Assignment 9.1: Creating a Graph (4 points)

A graph has n vertices but no edges between them yet. Your task is to implement a simple graph structure in Python. At any time, the graph can be modified by creating or removing edges between the vertices, or traversed using depth-first traversal method.

Create a class `Graph` which has the following methods:

- `__init__(n)`: initialize a new `Graph` object with `n` vertices
- `dft(start)`: traverses and prints the graph in depth-first order, from `start` vertex.
- `add(u, v)` (**2 pts**): creates an undirected edge between vertices `u` and `v`
- `remove(u, v)` (**2 pts**): removes an edge between vertices `u` and `v`

Vertices in the graph are labeled from 0 to $n-1$ as integers (`int`).

A code template with an example program:

```
# graph.py


class Graph:
    # TODO


if __name__ == "__main__":
    graph = Graph(6)
    vertices = ((0, 2), (0, 4), (2, 1),
                (2, 3), (2, 5), (3, 0),
                (3, 5), (4, 5), (5, 1))
    for u, v in vertices:
        graph.add(u, v)

    graph.dft(0)

    graph.remove(0, 2)
    graph.remove(2, 5)
    graph.remove(1, 4)

    graph.dft(0)
```

```
$ python graph.py
0 2 1 5 3 4
0 3 2 1 5 4
```

Submit your solution in CodeGrade as `graph.py`.

Assignment 9.2: Shortest path (4 points)

A new type of graph emerges where each edge has a specific weight and direction. When traversing this graph, it is important to find the least resource-consuming path, which means the path with the smallest sum of weights. This corresponds to finding the shortest path, with the weights representing the distances between the vertices.

Create a class `Graph` which has all the following methods:

- `__init__(n)`: initialize a new `Graph` object with `n` vertices.
- `add(u, v, w)`: creates a *directed* edge from vertex `u` to `v` with weight `w`
- `shortest_path(start, end)`: finds the path with the least total weight (the shortest path) from `start` to `end` and prints the traversed path including start and end vertices, and `-1` if not found.

A code template with an example program:

```
# shortestpath.py


class Graph:
    # TODO


if __name__ == "__main__":

    graph = Graph(10)
    edges = ((0, 1, 25), (0, 2,  6), (1, 3, 10),
                (1, 4,  3), (2, 3,  7), (2, 5, 25),
                (3, 4, 12), (3, 5, 15), (3, 6,  4),
                (3, 7, 15), (3, 8, 20), (4, 7,  2),
                (5, 8,  2), (6, 7,  8), (6, 8, 13),
                (6, 9, 15), (7, 9,  5), (8, 9,  1))
    for u, v, w in edges:
        graph.add(u, v, w)

    graph.shortest_path(0, 9)
```

Output:

```
$ python shortestpath.py
0 2 3 6 7 9
```

Submit your solution in CodeGrade as `shortestpath.py`.

Last modified: Thursday, 21 November 2024, 6:41 PM