

Kocaeli Üniversitesi

Bilgisayar Mühendisliği

Programlama Laboratuarı 2 Proje 2

**Dosya Organizasyonunda Hashing Uygulaması**

***Cihan Savaş 130202098***

***Cihat Duyku 150202106***

***Mehmet Samet Yıldız 150202104***

**Dosya Organizasyonunda Hashing Uygulaması**

1.CİHAN SAVAŞ(130202098), 2.CİHAT DUYKU(150202106),

3. MEHMET SAMET YILDIZ(150202104)

**Bilgisayar Mühendisliği Bölümü**

**Kocaeli Üniversitesi**

[cihansavas.85@gmail.com](mailto:cihansavas.85@gmail.com) , [cihatduyku@gmail.com](mailto:cihatduyku@gmail.com),

[msy3443@gmail.com](mailto:msy3443@gmail.com)

Özet

# Algoritmaların karşılaştırılması için 500 elemanlı bir rastgele (random) kayıt dosyası oluşturulacaktır. Kayıt dosyası öğrenci numarası(Kocaeli üniv . öğrenci on formatı), ad soyad( random 10 karakterli) bilgileri ile oluşturulmuştur. Oluşturulan 500 kayıt, Kayit.txt dosyasında tutulur bu bizim ana kayıt dosyamızdır. Kayıtlar, sırası ile fonksiyonlara girer ,her fonksiyonda farklı işlemler yapılarak 6 farklı bağıl dosyaya yazılmaktadır. 3 tane hash fonksiyonumuz vardır. Bunlar Bölen-Kalan,Kare Ortası ve Katlama dır. 2 tane ise çakışmaları gidermek için kullandığımız Ayrık Taşma ve Lineer yoklama yöntemleri bulunur.Her hash fonksiyon 2 durum için de kullanıldığı için 6 tane bağıl dosya elde edilmiştir. Her bir fonksiyon bloğunda yapılan işlem sayısı hesaplanmakta en son olarak adım sayılarını grafik ekranında kullanıcıya görsel olarak sunulmaktadır.

# 

# Giriş

Hash fonksiyonları bizlere bir kayıda en kısa zamanda ulaşmayı sağlamaktadır. Lineer bir aramada kullanılan adres yerine bağıl adresler oluşturulur ve hash fonksiyonlarına bu bağıl adreslere göre kayıt ve okuma işlemleri yapılır. Ancak bir bağıl adres birden fazla adres için oluşturulabilir. Böylece çakışmalar meydana gelir. Bu çakışmaları önlemek için Lineer Yoklama ve Ayrık Taşma yöntemleri kullanılmıştır.

* Bölen Kalan ve Lineer Yoklama
* Bölen Kalan ve Ayrık Taşma
* Katlama ve Lineer Yoklama
* Katlama ve Ayrık Taşma
* Kare ortası ve Lineer Yoklama
* Kare ortası ve Ayrık Taşma
* Lineer Arama

Kayıt dosyasının uğradığı fonkiyonlar yukarıdaki gibidir.Açıklamaları aşagıdadır.

Kayıt dosyasından üretilen bağıl dosyalar teker teker yukarıdaki gibi arama fonksiyonlarına gönderilimekte ve her işlem sonrası kayıtlar ekrana yazdırılıp adım sayıları tespit edilmektedir.

Bölen Kalan

Örnek bir numara: 1 1 0 2 0 2 0 0 1

**M**

**O**

**D**

**501**

**M**

**O**

**D**

**631**

bglAdr: 375 37

* 9 basamaklı öğrenci numarası alınır.
* Eğer Lineer Yoklama ise 631, Ayrık Taşma ise 501 e göre mod alınır.
* 631 sayısının sebebi doluluk oranını %80 yapmaktır. Bu performans açısından tavsiye edilen bir husustur. 500 kaydımız var. %80 doluluk oranının sağlanması için 500/0.8=625 sayısı bulunur.
* Sayımızın daha kararlı sonuçlar çıkarması için asal sayı olması tavsiye edilmiştir. 625 ten büyük en yakın asal sayı 631’dir.
* Ayrık taşma için bu yöntem söz konusu olmamıştır. Çünkü çakışmalar zaten ayrı bir yerde (501.bglAdresten sonra) tutulduğu ve çok sayıda çakışma gerçekleştiği için ayrıktaşma yönteminde %100 doluluk gözlenmemektedir.

Kare Ortası

kareOrt

dizi

En küçük sayı=110202001=>(110202001)2=1 2 1 4 4 4 8 1 0 2 4 4 0 4 0 0 1

**16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 İNDİS**

kareOrt

dizi

En büyük sayı=190202101=>(190202101)2=3 6 1 7 6 8 3 9 2 2 4 8 1 4 2 0 1

**16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 İNDİS**

* Sayının karesi alınır.
* En küçük sayıyıda alsak , en büyük sayıyıda alsak sonuç 17 basamaklı çıkmıştır.
* Karesi alınan sayı, dizi[17]’ye yazdırılır.
* Öyleyse dizimizin 7,8,9. İndisleri her zaman sayımızın kare ortası olmuş olur.
* 7,8,9. İndisler kareOrt[3] adlı bir diziye aktarılır.
* kareOrt[3] int’a dönüştürülür ve sayı max kayıt sayısına göre mod alınarak return edilir.

Katlama

Örnek bir numara: 1 1 0 2 0 2 0 0 1

**D**

**İ**

**Z**

**İ**

100

202

011

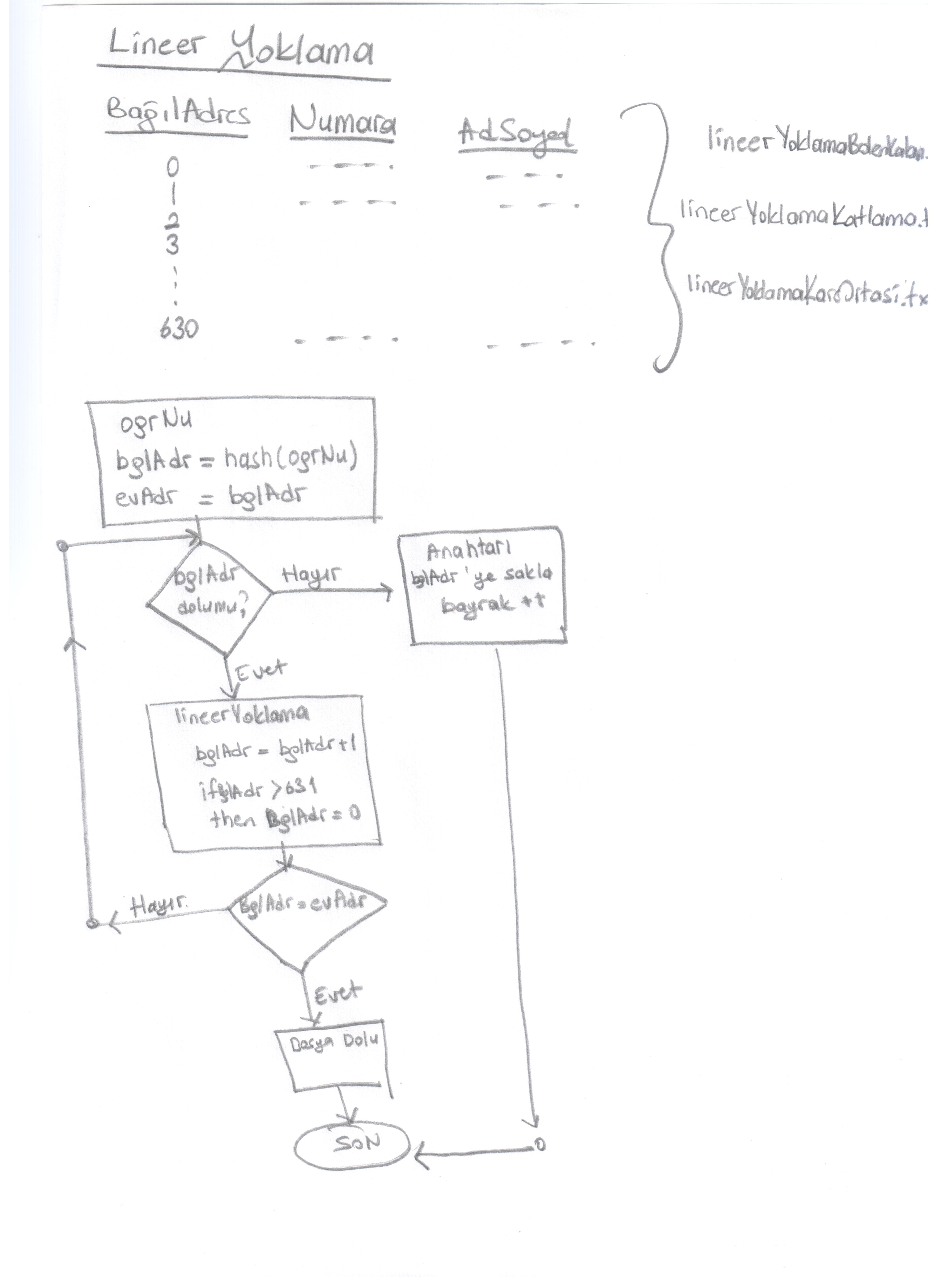
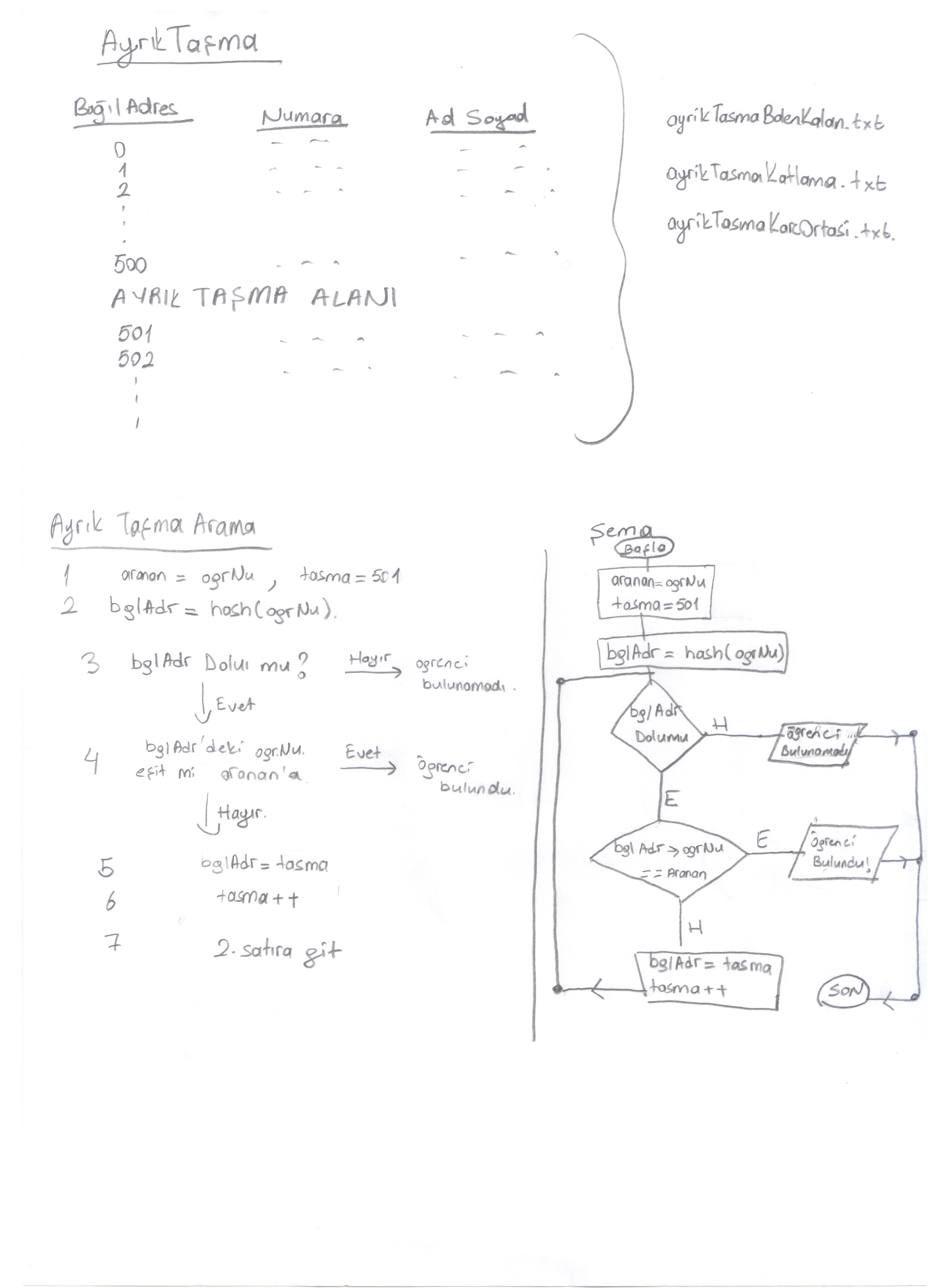
313

1 1 0 2 0 2 0 0 1

**8 7 6 5 4 3 2 1 0 İNDİS**

bagilAdr

* Sayıyı 9 basamaklı bir diziye aktarılır.
* Dizi indis numaralarına göre bir ters bir düz olmak üzere 3 parçaya ayrılır.
* Parçalar bas[3] orta[3] ve son[3] e aktarılır.
* Parçalar toplanır ve sonuç elde edilir.
* Sonuç her zaman 3 basamaklı olmak zorundadır. Eğer sayı 4 basamaklıysa 1000 e göre mod alınır böylece binler basamağı silinmiş olur.
* Son olarak sayı max kayıt sayısına göre mod alınarak return edilir.

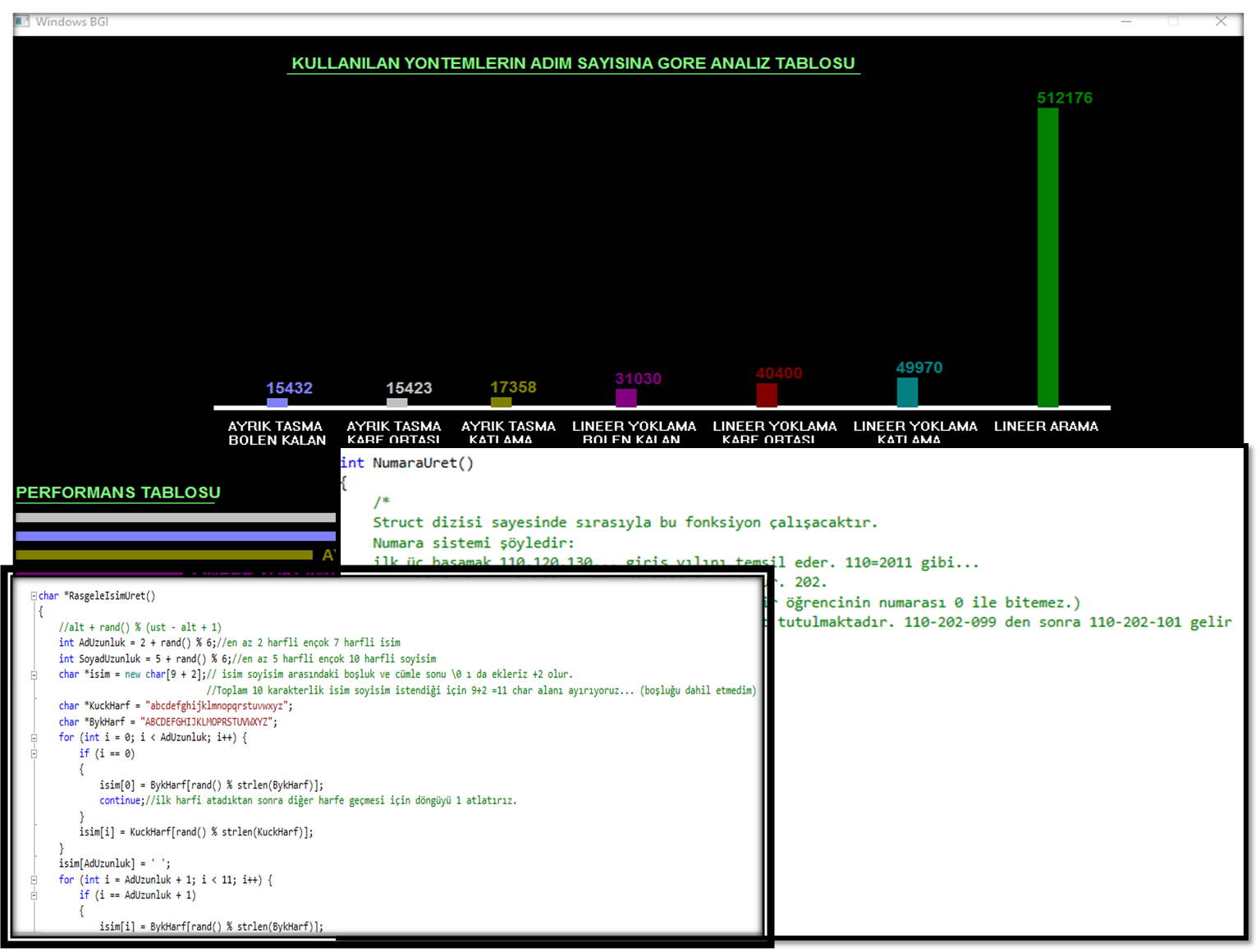


# PROGRAM’A AİT FONKSİYONLAR VE ÖRNEK GÖRÜNTÜLER

**FONKSİYONLAR**

* int NumaraUret()
* char \*RasgeleIsimUret()
* void anaKayitDosyasi(struct Kayit anaKayit[500])
* int bolenKalan(int numara,string cakismaTipi)
* int katlama(int numara,string cakismaTipi)
* int kareOrtasi(int numara,string cakismaTipi)
* void LineerYoklamaBolenKalan(struct Kayit lineer[500])
* int lineerYoklamaBolenKalanAra(struct Kayit lineer[500])
* void LineerYoklamaKatlama(struct Kayit lineer[500])
* int lineerYoklamaKatlamaAra(struct Kayit lineer[500])
* void LineerYoklamaKareOrt(struct Kayit lineer[500])
* int lineerYoklamaKareOrtAra(struct Kayit lineer[500])
* void AyrikTasmaBolenKalan(struct Kayit lineer[500])
* int ayrikTasmaBolenKalanAra(struct Kayit lineer[500])
* void AyrikTasmaKatlama(struct Kayit lineer[500])
* int ayrikTasmaKatlamaAra(struct Kayit lineer[500])
* void AyrikTasmaKareOrt(struct Kayit lineer[500])
* int ayrikTasmaKareOrtAra(struct Kayit lineer[500])
* int lineerAra(struct Kayit lineer[500])

# Sonuçlar



# Kullanıcı dostu uygulama olmasına özen gösterilmiştir.

* Hash fonksiyonların kullanılan çakışma yöntemlerine göre kayıt ve arama esnasında adım sayıları belirlenmiştir.
* Belirlenen adım sayılarına göre bir sıralama yapılmıştır ve en iyi performans gösteren metot belirlenmiştir.

# Alınan sonuçlar grafik ekranında listelenip kullanıcıya çıktısından inceleme olanağı sağlanmıştır.

**KAYNAKLAR:**

1. Çölkesen, R, “Hemen Herşeyiyle Standart C(ANSI C) ve Turbo C”, 3. baskı, Aralık 1996.
2. Robert Lafore,” Uzmanlar için Nesne Yönelimli C++ Programlama Kılavuzu”
3. www.stackoverflow.com