Grade: 4.5/9

Exercise 1 (for grade) ~ Monday, August 29, 2022 ~ CPSC 535 Fall 2022

Write one submission for your entire group, and write all group members' names on that submission. Turn in your submission before the end of class. The X symbol marks where you should write answers.

Recall that our recommended problem-solving process is:

1. **Understand** the problem definition. What is the input? What is the output?
2. **Baseline** algorithm for comparison
3. **Goal** setting: improve on the baseline how?
4. **Design** a more sophisticated algorithm
5. **Inspiration** (if necessary) from patterns, bottleneck in the baseline algorithm, other algorithms
6. **Analyze** your solution; goal met? Trade-offs?

Follow this process for each of the following computational problems. For each problem, your submission should include:

a. State are the input variables and what are the output variables
b. Pseudocode for your baseline algorithm, that needs to include the data type and an explanation for any variable other than input and output variables
a. The Θ-notation time complexity of your baseline algorithm, with justification.

and if you manage to create an improved algorithm:

c. Answer the question: how is your improved algorithm different from your baseline; what did you change to make it faster?
d. Pseudocode for your improved algorithm, that needs to include the data type and an explanation for any variable other than input and output variables
a. The Θ-notation time complexity of your improved algorithm, with justification.

Today's problems are:

1. *Indicate the class Θ*

For each of the following functions, indicate the class Θ(g(n)) the function belongs to. Apply all possible rules to simply g(n) as much as possible. Prove your assertions either using definitions or the limit theorem.

a. $\sqrt{4n^2 - 8b + 10}$
b. $2^{n/2-1} + 3^{n/3+1}$

2. *Parallel matrix update of size 3x3*

Given a matrix A of size 3x3, each cell (i.e. an element of the matrix) will be updated based on the values of (existing) immediate neighboring cells. This update must be done in parallel, this means A[0,0] will be updated in parallel with A[0,1], etc.. For a cell A[i,j], the update will be of the form:
A[i][j] = A[i-1][j] + A[i+1][j] + A[i][j-1] + A[i][j+1] - 4*A[i][j]
Some of the terms may be missing if the element is on the border of the matrix: i=0, j=0, i=2, j=2.
For example A[0][0] = A[1][0] + A[0][1] - 4*A[0][0].

In other terms, if $A^{(0)}$ is the initial matrix at time 0, then $A^{(1)}$ will be computed entirely based on the cells of $A^{(0)}$ at time 1, $A^{(2)}$ will be computed entirely based on the cells of $A^{(1)}$ at time 2, and so on.
To compute $A^{(1)}[0][0]$ one needs 3 operations (one addition, one subtraction, and one multiplication).
You need to state the number of simple mathematical operations needed to compute all the cells at time 1.
This value is obtained by adding the number of simple mathematical operations for each cell in the matrix.

3. *Parallel matrix update of size 3x3*

Given a matrix A of size *n* x *n*, each cell (i.e. an element of the matrix) will be updated based on the values of (existing) immediate neighboring cells. This update must be done in parallel, this means A[0,0] will be updated in parallel with A[0,1], etc.. For a cell A[i,j], the update will be of the form:
A[i][j] = A[i-1][j] + A[i+1][j] + A[i][j-1] + A[i][j+1] - 4*A[i][j]
Some of the terms may be missing if the element is on the border of the matrix: i=0, j=0, i=n-1, j=n-1.
For example A[0][0] = A[1][0] + A[0][1] - 4*A[0][0].

In other terms, if $A^{(0)}$ is the initial matrix at time 0, then $A^{(1)}$ will be computed entirely based on the cells of $A^{(0)}$ at time 1, $A^{(2)}$ will be computed entirely based on the cells of $A^{(1)}$ at time 2, and so on.
To compute $A^{(1)}[0][0]$ one needs 3 operations (one addition, one subtraction, and one multiplication).
You need to state the number of simple mathematical operations needed to compute all the cells at time 1.
This value is obtained by adding the number of simple mathematical operations for each cell in the matrix.

Names

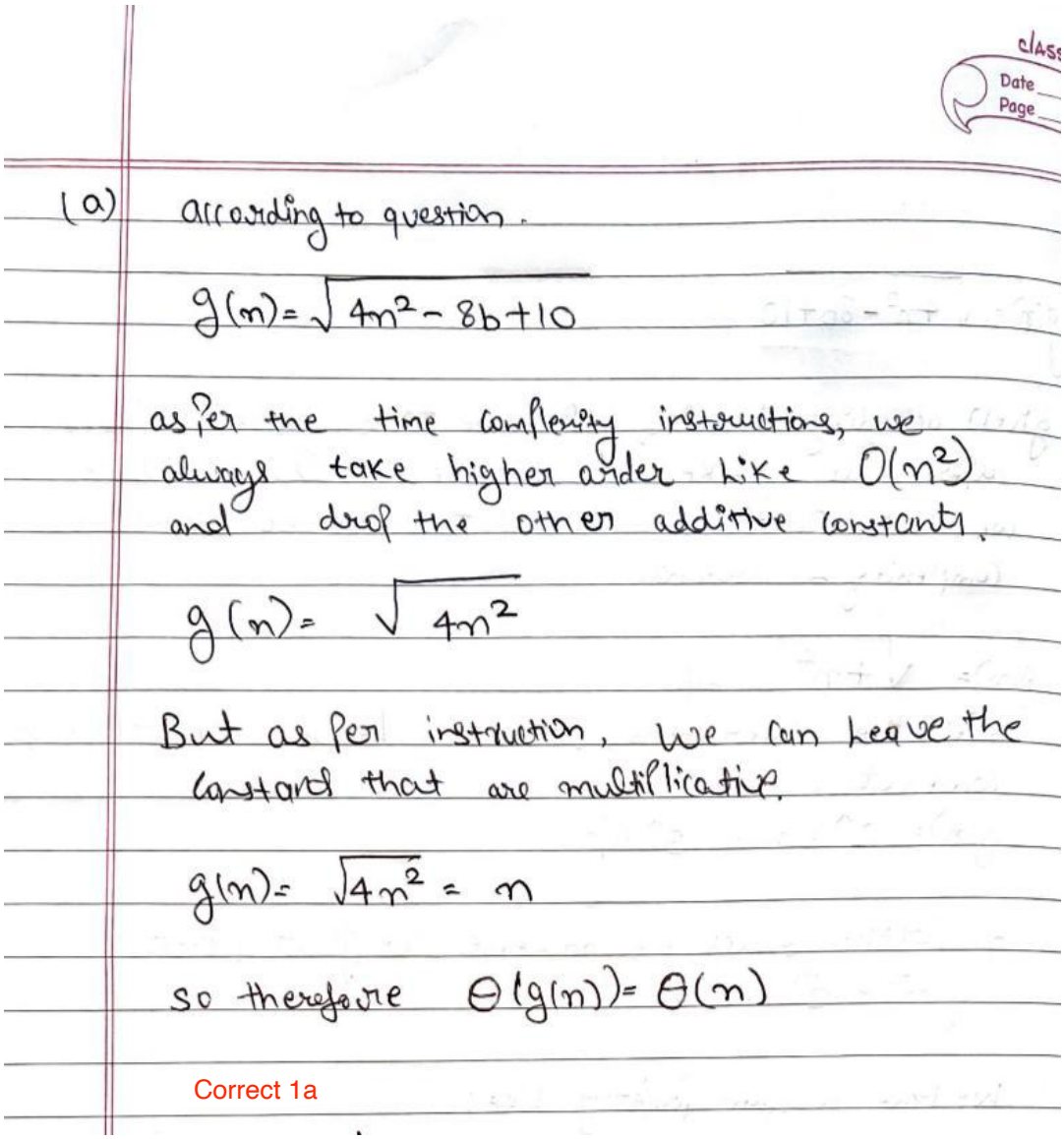
Write the names of all group members below.

X Sayali Ghorpade

Saurabh Jain

Rosa Cho

Exercise 1: Solve and provide answer



(a) according to question.

$g(n) = \sqrt{4n^2 - 8b + 10}$

as per the time complexity instructions, we always take higher order like $O(n^2)$ and drop the other additive constants.

$g(n) = \sqrt{4n^2}$

But as per instruction, we can leave the constant that are multiplicative.

$g(n) = \sqrt{4n^2} = n$

so therefore $\Theta(g(n)) = \Theta(n)$

Correct 1a

X

(b) according to question.

$$g(n) = 2^{n/2-1} + 3^{n/3+1}$$

So as according to the complexity rules, we can leave the constant like additive constants,

$$g(n) = 2^{n/2} + 3^{n/2}$$

But according to the protocole, we can drop Floor derators

$g(n) = 2^n + 3^n$

and as according to instruction property

$g(n) = a(n) + b(n)$

$= \max(a(n), b(n)$ - (1)

Here I am using property - (1)

So, $\theta(g(n)) = \theta(3^n)$

(-0.5 points): Incorrect 1b

Exercise 2: Solve and provide answer

X

Given

```
Procedure MatrixUpdate

Begin
Step 0:

Set A[0][0] as

   A[(i),(j)] = min A[(i),(j)]

Step 1:
   Add A[i-1] and A[i+1][j] and A[i][j-1] and A[i][j+1]
   Set as L(m=1) = A[(i),(j)]
   Set number of mathematical operations to with 3 for Step 1, 1
for step 2, 1 for step 3
Step 2:
   multiply 4 to A[i][j]
    L(m=2) = 4*A[(i),(j)]
Step 3:
         subtract L(m=1) from L(m=2)

Step 4: return new A[i][j]

End
```

(-2 points): incorrect exercise 2

Exercise 3: Solve and provide answer

X

Q-3

As per the given in the question.

Let if i=0, j=0, i=n-1 or j=n-1

the according to give in the question.

A[i][j]= A[i-1][j] + A[i+1][j] + A[i][j-1] +
A[j][j+1] - 4* A[i][j].

But maybe we get error if array out of bound reach.

If i=1 and j=1 then equation

A[1][1]= A[0][1] + A[2][1] + A[1]+[0]+
A[1][2] - 4* A[1][1].

If i=2, j= 3
then
A[2][3]= A[1][3]+ A[3]

Pseudocode.

```
Int count=0
  Int A[0][0]=0,    A[n-1][n-1]
                 ∴ we have to start loop
                   should 1 because
                   -ve value throws error.

for (int i=1; i<n-1; i++)
      for (int j=1; j<n-1; j++)

          A[i][j]= A[i-1][j] + A[i+1][j]
            + A[i][j-1] + A[i][j+1] → 4[A][i][j]
            ++ count;
          }
      }.
```

and as per the above iteration.

Time complexity (Θ) = $O(n^2)$
because of 2 nested loop.

If input $n=5$
it take 16 iterations

(-2 points): incorrect exercise 3