Lab 11: Graph Data Structures

Implement the Graph ADT with two data structures:

- `Graph_ES` - edge set
- `Graph_AS` - adjacency set

Note: you can find partial or full solutions for this lab in the textbook and video lectures for this module - the problems were chosen because they are good learning exercises, and they double as good teaching exercises. Avoid external resources *during* lab; do your best to figure things out with you and your partner. Feel free to access these resources afterwards if you do not finish during.

Graph

Each graph you implement should support the following ADT

Magic Metohds

- `init(V, E)`
  - initializes with optional sets of vertices `V` and edges `E`
- `len`
  - returns the number of vertices in the graph
- `iter`
  - iterates over all vertices in graph

Non-magic Methods

- `add_vertex(v)`
  - adds vertex `v` to graph
- `remove_vertex(v)`
  - removes vertex `v` from graph
  - raise a `KeyError` if `v` is not in graph
- `add_edge(e)`
  - adds edge `e` to graph (assume `e` is a 2-tuple)
- `remove_edge(e)`
  - removes edge `e` from graph (assume `e` is a 2-tuple)
  - raise a `KeyError` if `e` is not in graph
- `_neighbors(v)`
  - returns an iterable collection of neighbors of vertex `v`
  - Running time:
    - *O*(m), `Graph_ES` (`m` is the number of edges in the graph)
    - *O*(1), `Graph_AS` (time to return an iterator does **not** depend on the number of neighbors)

Examples

Any examples below are intended to be illustrative, not exhaustive. Your code may have bugs even if it behaves as below. Write your own tests, and think carefully about edge cases.

```
>>> from lab11 import *
>>> # 1 <==> 2 <==> 3
>>> vs = {1,2,3}
>>> es = {(1,2), (2,1), (2,3), (3,2)}
>>> g = Graph_ES(vs, es)
>>> assert len(g) == 3
>>> verts = set()
>>> for v in vs:
...     verts.add(v)
...
>>> assert verts == vs
>>> nbrs = {2:set()}
>>> for n in g._neighbors(2):
...     nbrs[2].add(n)
...
>>> assert nbrs == {2:{3, 1}}
>>>
```

Submitting

At a minimum, submit the following files:

- `lab11.py`
  - contains classes
    - `Graph_ES` - edge set implementation
    - `Graph_AS` - adjacency set implementation

Students must submit to Mimir **individually** by the due date (typically, two days after lab at 11:59 pm EST) to receive credit.

Grading

- 50 - `Graph_ES`
  - 5 - len
  - 5 - init
  - 5 - iter
  - 15 - add/remove vertices
  - 15 - add/remove edges
  - 5 - _neighbors
- 50- `Graph_AS`
  - 5 - len
  - 5 - init
  - 5 - iter
  - 15 - add/remove vertices

    - 15 - add/remove edges
    - 5 - _neighbors

Feedback

If you have any feedback on this assignment, please leave it here.

We check this feedback regularly. It has resulted in:

- A simplified, clear **Submitting** section on all assignments
- A simplified, clear **Grading** section on all assignments
- Clearer instructions on several assignments