הנדסת נתונים ומידע - תרגיל בית 1
תאריך הגשה 27.04.17

1. מטרות התרגיל

לתרגיל מספר מטרות.
במישור של הנדסת תוכנה:
1. תרגול עבודה עם מחלקות, בפרט: בנאים, מתודות והרשאות גישה, העמסת אופרטורים ומשתנים ומתודות סטטיים.
2. העברה נכונה של פרמטרים לפונקציות.
3. תרגול עבודה עם וקטורים ואיטרטורים.

במישור של הנדסת נתונים:
1. הבנה לעומק של אלגוריתמים K-Means ו K-Medoids.
2. תרגול המושגים שנלמדו בהרצאות.

הסבר לחלק הרטוב

עם התרגיל קיבלתם מספר קבצי קוד (חלקם ריקים – לצורך השלמה שלכם).
מותר לשנות את כל המחלקות **פרט למחלקה Tests**.
המחלקות דומות מאוד לאלה שקיבלתם במעבדה.
על מנת ליצור פרויקט חדש עם קבצים קיימים יש לבצע (במדויק) את הפעולות הבאות:
a) צרו תיקיה חדשה
b) העתיקו אליה את קבצי הקוד בלבד
c) פיתחו CLion ובחרו: **Import project from sources** ובחרו את התיקיה עם הקבצים
d) חכו קצת עד שנוצר לכם פרויקט חדש

שימו לב שהפרמטרים שהתוכנית מקבלת השתנו ביחס לקוד המעבדה.
זוהי שורת הפקודה שאיתה מריצים את התוכנית:
./runfile K max_iterations input_file test_name
runfile הוא קובץ ההרצה של הפרויקט שנמצא בתוך התיקיה הפנימית של הפרויקט
K הוא מספר הקלאסטרים.
max_iterations היא כמות מקסימלית של איטרציות.
input_file הוא קובץ הקלט. שימו לב לשים פה את ה path המלא של הקובץ.
test_name הוא שם הבדיקה שאנחנו רוצים להריץ. מקבל את אחד הערכים הבאים:
*{**testKMeans**, **testKMedoids**, **testKMeansSSE**, **testKMedoidsSSE**}*

לדוגמא, הפקודה הבאה מריצה את KMeans עם 2 קלאסטרים ומקסימום 5 איטרציות:
./runfile 2 5 /home/annabel/samples/in1 testKMeans
עם הכוונה לקובץ פלט (אפשר לעשות רק דרך הטרמינל):
/runfile 2 5 /home/annabel/samples/in1 testKMeans > /home/annabel/samples/out1

2. <u>חלק רטוב (50%)</u>

לאחר שיצרתם פרויקט חדש בצעו את הסעיפים הבאים:

1. עליכם תחילה להשלים את המימושים החסרים במחלקות Cluster ו Point (8%)

2. הפרידו את הדפסת תוצאת הקלאסטרינג למתודה נפרדת במחלקה Kmeans (כעת ההדפסה היא חלק מפונקציית run) .קריאה לפונקציה זו נמצאת בבדיקות ב Tests.cpp. (2%)

כעת על מנת לבדוק את האלגוריתם, הריצו את הבדיקה ***testKMeans*** עם הקונפיגורציות הבאות וראו שאתם מקבלים פלטים כמפורט בטבלה:

פלט	K	איטרציות	קלט
in1_out1	2	10	in1
in1_out2	1	10	in1
in1_out3	7	10	in1
in1_out4	3	10	in1
in2_out1	3	3	in2

<u>שימו לב 1:</u> כדי שהקוד יתקמפל בשלב זה יש לסגור בהערה את **תוכן** הפונקציות האחרות במחלקת Tests.

<u>שימו לב 2:</u> עליכם לדאוג לכך שהפלט שלנו זהה לפלט שלכם **<u>בדיוק</u>**. על מנת לעשות זאת אפשר להשתמש בפקודת bash הבאה:

```
diff out1 out2
```

כאשר out1 ו out2 שניהם קבצי פלט (אחד שלנו ואחד שלכם).

<u>שימו לב 3:</u>

על מנת לאפשר לכם את יצירת הפלטים בקלות, הריצו את התוכנה דרך הטרמינל עם הכוונת פלט לקבצים שלכם, כאשר ה main מכיל את הקריאה לtest המתאים.

3. עליכם לממש אופרטור >> במחלקות Cluster ו Point שיחליפו (ע"י הדפסה זהה) את פונקציות ה print בשתי המחלקות (8%)
   כעת חיזרו על הבדיקה מסעיף 1 וראו שאתם מקבלים עדיין פלטים נכונים.

4. ממשו בתוך מחלקת Cluster את המתודה הבאה שמטרתה להחזיר איטרטור למקום בוקטור הנקודות שמכיל את הנקודה הנתונה:

```
std::vector<double>::iterator find(size_t pointID)
```

   אם הנקודה אינה כלולה בוקטור- יוחזר איטרטור לסוף הוקטור.
   חשבו מהי הרשאת הגישה שצריכה להיות למתודה זו (מהו העקרון ההנדסי שלמדנו לפיו החלטתם?) שנו את הפונקציה removePoint כך שתשתמש במתודה הנ"ל. (5%)
   בדקו שהאלגוריתם עדיין עובד כמצופה.

5. ממשו את הפונקציה המחשבת מרחק אוקלידי בין 2 נקודות כפונקציה סטטית במחלקה Point.
   החליפו את הקריאות לפונקציה המקורית – בקריאות לפונקציה החדשה. (5%)

6. ממשו מחלקת KMedoids הממשת את האלגוריתם K-Medoids שבמקום centroid משתמש ב medoid וזהו ההבדל היחיד בינו לבין K-Means. הממשקים של שתי המחלקות צריכות להיות זהים. medoid היא נקודה שהיא חלק מהמדגם שאותו אנו מקלאסטרים, אשר קרובה ביותר לכל

הנקודות האחרות בקלאסטר. זה בניגוד לנקודת ממוצע שבה משתמש אלגוריתם ה K-Means, שהיא ברוב המקרים אינה תצפית אמיתית. (15%)
שימו לב לפונקציות שצריכות להתווסף למחלקת Cluster כדי להפעיל את K-Medoids.
<u>הערה</u>: בין 2 המחלקות K-Means ו K-Medoids תהיה חפיפה רבה, עוד מעט נלמד מנגנון שמאפשר להימנע משכפול קוד זה, אך כרגע אין לכם כלים להימנע ממנו.
על מנת לבדוק את האלגוריתם, הריצו את הפונקציה ***testKMedoids*** עם הקונפיגורציות הבאות וראו שאתם מקבלים פלטים כמפורט בטבלה:

פלט	K	איטרציות	קלט
in1_out5	2	2	in1
in1_out6	3	5	in1
in1_out7	7	10	in1
in2_out2	3	2	in2

7. ממשו מתודה SSE במחלקת Cluster שמטרתה לחשב את סכום ריבועי המרחקים מה prototype של הקלאסטר. (5%)

8. ממשו מתודה SSE במחלקות K-Means ו K-Medoids שמממשת את סכום כל ה SSE על כל הקלאסטרים. בדקו את המתודה באמצעות הרצת הבדיקות: ***testKMedoidsSSE***, ***testKMeansSSE*** (2%)

3. <u>הרצות וניתוח תוצאות (15%)</u>
את ההרצות הבאות יש לבצע על הקובץ נתונים של הצבעים.
ניתן לבצע את המשימות הבאות ע"י כתיבת סקריפט bash המריץ את ה test המתאים תוך שינוי הפרמטרים של הקלט. מטרת האלגוריתמים הנ"ל היא חלוקה של נקודות לקלאסטרים כך שסכום הטעות הריבועית SSE יהיה מינימלי.

9. **בדיקת ההשפעה של כמות האיטרציות**. הריצו את שני האלגוריתמים כמות עולה של איטרציות מ 1 עד 50. שרטטו גרף של SSE כפונקציה של כמות איטרציות (ציר x מספר איטרציות וציר y SSE) הסבירו את המגמות בגרף. התייחסו להבדלים בין K-Means ו K-Medoids ולכמות האיטרציות שבה האלגוריתם מתכנס. לאורך הניסוי קיבעו את מספר הקלאסטרים ל 20.
10. **בדיקת ההשפעה של כמות הקלאסטרים**. הריצו את שני האלגוריתמים עבור כמות עולה של קלאסטרים מ 2 עד 50. שרטטו גרף של SSE בפונקציה של כמות קלאסטרים (ציר x מספר קלאסטרים וציר y SSE). הסבירו את התוצאות. לאורך הניסוי קיבעו את מספר האיטרציות ל 30.
11. לפני ביצוע הסעיף הנוכחי שנו את ה seed בתוך מחלקה SeedGenerator להיות תלוי זמן ההרצה (השורה מופיעה כרגע בהערה בקוד). הריצו את שני האלגוריתמים 50 הרצות עבור 20 קלאסטרים ו30 איטרציות. דווחו מהו ה SSE האופטימלי בשני במקרים (עבור KMeans ו KMedoids).

4. <u>קבצי נתונים</u>
לרשותכם 3 קבצי קלט:
1. הקובץ **in1.txt** ו **in2.txt** שהם קבצים קטנים עם דוגמאות פשוטות מאוד לבדיקות התחלתיות.
2. הקובץ **color_dataset_ready.txt** שהוא קובץ שנוצר כתוצאה מתיוג אנושי של צבעים. כל שורה מכילה את שם הצבע כפי המשתמש החליט לקרוא לו מהסתכלות עליו ואת ערכי ה RGB של הצבע.
כאשר הקובץ עובר קלאסטרינג לפי וקטורי ה RGB, אנו מקבלים שבכל קלאסטר יש צבעים דומים.
למעשה זה יכול לשמש אותנו ל disambiguation (באור המשמעות) של צבעים על פי שמם בטקסט.
לדוגמא נקבל קלאסטר של כל הגוונים הירוקים ונדע שכל השמות בו מתארים את הצבע ירוק.

כמו כן ישנם כמה קבצי פלט לדוגמה שמוזכרים בחלק הרטוב.

5. <u>חלק יבש – הנדסת תוכנה (15%)</u>

1. הסבירו מדוע המתודה find שמימשתם בסעיף 3 בחלק הרטוב, אינה יכולה להיות קבועה.
2. הסבירו את ההיגיון מאחורי הפיכת המתודה שמחשבת מרחק אוקלידי – למתודה סטטית.
3. הסבירו מדוע אי-אתחול של חלק ממשתני המחלקה Kmeans בבנאי היא תקינה. הסבירו מה קורה בעת בניית האובייקט Kmeans.
4. הסבירו מדוע נכון יותר היה להוציא את הדפסת הקלאסטרים לפונקציה נפרדת.

6. <u>חלק יבש – הנדסת נתונים (20%)</u>

1. הוכח או הפרך: סכום הסטיות מהממוצע שווה לאפס.
2. הוכח או הפרך: החציון תמיד קטן או שווה לממוצע.
3. תנו דוגמא בממד אחד שבה תוצאת הרצת K-Means עם נקודות אתחול שונות גורמת ליצירת קלאסטרים שונים, אך ה SSE הכולל זהה.
4. הסבירו מדוע תנאי העצירה הבאים של K-Means זהים:
   a. ההשמות של נקודות לקלאסטרים אינן משתנות.
   b. נקודות המרכז של כל קלאסטר אינן משתנות.
5. הסבירו על איזה סוג של נתונים לא ניתן להריץ KMeans , אך ניתן להריץ KMedoids. תן דוגמה.
6. בהינתן ערך RGB של צבע כלשהו ששמו אינו ידוע (תצפית חדשה Y) הציע חיים (סטודנט מצטיין להנדסת נתונים) את השיטות הבאות למציאת השם של הצבע בעזרת קובץ הצבעים שברשותכם:
   a. לחפש את הנקודה הקרובה ביותר לפי ערכי ה RGB בקובץ הנתון לכם ולהחזיר את שמה.
   b. לקלאסטר את הנתונים ל 3 קלאסטרים ע"י KMedoids ואז למצוא את ה prototype הקרוב ביותר לצבע הנתון ולהחזיר את שמו
   c. לקלאסטר את הנתונים ל50 קלאסטרים ע"י KMedoids ואז למצוא את ה prototype הקרוב ביותר לצבע הנתון ולהחזיר את שמו

חיים הריץ עבור תצפית Y עם ערכי RGB=(35,200,12) את שלושת השיטות וקיבל את התוצאות הבאות:

a. Light-Green (38, 204, 8)
b. נעשו 50 הרצות (בשל הגרלת נקודות ההתחלה) והתקבלו בסה"כ 22 שמות שונים של גווני ירוק
c. נעשו 50 הרצות (בשל הגרלת נקודות ההתחלה) והתקבלו הסה"כ 14 שמות שונים של גווני ירוק

הסבירו ממה נובע ההבדל בכמות השמות שהתקבלו ומה ההמלצה שלכם (איזו שיטה עדיפה) בהינתן ערך RGB אחר.

7. **<u>תיעוד הקוד</u>**

מעל כל פונקציה בקוד יש לציין:

1. מה המטרה של הפונקציה
2. מהם המשתני הקלט (שאותם הפונקציה חייבת לקבל כדי לבצע את מטרתה)
3. מהם משתני הפלט (שהם התוצאה של הריצה של הפונקציה)

דוגמה לפורמט של התיעוד תתפרסם בהמשך במודל. יש לתעד את הקוד שקיבלתם **וגם** את זה שכתבתם בעצמכם.

8. <u>דגשים למימוש</u>

הקפידו על:

1. כתיבת קוד בהתאם למוסכמות שפורסמו באתר.

2. העברה נכונה של פרמטרים לפונקציות.
3. שימוש נכון ב const.
4. איתחול משתנים.
5. תיעוד בהתאם לדרישות.
6. הרשאות נכונות למתודות ומשתנים במחלקות.
7. חלוקה של הקוד לפונקציות.
8. מתן שמות משמעותיים למשתנים ופונקציות.

9. <u>אופן בדיקת התרגיל</u>

התרגיל יבדק בדיקה יבשה ורטובה. חלק מהקלטים והפלטים של התרגיל אינם נתונים לכם (כלומר יש לדאוג לנכונות האלגוריתמים מעבר למה שבדקתם כאן). **תרגיל שאינו מתקמפל עם הקומפיילר של CLion ב Linux – חלקו הרטוב לא יבדק ולא יבדקו התשובות לשאלות היבשות שהן תוצאה של הרצה של הקוד.**

יש לוודא את פורמט הפלטים ע"י diff כמתבקש למעלה.

10. <u>הגשת התרגיל</u>

- לצורך התרגיל יפתח פורום במודל. **מתן תשובות לשאלות בנוגע לתרגיל יתבצע דרך הפורום בלבד.**
- התרגיל להגשה בזוגות או ביחידים (ולא בשום הרכב אחר)
- לפני ההגשה, חובה לקמפל ולבדוק את התרגיל במעבדת הוראה ולא בסביבה אחרת .
- ההגשה חייבת להכיל קובץ **ZIP** יחיד בלבד (ולא קובץ RAR וכדומה) המכיל: כל קבצי קוד המקור, ללא קבצי הרצה. בקובץ זה יש לכלול מסמך word או pdf המכיל את התשובות לחלק היבש. יש לציין ברור את מספר השאלה עליה ניתנת התשובה. יש להדביק את הגרפים שביקשתי בתוך קובץ התשובות ולא להגישם בקבצים נפרדים.
- שם הקובץ חייב להיות , **hw1_yyyyyyyyyy_xxxxxxxxxx.zip** כאשר xxxxxxxxxx ו - yyyyyyyyyy הם מספרים תעודות הזהות של המגישים, כולל ספרת ביקורת .
- ההגשה היא אלקטרונית בלבד, דרך אתר ה moodle-של הקורס. תרגילים שיוגשו בכל דרך אחרת לא ייבדקו.
- אין להגיש את אותו הקובץ פעמיים. התרגיל יוגש על ידי אחד מבני הזוג.
- שימו לב שההגשה תיחסם בדיוק בשעה 23:55 ביום ההגשה. מומלץ להגיש לפחות שעה לפני המועד האחרון.
- ניתן להגיש כמה פעמים. רק ההגשה האחרונה תישמר.
- תרגיל בית שלא יוגש על פי הוראות ההגשה - לא ייבדק (כלומר יקבל ציון אפס).

<u>בהצלחה!</u>