CSCI 332, Fall 2024
Homework 2

Due before class on Tuesday, September 8, 2024–that is, due at 9:30am Mountain Time

---

Submission Requirements

- Type or clearly hand-write your solutions into a PDF format so that they are legible and professional. Submit your PDF on Gradescope.
- Do not submit your first draft. Type or clearly re-write your solutions for your final submission.
- Use Gradescope to assign problems to the correct page(s) in your solution. If you do not do this correctly, we will ask you to resubmit.
- You may work with a group of up to three students and submit ***one single document*** for the group. Just be sure to list all group members at the top of the document. When submitting a group assignment to Gradescope, only one student needs to upload the document; just be sure to select your groupmates when you do so.

Academic Integrity

Remember, you may access ***any*** resource in preparing your solution to the homework. However, you ***must***

- write your solutions in your own words, and
- credit every resource you use (for example: "Bob Smith helped me on this problem. He took this course at UM in Fall 2020"; "I found a solution to a problem similar to this one in the lecture notes for a different course, found at this link: www.profzeno.com/agreatclass/lecture10"; "I asked ChatGPT how to solve part (c)"; "I put my solution for part (c) into ChatGPT to check that it was correct and it caught a missing case.") If you use the provided LaTeX template, you can use the `sources` environment for this. Ask if you need help!

Grading

Remember, submitted homeworks are graded for completeness, not correctness. Correctness is evaluated using homework quizzes.

Each submitted problem will be graded out of six points according to the following rubric:

- Does the solution address the correct problem?
- Does the solution make a reasonable attempt at solving the problem, even if not fully correct?
- Is the presentation neat?
- Is the explanation clear?

- Does the solution list collaborators or sources, or state that the student did not use any collaborators or outside resources?
- Is the solution written in the student's own voice (not copied directly from an outside resource)?

1. (This is problem 1 from Chapter 2 of the textbook)

   suppose you have an algorithm with the five running times listed below. (Assume these are exact running times as a function of the input size n, not asymptotic running times.) How much slower do each of these algorithms get when you (i) double the input size, or (ii) increase the input size by one?

   (a) n^2
   (b) n^3
   (c) $100n^2$
   (d) $n \log n$
   (e) 2^n

2. (This is problem 2 from Chapter 2 of the textbook)

   Suppose you have an algorithm with the six running times listed below. (Assume these are the exact number of operations performed as a function of the input size n, not asymptotic running times.) Suppose you have a computer that can perform 10^10 operations per second, and you need to compute a result in at most an hour of computation. For each of the algorithms, what is the largest input size n for which you would be able to get the result within an hour?

   (a) n^2
   (b) n^3
   (c) $100n^2$
   (d) $n \log n$
   (e) 2^n
   (f) 2^{2^n}

3. (This problem is not in your textbook.)

   Recall the definition of big O:

   $f(n)$ is $O(g(n))$ if there exist positive constants n_0, c such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$.

   Using this definition, prove that $2n^2 + 3$ is $O(3n^3 + 2n^2 + 4n + 1$.

4. (This is problem 3 from Chapter 2 of the textbook)

   Take the following list of functions and arrange them in ascending order of growth rate. That is, if function $g(n)$ immediately follows $f(n)$ in your list, then it should be the case that $f(n)$ is $O(g(n))$.

   - $f_1(n) = n^{2.5}$
   - $f_2(n) = \sqrt{2n}$
   - $f_3(n) = n + 10$
   - $f_4(n) = 10^n$
   - $f_5(n) = 100^n$
   - $f_6(n) = n^2 \log n$

5. (This is problem 6 from Chapter 2)

   Consider the following basic problem. You're given an array A consisting of n integers $A[1], A[2], \ldots, A[n]$. You'd like to output a two-dimensional n-by-n array B in Which $B[i,j]$ (for $i < j$) contains the sum of array entries $A[i]$ through $A[j]$–that is, the sum $A[i] + A[i+1] + \cdots + A[j]$. (The value of array entry $B[i,j]$ is left unspecified whenever $i \leq j$, so it doesn't matter what is output by those values.)

```
For $i = 1, 2, \ldots, n$
  For $j = i + 2, i + 3, \ldots, n$
      Add up array entries $A[i]$ through $A[j]$
      Store the result in $B[i, j$
  Endfor
Endfor
```

   (a) For some function f that you should choose, give a bound of the form $O(f(n))$ on the running time of this algorithm on an input of size n (i.e., a bound on the number of operations performed by the algorithm).

   (b) For this same function f, show that the running time of the algorithm on an input of size n is also $\Omega(f(n))$. (This shows an asymptotically tight bound of $\Theta(f(n))$ on the running time.)

   (c) Although the algorithm you analyzed in parts (a) and (b) is the most natural way to solve the problem—after all, it just iterates through the relevant entries of the array B, filling a value for each—it contains some highly unnecessary sources of inefficiency. Give a different algorithm to solve this problem, with an asymptotically better running time. In other words, you should design an algorithm with running time $O(g(n))$, where $\lim_{n \to \inf} g(n)/f(n) = 0$.