Python Graph Library Documentation

Release 1

Manasvi Saxena

Jul 04, 2017

Contents

class `PyGraphLib.`**`graph`**

`add_connection`(*origin_node_key*, *destination_node_key*, *edge_weight*)

Adds connection between the specified nodes

Parameters

- **`origin_node_key`** – origin_node_key: key of origin node
- **`destination_node_key`** – destination_node_key: key of destination node

Returns status of operation

`add_new_node`(*node_key*, *node_data*)

Adds a new node to the dictionary

Parameters

- **`node_key`** – node_key: key of new node
- **`node_data`** – data associated with node

Returns status of the operation

`create_dot_file`(*filename*)

Outputs a file in dot format that can be used for generating images of the graph (see http://www.graphviz.org for more info)

Parameters **`filename`** – name of the file to write the output to

Returns success of the operation

`get_all_data`()

return all data in the graph in the form of a list

`get_all_neighbors`(*node_key*)

return all the neighbor nodes' keys

Parameters **`node_key`** – key of origin node

Returns List of all neighbors' keys and weights

`get_average_edge_cost`()

returns the average weight of all edges

`get_cheapest_edge`()

return the cheapest edge i.e. edge with minimum weight

`get_costliest_edge`()

returns costliest edge i.e. edge with most weight

`get_degree`(*node_key*)

Returns The number degree of specified node

Parameters **`node_key`** – The key of the node for finding the degree

Returns The degree or error code

`get_node_data`(*node_key*)

Gets the data associated with the node

Parameters **`node_key`** – key of the node

Returns data of the node or error message if node not found

get_nodelist_by_degree()
return a list of nodes sorted by degrees

get_path_weight(*origin_node*, *destination_node*)
gets the weight of path associated with specified origin node

Parameters **origin_node** – Key of the origin node

Returns Weight or error code

get_shortest_path(*origin_key*, *destination_key*)
Uses Djikstra's shortest path algorithm to return the shortest path between the specified nodes

Parameters

- **origin_key** – The key of the origin node
- **destinaiton_key** – The key of the destination node

Returns The path in the form of an array or path not found error

get_sorted_data_list()
returns all data in list sorted by key

is_direct_path_present(*origin_node*, *destinaiton_node*)
check whether direct path exists between two nodes

Parameters

- **origin_node** – key of the origin node
- **destination_node** – Key of the destination node

Returns Success or error codes

partitions()
Returns a list of all disconnected sub-graphs in the graph.

Returns a list of sub-graphs in the graph where each

sub-graph is represented as a list of nodes that belong to the sub-graph

remove_connection(*origin_node_key*, *destination_node_key*)
Removes the connection between the specified origin node and the specified destination node Keep in mind that this only removes the connection in one direction, for undirected graphs, the function must be called again with the destination node as the origin node and origin node as the destination node

Parameters

- **origin_node_key** – the key of the origin node
- **destination_node_key** – the key of the destination node

Returns status of the operation

remove_node(*node_key*)
Removes the specified node from the graph. Also removes the connections associated with the node

Parameters **node_key** – the key of the node to be removed

:return : Status message of the operation

CHAPTER 1

Indices and tables

- genindex
- modindex
- search

Index