Specification

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

The class **DirectedGraph** will provide following methods:

DirectedGraph(int size = 0)
: Constructs a graph with *size* vertices, the default size is 0

get_vertices(): int
: Returns the number of vertices of the graph

get_edges(): int
: Returns the number of edges of the graph

add_edge(int from_vertex, int to_vertex, int cost)
: Adds an edge to the graph

parse_vertices(): list
: Returns an iterable list of vertices

is_edge_defined(int having_start, int having_end): boolean
: Returns true if an edge is defined by the start and end vertices and false otherwise

get_vertex_in_out(int vertex): (int, int)
: Returns a pair of total in degree and out degree vertices for a given vertex

parse_vertex_in(int vertex): list
: Returns an iterable list of in degree vertices for a given vertex

parse_vertex_out(int vertex): list
: Returns an iterable list of out degree vertices for a given vertex

get_edge_cost(int having_start, int having_end): int
: Returns the cost of a defined edge for a given start and end vertex

set_edge_cost(int having_start, int having_end, int cost)
: Sets a given cost to a given edge defined by a start and end vertex

add_vertex()
: Adds a new vertex

remove_edge(int having_start, int having_end)

Removes an edge defined by the given start and end of a vertex

remove_vertex(int vertex)

Removes the given vertex

get_copy(): DirectedGraph

Returns a copy of the **DirectedGraph** as a **DirectedGraph**

Implementation

The implementation uses 3 maps:

```
def __init__(self, size=0):
  self.store_from = {}
  self.store_to = {}
  self.store_cost = {}

  for v in range(size):
     self.store_from[v] = []
     self.store_to[v] = []
```

One map is used to store a list of “out vertices”, one is used to store a list of “in vertices” and one stores the cost of a pair of vertices. In the constructor, the vertices are initialized for the given size.

store_from:

key: vertex

value: list of vertices

store_to:

key: vertex

value: list of vertices

store_cost:

key: pair of 2 vertices

value: cost