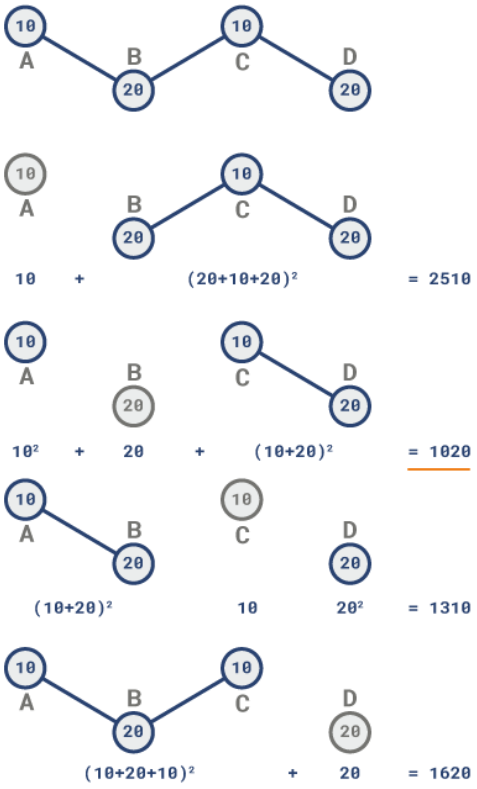
Задача сетевой связанности на графах

1 Постановка задачи

Представим, что в городе N есть мобильная сеть. Строительство нового района всегда начинается с возведения вышки мобильной связи. Все пользователи района подключаются к этой вышке, чтобы иметь возможность пользоваться услугами сети (отправлять\получать сообщения). Таким образом, независимо от того, насколько велик город, все пользователи могут отправлять сообщения друг другу пока все вышки подключены между собой (между ними есть сетевая связанность).

Одна из главных цель мэра города N – контролировать мобильную связанность граждан города. Показатель мобильной связанности города – это сумма мобильной связанности всех граждан города. В свою очередь, показатель мобильной связанности каждого гражданина равен количеству граждан (всегда включая себя), которым можно отправлять сообщения.

Поскольку город N растет, граждане решили, что мэру нужен помощник, чтобы тот мог сосредоточиться на защите вышек. Основная задача помощника – выяснить, какие вышки следует защищать первыми. Основная цель выбрать самую важную вышку в сети. Если несколько вышек имеют максимальную важность, надо обозначить их все. Важность вышки определяется ее влиянием на показатель мобильной связанности города – чем меньше показатель после уничтожения вышки, тем она важнее.



2 Формат входных данных

На вход два параметра: структура сети (как список соединений между вышками), пользователи каждой вышки (в виде dict, где ключи – это имя узла, а значения - количество пользователей).

Набор входных данных задается в файле. Формат файла предполагает задание массива из 2 объектов (по объекту для каждого входного параметра):

1. Объект, описывающий структуру сети в виде массива массивов (каждый из которых состоит из пары строк). В строке указано имя вершины.
2. Объект, описывающий пользователей на каждой вышке, представляется в виде dict, где ключи имя вышки, а значения - количество пользователей.

Далее приведены примеры входных данных в виде входного файла и соответствующие выходные результаты:

```
Вход1:
{
   [
        ['A', 'B'],
        ['B', 'C']
   ],{
        'A': 10,
        'B': 10,
        'C': 10
   }
}
Выход1: ['B']
```

```
Вход2:
{
   [
        ['A', 'B']
   ],{
        'A': 20,
        'B': 10
   }
}

Выход2: ['A']
```

3 Формат выходных данных

В стандартный вывод необходимо вывести один параметр: список строк – список наиболее важных вышек.

4 Задания и система оценки

Основное задание (обязательное, оценивается в 5 баллов):

1) реализовать базовый алгоритм решения задачи в виде программы на языке C++ или Python;
2) написать инструкцию к программе:
   - как собирать и запускать программу;

- описание формата входного файла.

Дополнительное задание 1 (оценивается в 2 балла):

1) реализовать решение задачи в виде клиент-серверного приложения с использованием docker:
   - клиент и сервер выполняются в отдельных docker-контейнерах;
   - клиент и сервер взаимодействуют посредством сетевых socket-ов.
   - у клиента имеется несколько комплектов входных данных;
   - клиент передает серверу комплект входных данных и ожидает ответа от сервера;
   - сервер выполняет решение задачи, передает результат клиенту и ожидает следующего комплекта входных данных;
   - после получения ответа, клиент передает серверу очередной комплект входных данных.
2) выложить docker-образы на Docker Hub (рекомендуется создать новую учетную запись и не афишировать ее);
3) написать инструкцию, как продемонстрировать решение на сайте https://labs.play-with-docker.com/

Дополнительное задание 2 (оценивается в 2 балла):

1) оформить сервер в виде web-server в docker контейнере, выложить docker образ на docker hub, составить yaml-файл для развёртывания решения при помощи docker compose: https://docs.docker.com/compose/.
   - клиент и сервер выполняются в отдельных docker-контейнерах;
   - клиент и сервер взаимодействуют посредством HTTP REST API.
   - у клиента имеется несколько комплектов входных данных;
   - клиент отправляет серверу REST-запросы с входными данными и ожидает ответа от сервера;
   - сервер выполняет решение задачи, передает результат клиенту в виде ответа на REST-запрос и ожидает следующего REST-запроса с входными данными;
2) выложить docker образы на Docker Hub (рекомендуется создать новую учетную запись и не афишировать ее);
3) написать инструкцию, как продемонстрировать решение на сайте https://labs.play-with-docker.com/

Баллы за основное и дополнительные задания суммируются. Если при проверке задания выявлен плагиат, оценка за все части задания обнуляется; проверяющие *не проводят* анализ, чье задание является «первоисточником», а чье – «заимствованием».

5 Требования к программной реализации

1) программа должна быть реализована на языке C++ или Python;
2) инструкции должны быть представлены в формате .txt или .pdf;
3) программа должна работать в среде ОС Linux (проверка будет проводиться в ОС Ubuntu 18.04 или Debian Stretch);
4) текст программы должен быть хорошо читаемым, структурированным, и содержать комментарии в количестве, необходимом для его понимания;
5) программа должна диагностировать ситуации некорректного пользовательского ввода и содержимого входных файлов;
6) исходный код программы не должен выкладываться в публичные репозитории (github и т.п.)

6 Отправка работ

1) результаты выполнения заданий отправляются на адрес: asvk-nabor@asvk.cs.msu.ru
2) тема письма должна иметь вид: [номер группы] – MGT – Фамилия Имя,
   например: [201] – MGT – Самоваров Иван
3) материалы основного и дополнительных (при наличии) заданий должны быть оформлены в виде отдельных архивов формата .tar.bz2