DATA STRUCTURES & ALGORITHMS

(I) SYSTEMS EXPERT

Design fundamentals → foundational knowledge reqd for systems design interviews

Data structures → foundational knowledge reqd for coding interviews

eg of design fundamentals: SQL, server, cache, polling, load balancer, HTTP, database, hashing, replica", client, proxies, Nginx, availability, leader elec", P2P

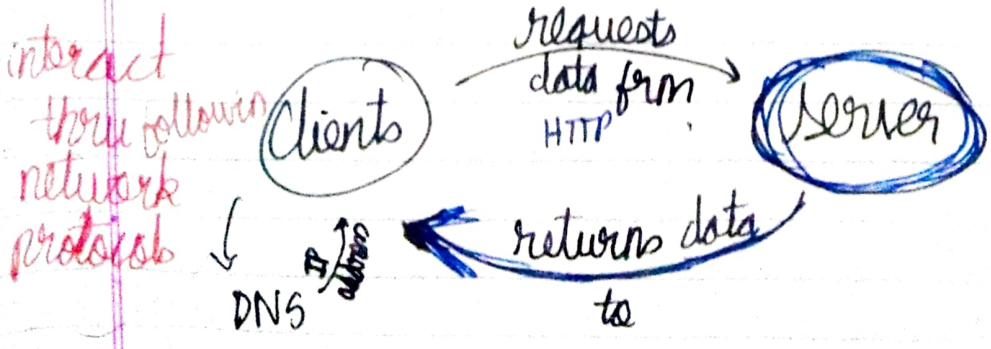
(IA) Client-Server Model/Architecture/Paradigm

Foundation of modern internet, helps us understand how computers speak to one another.

What we do? → Type URL in browser and hit 'Enter'

What happens?

i) Client: is something, a machine that speaks to the server.
ii) Server: something else, a machine that listens to the client, listens for clients to speak and then speaks back to the clients



speech is done by sending of data, demand (demand a type of data request for data back)

truth table output is exactly the same

NOTE: $\sim q \rightarrow \sim p$ is equivalent (one & the same) as p whereas, inverse & converse don't meet this condi"

- 2 propositions are equivalent if the truth values are same for all const. variables in the table

(iii) Browser : Client
Website : Server
(ones which have server)

⟹ NOTE: A Website (same) may have diff. servers in diff. loca"s. eg:- Netflix India has a diff. server from Netflix USA.

(iv) Browser doesn't really know what the server is. All that it knows is that it can communicate with server. Doesn't really know what the server represents. It just requests info from it and does stuff based on info recd. from server.

(v) To start ~~up~~ communica" with a server, a browser 1st sends a DNS (Domain Name Server) query to find out IP address of server.

DNS query → special request going to a ~~set of~~ predetermined set of servers asking for IP address of a server

IP address → unique identifier for a machine. All comps. connected to internet have ways to find out these IP addresses or discover routes to those addresses
They can send packets of data/info in form of bytes to IP address.

Analogy :- IP address is like a mailbox that some entity has granted to a machine
eg:- Algo Expert's IP address has been granted to it by ~~Google~~ its cloud provider - Google Cloud Platform. It reserved an IP address for Algo Expert.

eg:- Thus, browser makes DNS query for AlgoExpert.i. Receives back an IP address and thus starts communica with the server.

(vi) HTTP: way to send info that comp. can understand.

exercise: type "dig algoexpert.io" → does a DNS query and returns IP address of algoexpert.io

→ when browser sends HTTP req. to ~~a~~ ge server it basically it sends a bunch of bytes / char that are gonna get packed into some special format and sent to servers. This request also contains IP address of your PC (a-k-a source address)
When server receives source address, it knows that ~~to~~ on which IP address it needs to send a response to.

(vii) Ports - Servers usually listen for requests on specific port. Any machine having a distinct IP address has 16000 ports that progs. on the machine can ~~be~~ listen to
On communicating with machine, you (client) gotta specify what port you wanna communicate on

Analogy → IP address: Mailbox to an apartment complex
Ports: actual apartment no. that the mail (~~you~~ client req.) arriving at mailbox has to route to

Most clients know the port that they should use

depending on the protocol that they're trying to speak to server with.

Protocol	Port used by client
HTTP	80
HTTPS	443

exercise: netcat: allows you to read from/write to network connections using certain protocols

Terminal Window 1

`nc -l 8081`

- `-l`: says that listen to stuff happening on this port
- `8081`: this port

READING DATA

Anything that you type in terminal window 2 appears.

Terminal Window 2

`nc 127.0.0.1 8081`

- `127.0.0.1`: special IP address that always points to your local machine (local machine's IP address)
- `8081`: this port

WRITING DATA

This terminal is entering a communi. channel with the machine at this IP address at port 8081

(viii) Thus, once server receives req., it is able to read it cuz it understands the HTTP format
Server understands that when you go to ae.io, you're trying to see the HTML of "ae" and so it returns a response viz. the HTML of 'ae' to browser which receives response and renders the HTML on the page for you

renders: turns HTML code into text, imgs, multimedia, interactivity effects, etc.