תכנות מערכות 2 –README מטלה 2

תז: 323858324 shannya08@gmail.com

אנחנו נדרשנו לממש את הקבצים הנל למטלה1 ולכן השינויים שבוצעו עבור מטלה 2 הם:

Graph.cpp,Algorithms.cpp,Test.cpp,Makefile,Algoritms.hpp,Graph.hpp,demo.cpp

בקובץ **Test.cpp** הוספתי עוד טסטים שקשורים למטלה 2 של פונקציות העמסת אופרטורים וגם בקובץ **Demo.cpp** יש דוגמאות הרצה שקשורות למטלה2.

Graph.hpp:

הקובץ מכיל מחלקה המייצגת גרף כאשר גרף מיוצג על ידי מטריצת שכנויות כך שאם יש מספר שונה מ0 בתא **[i,j]** אז זה אומר שיש צלע בין הקודקוד **i** ל **j** ואם יש 0 אז זה אומר שאין צלע בין הקודקודים.

המחלקה מכילה פונקציה פומבית בשם **loadGraph** ,פונקציה פומבית נוספת בשם **printGraph** ,פונקציה פומבית נוספת **size** מחזירה את מספר הקודקודים בגרף.

,ופונקציה פרייבט בשם **validateGraph** המבצעת בדיקה על תקינות הגרף, היא מוודאת שהמטריצה ריבועית ומחזירה במקרה שלא חריגה של **invalid_argument**.

אנחנו נעבוד במטלה שלנו עם גרפים מכוונים כלומר אם יש לנו צלע מקודקוד **i** לקודקוד **j** בגרף לא מכוון אנחנו נתייחס אליה כאל שתי צלעות כי מתייחסים לגרף כמכוון.

הוספתי את המימושים של הפונקציות העמסת אופרטורים =+, פלוס אונרי מינוס אונרי ,=-, =*.

והצהרות של הפונקציות שב**Graph.cpp**.

Graph.cpp:

בנוסף לכל המימושים שהיו הוספתי לקובץ פונקציות של העמסת אופרטורים לפי ההנחיות שכתבו לנו במטלה.

חיבור בין שתי גרפים באותו גודל +

חיסור בין שתי גרפים באותו גודל -

שישה אופרטורי השוואה: גדול, גדול-או-שווה, קטן, קטן-או-שווה, שווה, לא-שווה

הגדלה ב-1 (++) והקטנה ב-1 (--) לפני ואחרי המספר.

הכפלת גרפים - פעולת הכפל בין גרף **G1** לגרף **G2** היא על ידי מכפלה של המטריצות המייצגות של שני הגרפים. התוצאה צריכה להיות מטריצה המייצגת גרף.

וביצעתי זריקת שגיאה אם אנחנו מנסים לכפול שני גרפים לא מאותו סדר גודל.

אופרטור פלט – הדפסה כמו הדפסה של מטריצה [ ]

Algorithms.hpp:

זהו קובץ שבו ישנם כל ההצהרות של הפונקציות שממומשות **Algorithms.cpp .**

Algorithms.cpp:

זהו הקובץ שבו ישנם את כל המימושים של הפונקציות שנדרשנו שממומשות לפי האלגוריתמים שלמדנו באלגוריתמים1

isConnected(g) - פונקציה זו בודקת אם הגרף הינו קשיר, באמצעות אלגוריתם BFS

shortestPath(g,src ,dest) - פונקציה זו מחשבת את המסלול הקצר ביותר בין צומת מקור (src) לצומת יעד (dest) בגרף, באמצעות אלגוריתם Bellman-Ford.

isContainsCycle(g) – פונקציה זו בודקת האם קיימים מעגלים בגרף על ידי בדיקה של כל צומת בגרף ומפעילה את הפונקציה isContainsCycleUtil על כל צומת, הפונק הזו מבצעת DFS כדי לזהות האם קיים בו מעגל.

isBipartite(g) - פונקציה זו בודקת אם הגרף הינו דו-צדדי, כלומר ניתן לצבוע את כל הצמתים של הגרף בשני צבעים כך שכל צומת יהיה מקושר לצמתים בצבע שונה ממנו

negativeCycle(g) - הפונקציה negativeCycle בודקת האם קיים מעגל שלילי בגרף, מעגל שלילי הוא מסלול בגרף שסכום המשקלים של הצלעות בו שלילי.

והיא עושה זאת על ידי שימוש בפונקציה shortestPath , היא עוברת על כל הצמתים בגרף, ומפעילה את shortestPath עם כל צומת כצומת המקור וכצומת היעד. אם shortestPath מחזירה את המחרוזת "**Negative cycle detected in the path**", זה אומר שמצאה מעגל שלילי.

אם אף אחד מהקריאות ל shortestPath לא מחזירה " **Negative cycle detected in the path"** היא מחזירה את המחרוזת " **The graph does not contain negative cycle**", שאומרת שאין מעגלים שליליים בגרף.

אופן ההרצה: כדי להריץ נכתוב את הפקודה בטרמינל make שיריץ את הקובץ demo.cpp כפי שכתוב בmakefile. ואז כדי להריץ את הטסטים נכתוב make test-> ./ .test

אם נרצה למחוק את הקבצים שנוצרו נבצע make clean.

נעזרתי בCHAT GPT

ויקיפדיה

StackOverFlow