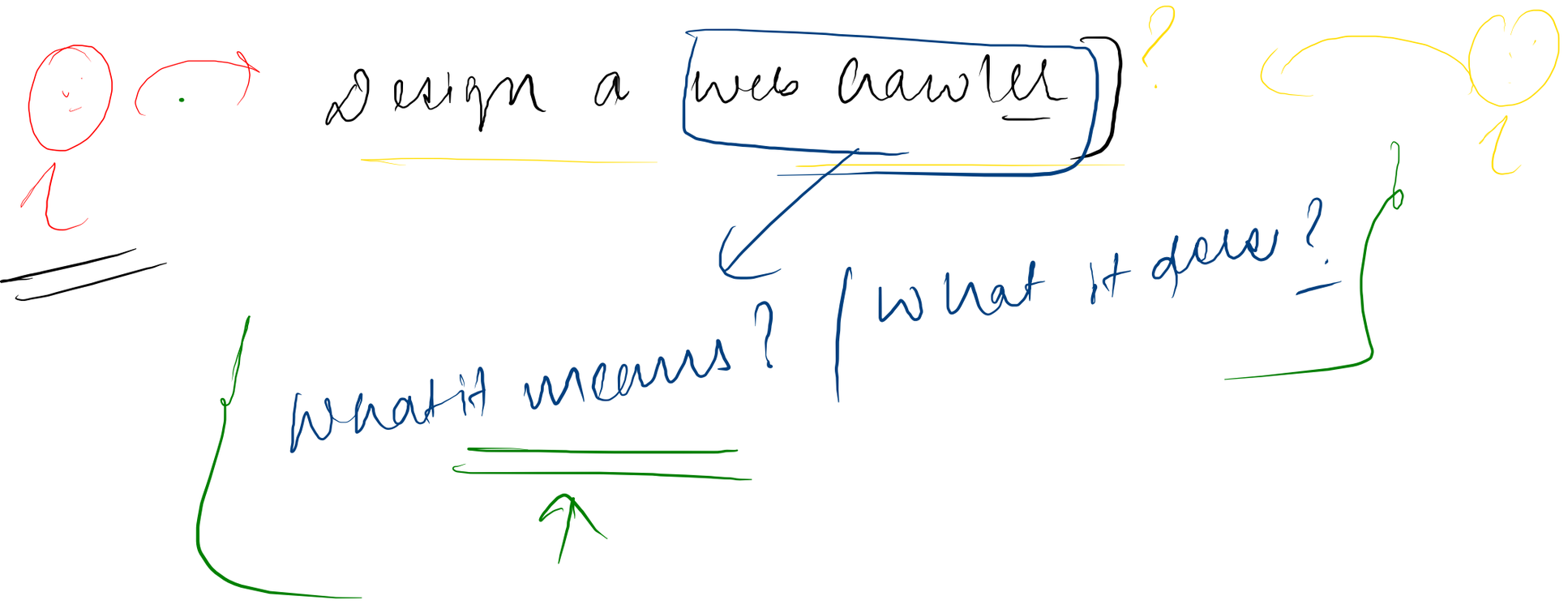
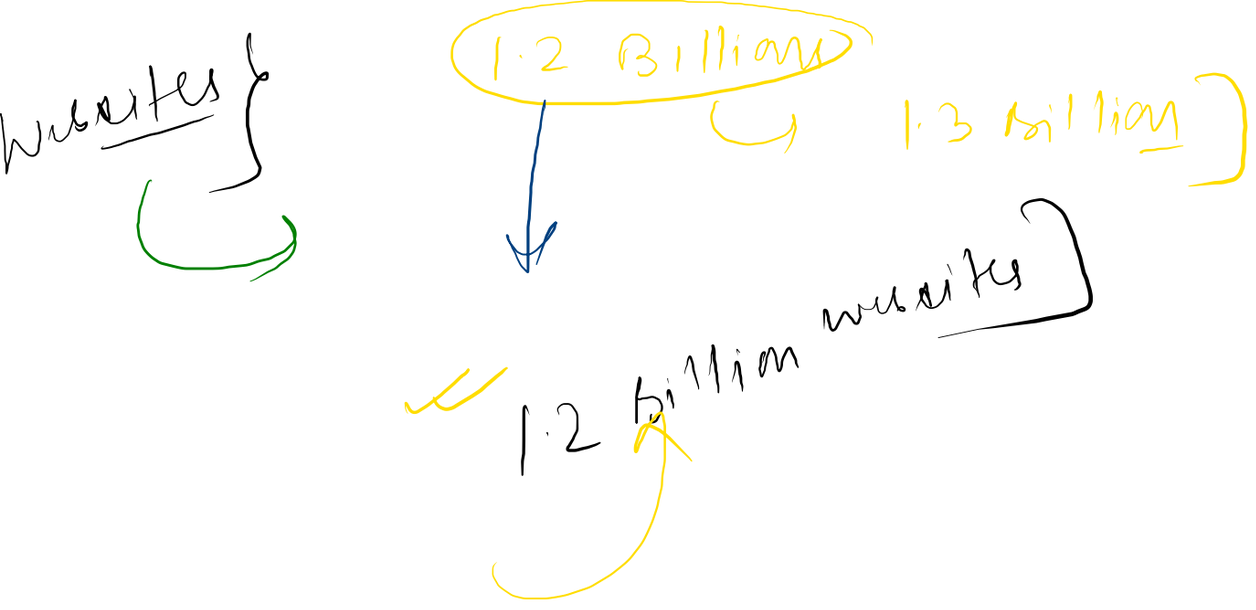
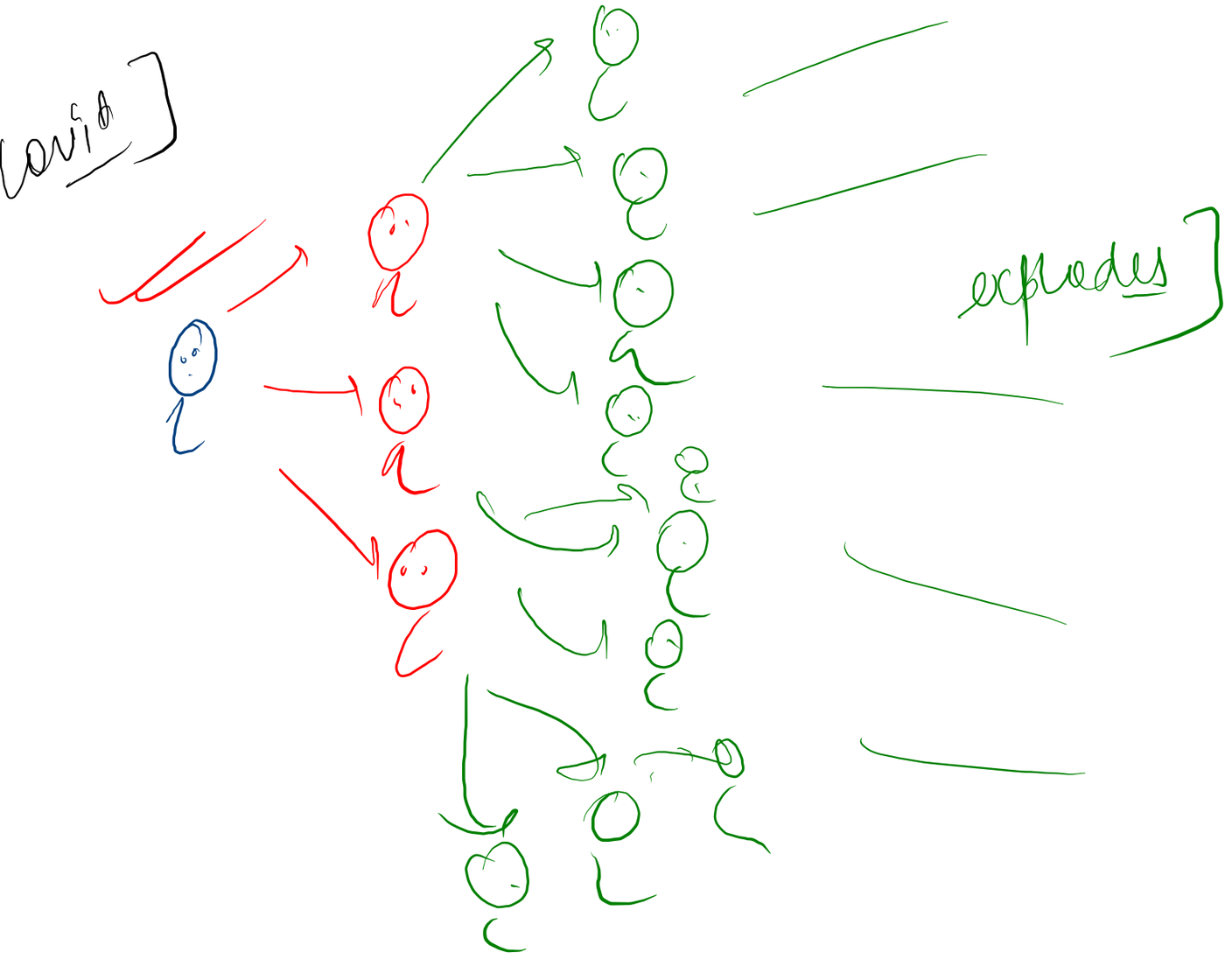
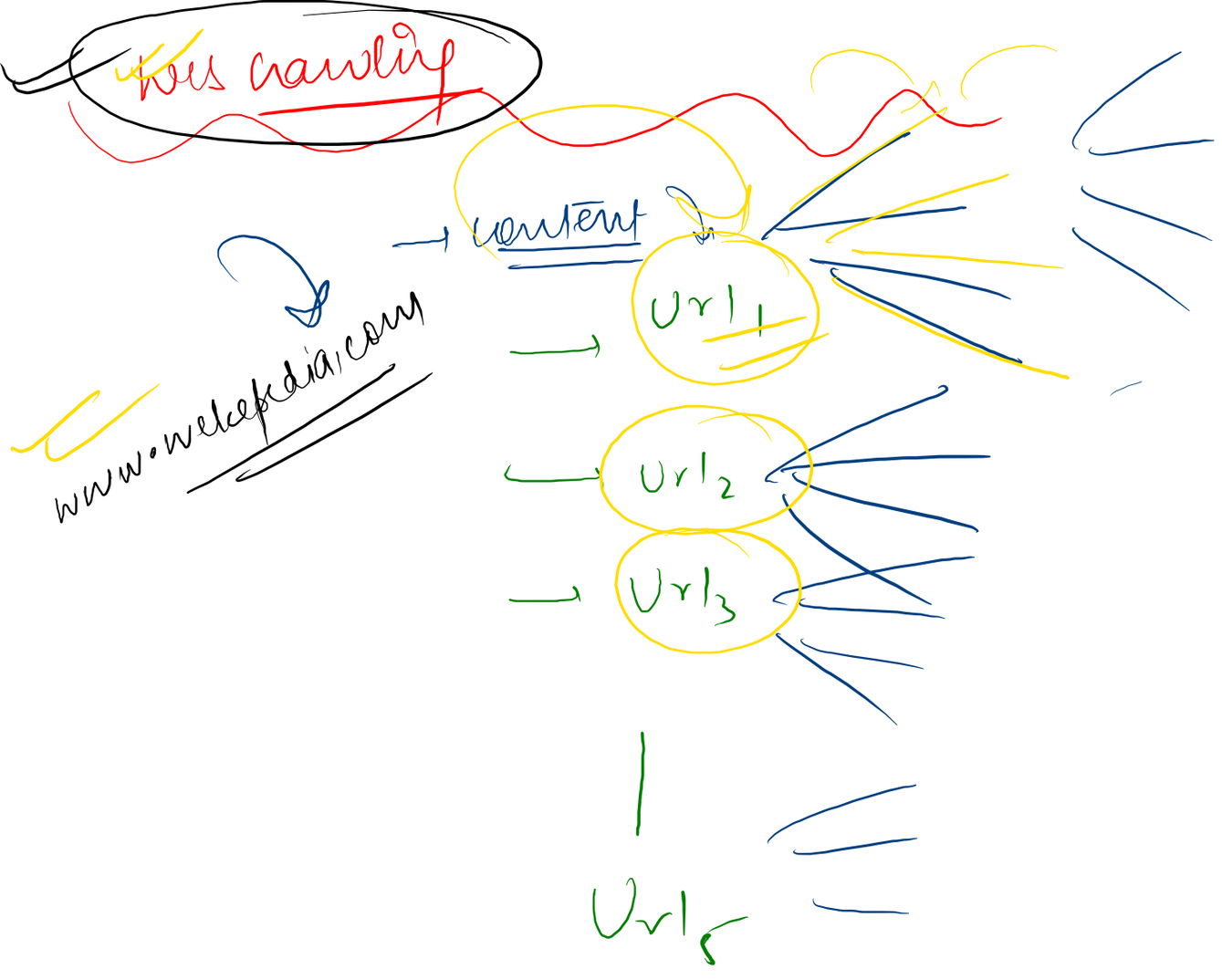
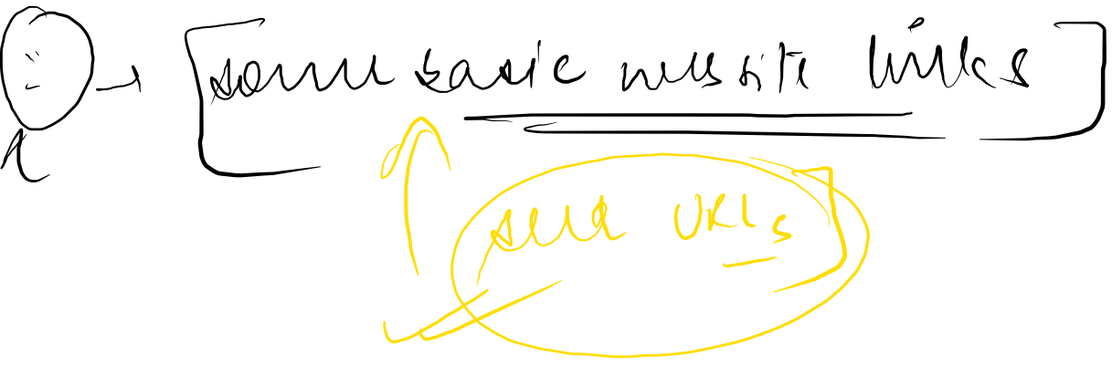

Design a web crawler ?
what it means ?
what it does ?

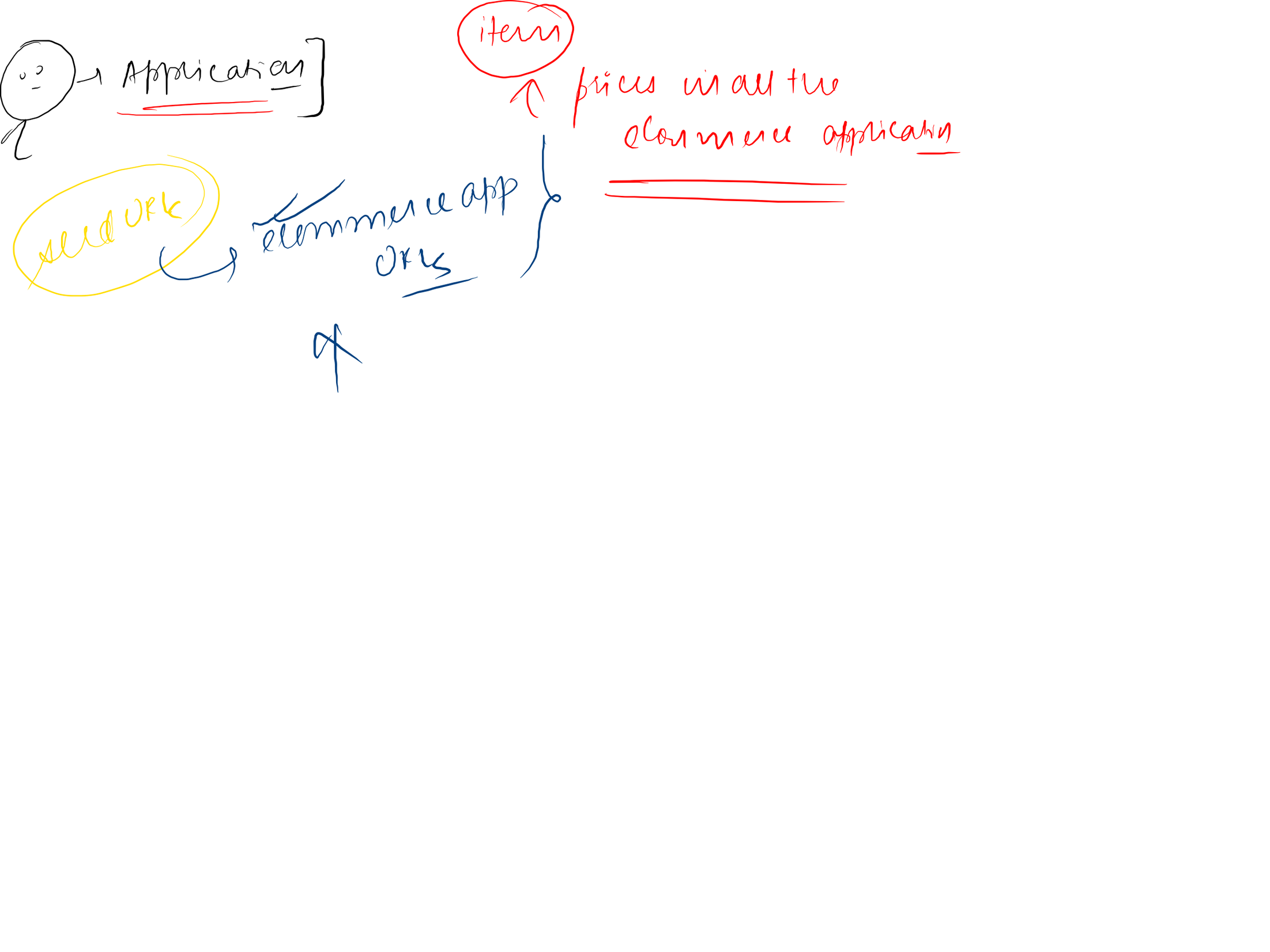

web crawler ?


writes
1·2 Billion
1·3 billion
1·2 billion writes


explodes


Web crawling
content
www.wikipedia.com
Url1
Url2
Url3
Url5


some basic website links
all URLs


Application
items
prices in all the
ecommerce applications
seed URL
ecommerce app
URLs

Web Crawling

1. Search engine → Google, Bing — Malware detection
2. Copyright violation detection / identifying the copying of reports — plagiarism
3. Data science → Data fetching
4. Keyword based search

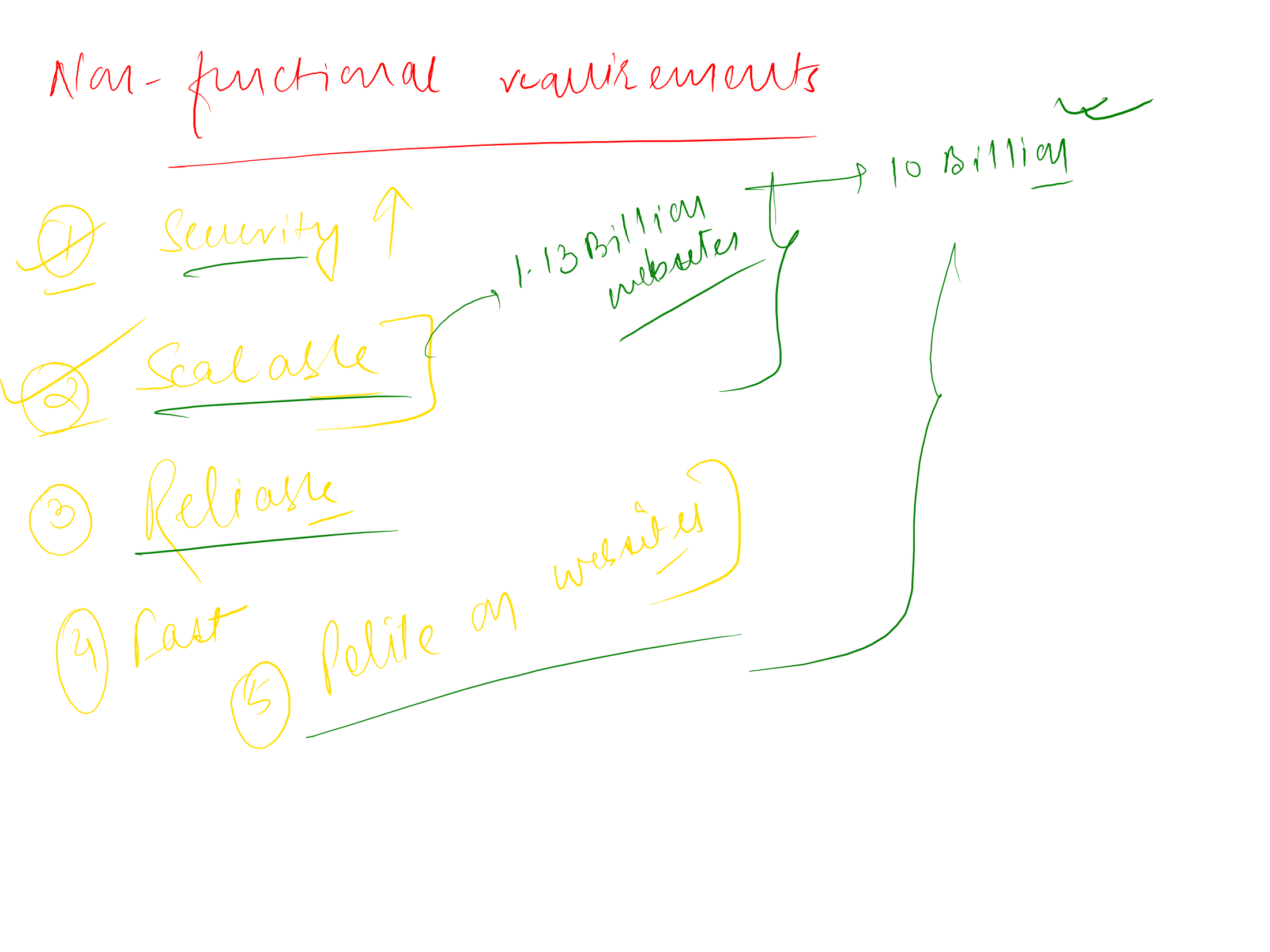
→ Build this system

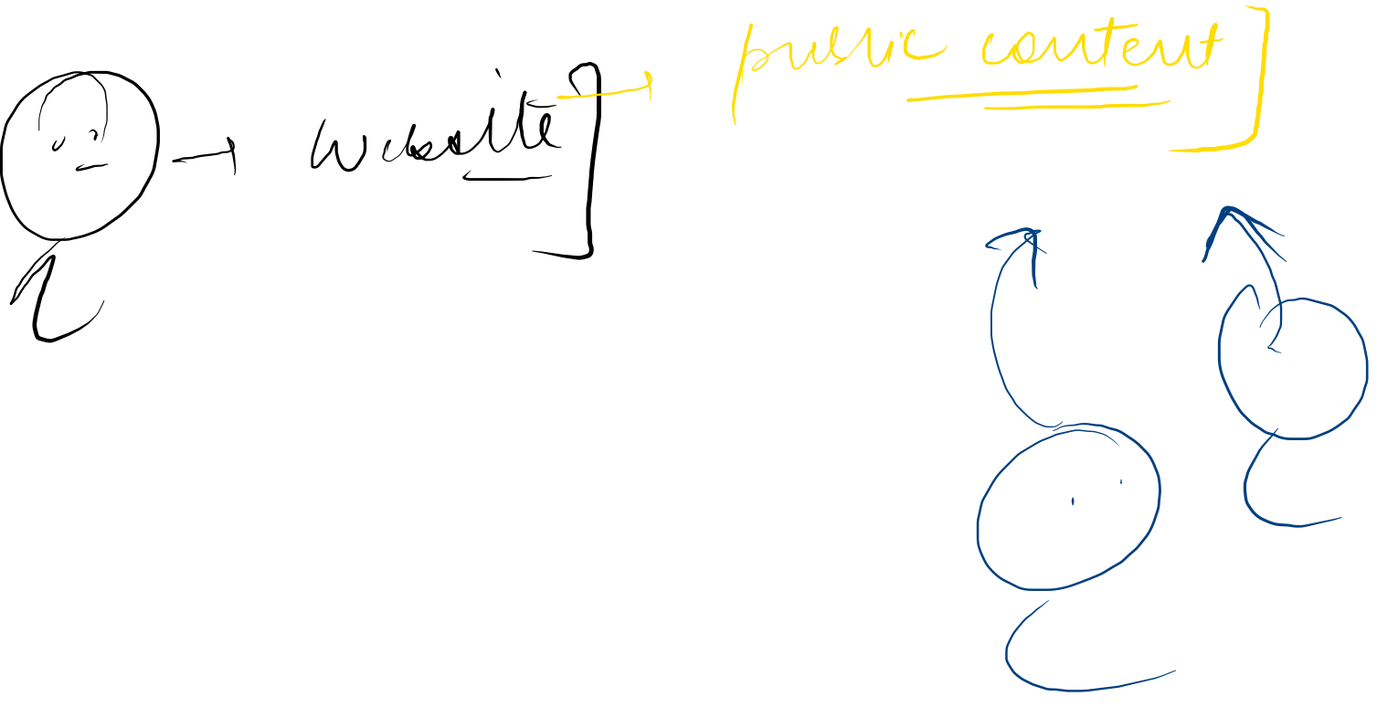
Functional Requirement

1. Scan and store data for all the websites
2. Don't store duplicate records

Non-functional requirements

1. Security ↑
2. Scalable → 1.13 Billion users → 10 Billion
3. Reliable
4. Fast
5. Polite on websites




website
public content

Guesstimating

↳ estimate the scale of the problem

Synopsis

Total no of website = 1.2 Billion

60% of website are active

↳ $\frac{60}{100} \times 1.2$ Million

= 720 Million

Facebook
Wikipedia

→ Average → No of web pages / website = 100

∴ Total no of webpages/URLs = 720 Million × 100

= 72000 Million

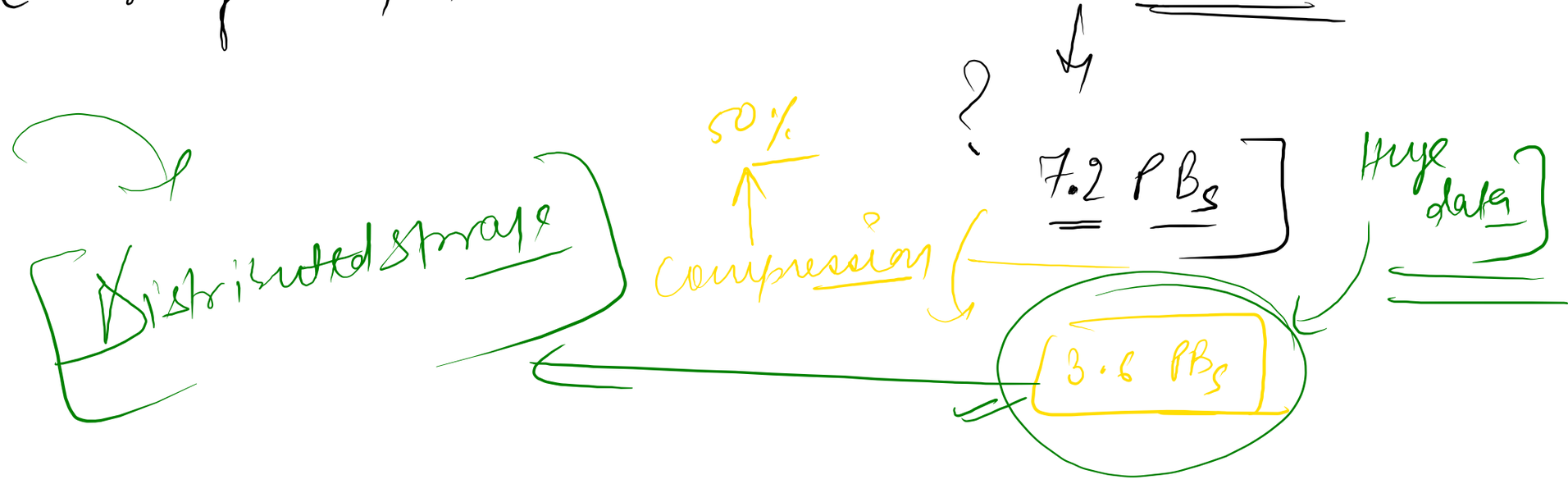
= 72 Billion

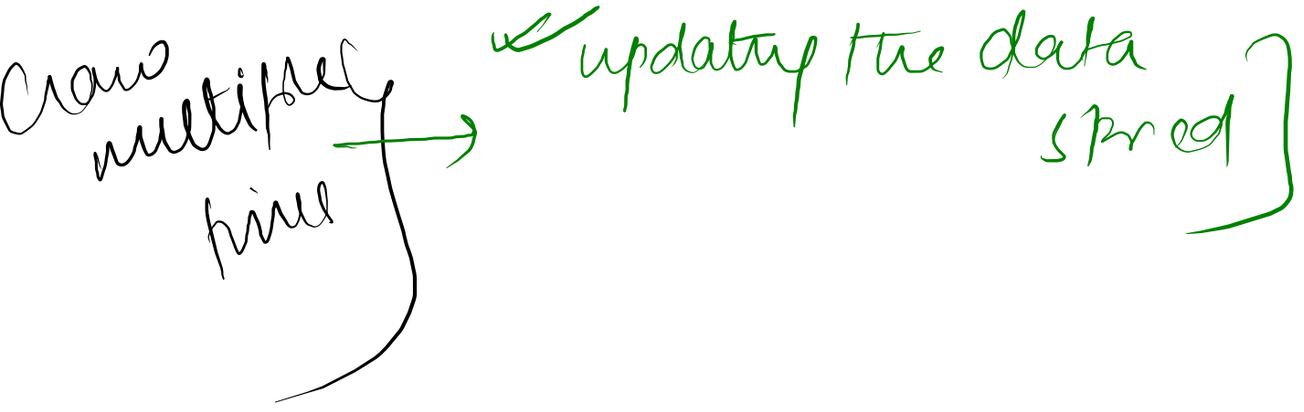

Crawler
72 Billian URLs
Huge Number
Crawler
Distributed
storing text data
storing image/video files
for new

Avg size of 1 webpage = 100KB

Nog pages = 72 billion

total size of webpage to be stored = 72 billion x 100 KB




Can multiple
pins
updating the data
shared

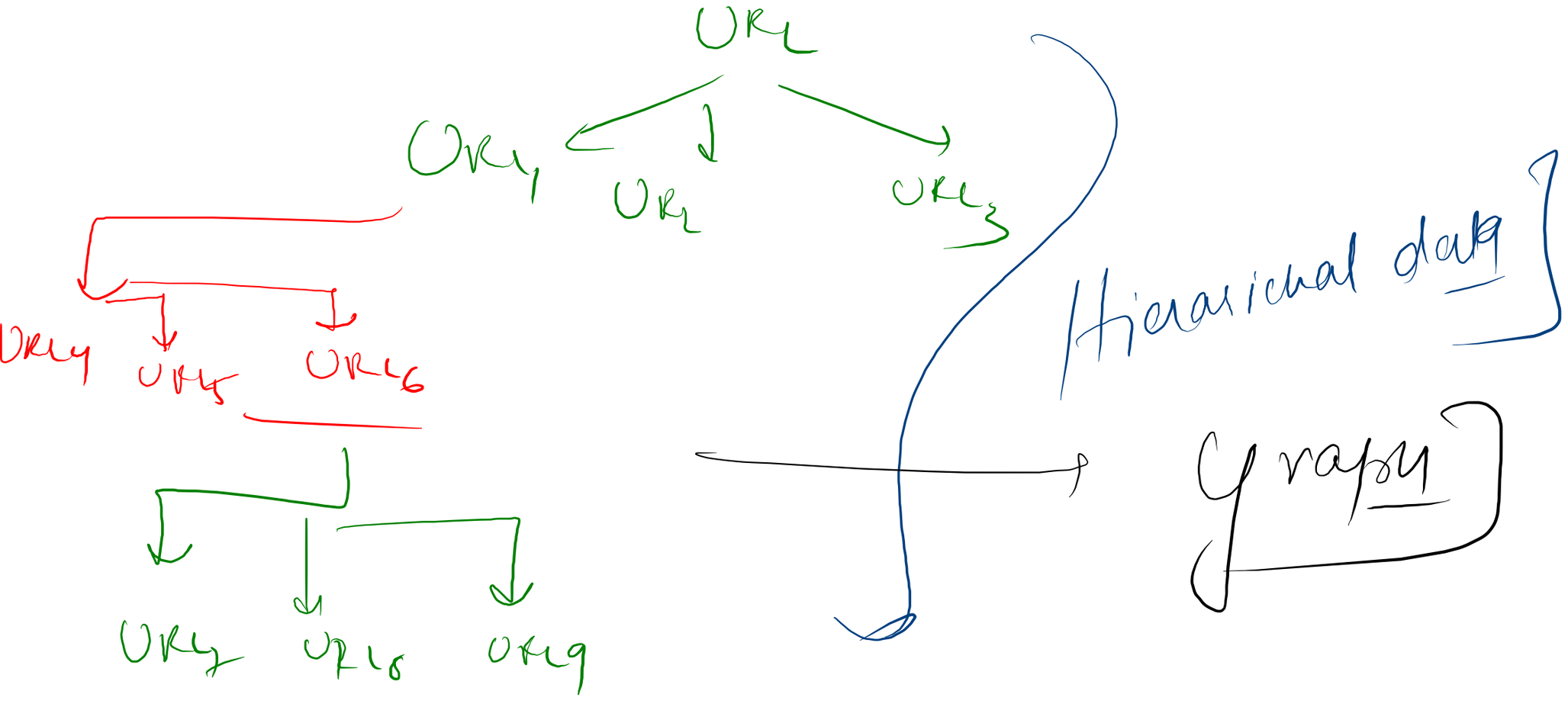
FR / NFR

Scale / Numbers

seed URLs →

- URL1
- URL2
- URL3
- —

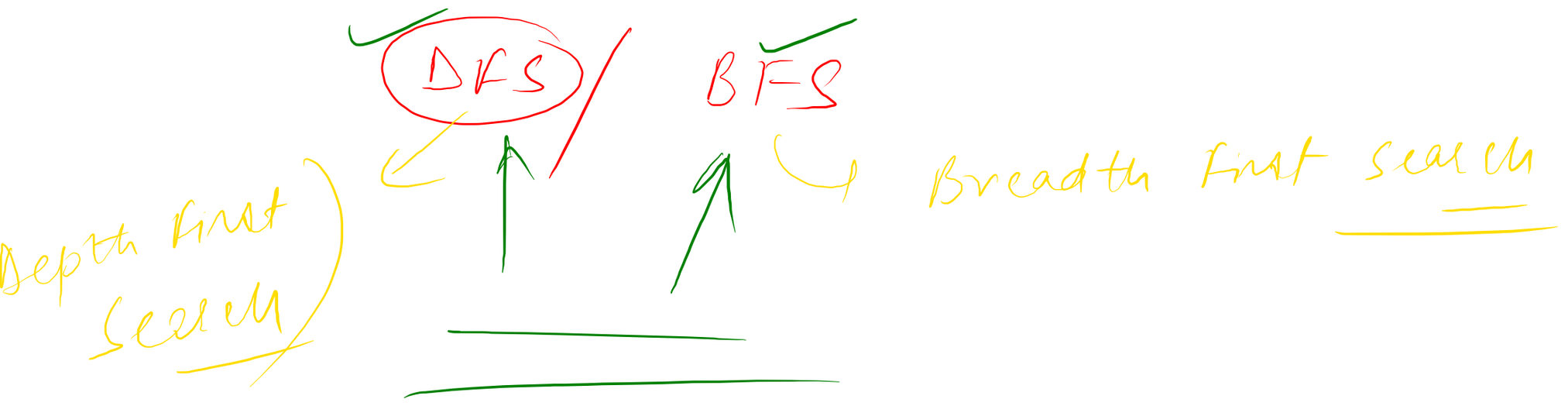
→ Visit all other URLs


URL
URL1
URL2
URL3
URL4
URL5
URL6
URL7
URL8
URL9
Hierarichal data
Graph

Complex Graph

Undirected Graph

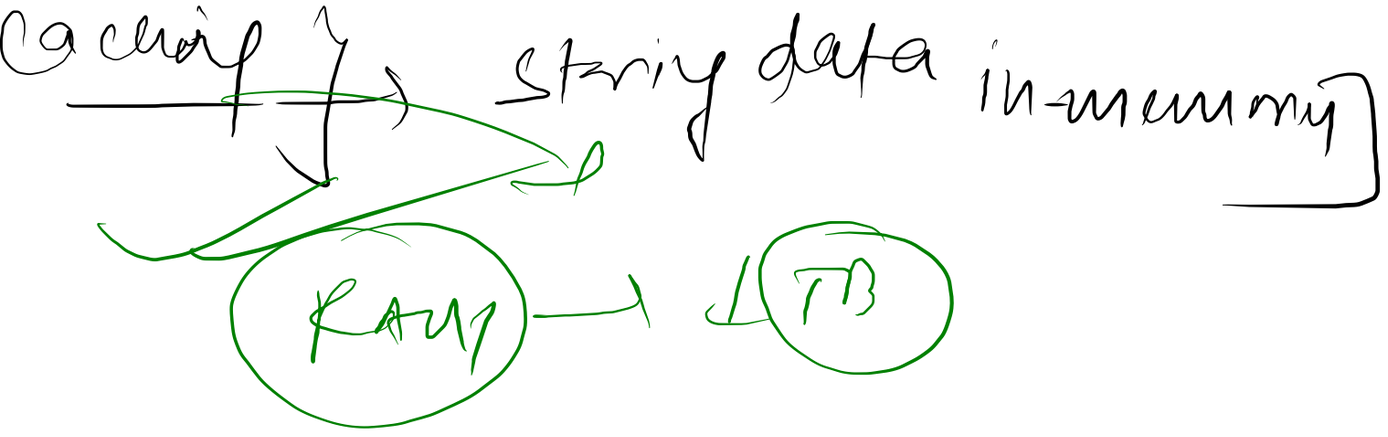
Graph traversal

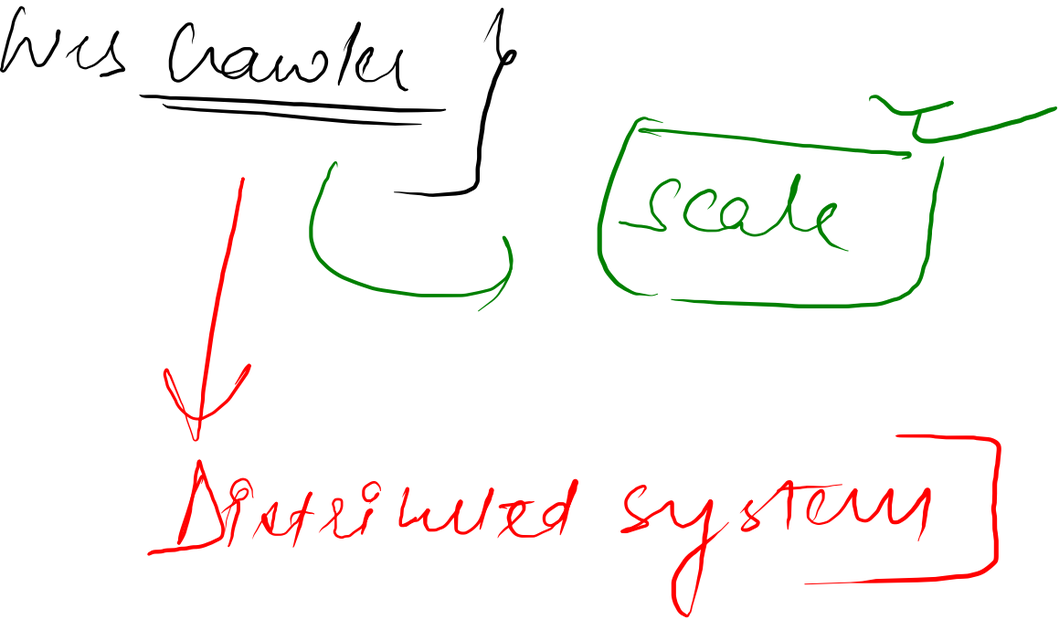

DFS
BFS
Depth first search
Breadth first search

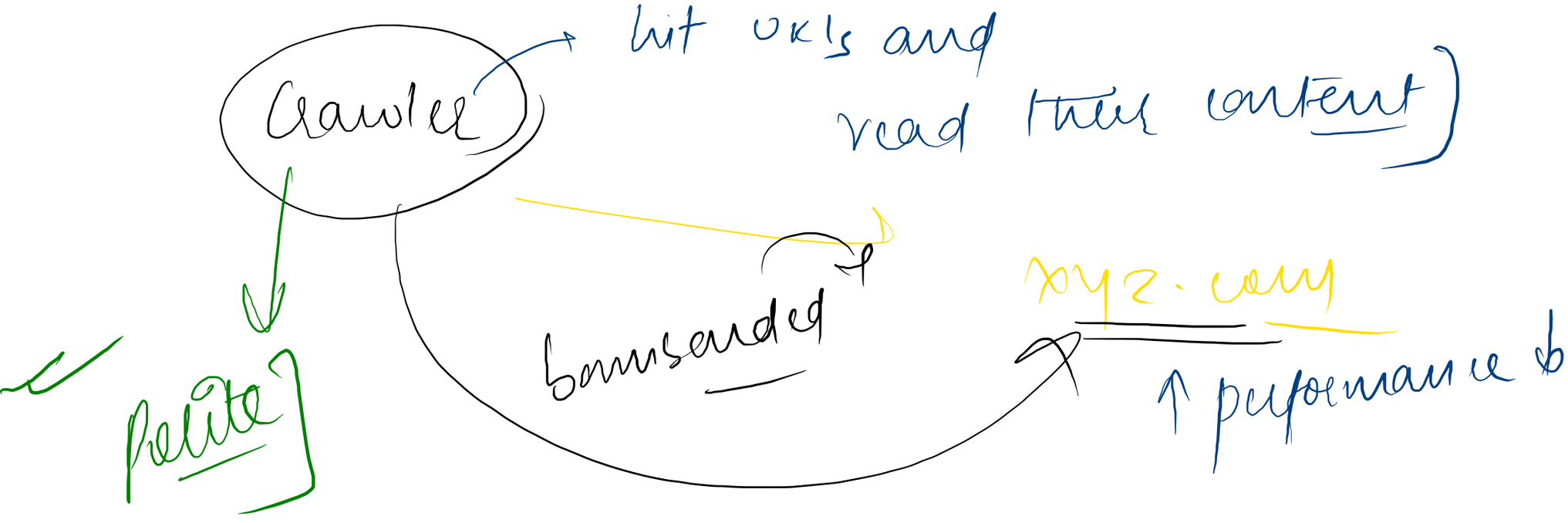
BFS ?

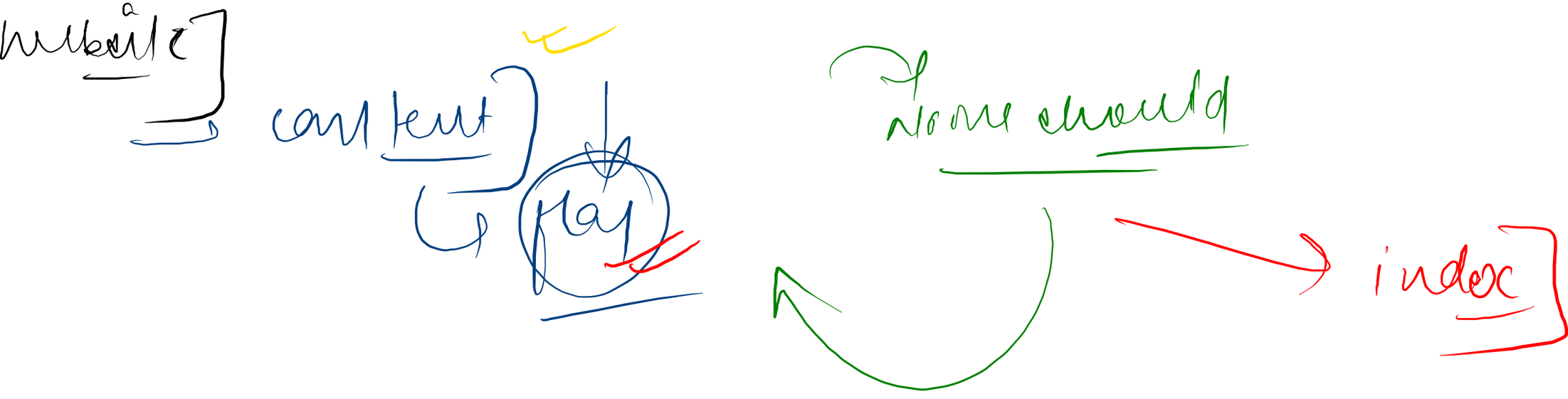
(DFS) ?

Both of time return all possibilities eventually

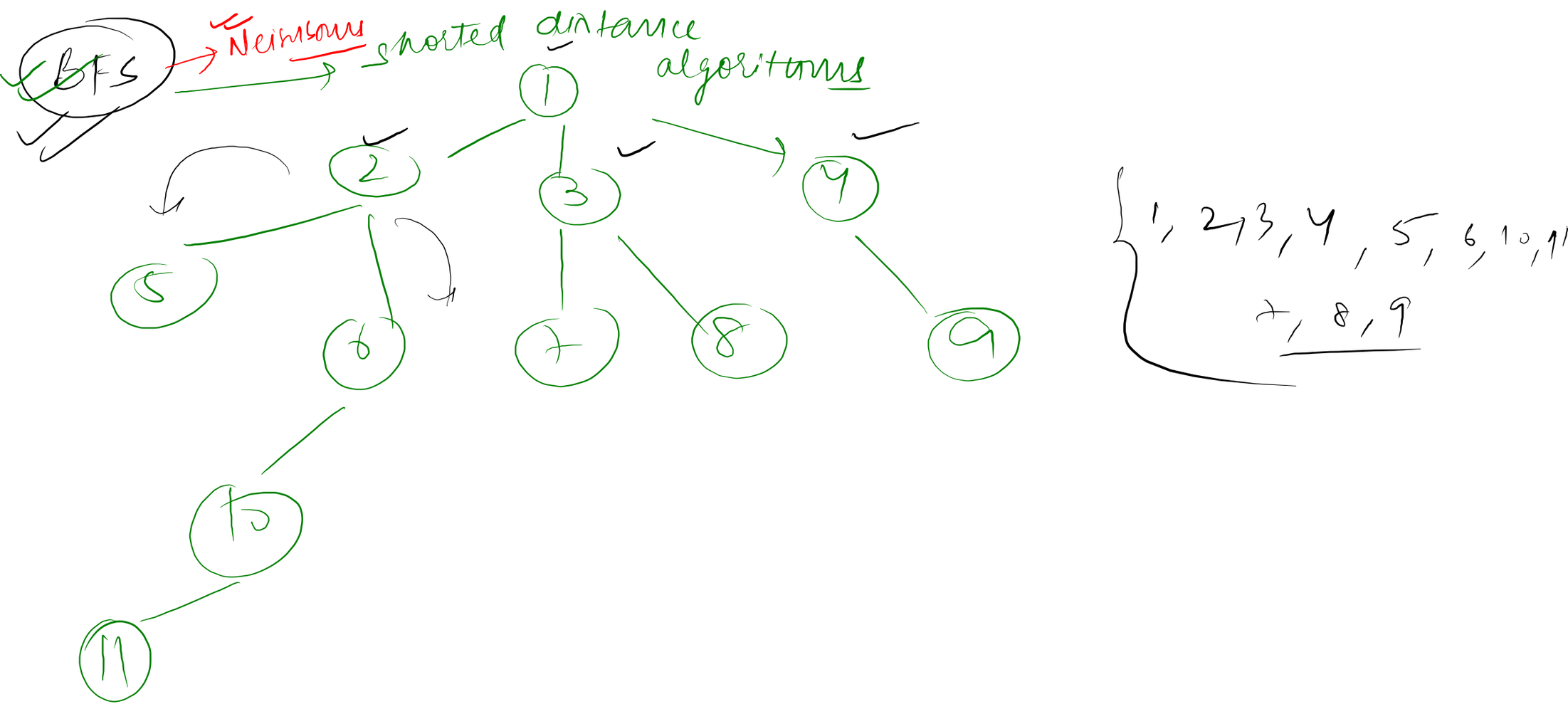

string data in-memory
RAM
1TB

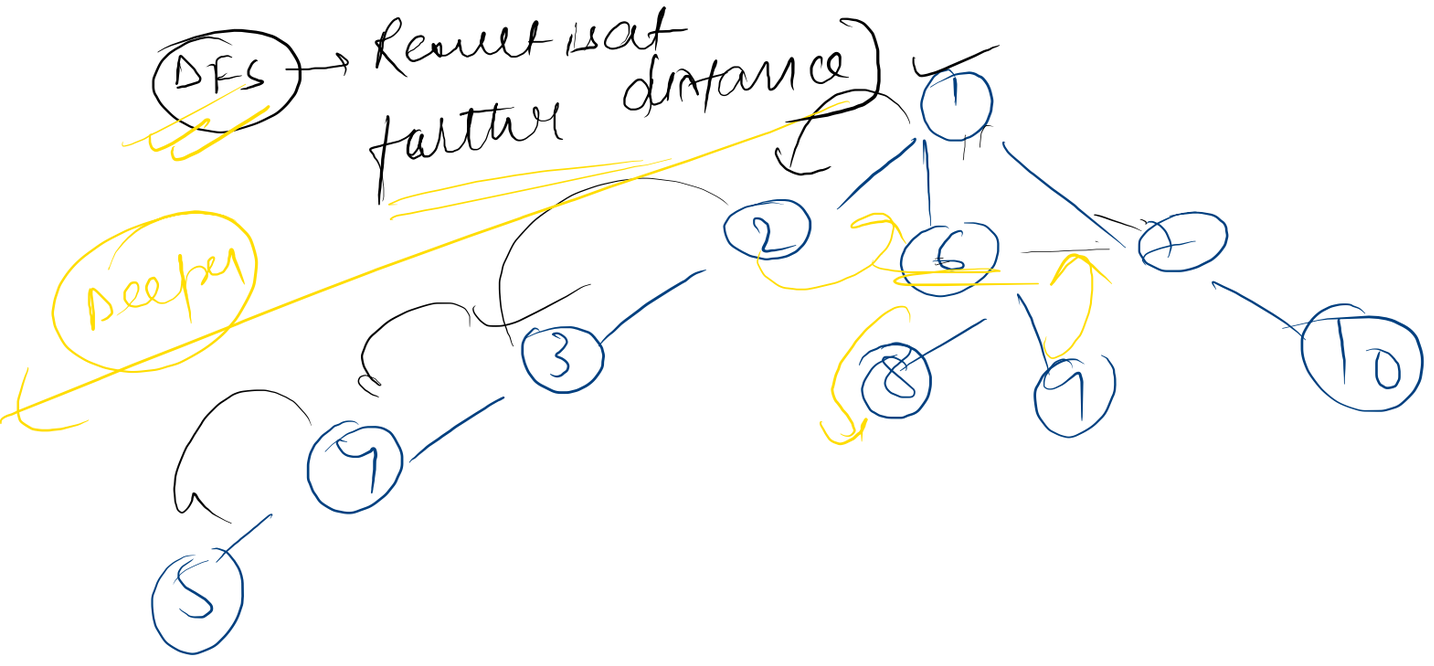

Web Crawler
scale
Distributed system


Crawler
hit URLs and
read their content
bombarded
xyz.com
↑ performance
politely

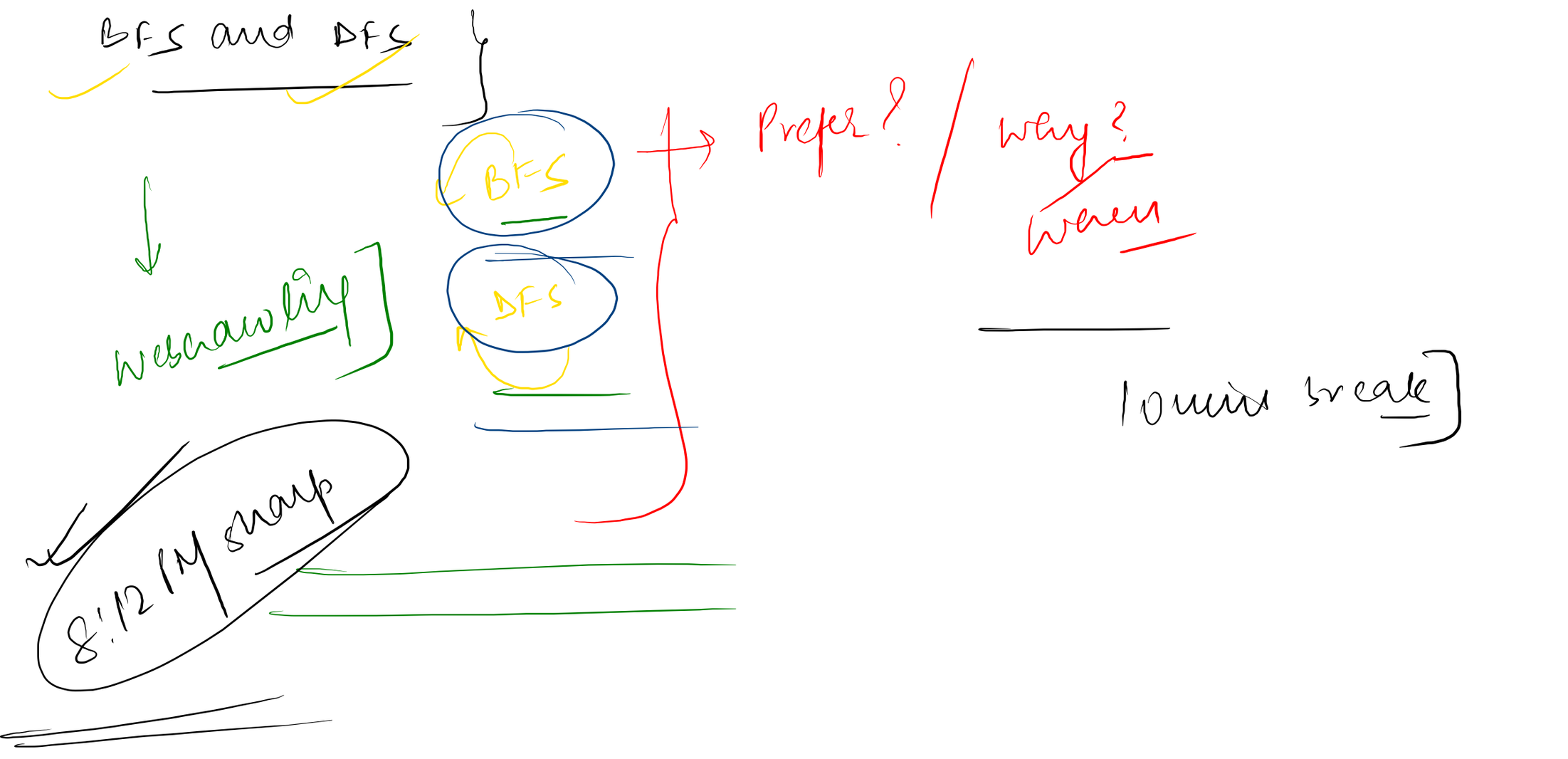

content
play
index

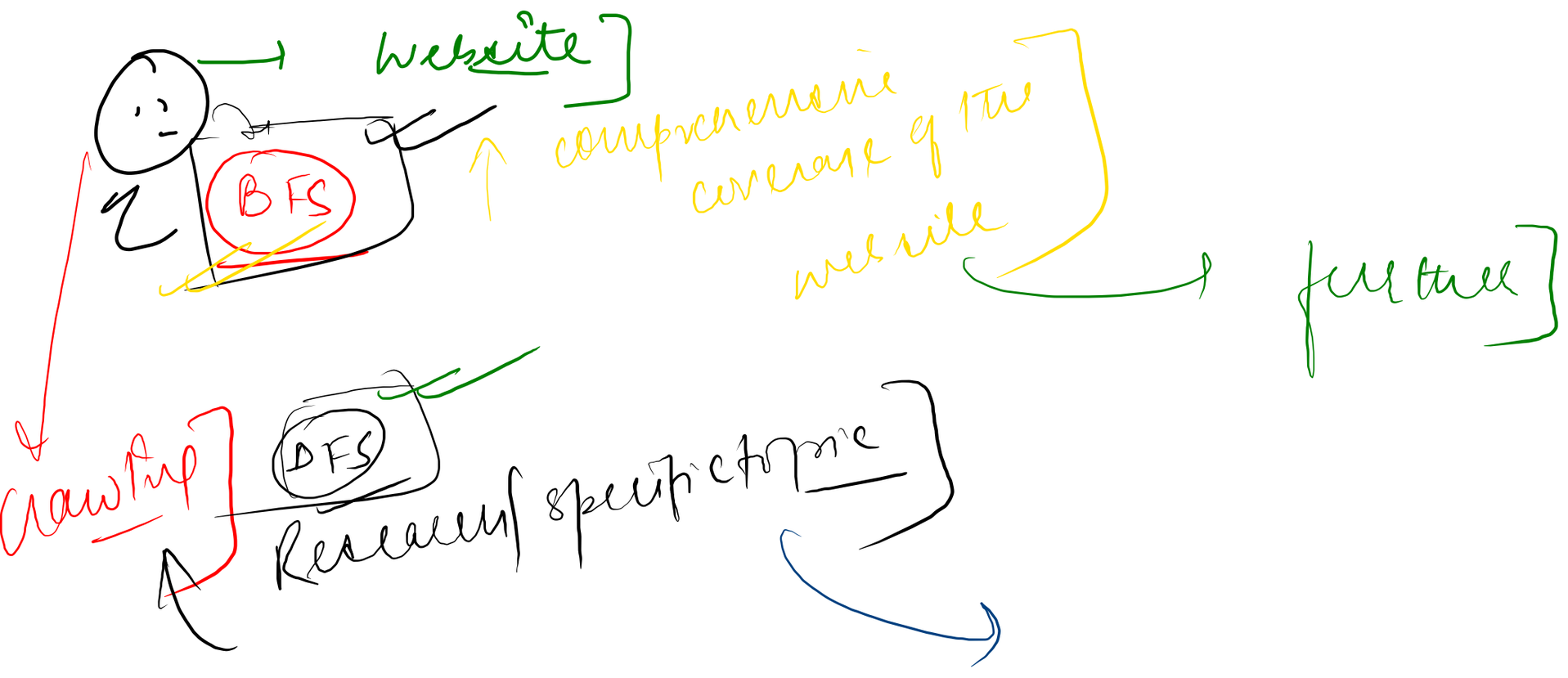

what/why
web crawling


BFS
Neighbours
shorted distance
algorithms
1
2
3
4
5
6
7
8
9
10
11
1, 2, 3, 4, 5, 6, 10, 11
7, 8, 9

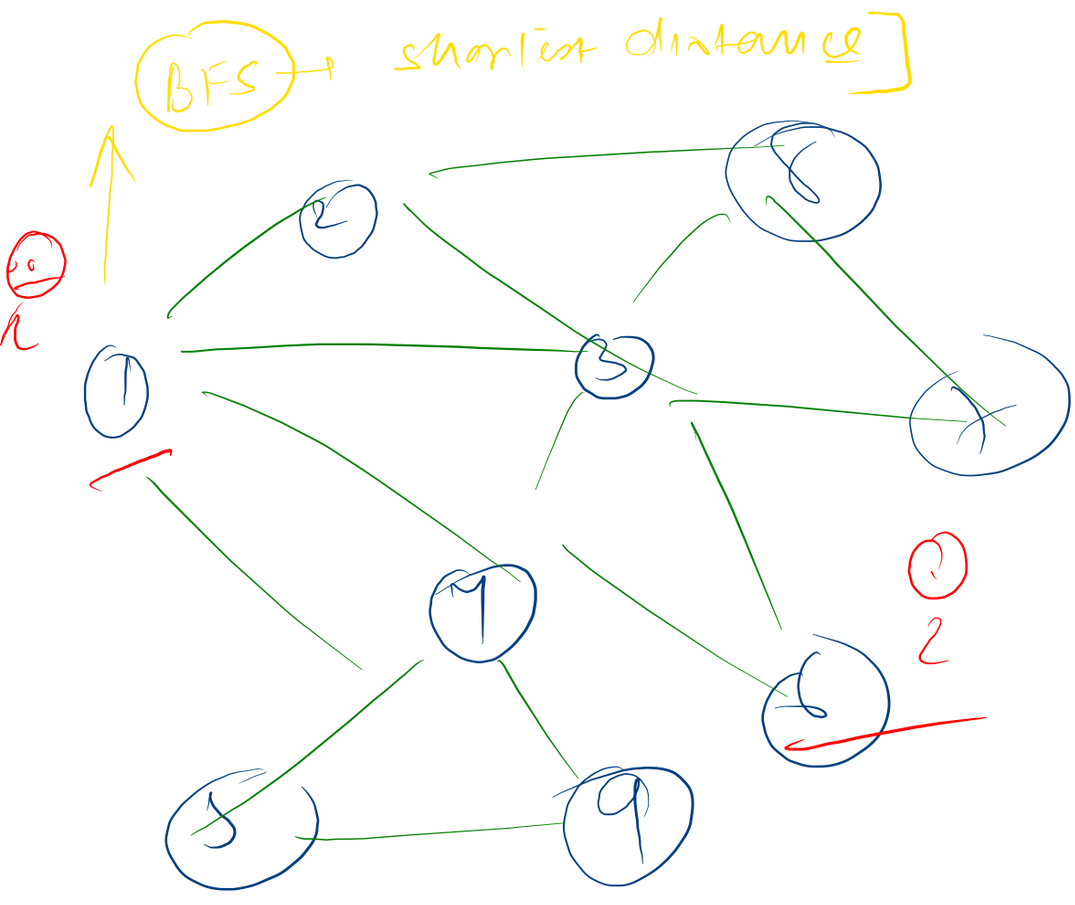


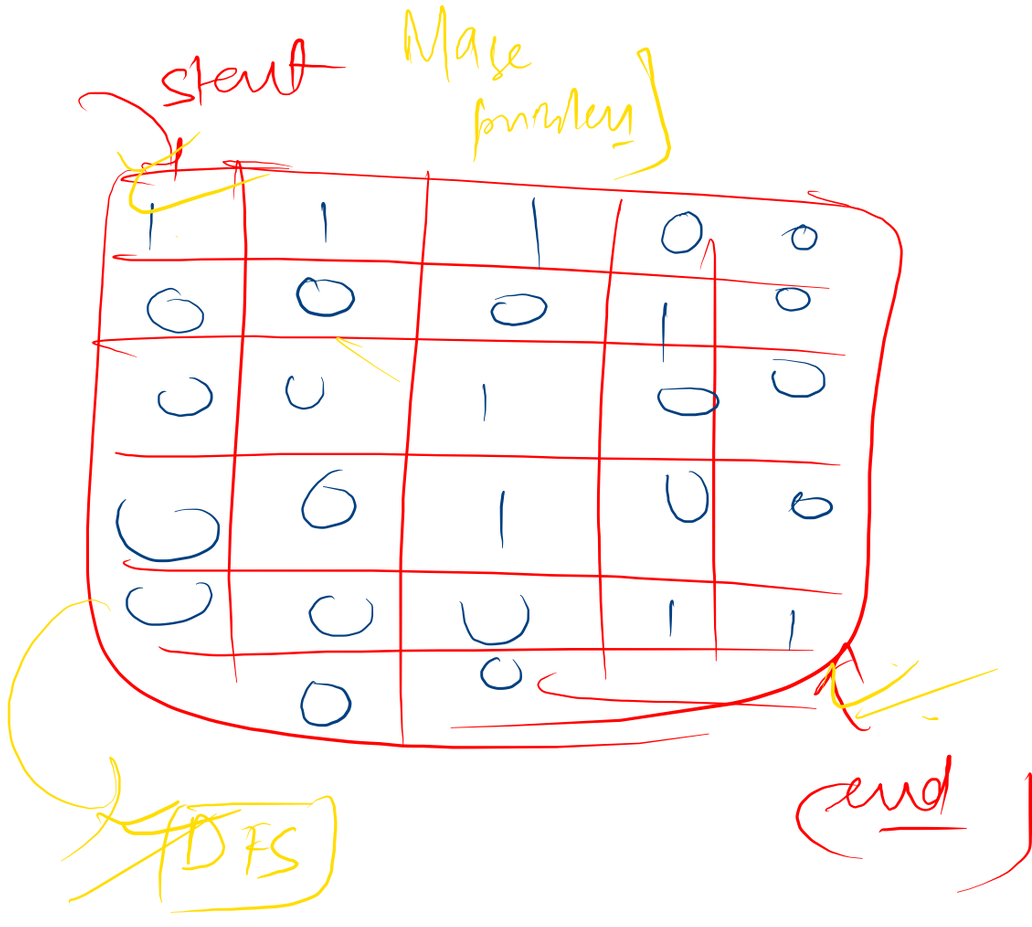
1, 2, 3, 4, 5, 6, 8, 9, 7, 10


BFS and DFS
BFS
DFS
Prefer? / why?
webcrawler
8:12 PM snap
10 min break

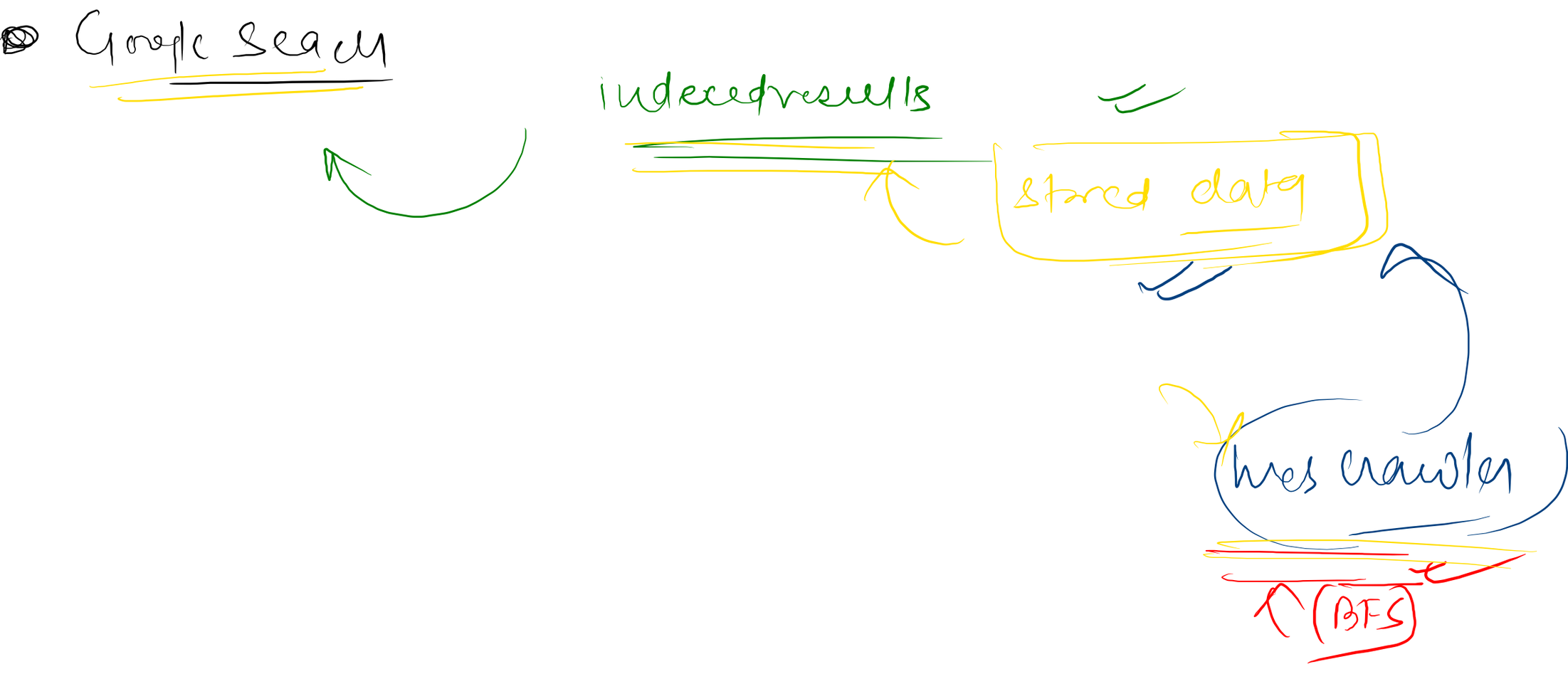

website
BFS
comprehensive coverage of the website
features
crawling
DFS
Research specific topic

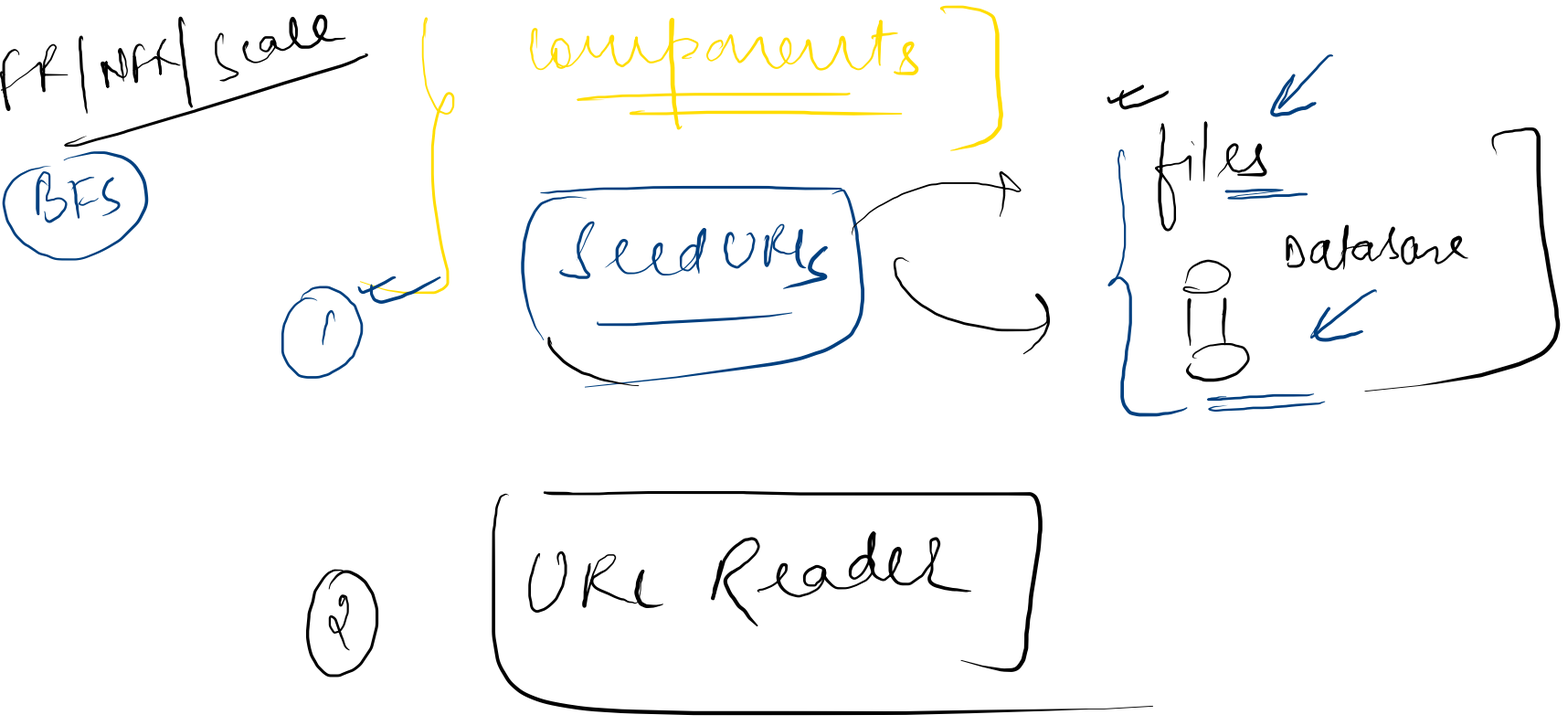
BFS — scanning/crawling all the websites


BFS
shortest distance
2
1
5
7
9
5
9
2

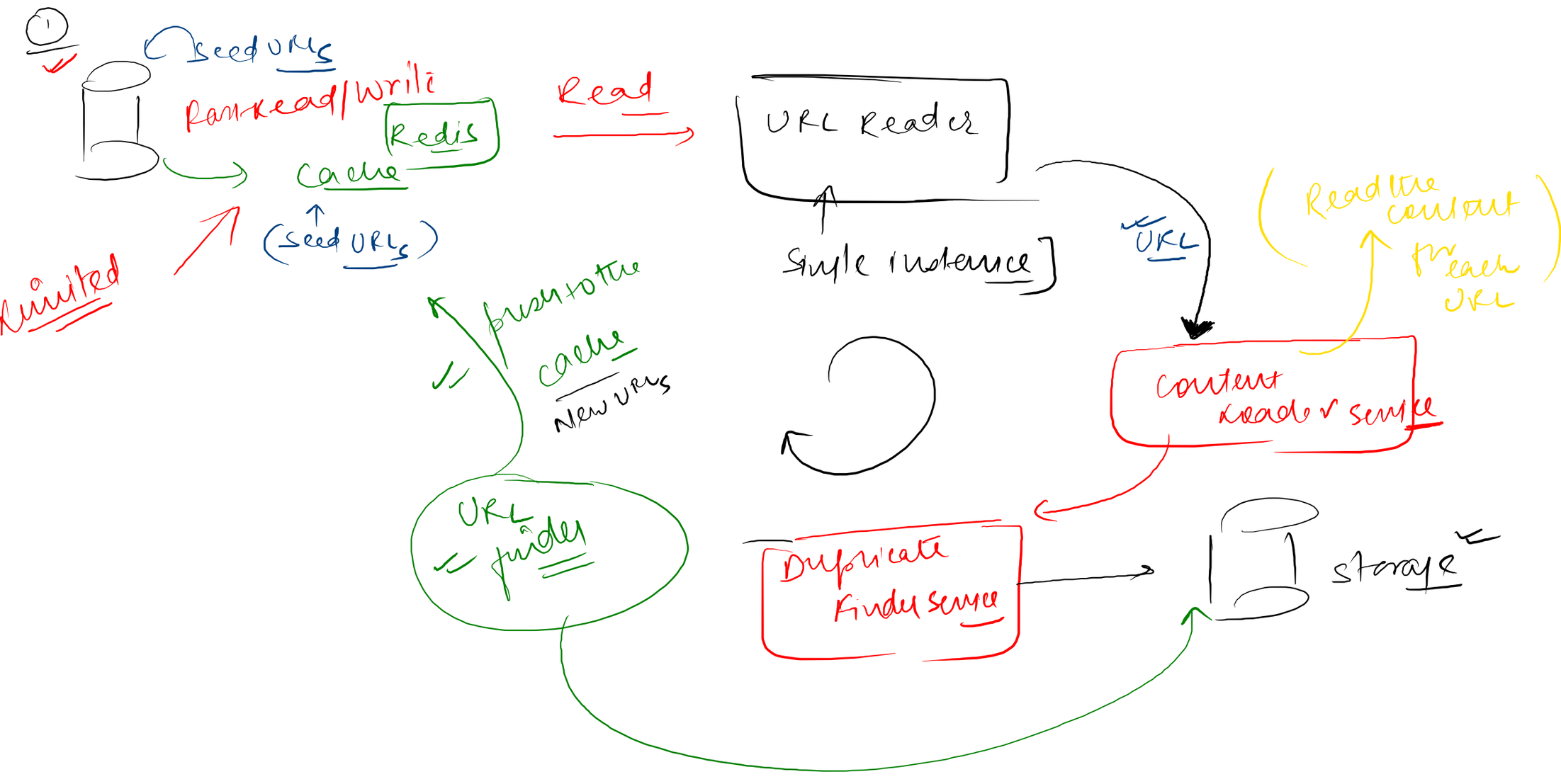

start
Maze problem
1 1 1 0 0
0 0 0 1 0
0 0 1 0 0
0 0 1 0 0
0 0 0 1 1
0
end
DFS

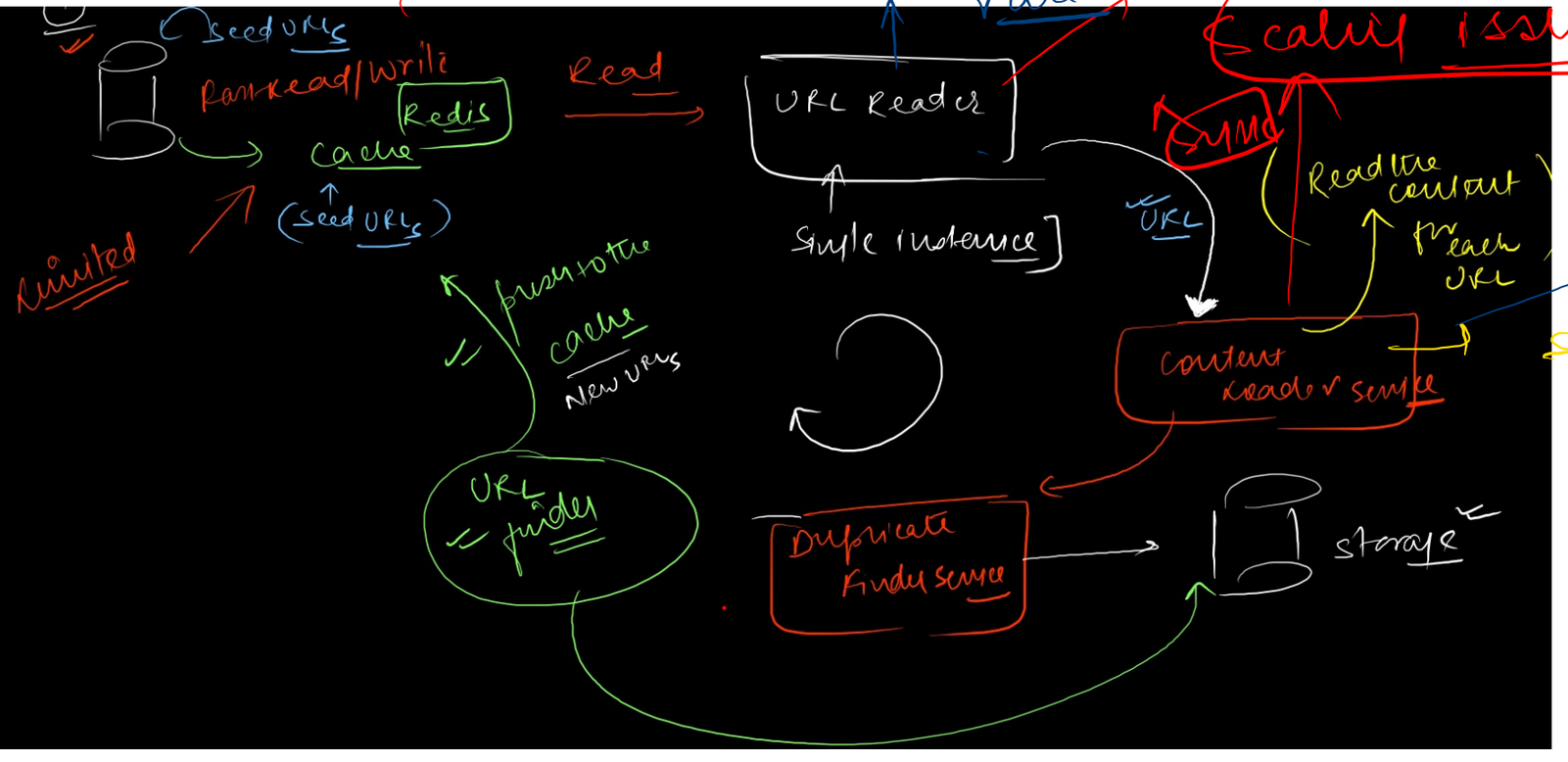
Google Search





2 URL Reader


1
Seed URLs
Ran-read/Writi
Redis
Cache
Read
URL Reader
Style instance
URL
Read the content for each URL
Content reader service
Seed URLs
Limited
push to the cache
New URLs
URL finder
Duplicate Finder service
storage

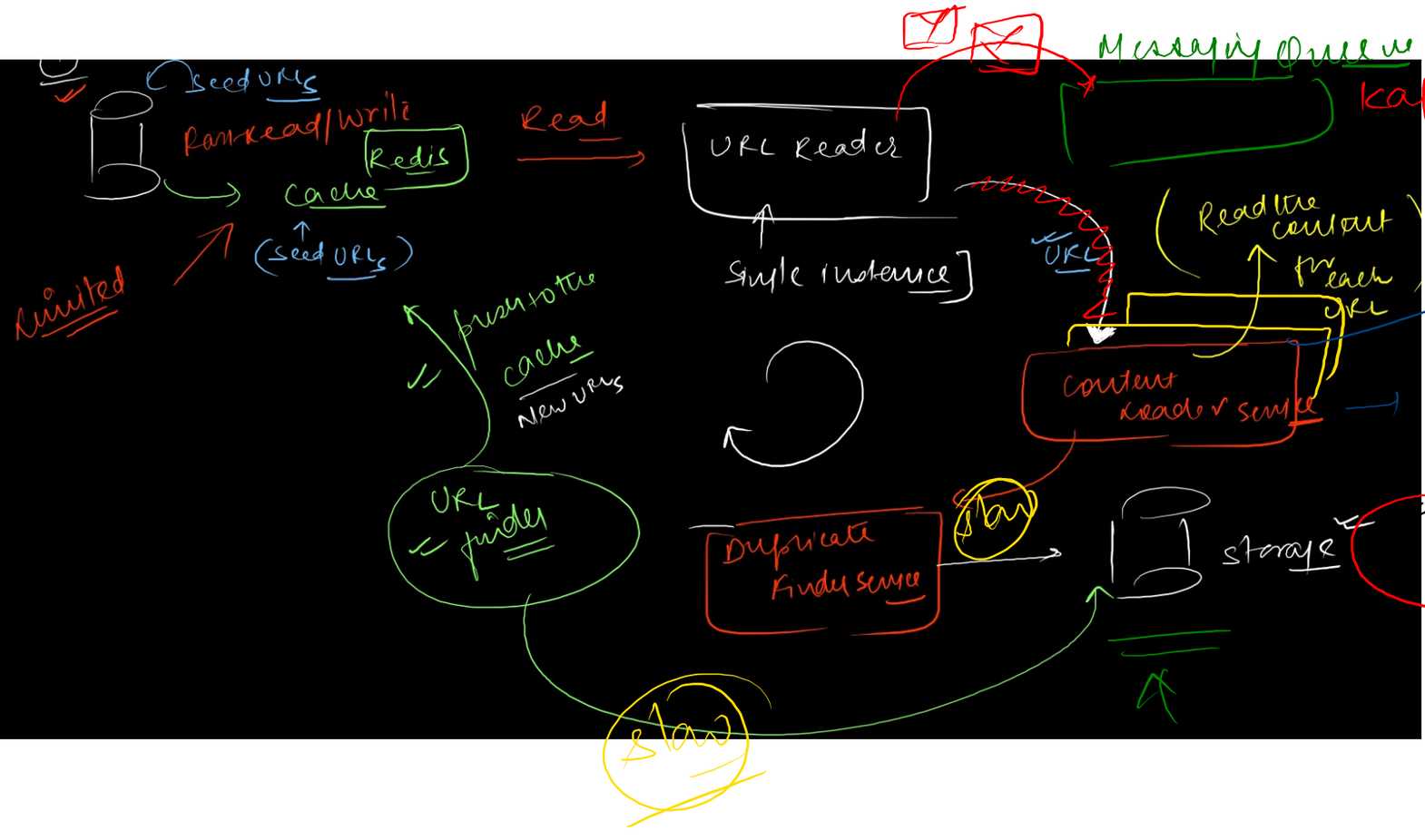


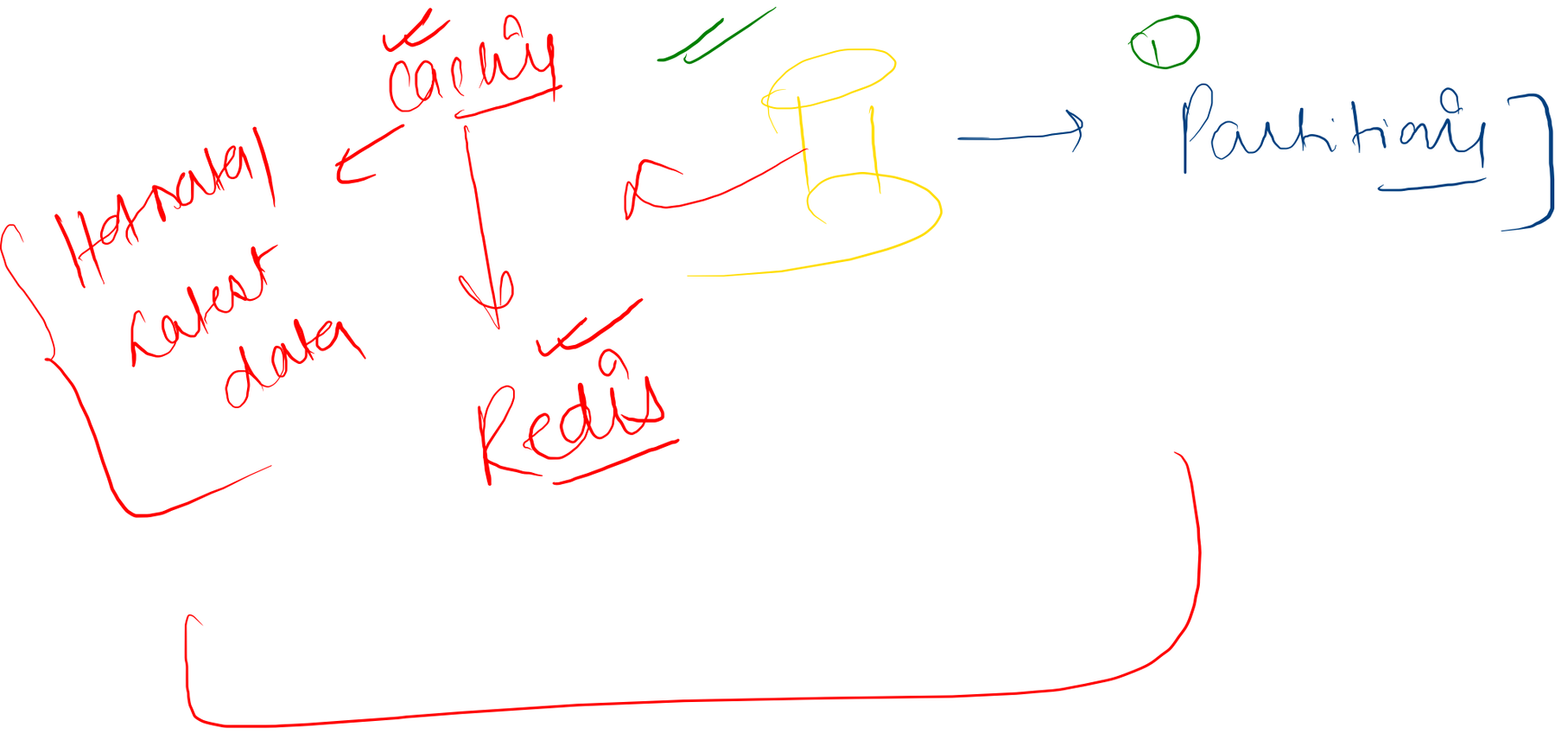
manufacture rate

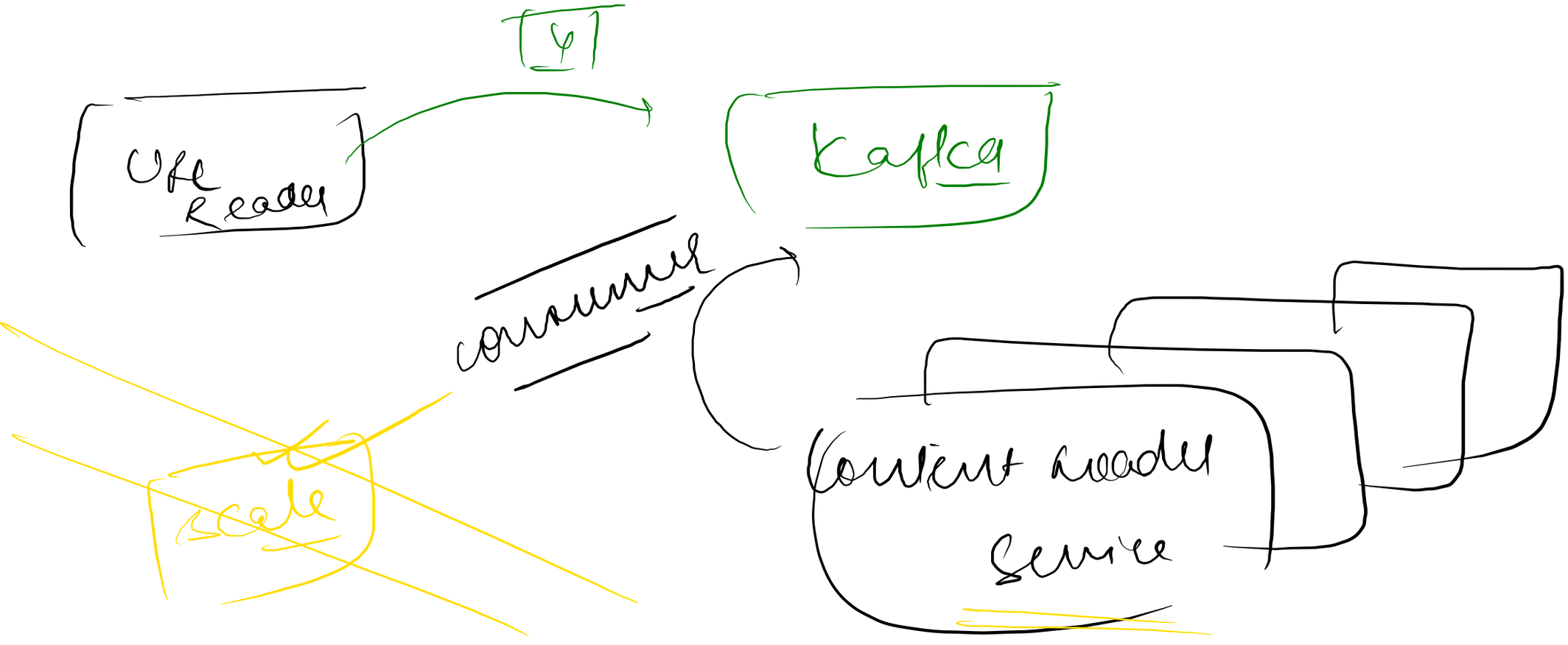
Scaling issue

slower rate

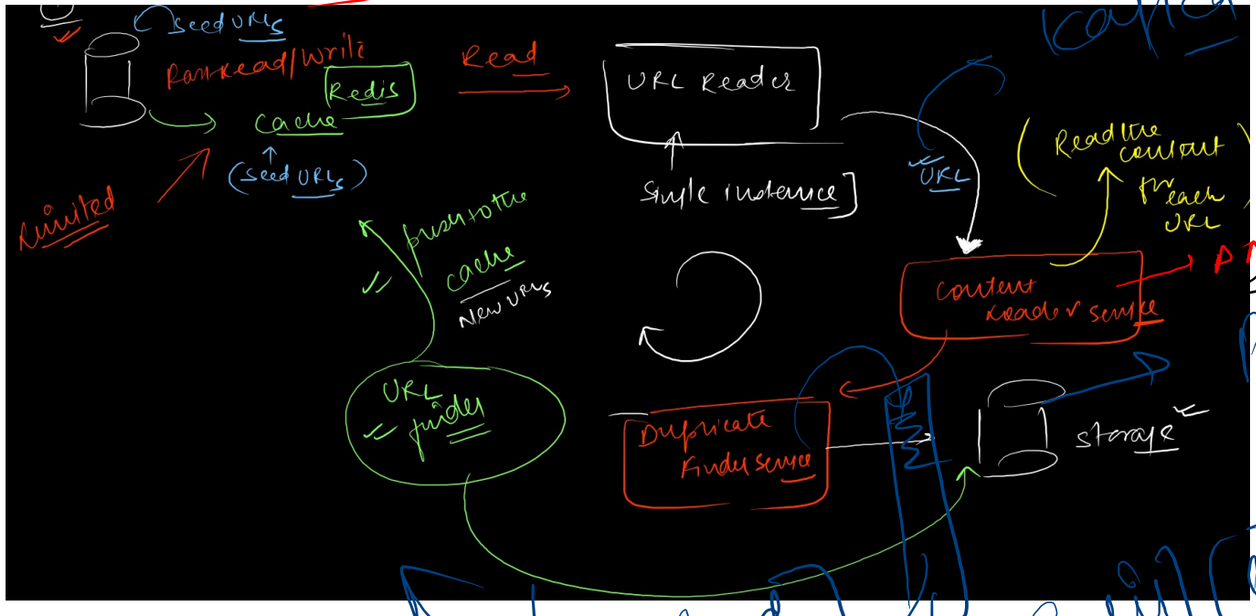
scale


Seed URLs
Ram-read/write
Redis
Cache
(Seed URLs)
Limited
Read
URL Reader
Single instance
Message Queue
kafka
consume
URL
Read the content
for each
URL
Content Reader service
Horizontal scaling
Read/write
I/O
slow
push to the
cache
New URLs
URL finder
Duplicate Finder service
slow
storage
6 PB
Distributed storage
HDFS
cloud storage
Data lake
Indexer
Write to any


Caching
Partitioning
Hot/cold/
Latest
data
Redis


CDC Reader
kafka
consume
scale
Content reader Service

Web Crawler



Kafka in S/W

Scope of improvement

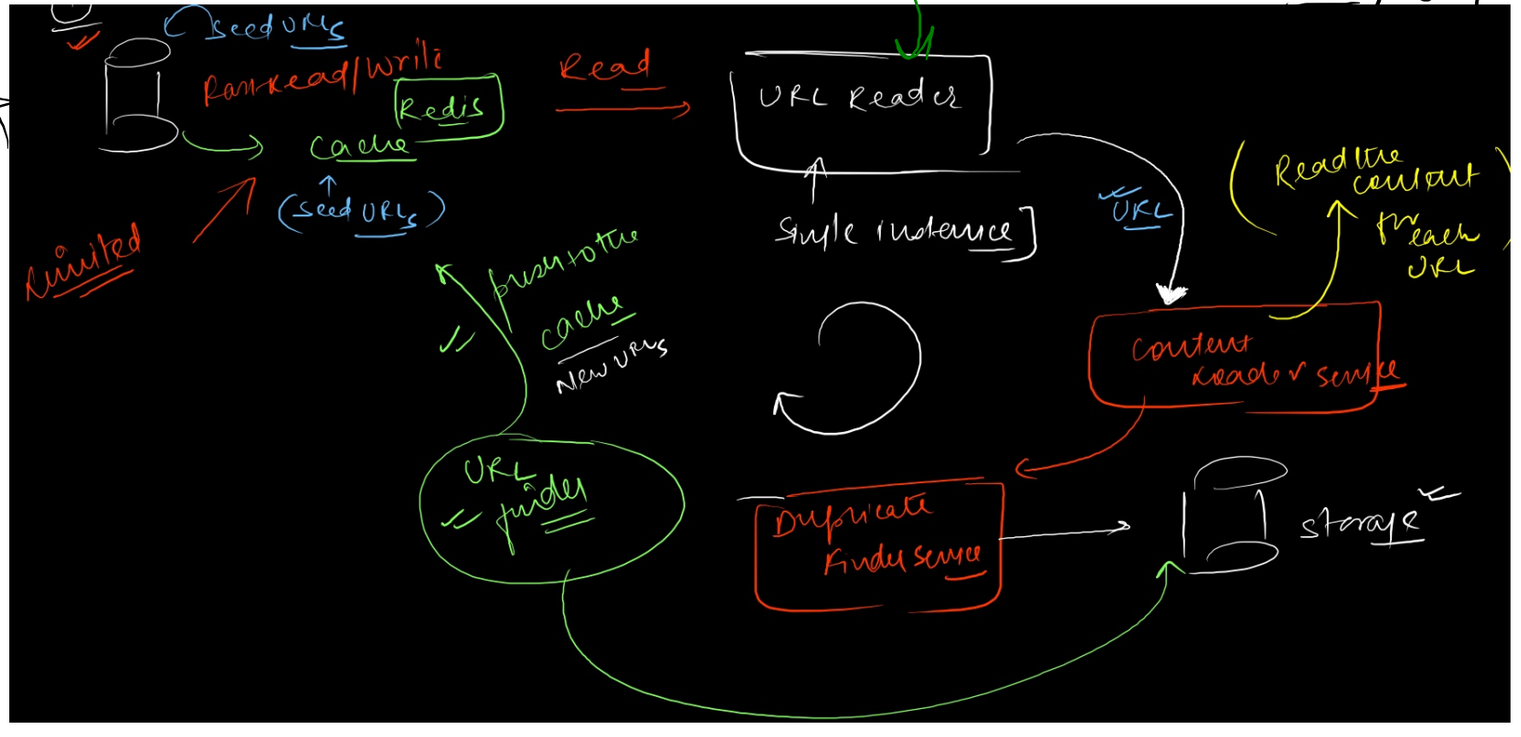
Cache → improve the speed of the system

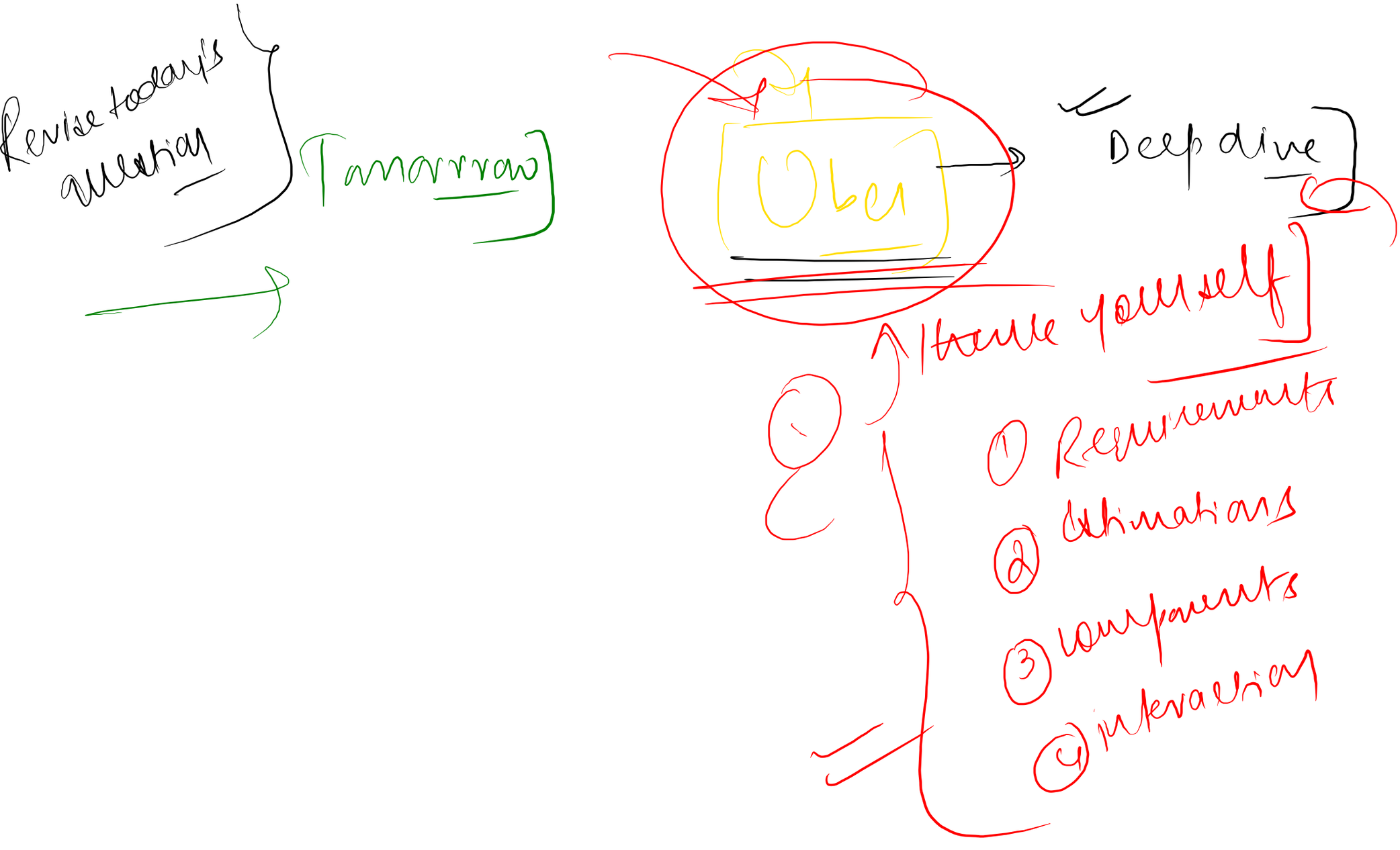
Partitioned

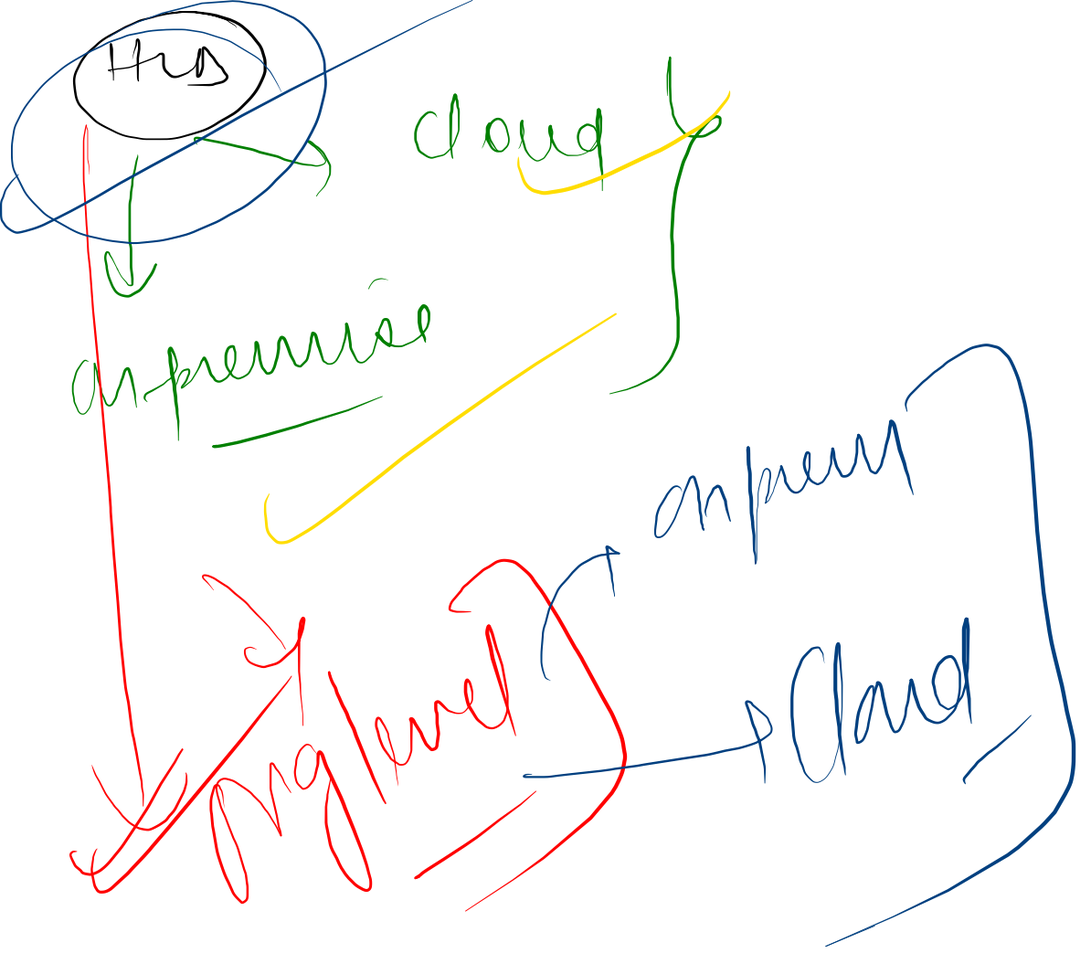
Ran Read

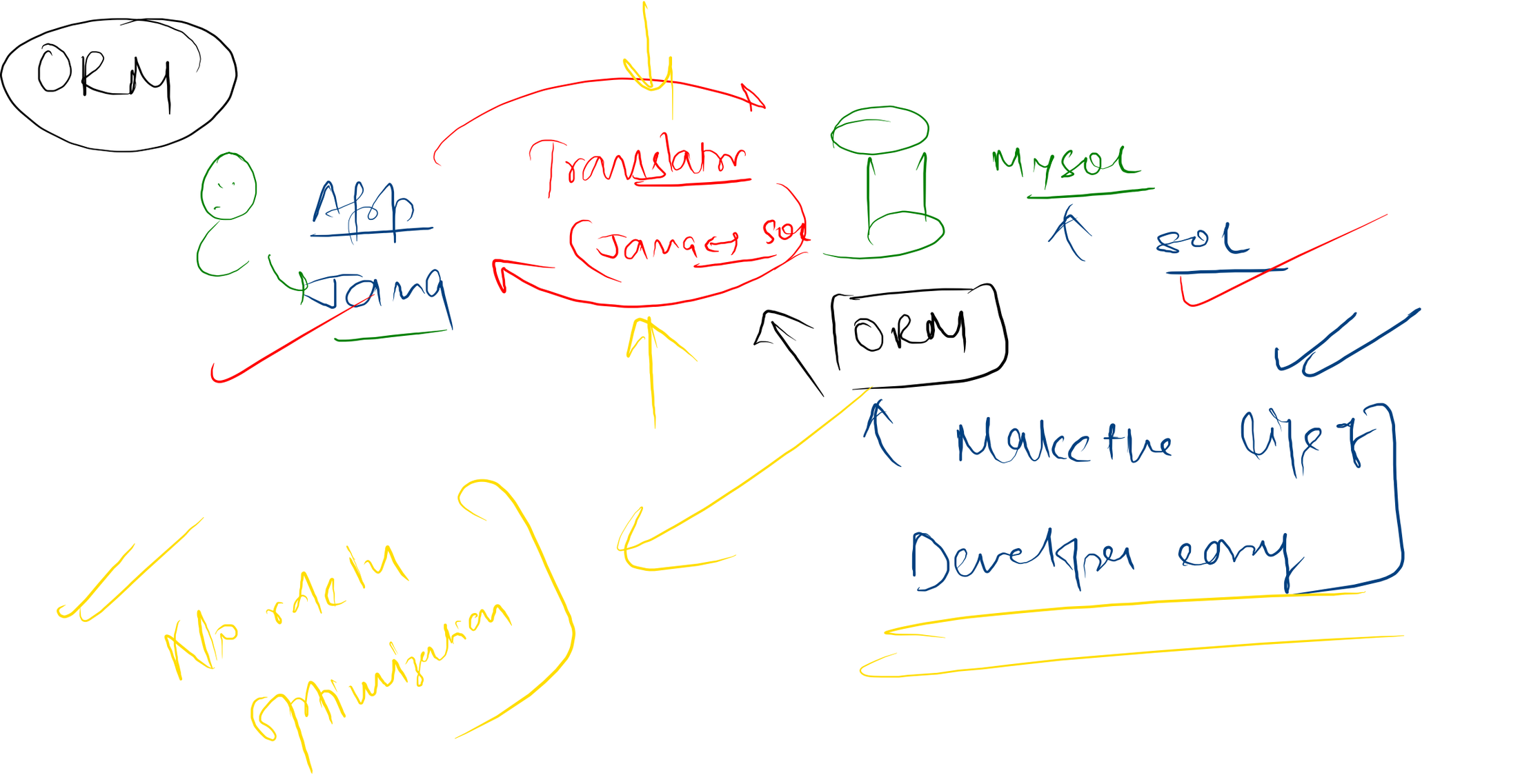
Redis Cache

LLD → language / objects / class diagrams




Revise today's
Tomorrow
Uber
Deep dive
Handle yourself
1. Requirements
2. Estimations
3. Components
4. Interaction

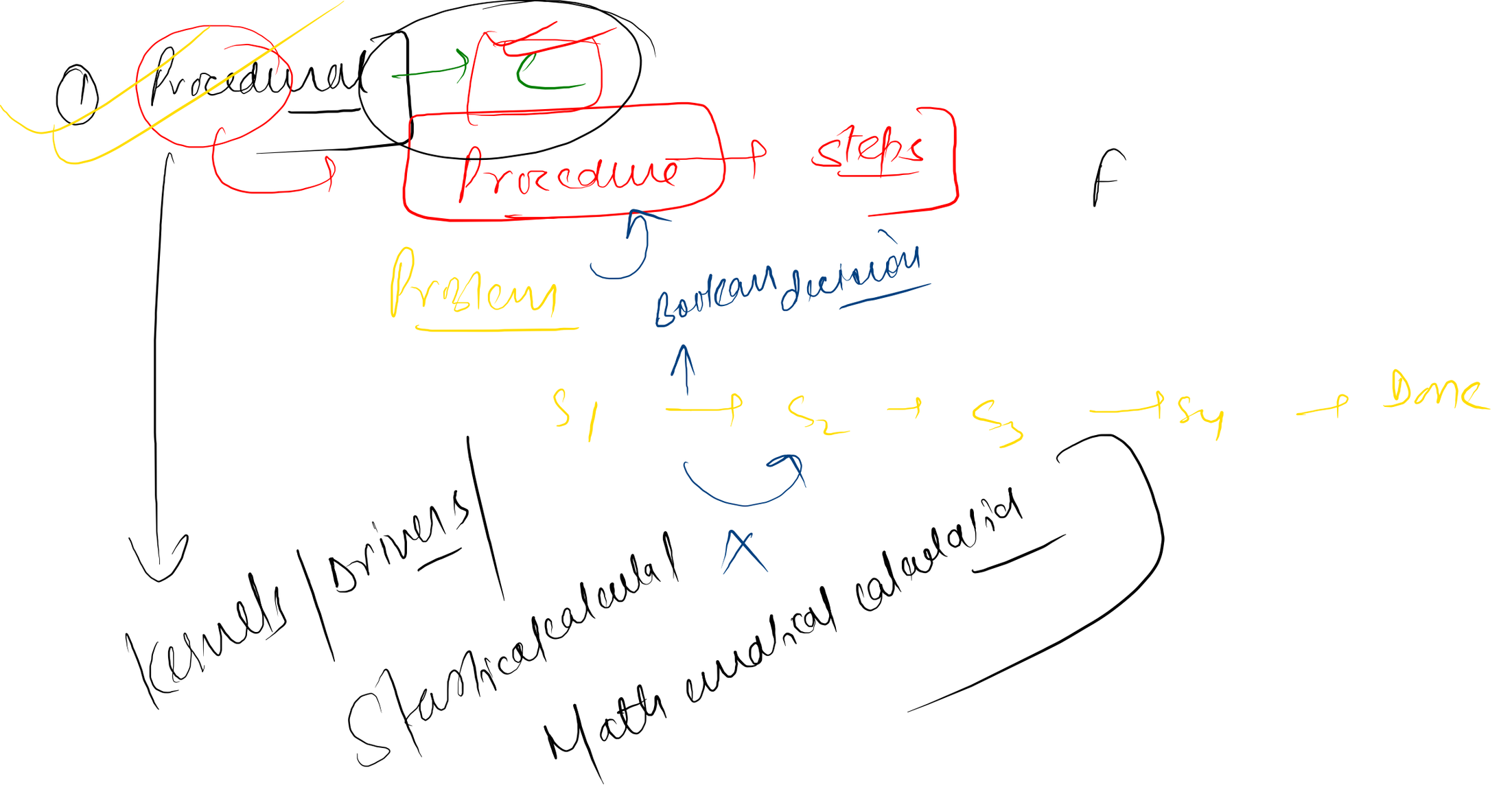

HYD
cloud
on premise
Hybrid
on prem
Cloud


ORM
App
Java
Translator
Java ↔ SQL
MySQL
SQL
ORM
Make the life of Developer easy
No need to write

C++ | Java | Python | →

Which to use when

2024 → Programming styles/Paradigms


1
Procedural
C
Procedure
steps
f
Problem
Boolean decision
S1
S2
S3
S4
Done
Kernels / Drivers /
Statistical calculation
Math numerical calculation


2
2000's
internet
real world
(App)

Real world

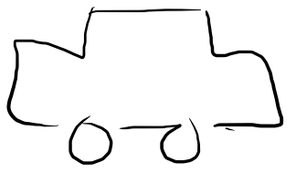
Programming

Represented

OOPs

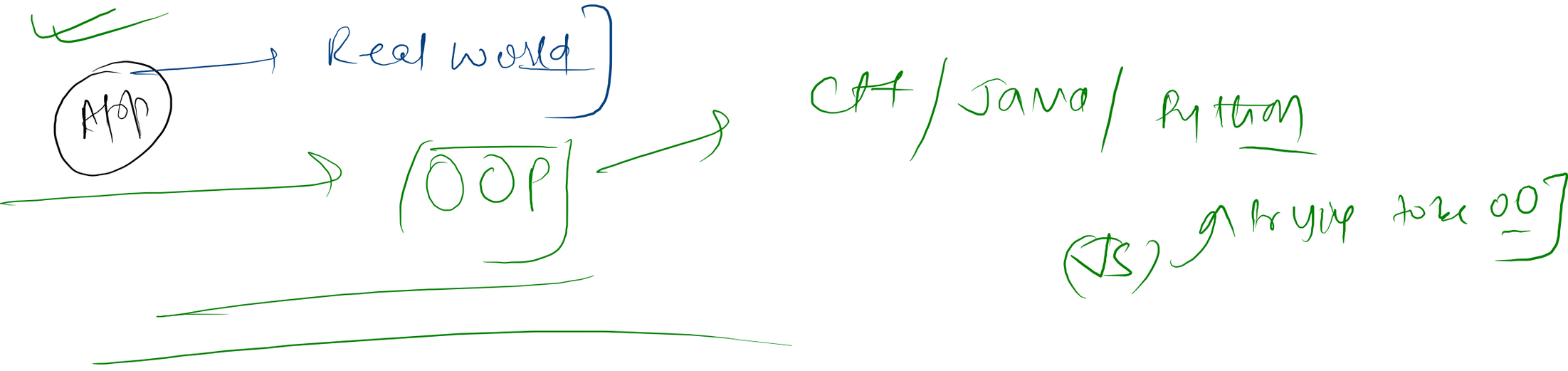
Java
C++

1



2

Relationships


App
Real world
OOP
C++ / Java / Python
(vs) any language not OO

③ Function PL → Everything is immutable

Write less, do more

Function is a first class citizen

(fn) → variable
→ argument
→ assigned
→ returned

→ like other literal

Workers → Get More] concurrent app

④ Declarative programming

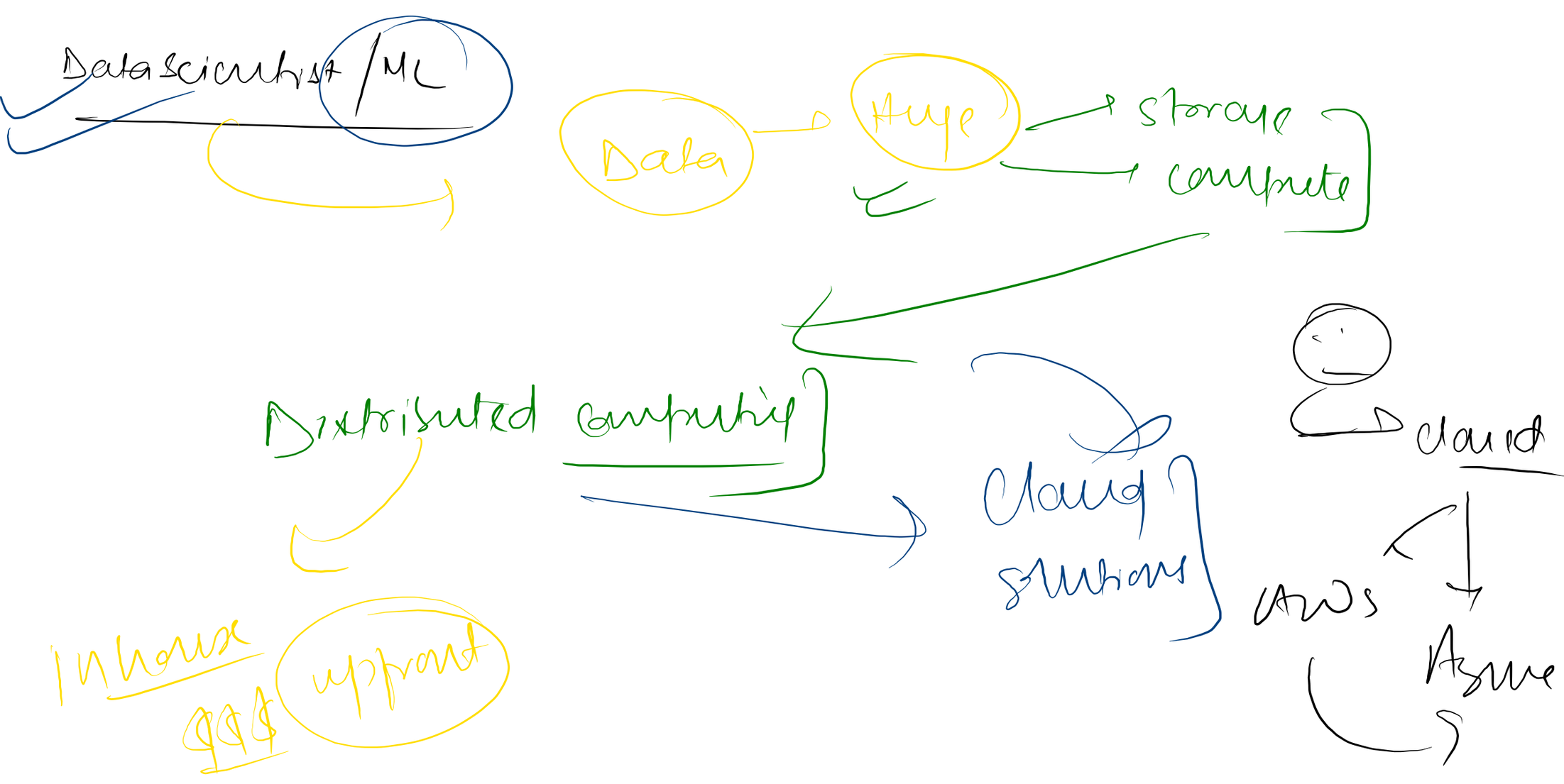
what and not how

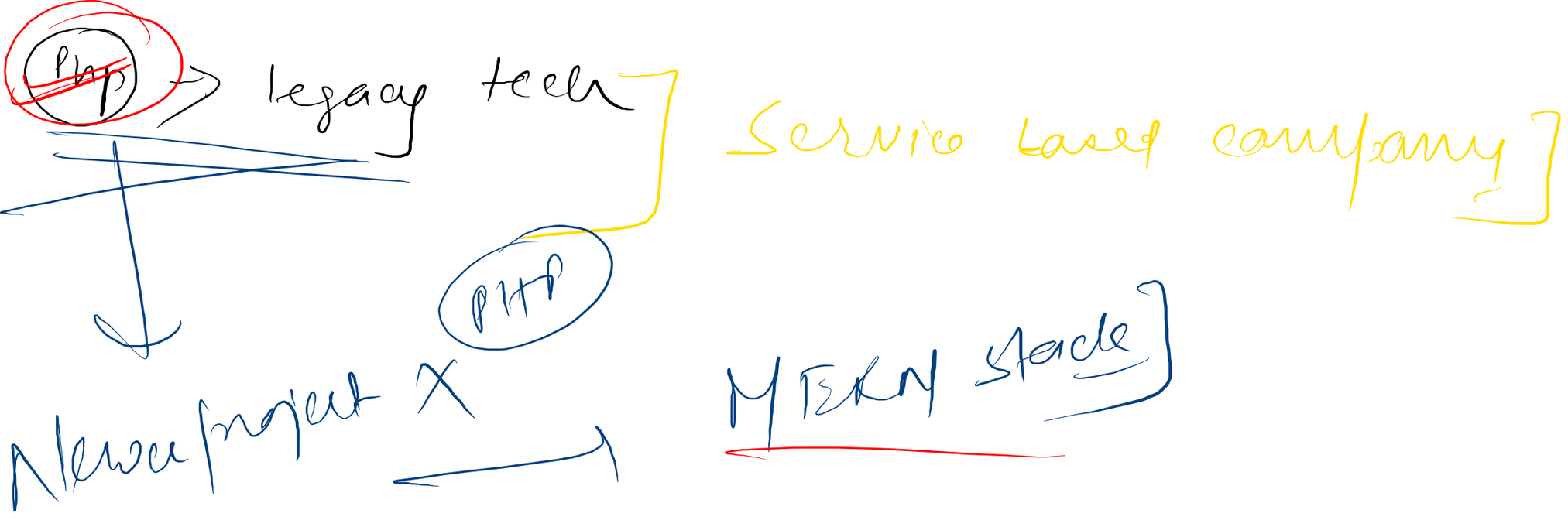
what

SQL/C++

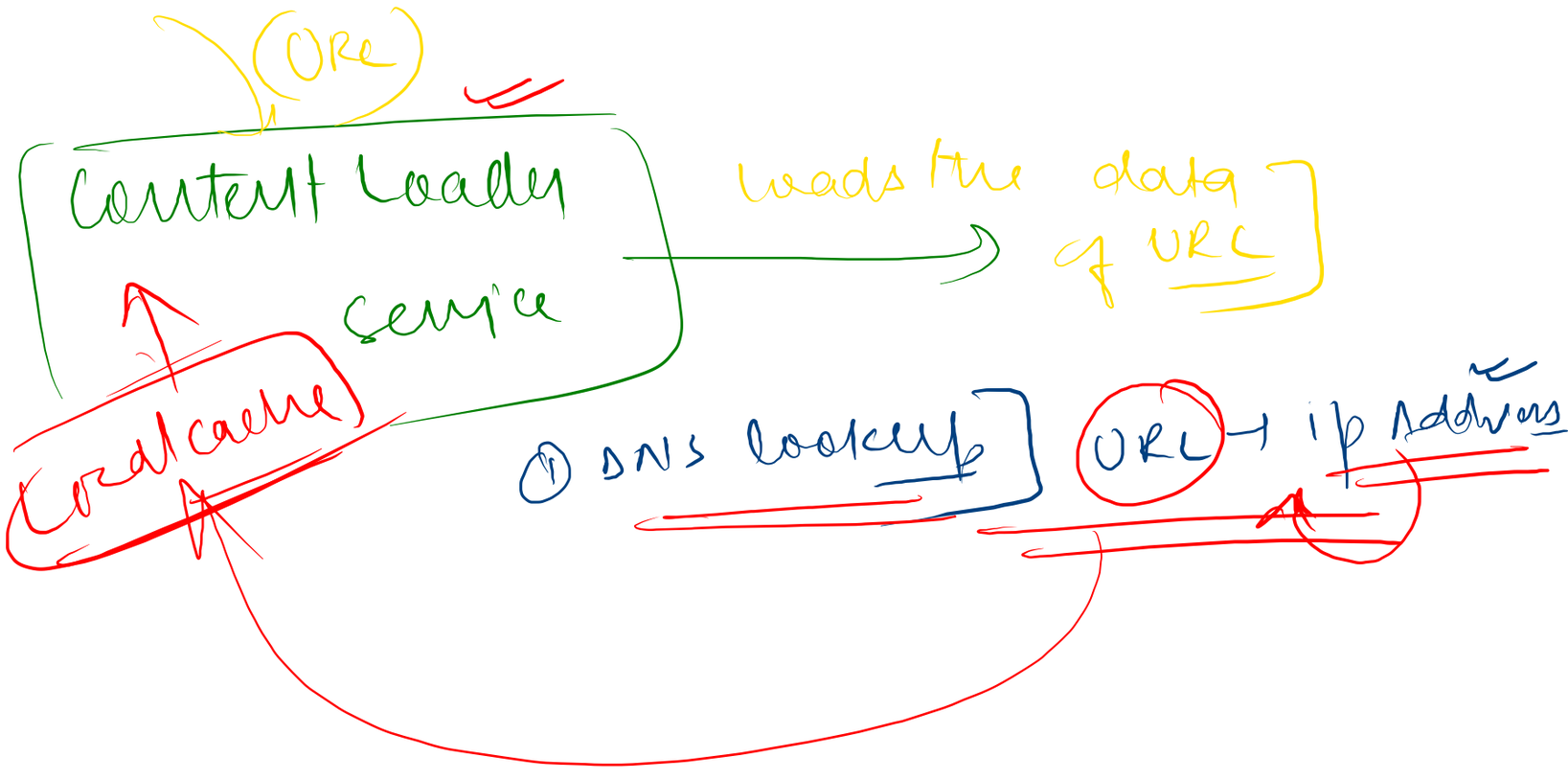
what and how (logic)

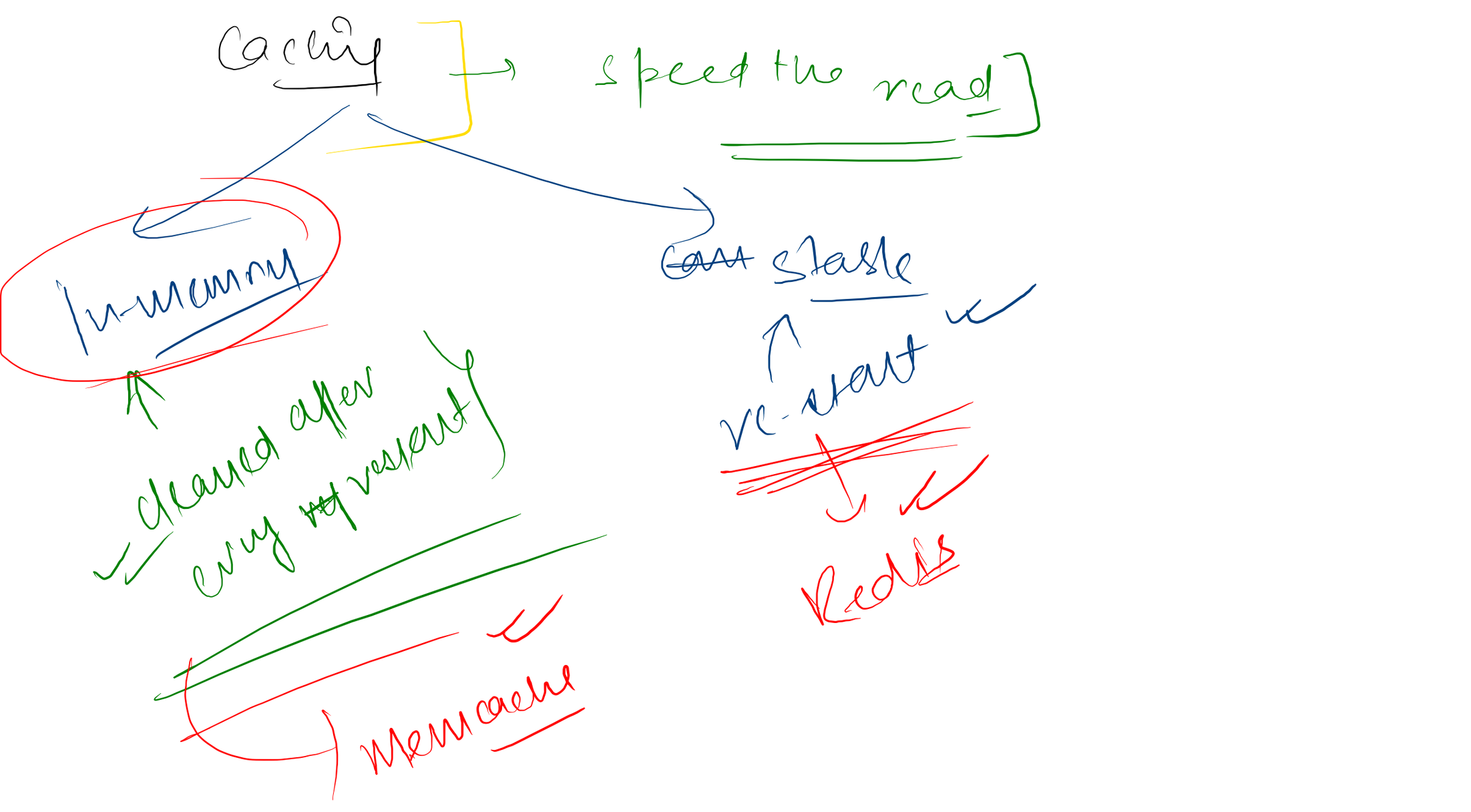
Select * from students;

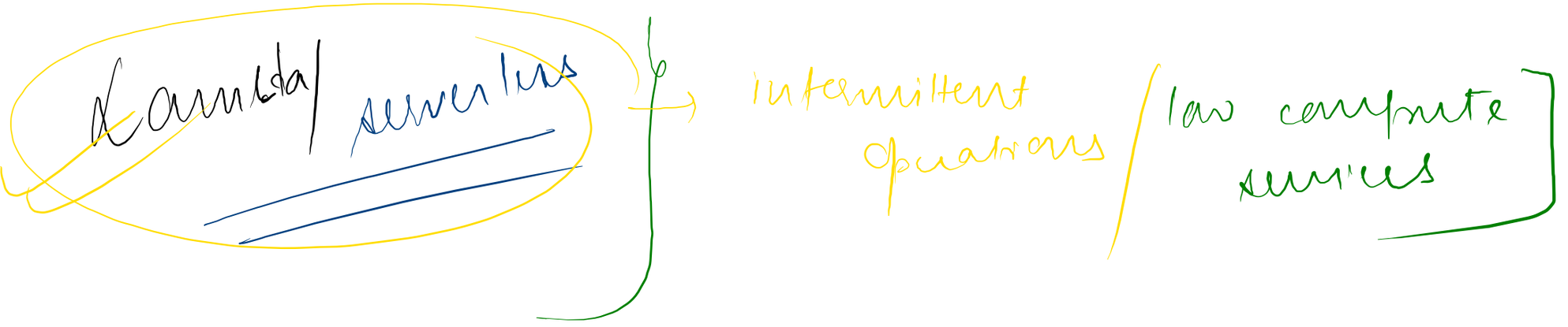

Data Scientist / ML
Data
Huge
storage
compute
Distributed computing
Cloud solutions
Cloud
AWS
Azure
In house
$$$
upfront


PHP
legacy tech
Service based company
PHP
New project X
MERN stack

Biggest players




Caching
speed the read
In-memory
cleared after every restart
Memcache
Stable
re-start
Redis


Lambda/ serverless
intermittent operations
low compute services