

**שם הקורס: מבני נתונים**

מס' קורס: 7016610-5  
מרצה: פרופ' שפירא דנה  
העורך: משה חנוקוגלו  
תאריך: תשע"ז סמ' ב'

- 2.....טבלת יעילויות למבני נתונים
- 3.....תאור מבני נתונים
- 5.....: AVL בעץ גלגולים
- 6.....: גלגולים בעץ אדום שחור
- 7.....אתרים מומלצים:

טבלת יעילויות למבני נתונים

מס'	סוג מבנה הנתונים	הכנסה	הוצאה	חיפוש	יתרונות וחסרונות של מבנה הנתונים
1	מערך	$^1O(1)$	$^1O(1)$	$^3O(\log(n))$	מאפשר גישה לאינדקסים
		$^2O(n)$	$^2O(n)$	$^4O(n)$	
2	מחסנית	$O(1)$	$O(1)$	$O(n)$	LIFO
3	תור	$O(1)$	$O(1)$	$O(n)$	FIFO
4	רשימה מקושרת	$O(n)$	$O(n)$	$O(n)$	שימושי לאיסוף נתונים רבים (אם שומרים את הזנב אז אפשר להכניס ולהוציא מהסוף ב $O(1)$ )
5	רשימה מקושרת ממוינת	$O(n)$	$O(n)$	$O(n)$	שימושי לאיסוף נתונים רבים שלא ידוע כמותם בתחילת העבודה. (אם שומרים את הזנב אז אפשר להכניס ולהוציא מהסוף ב $O(1)$ )
6	עץ בינארי	$^1O(\log(n))$	$^1O(\log(n))$	$O(n)$	
		$^2O(n)$	$^2O(n)$		
7	עץ חיפוש בינארי	$^5O(\log_2 n)$	$^5O(\log_2 n)$	$^5O(\log_2 n)$	לכל קדקוד מתקיים שעל הערכים שגדולים ממנו יהיו בתת העץ הימני, וכל הערכים הקטנים ממנו יהיו בתת העץ השמאלי.
8	עץ מאוזן	AVL	$O(\log(n))$	$O(\log(n))$	גובה העץ הוא: $\theta(\log(n))$
		אדום שחור	$O(\log(n))$	$O(\log(n))$	גובה העץ הוא: $\theta(\log(n))$
9	Hash table	$O(1)$	$O(1)$	$O(1)$	
10	עץ / Heap ערמה	$O(\log(n))$	מינימום (מקסימום) $O(\log(n))$	$O(n)$	גישה למינימום (מקסימום, תלוי בעץ) $O(1)$ .
		$O(n)$	כללי $O(n)$		
11	ערמה בינומית	$O(\log(n))$	מינימום (מקסימום) $O(\log(n))$	$O(\log(n))$	גישה למינימום (מקסימום, תלוי בערמה) $O(1)$ .
		$O(n)$	כללי $O(n)$		
12	עץ B+	$O(\log_t(n))$	$O(\log_t(n))$	$O(\log_t(n))$	כאשר t הוא מספר הבנים.
13	Disjoint-sets	$O(\alpha(n))$	$O(\alpha(n))$	$O(\alpha(n))$	אבא של העץ הנמוך יותר יהיה הבן של השורש הגבוה. $\alpha(n)$ – הפונקציה ההפוכה לפונקציית אקרמן.
14	Skip List	$O(\log(n))$	$O(\log(n))$	$O(\log(n))$	
15	עץ איחזור קומפקטי	$O(\text{key\_length})$	$O(\text{key\_length})$	$O(\text{key\_length})$	
16	עץ סופיות	תקרא בקישור זה את <a href="#">Functionality</a> .			

<sup>1</sup> ידוע מקום האיבר במבנה.

<sup>2</sup> לא ידוע מקום האיבר במבנה.

<sup>3</sup> ממוין.

<sup>4</sup> לא ממוין.

<sup>5</sup> במקרה של עץ מנוון (עץ שרוך) זה יעלה לנו  $O(n)$ .

תאור מבני נתונים

מס'	מבנה הנתונים	תיאור
1	מערך	רצף איברים כך שכל איבר מסומן עם אינדקס. מבנה זה יהיה נח אם ידועה כמות האיברים האוסף בתחילת התוכנית. וכן מבנה זה מאפשר גישה מהירה מאד לאיברים שלו.
2	מחסנית	מבנה זה משתמש בשיטת Last In First Out, מה שאומר שהאיבר הראשון שנכניס למחסנית יצא אחרון.
3	תור	מבנה זה משתמש בשיטת First In First Out, מה שאומר שהאיבר הראשון שנכניס לתור יצא ראשון.
4	רשימה מקושרת	לכל צומת יש קשר לאיבר אחריו (ישנם רשימות שיש קשר גם לאיבר הקודם) וגם את הערך של עצמו. היתרון של מבנה נתונים זה הוא שאין לנו מגבלה של כמות האיברים שאנחנו יכולים להכניס (להבדיל ממערך שאנחנו צריכים קבוע את הגודל שלו ואם נרצה להוסיף איבר מעבר לגודלו נדרש לבצע פעולות נוספות) אך החיסרון בו שאין לנו גישה ישירה לאיברים לפי האינדקס שלהם מה שאומר שאנחנו נצטרך לעבור על כל האיברים עד לאיבר שאנו רוצים כל פעם שנרצה לגשת אליו (כמובן אם אין לנו מצביע ישיר לאותו איבר). ברשימה אנחנו נחזיר תמיד את ראש הרשימה כדי שהוא יוביל אותנו לבא אחריו והוא לזה שאחריו עד שנגיע לסוף הרשימה. (ישנם ממוישים ששומרים גם את הזנב של הרשימה, זה עוזר כדי להכניס איברים לרשימה ב $O(1)$ )
5	רשימה מקושרת ממוינת	רשימה מקושרת שכל איבריה מחוברים בצורה ממוינת.
6	עץ בינארי	העץ מורכב מצמתים, לכל צומת יש את הערך שלו וכן לכל צומת או שאין "ילדים" כלל או שיש לו "ילד" אחד או שיש לו שני "ילדים". צומת שאין לו כלל "ילדים" יקרא עלה. הצומת הראשון בעץ יקרא שורש.
7	עץ חיפוש בינארי	לכל קדקוד מתקיים שכל הערכים שגדולים ממנו יהיו בתת העץ הימני, וכל הערכים הקטנים ממנו יהיו בתת העץ השמאלי. (את הערכים השווים לו לפעמיים שמים משמאלו ולפעמיים מימינו, תלוי במימוש).
8	AVL	כל צומת מקיים שהערך המוחלט של הפרש בין גובה תת העץ הימני לבין גובה תת העץ השמאלי הוא לכל היותר 1. כאשר על חסם הגובה, $O(\log(n))$ , אנחנו שומרים ע"י גלגולים. (הסבר בסוף בטבלה)
	עץ מאוזן (מבוסס על עץ חיפוש בינארי)	העץ חייב לקיים את כל 5 התנאים האלו: 1. לכל צומת יש צבע, שחור או אדום. 2. השורש הוא שחור. 3. לצומת אדום לא יכולים להיות בנים שהם גם אדומים. 4. עלי null בצבע שחור. 5. העץ מקיים גובה שחור – לכל מסלול מהשורש עד העלים (עלי ה null) יש את אותו מספר של קודקודיים שחורים. כאשר על חסם הגובה, $O(\log(n))$ , אנחנו שומרים ע"י גלגולים שישמרו גם על התנאים לעיל. (הסבר בסוף בטבלה)
9	Hash table	רצף של איברים שמהווים נציגים לקבוצות איברים גדולות יותר. את הפיזור של האיברים בקבוצות הגדולות לנציגים נעשה ע"י פונקציית hash. האחסון של האיברים אצל הנציגים יתבצע או ע"י רשימות מקושרות שכל נציג הוא הראש שלהן או ע"י הפעלת פונקציית hash כמה פעמיים עד שנמצא מקום פנוי (במקרה שלא משתמשים ברשימות מקושרות, כאשר נמחק איבר ממקומו נשים דגל שהיה בתא זה איבר). מבנה זה טוב בעיקר כאשר אנחנו רוצים לקחת קבוצה גדולה של איברים ולמיין אותה לכמה קבוצות קטנות, המבנה נותן לנו גישה מאד יעילה לאיברים שבו.
10	Heap / עץ ערמה	(העץ יכול להיות עץ מינימום או מקסימום נתייחס אליו כמקסימום ובצורה מקבילה יעשה בעץ מינימום). עץ שהשורש בו גדול מערך כל אחד מבניו. הוא עץ מלא עד לרמה אחת לפני האחרונה וברמה האחרונה כל האיברים נדחסים משמאל לימין (יכול להיות שגם הרמה האחרונה מלאה). העץ ממומש ע"י מערך (נשתמש בשיטה שהאינדקס מתחיל מ-0, אך יש שיטה שהמספור מתחיל מ-1). גישה במערך: <ul style="list-style-type: none"> <li>אבא של אינדקס <math>i</math>: <math>\lfloor (i - 1) / 2 \rfloor</math></li> <li>בן שמאלי של אינדקס <math>i</math>: <math>2(i+1)-1</math></li> <li>בן ימני של אינדקס <math>i</math>: <math>2(i+1)</math></li> </ul>
11	עץ בינומי	עץ המוגדר באופן רקורסיבי. עץ בינומי $B_0$ יש לו רק שורש.

<p>עץ בינומי בסדר <math>B_k</math> מורכב משני עצים בינומים בסדר <math>B_{k-1}</math> כך שהשורש של אחד מהם הוא הבן השמאלי של השורש השני.          בעץ בינומי יש <math>2^k</math> צמתים. וכן לעץ <math>B_k</math> יש גובה <math>k</math>. וישנם <math>\binom{k}{i}</math> בעומק <math>i</math> כאשר <math>0 \leq i \leq k</math>.</p>		
<p>שרשור של עצים בינומיים, שמקיימים תכונות של ערמה מינימום/מקסימום. כמו כן יש לה לכל היותר רק עץ בינומי אחד מכל דרגה. ושורשי כל העצים מחוברים יחד ברשימה מקושרת.          כמות הצמתים בערמה כולה שקולה להצגה הבינרית של דרגת כל אחד משורשי העצים. (לדוגמא: ערמה שיש לה עצים בגובה 0,2,3,5 אזי יש לה <math>2^0 + 2^2 + 2^3 + 2^5 = 45</math> צמתים).          הגובה של הערמה <math>\lceil \log_2 n \rceil</math>.          כאשר נרצה להכניס צומת חדש לעץ אנחנו נכניס אותו כעץ בינומי בגובה 0 ואז נבצע מיזוגים עד לקבלת ערמה תקינה (לכל היותר עץ בינומי אחד מכל דרגה).</p>	<p><b>12</b></p> <p><b>ערמה בינומית</b></p>	
<p>עץ זה מקיים כמה תכונות (כאשר <math>n</math> הוא המספר המקסימלי של ילדים לצומת):</p> <ol style="list-style-type: none"> <li>1. כל העלים במרחק שווה מהשורש.</li> <li>2. כל צומת שהוא לא עלה או שורש מכיל בין <math>\lfloor \frac{n}{2} \rfloor</math> לבין <math>n</math> ילדים.</li> <li>3. בכל עלה יש בין <math>\lfloor \frac{n-1}{2} \rfloor</math> לבין <math>n-1</math> ערכים פנימיים.</li> </ol> <p>אם לשורש אין ילדים אז יכולים להיות לו בין 0 ל- <math>n-1</math> ערכים פנימיים, אם השורש לא עלה אז חייבים להיות לו לפחות 2 ילדים.          גובה העץ שיש בו <math>k</math> מפתחות הוא לכל היותר: <math>\lceil \log_{\lfloor \frac{n}{2} \rfloor} (k) \rceil</math>.          כאשר מכניסים ערך לצומת מלא אזי לוקחים את האיבר האמצעי בצומת ואם הצומת יְלָה אז מעתיקים אותו לאבא של אותו עלה ויוצרים שני צמתים חדשים שהם ילדים על האיבר האמצעי שעולה לאבא של הצומת המפוצל. ואם הצומת פנימי אז מעבירים את האיבר לאבא של אותו צומת ויוצרים שני צמתים חדשים שהם בילדים על האיבר האמצעי. וממשיכים רקורסיבית עד לשורש כל עוד לצומת נכנס איבר אחרי שהוא מלא.</p>	<p><b>13</b></p> <p><b>עץ B+</b></p>	
<p>מבנה נתונים זה מתאר חלוקה לקבוצות של קבוצת איברים.          מטרתו של מבנה זה היא לגשת לראש הקבוצה ב- <math>O(\alpha(n))</math> ואפשרות קלה למזג בין קבוצות.</p>	<p><b>14</b></p> <p><b>Union Find Disjoint-sets</b></p>	
<p>מבנה זה בנוי ממגדל של רשימות מקושרות כך שלכל קבוצה של לכל היותר שלושה ברשימה מסוימת יש נציג אחד ברשימה שמעליה.</p>	<p><b>15</b></p> <p><b>Skip list</b></p>	
<p>העץ מיצג קבוצה של רצפי מחרוזות מתוך מאגר סימנים קבוע מראש (לדוגמא: האלף בית, מספרים וכו'). כאשר רוצים להכניס מילה חדשה לעץ אנחנו נחפש את האות הראשונה באחד הבנים של השורש, אם מצאנו אז אנחנו נחפש את האות השנייה באחד הבנים של הצומת עם האות הראשונה וכן הלאה עד שלא נמצא את האות המבוקשת ברצף המילה ואז אנחנו ניצור משאר האותיות של המילה שרשרת צמתים שכל צומת יכיל את אותיות המילה שנשארו עד לסיום המילה. וכל השרשרת הזו תהיה צאצא של הצומת האחרון שמצאנו בעץ שמיצג אות מהמילה.          כל אבא שיש לו רק בן יחיד אז אנחנו נרשום את האבא והבן באותה הצומת עד שנגיע לצומת שיש לו יותר מבן אחד ושם נשאיר את הצמתים כמו שהם בלי דחיסה.          כל הצמתים שבאותה רמה מחוברים בניהם או שהם נמצאים כתאים במערך שגודלו כגודל האלף בית.</p>	<p><b>16</b></p> <p><b>עץ אחזור קומפקטי</b></p>	
<p>מכל מילה אנחנו יוצרים את כל הסייפות שלה (סיפא – זה תת מילה שמתחילה מאינדקס <math>i</math> ועד סוף המילה כאשר <math>0 \leq i \leq s.length</math>) ואז את כל אחת המסייפות אנחנו מכניסים לעץ כמו עץ אחזור קומפקטי. ובכל עלה או אפילו אם יש מילה מסתיימת בצומת שאינו עלה אנחנו נקשר בנוסף לשאר הבנים של עוד בן שהערך שלו הוא \$, דבר זה יעיד על כך שקיימת מילה שמסתיימת בצומת זה.</p>	<p><b>17</b></p> <p><b>עץ סייפות</b></p>	

גלגולים בעץ AVL:

הקדמה: גלגול סביב קדקוד מסוים 'X' לכיוון ימין הכוונה שאנחנו לוקחים את הברן השמאלי של הקודקוד X והופכים אותו להיות במקום האבא שלו, ו X – האבא שלו, הופך להיות הברן הימני שלו. ומי שהיה הברן הימני של הברן השמאלי של X (מי שהחליף את X) יהיה הברן השמאלי של X.

גלגול סביב קדקוד מסוים 'X' לכיוון שמאל הכוונה שאנחנו לוקחים את הברן הימני של הקודקוד X והופכים אותו להיות במקום האבא שלו, והאבא שלו – X, הופך להיות הברן השמאלי שלו. ומי שהיה הברן השמאלי של הברן הימני של X (מי שהחליף את X) יהיה הברן הימני של X.

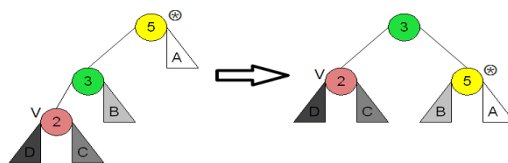
הגלגולים מתחלקים לארבעה סוגים, LR,RL,RR,LL. דבר ראשון נתחיל בזיהוי המקרה שבו אנו נמצאים.

נסמן ב V את הקדקוד שאנו הכנסנו בשלב האחרון, וכן נסמן ב \* את האב הקדמון הצעיר ביותר שבו הופר האיזון. (נוכל למצוא את האב הקדמון הצעיר ביותר שבו הופר האיזון ע"י כך שנתחיל מהצומת שנכנס עכשיו לעלות לאבא שלו וכן לסבא שלו עד שנגיע לצומת הראשון שבו הופר האיזון) ואז נשלים את המשפט " V נכנס בתת עץ \_\_\_\_\_ של בן \_\_\_\_\_ של \* " כאשר באזורים הריקים נוכל לשים או "ימני" או "שמאלי".

אם נכתוב שמאלי, שמאלי אז זה יהיה המקרה LL. במקרה זה אנו נבצע גלגול ימינה סביב \*.  
אם נכתוב ימני, שמאלי אז זה יהיה המקרה LR. במקרה זה אנו נבצע קודם גלגול שמאלה סביב הברן של \*  
ורק אחר כך גלגול ימינה סביב \*.

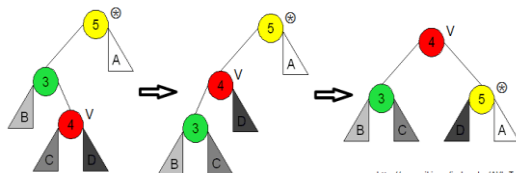
אם נכתוב ימני, ימני אז זה יהיה המקרה RR. במקרה זה אנו נבצע גלגול שמאלה סביב \*.  
אם נכתוב שמאלי, ימני אז זה יהיה המקרה RL. במקרה זה אנו נבצע קודם גלגול ימינה סביב הברן של \*  
ורק אחר כך גלגול שמאלה סביב \*.

(כל התמונות הן דוגמא, יכול להיות ש \* יהיה יותר רחוק מ V)  
: LL



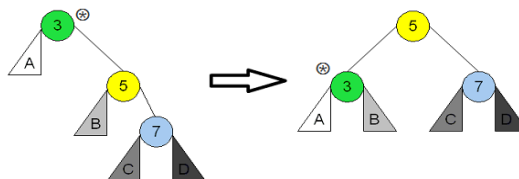
[http://occcwiki.org/index.php/AVL\\_Trees](http://occcwiki.org/index.php/AVL_Trees)

: LR



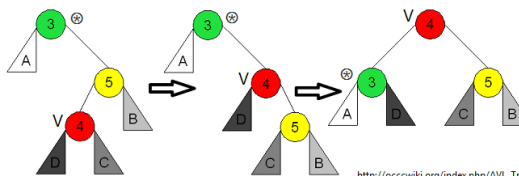
[http://occcwiki.org/index.php/AVL\\_Trees](http://occcwiki.org/index.php/AVL_Trees)

: RR



[http://occcwiki.org/index.php/AVL\\_Trees](http://occcwiki.org/index.php/AVL_Trees)

: RL

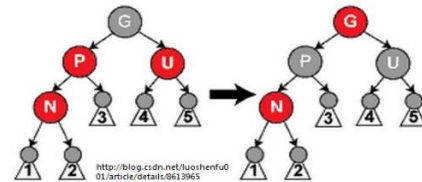


[http://occcwiki.org/index.php/AVL\\_Trees](http://occcwiki.org/index.php/AVL_Trees)

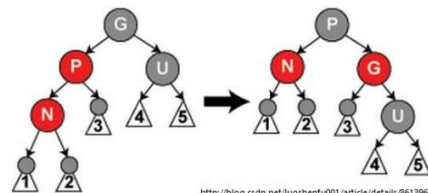
גלגולים בעץ אדום שחור :

הגלגול מתחלק לשלושה מקרים :  
 הערה : עלי ה null נחשבים כעלים שחורים. ובנוסף, הגדרות LL, RR, LR, RL נובעות מאותה הגדרה של עצי AVL.

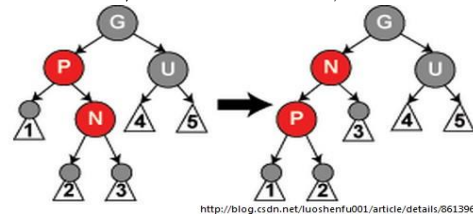
הערה חשובה : במקרה של עץ אדום שחור, להבדיל מעץ AVL, גם אם תיקנו את התת עץ הבעייתי אנחנו צריכים לבדוק רקורסיבית את כל המסלול עד השורש הראשי של כל העץ כיוון שיכול להיות שהסידור המקומי הזיק לחלק העליון בעץ.  
 שהאבא והדוד של הצומת החדש אדומים.  
 שהאבא אדום והדוד שחור. מקרה זה מתחלק לשני מקרים :  
 הצומת החדש הוא LL או RR.  
 הצומת החדש הוא LR או RL.  
 במקרה א' אנחנו נבצע רק צביעה חדשה של הצמתים, את האבא והדוד של הצומת החדש אנחנו נבצע בשחור ואת הסבא נבצע באדום.



מקרה ב' 1. אנחנו נבצע את האבא בשחור ואת הסבא באדום ואז נבצע גלגול על הסבא לכיוון המתאים אם זה LL אז מבצעים גלגולים ימינה ואם זה RR אז מבצעים גלגולים שמאלה.



מקרה ב' 2. קודם כל נבצע גלגול סביב האבא, במקרה של RL עושים גלגול ימינה ואם זה LR אז עושים גלגול שמאלה, ואז אנחנו נגיע למקרה ב' 1.



אתרים מומלצים :

---

- [/http://www.geeksforgeeks.org/data-structures](http://www.geeksforgeeks.org/data-structures)
- [/http://bigocheatsheet.com](http://bigocheatsheet.com)
- <https://www.cs.usfca.edu/~galles/visualization/Algorithms.html>
- <http://www.growingwiththeweb.com/p/explore.html?t=Data%20structure>