|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ | | | | | | | | | |
|  | | |  | | |  | | | |
| ФГБОУ ВО «Пермский государственный  национальный исследовательский университет» | | | | | | | | | |
|  | | |  | | | |  | | |
|  | | ОТЧЕТ  по лабораторной работе «Сравнение языков программирования» | | | | | |  | |
|  | | |  | | | |  | | |
|  | Работу выполнил  студент гр. ПМИ-1, 2016 | | |  | Проверил  \_\_\_\_\_\_\_\_\_\_\_\_\_кафедры МОВС  (доц., ст.преп., асс.)  ФИО \_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2018 | | | |  |
| Миков Р.О. \_\_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2018 | | |
|  |  | | |  |  | | | |  |
| Пермь 2018 | | | | | | | | | |

1 Постановка задачи

Ориентированный граф задан списком ребер. Каждое ребро представляет собой тройку (вершина, вершина, длина). Все длины являются натуральными числами. Найти расстояние между двумя заданными вершинами в графе. Если пути между указанными вершинами нет, вывести -1.

2 Выбранный алгоритм и его описание.

Для решения поставленной задачи был выбрал алгоритм Дейкстры. Данный алгоритм имеет сложность О(n^2).

Данный алгоритм является алгоритмом на графах, который изобретен нидерландским ученым Э. Дейкстрой в 1959 году. Алгоритмнаходит кратчайшее расстояние от одной из вершин графа до всех остальных и работает только для графов без ребер отрицательного веса.

Каждой вершине приписывается вес – это вес пути от начальной вершины до данной. Также каждая вершина может быть выделена. Если вершина выделена, то путь от нее до начальной вершины кратчайший, если нет – то временный. Обходя граф, алгоритм считает для каждой вершины маршрут, и, если он оказывается кратчайшим, выделяет вершину. Весом данной вершины становится вес пути. Для всех соседей данной вершины алгоритм также рассчитывает вес, при этом ни при каких условиях не выделяя их. Алгоритм заканчивает свою работу, дойдя до конечной вершины, и весом кратчайшего пути становится вес конечной вершины.

Шаг 1. Всем вершинам, за исключением первой, присваивается вес равный бесконечности, а первой вершине – 0.

Шаг 2. Все вершины не выделены.

Шаг 3. Первая вершина объявляется текущей.

Шаг 4. Вес всех невыделенных вершин пересчитывается по формуле: вес невыделенной вершины есть минимальное число из старого веса данной вершины, суммы веса текущей вершины и веса ребра, соединяющего текущую вершину с невыделенной.

Шаг 5. Среди невыделенных вершин ищется вершина с минимальным весом. Если таковая не найдена, то есть вес всех вершин равен бесконечности, то маршрут не существует. Следовательно, выход. Иначе, текущей становится найденная вершина. Она же выделяется.

Шаг 6. Если текущей вершиной оказывается конечная, то путь найден, и его вес есть вес конечной вершины.

Шаг 7. Переход на шаг 4.

3 Сравнение реализации алгоритма на языках C++ и Haskell.

1. Трудоёмкость разработки: реализовать алгоритм на языке Haskell оказалось намного сложнее. В первую очередь это связано с тем, что язык Haskell – чисто функциональный. Например, на С++ программа была разработана за 40 минут, а на Haskell, в общей сложности, за 12 часов.
2. Трудоёмкость тестирования: тестировать программу на языке Haskell было намного сложнее в виду отсутствия брейк поинтов, а также того, опять же, что Haskell функциональный, а отлаживать рекурсивные алгоритмы всегда сложнее. Пример: при реализации на С++ самый продолжительный поиск бага обошёлся в 20 минут, на Haskell – 3 с половиной часа.
3. По объёму полученного кода программа на С++ получилась длиннее. В общей сложности она составила 112 строк. На Haskell – 34.
4. По быстродействию сильных отличий не обнаружилось. Тестирование производилось на графе в 6 и 8 вершин. Оба языка в течение долей секунды выводили ответ. На С++ это время составило в среднем 0.003 секунды. На Haskell это время составило 0.000264 секунды.