ALGORİTMA ANALİZİ

HASHING

MİNE ELİF KARSLIGİL

ENES ÖZDOĞAN

20011056

YÖNTEM

PROBLEM

Sample.txt dosyasında birden fazla web sitesi adresi vardır. Verilen linklerle ulaşılan sitelerde geçen kelimeler link değerlerinin altındaki satırda verilmiştir. Bizden istenilen şey, öncelikle verilerin bu dosyadan C geliştirme ortamına uygun bir şekilde taşınıp sitenin içeriğinde geçen kelimeleri anahtar, adresleri ise değerler olacak şekilde tutan bir hash tablosu tasarlamaktır.

Verilerin Taşınması ve Depolanması

LINKS adında bir struct yapısında dosyadan alınan link değerleri links dizisinde tekil olarak tutulur. Content dizisinde ise bu linkte geçen kelimelerin tümü vardır. Main fonksiyonunda dosyadan okuma işlemleri gerçekleştirilir.

```
typedef struct n{
    char links[max];
    char content[max];
}LINKS;
```

Kelimelerin Bölütlenmesi ve Link Ataması

KEY_WORDS adında bir struct yapısı ile word_key dizisinde anahtar kelime, arr_of_links iç struct'ında ise linkler barındırılır. Link_no değişkeni ile ilgili anahtar kelimenin kaç adet sitede geçtiği tutulur. Open değişkeni link birleştirirken kullanılır. Try_time ile de hash tablosunda kaçında denemede bulunulduğu tutulur.

```
typedef struct m{
    char word_key[max];
    arr_of_links arr[5];
    int link_no;
    int open;
    int try_time;

}KEY_WORDS;
```

Linklerin Tutulması

```
typedef struct z{
    char key_links[max];

}arr_of_links;
```

Fonksiyonlar ve Görevleri

Main'de dosya alım işlemleri sonrasında ilk olarak kelime bölütleme ve link atama görevini yapan split_to_words() fonksiyonu çağrılır.

```
void split_to_words(LINKS *str_l,KEY_WORDS *key_words);
```

Bu fonksiyonda alt fonksiyon olarak bir 2 boyutlu bir string dizisi döndüren split fonksiyonu çağrılır. Döngü içerisinde dönen her kelime key_words[].word_key değerine atılır. Ardından bağlı olunan link değeri de key_words[].arr[].key_links kısmına kopyalanır.

Kelimeler bölütlendikten sonra aynı word_key değerine sahip kelimeler için link birleştirmesi yapılması için find_key_in_struct() fonksiyonu çağrılır.

```
void find_key_in_struct(KEY_WORDS *key_words,int k);
```

Tüm key_word değerleri gezilerek aynı word_key değerine sahip adaylar indis kontrolü ve open yani kontrol edilmişlik değerine bakılarak karşılaştırılır. Open değeri -1 olmayanlar diğer şartları sağladığında diğer indiste bulunan link değerleri öbür adaya kopyalanır ve kopyalanan aday kontrol edildiği için open değeri -1 yapılır.

Bu adımdan sonra verimiz hash tablosuna yerleştirilmek için hazırdır

```
void create_hash_table(KEY_WORDS* hash,KEY_WORDS* key_words,float load_factor,int table_size );
```

Bu fonkiyonda horner metodunu kullanan get_horner fonksiyonu ile key_index hesaplanır. Eğer çakışma yaşandıysa linear probing ile hash tablosu gezilir. Çakışma yaşanmazsa try_time değeri bir olarak atanır. Yaşanırsa try_time, ilerleme sayısı değerine eşitlenir. Hash tablosu load factor oranına göre oluşturulur ve key değerleri atanır.

Hash tablosunu oluşturduktan sonra arama yapabiliriz.

```
void get_value(KEY_WORDS* hash,int table_size);
```

Öncelikle buffer dizisine kullanıcıdan yapılacak sorgu alınır. Sorguda sscanf ile ayıklama işlemi ve bağlaç kontrolleri yapılır. Ve, veya tek kelime durumlarına göre program dallanır. Her dallanmada gey_key_horner ile key_index hesaplanır. Sorguya göre hash tablosuna girilen key_index değerleriyle ortaya çıkan link değerleri geçici bir struct yapısına kopyalanır ve ekrana bastırılır.

Detay modda kullanılacak olan yazdırma işlemleri için kullanılan print_hashTable()

```
void print_hashTable(KEY_WORDS* hash,int table_size);
```

Bu fonksiyon ile tablo boyutu, hash tablosunun her indisinde bulunan değerler ve null değer yoksa kaç denemede ilgili elemanın yerleştiği bilgileri bastırılır.

```
int find_table_length(int table_size);
```

Tablo boyutunun load factor hesaplaması ardından asal sayıya atan yapan fonksiyondur.

UYGULAMA

DETAY MOD

Programın detay modda çalıştırılmasından elde edilen çıktıdan farklı kesitler:

```
Detay 0,Normal 1 :0
--TABLE SIZE:563--
1.Anahtar: Deneme:0

2.Anahtar: Deneme:0

3.Anahtar:Computers Deneme:1

1.link: https://ce.yildiz.edu.tr

2.link: https://www.tutorialspoint.com

3.link: https://www.udemy.com

4.Anahtar:Entertainment Deneme:1

1.link: https://www.instagram.com

2.link: https://www.rottentomatoes.com

3.link: https://www.youtube.com
```

```
13.Anahtar:AI
 Deneme:4

1.link: https://www.coursera.org

14.Anahtar:RealEstate Deneme:5

1.link: https://www.sahibinden.com

15.Anahtar:University Deneme:6

1.link: https://ce.yildiz.edu.tr

16.Anahtar:Education
 Deneme:7

1.link: https://ce.yildiz.edu.tr

2.link: https://www.udemy.com

3.link: https://medium.com

17.Anahtar:Dataset Deneme:8

1.link: https://www.kaggle.com
```

```
7.Anahtar: Deneme:0

8.Anahtar:Coding Deneme:1

1.link: https://www.tutorialspoint.com

9.Anahtar:News
 Deneme:1

1.link: https://edition.cnn.com

2.link: https://www.youtube.com

3.link: https://weather.com
```

```
31.Anahtar:Business
 Deneme:11

1.link: https://www.linkedin.com

32.Anahtar:Education Deneme:12

1.link: https://www.coursera.org

2.link: https://www.youtube.com

33.Anahtar:Cars Deneme:13

1.link: https://www.sahibinden.com

34.Anahtar:Motorcycles
 Deneme:14

1.link: https://www.sahibinden.com
```

NORMAL MOD

Veya Örneği

```
Aramak istediginiz kelime/kelimeler:Movies veya AI
1. link: https://www.rottentomatoes.com
2. link: https://www.imdb.com
3. link: https://www.netflix.com
1. link: https://ce.yildiz.edu.tr
2. link: https://www.kaggle.com
```

Tekil Arama Örneği

```
Aramak istediginiz kelime/kelimeler:Movies
1.link: https://www.rottentomatoes.com
2.link: https://www.imdb.com
3.link: https://www.netflix.com
```

Olumlu ‘ve’ Araması

```
Detay 0,Normal 1 :1
Aramak istediginiz kelime/kelimeler:IT ve AI
1.link: https://ce.yildiz.edu.tr
```

Olumsuz ‘ve’ Araması

```
Aramak istediginiz kelime/kelimeler:SocialNetwork ve Coding
Ortak link yok
```

Var Olmayan Kelime/Kelimeler Araması

```
Detay 0,Normal 1 :1
Aramak istediginiz kelime/kelimeler:araba
Kelime yok
Cikmak icin 0, devam icin 1 bas:1
Detay 0,Normal 1 :1
Aramak istediginiz kelime/kelimeler:araba veya IT
Birinci kelime bulunamadi
1. link: https://ce.yildiz.edu.tr
2. link: https://www.udemy.com
3. link: https://www.coursera.org
Cikmak icin 0, devam icin 1 bas:1
Detay 0,Normal 1 :1
Aramak istediginiz kelime/kelimeler:IT veya araba
1. link: https://ce.yildiz.edu.tr
2. link: https://www.udemy.com
3. link: https://www.coursera.org
ikinci kelime bulunamadi
Cikmak icin 0, devam icin 1 bas:1
Detay 0,Normal 1 :1
Aramak istediginiz kelime/kelimeler:araba ve dikey
Kelimeler bulunamadi
Cikmak icin 0, devam icin 1 bas:0

--------------------------------
Process exited after 66.39 seconds with return value 0
Press any key to continue . . .
```

SONUÇ

Load Factor Değişimi

Load factor değeri arrtıkça tablomuzun anahtar sayısı/boyut oranı büyüyeceği için tablo boyutundaki artımdan dolayı 0.1 değerinde en az, 0.9 değerinde en fazla çakışma olduğunu gözlemledim.

Hash Tablo Karmaşıklığı

En iyi durumda anahtarları yerleştirirken çakışma olmazsa yerleştirme O(1) karmaşıklığındadır. En kötü durum için yerleştirmede çakışma durumu gözlenir ve n adet kontrol yapılırsa karmaşıklık O(n) olur. n adet eleman yerleştirileceği için karmaşıklık en iyide O(n) , en kötüde $O(n^2)$ olur.

Arama Karmaşıklığı

Horner metodu ile key_index hesaplanması O(1), en iyi durumda çakışma olmayacağı için karmaşıklık O(1) olur. En kötü durumda linear probing yöntemi ile çakışma önlediğimiz için karmaşıklık O(n) olur. Eğer hashing yerine brute force yöntemi kullanılsaydı en iyi, en kötü, ortalama değer farketmeksizin O(n) karmaşıklığında lineer arama yapılarak problem çözülebilirdi.

VİDEO LİNKİ: https://youtu.be/QgBICfxH_zA