Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики»

ОТЧЁТ

«Алгоритмы и структуры данных»

Неделя №1

Выполнил студент группы P3217:
Прохоров Д.А.

Проверил:
Романов А.А.
Волчек Д.Г.

г. Санкт-Петербург
2018

1

Задача «a+b»

ЭТОТ ЭЛЕМЕНТ КУРСА ОЦЕНИВАЕТСЯ КАК 'ЛАБОРАТОРНАЯ РАБОТА'
ВЕС: 1.0

Добавить страницу в мои закладки

Задача «a+b»

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 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 $-10^9 \le a \le 10^9$, $-10^9 \le b \le 10^9$.

Формат выходного файла

В выходной файл выведите единственное целое число — результат сложения $a + b$.

Примеры

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

Листинг l1.scala:

```
import scala.io._
import java.io._

object l1 {
  def main(args: Array[String]): Unit = {
    val input = "input.txt"
    val output = "output.txt"
    val outfile = new File(output)
    val bw = new BufferedWriter(new FileWriter(outfile))
    for (line <- Source.fromFile(input).getLines) {
      bw.write(line.split(' ').map(_.toInt).sum.toString + "\n")
    }
    bw.close()
  }
}
```

Результаты:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.812	66605056	25	12
1	OK	0.765	66469888	7	3
2	OK	0.796	66473984	8	4
3	OK	0.781	66531328	5	2
4	OK	0.718	66490368	5	2
5	OK	0.812	66494464	6	2
6	OK	0.734	66531328	9	5
7	OK	0.796	66584576	23	11
8	OK	0.750	66514944	25	12
9	OK	0.734	66519040	24	2
10	OK	0.812	66543616	24	2
11	OK	0.796	66547712	14	11
12	OK	0.781	66498560	23	11
13	OK	0.718	66605056	23	12
14	OK	0.765	66486272	20	10
15	OK	0.781	66547712	23	12
16	OK	0.750	66531328	20	10
17	OK	0.750	66392064	22	11
18	OK	0.765	66478080	23	12
19	OK	0.796	66588672	22	11
20	OK	0.687	66539520	22	11
21	OK	0.765	66465792	22	11

Задача «a+b^2»

ЭТОТ ЭЛЕМЕНТ КУРСА ОЦЕНИВАЕТСЯ КАК 'ЛАБОРАТОРНАЯ РАБОТА'
ВЕС: 1.0

Добавить страницу в мои закладки

Задача «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 $a + b^2$.

Формат входного файла

Входной файл состоит из одной строки, которая содержит два целых числа a и b. Для этих чисел выполняются условия $-10^9 \le a \le 10^9$, $-10^9 \le b \le 10^9$.

Формат выходного файла

В выходной файл выведите единственное целое число — результат вычисления выражения $a + b^2$.

Примеры

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

Листинг l2.scala:

```
import scala.io._
import java.io._
import scala.math.BigInt

object l2 {
  def main(args: Array[String]): Unit = {
    val input = "input.txt"
    val output = "output.txt"
    val outfile = new File(output)
    val bw = new BufferedWriter(new FileWriter(outfile))
    for (line <- Source.fromFile(input).getLines) {
      val arr = line.split(' ').map(_.toLong)
      val a : BigInt = arr(0)
      val b : BigInt = arr(1)
      bw.write((a+b.pow(2)).toString + "\n")
    }
    bw.close()
  }
}
```

Результаты:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.828	66043904	25	20
1	OK	0.734	65966080	7	4
2	OK	0.750	65933312	8	4
3	OK	0.796	65810432	5	2
4	OK	0.796	65859584	5	2
5	OK	0.750	65802240	6	2
6	OK	0.750	65982464	6	2
7	OK	0.734	65875968	23	20
8	OK	0.718	65961984	25	19
9	OK	0.812	66043904	24	19
10	OK	0.828	65933312	24	20
11	OK	0.656	65875968	23	19
12	OK	0.671	65916928	23	19
13	OK	0.734	65855488	20	16
14	OK	0.687	65921024	23	19
15	OK	0.718	65912832	20	19
16	OK	0.750	65835008	22	19
17	OK	0.687	65921024	23	19
18	OK	0.781	65974272	22	18
19	OK	0.734	65859584	22	18
20	OK	0.687	65982464	22	19

3

Сортировка вставками

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

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 n ($1 \le n \le 1000$) — число элементов в массиве. Во второй строке находятся n различных целых чисел, по модулю не превосходящих 10^9.

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

Листинг l3.scala:

```
import scala.io._
import java.io._
import scala.math.BigInt
import scala.Array
import scala.collection.Map

object l3 {
  /** Сортировка вставками */

  def sortIn(lst : List[Int]):List[Tuple2[Int, Int]] = {
      def _sortIn(unsorted : List[Int], tmp : List[Tuple2[Int, Int]], sorted :
List[Tuple2[Int, Int]]) : List[Tuple2[Int, Int]] = {
          if(unsorted != List()){
              /** unsorted head элемент для вставки */
              if(tmp == List()){
```

```
                    val newSorted : List[Tuple2[Int, Int]] = sorted :+ (unsorted
head, 1)
                    _sortIn(unsorted tail, newSorted, newSorted)
                }
                else if((unsorted head) < (tmp head)._1){
                    _sortIn(unsorted, tmp tail, sorted)
                }
                else {
                    /** Получаем сортированный список */
                    /** sorted 5 3 2 1 - tmp 3 2 1 + 4 :: tmp = sorted 5 4 3 2 1 */
                    val newSorted : List[Tuple2[Int, Int]] = (sorted diff tmp) :::
( (unsorted head, tmp.length + 1) :: tmp )
                    _sortIn(unsorted tail, newSorted, newSorted)
                }
            }
            else{
              sorted reverse
            }
        }
        _sortIn(lst tail, List((lst head, 1)), List((lst head, 1)))
    }
    def main(args: Array[String]): Unit = {
      val inpFname = "input.txt"
      val outFname = "output.txt"
      val outFile = new File(outFname)
      val bw = new BufferedWriter(new FileWriter(outFile))
      val lines  = Source.fromFile(inpFname).getLines.toArray
      val count = lines(0).toInt
      var listElements : List[Int]= lines(1).split(' ').map(_.toInt).toList
      val listWithIndex = sortIn(listElements)
      val elIndex = listWithIndex.toMap
      val outFirstLine = listElements.map(elIndex(_)) mkString " "
      val outSecondLine = listWithIndex.map(_._1) mkString " "
      bw.write(outFirstLine + "\n")
      bw.write(outSecondLine + "\n")
      bw.close()
    }
}
```

Результаты:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.562	93663232	10415	14296
1	OK	0.906	70840320	25	40
2	OK	0.796	68321280	7	5
3	OK	0.859	69537792	12	12
4	OK	0.859	68939776	8	8
5	OK	0.875	69545984	10	12
6	OK	0.812	69525504	29	31
7	OK	0.828	68911104	10	12
8	OK	0.843	69623808	10	12
9	OK	0.859	69554176	10	12
10	OK	0.812	69591040	10	12
11	OK	0.906	69697536	10	12
12	OK	0.828	70701056	57	63
13	OK	0.921	70733824	56	62
14	OK	0.875	69799936	57	63
15	OK	0.859	70746112	77	87
16	OK	0.859	70959104	76	86
17	OK	0.812	69799936	77	87
18	OK	0.890	70836224	112	127
19	OK	0.859	70889472	111	127
20	OK	0.828	69849088	110	125
21	OK	1.000	71847936	949	1190
22	OK	0.937	72081408	960	1219
23	OK	0.859	70238208	957	1134
24	OK	0.984	73084928	1490	1888
25	OK	1.046	73924608	1486	1944
26	OK	0.859	70320128	1481	1761
27	OK	1.187	81829888	3723	4888
28	OK	1.203	90714112	3729	5047
29	OK	1.000	72298496	3727	4437
30	OK	1.421	92930048	8456	11338
31	OK	1.359	92721152	8471	11609
32	OK	0.968	79466496	8415	10035
33	OK	1.562	92971008	10415	14035
34	OK	1.531	93663232	10410	14296
35	OK	1.046	83775488	10393	12386

4

Знакомство с жителями Сортлэнда

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

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 n ($3 \le n \le 9999$, n нечетно). Вторая строка содержит описание массива M, состоящее из n 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 10^6.

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

Листинг l4.scala:

```
import scala.io._
import java.io._
import scala.math.BigInt
import scala.Array

object l4 {
  def quickSort(lst : List[Tuple2[Double, Int]]) : List[Tuple2[Double, Int]] = {
      if(lst != List()){
          val x = lst.head
          val xs = lst.tail
          val left = xs.filter( _._1 < x._1 )
          val right = xs.filter( _._1 >= x._1 )
          quickSort(left) ++ List(x) ++ quickSort(right)
      }
      else{
         List()
      }
  }
```

```
  def main(args: Array[String]): Unit = {
    val inpFname = "input.txt"
    val outFname = "output.txt"
    val outFile = new File(outFname)
    val bw = new BufferedWriter(new FileWriter(outFile))
    val lines  = Source.fromFile(inpFname).getLines.toArray
    val count = lines(0).toInt
    var listElements = lines(1).split(' ').map(_.toDouble).toList.zip(1 to count)
    val listWithIndex = quickSort(listElements)
    val outFirstLine =
      listWithIndex(0)._2.toString + " " +
      listWithIndex((count / 2).toInt)._2.toString + " " +
      listWithIndex(count - 1)._2.toString
    bw.write(outFirstLine)
    bw.close()
  }
}
```

Результаты:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		1.093	83267584	98892	14
1	OK	0.843	68567040	30	5
2	OK	0.828	68599808	33	5
3	OK	0.828	68984832	1065	8
4	OK	0.890	69529600	3732	10
5	OK	0.906	71114752	14975	13
6	OK	0.937	71180288	14998	11
7	OK	0.890	73318400	28749	14
8	OK	0.984	74149888	34791	12
9	OK	0.937	74665984	38037	13
10	OK	0.968	74821632	38074	14
11	OK	0.953	75288576	39288	13
12	OK	1.000	76161024	48638	13
13	OK	0.984	76558336	50722	12
14	OK	0.984	76533760	52757	14
15	OK	1.015	77103104	58008	13
16	OK	1.031	78643200	66504	14
17	OK	1.046	79474688	71786	14
18	OK	1.046	79601664	72346	14
19	OK	1.000	79949824	73304	13
20	OK	1.046	79982592	76139	14
21	OK	1.031	81166336	83944	14
22	OK	1.062	81084416	85179	13
23	OK	1.031	81608704	86522	12
24	OK	1.093	82493440	89202	13
25	OK	1.031	83267584	98892	14

5

Секретарь Своп

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

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 n 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 n ($1 \le n \le 5000$) — число элементов в массиве. Во второй строке находятся n целых чисел, по модулю не превосходящих 10^9. 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 ($1 \le X, Y \le n$). 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.

Листинг l5.scala:

```
import scala.io._
import java.io._
import scala.Array

object l5 {
  def quickSort(inpArr : Array[Long], file : BufferedWriter): Array[Long] = {
      var buff = file
      def _quickSort(inpArr : Array[Long], first : Int, last : Int): Array[Long] =
{
          if(first >= last){
              /** return */
              inpArr
          }
          else {
              var i : Int = first
```

```
              var j : Int = last
              var arr = inpArr
              var tmpIndexes : List[Tuple2[Int, Int]]= List()
              /** Средний элемент */
              val center = arr(((first + last)/2).toInt)
              while( i <= j ){
                  while(arr(i) < center){ i += 1 }
                  while(arr(j) > center){ j -= 1 }
                  if( i <= j ){
                      if(i != j && arr(i) != arr(j)){
                        val tmp = arr(i)
                        arr(i) = arr(j)
                        arr(j) = tmp
                        tmpIndexes = (i, j)::tmpIndexes
                      }
                      i += 1
                      j -= 1
                  }

              }
               tmpIndexes.map(x => buff.write(s"Swap elements at indices ${x._1 +
1} and ${x._2 + 1}.\n"))
              _quickSort(arr, first, j)
              _quickSort(arr, i, last)
          }
      }
      _quickSort(inpArr, 0, inpArr.length - 1)
  }
  def main(args: Array[String]): Unit = {
    val inpFname = "input.txt"
    val outFname = "output.txt"
    val outFile = new File(outFname)
    val bw = new BufferedWriter(new FileWriter(outFile))
    val lines  = Source.fromFile(inpFname).getLines.toArray
    val count = lines(0).toInt
    var listElements = lines(1).split(' ').map(_.toLong).toArray
    val listWithIndex = quickSort(listElements, bw)
    val outText : String = "No more swaps needed.\n" + (listWithIndex mkString "
" )
    bw.write(outText)
    bw.close()
  }
}
```

Результаты:

№ теста	Результат	Время, с	Память	Размер входного файла	Размер выходного файла
Max		0.937	78180352	51993	620990
1	OK	0.843	68132864	14	133
2	OK	0.828	67293184	7	24
3	OK	0.796	67502080	12	29
4	OK	0.703	68050944	8	59
5	OK	0.796	67616768	10	27
6	OK	0.765	67473408	10	27
7	OK	0.750	67387392	29	46
8	OK	0.812	68173824	10	61
9	OK	0.812	68091904	10	61
10	OK	0.765	68018176	10	95
11	OK	0.781	68120576	10	61
12	OK	0.843	68128768	10	95
13	OK	0.750	68063232	50	135
14	OK	0.796	68120576	56	175
15	OK	0.812	67567616	57	74
16	OK	0.765	68055040	55	140
17	OK	0.796	68169728	75	296
18	OK	0.781	67510272	76	93
19	OK	0.796	68001792	78	197
20	OK	0.828	68075520	108	261
21	OK	0.781	67571712	107	123
22	OK	0.750	68050944	108	295
23	OK	0.796	68296704	948	5794
24	OK	0.859	67530752	947	963
25	OK	0.750	68157440	948	2575
26	OK	0.765	68784128	3720	31214
27	OK	0.796	67760128	3735	3750
28	OK	0.750	68337664	3722	10431
29	OK	0.828	69664768	8463	77117
30	OK	0.796	68059136	8441	8456
31	OK	0.796	68820992	8434	23769
32	OK	0.875	72523776	22822	246916
33	OK	0.843	68997120	22825	22839
34	OK	0.796	69918720	22877	65744
35	OK	0.937	78180352	51987	620990
36	OK	0.781	70311936	51940	51954
37	OK	0.937	72269824	51993	150900

Вывод:

Эта работа была достаточно полезной для меня, и в каком-то плане новой, потому что раньше я никогда не работал с таким вот тестовым окружением, потому что обычно это часто встречается на олимпиадах, а я на них не ходил. Также я сначала начал писать задания на python, потом подумал, посмотрел, что есть в списке языков Scala и начал писать на ней, потому что это просто интереснее, хотя кода больше и делать все это сложнее, но у меня все получилось. Отдельно стоит отметить 5 задание, потому что я достаточно долго над ним сидел и у меня ничего не получалось, хотя я использовал быструю сортировку, как сейчас мне кажется, у меня не получалось из-за того, что я в своей функции quickSort собирал большую структуру типа Tuple2[Array[Long], List[Tuple2[Int, Int]]], и потом ее возвращал, не знаю почему, но когда я начал писать в буфер сразу из функции, без вот этих вот лишних затрад на пересылку из запись во вспомогательные структуры данных, то тест прошел успешно, чему я был несказанно рад.