Algoritma Analizi 3.Ödev Raporu

Amacımız text dosyasında bize verilmiş olan siteler ve sitelerin içinde neler olduğunu dair bilgileri tutan cümleleri kullanarak,input olarak bir kelime verildiğinde o kelimenin geçtiği siteleri getirmek. Tabiki arama yaparken ve,veya gibi bağlaçlarada dikkat ederek siteleri getirmek. Bunun için öncellikle hash tablosunu nasıl dizayn edeceğimize kara vermemiz gerekiyor. Ben sitelerin key olamayacağına karar verdim çünkü aramaları kelimeler bazlı yapacağımız için siteleri key yapmak mantıksız kalıyor bu yüzden sitenin içinde neler olduğunu söyleyen kelimeleri key yapmak siteleri value olarak tutmak daha mantıklı. Fakat site içerikleri birden fazla kelimeden oluştuğu için her kelime için bir adet key oluşturmak daha mantıklı geldi. Yani;

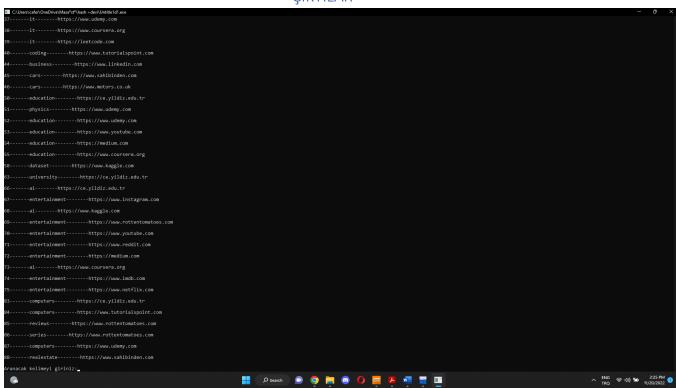
```
www.instagram.com
kişilik fotoğraf video
datası için;
key-> kisilik
value->www.instagram.com
key-> fotograf
value->www.instagram.com
key-> video
value->www.instagram.com
şeklinde hashtableda tutulucaklar. Hash tablosunu oluşturmaya gelirsek item tutmak için;
struct item {
  char key[50];
  char value[50];
};
Şeklinde bir yapı kullandım.Bu itemleride bir item dizisinde tuttum.
struct item **hash_table
```

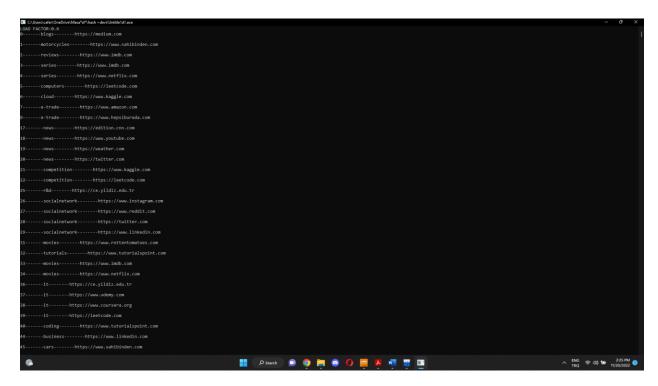
Yapıları oluşturduktan sonra yapılıcak ilk şey kaç adet key value ikilisine sahip olduğumuza bakmak çünkü

Hashtablosunu buna göre oluşturucaz.Bunun için say() adında bir fonksiyon kullanıyoruz.Bu fonksiyon text dosyasına gidip siteleri tarif etmek için yazılan cümleleri pars edip her kelime için sayma yapıyor ve bize int olarak geri dönüyor. Daha sonra kullanıcıdan load_Factor verisi float olarak alıyoruz.Elimizde kaç adet element olduğunu bulduk ,load_factor verisinide aldığımıza göre alfadan m(listenin)uzunluğunun kaç olması gerektiğini asal_bul() adlı fonksiyon yardımıyla load_factor verisinie en yakın asıl sayıyı bularak belirliyoruz.Hash tablosu uzunluğu bulunduktan sonra insert() adlı fonksiyonu kullanıyoruz.Bu

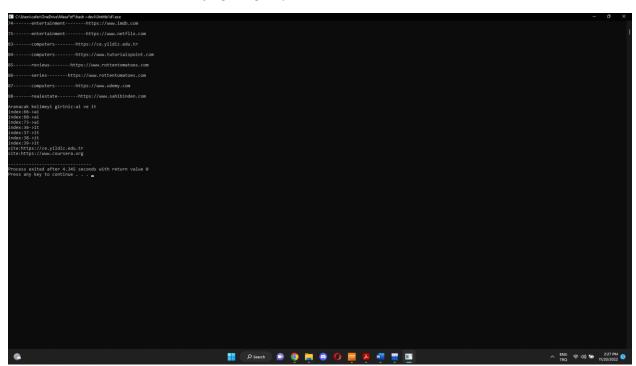
fonksiyon say() fonksiyonunda olduğu gibi baştan itibaren site içirisini, tarif eden her kelime için bir key value oluşturup bunu linear probing kullanarak yerleştiriyor.İnsert işlemi bittikten sonra aranacak kelime ya da kelimeleri kullanıcıdan alıyoruz.Bu aşamadan sonra 3 adet olasılık ele alıyoruz.Eğer kullanıcı sadece 1 adet kelime girerse ki bunu strtok() yapısı ile yapıyoruz.Tek kelime için search() fonksiyonu çalışıyor ve kelimenin keçtiği sitelerin indexsini bize dönüyor.Eğer bir bağlaç kullanıldıysa ve ve veya için farklı dallanmalarla iki durum içinde search() fonksiyonu çalışıp eğer bağlaç veya ise her iki aranan kelime için de gelen indexleri direk yazdırıyoruz eğer ve bağlacı kullanıldıysa 2 listenin sadece ortak olan siteleri yazdırılır.

ÇIKTILAR

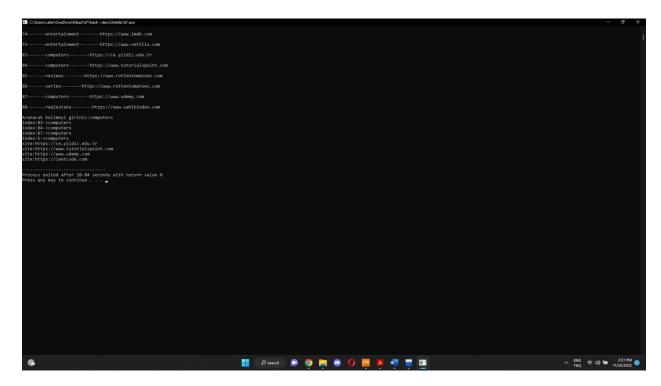




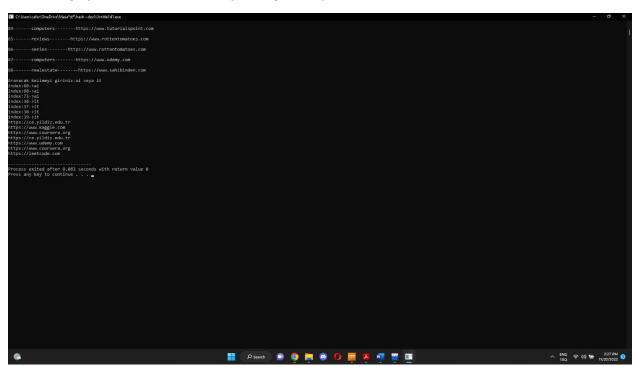
Burada hashtablosunun nasıl oluştuğunu görüyoruz.



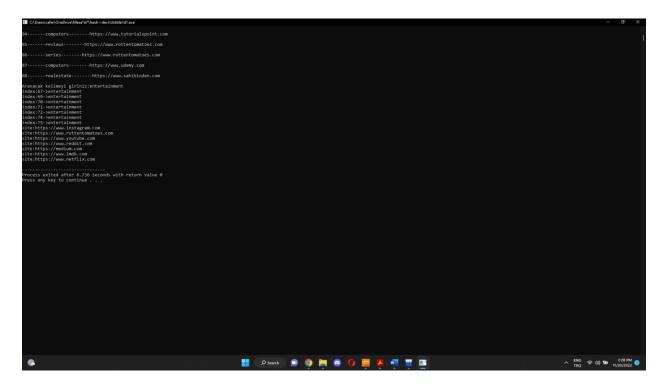
Burada ve bağlacı için çıktıları görüyoruz.(indexlerle).



Burada bağlaç olmadan tek kelimelik çıktılar gösteriliyor.



Burada ve bağlacı için çıktılar gösteriliyor.



Burada tek kelime içinçıktılargösteriliyor.

HASH Fonksiyonun Karmaşıklığı

Fonksiyon tek bir for döngüsü içeriyor ve stringin uzunluğu kadar dönüyor bu yzüden en kötü olasılıkla O(N) karmaşıklık oluyor.

LOAD FACTOR Değişimi

Load factor azaldıkça hash tablosu size değeri büyüyor çakışma ihtinmali daha da azalıyor çünkü çok boş slot oluyor.Load factor 1 e yaklaştıkça hash tablosu size değeri eleman sayısına eşit oluyor ve daha fazla çakışma oluyor.