СБОРНИК **НАУЧНЫХ ТРУДОВ НГТУ. – 2020. – № 4 (99).** – 59–70

СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.4'242 DOI: 10.17212/2307-6879-2020-4-59-70

ТЕХНОЛОГИЯ ПРОЕКТИРОВАНИЯ ВИЗУАЛИЗАТОРОВ АЛГОРИТМОВ*

Е.Л. РОМАНОВ[1], Т.А. РОМАНЕНКО[2]

[1] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, доцент кафедры вычислительной техники. E-mail: romanov@corp.nstu.ru*
[2] *630073, РФ, г. Новосибирск, пр. Карла Маркса, 20, Новосибирский государственный технический университет, старший преподаватель кафедры вычислительной техники. E-mail: romanenko@corp.nstu.ru*

Тема статьи относится к внедрению в систему дистанционного образования НГТУ таких форм обучения, как визуализаторы алгоритмов.

Ключевые слова: алгоритм, визуализатор, тренажер, автоматное программирование, автомат Мили, автомат Мура, автомат состояний

ВВЕДЕНИЕ

Для студентов факультета АВТФ и Института дистанционного образования НГТУ в рамках системы дистанционного обучения Dispace и ИДО НГТУ используется электронное учебное пособие «Структуры и алгоритмы обработки данных» [1].

Для пособия разработаны визуализаторы динамики работы наиболее сложных по логике и структуре фундаментальных алгоритмов:

– визуализаторы алгоритмов сортировки (пирамидальной сортировки, сортировки с помощью дерева выбора, сортировки разделением, сортировки слиянием, поразрядной сортировки);

– визуализаторы для сбалансированных деревьев поиска (рандомизированного дерева, АВЛ-дерева, красно-черного дерева, 2-3-дерева), внешнего B-дерева поиска;

* Статья получена 04 ноября 2020 г.

– визуализатор алгоритма формирования оптимального бинарного дерева поиска;

– визуализатор алгоритмов для хэш-таблиц с открытой адресацией;

– визуализаторы для алгоритмов на графах (обобщенного поиска на графе, алгоритмов Дейкстры, Джонсона, Флойда для построения кратчайших путей).

1. ОПИСАНИЕ ЗАДАЧИ

По ходу разработки и апробации этой серии визуализаторов формировались требования к функциональности, общие подходы к проектированию программных визуализаторов.

Представленные визуализаторы делятся на две группы – демонстрационные и тренажеры.

Демонстрационные визуализаторы предназначены для наблюдения на экране монитора анимированных действий алгоритма. Вмешательство наблюдателя минимально и сводится в основном к заданию алгоритму исходных данных. Возможны два режима анимации – автоматический под управлением программы визуализатора и пошаговый под управлением наблюдателя. Возможен вывод подстрочных комментариев о действиях алгоритма по ходу анимации его действий. Также используется отдельная справочная подсистема визуализатора с описанием алгоритма.

Визуализаторы-тренажеры в дополнение к демонстрации работы алгоритма предоставляют наблюдателю управление шагами алгоритма с помощью панели инструментов, оценивают ходы наблюдателя, сообщают об ошибочных ходах, позволяют выполнять откат назад путем отмены последних ошибочных действий.

Последние по хронологии разработки универсальные визуализаторы совмещают в своей функциональности оба режима – демонстрации и тренажера.

На рис. 1 показана типовая функциональность визуализаторов, предоставляемая для изучения алгоритмов.

Для реализации визуализаторов используется технология Java Applets, которая предоставляет максимальные преимущества для реализации приложений в Интернете [2]. Программы визуализаторов в виде отдельных модулей java-апплетов включаются в раздел визуализаторов электронного учебного пособия «Структуры и алгоритмы обработки данных» [1].

Традиционно при проектировании и программировании выделяются несколько событийно-управляемых подсистем визуализатора:

- подсистема взаимодействия с пользователем;
- подсистема отображения данных;

- подсистема демонстрации работы алгоритма;
- подсистема тренажера.

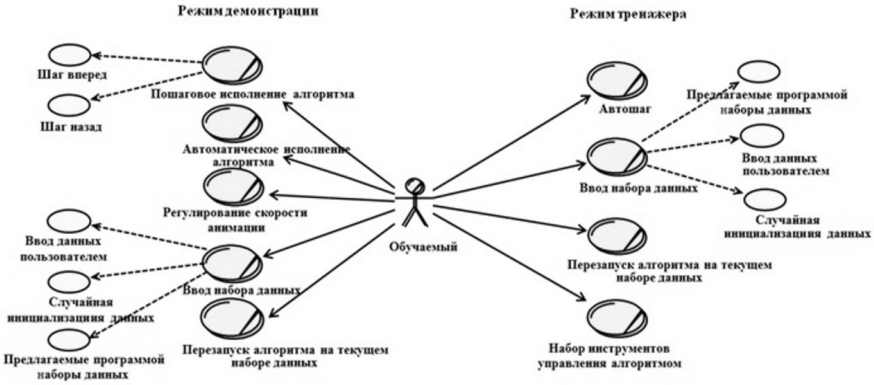

Рис. 1. Функциональная диаграмма визуализатора

Подсистемы со своими специфическими функциями проектируется в виде совокупности классов. Объекты подсистем и сами подсистемы взаимодействуют через механизм обработки событий (выбор команды или инструмента пользователем, выделение мышью фрагментов изображения на экране, таймер и т. п.).

Для сложных алгоритмов с запутанным ходом выполнения применение событийно-ориентированного программирования является обычным. Проектирование обработки событий, последовательности и взаимосвязи событий зачастую носит эвристический характер, приводит к громоздкой схеме взаимодействий обработчиков событий, к проверке многочисленных флагов для восстановления предыстории событий.

В этом случае целесообразно применить другую парадигму программирования – автоматное программирование [3]. Авторы статьи «Использование автоматного подхода для реализации вычислительных алгоритмов» [4] предложили использовать особенности автоматных программ для построения визуализаторов. В состав визуализатора вводится подсистема логики, управляющая с помощью автоматной модели передвижением по алгоритму и визуализацией данных и состояния алгоритма.

В качестве примера автоматного подхода к проектированию представлен фрагмент визуализатора для Б-дерева поиска – визуализация алгоритма вставки [6].

Визуализатор включает 4 модуля (рис. 2).

1. **Модуль интерфейса** – включает набор инструментов, через которые пользователь может формировать исходную структуру Б-дерева, выбирать для выполнения операцию, просматривать ее работу в режиме демонстрации, продвигаться по алгоритму по шагам.

2. **Автомат** – управляет пошаговым передвижением по алгоритму с помощью автоматной модели.

3. **B-дерево** – реализует структуру Б-дерева и выполняет алгоритм вставки (шаги), задаваемые автоматом, или команду управления визуализатором.

4. **Отображение** – отвечает за отображение структуры Б-дерева и иллюстрацию шагов алгоритма.

Рис. 2. Структурная схема визуализатора

В основе автомата для пошагового выполнения лежит логика «интересных» состояний, в которых информация об алгоритме и данных отображается пользователю.

Разработка автоматной модели алгоритма сводится к следующим шагам.

1. По визуализируемому алгоритму пишется процедурная программа, его реализующая.

2. Используемые переменные выносятся в модель данных (структуру данных). Передача параметров процедурам и результатов процедур осуществляется через модель.

3. Программа преобразуется с целью использования в ней следующих операторов:

- оператора присваивания,
- последовательности операторов (составного оператора),
- укороченного оператора ветвления (if-then),
- полного оператора ветвления (if-then-else),
- цикла с предусловием (while),
- вызова процедуры.

4. Действия программы преобразуются в систему взаимосвязанных состояний конечного автомата.

Примером проектирования служит автомат для алгоритма вставки элемента в Б-дерево [5].

Б-дерево – сбалансированное дерево, удобное для хранения информации во внешней (дисковой) памяти. Характерным для Б-дерева является его малая высота. Представление информации во внешней памяти в виде Б-дерева стало стандартным способом в системах баз данных.

В Б-дереве узел может иметь много сыновей. Количество сыновей (максимальное) определяет степень Б-дерева. Узлы Б-дерева содержат не один элемент, а группу элементов с суммарным объемом в один блок файла, который обычно по размеру соответствует одному сектору диска. Типичная степень узла t –50…2000 сыновей.

Например, Б-дерево степени 1000 и высоты 2 может хранить более миллиарда ключей.

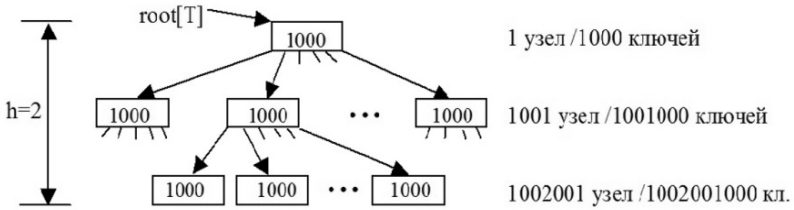

Рис. 3. Структура Б-дерева

Модель данных визуализатора представляет собой связную структуру узлов Б-дерева, в которой узлы дерева размещены в оперативной памяти и связаны между собой адресными указателями, степень узлов t – от двух до четырех сыновей. Отсутствуют данные, связанные с ключами, файловые указатели на блоки во внешней памяти. Отдельный узел $\boldsymbol{x}$ в модели Б-дерева включает следующие переменные:

leaf[$\boldsymbol{x}$] =TRUE, если $\boldsymbol{x}$ – лист дерева, FALSE – иначе;
n[$\boldsymbol{x}$] – количество ключей, хранящихся в $\boldsymbol{x}$;
$key_1[\boldsymbol{x}]$, $key_2[\boldsymbol{x}]$, … $key_n[\boldsymbol{x}]$ – массив ключей, хранящихся в $\boldsymbol{x}$;
$c_1[\boldsymbol{x}]$, $c_2[\boldsymbol{x}]$, … $c_{n+1}[\boldsymbol{x}]$ – массив указателей на дочерние узлы.

Программа вставки в Б-дерево, включающая три процедуры, размещена в модуле «Алгоритмы Б-дерева». Она составлена в соответствии со следующим псевдокодом алгоритма [5]:

Алгоритм вставки в Б-**дерево**

B_Tree_Insert (T, k) ▷ **T –** структура модели Б-дерева, **k –** ключ нового элемента

```
1.  r ← root [T]
2.  if  n [r] = 2t –1
3.      then s ← Allocate_Node( )          ▷ создание нового корня s
4.           root [T] ← s
5.           leaf [s] ← FALSE
6.           n [s] ← 0
7.           c_i [s] ← r
8.           B_Tree_Split (s, 1, r)        ▷ расщепление старого корня r
9.           B_Tree_Insert_Nonfull (s, k)  ▷ вставка в неполный узел
10. else     B_Tree_Insert_Nonfull (r, k)
```

Алгоритм расщепления полного узла

B_Tree_Split_Child (x, i, y)

```
1.  z ← Allocate_Node( )                   ▷ создание нового дочернего узла z
2.  leaf [z] ← leaf [y]
3.  n [z] ← t – 1
4.  for  j ← 1   to   t – 1                ▷ копирование ключей из y в z
5.      do  key_j [z] ← key_{t+j} [y]
6.  if not leaf [y]
7.      then for j ← 1  to t               ▷ копирование сыновей из y в z
8.           do c_j [z] ← c_{t+j} [y]
9.  n [y] ← t - 1
10. for j ← n [x]+1 downto i +1
11.     do  c_{j+1} [x] ← c_j [x]          ▷ смещение сыновей в узле x
12. c_{j+1} [x] ← z
13. for j ← n [x] downto i
14.     do  key_{j+1} [x] ← key_j [x]      ▷ смещение ключей в узле x
15. key_i [x] ← key_t [y]
16. n [x] ← n [x]+1
```

Алгоритм вставки в неполный узел
B_Tree_Insert_Nonfull (x, k)

```
1.  i ← n [x]
2.  if leaf [x] = TRUE
3.      then while i ≥ 1  и  k <  key_i [x]
4.              do key_{i+1} [x] ← key_i [x]
5.                 i ←  i –1
6.           key_{i+1} [x] ← k
7.           n [x]← n [x]+1
8.      else while i ≥ 1  и  k< key_i [x]
9.              do i ←  i –1
10.          i ←  i+1
11.          if   n[c_i[x]] = 2t – 1
12.             then B_Tree_Split (x, i, c_i[x])
13.                  if  k > key_i [x]
14.                     then i ←  i+1
15.           B_Tree_ Insert_Nonfull (c_i[x], k)
```

По псевдокоду алгоритма разработан смешанный конечный автомат Мили–Мура, в котором действия выполняются как в вершинах, так и на переходах. В соответствии с методом, изложенным в [3], для пошаговой визуализации алгоритма вставки определено 8 состояний (рис. 4).

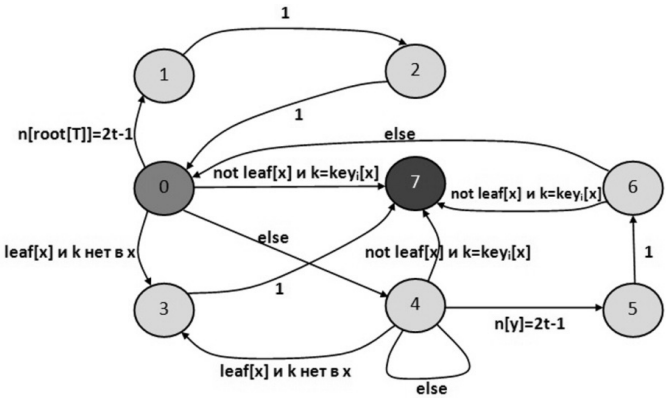

Рис. 4. Граф переходов конечного автомата

Вершины соответствуют «интересным» состояниям алгоритма, которые подлежат отображению на экране. На дугах заданы условия переходов между

вершинами согласно алгоритму. Безусловные переходы на дугах обозначены значением «1». В таблице приведен фрагмент описаний действий на переходах конечного автомата.

Описание действий на переходах автомата алгоритма вставки в Б-дерево

Номер состояния или перехода	Действия на переходе
0	Начальное состояние автомата (выделение корня красным цветом и выполнение начала алгоритма вставки (**B_Tree_Insert)**
7	Конечное состояние автомата
0 → 1	**if** n [root] = 2t –1, то мигание заполненного корня
0 → 7	Поиск позиции i, в которой key_i [**x**] < **k.** Если ключ **k** существует в узле **x,** то переход в конечное состояние 7.(**B_Tree_Insert_Nonfull)**
0 → 3	Поиск позиции i, в которой key_i [**x**] < **k. if leaf[x] = TRUE** (лист) и добавляемого ключа **k** нет в узле **x**, то добавление ключа **k** в позицию **i** узла и его выделение желтым цветом
0 → 4	Поиск позиции i, в которой key_i [**x**] < **k.** Если ключ **k** отсутствует во внутреннем узле **x,** то инкремент **i,** выделение дочернего узла y= c_i[**x**] и связи между узлами **x** и **y** красным цветом
4 → 5	**if** n**[y] = 2t-1,** то мигание узла **y**
1 → 2	Расщепление корня дерева и выделение образовавшихся новых узлов **y** и **z** зеленым цветом (**B_Tree_Split_Child)**

Результат визуализации перехода 0 → 1 под управлением автомата приведен на рис. 5.

Автоматное программирование успешно применено к визуализации таких сложных алгоритмов, как алгоритм удаления из Б-дерева [5], алгоритм построения оптимального дерева поиска [7].

Для алгоритма построения оптимального дерева поиска были разработаны два автомата для прямой и обратной логики алгоритма. Благодаря автомату обратной логики [8] стало возможным продвигаться по шагам алгоритма вперед и возвращаться к ранее пройденным шагам. Обратный ход по алгоритму особенно полезен в режиме тренажера при поиске ошибки, допущенной в пройденных шагах. Но разработка обратной логики для алгоритма построения оптимального дерева поиска оказалась сложной задачей. Более простой способ – на каждом шаге автомата прямой логики помещать модель данных (все переменные, с которыми оперирует алгоритм) в стек, а при обратном ходе просто восстанавливать состояние модели к нужному состоянию автомата.

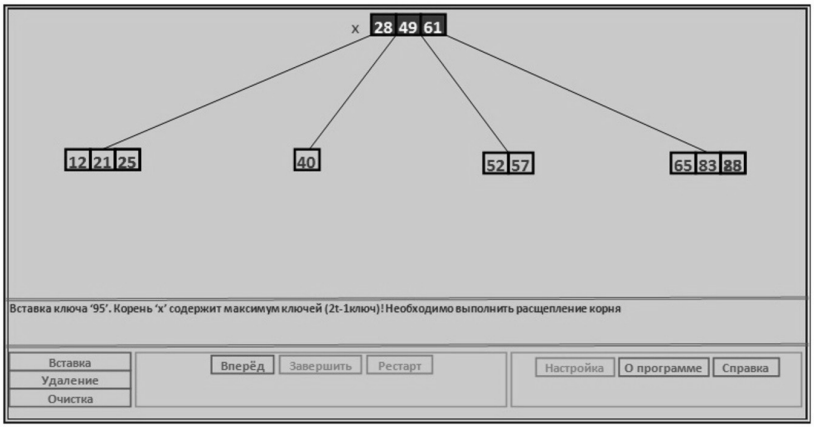

Рис. 5. Переход 0 → 1. Случай заполненного корня

Также в визуализаторы введены такие элементы обучения, как отображение модели данных алгоритма, подстрочные комментарии к каждому шагу алгоритма, синхронная демонстрация команд работающего алгоритма по его псевдокоду, раздел справки с общим описанием алгоритма (рис. 6).

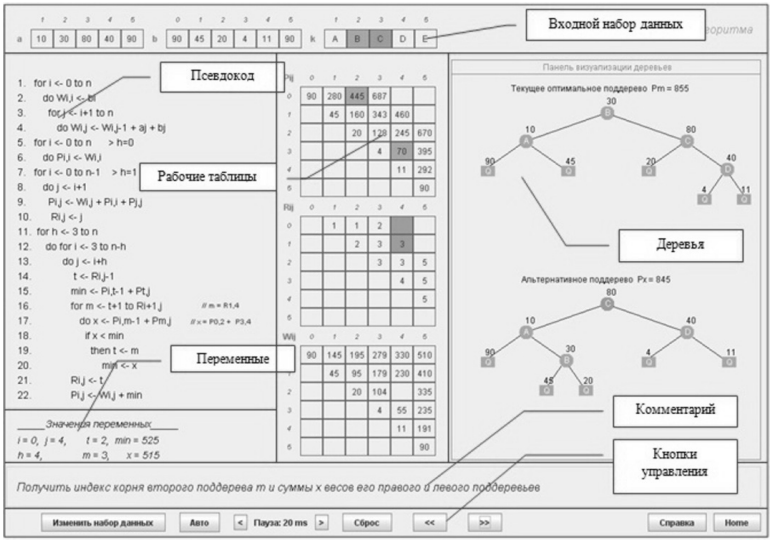

Рис. 6. Окно визуализатора построения оптимального дерева поиска

ЗАКЛЮЧЕНИЕ

Применение метода автоматного программирования для проектирования визуализаторов алгоритмов позволило:

- определить логику пошаговой работы с помощью конечного автомата состояний алгоритма;
- упростить проектирование и программирование визуализаторов за счет лучшей структурированности программ;
- существенно упростить отладку и рефакторинг программ.

Основываясь на опыте разработки представленных в статье визуализаторов, можно рекомендовать применение автоматного программирования как методику проектирования обучающих программ для алгоритмов со сложным поведением.

СПИСОК ЛИТЕРАТУРЫ

1. Структуры и алгоритмы обработки данных: электронное учебное пособие / Т.А. Романенко. – URL: http://edu.nstu.ru/courses/saod/index.htm (дата обращения: 15.03.2021).

2. *Поликарпова Н.И., Шалыто А.А.* Автоматное программирование. – СПб.: б. и., 2008. – 168 с.

3. *Казаков М.А., Шалыто А.А., Туккель Н.И.* Использование автоматного подхода для реализации вычислительных алгоритмов // Телематика-2001: труды Международной научно-методической конференции. – СПб., 2001. – С. 174–176.

4. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы. Построение и анализ. – 2-е изд. – М.: Вильямс, 2005. – 1926 с.

5. Казаков М.А., Шалыто А.А. Использование автоматного подхода для реализации визуализаторов // Компьютерные инструменты в образовании. – 2004. – № 2. – С. 19–33.

6. *Вирт Н.* Алгоритмы и структуры данных: пер. с англ. – 2-е изд., испр. – СПб.: Невский диалект, 2005. – 352 с.

7. *Казаков М.А., Столяр С.Е.* Визуализаторы алгоритмов как элемент технологии преподавания дискретной математики и программирования // Телематика-2000: тезисы докладов Международной научно-методической конференции. – СПб., 2000. – С. 189–191.

Романов Евгений Леонидович, кандидат технических наук, доцент кафедры вычислительной техники Новосибирского государственного технического университета. Основное направление научных исследований – клиент-серверные приложения. Имеет более 50 публикаций. E-mail: romanov@corp.nstu.ru

Романенко Татьяна Александровна, старший преподаватель кафедры вычислительной техники Новосибирского государственного технического университета. Основное направление научных исследований – технология программирования. Имеет более 22 публикаций. E-mail: romanenko@corp.nstu.ru

DOI: 10.17212/2307-6879-2020-4-59-70

Algorithm visualizers design technology*

E.L. Romanov[1], T.A. Romanenko[2]

[1] *Novosibirsk State Technical University, 20 Karl Marx Avenue, Novosibirsk, 630073, Russian Federation, docent of the computing engineering department. E-mail: romanov@corp.nstu.ru*
[2] *Novosibirsk State Technical University, 20 Karl Marx Avenue, Novosibirsk, 630073, Russian Federation, senior lecturer of the faculty of automation and computing engineering. E-mail: romanenko@corp.nstu.ru*

The topic of the article relates to the introduction of such forms of education as visualizers of algorithms into the distance education system of NSTU.

Keywords: algorithm, visualizer, simulator, automatic programming, Miles automaton, Moore automaton, state automaton

REFERENCES

1. Romanenko T.A. *Struktury i algoritmy obrabotki dannykh* [Training manual "Data processing structures and algorithms"]. Available at: http://edu.nstu.ru/courses/saod/index.htm (accessed 15.03.2021).

2. Polikarpova N.I., Shalyto A.A. *Avtomatnoe programmirovanie* [Automatic programming]. St. Petersburg, 2008. 168 p.

3. Kazakov M.A., Shalyto A.A., Tukkel' N.I. [Using of state machine approach in flow algorithms implementation]. *Telematika-2001*: *trudy Mezhdunarodnoi nauchno-metodicheskoi konferentsii* [Proceedings of the international scientific and

* *Received 04 November 2020.*

methodological conference "Telematics-2001"]. St. Petersburg, 2001, pp. 174–176. (In Russian).

4. Cormen T.H., Leiserson C.E., Rivest R.L. *Introduction to algorithms*. 2nd ed. Cambridge, MA, MIT Press, 2001 (Russ. ed.: Kormen T., Leizerson Ch., Rivest R. *Algoritmy. Postroenie i analiz*. 2nd ed. Moscow, Williams Publ., 2005. 1290 p.).

5. Kazakov M.A., Shalyto A.A. Ispol'zovanie avtomatnogo podkhoda dlya realizatsii vizualizatorov [Using an automatic approach to implement visualizers]. *Komp'yuternye instrumenty v obrazovanii = Computer Tools in Education*, 2004, no. 2, pp. 19–33.

6. Wirth N. *Algorithms and data structures*. Englewood Cliffs, NJ, Prentice-Hall, 1986 (Russ. ed.: Virt N. *Algoritmy i struktury dannykh*. 2nd ed. St. Petersburg, Nevskii dialect Publ., 2005. 352 p.).

7. Kazakov M.A., Stolyar S.E. [Visualizers of algorithms as an element of the technology of teaching discrete mathematics and programming]. *Telematika-2000: tezisy dokladov Mezhdunarodnoi nauchno-metodicheskoi konferentsii* [Abstracts of reports of the International scientific and methodological conference "Telematics-2000". St. Petersburg, 2000, pp. 189–191. (In Russian).

Для цитирования:

Романов Е.Л., Романенко Т.А. Технология проектирования визуализаторов алгоритмов // Сборник научных трудов НГТУ. – 2020 – № 4 (99). – С. 59–70. – DOI: 10.17212/2307-6879-2020-4-59-70.

For citation:

Romanov E.L., Romanenko T.A. Tehnologiya proektirovaniya visualizatorov algoritmov [Algorithm visualizers design technology]. *Sbornik nauchnykh trudov Novosibirskogo gosudarstvennogo tekhnicheskogo universiteta = Transaction of scientific papers of the Novosibirsk state technical university*, 2020, no. 4 (99), pp. 59–70. DOI: 10.17212/2307-6879-2020-4-59-70.