|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ и системы  
 управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа,**

**обработки и интерпретации больших данных**

**Отчет**

|  |  |
| --- | --- |
| **по лабораторной работе №** | 6 |

**Название:**

Коллекции

**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-22М |  |  | А.А. Морозова |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

**Цель работы** — ознакомление с коллекциями в языке Java.

1. Списки (стеки, очереди) I(1..n) и U(1..n) содержат результаты n измерений тока и напряжения на неизвестном сопротивлении R. Найти приближенное число R методом наименьших квадратов.

Часть кода задания приведена в листинге 1, результат выполнения – на рисунке 1.

Листинг 1 – Вычисление сопротивления

|  |
| --- |
| public static double calcluateResistance(List<Double> I, List<Double> U) {  Iterator<Double> I\_Iter = I.iterator();  Iterator<Double> U\_Iter = U.iterator();  double sum\_up = 0;  double sum\_down = 0;  I\_Iter = I.iterator();  U\_Iter = U.iterator();  while (I\_Iter.hasNext() && U\_Iter.hasNext()) {  double i\_val = I\_Iter.next();  double u\_val = U\_Iter.next();  sum\_up += u\_val \* i\_val; // IU  sum\_down += i\_val \* i\_val; // I^2  }  double R = sum\_up / sum\_down;  return R;  } |



Рисунок 1 – Результат вычисления сопротивления

1. С использованием множества выполнить попарное суммирование произвольного конечного ряда чисел по следующим правилам: на первом этапе суммируются попарно рядом стоящие числа, на втором этапе суммируются результаты первого этапа и т.д. до тех пор, пока не останется одно число.

Часть кода задания приведена в листинге 2, результат выполнения – на рисунке 2.

Листинг 2 – Выполнение попарного суммирования

|  |
| --- |
| public static int pairSum(List<Integer> nums) {  Set<Integer> s = new LinkedHashSet<>(nums);  Set<Integer> newS = new LinkedHashSet<>();  Iterator<Integer> it = s.iterator();  while (s.size() > 1) {  Integer n1 = it.next();  while (it.hasNext()) {  Integer n2 = it.next();  newS.add(n1 + n2);  n1 = n2;  }  s = new LinkedHashSet<>(newS);  it = s.iterator();  newS.clear();  }  return s.iterator().next();  } |

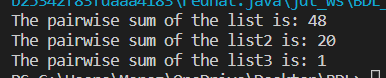


Рисунок 2 – Попарное суммирование

1. Во входном файле хранятся две разреженные матрицы А и В. Построить циклически связанные списки СА и СВ, содержащие ненулевые элементы соответственно матриц А и В. Просматривая списки, вычислить: а) сумму S = A + B; б) произведение P = A \* B.

Часть кода задания приведена в листинге 3, результат выполнения – на рисунке 3.

Листинг 3 – Умножение матриц

|  |
| --- |
| @ public SparseMatrix multiply(SparseMatrix mtx) {  int[][] matrix = new int[n][mtx.m];  for (int i = 0; i < n; i++) {  Node node1 = nodeHeads.get(i);  while (node1 != null) {  Node node2 = mtx.nodeHeads.get(node1.col);  while (node2 != null) {  matrix[i][node2.col] += node1.value \* node2.value;  node2 = node2.next;  }  node1 = node1.next;  }  }  return new SparseMatrix(matrix);  } |

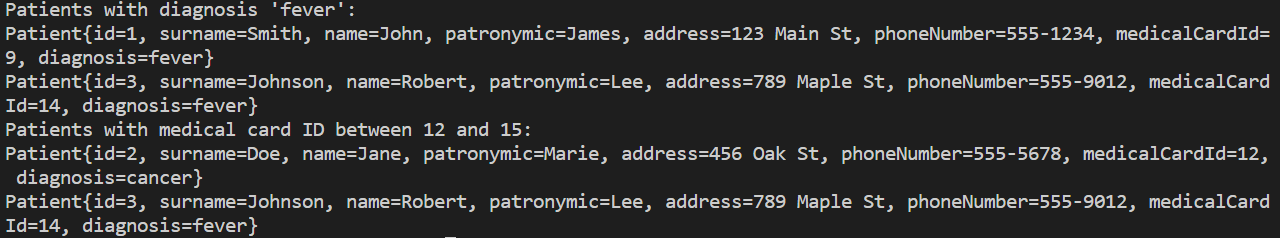


Рисунок 3 – Вывод по условиям задачи

1. Во входном файле хранятся наименования некоторых объектов. Построить список C1, элементы которого содержат наименования и шифры данных объектов, причем элементы списка должны быть упорядочены по возрастанию шифров. Затем “сжать” список C1, удаляя дублирующие наименования объектов.

Часть кода задания приведена в листинге 4, результат выполнения – на рисунке 4.

Листинг 4 – Сжатие списка

|  |
| --- |
| public static LinkedList<String> compressList(LinkedList<String> list) {  LinkedList<String> compressedList = new LinkedList<>();  int idx = 0;  boolean isDuplicate;  while (idx < list.size()) {  isDuplicate = false;  for (int i = 0; i < compressedList.size(); i++) {  if (list.get(idx).split(" ")[0].equals(compressedList.get(i).split(" ")[0])) {  isDuplicate = true;  break;  }  }  if (!isDuplicate) {  compressedList.add(list.get(idx));  }  idx++;  }  return compressedList;  } |

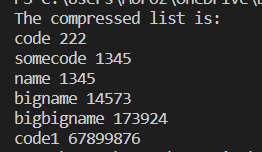


Рисунок 4 – Сжатый список

Полный код заданий размещен в репозитории по ссылке – https://github.com/moroz-matros/BDL.

**Вывод** — в ходе работы были получены навыки работы с коллекциями в языке Java.