Homework 2

(122 points)

Assigned: Wednesday, January 21, 2026
Due: Wednesday, February 4, 2026

Objective

The purpose of this assignment is to practice designing and implementing both iterative and recursive algorithms, and to analyze their time and space complexities by formulating and solving recurrence relations.

Part 1. Recursive vs. Iterative Algorithms (26 pts)

Given a sorted array (in ascending order) of arbitrary numbers, design and implement both recursive and iterative algorithms for binary search.

You must design and write the algorithms in formal pseudocode, starting with the following:

```
ALGORITHM BinarySearch (A[0..n-1], n, x)
INPUT: Array A of n floating-point numbers, and target x
OUTPUT: Index i such that A[i] = x, or -1 if x not found
```

For each algorithm, you are required to analyze both its *time* and *space* complexity. A formal mathematical analysis is not required; a clear explanation in your own words is sufficient.

For implementations, you must write code in `Java` using the following method signature:

```
static int binarySearch(double[] A, double x)
```

- Recursive Algorithm (8 pts)

  You may use a helper algorithm as defined below:

  ```
  ALGORITHM BinarySearchRec(A[0..n-1], x, left, right)
  ```

- Recursive Implementation (5 pts)
- Iterative Algorithm (8 pts)
- Iterative Implementation (5 pts)

Part 2. Recursive vs. Iterative Algorithms (40 pts)

```
ALGORITHM DoSomething(n)
INPUT: A positive integer n
OUTPUT: ???

IF n = 1
    return 1
ELSE
    return DoSomething(n-1) + n * n * n
```

Part 2.1 Understanding the Recursive Algorithm (8 pts)

Explain what the above algorithm does and specify the output it produces.

Part 2.2 Recurrence Relation (10 pts)

Set up the recurrence relation for the time complexity analysis of the above algorithm. To avoid common mistakes, you may find Chapter 2.4 of the textbook useful.

Part 2.3 Time Complexity (extra credit 10 pts)

Solve the recurrence relation using backward substitution, i.e., express it in closed form, and then provide the time complexity in Big-O notation. You must show all steps.

Note: if your answer to Part 2.2 is incorrect, no extra credit will be awarded, even if you solve the recurrence relation correctly.

Part 2.4 Space Complexity (2 pts)

Analyze the space complexity of the recursive algorithm in Big-O notation. You must justify your analysis; simply providing the final answer will not earn credit.

Part 2.5 Iterative Algorithm (20 pts)

Design and write an iterative algorithm in formal pseudocode that achieves the same goal as the recursive algorithm.

Analyze the *time* and *space* complexities of your iterative algorithm. You must justify your analysis; simply providing the final answer will not earn credit.

Part 3. Solve Recurrence Relations (56 pts)

Part 3.1 Substitution Method

Solve the following recurrence relations using backward substitution:

1. $T(n) = T(n-1) + n, T(1) = 1$ (8 pts)
2. $T(n) = 2T(n/2) + n, T(1) = 1$ (8 pts)

3. $T(n) = 4T(n/2) + n^2, T(1) = 1$ (8 pts)
4. $T(n) = 8T(n/2) + 1, T(1) = 1$ (8 pts)
5. $T(n) = 2T(n/2) + \sqrt{n}, T(1) = 1$ (extra credit 5 pts)

You must clearly show:

1. the exact closed form;
2. all detailed steps leading to the closed form;
3. the final complexity in Big-O notation.

Do not handwrite your answers; all responses must be typed.

Part 3.2 Master Method

Solve the following recurrence relations using the Master Theorem:

1. $T(n) = 2T(n/2) + n$ (4 pts)
2. $T(n) = 4T(n/2) + n^2$ (4 pts)
3. $T(n) = 8T(n/2) + 1$ (4 pts)
4. $T(n) = 2T(n/2) + \sqrt{n}$ (4 pts)
5. $T(n) = 9T(n/3) + n$ (4 pts)
6. $T(n) = T(2n/3) + 1$ (4 pts)

You may assume $T(1) = 1$ for all cases. Show all steps, including identifying a, b, k, and other relevant parameters.

What to Submit

All answers must be typed (no handwriting) and must show all detailed steps. Submit the entire homework as a single PDF file on Blackboard before the due date.