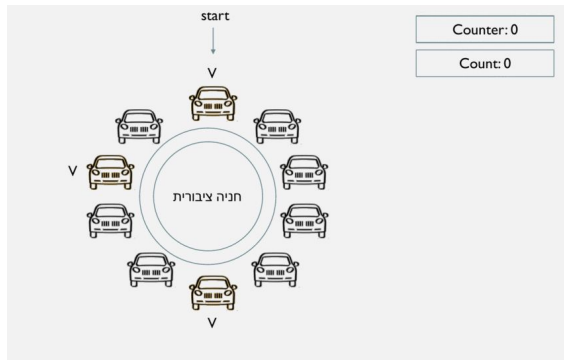


בעיית החניה



תיאור הבעיה

החוקר צריך לספור כמה מכוניות יש בחנייה מעגלית.

- 🚗 אורך המעגל אינו ידוע למחשב.
- 🚗 החנייה גדולה והמחשב רואה רק את המכוניות שנמצאת לידו ואת המכונית הבאה.
- 🚗 המחשב יכול לסמן את המכונית בסימן כלשהו, אך הסימן יכול להופיע כבר על מספר מכוניות.
- 🚗 המחשב יכול למחוק את הסימן הקודם ולכתוב סימן חדש.

אלגוריתם

מבנה נתונים - רשימה מקושרת דו-כיוונית או מערך מעגלי (בעזרת מודולו).

1. נסמן ב- V את הרכב הראשון.
2. נתקדם במעגל ונספור את המכוניות עד שנראה מכונית עם הסימן V .
3. נמחק את הסימן V ונרשום במקומו W .
4. נחזור אחורה לנקודת ההתחלה לפי מספר צעדים שספרנו.
5. אם נראה W - סגרנו מעגל, ומספר המכוניות שספרנו הינו מספר המכוניות במעגל. אחרת, אם נראה V - נחזור לסעיף 2.

סיבוכיות

במקרה הטוב

אין לנו אף מכונית שמסומנת ב- V (חוץ מהמכונית הראשונה שסימנו בהתחלה) ולכן נעבור על מעגל המכוניות פעמיים - (פעם אחת עד שנראה V נוסף ונהפוך ל- W ופעם נוספת כשנחזור אחורה). לכן סך הכל קיבלנו סיבוכיות $O(2n) = O(n)$.

במקרה הגרוע

על כל מכונית במעגל מסומן לנו V ולכן זמן הריצה יהיה סכום של סדרה חשבונית -

$$1 + 1 + 2 + 2 + \dots + n + n = 2(1 + 2 + \dots + n) = 2 \cdot \frac{n(n+1)}{2} = n \cdot (n+1) = O(n^2)$$

מכיוון שהסיבוכיות נמדדת לפי המקרה הגרוע אז סיבוכיות בעיית החניה היא $O(n^2)$.

מימוש פתרון הבעיה בעזרת רשימה מקושרת דו-כיוונית

מחלקת Node.java

```
public class Node {

    String signed;
    Node next,prev;
    static int id_counter = 0;
    int id;
    public Node() {
        this.signed = "null";
        this.next = null;
        this.prev = null;
        this.id = id_counter++;
    }

    @Override
    public String toString() {
        return "{id=" + id +
            ", next= " + this.next.id +
            ", prev= " + this.prev.id +
            ", signed= " + signed + "}->\n";
    }
}
```

מחלקת CircularList.java (מבנה הנתונים)

```
public class CircularList {
    Node head, tail;
    int size;

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

    public void add(Node newNode) {
        if(head == null) {
            head = newNode;
            tail = newNode;
            newNode.next = head;
            newNode.prev = tail;
        }else {
            Node current = tail;
```

```

        current.next = newNode;
        newNode.prev = current;
        newNode.next = head;
        tail = newNode;
    }
    size++;
}

public int size() {
    return this.size;
}

public Node getNode(int id) {
    Node current = head;
    while (current != tail) {
        if(current.id == id)
            return current;
        current = current.next;
    }
    if(tail.id == id)
        return tail;
    return null;
}

public void print(){
    Node current = head;
    while(current != tail) {
        System.out.print(current.id+"->");
        current = current.next;
    }
    System.out.print(tail.id+"->");
    System.out.println();
}
}

```

המחלקה הראשית ParkingProblem.java

```

public class ParkingProblem {

    public static int solution(CircularList list) {
        if(list.head == null)
            return 0;

        Node current = list.head;
        current.signed = "v";
        int current_counter = 1;
    }
}

```

```

    int main_counter = 0;
    boolean flag = false;

    while(!flag) {

        while (!current.next.signed.equals("v")) {
            current_counter++;
            current = current.next;
        }
        current.next.signed = "w";
        current_counter++;
        main_counter = current_counter;
        while (current_counter != 0) {
            current_counter--;
            current = current.prev;
        }
        if(current.signed.equals("w")) {
            flag = true;
        }
    }

    return main_counter;
}

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

    for(int i = 0 ; i < 5 ; i++)
        list.add(new Node());

    list.getNode(1).signed = "v";
    list.getNode(3).signed = "v";
    list.print();
    System.out.println(solution(list));
}
}

```

מימוש פתרון הבעיה בעזרת מערך מעגלי (מודולו)

כדי לממש פתרון כזה צריך בכלל להבין מה זה מודולו? אינטואיציה הקלאסית: חשבון מודולרי הוא מה שכולנו עושים כשאנחנו מנסים לדעת מה תהיה השעה עוד כך וכך שעות. אם עכשיו השעה היא 19:00 ואנחנו שואלים "מה תהיה השעה עוד 10 שעות?" אנחנו מוסיפים 10 ל-19, מקבלים 29, ואז מחלקים ב-24 (מספר השעות ביממה), לוקחים את השארית - 5, וזו התשובה. כאשר השעה כעת היא 20:00, ואנו רוצים לדעת מה תהיה השעה 9 שעות מאוחר יותר, הפעולה שאנו עושים היא $20 + 9 \equiv 5 \pmod{24}$.

מימוש בשיטת מודולו במערכים בדרך כלל נכתבת בצורה הבאה:

```
int[] a = {1,2,3,4,5};
int start = 3;
for (int i = 0; i < a.length; i++) {
    System.out.print(a[(start + i) % a.length] + ",");
}
```

כאשר start הוא מיקום ההתחלתי עבור המערך (חייב להיות בגבולות המערך).

נקבל את ההדפסה הבאה: "4,5,1,2,3".

כי עבור צעד ראשון $a[(3 + 0) \% 5] = a[3 \% 5] = a[3] = 4$ כלומר $3 \equiv 3 \pmod{5}$.

עבור צעד שני $a[(3 + 1) \% 5] = a[4 \% 5] = a[4] = 5$ כלומר $4 \equiv 4 \pmod{5}$.

וכך הלאה...

אם נרצה לצעוד לאחור נחליף במקום "start + i" ל-"start - i + a.length".

```
int[] a = {1,2,3,4,5};
int start = 3;
for (int i = 0; i < a.length; i++) {
    System.out.print(a[(start - i + a.length) % a.length] + ",");
}
```

נקבל את ההדפסה הבאה: "4,3,2,1,5".

המימוש דיי טיפשי כי ידוע לנו אורך המערך ואנו נעזרים בו כדי לפתור באמצעות מודולו.
אבל נניח כי האורך לא נתון לקריאת המשתמש אלא ערך פרטי.

מחלקת פתרון ParkingProblemModulo.java

```
public class ParkingProblemModulo {
    /**
     * 1 means someone marked the car as visited.
     * 2 means that I marked the car as visited.
     * @param arr a numeric array.
     * @return the number of cars.
     */
    public static int solution(int[] arr) {

        int temp_counter = 0;
        int main_counter = 0;
        int start = 1;
        boolean flag = false;
        while(!flag) {

            // go forward
            while(arr[(start + temp_counter) % arr.length] != 1) {
                temp_counter++;
            }

            arr[(start + temp_counter) % arr.length] = 2;
            temp_counter++;
            main_counter = temp_counter;

            // go backward
            while( temp_counter != 0 ) {
                temp_counter--;
            }
            if(arr[temp_counter] == 2) {
                flag = true;
            }
        }
        return main_counter;
    }

    public static void main(String[] args) {
        int[] a = {1,1,1,1,1};
        System.out.println(solution(a));
        a = new int[]{1,1,1,1,1,1,1,1,1,1};
        System.out.println(solution(a));
    }
}
```