**בס"ד**

**תִיכּוּן ותִכנוּת מונחה עצמים להנדסה**

**מספר קורס: 157109**

**תרגיל בית מספר 5**

**נושא: תכנות גנרי ו Java Streams (מצגות הרצאה 9-10)**

תזכורת: בתרגיל זה תתבצע הגנה פרונטלית בשעות המעבדה.

# מבוא: ניתוח אשכולות

מתוך ויקיפידה:

***בכריית מידע, סטטיסטיקה ולמידה חישובית, ניתוח אשכולות (באנגלית: Cluster Analysis) מתייחס למשימה של קיבוץ אובייקטים לקבוצות (אשכולות) כך שהאובייקטים הנמצאים באותה קבוצה דומים זה לזה יותר מאשר לאובייקטים השייכים לקבוצות אחרות.*** *פעולה זו היא מטלה עיקרית בכריית נתונים וטכניקה נפוצה בניתוח נתונים סטטיסטיים, המשמשת בתחומים רבים לרבות לימוד מכונה, זיהוי דפוסים, ניתוח תמונות, אחזור מידע, ביואינפורמטיקה, דחיסת נתונים וגרפיקה ממוחשבת.*

ישנם סוגים רבים של אלגוריתמי ניתוח אשכולות, ובתרגיל זה תממשו אלגוריתם ניתוח אשכולות היררכי (Agglomerative). האלגוריתם מקבל קבוצה של איברים שניתן למדוד ביניהם מרחק ופרמטר סף המרחק שממנו והלאה אשכולות נחשבים כנפרדים. הפסאודו-קוד של האלגוריתם מופיע להלן:

Cluster(*items*, *threshold*)

*clusters* = A set of singletons, where each item starts in its own cluster

while (*clusters* is not of size 1)

find the two closest clusters, *c1*,*c2* from *clusters*

if (distance(*c1*, *c2*) > *threshold*) return *clusters*;

replace *c1*,*c2* in *clusters* with the union of c1,c2

return *clusters*

המרחק בין שני האשכולות c1, c2 מוגדר כ:

כלומר, המרחק בין שני הפריטים הקרובים ביותר, הראשון מ-c1 והשני מ-c2.

למען הבהירות, דרישות התרגיל מתוארות בשני חלקים המהווים חלק בלתי נפרד ממנו. **שני החלקים מתייחסים לאותו פרוייקט ועליכם לממש אותו בצורה שתעמוד בשני החלקים.**

# דרישה א': בניית תוכנה גנרית

נרצה לממש את האלגוריתם באופן גנרי, כך שיתמוך בפריטים המממשים את הממשק הבא:

**public interface** Clusterable <T>{

**public double** distance(T other);

}

בפרט נשתמש באלגוריתם על מנת למצוא אשכולות של פריטים מהסוגים הבאים:

1. נקודות דו-מימדיות במישור
2. מחרוזת בעלת n ביטים

**עבור נקודות דו-מימדיות במישור-**

נשתמש במחלקה

**public** **class** TwoDPoint **…** {

**double** x;

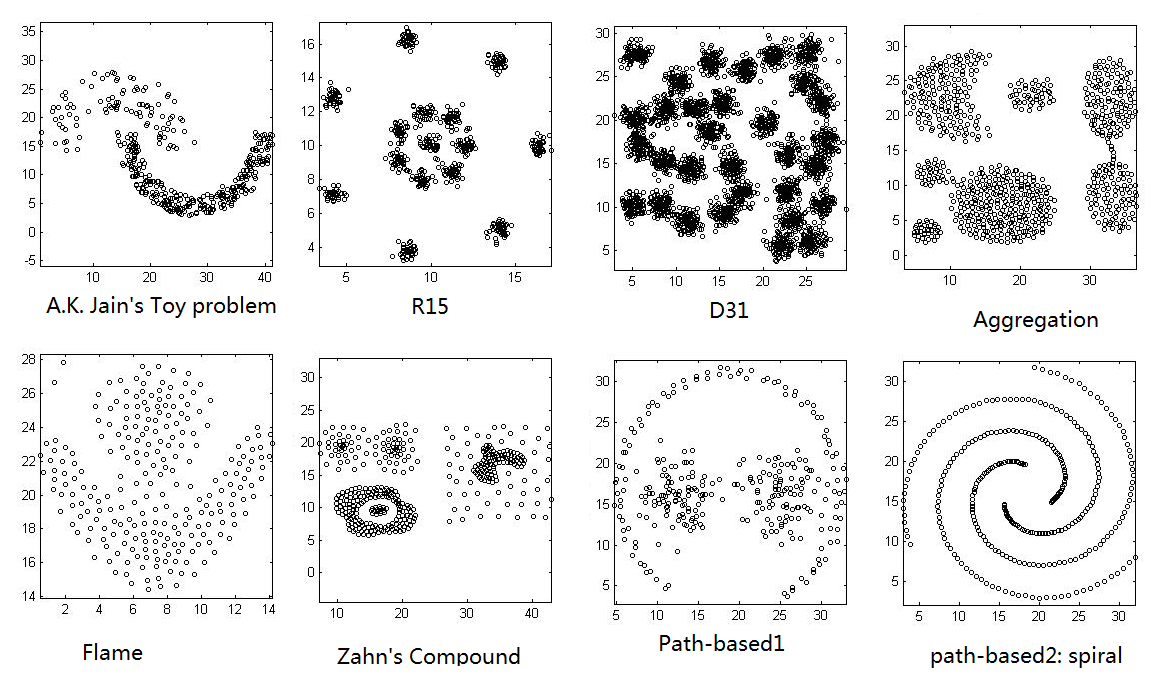
**double** y;

}

ונשתמש [במרחק האוקלידי](https://en.wikipedia.org/wiki/Euclidean_distance) לשם הגדרת המרחק.

עבור הבדיקות, תוכלו להשתמש [באוסף הנתונים הספירלי](http://cs.joensuu.fi/sipu/datasets/spiral.txt) (המוצג להלן).

מצאו את הסף המתאים עבור הפרדת אוספי הנתונים ל-3 ספירלות.



**ועבור מחרוזת בעלת n ביטים –**

נשתמש במחלקה

**public** **class** BitArray **… {**

**boolean**[] bits;

…

}

ונשתמש ב [hamming distance](https://en.wikipedia.org/wiki/Hamming_distance) לשם הגדרת המרחק.

עצה: על מנת להמנע מבאגים, כאשר אתם מעדכנים איבר שנמצא ב Set (לדוגמא באיחוד אשכולות), ראשית הסירו את האיבר מה Set, עדכנו אותו ואז החזירו אותו ל Set. התנהלות זו תמנע מצב של חוסר תאימות בין תוכן האיבר למפתח ה hash שלו.

# דרישה ב': שימוש ב Streams

להלן הדרישות למערכת ניתוח אשכולות גנרית באמצעות Java streams.

**דרישה חשובה: השימוש בלולאות ובפונקציית forEach של Stream אסור במהלך פרויקט זה. יש לממש כל פעולה החוזרת על עצמה באמצעות streams בלבד.**

**לולאת ה-while באלגוריתם היא היוצא מן הכלל היחיד בדרישה האוסרת על לולאות.**

עצה: אם אתם מתקשים בכתיבת קוד ללא לולאות, התחילו בכתיבה ואימות של הקוד שלכם עם לולאות, ולאחר מכן "תרגמו" כל חלק מהקוד בו יש לולאות לקוד המבוסס על Streams. תוכלו למצוא [כאן](https://stackoverflow.com/questions/40678892/replace-nested-loop-with-java-8-flatmap) דוגמא הממירה לולאה כפולה לביטוי ב Java 8 Streams.

עצה 2: מומלץ לפרק פונקציות מורכבות לפונקציות משנה. בפרט, מומלץ להגדיר במחלקה AgglomerativeClustering פונקציות עזר למציאת המרחק בין אשכולות ופונקציה המחזירה את זוג האשכולות הקרובים ביותר. על מנת לייצג זוג סדור, מומלץ להשתמש במחלקה AbstractMap.SimpleEntry.

מסופק לכם שלד של הקוד ודוגמאות לאוסף נקודות ומחרוזות ביטים והאשכולות שלהם. עליכם להשלים את המחלקות AgglomerativeClustering, BitArray, TwoDPoint ולהגישן דרך רכיב ה VPL במודל, על פי הנהלים.

**המערכת המסופקת**

בהצלחה!