

Министерство образования и науки РФ  
Федеральное государственное бюджетное образовательное учреждение высшего  
образования  
«Омский государственный технический университет»  
Кафедра «Информатика и вычислительная техника»

Отчёт по лабораторной работе № 3  
по дисциплине  
«Проектирование и тестирование программного обеспечения»

Выполнил:  
Студент гр. ПИН-211  
Сероухов Е.С. \_\_\_\_\_  
(подп., дата)

Проверил:  
Старший преподаватель каф. ИВТ  
Карабцов Р.Д. \_\_\_\_\_  
(подп., дата)

## ВВЕДЕНИЕ

Цель работы: научиться работать со строками и сортировать их.  
Решить поставленную задачу.

Задача работы: Задача 3.1 (Дубликаты)

PC/UVaIDs: 110307/10150

Дублетом называются два слова, которые отличаются ровно в одной букве (например, «booster» и «rooster», или «rooster» и «roaster», или «roaster» и «roasted»).

Вам задается словарь длиной не более 25 143 слов, состоящих из строчных букв, причем длина каждого слова не превышает 16 букв. Далее вам задается некоторое число пар слов. Для каждой пары слов найдите последовательность слов, имеющую наименьшую длину, причем первое слово последовательности должно совпадать с первым словом из заданной пары, а последнее слово последовательности - со вторым словом из пары. Каждая пара соседних слов последовательности должна быть дублетом. Например, если вам задана пара «booster» и «roasted», возможным решением является «booster», «rooster», «roaster», «roasted», при условии, что все эти слова присутствуют в словаре.

### Входные данные

Файл входных данных содержит словарь, за которым следует некоторое число пар слов. Словарь состоит из произвольного числа слов, по одному на строку, и завершается пустой строкой. Далее идут пары слов; каждая строка содержит пару слов, разделенных пробелом.

### Выходные данные

Для каждой введенной пары напечатайте набор строк, начинающийся с первого слова и заканчивающийся последним. Каждая пара соседних строк должна быть дублетом.

Если существует несколько вариантов минимальных решений, то подойдет любое. Если решения не существует, выведите строку "No solution". Между блоками должна быть пустая строка.

### ***Пример входных данных***

```
Booster
rooster
roaster
coasted
roasted
coastal
postal

booster roasted
coastal postal
```

### ***Соответствующие выходные данные***

```
booster
rooster
roaster
roasted

No solution.
```

## РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

1. Для решения задачи был написан код на ЯП Java в среде IntelliJ IDEA 2022.1.2.

Входные данные загружаются из файла “dictionary.txt” (рисунок 1).

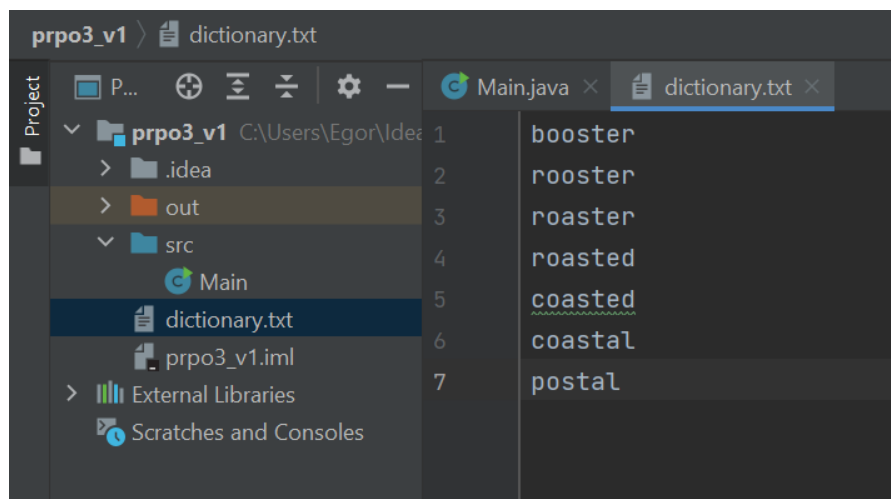


Рисунок 1 – Входные данные

```
import java.io.FileReader;
import java.io.IOException;
import java.util.*;

public class Main {
    private static Map<String, Set<String>> graph;
    final static List<String> dictionary = new ArrayList<>();

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        dictionaryEntry("dictionary.txt");
        String[] words = new String[dictionary.size()];
        for (int i = 0; i < words.length; i++)
            words[i] = dictionary.get(i);

        graph = new HashMap<>();
        buildGraph(words);

        System.out.print("Введите количество пар слов: ");
```

```

int N = scanner.nextInt();

for (int i = 0; i < N; i++) {
    System.out.print("A" + (i + 1) + ": ");
    String start = scanner.next();
    System.out.print("B" + (i + 1) + ": ");
    String end = scanner.next();
    List<String> path = findShortestPath(start, end);
    if (path != null)
        for (String word : path)
            System.out.print(word + " ");

    else
        System.out.println("No path found");
    System.out.println();
}

}

private static void buildGraph(String[] words) {
    for (int i = 0; i < words.length; i++)
        for (int j = i + 1; j < words.length; j++) // нет смысла
            // обходить во внутреннем цикле все слова, матрица смежности
            if (isDoublet(words[i], words[j])) {
                addEdge(words[i], words[j]); // соединяем i с j
                addEdge(words[j], words[i]); // соединяем j с i
            }
}

// строим граф, вершинах которого - слова, явл дуплетом хотябы для еще
// одного слова; иначе слово нет смысла включать в граф
private static void addEdge(String from, String to) { // Map<String,
    Set<String>> graph
    if (!graph.containsKey(from)) // если в графе еще нет такой
        // вершины
        graph.put(from, new HashSet<>()); // добавляем новую вершину

    graph.get(from).add(to); // добавляет "to" в множество
    // вершины "from"
}

private static boolean isDoublet(String word1, String word2) {
    if (word1.length() != word2.length())
        return false;

    int diffCount = 0;
    for (int i = 0; i < word1.length(); i++)
        if (word1.charAt(i) != word2.charAt(i)) {
            diffCount++;
            if (diffCount > 1)
                return false;
        }
    return diffCount == 1;
}

// является ли пара слов дуплетом (true/false)
private static List<String> findShortestPath(String start, String end) {
    Map<String, String> parentMap = new HashMap<>();
    Queue<String> queue = new LinkedList<>();
    Set<String> visited = new HashSet<>();

    queue.add(start);
    visited.add(start); // начало автоматически посещено

    while (!queue.isEmpty()) { // пока не обойдем в queue все элементы
        String current = queue.remove(); // извлекаем из всего списка
        // в текущую вершину
        if (current.equals(end)) // если текущая вершина =
            // концу, то
            return buildPath(parentMap, start, end); // путь завершен
    }
}

```

```

        for (String neighbor : graph.getOrDefault(current, new
HashSet<>())) {
            /*
            /\
            ||
            Возвращает значение, с которым сопоставлен указанный
ключ, или значение по умолчанию,
            если данная карта не содержит сопоставления для данного
ключа.
            */

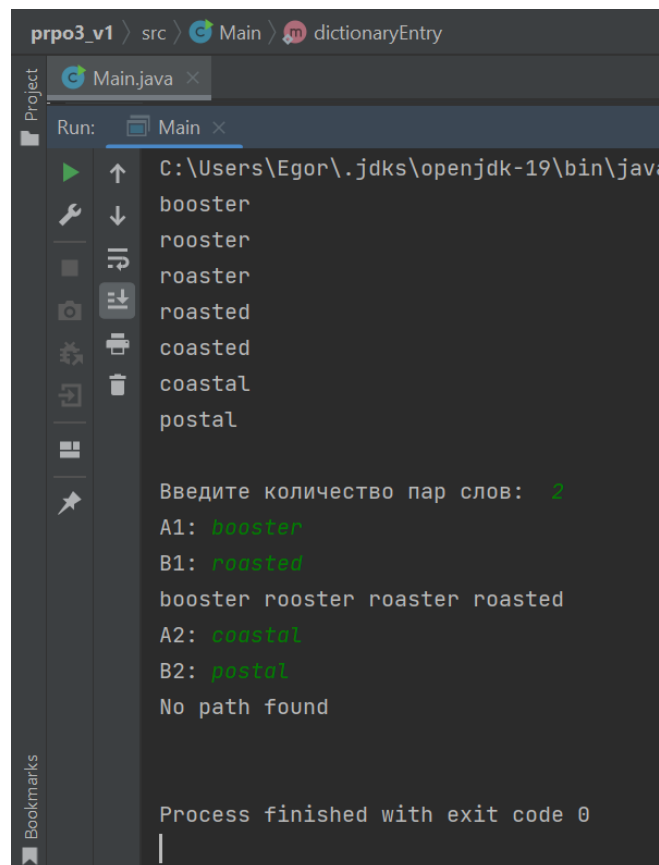
            if (!visited.contains(neighbor)) {
                visited.add(neighbor);
                parentMap.put(neighbor, current);
                queue.add(neighbor);
            }
        }
        return null;
    }

    private static List<String> buildPath(Map<String, String> parentMap,
String start, String end) {
        List<String> path = new ArrayList<>(); // хранение вершин в пути
        String current = end;
        while (current != null) {
            path.add(current); // добавляем текущую вершину в путь
            current = parentMap.get(current); // присваиваем текущую
вершину своему родителю (из которой вышла текущая)
        }
        Collections.reverse(path); // т.к. мы шли из конца, то нужно
перевернуть путь
        // если первый элемент списка path равен start, то возвращаем path,
иначе , вернуть null (для проверки)
        return path.get(0).equals(start) ? path : null;
    }

    public static void dictionaryEntry(String path) {
        try (FileReader fr= new FileReader(path)) {
            Scanner scan = new Scanner(fr);
            while (scan.hasNextLine())
                dictionary.add(scan.nextLine());
        }
        catch (IOException ex) {
            System.out.println(ex.getMessage());
        }
        for (String s : dictionary)
            System.out.println(s);
    } // заполнение и печать словаря в консоль
}

```

2. Было проведено тестирование работы программы (рисунок 2).



```
prpo3_v1 > src > Main > dictionaryEntry
Main.java x
Run: Main x
C:\Users\Egor\.jdk\openjdk-19\bin\java
booster
rooster
roaster
roasted
coasted
coastal
postal

Введите количество пар слов: 2
A1: booster
B1: roasted
booster rooster roaster roasted
A2: coastal
B2: postal
No path found

Process finished with exit code 0
```

Рисунок 2 – Тестирование работы программы.

## ЗАКЛЮЧЕНИЕ

Вывод: в ходе работы была написана программа на ЯП Java. Программа решила поставленную задачу и прошла проверку при тестировании.