CS 330, Fall 2019, Homework 1, Due Wednesday, September 11, 2019

Homework Guidelines

Please make sure you read and sign the collaboration policy before you start working on your homework. (We will grade your homework only if we have a signed copy on file.) Refer to the general information handout for the homework policy and additional options.

Collaboration policy Collaboration on homework problems, with the exception of programming assignments and reading quizzes, is permitted, but not encouraged. If you choose to collaborate on some problems, you are allowed to discuss each problem with at most 5 other students currently enrolled in the class. Before working with others on a problem, you should think about it yourself for at least 45 minutes. Finding answers to problems on the Web or from other outside sources (these include anyone not enrolled in the class) is strictly forbidden.

You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem. You must also identify your collaborators. If you did not work with anyone, you should write "Collaborators: none." It is a violation of this policy to submit a problem solution that you cannot orally explain to an instructor or TA.

Solution guidelines For problems that require you to provide an algorithm, you must give the following:

1. a precise description of the algorithm in English and, if helpful, pseudocode,
2. a proof of correctness,
3. an analysis of running time and space.

You may use algorithms from class as subroutines. You may also use any facts that we proved in class.

You should be as clear and concise as possible in your write-up of solutions. Understandability of your answer is as desirable as correctness, because communication of technical material is an important skill. A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error and because it is easier to read and understand. Points might be subtracted for illegible handwriting and for solutions that are too long. Incorrect solutions will get from 0 to 30% of the grade, depending on how far they are from a working solution. Correct solutions with possibly minor flaws will get 70 to 100%, depending on the flaws and clarity of the write up.

0. (0 points) Print a nameplate for yourself (using the template on Piazza) and bring it to class.
1. **Stable Matching** (2-page limit, 10 points)

   (a) (**Truthfulness in Stable Matching**) Chapter 1, problem 8.
   Hint: Try playing with several specific examples of preference lists.

(b) (**Unique Matching**) Give an algorithm that takes an instance of the stable matching problem as input and decides if there is *exactly one* stable matching for this instance (that is, the algorithm outputs either "unique stable matching", or "more than one stable matching").

Hint: You may want to read the section on "Extensions" at the end of KT Section 1.1.

2. **Proofs and Loop Invariants** (2-page limit, 10 points)

(a) Consider the following pseudocode:

```
Algorithm 1: TestAlg(A)
Input: A is an array of real numbers, indexed from 1 to n
n ← length(A) ;
for j ← 1 to ⌊n/2⌋ do
    k = n + 1 − j;
    A[j] ← A[j] + A[k] ;
    A[k] ← A[j] − A[k] ;
    A[j] ← A[j] − A[k] ;
```

Which of the following invariants are true at the end of every iteration of the **for** loop?

i. The sub-array $A[1 \ldots j]$ contains its original contents in their original order

ii. The sub-array $A[1 \ldots j]$ contains the original contents of sub-array $A[(n+1-j) \ldots n]$ in reverse order.

iii. The sub-array $A[1 \ldots j]$ contains its original contents in reverse order.

iv. The sub-array $A[(n+1-j) \ldots n]$ contains its original contents in their original order.

v. The sub-array $A[(n+1-j) \ldots n]$ contains the original contents of sub-array $A[1 \ldots j]$ in reverse order.

vi. The sub-array $A[(n+1-j) \ldots n]$ contains its original contents in reverse order.

(Just list the ones that are always true.)

(b) Recall the Merge Sort algorithm from your previous classes, for which we have provided concise pseudocode. When given an array A indexed from 1 to n, the initial call (that sorts the entire array) is Merge-Sort$(A, 1, n)$.

```
Algorithm 2: Merge-Sort(A, p, r)
if p < r then
    q = ⌊(p + r)/2)⌋;
    Merge-Sort(A, p, q);
    Merge-Sort(A, q + 1, r);
    Merge(A, p, q, r) /* Merges the sorted subarrays A[p], ..., A[q] and A[q + 1], ..., A[r] */
```

How many calls to Merge-Sort (including the first) are performed in total when we call Merge-Sort$(A, 1, n)$, as a function of n?

Give your answer as a precise, self-contained claim (e.g. "The total number of calls to Merge-Sort performed when we run Merge-Sort$(A, 1n)$ is $6^n - 123$."). [Hint: Run the algorithm by hand on a few inputs and look for a pattern.]

Using strong induction, prove that your claim is correct.

In order to make your proof precise, it will help to generalize your claim to give a formula for the total number of calls to Merge-Sort (including the first) when we run Merge-Sort(A, p, r), as a function of $r - p$.

As with any proof by induction, you

i. State the claim you want to prove.
ii. State and prove the base case.
iii. State the Induction hypothesis.
iv. Prove the induction step.
v. Apply the induction principle to prove the claim you wanted.

(For examples, see `https://courses.cs.washington.edu/courses/cse417/12wi/notes/proofs.pdf`).

3. **Asymptotics Review** (10 points, 1-page limit)

Rank the following functions by order of growth; that is, find an arrangement of the expressions in increasing order of their asymptotic behavior, in $O(\ )$ (big-Oh) notation. The answer you need to submit for this problem is simply an ordered list of the expressions below, from slowest to fastest, and giving a 1-sentence justification for each consecutive pair in the list. Whenever two consecutive expressions in the list are asymptotically equivalent, please clarify and justify, i.e., by writing "$n^2 + 5n + 1$ is $\Theta(n^2)$ because the low-order terms in the first expression can be dropped". (Note: in this class, $\log n$ means $\log_2 n$).

$$\begin{array}{ccccc}
n^{100} & (\sqrt{2})^{\log n} & n^2 & \sqrt{n} & 2^{2^{n+1}} \\
n^3 & \log^2 n & \log(n!) & 2^{2^n} & n^{1/\log n} \\
\log n & 2^{100 \log n} & n \cdot 2^n & n^{\log \log n} & \log_3 5n \\
\binom{n}{2} + n \log n & 4^{\log n} & n & n \log n & 2^{\log^{1.001} n}
\end{array}$$