בס"ד

קידוד הפונקציות:

תיאור הפונקציות הראשיות והפונקציות עזר שלהן:

1) build_random_graph:

לכל צלע באופן רנדומלי P פונקציה אשר מקבלת מס' צמתים מסוים ובונה גרף עם הסתברות.

בחרנו להשתמש במבני הנתונים "מערך של רשימות מקושרות" לייצג את הגרף.

נגדיר V- מס' הצמתים ו E- מס' הצלעות.

מבחינת יעילות למלא את הגרף יהיה בסיבוכיות של O(V+E) (ובמקרה הגרוע O(V^2). אם מדובר בגרף דליל, מימוש גרף דליל במטריצה תמיד יהיה בסיבוכיות של V^2 גם כאשר יש 2 צלעות על 1000 צמתים). המימוש שלנו עובד בצורה שכל צומת הינה תא במערך בגודל V וכל צומת מצביע לרשימה מקושרת המכילה את הצמתים המחוברים לצומת זה(שכניו). שימוש בזיכרון תלוי יותר בכמות הצלעות ופחות בכמות הצמתים.

לצורך מימוש זה השתמשנו בStructs הבאים:

```
//Structs for List
typedef struct list_node //represents every node
{
    int vertex;
    struct list_node* next;
}list_node;

typedef struct adj_list //represents the adjacency list
{
    struct list_node* head;
}adj_list;

typedef struct Graph //represents a gragh
{
    int V;
    struct adj_list* array;
}Graph;
```

יתרונות המימוש:

1. ניתן לעבור מהר יותר על כל הצלעות מכיוון שאפשר לגשת לכל השכנים של אותה צומת באופן ישיר (הרשימה אליו מצביע הצומת).
2. הוספה/הסרה של שכן (NODE) בסיבוכיות של O(1).
3. הוספת צלע הינה בסיבוכיות של O(1).

בס"ד

מהלך הפונקציה:

תהליך יצירת הגרף מתבצע ע"י פונקציית עזר **createGraph** אשר מקבלת מס V של צמתים.

*** createGraph-**הפונ' יוצרת מערך של מצביעים שבהמשך יצביעו לרשימה מקושרת שתכיל את השכנים של אותה צומת.

סיבוכיות הפונקציה הינה O(V).

אנו עוברים בשתי לולאות כאשר הלולאה החיצונית הינה צומת מסויימת והלולאה הפנימית הינה כלל הצמתים האפשריים שיכולים ליצור צלע עם צומת זו (תוך כדי בדיקה ששני האינדקסים אינם שווים כי צומת אינה מחוברת לעצמה). בעזרת פונקציית ()rand מוגרלים מספרים רנדומלים בתחום בין 0 ל-1 (תחום הסתברות), אם המספר שהוגרל גדול מההסתברות הנתונה לצלע אז נוסיף צלע לגרף באמצעות פונקציית העזר addEdge.

*** addEdge-**פונקציה זו מקבלת גרף, צומת שהינה המקור וצומת שהינה היעד (השכן של צומת המקור).
מתבצע חיבור בין הצמתים (ע"י הקצאת **Node** בעזרת פונקציית העזר (**new_list_node**)) של היעד (שכן) וחיבורו אל הפוינטר של צומת המקור וההפך (כי הגרף אינו מכוון).

סיבוכיות הפונקציה הינה O(1)

סיבוכיות הפונקציה הכוללת הינה O(V+E) במקרה הטוב ובמקרה הגרוע הסיבוכיות הינה O(V^2).

2)Is_Isolated:

פונקציה זו מקבלת גרף הנתון. היא עוברת על מערך הצמתים שבגודל V ובודקת האם קיימת צומת שמצביע לNULL, אם כן אזי הוא אינו מצביע לאף רשימת צמתים ולכן אין לצומת שכנים והינו צומת מבודד. אם ישנו צומת מבודד הפונקציה מחזירה 1, אחרת 0.

סיבוכיות הפונקציה הכוללת הינה O(V).

בס"ד

פונקציית עזר לפונקציות לבדיקת קשירות ומציאת הקוטר BFS_Check4ConnectAndDiam:

בפונקציה הנ"ל ניעזר באלגוריתם הידוע בשם BFS ביחד עם תוספת קטנה משלנו.

BFS- אלגוריתם זה הינו חיפוש לרוחב. הוא אלגוריתם המשמש למעבר על צומתי גרף, לרוב תוך חיפוש צומת המקיים תכונה מסוימת. חיפוש לרוחב סורק את הגרף בצורה שמבטיחה שכל צומת שנמצא **באותו רכיב קשירות** של הצומת ההתחלתי ייבדק ,וסריקה זו נעשית בזמן אופטימלי, הליניארי למספר הקשתות והצמתים בגרף.

הBFS יקבל גרף נתון, צומת התחלתית, מערך BY REF בשם visited המאותחל בערך "0", ומשתנה REF BY מסוג INT שיחזיק את המרחק המקסימלי מבין כל המרחקים המינימלים לצומת הנבדקת.

האלגוריתם משתמש במבנה נתונים מסוג תור על מנת לקבוע מהו הצומת הבא בו הוא עומד לבקר. בכל פעם שהוא מבקר בצומת הוא מסמן אותו ככזה שנבדק בעזרת הערך "1" בתוך המערך (Visited), ואז בודק את כל הקשתות שיוצאות ממנו . אם קשת מובילה לצומת שטרם נבדק, צומת זה מתווסף לתור. בדרך זו מובטח כי האלגוריתם יסרוק את הצמתים בסדר שנקבע על פי מרחקם מהצומת ההתחלתי (כי צומת שנכנס לתור יצא ממנו רק לאחר שכל הצמתים שהיו בו קודם יצאו). סיבוכיות האלגוריתם BFS הינה O(V+E).

בנוסף הפונקציה גם שומרת את המרחק המקסימלי של הצומת המקורית לשאר הצמתים ומחזירה BY REF את המרחק המקסימלי מבין כל המינימלים. (במשתנה temp_diam - אופציה לקוטר).

סיבוכיות הפונקציה הינה O(V+E).

3)connectivity:

פונקציה המקבלת גרף נתון ובודקת אם הינו קשיר או לא.

נבדוק קשירות בעזרת האלגוריתם **BFS_Check4ConnectAndDiam**

בכדי לדעת האם הגרף קשיר נותר לעבור על המערך visited , אם בתא מסויים יהיה ערך "0" כלומר שלצומת ההתחלתית (הצומת שנשלחה לפונ' עזר **BFS_Check4ConnectAndDiam**) אין מסלול לצומת זו ולכן הגרף אינו קשיר. עבור גרף לא קשיר נחזיר 0 ועבור גרף קשיר נחזיר 1 .

סיבוכיות הפונקציה הינה O(V+E) + O(V) = O(V+E).

בס"ד

4) diameter:

פונקציה המקבלת גרף נתון.

תחילה נבדוק ע"י הפונקציה connectivity האם הגרף קשיר, במידה והגרף אינו קשיר הקוטר הינו אינסוף (נחזיר INT_MAX).

נעבור על מערך הצמתים בגרף ונקרא לפונקציה **BFS_Check4ConnectAndDiam** עבור כל צומת בגרף.

תחילה נגדיר משתנה temp_diam שיהיה מאותחל ל-0. ובכך נבדוק כל פעם האם האופציה לקוטר שקיבלנו מהפונקציה **BFS_Check4ConnectAndDiam** גדולה יותר מהקודם לו ובכך נקבל בסוף את הקוטר (המרחק המקסימלי מבין כל המינימלים בגרף). בכל קריאה לפוקנציה **BFS_Check4ConnectAndDiam** נדרש לאפס את מערך ה-visited בשביל הצומת החדשה בסיבוכיות O(v).

סיבוכיות הפונקציה הינה O(V*(V+E)) = O(V*((V+E)+V)).

פונקציות עזר בסיסיות נוספות לרשימה ולתור:

- **New_list_node –** יוצרת Node חדש, במקרה שלנו יוצרת שכן.
- **freelist-** משחררת את הרשימה מהזיכרון.
- **freeGraph-** משחררת את הגרף מהזיכרון, כלומר את כל הרשימות (השכנים).
- **createQueue-** יוצרת תור.
- **enqueue/dequeue-** מכניסה איבר לתור/מוציאה איבר מהתור.
- **freeQueue-** משחררת את התור שהוקצאה דינמית.
- **isEmpty-** בודקת האם התור ריק.
- **Size-** מחזירה את גודל התור.

בס"ד

סימולציות:

תיאור הפונקציות עזר למען הסימולציות:

***createTable**- פונקציה זו מקבלת שם קובץ ה CSV שיבנה בהתאם לתכונה הנבדקת, הפונקציה שנעזר בנתונים שלה לבניית מערך הממוצעים, מספר הצמתים בגרף, מספר הריצות וה**Threshold** (מוגדר לפי כל תכונה בשליחה לפונקציה).

בתוך הפונקציה ניצור מערך ממוצעים שבעזרתו נחשב את ממוצע התכונות של הגרפים באותה הסתברות במס' ריצות מסויים.

בנוסף, בתוך הפונקציה ניצור מערך בגודל 10 של הסתברויות בעזרת פונקציית עזר בשם **makeProbsArray .**

***makeProbsArray**- יוצרת מערך עם בדיוק 5 מספרים גדולים מערך ה threshold-ו-5 מספרים קטנים ממנו.

לאחר מכן נרוץ ב2 לולאות, לולאה חיצונית רצה על מערך ההסתברויות(**Probs**) ולולאה פנימית הרצה על כל תא במערך ההסתברויות לפי מספר הריצות המוגדר(**Runs**). בתוך הלולאות יש קריאה לפונקציה **build_random_graph** שכאמור יוצר גרף רנדומלי בהסתברות מסויימת לכל צלע.

בכל ריצה על כל ערך במערך ההסתברויות ישנה קריאה לפונקציה הרלוונטית שאנו רוצים לבנות את התכונה שלה (קוטר, קשירות, צומת מבודד) ומתבצעת סכימה של הערכים המוחזרים לתוך מערך הממוצעים. בסוף הריצות על הסתברות מסויימת במערך נחלק את הערך שסכמנו במס' הריצות ובכך נקבל ממוצע לתכונה.

בסוף הפונקציה נקרא לפונקציית עזר**writeToCSV .**

***writeToCSV-** הפונקציה מקבלת שם של קובץ CSV (לפי התכונה), מערך הסתברויות ומערך ממוצעים.
פותחת קובץ CSV לכתיבה וכותבת לטבלה.

בשורה הראשונה את מערך ההסתברויות ובשורה השנייה את מערך הממוצעים.

***createTable4Diam**- הפונקציה הנ"ל זהה לפונקציה מעל (createTable), מלבד לבדיקה נוספת של הערך החוזר מהפונקציה Diameter.

נבצע נרמול- נספור את כמות הגרפים שעונים על התכונה שהקטור קטן שווה ל-2.

בס"ד

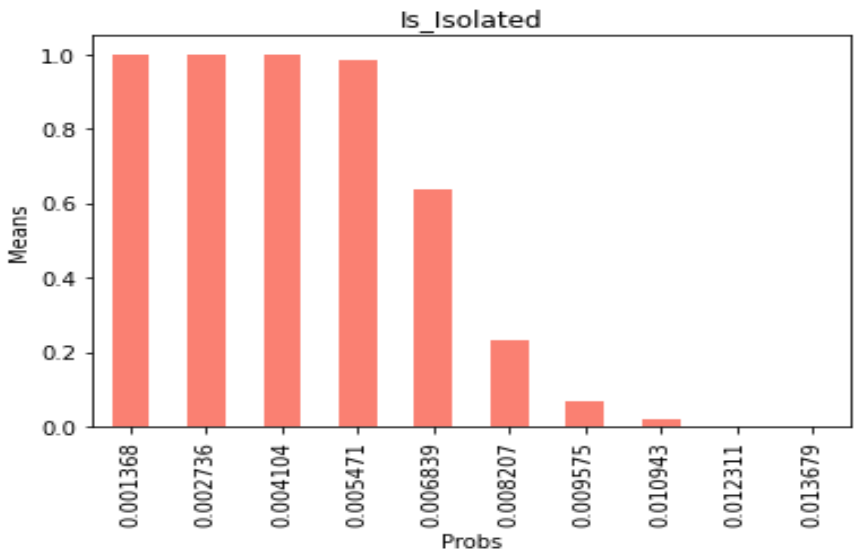
תוצאות הסימולציות:

הערה: כלל הנתונים מתייחסים ל-1000 צמתים ו-500 ריצות, סה"כ 5000 גרפים נבדקים לכל תכונה.

Is_Isolated:

K	J	I	H	G	F	E	D	C	B	A
p	0.013679	0.012311	0.010943	0.009575	0.008207	0.006839	0.005471	0.004104	0.002736	0.001368
m	0.002	0.006	0.02	0.048	0.25	0.658	0.986	1	1	1

Threshold: 0.006907755



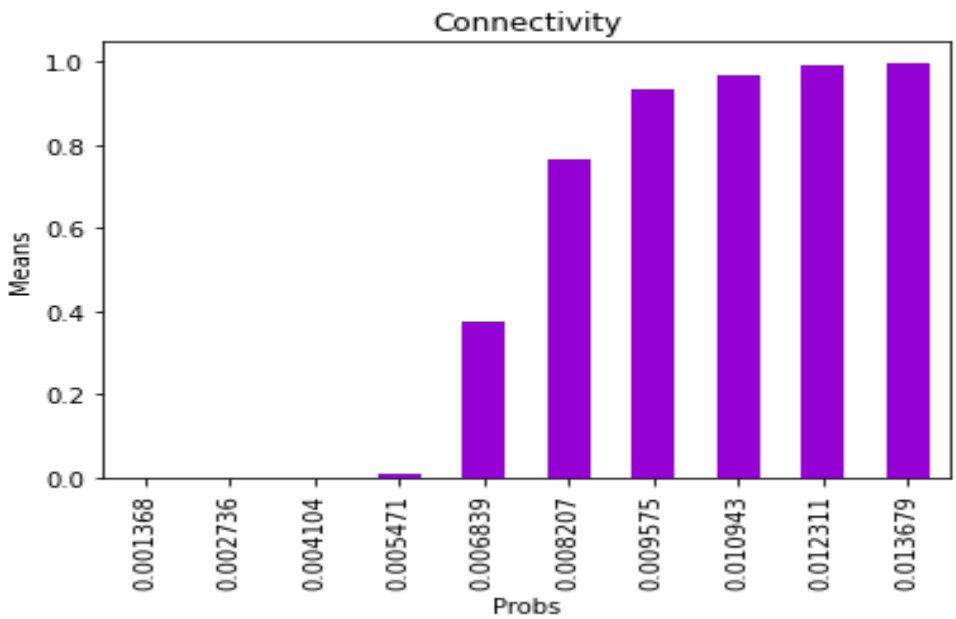
מסקנה- ניתן לראות לפי הגרף הנתון שכאשר יש גרפים הנבנים בהסתברות נמוכה יותר מהThreshold יש הסתברות גבוהה שקיים צומת מבודד בגרפים הנתונים. כאשר אנו מתקרבים לThreshold ניתן לראות ירידה בכמות הגרפים בעלי הצומת המבודד עד כדי 0. **מ.ש.ל**

בס"ד

Connectivity:

	J	I	H	G	F	E	D	C	B	A
p	0.013679	0.012311	0.010943	0.009575	0.008207	0.006839	0.005471	0.004104	0.002736	0.001368
m	1	0.998	0.98	0.946	0.778	0.402	0.028	0	0	0

Threshold: 0.006907755



מסקנה- ניתן לראות לפי הגרף הנתון שכאשר יש גרפים הנבנים בהסתברות נמוכה יותר מהThreshold יש הסתברות נמוכה שהגרפים הנתונים קשירים. כאשר אנו מתקרבים לThreshold ניתן לראות עלייה בכמות הגרפים הקשירים. **מ.ש.ל**

הערה נוספת:

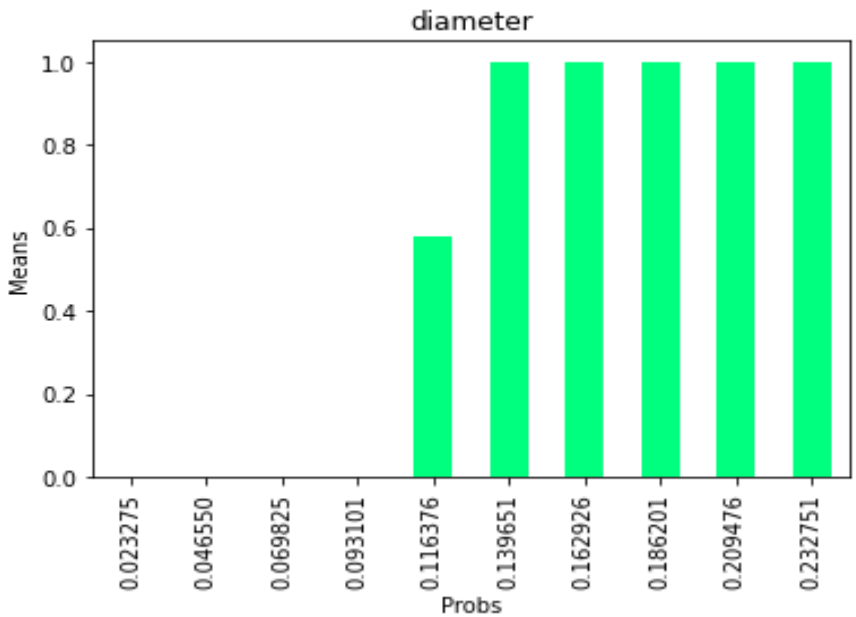
מ2 המסקנות שלמעלה ניתן לראות שהגרפים מתנהגים באופן מנוגד, כלומר כאשר יש הסתברות גבוהה שהגרפים הנתונים יהיו בעלי צומת מבודד אז נקבל שיש הסתברות נמוכה שהגרפים הנתונים קשירים, ולהפך.

בס"ד

Diameter:

K	J	I	H	G	F	E	D	C	B	A
P	0.232751	0.209476	0.186201	0.162926	0.139651	0.116376	0.093101	0.069825	0.04655	0.023275
M	1	1	1	1	0.998	0.572	0	0	0	0

Threshold: 0.1175394



מסקנה- ניתן לראות לפי הגרף הנתון שכאשר אנו מתקרבים לערך ה- Threshold מחצית מהגרפים הנחקרים הקוטר שלהם שווה לערך 2, ושאנו עוברים את ערך ה- Threshold הגרפים מגיעים לקוטר 2 עד כדי קבוע.
בהסתברויות הקטנות מהThreshold נקבל שהקוטר יהיה גדול מ2 (אינסוף במקרה של גרף לא קשיר או מספרים שיותר גדולים מ-2). **מ.ש.ל**

בס"ד

מסקנות נוספות:

חקרנו כאן את תכונות הגרפים -קשירות, צומת מבודדת, קוטר.

גרפים שנבנים באופן רנדומלי כלומר ללא כל התערבות מצד המשתמש!

כל צלע נבנתה בהסתברות P וכך הגרף נבנה לפי מודל "ארדש - ריניה" - מודל ליצירת גרפים אקראיים.

הצלחנו להוכיח את כלל הטענות ומשום שהוכחנו אותם עבור מדגם של גרפים רנדומליים באופן אקראי כמובן וללא תלות, הטענות נכונות תמיד.

ניתן לראות את החיבור בין התכונות - קשירות משפיעה על התכונה של צומת מבודדת ולהפך.
עצם היותו של גרף קשיר עם כמות צלעות גדולה כך הקוטר יהיה קטן יותר כי יהיו יותר מסלולים מכל צומת לצומת בדרך קצרה יותר.