CSE 015: Discrete Mathematics
Homework 7
Fall 2021

Preliminary Notes

- **This homework must be solved individually.** You can discuss your ideas with others, but when you prepare your solution you must work individually. Your submission must be yours and yours only. No exceptions, and be reminded of the CSE academic honesty policy discussed in class.
- Your solution must be exclusively submitted via CatCourses. Pay attention to the posted deadline because **the system automatically stops accepting submissions when the deadline passes. Late submissions will receive a 0**. You only need to submit the PDF and you have to use the template file provided in CatCourses. Please note that the system does not allow to submit any other file format. Do not submit the LaTeXsource of your solution.
- By now you should have become somewhat familiar with LaTeX. You still will not be penalized for poor typesetting, but it is in your own interest to prepare your submission in a way that is easy to understand, so try using the appropriate LaTeXsymbols. If you do not know how to type a certain math symbol, search on the Internet and you will quickly find the answer.[1] **If in your LaTeXsubmission you embed screenshots or scans of your handwritten solution those will not be graded.** You are encouraged to collaborate with other students to determine how to best format your submission or improve your LaTeXskills.
- Start early.

1 Asymptotic Notation

Determine if each of the following functions is $O(n^2)$ or not. You must motivate your answer. Simply answering "yes/no" will not give you any credit, even if the answer is correct.

a) $f(n) = 178n + 45$

b) $f(n) = n \log n + 12$

c) $f(n) = 34n^2 + 34n + 34$

d) $f(n) = \sqrt{n} + 2$

e) $f(n) = 0.001n^3 + 72n$

[1] see `https://www.caam.rice.edu/~heinken/latex/symbols.pdf` for example.

2 Asymptotic Notation

Arrange each of the following function in an ordered list such that each is big-O of the next in the list.

1. $n \log n$
2. n^2
3. $\log n$
4. n
5. $\sqrt{n}$
6. 2^n
7. n^4
8. 3^n
9. $n^2 \log n$

3 Asymptotic Growth

In the study of algorithms we focus on the *asymptotic growth* of functions describing time complexity. That means that we are interested in how the performance of an algorithm scales with the size of the input for *large* inputs. As you will study later on, algorithms whose time complexity is characterized by exponential functions are normally avoided because even great improvements in computational complexity result in marginal (if at all) improvements in the size of the problems that can be solved in a reasonable amount of time. To see why, consider the case where you have two hypothetical computers A and B, with B being 100 times faster than A. To be more precise, A can execute 10^6 operations per second, while B can execute 10^8 operations per second. Assume now that A and B are used to execute three algorithms whose *exact* time complexities are

- $f_1(n) = 5n^2 + 34n + 12$
- $f_2(n) = 10n + 4$
- $f_3(n) = 2^n$.

Here, *exact* time complexity means that when executing an algorithm of size n the algorithm executes $f(n)$ operations. Since we know how much time it takes to execute an operation on A and B, then we know how long it takes to execute each of these algorithms for a given value of n. Assume you have 1 hour of computational time at your disposal on both computer A and computer B. What is the maximum size of the problem you can execute on both computers with each of the algorithms?
To help you in determining your answer, here is how we compute the answer for algorithm 1 on computer A. In one hour there are 3600 seconds, i.e., in one hour computer A can execute $3600 \cdot 10^6$ operations, i.e., $3.6 \cdot 10^9$ operations. To process an input of size n, algorithm 1 executes $5n^2 + 34n + 12$ operations. So the question is to find the largest value of n such that $5n^2 + 34n + 12 \leq 3.6 \cdot 10^9$. This inequality can be solved in many ways (any will do) and the answer is $n = 26829$, i.e., 26829 is

the largest value of n such that $5n^2 + 34n + 12 \leq 3.6 \cdot 10^9$ (verify that!) This means that in one hour, in the hypothetical computer A, algorithm 1 could solve an instance of the problem of size $n = 26829$. Repeat the reasoning and find the answers for all the other cases, i.e., algorithms 2 and 3 on computer A, and algorithms 1,2 and 3 on computer B and compare the answers.
Why this exercise? This shows two things. First it shows that if you have an exponential algorithm, even substantial gains in computational power do not lead to an ability to solve significantly larger problem instances. This is why algorithms with exponential complexity are in most cases avoided. Second, it also shows that if you have a linear algorithm (i.e., an $O(n)$ algorithm) then its performance scales roughly linearly, irrespective of the constants. Likewise, a quadratic algorithm (i.e., an $O(n^2)$ algorithm) will roughly require four times as much time to process an input twice as big (again, irrespective of the constants.)