

# מבני נתונים ואלגוריתמים - הרצאה 17

27 בדצמבר 2011

## ערימת פיבונאצ'י - Fibonacci Heap

מתבססת על סדרת פיבונאצ'י  $1, 1, 2, 3, 5, 8, \dots$ .  
הכנסת נתון בעלות  $O(1)$ .  
שאלה לגבי המינימום בעלות  $O(1)$ .  
הוצאת מינימום בעלות  $O(\log(n))$  (באופן טיפוס).  
יש לנו רשימה מקושרת דו כיוונית ציקלית (מעגלית): האחרון מחובר לראשון).  
כל מרכיב ברשימה הוא ראש של ערימה - בכל מרכיב ברשימה הבן חייב להיות גדול מאביו (בערימת מינימום).  
בנוסף, יש לנו מצביע למרכיב המינימלי ברשימה.

### הערה טכנית

- בתוך כל תת ערימה, יותר נוח להחזיק מצביע לבכור וממנו מצביעים דו כיווניים לכל אחיו.
- כל אב מחזיק רישום של מס' ילדיו.

### פעולות בערימת פיבונאצ'י

- בדיקת מינימום - המצביע למינימום של הרשימה הוא המינימום של הערימה בכללי.
- הכנסה: הכנס בתור שכן של המינימום, ואם קטן מהמינימום, הצבע לערימה דרכו.
- הוצאה:

- שלוף ערך בראש הערימה (המינימום).
- החלף את הערך בראש הערימה ברשימת כל הבנים שלו.
- מחק ערך בראש הערימה.
- תקן ערימה.

תיקון הערימה נעשה כך:  
תחילה יש כלל נוסף לערימת פיבונאצ'י: לא מחזיקים בראש הערימה שתי תתי ערימות עם אותו מס' ילדים.  
לכן, אם קיימים שני ראשי ערימות עם אותו מס' ילדים, הופכים את הגדול לבן של הקטן.  
אח"כ עוברים על הרשימה הראשית ומעבירים את המצביע לערימה למינימום.  
הטענה (לא נוכח) היא שאם מס' הראשים ברשימת הערימות הוא  $N$ , מס' האיברים בערימת הפיבונאצ'י הוא:

$$\left[ \frac{1 + \sqrt{5}}{2} \right]^N$$

וזה בקירוב טוב המס' ה- $N$  בסדרת פיבונאצ'י.

## טבלת גיבוב - Hash Table

אנו רוצים לבנות מבנה נתונים שבהינתן מחרוזת כלשהי (למשל שם) יתן לנו מידע מסוים (למשל מידע אישי).  
הדרך הנוחה להעביר היא דרך אינדקס כלשהו לכל שם, למשל ת"ז, ואז להחזיק במערך את המידע לפי אינדקס.

אך אנו צריכים דרך להעביר מהמחרוזת לאינדקס. לשם כך אנו צריכים פונק' כלשהי שנקראת פונק' גיבוב,  $f$ , שמעבירה ממחרוזת לאינדקס.  $f$  צריכה לקיים שאם  $f(a) \approx f(b)$  אז  $f(a) \neq f(b)$ .  
 דרך פשוטה לעשות זאת זה גימטריה, אך זה מוביל לבעיות כיוון של "משה דוד" יהיה אותו האינדקס כמו "לדוד משה", ואנו רוצים פיזור אחיד.  
 הפונק' שבה משתמשים לטבלאות גיבוב היא:

$$f(A) = \left( \sum_i A_i \cdot k^i \right) \% B$$

כאשר  $A_i$  הערך הגימטרי של האות  $i$  במילה  $A$ , ו  $k$  ו  $B$  מס' ראשוניים גדולים. המערך שלנו נקרא טבלת גיבוב והפונק' נקראת פונקציית גיבוב.

## התנגשויות

יכול לקרות מצב (נדיר) שבו עבור  $x_1 \neq x_2$  יתקיים  $f(x_1) = f(x_2)$ . יש שתי אפשרויות:

- Closed Addressing - עבור כל התנגשות, אנו מחזיקים במערך רשימה מקושרת של השמות שמוביל-ים לאינדקס הזה.

$$\alpha = \frac{\text{Number of items in the database}}{\text{Array size}}$$

$\alpha$  היא תוחלת מס' ההתנגשויות, לכן גשיה למידע עבור שם מסויים היא  $1 + \alpha$ .

- Open Addressing - אם מתנגשים, הולכים למקום אחר במערך. אם גם שם כבר תפוס, אז הולכים לעוד מקום וכו'.

העלות של הגישה לשם מסוים היא  $1 + \alpha + \alpha^2 + \dots = \frac{1}{1-\alpha}$  וכאשר  $\alpha$  קטנה זה בקירוב טוב  $1 + \alpha$ .  
 ההבדל פה יכול להיות איך עוברים למקום אחר - פונקציית Rehash. פונק' Rehash הבסיסית ביותר היא  $f(x) = x + i$ , כלומר לעבור  $i$  מקומות קדימה במערך, אך יכול לקרות מצב שבו ייקח הרבה זמן להגיע למקום פנוי.  
 פונק' Rehash טובה שמשתמשים בה היא Cuckoo Hashing - גיבוב קוקיה.  
 לכל מחרוזת  $x$  יש שתי פונק' -  $f_1(x)$  ו  $f_2(x)$ . הפונק' האלה הן פונק' גיבוב שונות שמקיימות:

$$f_1(x) \neq f_2(x)$$

ואם  $f_1(x) = f_1(y)$  ו  $f_2(x) = f_2(y)$  אזי  $x = y$ .  
 תחילה מנסים להכניס את  $x$  ל  $f_1(x)$ . אם המקום תפוס, מנסים להכניס ל  $f_2(x)$ . אם גם הוא תפוס, מוציאים את מה שיש ב  $f_2(x)$  ושמים בו את  $x$ , ומחפשים למה שהיה ב  $f_2(x)$  מקום אחר (באותה שיטה).