Санкт-Петербургский государственный университет
информационных технологий, механики и оптики

Кафедра компьютерных технологий

А. А. Владыкин

Разработка визуализатора алгоритма поиска порядковых статистик и квантилей на основе технологии *Vizi*

Программирование с явным выделением состояний
Проектная документация

Проект создан в рамках
«Движения за открытую проектную документацию»
http://is.ifmo.ru/

Санкт-Петербург
2004

Содержание

Введение

Понятия *порядковая статистика* и *квантиль* связаны с задачей о выборе элемента с данным номером (selection problem, [1]). Эта задача формулируется следующим образом: в множестве из n чисел найти элемент, который будет k-ым по счету, если расположить элементы множества в порядке возрастания. Такой элемент называют k-ой порядковой статистикой (order statistic). Например, порядковая статистика номер 0 — это минимум, порядковая статистика номер $n-1$ — максимум.

k-квантилями (k-th quantiles) множества из n целых чисел называют $k-1$ его элементов, обладающих следующим свойством: если расположить элементы множества в порядке возрастания, то квантили будут разбивать множество на k равных (точнее, отличающихся не более чем на один элемент) частей. Более точно можно определить k-квантили как порядковые статистики с номерами $\lfloor n/k \rfloor$, $\lfloor 2n/k \rfloor$, $\ldots$, $\lfloor (k-1)n/k \rfloor$.

Простейшее решение задачи поиска k-ой порядковой статистики (k-квантилей) состоит в сортировке исходного множества и выборе из него интересующего элемента (элементов). При использовании лучших из известных алгоритмов сортировки это потребует времени $O(n \log_2 n)$. Можно, однако, пойти другим путем.

В настоящей работе приведена реализация алгоритма поиска k-ой порядковой статистики, предложенного Ч. Хоаром и работающего за время $O(n)$ в среднем (имеется в виду математическое ожидание времени его работы на любом входе). Иллюстрируется механизм поиска k-квантилей. Реализация выполнена при помощи технологии *Vizi* и представляет собой Java-апплет, пошагово визуализирующий работу алгоритма на разных входных данных.

Документация включает описание алгоритма и его возможных модификаций, а также тексты программ.

1. Анализ литературы

Информация об алгоритме поиска порядковых статистик взята из книг [1] и [2]. Работа [3], к сожалению, содержит лишь общие теоретические оценки числа сравнений при решении этой задачи.

Терминология в этих книгах авторами (и/или переводчиками) не согласована, поэтому наряду с «k-ой порядковой статистикой» можно встретить «k-ый в порядке возрастания (убывания) элемент», «k-ый наибольший (наименьший) элемент», «k-ый максимум».

Алгоритм поиска порядковых статистик тесно связан с быстрой сортировкой, так как использует ту же идею разделения массива. В литературе его обсуждение традиционно следует за алгоритмом быстрой сортировки и обычно носит вспомогательный характер. Для того, чтобы собрать всю информацию о рассматриваемом алгоритме воедино, приходится обращаться к разным главам используемой книги. Особенно это характерно для

работы [1].

Книга [2] дает довольно краткое, но достаточное для понимания описание алгоритма, формальное доказательство корректности которого приведено в работе Ч. Хоара [4].

С квантилями ситуация несколько сложнее. Этот термин обычно встречается в контексте математической статистики и имеет значение корня X_Q уравнения $F(X) = Q$, где F — функция распределения случайной величины, а $0 \leq Q \leq 1$ — некоторая вероятность (порядок квантиля).

Требуемое для настоящей работы определение квантилей приведено только в [1], где этому вопросу посвящен ровно один абзац. Кроме определения читателям предложено самостоятельно придумать алгоритм поиска квантилей. И все.

Настоящая работа предлагает обобщенный и переработанный материал из названных выше источников, причем изложение проведено независимо от быстрой сортировки. Отдельно обсуждается поиск k-квантилей.

2. Описание алгоритма

2.1. Поиск k-ой порядковой статистики

В соответствии с данным во введении определением, k-ая порядковая статистика множества из n чисел является элементом, который будет k-ым по счету, если расположить элементы множества в порядке возрастания.

Выберем представление данных в программе.

Пусть множество задано массивом $A[0\,..\,n-1]$ с произвольным порядком элементов. Такой массив, в общем случае, представляет мультимножество, так как может содержать повторящиеся элементы. Однако на корректность алгоритма это не влияет.

Кроме того, задано число k — номер искомой порядковой статистики ($k \in 0\,..\,n-1$).

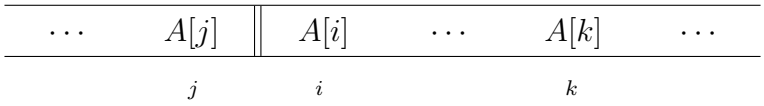
Перейдем к описанию алгоритма, который сводится к многократному применению следующей процедуры разделения массива.

1. Наугад выбрать некоторый элемент $A[s]$ и записать его значение в переменную x.
2. Просматривая массив слева направо, найти элемент $A[i] : A[i] \geq x$.
3. Просматривая массив справа налево, найти элемент $A[j] : x \geq A[j]$.
4. Если $i \leq j$, обменять найденные элементы $A[i]$ и $A[j]$ между собой, на единицу увеличить i и уменьшить j. После этого вернуться к шагу (2). Иначе — выход.

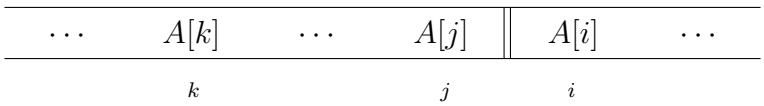
В этой процедуре начало массива — $A[0\,..\,i-1]$ — накапливает элементы, не превосходящие x. В конце — в $A[j+1\,..\,n-1]$ — оказываются элементы, большие или равные x. Процесс завершается, когда просмотры встречаются где-то в середине массива ($i > j$) и «накопители» покрывают массив целиком. Можно убедиться, что после завершения процедуры $i = j + 1$ или $i = j + 2$ — накопители не пересекаются или перекаются в одном элементе.

Проанализируем варианты расположения «места встречи» накопителей относительно k-ой ячейки массива.

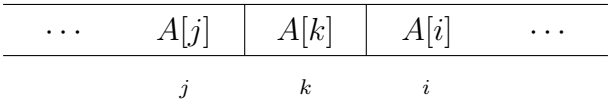
1. $j < i \le k$. Ни один элемент из левой части не сможет оказаться искомой k-ой порядковой статистикой. Дальше можно рассматривать только правую часть — $A[i\,..\,n-1]$, и повторить для нее процесс разделения.



2. $k \le j < i$. k-ая порядковая статистика находится среди элементов левой части: $A[0\,..\,j]$. Правую можно исключить из рассмотрения, а для левой провести новое разделение.



3. $j < k < i$. k-ая порядковая статистика оказалась на пересечении накопителей. Алгоритм поиска завершается и возвращает $A[k]$.



Процессы разделения повторяются до тех пор, пока не возникнет третий случай. Это рано или поздно случится — после каждого разделения отбрасывается хотя бы один элемент, а после исключения последнего элемента как раз выполняются условия третьего случая.

Кстати, именно ситуация, когда каждое разделение исключает один элемент, является для алгоритма самой неблагоприятной. При этом его трудоемкость составляет

$$n + (n-1) + \cdots + 2 + 1 = \frac{n(n+1)}{2} = O(n^2),$$

что даже хуже результата, достигаемого при использовании простейшего алгоритма, сортирующего массив за $O(n \log_2 n)$. Однако если предположить, что после каждого разделения в среднем массив укорачивается в $q > 1$ раз, то оценка получается гораздо более привлекательной:

$$n + \frac{n}{q} + \frac{n}{q^2} + \cdots + 1 \le \frac{nq}{q-1} = O(n).$$

Результат процедуры разделения (следовательно, и временна́я сложность всего алгоритма) определяется выбором граничного элемента $A[s]$. Неблагоприятным будет выбор в качестве граничного элемента $A[s]$ минимума или максимума. В этом случае массив сократится на один элемент. Удачный выбор — медиана, так как при этом разделение

осуществится пополам. Однако дополнительные временны́е затраты на выбор лучшего граничного элемента могут лишить алгоритм исходной простоты и скорости.

Литература содержит разные подходы к решению этой проблемы. В работе [1] предлагается $s = \mathrm{random}(p, r)$, где p и r — границы рассматриваемой на текущем шаге части массива, а $\mathrm{random}(a, b)$ — случайное целое число из диапазона $a\,..\,b$. В книге [2] используется $s = k$ — элемент, занимающий место искомой порядковой статистики.

2.2. Поиск k-квантилей

Выберем представление данных в программе.

Множество чисел, как и ранее, задано массивом $A[0\,..\,n-1]$. Число $k \in 2\,..\,n+1$ задает порядок квантилей.

Перейдем к описанию алгоритма. На основе определения k-квантилей (как порядковых статистик с номерами $\lfloor n/k \rfloor$, $\lfloor 2n/k \rfloor$, ..., $\lfloor (k-1)n/k \rfloor$) и алгоритма поиска статистик, можно предложить простой способ нахождения квантилей. Он состоит в последовательном поиске каждой порядковой статистики в исходном массиве A. Средняя трудоемкость такого алгоритма $(k-1)O(n) = O(nk)$.

Есть возможность оптимизировать поиск. Обратим внимание на то, что, при поиске первого квантиля ($\lfloor n/k \rfloor$-ой порядоковой статистики), попутно осуществляется разделение массива на части $A[0\,..\,\lfloor n/k \rfloor]$ и $A[\lfloor n/k \rfloor + 1\,..\,n-1]$. Элементы из первой части не смогут оказаться следующими квантилями. Поэтому в дальнейшем достаточно рассматривать только вторую часть массива. Аналогично, найдя второй квантиль, поиск третьего можно вести в $A[\lfloor 2n/k \rfloor + 1\,..\,n-1]$ и т. д. К сожалению, асимптотическая оценка трудоемкости модифицированного алгоритма остается $O(nk)$.

Для достижения лучшей асимптотической оценки необходимо изменить порядок поиска квантилей. Выберем из них ближайший к середине массива и запустим поиск соответствующей порядковой статистики. Массив окажется разделен на две примерно одинаковые части, которые в дальнейшем можно рассматривать независимо. Повторим это отдельно для первой и второй частей. И так до тех пор, пока не будут найдены все квантили. Алгоритм оказался рекурсивным и стал сильно напоминать быструю сортировку. Его средняя трудоемкость оценивается как

$$n + 2\frac{n}{2} + 4\frac{n}{4} + \cdots + k\frac{n}{k} = O(n \log_2 k).$$

3. Реализация визуализируемых алгоритмов

Исходный код на языке Java приведен в приложении 1.

В данной реализации при поиске k-ой порядковой статистики граничным выбирается элемент $A[k]$. Поиск квантилей ведется последовательно, слева направо, с отбрасыванием элементов массива, расположенных левее уже найденных.

Обратим внимание на то, что алгоритмы работают непосредственно с переданным им массивом чисел, изменяя его. Если первоначальное расположение элементов представляет интерес, то необходимо сделать копию исходного массива.

4. Описание модели данных

Технология *Vizi* предусматривает вынесение всех используемых алгоритмом переменных в отдельный класс, называемый моделью данных.

4.1. Переменные, доступные алгоритмам

`OrderStatisticsVisualizer visualizer` — экземпляр апплета.

`int[] a` — массив чисел для поиска.

`int algorithm` — идентификатор выбранного пользователем алгоритма.

`int k` — порядок искомых медиан/квантилей.

4.2. Переменные алгоритма OrderStatistic

`int p` — левая граница области поиска.

`int r` — правая граница области поиска.

`int k` — номер искомой порядковой статистики.

`int x` — граничный элемент при разделении.

`int i` — индекс элемента при просмотре слева.

`int j` — индекс элемента при просмотре справа.

`int t` — временная переменная для обмена.

4.3. Переменные алгоритма Quantiles

`int k` — порядок квантилей.

`int[] qn` — номера искомых порядковых статистик.

`int i` — переменная цикла.

5. Преобразование программы

Приложение 2 содержит текст программы, подготовленный к преобразованию к автоматному виду. Вся логика сведена к простейшим конструкциям, поддерживаемым *Vizi*: присваивание, условный оператор, цикл с предусловием, вызов автомата. Цикл с постусловием `do ... while` функции `OrderStatistic` заменен на цикл `while` с тем же предусловием, хотя в общем случае они не эквивалентны.

Изменена функция `Quantiles`: теперь она вначале вычисляет массив номеров порядковых статистик `int[] qn`. Это сделано для того, чтобы при последующей визуализации алгоритма информация о номерах искомых статистик использовалась без многократного пересчета.

На основе преобразованной программы создано XML-описание алгоритмов, приведенное в приложении 3. По нему автоматически сгенерированы исходные коды автоматов (см. приложение 4).

Всего автоматов шесть — три пары, каждая из которых содержит автоматы прямого и обратного хода алгоритма. Эти пары обозначены следующим образом: `Main`, `OrderStatistic` и `Quantiles`. Автомат `Main` получает управление и, в зависимости от переданных ему параметров (ввода пользователя), инициализирует и запускает автомат `OrderStatistic` или `Quantiles`. Последние два соответствуют одноименным алгоритмам.

Количество состояний в каждом из автоматов приведено в следующей таблице.

`Main`	8+8
`OrderStatistic`	27+27
`Quantiles`	7+7

Сравнительно большое количество состояний в автоматах связано с подробной визуализацией алгоритмов.

6. Описание интерфейса визуализатора

Внешний вид визуализатора представлен на рис. 1.

Верхняя часть окна демонстрирует состояние массива на текущем шаге алгоритма, а также некоторую дополнительную информацию. В процессе разделения в левом верхнем углу присутствует значение граничного элемента x. Кроме того, под текущими просматриваемыми ячейками отображаются индексы i и j. Стрелка между элементами означает обмен их местами.

Ниже расположена область, где отображаются пояснения к тому, что делает алгоритм, и элементы управления.

В любой момент пользователь полностью контролирует ход событий. При помощи кнопок << и >> в левой части можно переходить к следующему шагу алгоритма или возвращаться к предыдущему. Кнопка `Рестарт` позволяет начать все с самого начала. Кнопка `Авто` запускает визуализацию в автоматическом режиме. Пауза между шагами выставляется элементом управления `Задержка`. Нажатие на кнопку ? открывает окно с информацией об авторе визуализатора.

Выбор визуализируемого алгоритма (поиск порядковой статистики или поиск квантилей) осуществляется активизацией соответствующего пункта выпадающего меню. Рядом

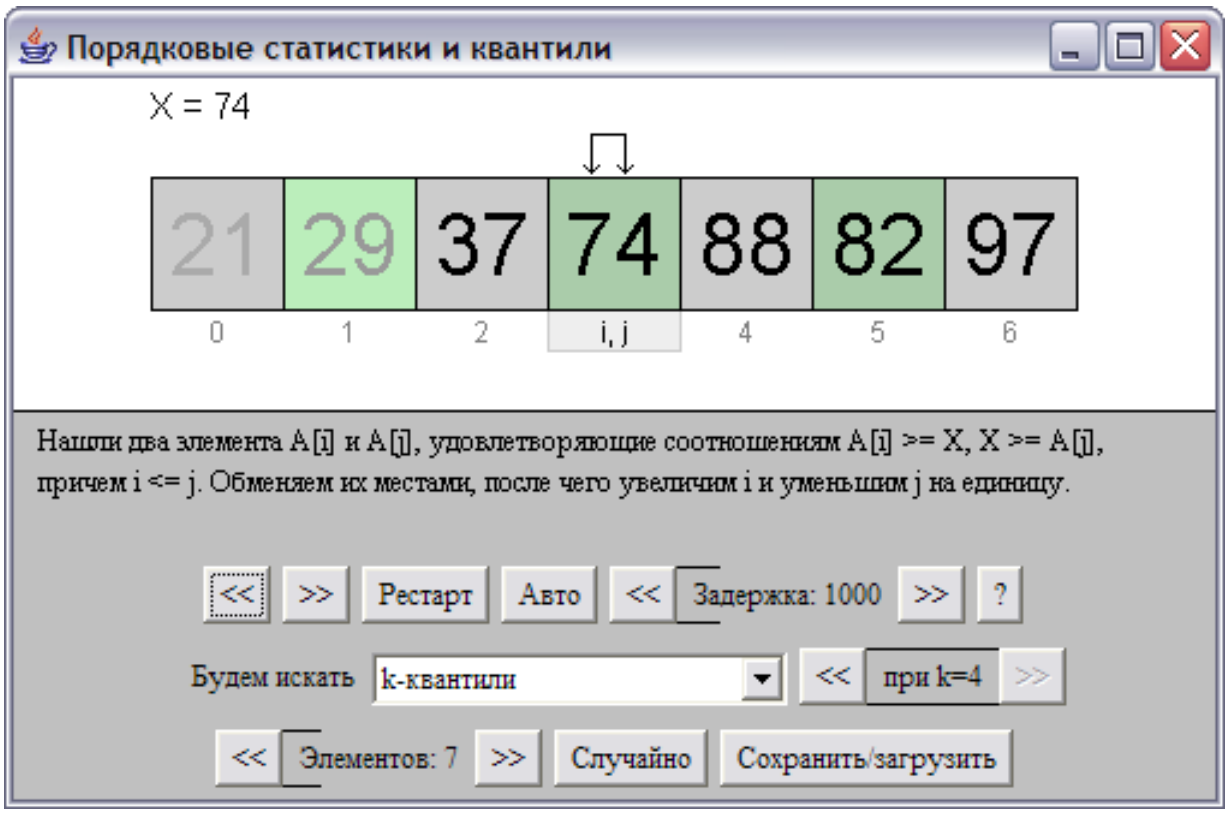


Рис. 1. Внешний вид визуализатора

с помощью соответствующих кнопок можно указать желаемое k, которое, в зависимости от выбранного алгоритма, имеет смысл номера порядковой статистики или порядка квантилей.

При помощи кнопок << и >> в самой нижней строке визуализатора можно можно устанавливать число элементов массива. Кнопка `Случайно` заполняет массив случайными числами. Кнопка `Сохранить/Загрузить` вызывает диалоговое окно сохранения и загрузки состояния визуализатора.

7. Описание конфигурации визуализатора

Параметры визуализатора хранятся в файле `OrderStatistics-Configuration.xml`. Благодаря этому можно без перекомпиляции апплета изменять цветовую гамму, вид элементов управления, а также некоторые другие настройки.

Таблица, приведенная ниже, содержит описание нестандартных параметров визуализатора.

Параметр	Описание
`algorithm`	Параметры, связанные с выбором алгоритма.
`-prompt`	Текст, выводимый перед меню с доступными алгоритмами.
`-orderStatistic`	Пункт, соответствующий поиску порядковой статистики.

-quantiles	Пункт, соответствующий поиску квантилей.
-k	Панель выбора параметра k.
elements	Панель установки размера массива.
button-random	Кнопка, заполняющая массив случайными числами.
button-saveLoad	Кнопка, открывающая диалог сохранения/загрузки.
SaveLoadDialog	Параметры, связанные с диалогом сохранения/загрузки.
-arrayComment	Комментарий к массиву чисел.
-algorithmComment	Комментарий к имени алгоритма.
-kComment	Комментарий к параметру k.
-stepComment	Комментарий к номеру шага.
-arraySizeError	Сообщение о неверном размере массива.
-algorithmNameError	Сообщение о неверном имени алгоритма.
cellStyles	Стили ячеек массива.
-style0	Нормальная ячейка.
-style1	Подсвеченная ячейка.
-style2	Ярко подсвеченная ячейка.
-style3	Нормальная ячейка вне области поиска.
-style4	Подсвеченная ячейка вне области поиска.
-style5	Ярко подсвеченная ячейка вне области поиска.
indexStyles	Стили индексов ячеек.
-style0	Обычный индекс.
-style1	Выделенный индекс.
arrowStyles-style0	Стиль стрелки, возникающей при обмене элементов.
watchStyles-style0	Стиль области, отображающей значение x.
watchMsg	Формат отображения переменной x.
minIndexHeight	Минимальная высота индекса ячейки, в пикселах.
maxElementValue	Максимальное значение элемента массива.

Полностью файл `OrderStatistics-Configuration.xml` приведен в приложении 3.

Заключение

Поставленные задачи — сбор и систематизация сведений об алгоритмах поиска порядковых статистик и квантилей, а также создание визуализатора — решены. Написание документации позволило вывести проект на новый уровень, способствовало исследованию алгоритма и принятию рациональных проектных решений. В процессе документирования автору открывались все новые и новые детали его же собственной программы.

Технология *Vizi* оказалась весьма удобной для создания визуализатора. Автоматическая генерация обратных шагов освобождает разработчика от многих забот и дает возмож-

ность реализовывать сложные алгоритмы. Встроенный в *Vizi* механизм интернационализации позволяет при минимальных усилиях получать версии визуализатора на разных языках (русский и английский).

Хочется выразить несколько пожеланий по дальнейшему развитию *Vizi*.

- В текущей версии *Vizi* — 0.4b5 — для рекурсивного запуска автоматов приходится создавать дополнительный стек и вручную производить с ним запутанные манипуляции, особенно если это рекурсивный запуск того же самого автомата. Возможно, в рамках *Vizi* стоит частично или полностью автоматизировать этот процесс.
- Иногда возникает необходимость заводить внутри XML-описания дополнительные переменные, хранящие информацию, связанную с визуализацией алгоритма. Кажется более предпочтительным вынести их, однако идеология *Vizi* не позволяет это сделать.
- Для полноценного использования технологии *Vizi* не хватает подробной документации, описывающей общие принципы построения визуализаторов и снабженной развернутыми примерами.

Список литературы

[1] *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. М.: МЦНМО, 2002. — 960 с.

[2] *Вирт Н.* Алгоритмы и структуры данных. СПб.: Невский Диалект, 2001. — 352 с.

[3] *Кнут Д. Э.* Искусство программирования, том 3. Сортировка и поиск. М.: Вильямс, 2004. — 832 с.

[4] *Hoare C. A. R.* Proof of a program: FIND //Communications of the ACM, Volume 14, Issue 1, pp. 39–45.

Приложение 1. Исходный текст реализации алгоритма на языке Java

```
public final class OrderStatistics {
public static int OrderStatistic(int a[], int p, int r, int k) {
    while (p < r) {
        int x = a[k];
        int i = p;
        int j = r;
        do {
            while (a[i] < x)
                i++;
            while (x < a[j])
                j--;
            if (i <= j) {
                int t = a[i];
                a[i] = a[j];
                a[j] = t;
                i++;
                j--;
            }
        } while (i <= j);
        if (j < k)
            p = i;
        if (k < i)
            r = j;
    }
    return a[k];
}
public static int[] Quantiles(int a[], int k) {
    int prev = -1;
    int q[] = new int[k - 1];
    for (int i = 1; i < k; i++) {
        int curr = a.length * i / k;
        q[i - 1] = OrderStatistic(a, prev + 1, a.length - 1, curr);
        prev = curr;
    }
    return q;
}
}
```

Приложение 2. Исходный текст преобразованной реализации алгоритма

```
public final class OrderStatisticsA {
public static int OrderStatistic(int[] a, int p, int r, int k) {
    while (p < r) {
        int x = a[k];
        int i = p;
        int j = r;
        int t;
```

```
            while (i <= j) {
                while (x < a[j])
                    j--;
                while (a[i] < x)
                    i++;
                if (i <= j) {
                    t = a[i];
                    a[i] = a[j];
                    a[j] = t;
                    i++;
                    j--;
                }
            }
            if (j < k)
                p = i;
            if (k < i)
                r = j;
        }
        return a[k];
    }
    public static int[] Quantiles(int[] a, int k) {
        int i;
        int[] qn = new int[k];
        qn[0] = -1;
        i = 1;
        while (i < k) {
            qn[i] = i * a.length / k;
            i++;
        }
        int[] q = new int[k - 1];
        i = 1;
        while (i < k) {
            q[i - 1] = OrderStatistic(a, qn[i - 1] + 1, a.length - 1,
               qn[i]);
            i++;
        }
        return q;
    }
}
```

Приложение 3. XML-описание визуализатора

Описание разбито на три файла:

- `OrderStatistics.xml` — содержит общую информацию о визуализаторе и ссылки на два следующих файла;
- `OrderStatistics-Algorithm.xml` — описывает структуру автоматов, реализующих алгоритм;
- `OrderStatistics-Configuration.xml` — содержит параметры визуализатора.

OrderStatistics.xml

```
<?xml version="1.0" encoding="WINDOWS-1251"?>

<!DOCTYPE visualizer PUBLIC
    "-//IFMO Vizi//Visualizer description"
    "http://ips.ifmo.ru/vizi/dtd/visualizer.dtd"
[
    <!ENTITY algorithm SYSTEM "OrderStatistics-Algorithm.xml">
    <!ENTITY configuration SYSTEM "OrderStatistics-Configuration.xml">
]>

<visualizer
    id="OrderStatistics"
    package="ru.ifmo.vizi.orderstatistics"
    main-class="OrderStatisticsVisualizer"

    preferred-width="500"
    preferred-height="300"

    name-ru="Порядковые статистики и квантили"
    name-en="Order Statistics &amp; Quantiles"

    author-ru="Алексей Владыкин"
    author-en="Aleksey Vladykin"
    author-email="vladykin@rain.ifmo.ru"

    supervisor-ru="Георгий Корнеев"
    supervisor-en="Georgiy Korneev"
    supervisor-email="kgeorgiy@rain.ifmo.ru"

    copyright-ru="Copyright \u00A9 Кафедра КТ, СПб ГИТМО (ТУ), 2004"
    copyright-en="Copyright \u00A9 Computer Technologies Department, SPb IFMO, 2004"
>
    &algorithm;
    &configuration;
</visualizer>
```

OrderStatistics-Algorithm.xml

```
<?xml version="1.0" encoding="WINDOWS-1251"?>

<algorithm>
    <variable
        description = "Экземпляр апплета"
        type        = "OrderStatisticsVisualizer"
        name        = "visualizer"
        value       = "null"
    />
    <variable
        description = "Алгоритм поиска порядковой статистики"
        type        = "static final int"
        name        = "ORDERSTATISTIC"
        value       = "1"
    />
    <variable
        description = "Алгоритм поиска квантилей"
        type        = "static final int"
        name        = "QUANTILES"
        value       = "2"
    />
    <variable
        description = "Массив чисел для поиска"
        type        = "int[]"
        name        = "a"
        value       = "new int[] {4,5,2,1,3}"
    />
    <variable
        description = "Алгоритм"
        type        = "int"
        name        = "algorithm"
        value       = "ORDERSTATISTIC"
    />
    <variable
```

```
    description = "Порядок искомых медиан/квантилей"
    type        = "int"
    name        = "k"
    value       = "2"
/>

<toString>
    StringBuffer buf = new StringBuffer();
    buf.append("k=").append(@k).append(";");
    buf.append("Quantiles_i=").append(@Quantiles@i).append(";");
    buf.append("OrderStatistic_p=").append(@OrderStatistic@p).append(";");
    buf.append("OrderStatistic_r=").append(@OrderStatistic@r).append(";");
    buf.append("OrderStatistic_x=").append(@OrderStatistic@x).append(";");
    buf.append("OrderStatistic_i=").append(@OrderStatistic@i).append(";");
    buf.append("OrderStatistic_j=").append(@OrderStatistic@j).append(";");
    return buf.toString();
</toString>


<auto
    id          = "Main"
    description = "Главный автомат"
>
    <start
        comment-ru  = "Готов к работе."
        comment-en  = "Ready."
    >
        <draw>
            @visualizer.syncArray();
            @visualizer.resetCellStyles();
            if (@algorithm == @ORDERSTATISTIC) {
                @visualizer.setCellStyles(new int[] {@k}, 2);
            } else if (@algorithm == @QUANTILES) {
                int q[] = new int[@k - 1];
                for (int j = 1; j &lt; @k; j++)
                    q[j - 1] = @a.length * j / @k;
                @visualizer.setCellStyles(q, 2);
            }
            @visualizer.resetIndices();
            @visualizer.updateExchangeArrow(-1, -1);
            @visualizer.updateWatch(-1);
        </draw>
    </start>
    <if
        id          = "Launch"
        description = "Установка параметров и запуск нужного автомата"
        test        = "@algorithm == @ORDERSTATISTIC"
        level       = "-1"
    >
        <then>
            <step
                id          = "PreOrderStatisticCall"
                description = "Подготовка к вызову OrderStatistic"
                level       = "-1"
            >
                <action>
                    @OrderStatistic@p @= 0;
                    @OrderStatistic@r @= @a.length - 1;
                    @OrderStatistic@k @= @k;
                </action>
            </step>
            <call-auto id="OrderStatistic"/>
        </then>
        <else>
            <step
                id          = "PreQuantilesCall"
                description = "Подготовка к вызову Quantiles"
                level       = "-1"
            >
                <direct>
                    @Quantiles@k = @k;
                </direct>
            </step>
            <call-auto id="Quantiles"/>
        </else>
    </if>
```

```
    <finish
        comment-ru  = "Алгоритм завершен."
        comment-en  = "Algorithm has finished."
    >
        <draw>
            @visualizer.syncArray();
            @visualizer.resetCellStyles();
            if (@algorithm == @ORDERSTATISTIC)
                @visualizer.setCellStyles(new int[] {@k}, 2);
            else if (@algorithm == @QUANTILES)
                @visualizer.setCellStyles(@Quantiles@qn, 2);
            @visualizer.resetIndices();
            @visualizer.updateExchangeArrow(-1, -1);
            @visualizer.updateWatch(-1);
        </draw>
    </finish>
</auto>


<auto
    id          = "OrderStatistic"
    description = "Поиск k-й порядковой статистики в массиве a, область поиска - p..r"
>
    <variable
        description = "Левая граница области поиска"
        type        = "int"
        name        = "p"
    />
    <variable
        description = "Правая граница области поиска"
        type        = "int"
        name        = "r"
    />
    <variable
        description = "Номер искомой порядковой статистики"
        type        = "int"
        name        = "k"
    />
    <variable
        description = "Граничный элемент при разделении"
        type        = "int"
        name        = "x"
    />
    <variable
        description = "Индекс элемента при просмотре слева"
        type        = "int"
        name        = "i"
    />
    <variable
        description = "Индекс элемента при просмотре справа"
        type        = "int"
        name        = "j"
    />
    <variable
        description = "Временная переменная для обмена"
        type        = "int"
        name        = "t"
    />
    <variable
        description = "Признак окончания основного цикла"
        type        = "boolean"
        name        = "br"
    />

    <step
        id          = "EmptyStep"
        description = "У Vizi проблемы, если в начале автомата стоит while"
        level       = "-1"
    >
        <direct/>
    </step>
    <while
        id              = "MainLoop"
        description     = "Главный цикл алгоритма"
        test            = "@p &lt; @r"
        level           = "-1"
```

```
>
    <step
        id          = "Introduction"
        description = "Введение"
        comment-ru  = "Ищем {0}-ю порядковую статистику. Это элемент, занимающий {0}-ю
             ячейку в отсортированном по возрастанию массиве. Область поиска:
            A[{1}..{2}]."
        comment-en  = "Searching for the {0}-th order statistic. It is the element,
            occupying the {0}-th cell in sorted array. The search area is
            A[{1}..{2}]."
        comment-args= "new Integer(@k), new Integer(@p), new Integer(@r)"
        level       = "1"
    >
        <draw>
            @visualizer.syncArray();
            @visualizer.resetCellStyles();
            if (@algorithm == @QUANTILES) {
                @visualizer.setCellStyles(@Quantiles@qn, 1);
                @visualizer.setCellStyles(@Quantiles@found, 2);
            }
            @visualizer.dimCells(0, @p - 1);
            @visualizer.dimCells(@r + 1, @a.length - 1);
            @visualizer.setCellStyles(new int[] {@k}, 2);
            @visualizer.resetIndices();
            @visualizer.updateExchangeArrow(-1, -1);
            @visualizer.updateWatch(-1);
        </draw>
        <direct/>
    </step>
    <step
        id          = "Barrier"
        description = "Выбор барьера"
        comment-ru  = "Запомним элемент A[{0}] = {1}, занимающий место искомой
            порядковой статистики, и обозначим его как X."
        comment-en  = "Let us remember the element A[{0}] = {1}, occupying the place
            of the order statistic, and denote its value as X."
        comment-args= "new Integer(@k), new Integer(@x)"
        level       = "1"
    >
        <draw>
            @visualizer.syncArray();
            @visualizer.resetCellStyles();
            if (@algorithm == @QUANTILES) {
                @visualizer.setCellStyles(@Quantiles@qn, 1);
                @visualizer.setCellStyles(@Quantiles@found, 2);
            }
            @visualizer.dimCells(0, @p - 1);
            @visualizer.dimCells(@r + 1, @a.length - 1);
            @visualizer.setCellStyles(new int[] {@k}, 2);
            @visualizer.resetIndices();
            @visualizer.updateExchangeArrow(-1, -1);
            @visualizer.updateWatch(@x);
        </draw>
        <action>
            @x @= @a[@k];
            @i @= @p;
            @j @= @r;
        </action>
    </step>
    <while
        id          = "Partition"
        description = "Разбиение массива"
        test        = "@i &lt;= @j"
        level       = "-1"
    >
        <step
            id          = "LoopIIntro"
            description = "Введение в LoopI"
            comment-ru  = "Просматривая массив слева направо, будем искать элемент
                A[i] &gt;= X."
            comment-en  = "Scanning the array from left to right we will look for an
                element A[i] &gt;= X."
            level       = "1"
        >
            <draw>
                @visualizer.syncArray();
```

```
            @visualizer.resetCellStyles();
            if (@algorithm == @QUANTILES) {
                @visualizer.setCellStyles(@Quantiles@qn, 1);
                @visualizer.setCellStyles(@Quantiles@found, 2);
            }
            @visualizer.dimCells(0, @p - 1);
            @visualizer.dimCells(@r + 1, @a.length - 1);
            @visualizer.setCellStyles(new int[] {@k}, 1);
            @visualizer.resetIndices();
            @visualizer.updateIndex(@i, "i");
            @visualizer.updateIndex(@j, "j");
            @visualizer.updateExchangeArrow(-1, -1);
            @visualizer.updateWatch(@x);
        </draw>
        <direct/>
    </step>
    <while
        id          = "LoopI"
        description = "Просмотр слева"
        test        = "@a[@i] &lt; @x"
        level       = "-1"
    >
        <step
            id          = "PreLoopIStep"
            description = "Объясняем действия"
            comment-ru  = "a[i] &lt; X, двигаемся дальше."
            comment-en  = "a[i] &lt; X, proceeding."
        >
            <draw>
                @visualizer.syncArray();
                @visualizer.resetCellStyles();
                if (@algorithm == @QUANTILES) {
                    @visualizer.setCellStyles(@Quantiles@qn, 1);
                    @visualizer.setCellStyles(@Quantiles@found, 2);
                }
                @visualizer.dimCells(0, @p - 1);
                @visualizer.dimCells(@r + 1, @a.length - 1);
                @visualizer.setCellStyles(new int[] {@k}, 1);
                @visualizer.resetIndices();
                @visualizer.updateIndex(@i, "i");
                @visualizer.updateIndex(@j, "j");
                @visualizer.updateExchangeArrow(-1, -1);
                @visualizer.updateWatch(@x);
            </draw>
            <direct/>
        </step>
        <step
            id          = "LoopIStep"
            description = "Сдвигаемся на один элемент вправо"
            level       = "-1"
        >
            <action>
                @i @= @i + 1;
            </action>
        </step>
    </while>
    <step
        id          = "LoopJIntro"
        description = "Введение в LoopJ"
        comment-ru  = "Нашли элемент A[i] &gt;= X. Теперь, просматривая массив
            справа налево, будем искать элемент A[j] &lt;= X."
        comment-en  = "We have found the element A[i] &gt;= X. Now, scanning the
            array from right to left we will look for an element A[j] &lt;= X."
        level       = "1"
    >
        <draw>
            @visualizer.syncArray();
            @visualizer.resetCellStyles();
            if (@algorithm == @QUANTILES) {
                @visualizer.setCellStyles(@Quantiles@qn, 1);
                @visualizer.setCellStyles(@Quantiles@found, 2);
            }
            @visualizer.dimCells(0, @p - 1);
            @visualizer.dimCells(@r + 1, @a.length - 1);
            @visualizer.setCellStyles(new int[] {@k}, 1);
            @visualizer.resetIndices();
```

```
            @visualizer.updateIndex(@i, "i");
            @visualizer.updateIndex(@j, "j");
            @visualizer.updateExchangeArrow(-1, -1);
            @visualizer.updateWatch(@x);
        </draw>
        <direct/>
    </step>
    <while
        id          = "LoopJ"
        description = "Просмотр справа"
        test        = "@x &lt; @a[@j]"
        level       = "-1"
    >
        <step
            id          = "PreLoopJStep"
            description = "Объясняем действия"
            comment-ru  = "X &lt; A[j], двигаемся дальше."
            comment-en  = "X &lt; A[j], proceeding."
            level       = "1"
        >
            <draw>
                @visualizer.syncArray();
                @visualizer.resetCellStyles();
                if (@algorithm == @QUANTILES) {
                    @visualizer.setCellStyles(@Quantiles@qn, 1);
                    @visualizer.setCellStyles(@Quantiles@found, 2);
                }
                @visualizer.dimCells(0, @p - 1);
                @visualizer.dimCells(@r + 1, @a.length - 1);
                @visualizer.setCellStyles(new int[] {@k}, 1);
                @visualizer.resetIndices();
                @visualizer.updateIndex(@i, "i");
                @visualizer.updateIndex(@j, "j");
                @visualizer.updateExchangeArrow(-1, -1);
                @visualizer.updateWatch(@x);
            </draw>
            <direct/>
        </step>
        <step
            id          = "LoopJStep"
            description = "Сдвигаемся на один элемент влево"
            level       = "-1"
        >
            <action>
                @j @= @j - 1;
            </action>
        </step>
    </while>
    <if
        id          = "ExchangeTest"
        description = "Меняем?"
        test        = "@i &lt;= @j"
        level       = "-1"
    >
        <then>
            <step
                id          = "PreExchange"
                description = "Объясняем действия"
                comment-ru  = "Нашли два элемента A[i] и A[j], удовлетворяющие
                    соотношениям A[i] &gt;= X, X &gt;= A[j], причем i &lt;= j.
                    Обменяем их местами, после чего увеличим i и уменьшим j на
                    единицу."
                comment-en  = "We have found such elements A[i] and A[j] that A[i]
                     &gt;= X, X &gt;= A[j] and i &lt;= j. Now we exchange them,
                    and finally we increase i and decrease j by one."
                level       = "1"
            >
                <draw>
                    @visualizer.syncArray();
                    @visualizer.resetCellStyles();
                    if (@algorithm == @QUANTILES) {
                        @visualizer.setCellStyles(@Quantiles@qn, 1);
                        @visualizer.setCellStyles(@Quantiles@found, 2);
                    }
                    @visualizer.dimCells(0, @p - 1);
                    @visualizer.dimCells(@r + 1, @a.length - 1);
```

```
                    @visualizer.setCellStyles(new int[] {@k}, 1);
                    @visualizer.resetIndices();
                    @visualizer.updateIndex(@i, "i");
                    @visualizer.updateIndex(@j, "j");
                    @visualizer.updateExchangeArrow(@i, @j);
                    @visualizer.updateWatch(@x);
                </draw>
                <direct/>
            </step>
            <step
                id          = "Exchange"
                description = "Обмен"
                level       = "-1"
            >
                <direct>
                    @t = @a[@i];
                    @a[@i] = @a[@j];
                    @a[@j] = @t;
                    @i++;
                    @j--;
                </direct>
                <reverse>
                    @i--;
                    @j++;
                    @t = @a[@i];
                    @a[@i] = @a[@j];
                    @a[@j] = @t;
                </reverse>
            </step>
        </then>
    </if>
</while>
<step
    id          = "PartitionComplete"
    description = "Разбиение завершено"
    comment-ru  = "Область поиска разбита на две части: A[{0}..i-1] и A[j+1..{1}].
         В первой все элементы не больше X, во второй - не меньше."
    comment-en  = "The search area is split into two parts: A[{0}..i-1] and
        A[j+1..{1}]. Every element of the first part is less or equal to X, every
        element of the second is greater or equal to X."
    comment-args= "new Integer(@p), new Integer(@r)"
    level       = "1"
>
    <draw>
        @visualizer.syncArray();
        @visualizer.resetCellStyles();
        if (@algorithm == @QUANTILES) {
            @visualizer.setCellStyles(@Quantiles@qn, 1);
            @visualizer.setCellStyles(@Quantiles@found, 2);
        }
        @visualizer.dimCells(0, @p - 1);
        @visualizer.dimCells(@r + 1, @a.length - 1);
        @visualizer.setCellStyles(new int[] {@k}, 1);
        @visualizer.resetIndices();
        @visualizer.updateIndex(@i - 1, "i-1");
        @visualizer.updateIndex(@i, "i");
        @visualizer.updateIndex(@j + 1, "j+1");
        @visualizer.updateIndex(@j, "j");
        @visualizer.updateExchangeArrow(-1, -1);
        @visualizer.updateWatch(@x);
    </draw>
    <action/>
</step>
<if
    id          = "LeftBoundTest"
    description = "Изменение левой границы"
    test        = "@j &lt; @k"
    level       = "-1"
>
    <then>
        <step
            id          = "LeftBound"
            description = "Изменение левой границы"
            comment-ru  = "j-я ячейка оказалась левее ячейки, соответствующей
                искомой порядковой статистике. Левой границей области поиска
                назначаем i-ю ячейку."
```

```
                comment-en  = "j-th cell is located to the left of the order
                    statistic's cell. We make i-th cell the left bound of the search
                    area."
                level       = "1"
            >
                <draw>
                        @visualizer.syncArray();
                        @visualizer.resetCellStyles();
                        if (@algorithm == @QUANTILES) {
                            @visualizer.setCellStyles(@Quantiles@qn, 1);
                            @visualizer.setCellStyles(@Quantiles@found, 2);
                        }
                        @visualizer.dimCells(0, @p - 1);
                        @visualizer.dimCells(@r + 1, @a.length - 1);
                        @visualizer.setCellStyles(new int[] {@k}, 1);
                        @visualizer.resetIndices();
                        @visualizer.updateIndex(@i, "i");
                        @visualizer.updateIndex(@j, "j");
                        @visualizer.updateExchangeArrow(-1, -1);
                        @visualizer.updateWatch(-1);
                </draw>
                <action>
                    @p @= @i;
                </action>
            </step>
        </then>
    </if>
    <if
        id          = "RightBoundTest"
        description = "Изменение правой границы"
        test        = "@k &lt; @i"
        level       = "-1"
    >
        <then>
            <step
                id          = "RightBound"
                description = "Изменение правой границы"
                comment-ru  = "i-я ячейка оказалась правее ячейки, соответствующей
                    искомой порядковой статистике. Правой границей области поиска
                    назначаем j-ю ячейку."
                comment-en  = "i-th cell is located to the right of the order
                    statistic's cell. We make j-th cell the right bound of the search
                    area."
                level       = "1"
            >
                <draw>
                        @visualizer.syncArray();
                        @visualizer.resetCellStyles();
                        if (@algorithm == @QUANTILES) {
                            @visualizer.setCellStyles(@Quantiles@qn, 1);
                            @visualizer.setCellStyles(@Quantiles@found, 2);
                        }
                        @visualizer.dimCells(0, @p - 1);
                        @visualizer.dimCells(@r + 1, @a.length - 1);
                        @visualizer.setCellStyles(new int[] {@k}, 1);
                        @visualizer.resetIndices();
                        @visualizer.updateIndex(@i, "i");
                        @visualizer.updateIndex(@j, "j");
                        @visualizer.updateExchangeArrow(-1, -1);
                        @visualizer.updateWatch(-1);
                </draw>
                <action>
                    @r @= @j;
                </action>
            </step>
        </then>
    </if>
</while>
<step
    id          = "OrderStatisticFound"
    description = "Порядковая статистика найдена"
    comment-ru  = "Правая граница области поиска A[{0}..{1}] не превосходит левой. Это
         означает, что искомая порядковая статистика заняла свое место."
    comment-en  = "The right bound of the search area A[{0}..{1}] does not exceed the
        left bound. It means that the order statistic has taken its place."
    comment-args= "new Integer(@p), new Integer(@r)"
```

```
        level       = "1"
    >
        <draw>
            @visualizer.syncArray();
            @visualizer.resetCellStyles();
            if (@algorithm == @QUANTILES) {
                @visualizer.setCellStyles(@Quantiles@qn, 1);
                @visualizer.setCellStyles(@Quantiles@found, 2);
            }
            @visualizer.dimCells(0, @p - 1);
            @visualizer.dimCells(@r + 1, @a.length - 1);
            @visualizer.setCellStyles(new int[] {@k}, 2);
            @visualizer.resetIndices();
            @visualizer.updateExchangeArrow(-1, -1);
            @visualizer.updateWatch(-1);
        </draw>
        <direct/>
    </step>
</auto>

<auto
    id          = "Quantiles"
    description = "Поиск k-квантилей в массиве a[]"
>
    <variable
        description = "Порядок квантилей"
        type        = "int"
        name        = "k"
    />
    <variable
        description = "Переменная цикла"
        type        = "int"
        name        = "i"
    />
    <variable
        description = "Номера искомых порядковых статистик"
        type        = "int[]"
        name        = "qn"
    />
    <variable
        description = "Индексы найденных порядковых статистик"
        type        = "int[]"
        name        = "found"
    />
    <step
        id          = "Init"
        description = "Инициализация"
        comment-ru  = "Ищем {0}-квантили. Это элементы, которые занимают выделенные ячейки
             в отсортированном массиве и разбивают его на {0} равных частей."
        comment-en  = "Searching for the {0}-th quantiles. These are the elements, which
            occupy the highlighted cells in sorted array and split it into {0} equal
            parts."
        comment-args= "new Integer(@k)"
        level       = "1"
    >
        <draw>
            @visualizer.syncArray();
            @visualizer.resetCellStyles();
            @visualizer.setCellStyles(@found, 2);
            @visualizer.setCellStyles(@qn, 2);
            @visualizer.resetIndices();
            @visualizer.updateExchangeArrow(-1, -1);
            @visualizer.updateWatch(-1);
        </draw>
        <direct>
            @qn = new int[@k];
            @qn[0] = -1;
            for (int j = 1; j &lt; @k; j++)
                @qn[j] = j * @a.length / @k;
            @found = new int[@k - 1];
            for (int j = 0; j &lt; @k - 1; j++)
                @found[j] = -1;
            @i = 1;
        </direct>
    </step>
    <while
```

```
            id          = "Loop"
            description = "Цикл по квантилям"
            test        = "@i &lt; @k"
            level       = "-1"
        >
            <step
                id          = "PreOrderStatisticCall"
                description = "Подготовка к вызову OrderStatistic"
                comment-ru  = "Найдем квантиль номер {0}. Для этого запустим поиск порядковой
                    статистики с соответствующими параметрами."
                comment-en  = "Let us find the quantile number {0}. We will do it via search
                    for order statistic with corresponding parameters."
                comment-args= "new Integer(@i)"
                level       = "1"
            >
                <draw>
                    @visualizer.syncArray();
                    @visualizer.resetCellStyles();
                    @visualizer.setCellStyles(@qn, 1);
                    @visualizer.setCellStyles(@found, 2);
                    @visualizer.setCellStyles(new int[] {@qn[@i]}, 2);
                    @visualizer.resetIndices();
                    @visualizer.updateExchangeArrow(-1, -1);
                    @visualizer.updateWatch(-1);
                </draw>
                <action>
                    @OrderStatistic@p @= @qn[@i - 1] + 1;
                    @OrderStatistic@r @= @a.length - 1;
                    @OrderStatistic@k @= @qn[@i];
                </action>
            </step>
            <call-auto id="OrderStatistic"/>
            <step
                id          = "PostOrderStatisticCall"
                description = "Завершение итерации цикла"
                level       = "-1"
            >
                <action>
                    @found[@i - 1] @= @qn[@i];
                    @i @= @i + 1;
                </action>
            </step>
        </while>
    </auto>

</algorithm>
```

OrderStatistics-Configuration.xml

```
<?xml version="1.0" encoding="WINDOWS-1251"?>

<configuration>
    <property
        description = "Comment pane height"
        param       = "comment-height"
        value       = "60"
    />
    <group
        description = "Algorithm selection"
        param       = "algorithm"
    >
        <message
            description = "Algorithm selection prompt"
            param       = "prompt"
            message-ru  = "Будем искать"
            message-en  = "Search for"
        />
        <message
            description = "OrderStatistic item"
            param       = "orderStatistic"
            message-ru  = "k-ю порядковую статистику"
            message-en  = "k-th order statistic"
        />
        <message
            description = "Quantiles item"
```

```
        param       = "quantiles"
        message-ru  = "k-квантили"
        message-en  = "k-th quantiles"
    />
    <adjustablePanel
        description     = "Algorithm parameter"
        param           = "k"
        caption-ru      = "при k={0,number,####}"
        caption-en      = "where k={0,number,####}"
        hint-ru         = "Параметр алгоритма"
        hint-en         = "Algorithm parameter"
        value           = "2"
        minimum         = "1"
        maximum         = "5"
        unitIncrement   = "1"
        blockIncrement  = "0"
        blockInterval   = "0"
    >
        <button
            param       = "incrementButton"
            caption-ru  = "&gt;&gt;"
            caption-en  = "&gt;&gt;"
            hint-ru     = "Увеличить k"
            hint-en     = "Increase k"
        />
        <button
            param       = "decrementButton"
            caption-ru  = "&lt;&lt;"
            caption-en  = "&lt;&lt;"
            hint-ru     = "Уменьшить k"
            hint-en     = "Decrease k"
        />
    </adjustablePanel>
</group>
<adjustablePanel
    description     = "Number of array elements"
    param           = "elements"
    caption-ru      = "Элементов: {0,number,####}"
    caption-en      = "Elements: {0,number,####}"
    hint-ru         = "Количество элементов в массиве"
    hint-en         = "Number of array elements"
    value           = "5"
    minimum         = "5"
    maximum         = "16"
    unitIncrement   = "1"
    blockIncrement  = "0"
    blockInterval   = "0"
>
    <button
        param       = "incrementButton"
        caption-ru  = "&gt;&gt;"
        caption-en  = "&gt;&gt;"
        hint-ru     = "Увеличить количество элементов"
        hint-en     = "Increase number of elements"
    />
    <button
        param       = "decrementButton"
        caption-ru  = "&lt;&lt;"
        caption-en  = "&lt;&lt;"
        hint-ru     = "Уменьшить количество элементов"
        hint-en     = "Decrease number of elements"
    />
</adjustablePanel>
<button
    description = "Fills the array with random values"
    param       = "button-random"
    caption-ru  = "Случайно"
    caption-en  = "Random"
    hint-ru     = "Заполнить массив случайными значениями"
    hint-en     = "Fill the array with random values"
/>
<button
    description = "Opens the save/load dialog"
    param       = "button-saveLoad"
    caption-ru  = "Сохранить/загрузить"
    caption-en  = "Save/load"
```

```
    hint-ru      = "Открыть диалог загрузки/сохранения"
    hint-en      = "Open the save/load dialog"
/>
<group
    description = "Save/load dialog configuration"
    param       = "SaveLoadDialog"
>
    <property
        description = "Comment pane height"
        param       = "CommentPane-lines"
        value       = "1"
    />
    <property
        description = "Text area width"
        param       = "columns"
        value       = "40"
    />
    <property
        description = "Text area height"
        param       = "rows"
        value       = "7"
    />
    <message
        description = "Comment for the array"
        param       = "arrayComment"
        message-ru  = "Массив с 0 на конце"
        message-en  = "Array ending with 0"
    />
    <message
        description = "Comment for the algorithm"
        param       = "algorithmComment"
        message-ru  = "Алгоритм: ORDERSTATISTIC или QUANTILES"
        message-en  = "Algorithm: ORDERSTATISTIC or QUANTILES"
    />
    <message
        description = "Comment for the order"
        param       = "kComment"
        message-ru  = "Параметр алгоритма"
        message-en  = "Algorithm parameter"
    />
    <message
        description = "Comment for the step"
        param       = "stepComment"
        message-ru  = "Номер шага алгоритма"
        message-en  = "Algorithm step number"
    />
    <message
        description = "Invalid array size"
        param       = "arraySizeError"
        message-ru  = "В массиве должно быть от {0} до {1} элементов."
        message-en  = "Array size must be in range {0}..{1}"
    />
    <message
        description = "Invalid algorithm name"
        param       = "algorithmNameError"
        message-ru  = "Извесные алгоритмы: ORDERSTATISTIC и QUANTILES."
        message-en  = "Known algorithms: ORDERSTATISTIC and QUANTILES."
    />
</group>
<styleset
    description = "Array cell styles"
    param       = "cellStyles"
>
    <style
        description     = "Normal cell, normal mode"
        text-color      = "000000"
        text-align      = "0.5"
        border-color    = "000000"
        border-status   = "true"
        fill-color      = "cccccc"
        fill-status     = "true"
        aspect-status   = "false"
        padding         = "0.2"
    >
        <font
            face    = "SansSerif"
```

```
            size    = "20"
            style   = "plain"
        />
    </style>
    <style
        description = "Highlighted cell, normal mode"
        text-color  = "000000"
        fill-color  = "aaccaa"
    />
    <style
        description = "Strongly highlighted cell, normal mode"
        text-color  = "000000"
        fill-color  = "aaffaa"
    />
    <style
        description     = "Normal cell, dimmed mode"
        text-color      = "aaaaaa"
        fill-color      = "cccccc"
    />
    <style
        description = "Highlighted cell, dimmed mode"
        text-color  = "999999"
        fill-color  = "bbccbb"
    />
    <style
        description = "Strongly highlighted, dimmed mode"
        text-color  = "999999"
        fill-color  = "bbeebb"
    />
</styleset>
<styleset
    description = "Cell indices"
    param       = "indexStyles"
>
    <style
        description     = "Main style"
        text-color      = "808080"
        text-align      = "0.5"
        border-color    = "000000"
        border-status   = "false"
        fill-color      = "ffffff"
        fill-status     = "false"
        aspect-status   = "false"
        padding         = "0.1"
    >
        <font
            face    = "SansSerif"
            size    = "20"
            style   = "plain"
        />
    </style>
    <style
        description     = "Highlighted style"
        text-color      = "000000"
        border-color    = "cccccc"
        border-status   = "true"
        fill-color      = "eeeeee"
        fill-status     = "true"
    />
</styleset>
<styleset
    description = "Arrow styles"
    param       = "arrowStyles"
>
    <style
        description     = "Normal style"
        text-color      = "000000"
        text-align      = "0.5"
        border-color    = "000000"
        border-status   = "true"
        fill-color      = "000000"
        fill-status     = "true"
        aspect-status   = "false"
        padding         = "0.2"
    >
        <font
```

```
                face    = "SansSerif"
                size    = "20"
                style   = "plain"
            />
        </style>
    </styleset>
    <styleset
        description = "Watch styles"
        param       = "watchStyles"
    >
        <style
            description     = "Main style"
            text-color      = "000000"
            text-align      = "0.0"
            message-align   = "0.0"
            border-color    = "000000"
            border-status   = "false"
            fill-color      = "ffffff"
            fill-status     = "true"
            aspect-status   = "false"
            padding         = "0.1"
        >
            <font
                face    = "SansSerif"
                size    = "20"
                style   = "plain"
            />
        </style>
    </styleset>
    <property
        description = "Watch format"
        param       = "watchMsg"
        value       = "X = {0}"
    />
    <property
        description = "Minimal index height"
        param       = "minIndexHeight"
        value       = "16"
    />
    <property
        description = "Maximal element value"
        param       = "maxElementValue"
        value       = "99"
    />
</configuration>
```

Приложение 4. Сгенерированные исходные коды визуализатора

```
package ru.ifmo.vizi.orderstatistics;

import ru.ifmo.vizi.base.auto.*;
import java.util.Locale;

public final class OrderStatistics extends BaseAutoReverseAutomata {
    /**
      * Модель данных.
      */
    public final Data d = new Data();

    /**
      * Конструктор для языка
      */
    public OrderStatistics(Locale locale) {
        super("ru.ifmo.vizi.orderstatistics.Comments", locale);
        init(new Main(), d);
    }

    /**
      * Данные.
      */
    public final class Data {
        /**
```

```
  * Экземпляр апплета.
  */
public OrderStatisticsVisualizer visualizer = null;

/**
  * Алгоритм поиска порядковой статистики.
  */
public static final int ORDERSTATISTIC = 1;

/**
  * Алгоритм поиска квантилей.
  */
public static final int QUANTILES = 2;

/**
  * Массив чисел для поиска.
  */
public int[] a = new int[] {4,5,2,1,3};

/**
  * Алгоритм.
  */
public int algorithm = ORDERSTATISTIC;

/**
  * Порядок искомых медиан/квантилей.
  */
public int k = 2;

/**
  * Левая граница области поиска (Процедура OrderStatistic).
  */
public int OrderStatistic_p;

/**
  * Правая граница области поиска (Процедура OrderStatistic).
  */
public int OrderStatistic_r;

/**
  * Номер искомой порядковой статистики (Процедура OrderStatistic).
  */
public int OrderStatistic_k;

/**
  * Граничный элемент при разделении (Процедура OrderStatistic).
  */
public int OrderStatistic_x;

/**
  * Индекс элемента при просмотре слева (Процедура OrderStatistic).
  */
public int OrderStatistic_i;

/**
  * Индекс элемента при просмотре справа (Процедура OrderStatistic).
  */
public int OrderStatistic_j;

/**
  * Временная переменная для обмена (Процедура OrderStatistic).
  */
public int OrderStatistic_t;

/**
  * Признак окончания основного цикла (Процедура OrderStatistic).
  */
public boolean OrderStatistic_br;

/**
  * Порядок квантилей (Процедура Quantiles).
  */
public int Quantiles_k;

/**
  * Переменная цикла (Процедура Quantiles).
```

```
      */
    public int Quantiles_i;

    /**
      * Номера искомых порядковых статистик (Процедура Quantiles).
      */
    public int[] Quantiles_qn;

    /**
      * Индексы найденных порядковых статистик (Процедура Quantiles).
      */
    public int[] Quantiles_found;

    public String toString() {
        StringBuffer buf = new StringBuffer();
        buf.append("k=").append(d.k).append(";");
        buf.append("Quantiles_i=").append(d.Quantiles_i).append(";");
        buf.append("OrderStatistic_p=").append(d.OrderStatistic_p).append(";");
        buf.append("OrderStatistic_r=").append(d.OrderStatistic_r).append(";");
        buf.append("OrderStatistic_x=").append(d.OrderStatistic_x).append(";");
        buf.append("OrderStatistic_i=").append(d.OrderStatistic_i).append(";");
        buf.append("OrderStatistic_j=").append(d.OrderStatistic_j).append(";");
        return buf.toString();
    }
}

/**
  * Главный автомат.
  */
private final class Main extends BaseAutomata implements Automata {
    /**
      * Начальное состояние автомата.
      */
    private final int START_STATE = 0;

    /**
      * Конечное состояние автомата.
      */
    private final int END_STATE = 7;

    /**
      * Конструктор.
      */
    public Main() {
        super(
            "Main",
            0, // Номер начального состояния
            7, // Номер конечного состояния
            new String[]{
                "Начальное состояние",
                "Установка параметров и запуск нужного автомата",
                "Установка параметров и запуск нужного автомата (окончание)",
                "Подготовка к вызову OrderStatistic",
                "Поиск k-й порядковой статистики в массиве a, область поиска - p..r
                    (автомат)",
                "Подготовка к вызову Quantiles",
                "Поиск k-квантилей в массиве a[] (автомат)",
                "Конечное состояние"
            }, new int[]{
                Integer.MAX_VALUE, // Начальное состояние,
                -1, // Установка параметров и запуск нужного автомата
                -1, // Установка параметров и запуск нужного автомата (окончание)
                -1, // Подготовка к вызову OrderStatistic
                CALL_AUTO_LEVEL, // Поиск k-й порядковой статистики в массиве a, область
                    поиска - p..r (автомат)
                -1, // Подготовка к вызову Quantiles
                CALL_AUTO_LEVEL, // Поиск k-квантилей в массиве a[] (автомат)
                Integer.MAX_VALUE, // Конечное состояние
            }
        );
    }

    /**
      * Сделать один шаг автомата в перед.
      */
    protected void doStepForward(int level) {
```

```
// Переход в следующее состояние
switch (state) {
    case START_STATE: { // Начальное состояние
        state = 1; // Установка параметров и запуск нужного автомата
        break;
    }
    case 1: { // Установка параметров и запуск нужного автомата
        if (d.algorithm == d.ORDERSTATISTIC) {
            state = 3; // Подготовка к вызову OrderStatistic
        } else {
            state = 5; // Подготовка к вызову Quantiles
        }
        break;
    }
    case 2: { // Установка параметров и запуск нужного автомата (окончание)
        state = END_STATE;
        break;
    }
    case 3: { // Подготовка к вызову OrderStatistic
        state = 4; // Поиск k-й порядковой статистики в массиве a, область поиска
            - p..r (автомат)
        break;
    }
    case 4: { // Поиск k-й порядковой статистики в массиве a, область поиска -
        p..r (автомат)
        if (child.isAtEnd()) {
            child = null;
            stack.pushBoolean(true);
            state = 2; // Установка параметров и запуск нужного автомата
                (окончание)
        }
        break;
    }
    case 5: { // Подготовка к вызову Quantiles
        state = 6; // Поиск k-квантилей в массиве a[] (автомат)
        break;
    }
    case 6: { // Поиск k-квантилей в массиве a[] (автомат)
        if (child.isAtEnd()) {
            child = null;
            stack.pushBoolean(false);
            state = 2; // Установка параметров и запуск нужного автомата
                (окончание)
        }
        break;
    }
}

// Действие в текущем состоянии
switch (state) {
    case 1: { // Установка параметров и запуск нужного автомата
        break;
    }
    case 2: { // Установка параметров и запуск нужного автомата (окончание)
        break;
    }
    case 3: { // Подготовка к вызову OrderStatistic
        startSection();
        storeField(d, "OrderStatistic_p");
        d.OrderStatistic_p = 0;
        storeField(d, "OrderStatistic_r");
        d.OrderStatistic_r = d.a.length - 1;
        storeField(d, "OrderStatistic_k");
        d.OrderStatistic_k = d.k;
        break;
    }
    case 4: { // Поиск k-й порядковой статистики в массиве a, область поиска -
        p..r (автомат)
        if (child == null) {
            child = new OrderStatistic();
            child.toStart();
        }
        child.stepForward(level);
        step--;
        break;
    }
```

```
            case 5: { // Подготовка к вызову Quantiles
                d.Quantiles_k = d.k;
                break;
            }
            case 6: { // Поиск k-квантилей в массиве a[] (автомат)
                if (child == null) {
                    child = new Quantiles();
                    child.toStart();
                }
                child.stepForward(level);
                step--;
                break;
            }
        }
    }

    /**
     * Сделать один шаг автомата назад.
     */
    protected void doStepBackward(int level) {
        // Обращение действия в текущем состоянии
        switch (state) {
            case 1: { // Установка параметров и запуск нужного автомата
                break;
            }
            case 2: { // Установка параметров и запуск нужного автомата (окончание)
                break;
            }
            case 3: { // Подготовка к вызову OrderStatistic
                restoreSection();
                break;
            }
            case 4: { // Поиск k-й порядковой статистики в массиве a, область поиска -
                p..r (автомат)
                if (child == null) {
                    child = new OrderStatistic();
                    child.toEnd();
                }
                child.stepBackward(level);
                step++;
                break;
            }
            case 5: { // Подготовка к вызову Quantiles
                break;
            }
            case 6: { // Поиск k-квантилей в массиве a[] (автомат)
                if (child == null) {
                    child = new Quantiles();
                    child.toEnd();
                }
                child.stepBackward(level);
                step++;
                break;
            }
        }

        // Переход в предыдущее состояние
        switch (state) {
            case 1: { // Установка параметров и запуск нужного автомата
                state = START_STATE;
                break;
            }
            case 2: { // Установка параметров и запуск нужного автомата (окончание)
                if (stack.popBoolean()) {
                    state = 4; // Поиск k-й порядковой статистики в массиве a, область
                        поиска - p..r (автомат)
                } else {
                    state = 6; // Поиск k-квантилей в массиве a[] (автомат)
                }
                break;
            }
            case 3: { // Подготовка к вызову OrderStatistic
                state = 1; // Установка параметров и запуск нужного автомата
                break;
            }
```

```
            case 4: { // Поиск k-й порядковой статистики в массиве a, область поиска -
                p..r (автомат)
                if (child.isAtStart()) {
                    child = null;
                    state = 3; // Подготовка к вызову OrderStatistic
                }
                break;
            }
            case 5: { // Подготовка к вызову Quantiles
                state = 1; // Установка параметров и запуск нужного автомата
                break;
            }
            case 6: { // Поиск k-квантилей в массиве a[] (автомат)
                if (child.isAtStart()) {
                    child = null;
                    state = 5; // Подготовка к вызову Quantiles
                }
                break;
            }
            case END_STATE: { // Начальное состояние
                state = 2; // Установка параметров и запуск нужного автомата (окончание)
                break;
            }
        }
    }

    /**
      * Комментарий к текущему состоянию
      */
    public String getComment() {
        String comment = "";
        Object[] args = null;
        // Выбор комментария
        switch (state) {
            case START_STATE: { // Начальное состояние
                comment = OrderStatistics.this.getComment("Main.START_STATE");
                break;
            }
            case 4: { // Поиск k-й порядковой статистики в массиве a, область поиска -
                p..r (автомат)
                comment = child.getComment();
                args = new Object[0];
                break;
            }
            case 6: { // Поиск k-квантилей в массиве a[] (автомат)
                comment = child.getComment();
                args = new Object[0];
                break;
            }
            case END_STATE: { // Конечное состояние
                comment = OrderStatistics.this.getComment("Main.END_STATE");
                break;
            }
        }

        return java.text.MessageFormat.format(comment, args);
    }

    /**
      * Выполняет действия по отрисовке состояния
      */
    public void drawState() {
        switch (state) {
            case START_STATE: { // Начальное состояние
                d.visualizer.syncArray();
                d.visualizer.resetCellStyles();
                if (d.algorithm == d.ORDERSTATISTIC) {
                    d.visualizer.setCellStyles(new int[] {d.k}, 2);
                } else if (d.algorithm == d.QUANTILES) {
                    int q[] = new int[d.k - 1];
                    for (int j = 1; j < d.k; j++)
                        q[j - 1] = d.a.length * j / d.k;
                    d.visualizer.setCellStyles(q, 2);
                }
                d.visualizer.resetIndices();
                d.visualizer.updateExchangeArrow(-1, -1);
```

```
                d.visualizer.updateWatch(-1);
                break;
            }
            case 4: { // Поиск k-й порядковой статистики в массиве a, область поиска -
                p..r (автомат)
                child.drawState();
                break;
            }
            case 6: { // Поиск k-квантилей в массиве a[] (автомат)
                child.drawState();
                break;
            }
            case END_STATE: { // Конечное состояние
                d.visualizer.syncArray();
                d.visualizer.resetCellStyles();
                if (d.algorithm == d.ORDERSTATISTIC)
                    d.visualizer.setCellStyles(new int[] {d.k}, 2);
                else if (d.algorithm == d.QUANTILES)
                    d.visualizer.setCellStyles(d.Quantiles_qn, 2);
                d.visualizer.resetIndices();
                d.visualizer.updateExchangeArrow(-1, -1);
                d.visualizer.updateWatch(-1);
                break;
            }
        }
    }
}

/**
  * Поиск k-й порядковой статистики в массиве a, область поиска - p..r.
  */
private final class OrderStatistic extends BaseAutomata implements Automata {
    /**
      * Начальное состояние автомата.
      */
    private final int START_STATE = 0;

    /**
      * Конечное состояние автомата.
      */
    private final int END_STATE = 26;

    /**
      * Конструктор.
      */
    public OrderStatistic() {
        super(
            "OrderStatistic",
            0, // Номер начального состояния
            26, // Номер конечного состояния
            new String[]{
                "Начальное состояние",
                "У Vizi проблемы, если в начале автомата стоит while",
                "Главный цикл алгоритма",
                "Введение",
                "Выбор барьера",
                "Разбиение массива",
                "Введение в LoopI",
                "Просмотр слева",
                "Объясняем действия",
                "Сдвигаемся на один элемент вправо",
                "Введение в LoopJ",
                "Просмотр справа",
                "Объясняем действия",
                "Сдвигаемся на один элемент влево",
                "Меняем?",
                "Меняем? (окончание)",
                "Объясняем действия",
                "Обмен",
                "Разбиение завершено",
                "Изменение левой границы",
                "Изменение левой границы (окончание)",
                "Изменение левой границы",
                "Изменение правой границы",
                "Изменение правой границы (окончание)",
                "Изменение правой границы",
```

```
            "Порядковая статистика найдена",
            "Конечное состояние"
        }, new int[]{
            Integer.MAX_VALUE, // Начальное состояние,
            -1, // У Vizi проблемы, если в начале автомата стоит while
            -1, // Главный цикл алгоритма
            1, // Введение
            1, // Выбор барьера
            -1, // Разбиение массива
            1, // Введение в LoopI
            -1, // Просмотр слева
            0, // Объясняем действия
            -1, // Сдвигаемся на один элемент вправо
            1, // Введение в LoopJ
            -1, // Просмотр справа
            1, // Объясняем действия
            -1, // Сдвигаемся на один элемент влево
            -1, // Меняем?
            -1, // Меняем? (окончание)
            1, // Объясняем действия
            -1, // Обмен
            1, // Разбиение завершено
            -1, // Изменение левой границы
            -1, // Изменение левой границы (окончание)
            1, // Изменение левой границы
            -1, // Изменение правой границы
            -1, // Изменение правой границы (окончание)
            1, // Изменение правой границы
            1, // Порядковая статистика найдена
            Integer.MAX_VALUE, // Конечное состояние
        }
    );
}

/**
  * Сделать один шаг автомата в перед.
  */
protected void doStepForward(int level) {
    // Переход в следующее состояние
    switch (state) {
        case START_STATE: { // Начальное состояние
            state = 1; // У Vizi проблемы, если в начале автомата стоит while
            break;
        }
        case 1: { // У Vizi проблемы, если в начале автомата стоит while
            stack.pushBoolean(false);
            state = 2; // Главный цикл алгоритма
            break;
        }
        case 2: { // Главный цикл алгоритма
            if (d.OrderStatistic_p < d.OrderStatistic_r) {
                state = 3; // Введение
            } else {
                state = 25; // Порядковая статистика найдена
            }
            break;
        }
        case 3: { // Введение
            state = 4; // Выбор барьера
            break;
        }
        case 4: { // Выбор барьера
            stack.pushBoolean(false);
            state = 5; // Разбиение массива
            break;
        }
        case 5: { // Разбиение массива
            if (d.OrderStatistic_i <= d.OrderStatistic_j) {
                state = 6; // Введение в LoopI
            } else {
                state = 18; // Разбиение завершено
            }
            break;
        }
        case 6: { // Введение в LoopI
            stack.pushBoolean(false);
```

```
        state = 7; // Просмотр слева
        break;
    }
    case 7: { // Просмотр слева
        if (d.a[d.OrderStatistic_i] < d.OrderStatistic_x) {
            state = 8; // Объясняем действия
        } else {
            state = 10; // Введение в LoopJ
        }
        break;
    }
    case 8: { // Объясняем действия
        state = 9; // Сдвигаемся на один элемент вправо
        break;
    }
    case 9: { // Сдвигаемся на один элемент вправо
        stack.pushBoolean(true);
        state = 7; // Просмотр слева
        break;
    }
    case 10: { // Введение в LoopJ
        stack.pushBoolean(false);
        state = 11; // Просмотр справа
        break;
    }
    case 11: { // Просмотр справа
        if (d.OrderStatistic_x < d.a[d.OrderStatistic_j]) {
            state = 12; // Объясняем действия
        } else {
            state = 14; // Меняем?
        }
        break;
    }
    case 12: { // Объясняем действия
        state = 13; // Сдвигаемся на один элемент влево
        break;
    }
    case 13: { // Сдвигаемся на один элемент влево
        stack.pushBoolean(true);
        state = 11; // Просмотр справа
        break;
    }
    case 14: { // Меняем?
        if (d.OrderStatistic_i <= d.OrderStatistic_j) {
            state = 16; // Объясняем действия
        } else {
            stack.pushBoolean(false);
            state = 15; // Меняем? (окончание)
        }
        break;
    }
    case 15: { // Меняем? (окончание)
        stack.pushBoolean(true);
        state = 5; // Разбиение массива
        break;
    }
    case 16: { // Объясняем действия
        state = 17; // Обмен
        break;
    }
    case 17: { // Обмен
        stack.pushBoolean(true);
        state = 15; // Меняем? (окончание)
        break;
    }
    case 18: { // Разбиение завершено
        state = 19; // Изменение левой границы
        break;
    }
    case 19: { // Изменение левой границы
        if (d.OrderStatistic_j < d.OrderStatistic_k) {
            state = 21; // Изменение левой границы
        } else {
            stack.pushBoolean(false);
            state = 20; // Изменение левой границы (окончание)
        }
```

```
            break;
        }
        case 20: { // Изменение левой границы (окончание)
            state = 22; // Изменение правой границы
            break;
        }
        case 21: { // Изменение левой границы
            stack.pushBoolean(true);
            state = 20; // Изменение левой границы (окончание)
            break;
        }
        case 22: { // Изменение правой границы
            if (d.OrderStatistic_k < d.OrderStatistic_i) {
                state = 24; // Изменение правой границы
            } else {
                stack.pushBoolean(false);
                state = 23; // Изменение правой границы (окончание)
            }
            break;
        }
        case 23: { // Изменение правой границы (окончание)
            stack.pushBoolean(true);
            state = 2; // Главный цикл алгоритма
            break;
        }
        case 24: { // Изменение правой границы
            stack.pushBoolean(true);
            state = 23; // Изменение правой границы (окончание)
            break;
        }
        case 25: { // Порядковая статистика найдена
            state = END_STATE;
            break;
        }
    }

    // Действие в текущем состоянии
    switch (state) {
        case 1: { // У Vizi проблемы, если в начале автомата стоит while
            break;
        }
        case 2: { // Главный цикл алгоритма
            break;
        }
        case 3: { // Введение
            break;
        }
        case 4: { // Выбор барьера
            startSection();
            storeField(d, "OrderStatistic_x");
            d.OrderStatistic_x = d.a[d.OrderStatistic_k];
            storeField(d, "OrderStatistic_i");
            d.OrderStatistic_i = d.OrderStatistic_p;
            storeField(d, "OrderStatistic_j");
            d.OrderStatistic_j = d.OrderStatistic_r;
            break;
        }
        case 5: { // Разбиение массива
            break;
        }
        case 6: { // Введение в LoopI
            break;
        }
        case 7: { // Просмотр слева
            break;
        }
        case 8: { // Объясняем действия
            break;
        }
        case 9: { // Сдвигаемся на один элемент вправо
            startSection();
            storeField(d, "OrderStatistic_i");
            d.OrderStatistic_i = d.OrderStatistic_i + 1;
            break;
        }
        case 10: { // Введение в LoopJ
```

```
            break;
        }
        case 11: { // Просмотр справа
            break;
        }
        case 12: { // Объясняем действия
            break;
        }
        case 13: { // Сдвигаемся на один элемент влево
            startSection();
            storeField(d, "OrderStatistic_j");
            d.OrderStatistic_j = d.OrderStatistic_j - 1;
            break;
        }
        case 14: { // Меняем?
            break;
        }
        case 15: { // Меняем? (окончание)
            break;
        }
        case 16: { // Объясняем действия
            break;
        }
        case 17: { // Обмен
            d.OrderStatistic_t = d.a[d.OrderStatistic_i];
            d.a[d.OrderStatistic_i] = d.a[d.OrderStatistic_j];
            d.a[d.OrderStatistic_j] = d.OrderStatistic_t;
            d.OrderStatistic_i++;
            d.OrderStatistic_j--;
            break;
        }
        case 18: { // Разбиение завершено
            startSection();
            break;
        }
        case 19: { // Изменение левой границы
            break;
        }
        case 20: { // Изменение левой границы (окончание)
            break;
        }
        case 21: { // Изменение левой границы
            startSection();
            storeField(d, "OrderStatistic_p");
            d.OrderStatistic_p = d.OrderStatistic_i;
            break;
        }
        case 22: { // Изменение правой границы
            break;
        }
        case 23: { // Изменение правой границы (окончание)
            break;
        }
        case 24: { // Изменение правой границы
            startSection();
            storeField(d, "OrderStatistic_r");
            d.OrderStatistic_r = d.OrderStatistic_j;
            break;
        }
        case 25: { // Порядковая статистика найдена
            break;
        }
    }
}

/**
  * Сделать один шаг автомата назад.
  */
protected void doStepBackward(int level) {
    // Обращение действия в текущем состоянии
    switch (state) {
        case 1: { // У Vizi проблемы, если в начале автомата стоит while
            break;
        }
        case 2: { // Главный цикл алгоритма
            break;
```

```
}
case 3: { // Введение
    break;
}
case 4: { // Выбор барьера
    restoreSection();
    break;
}
case 5: { // Разбиение массива
    break;
}
case 6: { // Введение в LoopI
    break;
}
case 7: { // Просмотр слева
    break;
}
case 8: { // Объясняем действия
    break;
}
case 9: { // Сдвигаемся на один элемент вправо
    restoreSection();
    break;
}
case 10: { // Введение в LoopJ
    break;
}
case 11: { // Просмотр справа
    break;
}
case 12: { // Объясняем действия
    break;
}
case 13: { // Сдвигаемся на один элемент влево
    restoreSection();
    break;
}
case 14: { // Меняем?
    break;
}
case 15: { // Меняем? (окончание)
    break;
}
case 16: { // Объясняем действия
    break;
}
case 17: { // Обмен
    d.OrderStatistic_i--;
    d.OrderStatistic_j++;
    d.OrderStatistic_t = d.a[d.OrderStatistic_i];
    d.a[d.OrderStatistic_i] = d.a[d.OrderStatistic_j];
    d.a[d.OrderStatistic_j] = d.OrderStatistic_t;
    break;
}
case 18: { // Разбиение завершено
    restoreSection();
    break;
}
case 19: { // Изменение левой границы
    break;
}
case 20: { // Изменение левой границы (окончание)
    break;
}
case 21: { // Изменение левой границы
    restoreSection();
    break;
}
case 22: { // Изменение правой границы
    break;
}
case 23: { // Изменение правой границы (окончание)
    break;
}
case 24: { // Изменение правой границы
    restoreSection();
```

```
            break;
        }
        case 25: { // Порядковая статистика найдена
            break;
        }
    }

    // Переход в предыдущее состояние
    switch (state) {
        case 1: { // У Vizi проблемы, если в начале автомата стоит while
            state = START_STATE;
            break;
        }
        case 2: { // Главный цикл алгоритма
            if (stack.popBoolean()) {
                state = 23; // Изменение правой границы (окончание)
            } else {
                state = 1; // У Vizi проблемы, если в начале автомата стоит while
            }
            break;
        }
        case 3: { // Введение
            state = 2; // Главный цикл алгоритма
            break;
        }
        case 4: { // Выбор барьера
            state = 3; // Введение
            break;
        }
        case 5: { // Разбиение массива
            if (stack.popBoolean()) {
                state = 15; // Меняем? (окончание)
            } else {
                state = 4; // Выбор барьера
            }
            break;
        }
        case 6: { // Введение в LoopI
            state = 5; // Разбиение массива
            break;
        }
        case 7: { // Просмотр слева
            if (stack.popBoolean()) {
                state = 9; // Сдвигаемся на один элемент вправо
            } else {
                state = 6; // Введение в LoopI
            }
            break;
        }
        case 8: { // Объясняем действия
            state = 7; // Просмотр слева
            break;
        }
        case 9: { // Сдвигаемся на один элемент вправо
            state = 8; // Объясняем действия
            break;
        }
        case 10: { // Введение в LoopJ
            state = 7; // Просмотр слева
            break;
        }
        case 11: { // Просмотр справа
            if (stack.popBoolean()) {
                state = 13; // Сдвигаемся на один элемент влево
            } else {
                state = 10; // Введение в LoopJ
            }
            break;
        }
        case 12: { // Объясняем действия
            state = 11; // Просмотр справа
            break;
        }
        case 13: { // Сдвигаемся на один элемент влево
            state = 12; // Объясняем действия
            break;
```

```
            }
            case 14: { // Меняем?
                state = 11; // Просмотр справа
                break;
            }
            case 15: { // Меняем? (окончание)
                if (stack.popBoolean()) {
                    state = 17; // Обмен
                } else {
                    state = 14; // Меняем?
                }
                break;
            }
            case 16: { // Объясняем действия
                state = 14; // Меняем?
                break;
            }
            case 17: { // Обмен
                state = 16; // Объясняем действия
                break;
            }
            case 18: { // Разбиение завершено
                state = 5; // Разбиение массива
                break;
            }
            case 19: { // Изменение левой границы
                state = 18; // Разбиение завершено
                break;
            }
            case 20: { // Изменение левой границы (окончание)
                if (stack.popBoolean()) {
                    state = 21; // Изменение левой границы
                } else {
                    state = 19; // Изменение левой границы
                }
                break;
            }
            case 21: { // Изменение левой границы
                state = 19; // Изменение левой границы
                break;
            }
            case 22: { // Изменение правой границы
                state = 20; // Изменение левой границы (окончание)
                break;
            }
            case 23: { // Изменение правой границы (окончание)
                if (stack.popBoolean()) {
                    state = 24; // Изменение правой границы
                } else {
                    state = 22; // Изменение правой границы
                }
                break;
            }
            case 24: { // Изменение правой границы
                state = 22; // Изменение правой границы
                break;
            }
            case 25: { // Порядковая статистика найдена
                state = 2; // Главный цикл алгоритма
                break;
            }
            case END_STATE: { // Начальное состояние
                state = 25; // Порядковая статистика найдена
                break;
            }
        }
    }

    /**
     * Комментарий к текущему состоянию
     */
    public String getComment() {
        String comment = "";
        Object[] args = null;
        // Выбор комментария
        switch (state) {
```

```
            case 3: { // Введение
                comment = OrderStatistics.this.getComment("OrderStatistic.Introduction");
                args = new Object[]{new Integer(d.OrderStatistic_k), new
                    Integer(d.OrderStatistic_p), new Integer(d.OrderStatistic_r)};
                break;
            }
            case 4: { // Выбор барьера
                comment = OrderStatistics.this.getComment("OrderStatistic.Barrier");
                args = new Object[]{new Integer(d.OrderStatistic_k), new
                    Integer(d.OrderStatistic_x)};
                break;
            }
            case 6: { // Введение в LoopI
                comment = OrderStatistics.this.getComment("OrderStatistic.LoopIIntro");
                break;
            }
            case 8: { // Объясняем действия
                comment = OrderStatistics.this.getComment("OrderStatistic.PreLoopIStep");
                break;
            }
            case 10: { // Введение в LoopJ
                comment = OrderStatistics.this.getComment("OrderStatistic.LoopJIntro");
                break;
            }
            case 12: { // Объясняем действия
                comment = OrderStatistics.this.getComment("OrderStatistic.PreLoopJStep");
                break;
            }
            case 16: { // Объясняем действия
                comment = OrderStatistics.this.getComment("OrderStatistic.PreExchange");
                break;
            }
            case 18: { // Разбиение завершено
                comment =
                    OrderStatistics.this.getComment("OrderStatistic.PartitionComplete");
                args = new Object[]{new Integer(d.OrderStatistic_p), new
                    Integer(d.OrderStatistic_r)};
                break;
            }
            case 21: { // Изменение левой границы
                comment = OrderStatistics.this.getComment("OrderStatistic.LeftBound");
                break;
            }
            case 24: { // Изменение правой границы
                comment = OrderStatistics.this.getComment("OrderStatistic.RightBound");
                break;
            }
            case 25: { // Порядковая статистика найдена
                comment =
                    OrderStatistics.this.getComment("OrderStatistic.OrderStatisticFound");
                args = new Object[]{new Integer(d.OrderStatistic_p), new
                    Integer(d.OrderStatistic_r)};
                break;
            }
        }

        return java.text.MessageFormat.format(comment, args);
    }

    /**
      * Выполняет действия по отрисовке состояния
      */
    public void drawState() {
        switch (state) {
            case 3: { // Введение
                d.visualizer.syncArray();
                d.visualizer.resetCellStyles();
                if (d.algorithm == d.QUANTILES) {
                    d.visualizer.setCellStyles(d.Quantiles_qn, 1);
                    d.visualizer.setCellStyles(d.Quantiles_found, 2);
                }
                d.visualizer.dimCells(0, d.OrderStatistic_p - 1);
                d.visualizer.dimCells(d.OrderStatistic_r + 1, d.a.length - 1);
                d.visualizer.setCellStyles(new int[] {d.OrderStatistic_k}, 2);
                d.visualizer.resetIndices();
                d.visualizer.updateExchangeArrow(-1, -1);
```

```
        d.visualizer.updateWatch(-1);
        break;
    }
    case 4: { // Выбор барьера
        d.visualizer.syncArray();
        d.visualizer.resetCellStyles();
        if (d.algorithm == d.QUANTILES) {
            d.visualizer.setCellStyles(d.Quantiles_qn, 1);
            d.visualizer.setCellStyles(d.Quantiles_found, 2);
        }
        d.visualizer.dimCells(0, d.OrderStatistic_p - 1);
        d.visualizer.dimCells(d.OrderStatistic_r + 1, d.a.length - 1);
        d.visualizer.setCellStyles(new int[] {d.OrderStatistic_k}, 2);
        d.visualizer.resetIndices();
        d.visualizer.updateExchangeArrow(-1, -1);
        d.visualizer.updateWatch(d.OrderStatistic_x);
        break;
    }
    case 6: { // Введение в LoopI
        d.visualizer.syncArray();
        d.visualizer.resetCellStyles();
        if (d.algorithm == d.QUANTILES) {
            d.visualizer.setCellStyles(d.Quantiles_qn, 1);
            d.visualizer.setCellStyles(d.Quantiles_found, 2);
        }
        d.visualizer.dimCells(0, d.OrderStatistic_p - 1);
        d.visualizer.dimCells(d.OrderStatistic_r + 1, d.a.length - 1);
        d.visualizer.setCellStyles(new int[] {d.OrderStatistic_k}, 1);
        d.visualizer.resetIndices();
        d.visualizer.updateIndex(d.OrderStatistic_i, "i");
        d.visualizer.updateIndex(d.OrderStatistic_j, "j");
        d.visualizer.updateExchangeArrow(-1, -1);
        d.visualizer.updateWatch(d.OrderStatistic_x);
        break;
    }
    case 8: { // Объясняем действия
        d.visualizer.syncArray();
        d.visualizer.resetCellStyles();
        if (d.algorithm == d.QUANTILES) {
            d.visualizer.setCellStyles(d.Quantiles_qn, 1);
            d.visualizer.setCellStyles(d.Quantiles_found, 2);
        }
        d.visualizer.dimCells(0, d.OrderStatistic_p - 1);
        d.visualizer.dimCells(d.OrderStatistic_r + 1, d.a.length - 1);
        d.visualizer.setCellStyles(new int[] {d.OrderStatistic_k}, 1);
        d.visualizer.resetIndices();
        d.visualizer.updateIndex(d.OrderStatistic_i, "i");
        d.visualizer.updateIndex(d.OrderStatistic_j, "j");
        d.visualizer.updateExchangeArrow(-1, -1);
        d.visualizer.updateWatch(d.OrderStatistic_x);
        break;
    }
    case 10: { // Введение в LoopJ
        d.visualizer.syncArray();
        d.visualizer.resetCellStyles();
        if (d.algorithm == d.QUANTILES) {
            d.visualizer.setCellStyles(d.Quantiles_qn, 1);
            d.visualizer.setCellStyles(d.Quantiles_found, 2);
        }
        d.visualizer.dimCells(0, d.OrderStatistic_p - 1);
        d.visualizer.dimCells(d.OrderStatistic_r + 1, d.a.length - 1);
        d.visualizer.setCellStyles(new int[] {d.OrderStatistic_k}, 1);
        d.visualizer.resetIndices();
        d.visualizer.updateIndex(d.OrderStatistic_i, "i");
        d.visualizer.updateIndex(d.OrderStatistic_j, "j");
        d.visualizer.updateExchangeArrow(-1, -1);
        d.visualizer.updateWatch(d.OrderStatistic_x);
        break;
    }
    case 12: { // Объясняем действия
        d.visualizer.syncArray();
        d.visualizer.resetCellStyles();
        if (d.algorithm == d.QUANTILES) {
            d.visualizer.setCellStyles(d.Quantiles_qn, 1);
            d.visualizer.setCellStyles(d.Quantiles_found, 2);
        }
```

```
                d.visualizer.dimCells(0, d.OrderStatistic_p - 1);
                d.visualizer.dimCells(d.OrderStatistic_r + 1, d.a.length - 1);
                d.visualizer.setCellStyles(new int[] {d.OrderStatistic_k}, 1);
                d.visualizer.resetIndices();
                d.visualizer.updateIndex(d.OrderStatistic_i, "i");
                d.visualizer.updateIndex(d.OrderStatistic_j, "j");
                d.visualizer.updateExchangeArrow(-1, -1);
                d.visualizer.updateWatch(d.OrderStatistic_x);
                break;
            }
            case 16: { // Объясняем действия
                d.visualizer.syncArray();
                d.visualizer.resetCellStyles();
                if (d.algorithm == d.QUANTILES) {
                    d.visualizer.setCellStyles(d.Quantiles_qn, 1);
                    d.visualizer.setCellStyles(d.Quantiles_found, 2);
                }
                d.visualizer.dimCells(0, d.OrderStatistic_p - 1);
                d.visualizer.dimCells(d.OrderStatistic_r + 1, d.a.length - 1);
                d.visualizer.setCellStyles(new int[] {d.OrderStatistic_k}, 1);
                d.visualizer.resetIndices();
                d.visualizer.updateIndex(d.OrderStatistic_i, "i");
                d.visualizer.updateIndex(d.OrderStatistic_j, "j");
                d.visualizer.updateExchangeArrow(d.OrderStatistic_i, d.OrderStatistic_j);
                d.visualizer.updateWatch(d.OrderStatistic_x);
                break;
            }
            case 18: { // Разбиение завершено
                d.visualizer.syncArray();
                d.visualizer.resetCellStyles();
                if (d.algorithm == d.QUANTILES) {
                    d.visualizer.setCellStyles(d.Quantiles_qn, 1);
                    d.visualizer.setCellStyles(d.Quantiles_found, 2);
                }
                d.visualizer.dimCells(0, d.OrderStatistic_p - 1);
                d.visualizer.dimCells(d.OrderStatistic_r + 1, d.a.length - 1);
                d.visualizer.setCellStyles(new int[] {d.OrderStatistic_k}, 1);
                d.visualizer.resetIndices();
                d.visualizer.updateIndex(d.OrderStatistic_i - 1, "i-1");
                d.visualizer.updateIndex(d.OrderStatistic_i, "i");
                d.visualizer.updateIndex(d.OrderStatistic_j + 1, "j+1");
                d.visualizer.updateIndex(d.OrderStatistic_j, "j");
                d.visualizer.updateExchangeArrow(-1, -1);
                d.visualizer.updateWatch(d.OrderStatistic_x);
                break;
            }
            case 21: { // Изменение левой границы
                d.visualizer.syncArray();
                d.visualizer.resetCellStyles();
                if (d.algorithm == d.QUANTILES) {
                    d.visualizer.setCellStyles(d.Quantiles_qn, 1);
                    d.visualizer.setCellStyles(d.Quantiles_found, 2);
                }
                d.visualizer.dimCells(0, d.OrderStatistic_p - 1);
                d.visualizer.dimCells(d.OrderStatistic_r + 1, d.a.length - 1);
                d.visualizer.setCellStyles(new int[] {d.OrderStatistic_k}, 1);
                d.visualizer.resetIndices();
                d.visualizer.updateIndex(d.OrderStatistic_i, "i");
                d.visualizer.updateIndex(d.OrderStatistic_j, "j");
                d.visualizer.updateExchangeArrow(-1, -1);
                d.visualizer.updateWatch(-1);
                break;
            }
            case 24: { // Изменение правой границы
                d.visualizer.syncArray();
                d.visualizer.resetCellStyles();
                if (d.algorithm == d.QUANTILES) {
                    d.visualizer.setCellStyles(d.Quantiles_qn, 1);
                    d.visualizer.setCellStyles(d.Quantiles_found, 2);
                }
                d.visualizer.dimCells(0, d.OrderStatistic_p - 1);
                d.visualizer.dimCells(d.OrderStatistic_r + 1, d.a.length - 1);
                d.visualizer.setCellStyles(new int[] {d.OrderStatistic_k}, 1);
                d.visualizer.resetIndices();
                d.visualizer.updateIndex(d.OrderStatistic_i, "i");
                d.visualizer.updateIndex(d.OrderStatistic_j, "j");
```

```
                d.visualizer.updateExchangeArrow(-1, -1);
                d.visualizer.updateWatch(-1);
                break;
            }
            case 25: { // Порядковая статистика найдена
                d.visualizer.syncArray();
                d.visualizer.resetCellStyles();
                if (d.algorithm == d.QUANTILES) {
                    d.visualizer.setCellStyles(d.Quantiles_qn, 1);
                    d.visualizer.setCellStyles(d.Quantiles_found, 2);
                }
                d.visualizer.dimCells(0, d.OrderStatistic_p - 1);
                d.visualizer.dimCells(d.OrderStatistic_r + 1, d.a.length - 1);
                d.visualizer.setCellStyles(new int[] {d.OrderStatistic_k}, 2);
                d.visualizer.resetIndices();
                d.visualizer.updateExchangeArrow(-1, -1);
                d.visualizer.updateWatch(-1);
                break;
            }
        }
    }
}

/**
  * Поиск k-квантилей в массиве a[].
  */
private final class Quantiles extends BaseAutomata implements Automata {
    /**
      * Начальное состояние автомата.
      */
    private final int START_STATE = 0;

    /**
      * Конечное состояние автомата.
      */
    private final int END_STATE = 6;

    /**
      * Конструктор.
      */
    public Quantiles() {
        super(
            "Quantiles",
            0, // Номер начального состояния
            6, // Номер конечного состояния
            new String[]{
                "Начальное состояние",
                "Инициализация",
                "Цикл по квантилям",
                "Подготовка к вызову OrderStatistic",
                "Поиск k-й порядковой статистики в массиве a, область поиска - p..r
                    (автомат)",
                "Завершение итерации цикла",
                "Конечное состояние"
            }, new int[]{
                Integer.MAX_VALUE, // Начальное состояние,
                1, // Инициализация
                -1, // Цикл по квантилям
                1, // Подготовка к вызову OrderStatistic
                CALL_AUTO_LEVEL, // Поиск k-й порядковой статистики в массиве a, область
                    поиска - p..r (автомат)
                -1, // Завершение итерации цикла
                Integer.MAX_VALUE, // Конечное состояние
            }
        );
    }

    /**
      * Сделать один шаг автомата в перед.
      */
    protected void doStepForward(int level) {
        // Переход в следующее состояние
        switch (state) {
            case START_STATE: { // Начальное состояние
                state = 1; // Инициализация
                break;
```

```
        }
        case 1: { // Инициализация
            stack.pushBoolean(false);
            state = 2; // Цикл по квантилям
            break;
        }
        case 2: { // Цикл по квантилям
            if (d.Quantiles_i < d.Quantiles_k) {
                state = 3; // Подготовка к вызову OrderStatistic
            } else {
                state = END_STATE;
            }
            break;
        }
        case 3: { // Подготовка к вызову OrderStatistic
            state = 4; // Поиск k-й порядковой статистики в массиве a, область поиска
                - p..r (автомат)
            break;
        }
        case 4: { // Поиск k-й порядковой статистики в массиве a, область поиска -
            p..r (автомат)
            if (child.isAtEnd()) {
                child = null;
                state = 5; // Завершение итерации цикла
            }
            break;
        }
        case 5: { // Завершение итерации цикла
            stack.pushBoolean(true);
            state = 2; // Цикл по квантилям
            break;
        }
    }

    // Действие в текущем состоянии
    switch (state) {
        case 1: { // Инициализация
            d.Quantiles_qn = new int[d.Quantiles_k];
            d.Quantiles_qn[0] = -1;
            for (int j = 1; j < d.Quantiles_k; j++)
                d.Quantiles_qn[j] = j * d.a.length / d.Quantiles_k;
            d.Quantiles_found = new int[d.Quantiles_k - 1];
            for (int j = 0; j < d.Quantiles_k - 1; j++)
                d.Quantiles_found[j] = -1;
            d.Quantiles_i = 1;
            break;
        }
        case 2: { // Цикл по квантилям
            break;
        }
        case 3: { // Подготовка к вызову OrderStatistic
            startSection();
            storeField(d, "OrderStatistic_p");
            d.OrderStatistic_p = d.Quantiles_qn[d.Quantiles_i - 1] + 1;
            storeField(d, "OrderStatistic_r");
            d.OrderStatistic_r = d.a.length - 1;
            storeField(d, "OrderStatistic_k");
            d.OrderStatistic_k = d.Quantiles_qn[d.Quantiles_i];
            break;
        }
        case 4: { // Поиск k-й порядковой статистики в массиве a, область поиска -
            p..r (автомат)
            if (child == null) {
                child = new OrderStatistic();
                child.toStart();
            }
            child.stepForward(level);
            step--;
            break;
        }
        case 5: { // Завершение итерации цикла
            startSection();
            storeArray(d.Quantiles_found, d.Quantiles_i - 1);
            d.Quantiles_found[d.Quantiles_i - 1] = d.Quantiles_qn[d.Quantiles_i];
            storeField(d, "Quantiles_i");
            d.Quantiles_i = d.Quantiles_i + 1;
```

```
            break;
        }
    }
}

/**
  * Сделать один шаг автомата назад.
  */
protected void doStepBackward(int level) {
    // Обращение действия в текущем состоянии
    switch (state) {
        case 1: { // Инициализация
            break;
        }
        case 2: { // Цикл по квантилям
            break;
        }
        case 3: { // Подготовка к вызову OrderStatistic
            restoreSection();
            break;
        }
        case 4: { // Поиск k-й порядковой статистики в массиве a, область поиска -
            p..r (автомат)
            if (child == null) {
                child = new OrderStatistic();
                child.toEnd();
            }
            child.stepBackward(level);
            step++;
            break;
        }
        case 5: { // Завершение итерации цикла
            restoreSection();
            break;
        }
    }

    // Переход в предыдущее состояние
    switch (state) {
        case 1: { // Инициализация
            state = START_STATE;
            break;
        }
        case 2: { // Цикл по квантилям
            if (stack.popBoolean()) {
                state = 5; // Завершение итерации цикла
            } else {
                state = 1; // Инициализация
            }
            break;
        }
        case 3: { // Подготовка к вызову OrderStatistic
            state = 2; // Цикл по квантилям
            break;
        }
        case 4: { // Поиск k-й порядковой статистики в массиве a, область поиска -
            p..r (автомат)
            if (child.isAtStart()) {
                child = null;
                state = 3; // Подготовка к вызову OrderStatistic
            }
            break;
        }
        case 5: { // Завершение итерации цикла
            state = 4; // Поиск k-й порядковой статистики в массиве a, область поиска
                - p..r (автомат)
            break;
        }
        case END_STATE: { // Начальное состояние
            state = 2; // Цикл по квантилям
            break;
        }
    }
}

/**
```

```
            * Комментарий к текущему состоянию
            */
        public String getComment() {
            String comment = "";
            Object[] args = null;
            // Выбор комментария
            switch (state) {
                case 1: { // Инициализация
                    comment = OrderStatistics.this.getComment("Quantiles.Init");
                    args = new Object[]{new Integer(d.Quantiles_k)};
                    break;
                }
                case 3: { // Подготовка к вызову OrderStatistic
                    comment =
                        OrderStatistics.this.getComment("Quantiles.PreOrderStatisticCall");
                    args = new Object[]{new Integer(d.Quantiles_i)};
                    break;
                }
                case 4: { // Поиск k-й порядковой статистики в массиве a, область поиска -
                    p..r (автомат)
                    comment = child.getComment();
                    args = new Object[0];
                    break;
                }
            }

            return java.text.MessageFormat.format(comment, args);
        }

        /**
            * Выполняет действия по отрисовке состояния
            */
        public void drawState() {
            switch (state) {
                case 1: { // Инициализация
                    d.visualizer.syncArray();
                    d.visualizer.resetCellStyles();
                    d.visualizer.setCellStyles(d.Quantiles_found, 2);
                    d.visualizer.setCellStyles(d.Quantiles_qn, 2);
                    d.visualizer.resetIndices();
                    d.visualizer.updateExchangeArrow(-1, -1);
                    d.visualizer.updateWatch(-1);
                    break;
                }
                case 3: { // Подготовка к вызову OrderStatistic
                    d.visualizer.syncArray();
                    d.visualizer.resetCellStyles();
                    d.visualizer.setCellStyles(d.Quantiles_qn, 1);
                    d.visualizer.setCellStyles(d.Quantiles_found, 2);
                    d.visualizer.setCellStyles(new int[] {d.Quantiles_qn[d.Quantiles_i]}, 2);
                    d.visualizer.resetIndices();
                    d.visualizer.updateExchangeArrow(-1, -1);
                    d.visualizer.updateWatch(-1);
                    break;
                }
                case 4: { // Поиск k-й порядковой статистики в массиве a, область поиска -
                    p..r (автомат)
                    child.drawState();
                    break;
                }
            }
        }
    }
}
```

Приложение 5. Исходные коды интерфейса

```
package ru.ifmo.vizi.orderstatistics;

import ru.ifmo.vizi.base.*;
import ru.ifmo.vizi.base.ui.*;
import ru.ifmo.vizi.base.widgets.*;
import ru.ifmo.vizi.orderstatistics.widgets.*;
```

```
import java.awt.*;
import java.awt.event.*;
import java.util.Stack;
import java.util.Vector;

/**
 * OrderStatistics applet.
 *
 * @author  Aleksey Vladykin
 * @version 1.0
 */
public final class OrderStatisticsVisualizer
    extends Base
    implements AdjustmentListener, ItemListener
{

    /**
     * Algorithm id: order statistic.
     */
    private static final int ORDERSTATISTIC = 1;

    /**
     * Algorithm id: quantiles.
     */
    private static final int QUANTILES = 2;

    /**
     * OrderStatistics automata instance.
     */
    private final OrderStatistics auto;

    /**
     * OrderStatistics automata data.
     */
    private final OrderStatistics.Data data;

    /**
     * Copy of automata's array for save/load operations.
     */
    private int array[];

    /**
     * Array cells. Stack of {@link Rect}.
     */
    private final Stack cells;

    /**
     * Array cell styles.
     */
    private final ShapeStyle[] cellStyles;

    /**
     * Cell indices. Stack of {@link Rect}.
     */
    private final Stack indices;

    /**
     * Index styles.
     */
    private final ShapeStyle[] indexStyles;

    /**
     * Minimal index height.
     */
    private final int minIndexHeight;

    /**
     * Arrow pointing at elements being exchanged.
     */
    private final Arrow arrow;

    /**
     * Component displaying current value of OrderStatistic's <code>x</code>.
     */
    private final Rect watch;
```

```
/**
 * Message for {@link #watch}'s.
 */
private final String watchMsg;

/**
 * Maximal cell value.
 */
private final int maxElementValue;

/**
 * Choice with available algorithms.
 */
private final Choice algorithmChoice;

/**
 * Control adjusting algorithm parameter <code>k</code>.
 */
private final AdjustablePanel orderPanel;

/**
 * Number of array elements.
 */
private final AdjustablePanel elements;

/**
 * Save/load dialog.
 */
private SaveLoadDialog saveLoadDialog;

/**
 * Creates a new OrderStatistics visualizer.
 *
 * @param parameters visualizer parameters.
 */
public OrderStatisticsVisualizer(VisualizerParameters parameters) {
    super(parameters);
    auto = new OrderStatistics(locale);
    data = auto.d;
    data.visualizer = this;

    maxElementValue = config.getInteger("maxElementValue");

    cells = new Stack();
    cellStyles = ShapeStyle.loadStyleSet(config, "cellStyles");

    indices = new Stack();
    indexStyles = ShapeStyle.loadStyleSet(config, "indexStyles");
    minIndexHeight = config.getInteger("minIndexHeight");

    arrow = new Arrow(ShapeStyle.loadStyleSet(config, "arrowStyles"));
    clientPane.add(arrow);

    watch = new Rect(ShapeStyle.loadStyleSet(config, "watchStyles"));
    clientPane.add(watch);
    watchMsg = config.getParameter("watchMsg");

    elements = new AdjustablePanel(config, "elements");
    elements.addAdjustmentListener(this);

    setArraySize(elements.getValue());
    randomizeArray();
    syncArray();

    algorithmChoice = new Choice();
    algorithmChoice.addItemListener(this);
    orderPanel = new AdjustablePanel(config, "algorithm-k");
    createInterface(auto);

    saveLoadDialog = new SaveLoadDialog(config, forefather) {
        public boolean load(String text) throws Exception {
            SmartTokenizer t = new SmartTokenizer(text, config);
            t.expect("array");
            t.expect("=");
            Vector v = new Vector();
```

```
            for (int p = t.nextInt(0, maxElementValue); p > 0; p = t.nextInt(0,
                maxElementValue))
                v.add(new Integer(p));
            if (v.size() < elements.getMinimum() || v.size() > elements.getMaximum())
                throw new Exception(
                        I18n.message(config.getParameter("SaveLoadDialog-arraySizeError"),
                                new Integer(elements.getMinimum()),
                                new Integer(elements.getMaximum())));

            t.expect("algorithm");
            t.expect("=");
            String alg = t.nextWord();
            if (!alg.equalsIgnoreCase("ORDERSTATISTIC") &&
                !alg.equalsIgnoreCase("QUANTILES"))
                throw new
                    Exception(config.getParameter("SaveLoadDialog-algorithmNameError"));

            t.expect("k");
            t.expect("=");
            int k;
            if (alg.equalsIgnoreCase("ORDERSTATISTIC"))
                k = t.nextInt(0, v.size() - 1);
            else
                k = t.nextInt(2, (v.size() + 1) / 2);

            t.expect("step");
            t.expect("=");
            int step = t.nextInt(0, Integer.MAX_VALUE);
            t.expectEOF();

            setArraySize(v.size());
            for (int i = 0; i < v.size(); i++) {
                data.a[i] = ((Integer) v.elementAt(i)).intValue();
                array[i] = data.a[i];
            }
            data.algorithm = alg.equalsIgnoreCase("ORDERSTATISTIC")?
                    OrderStatistics.Data.ORDERSTATISTIC : OrderStatistics.Data.QUANTILES;
            data.k = k;
            auto.getController().rewind(step);

            return true;
        }
    };
}


/**
 * Creates panel with visualizer controls.
 *
 * @return controls pane.
 */
public Component createControlsPane() {
    Container panel = new Panel(new BorderLayout());

    panel.add(new AutoControlsPane(config, auto, forefather, false), BorderLayout.NORTH);

    Panel elementsPanel = new Panel();
    elementsPanel.add(elements);
    elementsPanel.add(
        new HintedButton(config, "button-random") {
            protected void click() {
                auto.getController().doRestart();
                randomizeArray();
                auto.drawState();
            }
        }
    );
    elementsPanel.add(new HintedButton(config, "button-saveLoad") {
        protected void click() {
            saveLoadDialog.center();
            StringBuffer buf = new StringBuffer();
            buf.append("/* ").append(config.getParameter("SaveLoadDialog-arrayComment"))
                    .append(" */\n");
            buf.append("array =");
            for (int i = 0; i < array.length; i++)
                buf.append(" ").append(array[i]);
```

```
            buf.append(" 0\n");
            buf.append("/* ").append(
                    config.getParameter("SaveLoadDialog-algorithmComment"))
                    .append(" */\n");
            buf.append("algorithm = ").append(
                    data.algorithm == OrderStatistics.Data.ORDERSTATISTIC?
                            "ORDERSTATISTIC" : "QUANTILES").append("\n");
            buf.append("/* ").append(config.getParameter("SaveLoadDialog-kComment"))
                    .append(" */\n");
            buf.append("k = ").append(data.k).append("\n");
            buf.append("/* ").append(config.getParameter("SaveLoadDialog-stepComment"))
                    .append(" */\n");
            buf.append("step = ").append(auto.getStep());
            saveLoadDialog.show(buf.toString());
        }
    });
    panel.add(elementsPanel, BorderLayout.SOUTH);

    Panel algorithmPanel = new Panel();
    algorithmPanel.add(new Label(config.getParameter("algorithm-prompt"), Label.RIGHT));
    algorithmChoice.addItem(config.getParameter("algorithm-orderStatistic"));
    algorithmChoice.addItem(config.getParameter("algorithm-quantiles"));
    algorithmPanel.add(algorithmChoice);
    adjustOrderBounds(ORDERSTATISTIC, elements.getValue());
    orderPanel.addAdjustmentListener(this);
    algorithmPanel.add(orderPanel);
    panel.add(algorithmPanel, BorderLayout.CENTER);

    return panel;
}

/**
 * Sets new array size.
 *
 * @param size new array size.
 */
private void setArraySize(int size) {
    elements.setValue(size);
    data.a = new int[size];
    array = new int[size];
    while (cells.size() < size) {
        Rect cell = new Rect(cellStyles);
        cells.push(cell);
        clientPane.add(cell);
        Rect index = new Rect(indexStyles);
        indices.push(index);
        index.setMessage(Integer.toString(indices.size()));
        clientPane.add(index);
    }
    while (cells.size() > size) {
        clientPane.remove((Component) cells.pop());
        clientPane.remove((Component) indices.pop());
    }
    clientPane.doLayout();
}

/**
 * Randomizes array cell values.
 */
private void randomizeArray() {
    for (int i = 0; i < data.a.length; i++) {
        data.a[i] = 1 + (int) (Math.random() * maxElementValue);
        array[i] = data.a[i];
    }
}

/**
 * Adjusts bounds for <code>k</code> (via {@link #orderPanel})
 * according to algorithm and array length.
 *
 * @param algorithm algorithm id.
 * @param n         array length.
 */
private void adjustOrderBounds(int algorithm, int n) {
    switch (algorithm) {
    case ORDERSTATISTIC:
```

```
            orderPanel.setMinimum(0);
            orderPanel.setMaximum(n - 1);
            break;
        case QUANTILES:
            orderPanel.setMinimum(2);
            orderPanel.setMaximum((n + 1) / 2);
            break;
        }
    }

    /**
     * Invoked on adjustment event.
     *
     * @param event event to process.
     */
    public void adjustmentValueChanged(AdjustmentEvent event) {
        if (event.getSource() == elements) {
            auto.getController().doRestart();
            setArraySize(event.getValue());
            randomizeArray();
            syncArray();
            adjustOrderBounds(algorithmChoice.getSelectedIndex() == 0?
                    ORDERSTATISTIC : QUANTILES, event.getValue());
            data.k = orderPanel.getValue();
            auto.drawState();
        } else if (event.getSource() == orderPanel) {
            auto.getController().doRestart();
            data.algorithm = algorithmChoice.getSelectedIndex() == 0?
                    OrderStatistics.Data.ORDERSTATISTIC : OrderStatistics.Data.QUANTILES;
            data.k = event.getValue();
            auto.drawState();
        }
    }


    /**
     * Invoked on item event.
     *
     * @param event event to process.
     */
    public void itemStateChanged(ItemEvent event) {
        if (event.getSource() == algorithmChoice) {
            auto.getController().doRestart();
            adjustOrderBounds(algorithmChoice.getSelectedIndex() == 0?
                    ORDERSTATISTIC : QUANTILES, elements.getValue());
            data.algorithm = algorithmChoice.getSelectedIndex() == 0?
                    OrderStatistics.Data.ORDERSTATISTIC : OrderStatistics.Data.QUANTILES;
            data.k = orderPanel.getValue();
            auto.drawState();
        }
    }


    /**
     * Invoked when client pane shoud be laid out.
     *
     * @param clientWidth client pane width.
     * @param clientHeight client pane height.
     */
    protected void layoutClientPane(int clientWidth, int clientHeight) {
        int n = cells.size();
        int size = Math.min(clientWidth / (n + 1), 2 * clientHeight / 5);
        int y = (clientHeight - size) / 2;
        int x = (clientWidth - size * n) / 2;
        String maxValueString = Integer.toString(maxElementValue);
        for (int i = 0; i < n; i++) {
            Rect cell = (Rect) cells.elementAt(i);
            cell.setBounds(x + i * size, y, size + 1, size + 1);
            cell.adjustFontSize(maxValueString);
            Rect index = (Rect) indices.elementAt(i);
            index.setBounds(x + i * size, y + size, size + 1,
                    Math.max(size / 3, minIndexHeight));
            index.adjustFontSize("i-1, j+1");
        }
        Rect cell = (Rect) cells.elementAt(0);
        int watchHeight = Math.min(5 * size / 12, clientHeight / 5);
```

```
        watch.setBounds(cell.getX(), cell.getY() - 3 * size / 4, n * size + 1, watchHeight);
        watch.adjustFontSize(I18n.message(watchMsg, new Integer(maxElementValue)));
        if (arrow.isVisible())
            updateExchangeArrow(data.OrderStatistic_i, data.OrderStatistic_j);
    }

    /**
     * Synchronizes cell contents with actual automata's array.
     */
    public void syncArray() {
        for (int i = 0; i < cells.size(); i++) {
            Rect cell = (Rect) cells.elementAt(i);
            cell.setMessage(Integer.toString(data.a[i]));
        }
    }

    /**
     * Resets all cells' styles to 0 (normal style).
     */
    public void resetCellStyles() {
        for (int i = 0; i < cells.size(); i++) {
            Rect cell = (Rect) cells.elementAt(i);
            cell.setStyle(0);
        }
    }

    /**
     * Sets cells with given numbers to new style.
     *
     * @param numbers   array with cell indices.
     * @param style     new style to set.
     */
    public void setCellStyles(int numbers[], int style) {
        for (int i = 0; i < numbers.length; i++)
            if (0 <= numbers[i] && numbers[i] < cells.size()) {
                Rect cell = (Rect) cells.elementAt(numbers[i]);
                cell.setStyle(style);
            }
    }

    /**
     * Dims cells in given range.
     *
     * @param from  first cell index.
     * @param to    last cell index.
     */
    public void dimCells(int from, int to) {
        for (int i = from; i <= to; i++) {
            Rect cell = (Rect) cells.elementAt(i);
            cell.setStyle(3 + cell.getStyle() % 3);
        }
    }

    /**
     * Reset indices to cell numbers.
     */
    public void resetIndices() {
        for (int i = 0; i < indices.size(); i++) {
            Rect index = (Rect) indices.elementAt(i);
            index.setMessage(Integer.toString(i));
            index.setStyle(0);
        }
    }

    /**
     * Sets <code>i</code>-th index to given message.
     * If <code>i</code>-th index is not numeric, appends given message.
     *
     * @param i     index.
     * @param msg   message to set.
     */
    public void updateIndex(int i, String msg) {
        if (i < 0 || cells.size() <= i)
            return;
        Rect index = (Rect) indices.elementAt(i);
        try {
```

```
            Integer.valueOf(index.getMessage());
            index.setMessage(msg);
        } catch (NumberFormatException e) {
            index.setMessage(index.getMessage() + ", " + msg);
        }
        index.setStyle(1);
    }

    /**
     * Updates arrow position to connect <code>i</code>-th and
     * <code>j</code>-th cells. If <code>i</code> or <code>j</code>
     * is out of reasonable range, hides the arrow.
     *
     * @param i first cell.
     * @param j second cell.
     */
    public void updateExchangeArrow(int i, int j) {
        if (i < 0 || cells.size() <= i || j < 0 || cells.size() <= j) {
            arrow.setVisible(false);
            return;
        }
        if (i == j) {
            Rect c = (Rect) cells.elementAt(i);
            arrow.setTipSize(c.getWidth() / 15);
            int x = c.getX() + c.getWidth() / 3 - arrow.getTipSize();
            int y = c.getY() - c.getHeight() / 3;
            int w = c.getWidth() / 3 + arrow.getTipSize();
            int h = c.getY() - 1 - y;
            arrow.setBounds(x, y, w, h);
        } else {
            Rect c1 = (Rect) cells.elementAt(i);
            Rect c2 = (Rect) cells.elementAt(j);
            arrow.setTipSize(c1.getWidth() / 15);
            int x = c1.getX() + c1.getWidth() / 2 - arrow.getTipSize();
            int y = c1.getY() - c1.getHeight() / 3;
            int w = c2.getX() + c2.getWidth() / 2 + arrow.getTipSize() - x;
            int h = c2.getY() - 1 - y;
            arrow.setBounds(x, y, w, h);
        }
        arrow.setVisible(true);
    }

    /**
     * Updates {@link #watch} to display current <code>x</code> value.
     * If <code>x&lt;=0</code> - hides it.
     *
     * @param x <code>x</code> value.
     */
    public void updateWatch(int x) {
        if (x <= 0)
            watch.setVisible(false);
        else {
            watch.setMessage(I18n.message(watchMsg, new Integer(x)));
            watch.setVisible(true);
        }
    }

}
```