The University of Michigan
Electrical Engineering & Computer Science
EECS 281: Data Structures and Algorithms
Winter 2024

FINAL EXAM
KEY 1

Thursday, April 25, 2024
8:00AM – 10:00AM

Uniqname: _____ **Student ID:** _____

Name: _____

Uniqname of person to your left: _____

Uniqname of person to your right: _____

Honor Pledge:
"I have neither given nor received unauthorized aid on this examination,
nor have I concealed any violations of the Honor Code."

Signature: _____

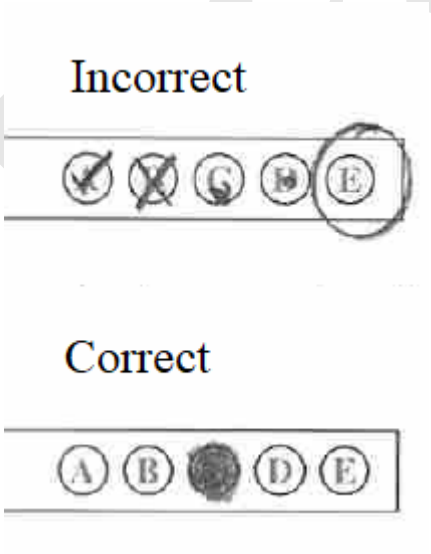
INSTRUCTIONS:

- The exam is closed book and notes except for one 8.5"x11" sheet of handwritten notes (both sides). All electronic device use is prohibited during the exam (calculators, phones, laptops, etc.).
- Print your name, student ID number, and uniqname **LEGIBLY**. Sign the Honor Pledge; this pledge applies to both the written and multiple choice sections. Your exam will not be graded without your signature.
- Record the **UNIQNAME** of the students to both your left and right. Write down `nullptr` if you are at the end of a row and there is no one on a given side.
- Record your **UNIQNAME** at the top of each odd-numbered page in the space provided. This will allow us to identify your exam if pages become separated during grading.
- This exam booklet contains **BOTH** the multiple choice and written portions of the exam. Instructions for each portion are provided at the beginning of their respective sections.
- **DO NOT** detach any pages from this booklet.
- There should be 22 pages in this book; if you are missing any pages, please let an instructor know.

Multiple Choice Portion [60 points]

MULTIPLE CHOICE INSTRUCTIONS:

- Select only **ONE** answer per multiple choice question.
- There are 24 multiple choice questions worth 2.5 points each.
- Record your choices for all multiple choice questions using the circles in the boxed area next to each question. Use a number 2 pencil, not a pen (so that you may change your answer). Fill the circle completely. There is no partial credit for multiple-choice questions. Make sure you bubble in the correct letter. See the example below for how to select your answer.



- There is no penalty for wrong answers: it is to your benefit to record an answer to each multiple-choice question, even if you're not sure what the correct answer is.
- The questions vary in difficulty — try not to spend too much time on any one question. Use your time wisely.
- When asked about memory complexity, consider all possible sources (stack and heap).

Record your answers in the bubbles next to each question.

1. Collision Resolution

Ⓐ Ⓑ Ⓒ Ⓓ

You are given a hash table with size $M = 10$ and hash function $H(n) = (2n + 3) \mod M$. Collisions are resolved using **quadratic** probing. What would the table look like (with an X denoting an empty location) after inserting the following elements in the order given?

4, 10, 12, 7, 2

A) [X, 4, X, 10, X, X, X, 12, 7, 2]

B) [X, 4, 2, 10, X, X, X, 12, 7, X]

C) [X, 4, X, 10, X, X, 2, 12, 7, X]

D) [X, 4, X, 10, X, X, 2, 12, X, 7]

2. Hash Table Applications

Ⓐ Ⓑ Ⓒ Ⓓ

In which one of the following situations would you want to use a hash table?

A) You have a classroom full of students each with a 10-digit ID number, and you want to find them by their ID number.

B) You want to find the highest priority thread to execute, each with its own assigned priority.

C) You have a rectangular floor composed of square tiles, and you want to know the color of a tile given a set of coordinates.

D) You are given a set of names, and want to print them out in alphabetical order.

Record your answers in the bubbles next to each question.

3. Tree ADT

Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

What does the following function return?

```
1       int count(Node* curNode) {
2         if (curNode == nullptr)
3           return 0;
4         else if (curNode->parent == nullptr)
5           return 1 + count(curNode->left) + count(curNode->right);
6         else if (curNode->left == nullptr && curNode->right == nullptr)
7           return 1;
8         else
9           return count(curNode->left) + count(curNode->right);
10      } // count()
```

A) The number of nodes in the tree

B) The number of internal nodes

C) The number of external (leaf) nodes

D) The number of external (leaf) nodes plus the root

E) The depth of the tree

4. *baugh Tree

Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

Suppose there existed something called a *baugh-Tree. A *baugh-Tree can be implemented similar to the **array**-based binary tree covered in lecture, except that its internal nodes can have up to six children. What are the best- and worst-case **space** complexities, respectively, of a *baugh-Tree with n nodes?

A) $\Theta(1)$, $\Theta(6n)$

B) $\Theta(n)$, $\Theta(2^n)$

C) $\Theta(2^n)$, $\Theta(6^n)$

D) $\Theta(n)$, $\Theta(6^n)$

E) $\Theta(6^n)$, $\Theta(6^n)$

Record your answers in the bubbles next to each question.

5. Tree Traversal

Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

Given the following preorder and inorder traversals of a binary tree, what is the postorder traversal of this same binary tree?

- Preorder: 11, 19, 31, 7, 10
- Inorder: 31, 19, 11, 10, 7

A) 11, 19, 7, 31, 10

B) 31, 19, 10, 7, 11

C) 19, 31, 10, 7, 11

D) 31, 10, 7, 19, 11

E) 10, 7, 31, 19, 11

6. Binary Search Tree Complexity

Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

Suppose you have a binary search tree, where h represents the tree's height and n represents the number of nodes. What is the worst-case complexity of a search operation on this tree in terms of h and/or n?

A) $\Theta(n \log h)$

B) $\Theta(n^2)$

C) $\Theta(\log n)$

D) $\Theta(h)$

E) $\Theta(n \log_n h)$

Record your answers in the bubbles next to each question.

7. Binary Search Tree vs. Binary Tree

Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

Given **pointer**-based representations of both, what **can** be said about a binary search tree that **cannot** be said about a binary tree?

A) The inorder traversal of a tree is always a sorted list.

B) The worst-case space complexity of a tree is $O(2^n)$.

C) A tree balances itself when a single branch becomes uneven.

D) The worst-case time complexity of inserting an element is $O(n)$.

E) It is always a complete binary tree.

8. Inorder Successor

Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

Suppose that in a binary search tree, when a node with two children is deleted, it is replaced by its inorder successor. Given a pointer to the node to be deleted, what is the time complexity of finding the inorder successor in the average case, if the tree contains n nodes?

A) $\Theta(\log n)$

B) $\Theta(n)$

C) $\Theta(n \log n)$

D) $\Theta(n^2)$

E) $\Theta(n!)$

Record your answers in the bubbles next to each question.

9. AVL Trees Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

Which of the following statements about AVL trees is **TRUE**?

A) Optimally re-balancing an AVL tree after a deletion can take more than $O(1)$ time.

B) For an insert operation, up to three rotations are needed.

C) Searching an AVL tree can take longer than $O(\log n)$.

D) Given an n-node binary search tree that has never been balanced, the BST can be made into a balanced AVL tree using $O(\log n)$ rotations.

E) A tree is AVL balanced if and only if the absolute difference in heights of the root's children is less than 2.

10. Sorting Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

Given an arbitrary array of data elements, we can use an AVL tree to sort them. What is the complexity of a sorting algorithm implemented using an AVL tree?

A) $\Theta(n)$ best case, $\Theta(n \log n)$ worst case

B) $\Theta(n \log n)$ best case, $\Theta(n^2)$ worst case

C) $\Theta(n)$ in all cases

D) $\Theta(n \log n)$ in all cases

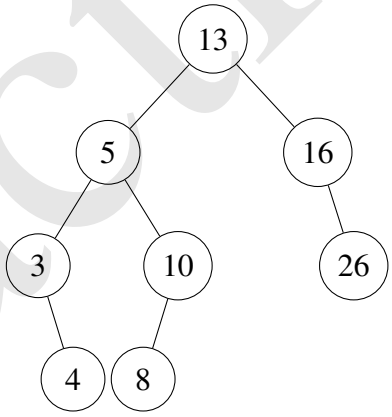
E) None of the above

Record your answers in the bubbles next to each question.

11. Insertion/Deletion

Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

Given the following AVL tree:



Perform the following operations:

1. Delete 13 (replace with its inorder successor)
2. Insert 9

What are the left and right children of "16" after performing the operations?

A) 8, 26

B) 9, 26

C) 10, 26

D) `nullptr`, 26

E) `nullptr`, `nullptr`

Record your answers in the bubbles next to each question.

12. Algorithm Families – Password Trouble

Ⓐ Ⓑ Ⓒ Ⓓ

You have forgotten your Wolverine Access password. Unfortunately, you used a random generator to create your password, so you have no idea what the password could be. The only feedback you get is whether what you typed is right or wrong. Which algorithm family would be best suited for finding your password?

A) Branch and Bound

B) Dynamic Programming

C) Divide and Conquer

D) Brute Force

13. Graphs

Ⓐ Ⓑ Ⓒ Ⓓ

Use the adjacency matrix given below. Starting from V1, determine a possible DFS sequence.

	V1	**V2**	**V3**	**V4**	**V5**	**V6**
V1	0	1	1	0	1	0
V2	1	0	0	1	0	0
V3	1	0	0	0	1	1
V4	0	1	0	0	1	0
V5	1	0	1	1	0	1
V6	0	0	1	0	1	0

Note: the order that adjacent vertices is visited is unspecified, you must determine which order is possible.

A) V1, V2, V3, V4, V5, V6

B) V1, V2, V4, V5, V6, V3

C) V1, V3, V5, V2, V4, V6

D) V1, V3, V4, V6, V5, V2

Record your answers in the bubbles next to each question.

14. MST

Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

Given the distance matrix below, what is the total weight of its minimum spanning tree?

$$\begin{bmatrix} 0 & 4 & 10 & 3 & 2 \\ 4 & 0 & 9 & 5 & 6 \\ 10 & 9 & 0 & 8 & 7 \\ 3 & 5 & 8 & 0 & 1 \\ 2 & 6 & 7 & 1 & 0 \end{bmatrix}$$

A) 13

B) 14

C) 15

D) 16

E) 18

15. Algorithm Families – Cookies

Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

As you leave the dining hall, you decide to grab some cookies to go. You have conveniently brought some containers for storing cookies. Dining hall cookies come in three sizes: 3 cm, 4 cm, and 5 cm (in radius). All cookies have the same thickness. Your containers are cylindrical, so you can only store cookies by stacking them on top of each other, and each container has a radius of 5 cm or less. What is the best method for taking cookies in order to maximize the total volume of cookies taken?

A) Brute Force

B) Greedy

C) Dynamic Programming

D) Backtracking

E) Branch and Bound

Record your answers in the bubbles next to each question.

NOTE: Both questions on this page use the following information.

Suppose you are given a maze represented as a graph with a start node, an end node, and a set of "black" (inaccessible) and "white" (accessible) nodes, with edges between them. The goal is to find a simple path from the start to end, going through only accessible nodes.

NOTE: This actual question is slightly poor because it's under-specified. Is the graph sparse or dense? Do we have an adjacency matrix or adjacency list? We should make the same assumptions for both questions. Let's assume worst case, which means approaching completely connected, which implies an adjacency matrix. Let's also assume we don't write a terrible solution, and never visit a node more than once.

16. Backtracking

Ⓐ Ⓑ Ⓒ Ⓓ

Which of the following represents the tightest-bound complexity of solving this problem using a recursive backtracking algorithm, where n is the number of nodes?

A) $\Theta(n)$

B) $\Theta(n^2)$

C) $\Theta(n^3)$

D) $\Theta(n!)$

17. Searching

Ⓐ Ⓑ Ⓒ Ⓓ

Which of the following represents the tightest-bound complexity of solving this problem using BFS, where n is the number of nodes?

A) $\Theta(n)$

B) $\Theta(n^2)$

C) $\Theta(n^3)$

D) $\Theta(n!)$

Record your answers in the bubbles next to each question.

18. Computing Sequence

Ⓐ Ⓑ Ⓒ Ⓓ

Consider the following recurrence relation:

- $F(0, k) = k$
- $F(n, k) = F(n-1, k+1) + F(n-1, k)$

Using dynamic programming, what is the best time complexity for computing $F(n, n)$?

A) $\Theta(\log n)$

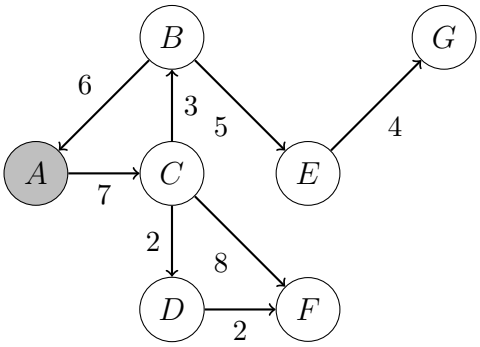
B) $\Theta(n)$

C) $\Theta(n^2)$

D) $\Theta(2^n)$

19. Dijkstra's Shortest Path

Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ



Starting at vertex A and trying to find the shortest path to G, which of the following progressions shows how the vertices are added to our set?

A) A, B, D, F, B, E, G

B) A, C, D, B, F, E, G

C) A, C, B, D, F, E, G

D) A, C, D, B, E, F, G

E) A, C, B, E, G

Record your answers in the bubbles next to each question.

20. Algorithm Families – Procrastination

Ⓐ Ⓑ Ⓒ Ⓓ Ⓔ

Your programming project is due in 4 hours, and you haven't even started! You have a few options: writing a test case gives you 2 points and takes between 10 and 15 minutes, writing a function gives you 7 points and takes between 30 and 50 minutes, working on the driver gives you 20 points and takes 2 hours. There are 6 functions and 10 test cases to write, and the times vary based on the specific test case or function. Given that starting early is no longer an option, what is the best approach for optimizing your project score?

A) Brute Force

B) Greedy

C) Dynamic Programming

D) Backtracking

E) Branch and Bound

21. Counting Paths

Ⓐ Ⓑ Ⓒ Ⓓ

In a certain game played on a grid, a game piece can either move 1 cell to the **right**, 1 cell **up**, or 1 cell **diagonally** (up and right) on a turn. The bottom-left corner is location (0, 0). For example, from cell (0, 0) to cell (2, 1) there are 5 paths: RRU, RUR, URR, DR, and RD (U = up, R = right, and D = diagonal). Using dynamic programming, what is the time complexity of counting the number of paths from $(0, 0)$ to (x, y)?

A) $\Theta(x + y)$

B) $\Theta(xy)$

C) $\Theta(x^2y^2)$

D) $\Theta(3^{x+y})$

Record your answers in the bubbles next to each question.

22. Complexity and Preprocessing Ⓐ Ⓑ Ⓒ Ⓓ

Suppose you're given an array such as {40, 20, 10, 30} and allowed to do $\Theta(n)$ preprocessing time with $\Theta(n)$ memory. After the preprocessing step, you will receive q queries asking for the sum of the elements in some interval, such as [1, 4) whose sum in the above example is 60. What is the worst-case **time** complexity of the best algorithm executing q queries (**not** including the time to do preprocessing)?

A) $O(qn)$

B) $O(q \log n)$

C) $O(qn^2)$

D) $O(q)$

Record your answers in the bubbles next to each question.

23. Dijkstra's Algorithm

Ⓐ Ⓑ Ⓒ Ⓓ

Which of the following is **FALSE** about Dijkstra's algorithm?

A) Dijkstra's algorithm uses a Greedy approach to solving its problem.

B) Dijkstra's algorithm cannot be used on an input graph with negative edge weights.

C) Dijkstra's algorithm only finds the shortest path between two input vertices.

D) Dijkstra's algorithm is capable of achieving a time complexity $O(|E| \log |V|)$.

24. Greedy Algorithms

Ⓐ Ⓑ Ⓒ Ⓓ

You are writing room scheduling software for the University. Of course, everyone's favorite room is 1670 BBB, so all instructors want to hold their classes there. Each of the classes has a firm start and end time. Your algorithm needs to find the maximum number of classes that can be held in the room without any conflicts between classes. Which of the following algorithms produces the optimal solution?

Note: classes can start any time and have any length.

Hint: create some examples.

A) Greedily choose classes with the earliest start time.

B) Greedily choose classes with the earliest end time.

C) Greedily choose classes with the fewest conflicts with other classes.

D) Greedily choose classes that take the shortest time (`endTime - startTime`).

Written Portion [40 points]

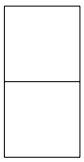
WRITTEN PORTION INSTRUCTIONS:

- Please write your code legibly in the space provided. Solutions that are difficult to read may receive fewer points.
- Use the back of the question page to work out solutions for each problem and then rewrite your final answer in the space provided.
- The directions for each problem will specify which STL algorithms/functions may be used in your solution.
- Solutions that exceed the allowed line count will lose one (1) point per line over the limit.
- Partial credit will be awarded for partial solutions.
- Errors in program logic or syntax may result in a reduced score.
- Be sure to consider all possible sources of memory (stack and heap) and time complexity.

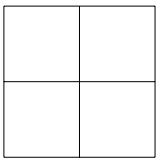
---

25. Dynamic Programming: Tiling [15 Points]

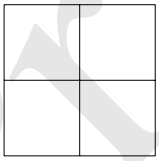
You are given a 2 x n board ($n > 0$) You are interested in the number of **distinct** ways to tile the given board using tiles of dimensions (2 x 1), (1 x 2) and (2 x 2) as shown below:

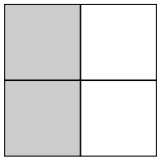

2 x 1

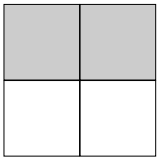

1 x 2


2 x 2

For example, for a 2 x 2 (i.e., $n = 2$) board, there are 3 ways:


Using one 2x2 tile


Using two 2x1 tiles


Using two 1x2 tiles

Requirements: Your solution must be $O(n)$ time. You may use up to $O(n)$ auxiliary space.

Implementation: Use the back of this page as a working area, then rewrite **neatly** on the front. Limit: 12 lines of code (points deducted if longer). You may use any C, C++, or STL function, algorithm, or data structure that you wish.

```
int number_of_tilings(int n) {
```

This page is intentionally left blank.
You may use this page as working space.

26. Programming: Searching in a Tree [10 Points]

A *set* is a data structure that can efficiently (less than $O(n)$ time) see if it contains a given element. Using the following definition of a binary tree node:

```
struct Node {
  int val;
  Node *left;
  Node *right;
};
```

Implement the `exists()` function on a BST-based set.

Requirements: In the average case, your solution must be $O(\log n)$, and you may use up to $O(\log n)$ auxiliary space.

Implementation: Use the back of this page as a working area, then rewrite **neatly** on the front. Limit: 15 lines of code (points deducted if longer). You may use any C, C++, or STL function/algorithm that you wish.

```
bool exists(Node *node, int val) {
```

This page is intentionally left blank.
You may use this page as working space.

27. Programming: Searching in a Hash Table [15 Points]

A *set* is a data structure that can efficiently (less than $O(n)$ time) see if it contains a given element. Given the following definition of a hash table structure:

```
enum class Status { Empty, Occupied, Deleted };
struct HashTable {
  std::vector<int> buckets;
  std::vector<Status> status;
};
```

Suppose this hash table implements the quadratic probing mechanism for collision resolution. When hashing an integer value `val`, the hash function is simply `val % M`, where `M` represents the size of the hash table. Implement the `exists()` function for a set based on a hash table.

Requirements: Your solution must implement quadratic probing. Your solution must be $O(n)$ time. You may use $O(1)$ auxiliary space.

Implementation: Use the back of this page as a working area, then rewrite **neatly** on the front. Limit: 16 lines of code (points deducted if longer). You may use any C, C++, or STL function/algorithm that you wish.

```
bool exists(const HashTable &tbl, int val) {
```

This page is intentionally left blank.
You may use this page as working space.