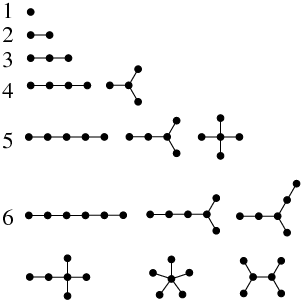
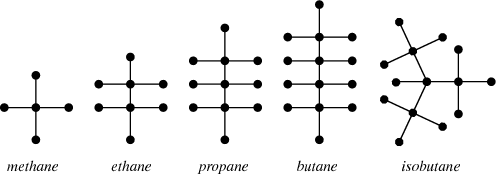
**תרגול 9**

**עץ**

**הגדרה:**

* יהיה G(V, E) גרף לא מכוון (סופי או אינסופי). נאמר כי G הוא **עץ** אם הוא קשיר וללא מעגלים.
* גרף בלתי מכוון ללא מעגלים נקרא **יער (forest or acyclic)**.
* גרף בלתי מכוון קשיר וללא מעגלים נקרא **עץ (חופשי, לא מכוון) (free tree).** במילים אחרות, עץ הוא יער מקושר.





**משפט:** יהיה G(V, E) גרף סופי כך ש- , אז הטענות הבאות שקולות:

1. G הוא עץ.
2. G חסר מעגלים וגם .
3. G קשיר (רכיב קשירות 1) וגם .

**הגדרה:** עלה הוא צומת שדרגתו 1.

**טענה (Berge’s Lemma):** **לכל עץ סופי לא טריויאלי, יש לפחות שני עלים.**

הוכחה: ניקח P - מסלול ארוך ביותר בין שני קדקודי העץ, . P הוא מסלול פשוט וקודקוד ה- הוא עלה. נניח כי הוא לא עלה אז הדרגה שלו תהיה לפחות 2, קודקוד הסמוך אליו, נסמן אותו ב-u, לא יכול להיות שייך ל- P (בעץ אין מעגלים). אם u לא שייך ל-, P אז קיים מסלול ארוך יותר מ-P. קיבלנו סתירה כי P מסלול ארוך ביותר. לכן הוא עלה. בדומה ניתן להוכיח כי גם עלה. מש"ל.

**עץ מכוון** הוא גרף מכוון G(V, E) שמקיים כי גרף התשתית הינו עץ וקיים שורש לגרף. **שורש** הוא צומת בגרף r∈𝑉 שממנה ניתן להגיע לכל צומת בגרף 𝑣∈𝑉 כלומר קיים מ-r מסלול מכוון אל כל אחת מהצמתים בגרף.

הערה: בגרף קשיר היטב, כל הצמתים הינן שורשים.

**יער מושרש** הוא גרף מכוון המורכב ממספר עצים מכוונים.

**משפט:** היה G(V, E) גרף מכוון סופי. נאמר כי G(V, E) הינו עץ מכוון אמ"מ גרף התשתית שלו חסר מעגלים וכן קיימת צומת אחת עבורה דרגת הכניסה (in-degree) ועבור כל שאר הצמתים מתקיים .

**סדרת הדרגות בעץ:**

**למת לחיצת הידיים (handshaking lemma):** אם G=(V,E) הוא **גרף בלתי מכוון**, אזי:

**משפט**: בכל גרף מספר צלעות בעלי דרגות אי-זוגיות הוא זוגי.

**משפט**: בכל גרף יש שני קדקודים שונים בעלי אותה דרגה, כלומר:

**טענה**: **G=(V,E) גרף קשיר לא מכוון הוא עץ אמ"מ**

הוכחה: באינדוקציה על כמות הקודקודים/הצמתים :

אם : טריוויאלי .

אם : טריוויאלי .

נניח עבור , נוכיח עבור . נוריד צלע,

אז נקבל:

**טענה**: **סדרת מספרים היא סדרת ההדרגות בעץ אמ"מ**

**, כאשר**

הוכחה (באינדוקציה על n – מספר הקודוקדים):

בסיס: אם , אז יש סדרת הדרגות היחידה שסכומן שווה ל-2 שהיא 1, 1 שהיא סדרת הדרגות בעץ .

הנחת האינדוקציה:

נניח כי לסדרת המספרים החיובים  *סכום הסדרה שווה ל- , והסדרה הזאת היא סדרת הדרגות בעץ .*

*נוכיח כי לסדרת עם סכום הדרגות קיים עץ עם סדרת הדרגות הללו.*

*נניח כי , היות ו מספר חיובי ו- אז*

*ו- . לכן, זאת סידרה של מספרים חיובים שסכומן של הדרגות הוא .*

*לפי הנחת האינדוקציה, קיים עץ עם .*

*כך ש, ו לכל .*

*ניצור עץ חדש כהעתקה של עץ ומוסיפים קודוקוד וצלע מ- ל-.*

*אז נקבל עץ כך ש-*

*מש"ל.*

**מציאת מרכז, רדיוס וקוטר בעץ**

הגדרות מהגיאומטריה במישור:

מרכז המעגל – נקודה ממנה יוצא רדיוס.

רדיוס המעגל – זה קטע שמחבר בין מרכז לשפת המעגל.

קוטר המעגל – זה קטע הכי ארוך בין שתי קצוות המעגל שעובר במרכז.

גם בעץ יש 3 מושגים לעיל:

**קוטר העץ** הוא מרחק מקסימלי בין שני קודקודים, מסלול ארוך ביותר.

**אקסצנטריות (eccentricity)** של קדקוד x הוא המרחק הגדול ביותר בין x לכל קדקוד אחר.  
 ex(x) = max{distance(x,v), v**∈**V}.   
מהגדרה זו נובע כי **קוטר העץ** הוא אקסצנטריות הגדול ביותר.

**הערה:** לגרפים לא קשירים אקסצנטריות מוגדרת כאינסופית.

**רדיוס העץ** הוא האקסצנטריות הקטן ביותר.  
radius(T) = min{ex(v), v∈V}=min{ { max{distance(v,x), x**∈**V }, v∈V }

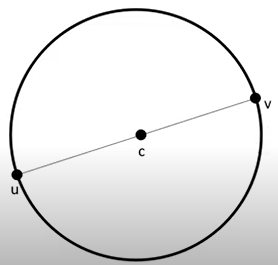
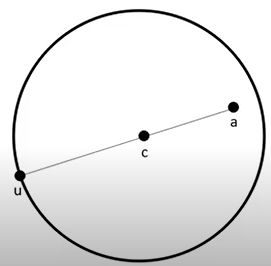
קדקודC הוא **מרכז הגרף** G אמ"מ ex(C)=radius(G)

**מציאת רדיוס, קוטר הגרף בשימוש האלגוריתם F-W:**

מציאת רדיאוס: לאחר הפעלת האלגוריתם F-W, עבור כל קודקוד למצוא מרחק מקסימלי ביותר לקודקוד אחר (max{distance(v,x)) ואחר כך על בין כל המרחקים הללו למצוא מרחק מינימלי min{ex(v)}. סיבוכיות האלגוריתם:

מציאת קוטר: לאחר הפעלת האלגוריתם F-W, למצוא מרחק מקסימלי ביותר. סיבוכיות האלגוריתם:

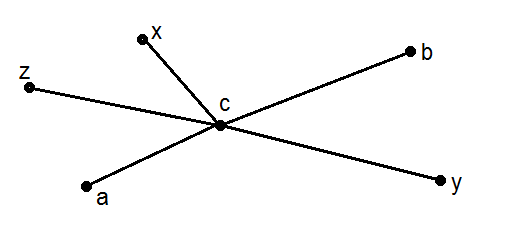
מטרה שלנו למצוא אלגוריתם יותר יעיל עבור עץ בסיבוכיות .



נתונה נקודה a בתוך המעגל, מה הנקודה הכי רחוקה ממנה שנמצאת בשפת המעגל? זאת נקודה u שמסלול אליה עובר דרך נקודת המרכז. אז נמצא מה הנקודה הכי רחוקה מ-u בשפת המעגל. אז מרחק בין u ל-v יהיה קוטר.

**משפט:** בוחרים בקודקוד ∈T x, מוציאים קודקוד y שהוא רחוק ביותר מ- ,xלאחר מכך מוצאים קדקוד z הרחוק ביותר מ- y, אזי dist(y,z) = diameter.

הוכחה בשלילה: נניח שקיים מסלול ארוך ביותר אחר בין a ל- bכך ש- dist(a,b)>dist(y,z). שני המסלולים עוברים דרך מרכז העץ c. קדקוד y רחוק ביותר מ-x, לכן dist(b,c)≤dist(y,c)

באופן דומה dist(c,a)≤dist(c,z), ובכך  
 dist(b,c)+ dist(c,a)≤dist(y,c)+dist(c,z) או dist(a,b)≤dist(y,z).   
 אזי diameter(T) = dist(z,y)

דוגמה לגרף:

Tree: 7 vertices (*n* vertices).

6 edges (*n* − 1 edges).

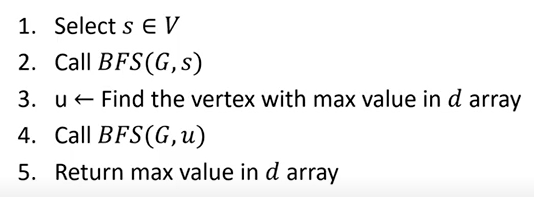
Diam = 5

**נראה אלגוריתם ראשון של מציאת מרכז, רדיוס וקוטר בעץ שמבוסס על שימוש בחיפוש ברוחב:**

רעיון:

נובע מגיאומטריה במישור של מציאת קוטר במעגל.

**מציאת קוטר בשימוש BFS:**

* 1. מפעיליםBFS על קדקוד ראשון ברשימה,
  2. מחשבים את אינדקס (ind) של קדקוד שמרחקו עד הקדקוד הראשון ברשימה גדול ביותר.
  3. שוב מפעילים BFS על קדקוד ind ומחשבים את אינדקס (ind1) של קדקוד שמרחקו עד הקדקוד ind גדול ביותר.
  4. המרחק (מספר צלעות) מקדקוד ind עד קדקוד ind1 הוא הקוטר של העץ.

וסיבוכיות כאן תהיה O(|V|+|E|)

**רדיוס** העץ הוא מרחק הגדול ביותר מהמרכז לעלה.

**מרכז העץ** – נקודה ממנה יוצא רדיוס, כוונה אמצע של הקוטר. כמה מרכזים? 2.

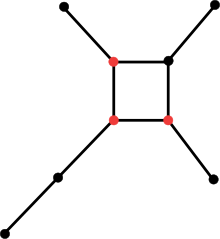
**משפט: לכל עץ T יש מרכז אחד או שניים. (שיטת שריפת עלים).**הוכחה:

המרחק הוא המקסימלי ביותר מקודקוד נתון v לכל קודקוד אחר vi רק כאשר vi הוא עלה.

נתבונן בעץ T בעל יותר משני קדקודים. ידוע לנו של- T יש לפחות שני עלים.   
שרפה ראשונה:   
מוחקים את כל העלים מ -T, הגרף T1 שהתקבל הוא שוב עץ.מחיקת כל העלים מ-T מפחיתה באחד באופן אחיד את האקסצנטריות של כל קדקוד שנשאר (קדקודי עץ T1). לפיכך, המרכזים של T הם גם המרכזים של T1.   
שרפה שנייה:  
נמחק כל העלים מ- T1 – נקבל עץ 2Tבעל אותם מרכזים.   
בהמשך לתהליך השריפות, נקבל קודקוד בודד, שהוא המרכז של T, או צלע אשר קודקודי הקצה שלה הם שני המרכזים של T.

**אִינְווָריִאָנְט** (שמורה): תכונה שאינה משתנה תחת הפעלת טרנספורמציה מסוימת.

**מונו-אִינְווָריִאָנְט** – תכונה שמשתנה תחת הפעלת טרנספורמציה מסוימת.

**הערה**: בגרף שהוא לא עץ מספר מרכזים יכול להיות גדול מ-2:

בגרף זה יש 3 מרכזים המסומנים באדום, האקסצנטריות של כל אחד מהם היא 3.

**בעץ יש 1 או 2 מרכזים.**

**מציאת מרכז/ים, רדיוס העץ** - נשלים את האלגוריתם לעיל:

* 1. אם ערך הקוטר הוא מספר אי-זוגי, אז יש 2 מרכזים, אם ערך הקוטר הוא מספר זוגי, אז יש מרכז אחד.

וסיבוכיות כאן תהיה O(|V|+|E|)

עבור עצים , לפי כך

**אלגוריתם נוסף של מציאת מרכז, רדיוס וקוטר בעץ – שריפת עלים**

רעיון:

מחיקת עלים בעץ מאפשרת לשמור על תכונות העץ: גרף קשיר, ללא מעגלים, .

תנאי העצירה למחיקת העלים כאשר נאשר קודקוד אחד או שנים.

דוגמה:

**centers: 1, 4 radius = 3 diameter = 5**

בסיום רעיון האלגוריתם נשאלות שאלות:

הקוטר זוגי או לא זוגי? אם נאשר קודקוד אחד אז קוטר זוגי, אם נשארו שני קודקודים אז קוטר אי-זוגי.

כמה מרכזים יש בעץ? כמה קודקודים נשארו (אחד או שנים).

מהו הרדיוס? כמות השריפות? אם זה תקין עבור עץ עם שני מרכזים?

קוטר הוא פעמים של הרדיוס. אם זה תקין עבור עץ עם שני מרכזים?

**מסקנה 1:**

כאשר לעץ יש מרכז אחד - **הרדיוס** של העץ שווה למספר השרפות.   
כאשר לעץ יש שני מרכזים - **הרדיוס** של העץ שווה למספר השרפות פלוס 1.

**מסקנה 2:**

כאשר לעץ יש מרכז אחד - **קוטר** של העץ שווה ל-2R.   
כאשר לעץ יש שני מרכזים - **קוטר** של העץ שווה ל-2R-1.

2\*radius-1≤diameter≤2\*radius

**{c1,c2} → d=2\*r-1, {c}→ d=2\*r**

**מימוש האלגוריתם:**

**קלט (מבנה הגרף): רשימת סמיכויות (מערך שכנויות):**

רשימת הסמיכויות מורכבת ממערך צמתי הגרף.

* תא המתאים לצומת v במערך מכיל את רשימת כל הצמתים (שכנים) אשר מחוברים לצומת v.
* מבנה הנתונים המקובל לצורך זה הוא מערך של רשימות מקושרות (linked list).

דוגמה: הגרף שלמעלה מוצג בצורה הבאה:

|  |  |
| --- | --- |
| **vertex** | **adjacency vertexes** |
| **0** | **1** |
| **1** | **0 2 4** |
| **2** | **1 3** |
| **3** | **2** |
| **4** | **1 5** |
| **5** | **4 6** |
| **6** | **5** |

שלב הבא : לזהות את העלים. לכל צומת (תא במערך צמתי הגרף), לקבל גודל של רשימת הצמתים (שכנים) שמחוברים לצומת זה. אם גודל של רשימה מקושרת של הצמתים שמחוברים לצומת הוא 1, אז הצומת הוא עלה.

**מימוש ראשון (שימוש בתור של עלים):**

נשמור כל עלים וינהל אותם באמצעות מבנה הנתונים תור. עבור כל עלה (מוחקים אותו מהתור) מזהים מהו שכן שלו ומעדכנים את רשימה מקושרת של עלה (מוחקים את צומת השכן) ואת רשימה מקושרת של צומת השכן (מוחקים את צומת העלה). לאחר עדכון/מחיקה ברשימה מקושרת של צומת השכן, בודקים האם צומת השכן הפך לעלה, אם כן, להכניס את הצומת לתור. חוזרים לפעולות הללו עד שהתור יכיל צומת אחד וגודל של רשימה מקושרת של הצמתים שמחוברים לצומת זה הוא 1 או 0 (או לכל היותר נשאר קודקוד אחד או שנים במערך צמתי הגרף). אם גודל של רשימה מקושרת של הצמתים שמחוברים לצומת זה הוא 1,אז יש בעץ שני מרכזים (צומת ושכן שלו), אם גודל הוא 0, אז יש בעץ מרכז אחד והוא הצומת.

1.

|  |  |  |
| --- | --- | --- |
| **vertex** | **adjacency vertexes** | **Q** |
| **0** | **1** | **0** |
| **1** | **0 2 4** | **3** |
| **2** | **1 3** | **6** |
| **3** | **2** |  |
| **4** | **1 5** |  |
| **5** | **4 6** |  |
| **6** | **5** |  |

2.

|  |  |  |
| --- | --- | --- |
| **vertex** | **adjacency vertexes** | **Q** |
| **0** |  | **3** |
| **1** | **2 4** | **6** |
| **2** | **1 3** |  |
| **3** | **2** |  |
| **4** | **1 5** |  |
| **5** | **4 6** |  |
| **6** | **5** |  |

3.

|  |  |  |
| --- | --- | --- |
| **vertex** | **adjacency vertexes** | **Q** |
| **0** |  | **6** |
| **1** | **2 4** | **2** |
| **2** | **1** |  |
| **3** |  |  |
| **4** | **1 5** |  |
| **5** | **4 6** |  |
| **6** | **5** |  |

4.

|  |  |  |
| --- | --- | --- |
| **vertex** | **adjacency vertexes** | **Q** |
| **0** |  | **2** |
| **1** | **2 4** | **5** |
| **2** | **1** |  |
| **3** |  |  |
| **4** | **1 5** |  |
| **5** | **4** |  |
| **6** |  |  |

5.

|  |  |  |
| --- | --- | --- |
| **vertex** | **adjacency vertexes** | **Q** |
| **0** |  | **5** |
| **1** | **4** | **1** |
| **2** |  |  |
| **3** |  |  |
| **4** | **1 5** |  |
| **5** | **4** |  |
| **6** |  |  |

6.

|  |  |  |
| --- | --- | --- |
| **vertex** | **adjacency vertexes** | **Q** |
| **0** |  | **1** |
| **1** | **4** |  |
| **2** |  |  |
| **3** |  |  |
| **4** | **1** |  |
| **5** |  |  |
| **6** |  |  |

**מימוש שני (שימוש במערך עזר של דרגות הצמתים):**

בעזרת מערך הדרגות של הצמתים מזהים את העלים, נשמור כל עלים וינהל אותם באמצעות מבנה הנתונים תור. עבור כל עלה (מוחקים אותו מהתור) מזהים מהו שכן שלו ומעדכנים לעלה את הדרגה שלו (מקטינים ב-1) ומעדכנים לצומת השכן את הדרגה שלו (מקטינים ב-1). לאחר העדכון בודקים האם צומת השכן הפך לעלה (דרגתו 1), אם כן, מכניסים את הצומת לתור. חוזרים לפעולות הללו עד שהתור יכיל צומת אחד ודרגתו הוא 1 או 0. אם דרגה היא 1,אז יש בעץ שני מרכזים (הצומת ושכן שלו), אם דרגה היא 0, אז יש בעץ מרכז אחד והוא הצומת.

1.

|  |  |  |  |
| --- | --- | --- | --- |
| **vertex** | **adjacency vertexes** | **Q** | **degree[]** |
| **0** | **1** | **0** | **1** |
| **1** | **0 2 4** | **3** | **3** |
| **2** | **1 3** | **6** | **2** |
| **3** | **2** |  | **1** |
| **4** | **1 5** |  | **2** |
| **5** | **4 6** |  | **2** |
| **6** | **5** |  | **1** |

2.

|  |  |  |  |
| --- | --- | --- | --- |
| **vertex** | **adjacency vertexes** | **Q** | **degree[]** |
| **0** |  | **3** | **0** |
| **1** | **2 4** | **6** | **2** |
| **2** | **1 3** |  | **2** |
| **3** | **2** |  | **1** |
| **4** | **1 5** |  | **2** |
| **5** | **4 6** |  | **2** |
| **6** | **5** |  | **1** |

3.

|  |  |  |  |
| --- | --- | --- | --- |
| **vertex** | **adjacency vertexes** | **Q** | **degree[]** |
| **0** |  | **6** | **0** |
| **1** | **2 4** | **2** | **2** |
| **2** | **1** |  | **1** |
| **3** |  |  | **0** |
| **4** | **1 5** |  | **2** |
| **5** | **4 6** |  | **2** |
| **6** | **5** |  | **1** |

4.

|  |  |  |  |
| --- | --- | --- | --- |
| **vertex** | **adjacency vertexes** | **Q** | **degree[]** |
| **0** |  | **2** | **0** |
| **1** | **2 4** | **5** | **2** |
| **2** | **1** |  | **1** |
| **3** |  |  | **0** |
| **4** | **1 5** |  | **2** |
| **5** | **4** |  | **1** |
| **6** |  |  | **0** |

5.

|  |  |  |  |
| --- | --- | --- | --- |
| **vertex** | **adjacency vertexes** | **Q** | **degree[]** |
| **0** |  | **5** | **0** |
| **1** | **4** | **1** | **1** |
| **2** |  |  | **0** |
| **3** |  |  | **0** |
| **4** | **1 5** |  | **2** |
| **5** | **4** |  | **1** |
| **6** |  |  | **0** |

6.

|  |  |  |  |
| --- | --- | --- | --- |
| **vertex** | **adjacency vertexes** | **Q** | **degree[]** |
| **0** |  | **1** | **0** |
| **1** | **4** |  | **1** |
| **2** |  |  | **0** |
| **3** |  |  | **0** |
| **4** | **1** |  | **1** |
| **5** |  |  | **0** |
| **6** |  |  | **0** |

**מימוש שריפת עלים בשימוש מערך עזר של דרגות הקודקודים, פסדו-קוד:**

**void fire(ArrayList<Integer>[] tree)**

n = tree.length, nVert = n

Queue<Integer> leaves

degrees[n], levels[n]

for i=0 to n-1 //2m=2(n-1)→O(n) = O()

degrees[i] = tree[i].size()

if (degrees[i] == 1) leaves.add(i)

end-for

maxLevel = 0

while (nVert > 2){ //2m=2(n-1)→O(n) = O()

leaf = leaves.poll()

degrees[leaf] = 0

v = tree[leaf].get(0)

degrees[v]--

tree[v].remove(leaf)

nVert--

if (degrees[v] == 1)

leaves.add(v)

levels[v] = levels[leaf] + 1

maxLevel = *max*(maxLevel, levels[v])

end-if

end-while

ArrayList<Integer> centers;

for i=0 to n-1

if (levels[i] == maxLevel) centers.add(i)

end-for

numCenters = centers.size()

if (numCenters == 2)

radius = maxLevel + 1

diameter = 2\*radius - 1

else

radius = maxLevel

diameter = 2\*radius

end-if

print(centers, radius, diameter)

**end-fire**

**וסיבוכיות כאן תהיה O(|V|)**