דף שער

מבני נתונים 1
234218

תרגיל יבש 2

מספר

הוגש ע"י:

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

ציון:

לפני בונוס הדפסה:

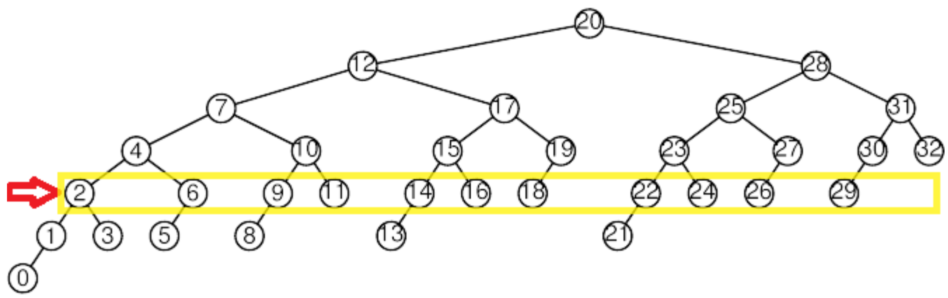
כולל בונוס הדפסה:

נא להחזיר לתא מס':

שאלה 1

סעיף א'

הטענה אינה נכונה. להלן דוגמה נגדית:



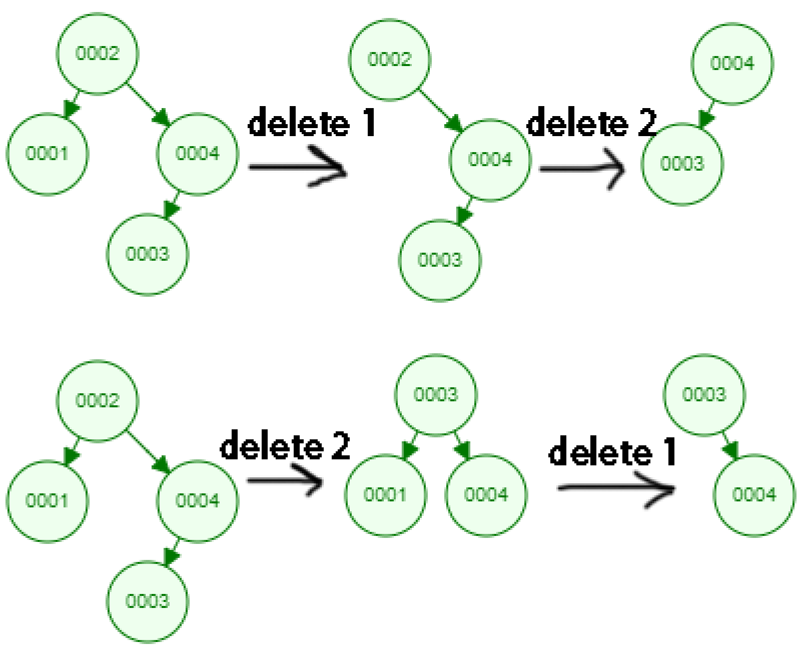
העץ שלעיל הוא אכן עץ AVL (זהו עץ פיבונאצ'י מסוג H_6), שורשו הוא בעל המפתח 20, גובהו $h = 6$, אך הרמה $h - 2 = 4$ אינה מלאה, שכן יש בה 11 צמתים, ואילו $2^4 = 16$.

סעיף ב'

הטענה אינה נכונה. להלן דוגמה נגדית:

נסתכל על העץ הבא, כאשר x הוא הצומת 1 ו־y הוא הצומת 2:

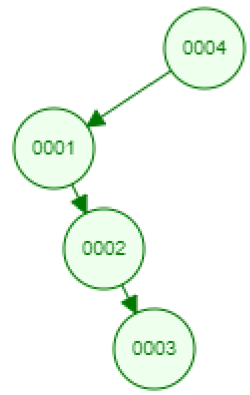
להלן 2 הרצפים האפשריים של הוצאה של הצמתים הללו:



ברור כי שני העצים המתקבלים בסוף התהליך אינם זהים.

סעיף ג'

הטענה אינה נכונה. להלן דוגמה נגדית:



אם ניקח את המסלול מ־3 ל־4, שאכן מקיימים $3 < 4$, נקבל שהמסלול הוא:

$$3 - 2 - 1 - 4$$

וברור כי זהו אינו מיון לפי סדר עולה, שכן $1 < 2$ אך 2 מופיע לפני 1 במסלול.

סעיף ד'

הטענה אינה נכונה.
נפריך על ידי כך שנראה ש־$D(n) \to \infty$:
יהא $n \in \mathbb{N}$.
נבנה 2 עצי 3־2 באופן הבא:
יהא a_n הטבעי הגדול ביותר המקיים:

$$\sum_{x=0}^{a_n} 2^x = 2^{a_n+1} - 1 \leq n$$

מכיוון ש־$2^{a_n+1} - 1$ הוא סכום חזקות של 2, קיים עץ בינארי שלם, שבפרט ניתן למלא בו ערכים כך שיהיה עץ 3־2, בעל מספר צמתים זה, שנסמנו A_n.
כמו כן, הגדרנו את a_n כך ש: $2^{a_n+1} - 1 \leq n$, ולכן ב־A_n יש לכל היותר n צמתים.
מתכונות של עצים בינאריים שלמים שראינו בהרצאה, מתקיים: $h(A_n) = a_n$.
כמו כן:

$$h(A_n) = a_n \leq \log_2(n+1) - 1 \leq \log_2(n+1)$$

באופן דומה, יהא b_n הטבעי הקטן ביותר המקיים:

$$\sum_{x=0}^{b_n} 3^x = \frac{3^{b_n+1} - 1}{2} \geq n$$

מכיוון שמספר זה הוא סכום חזקות של 3, קיים עץ טרינארי שלם, שבפרט ניתן למלא בו ערכים כך שיהיה עץ 3־2, בעל מספר צמתים זה, שנסמנו B_n.
כמו כן, לפי האופן שבו הגדרנו את b_n נקבל שב־B_n יש לכל הפחות n צמתים.
מתכונות של עצים טרינאריים שלמים מתקיים: $h(B_n) = b_n$.
כמו כן:

$$h(B_n) = b_n \geq \log_3(2n+1) - 1 \geq \log_3(n+1) - 1$$

בסך הכול, מכיוון שמספר הצמתים ב־A_n הוא לכל היותר n, ומספר הצמתים ב־B_n הוא לכל הפחות n (מהאופן שבו הגדרנו אותם),
נובע שבעץ 3־2 בעל n צמתים הגובה המקסימלי הוא לכל הפחות גובהו של A_n, והגובה המינימלי הוא לכל היותר גובהו של B_n.
(זאת משום שפעולת הכנסת צמתים לעץ 3־2 לא יכולות להקטין את גובהו).
נקבל:

$$D(n) \geq h(A_n) - h(B_n) = \log_2(n+1) - \log_3(n+1) + 1$$

ובפרט כפי שראינו באינפי:

$$\lim_{n\to\infty} D(n) = \lim_{n\to\infty} \log_2(n+1) - \log_3(n+1) + 1 = \infty$$

מכך נובע ש־$D(n) \neq O(1)$, שכן אם נניח בשלילה שקיימים $c, n_0 \in \mathbb{N}$ שעבורם לכל $n \geq n_0$:

$$D(n) \leq c \cdot 1$$

אז בפרט נקבל כי $D(n)$ חסומה, וזאת בסתירה לכך שהראינו שגבולה הוא ∞.

סעיף ה'

הטענה אינה נכונה.
נוכיח שעבור כל עץ בינארי בעל שני צמתים לפחות מתקיים אחד מהבאים:

- סריקות ה־$preorder$ וה־$inorder$ שלו שונות.
- סריקות ה־$postorder$ וה־$inorder$ שלו שונות.

ובפרט הדבר יהיה מספיק להוכיח שלכל עץ בינארי בעל שני צמתים לפחות לא כל 3 הסריקות בו זהות.

יהא T עץ בינארי בעל 2 צמתים לפחות.
בתורת הגרפים ראינו משפט שלפיו בכל עץ כזה קיים זוג צמתים המחוברים בקשת.
נסמן u, v שני צמתים מחוברים כאלה ב־T שקיומם מובטח, כאשר כיוון הקשת ביניהם הוא בלי הגבלת הכלליות מ־u אל v.
מתכונות של עץ בינארי נובע שייתכנו 2 מקרים:
<u>אם v הוא בן ימני של u:</u>
במקרה זה בסדר המפתחות מתקיים $u > v$, ומשום ש־$inorder$ סורק בסדר מפתחות עולה, אם נבצע הדפסת $inorder$ נקבל שבהדפסה זו u יודפס לפני v.
מנגד, לפי הגדרת $postorder$, סורקים כל בן של צומת לפני הצומת עצמו, לכן בהדפסת $postorder$ נדפיס את v לפני u.
בסך הכול מקבלים במקרה זה כי הדפסות ה־$postorder$ וה־$inorder$ שונות, שכן הסדר של u ו־v בהן שונה (מאחר שכל צומת מודפס בדיוק פעם אחת בכל סריקה).

<u>אחרת, v הוא בן שמאלי של u:</u>
במקרה זה בסדר המפתחות מתקיים $u < v$, ומשום ש־$inorder$ סורק בסדר מפתחות עולה, אם נבצע הדפסת $inorder$ נקבל שבהדפסה זו v יודפס לפני u.
מנגד, לפי הגדרת $preorder$, סורקים כל צומת לפני הבנים שלו, לכן בהדפסת $preorder$ נדפיס את u לפני v.
בסך הכול מקבלים במקרה זה כי הדפסות ה־$preorder$ וה־$inorder$ שונות, שכן הסדר של u ו־v בהן שונה (מאחר שכל צומת מודפס בדיוק פעם אחת בכל סריקה).

בסך הכול קיבלנו שבכל מקרה קיימות 2 הדפסות שונות, כפי שרצינו להוכיח.

שאלה 2

סעיף א'

ייתכנו 2 מקרים:

<u>אם העצים בגובה שווה:</u>

כלומר $h_1 = h_2$, אז ניצור צומת עם אינדקס min_2 (הערך המינימלי ב־T_2 שידוע מראש), ונחבר אליו כבנים שמאלי וימני את השורשים של T_1 ושל T_2 בהתאמה.

העץ החדש שנוצר הוא עץ 3־2, שכן כל העלים באותה רמה ($h_1 = h_2$), וכן האינדקס בשורש מקיים את התנאי עבור אינדקסים בעצי 3־2, כי לפי ההגדרה כל עלה בבן הימני שלו, T_2, גדול מ־min_2, וכל עלה בבן השמאלי שלו, T_1, קטן ממש מ־min_2 שכן נתון שכל מפתח ב־T_1 קטן מכל מפתח ב־T_2 ובפרט מ־min_2.

את תתי העצים T_1, T_2 לא שינינו כלל, ולכן נוצר עץ $3-2$ שמכיל את איחוד קבוצת המפתחות.

סיבוכיות יצירה של צומת חדש והשמה של מספר סופי של ערכים היא $O(1)$ ובפרט $O(|h_1 - h_2| + 1)$.

<u>אחרת, אם העצים בגובה שונה:</u>

נניח בלי הגבלת הכלליות שמתקיים $h_1 > h_2$ (המקרי השני סימטרי לחלוטין).

במקרה זה, נעתיק את המצביע לשורש של T_1, ונתחיל לרדת בעץ בכל פעם אל הבן הימני ביותר הזמין (כלומר בעל העלים הגדולים ביותר), עד שנגיע לצומת שנמצא בגובה $h_2 + 1$ בעץ T_1 (בהכרח קיים כזה, שכן $h_1 > h_2$).

אל צומת זה, נוסיף כבן הימני ביותר את שורש T_2, ונוסיף למערך האינדקסים שבו את min_2 כאינדקס הימני ביותר.

כעת ייתכן מצב שבו יצרנו ב־T_1 קיים צומת בעל 4 בנים. במקרה זה, נבצע את אלגוריתם התיקון הרגיל של עצי 3־2 שראינו בהרצאה החל מהצומת ששינינו ועד השורש.

ניתן לבצע זאת משום שתת העץ T_2 הוא עץ 3־2 תקין, ופרט לצומת בעל 4 הבנים שאר העץ T_1 תקין גם הוא - ואלגוריתם התיקון עבור עצי 3־2 נועד בדיוק עבור מקרים שבהם צומת אחד אינו תקין.

בסוף התהליך מתקבל עץ $3-2$ תקין בדומה למקרה הקודם, שכן מהנתונים ומהגדרת מינימום, האינדקס שהוספנו אכן מקיים את התנאי על אינדקסים בצמתים של עצי 3־2.

סיבוכיות הירידה בעץ T_1 היא $O(h_1 - h_2)$, שכן זהו מספר הצמתים שבהם חלפנו, סיבוכיות חיבור העצים ויצירת האינדקס היא $O(1)$, וסיבוכיות התיקון גם היא $O(h_1 - h_2)$ מכיוון שתיקנו צמתים רק החל מהצומת ששינינו ומעלה, ותיקון של כל צומת יחיד הוא $O(1)$.

בסך הכול הסיבוכיות היא $O(|h_1 - h_2| + 1)$ כנדרש.

סעיף ב'

נעבור על אוסף העצים משמאל לימין (מהקטן לגדול) ונאחד בכל פעם את הצומת הבא עם איחוד כל הצמתים הקודמים לו, באמצעות האלגוריתם של סעיף א'.

ניתן להפעיל את האלגוריתם של סעיף א', שכן מנתוני השאלה, לכל $1 \leq i \leq k$, איחוד העצים $T_1, ..., T_i$ מכיל רק מפתחות שקטנים מכל המפתחות ב־T_{i+1}.

כמו כן, עבור איחוד העצים $T_1, ..., T_i$ ניתן להסיק את ערכי המינימום והמקסימום הנחוצים לאלגוריתם: המינימום הוא המינימום ב־T_1, והמקסימום הוא המקסימום ב־T_i.

נותר להוכיח שהתהליך מתרחש בסיבוכיות הרצויה.

לכל $1 \leq i \leq k$, נסמן U_i את עץ ה־2־3 המתקבל מאיחוד העצים $T_1, ..., T_i$.

נוכיח באינדוקציה שלכל $1 \leq i < k$ ייתכנו 2 אפשרויות:

- או $h(U_i) \leq h(T_{i+1})$
- או $h(U_i) = h(T_{i+1}) + 1$, ובמקרה זה בהכרח לשורש של U_i יש 2 בנים וגם $h(T_{i+2}) > h(T_{i+1})$.

<u>בסיס:</u>

עבור $i = 1$ נקבל $U_i = U_1 = T_1$.

במקרה זה נתון כי $h(T_2) \geq h(T_1)$, ולכן $h(T_2) \geq h(U_1)$, כלומר מתקיים המקרה הראשון.

<u>צעד:</u>

כעת נניח כי הטענה נכונה עבור i כלשהו.

נחלק למקרים:

- <u>אם $h(U_i) \leq h(T_{i+1})$</u>, אז נסתכל על איחוד העצים U_i ו־T_{i+1}.

לפי האלגוריתם שנלמד בהרצאה, לאחר שרשרת תיקונים בעץ 2־3 גובה העץ יכול לגדול בלכל היותר 1.

אם $h(U_{i+1}) = h(T_{i+1}) + 1$, כלומר גובה העץ גדל, אז בהכרח התרחש פיצול של השורש. הדבר נובע מהאלגוריתם התיקון שראינו בהרצאה ומכך שגובהי שני העצים ההתחלתיים היו נמוכים ממש מהגובה הסופי.

לכן נקבל:

$$h(U_{i+1}) = h(T_{i+1}) + 1 \leq h(T_{i+2}) + 1$$

וכן לשורש העץ יש 2 בנים (שכן התחרש פיצול שלו באיטרציה האחרונה), וכן מכיוון שנתון שאין 3 עצים בעלי אותו גובה, נובע שאם $h(T_{i+1}) = h(T_i)$, אז $h(T_{i+2}) > h(T_{i+1})$, כלומר התנאי השני שהגדרנו מתקיים, ואחרת נקבל שבכל מקרה התנאי הראשון יתקיים כי:

$$h(U_{i+1}) \leq h(T_{i+1}) + 1 \leq h(T_{i+2})$$

כלומר במקרה זה, לפחות אחד מהתנאים שהגדרנו מתקיים.

- <u>אחרת</u>, נקבל שלפי הנחת האינדוקציה $h(U_i) = h(T_{i+1}) + 1$ וכן לשורש של U_i 2 בנים וכן $h(T_{i+2}) > h(T_{i+1})$.

במקרה זה לפי איך שהגדרנו את האלגוריתם בסעיף א', הפעולה שתתבצע תהיה הכנסה של T_{i+1} כבן ימני של U_i, ומכיוון ש־$h(U_i) = h(T_{i+1}) + 1$ לא יידרש תיקון.

כלומר נקבל:

$$h(U_{i+1}) = h(T_{i+1} + 1) \leq h(T_{i+2})$$

ובמקרה זה התנאי הראשון שהגדרנו מתקיים.
ובסך הכול הוכחנו את טענת העזר.

כעת נותר להוכיח את סיבוכיות התהליך כולו.
לפי טענת העזר, נקבל שלכל $1 \leq i \leq k$:

$$h(U_i) \leq h(T_{i+1}) + 1$$

בנוסף, כאשר מאחדים 2 עצי 2־3 גובהם לכל הפחות נשאר זהה (או גדל ב־1), לכן גם מתקיים:

$$h(U_i) \geq h(T_i)$$

לכן נקבל:

$$|h(T_{i+1}) - h(U_i)| \leq max\{h(T_{i+1}) - h(T_i), 1\}$$

לפי תוצאת סעיף א' ולפי התוצאה שקיבלנו כעת, סיבוכיות הזמן הכוללת היא:

$$\sum_{i=1}^{k-1}(|h_{i+1} - h_1| + 1) \leq$$

$$\begin{aligned}
&\leq \sum_{i=1}^{k-1} max\{h_{i+1} - h_i, 1\} + 1 \leq \\
&\leq \sum_{i=1}^{k-1}(h_{i+1} - h_i + 1) + \sum_{i=1}^{k-1}(1+1) \leq \\
&\leq (k-1) + (h_k - h_1) + 2 \cdot (k-1) = \\
&= 3 \cdot (k-1) + (h_k - h_1) \leq 3 \cdot (h_k - h_1 + k)
\end{aligned}$$

ולכן עבור $c = 3$ נקבל שלכל $n \geq 1$, הסיבוכיות קטנה מ־$c \cdot (h_k - h_1 + k)$,
ולכן לפי ההגדרה הסיבוכיות היא $O(h_k - h_1 + k)$ כנדרש.

סעיף ג'

ראשית נאתחל 2 רשימות מקושרות שיכילו בהמשך מצביעים לצמתים בעץ המקורי T.
רשימה $less$ תכיל בעתיד עצים שעליהם קטנים או שווים ל־x, ורשימה $greater$ תכיל בעתיד עצים שעליהם גדולים או שווים ל־x.
בנוסף נאתחל 2 רשימות נוספות שנעדכן במקביל ל־$less$ ול־$greater$, והן יכילו בהתאמה עבור כל תת עץ באחת מהרשימות את הגובה שלו וערכי המינימום והמקסימום שלו בחוליה המתאימה.
בסופו של דבר נוכיח שניתן להפעיל את האלגוריתם מסעיף ב' על כל אחת מהרשימות לקבלת העצים הסופיים הרצויים.

תחילה נרד בעץ T ונשמור משתנה מונה $height$ אשר נגדיר בכל פעם, כך שכשנגיע לעלה המשתנה יכיל את $h(T)$.
בנוסף נאתחל משתנים $range_1 = -\infty$ ו־$range_2 = \infty$ אשר באלגוריתם שיתואר ישמרו את טווח הערכים האפשרי בתת העץ הנוכחי.

נתחיל משורש העץ T ונרד מטה רמה־רמה תוך שאנו מעדכנים את 2 הרשימות באופן הבא:
בכל צומת שאליו אנו מגיעים, ניתן לבדוק את רשימת האינדקסים הסופית (לכל היותר 3 אינדקסים) על מנת לגלות באיזה בן (תת־עץ) עלול המפתח x להימצא.
בהכרח יש בן יחיד כזה, מהגדרת אלגוריתם החיפוש של עצי 3־2 שראינו בהרצאה.
בנוסף נסיק מערכי האינדקסים את טווח הערכים המצומצם החדש של תת העץ שעלול להכיל את x, כלומר נעדכן בהתאם את ערכי $range_1$ ו־$range_2$.
אם קיימים בצומת הנוכחי בנים שמאליים לאותו בן שמצאנו, נכניס אותם לתחילת רשימה $less$.
אם קיימים בצומת הנוכחי בנים ימניים לאותו בן שמצאנו, נכניס אותם לתחילת רשימה $greater$.
כאשר אנו מכניסים צומת לאחת הרשימות, בנוסף נכניס לרשימת עזר מתאימה חוליה המכילה את גובהו, ואת ערכי המינימום והמקסימום שלו:
הגובה הוא הגובה הנוכחי השמור במונה $height$.
עבור כל "אח שמאלי" של תת העץ שמצאנו, טווח הערכים שלו חסום על ידי $range_1$ מלמטה ועל ידי x מלמעלה על פי ההגדרה.
באופן דומה עבור כל "אח ימני", טווח הערכים שלו חסום על ידי $range_2$ מלמעלה ועל ידי x מלמטה.
את הערכים הללו נכניס לחוליה של רשימת עזר מתאימה עבור כל צומת שאנו מכניסים לרשימות $less$ ו־$greater$.
את הבן שמצאנו לא נכניס לאף רשימה, אלא נרד אליו ונפעיל עליו את האלגוריתם בשנית, תוך שאנו מקטינים ב־1 את המשתנה $height$ ששומר את גובה העצים שאנו מכניסים לרשימות בכל שלב.

נסיים את האלגוריתם כאשר נגיע לצומת שהוא עלה, כלומר חסר בנים, ונכניס אותו כעץ 3־2 בעל צומת יחיד אל אחת הרשימות:
- אם ערך העלה הזה הוא x ומטה, נכניס אותו ל־$less$, ואחרת נכניס אותו ל־$greater$.

לאחר מכן נפעיל את האלגוריתם של סעיף ב' על כל אחת מהרשימות $less$ ו־$greater$.

נראה שעץ ה־2־3 שיתקבל מ־$less$ הוא T_1 הרצוי, ושעץ ה־2־3 שיתקבל מ־$greater$ הוא T_2 הרצוי.

כעת נבצע מספר הבחנות על נכונות האלגוריתם:

- מאופן הכנסת הצמתים לרשימות שתיארנו: נובע ישירות שצמתים הנמצאים ב־$less$ הם שורשים של תתי־עצים של T המכילים בסך הכול את כל המפתחות הקטנים או שווים ל־x, וצמתים הנמצאים ב־$greater$ הם שורשים של תתי עצים של T המכילים את המפתחות הגדולים מ־x.
- כמו כן, מכיוון שבכל שלב של האלגוריתם ירדנו רמה בעץ, נובע שגובה כל שורש של תת עץ שהכנסנו לרשימות קטן, ומכיוון שהכנסנו לתחילת הרשימות בכל פעם, נקבל שתתי העצים בשתי הרשמיות ממוינים לפי גבהים בסדר עולה.
- בנוסף, מכיוון שכשהוספנו צמתים לרשימת $greater$, בשלב הבא של האלגוריתם ירדנו בבן שנמצא שמאלה מהם, נקבל שבכל פעם שהכנסנו לרשימה תת עץ T_i, כל המפתחות בו היו קטנים מכל המפתחות בעץ שהכנסנו לפניו T_{i-1}. מכיוון שהכנסנו לתחילת הרשימה, נקבל שרשימת $greater$ ממוינת לפי התנאים של סעיף ב'.
- רשימת $less$ גם היא ממוינת באופן דומה, אלא שמכיוון שבכל שלב ירדנו לתת עץ ימני של כל הצמתים שהכנסנו באותו שלב ל־$less$, נקבל שהרשימה ממוינת בסדר הפוך לפי החסמים על טווחי המפתחות בכל תת עץ שלה (כלומר המפתחות בכל עץ **גדולים** מהמפתחות של העץ הבא). עם זאת, הדבר לא מפריע באופן עקרוני לאלגוריתם של סעיף ב', שכן נכונותו התבססה על זרות הטווחים הללו בלבד, ולכן השינוי היחיד שנבצע בו הוא החלפת הסדר שבו אנו מחברים כל 2 עצים סמוכים.

בסך הכול, משום ששמרנו ברשימות עזר את הנתונים הנחוצים לאלגוריתם של סעיף ב', נוכל כעת לעבור על $less$ ועל $greater$ ולייצר מהם את T_1 ואת T_2 כדרוש.

סיבוכיות הזמן:

תחילה על מנת למצוא את גובה T ירדנו מהשורש לעלים, כלומר בסיבוכיות של גובה העץ שהיא $\log(n)$, ובנוסף אתחלנו מספר סופי של משתנים ב־$O(1)$.

לאחר מכן עבור כל רמה בעץ T ביצענו מספר סופי של פעולות העתקה של ערכים ומצביעים ב־$O(1)$, וזאת ביצענו פעם אחת עבור כל רמה.

לכן בסך הכול סיבוכיות בניית הרשימות היא $O(\log(n))$ גובה עץ 2־3 בעל n צמתים.

בכל רמה הכנסנו לכל רשימה לכל היותר 2 עצים (שכן בכל צומת של T קיימים לכל היותר 3 בנים שאת אחד מהם אנו לא מכניסים לאף רשימה).

לכן מספר האיברים בכל רשימה הוא $O(\log(n))$.

מכיוון שגובה העץ האחרון בכל רשימה חסום על ידי גובה העץ T כולו, נקבל שסיבוכיות הפעלת אלגוריתם סעיף ב' על כל רשימה היא:

$$O(h_k - h_1 + k) = O(\log(n) + \log(n))$$

ובסך הכול סיבוכיות רצף כל הפעולות הוא $O(\log(n))$ כנדרש.

שאלה 3

נפתור באמצעות מבנה רגיל של עץ AVL ששומר בנוסף את שדה הגודל שלו, והמפתחות בו יהיו נתונים שמייצגים קווים במישור, אלא שיחס הסדר בין המפתחות לא יהיה יחס הסדר הרגיל בין מספרים, אלא יחס סדר חדש שיוגדר בין ייצוגים של קווים במישור על מנת להבטיח שלא תהיה הכנסה של קווים נחתכים:

לאחר אתחול של מבנה עם פרמטר m, המפתחות של העץ יהיו זוגות סדורים של מספרים, (s,t), כך שעבור הישר $f(x) = ax + b$, הייצוג שלו כמפתח יהיה:

$$s = f(0) = b$$
$$t = f(m) = am + b$$

כלומר המפתח הוא זוג סדור שהאיבר הראשון בו הוא גובה החיתוך בין הישר לציר y, והאיבר השני בו הוא גובה החיתוך בין הישר הנתון לישר $x = m$ המוגדר כתלות בערך m שאיתו המבנה אותחל.

את יחס הסדר על המפתחות נגדיר באופן הבא:

- נגדיר $(s_2,t_2) \prec (s_1,t_1)$ אמ"מ $s_1 < s_2$ וגם $t_1 < t_2$.
- נגדיר $(s_2,t_2) \succ (s_1,t_1)$ אמ"מ $s_1 > s_2$ וגם $t_1 > t_2$.
- נגדיר $(s_2,t_2) \sim (s_1,t_1)$ שוויון בכל מקרה אחר.

כעת נוכיח שיחס הסדר שהגדרנו אכן משקף את הכוונות שלנו, כלומר שלכל m ולכל 2 ישרים מתקיים:
a_1x+b_1 ו־a_2x+b_2 נחתכים בקטע $[0,m]$ אם ורק אם המפתחות המתאימים שמייצגים אותם, (s_1,t_1) ו־(s_2,t_2), מקיימים $(s_2,t_2) \sim (s_1,t_1)$.
(ולאחר מכן נגדיר את מבנה הנתונים שנכונותו תתבסס על טענה זו)

<u>כיוון 1:</u>
נניח שמתקיים $(s_1,t_1) \sim (s_2,t_2)$ ונוכיח שהישרים נחתכים בקטע.
לפי ההגדרה של יחס הסדר, מתקיים אחד מבין הבאים:
$s_1 \geq s_2$ וגם $t_1 \leq t_2$, או $s_1 \leq s_2$ וגם $t_1 \geq t_2$.
נניח בלי הגבלת הכלליות שזהו המקרה הראשון.
אם $s_1 = s_2$ או $t_1 = t_2$ אז ברור שהישרים נחתכים בקטע: ב־$x = 0$ או ב־$x = m$ בהתאמה.
(כי לפי הגדרת המפתחות אלה הם הגבהים של הישרים בנקודות $x = 0, m$).
אחרת, נקבל $s_1 > s_2$ וגם $t_1 < t_2$ (במקרה זה גם נובע $m \neq 0$).
במקרה זה, נגדיר:

$$f_1(x) = a_1 x + b_1$$
$$f_2(x) = a_2 x + b_2$$

וכן נגדיר את פונקציית ההפרש:

$$g(x) = f_1(x) - f_2(x)$$

מכיוון שאלה כולן פונקציות ליניאריות: הן רציפות.

כמו כן, לפי הדרך שבה הגדרנו את המפתחות מתקיים:

$$g(0) = f_1(0) - f_2(0) = s_1 - s_2 > 0$$
$$g(m) = f_1(m) - f_2(m) = t_1 - t_2 < 0$$

ולכן לפי משפט ערך הביניים נקבל שקיימת נקודה $c \in [0,m]$ שעבורה:

$$g(c) = 0 \iff f_1(c) = f_2(c)$$

כלומר קיימת נקודה בקטע $[0,m]$ שבה הישרים נחתכים.

<u>כיוון 2:</u>
כעת נניח $(s_1,t_1) \prec (s_2,t_2)$ ונוכיח שהישרים אינם נחתכים.
(זה יוכיח גם עבור המקרה שבו $(s_1,t_1) \succ (s_2,t_2)$ שכן ינבע $(s_2,t_2) \prec (s_1,t_1)$ ונקבל בחזרה את המקרה שאנו מוכיחים כעת).
לפי ההנחה מתקיים:
$s_1 < s_2$ וגם $t_1 < t_2$.
נניח בשלילה שקיימת נקודה $c \in [0,m]$ שעבורה $f_1(c) = f_2(c)$, כלומר שבה הישרים נחתכים.
אם $m = 0$ אז בהכרח $m = c = 0$, אך זוהי סתירה להנחה, שכן:

$$s_1 = f_1(0) = f_1(c) = f_2(c) = f_2(0) = s_2$$

ואילו אנחנו הנחנו $s_1 < s_2$.
נובע שהישרים אינם שווים בנקודה c, שהיא היחידה בקטע, כלומר הם אינם נחתכים בקטע.

אחרת, אם $m \neq 0$, נסמן כמו בהוכחת הכיוון הראשון:

$$g(x) = f_2(x) - f_1(x)$$

לפי ההנחות נקבל:

$$\begin{aligned} g(0) &= f_2(0) - f_1(0) = s_2 - s_1 > 0 \\ g(c) &= f_2(c) - f_1(c) = 0 \\ g(m) &= f_2(m) - f_1(m) = t_2 - t_1 > 0 \end{aligned}$$

שוב נבחין כי $g(x)$ פונקציה ליניארית, כי היא הפרש של ליניאריות,
כלומר הנגזרת שלה שווה לערך **קבוע** $g'(x) = p$.
נגזרת של פונקציה ליניארית שווה לשיפוע הישר בין כל 2 נקודות בקטע.
בפרט נקבל מההנחה:

$$\begin{aligned} p &= \frac{g(m) - g(c)}{m - c} = \frac{t_2 - t_1}{m - c} > 0 \\ p &= \frac{g(c) - g(0)}{c - 0} = \frac{s_1 - s_2}{c} < 0 \end{aligned}$$

בפרט קיבלנו $0 < p < 0$ וזו סתירה $0 \neq 0$.
מכאן שההנחה שגויה, כלומר לא קיימת $c \in [0,m]$ שעבורה הישרים נחתכים.

בסך הכול הוכחנו שיחס הסדר שלנו "מתאים", כלומר 2 ישרים נחתכים בקטע $[0,m]$ אם ורק אם המפתחות המתאימים להם שווים לפי היחס החדש.

כעת נגדיר את מבנה הנתונים:

כאמור, מבנה הנתונים מכיל עץ AVL שהמפתחות בו הם זוגות סדורים המתארים ישרים, וההשוואה נעשית על פי יחס הסדר המיוחד שהגדרנו (היחס עצמו אינו תלוי ב־m).
כמו כן קיים במבנה שדה השומר את גודל העץ ושדה השומר את m, הערך שאיתו העץ מאותחל.

מכיוון שלא מתבצעות פעולות "קריאת ערכים" מהעץ: סוג הערכים בעלים שלו אינו משנה לצורך התרגיל, אך למען השלמות נגדיר שערך בכל עלה זהה למפתח שבעלה.

$Init(m)$:

נאתחל עץ AVL ריק (מתבצע ב־$O(1)$ כפי שידוע מההרצאה), נאתחל את שדה הגודל ל־0 ונשמור את m במשתנה פנימי שיהיה קבוע לאורך כל הריצה.

$InsertLine(a, b)$:

תוך שימוש בערך m השמור בעץ במשתנה פנימי ונתון לגישה ב־$O(1)$, נחשב מתוך הישר $ax + b$ את המפתח המתאים לו:

$$(b, a \cdot m + b)$$

פעולות אריתמטיות אלה מתבצעות ב־$O(1)$ ומספרן סופי.

לאחר מכן, נבצע **חיפוש והכנסה** של המפתח החדש בעץ: נכניס את המפתח לעץ רק אם לא קיים בעץ מפתח שווה.

הנקודה החשובה היא שההשוואה מתבצעת על פי יחס הסדר שהגדרנו, שבו 2 מפתחות ניתנים להשוואה על ידי מספר סופי של פעולות ב־$O(1)$.

אם נמצא בעץ מפתח ה"שווה" למפתח החדש, אזי לפי טענת העזר שהוכחנו בהתחלה הישרים המתאימים להם נחתכים בקטע $[0, m]$.

לכן לא נכניס במקרה זה את המפתח החדש.

אחרת, נובע (מתכונות עצים) שלא קיים מפתח בעץ "השווה" למפתח החדש, ושוב מכיוון שטענת העזר דו־כיוונית נובע שלא קיים בעץ ישר הנחתך עם הישר החדש $ax + b$.

במקרה זה נכניס לעץ את המפתח החדש (עם ערך זהה לו) ונגדיל ב־1 את השדה השומר את גודל העץ.

בסך הכול ביצענו מספר סופי של השוואות וחישובים ב־$O(1)$, נוסף על 2 פעולות חיפוש והכנסה לכל היותר לעץ AVL, כלומר הסיבוכיות הכוללת היא $O(\log(n))$.

$Size()$:

נחזיר את שדה הגודל השמור במבנה. מכיוון שזהו משתנה פנימי הסיבוכיות היא $O(1)$, ומכיוון שהגדלנו אותו אך ורק כאשר הכנסנו בהצלחה מפתח, נקבל שזהו אכן מספר האיברים בעץ, שהוא בדיוק מספר הקווים במבנה.

סיבוכיות מקום:

המימוש הוא למעשה עץ AVL רגיל, שסיבוכיות המקום שלו היא $O(\log(n))$ כאשר n הוא מספר המפתחות בעץ.

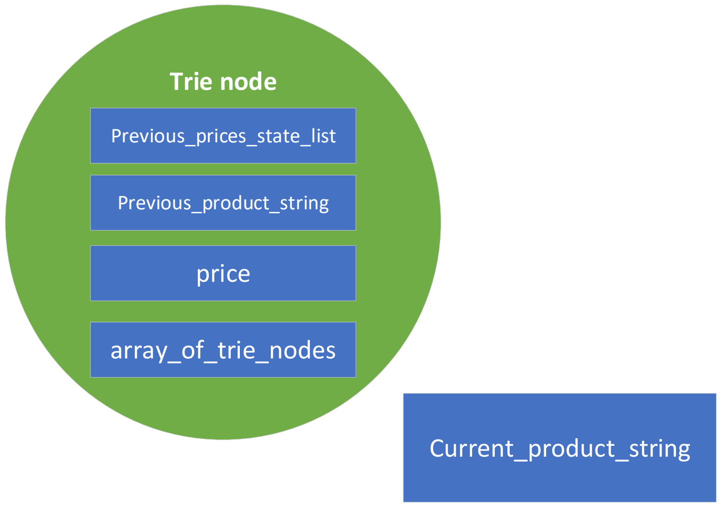
כמו כן, סיבוכיות המקום של כל מפתח וערך היא $O(1)$ שכן זהו מספר סופי וקבוע של ערכים.

לכן סיבוכיות המקום הכוללת היא $O(\log(n))$ כאשר n הוא מספר הישרים במבנה.

שאלה 4

חלק ראשון

נשתמש במבנה הנתונים Trie שלמדנו בקורס, כאשר כל צומת יהיה מהצורה:

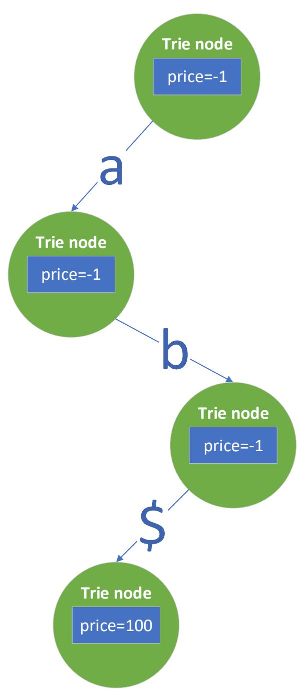


בנוסף, נשמור משתנה נוסף בצד ששמו $current\_product\_string$.

הגדרה: 2 מוצרים ייקראו **חופפים** אם תיאור 2 המוצרים מתחיל באותה תת מחרוזת s כלשהי. לדוגמא המחרוזות "abc" ו-"ab" הן חופפות, אך המחרוזות "cb" ו-"ab" אינן חופפות.

נסביר את המשמעות של כל אחד מהאיברים שהצומת $Trie\ node$ מחזיק:

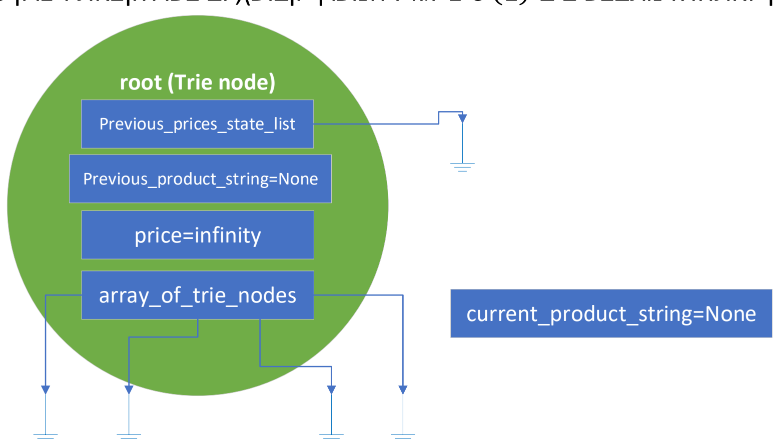
- $\boldsymbol{price}$: אם הצומת הוא הצומת האחרון שמייצג את המוצר בעץ (הצומת שמגיע לאחר $), ערך זה יכיל את מחיר המוצר. אחרת, ערך זה יכיל את המחיר המינימלי מבין כל המוצרים החופפים שצומת זה מייצג. לדוגמא עבור העץ המכיל את המחרוזות *abcd, abce, abcf*, הצומת המגיע לאחר הקשת *c* יכיל ב-price את המחיר המינימלי מבין כל שלושת המוצרים. דוגמא עבור מצב בסיסי במוצר *'ab'*:



- ***previous_product_string***: מכיל את המחרוזת של המוצר הקודם שהוכנס למערכת בזמן שבו הוכנסה המחרוזת הנוכחית. הצומת היחיד שמכיל previous_product_string בעל משמעות הוא הצומת האחרון שאליו אנו מגיעים לאחר $ (יוסבר בהמשך באופן יותר מפורט כיצד מעדכנים את איבר זה).
- ***array_of_trie_nodes***: מערך באורך $|\Sigma|+1$ שמחזיק מצביעים לבנים (כפי שלמדנו בתרגול).
- ***previous_prices_state_list***: רשימה מקושרת של המצב הקודם של כל ה-$prices$ בכל הצמתים שנמצאים במסלול של המוצר בגרף. לרשימה זו יש משמעות רק בעלים של העץ, כי היא מייצגת את המצב הקודם של ה-$prices$ בכל המסלול (מהשורש אל העלה). בכל אחת מהצמתים הפנימיים של העץ, ערך משתנה זה תמיד יהיה $null$.

נתאר כעת את מהלך הפעולות במבנה:

- $\boldsymbol{Init}()$: נאתחל עץ $Trie$ ידי יצירת $Trie\ node$ ששמו $root$, ולאחר מכן בנייה של מערך הבנים $array\_of\_trie\_nodes$ כך שכל האיברים במערך יצביעו ל-$null$. בנוסף, נבצע השמה: $current\_product\_string=None, price=\infty$, $previous\_product\_string=None$, ו-$previous\_prices\_state\_list=null$. **סיבוכיות הזמן היא $\boldsymbol{O(1)}$** מכיוון שביצענו כמות סופית של פעולות (גם הקצאת המערך ואתחולו מתבצעים ב-$O(1)$ כי גודל המערך קבוע), וביצעו הקצאת זיכרון סופית ולא דינמית.



- $\boldsymbol{Add(p,desc)}$: נאתחל רשימה מקושרת $prices\_list$. נתחיל לעבור על המסלול בעץ ה-$Trie$ המתאר את המחרוזת $desc$, ונייצר את הבנים הרלוונטיים במהלך הדרך. בכל בן שנייצר, נאתחל את משתניו להיות זהים למשתנים שאתחלנו בצומת $root$ בפונקציה $Init()$. עבור כל צומת שנעבור בו בדרך (כולל צומת שזה עתה ייצרנו – אם בכלל יצרנו - ולא כולל הצומת האחרון במסלול), נוסיף את הערך היושב ב-$price$ בצומת לראש הרשימה $prices\_list$. לאחר מכן, נעדכן את הערך היושב ב-$price$ להיות המינימלי מבין $price$ ובין p. לבסוף, בצומת האחרון שנייצר שמגיע לאחר הקשת $, נעדכן $price=p$, כלומר נשמור ב-$price$ את מחיר המוצר שעתה הכנסנו, $previous\_product\_string=current\_product\_string$, כלומר נשמור ב-$previous\_product\_string$ את שם המוצר שנכנס לפני המוצר שזה עתה הכנסנו, ונשים את הרשימה המקושרת $prices\_list$ שייצרנו ב-$previous\_prices\_state\_list$ (עדכונים אלו מתבצעים בצומת האחרון בלבד). לאחר מכן נבצע: $current\_product\_string=desc$, כלומר נשמור ב-$current\_product\_string$ את שם המוצר שבעת הכנסנו. סיבוכיות הזמן לייצר כל אחד מהצמתים בדרך, להוסיף איברים לרשימה המקושרת, ולעדכן את כל המשתנים בתוך כל צומת היא $O(1)$ כי עבור כל צומת אנו מבצעים כמות קבועה של פעולות. אנו מייצרים בסה"כ $|desc|+1$ צמתים ($|desc|$ צמתים עבור המחרוזת $desc$ ועוד צומת אחד עבור התו '$'),

ומבצעים מספר קבוע של השוואות והשמות בין משתנים בכל צעד ברקורסיה (שעומקה $|desc|+1$), **ולכן סיבוכיות הזמן בסה"כ היא $\boldsymbol{O(|desc|)}$.**

- $\boldsymbol{Undo()}$**:** אם $current\_product\_string$ אינו מכיל דבר (None), זה אומר שעוד לא הכנסנו אף מחרוזת למבנה הנתונים, ולכן לא נבצע דבר. אחרת, נייצר משתנה חדש בשם $current\_undo\_desc$, ונעביר אליו את המחרוזת הנמצאת ב-$current\_product\_string$. על פי הגדרת המבנה שלנו, כל צומת שהוא סיומת של מחרוזת של מוצר x כלשהו (צומת שמגיע לאחר $'\$'$) מכיל בתוכו את שם המוצר שנכנס לפניו, את מחיר המוצר x, ורשימה מקושרת של המצב הקודם לכל המשתנים $price$ הנמצאים בצמתים על המסלול של המוצר. נעבור על המסלול שמייצג את המחרוזת $current\_undo\_desc$ בעץ ה-$Trie$ שלנו. לבסוף, כשנגיע לצומת האחרון במסלול (הצומת שלאחר הקשת $'\$'$), נשלוף מצומת זה את ערך המשתנה $previous\_product\_string$ ונבצע השמה של מחרוזת זו אל תוך $current\_product\_string$. כלומר, נעדכן את $current\_product\_string$ כך שיכיל את שם המוצר לפני המוצר שאנו מבצעים לו $Undo()$. בנוסף, נשלוף את הרשימה המקושרת $previous\_prices\_state\_list$ מהצומת האחרון ונשמור אותה בצד לשימוש עתידי. נחזור אחורה במסלול, ובכל צומת שנעבור בו נשלוף את המספר הבא מ-$previous\_prices\_state\_list$. הערך שנמצא כל פעם בראש הרשימה $previous\_prices\_state\_list$ הוא ה-$price$ הנכון לעדכון מכיוון שהוא מייצג את מצב כל ערכי ה-$price$ בכל הצמתים במסלול של המוצר שאנו עושים לו $Undo()$, לפני שהכנסנו אותו. אם לצומת אין בנים, זה אומר שהצומת רלוונטית רק במסלול של המוצר שאנו מוציאים, ולכן נמחק את הצומת זה. אחרת, אם לצומת יש בנים, זה אומר שיש מוצר שחופף למוצר שאנו מוציאים, ולכן נעדכן את $price$ בצומת זה להיות הערך שזה עתה שלפנו מהרשימה המקושרת $previous\_prices\_state\_list$. מספר קבוע של השמות והעברת ערכים ממשתנה אחד לשני מתבצעת בסיבוכיות זמן $O(1)$. בנוסף, ביצענו מעבר רקורסיבי על מסלול בעץ ה-$Trie$ באורך $|current\_undo\_desc|+1$ כדי להגיע לצומת האחרון שמייצג את המוצר שאנו מבצעים לו $Undo()$. מעבר זה כמובן מתבצע בסיבוכיות זמן ומקום $O(|current\_undo\_desc|)$. לבסוף, מעבר חזרה על אותו מסלול בגרף שבמצע את המחיקה של המוצר יתבצע גם כן בסיבוכיות זמן $O(|current\_undo\_desc|)$, אך בסיבוכיות מקום $O(1)$ כי אנו מבצעים לכל היותר $|current\_undo\_desc|+1$ מחיקות, כמות קבועה של השמות והעתקות משתנים, ולא מבצעים שום הקצאה נוספת. **לסיכום, סיבוכיות הזמן היא $\boldsymbol{O(|current\_undo\_desc|)}$, כאשר $\boldsymbol{|current\_undo\_desc| = |desc|}$ הנתון בשאלה זו. לכן סיבוכיות הזמן היא $\boldsymbol{O(|desc|)}$ כנדרש.**
- $\boldsymbol{GetPrice(desc)}$**:** נעבור רקורסיבית על המסלול בעץ ה-$Trie$ שמייצג את המוצר ששמו $desc$ עד אשר נגיע לצומת האחרון במסלול. נשלוף את הערך שנמצא במשתנה $price$ בצומת האחרון במסלול, ונחזיר את ערך זה למשתמש במערכת. סה"כ ביצענו מעבר רקורסיב על מסלול באורך $|desc|+1$, ולכן המעבר על כל הצמתים והגעה לצומת האחרון במסלול מתבצע בסיבוכיות זמן ומקום $O(|desc|)$. שליפה של ערך ממשתנה מתבצעת בסיבוכיות זמן $O(1)$, וללא הקצאות נוספות, **בסה"כ נקבל שסיבוכיות הזמן היא $\boldsymbol{O(|desc|)}$.**

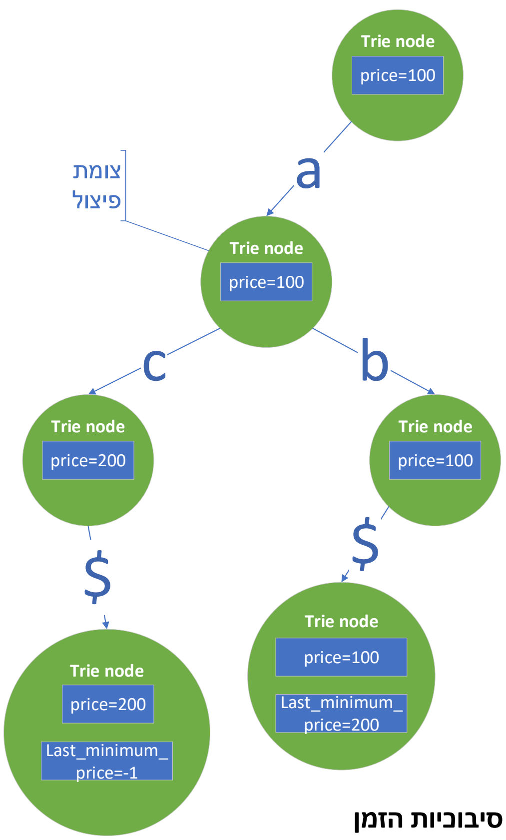
סיבוכיות המקום של המבנה היא $\boldsymbol{O(|total\_desc|)}$ כאשר $\boldsymbol{|total\_desc|}$ הוא האורך הכולל של כל תיאורי המוצרים שנמצאים כרגע במבנה הנתונים, זאת כי עומק העץ עבור כל מוצר שאורך התיאור שלו הוא $\boldsymbol{|desc|}$ הוא לפחות $\boldsymbol{|desc|}$, וייתכן כי כל המוצרים יהיו זרים אחד לשני.

חלק שני

המבנה שבנינו בחלק הראשון כבר מתאים לתמיכה בפעולה הנוספת של החלק השני, ועומד בכל דרישות הסיבוכיות גם של החלק הראשון וגם של החלק השני. נתאר את מהלך הפעולה החדשה שהתווספה:

- $\boldsymbol{CheapestOfType}(\boldsymbol{s})$**:** נניח ש-$s = c_1c_2 \dots c_n$. נעבור על המסלול בגרף $c_1c_2, \dots c_n$: במידה ועבור צומת כלשהו לא קיימת קשת c_k ($1 \le k \le n$), זה אומר שהמוצר לא קיים במערכת ונחזיר $NULL$. במקרה זה נעבור לכל היותר על $|s|$ קשתות ולכן נקבל סיבוכיות זמן $O(|s|)$. אחרת, אם הצלחנו לעשות מעבר על הקשת האחרונה c_n, בצומת שהגענו אליו קיים כבר הערך הרלוונטי להחזרה. על פי הגדרת מבנה הנתונים שלנו, המשתנה $price$ בצומת שנמצא לאחר הקשת c_n מכיל את המחיר המינימלי מבין כל המוצרים החופפים שצומת זה מייצג. לדוגמא, עבור העץ הבא ו-$CheapestOfType(a)$:

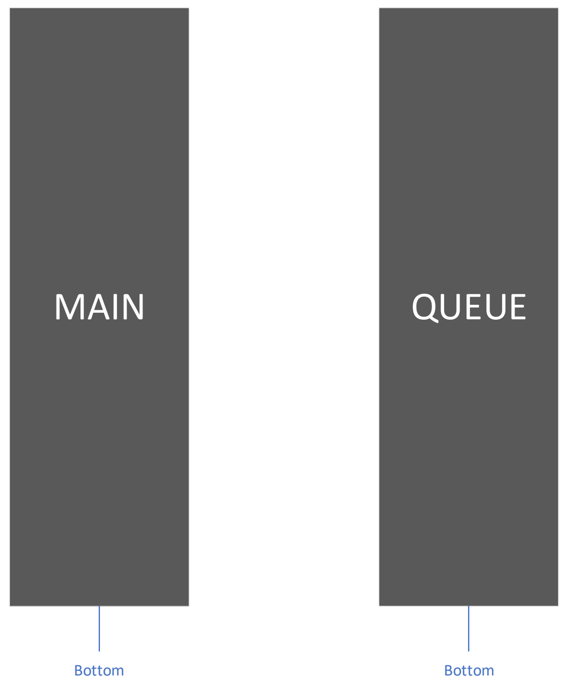
הצומת שלאחר הקשת a מכיל את המחיר 100, שהוא המחיר המינימלי מבין המוצרים ab ו-ac.



לכן, נחזיר למשתמש את הערך הנמצא במשתנה $price$ בצומת שלאחר הקשת c_n. **סה"כ עברנו על $n = |s|$ צמתים ברקורסיה, ולכן סיבוכיות הזמן של פעולה זו תהיה $O(|s|)$. סיבוכיות המקום נשארת זהה מכיוון שלא ביצענו אף הקצאה בפעולה זו.**

שאלה 5

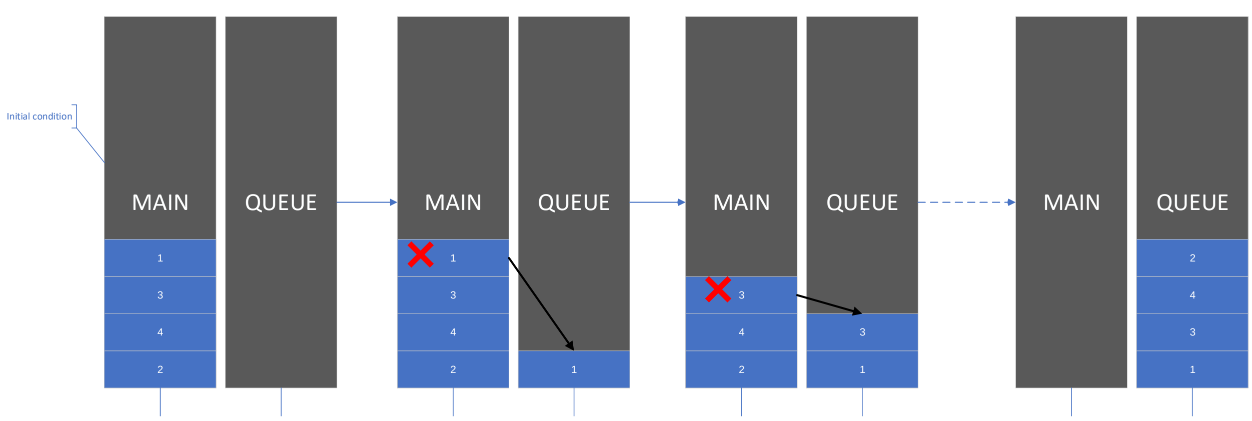
נשתמש ב-2 מחסניות LIFO לצורך המימוש של מחסנית ה-FIFO. להלן מחסנית MAIN ומחסנית QUEUE (הסימון Bottom מסמל את תחתית המחסניות):



InitQ(): נבנה 2 מחסניות LIFO ששמן MAIN ו-QUEUE על ידי קריאה ל-InitStack() פעמיים. סיבוכיות הזמן במקרה הגרוע היא O(1) כי ביצענו בסה"כ 2 פעולות שנתון שסיבוכיות הזמן שלהן במקרה הגרוע היא O(1), ולכן אנו עומדים בסיבוכיות הזמן הנדרשת.

EnQ(x): נבצע Push(x) למחסנית MAIN. סיבוכיות הזמן במקרה הגרוע עבור הכנסה אחת היא O(1) (נתון).

DeQ(x): נבדוק האם המחסנית QUEUE ריקה על ידי קריאה לפונקציה IsEmpty() בסיבוכיות זמן O(1) במקרה הגרוע (נתון). <u>**אם QUEUE ריקה**</u>, נעתיק את המחסנית MAIN אל המחסנית QUEUE באופן הבא: כל עוד המחסנית MAIN אינה ריקה, נבצע Pop() לאיבר x מהמחסנית MAIN ואז נכניס אותו למחסנית QUEUE על ידי שימוש ב-Push(x). נקרא לפעולה זו מעתה **העתקה הפוכה של MAIN ל-QUEUE:**



ברור כי כשנסיים לבצע את פעולת ההעתקה ההפוכה, נקבל שהמחסנית MAIN תהיה ריקה והמחסנית QUEUE תכיל את כל האיברים שהיו ב-MAIN בסדר הפוך מהסדר שבו הם הוכנסו ל-MAIN מלכתחילה (האיבר שהוכנס ראשון ב-MAIN יהיה האיבר בראש המחסנית QUEUE, והאיבר שהוכנס אחרון ב-MAIN יהיה בתחתית המחסנית QUEUE). אם לפני פעולת ההעתקה ההפוכה קיימים m איברים במחסנית MAIN, נראה שסיבוכיות הזמן לבצע העתקה הפוכה של MAIN ל-QUEUE היא O(m) במקרה הגרוע: אנו מבצעים כמות סופית של פעולות בעת העתקה של כל איבר מ-MAIN ל-QUEUE, כאשר כל פעולה לוקחת O(1) במקרה הגרוע (אנו מבצעים Push, Pop ו-IsEmpty עבור כל איבר). מכיוון שיש בסה"כ m איברים ב-MAIN, אנו מבצעים בסה"כ m פעולות בסיבוכיות של O(1) במקרה הגרוע, ולכן סיבוכיות הזמן הכוללת של ההעתקה ההפוכה היא O(m) במקרה הגרוע. כעת, לאחר ההעתקה, נבצע Pop() לאיבר הנמצא בראש המחסנית QUEUE, ובכך קיבלנו את האיבר הראשון שהכנסנו למחסנית MAIN ולכן בסה"כ קיבלנו מימוש Pop של מחסנית FIFO בסיבוכיות זמן O(m) במקרה הגרוע. **<u>אחרת, אם QUEUE אינה ריקה</u>**, זה אומר שביצענו בעבר העתקה הפוכה של MAIN ל-QUEUE (כי אין דרך אחרת שבה QUEUE יכולה להתמלא ע"פ מבנה הנתונים שהגדרנו), ולכן האיבר הראשון שהוכנס למחסנית MAIN נמצא בראש מחסנית QUEUE ע"פ מה שהסברנו קודם (גם אם MAIN המשיכה להתמלא, עדיין QUEUE מכילה את האיברים הראשונים שהוכנסו ל-MAIN). לכן אם נבצע Pop() למחסנית QUEUE נקבל כמו מקודם את האיבר הראשון שהוכנס למחסנית MAIN, אך בשונה ממקודם, מכיוון ש-QUEUE כבר מלאה ולא היינו צריכים להעתיק אליה את MAIN, ומכיוון שהשתמשנו פעם אחת בלבד ב-Pop() ופעם אחת בלבד ב-IsEmpty() כדי לבדוק אם QUEUE אינה ריקה, נקבל שסיבוכיות הזמן במקרה הגרוע להוצאת האיבר היא O(1). **לסיכום: אם QUEUE ריקה, סיבוכיות הזמן להוצאת האיבר הראשון שהוכנס ל-MAIN היא O(m) במקרה הגרוע כאשר m מסמל את כמות האיברים ב-MAIN, ואם QUEUE אינה ריקה, סיבוכיות הזמן להוצאת האיבר הראשון שהוכנס ל-MAIN היא O(1) במקרה הגרוע.**

לפני ניתוח סיבוכיות הזמן, נציין כי סיבוכיות המקום של המבנה הוא $O(n)$ מכיוון שאנו משתמשים ב-2 מחסניות שעבור כל אחת מהן סיבוכיות המקום היא $O(n)$, כאשר n זה מספר האיברים הכולל במבנה הנתונים.

כעת, ננתח את סיבוכיות הזמן המשוערכת של הפעולות EnQ(x), DeQ(x).

הוכחה בעזרת שיטת הצבירה: תהי סדרה של m פעולות מהסוגים EnQ(x) ו-DeQ(x). ונסמן את סדרת הפעולות ב-$a_1, a_2, \dots a_m$, כאשר a_1 היא הפעולה הראשונה בסדרה. **נחלק את סדרת הפעולות ל-k תתי סדרות** באופן הבא: נסרוק את סדרת הפעולות לפי הסדר (מהפעולה הראשונה לאחרונה). נייצר תת סדרה חדשה. כל עוד הפעולה הנוכחית היא EnQ(x), נוסיף את הפעולה אל תת הסדרה שייצרנו. אחרת (אם הפעולה הנוכחית היא DeQ(x)), נוסיף אותה לתת הסדרה שייצרנו, ונייצר תת סדרה חדשה שאליה נוסיף את הפעולות הבאות שנמצא בסריקה. נקבל k תתי סדרות כאשר כל תת סדרה של פעולות היא מהצורה: EnQ(x), EnQ(x), … DeQ(x). במידה ולא היה קיים אף DeQ(x), נקבל תת סדרה אחת שמכילה רק EnQ(x) (שגם תהיה שווה לסדרת הפעולות המקורית). **במקרה זה** (שלא קיים DeQ(x)), מכיוון שאנו מבצעים m פעולות EnQ(x), ומכיוון שכל פעולה מבוצעת בסיבוכיות זמן של O(1) במקרה הגרוע, נקבל שסיבוכיות הזמן עבור m פעולות היא O(m), ולכן סיבוכיות הזמן המשוערכת היא O(1). **אחרת**, אם קיימת פעולת DeQ(x) אחת לפחות, ננתח את סיבוכיות זמן הריצה במקרה המשוערך עבור תת סדרת הפעולות ה-i ($1 \leq i \leq k$): נניח ובתת סדרת הפעולות ה-i קיימות s_i פעולות בסה"כ. ידוע כי $s_i - 1$ הפעולות הראשונות יהיו EnQ(x) (ככה מוגדרת תת הסדרה), ומכיוון שכל פעולת EnQ(x) לוקחת O(1) סיבוכיות זמן במקרה הגרוע, נקבל שסיבוכיות הזמן במקרה הגרוע היא $O(s_i)$. כעת הפעולה ה-s_i היא בטוח DeQ(x) (על פי הגדרת תת סדרת הפעולות) וראינו שבמקרה הגרוע היא עולה לנו $O(s_i)$ (יש בסה"כ $s_i - 1$ במחסנית MAIN מכיוון שלפני שביצענו את תת סדרת הפעולות הנוכחית: או שלא ביצענו כלום והמחסנית $MAIN$ הייתה ריקה כי לא הכנסנו אף איבר למבנה (נתון), או שביצענו DeQ(x) מתישהו שרוקנה את MAIN). כלומר עבור סדרת הפעולות ה-i שבה יש s_i פעולות, קיבלנו שסיבוכיות הזמן במקרה הגרוע היא $O(s_i)$. מכיוון שבסה"כ יש m פעולות, ברור כי $s_1 + s_2 + \dots + s_k = m$. לכן, סיבוכיות זמן הריצה של כל תתי הסדרות ביחד תהיה סיבוכיות זמן הריצה של סדרת הפעולות המקורית (כי תתי הסדרות הן בעצם פירוק של סדרת הפעולות המקורית לחלקים נפרדים). סיבוכיות זמן הריצה של כל תתי הסדרות

היא $O(s_1+\dots+s_k)=O(m)$ במקרה הגרוע, ולכן סיבוכיות זמן הריצה המשוערכת של EnQ(x) ו-DeQ(x) היא $O(1)$ במקרה הגרוע.

מ.ש.ל.

הוכחה בעזרת שיטת החיובים: נגדיר תשלומים באופן הבא:

- $EnQ(x)$: $a_i=4$ המחיר המשוערך של $EnQ(x)$ (כל איבר שנכנס ל-MAIN חייב לצאת דרך QUEUE, ולכן נשלם עבור push אל MAIN, לאחר מכן pop מ-MAIN, לאחר מכן push למחסנית QUEUE, ואז pop נוסף במידה ונוציא את איבר זה בעתיד).
- $DeQ()$: $a_i=1$ המחיר המשוערך של $DeQ()$.

עבור $\boldsymbol{EnQ(x)}$ המחיר בפועל הוא $t_i=1$ (בדיוק כמו Push() רגיל). את היתרה (3) נצמיד לאיבר x. המחיר בפועל של **הפעולה** $\boldsymbol{DeQ()}$ תלויה בהאם $QUEUE$ ריקה או לא. **אם היא ריקה**, אז המחיר בפועל יהיה $t_i=2k_i+1$ כאשר k_i זה כמות האיברים הנוכחית ב-MAIN: לכל אחד מאיברים אלו יש יתרה של של 3. ניקח מכל איבר יתרה של 2 עבור ההוצאה מ-MAIN וההכנסה ל-QUEUE (מכאן בא ה-$2k_i$). בנוסף ניקח את היתרה 1 שנותרה לאיבר שאנו בפועל מוציאים. כלומר, נשלם את פעולה זו במקרה זה על ידי שימוש ביתרה שהותרנו בכל אחד מהאיברים. **אחרת, אם** $\boldsymbol{QUEUE}$ **לא ריקה**, המחיר בפועל הוא $t_i=1$ שזהו היתרה שנותרה לאיבר שאנו מוציאים מראש המחסנית $QUEUE$ (כל איבר שעובר מ-MAIN ל-QUEUE נותר עם יתרה של 1). לכן נשלם את פעולה זו בעזרת היתרה שנותרה. בשני המקרים, לא נשלם על פעולה זו כלל, כי יש לנו מספיק יתרה בבנק. נקבל:

$$0=\sum_{i=1}^{m}(a_i-t_i)$$

ניתן לחסום את זמן הריצה על ידי:

$$\sum_{i=1}^{m}(t_i)\le\sum_{i=1}^{m}(a_i)\le 4m=O(m)$$

מ.ש.ל.

הוכחה בעזרת שיטת הפוטנציאל: נגדיר את פוטנציאל המבנה: $\phi_i=3|D_i|$ כאשר $|D_i|$ זה מספר האיברים במחסנית $MAIN$ לאחר i פעולות. נראה שהמחיר המשוערך a_i חסום על ידי קבוע עבור שני פעולות המבנה:

- $EnQ(x)$: $a_i=t_i+\phi_i-\phi_{i-1}=1+3=4$. כלומר, לאחר הפעולה $EnQ(x)$ שעולה לנו בפועל $t_i=1$, הגדלנו את הפוטנציאל ב-3 כי הוספנו איבר אחד למחסנית $MAIN$.
- $DeQ()$: $a_i=t_i+\phi_i-\phi_{i-1}$. נבצע חלוקה למקרים:
  - **אם** $\boldsymbol{QUEUE}$ **ריקה,** מתקיים ש-

$$a_i=t_i+0-3|D_{i-1}|=2|D_{i-1}|+1-3|D_{i-1}|=1-|D_{i-1}|\le 1$$

    במקרה זה $t_i=2|D_{i-1}|+1$ מכיוון שזאת העלות בפועל שתעלה לנו להעביר את $|D_{i-1}|$ האיברים מ-$MAIN$ אל $QUEUE$ (הוצאה מ-MAIN והכנסה ל-QUEUE עבור כל אחד מהאיברים) ואז להוציא את האיבר שבראש $QUEUE$. בנוסף $\phi_i=0$ כי לאחר פעולה זו, מחסנית $MAIN$ מתרוקנת (ככה מוגדר המבנה כאשר $QUEUE$ ריקה).
  - **אם** $\boldsymbol{QUEUE}$ **אינה ריקה**, מתקיים ש-

$$a_i=t_i=1$$

    כלומר יעלה לנו רק להוציא את האיבר מ-$QUEUE$, ובכך גם לא שינינו את הפוטנציאל כי כמות האיברים ב-$MAIN$ נשארת זהה.

מכיוון ש-$3|D_m|=\phi_m\ge\phi_0=0$ נקבל:

$$\sum_{i=1}^{m}(a_i - t_i) = \sum_{i=1}^{m}(\phi_i - \phi_{i-1}) = \phi_m - \phi_0$$

ולכן:

$$\sum_{i=1}^{m}(t_i) = \sum_{i=1}^{m}(a_i) + \phi_0 - \phi_m \leq \sum_{i=1}^{m}(a_i) \leq 4m = O(m)$$

מ.ש.ל.