ВИЗУАЛИЗАЦИЯ МЕТОДОВ СОРТИРОВКИ МАССИВОВ

С.И. Гроздев, Т.Ж. Терзиева
Республика Болгария, г. София

Одним из важнейших дидактических принципов, который лежит в основе организации и эффективности обучения и восприятия, является наглядность. Его использование дает возможность для преодоления языковых неравенств у обучаемых, для развития правого полушария головного мозга, а также создает условия для синхронизации работы левого и правого полушария. Не случайно чешский педагог Я. Коменский, основоположник использования наглядности в образовании, определяет ее как «золотое правило дидактики».

Принцип наглядности является основным и в обучении по информатике. Новые информационные технологии дают возможность создать динамические учебные среды, а представление процессов и явлений в движении и развитии имеет особое значение для познавательной деятельности школьников [1]. Анимации алгоритмов разрабатываются еще с 1980 года, как способ для абстрактного графического представления части кода, что является полезной информацией о результатах каждого этапа исполнения программы. Практика показывает, что использование анимации для объяснения и анализа алгоритмов помогает их пониманию и запоминанию. Исполнение во времени прослеживает существенные изменения состояния процесса. С другой стороны, статичные визуализации позволяют фиксирование идеи как нечто целое, и одновременно как наблюдение некоторых деталей и др. Разделяем мнение, что сочетание статичной и динамической визуализации изучаемых понятий и алгоритмов в обучении по информатике обеспечивает оптимальные условия для их формирования.

В настоящей статье представляем статичные и динамичные инструменты для визуализации методов сортировки массивов.

По Кормену неформально алгоритм представляет произвольную корректно определенную вычислительную процедуру, на входе которой

подается какая-либо величина или набор величин, и результатом ее исполнения является выходная величина или набор величин [3]. Таким образом алгоритм рассматривается как последовательность вычислительных шагов, преобразующих исходные величины в выходные. Одной из наиболее часто возникающий задачи в практике является задача сортировки (Sorting problem). Сортировка рассматривается как процесс упорядочивания данного множества объектов в определенном порядке. Цель сортировки – упростить последующий поиск элементов в уже отсортированном множестве [7].

Сортировку можно определить формально следующим образом: [3]

Вход: Последовательность n чисел: $a_1; a_2; ...; a_n$.

Выход: Изменение входной последовательности таким образом $a_1'; a_2'; ...; a_n'$, что для ее членов выполнено соотношение $a_1' \le a_2' \le ... \le a_n'$.

Входная последовательность обычно выглядит как одномерный массив с n элементами, хотя иногда может иметь и другой вид.

Так как в информатике сортировка – основная операция, появилось много алгоритмов для сортировки. Выбор подходящего алгоритма зависит от многих факторов, среди которых число элементов для сортировки, порядок во входной последовательности, возможные ограничения на входные элементы, а также и вид устройства (памяти) для сохранения элементов. Алгоритм будет считаться корректным, если для всех входных данных получаются корректные выходные данные.

Одним из важных моментов при выборе компьютерного алгоритма является оценка его эффективности. Разные алгоритмы могут решать одну и ту же задачу (с одинаковым набором входных данных), но нужно выбрать какой из них наиболее эффективен относительно времени исполнения и (или) пространства (памяти). На практике преобладает игнорирование критерия для памяти, а учитывается время для исполнения.

Мерой определения эффективности могут быть C – необходимое число сравнений элементов и M – число присвоений (обменов) элементов [7]. Эти числа являются функциями числа n элементов для сортировки в массиве.

Хорошие алгоритмы для сортировки обеспечивают сравнения порядка $n * \log n$. Простые методы, которые дают сравнения порядка n^2, называются прямыми методами. По принципу работы прямые методы можно разделить на:

- сортировка с использованием вставки (insertion)
- сортировка с использованием выбора (selection)
- сортировка с использованием смен (exchange).

В рассмотренных программах, которые иллюстрируют отдельные методы, элементы массива – действительные числа. Однако на практике они представляют структуры и только одно поле из этих структур, с названием ключ, является определяющим для сортировки.

МЕТОД ПРОСТОЙ ВСТАВКИ

Элементы данного массива $a_1; a_2; ...; a_n$ условно делятся на две последовательности. Первая последовательность $a_1; a_2; ...; a_{i-1}$ состоит из уже упорядоченных элементов, вторая последовательность $a_i; a_{i+1}; ...; a_n$ состоит из неупорядоченных (входящих) элементов, $i = 1; \ldots; n$. На каждом шагу i-ый элемент из первоначальной последовательности перемещается на подходящее место в уже упорядоченную последовательность $a_1; a_2; ...; a_{i-1}$.

Начальное значение i равно 2.

Алгоритм находит подходящее место для вставки i- го элемента, сравнивая его последовательно справа налево с элементами последовательности. Если элемент последовательности больше, он перемещается на одну позицию вправо, i- ый элемент занимает его место и сравнение продолжается со следующим элементом последовательности.

Вставка на каждом шагу может закончиться при одном из двух различных условий:

- найден элемент $a_j < x$.
- достигнут левый конец массива.

В этом методе применяется так называемый прием фиктивного элемента (барьера). В данном случае его можно легко описать, полагая $a_0 = x$.

Существует много систем для визуализации алгоритмов, которые используются в обучении по информатике. Часть из них [3], [7], [13], [14], [19], представлены на рис. 1, показывают действие алгоритма в виде схем, таблиц и словесных описаний. У этих пособий для обучения есть неоспоримые преимущества, когда необходимо поэтапное формирование понятий и демонстрация порядка при выполнении данного алгоритма. Кроме того можно проследить целиком алгоритм путем визуализации всех промежуточных результатов при сортировке.

При этом выделяется графически:

- отсортированная (уже упорядоченная) часть массива;
- неотсортированная часть массива;
- очередной элемент, который ставится на соответствующее место;
- каждый шаг в процессе исполнения алгоритма.

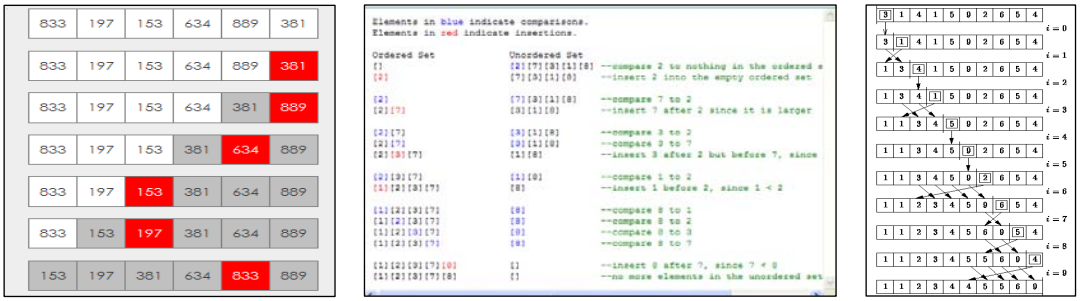

Рис. 1. Статические пособия для наглядности

Статические средства обучения однако не дают возможность следить за динамикой развития процесса. На рис. 2. показаны некоторые возможности для динамической визуализации [8], [15], [17]. Они представляют алгоритм через анимацию, тем самым создавая возможность для:

- прослеживания процесса шаг за шагом;
- визуализирования упорядоченных элементов;
- визуализирования неупорядоченных элементов;

- остановки процесса;
- возврата к началу наблюдения.

Представленные на рис. 3. анимации [10], [11], [18], кроме перечисленных выше преимуществ, показывают число сравнений и перестановок элементов.

Все анимированные визуализации показывают алгоритм сортировки в развитии, четко прослеживается изменение на каждом шагу процесса. Но результаты предыдущего действия не сохраняются и нет возможности увидеть весь алгоритм, а только текущее и конечное состояние.

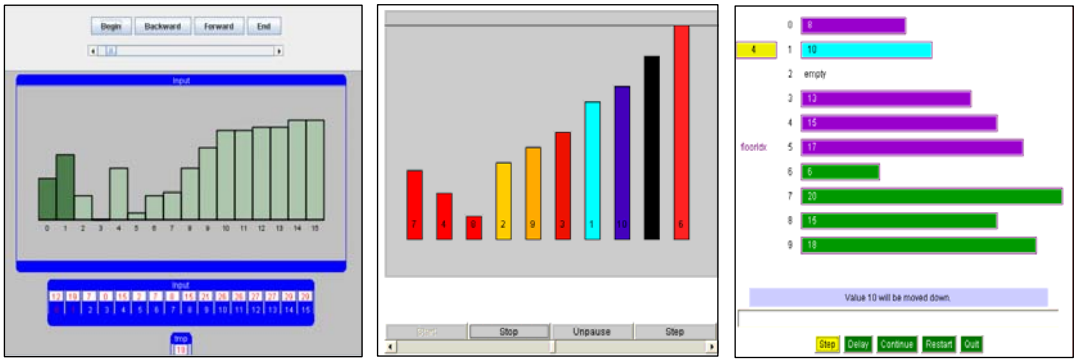

Рис. 2. Динамические средства

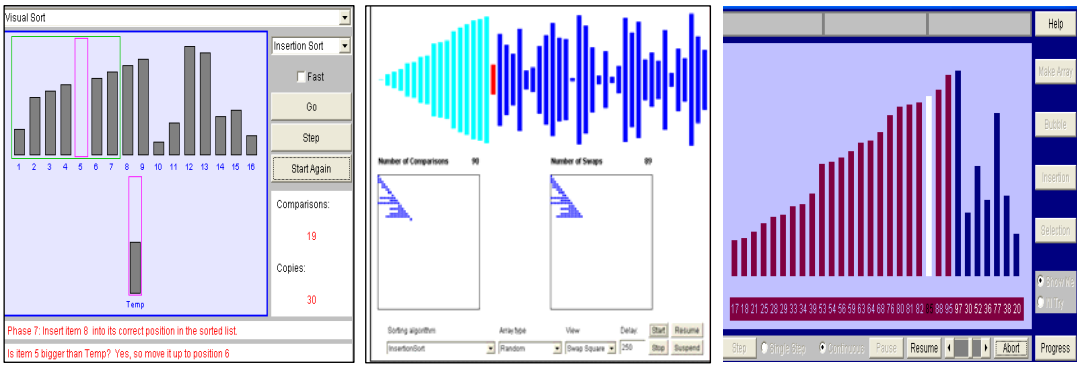

Рис. 3. Динамические средства с числом сравнений и перестановок

На рис. 4 и рис. 5 показана сортировка вставками; в левой части находится программный код алгоритма, реализованного на *C++*. Пример на рис. 4 [12] дает возможность для произвольного введения чисел, количество

которых не превосходит 25. Кроме того можно менять скорость выполнения алгоритма.

Пример на рис. 5 – для семи случайно выбранных целых чисел.

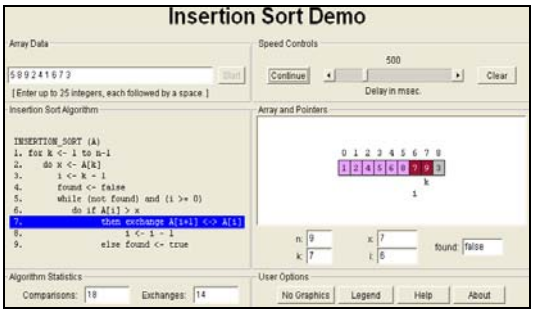

Рис. 4.

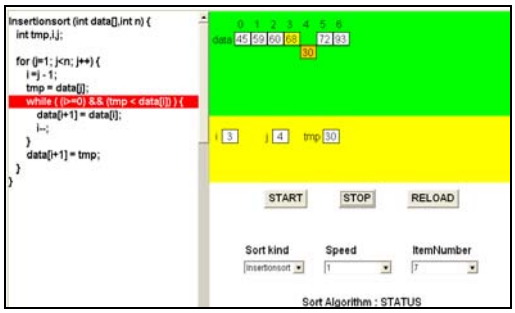

Рис. 5.

Сортировка прямыми вставками

Преимущество этого способа визуализации состоит в том, что в левой части показан алгоритм сортировки, реализованный на *C++*. На каждом шагу исполнения алгоритма можно проследить за значениями управляющих переменных i, j и переменной *tmp*, которая используется для перестановки местами, как и за числом упорядоченных элементов при каждом проходе через цикл. В случае необходимости можно воспользоваться кнопкой *stop*. Недостатком в данном случае является то, что нельзя вводить значения элементов массива, так как они выбираются случайно. Однако есть возможность для выбора количества чисел – с 3 до 16.

СОРТИРОВКА ПУЗЫРЬКОМ

Массив рассматривается вертикально, а не горизонтально. Можно представлять себе элементы в виде «пузырьков», расположенных в резервуаре с водой, с «весами», равными их значениям. Каждый проход массива будет передвигать пузырек к «всплыванию» на место, соответствующее его «весу».

Алгоритм сравнивает каждые два соседних элемента и при необходимости меняет их местами. Для упорядочивания массива с n элементами достаточно $n-1$ проходов. На $i-$том проходе массив проверяется с

первого до $n-1$ элемента. Видно, что при каждом проходе наибольший («самый тяжелый») элемент перемещается в конец («тонет»), а «самый легкий» «поднимается» (перемещается вперед) на один уровень. В худшем случае (когда наименьший элемент является последним) необходимы $n-1$ прохода массива.

На рис. 6 показан метод сортировки пузырьком для 10 произвольно выбранных целых чисел.

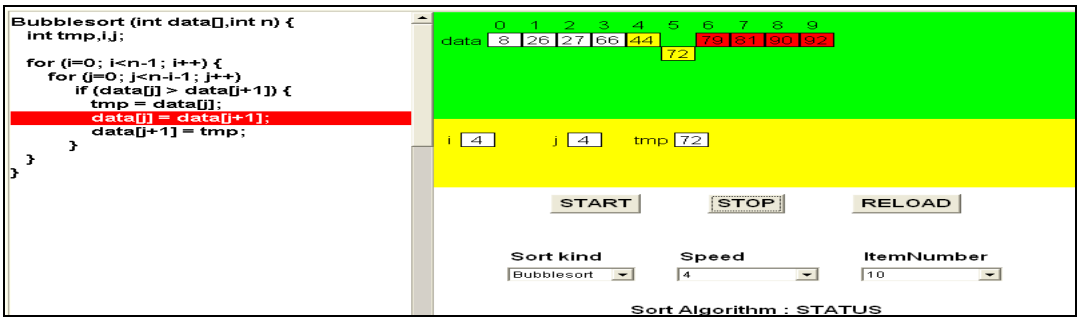

Рис. 6. Сортировка пузырьком

СОРТИРОВКА ВЫБОРОМ

В массиве $a_1; a_2; ...; a_n$ выбирается минимальный элемент и меняется местами с первым. В остатке $a_2; ...; a_n$ выбирается минимальный элемент и меняется местами с a_2 и т.д. В общем случае, на i-ом проходе находится минимальный элемент в последовательности $a_i; a_{i+1}; ...; a_n$ и меняется местами с a_i. В конце остается максимальный элемент, который уже упорядочен. В предложенной программе (во внутреннем цикле) в неупорядоченном остатке массива находится минимальный элемент и запоминается его индекс. После окончания внутреннего цикла производится обмен с первым элементом остатка.

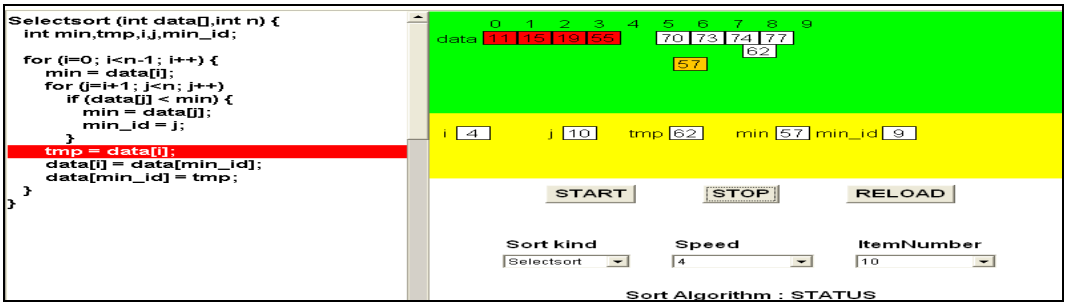

Рис. 7. Сортировка прямым выбором

БЫСТРАЯ СОРТИРОВКА (QUICK SORT)

Метод предложен К. Хоором в 1962 г. и является одним из наиболее эффективных (из самых быстрых) методов сортировки. Быстрая сортировка основывается на том, что для достижения высокой эффективности лучше всего менять местами элементы, находящиеся на больших расстояниях друг от друга.

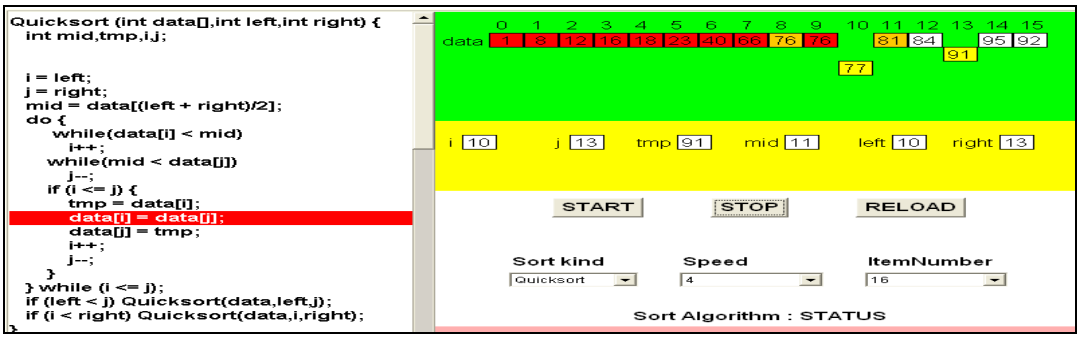

Рис. 8. Быстрая сортировка

Будем производить сортировку в возрастающем порядке. Выбирается случайный элемент x из массива (для определенности можно выбрать средний элемент массива). Проход массива производится слева направо до нахождения элемента, большего или равного x. После этого массив проходится справа налево до нахождения элемента, меньшего или равного x (равенство необходимо, чтобы можно было использовать x в качестве «барьера» при проходе массива). Оба элемента меняются местами. Аналогичным образом процесс «прохода с обменом» продолжается до тех пор, пока проходы массива слева и справа не «встретятся» где-то в массиве. В результате получается

разделение массива на две части – левая состоит только из элементов, меньших или равных x, а правая – из элементов, больших или равных x (в общем случае эти две части не равны между собой). Каждая из двух частей массива сортируется отдельно (сортировка каждой части массива может быть выполнена рекурсивным применением процедуры сортировки для соответствующей части массива).

Использованные анимации находятся по указанному адресу [15].

Моделирование представленных алгоритмов дает обучаемым возможность с помощью интуитивного и легко понятного интерфейса следить за действием каждого алгоритма шаг за шагом с начала и до конца. Предоставляется и возможность выбора числа элементов для сортировки (минимум 3, максимум 16), скорость процесса тоже можно изменять. Элементы, окрашенные в красный цвет, уже упорядочены. Элементы, которые меняются местами в данный момент, визуализируются в желтом цвете, и в каждый момент отслеживаются значения основных переменных, которые используются в алгоритмах. Исключительным преимуществом является код программы, расположенный в левой части экрана. Это дает возможность сделать параллель между реализованным кодом и технологическим упорядочиванием элементов шаг за шагом. Визуализация создает возможность для:

- графического и алгоритмического разъяснения действия алгоритмов для конкретных входных данных, причем их число можно менять;
- пошаговое исполнение алгоритма;
- отслеживания значений основных управляющих переменных цикла;
- отслеживания промежуточных результатов;
- изменения скорости исполнения алгоритма;
- отслеживания числа упорядоченных элементов после каждого исполнения основных действий;
- проверки действия алгоритма для разных входных данных;
- отслеживания алгоритма в действии.

Но этот тип визуализации не дает возможность сравнивать указанные алгоритмы. Каждый алгоритм рассматривается самостоятельно.

Если нужно сделать анализ и сравнение разных алгоритмов сортировки, можно воспользоваться следующим адресом: http://www.sorting-algorithms.com/.

На рис. 9 представлены восемь разных алгоритмов сортировки при четырех разных начальных условиях. Эти визуализации дают возможность:

- отслеживать действие каждого алгоритма;
- отслеживать преимущества и недостатки каждого алгоритма при разных входных данных;
- осознать, что определение «лучшего» алгоритма неоднозначно, а зависит от разных условий;
- сравнивать и анализировать разные алгоритмы;
- отслеживать действие данного алгоритма при разном порядке входных данных;
- показать, что начальное состояние (порядок введения и распределения элементов) влияет на исполнение и в зависимости от него нужно предпочесть тот или другой алгоритм;
- менять число сортированных элементов.

Рис. 9. Алгоритмы сортировки

В случае почти сортированных элементов на Рис. 9 видно, что метод сортировки пузырьком и метод простой вставки являются быстрыми, но при

случайном расположении элементов видна большая разница в скорости исполнения между простыми прямыми методами с вычислительной сложностью n^2 и более усовершенствованными методами.

На рис. 10 и рис. 11 представленны возможности для отслеживания эффективность алгоритмов при разных начальных условиях.

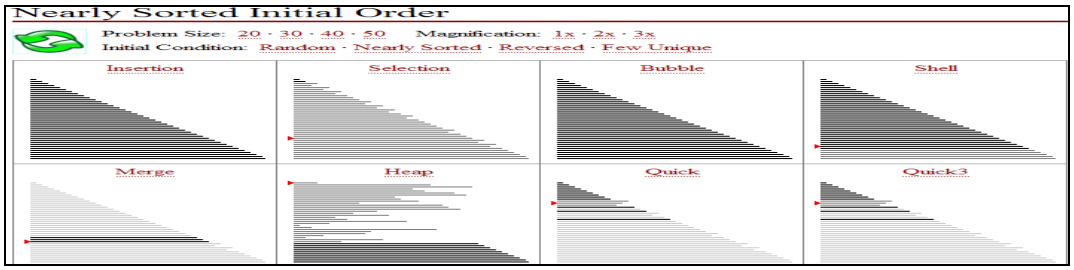

Рис. 10. Почти отсортированный исходный порядок элементов

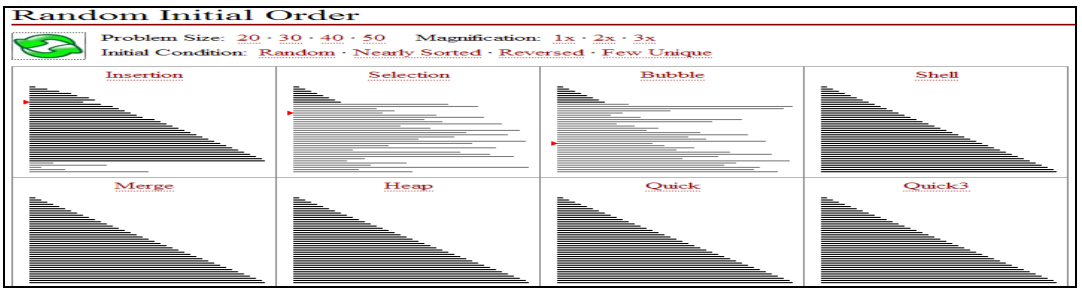

Рис. 11. Случайный исходный порядок элементов

Кроме представленных возможностей для сравнения возможно:

- сравнение алгоритмов по разным признакам;
- отслеживание действия разных алгоритмов при одинаковых начальных условиях (рис.12-14);
- каждый алгоритм можно отслеживать самостоятельно;
- демонстрировать действие данного алгоритма при разных начальных условиях;
- осуществить визуализацию программного кода алгоритма.

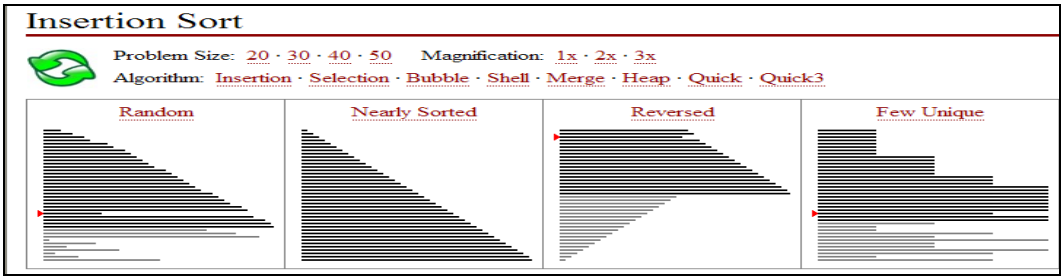

Рис. 12. Сортировка простой вставкой при разном исходном порядке элементов

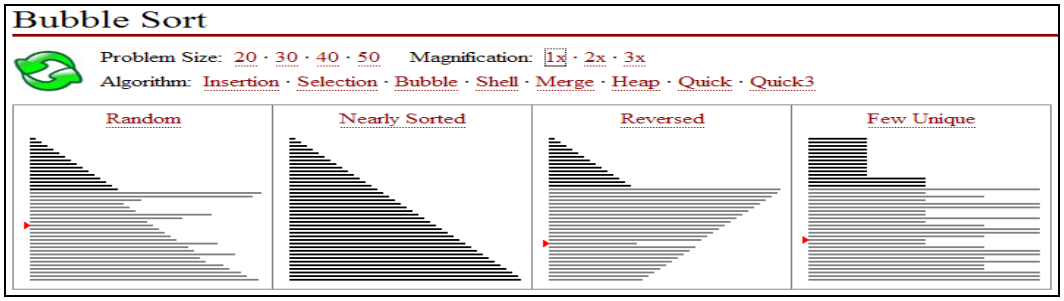

Рис. 13. Метод сортировки пузырьком
при разном исходном порядке элементов

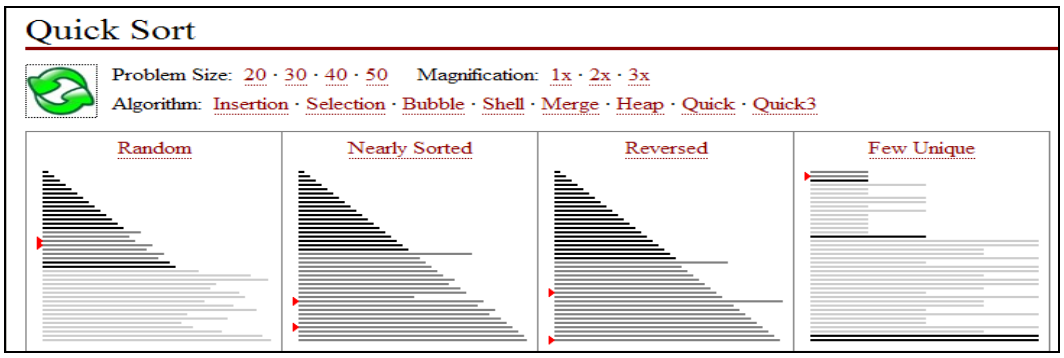

Рис. 14. Быстрая сортировка при разном исходном порядке элементов

Усовершенствование методики преподавания программирования является актуальной задачей исключительной важности как для средней школы, так и для высшей. Одна из возможностей для повышения качества усвоения учебного материала состоит в использовании анимации в преподавании программирования и в демонстрации исполнения алгоритмов. В

компьютерной анимации можно выделить две основные цели как инструментальное средство обучения [5]:

- предоставляет конкретные изображения абстракций и операций, присущих данному алгоритму или программе;
- представляет динамику развивающегося во времени процесса.

При применении визуализации в преподавании алгоритмов для сортировки реализуется наблюдение исполнения алгоритмов шаг за шагом. После этого возможны повторение, анализ и систематизирование полученных результатов. Понимание алгоритмов можно рассматривать в двух аспектах:

- Процессуальное понимание – способность прослеживать шаг за шагом поведение алгоритма в новой ситуации;
- Концептуальное понимание – возможность ответить на некоторые вопросы, связанные с приложением алгоритма при специальном наборе входных данных.

В современной дидактике принцип наглядности воспринимается как систематическая основа не только для конкретных предметов и их изображений, но и их моделей [2]. Важной дидактической особенностью мультимедийных средств обучения является возможность глубокого проникновения в сущность изучаемых явлений и процессов. Их специфическая особенность состоит в том, что они показывают эти явления и процессы в развитии и динамике. Представленная таким образом учебная информация обеспечивает более сильное эмоциональное воздействие на обучаемых, причем:

- дает полную и точную информацию об изучаемых объектах и таким образом способствует повышению качества обучения;
- полнее отвечает на вопросы и удовлетворяет естественной любознательности обучаемых;
- предоставляет возможность для критического осмысливания представляемой информации.

Отметим еще, что перед тем как стать средством для использования, визуализация являются объектом как исследования, так и преподавания. Для учителей важно знать существующие дидактические наглядные средства и

следить за появляющимися новыми. Тогда в зависимости от конкретных условий они смогут выбрать те из них, которые наиболее удобны для их работы. [1]

Литература

1. Андреев М. Процесът на обучението. Дидактика. София: Университетско издателство «Св. Климент Охридски», 1996. 420 с.

2. Прессман Л.П. Методика применения технических средств обучения: экранно-звуковые средства. М.: Просвещение, 1989. 191 с.

3. Cormen, T. H., C.E. Leiserson, R.L. Rivest, and C. Stein: Introduction to Algorithms, Cambridge, MA: MIT Press, 1990 – 984 c.

4. Grozdev, S. On the Visualness in Mathematics Education, Proc. 4th Mediterranean Conference on Mathematics Education, Palermo – Italy, 28 – 30 Jan. 2005, pp. 303-313.

5. Michael D. Byrn, Richard Catrambone, John T. Stasko: Do Algorithm Animations Aid Learning? Technical ReportGIT-GVU-96-18, Graphics, Visualization, 1996 – Citeseer.

6. Niklaus Wirt. Algoritms and Data Structures = Programs, Prentice-Hall, 1986. http://corewar.co.uk/assembly/insertion.htm

7. Animations for Data Structures and Algorithms http://www.csse.monash.edu.au/~dwa/Animations/index.html

8. CS 162 Program Demonstration. http://web.engr.oregonstate.edu/~minoura/cs162/javaProgs/sort/InsertSort.htm

9. Interactive data structure visualizations. http://nova.umuc.edu/~jarc/idsv/

10. http://www.cse.iitk.ac.in/users/dsrkg/cs210/html/sortingpage.html

11. http://www.brian-borowski.com/Software/Sorting/

12. Data Structures and Algorithms with Object-Oriented Design Patterns in C++ http://brpreiss.com/books/opus4/html/page487.html

[1] Статья была поддержана проектом НИ11-ФМИ-004 к НПД на ПУ.

13. MyCSTutorials.com Articles Database. http://www.mycstutorials.com/articles/sorting/insertionsort

14. Sorting Algorithm Animations http://www.sorting-algorithms.com/

15. Sorting Algorithms. http://maven.smith.edu/~thiebaut/java/sort/demo.html

16. The xSortLab Applet http://math.hws.edu/TMCM/java/xSortLab/index.html

17. Trakla, http://www.cs.hut.fi/Research/TRAKLA2/