EECS 325 – Fall 2023
Analysis of Algorithms

Homework 1

Instructions:

- This homework is due by 11:59PM Pacific Time on **Thursday, October 12th**. Assignments submitted up to two days late will incur a penalty of up to 20% (10% per day, rounded up; no penalty past a score of 80%). Assignments submitted more than two days late will not be graded and will receive a score of 0. Your lowest homework score of the quarter (after applying late penalties) will be dropped.

- You must submit your written solutions as a single PDF file on Canvas with your name at the top. Typesetting your written solutions using LaTeX is strongly encouraged, and you will receive three bonus points (+3) if you write your solutions to Homework 1 in Latex. To receive these points, write `''This homework was written in \LaTeX''` (which will render as "This homework was written in LaTeX") at the top of your solution in the .tex file with your solutions.

- You must write your programming solutions (for Question 1, Parts a and b and Question 2, Part a) using Python 3 using the format specified by the provided skeleton `max_subarray_homework1.py` and `factoring_homework1.py` files.

- You are welcome and encouraged to discuss the problems in groups of up to three people, but **you must write up and submit your own solutions and code**. You must also write the names of everyone in your group on the top of your submission. Students in the honors and non-honors section may collaborate.

- The primary resources for this class are the lectures, lecture slides, the CLRS and Erickson algorithms textbooks, the teaching staff, your (up to two) collaborators, and the class Piazza forum. We strongly encourage you only to use these resources. If you do use another resource, make sure to cite it and explain why you needed it. **Using ChatGPT or other generative AI tools is not allowed unless, except when explicitly noted.**

- There are three questions, worth 40 points in total (with the possibility of 3 points of extra credit from using LaTeX).

- You must justify all of your answers unless specifically stated otherwise.

Questions

Question 1. (Maximum subarray problem, 15 points.) In this question, you will study the following computational problem:

Maximum Subarray Problem
Input: An array $A[0, \ldots, n-1]$ of n integers.
Output: The maximum sum of all elements in some (possibly empty) contiguous subarray of $A[0, \ldots, n-1]$. Formally,

$$\max\left(0, \max_{i \le j} \sum_{k=i}^{j} A[k]\right) ,$$

where 0 corresponds to the value of the empty array.

You will implement the two algorithms described in Prof. Borradaile's "TCS crash course" pages 9 and 10.[1] You will then run your code on inputs of different sizes, and will report on the performance of each algorithm. *Make sure to save your work from this problem. Later in the course you will implement the remaining two algorithms in the TCS crash course and compare them against your results here.*

a. (4 points.) Implement the $O(n^3)$-time "enumeration" algorithm.

b. (4 points.) Implement the $O(n^2)$-time "iteration" algorithm.

c. (4 points.) Measure the running time of each of the algorithms from Items a and b on each of the 10 provided input arrays (which are of size $n = 500i$ for $i = 1, 2, \ldots, 10$).

   Make a scatter plot of the running time. The x-axis should be the size n of the input array, and the y-axis should be the running time of the algorithm in milliseconds. Make the points corresponding to each of Items a and b a different color and/or shape. (Making a scatter plot of this form should be easy in Excel, Google Sheets, etc.)

d. (3 points, ♣.) Briefly describe how your results compare with the runtime estimates from the 1984 Digital Equipment Corporation experiments presented on page 13 in the "TCS crash course" ($3.4n^3$ and $13n^2$ *microseconds*, respectively). Suppose that you connect adjacent scatter plot points from each experiment with a line to approximate their underlying "running time trend curves." Are the shapes of these curves the same as their estimates? Are the leading constants (i.e., 3.4 and 13) different?[2]

[1]Available at `https://web.engr.oregonstate.edu/~glencora/wiki/uploads/tcscrashcourse.pdf`.

[2]Questions marked with a ♣ are more open-ended, and will not be graded strictly on correctness. Instead, they will be graded on you clearly explaining your thought process.

Question 2. (Factoring and ChatGPT, 10 points.) In this question you will study the factoring problem and the effectiveness of ChatGPT (available at https://chat.openai.com/) at solving it.

Factoring Problem
Input: A positive integer $n = pq$, where p and q are prime.[3]
Output: The values p and q.
Example: Input: $n = 15$. Output: $p = 3, q = 5$.

a. (4 points.) Design and implement an algorithm for solving the factoring problem. (Do not use ChatGPT or other resources for this part.)

   (**Hint**: There are very sophisticated algorithms for factoring, but here you only need to implement a simple algorithm, which should just be a few lines of Python code.)

b. (2 points.) Use your algorithm from Part Item a to factor the number $n = 122581$, and give its prime factors. **You do not need to justify your answer for this part.**

c. (4 points, ♣.) Enter the prompt "`factor 122581 into primes`" into ChatGPT. Try several times using the "regenerate" option, variants of the prompt, and by giving ChatGPT feedback on its responses. Does it give the right answer?

d. (Extra credit: Automatic A in the class.) Factor

```
n = 22112825529529666435281085255026230927612089502470015394413748319128822941402
200198651272972656974659908590033003140005117074220456085927635795375718595429
298838958709229238491006703034124620545784566413664540684214361293017694020846
46391065875914794251435144458199
```

   into its two prime factors. You may use any tools or resources that you wish.[4]

Question 3. (Practice with asymptotics, 15 points.)

a. (5 points.) For each of the following pairs of functions $f(n), g(n)$, state whether $f(n) = O(g(n))$, $g(n) = O(f(n))$, both, or neither. **You do not need to justify your answers for this part.**

   i. $f(n) = 100n^2$, $g(n) = n^{\log_2 4}$.
   ii. $f(n) = n^3$, $g(n) = \sqrt{n}$.
   iii. $f(n) = 1.2023^n$, $g(n) = n^{2023}$.
   iv. $f(n) = n \cdot \log_2(n)$, $g(n) = n \cdot \log_{10}(n)$.
   v. $f(n) = n^2 \cdot |\sin(\pi n/2)|$, $g(n) = n$. (**Hint**: Plot $f(n)$.)

[3]Recall that an integer $p \geq 2$ is *prime* if its only divisors are 1 and itself. A more general version of the factoring problem considers factoring any integer $n \geq 1$. Here we are just considering the special case where the input n is the product of two primes.

[4]Much of the security of the internet is based on the assumption that the factoring problem is hard. See https://en.wikipedia.org/wiki/RSA_(cryptosystem). The number in this problem, called RSA-260, is an "RSA challenge number," and is the smallest such challenge number not yet factored. See https://en.wikipedia.org/wiki/RSA_Factoring_Challenge.

b. (3 points.) Recall that the formal definition of big-O says that $f(n) = O(g(n))$ if and only if there exist constants $c, n_0 > 0$ such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$. Using this definition, show that $5n + \sqrt{n} + 100 = O(n)$.

(**Hint**: Make sure to specify your choices of c and n_0.)

c. (2 points, ♣.) Would your solution to Part b still work if you had chosen c and n_0 to be larger? Smaller? Explain why.

d. (3 points.) Recall that if $\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$ then $f(n) = O(g(n))$. Using this fact, show that $100n^2 + 3n + 17 = O(4n^2 + 1000)$.

(**Hint**: Use L'Hopital's rule.)

e. (2 points, ♣.) Suppose that $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$ for some value $0 < c < \infty$. What does this imply about the big-O relationship(s) between $f(n)$ and $g(n)$?