MAKE SCHOOL

ALGORITHM ANALYSIS

Big O? What's the big deal?

WHAT IS AN ALGORITHM?

A sequence of steps to accomplish a task

…described precisely enough for a computer to perform them

Algorithm != Code

Algorithms can be written in any language, including *pseudocode*, English, Farsi, etc.

More info: algorithm definition, algorithm characterizations, pseudocode

MAKE SCHOOL

WHO COINED ALGORITHM?

Al Gore… Rhythm?

After all, he *did** invent the Internet *not

Muhammad ibn Mūsā al-Khwārizmī

9th century Persian mathematician

Also coined “*al-jabr*” – a.k.a. *algebra*

More info: word origin, etymology, Muhammad ibn Mūsā al-Khwārizmī

MAKE SCHOOL

ALGORITHM ANALYSIS

Correctness – does it solve the problem?

Formal verification methods – e.g., NASA

Resource usage – how efficient is it?

Time – idealized work steps (CPU cycles)

Space – working memory (RAM, disk, etc.)

More info: correctness, formal verification, algorithm analysis

WHY IS THIS IMPORTANT?

Cost – computers and electricity aren't free

Comparison – choose best solution based on analysis *before* implementing and running

Tractability – not all problems can be solved in your lifetime, even with moderate size datasets

Non-deterministic Polynomial-time, *NP-hard*

More info: [P versus NP problem](), [NP-hard](), [NP-complete]()

MAKE SCHOOL

RUNTIME IN PRACTICE

Actual runtime depends on many factors:

CPU speed – pace of instruction execution

Language – abstractions on top of CPU instructions

Compiler – optimization of instruction ordering, etc.

Environment – resources available during runtime (not used by other programs running concurrently)

RUNTIME COMPARISON

	Computer A	Computer B
Sorting Algorithm	Bubble Sort	Merge Sort
Time Complexity	n^2	$n \cdot \log_2 n$
Execution Speed	10 GHz (billion instr/sec)	10 MHz (million instr/sec)
Code Optimization	High (2 instr/step)	Low (50 instr/step)
Actual Complexity	$2n^2$	$50n \cdot \log_2 n$
Time to Sort 100K #s	2 seconds	8.3 seconds
Time to Sort 1M #s	3.33 minutes	1.67 minutes
Time to Sort 10M #s	5.56 hours	19.4 minutes
Time to Sort 100M #s	23.1 days!	3.69 hours

Example from Algorithms Unlocked

MAKE SCHOOL

RUNTIME ASSUMPTIONS

Need to make some assumptions to easily describe algorithm runtime in the abstract:

Time to execute basic operations is *constant*

Relative speed of operations is *not important*

Including arithmetic, logic, comparison, variable assignment, array indexing, function calls, etc.

RUNTIME ANALYSIS

What we really want to analyze is *complexity* – how algorithm runtime grows as the input gets larger

Describe runtime as a function of the size of the input

Input array/list length usually written as ***n***, ***m***, ***k***

Ignore constants and lower-order terms

e.g., $3n \rightarrow n$, $6n^2 + 15n + 24 \rightarrow n^2$

LINEAR SEARCH

Return index of `item` in `histogram`, or `None` if not found

```python
def linear_search(item, histogram):
    index = None  # not found
    i = 0
    for word, freq in histogram:
        if word == item:
            index = i  # found
        i += 1
    return index
```

PYTHONIC LINEAR SEARCH

Return last index of `item` in `histogram`, or `None` if not found

```python
def linear_search(item, histogram):
    index = None  # not found

    for i, (word, freq) in enumerate(histogram):
        if word == item:
            index = i  # found

    return index
```

BETTER LINEAR SEARCH

Return first index of `item` in `histogram`, or `None` if not found

```python
def linear_search(item, histogram):


    for i, (word, freq) in enumerate(histogram):
        if word == item:
            return i  # found
    else:
        return None  # not found
```

ASYMPTOTIC NOTATION

Worst case – upper bound

Algorithm is $O(f(n))$ – "big oh of $f(n)$"

Best case – lower bound

Algorithm is $\Omega(f(n))$ – "omega of $f(n)$"

If both bounds are the same, then

Algorithm is $\Theta(f(n))$ – "theta of $f(n)$"

RUNNING TIME SUMMARY

	Best Case	Worst Case	Bounded
Linear Search (return last match)	$\Omega(n)$	$O(n)$	$\Theta(n)$
Better Linear Search (return first match)	$\Omega(1)$	$O(n)$	N/A

RESOURCES

Introduction to Algorithms by Cormen, Leiserson, Rivest, and Stein – widely considered *the Bible of Algorithms*

Algorithms Unlocked by Thomas Cormen – introductory and more accessible, less technical detail than CLRS

Recursion and Recurrences by U. of Dartmouth Math

Sorting-Algorithms.com – animations of common sorting algorithms running in parallel on specific datasets

MAKE
SCHOOL