Documentation Assignment 1

Specification:

We shall define a class named *Graph* representing a directed graph.

The class *Graph* will provide the following methods:

- __init__(self)
  Creates an empty graph.
- add_vertex_in (self, vertex_in, vertex_out)
  Will add vertex_out to the dictionary vertices_in at key vertex_in.
- add_vertex_out(self, vertex_out, vertex_in
  Will add vertex_in to the dictionary vertices_out at key vertex_out.
- read_edge(self, origin, end, cost)
  Adds an edge read from the file (the edge is correct).
- add_edge(self, origin, end, cost)
  Adds an edge to the graph.
  Preconditions: The edge does not exist already (no edge between origin and end) and both the vertices exist.
- add_empty_vertex(self, vertex)
  Adds a new vertex to the graph.
  Preconditions: The vertex does not exist already.
- number_vertices(self)
  Returns the number of vertices.
- number_edges(self)
  Returns the number of edges.
- vertices_set(self)
  Returns a set of all vertices.
- edges_set(self)
  Returns a set of all edges.
- search_edge(self, origin, end)
  Searches if the edge between vertices origin and end exist.
- remove_vertex(self, vertex)
  Removes the vertex from the graph.
  Preconditions: The vertex exists.
- remove_edge(self, origin, end)

Removes the edge between origin and end from the graph.
Preconditions: The edge exists.

- change_cost(self, origin, end, cost)
  Changes the cost of the edge between origin and end.
  Preconditons: The edge exists.
- in_degree_vertex(self, vertex)
  Returns the in degree of the vertex.
  Preconditions: The vertex exists.
- out_degree_vertex(self, vertex)
  Returns the out degree of the vertex.
  Preconditions: The vertex exists.
- outbound_edges(self, vertex)
  Returns the set of vertices towards which this vertex has an outbound edge.
  Preconditions: The vertex exists.
- inbound_edges(self, vertex)
  Returns the set of vertices towards which this vertex has an inbound edge.
  Preconditions: The vertex exists.
- get_cost(self, origin, end)
  Returns the cost of the edge between origin and end.
  Preconditions: The edge exists.
- copy_graph(self)
  Returns a deep copy of the graph.

Implementation:

Class *Graph* will have the following data members:

- vertices_in
  Dictionary containing all vertices as keys and the vertices towards which they have an inbound edge as values.
- vertices_out
  Dictionary containing all vertices as keys and the vertices towards which they have an outbound edge as values.
- edges
  Dictionary containing all edges as keys (represented as pairs of vertices) and their costs as values.
- number_vertices
  The number of vertices.
- number_edges
  The number of edges.