PROGRAMLAMA LABORATUVARI 2
PROJE 1

TARIK GÖREN
Bilgisayar Mühendisliği
200202022@kocaeli.edu.tr

ISHAK ERDOĞAN
Bilgisayar Mühendisliği
200202035@kocaeli.edu.tr

ÖZET

Bu doküman Programlama Laboratuvarı 2 dersi 1. Projesi için hazırlanmıştır. Dokümanda projenin tanımı, çözüm yöntemi, kullanılan kütüphaneler gibi programın oluşumunu açıklayan başlıklara yer verilmiştir. Doküman sonunda projemizi hazırlarken yararlandığımız kaynaklar bulunmaktadır.

I. GIRIŞ

Bu projede bizden C dilini kullanarak C dilinde yazılmış kodların zaman karmaşıklığı (time complexity) ve hafıza karmaşıklığını (space complexity) Big O Notasyonu kullanarak hesaplamamız istenmiştir.

Big O Notasyonu bir algoritmanın çalışma zamanının veya bellekte ayrılan yer miktarının üst sınırını temsil eder. Big O Notation'ın rolü, bir algoritmanın yürütülmesi için alabileceği en uzun süreyi veya yeri hesaplamaktır, yani bir algoritmanın en kötü durumunu hesaplamak için kullanılır.

Bu proje kapsamında bizden yapılması istenen isterler ;

1) Dosya içerisinden kodu okunması ve Dosyanın içeriğinin kontrol edilmesi
2) Dosyadan okunan kod Big O notasyonuna göre Zaman karmaşıklığının hesaplanması
3) Dosyadan okunan kod Big O notasyonuna göre yer (Hafıza) karmaşıklığının hesaplanması
4) Dosyadan okunan kod çalıştırıldığında geçen süresin hesaplaması

II. YÖNTEM

A. Algoritma

Öncelikle dosyadan kodlar satır satır okunur. Her satırda çeşitli sorgulamalar yapılır. Zaman karmaşıklığı hesabında satırlar teker teker okunarak anahtar sözcük olarak belirlediğimiz for,while,do-while ve return anahtar sözcükleri aranır.bulunduktan sonra ise şartı sağlanan if'e girecektir. Anahtar sözcük bulunduktan sonra hemen '' işareti bir diziye eklenir. Sonra ise ++,–,*ve/ sembolleri aranmaya başlanır.ona göre hangi zaman karmaşıklığın adenk geldiği belirlenir. Zaman karmaşıklığı hesaplandıktan sonra ise zaman karmaşıklığına denk gelen integer değeri yeni bir diziye atanıyor.örneğin O(1) karmaşıklığı için 1 değeri atanıyor. O(logN) karmaşıklığı için ise 2 değeri atanıyor. Böyle yapmamızın sebebi ise olabildiğince hafızadan "az yer" kaplamaktır.
Yer karmaşıklığı hesabında okunan satırlar satirKontrol ve fonksiyonMu metotları tarafından kontrol edilir ve dönen değerlere göre işlemler yapılır.
Eğer fonksiyon tanımı yapılıyorsa o satırda fonksHesap satırı çalışır ve fonksiyon return type'a göre ve fonksiyon parametre listesinde tanımlanan değişkenlere göre hesaplar yapılır.
Eğer okunan satırda bir değişken tanımı yapılıyorsa hesapla fonksiyonu tarafından o satır gönderilir ve değişken tiplerine göre hesaplar yapılır. hesapla() fonksiyonunda gönderilen satırdaki değişken tipleri ve isimleri parça parça kontrol edilir. Parçalar dizi olması veya olmaması strstr fonskiyonu ile kontrol edilir ve ona göre bir hesaplama yapılır. Yapılan hesaplamalar char dizisi ile döndürülür. En son adımda bir string içerisinde toplanan hesaplamalar genel bir toplamda birleştirilir ve genel bir ifade çıkarılır. Ekrana yazdırılır.
Okunan kodun çalışma süresi hesabında time.h kütüphanesinden yararlanıldı. Çalışma süresinde kodu

bir blok içerisinde tanımlayıp o bloğun başlangıcına ve sonuna clockt değişkenleri oluşturup clock() fonksiyonu ile sistem zamanına eşitlendi. Başlangıç ve son değişkenlerini birbirinden çıkararak kodun gerçek çalışma süresini gecenZaman adlı değişkende kaydı sağlandı ve bulunan değer ekrana yazdırıldı.

B. Geliştirme Ortamı

Proje standart C dilinde geliştirilmiştir.
Windows sitemde geliştirilip IDE olarak Dev C++ kullanılmıştır. Dev C++ uygulamasında 'TDM-GCC 4.9.2 64-Bit Release' modda derlenip çalıştırılmıştır. Projeyi çalıştırmadan önce projenin bulunduğu klasörde 'sunumKod.txt' adında içerisinde C dilinde yazılmış kod parçaları bulunan bir metin dosyası oluşturulması gerekmektedir. Aksi takdirde dosya bulunamayacaktır ve program istenilen şekilde çalışmayacaktır.

C. Kütüphaneler ve Tanımlamalar

Kullandığımız kütüphaneler ve ne için kullandığımız aşağıdaki gibidir:

<stdio.h> : input ve output almak için
<stdlib.h> : malloc gibi fonksiyonları kullanmak için
<string.h> : strcmp,strcpy,strstr gibi string üzerinden işlem yapan fonksiyonları kullanmak için
<ctype.h> : isdigit fonksiyonunu kullanmak için

<time.h> : sistemin çalışma zamanına erişmek için

D. Yalancı Kod

1-menuden seçimi yap
'1' - Big O - Zaman Karmasikligi Hesabi
'2' - Big O - Hafiza Karmasikligi Hesabi
'3' - Calisma suresi"
(Programi sonlandirmak icin '-1')

-eğer '1' ise:
-dosyayı oku
-dosyadan satır satır al
-foru ara bulunursa fonkfor fonksiyonuna git
- bu sembolleri bul:<ve>.
-yanında sayı mı yoksa harf mi var
-eğer sayıysa 1 döndür
-aksi halde sorgulamaya devam et
- çarpım bölme sembolü varsa log(n) döndür
-toplama çıkarma varsa o(n) döndür
-return edilenler bir dizide toplanır.
-küme parantezleri bir dizide toplanır
-while ara bulunursa fonkWhile fonksiyonouna git
- bu sembolleri bul:<ve>.
-yanında sayı mı yoksa harf mi var
-eğer sayıysa 1 döndür
-aksi halde sorgulamaya devam et
- çarpım bölme sembolü varsa log(n) döndür
-toplama çıkarma varsa o(n) döndür
-return edilenler bir dizide toplanır.
-küme parantezleri bir dizide toplanır

-do ara bulunursa fonkWhile fonksiyonouna git
- bu sembolleri bul:<ve>.
-yanında sayı mı yoksa harf mi var
-eğer sayıysa 1 döndür
-aksi halde sorgulamaya devam et
- çarpım bölme sembolü varsa log(n) döndür
-toplama çıkarma varsa o(n) döndür
-return edilenler bir dizide toplanır.
-küme parantezleri bir dizide toplanır
-do whilede while bulunana kadar işlem sembolleri aranacak

-"int","char","float","double","(","main","printf" ara
-returnu sorgula
-sayı varsa zaman karmasikligi:O(1)'dir.diye bas
-harfse zaman karmasikligi:O(n)'dir.

-return bulunana kadar satırlar teker teker alınacaktır.
-eğer bu şartlardan herhangi birisine girmezse kume işaretini arayacak ve bir dizide toplayacak
eğer 2 ise yer karmaşıklığı hesabı
-dosyayı oku
-dosyadan satır satır al
-strdup fonk.çalıştır.eğer
-Cekilen satir fonksiyon ise tipini doner
-Degilse NULL doner.
-fonks tipini sorgula
-strcpy çalıştır
-fonksiyontipini fonksiyonmu fonksiyonuyla hesapla
-satır kontrol fonk.le saorgula
-kontrol satir 1 mi?
-strstr(satir,"," sorgula
- strstr(satir,";" sorgula
eğer 3 ise yer karmaşıklığı hesabı
-clock();Sistemin calismaya basladiktan itibaren gecen zamani getirir.
-ve son kadar geçensüre hesaplanır
eğer -1 ise yer karmaşıklığı hesabı program sonlanacak

III. Sonuçlar

Bizden istenen tüm isterlere cevap verilmeye çalışılmıştır.
Proje sonunda Big O Notasyonu hakkında daha detaylı bilgiler edindik. Yazdığımız kodların çalışma süresi ve bellek kontrolü tarafında göstermemiz gereken aksiyonları öğrendik.
Gerekli testler yapılarak programın runtime hataları vermesi önlendi.

Program Çıktıları

```
Ana Menu
'1' - Big O - Zaman Karmasikligi Hesabi
'2' - Big O - Hafiza Karmasikligi Hesabi
'3' - Calisma suresi
(Programi sonlandirmak icin '-1')
```

Şekil 1. Ana Menü

```
int main(){

int i,j;
int sum = 0;
int n=10;
int arrray[n][n][n][n];
int arr[n][n];

          for (i=0;i<n;i*=2){
                for (j=0;j<n;j++){
                    for (j=0;j<n;j++){
                          arr[i][j]=i*j;
                          sum = sum + arr[i][j];
                }
        }

        printf("%d", sum);
        return 0;

}
```

Şekil 2. Seçim 1

```
O(logN) x O(N) x O(N)
```

Şekil 3. Seçim 2

```
2

+ Yer karmasikligi: O(N^4) =   4*(I)^4 + 4*(I)^2 + 20
```

Şekil 4. Seçim 3

Şekil 5. Seçim 4

KAYNAKLAR

[1] Big O Notation
https://careerkarma.com/blog/
big-o-notation-time/#:~:text=The%
20Big%20O%20Notation%20for,
two%20to%20get%20our%20runtime.
https://medium.com/@zoebai_70369/
big-o-notation-time-and-space-complexity-305a1e301e35

[2] Veri Tipleri;
http://www.ibrahimbayraktar.net/2013/12/
c-programlama-veri-tipleri.html

[3] Stringler
https://stackoverflow.com/questions/26816548/
extract-a-specific-text-pattern-from-a-string
https://turkmuhendis.net/cprogramlama/
string-kutuphanesi/

[4] Bazı unutulmus kavramlar için;
Onur Gök ve Suhap Sahin Programlama Dersi
Slaytları

[5] Proje pdf'inde verilen kaynaklar;
https://bilgisayarnot.blogspot.com/2020/05/
algoritma-zaman-hafza-karmasiklik.html
https://www.javatpoint.com/big-o-notation-in-c