (5 points) $T(n) = 6T(n/3) + \Theta(n^3)$

   Answer:

   (c) (5 points) $T(n) = 2T(n/3) + n^c$, where $c \geq 1$ is a constant that doesn't depend on n.

   Answer:

6. **Proof of the Master Theorem.** (15 points) Now that we have practiced with the recursion tree method, the iteration method, and the Master method. The Master Theorem states that, suppose $T(n) = a \cdot T(n/b) + O(n^d)$, we have:

$$T(n) = \begin{cases} O(n^d \log n), & if\ a = b^d \\ O(n^d), & if\ a < b^d \\ O(n^{\log_b a}), & if\ a > b^d \end{cases}$$

   Prove that the Master Theorem is true by using either the recursion tree method or the iteration method.

   Answer:

7. **Algorithm design with sorting.** 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	2
2	1	4
3	2	5
4	7	8
5	9	10
6	6	10

Then, the number of distinct pairs of users who are on the site at the same time is five: these pairs are $(1, 2)$, $(1, 3)$, $(2, 3)$, $(4, 6)$, $(5, 6)$. (Drawing the intervals on a number line may make this easier to see).

(a) (5 points) 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.**]

Answer:

(b) (5 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.**]

Answer:

8. **Algorithm design with divide and conquer.** Given an array $A[0 : n-1]$ of n integers, compute the maximum sum of non-empty consecutive subsequence present in the array.

For example, if the input is $A = [3, 2, 5, 1, 6]$, then the output max_sum is 17 (the corresponding maximum sum consecutive subsequence is $[3, 2, 5, 1, 6]$. If the input is $A = [-2, 11, -4, 13, -5, -2]$, then the output max_sum is 20 (the corresponding maximum sum consecutive subsequence is $[11, -4, 13]$.

(a) (5 points). Given input $A[0 : n-1]$ and n, design a divide-and-conquer algorithm which outputs the maximum sum of non-empty consecutive subsequence in the array. You only need to output the sum value and do not need to output the exact subsequence. [**We are expecting a brief justification for the intuitive idea of your algorithm and the pseudocode of the algorithm.**]

(b) (5 points). Let $T(n)$ be the runtime of your algorithm when the input size is n. Establish the recurrence relation of $T(n)$ for your algorithm, and then solve the recurrence relation to provide the big-O runtime of your algorithm. [**You can use any method we introduced in class to solve your recurrence relation.**]