CSc 345 — Analysis of Discrete Structures
Fall 2024 (McCann)

http://cs.arizona.edu/classes/cs345/fall24/

Homework #2

(95 points)

Due Date: Thursday, September 26th, 2024, at the beginning of class

Write complete, legible answers to each of the following questions. Show your work, when appropriate, for possible partial credit. This is not a group project; do your own work. *If you need help, remember that the TAs and I have office hours for just this eventuality, and you may also post general questions on Piazza.*

On the due date, by the start of class, submit a PDF containing your answers on gradescope.com. Be sure to assign pages to problems after you upload your PDF. (Need help? See "Submitting an Assignment" on https://help.gradescope.com/.) If you need to submit your solutions within the 24–hour late window, Gradescope will be configured to accept them. Solutions submitted after 3:30pm on the 27th will not be accepted. Want to be safe and submit your homework early? Please do! You can resubmit an updated PDF before the due date, if necessary.

1. [15 points] Consider this summation: $\sum_{i=x}^{y} i^2$.

   (a) Determine the equivalent closed–form expression of this summation. (In this context, 'closed–form' means 'not a summation of $y - x + 1$ terms.') Include a short explanation of how you created the expression.

   (b) Prove that your answer to (a) is correct.

2. [10] Shown below are two pseudocode algorithms. For each algorithm, use step–counting (as demonstrated in class) to produce a polynomial expression for the number of steps performed by the algorithm, in terms of the instance characteristic n. Show your augmented code (with the counter increments and loop iterations, like I did in class).

   (a)
   ```
   sum <-- 0
   product <-- 1
   for i from 1 through 3n do
       product <-- product * i
       sum <-- sum + product
   end for
   ```

   (b)
   ```
   sum <-- 0
   for i from 1 through n*n do
       for j from 1 through i do
           sum <-- sum + i * j
       end for
   end for
   ```

3. [5] Arrange the following functions in ascending order by rate of growth (that is, slowest-growing to fastest-growing).

   $\frac{n}{2}$ $\quad n^n \quad$ $\log_2(6n)$ $\quad 12 \quad$ $4^{\frac{n}{2}}$ $\quad \log_6(2n) \quad \sqrt{n}$

(Continued . . .)

4. [10] Shown below are two time complexity functions. Answer the following question for each of them. See Section 3.3 of the PDF version of Shaffer's text for some background.

   Imagine that an algorithm's execution efficiency is described by the function $E()$. You execute that algorithm on Computer A and discover that it takes s seconds to complete a problem of size n. You buy Computer B that is 32 times faster than Computer A. All else being equal, how large of a problem (as an expression in terms of n) can Computer B complete in the same amount of time (s seconds), using the same algorithm?

   (a) $E(n) = 4n^2$

   (b) $E(n) = 3^n$

5. [5] Using the definition of Big–O presented in class, show that $f(n) = 100$ is $O(1)$ and also $O(n^2)$. (Need the definition? See the slides of Sept. 12th.)

6. [5] What is the smallest integer value of c such that $f(n) = \sqrt{n}$ is $O(n^c)$? (We're only asking for the answer, but, as the assignment directions say, show your work for possible partial credit.)

7. [10] Write an $O(n)$ method (in your choice of Java or pseudocode) that accepts references to two doubly–linked lists of integers (both in ascending order) and returns a reference to a new doubly–linked list, also in ascending order, that contains all of the values of the two input lists. Include an explanation of why it is $O(n)$.

8. [10] For each of the pairs of functions given below, is $f(n) \in O(g(n))$, $f(n) \in \Omega(g(n))$, or $f(n) \in \Theta(g(n))$? Use limits to determine your answers (as presented in class, and also in Shaffer Section 3.4.5).

   (a) $f(n) = 5n,\ g(n) = \frac{n}{100}$

   (b) $f(n) = \log_2^2 n,\ g(n) = \log_2 n$ *(Note:* $\log_2^2 n$ *means* $(\log_2 n)^2$*.)*

9. [10] Consider a $\Theta(\log_2 n)$ version of the binary search algorithm that accepts a 1D array of integers (stored in ascending order) plus a search key value, and returns the index of the last occurrence of that key within the array ('last' meaning the occurrence with the largest array index). Create this algorithm, in your choice of Java or pseudocode, and include an explanation of why it is $\Theta(\log_2 n)$. *Be aware:* To achieve the required level of performance, linear search cannot be used.

10. [15] For each of the following recurrence relations, if the Master Theorem can be used, use it to determine its Big-Θ asymptotic solution. If the Master Theorem cannot be used, solve the recurrence (showing your work!) and then determine its Big-Θ asymptotic solution.

    (a) $F(n) = 4F(n/2) + 2n$, assuming that $F(1) = 1$ and n is a power of 2.

    (b) $G(i) = 3G(i-3) + k$, assuming that $G(0) = 0$ and i is a multiple of 3.

    (c) $H(j) = H(j/3) + c$, assuming that $H(1) = 3$ and j is a power of 3.