

ESILV
ENGINEERING SCHOOL
DE VINCI PARIS

BEST TEACHING PRACTICES FOR ENGINEERING STUDENTS

« DIY »

https://chewbii.com/best-teaching-practices-for-engineering-students-diy/

Nicolas.travers@devinci.fr

NICOLAS TRAVERS

Full Professor of Computer Science at ESILV

Deputy director of the DVRC laboratory

Habilitated (HDR) in Computer Science from Sorbonne University in 2018. PhD thesis from Paris Saclay (UVSQ) in 2006.

Main topics: database management and optimization.
Specialization:

- Distributed databases,
- Graph databases,
- Information Retrieval
- Recommendation systems
- Ad hoc databases design in a transdisciplinary approach: Digital Tourism & Geography, Knowledge Management & History, IR & Musicology or Multimodal libraries.



LIKE,SHARE, FOLLOW ! @esilv_paris | esilv.fr
Nicolas.travers@devinci.fr

DIY (V1)

NoSQL course

- Learn how to query different databases / query languages
  - Cassandra
  - MongoDB
  - Elasticsearch
  - Neo4j
- Each course is based on a dataset with
  - Data modeling,
  - Queries to define,
  - Different complexities









Homework

- Each group chooses a unique dataset
  - IMdb, TPC-C, Employees, Grants, SalesDB, Epic Game, Yelp, etc.
- 4 reports / Apply it on each NoSQL database
  - One delivery for each database type
  - Find yourself the interesting queries
  - With corresponding complexities
- Grading
  - Evaluate the correct complexity & diversity of answers
  - Quality of the report
  - Time of delivery
  - External knowledge

Data model (query + explanation of choices)	Nope ***0 points***	so-so ***0.5 points***	good ***1 points***	Excellent ***1.5 points***	
Data import (transformation/cleaning script + queries)	Nope ***0 points***	so-so ***0.5 points***	good ***1 points***	Excellent ***1.5 points***	
Simple Queries : Proper number of queries, Diversity & Difficulty	Nope ***0 points***	So-so ***1 points***	Normal ***1.75 points***	Good ***2.25 points***	Excellent ***3 points***
Complex Queries : Proper number of queries, Diversity & Difficulty	Nope ***0 points***	So-so ***1 points***	Normal ***1.75 points***	Good ***2.25 points***	Excellent ***3 points***
Hard Queries : Proper number of queries, Diversity & Difficulty	Nope ***0 points***	So-so ***1 points***	Normal ***1.75 points***	Good ***2.25 points***	Excellent ***3 points***
Delay of deposit (D-1 < D < after D)	Too late ***0 points***	on-time (the day before) ***0.5 points***	1d in advance ***1 points***		
Report quality (writing, english, presentation/easy to find information, methodology, etc.)	Nope ***0 points***	Normal ***0.5 points***	Excellent ***1 points***		
Course knowledge (new features/external?)	Nope ***0 points***	Normal ***0.5 points***	Excellent ***1 points***		

Feedback:

Each group is unique

- No copy
- More commitment

Generated reports

- Not capturing diversity & the required complexity

Tasks distribution within groups

- Repartition can be added to the grades

Dataset choice

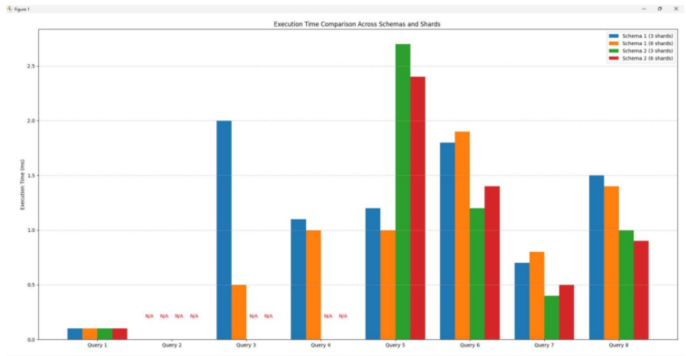
- Some datasets not proper for a given database

LIKE,SHARE, FOLLOW ! @esilv_paris | esilv.fr
Nicolas.travers@devinci.fr

DIY (V2)

Big Data Structure course

- Learn how to query scale up a database on the cloud
  - Data model transformation
  - MongoDB advanced queries
  - Sharding Strategy & cost estimation
  - Cloud deployment



Homework

- Each group chooses a unique dataset
  - IMdb, TPC-C, Employees, Grants, SalesDB, Epic Game, Yelp, etc.
- 4 reports delivery for choices on each step
  - 3 different data models (for the chosen dataset)
  - Translate queries for each data model
  - Estimate costs for each
  - Final challenge with scaling (up to 120 VM)
    - Try each data model (transform the dataset)
    - See evolution of
- Same grading strategy with a presentation

Data models performances (time charts for queries & shards)	Nope ***0 points***	1 or 2 queries ***0.5 points***	4 queries ***1 points***	All queries ***1.5 points***	All queries on both denormalization ***2 points***
Maximum nb of shards during experiments	0-1 ***0 points***	2-3 ***0.5 points***	4-5 ***1 points***	6-7 ***1.5 points***	8+ ***2 points***
Project management, balanced tasks, Git, detailed architecture, scripts	Nope ***0 points***	So-so ***0.5 points***	Good ***1 points***	Very good ***1.5 points***	Excellent ***2 points***
Administration statistics (distribution, clusters, size, indexes, etc.)	Nope ***0 points***	So-so ***0.5 points***	Good ***1 points***	Very good ***1.5 points***	Excellent ***2 points***
Delay of deposit	Too late ***0 points***		on-time (the 4 days before) ***0.5 points***		4d in advance ***1 points***
Report quality (writing, english, presentation/easy to find information, methodology, etc.)	Very poor ***0 points***		Normal ***0.5 points***		Excellent ***1 points***

⇒ Testing students limits in experimenting

⇒ Show their understanding of the behavior of a system

⇒ Project management

Nicolas.travers@devinci.fr

DIY (V3)

Graph Mining course

- Learn how to query different databases / query languages
  - Neo4j & GDS
  - Graph Data Modeling
  - Cypher Queries & GDS algorithms
  - => target knowledge extraction

Homework

- Choose a unique dataset per group
  - DBLP, Rddit, Wikispedia
  - Generate 2 graph data models (mono vs bi-partite graphs)
  - Find yourself the knowledge
    - Apply different Cypher queries using GDS
    - Provide unexpected knowledge (interpret results)
- Same grading strategy with a presentation

```
1
2  CALL gds.graph.project(
3      'author-paper-graph',
4      ['Author', 'Paper'],
5      {
6          AUTHORED: {
7              orientation: 'UNDIRECTED'
8          }
9      }
10 );
11
12 CALL gds.pageRank.stream('author-paper-graph')
13 YIELD nodeId, score
14 MATCH (n) WHERE id(n) = nodeId AND n:Author
15 RETURN n.name AS author, score
16 ORDER BY score DESC
17 LIMIT 10;
18
```

Query description

We apply the PageRank algorithm on the author-paper-graph, which includes authors and their published papers. It calculates the PageRank scores for each author to measure their influence in the network, returns the top 10 most influential authors along with their scores, and sorts them in descending order.

```
CALL gds.graph.project(
  'filmGraph',
  ['Title'],
  {
    SHARES_GENRE_YEAR: {
      type: 'SHARES_GENRE_YEAR',
      orientation: 'UNDIRECTED'
    }
  }
) YIELD graphName, nodeCount, relationshipCount;
```

graphName	nodeCount	relationshipCount
"filmGraph"	35145	1781384

```
CALL gds.labelPropagation.write(
  'filmGraph',
  {writeProperty: 'communityId'}
) YIELD communityCount;
```

communityCount
19900

```
MATCH (t:Title)
WHERE t.communityId IS NOT NULL
RETURN t.communityId AS community, COUNT(t) AS communitySize
ORDER BY communitySize DESC;
```

The results show that the biggest group has 271 movies, and the other groups are smaller. This means some movies are more closely connected by genre and release year. The smaller groups show that many movies have fewer connections, showing differences in genres or years.

We couldn't use a 300 nodes community because it had 16000 relations, which made our computers crashed, so we did it for 50

LIKE,SHARE, FOLLOW ! @esilv_paris | esilv.fr
Nicolas.travers@devinci.fr

DIY (V4)

Big Data Structure course at Polytechnic (Paris Saclay)

- Learn how to query scale up a database on the cloud
  - Data model transformation
  - Sharding Strategy
  - Cost estimation
    - Data storage
    - Filter queries
    - Join queries
    - Aggregate queries
    - Global cost model

Homework

- Target on the cost model estimation
  - Produce a program which computes the cost
    - Using the given input types and output required
  - They can use an LLM (but smartly)
  - Try with practice works examples (queries)
- Final challenge
  - Give new queries and settings
  - The program must work adapt itself

- I do not grade
  - The code
  - The presentation of the code
- I grade
  - Only the questions I ask during the challenge (on the code itself)

```
    # size for outer filter
    outer_doc_size = _compute_query_doc_size(outer_coll, outer_fields)

    outer_op = FilterOperator(
        collection=outer_coll,
        output_fields=outer_fields,
        filter_key=filter_key,
        sharding_key=outer_shard,
        num_servers=num_servers,
        query_size=args.query_size,
        selectivity=args.selectivity,
        output_doc_size=outer_doc_size,
    )

    join_key = parsed["join_key"]
    inner_shard = sharding_map.get(inner_name, join_key)

    # size for inner lookup
    inner_doc_size = _compute_query_doc_size(inner_coll, inner_fields)

    join_op = NestedLoopJoinOperator(
        outer_filter=outer_op,
        inner_collection=inner_coll,
        inner_output_fields=inner_fields,
        join_key=join_key,
        inner_sharding_key=inner_shard,
        num_servers=num_servers,
        inner_query_size=40,
        inner_selectivity=args.inner_selectivity,
        inner_output_doc_size=inner_doc_size,
    )
    res = join_op.execute()
    _format_query_result(output_lines, parsed, res)

elif parsed["type"] == "AGGREGATE":
    table_name = parsed["table"]
    if table_name not in collections:
        print(f"Error: table '{table_name}' not found in schema")
        sys.exit(1)
    coll = collections[table_name]

    out_fields = parsed["fields"]
    if out_fields == "*":
        out_fields = [f.name for f in coll.fields]

    filter_key = parsed["where_keys"]
    if len(filter_key) == 1:
        filter_key = filter_key[0]
```

⇒ Course notion assimilation

⇒ Deep understanding

⇒ Lost of LLM

- ⇒ Code explanation

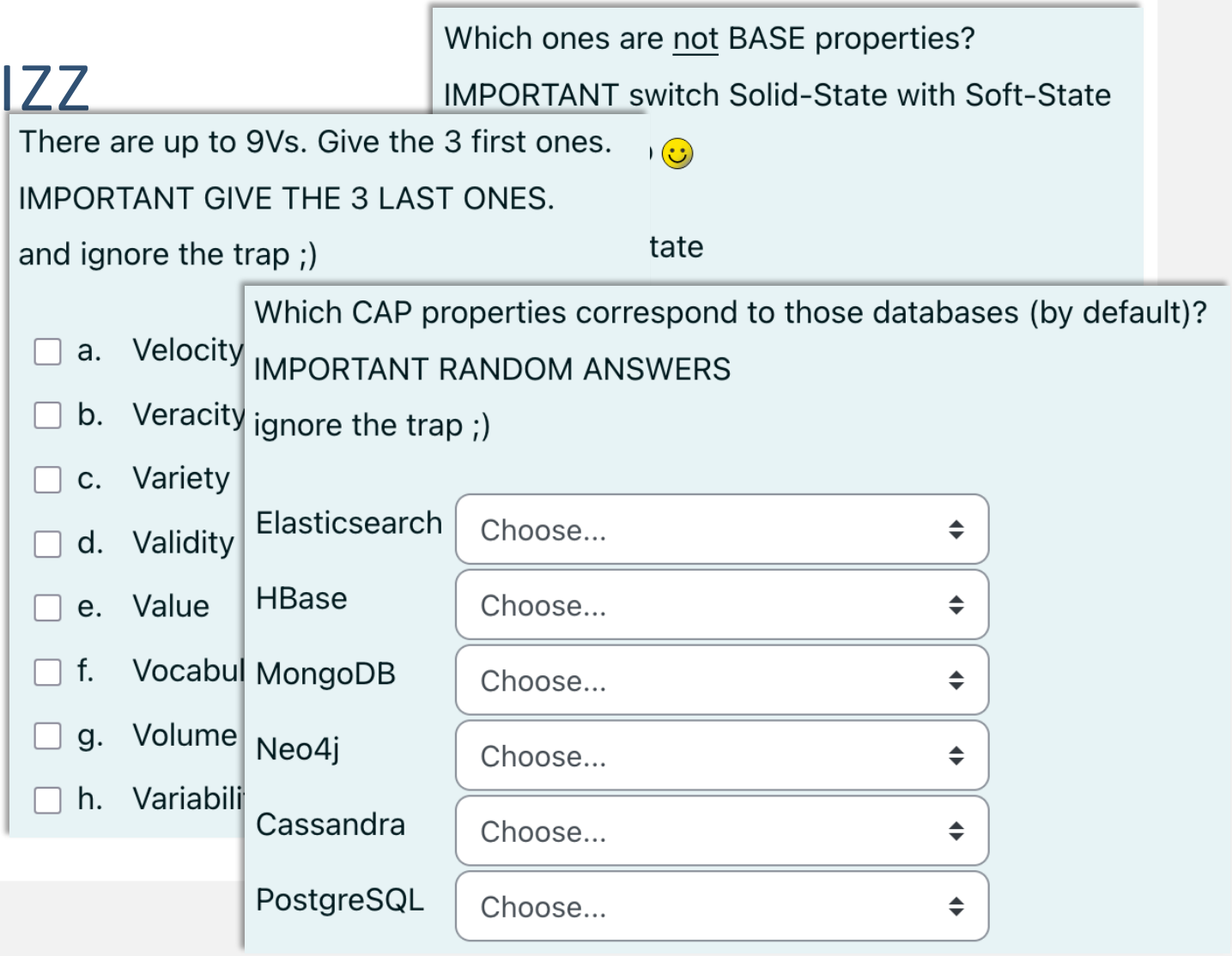
⇒ Challenge: generalisation

LIKE,SHARE, FOLLOW ! @esilv_paris | esilv.fr

Nicolas.travers@devinci.fr

ONLINE QUIZZ

- Traps for LLM
  - Warn students in advance
  - Tight time + trap
  - Negation with “IMPORTANT”
  - Gray instructions in images
  - Images with algo behavior



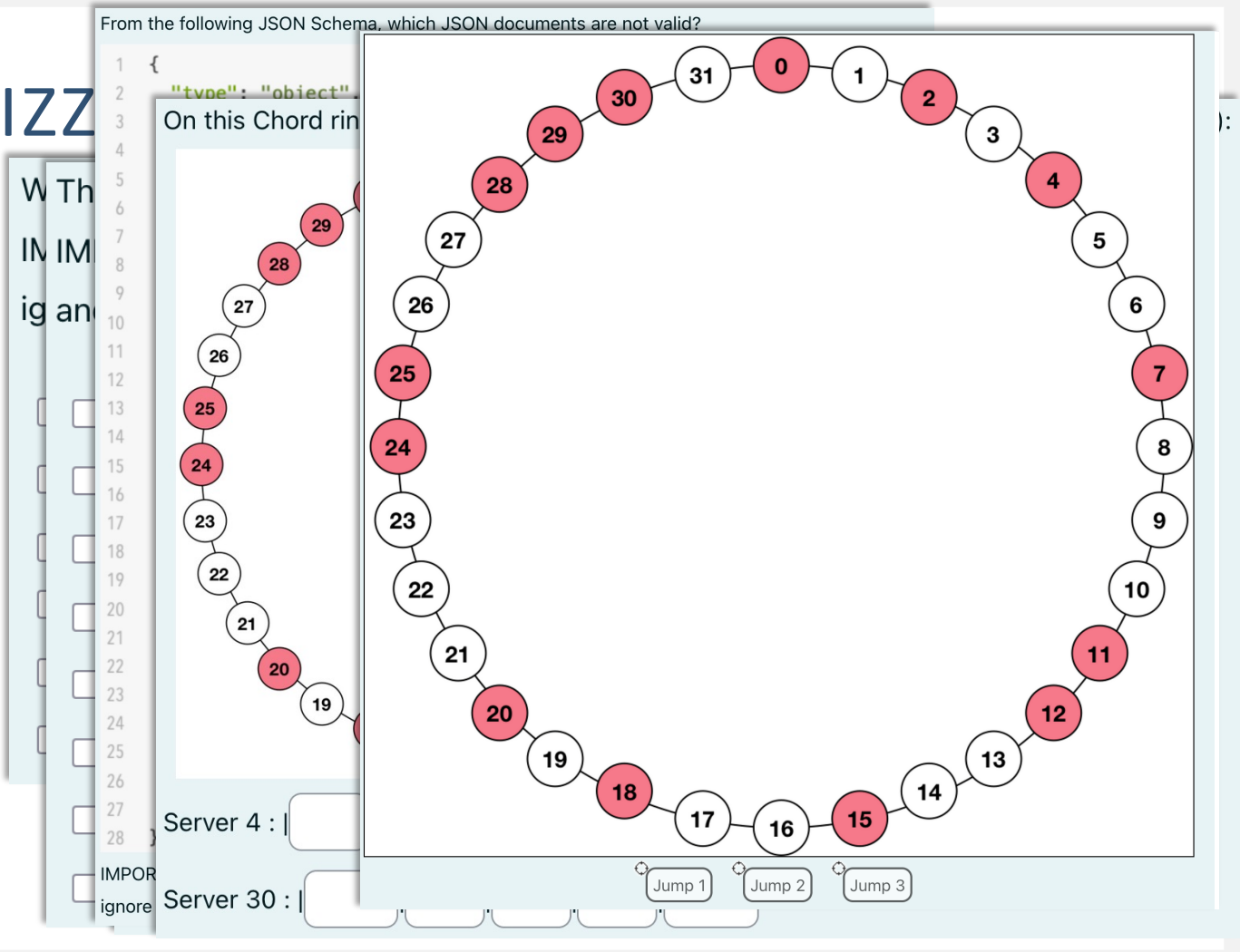
ONLINE QUIZZ

- Traps for LLM
  - Warn students in advance
  - Tight time + trap
  - Negation with “IMPORTANT”
  - Gray instructions in images
  - Images with algo behavior



ONLINE QUIZZ

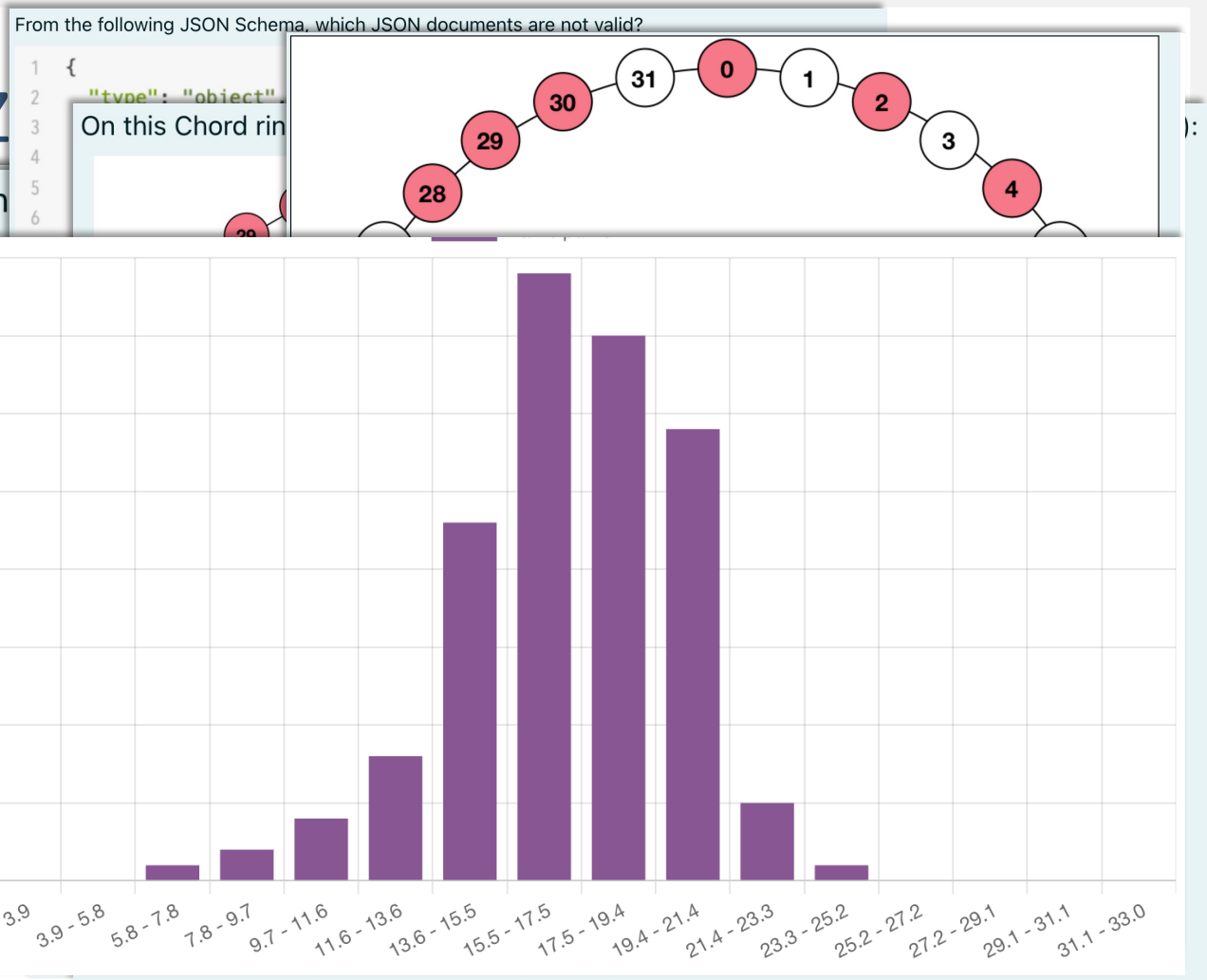
- Traps for LLM
  - Warn students in advance
  - Tight time + trap
  - Negation with “IMPORTANT”
  - Gray instructions in images
  - Images with algo behavior



LIKE,SHARE, FOLLOW ! @esilv_paris | esilv.fr

ONLINE QUIZZ

- Traps for LLM
  - Warn students in advance
  - Tight time + trap
  - Negation with “IMPORTAN
  - Gray instructions in images
  - Images with algo behavior



Grades distribution

On trap questions:

- +30% wrong answers
- +80% de standard deviation