האוניברסיטה הפתוחה

מבני נתונים ומבוא לאלגוריתמים
מפגש הנחיה מס' 11

מדעי המחשב, קורס מס' 20407

סמסטר 2016ב

מנחה: ג'ון מרברג

מה ראינו במפגש הקודם?

- עצים אדומים-שחורים (המשך)
  - סיבוב שמאלי וימני (פעולות עזר)
  - הכנסה, מחיקה

- הרחבה של מבני נתונים
  - הרחבה – מוטיבציה
  - עץ ערכי מיקום

מפגש אחד-עשר

- נושא השיעור
  - פרק 14 בספר – הרחבה של מבני נתונים (המשך)
    - הרחבה – הכללה של תחזוקת שדה מרחיב
    - דוגמאות נוספות להרחבה של עץ אדום שחור
  - חזרה (חלק ראשון)
    - הגדרת מבנה נתונים מופשט (ADT)
      - הפוטנציאל של כל אחד ממבני הנתונים הבסיסיים
      - שילוב מספר מבני נתונים בסיסיים לצורך הגדרת ADT

מבוסס על מצגת של ברוך חייקין ואיציק בייז

הרחבת מבני נתונים

- לעתים קרובות ניתן "להרחיב" מבנה נתונים קיים כך שיתמוך גם בפעולות נוספות
  - לצורך כך אפשר לשמור מידע נוסף במבנה הנתונים

- ארבעת השלבים לביצוע ההרחבה:

א. בחר מבנה נתונים בסיסי בעל תכונות רצויות
  - למשל עץ אדום שחור או ערימה

ב. קבע את המידע הנוסף שיש לאחסן במבנה הנתונים הבסיסי
  - למשל שדה נוסף בכל צומת במבנה

ג. ודא שאפשר לתחזק ביעילות את המידע הנוסף במהלך ביצוע הפעולות הרגילות שמשנות את מבנה הנתונים הבסיסי
  - למשל בעת הכנסה והוצאה מהמבנה

ד. ממש את הפעולות הנוספות הנדרשות, והוכח את סיבוכיותן
  - לפעמים מוכתבת מלכתחילה סיבוכיות נדרשת, ויש לעמוד בה

דוגמה – ערכי מיקום דינמיים

- הבעיה:
  - בהינתן קבוצה דינמית *S* בעלת *n* מפתחות, רוצים לבצע ביעילות שאילתות של חיפוש המפתח במיקום ה-*i*, *ושל דירוג מפתח נתון בקבוצה S*.
  - תזכורת: ערך המיקום ה-*i* הוא המפתח ה-*i* בגודלו בקבוצה
- נשתמש בהרחבה: עץ ערכי-מיקום (Order-Statistic Tree)
  - שלב א': נבחר כבסיס עץ אדום-שחור *T*
  - שלב ב': נוסיף לכל צומת x בעץ את השדה *size*[*x*], אשר מוגדר כמספר הצמתים בתת-העץ המושרש בצומת *x* (נגדיר $size[nil[T]] = 0$)
    - בכל צומת *x* בעץ מתקיים: $size[x] = size[left[x]] + size[right[x]] + 1$
  - שלב ג': נראה כיצד ניתן לשנות את פעולות ההכנסה והמחיקה כך שהשדה *size* של כל צומת יישאר מעודכן
  - שלב ד': נוסיף את הפעולות OS-Select ו-OS-Rank
- הרעיון המרכזי של השימוש בשדה ההרחבה size:
  - יהי *x* שורש של תת-עץ כלשהו ב- *T*; אזי דירוגו *של המפתח של x* בקרב קבוצת המפתחות המאוחסנים בתת-העץ הוא: $size[left[x]]+1$

הכללה: תחזוקת שדה מרחיב בהכנסה והוצאה מעץ א"ש

- ניתן לתחזק ביעילות, בלי תוספת לזמן הריצה האסימפטוטי, כל שדה מרחיב בעץ אדום שחור שמקיים תנאי "חישוב מקומי", כדלקמן
- משפט 14.1:
  - יהי f שדה המרחיב עץ אדום-שחור בן n צמתים.
  - אם ניתן לחשב את ערכו של f עבור כל צומת x עפ"י המידע המצוי <u>ב-x ובשני בניו בלבד</u> (כולל ערכי f של הבנים),

  אזי ניתן לעדכן את ערכי f בכל צמתי העץ במהלך פעולות ההכנסה וההוצאה, בלי להגדיל את זמן הריצה האסימפטוטי $O(\log n)$ של פעולות אלה.
- רעיון ההוכחה:
  - לפי תנאי המשפט, שינוי בשדה f בצומת כלשהו יכול להשפיע רק על האבות הקדמונים של הצומת שהשתנה
  - לפיכך, השינוי בשדה f נדרש לאורך מסלול מהצומת הנכנס/יוצא עד לשורש, כלומר לכל היותר עבור $O(\log n)$ צמתים בכל פעולת הכנסה/הוצאה
  - כל שינוי בשדה f בצומת נתון מתבצע בזמן $O(1)$ (לרבות סיבוב)

תרגיל: סכום המפתחות

הציעו מבנה נתונים S, שבעזרתו ניתן לבצע את הפעולות הבאות בזמנים הנדרשים (n מציין את מספר האיברים ב-S):

FIND (S,k): חיפוש אחר המפתח k במבנה S; זמן: $O(\lg n)$;

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

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

SUM_SMALL (S,k_1): חישוב והחזרת סכום המפתחות k ב- S המקיימים את התנאי: $k < k_1$;

זמן: $O(\lg n)$.

NUMKEYS (S,k): חישוב והחזרת מספר האיברים ב- S שהמפתח שלהם הוא k.

זמן: $O(\lg n)$.

מותר להניח כי $k \neq 0$.

תחזוקת השדה *sum* בהכנסה והוצאה מעץ א"ש

- תחזוקת השדה *sum* אינה משנה את הסיבוכיות, לפי משפט 14.1
  - הנוסחה לחישוב הערך *sum* בצומת *x* בעץ *T* היא:

$$sum[x] = sum[left[x]] + sum[right[x]] + key[x]$$

$$sum[nil[T]] = key[nil[T]] = 0$$

- בשלב הראשון של הכנסת מפתח *k* לעץ, נוסיף את הערך *k* לשדה *sum* בכל צומת במסלול מהשורש לנקודת החיבור
- בשלב הראשון של הוצאת צומת *z* מהעץ, יהי *y* הצומת שיוצא בפועל
  - נפחית את הערך *key*[*y*] מהשדה *sum* בכל צומת במסלול מ-*y* לשורש
  - אם $y \neq z$, נוסיף את הערך *key*[*y*]-*key*[*z*] לשדה *sum* בכל צומת במסלול מ-*z* לשורש
- בשלב התיקונים של הכנסה או הוצאה, בכל פעולת סיבוב נשתמש בנוסחה לעיל לתיקון השדה *sum*,
  - בדומה לתיקון השדה *size* בעץ ערכי מיקום

מימוש הפעולות

```
SUM_SMALL(S, k1)
x ← root[S]
count ← 0
while x ≠ nil[S] do
    if key[x] < k1
        then count ← count + key[x] + sum[left[x]]
             x ← right[x]
        else x ← left[x]
return count
```

- זמן ריצה: $O(\log n)$
- באופן סימטרי, ניתן להגדיר ולממש את הפעולה SUM_LARGE(S, k_1)

מימוש הפעולות (המשך)

NUMKEYS(S, k)

1. $sum_1 \leftarrow$ SUM_LARGE(S, k)
2. $sum_2 \leftarrow$ SUM_SMALL(S, k)
3. **return** $(sum[root[S]] - sum_1 - sum_2)/k$

- זמן ריצה: $O(\log n)$

תרגיל: שילוב מבני נתונים שונים

הציעו מבנה נתונים S, שבעזרתו ניתן לבצע את הפעולות הבאות בזמנים הנדרשים (n מציין את מספר האיברים ב- S):

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

POP (S) : מחיקה מהמבנה S של האיבר שנכנס האחרון ל- S ; זמן : $O(\lg n)$;

MINIMUM (S) : החזרת המפתח המינימלי במבנה S ; זמן : $O(1)$;

EXTRACT-MIN (S) : מחיקה מהמבנה S של האיבר בעל המפתח המינימלי ב-S ; זמן : $O(\lg n)$.

תרגיל: שכיחות של מפתחות

הציעו מבנה נתונים S התומך בפעולות הבאות (N מציין את מספר האיברים ב- S; n מציין את מספר המפתחות השונים זה מזה):

$\text{SEARCH}\ (S,k)$: חיפוש אחר המפתח k במבנה S;

$\text{INSERT}\ (S,k)$: הכנסת איבר חדש בעל המפתח k למבנה S;

$\text{DELETE}\ (S,k)$: מחיקת איבר כלשהו בעל המפתח k מהמבנה S;

$\text{FREQUENCY}\ (S,k)$: החזרת מספר האיברים בעלי המפתח k שבמבנה S;

$\text{SELECT}\ (S,i)$: החזרת ערך המיקום ה- i של המבנה S (האיבר ה- i הקטן ביותר בין כל N האיברים של S).

זמן הריצה הנדרש של כל אחת מהפעולות הינו $\Theta(\lg n)$.