+VERİ YAPILARI

Üst düzey programlamada veri içerisinde arama yapmak, veriye hızlı bir şekilde ulaşmak,

bilgisayarın işlemcisini verimli kullanmak, aynı anda birçok isteğe cevap verebilmek gibi

gereksinimler söz konusudur. Bilgisayar programlarının karmaşıklığı ve programda işle-

nen veri büyüklüğü arttıkça, verilerin daha sistematik ve verimli yönetilmesi gerekir.

Bilgisayar programlarında verilerin sistematik ve etkili bir şekilde organize edilmesi

için veri yapıları kullanılır. Bir veri yapısı, içerdiği elemanların mantıksal düzeni ve ele-

manlar üzerinde yapılabilecek işlemler ile tanımlanır.

• Diziler (Arrays)

• Bağlı Listeler (Linked Lists)

• Kuyruklar (Queues)

• Yığınlar (Stacks)

• Ağaçlar (Trees)

• Yığın Ağaçları (Heaps)

• Özetleme Tabloları (Hash Tables)

• Çizgeler (Graphs)

-DİZİLER

Dizi (array), aynı tipteki verilerin tek bir değişken altında tutulmasını sağlayan veri yapısı-

dır. Sabit bir değere sahip olan dizinin uzunluğu, dizi oluşturulurken belirlenir. Bir dizide

bulunan verilerin her biri, o dizinin bir elemanı olarak adlandırılır. Dizinin elemanlarına

erişim indis (index) adı verilen sayısal değerler aracılığıyla sağlanır. İndislerin numaralandı-

rılması 0 ile başlar, dizinin uzunluğunun 1 eksiğine kadar ardışık olarak artarak devam eder.

-BAĞLI LİSTELER

Bağlı liste (linked list), aynı türden nesnelerin doğrusal bir sırada ve birbirlerine bağlı

şekilde saklandığı veri yapısıdır. Bağlı listedeki nesnelere düğüm (node) adı verilir ve dü-

ğümler birbirlerine bir sonraki düğümü işaret eden göstericiler (next pointer) aracılığıyla

bağlanmışlardır. Ayrıca, bağlı listelerde listenin başlangıcını işaret eden bir baş gösterici

(head pointer) de bulunur.

Bağlı listeleri oluşturan düğümler genellikle iki kısımdan meydana gelir. Düğümün ilk

kısmında veri saklanırken, ikinci kısmında ise bir sonraki düğümün bilgisayar hafızasın-

daki yeri saklanır.

\*Bağlı Liste Türleri

Bağlı listelerin elemanları dolaşılırken ileriye doğru gitmek, geriye doğru hareket etmek

ve listenin sonundan listenin başına erişmek mümkün olabilir. Belirtilen bu hareket kabi-

liyetleri, çeşitli türlerde bağlı listelerin ortaya çıkmasına neden olur. Bağlı listelerdeki üç

tür aşağıda listelenmiştir:

i. Tek yönlü bağlı liste (Singly linked list)

Tek yönlü bağlı listelerde, liste düğümleri arasındaki gezinme yalnızca ileriye doğru ger-

çekleşir.

ii. Çift yönlü bağlı liste (Doubly linked list)

Çift yönlü bağlı listelerde, liste düğümleri arasında hem ileriye hem de geriye doğru gidi-

lebilir. Çift yönlü bağlı listenin bir düğümü, bir sonraki düğümü işaret eden göstericinin

(next pointer) yanı sıra, bir önceki düğümü işaret eden göstericiyi (previous pointer) de

içerir.

iii. Dairesel bağlı liste (Circular linked list)

Bir bağlı listenin son düğümünün bir sonraki düğümü işaret eden göstericisi (next poin-

ter) listenin ilk düğümünü işaret ettiğinde liste dairesel hale gelmiş olur. Bağlı listelerin bu

çeşidine dairesel bağlı liste denilmektedir.

-KUYRUKLAR

Günlük yaşantımızda kuyruğa girmek ve sıra beklemek oldukça rutin bir eylemdir. Ban-

kalarda, gişelerde, süpermarketlerde, trafikte vb. birçok alanda insanlar sıralar oluşturur,

işini bitiren kişiler kuyruktan ayrılır, yeni gelen kişiler kuyruğa dahil olur. Günlük hayat-

taki bu kuyruk mantığı, programlamada da yer almaktadır.

Programlamada kuyruk (queue), verilerin doğrusal sırada tutulmasını sağlayan bir

veri yapısıdır. Bir kuyruğun başı (front) ve sonu (rear) bulunur. Kuyruk yapısındaki temel

işlemler olan ekleme (enqueue) son taraftan, çıkarma (dequeue) ise baş taraftan gerçekleş-

tirilir. Dolayısıyla kuyruğa ilk giren eleman, kuyruktan ilk çıkan eleman olur .

Kuyruk mantığının bir veri yapısı olarak programlanmasında iki temel yöntem vardır:

i. Dizilerin kullanımı ile kuyruk programlama

ii. Bağlı listelerin kullanımı ile kuyruk programlama

-YIĞINLAR

Yaşantımızdaki çeşitli aktivitelerde nesnelerin üst üste dizilmesi gerekir. Üniversite ye-

mekhanesindeki tepsiler, restoran mutfağındaki tabaklar, elbise dolabı rafındaki kıyafet-

ler, nesnelerin günlük yaşamda üst üste dizilmesi için gösterilebilecek basit örneklerdir.

Bir üniversite yemekhanesindeki tepsilerden almak istediğimizde, temiz tepsilerin içeri-

sinden en üstte olanı alırız. Temiz tepsiler biriktirilirken, yeni gelen tepsiler var olanların

üstüne eklenir. Nesnelerin üst üste dizilimi, günlük hayatta olduğu gibi programlamada da

var olan bir gereksinimdir. Bu ihtiyaç, yığın (stack) adı verilen veri yapıları ile karşılanır.

Yığın, verilerin doğrusal bir şekilde tutulduğu, ekleme ve çıkarma işlemlerinin en üst

noktadan yapıldığı bir veri yapısıdır. Eklenen veri, yığının en üst noktasında saklanırken;

çıkarılan veri de yığının en üst noktasından alınır. Yığının en üst noktasının takibi, yığının

tepe noktası (top) aracılığıyla sağlanır .

-AĞAÇLAR

Ağaç veri yapısı, verilerin birbirlerine temsili bir

ağaç oluşturacak şekilde bağlandığı hiyerarşik bir

Basitveri modelidir. Bir ağaç düğümlerden ve düğüm-

leri birbirine bağlayan dallardan meydana gelir.

Ağaç veri yapısı, çizge veri yapısının bir alt kü-

mesidir. Bir çizgenin ağaç olabilmesi için, her iki

düğüm arasında sadece bir yol olmalı, düğümler

arasındaki yolda döngü (cycle) olmamalıdır.

Ağaç veri yapısında bilinmesi gereken başlı-

ca kavramlar aşağıda listelenmiştir:

• Kök (Root): Bir ağacın en üst noktasında bulunan düğümdür.

• Dal (Edge): Düğümleri birbirine bağlayan kenara verilen isimdir.

• Yol (Path): Birbirleri ile bağlantılı dal dizisine yol adı verilir.

• Yol Uzunluğu (Length of a Path): Bir yolu oluşturan dal dizisindeki dal sayısıdır.

• Ebeveyn (Parent): Bir düğümden önce yer alan ve o düğüme bir dal ile bağlı olan

düğüme ebeveyn denir. Kök hariç her düğümün bir ebeveyni bulunmaktadır.

• Çocuk (Child): Bir düğümden sonra yer alan ve o düğüme bir dal ile bağlı olan

düğüm/düğümlere çocuk denir.

• Ağaç Yüksekliği (Height of a Tree): Bir ağacın kökünden ağaçtaki en alt çocuğa

kadar olan yolun uzunluğudur.

• Düğüm Yüksekliği (Height of a Node): Bir düğümden ağaçtaki en alt çocuğa kadar

olan yolun uzunluğudur.

•Düğüm Derinliği (Depth of a Node): Bir düğümden ağaç köküne kadar olan yolun

uzunluğudur.

İkili Ağaçlar (Binary Trees)

İkili ağaçlar, her bir düğümün en fazla 2 çocuğa sahip

olabildiği ağaç türüdür. Bu veri yapısında ekleme, silme

ve arama işlemleri çok hızlı bir şekilde yapılabilmektedir.

İkili Ağaçlarda Gezinme Yöntemleri

Bir ağacın düğümlerini belirli bir algoritma ve sıra çerçevesinde dolaşma eylemine ikili

ağaçta gezinme adı verilir. Bir bilgisayar programındaki ağaç veri yapısında gezinmenin

düğümlerde arama yapma, düğümleri kullanıcıya gösterme, düğüm değerlerini ekrana

yazdırma gibi çeşitli sebepleri olabilir.

İkili ağaç veri yapısı kendi içerisinde alt ağaçlardan meydana geldiği için, ikili ağaçları

gezinmede özyinelemeli fonksiyonlar kullanılır. İkili ağaçlardaki düğümler dolaşılırken

farklı yöntemler uygulanabilirken, bilgisayar programında bu işi yapabilmek için kabul

görmüş üç gezinme yöntemi bulunmaktadır:

i. Preorder Gezinme (Kök başta)

Bu yöntemde öncelikle kök, daha sonrasında sol alt ağaç, en son olarak da sağ alt ağaç

üzerinde gezinme yapılır. Bu yöntemi akılda tutmak için “Root – Left – Right” terimini

kullanabiliriz.

ii. Inorder Gezinme (Kök ortada)

Bu yöntemde öncelikle sol alt ağaç, daha sonrasında kök, en son olarak da sağ alt ağaç

üzerinde gezinme yapılır. Bu yöntemi akılda tutmak için “Left – Root – Right” terimini

kullanabiliriz.

iii. Postorder Gezinme (Kök sonda)

Bu yöntemde öncelikle sol alt ağaç, daha sonrasında sağ alt ağaç, en son olarak da kök

üzerinde gezinme yapılır. Bu yöntemi akılda tutmak için “Left – Right – Root” terimini

kullanabiliriz.

-AVL Ağaçları

İkili ağaçlarda ve ikili arama ağaçlarında ağacın yüksekliği için herhangi bir ölçüt bu-

lunmamaktadır. N adet düğüme sahip bir ikili ağacın yüksekliği en fazla N-1 olabilir. Bu

ağaçlarda yükseklik için bir kısıtlama olmaması, ağaç içerisindeki düğümlerin dengesiz

dağılmasına sebep olabilir. Başka bir ifadeyle, sol alt ağaç ile sağ alt ağaç arasındaki yük-

seklik farkı 1’den fazla olabilir.

AVL (Adelson –Velsky – Landis) ağaçları, ikili arama ağaçlarının özel bir türüdür. Bu

veri yapısında ağaç içerisindeki denge korunmakta, sol alt ağaç ile sağ alt ağaç arasındaki

yükseklik farkı en fazla 1 olabilmektedir.

AVL ağaçlarındaki düğümler için denge faktörü aşağıdaki formül ile hesaplanır ve

dengeli bir ağaç için bu değerler yalnızca -1, 0 ve 1 olabilir:

=-

-YIĞIN AĞAÇLARI

Yığın ağaçları, bir veri kümesi içerisinde en küçük elemanın hızlıca bulunmasını sağlayan

bir veri yapısıdır. Bu veri yapısında en küçük elemanı bulma, en küçük elemanı silme ve

ağaca eleman ekleme işlemleri hızlıca yapılabilir.

Aşağıda verilen iki özelliği sağlayan bir ikili ağaç, yığın ağacı veri yapısı olarak sınıf-

landırılır:

1. Ağaç bütünlüğü: Ağacın son düzeyi hariç tüm düzeyleri, içerdikleri düğümler

bakımından eksiksiz olmalıdır. Ağacın son düzeyindeki düğümler de soldan sağa

doğru dolu olmalıdır.

2. Heap özelliği: Bir düğümün sahip olduğu değer, düğümün çocuklarına ait değer-

lerden küçük veya eşit olmalıdır.

-ÖZETLEME (HASH) TABLOLARI

Özetleme tabloları ekleme, silme ve arama işlemlerinin çok hızlı bir şekilde yapılmasını

sağlayan, verileri bir anahtar ve veri çifti şeklinde saklayan veri yapısıdır.

Özetleme tablolarındaki genel çalışma mantığı, verileri N boyutlu bir dizide tutmak ve

verilere erişim için sayı veya dizgiden oluşan anahtarı kullanmaktır. Özetleme tablosunda

saklanacak bir veri için hash fonksiyonuna anahtar değeri gönderilir, fonksiyonun hesap-

ladığı değer, verinin dizide tutulacağı indis olur.