УДК 519.1(075.8), 004.4(075.8)

СРЕДСТВА АВТОМАТИЗАЦИИ ВИЗУАЛИЗАЦИИ АЛГОРИТМОВ НА ГРАФАХ

Малышев А.А.
научный руководитель доктор. физ.-мат. наук. Касьянов В.Н.
Институт систем информатики СО РАН

Изучение теории графов значительно облегчается при наличии наглядного визуального представления как самих графов, так и работы различных графовых алгоритмов. Онлайн-энциклопедии WikiGRAPP по графам и WEGA по графовым алгоритмам, создаваемые и поддерживаемые лабораторией конструирования и оптимизации программ Института систем информатики им. А.П. Ершова СО РАН, являются крупными, постоянно пополняемыми источниками информации по теории графов. Понятно, что статьи энциклопедий, посвящённые графовым алгоритмам, становятся значительно нагляднее при наличии визуализации работы этих алгоритмов, так что полезными могут оказаться средства подготовки и воспроизведения анимационных последовательностей, визуализирующих работу этих алгоритмов.

Целью работы являлось создание средств визуализации работы графовых алгоритмов и их использование для построения анимационных роликов (визуализаций), демонстрирующих работу алгоритмов рекуррентного рисования деревьев, а также некоторых других алгоритмов обработки (кодирования и обхода) деревьев. Разработанные средства должны поддерживать возможность сохранения визуализаций графовых алгоритмов в графических анимированных файлах и простую расширяемость набора алгоритмов, а также быть обладать открытостью и переносимостью, а разработанные визуализации должны быть пригодны для включение в статьи энциклопедий WEGA и WikiGRAPP.

Отметим, что известные средства визуализации графовых алгоритмов (такие, как, например, HIGRES, JHAVE, EVEGA, Animal, Leonardo и Polka), хотя и решают близкие к поставленным задачи, но по отдельности не удовлетворяют одновременно всем выдвинутым требованиям. Также отметим, что, хотя задача визуализации алгоритмов обработки произвольных графов не ставилась, тем не менее, разработанные нами средства позволяют визуализировать алгоритмы рисования произвольных графов.

Система, созданная в рамках данной работы, рассчитана на использование двумя типами пользователей: пользователи, желающие получить визуализацию алгоритма, работающего с имеющимся у них графом, и пользователи - разработчики алгоритмов. Пользователь первого типа может загрузить существующий алгоритм и задать граф в таблице либо вручную, либо с помощью созданной программы (например, можно экспортировать дерево поиска в нужном формате и передать его визуализатору), а пользователь второго типа занимается разработкой сценариев визуализации при помощи созданной системы.

Сценарий отображения задаётся в виде программы на языке Python, расширенным специальными средствами поддержки визуализации. Исполняет его созданная программа Visualisator, включающая в себя модуль работы со встроенным интерпретатором Python, модуль рисования графических примитивов и сохранения кадров анимации в формате GIF, а также модуль разбора и трактовки командной строки. Программа Visalisator написана на языке C++ и скомпилирована при помощи компилятора MinGW. Программа предназначена для использования в ОС Microsoft Windows, однако номенклатура использованных библиотек и отсутствие прямых вызовов команд операционной системы позволяют без труда перенести эту программу и на другие платформы.

Для удобства работы с этой программой была создана ещё одна программа – среда разработки WEGA GUI, представляющая собой объединение редактора исходных текстов, редактора графов и редактора параметров итогового изображения (рис. 1). Среда создана с использованием компилятора MinGW и кросс-платформенной библиотеки QT. Ее интерфейс соответствует основным принципам построения SDI (Single Document Interface). Раздел “file” главного меню содержит команды открытия и сохранения сценария алгоритма на языке Python, а также, закрытия программы. Раздел “graph” содержит команды открытия и сохранения графов. Команда “Run” применяет загруженный алгоритм к загруженному графу. Открытие и сохранение файлов происходит с использованием стандартных файловых диалогов Windows. Команда “Help” выводит информацию о разработчике. В заголовке окна отображено имя файла открытого алгоритма. В статусной полосе (внизу окна) отображено имя открытого графа. Во вкладке Graph в левой части окна редактируется граф – парами номеров вершин задаются рёбра, а также задаётся количество вершин и рёбер.

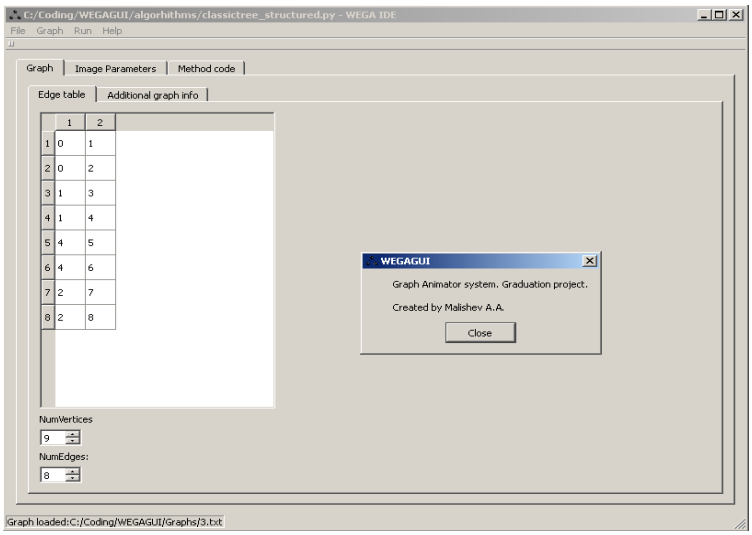

Рис. 1. Вид окна среды разработки WEGA GUI

Графическая оболочка WEGA GUI позволяет редактировать граф через таблицы и более удобно, по сравнению с использованием модуля визуализации через командную строку, редактировать параметры изображения, редактировать исходные коды программ визуализации, а также быстро применять предложенный алгоритм к предложенному графу.

Система Graph Animator была расширена сценариями визуализации различных алгоритмов (назовём их визуализаторами) рисования деревьев, нескольких алгоритмов рисования графов, а также некоторых алгоритмов, предложенных автором. В иллюстрациях приведены последние кадры каждой анимации. Все реализованные визуализаторы автоматически подстраивают изображение под предоставленную им прямоугольную область, что позволяет подбирать нужные масштабы.

Были созданы сценарии, визуализирующие работу рекуррентных алгоритмов рисования деревьев. В таких алгоритмах на каждом этапе алгоритм применяется ко всем поддеревьям, начинающимся с сыновей корневой вершины. Некоторые из алгоритмов, для которых были созданы визуализаторы, приведены ниже.

Первый реализованный алгоритм - алгоритм построения «классического» изображения (рис. 2). Видно, что в ряде случаев некоторые вершины могут оказаться слишком близко, в то время как часть области может оказаться незанятой.

Чтобы можно было уменьшить занимаемую область, избежав перекрытия изображений вершин, данный алгоритм был модифицирован – равномерное распределение блоков (общий шаблон подобных алгоритмов и сопутствующая терминология даны ниже) сменено на динамическое распределение субблоков по размеру поддерева. Ширина прямоугольника W, отдаваемая сыну вычисляется как W = WParent * SubTreeC / SubTreeP. Здесь SubTreeC – велечина (количество вершин) поддерева, начинающегося сыном, SubTreeP – величина дерева, начинающегося отцом, WParent – ширина прямоугольника, предоставленная алгоритмом отцу. Исходя из этого ясно, что чем большее дерево начинается сыном, тем больше места этот сын получит.

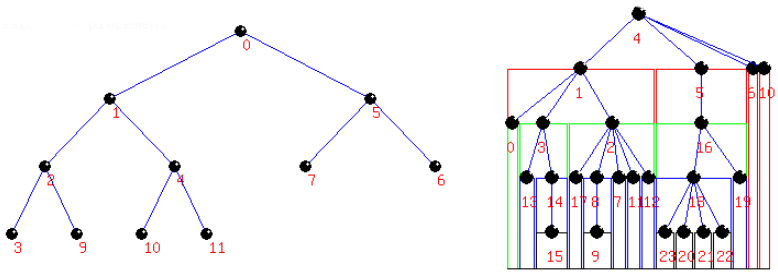

Рис. 2. Примеры работы рекуррентного алгоритма отображения и его модификации

Предложен также алгоритм с построчным расположением уровней дерева. В нём вершина ставится в верхний левый угол предлагаемой прямоугольной области. Алгоритм по временным затратам аналогичен предыдущему. Также реализован и модифицирован алгоритм радиального рисования (рис. 3). Он является близким аналогом первого приведённого алгоритма, но в нем используется в качестве блока размещения не прямоугольник, а сектор круга с вычтенным из него также ориентированным сектором, но с меньшим радиусом.

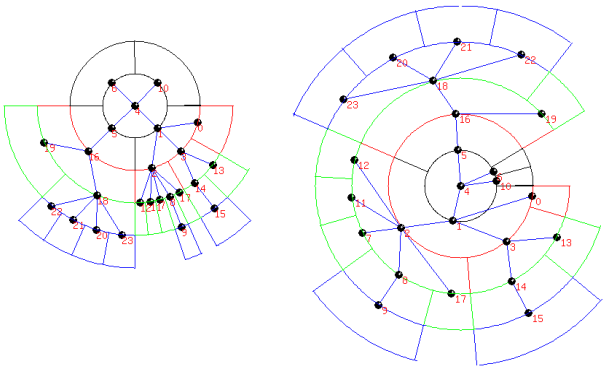

Рис. 3. Примеры работы рекуррентного радиального алгоритма и его модификации

Данный алгоритм аналогично модифицирован, как в предыдущем случае – равномерное распределение блоков (в данном случае – усечённых секторов) сменено на распределение в зависимости от величины соответствующего вершине поддерева. Также были реализованы алгоритм префиксного обхода бинарного дерева и алгоритм посещения дерева в глубину. В них сперва происходит рисование дерева первым алгоритмом, затем окружностью помечаются посещённые вершины, а пройденные рёбра перекрашиваются (рис. 4).

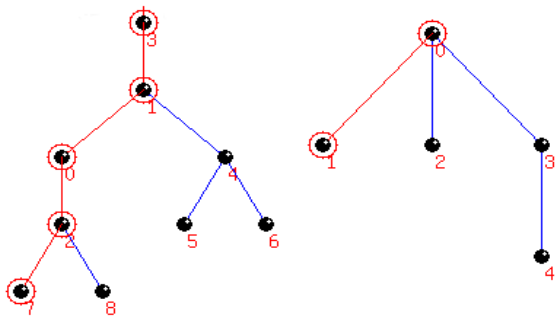

Рис. 4. Шаги алгоритма префиксного обхода бинарного дерева (слева)и алгоритма обхода дерева в глубину (справа)

Реализован сценарий визуализации алгоритма кодирования дерева кодом Прюфера с наглядной демонстрацией каждого шага алгоритма (рис. 5).

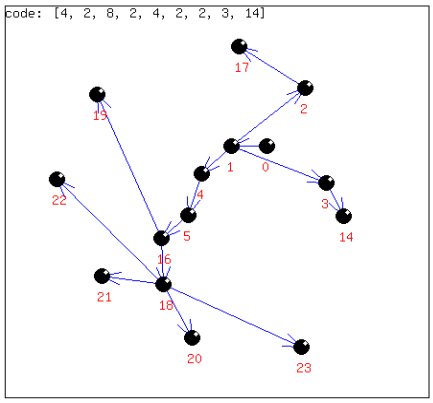

Рис. 5. Один из шагов работы алгоритма построения кода Прюфера по дереву

Как уже отмечалось, помимо визуализации алгоритмов работы с деревьями, были реализованы также некоторые алгоритмы рисования графов общего вида – рисование с расположением вершин на линии с их соединением дугами окружностей (можно использовать и эллипсы для более компактного изображения), а также рисование с расположением вершин на окружности с их соединением прямыми линиями (рис. 6).

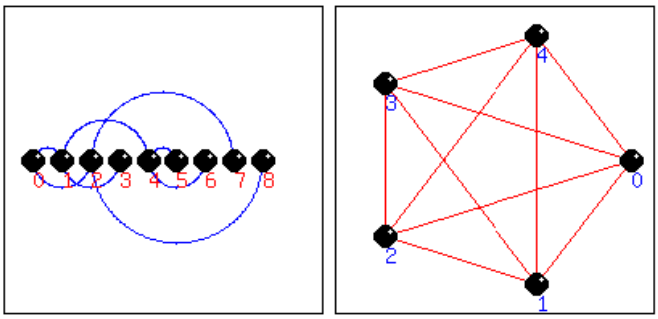

Рис. 6. Примеры работы реализованных алгоритмов рисования графов