שם: עומר אלכסנדר , ת"ז: 313703415

**ממ"ן 15**

חלק א'

כיצד להריץ ולהפעיל את האלגוריתם:

תחילה יש להכניס את קובץ הzip ששלחתי לתוך תיקיית הprojects של sinalgo. שם התיקייה שצריך להכניס הוא maman15.

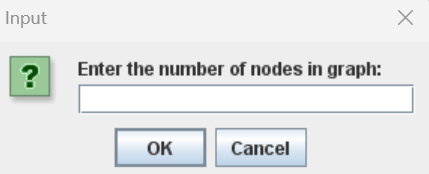
לאחר מכן יש לפעול ע"פ ההסבר הרשום בקובץ description.txt המצורף:

* 1. תחילה יש לפתוח את הפרויקט maman15.

Graphical user interface, text, application

Description automatically generated

* 1. לאחר מכן יש לבנות גרף חדש ולהכניס בו את כמות הרכיבים שנרצה.



* 1. לבחור את הרכיב שיהווה כשרת:



* 1. לאחר מכן יש להריץ את הפרויקט. האלגוריתם ייעצר אוטומטית ברגע שיסתיים.
  2. Graphical user interface, text, application

     Description automatically generatedפעולות נוספות שניתן לעשות בזמן האלגוריתם:
     1. הדפסת סכום הצלעות בגרף.
     2. הדפסת הפרגמנטים בגרף.
  3. פעולות נוספות שניתן לעשות לאחר סיום האלגוריתם:
     1. הדפסת סכום הצלעות בMST שנבנה.
     2. ייצוא התוצאות לקובץ אקסל results.csv.

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

* + 1. שליחת הודעות מרכיב נבחר לserver.



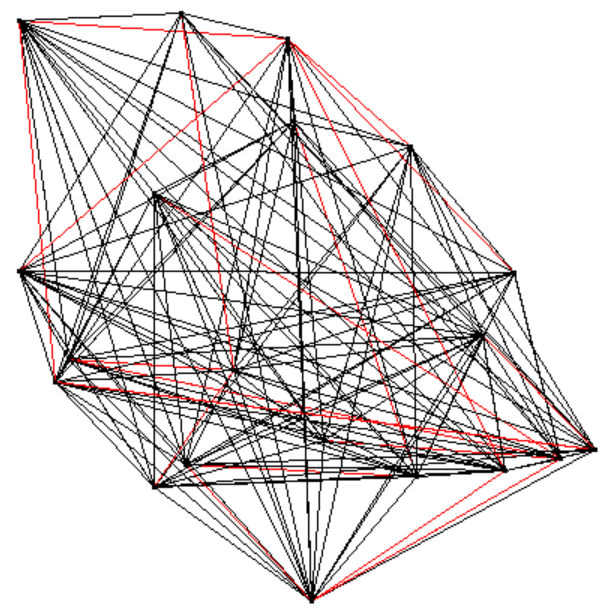
Text

Description automatically generated Graphical user interface, application

Description automatically generated

הצלעות יצבעו באדום כשהצלע היא חלק מהMST בסיום האלגוריתם, ובצהוב כשיש הודעה על הצלע:

Chart, diagram

Description automatically generated

הסבר ותיעוד מרכיבי הקוד העיקריים בפתרון:

הקוד מורכב מכמה חלקים עיקריים:

1. CustomGlobal:

זהו החלק שקיים כמו בשאר הפרויקטים, דרכו אנו שולטים במסך שהמשתמש רואה ובפעולות הניתנות לביצוע כפי שתיארתי למעלה.

נציין כי בניית הגרף מתבצעת כפי שהתבקשנו, תחילה מקבלים מהמשתמש את כמות הרכיבים בגרף. לאחר מכן יוצרים את הנקודות בפיזור אקראי, ולכל רכיב מייצרים 7 חיבורים לרכיבים אקראים אחרים בגרף ומגרילים לצלע בינהם משקל אקראי בין [1,….,1000000000] , בנוסף בבניית הגרף המשתמש קובע מי מהרכיבים יוגדר כserver.

1. GHSMessage:

זהו החלק אשר אחראי על כל שליחת ההודעות בין הרכיבים במהלך האלגוריתם GHS.

ע"פ השלבים באלגוריתם המתוארים בתיאור המשימה יש מספר סוגים של הודעות שיכולות לעבור בין רכיבים בגרף שמתוארים בMessageType , לאורך האלגוריתם מועברים בין הרכיבים המון הודעות שעוזרות לרכיבים להתקדם בשלבי האלגוריתם ומעבירים בינהם את המידע החשוב הרלוונטי.

(לדוגמה = עדכון פרגמנט ע"י הודעה מסוג FragmentIDUpdateMessage).

1. EdgeWithWeight:

זהו המימוש של הצלעות בין הרכיבים בגרף. כמו שציינתי קודם בבניית הגרף לכל צלע מוגרל משקל אקראי. בנוסף למשקל, יש משמשעות לצבע של כל צלע כך שיהיה ניתן לראות באופן ויזואלי את הנעשה במהלך ריצת האלגוריתם.

צבע צהוב – מהווה שיש הודעה כלשהי על הצלע אשר מועברת.

צבע אדום – מהווה שהצלע הנוכחית היא חלק מהMST של הגרף.

1. GHSNode:

זהו החלק העיקרי בבניית הפתרון. הגרף מורכב מN רכיבים כאלו ע"פ בקשת המשתמש.

הלוגיקה של רכיב זה מושתת על ההסבר השלבים של האלגוריתם המופיע בתיאור המשימה ולכן לכל רכיב כזה יש משתנה currentState אשר מהווה את השלב של הרכיב בהתקדמות האלגוריתם. מבנה זה מאפשר לגרף כולו לעבוד בצורה מבוזרת ובכך אין תנאי שכל רכיב יהיה באותו סטטוס במהלך הפתרון.

המעבר בין השלבים בפתרון מגולם במעבר של כל רכיב בסטטוס שלו.

בכל שלב יש פעולת איטרציה שמופעלת ע"פ הסטטוס של הרכיב וע"פ הזמני ריצה המבוקשים יש התקדמות בין השלבים והסטטוסים.

(לדוגמה = FRAGMENT\_ID\_DISCOVERY מתבצע רק N איטרציות כפי שמבוקש בתיאור העבודה)

1. RequestMessage / ResponseMessage:

לאחר בניית הMST, כפי שציינתי קודם יש אפשרות לשלוח בין הרכיבים לשרת הודעות. הדבר מתבצע בעזרת סוגי ההודעות הנ"ל כתלות בהאם ההודעה היא בקשה מהרכיב לשרת או תגובה מהשרת לרכיב.

תוצאות האלגוריתם עבור הגרפים השונים שבדקתי:

התוצאות נמצאות בקובץ results.csv המצורף, וצירפתי אותם גם פה:

|  |  |  |
| --- | --- | --- |
| Number of nodes | Sum of weights | Sum of weights in MST |
| 2,000 | 6,997,160,707,304 | 166,291,559,145 |
| 1,954 | 6,826,843,807,353 | 166,277,978,133 |
| 1,822 | 6,381,669,884,537 | 151,171,904,717 |
| 1,748 | 6,119,225,111,455 | 145,161,659,014 |
| 1,666 | 5,882,745,200,486 | 140,916,365,998 |
| 1,513 | 5,358,433,586,511 | 128,686,416,555 |
| 1,399 | 4,878,380,156,797 | 113,945,396,619 |
| 1,344 | 4,703,933,402,447 | 115,871,669,636 |
| 1,228 | 4,251,780,179,567 | 100,455,096,910 |
| 1,111 | 3,883,607,437,645 | 91,092,208,942 |
| 1,009 | 3,542,820,000,000 | 83,250,098,491 |
| 952 | 3,348,208,079,557 | 79,836,610,661 |
| 821 | 2,844,347,835,244 | 67,518,036,155 |
| 777 | 2,740,940,026,841 | 64,422,148,202 |
| 666 | 2,314,722,572,679 | 54,075,469,132 |
| 555 | 1,964,966,145,855 | 48,563,285,437 |
| 444 | 1,552,531,584,103 | 37,287,602,312 |
| 333 | 1,147,107,868,565 | 26,020,368,258 |
| 222 | 779,954,646,519 | 17,720,302,437 |
| 200 | 695,343,077,236 | 16,207,998,022 |

חלק ב' - רשתות SINR

מודל SINR (יחס רעש הפרעות אות) הוא מודל מבוזר ריאליסטי יותר. בדגם זה הודעות עלולות ללכת לאיבוד עקב שידורי צמתים אחרים שמפריעים להם.

כפי שמוצג ב-Goussevskaia, Moscibroda and Wattenhofer 2008, אם כל צומת ברשת מודל SINR ישדר את ההודעות שלו בהסתברות , ההודעה שלו תתקבל בהצלחה על ידי כל שכנותיו בסבירות גבוהה ב 𝑂(Δlog𝑛) סבבים.

זה אומר שבשביל שהפתרון הנ"ל יתמוך ברשתות SINR יאנחנו יכולים לשנות את האלגוריתם שלנו כך שיתאים למודל SINR על ידי הדמיית כל סיבוב שלו עם סיבובי 𝑂(Δlog𝑛) של האלגוריתם שהזכרתי קודם, אשר בצומת משדר את ההודעות שלו בהסתברות .

שיטה זו תגרום לזמן הריצה של האלגוריתם שלנו לנוע מ-𝑂(𝑛log𝑛) ל-𝑂(Δ𝑛log2𝑛).

כדי ליישם את זה בקוד, פשוט נרחיב כל מצב להיות 𝑐Δlog𝑛 ארוך יותר, ובכל פעם שנשלח הודעה נעשה זאת בהסתברות של . אנו יכולים להשיג זאת בקלות על ידי דריסת שיטת השליחה במחלקה GHSNode וליישם אותה כך.