Read me

מגישים: עומר כהן 208521294 , מתן זרחי 208389999

איזה ייצוג בחרתי לגרפים ולמה: בחרתי adjacency list מפני שלעבור על גרף ברשימה ייקח e+v לעומת מטריצה שזה v^2 ובחלק של הסימולציה שזה מגיע למספרים גדולים יחסית זה כבר נהיה הבדל משמעותי. כמו כן עבור הפונקציה isIsolated זה יעיל הרבה יותר ככה מפני שזה ייקח o(v) לעומת מטריצה שייקח o(v^2) אם היה פה הוצעת צלעות שעבור מטריצה זה היה לוקח o(1) אולי היה עדיף אבל אין כאן.

Int getVertex() Const – מחזיר לי את מספר הצמתים (1000 לפי מה שהגדרתם) , השתמשתי בזה ב- diam וגם ב- connectivity כדי לדעת על כמה צמתים לרוץ .

Void add Edge (int src,int dest) – בונה צלע בין 2 צמתים . דחפתי פעמיים לתוך התור עובר למשל 1-3 ו-3-1 כי זה גרף לא מכוון.

Int *BFS(int startVertex) – מריץ BFS מהצומת ההתחלתית ששלחנו לו ואותה הוא הופך על ההתחלה ל-Visited וגם נותן לה מרחק 0 במערך במקום ה-startvertex (למשל אם זה הצומת 1 אז במקום 1 במערך). לאחר מכן הוא יוצר תור שלשם הוא מכניס את הצומת שבא אנחנו מבקרים. כעת כל עוד התור לא ריק אנחנו נרוץ בלולאת for על השכנים של הצומת שאנחנו נמצאים עליה , אם לא ביקרנו באחד מהשכנים האלה אז נוסיף אותם לתור ונסמן אותם במערך visited ב-true ובמערך מרחקים נוסיף להם 1 לפי המרחק שיש לאבא שלהם, למשל אם d[v2]=3 ו-v4 שכן שלו אז d[v]= 3+1 =4 ← . ככה ממשיכים עד שהתור ריק ומחזירים את המערך מרחקים הקצר ביותר של הצמתים מהצומת ההתחלתית. סיבוכיות: O(V+E) כי רצים על כל הצמתים והצלעות.

Int diameter(graph &g) – מוצא את הקוטר של הגרף , בגדול צריך להריץ bfs מכל צומת. ה-diam יהיה שווה לצומת שהמרחק שלה במערך המרחקים של כולם הכי גדול (bfs מחזיר את המרחק הכי קצר של צומת מהצומת ההתחלתית). כמו כן יש התייחסות גם למקרי קצה : אם יש צומת אחת הפונקציה תחזיר 0. אם הגרף לא קשיר אז אין קוטר והפונקציה תחזיר 1-. סיבוכיות:
O(V*(V+E) כי אנחנו עושים BFS (V+E) על **כל הצמתים** לכן כפול v.

Int connectivity(Graph &g) – הפונקציה מקבלת גרף ומריצה BFS מהצומת 0 (לא באמת משנה מאיזו צומת מתחילים כדי לבדוק קשירות). הפונקציה רצה על המערך מרחקים שהתקבל מ-BFS בעזרת לולאת for מהאיבר **ה-1** (לא האפס) ובודקת אם יש מקום במערך ששווה 0 אם כן אז הגרף לא קשיר. הסיבה לכך שהתחלנו מ-1 ובדקנו כך שהתחלנו את BFS מהצומת 0 ולכן המרחק שלו תמיד יהיה 0, אז אם יש איבר נוסף במערך שהמרחק שלו הוא 0 אז בעצם לא הגענו אליו ולכן אפשר להסיק שגרף לא קשיר. כמו כן יש התייחסות למקרה קצה שיש צומת בודדת ולכן יוחזר שהגרף קשיר (באופן ריק). סיבוכיות:O(V+E) כי זה להריץ BFS מצומת מסוימת ולבדוק בעזרת לולאת for אם במערך מרחקים קיים 0 (אבל זה קבוע ולכן שולי).

Int Graph isIsolated() – הפונקציה עוברת על המערך רשימות , אם יש תא שהרשימה שלו ריקה זה אומר שהוא צומת בודד ולכן יוחזר 1 אחרת יוחזר 0.
סיבוכיות: O(v) כי אנחנו עוברים על המערך ומשתמשים ב-()empty כדי לבדוק אם ריק או לא.

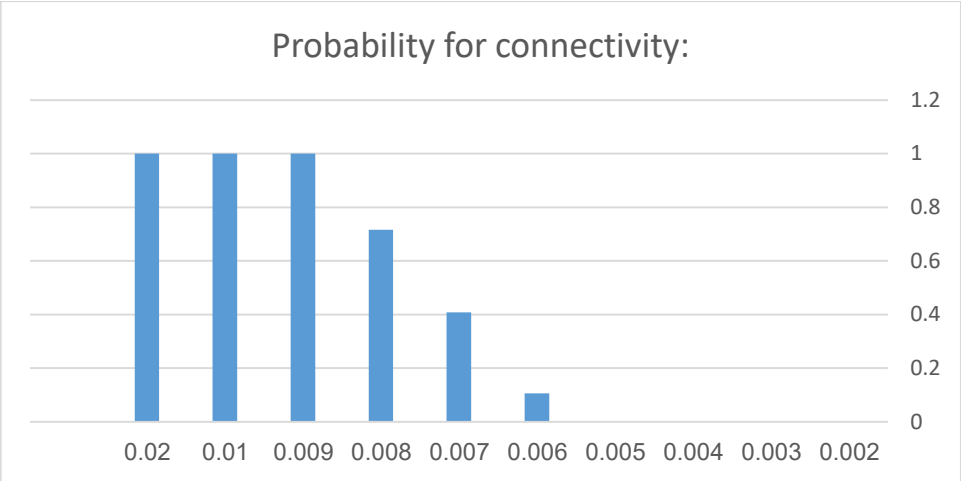
Graph build_random_graph(int v, double p) – בונה גרף רנדומלי לפי P , אם P שהגרלנו גדול מהמספר השני שהגרלנו אז נבנה צלע .
סיבוכיות: O(v^2) , כי אנחנו רצים עם לולאה בתוך לולאה לפי אותו גודל (v מספר הצמתים).

void WriteIntoCsv(int attribute,double *arrProbability,int *ArrAttribute) – הפונקציה מקבלת את מספר התכונה שבעזרתו ה-switch ידע אילו ערכי p של ה-threshold להכניס לתוך הקובץ csv, כאשר הערכים האלה נמצאים בתוך ה-arrProbability. כמו כן בעזרת ArrAttribute אנחנו נכניס לתוך הקובץ csv את מספר הגרפים שמקיימים את התכונה חלקי 500 שקיבלנו מהסימולציה עבור p מסוים.

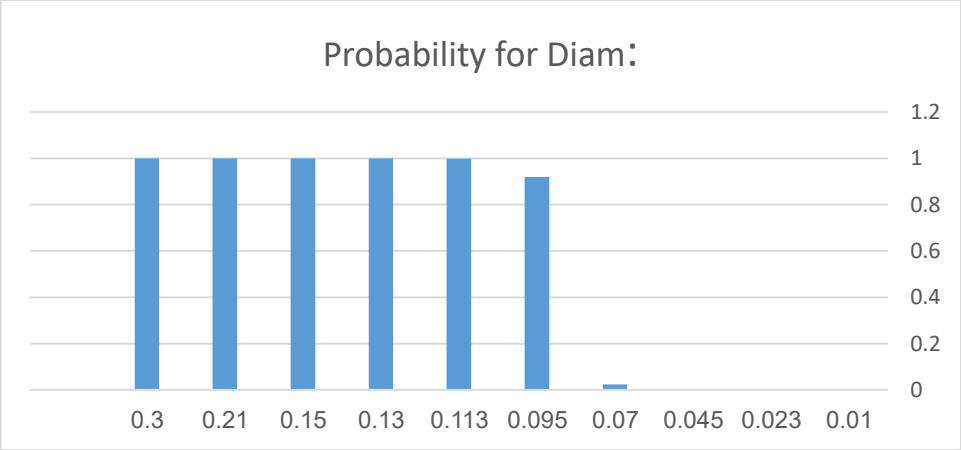
פונקציית מיין – בגדול יצרתי שלושה מערכים עבור כל תכונה , כאשר בכל אחד 5 p גדולים מערך ה- threshold ו5 קטנים והרצתי עבור כל תכונה לולאה בתוך לולאה על מנת לעשות את הסימולציה.

גרפים:

Connectivity



Diameter



Is Isolated