CSE 321
INTRODUCTİON TO ALGORITHM DESING
HOMEWORK 02

Şevval MEHDER
151044009

Gebze Teknik Üniversitesi
2017 Fall

Q1.

II. Dünya Savaşı sırasında Nazi Almanyası tarafından tüm önemli iletişimlerinde kullanılan bir makinedir Enigma. Herkesin görebildiği ama anahtarı yoksa kimsenin ne anlama geldiğini çözemeyeceği mesajlardır. Her gece yarısında ayarlarının değiştirildiği, 159 trilyon olası ayar ve müttefik güçlerin çözebilmesi için sadece 18 saatinin olduğu bir düzen.

Hükümet Enigma kodlarını çözmek için bir takım kurmuştu. Makinelere karşı makinelerin kazanabileceğine inanan Alan Turing de ekibe dahil olmuş ve birtakım zorluklardan sonra takımın liderliğini üstlenmiştir. “Burada yarattığım şeyin önemini asla anlayamayacaksınız” dediği makinesi için akla gelen ilk yöntemi kurgulamıştır: Olası tüm ihtimalleri denemek. Şifrenin kodlanmış olabileceği tüm olasılıkları buluyor ve daha sonra da doğruluğunu kontrol ediyordu. Ancak makine sadece çalışmakla kaldı. Sürekli çalışıyor olasılıkları deniyor ama gerekli sürede bir sonuç üretemiyordu.

Zamanla savaşta olduklarını bilen Turing ve ekibi daha hızlı olmanın yollarını arıyorlardı. Akıllarına gelen hiçbir şey çözüm üretmedi. Ancak bir gün Alan, başarıya ulaşacakları çözümü buldu. Peki ya makine tüm olasılıkları denemek yerine sadece şifreli mesajın içinde olduğunu bildikleri kelimeleri araştırsaydı? Yani üzerinde çalıştığı kümeyi daraltarak. Tekrarlanan ya da tahmin edilebilen. Her sabah mutlaka yayınlanan hava durumu mesajları gibi…

Elinizde çalıştığınız küçük bir veri ve çok da kompleks olmayan bir işlem için hemen aklınıza gelip kodladığınız brute-force size pek bir zarar vermiyor. Aksini yapmanın da size bir zaman kazandırmayacağını düşünüyorsunuz. Ancak görünüşe göre imkansız denen Enigma’yı çözmenin yolu o zamanlarda üzerinde çalışılan algoritmanın karmaşıklığını azaltmaktan geçiyordu 😜

Q2. Summary: $T(n) = a.T\left(\frac{n}{b}\right) + f(n)$, where $\cdot\ a \geq 1$ $\cdot\ b > 1$

$+\ f(n) \in \Theta(n^d)$

$$T(n) = \begin{cases} \Theta(n^{\log_b a}) & , \ a < b^d \ (d > \log_b a) \\ \Theta(n^{\log_b a} \lg n) & , \ a = b^d \ (d = \log_b a) \\ \Theta(f(n)) & , \ a > b^d \ (d < \log_b a) \end{cases}$$

condition: $af(n/b) < cf(n)$

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

$a = 0.5,\ b = 2\ \ f(n) = 1/n \rightarrow \quad a = 0.5 < 1 \rightarrow$ Doesn't apply!

$- \ x_2(n) = 3x_2\left(\frac{n}{4}\right) + n\lg n$

$a = 3,\ b = 4 \quad f(n) = n\lg n$

Compare $n^{\log_b a}$ and $f(n) \rightarrow n^{\log_4 3} < n\lg n \rightarrow$ (Case 3)

Regularity control

$af(n/b) < c.f(n)$

$3.(n/4)\lg(n/4) \leq c.n.\lg n \quad$, for $c = 3/4$ ✓

Res. for master theorem, $T(n) = \Theta(f(n)) = \Theta(n\lg n)$

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

$a = 3,\ b = 3,\ f(n) = \frac{n}{2}$

$f(n) \in \Theta(n^1) \rightarrow d = 1$

compare $\log_b a$ and $d \rightarrow \log_3 3 = 1 = d \rightarrow$ (Case 2)

$T(n) = \Theta(n^{\log_b a} \lg n) = \Theta(n \lg n)$

$- \ x_4(n) = 6\, x_4\left(\frac{n}{3}\right) + n^2\lg n$

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

Compare $n^{\log_b a}$ and $f(n) \rightarrow n^{\log_3 6} < n^2.\lg n \rightarrow$ (Case 3)

Regularity control:

$a.f(n/b) \leq c.f(n)$

$6.\left(\frac{n}{3}\right)^2.\lg\left(\frac{n}{3}\right) \leq cn^2.\lg n \quad$, for $c = 2/3$ ✓

$T(n) = \Theta(f(n)) = \Theta(n^2\lg n)$

$- \ x_5(n) = 4x_5\left(\frac{n}{2}\right) + \frac{n}{\lg n}$

$a = 4,\ b = 2,\ f(n) = \frac{n}{\lg n}$

Compare $n^{\log_b a}$ and $f(n) \rightarrow n^{\log_2 4} = n^2 > \frac{n}{\lg n} \rightarrow$ (Case 1)

$T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$

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

$a = 2^n,\ b = 2,\ f(n) = n^n$

a is 2^n, not constant! $\rightarrow$ Doesn't apply!

Q3.

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

$$T(n) = \begin{cases} 1 & , \quad n = 1 \\ T(n-1) + 2n - 1 & , \quad otherwise \end{cases}$$

$$\cancel{T(1)} = 1$$
$$\cancel{T(2)} = \cancel{T(1)} + 2.2 - 1 = 4$$
$$\cancel{T(3)} = \cancel{T(2)} + 2.3 - 1 = 9$$
$$\cancel{T(4)} = \cancel{T(3)} + 2.4 - 1 = 16$$
$$\cancel{T(5)} = \cancel{T(4)} + 2.5 - 1 = 25$$
$$\vdots$$
$$T(n) = \cancel{T(n\text{-}1)} + 2n - 1$$
$$+\underline{\hspace{8cm}}$$
$$T(n) = 2\,(2 + 3 + \ldots + n) - (1.\,(n\text{-}2))$$
$$T(n) = 2\left[\frac{n(n+1)}{2} - 1\right] - n + 2$$
$$T(n) = n^2 + n - 2 - n + 2$$
$$T(n) = n^2$$

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

$$T(n) = \begin{cases} 0, & n = 1 \\ T(n-1) + 1, & otherwise \end{cases}$$

$$T(1) = 0$$
$$T(2) = T(1) + 1 = 1$$

$$T(3) = T(2) + 1 = 2$$

$$T(4) = T(3) + 1 = 3$$

$$\vdots$$

Conjecture that for all $n > 0$, $T(n) = n - 1$

To show that:

$T(1) = 0$
$T(1) = 1 - 1 = 0$ ✓

$T(n) = T(n\text{-}1) + 1$
$n - 1 = (n - 1 - 1) + 1$ ✓

So, $T(n) = n - 1$

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

$$T(n) = \begin{cases} 0, & n = 1 \\ T(n-1) + 2, & otherwise \end{cases}$$

$$T(1) = 0$$

$$T(2) = T(1) + 2$$

$$T(3) = T(2) + 2 = 4$$

$$T(4) = T(3) + 2 = 6$$

$$\vdots$$

Conjecture that for all $n > 0$, $T(n) = 2n - 2$

To show that:

$T(1) = 0$

$T(1) = 2.1 - 2 = 0$ ✓

$T(n) = T(n\text{-}1) + 2$
$2n - 2 = (2(n\text{-}2) - 2) + 2$
$2n - 2 = (2n - 4) + 2$ ✓

So, $T(n) = 2n - 2$

Q5.

Explain your algorithm.

Algoritma öncelikle verilen listenin ortancasını bulur. Daha sonra listenin tek ya da çift olmasına göre bu listeyi sağ ve sol eşit parçalara böler. Eşit bölmek için:

- Tekse, ortanca elemanı sağa ve sola dahil etmez
- Çiftse, ortanca elemanı sağa dahil eder.

Bu iki listeyi compareScales fonksiyonuna gönderir.

- Sonuç 0 ise, rotten walnut middle indexteki elemandır. Bu index return edilir.
- Sonuç 1 ise, rotten walnut sol listededir. Sadece sol listede aramaya devam edilir. Sol liste ile birlikte recursive çağrı yapılır.
- Sonuç -1 ise, rotten walnut sağdadır. Sadece sağ listeyi aramak için recursive çağrı yapılır. Solda aramaya yapmayacak elemanların sayısı kadar değerle toplayarak return eder çünkü index hesaplanırken soldakiler de hesaba katılmalıdır. Tek ve çift olma durumunda ortancanın eklenip eklenmemesi değiştiği için duruma göre 1 eklenir.

Show this algorithm's number of operations in term of input array size (n) and complexity using the asymptotic rotations for best and worst case.

Liste her seferinde n/2 olmak tadır.

$$T(n) = \begin{cases} 1, & n = \text{middle} \\ T(\frac{n}{2}) + 1, & \text{otherwise} \end{cases}$$

toplama işlemi.

Her seferinde constant işlem yapmaktadır.

$$\frac{n}{2^i} = 1 \rightarrow i = \log n$$

<u>Best case</u>: Aranan elemanın ortanca eleman olması

$$B(n) \in \Theta(n)$$

<u>Worst case</u>: Aranan listede mutlaka olduğu varsayılırsa en sağ ya da en solda olması worst case olur.

Aynı şekilde yarıya bölünerek ilerlediği için

$$W(n) \in O(\log n)$$

Q6.

a) Using forward/backward substitution:

- $T1(n)=3T1(n-1)\ for\ n>1, T1(1)=4$

$$T_1(\mathrm{n}) = 3\ T_1\ (\mathrm{n}\text{-}1)$$
$$= 3\ [3\ T_1(\mathrm{n}-2)] = 3^2\ T_1\ (\mathrm{n}-2)$$
$$= 3^2[3\ T_1(\mathrm{n}-3)] = 3^3\ T_1\ (\mathrm{n}-3)$$
$$\vdots$$
$$= 3^{\mathrm{i}} T_1\ (\mathrm{n}-\mathrm{i}\ )$$
$$\vdots$$

For last step we must get T(1), so i = n – 1

$$= 3^{\mathrm{n}-1}\ T_1\ (1)$$

$$T_1\ (\mathrm{n}) = 3^{\mathrm{n}-1}.\ 4$$

- $T2(n)=T2(n-1)+n\ for\ n>1, T2(0)=0$

$$T_2\ (\mathrm{n}) = T_2\ (\mathrm{n}-1) + \mathrm{n}$$
$$= [T_2\ (\mathrm{n}-2) + (\mathrm{n}\text{-}1)] + \mathrm{n}$$
$$= [[T_2\ (\mathrm{n}\text{-}3) + (\mathrm{n}\text{-}2)] + (\mathrm{n}-1)] + \mathrm{n}$$
$$\vdots$$
$$= T_2\ (\mathrm{n}-\mathrm{i}) + (\mathrm{n}-\mathrm{i}+1) + (\mathrm{n}-\mathrm{i}+2) + \ldots\ldots + \mathrm{n}$$

For last step we must get T_2 (0), so i = n

$$= T_2\ (0) + 1 + 2 + \ldots + \mathrm{n}$$
$$= 0 + \frac{\mathrm{n}(\mathrm{n}+1)}{2}$$

$$T_2\ (\mathrm{n}) = \frac{\mathrm{n}(\mathrm{n}+1)}{2}$$

- $T3(n)= T3(n/2)+n \; for \; n>1, T3(1)=0$ ($solve \; for \; n=2^k$)

$$n = 2^k$$

$$T_3\ (2^k) = T_3(2^{k-1}) + 2^k$$

$$= [T_3\ (2^{k-2}\ ) + 2^{k-1}] + 2^k$$

$$= [[T_3\ (2^{k-3}) + 2^{k-2}]\ 2^{k-1}\ ] + 2^k$$

$$\vdots$$

$$= T_3\ (2^{k-i}\ ) + \ 2^{k-i+1} + \ 2^{k-i+2} + \cdots + \ 2^k$$

$$\vdots$$

For last step we must get T_3 (0), so i = k

$$= T_3(1) + \ 2^1 + \ 2^2 + \cdots + \ 2^k$$

$$= 0 + + 2^1 + \ 2^2 + \cdots + \ 2^k$$

$$= 0 + \frac{1- 2^{k+1}}{1-2}$$

$$= 2^{k+1} - 1$$

$$= 2\ .\ 2^k - 1$$

2^k *ifadesinin n değerine eşit olduğu düşünüldüğünde*

= 2. n – 1

T_3 (n) = 2n − 1

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

$$- \underbrace{T_1(n)}_{a_n} = 6\underbrace{T_1(n-1)}_{a_{n-1}} - 9\underbrace{T_1(n-2)}_{a_{n-2}}, \quad T_1(0) = 1, \ T_1(1) = 6$$

$$a_n = 6a_{n-1} - 9a_{n-2}$$

$$r^2 - 6r + 9 = (r-3)^2 = 0$$

→ If we have one root r, then

$$a_n = \alpha_1 . r^n + \alpha_2 . n . r^n$$

$$a_n = \alpha_1 . 3^n + \alpha_2 . n . 3^n$$

$n = 0$ $\quad a_n = 1 = \alpha_1 . 3^0 + \alpha_2 . 0 . 3^0 \longrightarrow \alpha_1 = 1$

$n = 1$ $\quad a_n = 6 = \alpha_1 . 3 + \alpha_2 . 1 . 3$

$$= 3 + \alpha_2 . 3 \longrightarrow \alpha_2 = 1$$

So, $a_n = 3^n + n . 3^n$

$$T_1(n) = 3^n + n . 3^n$$

$$= \underline{\underline{3^n (1+n)}}$$

$$- T_2(n) = 5T_2(n-1) - 6T_2(n-2) + \underline{\underline{7^n}},$$ (find the particular solution and then form of the real solution)

* This is a linear nonhomogenous recurrence relation

$$a_n = 5a_{n-1} - 6a_{n-2}$$

$$r^2 - 5r + 6 = (r-3).(r-2) = 0 \quad r_1 = 3, \ r_2 = 2$$

→ If we have two distinct real roots r_1 and r_2, then

homogenous part $\longleftarrow a_n^{(h)} = \alpha_1 . r_1^n + \alpha_2 . r_2^n$

$$= \alpha_1 . 3^n + \alpha_2 . 2^n + \boxed{C . 7^n} \rightarrow \text{particular}$$

particular part $\longleftarrow a_n^{(p)} = C . 7^n$ ↓ replace

$$C.7^n = 5C.7^{n-1} - 6C.7^{n-2} + 7^n$$

$$= 7^n \left[\frac{5C}{7} - \frac{6C}{49} + 1 \right]$$

$$49C.7^n = 7^n [35C - 6C + 49] \rightarrow 20C = 49 \rightarrow C = 49/20$$

So $\quad a_n = \alpha_1 . 3^n + \alpha_2 . 2^n + 49/20 . 7^n$, α_1 and α_2 are constant