



Karadeniz Teknik Üniversitesi

Mühendislik Fakültesi
Harita Mühendisliği Bölümü

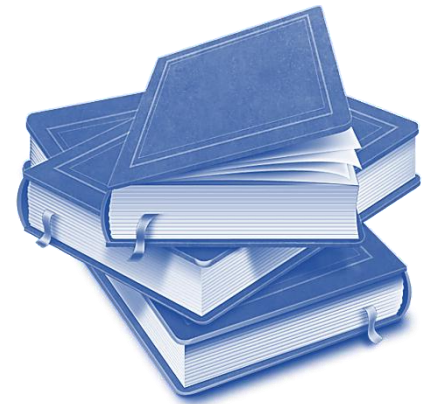
Veri yapısı, veri modeli

Kartografya ABD.



Ders İçeriği

- Veri yapısı
 - Diziler
 - Yığınlar
 - Kuyruklar
 - Ağaçlar
- Veri Modeli
 - Veri modellerinin sınıflandırılması
 - **Yüksek seviyeli veri modelleri**
 - **Mantıksal veri modelleri**
 - **Fiziksel veri modelleri**



Veri yapısı, veri modeli

- **Veri yapısı (Data Structure)**, verinin veya bilginin bellekte tutulma şeklini veya düzenini gösterir.

Tüm programlama dillerinin, genel olarak, tamsayı, kesirli sayı, karakter ve sözcük saklanması için temel veri yapıları vardır. Bir program değişkeni bile basit bir veri yapısı olarak kabul edilebilir.

- **Veri modeli (Data Model)**, Veri tabanının yapısını tanımlayabilmek için kullanılan kavramlar bütünüdür.

Verilerin birbirleriyle ilişkisel veya sıralı durumunu gösterir; problemin çözümü için kavramsal bir yaklaşım yöntemidir denilebilir.

Veri tabanı modeli, veri tipleri, operasyonlar ve kısıtlamalardan oluşur. Veri tabanı modellerinin büyük bir çoğunluğu veri tabanı üzerinde güncelleme yapan ve kayıtları alıp getiren bir operasyonlar kümesini de içermektedir.

Neden veri yapıları?

- Bilgisayar ortamında verilerin etkin olarak saklanması ve işlenmesi için gereklidir.
- Beklenen uygulamalar çok çeşitlidir ve bu uygulamaların ihtiyacını karşılayabilecek veri yapılarının seçilmesi çok önemlidir.
- Bilgisayar yazılımları gün geçtikçe daha karmaşık bir hal almaktadır. Örneğin Google kurulduğunda 25000 web sayfası indekslemiştir günümüzde milyarlarca sayfa indeksliyor. (2015-50 milyar)
- Yazılımların programlanması ve yönetimi zorlaşmaktadır.

Veri modellerinin sınıflandırılması

- **Yüksek seviyeli veri modelleri:** Bu modeller **nesneler** (entities), **öznitelikler** (attributes) ve **ilişkiler** (relationships) gibi kavramlar kullanırlar. Bu modellere örnek, ER, UML, vb.
- **Mantıksal veri modelleri:** Bu modeldeki kavramlar **verilerin bilgisayarda fiziksel olarak organize olma biçimine** çok benzerler. Bunlara örnekler, ilişkisel, nesneye-yönelik, nesne-ilişkisel, XML veri modelleridir.
- **Fiziksel veri modelleri:** Bu modeller **verilerin bilgisayarda nasıl tutulduklarını detaylı olarak gösteren** kavramlara sahiptirler. Bunlar kayıtların **biçimiyle**, kayıtların **sırasıyla**, **ulaşma yollarıyla** ilgili bilgiler içerir. Dizinleme (B+tree) fiziksel veri modelinin önemli bir ögesidir.

Veri yapıları (Fiziksel veri modelleri)

- Farklı sistemler farklı veri yapılarına sahiptir.
- Farklı CBS ler farklı veri yapıları kullanır.
- Veri yapısı seçiminde 2 faktör önemlidir.

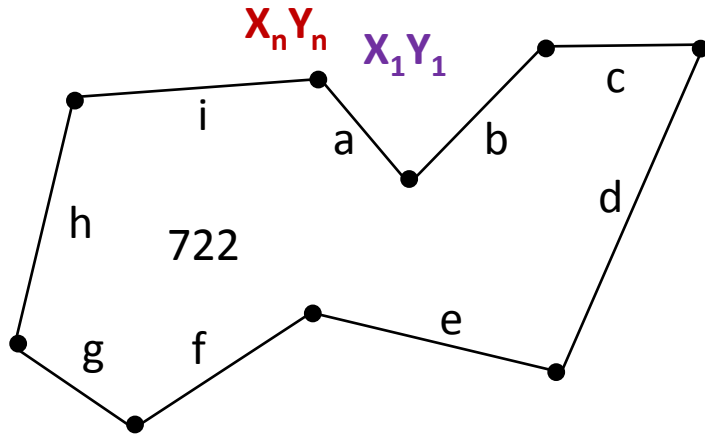
- * **Performans (Hız)**

- * **Depolama alanı** (Mümkün olduğu kadar az alan kullanmak, Güncellemeyi kolaylaştırmak için az veri kullanmak) İhtiyaç duyulandan fazla veri depolanıyorsa güncelleme zor olacaktır.

İkisi arasında trade-off ilişkisi vardır.

Örnek: ArcGIS ve CARIS poligon verisini depolama

"CARIS" (Computer Aided Resource Information System) – (1979 Canada)



ArcGIS poligon dosyası

ID	Çizgi parçaları	Sınırlayıcı dikdörtgen
...		
722	a,b,c,d,e,f,g,h,i	
...		

CARIS poligon dosyası

ID	Çizgi parçası
...	
722	x (parsel sınırının tamamı tek bir çizgi)
...	

CARIS

- Poligonları koordinat dizisi olarak kırık noktaları alır. Ancak çizgi parçalarına da etiket (ayrı bir işlem yapılması gerekir) verir. Topoloji oluştur işlemini yaparken bu işlem yapılır.
- CARIS'te poligon dosyasında bir poligon için tek bir çizgi tutulur.

ArcGIS

- Poligonların hepsi ifade edilir. **Bütün poligonlar poligon dosyasına kaydedilir.** Poligonu tek kayıt olarak tutar.
- Çok kırıklı poligon varsa tek bir tanesinin tutulması az alan gerektirir. (CARIS) -küçük ölçekli işler-
- **Depolama alanı için hangisi daha iyidir?**

Haritada birçok çizgi parçası içeren poligonlar mevcutsa CARIS daha iyidir.

- Performans için sorgulama yapmak gerekir. **Bir poligonun komşusunu bulabilmek için bütün çizgi parçalarını bulmak gerekir** . ArcGIS'te bütün dosyalar hazır durumdadır.
- Poligonun bütün çizgi parçalarını göster sorgusu için hangisi daha iyidir? ArcGIS daha iyi çünkü sorgunun cevabı zaten mevcut. **CARIS te ise sorguyu cevaplamak için ekstra hesaplama yapmak** gerekir.
- **Bir sistem geliştirilirken beklenen uygulamalar veri yapısını seçmede önemlidir.**

Genel amaçlı veri yapıları

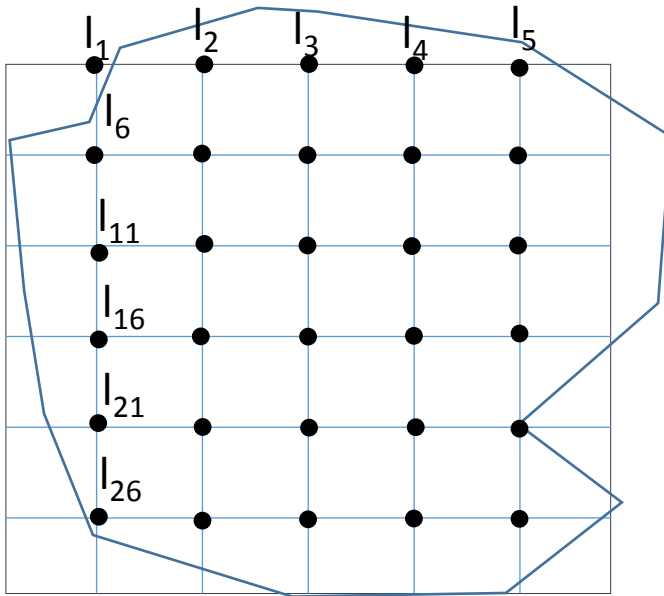
- **Diziler** (Arrays)
- **Kuyruklar** (Queues)
- **Yığınlar** (Stacks)
- **Ağaçlar** (Trees)
- ...

Diziler (Arrays)

- Dizi, dizinle (satır numarasıyla) erişilen bir veri topluluğu oluşturmak için kullanılan bir veri yapısıdır.
- Bir dizinin tuttuğu bütün öğeler aynı veri türündedir ve ardışık bellek adresleriyle gösterilen konumlarda saklanır.
- Aynı tipteki verileri tek bir dizide gruplayarak bu verilere daha kolay erişim sağlayarak, yazılım geliştirme sürecini daha basit ve yönetilebilir hale getirir.

1D ve 2D diziler

Topoğrafya düzenli bir grid ile, nokta yükseklikleriyle dizi olarak temsil edilebilir.
Belirli bir veri yapısı farklı şekillerde icra edilebilir.



$$\begin{bmatrix} l_1 & l_2 & l_3 & l_4 & l_5 \\ l_6 & l_7 & l_8 & l_9 & l_{10} \\ l_{11} & l_{12} & l_{13} & l_{14} & l_{15} \\ l_{16} & l_{17} & l_{18} & l_{19} & l_{20} \\ l_{21} & l_{22} & l_{23} & l_{24} & l_{25} \\ l_{26} & l_{27} & l_{28} & l_{29} & l_{30} \end{bmatrix}_{(6*5)}$$

2D dizi

1D Dizi (30)



12 nolu noktanın yükseklik değerinin girilmesi

2D dizi uygulaması program kodu	1D dizi uygulaması program kodu
.....	
.....	
2D Dizi (3,2)=125.46	1D Dizi (12)=125.46
.....	
.....	

Beklenen uygulamalar kullanılacak diziye belirler. Matris çarpımı uygulaması için 2D dizi kullanmak daha uygun olacaktır.

Yığın ve kuyruk veri modelleri

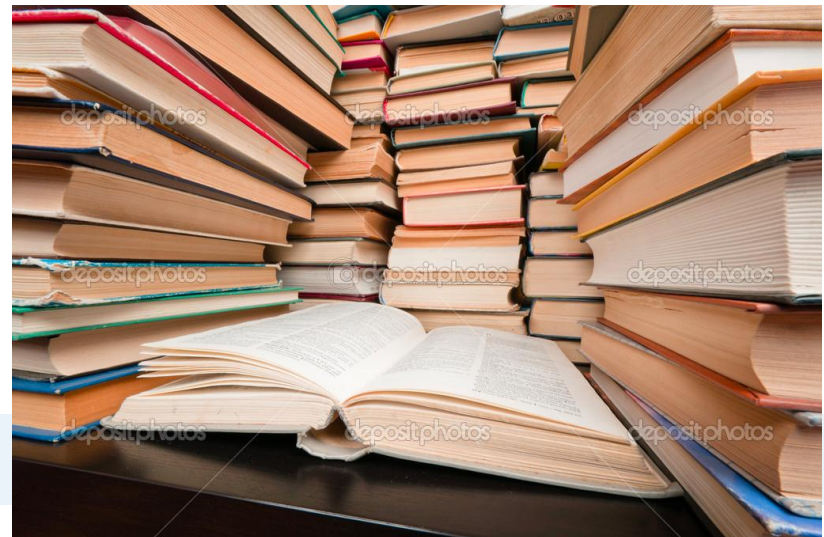
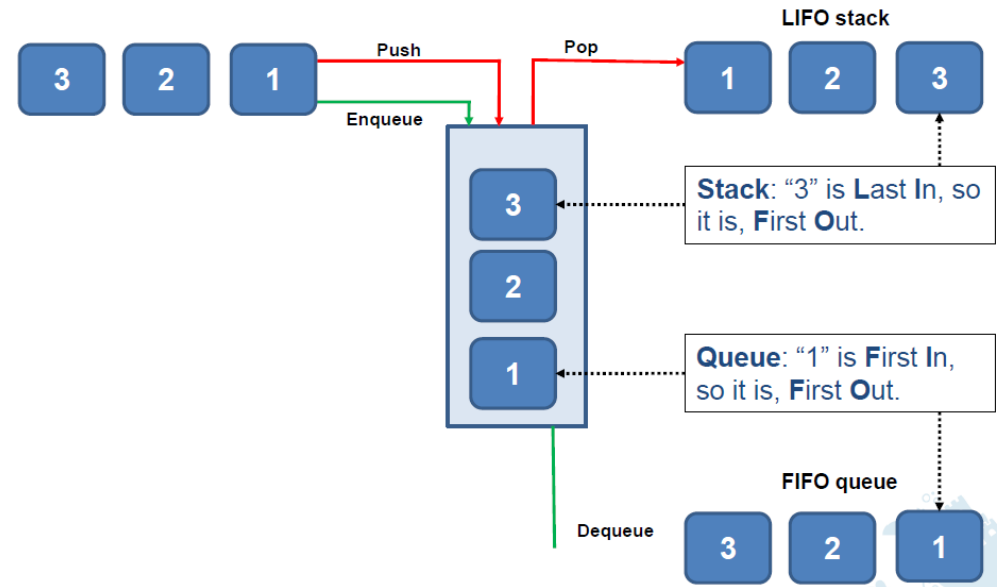
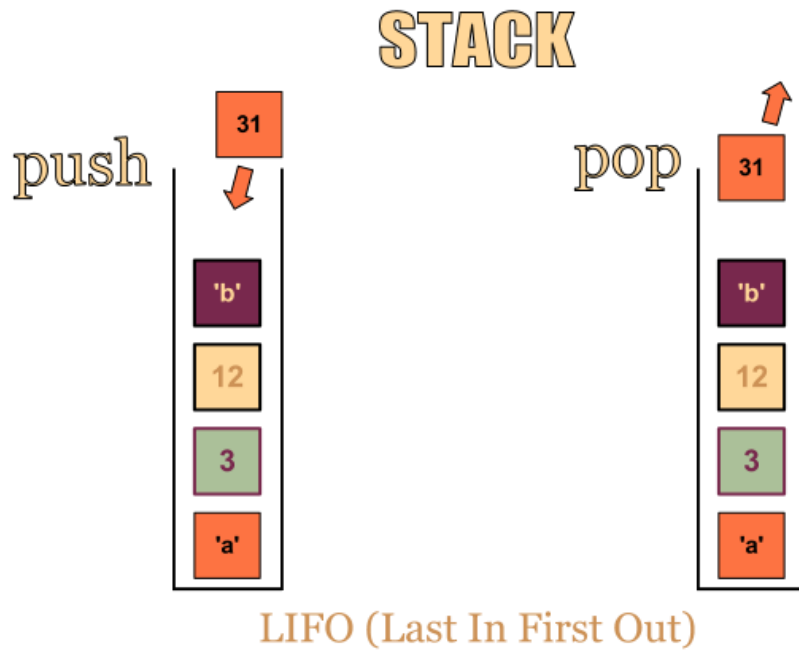
- Verileri geçici olarak saklayan, pointer (işaretçi) lar aracılığıyla verileri birbirine bağlayan mekanizmalar yığın ve kuyruk veri modelleridir.

Int[20] max -> **int** veri tipinde **20 tane** veri içerir.

Yığınlar (Stacks)

- Yığın, **son giren ilk çıkar** LIFO (last-in/first-out) ve **ilk giren son çıkar** FILO (First-in/Last-out) kuralıyla eleman erişiminin sınırlandırıldığı bir veri yapısıdır.
- Bu yapıda **ilk erişilecek eleman yığına en son eklenen elemandır**. Bu son eleman yığından çıkarıldıktan sonra diğer elemanlara erişmek mümkündür.
- İki temel işlem yapılır.
 - Push, **yığının sonuna yeni bir eleman ekleme**
 - Pop, **yığının en üstündeki elemanın alınması**

Yığınlar (Stacks)



Yığın-örnek

- Yazılım uygulamalarındaki "UNDO" işlemleri yığın ile yapılır. (UNDO işlemi için LIFO yapısı kullanılır)
- Web browser larındaki "Back" butonu (önceki sayfa) yığın veri yapısı kullanılır.
- Yazım kontrolündeki parantezlerin kontrolünde yığın veri yapısı kullanılır.
- Compiler (derleyici) program satırlarını depolamak için yığın veri yapısını kullanır.

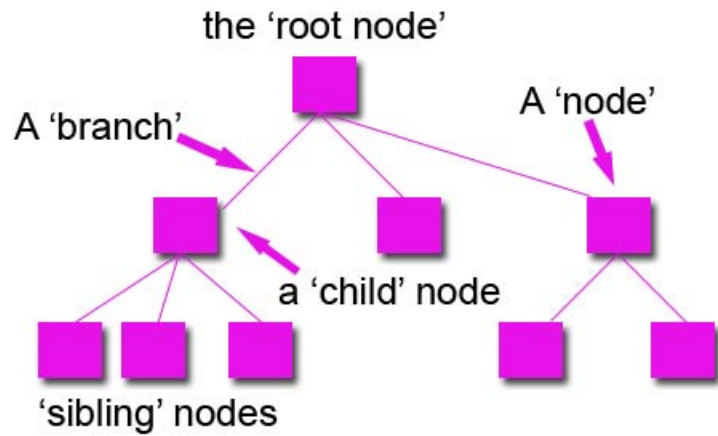
Kuyruklar (Queues)



Ağaçlar (Trees)

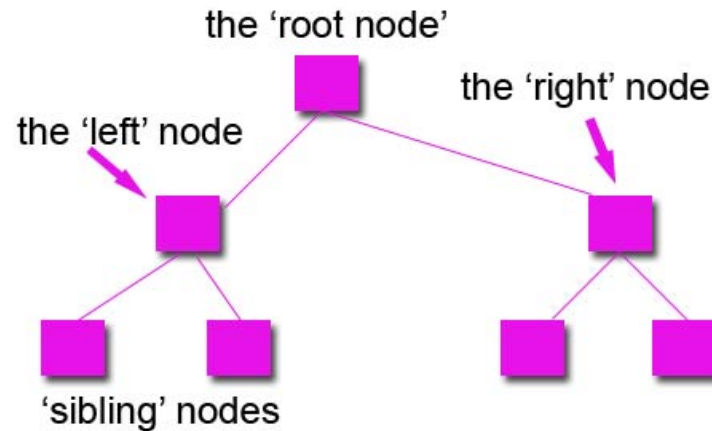
- Elemanların birbirlerine referanslar aracılığıyla bağlandığı bir veri yapısıdır. Hiyerarşik bir dizilim söz konusudur.
- Bir ağaç veri yapısında bir elemanın en fazla kaç elemana bağlanabileceği o ağaç veri yapısının derecesini (order) belirtmektedir.
- Derecesi 2 olan ağaç veri yapısı en basit ağaç türü olarak oluşmakta ve ikili ağaç (binary tree) olarak adlandırılmaktadır.
- Binary tree, Quadtree (dörtlü ağaç), Octree (sekizli ağaç), Btree

Ağaçlar (Trees)



PARTS OF A TREE DATA STRUCTURE

(c)www.teach-ict.com



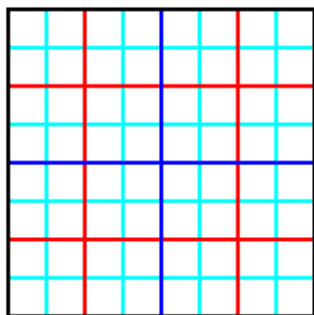
A BINARY TREE

(c)www.teach-ict.com

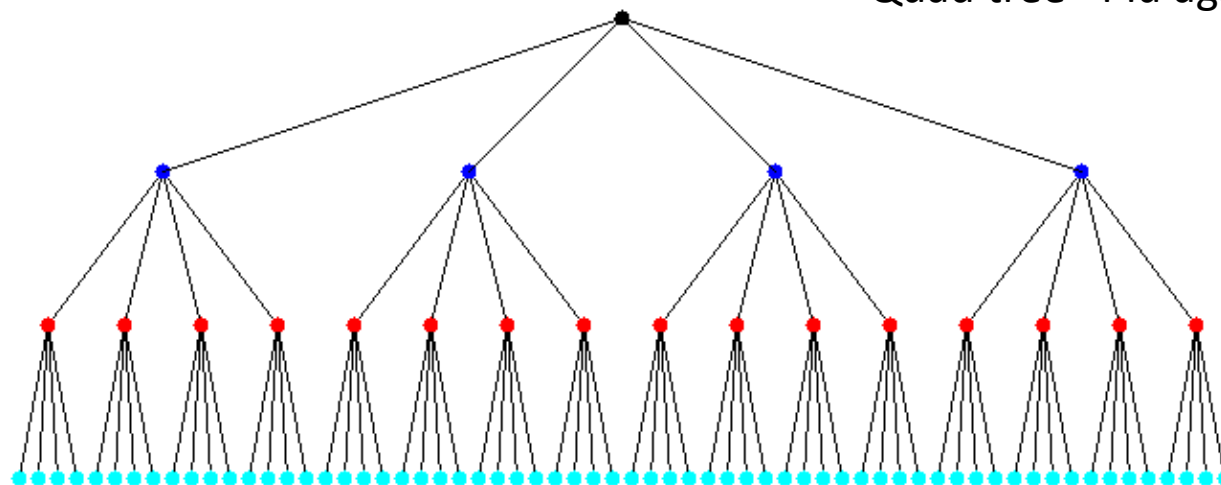
Binary Tree=ikili ağaç

A Complete Quadtree with 4 Levels

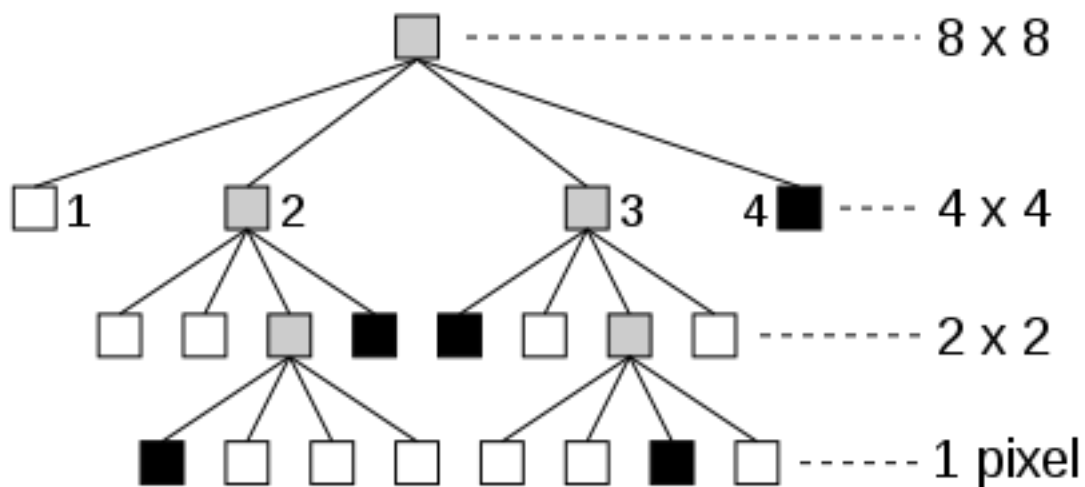
Quad tree=4 lü ağaç



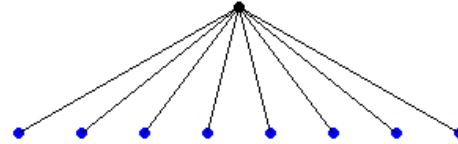
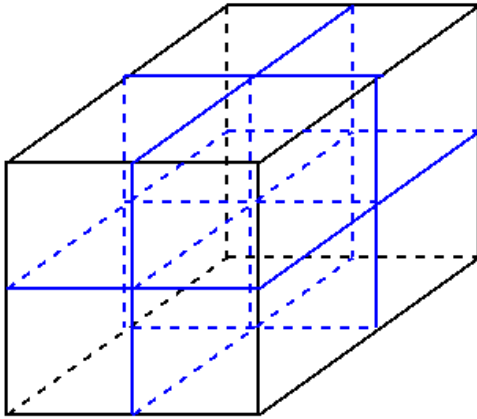
Raster Görüntü



Sıkıştırılmış Quad tree gösterimi

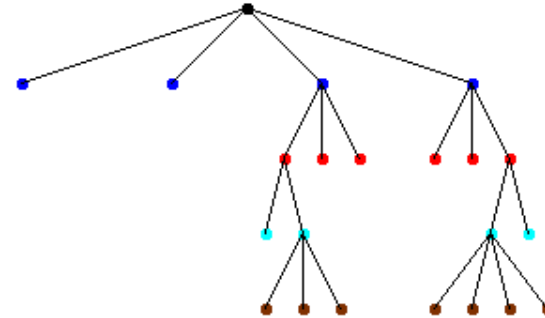
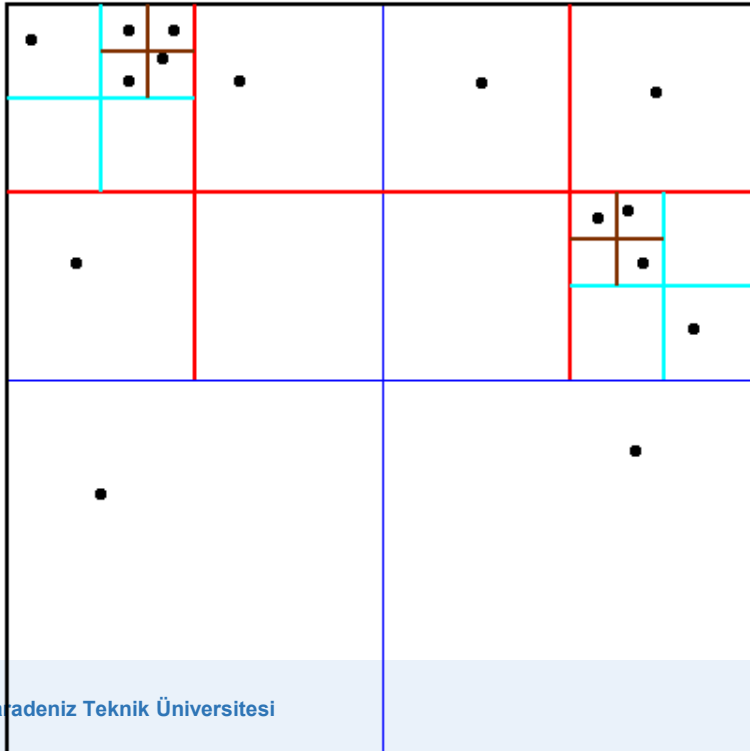


2 Levels of an Octree

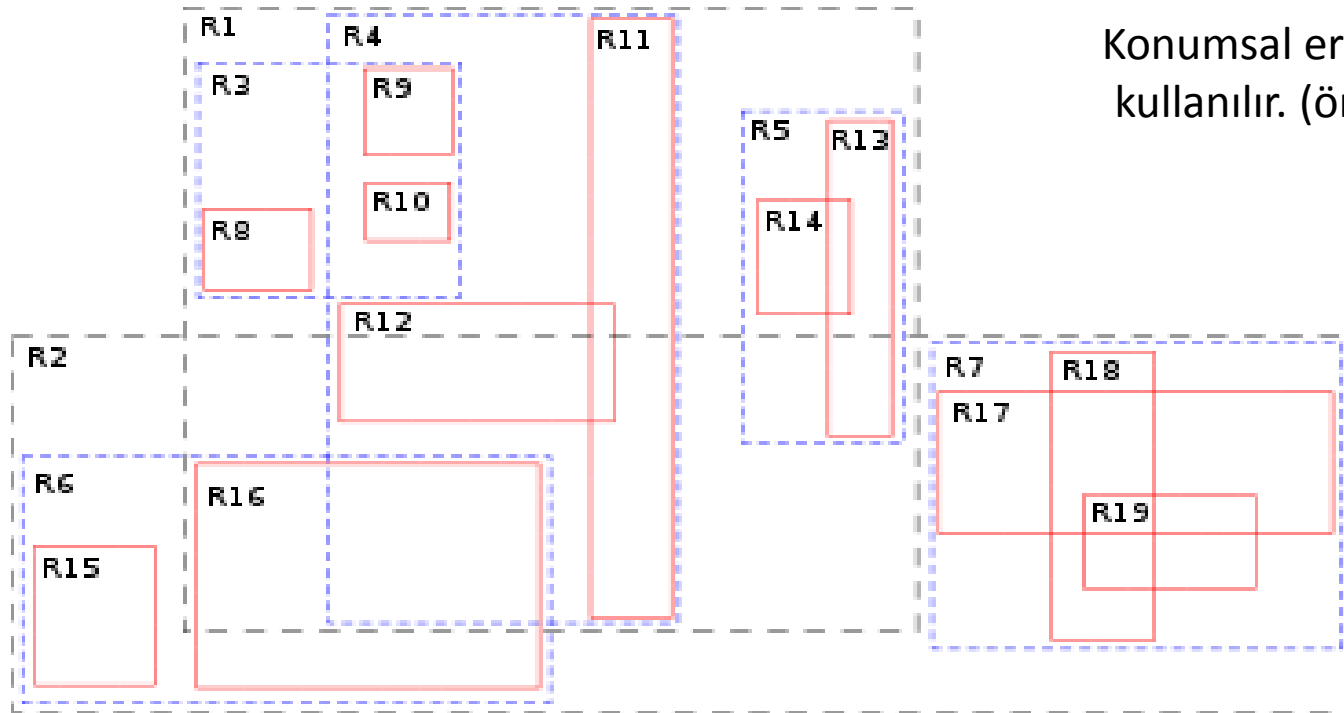


Octree= 8 li ağaç
3D için önemli

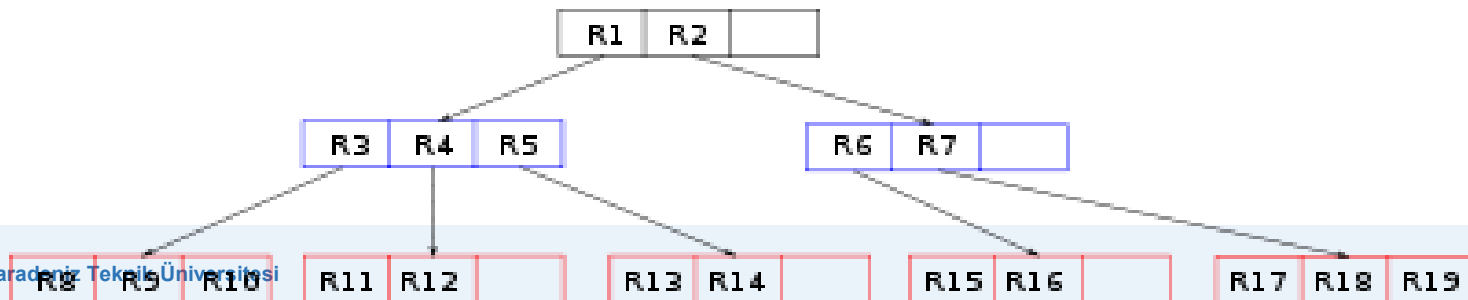
Adaptive quadtree where no square contains more than 1 particle



R-Tree

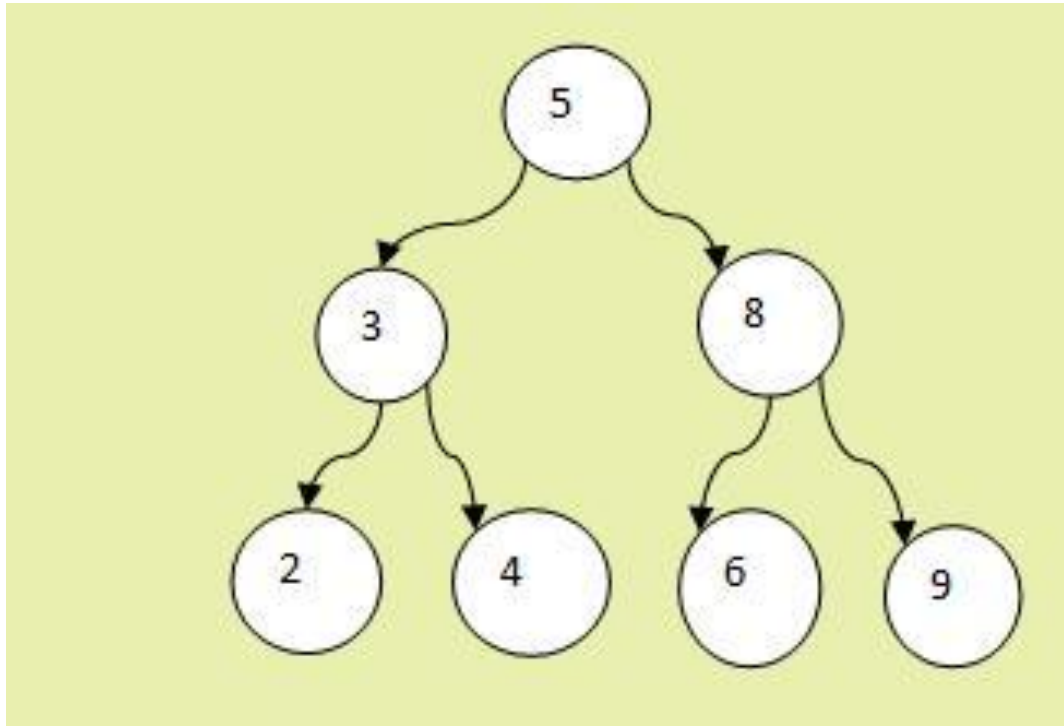


Konumsal erişim yöntemleri için kullanılır. (örn: çok boyutlu bilgi indeksleme)



İkili Arama Ağacı (Binary Search Tree)

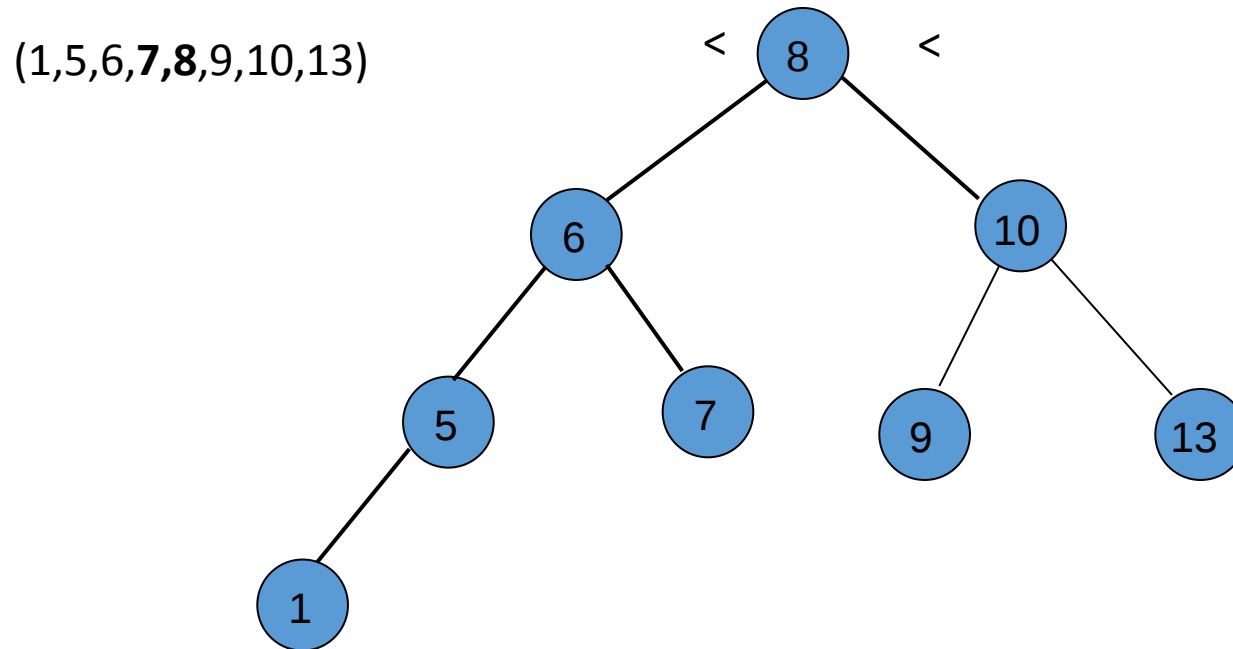
- İkili ağaçların özel bir hali olan ikili arama ağaçlarında, düğümlerde duran bilgilerin birbirine göre küçük-büyük ilişkisi bulunmalıdır.
- İkili arama ağacı, her düğümün solundaki koldan ulaşılacak bütün verilerin düğümün değerinden küçük, sağ kolundan ulaşılacak verilerin değerinin o düğümün değerinden büyük olmasını şart koşar.



- İkili arama ağacında arama işlemi

1. Aranan sayı kök düğüme eşitse-Aranan sayı bulunmuştur.
2. Aranan sayı kök düğümdeki değerden küçükse-Kök düğümün **sol kolu** takip edilir.
3. Aranan sayı kök düğümdeki değerden büyükse-Kök düğümün **sağ kolu** takip edilir.

- Verilen dizide "3" değerinin olup olmadığının sorgulanması için $\log_2 N$ kez karşılaştırma yapılır. N=eleman sayısıdır.



{3,2,6,4,7,11,14} dizisinde "5" değerinin olup olmadığını belirleyiniz.

Sıralı arama (sequential search)

Pseudo Code (sahte, yalancı kod)

BEGIN;

FOR EACH ELEMENT IN THE ARRAY DO

 IF ELEMENT=5 THEN RECORD THIS

ELSE

 CONTINUE

END FOR

END

Kod ile gösterildiği gibi aranan sayının olup olmadığını belirlemek için **her bir dizi elemanı test edilmelidir**. Bu **kaç tane dizi elemanı varsa o kadar IF döngüsü** demektir.

Sıralı aramada arama zamanı O (dizinin adı) (N) en iyi durumda $O(N)=1$ kez en kötü durumda $O(N)= N$ kez karşılaştırma yapılır.

- Aramayı hızlandırmak için veri (örneğin dizi) ikili ağaç olarak yapılandırılmalıdır. İkili ağaç için dizi elemanları sıralanmalıdır.(extra zaman harcanır ancak düzenleme 1 defa yapılır)
- Sıralı dizi={2,3,4,6,7,11,14} (orta değer alınır)
- N elemanlı bir dizide en fazla $\log_2 N$ defa karşılaştırma yaparak aranan değer yeri bulunur.
- İkili arama dizi elemanlarının sıralanmasını gerektirmesine rağmen tercih edilen bir arama yöntemidir.

