Homework 4 Test Cases

Introduction In this document, we will present the test cases for homework 4. They are divided into two parts:

- Part 1: Verify the implementation of the Graph data structure
- Part 2: Verify the implementation of the Dijkstra's algorithm for single-source shortest path algorithm

Note that the `csv` files for the graphs G_1, G_2, and G_3 are posted in the homework 4 folder:

Name	Type	Image	Related `csv` files
G_1	undirected, unweighted	`hw4-fig1.png`	`fig1.csv`
G_2	directed, unweighted	`hw4-fig2.png`	`fig2.csv`
G_3	directed, weighted	`hw4-fig3.png`	`fig3.csv` and `fig3-w.csv`

We will denote the empty directed graph by D_e and the empty undirected graph by U_e.

Part 1 Verify the implementation of the Graph data structure:

In test cases 1, 2, and 3, you may create constructor functions which will construct graphs by retrieving the required data from the given `csv` files.

Test Case 1 Reading `csv` files for graphs (undirected, unweighted) and output `dot` files for those graphs

Create a Graph object for G_1 by reading the required data from the corresponding `csv` file(s). Show the graph by producing the `dot` file `t1.dot`.

Test Case 2 Reading `csv` files for graphs (directed, unweighted) and output `dot` files for those graphs (with direction shown)

Create a Graph object for G_2 by reading the required data from the corresponding `csv` file(s). Show the graph by producing the `dot` file `t2.dot`. Directions of edges must be shown in the image generated from the `dot` file.

Test Case 3 Reading `csv` files for graphs (directed, weighted) and output `dot` files for those graphs (with both weights and directions shown)

Create a Graph object for G_3 by reading the required data from the corresponding `csv` file(s). Show the graph by producing the `dot` file `t3.dot`. Direction and weight for each edge must be shown in the image generated from the `dot` file.

Test Case 4 Use functions to add/remove vertices and/or edges to an existing graph (unweighted)

Carry out the following steps in the order specified:

1. Starting from the graph G_1, insert the vertex `z` to G_1.
2. Insert the (undirected) edges

   `(z,w), (z,x), (z,y)`

3. Show the resulting graph by creating a `dot` file (`t4a.dot`).
4. Remove the vertices `s` and then `x` (and of course, all the associated edges).
5. Remove the edge $(\mathtt{u}, \mathtt{t})$.
6. Show the resulting graph by creating a `dot` file (`t4b.dot`).

Test Case 5 Use functions to add/remove vertices and/or edges to the empty graph (undirected)

Carry out the following steps in the order specified:

1. Starting from the graph U_e, insert the vertices `6, ..., 10`.
2. Insert the (undirected) edges

   `(6,7), (7,8), (8,9), (9,10), (10,6)`

   Insert the vertices `1, ..., 5`.
3. Insert the (undirected) edges

   `(1,6), (2,7), (3,8), (4,9), (5,10)`

4. Show the resulting graph by creating a `dot` file (`t5a.dot`).
5. Remove the vertex `8` followed by vertex `6` (and of course, all the associated edges).
6. Show the resulting graph by creating a `dot` file (`t5b.dot`).

Test Case 6 Use functions to add/remove vertices and/or edges to the empty graph (directed)

Carry out the following steps in the order specified:

1. Starting from the graph D_e, insert the vertices `2, 4, 6, 8, 10` in random order to D_e.
2. Insert the (directed) edges

   `(2,4), (2,6), (4,6), (4,8), (6,8), (6,10), (8,10), (8,2)`

3. Insert the vertices `1, 3, 5, 7, 9` in random order.
4. Insert the (directed) edges

   `(1,2), (3,4), (5,6), (7,8), (9,10)`

5. Show the resulting graph by creating a `dot` file (`t6a.dot`).
6. Randomly choose a vertex from `2, 4, 6, 8, 10`, remove it and all of its associated edges.
7. Randomly choose a vertex from `1, 3, 5, 7, 9`, remove it and all of its associated edges.
8. Show the resulting graph by creating a `dot` file (`t6b.dot`).

Part 2 Verify the implementation of the Dijkstra's algorithm

In this group of test cases, we will apply Dijkstra's algorithm for solving single-source shortest path problems for both undirected graphs and directed graph. Let G_4 be the undirected graph by removing all the directions from the graph G_3 and the weight associated to an edge (v_1, v_2) in G_4, $w(v_1, v_2)$ is defined as follows:

$$w(v_1, v_2) = \begin{cases} \infty & \text{if there is no edge between } v_1, v_2 \text{ in } G_3 \\ \alpha & \text{if there is exactly one edge between } v_1, v_2 \text{ with weight } \alpha \text{ in } G_3 \\ \text{min. } \{\beta, \gamma\} & \text{if the weight of the directed edge } (v_1, v_2) \text{ is } \beta \text{ in } G_3 \\ & \text{and the weight of the directed edge } (v_2, v_1) \text{ is } \gamma \text{ in } G_3 \end{cases}$$

the same as the directed version. We will carry the following tasks to the graphs G_3 and G_4:

Tasks	G_3	G_4
Compute the shortest distance via Dijkstra's algorithm	Test Case 7	Test Case 8
Compute the shortest path via Dijkstra's algorithm	Test Case 9	Test Case 10
Visual Display for the shortest paths	Test Case 11	Test Case 12

Test Case 7 Use Dijkstra's algorithm to compute the shortest distance between any pairs of vertices in the graph G_3. Print the results (to the screen) in the form of table as shown below (with all the blanks filled):

	s	t	x	y	z
s	0	d_1			
t	d_2	0			
x			0		
y				0	
z					0

Note that d_1 should store the shortest distance from vertex s to vertex t and d_2 should store the shortest distance from vertex t to vertex s which may not be the same.

Test Case 8 Use Dijkstra's algorithm to compute the shortest distance between any pairs of vertices in the graph G_4. Print the results (to the screen) in the form of table as shown below (with all the blanks filled):

	s	t	x	y	z
s	0				
t		0			
x			0		
y				0	
z					0

Test Case 9 Use Dijkstra's algorithm to compute the shortest paths from the source vertex s to all other vertices in the graph G_3. Print the results (to the screen) in the form of table as shown below. Repeat the same experiment with the vertex z as source vertex. Again, Print the results to the screen in the form of a table as shown.

Vertex	The path from source vertex s to this vertex in G_3
s	nil
t	$\cdots$
x	$\cdots$
y	$\cdots$
z	$\cdots$

Vertex	The path from source vertex z to this vertex in G_3
s	$\cdots$
t	$\cdots$
x	$\cdots$
y	$\cdots$
z	nil

Test Case 10 Use Dijkstra's algorithm to compute the shortest paths from the source vertex s to all other vertices in the graph G_4. Print the results (to the screen) in the form of table as shown below. Repeat the same experiment with the vertex z as source vertex. Again, Print the results to the screen in the form of a table as shown.

Vertex	The path from source vertex s to this vertex in G_4
s	nil
t	$\cdots$
x	$\cdots$
y	$\cdots$
z	$\cdots$

Vertex	The path from source vertex z to this vertex in G_4
s	$\cdots$
t	$\cdots$
x	$\cdots$
y	$\cdots$
z	nil

Test Case 11 Use Dijkstra's algorithm to compute the following path P in graph G_3:

1. P is a shortest path from a vertex v_1 to another vertex v_2 in G_3.
2. The cost of the P, which is the sum of the weights of the edges in P, is maximum among all shortest paths between any two distinct vertices in G_3.

Show the path P by producing a `dot` file `t11.dot` for the graph G_3 and print that path to the screen. For extra credit, show the path by coloring each of its edges in *red*. Your `dot` file will produce a diagram to show that.

Test Case 12 Use Dijkstra's algorithm to compute the following path P in graph G_4:

1. P is a shortest path from a vertex v_1 to another vertex v_2 in G_4.
2. The cost of the P, which is the sum of the weights of the edges in P, is maximum among all shortest paths between any two distinct vertices in G_4.

Show the path P by producing a `dot` file `t12.dot` for the graph G_4 and print that path to the screen. For extra credit, show the path by coloring each of its edges in *blue*. Your `dot` file will produce a diagram to show that.