MESA

> StartBig

First Practical

2023

MESA

MESA

WHO IS THIS GUY?

Giovanni Manfredi

- MESA active Member since October 2022
- Participant at IT Sprint 2023 in Skopje, North Macedonia
- HO and creator of StartBig
- Tech savvy since I was 6 years old

AGENDA

- **MESA** - What is MESA?
- What **does** MESA do?
- How do I **join**?
- What about **StartBig**?

What is MESA?

Local Association

European Association

Milan **E**ngineering
Student **A**ssociation

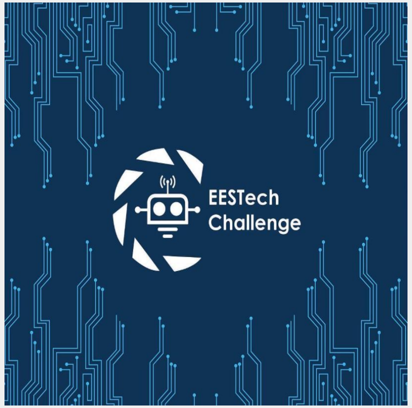
Electrical **E**ngineering **ST**udent
European Asso**C**iation

What does MESA do?

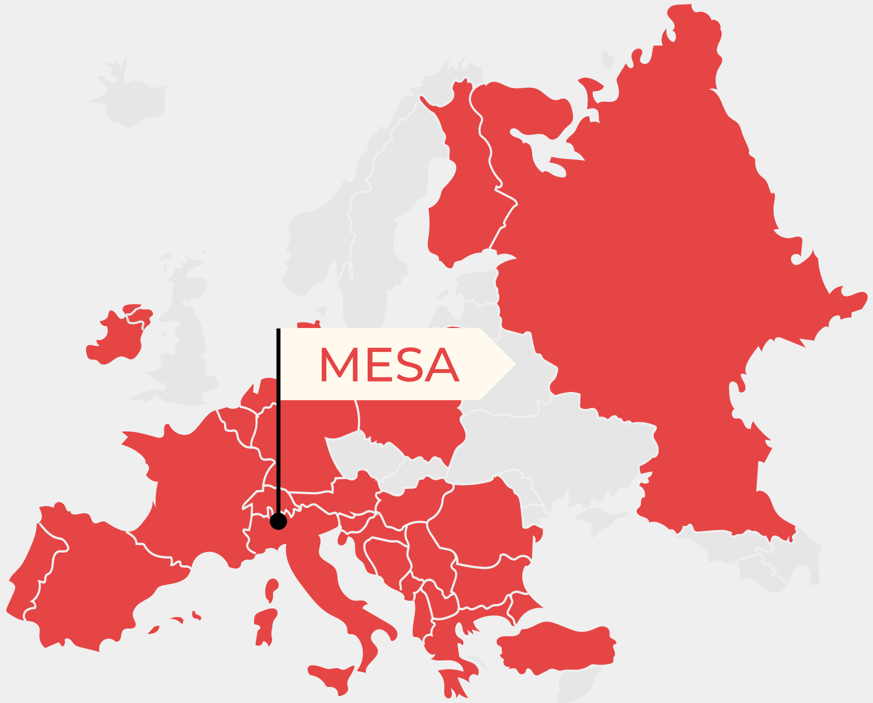
EVENTS!

Hard Skills

Soft Skills



An International Network



24 countries

43 universities

5000+ people

5 regions

How do I join?

Scan the QR and
click on the section Join us

What about StartBig?

CareerService session

How do we get to do a coding interview?

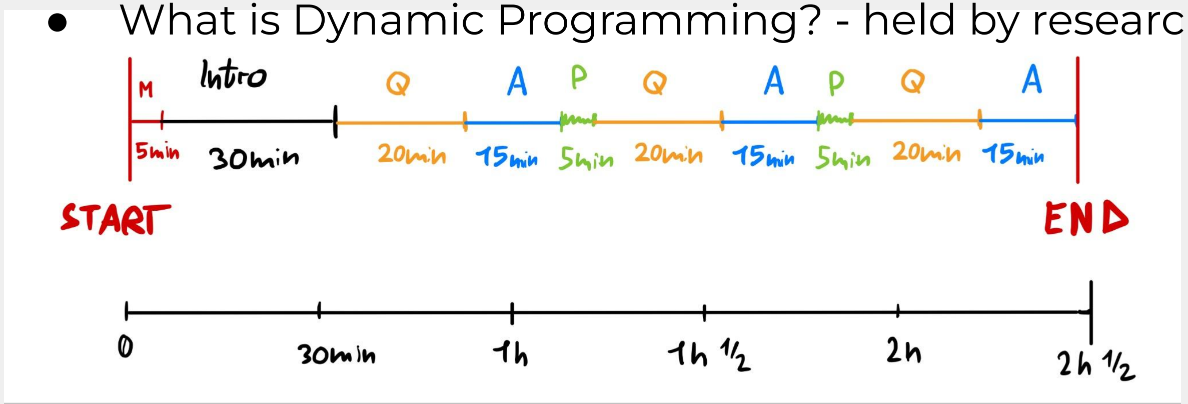
CareerService is here to help us find that out!

3 Practical sessions

How can we prepare for a coding interview?

Content of the sessions:

- Introduction to coding interview questions - held by me
- Trees & Graphs (DSF & BSF algorithms) - held by researcher Davide Yi Xian Hu
- What is Dynamic Programming? - held by researcher Nicolò Felicioni



LeetCode

GitHub

Company session

How is the true experience behind coding interviews?

Andrea from **Oracle** will help us understand that!

There will be a small ice-breaker interview and then open questions from you!

STAGE TO ...

Giovanni Manfredi

Yes, that’s still me

MESA

Tutors here to help!

FOURSIGMA
ENGINEERING

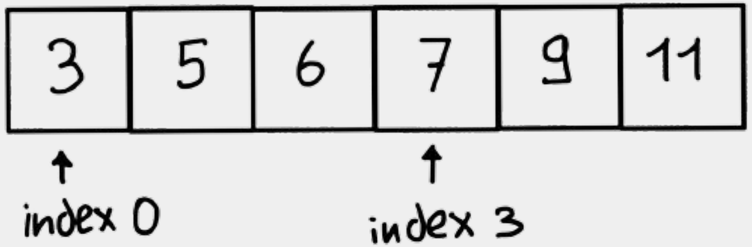
First Practical Meeting AGENDA

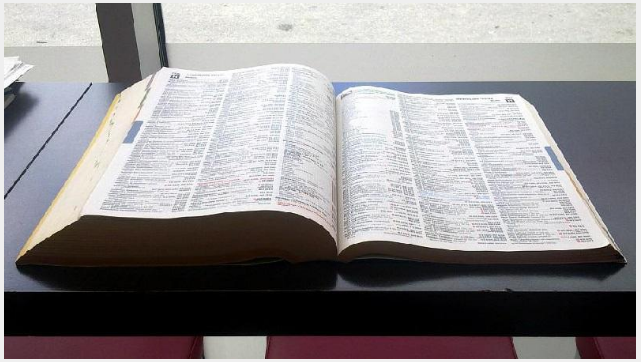
1. **How** do I **solve** a **coding interview** question?
2. **How** can I **optimise** my solution?
3. What's **LeetCode** and how does it work?
4. Let's **start coding**!
5. Let's see a **solution** together
6. **Repeat** 4. three times
7. **End** of the session!

MESA

Draw out the problem!

Example: Search in an ordered Array





AMENTO
25 34 77
4 81 34
4 67 75
26 14 40
4 67 20
25 51 82
49 27 69

TI ASCOLI
25 90 91
ETTI ROMA

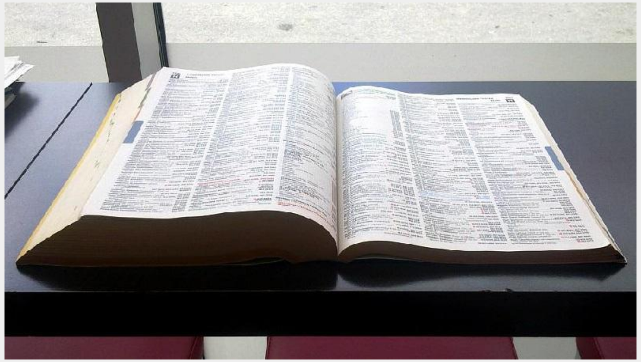
(06) 49 16 43
Ovest
4 29 51/34 21 00
4 50 42

» Domenico, 6 v. [illegible] Napoli
CANALA Albano, 120 v. Sassari
» Antonio, 48 v. Pesaro
» Antonio, 9 v.
» ANTONUCCI Tomassina
24 v. Perugia
» BAIOCCHI Adalgisa
15 v. S. Emidio Rosso
» Bruno, 34 v. Sassari
» Carlo, 1/b v. Picena Aprutina
» CINESI Lucia, 6 v. Faleria
» Elisa, 20/a v. Sassari
» Emidio, 220 vl. Treviri
» Emidio, 30 p. Arringo
» Fausto, 2 vl. Croce
» Fulvio, 19 v. S. Emidio Rosso
» Gino, 128 v. Salaria
» Giorgio, 9 v. Ravenna
» Giuseppe, 51 v. 267
» 44 v. Sassari
» 55 Filippo Giacor

How would you solve it by hand?

Example: How do you search into a address book? You use the index!

If you open the book at random, you will turn the pages right or left **depending** on how the letter you're searching for **compares** to the one present in the address book!



What are some more examples?

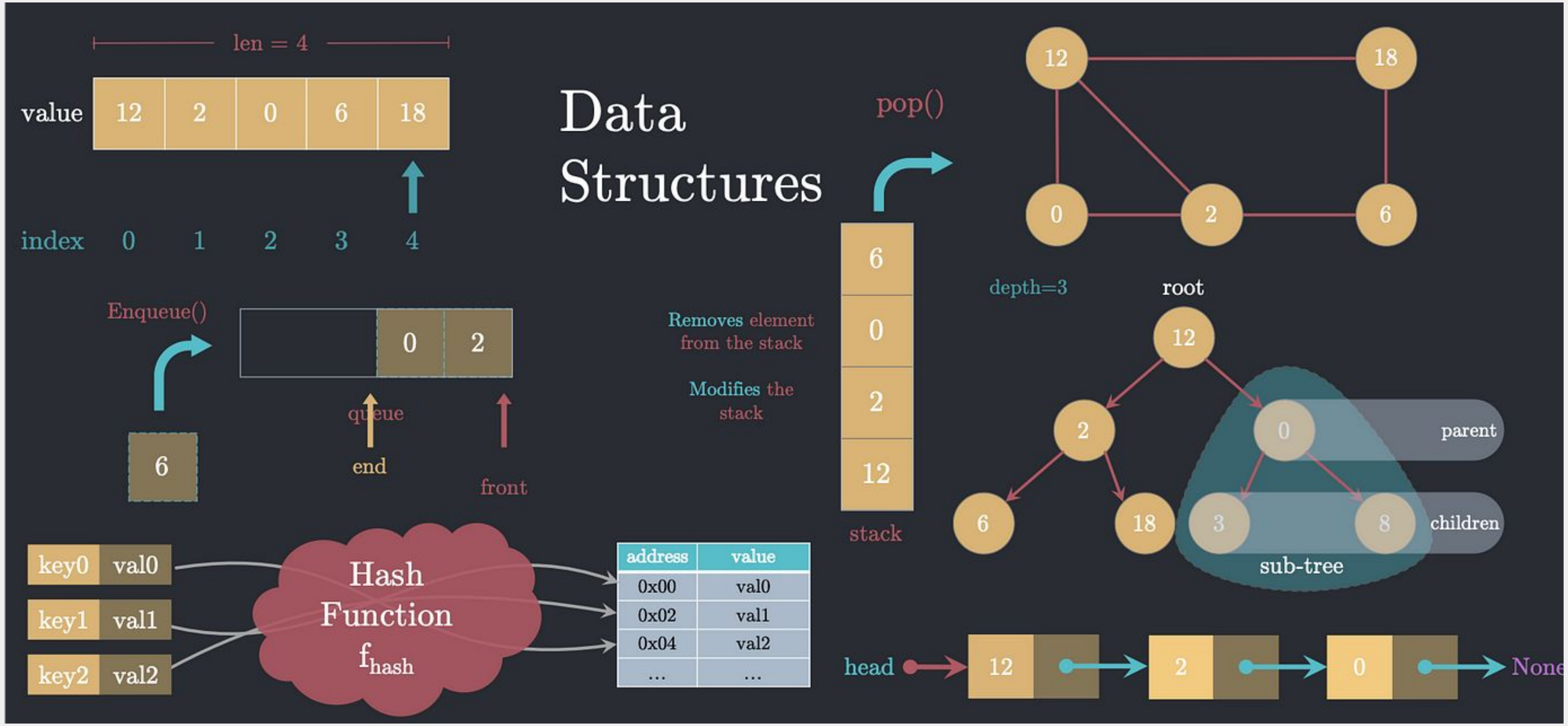
Example: think of other inputs to our program, does it still work with negative numbers? With zero numbers?

This also greatly helps with **test cases**!

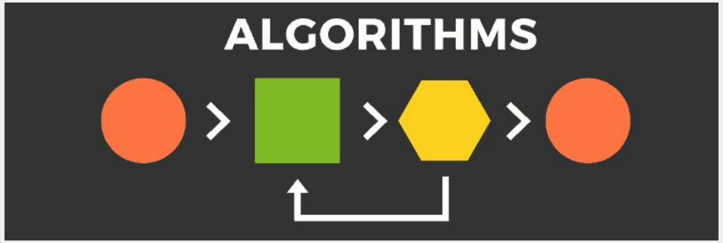
Cut the elephant into pieces

Divide your main **problem into** (ideally independent) **subproblems**

Remember and use your tools



Remember and use your tools



Common algorithms and approaches

- Sorting
- Binary Search
- Sliding window technique
- Two pointers approach
- Union find
- Breadth First Search (2° Practical meeting topic)
- Depth First Search (2° Practical meeting topic)
- Topological sorting

Optimising your solution

1. **Time complexity**
   the time complexity is the **computational complexity** that describes the **amount of** computer **time** it takes to run an algorithm
   a. Typically **more relevant** than space complexity in a coding interview
   b. We will use the **Big-O notation** → O(N), etc.
2. **Space complexity**
   The space complexity of an algorithm or a computer program is the **amount of memory** space required to solve an instance of the computational problemas a function of characteristics of the input. It is the **memory required** by an algorithm until it executes **completely**
   a. Typically **less relevant** than time complexity in a coding interview
   b. We will use the **Big-O notation** → O(N), etc.

Time complexity

We need to identify the **Best Theoretical Time Complexity** (from now on, **BTTC**) of the solution.
The **BTTC** is the the **time complexity that you cannot beat**.

Example:
The **BTTC** of finding the **sum of numbers** in array is O(N) because you have to l**ook at every value** in the array at least once.

NB! The **BTTC doesn't always correspond to the total number of elements in a data structure**. Think at the **Binary search** that uses the fact that the set is ordered to not look at every single element (time complexity O(logN)).

Where am I losing time?

1. **Identify overlapping and repeated computation**
   If your algorithm is doing something **repetitive**, that **you wouldn’t do** if you were to solve it by hand, think of another approach that it’s faster and doesn’t waste that time.

2. **Try different data structures**
   Knowing you data structures means to be able to understand when you need them and when you don’t. **Example**: if you’re struggling with lookup times, you might want to use an **hashMap**

Space complexity

1. **Changing data in-place/overwriting input data**
   If your solution creates a **support data structure** for the input, you can save some space by instead of creating it, **modifying directly the input**. This is discouraged in software engineering (hard to maintain), but **can be used in coding interviews** to reduce space complexity.

2. **Change the data structure**
   As for time complexity, also for space complexity **selecting the correct data structure is crucial** to reduce the complexity to the minimum.

LeetCode, your new best friend

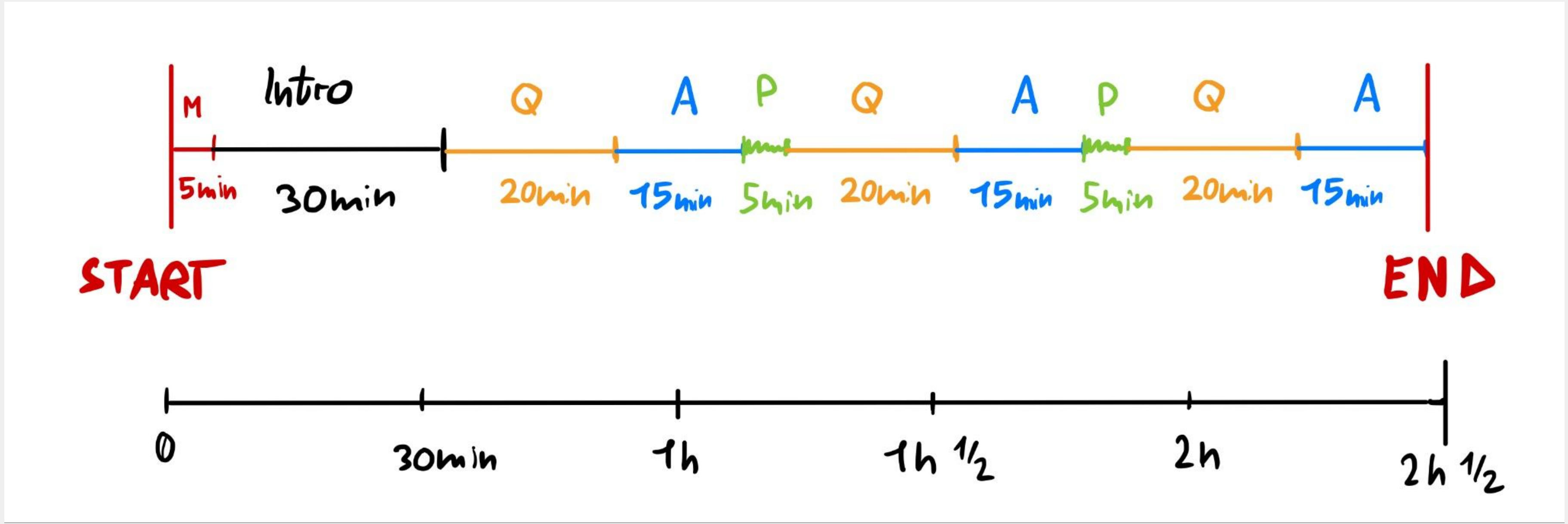
A perfect platform for **practicing coding problems** and to master coding interviews!

Problem difficulty:
As in a videogame, start easy, than go to higher difficulties

General advices

1. Try to **find a solution** even if not efficient
2. **Don’t look at solutions** straight away
3. **Tips** only **after trying**
4. Practice makes perfect
5. Learn from your mistakes
6. It’s more **fun with friends**

Let’s do some questions together!



3, 2, ... 1, CODE!

Your first exercise is **TWO SUM** (See GitHub repo)

GitHub

Start coding now.

3, 2, … 1, CODE!

Your first exercise is **Merge two sorted lists** (See GitHub repo)

GitHub

Start coding now.

3, 2, ... 1, CODE!

Your first exercise is **Top K frequent elements** (See GitHub repo)

GitHub

Start coding now.

T.HANKS everyone!

T.HANKS A LOT!