

מבנה נתונים

פתרון 5 - עצים

1. בהינתן עץ ביני (לא עץ חיפוש) המכיל מספרים ששורשו root , כתבו פונקציה שמחזירה את מספר הצמתים שערכם גדול מהערך של אביהם.

פתרון – לכל צומת, הפונקציה בודקת האם הערך של הבנים שלו גדולים משלו, וכן מחשבת באופן רקורסיבי בכל אחד מהתעצים שלו את מספר הצמתים שערכם גדול משל אביהם.

```
int BigChild(Node_type* root)
{
    if (root == null)
        return 0;
    int count = 0;
    if ((root->left ) && (root->left->data > root->data))  count++;
    if ((root->right ) && (root->right->data > root->data))  count++;
    return count + BigChild (root->left) + BigChild (root->right);
}
```

נכונות –

תנאי העזירה מתייחס למצב שהצומת לא קיים ולכן מוחזר 0.
לכל אחד מהצמתים בעץ, הפונקציה מבצעת בדיקה מפורשת לגבי בניו, וממשיכה רקורסיבית לבדוק בכל אחד מהתעצים שלו. בכל שלב מוחזר מספר הצמתים שמקיימים את התנאי הדרוש בכל התח-עץ.

סיבוכיות –

לכל צומת בעץ מתבצעות 2 קריאות רקורסיביות (ועוד כמה פעולות קבועות) ולכן זמן הריצה הוא $\Theta(n)$.

שאלה 2

נתון עץ בineriy שורשו root . כתבו פונקציה היוצרת רשימה מקוشرת אשר מחזיקה את האיבר העץ לפי סדר preorder.

תשובה

הfonקציה treeToLinkedList מקבלת את שורש העץ, יוצרת רשימה מקוشرת ריקה ומפעילה את PostList שהיא פונקציה רקורסיבית שמכניסה את צמותי העץ לרשימה בסדר סופי. הרשימה שנוצרת היא רשימה מקוורת עם זkip, שכן בסיום כל הקריאה הרקורסיבית treeToLinkedList מוחקת את האיבר הראשון, ואז מחזירה את ראש הרשימה המקוורת.

הfonקציה PostList מקבלת צומת v ביע וחוליה zws ברשימה המקוורת (האחרונה ברשימה שנבנתה עד כה) ומכניסה רקורסיבית את תת העץ שורשו v אל הרשימה המקוורת החל מהחוליה zws. הפונקציה מחזירה מצביע לאיבר האחרון ברשימה המקוורת.

כדי לבנות מהעץ רשימה מקוורת בסדר סופי post-order המעבר על צמותי העץ מתבצע בסדר הבא : קודם מעבר על תת העץ השמאלי, תת העץ ימני , ואז מעבר על השורש, כאשר בכל פעם הצומת מתווסף בסוף הרשימה המקוורת. לשם כך המצביע לרשימה המקוורת שעובר בקריות הרקורסיביות הוא המצביע לאיבר **הראשון** ברשימה. האיבר הראשון ברשימה נשמר בfonקציה treeToLinkedList, ומוחזר בסוף התהליך.

נדיר (כדי שנוכל להבדיל בין node של עץ לזה של רשימה -> בהנחה שם שונים) :

```
typedef struct {
    int value;
    Tree_Node_Type *left;
    Tree_Node_Type *right;
} TreeNode;
```

```
typedef struct {
    int value;
    List_Node_type *next;
} ListNode;
```

```
ListNode *treeToLinkedList(TreeNode *root)
{
    if (root == NULL)
        return NULL;

    ListNode *head = MakeNode(); // we will use this node as guard (only as a pointer to the list)
    ListNode *curr = head;      //temp pointer running all over the list
    PostList(root, curr);

    curr = head;                // -----
    head = head->next;          // Remove guard from list
    delete(curr);               // -----
    return head;
}
```

```

ListNode *PostList(TreeNode *v, ListNode *curr);
{
    if (v->left)
        curr = PostList(v->left, curr);

    if (v->right)
        curr = PostList(v->right, curr);

    curr->next = MakeNode(v->info);           // Create Next node with current tree node value
    return curr->next;                         // Return next Node (if we will not do it, we will return null
                                                // and we will lose our entire list)
}

```

נכונות : علينا לוודא כי מבקרים בכל צמתי העץ : הפונקציה פועלת על השורש, ובכל פעם מתבצעת קרייה וקורסיבית על תות העץ הימני ותות העץ השמאלי של השורש. מכיוון שככל צומת בעץ נמצא מתחתו הימני או השמאלי של השורש, נקבל באופן אינדוקטיבי כי מבקרים בכל צומת באחת מהתוצאות הרקורסיביות.

סיבוכיות הזמן : כל צומת קורא לפונקציה הרקורסיבית לכל היותר פעמיים – עבור שני בניו, רק במידה והם קיימים ככלומר – מספר הקריאות הרקורסיביות זה כמספר צמתי העץ. הולמת של ביקור בצומת הוא $\Theta(1)$ (רף הגדרה של מצביים), ולכן אם יש n צמותים בעץ, סיבוכיות הזמן הכוללת היא $\Theta(n)$.

דוגמת הרצה : (אני לא יודעת כמה זמן הלינק הזה יחזיק...)
<https://onlinegdb.com/SJxr5LrRr>

3. בהינתן עץ כלשהו המוצג בשיטת בן-شمאלית אחים-ימניים, כתבו פונקציה המדפיסה את כל צמחי העץ, לפי סדר הרמתה בעץ. (ראשון יודפס השורש, אחריו בניו, אח"כ בניו וכן').
רמז: היעזרו בשני תורים.

פתרונות

נשתמש בשני תורים : Brother עברו האחים של הצומת ו-Kids עברו בניו.
 עברו כל צומת אליו הפונקציה מגיעה, מודפסים נתוני הצומת, ואז מכניסים לתור Brother את האח הבא בראשימת האחים של הצומת, ולטור Kids את הבן הראשון של הצומת. בשיטה זו אנו מגיעים לכל צומת (או מאביו או מאחיו), כי כל צומת מכניס לתורים את שני הצמתים שהוא מצביע עליהם.
 כדי שההדפסה תתבצע לפי רמות העץ, החוצאה מן התורים תהיה בסדר זהה :
כל עוד התור Brother לא ריק – הרי שיש בו אחים של הצומת הנוכחי, כולמר צמתים ברמה הנוכחית – **לכן נוציא ממנו את הצומת הבא ונטפל בו**.
כאשר התור Brother מתרוקן – סימן שככל האחים שביהם טיפולנו הכניסו את כל הבנים הראשונים שלהם לתור הבנים – **לכן נוציא צומת מהتور Kids ונטפל בו**. במידה ויש לצומת זה אח – הוא יכניס את אחיו לתור Brother – ולכן באיטרציה הבאה נוציא את האח מטור Brother (שאז כבר לא יהיה ריק). אם אין לו אח – נוציא את הצומת הבא **בתור Kids**.

```
void printTree(Node_type* root)
{
    if (root==null)
        printf("Empty tree");
    Queue_type *Brother, *Kids ;
    Node_type *v;

    printf(root->info);                                // visit tree root.
    if (root->child)
        AddQueue(root->child, Kids);

    while (!(Empty(Brother) || !(Empty (Kids))))
    {
        if !(Empty(Brother))
        {
            DeleteQueue(v, Brother);
            printf(v->info);                           // visit current brother.
            if (v->brothers)
                AddQueue(v->brothers, Brother);
            if (v->child)
                AddQueue(v->child, Kids);
        }
    }
}
```

```

if (Empty(Brother) && !(Empty (Kids)))
{
    DeleteQueue(v, Kids);
    printf(v->info); // visit current child.
    if (v->brothers)
        AddQueue(v->brothers, Brother);
    if (v->child)
        AddQueue(v->child, Kids);
}
}
}

```

נכונות: עליינו לוודא כי [1] מבקרים בכל צמותי העץ, [2] ולפי סדר הרכומות שלו :

[1] הפונקציה מתחילה עם השורש, שמכניס את הבן הראשון שלו לטור הבנים [בלבד, שהרי לשורש אין אחים]. החל מהרמה הבאה בעץ, כאשר הפונקציה פועלת על צומת, היא קודם כל מדיפסה את הצומת עצמו. ואז מכניסה את שני הצמתים שהוא מצביע עליהם – האח הימני שלו והבן השמאלי שלו – כל אחד לטור המתאים לו. מכיוון שלאל כל צומת בעץ יש מצביע או מאביו (אם הוא הבן השמאלי ביותר) או מאחיו שימושו (בתוך הרשימה המקורית של האחים), קיבל כי מכניסים כל צומת לאחד הטרורים.

[2] **נוצר להוכחה – כל הצמתים מרמה L הודפסו ממשmaal לימיין לפני שהווצה מהטור Kids בן שייך לרמה 1+L.**
הוכחה – באינדוקציה על הרמה L בעץ.

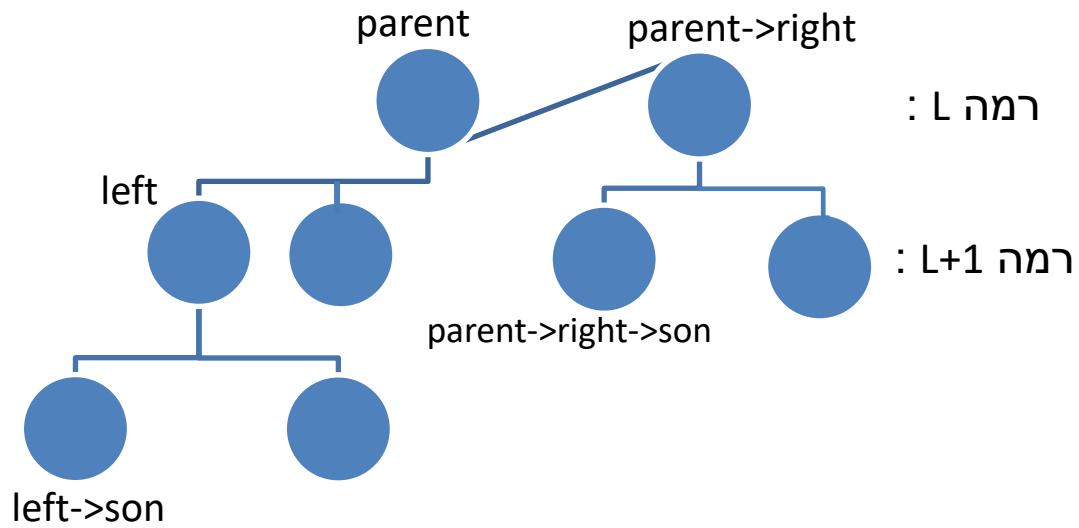
בסיס – בرمאה 0=L: השורש מודפס ראשון (לפני ש모וצאים צומת מיותר כלשהו) ואז מכניס את בנו השמאלי
لتורו.Kids

ברמזה 1=L: כל הצמתים בرمזה 1 הם בניהם של השורש. לאחר שהשורש המכניס את בנו השמאלי לתורו,Kids, בו זה מוצאת מהטור ומכניס את האח שמיימינו [במידה וישנו] לתור brother . לאחר שהטור brother אינו ריק, נטפל בהזה – שיכניס גם הוא את אחיו – וכך נמשיך עד שנסיים את כל האחים שיווכנסו ל-L, כלומר את כל הבנים של השורש. לסיכום – לא נוציאה צומת מהטור Kids [שהוא כבר יהיה שייך לרמה 2], עד שנדפיס את כל הצמתים שברמזה 1. ניתן לראות כי באופן זה הצמתים יודפסו לפי סדרם – ממשaal לימיין [שהרי משתמשים בתור – FIFO – ולכן הצמתים ייכנסו וייצאו לפי סדרם ברשימה המקורית].

עד האינדוקציה – נניח כי כל הצמתים מרמה L הודפסו ממשaal לימיין לפני שהווצה מהטור Kids בן שייך לרמה 1+L.
בסיום ההדפסה, התור brother ריק – כי אין עוד צמתים שהודפסו והאחים שלהם רק הוכנסו לתור ולא יצאו עדיין, שהרי כל הצמתים באותו רמה הודפסו. לעומת זאת, התור Kids אינו ריק, שהרי כל הצמתים מרמה L המכיסו אליו את בנייהם השמאליים.

마חר **ובכל הצמתים מרמה L הודפסו ממשaal לימיין** [הנחת האינדוקציה], הרי שהם המכיסו את בנייהם השמאליים לתור Kids לפי סדרם ממשaal לימיין. לכן הבן הראשון בתור הוא הצומת השמאלי ביותר בرمזה 1+L. נסמן צומת זה ב-left . נסמן את אביו ב-parent , את האח שצמוד לאב מימיין [במידה וישנו] ב-right->right , ונסמן את הבן השמאלי של כל אחד מהם [במידה וישנו] ב-left->son , parent->right->son , parent->right->son .
[ראו תרשימים בעמוד הבא].
כך, left יהיה הראשון בرمזה 1+L שiodפס [ולכן גם הראשון שיכניס את בנו השמאלי son left->son לתור].
אם left->left יש אחים – הוא ייכניס את האח הימני שלו לתור brother , ואז תור זה לא יהיה ריק, ולכן כל אחיו של left יטופלו לפני שייצא צומת נוסף מהטור Kids . לאחר הטיפול בכל האחים של left [במידה וישנו], התור brother יתורזן [לאח הימני ביותר אין אח ימני להכניס לתור זה]. לכן אז נוציאה את הצומת הבא מהטור Kids , שהוא הבן השמאלי של האח הבא של parent שיש לו בניהם, columr parent->right->son [ראו תרשימים].

באופן זה, כל הצמתים מרמה $1+L$ יודפסו משמאלי לימיו, לפני שנתחיל לטפל בצמתים מרמה $2+L$.
מ.ש.ל.



סיבוכיות זמן : נעשה מעבר על כל צמתי העץ ובכל מעבר מתבצעות מספר סופי של פעולות, ولكن העלות הכוללת היא (n) .