10 | Friday

8 am

Hashing

what is good Hashfunction →

9 - quick to complete
- Uniform distribution

10

How to deal with non-integer key?

11 Birthday paradox?
There always be collision.

12

hash code map key → integer

1 pm compression Map integer → [0, N-1]

2 Every time location must match.

3

Popular Hash code map

4 - Integer cast [32 bit intepretation]

5 - For more than 32bit add them up to get 32 bit

6 - Polynomial accumulation.
$a_0 + a_1x + \cdots + x^{n-1}a_{n-1}$.

7 $x = 33, 37, 39$ or 41 gives at most 6 collision on

8 50,000 english word.

9 Compression Map
$h(k) = k \bmod m$

Note: m-prime good not Powers of 2.

11 | Saturday

8 am

$0 \ldots k_{max} \longrightarrow [0 \ldots k_{max}]$ A

9 Take fractional part (mod 1)
map to into $0 \ldots m-1$

10 Choise of m is not ~~critical~~.
Critical.

11

Multiple, add, Divide

12 $h(k) = \lfloor a(kt + b) \rfloor \bmod N$
k - key, N - size of hash.

1 pm a should not multiple of N.

12 | Sunday

Universal Hashing:-
Randomly pick Hash function.

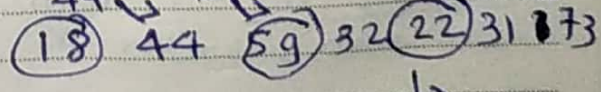
More on collisions:-
- Linear probing } open
- Double Hashing } Addressing.

Go on different location till find empty location.

No of element ≤ size of Table

Linear probing Example.
$h(k) = k \bmod 13$
18, 41, 22, 44, 59, 32, 31, 73

Note:



till ∴ go to next.

13 | Monday

Double Hash function.

$h_1(k)$ & $h_2(k)$

↳ position in table we 1st check for key

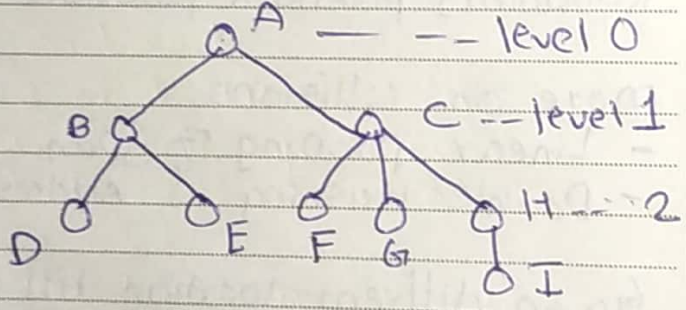
$h_2(k)$ → determine offset we use when searching key k.

e.g.

$h_1(k) = k \bmod 13$

$h_2(k) = 8 - (k \bmod 8)$

Trees



A → root

B, C, H → Internal nodes

D, E, F, G, I → leaves

Ordered tree - Tree in which the children of each node are ordered.

Note: Binary tree - At most 2 children, ordered.

14 | Tuesday

Complete Binary tree :-

level i has 2^i nodes.

Total nodes = $2^{h+1} - 1 = n$

internal nodes = $2^h - 1$

Height = $\log_2$ (no. of leaves)

No. of leave = $(n+1)/2$

Min. height of Binary tree with n nodes

$n <= 2^{h+1} + 1$

$h >= \log_2 (n+1)/2$

max. height = $(n-1)$ with n

no. of leaves - 1 <= no. of internal nodes.

ADT for trees:-

size(), isEmpty, elements positions(), isRoot(), root(), parent(), leftchild(), rightchild().

Tree Walks

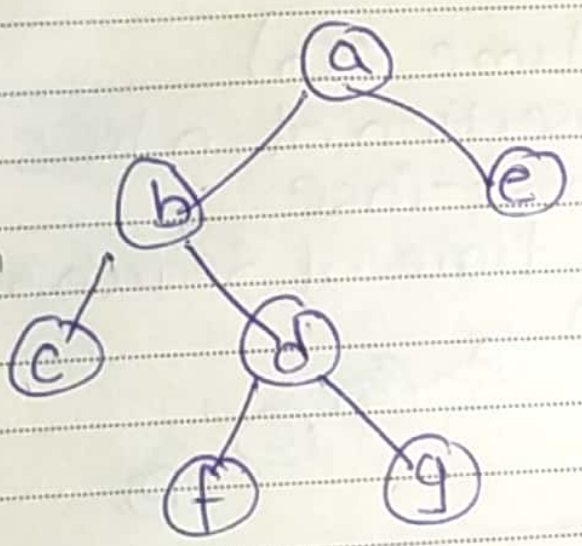
Preorder(v)

if (v == null) then return else visit(v)

Note: Preorder(v.leftchild())

Preorder(v.rightchild())

```
Postorder(v)
if (v==null) then return
else Postorder(v.leftchild())
     postorder(v.rightchild())
     visit(v)
```



Preorder - abcdfge
postorder - cfgdbea
Inorder - cbfdgae

* Given the preorder & Inorder traversals of binary tree we can uniquely determine tree. // same with post & inorder But can not build with pre-postorder

Note :

17 | Friday

Ordered Dictionaries

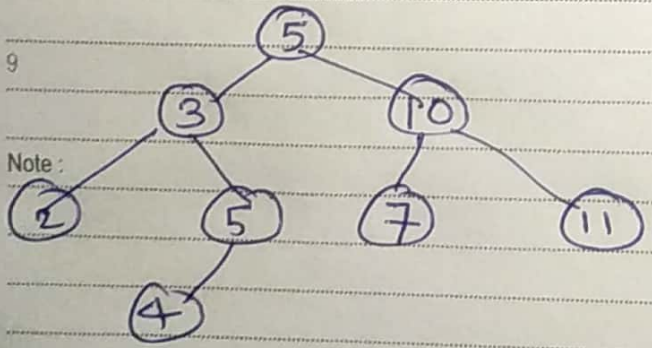
In addition to dictionary function we want additional function:
- Min()
- Max()
- Predecessor(S, k)
- Successor(S, k)

* Unordered list
34 - 14 - 12 - 22 - 18
Searching O(n)
inserting O(1)

Ordered
12 - 14 - 18 - 22 - 34
searching O(n)
insertion O(n)

Binary search runs in O(log n)

Searching in Bst

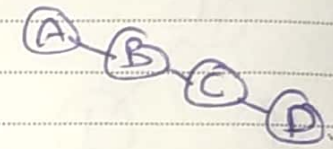


Note :

18 | Saturday

1) Compare 'k' with root.key
2) if k < root.key, search k in left of root
3) otherwise in right of root

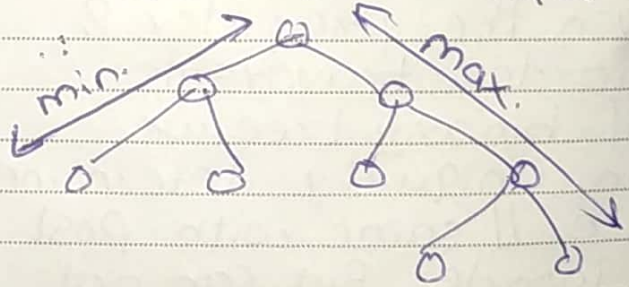
* Running time O(h)
* After insertion of n keys the worst-case running time of searching is O(n)



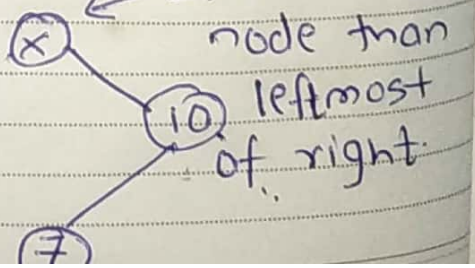
19 | Sunday

BST minimum

while left(root) ≠ NULL
do root ← left(root)

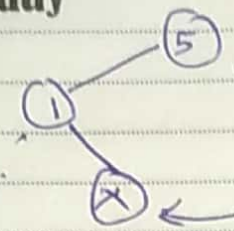


* Successor



Note :

20 | Monday



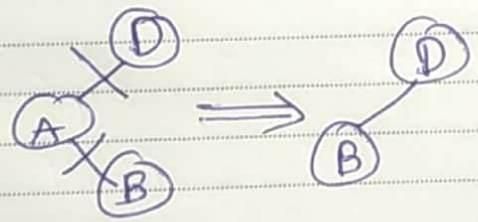
← If not have right nod than parent right

Deletion from tree

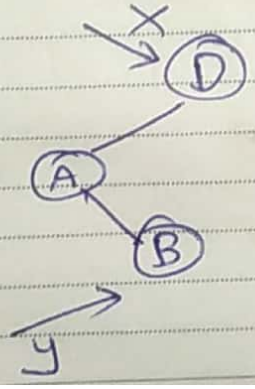
Delete node x from tree
we can have 3 cases
- x has no childern
- x has one childern
- x has two children

Case 1 -
Just remove x

case 2 -



case 3 -



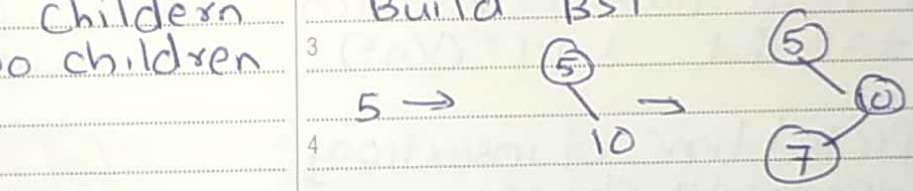
Note :

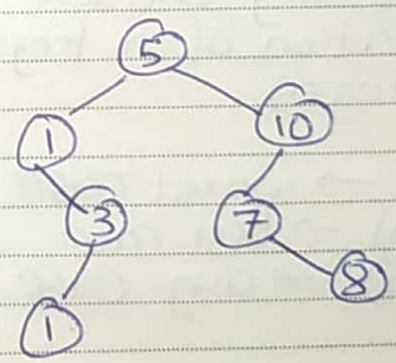
21 | Tuesday

For removing x find successor of x i.e. y & replace x with y (successor or predecessor)
Note y has at most one child (bcoz it has right child than that becomes successor)

BST Sorting

Sort the following no.
5 10 7 1 8 1 8
Build BST





call Inorder walk
1 1 3 5 7 8 10

Note :

22 | Wednesday

8 am

In what order should we insert?

- We want to sort numbers {1, 2, 3 ... n}
- Total time taken to insert these no. equals the sum of level no. of nodes
- Thus if no. inserted in ascending order we would get tree of (n-1) height
- So total time for insertion $1+2+3+\dots+n-1 = O(n^2)$

* Expected time of insertion a randomly chosen permutation of n keys is $O(n \log n)$

$\therefore O(n^2) \rightarrow$ worst case
$O(n \log n) \rightarrow$ Best case
$O(n \log n) \rightarrow$ avg. case

Note :