Agenda.

→ Quad Trees.

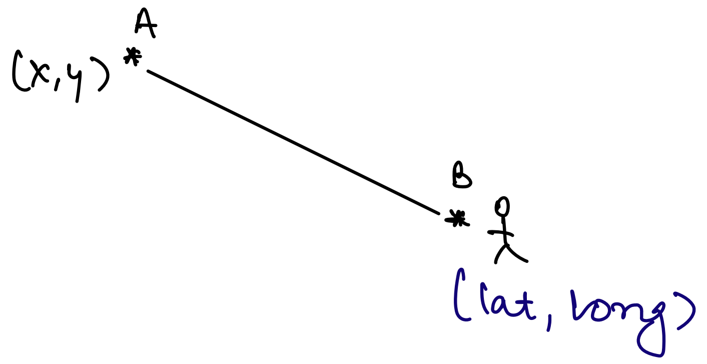
→ Design Uber.

# Google Maps ⇒ Nearest places of interest.

Uber | Ola ⇒ Nearest cabs.

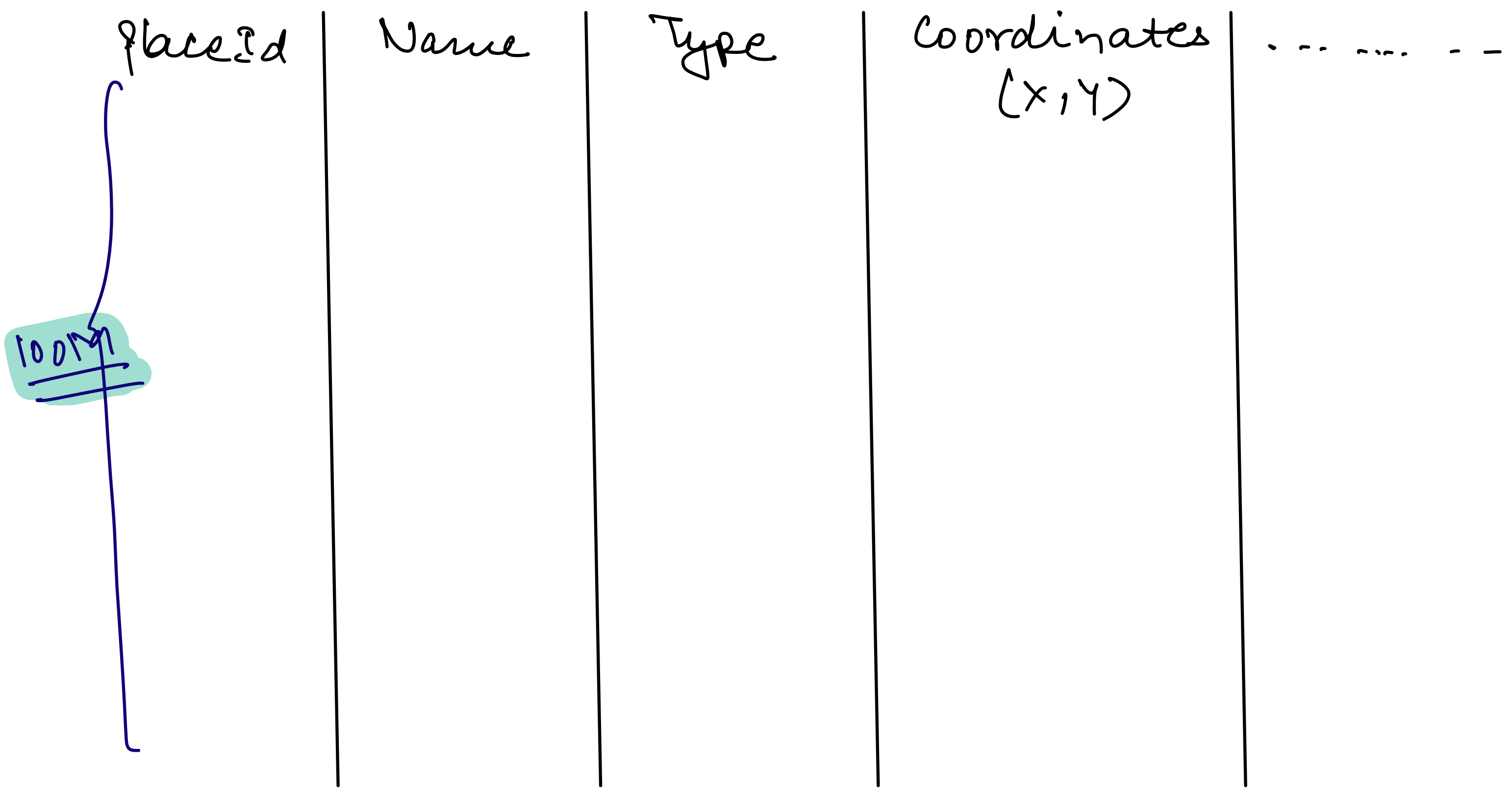
Swiggy | Zomato ⇒ Nearest restraunts.

#



$$\text{Dist}(A, B) = \sqrt{(x - lat)^2 + (y - long)^2}$$

I) Aggregate all the places of interest in table with their location coordinates.

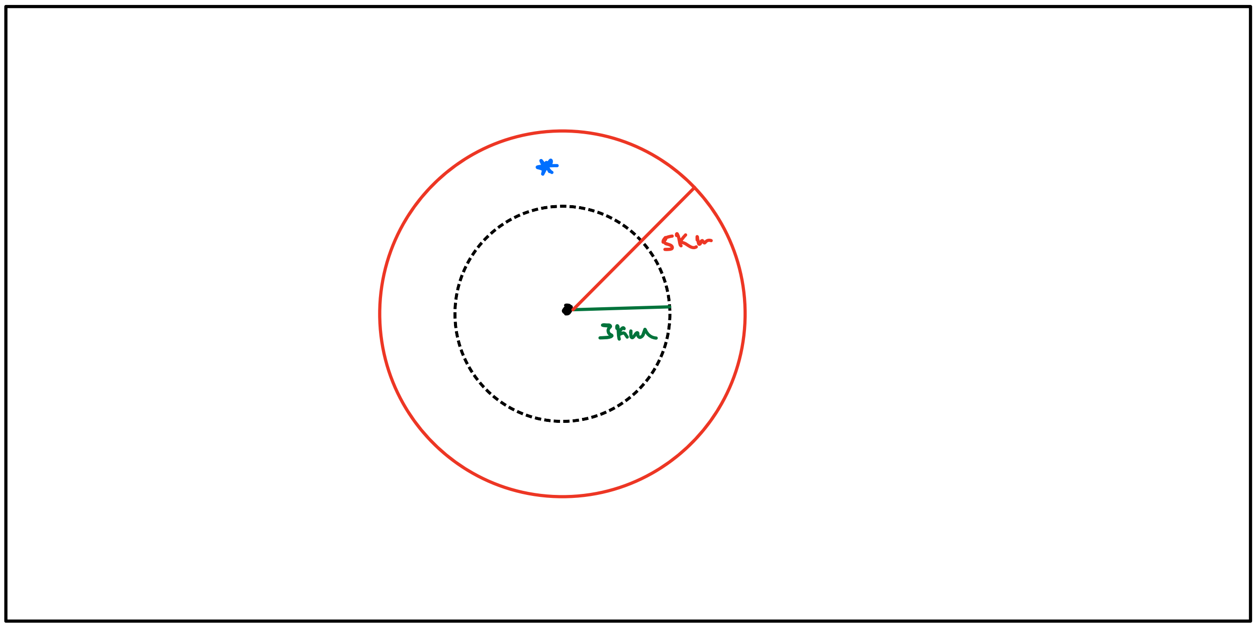


⇒ K nearest places for a particular user.

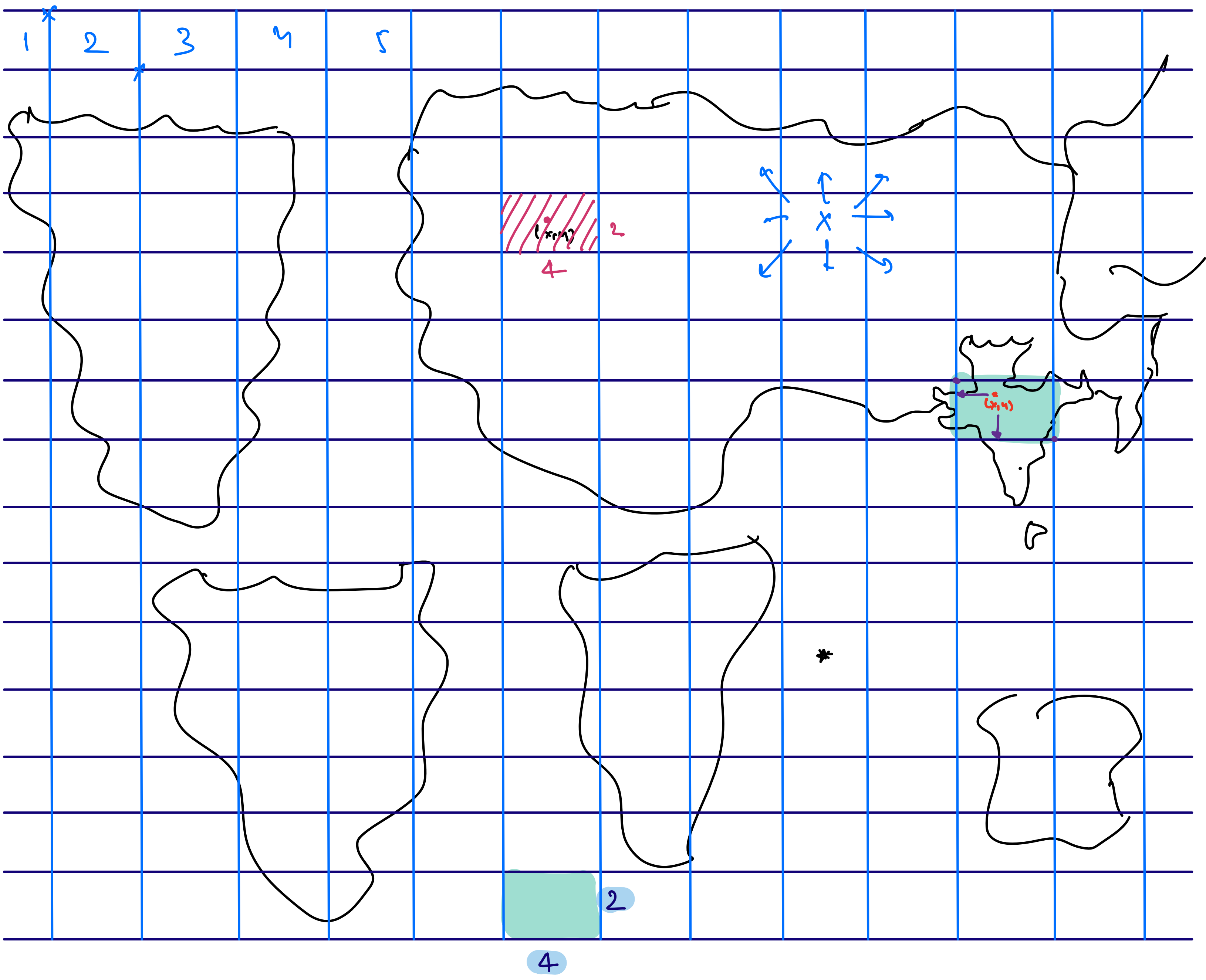
⇒ User location : (x, y).

Iterate through the table & find the first K places of interest wrt the user after sorting in the increasing order of their distance.

#


5km
3km

QUAD TREES.



I) Super-imposte a grid on the world map.

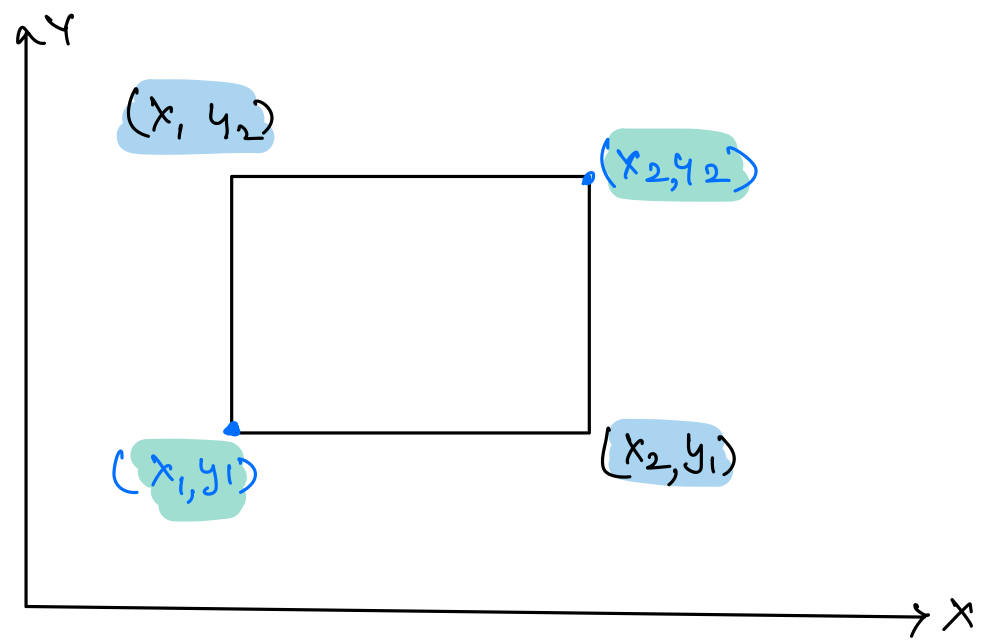
II) Every grid will have a unique id.

III) Grid size = 4×2 km
(Assumption)

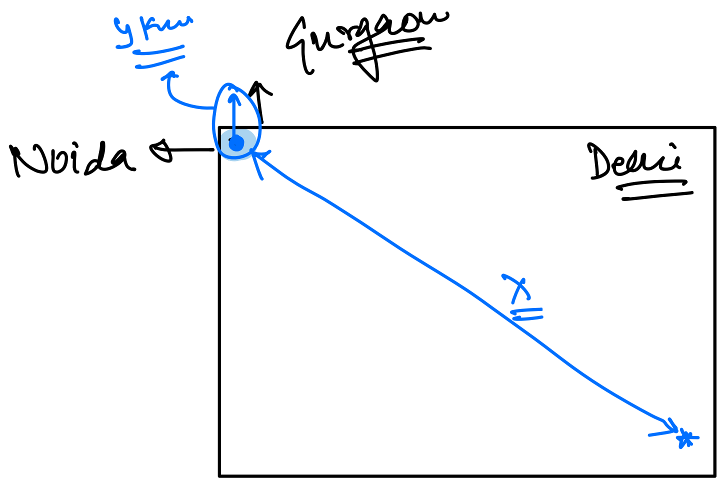
PlaceId	Name	Type	Coordinates (x,y)	gridId
—	—	—	—	1025
				107

→ Aggregate all the places of the interest with their details and location as well.

→ Store all the places of the interest in a table along with the gridId they belongs to.

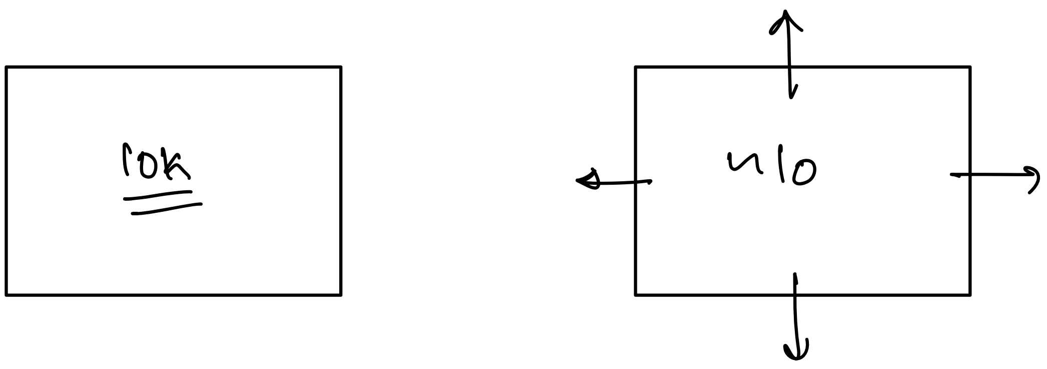


⇒ This idea of super imposing the grids on the entire world makes the world better addressable.



$y < x$.

⇒ No. of places of interest will be different grids.

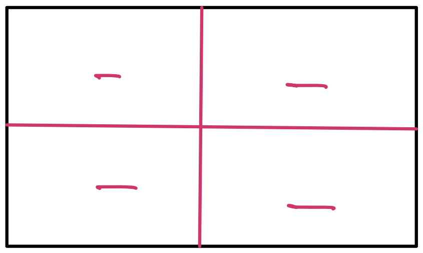


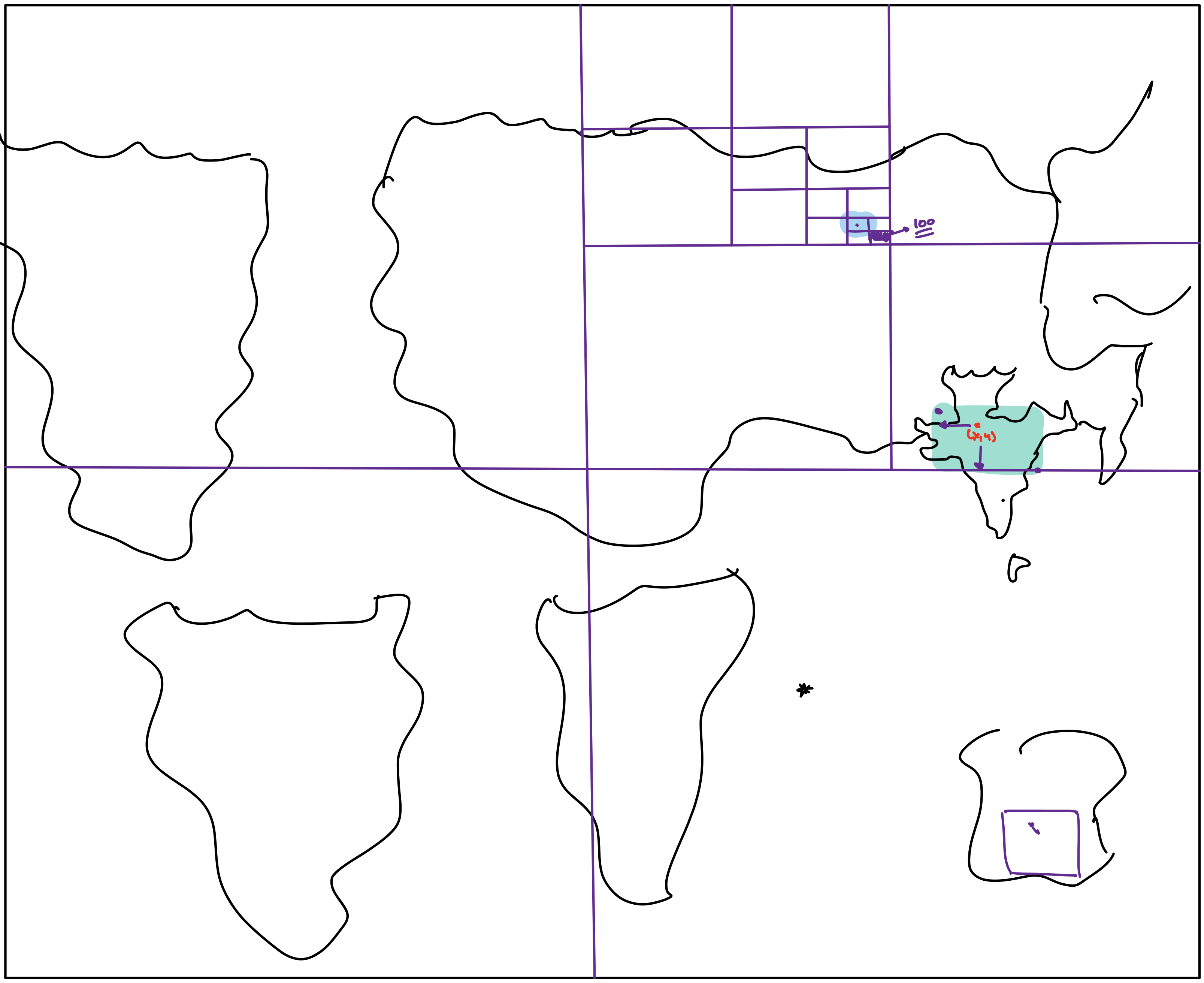
⇒ Sol^n : Dynamic sized grids.
↳ Based on # of places of interest in a Grid.

⇒ Recursive

⇒ Every grid should have <= 100 places of interest.

⇒ If a grid has > 100 places of interest then divide the grid further into 4 grids.

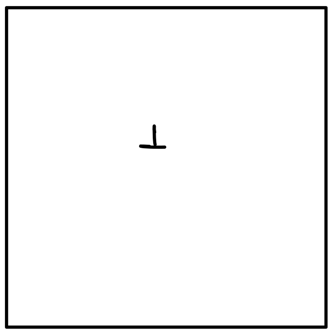




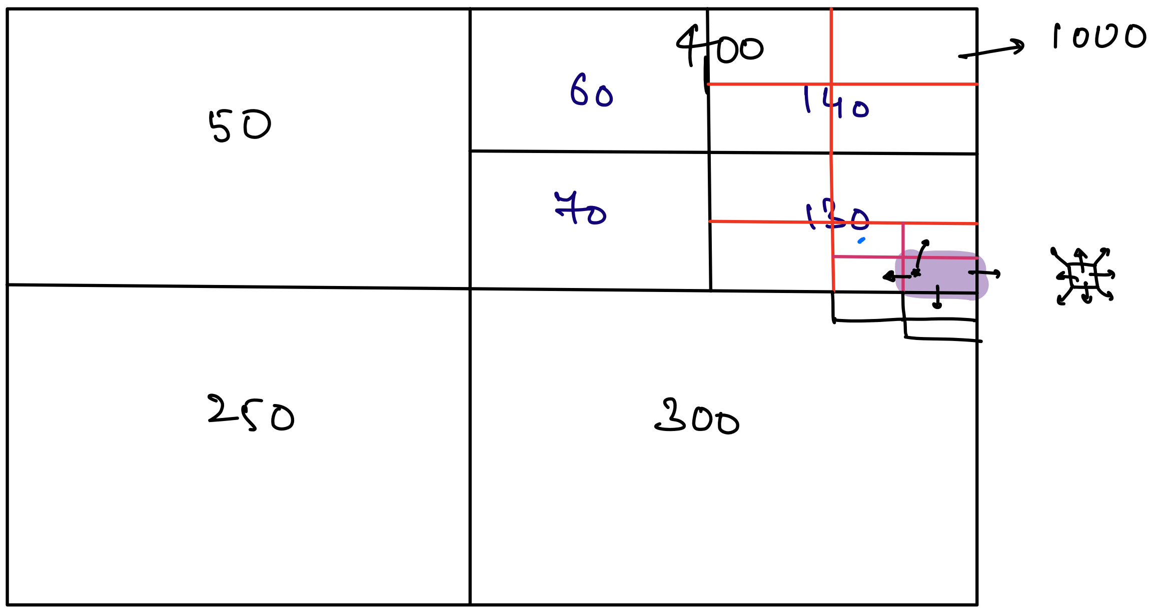
Recursively, keep dividing each grid into 4 equal sized if until we have # of places of interest >/ 100.

```
x = select count(*)
    from places
    where grid-id = 1;

while (x > 100) {
    keep dividing the
    grid into 4 grids.
}
```



⇒



⇒ Quad Tree : Pre processing step.

↳ Dividing each grid into 4 equal sized grids.

⇒ Storage Requirements.

↳ Quad tree Cells + $\underbrace{\text{places of interest.}}$

id	type	lat	long	gridId
16B	50B	16B	16B	16B

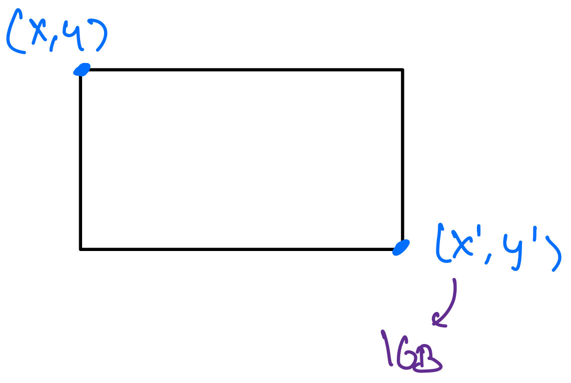
1 place ≈ 200B

100M places = 100M × 200B

$= 20 \times 10^3$ MB

= 20GB.

Quad tree Cells. ⇒



⇒ 64B.

No. of grids = $\frac{100M}{10}$ = 10M

Storage = 10M × 64B

= 640 MB < 1 GB.

= 0.64 GB.

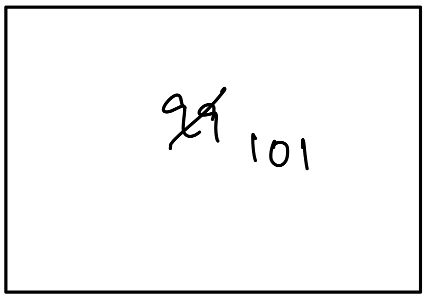
Total storage ≃ 21 GB.

⇒ No sharding required.

⇒ No. of places of interest can be added (or) removed from the system.

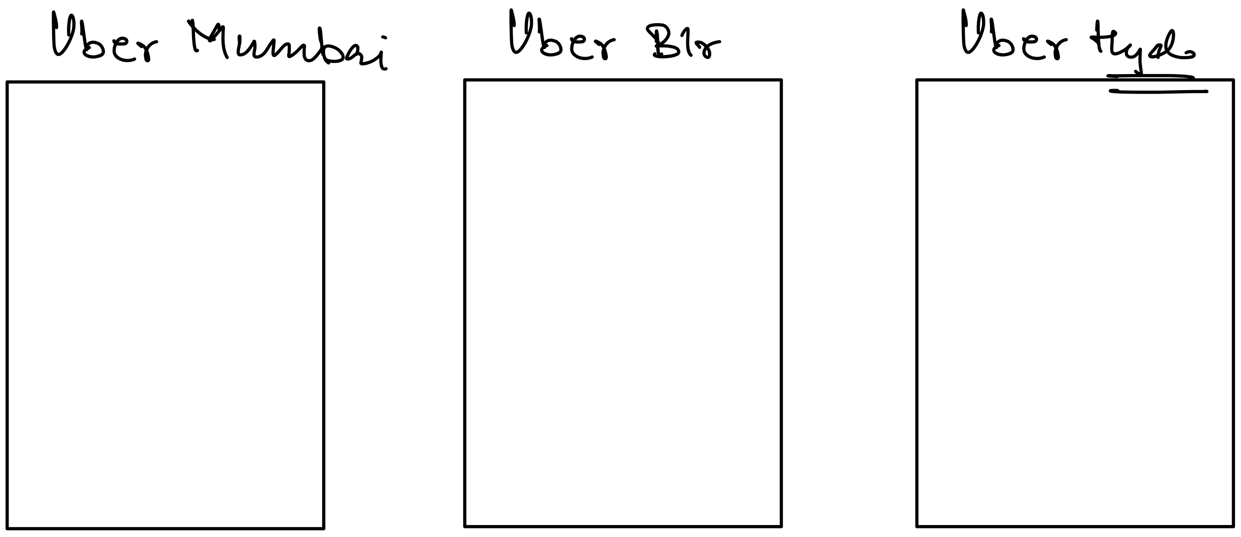
↳ This may require the division/merging of grids.

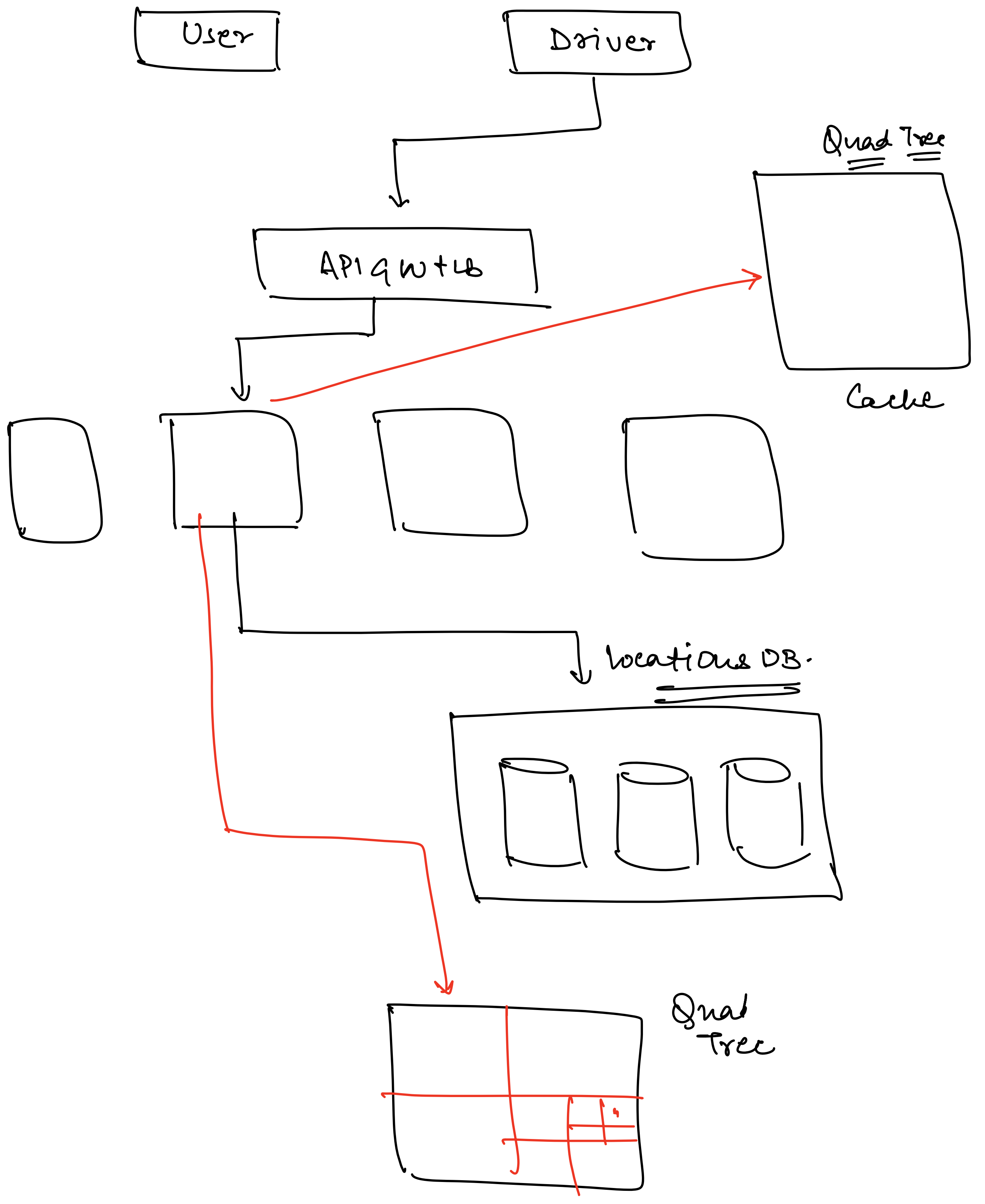
⇒ We can do the division/merging operation in batches.



Uber

$\hookrightarrow$ Intra City rides.




User
Driver
Quad Tree
API gw + LB
Cache
locations DB.
Quad Tree

⇒ Driver changes their location from one grid to another grid.

↓

Division | Merging every 10|20 mins

⇒ Drivers will keep sending their location constantly to Uber backend.

⇒ WebSockets

- → Http (Polling)
- → Long Polling
- → WebSocket.