Welcome to CSE 332

Data Structures and Parallelism

Welcome

We have 9 weeks to learn a lot!

- Fundamental data structures and algorithms.
- And their analysis
- Writing Parallel Code

Outline

Introductions

Course Mechanics

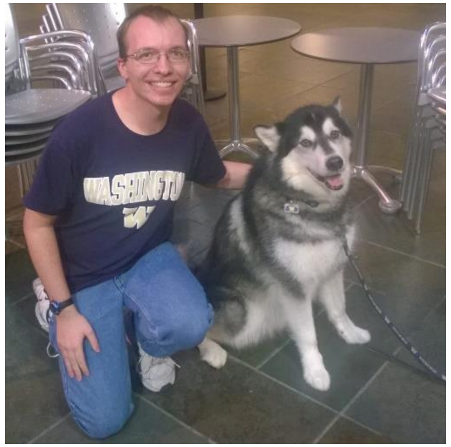
Start of content

- Review of queues and stacks

Course Staff

Instructor: Robbie Weber

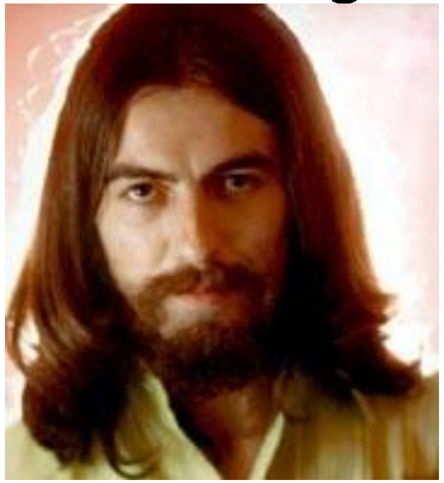
- PhD student in CSE
- Research in algorithm design



TAs

Caitlin Schaefer

Alon Milchgrub



What’s in this course?

Data Structures and Parallelism

Data structures and Algorithms (about 80% of the course)

- Starting to really think like a computer scientist.
  - Make design decisions, think about trade-offs.
- Core data structures and algorithms.
- Mathematically analyze those structures and algorithms.
- **Implement them**

Parallelism

- First serious treatment of programming with multiple threads

Logistics

Textbook:

Weiss, Data Structures and Algorithm Analysis in Java

OPTIONAL (useful if you want more info, or an alternative presentation)

Piazza (message board) please sign up.

Gradescope

Midterm: Friday July 13th (in lecture)

Final: Split over the last two days of classes:

- Thursday August 16th (in section)
- Friday August 17th (in lecture)

Email Robbie ASAP if you have a conflict with any of these dates.

Logistics – Projects

We have a lot to cover…

- in less time than usual.

3 Programming Projects

- Done with a partner
- Split over multiple weeks
  - “Checkpoints” along the way

Your first project comes out Wednesday

There is a partner form on the webpage, fill it out by Tuesday afternoon.

In the meantime: update your Java and Eclipse installs.

Logistics – Exercises

Assigned throughout the quarter.

More frequent (about 10-12). Not as significant time investment as programming projects.

Starting earlier is better (they can take some thought).

Practice the theoretical aspects of the course.

Individual

Academic Honesty

Partners can obviously talk about every aspect of your code in the projects

- And you should, pair programming is **highly** recommended.

In all other cases, high level discussion is fine.

But you must:

- Not take any written notes away from your discussion.
- List everyone you collaborated with on your assignment.

Goal is for you to learn the material.

More details in syllabus.

Abstract Data Type

An **Abstract Data Type (ADT)** is a set of expected behaviors for a set of operations.

Queue ADT

state

Set of elements

behavior

insert(element) – add a new element to the collection.

remove () – returns the element that has been in the collection the longest, and removes it.

peek () – find, but do not remove the element that has been in the collection the longest.

Stack ADT

state

Set of elements

behavior

insert(element) – add a new element to the collection.

remove () – returns the element that has been in the collection the shortest, and removes it.

peek () – find, but do not remove the element that has been in the collection the shortest.

Data Structure

A clever way of organizing data points

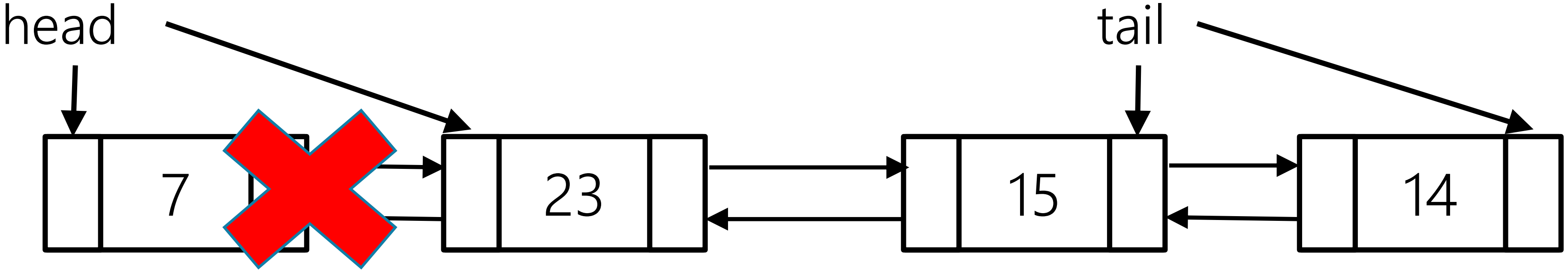
- A data structure is an implementation of an ADT.

Ways to implement a queue

Array

✗	17	3	4	15	13	

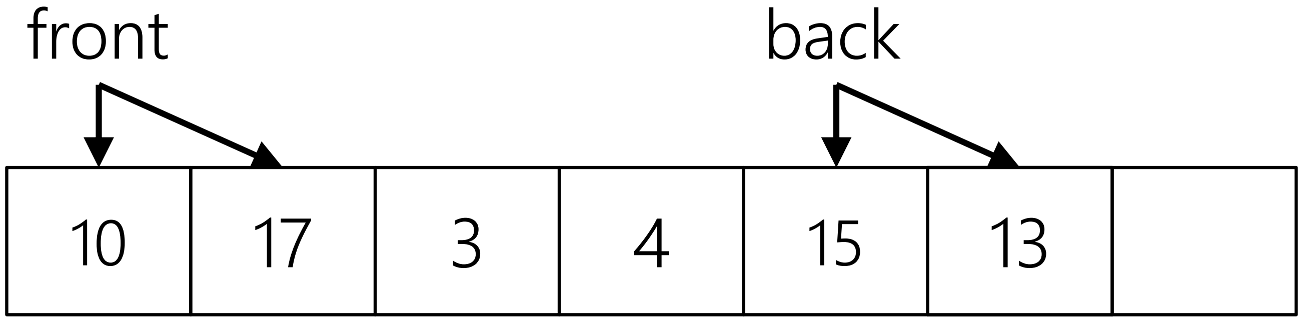
LinkedList



“Circular” Array

A different queue implementation

Removing elements is expensive. What if we just remember where the next element is?



front

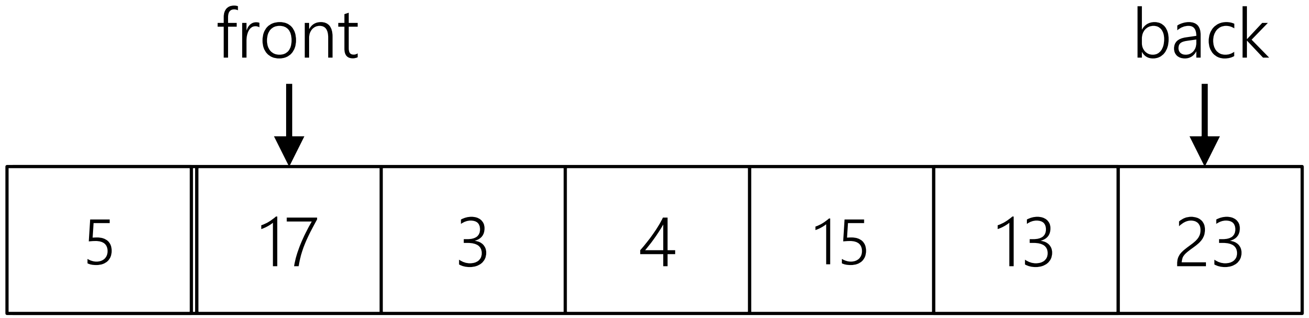
back

10	17	3	4	15	13	

“Circular” Array

What about insertions?

We can “wrap around”



5	17	3	4	15	13	23

At least until the array is completely full.

Tradeoffs

With a doubly-linked list, you can get $O(1)$ insertions and removals as well.

If they're both $O(1)$ why would you choose one over the other?

Updating all those pointers is a constant, but it's a larger constant than array lookups.

If you know the size in advance a circular array has less overhead

But if you don't a linked list easily handles growing. The circular array would be annoying to grow.

Things To Do

Now is a great time to find a partner

I’ll be up here if you have questions

Things to do:
Survey

Sign up for Piazza

Fill out partner form by tomorrow

Get your programming environment ready.