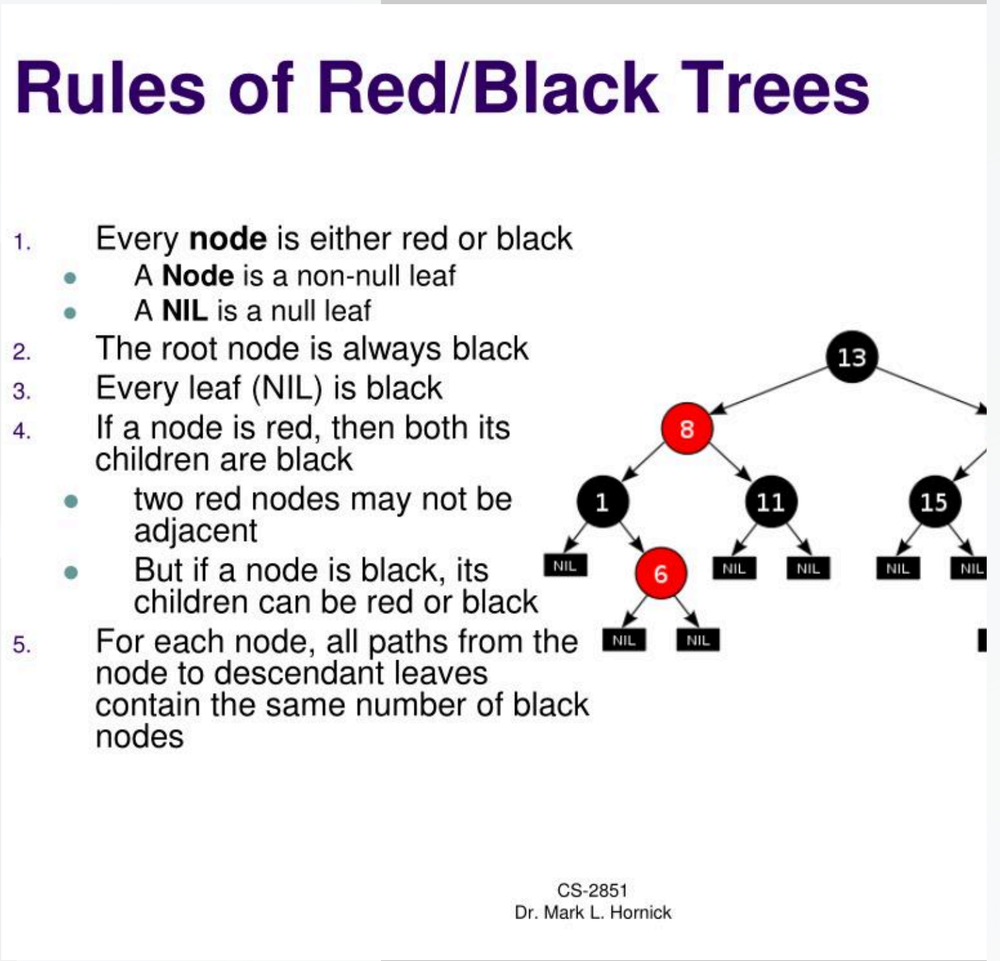
# RED BLACK TREES

JINANG SHAH (B22CS027)
VISHESH SACHDEVA (B22AI050)
AMBATI RAHUL REDDY (B22CS088)

# WHAT ARE RED BLACK TREES?

Red-Black tree is a binary search tree in which every node is colored with either red or black. It is a type of self balancing binary search tree. It has a good efficient worst case running time complexity.

Red Black Trees are a type of balanced binary search tree that use a specific set of rules to ensure that the tree is always balanced. This balance guarantees that the time complexity for operations such as insertion, deletion, and searching is always O(log n), regardless of the initial shape of the tree.




Courtesy : SlideServe.com

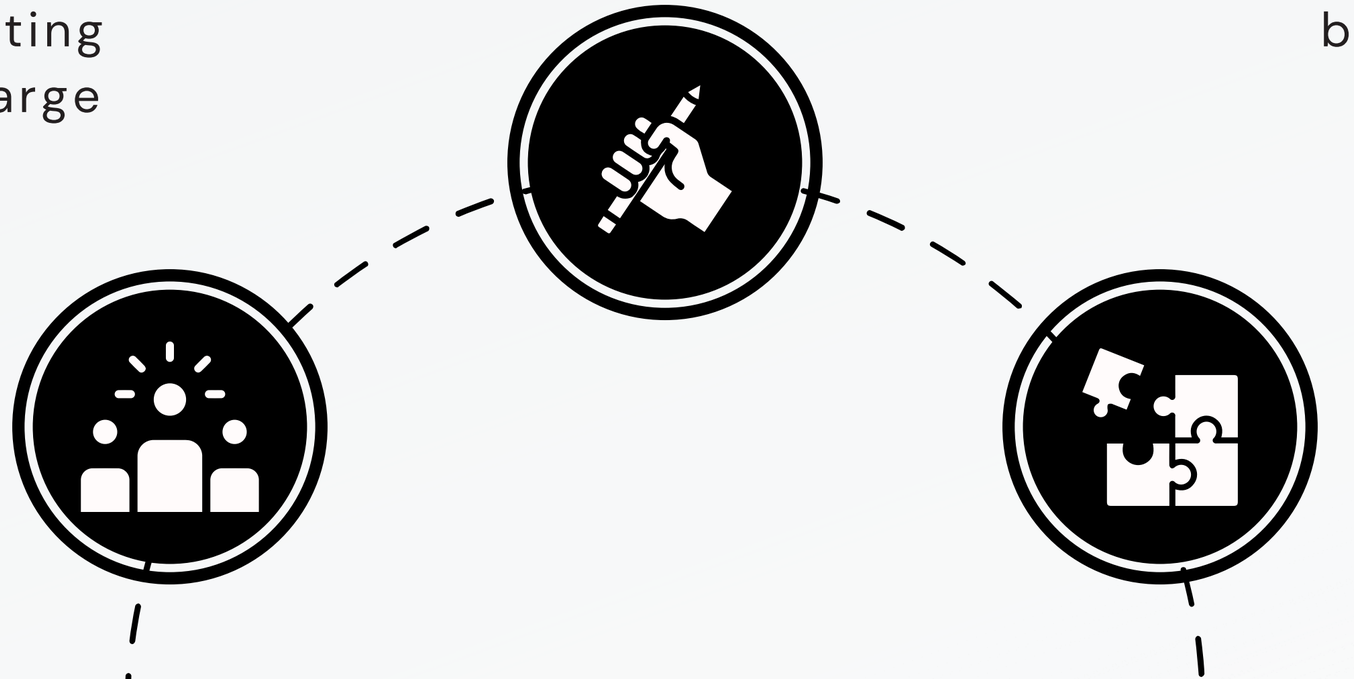
# OBJECTIVES

**Objective 1**

We aim to implement Red Black Trees which can be used for inserting, deleting and searching a large dataset.

**Objective 2**

To Provide a fast and efficient way of searching in a big datasets.

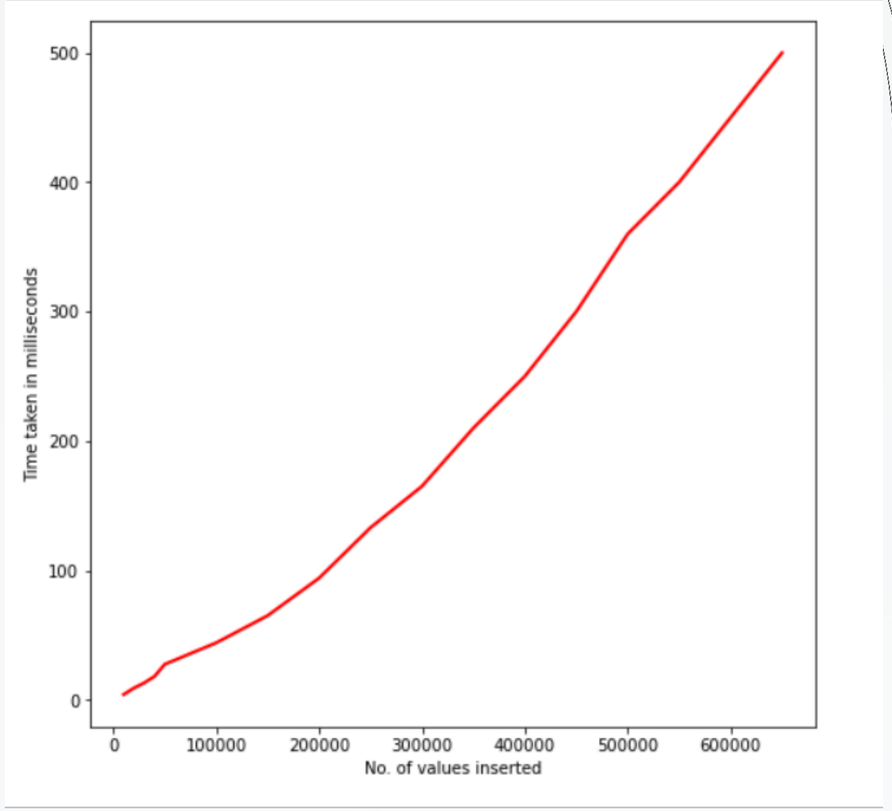


# TIME COMPLEXITIES IN RB TREES

- Insertion, Deletion and Searching are performed in O(logn) time complexity.
- Inspite insertion of large amount of data single searching and deletion is performed in 0.044 sec and 0.002 sec respectively.

```
y = np.array([4,9,13,18,27.4,44,65,94,133,165,210,250,300,360,400,450,
fig, ax = plt.subplots(figsize = (8,8))

ax.plot(x, y, "r", linewidth=2.0)
plt.xlabel("No. of values inserted")
plt.ylabel("Time taken in milliseconds")
plt.show()
```

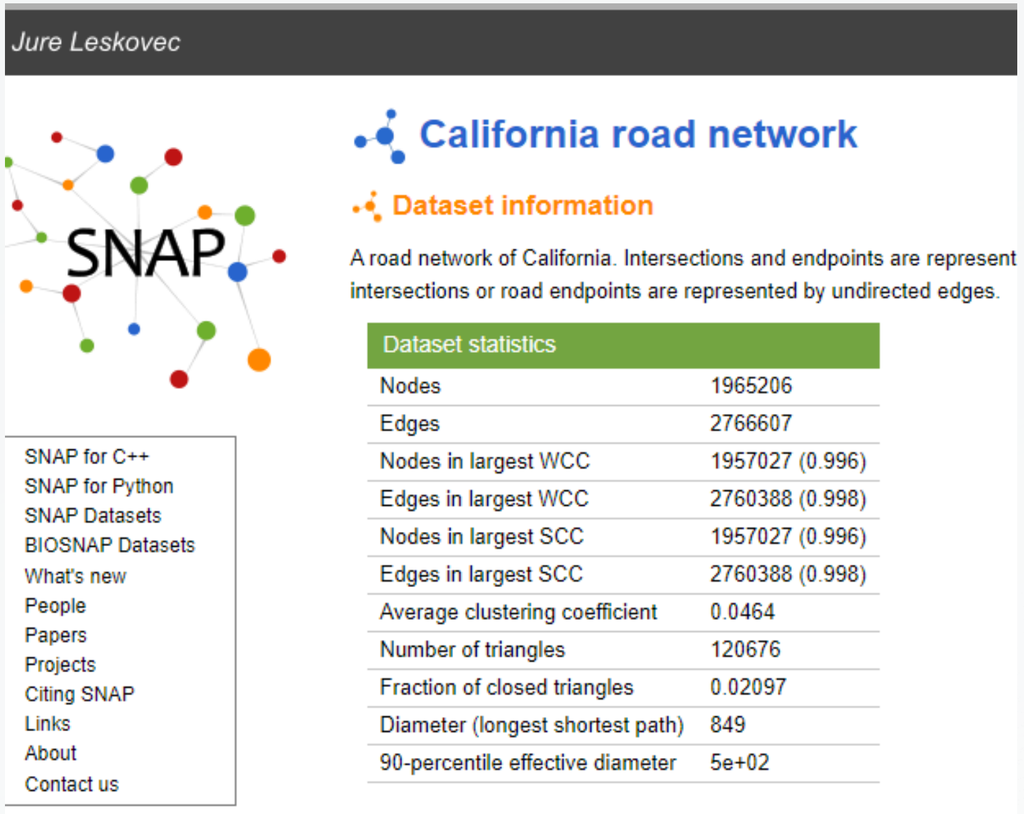



Number of data vs insertion time

# IMPLEMENTING RED BLACK TREES ON LARGE DATASET

**Space Used** :
1) RB node : key(string), rightchild, leftchild, parent, color, index(int)
2) Adjacency list (space : nodes + edges)
3) Vertices : It is an array which stores the string we input corresponding to its index.(so that access can be in constant time)

**Procedure** :
1) We are taking input and instantiating a RB node for every unique vertex in the graph.
2) As we insert new nodes we are assigning each node with an index value(which is initial 0 and incremented as we take input) and storing its key to an array of vertices at this assigned index.
3) Also simultaneously we are making an adjacency list of data.



# IMPLEMENTING RED BLACK TREES ON LARGE DATASET

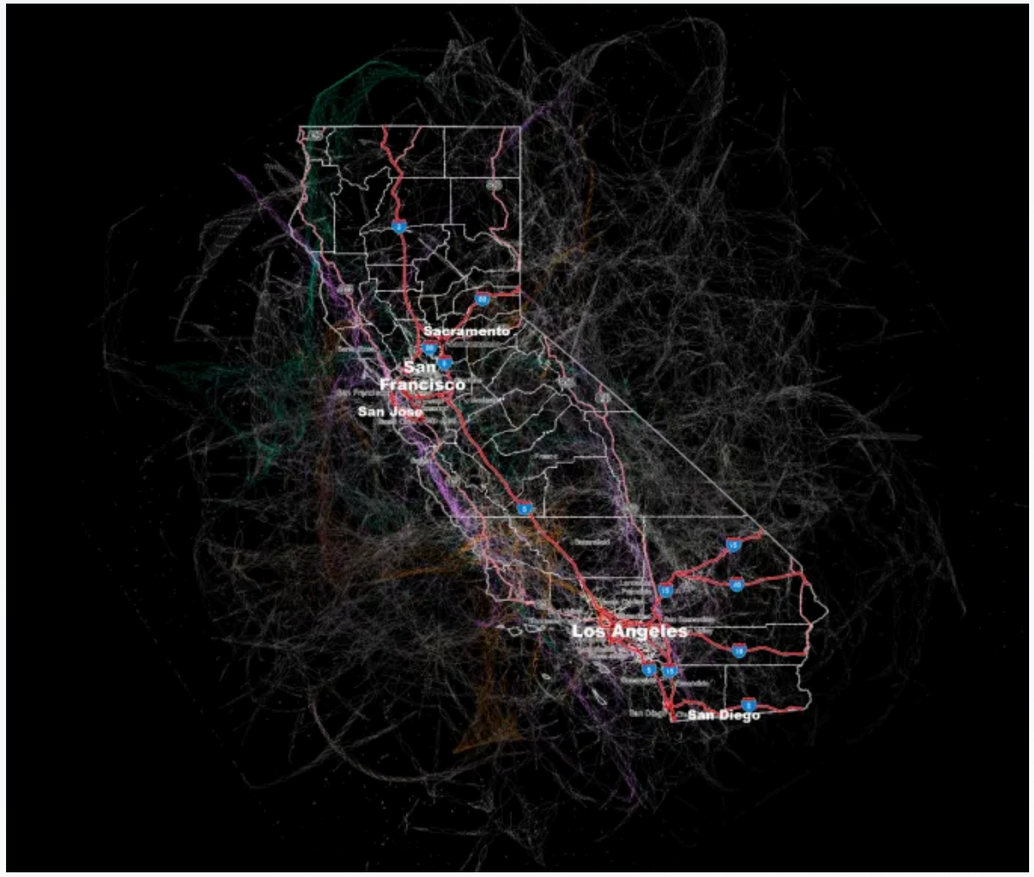
**How it works :**

Let’s say we have a starting point ABC and ending point XYZ.

1) We will search ABC and XYZ in the RB tree and get its index values.
2) After we get 2 indexes we have an adjacency list by which we can find paths by DFS or BFS.
3) If we want to print a path we can get it from the vertices array as it stores string corresponding to index.

**Time Taken** : (averaged)

1) Single insert after 1900000 insertion = 0.0067 ms
2) Single search atter 1900000 insertion = 0.0047 ms
3) Total time taken (insertion for all nodes) = 7.42 s




courtesy : Studentwork.prattisi.org

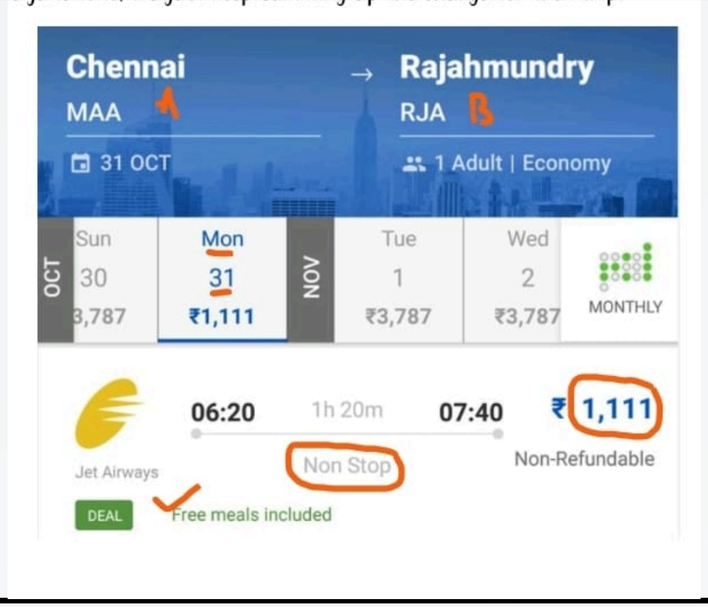
# ADVANTAGES

Red Black Tree Code is relatively easier to understand and implement. At same time it is faster also.

We can retrieve Information very Fast as we saw the analysis of RB Trees.

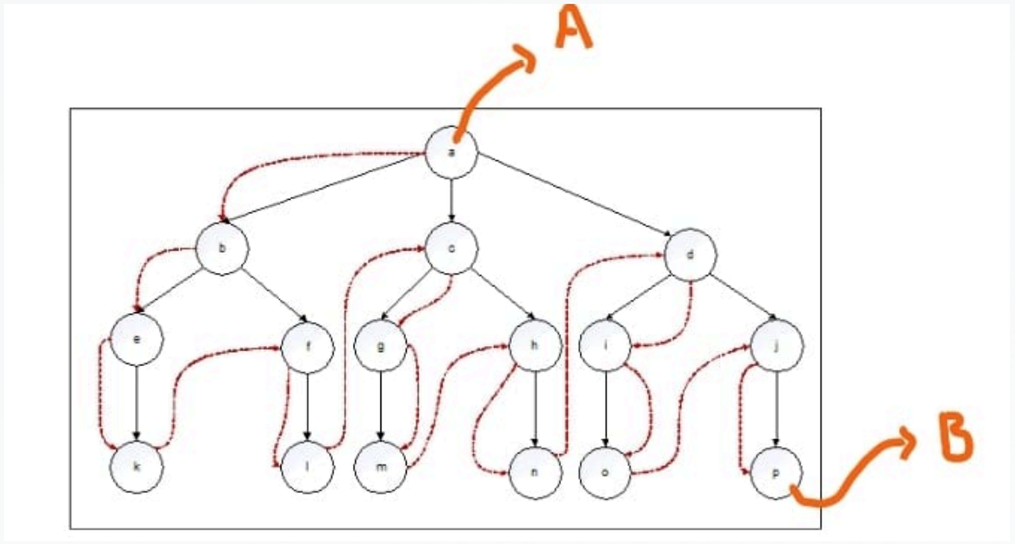
RB Trees take less steps to balance.In Max two Steps RB tree is balanced.

# TRAVEL BOOKING APPS




- Searching of origin and destiny will be faster if we make a RB Tree of Stops.

- If we don't have direct transport from origin to destiny then we will try another transport link from junction which are connected to origin and this info can be stored.



# SOME APPLICATIONS OF RB TREE


Github Link : https://github.com/jinangshah21/Travel-Desk-/tree/main/Travel_RBT

# THANK YOU