# CS 61B Spring 2024

# Final

Tuesday, May 7, 2024

Your Name: _____

Your SID: _____ Location: _____

SID of person to your left: _____ Right: _____

This exam is worth **100 points**, and consists of 8 questions. Questions are ordered by topic, not by difficulty.

| Question: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | Total |
|---|---|---|---|---|---|---|---|---|---|
| Points: | 9 | 22 | 4 | 7 | 5 | 16 | 22 | 15 | 100 |

- ◯ indicates only one circle should be filled in. ☐ indicates more than one box may be filled in. **Please fill in the shape completely.** If you change your response, **erase as completely as possible**.
- Anything you write that you ~~cross out~~ will not be graded.
- If you write multiple answers, your answer is ambiguous, or the bubble/checkbox is not entirely filled in, we will grade according to the worst interpretation.
- Unless otherwise specified, all data structures and algorithms behave according to their implementation in lecture, with no additional optimizations.
- If an implementation detail (e.g. tiebreaking scheme, linked list topology) is relevant, it will be explicitly noted in the question.

Coding questions:

- You may write at most one statement per blank and you may not use more blanks than provided.
- Your answer will be reformatted according to the 61B style guidelines. For example, any if-statement requires at least three lines for the purposes of determining line count.
- You may not use ternary operators, lambdas, streams, or multiple assignment.
- Unless otherwise specified, you may assume that everything on the reference card has been imported.

Write the statement below in the same handwriting you will use on the rest of the exam. Failure to do so may result in a voided exam.

> I have neither given nor received help on this exam (or quiz), and have rejected any attempt to cheat. If these answers are not my own work, I may be deducted up to 9,876,543,210 points.

_____

_____

_____

_____

_____

Signature: _____

# 1 Vroom (9 points)

For each of the following lines, write the result of each statement: Write "CE" if a compiler error is raised on that line, "RE" if a runtime error occurs on that line, "OK" if the line runs properly but doesn't print anything, and the printed result if the line runs properly and prints something. If a line errors, assume that the program continues to run as if that line did not exist. Each blank is worth 1 point.

```
1   public class TwoWheel {
2       public int topSpeed = 200;
3       private boolean counterSteer = true;
4       public void rideWith(TwoWheel t) { System.out.println("I'm TwoWheel"); }
5       public void printSteer() { System.out.println(counterSteer); }
6   }
7   public class Motorcycle extends TwoWheel {
8       public Motorcycle() { topSpeed = 400; }
9       public void rideWith(TwoWheel t) { System.out.println("I'm moto"); }
10  }
11  public class Bicycle extends TwoWheel {
12      public Bicycle() { topSpeed = 100; }
13      public void rideWith(TwoWheel t) { System.out.println("I'm bike"); }
14  }
```

```
TwoWheel scooter = new TwoWheel();
```
_____ (a)

```
Motorcycle kawi = new Motorcycle();
```
_____ (b)

```
Bicycle trek = new Bicycle();
```
_____ (c)

```
Motorcycle.rideWith(kawi);
```
_____ (d)

```
kawi.rideWith(trek);
```
_____ (e)

```
((TwoWheel) trek).rideWith(scooter);
```
_____ (f)

The following subparts are independent; the change in each part does not carry over to other parts.

(g) (1 point) If we change the signature on line 5 to `public static void printSteer()`, will the code still compile?

◯ Yes ◯ No

(h) (1 point) If we change line 3 to `private static boolean counterSteer = true`, will the code still compile?

◯ Yes ◯ No

(i) (1 point) If we change line 2 to `private int topSpeed = 200`, will the code still compile?

◯ Yes ◯ No

# 2 Polynomials (22 points)

An *Integer Polynomial* is an expression of the form

$$a_n x^n + \cdots + a_2 x^2 + a_1 x + a_0$$

where all $a_i$, $0 \leq i \leq n$, are integers. A *nonzero term* of a polynomial is a single $a_i x^i$ with $a_i \neq 0$. The *degree* of a polynomial is the largest exponent $n$ associated with a nonzero term, or 0 for the polynomial 0. Consider the `Polynomial` class below, which represents an integer polynomial as a linked list of terms:

```
public class Polynomial {
    private class Term {
        public int coef;
        public int exp;
        public Term next;

        public Term(int coef, int exp) {
            this.coef = coef;
            this.exp = exp;
        }
    }

    private Term terms;

    public Polynomial() {
        terms = new Term(0, -1);
    }

    public double eval(double input) {
        // Implemented below
    }

    public void addTerm(int coef, int exp) {
        // Implemented below
    }
}
```

Note the following:

- The special term $(0, -1)$ is used as a sentinel value, which is placed at the **end** of the linked list.
- The linked list contains only the nonzero terms of the polynomial, ordered from greatest exponent to least exponent. Thus, every polynomial has a unique representation.
- The linked-list topology is not circular; the sentinel term always has a `next` of `null`.

For example, the polynomial $-3x^{500} + 5x^3 + 4x + 7$ has four nonzero terms, and has degree 500; it is represented by the linked list $(-3, 500) \to (5, 3) \to (4, 1) \to (7, 0) \to (0, -1)$.

The polynomial 0 has no nonzero terms and has degree 0; it is represented by the linked list $(0, -1)$.

The `Polynomial` class, reprinted for your convenience:

```
public class Polynomial {
    private class Term {
        public int coef;
        public int exp;
        public Term next;

        public Term(int coef, int exp) { ... }
    }

    private Term terms;

    public Polynomial() {
        terms = new Term(0, -1);
    }
}
```

(a) (7 points) Implement the `eval` method, which receives a value `input` and returns the result of evaluating the polynomial when $x =$ `input`.

For example, the polynomial $5x^3 + 4x + 7$ evaluated at the input 2 would be $(5 \cdot 2^3) + (4 \cdot 2) + (7) = 55$. If instead, this polynomial is evaluated at 0.5, the result would be $(5 \cdot 0.5^3) + (4 \cdot 0.5) + (7) = 9.625$.

```
public double eval(double input) {
    double result = 0;

    _____1_____;

    while (_____2_____) {

        result += _____3_____ Math.pow(_____4_____, _____5_____);

        _____6_____;
    }

    _____7_____;
}
```

(b) (2 points) What is the asymptotic runtime of `eval`, in terms of $n$, the degree of the polynomial? Provide the most informative bound possible, in either $\Theta$, $\Omega$, or O notation.

(c) (11 points) Implement the `addTerm` method, which receives a coefficient and exponent representing a nonzero term, and updates the polynomial to include that term.

For example, the polynomial $5x^3 + 4x + 7$, after adding $2x^2$, would become the polynomial $5x^3 + 2x^2 + 4x + 7$, with the underlying linked list $(5, 3) \to (2, 2) \to (4, 1) \to (7, 0) \to (0, -1)$.

You may assume that `coef != 0`, `exp >= 0`, and that the polynomial does not already contain a nonzero term with the same exponent.

```
public void addTerm(int coef, int exp) {

    Term newTerm = _____1_____;

    if (_____2_____) {

        _____3_____;

        _____4_____;
        return;

    }
    Term curr = _____5_____;

    while (_____6_____) {

        _____7_____;
    }

    _____8_____;

    _____9_____;
}
```

(d) (2 points) What is the asymptotic runtime of `addTerm`, in terms of $n$, the degree of the **original** polynomial, and $k$, the exponent of the new term? Provide the most informative bound possible, in either $\Theta$, $\Omega$, or O notation.

# 3 Disjoint Sets (4 points)

(a) (1 point) True/False: The array below is a valid representation of a Weighted Quick Union.

| 1 | 0 | −1 | −1 | −1 |
|---|---|---|---|---|

◯ True ◯ False

(b) (1 point) True/False: The array below is a valid representation of a Weighted Quick Union.

| −1 | 2 | 3 | 4 | −4 |
|---|---|---|---|---|

◯ True ◯ False

(c) (1 point) You are given a mystery Disjoint Set implementation, and want to figure out if it is a **Quick Union**, or a **Weighted Quick Union**.

In order to determine this, you create a new instance of your Disjoint Set with 100 elements, and observe the values in the underlying array as you apply `union` operations. What is the minimum number of `union` operations needed to distinguish between these two disjoint set implementations?

◯ 1 ◯ 3

◯ 2 ◯ 4

(d) (1 point) You are given a mystery Disjoint Set implementation, and want to figure out if it is a **Weighted Quick Union**, or a **Weighted Quick Union with Path Compression**.

In order to determine this, you create a new instance of your Disjoint Set with 100 elements, and observe the values in the underlying array as you apply `union` operations. What is the minimum number of `union` operations needed to distinguish between these two disjoint set implementations?

◯ 1 ◯ 3

◯ 2 ◯ 4

# 4 Heaps (7 points)

For the following question, all heap implementations leave the 0th index empty. The empty index is denoted with a −.

(a) (1 point) Consider the following min-heap in array format: [−, 1, 9, 3, 10, 13, 4, 8].

What is the resulting array representation after inserting 7?

◯ [−, 1, 9, 3, 7, 13, 4, 8, 10]

◯ [−, 1, 3, 7, 9, 13, 4, 8, 10]

◯ [−, 1, 3, 7, 13, 9, 4, 8, 10]

◯ [−, 1, 7, 3, 9, 13, 4, 8, 10]

(b) (1 point) Consider the following min-heap in array format: [−, 1, 9, 3, 10, 13, 4, 8].

What is the resulting array representation after 2 remove operations?

◯ [−, 4, 9, 8, 10, 13]

◯ [−, 4, 8, 9, 10, 13]

◯ [−, 4, 8, 9, 13, 10]

◯ [−, 4, 9, 8, 13, 10]

(c) (2.5 points) Construct a max-heap with the values 1, 2, 3, 4, 5, 6, 7, such that the underlying array has 21 inversions.

(Note: This is the maximum possible number of inversions in a valid max-heap with 7 elements.)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| – | | | | | | | |

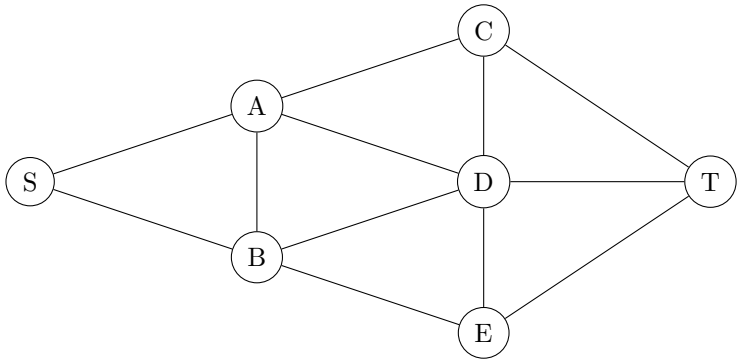
(d) (2.5 points) Construct a max-heap with the values 1, 2, 3, 4, 5, 6, 7, such that the underlying array has 12 inversions.

(Note: This is the minimum possible number of inversions in a valid max-heap with 7 elements.)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| – | | | | | | | |

# 5 In-Depth Analysis (5 points)

(a) (1 point) Select the correct DFS preorder for the graph below, starting at node S. Break ties prioritizing the node that is alphabetically first.




◯ S, A, B, D, C, T, E

◯ S, A, B, C, D, E, T

◯ S, A, C, T, D, B, E

(b) (2 points) A rooted binary tree of height at most 10 has the same **in-order** and **post-order** traversal.

What is the maximum number of nodes that could be in this tree?

Note: A tree with only the root has height 0.

(c) (2 points) A rooted binary tree of height at most 10 has the same **pre-order** and **post-order** traversal.

What is the maximum number of nodes that could be in this tree?

# 6 ddEei Sort (16 points)

Eddie is sorting a list of exam grades. Halfway through sorting them, he discovers a box of misplaced exams, and needs to integrate their grades into his sort!

Eddie starts off with the following initial list of grades, and decides to run insertion sort to sort them:

$$[28, 97, 44, 25, 11, 84]$$

Partway through the sorting algorithm, the list of grades has reached the following state:

$$[25, 28, 44, 97, 11, 84]$$

At this point, Eddie receives the following new grades: [32, 84, 28, 61].

(a) (2 points) To integrate the new grades into his list, Eddie appends the numbers to the end of his list, so that the list is now:

$$[25, 28, 44, 97, 11, 84, 32, 84, 28, 61]$$

Eddie then resumes running insertion sort from where he left off. What is the state of the list after **four additional** elements have been correctly inserted into their correct position?

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

(b) (4 points) Suppose that Eddie has completely sorted a list of $N$ items with insertion sort. At this point, he receives $P$ additional items, appends them to the end of the sorted list, and continues running insertion sort, starting from the first additional item.

What is the **additional** asymptotic runtime needed to sort the entire list of $N + P$ items? Do **not** include the time spent sorting the initial $N$ items.

Best Case:

$\Theta($ $)$

Worst Case:

$\Theta($ $)$

Eddie observes that insertion sort seems to work properly if new data is appended to the end of the list, regardless of when that new data arrives. Eddie is curious to know if any other sorting algorithms can similarly work properly if new data is added partway through the sorting algorithm.

For each of the following sorting algorithms, determine whether the sorting algorithm would still produce a correctly sorted list if new data is added in the specified manner. Unless otherwise specified, the new data may be added at any intermediate step of the sorting algorithm.

(c) (2 points) Selection sort

The new data is appended to the end of the list.

◯ Always produces a sorted list ◯ Does not always produce a sorted list

(d) (2 points) Heapsort (without in-place heapification)

The new data is inserted, one element at a time, into the heap.

◯ Always produces a sorted list ◯ Does not always produce a sorted list

(e) (2 points) Mergesort

The new data is sorted, then merged into any sublist immediately before a merge step.

◯ Always produces a sorted list ◯ Does not always produce a sorted list

(f) (2 points) MSD Radix sort

The new data is inserted into its corresponding subproblem list based on each element's most significant digits, immediately before a pass of counting sort.

◯ Always produces a sorted list ◯ Does not always produce a sorted list

(g) (2 points) Quicksort with a 3-scan partitioning scheme

The new data is appended to the end of the list immediately before a partitioning step.

◯ Always produces a sorted list ◯ Does not always produce a sorted list

# 7 Two Seemingly Unrelated Problems (22 points)

Problem I. Consider a connected[1], unweighted, directed acyclic graph with $N$ nodes.

(a) (4 points) What is the worst-case and best-case asymptotic growth of the longest path in the graph, in terms of $N$? Provide a $\Theta$ bound.

Best Case:

$\Theta($ $)$

Worst Case:

$\Theta($ $)$

(b) (4 points) Draw graphs that exhibit the best-case and worst-case longest path. Your examples must be clearly extendable to arbitrarily many nodes; for clarity, you may write in English exactly how to extend your example to arbitrarily many nodes.

Best Case:

Worst Case:

[1] Formally, a *weakly connected* directed graph, which is a directed graph whose underlying undirected graph (i.e. with arrow directions removed) is connected.

Problem II. Note: This question is inspired by a question written by *Ravi Fernando*.

Consider a new sorting algorithm: *Booksort* takes in a list of $N$ unique numbers from 1 to $N$ inclusive (representing, for example, books numbered 1 through $N$ on a bookshelf), and sorts the list in ascending order by the following process:

**Step 1:** Select a book that is not in the correct slot in the list; if this is not possible, the list is sorted.

**Step 2:** Place the book in its correct final location in the list, shifting all intermediate items by one. For this question, we will assume that this step can be accomplished in $\Theta(1)$ time.

**Step 3:** Repeat Steps 1 and 2 until the list is sorted.

For example, imagine we have five books in the following order:

| 2 | 4 | 3 | 5 | 1 |
|---|---|---|---|---|

Out of these books, book 3 is in its correct slot (and thus cannot be selected in Step 1), and all other books are in the wrong slot.

Suppose we select book 4. First, we take book 4 out of the bookshelf:

| 2 | | 3 | 5 | 1 | 4 |
|---|---|---|---|---|---|

Then, we shift books 3 and 5 to the left (in constant time):

| 2 | 3 | 5 | | 1 | 4 |
|---|---|---|---|---|---|

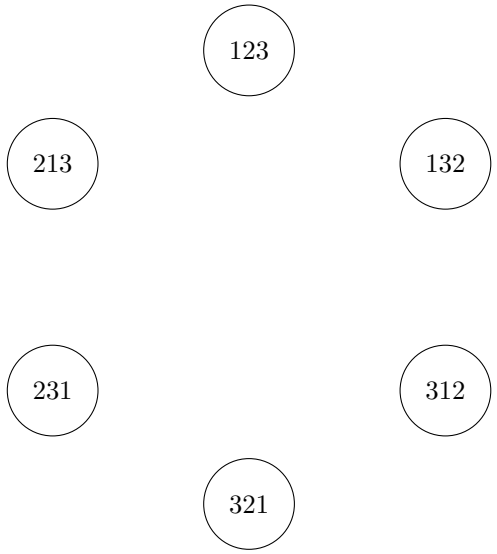
Finally, we insert 4 in its correct slot:

| 2 | 3 | 5 | 4 | 1 |
|---|---|---|---|---|

Note that by putting book 4 in its correct slot, we have also moved book 3 to an incorrect slot. The bookshelf can be fully sorted if, from the above position, we select book 1, then either book 5 or book 4.

(c) (2 points) A *state transition diagram* is a directed graph, where each node represents one of the possible orders that the books can be in. A directed edge from node A to node B is added if we can go from state A to state B in a single iteration of steps 1 and 2.

Draw the edges to complete the state transition diagram for $N = 3$ below. Draw at most one arrow between two states, even if multiple choices in step 1 yield the same transition.



Perhaps surprisingly, no matter how we choose the books in step 1, the algorithm is guaranteed to terminate in finite time. However, the way we choose the books in step 1 heavily affects the runtime of this algorithm.

(d) (2 points) What is the worst-case asymptotic runtime of this algorithm if, at each step 1, we always search for (in $\Theta(N)$ time) and select the **smallest** book that is not in its correct slot? Provide a $\Theta$ bound.

$\Theta($ $)$

For the next two parts, assume that in step 1, we always select (in $\Theta(1)$ time) the leftmost book that is not in its correct slot. As before, step 2 takes $\Theta(1)$ time.

(e) (4 points) Find as tight an $\Omega$ bound as possible for the worst-case runtime of this sorting algorithm.

Hint: Consider the initial sequence $2, 3, 4, 5, \ldots, N-1, N, 1$.

$\Omega($ $)$

(f) (6 points) Briefly explain why this sorting algorithm runs in worst-case $O(N!)$ runtime. You may assume (without explanation) that this algorithm is guaranteed to terminate in finite time.

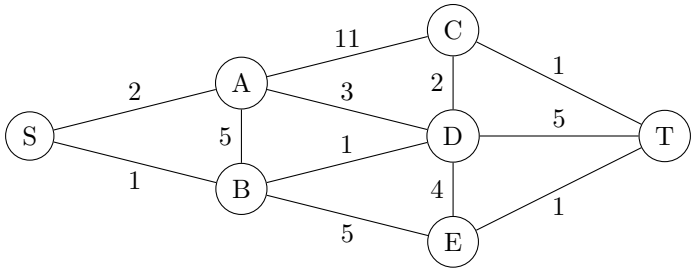
Hint: Apply Problem I to the state transition diagram.

# 8 Lost in the Ceryneian H.I.N.D. (15 points)

After being repeatedly hacked in past exams, you've finally tracked down the hackers responsible as members of the Ceryneian H.I.N.D. With the assistance of an investigator (codenamed Heracles), you have managed to infiltrate their internal network. Help Heracles catch the Ceryneian H.I.N.D.

The internal network is a weighted undirected graph, with start vertex S and target vertex T. You want to **delete one edge** from the graph, such that when deleted, the shortest path length from S to T is **increased** by as much as possible.

For example, consider the graph below:




Before deleting an edge, the shortest path from S to T has length 5. If we delete edge BS, the shortest path from S to T will instead have length 8; deleting any other edge will result in a shortest path from S to T that is shorter than 8. Thus, we should delete edge BS.

Design an algorithm (in English or pseudocode) that, given a graph, returns the edge that should be deleted. If multiple edges yield the same maximal shortest-path length, your algorithm may return any of them. You may assume that:

- All edges have positive integer edge weights.
- $S \neq T$
- The input graph is connected, and cannot be disconnected by removing any one edge.
- You have access to a working black-box implementation of Dijkstra's algorithm, which runs in $\Theta(E \log(V))$ time.

For credit, your algorithm must run in $O(VE \log(V))$ time, where V and E are the number of vertices and edges in the graph, respectively.

# Extra Answer Space for Questions 7-8

Do not tear this page off (even if you leave it blank).

If you need more space for your answers to Question 7(f) and/or Question 8, you may use this page. On this page, you must clearly indicate which parts are associated with 7(f) and/or 8.

If you want us to grade anything on this page, **you must draw an arrow (→)** at the end of the original box to tell us that your answer continues on this page.

**Nothing on this page is worth any points. Do not tear this page off.**

# 9 Crowdsourced Literature

**(0 Points)**

As a class, create a meaningful poem. You may write up to three words in the box below. After the exam, course staff will attempt to combine all valid fragments into a single poem.

# 10 Feedback

**(0 Points)**

Leave any feedback, comments, concerns, or more drawings below!