TÜRKİYE CUMHURİYETİ
YILDIZ TEKNİK ÜNİVERSİTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

ALGORİTMA ANALİZİ
ÖDEV - 3

Öğrenci No: 20011055
Öğrenci Adı Soyadı: Şeymanur KORKMAZ
Öğrenci e-posta: seymanur.korkmaz@std.yildiz.edu.tr

Ders/Grup: BLM3021- Algoritma Analizi / Gr-1

Ders Yürütücüsü:
Mine Elif KARSLIGİL
Kasım, 2022

İçindekiler

1. YÖNTEM

a. Problem

Verilen dosyadan okunan verilerle basit bir arama motoru algoritması oluşturmak.

b. Çözüm

Dosyadan okunan linkler value olarak, bu linklerin içinde geçen kelimeler ise key olarak kabul edilerek bir hash tablosu oluşturulur. Bu sayede aranan kelimenin bulunduğu linkler O(1) karmaşıklığıyla bulunabilir.

- **void ReadFromFile(DATA **data, int *uniqueWordNumber, int *wordNumber):**
  Dosyada verilen kelimeler ve içinde bulunduğu linkler data adlı struct dizisinde tutulur. Eşsiz ve toplam kelime sayısı bulunur.

- **int FindHashSize(int uniqueWordNumber, float loadFactor):**
  Eşsiz kelime sayısının kullanıcıdan alınan load faktöre bölünmesiyle bulunan sayıdan büyük en küçük asal sayı tablo boyutunu belirler.

- **void createTable(DATA *data, HashTable **Hash, int hashSize, int wordNumber):**
  Horner methodu yardımıyla kelimenin yerleşeceği key blunur ve o adrese kelime linki ile birlikte yerleştirilir. Eğer kelime daha önceden tabloya eklenmeişse yalnızca link dizisine yeni link eklenir.

- **void initialization(HashTable **Hash,int hashSize):**
  Hash tablosundaki flag ve linkNum değerleri 0, stepNum değeri -1 olarak ilklendirilir.

- **unsigned long hornerMethod(char str[]):**
  Horner Method yardımıyla gelen stringin key olarak karşılığı bulunur.

- **void SearchEngine(HashTable *Hash, int hashSize, char words[]):**
  Kullanıcının arama için girdiği stringi uygun şekilde parçalayarak aranan kelimeler için ayrı ayrı SearchinHash() fonksiyonu çağrılır. Kelimeler arasında kullanılan bağlaca göre Union() veya Intersection() fonksiyonlarını çağrılır.

- **int SearchinHash(HashTable *Hash, int hashSize, char word[]):**
  Girilen kelimenin key değeri bulunarak O(1) karmaşıklıkla tablodaki adresine erişilir.

- **void Union(HashTable *Hash, int hashKey1, int hashKey2):**
  Kelimeler arasında ‘or’ bağlacı kullanıldıysa kelimelerin link dizilerinin birleşimini ekrana yazdırır.

- **void Intersection(HashTable *Hash, int hashKey1, int hashKey2):**
  Kelimeler arasında ‘and’ bağlacı kullanıldıysa kelimelerin link dizilerinin kesişimini ekrana yazdırır.

2.UYGULAMA

a. Normal Mod

```
Enter the load factor : 0.5

1.NORMAL MODE
2.DETAILED MODE
3.EXIT
->SELECT MODE : 1

Enter the word/words you want to search : Movies
- https://www.rottentomatoes.com
- https://www.imdb.com
- https://www.netflix.com

Press any key to continue...
```

```
Enter the load factor : 0.5

1.NORMAL MODE
2.DETAILED MODE
3.EXIT
->SELECT MODE : 1

Enter the word/words you want to search : entertainment and series
- https://www.rottentomatoes.com
- https://www.imdb.com
- https://www.netflix.com

Press any key to continue...
```

```
Press any key to continue...
1.NORMAL MODE
2.DETAILED MODE
3.EXIT
->SELECT MODE : 1

Enter the word/words you want to search : it or dataset
- https://ce.yildiz.edu.tr
- https://www.udemy.com
- https://www.coursera.org
- https://leetcode.com
- https://www.kaggle.com

Press any key to continue...
```

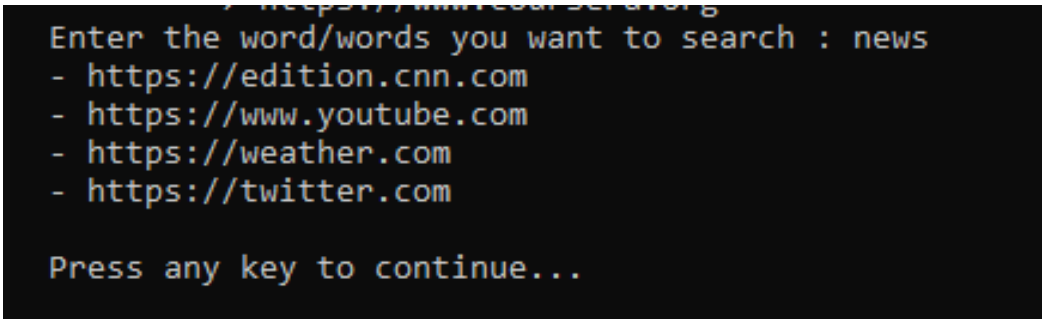
b. Detaylı Mod

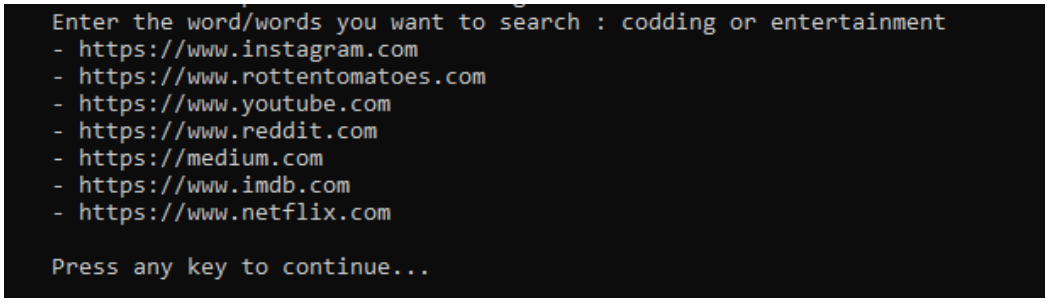
```
24 - university ------ settled in the table at the  1th  attempt
        -> https://ce.yildiz.edu.tr
25 - competition ------ settled in the table at the  2th  attempt
        -> https://www.kaggle.com
        -> https://leetcode.com
26 - physics ------ settled in the table at the  1th  attempt
        -> https://www.udemy.com
27 -
28 - realestate ------ settled in the table at the  3th  attempt
        -> https://www.sahibinden.com
29 - computers ------ settled in the table at the  1th  attempt
        -> https://ce.yildiz.edu.tr
        -> https://www.tutorialspoint.com
        -> https://www.udemy.com
        -> https://leetcode.com
30 - blogs ------ settled in the table at the  1th  attempt
        -> https://medium.com
31 -
32 -
33 - series ------ settled in the table at the  1th  attempt
        -> https://www.rottentomatoes.com
        -> https://www.imdb.com
        -> https://www.netflix.com
34 -
35 -
36 -
37 -
38 -
39 -
40 -
41 - dataset ------ settled in the table at the  1th  attempt
        -> https://www.kaggle.com
42 -
43 -
44 -
45 -
46 - cars ------ settled in the table at the  1th  attempt
        -> https://www.sahibinden.com
        -> https://www.motors.co.uk
47 -
48 - reviews ------ settled in the table at the  1th  attempt
        -> https://www.rottentomatoes.com
        -> https://www.imdb.com
49 -
50 -
51 - it ------ settled in the table at the  1th  attempt
        -> https://ce.yildiz.edu.tr
        -> https://www.udemy.com
        -> https://www.coursera.org
        -> https://leetcode.com
52 - education ------ settled in the table at the  2th  attempt
        -> https://ce.yildiz.edu.tr
        -> https://www.udemy.com
        -> https://www.youtube.com
        -> https://medium.com
        -> https://www.coursera.org
Press any key to continue...
```

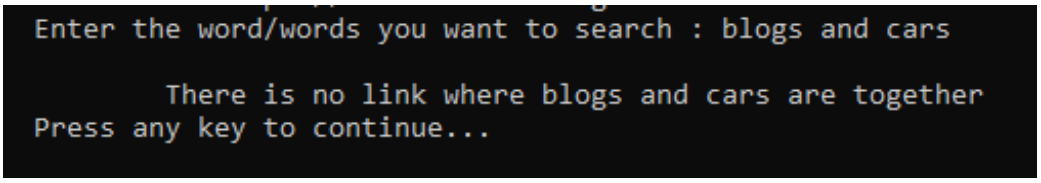
```
Enter the load factor : 0.5

1.NORMAL MODE
2.DETAILED MODE
3.EXIT
->SELECT MODE : 2

LENGTH OF HASHE TABLE => 53
0 -
1 - coding ------ settled in the table at the  1th  attempt
        -> https://www.tutorialspoint.com
2 -
3 - socialnetwork ------ settled in the table at the  1th  attempt
        -> https://www.instagram.com
        -> https://www.reddit.com
        -> https://twitter.com
        -> https://www.linkedin.com
4 - ai ------ settled in the table at the  1th  attempt
        -> https://ce.yildiz.edu.tr
        -> https://www.kaggle.com
        -> https://www.coursera.org
5 - r&d ------ settled in the table at the  2th  attempt
        -> https://ce.yildiz.edu.tr
6 -
7 -
8 -
9 -
10 - tutorials ------ settled in the table at the  1th  attempt
        -> https://www.tutorialspoint.com
11 -
12 -
13 -
14 - e-trade ------ settled in the table at the  1th  attempt
        -> https://www.amazon.com
        -> https://www.hepsiburada.com
15 - movies ------ settled in the table at the  1th  attempt
        -> https://www.rottentomatoes.com
        -> https://www.imdb.com
        -> https://www.netflix.com
16 - entertainment ------ settled in the table at the  1th  attempt
        -> https://www.instagram.com
        -> https://www.rottentomatoes.com
        -> https://www.youtube.com
        -> https://www.reddit.com
        -> https://medium.com
        -> https://www.imdb.com
        -> https://www.netflix.com
17 - business ------ settled in the table at the  2th  attempt
        -> https://www.linkedin.com
18 - motorcycles ------ settled in the table at the  2th  attempt
        -> https://www.sahibinden.com
19 - news ------ settled in the table at the  1th  attempt
        -> https://edition.cnn.com
        -> https://www.youtube.com
        -> https://weather.com
        -> https://twitter.com
20 -
21 - cloud ------ settled in the table at the  1th  attempt
        -> https://www.kaggle.com
22 -
23 -
```







3.SONUÇ

a. Load Factor'deki Değişikliğin Sonuca Etkisi

Load faktör değeri küçüldükçe tablo boyutu büyüdüğünden çakışmanın ciddi oranda azaldığını gözlemledim. Load faktör küçüldükçe hızda artış görülmesinin yanında negatif bir etki olarak kullanılan bellek miktarı da artacakatır.

b. Hash Boyutunu Hesaplayan Kodun Karmaşıklığı

```
int FindHashSize(int uniqueWordNumber, float loadFactor){
    int hashSize = uniqueWordNumber/ loadFactor;
    int flag=0;
    int i;
    while(flag == 0){
        hashSize++;
        if(hashSize%2 != 0 && hashSize%3 != 0){
            for(i=5;i<sqrt(hashSize);i=i+6){
                if(hashSize%i != 0 && hashSize%(i+2) != 0)
                flag=1;
            }
        }
    }
    return hashSize;
}
```

c. Arama Motoru Karmaşıklığı

```
void SearchEngine(HashTable *Hash, int hashSize, char words[]){
    char *token[3];
    char *w = strtok(words," ");
    int i=0;
    int hashKey_word1,hashKey_word2;
    while(w != NULL){
        token[i] = w;
        w = strtok(NULL, " ");
        i++;
    }
    if(strcmp(token[1],"or") != 0 && strcmp(token[1],"and") != 0){
        hashKey_word1 = SearchinHash(Hash,hashSize,token[0]);
        if(hashKey_word1 != -1){
            for(i=0;i<Hash[hashKey_word1].linkNum;i++)
                printf("         - %s\n",Hash[hashKey_word1].links[i]);
        }else{
            printf("\n      Word not found!!");
        }


    }else if(strcmp(token[1],"or") == 0 || strcmp(token[1],"Or") == 0){
        hashKey_word1 = SearchinHash(Hash,hashSize,token[0]);
        hashKey_word2 = SearchinHash(Hash,hashSize,token[2]);
        Union(Hash,hashKey_word1,hashKey_word2);
    }else if(strcmp(token[1],"and") == 0 || strcmp(token[1],"And") == 0){
        hashKey_word1 = SearchinHash(Hash,hashSize,token[0]);
        hashKey_word2 = SearchinHash(Hash,hashSize,token[2]);
        Intersection(Hash,hashKey_word1,hashKey_word2);
    }

}
```

```
int SearchinHash(HashTable *Hash, int hashSize, char word[]){
    int i,hashKey;
    int stepNum = 0;
    hashKey = CalculateHash(hornerMethod(word),hashSize);

    do{
        hashKey = (hashKey+stepNum) % hashSize;
        stepNum++;

    }while(Hash[hashKey].flag == 1 && strcmp(Hash[hashKey].word,word) != 0);

    if(strcmp(Hash[hashKey].word,word) == 0)
        return hashKey;
    else
        return -1;
}
```

Tek bir kelime aradığımız durumlarda karmaşıklık O(1) olur. Birden çok kelime aradığımız durumlarda ise key bulma işlemi yine O(1) karmaşıklıkta olur fakat kesişim ve birleşim kodları hesaba katıldığında karmaşıklık $O(N^2)$ olur.

Video linki

https://youtu.be/LWmXYjZUx_o