Прикладні алгоритми. Завдання 2, звіт

Михайло Голуб

19 вересня 2024 р.

Реалізація класу граф:

Створено клас *Graph* що при ініціалізації приймає на вхід матрицю ребер. Цей клас має наступні методи:

to_ll – повертає масив списків зв'язності;

clear – встановлює усі клітинки матриці в False;

from_ll – створює нову матрицю зв'язності з масиву списків зв'язності;

add_vertice – додає одну колонку та один рядок в матрицю суміжності;

add_edge – встановлює відповідну пару клітинок в True;

remove_vertice – видаляє колонку та рядок з вказаним індексом;

remove_edge – встановлює відповідну пару клітинок в False.

Реалізація інших класів:

Клас *OrientedGraph* є нащадком (не дитиною, від англ. inherit – наслідувати / успадковувати) *Graph* з перевизначенням додання та видалення ребер: значення встановлюється не для пари клітинок, а для однієї клітинки.

Клас *WeightedGraph* є нащадком *Graph* з перевизначенням методів: методи працюють не з булевими значеннями, а з числовими.

Клас *OrientedWeightedGraph* є нащадком *WeightedGraph* з перевизначенням додання та видалення ребер: значення встановлюється не для пари клітинок, а для однієї клітинки.

Реалізація рандомізованих класів:

Класи *RandomGraph, RandomOrientedGraph, RandomWeightedGraph, RandomOrienedWeightedGraph* є нащадками відповідних класів з перевизначенням *__init__*: створюється пуста (заповнена False) матриця суміжності потрібного розміру, і через неї проходить цикл що з вказаною ймовірністю записує True або випадкове число. Для того щоб в не орієнтованих графах кожне ребро було пройдено циклом лише один раз, ітератор другої координати працює від i до n, де i – позиція ітератора першої координати та n – кількість вершин. В орієнтованих графах обидва ітератори працюють від 0 до n