PROGRAMLAMA LABORATUVARI 2
1. PROJE
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi

Sinem ERCÜMERT 200202016@kocaeli.edu.tr
İlkay Mehmet BORA 200202030@kocaeli.edu.tr

I. Özet

Bu doküman Programlama Laboratuvarı 2 dersi 1. projesi için çözümümüzü açıklamaya yönelik oluşturulmuştur. Dokümanda projenin özet, giriş, yöntem, deneysel sonuçlar ve sonuç kısımlarına yer verilmiştir. Doküman sonunda projemizi hazırlarken kullandığımız kaynaklar bulunmaktadır.

II. Giriş

Bu projenin amacı, C dilinde kullanılan kodların zaman ve yer karmaşıklığının hesaplamasıdır. Zaman karmaşıklığı, problemi çözmek için algoritmanın harcadığı zamanın analizini; yer(space) karmaşıklığı, gerekli belleğin analizinin hesabını gerektirir. Bir algoritmanın çalışma süresi çok fazla etkene bağlıdır. Kullandığı bilgisayarın özelliklerinden kullandığı işletim sistemine arkada çalışan programlara gibi sayılabilecek bir çok faktör buna etki edebilir. Karmaşıklığı hesaplarken bu faktörler göz önüne alınmadan sadece kod üzerinde analiz yapılarak sonuca ulaşılır. Burada asimptotik notasyon (Asymptotic notation) adı verilen bir standart notasyon uygulamaktayız. Bu standart ile bu bağımlılıklardan kurtularak kod üzerinde bir analiz yapabilmekteyiz. Öncelikle bu notasyondaki birkaç terimden bahsetmeliyiz.
-Big O Notasyonu – en kötü durum
-Big Omega Notasyonu - en iyi durum
-Big Teta Notasyonu – ortalama durum
Bu proje de en kötü durum notasyonu olan Big O Notasyonunu hesapladık. Algoritma analizinde derleyici, donanım, bellek hızı gibi bağımlılıklardan dolayı belli bir komutun çalışma süresi farklılık gösterebilir. Amaç bu etkenlerden bağımsız bir şekilde, algoritmanın verimliliğini ölçmektir. Big O gösterimi ile algoritmadaki sabitler ve diğer gereksiz etmenler göz ardı edilir ve analizi daha da basitleştirir.

III. Yöntem

A. Geliştirme Ortamı

Projenin gelişimini takip edebilmek için Windows sistemde CodeBlocks IDE'si ve macOS sistemde Bootcamp'te CodeBlocks IDE'si kullanılmıştır. Proje C programlama dili ile yazılmıştır.

B. Yazılım Tasarımı

Projenin yazılım aşaması bu başlık altında bulunan fonksiyonlar tarafınca geliştirilmiştir:

struct yigin –> Kuyruk yapısının oluştuğu struct yapısıdır.
struct order-struct –> Coder fonksiyonu için kuyruk yapısıdır.
void sonaEkle –> Yığın struct yapısını oluşturma fonksiyonudur.
void ord-sonaEkle –> Order struct yapısını oluşturma fonksiyonudur.
void listele –> txt dosyasını boşluksuz bi şekilde ekrana yazdırma fonksiyonudur.
void order-Listele –> Coder'ı ekrana yazdırmak için kullanılan yapıdır.
void coder –> Big-o'su alınacak yapıların kodlanması oluşturulan yapıdır.
int char-to-int –> Char olan sayıyı int değere döndürür.
int digit –> Teker teker yollanan sayıları basamaklı

sayıya çevirir.
void memory-finder –> Dosyadan okunan kodun Big O notasyonuna göre yer (Hafıza) karmaşıklığını bulan fonksiyondur.
void variable-type –> Veri tiplerini return olarak döndürür.
void variable-fun –> variable-type'tan aldığı return değerleriyle veri tiplerinin hafıza karmaşıklığını hesaplar.
int type-bigO –> fonksiyonların bigO'sunu döndürür.
void operation-sequence –> Döngülerin iç içe olup olmadığını kontrol ederek ona göre Big-o hesaplamasında aralarına işlem işareti koyar.
void operation-sequence-printer –> operation-sequence'den alınanları ekrana yazdırır.
void bigO-analyzer –> Big-o değerlerinin tutulduğu fonksiyondur. Hem aynı big-o değerlerini kendi içinde hem de hepsini farklı bir dizide tutar.
void bigO-printer –> bigO-analyzer'dan gelen bilgileri 4 farklı big-o durumu şeklinde ekrana yazdırır.
void bigO-all-printer –> bigO-analyzer'dan gelen bütün durumların olduğu diziyi ekrana yazdırır.
void bigO-all-analyzer-2-n –> bigO-all-mult'dan gelen bilgilerin üst değerlerine göre(büyüklük hesabı için) tutulduğu yerdir.
void bigO-all-2-n-printer –> Big-o'su hesaplanırken çarpılan n'li değerlerin üstlerini yazdırır.
void bigO –> bigO-all-analyzer-2-n fonksiyonundan gelen bilgilere göre en büyük değeri bulur ve ekrana yazdırır.
void bigO-all-mult –> Bütün big-o değerlerinin tutulduğu dizide n'li ve sabit sayıları aralarında çarpma işareti varsa çarpma fonksiyonudur.
void fun-name-array –> Fonksiyon isimlerini çektikten sonra normal fonksiyon mu recursive fonksiyon mu diye karşılaştıran fonksiyondur.
void que-to-array –> name-array fonksiyonundan gelen fonksiyon isimlerini ekrana yazdırır.
void recursive-fun –> Recursive fonksiyonların Big-o'sunu bulur.
void while-fun –> While döngülerinin Big-o'sunu bulur.
void for-array –> For döngülerinin üçüncü partınıchar olarak dizide tutar.
void for-printer –> For döngülerinin üçüncü partını ekrana yazdırır.
void for-fun –> For döngülerinin Big-o'sunu bulur.
int thirdPart –> For döngülerinin üçüncü partındaki değerlerin char hallerini bularak returnle döndürür.
void identifier –> Coder fonksiyonundan gelern bilgiyle döngülerin iç içe olup olmadığını bulur.
void space-comp-printer –> Memory'nin hesaplarını ekrana yazdırır.
void zaman –> Dosyadaki kodun cmd çalışma süresini bulur.
int main –> Fonksiyonlar burada yazılıdı.

C. Yalancı Kod

struct veri yapıları yardımıyla kullanacagımız degiskenler integer degerde tanımlanır.

void coder(int size)

1-Kullanılacak struct yapısı ve değişkenler atanır.
2-if yapısında bağlı liste kullanarak 'main' harfleri bulunur, ord-sonaEkle fonksiyonuna 1 olarak gönderilir.
3-if yapısında bağlı liste kullanarak 'for' harfleri bulunur, ord-sonaEkle fonksiyonuna 2 olarak gönderilir.
4-if yapısında bağlı liste kullanarak 'while' harfleri bulunur, ord-sonaEkle fonksiyonuna 3 olarak gönderilir.
5-if yapısında bağlı liste kullanarak 'switch' harfleri bulunur, ord-sonaEkle fonksiyonuna 4 olarak gönderilir.
6-if yapısında bağlı liste kullanarak 'else if' harfleri bulunur, ord-sonaEkle fonksiyonuna 5 olarak gönderilir.
7-if yapısında bağlı liste kullanarak 'if' harfleri bulunur, ord-sonaEkle fonksiyonuna 6 olarak gönderilir.
8-if yapısında bağlı liste kullanarak 'else' harfleri bulunur, ord-sonaEkle fonksiyonuna 7 olarak gönderilir.
9-if yapısında bağlı liste kullanarak 'case' harfleri bulunur, ord-sonaEkle fonksiyonuna 8 olarak gönderilir.
10-if yapısında bağlı liste kullanarak 'do' harfleri bulunur, ord-sonaEkle fonksiyonuna 9 olarak gönderilir.
11-if yapısında bağlı liste kullanarak '' bulunur, ord$_s onaEkle fonksiyonuna 10 olarak gnderilir.$
$12-if yapßsßndebalßlistekullanarak''bulunur, ord-$

$sonaEklefonksiyonuna11olarakgnderilir.$

void decoder()

1-Kullanılacak struct yapısı tanımlanır.
2-Arananın order-data'sı 1 ise 'main' yazdırılır.
3-Arananın order-data'sı 2 ise 'for' yazdırılır.
4-Arananın order-data'sı 3 ise 'while' yazdırılır.
5-Arananın order-data'sı 4 ise 'switch' yazdırılır.
6-Arananın order-data'sı 5 ise 'if' yazdırılır.
7-Arananın order-data'sı 6 ise 'else' yazdırılır.
8-Arananın order-data'sı 7 ise 'else if' yazdırılır.
9-Arananın order-data'sı 8 ise 'do' yazdırılır.
10-Arananın order-data'sı 9 ise '' yazdırılır.
11-Arananın order-data'sı 10 ise '' yazdırılır.
12-Arananın order-data'sı 11 ise 'case' yazdırılır.

void memory-finder()

1-Kullanılacak değişkenler tanımlanır.
2-Dosyanın içinde dolaşarak içi boş iki boyutlu matris bulunur, gerekli memory değerleri atanır.
3-Dosyanın içinde dolaşarak içi boş bir boyutlu dizi bulunur, gerekli memory değerleri atanır.
4-Dosyanın içinde dolaşarak bir boyutlu dizi bulunur, gerekli değerleri memory atanır.
5-Dosyanın içinde dolaşarak içi iki boyutlu dizi bulunur, gerekli memory değerleri atanır.
6-Dosyanın içinde dolaşarak içinde sayı olan iki boyutlu dizi bulunur, gerekli memory değerleri atanır.
7-Dosyanın içinde dolaşarak içinde char olan iki boyutlu dizi bulunur, gerekli memory değerleri atanır.
8-Dosyanın içinde dolaşarak içi boş iki boyutlu matris bulunur, gerekli memory değerleri atanır.
9-Dosyanın içinde dolaşarak değişkenler bulunur, gerekli memory değerleri atanır.
10-Dosyanın içinde dolaşarak fonksiyonlar/printfler bulunur, gerekli memory değerleri atanır.

void variable-type(struct yigin *liste)

1-Dosyada 'char' bulduysa 1 döndürür.
2-Dosyada 'int' bulduysa 2 döndürür.
3-Dosyada 'float' bulduysa 3 döndürür.
4-Dosyada 'long' bulduysa 4 döndürür.
5-Dosyada 'short' bulduysa 5 döndürür.
6-Dosyada 'double' bulduysa 6 döndürür.
7-Dosyada 'return' bulduysa 7 döndürür.

void variable-fun()

1-Kullanılacak struct yapıları ve değişkenler tanımlanır.
2-variable-type fonksiyonundan 1 değeri geldiyse yerini ve boyutunu memory-finder'a gönderir.
3-variable-type fonksiyonundan 2 değeri geldiyse yerini ve boyutunu memory-finder'a gönderir.
4-variable-type fonksiyonundan 3 değeri geldiyse yerini ve boyutunu memory-finder'a gönderir.
5-variable-type fonksiyonundan 4 değeri geldiyse yerini ve boyutunu memory-finder'a gönderir.
6-variable-type fonksiyonundan 5 değeri geldiyse yerini ve boyutunu memory-finder'a gönderir.
7-variable-type fonksiyonundan 6 değeri geldiyse yerini ve boyutunu memory-finder'a gönderir.
8-variable-type fonksiyonundan 7 değeri geldiyse yerini ve boyutunu memory-finder'a gönderir.

int type-bigO()

1-Kullanılacak struct yapıları tanımlanır.
2-Dosyayı dolaşırken 'for' görürse hesaplamalar için for-fun'a gönderir.
3-Dosyayı dolaşırken 'while' görürse hesaplamalar için while-fun'a gönderir.
4-Dosyayı dolaşırken 'rint' görürse 'printf' ten geldiğini algılar ve geçer.
5-Dosyayı dolaşırken 'int' görürse hesaplamalar için recursive-fun'a gönderir.

void bigO-analyzer(int number,int magnitude)

1-case 1 durumunda Big-O'su 1 olan değerleri tutar.
2-case 2 durumunda Big-O'su n olan değerleri tutar.
3-case 3 durumunda Big-O'su logn olan değerleri tutar.

void for-fun(char ch)

1-Kullanılacak değişkenler tanımlanır.
2-Dosyada for bulunduktan sonra ’)’ işaretine kadar ilerletilir.
3-for’un ilk partı bulunacak şekilde dosyada ilerlenir ve char mı int mi tespit edilir.
4-for’un ikinci partı bulunacak şekilde dosyada ilerlenir ve char mı int mi tespit edilir.
5-for’un üçüncü partı bulunacak şekilde dosyada ilerlenir ve thirdPart fonksiyonundan gelen bilgilere göre tipi belirlenir.
6-for’un üç partının tipi de tespit edilerek thirdPart fonksiyonuna gönderilir.

int thirdPart(int number)

1-For’un üçüncü kısmı ’++’ ise bigO-analyzer’a gerekli değerler gönderilir.
2-For’un üçüncü kısmı ’–’ ise bigO-analyzer’a gerekli değerler gönderilir.
3-For’un üçüncü kısmı ’+=’ ise bigO-analyzer’a gerekli değerler gönderilir.
4-For’un üçüncü kısmı ’-=’ ise bigO-analyzer’a gerekli değerler gönderilir.
5-For’un üçüncü kısmı ’*=’ ise bigO-analyzer’a gerekli değerler gönderilir.
6-For’un üçüncü kısmı ’/=’ ise bigO-analyzer’a gerekli değerler gönderilir.
7-For’un üçüncü kısmında ’=’den sonra ’+’ geliyorsa bigO-analyzer’a gerekli değerler gönderilir.
8-For’un üçüncü kısmında ’=’den sonra ’-’ geliyorsa bigO-analyzer’a gerekli değerler gönderilir.
9-For’un üçüncü kısmında ’=’den sonra ’*’ geliyorsa bigO-analyzer’a gerekli değerler gönderilir.
10-For’un üçüncü kısmında ’=’den sonra ’/’ geliyorsa bigO-analyzer’a gerekli değerler gönderilir.

IV. Deneysel Sonuçlar

Yazdığımız tüm işlemler sonucunda klasörde bulunan .txt uzantılı dosyadan alınan bilgiyle:
1- Yazdığımız coder ve decoder fonksiyonları yardımıyla dosyada kod olup olmadığını kontrol edilmesi,
2- Dosyadan okunan kodun Big O notasyonuna göre Zaman karmaşıklığının hesaplanması,
3- Dosyadan okunan kodun Big O notasyonuna göre yer (Hafıza) karmaşıklığının hesaplanması,
4- Dosyadan okunan kod çalıştırıldığında geçen süresin hesaplaması isterlerini yerine getiriyoruz.

V. Sonuç

Yaptığımız projenin bize kazandırdıkları;
1- Zaman karmaşıklığının ne olduğunu ve nasıl hesaplandığını,
2- Yer(hafıza) karmaşıklığının ne olduğunu ve nasıl hesaplandığnı,
3- Dosyada bulunan kodu kopyala yapıştır yapmadan çalışma süresinin nasıl hesaplanacağını,
4- Bir problemle karşılaştığımızda çözüm için nasıl yollar arayacağımızı,
5- Algoritma mantığını kurma ve uygulamada kendimizi geliştirdik ve öğrendik.

VI. Kaynakça

1- Araştırma için:

-https://bilgisayarnot.blogspot.com/2020/05/algoritma-zaman-hafza-karmasiklik.html
-https://ibrahimkaya66.wordpress.com/2013/12/30/10-algoritma-analizi-algoritmalarda-karmasiklik-ve-zaman-karmasikligi/comment-page-1/
-https://www.javatpoint.com/big-o-notation-in-c
-https://birhankarahasan.com/algoritma-analizi-nedir-zaman-karmasikligi-big-o-gosterimi
-https://www.enjoyalgorithms.com/blog/time-complexity-analysis-of-loop-in-programming
-https://www.enjoyalgorithms.com/blog/time-complexity-analysis-of-recursion-in-programming
-https://developerinsider.co/big-o-notation-explained-with-examples/
-https://www.techiedelight.com/find-execution-time-c-program/
3- Karşılaştığımız çeşitli sorunlar için:

-https://www.geeksforgeeks.org/
-https://stackoverflow.com /
-https://www.quora.com/