**מבנה תוכנה ­מטלה 2**

**­ שיפור ספריית קוד אלגוריתמית**

**מגישות:**

מיכל קופרמן ת.ז 011845195

עדי בורנשטיין 300924727

מוריה פופובסקי 201390283

הדר לוי ת.ז 304838527.

**משימה 0**

שיפור קוד קיים לחישוב כל הקליקות בתחום גדלים מסוים בגרף לא מכוון

לאחר היכרות עם הקוד ולאור מה שלמדנו בכיתה עם ד"ר בועז הבנו שצריך לעשות שיפור בקוד.

את השיפור עשינו באמצעות הפונקציה הנ"ל:

Vector<Clique>allC\_seed(Clique edge, **int**min\_size, **int**max\_size)

הפונקציה מקבלת 3 משתנים: קליקה, ערך מינימלי שהוזן וערך מקסימלי שהוזן.

מהקליקה הנתונה, הפונקציה בונה את הקליקה המקסימלית (על ידי שימוש בקליקות קטנות יותר) או עד ל max size (הקטן מביניהם).

בפועל, בגרף יש קליקות רבות קטנות שנוצרות על אף שאינן יכולות להגיע למינימום הנדרש בקלט. לכן, תהליך זה מיותר ואף דורש משאבים רבים.

הדרך שלנו לשיפור הקוד הנ"ל והתמודדות עם הבעיה המוזכרת היא לא ליצור קליקות שמלכתחילה ידוע שאינן יכולות להגיע לגודל המינימלי הנדרש.

Vector<Clique>allC\_seed(Clique edge, **int**min\_size, **int**max\_size) {

Vector<Clique>ans = **new** Vector<Clique>();

ans.add(edge);

**int**i=0;

// int size = 2;

**while** (ans.size()>i) {

Clique curr = ans.elementAt(i);

**if**(curr.size()<max\_size) {

VertexSetNi = curr.commonNi();

**if**(curr.size()+Ni.size()<min\_size){}

**else**{

**for**(**int**a=0;a<Ni.size();a++) {

Clique c = **new**Clique(curr,Ni.at(a));

ans.add(c);

}

}

}

**else** {i=ans.size();} // speedup trick

i++;

}

**return**ans;

}

לדוגמא, אם קיימת קליקה בגודל N בעלת Mקודקודים פוטנציאליים, והערך המינימלי הנתון הינו גדול מN+M, אין צורך להמשיך בבניית הקליקה, מכיוון שלא תעמוד ביעד בכל מקרה.

עבור התנאי שהוספנו שמנו לב אחרי מספר הרצות כי כאשר מדובר בחישוב קליקות קטנות בגודלן יחסית ,לא ניראה שינוי ממשי בזמני הריצה.

לעומת זאת כאשר מדובר בקליקות גדולות בגודלן ניתן לראות שינוי משמעותי בזמני הריצה ואף תכניות שכשלו בקוד המקור כעת בזמן הריצה שלהם הן מחשבות את הקליקות כרצוי ואף בזמן קצר יחסית.

# המשך משימה 0

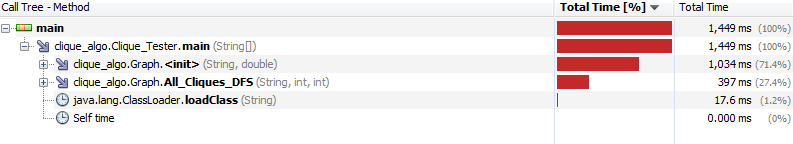
# השוואה בין הקוד המקורי לקוד המשופר

**השוואה בין שני הקודים בפרופיילר המובנה של NetBeans**

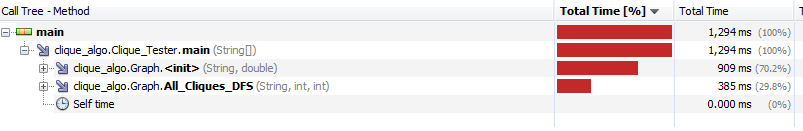
## ***ביצענו השוואת CPU :***

**מקרה 1: Min 6 Max 10 (0.75)**

הרצה על הקוד המקורי:

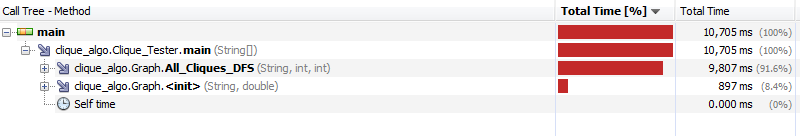
****

הרצה על הקוד המשופר:

****

**מקרה 2: Min 20 Max 30 (0.75)**

הרצה על הקוד המקורי:

****

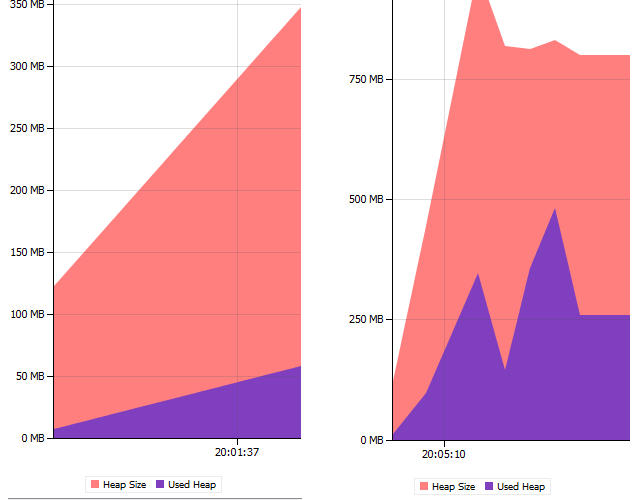
הרצה על הקוד המשופר:****

ניתן לראות, בהתאם לפרופיילר, שאכן זמן הריצה של הקוד **ירד** לאחר השינוי שביצענו.

בבדיקה שביצענועל מספרים קטנים (מקרה 1) הפער בין זמני הריצה בשני הקודים הוא לא מאוד גדול. אך כאשר אנו עוברים למספרים גדולים יותר (מקרה 2) ניתן לראות שזמן הריצה של הקוד ירד משמעותית.

ניתן לראות שהשינוי העיקרי מגיע מהפונקציה ששמה All\_Cliques\_DFS שממנה נקראת הפונקציה allC\_seed.

## ***ביצענו השוואת memory :***



**הקוד המקורי הקוד המשופר**

החלק הורוד: הקצאת זיכרון, החלק הסגול: זיכרון בשימוש.

רואים הבדל משמעותי בין הקוד המקורי לקוד המשופר. אין ספק שהקוד המשופר הוריד משמעותית את צריכת הזיכרון.

Unit – Tests

בדיקות יחידה הן בדיקות ברמת יחידת המערכת הקטנה ביותר (מודול) שמאמתות את פעילותה התקינה של היחידה. הבדיקות נערכות לרוב לאחר הכנת המודול או לאחר שבוצעו בו שינויים.

בוצעו בדיקות יחידה עבור המחלקות Graph, VertexSet, Clique.

בבדיקות נבדקו פונקציות שונות ובסיסיות עבור כל מחלקה. הבדיקות בוצעו באמצעות ספריית Junit Assert הקיימת ב java.

נעשה שימוש בבדיקת AssertEquals ובבדיקת AssertTrue. הטסטים עברו בהצלחה והוכיחו את תקינות הקוד.

**משימה 1**

שיפור ממשק טעינת הגרף

בחרנו להשתמש בפורמט CSV, כאשר כל שורה מייצגת קדקוד ואת רשימת שכניו.

הנחנו כי:

1. כל שורה ממויינת מספרית.
2. השורות הינן רציפות, כלומר אין "חורים", לכל קדקוד מ- 0 עד למס' הקדקודים קיימת שורה.
3. הקלט תקין, ז"א מספרים מ- 0 עד למס' הקדקודים.

בכדי לטעון את ממשק הגרף החדש הוספנו בנאי חדש המקבל קובץ CSV וקורא לפונקציה נוספת שבנינו- initNeighborsList.

הפונקציה עוברת שורה שורה ומכניסה עבור כל קדקוד את שכניו למערך vertexSet.

**משימה 2**

הוספת פונקציונלית ­קליקה מקסימלית

על מנת לחשב את הקליקות המקסימאליות הוספנו 3 פונקציות חדשות:

\* this function finds the max size clique from all cliques.

**public** **int** MaxSizeAllClique()

הפונקציה מחשבת מה הגודל של הקליקה המקסימאלית. בשביל זה היא צריכה לעבור על כל הקליקות.

הפונקציה מחזירה int שהוא הגודל המקסימאלי.

על מנת לשפר את צריכת הזיכרון, שלחנו מתוך פונקציה זו, כל Clique , לפונקציה נוספת:

this function checks size of clique by edges.

**public** **int** checkSizeSpecCliqebyedge(Clique e){

בפונקציה זאת בחרנו להשתמש במחסנית(stack) ובכך לשפר משמעותית את צריכת הזיכרון (לעומת שימוש בווקטור)

הפונקציה מחזירה int שהוא הגודל המקסימאלי של הקליקה שנשלחה אליה.

הפונקציה השלישית רק דואגת למצוא את כל הקליקות שהם בגודל המקסימאלי שמצאנו (ניתן גם להשתמש בפונקציה שקיימת בקוד המקורי allC\_seed. שמוצאת קליקות בטווח מסוים ולהכניס min ו max את אותו ערך מקסימאלי)

הרצנו את הטסטרים עם קריאה לפונקציות החדשות ואכן קיבלנו חזרה את הערך של הקליקה המקסימלית.

**Task 3**

Adding support the calculation based on DFS algorithm

The main problem:

Distribution of tasks to multiple computers

The solution:

The server receives graph with edges, he use the function “Minimal Cliques” that calculate the edges from graph.

Then the server sends each time edge to one of the computers array (As calculation algorithm DFS, that sends vertex each time).

The computers calculates the length of maximal clique

From an edge that was sent to him

The server receives databases from each of computer and updates the length of maximal clique as needed.

In the event, edge is not relevant and shorter then max clique, the server avoids sending.

A server is an [instance](http://www.computerhope.com/jargon/i/instance.htm) of a computer program that accepts and responds to requests made by another program, known as a [client](http://www.computerhope.com/jargon/c/client.htm).

The computers approached to server through a computer network to request information stored on the server.  
Servers are used to manage [network](http://www.computerhope.com/jargon/n/network.htm) resources.   
Some servers are committed to a specific task, often referred to as dedicated. As a result, there are a number of dedicated server categories, like print servers, file servers, network servers and database servers.  
Because they are commonly used to deliver services that are required constantly, most servers are never turned off. Servers are commonly high-end computers setup to be [fault tolerant](http://www.computerhope.com/jargon/f/faulttol.htm).

**Server** - Characterized server computers running multiple programs and computer processes concurrently.



**Computers** – Computer receives commands and tasks from the server, and sends the databases to server.