Veritabanı Yönetim Sistemleri (335)

Yrd.Doç.Dr. Ahmet Arif AYDIN

Özet

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

B+ Tree

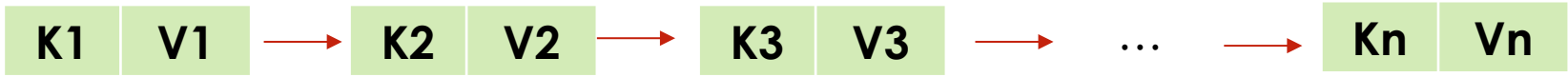
- Esnek ve dinamik bir yapısı bulunmaktadır.
- Her bir düğüm disk üzerinde bir sayfadır.
- İki adet düğüm çeşidi bulunmaktadır:
    1. Leaf nodes (Yaprak Düğümler)
    2. Diğer düğümler

B+ Tree

1- Leaf nodes (yaprak düğümler)

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

Verinin kaydedildiği sayfalar birbirine baglı Linked ile tanımlanabilir.

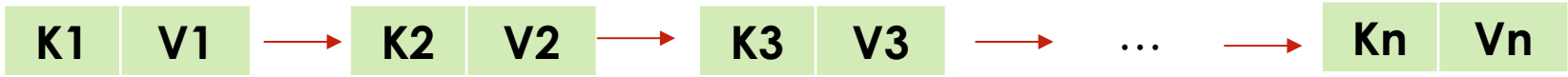


B+ Tree

1- Leaf nodes (yaprak düğümler)

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

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

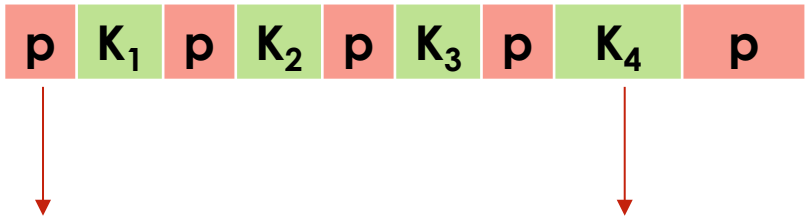


B+ Tree

2- Non-leaf (yaprak olmayan) düğümler

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

Pointer sayısı anahtar sayısından 1 fazla

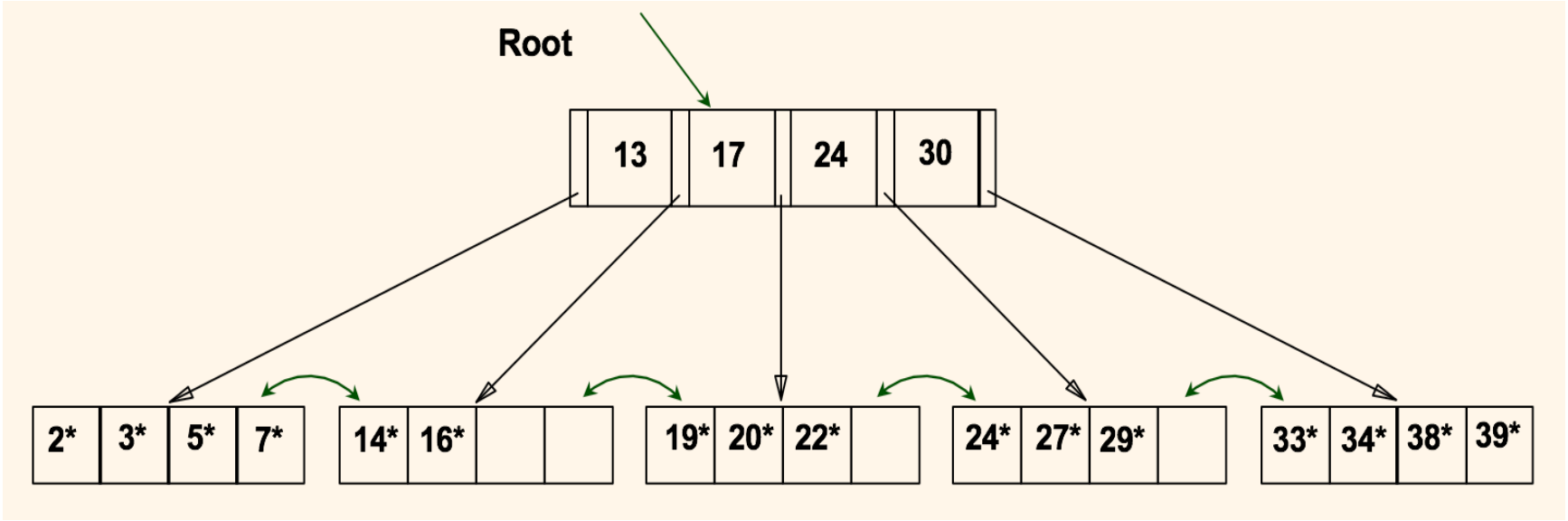


page (sayfa) pointeri

Anahtar

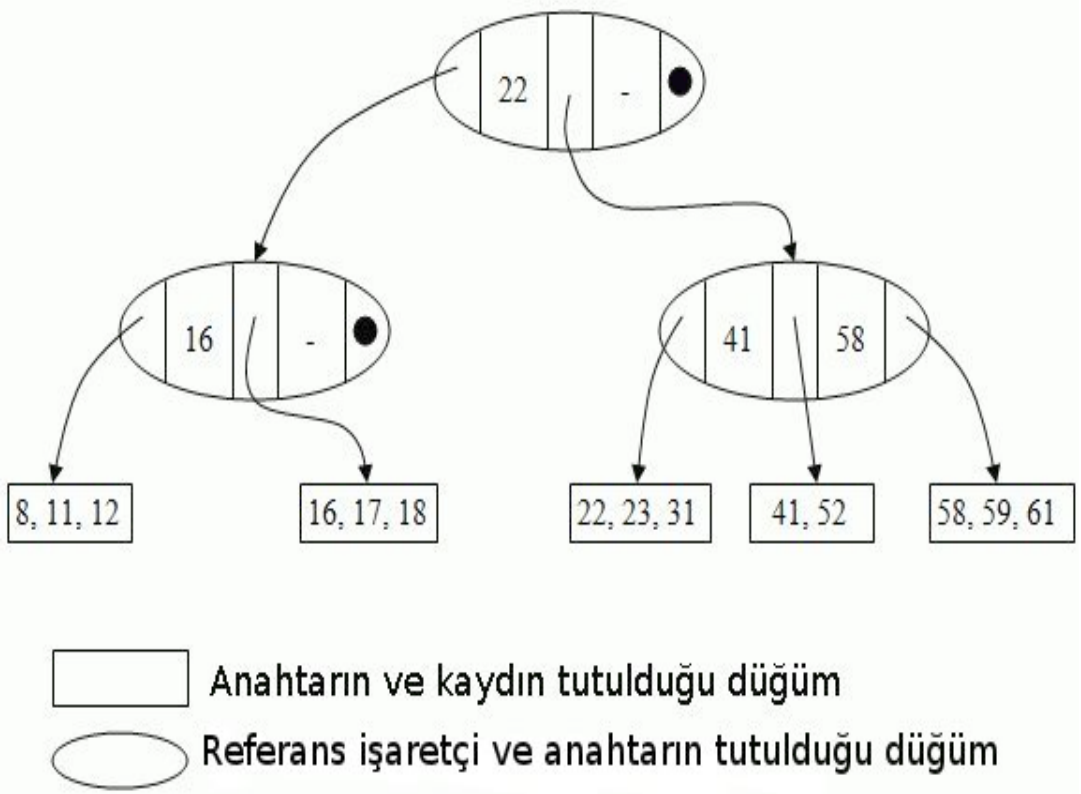
B+ Tree

- 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.

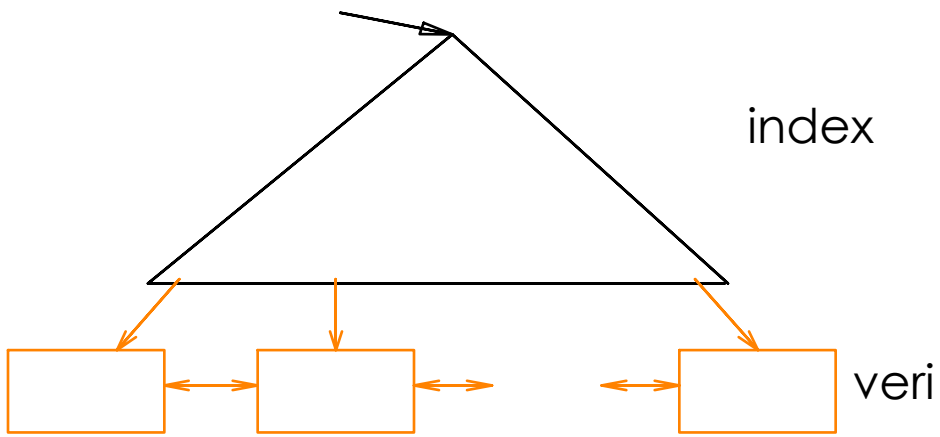


http://e-bergi.com/y/B+-Agaclari

B+ Tree

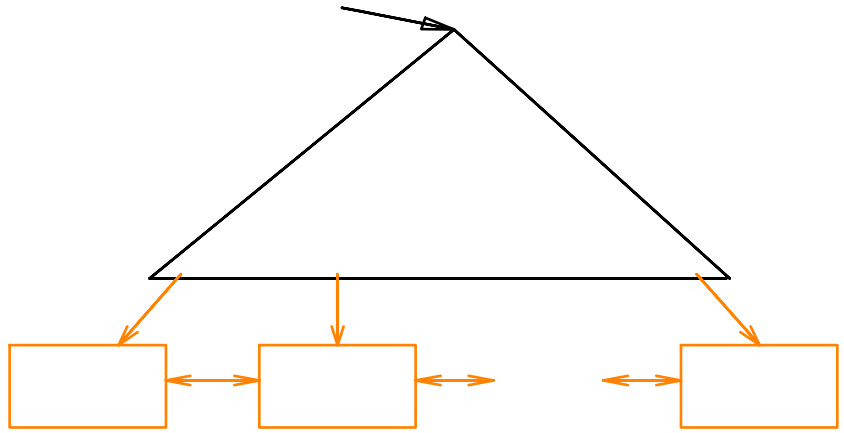
B+ Tree' nin ana karakteristikleri

- Bir tane root (ana kök) düğüm bulunmak zorundadır.
- Her bir düğüm için ortama %50 doluluk oranı (root hariç) istenmektedir
- **d <= düğümde bulunabilecek anahtar sayısı <= 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 (Pratikde ki değerler)

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



B+ Tree (arama)

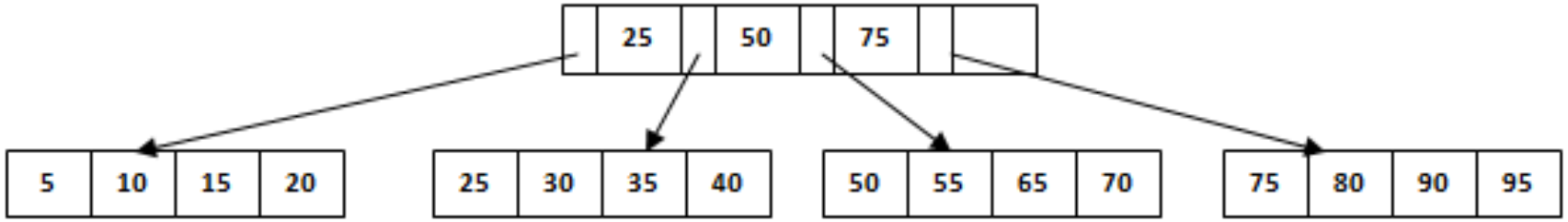
```
fune find (search key value K) returns nodepointer
// Given a search key value, finds its leaf node
return tree_search(root, K);                    // searches from root
endfune

fune tree_search (nodepointer, search key value K) returns nodepointer
// Searches tree for entry
if *nodepointer is a leaf, return nodepointer;
else,
      if K < K1 then return tree_search(P0, K);
      else,
            if K ≥ Km then return tree_search(Pm , K);     // 171 = # entries
            else,
                  find i such that Ki ≤ K < Ki+1;
                  return tree_search(Pi , K)
endfune
```

B+ Tree (arama)

1. Aramaya root (kök düğüm) ile başlanır
2. Arama yapılan node (düğüm) 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 guncellenir
   - 2.adıma geri dönülür.
3. Arama yapılan node (düğüm) bir yaprak ise **aranılan deger bu düğümdedir veya yoktur**.

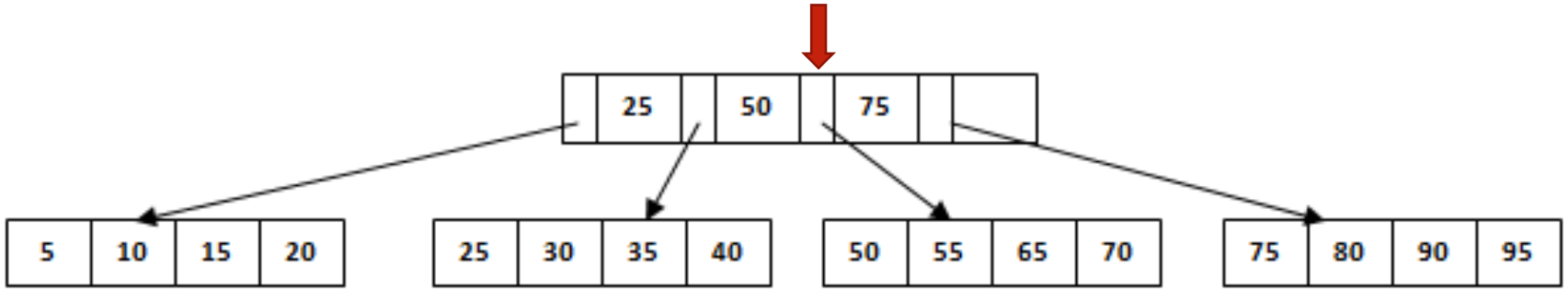
B+ Tree (search)



65 değerini arayalım

https://www.tutorialcup.com/dbms/b-tree.htm

B+ Tree (search)

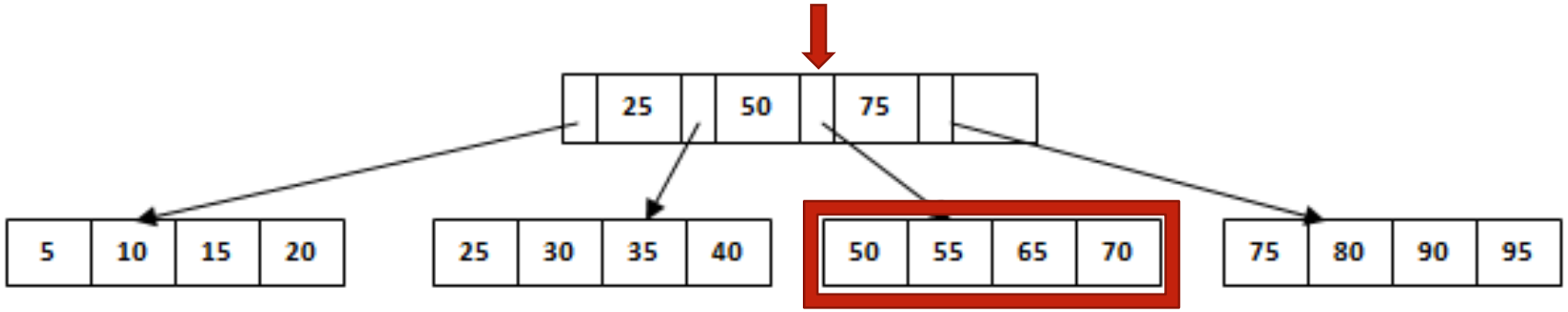


65 değerini bulmak için

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

https://www.tutorialcup.com/dbms/b-tree.htm

B+ Tree (search)

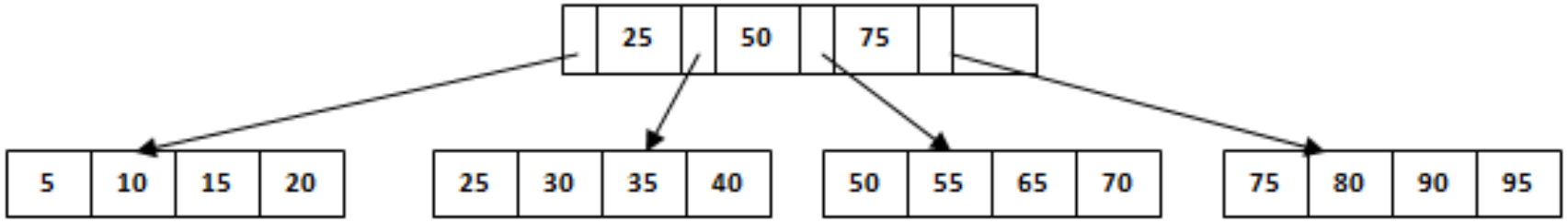


65 değerini bulmak için

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

https://www.tutorialcup.com/dbms/b-tree.htm

B+ Tree (search)



65 değerini bulmak için

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

Arama (search) işlemi gerçekleştirilirken *insert, delete ve update* İşlemlerine izin verilmemektedir

https://www.tutorialcup.com/dbms/b-tree.htm

B+ Tree (insert)

B+ tree düğümlerin de ki anahtar sayisi
d<= anahtar sayısı <= 2d
Root (kök) düğüm istisnadır. Kökte bulunan anahtar sayisi
1<= anahtar sayısı <= 2d

1-Eklenecek olan kayıtın bulunması gereken yer (düğüm) bulunur.

2-Eğer düğüm maxımum kaydedebileceği degerden az kayıt bulunduruyorsa

- kayıt sırası dikkate alınarak ekleme yapılır (işlem tamamlanır)

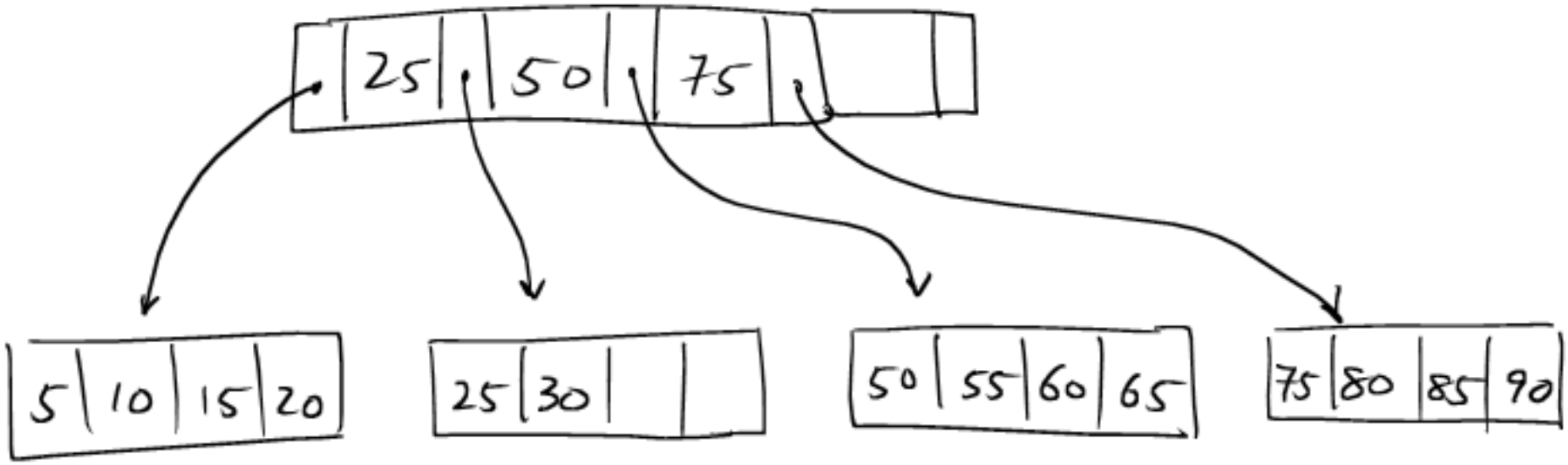
3- Eklenecek deger için yer yoksa

- Düğüm ikiye ayrılır
- Yeni bir yaprak (leaf) oluşturulur ve elemanların yarısı sırası degişmeden oluşturulan yaprağa eklenir.
- Yeni oluşturulan düğümün anahtarı bir üst seviyedeki düğüme eklenir
- Üst seviyedeki düğüm dolu ise ikiye ayrılır ve ortadaki anahtar ust seviyeye alınır.
- Bölünme gerektirmeyen düğüm oluşuncaya kadar devam eder.

4- Eğer root (kök) düğümünün bölünmesi gerekiyorsa tek bir degeri ve iki pointeri bulunan yeni bir root oluşturulur.

B+ Tree (insert)

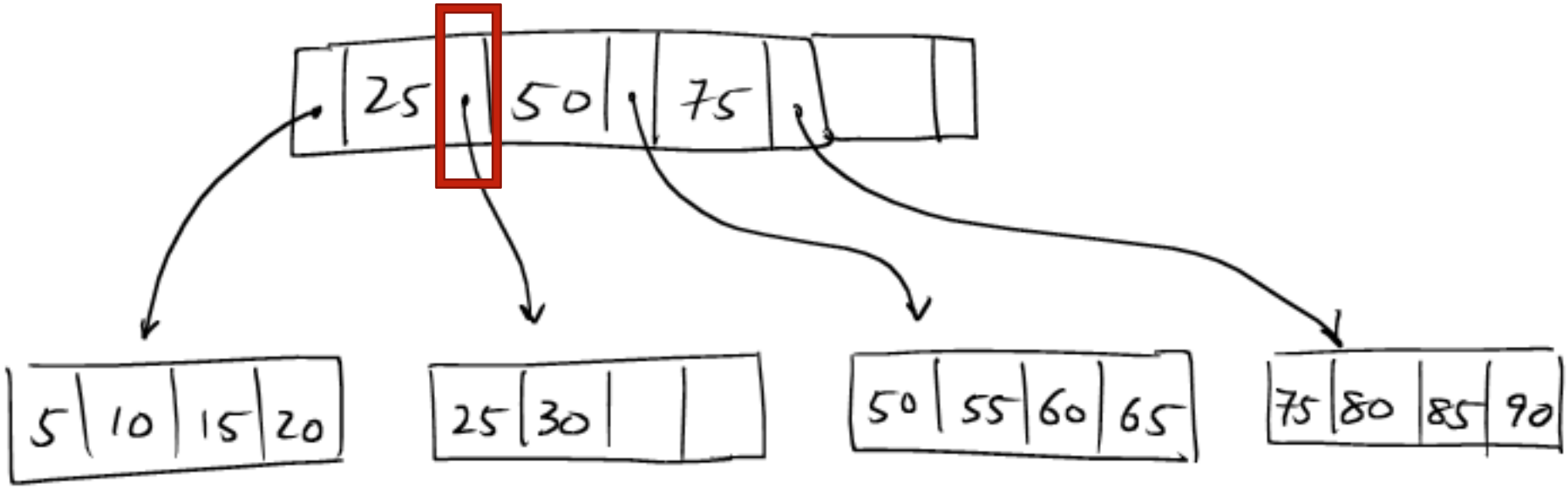
28 değerini ekleyelim



http://www.teach.cs.toronto.edu/~csc443h/fall/posted_practice/btree-index.html

B+ Tree (insert)

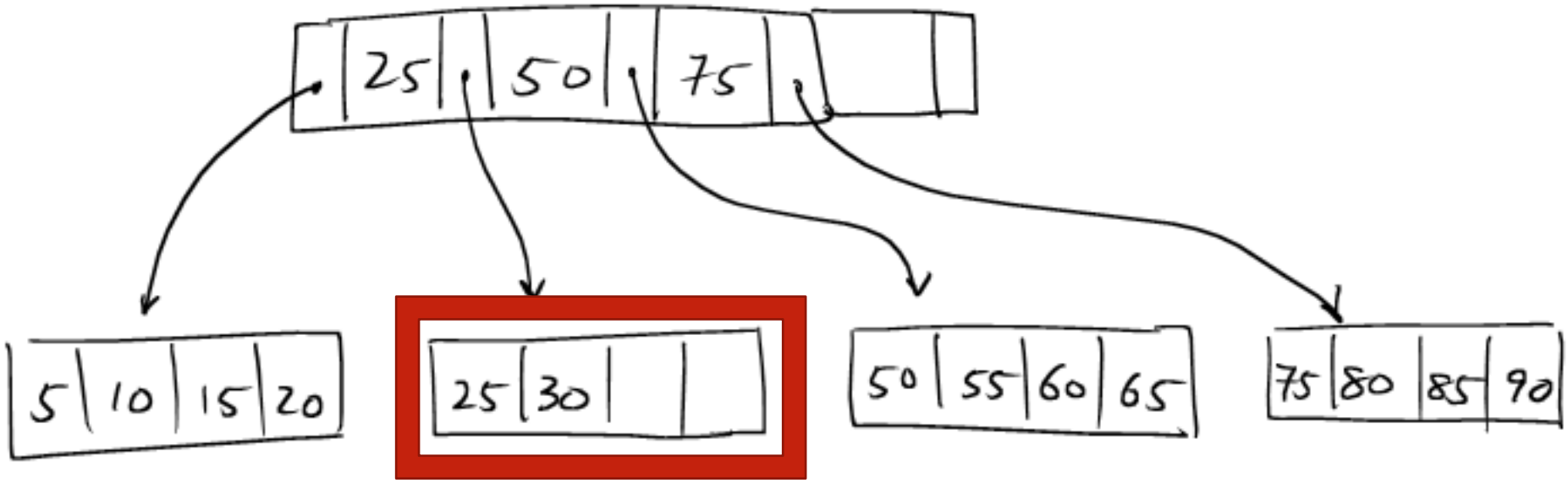
Search (root, 28)



http://www.teach.cs.toronto.edu/~csc443h/fall/posted_practice/btree-index.html

B+ Tree (insert)

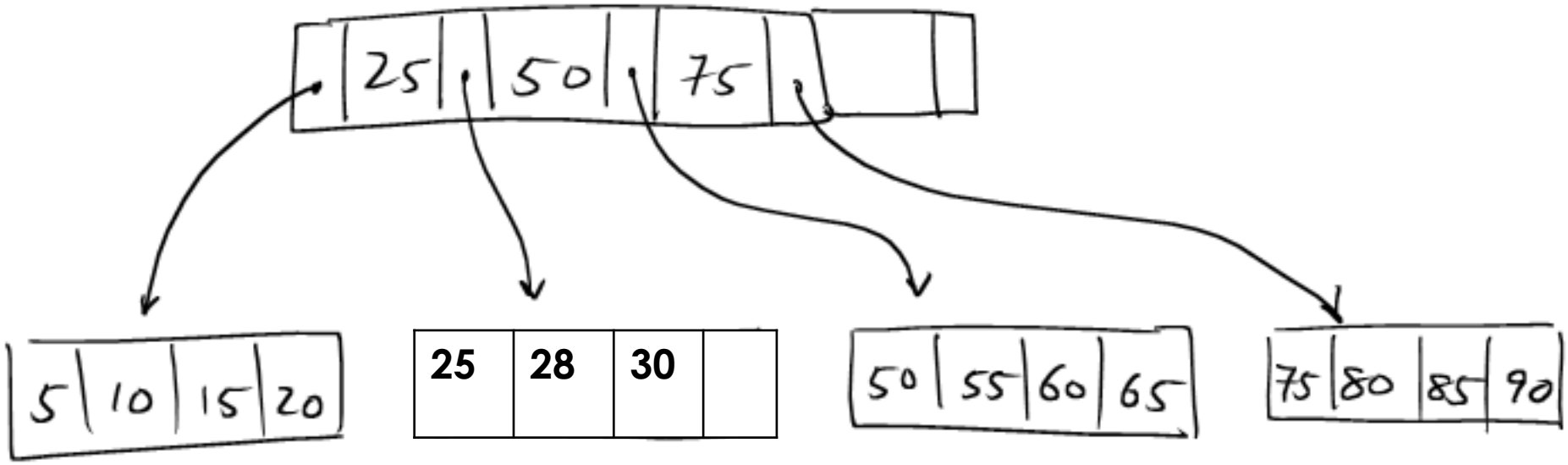
Search (root, 28)



Yaprak da veri eklenecek alan var !

http://www.teach.cs.toronto.edu/~csc443h/fall/posted_practice/btree-index.html

B+ Tree (insert)

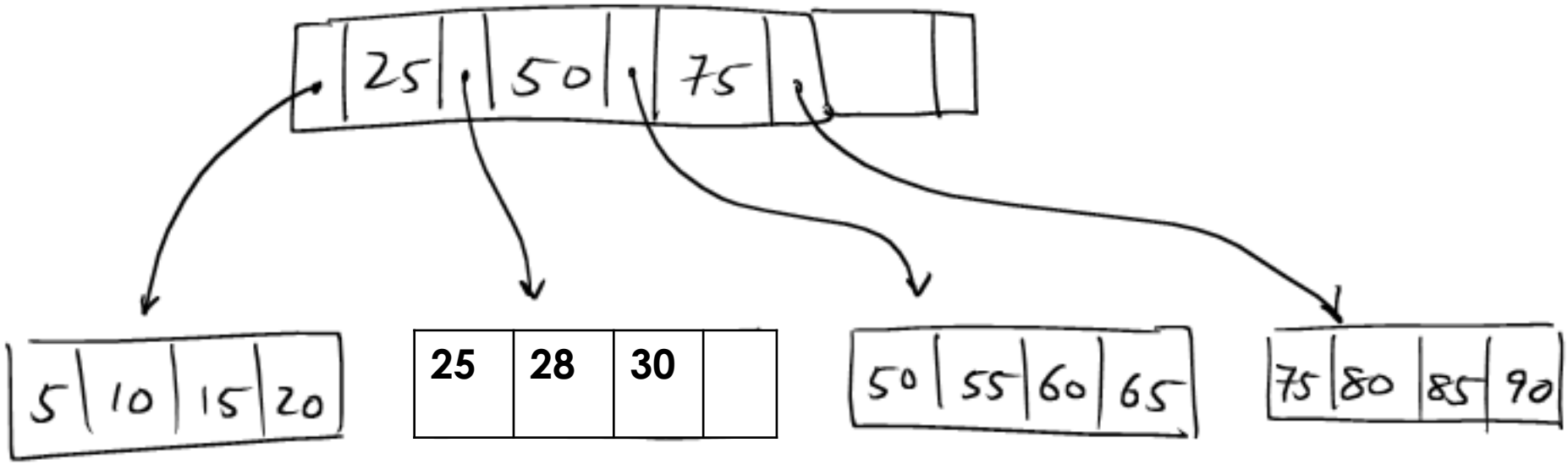


Yaprak düğümde bulunan degerler kendi içinde sıralandı

http://www.teach.cs.toronto.edu/~csc443h/fall/posted_practice/btree-index.html

B+ Tree (insert)

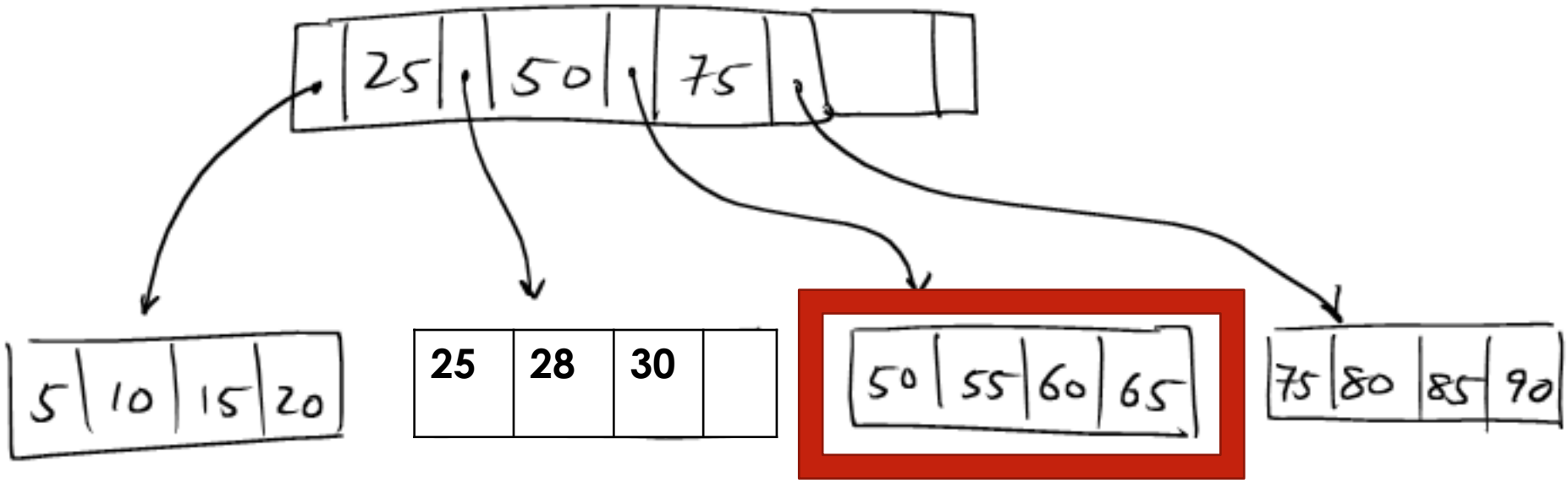
70 değerini ekleyelim



http://www.teach.cs.toronto.edu/~csc443h/fall/posted_practice/btree-index.html

B+ Tree (insert)

70 değerini ekleyelim

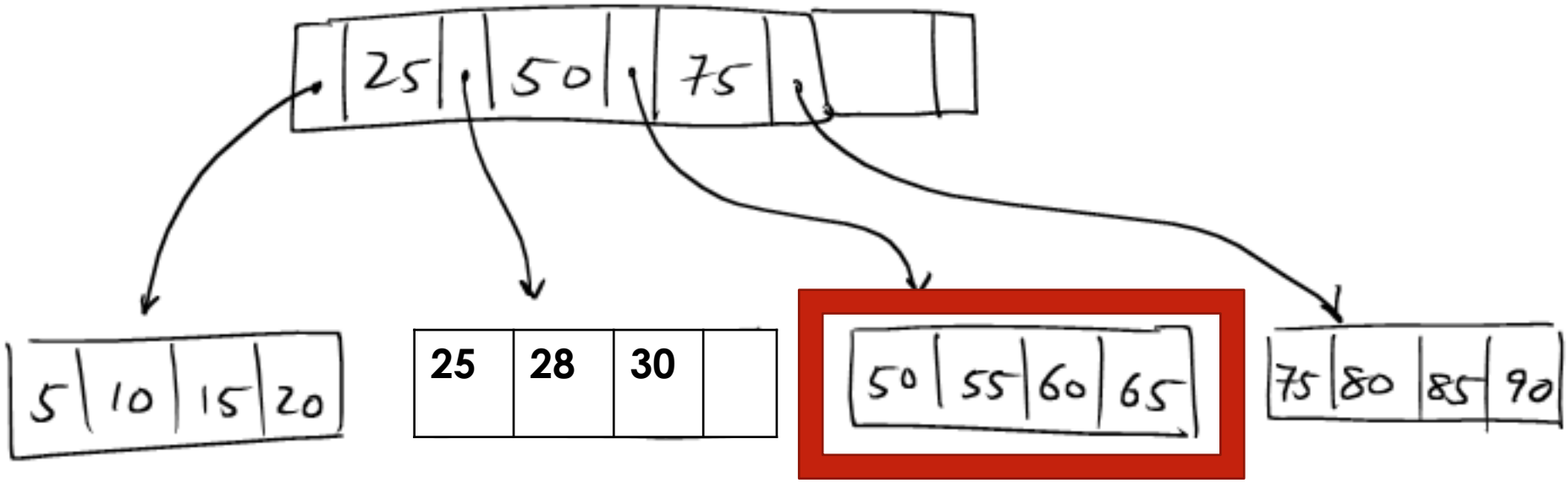


Değerin eklenmesi gereken düğüm bulunur.

http://www.teach.cs.toronto.edu/~csc443h/fall/posted_practice/btree-index.html

B+ Tree (insert)

70 değerini ekleyelim



düğüm ikiye bölündü

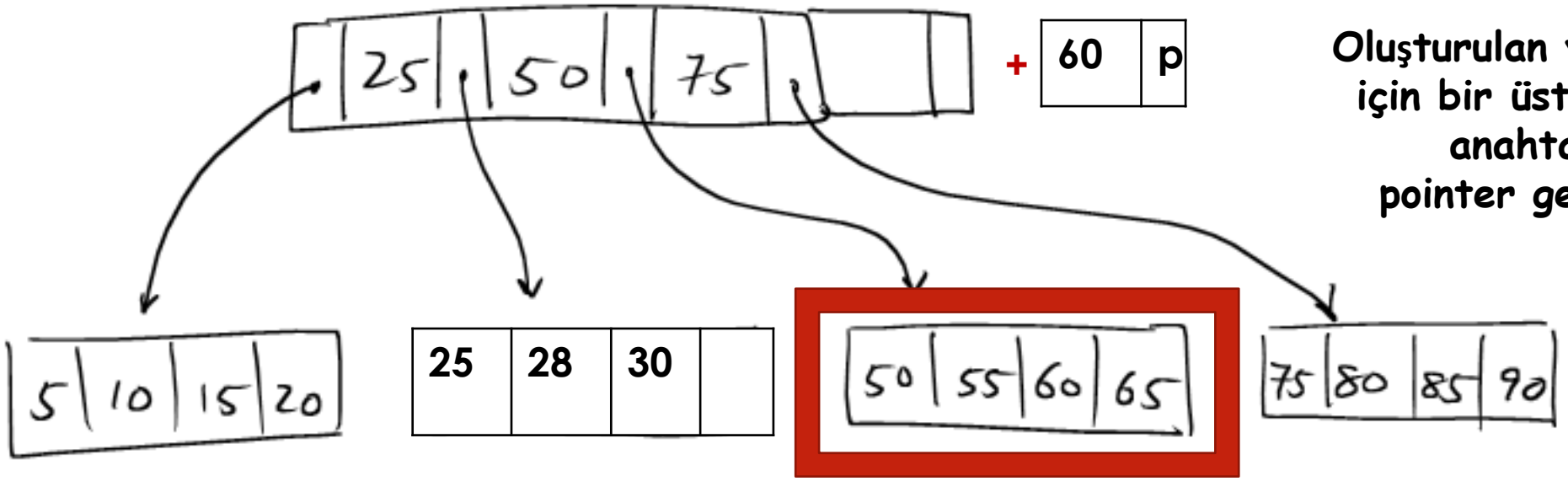
50	55

60	65	70

http://www.teach.cs.toronto.edu/~csc443h/fall/posted_practice/btree-index.html

B+ Tree (insert)

70 değerini ekleyelim



Oluşturulan yeni düğüm için bir üst seviyede anahtar ve pointer gerekecek

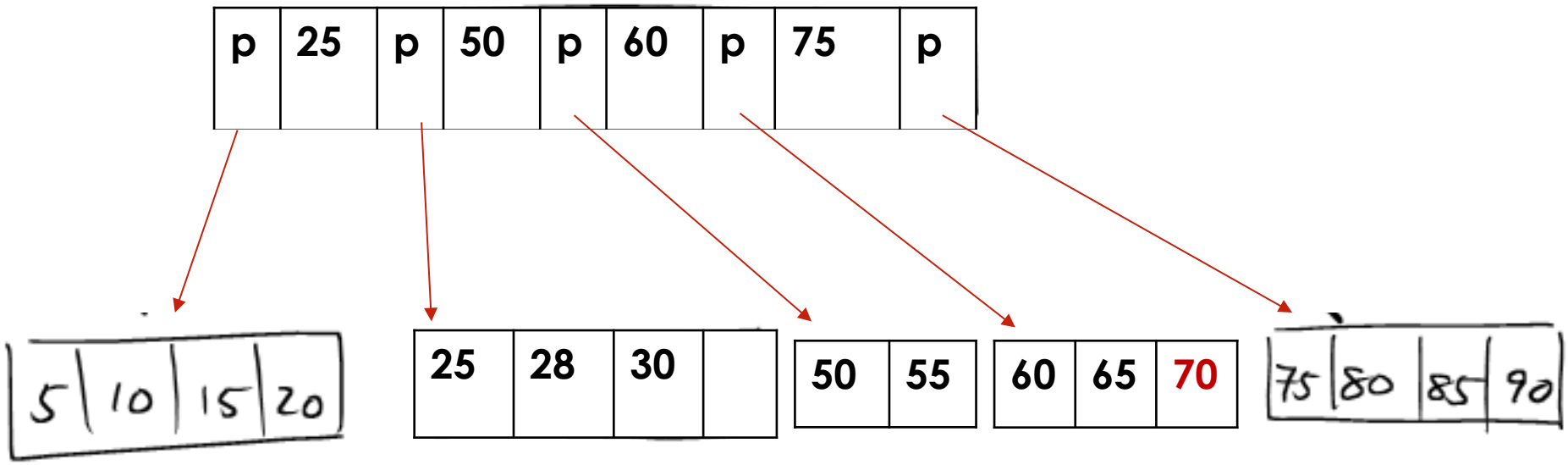
50	55

60	65	70

http://www.teach.cs.toronto.edu/~csc443h/fall/posted_practice/btree-index.html

B+ Tree (insert)

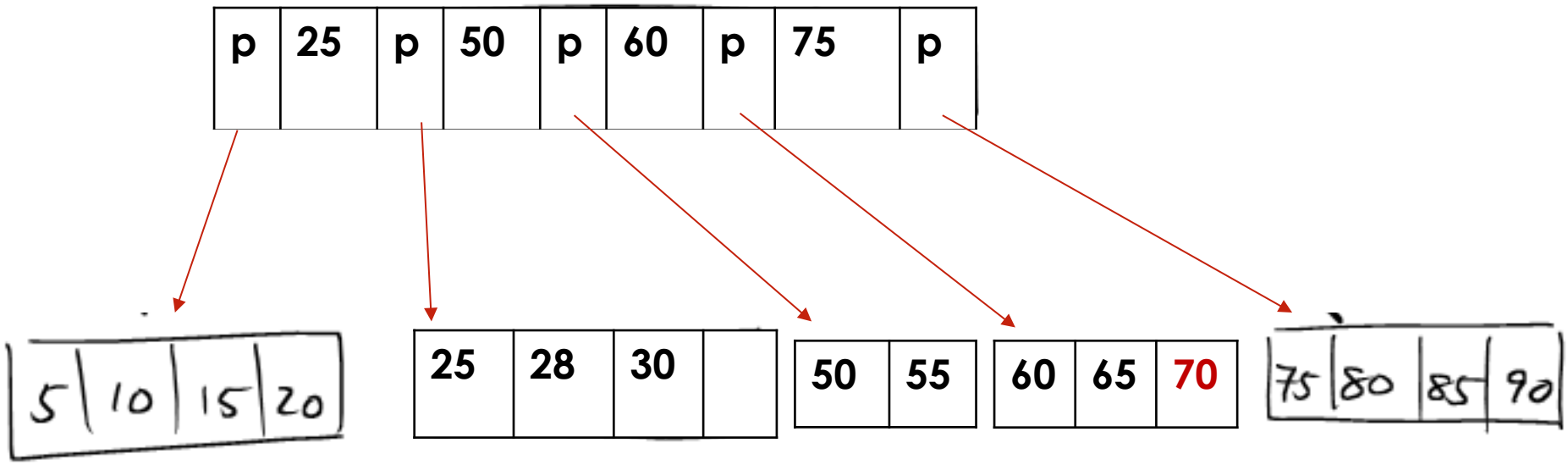
70 değerini ekledikten sonra *yeni ağaç*



http://www.teach.cs.toronto.edu/~csc443h/fall/posted_practice/btree-index.html

B+ Tree (insert)

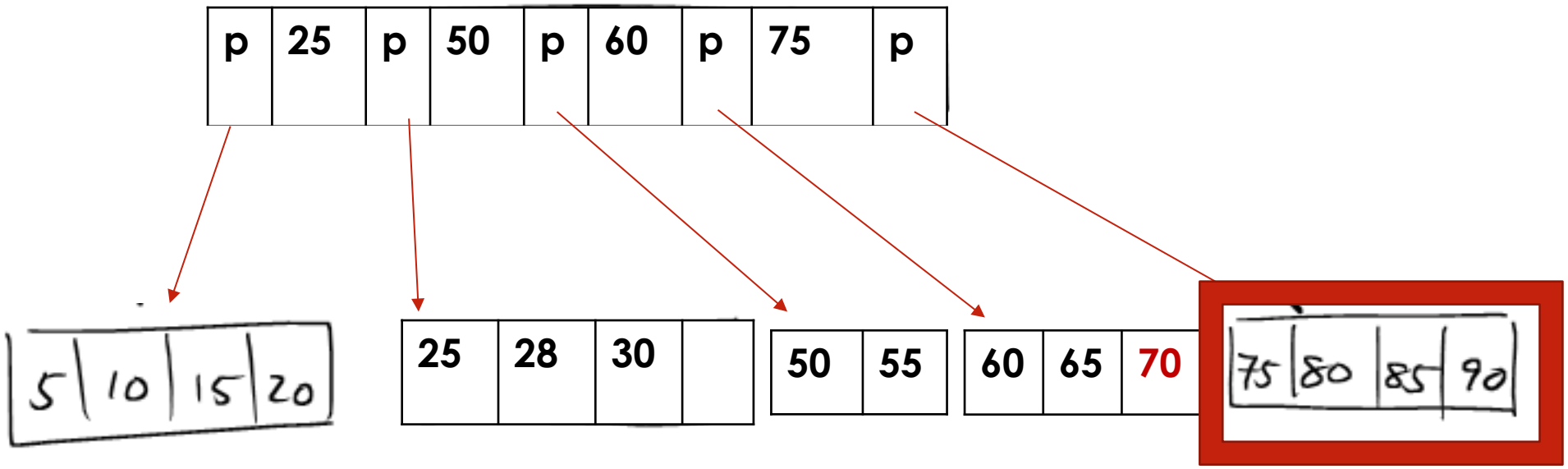
Yeni ağaca 95 değerini ekleyelim



http://www.teach.cs.toronto.edu/~csc443h/fall/posted_practice/btree-index.html

B+ Tree (insert)

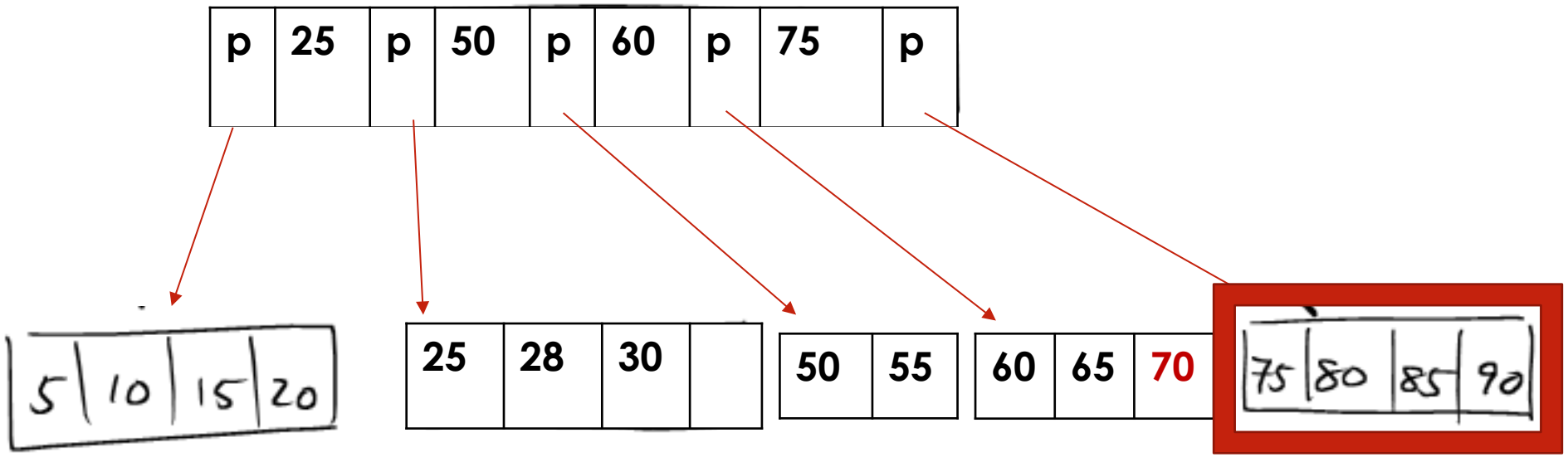
95 değerini ekleyelim



http://www.teach.cs.toronto.edu/~csc443h/fall/posted_practice/btree-index.html

B+ Tree (insert)

95 değerini ekleyelim



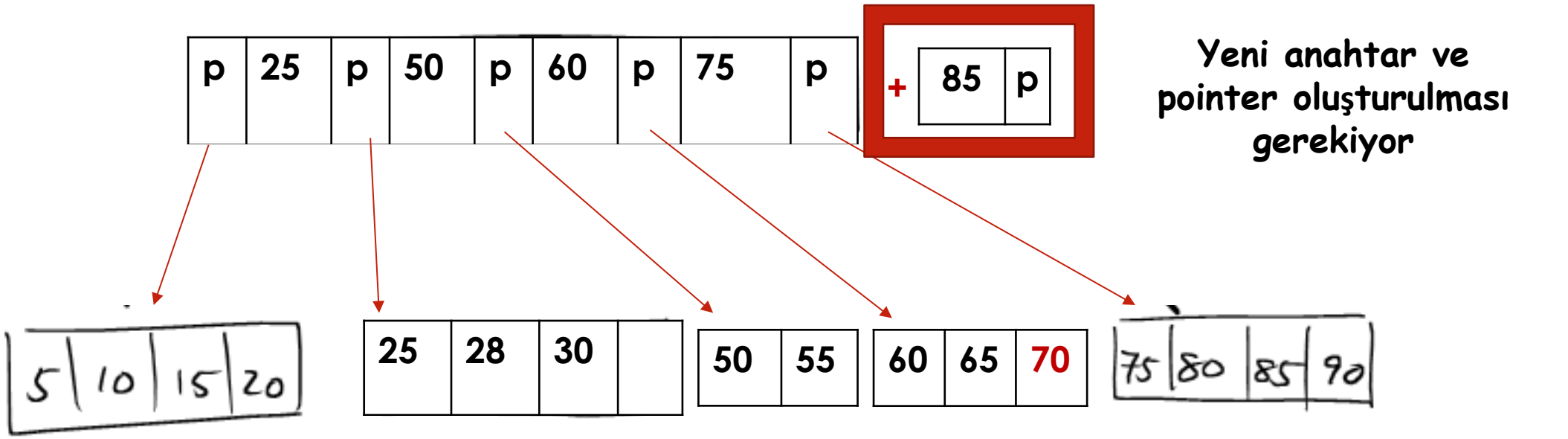
75	80

85	90	95

http://www.teach.cs.toronto.edu/~csc443h/fall/posted_practice/btree-index.html

B+ Tree (insert)

95 değerini ekleyelim

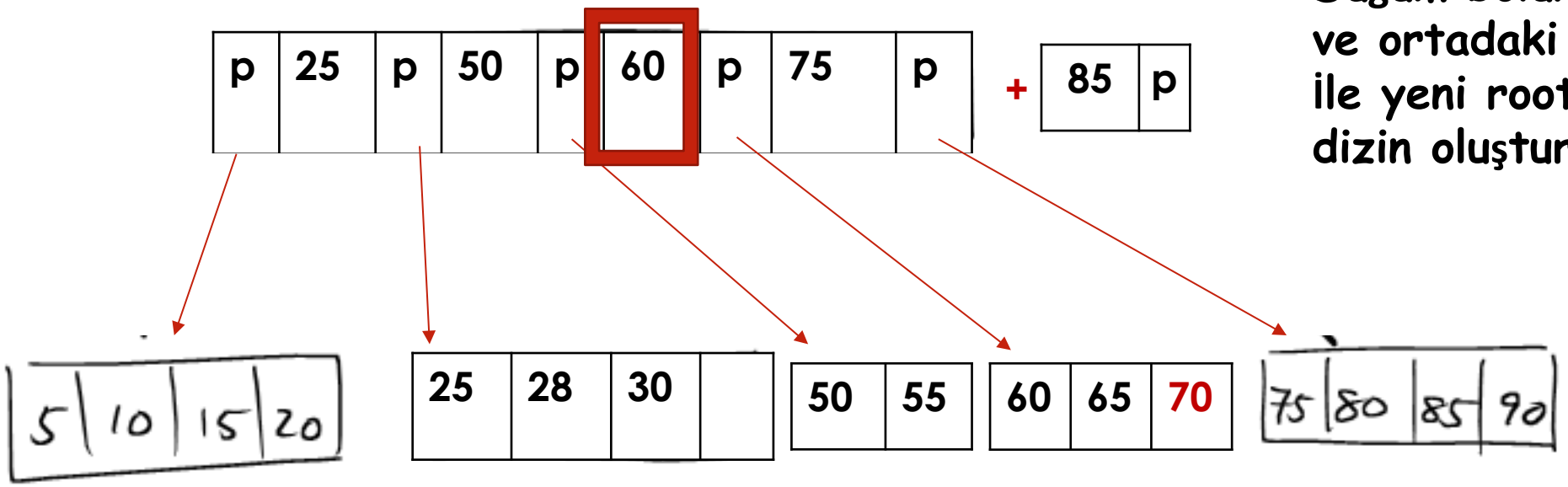


75	80	85	90	95

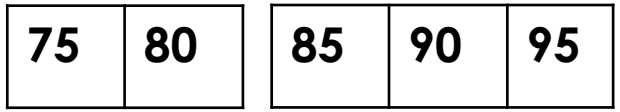
http://www.teach.cs.toronto.edu/~csc443h/fall/posted_practice/btree-index.html

B+ Tree (insert)

95 değerini ekleyelim



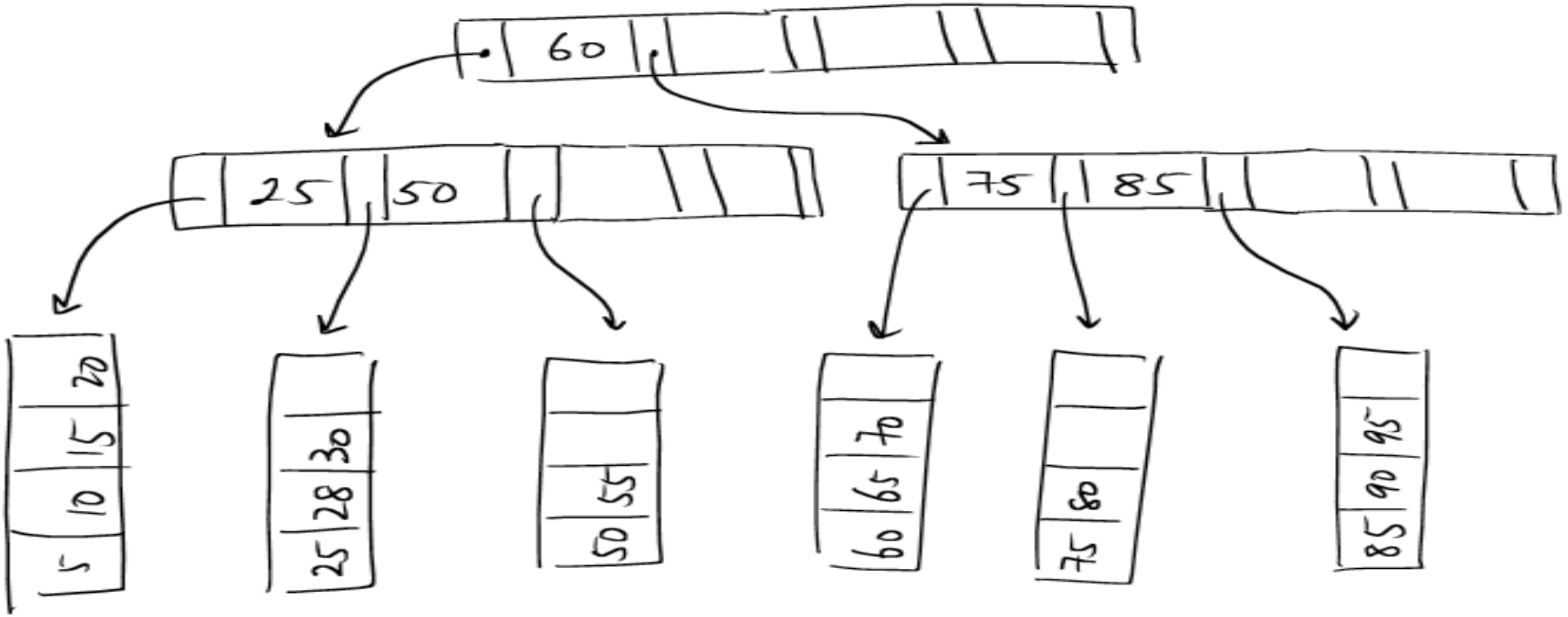
Düğüm bölünecek ve ortadaki değer ile yeni root (kök) dizin oluşturulacak



http://www.teach.cs.toronto.edu/~csc443h/fall/posted_practice/btree-index.html

B+ Tree (insert)

95 değerini ekleyelim



http://www.teach.cs.toronto.edu/~csc443h/fall/posted_practice/btree-index.html

B+ Tree (delete)

1. Silinecek kayıtın yaprak düğümü anahtar değeri yardımıyla bulunur.
2. Silme işleminin gerşekleştirileceği düğümde ağacın derecesine eşit veya büyük sayıda kayıt varsa silme işlemi gerçekleştirilir.
3. **d<= düğümdeki kayit sayisi <=2d** şartını saglamıyorsa düğümler aynı parent(ebeveyn) a sahıp düğümler birleştirilir (tekrar dagıtma işlemi)
4. Birleştirme işlemi gerşekleştirilirken bir ust seviyede (ebeveyn) sağdaki düğümü gösteren pointer ve anahtarı silinir
5. Bu işlem root (kök) a kadar devam eder.