SYSTEM DESIGN

~course by Rohit Negi [Coder Army]

L1 | 12-May-2025

Pre-requisite → only C++ required

What is LLD :-

Search element in Array

- if sorted → Binary search
- if unsorted → linear search

⇒ we have many **Isolated Problems**

ex :-

searching

sol^n :-
- Linear search
- Binary search

sorting

sol^n :-
- Quick sort
- Merge sort
- insertion sort

↓

in isolated problems k around when we design / develop any application

↓

this itself is called LLD.

⇒ So, when we build any application around these solutions for these isolated problems → it is called as LLD.

⇒ till now we haven't grasped the complete concept of LLD

↓

so, lets understand it by a story.

There are two friends

Anurag	Maurya
- knows DSA ✓	- knows DSA ✓
- donot know LLD ☹	LLD ✓

↓

just after their Graduation they get placed into Quickride - a taxi aggregator

↓

so, lets see how these both friends deal the same problem.

Anurag ↴

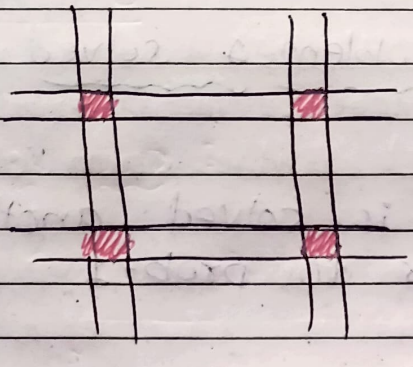
mgr comes and asks to make taxi-aggregator app and tells to start working.

so, he starts working. Thinks of problems in ride-booking.

Problem 1 :- user needs to src → desn

because he is DSA oriented, he directly jumps into solving the problem when he was asked to develop an application.

so, now he thinks that :-

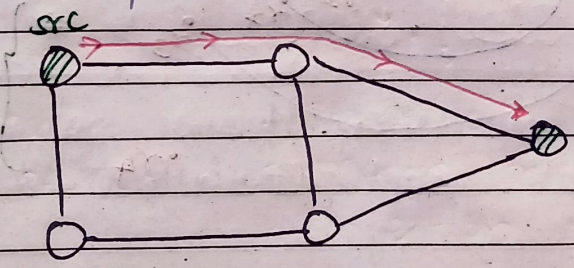


intersections of roads in city can be denoted as "Nodes"

and all the roads in between, lets consider them as "Edges".

this way I can map the entire city in a Graph ☺

So, lets say :-



this graph forms when entire city is mapped

(obviously, any real city will have humongous graph ☺, this is just for illustration ✓)

So, now he will implement Dijkstra's Algorithm

↓

find shortest path from src → dest

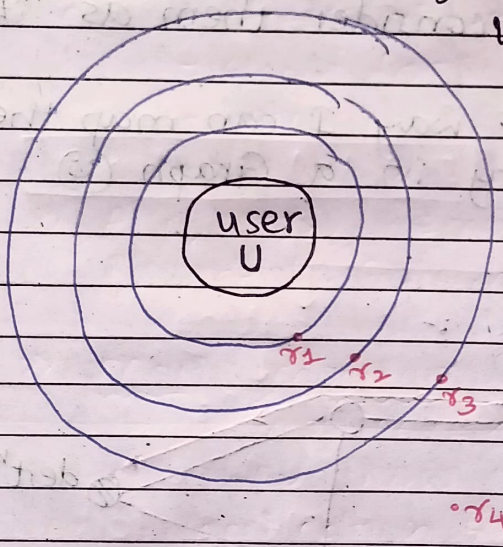
↓

problem 1 solved ✓ ☺

As Problem-1 is solved, another problem arises prior to prob-1.

Problem-2 →

Assigning rider to the user.



↓

lets denote surrounding of user by concentric-circles

these are riders (denoted by "r_i") in close-proximity of any user "U".

So, what if I map a Priority Queue corresponding to every user.

Priority queue (PQ) → implements "Heap Algo."

↓

which can be

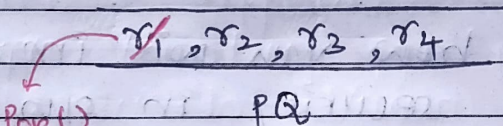
↓ minHeap

↓ maxHeap

If PQ is minHeap → smallest element at top

is maxHeap → largest element will be at top.

So, when minHeap is assigned to any user



if first-element is popped in minHeap, nearest rider to that particular user will be obtained ↓ and then we can assign it to the user.

∴ Problem-2 is also solved ✓ ☺

⇒ Now, Anurag goes to manager and pitches his solutions

↓

Manager says → Alright Anurag, these solutions are just algorithms. But where is the application ?

PTO

⇒ Basically, what mgr. wants to say is that →

» What are Objects / Entities in your application?

» What will be the relationship, how those entities will interact with each other?

» Also, how you will implement data security in our application?

» If we want to notify user about
- their rider,
- how far is rider,
- etc.

» Also, how we will integrate Payment Gateway in our application.

» Most Importantly ⇒ when our application reaches millions of user, how we will scale our application?

⇒ So, Anurag didn't discussed aforementioned nuances. He directly jumped into finding the algorithms to solve problems.

⇒ Now, let's see how Maurya approches application development.

Maurya's POV :

as maurya is well-versed with DSA as-well-as LLD

↳ He thinks : very firstly I need to design the structure of application, which is LLD.

DSA part will come very later

⇒ So, he thinks :

↳ » what will be objects / Entities in my application ?

- User
- Rider
- Location
- Notification
- Payment

} identifies these as his "objects" for app.

↳ » Now, he identifies relationship b/w the objects; that how each object will interact with each other.

» further he thinks of Data Security like sensitive data } ex:- phone number.

↓

ex:- when ride is completed, rider and user must not need each others ph. no. further so, it must be hidden from each other

such things come under Data security.

>> Also, he thinks how the application will scale as millions of user start on-boarding the application

↓

how will app run smoothly without experiencing any downtime!

↓

how will I write code so as my app sustains invariably high number of users concurrently.

⇒ So, when he designs the entire structure of app → further he tries to solve problems ↓ implementing optimum Algorithms.

(how Anurag approached ☺)

⇒ Until now, I guess we have got a crisp image what LLD actually is ☺.

So, firstly we design entire Architecture of the App → then only → implement algorithms to solve problems

{ LLD } { DSA }

So, LLD primarily focuses on 3 main things:-

- Scalability
- Maintainability
- Re-usability

a) Scalability :- when millions of users start onboarding concurrentey, how will app scale or sustain?

↓

how much effort will go into integrating new feature in app?

↓

how fast we can scale-up the app?

↓

All these factors are under scalability.

---

b) Maintainability :-

like how easy is to maintain the code-base of app?

↳ if I integrate new feature, is it like that, previous features start to throw bugs, any other object starts to behave strangely?

↓

these things come under maintainability.

- Code-base must be such that, maintaining and handling it must be very easy.
- Also, code must be easily debugged. We must be able to find bugs easily. or least no. of bugs should arise.

o) Re-usability : code-base must be highly-highly reusable

ex: how to integrate payments / notification?

↓

these are not any app dependent, pay/notific" are same for every other app.

↓

so, such codes must be very easy to integrate or use. working on "plug-and-play" model.

↓

means we can easily plug it into any "applic" and it must run smoothly without affecting other entities

↓

also it means code-base must not be tightly-coupled

⇒ like more or less we are assigning some person to any particular order. Ex:-

Amazon delivery agent to order

Zomato/swiggy delivery partner to restaurant & user

↓

can be generically called Rider-mapping Algo.

this rider-mapping algo must be generic and not for any particular app usecase.

↓

this should be re-usable across the apps like - amazon
- zomato / swiggy
- zepto / instamart / minutes
- ola / uber / rapido

↓

as per use-case, slight change and ready to plug-and-play.

↓

this way code must be re-usable.

Now,

Q. What is not LLD ?

vimp question, bcoz many students confuse b/w LLD vs. HLD.

So, HLD → High level design

↓

focuses on other things than that of LLD

it (HLD) focuses mainly on :-

» what will be Tech-stack ? (ex: Java / spring)

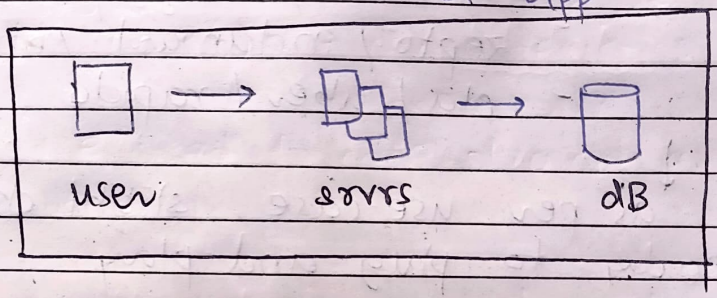
» which dB to use ? (SQL, NOSQL, etc.)

» How servers will scale

» Cost/Optimisation of srvrs.
(like as per requirement cost must ↑ or ↓)

So, in HLD interview :-

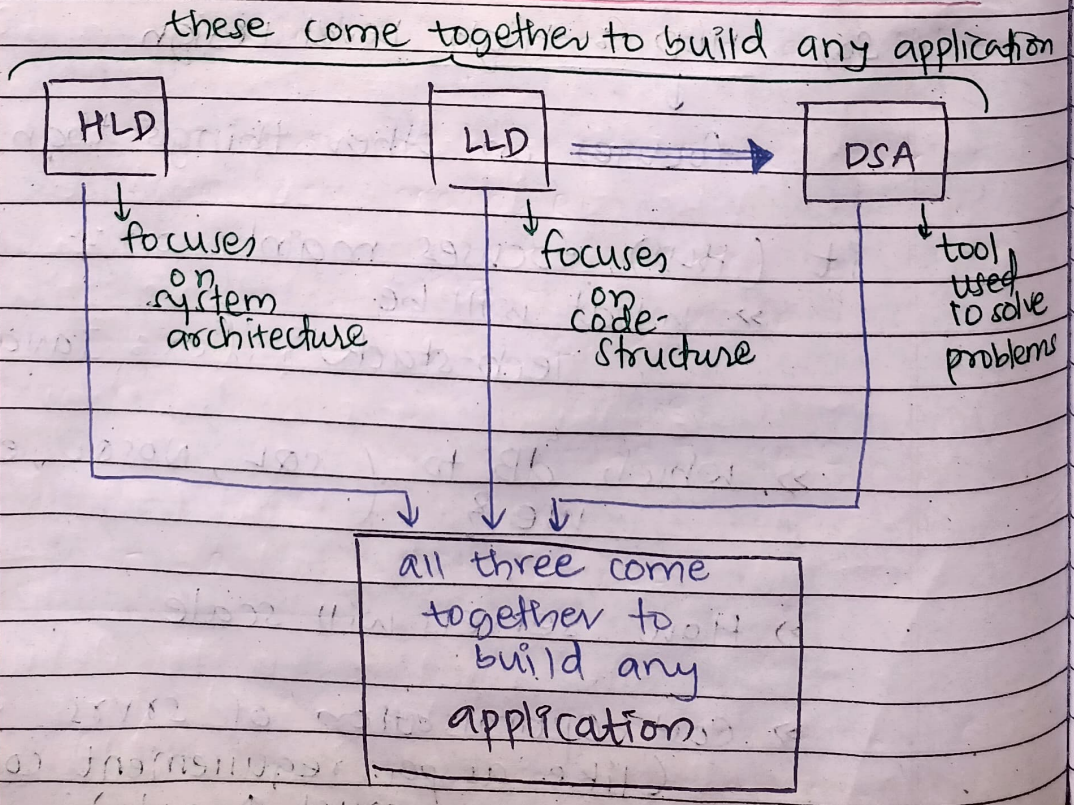
- we write close to zero code (negligible).
- we draft entire workflow, architecture of app.



in LLD interview :-

we need to write code and make class-diagrams (uml dia.) and their relationships.

So, this is difference b/w HLD vs. LLD. ✓ ☺



So, finally to summarise :-
if you remember following line, all concepts will be crystal clear →

"If DSA is brain of application, then,
LLD is its skeleton"