AFFAN ABDUL MOYEED WIN - 08560727

CS 4310 - Algorithms – Spring 2022

HomeWork1

1. (20pts) For the code snippets below, find time and space complexities. First derive the time complexity using summation notation, then find the close form and finally express it using tight asymptotic (big-Oh, big-Theta, big-Omega) notation. Assume constant $c_1>0$ overhead per iteration for a loop. **Don't bury / hide lower order terms or all constants** into one until the last step of expressing your solution using tight asymptotic notation. First, list all the basic instructions you are counting and then do the frequency counting.

```
a. acc=10;                          ——— 1
for(i=5; i <=n/2; i++)              ——— 3

    for(j=2*n; j >=1; j--)          ——— 4
    acc -= j + acc++;               ——— 3
```

Solution –

⟹ By taking space complexity

→ acc — 1
i — 1
j — 1
n — 1
4

⟹ So the constant proves that space complexity is O(1)

=> By evaluating time complexity with summation notation we get:-

$$\Rightarrow \sum_{i=5}^{n/2} \sum_{j=1}^{2n} 1 \quad \text{xxx}$$

$$\Rightarrow \sum_{i=5}^{n/2} (2n-1+1)$$

$$= \sum_{i=5}^{n/2} (2n)$$

$$= \frac{n}{2} \times 2n - 5$$

$$= \boxed{n^2 - 5}$$

∴ We can say that the big-oh of the code snippets is $O(n^2)$

We can also figure out that :-

$$\longrightarrow \boxed{f(n) = n^2 - 5}$$

1) from the Big-Oh equation.

$$f(n) \leq c \cdot n^4$$

taking $n=1$ & $c=1$

$$n^2 - 5 \leq c \cdot n^4$$

$$1 - 5 \leq 1 \cdot (1)^4$$

$$-4 \leq 1$$

2) from the Big Omega equation.

$$f(n) \geq g(n)$$

taking $n=1$ & $c=1$

$$n^2 - 5 \geq g(n)$$

$$1^2 - 5 \geq c \cdot (n)$$

$$-4 \geq 1$$

3) from θ Big theta equation

$$c_1 (f_1(n)) \leq f(n) \leq c_2 (g(n^2))$$

taking $n=4$ & $c=1$

$$1 \times 4 \leq (4)^2 - 5 \leq 1 (4)^2$$

$$4 \leq 16 - 5 \leq 16$$

$$4 \leq 11 \leq 16.$$

∴ These proved by the asymptotic (O, Θ, Ω) in the tight bond.

b. for(i=2; i<=n; i+=2); — 3

Solution →

→ By taking Space Complexity

→ $i - 1$

$n - 1$

$\frac{}{2}$

→ Solving this the constant proves as O(1)

→ By evaluating time complexity with summation notation we get

$$\sum_{i=2}^{n} 1 = n - 2 + 1 = n - 1 = n$$

→ we can say by evaluating that big-oh of the code snippet is O(n)

1) from the Big-Oh equation.

$f(n) \leq c\, g(n^3)$ taking $n=1$, $c=1$

$n - 1 \leq 1\ (1)^3$

$0 \leq 1$

2) from the Big Omega equation

$f(n) \geq g(\sqrt{n})$ taking $n=3$, $c=1$

$n - 1 \geq 1 \times \sqrt{n}$

$3 - 1 \geq 1 \times \sqrt{3}$

$2 \geq 1.73$

n!.

3) from Θ Big theta equation.

$c_1 (f_1(\sqrt{n})) \leq f(n) \leq c_2(g(n^2))$

$1 \times n^2 \leq 4n-1 \leq 3 \times 16$

$2 \leq 3 \leq 48$

Asuming
$c_1 = 1$
$c_2 = 3$
$n = 4$

∴ These proved by the assymptotic (O, Θ, Ω).
in a tight bond.

c. for(i=1;i<=n;i*=3) ——— 3

x = y^2; ——— 1

Solution →

→ By taking Space Complexity

→ i – 1
n – 1
x – 1
y – 1
4

→ Solving this the constant we get is O(1)

→ By evaluating time complexity with summation notation we get.

$$\Rightarrow \sum_{i=1}^{\log n} (1)$$

$$\Rightarrow \log n - 1 + 1$$

$$\Rightarrow \log n$$

→ we say that by evaluating that big-oh of the code snippet we get O(log n)

1) From Big Oh Equation.

$$f(n) \leq c \cdot g(n^2)$$

taking n = 2, c = 1

$$\log n \leq c \cdot g(n^2)$$

$$\log 2 \leq 1 \cdot 4$$

$$1 \leq 4$$

2. From the Big Omega equation

$$f(n) \geq c\,g(\log\log n) \qquad c=1,\ n=4$$

$$\log n \geq c\,g(\log\log n$$

$$2 \geq 1 \times 1$$

3) From the Big theta equation.

$$c_1(f_1(n)) \leq f(n) \leq c_2(g(n^2))$$

$$\log\log n \leq \log n \leq n^2 \qquad c=1,\ n=4$$

$$1 \times 1 \leq 2 \leq 16$$

∴ These proved by the asymptotic (O, Θ, Ω) in a tight bond.

```
d. y=0;  ——— 1
for (i=0; i<n; i++){  ——— 3
y -= 2;  ——— 1
if (n%3 == 0)  ——— 1
for (j=0; j<=2m; j++) x++;  ——— 4
else
for (j=0; j>m; j++) x--;  ——— 4
}
```

Solution →

→ By ~~time~~ time taking Space Complexity

y — 1 i — 1 n — 1 j — 1 m — 1 x — 1 = 6

→ Solving this the constant we get O(1)

→ By evaluating time complexity with summation notation we get.

$$\sum_{i=0}^{n} \sum_{j=0}^{2m} 1 \Rightarrow \sum_{i=0}^{n} (2m - 0 + 1) \Rightarrow \sum_{i=0}^{n} (2m+1)$$

$$= 2 \sum_{i=0}^{n} m + \sum_{i=0}^{n} 1$$

$$= 2(n \times m) + n$$

$$= 2n + 2m + n$$

$$= n \times m \quad \therefore \text{Constants are ignored}$$

we say that by evaluating that big-oh of the code snippet is $O(n^2)$

1) From Big oh equation :-

$c = 1$
$n = 2$

$$f(n^2) \leq c \cdot g(n^3)$$

$$n^2 \leq 1 \times (2)^3$$

$$4 \leq 8$$

2) from Big Omega equation :-

$c = 1$
$n = 2$

$$f(n^2) \geq c \cdot g(\sqrt{n})$$

$$(2)^2 \geq 1 \times \sqrt{2}$$

$$4 \geq 1.414$$

3) From Big theta equation :-

$c_1 = 1$
$c_2 = 1$
$n = 2$

$$c_1(f_1(\sqrt{n}) \leq f(n^2) \leq c_2 * g(n^4)$$

$$1 \times \sqrt{2} \leq 2^2 \leq 1 \times 2^4$$

$$\sqrt{2} \leq 2^2 \leq 16$$

∴ These proved by the asymptotic (O, Θ, Ω) in a tight bond

```
e. for(i=1; i<n; i++) {          ——— (3)
for(j=1; j<=i; j++) {            ——— (3)
for(k=n; k>j; k--)               ——— (3)
x=x+5; y /= 3;                   ——— (3)
}
x = x+y-i; }                     ——— (3)
```

Solution →

→ By taking Space Complexity

$i-1 \quad n-1 \quad j-1 \quad k-1 \quad x-1 \quad y-1 = 6$

Solving this the constant we get $O(1)$

By evaluating time complexity with summation notation we get

$$\rightarrow \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=j}^{n} 1$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{i} \sum_{k=1}^{n} 1 \qquad \because \text{given } j=1, \therefore k=1$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{i} (n-1+1)$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{i} n$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{1} n \qquad \because \text{given } i=1$$

$$= \sum_{i=1}^{n} (1 - 1 + n)$$

$$= \sum_{i=1}^{n} n$$

$$= (n \times n) - 1$$

$$= n^2 - 1$$

$$= n^2$$

$\therefore$ Constants are ignored

→ we say that by evaluating that big-oh of the code snippet is $O(n^2)$

1) From Big oh equation :- $c = 1$, $n = 2$

$$f(n^2) \leq c \cdot g(n^4)$$

$$n^2 \leq c \cdot n^4$$

$$(2)^2 \leq 1 \times 2^4$$

$$4 \leq 16$$

2) From Big Omega → $c = 1$, $n = 2$

$$f(n^2) \geq c \cdot g(\sqrt{n})$$

$$(2)^2 \geq 1 \times \sqrt{2}$$

$$4 \geq 1.414$$

3) From Big theta equation :-

$$c_1 (f_1(\sqrt{n})) \leq f(n^2) \leq c_2 \, g(n^4)$$

$$1 \times \sqrt{2} \leq 2^2 \leq 1 \times 2^4$$

$$\sqrt{2} \leq 2^2 \leq 16.$$

$\therefore$ These proved by the asymptotic (O, Θ, Ω) in a tight bond.

2. (20pts) This question concerns the asymptotic relations between functions; you can assume that all logarithmic functions are in base 2. It's a repeat from general comments above, but **you must justify your answers**, otherwise no credit. Sort the following functions in an asymptotically non-decreasing order of growth using big-oh and big- theta notations; in other words, provide a ranking of these functions such that g1 = O(g2), g2 = O(g3), g3 = Q(g4), g4 = O(g5), . . .

(sqrt(4))^logn, 2^(sqrt(n)), (100)*n log(2n), n^2, (n * 2^(n/2)), (2/n)*n!, 8^logn, log(n^1500000), (log n)^2, 2^100000, 2^(4^n)

→ Plotting all the functions on the graph for better understanding we get the ranking of the functions as a non-decreasing sequence.

→ The sequence is in a g1 > g2 where g1 to g2. g1 elements grow at a high pace compared to g2 where there are depreciation noted while solving the g2 order of growth.

→ The sequence is as follows:-

$$\frac{2}{n}n! > 100n\log(2n) > 2^{4^n} > 8^{\log n} > n\,2^{n/2} > n^2 > 2^{\sqrt{n}} > \sqrt{4}^{\log n} > \log n^{1500000} > (\log_2 n)^2 > 2^{100000}$$

→ Taking 2 elements from the sequence and proving them where $g1 > g2 \ldots$.

a)

$g1) \sqrt{4}^{\log n}$

where we can also write this as

$n^{\log \sqrt{4}} = n.$

$\therefore O(n)$

$g2) (\log n)^2$

where we can see that the graph of $\log n$ has more depreciating value compared to n.

$\therefore$ $\boxed{g1 > g2}$

b)

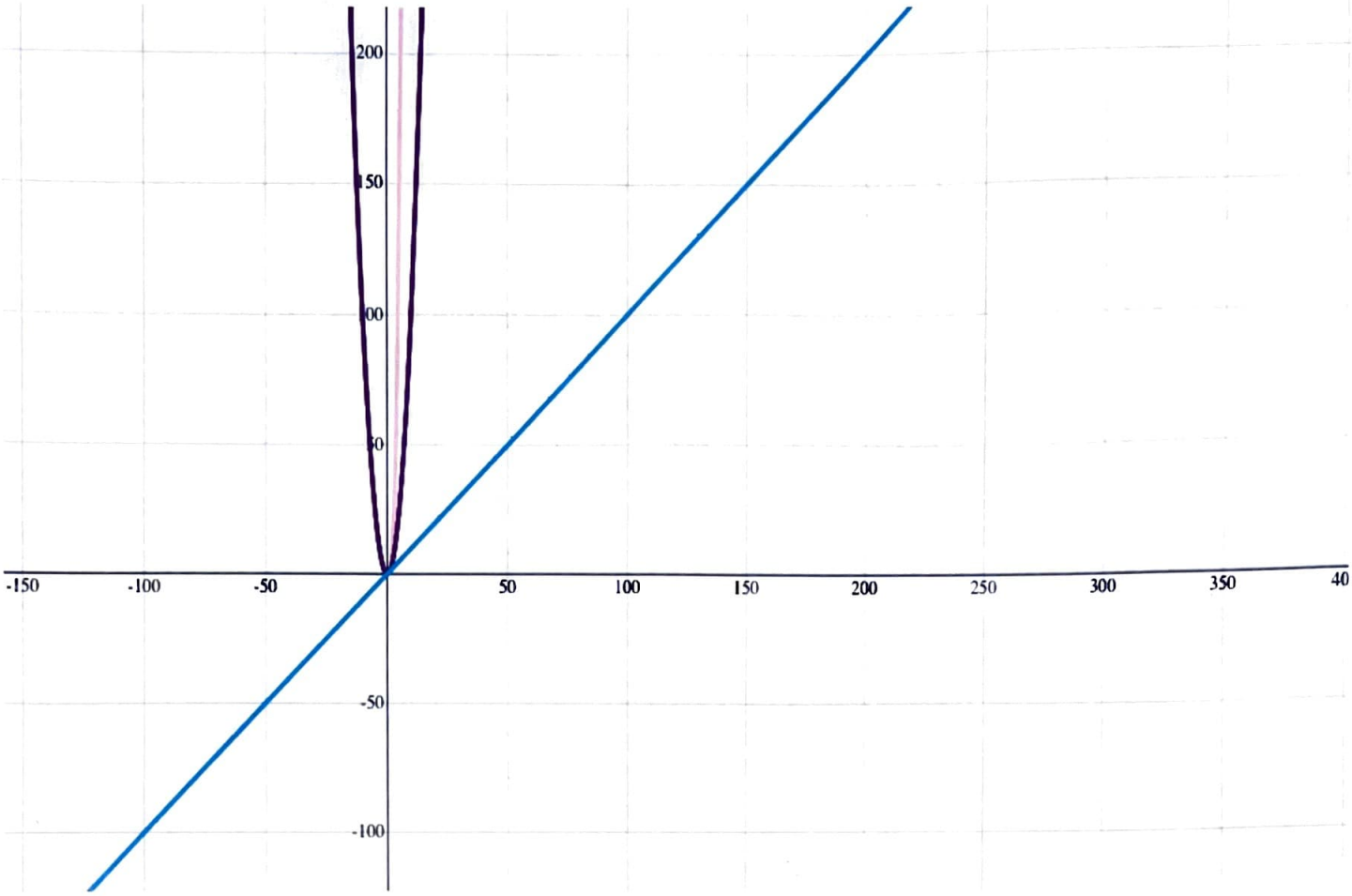
$g1) \left(\frac{2}{n}\right) n!$

where $n!$ is also $n(n-1)$ which may also have an infinite value.

$g2) 2^{100000}$

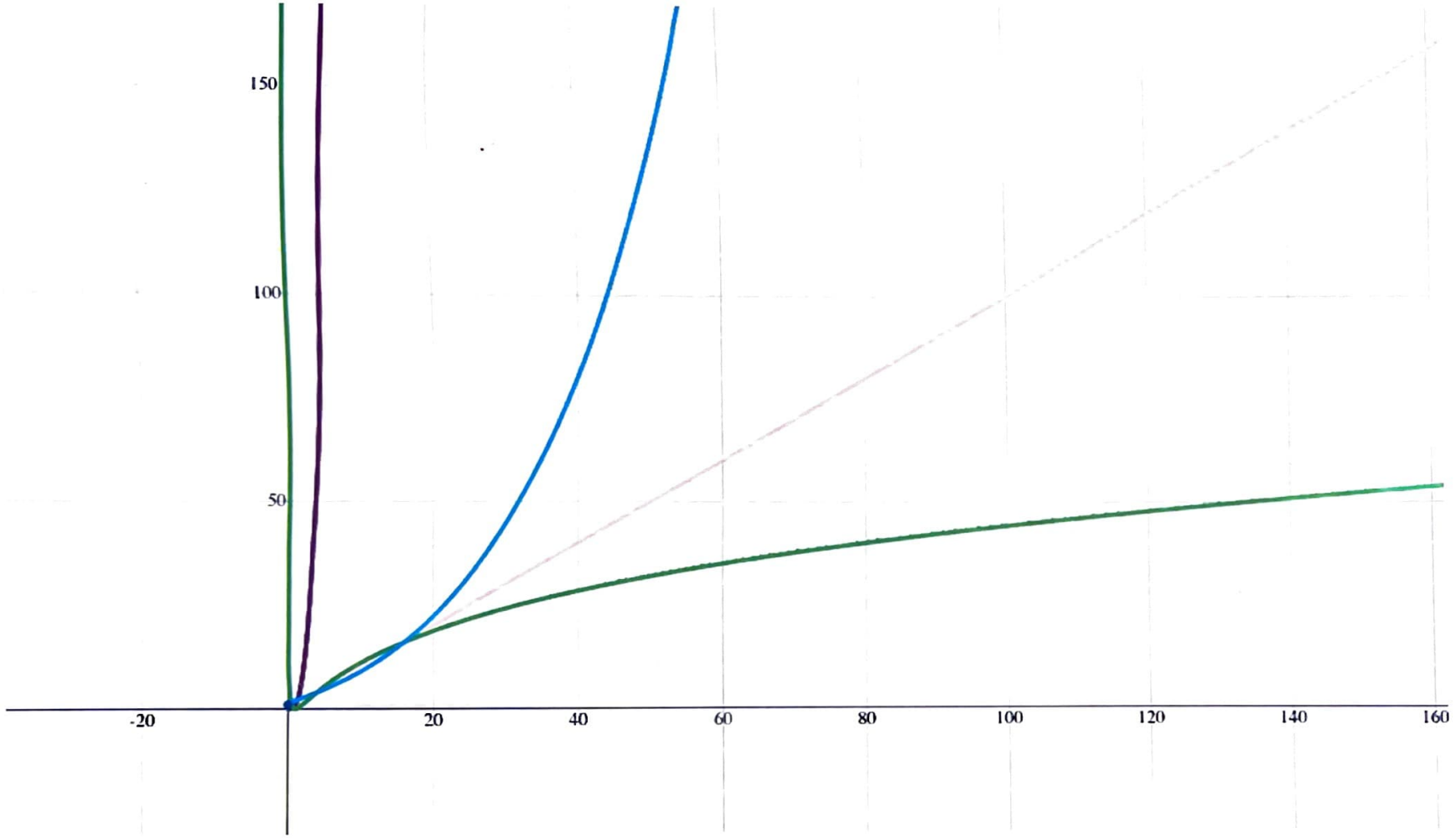
we see a constant value has no up growth on a graph. So we can conclude that $\boxed{g1 > g2}$

12.1



- $f(n) = 8^{\log_2(n)}$
- $f(n) = n^2$
- $f(n) = 100(n)\log_2(2n)$
- $f(n) = (n)2^{\frac{n}{2}}$
- $f(n) = 2^{4^n}$

12.2



- $f(n) = \sqrt{4}^{\log_2(n)}$
- $f(n) = 2^{100000}$
- $f(n) = \left(\log_2(n)\right)^2$
- $f(n) = 2^{\sqrt{n}}$
- $f(n) = \log_2(n)^{150000}$

3. (20pts) Suppose you are given a new hardware device that can merge d > 2 different sorted lists of total size n into a single sorted list in O(n) time, independent of the value of d. Such a device could, for example, be based on a hardware streaming system or could be based on a network protocol. Show that you can use this device to sort items efficiently by answering the following. (3.a) Suppose that you are given d sorted lists where list *i* has length n_i with $m = \sum n_i$. Design and analyze an efficient algorithm to sort the m items using the hardware device, i.e., your algorithm uses the hardware device as a method call mergeDSortedLists(L[][], d, m) to sort the m items, where L is a d X m two-dimensional array containing a total of m items and L[i] contains a sorted list of length n_i. Function mergeDSortedLists(...) returns a 1-dimensional sorted array of length m. What is the input size of this problem? What is the output-size of this problem? What are the time and space complexities of your algorithm? (3.b) Suppose now that you are given a list of n items (unordered). Show that you can use the hardware device to sort n elements in O(n log n / log d) time. Notice that, if d is $\Theta(\sqrt{n})$, then you can use this device to sort in linear time. Obviously, your algorithm will use method calls mergeDSortedLists(...) judiciously: What is the input size of this problem? What is the output-size of this problem? What are the time and space complexities of your algorithm?

3.a]

Solution — Algorithm to sort the m items in a d sorted lists.

→ function mergeDSortedLists (x, y, z)
let a, b, c be size of x, y, and z
let d be the stored sorted lists
while g < a and h < b and i < c
Get minimum of x[g], y[h], z[i]
if the minimum is from x, add it to d and proceed g
else if the minimum is from y add it to d and proceed h
else if the minimum is from z add it to d and proceed i

while g<a and h<b
    put minimum of x[g] and y[h] into d
    Proceed g if minimum is from x else proceed h
while g<a and i<c
    Put minimum of x[g] and z[i] into d
    Proceed h if minimum is from y else proceed i
if x and y have exhausted pull components from z
if y and z have exhausted pull components from x
if x and y have exhausted pull components from y
return d.

→ Given d > 2 sorted lists
Total size n
O(n) times
i = length ni
m = $\sum$ ni

→ Input size of the problem can be :-

→ $\sum_{i=0}^{d} n_i = m$

where the list of sorted arrays is d x m.
containing length ni

→ Output size :-
The problem's output is also m, as it returns only one single sorted list of size m from the inputs provided in the given statement.

Time complexity :- So by seeing the function mergeDsortedList we get O(m) items and it performs O(m) times in the algorithm.
so the time complexity is O(n) in a sorted lists

Space Complexity :- So the 2 dimentional array dx m we have takes O(dxm) space in the sorted list as calculating all the variables in the algorithm we get 10 as the space complexity. Thus we get space complexity as O(dm) when we ignore constants.

3.6] Given unordered list of n items, d is $O(\sqrt{n})$

→ To show elements O(nlogn / log d) time.

Algorithm - n number of list which are unordered.

→ I/p - n numbers.

a) divide list by poking in between we get 2 parts

b) Sort every element using recursive property

c) Arrange them in a sequence and merge all the chunks into one element

d) Returning the merged element.

→ So from given time taken to evaluate a merged list we get a $T(n)$

→ By Using Master's theorem,

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

where n – size of input

a = no. of subproblems in a recursive function

n/b – size of each subproblem

$f(n)$ – the recursive call done before and after merging the solutions

→ from the given we get

$$T(n) = O(n \log_d n)$$

$$= O\left(\frac{n \log_d n}{\log d}\right)$$

where $d = \Theta(\sqrt{n})$

$$\therefore T(n) = O\left(\frac{n \log n}{0.5 \log n}\right) = \frac{n}{0.5} \quad \text{(Removing commons)}$$

$$= 2n = O(n)$$

→ Space Complexity :- The space complexity according to the algorithm explained is O(n) as we have n elements in the list (unsorted).

→ Time Complexity :- This should be O(n log n) as the elements in the unordered list are getting divided into 2 halves and it takes linear time to merge the list chunks in A[i] list.

4. (20pts) **(4.a)** Algorithm A uses 2n log n + 5n operations, while algorithm B uses n^2 operations. Determine the value n_0 such that A is better than B for $n \geq n_0$. **(4.b)** Are there values of n such that B is better than A? If yes, list the ranges. If not, justify why not. **(4.c)** Determine the value n_0 such that A is better than B for $n \geq n_0$, when B uses n^3 operations.

4) Given - $2n \log n + 5n$ as A

n^2 as B

4.a] By trial and error method we can prove that while substituting $n = 1-12$ we get a result of $A > B$ and can also state that A is better than B.

n = 10

we get $A > B$

$116.43 > 100$

n = 11

we get $A > B$

$131 > 121$

n = 12

we get $A > B$

$146 > 144$

n = 13

we get $A \not> B$ X

$161 < 169$.

Hence in the range 1-12. A is better than B.

So. $n_0 = 0$

where $n \geq n_0$

$1 \geq 0$ values $[n = 1-12]$

4.6.) Given $A = 2n \log n + 5n$

$B = n^2$

→ If we substitute values from 13, 14, 15.... we see that B is better than A

$n = 13$ $A < B$

$161 < 169$

$n = 14$ $A < B$

$176.6 < 196$

$n = 15$ $A < B$

$192.20 < 225$

∴ we can prove that the values of n are such that B is better than A.

4.c) Given $A = 2n \log n + 5n$ (20)

$B = n^3$

→ By evaluating values $n = 0, 1, 2, 3 \ldots$ we conclude that $n = 1, 2$ accepts A is better than B. where we get $n_0 = 0$ as for the $n \geq n_0$.

$n = 0$

$A = B$

$0 = 0$

$n = 1$

$A > B$

$5 > 1$

$n = 2$

$A > B$

$14 > 8$

$n = 3$

$A < B$

$24.5 < 27$ ×

∴ Hence by taking the values we have proved our point that A is better than B for only 1,2 in n values.

5. (20pts) Solve the following recurrence relation T(n) = 2*T(n/2) + 4*n^2. if n>=2,

   = 8 otherwise.

First find a closed form using expansion (aka substitution) method and then express your solution using tight asymptotic notation. These kind of recurrence relations arise in solutions when manipulating matrices.

Solution –

→ Given :

$$T(n) = \begin{cases} 8 & \text{if } n \le 1 \\ 2T(n/2) + 4n^2 & \text{if } n \ge 2 \end{cases}$$

→ Solving by recussence relation –

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

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

$S_1 = 2^2 T\left(\frac{n}{2^2}\right) + 2n^2 + 4n^2$

$= 2^2\left[2T\left(\frac{n}{2^3}\right) + 4\left(\frac{n}{2^2}\right)^2\right] + 2n^2 + 4n^2$

$S_2 = 2^3 T\left(n/2^3\right) + n^2 + 2n^2 + 4n^2$

$= 2^3\left[2T\left(n/2^4\right) + 4\left(\frac{n}{2^3}\right)^2\right] + n^2 + 2n^2 + 4n^2$

$S_3 = 2^4 T\left(n/2^4\right) + \frac{n^2}{2} + n^2 + 2n^2 + 4n^2$

→ From S_1, S_2, S_3 we get the initial Recursive equation as.

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

→ we have a stopping condition here So taking $T(n)$ we see that

$$\Rightarrow \frac{n}{2^k} = T(1)$$

$$\Rightarrow n = 2^k$$

$T(1) = 8$ → we know

$\log_2 n = k$ [from $\log_b a = c$ $b^c = a$]

→ Substituting these values in $T(n)$

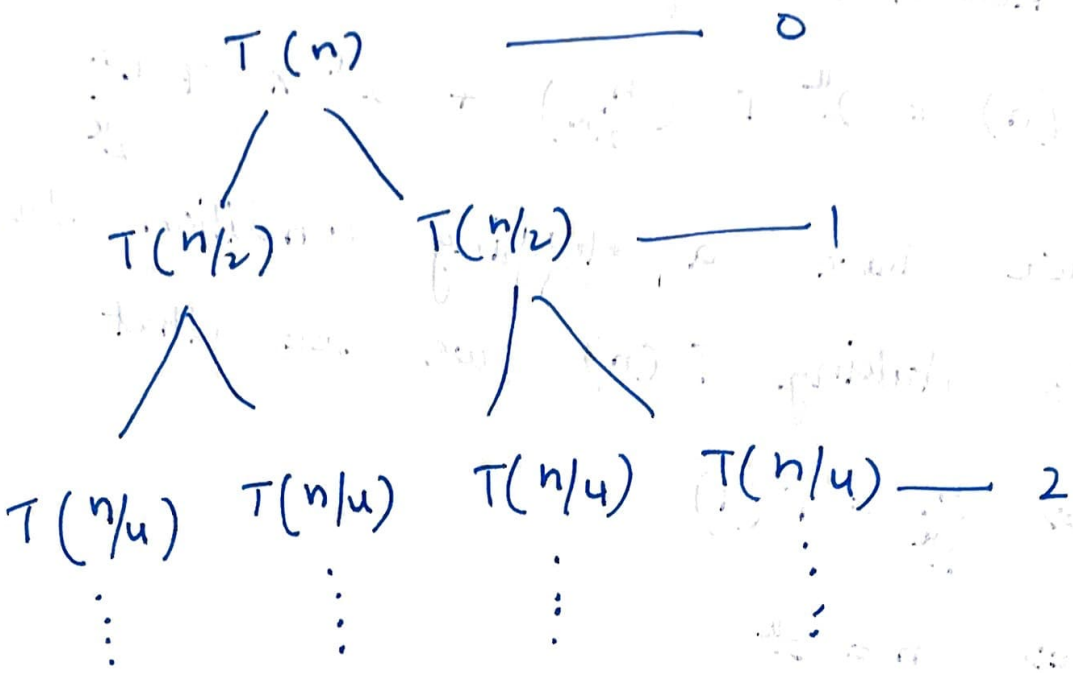
$$2^{\log_2 n} T(1) + 2^{\log_2 n} \times 4\frac{n^2}{2^{\log_2 n}}$$

where $2^{\log_2 n} = n^{\log 2} = n$

→ $8n + 4n^2$ (ignoring lower bond & constants)

∴ The asymptotic notation is $O(n^2)$

→ Recurrence Relation tree (For understanding)



(to clarify doubts)

Discussed the solutions with

Yashoda Varma

Sai Jyothi

PLAGIARISM DECLARATION

1. I know that plagiarism means taking and using the ideas, writings, programs, code, works, or inventions of another as if they were one's own. I know that plagiarism not only includes verbatim copying, but also the extensive use of another person's ideas without proper acknowledgement (which includes the proper use of quotation marks). I know that plagiarism covers this sort of use of material found in textual sources and from the Internet.

2. I acknowledge and understand that plagiarism is wrong.

3. This assignment is my own work, or my group's own unique group assignment. I acknowledge that copying someone else's assignment, or part of it, is wrong, and that submitting identical work to others constitutes a form of plagiarism.

4. I have not allowed, nor will I in the future allow, anyone to copy my work with the intention of passing it off as their own work.

Name AFFAN MOYEED Student # 08560172 7

Signed  Date 02/18/22