**מטלה 1 - גרפים אקראי במודל ארדש – ריניה (model Renyi –Erdös)**

**אלגוריתמים 2 – יוליה קמפנר**

**מגישים:**

מאי נתן– 318228350

אמתי ביטון – 316363357

**מימוש המטלה:** [**hit\_hw1\_alg\_rg.pdf**](hit_hw1_alg_rg.pdf)

**מחלקה Graph:**

במחלקה זו מימשנו את בניית הגרף תוך שימוש במבנה נתונים של מערך רשימות מקושרות.

לצורך בניית הגרף בנינו פונקציה שמקבלת צומת מקור וצומת יעד ומחברת ביניהם קשת.

כמו כן, מימשנו את האלגוריתם BFS – בו נשתמש לחישובים בהמשך.

*BFS()* – האלגוריתם עובר על מסלול של צמתים מצומת התחלתית ומוצא את המרחק בין אותה צומת לשאר הצמתים בגרף. את הרשימה של המרחקים הוא שומר במערך. במידה ויש יותר מרכיב קשירות אחד הצמתים שבהם הוא לא עבר יסומנו ב- -1. זמן ריצה של האלגוריתם O(E+V).

**Main:**

לפי הנתון המבוקש, יש להגריל 500 גרפים עם 1000 צמתים בהסתברויות שנבחרו מראש לפי 3 תכונות:

1. קשירות – Threshold1 = : אם p<Threshold1 אז הגרף לא קשיר בהסתברות גבוהה.

אם p>Threshold1 אז הגרף קשיר בהסתברות גבוהה.

1. קוטר הגרף – Threshold2 = : אם P>Threshold2 אזי בהסתברות גבוהה קוטר הגרף שווה ל2, אחרת קוטר הגרף גדול מ2.
2. צומת מבודד בגרף (ללא שכנים) – Threshold3 = : אם P<Threshold3 אזי בהסתברות גבוהה קיים גרף בצומת מבודד.

אם P>Threshold3 אזי בהסתברות גבוהה לא קיים בגרף צומת מבודד.

את הגרפים הגרלנו בצורה אקראית בעזרת פונקציה *build\_random\_graph():* הפונקציה קוראת לפונקציה *random\_bool\_with\_prob()*  – אשר מגרילה true/false לגבי קיום צמתים בגרף בעזרת ההסתברות הנתונה.

הצמתים נוספים בעזרת הפונקציה *addEdge(*) שממומשת במחלקה Graph.

רשימת הסתברויות שבחרנו לכל תכונה:

Threshold1 = 0.00690776

P1 = [0.002,0.0022,0.0027,0.003,0.0035,0.007,0.0072,0.0075,0.0077,0.0079]

Threshold2 = 0.117539

P2 = [0.02,0.03,0.033,0.038,0.04,0.13,0.15,0.18,0.2,0.22]

Threshold3 = 0.00690776

P3 = [0.002,0.0022,0.0027,0.003,0.0035,0.007,0.0072,0.0075,0.0077,0.0079]

לצורך חישוב הקוטר נשתמש בפונקציה *diameter()* :

הפונקציה מקבלת גרף, ומחשבת את הקוטר תוך שימוש באלגוריתם BFS (שמימשנו במחלקה Graph), במידה והגרף לא קשירת הפונקציה מחזירה -1 = כלומר הקוטר שווה ל-.

הפונקציה מריצה פעמיים BFS ובכל פעם מוצאת קודקוד אחר של הקוטר – פעם ראשונה מצומת התחלתית שבחרנו מראש (לפי הוכחה שנלמדה באלגוריתמים 1, הצומת הרחוק ביותר מן הקודקוד ממנו ביצענו את הסריקה הוא בהכרח קודקוד בקוטר), כמו כן בשלב זה הפונקציה בודקת האם הגרף קשיר. בסיום האלגוריתם ובמידה ומצאנו שהגרף קשיר, הפונקציה מבצעת שוב BFS, הפעם מהצומת הרחוקה ביותר שמצאנו מהצומת ההתחלתית, ומוצאת את הקודקוד השני של הקוטר.

O(V+E) – פעמיים BFS

לצורך בדיקת הקשירות נשתמש בפונקציה *connectivity()* :

הפונקציה מקבלת גרף, משתמשת באלגוריתם BFS לחישוב המרחק, במידה וקיים רכיב קשירות נוסף המרחק אליו יהיה -1 (כלומר ). במידה ופונקציה מצאה מרחק כזה היא תחזיר 0 (הגרף לא קשיר), אחרת היא תחזיר 1 (הגרף קשיר).

O(V+E) - BFS

לצורך בדיקה האם קיים צומת מבודד נשתמש בפונקציה *Is\_Isolated() :*

הפונקציה מקבלת גרף, עוברת על הצמתים וקוראת לפונקציה *getAdjList*() (אותה מימשנו במחלקה Graph). פונקציה זו מחזירה את רשימת השכנים של אותו צומת.

במידה והרשימה ריקה הפונקציה תחזיר 1 (קיים צומת מבודד), אחרת 0 (לא קיים צומת מבודד).

O(V) – מעבר על כל הצמתים

פונקציות עזר נוספות:

*double threshold1and3()*

*double threshold2() :*

חישוב הThreshold של כל תכונה (לפי הנתונים) – הפונקציה מקבלת גרף ומחזירה את תוצאת החישוב.

*int firstHabitApplies()*

*int secondHabitApplies()*

*int thirdHabitApplies() :*

בדיקת האם התכונות מתקיימות (לפי הנתונים) – הפונקציה מקבלת גרף והסתברות, ומחזירה 0 או 1 בהתאם לקיום התנאי של אותה פונקציה.

הראשונה – קשירות

השנייה – קוטר

השלישית – צומת מבודד

*void firstHabbit()*

*void secondHabbit()*

*void thirdHabbit() :*

במידה והתכונה מתקיימת, הפונקציה מוסיפה 1 לספירה ובסיום ההרצה קוראת לפונקציה היצוא לקבצי CSV.

*void exportResultToCsv() :* ייצוא התוצאות לקובץ CSV.

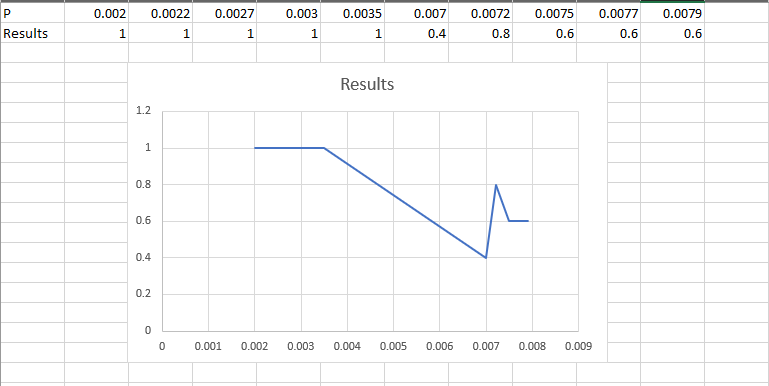
<habit1Results.csv>

<habit2Results.csv>

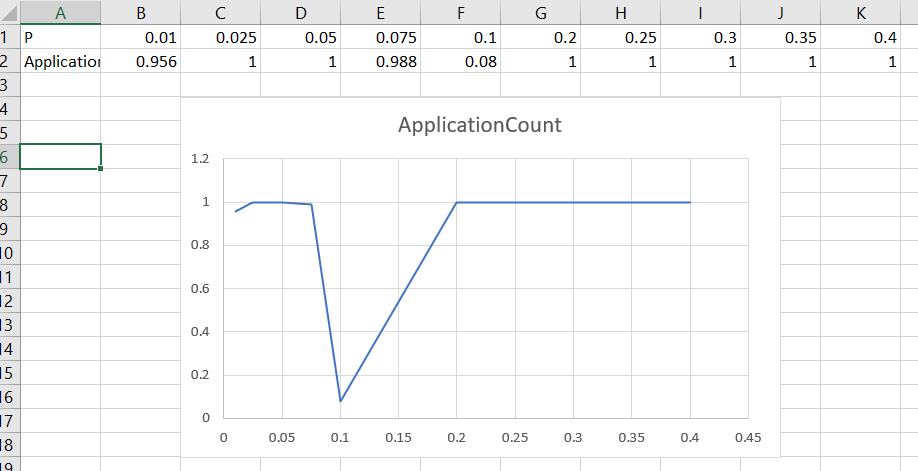
<habit3Results.csv>

תוצאות: <Results.csv>

תכונה 1:



תכונה 2:



תכונה 3:

