



## VI. GENEL YAPI

### A. Kullanıcı Kısmı

Program çalıştığında ilk olarak karşımıza kullanıcının seçim yapabileceği bir menü çıkmaktadır.

Burda Suffix Tree oluşturma işlemi yapıldıktan sonra istediğiniz işlem gerçekleştirilebilir.

```
Suffix Tree Program:
1.Veri Dosyasında ki Suffix'i build ve printle.
2.Gireceğiniz substring dosyada ki stringte var mı?
3.Tekrarlanan en uzun substring'i bul.
0.Cikis
Hangi islemi yapmak istersiniz:
```

Menüde karşımıza 4 buton çıkmaktadır. Bu butonlar:

- 1) Buton: Veri Dosyasında ki Suffix'i build ve printle.

```
$ [10]
bil [-1]
$ [7]
isimbil$ [0]
i [-1]
lisimbil$ [1]
mbil$ [5]
simbil$ [3]
lisimbil$ [-1]
$ [1]
mbil$ [6]
simbil$ [4]
```

- 2) Buton: Gireceğiniz substring dosyada ki stringte var mı işlemi.

```
Hangi islemi yapmak istersiniz:2
bili
<bili>Substringi bulunuyor ve 1 defa tekrarlanıyor
```

- 3) Buton: Tekrarlayan En uzun Substringi bulmak.

```
Hangi islemi yapmak istersiniz:3
Tekrarlanan en uzun substring: bil
```

- 4) Buton: Çıkış butonu.

### B. Kod Kısmı

Kod kısmına baktığımızda ise en üstte yapacağımız Node yapısı için struct tanımlamak bulunmaktadır.

Bundan sonra programın çalışması ve yazılabilirliğini kolaylaştırmak için tam onbir tane fonksiyon gelmektedir. Bunlar yukarda bahsettiğim butonların işlemlerini yapmaktadır.

Programın çalışması int main() kısmında test edilir. Kullandığım fonksiyonlar sadece menü de ki işlemleri gerçekleştirmeye yarar.

**Node \*newNode()**

Yeni node oluşturur.

**int edgeOfLength()**

Yollanılacak olan Node için uzunluk değerini geri döndürür.

**int down()**

Yollanılacak olan node'u son elementine kadar kadar tarayıp eğer aktif olan node karşılaştırılan node'dan daha kısaysa karşılaştırılan node'u aktif node olarak seçer.

**void extendSuffix()**

Projenin hem anlaması hemde yazması en zorlayıcı olan kısımdı. Ukkonen Suffix Tree'nin tüm kuralları bu fonksiyon içerisinde gerçekleşiyor. En önemli işlevi suffixlenmesi gereken başka bir substring var mı eğer varsa nodelarda onun için dinamik hafıza açmak.

**void printSufTree()**

Suffix Tree'yi printler.

**void setSuffixDFS()**

Suffix Tree'yi Depth-First Search şeklinde printlemek ve nodeları numaralandırma işlemini yapan fonksiyon.

**void freeSuffix()**

Hafızada Suffix Tree için açılan yeri boşaltmak.

**void buildSuffix()**

Tüm fonksiyonların ana test kısmı. Suffix Tree'yi inşa edip yazdırır.

**int countFreq()**

Girilen stringde aratılan substringin kaç defa çıktığını hesaplar.

**void substring()**

En uzun substringi bulmak için işimize yarayan fonksiyon.

## **int lcp()**

Longest Common Prefix Fonksiyonu.

Kullandığım kütüphaneler bir sonra ki bölümde belirtilmiştir.

### VII. KÜTÜPHANELER

Bu kısımda projeye include ettiğim kütüphaneler bulunmakta:

- 1) stdio.h
- 2) string.h
- 3) stdlib.h
- 4) math.h
- 5) graphics.h

### REFERANSLAR

- 1) Ukkonen's Suffix Tree  
"www.geeksforgeeks.org"
- 2) Suffix Tree Problems  
"www.geeksforgeeks.org"
- 3) Visualization of Ukkonen's Algorithm  
"brenden.github.io/ukkonen-animation/"
- 4) Ukkonen's suffix tree algorithm in plain English  
"www.stackoverflow.com"
- 5) Visualize a Tree  
"www.rosettacode.org"
- 6) Suffix Tree  
"www.rosettacode.org"
- 7) Generalized Suffix Tree  
"en.wikipedia.org"