И.Б. Бурдонов, А.С. Косачев
Исследование графа автоматом.
Программная инженерия. т.7, №11, 2016 , стр. 498-508.
11 стр.

---

УДК

**И. Б. Бурдонов**, д-р физ.-мат. наук, вед. науч. сотр., igor@ispras.ru,

**А. С. Косачев**, канд. физ.-мат. наук, вед. науч. сотр., kos@ispras.ru.

Институт системного программирования РАН, Москва

# Исследование графа автоматом

Статья является первой в серии из трёх статей, посвящённых исследованию графов автоматами. Автомат двигается по рёбрам графа, используя вершины графа как ячейки памяти для чтения/записи. Рассматриваются алгоритмы обхода графа (проход по всем рёбрам) с оценкой их сложности в зависимости от вида графа и типа автомата. В последующих двух статьях будет рассмотрено исследование графов (в том числе недетерминированных или динамически меняющихся) коллективом автоматов.

**Ключевые слова:** неориентированный граф, ориентированный граф, упорядоченный граф, нумерованный граф, неизвестный граф, обход графа, автомат, робот, полуробот.

## 1. Введение

Задача исследования графа возникает каждый раз, когда объект исследования моделируется графом. Вот лишь несколько примеров. 1) Компьютерные сети моделируются графом (сетевая топология [1]), в котором вершины соответствуют компьютерам, а рёбра – линиям связи. 2) Гипертексту, в частности, веб-страницам интернета [2], также соответствует граф, в котором вершины – это страницы (или их элементы), а рёбра – это гиперссылки. Заметим, что в этом случае рёбра ориентированные. 3) Сложные (составные) программные и аппаратные системы моделируются графами, в которых вершины соответствуют компонентам системы, а рёбра – связям между ними. 4) Автоматные и автоматоподобные (например, системы

И.Б. Бурдонов, А.С. Косачев
Исследование графа автоматом.
Программная инженерия. т.7, №11, 2016 , стр. 498-508.
11 стр.

---

размеченных переходов – LTS – Labelled Transition System) модели программных и аппаратных объектов [3,4,5]. Наиболее удобным и распространённым способом представления такой модели является ориентированный граф переходов, в котором вершины изображают состояния объекта, а рёбра – переходы.

Почему графы исследуются автоматами? Прежде всего, хочется, чтобы исследование выполнялось автоматически, т.е. с помощью некоторого алгоритма. В этом случае исследование графа может выполнять компьютер. Формализацией алгоритма является машина Тьюринга, т.е. автомат, «ползающий» по ленте. При исследовании графа автомат «ползает» по графу, перемещаясь вдоль его рёбер, а вершины играют роль ячеек ленты машины Тьюринга.

На практике, с одной стороны, нужно, чтобы автомат требовал поменьше памяти, а, с другой стороны, размеры графа, то есть число его вершин и рёбер, обычно как-то ограничены. Поэтому мы будем рассматривать и такие автоматы, которые не конечны на рассматриваемом классе графов, но конечны на любом подклассе графов ограниченного размера. Автомат, который конечен на всём классе графов, будем называть *роботом*. Если автомат не конечный, но граф не помещается в его память, то такой автомат будем называть *полуроботом*. По сути, это то же самое, что локальный алгоритм на графе. А если граф помещается в память автомата, то это *неограниченный автомат*.

Как правило, исследование графа сводится к *обходу графа*, под которым понимается построение и проход автоматом маршрута, содержащего все вершины и рёбра графа. Иногда имеет смысл говорить о покрытии рёбер графа не одним маршрутом, а набором маршрутов: либо из-за недостаточной связности графа, либо для ускорения времени обхода, если эти маршруты

И.Б. Бурдонов, А.С. Косачев
Исследование графа автоматом.
Программная инженерия. т.7, №11, 2016 , стр. 498-508.
11 стр.

---

проходятся параллельно несколькими автоматами. Обычно обход должен начинаться с выделенной начальной вершины графа, которую для краткости будем называть *корнем*.

Приведём только два, зато весьма показательных, примера исследования графа, сводимого к его обходу. 1) Обход лабиринта, который является моделью многих частных задач. Отсюда возникли как понятие обхода графа, так и сама теория графов, когда Эйлер решал проблему семи мостов Кёнигсберга [6]. 2) Тестирование, при котором граф – это граф переходов автоматной или автоматоподобной модели тестируемой системы [7,8,9,10]. Её начальное состояние – это корень, автомат на графе – тестирующая система, много автоматов на графе – распределённая тестирующая система. Проход автомата из текущей вершины по выходящему из неё ребру моделирует тестовое воздействие на систему, находящуюся в текущем состоянии, и наблюдение результата этого воздействия, в том числе изменённого состояния системы.

Во многих случаях такая модель, в которой автомат двигается по рёбрам графа, а вершины графа используется как память, из которой автомат читает и в которую он пишет, может быть «инвертирована» (см., например, [11]). В такой «инвертированной» модели в каждой вершине неподвижно «сидит» один автомат, а по рёбрам графа перемещаются сообщения, которые автоматы пересылают друг другу.

Данная статья – это первая статья серии из трёх статей, посвящённых исследованию графов автоматами. В этих статьях изучаются различные алгоритмы обхода графов в зависимости от типа графа и типа и количества автоматов. В данной, первой, статье рассматривается исследование графа одним автоматом. Граф может быть как неориентированным, так и ориентированным. Вторая статья посвящена исследованию ориентированных

графов коллективом двигающихся автоматов, в том числе исследованию недетерминированных графов. В третьей статье ориентированный граф исследуется коллективом неподвижных автоматов, которые обмениваются между собой сообщениями, пересылаемыми по рёбрам графа. Кроме исследования свойств самого графа, рассматриваются параллельные вычисления на графе, которые могут выполнять автоматы. Граф может быть как статическим, так и динамическим, т.е. меняющимся во времени.

## 2. Основные понятия и термины

### 2.1. Упорядоченный граф

Когда выполняется обход графа, на каждом шаге автомат должен указать ребро, по которому он «хочет» выйти из текущей вершины. Для этого используется нумерация таких ребер, начиная с 1, – для каждой вершины своя нумерация рёбер. Автомат указывает номер ребра в текущей вершине. В ориентированном графе ребро проходится только в направлении его ориентации, поэтому для данной вершины нумеруются только выходящие из неё рёбра, и ребро имеет номер только в вершине, из которой выходит. В неориентированном графе ребро может проходиться в обоих направлениях, поэтому для данной вершины нумеруются все инцидентные ей рёбра, и ребро имеет два номера: по одному в каждой из инцидентных ему вершин. Такой граф называется *упорядоченным* (Рис. 1).

И.Б. Бурдонов, А.С. Косачев
Исследование графа автоматом.
Программная инженерия. т.7, №11, 2016 , стр. 498-508.
11 стр.

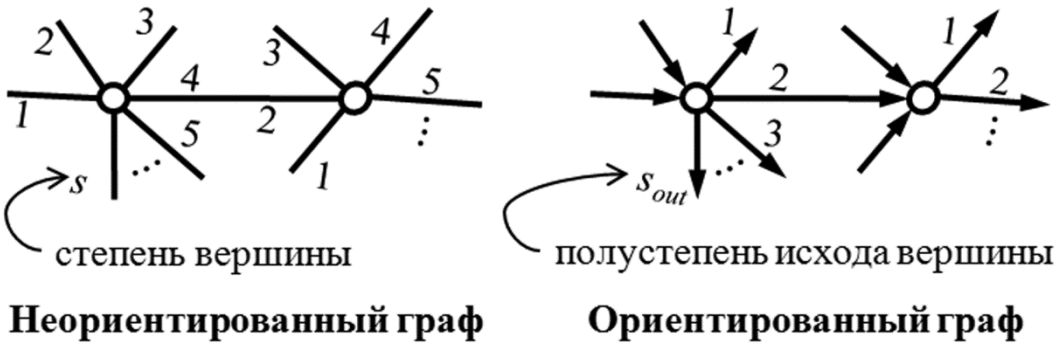



Рис. 1. Нумерация рёбер в упорядоченном графе.

### 2.2. Структура на графе

Посмотрим, какая структура возникает на графе при его обходе (Рис. 2).

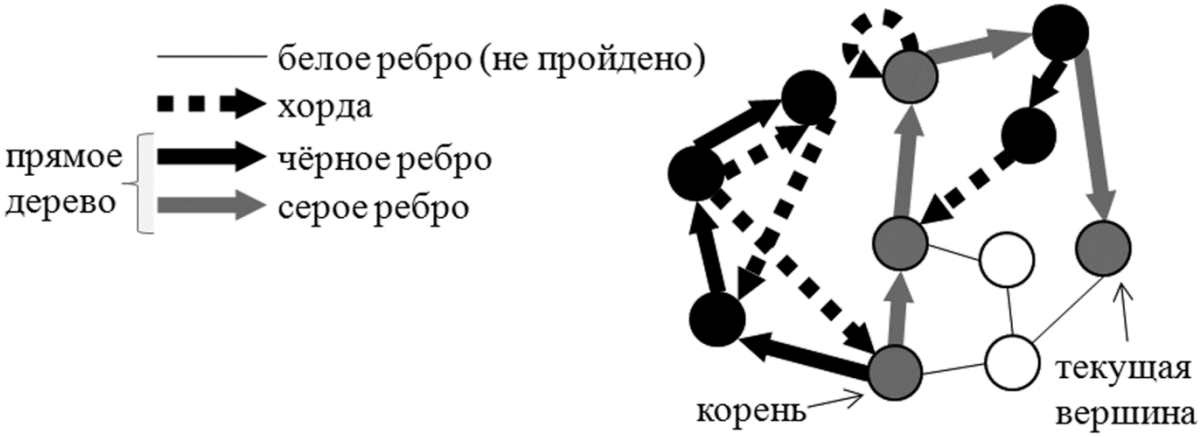



Рис. 2. Структура на графе во время обхода.

Для наглядности будем использовать три «краски»: белую, серую и чёрную. Непройденные рёбра, т.е. рёбра, по которым автомат не проходил, покрасим в белый цвет. Пройденные рёбра будут серого или чёрного цвета. Пройденное ребро ориентируем в том направлении, по которому автомат первый раз его прошёл. Эта операция не меняет заданную ориентацию рёбер в ориентированном графе, поскольку ориентированные рёбра проходятся только в направлении их заданной ориентации. Непройденная вершина, т.е. вершина, в которой автомат ещё не был, белая. Пройденные вершины серые или чёрные. В неориентированном графе пройденная вершина чёрная, если все инцидентные

ей рёбра не белые, т.е. автомат прошёл по всем инцидентным вершине рёбрам, иначе вершина серая. В ориентированном графе учитываются только те рёбра, которые выходят из вершины: пройденная вершины чёрная, если автомат прошёл по всем выходящим из вершины рёбрам, иначе вершина серая.

*Пройденный граф* – подграф, порождённый пройденными рёбрами (серыми и чёрными). *Прямое ребро* – это ребро, по которому автомат первый раз пришёл в какую-то вершину. Они порождают прямое дерево – остов пройденного графа, ориентированный от корня. Остальные пройденные рёбра будем называть *хордами* прямого дерева. Серое дерево – это поддерево прямого дерева, содержащее корень и все серые вершины, причём все его листья тоже серые. Рёбра серого дерева серые, остальные прямые рёбра чёрные.

### 2.3. Неизвестный граф, неизбыточные и свободные автоматы

Если граф известен (и помещается в память автомата), то можно вычислить его обход минимальной длины, а потом пройти по этому маршруту. Для неизвестного графа так сделать уже нельзя. Что значит, что граф неизвестен? Это значит, неизвестен заранее, до прохода по всем его рёбрам. На каждом шаге, выбирая очередное ребро, автомат «знает» только часть графа. Что он «знает»? Во-первых, пройденный граф, во-вторых, что-то о белых рёбрах, инцидентных серым вершинам. Что и когда автомат может «узнавать» об этих рёбрах?

Автомат будем называть *неизбыточным* [12], если он «узнаёт» степень вершины, когда первый раз в неё попадает. Поскольку пройденные рёбра он «знает», всегда известно и число белых рёбер, инцидентных вершине.

Автомат будем называть *свободным* [12], если он «узнаёт» о наличии или отсутствии белого ребра только при попытке пройти по нему. Если ребро есть, автомат идёт по нему, а если нет, то получает отказ и остаётся на месте.

И.Б. Бурдонов, А.С. Косачев
Исследование графа автоматом.
Программная инженерия. т.7, №11, 2016 , стр. 498-508.
11 стр.

---

Допуская вольность речи, мы будем говорить о цвете вершин и рёбер «с точки зрения» автомата: если вершина чёрная, но автомат ещё не «знает» об этом, то для него она серая; соответственно понимается серое дерево. Неизбыточный автомат «узнаёт» о том, что вершина чёрная тогда, когда проходит последнее инцидентное ей белое ребро, а свободный автомат – когда получает отказ на попытку пройти по ребру при условии, что все рёбра с меньшими номерами уже пройдены. Свободный автомат работает немного дольше, потому что может оказаться, что ему нужно лишний раз придти в вершину, чтобы убедиться, что она чёрная.

Поскольку автомат не знает, куда ведут белые рёбра, всё равно какое белое ребро он выберет в текущей серой вершине. Можно считать, что он проходит белые рёбра из серой вершины в порядке возрастания их номеров. В ориентированном графе это означает, что если из вершины выходят $k$ пройденных рёбер, то они имеют номера от 1 до $k$. В неориентированном графе это не обязательно так, потому что какое-то ребро $ab$ может быть пройдено первый раз не из вершины $a$, а из вершины $b$.

Для прохода по ребру из вершины $a$ в вершину $b$ автомат называет номер ребра в вершине $a$. Если граф неориентированный и такое ребро есть, то после прохода по нему автомат дополнительно «узнаёт» номер ребра в вершине $b$.

Мы будем оценивать алгоритмы обхода с точки зрения времени обхода (для одного автомата это, по сути, длина маршрута, который он проходит). Поэтому обход известного графа (неограниченным автоматом) нас будет интересовать только для оценки минимальной длины обхода, с которой мы будем сравнивать обход неизвестного графа различными автоматами.

И.Б. Бурдонов, А.С. Косачев
Исследование графа автоматом.
Программная инженерия. т.7, №11, 2016 , стр. 498-508.
11 стр.

---

### 2.4. Конечность степени/полустепени вершин графа

Исследование неизвестного графа двигающимся автоматом похоже на машину Тьюринга. Вершина графа соответствует ячейке ленты, а движение по ребру – движению по ленте влево или вправо. Тем самым, лента обычной машины Тьюринга – это граф специального вида.

Однако здесь возникает проблема для робота (конечного автомата). Дело в том, что номер ребра является выходным символом автомата. Для того чтобы автомат был конечным, в неориентированном графе степень вершины, а в ориентированном графе полустепень исхода вершины, должны быть ограничены сверху. Как обойти это ограничение?

Для этого преобразуем вершину ориентированного графа в цепочку вершин, каждая из которых имеет полустепень исхода не больше 2 (Рис. 3). Для машины Тьюринга это означает, что ячейка ленты превращается в цепочку ячеек конечной, но неограниченной длины, по которой можно перемещаться в вертикальном направлении. Получается что-то вроде ленты переменной ширины.

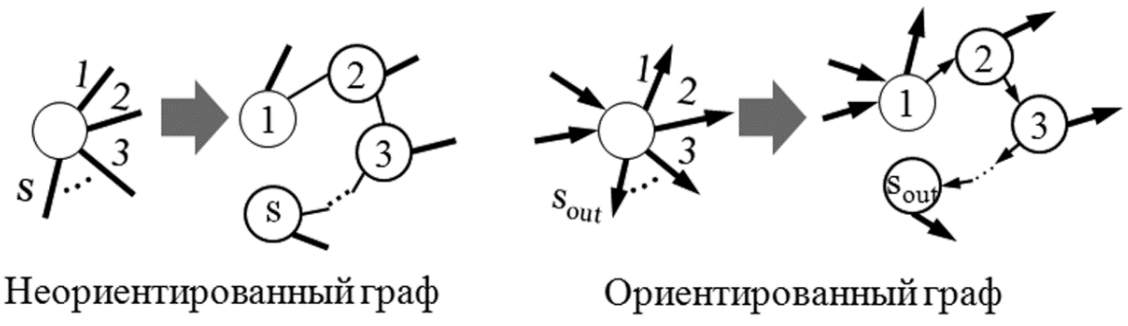



Рис. 3. Преобразование вершины в цепочку вершин.

### 2.5. Нумерованный граф

Если граф помещается в память автомата, то по мере обхода автомат может запоминать в своей памяти весь пройденный граф. Для этого достаточно нумеровать вершины в порядке их обнаружения, т.е. когда автомат по белому

ребру приходит в белую вершину, и записывать в вершине её номер. Тогда при проходе любого ребра автомат может узнать номера обеих вершин, инцидентных этому ребру. На самом деле номер вершины – это единственное, что нужно хранить в самой вершине. Всё остальное можно хранить в памяти автомата, индексируя номером вершины. Если вершины пронумерованы заранее и их номера в них записаны, то такой граф будем называть *нумерованным*. Тогда автомат будет только читать из вершины её номер и не будет в неё писать.

### 2.6. Классификация алгоритмов обхода графов

Резюмируя, скажем, что различные алгоритмы обхода графа автоматами классифицируются в зависимости от рассмотренных параметров графа и автомата. Прежде всего, граф может быть ориентированным или неориентированным. От этого существенно зависят как алгоритмы обхода, так и оценки их сложности. Во-вторых, граф известный или неизвестный. Известный граф нам нужен только для того, чтобы оценить минимальную длину обхода. Для неизвестного графа как раз и рассматриваются различные автоматы обхода. Эти автоматы могут быть неизбыточными или свободными, в зависимости от способа общения с графом. По размеру их памяти автоматы делятся на роботы, полуроботы и неограниченные автоматы. Также интересно различать первый обход графа и повторный, при котором можно использовать пометки на графе, оставленные автоматом при первом проходе.

Далее: один или несколько автоматов на графе. Зачем нужно несколько автоматов? Это либо решает проблему времени, уменьшая время обхода, либо решает проблему ограниченности памяти компьютера за счёт распределённого обхода. В данной, первой, статье мы ограничиваемся одним автоматом, а коллективы автоматов будут рассмотрены в двух последующих статьях.

И.Б. Бурдонов, А.С. Косачев
Исследование графа автоматом.
Программная инженерия. т.7, №11, 2016 , стр. 498-508.
11 стр.

---

Как уже говорилось во введении, есть две модели исследования графов автоматами. В первой модели автоматы двигаются по рёбрам графа, а вершины используются как ячейки памяти. Во второй модели автоматы неподвижно «сидят» в вершинах графа, а по рёбрам графа пересылаются сообщения от автомата к автомату. Первая модель рассматривается в первых двух статьях, а вторая – в третьей статье.

Кроме того, граф может быть недетерминированным – этот случай рассматривается во второй статье, или динамическим, т.е. меняющимся со временем, – этот случай изучается в третьей статье.

## 3. Неориентированный граф

### 3.1. Минимальная длина обхода

Обход неориентированного графа существует тогда и только тогда, когда граф связен. Минимальная длина обхода, т.е. обхода известного графа неограниченным автоматом, не более *2m-1*, где *m* число рёбер. Действительно, если каждое ребро удвоить, то граф становится эйлеровым (степени всех вершин чётные). Его эйлеров цикл соответствует обходу исходного графа длиной *2m*. Но последнее ребро можно не проходить, поскольку это был бы его повторный проход. Пример достижимости оценки на Рис. 4 слева.

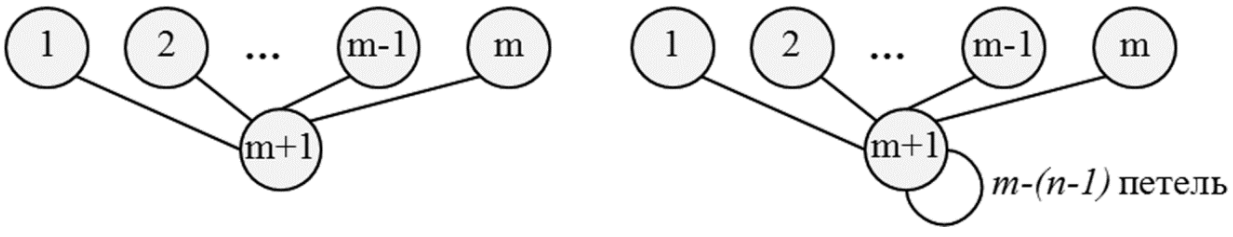



Рис. 4. Достижимость минимальной длины обхода неориентированного графа.

Если задано не только число рёбер *m*, но и число вершин *n*, то получается оценка *m+n-2*. Действительно, пусть *S* минимальное множество ребёр, которые нужно удвоить, чтобы получился эйлеров граф, а *r* –число рёбер в *S*. (Такое

множество $S$ определяется как минимальное паросочетание в полном графе, вершины которого – все вершины исходного графа с нечётной степенью, а вес ребра – длина минимального пути между вершинами.) Тогда минимальная длина обхода равна $m+r$, если корень не инцидентен рёбрам из $S$, или $m+r-1$ в противном случае. В первом случае проходится эйлеров цикл, а во втором – эйлеров путь с началом в корне. В $S$ нет циклов, поскольку любой цикл можно удалить без ущерба для чётности всех вершин. Поэтому $r \le n-2$, если корень не инцидентен ребру из $S$, и $r \le n-1$ в противном случае. Тем самым, минимальная длина обхода равна $m+n-2$. Пример достижимости этой оценки на Рис. 4 справа.

### 3.2. Алгоритм Тэрри

Для обхода неизвестных неориентированных графов существует хорошо известный алгоритм Тэрри [13]. Он реализуется свободным роботом, если степень вершины ограничена, или полуроботом для любых графов. По сути, это DFS-алгоритм, т.е. обход в глубину. В DFS-алгоритме серое дерево всегда состоит только из одного серого пути, который свободный робот размечает на графе: серое ребро отмечается в его конце. Каждое ребро автомат проходит 2 раза. Как работает алгоритм Тэрри? Два основных правила: 1) после прохода по хорде немедленно возвращаемся по ней назад (*откат по хорде*); 2) если идти некуда (текущая вершина стала чёрной), то идём по входящему серому ребру, т.е. в предыдущую вершину серого пути (*откат по дереву*). Если предыдущей вершины нет, т.е. чёрным стал корень, то конец обхода.

### 3.3. Оптимизации алгоритма Тэрри

В Таб. 1 сведены возможные оптимизации алгоритма Тэрри. Эта таблица демонстрирует различие между типами автоматов, а также дополнительные возможности повторных обходов графа.

И.Б. Бурдонов, А.С. Косачев
Исследование графа автоматом.
Программная инженерия. т.7, №11, 2016 , стр. 498-508.
11 стр.

---

Таб. 1. Оптимизации алгоритма Тэрри.

| обход выполняет | *A* | *B* | *C* | *D* | *E* | *F* | |
|---|---|---|---|---|---|---|---|
| неограниченный автомат или полуробот | да | да | да | да | да | нет | неизбыточный |
| | да | да | да | нет | нет | нет | свободный |
| робот | нет | да | да | нет | да | нет | неизбыточный |
| | нет | да | да | нет | нет | нет | свободный |
| свободный робот на повторном обходе № | 3 | | | | | 2 | |

*A = «Нет отката по всем петлям»,*
*B = «Нет отката по петлям с одинаковым номером в начале и конце»,*
*C = «Нет отката по петлям в корне (или в конечным множестве вершин)»,*
*D = «Нет отката по чёрному пути»,*
*E = «Нет отката по чёрному пути, если все его рёбра имеют максимальные номера (или число рёбер, имеющих большие номера, ограничено)*,
*F = «Нет отката по самой правой ветви прямого дерева».*

Оптимизации *A* , *B* и *C* – это проход по петле только один раз. Для этого автомат должен распознавать петли. Оптимизацию *A* (распознавание всех петель) может делать неограниченный автомат или полуробот, запоминая номер вершины, из которой он выходит по ребру, и сравнивая с номером вершины, куда попал. Они совпадают только для петли. Если граф ненумерованный, полуробот может сам нумеровать вершины, присваивая вершине очередной номер тогда, когда первый раз в неё попадает. Робот не может использовать номер вершины (он неограничен), но может не проходить петли в корне, если специально пометит корень. В общем случае робот, используя ограниченное множество разных пометок, может распознавать петли в ограниченном множестве вершин (оптимизация *C*). Также робот может распознать петли во всех вершинах, если петля имеет не два разных номера в вершине, а один (оптимизация B). Когда робот проходит по белому ребру, которое ведёт в другую вершину, то в ней это ребро ещё помечено как белое. А если он «видит», что ребро уже не белое, значит это петля.

И.Б. Бурдонов, А.С. Косачев
Исследование графа автоматом.
Программная инженерия. т.7, №11, 2016 , стр. 498-508.
11 стр.

---

Также робот может распознавать все петли при повторных обходах, то есть, используя пометки, оставленные на предыдущих обходах графа. Это можно сделать, начиная с третьего обхода. На первом обходе размечаются все хорды. Это можно сделать, т.к. по каждой хорде автомат идёт два раза подряд: первый раз ребро помечается как хорда в своём конце, а второй раз в начале. При втором обходе, перед тем как идти по белой хорде, автомат помечает вершину. Если, пройдя белое ребро, автомат приходит в вершину с пометкой, то это петля, и автомат отмечает это ребро. В любом случае, поскольку автомат возвращается по хорде, он снимает пометку в вершине. Поэтому, начиная с третьего обхода, петли проходятся по одному разу – это отмеченные рёбра.

Оптимизации *D*, *E* и *F* касаются отката по дереву. Когда серый путь становится чёрным, по нему можно не делать отката. Любую такую ситуацию может распознать неизбыточный неограниченный автомат или полуробот (оптимизация *D*). Для этого он должен «помнить» число белых рёбер, инцидентных вершинам серого пути. Путь становится чёрным, когда это число становится равным нулю. Свободный автомат (даже неограниченный) ничего не «знает» о числе белых рёбер: пока он не попробовал выйти из вершины по ребру, он не «знает», есть такое ребро или нет. Соответственно, для него все вершины серого пути серые, кроме, быть может, последней вершины, ставшей чёрной. У робота память ограничена, поэтому неизбыточный робот может «помнить» только ограниченное число белых рёбер, инцидентных вершинам серого пути. Если число таких рёбер не превосходит ограничения робота, то робот не делает отката по дереву (оптимизация *E*). Зато свободный робот при повторном обходе может не делать отката по всей самой правой ветви прямого дерева (оптимизация *F*). Для этого при первом обходе размечаются хорды, а при втором обходе в каждой вершине в первую очередь проходятся хорды, а

уже потом прямые рёбра. Поэтому все рёбра самой правой ветви прямого дерева проходятся в последнюю очередь, что робот и запоминает.

Сочетание оптимизаций *C* и *E* для неизбыточного автомата (в таблице выделено серым фоном ячеек) гарантирует достижение оценки *2m-1*. Последнее ребро проходится один раз: либо петля в корне (оптимизация *C*), либо прямое ребро (оптимизация *E*).

### 3.4. BFS- и «жадный» алгоритмы

BFS-алгоритм, т.е. алгоритм обхода в ширину, тоже реализуется свободным роботом, если степень вершины ограничена, или полуроботом для любых графов. Чем он отличается от алгоритма Тэрри? Робот проходит хорды по одному разу. Пройдя по хорде, робот не возвращается, а снова пытается идти по белым рёбрам. Из-за этого в сером дереве может «вырасти» новая ветвь.

Когда текущая вершина становится чёрной, робот ищет серую вершину, двигаясь по серому дереву. Если робот находится в листе дерева, то он делает откат по дереву, т.е. идёт по нему вниз к корню, либо до серой вершины, либо до развилки. Рёбра, которые проходятся, становятся чёрными, т.е. удаляются из серого дерева. Если робот доходит до развилки, и она оказалась чёрной, он выполняет *поиск серой вершины*: идёт вверх по дереву до серой вершины. Если робот находится не в листе дерева, он сразу идёт вверх до серой вершины. Конец обхода, когда в сером дереве остался один корень и он «почернел».

Когда робот движется вверх по дереву в поиске серой вершины, он может «выбирать», по какому серому ребру ему двигаться, если из вершины выходит несколько таких рёбер. Этот выбор зависит от алгоритма робота: например, он может выбирать серое ребро с наименьшим номером или перебирать серые рёбра по циклу. На оценку длины обхода в наихудшем случае это не влияет.

Длина обхода $m+\boldsymbol{O}(n^2)$. Докажем это. Робот проходит по каждой хорде ровно один раз. По каждому прямому ребру, число которых равно $n-1$, придётся пройти 4 раза: 1) первый проход по белому ребру, 2) возврат в начало ребра, чтобы пометить его как прямое в его начале, 3) повторить проход вперёд, 4) при откате по ребру, который для каждого прямого ребра выполняется 1 раз, поскольку после этого ребро удаляется из серого дерева.

Кроме этого, выполняются поиски серых вершин. Такой поиск начинается в двух случаях: 1) после прохода по белой хорде в не листовую вершину, которая из-за этого становится чёрной; 2) после отката по дереву, если робот попадает в чёрную развилку дерева. Число первых случаев не более $n$, так как вершина только один раз становится чёрной. Число вторых случаев тоже не больше $n$, так как откат по серому ребру делается один раз (после чего ребро становится чёрным). Каждый поиск – это проход простого пути (без самопересечения), т.е. имеет длину не более $n$.

Тем самым, длина обхода не больше $[m-(n-1)]+[4(n-1)]+[n^2] = m+\boldsymbol{O}(n^2)$. Эта оценка достижима на графе, который изображён на Рис. 5.

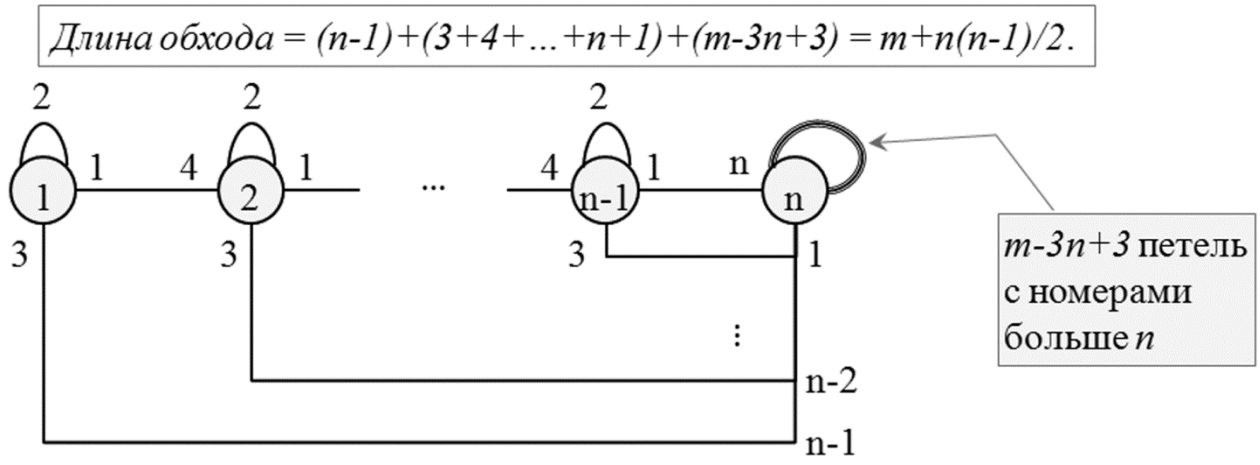



Рис. 5. Достижимость оценки длины обхода BFS-алгоритма.

Отличие «жадного» алгоритма от обхода в ширину в том, как делается поиск серой вершины. Вместо движения по серому дереву вниз, а потом вверх,

робот по всему пройденному графу сначала вычисляет кратчайший путь до серой вершины, а потом идёт по этому пути. Конечно, такое вычисление без движения по графу может делать только неограниченный автомат. В памяти полуробота и, тем более, робота граф полностью не помещается, поэтому им для вычисления кратчайшего пути придётся двигаться по графу, в том числе, по хордам. Поэтому для таких автоматов «жадный» алгоритм не имеет смысла.

Длина обхода неограниченным автоматом, работающим по «жадному» алгоритму, тоже не более $m+\boldsymbol{O}(n^2)$. Может быть, в этой оценке вместо $\boldsymbol{O}(n^2)$ должно стоять $\boldsymbol{O}(n)$, но это только гипотеза.

### 3.5. Сводка результатов по обходу неориентированного графа

В Таб. 2 представлена сводка результатов по обходу неориентированного графа (кроме оптимизаций для алгоритма Тэрри). Что следует отметить? Понятно, что обход неизвестного графа длиннее, чем известного. Однако в терминах числа рёбер разница всего лишь на единицу для свободного автомата при первом проходе: *2m* вместо *2m-1*. Если задано ещё и число вершин *n*, то алгоритм поиска в ширину и жадный алгоритм лучше алгоритма Тэрри при достаточно большом числе рёбер по сравнению с числом вершин: вместо *2m* будет $m + \boldsymbol{O}(n^2)$. Все алгоритмы реализуются роботом, но для жадного алгоритма это не имеет смысла (и оценка, очевидно, значительно выше).

Таб. 2. Длина обхода неориентированного графа.

<table>
<tr><td colspan="3">обход неограниченным автоматом известного графа<br>или повторный обход неизвестного графа</td><td>2m-1</td><td>m+n-2</td></tr>
<tr><td rowspan="4">неизвестный граф</td><td rowspan="2">свободный робот</td><td>DFS</td><td>2m</td><td></td></tr>
<tr><td>BFS</td><td></td><td>$m+\boldsymbol{O}(n^2)$</td></tr>
<tr><td>неизбыточный робот</td><td>DFS</td><td>2m-1</td><td></td></tr>
<tr><td>неограниченный неизбыточный автомат</td><td>«жадный»</td><td></td><td>$m+\boldsymbol{O}(n^2)$ ?</td></tr>
</table>

И.Б. Бурдонов, А.С. Косачев
Исследование графа автоматом.
Программная инженерия. т.7, №11, 2016 , стр. 498-508.
11 стр.

## 4. Ориентированный граф

Ориентированное ребро будем называть, как это принято, *дугой*. Дугу можно проходить только в направлении её ориентации. Напомним, что в ориентированном графе вершина становится чёрной, когда автомат прошёл не по всем инцидентным ей рёбрам, как в неориентированном случае, а по всем *выходящим* из вершины дугам. В чёрную вершину могут входить белые дуги.

### 4.1. Сильно-связные графы и графы 1-го и 2-го рода

Для существования обхода из любой вершины как начальной требуется сильная связность графа. Если начальная вершина фиксирована (корень), то граф должен быть 1-го рода [12] – цепочка компонентов сильной связности и разделяющих дуг, начальная вершина – в первом компоненте (Рис. 6).

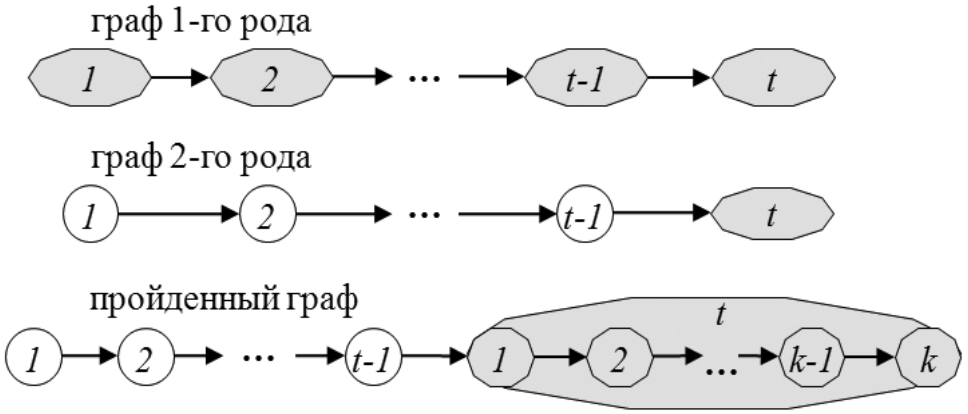



Рис. 6. Связность графов.

Если граф неизвестен, то он должен быть графом 2-го рода [12]: простой путь, ведущий в последний компонент сильной связности. Пройденный подграф является графом 1-го рода [12], а в конце обхода, конечно, совпадает со всем графом.

### 4.2. Минимальная длина обхода

Минимальный обход имеет длину $O(nm)$, где $n$ – число вершин, $m$ – число дуг. При любом линейном упорядочивании множества дуг после прохода

очередной дуги идём в начало следующей дуги, проходя простой путь длиной не более *n-1*. Пример достижимости оценки на Рис. 7 сверху. Здесь из вершины *n* в начальную вершину ведёт много кратных дуг.

На том же Рис. 7 снизу показан пример достижимости оценки для ограниченной полустепени исхода вершин. Это ограничение может быть меньше, чем число кратных дуг в примере сверху. Поэтому несколько последних вершин образуют сбалансированное по высоте дерево с ограниченной полустепенью исхода, из листьев которого дуги ведут уже в начальную вершину. Вообще же при ограничении полустепени исхода величиной *s* оценка становится $\boldsymbol{O}(n^2)$, т.к. $m \le sn$.

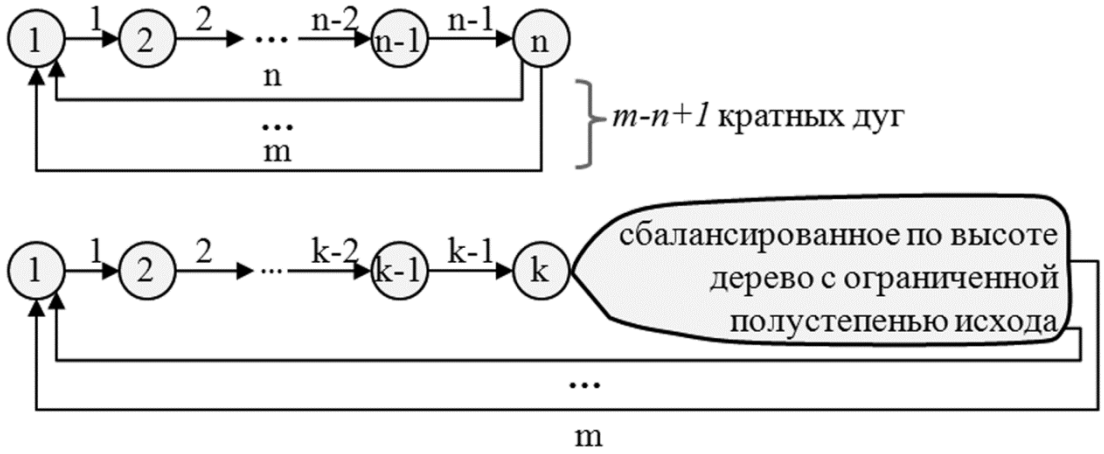



Рис. 7. Примеры достижимости минимальной оценки.

Более точную оценку можно получить, если вместо числа вершин *n* использовать диаметр графа *d* как максимальную длину пути в графе. Очевидно, что $d \le n-1$. Оценка получается $\boldsymbol{O}(dm)$.

### 4.3. Три алгоритма

DFS-алгоритм может применяться и в случае ориентированного графа. Дуга серого пути помечается в её начале. Но тут есть проблема с откатом, потому что автомат не может вернуться по дуге в обратном направлении.

Вместо этого он должен пройти некоторый ориентированный путь из конца дуги в её начало. Для этого строится лес *обратных деревьев*: по одному на каждый компонент сильной связности пройденного графа (Рис. 8).

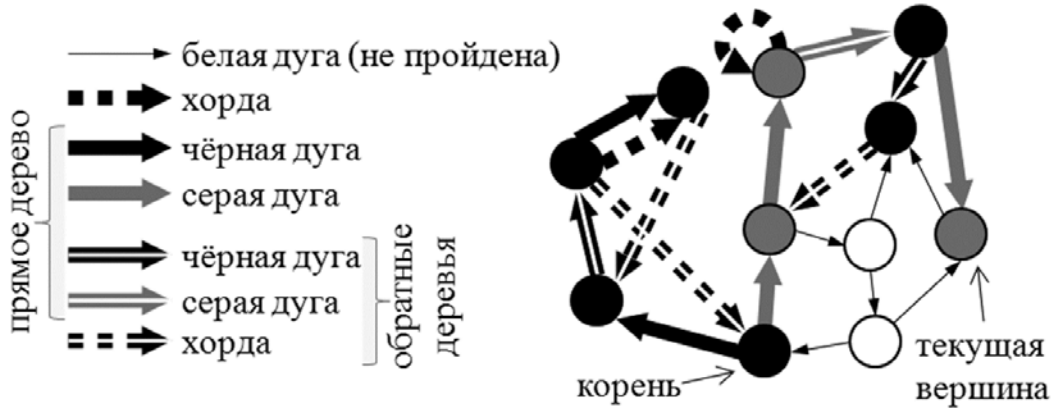



Рис. 8. Структура на графе во время обхода.

В каждом компоненте обратное дерево является остовом, ориентированным к корню компонента. В этот корень входит разделяющая дуга из предыдущего компонента, в первом компоненте это корень графа. Очевидно, все корни обратных деревьев лежат на сером пути. Обратная дуга отмечена в её начале.

Откат выполняется по дуге $ab$, если это хорда или последняя дуга серого пути. В первом случае серый путь заканчивается в $a$, а во втором случае – в $b$. Для отката всегда есть цикл, состоящий из дуги $ab$ и двух путей: 1) путь по обратным дугам от вершины $b$ до пересечения с серым путём, вершины $c$, лежащей на пути не выше $a$ (если $ab$ последняя дуга серого пути, то $c \neq b$), и 2) отрезок серого пути от вершины $c$ до вершины $a$. Вот по этому циклу автомат и ищет вершину $a$. Неограниченный автомат всегда «знает» вершину $a$ – это либо последняя вершина серого пути (откат по хорде), либо предпоследняя вершина (откат по дереву). Полуробот может нумеровать вершины серого пути 1,2,... и запоминать номер последней вершины. Для этого ему достаточно памяти

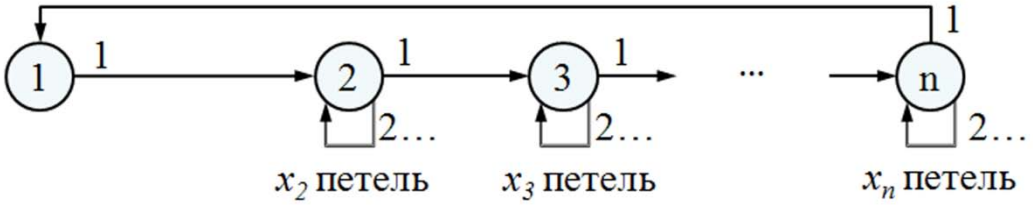
порядка *logn*. Если полуробот проходит хорду, то номер её начала – это запомненный номер вершины. Если полуробот попадает в белую вершину, то ей присваивается номер на единицу больше и запоминается. Для отката по дереву в предпоследнюю вершину серого пути полуробот «знает», что её номер на единицу меньше запомненного, а после отката запоминает этот уменьшенный на 1 номер. Откат не делается, если запомненный номер равен 1, то есть это номер корня, в этом случае конец обхода.

При откате по хорде *ab* нужно корректировать лес обратных деревьев, если на отрезке серого пути *ca* имеется корень обратного дерева, отличный от *c*. В каждой вершине этого отрезка, начиная с первого попавшегося корня обратного дерева, обратной дугой становится выходящая серая дуга, а в последней вершине *a* обратной дугой становится пройденная хорда *ab*. Эту коррекцию может делать и полуробот.

Неизбыточный автомат мог бы делать откат по серому пути не на одну, последнюю, дугу пути, а на весь постфикс серого пути, начиная с последней серой вершины. Правда, полуроботу для этого придётся несколько раз проходить цикл дуг: сначала, чтобы узнать номер последней серой вершины на сером пути, потом, чтобы отметить дуги постфикса как чёрные, и третий раз, чтобы придти в последнюю серую вершину.

В BFS-алгоритме нет отката по хорде, и серое дерево состоит не из одного пути, оно может ветвиться. После прохода по хорде автомат двигается по обратным дугам до пересечения с серым деревом, а затем по той или иной ветви серого дерева до серой вершины. Кроме того, автомат должен откорректировать обратные деревья и пометить пройденную дугу как хорду. Неограниченный автомат может это сделать в своей памяти, но памяти полуробота для этого может не хватить, поэтому ему нужно использовать

память вершин. Полуробот должен, во-первых, двигаться по обратным дугам не до первого пересечения с серым деревом, а до корня обратного дерева, а, во-вторых, из корня двигаться в начало хорды, корректируя обратные деревья и в конце отмечая хорду в её начальной вершине. Из-за этого, как правило, BFS-алгоритм для полуробота не даёт преимуществ по сравнению с DFS-алгоритмом. Хотя, конечно, возможны случаи, когда «лишняя» работа после прохода хорды отсутствует, например, если хорда – это петля. Пример на Рис. 9.




$x_i > 0.\ x_2+x_3+\ldots+x_n = m-n$
$L_{dfs} = n + n(x_2+x_3+\ldots+x_n) = n + n(m-n) = \Theta(mn).$
$L_{bfs} = n + (1+x_2+1+x_3+\ldots+1+x_n) = n + n-1 + m-n = \Theta(m).$

Рис. 9. Пример преимущества BFS над DFS для ориентированного графа.

«Жадный» алгоритм не использует ни серое дерево, ни прямое или обратное деревья. Каждый раз, когда текущая вершина оказывается чёрной, автомат выполняет поиск серой вершины: строит по пройденному графу кратчайший путь в серую вершину и проходит его. Как и в случае неориентированного графа «жадный» алгоритм имеет смысл только для неограниченного автомата.

Очевидно, что оценка длины обхода каждого из этих трёх алгоритмов в наихудшем случае совпадает с оценкой минимального обхода $O(nm)$ или $O(dm)$.

### 4.4. Робот

Задачу обхода ориентированного графа роботом впервые поставил Рабин в устной лекции в 1967 г. Некоторые работы по этой теме приведены в Таб. 3.

Таб. 3. Обход роботом неизвестного ориентированного графа.

И.Б. Бурдонов, А.С. Косачев
Исследование графа автоматом.
Программная инженерия. т.7, №11, 2016 , стр. 498-508.
11 стр.

---

| когда | кто | длина обхода | алгоритм | откат по циклу длины $k$ |
|---|---|---|---|---|
| 1967 | M.O.Rabin | | | постановка задачи |
| 1970 | И.Б.Бурдонов | $\boldsymbol{O}(nm+n^3)$ | DFS | Простой $\boldsymbol{O}(k^2)$ |
| 1971 | И.Б.Бурдонов | $\boldsymbol{O}(nm+n^2\log n)$ | BFS | Простой $\boldsymbol{O}(k^2)$ |
| 1993 | Y.Afek, E.Gafni | $\boldsymbol{O}(nm+n^2\log n)$ | DFS | Логарифмический $\boldsymbol{O}(k\log k)$ |
| 2003 | И.Б.Бурдонов | $\boldsymbol{O}(nm+n^2\log\log n)$<br>$\boldsymbol{O}(dm+dn\log\log n)$ | BFS | Суммарно: по хордам $\boldsymbol{O}(dm)$,<br>по дереву $\boldsymbol{O}(dn+dn\log\log n)$ |
| 2003 | И.Б.Бурдонов | $\boldsymbol{O}(nm+n^2\log^* n)$<br>$\boldsymbol{O}(dm+dn\log^* n)$ | BFS,<br>повторный | Суммарно: по хордам $\boldsymbol{O}(dm)$,<br>по дереву $\boldsymbol{O}(dn+dn\log^* n)$ |

$\log^* n$ – решение неравенства $1 \le \log^t(n) < 2$ относительно $t$,
где $\log^t = \log \circ \log \circ \ldots \circ \log$ (знак суперпозиции $\circ$ применяется $t-1$ раз).

Со временем выяснилось, что основная проблема для робота – это проблема отката, то есть возврата из конца дуги в её начало. Для этого используется цикл дуг, о котором шла речь в п. 4.3. Проходя такой цикл, робот при откате по хорде корректирует обратные деревья и помечает хорду, а при откате по дереву удаляет дугу из серого дерева.

Но как найти начальную вершину дуги? Она никак специально не помечена и робот не может запоминать её номер, поскольку его память ограничена. Самый простой способ отката предполагает проход по циклу дуг столько раз, какова длина цикла: одна из вершин цикла помечается, и при каждом проходе цикла пометка сдвигается на следующую вершину цикла. Первый робот, основанный на DFS-алгоритме и использующий такой простой откат, предложен в 1970 г. (см. в [14]). Суммарно на откаты дополнительно тратилось $n^3$ проходов дуг. В следующем году был предложен робот на основе BFS-алгоритма, что дало суммарное время отката $n^2 \log n$ (см. в [14]).

В 1994 г. появилась работа Афека и Гафни [11], в которой для DFS-алгоритма применялся логарифмический откат: все вершины в цикле помечаются, а потом, с учётом четности их числа пометки удаляются через одну, и вычисляется чётность числа оставшихся пометок. Это дало такую же

оценку $n^2 logn$. В 2003 г. опубликована работа [14], в которой BFS-алгоритм «скрещен» с логарифмическим откатом, что дало оценку $n^2 loglogn$.

Суммарное время откатов сокращается за счёт того, что помечаются не все вершины цикла, а только *значимые*. Что такое значимая вершина? Единственное требование: искомая вершина значимая. Откат по хорде делается тогда, когда робот первый раз по ней прошёл, поэтому значимы только начальные вершины однократно пройденных дуг цикла. Это даёт суммарную оценку откатов по хордам порядка $nm$.

Откат по дереву выполняется не на одну дугу, а до серой вершины или развилки дерева, они и считаются значимыми. Для BFS-алгоритма это как раз и даёт наилучшую на сегодняшний день оценку $n^2 loglogn$. В том же 2003 г. предложен способ отката по дереву, который назван *цифровым* [15], с уменьшенной суммарной оценкой (последняя строка Таб. 3). К сожалению, это не значит, что теперь второе слагаемое в оценке длины обхода тоже уменьшается. Дело в том, что пройденный граф, как сказано выше, – это граф 1-го рода, поэтому в нём не один остов, ориентированный к корню, а лес деревьев по одному в каждом компоненте сильной связности. Из-за этого длина обхода в общем случае не меняет свой порядок при цифровом откате. Уменьшенная оценка получается только тогда, когда граф специально упорядочить, то есть не при произвольной нумерации дуг, исходящих из вершины. В то же время такое упорядочивание можно сделать при первом обходе роботом, так повторный обход выполняется быстрее.

## 5. Заключение

Алгоритмы обхода графа одним автоматом, двигающимся по рёбрам графа, могут использоваться (и уже используются) в задачах исследования систем, моделируемых графами. К таким системам относятся компьютерные сети и

сеть интернета. Кроме того, важнейшей областью применения является тестирование программных и аппаратных систем на основе формальных моделей, сводимых к графам переходов. Практическое использование таких алгоритмов предъявляет к ним требования по объёму памяти и по времени работы. Размер памяти отражается в понятиях неограниченного автомата, полуробота и робота (конечного автомата), а время работы оценивается как длина обхода, выполняемого автоматом.

Дальнейшим обобщением этой задачи является исследование графов коллективом взаимодействующих автоматов. Ключевыми вопросами здесь являются типы графов, включая недетерминированные графы и графы, динамически меняющиеся со временем, а также способы взаимодействия автоматов между собой и с графом. Модель коллектива двигающихся автоматом будет рассмотрена во второй статье серии, а модель неподвижных автоматов в вершинах графа, обменивающихся между собой сообщениями, пересылаемыми по рёбрам графа, – в третьей статье.

**Литература**

1. https://ru.wikipedia.org/wiki/сетевая топология (дата обращения 31.05.2016).
2. Райгородский А. М. Математические модели интернета // Квант. — 2012. — № 4. — С. 12–16.
3. Milner R. A Calculus of Communicating Processes. // LNCS. Springer-Verlag. — 1980. Vol. 92.
4. Hoare C.A.R. Communicating Sequential Processes. // Englewood Cliffs, NJ: Prentice Hall International. — 1985.
5. Lynch N. Tuttle M. An Introduction to Input/Output Automata. // CWI Quart. — 1989. Vol. 2, № 3. — P. 219–246.

И.Б. Бурдонов, А.С. Косачев
Исследование графа автоматом.
Программная инженерия. т.7, №11, 2016 , стр. 498-508.
11 стр.

---

6. Euler L. Solutio Problematis ad Geometriam Situs Pertinentis. // Commentarii Academiae Scientarum Imperialis Petropolitanae. — 1736. — № 8. — P. 128–140.
7. Bernot G., Gaudel M.C., Marre B. Software Testing based on Formal Specifications: a Theory and a Tool. // Software Engineering Journal/ — 1991. — Vol. 6, № 6. — P. 387–405.
8. Fujiwara S., Bochmann G.v., Khendek F., Amalou M., Ghedamsi A. Test Selection Based on Finite State Models. // IEEE Transactions on Software Engineering. — 1991. — Vol. 17, № 6. — P. 591–603.
9. Tretmans J. A Formal Approach to Conformance Testing. // Proceedings of the IFIP TC6/WG6.1 Sixth International Workshop on Protocol Test systems VI. — 1993. — P. 257–276.
10. Lee D., Yannakakis M. Principles and Methods of Testing Finite State Machines – A Survey. // Proceedings of the IEEE 84. — 1996. — № 8. — P. 1090–1123.
11. Y. Afek and E. Gafni. Distributed Algorithms for Unidirectional Networks. // SIAM J. Comput. — 1994. — Vol. 23, № 6. — P. 1152-1178.
12. Бурдонов И.Б., Косачев А.С., Кулямин В.В. Неизбыточные алгоритмы обхода ориентированных графов. Детерминированный случай. // Программирование. — 2003. — №5. — С. 59–69.
13. Оре О. Теория графов. — М.: Наука, 1968. — С. 59, 68.
14. И.Б.Бурдонов. Обход неизвестного ориентированного графа конечным роботом. // Программирование. — 2004. — №4. — С.11-34.
15. И.Б.Бурдонов. Проблема отката по дереву при обходе неизвестного ориентированного графа конечным роботом. // Программирование. — 2004. — №6. — С. 6-29.