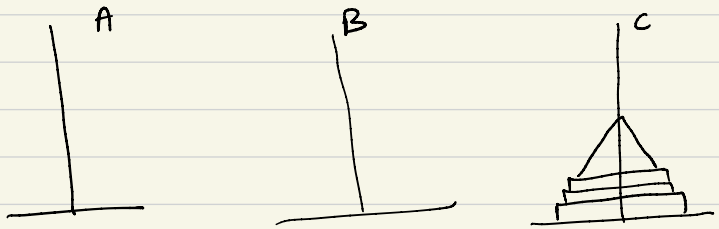
Recitation-8

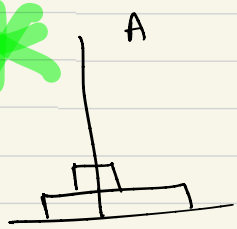
Thinking Recursively :-

Tower Of Hanoi :-

move A → C, such that a small ring always stays above a bigger ring

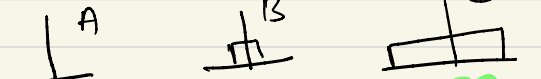
Need to make



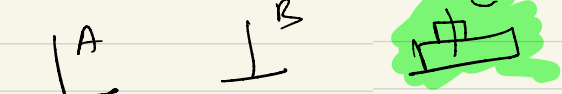


Move (A → B) ⇒

move (A → C)



move (B → C)



Ans:

move (1, A, B)
move (2, A, C)
move (1, B, C)

} Printed

Base:-

Ans: move ring 1 , A → C

Solution:-

Hanoi (N , A, C) ⇒ print all instructions to do this transf.

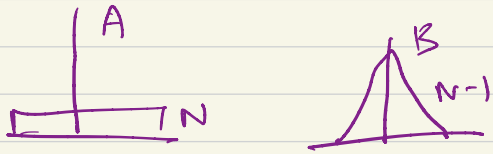
if I had a solution ath ready:- the

I would also know Hanoi (N-1, A, C)

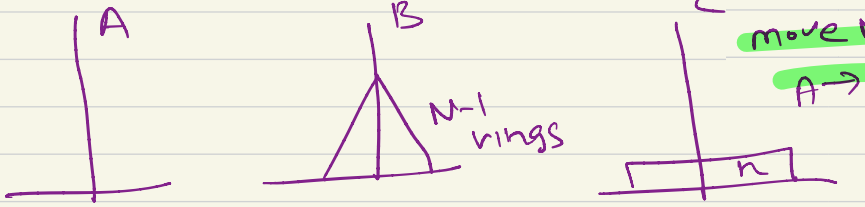
I would also know Hanoi (N, B, C)

" Hanoi (R, A, B)

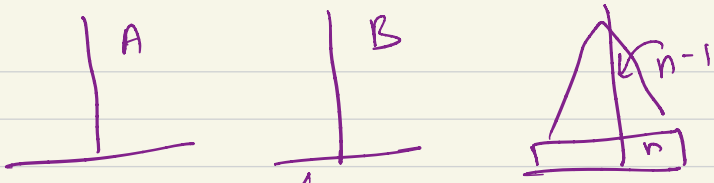
Step1:- move N-1 rings from A → B ⇒ Hanoi (N-1, A, B)



Step2:



Step 3:- Move N-1 rings from B → C :→ Hanoi (N-1, B, C)



Hanoi (N, A, C) = { Hanoi (N-1, A, B);
Print (move Nth ring from A → C);
Hanoi (N-1, B, C) }

Solution

For any Rec Qs:-

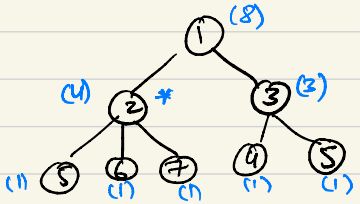
① Assume you have a soln → write down its form (Hanoi (N, A, C))

② Try to write down the Rec Eq

③ write down the base case

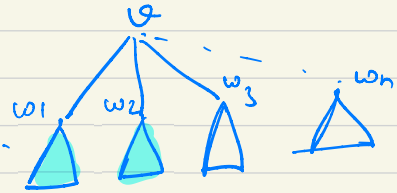
Q1] Given any tree, give me number of nodes in the tree :-

↳ Store @ each node ⇒ the # of nodes in the subtree.



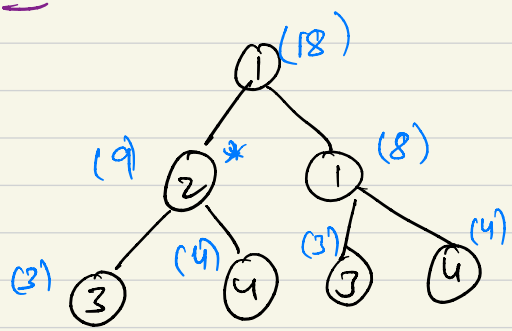
Node :-
value :- 2
children : [Node 5, Node 6, Node 7]
nds : 4 ←

NC(v):
v.nds ← 1
for each w in v.children :-
NC(w);
v.nds ← v.nds + w.nds
end for

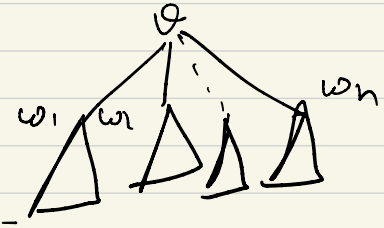


Num of nodes in v :- NC(v)
return v.nds;

eg Sum of each subtree :-



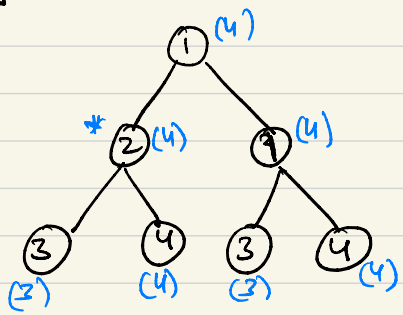
Node :-
val :- 2
children :- Node 3, Node 4
sum-nodes :- 9



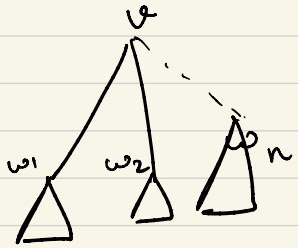
sum(v) :-
v.sum_nodes = v.val ;
for each wi in v.children :-
sum(wi);
v.sum_nodes = v.sum_nodes + wi.sum_nodes ;
end for :

v.sum_nodes = v.val + $\sum$ wi.sum_nodes

Q3] Maximum Node :-



Node :-
value : 2
child : 3, 4
max : 4



$v.max = max(v.value, \{Any\ child's\ w.max\})$

maximize (v) :-

```
v.max = v.val
for each wi in v.children :
        maximize (wi);
        v.max = max (v.max, wi.max)
end for;
```