**ממ"ן 17**

**שאלה 1**

**סעיף א'**

נניח כי קיימת לנו רשימה של מפתחות (skey,pkey).

אם ברצוננו לבנות עץ SP כפי שמוגד בשאלה, ברור כי השורש של העץ יהיה צמד המפתחות ברשימה שבו pkey הכי קטן - מכיוון שהעץ שומר על תוכנת ערימת המינימום.

עתה נתבונן על המפתחות skey, על מנת לשמור על תכונת העץ חיפוש הבינארי:

* כל הצמדי מפתחות שבהם skey קטן מהשורש - יהיו בצד שמאל של השורש.
* כל הצמדי מפתחות שבהם skey גדול מהשורש - יהיו בצד ימין של השורש.

ובעצם, חילקנו את הבעיה ל 2 רשימות של צמדי מפתחות, ועתה בכל רשימה ברור שוב מי יהיה בראשה - המפתח pkey הכי קטן ברשימה, וכמו כן שוב תתבצע אותה חלוקה לפי skey.

אתה רומז להוכחה באינדוקציה, אבל שכחת כמה דברים הדרך: להראות שהתהליך עוצר, ולהראות שכשהתהליך עוצר ניתן לבנות עץ כנדרש. נסח את ההוכחה בצורה פורמלית כהוכחה באינדוקציה ולא תשכח כלום. 4-

לסיכום:

pkey - קובע בכל פעם מי יהיה הצומת בראש הרשימה הנותרת.

skey - קובע את החלוקת מפתחות לימין ולשמאל.

מכיוון שכל המפתחות pkey שונים זה מזה, בכל שלב ישנה רק אפשרות אחת ויחידה לבחירת המפתח בעל הpkey הקטן ביותר, ולכן קיים מבנה SP יחיד כזה.

**סעיף ב'**

T - העץ שלנו.

K - המפתח אשר מכיל בתוכו (skey,pkey)

נשתמש בשיטה Tree-Insert(T,key) עם שינוי קטן שבו בסוף השיטה יוחזר הצומת שהכנסנו.

כמו כן השיטה תכניס את האיבר לפי ה skey כפי שמוגדרת.

**Insert-SP(T, K)**

**1.** x <-- Tree-Insert (T,key)

**2.** while x-not-root[T] and pkey[x] < pkey [ p[x] ] do

**3.** if right[p[x]] = x then

**4.** Left-Rotate(T,x)

**5.** if left[p[x]] = x then

**6.** Right-Rotate(T,x)

נכונות השגרה:

השגרה מכניסה איבר לעץ לפי ה skey, ובכך שומרת על נכונות תכונת העץ חיפוש הבינארי - הסדר תוכי נשמר ( שורה 1 ), ולכן נותר לבדוק האם תכונת הערימת מינימום נשמרת.

שורה 2 מוודאת שהצומת שהוכנס יהיה גדול מאביו ( תוכנת ערימת המינימום ), במידה ולא, נשתמש ברוטציות על מנת לתקן את העץ. הלולאה כמובן תעצר בשלב מסויים כאשר האב יהיה אכן קטן מ x וישמור על התכונה של ערימת המינימום, או כאשר x יטפס עד השורש, ושם יעצר.

אחד התכונות החזקות של רוטציה - היא בכך שהיא שומרת על סדר תוכי, ולכן בכל רוטציה שנעשה תשמר תוכנת העץ חיפוש הבינארי.

* אם הצומת הוא בן ימני - נבצע רוטציה לשמאל (שורות 3-4).
* אם הצומת הוא בן שמאלי - נבצע רוטציה לימין (שורות 5-6).

בכל שלב בלולאה, אם הצומת גדולה מאביו אנו מעלים אותו כלפי מעלה, ובכך דואגים שבסוף הלולאה, הצומת יקבל את מיקומו המתאים בעץ, כך שהסדר תוכי נשמר, כל הצמתים שמעליו קטנים ממנו וכל הצמתים שמתחתיו גדולים ממנו.

כל מה שנותר להוכיח הוא - שהרוטציה אינה משנה את התכונה של שאר ערימת המינימום.

יהי x צומת בערימת מינימום, ויהי y אביו של x.

אם נבצע על x רוטציה לשמאל:

* כל הבנים השמאליים של y נשארים כפי שהם - ולכן קטנים ממנו ושומרים על התכונה.
* y מקבל את הבן הימני של x. מאחר ובן ימני זה היה בן כלשהו של y, כמובן שקטן ממנו, ולכן שוב נשמרת התכונה.
* וכמובן שהבנים השמאליים של x היו קטנים ממנו - ונשארו בניו.

לכן היחס היחידי שנשאר לבדוק אם השתנה זה האם x אכן קטן מ y, אך בשורה 2, דאגנו לבצע את הרוטציה אך ורק אם אכן x קטן מ y , ולכן אחרי הרוטציה נקבל ש x קטן מ Y ונשמרה לנו התכונה של ערימת המינימום.

כלומר כל התכונות של SP נשמרות - ועל כן האלגוריתם נכון.

סיבוכיות השגרה:

* Tree-Insert (T,key) - מתבצעת בסיבוכיות של O(n)במקרה הגרוע.
* לולאת ה While - מעלה בכל שלב את הצומת x קומה אחת למעלה, ולכן סה"כ יכולה לעלות רק n-1 קומות - כלומר סיבוכיות של O(n) במקרה הגרוע.
* Left-Rotate(T,x) ו Right-Rotate(T,x) - מתבצעים ב O(1).

ולכן סה"כ סיבוכיות השגרה :

**שאלה 2**

**סעיף א'**

על מנת ליצור עץ אדום שחור אשר מקיים את התנאים:

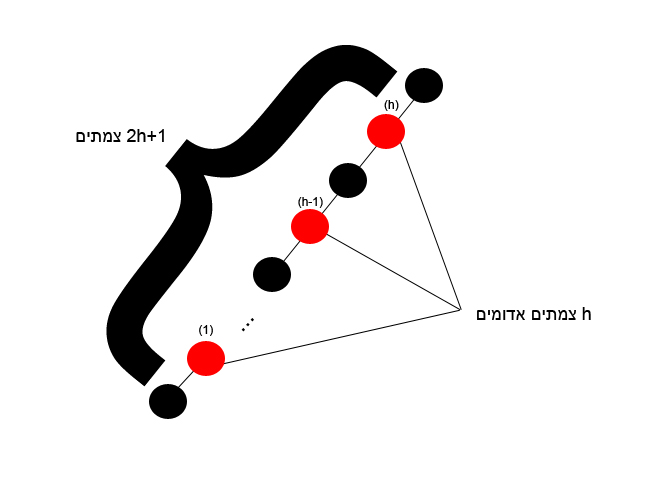
1. גובה העץ T הוא 2h.
2. העץ T מכיל h צמתים אדומים בדיוק - וכולם נמצאים על המסלול השמאלי שלו.

נתחיל בתיאור המסלול השמאלי:

ע"פ הגדרת עץ אדום שחור -

* השורש תמיד שחור.
* לכל בן אדום יש 2 בנים שחורים.

ולכן הדרך היחידה ליצור את המסלול השמאלי כך שיופיעו בו h צמתים אדומים וגובהו יהיה 2h הוא בצורה הבאה:



כלומר במסלול השמאלי , ישנם h צמתים אדומות, h+1 צמתים שחורים, ובסה"כ - 2h+1 צמתים.

חשוב לזכור שעץ בעל צומת אחת - גובהו 0, ולכן עץ בעל 2h+1 צמתים - גובהו 2h.

עץ כזה כמובן מקיים את כל ארבעת התכונות של עץ אדום שחור, עתה נתרכז בתיקון העץ כך שיתאים גם לתכונה החמישית, אשר מספרת כי בכל מסלול פשוט מצומת לעלים , יש אותו מספר עלים שחורים.

אם נמספר את העלים האדומים כפי שמספרנו בתמונה, נשים לב שלעלה האדום האחרון (1) , ולאביו, חייבים להיות בנים ימניים שחורים על מנת שיתקיים התכונה - כלומר מוסיפים כבנים ימניים תת עץ שחור מלא בגובה 0.

לעלה אחריו (2), ולאביו חייבים להיות בנים ימניים אשר הם תת עץ שחור מלא מגובה 1 - זאת כדי שבכל מסלול של צומת (2) ואביו לעלה פשוט יהיה בדיוק 2 צמתים שחורים.

וכן הלאה עד שלצומת אדום (h) ולאביו ישנם בנים ימניים שהם תת עץ שחור מלא מגובה h-1.

בצורה זאת בכל מסלול פשוט מצומת לעלים נקבל אותו מספר עלים שחורים. וכך בנינו עץ אדום שחור אשר מקיים את התכונות הנ"ל.

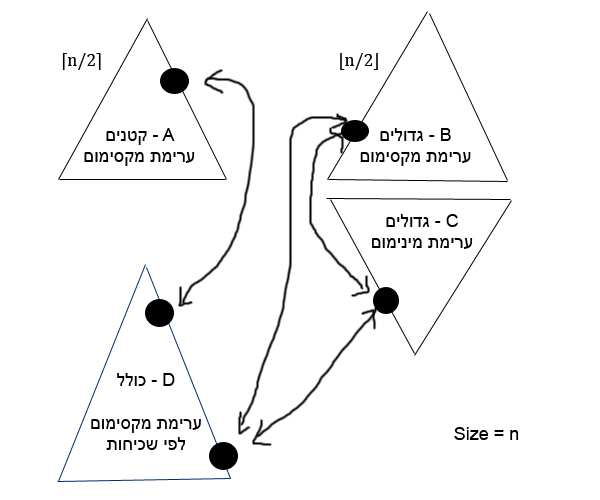
**סעיף ב'**

מספר הצמתים בעץ מלא מגובה h הוא : .

אם נספור רק את הצמתים שאנחנו מוסיפים למסלול השמאלי, אנו בעצם מוסיפים לכל עלה אדום i ולאביו - תת עץ ימני מגובה i-1. כלומר סכום כל התתי עצים שאנו מוסיפים הוא:

וכמובן, הראנו מקודם שמספר הצמתים במסלול השמאלי הוא , ואם כן, קיבלנו נוסחה לחישוב מספר הצמתים לפי גובה העץ והיא:מהו *h* כתלות ב-*n*? *h*=lg(*n*+3)-2 3-

**שאלה 3**

****

**מבנה הנתונים הוא:**

חלוקת הסדרה לשני רשימות של מספרים גדולים ומספרים קטנים.

* במספרים הקטנים - בונים ערימת מקסימום.
* במספרים הגדולים - בונים ערימת מינימום וערימת מקסימום, ומותחים מצביעים בין האיברים הדומים.

כמו כן, ערימת מקסימום נפרדת אשר מורכבת משני מפתחות, מפתח אשר אומר מהו המספר, ומפתח אשר אומר מה השכיחות של המספר - הערימת מקסימום היא לפי השכיחות. וכמו כן גם בינה לבין הערימות יש מצביעים עבור האיברים הדומים.

שמירת משתנה Size אשר מחזיק את מספר האיברים הכולל.

**תיאור הפעולות:**

בניה - Build(S)

1. משתנה size מקבל את גודל הרשימה.

2. נחלק את הרשימה הנתונה ל איברים קטנים ונבנה מהאיברים ערימת מקסימום A. O(n)

3. נחלק את הרשימה הנתונה ל איברים גדולים, ונבנה מהאיברים ערימות מקסימום B וערימת מינימום C. O(n)

4. נמתח מצביעים בין האיברים המתאימים בערימות המקסימום והמינימום של האיברים הגדולים.

5. במקביל, נבנה ערימת מקסימום D ע"פ שכיחות האיברים ( ערימה בעלת 2 מפתחות, מספר האיבר וכמות השכיחות ) ונמתח מצביעים בין האיברים המתאימים אשר נמצאים בערימות.

הסבר על שלב 5: בכל הכנסה של איבר לערימה, נכניס אותו בהתאם גם לערימת המקסימום ונמתח מצביעים אחד לשני, עקב כך שהאיברים **ממויינים** נוכל בקלות לדעת האם להוסיף איבר חדש לערימה או רק לעלות את השכיחות של איבר קיים ב 1, ולכן, גם זמן זה הוא ליניארי - o(n)

סה"כ זמן ריצה : - בניית ערימות נעשה בזמן ליניארי.

הכנסה - Insert(S,k)

1. בדיקה לאיזה ערימה יש להכניס את האיבר:

* אם size זוגי - נכניס לערימת האיברים הקטנים A.
* אם size אי זוגי - נכניס לערימות האיברים הגדולים B ו C, מותחים ביניהם מצביעים.

2. מכניסים את האיבר לערימה הנכונה ומבצעים Heapify - תוך כדי בודקים האם האיבר כבר קיים בערימה או אם הוא במקסימום של A או במינימום של C.

* אם כן - בעזרת המצביע לערימה D מעלים לאיבר זה את השכיחות ב 1 ובמצעים Heapify.
* אם לא - מוסיפים עלה חדש לערימה D עם המספר ושכיחות ברירת מחדל - 1,

וכמובן שמותחים מצביעים בהתאם.

3. מעלים את Size ב 1.

סה"כ זמן ריצה : - הכנסות לערימות.

מחיקה - Delete(S,z)

1. בדיקה אם יש צורך לתקן את הערימה אחרי המחיקה.

* אם Size זוגי ואנו מוחקים מערימת A - אז המינימום ב C יעבור לערימה A ונבצע Heapify,

כמו כן נדאג למחוק את האיבר שהעברנו גם ב B ולבצע Heapify.

* אם Size אי זוגי ואנו מוחקים מערימת B - אז המקסימום ב A יעבור לערימות B ו C ונבצע Heapify על 3 הערימות.
* שאר המקרים - מחיקה רגילה מערימה.

2. מחיקת המצביעים המתאימים בין C ל B, כמו כן בעזרת מצביע לערימה D, נבדוק:

* אם השכיחות של האיבר היא 1 - נמחוק אותו מהערימה ונבצע Heapify.
* אם השכיחות של האיבר גדולה מ 1 - נוריד את השכיחות שלו ב 1 ונבצע Heapify.

3. הורדת Size ב 1.

סה"כ זמן ריצה : - מחיקה מערימות.

מחיקת חציון - Del-Median(S)

1. בדיקה אם יש צורך לתקן את הערימה :

* אם size זוגי - נמחק את המקסימום של A ונשים במקומו את המינימום של C.
* אם size אי זוגי - נמחק את המקסימום של A ונבצע Heapify.

2. מחיקת האיבר גם ב B אם צריך, כמו כן בעזרת מצביע לערימה D, נבדוק :

* אם השכיחות של האיבר היא 1 - נמחוק אותו מהערימה ונבצע Heapify.
* אם השכיחות של האיבר גדולה מ 1 - נוריד את השכיחות שלו ב 1 ונבצע Heapify.

3. הורדת Size ב 1.

סה"כ זמן ריצה : - מחיקה מערימות.

החזרת המפתח השכיח - Mode(S)

1. החזרת המקסימום בערימת D - O(1).

סה"כ זמן ריצה : - החזרת מקסימום בערימת מקסימום.

מכיוון שלא נדרשת בנייה מרשימה לא ממויינת בזמן לינארי, הרבה יותר פשוט להשתמש בעץ א"ש.

**שאלה 4**

**מבנה הנתונים הוא:**

עץ אדום שחור A בעל מפתחות k.

בכל צומת בעץ יש משתנה Sum אשר סוכם את כל המפתחות של אותו תת עץ שמתחתיו.

כמו כן, ישנו עץ אדום שחור נוסף B, עם מצביעים מתאימים בין שני העצים, אשר מכיל 2 מפתחות:

* m - מיקום הכנסת האיבר
* k - מפתח האיבר

עץ זה ממוין לפי המיקומים m, כלומר ישנו counter שבכל הכנסה עולה ב 1, כך שבהכנסה לעץ A מוכנס איבר גם לעץ B והm מקבל את ערכו של ה counter ובכך מצביע על מיקומו להכנסת העץ, וכמובן שומר את המפתח במיקום זה.

**תיאור הפעולות:**

הכנסה - Insert(S,k)

1. הכנסה ל A בצורה הבאה: מבצעים הכנסה רגילה לעץ, מאפסים את sum ל 0 ו n ל 1.

מכניסים גם לעץ B בצורה רגילה עם הערך של counter ומותחים מצביעים בין העצים.

2. אחרי ההכנסה בעץ A מבצעים את הדברים הבאים :

* מוסיפים ל sum בצומת הנ"ל את הערך k + את ה sum של הבנים
* עולים לכל האבות ומעלים לכולם k בשדה sum.

3. מעלים את counter ב 1.

סה"כ זמן ריצה : - הכנסות לעץ אדום שחור.

מחיקה - Delete(S,x)

1. מחיקה בצורה רגילה מצומת A ובמקביל מצומת B.

2. טיפוס לכל האבות של הצומת והורדת הערך k משדה ה sum שלהם.

סה"כ זמן ריצה : - מחיקה לעץ אדום שחור.

הפרש d - Pair-Diff(S,d)

נגדיר פעולה S-To-Array אשר סורקת את העץ בסדר תוכי (ממויין) לרשימה .

ולאחר מכן יוצרת מערך ומכניסה את הערכים למערך ללא כפילויות .

Pair-Diff(S,d)

**1.**Array <-- S-To-Array

**2.** i<--1

**3.** j<--2

**4.** while j≤ Array.length do

**5.** if Array[j]-Array[i] = d then

**6.** return (j,i)

**7.** else if Array[j]-Array[i] < d

**8.** j<--j+1

**9.** else

**10.** i<--i+1

**11.** if i=j then

**12.** j<--j+1

**13.** return NIL

נכונות השגרה

מתחילים משני האיברים הסמוכים הראשונים. אם ההפרש ביניהם קטן מ d , אז מעלים את j באחד.

אם ההפרש ביניהם גדול מ d, אז אין טעם לבדוק על שאר האיברים אשר גדולים מ Array[j] כי ההפרש רק ילך ויגדל, ולכן לאיבר Array[i] אין איבר שמתאים לו כך שההפרש יהיה d, ואפשר להגדיל את i באחד.

כך אנחנו מתקדמים במערך עד אשר מוצאים שני איברים אשר מקיים את הפרש d, או עד אשר מגיעים לסוף המערך - כלומר לא מצאנו זוג איברים כזה.

סיבוכיות

בכל שלב או i או j גדלים ב 1, ולכן במקרה הגרוע יהיו 2N חזרות על לולאת ה while, כלומר הסיבוכיות היא .

הפרש d - Sum(S,k)

1. מתחילים מהשורש, ובכל צומת x שעוברים שואלים האם ערך הצומת קטן מ .k

* אם כן - מוסיפים ל Sum-K את sum[x]-sum[right[x]], וזזים לבן הימני.
* אם לא - זזים לבן השמאלי.

2. ממשיכים עד שמגיעים ל NIL, ומחזירים את הערך Sum-K.

סה"כ זמן ריצה : - טיול בעץ אדום שחור.

האיבר ה m הותיק - Old(S,m)

1. עוברים על עץ B ומחפשים את ערך m- אם נמצא הערך m אז מחזירים את הערך k שלו.

2. אם לא נמצא הערך k (משמע הגענו לצומת x שהוא NIL) אז:

* אם x=right[p[x]] (בן ימני) - הולכים בעץ אחד שמאלה וימינה עד הסוף.
* אם x=left[p[x]] (בן שמאלי)- אז הולכים אחד ימינה.

ובמילים אחרות - הולכים לעוקב בסדר תוכי, ומחזירים את הערך הזה.

סה"כ זמן ריצה : - טיול בעץ אדום שחור.

תמיר אביב,

305652000