Научная визуализация, 2022, том 14, номер 1, страницы 41 - 49, DOI: 10.26583/sv.14.1.04

Визуализация графов при разработке программы проверки знаний по теории графов

Т.М. Кузьмина[1], О.А. Ветрова[2]

РГУ им. А.Н. Косыгина (Технологии. Дизайн. Искусство)

[1] ORCID: 0000-0001-5872-8107, kuzmina_t_m@mail.ru
[2] ORCID: 0000-0001-6935-0787, ve-olga@rambler.ru

Аннотация

В статье рассматривается междисциплинарная задача, в которой объединяются педагогические аспекты и вопросы визуализации. Поскольку модели на графах получили широкое распространение, то изучение теории графов в вузах стало постоянной практикой. В статье речь идет о разработке прикладной программы, которая, с одной стороны, помогает постичь теорию графов, в частности, алгоритмы на графах, а с другой стороны, позволяет объективно оценить полученные знания. Если говорят о проверки знаний с помощью компьютеров, то как правило, речь идет о тестировании. Но для проверки знаний алгоритмов на графах возможности тестов весьма ограничены. Например, при работе с алгоритмом «поиск в глубину» (или «поиск в ширину»), мы имеем дело с задачами, имеющими более сотни(!) положительных ответов. В других задачах, количество правильных ответов измеряется единицами (например, при поиске кратчайшего пути), но велика вероятность угадывания, нахождения ответа методами, не связанными с изучаемыми алгоритмами. Можно, конечно, разбить исходные задачи на множество более мелких, уже пригодных для тестирования, но знание деталей и особенностей, далеко не всегда говорит о знании алгоритма в целом. В статье описана прикладная программа, которая позволяет пользователю совершать действия, согласно выбранному алгоритм. Разработанная программа визуализации воспроизводит результат этих действий на экране и при этом выполняет проверку корректности этих действий.

Ключевые слова: Визуализация алгоритмов, алгоритмы на графах, прикладная программа, остовное дерево, обход вершин, кратчайший путь, алгоритм Форда-Беллмана, алгоритм Дейкстры, цикломатическая матрица.

1. Введение

В статье рассматриваются вопросы визуализации графов при создании прикладной программы контроля знаний по теории графов. Графы широко используются в различных областях знаний, таких, как социология, математическая лингвистика, экономика, биология, медицина, география, программирование, электроника. Сейчас, наверное, трудно привести пример направления человеческой деятельности, где совсем не используются графы, поскольку они предоставляют удобный язык для описания различных моделей [1-3]. Все это приводит к тому, что изучению теории графов отводится большое значение.

Поскольку считается, что около 90% всей получаемой информации человек получает через зрение, а само название граф, говорит о связях с графикой, изображениями, то работы по компьютерной визуализации графов появились сразу, как только прогресс в развитии аппаратных средств, позволил сделать графический интерфейс и компьютерную графику удобными для человека. Отметим, что не все задачи по компьютерной визуализации графов оказались простыми и однозначными.

Особенно сложные проблемы возникают при визуализации больших графов [1,2,4,5,6,7].

При изучении теории графов в курсе дискретной математики работа с большими графами не требуется, демонстрация рисунков графов, содержащих не более 2-х десятков вершин и чуть более ребер не вызывают затруднений и хорошо воспринимается обучающимися. Тем более, что во многих случаях можно ограничиться рассмотрением плоских графов. Но все меняется, когда дело доходит до алгоритмов на графах. При изучении алгоритмов на графах статических изображений явно не хватает. Алгоритмы на графах нередко имеют сложную структуру, большое число шагов и проверок условий. Для быстрого понимания такого алгоритма недостаточно только прочитать его описание и просмотреть статическую схему, даже весьма подробную. Желательно иметь возможность проследить процесс обработки графа рассматриваемым алгоритмом.

Для решения описанных проблем при изучении теории графов авторы сформулировали междисциплинарную задачу, в которой объединяются педагогические аспекты и вопросы визуализации. С точки зрения преподавания важно учесть последовательность изложения материала по нарастанию его сложности. В то же время эта последовательность должна определять принцип создания прикладной программы и выделять наиболее важные аспекты визуализации методических разработок. Для визуализации можно использовать приёмы различных цветовых решений, обозначений, сложных образов действий над графами, интерактивности. В качестве программного решения реализации приёмов различных цветовых решений, обозначений, сложных образов действий над графами, интерактивности были выбраны принципы объектно-ориентированного программирования. В представленной прикладной программе проверки знаний созданы, например, классы «Граф», «Вершина графа», «Ребро графа». Приёмы цветовых решений, различные обозначения, действия над графами, интерактивность определены как методы этих классов. Программа разрабатывалась в среде Microsoft Visual Studio на языке C#.

Многие авторы создают анимации таких алгоритмов [8-11], но для более быстрого и глубокого изучения вопроса полезной оказывается возможность проведения самостоятельных испытаний, т.е. выполнении действий согласно выбранному алгоритму. Конечно, правильность действий должна проверяться. На занятиях проверкой занимается преподаватель, но ее можно переложить на компьютер, что важно при самостоятельном обучении или дистанционном образовании. Причем важны проверки не только завершенной работы, но и промежуточных решений. Работа с программой должна быть интерактивной. И вот для обеспечения успешной интерактивной работы нам потребуется масса обозначений и использование различных цветовых решений. Прикладная программа должна предоставить пользователю возможность совершать различные действия, которые определяются выбранным алгоритмом, при этом картина на экране должна меняться согласно выполненным действиям и быть понятной пользователю.

Поэтому была поставлена задача создания прикладной программы проверки знаний по теории графов, которая с одной стороны будет полезна студентам при изучении алгоритмов за счет их визуализации, а с другой стороны ее может использовать преподаватель для оценки знаний [12-14]. Разработка такой программы будет способствовать формированию методических и программных инструментов, позволяющих качественно и эффективно воспринимать информацию студентами в процессе обучения на основе интерактивной компоненты, а преподавателю достаточно быстро оценивать приобретенные знания.

2.1. Визуализация действий по алгоритмам «поиск в ширину» и «поиск в глубину»

Довольно простыми для визуализации являются алгоритмы «поиск в ширину» и «поиск в глубину». С выбранной вершиной можно выполнить только два действия: либо поместить ее во вспомогательный объект (стек или очередь), либо поместить в список обхода, студент с помощью переключателей определяет, какие действия он хочет выполнить в конкретный момент времени и выбирает вершину, отметив ее щелчком мыши (см. рисунок 1). Поскольку вершина, попав во вспомогательный объект (стек или очередь), должна быть помечена, то эта метка изображается, как окрас вершины в более бледный цвет. В момент выбора вершины, она окрашивается красным цветом, вне зависимости от операции, которая будет выполнена.

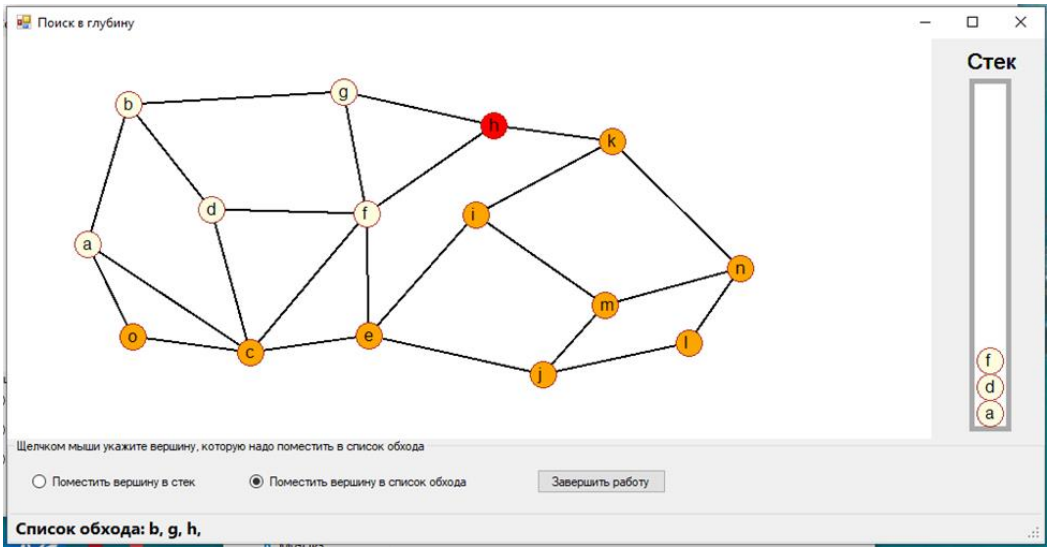

Рисунок 1.Используя алгоритм «поиск в ширину» формируется список обхода графа. На рассматриваемом шаге выбранная вершина "h" помещена в список обхода.

Программа, в случае ошибки, выводит сообщение и считает ошибки. Поскольку при такой работе ошибки могут совершаться не только от незнания алгоритма, но и от волнения или невнимательности, то студенту разрешается совершить 3 ошибки. Если студент совершит 4 ошибки, то его работа обнуляется и ему приходится выполнять задание заново на том же самом графе. Студент работает с одним и тем же графом, но при запуске программы этот граф выбирается с помощью генератора случайных чисел из некоторого списка заранее заготовленных графов. В разработанной ранее программе [11] использовалась случайная генерация самого графа, но в последствии мы пришли к выводу, что лучше использовать заранее подготовленные варианты [12], а случайному процессу предоставить только выбор одного из них. Для создания вариантов заданий разработан специальный редактор, который позволяет строить графы необходимого уровня сложности. Если рассматривается взвешенный граф, то веса ребер, определяются случайным образом при каждом запуске программы.

2.2. Визуализация действий алгоритмов поиска кратчайшего пути

В программе проверки знаний [12] реализованы задания по двум алгоритмам поиска кратчайшего пути – алгоритму Форда-Беллмана и алгоритму Дейкстры. Первый алгоритм состоит из двух этапов: этапа вычисления индексов и этапа построения кратчайшего пути. Поэтому сначала программа предлагает вычислить индексы вершин (см. рис. 2). Для того что бы записать индекс вершины студент выделяет саму вершину, она закрашивается в красный цвет, а в нижнем правом углу формы

появляется поле ввода, в которое можно записать вычисленный индекс, и кнопка «Сохранить», нажав на которую, студент свяжет введенное значение с выбранной вершиной.

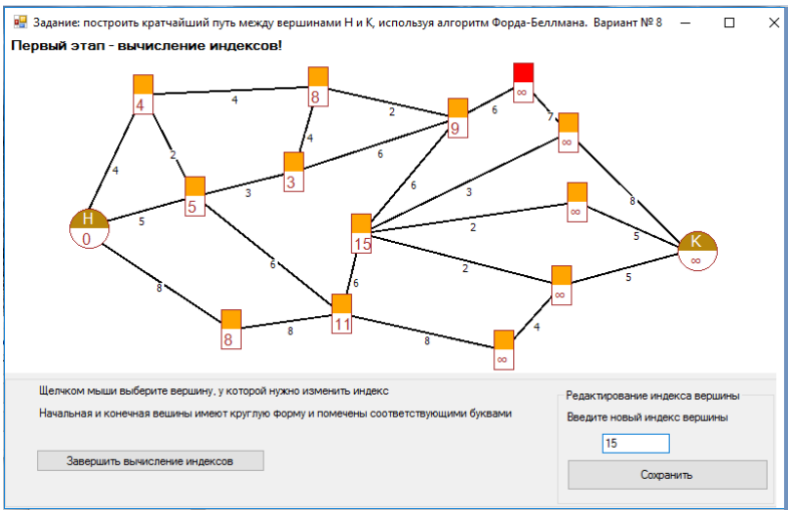

Рисунок 2. Алгоритм Форда-Беллмана. Выбранная в данный момент вершина обозначается красным цветом, именно ее индекс корректируется в поле ввода, расположенном в нижней правой части окна.

После вычисления индексов всех вершин графа студент нажмет кнопку «Завершить вычисление индексов». Если в рассчитанных индексах есть ошибки, программа спросит, не хочет ли студент их найти и исправить самостоятельно. Студент может согласиться и исправить ошибки, и снова отправить решение на проверку. Если студент отказывается искать ошибки, то программа отобразит анализ этих ошибок и студент не сможет их исправить. Если ошибки не касаются кратчайшего пути, то программа позволит продолжить работу и построить один из кратчайших путей. Поскольку анализ ошибок выводится на экран (рис 3), то преподаватель может оценить работу в полном объеме.

Например, на рис. 3 работа выполнена, но с ошибками. Для отображения результатов анализа ошибок были пронумерованы вершины графа, номера вершин изображены белым цветом на оранжевом фоне, сами ошибки перечислены в левом нижнем углу окна. Кратчайший путь построен правильно, ребра, входящие в этот путь, окрашены оранжевым цветом.

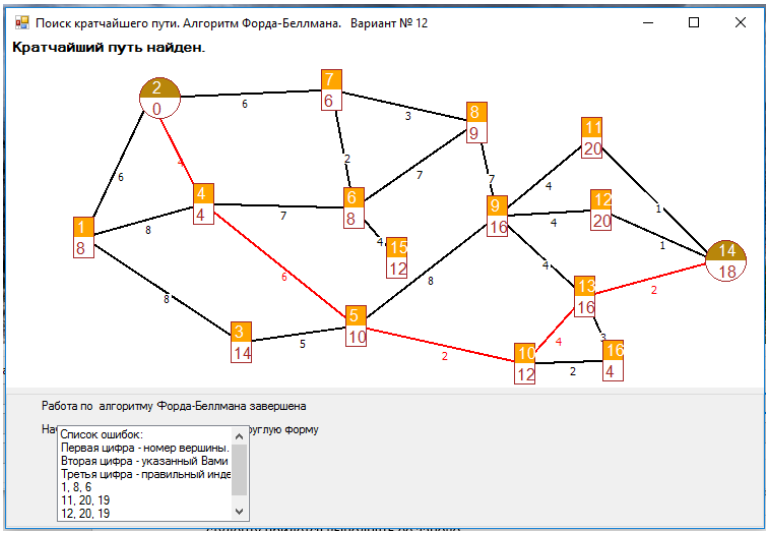

Рисунок 3.Алгоритм Форда-Беллмана: завершение работы, в которой были допущены некритичные ошибки.

Если ошибочные индексы лежат на кратчайшем пути, то продолжение работы невозможно, и студенту придется выполнять ее заново.

В алгоритме Дейкстры также приходится вычислять индексы вершин, но в этом алгоритме процедура сложнее, вершина может быть пройденной, активной, выделенной или вообще нерассмотренной. Да и переключателей режимов действий на форме не два, а три. Если проанализируем ситуацию на рисунке 4, то увидим, что фиолетовый цвет показывает пройденные вершины. Одна из пройденных вершин активна, она отмечена голубой рамкой. Выбранная в данный момент вершина рисуется красным цветом, именно ее индекс корректируется в поле ввода, расположенном в нижней правой части окна. Остальные вершины окрашены оранжевым цветом, так как еще не рассматривались.

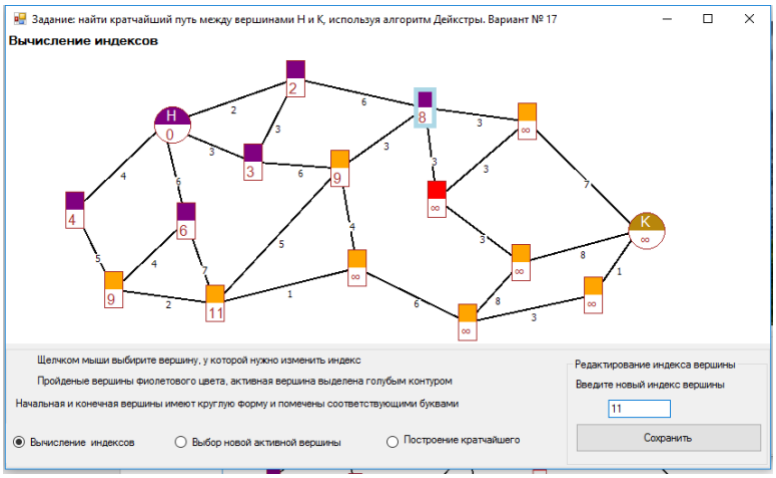

Рисунок 4. Алгоритм Дейкстры.

Каждое завершенное действие студента проверяется, об ошибках сообщается, и они подсчитываются. Так же как при проверке алгоритмов обхода, позволяется сделать не более 3 ошибок. На четвертой ошибке все действия обнуляются, и студент начинает выполнять работу заново.

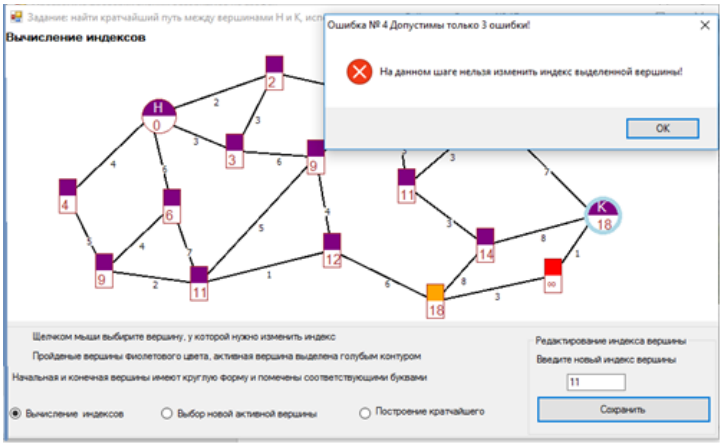

Рисунок 5.Алгоритм Дейкстры. Работа почти завершена, но из-за четвертой ошибки придется все начать сначала.

Такие «откаты» (см рис. 5) мобилизуют студента, появляется азарт, желание победить программу. Тем более, что программа дает возможность просмотреть описания всех рассматриваемых алгоритмов.

2.3. Закраски ребер

При построении остовных деревьев и циклов используется закраска ребер, в некоторых случаях используется частичная закраска. Например, при построении остовного дерева минимального веса, если пользователь совершил ошибку, то ему дается возможность посмотреть на один из вариантов правильного ответа. Для того, чтобы студент мог сравнить это дерево со своим есть режим просмотра с половинным закрасом ребер (см рис.6). На рисунке 6 хорды окрашены в черный цвет, ребра остовного дерева – в желтый.

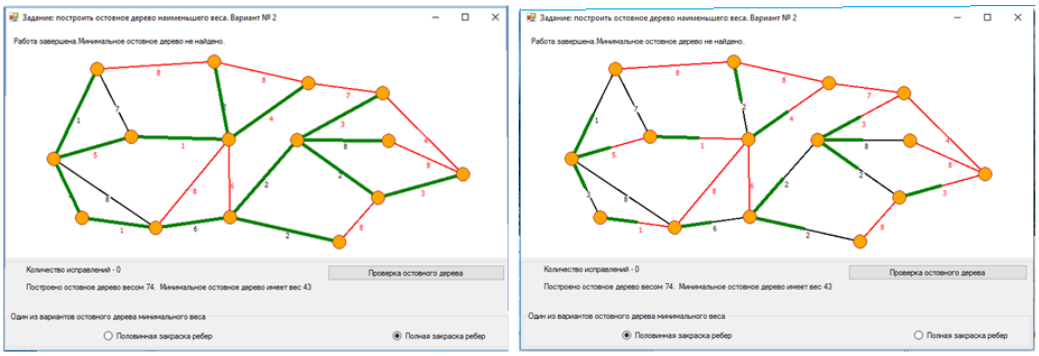

Рисунок 6. Нахождение остовного дерева минимального веса. Студент допустил ошибку и ему предложено посмотреть на один из правильных ответов. Дерево окрашено зеленым цветом. В пунктах a) и b) разные виды закраски ребер.

Частичная закраска используется и при работе с циклами. Например, по изображению цикла надо найти базисные циклы, на которые раскладывается исходный. Поскольку для решения этой задачи важно знать, какие хорды входят в исходный цикл, то используется частичная закраска ребер, входящих в него. Но если надо, то можно включить и полную закраску, подняв флажок «полная закраска ребер цикла», в этом случае сложнее найти необходимые хорды, но лучше просматривается сам цикл. Поскольку цикл подбирается (конечно, случайным образом) таким образом, что его можно представить в виде суммы 2 или 3 базисных циклов, то на форме находятся три раскрывающихся списка, в каждом из которых записаны имена всех базисных циклов. Для составления формулы нужно выбрать имена слагаемых циклов.

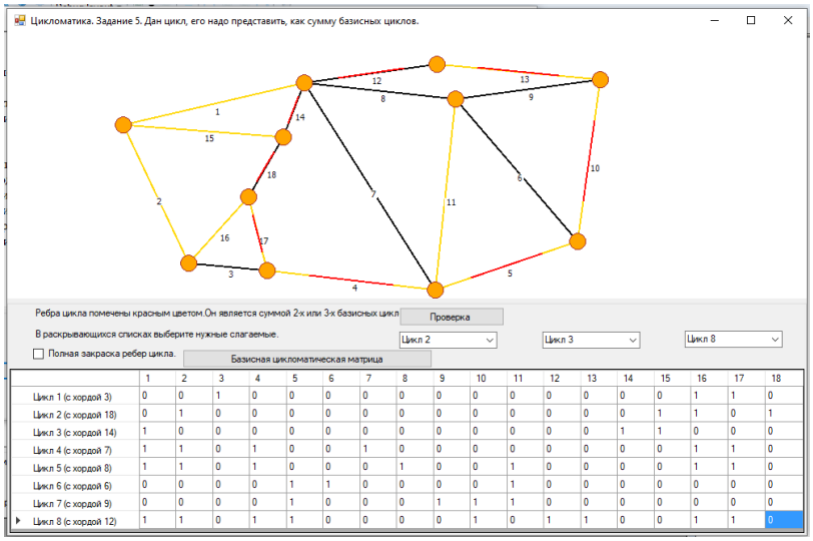

Рисунок 7. Дан цикл, надо найти базисные циклы, которые его определяют. Ребра цикла имеют частичную закраску красным цветом. Частичную закраску можно заменить на полную.

На рисунке 7 показана одна из задач по цикломатике. В этом случае рядом с ребром печатается не его вес, а его номер, который затем используется в цикломатической матрице. Программа предлагает несколько задач по теории графов, касающихся тем, связанных с циклами.

Все эти задачи связаны одной целью – построением базисной цикломатической матрицы и использованием ее. Поэтому работа строится по следующему плану:

1. Определение цикла с помощью вектора, состоящего из нулей и единиц.
   a. Построить цикл по заданному вектору.
   b. Построить вектор по заданному циклу.
2. Базисная система векторов.
   a. Построение остовного дерева.
   b. Определение цикломатического числа.
   c. Построение базисных циклов, входящих в систему, определенную построенным остовным деревом.
3. Разложение произвольного цикла на базисные циклы.
   a. Задаются два или три базисных цикла (указываются их имена, которые используются в базисной цикломатической матрице) нужно построить цикл, равный сумме (рассматривается сложение по модулю 2) этих циклов.
   b. Дан цикл (см. рис 7), надо найти базисные циклы, на которые он раскладывается. Имена циклов, на которые раскладывается исходный, можно найти в раскрывающихся списках.

Поскольку на каждом шаге программа выполняет различные виды проверок, то она подсчитывает ошибки и в итоге печатает их количество.

Заключение

В предлагаемой статье была рассмотрена междисциплинарная задача, которая реализована в виде прикладной программы [12], использующейся при изучении дисциплины «дискретная математика». Рассмотренная компьютерная программа хорошо себя зарекомендовала при дистанционном обучении, кроме того, и при очных занятиях, она разнообразит процесс обучения, делая его более занимательным, но не менее информативным. Несмотря на то, что при реализации различных ситуаций на экране используется много условностей, студенты быстро в них разбираются и с уверенностью используют.

Новизна наших исследований заключается в том, что разработана интерактивная компьютерная программа, использующая визуализацию графов, которая помогает изучить такие алгоритмы, как «поиск в ширину», «поиск в глубину», алгоритм Форда-Беллмана (поиск кратчайшего пути), алгоритм Дейкстры (поиск кратчайшего пути), алгоритм нахождения остовного дерева наименьшего веса, алгоритм Терри и правила построения цикломатической матрицы. При введении этой программы в учебный процесс, было отмечено, что скорость обучения возросла. До использования разработанной программы для изучения алгоритмов поиска кратчайших путей, требовалось 2 практических занятия, по занятию на каждый алгоритм. После введения программы в учебный процесс для изучения обоих алгоритмов потребовалось всего одно занятие, при этом, преподаватель не уносил с собой стопки листов бумаги на проверку. На экзамене ответы студентов на вопросы и задачи по алгоритмам стали более уверенными, а в основную программу дисциплины были переведены 2 дополнительные темы, которые ранее относились к факультативным.

Список литературы

1. Касьянов, В. Н., Евстигнеев, В. А. Графы в программировании: обработка, визуализация и применение. – СПб.: БХВ-Петербург, 2003. – 1104 с. – 3000 экз. – ISBN 5-94157-184-4.
2. Касьянов В.Н., Касьянова Е.В. Визуализация информации на основе графовых моделей // Научная визуализация. - 2014.- Том. 6, N 1. - С. 31 - 50.
3. Овчинников В.А. Графы в задачах анализа и синтеза структур сложных систем. - Москва, 2014.
4. Lisitsyn I.A., Kasyanov V.N. Higres – Visualization system for clustered graphs and graph algorithms // Proc. of Graph Drawing 99. – Lect. Notes in Comput. Sci. – 1999. – Vol. 1731. – P. 82–89.
5. Касьянов В.Н., Золотухин Т.А. Visual Graph – система для визуализации сложно структурированной информации большого объема на основе графовых моделей // Научная визуализация. - 2015. - Том. 7, N 4. – С. 44-59.
6. Kasyanov V.N., Kasyanova E.V. Graph- and cloud-based tools for computer science education // Lecture Notes of Computer Science. – Springer, 2015. – Vol. 9395. – pp. 41-54.
7. Demetrescu C., Finocchi I., Stasko J. T., Specifying Algorithm Visualizations: Interesting Events or State Mapping? // In Proc. of Dagstuhl Seminar on Software Visualization – Lect. Notes in Comput. Sci. – 2001. – P. 16–30.
8. Гордеев Д.С. Обзор техник визуализации алгоритмов на графах. // Научная визуализация. 2018. Т. 10. № 1. С. 18-48.
9. Карпович С.Е., Дайняк И.В., Баев В.С., Разработка анимационных моделей для автоматизированной обучающей системы. //Инновационные образовательные технологии. 2014. № 2 (38). С. 18-24.
10. Романов Е.Л., Романенко Т.А. Технология проектирования визуализаторов алгоритмов. //Сборник научных трудов Новосибирского государственного технического университета. 2020. № 4 (99). С. 59-70.
11. Кузьмина Т.М. Программа «Электронный учебник по теории графов»., //Свидетельство об официальной регистрации программы для ЭВМ. №2003611422. Заявка № 2003610962 от 29.04.2003.
12. Кузьмина Т.М., Программа проверки знаний алгоритмов на графах., //Свидетельство о регистрации программы для ЭВМ RU 2018666894, 24.12.2018. Заявка № 2018664516 от 17.12.2018.
13. Кузьмина Т.М., Ветрова О.А., Использование компьютерной программы "алгоритмы на графах" в учебном процессе., Дизайн и технологии. 2019. № 70 (112). С. 135-139.
14. Кузьмина Т.М., Ветрова О.А., Автоматическая проверка знаний при изучении алгоритмов на графах., Дизайн и технологии. 2018. № 65 (107). С. 136-140.

Scientific Visualization, 2022, volume 14, number 1, pages 41 - 49, DOI: 10.26583/sv.14.1.04

Graph Visualization in the Development of the Knowledge Testing Program on Graph Theory

T.M. Kuzmina[1], O.A. Vetrova[2]

A.N. Kosygin RSPU (Technologies. Design. Art)

[1] ORCID: 0000-0001-5872-8107, kuzmina_t_m@mail.ru
[2] ORCID: 0000-0001-6935-0787, ve-olga@rambler.ru

Abstract

The article considers an interdisciplinary task that combines pedagogical aspects and visualization issues. Since graph models have become widespread, the study of graph theory in universities has become a constant practice. The article deals with the development of an application program that, on the one hand, helps to comprehend graph theory, in particular, algorithms on graphs, and on the other hand, allows you to objectively evaluate the knowledge gained. If we talk about testing knowledge using computers, then as a rule, we are talking about testing. But for testing the knowledge of algorithms on graphs, the tests possibilities are very limited. For example, when working with the algorithm "search in depth" (or "search in width"), we deal with tasks that have more than a hundred (!) positive responses. In other tasks, the number of correct answers is measured in units (for example, when searching for the shortest path), but there is a high probability of guessing, finding the answer by methods unrelated to the algorithms being studied. Of course, it is possible to divide the initial tasks into many smaller ones that are already suitable for testing, but knowledge of the details and features does not always indicate knowledge of the algorithm as a whole. The article describes an application program that allows the user to perform actions according to the selected algorithm. The developed visualization program reproduces the result of these actions on the screen and at the same time checks the correctness of these actions.

Keywords: Visualization of algorithms, algorithms on graphs, application program, spanning tree, vertex traversal, shortest path, Ford-Bellman algorithm, Dijkstra algorithm, cyclomatic matrix.

References

1. Kas'janov, V. N., Evstigneev, V. A. Grafy v programmirovanii: obrabotka, vizualizacija i primenenie. – SPb.: BHV-Peterburg, 2003. – 1104 c. – 3000 jekz. – ISBN 5-94157-184-4 [in Russian].
2. Kas'janov V.N., Kas'janova E.V. Vizualizacija informacii na osnove grafovyh modelej // Nauchnaja vizualizacija. - 2014.- Tom. 6, N 1. - S. 31 – 50 [in Russian].
3. Ovchinnikov V.A. Grafy v zadachah analiza i sinteza struktur slozhnyh sistem. - Moskva, 2014 [in Russian].
4. Lisitsyn I.A., Kasyanov V.N. Higres – Visualization system for clustered graphs and graph algorithms // Proc. of Graph Drawing 99. – Lect. Notes in Comput. Sci. – 1999. – Vol. 1731. – P. 82–89.
5. Kas'janov V.N., Zolotuhin T.A. Visual Graph - sistema dlja vizualizacii slozhno strukturirovannoj informacii bol'shogo ob#ema na osnove grafovyh modelej // Nauchnaja vizualizacija. - 2015. - Tom. 7, N 4.- S. 44 – 59 [in Russian].
6. Kasyanov V.N., Kasyanova E.V. Graph- and cloud-based tools for computer science education // Lecture Notes of Computer Science. - Springer, 2015. - Vol. 9395. - pp. 41-54.

7. Demetrescu C., Finocchi I., Stasko J. T., Specifying Algorithm Visualizations: Interesting Events or State Mapping? // In Proc. of Dagstuhl Seminar on Software Visualization – Lect. Notes in Comput. Sci. – 2001. – P. 16–30.

8. Gordeev D.S., Obzor tehnik vizualizacii algoritmov na grafah. // Nauchnaja vizualizacija. 2018. T. 10. № 1. S. 18-48 [in Russian].

9. Karpovich S.E., Dajnjak I.V., Baev V.S., Razrabotka animacionnyh modelej dlja avtomatizirovannoj obuchajushhej sistemy. //Innovacionnye obrazovatel'nye tehnologii. 2014. № 2 (38). S. 18-24 [in Russian].

10. Romanov E.L., Romanenko T.A. Tehnologija proektirovanija vizualizatorov algoritmov. //Sbornik nauchnyh trudov Novosibirskogo gosudarstvennogo tehnicheskogo universiteta. 2020. № 4 (99). S. 59-70 [in Russian].

11. Kuz'mina T.M. Programma «Jelektronnyj uchebnik po teorii grafov»., //Svidetel'stvo ob oficial'noj registracii programmy dlja JeVM. №2003611422. Zajavka № 2003610962 ot 29.04.2003 [in Russian].

12. Kuz'mina T.M., Programma proverki znanij algoritmov na grafah., //Svidetel'stvo o registracii programmy dlja JeVM RU 2018666894, 24.12.2018. Zajavka № 2018664516 ot 17.12.2018 [in Russian].

13. Kuz'mina T.M., Vetrova O.A., Ispol'zovanie komp'juternoj programmy "algoritmy na grafah" v uchebnom processe., Dizajn i tehnologii. 2019. № 70 (112). S. 135-139 [in Russian].

14. Kuz'mina T.M., Vetrova O.A., Avtomaticheskaja proverka znanij pri izuchenii algoritmov na grafah., Dizajn i tehnologii. 2018. № 65 (107). S. 136-140 [in Russian].