**COMPSCI 311: Introduction to Algorithms** **Spring 2019**

# Homework 1

Released 1/24/2019 Due ~~2/7/2019~~ 2/8/2019 11:59pm in Gradescope

**Instructions.** You make work in groups, but you must write solutions yourself. List collaborators on your submission.

If you are asked to design an algorithm, please provide: (a) the pseudocode or precise description in words of the algorithm, (b) an explanation of the intuition for the algorithm, (c) a proof of correctness, (d) the running time of your algorithm and (e) justification for your running time analysis.

**Submissions.** Please submit a PDF file. You may submit a scanned handwritten document, but a typed submission is preferred. Please assign pages to questions in Gradescope.

1. **(15 points) Stable Matching Running Time.** In class, we saw that the Propose-and-reject algorithm terminates in at most $n^2$ iterations, when there are $n$ students and $n$ colleges.
   (a) Construct an instance so that the algorithm requires only $O(n)$ iterations, and prove this fact. Your construction should be possible for all values of $n$.
   (b) Construct an instance so that the algorithm requires $\Omega(n^2)$ iterations (that is, it requires at least $cn^2$ iterations for some constant $0 < c \leq 1$), and prove this fact. Your construction should be possible for all values of $n$.

2. **(20 points) Stable Matchings: K&T Ch 1, Ex 5.** Consider a version of the stable matching problem where there are $n$ students and $n$ colleges as before. Assume each student ranks the colleges (and vice versa), but now we allow ties in the ranking. In other words, we could have a school that is indifferent two students $s_1$ and $s_2$, but prefers either of them over some other student $s_3$ (and vice versa). We say a student $s$ <u>prefers</u> college $c_1$ to $c_2$ if $c_1$ is ranked higher on the $s$'s preference list and $c_1$ and $c_2$ are not tied.
   (a) **Strong Instability.** A strong instability in a matching is a student-college pair, each of which prefer each other to their current pairing. In other words, neither is indifferent about the switch. Does there always exist a matching with no strong instability? Either give an example instance for which all matchings have a strong instability (and prove it), or give and analyze an algorithm that is guaranteed to find a matching with no strong instabilities.
   (b) **Weak Instability.** A weak instability in a matching is a student-college pair where one party prefers the other, and the other may be indifferent. Formally, a student $s$ and a college $c$ with pairs $c'$ and $s'$ form a weak instability if either
   - $s$ prefers $c$ to $c'$ and $c$ either prefers $s$ to $s'$ or is indifferent between $s$ and $s'$.
   - $c$ prefers $s$ to $s'$ and $s$ either prefers $c$ to $c'$ or is indifferent between $c$ and $c'$.

   Does there always exist a perfect matching with no weak instability? Either give an instance with a weak instability or an algorithm that is guaranteed to find one.

3. **(15 points) Big-O.** For each function $f(n)$ below, find (1) the smallest <u>integer</u> constant $H$ such that $f(n) = O(n^H)$, and (2) the largest <u>positive real</u> constant $L$ such that $f(n) = \Omega(n^L)$.
   Otherwise, indicate that $H$ or $L$ do not exist. All logarithms are with base 2. Your answer should consist of: (1) the correct value of $H$, (2) a proof that $f(n)$ is $O(n^H)$, (3) the correct value of $L$, (4) a proof that $f(n)$ is $\Omega(n^L)$.
   (a) $f(n) = \frac{1}{2}n^2$.
   (b) $f(n) = n(\log n)^3$
   (c) $f(n) = \sum_{i=0}^{\lceil \log n \rceil} \frac{n}{2^i}$.
   (d) $f(n) = \sum_{i=1}^{n} i^3$.
   (e) $f(n) = 2^{(\log n)^2}$

4. **(20 points) Asymptotics. K&T Ch 2, Ex 6.** Given an array $A$ of $n$ integers, you'd like to output a two-dimensional $n \times n$ array $B$ in which $B[i, j] = \max\{A[i], A[i+1], \dots, A[j]\}$ for each $i < j$. For $i \geq j$ the value of $B[i, j]$ can be left as is.

```
for $i = 1, 2, \dots, n$
    for $j = i + 1, \dots, n$
        Compute the maximum of the entries $A[i], A[i+1], \dots, A[j]$.
        Store the maximum value in $B[i, j]$.
```

   (a) Find a function $f$ such that the running time of the algorithm is $O(f(n))$, and clearly explain why.

   (b) For the same function $f$ argue that the running time of the algorithm is also $\Omega(f(n))$. (This establishes an asymptotically tight bound $\Theta(f(n))$.)

   (c) Design and analyze a faster algorithm for this problem. You should give an algorithm with running $O(g(n))$, where $\lim_{n\to\infty} g(n)/f(n) = 0$.

5. **(10 points) DFS and BFS. K&T Ch 3, Ex 5.** Suppose we have a connected graph $G = (V, E)$ and a vertex $u \in V$. If we run DFS from $u$, we obtain a tree $T$. Suppose that if we run BFS from $u$ we obtain exactly the same tree $T$. Prove that $G = T$. (Hint: see Facts (3.4) and (3.7) from the book.)

6. **(20 points) Butterfly ID. K&T Ch 3 Ex 4.** Some of your friends have become amateur lepidopterists (they study butterflies). Often when they return from a trip with specimens of butterflies, it is very difficult for them to tell how many distinct species they've caught—thanks to the fact that many species look very similar to one another.
One day they return with $n$ butterflies, and they believe that each belongs to one of two different species, which well call $A$ and $B$ for purposes of this discussion. They'd like to divide the $n$ specimens into two groups—those that belong to $A$ and those that belong to $B$—but it's very hard for them to directly label any one specimen. So they decide to adopt the following approach.
For each pair of specimens $i$ and $j$, they study them carefully side by side. If theyre confident enough in their judgment, then they label the pair $(i, j)$ either "same" (meaning they believe them both to come from the same species) or "different" (meaning they believe them to come from different species). They also have the option of rendering no judgment on a given pair, in which case we'll call the pair ambiguous.
So now they have the collection of $n$ specimens, as well as a collection of $m$ judgments (either "same" or "different") for the pairs that were not declared to be ambiguous. They'd like to know if this data is consistent with the idea that each butterfly is from one of species $A$ or $B$. So more concretely, we'll declare the $m$ judgments to be consistent if it is possible to label each specimen either $A$ or $B$ in such a way that for each pair $(i, j)$ labeled "same," it is the case that $i$ and $j$ have the same label; and for each pair $(i, j)$ labeled "different," it is the case that $i$ and $j$ have different labels. They're in the middle of tediously working out whether their judgments are consistent, when one of them realizes that you probably have an algorithm that would answer this question right away. Give an algorithm with running time $O(m + n)$ that determines whether the $m$ judgments are consistent.

7. **(0 points).** How long did it take you to complete this assignment?