מבני נתונים 07b

תרגול 4
21/3/2007

ליאור שפירא

תזכורת – B-trees

- לכל צומת x יש השדות הבאים
  - n[x] מס' מפתחות ב-x
  - המפתחות עצמם בסדר לא יורד
  - כל צומת פנימי מכיל גם n[x]+1 מצביעים לילדי הצומת
  - המפתחות בצומת מפרידים את ערכי ילדי הצומת
- כל העלים בעץ הינם באותו עומק
- יש חסם עליון ותחתון למס' מפתחות שצומת מכיל
- נבטא חסם זה ע"י t≥2, הדרגה המינימלית של העץ
  - לכל צומת חוץ מהשורש יש לפחות t-1 מפתחות
  - לפיכך, לכל צומת פנימית (חוץ מהשורש) יש לפחות t ילדים
  - בכל צומת יש לכל היותר 2t-1 מפתחות, ז"א מקסימום 2t ילדים
  - צומת תיקרא מלאה אם מכילה בדיוק 2t-1 מפתחות

הכנסה ל-B-tree

- אופציה א'
  - מצא את הצומת בו אמור להיכנס הערך
  - אם הצומת מלאה פצל אותה וחזור רקורסיבית
- אופציה ב'
  - תוך כדי חיפוש מקום להכניס את הערך החדש נפצל כל צומת מלאה שניתקל בה
    - יתרון: לא נצטרך לגשת פעמיים לצמתים (ייתכן שיושבות בזיכרון יקר לגישה)
    - חסרון: דורש שמס' בנים מקסימלי יהיה 2t (עצי 2-3 אינם B-Trees לפי הגדרה זו)

הכנסה ל-B-tree (אופציה ב')

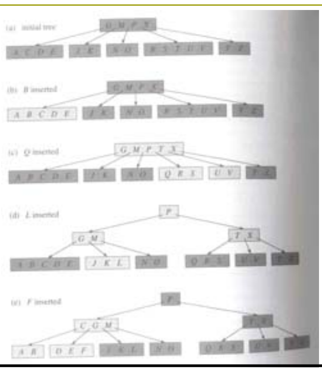
- B-Tree-Insert(T,k)
  - **r**←root[T]
  - If n[r]=2t-1
    - Create new root **s** and split **r** under it
    - B-Tree-Insert-Nonfull(s,k)
  - Else
    - B-Tree-Insert-Nonfull(r,k)

הכנסה ל-B-tree (אופציה ב')

- B-Tree-Insert-Nonfull(**x**,k)
  - i←n[**x**]
  - If leaf[**x**]
    - Find correct place and insert value k
  - Else
    - Find c_i[**x**] - child of **x** into which recursion continues
    - If n[c_i[**x**]]=2t-1
      - Split c_i[**x**]
    - B-Tree-Insert-Nonfull(c_i[**x**],k)

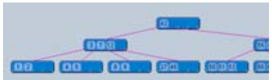
דוגמה – עץ בו t=3

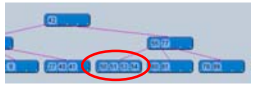
- t=3 ז"א זהו עץ 3-6



עצי 2-4 (b-trees)

- תרגיל: אם נתחיל מעץ 2-4 ריק ונבצע m פעולות insert, כאשר בכל פעולה נקבל את מיקום ההכנסה, אזי העלות הכללית של הסדרה היא O(m) זמן
- אינטואיציה
  - פעולות זולות – כשיש הרבה "חופש" בעץ
  - פעולות יקרות – כשאין "חופש" בעץ
  - מה מסמל "חופש" בעץ? צמתים בדרגה 2-3

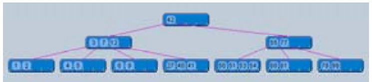




יקר

הוכחה

- נקרא לרמת ה"חופש" בעץ – slack
- נגדיר פונקצית פוטנציאל על פי slack
  - אם אין slack – הפוטנציאל גבוה
  - אם אין slack בצומת – יש 4 ילדים
- פונקצית הפוטנציאל: Φ(D)=#vertices of degree 4



Φ=1

הוכחה

- נצטרך להוכיח שני דברים
  1. פ' הפוטנציאל 0 בהתחלה ואי-שלילית בכל שלב
  2. לכל פעולה op מתקיים: cost(op)+ΔΦ≤O(1)
  - 1 ברור
  - עבור 2 נוכיח כי כל פעולה עולה O(1)+k כאשר k מס' פעולות ה-split שמתבצעות.

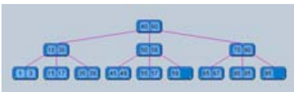
הוכחה

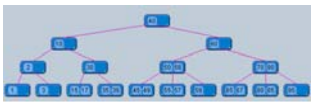
- כל split מפצל צומת 4 לצומת דרגה 2 וצומת דרגה 3
- לכן, כל פעולת split (כמעט) מורידה את הפוטנציאל ב-1
  - ה-split הראשון, על העלה, מפצל צומת 0 לשני צמתים מדרגה 0
  - ה-split האחרון מפצל צומת v מדרגה 4 לשני צמתים (2 ו-3) אך אביו עלול להשתנות לדרגה 4 – אין שינוי פוטנציאל
- לכן:
  - ΔΦ≤-(k-2)
  - cost(op)+ΔΦ=O(1)

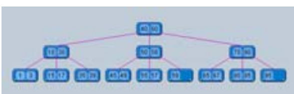
האם זה נכון לעצי 2-3?

- ניתן להראות כי בעצי 2-4 עבור כל סדרת פעולות insert ו-delete, עלות amortized נשארת O(1) לפעולה
- האם הדבר נכון לעצי 2-3?
  - הדבר לא נכון, בעץ 2-3 מלא, אם נבצע סדרת פעולות insert ו-delete של אותו אבר, כל הפעולות יהיו יקרות ולכן O(logn) לפעולה!

האם זה נכון לעצי 2-3?



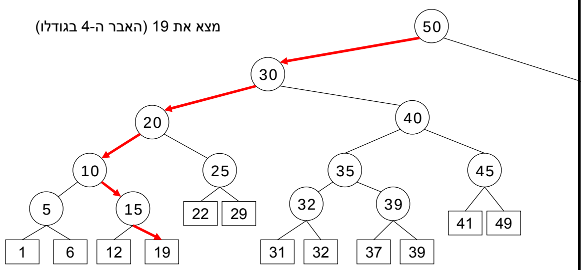




תרגיל

- הראה כי בשינויים קלים למבנה עץ בינארי (מאוזן), ניתן למצוא את האבר ה-k בגודלו, בזמן O(logk)

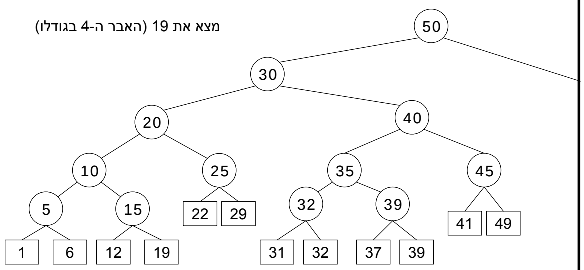
מצא את 19 (האבר ה-4 בגודלו)



תרגיל

- הראה כי בשינויים קלים למבנה עץ בינארי (מאוזן), ניתן למצוא את האבר ה-k בגודלו, בזמן O(logk)

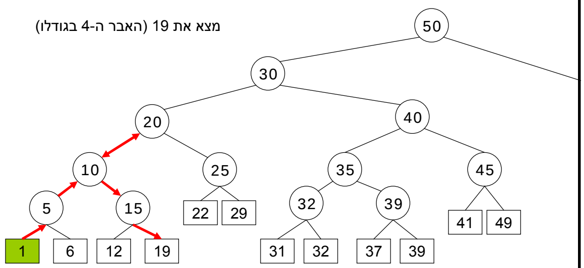
מצא את 19 (האבר ה-4 בגודלו)



תרגיל

- הראה כי בשינויים קלים למבנה עץ בינארי (מאוזן), ניתן למצוא את האבר ה-k בגודלו, בזמן O(logk)

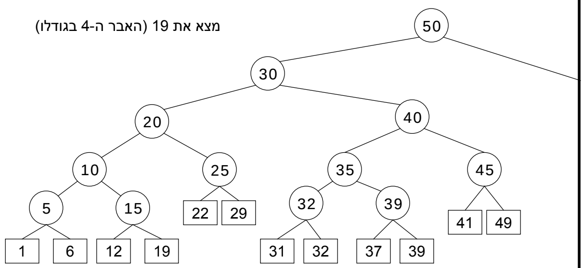
מצא את 19 (האבר ה-4 בגודלו)



תרגיל

- הראה כי בשינויים קלים למבנה עץ בינארי (מאוזן), ניתן למצוא את האבר ה-k בגודלו, בזמן O(logk)

מצא את 19 (האבר ה-4 בגודלו)



תרגיל 4 (שיעורי בית)

Suppose that we start from an empty Red-Black tree, and that we perform *n* insert operations, where *n* > 1. Prove that we necessarily end up with a Red-Black tree which has at least one red vertex.

Assume that we are dealing with a Red-Black tree that stores its elements at the leaves.

(Hint: One way to prove this is: (1) prove that the second insert operation creates a red node; (2) then prove that for any insert operation, if there is a red node before applying it, then there is also a red node after applying it. Note that there may be other ways to prove the claim.)

תרגיל

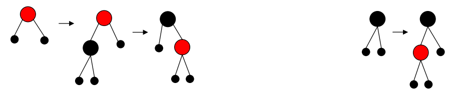
- הראה כי בשינויים קלים למבנה עץ בינארי (מאוזן), ניתן למצוא את האבר ה-k בגודלו, בזמן O(logk)
- פתרון: נניח עץ AVL בעל n צמתים

- p← מצביע לעלה השמאלי ביותר (קטן)
- While (p!=root) do
  - p←p.parent
  - If (p.key>x) then
    - x is found in left sub-tree so search there
- End while
- We got to the root so perform regular find

Max. O(logk) iterations

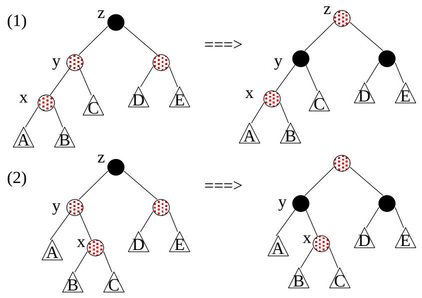
O(k) elements in sub-tree → O(logk) search

פתרון



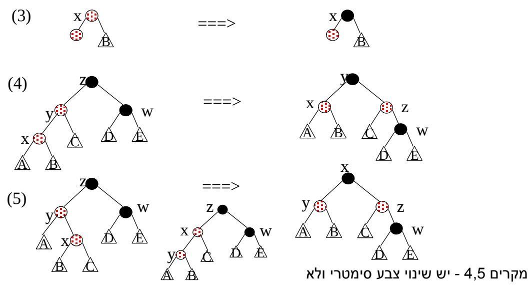
- לאחר 2 הכנסות נישאר עם שורש שחור וצומת אדום
- באופן כללי, בהכנסת צומת חדש אנו
  - הופכים עלה לצומת פנימי אדום עם 2 בנים שחורים
  - אם אין הפרה של החוקים, אזי בוודאות יש צומת אדום
  - אם יש הפרה, זו הפרה של החוק האדום ונפתור ע"י אחד מחמשת המקרים, נראה שבכל אחד מהם נשאר צומת אדום...

פתרון



מקרים 1,2 – מספר צמתים אדומים קטן ב-1, אך בשביל מקרה זה חייבים להיות לפחות 3 צמתים אדומים ולכן יישארו לפחות 2 לאחר הפעולה

פתרון



מקרים 4,5 - יש שינוי צבע סימטרי ולא משתנה מס' הצמתים האדומים
מקרה 3 – מספר צמתים אדומים קטן ב-1 חייבים להיות לפחות 2 בתחילת הפעולה

שלום וביי