הפקולטה למדעי הטבע - המחלקה למדעי המחשב ומתמטיקה

סמסטר א', מועד ב', יום שישי כ"ט אדר תשע"ה, Friday, March 20, 2015

מחלקה: **מדעי המחשב ומתמטיקה**
מרצה: **פרופ' ואדים (דוד) לויט, גב' אליזבת איצקוביץ**
שם הקורס: **אלגוריתמים 1**
מס' הקורס: **7022110**
משך הבחינה: **3 שעות**
חל איסור על שימוש בכל חומר עזר.

הנחיות כלליות:

- המבחן ייבדק בצורה אוטומטית ע"י תוכנת מחשב שתשתמש בשמות המוזכרים להלן.
- יש לרשום אלגוריתמים, סיבוכיות והוכחות בקובץ **word** בשם **algorithms.**
- יש לדחוס את קובץ **word** וכל קבצי **java** בקובץ אחד. שם הקובץ – מספר תעודת זהות, סוג הקובץ - **rar** או **zip**.
- שפת תכנות – **java.**

הנחיות לתכנות:

- יש לבנות java project בשם **Exam.**
- לפתרון של כל שאלה צריך לבנות **מחלקה נפרדת**.
- את קובץ הפרויקט ואת השאלון ניתן להוריד מ"מתזמן מבחנים".

המבחן שלא יעמוד בדרשות אלו לא יזכה בנקודות!

20

בעיה מס' 1: בעיית הרוב

איבר הרוב של מערך בגודל ***N*** הוא איבר המערך שמופיע יותר מ- ***N/2*** פעמים.

קלט : מערך בגודל ***N*** של מספרים ממשיים **חיובים**.
פלט : **איבר הרוב** - במידה והוא קיים,
∞ - במידה ואיבר כזה לא קיים.

בטקסט האלגוריתם :
ניתן לשאול שאלה מסוג a[i]==a[j]
ואסור לשאול שאלה מסוג a[i]<a[j].

דוגמה 1:
קלט : **3,3,4,2,4,4,2,4,4**
פלט : **4**
דוגמה 2:
קלט : **3,3,4,2,4,4,2,4**
פלט : ∞-

יש לרשום מחלקה בשם **Question1**. בתוך המחלקה יש לכתוב פונקציה סטטית שמקבלת מערך בגודל ***N*** של מספרים ממשיים חיוביים ומחזירה את איבר הרוב במידה והוא קיים. במידה ואין מספר כזה במערך הפונקציה מחזירה
Double.*NEGATIVE_INFINITY*

public static double majority (double[] arr){…}

אלגוריתם, סיבוכיות, דוגמא והוכחות

בעיה מס' 2:

40

א) יישמו את האלגוריתם:
קלט : $X=x_1x_2\ldots x_n$ ו- $Y=y_1y_2\ldots y_m$ שתי מחרוזות.
פלט: אורכה של תת-מחרוזת משותפת הארוכה ביותר.

דוגמה: קלט: X = "abcbdab", Y = "bdcaba", **פלט:** 4

ב) פָּלִינְדְרוֹם הוא רצף סמלים, שניתן לקרוא משני הכיוונים, משמאל לימין ומימין לשמאל, ללא שינוי בתוצאה. משפט "**WAS IT A CAR OR A CAT I SAW**", למשל, הוא משפט פלינדרומי.

יש ליישם את האלגוריתם המחשב את אורכה של תת-מחרוזת פלינדרומית הארוכה ביותר של המחרוזת הנתונה.

דוגמה 1 :
קלט : "**alfalfa**",
פלט : **5**. (התת-מחרוזת הפלינדרומית הארוכה ביותר היא "**alala**")

דוגמה 2 :
קלט : "**aubcxctybza**",
פלט : **7**. (התת-מחרוזת הפלינדרומית הארוכה ביותר היא "**abcxcba**").

יש לרשום מחלקה בשם **Question2**. בתוך המחלקה יש לכתוב שתי פונקציות סטטיות הבאות :

א) פונקציה סטטית שמקבלת שתי מחרוזות ומחזירה את אורכה של המחרוזת המשותפת הארוכה ביותר :

public static int lcs(String X, String Y) {…}

ב) פונקציה סטטית שמקבלת מחרוזת ומחזירה את אורכה של התת-מחרוזת הפלינדרומית הארוכה ביותר.
public static int lps(String s) {…}

אלגוריתם, סיבוכיות, דוגמא והוכחות

בעיה מס'3:

40

א) יישמו את האלגוריתם:

קלט:
"דף חשבוני" (עם משקלים מונחים על הצלעות) בגודל M על N, וקודקוד עם קואורדינאטות (p,q)

פלט:
"**כן**" אם הקודקוד (p,q) שייך למסלול קצר ביותר (אחד לפחות) בין שני הקודקודים הבאים: $(0,0)$ ו- (N,M).
"**לא**" אם הקודקוד (p,q) לא שייך לאף מסלול קצר ביותר בין שני הקודקודים הבאים: $(0,0)$ ו- (N,M).

ב) יישמו את האלגוריתם:

קלט:
"דף חשבוני" (עם משקלים מונחים על הצלעות) בגודל M על N, וסדרת קודקודים: $(p_1,q_1), \ldots (p_k,q_k)$.

פלט:
"**כן**" אם הקודקודים $(p_1,q_1), \ldots (p_k,q_k)$ שייכים **לאותו** מסלול קצר ביותר (אחד לפחות) בין שני הקודקודים הבאים: $(0,0)$ ו- (N,M).

"**לא**" אם הקודקודים $(p_1,q_1), \ldots (p_k,q_k)$ לא שייכים **לאותו** מסלול קצר ביותר בין שני הקודקודים הבאים: $(0,0)$ ו- (N,M).

יש לרשום מחלקה בשם **Question3**. יש לכתוב בנאי המחלקה שמקבל מטריצה של משקלים כמערך דו-ממדי של קדקודים:

public Question3(Node [][]mat) {...}

מבנה של קודקוד אחד מיוצג ע"י מחלקת **Node**:

```
class Node{
        int x, y, price;
        public Node(int x, int y){
           this.x  =  x;
           this.y  =  y;
           this.price = 0;
        }
}
```

יש לכתוב שתי שיטות:

1) שיטה שמקבלת קואורדינטות של קודקוד ומחזירה **true** אם הקודקוד שייך למסלול קצר ביותר (אחד לפחות) בין שני הקודקודים הבאים: ***(0,0)*** ו- ***(N,M)***.
השיטה מחזירה **false** אם הקודקוד לא שייך לאף מסלול קצר ביותר בין שני הקודקודים הבאים: ***(0,0)*** ו- ***(N,M)***.

public boolean belongs(Node n){...}

2) שיטה שמקבלת מערך של קדקודים ומחזירה **true** אם כל הקודקודים שייכים **לאותו** מסלול קצר ביותר (אחד לפחות) בין שני הקודקודים הבאים: ***(0,0)*** ו- ***(N,M)***.
השיטה מחזירה **false** אם הקודקודים לא שייכים **לאותו** מסלול קצר ביותר בין שני הקודקודים הבאים: ***(0,0)*** ו- ***(N,M)***.

public boolean allNodesBelong(Node[] nodes){...}

אלגוריתם, סיבוכיות, דוגמא והוכחות.

*****

הסבר למבנה של מטריצת הקודקודים

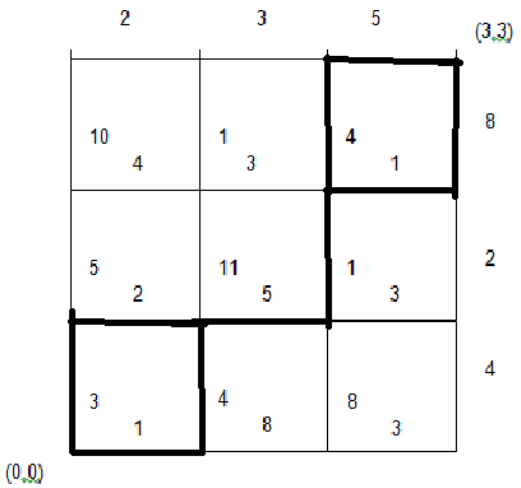
כל קדקוד (**Node**) מכיל את המשקלים של שתי הצלעות היוצאות ממנו לכיוונים ימינה ומעלה:
x - משקל של הצלע האופקי, (הכיוון, כמו בציר ה-**X**, ימינה)
y - עלות של הצלע האנכי, (כיוון כלפי מעלה, כמו בציר ה-**Y**)

<u>דוגמה</u>: במטריצה שלהלן:

קודקוד **(0,0)** נראה כך: **x = 1; y = 3;**

קודקוד **(0,3)** נראה כך: **x = 0; y = 4;**

קודקוד **(3,3)** נראה כך: **x = 0; y = 0;**



בדוגמה זו עלות המסלול הזול ביותר היא 20.
מספר המסלולים הקצרים ביותר הוא 4.
המסלולים הקצרים ביותר מסומנים בקו עבה. קדקוד (1,2) שייך למסלול קצר ביותר.

דוגמה 1: קדקודים
(1,0), (1,1), (2,3)
שייכים לאותו מסלול קצר ביותר.

דוגמה 2: קדקודים
(0,1), (1,0)
לא שייכים לאותו מסלול קצר ביותר, למרות שכל אחד מהם שייך למסלול קצר ביותר.

דוגמה 3: קדקודים
(1,2), (0,1)
לא שייכים לאותו מסלול קצר ביותר בגלל שמקודקוד (1,2) לא ניתן להגיע לקודקוד (0,1).
אולם קודקודים (1,2), (0,1) כן שייכים לאותו מסלול קצר ביותר.

הקוד שמבצע מילוי מטריצה זו:

```
public static Node[][] initMatOfNodes(){ // n = 4
        int n=4;
        Node mat[][] = new Node[n][n];
        mat[0][0] = new Node(1,3);
        mat[0][1] = new Node(8,4);
        mat[0][2] = new Node(3,8);
        mat[0][3] = new Node(0,4);

        mat[1][0] = new Node(2,5);
        mat[1][1] = new Node(5,11);
        mat[1][2] = new Node(3,1);
        mat[1][3] = new Node(0,2);

        mat[2][0] = new Node(4,10);
        mat[2][1] = new Node(3,1);
        mat[2][2] = new Node(1,4);
        mat[2][3] = new Node(0,8);

        mat[3][0] = new Node(2,0);
        mat[3][1] = new Node(3,0);
        mat[3][2] = new Node(5,0);
        mat[3][3] = new Node(0,0);
        return mat;
}
```

בהצלחה!