Разработка приложения для визуализации алгоритмов на графах

Огнева Т.А.[1], Казачкова А.А.[2]
[1]ognevata28@yandex.ru, [2]kazachkova.anna@gmail.com
Саратовский государственный университет имени Н.Г. Чернышевского

Аннотация. В статье рассмотрено создание приложения, которое может использоваться как на занятиях по теории графов в ВУЗе и в школе, так и при самостоятельном изучении алгоритмов на графах.

Ключевые слова: граф, алгоритмы, визуализация.

Теория графов изучается в ВУЗах на направлениях, связанных с математикой и компьютерными науками [1]. Алгоритмы на графах используются при решении олимпиадных задач, как в старшей школе, так и в ВУЗе. С необходимостью применения графовых алгоритмов все чаще и чаще сталкиваются профессиональные разработчики. Поэтому актуальным является создание приложений для визуализации алгоритмов на графах, которые делают изучение материала более наглядным и эффективным [2].

Основные определения

Графом (простым) называется пара $G = (V, E)$, где V – конечное множество вершин, а E – множество неупорядоченных пар различных элементов множества V (ребра), иначе – ориентированным (элементы множества E – дуги) [3].

Подграфом графа G называется граф H, для которого выполняются следующие условия:

1. $V(H) \subseteq V(G)$;
2. $E(H) \subseteq E(G)$;
3. любое ребро $e \subseteq E(H)$, соединяющее пару вершин x и y в H, должно соединять ту же самую пару в G.

Можно сказать, что любой подграф H графа G – это граф, полученный из исходного с помощью последовательного выполнения двух операций:

– Удаление ребра e.
– Удаление вершины x.

Пусть имеется граф G и пусть для построения его подграфа H будем использовать только операцию удаления ребер. Тогда в полученном подграфе множество вершин будет совпадать с множеством вершин G. Такой подграф называется остовным подграфом графа G.

Маршрутом в графе G из вершины x_0 в вершину x_k называется последовательность вершин и ребер $x_0, e_1, x_1, e_2, x_2, \dots, x_{k-1}, e_k, x_k$ вершин $x_i \in V$ и ребер $e_i \in E$, где $e_i, = (x_{i-1}, x_i)$ [3]. Путем называется маршрут, в котором все ребра различны. Простой путь - это путь, в котором различны все вершины. Если в простом пути начальная и конечная вершина совпадают, то это замкнутый путь. Циклом называется замкнутый простой путь. Дерево – простой связный граф без циклов.

Способы представления графа

Информационные технологии в образовании. 2022. Вып 5

Для представления графов в компьютере чаще всего используют матрицы смежности или списки смежности.

Матрица смежности взвешенного графа G с конечным числом вершин n – это квадратная матрица размера n × n, в которой элемент $a_{i,j}$ на пересечении i столбца и j строки задается следующим образом:

$$a_{i,j} = \begin{cases} k, & \text{если вершины } i, j \text{ смежны, } k_{i,j} - \text{вес ребра (дуги)} \\ 0, & \text{если вершины } i, j \text{ не смежны} \end{cases}$$

Матрица смежности позволяет быстро определить, соединены ли две данные вершины ребром (дугой), однако она требует n^2 ячеек памяти и может быть сильно разрежена в случае, когда число ребер (дуг) намного меньше n^2. Поэтому матрицу смежности имеет смысл использовать для небольших графов или плотных (количество ребер (дуг) близко к n^2).

Если для каждой вершины графа задан список смежных с ней вершин (список смежности), то будем говорить, что граф задан с помощью списков смежности. Этот способ представления эффективно расходует ячейки памяти (нужно $|V| + |E|$ ячеек), что критично для графов с большим количеством вершин. Также в нём удобно искать смежные непросмотренные вершины. Однако для того, чтобы выяснить, смежны ли вершины i и j, нужно найти список смежности вершины i и пройти по нему.

Есть и другие, более специфичные способы представления графов. Но они обычно создаются для конкретных случаев (так, например, список ребер используется для доказательства некоторых теорем), поэтому подробно рассматриваться не будут.

Алгоритмы построения каркаса графа

Задача об оптимальном каркасе (стягивающем дереве) состоит в следующем: Дан взвешенный неориентированный граф $G = (V, E)$. Требуется в графе G найти каркас минимального веса. Будем предполагать, что граф G связен, так что решением задачи всегда будет дерево. Существует несколько алгоритмов для построения каркаса минимального веса. Наиболее известные из них – это алгоритмы Прима и Крускала [4]. Оба они являются жадными.

Алгоритм Прима начинает свою работу от начальной (произвольной) вершины. На каждом шаге к уже построенному решению добавляются ребра и вершины, пока не будет получен остов. Причем новое ребро всегда соединяет вершину в уже построенном дереве с вершиной, которая еще ему не принадлежит, причем из всех подходящих ребер выбирается то, которое имеет наименьший вес. Алгоритм завершается, когда использованы все вершины исходного графа.

В алгоритме Краскала сначала сортируются ребра в зависимости от их веса. Затем на каждом шаге алгоритма выбирается ребро минимального веса, которое не образует цикл с уже выбранными ребрами. Алгоритм завершаются, когда таких ребер не найдется.

189

Информационные технологии в образовании. 2022. Вып 5

Визуализатор алгоритма обходов

189

Информационные технологии в образовании. 2022. Вып 5

Было реализовано приложение – визуализатор алгоритма построения каркаса взвешенного неориентированного графа. Приложение было написано на языке C# с использованием Windows Forms, которые позволяют сделать удобный интерфейс. Граф задавался списком смежности, для реализации которого использовались словари. Словари имели несколько уровней вложенности: 1) ключ: <название вершины>, значения: <список смежности> 2) ключ: <вершина>, значение: <вес ребра>. Проект содержит класс WeightedGraph для взвешенных графов с полной реализацией, класс-обертку GraphMaker для сокрытия исходной реализации от пользователя. В качестве алгоритма построения каркаса был выбран алгоритм Прима, его реализация представлена методом WireframeByPrimAlgorithm класса WeightedGraph.

Отрисовка выполняется следующим образом. Для наилучшего расположения вершин на плоскости задается минимальное расстояние между вершинами. Затем по очереди случайным образом генерируются координаты вершин. Для каждой следующей координаты проверяется, соответствует ли расстояние между ней и остальными заданному минимуму. Если условие нарушено, координаты для этой точки генерируются заново.

Вершины отрисовываются в виде окружностей с подписями внутри. Для этого используются функции DrawEllipse и DrawString из библиотеки Drawing. Ребра представляют собой отрезки с подписанными весами. Для их отрисовки используются функции DrawLine и DrawString из библиотеки Drawing. По умолчанию цвет графа синий, по мере построения каркаса, входящие в него ребра перекрашиваются в бирюзовый цвет для наглядности.

Возможности приложения:

1. Считать граф из файла (есть несколько примеров, также можно создать свой файл с представлением графа в заданном формате).
2. Создать граф в приложении с помощью добавления (удаления) вершин и ребер.
3. Сохранить созданный граф в файл.
4. Построить каркас графа.
5. Посмотреть поэтапное построение каркаса (поддерживается как последовательное построение, так и переход на произвольно выбранный этап).
6. Отрисовать каркас графа, посмотреть поэтапную отрисовку (поддерживается как последовательное построение, так и переход на произвольно выбранный этап).

Для удобства пользователя в приложении имеется возможность посмотреть инструкцию (в инструкцию также включён предпросмотр графов-примеров); построение каркаса в виде списка смежности идет параллельно со списком смежности графа; при добавлении/удалении вершины/ребра каркас динамически изменяется. В приложении предусмотрены предупреждения в случаях попыток пользователя выполнить некорректную операцию (таких как добавить уже существующую вершину или удалить ребро, одним из концов которого является несуществующая вершина).

Информационные технологии в образовании. 2022. Вып 5

Данное приложение можно использовать при изучении алгоритмов на графах как в рамках занятий в старших классах школы и в ВУЗах, так и при самостоятельном изучении.

Список литературы

[1] *Кудрина Е.В., Огнева М.В.* Теория графов: от школы до вуза // Материалы V Всероссийской научно-практической конференции «Информационные технологии в образовании». Саратов: Издательский центр «Наука». 2013. – С.23-26.

[2] *Simonak Slavomir*. (2014). Using algorithm visualizations in computer science education. Open Computer Science. 4. 10.2478/s13537-014-0215-4.

[3] *Богомолов А.М., Салий В.Н.* Алгебраические основы теории дискретных систем. / А.М. Богомолов, В.Н. Салий. М.: Наука. Физматлит. 1997. – 368 с.

[4] *Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein* Introduction to algorithms. Third edition // The MIT Press, 2009, 1313 p.