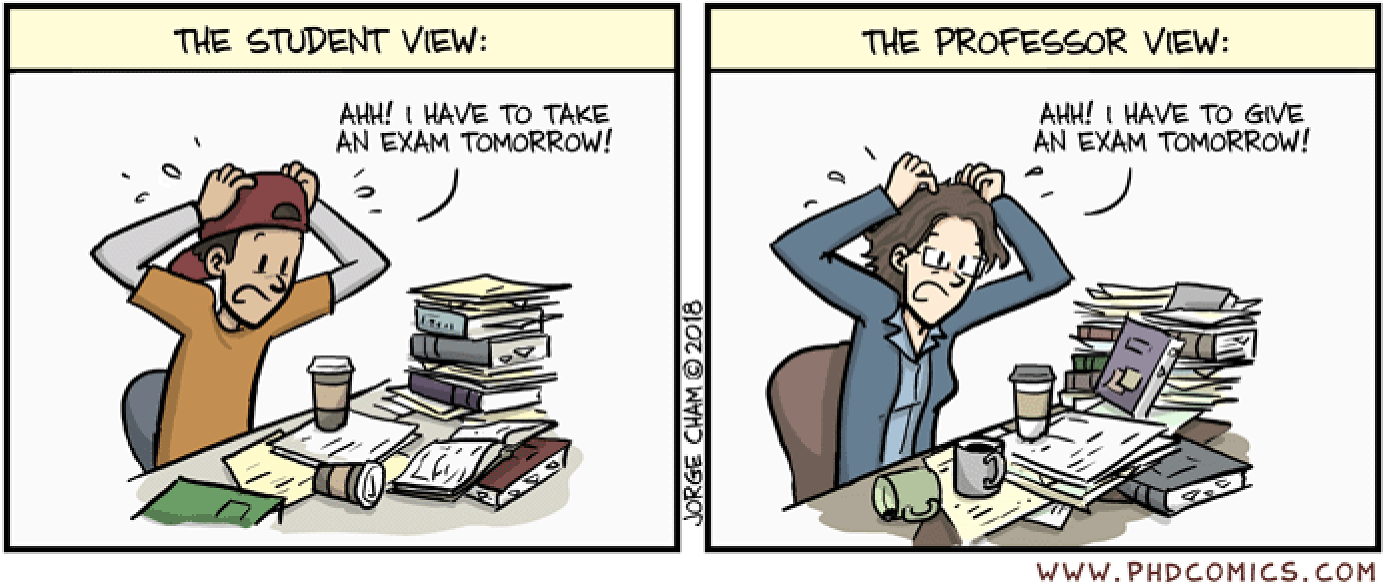
Lecture 14
Midterm Exam Review

EECS 281: Data Structures & Algorithms

Good News!


THE STUDENT VIEW:
AHH! I HAVE TO TAKE AN EXAM TOMORROW!
JORGE CHAM © 2018
THE PROFESSOR VIEW:
AHH! I HAVE TO GIVE AN EXAM TOMORROW!
WWW.PHDCOMICS.COM

http://phdcomics.com/comics/archive.php?comicid=2022

When/Where

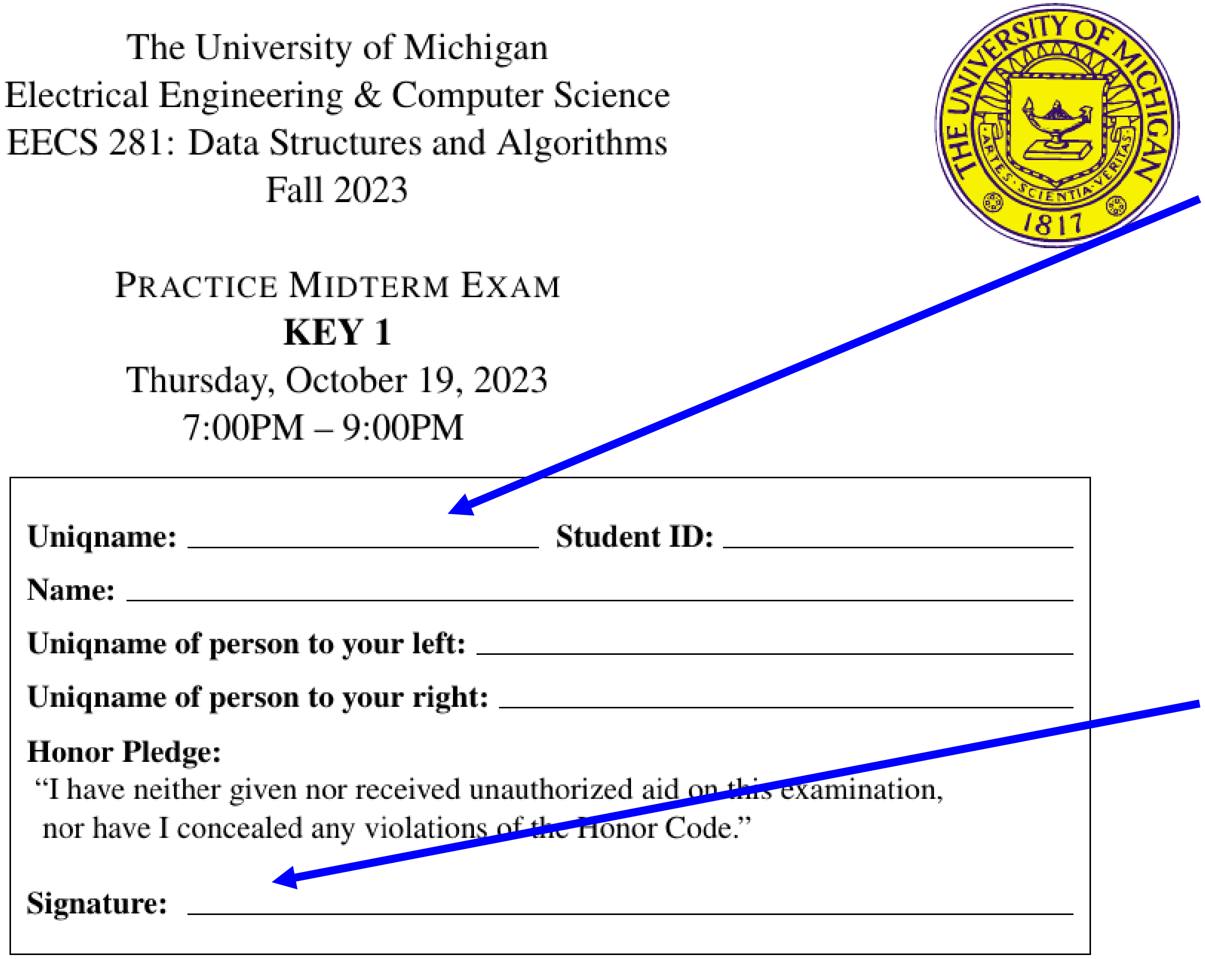
- When: Thursday October 19
  - 7:00pm – 9:00pm
- Locations on Piazza
  - You will be assigned a room based on your uniqname
- Accommodation (extended time) will be emailed directly from eecs281admin
  - 3hr for 1.5x time
  - 4hr for 2x time

Policies

- Closed book and closed notes
- One “cheat sheet”, limited to 8.5"x11", (both sides), with your name on it
  - – Writing it by hand will make you much better prepared
- No calculators or electronics of any kind
- Engineering Honor Code applies

Don't forget!

Bring your Mcard with you!



The University of Michigan
Electrical Engineering & Computer Science
EECS 281: Data Structures and Algorithms
Fall 2023

PRACTICE MIDTERM EXAM
KEY 1
Thursday, October 19, 2023
7:00PM – 9:00PM

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: ____

Record your NAME, Uniqname and Student ID# LEGIBLY!

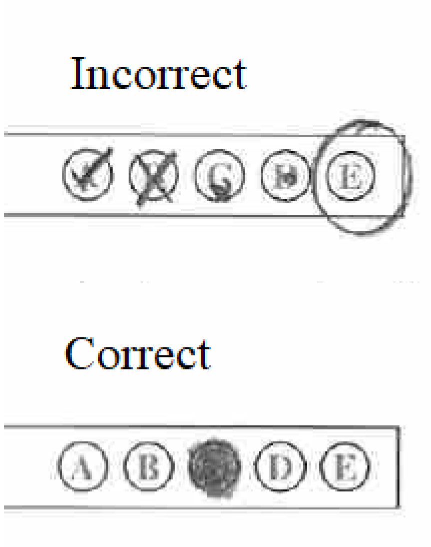
SIGN THE HONOR PLEDGE!

Multiple Choice Portion

- 24 questions, 2.5 points each
- 4-5 possible answers per question
- No deduction for being wrong
  - Make sure to answer all 24 questions
  - **ONE** answer per question
- **DO NOT wait until after time is called to <u>BUBBLE</u> in your answers**
  - **Do not just circle the letters to the left**

Filling in Bubbles

- Added to the instructions on the practice exam and actual exam
- **DO NOT** just circle the letters next to the answers, **FILL IN THE BUBBLES**



NOTE

- Bring a #2 pencil, or #2 lead for mechanical pencils (also listed as “HB”)
  - #3 pencils are too hard, and don’t scan well
- Odd-numbered pages have room at the top to write your uniqname
  - This is a backup in case pages become separated between collecting and scanning

Study Materials

- Practice exam posted on Canvas
  - Answers auto-reveal after last lecture
- Lecture slides and recordings
- In-class exercises
- Lab materials
- Projects
- Study group

Topics

- Everything we have covered so far, especially:
- Complexity analysis, including recurrences
- Contiguous (array) versus linked containers
- Stacks, queues and priority queues
- Binary heap (**not** pairing) and Heap Sort
- Elementary, Quick ~~and Merge~~ sorts
- ~~Strings and sequences~~

Answering Coding Questions

- If you decide you want a helper function, write it below the “given” function
- If you need a structure, write that inside the “given” function, below it, on the right, etc.
  - Some coding problems given in some semesters can ONLY be solved if you create a structure (or use a `pair<>`)
- Make it legible

Coding Questions – Lines

- How many lines of code is this?

```
if (x > 0) result = 0;
```

- 2 lines of code, same as this:

```
if (x > 0)
    result = 0;
```

Coding Questions – Lines

- How many lines of code is this?

```
if (x > 0) {
    result = 0;
    return result;
} // if
```

- 3 lines of code: the closing curly brace never counts as a "line of code"

Coding Questions – Lines

- How many lines of code is this?

```
if (x > 0)
    result = 0;
else
    result = x;
```

- 4 lines of code, the `else` statement counts as a line
- One line with ternary operator:

```
result = (x > 0) ? 0 : x;
```

Coding – Container of `struct`

- Once you create a structure, how can you easily add a member of that structure to a container? (OBTW: line count = 5)

```
struct WordCount {
    string word;
    int count;
};

vector<WordCount> vwc;
vwc.push_back({ "abc", 1});
```

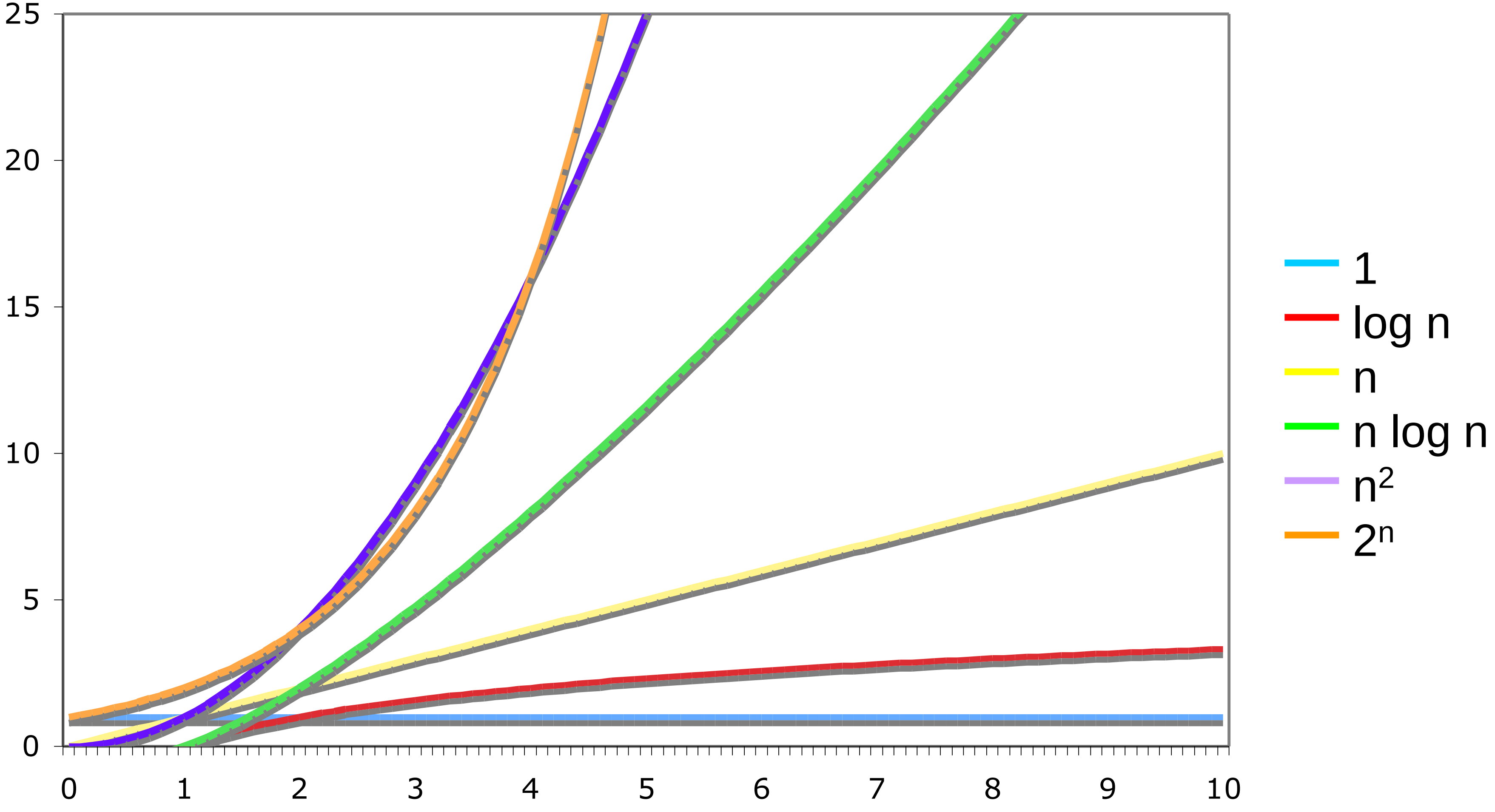
Coding Questions – Libraries

- Each coding question will tell you what you can or cannot use from the C and STL libraries
- The function header that we give you will not be part of the line limit
  - If you add a struct or helper function, those lines will count
  - If this is a reasonable way to solve the question, it is already factored in the line limit

Coding Questions – Integers

- For loop variables, use whatever type makes sense (`size_t`, `int`, etc.)
  - You don’t have to worry about implicit conversions on loop variables
- If we pass a `vector<int>` to your function and you need to keep a copy of one or more of those values, use an `int` variable, or a container of `int`
  - Stay consistent with data

Complexity Analysis



What is the complexity? Θ(...)

```
int* bsearch (int* lo, int* hi, int val) {
  while (hi >= lo) {
    int* mid = lo + (hi - lo) / 2;
      if (*mid < val) lo = mid + 1;
      else if (*mid > val) hi = mid - 1;
      else return mid;
  } // while
  return nullptr;
} // bsearch()
```

```
void f(int *out, const int *in, int size) {
  for (int i = 0; i < size; ++i) {
    out[i] = 1;
    for (int j = 0; j < size; ++j) {
      if (i == j)
        continue;
      out[i] *= in[j];
    } // for
  } // for
} // f()
```

What is the complexity? Θ(...)

- Write the recurrence relation
- Solve

```
void merge_sort(Item a[], int left, int right) {
  if (right <= left)
    return;
  int mid = left + (left - right) / 2;
  merge_sort(a, left, mid);
  merge_sort(a, mid, right);
  merge(a, left, mid, right);
} // merge_sort()
```

Containers

- What is the **best** container if it will be used primarily to locate objects within it using binary search?
- What is the **best** container if new objects will often be added immediately before specific existing objects?
- What is the **best** container if you must store a small number of very large objects. Memory is scarce and the most important consideration is to store as many of these objects as possible in the available space?

- Options: singly-linked list, doubly-linked list, vector
- Also: WHY?

Containers

- What is the **worst** container if you must store a large number of one byte items and memory is the scarcest resource?
- What is the **worst** container if you will frequently insert new items anywhere within the structure?
- What is the **worst** container if you will frequently insert new items at the beginning of the structure?

- Options: singly-linked list, doubly-linked list, vector
- Also: WHY?

Stacks and queues

- Implement a queue using two stacks. Given the class below, write the `pop()` function.

```
class MyQueue {
  stack<int> s1, s2;
public:
  void push(int num) {
    s1.push(num);
  } // push()
  void pop();
  int front();
}; // MyQueue
```

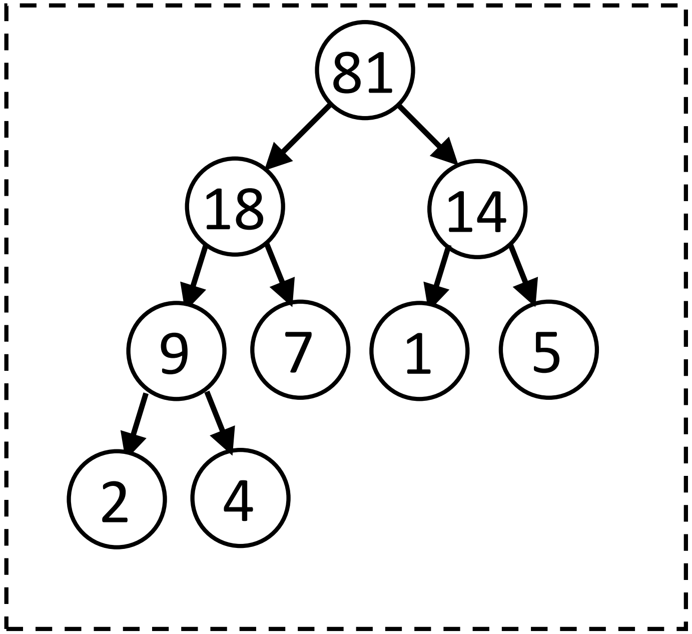
Sorting

Unless stated otherwise, use the best (most adaptive) version of a sort that we’ve developed. Which sort is best in these circumstances?

- Array that is "almost" already sorted
- Very small array
- Medium size array
- Large array (about as big as main memory)
- Very large tape drive

You’re using a quicksort on a very large input, and it’s taking longer than normal. What happened?

Binary Heaps



- Draw the underlying array for this heap
- Push the value 47
  - Use fixUp()
- Draw the resulting tree and array

Priority Queues

- What is the complexity?

	Unordered Array	Ordered (Sorted) Array	Binary Heap
`create(range)`			
`push()`			
`top()`			
`pop()`			

Strings and Sequences

- What is a fingerprint?
- Why do we use them?
- What does it tell you if two strings have the same fingerprint?
- Different fingerprints?
- You don’t need to know exactly how to compute a fingerprint, but know how to use them once they’re calculated