Berkcan EKİCİ
171044015

Question 1:

master theorem rules: $a \geq 1$, $b > 1$, $f(n)$ is asymptotically positive function.

$$T(n) = a \cdot T(\frac{n}{b}) + \Theta(n^k \cdot \log^p n)$$

a) $T(n) = 16T(\frac{n}{4}) + n!$

$a = 16$, $b = 4$, $f(n) = n!$

$f(n) = \Omega(n^{\log_b^a})$, so $T(n) = \Theta(n!)$
$= O(n!)$

b) $T(n) = \sqrt{2}\,T(\frac{n}{4}) + \log n$

$a = \sqrt{2}$, $b = 4$, $k = 0$, $p = 1$

Now, $a = \sqrt{2} = 1.41$ and $b^k = 4^0 = 1$

Clearly $a > b^k$

So we have $T(n) = \Theta(n^{\log_b^a})$
$= \Theta(n^{\log_4^{\sqrt{2}}})$
$= \Theta(n^{1/4})$

c) $T(n) = 8T(\frac{n}{2}) + 4n^3$

$a = 8$, $b = 2$, $f(n) = 4n^3$

$f(n) = \Theta(n^{\log_b^a})$ $\Rightarrow T(n) = \Theta(n^{\log_2^8} \cdot \log n)$
$T(n) = \Theta(n^3 \cdot \log n)$

d) $T(n) = 64T(\frac{n}{8}) - n^2 \log n$

$a = 64$, $b = 8$, $f(n) = -n^2 \log n$

It cannot be solved Master Theorem. Because $f(n) < 0$

e) $T(n) = 3T(\frac{n}{3}) + \sqrt{n}$

$a = 3$, $b = 3$, $k = 1/2$, $p = 0$

Now $\log_b^a = \log_3^3 = 1$ $\quad 1 > 1/2$ so we follow case-01

$T(n) = \Theta(n^{\log_b^a}) = \Theta(n^{\log_3^3}) = \Theta(n)$

f) $T(n) = 2^n T(\frac{n}{2}) - n^n$

Doesn't apply, because a is not a constant. (2^n)

g) $T(n) = 3T(\frac{n}{3}) + \frac{n}{\log n}$

$a=3, b=3, f(n) = \frac{n}{\log n} \Rightarrow O(n^{\log_b^a})$

$T(n) = \Theta(n^{\log_3^3})$

$= \Theta(n)$

Berkcan EKİCİ
171044015

Question 2 :

Algorithm x

$$* \ T(n) = a.T(n/b) + \Theta(n^k.\log^p n)$$

a) $T(n) = 9.T(\frac{n}{3}) + n^2$

$a=9, b=3, k=2, p=0$

$a=b^k$ we follow case 2

$T(n) = \Theta(n^{\log_b^a}.\log^{p+1} n) = \boxed{O(n^2.\log n)}$

Algorithm y

b) $T(n) = 8.T(\frac{n}{2}) + n^3$

$a=8, b=2, f(n)=n^3 \rightarrow k=3, p=0$

$a=b^k$

We follow case 2

$T(n) = \Theta(n^{\log_b^a}.\log^{p+1} n) = \boxed{O(n^3.\log n)}$

Algorithm z

c) $T(n) = 2.T(\frac{n}{4}) + \sqrt{n}$

$a=8, b=4, f(n)=\sqrt{n} \Rightarrow f(n) = O(n^{\log_b^a}) \rightarrow \boxed{T(n) = O(n^{1.5})}$

$n^{\log_4^8} > \sqrt{n}$

Compare x and y $\Rightarrow \lim_{n\to\infty} \frac{n^2 \log n}{n^3 \log n} = \frac{1}{n} = 0$ $\quad Algo_x > Algo_y$

Compare x and z $\Rightarrow \lim_{n\to\infty} \frac{n^2 \log n}{n^{1.5}} = n^{0.5}\log n = \infty$ $\quad Algo_z > Algo_x$

Compare y and z $\Rightarrow \lim_{n\to\infty} \frac{n^3.\log n}{n^{1.5}} = n^{1.5}.\log n = \infty$ $\quad Algo_z > Algo_y$

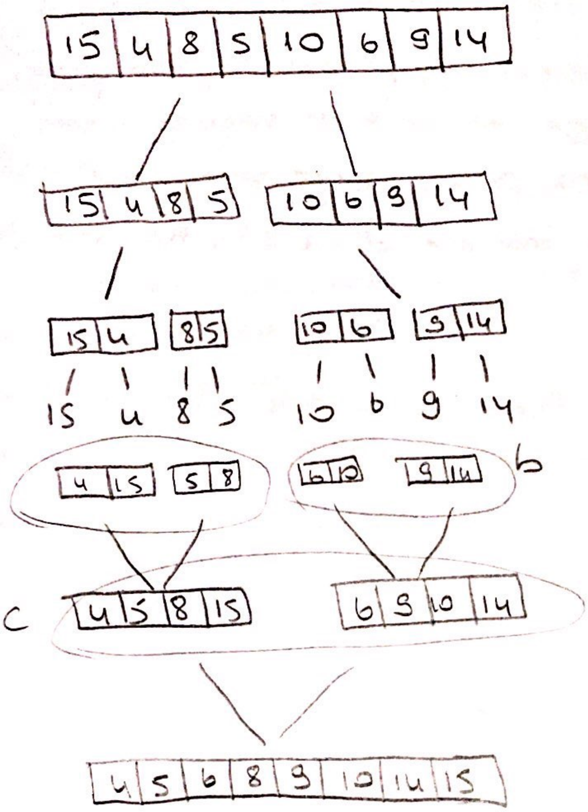
- I would choose Algorithm z because it is the fastest algorithm.

CamScanner ile tarandı

Berkcan EKİCİ
171044015

3)

a.i)



For doing comparison a, 4 compares itself with 6 and gets first place 15 compares itself with 2 and 4. (3 comparison occured that place is maximum.) b section same with a. At c comparison 4 compares with 6, puts itself at first place, 5 compares itself with 6. 8 compares itself with 6 and 9 : 15 compares itself with 9,10,14 and right side automatic puts themselves at the right place between 8 and 15

In the worst case and assuming a straight forword implementation, the number of comparisons to sort n element is

$n\lceil \lg n \rceil - 2^{\lceil \lg n \rceil} + 1$ lg n indicates the base-2 algorithms,

$8 \cdot 3 - 2^3 + 1 = 17$ comparison

Berkcan EKİCİ
171044015

3.a.ii)

→ [1,2,3,4,5,6,7,8]

The minimum number of comparisons for the merge step is approximately n/2, assuming a sane implementation once one of the lists has been fully traversed.

For example, if two lists that are effectively already sorted are being merged, then the first member of the larger list is compared n/2 times with the smaller list until it is exhausted; then the larger list can be copied over without further comparisons.

Minimum number of comparisons = 8/2 = 4

Berkan Ekici
171044015

3.b.i) I don't know the swap numbers. But I want to carry out an idea of the worst case, and I consider the case where there is the longest number of swaps to be the worst case.

Maximum swap number array → [8, 7, 6, 5, 4, 3, 2, 1] is reverse order and pivot is the first element.

3.b.ii) Minimum number of swap operation:
ordered array = [1, 2, 3, 4, 5, 6, 7, 8]

Berkcan EKİCİ
171044015

4) $T(n) = T(n/2) + c$

$T(n/2) = T(n/4) + c$

$T(n) = [T(n/4) + c] + c$

$= T(n/4) + 2c$

$T(n/4) = T(n/8) + c$

$T(n) = T(n/8) + 3c$

$T(n) = T(n/2^k) + k.c$

$n = 2^k$, $T\left(\frac{n}{2^k}\right) = T\left(\frac{2^k}{2^k}\right) = T(1)$

$\log_{base}^{result} = exponent$

$\log_2^n = k$

$T(n) = T(1) + \log_2 nc$

$= c + \log_2 nc$

$= c(1 + \log n)$

$= O(1 + \log n)$

$\boxed{= O(\log n)}$

or Master Theorem rules:

$T(n) = T(n/2) + c$

$a = 1,\ b = 2,\ f(n) = c$

$n^{\log_b^a} = n^{\log_2^1} = n^0 = 1$

$T(n) = \Theta(\log n)$