CSE 321

Introduction to Algorithm Design

HW2

Hakkı Erdem Duman

151044005

Q1)

The movie is about a part of Alan Turing's life. Alan Turing is the mathematician, who laid the fondations of the Computer Science. In this movie, he designed a machine called Christopher to beat Enigma, which is the machine that is designed to bring the messages secretly. It is clear to see that, a programmable stuff is a better thing to solve problems. A machine can think faster than a human brain.

Besides, the machine called Christopher changed the war's fate. That means, the machine changed the world's fate. These programmable stuffs are more important right now, and they will be the most important weapon of a war.

Q2)

$X_1(n)$ = Cannot be applied. Constant is less than one.

$X_2(n) = n^{\log_4 3} < n\log n \implies \theta(n\log n)$ Third case

$X_3(n) = n^{\log_3 3} \equiv \frac{n}{2} \implies \theta(n.\log n)$ Second case

$X_4(n) = n^{\log_3 6} < n^2\log n \implies \theta(n^2\log n)$ Third case

$X_5(n) = n^{\log_2 4} > \frac{n}{\log n} \implies \theta(n^2)$ First case

$X_6(n)$ = Cannot be applied. 2^n is not a constant.

<u>Cases</u>

1. If $f(n) = O(n^{\log_b a - e})$, then $T(n) = \theta(n^{\log_b a})$
2. If $f(n) = \theta(n^{\log_b a}\log^k n)$, then $\theta(n^{\log_b a}.\log^{k+1} n)$
3. If $f(n) = \Omega(n^{\log_b a + e})$, then $T(n) = \theta(f(n))$

Q3)

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

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

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

$T(n-2) = T(n-3) + 2n - 5$

So,

$T(n) = T(n-3) + 6n - 9$

$T(n) = T(n-k) + 2kn - \sum_{i=1}^{k} 2i - 1$

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

We need to make $T(n-k) = T(1)$. So $k = n-1$

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

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

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

$T(n) = n^2$

b) There is just one multiplication for each calling. So,

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

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

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

So,

$T(n) = T(n-3) + 1+1+1$

$T(n) = T(n-k) + \underbrace{1+1+1+\dots}_{k}$

For $k = n-1$,

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

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

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

c) There are two addition/substrations for each calling. So,

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

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

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

So,

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

$T(n) = T(n-k) + \underbrace{2+2+2 \dots +2}_{k}$

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

For $k = n-1$,

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

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

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

Q5)

a) Code

b) This algorithm finds the smaller value than the other list elements. To find that, we use a helper function called "compareScales". This function gets the left and the right side of the list and returns a flag value according to difference of their summation. However, there is trick because of the number of elements in the list. If the number of elements in the list is odd, left and right parts of the list will not involve the middle element.

After getting the flag value, index number will be changed with calling the function recursively. If the small value is at the left part of list, the new list will be the left part. If the small value is at the right part of list, the new list will be the right part and index will be increased as much as the right part's element number. Half of the remaining list's element number will be returned when the small number is found.

$T(n) = T(n/2) + O(1)$

As Master, Theorem;

$a = 1$

$b = 2$

$f(n) = O(1)$

$n^{\log_b a} \equiv O(1) \implies$ This is the case 2.

$n^0 \equiv O(1)$

If the situations are equal, it is multiplied by logn.

the result is $O(\log n)$

Best case is $O(1)$ becouse if the number of elements are odd and the smallest number is at the middle of list, it will be found in first move and will be returned.

Worst case is $O(\log n)$

Q6)

a) $T_1(n) = 3T_1(n-1)$, $T_1(1)=4$

$T_1(n+1) = 3T_1(n)$

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

$T_1(n+2) = 3.3.T_1(n)$

So,

$T_1(n+k) = \underbrace{3.3.3.3.3 \dots}_{k} T_1(n)$

$T(n) = \frac{T(n+k)}{3^k} \Rightarrow$ for $n=1 \Rightarrow T(1) = \frac{T(1+k)}{3^k}$

$= 4 = \frac{T(1+k)}{3^k} \Rightarrow 3^k.4 = T(1+k)$, for $k=n-1$

$= 3^{n-1}.4 = T(n)$

$T_2(n) = T_2(n-1) + n \quad , \quad T_2(0) = 0$

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

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

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

$T_2(n+4) = T_2(n+3) + n + 4$

$T_2(n+4) = T_2(n) + 4n + 10$

So,

$$T_2(n+k) = T_2(n) + kn + \sum_{i=1}^{k} i$$

$$= T_2(n) + kn + \frac{k^2+k}{2}$$

For $n=0$,

$$T_2(k) = T_2(0) + \frac{k^2+k}{2}$$

$$T_2(k) = 0 + \frac{k^2+k}{2}$$

For $k=n$,

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

$T_3(n) = T_3(n/2) + n \quad , \quad T(1) = 0$

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

$T_3(n/2^2) = T_3(n/2^3) + \frac{n}{2^2}$

So,

$T_3(n) = T_3\left(\frac{n}{2^3}\right) + \frac{n}{2^0} + \frac{n}{2^1} + \frac{n}{2^2}$

If we generalize that,

$T_3(n) = T_3\left(\frac{n}{2^k}\right) + \sum_{i=0}^{k} \frac{n}{2^i} \implies \frac{n}{2^k} = 1 \quad k = \log_2 n$

$$T_3(n) = T_3(1) + n \cdot \frac{1-\left(\frac{1}{2}\right)^{\log_2 n}}{1/2}$$

$$T_3(n) = n \cdot \frac{1-\left(\frac{1}{2}\right)^{\log_2 n}}{\frac{1}{2}}$$

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

$x^2 = 6x - 9$

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

$\underbrace{(x-3)(x-3) = 0}$

1 distinct root

$T_1(n) = a \cdot x_1^n + bn \cdot x_1^n$

$= a \cdot 3^n + bn \cdot 3^n$ ✓

$T_1(0) = a = 1$

$T_1(1) = 3a + 3b = 6 \implies b = 1$

So,

$T_1(n) = 3^n + n \cdot 3^n$

$T_1(n) = (n+1)3^n$

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

$a_n = a_n^h + a_n^p$

(a_n^h) $x^2 - 5x + 6 = 0$

$(x-3)(x-2) = 0$

$x_1 = 3, \; x_2 = 2$

$a_n^h = k \cdot 3^n + m \cdot 2^n$

(a_n^p) $a_{n+2} - 5a_{n+1} + 6a_n = 7^n$ Our Expectation $= A \cdot 7^n$

$$\frac{A \cdot 7^{n+2} - 5 \cdot A \cdot 7^{n+1} + 6 \cdot A \cdot 7^n = 7^n}{7^n} = 49A - 35A + 6A = 1$$

$A = 1/20$

So,

$a_n = a_n^h + a_n^p$

$= k \cdot 3^n + m \cdot 2^n + \frac{1}{20} \cdot 7^n$