

**GIT Department of Computer Engineering**  
**CSE 222/505 - Spring 2018**  
**Homework 07**  
**Deadline: 29.05.2018 - 23:55**

**Q1:** Create directed acyclic graph have random weight ( $v=10$ ,  $e=20$ ), plot this graph using `plot_graph` function. Prove that using `is_undirected` and `is_acyclic_graph` functions. Then run `shortest_path` function on this graph, use least 3 different label pair (`shortest_path(g,v1,v2)`, `shortest_path(g,v3,v4)`, .....)

**Q2:** Create undirected and acyclic graph have no weight ( $v=15$ ), plot this graph using `plot_graph` function. Prove that using `is_undirected` and `is_acyclic_graph` functions. Then run `is_connected` function on this graph, use least 3 different label pair ( `is_connected(g,v1,v2)`, `is_connected(g,v3,v4)`, .....)

**Q3:** Create undirected and cyclic graph have no weight ( $v=10$ ), plot this graph using `plot_graph` function. Prove that using `is_undirected` and `is_acyclic_graph` functions. Then run `DepthFirstSearch` and `BreathFirstSearch` functions on text book and plot spanning trees.

**Q4:** This answer of this question should be **only 1 page**. Explain what is the differences of BFS and DFS. (usage areas, advantages, ...). Consider the undirected graph below which is represented by its adjacency matrix.

- Run the DFS algorithm starting from vertex 1, and draw the DFS tree.
- Run the BFS algorithm starting from vertex 1, and draw the BFS tree.

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

---

Function - `is_connected`

- Input** - `g`, a graph object; `v1`, a vertex label in `g`; `v2`, a vertex label in `g`.
- Output** - **TRUE** if there is a path from `v1` to `v2` in `g`, **FALSE** if not.
- Description - Determine if there is any path between vertex `v1` and vertex `v2` in graph `g`. If `v1` or `v2` are not in `g` then throw an error.

Function - `shortest_path`

- Input** - `g`, graph object; `v1`, a vertex label in `g`; `v2`, a vertex label in `g`.
- Output** - **path**, a vector of the names of vertices that make up the shortest path, in order. If there is no path between the vertices then return an empty vector; **distance**, total weight of path.
- Description - Find the shortest path from vertex `v1` to vertex `v2` using Dijkstra's algorithm. Note that there may not be a unique solution for any given graph, you are only required to return one path.

Function - is\_undirected

- **Input** - g, a graph object.
- **Output** - **TRUE** if g is undirected, **FALSE** if not.
- **Description** - Check if the graph object is undirected, this is true if all directed edges have a complementary directed edge with the same weight in the opposite direction.

Function - is\_acyclic\_graph

- **Input** - g, a graph object.
- **Output** - **TRUE** if g is undirected, **FALSE** if not.
- **Description** - The graph may or may not have cycles. To check do a graph traversal (BFS or DFS).

Function - plot\_graph

- **Input** - g, a graph object
- **Output** - plot showing all vertices (labeled) and edges.
- **Description** - This function should be able to take any graph object and produce a reasonably attractive visual representation of that graph. Your algorithm should make use edge weights to layout the distance between vertices.

---

Book Student source code:

<http://bcs.wiley.com/he-bcs/Books?action=resource&bcsId=5643&itemId=0470128704&resourceId=21295>

Note:

- Obey OOP principles and clean code standards.
- Write a main and maintest for each function
- Your submission is studentnumber.zip and include following files:
  - o intelliJ project file
    - o Q1 folder
    - o Q2 folder
    - o Q3 folder
  - o Report.pdf
  - o Javadoc
- The report must be in format "ReportFormat\_hw7"
- For contact and questions "fesirci@gtu.edu.tr"