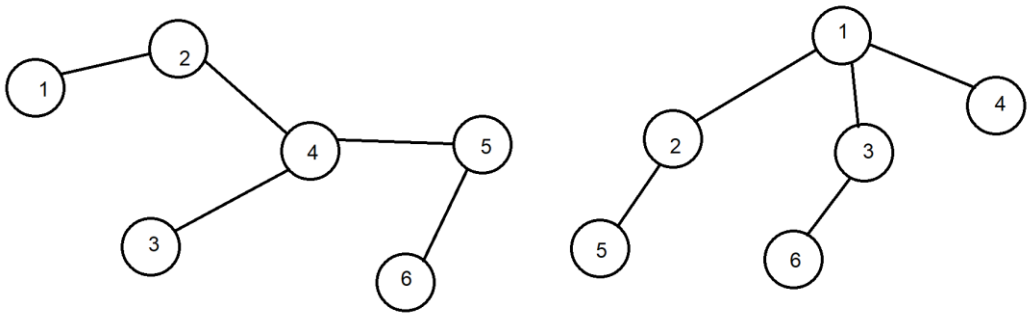
איזומורפיזם של עצים

<u>מטרה:</u> נתונים שני עצים: T_1, T_2. יש לבדוק האם הם איזומורפיים. כלומר, יש להם את אותה מבניות בדיוק, פשוט לא באותו הסדר.
אם העצים איזומורפיים, יש להחזיר את הפונקציה המתאימה בין הקודקודים, אחרת יש להחזיר שאין איזומורפיזם.

<u>דוגמא:</u>



העצים הנ"ל איזומורפיים עם הפונקציה: $h(1) = 5, h(2) = 2, h(3) = 4, h(4) = 1, h(5) = 3, h(6) = 6$.
הרצת האלגוריתם על הדוגמא: בעץ הימני כולם u_i בעץ השמאלי כולם v_i
מציאת המרכזים: שריפת עלים: $\{v_4, u_1\}$
הרצת BFS החל מהמרכז תתן מערך של המרחקים של כל קודקוד מהמרכז שלו.
יוצרים רשימה של כל הקודקודים באותו המרחק מתוך BFS:

$$L[0] = \{v4, u1\}$$
$$L[1] = \{v2, v3, v5, u2, u3, u4\}$$
$$L[2] = \{v1, v6, u5, u6\}$$

לכל קודקוד יהיו: השם שלו, המרחק שלו מהמרכז, label, רשימת הילדים שלו, רשימת ה lable של כל הילדים
מתחילים מהקודקודים במרחק 2. מכניסים לרשימה של האבות את שמות הבנים והlable שלהם.
מסדרים את הרשימה

רשימת lables	רשימת ילדים	lable	מרחק ממרכז	שם
()	()	0	2	v1
(0)	(v1)	1	1	v2
()	()	0	1	v3
(0,1,1)	(v3, v2,v5)	0	0	v4
(0)	(v6)	1	1	v5
()	()	0	2	v6
(0,1,1)	(u4,u2,u3)	0	0	u1
(0)	(u5)	1	1	u2
(0)	(u6)	1	1	u3
()	()	0	1	u4

u5	2	0	()	()
u6	2	0	()	()

לאחר שלב של מרחק 2 קיבלנו 2 סוגי רשימות: (0), $()$.
לאחר המיון, ממספרים את סוגי הרשימות החל מ 0. במקרה שלנו: $()$ תקבל lable 0 והרשימה (0) תקבל lable 1.
לאחר שלב של מרחק 1 קיבלנו סוג רשימה אחת: $(0,1,1)$.

<u>אלגוריתם:</u>

בהינתן T_1, T_2 עצים.

1. $n_1 = |V_1|, n_2 = |V_2|$. אם $n_1 \neq n_2$. החזר: Φ. אחרת: $n = n_1$.
2. מצא את מרכזי העצים C_1, C_2 באמצעות שריפת עלים.
3. לכל קודקוד בכל אחד מהעצים הוסף את השדות:
   $dist, label, childs, labelChilds, parent$
4. עבור כל $c_1 \in C_1$ ועבור כל $c_2 \in C_2$:
   a. הרץ BFS על T_1 החל מקודקוד c_1.
   ועדכן לכל קודקוד את $dist, parent$
   b. הרץ BFS על T_2 החל מקודקוד c_2.
   ושמור את מה שחזר: $dist, parent$
   c. $h_1 = max_{T_1}(dist), h_2 = max_{T_2}(dist)$
   d. אם $h_1 \neq h_2$ החזר Φ. אחרת, $h = h_1$.
   e. הגדר את מערך הרשימות: $L[h+1]$
   f. עבור כל קודקוד v ב T_1, T_2:
      i. $v.label = 0$
      ii. $L[v.dist].add(v)$ - כל קודקוד נכנס לרשימה במיקום שמייצג את המרחק שלו מהמרכז.
   g. עבור i מ $h-1$ עד ל 0 (יורדים ל 0)
      i. עבור כל $v \in L[i+1]$.
         1. $v.parent.childs.add(v)$
         2. $v.parent.lableChilds.add(v.label)$
      ii. $Sort(L[i])$ לפי סדר לקסיקוגרפי של רשימות.
      רשימה ריקה היא ראשונה, ואז רשימות שמתחילות ב 0 ואז ב 1 וכו'
      ואם יש 2 רשימות המתחילות באותו מספר, הולכים לפי המספר השני וכן הלאה כמו במיון מחרוזות.
      iii. הגדר: $k = 0$.
      iv. לכל $v \in L[i]$:
         1. $v.label = k$
         2. אם בקודקוד הבא יש רשימת label שונה אז $k++$
   h. הגדר $f = \Phi$.
   i. אם $c_1.label = c_2.label$ אז קרא לפונקציה: $makePair(f, c_1, c_2)$
   j. החזר את f.

אלגוריתם $makePair(f, c_1, c_2)$

5. $f.add(c_1, c_2)$
6. עבור i מ 1 עד $c_1.childs.size()$
   a. $makePair(f, c_1.childs[i], c_2.childs[i])$

סיבוכיות: $O(n)$