8. מחלקת עץ-בינארי

התבניות שבמחלקה

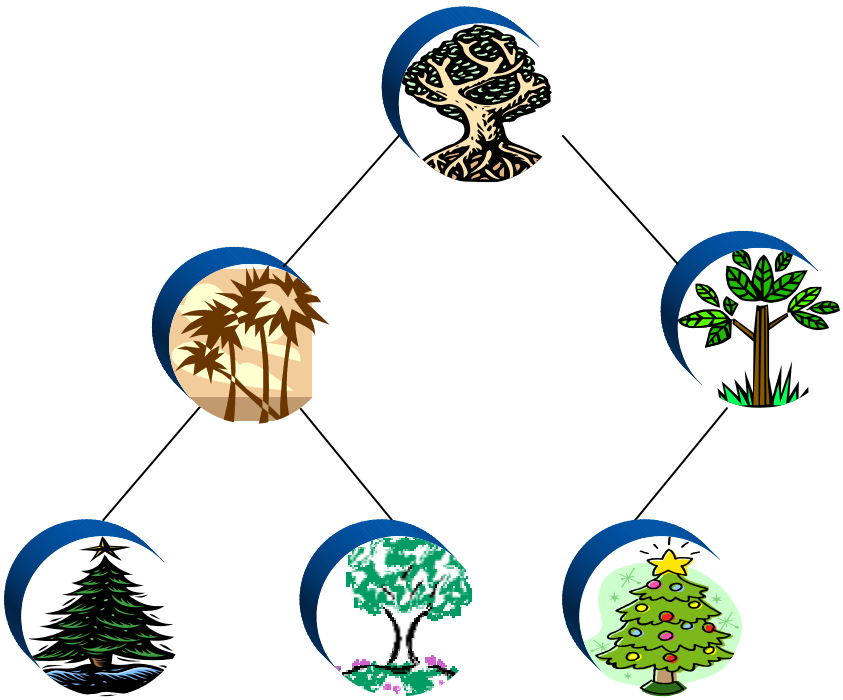
8.1 - בנית עץ

8.2 - סריקה בסדר תחילי - סריקה לעומק

8.3 - סריקה בסדר תוכי - סריקה לעומק

8.4 - סריקה בסדר סופי - סריקה לעומק

8.5 - סריקה לרוחב (בעזרת תור)



עץ הינו מבנה נתונים מתקדם יחסית, והמתקדם ביותר בחומר הלימוד של עיצוב תכנה. השימוש בעץ רלוונטי כאשר יש צורך, או כדאי, לשמור נתונים בצורה היררכית, אשר מבטאת סדר מורכב יותר מסדר לינארי המתאפשר על-ידי רשימה. לעץ יש שורש אחד, ולכל צומת החל מן השורש יש אפס או יותר צאצאים ישירים, הנקראים "בנים". כאשר לצומת יש אפס צאצאים הוא נקרא "עלה". כאשר יש לו לפחות צאצא אחד הוא נקרא "צומת פנימי". כאשר לכל צומת בעץ יש לכל היותר שני בנים, העץ נקרא "עץ בינארי".

אחד מן השימושים בעץ בינארי הוא ייצוג של ביטוי חשבוני, אשר האופרטורים (פעולות חישוב, כגון חיבור או כפל) שבו הם אופרטורים שלכל אחד מהם לכל היותר שני ערכים עליהם הוא מופעל. (למשל, פעולת חיבור מופעלת על שני ערכים; פעולה של העלאה בחזקה מופעלת על ערך בודד) בעץ כזה הערכים עליהם מבוצעים החישובים יישמרו בעלים, והאופרטורים יישמרו בצמתים הפנימיים. מבנה העץ יבטא את סדר ההפעלות של האופרטורים בביטוי החשבוני.

ייצוג ביטוי חשבוני באמצעות עץ בינארי הוא דוגמה אחת לייצוג של נתונים אשר יש להם מאפיינים של ייצוג בינארי. דוגמה נוספת היא ייצוג של טורניר טניס. העלים ייצגו את השחקנים המשתתפים בטורניר, וצמתי הביניים ייצגו את תוצאות המשחקים (המנצחים במשחקים). השורש ייצג את תוצאת (המנצח ב) משחק הגמר. דוגמה אחרת היא ייצוג של סכומים של רצפים של ערכים. כל עלה בעץ ייצג ערך בודד, וכל צומת פנימי ייצג את סכום העלים שבתת-העץ שהוא השורש שלו. ייצוג מסוג זה שימושי בחישובים שונים אשר בהם יש עניין ב"משקל יחסי" של רצפים של ערכים לצורך חישובים שונים. דוגמה נוספת היא ייצוג של מערכת ניתוב אשר יש לה שורש ולכל צומת בה שתי אפשרויות הסתעפות – ימינה או שמאלה.

מבנה היררכי של עץ בינארי שימושי מאד במדעי המחשב לייצוג מבנה נתונים הנקרא ערימה (heap). מבנה זה שימושי לשמירת נתונים לפי גודלם היחסי. בין השאר, מבנה זה שימושי למיון מאד יעיל הנקרא "מיון ערימה", ולעיבודים שונים בהם יש צורך לעדכן שוב ושוב את היחסים בין נתונים שמוכנסים ומוצאים במהלך ביצוע חישובים. שימוש בערימה הינו מעבר לחומר הלימוד ביחידה זו.

בפרק זה מוצגות התבניות המרכזיות השימושיות בעיבודים המשלבים עץ בינארי. תבניות אלה כוללות את בניית העץ, וסריקתו בצורות שונות. שלוש צורות סריקה, הנקראות סריקה בסדר תחילי (PreOrder), סריקה בסדר תוכי (InOrder) וסריקה בסדר סופי (PostOrder) הן סריקות לעומק, אשר מבוססות על גישה רקורסיבית, וניתן גם להמירן לסריקה לא-רקורסיבית המבוססת על שימוש במחסנית. צורת סריקה נוספת היא סריקה לרוחב, ובה נעשה שימוש בתור. רוב השאלות בפרק זה כוללות תרגולים שונים בעיבוד נתונים השמורים בעצים. שאלה בה יש לבחור במבנה הנתונים עץ בינארי כדי לפתור בעיה נתונה מופיעה בפרק השאלות המסכמות בספר.

8.1 - בנית עץ

נקודת מוצא: סדרת נתונים.

מטרה: החזרת עץ בינארי ששומר את סדרת הנתונים. בחירת הבנים של צומת במהלך הבניה נעשית בצורה אקראית.

אלגוריתם: **בנה-עץ-אקראי**

```
אתחל-עץ ← T
איבר-הבא-בסדרה ← x
כל עוד לא סוף הסדרה בצע
    הכנס-באופן-אקראי (T, x)
    איבר-הבא-בסדרה ← x
החזר את T
```

אלגוריתם: **הכנס-באופן-אקראי** (T, x)

```
אם עץ-ריק? (T) אזי
    אתחל-עץ-צומת (x) ← T
אחרת
    קבע-כיוון-אקראי ← d   { 0 תת-עץ שמאלי, 1 תת-עץ ימני }
    אם d = 0, אזי
        אם לא עץ-ריק? (תת-עץ-שמאלי (T)) אזי
            הכנס-באופן-אקראי (x , תת-עץ-שמאלי (T))
        אחרת
            אתחל-עץ-צומת (x) ← T1
            החלף-תת-עץ-שמאלי (T, T1)
    אחרת
        אם לא עץ-ריק? (תת-עץ-ימני (T)) אזי
            הכנס-באופן-אקראי ( x , תת-עץ-ימני (T))
        אחרת
            אתחל-עץ-צומת (x) ← T1
            החלף-תת-עץ-ימני (T, T1)
```

אלגוריתם: **אתחל-עץ-צומת** (x)

```
אתחל-עץ ← ltree
אתחל-עץ ← rtree
בנה-עץ (ltree, rtree, x) ← T
החזר את T
```

הערות

- תבנית זו מציגה את הבניה של עץ בינארי מן השורש כלפי העלים; כלומר, הנתון הראשון בסדרת הנתונים הנתונה יהווה את השורש. כל נתון נוסף יהפוך להיות צומת בתת-העץ השמאלי או תת-העץ הימני של כל אחד מן הצמתים דרכם הוא "מגולגל". כל נתון "מוכנס" לעץ דרך השורש ו"מגולגל" במורד ענף כלשהו עד שהופך להיות עלה חדש בעץ. ייתכן ואביו לא היה צומת פנימי לפני שהנתון "גולגל" דרכו. במקרה כזה אביו הופך מעלה לצומת פנימי כתוצאה מן ה"הכנסה". בדרך כלל, הנתונים שמופיעים בתחילת סדרת הנתונים הופכים להיות צמתים פנימיים ונתונים שבסוף הסדרה נותרים עלים בסוף תהליך הבניה.

- בתבנית המוצגת, הבחירה בכיוון הגלגול היא אקראית; אך ישנן בעיות רבות בהן הבחירה אינה אקראית, אלא נקבעת לפי קריטריון כלשהו. קריטריון מצוי הוא השוואת ערכים – כאשר נתון חדש ה"מגיע" לצומת גדול מהנתון שבצומת, אזי הנתון החדש מגולגל לתת-העץ הימני, אחרת הוא מגולגל לתת-העץ השמאלי. בניה על-פי קריטריון זה היא בניה של "עץ חיפוש בינארי".

- קריטריון אחר לכיוון הגלגול יכול להיות משקל של תתי-עצים. בכל צומת יישמר משקל תת-העץ השמאלי, שהוא מספר הצמתים שבתת-העץ זה. באופן דומה יישמר משקל תת-העץ הימני. נתון המגיע לצומת יגולגל לתת-העץ שמשקלו קטן יותר (אם המשקלים שווים, יגולגל שמאלה). בניה על-פי קריטריון זה מובילה ל"עץ מאוזן", שגובהו הוא המינימלי האפשרי.

- ישנם מקרים בהם ייבנה העץ מן העלים לכיוון השורש; כלומר, תחילה ייווצרו העלים, ורק מאוחר יותר ייווצרו הצמתים הפנימיים, והאחרון שייווצר יהיה השורש. בניה זו מורכבת יותר. זוהי הבניה המומלצת כאשר נבנית ערימה (ראה מבוא לפרק), כיון שסיבוכיותה מינימלית. בניה זו יכולה להיות רלוונטית גם בבניית הייצוג של טורניר טניס.

- ניתן גם לבנות עץ בצורה שתשלב יצירה של העלים ויצירה של הצמתים הפנימיים לפי חוקיות מסוימת. תהליך כזה יקרה בבניית עץ לייצוג ביטוי חשבוני, במהלך מעבר על הביטוי – בכל פעם שייקרא ערך ייוצר עלה חדש, ובכל פעם שייקרא אופרטור ייווצר צומת פנימי חדש, ברמה פנימית מתאימה.

- בתבנית המוצגת, ערכו של כל צומת הוא ערך מסדרת הנתונים, המוצגת בנקודת המוצא. ייתכנו מקרים בהם ערכו של צומת לא יכלול דווקא נתון מסדרת הנתונים, אלא נתון כלשהו הנגזר למשל מחישוב אודות תתי-העצים של הצומת.

- במידה וכל נתון חדש המוכנס מגולגל באותו הכיוון, למשל שמאלה, נבנה עץ מנוון, שהוא בעצם רשימה. זהו מצב שבעצם אין בו ניצול משמעותי של תכונות ההיררכיה של עץ.
  רצוי לתרגל צורות בניה שונות, מעבר לזו המוצגת בתבנית, ובפרט – בניה של עץ מאוזן.

Pascal

```
(*  האלגוריתם בונה עץ בינארי אקראי  *)

function random_tree_build (var T: tree_type);
var
    x: tree_info_type;
begin
    T:= tree_init;
    while not end_of_data do
    begin
        x := get_element;
        insert_randomally (T, x);
    end;
end;

function tree_init_node (x: tree_info_type): tree_type;
var
    ltree, rtree: tree_type;
begin
    ltree:= tree_init;
    rtree:= tree_init;
    tree_init_node:= tree_build (ltree, rtree, x);
end;
```

(* המשך האלגוריתם *)

Pascal

```
procedure insert_randomally (T: tree_type; x: tree_info_type);
var
    T1: tree_type;
    d: integer;
begin
    if tree_is_empty (T) then
        T := tree_init_node (x)
    else
        begin
            d := random (2);
            if d = 0 then                    { add node to left sub-tree }
                if not tree_is_empty (tree_lsub (T)) then
                    insert_randomally (tree_lsub (T), x)
                else
                    begin
                        T1 := tree_init_node;
                        tree_change_lsub (T, T1);
                    end
            else                             { add node to right sub-tree }
                if not tree_is_empty (tree_rsub (T)) then
                    insert_randomally (tree_rsub (T), x)
                else
                    begin
                        T1 := tree_init_node;
                        tree_change_rsub (T, T1);
                    end
        end;
end;
```

C

```
/*  האלגוריתם בונה עץ בינארי אקראי  */

tree_type random_tree_build (void)
  {
     tree_info_type x;

     T =  tree_init;
     while (! end_of_data)
       {
          x = get_element ();
          insert_randomally (T, x);
       }
  }

tree_type tree_init_node (tree_info_type x)
  {
     tree_type ltree, rtre;

     ltree = tree_init ();
     rtree = tree_init ();
     return (tree_build (ltree, rtree, x));
  }
```

C

```
/*  המשך האלגוריתם  */

void insert_randomally (tree_type T, tree_info_type x)
 {
    tree_type T1;
    int d;

    if (tree_is_empty (T))
        T = tree_init_node (x);
    else
     {
        d = random (2);
        if (d = 0)                  // add node to left sub-tree
            if (! tree_is_empty (tree_lsub (T)))
                insert_randomally (tree_lsub (T), x);
            else
              {
                T1 = tree_init_node ();
                tree_change_lsub (T, T1);
              }
        else                        // add node to right sub-tree
            if (! tree_is_empty (tree_rsub (T)))
                insert_randomally (tree_rsub (T), x);
            else
              {
                T1 = tree_init_node ();
                tree_change_rsub (T, T1);
              }
     }
}
```

שאלות

שאלה 8.1.1

פתח וישם אלגוריתם שיקבל כפרמטר מספר שלם n, יקלוט n מספרים שלמים ויבנה מהם עץ חיפוש.

■

שאלה 8.1.2

פתח וישם אלגוריתם שיקבל כפרמטר מספר שלם n ויבנה עץ בעל n צמתים בצורה אקראית, כאשר לכל צומת חייב להתקיים התנאי: ערכו גדול מערך בנו הימני, אם קיים, וקטן או שווה לערך בנו השמאלי, אם קיים. (ההפך מעץ חיפוש)

■

שאלה 8.1.3

פתח וישם אלגוריתם שיקבל סדרת מספרים שכל אחד גדול מקודמו. יש לבנות מסדרת המספרים עץ שרשרת ימני ממוין (לכל צומת בעץ יש רק בן ימני).

■

שאלה 8.1.4

פתח וישם אלגוריתם שיקבל סדרת מספרים שכל אחד גדול מקודמו. יש לבנות מסדרת המספרים עץ שרשרת שמאלי ממוין (לכל צומת בעץ יש רק בן שמאלי).

■

8.2 - סריקה בסדר תחילי

נקודת מוצא: עץ מאותחל.

מטרה: סריקת צמתי העץ בסדר תחילי.

אלגוריתם: **<u>סדר-תחילי</u>** (T)

אם לא סוף-פיקוד? (T) *אזי*

הפעל (T)

סדר-תחילי (*תת-סוף-שמאלי* (T))

סדר-תחילי (*תת-סוף-ימני* (T))

הערות

- תבנית זו מציגה עיבוד בעץ, אשר כולל תחילה עיבוד בצומת האב ורק אחר-כך בתתי-העץ שלו. עיבוד זה מצוי כאשר יש צורך לעבד תחילה את המידע ששמור בצומת אב, ורק אחר-כך בבניו. עיבוד זה רצוי במיוחד אם בצומת שמור מידע אודות תתי-העץ שלו, ויתכן שבעקבות מידע זה אין צורך לבצע עיבוד של תתי-העץ. במקרים כאלה יתכן עיבוד חסכוני אשר ימנע מעבר על כל צמתי העץ. פניה לתתי-העץ תהיה מותנית, ושונה במקצת מזו (הלא-מותנית) המוצגת בתבנית.
- התבנית מציגה שתי קריאות רקורסיביות. ניתן גם לכתוב הסריקה שבתבנית שלא באמצעות רקורסיה, תוך שימוש במחסנית. המעוניינים באתגר מוזמנים להמיר הרקורסיה במחסנית.

Pascal

```
(*  האלגוריתם מקבל עץ בינארי מאותחל וסורק אותו בסדר תחילי  *)

procedure preorder ( T: tree_type);
begin
        If not tree_is_empty (T) then
        begin
                visit (T);
                preorder (tree_lsub (T));
                preorder (tree_rsub (T));
        end;
end;
```

C

```
/*  האלגוריתם מקבל עץ בינארי מאותחל וסורק אותו בסדר תחילי  */

void  preorder (tree_type  T)
{
 if (! tree_is_empty (T))
  {
        visit (T);
        preorder (tree_lsub (T));
        preorder (tree_rsub (T));
   }
}
```

8.3 - סריקה בסדר תוכי

נקודת מוצא: עץ מאותחל.

מטרה: סריקת צמתי העץ בסדר תוכי.

אלגוריתם: **<u>סדר-תוכי</u> (T)**

אם לא עלה-ריק? (T) אזי

סדר-תוכי (תת-עץ-שמאלי (T))

הדפס צ- (T)

סדר-תוכי (תת-עץ-ימני (T))

הערות

- תבנית זו מציגה עיבוד בעץ, אשר כולל תחילה עיבוד של תת-העץ השמאלי, אחר-כך עיבוד בצומת האב ולבסוף עיבוד בתת-העץ הימני. עיבוד זה מצוי כאשר יש צורך לעבד תחילה את המידע שתת-העץ השמאלי. דוגמה למקרה כזה היא איסוף נתונים מעץ בינארי שהינו עץ חיפוש. איסוף זה יציג את הנתונים השמורים בעץ כסדרת ערכים ממוינת.
- התבנית מציגה שתי קריאות רקורסיביות. ניתן גם לכתוב הסריקה שבתבנית שלא באמצעות רקורסיה, תוך שימוש במחסנית. המעוניינים באתגר מוזמנים להמיר הרקורסיה במחסנית.

Pascal

```
(*  האלגוריתם מקבל עץ בינארי מאותחל וסורק אותו בסדר תוכי  *)

procedure inorder (T: tree_type);
begin
        If not tree_is_empty (T) then
        begin
                inorder (tree_lsub (T));
                visit (T);
                inorder (tree_rsub (T));
        end;
end;
```

C

```
/*  האלגוריתם מקבל עץ בינארי מאותחל וסורק אותו בסדר תוכי  */

void inorder (tree_type T)
{
 if  (! tree_is_empty (T))
  {
        inorder (tree_lsub (T));
        visit (T);
        inorder (tree_rsub (T));
   }
}
```

8.4 - סריקה בסדר סופי

נקודת מוצא: עץ מאותחל.

מטרה: סריקת צמתי העץ בסדר סופי.

אלגוריתם: **סדר-סופי** (T)

אם לא עלה-פיקוד (T) אזי

סדר-סופי (תת-עץ-שמאלי (T))

סדר-סופי (תת-עץ-ימני (T))

התייחס ל- (T)

הערות

- תבנית זו מציגה עיבוד בעץ, אשר כולל תחילה עיבוד של שני תתי-העץ של צומת ורק אחר-כך עיבוד הצומת עצמו. עיבוד זה מצוי כאשר יש צורך לעבד תחילה את המידע שבתתי-העץ ואחר-כך להשתמש במידע זה בעיבוד צומת האב. דוגמה לעיבוד מסוג זה הוא חישוב הגובה של עץ. לאחר שיחושב גובהו של כל אחד מתתי-העץ של צומת, ייבחר הגבוה מביניהם כדי לקבוע את גובהו של העץ ששורשו הוא צומת האב.
- התבנית מציגה שתי קריאות רקורסיביות. ניתן גם לכתוב הסריקה שבתבנית שלא באמצעות רקורסיה, תוך שימוש במחסנית. המעוניינים באתגר מוזמנים להמיר הרקורסיה במחסנית.

Pascal

```
(*  האלגוריתם מקבל עץ בינארי מאותחל וסורק אותו בסדר סופי  *)

procedure postorder (T: tree_type);
begin
        If not tree_is_empty (T) then
        begin
                postorder (tree_lsub (T));
                postorder (tree_rsub (T));
                visit (T);
        end;
end;
```

C

```
/*  האלגוריתם מקבל עץ בינארי מאותחל וסורק אותו בסדר סופי  */

void postrder (tree_type T)
{
 if  (! tree_is_empty (T))
  {
        postorder (tree_lsub (T));
        postorder (tree_rsub (T));
        visit (T);
   }
}
```

שאלות

שאלה 8.4.1

פתח וישם אלגוריתם שיקבל כפרמטר עץ בינארי T ויחזיר 'אמת' אם העץ מכיל מספר זוגי של צמתים ו-'שקר' אחרת.

■

שאלה 8.4.2

פתח וישם אלגוריתם שיקבל כפרמטר עץ בינארי T ויחזיר 'אמת' אם ערך כל צומת גדול מערך אביו, ו-'שקר' אחרת.

■

שאלה 8.4.3

פתח וישם אלגוריתם שיקבל כפרמטר עץ בינארי T ויחזיר את מספר הבנים היחידים בעץ.

■

שאלה 8.4.4

פתח וישם אלגוריתם שיקבל כפרמטר עץ בינארי T ויחזיר 'אמת' אם אין בן יחיד בעץ ו-'שקר' אחרת.

■

שאלה 8.4.5

פתח וישם אלגוריתם בסביבת העבודה שיקבל כפרמטר עץ בינארי T ויחזיר את מספר העלים בעץ.

■

שאלה 8.4.6

פתח וישם אלגוריתם שיקבל עץ בינארי T ומספר רמה k, וידפיס את ערכי הצמתים ברמה k.

■

שאלה 8.4.7

פתח וישם אלגוריתם שיקבל כפרמטר **עץ-חיפוש-בינארי** T וידפיס את ערכי העץ בסדר עולה.

■

שאלה 8.4.8

פתח וישם אלגוריתם שיקבל כפרמטר עץ בינארי T ויחזיר את סכום ערכי הבנים היחידים השמאליים.

■

שאלה 8.4.9

פתח וישם אלגוריתם שיקבל כפרמטר עץ בינארי T ויחזיר 'אמת' אם לכל אב לשני בנים ערך הצומת גדול מסכום ערכי כל הצמתים בתת-עץ-שמאלי שלו וקטן מסכום ערכי כל הצמתים בתת-עץ-ימני שלו, ו-'שקר' אחרת.

■

שאלה 8.4.10

פתח וישם אלגוריתם שיקבל כפרמטר עץ בינארי T ויחזיר 'אמת' אם העץ הינו **עץ-חיפוש-בינארי**, ו-'שקר' אחרת.

■

שאלה 8.4.11

פתח וישם אלגוריתם שיקבל כפרמטר עץ בינארי T ויחזיר 'אמת' אם קיים לפחות עלה אחד שערכו גדול מערך כל אב קדמון שלו.

■

שאלה 8.4.12

א. פתח וישם אלגוריתם שיקבל כפרמטר עץ בינארי T ויחזיר 'אמת' אם ערך הצומת של כל אב לשני בנים גדול מערך הצומת שבשורש התת-עץ-שמאלי וקטן מערך הצומת שבשורש התת-עץ-ימני, ו-'שקר' אחרת.

ב. האם עץ זה הוא **עץ-חיפוש-בינארי**?

■

8.5 - סריקה לרוחב

נקודת מוצא: עץ מאותחל.

מטרה: סריקת צמתי העץ לפי רמות.

אלגוריתם: **סריקה-לרוחב** (T)

```
אם לא שלם-פיקוד (T) אזי
    אתחל-תור ← Q
    הכנס-לתור (Q, T)
כל עוד לא תור-פיקוד (Q) בצע
    הוצא-מתור ← T
    התייחס ל- (T)
    אם לא שלם-פיקוד (תת-שמאלי (T)) אזי
        הכנס-לתור (תת-שמאלי (Q, (T)
    אם לא שלם-פיקוד (תת-ימני (T)) אזי
        הכנס-לתור (תת-ימני (Q, (T)
```

הערות

- תבנית זו מציגה עיבוד בעץ, אשר מתבצע בצורה הדרגתית רמה אחרי רמה, כאשר הצמתים בכל רמה הם הצמתים שמרחקם מן השורש הוא אותו המרחק. הרעיון בעיבוד זה הוא של שימוש בתור, אשר אליו מוכנסים תחילה כל הצמתים שבמרחק 1 מן השורש, אחר-כך את כל הצמתים שבמרחק 2 מן השורש, וכך הלאה. סדר עיבוד הצמתים הוא לפי סדר הכנסתם לתור.
- השימוש בתבנית זו מתאים כאשר למשל מעוניינים לזהות את הצומת הקרוב ביותר לשורש שעבורו מתקיימת תכונה מסוימת; למשל, הצומת הקרוב ביותר לשורש שהערך השמור בו גדול מ-100. צורת עיבוד זו מבטאת מעבר שיטתי על צמתים בעץ, אשר עלול לא פעם לחסוך מעבר על כל צמתי העץ, וזאת כיון שכאשר יתגלה הצומת שעבורו מתקיימת התכונה הרצויה, ניתן לסיים את הסריקה. לו היה שימוש באחת משיטות הסריקה המוצגות בתבניות הקודמות, היה <u>בכל מקרה</u> צריך לעבור על כל צמתי העץ.
- בתבנית זו נעשה שימוש בתור, להבדיל מן השימוש ברקורסיה (או מחסנית) שבתבניות הסריקה הקודמות.

Pascal

```
(* פעולה המקבלת עץ בינארי T וסורקת אותו לרוחב (לפי רמות) *)
procedure level_scan (T: tree_type);
var
    Q: queue_type;
begin
    queue_init (Q);
    if  not tree_is_empty (T)  then
        begin
            queue_insert (Q, T);
            while not queue_empty (Q) do
                begin
                    queue_remove (Q, T);
                    visit (T);
                    if not tree_is_empty (tree_lsub (T))  then
                        queue_insert (Q, tree_lsub (T));
                    if not tree_is_empty (tree_ rsub (T))  then
                        queue_insert (Q, tree_ rsub (T));
                end;
        end;
end;
```

C

```
/* פעולה המקבלת עץ בינארי T וסורקת אותו לרוחב (לפי רמות) */
void level_scan (tree_type *T)
 {
    queue_type Q;
    queue_init (Q);
    if ( ! tree_is_empty (T))
      {
        queue_insert (Q, T);
        while (! queue_empty (Q))
          {
            queue_remove (Q, T);
            visit (T);
            if  (! tree_is_empty (tree_lsub (T)))
                queue_insert (Q, tree_lsub (T));
            if  (! tree_is_empty (tree_ rsub (T)))
                queue_insert (Q, tree_ rsub (T));
          }
  }
```

שאלות

שאלה 8.5.1

פתח וישם אלגוריתם שיקבל כפרמטר עץ בינארי T וידפיס את צמתי העץ לפי רמות.

■

שאלה 8.5.2

פתח וישם אלגוריתם שיקבל כפרמטר עץ בינארי T ויחזיר את מספרה של הרמה בעלת מספר הצמתים הגבוה ביותר.

■

שאלה 8.5.3

פתח וישם אלגוריתם שיקבל כפרמטר עץ בינארי T ויחזיר 'אמת' אם כל רמה ממוינת בסדר עולה משמאל לימין.

א. בהנחה שהעץ הוא **עץ חיפוש בינארי,** האם הפעולה תמיד תחזיר 'אמת'? נמק.

ב. אם הפעולה מחזירה 'אמת', האם ניתן להסיק שהעץ T הוא עץ חיפוש בינארי?

■

שאלה 8.5.4

פתח וישם אלגוריתם שיקבל כפרמטר עץ בינארי T ויחזיר 'אמת' אם כל רמה שמספרה זוגי ממוינת בסדר עולה משמאל לימין, וכל רמה שמספרה אי-זוגי ממוינת בסדר יורד משמאל לימין, ו'שקר' אחרת.

■

שאלה 8.5.5

פתח וישם אלגוריתם שיקבל כפרמטר עץ בינארי T וידפיס את ערכי הצמתים החל מהרמה הגבוהה ביותר ועד הרמה הנמוכה ביותר (השורש של העץ).

א. הצע פתרון איטרטיבי.

ב. הצע פתרון רקורסיבי.

■

שאלות סיכום

שאלה 1

פתח אלגוריתם שיקבל כפרמטר עץ-בינארי מאותחל ויחזיר את מספר הצמתים בעץ.

■

שאלה 2

פתח אלגוריתם שיקבל כפרמטר עץ-בינארי מאותחל ויחזיר את מספר הצמתים ברמה k.

■

שאלה 3

פתח אלגוריתם שיקבל כפרמטר עץ-בינארי T ויחזיר את גובה העץ .

■

שאלה 4

עץ בינארי שלם ברמה k (complete binary tree) הוא עץ בינארי שבו כל צומת ברמה k הוא עלה, ולכל צומת ברמה הקטנה מ- k יש תת עץ שמאלי ותת-עץ ימני שאינם ריקים.

פתח אלגוריתם שיקבל כפרמטר עץ-בינארי T ויחזיר 'אמת' אם העץ T הוא **עץ בינארי שלם**, ו-'שקר' אחרת.

מספר העלים **בעץ בינארי שלם** שגובהו n הוא 2^n . נמק!

קבע נכון/לא נכון:

ניתן לטעון כי: אם מספר העלים בעץ שווה ל- 2 בחזקת גובה העץ, אזי העץ הוא **עץ בינארי שלם**.

■

שאלה 5

פתח אלגוריתם שיקבל כפרמטר עץ-בינארי מאותחל ויחזיר את סכום ערכי הצמתים בעץ שאינם עלים.

הצע שתי דרכים שונות לפתרון הבעיה.

■

שאלה 6

נתון עץ ביטוי. פתח אלגוריתם שיקבל כפרמטר מצביע לראש העץ יחשב ויחזיר את ערך הביטוי שהעץ מייצג.

■