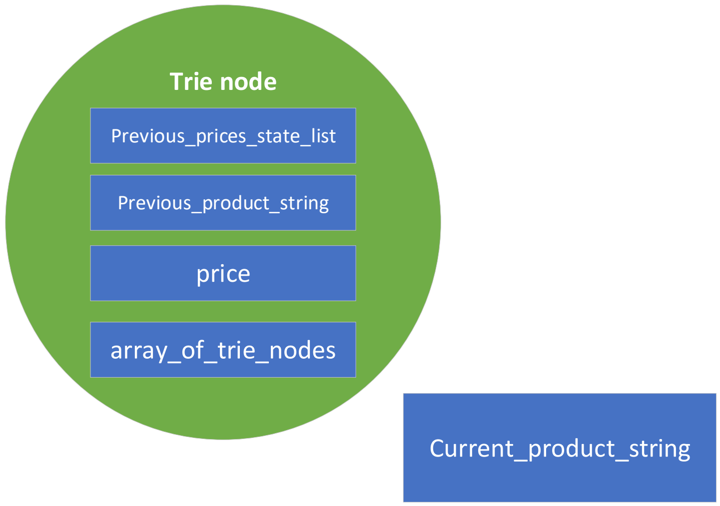
שאלה 4

חלק ראשון

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

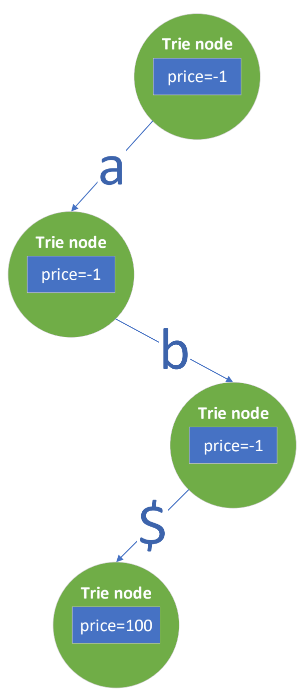


בנוסף, נשמור משתנה נוסף בצד ששמו $current\_product\_string$.

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:

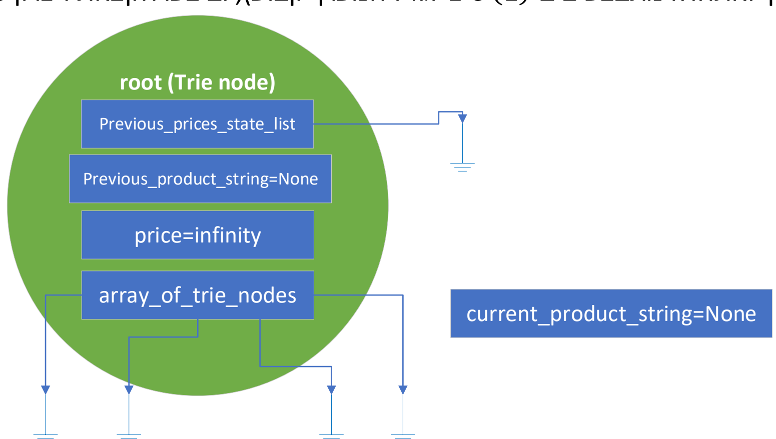
- $\boldsymbol{price}$: אם הצומת הוא הצומת האחרון שמייצג את המוצר בעץ (הצומת שמגיע לאחר $), ערך זה יכיל את מחיר המוצר. אחרת, ערך זה יכיל את המחיר המינימלי מבין כל המוצרים החופפים שצומת זה מייצג. לדוגמא עבור העץ המכיל את המחרוזות *abcd, abce, abcf*, הצומת המגיע לאחר הקשת *c* יכיל ב-price את המחיר המינימלי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



- $\boldsymbol{previous\_product\_string}$: 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).
- $\boldsymbol{array\_of\_trie\_nodes}$: מערך באורך $|\Sigma| + 1$ שמחזיק מצביעים לבנים (כפי שלמדנו בתרגול).
- $\boldsymbol{previous\_prices\_state\_list}$: 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 $Trie$ ע"י 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 $\boldsymbol{O(|desc|)}$.**

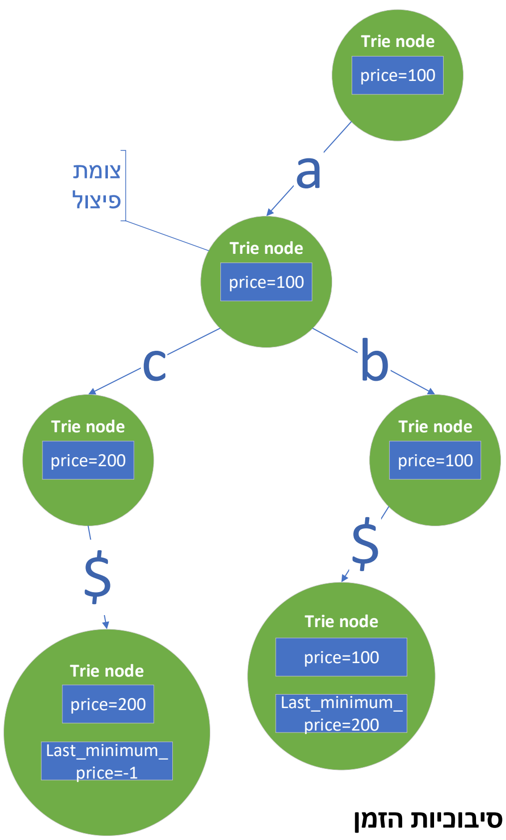
- $\boldsymbol{Undo()}$**:** אם $current\_product\_string$ אינו מכיל דבר (None), זה אומר שעוד לא הכנסנו אף מחרוזת למבנה הנתונים, ולכן לא נבצע דבר. אחרת, נייצר משתנה חדש בשם $current\_undo\_desc$, ונעביר אליו את המחרוזת הנמצאת ב-$current\_product\_string$. על פי הגדרת המבנה שלנו, כל צומת שהוא סיומת של 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-$s = c_1c_2 \ldots c_n$. נעבור על המסלול בגרף $c_1c_2, \ldots c_n$: במידה ועבור צומת כלשהו לא קיימת קשת c_k $(1 \leq k \leq n)$, 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-$CheapestOfType(a)$:

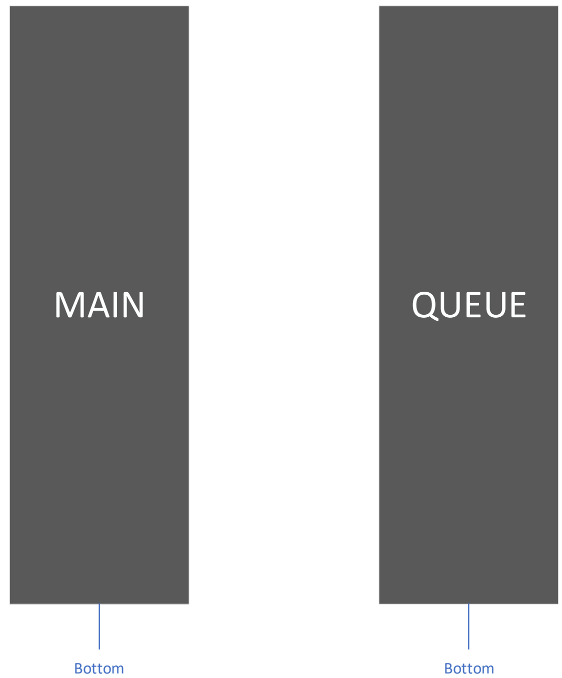


הצומת שלאחר הקשת a מכיל את המחיר 100, שהוא המחיר המינימלי מבין המוצרים ab ו-ac.

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

שאלה 5

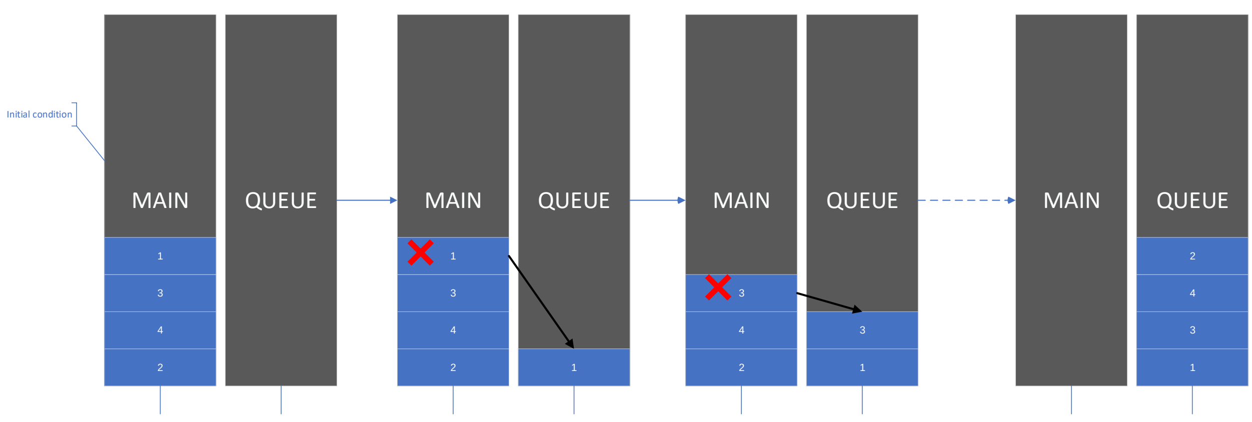
נשתמש ב-2 מחסניות LIFO לצורך המימוש של מחסנית ה-FIFO. להלן מחסנית MAIN ומחסנית QUEUE (הסימון Bottom מסמל את תחתית המחסניות):



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). __**אם QUEUE ריקה**__, 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 בשנסיים 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. **__אחרת, אם QUEUE אינה ריקה__**, 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 $s_1 + s_2 + \cdots + s_k = m$. 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 $O(s_1 + \cdots + s_k) = O(m)$ 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:

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

מ.ש.ל.