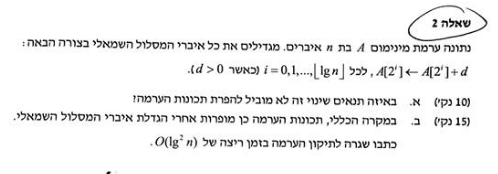


**השוואות:**

חוץ מהאיבר הראשון, כל איבר במיקום אי-זוגי יושווה רק פעם אחת לאיבר שלפניו וכל איבר במיקום זוגי יושווה פעמיים (לשני האיברים שלפיו) עד שיגיע למקום שלו.

**העתקות:**

רק המספרים האי-זוגיים לנמצאים במקום שלהם , ויועתקו רק פעם אחת שמאלה



**סעיף א**

השינוי לא יוביל להפרת תכונות הערימה כל עוד האיבר ששונה עדיין יותר קטן מהבנים שלו – הבעיה צריכה להיות קיימת רק מול הבן הימני, כי מול השמאלי גם האב וגם הבן גדלו ב-d

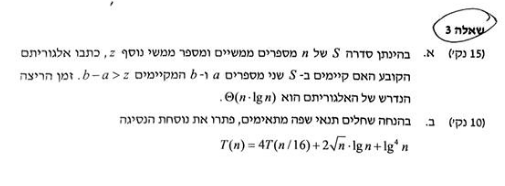
**סעיף ב**

לכל צומת במסלול השמאלי נפעיל את הפונקציה Heapify-down(S,x)

Heapify-down(S,x)

1. If key[x] > key[left[x]] or key[x] > key[right[x]] then
   1. If key[left[x]] < key[right[x] then
      1. Swap(x,left[x])
      2. Heapify-down(S,x)
   2. Else
      1. Swap(x,right[x])
      2. Heapify-down(S,x)

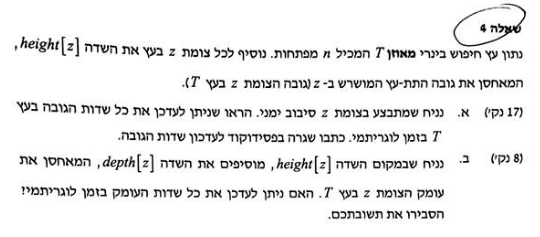
הפונקציה רצה על כל האיברים במסלול השמאלי כלומר O(lgn) ועבור כל איבר יכולה להוריד אותו במורד המסלול ולכן יש לנו עוד O(lgn) ולכן סה"כ זמן הריצה של הפונקציה במונחים אסימפטוטיים הינו O(lg2n)



**סעיף א**:

1. נמיין את סדרת המספרים למערך ממוין בעזרת אלגוריתם מיון אופטימלי ()
2. נבדוק את ההפרש של האיבר האחרון והאיבר הראשון, אם הוא >z נחזיר True
3. אחרת נחזיר FALSE

**סעיף ב**



**סעיף א**



הערך של A , B , C לא משתנה

הערך של Q הופך להיות המקסימום מהערכים של B , C + 1

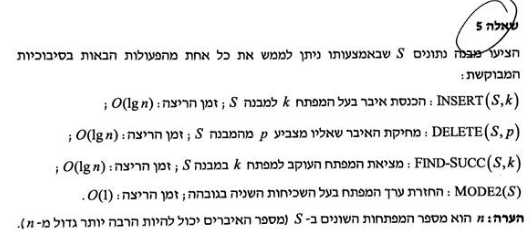
הערך של P הופך להיות המקסימום מהערכים של A,Q + 1

\*\* כל מה שנשאר זה לעדכן את העץ מ-P ועד לשורש כך שבכל צומת ניקח את המקסימום משני הבנים+1

כל הפעולות חוץ מ\*\* מתבצעות ב-O(1) ו-\*\* מבצעת בזמן לוגריתמי על גובה העץ T כלומר כל האלגוריתם רק ב-

**סעיף ב**

לא ניתן לבצע באופן לוגריתמי. בעת סיבוב ערך ה-depth נדרש להתעדכן עבור כל צומת בתת עץ ולא רק עבור הצמתים במסלול מסוים ולכן במקרה הגרוע נידרש לעדכן כל הצמתים בעץ ולכן העלות היא לינארית



נשתמש בעץ אדום שחור T עבור המפתחות השונים , כאשר כל צומת מכילה רשימה דו-מקושרת עבור כל האיברים עם אותו המפתח.

נשתמש בעוד עץ אדום שחור Q שהמפתחות בו הינם השכיחות של המפתחות בעץ T , ונחזיר מצביע לערך השני בגודלו. בין העצים T ו-Q ישנן הצבעות דו כיווניות.

Insert(S,k) –

1. הכנסה לעץ T , אם נתקלים במסלול בצומת עם אותו הערך נוסיף את הערך לרשימה נעדכן את הערך של המפתח הזה בעץ Q ואם נדרש גם את מצביע q
2. אם מגיעים לעלה אז יש לנו מפתח חדש-
   1. הוספת המפתח לראש הרשימה בצומת החדש של העץ T
   2. הוספת המפתח עם הערך 1 לעץ Q
   3. עדכון העץ Q
   4. עדכון המפתח q אם נדרש

כל הפעולות כן הינן פעולת של הכנסה לעץ אדום שחור \ עדכן ראש רשימה מקושרת \ מצביעים ולכן העבודה הינה O(lgn)

Delete(S,k) –

1. חיפוש ערך המפתח של p בעץ T
2. מחיקת p מהרשימה המקושרת
3. אם הוא האחרון מחיקת המפתח מ-T ומ-Q
4. אחרת חיסור 1 מהערך המתאים ב-Q ועדכון העץ והמצביע q

כל הפעולות הינן פעולות של מחיקה מעץ אדום שחור ומחיקה מרשימה דו-מקושרת שיש לנו מצביע לערך הנמחק ולכן מחיקה מתבצעת ב-O(lgn)

Findsuccessor(S,k)

החזרת הערך העוקב בעץ אדום שחור עם הפונקציה הרגילה שלו

Mode2(S,k)

החזרת ערך המצביע q