שאלה 1

1. הרעיון שלנו הוא לאפשר גישה לצמתים ולקשתות של הגרף בעזרת פעולות הוספה והסרה של צמתים וקשתות, ובכך הגרף יכול להיות דינאמי (אם למשל לא הייתה פעולת הסרה, אז היכולות של הגרף היו קטנות משמעותית לפי דעתנו).  
   בנוסף, רצינו שבפעולות אלו יהיה פידבק למי שמשתמש במחלקה, כלומר שהוא ידע האם הפעולה שביקש הצליחה או לא.  
   לדעתנו צריכה להיות גישה ל-data של כל צומת – גם קריאה שלו וגם החלפתו, ולכן הוספנו מתודות set/get ל-data של צמתים.
2. חילקנו את הגרף לשני חלקים – חלק אחד הוא הצמתים עצמם (המחלקה Node) וחלק שני המיפוי בין הצמתים לבין התוויות שלהם.

השיקול המרכזי בכך היה לפשט את המימוש – הגרף עצמו יהיה אחראי על נכונות הוספת\הסרת צמתים וקשתות, והצמתים עצמם יכילו את הקשרים הפנימיים של הגרף.  
הרעיון הוא שכל צומת "יודעת" לאיזה צמתים היא מחוברת ואיך, ומכאן הגיע המימוש.  
לשם כך בחרנו במפות בשביל לקשר בין התוויות לבין הצמתים, ובין (תוויות של) צמתים לבין (תוויות של) קשתות (בפרט ב-HashMap בגלל הביצועים שלה).

מכאן יעילות הפעולות טובה:

|  |  |
| --- | --- |
| פעולה | יעילות |
| removeNode | מעבר על הבנים והאבות של הצומת |
| addNode, addEdge, removeEdge, getNodeData, setNodeData, getChildByEdgeLabel, getParentByEdgeLabel | מציאת ערך במפה ו/או הוספה למפה – בממוצע מתבצעות בזמן קבוע |
| getNodes(boolean) | שכפול של הצמתים בצבע הרצוי (מעבר על כל הצמתים בגרף) |
| getChildren, getParents, getNodes, getOutgoingEdges, getIncomingEdges | מתבצעות בזמן קבוע |

1. מימוש חלופי – לשמור 3 מפות:
   1. מפה של הצמתים השחורים
   2. מפה של הצמתים הלבנים
   3. מפה של הקשתות

קשת תכיל מידע על צומת ההתחלה וצומת הסיום שלה.

השוואה למימוש שבחרנו:

* פעולות על צומת בודד (הוספה, הסרה, עריכת data וכו') מתבצעות באופן דומה למימוש שבחרנו.
* הוספת/הסרת קשת – פשוטה – בדיקה האם הקשת קיימת, ואם כן אז הוספה/הסרה מהמפה של הקשתות.
* get של הצמתים – פשוטה – לקיחת הצמתים מהמפה המתאימה לצבע (או איחוד של שתי המפות).
* הבעייתיות במימוש הזה מתחילה בשאר ה-get'ים:
  + בשביל לקבל את הצמתים האבות/בנים של צומת, צריך לחפש **בכל המפה** של הקשתות.
  + בשביל לקבל צומת הורה או בן לפי edge label, גם צריך לחפש **בכל המפה** של הקשתות.

היתרון במימוש החלופי הזה הם ב-get של צמתים צבועים, שמתבצע בזמן קבוע.  
החסרונות הן ב-get של הורה/בן לפי edge label, וב-get של אבות/בנים של צומת (שהדגישו שאלה פעולות שצריכות להתבצע בזמן קבוע), **שלא** מתבצעות בזמן קבוע.

|  |  |
| --- | --- |
| בדיקה | מטרה |
| testExample | בדיקת דוגמה |
| anotherGeneralTest | בדיקה כללית של הגרף |
| addingIllegalNodes | בדיקת הוספת צמתים לא חוקיים – צמתים שכבר קיימים בגרף |
| addingIllegalEdges | בדיקת הוספת קשתות לא חוקיות – קשתות בין צמתים עם אותו צבע, בין צומת לעצמו. |
| removingNodes | בדיקת הסרת קשתות |
| removingEdges | בדיקת הסרת צמתים |
| changingNodesData | בדיקת שינוי המידע שמחזיקים הצמתים |

בכל אחת מהבדיקות משולבות גם בדיקות שלאו דווקא קשורות לנושא הבדיקה.

הבדיקות מכסות את כל היכולות של הגרף שמימשנו, לכן לפי דעתנו הן מספיקות.

הערה: הסרנו חלק מההנחות על המתודות של BipartiteGraphTestDriver.

שאלה 2