امیرحسین نجفی زاده 9831065

الف) اعداد زیر را به ترتیب از چپ به راست وارد یک درخت قرمز-سیاه خالی کنید و در هر insertion ، در صورت نقض یکی از خصوصیات درخت قرمز-سیاه، caseآن را بیان کنید و مراحل تصحیح آن را نشان دهید.

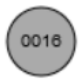
```
16,33,21,17,13,25,12,19,26,40,22,24
```

Empty tree

Insert 16:

1. Create a red node with key 16
2. Put the new node at root (violate property 2)
3. Change root color to black (Fixing property 2)

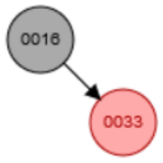
Tree :



Insert 33 :

1. Find the 33 place, which is the right of the 16
2. Create a red node with key 33
3. Put the new node to the right of the 16
4. No violation

Tree :

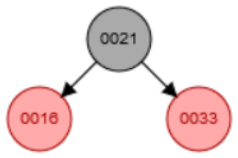


Insert 21 :

1. Find the 21 place, by inserting algorithm in BST. It will be at the left of 33
2. Create a red node with key 21
3. Put the new node to the left side of 33 (violate property 4)
4. 33 is the left child of p[p[new node]] so look for the 16 left child => left child of 16 is black (case 2 / 3)

5. New node is the left child of p[new node] => new node is now 33
6. Left rotate on 33 (case 2)
7. Color p[new node] black => 21 is now black (fix property 4)
8. Color the p p [new node] red => 16 is now red (violate property 5)
9. Left rotate on p p [new node] left rotate on 16 (property 5 fixed) (case 3)
10. Color the root black

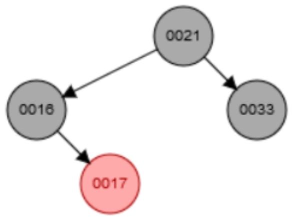
Tree :



Insert 17:

1. Find the place of 17 => which will be the right side of 16
2. Create a new red node with key 17
3. Put the red node at the right of 16 (violate property 4)
4. 17 parent is 16 and it is the left side of 21 (p p 17)
5. Look for the right child of p p 17 => which is 33 and red (case 1)
6. Set p 17 to black => 16 is now black (Fixe property 4 / violate property 5)
7. Set right p p 17 to black => 33 is now black (fixed property 5)
8. Color the p p 17 to red => 21 is now red (violate property 2)
9. Set new node to p p 17 => current node is 21
10. P 16 is not red => exit loop
11. Set root to black => 21 is black (fix property 2)

Tree :

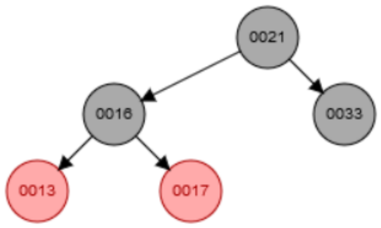


Insert 13:

1. Find the place of 13 => which is the left side of 16
2. Create a new red node with key 13

3. Place it at the left side of 16
4. No violation

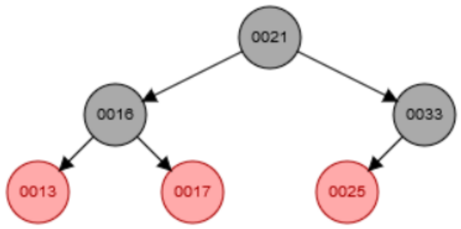
Tree :



Insert 25 :

1. Find the place of 25 => which is the left side of 33
2. Create a red node with key 25
3. Place it at the left of 33
4. No violation

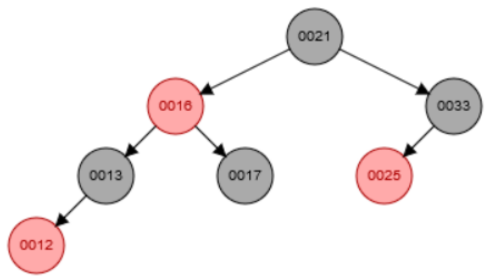
Tree :



Insert 12:

1. Find the place of 12 => which is the left side of 13
2. Create a new red node with key 13
3. Place it to the left side of 13 (violate property 4)
4. 13 is the left child of p p 12
5. Look for the right child of p p 12 => which is 17 and red (case 1)
6. Color p 12 to black => 13 is now black (violate property 5)
7. Color right p p 12 to black => 17 is now black
8. Color p p 12 to red => 16 is now red (fix property 4 and 5)
9. Set new node to p p 12 => 16 is new node
10. P new node is black
11. No violation

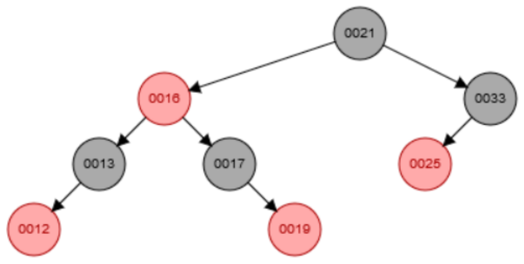
Tree :



Insert 19:

1. Find the place of 19 which is the right side of 17
2. Create a new red node with key 19
3. Place it at the right of 17
4. No violation

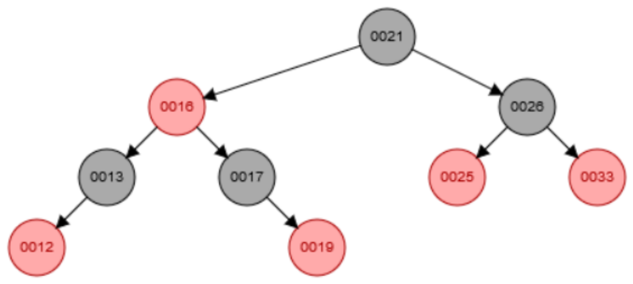
Tree :



Insert 26:

1. Find the place of 26 => which is the right side of 25
2. Create a new red node with key 25
3. Place it to the right side of 25 (violate property 4)
4. P 26 is the left side of the p p 26 which is 33
5. Look for the right child of p p 26 => which is nil and black (case 2 / 3)
6. 26 is the right child of its parent => Set new node to 25
7. Left rotate on 25 (case 2)
8. Set p z color to black => 26 is now black (violate property 5 / fix property 4) (case 3)
9. Color p p 26 to red => 33 is now red
10. Right rotate on p p 26 => right rotate 33 (fix property 5)
11. No violation

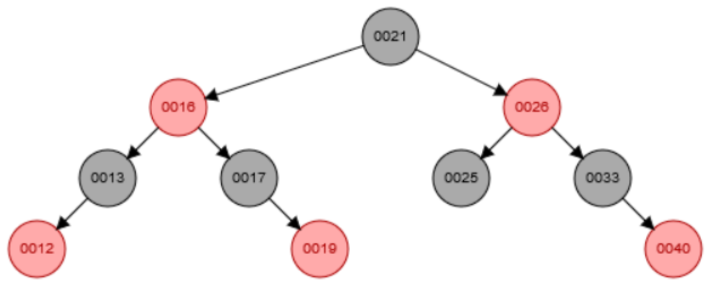
Tree :



Insert 40:

1. Find the place of 40 => which is the right side of 33
2. Create a new red node with key 40
3. Place it at the right of 33 (violate property 4)
4. P 40 is the right of p p 40
5. Look for the left child of p p 40 which is 25 and red (case 1)
6. Set p 40 color to black => 33 is now black (violate property 5 / fix property 4)
7. Set left p p 40 color to black => 25 is now black (fix property 5)
8. Set p p 40 to red => 26 is now red (
9. Set new node to p p 40 => new node is now 26
10. P 26 is black => exit the loop
11. No violation

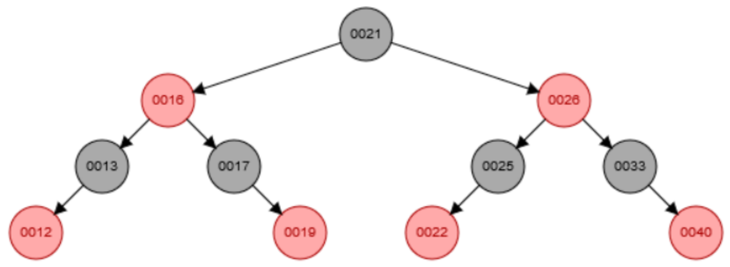
Tree :



Insert 22:

1. Find the place of 22 => which is the left side of 25
2. Create a new red node with key 25
3. Place it at the left of 25
4. No violation

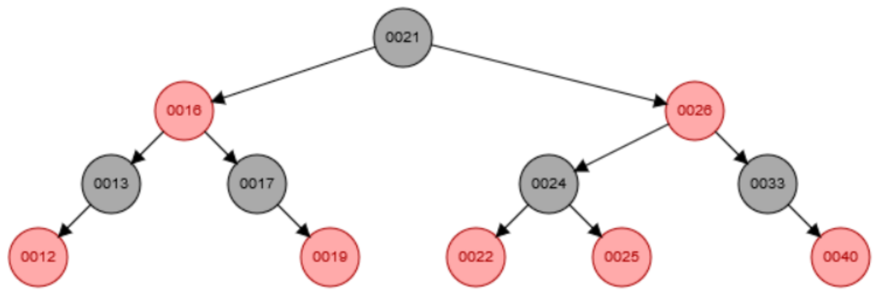
Tree:



Insert 24:

1. Find the place of 24 which is the right of 22
2. Create a red node with key 24
3. Place it at the right of 22 (violate property 4)
4. P 24 is the left child of p p 24
5. Look for the right child of p p 24 => which is nil and black (case 2 / 3)
6. 24 is the right child of 22 => set new node to 22
7. Left rotate on 22 (case 2)
8. Set p new node color to black => 24 is now black (fix property 4 / violate property 5) (case 3)
9. Set p p 22 color to red => 25 is now red (violate property 4)
10. Right rotate on p p 22 => right rotate on 25 (fix property 4 and 5)
11. No violation

Tree :



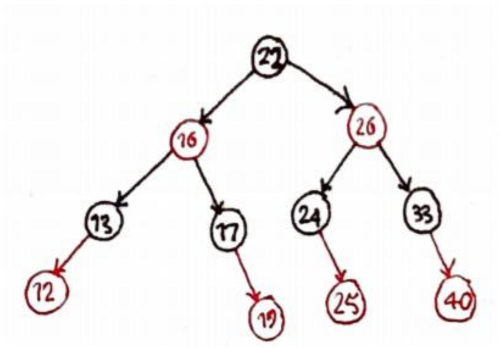
ب) اعداد زیر را از درخت به دست آمده حذف کنید، مراحل حذف را توضیح دهید و در صورت نقض هر یک از خصوصیات درخت قرمز-سیاه، caseآن را بیان کنید و مراحل تصحیح آن را نشان دهید.

21,22,24

Delete 21:

1. Find the replace for 21 => which is 22 (21 successor)
2. Change the 21 node key to 22
3. 22 is red with no child (case 0)
4. Remove 22
5. No violation

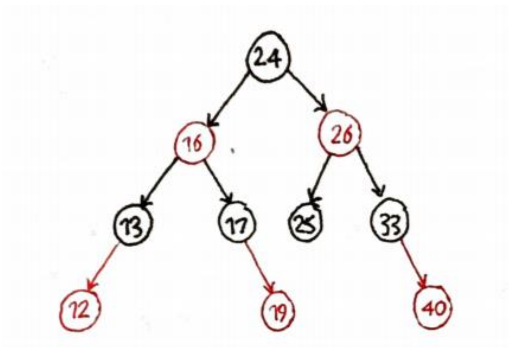
Tree:



Delete 22:

1. Find the replace for 22 => which is 24
2. 24 is black with a right red child => 25
3. Remove node with key 24 (violate property 4)
4. Change the color of 25 to black (fixed property 4)
5. Replace the node 22 key to 24
6. No violation

Tree:

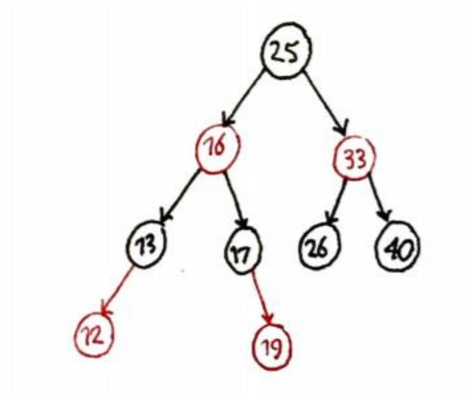


Delete 24:

1. Find the replace for 24 => which is 25
2. 25 is black with no children

3. Remove node with key 25 (violate property 5)
4. Color the nil into extra black (violate property 1 => cause of double black)
5. Copy the key 25 into node
6. Nil is the left child of p[nil] which is 26 => Set W to 33 (right of p[nil])
7. W is black with one red and one black child (not case 1 not case 2)
8. Right of W is red (not case 3)
9. We are in case 4 so => set w color as p[nil] color => w (33) is red (violate property 4)
10. Set p[nil] color to black => 26 is now black
11. Set color right W to black => 40 is now black (fix property 4)
12. Left rotate of p[nil] => left rotate on 26 (fix property 5)
13. No violation remain

Tree:



ج) ارتفاع اصلی و ارتفاع سیاه درخت نهایی را به دست آورید.

ارتفاع درخت برابر است با طولانی ترین مسیر به سمت برگ ها که در اینجا مسیر 25 تا 19 است به اندازه 4.

ارتفاع سیاه نیز برابر است با تعداد نود های مشکی از ریشه تا یک برگ که در اینجا برابر با 2 است. (25 شمرده نمی شود ولی nil شمرده می شود)