מבוא למדעי המחשב - מועד א – פתרונות והערות
סמסטר א תשפ"ב

הערה חשובה מאוד על ערעורים

מאחר ולא ניתן לקיים הליך של "ערעור על ערעור" עליכם לוודא כי הערעור שאתם מגישים מפורט ומנומק היטב. במהלך הטיפול בערעורים על בוחן האמצע קיבלנו לא מעט ערעורים בהם נטען שהפתרון שהוגש נכון (ללא פירוט) ובקשה לבדיקה חוזרת. אנו מדגישים כי ערעורים לא מפורטים ומנומקים היטב ידחו על הסף ללא בדיקה. המלצתנו היא להריץ את הקוד שלכם ולצרפו לערעור מפורט ומנומק אותו תגישו. אנא קיראו בעיון את ההנחיות להגשת ערעור המפורסמות ביחידת ההוראה במודל.

שאלה 1

סעיף א: פתרון לינארי במספר הקודקודים בגרף תוך שימוש במערך עזר

```
public boolean isPartition() {
   int n = input.numberOfVertices();
   if (setA.size() + setB.size() != n)
       return false;
   boolean[] exist = new boolean[n];
   for (Integer e : setA) {
       if (e < 0 | e > n - 1)
           return false;
       exist[e] = true;
   }
   for (Integer e : setB) {
       if (e < 0 | e > n - 1 || exist[e])
           return false;
       exist[e] = true;
   }
   return true;
}
```

הערה: לסעיף זה קיים פתרון לינארי במספר הקודקודים המשתמש במערך עזר. נקודות הורדו רק עבור פתרונות מסורבלים ולא יעילים במיוחד.

סעיף ב': פתרון לינארי בגודל הגרף (|V| + |E|) תוך שימוש במערך עזר:

```
public Boolean runAlgorithm() {
   if (!isPartition()) {
       return false;
   }
   // construct partition array: partition[i] is true if and
   // only if i is in set b
   boolean[] partition = new boolean[input.numberOfVertices()];
   for (Integer i : setB) {
       partition[i] = true;
   }
   for (Edge e: input.edgeSet()) {
       if (partition[e.getLeft()] == partition[e.getRight()]){
           return false;
       }
   }
   return true;
}
```

שאלה 2 - פתרון לדוגמא וטעויות נפוצות:

סעיף א':

```
public static boolean isBalanced(String s) {
     boolean result = true;
     if (s.length() % 2 != 0) {
          result = false;
     }
     int balance = 0;
     for (int i = 0; i < s.length() & result; i = i + 1) {
          if (s.charAt(i) == '(') {
               balance = balance + 1;
          }
          else {
               balance = balance - 1;
          }
          result = (balance >= 0);
     }
     return result & (balance == 0);
}
```

הערות:

תחילה, יש להדגיש שהבדיקה הראשונה אינה הכרחית (או מהותית), ותלויה בכך שבג'אווה בדיקת האורך של מחרוזת אינה תלויה באורכה. על מנת לוודא שמחרוזת היא מאוזנת, עלינו לבדוק שני תנאים מרכזיים:

1. אין מקום במחרוזת שעד אליו מספר הסוגרים **גדול** ממספר הפותחים. כלומר ())(***אינה*** מחרוזת מאוזנת.
2. בסוף המחרוזת מספר הפותחים ומספר הסוגרים **שווה**. כלומר (()) ***אינה*** מחרוזת מאוזנת.

עובדה זו ניתנת לייצוג על ידי משתנה יחיד, וניתנת לחישוב במעבר בודד.

טעויות רבות שראינו הן:

1. סטודנטים אשר בודקים את התנאי הראשון, אך לא מוודאים שיש לכל פותח סוגר.
2. סטודנטים אשר בודקים רק האם בסוף המחרוזת מספר הסוגרים והפותחים שווה, בלי להתייחס לדרישה של הסדר שלהם בתוך המחרוזת.
3. סטודנטים אשר עושים מעבר חוזר בעבור כל תת מחרוזת, במקום להשתמש בספירה שראינו עד כה. ריצה זו היא איטית משמעותית מאשר ריצה בודדת. גם ריצה כפולה היא מיותרת, מכיוון שיש בידינו את המידע הרלוונטי.
4. סטודנטים אשר משתמשים בפונקציה substring, אשר ***מייצרת*** מחרוזת חדשה - כלומר משתנה מורכב חדש - בניגוד לתנאי השאלה.

סעיף ב':

```
public static int count (int n) {
     return count(n, 0, 0);
}

public static int count(int n, int open, int close) {
     int result;

     if (open < close | open > n | close > n) {
          result = 0;
     }
     else if (open == close & open == n) {
          result = 1;
     }
     else {
          result = count(n, open + 1, close)
                 + count(n, open, close + 1);
     }
     return result;
}
```

נוסף על תשובה זו, התקבלו שני פתרונות נוספים:

1. ניתן לפתור את הרקורסיה באמצעות מונה יחיד, אך יש צורך להקטין את הערך n בצורה זהירה; למעשה, יש להסתכל על n כמספר הזוגות שנותר לפתוח במחרוזת שנספרת, ולא אורכה.
2. ניתן לספור את האפשרויות למחרוזות מאוזנות באמצעות מספרי קלטן (Catalan), תכונה ***שלא*** נלמדה בקורס, אך קיבלנו תשובות שחישבו ערך זה ישירות.

טעויות נפוצות שראינו:

1. קוד שלא בדק לאורך המחרוזת "שמיוצרת" האם היא תקינה מבחינת סדר הסוגריים, כלומר ספר מחרוזות כגון)))(((שאינן מאוזנות.
2. קוד שספר מחרוזות ללא בדיקה של ספירת הפותחים והסוגרים, כלומר ספר גם מחרוזות מהסוג ()) שאינן מאוזנות.
3. קוד שספר מחרוזות מאוזנות, אך מסדר n/2 במקום מסדר n (כפי שנתבקשתם בשאלה). שגיאה זו הייתה נפוצה בקרב פתרונות שעקבו אחרי אורך "המחרוזת" במקום לעקוב על חריגה של מספר הסוגרים.
4. קוד שמחזיק באופן מפורש את המחרוזת שמיוצגת על ידי הקריאה הרקורסיבית הנוכחית, **בניגוד** לדרישת השאלה. מספיק להחזיק מונים (יחיד או זוג) על מנת לעקוב אחר המחרוזת, ללא התצוגה המפורשת שלה (שכן היא יקרה מאד!)
5. קוד שהשתמש במספרי קטלן, אבל לא חישב נכון את הנוסחה שמגדירה אותם.
6. קוד שבו בעיות ברקורסיה, ובפרט, יצירת רקורסיה אינסופית וחוסר מעבר על כל המקרים האפשריים, מה שיצר תוצאות שגויות או מקרים בהם אין בהכרח ערך החזרה (שגיאת קומפילציה).

שאלה 3 - פתרון לדוגמא וטעויות נפוצות:

סעיף א': פתרון רקורסיבי לדוגמה:

```
public static String decimalAddDigit(String s, int digit) {
    if (s.length() == 0)
        return "" + digit;
    else {
        if ((s.charAt(0) - '0') + digit > 9)
            return ((s.charAt(0) - '0') + digit - 10) +
decimalAddDigit(s.substring(1), 1);
        else return (s.charAt(0) - '0') + digit + s.substring(1);
    }
}
```

סעיף א': פתרון איטראטיבי לדוגמה:

```
public static String decimalAddDigit(String s, int digit) {
    String result = "";
    int carry = digit;
    for(int i = 0; i < s.length(); i++) {
        result = result + (((s.charAt(i)-'0') + carry) % 10);
        carry = ((s.charAt(i)-'0') + carry) / 10;
    }
    if(carry == 1)
        result = result + carry;
    return result;
}
```

סעיף א' + סעיף ב': טעויות נפוצות:

- למרות ההנחייה בשאלה היו נסיונות לחשב את התשובה בצורה מספרית ולהמירה חזרה למחרוזת. בין אם נעזה באופן ישיר על ידי לולאה שנלמדה בהרצאה 8 או על ידי שימוש בפונקציה parseInt ובין אם זה נעשה על ידי לולאה שעובדת על מחרוזות, המשתנה אינו גדול מספיק בכדי לעמוד בחישוב.
- המרה שגויה של תו המייצג ספרה לערך המספרי שלה. שימו לב שישנן מספר דרכים לעשות זאת.
- לחלופין, נסיון שגוי להמיר מספר חד ספרתי למחרוזת (לעיתים על ידי המרות שגויות/הפעלת שיטות).
- חלוקה למקרים רבים אשר סרבלה את הפתרון והיתה לרוב מאוד מיותרת. שימו לב שברוב המקרים הטיפול במקרה הכללי עובד יפה גם למקרי קצה (כגון אינדקס שנמצא בהתחלה או בסוף).
- שגיאות שונות בעת חישוב התוצאה בלולאה: חישוב התוצאה בסדר הפוך/המשך סכימה של digit/ התעלמות מהנשא/חיבור שגוי.. ועוד
- רלוונטי רק לסעיף א': פתרון שמוסיף 1 בדומה לשאלה מעבודת הבית.
- רלוונטי רק לסעיף ב': בלבול הסדר בין פעולת ההכפלה ב-8 ובין הוספת הספרה.

סעיף ב': פתרון רקורסיבי לדוגמה:

```
public static String octal2Decimal(String s) {
    if (s.length() == 1)
        return s;
    return
decimalAddDigit(decimalDouble(decimalDouble(decimalDouble(octal2Decimal(
s.substring(1))))), s.charAt(0) - '0');
}
```

סעיף ב': פתרון איטראטיבי לדוגמה:

```
public static String octal2Decimal(String s) {
    String ans = "0";
    for(int i = s.length()-1; i >= 0; i = i - 1)
        ans =
decimalAddDigit(decimalDouble(decimalDouble(decimalDouble(ans))),
s.charAt(i)-'0');
    return ans;
}
```

סעיף ב': טעויות נפוצות: פירוט בעמוד הקודם.

שאלה 4

פתרון:

```
public BufferAsArray(int capacity){
     elements = new Object[capacity];
     size = 0;
     index = 0;
}
public BufferAsArray(){
     this(DEFAULT_CAPACITY)
}
public void insert(E element){
     elements[index] = element;
     index = (index + 1) % elements.length
     if (size < elements.length)
          size = size + 1;
}
public E remove(){
     if(isEmpty())
          throw new NoSuchElementException();
     int len = elements.length;
     int removeIndex = (len - size + index) % len;
     E removed = (E) elements[removeIndex];
     size = size -1
     return removed;
}
```

פתרון נוסף עבור סעיפים ג',ד'

```
public void insert(E element){
     int insertIdx = (index + size) % elements.length;
     elements[insertIdx] = element;
     if (size < elements.length)
          size = size + 1;
     else
          index = (index + 1) % elements.length;
}
public E remove(){
     if(isEmpty())
          throw new NoSuchElementException();
     E removed = (E) elements[index];
     size = size - 1;
     index = (index + 1) % elements.length;
     return removed;
}
```

הערות לסעיף ב:

- פתרונות שלא השתמשו בבנאי מסעיף א' לא קיבלו ציון מלא.
- יש להשתמש בשדה DEFAULT_CAPACITY ולא בערך 1000 שהוקצה לו.

הערות לסעיפים ג' ו ד':

- קיים פתרון בזמן קבוע ללא שימוש בלולאה.
- פתרונות שהתבצעו בזמן שאינו קבוע לא קיבלו ניקוד מלא.