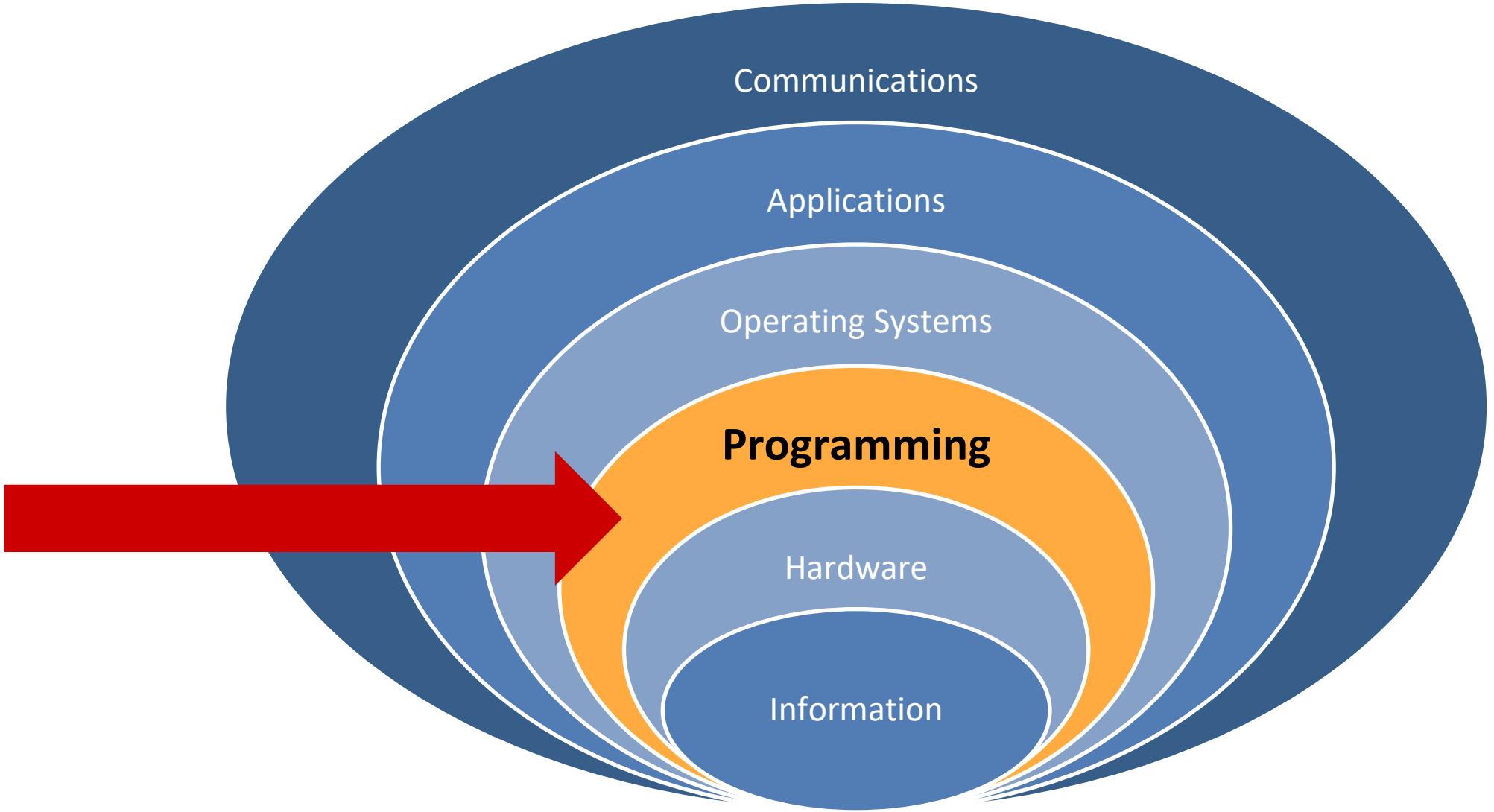
Computer Science Fundamentals

Lecture topic: Programming Layer

Lecturer: Olga Yugay

Today's focus



Agenda

- Problem solving and problem-solving strategies
- Algorithm
- Algorithm with selection
- Algorithm with repetition
- Searching algorithms
  - Sequential search
  - Binary search
- Algorithm complexity

Let’s refresh your understanding

What is Programming? | Intro to JS: Drawing & Animation | Computer programming | Khan Academy [1]

[1] Khan Academy. (2015). *What is Programming?* | *Intro to JS: Drawing & Animation* | *Computer programming* . Retrieved June 9, 2022, from https://www.youtube.com/watch?v=FCMxA3m_Imc.

Problem solving

- Problem solving - the act of finding a solution to a perplexing question
- A computer is not intelligent
  - Computer can do nothing without being told what to do.
  - Human (the *programmer) must analyze the problem, develop the instructions* for solving the problem (the *program),* and then have the computer carry out the instructions. [2]

[2] Dale, N., & Lewis, J. (2020). Computer Science Illuminated (7th ed.). Jones & Bartlett Learning, Ch 7-9

Problem solving strategies - *heuristics*

- **Ask questions**
  - What do I know about the problem?
  - What is the information that I have to process in order the find the solution?
  - What does the solution look like?
  - What sort of special cases exist?
  - How will I recognize that I have found the solution?
- **Look for Familiar Things**
  - never reinvent the wheel. If a solution exists, use it. If you've solved the same or a similar problem before, just repeat the successful solution.
  - E.g. Use Design Patterns (more about it at level 5)
- **Divide and Conquer**
  - break up a large problem into smaller pieces that we can solve individually. [3]

[3] Dale, N., & Lewis, J. (2020). Computer Science Illuminated (7th ed.). Jones & Bartlett Learning, Ch 7-9

The computer problem-solving process [4]

Analysis and specification phase	
Analyze	Understand (define) the problem.
Specification	Specify the problem that the program is to solve.
Algorithm development phase	
Develop algorithm	Develop a logical sequence of steps to be used to solve the problem.
Test algorithm	Follow the steps as outlined to see if the solution truly solves the problem.
Implementation phase	
Code	Translate the algorithm (the general solution) into a programming language.
Test	Have the computer follow the instructions. Check the results and make corrections until the answers are correct.
Maintenance phase	
Use	Use the program.
Maintain	Modify the program to meet changing requirements or to correct any errors.

FIGURE 7.2 The computer problem-solving process

[4] Dale, N., & Lewis, J. (2020). Computer Science Illuminated (7th ed.). Jones & Bartlett Learning, Ch 7-9

The interactions among the problem solving phases

The **black lines** show the general flow through the phase.

The **red lines** represent paths that can be taken to backtrack to a previous phase if a problem occurs. [5]

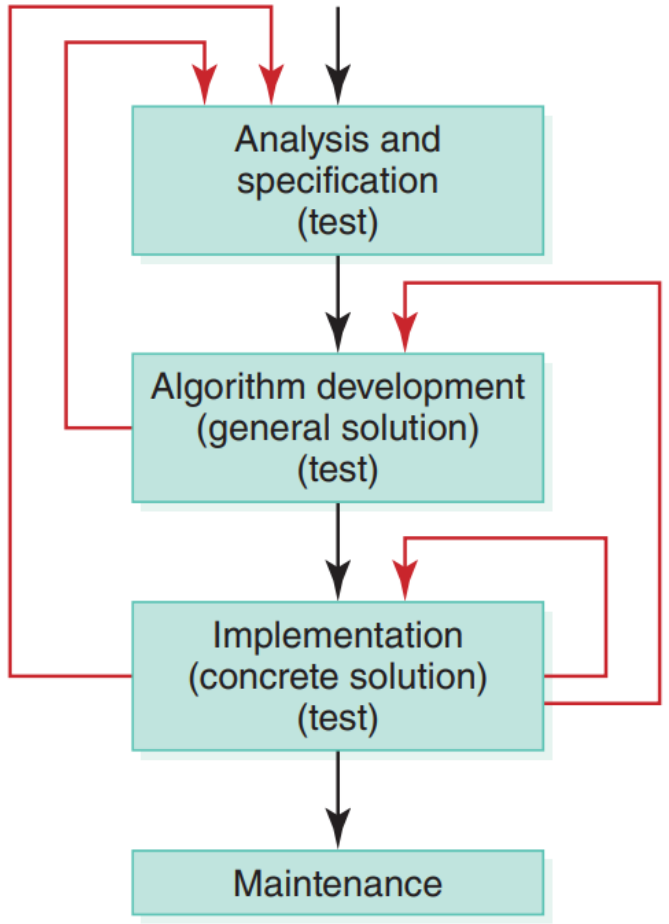


FIGURE 7.3 The interactions among the four problem-solving phases

[5] Dale, N., & Lewis, J. (2020). Computer Science Illuminated (7th ed.). Jones & Bartlett Learning, Ch 7-9

Algorithm

- Algorithm – set of instructions for solving a problem or subproblem in a **finite amount of time** using a **finite amount of data** [6]

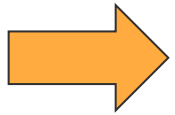
- "Algorithm is a tool for solving a well-specified computational problem. The statement of the problem specifies in general terms the desired input/output relationship. The algorithm describes specific computational procedures for achieving input/output relationship." (Cormen et. al.)

[6] Dale, N., & Lewis, J. (2020). Computer Science Illuminated (7th ed.). Jones & Bartlett Learning, Ch 7-9

Presenting algorithm - pseudocode

- *Pseudocode* - uses a mixture of English and formatting to make the steps in the solution explicit. Used to present the algorithm

- Have a look at sample algorithm for **getting up in the morning** [7]



Alarm goes off
Hit sleep button
Alarm goes off
Hit sleep button
Alarm goes off
Turn off alarm
Move dog
Throw back covers
Put feet over side of the bed
Stand up

[7] Dale, N., & Lewis, J. (2020). Computer Science Illuminated (7th ed.). Jones & Bartlett Learning, Ch 7-9

What is algorithm and why you should care?

What's an algorithm? - David J. Malan [8]

What is algorithm and why should you care? [9]

[8] Malan, D. J. (2013, May 20). What's an algorithm? - David J. Malan. YouTube. Retrieved June 9, 2022, from https://youtu.be/6hfOvs8pY1k

[9] Khan Academy. (2015, July 27). What is an algorithm and why should you care? | algorithms | computer science . YouTube. Retrieved June 9, 2022, from https://www.youtube.com/watch?v=CvSOaYi89B4

Summary of methodology for designing algorithms[10]

1. Analyze the Problem
    a. list information to work with (data)
    b. specify potential solution
    c. list assumptions
    d. develop overall plan
2. List the Main Tasks
    a. Main module
3. Write the Remaining Modules
    a. modules on one level can specify more modules on lower levels
4. Re-sequence and Revise as Necessary

[10]Dale, N., & Lewis, J. (2020). Computer Science Illuminated (7th ed.). Jones & Bartlett Learning, Ch 7-9

Algorithms with selection[11]

Case: write an algorithm to express what dress is appropriate for a given outside temperature.


```
Write "Enter the temperature"
Read temperature
Determine dress
```

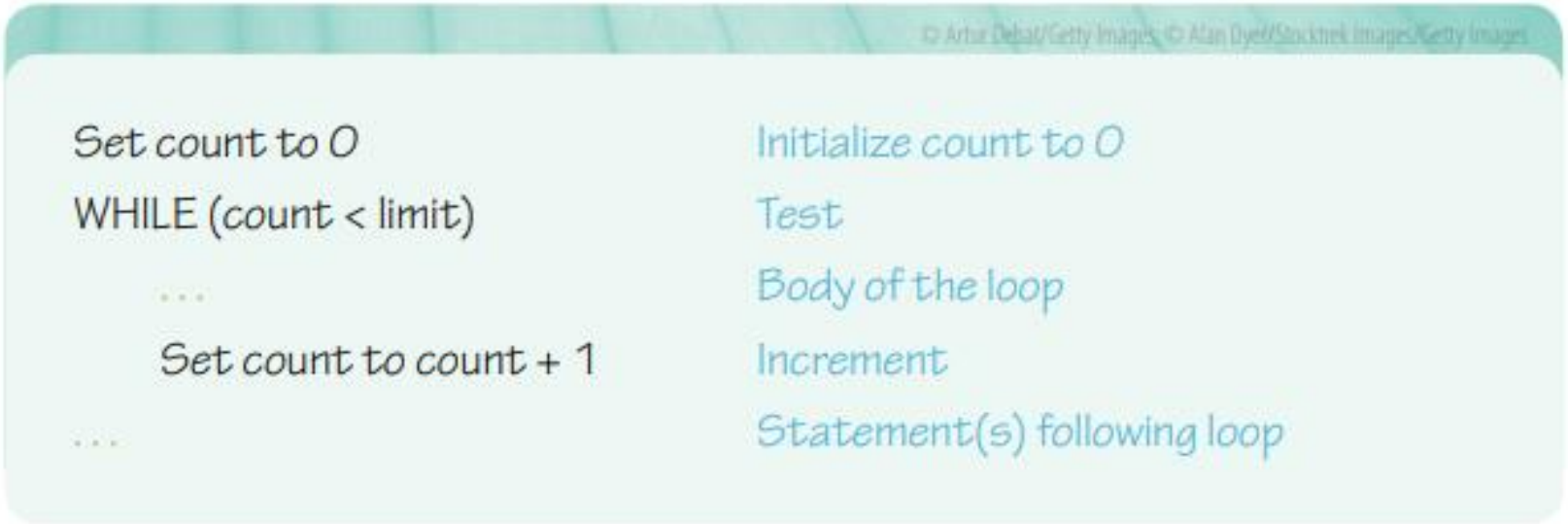

Determine dress

```
IF (temperature > 90)
    Write "Texas weather: wear shorts"
ELSE IF (temperature > 70)
    Write "Ideal weather: short sleeves are fine"
ELSE IF (temperature > 50)
    Write "A little chilly: wear a light jacket"
ELSE IF (temperature > 32)
    Write "Philadelphia weather: wear a heavy coat"
ELSE
    Write "Stay inside"
```

[11]Dale, N., & Lewis, J. (2020). Computer Science Illuminated (7th ed.). Jones & Bartlett Learning, Ch 7-9

Algorithms with repetition [12]

Count-controlled loops



[12]Dale, N., & Lewis, J. (2020). Computer Science Illuminated (7th ed.). Jones & Bartlett Learning, Ch 7-9

Algorithms with repetition [13]

Event-controlled loops

When implementing an event-controlled loop using a while statement, there are again three parts to the process:

1. the event must be initialized
2. the event must be tested
3. and the event must be updated

The algorithm o the right reads and sums data values until negative value is read

[13]Dale, N., & Lewis, J. (2020). Computer Science Illuminated (7th ed.). Jones & Bartlett Learning, Ch 7-9

Algorithms with repetition [14]

event controlled loop continued

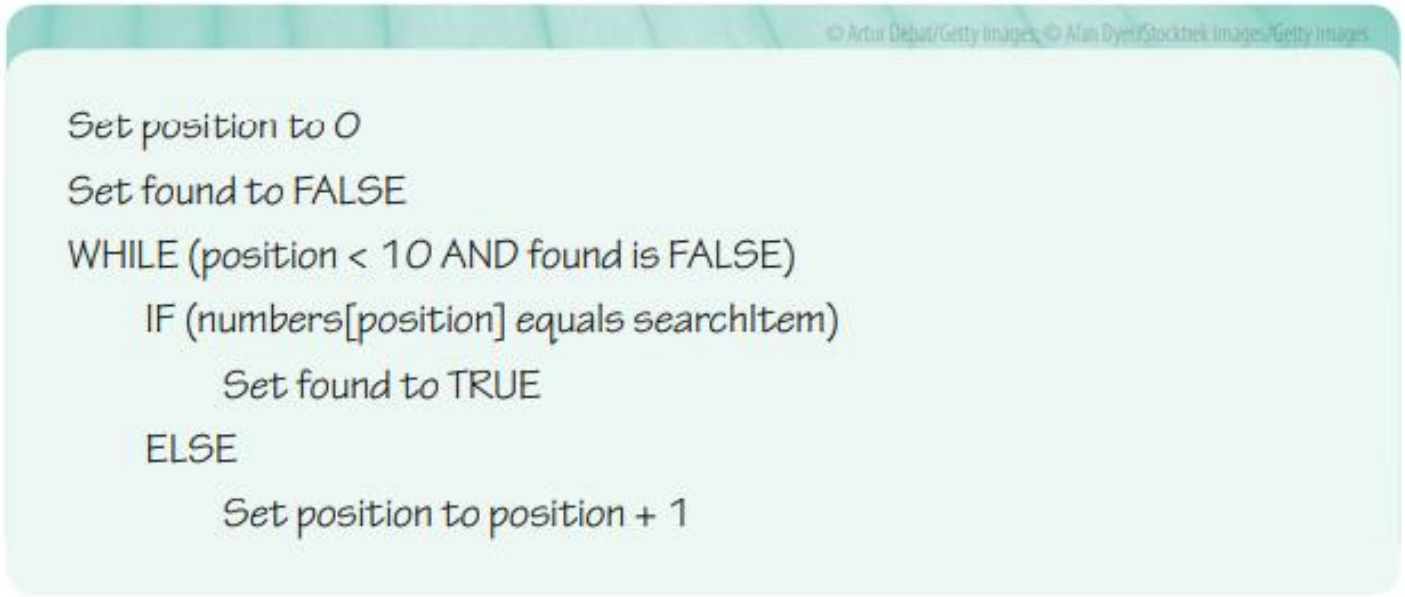
```
Set sum to 0                                  Initialize sum to zero
Set posCount to 0                             Initialize event
WHILE (posCount < 10)                         Test event
    Read a value
    IF (value > 0)                            Should count be updated?
        Set posCount to posCount + 1          Update event
        Set sum to sum + value                Add value into sum
...                                           Statement(s) following loop
```

[14]Dale, N., & Lewis, J. (2020). Computer Science Illuminated (7th ed.). Jones & Bartlett Learning, Ch 7-9

Searching algorithms: sequential search [15]

Each item is looked at in turn and compared to the one being searched.

Stop either when the item is found, or when all items are looked at and not found a match



```
Set position to 0
Set found to FALSE
WHILE (position < 10 AND found is FALSE)
    IF (numbers[position] equals searchItem)
        Set found to TRUE
    ELSE
        Set position to position + 1
```

[15]Dale, N., & Lewis, J. (2020). Computer Science Illuminated (7th ed.). Jones & Bartlett Learning, Ch 7-9

Searching algorithm: Binary search [16]

Binary search is an efficient algorithm for finding an item from a sorted list of items. It works by repeatedly dividing in half the portion of the list that could contain the item, until you've narrowed down the possible locations to just one.

Watch video on binary search algorithm [17]

[16] Dale, N., & Lewis, J. (2020). Computer Science Illuminated (7th ed.). Jones & Bartlett Learning, Ch 7-9

[17] HackerRank. (2016, September 27). Algorithms: Binary search. YouTube. Retrieved June 9, 2022, from https://www.youtube.com/watch?v=P3YID7liBug

Example

Given list of numbers find number 5

1, 2, 5, 7, 8, 10, 50, 60, 71, 77, 78, 79, 80

x=5

Example

Given list of numbers find number 5

1, 2, 5, 7, 8, 10, 50, 60, 71, 77, 78, 79, 80

x=5

1) Midpoint is 50 (13/2=6.5, 7th number is the midpoint)

To identify midpoint:

1. Count the total number of number entries (which is 13 in this example)
2. Divide the total number by 2 (in this example, 13/2=6.5
3. Round the result 6.5 ≈ 7, 7th number is the midpoint

Example

Given list of numbers find number 5

1, 2, 5, 7, 8, 10, ~~50, 60, 71, 77, 78, 79, 80~~

x=5

1) Midpoint is 50 (13/2=6.5, 7th number is the midpoint)
2) x< midpoint (5<50), ignore the range on the right (50-80)

Example

Given list of numbers find number 5

1, 2, 5, 7, 8, 10, ~~50, 60, 71, 77, 78, 79, 80~~

x=5

1) Midpoint is 50 (13/2=6.5, 7th number is the midpoint)
2) x< midpoint (5<50), ignore the range on the right (50-80)
3) Midpoint is 5

To identify new midpoint:

1. Count the total number of left number entries (which is 6)
2. Divide the total number by 2 (in this example, 6/2=3,
3. 3rd number is the new midpoint

Example

Given list of numbers find number 5

1, 2, 5, 7, 8, 10, ~~50, 60, 71, 77, 78, 79, 80~~

x=5

1) Midpoint is 50 (13/2=6.5, 7th number is the midpoint)
2) x< midpoint (5<50), ignore the range on the right (50-80)
3) Midpoint is 5
4) x is equal to midpoint (5==5). The number is found!

Algorithm complexity[18]

Complexity estimates how an algorithm performs regardless the kind of machine it runs on.

- time complexity - the amount of time algorithm needs to execute and produce the result
- space complexity - the amount of memory used by the algorithm (including the input values to the algorithm) to execute and produce the result.

[18]Dale, N., & Lewis, J. (2020). Computer Science Illuminated (7th ed.). Jones & Bartlett Learning, Ch 7-9

Algorithm complexity

Complexity is defined as function of the input size n using Big-O notation.

n indicates the size of the input

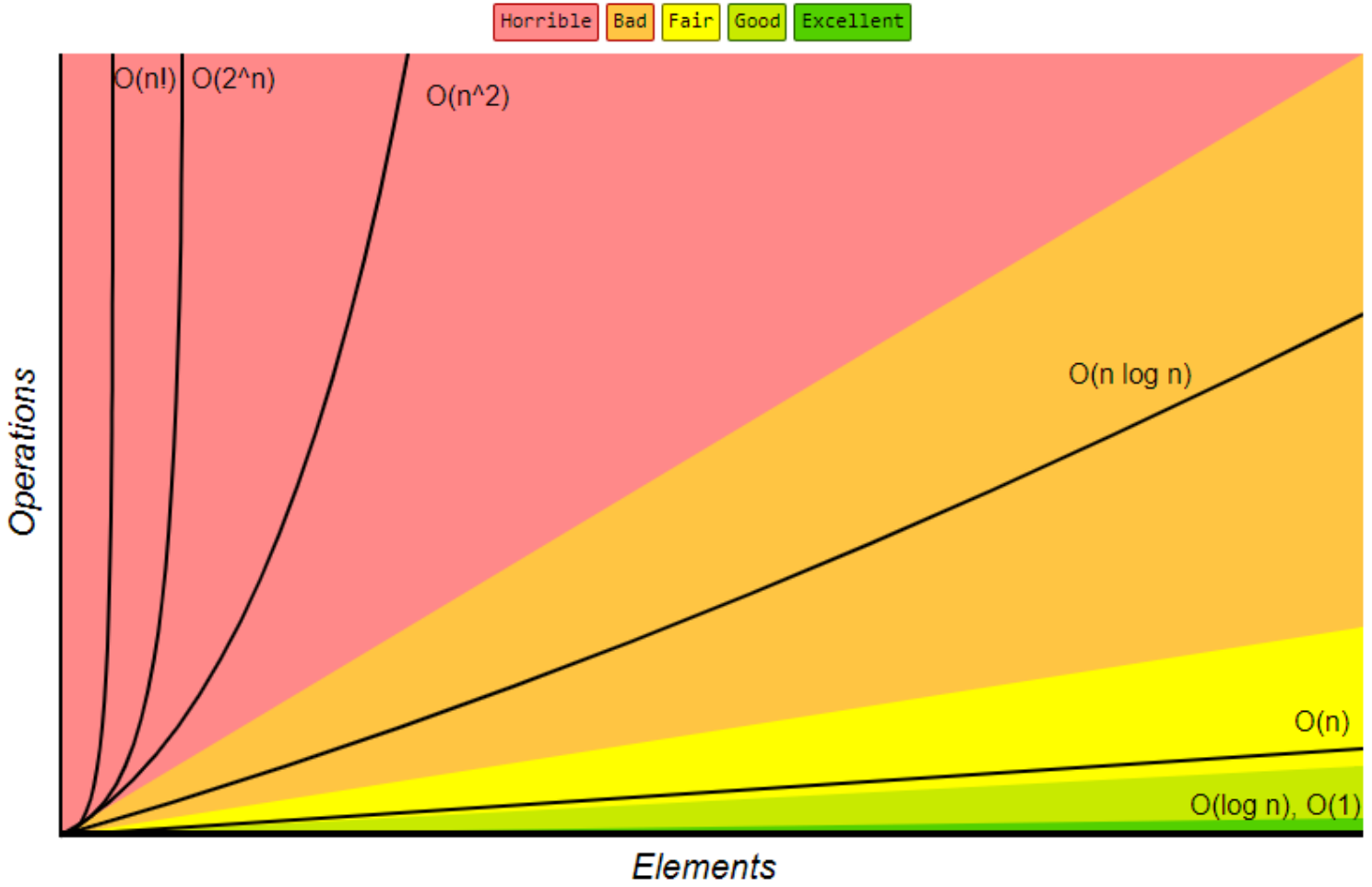
O is the worst-case scenario growth rate function.

- **O(1)** - the algorithm takes the same amount of time to compute regardless of the input size.
- **O(n)** - linear time, as the input grows, the algorithms take proportionally longer to complete
- **O(log n)** - logarithmic time complexity, a time execution is proportional to the logarithm of the input size

And others... Have a look here: 8 time complexities that every programmer should know [19]

[19] Mejia, A. (2019, September 19). *8 time complexities that every programmer should know*. Adrian Mejia Blog. Retrieved June 9, 2022, from https://adrianmejia.com/most-popular-algorithms-time-complexity-every-programmer-should-know-free-online-tutorial-course/

Algorithm complexity



[20] *Know thy complexities!* Big. (n.d.). Retrieved June 9, 2022, from https://www.bigocheatsheet.com/

Common data structure operations

Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

[21] *Know thy complexities!* Big. (n.d.). Retrieved June 9, 2022, from https://www.bigocheatsheet.com/

Array sorting algorithms

Algorithm	Time Complexity			Space Complexity
	Best	Average	Worst	Worst
Quicksort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(log(n))
Mergesort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(n)
Timsort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)
Heapsort	Ω(n log(n))	Θ(n log(n))	O(n log(n))	O(1)
Bubble Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Insertion Sort	Ω(n)	Θ(n^2)	O(n^2)	O(1)
Selection Sort	Ω(n^2)	Θ(n^2)	O(n^2)	O(1)
Tree Sort	Ω(n log(n))	Θ(n log(n))	O(n^2)	O(n)
Shell Sort	Ω(n log(n))	Θ(n(log(n))^2)	O(n(log(n))^2)	O(1)
Bucket Sort	Ω(n+k)	Θ(n+k)	O(n^2)	O(n)
Radix Sort	Ω(nk)	Θ(nk)	O(nk)	O(n+k)
Counting Sort	Ω(n+k)	Θ(n+k)	O(n+k)	O(k)
Cubesort	Ω(n)	Θ(n log(n))	O(n log(n))	O(n)

[22] *Know thy complexities!* Big. (n.d.). Retrieved June 9, 2022, from https://www.bigocheatsheet.com/

Recommended sources:

- Computer Science Illuminated, Chapter 7-9, end of the book exercises
- What is algorithm and why should you care? [video]

- https://adrianmejia.com/most-popular-algorithms-time-complexity-every-programmer-should-know-free-online-tutorial-course/

- Explore:
  - http://carlcheo.com/startcoding
  - http://carlcheo.com/wp-content/uploads/2014/12/which-programming-language-should-i-learn-first-infographic.png

References

1. Dale, N., & Lewis, J. (2020). Computer Science Illuminated (7th ed.). Jones & Bartlett Learning, Ch 7-9
2. Khan Academy. (2015). What is Programming? | Intro to JS: Drawing & Animation | Computer programming . Retrieved June 9, 2022, from https://www.youtube.com/watch?v=FCMxA3m_Imc.
3. Malan, D. J. (2013, May 20). *What's an algorithm? - David J. Malan*. YouTube. Retrieved June 9, 2022, from https://youtu.be/6hfOvs8pY1k
4. HackerRank. (2016, September 27). *Algorithms: Binary search*. YouTube. Retrieved June 9, 2022, from https://www.youtube.com/watch?v=P3YID7liBug
5. Mejia, A. (2019, September 19). *8 time complexities that every programmer should know*. Adrian Mejia Blog. Retrieved June 9, 2022, from https://adrianmejia.com/most-popular-algorithms-time-complexity-every-programmer-should-know-free-online-tutorial-course/
6. *Know thy complexities!* Big. (n.d.). Retrieved June 9, 2022, from https://www.bigocheatsheet.com/
7. Cheo, C. (n.d.). *Which programming language should I learn first? [infographic]*. CarlCheo.com: Technology, Software, and Computer Tips. Retrieved June 9, 2022, from http://carlcheo.com/startcodin

Thank you