דף שער

מבני נתונים 1
234218

תרגיל יבש 3

מספר

הוגש ע"י:

315823856	גיא אוחיון
מספר זהות	שם
207733015	רון קנטורוביץ'
מספר זהות	שם

ציון:

לפני בונוס הדפסה:

כולל בונוס הדפסה:

נא להחזיר לתא מס':

שאלה 1

במהלך הפתרון נרצה להיות מסוגלים להתייחס לתחנה באמצעות מספר התחנה שלה או באמצעות הקואורדינטות שלה ששני הייצוגים יהיו שקולים, ולכן נשמור את התחנות באובייקטים בעלי כמה שדות, כך שבכל אובייקט המייצג תחנה יהיו שמורים כל הנתונים שלה: מספר התחנה וזוג הקואורדינטות.
מבנה הנתונים יכיל את המשתנים הבאים:

1. 2 מערכים, כל אחד מכיל את n התחנות. מערך אחד יהיה ממוין בסדר עולה לפי קואורדינטת ה־x ובמיון משני לפי קואורדינטת ה־y, (כלומר זוגות סדורים בעלי אותה קואורדינטת x ימוינו לפי קואורדינטות y), והמערך השני יהיה ממוין בסדר עולה לפי קואורדינטת ה־y ובמיון משני לפי קואורדינטת ה־x. נכנה אותם $WestToEast$ ו־$SouthToNorth$ בהתאמה.

2. מערך באורך n של התחנות שנכנה "$Stations$" שיהיה ממוין לפי מספרי התחנה בסדר עולה.

3. 2 מבני $Master/Close$ במימוש שראינו בתרגול, שנכנה "$MCEast$" ו"$MCNorth$" בהתאמה. במבנים אלה נשתמש על מנת לייצג עבור כל תחנה את התחנה הפעילה הקרובה אליה ביותר לכיוון מזרח או צפון בהתאמה.

להלן מימוש הפעולות:

:<u>$Init((x_1, y_1), ..., (x_n, y_n))$</u>

ראשית נאתחל את מערך $Stations$ כאשר בכל תא שלו נכניס לפי סדר הקלט אובייקט עבור כל תחנה, שיכיל את מספרה הסידורי בקלט ואת שתי הקואורדינטות שלה כאמור לעיל.

לאחר מכן נאתחל את המערכים $WestToEast$ ו־$SouthToNorth$: בתור התחלה נעתיק אליהם את המערך $Statins$ (בהמשך נמיין אותם).

כל אתחולים המערכים הללו מבוצעים ב־$O(n)$ משום שקיימים מספר קבוע של מערכים (3), ובכל אחד מהם מספר האיברים הוא כמספר התחנות, ועבור כל איבר אנו מבצעים מספר קבוע של פעולות השמה ב־$O(1)$.

לאחר האתחול המערכים שניים מהמערכים אינם ממוינים, ולכן נפעיל על כל אחד מהם פעמיים את אלגוריתם $CountingSort$ שראינו בהרצאות:

על מערך $WestToEast$ נפעיל את $CountingSort$ תחילה כך שימיין לפי קואורדינטת y, ובהפעלה השנייה נמיין לפי קואורדינטת x.

באופן דומה, על מערך $SouthToNorth$ נמיין תחילה לפי x ולאחר מכן לפי y.

מכיוון שראינו בהרצאה ש־$CountingSort$ הוא אלגוריתם מיון יציב, נקבל שהמערכים אכן ממוינים באופן שהגדרנו בתחילת הפתרון.

כמו כן, מכיוון שאורכי המערכים הם n, וכן לפי הנתון n הוא גם חסם על קואורדינטות מיקומי התחנות, נקבל שסיבוכיות הפעלת $CountingSort$ היא אכן $O(n)$, שכן גודל טווח הערכים בכל מערך שווה למספרם.

כעת נבצע אתחול של 2 מבני ה־$MasterClose$:

עבור המבנה $MCEast$, נבנה אילן יוחסין, עץ הפוך, באופן הבא:

ניצור מערך באורך $n+1$ של חוליות לאילן היוחסין. נאתחל כל חוליה ב־n החוליות הראשונות עם מספר התחנה בתא המקביל במערך $WestToEast$ (נוכל לעשות זאת משום

שבאמצעות האובייקטים שהגדרנו יש לנו גישה ב־$O(1)$ למספר התחנה של כל קואורדינטה). ובחוליה האחרונה נציב את הערך המספרי $n+1$, שהוא יהיה הערך שיסמל חריגה מגבולות המפה, כלומר תחנה תקושר אל הצומת $n+1$ אם אין תחנה ממזרח לה.

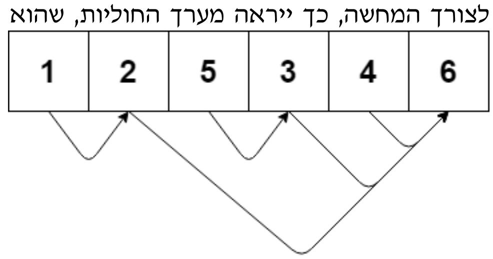
נעבור על המערך שיצרנו מהסוף להתחלה ונקשר כל תחנה לראשונה שנמצאת ממזרח לה:

את החוליה ה־$n+1$ לא נקשר לאף חוליה משום שחוליה זו תהיה שורש אילן היוחסין.

עבור כל חוליה אחרת, אם החוליה העוקבת לה במערך נמצאת באותה שורה (כלומר קואורדינטות ה־x שלהן זהות) נקשר את החוליה לחוליה הבאה אחריה (כלומר ניצור מצביע ונבצע השמה לתוכה).

אחרת, אם החוליה העוקבת נמצאת בשורה אחרת מהחוליה, נובע שהחוליה הנוכחית מייצגת תחנה שהיא המזרחית ביותר בשורה שלה, ולכן נקשר אותה לחוליה ה־$n+1$ שכן אין לאחר התחנה הזו תחנה נוספת.

לצורך המחשה, כך ייראה מערך החוליות, שהוא גם אילן היוחסין, עבור הקלט בתרגיל:



ולבסוף נאתחל את $MCEast$ עם אילן היוחסין שיצרנו. אתחול זה חוקי, שכן ראינו בתרגול שניתן לאתחל $Master-Close$ עם אילן יוחסין הממומש כעץ הפוך (בפרט המצביעים מחוליה לחוליה מממשים עץ הפוך).

עבור המבנה $MCNorth$ נפעל באופן סימטרי, אך נאתחל את אילן היוחסין שלו עם סדר מספרי תחנות הלקוחים מהמערך $SouthToNorth$ ונקשר כל תחנה לתחנה הצפונית לה באותה עמודה, או לתחנה המדומה ה־$n+1$ אם אין תחנה צפונית יותר.

בסופו של דבר נקבל 2 מבני $MasterClose$, ומהגדרת אילנות היוחסין נקבל שה־$Master$ של כל חוליה המייצגת תחנה היא התחנה הקרובה ביותר (הפעילה) ממזרח או מצפון בהתאמה, או הערך $n+1$ אם אין תחנה כזו.

בסך הכול סיבוכיות הזמן היא סיבוכיות של הקצאת מספר קבוע של מערכים בגודל n ואתחול 2 מבני $MasterClose$ בגודל n, כלומר זוהי סיבוכיות זמן של $O(n)$ במקרה הגרוע.

<u>$DriveInDirection(i,d)$:</u>

אם d הוא $East$, נקרא ל־$Master(i)$ במבנה $MCEast$.

לפי האופן שבו הגדרנו את המבנה, אכן נקבל את מספר התחנה הפעילה הבאה ממזרח לתחנה i.

אם נקבל $n+1$, משמעות הדבר היא שהתחנה i היא התחנה האחרונה בכיוון מזרח, ולכן נדע להחזיר ערך מתאים.

באופן דומה, אם d הוא $North$ נקרא ל־$Master(i)$ במבנה $MCNorth$ ונחזיר את הערך שאנו מקבלים, כיוון שמהגדרת המבנה זהו מספר התחנה הבאה הפעילה מצפון (או $n+1$ אם לא קיימת כזו).

כפי שראינו בתרגול, פעולת $Master$ ממומשת בסיבוכיות משוערכת $\log^*(n)$ כאשר n מספר האיברים בעץ, ואילו אצלנו מספר האיברים בעץ הוא $n+1$ כאשר n הוא מספר התחנות.

לכן הסיבוכיות הכוללת היא $\log^*(n+1) \le 2\log^*(n)$, כלומר $O(\log^*(n))$ משוערך כנדרש.

<u>$CloseStation(i)$</u>:

נקרא ל־$Close(i)$ עבור כל אחד מהמבנים $MCEast$ ו־$MCNorth$.

כפי שראינו בתרגול, סיבוכיות $Close$ היא $O(\log^*(n))$ משוערך, ומכיוון שאנו מבצעים 2 קריאות לפונקציה זו נקבל שזו גם הסיבוכיות המשוערת של הפעולה כולה.

<u>$MaxPath(start, end)$</u>:

נקצה מערך זמני באורך n מספר התחנות, שבכל תא יכיל ערך מספרי.

נכנה מערך זה "$PathsArray$".

בהמשך, עבור כל תא שמייצג תחנה, הערך שבתא ייצג את אורך המסלול המקסימלי מהתחנה עד לתחנה end.

לא נאתחל ולא ניגש לתאים במערך שהתחנה שהם מייצגים נמצאת ממזרח או מצפון ל־end, וכן לא לתאים שהתחנה שהם מייצגים נמצאת מדרום או ממערב ל־$start$.

כלומר מכיוון שנתון שאכן קיים מסלול חוקי מ־$start$ ל־end, אנו עובדים אך ורק עם התחנות הנמצאות "בריבוע" שקודקודיו המנוגדים הן התחנות $start$ ו־end.

כעת ניגש לתא ה־end במערך $PathsArray$ שיצרנו ונאחסן בו את הערך 0 כאורך מסלול .

כלומר אורך המסלול הארוך ביותר מ־end ל־end הוא "0" עבורנו.

עבור כל תא אחר, נאחסן בו -1. מכיוון שערך זה אינו מסמל אורך מסלול תקין, נוכל לקרוא ערך זה כ"לא קיים מסלול מהתחנה הנוכחית ל־end".

אתחול זה מתבצע ב־$O(n)$ שכן מספר הפעולות בכל תא במערך באורך n הוא סופי.

לאחר מכן, נבצע איטרציה על המערך $WestToEast$ המכיל את קואורדינטות התחנות לפי שורות מלמטה למעלה.

נמצא במערך זה את הקואורדינטות המתאימות לתחנה end (בסיבוכיות $O(n)$ אורך המערך). ונתחיל מהתחנה הנמצאת במערך זה לפני end ונלך בכל איטרציה **אחורנית** עד שנגיע לתא בעל הקואורדינטות של התחנה $start$ (נדע לזהות זאת ב־$O(1)$ שכן כל אובייקט המייצג תחנה מכיל את כל הנתונים עליה). מכיוון שבשאלה מובטח שקיים מסלול חוקי מ־$start$ ל־end, נובע לפי האופן שבו מיינו את המערכים ש־$start$ ושאר התחנות במסלול אכן נמצאות במערך לפני end.

נאמר שתחנה היא "חוקית" אם היא קיימת (כלומר לא התחנה $n+1$), וכן קיים ממנה מסלול ל־end (כלומר במערך $PathsArray$ הערך המספרי בה הוא אינו -1).

נשים לב שבהינתן מספר תחנה i, נוכל לבדוק ב־$O(1)$ האם היא חוקית, שכן הגישה לתא המתאים לה ב־$PathsArray$ וקריאת ערכו היא ב־$O(1)$.

עבור כל זוג קואורדינטות\תחנה נבצע את הפעולה הבאה:

נמצא על ידי $DriveInDirection$ את התחנות הפעילות הראשונות שנמצאות ממזרח ומצפון לתחנה הנוכחית.

<u>אם 2 התחנות שמצאנו אינן חוקיות</u>, סימן שלא קיים מסלול מהתחנה באיטרציה הנוכחית אל end, שכן מסלול חוקי ממנה אל end בהכרח יעבור דרך התחנה הפעילה הראשונה ממזרח או מצפון לפי הגדרת התרגיל.

לכן נשאיר במקרה זה את הערך בתא המתאים לה ב־$PathsArray$ כ־-1 (מסמל שלא קיים ממנה מסלול ל־end) ונעבור לאיטרציה הבאה.

אם רק אחת מהתחנות שמצאנו חוקית, סימן שהמסלול הארוך ביותר מהתחנה הנוכחית ל-end בהכרח עובר בתחנה היחידה שמצאנו.
במקרה זה, נעתיק את אורך המסלול הקיים ב-$PathsArray$ עבור התחנה שהוחזרה מ-$DriveInDirection$ (הבאה במסלול), נוסיף לו 1 ונשמור ב-$PathsArray$ באינדקס התחנה הנוכחית (שכן המסלול מהתחנה הנוכחית ל-end ארוך בתחנה אחת מהמסלול שמתחיל מהתחנה הפעילה הבאה אחריה).
אם שתי התחנות שמצאנו חוקיות, נשווה בין אורכי המסלולים המקסימליים ששמורים עבורן ב-$PathsArray$ באינדקסים המתאימים (גישה ב-$O(1)$).
עם התחנה שהמסלול ממנה ארוך יותר נבצע את אותן פעולות שאנו מבצעים במקרה הקודם (נאחסן את אורך המסלול ממנה ל-end בהוספת 1 בתא הנוכחי).

האיטרציה שאנו מבצעים היא בסדר הפוך ממיקום התחנות שלהן ב-$WestToEast$.
מסיבה זו ובגלל המיון של $WestToEast$, נקבל שכאשר אנו מבצעים את הפעולות שהגדרנו עבור תחנה נתונה, כל תחנה הנמצאת מזרחית או צפונית לה על המפה נמצאת באינדקס גדול יותר ב-$WestToEast$, ועקב כך היא כבר בעלת ערך מעודכן במערך $PathsArray$ (שכן כבר הפעלנו איטרציה עבור תחנה זו בעבר).
לכן כאשר אנו מסיימים לבצע את הפעולות שהגדרנו על התא ה-$start$ במערך $PathsArray$, נקבל בתא זה את אורך המסלול הארוך ביותר מ-$start$ ל-end, ואת אורך זה נחזיר למשתמש.
לבסוף נשחרר בסיבוכיות $O(n)$ את המערך $PathsArray$ באורך n שהקצינו לאורך הפעולה.
מכיוון שבהרכבת המסלול הפקנו את מספר התחנה הבאה ממזרח\מצפון באמצעות הפעולה $DriveInDirection$, נקבל שכל התחנות במסלול הן אכן תחנות פעילות בלבד (שכן $DriveInDirection$ לא מחזירה מספרי תחנות שאינן פעילות לפי האופן שבו מימשנו אותה).

סיבוכיות הפעולה:
על מנת ליצור ולאתחל את $PathsArray$ ביצענו מספר קבוע של פעולות עבור כל תא בו, כלומר הדבר התבצע בסיבוכיות $O(n)$.
לאחר מכן במהלך הרכבת המסלול אנו עוברים לכל היותר על כל התחנות (במקרה שבו כולן נמצאות בין $start$ ל-end), ועבור כל תחנה אנו מבצעים 2 קריאות ל-$DriveInDirection$ ומספר קבוע של פעולות השוואה והשמה עם ערכים שיש לנו גישה מיידית אליהם ב-$PathsArray$.
כלומר בסך הכול אנו מבצעים לכל היותר $2n$ פעולות $DriveInDirection$ שהסיבוכיות המשוערכת שלהן היא $O(\log^*(n))$, ולכן **לפי הגדרת סיבוכיות משוערכת** נקבל שאנו עושים זאת בסיבוכיות $2n \cdot \log^*(n) = O(n \cdot \log^*(n))$ **במקרה הגרוע**.
ולכן בסך הכול זו גם הסיבוכיות הכוללת של הפעולה.

סיבוכיות מקום:
בכל רגע נתון המבנה מאחסן מספר קבוע של מערכים שאורכם כמספר התחנות n.
בנוסף לאחר האתחול המבנה מכיל 2 מבני $MasterClose$, שכפי שראינו בתרגול סיבוכיות המקום שלהם מורכבת ממקום אילן היוחסין וכן מסיבוכיות מקום של $UnionFind$ במימוש "האופטימלי" שראינו בהרצאה בעל n מפתחות. בסך הכול זוהי סיבוכיות $O(n)$.
בפעולה $MaxPath$ אנו מקצים מערך $PathsArray$ באורך n, אך פרט לזאת אין הקצאות זיכרון נוספות או רקורסיות, ולכן סיבוכיות המקום הכוללת של המבנה היא $O(n)$.

שאלה 3

סעיף 1

הטענה נכונה. נוכיח זאת באינדוקציה על n:

<u>בסיס:</u>

עבור $n=1$ נקבל שקיימת הערימה בעלת איבר יחיד (למשל המספר 1).

אינדקס איבר זה הוא $r=1$ משום שזהו האיבר היחיד בסדרה הממוינת, וכן עומקו $h=0$ שכן זהו האיבר היחיד בעץ, ובפרט השורש.

מכאן שאכן מתקיים:

$$\lfloor\log_2(r)\rfloor = \lfloor\log_2(1)\rfloor = 0 \leq= d(a)$$

<u>צעד:</u>

נניח שעבור $n-1$ קיימת ערימת מינימום בעלת $n-1$ איברים (נסמנה H_1) הממומשת ע"י עץ כמעט שלהם שבה לכל איבר a בעומק $d(a)$ ואינדקס $r(a)$ מתקיים $\lfloor\log_2(r(a))\rfloor \leq$ $d(a)$.

נסמן את גובה ערימה זו h_1.

מכיוון שמספר האיברים בערימה זו סופי, קיים להן מקסימום שנסמנו m.

ניקח חוליה לעץ המכילה את המספר $m+1$ ונכניס אותה כעלה ימני בערימה H_1.

כלומר אם H_1 היא עץ שלם, נכניס את $m+1$ כעלה שמאלי ביותר בעומק h_1 (כלומר ברמה חדשה).

אחרת, נכניס את $m+1$ מימין לעלה הימני ביותר הנוכחי ב־H_1 (נוכל לעשות זאת אם H_1 אכן אינה עץ שלם), באותה רמה.

כעת, המבנה החדש שנסמן H_2 הוא אכן ערימת מינימום, שכן מכיוון שהכנסנו את המקסימום החדש $m+1$ כעלה ולא שינינו את סדר שאר האיברים, תנאי הערימה אינו מופר (המקסימום גדול מכל שאר האיברים ובפרט גדול מכל אב קדמון שלו).

מכיוון שהוספנו איבר גדול מכולם, לא שינינו את האינדקסים של שאר האיברים, וכן מהאופן שבו הוספנו אותו לא שינינו את עומקם.

כלומר כל האיברים פרט ל־$m+1$ מקיימים את הטענה מהנחת האינדוקציה.

נותר להראות ש־$m+1$ מקיים את התנאי:

אם בהכנסת $m+1$ הוספנו רמה חדשה לעץ, נקבל ש־H_1 היה עץ שלם, כלומר מספר הצמתים בו מקיים:

$$n-1 = 2^{h_1+1}-1 = 2^{h_2}-1$$

ולכן:

$$\begin{aligned} r(m+1) &= n = 2^{h_2} \\ &\Rightarrow \lfloor\log_2(r(m+1))\rfloor = h_2 = d(m+1) \end{aligned}$$

כלומר $m+1$ מקיים את התנאי.

אחרת, אם בהכנסת $m+1$ לא הוספנו רמה חדשה לעץ, מתקיים:

$$r(m+1) = n \leq 2^{h+1} - 1 < 2^{h+1}$$

שכן 2^{h+1} הוא מספר הצמתים בעץ בינארי שלם בגובה h.
ולכן נקבל:

$$\begin{aligned}&\log_2(r(m+1)) < h+1 \Rightarrow \\ &\qquad \Rightarrow \lfloor \log_2(r(m+1)) \rfloor \leq h = d(m+1)\end{aligned}$$

כלומר $m+1$ אכן מקיים את התנאי.

כלומר בסך הכול קיבלנו שכל האיברים בערימה החדשה H_2 מקיימים את התנאי, ולכן לפי עיקרון האינדוקציה הטענה נכונה לכל $n \in \mathbb{N}$.

סעיף 2

הטענה נכונה.

נממש את מבנה הנתונים באמצעות עץ AVL, שמפתחותיו יהיו האיברים בערימה, ובכל צומת הערך השמור יהיה מספר האיברים בערימה בעלי אותו המפתח.

כלומר כל ערך של איבר בערימה מופיע פעם אחת בלבד כצומת ב־AVL, וכפילויות ערכים בערימה מיוצגות באמצעות העלאת הערך בצומת המתאים בלבד.

מכאן שמספר האיברים בעץ ה־AVL בפועל הוא לכל היותר מספר האיברים השונים בערימה, כלומר $\log(n)$ על פי הנתון.

כמו כן, נשמור במשתנה מקומי את המפתח המינימלי בעץ, כלומר את ערך האיבר המינימלי בערימה.

נתאר את הפעולות ונוכיח את הסיבוכיות שלהן:

<u>$Insert$:</u>

נחפש את האיבר שרוצים להכניס בעץ ה־AVL.

סיבוכיות החיפוש היא לוגריתמית בגובה העץ, ולכן $O(\log(\log(n)))$.

אם מצאנו איבר בעל מפתח זה, נגדיל את הערך השמור בו ב־1.

אחרת, נכניס לעץ ה־AVL את המפתח החדש עם הערך 1 (מסמל שקיים רק איבר אחד מערך זה בערימה).

סיבוכיות ההכנסה היא לכל היותר $O(\log(\log(n)))$ לוגריתמית בגובה העץ, ולכן זוהי גם סיבוכיות הפעולה כולה.

<u>$DeleteMin$:</u>

ניגש למשתנה המקומי ששומר את ערך האיבר המינימלי בערימה ונחפש את האיבר שרוצים למחוק בעץ ה־AVL.

בדומה לעיל, סיבוכיות החיפוש היא $O(\log(\log(n)))$.

על מנת לדמות הוצאה מהערימה, נקטין את הערך של הצומת שמצאנו ב־1.

אם הערך הגיע ל־0 כתוצאה מההקטנה, סימן שלא נותרו איברים מערך זה בערימה.

במקרה זה נוציא את הצומת מהעץ בסיבוכיות $O(\log(\log(n)))$.

לאחר מכן יש למצוא מהו המינימום החדש בערימה, כלומר מהו המפתח המינימלי בעץ:

נבצע זאת על ידי ירידה בעץ, כאשר בכל פעם נרד לבן השמאלי של הצומת הנוכחי החל מהשורש.

ממבנה עץ AVL מובטח לנו שהצומת האחרון שנוכל לרדת אליו הוא המינימום החדש, ולכן נעדכן את המפתח שלו אל תוך המשתנה המקומי שאנו שומרים.

סיבוכיות חיפוש זה היא ליניארית בגובה העץ, שכן על כל צומת הגדלנו את עומק החיפוש ב־1 וביצענו $O(1)$ פעולות.

כלומר חיפוש המינימום החדש, ועקב כך כל הפעולה, מתבצעים ב־$O(\log(\log(n)))$ כדרוש.

<u>$FindMin$:</u>

מכיוון שאנו שומרים במשתנה מקומי את ערך האיבר המינימלי, נחזיר אותו ב־$O(1)$.

<u>$MakeHeap$:</u>

ניצור את הערימה על ידי יצירת עץ AVL עם כל הערכים, כאשר בכל הכנסה של ערך אנו ראשית מחפשים אותו בעץ, ואם הוא כבר קיים מגדילים את ערך הצומת הנמצא ב־1 במקום להכניס איבר חדש.

כלומר זהו אלגוריתם שקול לאלגוריתם לאתחול עץ AVL בעל $\log(n)$ צמתים שראינו בהרצאה, ולכן הסיבוכיות שלו היא:

$$O(\log(n) \cdot \log(\log(n)))$$

בנוסף, מתקיים:

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

$$\lim_{n\to\infty} \frac{\log^2(n)}{n} = \lim_{n\to\infty} \frac{2 \cdot \log(n) \cdot \frac{1}{n}}{1} =$$
$$= \lim_{n\to\infty} \frac{2 \cdot \log(n)}{n} = \lim_{n\to\infty} \frac{2 \cdot \frac{1}{n}}{1} = 0$$

כלומר לפי ההגדרה $\log^2(n) = o(n)$ ובפרט הסיבוכיות לפעולה זו מקיימת:

$$\log(n) \cdot \log(\log(n)) = O(n)$$

$\underline{DecKey(p, x)}$:

נחפש צומת בעל המפתח p, ואם הוא לא נמצא נחזיר שגיאה.

נקטין את ערך הצומת ב־1 (שכן בערימה יש איבר אחד פחות בעל ערך זה לאחר ההקטנה), ואם ערכו הגיע ל־0 נוציא את הצומת.

לאחר מכן נבצע $Insert(x)$ על מנת להוסיף לערימה מחדש את האיבר שמפתחו שונה ל־x.

סיבוכיות חיפוש והוצאה אפשרית בעץ AVL בעל $\log(n)$ צמתים היא כאמור $\log(\log(n))$ וזו גם הסיבוכיות של $Insert$ כפי שמימשנו.

לכן סיבוכיות הפעולה היא $O(\log(\log(n))$ ובפרט $O(\log(n))$

כלומר סיבוכיות $Insert, DeleteMin$ הן כנדרש, וכן סיבוכיות שאר הפעולות נותרו כשהיו.

סיבוכיות המקום של המבנה היא סיבוכיות מקום של עץ AVL בעל $\log(n)$ צמתים, כלומר $O(\log(n))$.

סעיף 3

הטענה אינה נכונה.

נפריך באמצעות כך שנראה שמנכונות הטענה נובע שקיים מיון מבוסס השוואות בסיבוכיות $o(n\log(n))$:

נניח בשלילה שקיים מבנה מהסוג שבשאלה.

יהא A מערך כלשהו באורך n שנרצה למיין.

נאתחל מאיברי A ערימת מינימום הממומשת כעץ כמעט שלם לפי האלגוריתם שראינו בהרצאה, שסיבוכיותו $O(n)$.

לאחר מכן נאתחל מערך ריק B שאורכו n (אתחול ב־$O(n)$).

לאחר מכן נבצע למבנה בשאלה $Init$ ב־$O(1)$.

לאחר מכן נבצע איטרציה לפי הסדר מ־1 עד n, נקרא ל־$Select(k)$ על כל איבר לפי סדר עולה ונכניס אותם ל־B.

בסיום התהליך נקבל לפי ההגדרה ש־B הוא המיון של המערך A.

נחשב את סיבוכיות האיטרציה:

בכל שלב אנו מבצעים $O(1)$ פעולות השמה וכן קריאה ל־$Select(k)$ ב־$O(h_k+1)$.

לפי הגדרת הערימה לפי מערך A, אנו קוראים ל־$Select(k)$ עבור כל איבר בערימה פעם אחת בדיוק.

לכן מכיוון שהערימה ממומשת כעץ כמעט שלם, הסיבוכיות של הלולאה כולה נתונה על ידי הסכום:

$$\sum_{k=1}^{n}(h_k+1)=n+\sum_{k=1}^{n}h_k\leq$$

בהרצאה ראינו שסכום גובהי האיברים בערימה הממומשת כעץ כמעט שלם בעל n איברים מקיים:

$$\leq n+n=2n$$

ולכן סיבוכיות הלולאה היא $O(n)$, וזו גם סיבוכיות המיון כולו.

קיבלנו סתירה, שכן מיינו מערך במיון מבוסס השוואות ללא הנחות נוספות על הקלט ב־$O(n)$, ובפרט ב־$o(n\log(n))$, וזו סתירה לפי ההוכחה שלא קיים מיון כזה שראינו בהרצאה.

מכאן שהטענה שגויה, כלומר לא קיים מבנה נתונים כנדרש.

שאלה 5

סעיף א'

נממש את המבנה באמצעות עץ $Trie$ כפי שראינו בהרצאות. נכנה עץ זה $Strings$.

בנוסף, בסוף כל מחרוזת בעץ (בעלה $) נשמור עותק של המחרוזת ואת מספר ההופעות שלה בעץ.

בנוסף, נשמור רשימה מקושרת של תווים לאחסון המחרוזת הבאה להכנסה. נשמור מצביעים לראש ולסוף רשימה זו. נכנה אותה $NextString$.

בנוסף נשמור משתנה מקומי $Common$ שיכיל מצביע לעלה המחרוזת הנפוצה ביותר במבנה.

להלן מימוש הפעולות:

:$\underline{Init}$

נאתחל עץ $Trie$ ורשימה ריקים ב־$O(1)$ כפי שראינו בהרצאה.

את $Common$ נאתחל ב־$O(1)$ ל־$NULL$ כערך לא חוקי כלשהו.

כאמור סיבוכיות הפעולה הכוללת היא $O(1)$.

:$\underline{Push(c)}$

נוסיף את התו c לסוף הרשימה $NextString$. נוכל לעשות זאת ב־$O(1)$ מכיוון שאנו שומרים מצביע לסוף הרשימה.

:$\underline{PopString()}$

נמיר בלולאה את הרשימה $NextString$ למחרוזת בסיבוכיות $O(n)$ כאשר n הוא מספר האיברים ברשימה, ולאחר מכן נכניס את המחרוזת ל־$Strings$.

לפי האלגוריתם שראינו בהרצאה, ההכנסה היא ב־$O(|s|)$ כאשר s היא המחרוזת השמורה ב־$NextString$.

אם במהלך ההכנסה הגענו לתו $ קיים, כלומר המחרוזת קיימת כבר במבנה ־ ההכנסה תתבצע על ידי העלאת מונה מספר ההופעות שלה בעץ ב־1.

בנוסף, נבצע השוואת מחרוזות בין המחרוזת שמכניסים לבין המחרוזת המוצבעת על ידי $Common$, לפי המידע שאנו מאחסנים בעלה של כל אחת מהן.

<u>אם מספר ההופעות של המחרוזת המוכנסת זהה למספר ההופעות של המחרוזת השכיחה,</u> נשווה ביניהן לקסיקוגרפית (השוואה לקסיקוגרפית איטרטיבית מתבצעת ב־$O(|s|)$, ליניארית באורך המחרוזת המוכנסת, שכן כשמגיעים לסופה מפסיקים את ההשוואה גם אם המחרוזת השנייה ארוכה יותר).

לאחר ההשוואה, אם המחרוזת החדשה קטנה יותר לקסיקוגרפית ־ נשנה את $Common$ כך שיצביע על העלה שלה, אחרת לא נשנה אותו.

<u>אם מספר ההופעות של המחרוזת המוכנסת גדול ממספר ההופעות של המחרוזת השכיחה,</u> נבצע ל־$Common$ השמה לעלה המחרוזת החדשה.

קיבלנו שלאחר ההכנסה $Common$ אכן מצביע למחרוזת השכיחה ביותר במבנה לאחר השינוי.

לבסוף נהרוס את הרשימה $NextString$ ב־$O(n)$ אורך הרשימה ונאתחל את המצביעים שלה להצביע לרשימה ריקה.

בסך הכול הסיבוכיות במקרה הגרוע של פעולה זו מורכבת מסיבוכיות הכנסת מחרוזת באורך $|s|$ לעץ $Trie$, השוואה לקסיקוגרפית שלה עם מחרוזת אחרת, הריסת רשימה שאורכה כאורך המחרוזת המוכנסת, ומספר קבוע של פעולות השוואת משתנים ב־$O(1)$.

לכן בסך הכול הסיבוכיות במקרה הגרוע של הפעולה היא $O(|s|)$ כאשר s היא המחרוזת החדשה במבנה.

נוכיח כעת שהסיבוכיות של 2 הפעולות האחרונות היא $O(1)$ משוערכת:
נשתמש בשיטת הפוטנציאל:
נגדיר את Φ_i הפוטנציאל לאחר i פעולות $Push$ ו־$PopString$ להיות אורך הרשימה $NextString$.
מכיוון ש־$Push$ מגדילה ב־1 את אורך $NextString$ ומכיוון שהסיבוכיות במקרה הגרוע שלה היא $O(1)$ נקבל שמחירה המשוערך הוא:

$$1 + (\Phi_i - \Phi_{i-1}) = 2$$

מכיוון ש־$PopString$ מתבצעת בסיבוכיות ליניארית באורך המחרוזת $NextString$, אם נסמן את המחיר בפועל שלה כ־$|s|$ נקבל שמחירה המשוערך הוא:

$$|s| + (\Phi_i - \Phi_{i-1}) = |s| + (0 - |s|) = 0$$

שכן הפעולה מאפסת את הרשימה $NextString$ לאורך 0 מאורך $|s|$.

בסך הכול קיבלנו שמחיר משוערך של הפעולה ה־i בסדרת m פעולות מקיים $a_i \leq 2$.
ולכן אם נסמן t_i את המחיר בפועל של הפעולה ה־i כפי שראינו בתרגול נקבל שמחיר סדרת הפעולות כולה הוא:

$$\sum_{i=1}^{m} t_i \leq \sum_{i=1}^{m} t_i + \Phi_m - \Phi_0 = \sum_{i=1}^{m} (t_i + \Phi_i - \Phi_{i-1}) =$$
$$= \sum_{i=1}^{m} a_i \leq 2 \cdot m$$

כלומר הסיבוכיות של כל סדרת m פעולות $Push$ ו־$PopString$ היא $O(m \cdot 1)$ במקרה הגרוע, ולכן לפי ההגדרה הסיבוכיות המשוערת של 2 הפעולות היא $O(1)$.

<u>סיבוכיות מקום:</u>
סיבוכיות המקום של עץ מחרוזות $trie$ שראינו בהרצאה היא ליניארית בסכום אורכי המחרוזות.
בעץ שבמבנה אנו שומרים בנוסף בכל עלה מספר ואת המחרוזת שעלה זה מייצג, אך הדבר לא משנה את סיבוכיות המקום של העץ מכיוון שקיימת התאמה בין העלים בעץ לבין המחרוזות.
בנוסף סיבוכיות המקום של הרשימה $NextString$ היא ליניארית באורך הרשימה.
בסך הכול סיבוכיות המקום של המבנה היא $O(s + |n|)$ כאשר s הוא סכום אורכי המחרוזות במבנה ו־n הוא אורך המחרוזת במחסנית בכל רגע נתון.

סעיף ב'

נוסיף למבנה משתנה $LastSuffixTree$ ששומר עץ סיומות עבור המחרוזת האחרונה שהכנסנו לעץ.

את $LastSuffixTree$ נבנה באמצעות אלגוריתם הקופסה השחורה שהוצג בהרצאה בסוף כל פעולת $PopString$ כאשר ידועה לנו המחרוזת שאנו מכניסים למבנה.

(על מנת למנוע דליפות זיכרון עתידיות ניתן לשמור את עץ הסיומות הישן בעלה של המחרוזת שאותה הוא מייצג אם נשמור אליה מצביע כל עוד זאת המחרוזת האחרונה שהכנסנו לעץ. נציין זאת רק לצורך השלמות, אך הדבר אינו רלוונטי לשאלה שכן לא נתבקשנו להרוס את המבנה).

בנוסף, נבצע סיור "$PostOrder$" על $LastSuffixTree$ לשמירת מידע נוסף:

עבור כל צומת, נשמור בו כמידע נוסף את מספר העלים בתת העץ שהוא שורשו.

כלומר עבור העלים (שבהם אנו עוברים בסיור לפני האבות שלהם) נאחסן את הערך 1.

כאשר אנו עולים לצומת כלשהו במהלך הסיור, נשלוף ב־$O(1)$ את המידע השמור בכל אחד מהבנים הניתנים לגישה ממערך הקשתות באורך Σ, ונסכום את המידע הנוסף שבכל אחד מהם.

נקבל שמספר העלים בתת העץ של הצומת הנוכחי הוא אכן סכום מספרי העלים בתתי העצים ששורשיהם הבנים של הצומת.

סיבוכיות האלגוריתם ליצירת עץ סיומות שראינו היא ליניארית באורך המחרוזת, וכן ראינו בהרצאה שהעץ המתקבל הוא בעל מספר צמתים ליניארי באורך המחרוזת.

לכן גם סיבוכיות הסיור היא ליניארית באורך המחרוזת, שכן עבור כל צומת אנו מבצעים מספר קבוע של פעולות השמה ב־$O(1)$ (משום שהאלף בית Σ מגודל קבוע).

בסך הכול סיבוכיות הפעולות שהוספנו ל־$PopString$ הוא ליניארי באורך המחרוזת המוכנסת לעץ, וסיבוכיות $PopString$ הקודמת גם היא ליניארית ־ לכן אנו לא פוגעים בסיבוכיות המשוערכת שלה.

מימוש הפעולה החדשה:

<u>$CountSubstring(r)$:</u>

נסייר מהשורש בעץ $LastSuffixTree$ עד שנמצא את המחרוזת r המבוקשת באלגוריתם חיפוש תת מחרוזת בעץ סיומות שראינו בהרצאה (אם לא נמצא אותה נחזיר שגיאה).

כפי שראינו בהרצאה, סיבוכיות החיפוש היא $O(|r|)$ במקרה הגרוע.

אם מצאנו את המחרוזת, ייתכן שהיא הסתיימה באמצע קשת ולא בצומת בעץ. במקרה זה נתקדם לצומת הבא שלאחר קשת זו ב־$O(1)$.

קיבלנו שהמסלול מהשורש עד לצומת שמצאנו מייצג תת מחרוזת r_2 כלשהי.

ומכיוון שמסלול זה התחיל ממסלול מהשורש עד r, נקבל מהתכונה של עץ סיומות שראינו בהרצאה, שכל מופע של r כתת מחרוזת הוא בתוך מופע של r_2, וכן שבתוך כל מופע שונה של r_2 קיים מופע של r.

בסך הכול, מספר המופעים של r_2 (ולכן של r) כתת מחרוזת שווה למספר העלים בתת העץ שמצאנו, מכיוון שכל עלה מציין סיומת שונה שמתחילה ב־r_2 ובפרט ב־r.

מכאן שהמידע הנוסף ששמרנו בצומת של r_2 בעת יצירת העץ נתון לקריאה ב־$O(1)$ ומכיל את הערך הרצוי, שנחזיר למשתמש.

בסך הכול סיבוכיות הפעולה הוא החיפוש של r כתת־מחרוזת בעץ הסיומות, שראינו בהרצאה שמתבצע ב־$O(|r|)$ ומספר סופי של פעולות ב־$O(1)$, כנדרש.

<u>סיבוכיות מקום:</u>

הוספת המידע הנוסף לכל צומת בעץ המחרוזות לא שינתה את הסיבוכיות שלו, שכן סיבוכיות המקום של המידע הנוסף היא $O(1)$ בכל צומת.

בנוסף עבור כל מחרוזת שנכנסת למבנה אנו בונים עץ סיומות ולא הורסים אותו.

בהרצאה ראינו שסיבוכיות המקום של עץ סיומות למחרוזת היא ליניארית באורך המחרוזת.

לכן בדומה להוספת עותק של המחרוזת בסעיף הקודם, שמירת עצי הסיומות אינה משנה את סיבוכיות המבנה.

בסך הכול סיבוכיות המקום הכוללת נותרת $O(s+|n|)$ כמו בסעיף הקודם.

שאלה 2

מבנה הנתונים שלנו יכיל 2 מערכים A, B שעובדים בשיטת $Double\ Hashing$ שלמדנו בקורס, כך שעבור כל מערך נגדיר סדרת פונקציות ערבול שמתאימות לו. בכל עת, אחד המערכים ייקרא **המערך הראשי** והמערך השני ייקרא **המערך המשני** (יובהר טוב יותר בהמשך). סדרת פונקציות הערבול עבור כל מערך יוגדרו כפי שראינו בקורס:

- נגדיר $h_A(x) = x\ mod\ |A|, h_B(x) = x\ mod\ |B|$ כאשר $|A|, |B|$ הם גדלי המערכים A, B בהתאמה.
- נגדיר $r_A(x) = 1 + \big(x\ mod\ (|A| - c)\big), r_B(x) = 1 + (x\ mod\ (|B| - c))$ עבור $c > 0$ קטן כלשהו (לדוגמא ניקח $c = 3$).
- לבסוף נגדיר את סדרת פונקציות הערבול להיות:
  - $h_{k-A}(k) = \big(h_A(x) + k \cdot r_A(x)\big)\ mod\ |A|$
  - $h_{k-B}(k) = \big(h_B(x) + k \cdot r_B(x)\big)\ mod\ |B|$

על מנת להשתמש בפונקציות הערבול בצורה פשוטה, נגדיר במבנה שלנו את הפעולה (הפונקציה):

$$h(x, m) = x\ mod\ m$$

ובנוסף נגדיר במבנה את הפעולה (הפונקציה):

$$r(x, m) = 1 + \big(x\ mod(m - 3)\big)$$

ככה שעבור כל k וגודל מערך כלשהו נוכל לקרוא לפונקציות אלו ולקבל את תוצאת הערבול בקלות.
2 פעולות אלו מתבצעות בסיבוכיות זמן $O(1)$ (חישוב mod מתבצע ב-$O(1)$).

בנוסף, נצטרך לשמור משתנים ולציין נקודות חשובות נוספות:

- נגדיר 2 משתנים עבור כל מערך, שישמרו את כמות האיברים הנמצאים בכל מערך ואת הגודל הכולל של כל מערך, כל שנוכל לשלוף כל אחד מהם בסיבוכיות זמן $O(1)$ ונעדכן אותם כאשר אנו מגדילים את גודל המערך ו\או מכניסים איברים למערך.
- בכל פעם שנכניס איבר למבנה הנתונים, ההכנסה תתבצע רק לאחד מן המערכים, ולכן נשמור דגל בשם $current$ שיגיד לנו לאיזה מערך נדרש לבצע את ההכנסה (המערך הראשי). עבור כל הכנסה של איבר חדש למבנה, אנו נכניס את האיבר החדש למערך הראשי, ובנוסף נעביר איבר מהמערך המשני למערך הראשי, כך שבסה"כ בכל הכנסה נכניס 2 איברים למערך הראשי. אם המערך הראשי יהיה מלא בזמן ההכנסה, אנו נגדיל את המערך המשני ונהפוך אותו להיות הראשי, כך שההכנסות הבאות יהיו אליו, ונעדכן את $current$ כך שיגיד לנו מיהו המערך הראשי בכל עת. $current$ יכול להכיל את הערך "A" או את הערך "B", המייצגים מי הוא המערך הראשי הנוכחי.
- כפי שאמרנו קודם, בכל פעם שאנו מכניסים איבר חדש למערך הראשי, אנו מעבירים איבר נוסף מהמערך המשני אל המערך הראשי (כל פעם את האיבר הבא שנמצא בקצה המערך המשני). לכן, נשמור משתנה $iterator$ ששומר את האינדקס של האיבר הבא שיש להוציא מהמערך המשני, הנדרש להעברה למערך הראשי. בכל פעם שנעביר איבר מהמערך המשני אל המערך הראשי, נקטין את $iterator$ ב-1 כדי שיכיל את האינדקס של האיבר הבא שיש להעביר מהמערך המשני אל המערך הראשי.

כעת, נתאר את מהלך הפעולות על המבנה:

- Init(): נאתחל את מבנה הנתונים על ידי אתחול המערך A לגודל קבוע (לדוגמא 100), ונעדכן שגודל המערך A הוא 100 וכמות האיברים בו היא 0 בסיבוכיות $O(1)$ (אתחול קבוע והשמה פשוטה למשתנים). נאתחל את B להיות באורך 200, ונעדכן שגודל המערך B הוא 200 וכמות האיברים בו היא 0 בסיבוכיות $O(1)$ (אתחול קבוע והשמה פשוטה). נעדכן את $current$ להיות "A" בסיבוכיות $O(1)$ כדי לציין שהמערך A הוא המערך שמבצעים אליו הכנסות בעת (המערך הראשי). לבסוף, נאתחל את $iterator$ ל-1- בסיבוכיות זמן $O(1)$, כדי לציין את

העובדה שאין אף איבר ב-B שצריך להעביר ל-A עבור ההכנסות החדשות העומודת לבוא. סה"כ סיבוכיות הזמן והמקום הם $O(1)$.

- Insert(x): נבדוק את הערך הנמצא בדגל $current$. נניח בה"כ שהערך הוא "A", ולכן ההכנסה ככל הנראה צריכה להתבצע למערך A (אלא אם A מלא ואז ההכנסה תהיה ל-B). נבדוק מה הוא הגודל הכולל של A וכמה איברים נמצאים כעת ב-A בסיבוכיות זמן $O(1)$ (אנו שומרים את ערכים אלו במשתנים). נחלק למקרים:
  - **אם A אינו מלא (אם הכמות הנוכחית של האיברים בו קטנה מגודלו – נגלה זאת בסיבוכיות זמן $O(1)$ כי אנו שומרים את גדלים אלו):**
    - **תחילה נציין כי ב-A יש לפחות 2 מקומות פנויים. הסיבה לכך היא שאנו מתחילים כאשר A הוא בגודל זוגי, ובכל פעם אנו מכניסים אליו 2 איברים. כאשר A מלא אנו מגדירים את B להיות פעמיים גודלו של A, וכאשר B מלא אנו מגדירים את A להיות פעמיים הגודל של B, ולכן A גדל בכפולות של 2.**
    - נכניס את האיבר x למערך A על ידי שימוש בפונקציה
$$h_k(x) = \big(h(x,|A|) + k \cdot r(x,|A|)\big) mod |A|$$
כפי שלמדנו בקורס, תוך קידום של k עד אשר לא תהיה התנגשות איברים. בממוצע על הקלט, סיבוכיות הזמן להכנסה היא $O(1)$ כפי שלמדנו.
    - נגדיל ב-2 את המשתנה ששומר את כמות האיברים במערך A כי הוספנו איבר חדש ל-A. ההגדלה מתבצעת בסיבוכיות זמן $O(1)$.
    - אם $iterator$ מצביע על אינדקס חוקי, כלומר אם הוא גדול או שווה ל-0, נעביר את האיבר הנמצא באינדקס $iterator$ במערך B אל המערך A על ידי מחיקתו מ-B והכנסתו ל-A תוך שימוש בפונקציה
$$h_k\left(x_{from-B}\right) = \Big(h\left(x_{from-B},|A|\right) + k \cdot r\left(x_{from-B},|A|\right)\Big) mod |A|$$
ובמקרה של התנגשות נקדם את k עד אשר לא תהיה התנגשות. נחסר 1 מ-iterator כדי שיצביע על האינדקס של האיבר הבא שיש להעביר מ-B ל-A. בממוצע על הקלט, סיבוכיות הזמן למחיקה והכנסה היא $O(1)$ כפי שלמדנו.

  הכנסת איבר חדש ל-A והעברת איבר מ-B ל-A מתבצעים בסיבוכיות זמן $O(1)$ בממוצע על הקלט, כפי שהראינו, ובסיבוכיות מקום $O(1)$ כי לא הקצינו זיכרון דינמי חדש.

  - **אחרת, אם A מלא, כלומר אם כמות האיברים ב-A זהה לגודל המערך A, זה אומר שמערך B ריק – כלומר אין בו אף איבר ששייך למבנה (נוכיח בהמשך – (1)):**
    - נמחק את מערך B בסיבוכיות זמן $O(1)$, כי הוא ריק.
    - נאתחל מחדש את מערך B להיות בגודל $2|A|$ בסיבוכיות זמן $O(1)$ כאשר $|A|$ הוא גודל המערך A (הסיבוכיות היא $O(1)$ כאשר נשתמש בשיטה של אתחול מערכים ב-$O(1)$ שלמדנו בקורס).
    - נעדכן את המשתנה השומר את גודל המערך B להיות $2|A|$, ונעדכן את המשתנה ששומר את כמות האיברים ב-B להיות 2 כי אנו עומדים להכניס אליו 2 איברים חדשים. סיבוכיות הזמן לביצוע פעולות אלו הוא $O(1)$.
    - נכניס את האיבר x למערך B על ידי שימוש בפונקציה
$$h_k(x) = \big(h(x,|B|) + k \cdot r(x,|B|)\big) mod |B|$$
כפי שלמדנו בקורס. לא יכולה להיות התנגשות כי מערך B הוא מערך שאין בו אף איבר, ולכן נבחר ב-$k = 0$. סיבוכיות זמן ההכנסה אם כן היא $O(1)$.

- נעביר את הגודל $|A|-1$ למשתנה $iterator$ כדי לציין את העובדה שבהכנסה הבאה האיבר שאותו יהיה להעביר מ-A ל-B נמצא באינדקס $|A|-1$ במערך A.
- נמחק את האיבר הנמצא באינדקס $iterator$ במערך A, ונעביר אותו למערך B על ידי שימוש בפונקציה
$$h_k(x) = \big(h(x,|B|) + k \cdot r(x,|B|)\big) mod|B|$$
כפי שלמדנו בקורס. אם יש התנגשות עבור $k=0$, נשתמש ב-$k=1$. לכן סיבוכיות הזמן של ההכנסה היא $O(1)$.
- נחסר 1 מ-$iterator$, כך שיהיה שווה ל-$|A|-2$ כדי לציין את העובדה שבהכנסה הבאה האיבר שאותו יהיה להעביר מ-A ל-B נמצא באינדקס $|A|-2$ במערך A, ונקטין ב-1 את המשתנה ששומר את כמות האיברים הנמצאים ב-A מכיוון שהסרנו איבר 1 מ-A.

יצירת המערך B, הכנסת האיבר אליו והעברת איבר מ-A ל-B מתבצעים בסיבוכיות זמן $O(1)$. אם היה n איברים במבנה לפני יצירת B, סיבוכיות המקום היא $O(n)$, ולאחר יצירת B סיבוכיות המקום תישאר $O(n)$ כי גודלו של B הוא $2n$ והכנסנו בסה"כ איבר אחד נוסף למבנה.

- נוכיח את (1): לפי האלגוריתם שהצענו, נניח בה"כ ש-A מלא. אם A מלא וגודלו הוא $2n$, זה אומר שגודל המערך B הוא n, כי בכל פעם שמערך אחד מתמלא אנו מגדירים את גודלו של השני להיות פעמיים הגודל של המערך המלא. לכן, אם A מלא, זה אומר שיש בו $2n$ איברים, כי על מנת שהוא יתמלא אנו צריכים לבצע n פעולות הכנסה, כאשר כל אחת מהן מכניסה איבר חדש ל-A ומעבירה איבר נוסף מ-B ל-A. לאחר n פעולות הכנסה, אנו בעצם מוציאים n איברים מ-B, ולכן כאשר A מלא B ריק בוודאות.

❖ $find(x)$: נשלוף את גודלם של A ו-B, ועבור כל אחד מהם נחשב את פונקציית הערבול המתאימה:

$$h_A(x) = \big(h(x,|A|) + k \cdot r(x,|A|)\big) mod|A|$$

$$h_B(x) = \big(h(x,|B|) + k \cdot r(x,|B|)\big) mod|B|$$

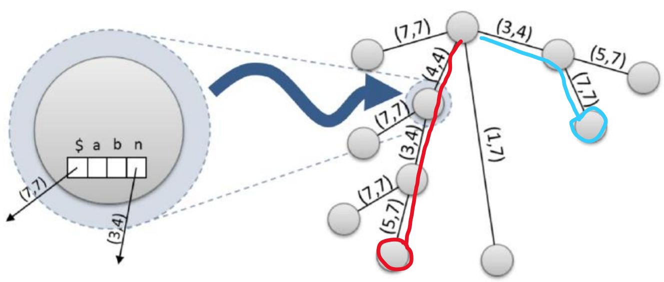
נחפש את x בכל אחד מהמערכים בסיבוכיות זמן $O(1)$ בממוצע על הקלט, כפי שלמדנו בקורס. אם מצאנו את x ב-1 מהם, נחזיר אותו.

<u>נוכיח את סיבוכיות המקום:</u>

נניח ש-2 המערכים במבנה הם מאורך $2n$ ו-n. אנו מבצעים הגדלה למערך מסוים רק אם המערך השני התמלא. לכך המערך באורך $2n$קיבל את אורכו לאחר מילוי המערך באורך n באיברים. מכיוון שאנו לא מבצעים הסרה של איברים, בכל עת במבנה יש לנו לפחות n איברים (כי אחרת לא היינו מבצעים הגדלה למערך השני להיות באורך $2n$). בנוסף, יש לנו לכל היותר $2n$ איברים במבנה כי אחרת היינו מבצעים הגדלה של המערך שאורכו n להיות באורך $2 \cdot (2n)$, וזו סתירה לכך שהגודל שלו הוא n. כלומר יש לנו לפחות n ולכל היותר $2n$ איברים במבנה. בנוסף, קיימים לנו כמות משתנים סופית ששומרת פרמטרים נוספים, ולכן עבור n איברים במבנה סיבוכיות המקום היא $O(n)$ כי אנו שומרים מערכים שאוכם הכולל הוא לכל היותר $3n$ ועוד כמות סופית של משתנים לא דינמיים.

שאלה 4

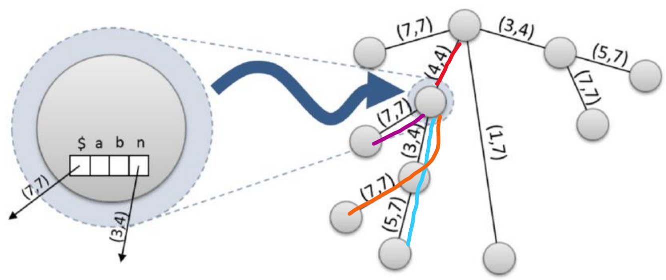
יהי עץ סיומות דחוס T המייצג את הסיומות של מחרוזת s כלשהי. אזי, כל קשת היוצאת משורש העץ מייצגת התחלה של סיומת בעץ, כך שכל הקשתות מייצגות את התחלתן של כל הסיומות הקיימות במחרוזת. בנוסף, כל עלה בעץ מייצג סיומת אחת בלבד, כפי שלמדנו. הקשת שבה האינדקסים הם (x, y) מייצגת את תת המחרוזת במחרוזת s, המתחילה בתו שהאינדקס שלו הוא x ומסתיימת בתו שהאינדקס שלו הוא y במחרוזת s. לכן כדי לקבוע את אורך של סיומת כלשהי נצטרך לסכום את אורכי כל תתי המחרוזות המובלות בסיומת ומיוצגים על ידי הקשתות הנמצאות במסלול של הסיומת. עבור כל צומת שהאינדקסים שלה הם (x, y), ההפרש $(y - x + 1)$ מייצג את כמות התווים של התת מחרוזת אותה הקשת מייצגת. אם $x = y$ נקבל שהאורך הוא 1, כלומר הקשת מייצגת תו בודד. אם $y = x + 1$ נקבל ש-$y - x + 1 = 2$ כלומר הקשת מייצגת 2 תווים, וכו'. לכן, נגדיר את **אורך המסלול של סיומת** כסכום ההפרשים $(y - x + 1)$ עבור כל קשת v שהאינדקסים בה הם (x, y), אשר נמצאת במסלול היוצא מהשורש ומסתיים בעלה המייצג את הסיומת. לדוגמא:



- אורך מסלול הסיומת של הסיומת המיוצגת על ידי העלה האדום הוא 6=(7-5+1)+(4-3+1)+(4-4+1). במקרה של $banana\$$ מספר זה מייצג את אורך הסיומת $anana\$$, שאורכה הוא 6.
- אורך מסלול הסיומת של הסיומת המיוצגת על ידי העלה הכחול הוא 3=(7-7+1)+(4-3+1). במקרה של $banana\$$ מספר זה מייצג את אורך הסיומת $na\$$ שהוא כמובן 3.

כעת נתאר את מהלך האלגוריתם:

- נבנה מערך של תווים באורך $|s|$ בשם $result$, אשר בסוף יכיל את המחרוזת s אותה אנו משחזרים. סיבוכיות הזמן לאתחול היא $O(1)$.
- לכל קשת היוצאת מהשורש אל בן שנסמנו son, המסמלת את תחילתה של סיומת המתחילה בתו שנסמנו c:
  - נבנה מערך $counts$ של מספרים באורך $|s| + 1$ בסיבוכיות זמן $O(1)$.
  - נבנה מצביע p למספר i שמייצג את כמות המספרים הנוכחית במערך, ונאתחל את i ל-0 כדי לציין את העובדה שבעת אין אף איבר במערך $counts$.
  - **נכניס ל-$counts$ את כל אורכי מסלולי הסיומות היוצאים מהשורש, עוברים דרך son, ומגיעים לעלה (נסביר בהמשך כיצד נבצע זאת – (1), ונראה שסיבוכיות הזמן לבצע זאת היא $O(|s|)$).** ייתכנו בסה"כ $|s| + 1$ עלים מכיוון שיש בסה"כ $|s| + 1$ סיומות, ולכן יהיו לכל היותר $|s| + 1$ מסלולים היוצאים מהשורש, העוברים דרך son, ומגיעים לעלה. כלומר, המערך $counts$ לכל היותר יתמלא, אך לא תיתכן שגיאה הקשורה לגודלו. המערך $counts$, המכיל בעת את כל אורכי מסלולי הסיומות, בעצם מכיל את כל אורכי הסיומות המתחילות בתו c. לדוגמא, עבור הקשת המסומנת באדום:



המערך $counts$ יכיל את המספרים 2,6,4, כאשר 2 מייצג את המסלול האדום ואז הסגול (אורך הסיומת $a\$$ המתחילה בתו a), 6 מייצג את המסלול האדום ואז התכלת (אורך הסיומת $anana\$$ המתחילה גם בתו a), ו-4 מייצג את המסלול האדום ואז הכתום (אורך הסיומת $ana\$$ המתחילה גם בתו a). אורכים אלו, כפי שציינתי קודם, מייצגים את כל האורכים של כל הסיומות המתחילות בתו a במחרוזת $banana\$$.

אורכי המסלולים מייצגים את אורכי הסיומות, ולכן נמקם את התו c באינדקסים הרלטיביים לסוף המערך $result$. כלומר, לכל מספר i הנמצא במערך $counts$, נמקם את התו c במקום $|s| - i$ במערך $result$. למשל עבור הדוגמא לעיל, $|s| = 7$ עבור $banana\$$, ולכן נמקם את התו a באינדקסים $|s| - 2, |s| - 4, |s| - 6$ שהם 5,3,1 במערך $result$. נקבל כעת ש-$result$ הוא:

$$?\,, a, ?\,, a, ?\,, a, ?$$

סימני השאלה מייצגים את התווים שעוד לא גילינו, אשר נגלה כאשר נעבור על הקשתות הנוספות היוצאות מהשורש.

- מכיוון שכל סיומת אפשרית במחרוזת מתחילה בקשת היוצאת מהשורש, אחרי שנעבור על כל הקשתות היוצאות מהשורש נקבל את המילה המשוחזרת ב-$result$, מכיוון שמילאנו את מערך זה בכל התחלות הסיומות של המחרוזת אותה אנו משחזרים – כלומר בכל התווים הנמצאים במחרוזת.

(1) נסביר כיצד נכניס ל-$counts$ את כל אורכי מסלולי הסיומות היוצאים מהשורש, עוברים דרך son, ומגיעים לעלה:
נעשה מעבר $post - order$ רקורסיבי על כל הבנים של son, כך שנעביר ברקורסיה את המערך $counts$, את המצביע p שמצביע למספר המייצג כמה איברים נמצאים במערך $counts$, ואת אורך מסלול הסיומות הנוכחי. בכל בן ברקורסיה נוסיף לאורך מסלול הסיומות את התרומה של הבן לאורך המסלול, וכאשר נגיע לעלה (תנאי העצירה שלנו) נוסיף את תרומתו לאורך המסלול, נכניס את הסכום למקום הבא במערך $counts$, ונקדם את הערך שאליו מצביע p ב-1, כדי לציין שגודל המערך גדל ב-1 ועל מנת שהעלה הבא יכניס את אורך המסלול אליו הוא שייך לאינדקס הבא במערך $counts$. כשנסיים את תהליך ה-$post - order$ ונחזור מהרקורסיה son, נוסיף את אורך הסיומת המיוצגת על ידי הקשת היוצאת מהשורש אל son לסכום עבור כל איברי המערך $counts$ (כלומר נוסיף את תרומתה של הקשת היוצאת מהשורש אל son אל כל אורכי הסיומות היוצאים מקשת זו) בסיבוכיות זמן $O(|s|)$ (סריקה לינארית ועדכון הערכים) ונחזור אל השורש. הוכחנו בקורס שכמות הצמתים בעץ סיומות של מחרוזת s היא $O(|s|)$, ולכן בסה"כ נקבל שסיבוכיות הזמן לסריקה $post - order$ היוצאת מכל בן של השורש היא $O(|s|)$.

לשורש יש מספר בנים שהוא לכל היותר מספר קבוע - $\Sigma + 1$, ולכן לבצע את כל סריקות ה-$post - order$ עבור כל אחד מהבנים מתבצע בסיבוכיות זמן $O(|s|)$ כי כל סריקת $post - order$ בעצמה מתבצעת בסיבוכיות זמן $O(|s|)$, והרי שיש לנו מספר קבוע של סריקות כאלו.
לסיכום, במחרוזת $result$ תימצא המחרוזת המשוחזרת, וביצענו את האלגוריתם בסיבוכיות זמן $O(|s|)$. בסה"כ הקצינו מערך חדש באורך $|s| + 1$ ועומק הרקורסיה הוא לכל היותר $|s| + 1$ במהלך ריצת האלגוריתם, כי אורך הסיומת הארוכה ביותר היא אורך המחרוזת עצמה, $|s|$. לכן סיבוכיות המקום גם תהיה $O(|s|)$.