Homework 2

Due Wednesday, February 2nd at 11:59 pm ET on Gradescope

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 pseudocode (*),
2. a proof of correctness,
3. an analysis of the asymptotic running time and space.

You may use algorithms from class as subroutines. You may also use any facts that we proved in class, e.g. correctness of subroutines, running time of subroutines.

You should be as clear and concise as possible in your write-up of solutions.

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.

(*) It is fine if the English description concentrates on the high level ideas and doesn't include all the details. But the reader should not have to figure out your solution solely based on the pseudocode. You can also add comments to your pseudocode, in fact that is best practice. FYI we will share a document on good pseudocode style with you and it will also be discussed in labs and lecture.

Problem 1. *(10 points)*

Write the functions given to you as well as their asymptotic growth functions (in terms of Θ) in *increasing* order of complexity. You don't need to write any proofs on top of this. Note that some of the functions might be of the same complexity, in that case write them on the same line.

As we saw in class, in terms of asymptotics the base of the logarithm doesn't matter. For this class we will always assume it's base-2 logarithm, and will omit the subscript from the notation.

Here is an example on the format: suppose that the functions are $n^2 + 2$, $n + \log n$, $n^2 + \log n$, $3n$, $\log n$. *Then you would write*

1. $\log n = \Theta(\log n)$
2. $n + \log n$, $3n = \Theta(n)$
3. $n^2 + 2$, $n^2 + \log n = \Theta(n^2)$

These are the functions you need to order:

1. $n^{0.98898} \log(n)$
2. 1.00001^n
3. $99n^2 + \sqrt{n} \log n$
4. $\log(2^n) \cdot \log(n^2)$
5. $n\sqrt{n}$
6. $1.01 \cdot n \log(\mathrm{n})$
7. $15 \cdot 2^n$
8. $\binom{2n}{2}$
9. 2^{2n}
10. $2^{n/2}$
11. $n!$
12. $n^{\sqrt{n}}$
13. $2 - 3n^2$
14. $\sum_{i=1}^{n} i$
15. $(n-1)!$

Problem 2. *(10 points)*

During the internship season m tech companies are looking to hire CS undergrads, each company has k number of available positions. There are also n CS undergrads, each interested in joining one of the companies. After all the interviews each company has rankings of students in order of preference, and each student has a ranking of companies in order of preference. We can assume that there are more internship slots available than the number of students. (This means that at the end not all slots will be filled.)
Describe an algorithm that finds a stable assignment of students to companies. That is, each student will get assigned to one company and the resulting matching is stable.
In your answer make sure to follow the guidelines of solving an algorithmic problem, as described above.