**עצים איזומורפיים**

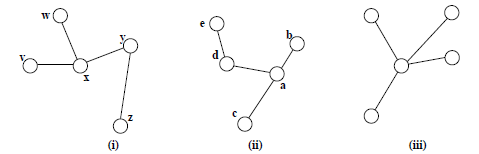
בהינתן שני עצים, האם הם איזומורפיים?

**הגדרה פורמלית**: שני **גרפים** ו- הם **איזומורפיים** אם קיימת פונקצייה חד-חד-ערכית ועל כך שעבור כל מספר הצלעות המקושרות בין כאשר זהה למספר הצלעות המקושרות בין כאשר . פונקציית ההעתקה הזאת נקראת איזומורפיזם.

**הגדרה לא פורמלית**: שני גרפים איזומורפיים כאשר גרף אחד ניתן להפוך לגרף אחר ע"י שינוי שמות הצמתים, כלומר הם מייצגים את אותו הגרף רק "נראים" בצורה שונה.

**דוגמה**:

נתונים 3 עצים: שני עצים ראשונים (i, ii) הם איזומורפיים, עץ שלישי לא איזומורפי לשנים הראשונים.



**עצים עם שורש (rooted trees)** עציםבהם מוגדר שורש. שורש r בגרף G הוא צומת המקיים כי יש מסלול ממנו לכל צומת בגרף. בחישוב אנו משתמש בשורש כצומת שממנו מתחילים סריקה בעץ.

**נראה אלגוריתם אשר בודק האם שני עצים מושרשים (rooted trees) הם אזומורפיים**. כלומר, שני **עצים** ו- עם שורשים בהתאמה הם **איזומורפיים** אם קיימת פונקציית ההעתקה בין העצים כך ש- .

מכאן נובע כי לכל ילד של צומת קיים ילד של צומת  *כך ש- אז תת- עץ עם שורש הוא איזומורפי לתת-עץ עם שורש .*

***אלגוריתם רקורסיבי של פתרון הבעיה****: האלגוריתם רץ על שני עצים מושרשים באופן רקורסיבי וקובע אילו תת-עצים המושרשים בילדי שורשי העץ הם איזומורפיים. בסיום הקריאות הרקורסיביות ניתן לקבוע האם העצים הם איזומורפיים או לא. הפתרון הרקורסיבי לא יעיל כי צריך לזהות כל זוג תת-עצים האם הוא איזומורפי.*

מ*האלגוריתם* הרקורסיבי נובעת מסקנה: לכל צומת בעומק בעץ יש צומת באותו עומק בעץ  *כך ש-אז תת- עץ עם שורש הוא איזומורפי לתת-עץ עם שורש .*

***רעיון לאלגוריתם יותר יעיל****:* תת-העצים אזומורפיים אם הם מכילים אותה כמות הילדים.

אז נקבע סדר בין הילדים של כל צומת כך שנוכל לשוות ביניהם בשני עצים.

*במקום להתחיל סריקה משורש העץ, מתחילים סריקה מהקודקודים שבעומק h כאשר h הוא הגובה של כל עץ. בכל האינטרצית האלגוריתם בודקים האם יש תת-עצים מושרשים בשני העצים באותה רמה. אם כן, עולים רמה כלפי למעלה לכיוון שורש העץ. אם הגענו לשורש באותה רמה (עומק 0) של שני עצים אז עצים איזומורפיים. לכן כל תת-העצים האלו הם איזומורפיים אחד לשני.*

*תת-העצים בעומק h מכילים קדקוד אחד בלבד, עלה. נגדיר משתנה (מספר שלם) שמהווה תווית לעץ אשר מציגה כמות הילדים שיש לשורש העץ. לעצים בעומק h תווית היא 0.*

רמה כלפי למעלה היא בגובה h-1, תת העצים ברמה זאת מכילים שורש וילדים (0 או יותר) בגובה h.

לכל רמה i כך ש- מעדכנים תווית לכל תת-העצים אשר בעומק , תתי העצים בכל רמה הם איזומורפיים אם יש להם אותה תווית.

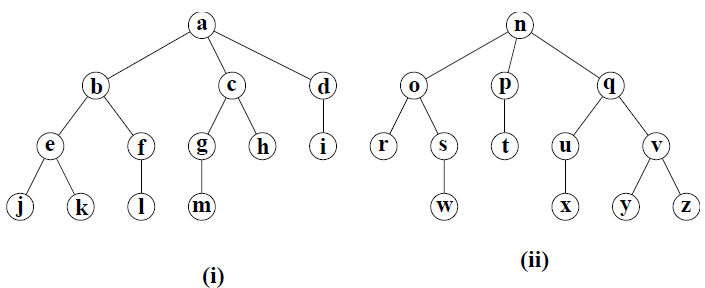
אם עצים איזומורפים, האלגוריתם יוצר התאמות קודם לזוג של שורשים ולאחר מכך באופן רקורסיבי לתת-עצים של ילדי השורשים.

נגדיר לכל צומת בנוסף לתווית, רשימת הילדים ממוינת לפי תווית הילד, רשימת תוויות הילדים ממוינת.

בהתאם למספר הרמות בעצים נגדיר רשימות של צמתים בכל רמה.

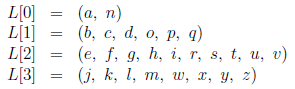
נראה **דוגמה**:

נתונים שני עצים להלן:

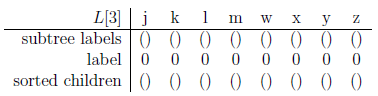


נראה הרצת האלגוריתם:

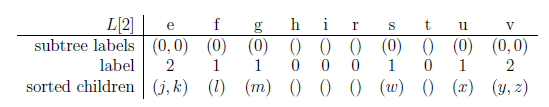
נגדיר רשימות של צמתים בכל רמה בהתאם למספר הרמות בעצים:



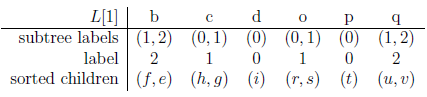
מתחילים סריקה מהעלים כלומר מהרשימה ברמה 3 (לכל צומת מאתחלים תווית, רשימת הילדים ממוינת לפי תווית הילד, רשימת תוויות הילדים ממוינת):



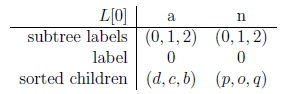
באיטרציה שניה סורקים עומק 2:



באיטרציה שלישית סורקים עומק 1:



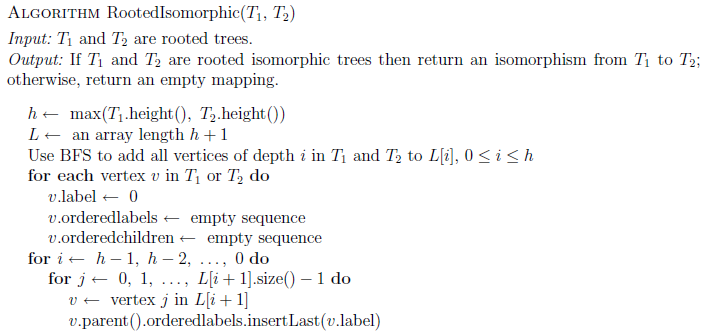
באיטרציה רביעית סורקים שורשים:

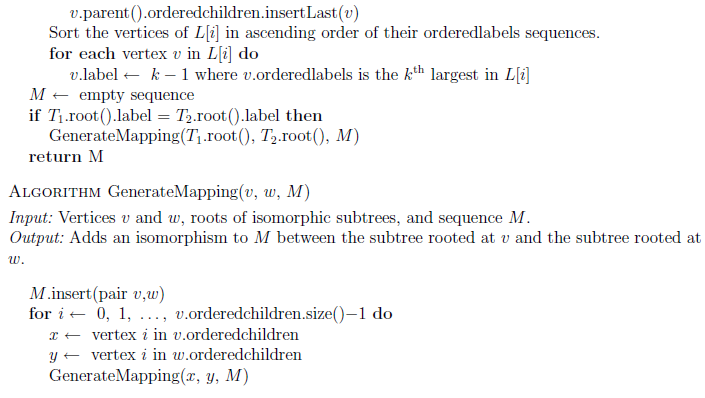


היות ותויות לשורשים של העצים הן 0, אז עצים אזומורפיים. פלט ההתאמות זוגות צמתים בין שני עצים הוא:



האלגוריתם מסדר לכל צומת בעומק i מערך ילדיו (שורשי ילדיו) לפי מערך התווית של ילדיו אשר ממוין לפי סדר לקסוגרפי של תוויות. לדוגמה, מערך L[1] = (b, c, d, o, p, q) לאחר המיון יראה כך L[1] = (d, p, c, o, b, q) כי (0) <= (0) <= (0, 1) <= (0, 1) <= (1, 2) <= (1, 2)





**סיבוכיות האלגוריתם -**

**נראה אלגוריתם נוסף שבודק האם שני עצים מושרשים (rooted trees) הם אזומורפיים - סיבוכיות האלגוריתם -**

**AHU algorithm: Algorithm by Aho, Hopcroft and Ullman**

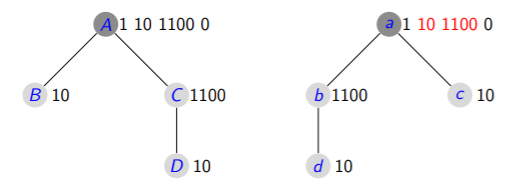
רעיון מבוסס על ביצוע סריקה בעץ לעומק וקביעת סדר בין הבנים של כל קודקוד וכך נוכל להשוות ביניהם.

**טענה**: אם לשני עצים תהיה את אותה מחרוזת הסריקה אז הם יהיו איזומורפיים.

סריקה מבצעים בצורה הבאה. פלט של סריקה תהיה מחרוזת.

* לסרוק את הגרף לעומק.
* כאשר מגיעים לעלה, המחרוזת שלו תהיה '10'.
* כאשר כותבים את המחרוזת של האבא, היא תתחיל ב '1' ותסתיים ב '0' והאמצע יהיה שרשור ממויין של מחרוזות הבנים שלו .

דוגמה:

****

**Algorithm-AHU(v)**

1. if v is a leaf then
2. Give =”10”
3. else
4. for all child w of v do
5. Algorithm-AHU(w)
6. end for
7. end if
8. Sort the names of children of v
9. Concatenate the names of all children of v to
10. =”1”&&”0”

**End-Algorithm-AHU**

**אם לעץ לא מוגדר שורש, אז מה להגדיר בתור שורש העץ?**

**אחת השיטות היא להגדיר מרכז העץ בתור שורש.**

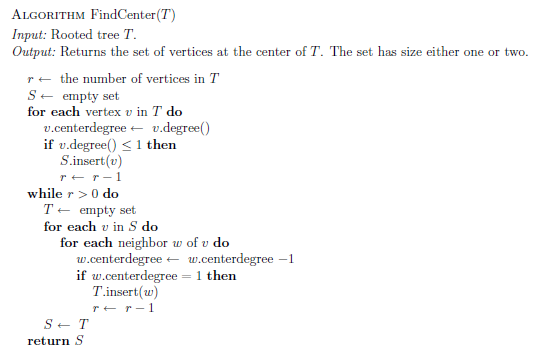
שלב ראשון: למצוא מרכז העץ.

מרכז העץ הוא קבוצת הקודוקדים המרוחקים ביותר מהעלה.

אחד האלגוריתמים למציאת מרכז העץ הוא אלגוריתם השריפה: מחיקת כל העלים עד שנשאר קודקוד אחד או שניים אשר מחוברים בצלע.

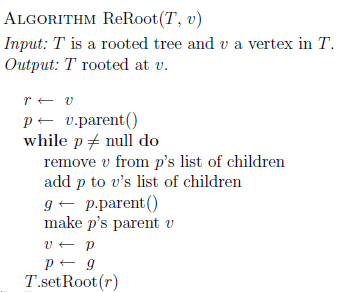
להלן האלגוריתם של מציאת מרכז העץ ללא הסרת הצמתים אלא ע"י מעקב אחרי צמתים בעלי דרגה 1 (עלים), עדכון דרגתם ודרגתם של שכנם:

*סיבוכיות -*



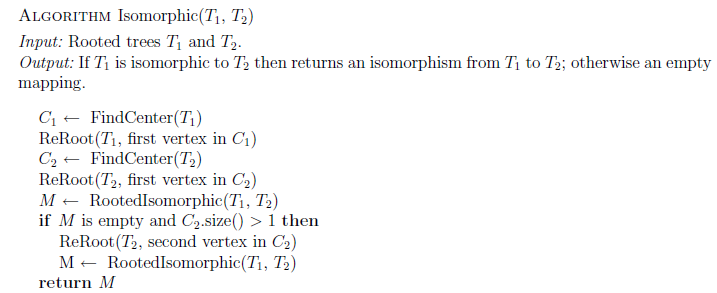
שלב שני: להגדיר בתור שורש את מרכז העץ.

לעץ מרכז מכיל קודקוד אחד או שנים. אם מרכז העץ הוא קודקוד אחד מגדירים לכל עץ קודקוד המרכז שהוא שורש. אם מרכז מכיל שני קודוקודים , אז עצים איזומורפיים כאשר צלע בעץ ראשון ו-צלע בעץ שני וקיים איזומורפיזם בין שני עצים כך ש- או -*. לכן, נגדיר כי שורש של עץ ראשון הוא קודקוד ושורש של עץ שני הוא קודקוד , ונבדוק האם עצים איזומורפיים, אם לא, אז נגדיר בתור שורש של עץ שני את קודקוד ושוב נבדוק האם עצים איזומורפיים.*

**

*אלגוריתם סופי:*

*סיבוכיות -*

**

**Algorithm RootedIsomorphic(, ) // for Algorithm-AHU**

1. Algorithm-AHU()
2. Algorithm-AHU()
3. If then
4. return True
5. else
6. return False
7. End if