**מונחה עצמים**
**הגשה 2 חורף 2020-2021**
**צחי פרץ 203840095**
**שימי הללויה 205836455**

<u>שאלה 1</u>
המתודות אותן מימשנו למחלקה graph הן ->
א.

```
public Graph(String name)
public boolean AddNode
public boolean AddEdge E srcNode, E dstNode
public ArrayList<E> ListNodes() {
public ArrayList<E> ListChildren E node
```

הפעולות הללו מספיקות מכיוון שעלינו לתמוך ביצירה של גרף , בהוספה של צמתים או קשתות אליו, וכן לשם חישוב מסלולים נרצה פ׳ שתחזיר רשימה של כל הילדים של צומת כלשהו , ורשימת כל הצמתים בצומת כלשהוא .

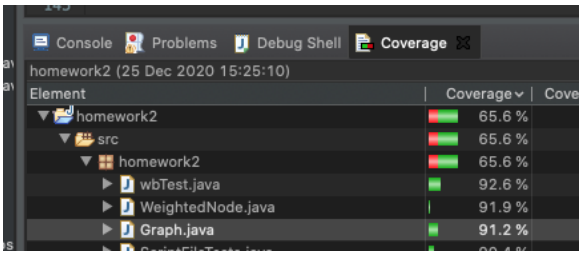
ב.
הרעיון היה להשתמש ב map – מבנה נתונים שממפה key ל value .
כאשר הkey שלנו הוא הצומת והערך אליו הוא ממפה הוא סט של צמתים נוסף שהם למעשה הבנים של אותו צומת. בשיטה הזו ניתן לעקוב אחר המסלולים האפשריים בגרף.
שיטה נוספת למימוש יכולה להיות אחזקה של רשימה של צמתים ביחד עם מטריצת שכנויות ( לכל צומת , נחזיק במטריצה מי השכנים שלו ) – החיסרון של השיטה הזו לעומת השיטה שלנו ,הוא סיבוכיות חישוב בעת הוספה של קשת לדוגמא. כי בעת הוספה של קשת יש לעבור על כל השורות במטריצה ולבדוק האם יש לעדכן את רשימת השכונויות .

ג.
<u>בדיקות קופסא שחורה שביצענו</u>
createGraphs – ניצור מספר גרפים
createNodes – ניצור מספר צמתים
addOperations – נבדוק הוספה של קשתות או צמתים לגרפים
childrensTest – נבדוק שההדפסה של הילדים של צומת נכונה
listNodes – רשימת הצמתים בגרף מסוים

<u>בדיקות קופסא לבנה</u>
הגענו למעל 90% כיסוי מהקוד של מחלקת graph
הרעיון היה לבדוק את רוב שורות הקוד בכל מתודה , כולל חריגות , לשם כך בכל מבחן אילצנו תנאי התחלה שונים וכך גרמנו לכיסוי כמעט מלא .

```
createGraph()
addNodeGraphGood
addNullNodeGraph
addNodeAlreadyInGraph
addEdgeGood
```

addEdgeBad
addEdgeNullSrc
addEdgeNullDst
listChildrenNullNode
listChildrenNotInGraph

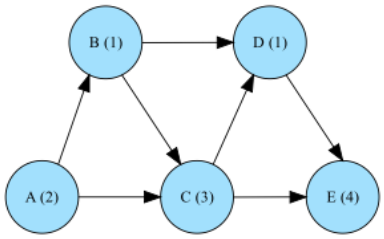



ד.
מהיבט של קריאות הקוד – בנינו מחלקה של גרף ולקרוא לטיפוס שהוא שם של קובץ בשם path ,עשוי לגרום לחוסר בהירות. כאשר רואים path, בהקשר של גרף , מקובל לחשוב שמדובר במסלול בגרף ולא במסלול של קובץ חיצוני .

## שאלה 2

א.
אלגוריתם דיאקסטרה , תיאור כללי –
מטרתו של האלגוריתם הוא מציאת המסלול הקצר ביותר בגרף מ start לgoal .
בגרף שלנו , לכל צומת יש משקל w.
מבנה הנתונים שמשמשים לעזר באלגוריתם –
א. שדה d לכל צומת , שמסמן את עלות ההגעה ממנו ל goal.
ב. שדה path (pi) – מסמן לכל צומת , את הצומת שממנו הגענו אליו .
ג. נשתמש במבנה נתונים שהוא priority queue , שמשמש כערימת מינימום , לפי מפתח של שדה הd של כל צומת.

שלבי האלגוריתם
1.נאתחל את d של נק׳ start ל0 , ואת ה d של שאר הצמתים לmaximum_value
2.לאחר מכן , בצורה איטרטיבית , נעבור על ערימת המינימום בצורה הבאה –
נשלוף איבר מראש הערימה (d מינימלי )
נביט על כל הילדים של הצומת ששלפנו , ונבדוק האם ערך הd שלהם גדול מהערך d של האיבר הנשלף ועוד המשקל של הילד עצמו .
אם כן , נעדכן עבור הילד את שדה ה. $d_{child} = d_{parent} + w_{child}$ , $pi_{child} = pi_{parent} + child$
נכניס את הילד לערימת מינמום , אם הוא לא קיים שם ולא סיימתי
3. תנאי העצירה של 2 הוא כאשר ערימת המינימום ריקה .




התחלה

| | A | B | C | D | E |
|---|---|---|---|---|---|
| Active | + | | | | |
| Paths | a | | | | |
| cpath | 2 | | | | |
| Finished | | | | | |

איטרציה 1

| | A | B | C | D | E |
|---|---|---|---|---|---|
| Active | | + | + | | |
| Paths | a | a ,b | a,c | | |
| cpath | 2 | 2+1=3 | 2+3=5 | | |
| Finished | + | | | | |

איטרציה 2

| | A | B | C | D | E |
|---|---|---|---|---|---|
| Active | | | + | + | |
| Paths | a | a ,b | a,c | A,b,d | |
| cpath | 2 | 3 | 5 | 3+1=4 | |
| Finished | + | + | | | |

איטרציה 3

| | A | B | C | D | E |
|---|---|---|---|---|---|
| Active | | | + | | + |
| Paths | a | a ,b | a,c | A,b,d | A,b,d,e |
| cpath | 2 | 3 | 5 | 3+1=4 | 4+4=8 |
| Finished | + | + | | + | |

איטרציה 4

| | A | B | C | D | E |
|---|---|---|---|---|---|
| Active | | | | | + |
| Paths | a | a ,b | a,c | A,b,d | A,b,d,e |
| cpath | 2 | 3 | 5 | 3+1=4 | 4+4=8 |
| Finished | + | + | + | + | |

איטרציה 4

| | A | B | C | D | E |
|---|---|---|---|---|---|
| Active | | | | | |
| Paths | a | a ,b | a,c | A,b,d | A,b,d,e |
| cpath | 2 | 3 | 5 | 3+1=4 | 4+4=8 |
| Finished | + | + | + | + | + |

<u>פתרון שאלה 3:</u>

פתרון:

נשים לב כי ברירת המחדל בJava היא הגדרת גישה של package כלומר בשני המחלקות מאחר ולא מוגדרת הגישה למחלקות, הmembers שלהן חשופות לפונקציות חיצוניות למחלקה ובכך נקבל representation exposure.
ניתן לפתור את הבעיה ע"י הצהרת גישה למשתנים החברים במחלקות בprivate והוספת מתודות get וset שיתמכו בקבלת המשתנים ועריכתם מתוך המחלקה עצמה ולא ע"י גישה מחוץ למחלקות.
כמו כן יש לשים לב אם מוציאים החוצה את שורש העץ, לא להחזיר את המצביע עצמו אלא העתק של העץ כדי למנוע representation exposure.

פתרון:

הסטודנט טועה.
אם הסטודנט היה מוחק את הבנאי חסר הפרמטרים ומשאיר את הבנאי שמקבל פרמטרים הוא היה מקבל שגיאת קומפילציה.
זאת מכיוון שניתן באמת לוותר על הבנאי חסר הפרמטרים רק אם לא מוגדר שום בנאי אחר (ככה זה בjava) ואז היה נקרא הבנאי הדיפולטי אבל כל עוד ממומש בנאי נוסף זאת תהיה שגיאה!

פתרון:

@requires
Node != root.left && node != root.

@modifies
this

@effects set the right son of the root of this tree with node, if node == null the root loss his right node son.

פתרון:

נגדיר את המחלקה כמחלקה גנרית כך: class<T>
ובשדה של key: T key.
במצב כזה בעת יצירה של האינסטנס node אנו מגדירים מה יהיה הטיפוס של ערך המפתח שלו.

פתרון:

בשביל שנוכל לבצע השוואה בין ערכי המפתחות של הnode נצטרך לדרוש שאותו טיפוס נתונים T שייצג לנו את ערך המפתח של הצומת יירש מהמחלקה של comparable לכן נצהיר על המחלקה באופן הבא:
Class Node<T extends comparable<? Super T>>

פתרון:

Node abstraction function

A node represents a node which is a element of a BinaryTree tree such that node.key is an integer key
node.left is the left child or null if doesn't exist
node.right is the right child or null if doesn't exist

Node representation invariant

For each Node x
if (x.right != null && x.left != null)
x.right != x.left
x != x.right
x != x.left

BinaryTree abstraction function

A BinaryTree BT represents a tree that consists of maximum 2 nodes for each node
BT.root represents the root of the tree or null if it is empty.

BinaryTree representation invariant

BinaryTree has no circles in it.

Node representation invariant

For each Node x
if (x.right != null && x.left != null)
x.right != x.left
else if (n.right != null)
x != x.right
x = x.right.parent
else if (x.left != null)
x != x.left
x = x.left.parent

BinaryTree representation invariant

For each BinaryTree BT
BT.parent == null iff BT.node is the root
Only allowed circles in BT are circles between a parent and his child