

תכנות מערכות 2

מטלה 1 – גרפים

הגדרות

- < גרף G הוא קבוצה סופית ולא ריקה של קודקודים.
- < גרף ריק ($empty\ graph$) הוא גרף ללא צלעות, אך עם קודקודים.
- < הצלע $u \rightsquigarrow v$ והצלע $u \rightsquigarrow v$ יצרו מעגל אם $w(uv) \neq w(vu)$. אם $w(uv) = w(vu)$ הצלע $u \rightsquigarrow v$ היא צלע לא מכוונת ולא תיצור מעגל.
- < עבור משתנים ומתודות שאינם, לפי הגדרה, יכולים להחזיק מספרים שליליים נבחר הטיפוס $size_t$.

Graph

עבור המחלקה לייצור גרף קיימים שני קבצים: $graph.cpp$ ו- $graph.hpp$.

המחלקה כוללת את השדות הפרטיים הבאים:

שדה	הסבר
<code>vector<vector<int>> adjacencyMatrix</code>	מטריצת שכנויות המייצגת גרף
<code>size_t _numVertices</code>	משתנה המחזיק את מספר הקודקודים בגרף
<code>size_t _numEdges</code>	משתנה המחזיק את מספר הצלעות בגרף
<code>bool _isDirected</code>	דגל המסמל אם הגרף מכוון או לא

בנוסף, המחלקה גרף כוללת את המתודות הפומביות הבאות:

מתודה	הסבר
<code>Graph()</code>	בנאי ברירת מחדל. עם יצירת מופע חדש של גרף, מספר הקודקודים והצלעות מאותחלים להיות 0. בנוסף, הגרף מוגדר להיות גרף לא מכוון.
<code>void Graph::loadGraph(vector<vector<int>>& matrix)</code>	המתודה טוענת גרף ממטריצת שכנויות המתקבלת כקלט. אם המטריצה ריקה, מוחזרת הודעת שגיאה; וכן גם אם המטריצה אינה ריבועית. בנוסף, המתודה משתמשת בפונקציות עזר כדי לסמן אם הגרף מכוון או לא; וכן כדי לספור את מספר הקודקודים בהתאם לסוג הגרף (מכוון או לא).
	פונקציות עזר: - <code>checkDirected()</code> - <code>size_t countEdges()</code>
<code>void Graph::printGraph() const</code>	המתודה מדפיסה כמה קודקודים וצלעות יש בגרף.

המתודה מחזירה כמה קודקודים קיימים בגרף.	size_t Graph::getNumVertices() const
המתודה מחזירה כמה צלעות קיימות בגרף.	size_t Graph::getNumEdges() const
המתודה מחזירה אם גרף הוא מכוון או לא.	bool Graph::isGraphDirected() const
המתודה מחזירה מצביע (reference) למטריצת השכנויות המייצגת כאמור גרף.	vector<vector<int>>& Graph::getAdjacencyMatrix()

הערה: חלק מהמתודות הוגדרו כ-*const*, כהגדרה, מאחר שהן אינן אמורות לשנות את האובייקט גרף.

בנוסף, המחלקה גרף כוללת את הפונקציות **הפרטיות** הבאות:

פונקציה	הסבר
bool checkDirected()	הפונקציה מחזירה אם גרף הוא מכוון או לא, על-ידי מעבר על כל הצלעות בגרף ובדיקה אם הערך במטריצת השכנויות $u \rightsquigarrow v$ שקול לערך $u \rightsquigarrow v$, או לא.
size_t countEdges()	הפונקציה מחזירה את מספר הצלעות בגרף, תוך שימת דגש אם הגרף מכוון או לא.

Algorithms

עבור המחלקה של האלגוריתמים קיימים שני קבצים: *Algorithms.cpp* ו-*Algorithms.hpp*.

המחלקה כוללת את המתודות **הפומביות** הבאות:

מתודה	הסבר
bool isConnected(Graph& graph)	המתודה בודקת אם גרף הוא קשיר או לא, בהנחה ששורש הגרף הוא הקודקוד 0. אם הגרף הוא בעל קודקוד אחד – מחזירה אמת מאחר שהגרף קשיר באופן טריוויאלי. אחרת, בודקת האם ניתן להגיע מקודקוד 0 לכל קודקוד אחר באמצעות האלגוריתם BFS. אם אחד הקודקודים לא "בוקר" מחזירה שקר; אחרת אמת.
	פונקציית עזר: - bfs(graph, startVertex, visited's vector)
bool isStronglyConnected(Graph& graph)	המתודה בודקת אם גרף הוא קשיר חזק או לא. אם הגרף הוא בעל קודקוד אחד – מחזירה אמת מאחר שהגרף קשיר חזק באופן טריוויאלי. אחרת, בודקת האם ניתן להגיע מכל קודקוד לכל קודקוד באמצעות האלגוריתם BFS. אם אחד מקודקודים הגרף באחת הריצות לא "בוקר" מחזירה שקר; אחרת אמת.
	פונקציית עזר: - bfs(graph, startVertex, visited's vector)
string shortestPath(Graph& graph, size_t start, size_t end)	המתודה בודקת מהו המסלול הקצר ביותר (אם קיים) בין קודקוד מקור לקודקוד יעד. אם קיים מסלול, המתודה מחזירה את המסלול כמחרוזת; אחרת מחזירה שלא קיים מסלול כזה. המתודה משתמשת באסטרטגיה הבאה כדי להחליט באיזה צורה לבצע את המשימה: אם הגרף אינו ממושקל, המתודה משתמשת באלגוריתם BFS; אם הגרף ממושקל אך עם משקלים חיוביים, המתודה משתמשת באלגוריתם של Dijkstra; ואם הגרף ממושקל אך עם צלעות בעלות משקל שלילי, המתודה משתמשת באלגוריתם של Bellman-Ford. המתודה גם בודקת תקינות קלט: (א) האם הקודקודים חוקיים (מאחר שנעשה שימוש בטיפוס <code>size_t</code> לא ניתן לקבל קודקוד מקור כשלילי); (ב) קודקוד המקור שונה מקודקוד היעד.
	פונקציית עזר: - bfsShortestPath(graph, start, end) - bellmanFordShortestPath(graph, start, end) - dijkstraShortestPath(graph, start, end) - checkGraphType(graph)

<p>המתודה בודקת אם הגרף קיים מעגל חיובי. אם קיים מעגל, מחזירה אותו כמחרוזת; אחרת מחזירה "0".</p> <p>על מנת לבצע את המשימה המתודה משתמשת באלגוריתם DFS. המתודה עוברת על כל קודקודי הגרף ומנסה למצוא מעגל באמצעות איתור back-edge המהווה אינדיקציה למעגל.</p>	<pre>string isContainsCycle(Graph& graph)</pre>
<p>פונקציית עזר:</p> <pre>dfs_cycle(graph, vertex, visited's vector, - recStack's vector, parent's vector, isDirected)</pre>	
<p>המתודה בודקת אם הגרף הוא דו-צדדי. אם ניתן לחלק את קודקודי הגרף לשתי קבוצות, כך שלא תהיה צלע בין הצלעות בכל קבוצה, המתודה מחזירה את החלוקה; אחרת, מחזירה שהגרף אינו דו-צדדי.</p> <p>על מנת לבצע את המשימה, המתודה משתמשת בתצורה שונה של אלגוריתם DFS הכוללת צביעת קודקודים.</p>	<pre>string isBipartite(Graph& graph)</pre>
<p>פונקציית עזר:</p> <pre>dfs_check(graph, currentVertex, colVec, color) - buildSet(set's vector) -</pre>	
<p>המתודה בודקת אם קיים מעגל שלילי בגרף. אם קיים מעגל שלילי, היא מחזירה אותו כמחרוזת; אחרת, מחזירה שאין מעגל שלילי בגרף. המתודה עושה שימוש בתצורה מסוימת של BF כאשר היא מנסה לבצע פעולת relax על צלעות הגרף.</p>	<pre>string negativeCycle(Graph& graph)</pre>
<p>פונקציית עזר:</p> <pre>findNegativeCircle(graph) -</pre>	

וכן את הפונקציות הפרטיות הבאות:

פונקציה	הסבר
<pre>void bfs(Graph& graph, size_t startVertex, vector<bool>& visited, vector<size_t>& parent, size_t end)</pre>	<p>פונקציית עזר זו מבצעת תהליך של BFS על קודקוד בגרף על מנת לבדוק אם ניתן להגיע ממנו לכל קודקודי הגרף. התוצאות נשמרות בווקטור המועבר לפונקציה כפרמטר בשם visited.</p> <p><u>הערך:</u> הפונקציה "הורחבה" כך שתקבל גם וקטור עבור לשמירת הורי הקודקודים וקודקוד יעד, על מנת שתהיה שמישה גם עבור מציאת המסלול הקצר ביותר בין שתי נקודות.</p>
<pre>isConnected(graph) - isStronglyConnected(graph) - shortestPath(graph, start, end) -</pre>	<p>מתודות קשורות:</p>
<pre>string bfsShortestPath(Graph& graph, size_t start, size_t end)</pre>	<p>פונקציית עזר זו מקבלת קודקוד מקור וקודקוד יעד ובודקת עבור גרף לא ממושקל אם קיים מסלול בין הקודקודים. אם קיים מסלול, הפונקציה מחזירה לפונקציית המקור את המסלול</p>

<p>כמחרוזת; אחרת, מחזירה שאין מסלול בין קודקוד המקור לקודקוד היעד. ביצוע התהליך ממומש באמצעות תור, שבאמצעותו בכל פעם בודקים את השכנים של הקודקוד שנמצא בראש התור, ומעדכנים בין היתר את האב על מנת לשחזר את המסלול הקצר (ככל שיימצא). שחזור המסלול מבוצע באמצעות פונקציית עזר.</p>	
<p>מתודות קשורות:</p> <p>- <code>shortestPath(graph, start, end)</code></p> <p>פונקציות עזר:</p> <p>- <code>bfs(graph, startVertex, visited's vector, parent's vector, end)</code></p> <p>- <code>buildPath(start, end, parent's vector)</code></p>	
<p>פונקציית עזר זו מקבלת קודקוד מקור וקודקוד יעד ובודקת עבור גרף ממושקל עם צלעות שליליות אם קיים מסלול בין הקודקודים. אם קיים מסלול, הפונקציה מחזירה לפונקציית המקור את המסלול כמחרוזת; אחרת, מחזירה שאין מסלול בין קודקוד המקור לקודקוד היעד. ביצוע התהליך מבוצע באמצעות ביצוע פעולת relax על קודקודי הגרף $V - 1$ פעמים, באמצעות פונקציית עזר המיועדת לכך. לאחר מכן, מבוצעת פעולה נוספת שכזו, המבוצעת גם היא באמצעות פונקציית עזר. ככל שלא נמצא מעגל שלילי, אזי נכנסת לפעולה פונקציית עזר נוספת שמחזירה את המעגל כמחרוזת.</p>	<pre>string bellmanFordShortestPath(Graph& graph, size_t start, size_t end)</pre>
<p>מתודות קשורות:</p> <p>- <code>shortestPath(graph, start, end)</code></p> <p>פונקציות עזר:</p> <p>- <code>relaxEdges(g, distance's vector, parent)</code></p> <p>- <code>buildPath(start, end, parent's vector)</code></p>	
<p>פונקציית עזר זו מקבלת קודקוד מקור וקודקוד יעד ובודקת עבור גרף ממושקל עם צלעות אי-שליליות אם קיים מסלול בין הקודקודים. אם קיים מסלול, הפונקציה מחזירה לפונקציית המקור את המסלול כמחרוזת; אחרת, מחזירה שאין מסלול בין קודקוד המקור לקודקוד היעד. ביצוע התהליך מבוצע באופן הבא: נעבור כל קודקודי הגרף; נמצא בכל פעם את הקודקוד עם המרחק הקטן ביותר, באמצעות פונקציית עזר, ונצבע פעולת relax על הקודקודים שטרם "בוקרו". נחזור על התהליך האמור עבור כל הקודקודים. אם בסוף התהליך, קודקוד היעד הוא במרחק של <code>INT_MAX</code> מקודקוד המקור, הרי שאין מסלול. אחרת, נשתמש בפונקציית עזר לשחזור המסלול הקצר ביותר באמצעות שמירת ההורה של כל קודקוד שנעשתה בתהליך הריצה.</p>	<pre>String dijkstraShortestPath(Graph& graph, size_t start, size_t end)</pre>

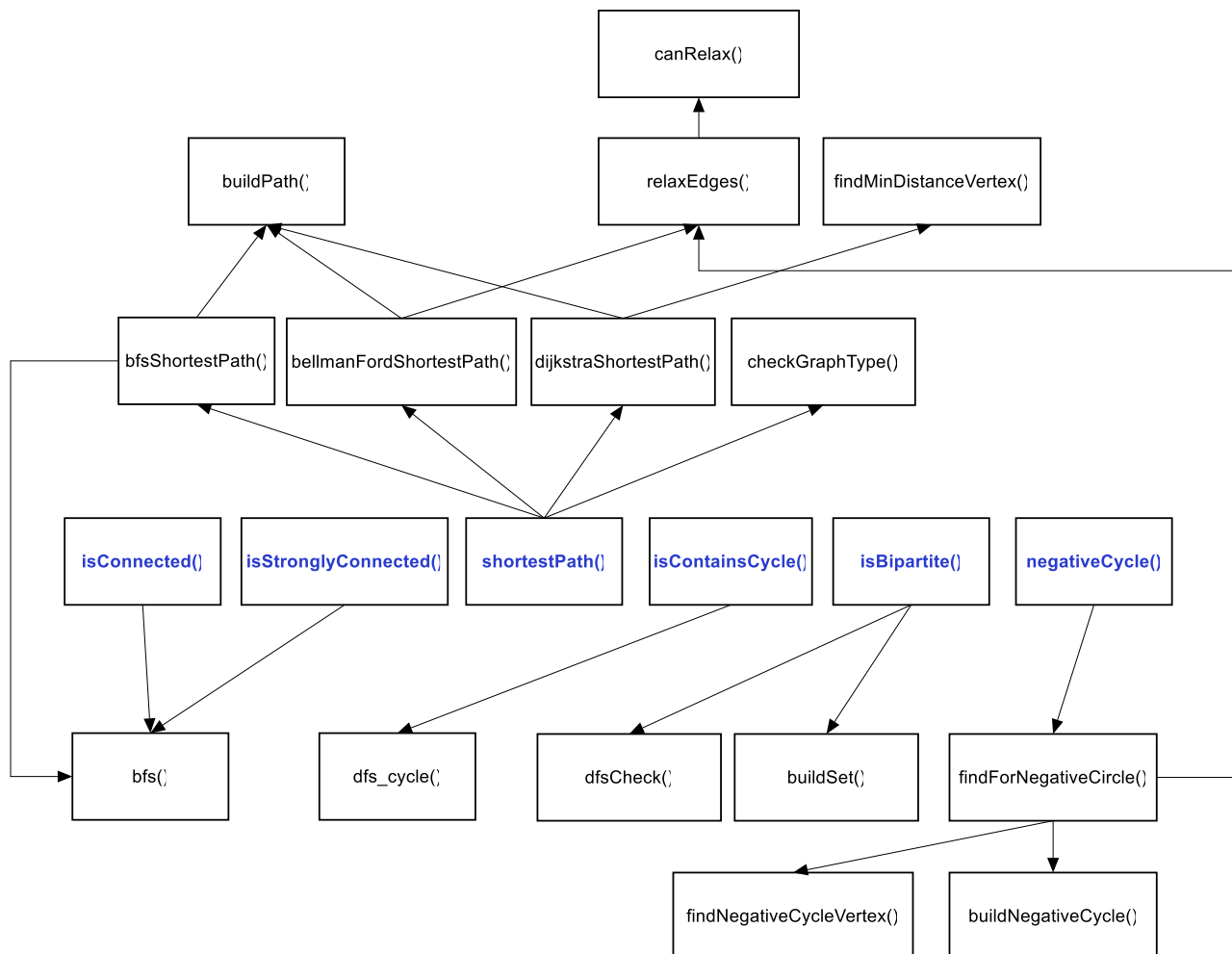
<p>מתודות קשורות:</p> <p>- <code>shortestPath(graph, start, end)</code></p> <p>פונקציות עזר:</p> <p>- <code>findMinDistanceVertex(distance's vector, visited's vector)</code></p> <p>- <code>buildPath(start, end, parent's vector)</code></p>	
<p>פונקציית עזר זו בודקת אם הגרף ממושקל, וכן אם קיימת צלע קטנה מ-0. אם הגרף ממושקל, מחזירה אמת; אחרת, שקר. התהליך הוא בדיקת כל צלעות הגרף כך שהן לא 0 (משמע, לא קיימת צלע) וגם לא 1 (משמע, הצלע חסרת משקל). בנוסף, אם נמצאה צלע שלילי, מחזירה אמת; אחרת, שקר. הפונקציה מחזירה את התוצאה כזוג של משתנים בוליאניים.</p>	<p><code>pair<bool, bool></code> <code>checkGraphType(Graph& graph)</code></p>
<p>מתודות קשורות:</p> <p>- <code>shortestPath(graph, start, end)</code></p>	
<p>פונקציית עזר זו מבצעת ריצה של DFS באופן רקורסיבי על מנת לאתר מעגל בגרף, באמצעות ניסיון לאתר back-edge (כאשר מדובר בגרף לא מכוון). ככל שנמצאת צלע שכזו, הפונקציה בונה את המעגל תוך התחקות אחרת הורי-הקודקודים; אחרת מחזירה ערך ריק. אם הגרף הוא מכוון, אזי הפונקציה מחפש מעגל פשוט.</p>	<p><code>string</code> <code>dfs_cycle(Graph& graph, size_t vertex, vector<bool>& visited, vector<bool>& recStack, vector<size_t>& parent, bool isDirected)</code></p>
<p>מתודות קשורות:</p> <p>- <code>isContainsCycle(Graph& graph)</code></p>	
<p>פונקציית עזר זו מקבלת בין היתר גרף, צביעה של קודקוד התחלה וקודקוד התחלה, ובודקת אם הגרף הוא 2-צביע. זאת, באמצעות תהליך של מעבר על כל קודקודי הגרף וצביעתם לסירוגין. אם נוצרת סתירה בצביעה – משמע שני קודקודים סמוכים נצבעו באותו צבע – הפונקציה מחזירה שקר; אחרת, אמת.</p>	<p><code>bool</code> <code>dfsCheck(Graph& graph, size_t currentVertex, vector<int>& colorVec, int color)</code></p>
<p>מתודות קשורות:</p> <p>- <code>isBipartite(graph)</code></p>	
<p>פונקציית עזר זו מקבלת קבוצה של קודקודים, ומחזירה אותם כמחרוזות.</p>	<p><code>string</code> <code>buildSet(vector<size_t>& set)</code></p>
<p>מתודות קשורות:</p> <p>- <code>isBipartite(graph)</code></p>	
<p>פונקציית עזר זו מנסה לאתר מעגל בגרף באמצעות אלגוריתם Bellman-Ford (עם שינוי מסוים, שכן האלגוריתם הקלאסי מאתר מסלול בין קודקוד מקור לקודקוד יעד). בפונקציה מבצעים פעולת relax על כל הקודקודים, ואם הפעולה האחרונה מביאה ליצירת מעגל, אזי הפונקציה מתחקה אחר הורי</p>	<p><code>String</code> <code>findNegativeCircle(Graph& graph)</code></p>

הקודקודים כדי לשחזר אותו. זהו גם הערך המוחזר אם קיים; אחרת, הפונקציה מחזירה שאין מעגל שלילי.	
מתודות קשורות: - <code>negativeCycle(graph)</code> פונקציות קשורות: - <code>relaxEdges(graph, distance's vector, parent's vector)</code> - <code>findNegativeCycleVertex(graph, distance's vector, parent's vector)</code> - <code>buildNegativeCycle(graph, vertex, parent's vector)</code>	
פונקציית עזר זו מקבל קודקוד מקור ויעד, וכן וקטור של הורי הקודקודים ומשחזרת את המסלול מקודקוד המקור לקודקוד היעד. זו גם הערך המוחזר.	String <code>buildPath(size_t start, size_t end, vector<size_t>& parent)</code>
פונקציות קשורות: - <code>bfsShortestPath(graph, start, end)</code> - <code>bellmanFordShortestPath(g, start, end)</code> - <code>dijkstraShortestPath(graph, start, end)</code>	
פונקציית עזר זו מבצעת פעולת relax על קודקודי גרף. בעבור פעולה זו היא משתמשת בפונקציית עזר.	bool <code>relaxEdges(Graph& graph, vector<int>& distance, vector<size_t>& parent)</code>
פונקציות קשורות: - <code>canRelax(graph, vertex_u, vertex_v, weight, distance's vector, parent's vector)</code>	
פונקציית עזר זו בודקת מספר תנאים כדי לקבוע אם ניתן לבצע פעולת relax על צלע, ביניהן, אם משקלה לא 0, וגם אם המרחק שלה לא <code>INT_MAX</code> , וגם המרחק של <code>u</code> בתוספת הצלע <code>uv</code> קטן מהמרחק ישירות ל- <code>v</code> . כמו כן, נעשית בדיקה אם מדובר בגרף מכוון, ועבור גרף לא מכוון, אם ההורה של <code>u</code> אינו <code>v</code> כדי למנוע בדיקות כפולות.	bool <code>canRelax(Graph& graph, size_t vertex_u, size_t vertex_v, int weight, vector<int>& distance, vector<size_t>& parent)</code>
פונקציות קשורות: - <code>canRelax(graph, vertex_u, vertex_v, weight, distance's vector, parent's vector)</code>	
פונקציית עזר זו מאתרת קודקוד שנמצא על מעגל שלילי. הפונקציה עוברת על כל הצלעות לאחר שכבר בוצע להם פעולת relax. אם ניתן לבצע פעולה נוספת, הרי שהקודקוד שממנו יצאה הצלע הוא חלק ממעגל שלילי. קודקוד זה יוחזר.	size_t <code>findNegativeCycleVertex(Graph& graph, vector<int>& distance, vector<size_t>& parent)</code>
- <code>canRelax(graph, vertex_u, vertex_v, weight, distance's vector, parent's vector)</code>	

פונקציית עזר זו בונה מעגל שלילי שנמצא בגרף, ובונה ייצוג מחרוזות של המעגל בסדר הפוך מזה שנשמר. אם אכן נמצא מעגל שלילי, הפונקציה מחזירה מחרוזות של המעגל השלילי.

```
String
buildNegativeCycle(Graph&
graph, vector<int>& distance,
vector<size_t>& parent)
```

להלן תרשים המתאר את היחס בין המתודות והפונקציות השונות במחלקת האלגוריתמים:¹



¹ בעיצוב התרשימים נעשה שימוש באפליקציה של NCH.

Makefile

לצורך הרצת התוכנית והבדיקות נערך קובץ *makefile* כדלקמן:

```
# General macros
CXX = clang++
CXXFLAGS = -std=c++11 -Werror -Wsign-conversion
VALGRIND_FLAGS = -v --leak-check=full --show-leak-kinds=all --error-
exitcode=99

# Macros for source files and headers files
SOURCES = Graph.cpp Algorithms.cpp
HEADERS = Graph.hpp Algorithms.hpp
DEMO_SRC = Demo.cpp
TEST_SRC = Test.cpp
TEST_COUNTER_SRC = TestCounter.cpp

# Macros for object and headers files
OBJECTS = Graph.o Algorithms.o

# Main target: Build and run the demo
run: demo
    ./demo

# Build the demo exe file from object files
demo: Demo.o $(OBJECTS)
    $(CXX) $(CXXFLAGS) Demo.o $(OBJECTS) -o demo

# Build the test exe file that includes the tests
test: TestCounter.o Test.o $(OBJECTS)
    $(CXX) $(CXXFLAGS) TestCounter.o Test.o $(OBJECTS) -o test

# Run clang-tidy
tidy:
    clang-tidy $(SOURCES) -checks=bugprone-*,clang-analyzer-
*,cppcoreguidelines-*,performance-*,portability-*,readability-*,
cppcoreguidelines-pro-bounds-pointer-arithmetic,-cppcoreguidelines-owning-
memory --warnings-as-errors=-* --

# Run valgrind
valgrind: demo test
    valgrind --tool=memcheck $(VALGRIND_FLAGS) ./demo 2>&1 | { egrep "lost|
at " || true; }
    valgrind --tool=memcheck $(VALGRIND_FLAGS) ./test 2>&1 | { egrep "lost|
at " || true; }

# Rule to compile Graph object file
Graph.o: Graph.cpp Graph.hpp
    $(CXX) $(CXXFLAGS) -c Graph.cpp -o Graph.o
```

```
# Rule to compile Algorithms object file
Algorithms.o: Algorithms.cpp Algorithms.hpp
    $(CXX) $(CXXFLAGS) -c Algorithms.cpp -o Algorithms.o

# Rule to compile Demo object file
Demo.o: Demo.cpp $(HEADERS)
    $(CXX) $(CXXFLAGS) -c Demo.cpp -o Demo.o

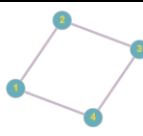
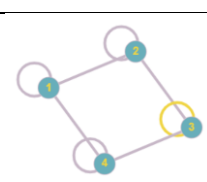
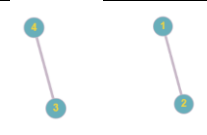
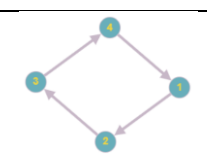
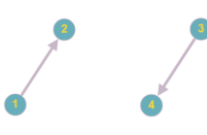
# Rule to compile Test object file
Test.o: Test.cpp $(HEADERS)
    $(CXX) $(CXXFLAGS) -c Test.cpp -o Test.o

# Rule to compile TestCounter object file
TestCounter.o: TestCounter.cpp $(HEADERS)
    $(CXX) $(CXXFLAGS) -c TestCounter.cpp -o TestCounter.o


# Clean up command to remove all compiled files
clean:
    rm -f *.o demo test
```

הבדיקות בוצעו עבור המחלקות Graph ו-Algorithms².




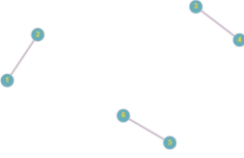


בדיקות עבור Graph


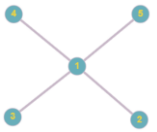







מס'	שם הבדיקה	הקלט לבדיקה	מהות הבדיקה
1	Test empty graph	מופע לא מאותחל של גרף (משמע, הפעלת בנאי ריק)	האם מספר הקודקודים והצלעות הוא 0? האם מטריצת השכנויות ריקה
2	Test undirected graph		האם יש בגרף 4 קודקודים ו-4 צלעות? האם טעינת המטריצה התבצעה בהצלחה? האם הגרף לא מכוון?
3	Test undirected graph with self-loops		האם יש בגרף 4 קודקודים ו-8 צלעות? האם טעינת המטריצה התבצעה בהצלחה? האם הגרף לא מכוון?
4	Test disconnected undirected graph		האם יש בגרף 4 קודקודים ו-2 צלעות? האם טעינת המטריצה התבצעה בהצלחה? האם הגרף לא מכוון?
5	Test directed graph		האם יש בגרף 4 קודקודים ו-4 צלעות? האם טעינת המטריצה התבצעה בהצלחה? האם הגרף מכוון?
6	Test disconnected directed graph		האם יש בגרף 4 קודקודים ו-2 צלעות? האם טעינת המטריצה התבצעה בהצלחה? האם הגרף מכוון?
7	Test invalid graph	$\{\{0,1,0\}, \{0,1,0,1\}, \{0,1,0\}\}$	האם נזרקה השגיאה: Invalid graph: The graph is not a square matrix
8	Test invalid graph with empty matrix	מופע לא מאותחל של גרף	האם נזרקה השגיאה: Invalid graph: The graph matrix is empty
9	Test invalid graph: non-square matrix	$\{\{0,2,1,0\}, \{0,3,0,1\}, \{4,0,3,2\}, \{0,4,0,0\}, \{0,0,0,5\}\}$	האם נזרקה השגיאה: Invalid graph: The graph is not a square matrix


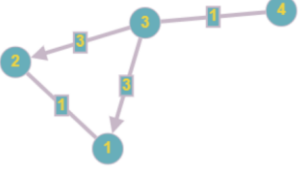

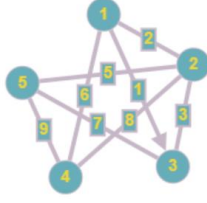


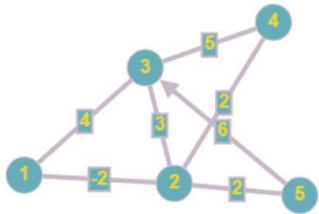

² עבור ציור הגרפים נעשה שימוש בכלי שנמצא באתר: <https://graphonline.ru/en>.


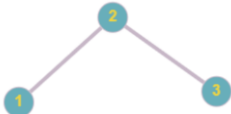

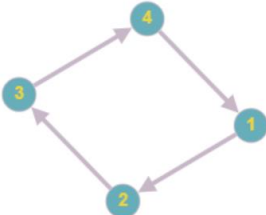
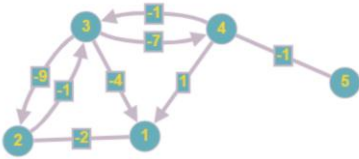


	וגם: $\{1, 0, 2, 1, 0\},$ $\{2, 0, 3, 0, 1\},$ $\{4, 4, 0, 3, 2\},$ $\{0, 0, 4, 0, 3\}$		
האם לא נזרקה שגיאה מכל סוג? לא.		Test invalid graph: Graph with 1 vertex and 0 edges is valid	10

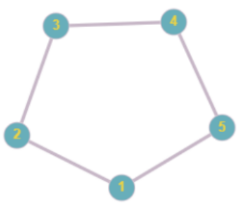


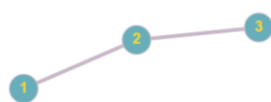

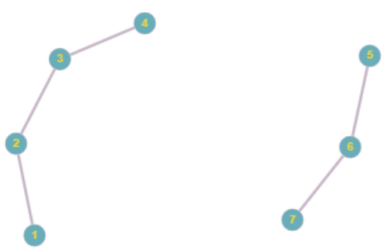
בדיקות עבור Algorithms

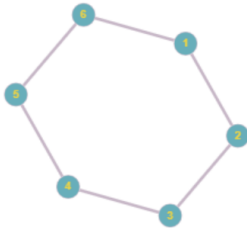
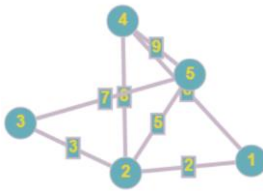


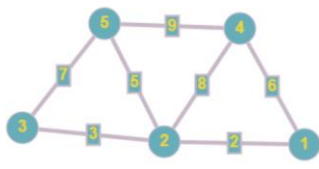
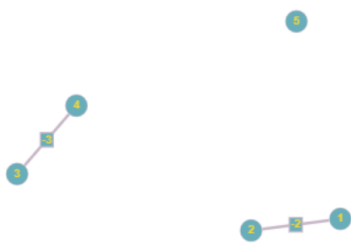

מס'	שם הבדיקה	הקלט לבדיקה	מהות הבדיקה
<i>isConnected</i>			
1	Test isConnected: Single vertex graph		האם גרף של קודקוד אחד קשיר באופן טריוויאלי?
2	Test isConnected: Undirected graph		האם גרף לא מכוון קשיר?
3	Test isConnected: Undirected Disconnected graph		האם לא גרף מכוון לא קשיר?
4	Test isConnected: Disconnected graph with multiple components		האם לא גרף מכוון עם יותר רכיבי קשירות לא קשיר?
5	Test isConnected: Graph with self-loops		האם גרף לא קשיר עם לולאות-עצמיות קשיר?
6	Test isConnected: Graph with a single edge		האם גרף עם צלע אחת קשיר?

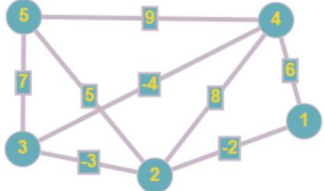
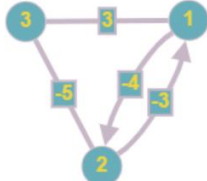
האם גרף מלא קשיר?		Test isConnected: Complete graph	7
האם גרף כוכב קשיר?		Test isConnected: Star graph	8
האם גרף עם קודקוד בודד קשיר?		Test isConnected: Graph with a single isolated vertex	9
האם גרף של קודקודים בודדים קשיר?		Test isConnected: Graph with all vertices isolated	10
האם גרף שלכל הקודקודים יש לולאה עצמית קשיר?		Test isConnected: Graph with all vertices connected to themselves	11
<i>isStronglyConnected</i>			
האם גרף עם צלעות שליליות קשיר חזק?		Test isStronglyConnected: Graph with negative edges	12
האם גרף מכוון עם צלע מכוונת אחת קשיר חזק? לא.		Test isStronglyConnected: Directed graph with one edge	13
האם גרף מכוון עם קודקוד בודד קשיר חזק? לא.		Test isStronglyConnected: Directed graph with one isolated vertex	14
האם גרף מכוון לשני הצדדים קשיר חזק? אכן.		Test isStronglyConnected: Directed graph	15

shortestPath			
אין מסלול בין קודקוד לעצמו		Test shortestPath: Single vertex graph	16
בדיקת BFS: האם יש מסלול בין מ-1 ל-4? לא.		Test shortestPath (BFS): No path exists	17
בדיקת BFS: האם יש מסלול מ-1 ל-4? אכן.		Test shortestPath (BFS): Multiple paths exist	18
בדיקת Dijkstra: האם יש מסלול בין 1 ל-4? אכן.		Test shortestPath (Dijkstra): Shortest path with weighted edges	19
בדיקת BFS: האם יש מסלול ארוך בין 1 ל-4? אכן		Test shortestPath: Graph with a longer path	20
בדיקת Dijkstra: האם יש מסלול בין 1 ל-5? האם יש מסלול בין 2 ל-3? האם יש מסלול בין 3 ל-1? האם יש מסלול בין 5 ל-1?		Test shortestPath (Dijkstra): Graph with multiple paths	21
בדיקת BF: האם קיים מסלול מ-1 ל-3? האם קיים מסלול מ-2 ל-4? האם קיים מסלול מ-3 ל-5? האם קיים מסלול מ-4 ל-1?		Test shortestPath (BF): Graph with multiple paths	22
האם קיים מסלול מ-1 ל-5? או מ-5 ל-1? נצפה לקבל: Invalid start or end vertex		Test shortestPath (Dijkstra): Invalid vertex	23

<p>האם קיים מסלול בין 1 ל-1 (כאשר ל-1 אין לולאה לעצמו)? נצפה לקבל:</p> <p>No path exists between a vertex and itself</p>			
<i>isContainsCircle</i>			
<p>האם קודקוד בודד לא כולל מעגל?</p>		Test isContainsCycle: Single vertex graph	24
<p>האם בגרף הלא מכוון אין מעגל?</p>		Test isContainsCycle: Undirected graph with no cycle	25
<p>האם בגרף הלא מכוון יש מעגל של 1, 2, 3, 1?</p>		Test isContainsCycle: Undirected graph with a cycle	26
<p>האם בגרף המכוון יש מעגל למשל 1, 2, 3, 4, 1?</p>		Test isContainsCycle: Directed graph with a cycle	27
<p>האם הגרף כולל מעגל למשל 1, 2, 3, 1?</p>		isContainsCycle: Complex graph with cycles	28
<p>האם קיים מעגל בין קודקוד לעצמו?</p>		Test isContainsCycle: Graph with self-loop	29
<p>האם קיים מעגל הכולל את כל הקודקודים?</p>		Large graph with long cycle	30

<i>isBipartite</i>			
<p>האם גרף שכולל מעגל אי-זוגי הוא גרף דו"צ? נצפה להדפסה:</p> <p>The graph is not bipartite</p>		<p>Test isBipartite: Graph with an odd cycle (not bipartite)</p>	31
<p>האם גרף עם קודקוד אחד הוא גרף דו"צ? נצפה להדפסה:</p> <p>The graph is bipartite: A={1}, B={}</p>		<p>Test isBipartite: Single vertex graph</p>	32
<p>האם גרף ללא צלעות דו"צ? נצפה להדפסה:</p> <p>The graph is bipartite: A={0,1}, B={}</p>		<p>Test isBipartite: Graph without edges</p>	33
<p>האם הגרף דו"צ? נצפה להדפסה:</p> <p>The graph is bipartite: A={1,3}, B={2}</p>		<p>Test isBipartite: Bipartite graph</p>	34
<p>האם גרף דו"צ? נצפה להדפסה:</p> <p>The graph is not bipartite</p>		<p>Test isBipartite: Non-bipartite graph</p>	35
<p>האם גרף דו"צ? נצפה להדפסה:</p> <p>The graph is bipartite: A={1,3,5,7}, B={2,4,6} Or The graph is bipartite: A={1,3,6}, B={2,4,5,7}</p>		<p>Test isBipartite: Bipartite graph with connected components</p>	36

<p>האם גרף דו"צ? נצפה להדפסה:</p> <p>The graph is bipartite: A={0,2,4}, B={1,3,5}</p>		<p>Test isBipartite: Bipartite graph with an even cycle</p>	37
negativeCycle			
<p>האם הגרף מכיל מעגל שלילי? נצפה לקבל:</p> <p>No negative cycle exists</p>		<p>Test negativeCycle: Graph with negative weights and a positive cycle</p>	37
<p>האם הגרף מכיל מעגל שלילי? נצפה לקבל:</p> <p>No negative cycle exists</p>		<p>Test negativeCycle: Graph with negative weights but no cycle</p>	38
<p>האם הגרף מכיל מעגל שלילי? נצפה לקבל:</p> <p>No negative cycle exists</p>		<p>Test negativeCycle: Graph with negative weights and a zero-weight cycle</p>	39
<p>האם הגרף מכיל מעגל שלילי? נצפה לקבל:</p> <p>No negative cycle exists</p>		<p>Test negativeCycle: Graph with positive weights</p>	40
<p>האם הגרף מכיל מעגל שלילי? נצפה לקבל:</p> <p>No negative cycle exists</p>		<p>Test negativeCycle: Graph with negative weights and disconnected components</p>	41
<p>האם הגרף מכיל מעגל שלילי? אכן - 1,1</p>		<p>Test negativeCycle: Graph with negative weights and a self-loop</p>	42

<p>האם הגרף מכיל מעגל שלילי? אכן – 1, 4, 3, 2, 1 (קיימות בדיקות נוספות לסוג מקרים זה)</p>		<p>Test negativeCycle: Graph with negative cycle</p>	<p>43</p>
<p>האם הגרף מכיל מעגל שלילי? אכן – 1, 2, 1</p>		<p>Test negativeCycle: Directed weighted graph with cycle</p>	<p>44</p>