# Ковыляев Д. А., ПМИ-1, Вариант 3.

# 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.

# Алгоритм:

Решает задача следующим образом: есть два словаря. Первый «ways» содержит все вершины и возможные пути до других вершин вместе с расстоянием. Второй словарь «wayLen» содержит текущее известное расстояние от изначальной вершины до всех остальных (если пути нет, значение -1).

1. Начало
2. Сделать текущей вершиной ту, от которой идёт поиск. Текущую длину пути сделать равной нулю
3. Пока wayLen содержит вершины, в которые можно попасть (расстояние не равно -1)
   1. Вычислить новый wayLen
   2. Сделать текущей вершину, у которой минимальным расстояние в wayLen. Текущую длину установить равной минимальному расстоянию в wayLen
   3. Если текущая вершина совпадает с вершиной, до которой ведётся поиск, вывести текущий путь. Иначе удалить из wayLen текущую вершину
4. Результат работы алгоритма -1.
5. Конец

Вычисление нового wayLen:

1. Начало
2. Для всех вершин ver из wayLen делать следующее
   1. Если длина от текущей вершины до ver в словаре ways + текущая длина пути < длина пути, хранящаяся в wayLen или если ver = -1, то изменить соответствующее значение на данную сумму.
3. Конец

# Сравнение языков:

По трудоёмкости разработки считаю C# более простым для данного алгоритма, так как в C# удобное использование выбранного типа данных – Dictionary.

Так как разработка на C# происходила в Visual studio, отладка происходила быстро и легко. В Haskell разработка велась через приложение WinGHCi, где отлаживать программу достаточно сложно. Для отладки отдельных компонентов на языке Haskell приходилось отдельно компилировать эти компоненты и проверять на правильность результата.

Программы содержат почти одинаковое количество кода, однако на C# по итогу меньше. На Haskell 150 строчек, включая функции, более не используемые в основной программе. На C# в двух файлах по 60 строчек, всего 120.

Для оценки скорости работы была написана программа, составляющая случайных граф, состоящий из 200 вершин. У каждой вершины есть в три раза меньше, чем общее количество вершин путей в другие вершины, то есть 66. Программы запускались между 50-ой и 150-ой вершинами.

Программа на C# работает за 0.03 секунды в среднем. Haskell выдавал результат, равный 0.56. При поиске от 1 до несуществующей вершины 202 результат был на C# 0.05с на Haskell 1.96с.