[redacted]

Q1.1

$Q1$

A

$\mathcal{T}(n) \leq \mathcal{O}(1)\ if\ n \leq a\ else\ \mathcal{T}(n-a) + \mathcal{T}(a) + n$
$\implies \exists c > 0 : \mathcal{T}(n) \leq c\ if\ n \leq a\ else\ \mathcal{T}(n-a) + c + n$

We get that $\forall n > a\ \mathcal{T}(n) \leq \frac{n^2+2\cdot n\cdot c+n\cdot a-a^2}{2\cdot a}$

We will prove this in Induction:

$-Base\ Case:$
$\mathcal{T}(a) = \frac{a^2}{2\cdot a} + \frac{a\cdot c}{a} + \frac{a}{2} - a = \frac{a}{2} + \frac{a}{2} + c - a = c$
As required for base case.

$-Assumption:$
Let's assume that for some $n > a$ that the following assumption holds:
$\mathcal{T}(n) \leq \frac{n^2+2\cdot n\cdot c+n\cdot a-a^2}{2\cdot a}$

$Step:$
$\mathcal{T}(n+a) \leq \mathcal{T}(n) + c + n + a \leq (I.h)\ \frac{n^2+2\cdot n\cdot c+n\cdot a-a^2}{2\cdot a} + c + n + a = \frac{n^2+2\cdot n\cdot c+n\cdot a-a^2+2ac+2na+2a^2}{2\cdot a} = \frac{n^2+2\cdot a\cdot n+n\cdot a+2ac+a^2+2nc}{2\cdot a} =$
$\frac{n^2+2\cdot a\cdot n+a^2+2nc+2ac+na}{2\cdot a} = \frac{(n+a)^2+2c(n+a)+(n+a)a-a^2}{2a}$
$\implies$ Thus Induction Proof holds and we proved the $\mathcal{T}(n)$ form as required.

to summarize:
$\exists c > 0 : \mathcal{T}(n) \leq c\ if\ n \leq a\ else\ \frac{n^2+2\cdot n\cdot c+n\cdot a-a^2}{2\cdot a}$ ■

B
assume that general form is the following:
$\exists c > 0 : \mathcal{T}(n) \leq \sum_{i=1}^{n} \frac{1}{i} + c\ if\ n > 0\ else\ c$

$-Base\ Case:$
$n = 0 : \mathcal{T}(0) = 0 + c = c$

$-Assumption:$
for some $n > 0$ such that $\mathcal{T}(n) \leq \sum_{i=1}^{n} \frac{1}{i} + c$ holds.

$-Step:$

$\mathcal{T}(n+1) \leq \mathcal{T}(n) + \frac{1}{n+1} \leq (I.h) \sum_{i=1}^{n} \frac{1}{i} + c + \frac{1}{n+1} = \sum_{i=1}^{n+1} \frac{1}{i} + c$
as required.
$\implies$ Therefore, proof is complete and the induction holds.

$\exists c > 0 : \mathcal{T}(n) \leq \sum_{i=1}^{n} \frac{1}{i} + c\ if\ n > 0\ else\ c$ ■
$(\exists c > 0 : \mathcal{T}(n) \leq \sum_{i=1}^{n} \frac{1}{i} + \mathcal{O}(1)\ if\ n > 0\ else\ \mathcal{O}(1)\ )$

שאלה 1:

2. מתקיים לכל $n \geq 10$ $T(n) = 5^n + 3T(\lfloor n^{2/5} \rfloor$ ולכל $T(n) = c\ n < 10$

מתקיים לכל $n \geq 10$, $5^n + 3T(\lfloor n^{\frac{2}{5}} \rfloor \geq 5^n$ ולכן עבור $no = 10$ ו- *c= 1* יתקיים כי $T(n) = \Omega(5^n)$

כעת נוכיח באינדוקציה כי קיים קבוע t כך שלכל *n* מתקיים $T(n) \leq 5^n * t$

בסיס: עבור *n=0* ועבור *t>=c* מתקיים $T(0) = c \leq t$

צעד: נניח כי) הטענה מתקיימת לכל גודל שקטן מ*n* ונוכיח עבור *n*

מתקיים: $T(n) = 5^n + 3T\left(\lfloor n^{\frac{2}{5}} \rfloor\right) \leq 5^n + 3 * 5^{\lfloor n^{\frac{2}{5}} \rfloor} * t$

נובע מהנחת האינדוקציה ומכך ש$n > \lfloor n^{2/5} \rfloor$ לכל *n>=10*.

כעת נרצה שיתקיים : $5^n + 3 * 5^{n^{\frac{2}{5}}} * t \leq 5^n * t$

כלומר $1 + \frac{3t}{5^{n-n^{\frac{2}{5}}}} \leq t$

ובאופן יותר פשוט: $t \geq \frac{5^{n-n^{\frac{2}{5}}}}{5^{n-n^{2/5}} - 3}$

מתקיים כי הערך המקסימלי של הביטוי האחרון שקיבלנו הוא עבור *n= 10* עבור *n>= 10*

ולכן נדרוש ש t יהיה גדול מערך זה כלומר *t>= 1.00001* ולכן עבור $t = \max\{c, 2\}$ מתקיימת הטענה לכל *n*.

ובסה"כ הראנו כי $5\hat{}n$ הינו חסם אסימפטוטי אדוק.

Q2

1. $find\_vote(A)$:
2. $\quad while\ n > 1:$
3. $\quad\quad A, n = reduce(A, n)$
4. $\quad\quad if\ n < 2:$
5. $\quad\quad\quad return\ getVote(A[A.length - 1])$
6.
7. $reduce(A, n):$
8. $\quad Helper = []$
9. $\quad loop\ through\ i = 0, 2, 4, \ldots, n-2$
10. $\quad\quad if\ isMatching(A[i] = A[i+1])$
11. $\quad\quad\quad Helper.append[A[i]]$
12. $\quad Helper.append(A[n-1])$ //in case n is odd and we didn't compare it
12. $\quad return\ Helper,\ \lceil \frac{n}{2} \rceil$

Correctness:
As we iterate through $find\_vote$ in each iteration we half the amount of votes that we look at. Since the candidate that get's chosen will definitely have a majority of at least $\lceil \frac{n}{2} \rceil$ out of n votes, if each time we're only saving half the votes at most where we compare between each vote $\forall i \in [n-1] : i, i+1$ since according to the pigeon theory, if the winning candidate has over $\frac{n}{2}$ votes and there are n votes total in the first iteration and we have $\frac{n}{2}$ "cages" with two voters in each, then we will have at least one "cage" (comparison) where the two voters voted for the same candidate. Then we are left with $\frac{n}{2}$ voters, where we know at least one is definitely the winning candidate. At most we have $\frac{n}{4}$ if all the same voters were compared with each other, Otherwise we equally knock off other voters if they get compared with the candidate who will win. $\implies$ we are still left with a majority of the votes to the candidate who will win out of the remaining $\frac{n}{2}$ voters and each step of the iterative steps and if we continue inductively until we remain 1, 2 votes left where the last one voter (if length is odd then $A[0]$ else $A[1]$) is the candidate who won the election.
Reason being that in the case of 2, perhaps throughout the whole process $A[0]$ or some voter that got put into $A[0]$ survived all the comparisons but isn't the winner, but we know out of the two that there is a winner therefore the second one must be the winner.

Proof of runtime:
$reduce(A, n)$ has a runtime of $\mathcal{O}(n)$ (recall n changes throughout calls to $find\_votes$ where n is the size of the array input into the function. Runtime is derived from the loop in line 9, every other action runs in $\mathcal{O}(1)$.

$find\_vote$ starts with n and we iteratively reduce the size of a, n by calling $reduce$ which divides the size of A, n by half in each call in the worse case scenario. As we iterate through the loop goes through half the items in each call.
Therefore we get a runtime in $find\_vote$ that get's called as follows: $n + \frac{n}{2} + \frac{n}{4} + \ldots + \frac{n}{2^j}$ where $\frac{n}{2^j}$ is smaller than 2.
$\implies$ runtime of $\mathcal{O}(n \cdot log(n))$ (well known that this series sums up to $n \cdot log(n)$)
rest of the lines in the function run at $\mathcal{O}(1)$ runtime thus making the total runtime to be $\mathcal{O}(n \cdot log(n))$ $\blacksquare$

שאלה 3:

ראינו בתרגול 6 שאלות דומות עם מבנה נתונים דינמי שעושה *INIT INSERT* ו *DELETE* באופן דומה לדרישות בתרגיל שלנו.

כלומר *D* יהיה עץ 3_2 עם התכונות הנוספות הבאות :

v.size = number of leaves in the subtree rooted at v

updating the value of size attribute will occur during insert and delete operations in O(log n) time each

נבצע את הפעולה *Average of* ע"י הפעולות מהמבנים הדינמיים בתרגול.

הפעולה תבוצע באופן הבא:

- בשלב הראשון נחפש ע"י *search_3_2*
  את הצומת שהמפתח שלה שווה ל*k1* , במידה ולא נמצא נרצה למצוא את הצומת עם המפתח הכי קטן שגדול מ*k1*
  נעשה ע"י *successor_3_2* ונסמן את המפתח שחזר *v1*

- עכשיו באופן דומה נרצה לחפש ע"י *2_3_search* את הצומת שהמפתח שלה שווה ל*k2* , אם לא נמצא נרצה למצוא את הצומת עם המפתח הכי גדול שמקיימת שהמפתח שלה קטן מ*k2* , דיברנו בהרצאה על כך שנוכל לממש *predecessor* באופן דומה לאיך שמימשנו *successor*
  וכך נוכל מצוא את המבוקש, נסמן את המפתח שחזר *v2*

- כעת קיבלנו את הצמתים עם המפתחות המינימליים והמקסימלים בקבוצה שעליה נרצה לחשב את הממוצע , לכן ע"י שימוש בפעולות:
  Sum_of_smaller ו *rank* שראינו בתרגול נוכל לקבל את הדרוש.
  אם המפתח שקיבלנו מהחלק הראשון גדול מהמפתח שקיבלנו בחלק השני נחזיר 0 אחרת נחזיר:

$$\frac{Sum_{of_{smaller(D,v2)}} - Sum_{of_{smaller(D,v1)}} + v1.key}{Rank(v2) - Rank(v1) + 1}$$

השתמשנו בפונקציות מהתרגול שעומדות בסיבוכיות הרצוייה באופן נפרד אחת מהשנייה ולכן הסיבוכיות של הפונקצייה שכתבנו עומדת בדרישות.

Q4

We will add a separate Minimum heap called A for H

Algorithm description:
$1.$ add the children of root H to A
$2.$ Print $H(1)$ as it's the minimal item.
$3.$ Bring k down by 1. (if $k < 1$ then don't print anything)
$4.$ as long as the $k > 0$ we iterate over the following process:
$a.$ Get minimal value from A (and remove it from A) via $Extract\_Min$ and save it to a variable x
$b.$ print x
$c.$ add a's children from H into the heap A

Correctness:

$Side\ lemma$: Given a set of i items with minimal keys such that $X = \{a_1, a_2, \ldots, a_i\}$ in the $heap$ H then the following holds such that the item a_{i+1} makes it so that $parent.\, a_{i+1} \in X$
$Proof$: Let's assume by contradiction that the lemma is incorrect. Therefore, there exists an item a_i that isn't in X such that $parent.\, a_{i+1} = a_j$. This is in contradiction to the $heap$ characteristic where $H(a_j).\, key < H(a_{i+1})$ so that the key of a_{i+1} is the next item in the order.

Since every child that's printed is always added to A, therefore, the next item to be printed is always in A and we will always print the smallest item.

$Runtime$: the size of the $heap$ A is at most $2k$, we know that adding and removing an item from A will take $log(k)$.
$\implies$ runtime is at most $\mathcal{O}(2 \cdot k \cdot log(k)) \implies \mathcal{O}(k \cdot log(k))$

שאלה 5:

1. תהיה צומת x שיש לה שני בנים , מתקיים כי הצומת העוקב נמצא בתת העץ המושקש בבן הימני של x כיוון שהצומת העוקב בעל מפתח גדול יותר מהמפתח של x ולפי תכונות של עץ חיפוש בינארי זה מתקיים רק בתת העץ המושרש בבן הימני.
מתקיים כי הצומת העוקב הוא הצומת עם המפתח המינימלי שגדול מהמפתח של x ולכן הוא המפתח המינימלי בתת העץ המושרש בx , לפי ההרצאה מינימום בתת עץ חיפוש בינארי הינו הליכה שמאלה עד שמגיעים לעלה ולכן הצומת העוקב לא יכול להיות בעל בן שמאלי שהרי במקרה זה הבן שלו היה הצומת העוקב ולא הוא .
באופן דומה נקבל את הטענה ההפוכה עבור הליכה ימינה.

2. נסמן ב-y את אב קדמון הכי עמוק של x, שהבן השמאלי שלו אב קדמון ל-x.

נניח בשלילה כי קיים $v \neq v_2 \in V$ כל שמתקיים ש- $v.key > v_2.key >$ $x.key$ על פי ההגדרה של עץ בינארי ואיך שהגדרנו את הצמתים.

נחלק למקקים השונים ונבדוק כל אחד בנפרד:

<u>מקרה 1, v_2 אב קדמון של v:</u>
לפי ההנחה מתקיים כי $v.key > v\_2.key$ ולכן מתקיים כי v בעץ המושרש בצד ימין של v_2 ובפרט v אב קדמון של x ולכן הוא גם בעץ המושרש ב v_2. נובע מזה ש- $x.key > v\_2.key$ בסתירה לזה ש $v_2.key >$ $x.key$לא קיים צומת v_2.

<u>מקרה 2, אם v_2 צאצא של v:</u>

אזי, אנחנו יודע כי v_2 נמצא בעץ השמאלי מכיוון ש $v_2.key < v.key$ לכן קיים צומת שנסמנו $v_3 \in V$ בעץ השמאלי המושרש ב v ומתקיים כי $v_2 = v\_3$ או לx in v_3 $left$ $tree$ וגם כי v_2 in $right$ $tree$
ולכן, v_3 אב קדמון של x וזה גורר כי הבן השמאלי של v_3 אב קדמון של x בסתירה להגדרה של v ולכן אין v_2 כזו.

<u>נסתכל על היחס בין v_2, v:</u>
יש צומת $v_3 \in V$ כך ש-v_2 בעץ השמאלי המושרש ו-v בעץ המושרש שלו מצד ימין. (מתקיים $v_2.key < v.key$) ובפרט x צאצא של v ולכן מופיע בעץ המושרש של v_3 ומתקיים $x.key >.key > v\_2.key$
בסתירה לזה כי $x.key > v\_2.key$ ולכן אין v_2 כזו.

לסיכום, מכיוון שנתון כי כל המפתחות ייחודיים אזי ה $successor$ של x ייחודי ומתקיים את מה שרצינו להוכיח.

3. יהי x עלה , אם x בן שמאלי מתקיים כי $x.key < parent.key$ וכן מתקיים כי לא x אין ילדים ובפרט אין לו בן ימני ולכן לפי סעיף 2 הצומת העוקב שלו הוא האב הקדמון העמוק ביותר שהבן השמאלי שלו הוא גם אב קדמון של x כלומר זהו ההורה של x.

אם x בן ימני מתקיים $x.key > parent.key$ ואפשר להראות בצורה דומה לסעיף 2 כי אם לצומת x אין בן שמאלי אזי הצומת הקודם לא x היא האב הקדמון העמוק ביותר שהילד הימני שלו הוא גם אב קדמון של x .

ולכן באופן דומה למקרה הראשון לא x אין ילדים ובפרט אין לו בן שמאלי ולכן הצומת הקודם שלו הוא האב הקדמון העמוק ביותר שהבן הימני שלו הוא אב קדמון של x כלומר ההורה של x .

כלומר הראנו שבתנאים של הסעיף ההורה של x יהיה הצומת העוקב או הקודם של x .