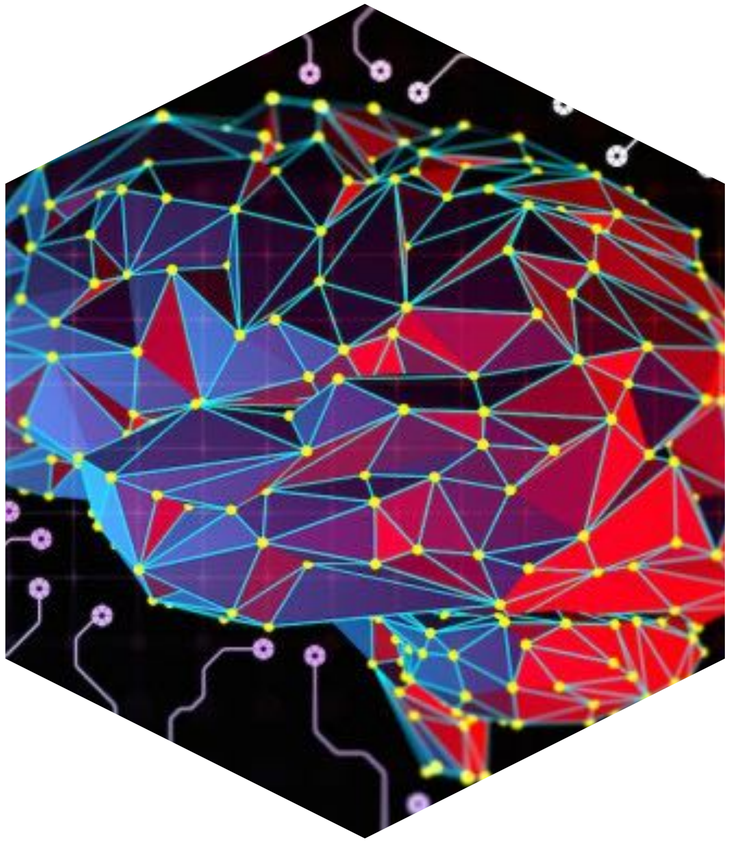
DATA STRUCTURES AND ALGORITHMS



Jan P.

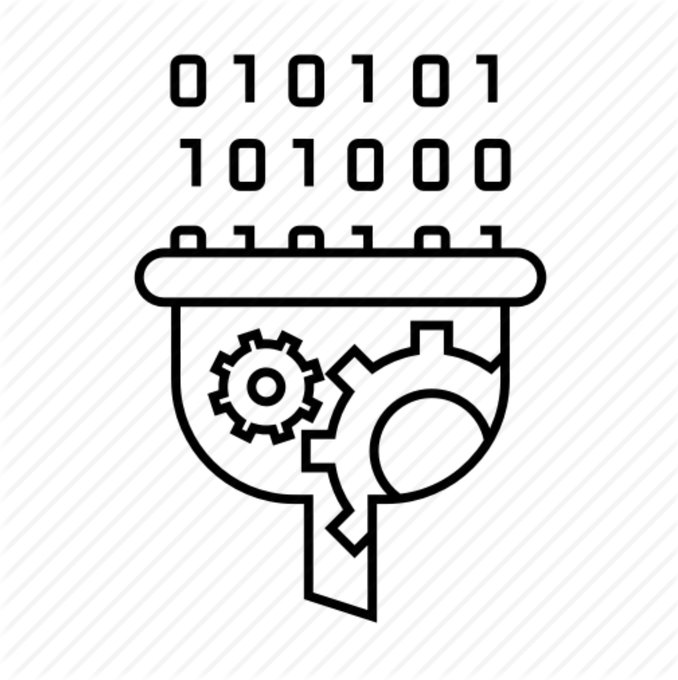
GitHub.com/fafk
https://fafk.github.io

JavaScript, Rust, Java, LISP & crypto
patomil@gmail.com

OVERVIEW

- Why algorithms and data structures?
- Arrays & Objects
- Binary Search
- Queue, Stack
- Trees, Heap
- Big O (time/space complexity)
- Recursion

Data Structures & Algorithms



algorithm: a step-by-step procedure for solving a problem

Computation:
input data
+
instructions (algorithm)
=>
output data

A computer program is a set of instructions (an algorithm) operating on data structures.

We can make our program run efficiently
by choosing the right data structures and algorithms.

Basic data structures

Array

A sequence of indexed values.

```
const array = [“a”, “b”, “c”];
Array.of(“a”, “b”, “c”);
new Array(“a”, “b”, “c”);

console.log(array[0], array[1]);
// => “a”
console.log(array[1]);
// => “b”

array.push(“new element”);
array.pop(“get last element”);
console.log(array.length);
// => 3
```

Arrays are everywhere.

Queue

FiFo - first in, first out

```
const queue = [];

// put value on end of queue
queue.push(1);

// take first value from queue
queue.shift();
```

Stack

FiLo - First in, last out

StackOverflow.com?

```
var queue = [];

// push value onto the stack
queue.push(1);

// take the value from the top
var value = queue.pop();
```

“Queue” and “stack” are commonplace terminology.

There are no “maps” on the low level. We can have a look at how a map can be implemented with an array using modulo.

Objects (Maps)

Object:

```
{
  name: “Jan”,
  hobbies: [“JavaScript”, “Books”],
  height: 181
}
```

Array of objects:

```
[{
  name: “Jan”,
  hobbies: [“JavaScript”, “Books”],
  height: 181
}, {
  name: “Bob”,
  hobbies: [“PHP”, “Hiking”],
  height: 176
},{
  name: “Alice”,
  hobbies: [“Haskell”, “Cycling”],
  height: 171
}]
```

Objects store structured data.

Sometimes we call objects “key value stores”.
Sometimes we call them “hash maps”.
Sometimes we call them “dictionaries”.

Objects are native to JavaScript —that’s not the case in low-level programming languages.

We access values (after the “:”) by keys:
{“myKey”: “I am a string value!”, “numberKey”: 5}

```
const obj = {“myKey”: “I am a value!”, “numberKey”: 123};
console.log(obj.myKey);
=> “I am a value!”
console.log(5 * obj.myKey);
=> 25
```

https://repl.it/@fafk/hashMap

Binary Search

O(n) to 0(log(n)) magic.

https://repl.it/@fafk/binarySearch#index.js

Let people toy with the exercise to get a practical feel for it.

Traversing an array

```
const arr = [2, 4, 6];

let i = 0;

while (i < arr.length){
  console.log("Value:", arr[i]);
  console.log("Value squared:", arr[i]**2);
  i++; // i = i + 1;
}

for (i = 0; i < arr.length; i++) {
  console.log(arr[i]);
}
```

We traverse arrays to process the data stored in them.

In real life we do this very often.

Let’s code a real-life example: how many city bikes are there in the 2nd district?

https://repl.it/@fafk/CityBikes

```
const arrOfObjects = [{val: 2}, {val: 4}];
i = 0;

while (i < arrOfObjects.length) {
  console.log("Value:", arrOfObjects[i].val);
  i++; // i = i + 1;
}
```

Dive into Bubble Sort
Dive into Merge Sort

Sorting arrays

You will probably never need to write a sorting algorithm outside of an interview.
And yet, people might ask about it.

Learning about sorting algorithms is a good segue into a discussion on complexity.

👉 **Bubble Sort** https://repl.it/@fafk/BubbleSort
👉 **Merge Sort**
👉 **Heap Sort**

Sorting in JavaScript

```
console.log([“c”, “b”, “a”].sort());
=> [“a”, “b”, “c”]
```

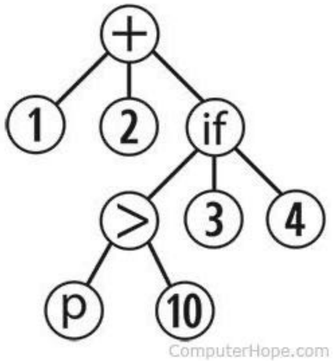
```
console.log([5,3,10].sort());
=> [ 10, 3, 5 ]
```

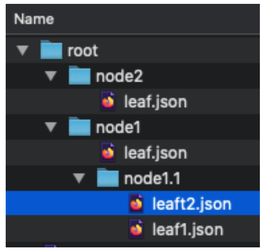
```
console.log([5,3,10].sort((a, b) => a > b)
=> [3, 5, 10]
```

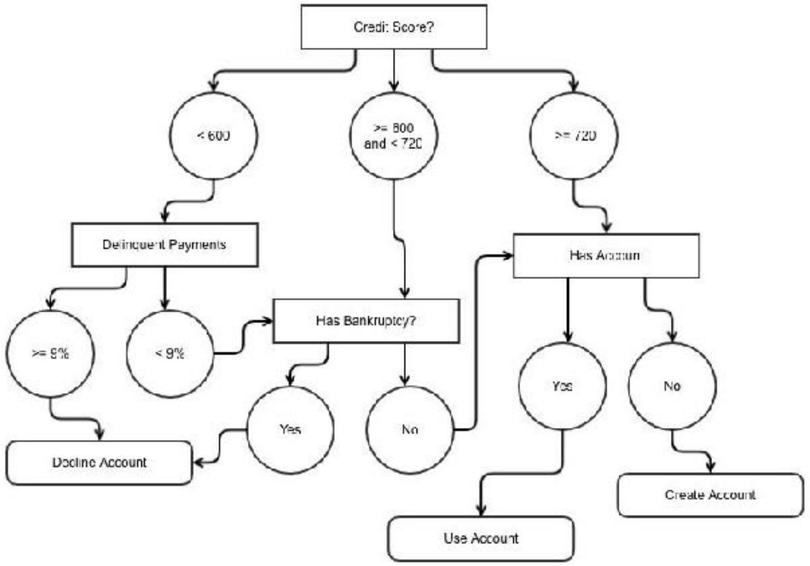
```
const unsorted = [{name: “Zeus”}, {name: “Andromeda”}];
console.log(unsorted.sort((a, b) => a.name > b.name)
=> [{name: “Andromeda”}, {name: “Zeus”}]
```

Trees

Explain why the tree on the left is a computer program/algorithm.







Trees. are. everywhere.

Trees? 

Define and draw a graph, show what makes graph a tree.

Directed acyclic graphs. Graphs?

Root node, node, leaf.

Parent, child. Ancestor, descendant.

Binary tree.

Heap

A tree that:

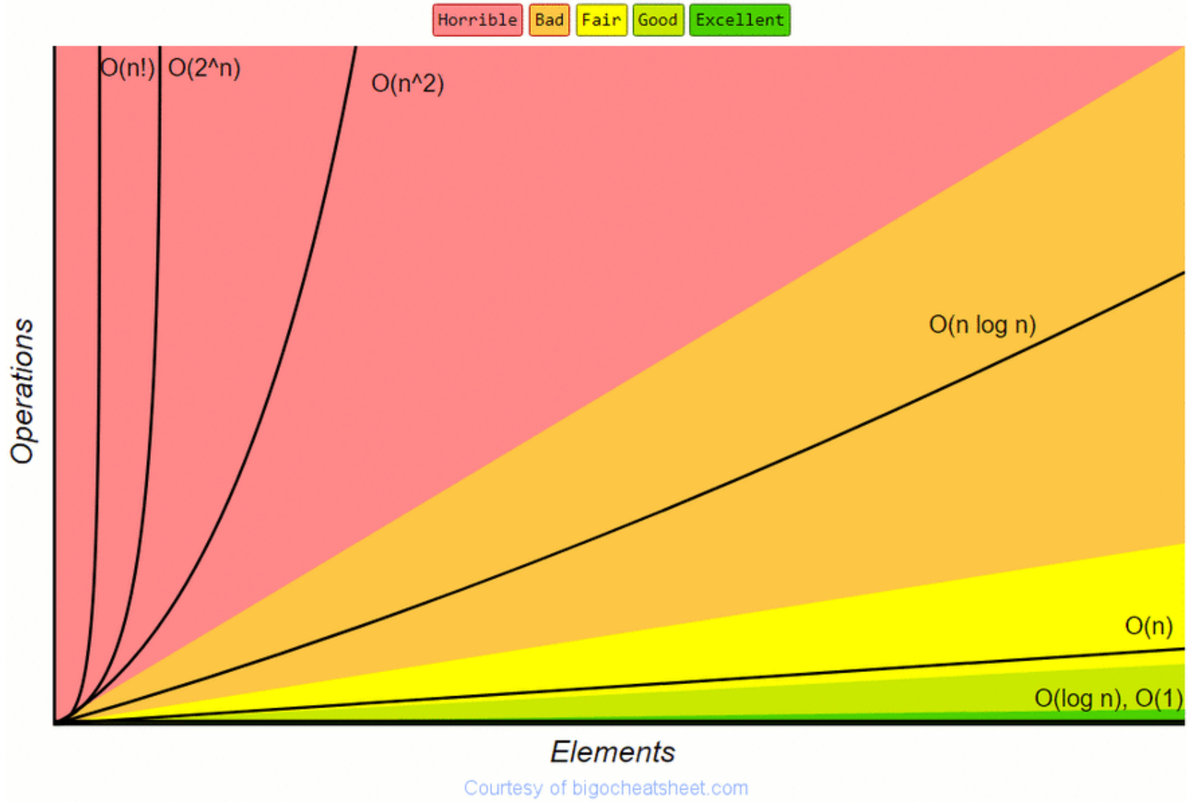
- is **complete**: all levels of heap should be full, except the last one;
- maintains **heap ordering**: the value of each node or child is greater than or equal to the value of its parent, with the minimum value at the root node (= min-heap).

Complexity

Explain big O notation (worst case stripped off constants).

Explain motivation: to reason about algorithms. Explain on bubble sort.

For an input of length N... how many steps will an algorithm take to finish?



Recursion 

Let's talk about what recursion is, how it's used to traverse lists and how it's used to implement merge sort.

Requires a stop condition.

Can be used instead of iteration.

https://twitter.com/joelnet/status/1214268814294016000

Find how how many bikes are there in the 2nd district using recursion!
https://repl.it/@fafk/CityBikes

Thanks!

Any questions?

You can find me at patomil@gmail.com