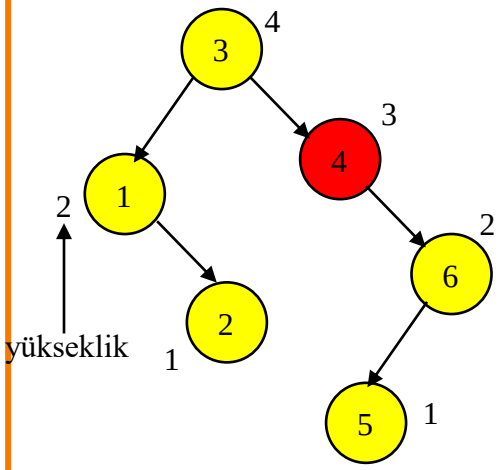
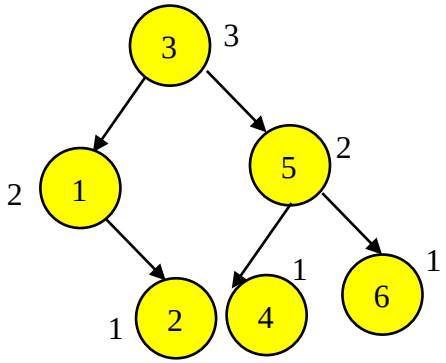
AVL Trees

- Eğer özel önlem alınmaz ise İkili Arama Ağaçları (Binary Search Trees – BST) gelişigüzel dengesiz olabilir ve işlemlerin en kötü O(n) zaman almasına yol açabilir
- AVL Trees (ismini bulucularından alıyor - Russians G.M. Adelson-Velsky and E.M. Landis) bir dengeli ikili arama ağacıdır
  - ✓ Dengeli (hemen-hemen) ikili arama ağacıdır
  - ✓ Ağaçtaki her bir düğüm için sol alt ağaç ile sağ alt ağaç yükseklikleri farkı en fazla 1'dir
- AVL ağaçlarının yüksekliği aşağı yukarı $1.44 \times \log_2 n$ (n düğüm sayısı olmak üzere) ile sınırlıdır.
- Düğümlerde yükseklik bilgisi saklanır.
- AVL ağaçlarında düğüm ekleme $O(\log_2 n)$ rotasyona gereksinim duyabilir

AVL Trees



BST fakat AVL değil



Hem BST hem de AVL

```
struct node{
    int key;
    struct node *left;
    struct node *right;
    int height; // derinlik
};
```

AVL Trees

- Ekleme ve silmeler AVL ağacının denge koşulunu bozabilir
- Denge koşulu bozulan AVL ağacına rotasyonlar uygulanarak dengesi düzeltilir
- 4 farklı rotasyon mevcut
  - ✓ Tekli sol rotasyon
  - ✓ Tekli sağ rotasyon
  - ✓ Sol-sağ rotasyon
  - ✓ Sağ-sol rotasyon

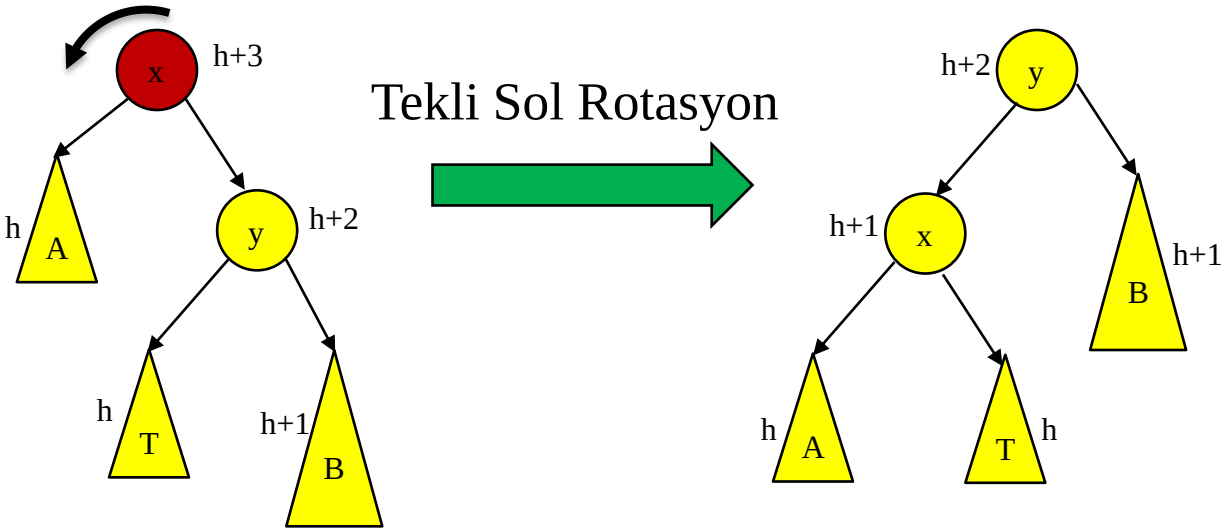
Tekli Sağ Rotasyon

y düğümünün sol çocuğunun (x düğümü) sol alt ağacına (A) ekleme yapıldı ve denge bozuldu (A alt ağacının yüksekliği h dan h+1 e çıktı). Denge, y düğümünden itibaren bozulduğu için y düğümü referans alınarak tekli sağ rotasyon uygulandı ve denge koşulu yeniden sağlandı.

```
struct node{
    int key;
    struct node *left;
    struct node *right;
    int height; // derinlik
};
```

```
struct node *rightRotate(struct node *y){
    struct node *x=y->left, *T=x->right;
    x->right=y;
    y->left = T;

    y->height=max(height(y->left),height(y->right))+1;
    x->height=max(height(x->left),height(x->right))+1;

    return x;
} //
```

Tekli Sol Rotasyon

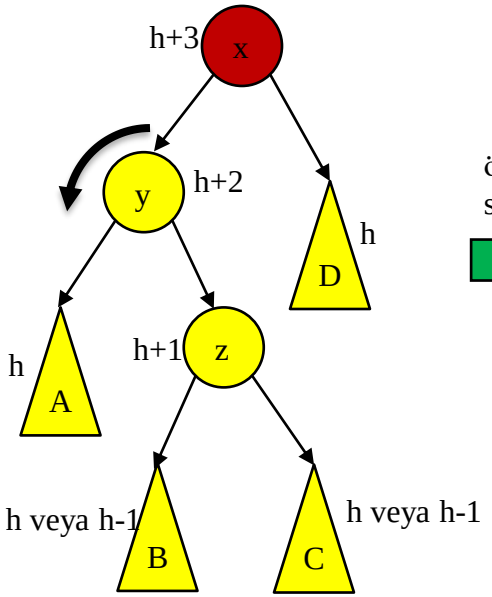


x düğümünün sağ çocuğunun (y düğümü) sağ alt ağacına (B) ekleme yapıldı ve denge bozuldu (B alt ağacının yüksekliği h dan h+1 e çıktı). Denge, x düğümünden itibaren bozulduğu için x düğümü referans alınarak tekli sol rotasyon uygulandı ve denge koşulu yeniden sağlandı.

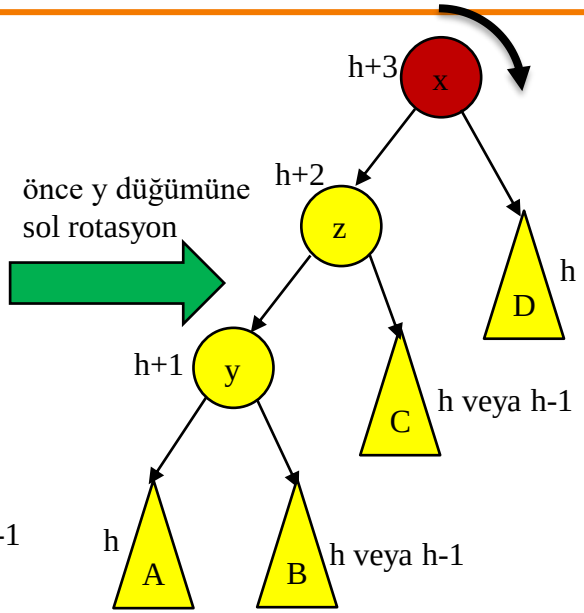
```
struct node{
    int key;
    struct node *left;
    struct node *right;
    int height; // derinlik
};
```

```
struct node *leftRotate(struct node *x){
    struct node *y=x->right, *T=y->left;
    y->left=x;
    x->right = T;

    x->height=max(height(x->left),height(x->right))+1;
    y->height=max(height(y->left),height(y->right))+1;

    return y;
} // sag sag
```

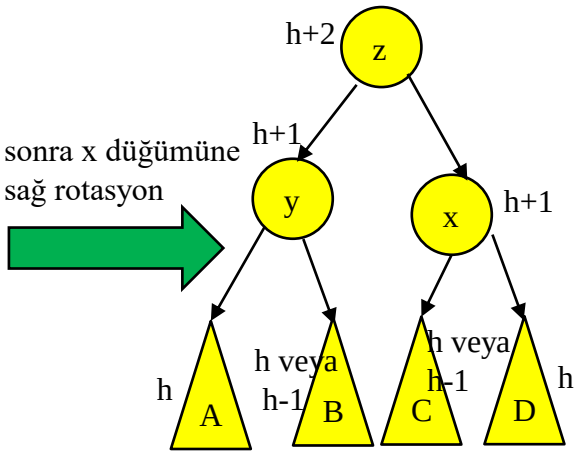
Sol-sağ Rotasyon



x düğümünün sol çocuğunun (y düğümü) sağ alt ağacına (ya B ya da C ye) düğüm eklenerek denge bozuldu. Önce y düğümüne sol rotasyon uygulanır.



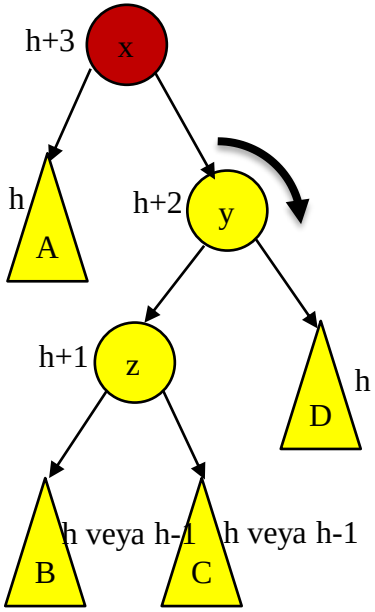
Sonra, x düğümü referans kabul edilerek sağ rotasyon uygulanırsa tekrar denge sağlanır



Ekleme/silme sonucunda dengesi bozulan AVL ağacı tekrar dengeli hale getirilmiş oldu

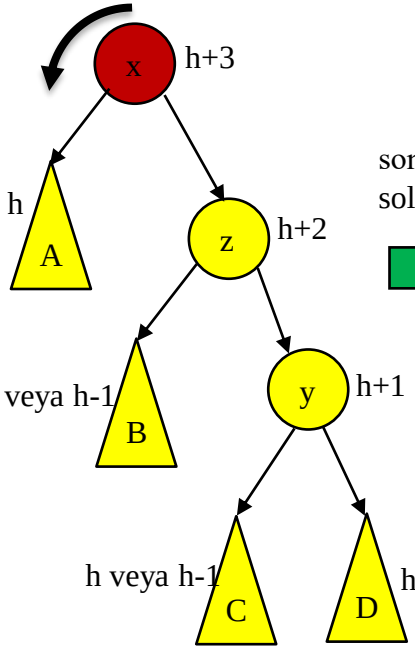
```
x->left=leftRotate(x->left);
rightRotate(x);
```

Sağ-sol Rotasyon



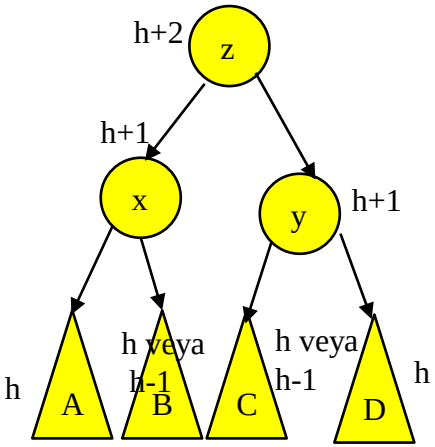
x düğümünün sağ çocuğunun (y düğümü) sol alt ağacına (ya B ya da C ye) düğüm eklenerek denge bozuldu. Önce y düğümüne sağ rotasyon uygulanır.

önce y düğümüne sağ rotasyon



x düğümü referans kabul edilerek sol rotasyon uygulanırsa tekrar denge sağlanır

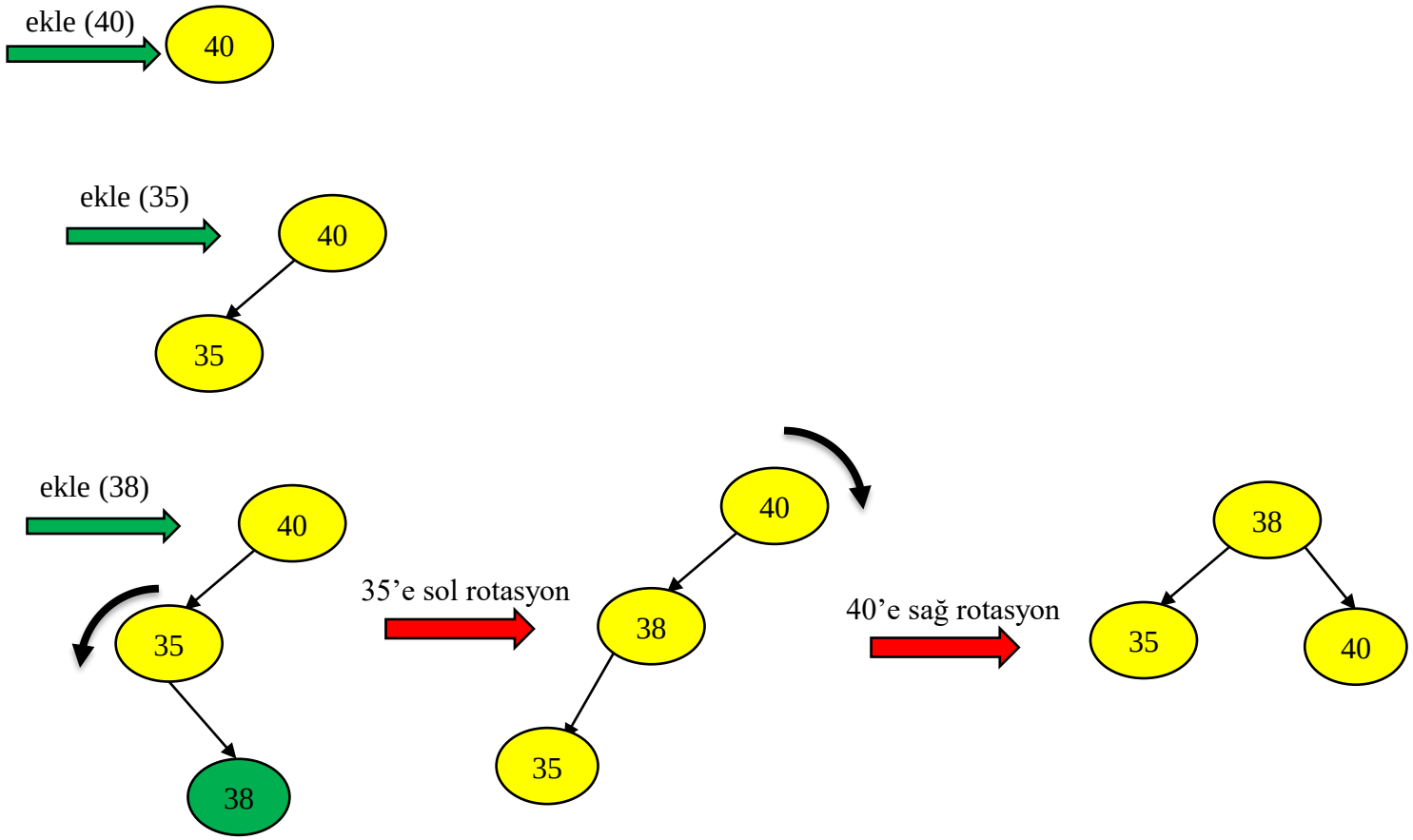
sonra x düğümüne sol rotasyon



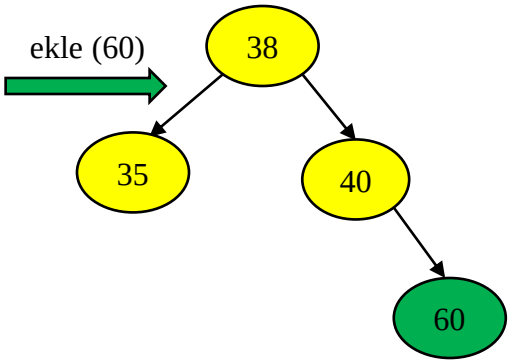
Ekleme/silme sonucunda dengesi bozulan AVL ağacı tekrar dengeli hale getirilmiş oldu

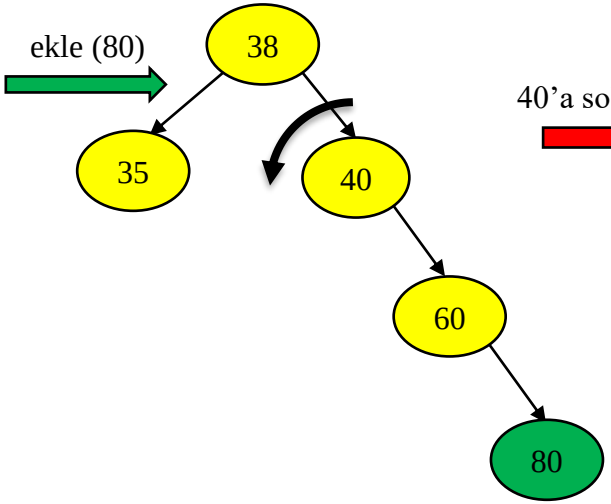
```
x->right=rightRotate(x->right);
leftRotate(x);
```

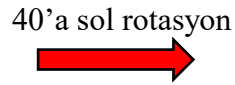
Ekleme

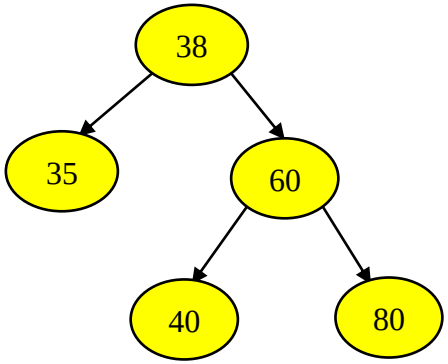

ekle (40)
40
ekle (35)
40
35
ekle (38)
40
35
38
35'e sol rotasyon
40
38
35
40'e sağ rotasyon
38
35
40

Ekleme

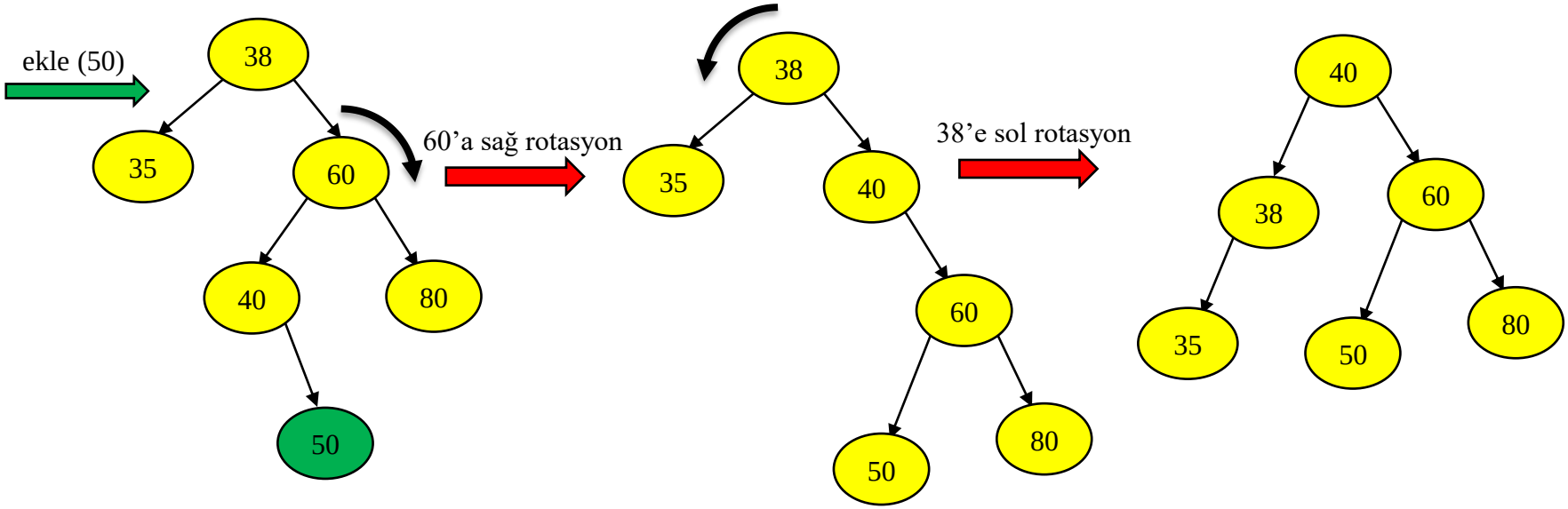

ekle (60)
38
35
40
60


ekle (80)
38
35
40
60
80

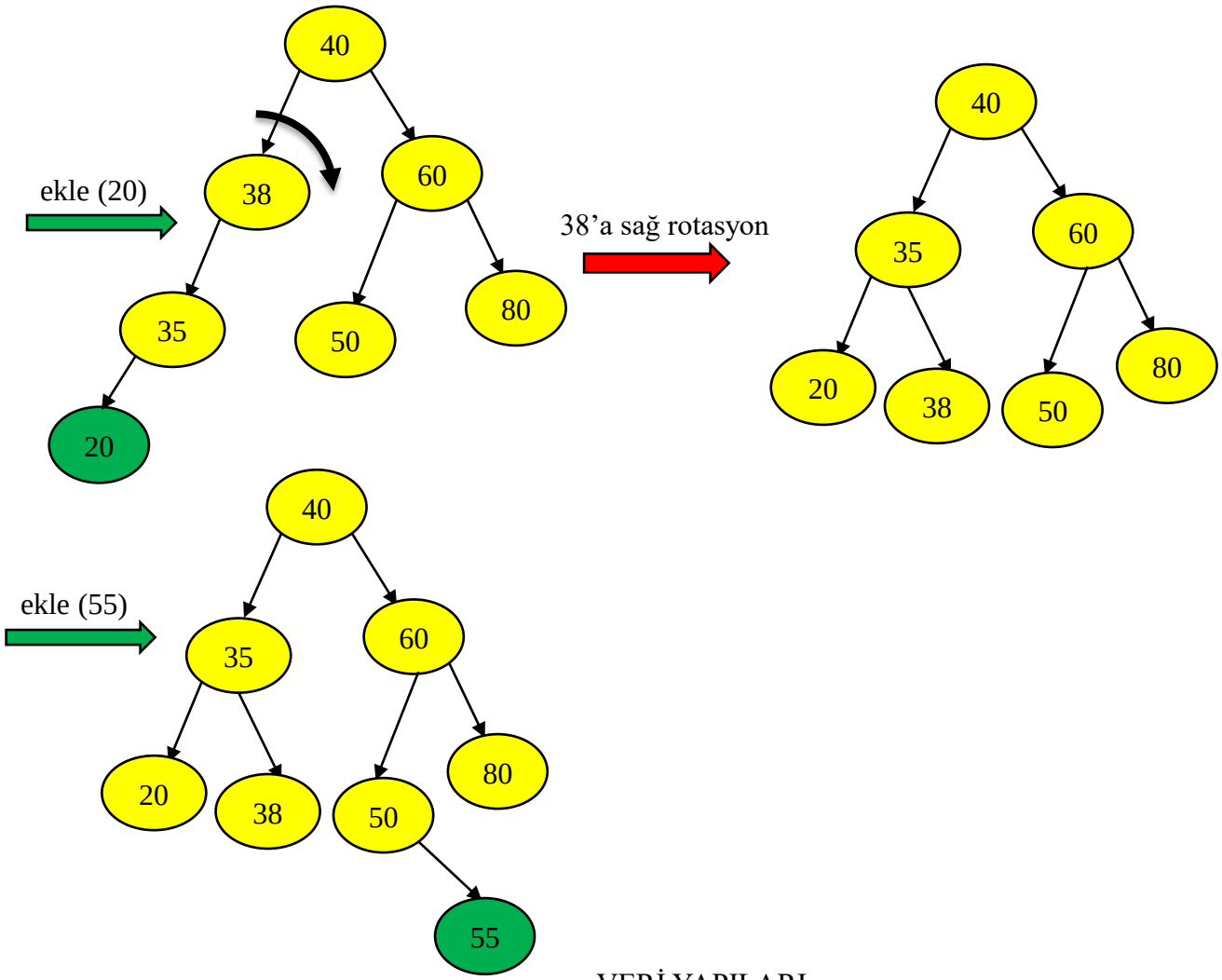

40'a sol rotasyon


38
35
60
40
80

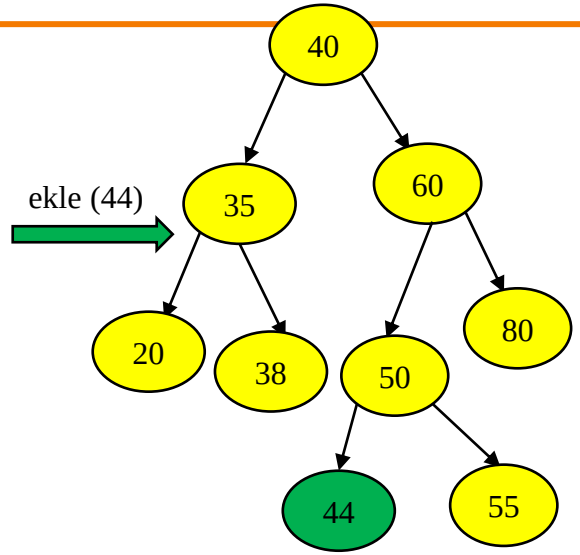
Ekleme

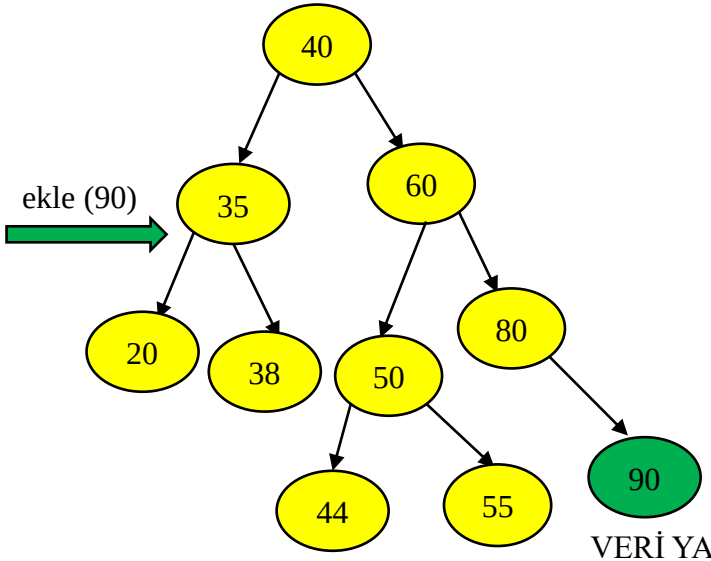

ekle (50)
38
35
60
40
80
50
60'a sağ rotasyon
38
35
40
60
50
80
38'e sol rotasyon
40
38
60
35
50
80

Ekleme

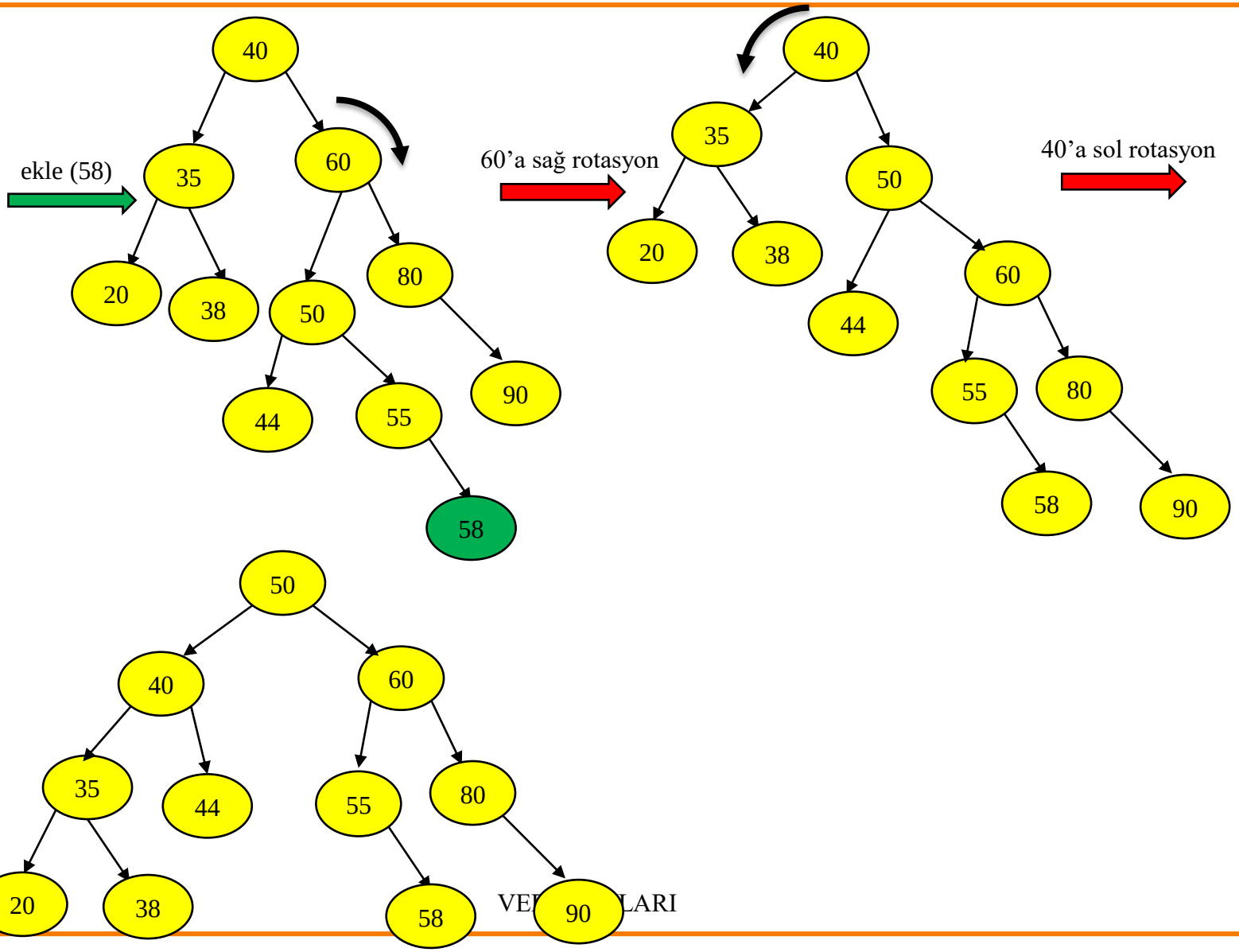

40
ekle (20)
38
60
35
50
80
20
38'a sağ rotasyon
40
35
60
20
38
50
80
ekle (55)
40
35
60
20
38
50
80
55

Ekleme


40
ekle (44)
35
60
20
38
50
80
44
55


40
ekle (90)
35
60
20
38
50
80
44
55
90

Ekleme


ekle (58)
60'a sağ rotasyon
40'a sol rotasyon
40
35
60
20
38
50
80
44
55
90
58
40
35
50
20
38
44
60
55
80
58
90
50
40
60
35
44
55
80
20
38
58
90