ARAMA YÖNTEMLERİ

1) Sırayla Arama: Arama yöntemleri içinde en basit yöntemdir. Dizi ya da bağlı liste içinde yer alan elemanlar içinde belirli bir verinin elemanların sırayla kontrol edilerek aranması şeklinde gerçekleştirilir. Veri yapısı ilk elemandan başlanarak son elemana kadar tek tek kontrol edilecektir.

```
BListe *Arama(int aranan)
{     BListe *gecici;
      gecici=ilk;
      while(gecici!=NULL)
      {
            if (gecici->numara=aranan) return gecici;
            gecici=gecici->sonraki;
      }
      return NULL;
}
```

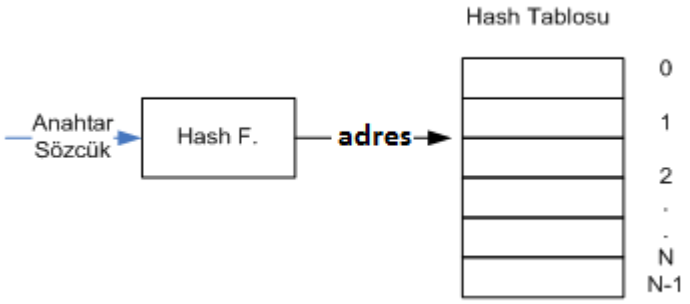
Ortalama olarak her elemana *N/2* adımda ulaşılır. (1+2+…+N-1+N)/N =N/2 dir.

2) İkili Arama: Aranacak kayıt sayısı çok fazla olduğu durumlarda elemanları tek tek aramak işlem yükü getirecektir. Bu durumda işlem yükünü azaltan yöntem olan ikili arama kullanılabilir. **İkili aramanın gerçekleşebilmesi için veri yapısındaki elemanların önceden *sıralanmış* olması gerekir**. Buda ikili aramada ek işlem yükü getirecektir. Elemanlar sıralı olduğu durumda ikili arama metodu önce aranan elemanın veri yapısının tam ortasında yer alan elemanla karşılaştırılmakta ve bu elemandan büyük veya küçük olma durumuna göre aramaya aynı yöntemle veri yapısının alt veya üst bölümünde devam etmektedir. Her bir adımda aramanın gerçekleştiği verinin büyüklüğü yarıya düşürülmektedir.

```
int ikili_arama(in dizi[], int aranan, int boyut)
{     int alt_sinir=0, ust_sinir=boyut-1,orta;
      while(ust_sinir >= alt_sinir)
      {
            orta=(alt_sinir + ust_sinir)/2;      // ortanca indisi belirle
            if (aranan==dizi[orta]) return 1;     // başarılı arama
            if (aranan<dizi[orta]) ust_sinir= orta-1;   // sol parçada ara
            else alt_sinir= orta+1;          // sağ parçada ara
      }
      return 0;               // başarısız arama
}
```

3) **HASHING**

Hash fonksiyonu arama işleminde aranan veriye doğrudan ulaşmak için kullanılan bir fonksiyondur. Fonksiyona arama işleminde kullanılacak anahtar sözcük değeri girilir. Girilen değere karşılık fonksiyon bir tamsayı değer üretir. Elde edilen bu tamsayı değeri indis gibi kullanılarak dizi üzerinde tutulan (aranan) veriye ulaşım sağlanır. İdeal olarak her bir veriye 1 adımda ulaşılması amaçlanır. Ancak bazı durumlarda hash fonksiyonu farklı değerler için aynı sonucu üretebilir. Bu duruma **çakışma (collision)** denir.



Hashing, elimizdeki veriyi kullanarak o veriden elden geldiği kadar benzersiz bir tamsayı elde etme işlemidir. Bu elde edilen tamsayı, dizi şeklinde tutulan verilerin indisi gibi kullanılarak verilere tek seferde erişmemizi sağlar. Bu daha önceden incelediğimiz arama yöntemlerine nazaran çok daha hızlı sonuç verecektir. Hashing konusunu alt başlıklara ayıracak olursak;

Hash Fonksiyonu

Bu fonksiyonun görevi kendisine verilen bir değerden, benzersiz bir tamsayı (key) üretmesidir. Fakat uygulama alanında her zaman benzersiz bir sayı üretecek uygun fonksiyonu bulmak çok zor hatta bazı durumlarda imkânsızdır. Eğer farklı iki değerden aynı sayı üretilirse bu duruma çatışma denir (bu konuyu daha sonra inceleyeceğiz). Bir hashing fonksiyonundan beklenenler:

- Herhangi bir uzunlukta değer alabilmelidir.
- Çıktı olarak belirlenen uzunlukta anahtar (key) üretebilmelidir.
- Tek yönlü olmalıdır. Fonksiyon tarafından üretilen anahtardan fonksiyona verilen değer elde edilmemelidir.
- Çatışmalara olanak vermemelidir.

Örnek: İngilizce deki 26 harfin her biri bir komuta karşılık gelsin. Her bir harfe karşılık gelen komutun çalışması için şu şekilde bir hash fonksiyonu belirlenebilir.

Hash(karakter)=karakter-65

Burada A için 65-65 yani 0, B için 1, Z için 25 değeri üretilerek her bir harfe uygun bir tamsayı değeri üretilmiş olur. Bu tamsayı değerleri ile de hash tablosuna ulaşılarak hash tablosunun uygun indisinden faydalanılarak istenilen komut çalıştırılabilir.

Bazı hash fonksiyonları:

Key mod N: Burada Key aranan ifade N ise tablo boyutunu (toplam eleman sayısını) gösterir.

Key mod P: Burada P tablo boyundan büyük en küçük asal sayı.

Truncation: Bu bölümde key'in çoklukla değişen ve anlamlı olan kısmı alınır. Örneğin 19 kodlu öğrenciler içinde yalnızca son 3 haneyi kullanabiliriz.

191213xxx veya 191220xxx gibi

191213010 ile 191220010 numaralı öğrenciler için hash fonksiyonundan çıkan değer 10'dur. İki farklı anahtar değeri için hash fonksiyonundan çıkan değer aynı olduğu durumda çarpışma (collision) ortaya çıkmaktadır. Çarpışmanın çözülmesi gerekmektedir.

Folding: bu fonksiyonda key değeri adres uzayının kapasitesinin karşılayabileceği kadar haneye bölünür ve bölünen parçalar toplama işlemine tabi tutulur. Örneğin 123456789 key değerimiz olsun.

123
456
789
topla
258

Squaring: key değerinin karesi alınır ve yeni değerin belirli bir kısmı truncation işlemine tabi tutulur.

Afabetic Keys: Key değerini oluşturan karakterlerin ASCII kodlarından yararlanılabilir. Bu kodlar 4 işleme tabi tutularak hash fonksiyonu oluşturulabilir.

Hashing örneği:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>


#define Max_Satir 16


typedef struct _Satir
{     int index;
      int numara;
      struct _Satir *sonraki;
}Satir;


typedef struct _Tablo
{
      Satir satirlar[Max_Satir];
}Tablo;


Tablo hash_tablosu;


int Hash(int no)
{
      return no % Max_Satir;
}
```