ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОМСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Информатика и вычислительная техника»

Отчёт по лабораторной работе № 5

по дисциплине

«Проектирование и тестирование программного обеспечения»

Выполнил:

Студент гр. ПИН-222

Лубенский И. В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подп., дата)

Проверил:

Старший преподаватель каф. ИВТ

Блохин А. В. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

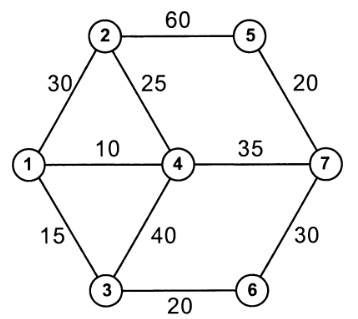
(подп., дата)

Омск, 2024

**ОПИСАНИЕ ЗАДАЧИ “ЭКСКУРСОВОД”**

М-р Ж. работает экскурсоводом в Республике Бангладеш. Его текущее задание состоит в том, чтобы показать группе туристов удаленный город. Как и во всех странах, определенные пары городов соединены дорогами с двусторонним движением. В каждой паре между соединенными городами и только между ними действует междугороднее автобусное сообщение, использующее дорогу, соединяющую их напрямую. В каждой автобусной службе есть определенный предел количества перевозимых пассажиров. У м-ра Ж. есть карта, на которой показаны города и соединяющие их дороги, а также пассажирские лимиты соответствующих автобусных служб.

Он не всегда может перевезти всех туристов в место назначения за одну поездку. Например, рассмотрим следующую карту дорог для семи городов, где дороги представляются ребрами, а число, написанное на каждом ребре, задает пассажирский лимит соответствующей автобусной службы.



М-ру Ж. потребуется минимум пять поездок, чтобы перевезти 99 туристов из города 1 в город 7, так как ему нужно сопровождать автобус с каждой группой. Наилучшим маршрутом будет 1 - 2 - 4 - 7.

Помогите м-ру Ж. найти маршрут, чтобы перевезти всех туристов в требуемый город за минимальное число поездок.

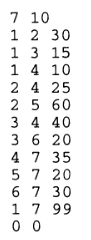
**Входные данные**

Входные данные будут состоять из одного и более тестовых блоков. Первая строка каждого блока содержит два целых числа: N (N ≤ 100) и R, задающие соответственно число городов и число дорожных сегментов. Каждая из следующих R строк содержит три целых числа (С1, С2 и Р), где C1 и С2 - это номера городов и Р (Р > 1) -это максимальное число пассажиров, которые могут перевозиться между двумя городами. Номера городов - это положительные целые числа от 1 до N.  
(R + 1)-я строка содержит три целых числа (S, D и T), задающих, соответственно начальный город, конечный город и число туристов, которых необходимо перевезти. Входные данные завершаются при N и R равных нулю.

**Выходные данные**

Для каждого введенного тестового блока сначала выведите номер сценария, а затем на отдельной строке минимальное число необходимых поездок. После каждого тестового блока выведите пустую строку.

**Пример входных данных**



**Соответствующие выходные данные**

Scenario #1

Minimum Number of Trips = 5

**АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ “ЭКСКУРСОВОД”**

Схема алгоритма решения поставленной задачи показана на рисунке 1.

Рисунок 1 — Схема составленного алгоритма

Исходный код программы показан на рисунке 2.

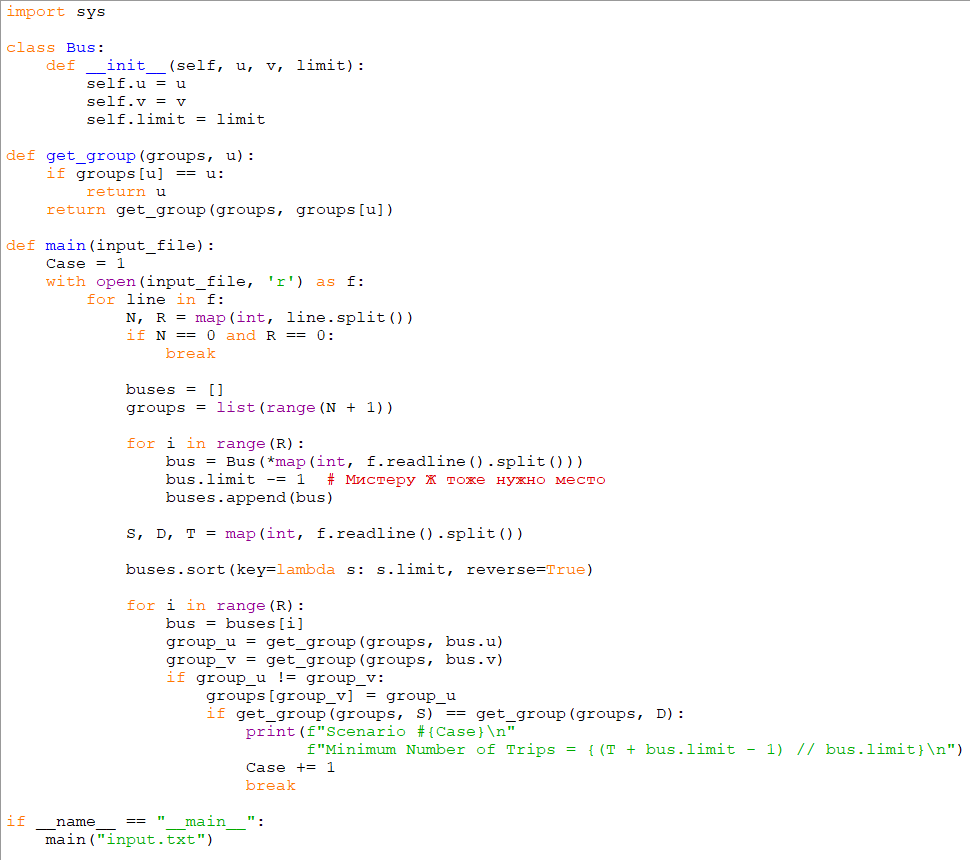


Рисунок 2 — Исходный код для решения задачи

Программа читает данные из файла, содержащего информацию о количестве городов, количестве автобусных маршрутов, информацию о каждом маршруте (начальная и конечная точки, ограничение на количество мест в автобусе), начальную и конечную точки путешествия, а также время, необходимое для путешествия.

В начале работы программы создается класс Bus, который представляет собой автобусный маршрут с начальной и конечной точками и ограничением на количество мест. Функция get\_group используется для определения группы, к которой принадлежит точка, используя алгоритм поиска сжатия пути.

В функции main происходит чтение данных из файла и инициализация списка автобусов. Для каждого автобуса уменьшается лимит мест на 1, чтобы учесть место для Мистера Ж. Автобусы сортируются по убыванию лимита мест.

Далее программа проверяет, можно ли добраться от начальной точки до конечной, используя автобусы в указанном порядке. Если это возможно, выводится минимальное количество поездок, необходимых для путешествия, учитывая время путешествия и лимит мест в автобусе.

**ТЕСТОВЫЕ ДАННЫЕ И РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

На рисунке 3 показаны входные данные, находящиеся в файле.

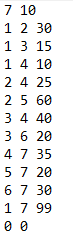


Рисунок 3 — Входные данные

Входные данные были полностью взяты из методических указаний.

Результат обработки указанных входных данных показан на рисунке 4.

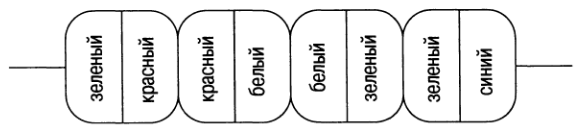


Рисунок 4 — Результат работы программы

Результаты обработки тестов, взятых из методических указаний, сходятся с выводом, указанном там же, что свидетельствует о правильности работы программы.

**ОПИСАНИЕ ЗАДАЧИ “ОЖЕРЕЛЬЕ”**

У Мэри есть красивое ожерелье, сделанное из цветных бусин. Любые две соседние бусины в ожерелье имеют один и тот же цвет в точке касания, как показано ниже.



Но, ах! Ожерелье порвалось, и бусины разлетелись по всему полу. Мэри постаралась собрать их все, но она не уверена, что у нее это получилось. Тогда она попросила вас помочь. Она хочет знать, можно ли собрать ожерелье по тому же принципу, что и старое, из тех бусин, что у нее есть. И если можно, то как?

**Входные данные**

Первая строка входных данных содержит целое число *Т*, задающее число тестовых блоков. Первая строка каждого тестового блока содержит целое число N (5 ≤ N ≤ 1000), задающее количество бусин, найденных Мэри. Каждая из следующих N строк содержит два целых числа, задающих цвета бусины. Цвета задаются целыми числами от 1 до 50.

**Выходные данные**

Для каждого тестового блока выведите его номер как показано в примере выходных данных. Если восстановить ожерелье невозможно, выведите строку, содержащую "impossibleee". В противном случае выведите N строк, каждая из которых содержит описание одной бусины. Для любого i (1 ≤ i ≤ N- 1) второе число строки i должно совпадать с первым числом строки i + 1. Кроме того, второе число строки N должно равняться первому числу строки 1 (должна выполняться собираемость ожерелья). Если возможны несколько решений, то подойдет любое.

Выходные данные для двух последовательных блоков должны быть разделены пустой строкой.

|  |  |
| --- | --- |
| **Пример входных данных** | **Выходные данные**  CASE #1  Impossibleee  CASE #2  2 1  1 3  3 4  4 2  2 2 |

**АЛГОРИТМ РЕШЕНИЯ ЗАДАЧИ “ОЖЕРЕЛЬЕ”**

Схема алгоритма решения поставленной задачи показана на рисунке 5.

Рисунок 5 — Схема алгоритма решения задачи

Исходный код программы показан на рисунке 6.

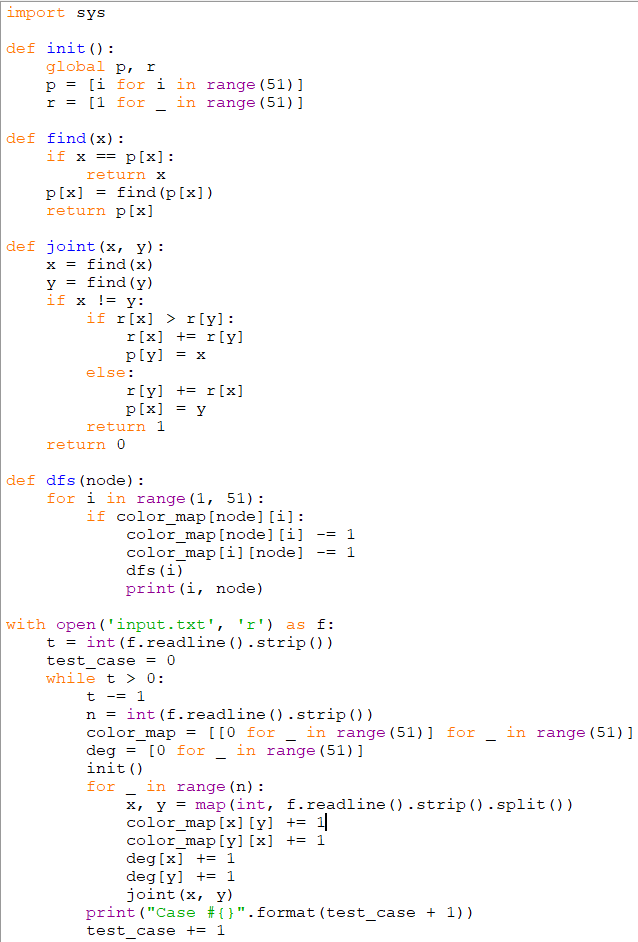


Рисунок 6 — Исходный код решения задачи

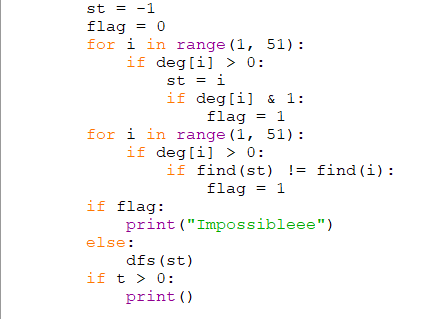


Рисунок 6.1 — Исходный код решения задачи

**Функция** (init): Создает два глобальных списка p и r размером 51. p используется для отслеживания родителя каждого элемента, а r для хранения размера каждого набора.

**Функция** (find): Реализует алгоритм поиска сжатия пути для нахождения корня элемента в дереве сжатия пути. Это используется для определения, к какому набору принадлежит элемент.

**Функция** (joint): Служит для объединения двух наборов. Если два элемента принадлежат разным наборам, они объединяются в один набор, и размер нового набора увеличивается на размер одного из исходных наборов.

**DFS (Функция Глубина-Первым-Поиск)** (dfs): Реализует обход в глубину для вывода пар элементов, которые должны быть соединены в ожерелье.

**Основной блок кода**: Читает входные данные из файла input.txt, инициализирует структуры данных, обрабатывает каждый тестовый случай, выполняет операции слияния и, наконец, использует DFS для вывода пар элементов, которые должны быть соединены.

**ТЕСТОВЫЕ ДАННЫЕ И РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

На рисунке 7 показаны входные данные, находящиеся в файле.



Рисунок 7 — Входные данные для задачи

Результат обработки входных данных показаны на рисунке 8.

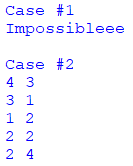


Рисунок 8 — Результат работы программы

Программа верно обрабатывает оба теста из методических указаний. Отличие полученного написанной программной ответа на второй тест от ответа в методических указаниях заключается в том, какими цветами были соединены “бусины”, что является допустимым и указано непосредственно в методических указаниях (если есть несколько ответов, то подойдёт любой). Их набор и возможность соединения сходятся в обоих ответах.