מבני נתונים – תרגיל 5 – מעשי

תאריך פרסום: 29.5.2016

תאריך הגשה: 19.6.2016

מרצה ומתרגלים אחראים: ד"ר צחי רוזן, ענר בן-אפרים, מוחמד גנאים

נושאי העבודה: כל החומר בקורס עד מיונים לינאריים, כולל

1. מבנה העבודה:

חלקים א + ב:

- תכנון מבנה נתונים יעיל לפתרון שתי בעיות נתונות (ראו בהמשך) באמצעות שימוש במבני הנתונים שנלמדו בקורס.
- מימוש המבנה המתוכנן ב-Java ובדיקתו.
- ניתוח תיאורטי של זמני הריצה של המימוש שהצעתם.

חלק ג (בונוס) (10 נק'):

- בדיקה של מבנה הנתונים והמימוש שהצעתם (בחלקים א' + ב') לבעיה נוספת.

2. מבוא (סיפור):

אתם מתבקשים לתכנן ולבצע שני פרוייקטים:

פרויקט א': הפרויקט התקבל מחברה בשם "נסה ותיווכח". החברה עורכת ניסויים מתמשכים. התוצאות של n ניסויים מתקבלים כקבוצה בת n זוגות סדורים מהצורה (X,Y), כאשר X מציין את מספר הניסוי, מספר שלם לא שלילי בתחום בין 0 ל- n-1, ו- Y את תוצאת הניסוי, מספר שלם לא שלילי כלשהו. החברה מעוניינת שתארגנו את תוצאות הניסויים שלה באופן כזה שהיא תוכל לבצע את הפעולות הבאות בצורה המהירה ביותר:

1. לקבל את מספר הניסוי עם התוצאה החציונית (כלומר עם ערך ה- Y החציוני).
2. לקבל את הממוצע של תוצאות הניסויים בתחום נתון [start, finish], $0 \le start \le finish \le n$ טבעיים.
3. לקבל מערך שמכיל את הניסויים (הזוגות הסדורים (X, Y)) בתחום נתון [start, finish], כאשר $0 \le start \le finish \le n$ טבעיים.
4. להוסיף ניסוי חדש. כל ניסוי חדש מקבל באופן אוטומטי את המספר n הנוכחי, ואחר כך n מתעדכן ל- n+1.

למרבה הצער מנהלי חברת "נסה ותיווכח" לא ארגנו עד עתה את תוצאות הניסויים בשום צורה, ולכן הם מעבירים לכם את n הניסויים הראשונים באמצעות מערך בן n זוגות סדורים מהצורה (X, Y) ללא שום סדר מסוים בין הזוגות, לא לפי ה- X-ים ולא לפי ה- Y-ים.

פרוייקט ב': הפרויקט התקבל מחברה בשם "תחזק ותתחזק". החברה אחראית על תחזוקת התשתיות של שתי שכונות: שכונה דרומית ושכונה צפונית. החברה מקבלת קריאות מדיירי השכונות לגבי תקלות בנקודות ציון מסוימות מהצורה (X, Y), כאשר X הוא הערך של נקודת הציון על הציר מערב – מזרח (ככל שהמספר קטן יותר נקודת הציון יותר מערבית), ו- Y הוא הערך של נקודת הציון על הציר צפון – דרום

(ככל שהמספר קטן יותר נקודת הציון יותר צפונית), ושולחת את עובדיה לתקן את התקלות בנקודות הציון (X, Y) שהיא מקבלת.

לאחרונה הגיעו לחברה שתי תלונות סותרות: ועד השכונה הצפונית התלונן שהשכונה הדרומית מקבלת תיעדוף בטיפול בתקלות של דייריה, ואילו ועד השכונה הדרומית טוען שההיפך נכון – השכונה הצפונית מקבלת תיעדוף בטיפול בתקלות.

עקב כך, מנהלי חברה "תחזק ותתחזק" מעוניינים לבחון שיטת עבודה חדשה. השיטה החדשה קובעת שסדר הטיפול בתקלות יקבע על פי המיקום היחסי של התקלה על גבי הציר צפון - דרום (ציר Y) ביחס לשאר התקלות, כך שהתקלות הראשונות שיזכו לטיפול יהיו אלה שמיקומם היחסי על גבי הציר הוא יותר חציוני (קרוב לחציון).

לשם פשטות מנהלי החברה החליטו שלא להתחשב במיקום התקלה על גבי הציר מערב-מזרח (ציר X), אלא אם יש מספר תקלות שהערך של נקודת הציון שלהן על הציר צפון-דרום (ציר ה- Y) זהה. במקרה כזה, בהתייחס לתקלות הנ"ל בלבד, התקלות יסודרו לפי הערך של נקודת הציון שלהן על הציר מערב-מזרח (ציר ה- X). הם גם החליטו שנקודות הציון תהיינה נתונות תמיד במספרים טבעיים (שלמים לא שליליים).

החברה מעוניינת שתארגנו את הקריאות לתיקון התקלות שהיא מקבלת באופן כזה שהיא תוכל לבצע את הפעולות הבאות במהירות המרבית:

1. מחיקת התקלה שערך המיקום שלה על גבי הציר צפון-דרום (ציר ה- Y) הוא החציון.
2. מציאת k התקלות הקרובות ביותר לחציון על גבי הציר דרום-צפון (הסבר מפורט בהמשך העבודה), כאשר k הוא מספר טבעי (שלם גדול מאפס).
3. הוספת תקלה חדשה באמצעות נקודת הציון שלה (X, Y).
4. קבלת מספר התקלות שנמצאות בטווח נתון [E, W], כלומר מספר התקלות (X, Y) כך ש- $E \le X \le W$.
5. קבלת הממוצע של ערכי ה- Y של מיקום התקלות בטווח נתון [E, W]. כלומר הממוצע של ערך ה- Y של התקלות (X, Y) כך ש- $E \le X \le W$.

למרבה הצער, השיטה בה חברת "תחזק ותתחזק" עבדה עד עכשיו היא שסדר הטיפול בתקלות נקבע דווקא לפי המיקום היחסי שלהם על הציר מערב-מזרח (ציר ה- X) ולכן החברה מעבירה לכם את n הקריאות הנוכחיות שלה במערך של זוגות מהצורה (X, Y) הממוין על פי ציר ה- X.

ולבסוף, עקב זה שאתם סטודנטים ונמצאים תדיר במצוקת זמן, אתם מתבקשים (כלומר, חייבים בלשון מנומסת) לתכנן ולממש מבנה נתונים אחד ויחיד שיתאים לשני הפרויקטים גם יחד, כדי לחסוך מזמנכם היקר.

3. המחלקה Point והסבר על נקודות אמצעיות:

לתרגיל מצורפת מחלקה בשם Point (ראו בהמשך). המחלקה מכילה שלושה שדות: X, Y, ו- name.
אתם צריכים להשתמש במחלקה Point הן כדי לייצג את תוצאות הניסויים של הפרויקט הראשון והן כדי לייצג את הקריאות לתקלות של הפרויקט השני.

הערות חשובות והבהרות:

1. **אין לשנות את המחלקה Point!**
2. בשני הפרויקטים אתם יכולים להניח שלא יהיו שתי נקודות שונות בעלות אותו ערך X, אף על פי **שיתכנו מספר נקודות עם אותו ערך Y**. למשל קבוצת הנקודות הבאה בסדר (7,5), (3,5), (9,5).

3. במימוש מבנה הנתונים תצטרכו להשתמש בשני סוגים של סדר על הנקודות – סדר לפי ציר ה-X וסדר לפי ציר ה- Y. כיוון שייתכנו כמה נקודות עם אותם ערכי Y, הסדר ביניהן יהיה לפי ערך ה-X. לדוגמה, הנקודות של הדוגמה שלמעלה מסודרות ביניהן כך: (9,5) > (7,5) > (3,5).

4. הנקודה החציונית של קבוצת נקודות לפי ציר ה-Y הינה הנקודה שתופיע בתא האמצעי במערך בו הנקודות ממוינות לפי ערך ה-Y, כלומר התא ה $\left\lfloor \frac{n}{2} \right\rfloor$ במערך עם n תאים [0,...,n-1].

דוגמאות:

0	1	2	3	4
(0,0)	(3,1)	(1,2)	(2,3)	(4,4)

0	1	2	3
(1,15)	(4,20)	(3,21)	(19,24)

0	1	2	3	4	5	6	7	8	9
(8,2)	(1,4)	(12,5)	(2,7)	(7,7)	(13,7)	(9,9)	(35,12)	(3,15)	(15,15)

שימו לב שבדגומאות הללו המערך מכיל את הנקודות ממויינות לפי ערך ה-Y. הנקודה בתא המסומן היא הנקודה החציונית.

5. עבור k טבעי (שלם גדול מאפס), k הנקודות האמצעיות הינו טווח התאים בין $\left\lceil \frac{k-1}{2} \right\rceil - \left\lfloor \frac{n}{2} \right\rfloor$ ל-$\left\lfloor \frac{k-1}{2} \right\rfloor + \left\lfloor \frac{n}{2} \right\rfloor$ (כולל) במערך בסעיף הקודם. אם k=1 אז זהו למעשה החציון, כמו בסעיף הקודם.

דוגמאות:

k=2

0	1	2	3	4
(0,0)	(3,1)	(1,2)	(2,3)	(4,4)

0	1	2	3
(1,15)	(4,20)	(3,21)	(19,24)

0	1	2	3	4	5	6	7	8	9
(8,2)	(1,4)	(12,5)	(2,7)	(7,7)	(13,7)	(9,9)	(35,12)	(3,15)	(15,15)

k=3

0	1	2	3	4
(0,0)	(3,1)	(1,2)	(2,3)	(4,4)

0	1	2	3
(1,15)	(4,20)	(3,21)	(19,24)

0	1	2	3	4	5	6	7	8	9
(8,2)	(1,4)	(12,5)	(2,7)	(7,7)	(13,7)	(9,9)	(35,12)	(3,15)	(15,15)

k=4

0	1	2	3	4	5	6	7	8	9
(8,2)	(1,4)	(12,5)	(2,7)	(7,7)	(13,7)	(9,9)	(35,12)	(3,15)	(15,15)

k=5

0	1	2	3	4	5	6	7	8	9
(8,2)	(1,4)	(12,5)	(2,7)	(7,7)	(13,7)	(9,9)	(35,12)	(3,15)	(15,15)

4. פירוט דרישות מבנה הנתונים:

להלן הפעולות שמבנה הנתונים צריך לתמוך בהן, והזמן שלהן במקרה הגרוע:

#		OPERATION	Running Time
1	(constructor)	**PointDataStructure**(Point[] points)	O(n)
2*	void	**addPoint**(Point point)	O(log n)
3	Point[]	**getPointsInRange**(int XLeft, int XRight)	O(k+log n)
4	int	**numOfPointsInRange**(int XLeft, int XRight)	O(log n)
5	double	**averageHeightInRange**(int XLeft, int XRight)	O(log n)
6*	void	**removeMedianPoint**()	O(log n)
7	Point[]	**getMedianPoints**(int k)	O(k log k)
8	Point[]	**getAllPoints**()	O(n)

שימו לב: לצורך המימוש ניתן להניח כי הפעולות addPoint() ו- removeMedianPoint() תזומנה לכל היותר O(log n) פעמים במהלך הריצה. בחלק התיאורטי, תצטרכו להסביר מה השינוי שיש לבצע במימוש שלכם על מנת לאפשר קריאה של O(n) פעמים לפונקציות הללו (תוך שמירה על זמני הריצה הנתונים של **כל** הפונקציות).

לפרויקט מצורפים שני קבצים (ראו בהמשך) שבאמצעותם אתם צריכים להגדיר ולממש את הפעולות הנ"ל. הקובץ הראשון מכיל ממשק בשם PDT המגדיר את הפעולות והקובץ השני מכיל מחלקה ריקה בשם PointDataStructure שמממשת את הממשק. הפתרונות שלכם צריכים להכתב במחלקה PointDataStructure, ואתם רשאים להוסיף מחלקות עזר כרצונכם.

4.1. פירוט הפעולות

בכל אחת מהפעולות הבאות, n מייצג את מספר הנקודות במבנה. עבור הבנאי, n הוא מספר הנקודות במערך הקלט.

1. `PointDataStructure(Point[] points, Point initialYMedianPoint)`

בנאי

פרמטרים: מערך של n נקודות, והנקודה החציונית במערך לפי ערך ה- Y. ניתן להניח שכל הנקודות שונות זו מזו, אך עשויות להיות מספר נקודות עם אותו ערך Y. לא יהיו שתי נקודות שונות עם אותו ערך X. בנוסף ניתן להניח שהמערך מתאים לאחד משני הפרויקטים (אך לא נתון לאיזה), כלומר או שהמערך ממוין לפי ערכי ה- X, מספרים שלמים לא שליליים כלשהם, או שכל ערכי ה- X הם בין 0 ל n-1, אך המערך לא בהכרח ממוין.

שימו לב: מבנה הנתונים צריך לתמוך בכל הפעולות הרשומות בטבלה בשני המקרים הללו. כמו-כן שימו לב שהחציון משתנה כאשר מוסיפים\מסירים נקודות ממבנה הנתונים.

זמן ריצה: O(n)

2. `void addPoint (Point point)`

שיטה המוסיפה נקודה למבנה הנתונים. ניתן להניח שאין במבנה הנתונים נקודה עם ערך ה- X של הנקודה החדשה

פרמטרים: הנקודה להוספה.

זמן ריצה: O(log n)

הערה: אפשר להניח שפעולה זו תזומן לכל היותר O(log n) פעמים במהלך הריצה כולו.

3. `Point[] getPointsInRange(int XLeft, int XRight)`

שיטה המחזירה את הנקודות הנמצאות בתחום נתון על ציר ה- X.

פרמטרים: קצוות התחום הנתון משמאל XLeft (גבול תחתון) ומימין XRight (גבול עליון)

ערך מוחזר: מערך המכיל את כל הנקודות במבנה הנתונים שנמצאות בתחום הנתון.

על השיטה להחזיר גם נקודות עם ערך X ששווה ממש ל- XLeft או ל- XRight

אין חשיבות לסדר הנקודות במערך המוחזר.

זמן ריצה: O(k+log n), כאשר k שווה למספר הנקודות בתחום.

4. `int numOfPointsInRange(int XLeft, int XRight)`

שיטה המחזירה את מספר הנקודות הנמצאות בתחום נתון על ציר ה- X.

פרמטרים: קצוות התחום XLeft (גבול תחתון) ו- XRight (גבול עליון).

ערך מוחזר: מספר הנקודות במבנה הנתונים שנמצאות בתחום.

על השיטה לספור גם נקודות עם ערך X שהוא ממש XLeft או XRight

זמן ריצה: O(log n)

5. `double averageHeightInRange (int XLeft, int XRight)`

שיטה המחזירה את ממוצע ערכי ה- Y של כל הנקודות בתחום נתון על ציר ה- X.

פרמטרים: קצוות התחום הנתון XLeft (גבול תחתון) ו- XRight (גבול עליון).

ערך מוחזר: ממוצע ערכי ה- Y של הנקודות במבנה הנתונים שנמצאות בתחום הנתון.

על השיטה לכלול בממוצע גם נקודות עם ערך X שממש שווה ל- XLeft או ל- XRight.

זמן ריצה: O(log n)

6. `void removeMedianPoint()`

שיטה המסירה את הנקודה בעלת ערך ה-Y החציוני.

שימו לב: אם קיימות כמה נקודות עם אותו ערך Y הן מסודרות ביניהן לפי ערך ה- X.

זמן ריצה: O(log n)

הערה: אפשר להניח ששיטה זו תזומן לכל היותר O(log n) פעמים במהלך הריצה כולו.

7. `Point[] getMedianPoints(int k)`

שיטה המחזירה את k הנקודות הקרובות ביותר לחציון לפי ערך ה- Y, כפי שמוסבר בסעיף 3.

השיטה **אינה מסירה את הנקודות ממבנה הנתונים.**

פרמטרים: מספר טבעי (שלם גדול מ- 0) k

זמן ריצה: O(k log k)

ערך מוחזר: k הנקודות החציוניות לפי ערך ה- Y

אין חשיבות לסדר הנקודות במערך המוחזר

8. `Point[] getAllPoints ()`

שיטה המחזירה את כל הנקודות במבנה.

ערך מוחזר: כל הנקודות במבנה הנתונים

אין חשיבות לסדר הנקודות במערך המוחזר

זמן ריצה: O(n)

5. חלק א' – חלק התרגיל המעשי:

לתרגיל מצורף קובץ ZIP ובו:

- המחלקה Point שאסור לכם לשנות
- הממשק PDT שאסור לכם לשנות
- המחלקה PointDataStructure שעליכם להשלים
- מחלקה בשם Main עם מספר פונקציות לבדיקת נכונות המבנה. הבדיקות אינן מכסות את כל המקרים ומומלץ להוסיף בדיקות משלכם.
- מחלקה בשם GUI לצורך הבונוס (ראו חלק ג')

עליכם להשלים את המחלקה PointDataStructure כך שתענה על כל המתודות הנדרשות, בזמן שצויין.

מותר לכם להוסיף קבצים ומחלקות כרצונכם בהתאם לצרכי המימוש שבחרתם.

אתם מקבלים את קובץ המחלקה PointDataStructure עם מימוש ריק. ניתן להוסיף לה מתודות, בנאים, ושדות כרצונכם. אולם, שימו לב שהבדיקות שלנו יזמנו אך ורק את המתודות והבנאי שצוינו.

6. חלק ב' – הסבר תיאורטי של חלק א':

6.1. תארו בקצרה את המימוש שלכם במסמך מוקלד. הסבירו באילו מבני נתונים השתמשתם ותארו **במילים** את האלגוריתמים שבהם השתמשתם למימוש השיטות של הממשק PDT.

6.2. הסבירו בקצרה את זמן הריצה של **כל אחת** מן הפעולות.

6.3. הסבירו בקצרה (מילים או פסאודו-קוד) כיצד ניתן להשתמש במבנה הנתונים שתכננתם כדי לענות על כל אחת מהשאילתות בכל אחד מהפרוייקים. יש להסביר תוך שימוש במתודות

המצוינות לעיל של מבנה הנתונים וללא התייחסות למימוש שלו (לא להסביר באמצעות המבנה הפנימי או ע"י קריאה לפונקציות פנימיות של PointDataStructure שאינן מצויינות לעיל).

6.4. תארו בקצרה מה השינויים אותם יש לבצע במבנה הנתונים ומימוש המתודות addPoint() ו-removeMedianPoint(), כך שגם אם נזמן כל אחת מהן O(n) פעמים, עדיין נוכל לשמור על זמני הריצה של כל הפעולות של הממשק PDT שמימשתם.

על המסמך להיות בפורמט PDF בלבד.

7. חלק ג' – ממשק משתמש גרפי (GUI):

בחלק זה תוכלו לבחון שימוש אפשרי אחר למבנה הנתונים שבניתם.

בקובץ ה-ZIP תמצאו גם מחלקה בשם GUI, שהרצתה תפתח ממשק משתמש. לאחר מימוש חלק א', ניתן להשתמש בממשק זה לבדיקת הקוד שלכם.

האפליקציה יודעת לטעון תמונה (בפורמט JPG) וקובץ TXT המתאר אילו עצמים מופיעים בתמונה. תוך שימוש בפתרון חלק א' - האפליקציה מאפשרת לכם לסמן בצורה גרפית על גבי התמונה תחום מסוים ולהחזיר את העצמים בתמונה שנמצאים בתחום שסימנתם או את מספר העצמים בתחום.

האפליקציה מאוד פשוטה לתפעול. תחילה יש לטעון את התמונה וקובץ ה TXT שנכתב בפורמט מיוחד (ומתואר למטה) המתאים לתמונה המסוימת. לאחר מכן יש לבחור תחום באמצעות לחיצה על העכבר וגרירה על פני התמונה. לאחר עזיבת העכבר הפלט יופיע למטה.

כמו-כן, בפעולות getMedianPoints ו getPointsInRange יודפסו הנקודות המתאימות במבנה הנתונים על התמונה (שימו לב שבמצב הסטנדרטי, הנקודות אינן מוצגות על התמונה).

בקובץ ה- zip תוכלו למצוא גם שני סטים המכילים תמונה (בפורמט JPG) וקובץ בפורמט TXT המתאר את העצמים בתמונה. אתם יכולים להשתמש גם בהם כדי לבדוק את נכונות מבנה הנתונים.

כמו כן, אתם יכולים להוסיף סטים של קבצים.

הפורמט של קובץ הנקודות מסוג TXT צריך להיות:

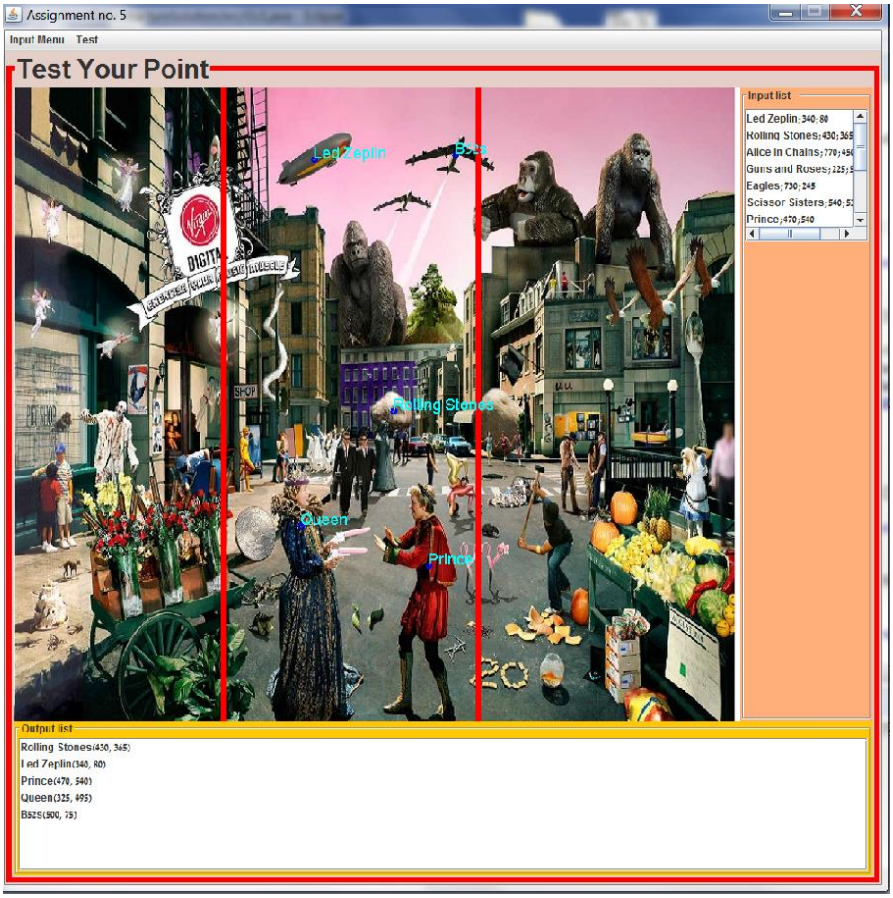
<name>;<x-coordinate>;<y-coordinate>

לדוגמה:

Moshe;30;55

אתם יכולים לצרף לעבודה את הקבצים של הסטים שהשתמשתם בהם. הם אינם צריכים בהכרח להתאים לאחד הפרויקטים. נשקול לתת **בונוס של עד 10 נקודות** לציון העבודה במידה והסטים שתצרפו יהיו מקוריים במיוחד.

צילום מסך לדוגמה:



הערות חשובות ודרישות הגשה:

1. מומלץ מאוד לקרוא את העבודה מהתחלה עד הסוף לפחות פעמיים לפני שתתחילו לפתור אותה. כדאי לתכנן היטב באילו מבני נתונים להשתמש.
2. אין להשתמש במבני נתונים גנריים הקיימים ב - Java במימוש העבודה. כלומר, עליכם לממש כל מבנה שתרצו בעצמכם. (במחלקת הבדיקה Main אתם רשאים להשתמש במבנים מוכנים של Java, כיוון שאתם לא מגישים את המחלקה הזאת.)
3. ניתן תמיד להניח שהקלט יהיה תקין. לא תתבקשו להוסיף נקודה שכבר קיימת. לא תתבקשו למחוק נקודה ממבנה ריק. לא תקבלו מתכנית הבדיקה שלנו ערך null כפרמטר לאף אחת מהשיטות של הממשק.
4. בקובץ ה- zip יש גם קובץ בשם Main.java שיכול לשמש אתכם לבדיקה ראשונית של העבודה שלכם. לאחר שתסיימו את התכנית אתם יכולים להריץ את ה- main כדי לדעת אם התכנית עובדת כמו שצריך. שימו לב: **הבדיקות אינן מכסות את כל המקרים ומומלץ להוסיף עוד בדיקות משלכם**.
5. את העבודה יש להגיש ב- Submission system.
6. **חשוב: כל הקבצים של העבודה שאתם מגישים צריכים להיות ב- default package** (זה שנוצר אוטומטית ביצירת פרויקט חדש בeclipse). אין ליצור package חדש. package חדש ימנע העלאת הקבצים ל- submission system.
7. עליכם להגיש קובץ מסוג zip בשם assignment5.zip המכיל בתוכו:
   a. תיקיית src ובה קבצי הג'אווה של העבודה, ללא המחלקות GUI, Point ו- Main.
   b. מסמך PDF המתאר בקצרה את הפתרון ומסביר בקצרה את זמני הריצה, וכמו-כן הסבר איך ניתן להשתמש במבנה הנתונים כדי לענות על דרישות 2 הפרוייקטים ומה צריך לשנות על-מנת לאפשר עד O(n) הוספות ומחיקות (הפתרון של חלק ב')
8. סביבת העבודה בה תיבדקנה העבודות הינה JavaSE-1.7

9. עליכם לדאוג כי עבודותיכם יתקמפלו וירוצו בסביבת eclipse תחת גרסאות Java הנזכרות לעיל.
10. עבודות שלא יתקמפלו – יקבלו ציון 0.
11. עבודותיכם יבדקו באמצעות כלי בדיקה אוטומטים הבודקים קורלציה בין עבודות. אין להעתיק! להזכירכם, המחלקה רואה בחומרה רבה העתקות.
12. נרצה לראות קוד מתועד, מתוכנן היטב ויעיל שמייצג הבנה.

בהצלחה!