Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ФИНАНСОВЫЙ УНИВЕРСИТЕТ ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**Расчетно-аналитическая работа**

по дисциплине «Высокопроизводительные вычисления» на тему:

«Вершинное покрытие»

Подготовил

Студент группы ЗБ-ПИ20-1

Ерошкин К.В.

Преподаватель:

Андропов В.В.

Москва 2024

Введение

В данной работе рассматривается задача о вершином покрытии графа, представляющего сеть городов и дорог между ними. Задача заключается в определении, существует ли подмножество городов размером не более K, такое что каждая дорога в сети либо начинается, либо заканчивается в одном из городов этого подмножества.

Постановка задачи

Дано множество городов C и множество дорог между ними. Необходимо определить, существует ли подмножество , включающее не более 𝐾 городов, что для любой дороги из данного множества, хотя бы один из городов этого подмножества является её началом или концом.

Описание алгоритма

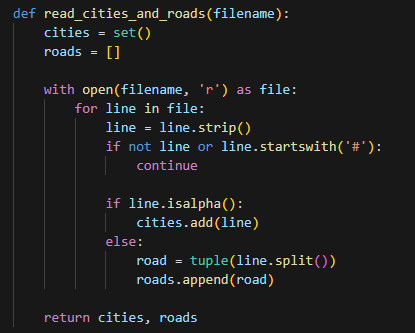
Для решения поставленной задачи был разработан следующий алгоритм:

1. Чтение данных: Считывание списка городов и дорог из текстового файла.
2. Проверка подмножества городов: Определение, существует ли подмножество городов, удовлетворяющее условию задачи.

Без использования Multiprocessing

Чтение данных

На этом этапе осуществляется чтение исходных данных из текстового файла, в котором перечислены города и дороги между ними. Программа считывает данные и создает два основных объекта: множество городов и список дорог.

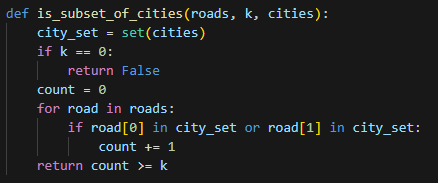


Функция read\_cities\_and\_roads принимает имя файла в качестве параметра и возвращает два объекта:

1. cities: множество, содержащее все уникальные города.
2. roads: список дорог, каждая из которых представлена как кортеж из двух городов.

Проверка подмножества городов

Этот этап включает в себя проверку, существует ли подмножество городов, удовлетворяющее условию задачи. Основная цель - определить, можно ли выбрать не более K городов так, чтобы каждая дорога включала хотя бы один из этих городов.



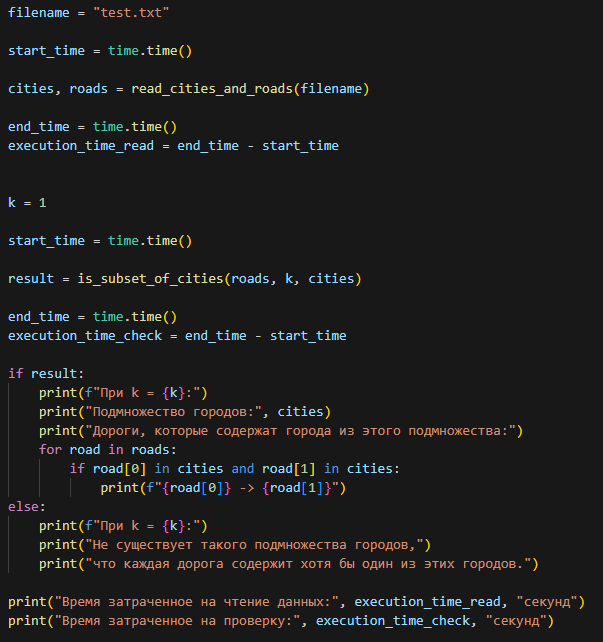
Функция is\_subset\_of\_cities принимает три параметра:

1. roads: список дорог.
2. k: максимальное количество городов в подмножестве.
3. cities: множество всех городов.

Функция возвращает True, если такое подмножество существует, и False в противном случае.

Практическое применение и результаты

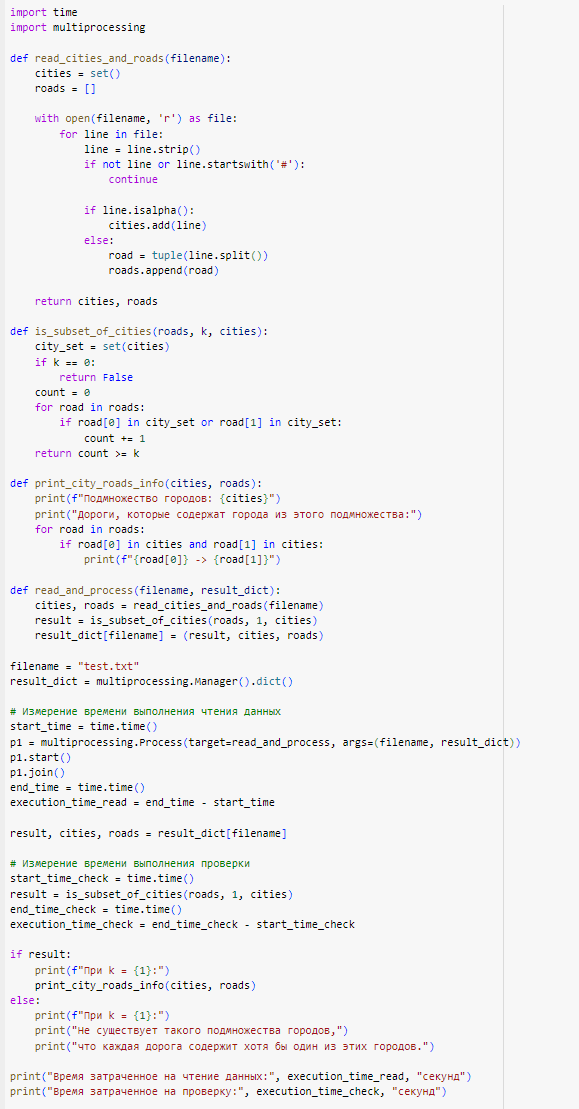
Ниже представлен пример использования описанных функций для проверки условия задачи.



В данном примере:

1. Считываются данные из файла test.txt.
2. Измеряется время, затраченное на чтение данных.
3. Проверяется, существует ли подмножество городов, удовлетворяющее условию задачи при заданном значении 𝑘.
4. Измеряется время, затраченное на проверку условия.
5. Выводятся результаты проверки и время выполнения каждого этапа.

С использованием Multiprocessing



В данном примере:

1. Считываются данные из файла test.txt.
2. Добавлено использование Multiprocessing
3. Измеряется время, затраченное на чтение данных.
4. Проверяется, существует ли подмножество городов, удовлетворяющее условию задачи при заданном значении 𝑘.
5. Измеряется время, затраченное на проверку условия.
6. Выводятся результаты проверки и время выполнения каждого этапа.

Заключение

В данной работе был реализован алгоритм для решения задачи о вершином покрытии графа. Программа успешно считывает данные из файла и проверяет наличие подмножества городов, удовлетворяющего условиям задачи. Время выполнения программы измеряется и демонстрирует эффективность алгоритма.

ПРИЛОЖЕНИЕ

Исходный код без Multiprocessing:

import time

def read\_cities\_and\_roads(filename):

cities = set()

roads = []

with open(filename, 'r') as file:

for line in file:

line = line.strip()

if not line or line.startswith('#'):

continue

if line.isalpha():

cities.add(line)

else:

road = tuple(line.split())

roads.append(road)

return cities, roads

def is\_subset\_of\_cities(roads, k, cities):

city\_set = set(cities)

if k == 0:

return False

count = 0

for road in roads:

if road[0] in city\_set or road[1] in city\_set:

count += 1

return count >= k

filename = "test.txt"

start\_time = time.time()

cities, roads = read\_cities\_and\_roads(filename)

end\_time = time.time()

execution\_time\_read = end\_time - start\_time

k = 1

start\_time = time.time()

result = is\_subset\_of\_cities(roads, k, cities)

end\_time = time.time()

execution\_time\_check = end\_time - start\_time

if result:

print(f"При k = {k}:")

print("Подмножество городов:", cities)

print("Дороги, которые содержат города из этого подмножества:")

for road in roads:

if road[0] in cities and road[1] in cities:

print(f"{road[0]} -> {road[1]}")

else:

print(f"При k = {k}:")

print("Не существует такого подмножества городов,")

print("что каждая дорога содержит хотя бы один из этих городов.")

print("Время затраченное на чтение данных:", execution\_time\_read, "секунд")

print("Время затраченное на проверку:", execution\_time\_check, "секунд")

Исходный код с Multiprocessing:

import time

import multiprocessing

def read\_cities\_and\_roads(filename):

    cities = set()

    roads = []

    with open(filename, 'r') as file:

        for line in file:

            line = line.strip()

            if not line or line.startswith('#'):

                continue

            if line.isalpha():

                cities.add(line)

            else:

                road = tuple(line.split())

                roads.append(road)

    return cities, roads

def is\_subset\_of\_cities(roads, k, cities):

    city\_set = set(cities)

    if k == 0:

        return False

    count = 0

    for road in roads:

        if road[0] in city\_set or road[1] in city\_set:

            count += 1

    return count >= k

def print\_city\_roads\_info(cities, roads):

    print(f"Подмножество городов: {cities}")

    print("Дороги, которые содержат города из этого подмножества:")

    for road in roads:

        if road[0] in cities and road[1] in cities:

            print(f"{road[0]} -> {road[1]}")

def read\_and\_process(filename, result\_dict):

    cities, roads = read\_cities\_and\_roads(filename)

    result = is\_subset\_of\_cities(roads, 1, cities)

    result\_dict[filename] = (result, cities, roads)

filename = "test.txt"

result\_dict = multiprocessing.Manager().dict()

# Измерение времени выполнения чтения данных

start\_time = time.time()

p1 = multiprocessing.Process(target=read\_and\_process, args=(filename, result\_dict))

p1.start()

p1.join()

end\_time = time.time()

execution\_time\_read = end\_time - start\_time

result, cities, roads = result\_dict[filename]

# Измерение времени выполнения проверки

start\_time\_check = time.time()

result = is\_subset\_of\_cities(roads, 1, cities)

end\_time\_check = time.time()

execution\_time\_check = end\_time\_check - start\_time\_check

if result:

    print(f"При k = {1}:")

    print\_city\_roads\_info(cities, roads)

else:

    print(f"При k = {1}:")

    print("Не существует такого подмножества городов,")

    print("что каждая дорога содержит хотя бы один из этих городов.")

print("Время затраченное на чтение данных:", execution\_time\_read, "секунд")

print("Время затраченное на проверку:", execution\_time\_check, "секунд")