# Gebze Technical University
# Computer Engineering

# CSE 321
# INTRODUCTION TO ALGORITHM DESIGN

# HOMEWORK 2

**KADİR BULUT**
**121044005**

**Course Assistant: Ahmet Soyyiğit**

**QUESTION 1**

I would like to say first , I have not watched this movie so far is a very big misfortune for me. Adam Turing who is the main character of film is a great example for us to go after our dreams.Especially his perseverance , his self-belief , make a lot of effort to achieve this success are amazing , puzzling and extraordinary. The film's quality is very nice , too. It is extremely realistic. Animations , characters were so beautiful . Turing is trying to solve the enigma's secret with 40's life conditions and revolting from his colleagues in charge like himself . He trust himself very much and he believes it will succeed . Besides the his intelligence , he is putting forth his perseverance's power against everyone.Let's imagine a man . He finds something with his intelligence and impossibilities and this is affecting all humanity even the whole world. I am sure that , a few engineers can not do that nowadays . Really it pushes the boundaries of mind and brain...At first the commander does not accept him for mission since the commander believe that enigma is impossible and thinks it can not be solved like russians , germans , frenchs and americans . Turing says: " You need to me more than I need to you . Enigma is the most difficult problem in the world but is not impossible. Allow me to try it to know for sure. ". Although he does not know German...This conversation is really my favorite part of the film . Because of his perseverance , his commander defines him as a soviet spy. Because there are many experts outside Turing. One of them won the world chess tournament twice and no one believes his claims . Ali trusts himself so much and believes other experts will slow him down . Turing is actually proof of an impossibility , proof of turning the impossibilities into success. But at the end his success is becoming salvation. Turing produces Christopher and proves himself against everyone. During his talk , Turing says: "Why people love violence ? Because it feels good. Sometimes we have to do what makes sense , it does not feel good." . This was one of the interesting parts in the film. Of course there were some missing things I found in this film. He produces a huge machine but he does not know exactly how it works . On the other hand connection between childhood and main character of film did not seem so convincing me . This man who affects the world and computer science should be introduced with his success and determination , not by his sexual preference . Also , it is a bit strange that a person who contributes so much to computer science is not recognized too much. Overall it was a good movie and I will watch it again.

**QUESTION 2**

*** $x_1(n) = 0.5\ x_1(n/2) + 1/n$**

a=0.5 →katsayı 1'den küçük

a katsayısı 1'den küçük olduğu için master teoremi ile ~~çözülemez.~~

***$x_2(n) = 3x_2(n/4) + nlogn$**

a=3 , b=4 , f(n)=nlogn

$\log_b{}^a = \log_4{}^3 \approx 0.79248 < 1$

3.durum yani $f(n)=\Omega(n^{\log_b{}^{a+\epsilon}})$ $\epsilon>0$ a.f(n/b) <= c.f(n) c<1

$n^1=n^{0.79248+\epsilon}$ → $\epsilon > 0$

a.f(n/b)=3.f(n/4)=3.(n/4).log(n/4)

3.(n/4).log(n/4) <= (3/4) .n.logn buradan c=3/4 < 1 dir.

Dolayısıyla 3.kural gereği $x_2(n)=\Theta(n.logn)$ dir.

***$x_3(n) = 3x_3(n/3) + (n/2)$**

a=3 , b=3 , c=1

$\log_b a = \log_3 3 = 1 = c$ ( 2.durum geçerlidir)

$x_3(n)=\Theta(n^{\log_b{}^a})=\Theta(n^c logn)$

$x_3(n)= \Theta(nlogn)$

***$x_4(n)=6x_4(n/3) + n^2logn$**

a=6 , b=3 , $f(n)=n^2logn$

$\log_b a = \log_3 6 \approx 1.6309$

$n^2 = n^{1.6309+\epsilon}$ → $\epsilon > 0$

Yine 3.durum geçerlidir;

$a.f(n/b) = 6.f(n/3) <= 6.(n/3)^2.\log(n/3) <= (2/3)n^2logn$

c=2/3 < 1

3.kural gereği $x_4(n)= \Theta(n^2logn)$

*** $x_5(n)=4x_5(n/2) + (n/\log n)$**

a=4 , b=2, f(n)=n/logn

$\log_b{}^a = \log_2{}^4 = 2$

$f(n) = n/\log n = O ( n^{\log_b{}^a - \epsilon}) = O(n^{2-\epsilon})$

$n^1 \approx n^{\log_b{}^a - \epsilon} = n^{2-\epsilon} \rightarrow \epsilon > 0$

1.durum söz konusudur.Dolayısıyla $x_5(n)=\Theta(n^{\log_b{}^a})$ $x_5(n)=\Theta(n^2)$

*** $x_6(n)= 2^n x_6(n/2) + n^n$**

$a=2^n \rightarrow$ sabit sayı değil

a katsayısının 1 den büyük sabit bir değer olması gerekir . $2^N$ sabit bir değer olmadığından master teoremi kullanılarak ~~çözülemez.~~

## QUESTION 3

```
def chocolateAlgorithm(n):
    #Input is a positive integer n
    if n==1:
        return 1
    else:
        return chocolateAlgorithm(n-1) + 2 * n -1
```

**a)**

Yukarıda bize verilen fonksiyonu T(n) ile ifade edecek olursak ;
mavi renkli kısımdan → T(1)=1
kırmızı renkli kısımdan → T(n) = T(n-1) + 2n – 1 bağıntıları elde edilir.
Fonksiyonun birkaç terimini hesaplayacak olursak;
T(1) = 1
T(2) = T(1) + 2*2 – 1 = 4
T(3) = T(2) + 2*3 – 1 = 9
T(4) = T(3) + 2*4 – 1 = 16
T(5) = T(4) + 2*5 – 1 = 25
T(6) = T(5) + 2*6 – 1 = 36
.
.
.
Yukarıda görüldüğü üzere fonksiyon $n^2$ değerini hesaplıyor. $T(n) = n^2$

**b)**

Genel formülü T(n) = T(n-1) + 2*n -1 şeklinde elde etmiştik ve formülde 1 adet multiplication bulunuyor. Multiplication sayısı için yeni bir bağıntı yazılacak olursa ;

X(n) = X(n-1) + 1
= X(n-2) + 1 + 1
= X(n-3) + 1 + 1 + 1
= X(n-4) + 1 + 1 + 1 + 1
= X(n-5) + 1 + 1 + 1 + 1 + 1
.
.
.
= X(1) + 1 + 1+ 1 + 1 + 1 + .... ( n değeri için (n-1) adet 1 vardır)

n = 1 için fonksiyon return değeri döndürdüğünden herhangi bir multiplication işlemi yapılmaz ve X(1) = 0 dır.

X(n) = X(1) + (n-1)*1 = 0 + n-1 = n-1

X(n) = n-1

**c)** Genel formülü T(n) = T(n-1) + 2*n -1 şeklinde elde etmiştik ve formülde 3 adet (2n=işlemi için +n + n ve -1 işlemleri ) add/sub işlemi bulunuyor. Add/sub sayıları için yeni bir bağıntı yazılacak olursa ;

X(n) = X(n-1) + 3
= X(n-2) + 3 + 3
= X(n-3) + 3 + 3 + 3
= X(n-4) + 3 + 3 + 3 + 3
= X(n-5) + 3 + 3 + 3 + 3 + 3
.
.
.
= X(1) + 3 + 3+ 3 + 3 + 3 + .... ( n değeri için (n-1) adet 3 vardır)

n = 1 için fonksiyon return değeri döndürdüğünden herhangi bir add/sub işlemi yapılmaz ve X(1) = 0 dır.

X(n) = X(1) + (n-1)*3 = 0 + 3n - 3 = 3n-3

X(n) = 3n-3 = 3(n-1)

## QUESTION 5

**a)** Sorunun kod kısmı ektedir.
**b)**

**Algoritmam**
Algoritmamda tek recursive fonksiyon kullanmayı tercih ettim.

**i.** İlk önce verilen listede sağ ve sol kısmın ağırlıklarının eşit olup olmadığını kontrol ettim bize verilen fonksiyonla.Eğer öyleyse -1 return ettim. ( bu durum verilen listenin uzunluğunun çift bir sayı olması ve rotten walnut olmaması durumu.)

**ii.** Daha sonra durdurma kriteri olarak son eleman olup olmadığını kontrol ettim.Yani aradığım listede tek elemanının sona kalması durumu.

**iii.** Verilen listenin uzunluğunun tek sayıda olması işi zorlaştırıyordu. Tüm sayılar eşit olsa dahi bir kısmın 1 eleman fazla olmasından dolayı terazi fonksiyonu yanıltıp yanlış kısma yönelmeme sebep oluyordu. Bunun için tek sayılı durumlarda ilk elemanı çürük eleman mı diye kontrol ettim. Eğer oysa sonuç return edip değilse listenin diğer kısmıyla devam ettim.Tabi index sayısını 1 artırarak. Bunun için bir flag tuttum.

**iiii.** Yaptığım bu eleman çıkarma işleminden sonra çürük eleman bulamadıysam tekrar terazi fonksiyonuyla geri kalan kısmı kontrol ederek devam etmeye çalıştım.

**iiiii.** Terazi fonksiyonu yardımıyla her defasında listenin sağ yada sol kısmını alarak ; sağ taraf daha hafifse sağa , sol taraf daha hafifse sola yönelmeye çalıştım recursive olarak . Sola yönelmelerimde index kısmında değişiklik yapmadım. Ancak sağa yönelmelerde belli ölçülerde index sayısını artırarak return edeceğim index sayısını artırıyorum .

**iiiiii.** Eğer listeden bir elaman çıkardıysam (flag=1 olma durumuna göre ) sağa ve sola gitme kısımlarını kontrol edip ek olarak indexi 1 artırdım.

```
flag=0  → O(1)
a= compareScales(inputList[:int(len(inputList)/2)],inputList[int(len(inputList)/2):])  → O(1)
if a==0: → O(1)
          return -1
if len(inputList) == 1:  → O(1)
       return 0

if len(inputList) % 2 == 1:
       if inputList[0] < inputList[1]: → O(1)
          return 0
       else:
```

```
        del inputList[0]  → O(1)
        if compareScales(inputList[:int(len(inputList)/2)],inputList[int(len(inputList)/2):]) == 0:  → O(1)
            return -1
        else:
            flag=1 → O(1)
            a= compareScales(inputList[:int(len(inputList)/2)],inputList[int(len(inputList)/2):]) → O(1)

if a==1:
    if flag == 1:  → O(1)
        return findRottenWalnut(inputList[:int(len(inputList)/2)]) + 1 → O(logn)
    else:
        return findRottenWalnut(inputList[:int(len(inputList)/2)])  → O(logn)
  if a==-1: → O(1)
    if flag == 1: → O(1)
        return findRottenWalnut(inputList[int(len(inputList)/2):]) + int(len(inputList)/2) + 1  → O(logn)
    else:
        return findRottenWalnut(inputList[int(len(inputList)/2):]) + int(len(inputList)/2) → O(logn)

if a==-1:
    if flag == 1: → O(1)
        return findRottenWalnut(inputList[int(len(inputList)/2):]) + int(len(inputList)/2) + 1 → O(logn)
    else:
        return findRottenWalnut(inputList[int(len(inputList)/2):]) + int(len(inputList)/2) → O(logn)
```

## Best Case and Worst Case Durumlarında Complexity

**Best Case**: Fonksiyonumun best case olması durumu listenin çift elemanlı olduğu ve listede herhangi bir rotten walnut olmaması durumudur.Bunu zaten fonksiyonumun başında şu kodları çalıştırarak buluyorum:

```
flag=0 → → → O(1)
a= compareScales(inputList[:int(len(inputList)/2)],inputList[int(len(inputList)/2):]) → → → O(1)
if a==0: → → → (O1)
    return -1
```

Dolayısıyla best case durumunda time complexity O(1) dir.

**Worst Case:** Bu durum için listede bir rotten walnut bulunması gerekiyor. Çift sayıdaki uzunlukta bir listenin en sağında ya da en solunda olması durumlarında ta ki listemde tek eleman yani rotten walnut kalana kadar listeyi hep ikiye bölüp listenin sağına yada soluna yönelerek yeni listeyle ilerliyorum. Tek sayıdaki uzunluktaki bir listenin en sağında rotten walnut bulunmasıda worst case. Fakat dediğim gibi hepsinde ikiye bölerek listeleri ilerliyorum.Tek sayıdaki ilk sayıyı if statement ile ve verilen scale fonksiyonuyla kontrol etmem O(1) karmaşıklığına sahip olduğundan karmaşıklığımı etkilemiyor.Etkileyen kısım recursive olarak çevirdiğim kısım:

T(n) = T(n/2) + n/2 +1  ( T(n/2) de listeyi ikiye ayırıyorum. Buradaki listeyi ikiye ayırmayı ve 1 eklemeyi O(1) kabul ediyoruz.)

T(n) = O(1) + (n/2) + O(1)
T(n/2) = O(1) + (n/4) + O(1)
T(n/4) = O(1) + (n/8) + O(1)
T(n/8) = O(1) + (n/16) + O(1)
.
.
.
T(2) = O(1) + T(1) + O(1) ( eleman sayısı en az 2 olacak)

T(n) = logn * 2O(1) – 2*O(1)
T(n)=O(logn)

Dolayısıyla worst case durumunda time complexity O(logn) dir.

**QUESTION 6**

**a)**

*** $T_1(n)=3T_1(n-1)$ for n>1, $T_1(1)=4$**

$T_1(1)=4$
$T_1(2)=3T_1(1)=3*4$
$T_1(3)=3T_1(2)=3*3*4$
$T_1(4)=3T_1(3)=3*3*3*4$
.
.
.
$T_1(n)=3T_1(n-1)=3^{(n-1)}*4 \rightarrow O(3^n)$

*** $T_2(n)= T_2(n-1)+n$ for n>1, $T_2(0)=0$**

$T_2(1)= T_2(0) + 1=1$
$T_2(2)= T_2(1) + 2=1+2=3$
$T_2(3)= T_2(2) + 3=3+3$
$T_2(4)= T_2(3) + 4=3+3+4$
$T_2(5)= T_2(4) + 5=3+3+4+5$
$T_2(6)= T_2(5) + 6=3+3+4+5+6$
.
.
.
$T_2(n)= T_2(n-1) + n=3+3+4+5+...+(n-1)+n$
$=(1+2)+3+4+5+...+n$
$=1+2+3+4+5+...+n$
$=n(n-1) / 2 = (n^2-n)/2 \rightarrow O(n^2)$

*** $T_3(n)=T_3(n/2) + n$  for n>1**

$n=2^k$

$T_3(2^k)=T_3(2^{k-1}) + 2^k$
$T_3(2^{k-1})=T_3(2^{k-2}) + 2^{k-1}$
$T_3(2^{k-2})=T_3(2^{k-3}) + 2^{k-2}$
$T_3(2^{k-3})=T_3(2^{k-4}) + 2^{k-3}$
$T_3(2^{k-4})=T_3(2^{k-5}) + 2^{k-4}$
.
.
.
.
$T_3(2)=T_3(1)+2=2$
$T_3(1)=0$

$T_3(2^k) = 2^k + 2^{k-1} + 2^{k-2} + 2^{k-3} + 2^{k-4} + 2^{k-5} + ... + 2$ ( geometrik seri oluşur)

► Geometrik seri toplam formulünden $a_1(1-r^n)/(1-r)$

$2^k(1+1/2+1/4+1/8+...)$
$=2^k(1-(1/2)^{k+1})/(1-1/2)$
$=2^k(2-(1/2)^k$
$=2.2^k - 1^k.2^k.2^{-k}$
$=2.2^k-1$
$2^k$ yerine n yazılırsa 2n-1 sonucu elde edilir.
Toplam formülünde yerine 0. kuvvet yani 1 eksik olduğu için 1 eklemiştim.Toplamdan çıkarılırsa sonuç 2n-2 elde edilir.

$T_3(n)=2n-2 \rightarrow O(n)$

**b)**

*** $T_1(n) = 6T_1(n-1) – 9T_1(n-2)$  $T_1(0)=1$ , $T_1(1)=6$**

lineer homojen bir denklem olduğundan;

$x^2 = 6x-9$
$x^2-6x+9=0$
$(x-3)^2=0$
x=3 (!!!çift katlı kök)

$T_1(n)=(A+Bn)3^n$
$T_1(0)=(A).1 = 1 \rightarrow A=1$

$T_1(1)=(A+B)3=6 \rightarrow 3A+3B=6 \rightarrow 3+3B=6 \rightarrow 3B=3 \rightarrow B=1$

$T_1(n) = (1+1.n)3^n \rightarrow T_1(n) = (1+n)3^n \rightarrow O(n.3^n)$

*** $T_2(n)= 5T_2(n-1) – 6T_2(n-2) + 7^n$**

lineer homojen olmayan denklem olduğundan dolayı homojen kısmı ayrı , homojen olmayan kısmı ayrı çözülür.

Homojen kısım için:

$x^2=5x-6 \rightarrow x^2-5x+6=0 \rightarrow (x-3)(x-2)=0 \rightarrow x_1=3 , x_2=2 \rightarrow T(n) = c_1 3^n + c_2 2^n$

Homojen olmayan kısım için:
$a_n=c_3 7^n$
$a_{n-1}=c_3 7^{n-1}$
$a_{n-2}=c_3 7^{n-2}$ formüllerini genel $T_2$ formülünde yerine yazacak olursak;

$T_2(n)= 5T_2(n-1) – 6T_2(n-2) + 7^n$
↓
$c_3 7^n = 5c_3 7^{n-1} – 6c_3 7^{n-2} + 7^n$ formülü elde edilir.
$7^n(c-1) = 7^{n-2}(7.5c_3 – 6c_3)$ (her iki tarafta $7^2$ ile çarpılır)
$49.7^n(c_3-1) = 7^n(29c_3)$ ($7^n$ ler sadeleşir)
$49c_3 - 49 = 29c_3 \rightarrow 20c_3=49 \rightarrow c_3=49/20$

$T_2$ için general form :

$T_2(n) = c_1 3^n + c_2 2^n + (49.7^n)/20 \rightarrow O(7^n)$