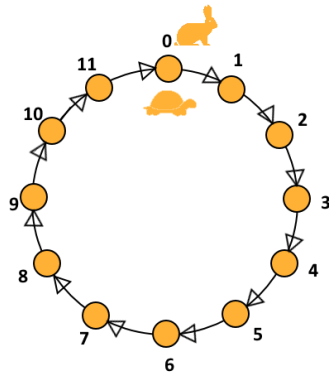


מציאת מעגל ברשימה מקושרת חד-כיוונית בלי זרוע (הארנב והצב)



תיאור הבעיה

- צריך להראות כי המסלול הוא מעגלי.
- נתון מסלול וקיימים 2 רובוטים - אחד מהיר (הארנב) והשני איטי (הצב), מהירות הארנב גדולה פי 2 ממהירות הצב.
- שניהם נעים באותו כיוון במעגל ומתחילים באותו נקודת התחלה.
- כדי להוכיח כי מדובר במסלול מעגלי, נבדוק אם הארנב והצב ייפגשו בשלב כלשהו, אם הם יפגשו - המסלול אכן מעגלי, אחרת המסלול לא מעגלי.

הוכחה

נוכיח כי הארנב והצב אכן נפגשים (כלומר שקיים מסלול מעגלי).
נסמן:

- n - מספר איברי הרשימה,
- k - מספר האיברים מנקודת ההתחלה 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:

$$i = n \cdot p + k$$

$$2i = n \cdot q + k$$

$$2n \cdot p + 2k = n \cdot q + k$$

קיבלנו שמספר האיברים מנקודת ההתחלה ועד לנקודת המפגש הוא $k = n(q - 2p)$. מכאן נובע ש- k היא כפולה של n , כלומר הצב והארנב נפגשים בנקודת ההתחלה.

I

סיבוכיות

- מכיוון שאנו עוברים מספר פעמים על כל האיברים במעגל ובודקים האם קיים מעגל או לא, הסיבוכיות של מציאת מעגל ברשימה מקושרת חד-כיוונית בלי זרוע היא $O(n)$.
- זה לא משנה אם אנחנו עוברים פעם אחת על המעגל או למשל 5 פעמים בסופו של דבר $O(5n) = O(n)$.

מימוש פתרון הבעיה בעזרת רשימה מעגלית

מחלקת Node.java

```
public class Node {  
  
    Node next;  
    int id;  
    static int unique_id = 0;  
  
    public Node() {  
        this.next = null;  
        this.id = unique_id++;  
    }  
  
}
```

מחלקת CircularList.java (מבנה הנתונים)

```
public class CircularList {  
  
    Node head;  
    private int size = 0;  
  
    public CircularList(){  
        this.head = null;  
    }  
  
    public void add(Node newNode) {  
        if(head == null) {  
            head = newNode;  
            newNode.next = head;  
        }  
        else {  
            Node current = head;  
            for(int i = 0 ; i < size-1 ; i++) {  
                current = current.next;  
            }  
            current.next = newNode;  
            newNode.next = head;  
        }  
        size++;  
    }  
  
    public int size(){  
        return this.size;  
    }  
  
}
```

```

public Node getNode(int id) {
    Node current = head;
    for(int i = 0 ; i < size ; i++) {
        if (current.id == id)
            return current;
        current = current.next;
    }
    return null;
}
}

```

מחלקת פתרון

```

public class HareAndTortoiseProblem {

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

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

    public static void main(String[] args) {
        CircularList list = new CircularList();

        for(int i = 0 ; i < 8 ; i++)
            list.add(new Node());

        System.out.println(isCycled(list)); // prints true
        list.getNode(7).next = null;
        System.out.println(isCycled(list)); // prints false
    }
}

```