CSCE 411, Fall 2025: Homework 1

**Problem 0.** (-5 for leaving blank.) Write down all outside resources you consulted, and the names of any other students that you worked with in any way. By turning in this homework, you acknowledge that you have read and abide by all course policies on outside resources and collaboration as stated in the course syllabus. Use of Generative AI tools must be explicitly noted. *If you did not use outside resources, AI tools, or collaborators, state that explicitly.*  
**Problem 1.** Let BinSearch be a standard binary search algorithm that takes in a sorted array of integers, and an integer that is guaranteed to be contained somewhere in , and returns an index such that .

(a) Write pseudocode to implement this method. You may assume that is a power of 2. For your code, assume is 1-indexed (i.e., is the first element).

(b) Write down a recurrence relation for this algorithm.

*Answer.*

(a) Pseudocode is given below. Slight variations are also acceptable.

BinSearch

**if** **then**  
    return   
**if** **then**  
    return BinSearch  
**else**  
    return BinSearch  
(b) The recurrence relation is .

**Problem 2.** Consider the recurrence relations below. For each, there is some text that attempts to apply the Master Theorem (as shown at the end of the homework) to determine the runtime. In each case, state whether the analysis is correct, or whether there is an error in the application of the Master Theorem. If there is an error, state what the error is and what the correct answer should be.

(a) Recurrence: .

Candidate explanation: In this case, . Hence, for a sufficiently small , and therefore Case 1 applies and we know .

(b) Recurrence: .

Candidate explanation: In this case, . Meanwhile, . Hence, is polynomially larger than and we apply Case 3. The runtime is therefore .

(c) Recurrence: .

Candidate explanation: In this case, . Meanwhile, , which does not grow like a polynomial and is therefore neither polynomially larger nor polynomially smaller than . For this reason, the Master Theorem does not apply.

(d) .

Candidate explanation: In this case, . Meanwhile, . The comparison between and does not fit into any cases of the Master Theorem, so the theorem does not apply here.

*Answer.*

(a) This is incorrect. , so Case 2 applies, and the runtime is .

(b) This is incorrect. Although is asymptotically larger than , it is not polynomially larger than , so Case 3 does not apply. Cases 1 and 2 also don’t apply, so the Master Theorem below simply does not apply. Note furthermore that the candidate explanation did not check the additional condition .

(c) This is incorrect. Whether or not is a polynomial, the function is polynomially larger. Therefore, Case 1 applies and the runtime is .

(d) This is correct. The recurrence does not fit into any of the cases.

**Problem 3.** Say you have discovered a way to multiply matrices using multiplications and 30 additions.[[1]](#footnote-9)

(a) Write down the recurrence relation you could get for a new recursive matrix-matrix multiplication algorithm you could design by breaking up an matrix into blocks of size and using your approach for multiplying matrices. You may assume where is an integer.

(b) Write down an expression for the asymptotic runtime of this method in terms of and .

(c) What is the largest value for such that the asymptotic runtime of your new method is better than the runtime you get from Strassen’s algorithm.

*Answer.* Let be the runtime of the algorithm for .

(a) The recurrence relation you get is

since is the time for multiplying two blocks (which we do times), and is the time it would take to do 30 additions of matrices.

(b) Because , we know that , which means that for a small enough . Applying the Master Theorem, we know that the runtime for the new method is

(c) The largest value of such that is , so this is the answer since the runtime for Strassen’s method is .

**Master Theorem.** Let and be constants, let be a function, and let be defined on the nonnegative integers by the relation:

1. If for some constant , then .
2. If , then .
3. If for some constant , and if for some constant and all sufficiently large , then .

1. Assume that this works without relying on the fact that for . [↑](#footnote-ref-9)