מבני נתונים 07b

תרגול 6
6/6/2007

מיון

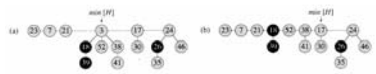
ליאור שפירא

שאלה 6 מתרגיל 5

- בהינתן מימוש של Fibonacci heaps בו לא מתבצע cascading cuts, הראו שעבור סדרת m פעולות על מקס' n אברים, עלות פעולה ממוצעת גבוהה ככל האפשר
- קצת תזכורת
  - פעולות decrease-key מאד מהירות (זמן קבוע)
  - בעת פעולת extract-min נבצע consolidate
  - דרגת כל צומת הינה D(n) והיא חסומה ע"י O(logn)

שאלה 6 מתרגיל 5

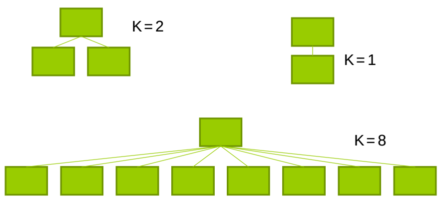
- בהינתן שאין cascading cuts, מה משתנה ב"מה שמובטח לנו"? ז.א איפה השתמשנו בידע שלא ניתן להוריד 2 בנים מצומת בלי שהוא ייחתך?
  - בהוכחה שהדרגה המקסימלית O(logn) (חסם על D(n))
- איפה משתמשים בעובדה זו?
  - ב-extract min נעבור על כל השורשים (מקס' D(n))



  - ז"א אם נגיע למצב שבו יש הרבה מאד עצים בערמה, כל פעולה תהיה יקרה
  - מעבר לכך, אם כל עץ מדרגה שונה, פעולת ה-consolidate לא תחבר עצים ביחד

שאלה 6 מתרגיל 5

- כיצד נגדיר עץ מדרגה k שהוא "ממש גרוע"?
  - נגדיר עץ מיוחד מדרגה k k-rank star. זהו שורש עם k בנים, כולם עלים.



שאלה 6 מתרגיל 5

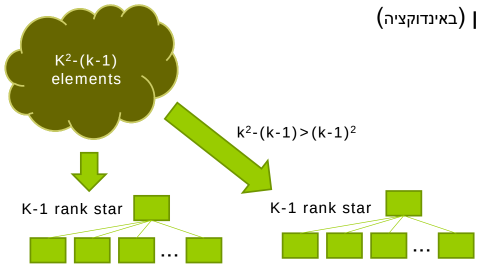
- הפתרון יורכב משני שלבים
  1. בסדרה של f(n) שלבים, בנו ערמת פיבונצ'י כך שיש
     - 1 rank-0 star
     - 1 rank-1 star
     - ...
     - 1 rank-sqrt(n) star
  2. חזרו על הפעולות הבאות מספר רב של פעמים (O(m))
     - הכנס ערך X ממש קטן
     - בצע extract-min (מוחק את X)
     - **כל אחת מהפעולות לוקחת Ω(√n) זמן, כך שעבור n >> m פעולה תיקח בממוצע √n**

שאלה 6 מתרגיל 5

- אך כיצד בונים k-rank star? כיצד נוודא שבשום שלב לא יהיו יותר מ-n אברים בערמה?
- פתרון
  - נראה שבהינתן k^2 אלמנטים, נוכל לבנות כוכב מדרגה k.
  - סדרת פעולות זו תיקרא A(k) ונבנה אותה רקורסיבית
  - נניח שביצענו כבר את A(k-1), כיצד נבנה את A(k)?
  - לאחר בניית k-1 star יש לנו k^2-(k-1) אברים
  - יש עדיין יותר מ-$(k-1)^2$ אברים ולכן נוכל לבנות עוד k-1 star
  - נמזג את שני העצים ונשתמש ב-decrease כדי "לגזום" את העץ שנוצר ולהפוך אותו ל-k-rank star

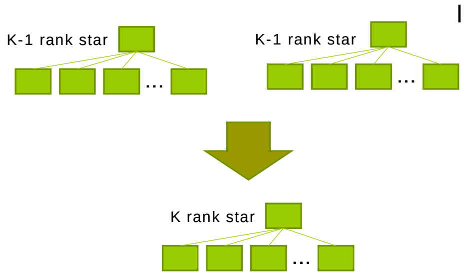
שאלה 6 מתרגול 5

- אך כיצד בונים k-rank star? כיצד נוודא שבשום שלב לא יהיו יותר מ-n אברים בערמה?
- פתרון (באינדוקציה)



שאלה 6 מתרגול 5

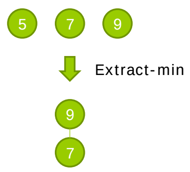
- אך כיצד בונים k-rank star? כיצד נוודא שבשום שלב לא יהיו יותר מ-n אברים בערמה?
- פתרון



תרגיל 2 – ערמות פיבונאצ'י

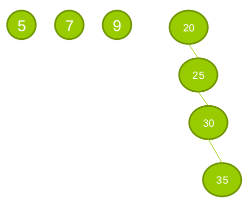
- האם ניתן לבנות ערמת פיבונאצ'י ובה עץ אחד מעומק n?
- פתרון
  - עבור n=1

  - עבור n=2



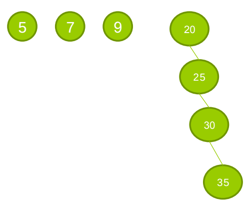
תרגיל 2 – ערמות פיבונאצ'י

- האם ניתן לבנות ערמת פיבונאצ'י ובה עץ אחד מעומק n?
- פתרון
  - נניח שאנו יודעים לפתור עבור n=k
  - נפתור עבור n=k+1
    - נגדיר x',y',z' כך ש- $key[z'] < key[y'] < key[x'] < \min[H]$
    - נכניס אותם לתוך הערמה



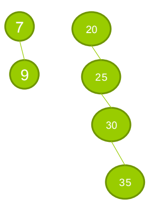
תרגיל 2 – ערמות פיבונאצ'י

- האם ניתן לבנות ערמת פיבונאצ'י ובה עץ אחד מעומק n?
- פתרון
  - נניח שאנו יודעים לפתור עבור n=k
  - נפתור עבור n=k+1
    - נגדיר x',y',z' כך ש- $key[z'] < key[y'] < key[x'] < \min[H]$
    - נכניס אותם לתוך הערמה
    - נפעיל extract-min



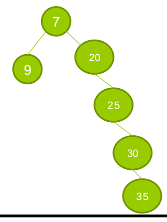
תרגיל 2 – ערמות פיבונאצ'י

- האם ניתן לבנות ערמת פיבונאצ'י ובה עץ אחד מעומק n?
- פתרון
  - נניח שאנו יודעים לפתור עבור n=k
  - נפתור עבור n=k+1
    - נגדיר x',y',z' כך ש- $key[z'] < key[y'] < key[x'] < \min[H]$
    - נכניס אותם לתוך הערמה
    - נפעיל extract-min



תרגיל 2 – ערמות פיבונאצ'י

- האם ניתן לבנות ערמת פיבונאצ'י ובה עץ אחד מעומק n?
- פתרון
  - נניח שאנו יודעים לפתור עבור n=k
  - נפתור עבור n=k+1
    - נגדיר x',y',z' כך ש- $key[z'] < key[y'] < key[x'] < \min[H]$
    - נכניס אותם לתוך הערמה
    - נפעיל extract-min
    - נמחק את x' (9)



תרגיל 3

- בהינתן n מספרים שלמים בטווח $1..n^{\lg\lg n}$ כיצד ניתן למיין בזמן O(nlglgn)
- פתרון
  - נתחיל ממספרים בטווח $1..n^2$ האם ניתן למיין אותם בO(n)?
  - אילו אלגוריתמים אנו מכירים שממיינים בזמן ליניארי?
    - Counting sort – בהנתן n מספרים בטווח 0..k $\theta(k+n)$
    - $k=O(n) \to \theta(n)$
  - נכתוב כל מספר בבסיס n, כמה ספרות?
  - נמיין כמו radix sort, O(n) לכל ספרה ולכן 2O(n)=O(n)
  - עבור הבעיה המקורית כמה ספרות יהיו לנו?

חסם תחתון על מיון על ידי השוואות

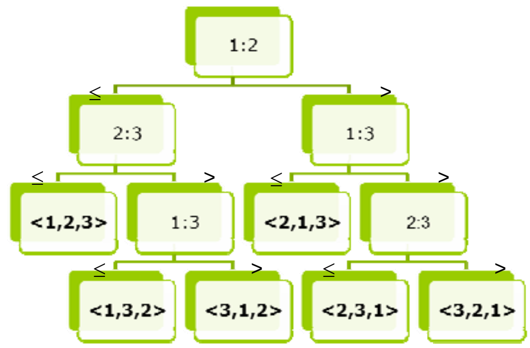
- אם אלגוריתם המיון שלנו מתבצע רק ע"י השוואות, כמה סידורים אפשריים יש ל-n אברים?
  - יש n! סידורים אפשריים
- For <a,b,c>
  - a,b,c
  - a,c,b
  - b,a,c
  - b,c,a
  - c,a,b
  - c,b,a

n!

חסם תחתון על מיון על ידי השוואות

- ניתן לייצג כל אלגוריתם מיון (השוואתי) על ידי עץ החלטות בינארי
  - למה ניתן לייצג כעץ?
  - למה דווקא כעץ בינארי?
- בעץ עצמו
  - כל צומת מייצגת את הסידור החלקי כפי שידוע לנו עד נקודה זו
  - קשתות העץ הן תוצאות ההשוואה

חסם תחתון על מיון על ידי השוואות



חסם תחתון על מיון על ידי השוואות

- מס' ההשוואות המקסימאלי = עומק העלה העמוק ביותר בעץ
  - מס' ההשוואות הממוצע = עומק עלה ממוצע
- עץ החלטה למיין n אברים הוא בעל n! עלים בהכרח
  - עץ בינארי מעומק d הוא בעל 2^d עלים לכל היותר
    - עץ בינארי בעל 2^d עלים חייב להיות בעל עומק d
    - עץ בעל n! עלים חייב להיות מעומק של לפחות |log(n!)|
  - ולכן כל אלגוריתם מיון מבוסס השוואה דורש לפחות |log(n!)| השוואות במקרה הגרוע ביותר.

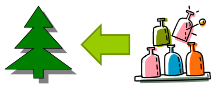
חסם תחתון על מיון על ידי השוואות

$$\log_2 n! = \log(n(n-1)(n-2)...(2)(1))$$
$$= \log n + \log(n-1) + ... + \log 2 + \log 1$$
$$\geq \log n + \log(n-1) + ... + \log(n/2)$$
$$\geq n/2 \log n/2$$
$$= n/2 \log(n - n/2)$$
$$= \Omega(n \log n)$$

חסם תחתון על חיפוש

- מה החסם התחתון על חיפוש במערך ממוין?
  - במקרה הגרוע ביותר נשווה לכל איבר
  - ז"א בעץ יש n עלים
  - מה העומק המינימלי של עץ בינארי בעל n עלים?
    - זמן לוגריתמי O(logn)

תרגיל 2



- כמה "יקר" להפוך ערמה לעץ חיפוש בינארי?
- נראה שהחסם התחתון הוא nlogn
- נפתור ברדוקציה על דרך השלילה
  - נניח שניתן להפוך ערמה לעץ חיפוש בינארי ב-f(n)
  - בהינתן n מספרים, מיינו את המספרים (במודל ההשוואה)
    - נבנה ערמה בזמן ליניארי (O(n))
    - נמיר את הערמה לעץ בינארי מאוזן בזמן f(n)
    - נעבור על העלים ב-in order ונדפיס אותם (ממוינים) בזמן ליניארי
    - **סה"כ זמן – O(n) + f(n) + O(n)**
    - לכן, אם f(n)<<log(n) ניתן למיין n מספרים בזמן קטן מ-O(nlogn) ואנו יודעים שזה בלתי אפשרי. מ.ש.ל.

תרגיל 3

1. כמה השוואות נדרשות למצוא אבר MIN/MAX במערך?

1	2	3	4	5	6	7	8

1

מינימום

תרגיל 3

1. כמה השוואות נדרשות למצוא אבר MIN/MAX במערך? n-1
2. כמה השוואות נדרשות למצוא את ה-MIN וה-MAX במערך בבת אחת?

1	2	3	4	5	6	7	8

x	y
מקסימום	מינימום

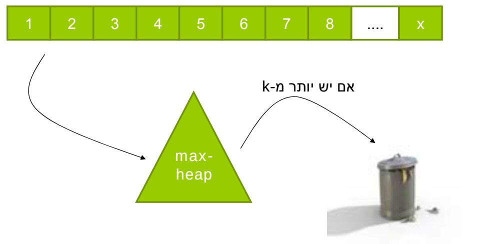
תרגיל 3

1. כמה השוואות נדרשות למצוא אבר MIN/MAX במערך? n-1
2. כמה השוואות נדרשות למצוא את ה-MIN וה-MAX במערך בבת אחת? 3n/2
3. חשב k האברים הקטנים (גדולים) ביותר במערך בגודל n בזמן O(nlogk), מצא אותם לפי הסדר

1	2	3	4	5	6	7	8	....	x

תרגיל 3

3. חשב k האברים הקטנים (גדולים) ביותר במערך בגודל n בזמן O(nlogk)



תוצאה סופית: כל מה שנשאר ב-heap

הסוף

שבוע נעים