A Python/Cython software library that performs algorithms on a graph.

Software running platform:

Linux

Installation and Compilation

If you are a user:

Dependency:

Python package-management system: `pip`

```
(base) stephen@pop-os:~$ pip --version
pip 20.0.2 from /home/stephen/miniconda3/lib/python3.7/site-packages/pip (python 3.7)
```

Commands:

`pip install Cython`

This will install a Cython compiler for you 😀

`pip install ./GraphMode/`

This command will do everything for ya!

It will compile necessary Python codes into C codes and set up the software.

you may see something like "ERROR: After October 2020 you may experience..." Please ignore that.

```
(base) stephen@pop-os:~/Documents/gm_demos$ pwd
/home/stephen/Documents/gm_demos
(base) stephen@pop-os:~/Documents/gm_demos$ ls
GraphMode
(base) stephen@pop-os:~/Documents/gm_demos$ python --version
Python 3.7.7
(base) stephen@pop-os:~/Documents/gm_demos$ pip --version
pip 20.0.2 from /home/stephen/miniconda3/lib/python3.7/site-packages/pip (python 3.7)
(base) stephen@pop-os:~/Documents/gm_demos$ pip install Cython
Requirement already satisfied: Cython in /home/stephen/miniconda3/lib/python3.7/site-packages (0.29.21)
(base) stephen@pop-os:~/Documents/gm_demos$ pip install ./GraphMode/
Processing ./GraphMode
Requirement already satisfied: Cython in /home/stephen/miniconda3/lib/python3.7/site-packages (from Graph-Mode==0.0) (0.29.21)
Requirement already satisfied: numpy in /home/stephen/miniconda3/lib/python3.7/site-packages (from Graph-Mode==0.0) (1.19.2)
Requirement already satisfied: pytest in /home/stephen/miniconda3/lib/python3.7/site-packages (from Graph-Mode==0.0) (6.1.2)
Requirement already satisfied: matplotlib in /home/stephen/miniconda3/lib/python3.7/site-packages (from Graph-Mode==0.0) (3.3.3)
Requirement already satisfied: iniconfig in /home/stephen/miniconda3/lib/python3.7/site-packages (from pytest->Graph-Mode==0.0) (1.1.1)
Requirement already satisfied: pluggy<1.0,>=0.12 in /home/stephen/miniconda3/lib/python3.7/site-packages (from pytest->Graph-Mode==0.0) (0.13.1)
Requirement already satisfied: importlib-metadata>=0.12; python_version < "3.8" in /home/stephen/.local/lib/python3.7/site-packages (from pytest->Graph-Mode==0.0) (1.7.0)
Requirement already satisfied: packaging in /home/stephen/.local/lib/python3.7/site-packages (from pytest->Graph-Mode==0.0) (20.4)
Requirement already satisfied: py>=1.8.2 in /home/stephen/miniconda3/lib/python3.7/site-packages (from pytest->Graph-Mode==0.0) (1.9.0)
Requirement already satisfied: toml in /home/stephen/.local/lib/python3.7/site-packages (from pytest->Graph-Mode==0.0) (0.10.1)
Requirement already satisfied: attrs>=17.4.0 in /home/stephen/.local/lib/python3.7/site-packages (from pytest->Graph-Mode==0.0) (19.3.0)
Requirement already satisfied: pillow>=6.2.0 in /home/stephen/miniconda3/lib/python3.7/site-packages (from matplotlib->Graph-Mode==0.0) (8.0.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /home/stephen/miniconda3/lib/python3.7/site-packages (from matplotlib->Graph-Mode==0.0) (1.3.1)
Requirement already satisfied: python-dateutil>=2.1 in /home/stephen/miniconda3/lib/python3.7/site-packages (from matplotlib->Graph-Mode==0.0) (2.8.1)
Requirement already satisfied: cycler>=0.10 in /home/stephen/miniconda3/lib/python3.7/site-packages (from matplotlib->Graph-Mode==0.0) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in /home/stephen/miniconda3/lib/python3.7/site-packages (from matplotlib->Graph-Mode==0.0) (2.4.7)
Requirement already satisfied: zipp>=0.5 in /home/stephen/.local/lib/python3.7/site-packages (from importlib-metadata>=0.12; python_version < "3.8"->pytest->Graph-Mode==0.0) (3.1.0)
Requirement already satisfied: six in /home/stephen/miniconda3/lib/python3.7/site-packages (from packaging->pytest->Graph-Mode==0.0) (1.14.0)
Building wheels for collected packages: Graph-Mode
  Building wheel for Graph-Mode (setup.py) ... done
  Created wheel for Graph-Mode: filename=Graph_Mode-0.0-cp37-cp37m-linux_x86_64.whl size=1446360 sha256=2f0ba592dba17326f57a0461a34f8328b59ee474e267679abc70a3574a81f41d
  Stored in directory: /tmp/pip-ephem-wheel-cache-ia_dlck4/wheels/13/65/20/6ee1180a2d18b4bc35cccbba2c8113223dd9a503f93196e6b3
Successfully built Graph-Mode
Installing collected packages: Graph-Mode
  Attempting uninstall: Graph-Mode
    Found existing installation: Graph-Mode 0.0
    Uninstalling Graph-Mode-0.0:
      Successfully uninstalled Graph-Mode-0.0
Successfully installed Graph-Mode-0.0
```

If you are a developer (If you wanna hack codes or run the tests using pytest):

Dependency:

Anaconda or Miniconda

```
(base) stephen@pop-os:~$ conda --version
conda 4.8.5
```

Steps:

1, Use Anaconda or Miniconda to generate an environment, we call the environment `graph_mode_conda_env` here.

```
(base) stephen@pop-os:~/Documents/gm_demos$ pwd
/home/stephen/Documents/gm_demos
(base) stephen@pop-os:~/Documents/gm_demos$ ls
GraphMode
(base) stephen@pop-os:~/Documents/gm_demos$ cd GraphMode/
(base) stephen@pop-os:~/Documents/gm_demos/GraphMode$ conda create --name graph_mode_conda_env python=3.7
Collecting package metadata (current_repodata.json): done
Solving environment: done
```

2, Activate the conda environment just generated.

```
(base) stephen@pop-os:~/Documents/gm_demos$ pwd
/home/stephen/Documents/gm_demos
(base) stephen@pop-os:~/Documents/gm_demos$ ls
GraphMode
(base) stephen@pop-os:~/Documents/gm_demos$ conda activate graph_mode_conda_env
(graph_mode_conda_env) stephen@pop-os:~/Documents/gm_demos$
```

3, make sure you are now at the root of the project:

```
(graph_mode_conda_env) stephen@pop-os:~/Documents/gm_demos$ pwd
/home/stephen/Documents/gm_demos
(graph_mode_conda_env) stephen@pop-os:~/Documents/gm_demos$ ls
GraphMode
(graph_mode_conda_env) stephen@pop-os:~/Documents/gm_demos$ cd GraphMode/
(graph_mode_conda_env) stephen@pop-os:~/Documents/gm_demos/GraphMode$ pwd
/home/stephen/Documents/gm_demos/GraphMode
(graph_mode_conda_env) stephen@pop-os:~/Documents/gm_demos/GraphMode$
```

4, pip install all the dependencies in terms of `Cython`, `numpy`, `pytest` and `matplotlib`:

```
>>> pip install Cython
>>> pip install numpy
>>> pip install pytest
>>> pip install matplotlib
```

and pip install this project itself to setup everything.

```
>>> pip install --editable .
```

You are good to go!

Test if the installation is successful (for both users and developers):

Basically, if you can do the things below, you are legitimately good to go!

```
(base) stephen@pop-os:~/Documents/gm_demos$ conda activate graph_mode_conda_env
(graph_mode_conda_env) stephen@pop-os:~/Documents/gm_demos$ python
Python 3.7.9 (default, Aug 31 2020, 12:42:55)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from gm.main import show_funcs
>>> show_funcs()
hello_world()
sort_vertices_based_on_comparison
Dijkstra_to_find_shortest_distances(src_vertex, vertices, graph)
dynamic_programming_to_find_the_shortest_path(from_vertex, to_vertex, path, n, dist, foot_print)
compute_influence_of_a_vertex(before_removal, after_removal)
sort_vertices_based_on_comparison(vertices)
>>> from gm.main import hello_world
>>> hello_world()
hello GraphMode!
```

main objects introduction:

Vertex

implemented in: `./GraphMode/src/gm/main_objects/graph.py`
An object having an ID (index) and a weight (impact) to define a vertex in a graph.
usages:

```
>>> from gm.main_objects.graph import Vertex
>>> vertex = Vertex(index=0, impact=10)
```

Edge

implemented in: `./GraphMode/src/gm/main_objects/graph.py`
An object having two vertices as the ends and a cost to define an edge in a graph.
usages:

```
>>> from gm.main_objects.graph import Vertex, Edge
>>> edge = Edge(vertices=(0, 1), cost=3)
```

Graph

implemented in: `./GraphMode/src/gm/main_objects/graph.py`
An object having an array of vertices and an array of edges to define the graph.
usages:

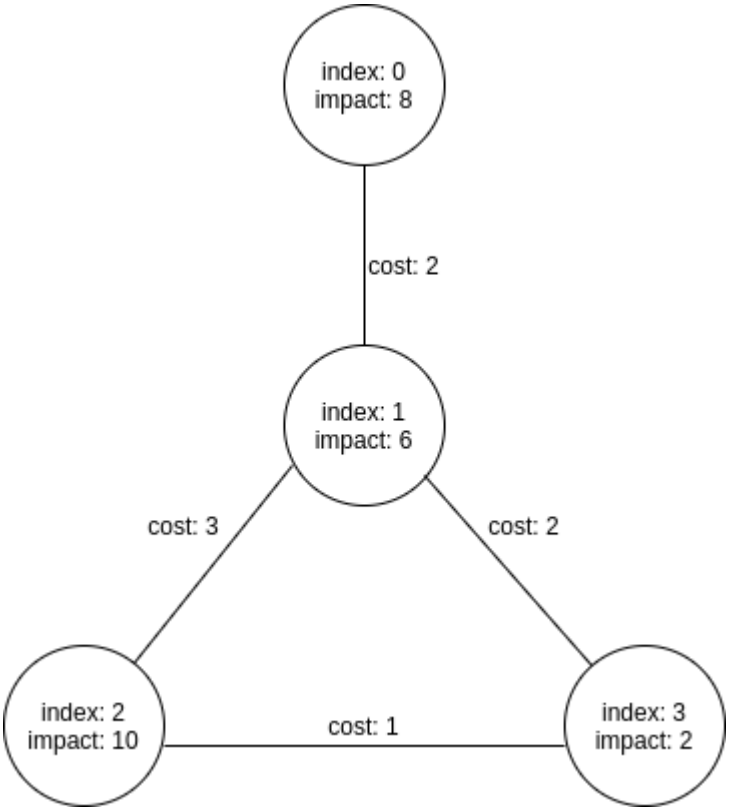
```
>>> from gm.main_objects.graph import Edge, Graph, Vertex
>>> graph = Graph()
>>> vertices = [Vertex(index=0, impact=10), Vertex(index=1, impact=8)]
>>> edges = [Edge(vertices=(0, 1), cost=3)]
>>> graph.vertices = vertices
>>> graph.edges = edges
```

Software DEMO

Option 1: Test or assess algorithms by custom objects

Detailed docstrings about explanations have been added in each implementation file.

let us first construct a graph using the objects mentioned above:



minimum spanning tree + Dijkstra algorithm DEMO:

```
>>> from gm.utils.graph_construction import GraphConstructor
>>> vertices, distance_matrix = graph_constructor.dijkstra_input_vertices_and_distance_matrix()
>>> src_vertex = 0
>>> from gm.main import Dijkstra_to_find_shortest_distances
>>> got = Dijkstra_to_find_shortest_distances(src_vertex, vertices, distance_matrix)
>>> for v in got:
...     print(v)
...
0
2
5
4
```

it worked!

the shortest distance from vertex 0 to vertex 0 is 0;

the shortest distance from vertex 0 to vertex 1 is 2;

the shortest distance from vertex 0 to vertex 2 is 5;

the shortest distance from vertex 0 to vertex 3 is 4;

comparison-based sorting algorithm DEMO:

```
>>> from gm.main_objects.graph import Edge, Graph, Vertex
>>> vertices = [Vertex(0,8), Vertex(1,6), Vertex(2,10), Vertex(3,2)]
>>> import numpy
>>> impacts = numpy.array([v.impact for v in vertices]).astype(numpy.int64)
>>> from gm.main import sort_impacts_based_on_comparison
>>> for impact in impacts:
...     print(impact)
...
8
6
10
2
>>> sort_impacts_based_on_comparison(impacts)
>>> for impact in impacts:
...     print(impact)
...
2
6
8
10
```

it worked!

The impacts array had been successfully sorted.

Floyd-Warshall Dynamic Programming DEMO:

prepare essential input parameters related to the graph mentioned above:

```
(graph_mode_conda_env) stephen@pop-os:~$ conda activate graph_mode_conda_env
(graph_mode_conda_env) stephen@pop-os:~$ python
Python 3.7.9 (default, Aug 31 2020, 12:42:55)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from gm.main_objects.graph import Edge, Graph, Vertex
>>> vertices = [Vertex(0,8), Vertex(1,6), Vertex(2,10), Vertex(3,2)]
>>> edges = [Edge((0,1),2), Edge((1,2),3), Edge((1,3),2), Edge((2,3),1)]
>>> from gm.utils.graph_construction import GraphConstructor
>>> graph = Graph()
>>> graph.vertices = vertices
>>> graph.edges = edges
>>> graph_constructor = GraphConstructor(graph)
>>> dp_input_edges = graph_constructor.dp_input_edges()
>>> from_vertex = 0
>>> to_vertex = 3
>>> import numpy
>>> n = 4 + 1
>>> path = numpy.array([-1 for _ in range(n)]).astype(numpy.int64)
>>> dist = numpy.array([[float('inf')] * n for _ in range(n)]).astype(numpy.double)
>>> foot_print = numpy.array([[-1] * n for _ in range(n)]).astype(numpy.int64)
>>> dp_input_edges += [(v, v, 0.00001) for v in range(n)]
>>> for first, second, weight in dp_input_edges:
...     dist[first][second] = weight
...     foot_print[first][second] = second
...
```

it worked!

```
>>> from gm.main import dynamic_programming_to_find_the_shortest_path
>>> got = [v for v in dynamic_programming_to_find_the_shortest_path(from_vertex, to_vertex, path, n, dist, foot_print) if v != -1]
>>> print(got)
[0, 1, 3]
```

Option 2: If the above was overwhelming, we can also test or assess algorithms using prepared data sets.

data sets can be found in `./GraphMode/tests/data` (5 data sets for each functionality)

You just need:

1, Activate the conda environment (`graph_mode_conda_env`);

2, Import a function `test` from `gm.main`;

3, Invoke func `test` with passing the path to the test txt file you interested. 😃

minimum spanning tree + Dijkstra algorithm DEMO:

```
(base) stephen@pop-os:~/Documents/dev/repos/GraphMode$ conda activate graph_mode_conda_env
(graph_mode_conda_env) stephen@pop-os:~/Documents/dev/repos/GraphMode$ python
Python 3.7.9 (default, Aug 31 2020, 12:42:55)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from gm.main import test
>>> test("/home/stephen/Documents/dev/repos/GraphMode/tests/data/test-f1-1.txt")
you are invoking minimum spanning tree + Dijkstra algorithm
```

comparison-based sorting algorithm DEMO:

```
(base) stephen@pop-os:~/Documents/dev/repos/GraphMode$ conda activate graph_mode_conda_env
(graph_mode_conda_env) stephen@pop-os:~/Documents/dev/repos/GraphMode$ python
Python 3.7.9 (default, Aug 31 2020, 12:42:55)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from gm.main import test
>>> test("/home/stephen/Documents/dev/repos/GraphMode/tests/data/test-f2-1.txt")
Comparison-based algorithm: Insertion Sort
```

Floyd-Warshall Dynamic Programming DEMO:

```
(base) stephen@pop-os:~/Documents/dev/repos/GraphMode$ conda activate graph_mode_conda_env
(graph_mode_conda_env) stephen@pop-os:~/Documents/dev/repos/GraphMode$ python
Python 3.7.9 (default, Aug 31 2020, 12:42:55)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from gm.main import test
>>> test("/home/stephen/Documents/dev/repos/GraphMode/tests/data/test-f3-1.txt")
Floyd-Warshall Dynamic Programming
```

Pytest

All the logic, instructions and explanations have been added in detailed docstrings in `./GraphMode/tests`

Feel free to inspect it to know what's going on 😃

Testing

1, first you need to install the project as a developer (instructions added above in section `Installation`);

2, activate the conda environment:

`conda activate graph_mode_conda_env` 3, head into the project:

`>>> cd ./GraphMode/`

4, execute Pytest:

`>>> pytest tests`

DEMO:

```
(base) stephen@pop-os:~/Documents/dev/repos$ conda activate graph_mode_conda_env
(graph_mode_conda_env) stephen@pop-os:~/Documents/dev/repos$ cd ./GraphMode/
(graph_mode_conda_env) stephen@pop-os:~/Documents/dev/repos/GraphMode$ pytest tests
============================================================ test session starts ============================================================
platform linux -- Python 3.7.9, pytest-6.1.2, py-1.9.0, pluggy-0.13.1
rootdir: /home/stephen/Documents/dev/repos/GraphMode
collected 15 items

tests/Dijkstra_tests/test_Dijkstra.py ..
```