טענה (Berge) לכל עץ בעל $n \geq 2$ קדקודים יש לפחות שני עלים.

הוכחה באינדוקציה.

בסיס: $n = 2$  לעץ זה ישני קדקודים.

הנחת אינדוקציה: הטענה נכונה עבור n כלשהו.

שלב ההוכחה: יש להוכיח את הטענה עבור $n + 1$.

יש לציין כי בהוספת קדקוד חדש לעץ הקדקוד החדש הוא עלה, כוון אם הוא לא עלה אז סוגרים מעגל והגף החדש הוא כבר לא עץ. כלומר, כאשר לעץ בעל n קדקודים מוסיפים קדקוד חדש, מספר עלים נאו שנשאר ללא שינוי או גדל באחד., כאשר מוסיפים קדקוד חדש לעלה מספר עלים לא משתנה,
כאשר מוסיפים קדקוד חדש לקדקוד שהו לא עלה מספר עלים גדל ב-1.
כלומר בהוספת קדקוד חדש ל-עץ מספר עלים גדול או שווה ל-2.

Invariance and Monovariance Principle

כאן אנו רואים עקרון חדש של monovariant, (mono – ממילה monotony).

When solving problems involving sequences, recursions, or iterative processes, in which one is to determine if a certain state can be achieved, there are two related tools at hand.

An invariant is a function that stays constant when transformations are applied to the object of interest. If the desired result has a different form the invariant value, then one can never arrive at that result.

A monovariant is a function whose value only changes in one direction.

It either always **increases** or always **decreases**.

בבניית עץ בעל סדרה נתונה של הדרגות, ה- ***monovariant*** הוא סכום הדרגות, שקטן.

עוד דוגמה:
נתונה מטריצה n*m שאיבריה הם 1 ו- -1. יש לבנות מטריצה חדשה שסום איברים בכל עמודה ובכל שורה יהיה חיובי. בכל שלב ניתן להפוך את סימן המספרים בעמודה כולה או בשורה כולה. האם ניתן בעזרת פעולה זו להגיע למצב שסכום איברים בכל שורה ובכל עמודה יהיה חיובי?

-1	-1	-1	1	-1
1	1	-1	-1	-1
1	1	1	-1	-1
1	-1	1	1	-1
-1	-1	-1	1	-1

1	1	1	-1	1
1	1	-1	-1	-1
1	1	1	-1	-1
1	-1	1	1	-1
-1	-1	-1	1	-1

1	1	1	-1	-1
1	1	-1	-1	1
1	1	1	-1	1
1	-1	1	1	1
-1	-1	-1	1	1

1	1	1	-1	-1
1	1	-1	-1	1
1	1	1	-1	1
1	-1	1	1	1
1	1	1	-1	-1

משפט נתבונן במערך מלבני עם m שורות ו- n עמודות, שהערכים שלהם הם מספרים ממשיים. מותר להפוך את הסימנים של כל המספרים בכל שורה או עמודה. הוכיח שאחרי מספר פעולות אלה נוכל להפוך את סכום המספרים לאורך כל שורה (שורה או עמודה) לאי-שלילי.

הוכחה. הנה הטריק: נניח שבשורה (עמודה) כלשהי סכום איבריה הוא שלילי ושווה x, כאשר $x < 0$. נסמן בסכום של איברי המטריצה ב-sum. לאחר החלפת סימנים בשורה (עמודה) זו נקבל:

$$newsum = sum - x + (-x) = sum - 2x > sum, \quad (x < 0)$$

לכן, כל שלב באלגוריתם שלנו מוביל לטבלה חדשה עם סכום גדול יותר.

ועכשיו נוכל לראות מדוע האלגוריתם שלנו לא יכול לפעול לנצח. הפעולה שלנו יכולה לייצר רק מספר סופי של מערכים, מכיוון שכל אחד מהערכים בגודל mn יכול לקבל על עצמם רק ערכים ((שונים בסימנם), ולכן בסך הכול יכולות להיות רק מספר סופי של טבלאות שונות שהושגו בדרך זו. לכן יש רק הרבה ערכים שונים של ***sum***. אם האלגוריתם לא יפסיק, הוא יפיק אינסוף ערכים הולכים וגדלים של ***sum***: סתירה.

שתי דרכים למצוא קוטר העץ:

1) שרפת עלים.
2) שימוש ב-BFS.

BFS הוא אחד האלגוריתמים הפשוטים ביותר לסריקת גרף ואב-טיפוס של אלגוריתמים חשובים רבים עבור גרפים.

בהינתן גרף G=(V, E) וקדקוד מסוים s המשמש כמקור (source), BFS בוחן בשיטתיות את הצלעות של G כדי "לגלות" כל קדקוד שניתן להגיע אליו מ-s . BFS מחשב את המרחק (מספר צלעות מינימאלי) מ-s לכל הקדקודים שניתן להגיע אליהם מ-s וכן בונה "עץ רוחב" (“breadth-first tree”) ששורשו s והוא מכיל את כל הקדקודים הללו.

עבור כל קדקוד v שניתן להגיע אליו מ-s המסלול מ-s ל- v בעץ הרוחב הוא המסלול הקצר ביותר מ-s ל- v ב-G. האלגוריתם פועל על גרפים מכוונים ולא מכוונים כאחד. חיפוש לרוחב מכונה כך מפני שהאלגוריתם מגלה את כל הקדקודים הנמצאים במרחק k מ-s לפני שהוא מגלה איזשהו קדקוד שנמצא במרחק k+1 מ-s .

כדי לעקוב אחר התקדמותו, BFS צובע כל קדקוד בלבן, באפור ובשחור. בתחילה כל הקדקודים הם לבנים. קדקוד המתגלה בפעם הראשונה הופך להיות אפור, כלומר לקדקוד אפור יש שכנים לבנים. קדקוד, שכל השכנים שלו התגלו הופך להיות שחור, כלומר שכנים של קדקוד שחור הם אפורים או שחורים.

BFS בונה עץ רוחב, שהשורש שלו הוא המקור s. בכל פעם שמתגלה קדקוד לבן v כשכן של קדקוד u שכבר התגלה, הקדקוד v והצלע (u,v) נוספים לעץ. אומרים ש-u הוא קודם (predecessor) ל-v או האב (parent) של v בעץ הרוחב. מכוון שכל קדקוד מתגלה לכל היותר פעם אחת, יש לו לכל היותר אב אחד.

BFS Pseudo code:

```
G - the graph is represented using adjacency-list
Q - the queue to manage the set of gray vertices
color[] - the color of  vertex u is stored in color[u]
s - the source vertex
p[] - the predecessor (parent) of vertex u is stored
      in p[u]. If u has no predecessor then [u] = nil
d[] - the distance from the source vertex s to vertex u,
      computed by the algorithm is stored in d[u].
nil = -1 inexistent distance and inexistent vertex number
 Adj[u]  - adjacency list of neighboring vertices of vertex u
```

```
BFS(G, s)
    for each vertex u in V[G]
       color[u] = WHITE
       d[u] = nil
       p[u] = nil
    end for
    color[s] = GRAY
    d[s] = 0
    p[s] = nil
    Q = empty

    Q.add(s)

    while (Q != EMPTY)
       u = Q.dequeue()
       for each vertex v in Adj[u]
          if (color[v] == WHITE)
              color[v] = GRAY
              d[v] = d[u] + 1
              p[v] = u
              Q.add(v)
          endif
       endfor
       color[u] = BLACK
    end while
    return (d, p)
end BFS
```

BSF פותר בעיות הבאות:

א) בדיקה אם גרף קשיר:

עוברים על מערך של מרחקים d ובודקים אם נשאר קדקוד u כלשהו שמרחקו עד המקור s הוא אינסופי (d[u]==nil). כאשר קדקוד כזה קיים הגרף אינו קשרי. כאשר שמרחקו של כל קדקודי הגרף עד המקור s הוא מספר סופי – הגרף קשיר.

ב) הדפסת המסלול הקצר ביותר בין שני קדקודי הגרף

```
String getPath (s, v)
        if (d[v]==nil) return null
        path = ""
        if (v==s) path = s
        else
            path = v + path
            t = p[v]
            while (t != nil)
                path = t + "->" + path
                t = p[t]
            end-while
        end-if
        return path
    end-getPath
```

ג) חישוב מספר רכיבי קשירות:

1. אתחול: מגדירים קדקוד התחלתי שהוא קדקוד ראשון ברשימה index = 0
מגדירים count = 1 – מספר רכיבי קשירות של הגרף
2. מפעילים BFS על קדקוד index
3. בודקים האם קיים קדקוד שמרחקו עד קדקוד index שווה אינסוף.
   - ♦ במקרה שקדקוד כזה קיים (קדקוד שמספרו i) מקדמים ב-1 את מספר רכיבי קשירות של הגרף:
count ומעדכנים ערך של index ל- i וחוזרים לשלב 2.
   - ♦ במקרה שקדקוד כזה כבר לא קיים הפונקציה מחזירה את count - מספר רכיבי הקשירות המחושב.

ד) מציאת רכיבי קשירות עצמם. האלגוריתם מאוד דומה לאלגוריתם המתואר בסעיף הקודם:
חוזרים לסעיפים 1,2 ולסעיף 3 מוסיפים שמירת קדקודים הנמצאים באותו רכיב הקשירות.

ה) חישוב קוטר העץ:

א. מפעילים BFS על קדקוד ראשון ברשימה,
ב. מחשבים את אינדקס (ind) של קדקוד שמרחקו עד הקדקוד הראשון ברשימה גדול ביותר.
ג. שוב מפעילים BFS על קדקוד ind ומחשבים את אינדקס (ind1) של קדקוד שמרחקו
עד הקדקוד ind גדול ביותר.
ד. המרחק (מספר צלעות) מקדקוד ind עד קדקוד ind1 הוא הקוטר של העץ.