Name:

ID:

CSCI 3104, Algorithms **Escobedo & Jahagirdar**

Homework 2A (45 points) **Summer 2020, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solution:

- The solutions **should be typed**, we cannot accept hand-written solutions. Here's a short intro to **Latex**.
- In this homework we denote the asymptomatic *Big-O* notation by $\mathcal{O}$ and *Small-O* notation is represented as o.
- We recommend using online Latex editor **Overleaf**. Download the **.tex** file from Canvas and upload it on overleaf to edit.
- You should submit your work through **Gradescope** only.
- If you don't have an account on it, sign up for one using your CU email. You should have gotten an email to sign up. If your name based CU email doesn't work, try the identikey@colorado.edu version.
- Gradescope will only accept **.pdf** files (except for code files that should be submitted separately on Canvas if a problem set has them) and **try to fit your work in the box provided**.
- You cannot submit a pdf which has less pages than what we provided you as Gradescope won't allow it.

Name:

ID:

CSCI 3104, Algorithms **Escobedo & Jahagirdar**

Homework 2A (45 points) **Summer 2020, CU-Boulder**

Piazza threads for hints and further discussion

Piazza Threads
Question 1
Question 2
Question 3
Question 4
Question 5

Recommended reading: Divide & Conquer; Recurrence Relations: Ch. 2 2.3; Ch. 4 4.1, 4.2, 4.3, 4.4, 4.5. The three methods for solving recurrences from **4.3 - 4.5** is important.

Name:

ID:

CSCI 3104, Algorithms **Escobedo & Jahagirdar**
Homework 2A (45 points) **Summer 2020, CU-Boulder**

1. *(5 × 4 = 20 pts) Solve the following recurrence relations. For each case, show your work.*

   (a) $T(n) = 3T(n-1) + 1$ *if* $n > 1$ *and* $T(1) = \Theta(1)$

$T(n) = 3T(n-1) + 1$
$= 3(3T(n-2) + 1) + 1 = 9T(n-2) + 3 + 1$
$= 3(9T(n-3) + 2 + 1) + 1 = 27T(n-3) + 6 + 3 + 1$
.
.
.
$= 2^{n-1}T(1) + 3^{n-2} + ... + 3^2 + 1$

$\sum_{i=0}^{n} 3^i$

$= \Theta(3^n)$

(b) $T(n) = 2T(\frac{n}{3}) + \Theta(n)$ *if* $n > 1$ *and* $T(1) = \Theta(1)$

$T(n) = 2T(\frac{n}{3}) + \Theta(n)$
Using Master Method
$a = 2,\ b = 3,\ c = 1$

We use the case 3 of the Masters Theorem. 3rd case states $\Theta(n^c)$ if $a < b^c$

Plugging in the respective values we get:
$a < b^c \leftrightarrow \log_b a < \log_b b^c = c$

$T(n) = n^c \sum_{k=0}^{\log_b n-1} (\frac{a}{b^c})^k + n^{\log_b a}$

$= n^c \cdot \Theta(1) + n^{\log_b a}$
$= \Theta(n)$

(c) $T(n) = 3T(\frac{n}{2}) + \Theta(n)$ *if* $n > 1$ *and* $T(1) = \Theta(1)$

$T(n) = 3T(\frac{n}{2}) + \Theta(n)$ is best solved using the masters method.
$a = 3$, $b = 2$, $c = 1$

plugging in our values to the third case of the masters theorem $(a > b^c)$we see that the inqualities hold true.
$3 > 2^1$

Therefore by the master's theorem it states $T(n) = n^c \cdot (\frac{a^{\log_b n}}{n^c}) + n^{\log_b a}$
$= n^c \cdot (n^{\log_2 3}) + n^{\log_2 3}$
$= \Theta(n^{\log_2 3})$

(d) $T(n) = 4T(\frac{n}{2}) + \Theta(n^2)$ *if* $n > 1$ *and* $T(1) = \Theta(1)$

$T(n) = 4T(\frac{n}{2}) + \Theta(n^2)$ is best solved using the masters theorem.

$a = 4$, $b = 2$, $c = 2$
Plugging in our values into the second case of the masters method ($a = b^c$) we can se that it holds true.
$4 = 2^2$

Therefore by the masters thereom, the second case states:
$T(n) = \sum_{k=0}^{\log_b n-1} (\frac{a}{b^c})^k + n^{\log_b a}$

$= n^c \cdot \Theta(\log_b n) + n^{\log_b a}$
$= \Theta(n^2 \log_2 n)$

Name:

ID:

CSCI 3104, Algorithms **Escobedo & Jahagirdar**
Homework 2A (45 points) **Summer 2020, CU-Boulder**

2. *(5 pts) Can master's theorem be applied to solve the following recurrence* $T(n) = 8T(\frac{n}{2}) + n^3 \log(n)$. *If* $n > 1$ *and* $T(1) = \Theta(1)$
   Prove that the above recurrence cannot be solved using Master's theorem by showing that none of the three conditions required to apply the theorem are satisfied by the recurrence.

Yes we can apply Master's Theorem to the recurrence however it cannot be used to solve it because none of the three cases are satisfied by the recurrence.
$T(n) = 8T(\frac{n}{2}) + n^3 \log(n)$

$a = 8,\ b = 2,\ f(n) = n^3 \log n,\ n^{\log_b a} = n^{\log_2 8} = \mathcal{O}(n^3)$

Case 01:
$states f(n) = \mathcal{O}(n^{\log_b a + c})$ where c represents some constant.
$n^3 \cdot \log n \neq n^3$ We can see that this expression does not hold and therefore does not satisfy the first case of masters theorem.

Case 02, 03:
$f(n) > n^{\log_b a} = n^3$, although this is a true statement, f(n) is not polynomially larger. $\frac{f(n)}{n^{\log_b a}} = \frac{n^3 \log n}{n^3} = \log n$. Thi ratio is asymptotically less than n^k for any positive constant k. Therefore this recurrrence falls into the gap of Case 02 and Case 03.

Name:

ID:

CSCI 3104, Algorithms — **Escobedo & Jahagirdar**

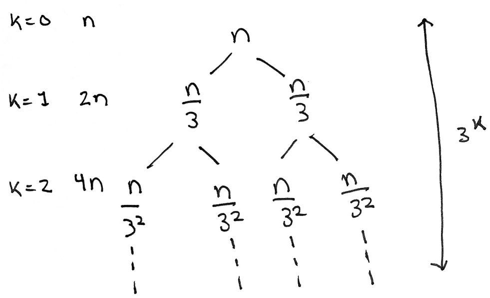
Homework 2A (45 points) — **Summer 2020, CU-Boulder**

3. *(5 × 2 = 10 pts) Consider the following functions. For each of them, determine how many times is hi printed in terms of the input n (i.e in Asymptotic Notation of n). You should first write down a recurrence and then solve it using the* ***recursion tree method****. That means you should write down the first few levels of the recursion tree, specify the pattern, and then solve.*

(a)

```
def fun(n) {
    if (n > 1) {
        print( hi hi hi )
        fun(n/3)
        fun(n/3)
    }
}
```

$T(n) = 2T(\frac{n}{3}) + \Theta(1)$



$n = 3^k \Rightarrow k = \log_3 n$ or more precisely $\log_3 n - 1$

total # of nodes: $2^0 + 2^1 + 2^2 + \ldots + 2^{k-1} \Rightarrow 2^{\log_3 n - 1}$

'hi' is printed $O(2^{\log_3 n - 1})$ or $O(2^{\log_3 n})$

$T(n) = \mathcal{O}(2^{\log_3 n})$

(b)

```
def fun(n) {
    if (n > 1) {
        for i=1 to n {
            print( hi hi )
        }
        fun(n/3)
        fun(n/3)
    }
}
```

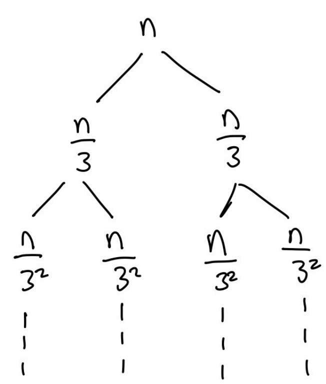
$T(n) = 2T(\frac{n}{3}) + n$

\# times "hi" is printed

$k=0 \quad n$

$k=1 \quad 2n$

$k=2 \quad 4n$



$n \times 2 \times 1$

$\frac{n}{3} \times 2 \times 2$

$\frac{n}{3^2} \times 2 \times 4$

$\left(\frac{n}{3^k}\right)(2)(2^{k-1})$

$\frac{n}{3^k} < 1 \Rightarrow k = \log_3 n - 1$

$\left(\frac{n}{3^{\log_3 n}}\right) \cdot (2)(2^{\log_3 n - 1})$

$= 2n(2^{\log_3 n - 1})$

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

$T(n) = \mathcal{O}(n)$

Name:

ID:

CSCI 3104, Algorithms — **Escobedo & Jahagirdar**

Homework 2A (45 points) — **Summer 2020, CU-Boulder**

4. *(5 pts) Let $T(n) = 3T(\frac{n}{3}) + n$, where $T(n)$ is constant when $n \leq 3$. Using unrolling, determine tight asymptotic bounds for $T(n)$. That is, find a function $g(n)$ such that $T(n) \in \Theta(g(n))$. Clearly show all your work*

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

$$\Downarrow$$

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

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

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

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

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

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

$$T(n) = \cdots$$

$$\cdots$$

$$= 3^k T\left(\frac{n}{3^k}\right) + kn$$

$$\frac{n}{3^k} = 1$$

$\Downarrow$ solve for k

$$k = \log_3 n$$

$$T(n) = 3^{\log_3 n} T(1) + \log_3 n \cdot n$$

ignoring constants

$$\Downarrow$$

$$n + n \cdot \log_3 n$$

$$\boxed{g(n) = \Theta(n \cdot \log n)}$$

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

Name:

ID:

CSCI 3104, Algorithms — **Escobedo & Jahagirdar**

Homework 2A (45 points) — **Summer 2020, CU-Boulder**

5. *(5 pts) Consider the following pseudo-code.*

```
def fun(A[1,2,3....n]) {
    if A.length == 0:
        return 0
    count = 1
    for i=1 to n:
        for j=1 to n:
            count++
    return 1 + fun(A[1,2,3.....n-2])
}
```

Find a recurrence for the worst-case runtime complexity of this algorithm. Then solve your recurrence and get a tight bound on the worst-case runtime complexity.

Evaluating time complexity of each line we get:
$T(n) = c \cdot \mathcal{O}(1) + \mathcal{O}(n^2) + T(n-2)$
disregarding the constants we get:
$T(n) = \mathcal{O}(n^2) + T(n-2)$
solving the recurrence relation we get $T(n) = \mathcal{O}(n^3)$

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

$T(n) = T((n-2)-2) + (n-2)^2 + n^2 = T(n-4) + (n-2)^2 + n^2$

$T(n) = T((n-2)-4) + ((n-2)-2)^2 + (n-2)^2 + n^2 = T(n-6) + (n-4)^2 + (n-2)^2 + n^2$

$\downarrow$

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

$T(n) = T(0) + \frac{n^3}{2}$ → assuming $T(0) = 1$

$T(n) = 1 + \frac{n^3}{2}$

ignoring constants $\boxed{T(n) = O(n^3)}$

6. ***Extra Credit (5% of total homework grade)*** *For this extra credit question, please refer the leetcode link provided below or click here. Multiple solutions exist to this question ranging from brute force to the most optimal one. Points will be provided based on Time and Space Complexities relative to that of the most optimal solution.*

   Please provide your solution with proper comments which carries points as well.

   https://leetcode.com/problems/maximum-subarray/

   ```
   Replace this text with your source code inside of the .tex document
   ```