

Veritabanı Yönetim Sistemleri

Dr. Öğr. Üyesi Ahmet Arif AYDIN

B+ Tree

- Dosya Yapıları
 - Pages
 - Files
 - Disk space manager
 - Buffer manager
- Index yapıları
 - Hash based indexing
 - Tree based indexing

- Esnek ve dinamik bir yapısı bulunmaktadır.
- Her bir **düğüm** disk üzerinde bir **sayfadır**.
- İki adet düğüm çeşidi bulunmaktadır:
 1. Yaprak Düğümler (leaf nodes)
 2. Yaprak olamayan düğümler (non-leaf nodes)

B+ Tree : 1 - Yaprak Düğümler(leaf nodes)

```
struct YaprakNodeNode {  
    vector<Key> keys;  
    vector<Value> values;  
    PagePointer sonraki_sayfa;  
}
```

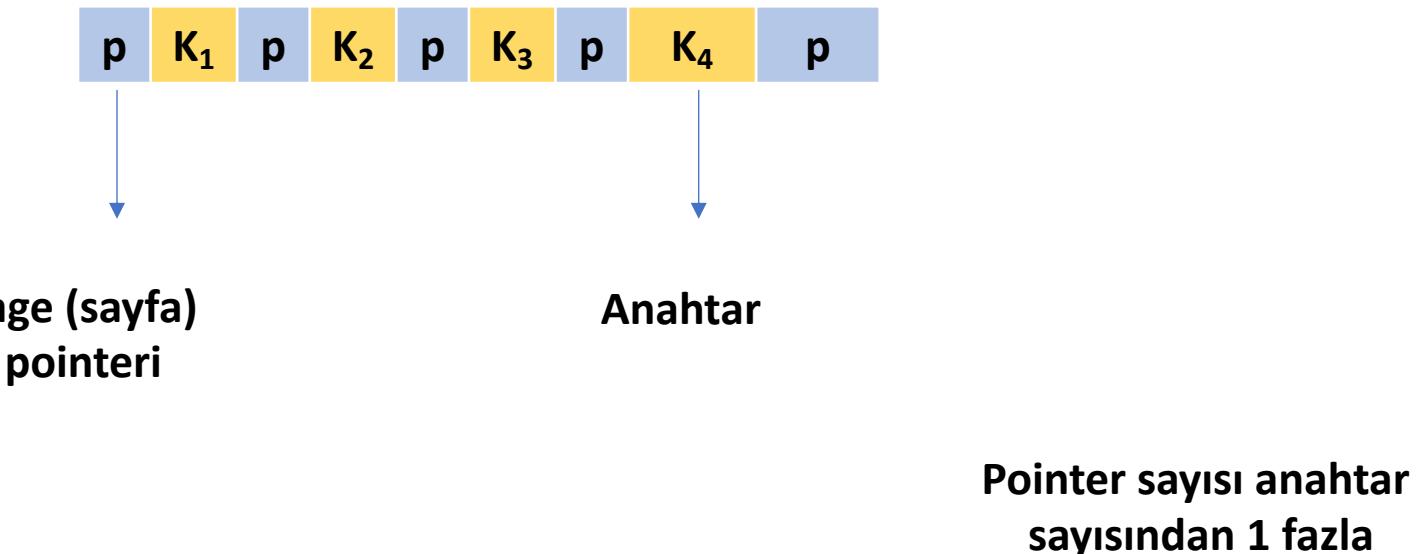


Verinin kaydedildiği sayfalar birbirine bağlı
Linkedlist ile tanımlanabilir.

Düğümlerde bulunan anahtarlar (key) sıralıdır
Yapraklarda (leaf) bulunan değerlerde sıralıdır.

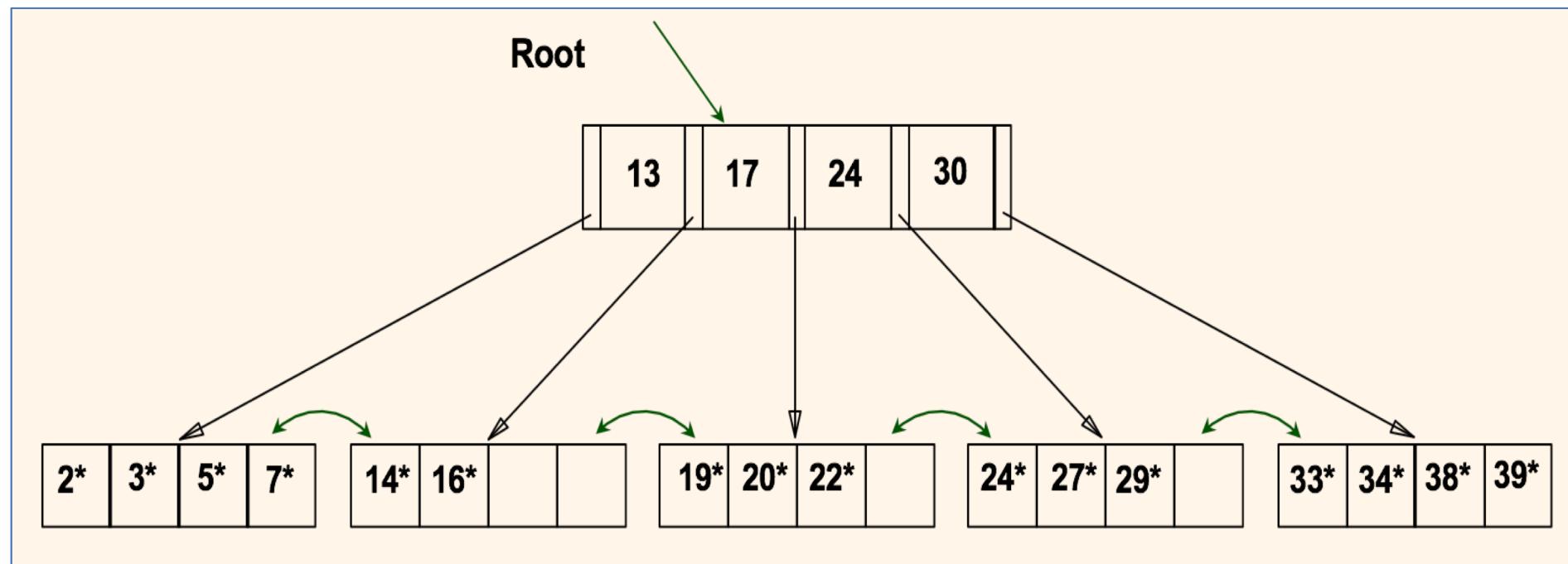
B+ Tree : 2-Yaprak Olmayan Düğümler(non-leaf nodes)

```
struct InteriorNode {  
    vector<Key> keys;  
    vector<PagePointer> pointers;  
}
```



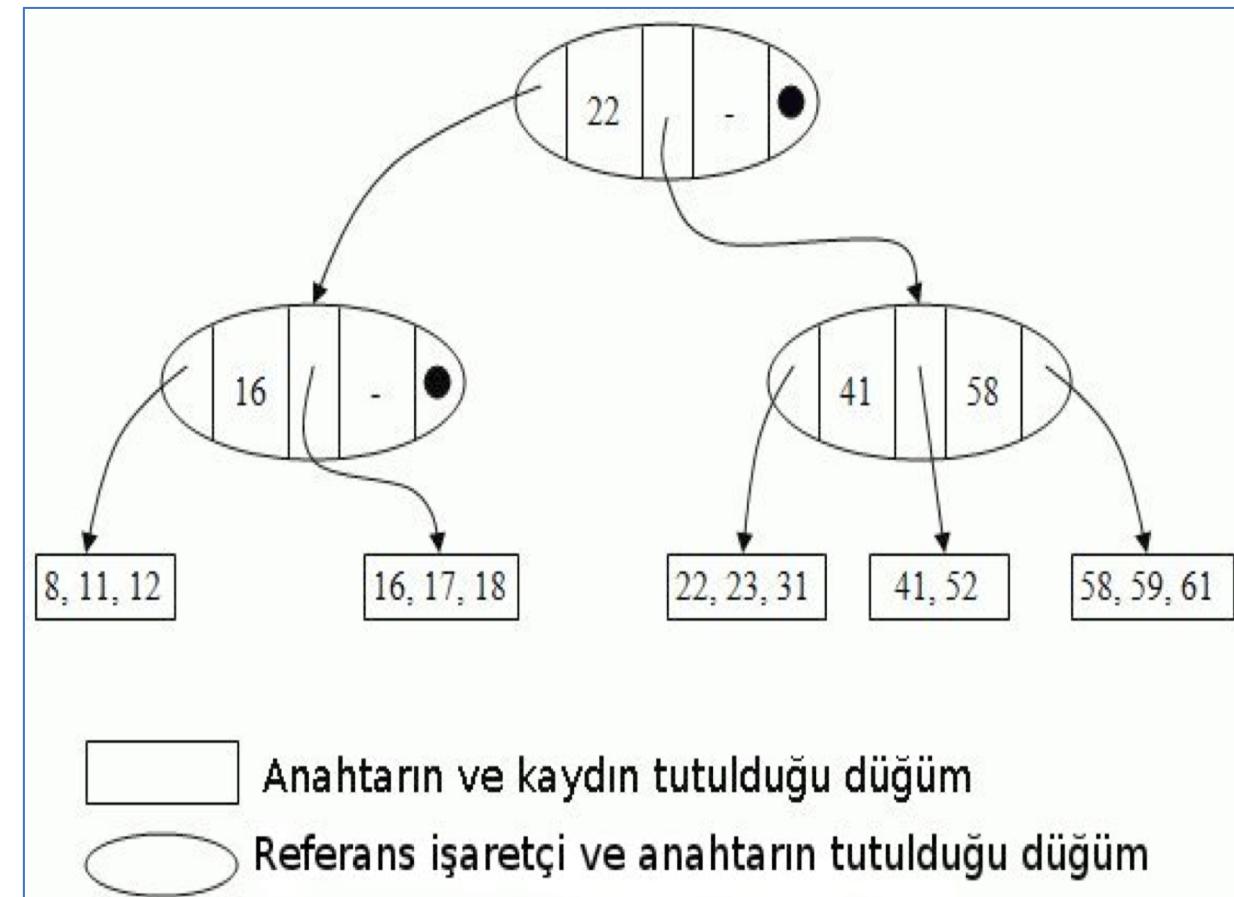
B+ Tree

- B+ ağacının arama (search) yapısının ara düğümleri (node) aramayı yönlendirmektedir.
- Leaf nodes (yaprak düğümleri) kaydedilen verileri içermektedir.
- B+ ağacının yapısı seçilen anahtara göre değişir.



B+ Tree

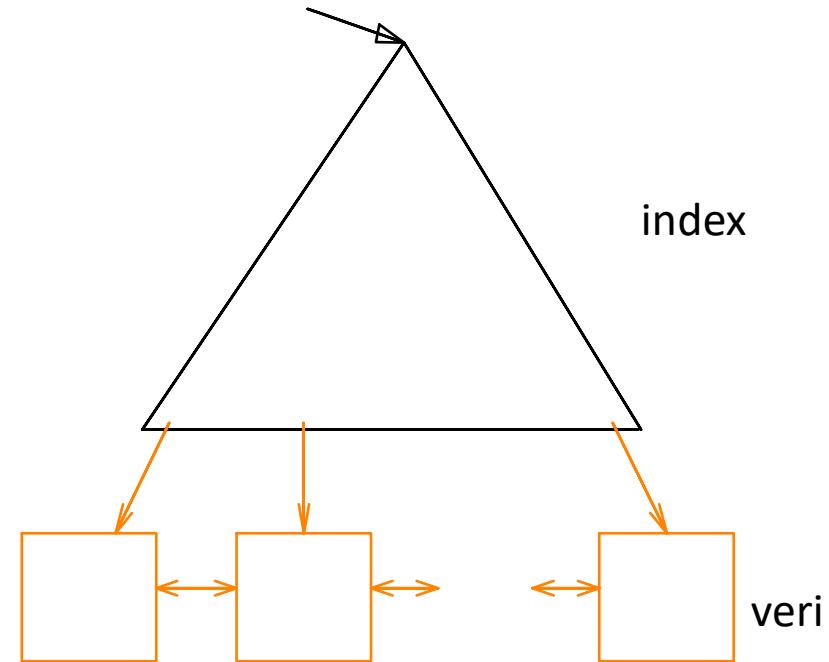
- B+ Tree dinamik olarak büyüyüp (grow) küçülmektedir (shrink) ve ağacın yüksekliği de dinamik olarak değişmektedir.
- Anahtarlar ve işaretçileri (pointer) yaprak olmayan (non leaf) düğümlerde tutulur.



<http://e-bergi.com/y/B+-Agaclari>

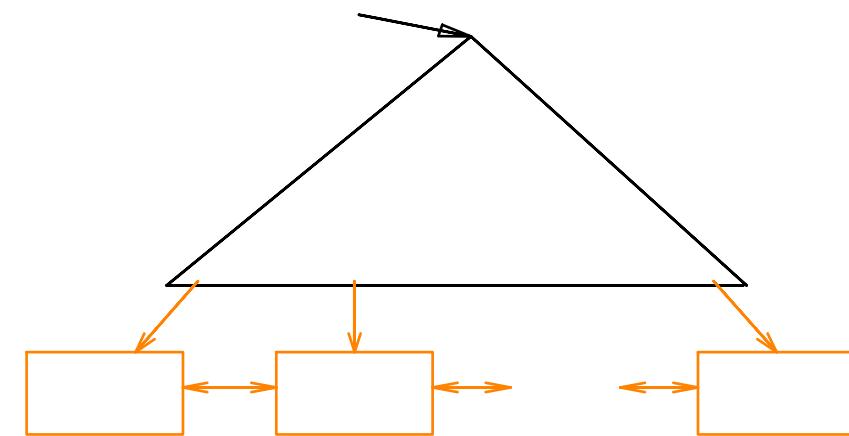
B+ Tree 'nin temel karakteristikleri

- Bir tane root (ana kök) düğüm bulunmak zorundadır.
- Her bir düğüm için ortama %50 doluluk oranı (root hariç) istenmektedir
- $d \leq \text{düğümde bulunabilecek anahtar sayısı} \leq 2d$
 - d parametresi (order of tree) bir düğümde bulunabilecek minimum kayıt sayısı.
 - d değeri sayfada tutulacak olan kayıtların boyutuna göre belirlenir
- B+ ağacının yüksekliği: $h = \log_k N$
 - k= düğümün kapasitesi , N= kayıt sayısı



B+ Tree (Pratikdeki değerler)

- Tipik order 100 ve doluluk oranı % 67
 - Düğümlerde bulunan ortalama alt düğüm sayısı (fanout) 133
- Kapasite
 - Yükseklik 4: $133^4 = 312,900,700$ kayıt
 - Yükseklik 3: $133^3 = 2,352,637$ kayıt
- Ara bellekte tutulan değerler :
 - Seviye 1 = 1 page = 8 Kbytes
 - Seviye 2 = 133 pages = 1 Mbyte
 - Seviye 3 = 17,689 pages = 133 MBytes



B+ Tree (search: arama)

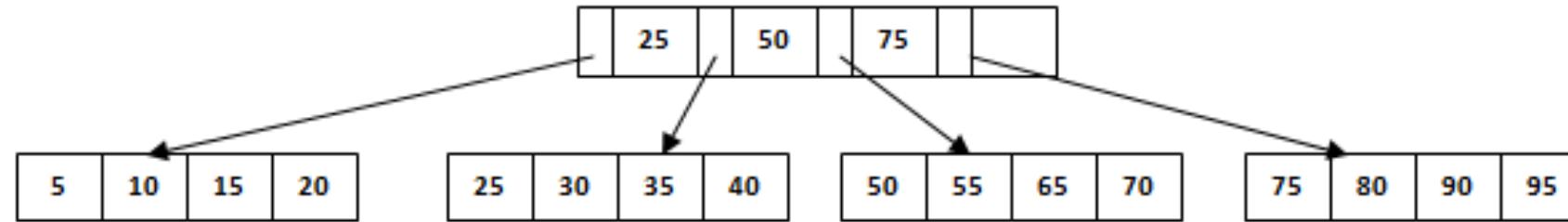
```
fune find (search key value  $K$ ) returns nodepointer  
    // Given a search key value, finds its leaf node  
    return tree_search(root,  $K$ );                                // searches from root  
endfune  
  
fune tree_search (nodepointer, search key value  $K$ ) returns nodepointer  
    // Searches tree for entry  
    if *nodepointer is a leaf, return nodepointer;  
    else,  
        if  $K < K_1$  then return tree_search( $P_0$ ,  $K$ );  
        else,  
            if  $K \geq K_m$  then return tree_search( $P_m$ ,  $K$ );    //  $l_1 = \#$  entries  
            else,  
                find  $i$  such that  $K_i \leq K < K_{i+1}$ ;  
                return tree_search( $P_i$ ,  $K$ )  
endfune
```

B+ Tree (arama)

1. Aramaya root (kök düğüm) ile başlanır
2. Arama yapılan düğüm (node) bir yaprak değilse;
 - Anahtar değerine göre düğümde arama yapılır
 - Aranılan anahtar değerden küçük olan en yüksek değerli anahtardan sonraki işaretleyicinin gösterdiği düğüme gidilir.
 - Arama yapılan düğüm güncellenir
 - 2.adıma geri dönülür.
3. Arama yapılan node (düğüm) bir yaprak ise aranılan deger bu düğümdedir veya yoktur.

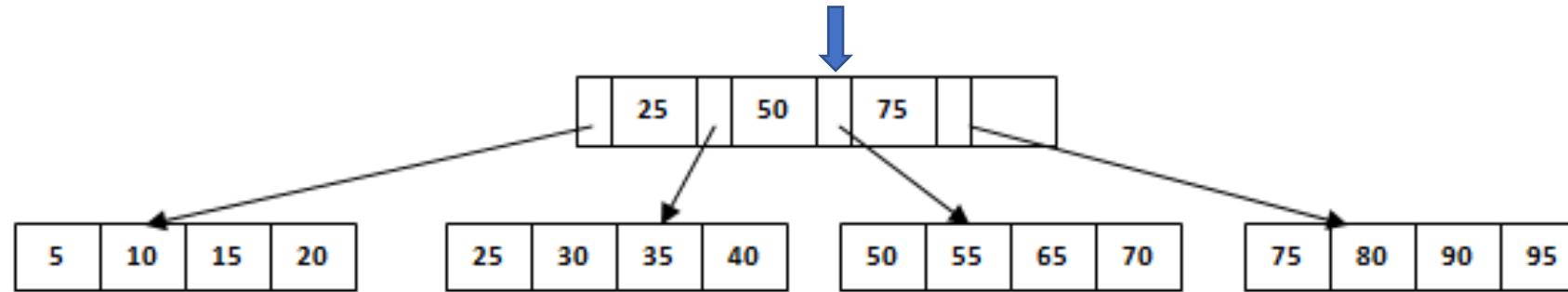
B+ Tree (search)

65 değerini arayalım



B+ Tree (search)

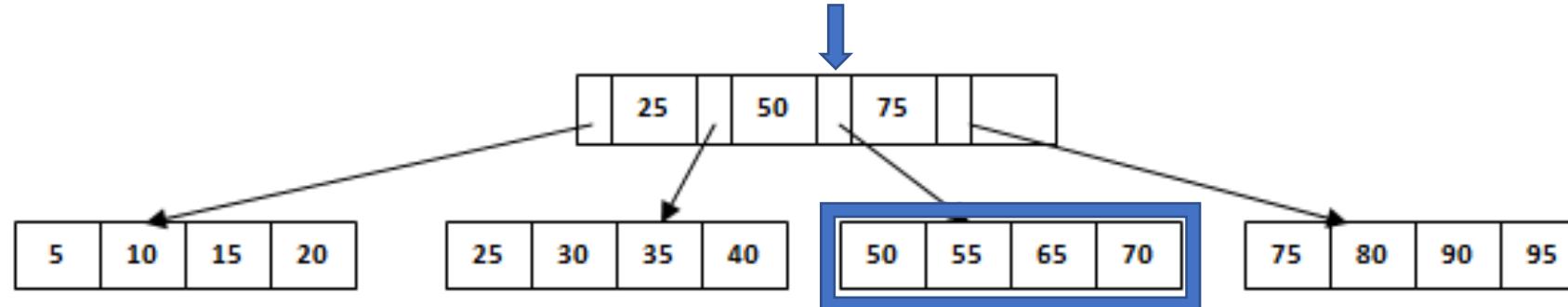
65 değerini arayalım



1. 65 değerinden küçük olan en büyük anahtardan sonraki pointera gidilir

B+ Tree (search)

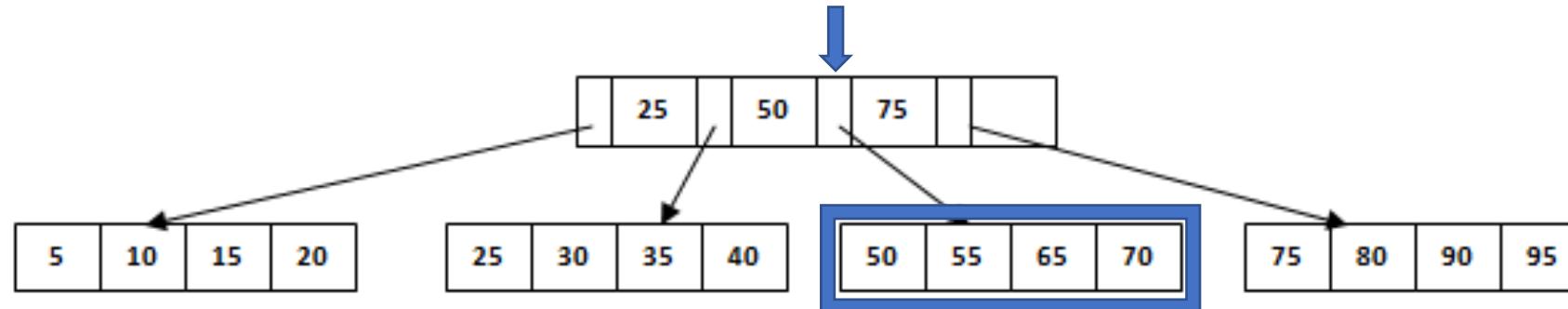
65 değerini arayalım



1. 65 değerinden küçük olan en büyük anahtardan sonraki pointera gidilir
2. Bulunan düğüm yaprak (leaf) olduğu için bu node içerisinde bulunan kayıtlar seri olarak (sequential) olarak taranır.

B+ Tree (search)

65 değerini arayalım



1. 65 değerinden küçük olan en büyük anahtardan sonraki pointera gidilir
2. Bulunan düğüm yaprak (leaf) olduğu için bu node içerisinde bulunan kayıtlar seri olarak (sequential) olarak taranır.

Arama (search) işlemi gerçekleştirilirken

insert, delete ve update

İşlemlerine izin verilmemektedir

B+ Tree (insert) : ekleme

- 1-Eklenmesi gereken kayıt için (düğüm) araması yapılır
- 2-Eğer düğüm maximum kaydedebileceği değerden az kayıt bulunduruyorsa
 - kayıt sırası dikkate alınarak ekleme yapılır (işlem tamamlanır)
- 3- Eklenecek değer için düğümde yer yoksa
 - Düğüm ikiye ayrılır, yeni bir yaprak (leaf) oluşturulur ve elemanların yarısı sırası değişmeden oluşturulan yeni yaprağa eklenir.
 - Yeni oluşturulan düğümün anahtarı bir üst seviyedeki düğüme eklenir
 - Üst seviyedeki düğüm dolu ise ikiye ayrılır ve ortadaki anahtar üst seviyeye alınır.
 - Bölünme gerektirmeyen düğüm oluşuncaya kadar devam eder.
- 4- Eğer root (kök) düğümünün bölünmesi gerekiyorsa tek bir değeri ve iki pointeri bulunan yeni bir root oluşturulur.

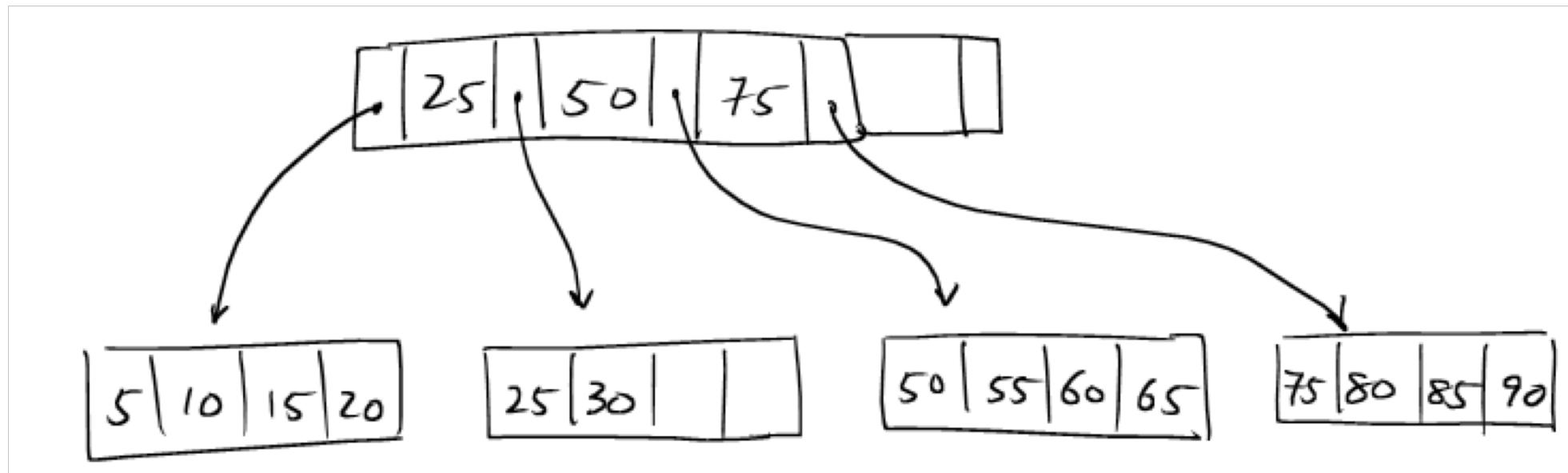
B+ Tree (insert) : ekleme

B+ tree düğümlerinin ki anahtar sayısı
 $d \leq \text{anahtar sayısı} \leq 2d$

Root (kök) düğüm istisnadır. Kökte bulunan
anahtar sayısı
 $1 \leq \text{anahtar sayısı} \leq 2d$

B+ Tree (insert) : ekleme

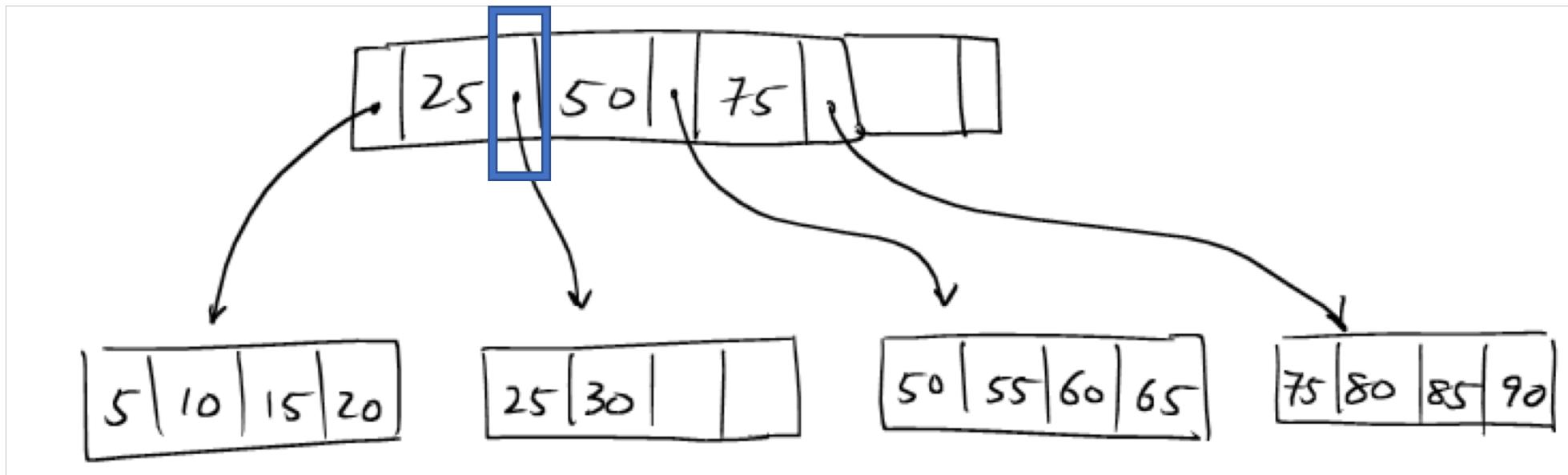
28 değerini ekleyelim



B+ Tree (insert) : ekleme

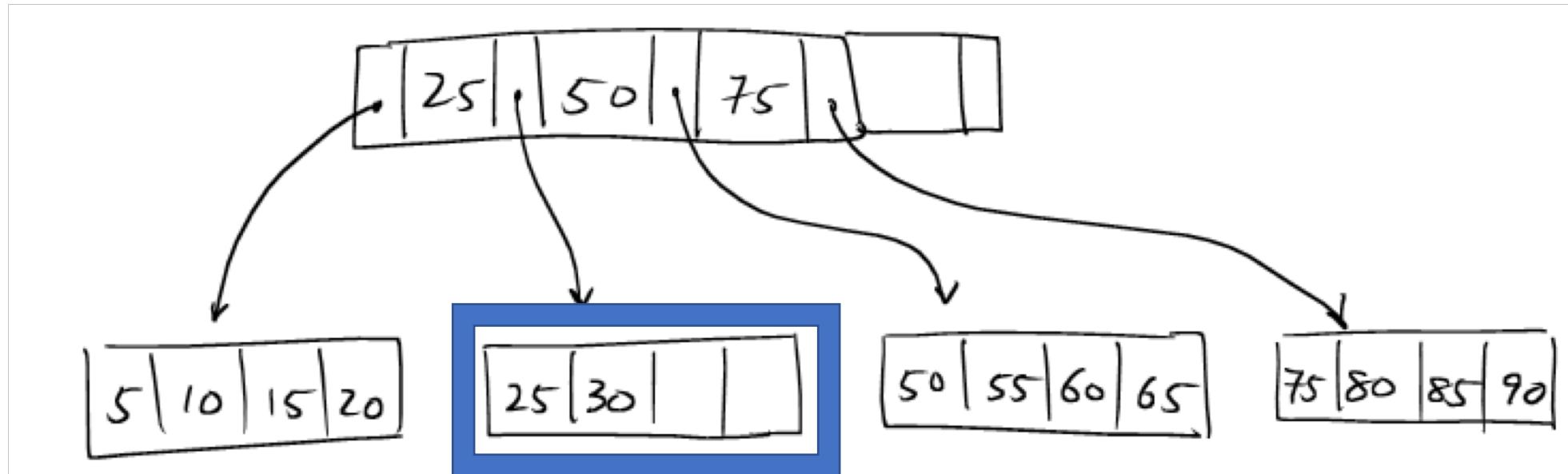
Search (root, 28)

Ekleyeceginiz degerden kucuk olan en büyük degerden sonraki pointeri bul



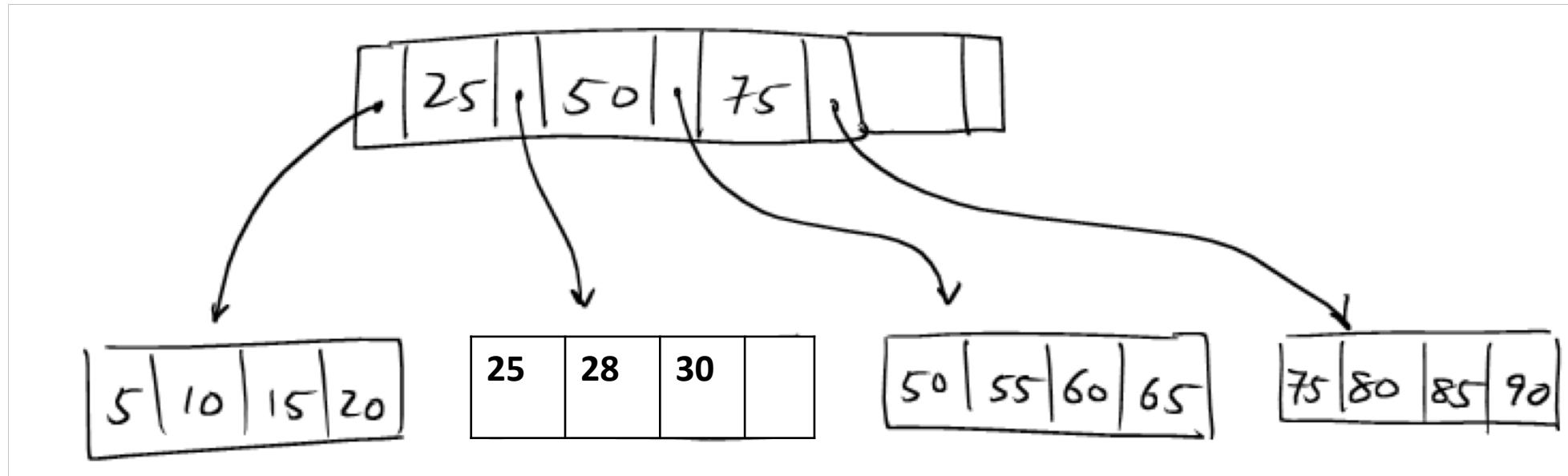
B+ Tree (insert) : ekleme

Search (root, 28)



Yaprak da veri eklenecek alan var !

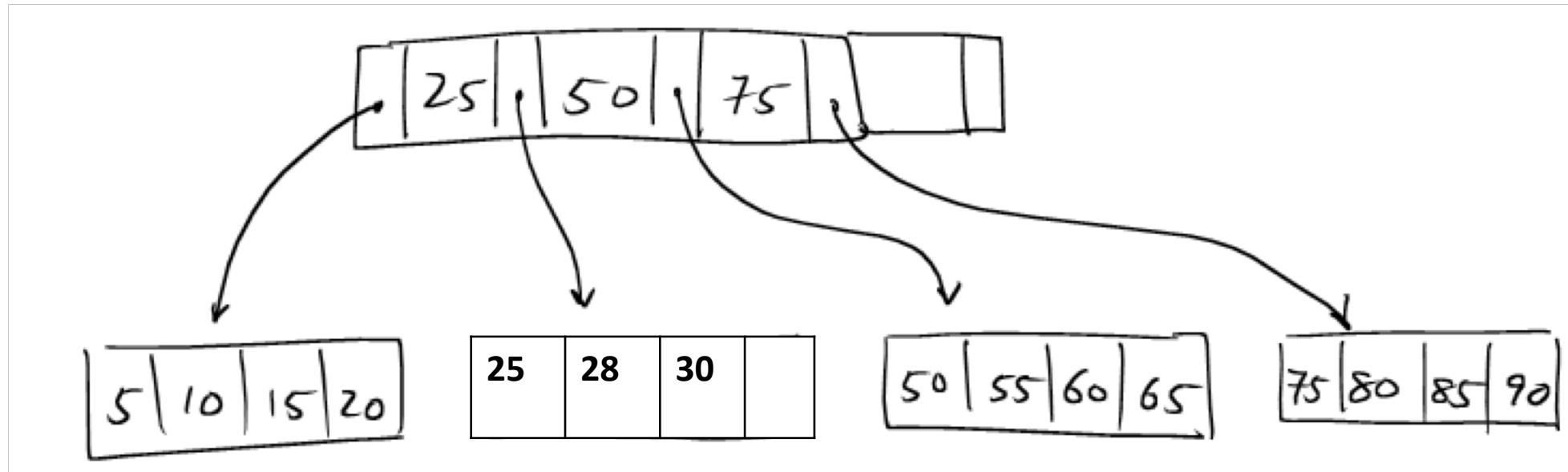
B+ Tree (insert) : ekleme



Yaprak düğümde bulunan değerler
kendi içinde sıralandı

B+ Tree (insert) : ekleme

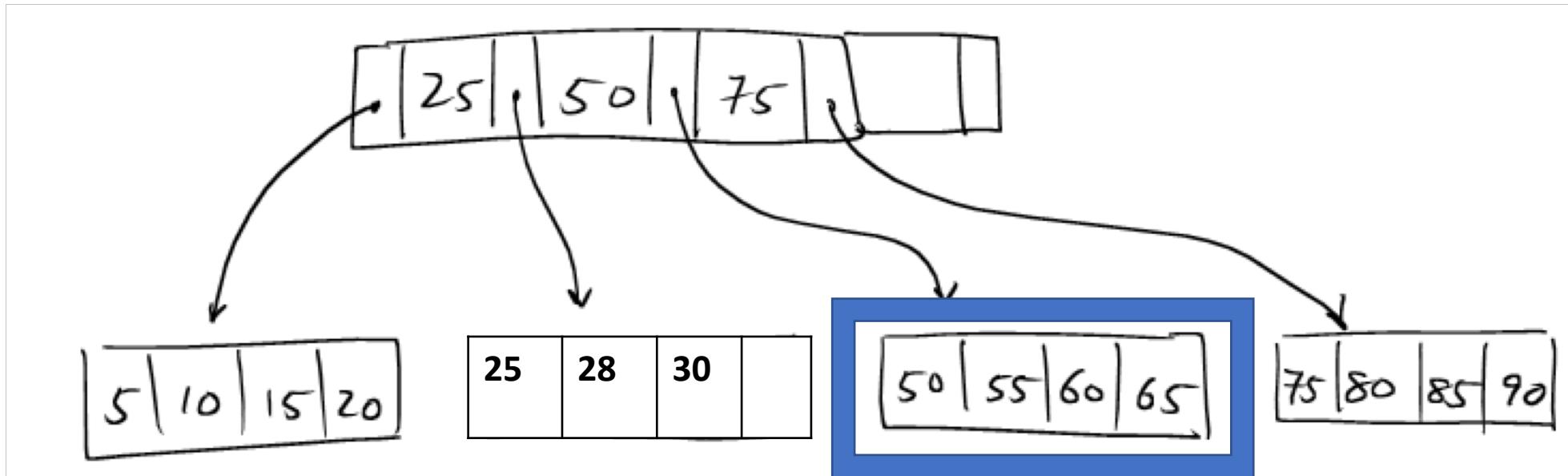
70 değerini ekleyelim



B+ Tree (insert) : ekleme

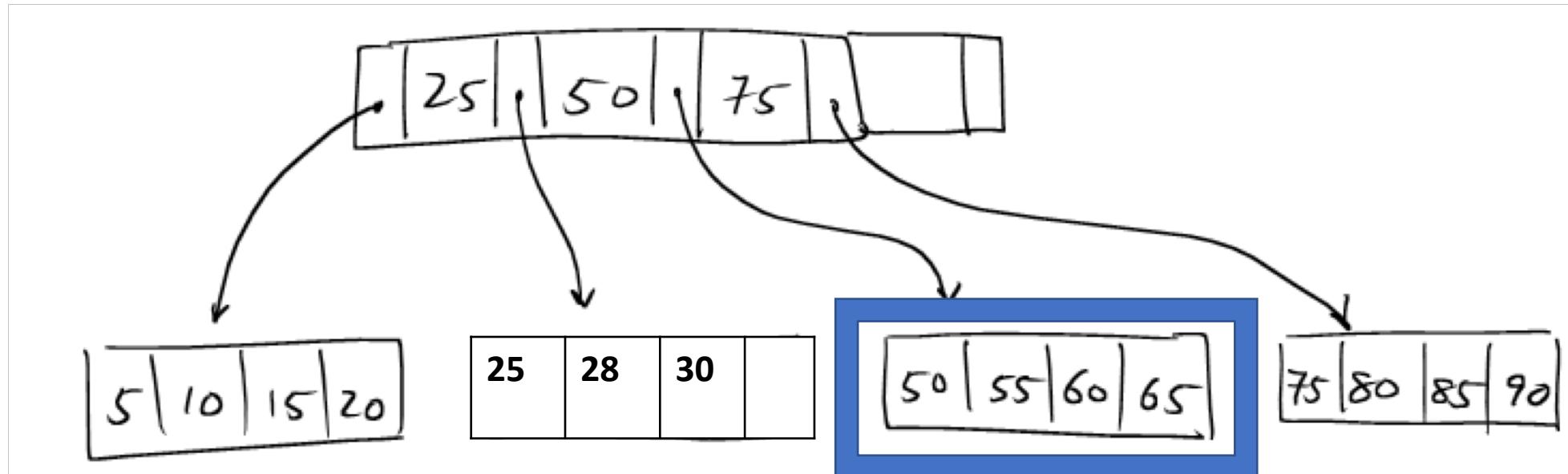
70 değerini ekleyelim

Ekleyeceginiz degerden kucuk olan en büyük degerden sonraki pointeri bul



Değerin eklenmesi gereken düğüm bulunur.

B+ Tree (insert) : ekleme

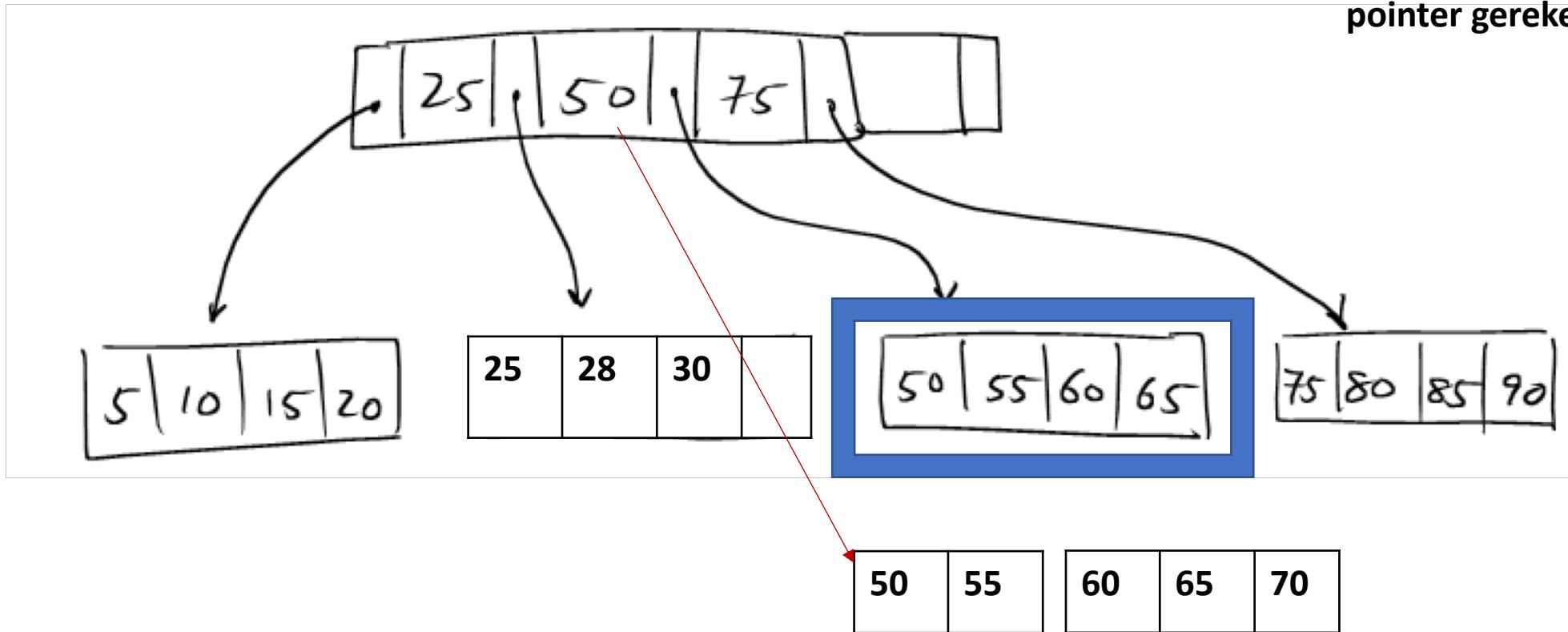


Değerin eklenmesi gereken düğüm dolu olduğundan düğüm ikiye bölünür

50	55	60	65	70
----	----	----	----	----

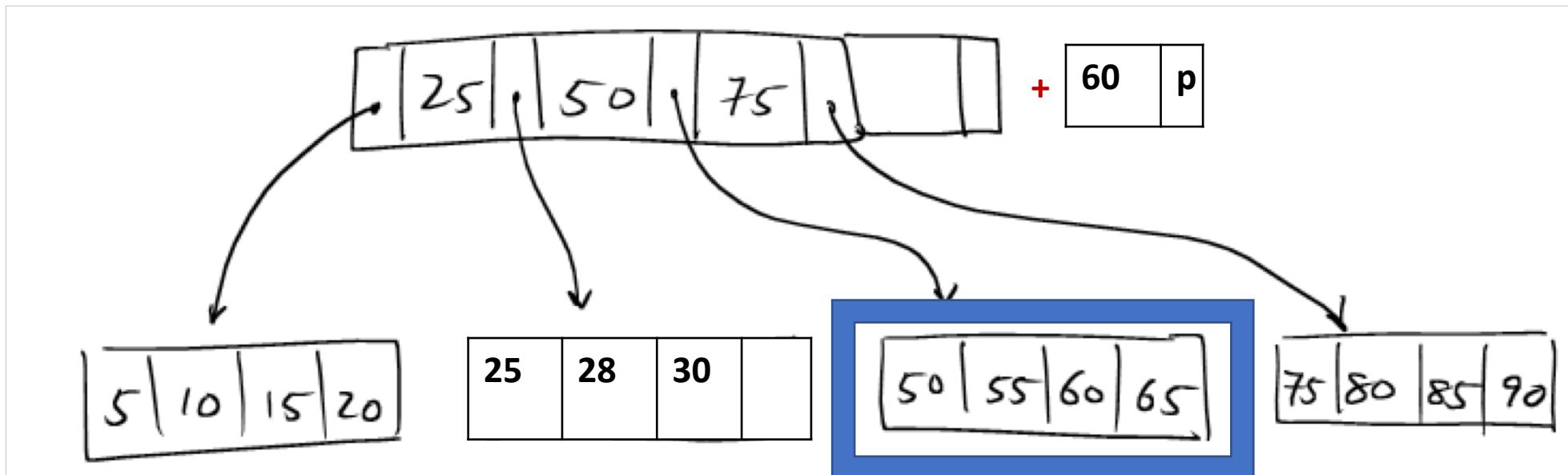
B+ Tree (insert) : ekleme

Oluşturulan yeni düğüm için bir üst seviyede anahtar ve pointer gerekecek



B+ Tree (insert) : ekleme

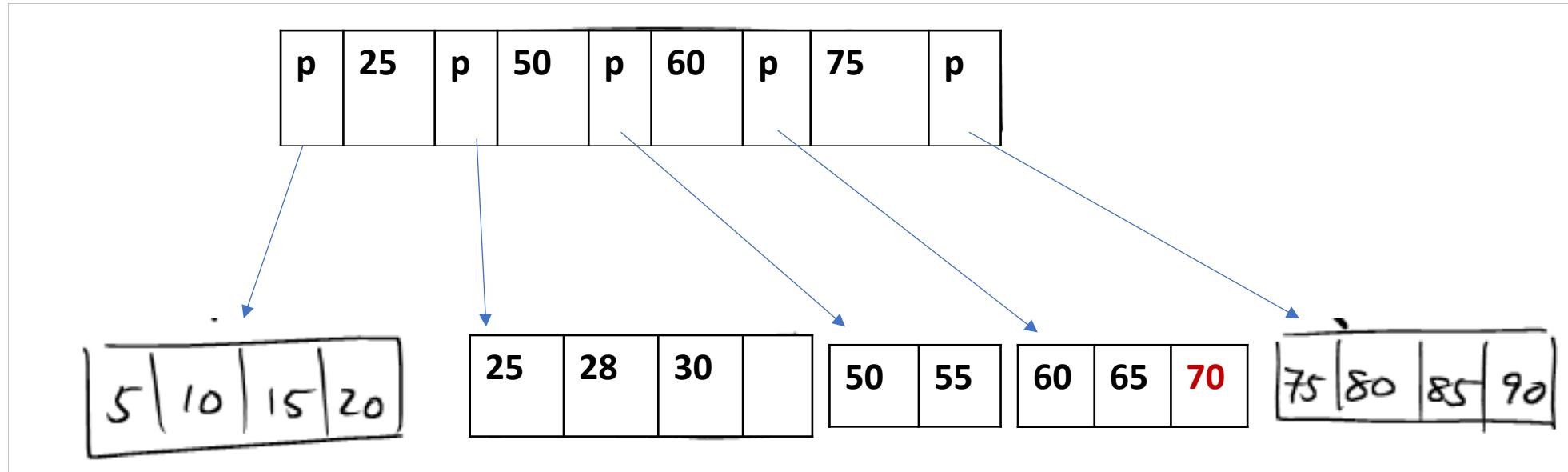
Oluşturulan yeni düğüm anahtarı (60) ve pointeri eklendi



50	55
60	65

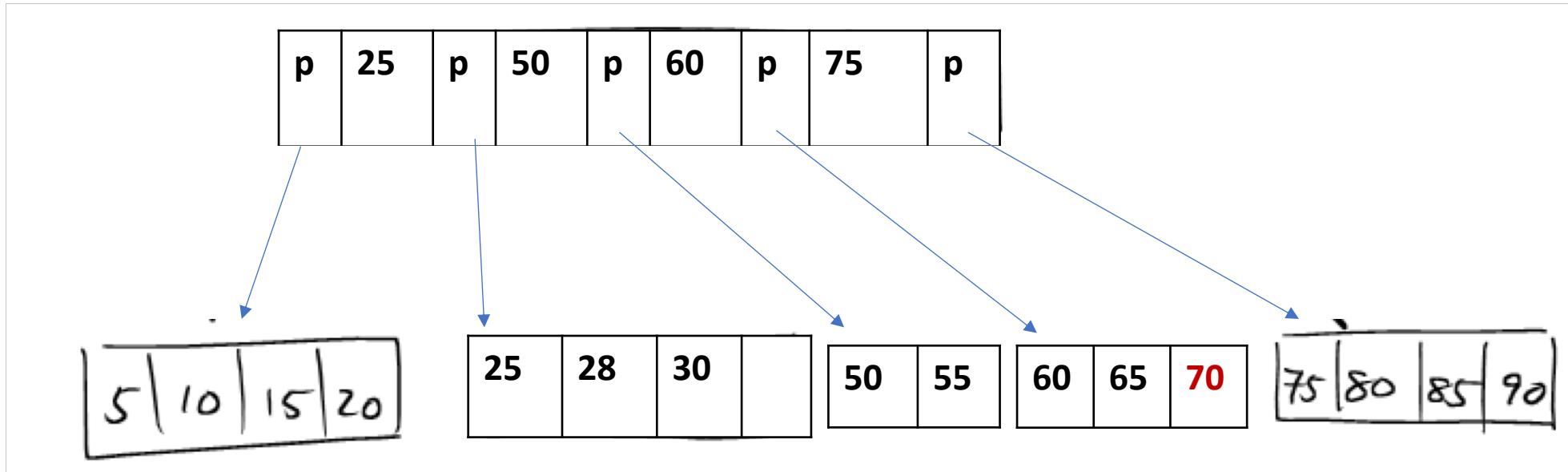
B+ Tree (insert) : ekleme

70 değerini ekledikten sonra yeni ağaç



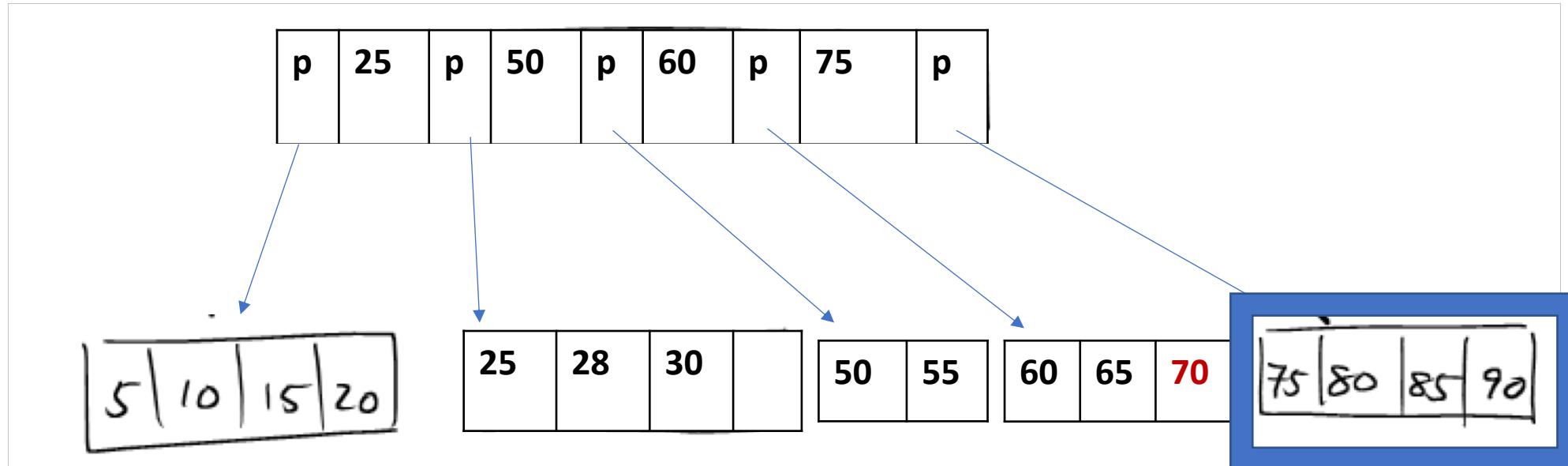
B+ Tree (insert) : ekleme

Yeni ağaca 95 değerini ekleyelim



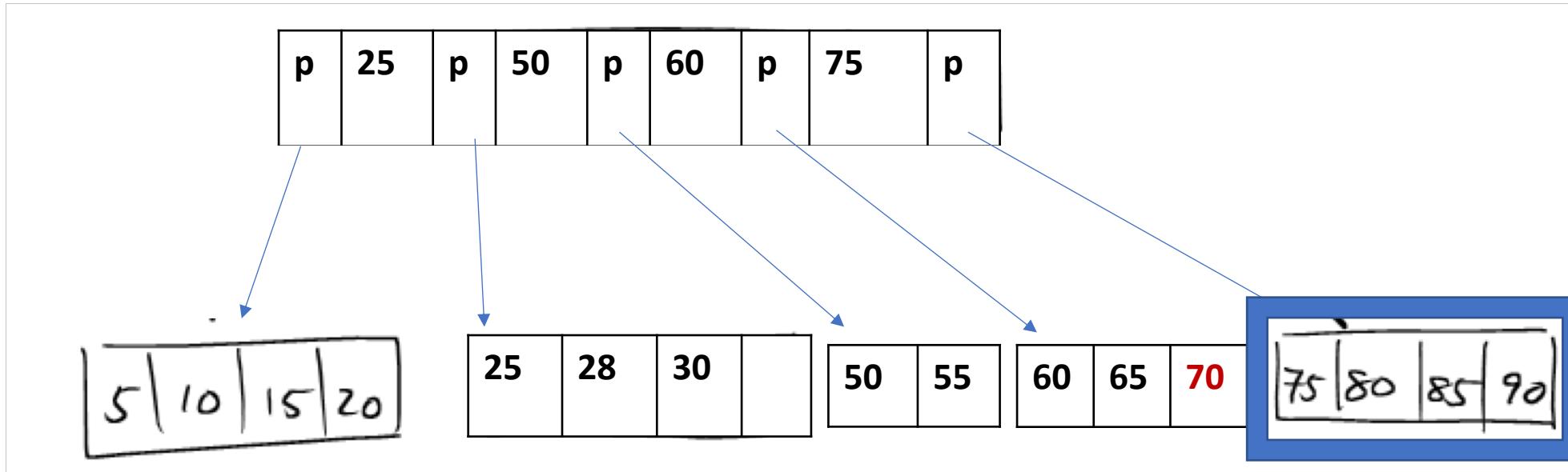
B+ Tree (insert) : ekleme

95 değerini ekleyelim



B+ Tree (insert) : ekleme

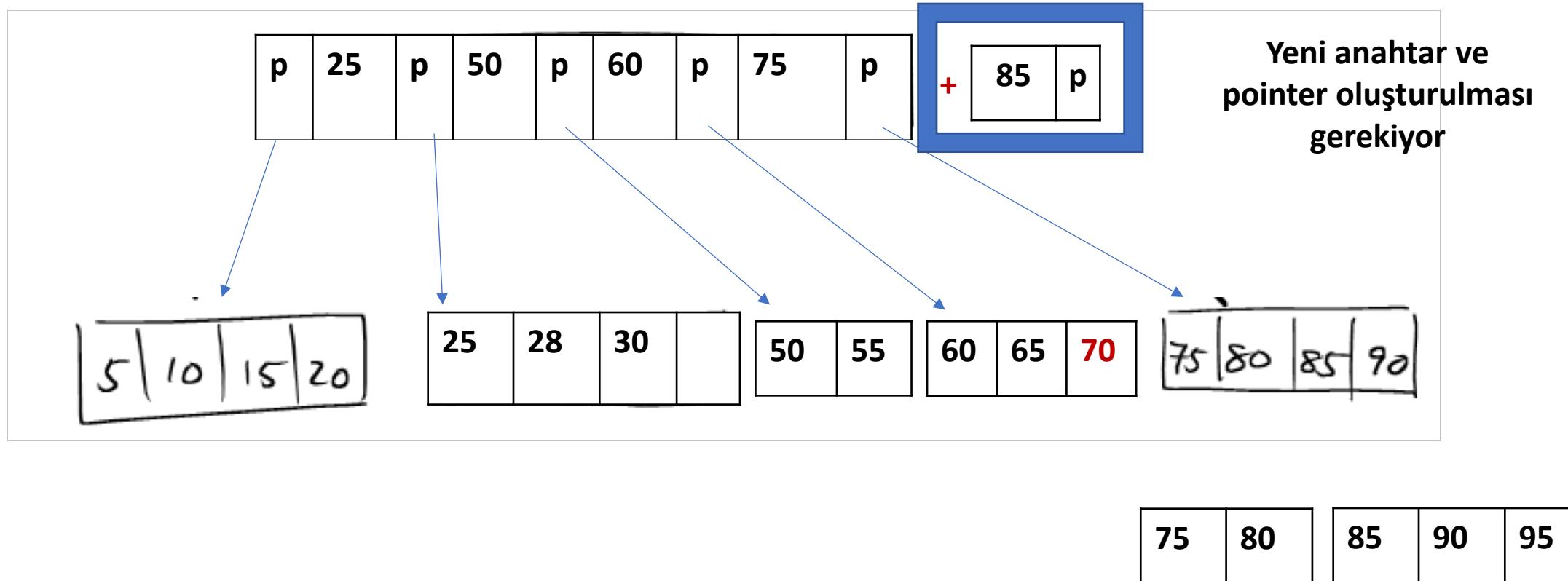
95 değerini ekleyelim



75	80	85	90	95
----	----	----	----	----

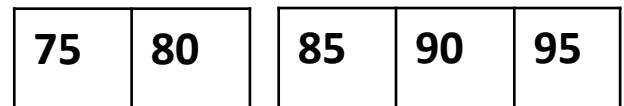
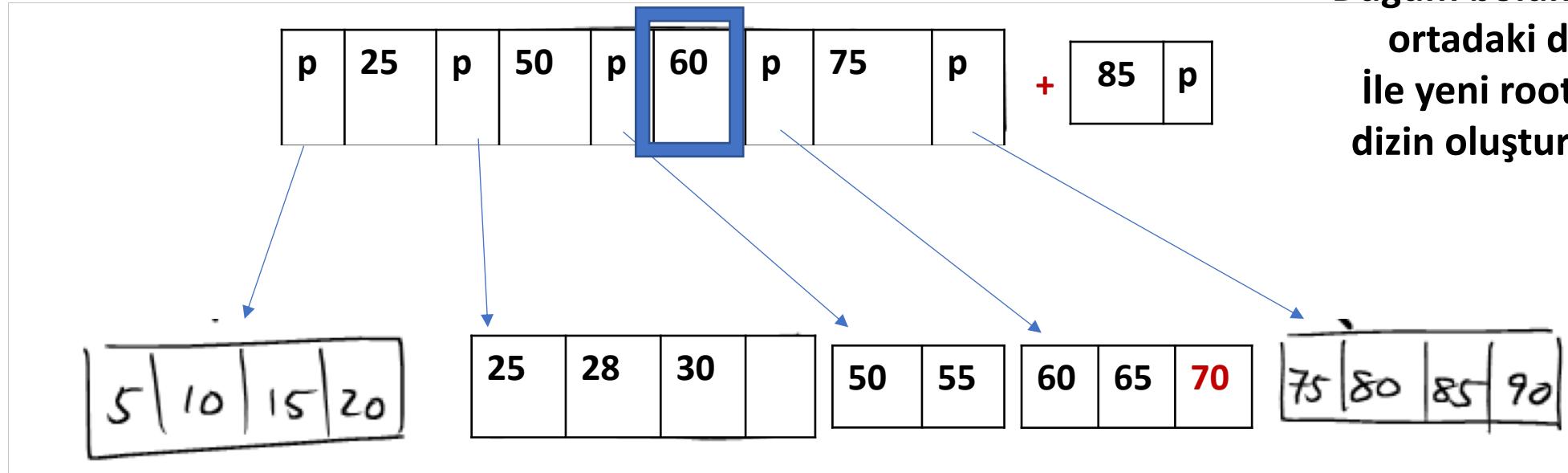
B+ Tree (insert) : ekleme

95 değerini ekleyelim



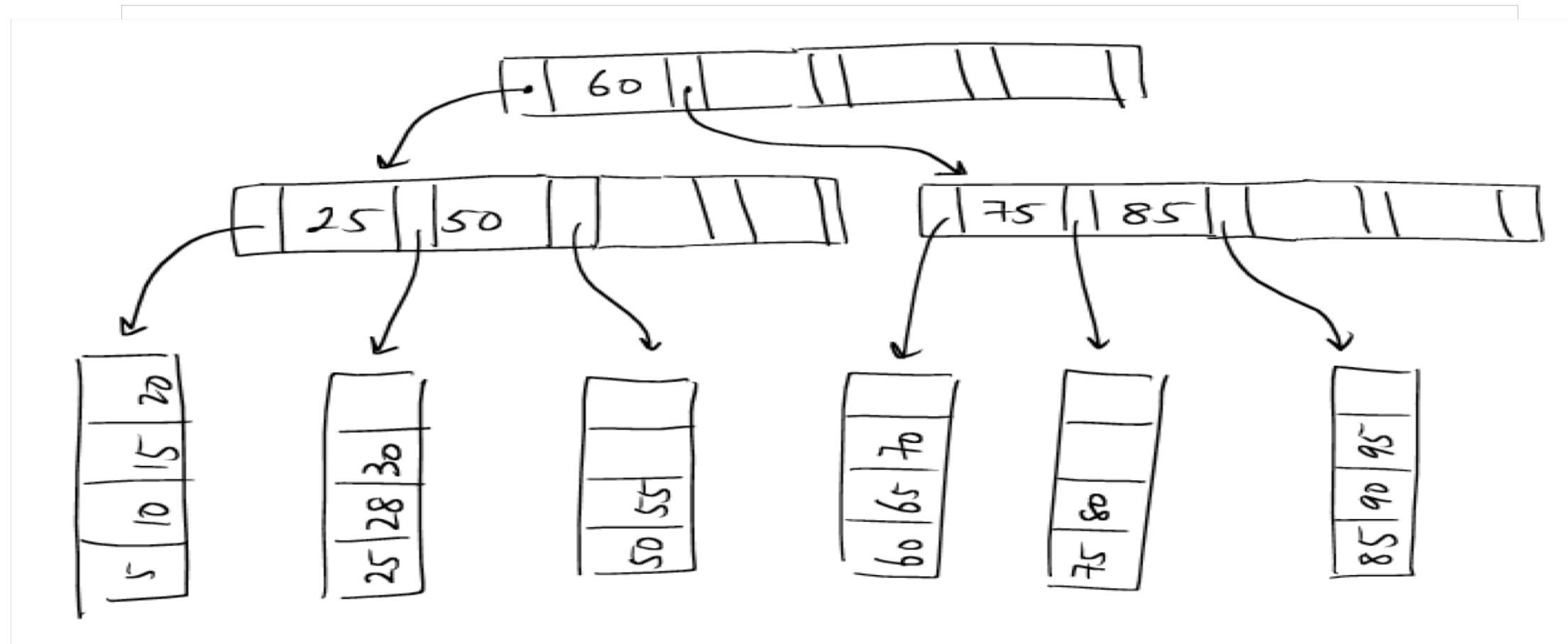
B+ Tree (insert) : ekleme

95 değerini ekleyelim



B+ Tree (insert) : ekleme

95 değerini ekleyelim



B+ Tree (delete)

1. Silinecek kayıtın yaprak düğümü anahtar değeri yardımıyla bulunur.
2. Silme işleminin gerçekleştirileceği düğümde ağacın derecesine eşit veya büyük sayıda kayıt varsa silme işlemi gerçekleştirilir.
3. $d \leq$ düğümdeki kayıt sayısı $\leq 2d$ şartını sağlamıyorsa düğümler aynı parent(ebeveyn) a sahip düğümler birleştirilir (tekrar dağıtma işlemi)
4. Birleştirme işlemi gerçekleştirilirken bir üst seviyede (ebeveyn) sağdaki düğümü gösteren pointer ve anahtarı silinir
5. Bu işlem root (kök) a kadar devam eder.