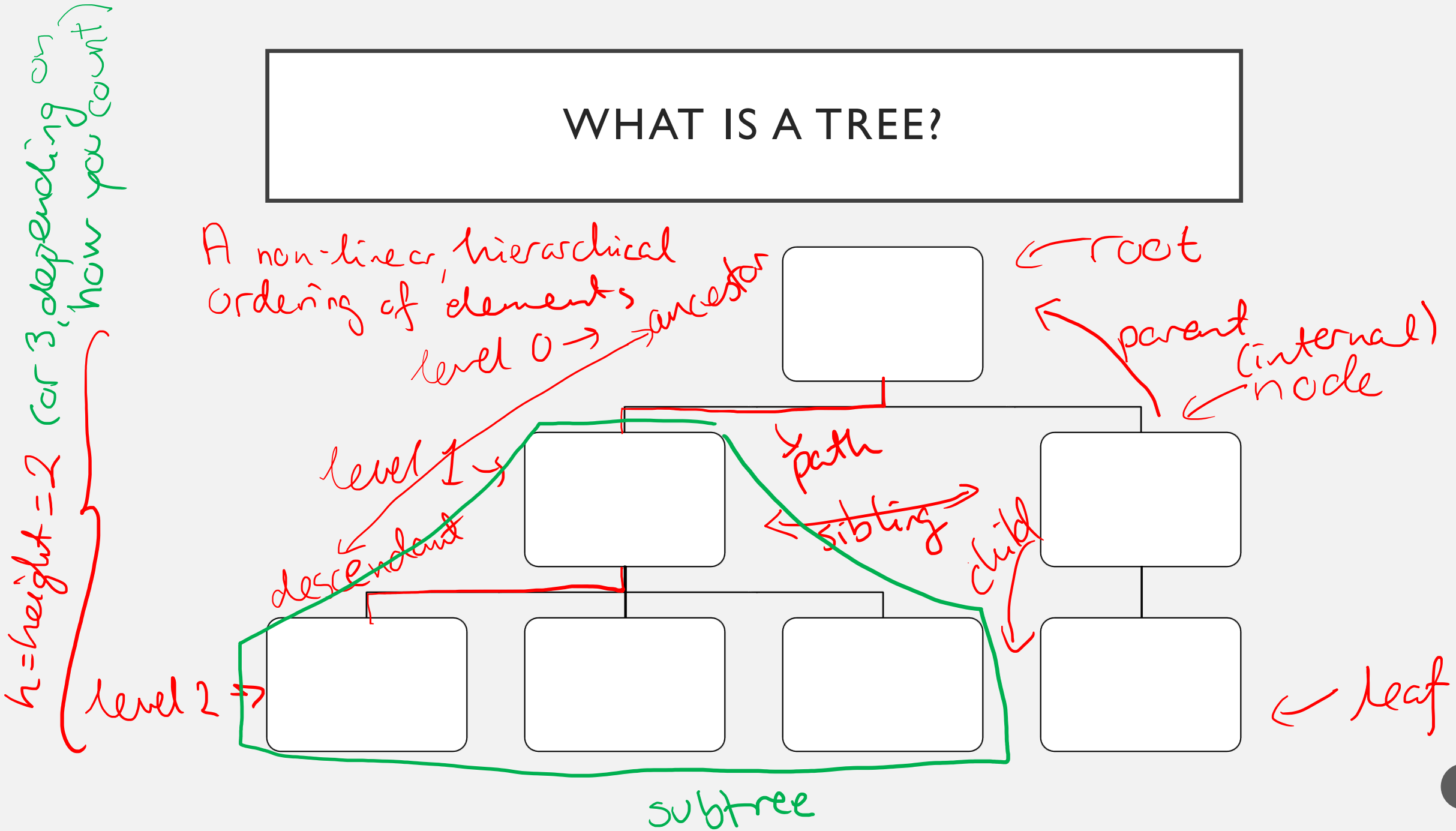
BINARY TREES

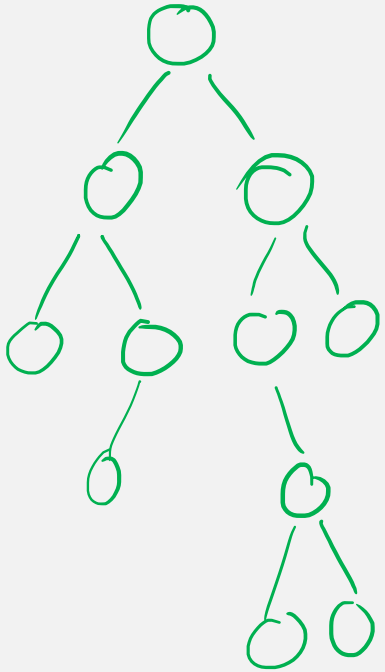
ADS1, S2023

WHAT IS A TREE?

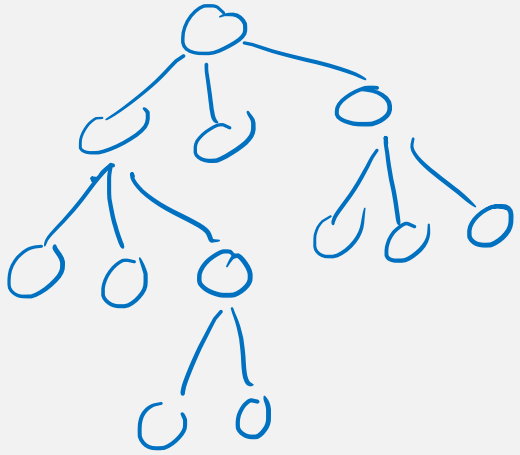


n-ARY TREES

Each node has at most n children



binary tree

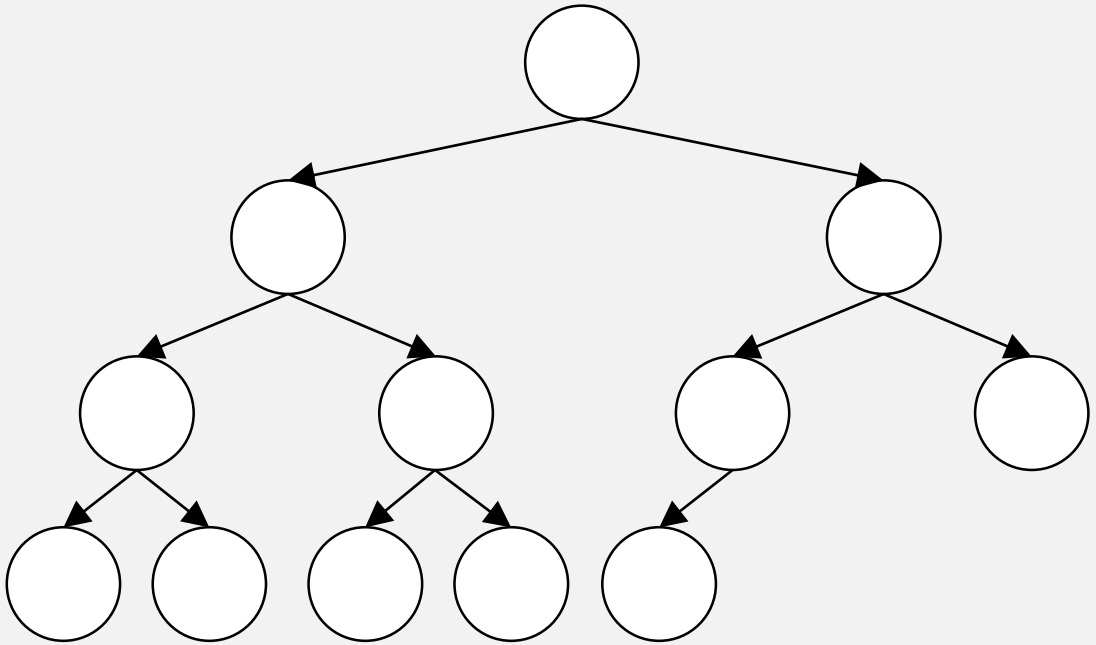


ternary tree

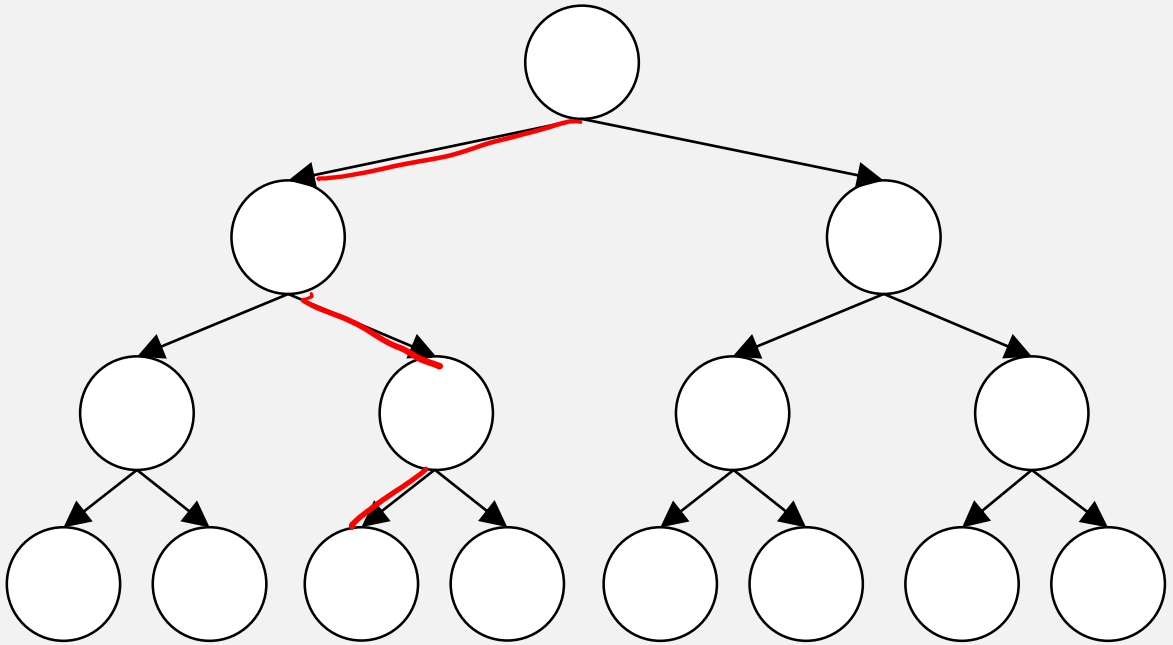
and so on...

COMPLETE AND FULL TREES

BINARY



COMPLETE



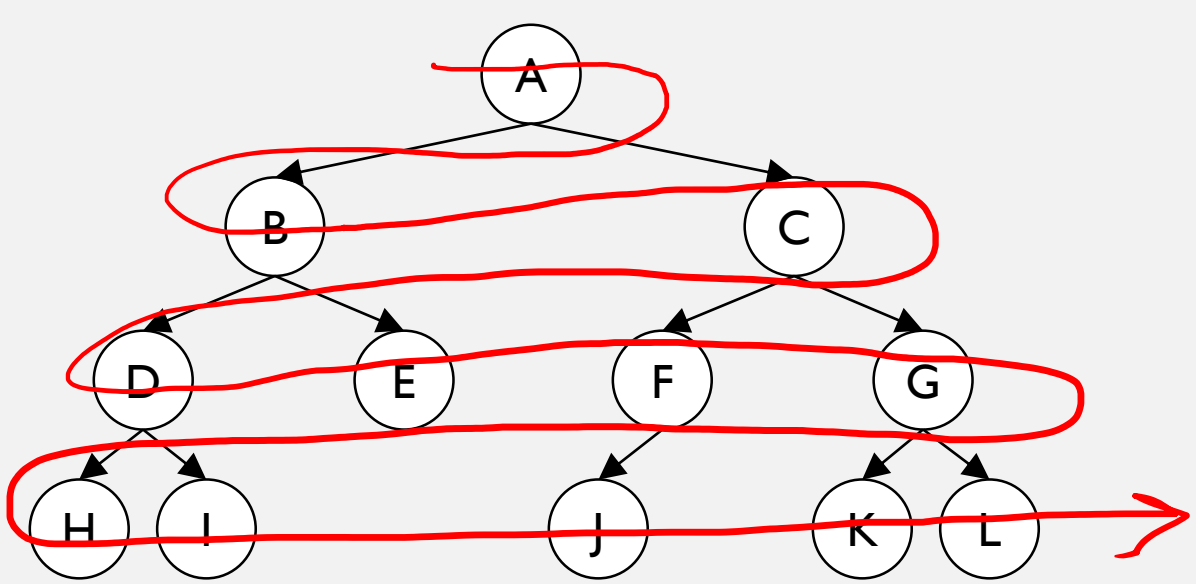
FULL

here, height is "path length"

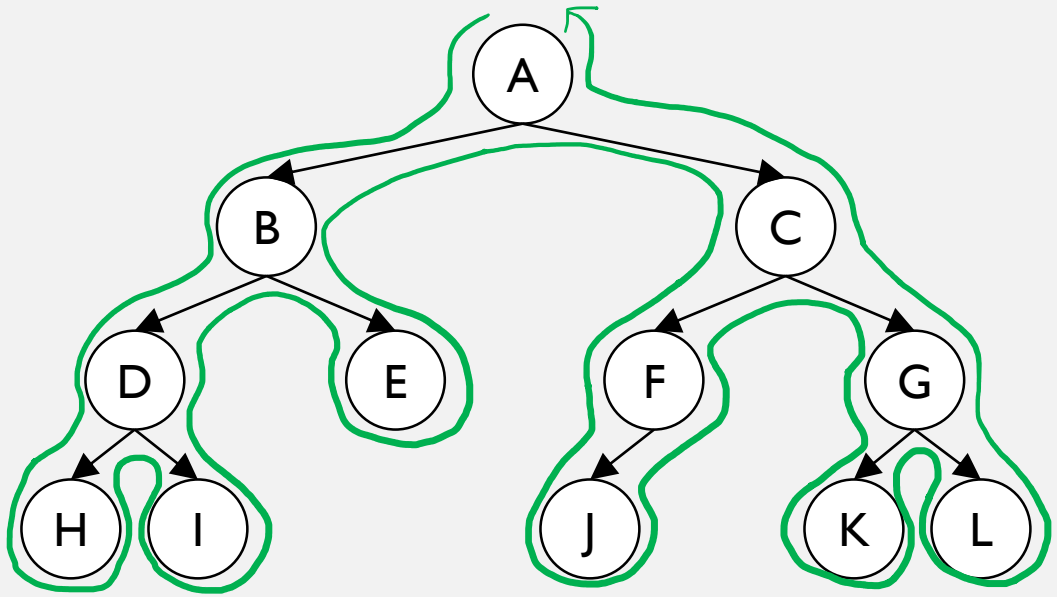
$\#nodes = 2^{h+1} - 1$

TREE TRAVERSAL

Breadth first traversal (BFT)



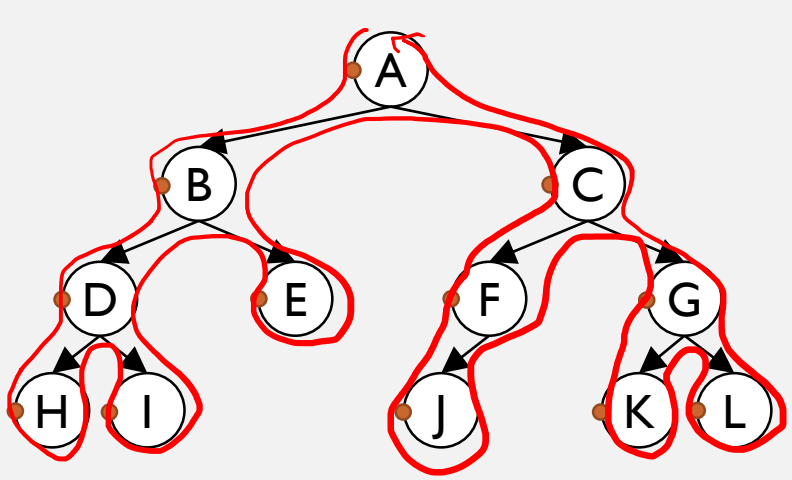
Depth first traversal (DFT)



Traversal order: ABCDEFGHIJKL

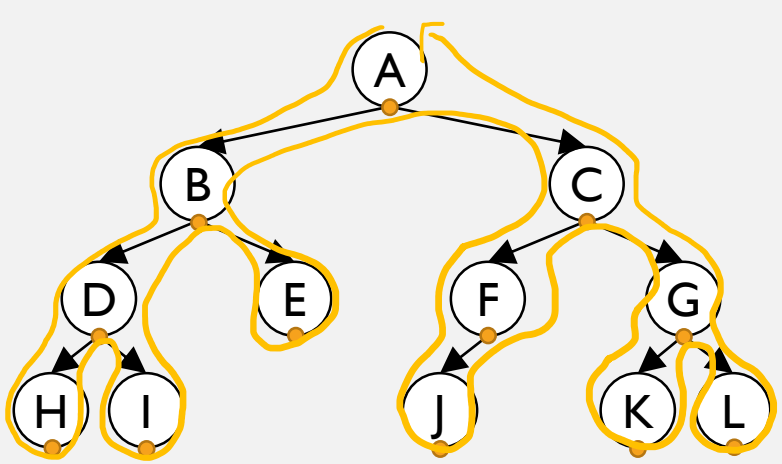
TREE TRAVERSAL

Pre-order DFT



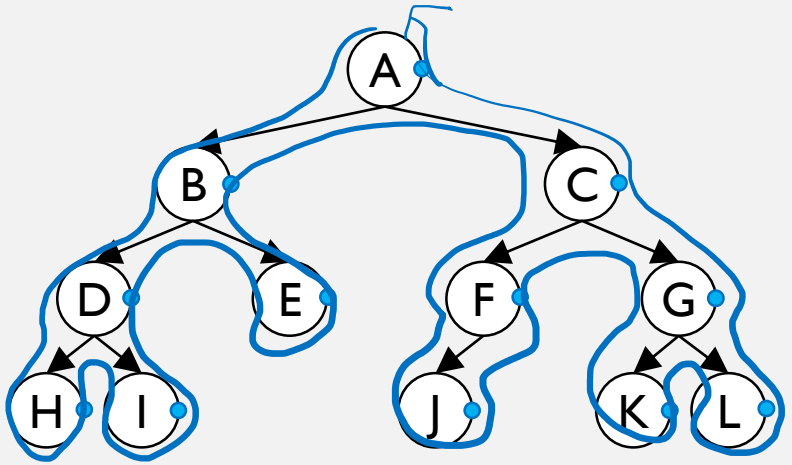
ABDHIECFJGKL

In-order DFT



HDIBEAJFCKGL

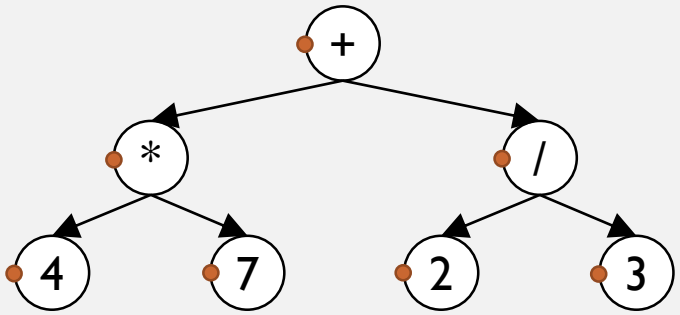
Post-order DFT



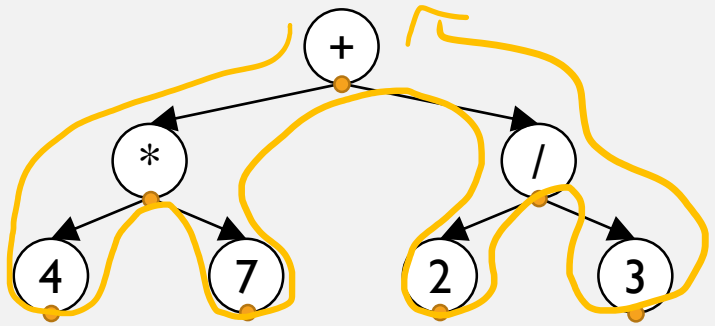
HIDEBJFKLGCA

EXPRESSION TREES

Pre-order DFT

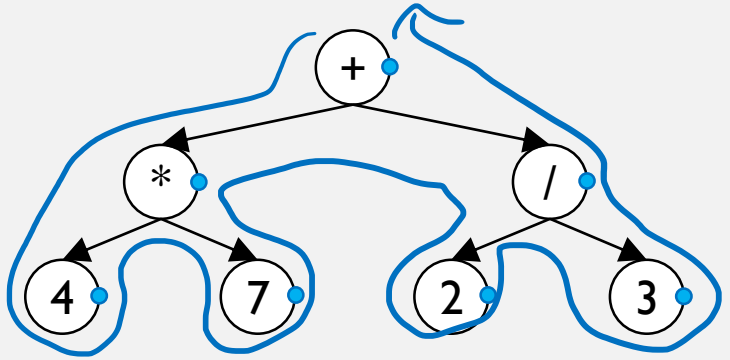


In-order DFT



(4*7) + (2/3)

Post-order DFT



47*23/+

postfix notation

THE (BINARY) TREE ADT

- getRoot
- isEmpty
- size
- contains
- inOrder
- preOrder
- postOrder
- levelOrder → BFT
- height

THE NODE ADT

- setElement
- getElement
- addLeftChild
- addRightChild
- getLeftChild
- getRightChild

REPRESENTING A BINARY TREE

"linked" binary tree

Expand the idea of a linked list

~~next~~ right left parent

"Array-like" binary tree

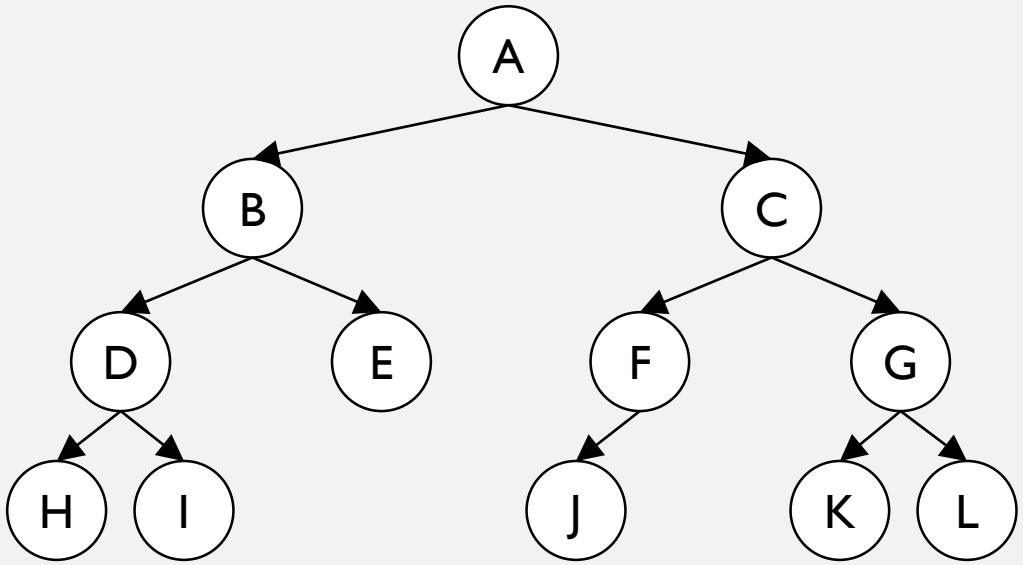
use an arraylist

→ not obvious

if we don't know if c=l or r

"ARRAY-LIKE" REPRESENTATION

$p = \left\lfloor \frac{c-1}{2} \right\rfloor$



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	B	C	D	E	F	G	H	I	•	•	J	•	K	L

parent p children
1 3 4

parent
6

children
13 14

danger: wasting space

In general, parent p:

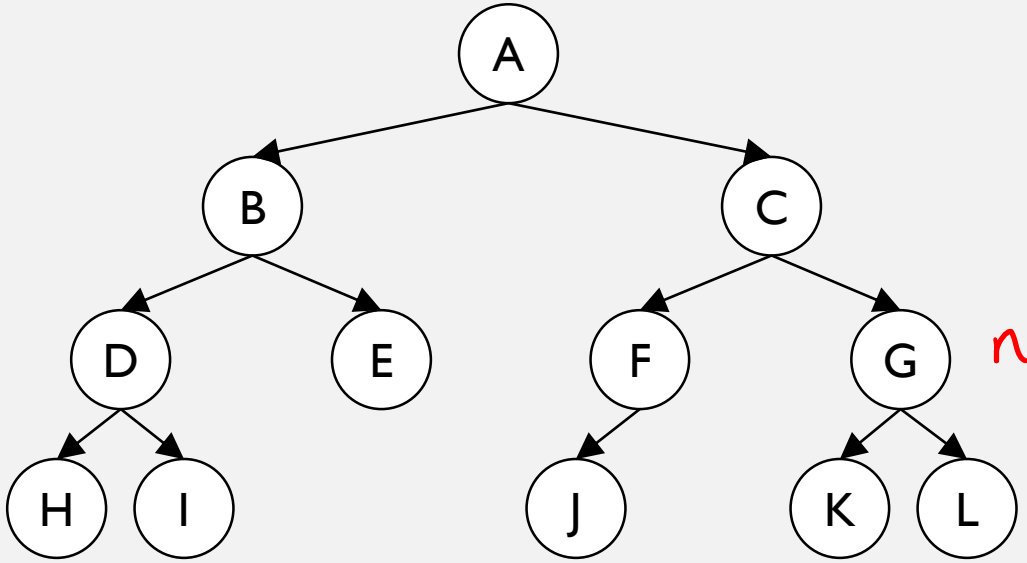
left child $l = 2p+1$

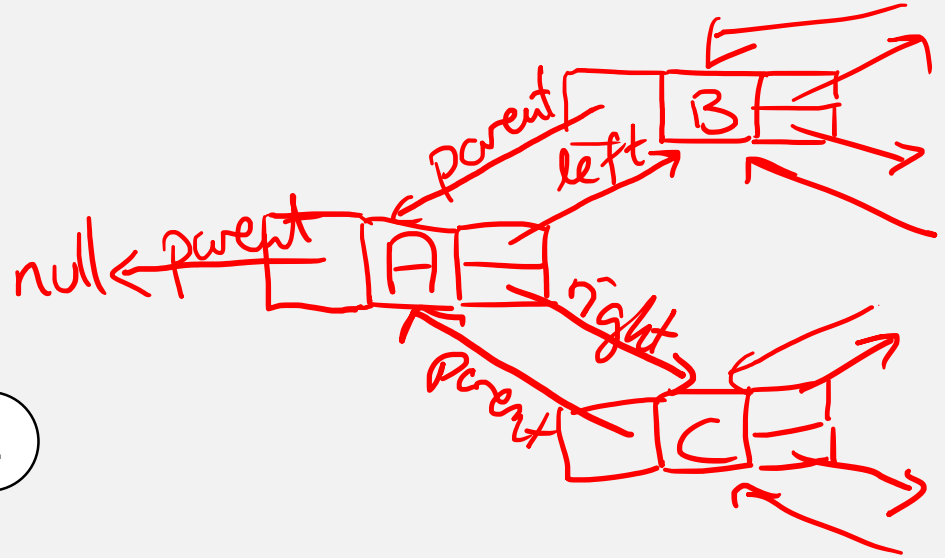
right child $r = 2p+2$

$p = \frac{r}{2} - 1 = \frac{l-1}{2}$

"computed child links"

"LINKED" REPRESENTATION



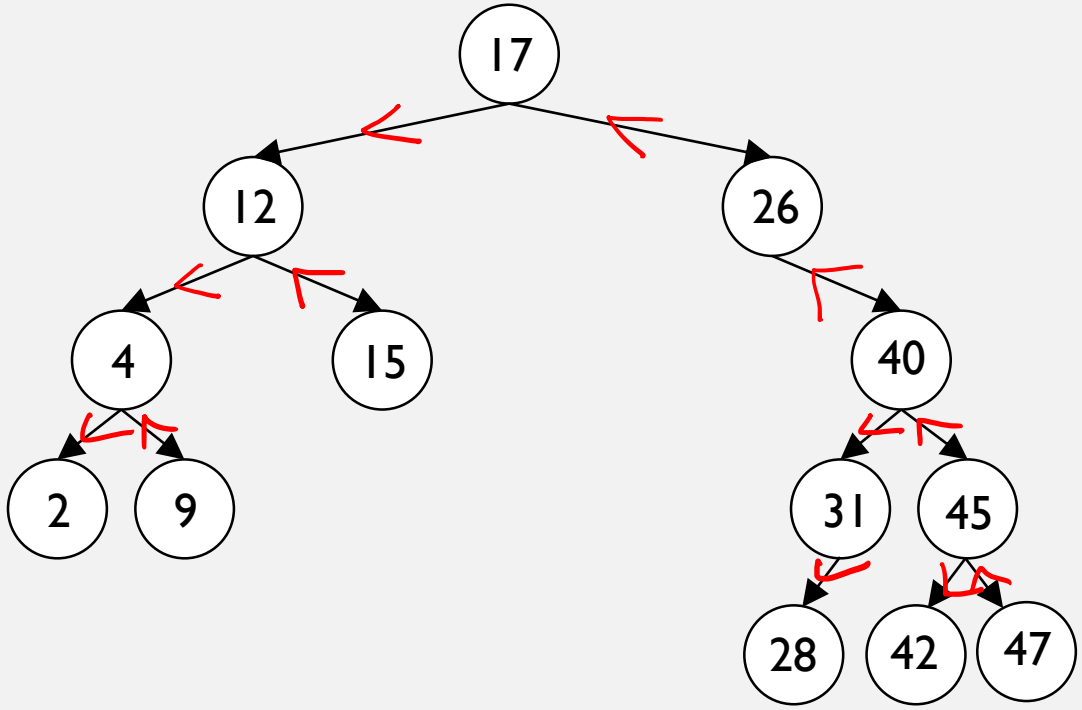


and so on...

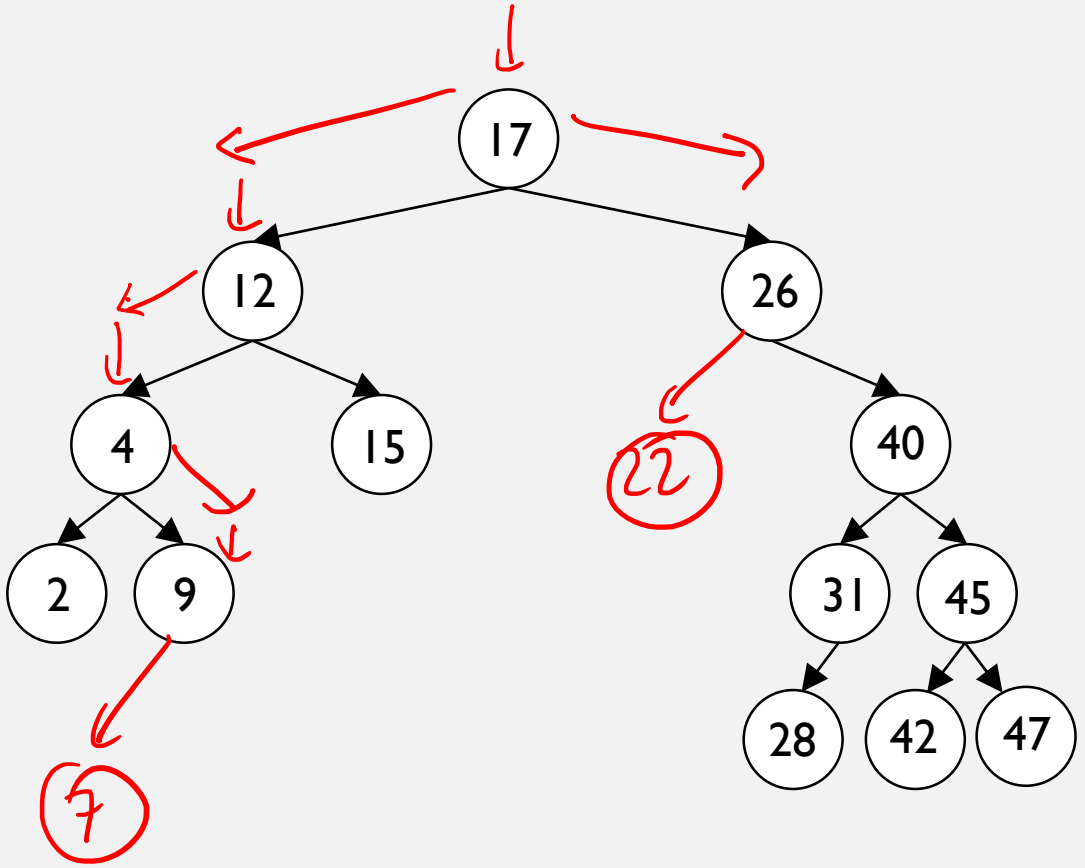
LinkedBinaryTree in Java

BINARY SEARCH TREES

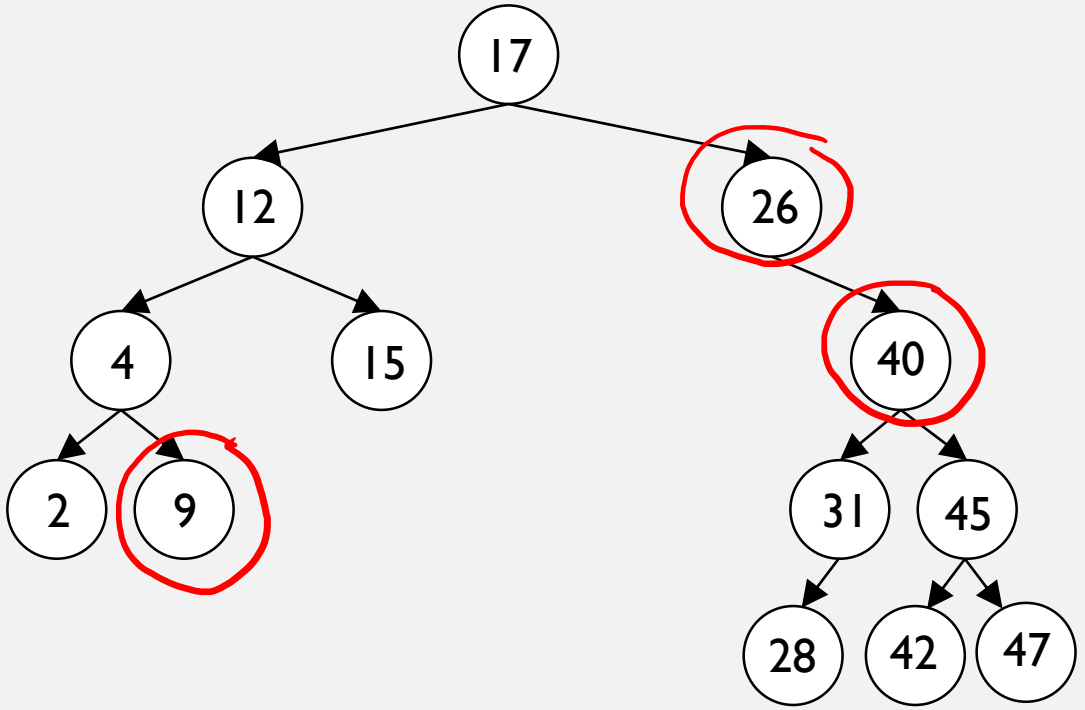
For every node n:
Each element in left subtree < element in n and each element in right subtree > element in n.



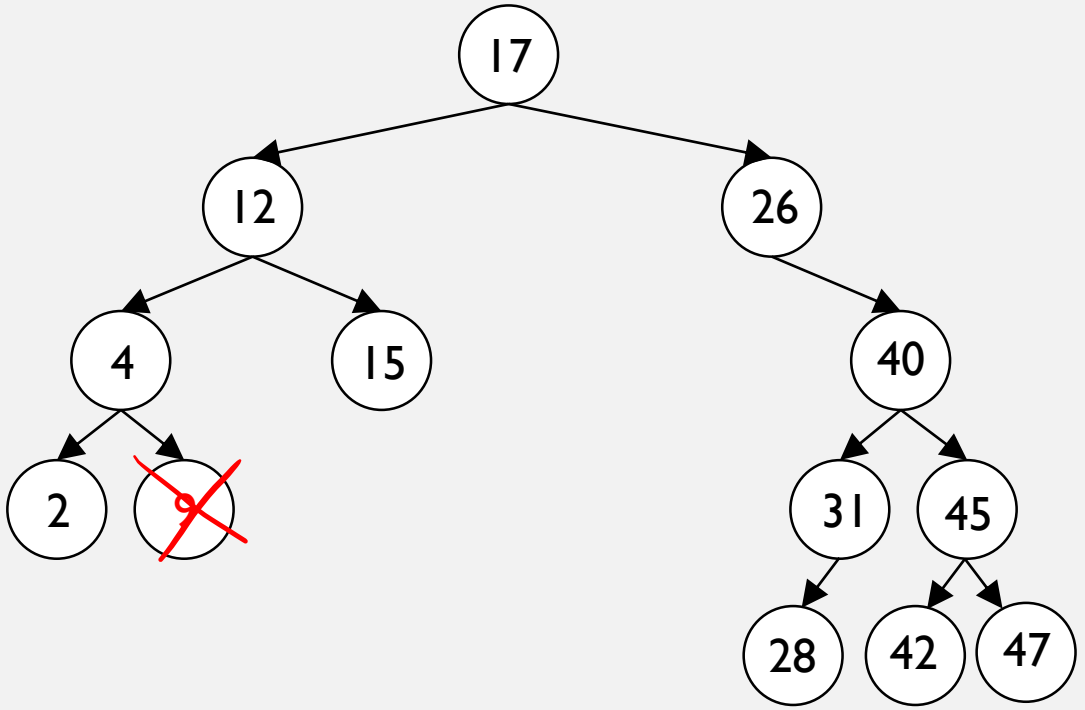
ADDING TO A BINARY SEARCH TREE



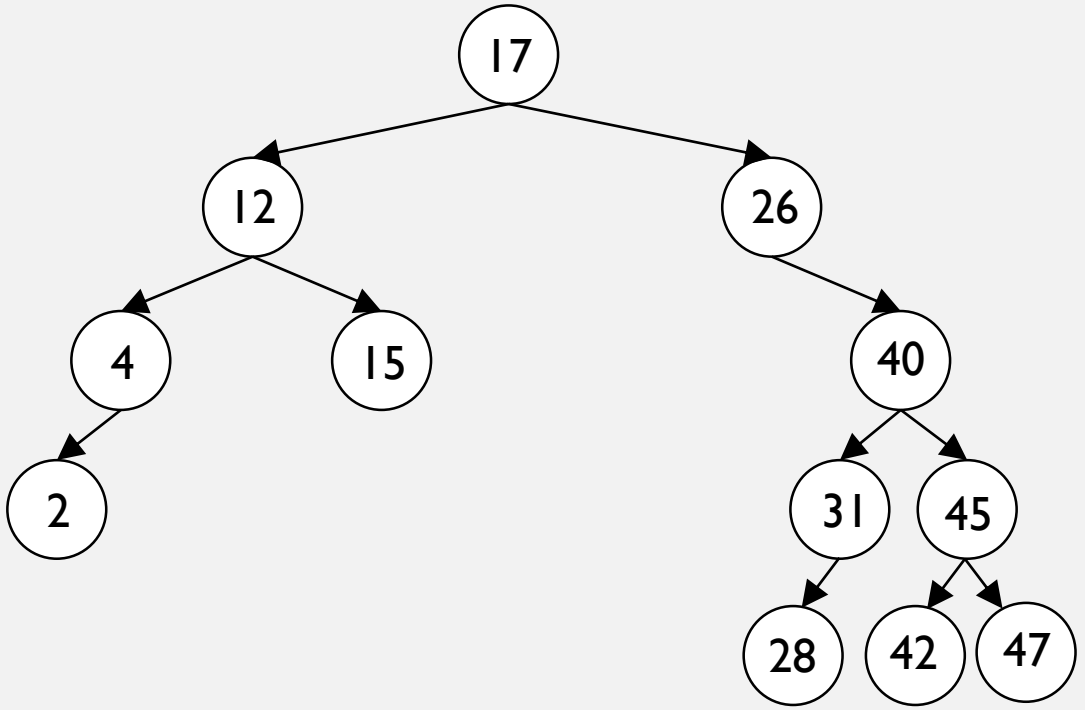
DELETING FROM A BINARY SEARCH TREE



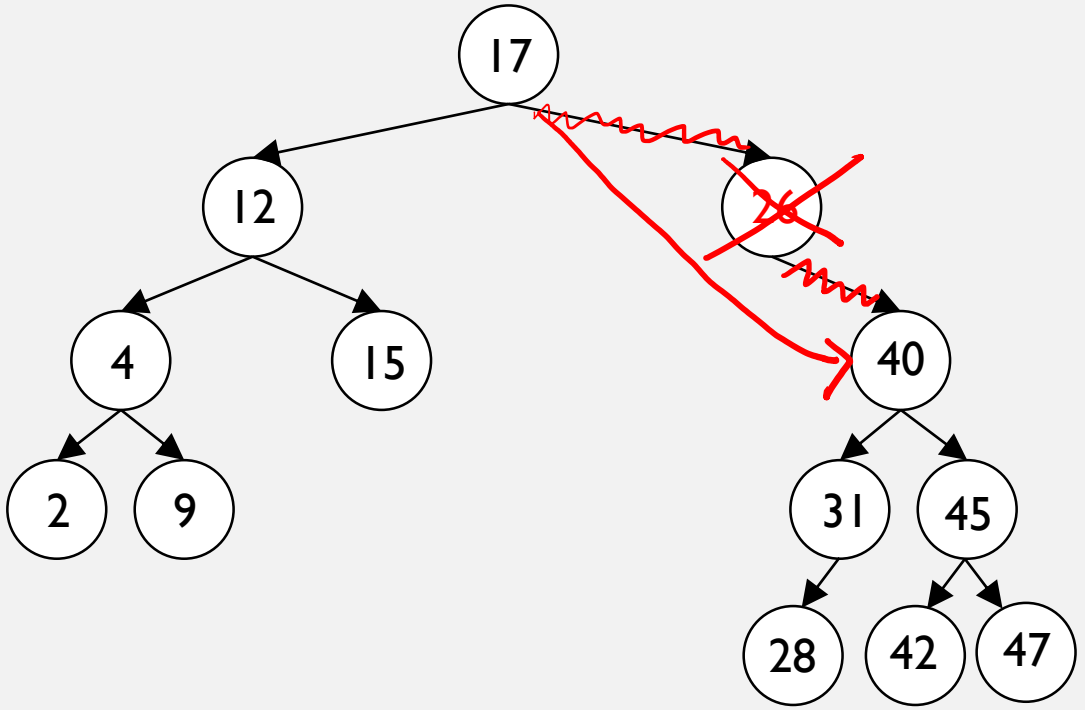
CASE 1



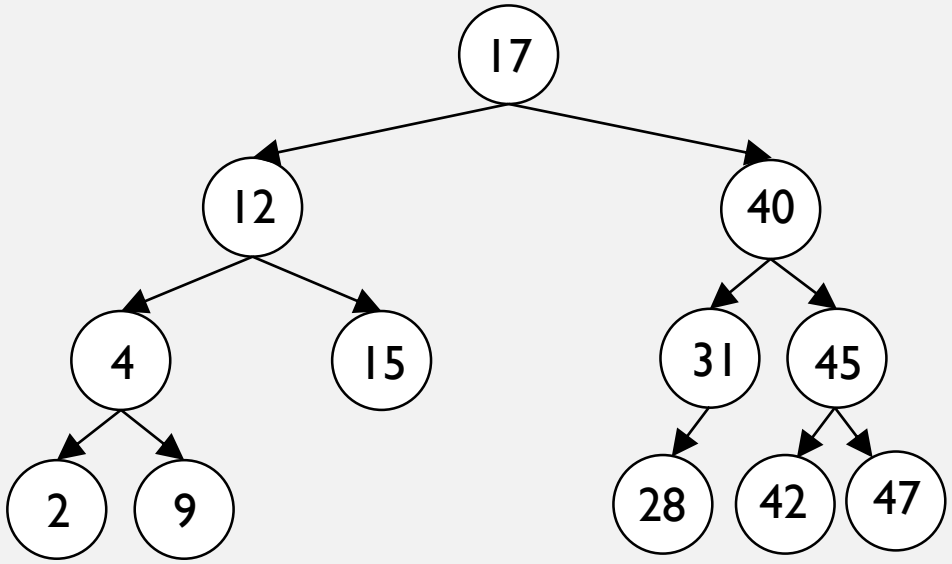
CASE 1



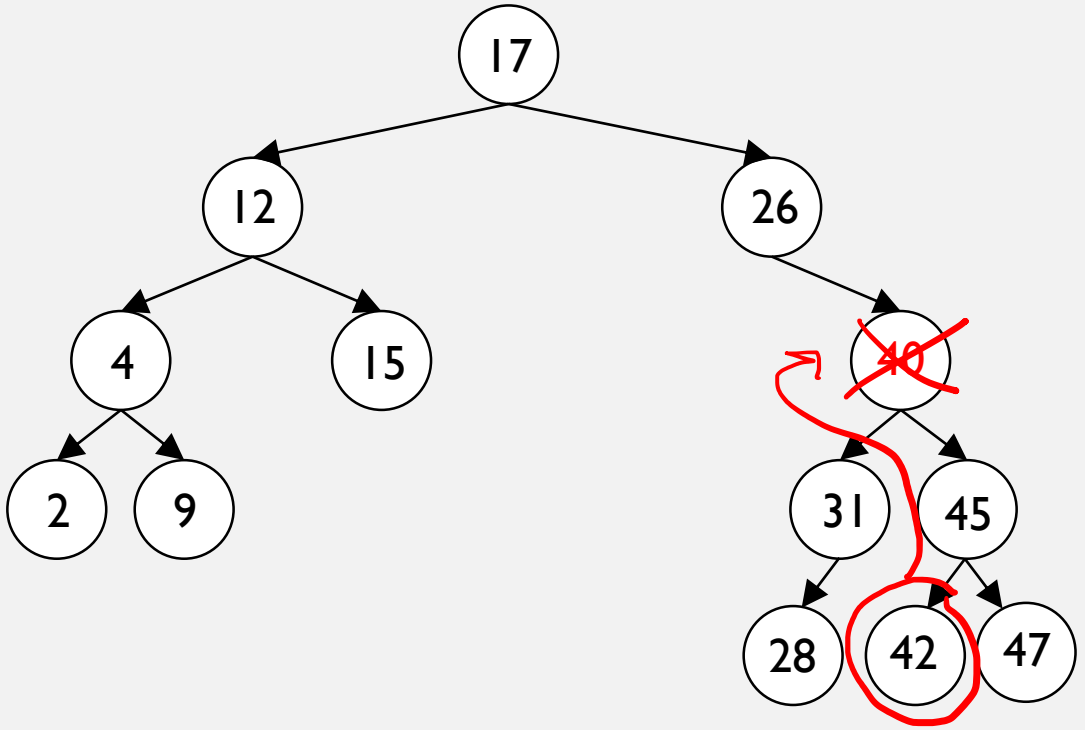
CASE II



CASE II

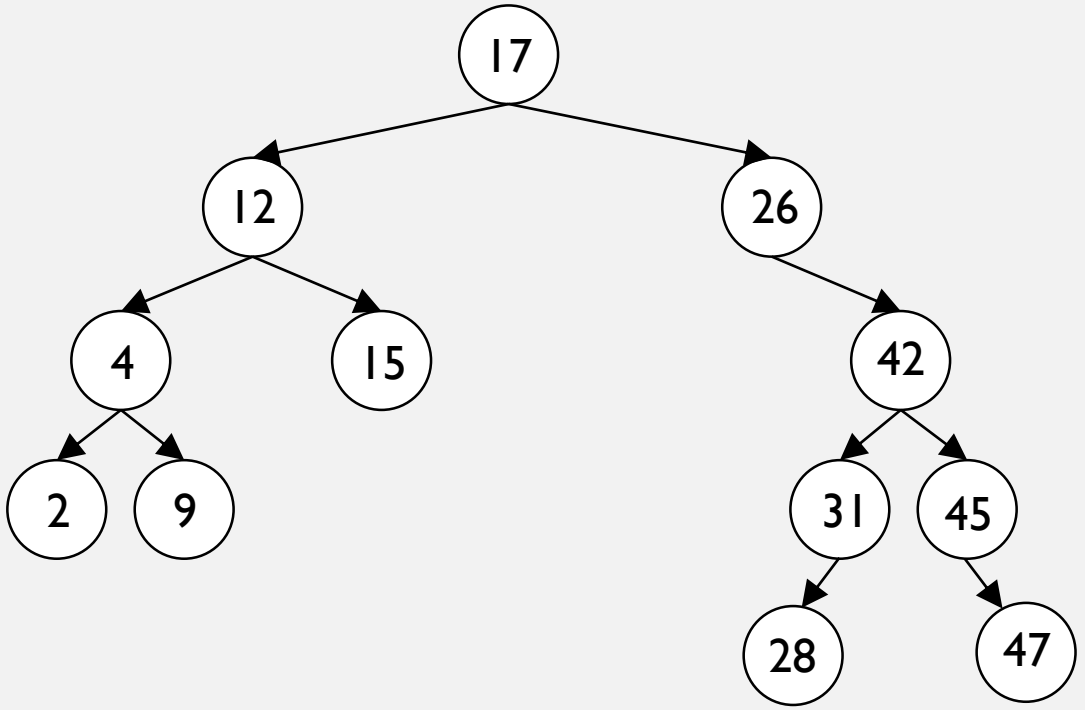


CASE III



leftmost element in right subtree

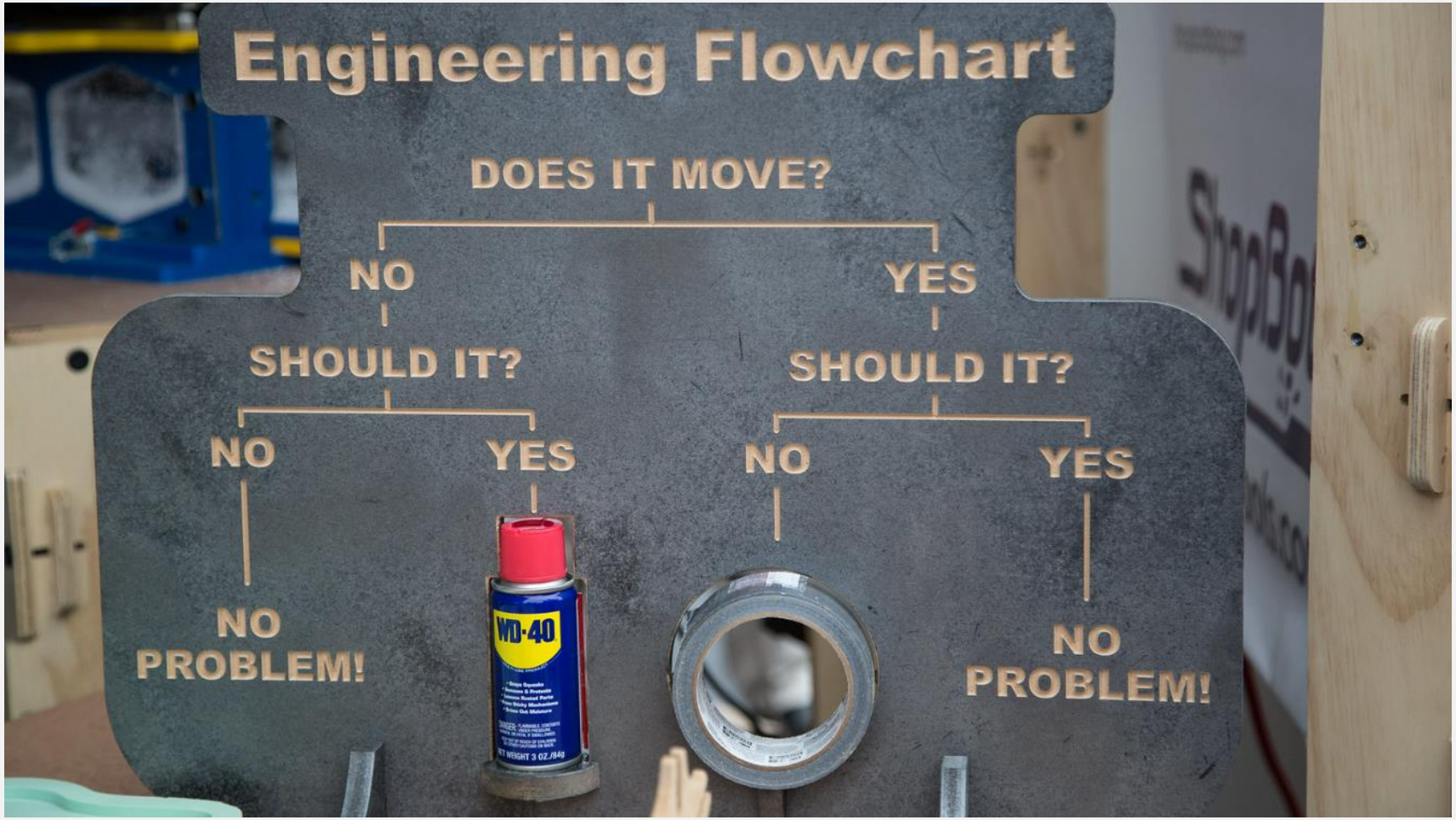
CASE III



SETS AND MAPS IMPLEMENTED AS BINARY SEARCH TREES

Adding
Deleting } all $O(\log n)$
Access

APPLICATIONS OF TREES



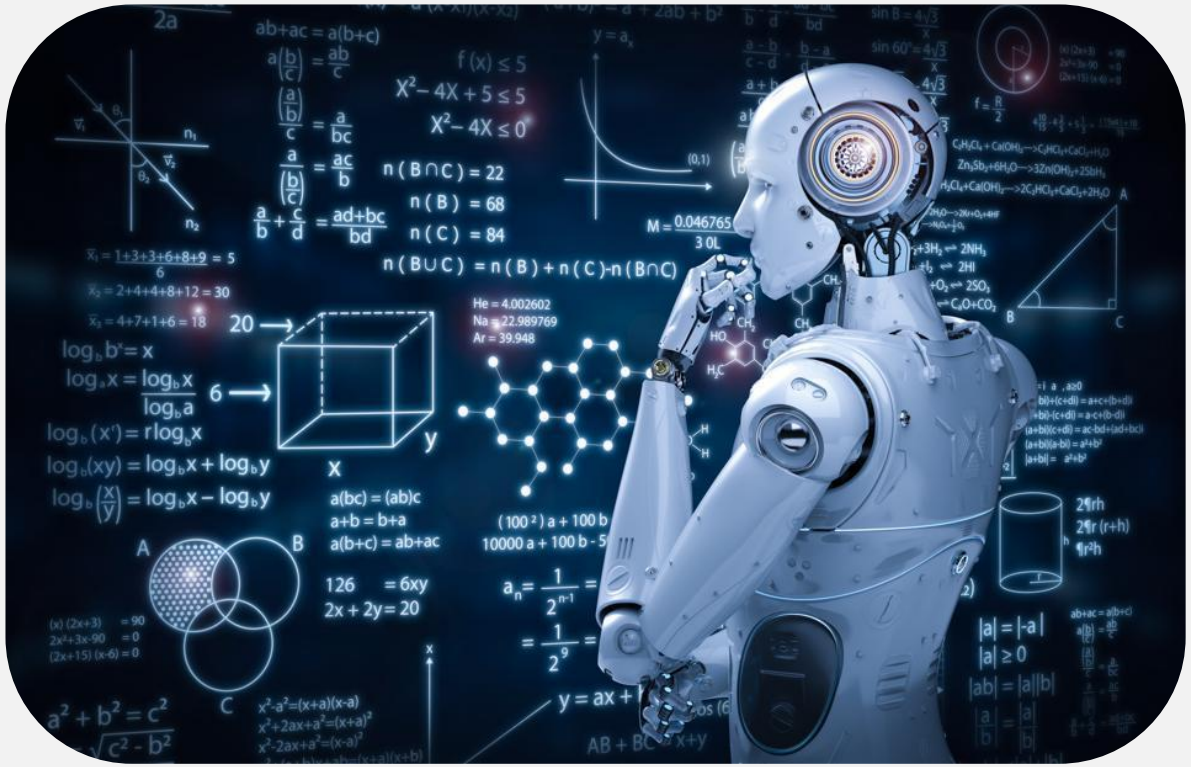
DECISION TREES

A machine learning model
→ Classify things

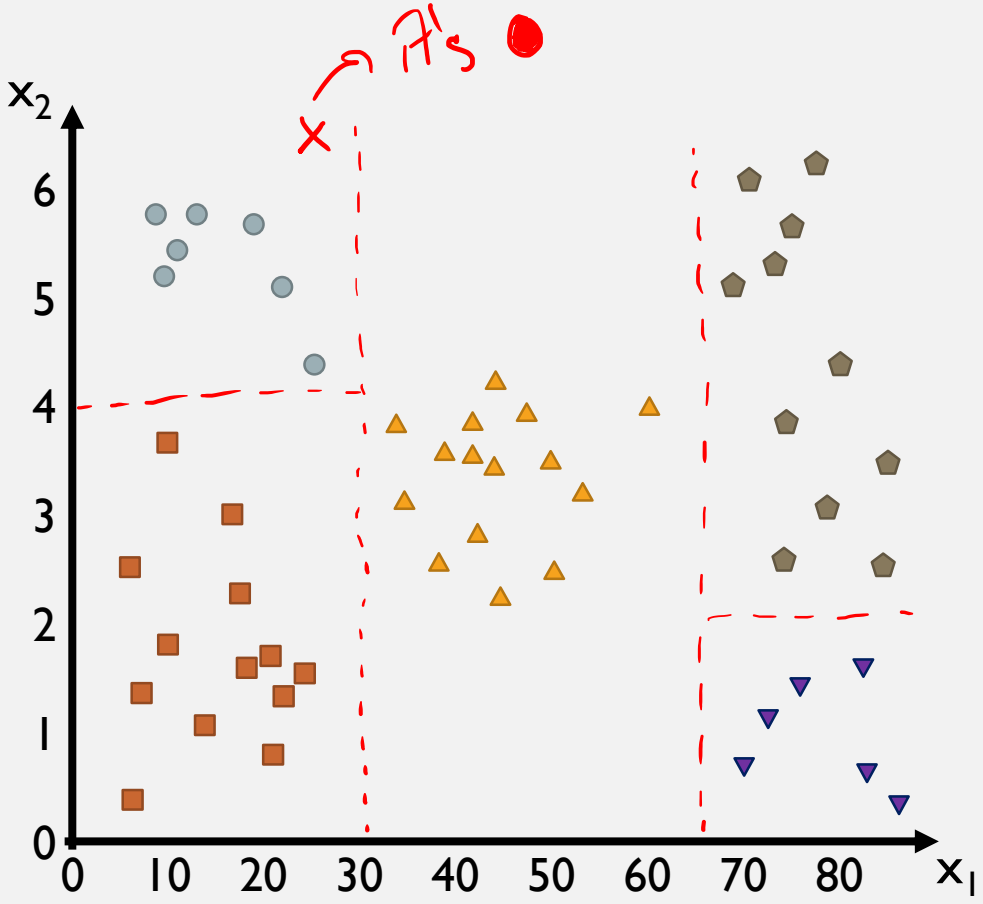
images of traffic signs
(self-driving cars)

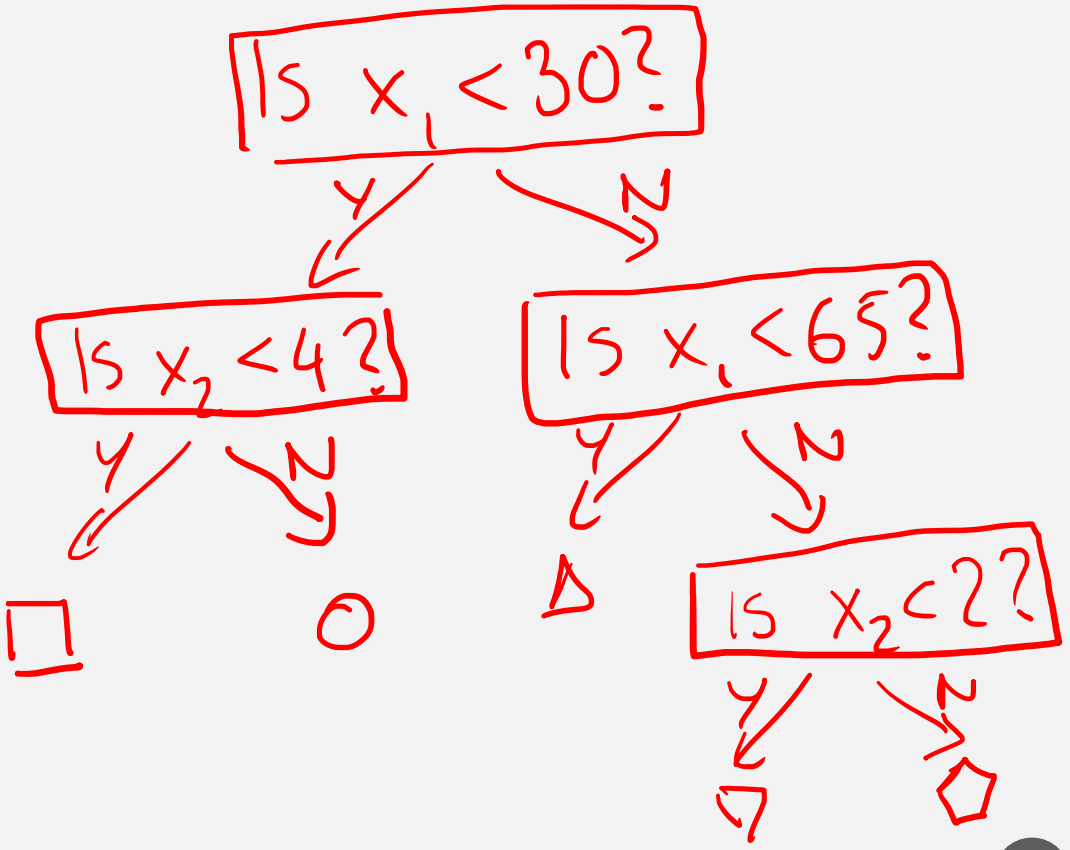
mushroom edible?
(make an app)

recognizing diseases
(tumor malignant or benign)



DECISION TREES





HUFFMAN CODING

Efficient lossless data compression

HUFFMAN CODING

AN_ANT_ATE_A_BANANA

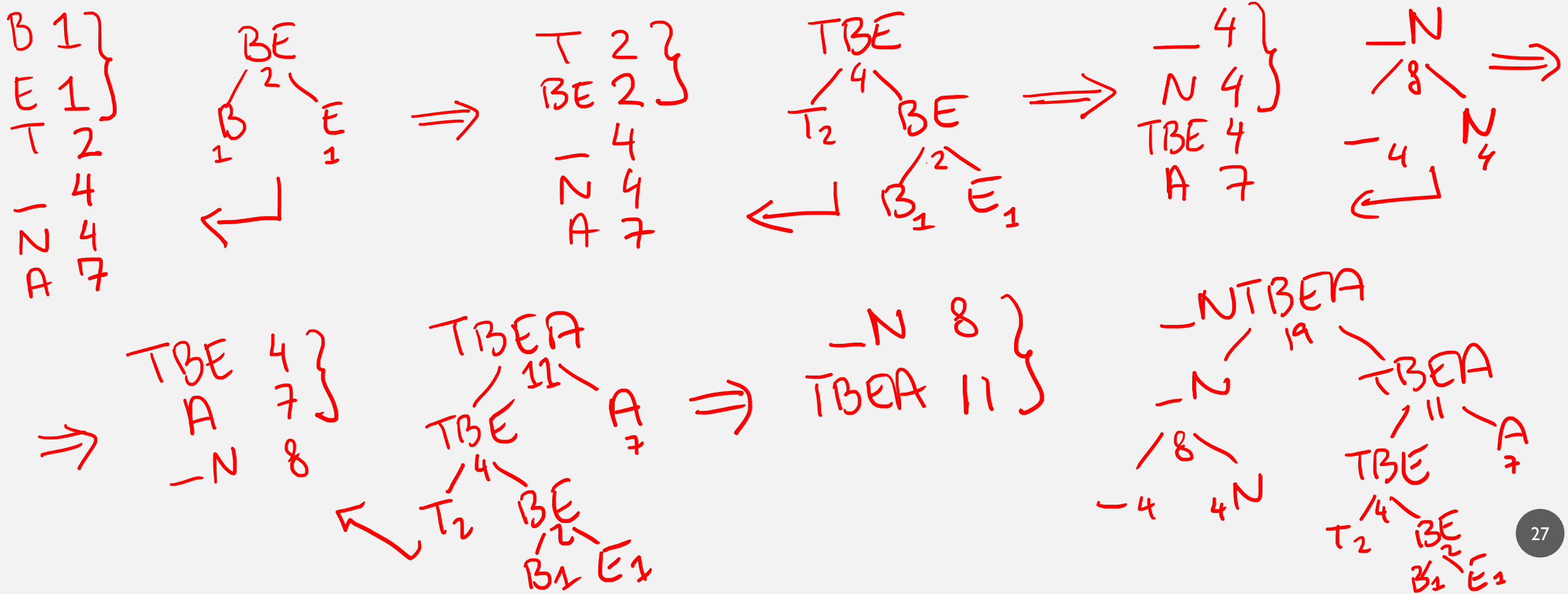
Symbol	Binary code
A	01000001
B	01000010
E	01000101
N	01001110
T	01010100
_	01011111

```
010000010100111001011111010000010100111001011111
010111110100000101010100010001010101111101000001
010111110100001001000001010011100100000101001110
01000001
```

152 bits

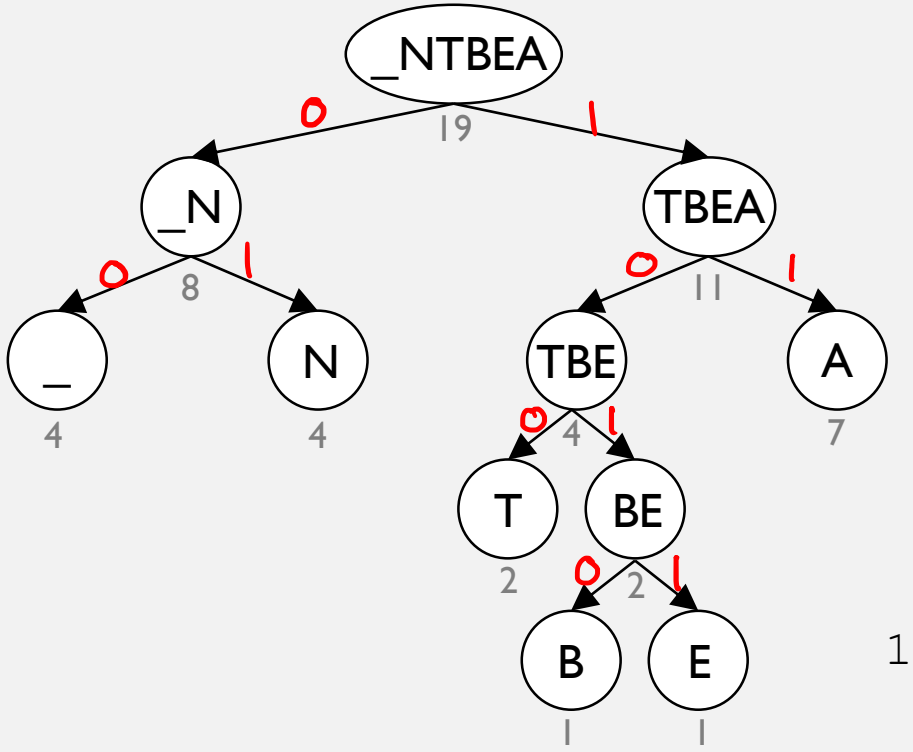
HUFFMAN CODING

AN_ANT_ATE_A_BANANA



HUFFMAN CODING

AN_ANT_ATE_A_BANANA



Symbol	Huffman code
A	11
B	1010
E	1011
N	01
T	100
_	00

11010011011000011100101100110010101101110111

44 bits

EVERYTHING RUNS IN LOGARITHMIC TIME … IF …

the tree is balanced.