Założenia ogólne

1. Funkcjonalność systemu ma być realizowana przez szereg klas, z których głównymi klasami są Graph (graf), Vertex (wierzchołek), Edge (krawędź) oraz Path (ścieżka). Pozostałe klasy należy utworzyć zgodnie z zapotrzebowaniem.
2. Graf skierowany jest parą zbiorów – krawędzi i wierzchołków. Zbiór wierzchołków przechowuje informacje o wierzchołkach w grafie, a zbiór krawędzi przechowuje informacje o koszcie ruchu pomiędzy parą wierzchołków.
3. Format nazewnictwa wierzchołków jest dowolny. Mogą być opisane nazwą, liczbą porządkową, współrzędnymi, etykietą i dowolnym innym typem danych.
4. W związku z tym, że graf jest skierowany, koszt ruchu od wierzchołka v1 do v2 może się różnić od kosztu ruchu od wierzchołka v2 do v1.
5. Użytkownik powinien mieć możliwość przypisania wartości zarówno do krawędzi, jak i do wierzchołka. Wartości mogą być dowolnego typu.
6. Krawędzie mogą być przechowywane w formie listy lub w formie macierzy incydencji. Wybór implementacji pozostaje w gestii realizatora.
7. Należy zaimplementować algorytm A*, algorytm Dijkstry lub algorytm Bellmana-Forda. Wystarczy zaimplementować jeden z tych algorytmów.
8. Należy zaimplementować funkcję statyczną, która pozwala skonwertować obiekt klasy Graph na obiekt analogicznej klasy z pakietów SciPy CSGraph lub NetworkX.
9. Należy zaimplementować funkcję statyczną, która pozwala skonwertować obiekt grafu z pakietów SciPy CSGraph lub NetworkX na obiekt typu Graph.

Klasa Vertex

Klasa powinna posiadać następujące pola i metody:

- Pole „graph”, które przechowuje referencję do grafu macierzystego
- Pole „neighbours”, które pozwala pobrać listę wierzchołków, z którymi wierzchołek sąsiaduje w grafie
- Pole „name”, które przechowuje nazwę wierzchołka
- Pole „value”, które przechowuje opcjonalną wartość przypisaną do wierzchołka
- Przeładowany operator „[]” (metody __getitem__, __setitem__ i __delitem__, https://docs.python.org/3/reference/datamodel.html#emulating-container-types), który pozwala dodać lub pobrać wierzchołek sąsiadujący
- Metodę „isConnected(v2)”, która zwraca informacje o tym, że aktualny wierzchołek i wierzchołek v2 sąsiadują ze sobą

Klasa Edge

Klasa powinna posiadać następujące pola i metody:

- Pole „graph”, które przechowuje referencję do grafu macierzystego
- Pole „predecessor”, które pozwala pobrać wierzchołek będący poprzednikiem krawędzi
- Pole „successor”, które pozwala pobrać wierzchołek będący następnikiem krawędzi
- Pole „value”, które przechowuje wartość krawędzi
- Metodę reverse, która zwraca krawędź w odwrotnym kierunku lub None, jeżeli takiej krawędzi nie ma w grafie
- Przeładowany operator „*”, który pozwala przemnożyć wartość krawędzi przez skalar

Klasa Route

Klasa powinna posiadać następujące pola i metody:

- Pole „graph”, które przechowuje referencję do grafu macierzystego
- Przeładowany operator „[]”, który pozwala iterować po krawędziach, które tworzą ścieżkę
- Pole „value”, które przechowuje sumaryczną wartość ścieżki
- Przeładowany operator „*”, który pozwala przemnożyć wartość krawędzi przez skalar

Klasa Graph

Klasa powinna posiadać następujące pola i metody:

- Przeładowany operator „[]”, który pozwala dodać lub pobrać wierzchołek o określonym indeksie
- Przeładowany operator „+”, pozwalający dodać do aktualnego grafu inny graf, wierzchołek lub krawędź
- Przeładowany operator „*”, który pozwala przemnożyć występujące wagi krawędzi przez skalar
- Pole „vertices”, które pozwala pobrać listę wierzchołków (obiektów klasy Vertex)
- Pole „edges”, które pozwala pobrać listę krawędzi (obiektów klasy Edge)
- Metoda „incidence”, która zwraca macierz incydencji. Metoda przyjmuje jeden argument typu bool, nazwany „weighted”. Jeżeli „weighted” jest równe „false” (wartość domyślna), zwracana jest macierz nieważona (taka, która zawiera wartości 0 i 1). Jeżeli „weighted” jest równe „true”, zwracana jest macierz ważona, gdzie waga odpowiada kosztowi krawędzi pomiędzy wierzchołkami
- Metodę __init__, której argumentem jest macierz incydencji
- Metodę „findPath”, która zwraca wynik wybranego algorytmu wyszukiwania ścieżek dla dwóch podanych jako argumenty wierzchołków. Metoda powinna zwracać obiekt klasy Path.

- Metodę „isGrid”, która zwraca informację czy graf jest siatką (https://en.wikipedia.org/wiki/Lattice_graph)

Elementy opcjonalne

1. Zaimplementować obie metody przechowywania krawędzi (jako listę i jako macierz incydencji) oraz wybierać metodę przechowywania w zależności od liczby krawędzi. Jako listę, jeżeli krawędzi jest mało lub jako macierz incydencji, jeżeli krawędzi jest wiele.
2. Dodatkowe metody klasy Graph, które pozwalają zbadać jego właściwości: czy jest pełny, czy jest skierowany, czy jest ważony, czy jest zawiera cykle, i tak dalej
3. Zaimplementować dodatkowe, specjalistyczne algorytmy wyszukiwania ścieżek (np. D* Lite lub JPS)
4. Inne, zgodnie z własną inwencją