תכנות מערכות 2: מטלה 1

נועה פישמן – 319055430
noa.fishman@gmail.com

Graph:

במחלקה זו הגדרתי משתנה מסוג גרף שמכיל בתוכו את מטריצת השכנויות המייצגת את הגרף, את גודל הגרף ואת מספר הצלעות שיש לו. במחלקה זו נדרשנו לממש שתי פונקציות פונקציית ההדפסה ופונקציית הטעינה. המחלקה נמצאת תחת מרחב השמות "noa".

בקוד שלי אני מתייחסת לכל הגרפים בתור גרפים מכוונים ככה שאם הם אינם מכוונים מטריצת השכנויות שלהם תהיה מטריצה סימטרית. בנוסף, אם משקל בין שני קודקודים הוא 0 סימן שאין שם צלע.

LoadGraph: פונקציה זו מקבלת וקטור דו מימדי המייצג את מטריצת השכנויות. ראשית הפונקציה בודקת שגודל הגרף באמת תקין וקיבלנו מטריצה ריבועית, לאחר מכן היא בונה גרף חדש ריק בעזרת הבנאי הריק ואז מעדכנת את מטריצת השכנויות שלו, מודדת את הגודל שלו וסופרת כמה צלעות יש לו, ומעדכנת בהתאם.

printGraph: פונקציה זו מדפיסה את הגרף בכך שמדפיסה כמה קודקודים וכמה צלעות יש לו, ובנוסף מדפיסה גם את מטריצת השכנויות שלו.

getSize: פונקציה שאני הוספתי שמחזירה את הגודל של הגרף (כמות עמודות או שורות). הוספתי פונקציה זו מכיוון שרציתי לשמור על המשתנה size של האובייקט משתנה פרטי כלומר שלא יהיה ניתן לשינוי מבחוץ.

getWeight: פונקציה שאני הוספתי שמקבלת שני קודקודים ומחזירה את המשקל של צלע ביניהם. כלומר את הערך במיקום הזה במטריצת השכנויות. ככה אני שומרת על המטריצה פרטית (לא ניתנת לשינוי).

Algorithms:

במחלקה זו מימשתי את הפונקציות המבוקשות, כל הפונקציות במחלקה הן סטטיות. המחלקה נמצאת גם היא תחת מרחב השמות "noa". כעת אסביר מעט על כל אחת מהפונקציות ועל אופן המימוש שלה.

isConnected: הפונקציה מקבלת גרף ובודקת האם הוא גרף קשיר או לא. אם כן הפונקציה תחזיר 1 אחרת תחזיר 0. הפונקציה בונה ווקטור flag באורך n (גודל הגרף) אשר מייצג עבור כל קודקוד האם אפשר להגיע דרכו לכל שאר הקודקודים. כעת נבצע את האלגוריתם בלמן פורד עבור כל הקודקודים (n פעמים) ובסוף כל איטרציה נבדוק האם קיים מסלול מהקודקוד שעכשיו אנו בודקים לכל שאר הקודקודים, נשנה את הערך בווקטור flag כך שיציג true במקום המתאים. לבסוף נעבור על הווקטור flag ונבדוק שאכן מכל קודקוד ניתן להגיע לכל קודקוד ונחזיר את התוצאה המתאימה.

shortestPath: הפונקציה מקבלת גרף ושני קודקודים (מספרים המייצגים אותם ולכן הם ללא סימן). הפונקציה מממשת את אלגוריתם בלמן פורד כדי למצוא את המסלול הקצר ביותר. השתמשתי באלגוריתם זה מכיוון שקיימת אפשרות למשקלים שליליים בגרף מה שיכול להוביל למעגלים שליליים ובלמן פורד יודע לאתר את המעגלים הללו. במהלך הריצה הגדרתי שני וקטורים אחד המייצג את הקודקוד הקודם של כל קודקוד והשני מייצג את משקל המסלול מקודקוד ההתחלה לכל קודקוד. לאורך האלגוריתם אני ממעדכנת את הווקטורים הללו בהתאם ככה שבסוף האלגוריתם אני ניגשת לווקטורים במיקום ה-y שלהם (קודקוד היעד של המסלול), והולכת כל פעם לקודקוד הקודם עד שמגיע לקודקוד x (קודקוד ההתחלה). בזמן המעבר על המסלול אני מכניסה את כל הקודקודים למחרוזת כך שלבסוף אני מקבלת מחרוזת המייצגת את המסלול הקצר ביותר בין שני הקודקודים

ומחזירה אותה, קודקוד שאין מסלול אליו הערך שלו בווקטורים יהיה אינסוף (הערך הגדול ביותר עבור משתנה מסוג זה). בכל מסלול שלפחות אחת מהנקודות בו היא חלק ממעגל שלילי זה נקרא שלא קיים מסלול.

isContainsCycle: הפונקציה מקבלת גרף ומחזירה האם קיים מעגל כלשהו בגרף או שלא. הפונקציה משתמשת באלגוריתם DFS אשר "סורק" את הגרף החל מקודקוד מסוים. אנו מבצעים את האלגוריתם n DFS פעמים בתוך לולאה (כל פעם מתחילים לסרוק מקודקוד אחר). האלגוריתם רץ על הקודקודים לעומק וכל קודקוד שנמצא בתהליך הוא מסמן באפור (1) כך שאם הוא מגיע שוב לקודקוד שכבר מסומן באפור זה אומר שנמצא מעגל ולכן יסמן את הקודקוד שדרכו הגיע ב-3 (צבע חדש) כך שבסוף הסריקה של הגרף אם קיים קודקוד המסומן ב-3 אנו יודעים שקיים מעגל ונצא מהלולאה הגדולה. בסוף הלולאה הגדולה אנו נלך אל הקודקוד המסומן ונחזור לאחור עד שנגיע לקודקוד זה שוב. בזמן הזה אנו מכניסים לתוך מחרוזת את כל הקודקודים במעגל כך שבסוף המחרוזת תייצג את המעגל ונוכל להחזיר אותה. אם לא קיים מעגל כלומר, אף קודקוד אינו מסומן ב-3, נחזיר 1-.

isBipartite: הפונקציה מקבלת גרף ומחזירה 1 אם הוא דו צדדי ו-1- אחרת. בתחילת הפונקציה אני מגדירה משתנה בוליאני flag ווקטור שמייצג את הצבע של כל קודקוד (1-, 0, 1) וכל הקודקודים מאותחלים להיות 1-. נבחר רנדומלית קודקוד אחת ונשנה את צבעו ל-0 וכעת נרוץ כמו בבלמן פורד ודיג'קסטרה n פעמים וכל פעם כל קודקוד שלא מסומן ב1- נסמן את שכניו בצבע השני (אם הוא 1 אז 0 ולהיפך). בסוף n הריצות נבדוק האם קיים קודקוד שעדיין מסומן ב-1- אם כן סימן שהוא בקבוצת קשירות שונה וכעת נשנה את הסימון שלו ונרוץ שוב n פעמים. נעשה את זה עד שלא יהיה קודקוד במסומן ב-1- ואז נשנה את הדגל שלנו ונצא מהלולאה. אם לאורך הריצה הגענו לקודקוד שאחד השכנים שלו מסומן באותו מספר כמוהו (לא 1-) גילינו שהגרף אינו דו צדדי ולכן נחזיר 0. כאשר נסיים את הריצה נכניס את כל הקודקודים המסומנים ב-0 למחרוזת אחת ואת הקודקודים המסומנים ב-1 למחרוזת אחרת ולבסוף נאחד את שתי המחרוזות למחרוזת אחת ונחזיר אותה.

negativeCycle: הפונקציה מקבלת גרף ומחזירה 0 אם לא קיים בגרף מעגל שלילי, ואם קיים מעגל שלילי מחזירה מחרוזת המייצגת את המסלול שלו. במימוש של הפונקציה השתמשתי באלגוריתם של בלמן פורד כך, שכאשר מצאתי מעגל שלילי בעזרת האלגוריתם השתמשתי בווקטור ששומר בתוכו את הקודקוד הקודם של כל קודקוד כדי למצוא את המעגל. כך הלכתי מהקודקוד שמצאתי בו את המעגל אחור עד שחזרתי לאותו הקודקוד והכנסתי בהתאם את כל הקודקודים שעברתי דרכם למחרוזת שאותה אציג. אם לא מצאתי מעגל בסוף הריצה של בלמן פורד אני מחזירה 0.

DFS: פונקציה שאני הוספתי כדיי לממש את האלגוריתם DFS שעוזר לי למצוא מעגל בגרף. הפונקציה היא פונקציה רקורסיבית אשר מקבלת את הגרף, קודקוד שעליו רוצים לבצע את הבדיקה, רפרנס לווקטור של צבע ורפרנס לווקטור של הורה. לפי האלגוריתם אנו עוברים על כל הצלעות שיוצאות מהקודקוד ומבצעים את הפונקציה על הקודקודים בצד השני של הצלעות בנוסף אנו מעדכנים לכל קודקוד את ההורה שלו בהתאם לקודקוד דרכו הגענו אליו. אם הגענו לצלע שכבר בעבודה (מסומנת בצבע אפור – 1) הגענו למעגל ונסמן אותה בצבע – 3 כדי למצוא אותה אחר כך ונחזיר להורה שלו 1 כדי שנדע שמצאנו מעגל. לאחר סיום הפונקציה על כל הילדים של קודקוד זה נסמן אותו בצבע שחור – 2 ונחזיר להורה שלו. כך נעשה עד שנחזור לקודקוד ההתחלה.

Demo:

במחלקה הזו אני מממשת את כל הפונקציות על 10 גרפים, 4 שנתנו לנו ו6 חדשים שהוספתי. גל הגרפים הללו אני מפעילה את כל הפונקציות שממשתי בפונקציית main. המחלקה הזאת נועדה בעיקר כדי לבחון את כל הגרפים ולהפעיל על כמה סוגים של גרפים את כל הפונקציות כדי שאני אוכל לבדוק תקינות של הפונקציות ומקרי קיצון.

Test:

את המחלקה test קיבלנו עם 10 טסטים כתובים מכיוון שנדרשנו לכתוב 20 טסטים הוספתי עוד 20 טסטים לפחות לעשרה שקיבלנו. בראש המחלקה בהערה יש הסבר על כל הגרפים במתומצת כדי

להבין מה הם באים לבדוק בנוסף, במחלקה demo על כל הגרפים הללו מופעלות כל הפונקציות ככה אפשר לראות עד בדיקות של הפונקציות מעבר למה שכתבתי בטסטים.

Makefile:

בקובץ זה קימפלתי את כלל הקבצים והגדרתי שכדי להריץ את ה-main של הדמו צריך קודם לעשות בטרמינל make Demo ולאחר מכן Demo/. ועבור ה- main של הטסטים צריך לעשות make Test ולאחר מכן Test/.