Школа современного программирования

ПОПУЛЯРНАЯ ВЕРСИЯ НАУЧНОЙ СТАТЬИ

Апанович Зинаида Владимировна

МЕТОДЫ ВИЗУАЛИЗАЦИИ ГРАФОВ КАК ИНСТРУМЕНТ, СПОСОБСТВУЮЩИЙ ПОНИМАНИЮ ИНФОРМАЦИИ

1. ВВЕДЕНИЕ

В последние годы графы начинают активно использовать в школьном обучении при проведении факультативных занятий. Кроме того, задачи, решаемые с помощью графов, постоянно встречаются не только в олимпиадах, но и в ЕГЭ по информатике. В этой работе мы расскажем о новом направлении современной информатики – методах визуализации графов. Сначала будет рассмотрен пример задачи, которая решается при помощи построения графа и поиска специфического пути в этом графе. Будет показано несколько различных изображений этого графа, демонстрирующих, что одни изображения лучше подходят для решения задачи, чем другие. Затем будет рассказано об эстетических критериях, используемых при построении изображения графа и об основных стилях построения изображения. Наконец, будет описан один простой пример построения размещения вершин графа, который ученики старших классов могут запрограммировать самостоятельно.

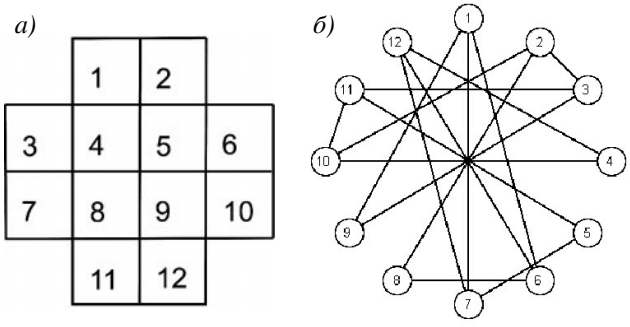


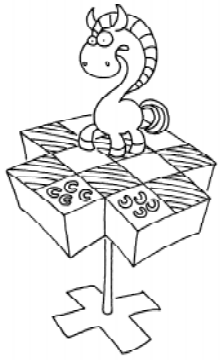
Рис. 1: *а)* шахматная доска с перенумерованными клетками; *б)* изображение графа, соответствующее всем возможным перемещениям коня по клеткам изображенной шахматной доски

2. ПРИМЕР ЗАДАЧИ, ПОКАЗЫВАЮЩИЙ, КАК ПОДХОДЯЩЕЕ ИЗОБРАЖЕНИЕ ГРАФА СПОСОБСТВУЕТ ЛУЧШЕМУ ПОНИМАНИЮ ИНФОРМАЦИИ

Рассмотрим знакомую многим задачу:

Дана шахматная доска, имеющая форму двойного креста, который получается, если из квадрата 4×4 *убрать угловые клетки. Можно ли обойти ее ходом шахматного коня и вернуться на исходную клетку, побывав на всех клетках ровно по одному разу?*

Для решения этой задачи следует перенумеровать последовательно клетки доски, а затем построить граф, вершины которого будут соответствовать клеткам этой доски, а каждое ребро будет соединять две вершины в том случае, когда шагом шахматного коня можно попасть из одной клетки в другую, как это показано на рис. 1.



Для того чтобы решить задачу, необходимо построить цикл, который проходит по всем вершинам ровно один раз и возвращается в ту же самую клетку. Изображение, показанное на рис. 1, весьма полезно при отображении *условия* задачи. Все вершины графа расположены по порядку на одной окружности, и при таком расположении вершин достаточно легко изобразить все ребра, соответствующие ходу шахматного коня. К сожалению, на полученном изображении так много пересечений ребер, что проследить возможные пути, а тем более найти циклы, становится непростой задачей.

Теперь рассмотрим другое изображение этого же самого графа, полученное при помощи самого *обычного силового алгоритма* (описан в конце статьи). Изображение, показанное на рис. 2, отличается от предыдущего тем, что оно симметричное и на нем имеется всего два пересечения ребер. Поэтому обнаружить нужные циклы при таком изображении не представляет большого труда. Можно видеть, что один из искомых циклов имеет вид 1, 9, 3, 2, 8, 6, 12, 4, 10, 11, 5, 7. Изменив порядок обхода этого цикла на противоположный (1, 7, 5, 11, 10, 4, 12, 6, 8, 2, 3, 9), мы получим второе решение задачи об обходе шахматной доски.

Для того чтобы полученный результат был более наглядным, перерисуем изображение графа, расположив вершины на одной окружности в соответствии с найденным порядком, как это показано на рис. 3*а*. Но этим варианты решения задачи про шахматного коня не исчерпываются. Изображение на рис. 2 нам подсказывает, что есть и другие циклы. Например, такой: 1, 6, 8, 2, 10, 4, 12, 7,5, 11, 3, 9 (рис. 3*б*). Еще один вариант решения соответствует инверсии этого порядка.

3. ВИЗУАЛИЗАЦИЯ ГРАФОВ – НОВОЕ НАПРАВЛЕНИЕ СОВРЕМЕННОЙ ИНФОРМАТИКИ

Конечно, рассмотренный пример является немного искусственным, потому что если у нас есть программа, которая строит хорошее изображение графа, то, наверное, эта программа сама может найти нужный цикл в графе и сразу построить изображения, аналогичные тем, что показаны на рис. 3. Но в реальной жизни имеется достаточно много ситуаций, информацию о которых легко получить, рассматривая изображение графа, соответствующего этой ситуации. Поэтому рисовать графы начали задолго до того, как появилось само слово «граф». Еще в древнем Риме знатные римляне изображали свои генеалогические деревья на стенах атриумов. В средние века изображения генеалогических деревьев использовались при рассмотрении в суде имущественных спо-

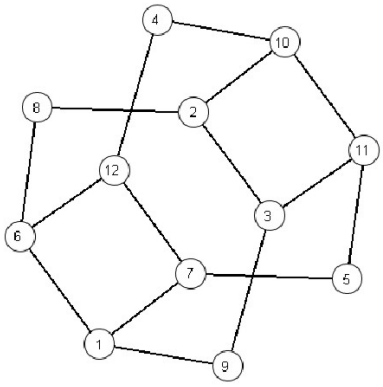


Рис. 2. Изображение графа, полученное силовым алгоритмом

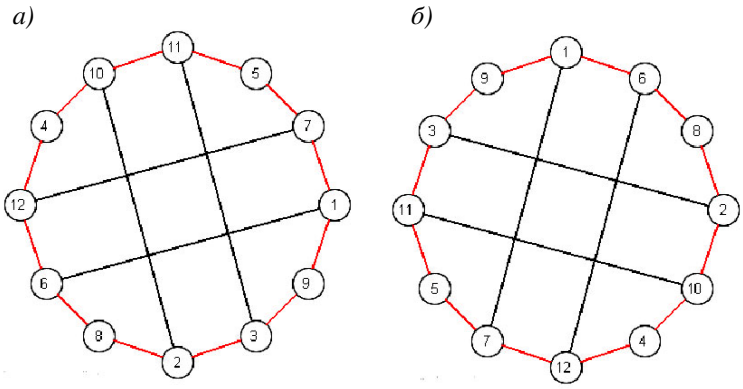
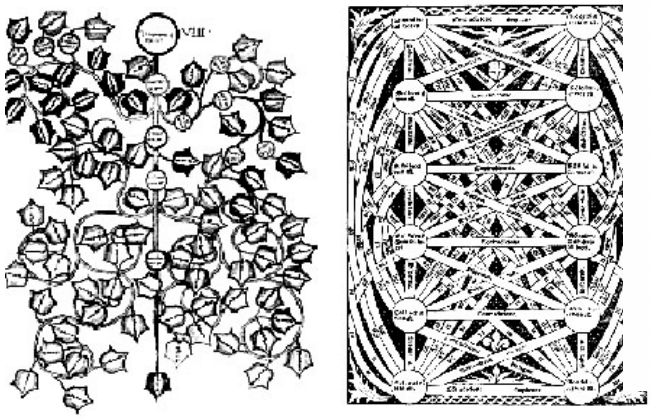


Рис. 3. Различные решения задачи об обходе доски шахматным конем

а) *б)*



...знатные римляне изображали свои генеалогические деревья на стенах атриумов.

Рис. 4. Пример средневековых изображений графов: *а)* генеалогическое дерево, *б)* квадрат оппозиции

ров (рис. 4*а*), а при обучении средневековых студентов логике рисовались так называемые «квадраты оппозиции». Вершинами квадратов оппозиции были логические высказывания, а ребра соответствовали отношениям между этими высказываниями, таким как следование одного высказывания из другого или противоречие между высказываниями (рис. 4*б*).

В конкретных ситуациях одни способы изображения графов подходят больше других, потому что лучше отражают какую-то специфическую информацию. Например, при изображении генеалогического древа мы хотели бы видеть соответствие изображаемых событий реальной хронологии, например, дед не должен появляться в таком изображении после внука. В случае схемы метро, было бы удивительно увидеть расположение станций, разбросанных произвольным образом по всей схеме. Они должны быть расположены в соответствии с их реальным географическим расположением.

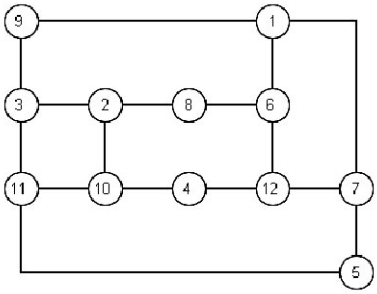


Рис. 5. Планарное изображение графа с рис. 1*а*

Наука, которая изучает разные способы построения изображения графа *при помощи компьютера*, возникла сравнительно недавно, в начале девяностых годов прошлого века. Она называется «Визуализация графов». В ее задачи входит создание классификации различных способов изображения графов, определение критериев, по которым можно сравнивать изображения, и, конечно же, создание новых алгоритмов и исследование свойств этих алгоритмов.

Заметим, что одна из самых старых задач теории графов формулируется как задача построения изображения: «Дан граф *G*. Можно ли его нарисовать на плоскости без пересечений ребер?» Кстати, для рассматриваемого нами графа на рис. 1*б* можно с уверенностью сказать, что такое изображение построить можно. Оно показано на рис. 5. На этом изображении тоже легко найти один из циклов. Хотя оно совсем не содержит пересечений ребер, в нем отсутствует свойство симметричности, и поэтому по данному изображению труднее заметить, что задача о поиске цикла имеет несколько решений.

Существуют и другие свойства изображений графов, которые можно попробовать оптимизировать. В рассмотренном выше примере существенным свойством

изображения, упрощающим задачу визуального поиска циклов, было требование минимального количества пересечений ребер. Можно также потребовать, чтобы изображение содержало минимальное количество сгибов или чтобы ребра были как можно более короткими, а площадь изображения была как можно меньше. Такие требования называются *эстетическими критериями*. Эстетические критерии выражаются при помощи целевых функций, а затем создаются алгоритмы, оптимизирующие заданные свойства.

Разные эстетические критерии могут быть взаимоисключающими. Например, изображение графа с минимальным количеством пересечений может не иметь минимального количества сгибов, как это показано на рис. 6.

Поэтому для разных практических ситуаций возникают различные стили или методологии изображения. Они зависят от конкретных приложений и от доминирующих в приложении типов графа. *Прямолинейное* изображение, показанное на рис. 2, используется часто для неориентированных графов. При построении таких изображений основным требованием является то, что все ребра должны иметь примерно одинаковую длину, вершины должны быть распределены равномерно. Они строятся при помощи методов, *основанных на физических аналогиях*. Изображение, показанное на рисунке 3, относится к классу *круговых* изображений, они тоже применяются для визуализации неориентированных графов. Изображение, показанное на рис. 5, называется *ортогональным изображением*, потому что все ребра этого изображения направлены либо по вертикали, либо по горизонтали. Также оно является *плоским*, потому что не содержит пересечений ребер. Такие изображения используются, например, для описания электрических схем. Для визуализации ориентированных графов используются *иерархические* или *поуровневые* методы изображения. При построении изображения ориентированного графа часто желательно, чтобы все ребра были направлены в одну сторону: сверху вниз или слева направо. Это требование очень естественно при построении изображения всевозможных хронологий. Пример такого изображения показан на рис. 7.

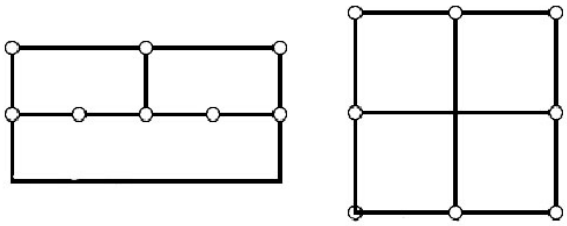
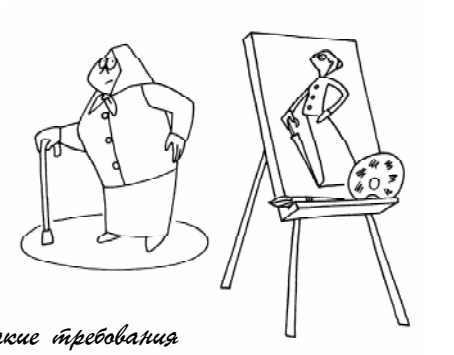


Рис. 6. Два ортогональных изображения одного и того же графа:
а) без пересечений ребер, имеющее 2 сгиба,
б) без сгибов, но с пересечением ребер

Чтобы поближе познакомиться с различными способами построения изображений графов, можно воспользоваться одной из свободно распространяемых программ, доступных через Интернет. Одной из самых известных свободно распространяемых программ является про-



Такие требования называются эстетическими критериями...

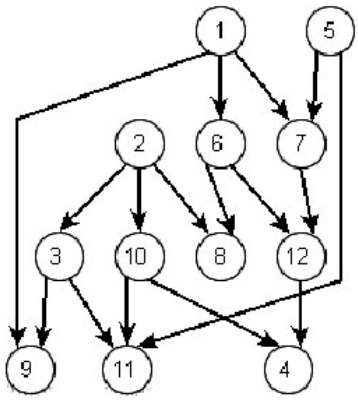


Рис. 7. Поуровневое изображение графа с рис. 2

грамма GraphViz [1]. Для целей использования в школе может подойти свободно распространяемая программа yed.exe [2]. Она весьма проста в использовании и имеет очень удобный и эргономичный редактор графов. В этом редакторе можно нарисовать любой граф вручную, поэкспериментировать с его изображением, а затем сгенерировать автоматически одно из стандартных изображений графа. Программы построения изображений графов входят также и в универсальную систему компьютерной математики Maple [3].

4. ПРИМЕР НЕСЛОЖНОГО АЛГОРИТМА РАЗМЕЩЕНИЯ ВЕРШИН ГРАФА

Следует заметить, что разработка алгоритмов и программ рисования графов требует существенных познаний как в области теории графов, так и в области алгоритмов построения изображений графов [5]. Методы, используемые для построения изображений, содержат, как правило, много этапов и настроечных возможностей. Тем не менее, в данной работе мы хотим показать пример одного несложного алгоритма, который школьники могут запрограммировать самостоятельно. Данный алгоритм был предложен Идесом [4] в 1984 году. Он может находить приемлемые размещения вершин графов, содержащих 30–50 вершин.

Итак, предположим, что дан связный неориентированный граф $G = (V, E)$. V – множество вершин графа G, E – множество ребер. Мы хотим получить такое размещение вершин графа G, чтобы все его ребра были примерно одинаковой длины, а само размещение было как можно более симметричным.

Для построения размещения рассмотрим следующую механическую модель. Будем считать, что вершины графа – это стальные шары, а ребра графа – пружины, их соединяющие. Сначала вершины размещаются в произвольное начальное состояние. Затем им разрешается двигаться так, что силы, действующие на шары, переводят систему в состояние равновесия, то есть такое состояние, где сумма всех сил, действующих на каждую вершину, равна нулю. Обычно, такое состояние недостижимо, поэтому ищется приближенное решение. Особенностью данного алгоритма является то, что «силы», действующие в такой системе, не следуют физическим законам, а являются их «аналогией».

Пусть $P = (p_v)$, $v \in V$ – это вектор позиций вершин. Каждая вершина v имеет координату $P_v = (x_v, y_v)$.

Расстояние между вершинами v и u обозначается $\| p_v - p_u \|$ и вычисляется по формуле: $\| p_v - p_u \| = \sqrt{(x_v - x_u)^2 + (y_v - y_u)^2}$.

Будем обозначать $\overrightarrow{p_u p_v} = \frac{p_v - p_u}{\| p_v - p_u \|}$ – единичный вектор, направленный от p_u к p_v.

Для решения задачи размещения вершин предположим, что на вершины, *связанные ребром*, действует сила, которая заставляет ребро растягиваться, если его длина меньше выбранной идеальной длины, и сжиматься, если его длина превышает идеальную длину. Назовем эту силу «силой пружины» и обозначим f_{spring}. Для вычисления «силы пружины» будем пользоваться такой формулой:

$$f_{spring}(p_u, p_v) = c_{spring} \log \frac{\| p_u - p_v \|}{l} \overrightarrow{p_v p_u},$$

где l – «идеальная» длина пружины, то есть идеальное расстояние, на котором мы хотим располагать вершины графа. c_{spring} – настроечный параметр, управляющий силой пружины, его можно подбирать по своему усмотрению. Как мы уже предупредили, f_{spring} отличается от силы, действующей на реальные пружины. Реальные пружины подчиняются закону Гука. Зато наша пружина будет растягиваться, если ее реальная длина (расстояние между вершинами u и v) окажется меньше идеальной длины, а если ее реальная длина станет больше идеальной длины, пружина будет сжиматься. Это происходит благодаря использованию функции логарифм.

Помимо силы пружины, введем еще и силу отталкивания, которая действует между каждыми двумя вершинами u, $v \in V$,

не связанными ребром. Эта сила нужна нам для того, чтобы вершины не сдвигались слишком близко. Обозначим силу отталкивания f_{rep} и будем ее вычислять по формуле:

$$f_{rep}(p_u, p_v) = \frac{c_{rep}}{\| p_v - p_u \|^2} \overrightarrow{p_u p_v},$$

где c_{rep} является тоже настроечной константой.

Осталось решить вопрос о том, как получить конфигурацию равновесия для описанной системы.

Чтобы достичь положения равновесия данной системы, будем итеративно сдвигать вершины в момент времени t в соответствии с вектором равнодействующих сил $F_v(t)$. Этот вектор является суммой всех сил отталкивания и сил пружины, действующих на вершину v:

$$F_v(t) = \sum_{u:\{u,v\}\notin E} f_{rep}(p_u, p_v) + \sum_{u:\{u,v\}\in E} f_{spring}(p_u, p_v)$$

После вычисления вектора $F_v(t)$ для всех $v \in V$ каждая вершина сдвигается по нему на величину $d*F_v(t)$. Константа-множитель $d < 1$ используется для предотвращения излишнего перемещения вершин, которые могут возникнуть при синхронных заменах позиций вершин. Итеративно вычисляя силы, действующие на каждую вершину, и изменяя соответственно позиции этих вершин, система приходит в стабильное состояние, в котором дальнейшие локальные улучшения невозможны. Схема алгоритма, который ищет размещение вершин графа, выглядит следующим образом:

Алгоритм размещения вершин графа

Вход: Связный неориентированный граф $G = (V, E)$ и начальное размещение его вершин $p = (p_v)$, $v \in V$.

Выход: Размещение вершин, соответствующее положению равновесия системы.

t:=1;

1. Вычислить значение силы $F_v(t)$ для каждой вершины $v \in V$;

2. Вычислить новую координату каждой вершины по формуле: $p_v := p_v + dF_v(t)$; t:=t+1;

3. Продолжать процесс, пока $t < M$.

Обычно для получения приемлемого размещения хватает пятидесяти итераций.

5. ЗАКЛЮЧЕНИЕ

В данной работе мы рассказали о новом направлении современной информатики – методах визуализации графов и показали, как различные стратегии изображения графов могут влиять на восприятие изображаемой информации. Программные средства визуализации графов можно найти в Интернете. Использование таких программных продуктов в школе может способствовать более быстрому усваиванию материалов, связанных с применением графов для решения различных практических задач.

Литература

1. www.graphviz.org (в свободном доступе программа визуализации графов GraphViz).
2. http://www.yworks.com/en/products_yed_about.html (в свободном доступе графический редактор и программа генерации изображений графов yed).
3. www.maplesoft.com/ (программное обеспечение для преподавателей, студентов и инженеров).
4. *Peter Eades*. A Heuristic for Graph Drawing. // Congressus Numerantium. Vol. 42. 1984. P. 149–160.
5. *Апанович З.В.* Методы визуализации информации – наукоемкое направление современных ИТ // Компьютерные инструменты в образовании, 2010. № 2. С. 20–27.

Апанович Зинаида Владимировна, кандидат физико-математических наук, старший научный сотрудник Института систем информатики им. А.П. Ершова СО РАН и Новосибирского государственного университета (НИУ НГУ).