<u>תרגיל יבש 3 במבני נתונים</u>

לירן רדל, 314989427 אנה גריגוריבקר 321931396

<u>שאלה 1</u>

<u>הפרכה:</u> נניח בשלילה כי האלגוריתם המתואר בתרגיל קיים. ניקח n מספרים כלשהם, נכפיל כל אחד מהם במספר שאינו ראשוני כלשהו על מנת להפטר מקיום אפשרי של מספרים ראשוניים כלשהם, לדוגמא נכפילם ב-4 ($O(n)$) , כעת יש בידינו n מספרים שאינם ראשוניים. נאתחל ערימת מינמום ב-$O(n)$ כפי שלמדנו , עבור n האיברים בערימה נמצא בכל פעם את האיבר המינימלי ע"י הפעולה $Findmin()$ נשבץ את האיבר במערך (סה"כ שתי הפעולות הללו יעשו בסיבוכיות של $O(1)$) ונשתמש באלגוריתם הנתון כדי לבצע את פעולת $DelMin()$ בסיבוכיות $O\left(\log^* n\right)$ נחזור על פעולות אלו עבור n האיברים תוך שיבוץ האיברים במערך לפי סדר הוצאתם מהערימה. כעת נחלק את כל המספרים ב-4 ע"מ לחזור למספרים המקוריים איתם התחלנו, נקבל מערך ממויין לפי גודל של n מספרים כלליים. סה"כ הסיבוכיות שנקבל היא $O(n) + n \cdot O(1) + n \cdot O\left(\log^* n\right) \ = \ O\left(n \log^* n\right) \underset{*}{=} o(n \log n)$ בסתירה לחסם התחתון על סיבוכיות של אלגוריתמי מיון אותו למדנו בהרצאה ($\Omega(nlogn)$).

*כשאר $\log^* n = \min\left\{i \geqslant 0 \mid \log^{(i)}(n) \leqslant 1\right\}$ למדנו בהרצאה כי $\log^* n = o(\log n)$ ולכן אי השוויון מתקיים.

<u>שאלה 2</u>

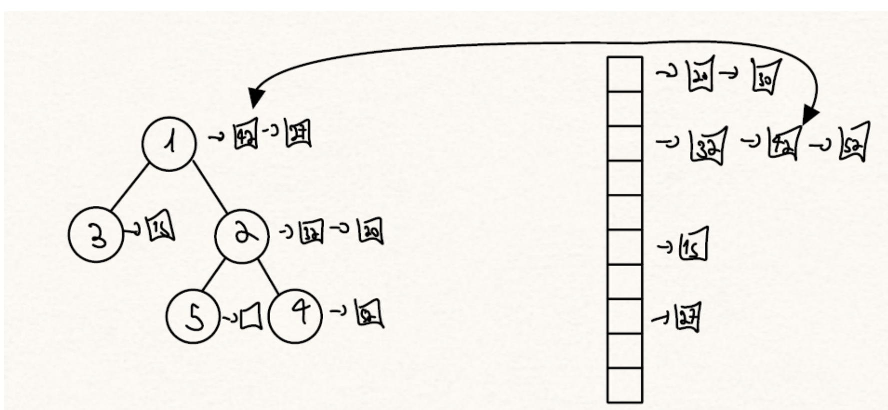
נממש את הפעולה $MaxPalindromeSuffix(s)$:
ראשית נבנה עץ סיומות דחוס של המחרוזת s בעזרת האלגוריתם עליו למדנו בהרצאה בסיבוכיות זמן ומקום של $O(|s|)$. כעת נקח את המחזורת הנתונה ונהפוך אותה (סיבוכיות זמן ומקום של $O(|s|)$). ע"מ לעקוב אחר האורך של הסיומת הכי ארוכה שהיא פלינדרום שמצאנו עד כה נחזיק משתנה k אותו נעדכן בהתאם. כעת נבצע סיור בעץ הסיומות הדחוס לפי המחרוזת ההפוכה, כל פעם שנגיע לצומת שממנו קשת לעלה או שהוא בעצמו עלה נעדכן את k להיות אורך הסיומת עליה ביצענו את המעבר במידה והוא גדול יותר מהערך הנוכחי ש-k מחזיק. נסיים כאשר נגיע לעלה ונחזיר את k. כיוון שהסיור על הסיומות נעשה לפי סדר האותיות שבמחרוזת s מהסוף להתחלה נקבל סיור על סיומות שהן פלינדרום בלבד ,בעצם אנחנו בודקים אם ההתחלה של הסיומת זהה לסוף של המחרוזת, כאשר |s| הוא גודל המחרוזת נמצא את הקשת שמחזיקה את האות

הנמצאת במקום ה $s[|s|]$, אם היא האות היחידה על הקשת נמשיך לצומת הבא אם לא נבדוק אם האות אחריה זהה לאות במקום ה-$s[|s|-1]$ וכך הלאה עד שנעבור על כל האותיות בקשת או עד שלא תהיה זהות בין האות שעל הקשר לבין האות במקום המתאים במחרוזת ההפוכה, במידה ועברנו על כל האותיות בקשת נעבור לצומת הבא, אם קיימת ממנו קשת נוספת (הוא לא עלה) נבדוק אם הקשת הזאת מחזיקה את האות הנמצאת במקום המתאים בהתאם למעבר שלנו במחרוזת ההפוכה, נמשיך כך עבור כל קשת נוספת או עבור כל אות נוספת שנמצאת על אותה הקשת,תוך התקדמות במקביל על המחרוזת ההופכית. נשים לב שמקרה הגרוע נבצע סיור הכולל מעבר על הקשת שמחזיקה את האות האחרונה של s וסיור $postorder$ על תת העץ אשר מתחבר אל השורש עם הקשת המדוברת ,קשת כזו היא יחידה בגלל המבנה של עץ הסיומות, לכן סיור זה יעשה בסיבוכיות של$O(|s|)$. סה"כ סיבוכיות זמן ומקום $O(|s|)$ כנדרש.

<u>שאלה 3</u>

<u>נגדיר את המבני נתונים הבא:</u>

טבלת עירבול דינאמית הממומשת בעזרת Chain-Hashing של כל האנשים במערכת, וערימת מינימום שתייצג את המיקומים בתור של האנשים. כל איבר בערימה יחזיק בנוסף למיקום שלפיו הערימה ממויינת מצביע לרשימה מקושרת שבראשה יהיה האדם שצריך להכנס ראשון מבין חבריו. כל איבר ברשימה מקושרת זו יחזיק מצביע אל המיקום שלו בטבלת העירבול, ובטבלת הערבול כל תא המייצג אדם יחזיק מצביע אל התא שלו ברשימה המקושרת בערימה במיקום המתאים.



<u>מימוש הפונקציות:</u>

$Init(people)$**:** ניצור טבלת ערבול בגודל המערך people (ניתן לגלות אותו ב-$O(n)$), ונכניס את האנשים אל טבלת הערבול, ותוך כדי הכנסת כל אדם אל טבלת הערבול ניצור ערימת מינימום, ונעדכן את המצביעים עבור כל אדם. הכנסת כל איבר אל טבלת הערבול כאשר גודל המערך אינו גדל נעשה ב-$O(1)$ בממוצע, לכן הכנסת כל האנשים אל טבלת הערבול תעשה ב-$O(n)$ בממוצע. יצירת ערימת מינימום לפי המיקומים בתור של האנשים במערך, תוך כדי בהתאמה אחרי כל הכנסה עדכון המצביעים הרלוונטים והוספת אל ראש

הרשימה המקושרת שכל תור מחזיק כפי בנלמד בהרצאה, נעשה ב-$O(n)$. לכן בסה"כ סיבוכיות של $O(n)$ בממוצע כנדרש. סיבוכיות הזכרון תהייה $O(n)$ מאחר ויצרנו ערימה עם n איברים, טבלת ערבול בעלת n איברים כשהמערך שלה בעל n תאים, וכל איבר בערימה עם רשימה מקושרת של איבר אחד, ולכן בסה"כ סיבוכיות מקום של $O(n)$.

$Insert(id, friend\_id)$: ראשית נחפש את האדם בעל ת"ז של friend_id בטבלת הערבול ב-$O(1)$ בממוצע כפי שנלמד בהרצאה. לאחר מכן נלך אל המצביע שלו לערימת המינימום של הרשימה המקושרת של המיקום בתור, ונשמור את המצביע בצד (אם friend_id אינו האיבר האחרון ברשימה סימן שהאדם id לא יכול לבוא מייד אחריו לכן נחזיר שגיאה). לאחר מכן נוסיף את האדם בעל המזהה id אל טבלת הערבול בסיבוכיות של $O(1)$ בממוצע על הקלט משוערך כפי שנלמד בהרצאה (מאחר ויש מצב נצטרך להגדיל את המערך הדינאמי ולבצע העתקה והגדלה), ונשמור גם את המצביע אל המיקום ברשימה המקושרת במערך. לאחר מכן נלך את המצביע הראשון ששמרנו, נוסיף מימינו(סוף הרשימה) את האדם בעל המזה id שיהיה בסוף התור של המיקום, ואז בעזרת המצביע השני ששמרנו נגרום להם להצביע אחד על השני(הצבעה דו-כיוונית) ב-$O(1)$. לכן סיבוכיות בסה"כ של $O(1)$ בממוצע על הקלט משוערך, כנדרש.

$LetsEat()$: אנו רוצים את האדם הראשון בתור, ולכן ניגש אל האיבר בראש ערימת המינימום ב-$O(1)$, נפנה את האיבר הראשון ברשימה המקושרת המסמל את האדם הראשון האמור להכנס אל המסעדה, נשמור את המצביע אל המיקום שלו בטבלת העירבול ואז נמחק את אותו מן הרשימה המקושרת. אם האיבר היחידי ברשימה, נמחק את האיבר בראש הערימה בסיבוכיות של $O(logn)$ כפי שנלמד בהרצאה, ונשמור את המזהה שלו. לאחר מכן נלך אל המצביע בטבלת הערבול של האדם ונמחק אותו משם על ידי שינויי מצביעים ברשימה המקושרת בטבלת הערבול
ב-$O(1)$ מאחר ואנחנו לא צריכים לחפש את האדם בטבלת הערבול אלא ניגשים לאיבר ישירות. לבסוף נחזיר את המזהה של האדם. יש לציין כי אינינו מקטינים את גודל המערך, על מנת לא להפוך את הפעולה למשוערכת מאחר ומדובר במערך דינאמי וכפי שנלמד בהרצאה זאת פעולה משוערכת, לכן לא נקטינו. הגדלת המערך תיעשה רק כאשר נגיע למצב שבו קיים תור יותר ארוך ממה שהיה בעבר לכן סיבוכיות המקום של $O(n)$ נשמרת כי אנו דואגים להסיר בפעולה זו את האיבר מן המערך. לכן בסה"כ סיבוכיות של $O(logn)$ כנדרש.

שאלה 4

סעיף א:

מבני הנתונים שנשתמש בהם בשביל לפתור שאלה זו הם:
נחזיק מילון שימומש ע"י טבלת ערבול בשיטת ערבול double hashing הממומשת בעזרת מערך

דינאמי, ונחזיק מונה t. המפתח של המילון יהיה בעצם x , והתוכן יהיה מצביע לעץ Avl , כאשר צמתיו הן גם מילון. כלומר כל צומת תכיל מפתח שהוא בעצם הערך t, והתוכן שלו הוא משתנה בוליאני שנקרא לו im_here שבעצם יאמר לנו האם בזמן t האיבר נמצא או לא נמצא במבני נתונים.

<u>מימוש הפונקציות:</u>

$Init()$: נאתחל את המילון בעזרת טבלת ערבול ריקה כפי שנלמד בהרצאה ואת את t להיות 0 , ולכן האתחול קורה ב - $\boldsymbol{O}(\mathbf{1})$.

$Insert(x)$: נגדיל את המונה t ב- 1 ואז נוסיף את האיבר x אל המילון שלנו. ייתכנו 2 מצבים: אם הוא לא קיים, נכניסו ונחשיב את האפשרות כי נצטרך להגדיל את המערך הדינאמי פי 2, נאתחל את העץ של האיבר, ונוסיף לתוכו את הצומת עם המפתח t הנוכחי, כשתוכנה היא im_here=true . במקרה שכבר איתחלנו את האיבר x מקודם, כלומר הוא הוסף בעבר פעם אחת לפחות למבני נתונים בעזרת פעול ת Insert , אז בעזרת חיפוש בטבלת בערבול ב - $\boldsymbol{O}(\mathbf{1})$ בממוצע על הקלט, נמצא את האיבר, נפנה לעץ Avl שלו ונוסיף את הצומת עם המפתח t הנוכחי ועם תוכן שהוא הערך בוליאני im_here=true . ב- 2 במקרים אנו מכניסים לתוך העץ צומת חדשה, שפעולה זאת לוקחת לפי מה שנלמד $\boldsymbol{O}(\boldsymbol{log}(\boldsymbol{n}_x))$, ומאחר וקיים מקרה שבו אנו עלולים להגדיל את המערך הדינאמי, אז בסה"כ פעולת Insert(x) לוקחת $\boldsymbol{O}(\boldsymbol{log}(\boldsymbol{n}_x))$ בממוצע על הקלט, משוערך.

$Remove(x)$: נגדיל את המונה t ב- 1 , ונבדוק האם האיבר x קיים במילון. אם אינו קיים, אז אנו מנסים להסיר איבר שאינו קיים ולכן לא נעשה כלום, מאחר ואין צורך לנו לזכור זאת בהיסטוריה של המבני נתונים. אם נמצא איבר כזה (כמו שהוסבר בפעולה הקודמת, חיפושו לוקחת $\boldsymbol{O}(\mathbf{1})$ בממוצע על הקלט) אז לאחר מציאתו נפנה אל העץ Avl המשוייך אליו, ונכניס לתוכו את הצומת עם המפתח t הנוכחי עם התוכן is_here=false . בפעולה זאת אנו לא משנים כלל את גודל המערך הדינאמי מאחר ופעולת ההסרה משפיע נטו על גודל העץ ולא על המערך כלל, לפיכך בסה"כ סיבוכיות פעולה זאת הינה $\boldsymbol{O}(\boldsymbol{log}(\boldsymbol{n}_x))$ בממוצע על הקלט.

$Find(x,t)$: ראשית נחפש את האיבר x בתוך טבלת הערבול. אם אינו נמצא, אז מעולם לא הוסף ולכן לא נמצא בהיסטוריית במבני ולכן נחזיר False . אחרת האיבר הוסף לפחות פעם אחת בעבר. חיפוש האיבר כפי שהוסבר מקודם לוקח לנו $\boldsymbol{O}(\mathbf{1})$ בממוצע על הקלט. לאחר מציאתו נפנה אל העץ Avl שלו, ונבצע את האלגוריתם הבא: האלגוריתם מבוסס על אלגוריתם חיפוש איבר בעץ בינארי, עם טוויסט קטן: נחפש בעזרת אותו אלגוריתם את הצומת בעלת המפתח או השווה אל t או הכי קרובה בערך המפתח שלה ל- t בעץ (כלומר את הצומת הראשונה בחיפוש שלנו ש המפתח שלה ושל הבן השמאלי שלה קטן או שווה לערך t, ושהמפתח של הבן הימני שלה גדול או שווה אל t). כשהגענו אלייה, פשוט נחזיר את is_here שאומר לנו על הפעולה הכי אחרונה שבוצעה על x , ובמקרה שמפתח הצומת גדול מ- t זה לא משנה את התוצאה, מאחר ו -או הכנסנו או הסרנו את x ומאז עבדנו על איברים אחרים מה שלא השפיע על הפעולה האחרונה שבוצעה על x, לכן מספיק לנו להחזיר את הערך x. במקרה שבו לא מצאנו צומת כזאת, סימן שעד הזמן t לא הוכנס כלל x , לכן נחזיר false . מאחר ואלגוריתם חיפוש בעץ ייקח לנו $\boldsymbol{O}(\boldsymbol{log}(\boldsymbol{n}_x))$ כשיש n_x איברים בעץ

כמספר פעולות Insert ו- Remove שבוצעו על x, אז בסה"כ סיבוכיות הפעולה הינה$O(log(n_x))$
 בממוצע על הקלט. מאחר ועבור על פעולת Insert ו- Remove שאנו מבצעים על איבר x כלשהו נכניס איבר חדש בעץ, סיבוכיות המקום של כלל העצים תהייה סכום כל פעולות ההכנסה וההוצאה שנבצע, ומאחר ופעם אחת בלבד בפעם הראשונה שנקרא לפעול ת Insert נכניס איבר חדש למערך הדינאמי שישאר לתמיד, אז גודל המערך יהיה כמספר איברי x השונים שנכניס, שמספר זה קטן שווה אל סכום על פעולות Insert ו- Remove החוקיות, ולכן בסה"כ סיבוכיות המקום תהייה $O(k)$ כאשר k זה מספר פעולות Insert ו- Remove החוקיות שבוצעו על כל האיברים במבנה, כנדרש.

<u>סעיף ב:</u>

ראשית נסביר את מבני הנתונים:
נגדיר רשימה מקושרת אשר כל איבר מסמל מספר, ומצביע אל תחילת הרשימה וסוף הרשימה.
<u>מימוש הפונקציות:</u>
$Init()$: נאתחל רשימה מקושרת ריקה ואת המצביעים ל-NULL, ב-$O(1)$.
$Insert(x)$: נבדוק האם הרשימה ריקה לפי מצביע לתחילת הרשימה. אם ריק נוסיף את x ללא בדיקה. אחרת, נשווה את x אל האיבר בסוף הרשימה לפי המצביע. אם x גדול מהאיבר האחרון, נכניס אל סוף הרשימה ונשנה את מצביע לסוף הרשימה להצביע ל-x. אחרת, נוציא את כל האיברים החל מתחילת הרשימה בעזרת המצביע לתחילת הרשימה ותוך כדי ההוצאה נדפיס אותם ולבסוף נכניס את x לרשימה ונשנה את המצביעים בהתאם.

הוכחת הסיבוכיות של $Insert(x)$:

נסתכל על סדרה כלשהי של m פעולות Insert(x), כאשר נפריד את פעולת ה-Insert ל-2 מקרים שונים:
מקרה ראשון - x גדול מהאיבר האחרון שהוכנס, נקרא לזה $Insert_1(x)$ שלוקחת $O(1)$.
מקרה שני - x קטן או שווה לאיבר האחרון שהוכנס. נראה לזה $Insert_2(x)$ שלוקחת $O(m)$ כאשר m זה מספר האיברים ברשימה ברגע קריאת הפעולה.

<u>שיטת הצבירה:</u>
נחלק את סדרת הפעולות ל-L מקטעים באורכים של $m_1, m_2, ..., m_L$, כאשר כל מקטע יכיל אוסף פעולות בין פעולת $Insert_1(x)$ (לא כולל) לפעולת $Insert_2(x)$ העוקבת (כולל).
נראה שכל מקטע באורך m_i רץ בזמן $O(m_i)$ כאשר $1 \leqslant i \leqslant L$, ומכאן נסיק שהסדרה בת m הפעולות רצה בזמן ריצה של $O(m)$ (כפי שראינו בתרגול). נשים לב ש-$Insert_1(x)$ נעשה ב-$O(1)$ ו-$Insert_2(x)$ נעשה ב-$O(k_i)$ כאשר k_i הוא מספר האיברים אשר נמצאים במחסנית לאחר ביצוע של i פעולות, בנוסף נשים לב שמיד לאחר פעולה מסוג $Insert_2(x)$ יהיה בדיוק איבר אחד במחסנית, וכי עבור כל פעולה מסוג זה כמות האיברים שנוציא לא יכולה להיות יותר גדולה מכמות האיברים שהכנסנו בין 2 פעולות כאלה, לכן $k_i \leqslant m_i$.
לכן זמן הריצה הכולל עבור הקטע m_i הוא: $m_i \cdot O(1) + O(k_i) = O(m_i)$.

ולכן זמן הריצה הכולל עבור m פעולות $Insert(x)$ הינו $O(m)$ לכן זמן הריצה המשוערך של הפעולה הוא $O(1)$ כפי שלמדנו.

<u>שיטת החיובים:</u>

נגדיר את התשלומים עבור כל פעולה באופן הבא:

נשלם עבור כל פעולת $Insert_1(x)$ 2 שקלים, כלומר $a_i = 2$ (שקל אחד יישמר בצד עבור הוצאת המספר בהמשך)

כאשר המחיר בפועל עבורה הינו $t_i = 1$, ונשלם עבור כל פעולת $Insert_2(x)$ שקל אחד לכל מספר שנצטרך להדפיס ולהוציא, כלומר $a_i = 1$ שמחירה בפועל הינו $t_i = m_i + 1$ כאשר m_i זה מספר האיברים ברשימה ברגע קריאת הפונקציה.

נראה כי לאחר סדרה של m פעולות(המתחיל ממבני נתונים ריק) הבנק יישאר בעודף:$\sum_{i=1}^{m}(a_i - t_i) \geqslant 0$,

ומתקיים:

$2m \geqslant \sum_{i=1}^{m} a_i \geqslant \sum_{i=1}^{m} t_i \geqslant m$, ומאחר ויש m פעולות אז הסיבוכיות המשוערכת הינה $O(1)$.

<u>שיטת הפוטנציאל:</u>

נגדיר $\phi_i = |D_i|$ כאשר $|D_i|$ הוא מספר האיברים במחסנית לאחר i פעולות.

עבור $Insert_1(x)$:

$a_i = t_i + \phi_i - \phi_{i-1} = 1 + 1 = 2$ (מספר האיברים במחסנית גדל בדיוק ב-1)

עבור $Insert_2(x)$:

$a_i = t_i + \phi_i - \phi_{i-1} = 1 + |D_{i-1}| + 1 - |D_{i-1}| = 2$ (נוציא את כל האיברים שהיו במחסנית ונכניס איבר אחד)

היות ו-$\phi_m \geqslant \phi_0$ נקבל:

$$\sum_{i=1}^{m} a_i = \sum_{i=1}^{m} t_i + \phi_m - \phi_0 \implies \sum_{i=1}^{m} t_i = \sum_{i=1}^{m} a_i + \phi_0 - \phi_m \leqslant \sum_{i=1}^{m} a_i \leqslant 2m = O(m)$$

קיבלנו שסדרה של m פעולות מתבצעת בסיבוכיות זמן של $O(m)$ ולכן זמן הריצה המשוערך של הפעולה הוא $O(1)$ כפי שלמדנו.