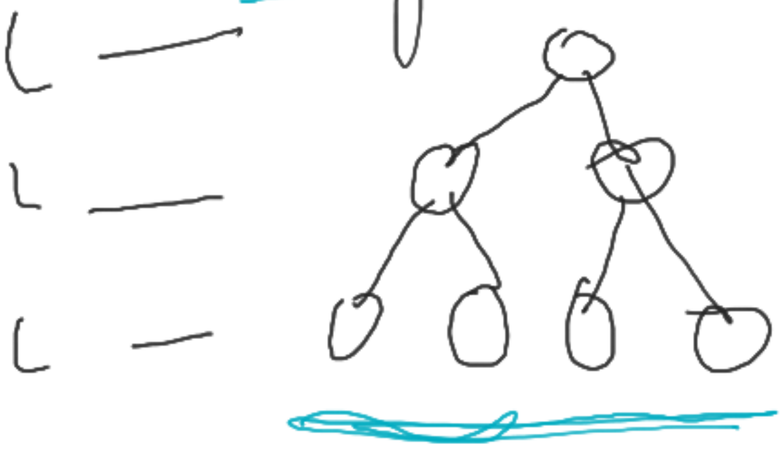
n 7

Binary Search Tree

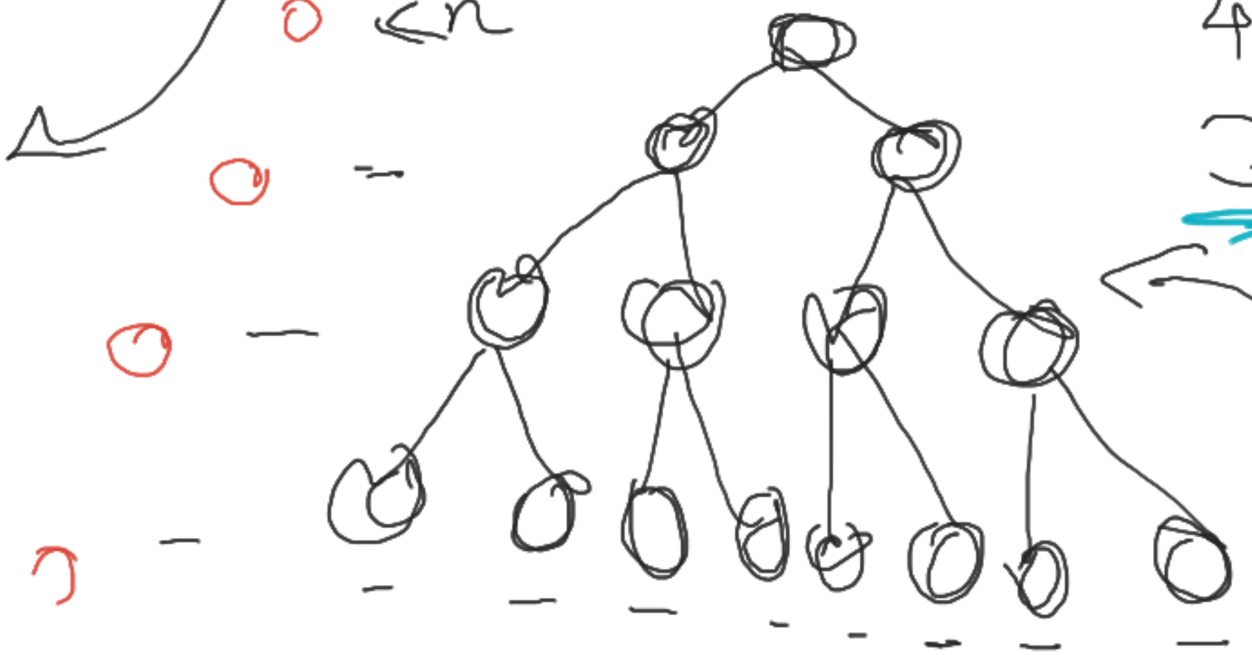


Total
3 levels

2 Height

Leaves 4

B+ Tree



4 levels

3 Height

Assuming $2^3 \Rightarrow 8$

n = 8

Leaves 8

Merge Sort

$cn\left(\log_2 8 + 1\right)$

$cn\left(\log_2 2^3 + 1\right)$

$(cn(3+1)$

$(cn(4)$

We analysed B+ Tree

Because, merge sort example, all data is at leaf level. (Bare cases)

$$cn\left(\underbrace{\log_2 n}_{\text{no. of heights}} + \underbrace{1}_{\text{root}}\right)$$

$\rightarrow$ no. of levels

$$\Rightarrow cn\log_2 n + cn$$

$$\Rightarrow n\lg n \qquad \Theta(n\lg n)$$

Time analysis of Binary Search method

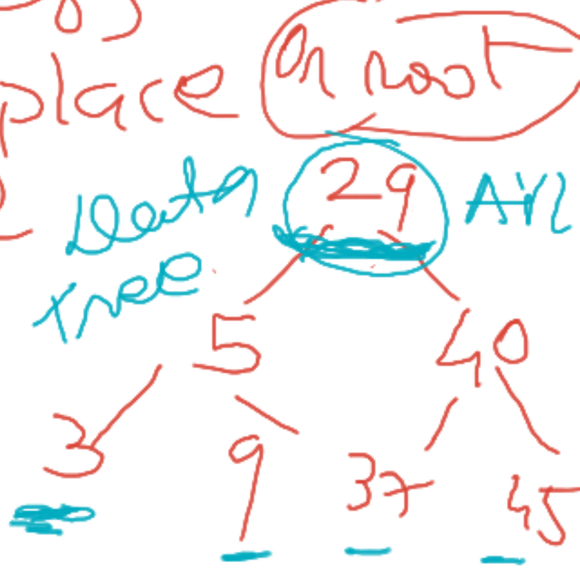
(Searching Technique)

(Prerequisite: Data has to be sorted)

If the data store is an array

(Assumption: access to ith element is of O(1) ⇒ Direct access)

Best Case: Key present at first place (root)



Data tree

AVL

In binary search

- mid = (low + high) / 2

if ____ <

bs(low, mid-1, key)

else if >

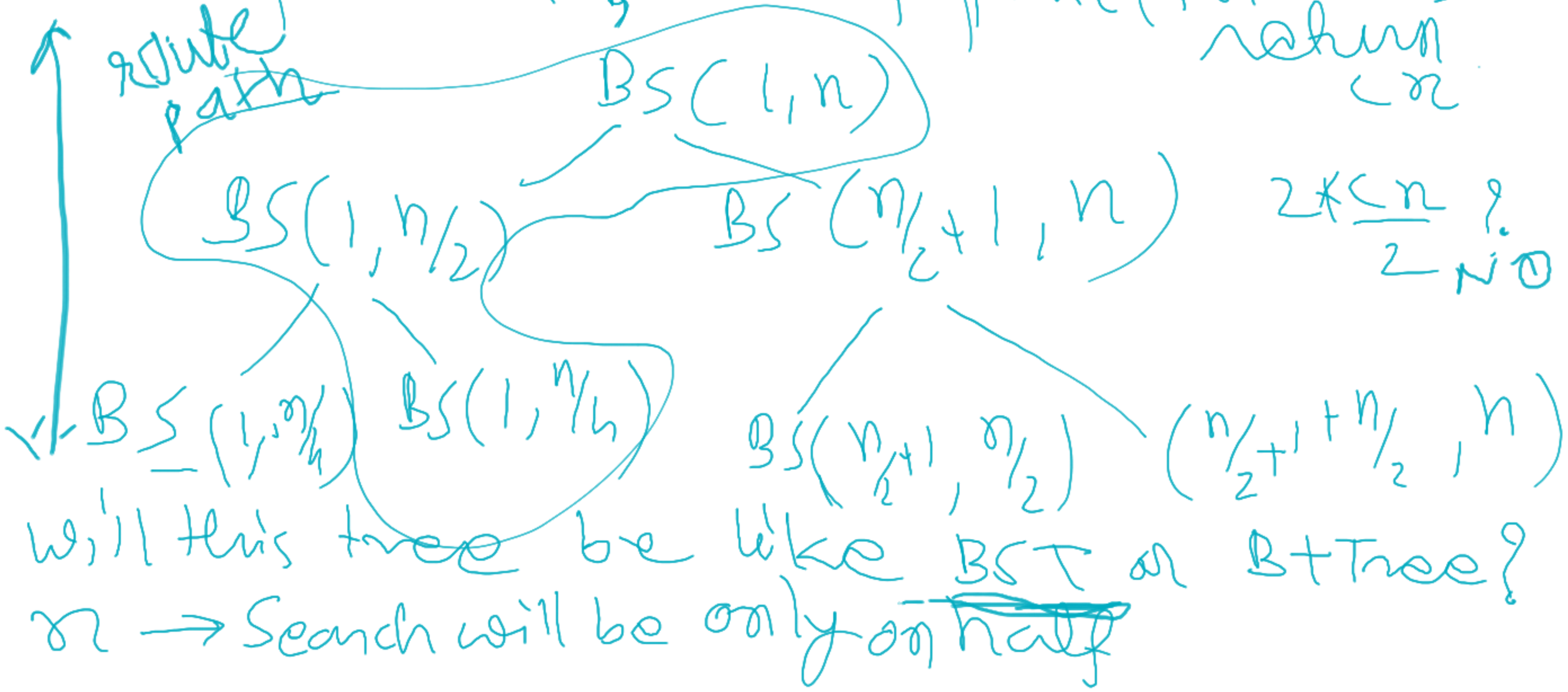
bs(mid+1, high, key)

else

return mid

return -1 // (not found)

Tree which got created is actually not of data, but of function calls



return x

Will this tree be like BST or B+Tree?

x → Search will be only on half

Binary search tuples
order of $\log n$ time analysis

What is the number of height - for complete BST having n nodes?

$\log n$ $\log_2 8$ $\log_2 2^3 \Rightarrow 3$

4 level • 3 Height

Path is nothing but
a height
length

What will be the worst case for searching?

(1) Data found at end (linearSearch)
(leaf level) (partition size 1)

(2) Data not found (linear + BST)

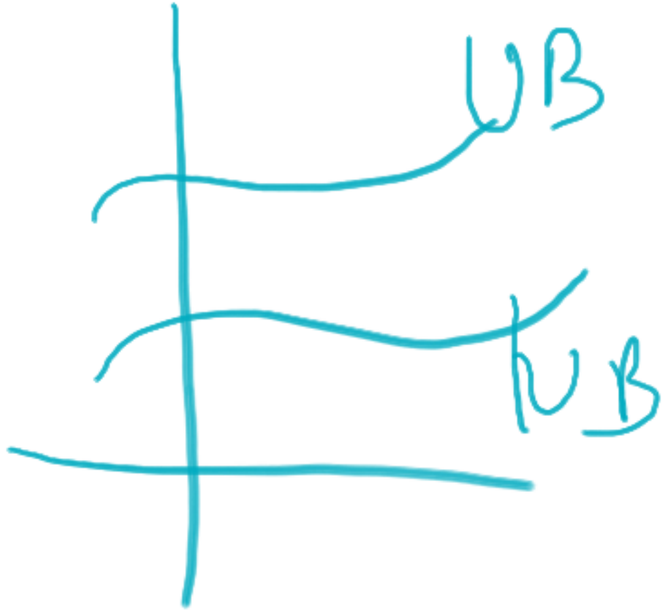
Big O

Big Θ

Big Ω

little o

little ω



Assignment:

Describe a $\Theta(n \lg n)$ algorithm that given a set S of n integers and another integer x, determines whether or not there exist two elements in S, sum is exactly x.