CSE 321 - HW2

Muzaffer Metehan ALAN
151044038

1)

Öncelikle bu filmin konusu olan "Alan Turing" ve onun 2. Dünya savaşı esnasındaki harika düşüncesiyle Nazi Almanyasının Enigma'sının şifrelerinin çözülmesi neredeyse bilgisayar bilimlerinin temelidir. Almanya'nın Enigma'sı oldukça karışıktı. Alet mesajları rotor ve elektrik akımları yardımıyla milyonlarca farklı şekle dönüştürüyordu. Ve bundan faydalanan Almanya başlayacak olan savaş için büyük avantaj sahibiydi

Ancak Turing olası bir çözümlemenin toplam bağlamında çok büyük sayılardan olduğunu farketti. Kısacası Turing'in bir mesajdaki kelimelerin trilyonlarca olası çözümlemelerini çıkarıp, sadece işe yarayanları ayırabilen bir makineye ihtiyacı vardı. Sonuç olarak bu makine geliştirilerek Alman şifreleri çözülmüştür.

2) Master Theorem

$$T(n) = a\,T(n/b) + f(n)$$

where $a \geq 1$, $b > 1$ are constants and $f(n)$ is an asymptotically positive function

There are 3 cases.

1) If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$ then $T(n) = \Theta(n^{\log_b a})$

2) If $f(n) = \Theta(n^{\log_b a} \log^k n)$ with $k \geq 0$ then $T(n) = \Theta(n^{\log_b a} \log^{k+1} n$

3) If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ with $\varepsilon > 0$ then $T(n) = \Theta(f(n))$

- $x_1(n) = 0.5\, x_1\left(\frac{n}{2}\right) + \frac{1}{n}$ (It can't be solved by Master Theorem because $(a < 1)$)

- $x_2(n) = 3x_2\left(\frac{n}{4}\right) + n\log n$ ($n^{\log_4 3} < n\log n$ so $T(n) = \Theta(n\log n)$ <u>case 3</u>)

- $x_3(n) = 3x_3\left(\frac{n}{3}\right) + \frac{n}{2}$ ($T(n) = \Theta(n\log n)$ <u>case 2</u>)

- $x_4(n) = 6x_4\left(\frac{n}{3}\right) + n^2\log n$ ($f(n) = \Omega(n^{\log_3 6})$ so $T(n) = \Theta(n^2\log n$ <u>Case 3</u>

- $x_5(n) = 4x_5\left(\frac{n}{2}\right) + \frac{n}{\log n}$ ($f(n) = O(n^{\log_2 4})$ so $T(n) = \Theta(n^2)$) <u>case 1</u>

- $x_6(n) = 2^n x_6\left(\frac{n}{2}\right) + n^2$ (It can't be solved by Master Theorem because <u>a</u> is not constant)

3)

a) $T(n) = T(n-1) + 2n - 1 \rightarrow T(1) = 1$

$T(2) = \underbrace{T(1)}_{1} + 3 = 4$

$T(3) = \underbrace{T(2)}_{4} + 5 = 9$

$T(4) = \underbrace{T(3)}_{9} + 7 = 16$

Assume that $T(n) = n^2$

Prove : $T(k) = T(k-1) + 2k - 1$

$T(k) = (n-1)^2 + 2k - 1$

$n^2 - 2k + 1 + 2k - 1 = n^2$ so

$\boxed{T(n) = n^2}$

b) $T(n) = T(n-1) + 1$ → çarpma işlemi

$T(\cancel{2}) = T(\cancel{1}) + 1$

$T(\cancel{3}) = T(\cancel{2}) + 1$

$\vdots$

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

$(n-1) \cdot 1$

$T(n) = (n-1) \cdot 1 + 1$

$\boxed{T(n) = n}$

c)

toplama ve çıkarma

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

$T(\cancel{2}) = T(1) + 2$

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

$\vdots$

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

$T(n) = 1 + (n-1) \cdot 2$

$\boxed{T(n) = 2n - 1}$

5)

b) Bu algoritma öncelikle kendisine verilen array in size'ına göre belirli işlemler yapıyor

Bu işlemler;

Eğer size (array size'ı) çift ise array ortadan 2'ye ayrılır Left ve Right arrayleri oluşturularak verilen helper fonksiyonu ile elde edilen return değerine göre, eğer küçük değer soldaysa yani subarray (leftarray) olacaksa fonksiyon recursive olarak tekrar çağırılır. Eğer subarray (right array) olacaksa array size'nın yarısı toplama eklenecek şekilde recursive olarak fonksiyon tekrar çağrılır, bu böyle devam ederek index değeri return edilir

Eğer ki array size'ı tek sayı ise öncelikle arrayin ortadaki değeri yok sayılarak sol ve sağ kısımları helper fonksiyona verilir Eğer ki aranan küçük değer ortadaki değerse helper fonksiyon '0' değerini döndürecektir. Ve bu durumda küçük sayı bulunmuş olup indexi return edilir. Aksi taktirde '1' veya '-1' döndürmesine bağlı olarak recursive devam eder.

Worst Case:

$$T(n) = T(n/2) + \Theta(1)$$

$$T(n/2) = \underbrace{T(n/4) + \Theta(1)} + \Theta(1)$$

$$T(n) = T(n/4) + \Theta(1) + \Theta(1) + \Theta(1)$$

so $\rightarrow 2^k = n$

$T\left(\frac{n}{2^k}\right) = \quad +1 +1 +1 +\!+1 +1$ up to k

$= T(1) + k.(1)$

$2^k = n \quad k = \log_2 n$

$= T(1) + \log_2 n$

so $\boxed{T(n) = \log n}$

Best Case.

Arrayin tek sayı size'ı olması ve kaçok değerin ortanca indexte bulunmasıdır.

$$T(n) = 1$$

6) a)

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

$T(2) = 3 \cdot T(1) = 3 \cdot 4$

$T(3) = 3 \cdot T(2) = 3^2 \cdot 4$

$T(4) = 3 \cdot T(3) = 3^3 \cdot 4$

Guess $T(n) = 3^{n-1} \cdot 4$

ispat =

$T(k) = 3\underbrace{T(k-1)}$

$T(k) = 3 \cdot 3^{k-2} \cdot 4 = 3^{k-1} \cdot 4$

So $\boxed{T_1(n) = 3^{n-1} \cdot 4}$

2)

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

$T_2(1) = T(0) + 1$

$T_2(2) = T(1) + 2$

$T_2(3) = T(2) + 3$

$\vdots$

$T_2(n) = T(n-1) + n$

$T_2(n) = 1 + 2 + 3 + 4 + \dots n$

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

Scanned by CamScanner

6.a) 3)

$$T_3(n) = \cancel{T_3}(n/2) + n$$

$$\boxed{2^k = n}$$

$$\cancel{T3(2)} = T3(1) + 2$$

$$\cancel{T3(4)} = \cancel{T3(2)} + 4$$

$$T3(8) = \cancel{T3(4)} + 8$$

$\vdots$ $\qquad\qquad$ $\vdots$

$$T3(n) = \cancel{T3(n/2)} + n$$

$$T_3(n) = 2 + 4 + 8 + \cdots 2^{k-1} + 2^k + 1 - 1$$

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

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

$$T(n) = 2^k \cdot 2 - 2$$

$$n \cdot 2 - 2$$

$$\boxed{T(n) = 2n - 2}$$

---

6) b)

$$T1(n) = 6\,T1(n-1) - 9\,T_1(n-2), \quad T1(0) = 1, T1(1) = 6$$

$$x^2 = 6x - 9$$

$$x^2 - 6x + 9 = 0$$

$$(x-3)\cdot(x-3) = 0$$

$$T1(n) = a x_1^n + b n x_1^n$$

$$= a\, 3^n + bn \cdot 3^n$$

$$= 3^n(a + bn)$$

$$T_1(0) = a = 1$$

$$T_1(1) = 3\underline{a} + 3b \to b = 1$$

6)

b)

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

$x^2 - 5x + 6 = 0$

$(x-3)\cdot(x-2)$

$x_0 = 3 \qquad x_1 = 2$

$T(n)^h = c_1(3)^n + c_2(2)^n$

$T(n)^p = \alpha \cdot 7^{n+2} - 5\alpha\, 7^{n+1} + 6\alpha\, 7^n = 7^n$

$n=0 \rightarrow 49\alpha - 35\alpha + 6\alpha = 1$

$\alpha = 1/20$

$$\boxed{T(n) = 3^n c_1 + 2^n c_2 + 7^n (1/20)}$$