הסבר על הפונקציה המרכזיות והצגת התוצאות גרפים:

P\_arrayCalculate(double median, double(&returnArray)[10]);

פונקציה זאת נועדה עבור חישוב ה P reference, הנתונים שנכנסו ל P היו לפי חישוב הP בהתאמה ל Threshold

לפי האלגוריתם הבא :

הפונקציה מקבלת את ערך ה P (המחושב לפי הנוסחה בשאלה, לכל P בהתאמה) ומערך בגודל 10.

הפונקציה מתייחסת ל P כחציון מכניסה 10 ערכים לתוך מערך כאשר 5 מהם היו תמיד קטנים ו- 5 מהם היו תמיד גדולים מה P שהתקבל, כאשר הגבול של P חייב להיות בין 0 ל- 1 כולל.

בכל אחת מהטבלות של ה P המחושב בהתאמה לפונקציה, יצרתי מערך בגודל 10 שעליו הכנסתי 10 מספרים שכאשר 5 מהם קטנים מה P המחושב ו 5 מהם גדולים מה P המחושב לפי מה שהתבקש בתרגיל

Vertice\*\* build\_random\_graph(int v, double p);

הפונקציה נועדה להחזיר גרף שנבנה בצורה רנדומלי לפי כמות הצמתים שלו וההסתברות שיהיה חיבור בין הצמתים.

תיאור האלגוריתם :

שלבים עיקריים

שלב ראשון:

הפונקציה מייצרת V צמתים בודדים

שלב שני:

נרוץ בלולאה מקוננת כאשר הלולאה הפנימית תמיד תתחיל בערך של הלולאה החיצונית + 1

התפקיד של הלולאה המקוננת הוא לחבר בין הצומת בלולאה החיצונית לבית הצומת בלולאה הפנימית בהסתברות רנדומלית שמחושבת בין 0-1.

כאשר ההסתברות שיצא לנו בחישוב הרנדומלי יהיה קטן שווה לערך P שקיבלנו בפונקציה נבצע חיבור בין הצמתים על ידי שימוש בפונקציה Insert.

void insert(Vertice\*\* Graph, int v\_data);

פונקציה נועדה בשביל לבצע הכנסות לרשימה מקושרת שמכילה רשימות מקושרות שהם בעצם תיאור של כל הצמתים והשכנים שלהם.

לפי האלגוריתם הבא:

הפונקציה מקבלת מצביע לראש הרשימה שזה השכן שאליו אנחנו רוצים לבצע חיבור בשביל לחסוך זמן ריצה כל צומת חדשה שמתחברת נכנסה למקום השני בעצם למקום אחד אחרי הפוינטר שמצביע על ראש הרשימה של השכנים.

int Connectivity(Vertice\*\* graph);

הפונקציה נועדה לבדיקה האם הגרף קשיר במידה וכן היא תחזיר 1 במידה ולא היא תחזיר 0.

לפי האלגוריתם הבא:

השתמשתי באלגוריתם של BFS

int Is\_Isolated(Vertice\*\* graph);

הפונקציה נועדה לבדוק האם קיים בגרף לפחות צומת אחת בודדת אם כן תחזיר 1 אם לא 0.

לפי האלגוריתם הבא:

עובר על כל הצמתים ובודק אם יש לאחד הצמתים דרגה 0 במידה ומצאתי מחזיר 1 אחרת 0.

int FindDiameter(Vertice\*\* graph)

הפונקציה מחזירה את הקוטר של הגרף.

לפי האלגוריתם הבא:

נשתמש באלגוריתם של BFS

void SaveResultToFile(double P\_threshold[], double P\_threshold\_result[], string p\_test)

הפונקציה נועדה לבצע שמירה לקבוץ CSV של ה P referance שחושב ב P\_arrayCalculate ו ושל ה P שחושב כהסתברות של כמות הגרפים שהתכונה התקיימה בהם לבין הכמות הגרפים הכלליים שנוצרו.

**Results:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P connectivity | 0.001 | 0.002 | 0.003 | 0.005 | 0.006 | 0.022 | 0.039 | 0.055 | 0.072 | 0.089 |
| Probability that the feature exists | 0 | 0 | 0 | 0 | 0.046 | 1 | 1 | 1 | 1 | 1 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P diameter | 0.02 | 0.04 | 0.06 | 0.08 | 0.10 | 0.11 | 0.13 | 0.14 | 0.16 | 0.17 |
| Probability Exists | 0 | 0 | 0 | 0.158 | 0.948 | 0.996 | 1 | 1 | 1 | 1 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P isolated | 0.001 | 0.002 | 0.003 | 0.005 | 0.006 | 0.022 | 0.039 | 0.055 | 0.072 | 0.089 |
| Probability Exists | 0 | 0 | 0 | 0 | 0.044 | 1 | 1 | 1 | 1 | 1 |