

## מבני נתונים ואלגוריתמים - הרצאה 4

24 בנובמבר 2011

### נושאי ההרצאה היום

- תור - Queue
- תור קדימויות - Priority Queue
- ערימה - Heap
- עצים - Tree

### ייצוג תור ע"י רשימה מקושרת

גם פה נצטרך פוינטר של inp ופוינטר של out.

• IsEmpty:

1. אם  $out == null$ , כן, אחרת לא.

• New:

1.  $outp = null$

2.  $inp = null$

•  $x = A.Front$ :

1. בדוק אם התור ריק, אם לא:

2.  $x = out \rightarrow value$

•  $x = A.Dequeue$ :

1. האם התור ריק? אחרת:

2.  $x = out \rightarrow value$

3.  $out1 = out \rightarrow next$

4.  $free(out)$

5.  $out = out1$

•  $A.Enqueue(x)$ :

1.  $inp1 = malloc(queue)$

2.  $inp1 \rightarrow next = null$

3.  $inp1 \rightarrow value = x$

4. אם התור לא ריק:

(א)  $inp \rightarrow next = inp1$

5.  $inp = inp1$

6. אם התור ריק:  $out = inp1$

שימו לב שניתן לממש בעזרת פוינטר אחד בלבד, אם מחזיקים את out בתור  $inp \rightarrow next$  (רשימה מעגלית).

• A.EmptyQueue:

1. כל עוד התור לא ריק, הוצא איבר מהתור.

## תור קדימויות

יש שתי אפשרויות:

1. לכל ערך יש מספר קטן של קדימויות אפשריות. (למשל, לכל ערך יש קדימות מהקבוצה  $\{1, 2, 3, 4\}$ ).
2. הרבה מאוד קדימויות אפשריות. (למשל, קדימות של ערך יכולה להיות כל ערך ב- $\mathbb{R}$ ).

המקרים שונים בדרך הפתרון:

1. אם יש קצת קדימויות, מחזיקים מספר תורים, אחד לכל ערך קדימות אפשרי.
2. כאשר יש לנו הרבה קדימויות, מחזיקים תור בודד ובכל שלב מסדרים את התור. ניתן לסדר בהכנסה או בהוצאה, אך זה הורס את החוקים שקבענו - ההכנסה או ההוצאה תהיה  $O(n)$ . האם ניתן לסדר את הנתונים כך שבכל שלב - הכנסה או הוצאה, העלות של תור קדימויות תהיה  $\log(n)$ ?

## ערימה Heap-

אנו מסדרים איברים בעץ בינרי, כך שכל שורה מלאה מלבד אולי השורה האחרונה בה יש רק עלים, ושכל אבא גדול מהילדים שלו. (זו נקראת ערימת מקסימום, בערימת מינימום האב תמיד קטן מהילדים שלו).

אנו מייצגים את הערימה במערך, כאשר אנו רושמים את הערכים לפי סדר השורות, ומשמאל לימין - הראש יהיה האיבר הראשון, האיברים הבאים יהיו איברי השורה השניה, האיברים הבאים יהיו איברי השורה השלישית וכו'.

כך נקבל:

מיקום האב במערך	מיקום הילדים שלו במערך
0	1 2
1	3 4
2	5 6
3	7 8

כלומר הבנים של האיבר במקום  $k$  הם  $2k + 1$ ,  $2k + 2$  (הבן השמאלי ב- $2k + 1$ , הבן הימני ב- $2k + 2$ ). האבא של הבן במקום  $k$  הוא  $\frac{k-1}{2}$  (עם עיגול כלפי מטה). אנו צריכים לשמור בנוסף למערך את המקום האחרון שיש בו ערך במערך, ואחריו כל המקומות ריקים. נניח ועכשיו רוצים להוציא את הראש (למשל, מוציאים אותו מתור הקדימויות). כעת הערימה לא מסודרת.

נעביר את האיבר האחרון למקום הראשון, כעת הערימה במבנה הנכון אך לא לפי החוק שהאב תמיד גדול מהבנים. כעת, נחליף אותו בבן הגדול ביותר שלו. כעת ברמה העליונה הכל בסדר, וגם בחלק של העץ של הבן הקטן יותר.

כעת שוב, נסתכל על המקום שהחלפנו. אם הוא לא מסודר, כלומר יש לו בן שגדול ממנו, נחליף בינו לבין הבן הגדול יותר שלו, וכך הלאה, עד שהעץ מסודר. כך, ב- $\log_2 n$  נוכל להוציא איבר מהערימה.

• הוצאת איבר מהערימה:

1. הוצא את הראש ( $x(0)$ ,  $x$  הוא המערך)
2. הכנס את האיבר האחרון ל- $x(0)$ :  
 $x(0) = x(n)$  כאשר  $n$  המקום האחרון.  
 $n - 1$
3. התחל מראש הערימה. אם האבא קטן מאחד מילדיו, החליף בין האב לבן הגדול ביותר.
4. חזור על 3 עבור הבן שהוחלף.

• הכנסה לערימה:

1. הכנס לקצה הערימה
2. אם גדול מאביו, החליף ביניהם
3. חזור על 2 עבור האב

גם פה העלות היא  $O(\log n)$ . לכן, נשתמש בערימה כדי לייצג תור קדימויות - הקדימויות יסודרו בערימה.

## עצים

אפשר גם פה לסדר את העץ במערך כך שהבנים של מקום  $i$  הם  $2k + i$  ו  $2k + 2i$  והאבא של מקום  $i$  הוא  $\frac{k-1}{2}$ .  
דרך אחרת לסדר עצים היא ע"י כך שלכל איבר בעץ יש ערך ומצביעים לבנים שלו - Left, Right ולרוב גם לאב שלו - Father.  
אך כך אנו יכולים לסדר רק עץ בינארי. אם אנו רוצים עץ שהוא לא בהכרח בינארי, נשתמש במבנה אחר - האב מצביע על הבן הבכור שלו, והבן מצביע על האח שלו.  
כלומר לכל איבר יש את הערכים הבאים:

- value
- parent
- child
- sibling

החסרון פה הוא שאם לאב יש  $k$  ילדים, להגיע לבן האחרון ייקח  $k$  פעולות.

## עצי חיפוש

לכל קודקוד יש ערך.  
הערך של הבן השמאלי תמיד קטן (או שווה) מהאבא והערך של הבן הימני תמיד גדול (או שווה) מהאבא, ולא רק הבן אלא כל צאצאיו של הבן הימני גדולים מהאב וכל צאצאיו של הבן השמאלי קטנים מהאב.

• חיפוש בעץ חיפוש:

1. אם הגעת לערך - מצאת.
2. אם הגעת לnull, אז הערך שאתה מחפש לא קיים. בעץ.
3. אם הגעת למספר קטן מהערך שאתה מחפש, חפש בתת העץ הימני.
4. אם הגעת למספר גדול מהערך שאתה מחפש, חפש בתת העץ השמאלי.

• הכנסת ערך לעץ חיפוש:

1. חפש את הערך. אם מצאת, סיים.
2. אחרת, הכנס את הערך למקום שהגעת לnull. וסמן את שני בניו בתור null.

• הוצאת ערך מעץ חיפוש:

1. חפש את הערך. אם לא מצאת, אין מה להוציא.
2. אם מצאת:
  - (א) אם עלה, מחק וסמן null במקום שהצביע אליו.
  - (ב) אם יש בן יחיד, הצבע מהאב של הערך (מהסבא) לבן, ומחק את הערך.
  - (ג) אם יש 2 בנים, הולכים צעד אחד ימינה ואז כל הזמן שמאלה (הכי שמאלי בתת העץ הימני) ומחליפים אותו בערך, ואם יש לו ילד מימין, אנו יודעים לטפל בו לפי מקרה (b).