מבני נתונים – עצי חיפוש בינאריים
דף תרגילים

שאלה 1

תארו אלגוריתם המקבל m עצי חיפוש בינריים שבכל אחד מהם n איברים, ומאחד אותם לעץ חיפוש בינארי אחד.

סיבוכיות זמן הריצה הנדרשת היא O(nmlgm).

שאלה 2

עליכם לתכנן מבנה נתונים לאחסון ותחזוקת קבוצת רשומות בעלות שני מפתחות. לכל רשומה R נציין ב-$key1[R]$ את המפתח הראשון וב-$key2[R]$ את המפתח השני. ידוע שהמספר הכולל של רשומות R בקבוצה הינו N; מספר הערכים השונים של המפתח הראשון $key1[R]$ הוא n_1 ומספר הערכים השונים של המפתח השני $key2[R]$ הוא n_2.

הציעו מבנה נתונים S לאחסון כל רשומות הקבוצה, שבאמצעותו ניתן לממש כל אחת מהפעולות הבאות בזמנים הנדרשים:

1. SEARCH (k_1,k_2,S): חיפוש ב- S אחר רשומה כלשהי R בעלת זוג המפתחות (k_1,k_2); זמן הריצה $O(\lg n_1 + \lg n_2)$;
2. INSERT (k_1,k_2,S): הכנסה ל- S של רשומה כלשהי R בעלת זוג המפתחות (k_1,k_2); זמן הריצה $O(\lg n_1 + \lg n_2)$;
3. DELETE (p,S): מחיקה מ- S של הרשומה R, שאליה מצביע p; זמן הריצה $O(\lg n_1 + \lg n_2)$;
4. MAX2 (S): מציאה, בין כל הרשומות R של S, של הערך המכסימלי של המפתח השני $key2[R]$; זמן הריצה $O(1)$;

שאלה 3

תארו אלגוריתם המקבל מצביע לשני עצים בינריים t_1 ו-t_2, ומחזיר true אם ורק אם ניתן להפוך את t_1 ל-t_2 בעזרת רוטציות בלבד.

זמן הריצה הנדרש של האלגוריתם O(n), כאשר n הוא מספר הצמתים הכולל בשני העצים.

מבני נתונים – עצי חיפוש בינאריים
דף תרגילים – המשך

שאלה 4

הוכיחו או הפריכו:

כאשר מבטלים שני איברים בעץ חיפוש בינרי, סדר ביטול האיברים אינו משפיע על מבנה העץ המתקבל.

שאלה 5

נניח שבעץ חיפוש בינרי מאוכסנים מספרים בין 1 ל- 1000, וברצוננו לחפש את המספר 363.

אילו מבין הסדרות הבאות **אינן** יכולות להיות סדרות של צמתים שנבחנו במהלך החיפוש? (קראו משמאל לימין).

1. 2, 252, 401, 398, 330, 344, 397, 363
2. 924, 220, 911, 244, 898, 258, 362, 363
3. 925, 202, 911, 240, 912, 245, 363
4. 2, 399, 387, 219, 266, 382, 381, 278, 363
5. 935, 278, 347, 621, 299, 392, 358, 363

שאלה 6

כתבו אלגוריתם המקבל עץ חיפוש בינרי וערך x, ומחזיר מצביע לצמת שערכו קרוב ביותר לערך x.

שאלה 7

כתבו אלגוריתם המקבל עץ חיפוש בינרי וערך x, ומחזיר מצביע למופע הראשון של x בעץ (המופע הראשון הוא המופע הראשון שייסרק בסדר תוכי). אם x לא בעץ יוחזר nil.

שאלה 8

תארו אלגוריתם המקבל עץ חיפוש בינרי וערך x, ומחזיר את המופע של x הקרוב ביותר אל השורש. אם x לא בעץ, יוחזר nil.

שאלה 9

תארו אלגוריתם המקבל עץ חיפוש בינרי וערך x, ומחזיר את המופע של x שייסרק ראשון בסריקה בסדר תחילי. אם x לא בעץ, יוחזר nil.

מבני נתונים – עצי חיפוש בינאריים
דף תרגילים – המשך

שאלה 10

תארו אלגוריתם שמקבל עץ AVL וערך x, ומוחק מהעץ את כל המופעים של x. זמן הריצה של האלגוריתם $O(\lg n)$.

שאלה 11

נתאר אלגוריתם חלופי עבור מחיקת צומת z מעץ חיפוש בינרי.

במקרה השלישי, כאשר לצומת שני בנים, מאתרים את העוקב שלו y, ואז מחליפים בין $left[y]$ לבין $left[z]$; עכשיו אפשר להסיר את z כמו במקרה השני.

א. תארו יתרון אחד וחיסרון אחד לפחות של אלגוריתם זה יחסית לאלגוריתם המחיקה המקורי (המתואר בספר).

ב. האם ניתן להשתמש באלגוריתם החדש למחיקת צומת מעץ AVL?

שאלה 12

הציעו מבנה נתונים המאפשר לבצע את הפעולות הבאות בזמנים הנדרשים:

(S,z) INSERT: הכנסת איבר בעל המפתח z למבנה S; זמן: $O(\lg n)$;

(S,p) DELETE: מחיקת האיבר שאליו מצביע p מהמבנה S; זמן: $O(\lg n)$;

(S,z,k) SUCCESSOR: מציאת העוקב ה- k של האיבר z במבנה S; זמן: $O(\lg n)$.

הסבר: העוקב ה- k של z הוא העוקב של העוקב (k פעמים) של z.

שאלה 13

א. נתון מערך ממוין A בגודל n.

כתבו אלגוריתם רקורסיבי בשם BUILD-TREE, אשר יבנה מאיברי המערך עץ חיפוש בינרי **מאוזן**. זמן הריצה של האלגוריתם צריך להיות $O(n)$.

ב. נתונים שני עצי חיפוש בינריים. כל אחד מהעצים מכיל n איברים.

ברצוננו להעביר את כל $2n$ האיברים לעץ חיפוש בינרי מאוזן (עץ החיפוש המאוזן ההתחלתי הוא ריק). הסבירו איך ניתן לבצע את המשימה בזמן $O(n)$.

מבני נתונים – עצי חיפוש בינריים
פתרונות דף התרגילים - למרצה

עצי חיפוש בינאריים – פתרון שאלה 1

האלגוריתם יסרוק, בסדר תוכי, כל עץ למערך ממוין.

כל עוד מספר המערכים גדול מ-1:

האלגוריתם יחלק את המערכים לזוגות וימזג כל זוג

(אם יש מספר אי-זוגי של מערכים אז המערך האחרון יישאר ללא שינוי).

מכיוון שבתחילת האלגוריתם מספר המערכים הוא m, מספר האיטרציות הוא O(lgm). זמן הריצה של כל איטרציה לינארי במספר האיברים: O(mn). סה"כ זמן הריצה: O(mnlgm).

עצי חיפוש בינאריים – פתרון שאלה 2

מבנה הנתונים יהיה עץ AVL לפי k_2 שבכל צמת שלו מצביע לעץ AVL לפי k_1. העץ, שהמפתח שלו k_2, יכיל מצביע למפתח המקסימלי בו.

חיפוש יעשה קודם כל לפי k_2 ואח"כ, בעץ המתאים, לפי k_1.

בהכנסה נחפש את המפתח k_2 . אם הוא קיים נכניס את האיבר החדש לעץ זה, אם הוא לא קיים ניצור עץ חדש למפתח k_2 ונכניס את האיבר החדש לעץ זה. נחפש את המקס' בעץ ונעדכן מצביע למקס'

במחיקה נחפש את k_2 ומהעץ שלו נמחק את k_1. אם העץ ריק, נמחק את k_2 מהעץ. נעדכן מצביע למקס'.

מציאת המפתח המקסימלי תעשה בעזרת המצביע למפתח זה.

עצי חיפוש בינאריים – פתרון שאלה 3

טענה 1

ניתן להפוך כל עץ בינרי לעץ שאין בו בנים שמאליים על-ידי רוטציות בלבד.

הסבר: מבצעים רוטציות ימינה על השורש עד שאין לו בן שמאלי. יורדים אל בנו הימני ומבצעים עליו רוטציות ימינה עד שאין לו בן שמאלי. יורדים אל בנו הימני...

טענה 2

אם ניתן להפוך עץ בינרי A לעץ בינרי B בעזרת רוטציות בלבד, אז אפשר להפוך עץ בינרי B לעץ בינרי A בעזרת רוטציות בלבד.

הסבר: מבצעים רוטציות הפוכות בסדר הפוך.

תזכורת:

רוטציה לא משנה את הסדר התוכי של העץ.

טענה 3

לשני עצים אותם ערכים בסריקה בסדר תוכי אם ורק ניתן לעבור מאחד לשני בעזרת רוטציות בלבד.

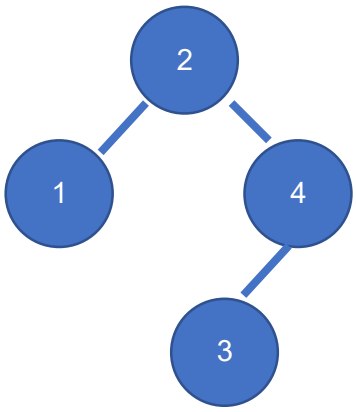
הוכחה: ברור שאם ניתן לעבור מעץ אחד לשני אז הסריקות בסדר תוכי תהיינה זהות (לאור התזכורת). בכיוון השני: אם לשני העצים אותה סריקה בסדר תוכי, אז ניתן להעביר את הראשון לצורה המתוארת בטענה 1; מצורה זו ניתן לעבור לעץ השני בעזרת רוטציות בלבד לפי טענות 1ו-2; בסה"כ הפכנו את העץ הראשון לעץ השני בעזרת רוטציות בלבד.

האלגוריתם המבוקש צריך לבדוק שסריקה תוכית של שני העצים מניבה אותה רשימת ערכים בדיוק.

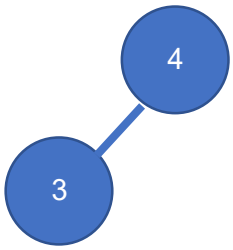
עצי חיפוש בינאריים – פתרון שאלה 4

נפריך בעזרת דוגמה נגדית.

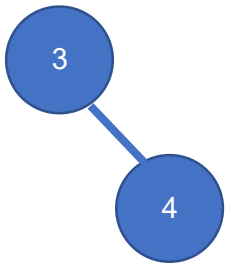
בעץ:



אם נמחק את 1 ואחריו את 2 נקבל:



אם נמחק את 2 ואחריו 1 נקבל:



עצי חיפוש בינאריים – פתרון שאלה 5

אפשרויות 3, 5:

1. 2, 252, 401, 398, 330, 344, 397, 363
2. 924, 220, 911, 244, 898, 258, 362, 363
3. **925, 202, 911, 240, 912, 245, 363**
4. 2, 399, 387, 219, 266, 382, 381, 278, 363
5. **935, 278, 347, 621, 299, 392, 358, 363**

עצי חיפוש בינאריים – פתרון שאלה 6

הצמת המבוקש חייב להימצא על מסלול החיפוש של x בעץ: אם x נמצא בעץ אז הצמת המכיל אותו הוא הצמת המבוקש; אם x לא בעץ, אז הצמת המבוקש הוא הקודם בסדר תוכי או העוקב בסדר תוכי של x (אם x היה בעץ).

```
findClosest(t,x)
c←t
while t≠null do
  if key[t]=x
  then return t
  if |key[c]-x|>|key[t]-x|
  then c←t
  if key[t]<x
  then t←right[t]
  else t←left[t]
return c
```

עצי חיפוש בינאריים – פתרון שאלה 7

נבצע חיפוש בעץ. כאשר ניתקל בערך x נשמור מצביע לצמת, ונמשיך לסרוק בתת-עץ השמאלי לחיפוש מופע "שמאלי" יותר של x.

```
searchFirst(t,x)
q←t
while t≠nil do
  if key(t)≤x
  then
    if key(t)=x
    then q←t
    t←left(t)
  else
    t←right(t)
return q
```

עצי חיפוש בינאריים – פתרון שאלה 8

המופע הראשון של x שאלגוריתם החיפוש בעץ חיפוש בינרי נתקל בו (נסמן אותו ב-f) הוא הצמת המבוקש.

<u>הוכחה</u>

X לא הופיע באבות הקדמונים של f.

אם f אינו המופע הראשון של x בסדר תוכי אז למופע הראשון של x ול-f יש אב קדמון משותף. ערכו של אב קדמון זה חייב להיות x, בניגוד לכך שאין ל-f אבות קדמונים שערכם x.

עצי חיפוש בינאריים – פתרון שאלה 9

המופע הראשון של x שאלגוריתם החיפוש בעץ חיפוש בינרי נתקל בו (נסמן אותו ב-f) הוא הצמת המבוקש.
ההוכחה זהה להוכחה בשאלה 8.

עצי חיפוש בינאריים – פתרון שאלה 10

נמצא את המופע הראשון, בסדר תוכי, של x, ובעזרת ספליט ניצור עץ שבו כל הערכים הקטנים מ-x.
נמצא את המופע האחרון, בסדר תוכי, של x, ובעזרת ספליט ניצור עץ שבו כל הערכים הגדולים מ-x.
נאחד את שני העצים הללו לקבלת העץ המבוקש.

עצי חיפוש בינאריים – פתרון שאלה 11

א. היתרון הוא שיש כאן החלפה בין צמתים ולא בין תכולה של צמתים. אם יש איברים נוספים שמצביעים לצמת מסוים, הם לא יצביעו אליו יותר אחרי המחיקה.
החיסרון הוא שהעץ עלול להפוך למאוד לא מאוזן. המחיקה המקורית משנה את גבהי התת-עצים ב-1 לכל היותר.

ב. אלגוריתם המחיקה ב-AVL מסתמך על כך שגובהי התת-עצים קטן ב-1 לכל היותר. לכן האלגוריתם החלופי לא יכול לשמש למימוש AVL.

עצי חיפוש בינאריים – פתרון שאלה 12

נשתמש בעץ ערכי מיקום המתואר בשבוע השישי של המוק.

הפעולה successor תבוצע על-ידי מציאת x, ערך המיקום של z, ומציאת האיבר שערך המיקום שלו $x+k$.

עצי חיפוש בינאריים – פתרון שאלה 13

א.

```
BUILD-TREE(A,p,r)
if p>r
then return nil
q←⌊(p+r)/2⌋
t←ALLOCATE-NODE(A[q])
p[t]←nil
left[t]←BUILD-TREE(A,p,q-1)
if left[t]≠nil
then p[left[t]]←t
right[t]←BUILD-TREE(A,q+1,r)
if right[t]≠nil
then p[right[t]]←t
return t
```

ב. נסרוק את העצים בסדר תוכי לתוך שני מערכים ממוינים. נמזג את שני המערכים למערך אחד ממוין. נבנה מהמערך עץ מאוזן.