Sorting & Searching Algos

Week 2 - 2019 AI Inspire

What is Sorting & Searching

- Search for items in our daily lives
- Sort through items alphabetically, numerically, etc to make life easier
- Relationship?
  - Sorting data can make searching a whole lot easier

Applications

- Finding phone number in phone book
- Search for exact book in huge library
  - Library uses sorting algo to make books organized
- Google search engine use ranking system
  - Sorts through billions of pages and searches for answer to query
  - Search algos → look at several different factors
  - Sorting helps with searching result

Measuring Efficiency

Both Space and Time Complexity

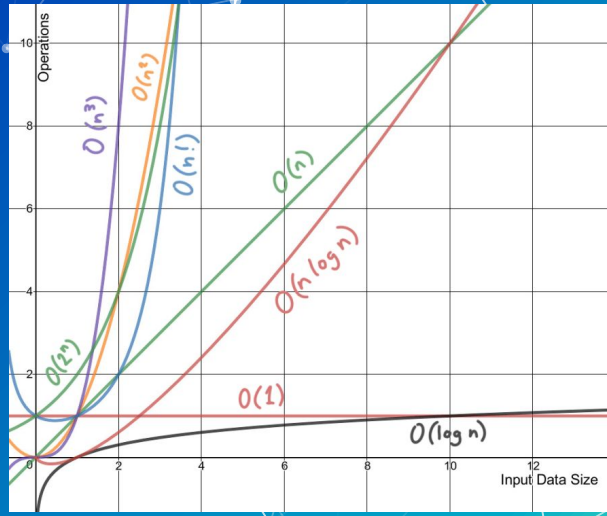
Space Complexity

- Amt of **space/storage/memory** algorithm takes
- Space = storage = memory
- Only finite amount of memory computer has to give to algo
  - Need to use the space wisely
- Seems abstract but will go over more examples
- Can measure by looking at diff data structures used

Time Complexity

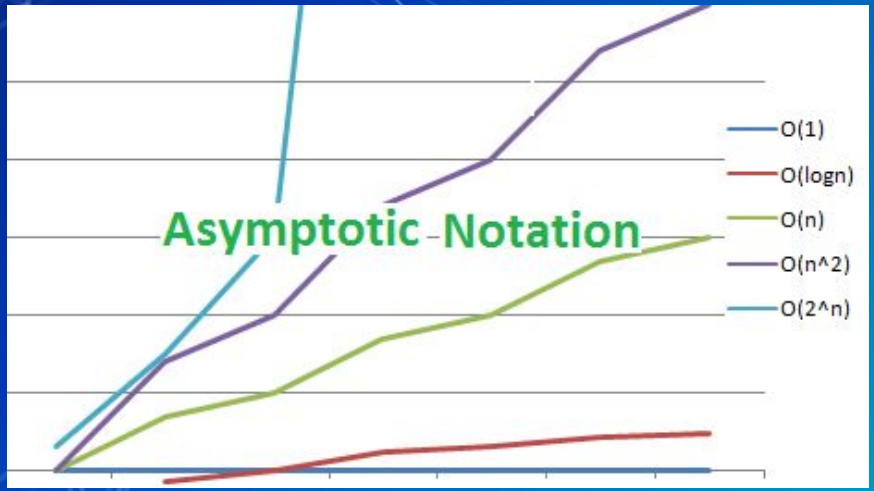
- Amt of **time** algorithm takes to run
- Can literally measure by using StopWatch API in java and measuring elapsed time

- Tradeoffs between time and space complexity



How do we actually define this on paper?

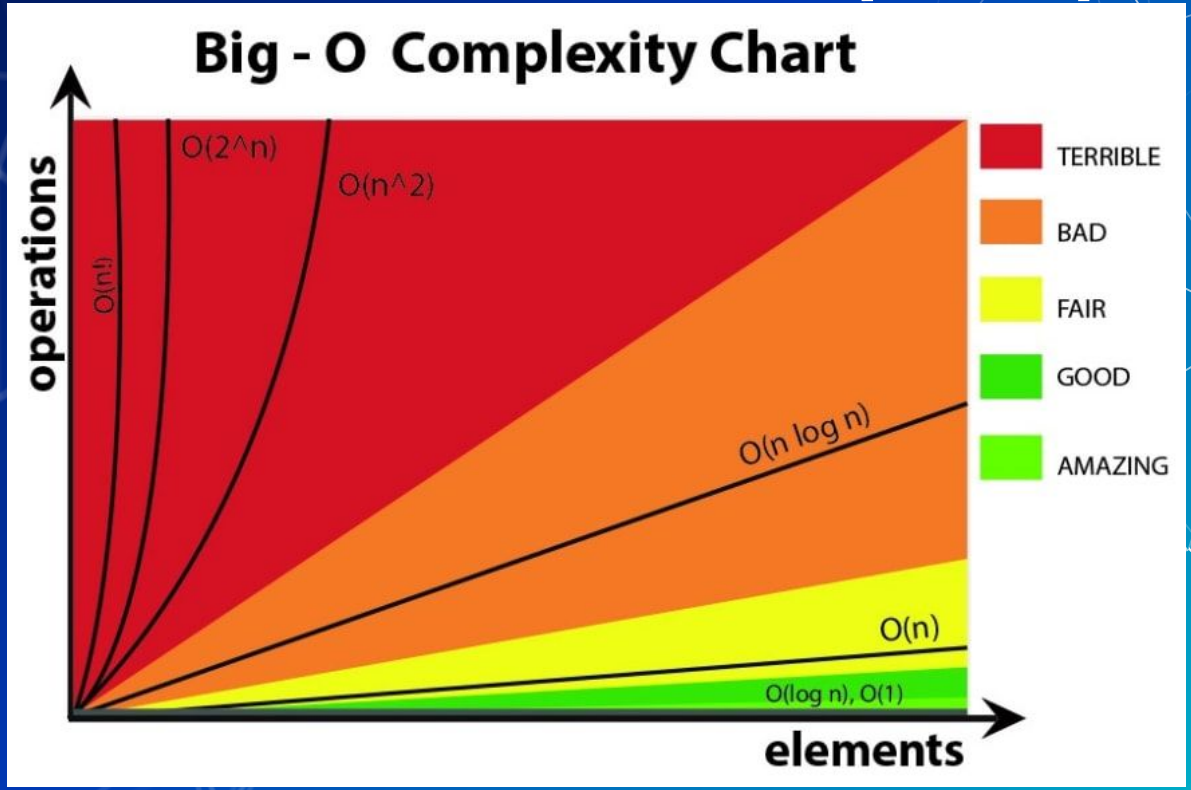
- ASYMPTOTIC NOTATION
  - Mechanism to calculate time algorithm takes to run along with space it takes up as FUNCTION OF LENGTH OF INPUT



Big O Notation to Measure Complexity

- Upper bound of input
- How much time the algorithm would take in WORST case scenario?
  - Ex - time insertion sort takes in worst case vs. time selection sort takes in worse case
- Ignore coefficients and lower order time
- Ex -
  - **Number of times algorithm has to access an array as input changes is Big-O notation**
    - **Array access = cost model**
    - **Other types of cost models = increment. Variable declaration, # times of comapring with less than and greater than, etc.**

Big O Notation to Measure Complexity



Big O Notation Example

2 Sum Problem - Iterate through 2 arrays and increment count whenever 2 numbers sum to 0

1. Develop code
2. Analyze run time using Big O Notation
    a. Keep cost model as number array accesses

Solution to Algo Analysis

$$1 + 2 + \ldots + (n-1) = \frac{1}{2} n(n-1)$$

Above, is the amt of time it takes to access the array in total for only a[i].

Then, to get array accesses for a[j] also → multiply run time by 2 → result is Big-Oh of n^2.

https://www.youtube.com/watch?v=506f1GTLLeQ

Efficiency + Asymptotic Notation

Sorting Algorithms = ordering list of items

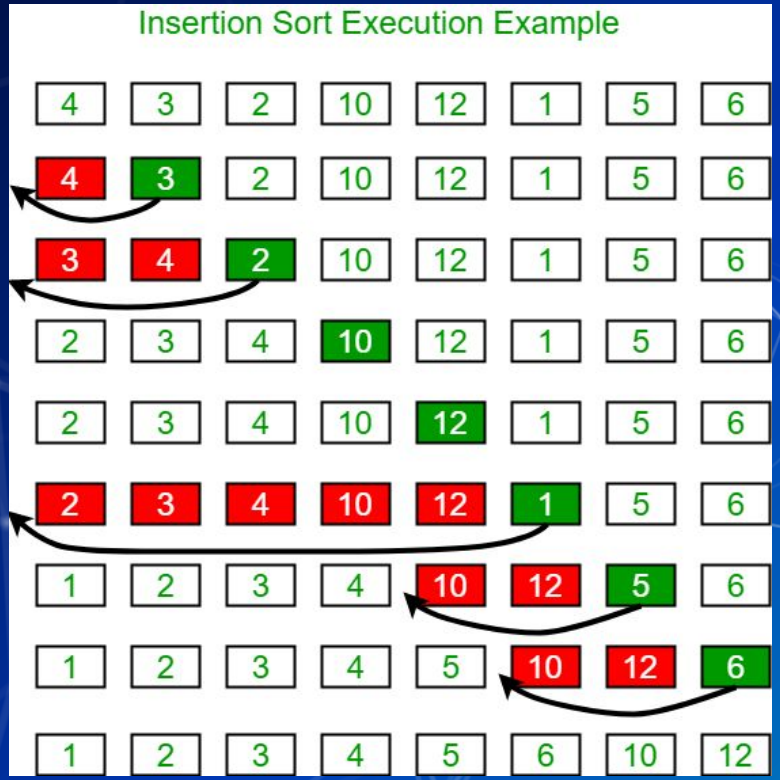
Why are we studying so many diff types of sorting algos?

- Different sorting algos have different tradeoffs
  - Some are more efficient timewise, others are more efficient from a space complexity standpoint
- Interesting to see how these trade offs work
  - Compare efficiencies of diff algos

Notions of Swapping & Comparing

- Swapping - exchange two items in list
- Compare - comparing two items in list
  - compareTo() method used to compare two items
    - Used freq. in sorting & searching algos

Insertion Sort




```
for (int i = 0; i < N; i++) {

    // Iterate through elements from right to left
    for (int j = i; j > 0; j--) {

        // Swap a[j] with a[j-1] as long as a[j] < a[j-1]
        // Otherwise, break.
        if(less(a[j], a[j-1])) swap(a, j, j-1);
        else                   break;
    }
}
```

Insertion Sort - Analysis

Big O Analysis

Amount of time it takes for algo to run in worst case?

- First determine worst case scenario
- Look at number of compares to analyze algo
  - Need to choose a cost function or operation that is one of the main ops.

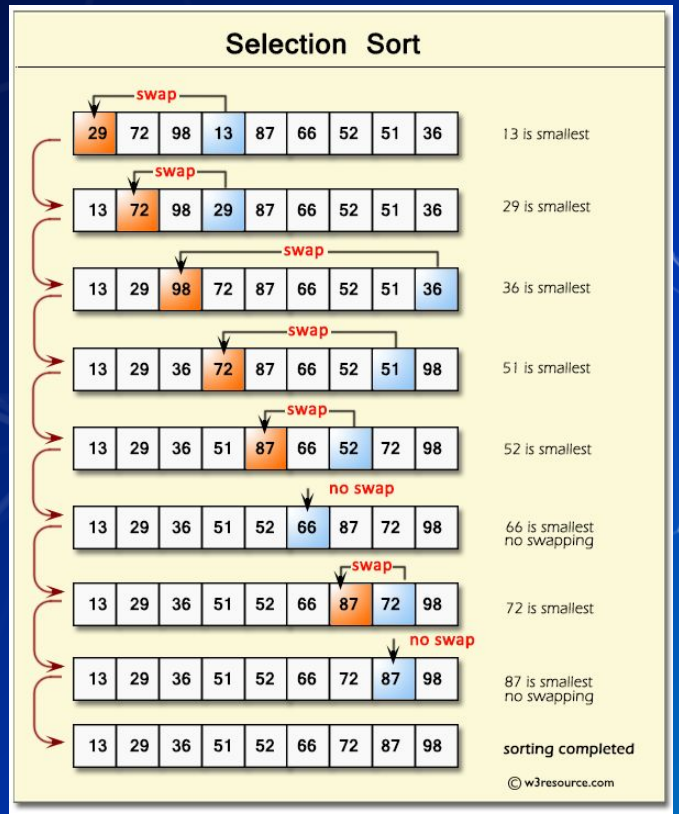
Insertion Sort - Analysis

Big O Analysis

Amount of time it takes for algo to run in worst case?

- First determine worst case scenario
- Look at number of compares to analyze algo
  - Need to choose a cost function or operation that is one of the main ops.

Selection Sort




```
for (int i = 0; i < N; i++) {
    int min = i;

    // Iterate through all entries starting from i.
    for (int j = i+1; j < N; j++) {

        // Find index min; the index with the smallest value.
        if (less(a[j], a[min])) min = j;
    }

    // Swap a[i] with a[min].
    swap(a, i, min);
}
```

Selection Sort - Analysis

Big O Analysis

Amount of time it takes for algo to run in worst case?

- First determine worst case scenario
- Look at number of compares to analyze algo
  - Need to choose a cost function or operation that is one of the main ops.

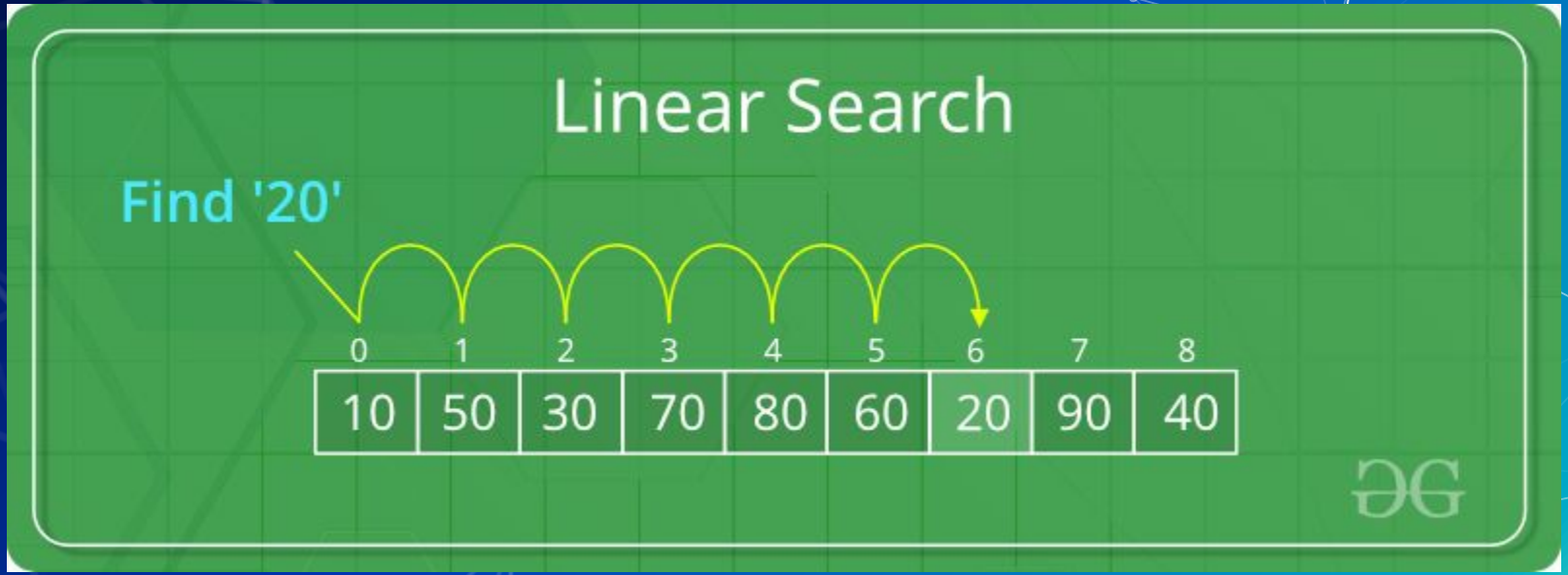
So because their worst cases are same does it mean they are equally good?

- It DEPENDS!!!!
  - Real world performance has many other factors other than just worst case performance
    - Processing Speed, Operating System, Amount of RAM, etc.

Searching Algorithms

Make sure to sort list before searching for element!

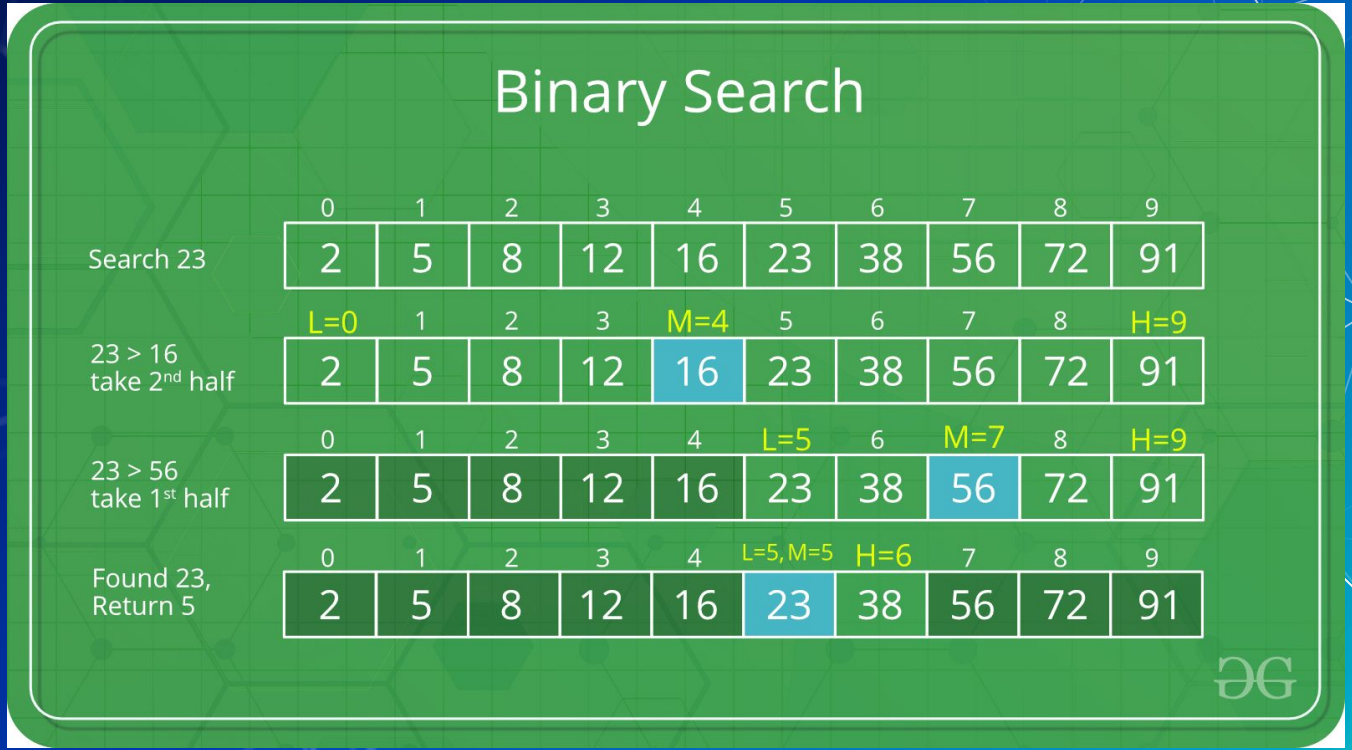
Linear/Sequential Search



Linear Search Analysis

- Operation we are focusing on is “CHECK IF EQUAL” (serves as cost model)
- Worst case run time?
  - Make sure to first picture worst case, then derive it
- Best case run time?
  - Picture best case ⟶ then derive number of hops and checks needed

Binary Search



Binary Search Analysis

- Operation we are focusing on is “CHECK IF EQUAL” (serves as cost model)
- Best case
  - Picture best case → derive result
- Worst case
  - More challenging - Need to think more deeply about
  - Goal - look at # of “check if equals” needed to get to end result in worst case
  - Task → figure out this run time

Next Session

- Look at Github to see the material + instructions for installing IDE
- More material will be posted on Sorting & Searching
  - More sorting & searching algos with time complexities + code
- Next session will be on Stacks & Queues algos

Hope you learnt something new today! See you next week!