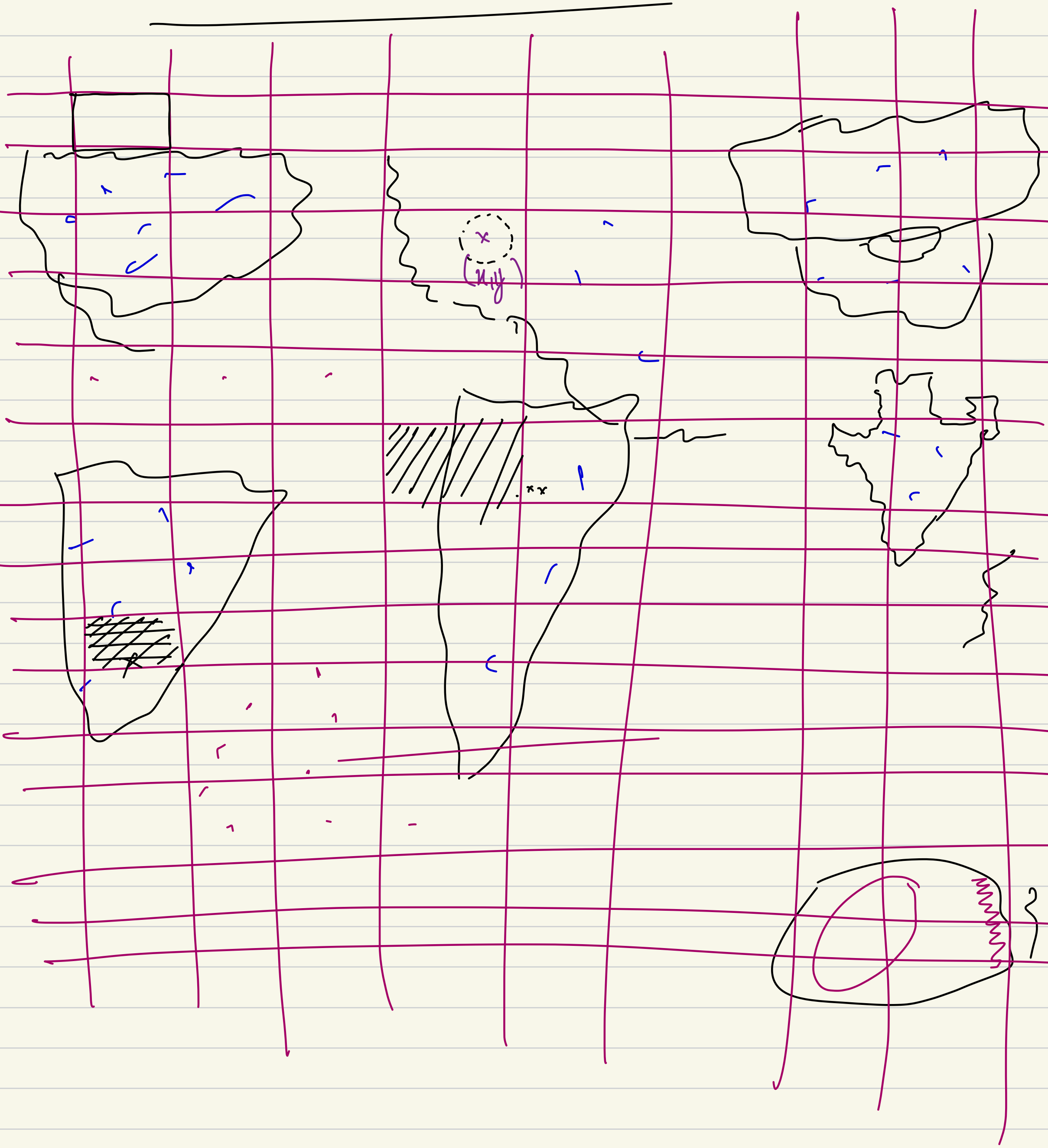
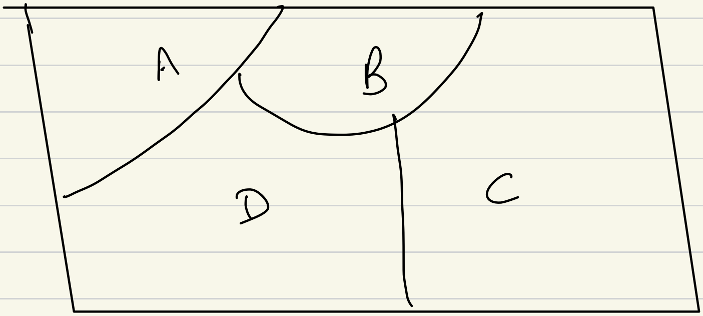
Quad Trees + Uber



mutually exclusive

+

collectively exhaustive



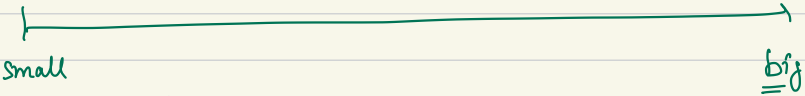
[0,4]

[0,4)

(0,4)

Creating grid cells of <u>equal</u> size across the world is a bad strategy.

→ unequal distribution 😛 😛



less densely

0 places

☹ ☹ ☹

densely

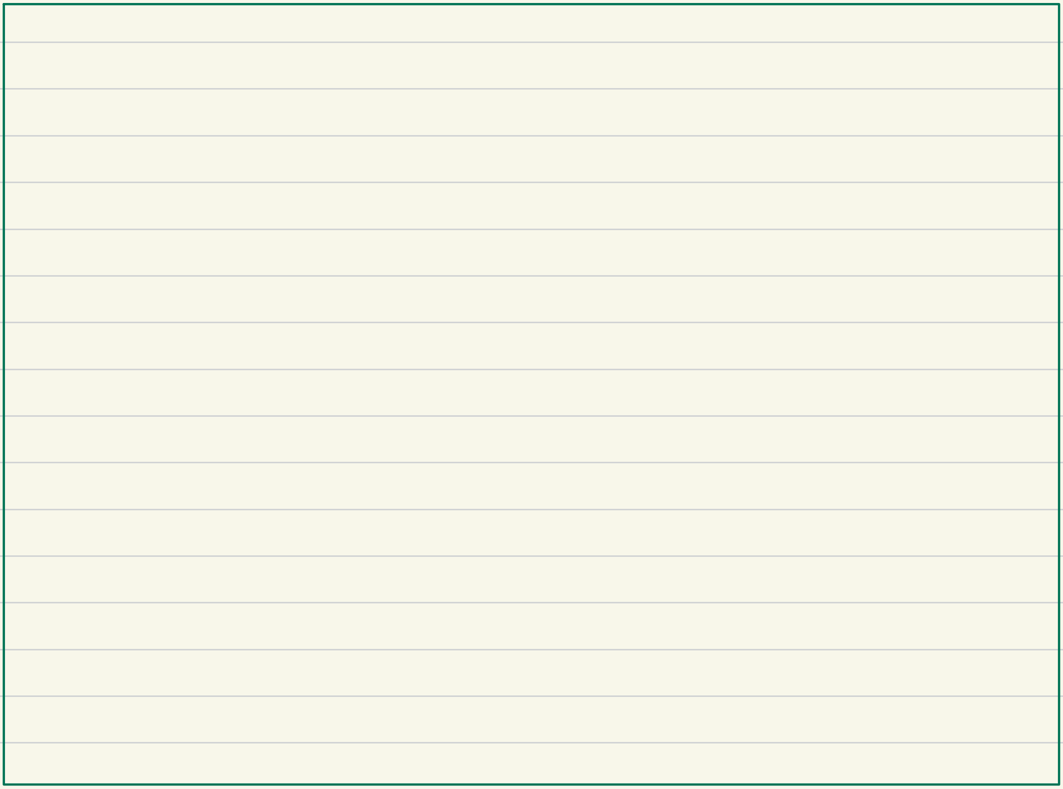
places ↑↑↑

query time ↑↑↑

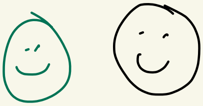
Quad Trees

divide the underlying space | world into grid cells, but variable sized grid cells. ☺

Step 1: Entire world is 1 grid cell

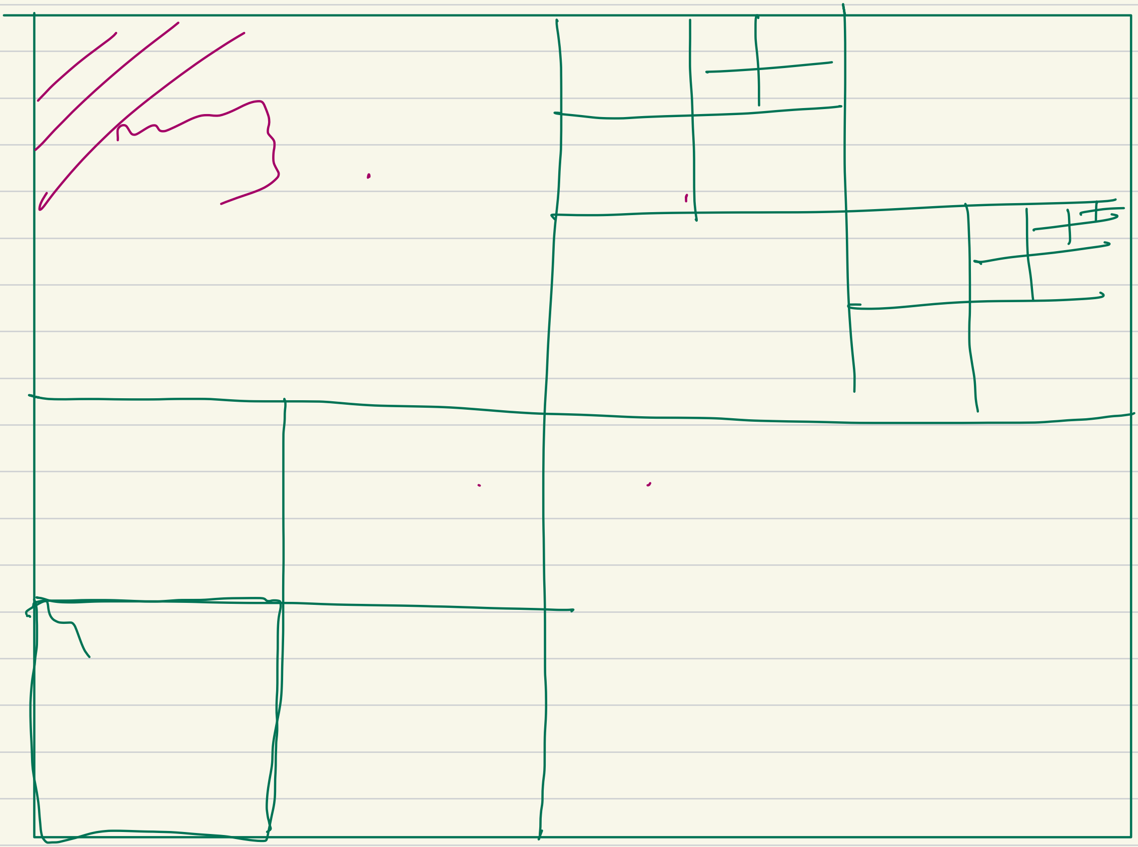


Threshold: 100 places | grid cell

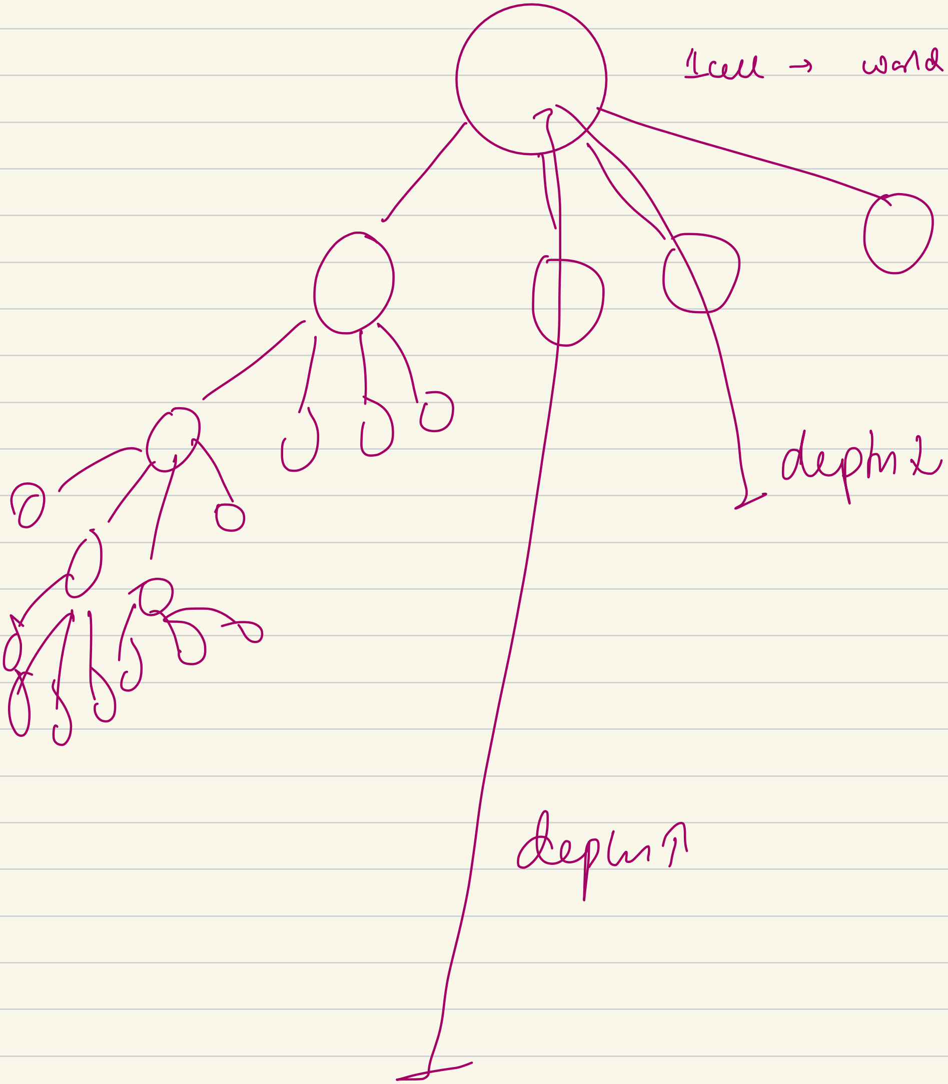


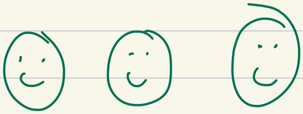
Question → do we have ≤ 100 places of interest

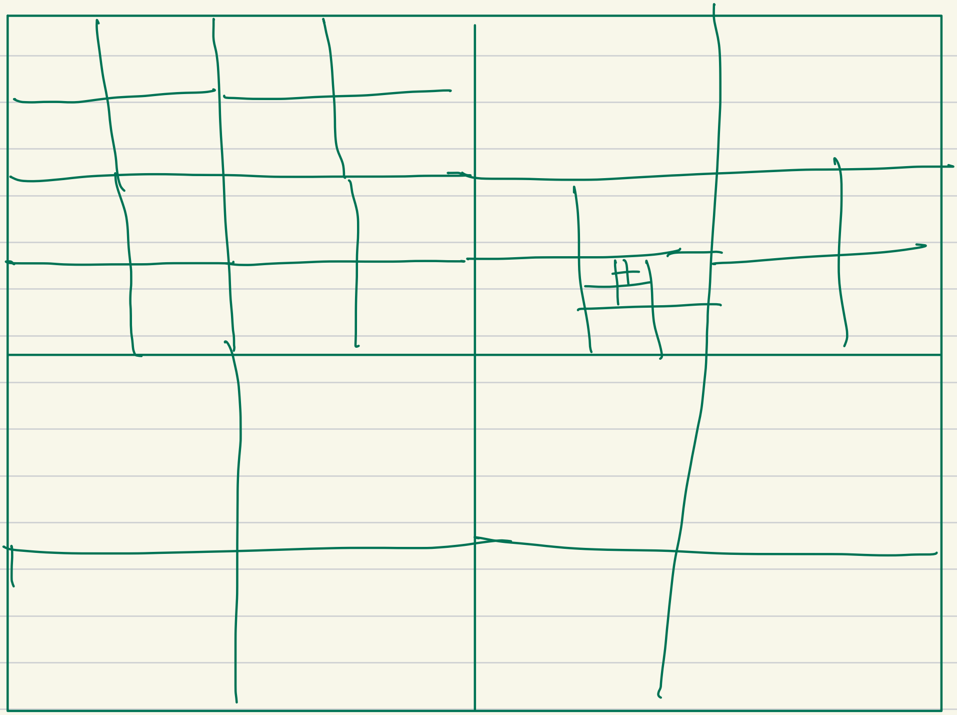
☆ No



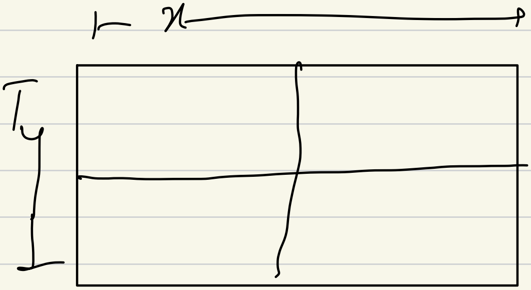
Recursive problems

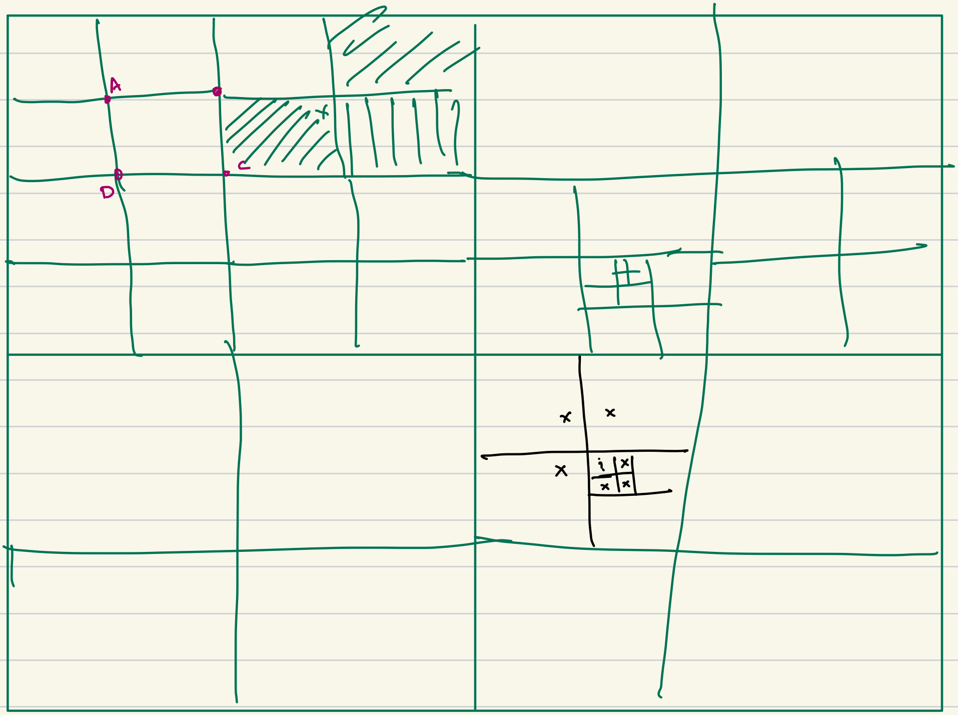






Why Quad??





cell_id to each cell

(A) Quad Tree Representation

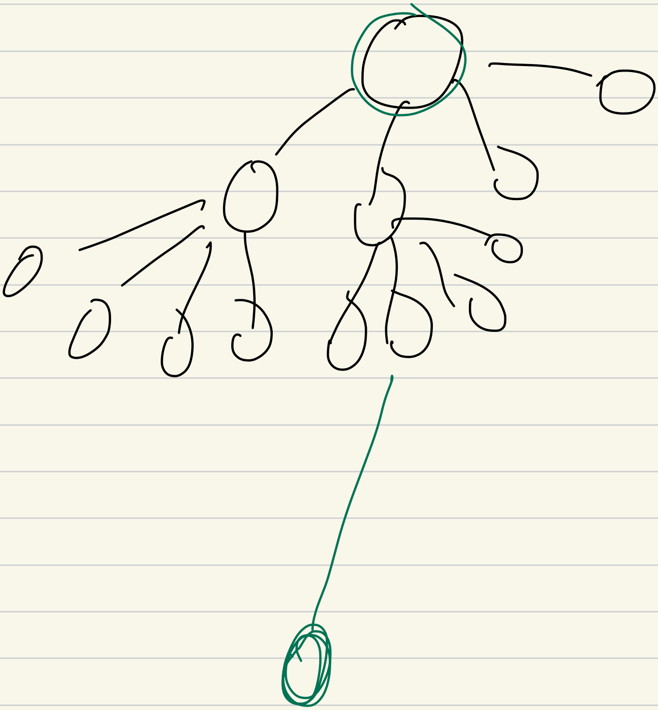
8B + 64B + 1B + 8B + 32B

cell id (children)

cell_id	a, b, c, d	is_leaf	parent_cell_id
1	(x,y) (x,y) (x,y)		

(B) Places

8B	8B	10B	index 4B	4B	index 8B
place_id	[x, y]	name	type	meta	cell_id



Quad Tree compromises of 2kinds of nodes

leaf node

are actual cells of quad Tree

non leaf nodes

logical amalgations of leaf cells

Algorithms

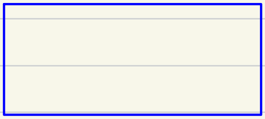
(I) Find Nearby Places (userx
userY
Type of Place
~~at max~~ limit
)

```
(I)   find Nearby Places ( user x
                           user y
                           Type of Place
                           at_max
                             ↙
                           limit )

☆

  (1)  leaf_cell_id_user    =  find Grid (x,y)
  (2)  list_of_places_cell  =  SQL Query Places [ ]
  (3)  SORT ( list_of_places ) → distance
                               → popularity
  (4)  Return   sorted_list
```

PROBLEM:- Maybe one cell where user is located is not enough.

①  donot have enough no. of Temples

② User was at the boundary | near the boundary of cell

To solve for this problem

Updated ☆

(I) find Nearby Places (userx

userY

Type of Place

at_~~max~~

→ limit)

☆

☺ ⇒ (1) leaf_cell_id_user = find Grid (x,y)

⇒ (2) list_neighbour_cell_id

(2) list_of_places_cell = SQL Query Places [ ]

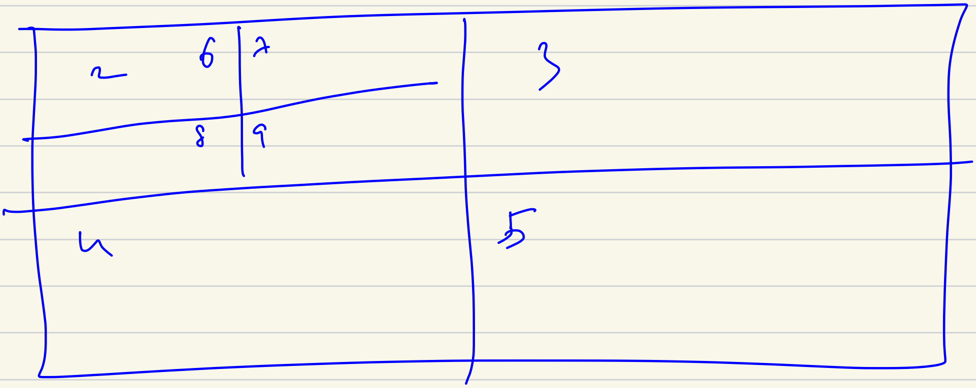
(3) SORT (list_of_places) → distance → popularity

(4) Return sorted_list

100 M
place
(world)

600 (6 cell)

100 (1 cell)

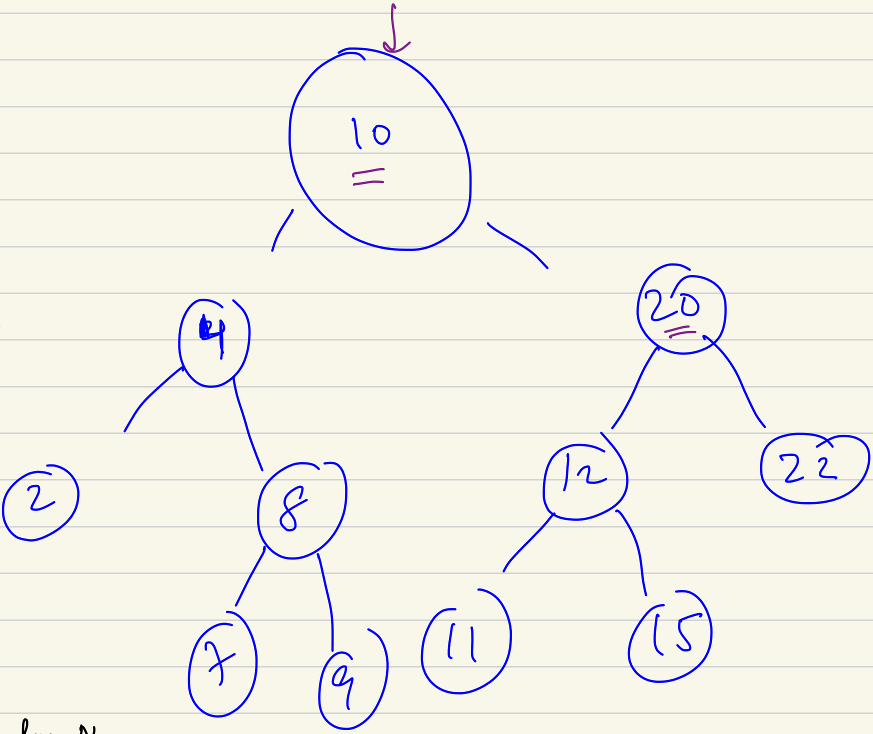


1
2
3
4
5
6
7
8
9

10:36 – 10:40 PM Break ☺ ☺

Ex. B-S-T

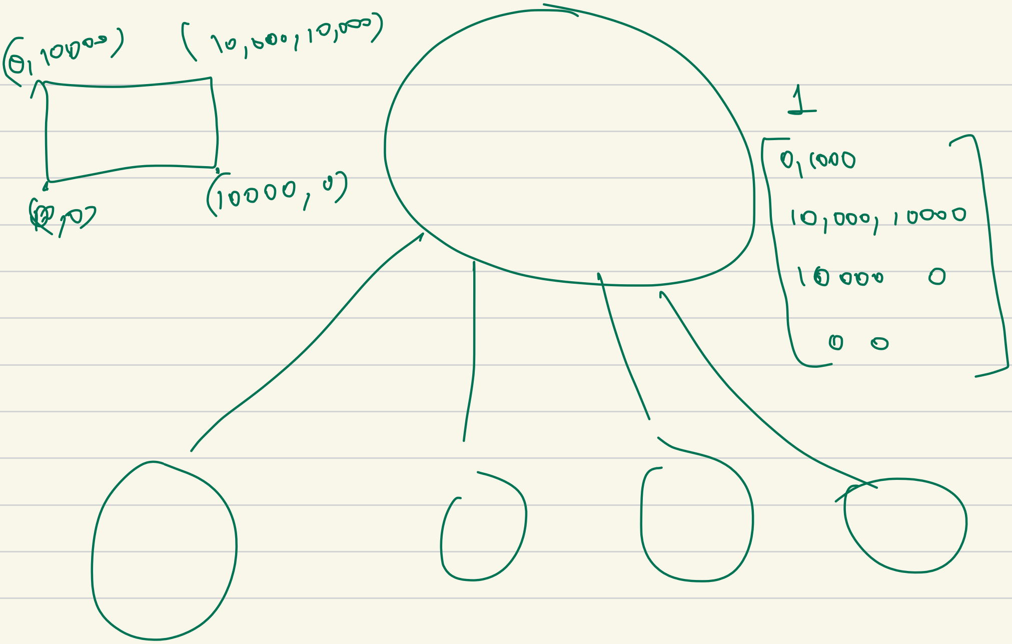
Where is 12 ??

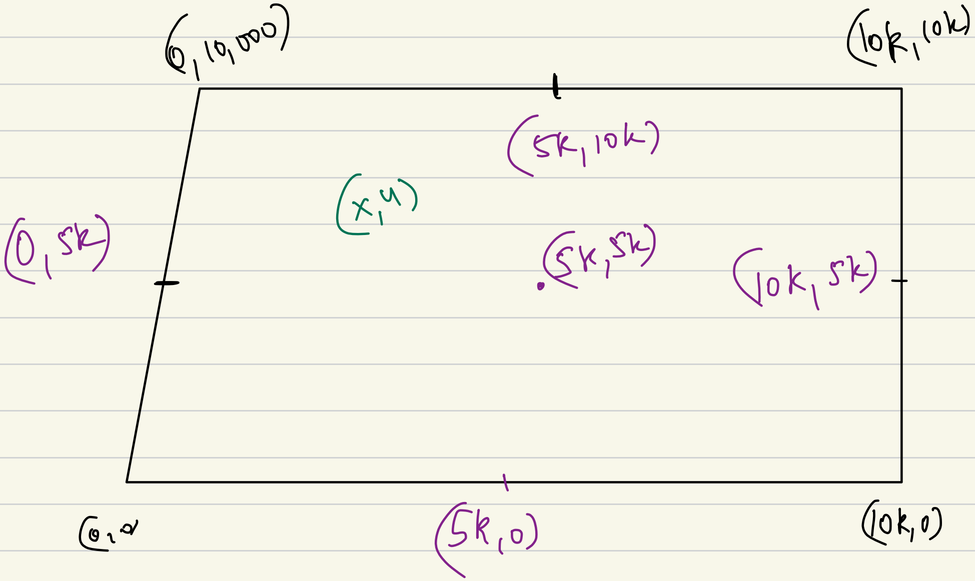


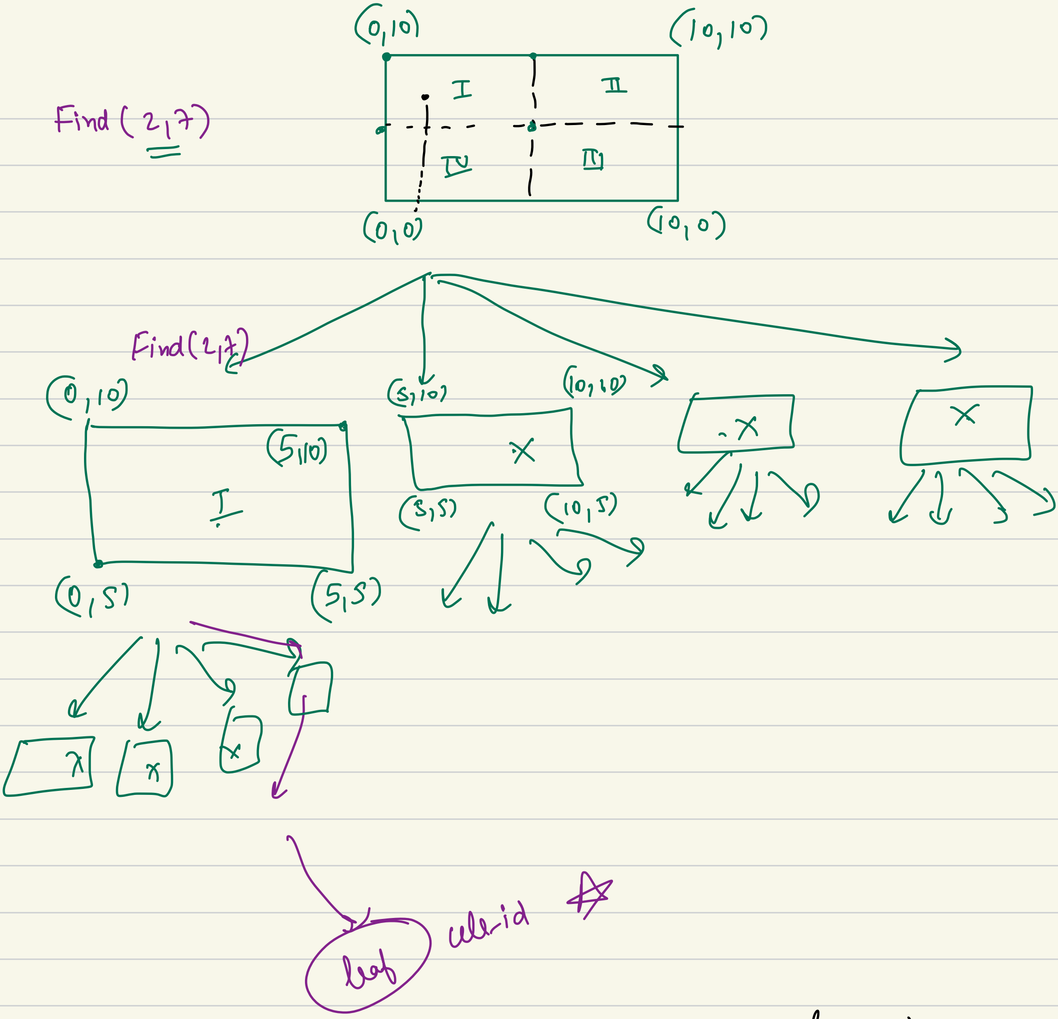
T.C = $O(1) \times \log_2 N$

findGrid(x,y)

Just like in a binary search Tree, we can find a node by comparing and selecting one child (automatically rejecting the other), we can do the same in Quad Trees and identify the cell-id by using [x,y]


(0, 10000)
(10,000, 10,000)
(0,0)
(10000, 0)
1
0, 1000
10,000, 10000
10000 0
0 0


(0, 10,000)
(10k, 10k)
(5k, 10k)
(x, y)
(0, 5k)
(5k, 5k)
(10k, 5k)
(0, 0)
(5k, 0)
(10k, 0)

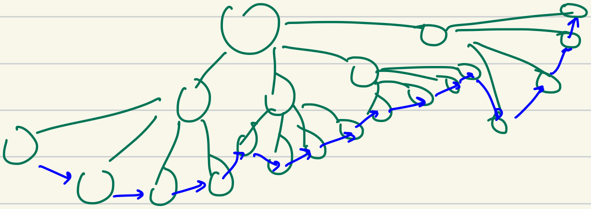


$$T.C = O(1) \times \log_4 N$$

$$= O\left(\log_4 N\right)$$

Neighbouring Cells ?

①

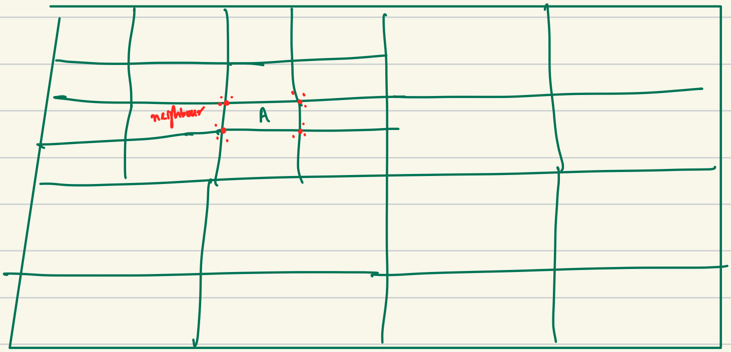


next node problem ☆

OR

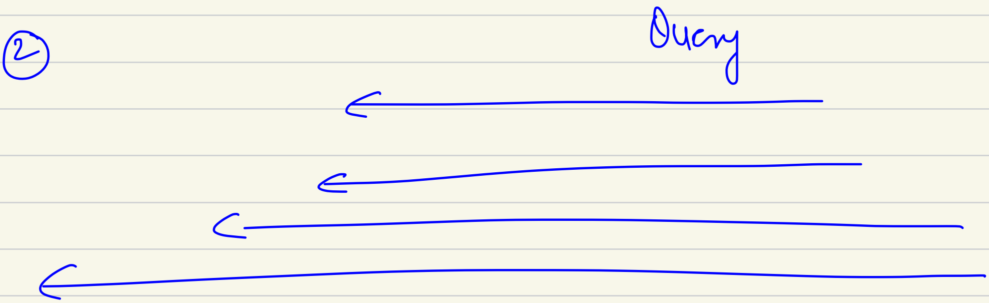
cell_id , parent-cell-id.

neighbouring cells ☆



① Pre computed The Quad Tree

②



findNearbyPlaces( )

③

addNewPlace()

[x, y] Temple meta_data Ⓐ

cell_id ← findGrid(x, y) Ⓑ

if |cell| = Threshold Ⓒ

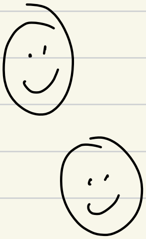
divide cell

I

II

1 + III

IV



Update Places Ⓓ

4

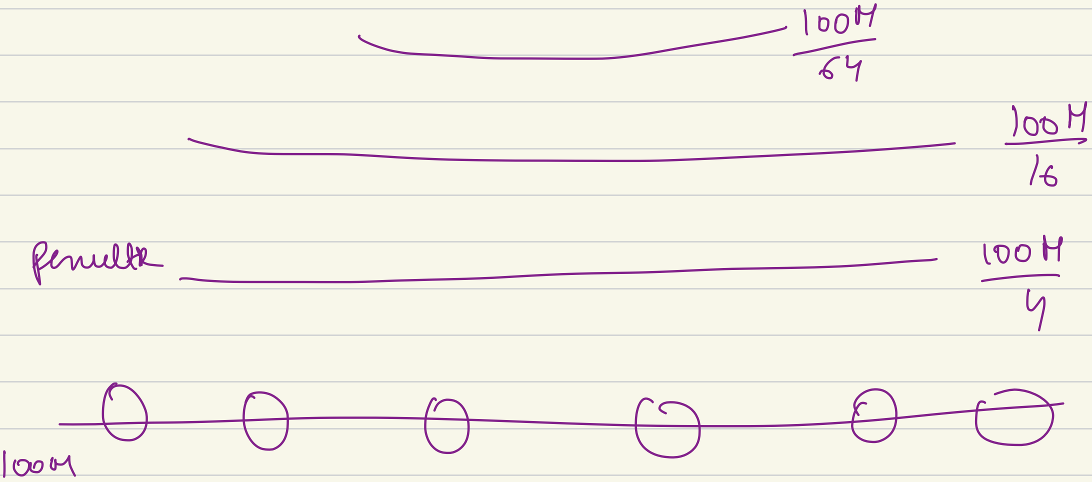
delete Place

① 100 M places in the world

② 100 = Threshold

③ # leaf cells = 100 M

worst case



Total Nodes

$$= 10^8 + \frac{10^8}{4} + \frac{10^8}{16} + \frac{10^8}{64} + \cdots + 1$$

$$= 10^8 \left(1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} \cdots \frac{1}{10^8}\right)$$

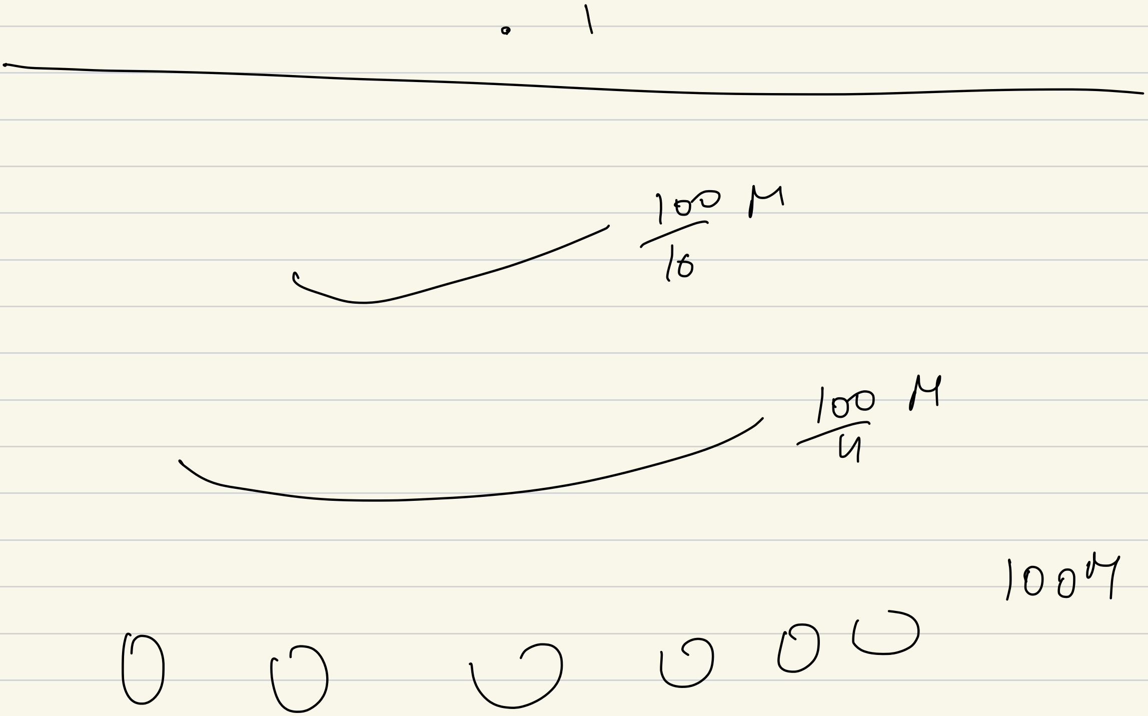
$$= 10^8 \left[\frac{1}{1 - \frac{1}{4}} \right]$$

$$\frac{4}{3} \times 10^8$$

$$= 1.33 \times 10^8$$

Total nodes = 133 Million

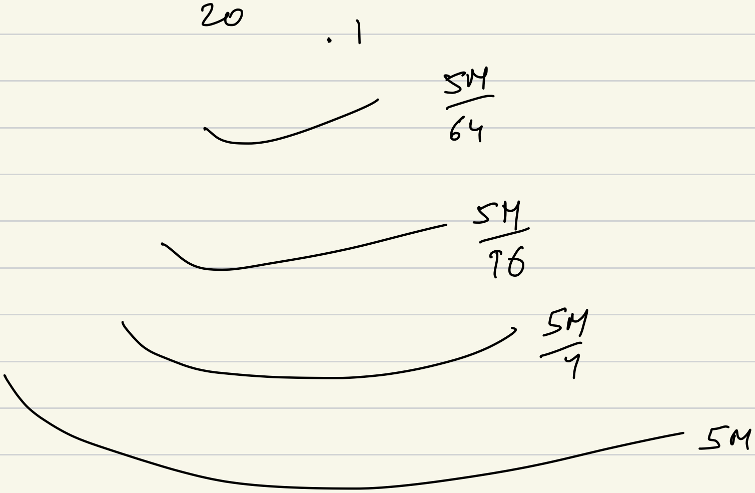
	Best Case	Avg Case	Worst Case
Places	100 / Node	20 Places / node	1 Place / Node



Total Nodes = 133 Million

○ leaf node ≡ 20 places

$\frac{100M}{20}$ = 5M leaf nodes



Total Node = $5M + \frac{5M}{4} + \frac{5M}{16} + \frac{5M}{64} + 1$

$= 5M\,(1.33)$

= 6.5 million nodes overall

Cell → 100 Byts / cell

6.5 M × 100 B

650 MB

1GB cells ☺

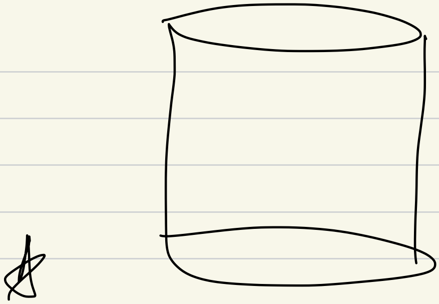
Places

100 M × 50 B

5000 MB

5GB → Places

Space ⇒ 6GB Quad Tree



Commodity hardware
for storage

↓

5TB - 8TB

Sharding is NOT required ☺ ☺

Time complexity ⇒ 6.5 M

6,500,000

6.5×10^6

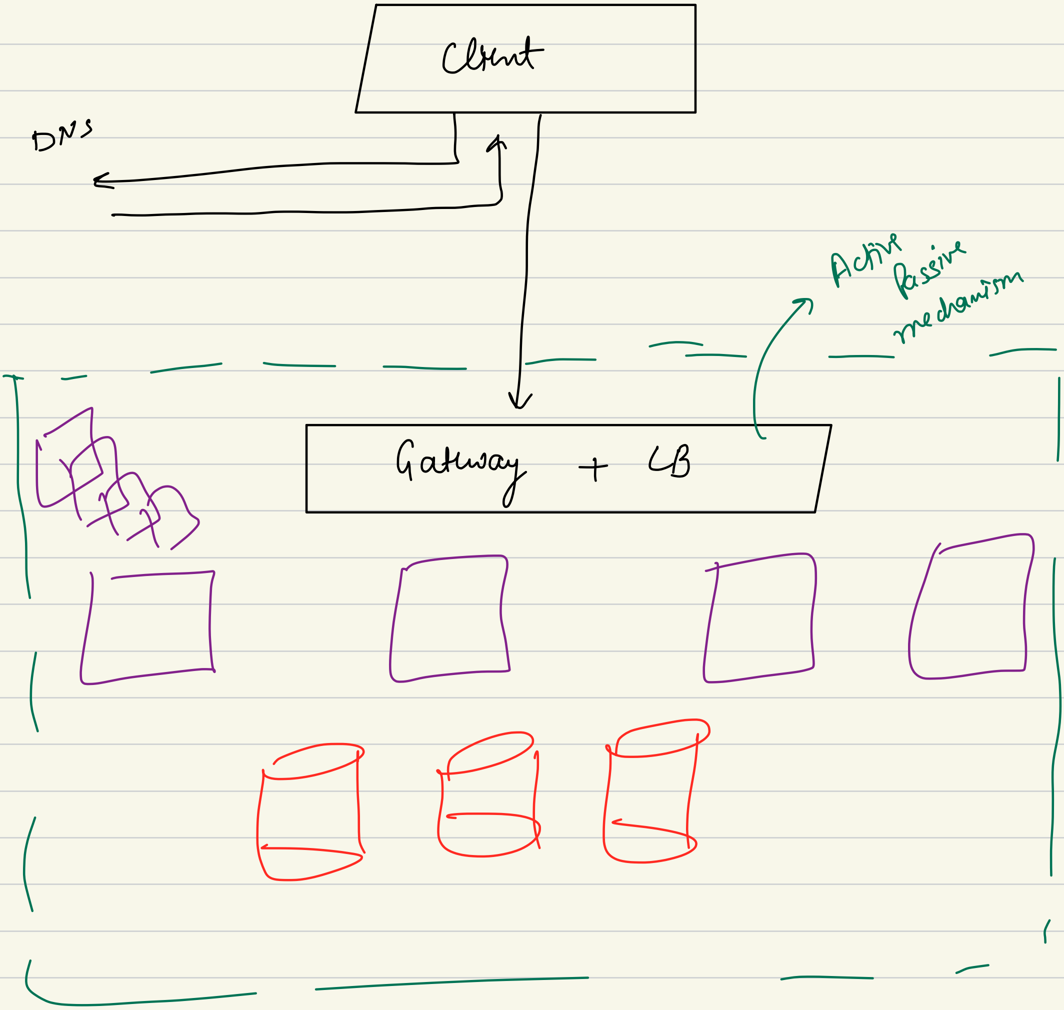
$T.C = \log_4 N$

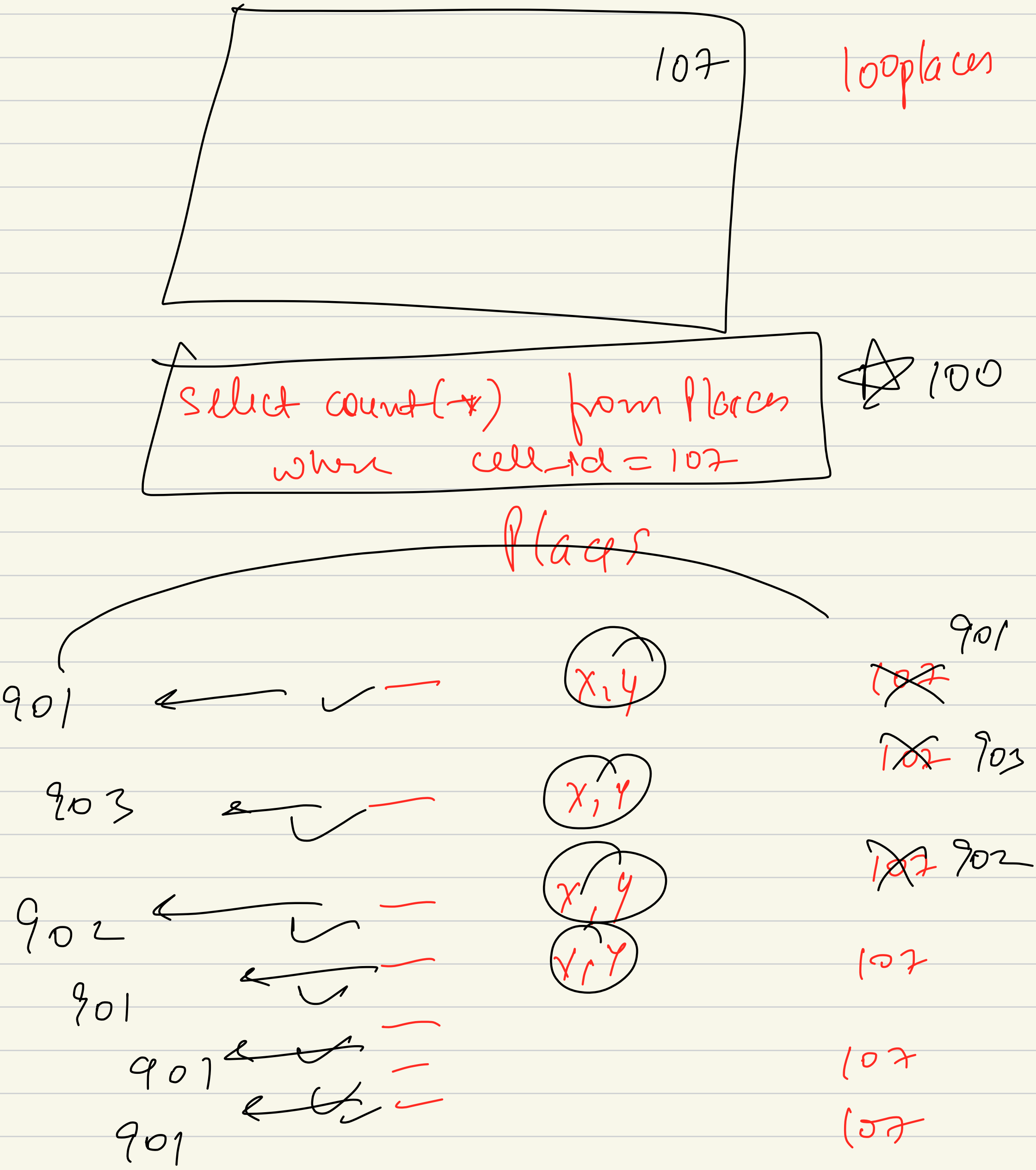
$\log_4 (6.5 \times 10^6)$

Quad Trees

Uber ⇒ Quad Tree + +

Question





```
select count(*) from Places
where cell_id = 107
```

divide (107) → 901, 902, 903, 904

cell

107	~~True~~ False	→ 901, 902, 903, 904
901	T	107
902	T	107
903	T	107
904	T	107

places