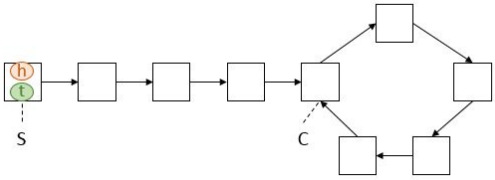
מציאת מעגל ברשימה מקושרת חד-כיוונית עם זרוע

(הארנב והצב / Floyd's Cycle Detection Algorithm)



תיאור הבעיה

- ☚ נתון מסלול מעגלי שמחובר אליו מסלול נוסף - זרוע.
- ☚ קיימים 2 רובוטים - אחד מהיר (הארנב) והשני איטי (הצב).
- ☚ מהירות הארנב גדולה פי 2 ממהירות הצב.
- ☚ נקודת ההתחלה של הצב והארנב היא על הזרוע.
- ☚ הצב והארנב אינם יודעים מתי התחילו את הספירה, אך הם יודעים את נקודת ההתחלה.

הוכחה

נוכיח כי הארנב והצב אכן נפגשים.

נסמן:

n - אורך המעגל ,

m - אורך הזרוע,

k - מספר האיברים מנקודת ההתחלה של המעגל ועד לנקודת המפגש,

p - מספר סיבובי הצב,

q - מספר סיבובי הארנב,

i - מספר הצעדים שעשה הצב,

2i - מספר הצעדים שעשה הארנב.

נציג את המשוואות עבור מספר הצעדים של כל אחד מהם:

מספר הצעדים שעשה הצב הוא $i = m + n \cdot p + k$,

מספר הצעדים שעשה הארנב הוא $2i = m + n \cdot q + k$.

נכפיל את משוואת הצב ב-2 ונשווה: $2m + 2np + 2k = m + nq + k$

קיבלנו שמספר האיברים מנקודת ההתחלה ועד לנקודת המפגש הוא $k = n(q - 2p) - m$.

מכאן נובע ש- $k = n - m$ (לא משנה כמה סיבובים הם עשו),

כלומר הצב והארנב נפגשים במרחק n-m מתחילת המעגל. (אורך המעגל פחות אורך הזרוע).

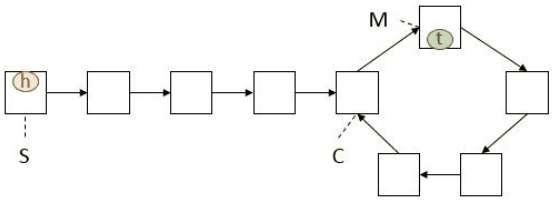
- נשים לב - יש לנו כאן מקרה קצה - עבור המקרה בו $n < m$ ניקח $m = m\%n$.

שאלות למחשבה

נראה פתרון קוד עבור שאלות אלה בדף הבא.

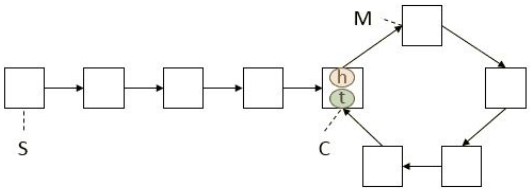
1. החזירו איבר שבוודאות נמצא במעגל
תשובה: נחזיר את האיבר במיקום ה-k (נקודת המפגש).

2. החזירו את נקודת תחילת המעגל
תשובה: לאחר ששני הרובוטים נפגשים, שמים את הארנב בתחילת הרשימה והצב נשאר בנקודת המפגש. שניהם מתחילים לזוז במהירות של הצב. לשני הרובוטים יש לעשות m צעדים עד נקודת ההתחלה של המעגל, כלומר נקודת המפגש שלהם היא נקודת ההתחלה של המעגל.



3. החזירו את אורך הזרוע
תשובה: אורך הזרוע הוא מספר הצעדים שהם עשו בשאלה 2.
פשוט נפעיל counter שיספור לנו את מספר הצעדים עד לנקודת המפגש ואז נבצע שוב את התשובה של שאלה 2.

4. החזירו את אורך המעגל
תשובה: הארנב נשאר בנקודת ההתחלה של המעגל והצב הולך עד שהוא יפגוש את הארנב - מספר הצעדים שהצב עשה - הוא אורך המעגל.



סיבוכיות

- מציאת איבר שבוודאות נמצא במעגל - $O(n+m)$
- מציאת נקודת תחילת המעגל $O(m)$
- מציאת אורך הזרוע $O(m)$
- מציאת אורך המעגל $O(n)$

מימוש פתרון הבעיה בעזרת רשימה מקושרת

מחלקת Node.java

```
public class Node {

   int id;
   static int id_counter = 0;
   Node next;

   public Node() {
       this.next = null;
       this.id = id_counter++;
   }
}
```

מחלקת LinkedList.java (מבנה הנתונים)

```
public class LinkedList {

   Node head, tail;
   int size;

   public LinkedList(){
       this.head = null;
       this.tail = null;
       this.size = 0;
   }

   public void add(Node newNode){
       if(head == null) {
           head = newNode;
           tail = newNode;
       }
       else{
           tail.next = newNode;
           tail = tail.next;
       }
       size++;
   }


   public int size(){
       return size;
   }
```

```
    public Node getNode(int id) {
        Node current = head;
        if(tail.id == id)
            return tail;

        while (current != tail) {
            if(current.id == id)
                return current;
            current = current.next;
        }
        return null;
    }
}
```

מחלקת הפתרון

```
public class HareAndTortoise {

    static Node theKiss = null;
    static int length_of_arm = 0;

    public static boolean isCycled(LinkedList list) {
        if(list.head == null)
            return false;
        Node hare = list.head;
        Node turtle = list.head;

        while (hare.next.next != null) {
            hare = hare.next.next;
            turtle = turtle.next;
            if(hare == turtle) {
                theKiss = hare;
                return true;
            }
        }

        return false;
    }

    public static Node firstNodeOfCircle(LinkedList list) {
        if(list.head == null)
            return null;

        length_of_arm = 0;
        Node turtle = theKiss;
        Node hare = list.head;
```

```
        while (hare.next != null) {
            if (turtle != null) {
                turtle = turtle.next;
            }
            length_of_arm++;
            hare = hare.next;
            if(hare == turtle) {
                return hare;
            }
        }
        return null;
    }

    public static int lengthOfCircle(Node firstNodeOfCircle) {
        int length = 0;
        Node tortoise = firstNodeOfCircle;
        length++;
        tortoise = tortoise.next;
        while (tortoise != firstNodeOfCircle) {
            length++;
            tortoise = tortoise.next;
        }
        return length;
    }
    public static void main(String[] args) {
        LinkedList list = new LinkedList();

        for(int i = 0 ; i < 6 ; i++)
            list.add(new Node());

        System.out.println("isCycled?: " + isCycled(list));
        list.tail.next = list.getNode(2);
        System.out.println("isCycled?: " + isCycled(list));

        // Question 1 - return a node in the circle.
        System.out.println(theKiss.id);
        // Question 2 - return the first node of the circle.
        Node firstNode = firstNodeOfCircle(list);
        System.out.println(firstNode.id);
        // Question 3 - return the length of the arm.
        System.out.println(length_of_arm);
        // Question 4 - return the length of the circle.
        System.out.println(lengthOfCircle(firstNode));
    }
}
```

דור עזריה