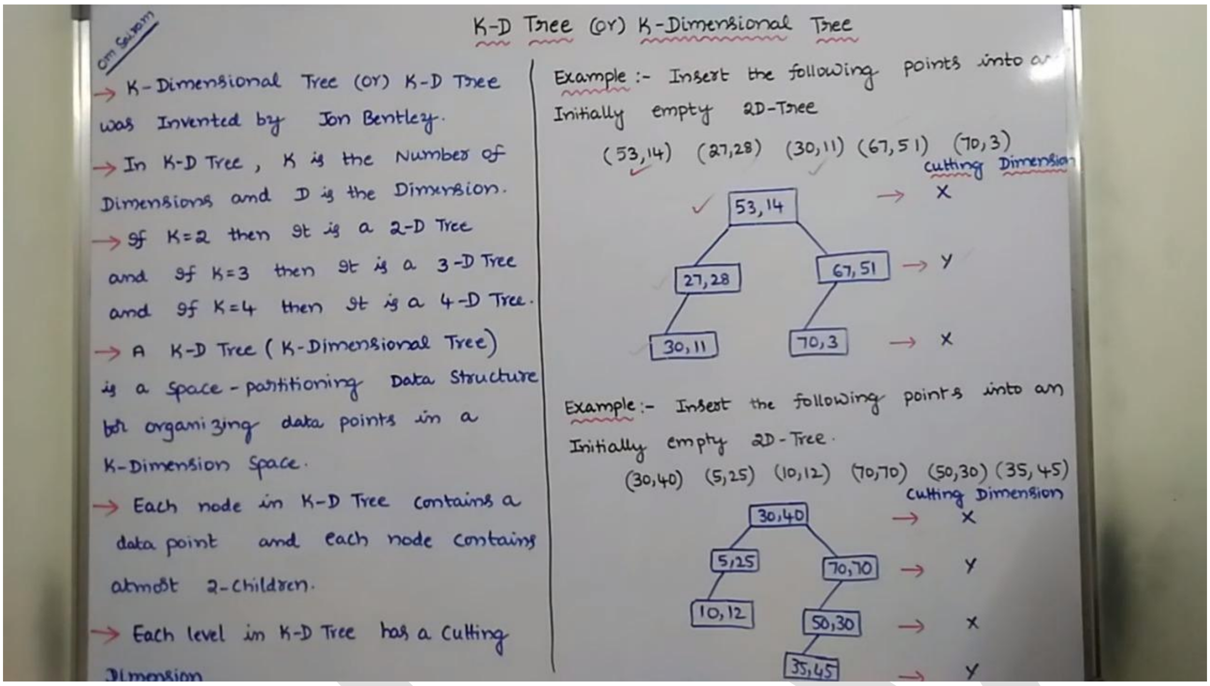
KD TREE: https://www.youtube.com/watch?v=-e7oHgAepQA



QUAD TREES:

https://youtu.be/fA1Omsl1msU

https://youtu.be/xFcQaig5Z2A

PRIORITY SEARCH TREE:

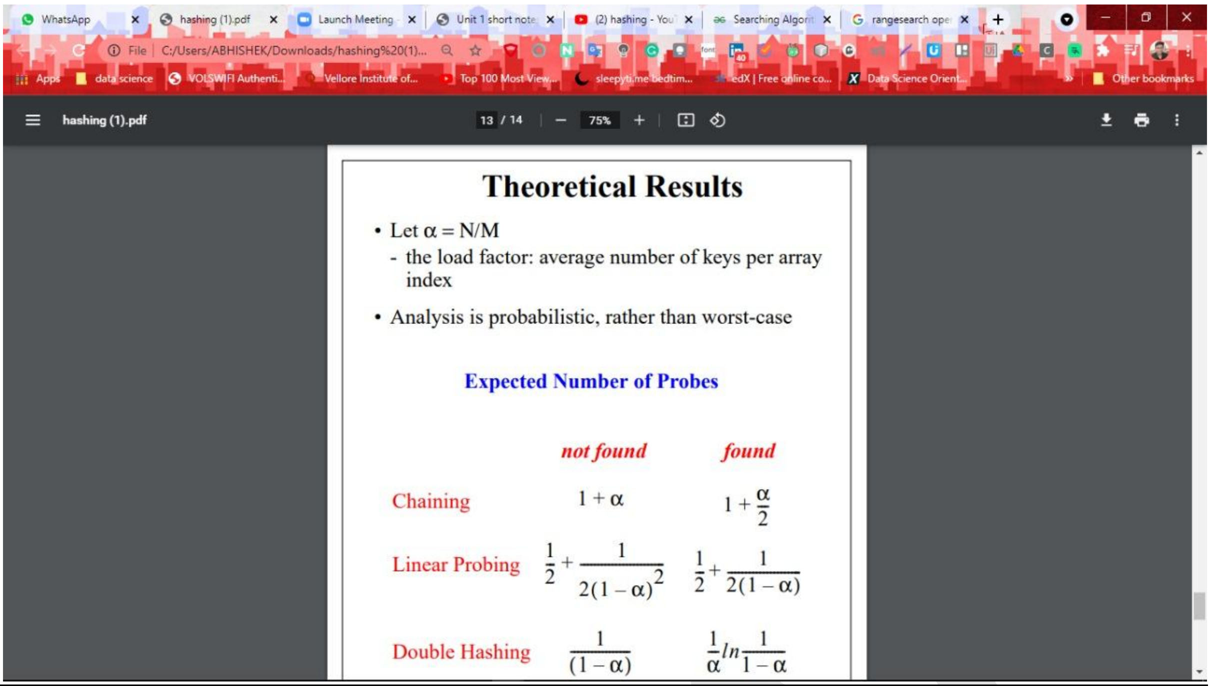
https://youtu.be/toqAuY8K0Rg

https://youtu.be/Rcbcapr46UY

~~~~~~

EXTRAS

Calculate load Factor
~~~~~~



Theoretical Results

- Let $\alpha = N/M$
  - the load factor: average number of keys per array index
- Analysis is probabilistic, rather than worst-case

Expected Number of Probes

	not found	*found*
Chaining	$1+\alpha$	$1+\frac{\alpha}{2}$
Linear Probing	$\frac{1}{2}+\frac{1}{2(1-\alpha)^2}$	$\frac{1}{2}+\frac{1}{2(1-\alpha)}$
Double Hashing	$\frac{1}{(1-\alpha)}$	$\frac{1}{\alpha}ln\frac{1}{1-\alpha}$

~~~~~~

TIME COMPLEXITY:

Data Structure	Time Complexity			
	Average			
	Access	Search	Insertion	Deletion
Array	O(1)	O(n)	O(n)	O(n)
Stack	O(n)	O(n)	O(1)	O(1)
Queue	O(n)	O(n)	O(1)	O(1)
Singly-Linked List	O(n)	O(n)	O(1)	O(1)
Doubly-Linked List	O(n)	O(n)	O(1)	O(1)
Skip List	O(log(n))	O(log(n))	O(log(n))	O(log(n))
Hash Table	N/A	O(1)	O(1)	O(1)
Binary Search Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))
Cartesian Tree	N/A	O(log(n))	O(log(n))	O(log(n))
B-Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))
Red-Black Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))
Splay Tree	N/A	O(log(n))	O(log(n))	O(log(n))
AVL Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))
KD Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))

~~~~~~

Unit - 2

How to differentiate between different types of Skip lists?

Deterministic skip lists: In deterministic skip lists, we have links between alternate elements at each level

Randomized skip lists:

In Randomized skip lists, we use a random() function which basically behaves like a coin flip.
For a node,
If random() = 1 then:
We add a link at that level of the node.
If random() = 0 then:
We don't add a link at that level of the node.

Perfect skip lists:

The number of nodes linked at each level is p = 1/2^n where n is the level number.
Let there be 8 elements.
At level 0, n = 0, p= 1:
8/ 1 = 8.
All 8 nodes are linked.
At level 1, n = 1, p = 1/2:
8/2 = 4
4 nodes are linked.
At level 2, n = 2, p = 1/4
8/4 = 2
2 nodes are linked.

~~~~~~

Splay Tree- https://www.youtube.com/watch?v=p6jiiP4i-Qo

Advantages of Splay tree: In the splay tree, we do not need to store the extra information. In contrast, in AVL trees, we need to store the balance factor of each node that requires extra space, and Red-Black trees also require to store one extra bit of information that denotes the color of the node, either Red or Black.

It is the fastest type of Binary Search tree for various practical applications. It is used in Windows NT and GCC compilers.

It provides better performance as the frequently accessed nodes will move nearer to the root node, due to which the elements can be accessed quickly in splay trees. It is used in the cache implementation as the recently accessed data is stored in the cache so that we do not need to go to the memory for accessing the data, and it takes less time.

Drawback of Splay tree: The major drawback of the splay tree would be that trees are not strictly balanced, i.e., they are roughly balanced. Sometimes the splay trees are linear, so it will take O(n) time complexity.

Algorithm for Insertion operation

```
Insert(T, n)
temp= T_root
y=NULL
while(temp!=NULL)
y=temp
if(n->data <temp->data)
temp=temp->left
else
temp=temp->right
n.parent= y
if(y==NULL)
```
~~~~~~

T_root = n
else if (n->data < y->data)
y->left = n
else
y->right = n
Splay(T, n)

Properties of 2-3 Trees

A 2-3 tree follows the below mentioned properties.

- Every internal node in the tree is a 2-node or a 3-node i.e it has either one value or two values.
- A node with one value is either a leaf node or has exactly two children. Values in left sub tree < value in node < values in right sub tree.
- A node with two values is either a leaf node or has exactly 3 children. It cannot have 2 children. Values in left sub tree < first value in node < values in middle sub tree < second value in node < value in right sub tree.
- All leaf nodes are at the same level.
- You can notice that the previous example given satisfies all the above mentioned properties.It is important to note from the above properties that all data in a 2-3 tree is stored in a sorted manner which makes search operations fast and effective.

Refer: https://www.google.com/amp/s/iq.opengenus.org/2-3-trees/amp/

B-Tree of Order m has the following properties...

Property #1 - All leaf nodes must be at same level.
Property #2 - All nodes except root must have at least [m/2]-1 keys and maximum of m-1 keys.
Property #3 - All non leaf nodes except root (i.e. all internal nodes) must have at least m/2 children.
Property #4 - If the root node is a non leaf node, then it must have atleast 2 children.
Property #5 - A non leaf node with n-1 keys must have n number of children.
Property #6 - All the key values in a node must be in Ascending Order.

Refer: https://www.youtube.com/watch?v=94ErZ5K8XZg
http://www.btechsmartclass.com/data_structures/b-trees.html

UNIT 4
KMP: http://www.btechsmartclass.com/data_structures/knuth-morris-pratt-algorithm.html