2017

# CSE 321 – Inroduction to Algorithm Design

HOMEWORK 2

RIDVAN DEMİRCİ
141044070

## THE IMITATION GAME

***İmitation game filmi ikinci dünya savaşındaki Almanya'nın yani nazilerin şifreli metinlerini çözme meselesini ele alan bir filmdir.O dönemde güçlü olan naziler kendi karargahları arasında yapıcakları operasyonları nerelere saldıracaklarını önceden şifrelenmş metinler halinde(Enigma) kendi karargalarına ulaştırıyolarmış metinler diğer düşmanların ellerinde ama şifreli oldukları için anlayamıyolardı.***

***Bu metinleri çözmek için bir ekip kurulur ekipte yer alan Alan Turing metinlerin makine tarafından çözülebileceğini söyler ve onun için ilk bilgisayarı kurdurur .Alan Turing 'i ilk başta ekip arkadaşları desteklememlerine rağmen en sonunda bu kadar metnin iş işten geçmeden çözülebileceğinin bir makine tarafında olağı konusunda hem fikir oldurlar ve bunun üzerine çalıştılar***

***Filmin ilerleyen anlarında gelen şifreli metnin anlamlı bir metne çeviren programı yapmışlardır ,yaptıkları makinenin adına ise christopher koymuşdur***

***Ve bu arada şifrelenmiş metinler Almanca olmasına rağmen Alan Turing Almanca bilmiyodur.Ilk metinleri çözdükleri zaman şifrelerin çözüldüğü anlaşılmaması için başlangıçtaki birkaç operasyona izin verilmiştir.Daha sonra gene metinlerin çözüldüğü anlaşılmaması için şifreleri kendileri adına kullanmamaş lar ve metinleri müttefiklerine vermiştir ve Almanyanın ikinci dünya savaşını kaybetme sebebi olmuştur.***

***Savaş bittikten sonra Turing makineyi dahada geliştirmiştir ve en sonunda zehirli elma yiyerek intihar etmiştir***

**Q2-)** Master teorem kurallına göre;

**Theorem (Master Theorem)**

*Let $T(n)$ be a monotonically increasing function that satisfies*

$$T(n) = aT(\tfrac{n}{b}) + f(n)$$
$$T(1) = c$$

*where* $a \geq 1, b \geq 2, c > 0$. *If* $f(n) \in \Theta(n^d)$ *where* $d \geq 0$, *then*

$$T(n) = \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Ve genelleştirilmiş formül olarak da Şu uygulanabilir.

# Master Theorem Summarized

- Given a recurrence of the form $T(n) = aT(n/b) + f(n)$ ,F(n) artan olmalı

1. $f(n) = O\left(n^{\log_b a - \varepsilon}\right)$
   $\Rightarrow T(n) = \Theta\left(n^{\log_b a}\right)$
2. $f(n) = \Theta\left(n^{\log_b a}\right)$
   $\Rightarrow T(n) = \Theta\left(n^{\log_b a} \lg n\right)$
3. $f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$ and $af(n/b) \leq cf(n)$, for some $c < 1, n > n_0$
   $\Rightarrow T(n) = \Theta(f(n))$

$$*x_1(n) = 0.5x_1\left(\frac{n}{2}\right) + \frac{1}{n}$$

a=0.5 b=2 $x_1(1)$ = C

f(n) = $\frac{1}{n}$ ,f(n) bir harmonik seri… ve F(n) azalan bir fonksyon olduğu için master teorem ile çözülemez.

$$*x_2(n) = 3x_2\left(\frac{n}{4}\right) + \ nlogn$$

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

$log_b a = log_4 3$  f(n) Є Ω($n^{log_4 3}$)

2.resimdeki son kurala göre c < 1 için af(n/b)<c.f(n) için

$3\frac{n}{4}.\log\frac{n}{4}$ < c.nlogn  c<1 c = $\frac{n}{4}$ için sağlar.

Sağlayan c değeri de mevcuttur ,bu durumda sonuç Ө(nlogn)'dir.

$$*x_3(n) = 3x_3\left(\frac{n}{3}\right) + \ \frac{n}{2}$$

a = 3

b = 3

f(n) Є Θ($n^1$)  d = 1

$b^d$= 3  a = b olduğundan sonuç, Ө(f(n)logn) ' dir.

Yani Θ($n^1$logn)

$$*x_4(n) = 6x_4\left(\frac{n}{3}\right) + n^2logn$$

a = 6

b = 3

f(n) = $n^2logn$

$n^{log_b a} = n^{log_3 6}$ => $n^2logn$ Є Ω($n^{log_3 6}$)

af(n/b)<c.f(n) c < 1 icin ;

$\frac{6n^2}{9}.log\frac{n}{3} < c.n^2 logn$ c = $\frac{6}{9}$ olursa denklem sağladığına göre bu durumda 2. Resimdeki 3. Kurala göre cevap Θ(f(n)) ‘ dir. Yani **Θ($n^2 logn$).**

$$*x_5(n) = 4x_5\left(\frac{n}{2}\right) + \frac{n}{logn}$$

a = 4

b = 2

f(n) = $\frac{n}{logn}$

$n^{log_b a} = n^{log_2 4} = n^2$ , f(n) Є O($n^2$)

Bu durumda ikinci resimdeki birinci kural uygulanır.Yani

T(n) = Θ($n^{log_b a}$) ,T(n) = Θ($n^2$)

$$*x_6(n) = 2^n x_6\left(\frac{n}{2}\right) + n^n$$

Bu kural master teorem ile çözülemez çünkü format master teorem değil.

Format: T(n) = aT(n/b)+f(n)

a = 1

b = 2

f(n) = $n^n$ ancak $2^n$ yüzünden master teorem uygulanamaz

# Q3-)

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

**a-)** Set up a recurrence relation for this function's values and solve it to determine what this algorithm computes.

Fonksiyon değerleri için tek tek denenirse ;

n = 1 için T(n) = 1 gerisi için ise tekrar recursive çalışır.

Yani $T(n) = \begin{cases} 1 & , n = 1 \\ T(n-1) + 2n - 1, & diğer\ durumlar \end{cases}$

Relation'u kurduktan sonra çözüm için backward subsutition yapılır.

| |
|---|
| T(1) = 1 |
| T(2) = T(1) + 3 |
| T(3) = T(2) +5 |
| . . . |
| . . . |
| . . . |
| T(n) = T(n-1) + 2n - 1 |

İfadeler alt alta toplanırsa en son

T(n) = 1 + 3 +5 +7 +9 … 2n -1 kalır buda toplam sembolünden T(n) =$\sum_{i=1}^{n} 2i - 1$

T(n) =ϴ($\boldsymbol{n^2}$)

**b-)** Set up a recurrence relation for the number of multiplications made by this algorithm and solve it.

M(n) multiplication'ın performans polinomunu göstersin

n = 1 iken çarpma sayısı 0 olur. Gerisi için recursive + 1 olur.

Yani denkleme dökersek;

$$M(n) = \begin{cases} 0 & , \quad n = 1 \\ M(n-1) + 1, & \quad diğer\ durumlar \end{cases}$$

Relation'u kurduktan sonra çözüm için backward subsutition yapılır.

M(1) = 0

M(2) = M(1) + 1

M(3) = M(2) +1

. . .

. . .

. . .

M(n) = M(n-1) + 1

İfadeler alt alta toplanırsa ;

M'ler götürür birbirini ve enson

**T(n) =$\sum_{i=1}^{n-1} 1$ =1*(n-1)**

**M(n) = 1*(n-1)** olur ve sonuç olarak

**M(n) Є ϴ(n) çıkar.**

**C-)** Set up a recurrence relation for the number of additions/subtractions made by this algorithm and solve it

S(n) multiplication'ın performans polinomunu göstersin n = 1 iken toplama ve çarpma  sayısı 0 olur. Gerisi için

recursive + 2 olur.      (choclateAlgo(n-1) **+** 2* **-**1)

Yani denkleme dökersek;

$$S(n) = \begin{cases} 0 & , \quad n = 1 \\ S(n-1) + 2, & \text{diğer durumlar} \end{cases}$$

| |
|---|
| S(1) = 0 |
| S(2) = S(1) + 2 |
| S(3) = S(2) +2 |
| .  .  . |
| .  .  . |
| .  .  . |
| S(n) = S(n-1) + 2 |

İfadeler alt alta toplanırsa ;

M'ler götürür birbirini ve en son

**T(n) =$\sum_{i=1}^{n-1} 2$ =2*(n-1)**

**S(n) = 2*(n-1)** olur ve sonuç olarak

**S(n) Є ϴ(n) çıkar.**

**Q5-) Explain your algorithm. Show this algorithm's number of operations in terms of input array size (n) and complexity using the asymptotic notations for best and worst cases**

```
def findRotten(listOfwalnut,size):  #logaritmik time da olması icin array size da girilmeli

    if(size == 1):
        return 0
    if(size % 2 == 0):      # cift ise direk yarirsi ile islemler yapar
        temp = compareScales(listOfwalnut[0:size/2],listOfwalnut[size/2:]) # sol ve sag tarafın farkı alınır
        #print temp
        if(temp == 1):
            return findRotten(listOfwalnut[0:size/2],size/2) # sag taraf buyuk ise rotten sol taraftadir
        elif(temp == -1):
            return size/2 + findRotten(listOfwalnut[size/2:],size/2)
    else:
        temp = size/2;
        tempSum = compareScales(listOfwalnut[0:temp],listOfwalnut[temp+1:])
        if(tempSum == 0):           #sag ve sol esit ise ortadaki eleman en kucuktur
            return 1
        elif(tempSum == -1):
            return temp + 1 + findRotten(listOfwalnut[temp+1:],temp)    #yaridan sonra
        else:
            return findRotten(listOfwalnut[0:temp],temp)    #yaridan once
```

**b-) algoritma input olarak input array ve array size alır,array size arrayin boyutunu tekrar almamak için yoksa time complexity fazla olur.eğer size sıfır ise dire çıkar döngüden**

**array size'ın tek veya çift olma durumuna göre iki farklı if statementı var çift ise sol ve sağı CompareScale() fonksyonuna verir fark var ise az olan tarafı tekrar findrotten()'a gönderir.**

**Eğer size tek ise ortadakini alır fark var ise az olan tarafı tekrar gönderir findRotten()'a eğer fark yok ise ozman ortadaki sayı rotten walnut tır.**

**İndis bulma durumu ise eğer rotten sol tarafta ise indis de değişiklik yapmaz,ama sağ tarafta ise indise size/2 eklenir.**

$$T(n) = \begin{cases} O(1) \;, & n == 1 \\ 2T\left(\frac{n}{2}\right) + 1, & others \end{cases}$$

**Best case gelen inputun 1 olması budurmda B(n) = ϴ(1).**

**Worst case ise rotten walnut'ın L[0] ve L[n-1] konumda olması. Budurumda ise complxity**

**B(n) = ϴ(logn) olur çünkü iterasyonda yarısını eliyeceği için.**

## 6.) **Solve the following recurrence relations,**

**a)** Using forward/backward substitution:

$*\boldsymbol{T_1 = 3T_1(n-1) \quad , \quad for\ n > 1\ , \quad T_1(1) = 4}$

- Backword subsitition kullanılırsa…

$T_1(1) = 4$

$T_1(2) = 3T_1(1)$

$T_1(3) = 3T_1(2) = 3.3.T_1(1)$

$T_1(4) = 3T_1(3) = 3.3.3.T_1(1)$

.            .            .

.            .            .

.            .            .

$\boldsymbol{T_1(n) = 3.\underbrace{3.3\ \ldots\ldots 3.T_1(1)}_{\text{(n-1 adet)}} = 3^{n-1}T(1) = 4.3^{n-1} = \Theta(3^n)}$

---

$*\boldsymbol{T_2(n) = T_2(n-1) + n \quad , \quad for\ n > 1\ , \quad T_2(0) = 0}$

$\boldsymbol{T_2(0) = 0}$

$\boldsymbol{T_2(1) = T_2(0) + 1}$

$\boldsymbol{T_2(2) = T_2(1) + 1 = T_2(0) + 2}$

$\boldsymbol{T_2(3) = T_2(2) + 1 = T_2(0) + 3}$

.            .            .

.            .            .

$\boldsymbol{T_2(n) = T_2(n-1) + 1 = T_2(0) + 1 + 2 + 3 + \cdots + n}$

$$T_2(n) = \sum_{i=0}^{n} i = \frac{n.(n+1)}{2} = \Theta(n^2)$$

$$T_3(n) = T_3\left(\frac{n}{2}\right) + n\,,\ \ for\ n > 1\,,\ T_3(1) = 0\,, for\ n = 2^k$$

| | |
|---|---|
| $T_3(1) = 0$ | k = 0 |
| $T_3(2) = 0 + 2$ | k = 1 |
| $T_3(4) = 0 + 2 + 4$ | k = 2 |
| $T_3(8) = 0 + 2 + 4 + 8$ | k = 3 |
| . . . . . . | . |
| . . . . . . | . |
| $T_3(n) = 0 + 2 + 4 + \cdots + 2^k$ | k = $log_2 n$ |

Summation olarak yazarsak , $T_3(n) = \sum_{i=1}^{k+1} 2^i = \frac{1-2^{k+1}}{1-2}$

$$= 2^{k+1} - 1 = 2.2^{log_2 n} - 1 = 2.n - 1 = \Theta(n)$$

**b)** **Using the properties of linear homogeneous/inhomogeneous equations:**

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

$T_1(n) - 6T_1(n-1) + 9T_1(n-2) = 0$

$r^n - 6.r^{n-1} + 9.r^{n-2} = 0$ **,denklem** $.r^{n-2}$ **'e bölünür.Ve sonuç;**

$r^2 - 6r + 9 = 0$ **buda eşittir** $(r-3)^2 = 0$ **buradan r çif katlı kök olarak r=3 .**

**Homogeneous denklemine göre çift katlı kök için yazarsak .**

$$T(n) = \alpha\, 3^n + B.n.3^n$$

**n = 0 için T(0) =** $\alpha$ **,** $\alpha = 1$ **; n=1 için T(1) = 1.3 +** $B.1.3$ $B = 1$

$T(n) = 3^n + .n.3^n$ **bu denleme göre karmaşıklık.** $\Theta(n.3^n)$

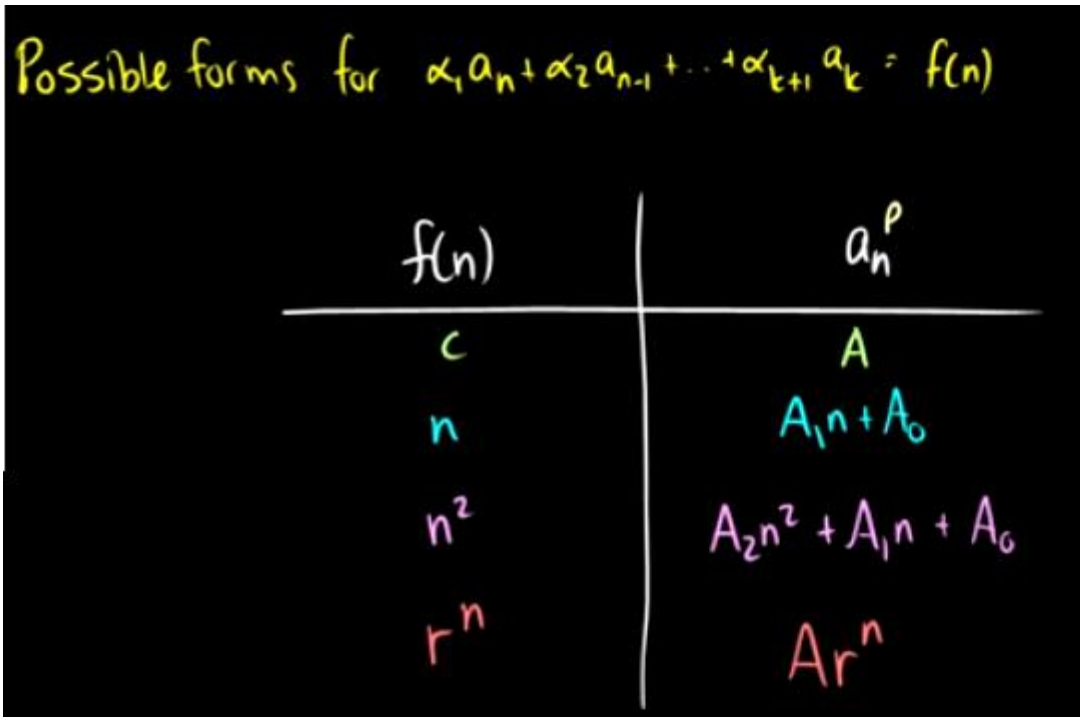
$$T_2(n) = 5T_2(n-1) - \ 6T_2(n-2) + 7^n,$$

Denklem non-homogenous olduğu için hem homojen hem de particular solution yapmamız gerekir.

$$T(n) = \ T_h(n) + \ T_p(n)$$

Homjen kısım normal olarak çözülür, particular yokmuş gibi ;

$$T_2(n) - 5T_2(n-1) + 6T_2(n-2) = 0\,, \qquad r^2 - 5r + 6 = 0$$



Possible forms for $\alpha_1 a_n + \alpha_2 a_{n-1} + \dots + \alpha_{k+1} a_k = f(n)$

| $f(n)$ | $a_n^p$ |
|---|---|
| $c$ | $A$ |
| $n$ | $A_1 n + A_0$ |
| $n^2$ | $A_2 n^2 + A_1 n + A_0$ |
| $r^n$ | $A r^n$ |

$$= (r-3)(r-2) = 0 \quad r = 2\, r = 3 \quad ,$$

$T_h(n) = a2^n + B3^n$ *denklemi çıkar*

Particular çözüm için tahmin yapılır bu tahmin ise şu tabloya dayanır

$T_2(n) = 5T_2(n-1) - 6T_2(n-2) + 7^n$, particular f(n) =$7^n$ için tahmin olarak T(n) = $A7^n$ yapılır ve denklemde yerine yazılır.

$A7^n - 5A7^{n-1} + 6A7^{n-2} = 7^n$ her taraf $7^n$ 'e bölünürse A = $\frac{49}{20}$ gelir. $T_p(n) = \frac{49}{20}7^n$ 'gelir.

$$T(n) = T_h(n) + T_p(n) = a2^n + B3^n + \frac{49}{20}7^n$$

Bu sonuca göre karmaşıklık ϴ($7^n$) 'gelir

Resimlerin referansları

https://www.google.com.tr/search?q=master+teorem&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjR8L2skpPXAhXFJ5oKHRFOA34Q_AUICigB&biw=1366&bih=662#imgrc=kZo_2H3NqIJ6DM:

https://www.google.com.tr/search?q=master+teorem&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiTjaXh_ZPXAhXId5oKH

videodan resim kestim

https://www.youtube.com/watch?v=EfF_XSEX1SkWShCksQ_AUICigB&biw=1366&bih=662#imgrc=oc8FlWpiTHKDSM: