

**KOCAELİ ÜNİVERSİTESİ**  
**BİLGİSAYAR MÜHENDİSLİĞİ**  
**PROGRAMLAMA LABORATUVARI 2 PROJE 1**



**BELİNAY POLATCAN 210202104**

**AHMET CAN OKUMUŞ 200202015**

## **ÖZET**

Programlama Laboratuvarı 2 dersinin 1. Deneyinde bizden txt dosyasında verilen bir kodun dosyadan okunma işleminin yapılması, konsola yazdırılması ve txt dosyasında verilen kodun zaman ve yer karmaşıklığının hesaplanması istenmiştir.

## GİRİŞ

Proje için C programlama dili ve CodeBlocks geliştirme ortamı kullanıldı. AT&T Bell laboratuvarlarında, KenThompson ve DennisRitchie tarafından UNIX İşletim Sistemi'ni geliştirebilmek amacıyla B dilinden türetilmiş yapısal bir programlama dilidir. Code:Blocks özellikle C ve C++ gibi programlama dillerinde kodlama geliştirme yapmamıza imkan veren, açık kaynak kodlu ve crossplatform bir IDE'dir.

## YÖNTEM

Projemiz calculatebigo() ve main() fonksiyonlarından oluşuyor. Txt dosyasından okuduğumuz kodu main fonksiyonunda satır satır okutarak konsola yazdırdık. Dosyanın adını filename adında const olarak tanımlanmış char değişkenine atadık.

```
char const* const fileName = "kodlar.txt";
```

Projemizde txt dosyasındaki kodu satır satır okuduktan sonra strstr() fonksiyonu ile dosyadaki kodda algoritma analizi için gereken 'for, while' ve benzeri terimleri kontrol ettik.

```
if(strstr(line,"for ") || strstr(line,"while "))
```

```
{
```

```
if(strstr(line,"} ")){
```

```
if (sayac > maxsayac){  
  
    maxsayac = sayac;  
  
}
```

Calculatebigo()

Strstr fonksiyonu ile kodda for ve while var mı? Sonucunu kontrol ettik. Eğer for ve while bulunuyorsa sayaç değeri 1 artıyor. Daha sonra strstr fonksiyonu ile ‘}’ ibaresini kontrol ettik. Satır satır tarama işlemi yapıldığında bu ibare bulunuyorsa sayaç değeri 1 azalıyor.

```
if(strstr(line," } ")){  
  
    if (sayac > maxsayac){  
  
        maxsayac = sayac;  
  
    }  
  
    sayac = sayac - 1 ;
```

Maxsayaç değerini ise iç içe for döngülerinde n değerini 1 arttırmak için kullandık. Bu değeri başlangıçta 0 olacak şekilde atadık. Eğer sayac değeri maxsayac değerinden büyükse maxsayac değeri sayaç değerine eşit olacak şekilde if bloğu açtık.

$O(\log n)$  değerlerini bulmak için öncelikle logsayaç adında bir değişken tanımladık. Bu değişkenin başlangıç değeri 0. Daha sonra for döngüsü içine baktık. Eğer i değerlerini çarpım olarak görürse logsayaç değerimiz 1 artıyor. Bu logsayac değerini zaman karmaşıklığını yazdırırken  $O(\log^{\text{logsayac}})$  olacak şekilde düşündük.

```
if(strstr(line,"i=i* ") || strstr(line,"j=j* ")){
```

```
    logsayac=logsayac+1;

}
```

Verilen kodun yer karmaşıklığını bulmak için ise yine strstr fonksiyonuyla koddaki int, float, char, return vb. boyutu 4 byte olan değerlere baktık. Eğer kodda bu ifadeler görülürse yer karma adını verdiğimiz sayaç değeri 4 artıyor.

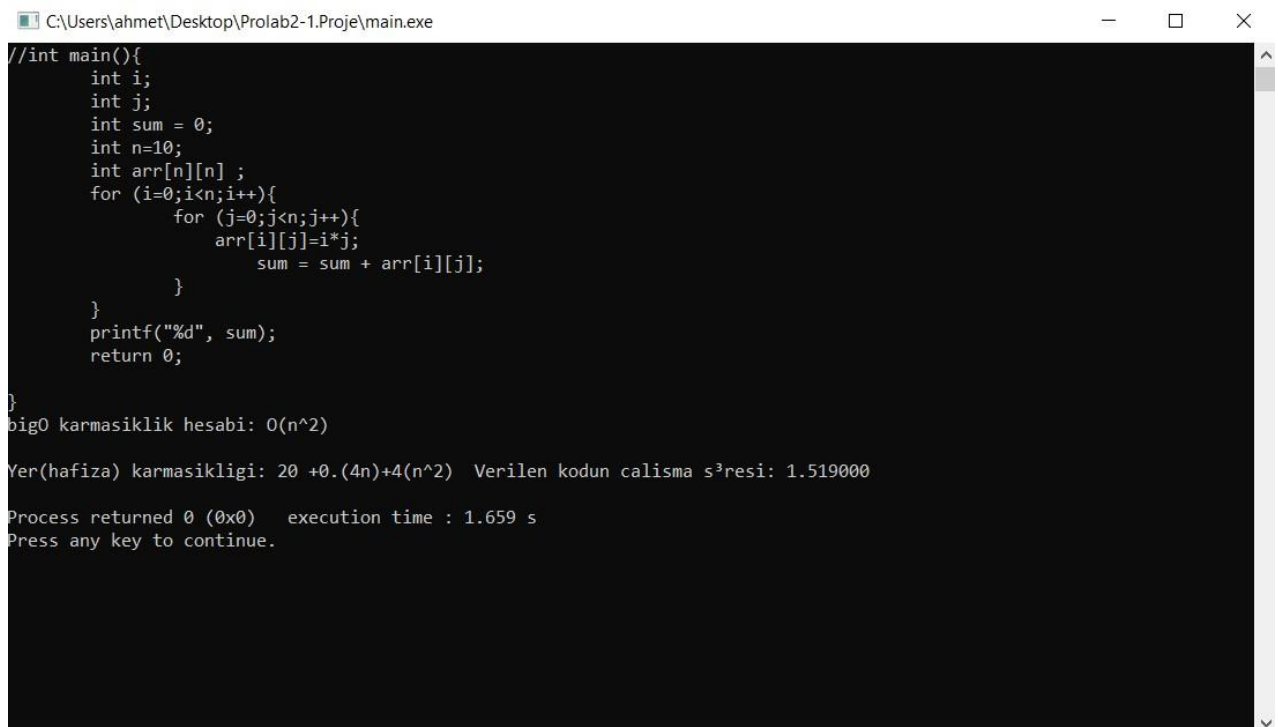
```
    if(strstr(line,"int ") || strstr(line,"return ") || strstr(line,"float ") ||
    strstr(line,"long ")){

        yerkarma=yerkarma+4;

    }
```

Kodda verilen dizilerde ise her dizide yer karmaşıklığı  $4n$  artacak şekilde işlem yaptık.

## DENEYSEL SONUÇLAR



```
C:\Users\ahmet\Desktop\Prolab2-1.Proje\main.exe

//int main(){
    int i;
    int j;
    int sum = 0;
    int n=10;
    int arr[n][n] ;
    for (i=0;i<n;i++){
        for (j=0;j<n;j++){
            arr[i][j]=i*j;
            sum = sum + arr[i][j];
        }
    }
    printf("%d", sum);
    return 0;
}

big0 karmasiklik hesabi: 0(n^2)

Yer(hafiza) karmasikligi: 20 +0.(4n)+4(n^2) Verilen kodun calisma s^resi: 1.519000

Process returned 0 (0x0) execution time : 1.659 s
Press any key to continue.
```

```
C:\Users\ahmet\Desktop\Prolab2-1.Proje\main.exe
#include <stdio.h>
#include <stdlib.h>
main()
{
    int count=10;
    int i;
    int n=10;
    int arr[n] ;
    i=1;
    do{
        arr[i]=i*count;
        printf("%d * %d = %d\n",i,count,i*count);
        i++;
    } while (i<=n);
    return 0;
}
bigO karmasiklik hesabi: O(n^1)
Yer(hafiza) karmasikligi: 16 +1.(4n)+4(n^0) Verilen kodun calisma s^3resi: 1.535000
Process returned 0 (0x0)   execution time : 1.679 s
Press any key to continue.
```

```
Seç C:\Users\ahmet\Desktop\Prolab2-1.Proje\main.exe
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

main()
{
    int i;
    int j;
    int n;
    int k;
    int dizi[][] ;

    for (i=0;i<n;i++){

        k=k+8;
    }

    printf("Lutfen dizi eleman sayisini girin: ");
    scanf("%d",&n);

    printf("\n lutfen dizi elemanlarini girin: ");

    for (i=0;i<n;i++){
        for (j=0;j<n;j++){
            scanf("%d",&dizi[i][j]);
        }
    }

    return 0;
}
bigO karmasiklik hesabi: O(n^2)
Yer(hafiza) karmasikligi: 20 +0.(4n)+4(n^2) Verilen kodun calisma s^3resi: 1.209000
Process returned 0 (0x0)   execution time : 1.325 s
```

```
C:\Users\ahmet\Desktop\Prolab2-1.Proje\main.exe
#include <stdlib.h>
#include <stdio.h>

main()
{
    int i;
    int j;
    int k;

    for (k=0;k<n;k++){
        for (j=0;j<n;j++){
            for (i=0;i<n;i=j* 2){
                printf("Hello world");
            }
        }
    }

    return 0;
}

bigO karmasiklik hesabi:  $O(n^2 \log^1(n))$ 

Yer(hafiza) karmasikligi:  $16 + 0.(4n) + 4(n^0)$  Verilen kodun calisma s3resi: 1.130000

Process returned 0 (0x0)   execution time : 1.235 s
Press any key to continue.
```

```
C:\Users\ahmet\Desktop\Prolab2-1.Proje\main.exe
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

main()
{
    int dizi[3]={1,2,3};

    dizi[0]=dizi[1]+2;

    dizi[2]=dizi[2]*5;

    return 0;
}

bigO karmasiklik hesabi:  $O(1)$ 

bigO karmasiklik hesabi:  $O(n^0)$ 

Yer(hafiza) karmasikligi:  $8 + 0.(4n) + 4(n^0)$  Verilen kodun calisma s3resi: 1.011000

Process returned 0 (0x0)   execution time : 1.113 s
Press any key to continue.
```

**YALANCI KOD**

- Sayaçlar tanımlandı.
- Dosya açıldı.
- 1000 boyutlu line değişkeni atandı.
- While döngüsü dosyanın sonuna kadar okunması için açıldı.
- Line değişkenine atılanlar konsola printf ile yazıldı.
- Eğer dosyada int,float,long değişkenleri varsa yer karma sayacı 4 arttırılacak şekilde if bloğu açıldı.
- Eğer tek boyutlu dizi varsa yer karmaşıklığı  $4n$  artacak şekilde if bloğu açıldı.
- Eğer çift boyutlu dizi varsa yer karmaşıklığı  $4n^2$  artacak şekilde if bloğu açıldı.
- Strstr fonksiyonu ile for döngüsünün değeri  $i=i*$  vb. şekillerde artacak şekildeyse logsayac değişkeni 1 artacak şekilde if bloğu açıldı.
- Calculatebigo fonksiyonu çağırıldı.
- İnt tanımlanan Calculatebigo fonksiyonunda tekrar satır satır okuma işlemi yapıldı.
- For veya while döngüsü görüldüğünde sayac değeri 1 artıyor.
- ‘}’ ifadesi görüldüğünde sayac değeri 1 azalıyor.
- Eğer sayac değeri maxsayac değerinden büyükse maxsayac değeri sayac değerine eşit oluyor.
- Fonksiyon maxsayac değerini döndürüyor.
- Fonksiyonda döndürülen değer main fonksiyonunda bigo değişkenine tanımlanıyor.
- Eğer logsayac 0’den küçük ve eşitse ve bigo değeri 0’a eşitse yer karmaşıklığı  $O(1)$  olacak şekilde ekrana yazdırıldı.
- Eğer logsayac değeri 0’den büyükse bigo değerinden logsayac değeri kadar çıkarma işlemi yapılıyor. Bigo nun yeni değeri logsayac değeri ile zaman karmaşıklığını konsola yazdırmak için ekrana yazdırılıyor.

- Yer karmaşıklığı değeri yerkarma, dizisayac, dizisayac2 değerleri kullanılarak ekrana yazdırılıyor.
- Dosya kapatıldı.

## SONUÇ

Bize verilen projede algoritma karmaşıklıklarını, zaman ve yer hesaplamalarının ne olduğunu ve nasıl hesaplandığını, kod üzerinden nasıl hesaplandığını öğrendik. Dosyadan metin okuma işlemini daha iyi kavradık.

## KAYNAKÇA

- <https://bilgisayarnot.blogspot.com/2020/05/algoritma-zaman-hafza-karmasiklik.html>
- <https://ibrahimkaya66.wordpress.com/2013/12/30/10-algoritma-analizi-algoritmalar-da-karmasiklik-ve-zaman-karmasikligi/comment-page-1/>
- <https://www.javatpoint.com/big-o-notation-in-c>
- <http://cagataykiziltan.net/programin-calisma-hizi-ve-algoritma-verimlilik/zaman-karmasikligi-ve-buyuk-o-notasyonu-time-complexity-and-big-o-notation/>
- <https://www.mygreatlearning.com/blog/why-is-time-complexity-essential/>
- <https://medium.com/dataseries/how-to-calculate-time-complexity-with-big-o-notation-9afe33aa4c46>