Прикладні алгоритми. Завдання 2, звіт

Михайло Голуб

5 жовтня 2024 р.

Реалізація UnionFind:

Для зберігання інформації про належність елементу до множини створено клас Node з значеннями value, next та header.

Для зберігання інформації про множину створено клас UnionFindSetMeta з значеннями head, tail, root та size, а також методами recalc_size та recalc_tail.

Для реалізації структури UnionFind створено клас UnionFindHandler з наступною логікою роботи:

- існує словник універсуму, в якому знаходяться пари елемент-Node, елемент виступає ключем словника;
- метод make_set додає нову пару з вказаним елементом, при цьому в класі Node міститься вказівник на відповідний UnionFindSetMeta;
- метод find повертає корінь множини якій належить шуканий елемент, або -1 якщо його немає серед ключів словника;
- метод union об'єднує дві множини до яких належать вказані елементи;
- метод everything_in_one_set повертає True якщо всі елементи містяться в одній множині, інакше – False.

Реалізація алгоритму Крускала:

З другого завдання імпортовано класи графів та класів що генерують випадкові графи. Алгоритм Крускала реалізовано наступним чином:

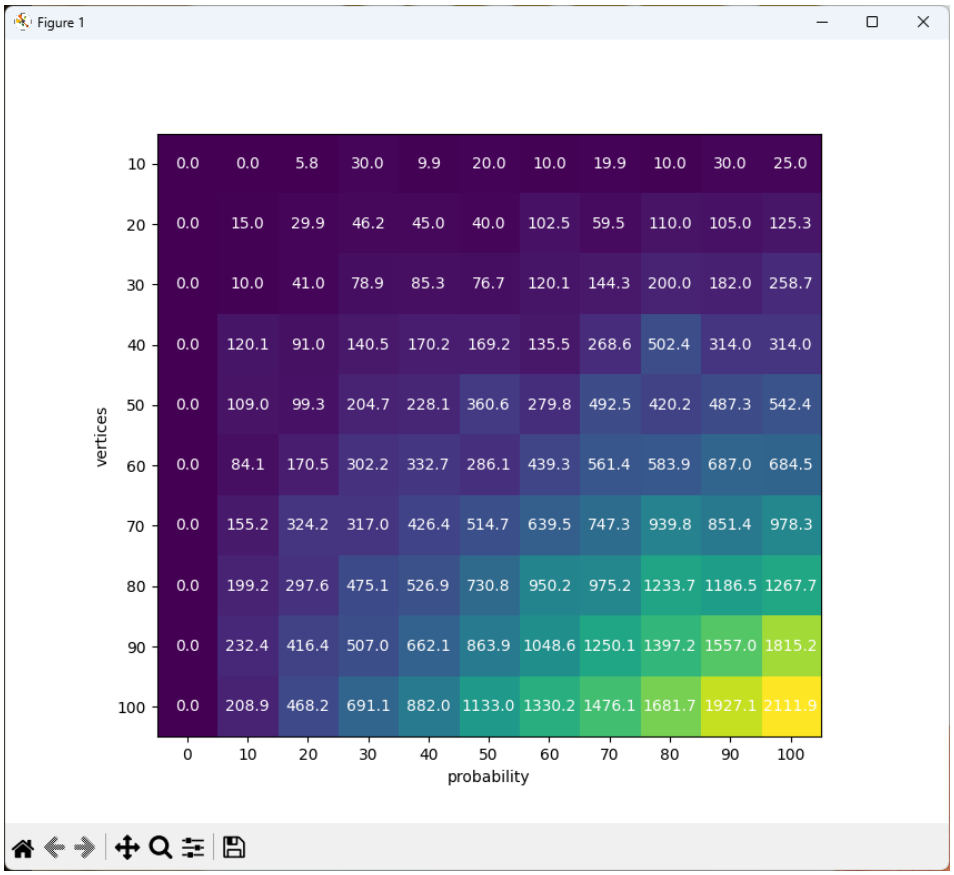
1. Алгоритм приймає на вхід зважений неорієнтований граф та видаляє з нього усі петлі;
2. Створюється масив усіх ребер: масив трійок значень (вага, вершина_1, вершина_2). При цьому кожна пара вершин опрацьовується лише один раз;
3. Масив усіх ребер сортується від найменшої ваги до найбільшої
4. Створюється зважений неорієнтований граф minimal_tree з кількістю вершин рівною вхідному графу, але без ребер;
5. Створюється пуста структура UnionFind
6. Обрати неопрацьоване ребро з найменшою вагою;
7. Якщо обидві вершини відсутні в UnionFind – додати їх туди та об'єднати їх в одну множину; запам'ятати ребро яке їх з'єднує;
8. Якщо одна з множин відсутня в UnionFind – додати її туди та об'єднати з множиною якій належить інша вершина; запам'ятати ребро яке їх з'єднує;

9. Повторювати 6-8 доки не опрацьовані усі ребра, або "кількість вершин в універсумі рівна кількості вершин в графі та усі елементи UnionFind належать одній множині";

10. Записати в minimal_tree усі запам'ятовані ребра та повернути його.

Аналіз часу роботи алгоритму Крускала:

В найгіршому випадку алгоритм Крускала має складність O(E*logV), де E це кількість ребер і V це кількість вершин. При подвоєнні вірогідності створення ребра очікуване подвоєння часу роботи. При збільшенні кількості вершин очікується спадання швидкості росту часу роботи. На тепловій карті нижче зображено залежність часу роботи алгоритму Крускала від кількості вершин у графі та ймовірності створення ребра.



З теплової карти видно, що колонки 10 та 20, 20 та 40, 30 та 60, 40 та 80 50 та 100 відрізняються приблизно в два рази, тож лінійну залежність від

кількості ребер підтверджено.

Для оцінки залежності часу роботи від кількості вершин потрібно створити генератор випадкових графів з фіксованою кількістю ребер при змінній кількості вершин. Створено клас PreciseRandomWeightedGraph який робить саме це. Нижче наведено графік залежності часу роботи алгоритму від кількості вершин при фіксованій кількості ребер $\frac{500*499}{2} = 124750$.