מסמך תיעוד פרויקט גמר בקורס אלגוריתמים מבוזרים לרשתות תקשורת – תיעוד יומי

מגיש: שלומי דומננקו

מסמך זה מסכם את התיעוד היומי שלי של הפרוייקט. כלומר כל יום אני רושם מה עשיתי והתוצאות הסופיות ש לאותו יום. בנוסף יש את הטבלה הסופית של 20 ריצות.

הקוד נמצא בגיטהאב, אך גם צירפתי אותו לקובץ זיפ:

<https://github.com/ShlomiRex/Sinalgo_Distributed_Systems_Matala15>

מסמך זה מתעד באופן כרונוגרפי (לפי זמן) את מימוש הפרויקט גמר באלגוריתמים מבוזרים לרשתות תקשורת.

**יצרתי סרטונים ביוטיוב שמראים את הקוד שלי עובד, מומלץ לצפות:**

* סרטון שמראה איך האלגוריתם מוצא MST: <https://www.youtube.com/watch?v=_5p9-792u5g>
* סרטון שמראה איך הלקוח שולח הודעה ואיך השרת שולח תשובה: <https://www.youtube.com/watch?v=PrCY5Sg487c>
* סרטון שמסביר בפירוט את כל הקוד שלי (זה ארוך): <https://youtu.be/VQf7QPvxaiE>

Contents

[23 בפברואר (יום חמישי) 3](#_Toc129962697)

[25 בפברואר (יום שבת) 5](#_Toc129962698)

[27 בפברואר (יום שני) 7](#_Toc129962699)

[28 בפברואר (יום שני) 14](#_Toc129962700)

[1 במרץ (יום רביעי) 16](#_Toc129962701)

[3 במרץ (יום שישי) 21](#_Toc129962702)

[5 במרץ (יום ראשון) – פאזה 7 23](#_Toc129962703)

[6 במרץ (יום שני) – פאזה 8 29](#_Toc129962704)

[8 במרץ (יום רביעי) 34](#_Toc129962705)

[9 במרץ (יום חמישי) 41](#_Toc129962706)

[12 במרץ (יום ראשון) 43](#_Toc129962707)

[14 במרץ (יום שלישי) 43](#_Toc129962708)

[15 במרץ (יום רביעי) 53](#_Toc129962709)

[16 במרץ (יום חמישי) 59](#_Toc129962710)

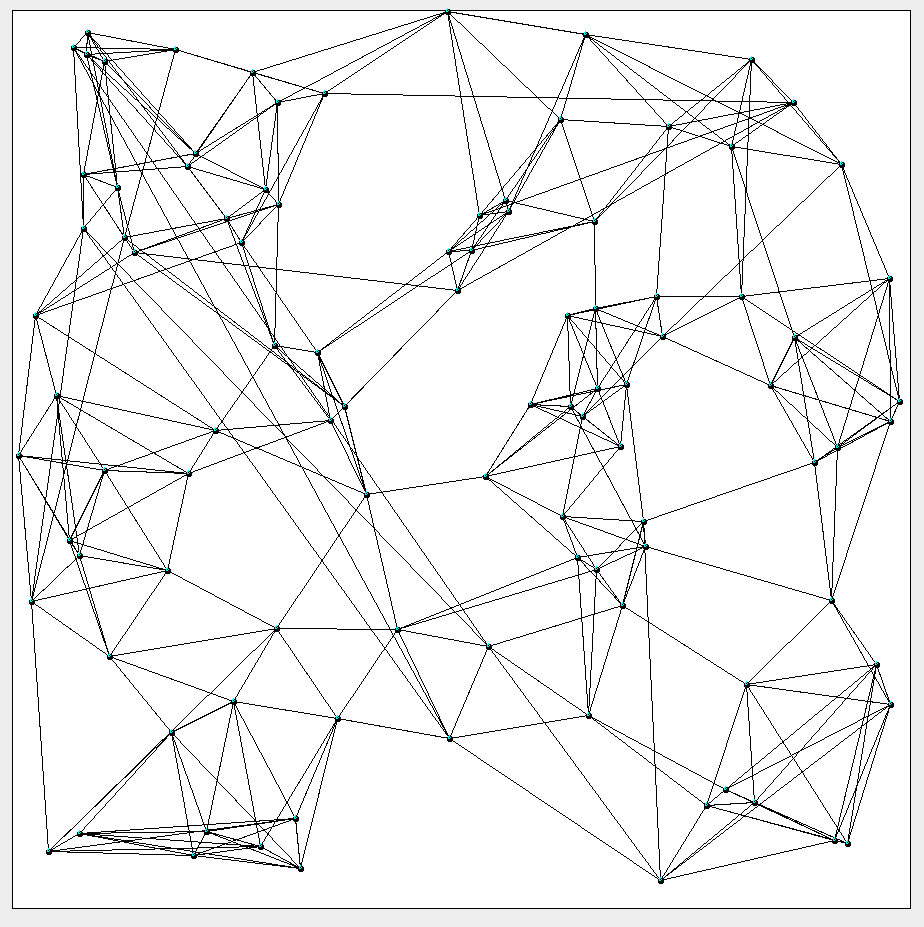
[17 במרץ (יום שישי) 60](#_Toc129962711)

תיעוד

# 23 בפברואר (יום חמישי)

הצלחתי למממש את החלק הראשון של מרכיב 2 אשר בהנתן מספר קודקודים, יוצרים n קודקודים, ואז כל קודקוד בוחר 7 קודקודים אחרים ומחבר אותם עם צלע. זה היה מאד מסובך כי הדרך היחידה לקבל קשתות זה דרך outgoingConnections לכל קודקוד, וצריך למצוא 2 קשתות שמחברות בין 2 קודקודים ( כי bidirectional = 2 קשתות).

אני עשיתי משהו מיוחד, במקום לבחור רנדומלית 7 קודקודים, בחרתי 7 קודקודים הכי קרובים, ואז הגרף יראה הרבה יותר יפה (מאשר למשל קליקה, יש יותר מדי צלעות שחוצות אחת את השני). עשיתי את זה ע"י שכל קודקוד A מחשב רשימה של זוג סדור: קודקוד אחר B, והמרחק בין A לבין B. ואז עושה sort ובוחר קודקוד B שיש לו פחות מ-7 קשתות. ואם לקודקוד A יש 7 או יותר צלעות, מדלגים ועוברים הלאה.



אפשר לראות שהגרף דומה לגרף עם רכיבי קשירות כמעט.

כעת בשביל ההגרלה של weighted edge אני שיניתי את הקונפיגורציה שישתמש בצלע (קלאס) משלי:

<edgeType value="matala15:WeightedEdge" />

עכשיו ה constructor היה מקבל long weight אבל זה קורס כאשר אני מוסיף צלע, ולכן אני צריך לאתחל קונסטקטור ללא פרמטרים, ורק אחרי האתחול שלו אני מציב לו weight. אז זה מה שעשיתי וזה עבד.

כעת נתקלתי בבעיה שנמשכה כמעט כל היום וניסיתי לתקן אותה, זה אחרי שבחרתי weight לכל צלע, ואני מדפיס על כל צלע את המשקל, אני מקבל פעמיים טקסט:

Text, letter

Description automatically generated

מסתבר שהייתי צריך שהצלע תהיה bidirectional אבל זה לא הספיק, הייתי צריך גם להוסיף מנגנון "neighbors" כי ב sinalgo אי אפשר לקבל את הצלעות (חוץ מ- outgoingConnections), ולכן צריך לאתחל פעם אחת bidirectional edge.

תיקנתי את הבעיה הזו ע"י מציאת outgoingEdge של currentNode והצבה weight והצבה flag בוליאני שאומר אם לצייר טקסט ב edge הזה:

A picture containing diagram

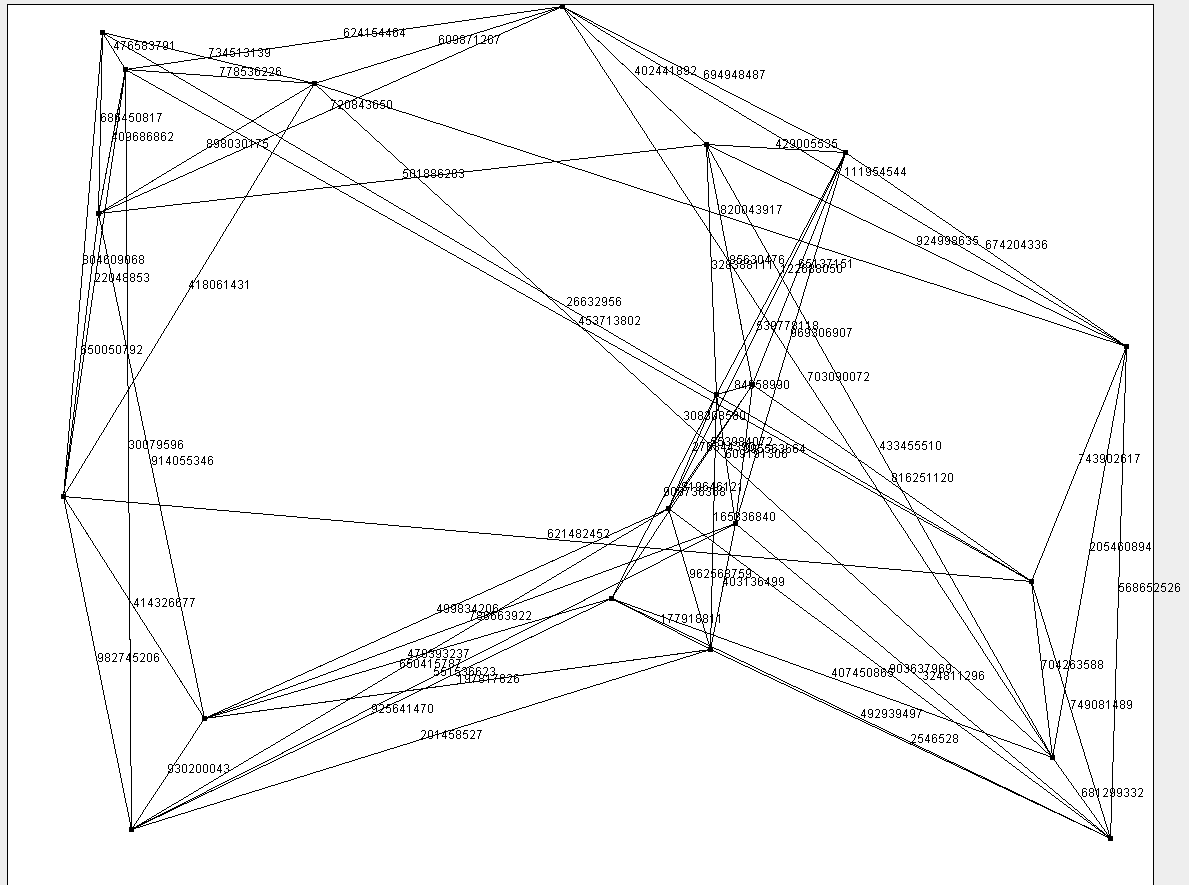
Description automatically generated

Text, letter

Description automatically generated

אפשר לראות שעכשיו שני ה edges (קיים אחד bidirectional אבל שניים outgoingEdges) יש להם אותו weight ורק אחד מצייר. זה לקח לי הרבה שעות (מעל 5). כי בסינלגו אין דרך לקבל edge בין 2 קודקודים. וזה מציק. אז צריך לעשות מלא חיפושים ב outgoing edges ומלא סיבוכים.

תוצאה סופית להיום:



בונוס: גיליתי שאם עושים hovering על צלע רואים את פונקציה toString שלו:

Diagram

Description automatically generated

# 25 בפברואר (יום שבת)

כאשר הגרף נראה כמו למעלה, ועושים step פעם אחת, משום מה, כל הקשתות נמחקות. אחרי כמה שעות של בדיקות, מסתבר שקובץ ה configuration היה הבעיה:

לפני:

<!--Default connectivity model used when none is specified-->

<DefaultConnectivityModel value="UDG" />

אחרי:

<DefaultConnectivityModel value="StaticConnectivity" />

ועכשיו הקשתות לא נמחקות כאשר עושים צעד ראשון.

הנה תמונה להמחשה (לפני התיקון – כלומר UDG): בצעד 0:

Diagram

Description automatically generated

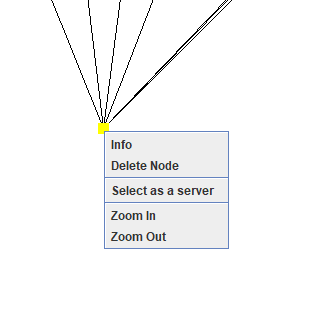
בצעד 1 (לא עוברות שום הודעות, זה לפני מרכיב 1) מקבלים שכל הצלעות 'נעלמו':

A picture containing text, flying

Description automatically generated

אפשר לראות שיש קשת אחת בין 2 קודקודים, וזה היה רמז לזה שזה קשור ל connectivityModel כי קראתי על זה קצת (בייחוד UDG כי שיחקתי עם Generate Nodes ב GUI).

בנוסף הוספתי אפשרות לבחור קודקוד כשרת (כפי שביקשו ברכיב 2):

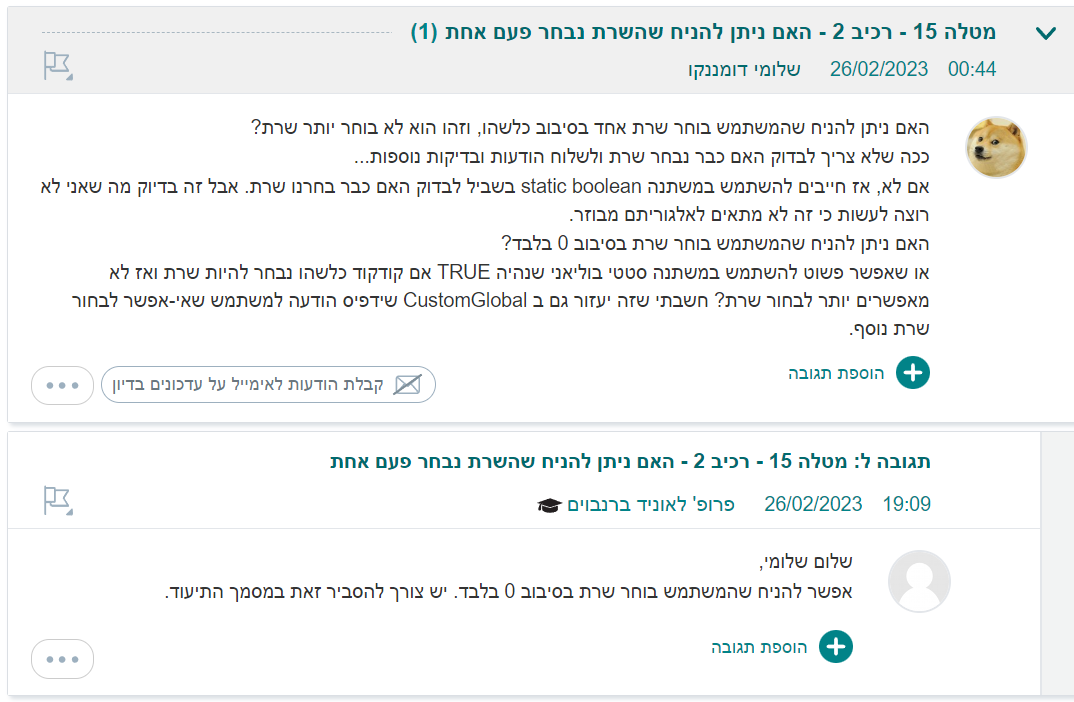


כאשר לוחצים על “Select as a server” הוא נצבע ישר בצהוב באותו סיבוב.

כעת נשאר למממש רכיב 1.

# 27 בפברואר (יום שני)

שאלתי את המרצה האם אפשר להניח שמשתמש בוחר בסיבוב 0 את השרת, הוא אמר שכן:

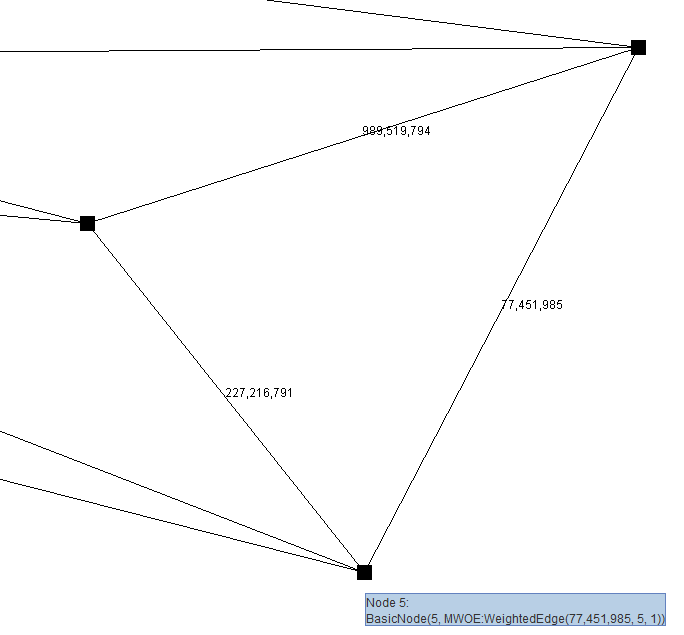


ולכן לא צריך לבדוק האם כבר בחרנו שרת וכו.

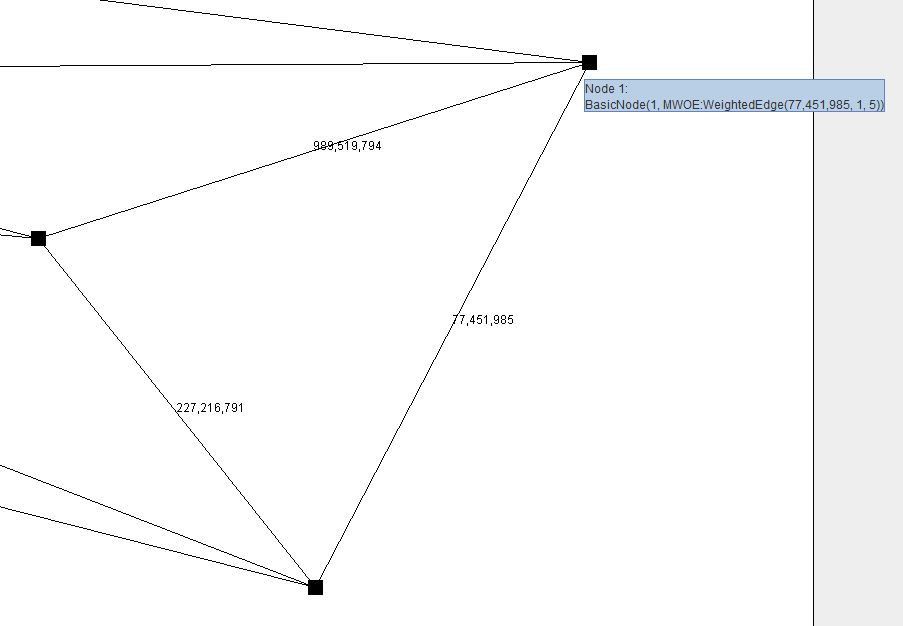
כעת רושמים את מרכיב 1.

נממש בחירת צלע הקלה ביותר לכל קודקוד ב-preStep. הסיבה שזה ב-preStep זה בגלל שזה מאפשר לנו להשתמש ב MWOE בהמשך האלגוריתם, כלומר לחשב MWOE זה נעשה בזמן ולכן זה מתאים שם.

אפשר לראות שב toString ה MWOE של קודקוד 5 הוא 77 מיליון (והוא מחובר לקודקוד 1):

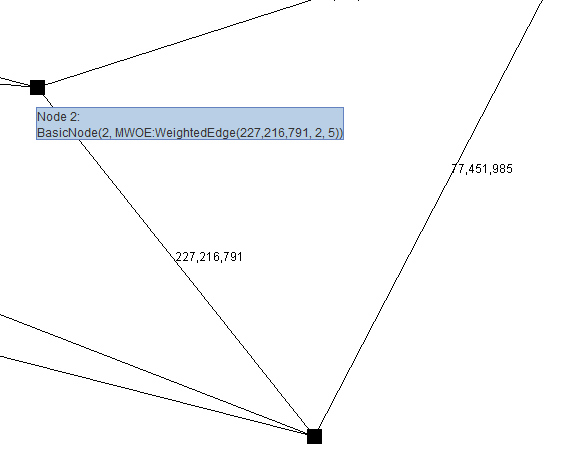


אותה צלע היא גם MWOE של קודקוד 1:



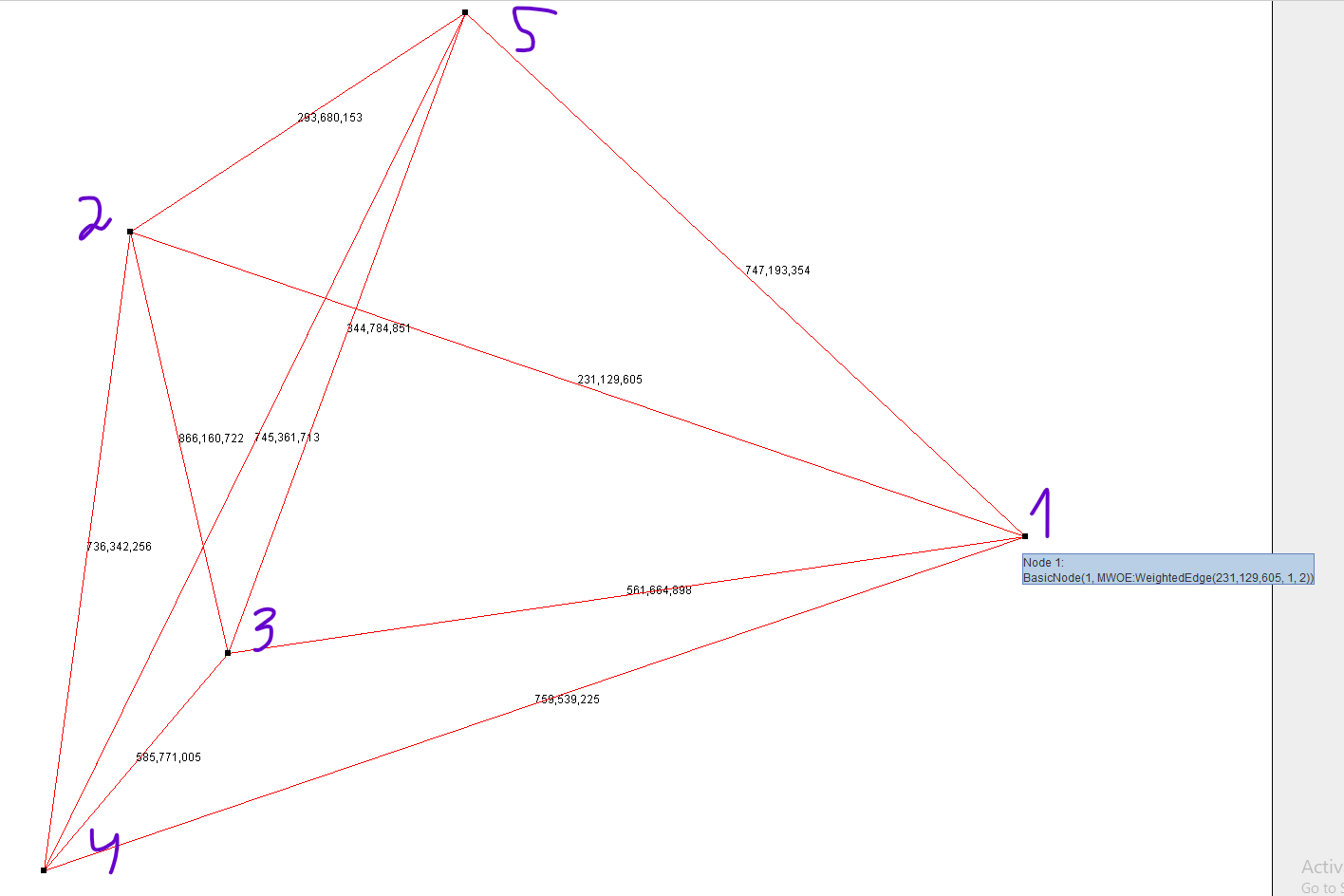
ואפשר לראות שצלע הזו מחברת בין 1 לבין 5 (כלומר הסדר התהפך).

ואפשר לראות שקודקוד 2 מחובר לקודקוד 5, עם MWOE של 227 מיליון, אבל 227 מיליון זה לאו דווקא ה-MWOE של קודקוד 5:



בקיצור זה עובד.

כעת אני מימשתי את ה broadcast של כל ה MWOE:



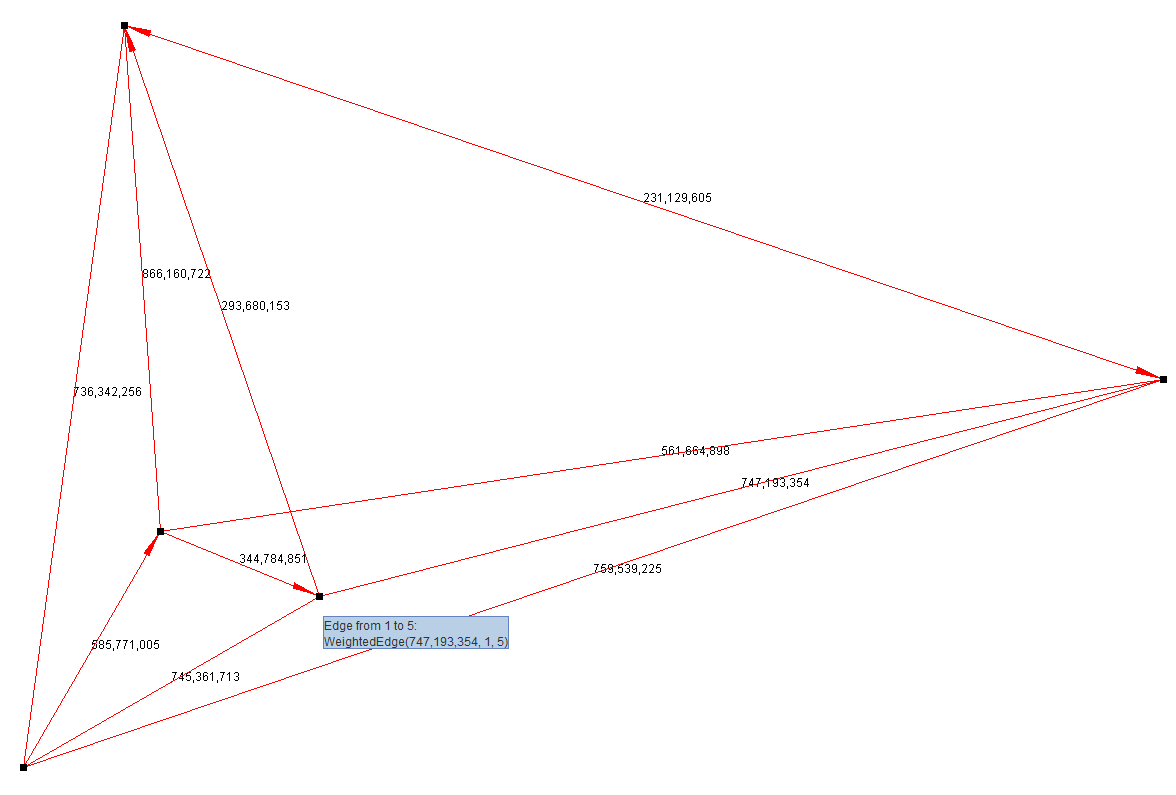
אפשר לראות בלוגים שזה בסדר, כל קודקוד שולח MWOE הנכון לכל השכנים ע"י broadcast (יצרתי גרף בגודל 5 כדי לדבאג יותר טוב):

|  |
| --- |
| 1 got message: MWOEMessage(231,129,605) from: 2  1 got message: MWOEMessage(344,784,851) from: 3  1 got message: MWOEMessage(585,771,005) from: 4  1 got message: MWOEMessage(293,680,153) from: 5  2 got message: MWOEMessage(231,129,605) from: 1  2 got message: MWOEMessage(344,784,851) from: 3  2 got message: MWOEMessage(585,771,005) from: 4  2 got message: MWOEMessage(293,680,153) from: 5  3 got message: MWOEMessage(231,129,605) from: 1  3 got message: MWOEMessage(231,129,605) from: 2  3 got message: MWOEMessage(585,771,005) from: 4  3 got message: MWOEMessage(293,680,153) from: 5  4 got message: MWOEMessage(231,129,605) from: 1  4 got message: MWOEMessage(231,129,605) from: 2  4 got message: MWOEMessage(344,784,851) from: 3  4 got message: MWOEMessage(293,680,153) from: 5  5 got message: MWOEMessage(231,129,605) from: 1  5 got message: MWOEMessage(231,129,605) from: 2  5 got message: MWOEMessage(344,784,851) from: 3  5 got message: MWOEMessage(585,771,005) from: 4 |

הכל תקין. נשים לב שההודעה נשלחת בסיבוב הראשון, בסיבוב השני אני מדפיס את מה שקיבלתי.

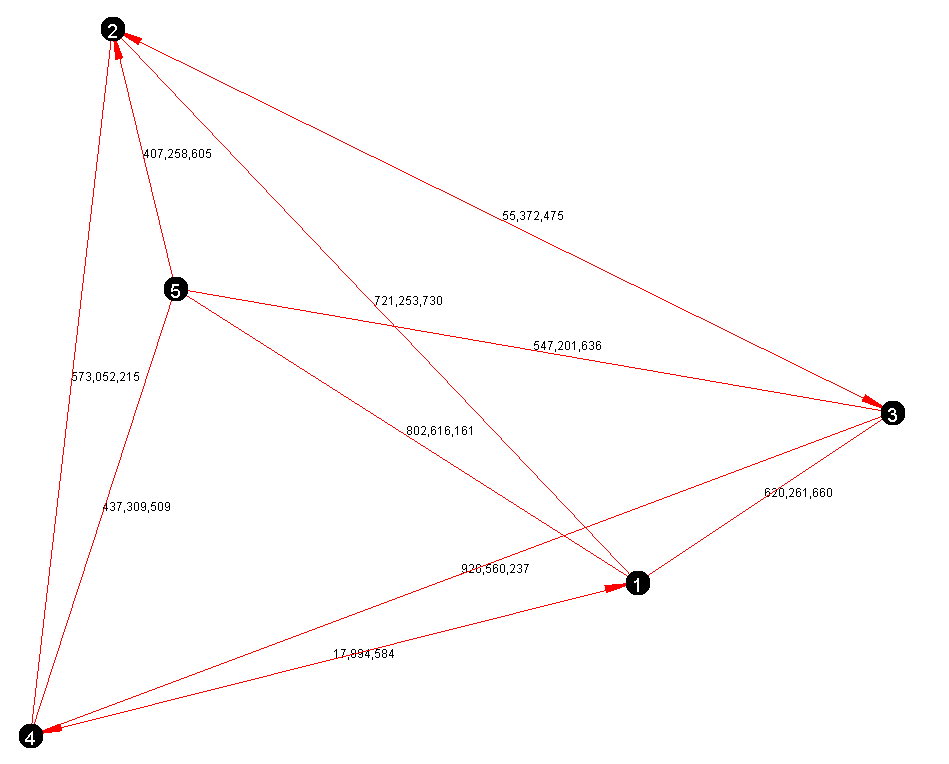
בסיבוב 0 הגרף נוצר והמשתמש בוחר שרת. (לא נשלחים שום הודעות בסיבוב 0).

כעת אני מתעסק בעץ פורש מינימלי. כדי להראות פרגמנטים של עץ פורש מינימלי, החלטתי לצייר את האבות שנבחרו להיות בעץ פורש מינימלי (כלומר קודקוד שחישב MWOE אז הקודקוד שבקצה השני יהיה האבא), וזה נראה כך:



אני הבנתי שהמשקלים שעל הצלעות צריכים להיות unique, ולכן אני מניח שכל המשקלים הם יחודיים.

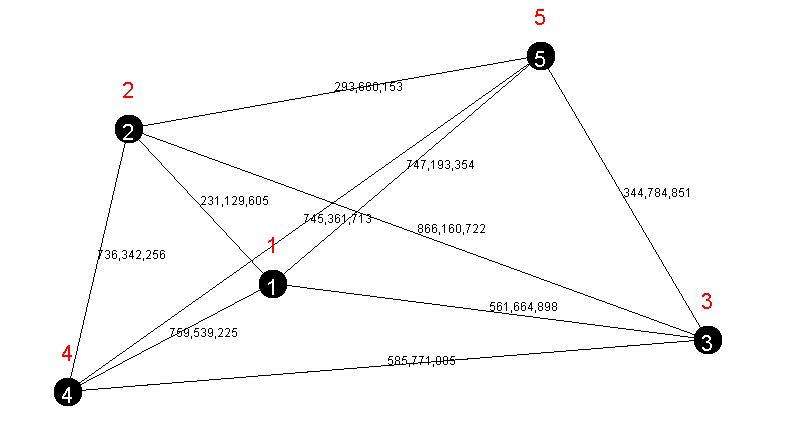
כעת אני צריך להתמודד עם צלע כפולה. כלומר בהינתן MWOE בין A ל-B, הקודקוד A בחר את הצלע {A,B} להיות MWOE, והקודקוד B בחר את הצלע {B,A} להיות MWOE. אז המנהיג שיבחר הוא A אם ה-ID שלו גדול יותר מה-ID של B, או הפוך, B יבחר להיות מנהיג הפרגמנט אם ה-ID שלו גדול יותר מה-ID של A. אפשר לראות ויזואלית מצב כזה:



קודקוד 3 יהיה מנהיג הפרגמנט (צלע {2,3}) והקודקוד 4 יבחר להיות מנהיג הפרגמנט (צלע {4,1}).

נשים לב שהוספתי הדפסה של ה-ID של כל הקודקודים כדי שוויזואלית יהיה קל לראות שהאלגוריתם עובד.

בנוסף הוספתי לקלאס של הקודקוד את הפרגמנט ID שהוא נמצא בו כרגע (בהתחלה כל קודקוד הוא פרגמנט בפני עצמו ולכן פרגמנט ID = ה ID של הקודקוד), אפשר לראות את זה כך:

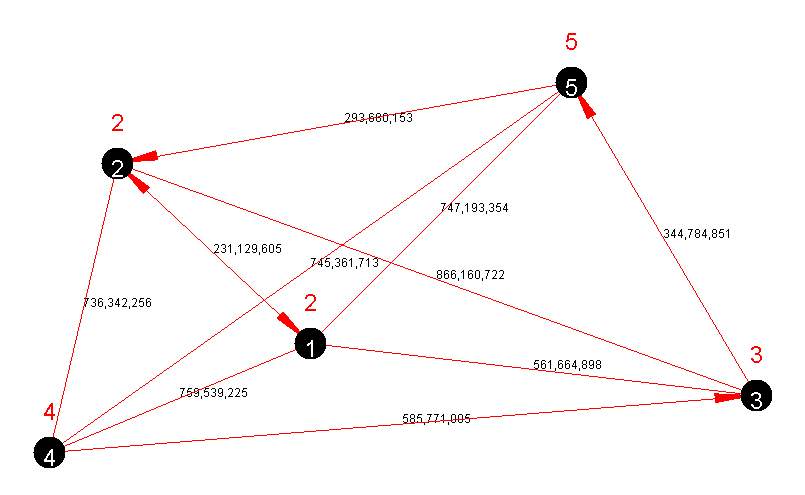


כעת הקודקודים שולחים את ה MWOE שהם בחרו לשכנים:

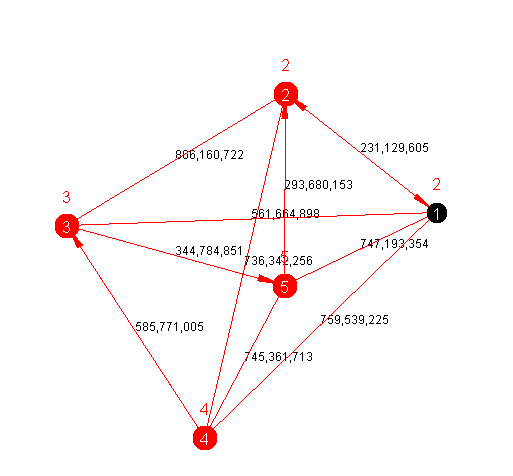
Chart

Description automatically generated

אפשר לראות שהצלע {1,2} נבחרה להיות MWOE של שני קודקודים ולכן מנהיג הפרגמנט יהיה 2 כי הID שלו יותר גדול וזה מה שקורה, הפרגמנט של קודקוד 1 נהיה 2:

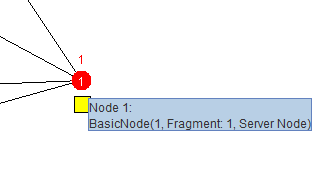


כעת אני יוסיף אפשרות לראות מי המנהיג של הפרגמנט:



אפשר לראות שקודקודים שהם ה-fragmentLeaders מסומנים באדום (highlighted), ואפשר לראות שבסיבוב 2 קודקוד 1 כבר אינו ה-fragmentLeader.

בנוסף הוספתי אינדיקטור מתחת לקודקוד שנבחר כשרת:



האייקון של ריבוע צהוב מסמן שרת. אם מוזיזים את הקודקוד גם האינדיקטור יזוז.

# 28 בפברואר (יום שני)

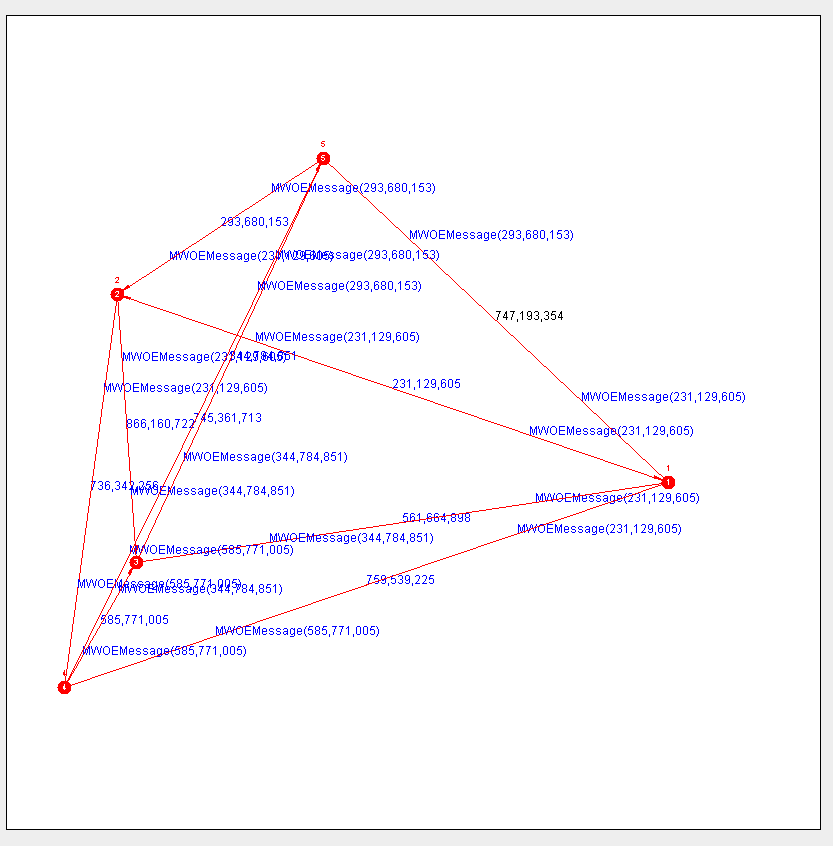
הוספתי ב WeightedEdge אפשרות לצייר את ההודעה שנשלחת על הצלע. כל הודעה שנשלחת מקודקוד A אל קודקוד B, ההודעה תופיע ברבע מהדרך אל B. כלומר מי ששולח הודעה, הטקסט יופיע קרוב יותר לקודקוד השולח.

דוגמה:

A שולח ל-B, ייראה כך:

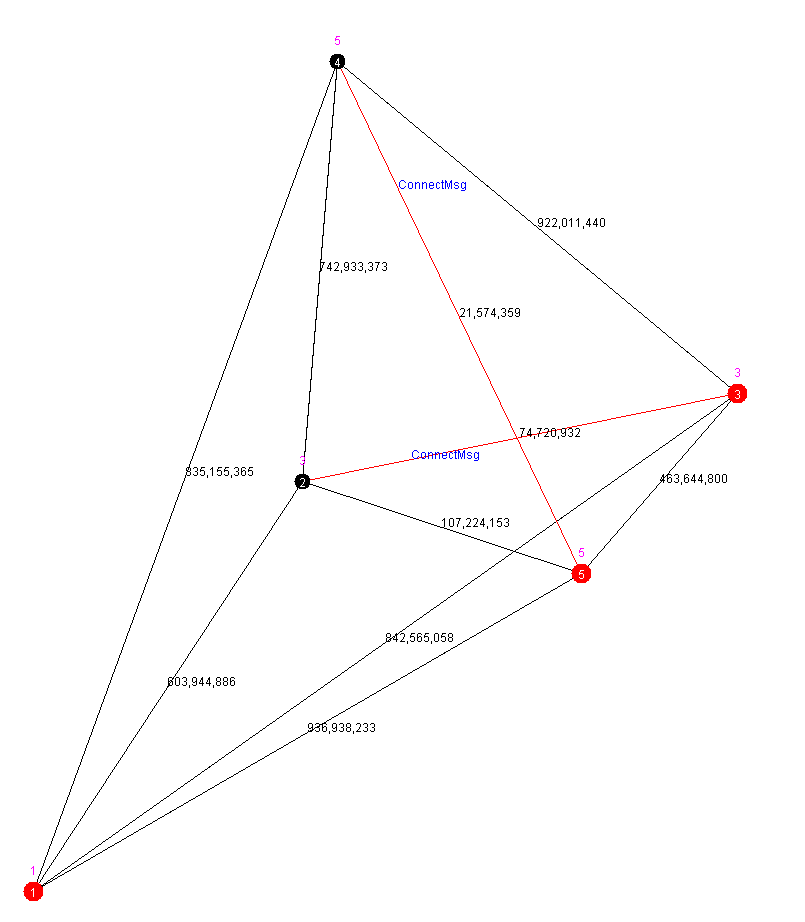
כלומר ההודעה תופיע קרוב יותר ל-A, בערך רבע מהדרך.





אחרי זה סידרתי את הצבע של ה-weight שיהיה שחור ולא כחול כמו ההודעות (לא צריך להראות תמונה, זה תיקון פשוט).

כעת הוספתי מנגנון "ניקוי הודעות" שעל הצלעות, כך שכל סיבוב יראה אך ורק את ההודעה הנוכחית שנשלחת. ככה שאם נשלח הודעה “A” ובסיבוב הבא לא נשלח כלום, אז לא נראה “A” על הצלע:

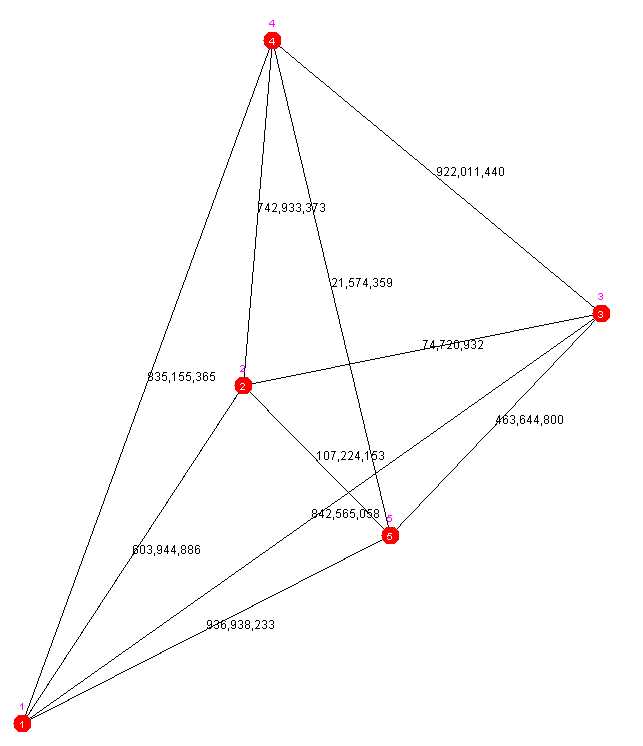


הוספתי סוג הודעה חדש: ConnectMsg אשר קודקוד שרוצה להתחבר לפרגמנט שולח. אפשר לראות לדוגמה שקודקוד 2 מתחבר לפרגמנט 3 (קודקוד 3), וקודקוד 3 נהיה מנהיג בפרגמנט 3. קודקוד 2 נהיה שחור, כי הוא לא מנהיג בפרגמנט 3.

בנוסף שאלתי בפורום שאלות לגבי מה עושים בסיבוב 1,2,3 כי לא ברור לי.

# 1 במרץ (יום רביעי)

הצלחתי לממש את החלק ההתחלתי של איחוד פרגמנטים. גרף התחלתי:



אחרי זה שולחים MWOE:

Chart

Description automatically generated

ואז קודקוד שרוצה להתחבר לפרגמנט שולח הודעה:

Chart, radar chart

Description automatically generated

ואז השורש שולח הודעה בחזרה – הודעת אישור:

Chart, radar chart

Description automatically generated

ורק אז הקודקוד מעדכן את הזכרון הפנימי שלו כך שיהפוך לקודקוד בעץ:

Chart, radar chart

Description automatically generated

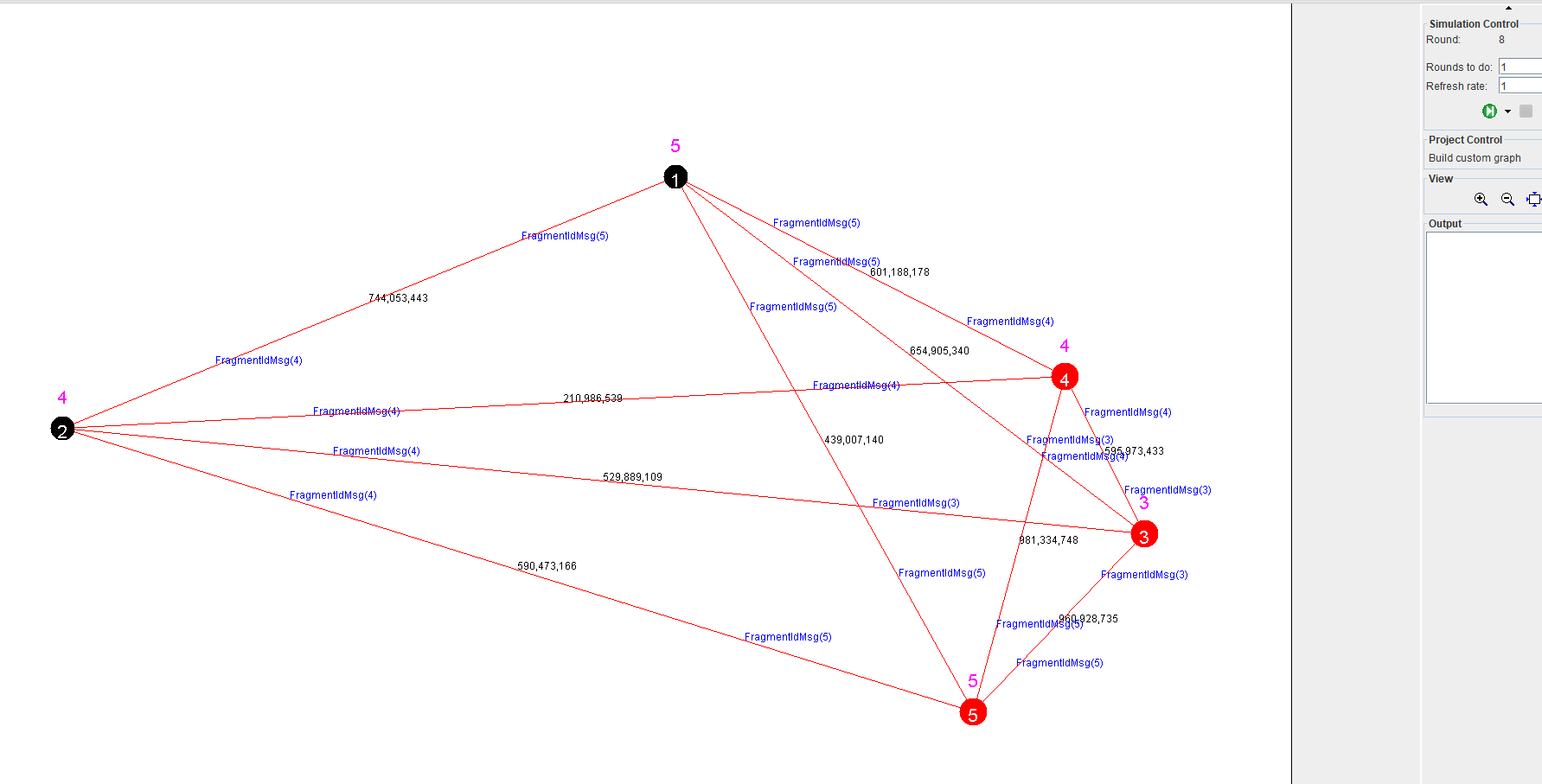
# 3 במרץ (יום שישי)

היום יצרתי מנגנון broadcast עבור ה fragment בלבד.

כל קודקוד שמקבל fragmentBroadcastMsg (אשר מכיל בפנים הודעה מכל סוג אפשרי – כלומר זה כמו מעטפה) הוא ישלח לקודקודים שבאותו הפרגמנט. בנוסף הוספתי מנגנון מניעת לולאות broadcast, מאד פשוט: אם קודקוד מקבל הודעת broadcast, הוא עושה broadcast לה, רק במקרה שלא קיים לו בזיכרון ששולח המקורי של ה broadcast (ה-ID של ה NODE המקורי שהתחיל את ה broadcast) וגם שאין לו את אותו broadcastId . ה-broadcastId מופיע בכל הודעה fragmentBroadcastMsg, והוא לא משתנה כאשר עובר בין קודקודים (למעשה שום דבר לא משתנה בהודעה כזאת). מימשתי את זה ע"י מפה, כל שהמפתחות זה ה id של הקודקודים, והערך בכל תא הוא רשימה של מספרים. רשימת מספרים זו מייצרת את כל ה-broadcastIds של originalSenderId שהקודקוד הנוכחי עשה broadcast. ככה שאם יתקבל הודעה broadcast עם אותו originalSenderId וגם broadcastId שקיים בזיכרון, הקודקוד לא ישלח שוב הודעה זו.

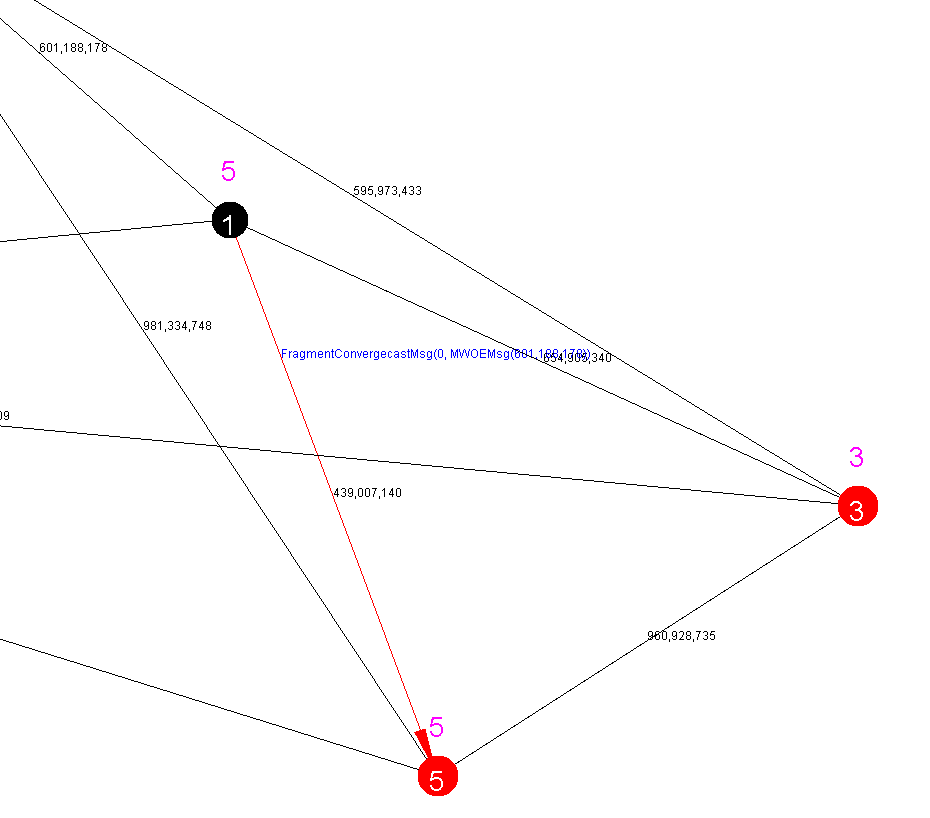
ניסיתי ליצור גרף לדמו אבל לא הצלחתי ולכן אני רושם במילים. אולי כשאתקדם בפרוייקט אצלם דמו.

בנוסף הוספתי תמיכה בפאזה 3 (כל קודקוד שולח את מספר זיהוי הפרגמנט שלו לכל שכניו)



מספר ה round=8 כי N=5 ובסוף סיבוב 2 מסיימים את פאזה 2).

כעת אני מבצע את שלב 5, עושים convergecast ל MWOE החדש שמצאנו (אם קיים כזה):



אפשר לראות שקודקוד 1 מצע MWOE חדש (601 מיליון, זו צלע MWOE תקינה כי מחפשים MWOE שנמצא בפרגמנט אחר עם משקל מינימלי)

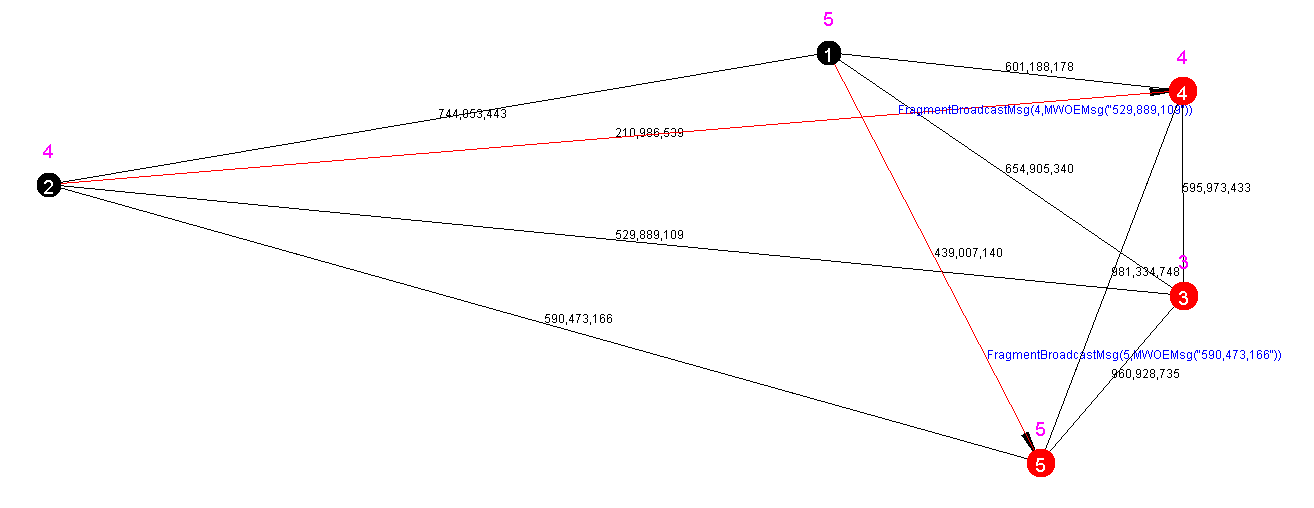
נשים לב שהחזרתי את ה mst\_parent ואת הכיווני צלעות, זה מראה את ה MST, כלומר קודקוד 5 הוא אבא של 4, וגם קודקוד 5 הוא ה leader.

בנוסף הוספתי הרבה logs כך שיהיה קל מאד להבין באיזה פאזה אני נמצא כרגע, איזה הודעות כל קודקוד מקבל, מה הוא עושה בכל שלב, וכו.

# 5 במרץ (יום ראשון) – פאזה 7

הצלחתי למממש מציאת MWOE גלובלי (של כל הפרגמנט) ע"י יצירת convergecast\_buffer אשר שומר את כל ההודעות convergecast (לפחות ה-leader) שקיבל. וכאשר מגיע שלב 6, ה leader עובר על כל ההודעות שהם MWOEMsg, ועושה sort וזה ה-MWOE של כל הפרגמנט.

להלן דוגמה:



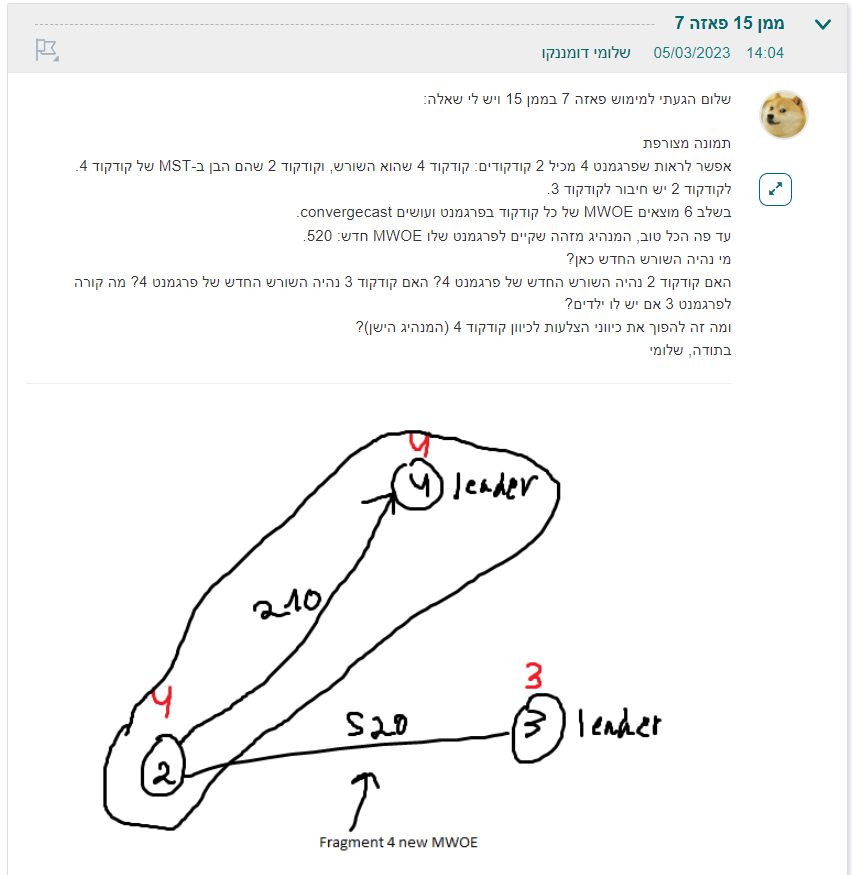
קצת קשה לראות, אבל אפשר לראות שפרגמנט 4 (שמכיל קודקודים 4,2) – ה fragmentLeader (קודקוד 4) שולח broadcast עם MWOE של כל הפרגמנט. המשקל של MWOE זה הוא 529,889,109 וכפי שניתן לראות, קודקוד 2 בחר אותו כאשר עשה convergecast לשורש (צלע {2,3}).

לעומת זאת, אם קיים MWOE של כל הפרגמנט אשר נובע מצלע החל מה-leader עצמו, אפשר לראות בדוגמה הזו, שקודקוד 5 שולח MWOE עם משקל 590 מיליון, כלומר צלע {2,5}. בשלב 5, לא רשום, אבל גם הקודקוד השורש עושה לעצמו convergecast (אני פשוט הוספתי ל convergecast\_buffer, בלי הודעה) את ה-MWOE שהוא מצא.

ודבר אחרון, נשים לב שקודקוד 3 לא שולח אף MWOE כ-broadcast וזה מכיוון שלא קיים MWOE מפרגמנט 3 אל פרגמנט אחר. וזה מכיוון שאין לו ילדים, אשר יעשו convergecast של MWOE משלהם. הצלע MWOE הישנה היא אותו דבר כמו עכשיו. לא השתנה.

בסופו של דבר, שלב 6 עובד.

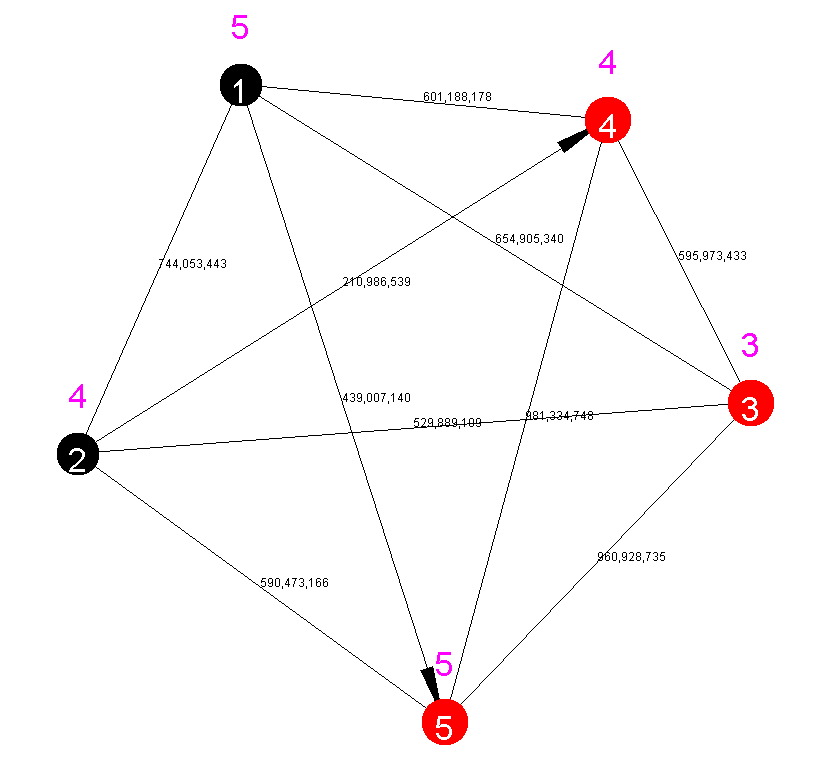
הגעתי לשלב 7, אני שאלתי בפורום הסבר על שלב זה:



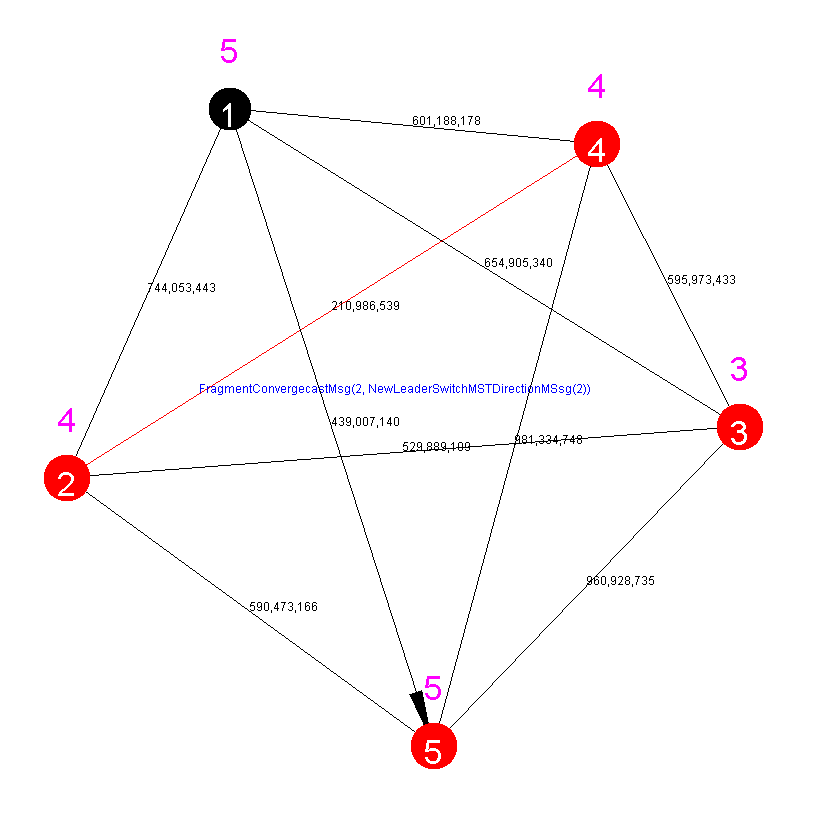
הבנתי את זה כך: נהפוך את קודקוד 2 למנהיג חדש, ונשלח הודעה זו לכל הפרגמנט, כלומר convergecast לשורש. אני מסתבך בחלק של להפוך צלעות. במיוחד בפונקציית ה-draw.

הצלחתי!

להלן המצב שלנו בסיבוב 18 (שזה פאזה 6):



כעת מגיעים לסיבוב 19 (פאזה 7 מתחיל):

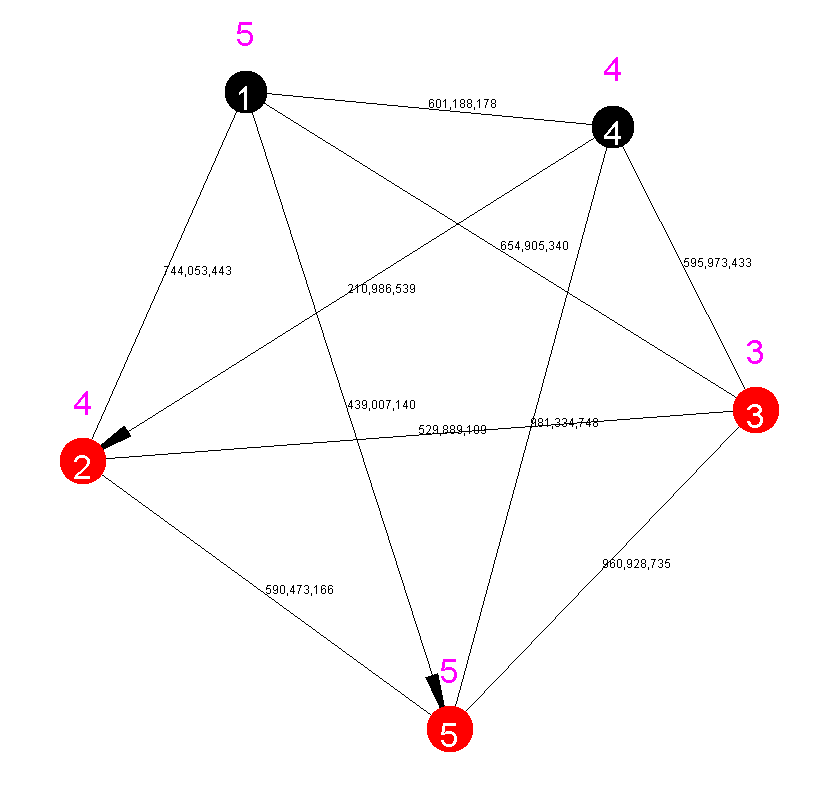


מה קרה כאן? קודקוד 2 יש לו MWOE (צלע {2,3}) והשורש החליט שזה נכון.

כעת קודקוד 2 קיבל הודעה בפאזה 6 שהוא יהפוך לשורש החדש.

אז בסיבוב הראשון של פאזה 7, קודקוד 2 הופך (לוקאלית בינתיים) ל fragment leader, הוא מוחק את היכוון של MST PARENT (כלומר הוא ה LEADER החדש ולכן אין לו mst leader), ולכן הכיוון של הצלע נמחק.

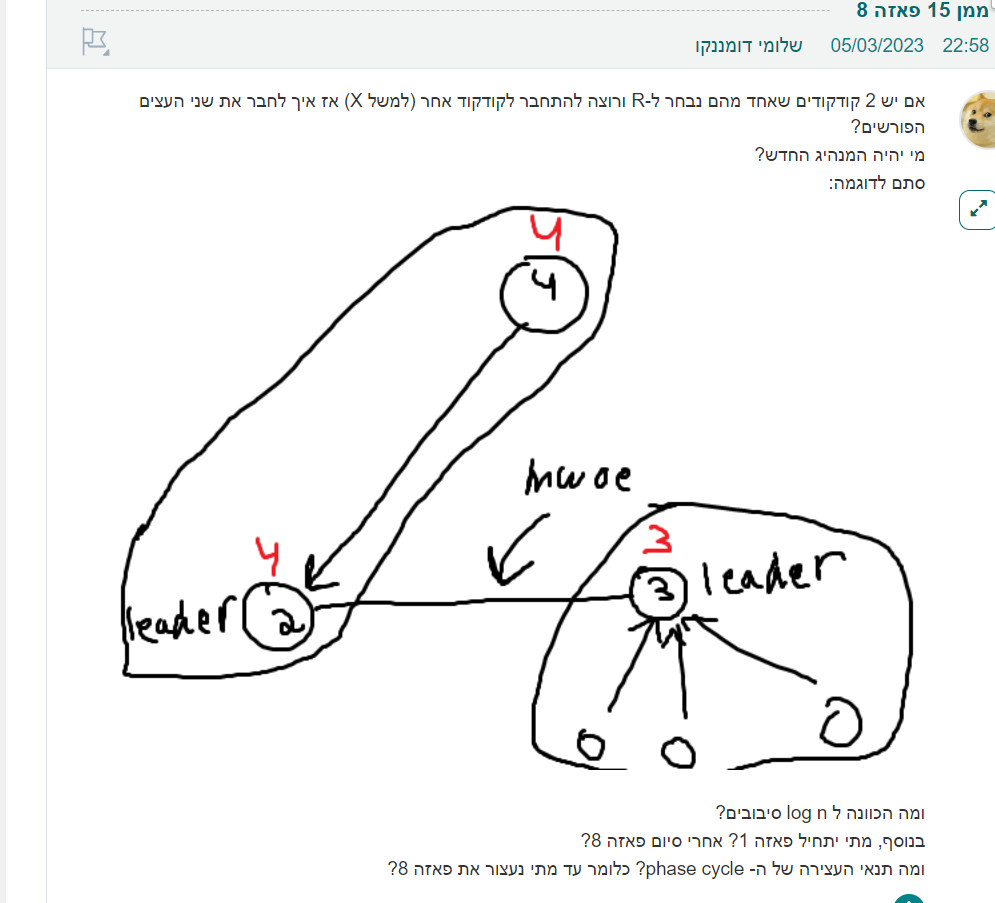
כעת קודקוד 2 שולח הודעה שהוא המנהיג החדש והקודקודים בפרגמנט צריכים להחליף כיוונים. סיבוב הבא:



וזה מה שקורה, קודקוד 4 היה המנהיג וכעת הוא הילד של המנהיג החדש, הקודקוד 2.

כעת אני צריך למממש את פאזה 8 אשר מגדיר את 3 להיות גם בפרגמנט 4 (בגלל MWOE).

שאלתי בפורום שאלה:



בינתיים מפסיק להיום.

# 6 במרץ (יום שני) – פאזה 8

בשביל פאזה 8 יצרתי סוג הודעה חדש: ConnectFragmentsMsg אשר קודקוד r בפאזה 7 שולח ל MWOE שלו בשביל לחבר פרגמנטים (ולחבר MST).  
סיום פאזה 7:

Chart, radar chart

Description automatically generated

התחלת פאזה 8:

Chart

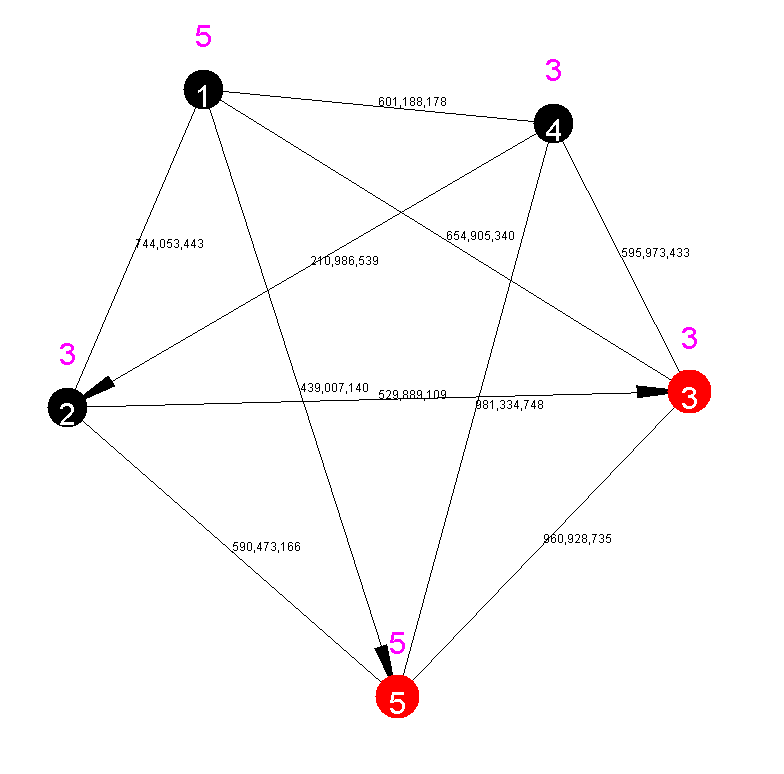
Description automatically generated

קודקוד 2 רוצה להתחבר לקודקוד 3 בגלל GLOBAL MWOE. מה שאומר שרוצים לחבר פרגמנט 3 ופרגמנט 4. (ולכן קודקוד 2 שולח הודעה ConnectFragmentMsg). (נשים לב שקודקוד 2 הפרגמנט שלו היה 4, אבל בסיבוב זה, הוא הפך ל-3).

קודקוד 2 משנה את ה fragmentId שלו ל-3, והופך את קודקוד 3 לאבא שלו (אפשר לראות כיוון חיצים).

בנוסף, באותו סיבוב, קודקוד 2 שולח broadcast לקודקודי הפרגמנט 4 כדי שיעדכנו עצלם גם את ה fragmentId וגם את ה leaderId.

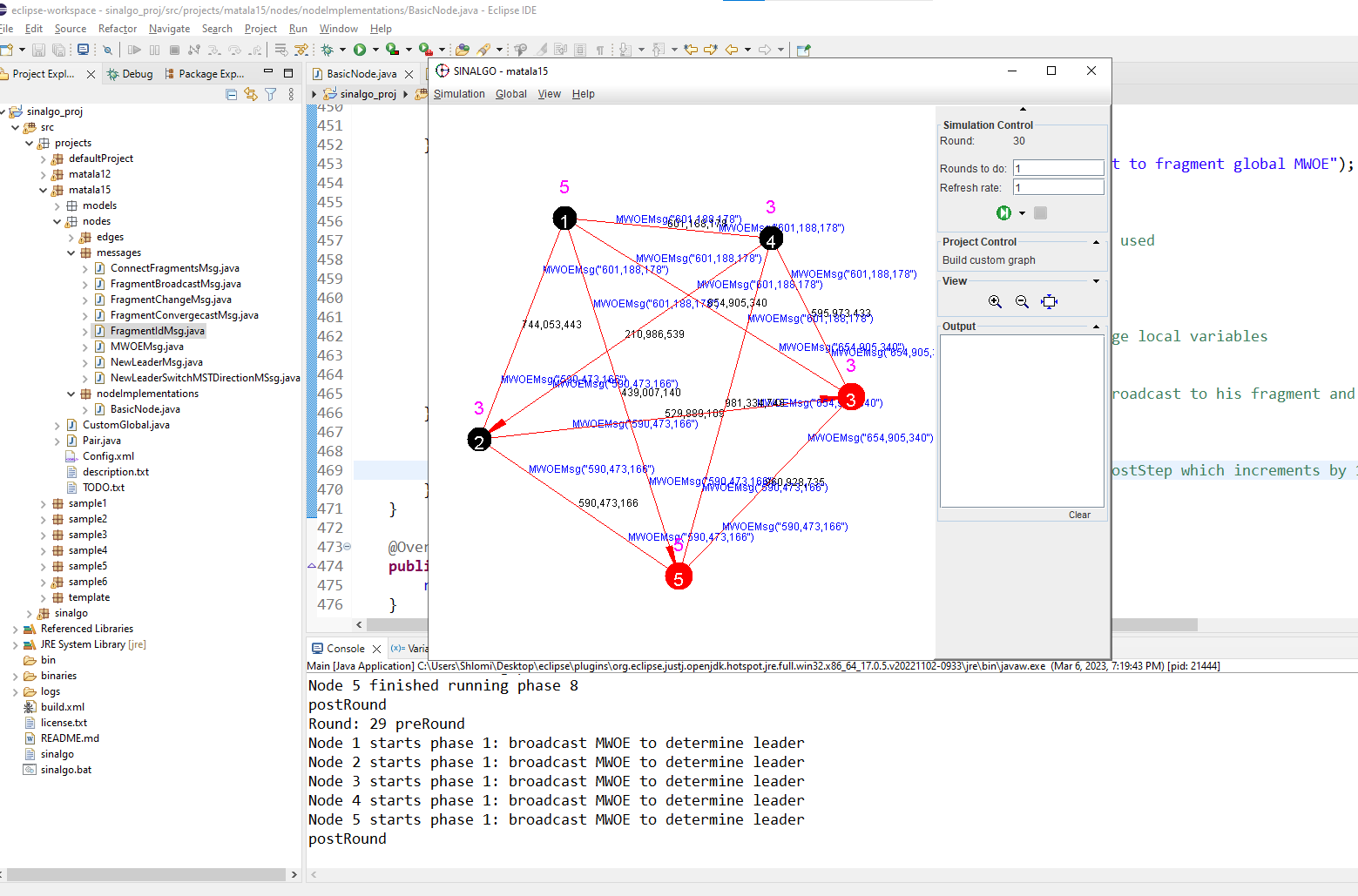
וזה הסיבוב הבא:



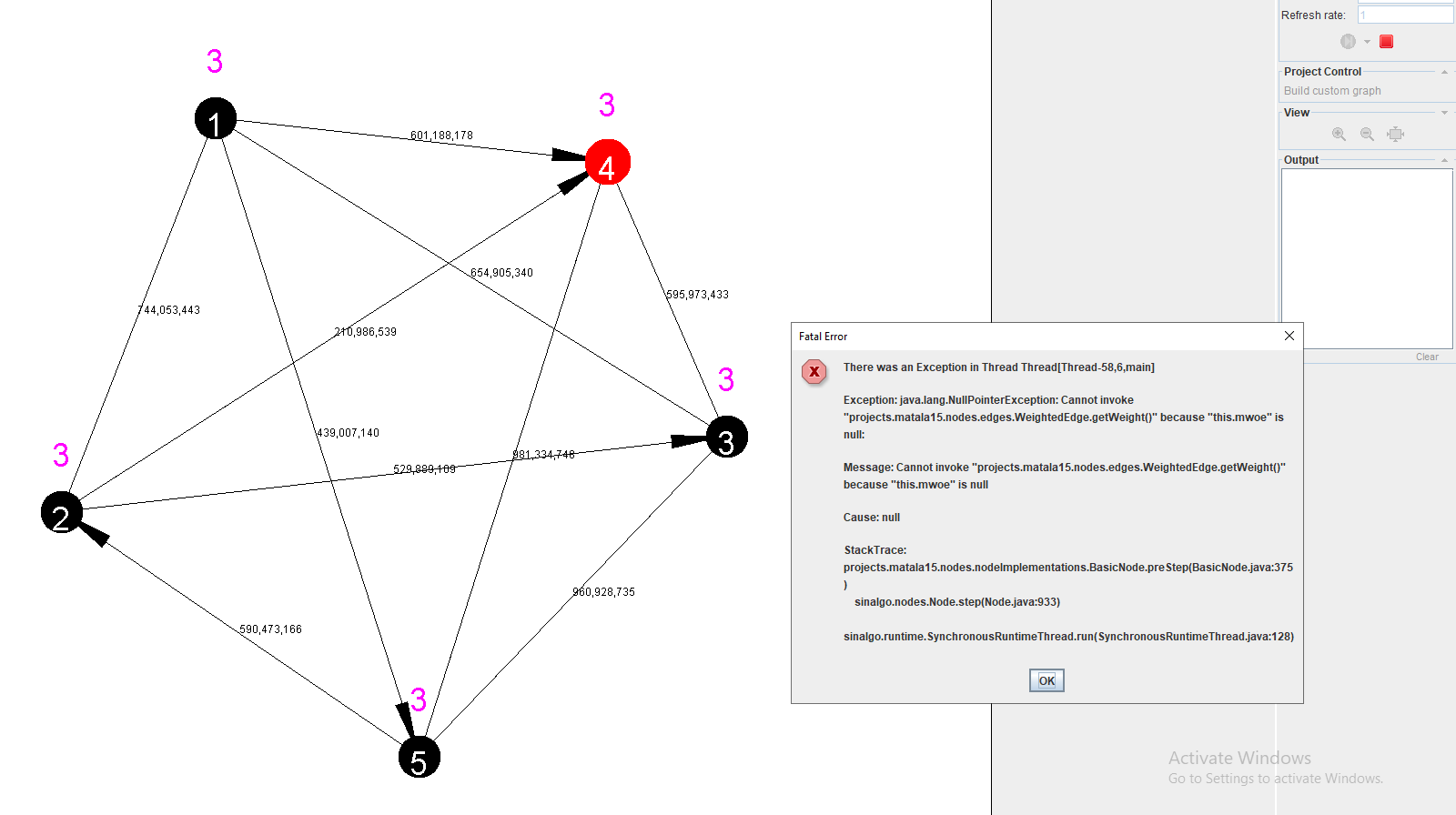
אפשר לראות שגם קודקוד 4 שינה את הפרגמנט שלו ל-3.

נשים לב שחיבור פרגמנטים – לא צריך לשנות כיווני צלעות של MST – חוץ מ 2 קודקודים שהם רוצים להתחבר. כיווני הצלעות בתוך הפרגמנט לא משתנה.

ובסוף כשמסיימים פאזה 8, מגדירים את ה-roundNum=0 (שזה משתנה לוקאלית עצל כל קודקוד אשר יודע מתי מתחיל\מתסיים פאזות) ועכשיו מתחיל פאזה 1 מהתחלה:



רציתי להתעניין מה קורה אם אני ממשיך להריץ את האלגוריתם – האם הצלחתי ליצור גרף MST שלם? כן:



אפשר לראות שכל הקודקודים הם נמצאים בפרגמנט 3!

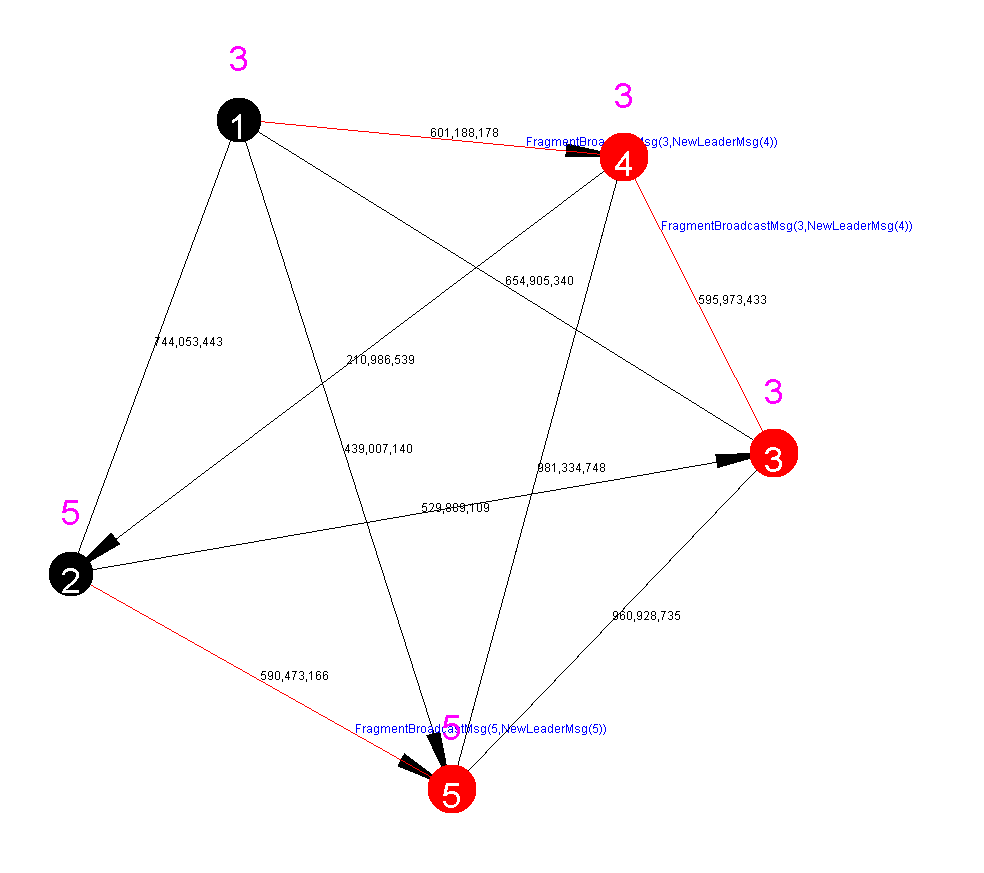
עכשיו זה קורס כי לא בדקתי מתי כל הקודקודים נמצאים באותו פרגמנט (צריך ליצור פונקציה שרצה אחרי שלב 8 שבודקת שכל הקודקודים הם נמצאים באותו MST).

ואז זה אומר שסיימתי!

# 8 במרץ (יום רביעי)

החלטתי שכדי שקודקוד ידע מתי לעצור את האלגוריתם צריך לספור כמה קודקודים יש בפרגמנט, וכאשר כל הקודקודים בפרגמנט, כל הקודקודים ידעו בסוף פאזה 8 מתי לעצור.

אך לפני כן אני בודק את מהלך האלגוריתם אחרי פאזה 8, אני רואה אחרי פאזה 1 ששולחים MWOE, קודקוד 2 מתחבר באופן לא נכון לקודקוד 5:

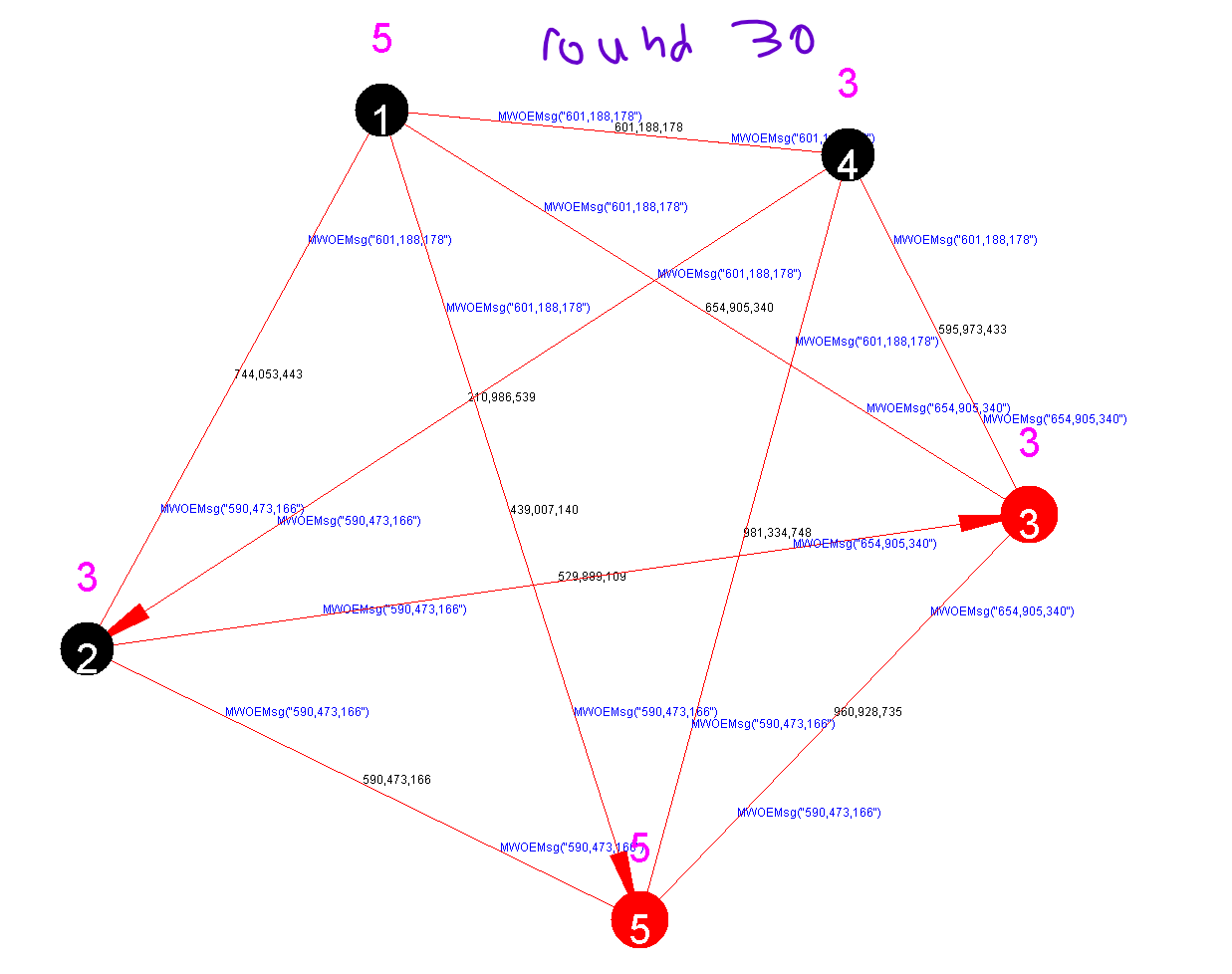


אפשר לראות שלקודקוד 2 יש 2 אבות?!

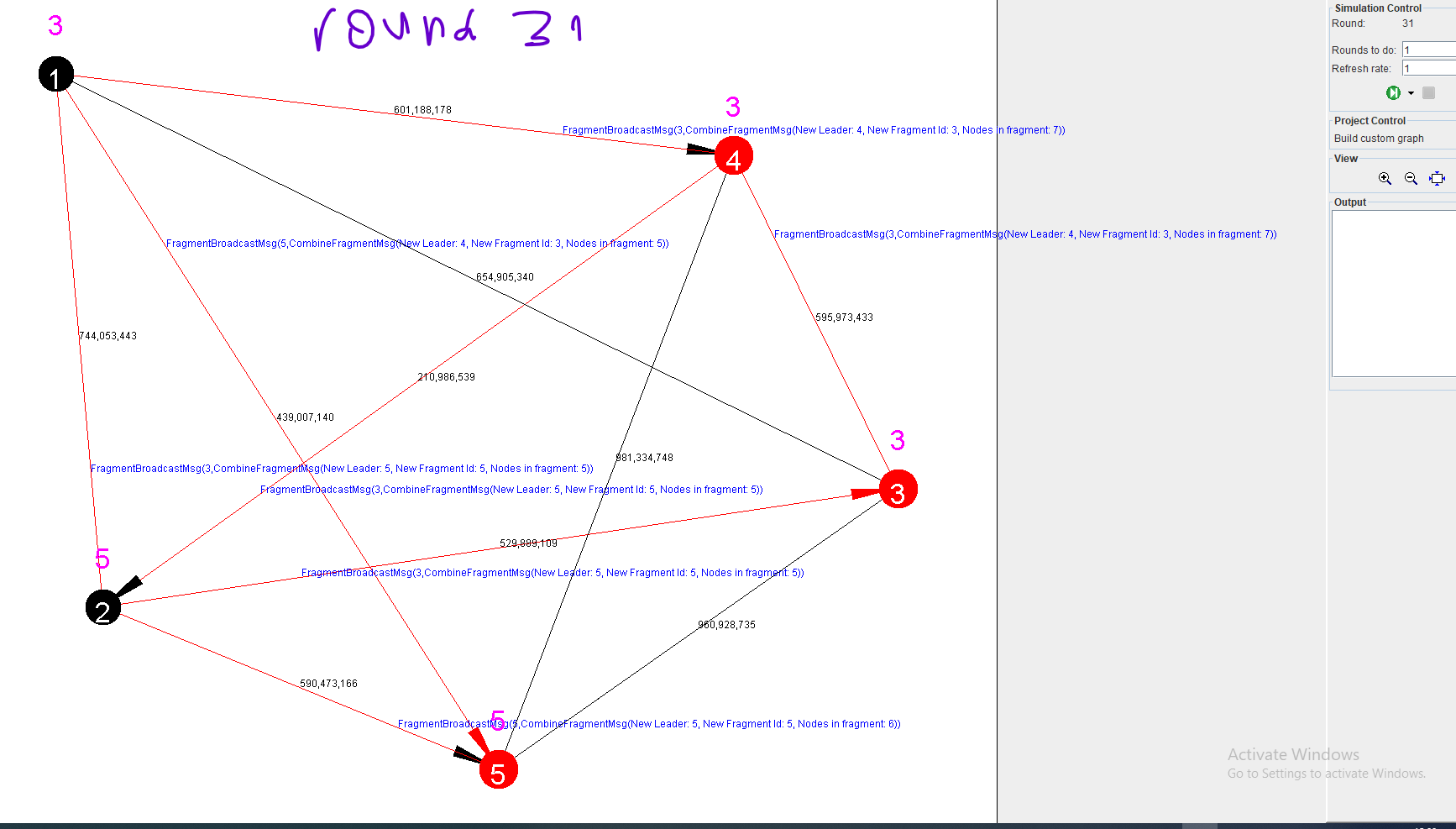
מנסה לפתור בעיה זו.

החלטתי לנסות לפשט את הקוד, החלפתי הודעות והוספתי לכל קודקוד totalNodesInFragment אשר בפאזה 8 נדע מתי לעצור את האלגוריתם, אם totalNodesInFragment == N.

הצלחתי למממש את ה-totalNodesInFragment. עכשיו אני חוזר לבעיה המרכזית, שלב 30:

\

הכל בסדר פה. אבל בשלב 31 (פאזה 2):



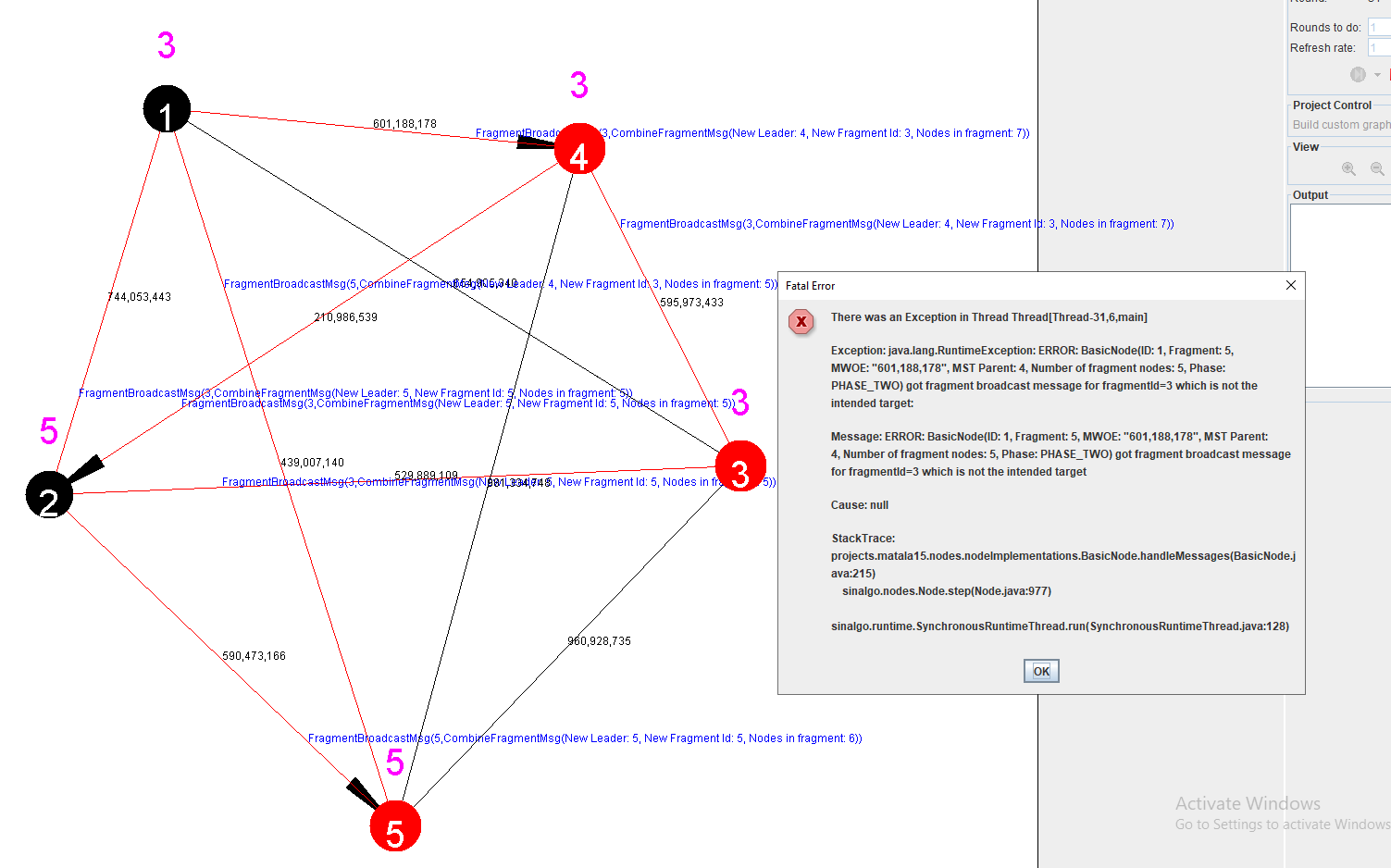
אפשר לראות שקודקוד 1 כאילו יש לו 2 אבות (קודקוד 5 וקודקוד 4).

קודקוד 5 היה אבא של קודקוד 1 לפני כן, אבל כעת קודקוד 4 הצטרף להיות אבא של 1.

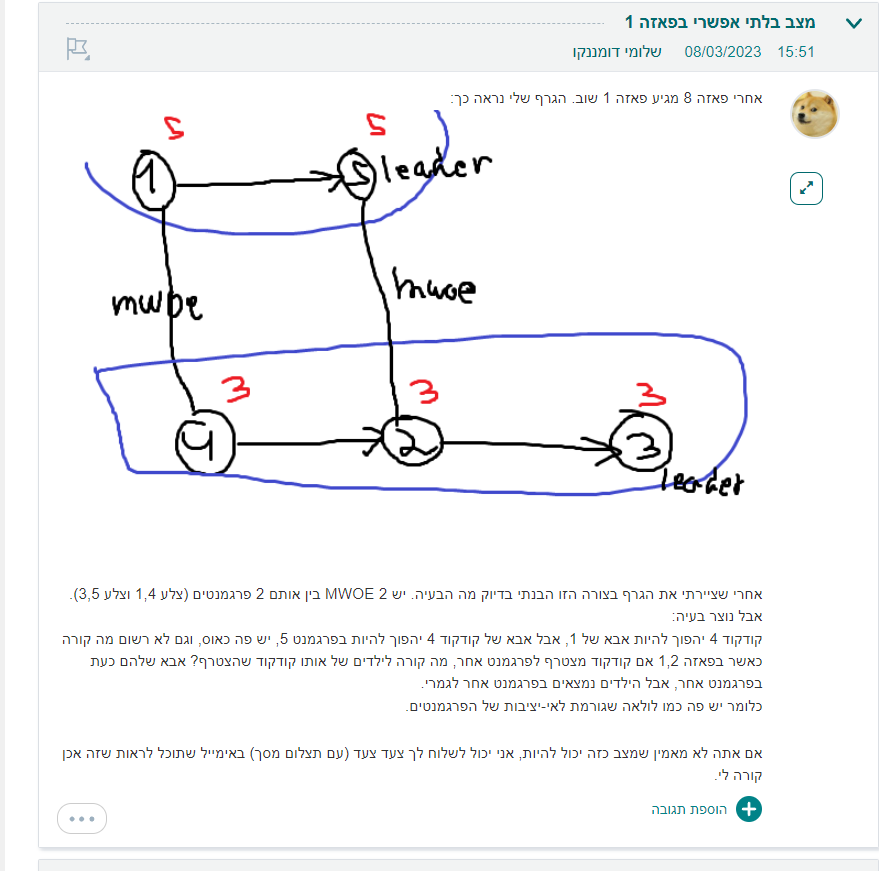
זה קורה בגלל שצלע 1,4 נבחרה להיות MWOE ולכן לפי פאזה 1,2 קודקוד 4 צריך להיות אבא של קודקוד 1.

וזה באמת מה שקורה, אבל מסתבר שפשוט לא עדכנתי את ה-GUI בכך.

הצלחתי בערך לתקן את הצלע 1,5 אבל בשלב הבא אני נתקבל בבעיות:

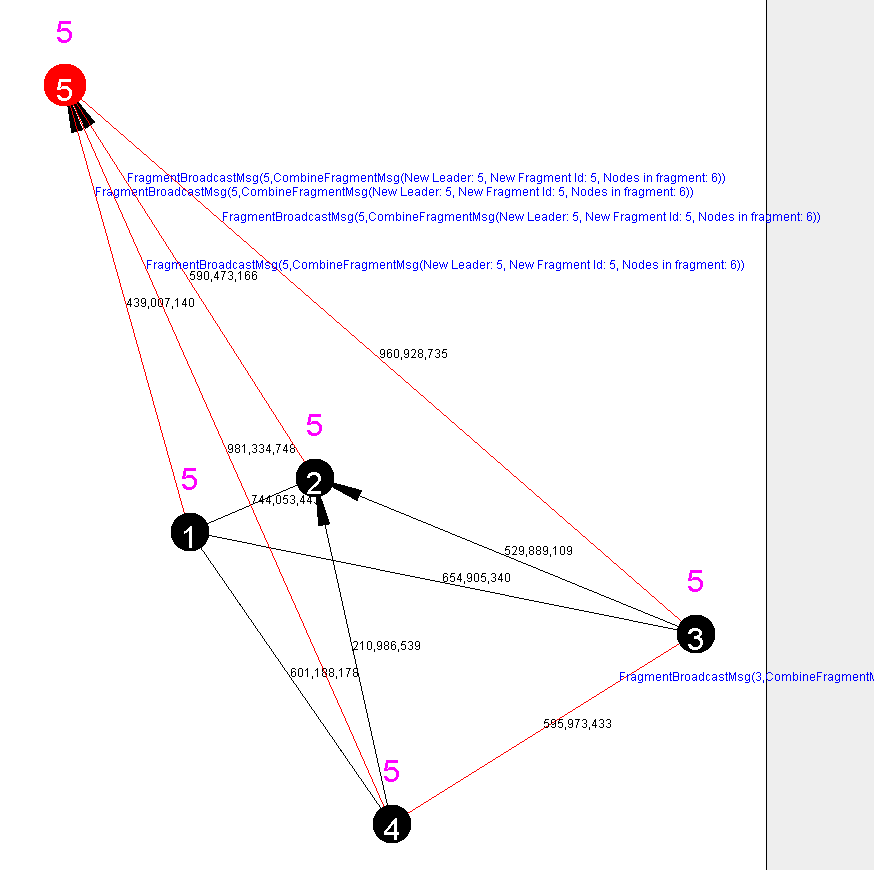


שלחתי על זה בפורום את הבעיה:



החלטתי שבמקום להתחיל פאזה 1 אחרי סיום פאזה 8, אני יתחיל פאזה 4 במקום זאת. זה הרבה יותר הגיוני.

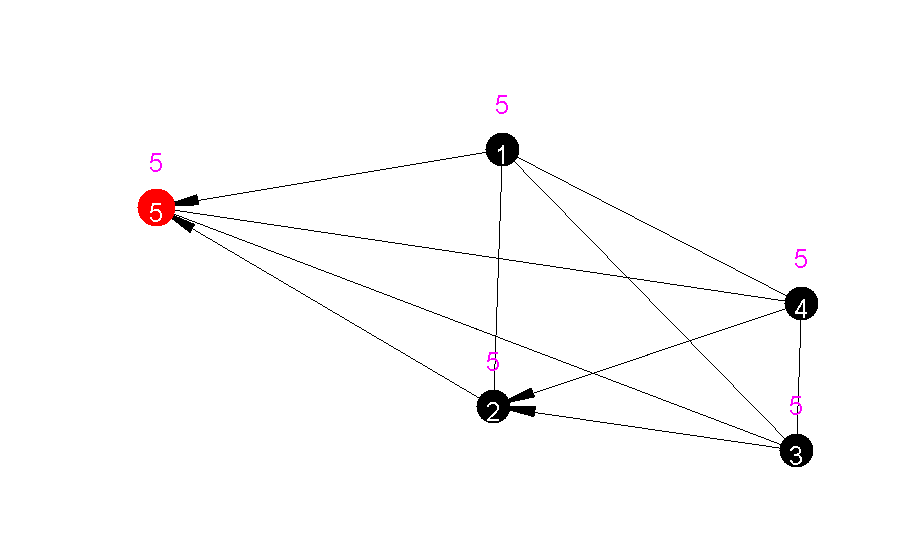
עשיתי הרבה שינויים ובקוד ועכשיו בסיבוב 46 (הסיבוב הכי עמוק שלי) הצלחתי לקבל גרף כזה:



יש רק בעיה קטנה: מספר הילדים יצא 6, במקום 5. בנוסף אפשר לראות שקודקוד 3 שולח הודעה בפרגמנט הישן שלו (פרגמנט 3) כהודעה לקודקוד 4, ואז זה קורס (עשיתי בדיקה שאם מקבלים הודעת BROADCAST אבל ה INTENDED FRAGMENT ID שונה, עושים RuntimeException).

אחרי בדיקה מעמיקה, מסתבר שהכל בסדר, בגלל שקודקוד 3 וקודקוד 4 משנים באותו סיבוב את הפרגמנט שלהם, צריך להתעלם מהודעות broadcast של פרגמנטים אחרים (כי קודקוד 3 לא יודע שבאותו סיבוב, קודקוד 4 יחליף את מספר הפרגמנט שלו מ-3 אל 5). ולכן פשוט מתעלמים ולא צריך לזרוק שגיאה.

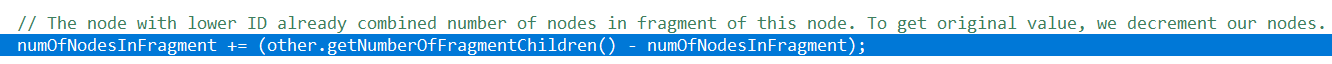
אחרי הרבה שעות של debugging, הצלחתי לקבל עץ מינימלי:



ה-breakthrough היה להריץ פאזה 4 במקום פאזה 1 אחרי סיום פאזה 8.

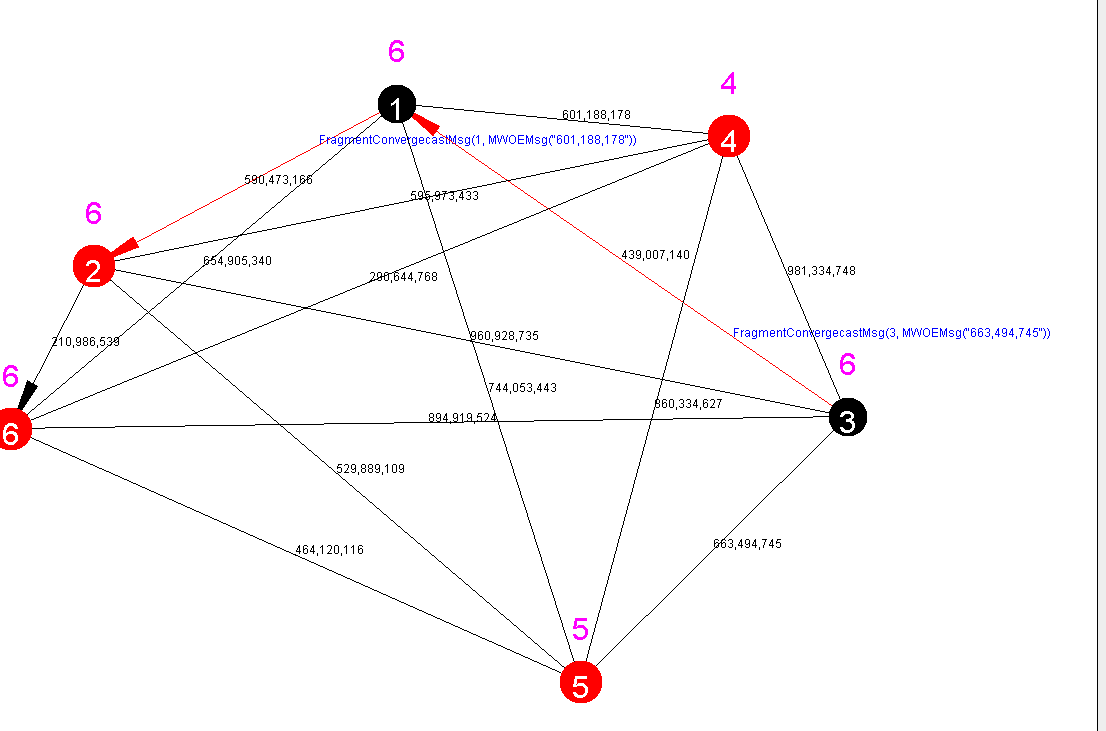
בנוסף אחרי שהאלגוריתם מסתיים, אני עוצר ב -hasTerminated ע"י לשאול כמה קודקודים הפסיקו לרוץ (לכל קודקוד יש משתנה שאומר האם הפסיק לרוץ עקב מספר הקודקודים בפרגמנט).

בנוסף החלק של מספר הילדים, הצלחתי להסתדר ללא הצבת ערכים בסיבובים הבאים:



מה שעשיתי זה בעצם שהקודקוד עם ה-ID הנמוך יותר, המספר שלו מתעדכן לפני הקודקוד עם ה-ID הגבוהה, אז חשבתי שאפשר להוריד את הערך שהוספתי, כאשר הקוד מגיע לשורה עם ה-ID הגדול יותר.

כעת ניסיתי להריץ את האלגוריתם על 6 קודקודים במקום 5 אבל רואים שיש 2 קודקודים בפרגמנט 6 שהם השורש, שזה לא הגיוני, אני מנסה לתקן:



# 9 במרץ (יום חמישי)

מסתבר שפשוט קראתי לפונקציה הלא נכונה בשביל לעדכן את ה leaderId, קודקוד 6 כן עדכן את ה leaderId שלו אבל לא ל-2 אלה ל -6 (מספר הפרגמנט). כלומר, זה היה:

fragmentLeaderId = msg.getNewFragmentId();

ועכשיו זה:

fragmentLeaderId = msg.getNewLeaderId();

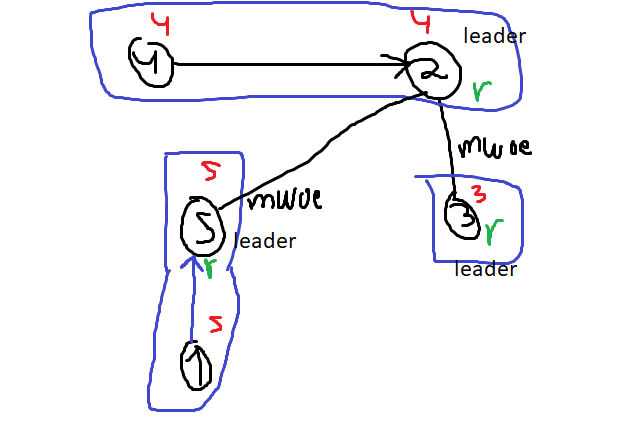
בנוסף קודקוד 6 לא משנה את הכיוון צלעות שלו אל עבר קודקוד 2. מנסה לפתור בעיה זו.

תיקנתי, סיבוב 32:

A picture containing chart

Description automatically generated

שאלתי בפורום, יש לי בעיה בלתי אפשרית:



אני כרגע עובר למרכיב 2.

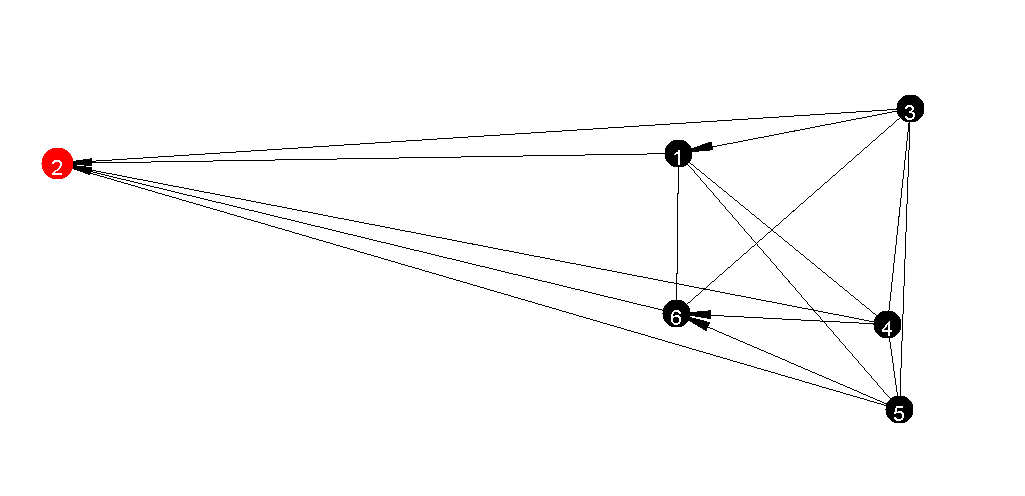
# 12 במרץ (יום ראשון)

החלטתי לכתוב מחדש את כל הקוד של כל הפרוייקט. משהו לא מסתדר לי – ה MST יוצא לא נכון.

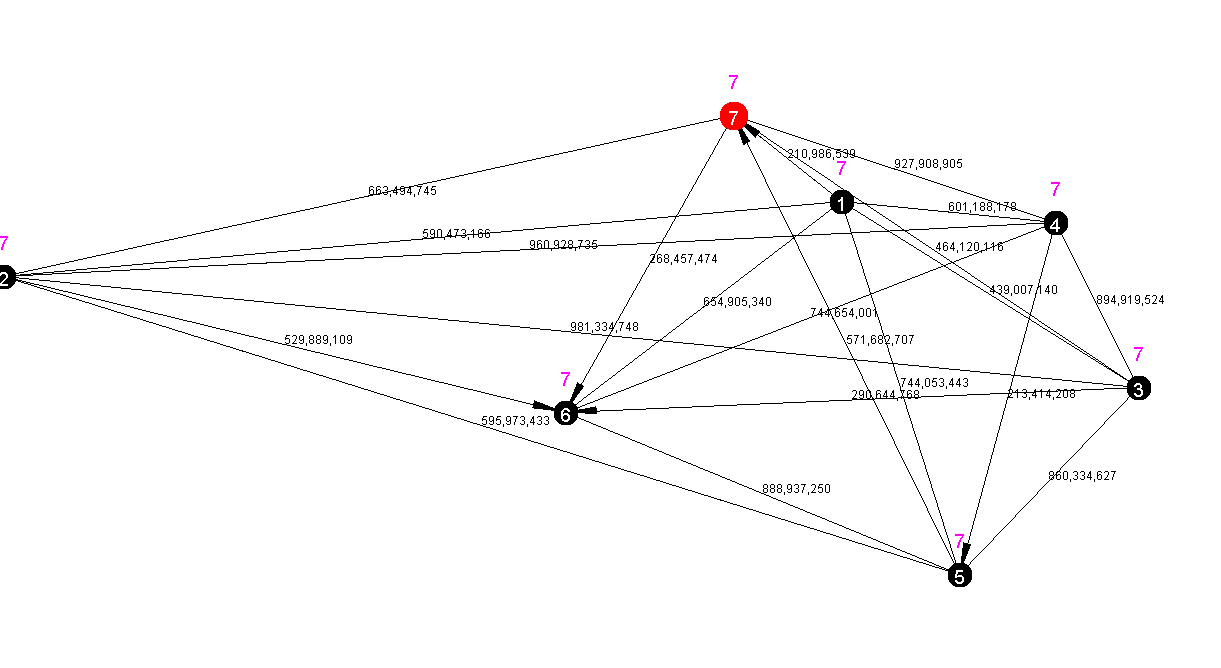
# 14 במרץ (יום שלישי)

נוצר לי MST תקין כאשר הגרף מכיל 3,4,5 קודקודים, אבל קורס כאשר יש 6. מנסה לתקן.

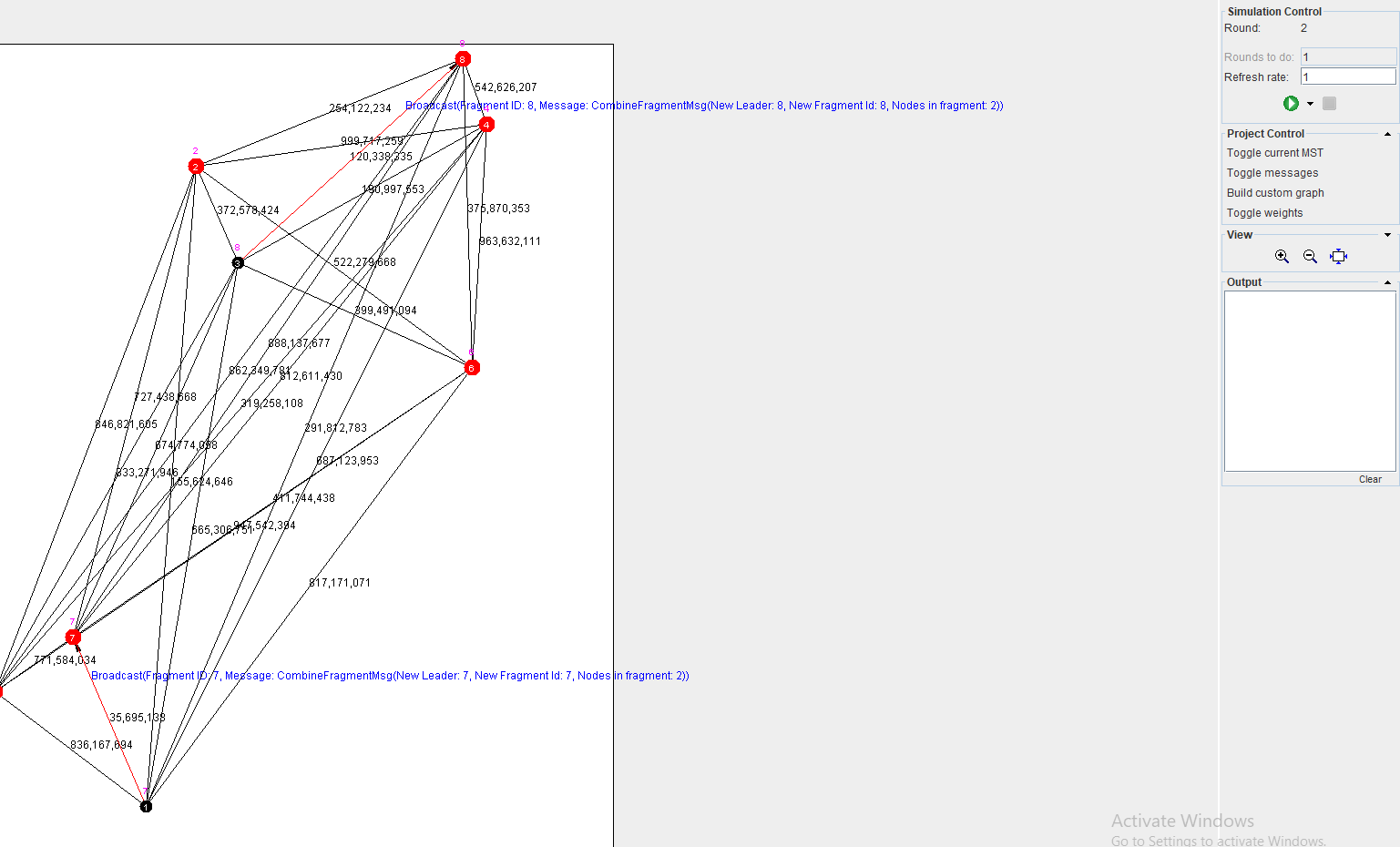
הצלחתי, הבעיה הייתה פשוט ששחכתי לשים mst\_parent = null בפאזה 7. כעת מוצא בגרף עם 6 קודקודים:



והאלגוריתם שלי עובד כמעט עם 7 קודקודים, רק בעיה קטנה, שמספר הילדים בפרגמנט אינו תקין (4 ילדים, למרות שאפשר לראות שכל הקודקודים בפרגמנט):



כדי להקל עליי הוספתי כלים לשינוי ה-DRAW של צלעות, הודעות, ומשקלים:



אם רוצים לראות רק את ה-MST:

Chart

Description automatically generated

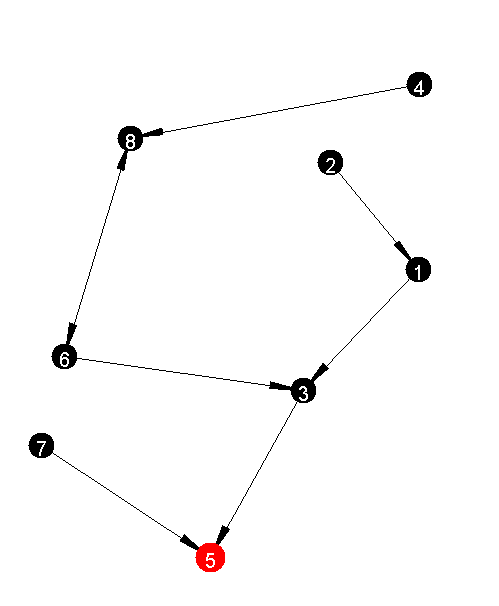
אפשר לעשות toggle למשקלים, MST, הודעות, בכדי כשאני עושה DEBUGGING לא צריך לחפש באלפי צלעות.

תמונה של הכפתורים:

Graphical user interface, text, application, chat or text message

Description automatically generated

סדרתי את הבעיה שקורס כאשר יש 8 קודקודים בגרף (המרה של הודעות לא תקינה):



הקוד שלי כעת עובד גם עם 9 קודקודים בגרף. (הוא קרס עם 9 אבל תיקנתי, קרס בגלל ש mwoe היה null ואני בדקתי את ה mwoe.weight. תיקון מהיר)

אחרי זה גיליתי שעבור 8 קודקודים יש צלעות שהם double directed, אבל תיקנתי ע"י:

replaceMSTParentDirection(sender); // Its important to first remove old connection and only then update mst parent (my mistake)

mst\_parent = sender;

בעצם החלפתי MST קודם ורק אז החלפתי DIRECTION. (כמובן זה ויזואלי [הפונקציה של להחליף direction], זה לא משפיע על האלגוריתם בכלל)

הרצתי את הגרפים שמכילים 3 ועד 15 קודקודים, בדקתי כל אחד ואחד, והכל תקין!

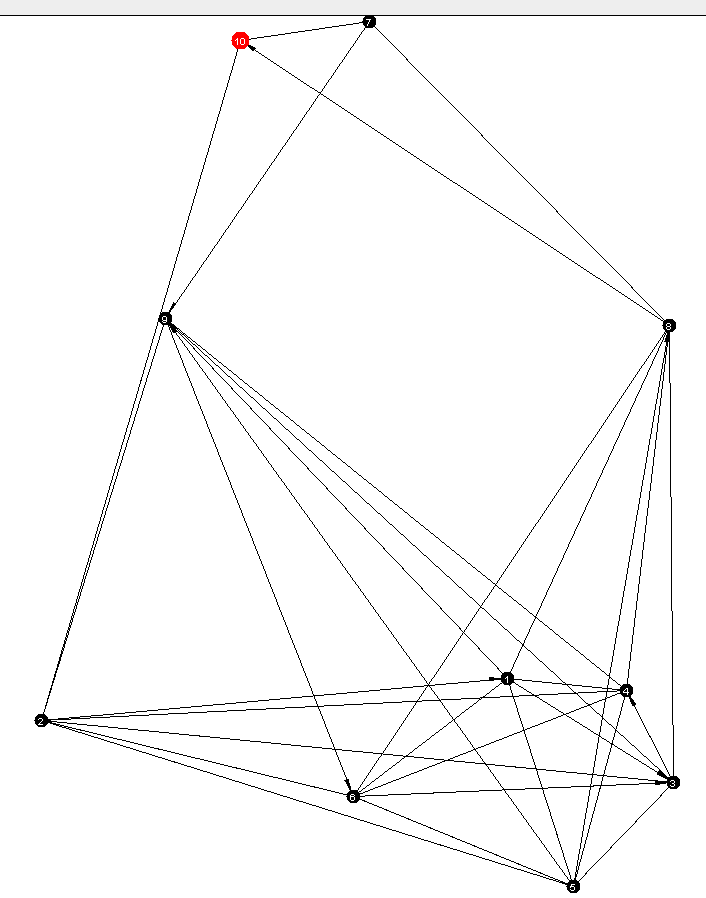
יצרתי תיעוד של הפרוייקט שלי ביוטיוב:

<https://www.youtube.com/watch?v=_5p9-792u5g>

נשאר לי לממש את החלק הבא:

"בהמשך, להפוך את השרת לשורש החדש בעץ ולהפוך כיווני צלעות מסוימות, כך שכל המסלולים בעץ יובילו אל השרת."

הצלחתי לממש בדי פשטות את שלב זה ע"י שליחת convergecast של להפוך כיווני צלעות, כפי שעשיתי בפאזה 7. הנה דוגמה, עם 10 קודקודים, אחרי סיום GHS, כאשר לא בחרנו שרת:



ה – LEADER הוא קודקוד 10, וכעת אם אני בוחר את קודקוד 3 להיות השרת אחרי GHS האלגוריתם באופן אוטומטי מריץ את שלב של הפיכת צלעות אל עבר שרת קודקוד 3 (זה פאזה 9 עבורי, פאזה חדשה):

A close-up of a power line

Description automatically generated with low confidence

כעת קודקוד 3 הוא גם השרת וגם הצלעות פונות אליו:

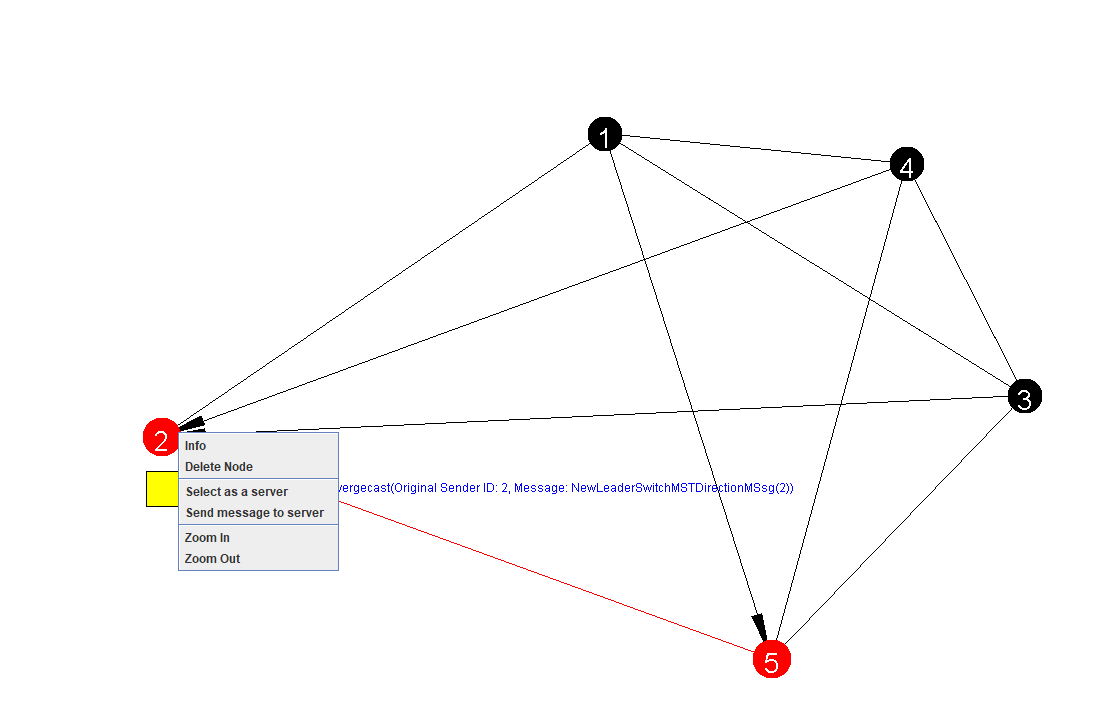
Diagram, engineering drawing

Description automatically generated

כעת נשאר לי לעשות:

"כעת, יש לאפשר משלוח הודעות מלקוח הנבחר על ידי המשתמש, אל השרת ובחזרה."

מימשתי את החלק שמשתמש מכניס הודעה ל textbox וכל קודקוד יכול לשלוח לשרת הודעה:



קודקוד 5 ישלח הודעה לקודקוד 2 (השרת):

Diagram

Description automatically generated with medium confidence

אבל אני מקבל:

A picture containing graphical user interface

Description automatically generated

מחר אנסה לטפל בזה.

# 15 במרץ (יום רביעי)

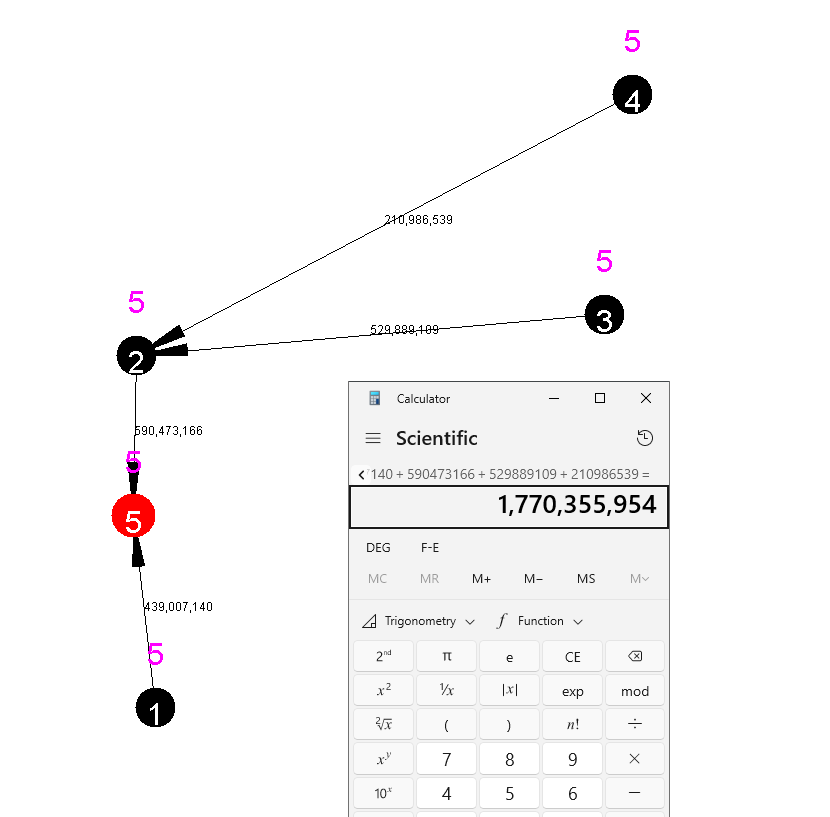
התחלתי לעבוד על השגיאה שמופיע למעלה. זה קשור ללוגיקה שלי ב- hasTerminated. מחזיר true.

הצלחתי לממש את החלק שהלקוח שולח לשרת הודעה, והשרת מחזיר תשובה, והקודקודים באמצע מעבירים לפי המסלול.

יצרתי סרטון שמדגים זאת:

<https://www.youtube.com/watch?v=PrCY5Sg487c>

הצלחתי לממש מרכיב 3 המחשב משקל של כל הגרף ומשקל של כל ה MST:



Graphical user interface, text

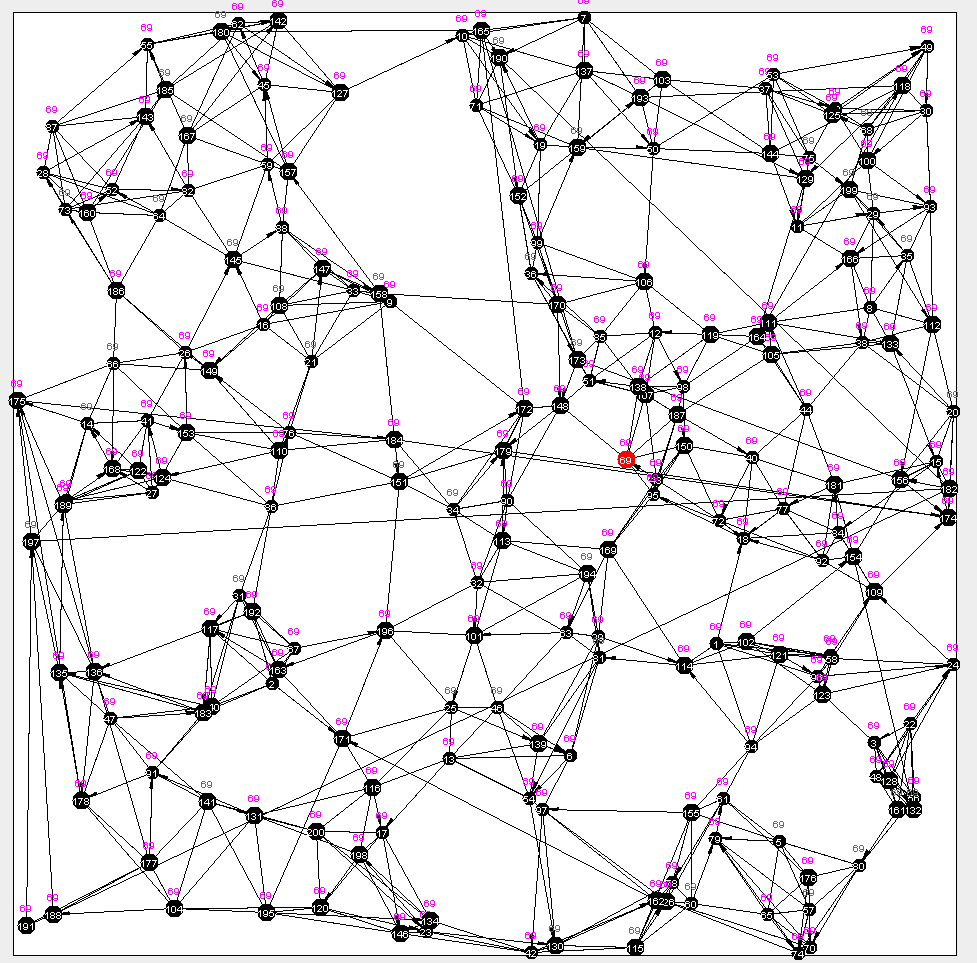
Description automatically generated

החישוב משקל של MST נעשה ע"י פונקציה רקורסיבית שמתחילה מהשורש ועוברת על כל השכנים שנמצאים ב MST ומחוברים לשורש.

ואז כל ילד כזה, מחשבים משקל של ה-subtree שלו כפי שעשינו בהתחלה.

וככה זה באופן רקורסיבי עד שמקבלים את כל המשקל של ה-MST.

עבור מרכיב 3, הרצתי עם 200 קודקודים וקיבלתי את הגרף הבא:



ואם רואים רק MST אז זה יראה כך:

A picture containing background pattern

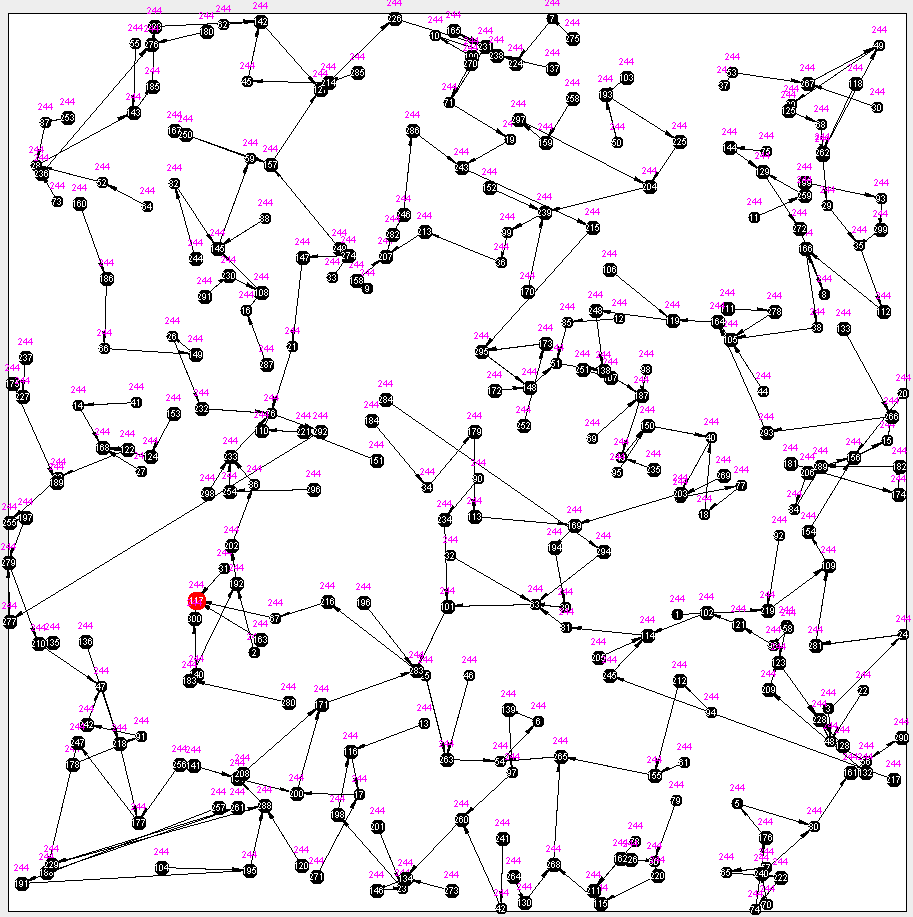
Description automatically generated

לקח לזה 32 אלף סיבובים לסיום.

Text

Description automatically generated

גם עם 300 ניסיתי, לקח 64 אלף סיבובים:



Text, table

Description automatically generated

ואני יוצר טבלת EXCEL שמראה את התוצאות:

Graphical user interface, application, Word

Description automatically generated

# 16 במרץ (יום חמישי)

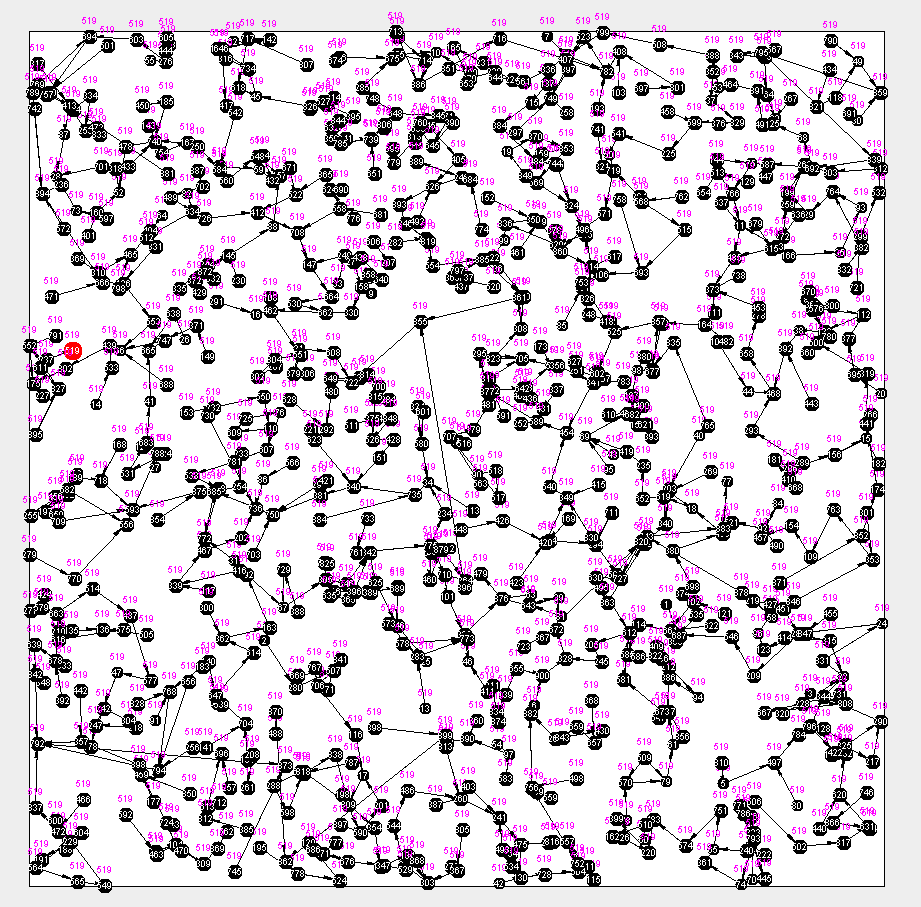
אני ממשיך להריץ את התוכנה בכדי ליצור את הטבלה.

בינתים זה התוצאות:

Graphical user interface, application

Description automatically generated with medium confidence

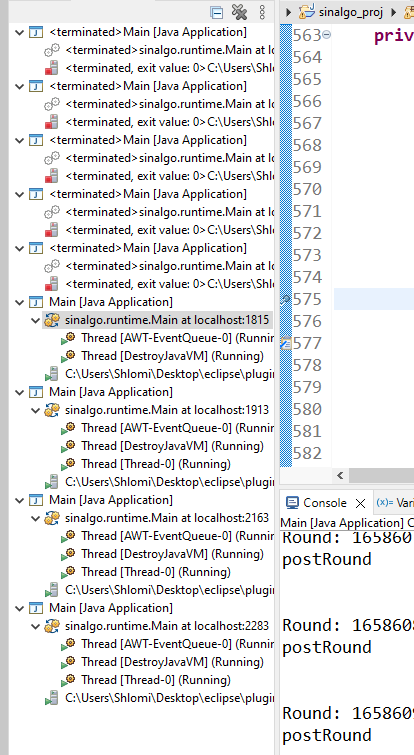
דוגמה לריצה של 900 קודקודים:



אני ממשיך להריץ את האלגוריתם.

# 17 במרץ (יום שישי)

אני ממשיך להריץ סימולטורים במקביל:



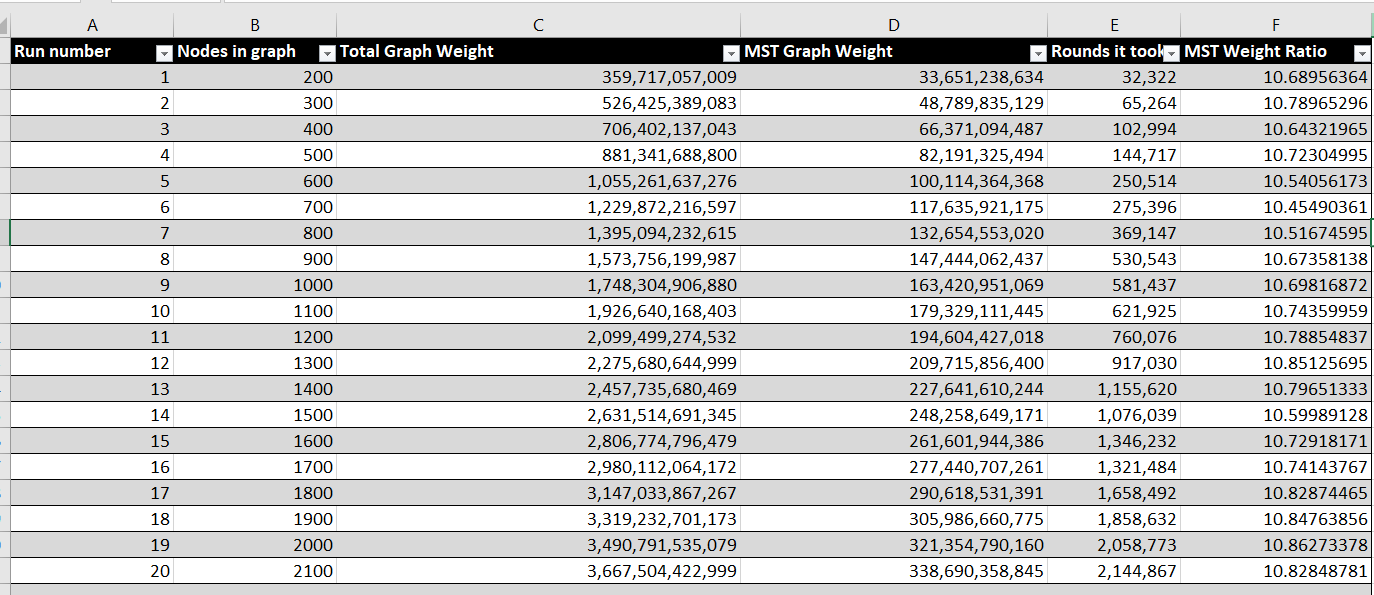
דוגמה לגרף עם 1800 קודקודים שסיים לרוץ:

Qr code

Description automatically generated

הרצתי את הסימולטורים בלילה.

תוצאה סופית:



*אני מצרף את טבלת ה excel לממן 15. נשים לב ש-MST Weight Ratio הוא סתם סטטיסטיקה, לראות עד כמה המשקל של MST שלי קטן לעומת הגרף כולו. למעשה ה ratio לא משתנה כמעט, הוא נשאר 10% מכל הגרף 😊*

*צירפתי את Rounds It Took כמות הצעדים שלקח להריץ את הגרף.*