'Assignment - 01'.

Amirun Nahin

ID: 23201416

Sec: 06

Course: CSE221

Date: 17/03/2025

Ans to the que no - 01

```
def binary_search (A, n, T) :
    L := 0
    R := n-1
    first = None
    while L ≤ R do
        m := floor ((L+R)/2)
        if A[m] == T Then
            first = m
            R := m-1
        else if A[m] > T
            R := m-1
        else
            L := m+1
    if first is not None Then
        return first
    else
        return unsuccessful
```

Ans. to the que. no - 02

A

given,

$$T(n) = 2T(n/2) + \frac{1}{n}$$

using master theorem, we get,

$a = 2$
$b = 2$
$K = -1$
$P = 0$

here, $a > b^K \Rightarrow 2 > 2^{-1}$ so,

$$T(n) = \Theta(n^{\log_b a})$$
$$= \Theta(n^{\log_2 2})$$
$$= \Theta(n)$$

(Ans.)

B

given,

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

using master theorem, we get,

$a = 2$
$b = 3$
$K = 1$
$P = 0$

so, $a < b^k \Rightarrow 2 < 3^1$; also $p >= 0$. Therefore,

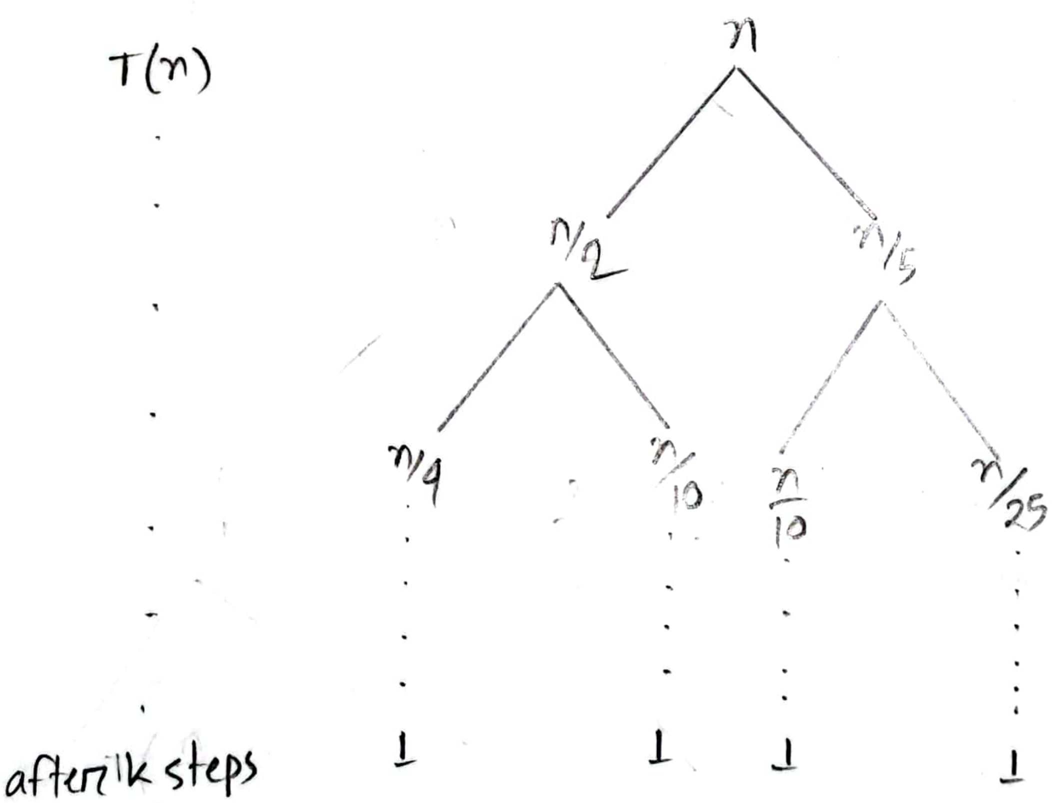
$$T(n) = \Theta(n^k \log^p n)$$
$$= \Theta(n^1 \log^0 n)$$
$$= \Theta(n)$$ (Ans.)

c

given,

$$T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{5}\right) + n$$

using recursive tree method,



total operation

n

$\frac{n}{2} + \frac{n}{5} = \frac{7n}{10}$

$\frac{n}{4} + \frac{n}{10} + \frac{n}{10} + \frac{n}{25} = \frac{49n}{100}$

$n + \frac{7n}{10} + \frac{49n}{100} + \dots$

so, we get a geometric series,

$$n + \frac{7n}{10} + \frac{49n}{100} + \cdots$$

$$= n + \frac{7n}{10} + \frac{7^2 n}{10^2} + \frac{7^3 n}{10^3} + \cdots$$

here, $a = n$

$r = \frac{7}{10}$

so, $T(n) = \frac{a}{1-r}$

$$= \frac{n}{1 - \frac{7}{10}}$$

$$= \frac{10n}{3}$$

$$= \theta(n)$$

(Ans.)

D

given, $T(n) = 2T\left(\frac{n}{4}\right) + n^2$

using master theorem, we get,

$a = 2$

$b = 4$

$k = 2$

$p = 0$

here, $a < b^k \Rightarrow 2 < 4^2$ also, $p >= 0$;

So, $T(n) = \Theta(n^k \log^p n)$

$= \Theta(n^2 \log^0 n)$

$= \Theta(n^2)$

(Ans.)

Ans. to the question no - 03

From the given function, the recurrance relation will be,

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

$= T(n-4) + 1 + 1$

$= T(n-4) + 2$

$= T(n-6) + 1 + 2$

$= T(n-6) + 3$

$\vdots$

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

here, $n - 2k = 0$

$\Rightarrow n = 2k$

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

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

$= T(0) + \frac{n}{2}$

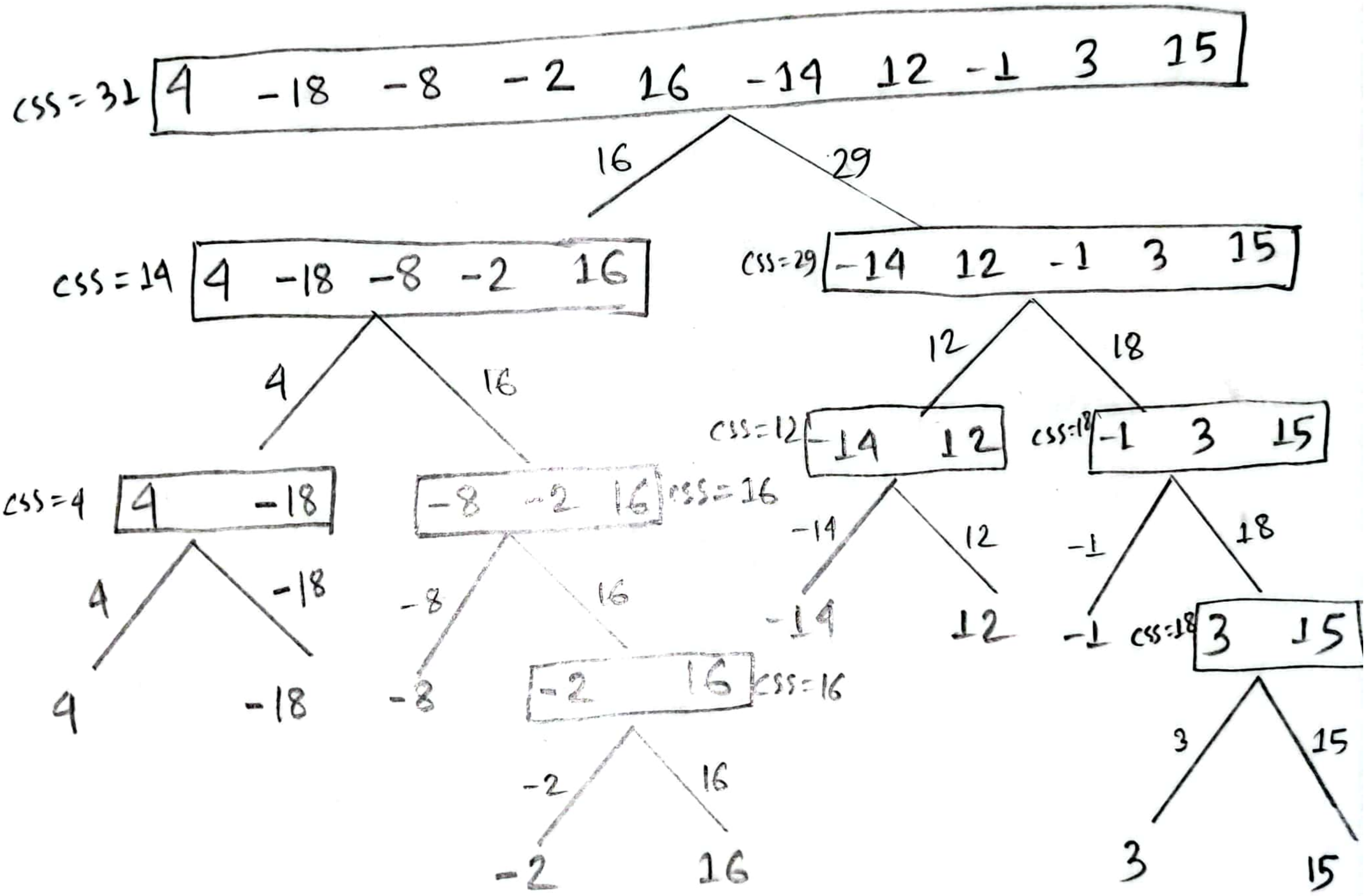
$$T(n) = 1 + \frac{n}{2}$$

$$T(n) = O(n)$$

(Ans.)

Ans. to the que. no - 04

We can use the following algorithm to maximize the farmer's profit,



So, the maximum profit will be 31 and the sequence of fields he needs to select will be

[16, -14, 12, -1, 3, 15]

b

from the simulation, we can see that the recurrance relation of this algorithm will be,

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

using master theorem,

$a = 2$
$b = 2$
$k = 1$
$p = 0$

here, $a = b^k \Rightarrow 2 = 2^1$ also, $p > -1$, so,

$$T(n) = \Theta\left(n^{\log_b a} \cdot \log^{p+1} n\right)$$

$$= \Theta\left(n^{\log_2 2} \cdot \log^{-1+1} n\right)$$

$$= \Theta\left(n^1 \cdot \log^0 n\right)$$

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

So, the time complexity of this algorithm is $\Theta(n\log n)$ which can be considered as an efficient algorithm.

Ans. to the question no-5

(i)

given, $f(n) = 4^n$ $\quad g(n) = 16^{\log_2(n)}$

Now, let if

$$f(n) = O(g(n))$$

$$\Rightarrow f(n) \le c \cdot g(n)$$

$$\Rightarrow 4^n \le c \cdot 16^{\log_2 n}$$

$$\Rightarrow 4^n \le c \cdot 2^{4\log_2 n}$$

$$\Rightarrow 4^n \le c \cdot 2^{\log_2 n^4}$$

$$\Rightarrow 4^n \le c \cdot n^4$$

This is not true, because 4^n is an exponential function and n^4 is a polynomial function. So, we know polynomial ~~is~~ grows slower than exponential. Therefore, 4^n cannot be bound by n^4.

$$\therefore f(n) \ne O(g(n))$$

However, if, $g(n) = O(f(n))$

$$\Rightarrow n^4 \le c \cdot 4^n$$

This is true for all, $n_0 = 1$, and $c = 1$ and $n \ge n_0$. So, $g(n) = O(f(n))$ (Ans.)

(ii)

given, $f(n) = (\sqrt{n}+n)\sqrt{n}$ $g(n) = n^2$

Let, firstly,

$$f(n) = O(g(n))$$

$$\Rightarrow f(n) \leq c \cdot g(n)$$

$$\Rightarrow (\sqrt{n}+n)\sqrt{n} \leq c \cdot n^2$$

$$\Rightarrow n + n\sqrt{n} \leq c \cdot n^2$$

$$\Rightarrow n(1+\sqrt{n}) \leq c \cdot n^2$$

$$\Rightarrow 1+\sqrt{n} \leq c \cdot n$$

This is always true for $c=2$, $n_0 = 1$. So, for all $c=2$ $n \geq n_0$ this equation satisfies.

Therefore, $f(n) = O(g(n))$

Again, $g(n) = O(f(n))$

$$\Rightarrow n^2 \leq c \cdot \sqrt{n}(n+\sqrt{n})$$

$$\Rightarrow n^2 \leq c \cdot (n\sqrt{n}+n)$$

$$\Rightarrow n^2 \leq c \cdot n(1+\sqrt{n})$$

$$\Rightarrow n \leq c \cdot (1+\sqrt{n})$$

here, as n grows linearly, there will always be a n for which this equation does not satisfy. So, we can say that, $g(n) \neq O(f(n))$

(Ans.)

Ans. to the que. no - 06

(a)

The assymptotic upper bound for the given functions are:

1. $f_1(n) = (\log n)^{2023} = O((\log n)^{2023})$
2. $f_2(n) = n^2 \log_n n^n = n^3 . \log_n n = n^3 = O(n^3)$
3. $f_3(n) = n^3 + 7n^2 = O(n^3)$
4. $f_4(n) = 2.023^n = O(2.023^n)$
5. $f_5(n) = n \log n = O(n \log n)$
6. $f_6(n) = n \cancel{*} \sqrt[3]{n^2} = n . n^{2/3} = n^{5/3} = O(n^{5/3})$

(b)

The sorted order will be,

$f_1(n) < f_5(n) < f_6(n) < f_2(n) \leq f_3(n) < f_4(n)$

Ans. to the question no- 07

We can use binary search to do the following solve. The pseudocode is given below:

```
def solve (arr):
    L := 0
    R := len(arr)
    while L < R do
        mid = floor((L+R)/2)
        if arr[mid] > mid Then
            R := mid
        else
            L := mid + 1
    return L
```

Ans. to the que. no-08

We can use quick-sort algorithm to solve the above list. The code is given below:

```
def partition (arr, lo, hi):
    pivot = arr[lo]
    i = lo+1
    for i in range (lo+1, hi+1):
        if arr[j] > pivot:
            arr[j], arr[i] = arr[i], arr[j]
            i+=1
    arr[lo], arr[i-1] = arr[i-1], arr[lo]
    return i-1

def quick_sort (arr, lo, hi):
    if lo < hi:
        pivot_index = partition (arr, lo, hi)
        quick_sort ( arr, lo, pivot_index-1)
        quick_sort (arr, pivot_index+1, hi)
    return arr
```