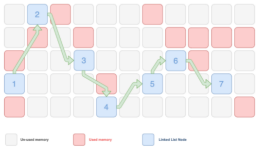
Veri Yapıları

Array

- Aynı tip veri türünde bir grup öğe içeren ve bunları bitişik bellek konumlarında birlikte depolayan bir veri yapısıdır.
- Arraye ait değişkenler, sıra numarası verilmiş bir liste gibidir. Bu listeye ait olan öğelere de sıra numarasıyla erişilir.
- Örneğin 50 öğesi olan bir sonlu dizinin biçimi aşağıdaki gibidir.
  - d = {d[0], d[1], d[2], d[3], d[4], ….. , d[49]}
- Dynamic Arrays (Dinamik Diziler) yeni bir eleman için boşta yer tutmasından dolayı esnektir. Dezavantajlarından biri ise hafızada fazladan yer kaplaması, gerçekleşecek olan bir diğer olayı engelleyebilir.
- Array'lerin birbirine bağlı olması ulaşılabilirliği kolaylaştırır.

Linked List

- Yan yana zorunluluğu olmadan veri tutmamızı sağlayan yapılardır. Yeni gelen eleman için hafızada yeni bir alan açmamız gerekmez. Array'den farklı olarak evet elemanlar hafıza içerisinde dağılmış olabilir, fakat son gelen eleman kendinden bir önceki elemana adresini bildirmek zorundadır.



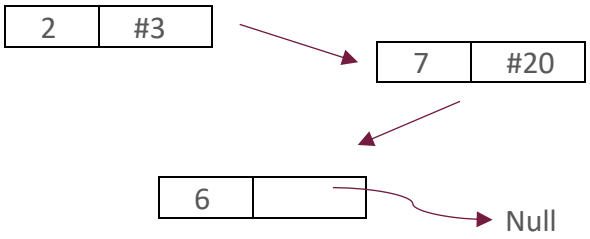
Array	Linked List
- Arrayin herhangi bir elemanına ulaşmak aynı sürede gerçekleşir. (Random Access)	- Linked List'de ulaşmak istediğimiz elemana gidebilmek için birbirine bağlı olan elemanları ziyaret etmemiz gerekiyor.
- Array'ler, sadece elemen tuttuğu için hafızada daha az yer kaplar.	- Linked-List'ler, eleman ile birlikte adres tuttuğundan dolayı hafızada daha fazla yer kaplar.
- Daha çok statik (durağan) durumlarda daha fazla performans gösterir.	- Ekleme, çıkarmanın fazla olduğu durumlarda linked-list daha fazla performans gösterir.

Dizilerde ulaşmak istediğimiz elemana indisini girerek ulaşırız. Linked List'lerde ise ulaşmak istediğimiz elemanlara point eden pointerlar vasıtasıyla ulaşırız.

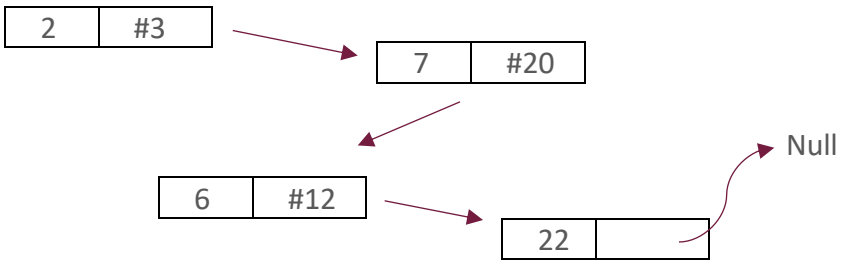
Dizilerde eleman ekleme, silme gibi işlemler Linked List'lere göre performans açısından daha maliyetlidir. Örneğin; 1000 elemanlı bir dizimiz tanımlı olsun. Bu dizinin 500.cü

elemanını silmek istediğimizde, bu elemandan sonra gelen her eleman bir sıra geri kaydırılacak bu da performans kaybına yol açacaktır. Linked List'te ise bu işlem sadece basit pointer operasyonlarıyla gerçekleştirilir ve kaydırma işlemlerine gerek kalmaz. Bu sayede performanstan kazanç sağlanmaktadır.

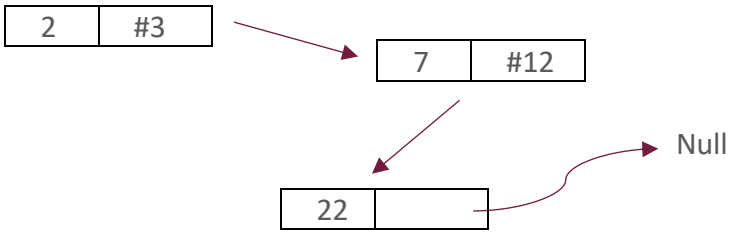
Linked List Eleman Ekleme/Silme



Adresi #12 olan 22 sayısını listeye eklemek istediğimizi düşünelim. Yapmamız gereken 6 hücresine 22 sayısının adresini yazmak.



Adresi #20 olan 6 numaralı hücresini çıkarmak istiyoruz. Linked-List'de bir önceki eleman adresini tutuyordu. Yani 7 numaralı hücrede bulunan 6'nın hücre adresini siliyoruz. Yerine 22 numaralı hücrenin adresini yazıyoruz.



Stack

- Stack (yığıt) sınıfı nesnelerin LIFO (last-input-first-output) yapısıyla depolanmasını sağlayan bir veri tipidir. Bilgisayar uygulamalarında çok sık kullanılır. Üst üste konulmuş kutular gibidir. Yani gelen kutu en üste konur. Alttaki ya da aradaki bir kutuyu almak için, en üsttekinden başlayarak, istenen kutuya kadar üsttekileri sırayla almak gerekir.
- Yığınlara eleman eklerken veya çıkartırken bazı methodlar uygulanır. Bunlardan biri push, diğeri ise pop. Push, yığının üzerine eleman eklemek için kullanılır (Koliye kitap koymak). Pop ise, yığından eleman çıkarmak için kullanılır.

Queue

- Queue(Kuyruk), FIFO (First in First out) (İlk giren ilk çıkar) prensibine dayanan, girişlerde ve çıkışlarda belirli bir kurala göre çalışan yapıdır.
- Queue veri yapısında, verilere ***iki uçtan erişim*** vardır. Bir uçtan eleman ekleme ***(enqueue)***, diğer uçtan eleman çıkarma ***(dequeue)*** işlemleri yapılır.

Hash Function/ Hash Table

- Hash Table, key value prensibine dayanan bir array kümesidir. Key olarak çağırdığınız elemanın değerini (value) yansıtır.
- Hash Table ürünlerin fiyatını ezbere bilen çalışan gibi.
- Hash Table yerine dizileri kullanabilirdik. Fakat her ürünü ve fiyatını tek tek aramak istemediğimiz için hash table kullanıyoruz.
- Bunu nasıl çözeriz?
  - İlk eleman sayısı ürün sayısına eşit bir array oluşturacağız.
  - Ürünlerin isimlerini bir fonksiyona sokup çıktılar alacağız.
  - Fonksiyonun çıktılarını oluşturduğumuz array'in indexleri olarak kullanıp, ürünlerin fiyatlarını bu indexlerde tutacağız.
  - Arrayler bize kaçıncı eleman olursa olsun sabit sürede istenen lokasyondaki elemanı verebiliyordu.
  - Bu sabit sürede erişmeyi lokasyon bazlı değil, tanım bazlı kullanmak istiyorum. Bana 3. elemanı getir değil, bana elmaya karşılık gelen elemanı/fiyatı getir demek istiyorum.
  - Biri bize bir ürünün fiyatını sorduğunda bu ürünü oluşturduğumuz fonksiyona besleyip arraydeki indexi neredeymiş onu bulacağız.
  - Bu fonksiyona Hash Function, Hash Function + Array yapısına da Hash Table deniyor.

Hash Function

1) Hash Function her seferinde aynı girdiye aynı sonucu vermeli.
2) Farklı girdilere farklı çıktılar versin istiyoruz.
3) Hash Function'ın çıktılarının sınırlarında (range) olmalı. Arrayin boyutu

Hash Collision

- Hash Function farklı iki değerden aynı sayı üretilirse bu duruma Collision (Çarpışma) denir. Bu istediğimiz bir olay değildir.
- Collision sorunuyla az karşılaşabilmek için kaliteli bir hash function olmalı. Bu sayede verimli bir Hash Table elde etmiş oluyoruz.