Отчёт по решению задачи поиска пар IP-адресов

Весь код и всё выполненное можно найти в репозитории:

https://github.com/Ultimatereo/YaProfiPI

1. Описание решения и выбранных признаков

Общее описание:
Данное решение реализовано с использованием объектно-ориентированного и модульного подхода. Основная идея состоит в следующем:

- **Получение IP-адресов.** Список IP-адресов формируется посредством разрешения доменных имён популярных веб-сайтов (например, google.com, youtube.com, facebook.com и т.д.) через DNS.
- **Проверка доступности.** Для каждого полученного IP-адреса выполняется проверка его доступности с помощью команды *ping*. Если адрес отвечает на запрос, он считается доступным.
- **Извлечение свойства.** Для каждого доступного IP-адреса вычисляется заданное целочисленное свойство. В реализации предложено два варианта:
  - *SumOctetsExtractor*: вычисляется сумма всех октетов IP-адреса.
  - *CountOnesExtractor*: определяется количество единиц в 32-битном представлении IP-адреса.
- **Поиск пар IP.** Для каждого доступного IP производится серия попыток сгенерировать кандидатный IP, обладающий таким же значением свойства. Если кандидатный IP также оказывается доступным, формируется пара адресов.
- **Сохранение результатов.** Результаты (найденные пары IP) записываются в CSV-файл в следующем формате:
  - IPv4-адрес 1, Критерий доступности 1, IPv4-адрес 2, Критерий доступности 2, Признак совпадения (например, «Свойство=XXX»).

Выбранные признаки:
Для демонстрации реализованы два подхода к извлечению свойства:

- Сумма октетов IP-адреса.
- Количество единиц в двоичном представлении IP-адреса.

---

2. Методы поиска адресов

Описание метода:
Список исходных IP-адресов получается с помощью класса, реализующего

интерфейс IIPGenerator. Конкретная реализация, класс IPGenerator, разрешает доменные имена из заранее составленного списка популярных сайтов с использованием DNS. Если домен разрешается успешно, полученный IP-адрес добавляется в список.

Основные моменты реализации:

- Использование библиотеки socket для вызова функции gethostbyname.
- Обработка исключений для случаев, когда разрешение домена не удаётся.
- Логгирование каждого успешно разрешённого домена с указанием соответствующего IP.

---

3. Методы проверки доступности адресов

Описание метода:
Проверка доступности осуществляется с помощью класса, реализующего интерфейс IIPAvailabilityChecker – класса IPAvailabilityChecker. Основной способ проверки – запуск системной команды *ping* с передачей IP-адреса. Если команда возвращает код 0, адрес считается доступным.

Основные моменты реализации:

- Использование модуля subprocess для выполнения команды ping.
- Определение параметра для Windows (-n) или Unix-систем (-c).
- Логгирование результатов проверки для каждого IP.

---

4. Детальное описание алгоритма

Архитектура решения:
Решение построено на основе набора интерфейсов и их конкретных реализаций, что обеспечивает модульность и возможность легко менять критерии поиска. Основные интерфейсы:

- **IIPAvailabilityChecker:** проверка доступности IP.
- **IIPGenerator:** получение списка IP-адресов.
- **IIPPropertyExtractor:** извлечение заданного свойства из IP и генерация кандидатного IP с этим свойством.
- **IIPPairFinder:** поиск пар IP с совпадающим свойством.
- **ICSVWriter:** запись результатов в CSV-файл.

Этапы работы алгоритма:

1. **Генерация списка IP:**
   Класс IPGenerator осуществляет DNS-разрешение для списка популярных доменов и формирует список IP-адресов.
2. **Проверка доступности:**
   Каждый IP-адрес из списка проверяется на доступность с использованием метода is_available класса IPAvailabilityChecker (проверка осуществляется через ping).
3. **Извлечение и генерация свойства:**
   Для каждого доступного IP-адреса с помощью объекта, реализующего интерфейс IIPPropertyExtractor (например, SumOctetsExtractor или CountOnesExtractor), извлекается заданное целочисленное свойство. Затем выполняется серия попыток сгенерировать кандидатный IP, обладающий тем же свойством, с помощью метода generate_candidate.
4. **Поиск пар IP:**
   Если кандидатный IP также проходит проверку доступности, формируется пара адресов. Для каждого исходного IP выполняется заданное число попыток (например, 10) для поиска подходящего кандидата.
5. **Запись результатов:**
   Найденные пары сохраняются в CSV-файл. Формат строки:

   IPv4-адрес 1, Критерий доступности 1, IPv4-адрес 2, Критерий доступности 2, Признак совпадения

Преимущества архитектурного подхода:

- Гибкость – можно легко изменить критерий поиска пар, просто подставив другую реализацию IIPPropertyExtractor.
- Модульность – каждый компонент разрабатывается и тестируется отдельно.
- Лёгкость расширения – добавление новых методов проверки или генерации IP выполняется путём создания новой реализации интерфейса.

---

5. Описание результатов

Формат результатов:
После выполнения приложения формируются два CSV-файла с результатами:

- **pair_results_sum.csv** – содержит пары IP-адресов, найденных с использованием свойства суммы октетов.
- **pair_results_ones.csv** – содержит пары IP-адресов, найденных с использованием свойства количества единиц в двоичном представлении IP.

Каждая строка CSV-файла имеет следующий формат:

IPv4-адрес 1, Критерий доступности 1, IPv4-адрес 2, Критерий доступности 2, Свойство=XXX

Заключение

В данной работе реализовано приложение, позволяющее находить пары IP-адресов с совпадающими свойствами. Используемые методы включают DNS-разрешение для получения IP-адресов, проверку доступности через ping, а также модульное извлечение свойств и генерацию кандидатных адресов. Применённый подход с использованием абстрактных интерфейсов обеспечивает гибкость и расширяемость системы, позволяя легко изменять критерии поиска без изменения общей архитектуры приложения.

Результаты работы системы сохраняются в виде CSV-файлов, где каждая строка соответствует найденной паре IP-адресов с указанием признака совпадения.
Подробную информацию о работе системы можно найти в логах, выводимых модулем logging.
