echo "# Datastructures\_and\_algorithms" >> README.md

git init

git add README.md

git commit -m "first commit"

git branch -M main

git remote add origin https://github.com/seppotk/Datastructures\_and\_algorithms.git

git push -u origin main

**…or push an existing repository from the command line**

git remote add origin https://github.com/seppotk/Datastructures\_and\_algorithms.git

git branch -M main

git push -u origin main

<https://github.com/TT00FE39-3001/lecture1>

<https://cpp.sh/>

**Outline**

* Data structures & Abstract Data Types (ADT)
* Algorithms
* Course Mechanics?

**Part 1**

* [Abstract Data Types (ADT)](https://en.wikipedia.org/wiki/Abstract_data_type)
* [Data Structures](https://en.wikipedia.org/wiki/Data_type)
* Data structures are language dependant

[**Data Structures In C++**](https://www.softwaretestinghelp.com/data-structures-in-cpp/)

* Introduction
* Classification
* [Advantages](https://www.softwaretestinghelp.com/data-structures-in-cpp/)

**Stacks vs Queues (1/2)**

* [Stack Data Structure In C++ With ARRAY/Linked list implementation](https://www.softwaretestinghelp.com/stack-in-cpp/)
* [Queue Data Structure In C++ With Illustration](https://www.softwaretestinghelp.com/queue-in-cpp/)

**Stacks vs Queues (2/2)**

* [Data structures 101: Stacks vs Queues](https://www.educative.io/blog/data-structures-stack-queue-java-tutorial)
* [Visualization](https://www.cs.usfca.edu/~galles/visualization/Algorithms.html)
* [FIFO](https://en.wikipedia.org/wiki/FIFO_(computing_and_electronics))
* [LIFO](https://en.wikipedia.org/wiki/Stack_(abstract_data_type))

**Applications**

* Recursion
* Browser history
* Undo (Word processor)

**Part 2**

* Classification
* Analysis of Algorithm Efficiency
* Space vs Time
* Algorithms language agnostic
  + Pseudo code / C++ etc
* [Introduction To Searching Algorithms In C++](https://www.softwaretestinghelp.com/searching-algorithms-in-cpp/)
* [Introduction To Sorting Techniques In C++](https://www.softwaretestinghelp.com/sorting-techniques-in-cpp/)
* [Algorithms In STL](https://www.softwaretestinghelp.com/algorithms-in-stl/)

**Classification**

* Brute Force
* Divide-and-Conquer
* Transform-and-Conquer
* Greedy Technique
* Dynamic Programming

**Analysis of Algorithm Efficiency**

[Big-O, Little-O, Theta, Omega](https://cathyatseneca.gitbooks.io/data-structures-and-algorithms/content/analysis/notations.html)

* Big �, Little �
* Big Ω, Little �
* Θ

**Space vs Time**

* Same tools
* Space and Time Trade-Offs

**Tools**

* [Algorithm Visualizations 1](https://www.youtube.com/playlist?list=PLlzjt-kuLwlo6QuXHSZCkITkfn9M2yHg1)
* [Algorithm Visualizations 2](https://gbhat.com/algorithms/selection_sort.html)
* [Animation engine for explanatory math videos](https://github.com/3b1b/manim)

first try:

#include <iostream>

using namespace std;

int main()

{

cout<<"Hello,World!! This is C++ Tutorial!!\n";

cin.get();

return 0;

}

**# Homework**

**## Reading**

- [Stack Data Structure In C++](<https://www.softwaretestinghelp.com/stack-in-cpp/>)

- [Queue Data Structure In C++](<https://www.softwaretestinghelp.com/queue-in-cpp/>)

**## Pre-Lecture**

- [Data Structures & Algorithms (~16min)](<https://youtu.be/bum_19loj9A>)

- [An Overview of Arrays and Memory (~20min)](<https://youtu.be/pmN9ExDf3yQ>)

- [Introduction to Big O Notation and Time Complexity (~36min)](<https://youtu.be/D6xkbGLQesk>)

**## Recommended: Review**

- Videos:

  - [C++ Tutorial for Beginners - Full Course (4 HOURS!)](<https://youtu.be/vLnPwxZdW4Y>)

  - C[++ Programming All-in-One Tutorial Series (10 HOURS!)](<https://youtu.be/_bYFu9mBnr4>)

  - [ C++ Programming Course - Beginner to Advanced (30 HOURS!)](<https://youtu.be/8jLOx1hD3_o>)

- <https://www.learncpp.com/>

I watched the videos.

ACTIVITY 1

**# Activities**

**## Task 1**

Watch this video (~2min):

<https://youtu.be/ohSzM7WtwOk>

Discuss the difference between:

- Queue vs Stack

- LIFO vs FIFO

**## Task 2**

- Use the following link to push/pop data from the stack:

<https://www.cs.usfca.edu/~galles/visualization/StackArray.html>

- Use the following tool to enqueue / dequeue data from the queue:

<https://www.cs.usfca.edu/~galles/visualization/QueueArray.html>

**## Task 3**

- What is the difference between Array Implementation and the Linked List Implementation of Stacks. Refer to the following link:

<https://www.cs.usfca.edu/~galles/visualization/StackLL.html>

- What is the difference between Array Implementation and the Linked List Implementation of Queues. Refer to the following link:

<https://www.cs.usfca.edu/~galles/visualization/QueueLL.html>

**## Links**

- <https://www.educative.io/blog/data-structures-stack-queue-java-tutorial>

ACTIVITY 2

**# Activities**

Many data structures are used in four basic ways, which we refer to as operations. These operations are:

- Read: Reading refers to looking something up at a particular spot within the data structure. With an array, this means looking up a value at a particular index.

- Search: Searching refers to looking for a particular value within a data structure. With an array, this means looking to see if a particular value exists within the array, and if so, at which index.

- Insert: Insertion refers to adding a new value to our data structure. With an array, this means adding a new value to an additional slot within the array.

- Delete: Deletion refers to removing a value from our data structure. With an array, this means removing one of the values from the array.

Measuring the speed of an operation is also known as measuring its time complexity, efficiency, performance interchangeably. They all refer to the number of steps a given operation takes.

> In the following tasks we will analyze the time complexity of arrays and an array-based set. This is a primitive first order analysis. We will use special tools later

> A set is a data structure that does not allow duplicate values to be contained within it e.g. an array-based set is an array with one additional constraint of barring duplicates.

**## Task 1: Arrays**

Discuss in group whether the following statements are True or false.

- Reading from an array takes one step.

- Searching an array of N elements takes up to N steps e.g. for an array of 5 elements, the maximum number of steps is 5. For an array of 500 elements, the maximum number would take is 500.

- Insertion of an element in an array of length N, takes (N + 1) steps in worst-case scenario.

- Deletion of an element from an array of length N, takes N steps in worst-case scenario.

**## Task 2: Sets**

Discuss in group whether the following statements are True or false.

- Reading from an an array-based set takes one step.

- Searching an array-based set of N elements takes up to N steps.

- Insertion of an element in an array-based set of length N, takes (2N + 1) steps steps in worst-case scenario.

- Deletion of an element from an array-based set of length N, takes N steps in worst-case scenario.

**## Taks 3**

- Array-based sets are arrays with one additional constraint of barring duplicates. How does this single Rule affect efficiency?

- Should we avoid sets because insertion is slower for sets than regular arrays?

**## Reference**

- A Common-Sense Guide to Data Structures and Algorithms,Jay Wengrow

ANSWERS

Many data structures are used in four basic ways, which we refer to as operations. These operations are:

- Read: Reading refers to looking something up at a particular spot within the data structure. With an array, this means looking up a value at a particular index.

- Search: Searching refers to looking for a particular value within a data structure. With an array, this means looking to see if a particular value exists within the array, and if so, at which index.

- Insert: Insertion refers to adding a new value to our data structure. With an array, this means adding a new value to an additional slot within the array.

- Delete: Deletion refers to removing a value from our data structure. With an array, this means removing one of the values from the array.

Measuring the speed of an operation is also known as measuring its time complexity, efficiency, performance interchangeably. They all refer to the number of steps a given operation takes.

> In the following tasks we will analyze the time complexity of arrays and an array-based set. This is a primitive first order analysis. We will use special tools later

> A set is a data structure that does not allow duplicate values to be contained within it e.g. an array-based set is an array with one additional constraint of barring duplicates.

**## Task 1: Arrays**

Discuss in group whether the following statements are True or false.

- Reading from an array takes one step.

- Searching an array of N elements takes up to N steps e.g. for an array of 5 elements, the maximum number of steps is 5. For an array of 500 elements, the maximum number would take is 500.

- Insertion of an element in an array of length N, takes (N + 1) steps in worst-case scenario.

- Deletion of an element from an array of length N, takes N steps in worst-case scenario.

- Reading from an array takes one step.

Reading from an array is, therefore, an efficient operation, since the computer

can read any index by jumping to any memory address in one step.

- Searching an array of N elements takes up to N steps e.g. for an array of 5 elements, the maximum number of steps is 5. For an array of 500 elements, the maximum number would take is 500.

So, it turns out that for an array of five cells, the maximum number of steps

linear search would take is five. For an array of 500 cells, the maximum

number of steps linear search would take is 500.

- Insertion of an element in an array of length N, takes (N + 1) steps in worst-case scenario.

We can say that insertion in a worst-case scenario can take *N + 1 steps* for

an array containing N elements. This is because we need to shift all N elements

over, and then finally execute the actual insertion step.

- Deletion of an element from an array of length N, takes N steps in worst-case scenario.

We can say then, that for an array containing N elements,

the maximum number of steps that deletion would take is N steps.

**## Task 2: Sets**

Discuss in group whether the following statements are True or false.

- Reading from an an array-based set takes one step.

- Searching an array-based set of N elements takes up to N steps.

- Insertion of an element in an array-based set of length N, takes (2N + 1) steps steps in worst-case scenario.

- Deletion of an element from an array-based set of length N, takes N steps in worst-case scenario.

- Searching an array-based set of N elements takes up to N steps.

**## Taks 3**

- Array-based sets are arrays with one additional constraint of barring duplicates. How does this single Rule affect efficiency?

- Should we avoid sets because insertion is slower for sets than regular arrays?

- Array-based sets are arrays with one additional constraint of barring duplicates. How does this single Rule affect efficiency?

Reading from a set is exactly the same as reading from an array—it takes just

one step for the computer to look up what’s contained within a particular index.

As I described earlier, this is because the computer can jump to any index

within the set since it can easily calculate and jump to its memory address.

Searching a set also turns out to be no different than searching an array—it

takes up to N steps to search for a value within a set. And deletion is also

identical between a set and an array—it takes up to N steps to delete a value

and move data to the left to close the gap.

Insertion, however, is where arrays and sets diverge. Let’s first explore

inserting a value at the *end* of a set, which was a best-case scenario for an

array. We saw that with an array, the computer can insert a value at its end

in a single step.

With a set, however, the computer first needs to determine that this value

doesn’t already exist in this set—because that’s what sets do: they prevent

duplicate data from being inserted into them.

Now, how will the computer ensure that the new data isn’t already contained

in the set? Remember, a computer doesn’t know offhand what values are

contained within the cells of an array or set. Because of this, the computer

will first need to *search* the set to see whether the value we want to insert is

already there. Only if the set does not yet contain our new value will the

computer allow the insertion to take place.

So, every insertion into a set *first requires a search.*

**## Reference**

- A Common-Sense Guide to Data Structures and Algorithms,Jay Wengrow

<https://pragprog.com/titles/jwdsal2/a-common-sense-guide-to-data-structures-and-algorithms-second-edition/>

<https://www.linkedin.com/in/jaywengrow/>

ACTIVITY 3

**# Git / Github**

- Clone today's repo

- Create a new branch e.g answers

- Create a repository in GitHub

- Change remote to point to your repo

**## Links**

- [How To Change Git Remote Origin](<https://devconnected.com/how-to-change-git-remote-origin/>)

DONE

<https://github.com/seppotk/Datastructures_and_algorithms>