



ALGORİTMA ANALİZİ ÖDEV 2

DERS: ALGORİTMA ANALİZİ BLM

HAZIRLAYAN : YUNUS KARATEPE 17011051

ÖDEV KONUSU : HASHING ALGORİTMASI

YÖNTEM:

Problemimiz dosyaları n (dosya sayısı) sayısına bağılı olmayan bir karmaşıklıkta arama işlemi yapılabilecek bir map oluşturma işlemi. Biz yazdığımız c kodu ile hash-map yapısı oluşturuyoruz. Böylece arama işlemi yapmak istendiğinde o dosyanın içeriğine ait key değeriyle n den bağımsız bir c sabit sayısına bağılı olarak hızlı bir arama işlemi yapabilmış olacağız. Bu ödevde bize sadece hash-mapi oluşturma ve hash-mape ekleme kısımları sorulmuş

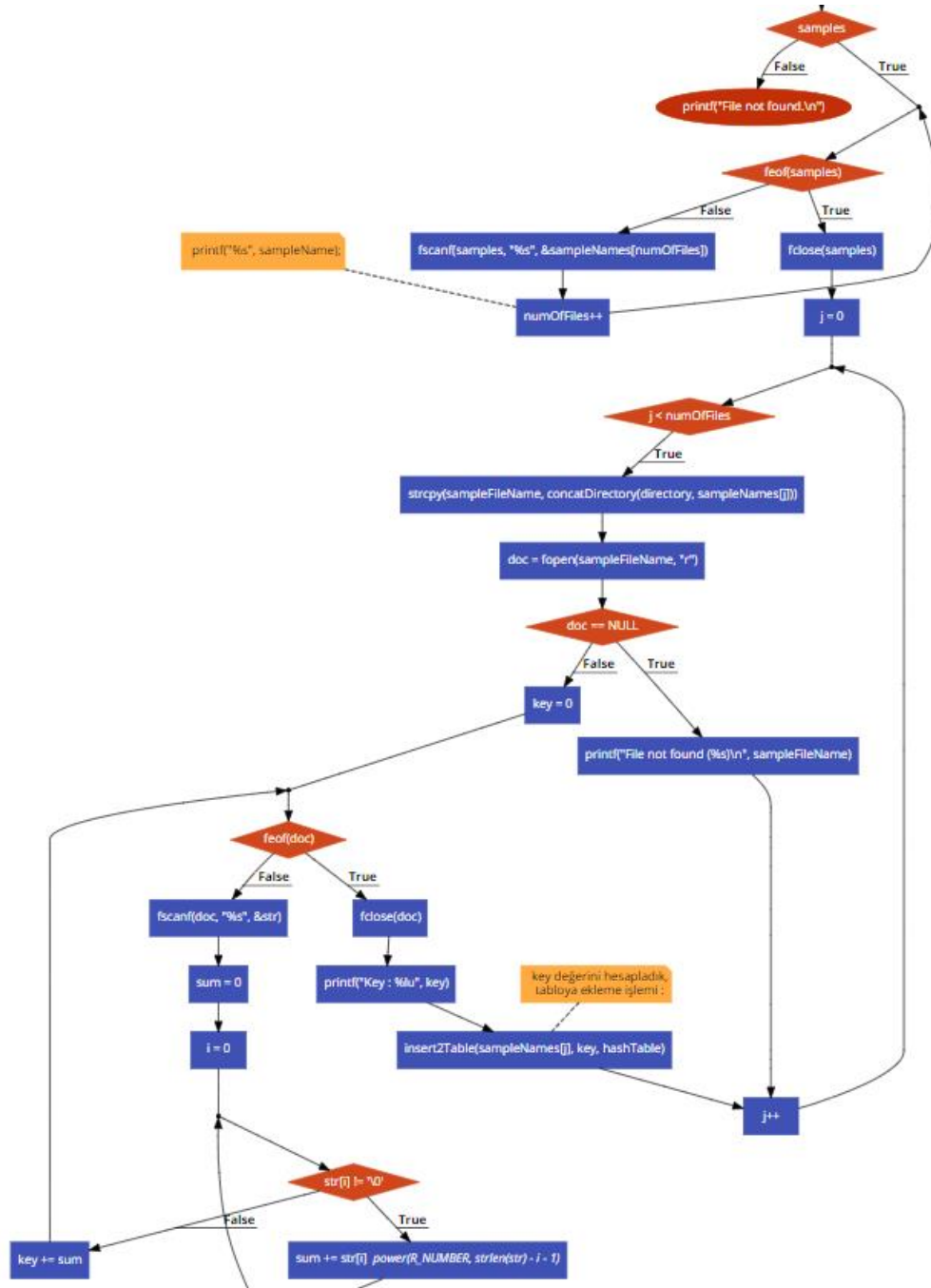
1-) HASH-MAP OLUŞTURMA:

Hash-map oluştururken kullanıcının kendisinin hesaplayıp #define TABLE_SIZE .. şeklinde define etmesi gerekmekte. Bu tablonun boyutunu alma işlemi. Bunu yaparken de :

$$\text{TabloUzunlugu} = \text{EnküçükAsalSayı} \geq \text{TablodakiElemanSayısı} / \text{LoadFactor}$$

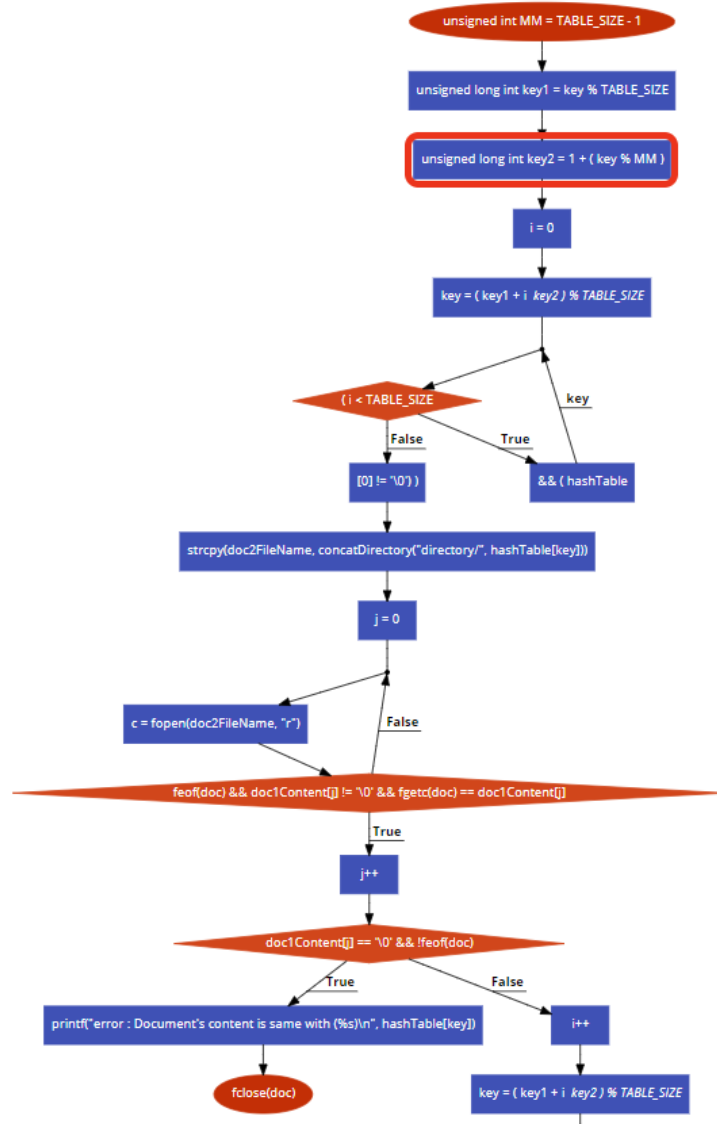
denklemini kullanıyoruz. Daha sonrasında ise “index.txt” dosyasının içinde “directory/” lokasyonunda olan dosyaların isimleri bulunuyor. Biz bu dosyaların içeriklerini kullanarak bir “key” değeri hesaplıyoruz ve buna göre hash-mapin ilgili kısmına dosyanın adını yazıyoruz.

Akış şeması:

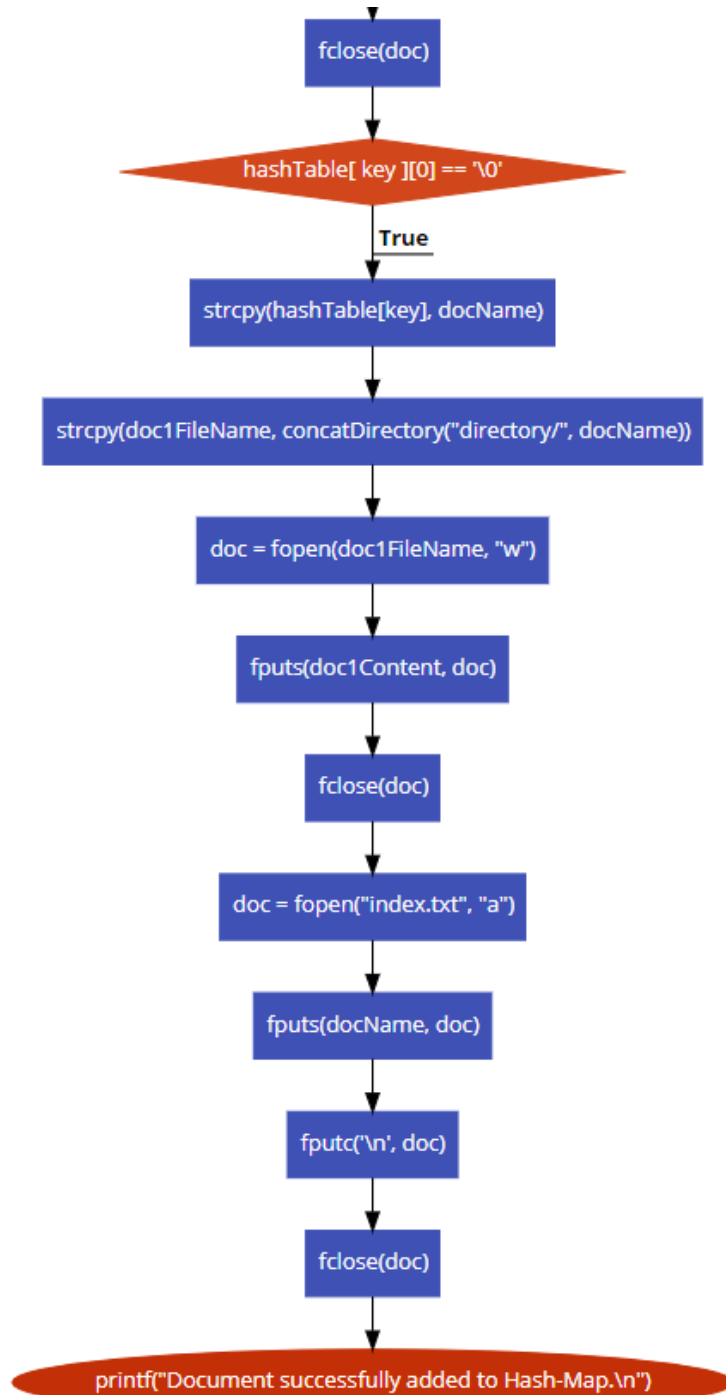


2-) HASH-MAPE VE VERİ TABANINA DOSYA EKLEME:

Bu işlemi gerçekleştirirken dosyanın içeriği ile aynı olan başka bir dosya var mı diye “index.txt” (isimler dosyası) ile veri tabanımızı kontrol ediyoruz. Eğer aynı içerikli dosya var ise işlemi sonlandırıyoruz. Bu işlemi yaparken en kötü ihtimal ile veri tabanında full collision oluşmuş olma durumudur. Bu durumda n adet dosya var ise n adet de dosya içeriği ile kontrol etmemiz gerekecektir. Eğer içerikten oluşturulan key değerinin adresi boş ise oraya dosyayı yazıp aynı şekilde veri tabanına ve “index.txt” dosyasına da isim olarak ekleyip işlemi sonlandırıyoruz.



-1-



UYGULAMA:

Örnek olarak A, B, C, D, E dosyaları veri tabanımızda bulunsun. 5 adet dosya bulunduğu için tablo uzunluğumuzu hesaplarken kullandığımızı formüle göre:

$\text{TabloUzunlugu} = \text{EnküçükAsalSayı} \geq \text{TablodakiElemanSayısı} / \text{LoadFactor}$

$\text{TabloUzunlugu} = 11 \geq 8 = 5 / 0.6$

$\text{TabloUzunlugu} = 11$ oluyor.

A = aaa

B = abc

C = Aaa

D = aal

E = aaa

olsun

keyA = 291

keyB = 294

keyC = 259

keyD = 302

keyE = 259

1-) A'yı 5. indise yerleştiriyoruz.

2-) Ardından B'yi 8. indise yerleştiriyoruz

3-) C'yi 6. indise yerleştiriyoruz.

4-) D'yi yerleştirirken, önce key = 5 oluyor. Daha sonrasında ise 5. indis dolu olduğu için yeniden key hesaplıyoruz i = 1 değerini almış oluyor böylece

yeni key = 8 oluyor. Burası da dolu durumda. Yeniden key hesaplıyoruz (i = 2)

yeni key = 0

D'yi 0. indise yerleştiriyoruz.

5-) E'yi yerleştirmeye çalışırken ise E'nin içeriği A ile birebir olduğu için yerleştirme yapamıyoruz.

HASH TABLE

Table 0 => D

Table 1 =>

Table 2 =>

Table 3 =>

Table 4 =>

Table 5 => A

Table 6 => C

Table 7 =>

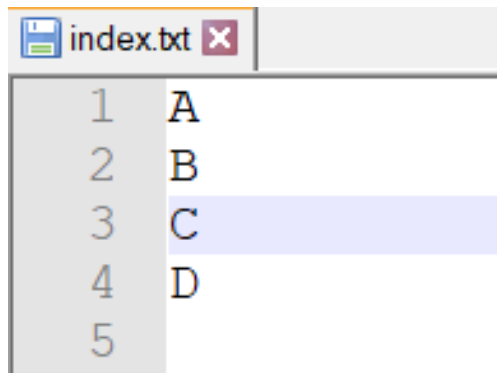
Table 8 => B

Table 9 =>

Table 10 =>

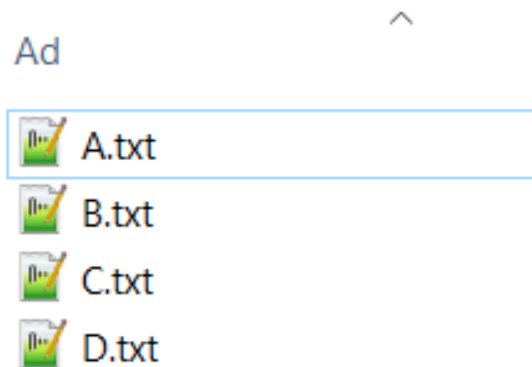
son durumda hash-table miz bu şekilde dolmuş oluyor.

index.txt dosyamızın ve directory klasörümüzün içi ise bu şekilde :



1	A
2	B
3	C
4	D
5	

index.txt



directory klasörü

SONUÇ:

Raporun “HASH-MAPE VE VERİ TABANINA DOSYA EKLEME” kısmında da belirtildiği üzere, en kötü durum eklemek için o an tabloda olan bütün elemanları gezdikten sonra boş alan bulup ekleme işlemidir. Eğer hash-mapte o anda bulunan eleman sayısına n dersek. Öyle bir keye sahip dosya gelecek ki en kötü durum olarak n adet kontrol yapmamız gerekecek. Bu durumda eleman ekleme yani insert()'in karmaşıklığı $O(n)$ olacaktır.

Tablo oluşturma karmaşıklığına bakacak olursak, en kötü durum olarak her seferinde yeni okunan eleman için o an tabloda bulunan eleman kadar kontrol yapmamız gerekecektir. Çünkü hepsinin keyleri çok düşük de olsa aynı gelme ihtimaline sahiptir. Bizim kullandığımız R sayısına göre bu ihtimal oldukça düşürülebilir ama mevcuttur.

En son durumda tabloda olan eleman sayısı = n kabul edilirse.

Son durumda $(n-1)$ defa kontrol yapılacaktır.

Sondan 1 önceki durumda $(n-2)$ defa kontrol yapılacaktır.

Sondan 2 önceki durumda $(n-3)$ defa kontrol yapılacaktır.

Sondan 3 önceki durumda $(n-4)$ defa kontrol yapılacaktır.

...

Sondan $n-1$ önceki durumda 0 defa kontrol yapılacaktır.

Yapılan toplam kontrol sayısı = $((n-1)*(n-2)) / 2$ olacaktır.

Bu da $O(n^2)$ karmaşıklık demektir.

Tablo oluşturma işlemi + ekleme işlemleri toplam karmaşıklığı = $O(n^2)$ olur.