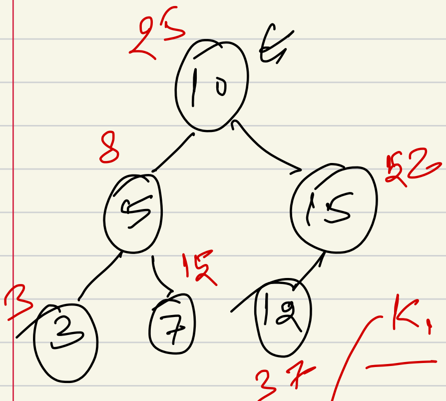
BST T   n nodes

height - $\frac{n}{2}$ → $O(n)$



$\Omega(\log n)$ $O(h)$

K_1 — ... K_n n keys (sorted order)

Prefix sum

$P_i = \sum_{j=1}^{i} K_j$

$P_1 = K_1$ $P_2 = K_1 + K_2$ $P_n = \sum_{i=1}^{n} K_i$

inorder traversal

$[K_1 \dots K_n]$

$[P_1 \dots P_n]$

Can you give an algo that gives space comp. $O(h)$

Replace keys in T with ~~their~~ Prefix sums

Should still remain BST

Time - $O(n)$ Space - $O(n)$

sum = 0
PrefixSum(root)

```
prefixSum(BST T)
{
    if (T == NULL) return;
    PrefixSum(T->L)
    Sum += T->key;    T->key = sum;
    PrefixSum(T->R)
}
```

O(h) - space
O(n) - time

Suppose you're given a seq. of integers

$a_1, \dots, a_n$

Insert these in a BST (in this order)

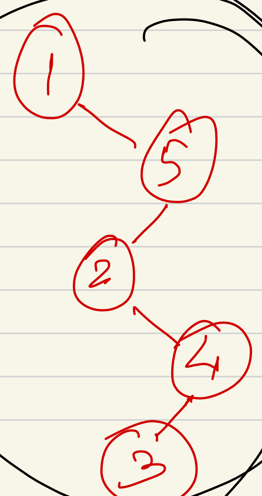
What is the max. height of the BST?

$n-1$

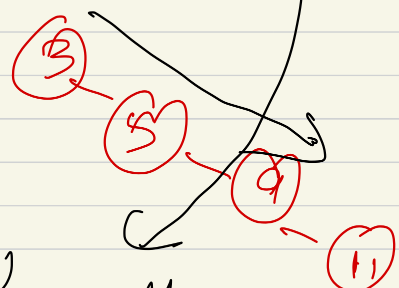
When? → Need not be sorted

You're to tell whether inserting these elements in this order will lead to a BST of the worst height $(n-1)$?

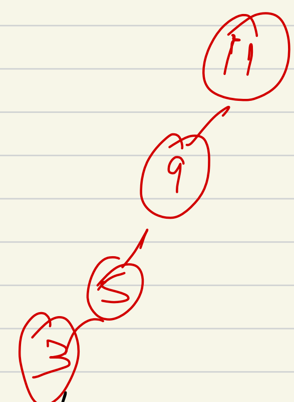
3 5 9 11



→ $O(n)$



You can't store the elements



→ Construct BST & find the height

Time Complexity →

$O(n \log n)$

$O(n^2)$ $O(n^2)$

Insertion - $O(h)$ Ch

$c + 2c + 3c + \dots nc$

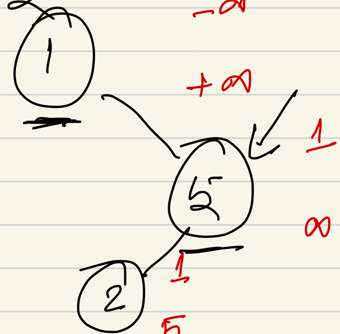
$= O(n^2)$

1 5 2 4 3

1 2 n

let's assume distinct

BST



lower_limit $= -\infty$

upper_limit $= +\infty$

$n \le 2$ return true;

for $i = 2$ -- n

if ($a_i <$ lower_limit) or ($a_i >$ upper_limit)

return false;

-3

if ($a_i > a_{i-1}$)

lower_limit $= a_{i-1}$

if ($a_i < a_{i-1}$)

upper_limit $= a_{i-1}$

return true;

Why do we need BSTs?

Keys

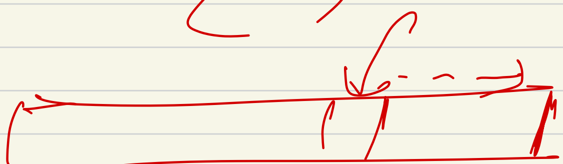
English { dictionary entries }

Set of words + some new words

↓ store these in a DS

search for a word

'apple' 'mango' 'moon'



Sorted array

bin-search

O(n)
insertion

BSTs

Balanced

O(n)
↓
O(log n)

List all the words starting from mon

mon*

monday

Find successors

single

$O(h)$

$t \cdot h$

$O(t \cdot \log n)$

Suppose these are t such words

mon

$\text{mon} \le K < \text{moo}$

$O(n)$

$O(n)$

inorder traversal

Modify BS operation

K_1 K_2

Key(v)

Key(v) $< K_1$: Search the right s.t. of v

$K_1 \le$ Key(v) $< K_2$: print(v);

recursively search both children

Key(v) $> K_2$: search the left s.t.

Complexity?

$\underline{h}$ $\underline{t}$ $\underline{n}$

$\underline{O(h+t)}$

$O(h) + t$

$t + 2h$

$O(h+t)$

$O(h+t)$

$\sum |S_i| + 1$

go to $S_i \subset (T_1, T_2)$ right left

$O(h)$

$O(h)$

From all nodes in T_1 → right subtrees

T_2 → left subtrees

all elements are in my answer

$K_1 = 30$

$K_2 = 80$

$K_1 \le K \le K_2$

all right subtree of nodes in T_1 +

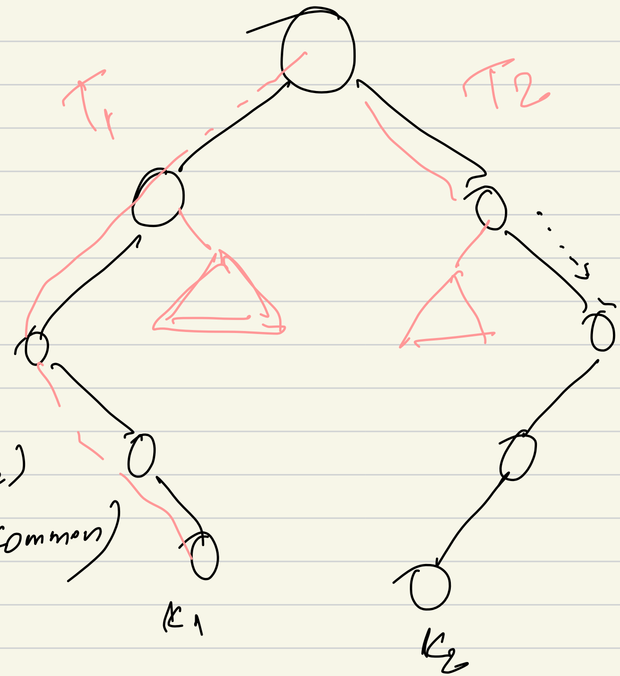
all left st of nodes in T_2

all elements in my answer

$2 * O(h)$

$+$

$$\sum_{S_i \in (r(T_1),\, l(T_2) \text{ - common})} \underline{|S_i| + 1}$$



$$= \frac{\sum |S_i|}{t} + \frac{\sum 1}{2h}$$

$O(t + h)$

Rotations

$\frac{m}{n}$

Two sorted linked lists

Merge these into a single sorted linked list

$O(1)$ space?

Two BSTs

m keys n keys

Merge these BSTs into a single BST in $O(1)$ space?

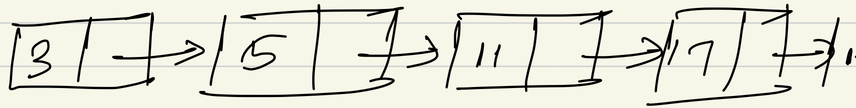
Inorder → linked list $O(n+m)$

Time - $O(n+m)$

Space - $O(n+m)$ X

→ Merge → BST

$O(n+m)$ → $O(n+m)^2$



already a BST

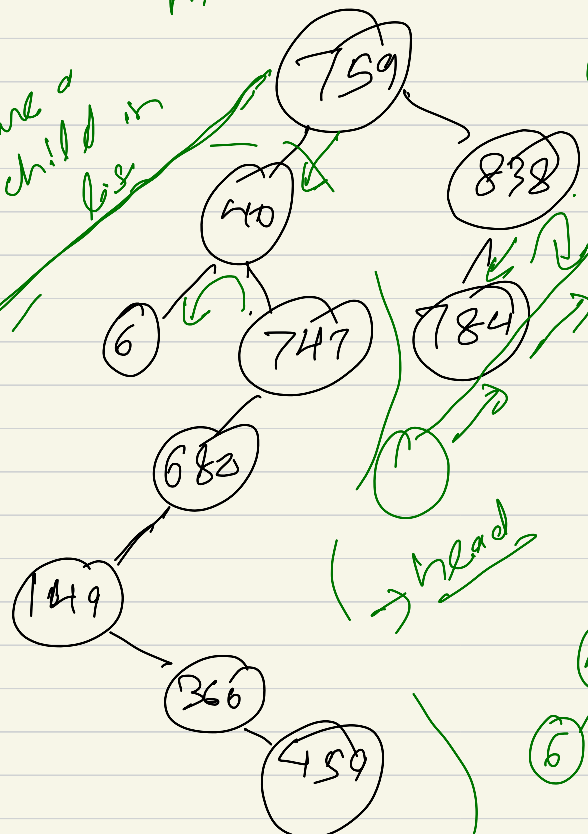
$O(n+m)$ time

$O(n+m)$ — space

759

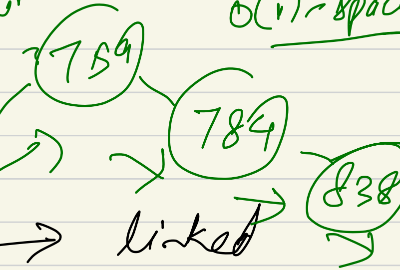
m

No node to have a right child in list



O(m) - time

O(1) - space



linked list ?

→ head