# Galatasaray Üniversitesi
# Bilgisayar Mühendisliği Bölümü
# INF224 - Veri Yapısı ve Algoritmalar

## 11 Aralık 2023
## TP 08 - Hashing

---

**Dikkat!**
– Soruların açıklamalarını dikkatlice okuyup, cevaplarınızı sizden istenen şekilde teslim edin.
– Dosya, değişken, vb. isimlerinde Türkçe veya özel karakter kullananlardan puan kıracağım.
– Programı derleme hatası tetikleyenler doğrudan sıfır alacaktır. Ödevinizi göndermeden önce kodlarınızın çalıştığını mutlaka kontrol edin, eksik olsa da düzgün şekilde çalışan bir kod gönderin. Gerekirse hatalı kısımları yorum satırına alıp, bana not yazın.
– Kopya durumunda başınıza ne geleceğine dair uyarı yapmama gerek yoktur herhalde...

---

Bir dizide herhangi bir elemana erişmek için karmaşıklığın $O(n)$ olduğunu ama erişmek istediğimiz elemanın indeksini bildiğimiz takdirde bunun $O(1)$ olduğunu biliyoruz, sene başından beri bol bol gördük. Hash tabloları bu prensibi kullanarak mevcut veri yapılarındaki eleman erişiminin hızlanmasını sağlar. Bunun için de hash fonksiyonları kullanılır. Bir hash fonksiyonu, mevcut elemanların bir takım sayı değerleri ile eşlenmesini sağlayan matematiksel fonksiyonlardır. Fonksiyon sonucu oluşan değerlerin özgün değerler olması, yani birden fazla elemanın aynı sayı ile eşleşmemesi hedeflenir.

Bu TP'de uydurduğumuz bir hash fonksiyonu kullanarak elimizdeki bir kitap içeriğini inceleyeceğiz. Bunun için size verilen 3 adet kaynak kod dosyası göreceksiniz:

– `hash.c` ve `hash.h` dosyaları hash tablosuna ait fonksiyonların tanımlandığı kütüphane dosyalarıdır.

– `main.c` ise, bildiğiniz gibi, ana programın çalıştığı dosyadır.

`hash.h` dosyası içerisinde bazı hash fonksiyonlarının ve yeni bir veri yapısının tanımlanmış olduğunu göreceksiniz. Bu veri yapısı şu şekilde verilmiştir:

```
1 typedef struct key {
2   char* word;
3   unsigned long count;
4   struct key *next;
5 } Key;
```

`Key` adındaki veri yapısı bir bağlı liste düğümüdür. Bu düğümün 3 özelliği vardır: kelime (`word`), bu kelimenin dosya içerisinde kaç kere geçtiği bilgisi (`count`) ve bir sonraki elemanın adresi (`next`).

Hash tablosunu oluşturmak için gerekli hash fonksiyonu ise `hash.c` içerisinde `hash_compute()` adıyla verilmiştir. Fonksiyon parametre olarak aldığı string tipindeki eleman için bir integer değer hesaplayıp bu değeri döndürür. Bu değeri hesaplarken, karakter dizisindeki her bir eleman için önceki elemanın hash değerini kullanır. Çarpım katsayısı, $x$, bir asal sayı seçilir, aşağıdaki örnekte 29 seçilmiştir.

```
dizi = "naber"
MAXBUCKETS = 1000
h0 = 0 * 29 + 110 = 110
h1 = h0 * 29 + 97 = 3287
h2 = h1 * 29 + 98 = 95421
h3 = h2 * 29 + 101 = 2767310
h4 = h3 * 29 + 114 = 80252104
h4% MAXBUCKETS = 104
```

`MAXBUCKETS` hash fonksiyonu sonucu sayıları tutan kova boyutunu verir. Daha önce tanımladığımız `Key` veri yapısındaki bir `table` dizisinde hash fonksiyonlarını tutacağız. Bu şekilde dizideki her bir elemanın indeksi hash fonksiyon sonucuna eş olacak. Mesela "`naber`" örneğini düşünecek olursak, indeksi 104 olacak, yani `table[104].word = "naber"` olacak.

Bu bilgileri göz önünde bulundurarak, sırayla istenen fonksiyonları `hash.c`'de tamamlayın:

1. `hash_lookup()` fonksiyonu parametre olarak aldığı bağlı liste yapısındaki tabloda, istenen string değerini arar. Bunun için :
   - öncelikle verilen string için hash fonksiyon değerini bulun.
   - sonra bu değerden itibaren dizide bir kontrol yapın ve kelimeleri karşılaştırın (strcmp).
   - değer bulunursa bulunan değeri döndürün.
   - değer bulunmasza ve parametre olarak verilen create değeri 1 ise elemanı ekleyin.
   - create 0 ise sadece NULL döndürün.

   Bu adımları takip ederek fonksiyonu kodlayın.
2. `hash_free()` fonksiyonu parametre olarak verilen bağlı liste yapısındaki tablo için bellekte açılan yeri boşaltır. Verilen tablonun boyutunun `MAXBUCKETS` ile tanımlandığını unutmayın. Ayrıca her bir elemanı free yapmadan önce taşıdığı word özelliğini bellekten boşaltmanız gerekmektedir. Bunları dikkate alarak fonksiyonu kodlayın.
3. `hash_dump()` fonksiyonu parametre olarak bir hash tablosu ve bir tam sayı değişken alır. Aldığı tablo içerisinde `count` değeri parametre olarak alınan tam sayı değişkenden büyük olan değerleri ekrana basar. `table` değiskeninde her bir eleman sayısını kontrol etmeniz gerektiğini unutmayın.

`main.c` ana programı çalıştırır. Ana program komut satırı üzerinden aldığı dosya adını okur (zip içindeki `book.txt` dosyasını kullanabilirsiniz). Bu dosyadaki kelimeleri sonsuz bir döngü içerisinde okur. Dosya sonuna gelince döngüden çıkar. Her yeni kelime okuduğunda `hash_lookup()` fonksiyonunu çağırır ve kelime için bir tam sayı değer hesaplar. Kelime `hash_table` tablosu içerisinde yoksa ekleme yapar. Döngüden çıktıktan sonra, yani tüm elemanları okuduktan sonra ekrana sayısı 500'ü geçen kelimeleri ve metin içerisinde kaç kez geçtiklerini basar. En sonunda da `hash_table` tablosu için bellekte açılan yeri boşaltır.

Ödev teslimi için `hash.c, hash.h, main.c` dosyalarını her zamanki gibi, `TP08_OgrenciNo_AdSoyad.zip` ismiyle zipleyip yüklemeniz gerekmektedir.