**Министерство образования и науки Российской Федерации**

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,   
МЕХАНИКИ И ОПТИКИ**

Факультет программной инженерии и компьютерной техники  
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Aлгоритмы и структуры данных»

**ОТЧЁТ**

по лабораторной работе №1 (Week 1 Openedu)

Студенкт Коваленко Егор группы P3217

Преподаватель Муромцев Дмитрий Ильич

Санкт-Петербург

2019 г.

Содержание

[Задача 1 «a+b» 3](#_Toc2639133)

[Исходный код к задаче 1 3](#_Toc2639134)

[Бенчмарк к задаче 1 4](#_Toc2639135)

[Задача 2 «a+b^2» 4](#_Toc2639136)

[Исходный код к задаче 2 5](#_Toc2639137)

[Бенчмарк к задаче 2 5](#_Toc2639138)

[Задача 3 Сортировка вставками 6](#_Toc2639139)

[Исходный код к задаче 3 7](#_Toc2639140)

[Бенчмарк к задаче 3 8](#_Toc2639141)

[Задача 4 Знакомство с жителями Сортлэнда 9](#_Toc2639142)

[Исходный код к задаче 4 10](#_Toc2639143)

[Бенчмарк к задаче 4 11](#_Toc2639144)

[Задача 5 Секретарь Своп 12](#_Toc2639145)

[Исходный код к задаче 5 13](#_Toc2639146)

[Бенчмарк к задаче 5 14](#_Toc2639147)

# Задача 1 «a+b»

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

В данной задаче требуется вычислить сумму двух заданных чисел.

#### Формат входного файла

Входной файл состоит из одной строки, которая содержит два целых числа a и b. Для этих чисел выполняются условия .

#### Формат выходного файла

В выходной файл выведите единственное целое число — результат сложения.

#### Примеры

|  |  |
| --- | --- |
| input.txt | output.txt |
| 23 11 | 34 |
| -100 1 | -99 |

## Исходный код к задаче 1

public static void main(String[] args) {  
 try{  
 BufferedReader reader = new BufferedReader(new InputStreamReader(new FileInputStream("home/ekov/IdeaProjects/Algos/src/input.txt")));  
 String[] input = reader.readLine().split(" ");  
 BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("home/ekov/IdeaProjects/Algos/src/output.txt")));  
 writer.write( Integer.*parseInt*(input[0]) + Integer.*parseInt*(input[1]) + "");  
 writer.close();  
 reader.close();  
 }catch (IOException e){  
 System.*out*.println(e.getMessage());  
 }  
  
  
}

## Бенчмарк к задаче 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № **теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.156 | 20987904 | 25 | 11 |
| 1 | OK | 0.109 | 20942848 | 7 | 2 |
| 2 | OK | 0.125 | 20930560 | 8 | 3 |
| 3 | OK | 0.109 | 20938752 | 5 | 1 |
| 4 | OK | 0.156 | 20914176 | 5 | 1 |
| 5 | OK | 0.125 | 20955136 | 6 | 1 |
| 6 | OK | 0.109 | 20971520 | 9 | 4 |
| 7 | OK | 0.125 | 20873216 | 23 | 10 |
| 8 | OK | 0.125 | 20934656 | 25 | 11 |
| 9 | OK | 0.156 | 20918272 | 24 | 1 |
| 10 | OK | 0.109 | 20959232 | 24 | 1 |
| 11 | OK | 0.125 | 20946944 | 14 | 10 |
| 12 | OK | 0.109 | 20873216 | 23 | 10 |
| 13 | OK | 0.109 | 20938752 | 23 | 11 |
| 14 | OK | 0.109 | 20930560 | 20 | 9 |
| 15 | OK | 0.125 | 20987904 | 23 | 11 |
| 16 | OK | 0.109 | 20938752 | 20 | 9 |
| 17 | OK | 0.109 | 20901888 | 22 | 10 |
| 18 | OK | 0.109 | 20963328 | 23 | 11 |
| 19 | OK | 0.109 | 20930560 | 22 | 10 |
| 20 | OK | 0.109 | 20922368 | 22 | 10 |
| 21 | OK | 0.125 | 20897792 | 22 | 10 |

## Задача 2 «a+b^2»

1.0 из 1.0 балла (оценивается)

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

В данной задаче требуется вычислить значение выражения .

#### Формат входного файла

Входной файл состоит из одной строки, которая содержит два целых числа a и b. Для этих чисел выполняются условия .

#### Формат выходного файла

В выходной файл выведите единственное целое число — результат вычисления выражения .

#### Примеры

|  |  |
| --- | --- |
| input.txt | output.txt |
| 23 11 | 144 |
| -100 1 | -99 |

## Исходный код к задаче 2

public static void main(String[] args) {  
 try{  
 BufferedReader reader = new BufferedReader(new InputStreamReader(new FileInputStream("input.txt")));  
 String[] input = reader.readLine().split(" ");  
 BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("output.txt")));  
 long b = Integer.*parseInt*(input[1]);  
 writer.write( Integer.*parseInt*(input[0]) + b\*b + "");  
 writer.close();  
 reader.close();  
 }catch (IOException e){  
 System.*out*.println(e.getMessage());  
 }  
  
  
}

## Бенчмарк к задаче 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № **теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.156 | 20971520 | 25 | 19 |
| 1 | OK | 0.109 | 20938752 | 7 | 3 |
| 2 | OK | 0.156 | 20963328 | 8 | 3 |
| 3 | OK | 0.140 | 20914176 | 5 | 1 |
| 4 | OK | 0.109 | 20942848 | 5 | 1 |
| 5 | OK | 0.140 | 20938752 | 6 | 1 |
| 6 | OK | 0.125 | 20889600 | 6 | 1 |
| 7 | OK | 0.109 | 20938752 | 23 | 19 |
| 8 | OK | 0.125 | 20938752 | 25 | 18 |
| 9 | OK | 0.109 | 20938752 | 24 | 18 |
| 10 | OK | 0.109 | 20951040 | 24 | 19 |
| 11 | OK | 0.109 | 20971520 | 23 | 18 |
| 12 | OK | 0.109 | 20955136 | 23 | 18 |
| 13 | OK | 0.109 | 20938752 | 20 | 15 |
| 14 | OK | 0.109 | 20901888 | 23 | 18 |
| 15 | OK | 0.109 | 20946944 | 20 | 18 |
| 16 | OK | 0.109 | 20918272 | 22 | 18 |
| 17 | OK | 0.109 | 20930560 | 23 | 18 |
| 18 | OK | 0.093 | 20893696 | 22 | 17 |
| 19 | OK | 0.109 | 20926464 | 22 | 17 |
| 20 | OK | 0.125 | 20926464 | 22 | 18 |

# Задача 3 Сортировка вставками

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Дан массив целых чисел. Ваша задача — отсортировать его в порядке неубывания с помощью сортировки вставками.

Сортировка вставками проходится по всем элементам массива от меньших индексов к большим («слева направо») для каждого элемента определяет его место в предшествующей ему отсортированной части массива и переносит его на это место (возможно, сдвигая некоторые элементы на один индекс вправо). Чтобы проконтролировать, что Вы используете именно сортировку вставками, мы попросим Вас для каждого элемента массива после того, как он будет обработан, выводить его новый индекс.

#### Формат входного файла

В первой строке входного файла содержится число  — число элементов в массиве. Во второй строке находятся различных целых чисел, по модулю не превосходящих .

#### Формат выходного файла

В первой строке выходного файла выведите n чисел. При этом i-ое число равно индексу, на который, **в момент обработки его сортировкой вставками**, был перемещен i-ый элемент **исходного массива**. Индексы нумеруются, начиная с единицы. Между любыми двумя числами должен стоять ровно один пробел.

Во второй строке выходного файла выведите отсортированный массив. Между любыми двумя числами должен стоять ровно один пробел.

#### Пример

|  |  |
| --- | --- |
| input.txt | output.txt |
| 10 1 8 4 2 3 7 5 6 9 0 | 1 2 2 2 3 5 5 6 9 1 0 1 2 3 4 5 6 7 8 9 |

#### Комментарий к примеру

В примере сортировка вставками работает следующим образом:

1. Первый элемент остается на своем месте, поэтому первое число в ответе — единица. Отсортированная часть массива: [1]
2. Второй элемент больше первого, поэтому он тоже остается на своем месте, и второе число в ответе — двойка. [1 8]
3. Четверка меньше восьмерки, поэтому занимает второе место. [1 4 8]
4. Двойка занимает второе место. [1 2 4 8]
5. Тройка занимает третье место. [1 2 3 4 8]
6. Семерка занимает пятое место. [1 2 3 4 7 8]
7. Пятерка занимает пятое место. [1 2 3 4 5 7 8]
8. Шестерка занимает шестое место. [1 2 3 4 5 6 7 8]
9. Девятка занимает девятое место. [1 2 3 4 5 6 7 8 9]
10. Ноль занимает первое место. [0 1 2 3 4 5 6 7 8 9]

## Исходный код к задаче 3

public static void main(String[] args) {  
 try{  
 BufferedReader reader = new BufferedReader(new InputStreamReader(new FileInputStream("input.txt")));  
 int l = Integer.*parseInt*(reader.readLine());  
 String[] input = reader.readLine().split(" ");  
 reader.close();  
  
 int[] output2 = new int[l];  
 output2[0] = 1;  
 for (short j = 1; j < l; j++) {  
 int key = Integer.*parseInt*(input[j]);  
 int i = j - 1;  
 while(i >= 0 && Integer.*parseInt*(input[i]) > key){  
 input[i+1] = input[i];  
 i--;  
 }  
 output2[j] = i+2;  
 input[i+1] = key + "";  
 }  
  
 BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("output.txt")));  
  
 for(int i: output2){  
 writer.write(i + " ");  
 }  
 writer.write("\n");  
 for (String s :  
 input) {  
 writer.write(s + " ");  
 }  
  
 writer.close();  
 }catch (IOException e){  
 System.*out*.println(e.getMessage());  
 }  
  
  
}

## Бенчмарк к задаче 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № **теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.250 | 24981504 | 10415 | 14297 |
| 1 | OK | 0.109 | 20930560 | 25 | 41 |
| 2 | OK | 0.109 | 21000192 | 7 | 6 |
| 3 | OK | 0.125 | 20922368 | 12 | 13 |
| 4 | OK | 0.109 | 20910080 | 8 | 9 |
| 5 | OK | 0.109 | 20926464 | 10 | 13 |
| 6 | OK | 0.109 | 20979712 | 29 | 32 |
| 7 | OK | 0.109 | 20946944 | 10 | 13 |
| 8 | OK | 0.125 | 20934656 | 10 | 13 |
| 9 | OK | 0.125 | 20873216 | 10 | 13 |
| 10 | OK | 0.109 | 20938752 | 10 | 13 |
| 11 | OK | 0.140 | 20955136 | 10 | 13 |
| 12 | OK | 0.125 | 20959232 | 57 | 64 |
| 13 | OK | 0.109 | 20910080 | 56 | 63 |
| 14 | OK | 0.125 | 20930560 | 57 | 64 |
| 15 | OK | 0.140 | 20946944 | 77 | 88 |
| 16 | OK | 0.109 | 20963328 | 76 | 87 |
| 17 | OK | 0.171 | 20987904 | 77 | 88 |
| 18 | OK | 0.125 | 20963328 | 112 | 128 |
| 19 | OK | 0.125 | 20959232 | 111 | 128 |
| 20 | OK | 0.156 | 20918272 | 110 | 126 |
| 21 | OK | 0.125 | 21839872 | 949 | 1191 |
| 22 | OK | 0.109 | 20979712 | 960 | 1220 |
| 23 | OK | 0.125 | 22028288 | 957 | 1135 |
| 24 | OK | 0.171 | 22126592 | 1490 | 1889 |
| 25 | OK | 0.125 | 21188608 | 1486 | 1945 |
| 26 | OK | 0.140 | 22609920 | 1481 | 1762 |
| 27 | OK | 0.140 | 23162880 | 3723 | 4889 |
| 28 | OK | 0.125 | 22003712 | 3729 | 5048 |
| 29 | OK | 0.234 | 23609344 | 3727 | 4438 |
| 30 | OK | 0.203 | 24006656 | 8456 | 11339 |
| 31 | OK | 0.156 | 22327296 | 8471 | 11610 |
| 32 | OK | 0.250 | 24678400 | 8415 | 10036 |
| 33 | OK | 0.156 | 24154112 | 10415 | 14036 |
| 34 | OK | 0.125 | 22495232 | 10410 | 14297 |
| 35 | OK | 0.234 | 24981504 | 10393 | 12387 |

# Задача 4 Знакомство с жителями Сортлэнда

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Владелец графства Сортлэнд, граф Бабблсортер, решил познакомиться со своими подданными. Число жителей в графстве нечетно и составляет n, где n может быть достаточно велико, поэтому граф решил ограничиться знакомством с тремя представителями народонаселения: с самым бедным жителем, с жителем, обладающим средним достатком, и с самым богатым жителем.

Согласно традициям Сортлэнда, считается, что житель обладает средним достатком, если при сортировке жителей по сумме денежных сбережений он оказывается ровно посередине. Известно, что каждый житель графства имеет уникальный идентификационный номер, значение которого расположено в границах от единицы до n. Информация о размере денежных накоплений жителей хранится в массиве M таким образом, что сумма денежных накоплений жителя, обладающего идентификационным номером i, содержится в ячейке M[i]. Помогите секретарю графа мистеру Свопу вычислить идентификационные номера жителей, которые будут приглашены на встречу с графом.

#### Формат входного файла

Первая строка входного файла содержит число жителей  (, нечетно). Вторая строка содержит описание массива M, состоящее из положительных вещественных чисел, разделенных пробелами. Гарантируется, что все элементы массива M различны, а их значения имеют точность не более двух знаков после запятой и не превышают

#### Формат выходного файла

В выходной файл выведите три целых положительных числа, разделенных пробелами — идентификационные номера беднейшего, среднего и самого богатого жителей Сортлэнда.

#### Пример

|  |  |
| --- | --- |
| input.txt | output.txt |
| 5 10.00 8.70 0.01 5.00 3.00 | 3 4 1 |

#### Комментарий к примеру

Если отсортировать жителей по их достатку, получится следующий массив:

[0.01, 3] [3.00, 5] [5.00, 4] [8.70, 2] [10.00, 1]

Здесь каждый житель указан в квадратных скобках, первое число — его достаток, второе число — его идентификационный номер. Таким образом, самый бедный житель имеет номер 3, самый богатый — номер 1, а средний — номер 4.

## Исходный код к задаче 4

public static void main(String[] args) {  
 try{  
 BufferedReader reader = new BufferedReader(new InputStreamReader(new FileInputStream("input.txt")));  
 int l = Integer.*parseInt*(reader.readLine());  
 String[] input = reader.readLine().split(" ");  
 reader.close();  
 int r = 1;  
 int p = 1;  
 int m = 1;  
 double[] toSort = new double[l];  
 double[] base = new double[l];  
 for (int i = 0; i < l; i++) {  
 toSort[i] = Double.*parseDouble*(input[i]);  
 base[i] = toSort[i];  
 }  
  
 for (int j = 1; j < l; j++) {  
 double key = toSort[j];  
 int i = j - 1;  
 while(i >= 0 && (key - toSort[i])<0.01){  
 toSort[i+1] = toSort[i];  
 i--;  
 }  
 if(i<0)  
 p = j + 1;  
 toSort[i+1] = key;  
 if(i == j-1)  
 r = j + 1;  
 }  
 double f = toSort[(l-1)/2];  
  
 for (int i = 0; i < l; i++) {  
 double diff = f-base[i];  
 if(diff < 0.01 && -diff < 0.01) {  
 m = i + 1;  
 break;  
 }  
 }  
 BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("output.txt")));  
 writer.write(p + " " + m + " " + r);  
 writer.close();  
 }catch (IOException e){  
 System.*out*.println(e.getMessage());  
 }  
  
  
}

## Бенчмарк к задаче 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № **теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 0.328 | 27516928 | 98892 | 14 |
| 1 | OK | 0.125 | 21012480 | 30 | 5 |
| 2 | OK | 0.109 | 21032960 | 33 | 5 |
| 3 | OK | 0.109 | 21037056 | 1065 | 8 |
| 4 | OK | 0.140 | 22368256 | 3732 | 10 |
| 5 | OK | 0.171 | 23031808 | 14975 | 13 |
| 6 | OK | 0.171 | 23003136 | 14998 | 11 |
| 7 | OK | 0.218 | 24653824 | 28749 | 14 |
| 8 | OK | 0.171 | 24834048 | 34791 | 12 |
| 9 | OK | 0.187 | 24944640 | 38037 | 13 |
| 10 | OK | 0.203 | 24932352 | 38074 | 14 |
| 11 | OK | 0.156 | 24956928 | 39288 | 13 |
| 12 | OK | 0.171 | 25153536 | 48638 | 13 |
| 13 | OK | 0.203 | 26112000 | 50722 | 12 |
| 14 | OK | 0.203 | 26025984 | 52757 | 14 |
| 15 | OK | 0.218 | 26112000 | 58008 | 13 |
| 16 | OK | 0.218 | 26648576 | 66504 | 14 |
| 17 | OK | 0.218 | 26664960 | 71786 | 14 |
| 18 | OK | 0.187 | 26841088 | 72346 | 14 |
| 19 | OK | 0.250 | 26763264 | 73304 | 13 |
| 20 | OK | 0.218 | 26959872 | 76139 | 14 |
| 21 | OK | 0.218 | 27099136 | 83944 | 14 |
| 22 | OK | 0.328 | 27115520 | 85179 | 13 |
| 23 | OK | 0.265 | 27144192 | 86522 | 12 |
| 24 | OK | 0.187 | 27238400 | 89202 | 13 |
| 25 | OK | 0.203 | 27516928 | 98892 | 14 |

# Задача 5 Секретарь Своп

|  |  |
| --- | --- |
| Имя входного файла: | input.txt |
| Имя выходного файла: | output.txt |
| Ограничение по времени: | 2 секунды |
| Ограничение по памяти: | 256 мегабайт |

Уже знакомый нам из предыдущей задачи граф Бабблсортер поручил своему секретарю, мистеру Свопу, оформлять приглашения беднейшему, богатейшему и среднему по достатку жителю своих владений. Однако кто же, в отсутствие мистера Свопа, будет заниматься самым важным делом — сортировкой массивов чисел? Видимо, это придется сделать Вам!

Дан массив, состоящий из

целых чисел. Вам необходимо его отсортировать по неубыванию. Но делать это нужно так же, как это делает мистер Своп — то есть, каждое действие должно быть взаимной перестановкой пары элементов. Вам также придется записать все, что Вы делали, в файл, чтобы мистер Своп смог проверить Вашу работу.

#### Формат входного файла

В первой строке входного файла содержится число — число элементов в массиве. Во второй строке находятся n целых чисел, по модулю не превосходящих . Числа могут совпадать друг с другом.

#### Формат выходного файла

В первых нескольких строках выведите осуществленные Вами операции перестановки элементов. Каждая строка должна иметь следующий формат:

Swap elements at indices X and Y.

где X и Y — различные индексы массива, элементы на которых нужно переставить . Мистер Своп любит порядок, поэтому сделайте так, чтобы X < Y.

После того, как все нужные перестановки выведены, выведите следующую фразу:

No more swaps needed.

Во последней строке выходного файла выведите отсортированный массив, чтобы мистер Своп не переделывал работу за Вас. Между любыми двумя числами должен стоять ровно один пробел.

#### Пример

|  |  |
| --- | --- |
| input.txt | output.txt |
| 5 3 1 4 2 2 | Swap elements at indices 1 and 2. Swap elements at indices 2 and 4. Swap elements at indices 3 and 5. No more swaps needed. 1 2 2 3 4 |

#### Послесловие и предостережение

Семья секретаря Свопа занималась сортировками массивов, и именно с помощью перестановок пар элементов, как минимум с XII века, поэтому все Свопы владеют этим искусством в совершенстве. Мы не просим Вас произвести минимальную последовательность перестановок, приводящую к правильному ответу. Однако учтите, что для вывода слишком длинной последовательности у Вашего алгоритма может не хватить времени (или памяти — если выводимые строки хранятся в памяти перед выводом). Подумайте, что с этим можно сделать. Решение существует!

## Исходный код к задаче 5

public static void main(String[] args) {  
 try{  
 BufferedReader reader = new BufferedReader(new InputStreamReader(new FileInputStream("input.txt")));  
 int l = Integer.*parseInt*(reader.readLine());  
 String[] input = reader.readLine().split(" ");  
 reader.close();  
 int[] toSort = new int[l];  
 for (int i = 0; i < l; i++) {  
 toSort[i] = Integer.*parseInt*(input[i]);  
 }  
 ArrayList<Integer> list= new ArrayList<Integer>();  
  
 BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(new FileOutputStream("output.txt")));  
  
 int gap = l-1;  
 while(gap >= 1){  
 int i = 0;  
 while(i + gap < l){  
 if(toSort[i] > toSort[i+gap]) {  
 int tmp = toSort[i + gap];  
 toSort[i + gap] = toSort[i];  
 toSort[i] = tmp;  
 writer.write("Swap elements at indices " + (i + 1) + " and " + (i + 1 + gap) + ".\n");  
 }  
 i++;  
 }  
 gap--;  
 }  
  
 writer.write("No more swaps needed.\n");  
 for (int i :  
 toSort) {  
 writer.write(i + " ");  
 }  
  
 writer.close();  
 }catch (IOException e){  
 System.*out*.println(e.getMessage());  
 }  
  
}

## Бенчмарк к задаче 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № **теста** | **Результат** | **Время, с** | **Память** | **Размер входного файла** | **Размер выходного файла** |
| Max |  | 1.328 | 59215872 | 51993 | 82541750 |
| 1 | OK | 0.125 | 20975616 | 14 | 168 |
| 2 | OK | 0.171 | 20934656 | 7 | 25 |
| 3 | OK | 0.125 | 20942848 | 12 | 30 |
| 4 | OK | 0.140 | 20942848 | 8 | 60 |
| 5 | OK | 0.140 | 21037056 | 10 | 28 |
| 6 | OK | 0.125 | 20938752 | 10 | 28 |
| 7 | OK | 0.125 | 20938752 | 29 | 47 |
| 8 | OK | 0.109 | 20955136 | 10 | 62 |
| 9 | OK | 0.109 | 20934656 | 10 | 62 |
| 10 | OK | 0.156 | 20959232 | 10 | 96 |
| 11 | OK | 0.125 | 20955136 | 10 | 62 |
| 12 | OK | 0.093 | 20955136 | 10 | 96 |
| 13 | OK | 0.109 | 20959232 | 50 | 136 |
| 14 | OK | 0.109 | 20967424 | 56 | 176 |
| 15 | OK | 0.125 | 20901888 | 57 | 75 |
| 16 | OK | 0.109 | 20938752 | 55 | 141 |
| 17 | OK | 0.140 | 20963328 | 75 | 297 |
| 18 | OK | 0.125 | 20942848 | 76 | 94 |
| 19 | OK | 0.109 | 20926464 | 78 | 198 |
| 20 | OK | 0.109 | 20893696 | 108 | 399 |
| 21 | OK | 0.125 | 20971520 | 107 | 124 |
| 22 | OK | 0.109 | 20922368 | 108 | 296 |
| 23 | OK | 0.140 | 21495808 | 948 | 26453 |
| 24 | OK | 0.125 | 20955136 | 947 | 964 |
| 25 | OK | 0.156 | 20983808 | 948 | 2576 |
| 26 | OK | 0.171 | 29380608 | 3720 | 404020 |
| 27 | OK | 0.125 | 21667840 | 3735 | 3751 |
| 28 | OK | 0.140 | 21790720 | 3722 | 10432 |
| 29 | OK | 0.296 | 48201728 | 8463 | 2073063 |
| 30 | OK | 0.156 | 22331392 | 8441 | 8457 |
| 31 | OK | 0.156 | 22790144 | 8434 | 23770 |
| 32 | OK | 0.531 | 56852480 | 22822 | 15923882 |
| 33 | OK | 0.156 | 23879680 | 22825 | 22840 |
| 34 | OK | 0.234 | 24608768 | 22877 | 65745 |
| 35 | OK | 1.328 | 59215872 | 51987 | 82541750 |
| 36 | OK | 0.156 | 24715264 | 51940 | 51955 |
| 37 | OK | 0.250 | 25358336 | 51993 | 150901 |