План тестирования

1. Введение

1.1. Основная информация

Документ описывает методы и подходы к тестированию, которые будут использоваться тестировщиками команды, для тестирования проекта. Объект тестирования — это деятельность, направленная на проверку работоспособности функций проекта.

1.2. Цель

Тест-план проекта преследует следующие цели:

- Определить существующую информацию о проекте и программных компонентах подлежащих тестированию.
- Описать стратегии тестирования, которые будут использоваться.
- Определить необходимые ресурсы для проведения работ по тестированию.
- Привести результаты тестирования.

Результаты будут отправлены преподавателю в виде отчетов.

1.3. Описание продукта

Программа представляет собой визуализатор алгоритма Прима. Визуализатор алгоритма Прима - это программное средство, которое позволяет наглядно отобразить процесс работы алгоритма и его результаты.

Основная цель визуализатора алгоритма Прима - помочь пользователям лучше понять и визуально представить, как алгоритм Прима выбирает ребра для построения минимального остовного дерева. Визуализатор представляет собой графическое изображение графа, на котором происходит выполнение алгоритма. Каждый шаг алгоритма отображен на графе, показывая, какие ребра были добавлены в остовное дерево на данном шаге.

1.4. Область применения

Алгоритм Прима широко применяется в различных областях, включая:

1. Сетевое планирование: Алгоритм Прима может использоваться для определения минимальной стоимости построения сети связей между различными узлами или точками.

2. Телекоммуникации: Алгоритм Прима может быть использован для оптимизации маршрутизации данных в сети связи, чтобы минимизировать затраты на передачу данных.

3. Транспортная логистика: Алгоритм Прима может быть применен для определения оптимальных маршрутов доставки грузов, чтобы минимизировать расходы на транспортировку.

4. Графический дизайн: Алгоритм Прима может использоваться для создания минимального остовного дерева для соединения различных компонентов графического дизайна, таких как точки или линии.

5. Биоинформатика: Алгоритм Прима может быть применен для анализа генетических данных и построения эволюционных деревьев, чтобы определить связи между различными видами или организмами.

2. Стратегия тестирования

2.1. Типы тестирования

Функциональное: цель функционального тестирования состоит в том, чтобы убедиться, что весь программный продукт работает в соответствии с требованиями, и в приложении не появляется существенных ошибок.

Нефункциональное: Нефункциональное тестирование описывает тесты, необходимые для определения характеристик программного обеспечения, которые могут быть измерены различными величинами.

- Нагрузочное тестирование
- Объёмное тестирование

2.2. Подход к тестированию

Подход к тестированию визуализатора алгоритма Прима следующий:

- Идентификация функциональных требований: определение основных функций и возможностей визуализатора, которые требуется протестировать.
- Разработка тестовых сценариев: создание набора тестовых сценариев, которые покрывают все основные функции и возможности визуализатора. Каждый сценарий должен содержать шаги для проверки конкретных требований.
- Разработка тестовых данных: создание тестовых данных, которые будут использоваться при выполнении тестовых сценариев. Это может включать создание графов различной сложности, со случайными или заданными свойствами.
- Выполнение тестовых сценариев: запуск тестовых сценариев с использованием визуализатора и проверка результатов выполнения каждого шага и конечного результата. При необходимости можно использовать ручное или автоматическое сравнение результатов с ожидаемыми значениями.
- Регистрация и анализ результатов: фиксация результатов выполнения каждого тестового сценария и анализ полученных данных. Если обнаружены ошибки, они должны быть документированы и переданы разработчикам для исправления.

2.3. Критерии завершения тестирования

Критерий завершения тестирования визуализатора алгоритма Прима определяется как достижение требуемого уровня качества и надежности, а также выполнение всех функциональных требований и ожиданий пользователей. Это может быть подтверждено успешным выполнением всех тестовых сценариев, отсутствием критических ошибок.

3. Ресурсы

3.1. Оборудование и программное обеспечение

Для тестирования визуализатора алгоритма Прима необходимо следующее оборудование и программное обеспечение:

- Компьютер или сервер с достаточными ресурсами для запуска визуализатора и обработки данных.
- Операционная система, совместимая с выбранным визуализатором.
- IntelliJ IDEA

Тестирование может проводиться как на реальном оборудовании и программном обеспечении, так и на виртуальных машинах или в контейнерах, чтобы иметь возможность воспроизвести и исправить ошибки.

3.2. Тестовые данные

Тестовые данные для визуализатора алгоритма Прима включают различные типы графов, такие как деревья, связные графы, ориентированные графы и т.д. Каждый граф имеет свои характеристики, такие как количество вершин и ребер, веса ребер и т.д.

Тестовые сценарии включают:

- Визуализацию процесса построения минимального остовного дерева по алгоритму Прима.

- Визуализацию различных шагов алгоритма Прима, например, выбор следующего ребра.
- Визуализацию различных конфигураций графа, например, с разными начальными вершинами или разными весами ребер.
- Тестирование производительности и нагрузки, например, проверка скорости визуализации для больших графов или множества одновременных пользователей.

Тестовые данные могут быть созданы вручную. Важно также учесть различные граничные случаи и ошибочные ситуации, чтобы убедиться, что визуализатор обрабатывает их корректно и не выдает неправильные результаты.

4. План тестирования

4.1.1. Тестирование основных функций продукта

- Тест на граф с отрицательными весами ребер:
  - Входные данные: граф с несколькими вершинами и ребрами, имеющими отрицательные веса.
  - Ожидаемый результат: алгоритм должен вернуть минимальное остовное дерево, учитывая отрицательные веса ребер.
- Тест на граф с одинаковыми весами ребер:
  - Входные данные: граф с несколькими вершинами и ребрами, имеющими одинаковые веса.
  - Ожидаемый результат: алгоритм должен вернуть минимальное остовное дерево, учитывая одинаковые веса ребер.
- Тест на граф с большим количеством вершин и ребер:
  - Входные данные: граф с большим количеством вершин и ребер.
  - Ожидаемый результат: алгоритм должен вернуть минимальное остовное дерево, учитывая все вершины и связанные ребра в графе.

- Тест на граф с множеством возможных минимальных остовных деревьев:
  - Входные данные: граф с несколькими вершинами и ребрами, где существует несколько различных минимальных остовных деревьев.
  - Ожидаемый результат: алгоритм должен вернуть одно из возможных минимальных остовных деревьев, не обязательно совпадающее с другими возможными решениями.
- Тест на граф с большим количеством вершин и небольшим количеством ребер:
  - Входные данные: граф с большим количеством вершин и небольшим количеством ребер.
  - Ожидаемый результат: алгоритм должен вернуть минимальное остовное дерево, учитывая все вершины и связанные ребра в графе, даже при ограниченном количестве ребер.

4.1.2. Тестирование граничных условий

- Тест на пустой граф:
  - Входные данные: пустой граф (без вершин и ребер).
  - Ожидаемый результат: алгоритм должен вернуть пустое минимальное остовное дерево.
- Тест на граф с одной вершиной:
  - Входные данные: граф с одной вершиной.
  - Ожидаемый результат: алгоритм должен вернуть пустое минимальное остовное дерево.
- Тест на граф без ребер:
  - Входные данные: граф с несколькими вершинами, но без ребер.
  - Ожидаемый результат: алгоритм должен вернуть пустое минимальное остовное дерево.
- Тест на граф с одним ребром:
  - Входные данные: граф с двумя вершинами и одним ребром.

- Ожидаемый результат: алгоритм должен вернуть минимальное остовное дерево, состоящее из этого единственного ребра.

– Тест на граф с циклом:
  - Входные данные: граф с несколькими вершинами и циклическим путем.
  - Ожидаемый результат: алгоритм должен вернуть минимальное остовное дерево, исключив ребра, образующие цикл

4.2. Тестирование производительности

4.1.3. Тестирование производительности при нагрузке

Для тестирования производительности алгоритма Прима при нагрузке используется следующий сценарий:

a) Создается случайный связный граф с большим количеством вершин и ребер.
b) Замеряется время выполнения алгоритма Прима для построения минимального остовного дерева на данном графе.
c) Повторяются шаги 1-2 несколько раз, чтобы получить среднее время выполнения.
d) Происходит увеличение размера графа, и повторяются шаги 1-3 для разных размеров графов.
e) Происходит сравнение полученных результатов и оценка производительность алгоритма Прима при различных нагрузках.

4.1.4. Тестирование времени отклика

Для тестирования времени отклика алгоритма Прима используется следующий сценарий:

a) Создается случайный связный граф с небольшим количеством вершин и ребер.
b) Замеряется время выполнения алгоритма Прима для построения минимального остовного дерева на данном графе.

c) Повторяются шаги 1-2 несколько раз, чтобы получить среднее время выполнения.
d) Происходит изменение нагрузки на алгоритм, добавляя или удаляя вершины и ребра в графе, и повторять шаги 1-3 для разных нагрузок.
e) Сравниваются полученные результаты, и оценивается, как время отклика алгоритма Прима изменяется при различных нагрузках.

5. Риски и ограничения

5.1. Идентификация потенциальных рисков

Потенциальные риски при тестировании алгоритма Прима могут быть следующими:

- Неправильная реализация алгоритма.
- Недостаточное покрытие тестами: Если не все возможные сценарии использования алгоритма Прима покрыты тестами, то могут быть упущены ошибки или проблемы в его работе. Важно создать достаточное количество тестовых случаев, чтобы проверить все возможные варианты работы алгоритма.
- Неправильный выбор тестовых данных: Если тестовые данные не являются репрезентативными для реальных сценариев использования алгоритма Прима, то результаты тестирования могут быть искажены.
- Проблемы с производительностью: Если алгоритм Прима работает слишком медленно или требует слишком много ресурсов, то это может быть проблемой в реальных сценариях использования. Важно оценить производительность алгоритма и убедиться, что он работает достаточно быстро и эффективно.
- Неправильная интерпретация результатов: Если результаты тестирования неправильно интерпретируются или анализируются, то могут быть сделаны неверные выводы о работе алгоритма Прима.

5.2. Описание ограничений тестирования

Ограничения тестирования алгоритма Прима могут включать:

- Ограниченные ресурсы: Тестирование может быть ограничено доступными ресурсами, такими как вычислительная мощность или память. Это может ограничить возможность проведения тестов на больших наборах данных или с большими графами.
- Ограниченное время: Время, выделенное для тестирования, может быть ограничено. Это может ограничить количество и сложность тестов, которые могут быть выполнены.
- Ограничения на входные данные: Алгоритм Прима может иметь ограничения на входные данные, например, размер графа или тип данных, которые могут быть обработаны. Тестирование должно учитывать эти ограничения и проверять, как алгоритм обрабатывает граничные случаи.
- Ограничения на выходные данные: Алгоритм Прима может иметь ограничения на выходные данные, например, максимальный размер остовного дерева или формат вывода. Тестирование должно учитывать эти ограничения и проверять соответствие результатов алгоритма ожидаемым выходным данным.

6. Расписание тестирования

Планируемые даты окончания каждого типа тестирования:

7 июля (пятница) – тестирование основных функций продукта.

10 июля (понедельник) – тестирование визуализации процесса построения минимального остовного дерева по алгоритму Прима. Визуализацию различных конфигураций графа, например, с разными начальными вершинами или разными весами ребер.

12 июля (среда) - тестирование производительности и нагрузки.