Sunday, December 4, 2022

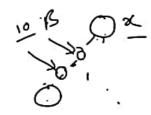
(1) Every mode is either Ped or Black

(2) The root la Black

(3) Every Reaf in Black

(4) Il a mod in Red both its Children are black

(5) & for each node all simple party from that mode to descendent leaver Contain the same number of black node



(1) A red black Tree of nihlernal moder her higher of "1"

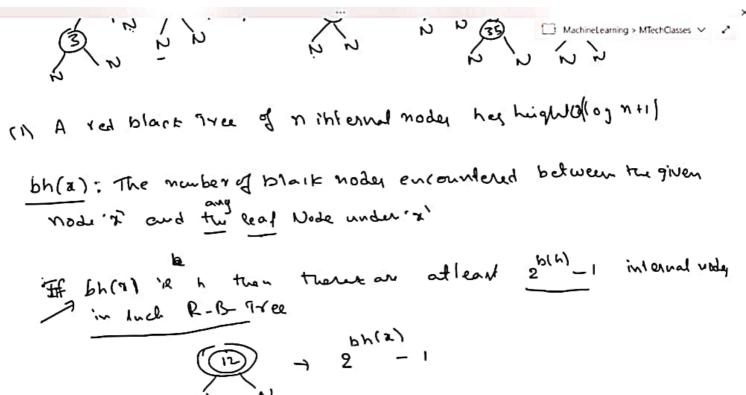
bh(a): The number of black noder encountered between the given noder it and the leaf Node under it

•





△ 0 IN ← d0 M 12/4/2022 0



The bh(7) is he there are at least 2 b(h) -1 interval volg

in had R-B-9-ree

bh(2)

2 -1

2 -1

> 2 -1

> 2 -1

> 2 -1

> 2 -1

> 2 -1

> 2 -1

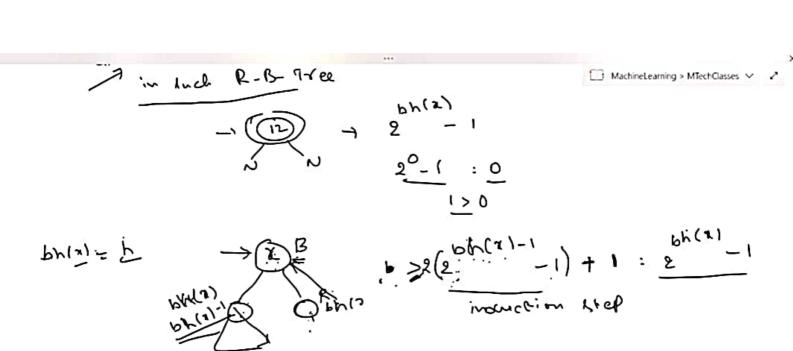
> 3 -1

> 3 -1

> 3 -1

> 4 -1

| 5 -1 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
| 5 -1
|



=/ 1

MachineLearning > MTechClasses

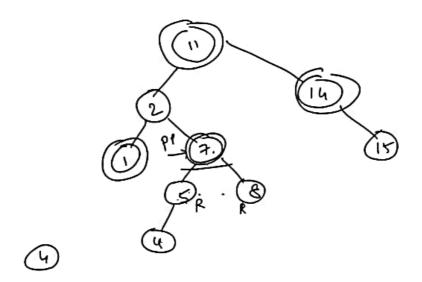
The blumber of internal moder of a PB.T with bh(T)

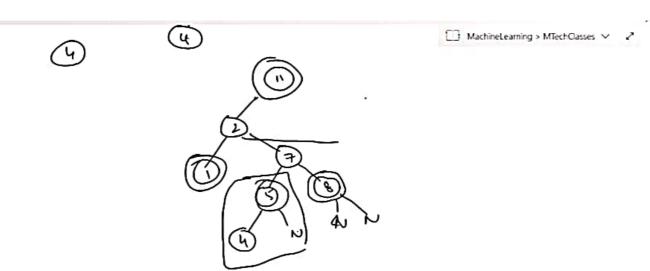
ic adleast > 2 bh(T)

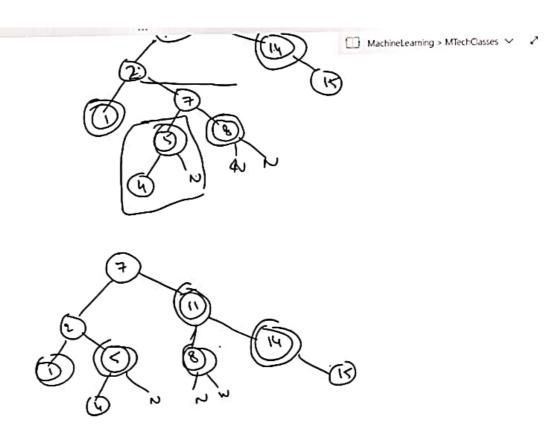
a tree with moder has height bound boy o(logn)

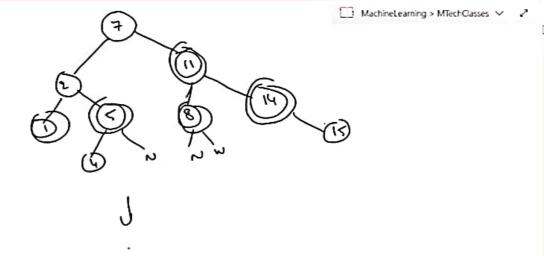
n-internal moder has height bounds of Machinelearning> MTechClasses v i

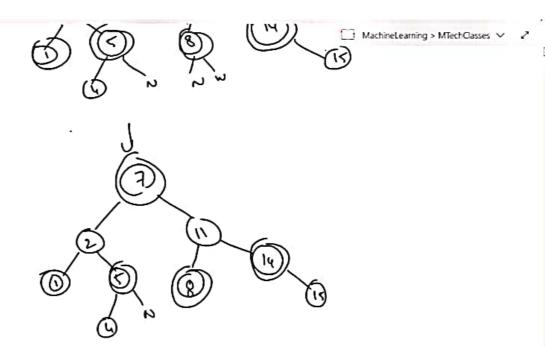
259 PM











```
# define RED = 1

# dufine BLACK: 0

PBTNode f

int dola;

PBTNode * Left , xeriqui;

int (olor);

RBTNode * insert ( RBT Node * P , int * X )

(F:= NULL) f

P= (RBTNode **) malloc( Size of (RBT Node));

P= Ada = X;

P> left: P= right: NULL;
```

```
PSINODE * Left , XPright;

int (olor);

RBTNode * insert ( PBT Node * P, int x)

if (P:= NULL) {

P= (RBTNode *) malloc( Size of (RBT Node));

P>100 = RED;

P-> (olor = RED;
```

```
P=> left: P=> right: NOLL;
P=> (0) or = RED;

else if (x< P=> ala) {
P=> left = p=> intent (P=> keft, x);

else if (x> P=> alata) {
P=> right = intent (P=> right, x);
}
```

```
else if (x< p->data) {

P-> left = p-> invert (p->keft, x);

else if (x> p->data) {

p-> right = invert (p-> vight, x);

}

return fix-rolova(p);
```

MachineLearning > MTechClasses

```
BTNOdex fix_(olor) {

if (P = olor) = BLA(c) {

if (P = left = olor) = RED) & (P = right = rolor = RED)

{

if (P = left = left = rolor = RED) | (P = left = right = left + rolor = RED)

| (P = right = left = rolor = RED) | (P = left = rolor = RED)

| (P = right = left = rolor = RED) | (P = right = left + rolor = RED)

| P = right = left = rolor = BLACK;

| P = right = rolor = RED | RED;

| P = rolor = R
```

```
if (P=) (olov = : BLA(K) {

If ((P=) left=) (olov = : RED) & (P=) vight=) (olov = RED))

{

If ((P=) left=) (olov = : RED) & (P=) vight=) (olov = RED)

If ((P=) left=) (olov = : RED) & (P=) left=) vight=) (olov = RED)

If (P=) left=) (olov = RED) & (P=) vight=) (olov = RED)

{

P=) (olov = RED) & P=) left=) (olov = BLAKK) ((P=) vight=) (olov = RED)

If (P=) left=) (olov = RED) & (P=) left=) (olov
```

```
if ((P > left > (olov = = RED) & & (P > right -) [] Machinelearning > Michicasses > []

[(P > left > left -) (olon = = RED) | (P > left -) vight -) (olon = RED)

[(P > left +) left -) (olon = = RED) | (P > kight -) left + volov == RED)

[(P > left +) (olon = = RED) | (P > kight -) left + volov == RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = REACIL;

]

[(P > left -) (olon = = RED) | (P > kight -) (olon = REACIL;

]

[(P > left -) (olon = = RED) | (P > kight -) (olon = REACIL;

]

[(P > left -) (olon = = RED) | (P > kight -) (olon = REACIL;

]

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

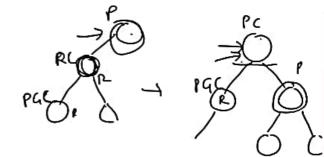
[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

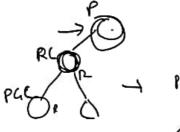
[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

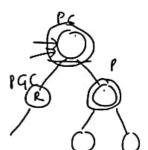
[(P > left -) (olon = RED) | (P > kight -) (olon = RED)

[(P > left -) (olon = RED) | (lon = RED)

[(P > left -) (ol
```







```
II (Paright aleft a color = RED) II (Paright al. Machinelearning > Milet Classes >

{

Pacolor = RED; Paleft a color = BLAKK; Paright a) (olor = BLAKK;

}

If (Paleft a) color = RED) {

if (Paleft a) color = RED) {

Pacolor = BLAKK; RED;

Pacolor = BLAKK; RED;

Paright a) Color = RED;
```

```
if (Palett -> colon = = RED) {

if (Palett -> left -> left -> colon == RED) {

P = rotate right(P);

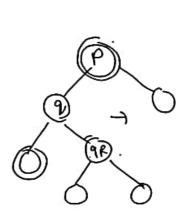
P => (olon = BLACK; RED;

P = right -> colon == RED) {

if (P -> left -> right -> colon == RED) {

Potate left Right(P);

Potate left Right(P);
```



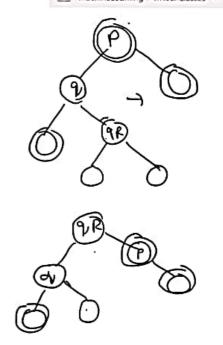
MachineLearning > MTechClasses

chineLearning > MTechClasses ∨

P-> vight > Color = re-;

if (P-> left-> right -> rolor == RED) {

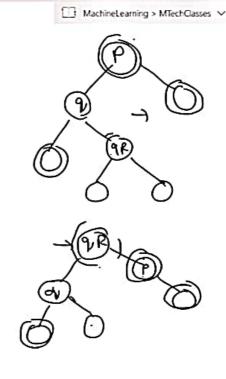
rotale left Right (P) i



Paright & color= re-;

if (Parleft aright arolor == RED) [

P = rotate left Right (P) i



P-> vight > (olov = re-)

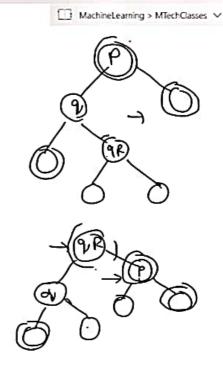
if (P-> left -> vight -> rolod == RED)

p = rotale left Right (P);

p -> rolov = BLACK

p -> right + Colov = RED

p -> vight + Colov = RED



```
If (P=left = color = RED) {

if (P=left = left = rolor = RED) {

P = rolateright(P);

P=rolateright (P);

P=rolateright = rolor = RED);

if (P=left=right=rolor) = RED);

P = rotate left Right (P);

P=rolar = BLACC

P=right = Color = RED

}

ImachineLearning > MicchClasses >

MachineLearning > MicchClasses >

MachineLearning > MicchClasses >

Image: P=RED);

P=rolaterial = rolor = RED);

ImachineLearning > MicchClasses >

Image: P=RED);

Image: P=RED);

ImachineLearning > MicchClasses >

Image: P=RED);

Image:
```

☐ MachineLearning > MTechClasses ∨

P= right > (0/04 = real)

if (P=) left=right - 1/0/04 == RED)

P = rotale left Right (P);

P=(0/04 = BLACK

P=(0/04 = RED

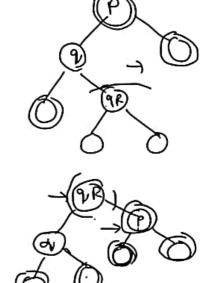
p=right + Color = RED)

if (P=) right - 3/0/04 == RED)

if (P=) right = 3/0/04 = - RED)

if (P=) right = 3/0/04 = - 7/0/04 == - RED)

Example 1



P=rotale left Right (P) i

if (P=right=rolon==RED) &

if (P=right=rolon==RED) {

P= totale left (P))

P=rotale left (P))

P=rotale left (P)

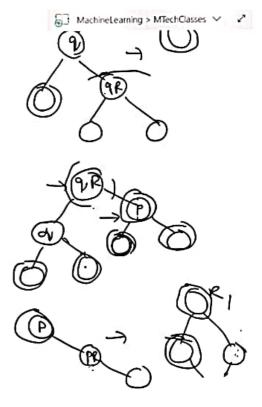
P=rotale left (P)

P=rotale left (P)

P=rotale left (P)

P=rotale left Right (P) i

P=rotale lef



P= rotale left Right (P) i

if (P= right = right = rolon== RED) {

P= rotale left (P))

P= rotale left (P))

P= rotale left (P) i

Blif (P= right = right left (P))

P= rotale right left (P);

P= rotale right left (P);

P= rotale right left (P);

P= right + (old)

if (P= right - scolor = : RCD) {

if (P= right - scolor = : RCD) {

P= sot all left(P))

P= lolar : RLACK;

P= lolar : RLACK;

P= lolar : RLACK;

P= lolar : RLACK;

}

Machinetearning > MicchClasses > P

Machinetearning > MicchC

P-> lest -> (olor = K&D)

Blif (P-> Vight -> lest +> color = = RED) &

P>= votate vight lest (P);

P== lest -> (olor : BLACK;

P-> lest -> (olor : BLACK;
}

MachineLearning > MTechClasses > 2