Проблема отката в ориентированной распределенной системе

И.Б. Бурдонов <igor@ispras.ru>[1]
А.С. Косачев <kos@ispras.ru>
Институт системного программирования РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. Для распределенной системы, в основе которой лежит ориентированный граф без кратных ребер и петель, рассматривается проблема отката (backtracing problem): как передать сообщение от конца дуги в ее начало. Ставится задача создания на графе структуры, позволяющей передавать сообщение из конца любой дуги в ее начало по кратчайшему пути. Такая структура в каждой вершине a задаётся отображением номера вершины b в номер исходящей дуги, через которую проходит кратчайших путь от a к b. В частности, такое отображение позволяет симулировать в ориентированных распределенных системах алгоритмы решения задач на графе, разработанные для неориентированных распределенных систем. Это увеличивает время работы таких алгоритмов (в тактах, где время пересылки сообщения по дуге не превосходит 1 такт) не более чем в k раз, где k диаметр графа, $k < n$, и n – число вершин графа. В разделе 2 описывается используемая асинхронная модель распределенной системы. Раздел 3 содержит основные определения и обозначения, а раздел 4 – постановку задачи. В разделе 5 описываются два вспомогательных алгоритма коррекции поддеревьев, применение которых позволяет строить остовные деревья кратчайших путей: прямого дерева, ориентированного от корня графа, и обратного дерева, ориентированного к корню графа. Раздел 6 содержит описание различных способов передачи сообщений по графу. В разделе 7 предлагаются два алгоритма построения в памяти автомата корня графа описаний прямого и обратного остовных деревьев кратчайших путей, а в разделе 8 – основанные на них алгоритмы построения требуемого отображения: «быстрый» алгоритм с оценками $T=\boldsymbol{O}(n)$ и $N=\boldsymbol{O}(n)$ и «экономный» алгоритм с оценками $T=\boldsymbol{O}(n^2)$ и $N=\boldsymbol{O}(1)$, где T – время (в тактах) работы алгоритма, N – число сообщений, одновременно передаваемых по дуге. В разделе 9 доказано, что эти оценки времени не улучшаемые. В разделе 10 «быстрый» алгоритм модифицируется для синхронной модели с $N=1$. Заключение подводит итоги и намечает направления дальнейших исследований.

Ключевые слова: ориентированный граф; корневой граф; нумерованный граф; распределенные алгоритмы; задачи на графах; проблема отката; кратчайшие пути.

DOI: 10.15514/ISPRAS-2018-30(2)-9

[1] Работа частично поддержана проектом РФФИ № 17-07-00682 А.

Для цитирования: Бурдонов И.Б., Косачев А.С. Проблема отката в ориентированной распределенной системе. Труды ИСП РАН, том 30, вып. 2, 2018 г., стр. 167-194. DOI: 10.15514/ISPRAS-2018-30(2)-9

1. Введение

Распределенная система – это граф, в вершинах которого располагаются вычислительные единицы (автоматы), которые могут обмениваться между собой сообщениями, посылаемыми по ребрам графа. Если граф ориентированный, то ориентированное ребро называется дугой и соответствует симплексному каналу передачи сообщений: сообщение может передаваться по дуге только в одном направлении: от начала дуги к концу дуги. Это создает проблему отката (*backtracking problem*): как передать сообщение от конца дуги в ее начало?

Первоначально эта проблема изучалась в контексте обхода ориентированного графа роботом (конечным автоматом), двигающимся по дугам графа [1][2][3]. Модель с одним двигающимся роботом эквивалентна «инвертированной» модели с одним сообщением, передаваемым по графу, и идентичными роботами, находящимися в вершинах графа: робот в вершине, принимая сообщение по некоторой входящей дуге, посылает одно сообщение по одной исходящей дуге или завершает работу алгоритма. Длина обхода ориентированного графа без кратных ребер равна $\boldsymbol{O}(n^2)$, где n – число вершин. Однако из-за проблемы отката оценка алгоритмического обхода увеличивается, в зависимости от алгоритма, до $\boldsymbol{O}(n^2 log n)$ [1], $\boldsymbol{O}(n^2 log log n)$ [2], а наилучший известный результат равен $\boldsymbol{O}(n^2 log^* n)$ [3], где log^* – итерированный логарифм (количество итеративных логарифмирований аргумента, необходимых для того, чтобы результат стал меньше или равен 1).

В данной работе мы исследовали проблему отката для распределенной системы, в которой может быть несколько сообщений, одновременно передаваемых по дугам графа. Граф предполагается *нумерованным*: вершинам присвоены номера от 1 до n. Граф *корневой*, т.е. имеется выделенная вершина, называемая *корнем* графа, с которой начинается и в которой заканчивается выполнение алгоритма. Граф *упорядоченный*: дуги, выходящие из вершины перенумерованы от 1 до полустепени исхода этой вершины. Заметим, что упорядоченность графа необходима для того, чтобы автомат в вершине мог указывать исходящую дугу, по которой он хочет передать сообщение. Сообщение перемещается по дуге за время не более 1 такта.

Мы поставили задачу создания на графе структуры, позволяющей передавать сообщение из конца любой дуги в ее начало по кратчайшему пути, т.е. за минимальное время в наихудшем случае, когда по каждой дуге сообщение пересылается 1 такт. В памяти автомата каждой вершины a эта структура задается *отображением* номера вершины b в номер исходящей дуги, через которую проходит кратчайший путь от a к b. В частности, такое отображение позволяет симулировать в ориентированных распределенных системах

алгоритмы решения задач на графе, разработанные для неориентированных распределенных систем (например, [4][5]). Вместо прохода по одной дуге в обратном направлении проходится ориентированный путь из конца дуги в её начало, что увеличивает время работы таких алгоритмов не более чем в k раз, где $k < n$ диаметр графа (максимальное расстояние между вершинами).

В разд. 2 описывается используемая асинхронная модель распределенной системы. Разд. 3 содержит основные используемые определения и обозначения, а разд. 4 – постановку задачи. В разд. 5 описываются два вспомогательных алгоритма коррекции поддеревьев, применение которых в последующих алгоритмах позволяет строить остовные деревья кратчайших путей: прямого дерева, ориентированного от корня графа, и обратного дерева, ориентированного к корню графа. Разд. 6 содержит описание пяти различных способов передачи сообщений по графу, которые используются в алгоритмах, описываемых ниже.

В разд. 7 предлагаются два алгоритма построения в памяти автомата корня графа описаний прямого и обратного остовных деревьев кратчайших путей. Эти алгоритмы отличаются соотношением времени T работы и числа N сообщений, одновременно передаваемых по дуге: в «быстром» алгоритме $T=\boldsymbol{O}(n)$ и $N=\boldsymbol{O}(n)$, а в «экономном» алгоритме $T=\boldsymbol{O}(n^2)$ и $N=\boldsymbol{O}(1)$.

Разд. 8 содержит описание двух основных алгоритмов построения требуемого отображения, т.е. отображения в каждой вершине a номера другой вершины b в номер первой дуги на кратчайшем пути из a в b. «Быстрый» алгоритм имеет оценки $T=\boldsymbol{O}(n)$ и $N=\boldsymbol{O}(n)$, «экономный» алгоритм имеет оценки $T=\boldsymbol{O}(n^2)$ и $N=\boldsymbol{O}(1)$. В разд. 9 доказано, что эти оценки не улучшаемы. Разд. 10 посвящён синхронной модели, в которой любое сообщение перемещается по любой дуге ровно за 1 такт. Заключение подводит итоги и намечает направления дальнейших исследований.

2. Модель

Граф распределенной системы предполагается ориентированным, сильно связным, нумерованным, корневым и упорядоченным без кратных дуг и петель. *Емкостью* дуги называется число сообщений, которые могут одновременно перемещаться по дуге. В рассматриваемой модели предполагается неограниченная емкость дуги. Это означает, что автомат в вершине не получает каких-либо «извещений» о том, что исходящая дуга уже полностью «заполнена» или, наоборот, «освободилась». Но для предлагаемых алгоритмов мы будем давать оценку числа сообщений на дуге.

Будем считать, что в одном срабатывании автомат принимает ровно одно сообщение ровно по одной входящей дуге и посылает по каждой исходящей дуге не более одного сообщения. Для удобства описания алгоритма мы будем иногда говорить, что в одном срабатывании автомат посылает по одной исходящей дуге несколько (но конечное число) сообщений $m_1,m_2,...m_e$, имея в виду, что они «склеиваются» в одно сообщение $m = m_1 \cdot m_2 \cdot ... m_e$. Число e таких

сообщений во всех предлагаемых ниже алгоритмах не превосходит 3. Принимая такое «склеенное» сообщение m, автомат запоминает его в своей памяти и обрабатывает его компоненты $m_1,m_2,...m_e$ последовательно, как если бы он принял их в последовательных e срабатываниях. Мы будем пренебрегать временем срабатывания автомата в вершине, т.е. будем оценивать сложность алгоритмов как время перемещения сообщений по дугам.

Для краткости там, где это не приведёт к недоразумениям, мы будем вместо «автомат в вершине» или «память автомата в вершине» говорить просто «вершина» или «память вершины». Память вершины будет рассматриваться как набор *переменных*, а сообщение – как набор *параметров*.

Мы будем давать следующие оценки алгоритмов: T – время работы алгоритма, A – размер памяти вершины как сумма размеров (в битах) переменных, M – размер сообщения как сумма размеров (в битах) параметров сообщения, N – число сообщений, одновременно передаваемых по одной дуге. Время перемещения сообщения по одной дуге считается ограниченным сверху одним тактом.

3. Определения и обозначения

Определения:

- *Маршрут* – последовательность дуг графа, в которой конец не последней дуги совпадает с началом следующей дуги.
- *Путь* – маршрут, в котором никакие две дуги не имеют общего конца, и конец последней дуги не совпадает с началом первой дуги.
- *Расстояние* от вершины a до вершины b – длина кратчайшего пути от вершины a до вершины b.
- *Прямое дерево* – дерево с выделенным корнем дерева, в котором все дуги ориентированы от корня дерева.
- *Обратное дерево* – дерево с выделенным корнем дерева, в котором все дуги ориентированы к корню дерева.
- *Прямое дерево кратчайших путей* – такой подграф, являющийся прямым деревом, что расстояние от корня дерева до каждой вершины дерева по дереву и по графу одинаковы.
- *Обратное дерево кратчайших путей* – такой подграф, являющийся обратным деревом, что расстояние от каждой вершины дерева до корня дерева по дереву и по графу одинаковы.
- *Остов графа, остовное дерево* – подграф, являющийся деревом и содержащий все вершины графа.

Далее граф G, лежащий в основе распределенной системы, предполагается ориентированным сильно-связным простым нумерованным упорядоченным корневым графом. В переменных вершин и параметрах сообщений описание вершины – это ее номер, а описание дуги – это тройка чисел (a,i,b), где a – номер

начала (начальной вершины) дуги, i – номер дуги в вершине a, b – номер конца (конечной вершины) дуги. Связный подграф будем задавать множеством таких троек чисел, представляющих все его дуги. В тривиальном случае, когда связный подграф не имеет дуг, он состоит из одной вершины, и будет задаваться синглетоном, содержащим номер этой вершины. Маршрут будет задаваться последовательностью дуг (троек чисел); маршрут нулевой длины – пустая последовательность $\langle\rangle$.

Обозначения:

- r – корень графа.
- $d^-(a)$ – полустепень исхода вершины a (число исходящих из нее дуг).
- $d^+(a)$ – полустепень захода вершины a (число входящих в нее дуг).
- $a—i\to b$ – дуга, задаваемая тройкой (a,i,b).
- $a—i\to$ – означает, что для некоторого b существует дуга $a—i\to b$.
- $a\to b$ – означает, что для некоторого i существует дуга $a—i\to b$.
- $V(H)$ – множество вершин, инцидентных дугам множества дуг H.

4. Постановка задачи

Для того, чтобы в графе G с n вершинами из каждой вершины можно было попасть в каждую вершину по кратчайшему пути, нужно отображение $f(G) : [1..n]\times[1..n]\to[0..n\text{-}1]$, которое каждой паре вершины с номерами a и b ставит в соответствие номер дуги, с которой начинается кратчайший путь из a в b, если $a\ne b$, или 0 в противном случае. В каждой вершине $a\in[1..n]$ достаточно хранить часть этого отображения как отображение $f_a(G) : [1..n]\to[1..d^-(a)]$ такое, что $\forall b\in[1..n]\ f_a(G)(b) = f(G)(a,b)$. Если граф G подразумевается, мы будем вместо $f(G)$ и $f_a(G)$ писать просто f и f_a.

Размер памяти вершины a, необходимый для хранения такого отображения, равен $\boldsymbol{O}(n\log d^-(a)) = \boldsymbol{O}(n\log n)$, что на порядок меньше памяти для хранения всего неупорядоченного графа $\boldsymbol{O}(n^2)$ (матрица смежности $[1..n]\times[1..n]\to\{0,1\}$) и, тем более, упорядоченного графа $\boldsymbol{O}(n^2\log n)$ (матрица смежности с указанием номеров дуг $[1..n]\times[1..n]\to[0..n]$) [6]. Поскольку отображение имеет размер $\boldsymbol{O}(n\log n)$, то в любом алгоритме размер памяти вершины $A = \boldsymbol{\Omega}(n\log n)$.

Мы ставим задачу: построить отображение f и разместить в каждой вершине a отображение f_a. В этой статье мы предлагаем четыре результата, графически изображенные на рис. 1 и описываемые в следующих разделах.

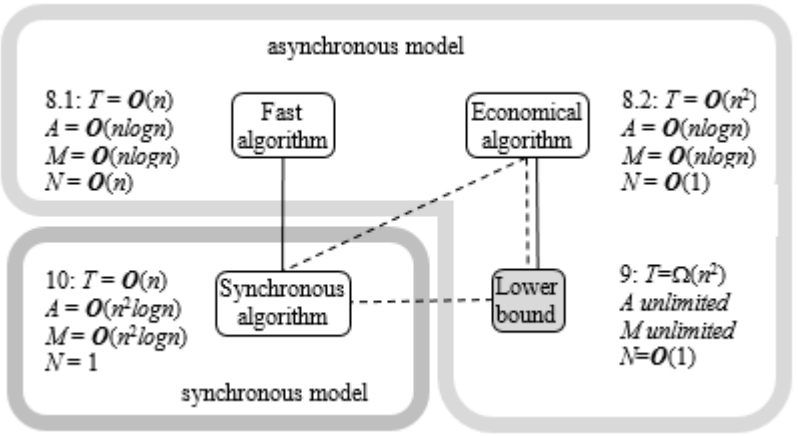

Рис. 1. Алгоритмы построения отображения и их оценки
Fig. 1. Algorithms for constructing a map and their estimates

5. Коррекция деревьев

5.1. Коррекция прямого дерева

Пусть все вершины графа H достижимы из вершины x, и имеется прямое дерево $D\subseteq H$ с корнем в x. Определим коррекцию дерева, когда становится известным о существовании дуги $a—i\to b\in H\setminus D$ с началом в D, т.е. $a\in V(D)$. Если $b\notin V(D)$, то дуга $a—i\to b$ добавляется в D: $D := D \setminus \{a\} \cup \{a—i\to b\}$. Если $b\in V(D)$, то в D есть единственная дуга $a`—i`\to b$, заканчивающаяся в b. Если по дереву D расстояние от x до a, увеличенное на 1, меньше, чем расстояние от x до b, то дуга $a—i\to b$ заменяет в D дугу $a`—i`\to b$: $D := D \cup \{a—i\to b\} \setminus \{a`—i`\to b\}$. Следующее утверждение очевидно, и мы приводим его без доказательства: если такая коррекция применяется последовательно для каждой дуги графа, то в итоге будет получено прямое остовное (для H) дерево кратчайших путей с корнем в x.

5.2. Коррекция обратного дерева

Пусть из всех вершин графа H достижима вершина x, и имеется обратное дерево $D\subseteq H$ с корнем в x. Определим коррекцию дерева, когда становится известным о существовании дуги $a—i\to b\in H\setminus D$ с концом в D, т.е. $b\in V(D)$. Если $a\notin V(D)$, то дуга $a—i\to b$ добавляется в D: $D := D \setminus \{b\} \cup \{a—i\to b\}$. Если $a\in V(D)$, то в D есть единственная дуга $a—i`\to b`$, начинающаяся в a. Если по дереву D расстояние от b до x, увеличенное на 1, меньше, чем расстояние от a до x, то дуга $a—i\to b$ заменяет в D дугу $a—i`\to b`$: $D := D \cup \{a—i\to b\} \setminus \{a—i`\to b`\}$. Следующее утверждение очевидно, и мы приводим его без доказательства: если

такая коррекция применяется последовательно для каждой дуги графа, то в итоге будет получено обратное остовное (для H) дерево кратчайших путей с корнем в x.

6. Способы передачи сообщений

В описываемых ниже алгоритмах используется пять типов передачи сообщений. Мы определим эти способы здесь, указывая параметры сообщения и переменные в вершинах, которые используются в этих способах передачи. В дальнейшем будем просто ссылаться на способ передачи того или иного сообщения, указывая также дополнительные параметры.

6.1. Рассылка из корня

Параметры сообщения: отсутствуют. Это сообщение должно пройти по каждой дуге графа ровно один раз. В каждой вершине x имеется булевская переменная $s(x)$; вначале $s(x) = \textbf{\textit{false}}$. Сообщение создается корнем r и посылается им по каждой исходящей из корня дуге графа. Когда вершина $x \neq r$ получает сообщение, она анализирует переменную $s(x)$. Если $s(x) = \textbf{\textit{false}}$, сообщение пересылается дальше по каждой исходящей из x дуге графа и $s(x) := \textbf{\textit{true}}$. Если $s(x) = \textbf{\textit{true}}$, ничего не делается. Заметим, что условие $s(x) = \textbf{\textit{false}}$ является единственным условием, которое должно быть выполнено в каждой вершине перед началом работы любого алгоритма; можно считать, что это часть начального состояния вершины.

6.2. Множественная рассылка

Параметры сообщения: номер вершины x, создавшей сообщение. Это сообщение распространяется по графу аналогично рассылке из корня, но только создателем сообщения может быть другая вершина, и таких вершин может быть несколько. В каждой вершине x имеется множество $S(x)$ вершин-создателей, от которых получено сообщение; вначале $S(x) = \varnothing$. Когда вершина x создаёт сообщение, $S(x) := S(x) \cup \{x\}$. Когда вершина $x \neq y$ получает сообщение, созданное вершиной y, она анализирует множество $S(x)$. Если $y \notin S(x)$, сообщение пересылается дальше по каждой исходящей из x дуге графа и $S(x) := S(x) \cup \{y\}$. Множество $S(x)$ можно задавать битовой шкалой длины n.

6.3. Пересылка по прямому дереву

Параметры сообщения: описание прямого дерева $F(x)$ с корнем в вершине x, номер вершины $y \in V(F(x))$. Сообщение передается по прямому дереву $F(x)$ от x до y. Когда вершина $z \neq y$ получает сообщение, она пересылает его дальше по исходящей из z дуге дерева $F(x)$, ведущей в y, т.е. первой дуге на пути от x до y в дереве $F(x)$.

6.4. Пересылка по обратному дереву

Параметры сообщения: описание обратного дерева $R(x)$ с корнем в вершине x. Сообщение передается по обратному дереву $R(x)$ до вершины x. Когда вершина $z \neq x$ получает сообщение, она пересылает его дальше по исходящей из z дуге дерева $R(x)$.

6.5. Сбор по обратному дереву

Параметры сообщения: описание обратного дерева $R(x)$ с корнем в вершине x, булевский признак q. В каждой вершине z имеются счетчик $c(z)$ и переменная $m(z)$, в которой может храниться сообщение; вначале $c(z) = 0$. Из каждой вершины $z \in V(R(x)) \setminus \{x\}$ будет послано одно сообщение по исходящей из z дуге дерева $R(x)$, но только после того, как вершина z получит сообщение по каждой входящей в z дуге дерева $R(x)$ и сама создаст сообщение. Для подсчета числа сообщений используется счетчик $c(z)$, который увеличивается на 1, когда вершина z получает сообщение или создает его сама. Полученное или созданное вершиной z сообщение запоминается в переменной $m(z)$, если там уже не запомнено сообщение с $q = \textbf{\textit{true}}$. Когда $c(z)$ на 1 больше числа дуг дерева $R(x)$, входящих в z, запомненное в $m(z)$ сообщение посылается по исходящей из z дуге дерева $R(x)$. В результате вершина x получит ровно одно сообщение по каждой входящей в x дуге дерева $R(x)$. Среди полученных вершиной x сообщений (включая сообщение, созданное вершиной x) будет сообщение с $q = \textbf{\textit{true}}$ тогда и только тогда, когда хотя бы одна вершина из $V(R(x))$ создала сообщение с $q = \textbf{\textit{true}}$. Такой способ передачи сообщений будем называть «сбором по обратному остову».

7. Алгоритмы построения в корне описаний прямого и обратного остовных деревьев кратчайших путей

В этом разделе предложены два алгоритма построения в корне описаний прямого F и обратного R остовных деревьев кратчайших путей, а в каждой вершине x – описания множества $In(x)$ входящих дуг. Оценки: в «быстром» алгоритме $T = \boldsymbol{O}(n)$, $N = \boldsymbol{O}(n)$, в «экономном» алгоритме $T = \boldsymbol{O}(n^2)$, $N = \boldsymbol{O}(1)$, остальные оценки совпадают: $A = \boldsymbol{O}(n\log n)$ и $M = \boldsymbol{O}(n\log n)$.

7.1. «Быстрый» алгоритм

Идея алгоритма заключается в следующем. Сначала рассылкой из корня по графу распространяется сообщение «Старт». Оно пройдет по каждой дуге ровно один раз, что требует времени не более n тактов. Это сообщение накапливает в себе описание пройденного им маршрута Fp. Когда сообщение приходит в вершину x, Fp запоминается в ней. Множество маршрутов, запомненных в x, образует подграф $H(x)$, состоящий из прямого дерева с корнем в r и множества $In(x)$ дуг, ведущих из его листьев в x. Поэтому число дуг в $H(x)$ ограничено по порядку n, что ограничивает размер описания $H(x)$ порядком

nlogn. Получив «Старт» с параметром *Fp*, вершина создает сообщение «Возврат», которое распространяется по графу множественной рассылкой, что гарантирует доставку сообщения в корень. На одной дуге может оказаться $\boldsymbol{O}(n)$ сообщений «Возврат». «Возврат» содержит как параметр *Fp* (из сообщения «Старт»), и тоже накапливает в себе описание пройденного им маршрута *Rp*. Получая «Старт» с *Fp* (это цикл) или «Возврат» с *Fp* и *Rp* (вместе образуют цикл), корень формирует описание текущих прямого *F* и обратного *R* деревьев на множестве вершин $V = V(F) = V(R)$; размер описания тоже ограничен по порядку *nlogn*. Корень в цикле организует опрос вершин в старт-стопном режиме.

Сообщение «Опрос» передается рассылкой по дереву *F*, а корень запоминает число $w := |V|$ вершин, которые получат «Опрос». Получив «Опрос», вершина *x* создает сообщение «Ответ», в котором указывает $H(x)$ и полустепень исхода $d^-(x)$. «Ответ» передается пересылкой по дереву *R*. Поэтому на одной дуге может оказаться $\boldsymbol{O}(n)$ сообщений «Ответ». Корень, получая $H(x)$, образованный маршрутами от корня до $x \in V$, корректирует (см. п.5.1 и п.5.2) описания деревьев *F* и *R* (соответственно меняется *V*) так, чтобы это были деревья кратчайших путей. Корень запоминает $D^+(x) := |In(x)|$ и $D^-(x) := d^-(x)$. Когда корень получит «Ответ» от всех *w* вершин, проверяется условие завершения алгоритма: 1) полустепень исхода $d^-(x)$ запомнена в $D^-(x)$ для каждой $x \in V$, т.е. $D^-(x) > 0$, 2) суммарное число известных входящих дуг равно суммарному числу исходящх дуг $\Sigma_{x\in V} D^+(x) = \Sigma_{x\in V} D^-(x)$.

7.1.1. Начало работы

Корень *r* инициализирует $S(r) := \{r\}$, $F := \{r\}$, $R := \{r\}$, $D^-(r) := d^-(r)$, $D^+(r) := 0$, $In(r) := \varnothing$. Если нет дуг, исходящих из *r*, алгоритм заканчивается. В противном случае создается сообщение «Старт» и посылается по каждой исходящей из *r* дуге *j* с параметрами ($\langle\rangle$, *r*, *j*).

7.1.2. Сообщение «Старт», рассылка из корня

Параметры сообщения при передаче по дуге *a*—*i*→*x*: *a*, *i*, *Fp* – маршрут, пройденный сообщением от *r* до *a*. Обозначим $Fp` := Fp{\cdot}\langle a—i\to x\rangle$. Вершина $x \neq r$, получив «Старт» первый раз ($s(x) = \boldsymbol{false}$), инициализирует $S(x) := \{x\}$, $H(x) := Fp`$, $In(x) := \{a—i\to x\}$, пересылает «Старт» по каждой исходящей из *x* дуге *j* с параметрами ($Fp`$, *x*, *j*), и создает сообщение «Возврат», которое тоже посылает по каждой исходящей из *x* дуге *j* с параметрами ($Fp`$, $\langle\rangle$, *x*, *j*). При повторном ($s(x) = \boldsymbol{true}$) получении «Старта» $H(x) := H(x) \cup Fp`$, $In(x) := In(x) \cup \{a—i\to x\}$, «Старт» дальше не посылается и «Возврат» не создается. Корень *r*, получая «Старт», увеличивает $D^+(r) := D^+(r)+1$, $In(r) := In(r) \cup \{a—i\to r\}$, для каждого $y \in V(\boldsymbol{Im}(Fp))\backslash V$ добавляет $D^+(y) := 0$ и $D^-(y) := 0$. Затем корректируется *F* (см. п.5.1) перебором дуг $Fp`$ от начала к концу, и корректируется *R* (см. п.5.2) перебором дуг $Fp`$ от конца к началу.

После этого корень создает сообщение «Опрос» с параметрами (F, R) и посылает его по исходящим из корня дугам *F*, запоминая $w := |V|$.

7.1.3. Сообщение «Возврат», множественная рассылка

Параметры сообщения при передаче по дуге *a*—*i*→*z*: *a*, *i*, *Fp*, *Rp*, где *Fp* –это маршрут, пройденный сообщением «Старт» от *r* до некоторой вершины *x*, которая создает сообщение «Возврат», *Rp* – это маршрут, пройденный сообщением «Возврат» от вершины *x* до вершины *a*. Обозначим $Rp` := Rp{\cdot}\langle a—i\to z\rangle$. Когда «Возврат» приходит в вершину *z*, проверяется, является ли это сообщение первым или повторным от вершины *x*. Если $x \notin S(z)$, это первое сообщение. Тогда, если $z \neq r$, «Возврат» пересылается дальше по каждой исходящей из *z* дуге *j* с параметрами (Fp, $Rp`$, *z*, *j*). Если $z = r$, корень для каждого $y \in V(\boldsymbol{Im}(Fp) \cup \boldsymbol{Im}(Rp))\backslash V$ добавляет $D^+(y) := 0$ и $D^-(y) := 0$, корректирует *F* (см. п.5.1), перебирая дуги $Fp{\cdot}Rp`$ от начала к концу, и корректирует *R* (см. п.5.2), перебирая дуги $Fp{\cdot}Rp`$ от конца к началу. Повторный ($x \in S(z)$) «Возврат» игнорируется.

7.1.4. Сообщение «Опрос», рассылка по дереву F

Параметры сообщения: *F*, *R*, где *F* прямое, а *R* обратное остовные деревья кратчайших путей с корнем в *r*. Когда сообщение «Опрос» приходит в вершину *x*, оно пересылается дальше по всем исходящим из *x* дугам дерева *F*. Кроме того, вершина *x* создает сообщение «Ответ», которое посылается по исходящей из *x* дуге дерева *R*.

7.1.5. Сообщение «Ответ», пересылка по дереву R

Параметры сообщения при создании в вершине *x*: *R*, $H(x)$, $d^-(x)$. Когда вершина $z \neq r$ получает «Ответ», она пересылает его по исходящей из *z* дуге дерева *R*. Когда корень получает «Ответ», он запоминает $D^+(x) := |In(x)|$, $D^-(x) := d^-(x)$, для каждого $y \in V(H(x))\backslash V$ добавляет $D^+(y) := 0$ и $D^-(y) := 0$. Далее корректируется *F* (см. п.5.1) перебором дуг из $H(x)$ от корня к *x*, и корректируется *R* (см. п.5.2) перебором дуг из $H(x)$ от *x* к корню. Корень дожидается *w* «Ответов» и проверяет условие конца работы.

7.1.6. Конец алгоритма

Конец работы определяет корень при выполнении условия: 1) $\forall x \in V\ D^-(x) > 0$, 2) $\Sigma_{x\in V} D^+(x) = \Sigma_{x\in V} D^-(x)$. Если условие не выполнено, корень снова создает и рассылает сообщение «Опрос», запоминая $w = |V|$.

Покажем, что алгоритм заканчивается через конечное время (в тактах) и определим его оценки. За время $t_1 \le n$ сообщение «Старт» пройдет по каждой дуге, в том числе по каждой входящей дуге корня. После этого за время $t_2 \le n$-1 сообщение «Возврат» от каждой вершины дойдет до корня. Поэтому через время не более $t_1 + t_2$ будет $V = V(G)$. Сообщения «Опрос» проходят дерево *F* за время не более *n*-1, а сообщения «Ответ» проходят дерево *R* за время не более *n*-1. Поэтому цикл «Опрос»-«Ответ» длится не более 2(*n*-1) тактов. Поэтому

через время не более $t_1 + t_2 + 2(n\text{-}1)$ начнется очередной цикл, в котором каждая вершина $x \neq r$ создает «Ответ» с $|In(x)| = d^+(x)$. После получения корнем всех «Ответов» на этот «Опрос» будет выполнено условие конца работы. Следовательно, время работы алгоритма $T \le t_1+t_2+2(n\text{-}1)+2(n\text{-}1) = \boldsymbol{O}(n)$. Из описания алгоритма следует, что $N = \boldsymbol{O}(n)$, $A = \boldsymbol{O}(nlogn)$, $M = \boldsymbol{O}(nlogn)$.

Покажем, что в конце работы алгоритма F и R остовные деревья кратчайших путей. В этот момент времени $V = V(G)$, т.е. деревья F и R остовные. Кроме того, для каждой вершины $x \in V$ в корне хранится $D^-(x) = d^-(x)$. Поэтому условие окончания алгоритма может быть выполнено только в том случае, когда $D^+(x) = d^+(x)$ для каждой вершины $x \in V$. Следовательно, для каждой дуги $y—i\rightarrow x$ корень получил такой «Ответ» от x, что $y—i\rightarrow x \in H(x)$. В этот момент времени $x \in V$, а в $H(x)$ существует путь, начинающийся в корне, последняя дуга которого – это дуга $y—i\rightarrow x$. Это значит, что каждая дуга графа участвует в коррекции деревьев F и R. Следовательно, по утверждениям в 5.1 и 5.2 F и R деревья кратчайших путей.

7.2. «Экономный» алгоритм

Идея этого алгоритма заключается в такой модификации «быстрого» алгоритма из п.7.1, чтобы уменьшить оценку N с $O(n)$ до $O(1)$ за счет увеличения времени работы не более чем в n раз. Причиной оценки $N = O(n)$ являются способы передачи сообщения «Возврат» – множественная рассылка, и сообщения «Ответ» – пересылка по дереву R. Сообщение «Возврат» мы удалим. Теперь у нас не будет времени t_2 и гарантии того, что корень узнает о всех вершинах графа через время $t_1 + t_2 = \boldsymbol{O}(n)$. Корень будет узнавать о вершинах только при получении «Старт» и «Ответ». «Ответ» на данный «Опрос» будет передаваться не пересылкой по дереву R, а сбором по дереву R. При создании «Ответа» вершиной x признак $q := (D^+(x) < |In(x)|)$ показывает, что множество $H(x)$ (и, тем самым, и множество $In(x)$) изменилось по сравнению с тем, которое уже попало в корень.

В результате при каждом опросе в корень придет только один «Ответ» по каждой входящей в корень дуге дерева R, причем признак q в нем будет истинен тогда и только тогда, когда в соответствующем поддереве дерева F хотя бы в одной вершине x изменилось $H(x)$. Заметим, что при ложном признаке q в «Ответе» от x не нужны параметры $H(x)$ и $d^-(x)$, но эта оптимизация не влияет на оценки алгоритма. За все время работы алгоритма от каждой вершины x может придти только один «Ответ» с истинным признаком q и $|In(x)| = d^+(x)$. Поэтому после времени t_1 будет не более n циклов «Опрос»-«Ответ», после чего условие конца работы будет выполнено.

7.2.1. Начало работы

Корень r инициализирует $F := \{r\}$, $R := \{r\}$, $D^-(r) := d^-(r)$, $D^+(r) := 0$, $In(r) := \varnothing$. Затем корень посылает «Старт» по каждой исходящей из r дуге j с параметрами $(\langle\rangle, r, j)$, если такие дуги есть. Если дуг нет, то конец алгоритма.

7.2.2. Сообщение «Старт», рассылка из корня

Параметры сообщения при передаче по дуге $a—i\rightarrow x$: a, i, Fp – маршрут, пройденный сообщением от r до a. Обозначим $Fp` := Fp\cdot\langle a—i\rightarrow x\rangle$. Вершина $x \neq r$, получив «Старт» первый раз ($s(x) = \textbf{\textit{false}}$), инициализирует $H(x) := Fp`$, $In(x) := \{a—i\rightarrow x\}$, пересылает «Старт» по каждой исходящей из x дуге j с параметрами $(Fp`, x, j)$. При повторном ($s(x) = \textbf{\textit{true}}$) получении «Старта» $H(x) := H(x) \cup Fp`$, $In(x) := In(x) \cup \{a—i\rightarrow x\}$, «Старт» дальше не посылается. Корень, получая «Старт», увеличивает $D^+(r) := D^+(r)+1$, $In(r) := In(r) \cup \{a—i\rightarrow r\}$, для каждого $y \in V(\textbf{\textit{Im}}(Fp))\backslash V$ добавляет $D^+(y) := 0$ и $D^-(y) := 0$, корректирует F (см. п.5.1), перебирая дуги $Fp`$ от начала к концу, и корректирует $R$ (см. п.5.2), перебирая дуги $Fp`$ от конца к началу. Корень сбрасывает счетчик $c(r) := 1$, создает сообщение «Опрос» с параметрами (F, R, D^+) и посылает его по исходящим из корня дугам F.

7.2.3. Сообщение «Опрос», рассылка по дереву F

Параметры сообщения: F, R, D^+. Когда сообщение «Опрос» приходит в вершину x, оно пересылается дальше по всем исходящим из x дугам дерева F. Также создается сообщение «Ответ» с $q = (D^+(x) = |In(x)|)$. Если x – листовая вершина R, «Ответ» посылается по исходящей из x дуге дерева R. Иначе «Ответ» запоминается в $m(x)$ и $c(x) := 1$.

7.2.4. Сообщение «Ответ», сбор по дереву R

Параметры сообщения при создании в вершине x: признак q, R, $H(x)$, $d^-(x)$. Когда вершина $z \neq r$ получает «Ответ», $c(z) := c(z)+1$. Если $q = \textbf{\textit{true}}$, вершина z запоминает «Ответ» в $m(z)$ вместо ранее запомненного «Ответа». Далее, если счетчик $c(z)$ на 1 больше числа дуг дерева R, входящих в вершину z, запомненный в $m(z)$ «Ответ» посылается по исходящей из z дуге дерева R. Когда корень получает сообщение, $c(r) := c(r)+1$. Если признак $q = \textbf{\textit{true}}$, корень запоминает $D^+(x) := |In(x)|$, $D^-(x) := d^-(x)$, для каждого $y \in V(H(x))\backslash V$ добавляет $D^+(y) := 0$ и $D^-(y) := 0$. Далее корректируется F (см. п.5.1) перебором дуг из $H(x)$ от корня к x, и корректируется R (см. п.5.2) перебором дуг из $H(x)$ от x к корню. Если $c(r)$ меньше или равен числа дуг дерева R, входящих в корень, корень продолжает ждать «Ответы», иначе проверяет условие конца работы.

7.2.5. Конец алгоритма

Конец работы определяет корень при выполнении условия: 1) $\forall x \in V\ D^-(x) > 0$, 2) $\Sigma_{x \in V} D^+(x) = \Sigma_{x \in V} D^-(x)$. Если условие не выполнено, корень снова создает и рассылает сообщение «Опрос», сбрасывая счетчик $c(r) := 1$.

Покажем, что алгоритм заканчивается через конечное время и определим его оценки. За время $t_1 \le n$ сообщение «Старт» пройдет по каждой дуге, в том числе по каждой входящей дуге корня. От каждой вершины x сообщение «Ответ» с $q = \textbf{\textit{true}}$ и $|In(x)| = d^+(x)$ может придти в корень не более одного раза. Поэтому таких сообщений «Ответ» может придти в корень не более n. Сообщения «Опрос» проходят дерево F за время не более n-1, а сообщения «Ответ»

проходят дерево R за время не более n-1. Поэтому цикл «Опрос»-«Ответ» длится не более 2(n-1) тактов. Первый цикл «Опрос»-«Ответ», начинающийся после времени t_1, начнется не позже времени t_1+2(n-1), а всего циклов, начинающихся после времени t_1, будет не более n. Следовательно, время работы алгоритма $T \le t_1+2(n-1)+n(2(n-1)) = \boldsymbol{O}(n^2)$. Из описания алгоритма следует, что на одной дуге может находиться либо не более одного сообщения «Старт», либо не более одного сообщения «Опрос» и (если это дуга из $F \cap R$) не более одного сообщения «Ответ». Поэтому $N \le 2 = \boldsymbol{O}(1)$. Также из описания алгоритма следует, что $A = \boldsymbol{O}(n\log n)$, $M = \boldsymbol{O}(n\log n)$.

Покажем, что в конце работы алгоритма F и R остовные деревья кратчайших путей. По условию конца работы для каждой вершины $x \in V$ будет $D^-(x) = d^-(x)$, и $\Sigma_{x\in V} D^+(x) = \Sigma_{x\in V} D^-(x)$. Из описания алгоритма следует, что для каждой дуги $y \longrightarrow i \rightarrow x$, учтенной в $D^+(x)$, $y \in V$. Поэтому условие конца работы может быть выполнено только в том случае, когда $D^+(x) = d^+(x)$ и $V = V(G)$. Тем самым, в конце работы алгоритма деревья F и R остовные. Кроме того, для каждой дуги $y \longrightarrow i \rightarrow x$ корень получит такой «Ответ» от x, что $q =$ ***true*** и $y \longrightarrow i \rightarrow x \in H(x)$. В этот момент времени $x \in V$, а в $H(x)$ существует путь, начинающийся в корне, последняя дуга которого – это дуга $y \longrightarrow i \rightarrow x$. Это значит, что каждая дуга графа участвует в коррекции деревьев F и R. Следовательно, по утверждениям в 5.1 и 5.2 F и R деревья кратчайших путей.

8. Алгоритмы построения отображения

Мы рассмотрим два алгоритма построения отображения f, которое для каждой вершины a и каждой другой вершины b ставит в соответствие номер дуги, с которой начинается кратчайший путь из a в b. У этих алгоритмов одинаковые оценки $A = \boldsymbol{O}(n\log n)$ и $M = \boldsymbol{O}(n\log n)$, но разные оценки T и N: в «быстром» алгоритме оценки $T = \boldsymbol{O}(n)$ и $N = \boldsymbol{O}(n)$, в «экономном» алгоритме оценки $T = \boldsymbol{O}(n^2)$ и $N = \boldsymbol{O}(1)$.

В каждом из этих алгоритмов размер сообщения $M = \boldsymbol{O}(n\log n)$. Легко модифицировать алгоритмы, чтобы уменьшить оценку M в n раз до $\boldsymbol{O}(\log n)$, увеличив в n раз оценку N. Для этого достаточно при передаче сообщения размером $\boldsymbol{O}(n\log n)$ разбивать его на серию из n мини-сообщений размером $\boldsymbol{O}(\log n)$ с пометкой последнего мини-сообщения в серии. Эти мини-сообщения посылаются по дуге в одном срабатывании автомата подряд от первого до последнего. Вершина принимает мини-сообщения и сохраняет их в своей памяти, пока не получит последнее в серии мини-сообщение. Аналогично можно уменьшить оценку M до $\boldsymbol{O}(1)$, увеличив в $n\log n$ раз оценку N.

Идея алгоритмов заключается в следующем. Сначала применяется один из двух алгоритмов построения в корне описаний прямого F и обратного R деревьев кратчайших путей, а в каждой вершине x – описания $In(x)$ множества входящих дуг. Затем корень организует доставку в каждую вершину x описания деревьев F и R, а из каждой вершины y в каждую вершину x описания множества

входящих дуг $In(y)$. По F и R вершина x строит описание прямого $F(x)$ и обратного $R(x)$ остовов с корнем в x. Получая от вершины y множество $In(y)$, вершина x для каждой дуги из $In(y)$ корректирует деревья $F(x)$ и $R(x)$. В конечном итоге эти деревья станут деревьями кратчайших путей с корнем в x, что позволяет вершине x создать требуемое отображение f_x.

8.1. «Быстрый» алгоритм

Применяется «быстрый» алгоритм построения в корне описаний деревьев F и R (см. п.7.1). Доставка из корня в каждую вершину описания деревьев F и R, а из каждой вершины в каждую вершину описания множества входящих дуг выполняются параллельно.

8.1.1. Начало работы

После выполнения «быстрого» алгоритма построения в корне описаний деревьев корень r создает сообщение «Остовы» с параметрами (F, R, $F(r) = F$, $In(r)$), которое посылает по каждой исходящей из корня дуге дерева F. Также корень инициализирует счетчик $ce := 0$, показывающий число сообщений «Конец», полученных корнем. Предполагается, что в каждой вершине $x \ne r$ имеется счетчик $ct(x)$ числа сообщений «Остовы», полученных вершиной x. Вначале $ct(x) = 0$.

8.1.2. Сообщение «Остовы», рассылка по дереву F(x)

Параметры сообщения, созданного вершиной x: F, R, $F(x)$, $In(x)$. Вершина z, получив «Остовы», пересылает его дальше по каждой исходящей из z дуге дерева $F(x)$, и увеличивает счетчик $ct(z) := ct(z)+1$. Далее проверяется, первое ли это сообщение «Остовы», полученное вершиной z. Если $ct(z) = 1$, вершина z строит по F и R прямое остовное дерево $F(z)$ с корнем в z, корректирует его по каждой дуге из $In(z) \cup In(x)$, создает сообщение «Остовы» с параметрами (F, R, $F(z)$, $In(z)$), и посылает его по каждой исходящей из z дуге дерева $F(z)$. Если $ct(z) > 1$, вершина z только корректирует $F(z)$ по каждой дуге из $In(x)$. В любом случае вершина z проверяет, получила ли она «Остовы» от всех вершин $x \ne z$. Если $ct(z) = |V|-1$, вершина z по дереву $F(z)$ строит требуемое отображение f_z, создает сообщение «Конец» с параметрами (R, z), которое посылает по исходящей из z дуге дерева R.

8.1.3. Сообщение «Конец», пересылка по дереву R

Параметры сообщения, созданного вершиной x: R, x. Вершина $z \ne r$, получив «Конец», пересылает его дальше по исходящей из z дуге дерева R. Корень, получая сообщение «Конец», увеличивает счетчик $ce := ce+1$. Далее проверяется условие конца работы.

8.1.4. Конец алгоритма

Конец работы алгоритма проверяется корнем по условию: $ce = |V|-1$.

Покажем, что алгоритм заканчивается через конечное время и определим его оценки. «Быстрый» алгоритм построения в корне описаний деревьев имеет оценки $T = \boldsymbol{O}(n)$, $A = \boldsymbol{O}(nlogn)$, $M = \boldsymbol{O}(nlogn)$, $N = \boldsymbol{O}(n)$. После этого сообщение «Остовы» рассылкой по прямому дереву F за время $t_1 \le n$-1 дойдет от корня до каждой вершины. Следовательно, каждая вершина x получит первое сообщение «Остовы» за время не более t_1, и создаст свое сообщение «Остовы». После этого от каждой вершины x до каждой другой вершины сообщение «Остовы» рассылкой по прямому дереву $F(x)$ дойдет за время не более $t_2 \le n$-1. После этого от каждой вершины до корня сообщение «Конец» пересылкой по обратному дереву R дойдет за время не более $t_3 \le n$-1. Тем самым, алгоритм закончится через время $T = \boldsymbol{O}(n)$. Из описания алгоритма следует, что $A = \boldsymbol{O}(nlogn)$, $M = \boldsymbol{O}(nlogn)$. Каждая вершина создает только одно сообщение «Остовы» и только одно сообщение «Конец». Поэтому $N = \boldsymbol{O}(n)$.

Покажем, что в конце работы алгоритма в каждой вершине z будет создано требуемое отображение f_z. Для этого достаточно показать, что прямое дерево $F(z)$ является деревом кратчайших путей. При получении первого сообщения «Остовы», созданного вершиной x, вершина z получает прямой F и обратный R остовы. По ним она строит прямой остов $F(z)$ с корнем в z, который корректируется по дугам из $In(z) \cup In(x)$. Далее до конца работы вершина z получит «Остовы» от каждой вершины $y \ne z$ и $y \ne x$ с параметром $In(y)$. Тем самым, вершина z получит описание всех дуг графа. Поскольку $F(z)$ остов, его коррекция по любой дуге графа всегда выполнима. Следовательно, в конце работы $F(z)$ является деревом кратчайших путей.

8.2. «Экономный» алгоритм

Применяется «экономный» алгоритм построения в корне описания деревьев F и R (см. п.7.2). Корень организует цикл по всем вершинам. Для каждой вершины x выполняется доставка из корня в x описания деревьев F и R, а из x в каждую вершину описания множества входящих дуг $In(x)$.

8.2.1. Начало работы

После выполнения «экономного» алгоритма построения в корне описания деревьев корень проверяет $|V| = 1$. Если корень единственная вершина, конец алгоритма. Иначе корень r выбирает любую вершину $x \ne r$, инициализирует множество вершин $SV := \{r, x\}$, создает сообщение «Остовы» с параметрами $(x, F, R, In(r))$, которое посылает по исходящей из корня дуге дерева F, ведущей в x. Также корень инициализирует счетчик $ce := 0$, показывающий число сообщений «Конец», полученных корнем. Предполагается, что в каждой вершине $x \ne r$ имеется счетчик $ci(x)$ числа сообщений «Дуги», полученных вершиной x, учитывающий также одно сообщение «Остовы» от корня, если вершина x его получала. Вначале $ci(x) = 0$.

8.2.2. Сообщение «Остовы», пересылка по дереву F

Параметры сообщения: x, F, R, $In(r)$. Вершина $z \ne x$, получив «Остовы», пересылает его дальше по исходящей из z дуге дерева F, ведущей в x. Когда сообщение «Остовы» получит вершина x, она увеличивает счетчик $ci(x) := ci(x)+1$, после чего проверяет его. Если $ci(x) = 1$, вершина x строит по F и R прямое остовное дерево $F(x)$ с корнем в x, и корректирует его по каждой дуге из $In(x) \cup In(r)$. Если $ci(x) > 1$, вершина x только корректирует $F(x)$ по каждой дуге из $In(r)$. В любом случае, если $ci(x) = |V|$-1, вершина x по дереву $F(x)$ строит требуемое отображение f_x. В любом случае вершина x создает сообщение «Дуги» с параметрами $(F(x), R, In(x))$, и посылает его по каждой исходящей из x дуге дерева $F(x)$.

8.2.3. Сообщение «Дуги», рассылка по прямому дереву F(x)

Параметры сообщения, созданного вершиной x: $F(x)$, R, $In(x)$. Вершина z, получив «Дуги», пересылает его дальше по каждой исходящей из z дуге дерева $F(x)$, увеличивает счетчик $ci(z) := ci(z)+1$, после чего проверяет его. Если $ci(z) = 1$, вершина z строит по F и R прямое остовное дерево $F(z)$ с корнем в z, и корректирует его по каждой дуге из $In(z) \cup In(x)$. Если $ci(z) > 1$, вершина z только корректирует $F(z)$ по каждой дуге из $In(x)$. В любом случае, если $ci(z) = |V|$-1, вершина z по дереву $F(z)$ строит требуемое отображение f_z. В любом случае, вершина z создает сообщение «Конец» с параметрами $(R, q = \textbf{\textit{true}})$, которое передается сбором по обратному дереву R: если z листовая вершина дерева R, сообщение посылается по исходящей из z дуге дерева R, иначе сообщение не посылается до тех пор, пока вершина z не получит сообщения «Конец» по всем входящим в z дугам дерева R.

8.2.4. Сообщение «Конец», сбор по обратному дереву R

Параметры сообщения: $R, q = \textbf{\textit{true}}$. Вершина $z \ne r$, получив «Конец», пересылает его дальше сбором по обратному дереву R: если z листовая вершина дерева R, сообщение посылается по исходящей из z дуге дерева R, иначе сообщение не посылается до тех пор, пока вершина z не получит сообщения «Конец» по всем входящим в z дугам дерева R. Корень, получая сообщение «Конец», увеличивает счетчик $ce := ce+1$. Если $ce = |In(r)|$ – число входящих в корень дуг дерева R, проверяется условие конца работы.

8.2.5. Конец алгоритма

Конец работы алгоритма проверяется корнем по условию: $SV = V$. Если условие не выполнено, корень выбирает любую вершину $x \in V \setminus SV$, добавляет ее во множество вершин $SV := SV \cup \{x\}$, сбрасывает счетчик $ce := 0$, создает сообщение «Остовы» с параметрами $(x, F, R, In(r))$, и посылает его по исходящей из корня дуге дерева F, ведущей в x.

Покажем, что алгоритм заканчивается через конечное время и определим его оценки. «Экономный» алгоритм построения в корне описаний деревьев имеет оценки $T = \boldsymbol{O}(n^2)$, $A = \boldsymbol{O}(nlogn)$, $M = \boldsymbol{O}(nlogn)$, $N = \boldsymbol{O}(1)$. После этого корень

перебирает все вершины $x \neq r$ в цикле, число проходов цикла равно n-1. На каждом проходе цикла сообщение «Остовы» пересылкой по прямому дереву F за время $t_1 \leq n$-1 дойдет от корня до выбранной вершины x. После этого от вершины x до каждой другой вершины сообщение «Дуги» рассылкой по прямому дереву $F(x)$ дойдет за время не более $t_2 \leq n$-1. После этого от каждой вершины до корня сообщение «Конец» сбором по обратному дереву R дойдет за время не более $t_3 \leq n$-1. Тем самым, один проход цикла требует времени $t_1 + t_2 + t_3 \leq 3(n\text{-}1)$, и алгоритм закончится через время $T = \boldsymbol{O}(n^2)$. Из описания алгоритма следует, что $A = \boldsymbol{O}(nlogn)$, $M = \boldsymbol{O}(nlogn)$, $N = \boldsymbol{O}(1)$.

Покажем, что в конце работы алгоритма в каждой вершине z будет создано требуемое отображение f_z. Для этого достаточно показать, что прямое дерево $F(z)$ является деревом кратчайших путей. При получении вершиной z первого сообщения «Остовы» от корня или «Дуги» от вершины x вершина z получает прямой F и обратный R остовы. По ним она строит прямой остов $F(z)$ с корнем в z, который корректируется по дугам из $In(z) \cup In(r)$ или $In(z) \cup In(x)$. Далее до конца работы вершина z получит «Дуги» от каждой вершины $y \neq z$ и $y \neq r$ с параметром $In(y)$. Тем самым, вершина z получит описание всех дуг графа. Поскольку $F(z)$ остов, его коррекция по любой дуге графа всегда выполнима. Следовательно, в конце работы $F(z)$ является деревом кратчайших путей.

9. Нижние оценки сложности

Для «быстрого» алгоритма построения отображения оценка $T = \boldsymbol{O}(n)$ не может быть улучшена, поскольку, очевидно, ограничена снизу временем распространения сообщения от одной вершины до другой, т.е. диаметром графа, который может достигать величины n-1. В этом разделе мы докажем, что для «экономного» алгоритма построения отображения оценка $T = \boldsymbol{O}(n^2)$ также не может быть улучшена: для алгоритмов построения отображения с оценками $A = unlimited$, $M = unlimited$, $N = \boldsymbol{O}(1)$ имеет место $T = \Omega(n^2)$.

Нам достаточно доказать, что алгоритм, который строит требуемое отображение f_r только в корне r с оценкой $N = \boldsymbol{O}(1)$, имеет оценку $T = \Omega(n^2)$. Произвольный алгоритм будем называть *корневым*, если на любом графе в конце его работы в памяти корня r построено отображение f_r. Оценка $N = \boldsymbol{O}(1)$ означает, что существует такая константа k, что $N \leq k$; ее будем считать емкостью дуги.

До сих пор мы говорили о том, что сообщение проходит маршрут «без задержек»: когда вершина принимает сообщение по дуге маршрута, она сразу же посылает сообщение по следующей дуге маршрута, т.е. не дожидаясь каких-либо еще сообщений. Теперь такой маршрут будем называть *главным маршрутом* сообщения, но нам понадобится более общее понятие маршрута, проходимого сообщением, возможно, с какими-то задержками. Для любого алгоритма будем говорить, что в момент времени t *сообщение прошло маршрут* как последовательность смежных дуг $P = x_1 \to x_2 \to ... \to x_{p\text{-}1} \to x_p$, если существует

неубывающая последовательность моментов времени $t_1 \leq t`_2 \leq t_2 \leq ... \leq t`_{p\text{-}1} \leq t_p \leq t`_p = t$ такая, что для $i = 1..p$-1 вершина $x_i$ послала сообщение по дуге $x_i \to x_{i+1}$ в момент времени $t_i$, а вершина $x_{i+1}$ получила это сообщение в момент времени $t`_{i+1}$. Из этого определения следует, что в общем случае сообщение в момент времени t может пройти несколько маршрутов, заканчивающихся в разных вершинах, и в одной вершине могут заканчиваться несколько из этих маршрутов. Это объясняется тем, что сообщения могут «размножаться», когда вершина, получив сообщение, посылает сообщения по нескольким исходящим дугам, и «склеиваться», когда вершина не посылает сообщение, пока не получит несколько сообщений. Будем говорить, что *вершина получает сообщение от вершины* x, если она получает сообщение, прошедшее маршрут, начинающийся в x. Будем говорить, что алгоритм *покрывает дуги графа*, если для каждой дуги графа при работе алгоритма есть сообщение, которое проходит маршрут, проходящий по этой дуге и заканчивающийся в корне.

Утверждение 1. Если алгоритм корневой, то он покрывает дуги графа.

Доказательство: Допустим противное. Пусть D корневой алгоритм, а G – граф, в котором есть дуга $a—j\to$, и при работе алгоритма D нет сообщения, которое проходит маршрут, проходящий по этой дуге и заканчивающийся в корне. До конца работы алгоритма D корень получал сообщения, маршруты которых – это маршруты, заканчивающиеся в корне. Поэтому корень может определить конец работы алгоритма D, только на основании этих маршрутов, включая номера дуг и номера вершин, через которые проходили эти маршруты, а также полустепени исхода этих вершин. Рассмотрим граф $G`$, который отличается от графа $G$ тем, что добавлена новая вершина $b$ и новая дуга $b—1\to a$, а дуга $a—j\to$ перенаправлена в вершину $b$, т.е. это дуга $a—j\to b$. Тогда должно быть: $b \notin \boldsymbol{Im}(f_r(G))$ и $b \in \boldsymbol{Im}(f_r(G`))$. На обоих графах в алгоритме D корень r получит одинаковые маршруты и, следовательно, будут построены одинаковые отображения в корне, чего быть не может, если алгоритм D корневой. Утверждение доказано.

В алгоритме, не нарушающем емкость дуги k, вершина может посылать сообщение по дуге только в том случае, когда на этой дуге гарантированно меньше k сообщений. При произвольном времени прохождения сообщений по дугам такую гарантию может дать только получение сообщения-подтверждения, прошедшего по этой дуге и далее по циклу вернувшегося в начало дуги. Если вершина послала по дуге несколько сообщений, подтверждать нужно каждую из таких пересылок. Проблема в том, что сообщения могут «размножаться» и «склеиваться».

Опишем формальную модель подтверждения пересылок. Для этого модифицируем алгоритм так, чтобы вершина нумеровала пересылки сообщений по исходящим дугам и учитывала возникающие циклы передачи сообщений. Пересылка задается тройкой (v,j,h), где v – номер вершины, j –

номер исходящей из нее дуги, h – номер пересылки из этой вершины. В каждое сообщение m добавляется множество $A(m)$ всех пересылок этого сообщения. В каждую вершину с номером v добавляется максимальный номер $h_{max}(v)$ пересылки из этой вершины, множество $B(v)$ всех неподтвержденных пересылок из этой вершины, а также множество $A(v)$ пересылок из сообщений, принятых этой вершиной после отправки ею последнего сообщения. Вначале $h_{max}(v) = 0$, $B(v) = \varnothing$, $A(v) = \varnothing$. Можно считать, что в корне эти переменные инициализируются в начале работы, а в любой другой вершине при получении первого сообщения. Когда вершина с номером v посылает сообщение m по дуге с номером j, она увеличивает номер неподтвержденной пересылки: $h_{max}(v) := h_{max}(v)+1$, запоминает эту пересылку как неподтвержденную $B(v) := B(v) \cup \{ (v, j, h_{max}(v)) \}$, добавляет ее в сообщение $A(m) := A(v) \cup \{ (v, j, h_{max}(v)) \}$, затем посылает сообщение m по дуге j и «забывает» пересылки из принятых сообщений $A(v) := \varnothing$. Когда вершина с номером v получает сообщение m, она подтверждает неподтвержденные пересылки, имеющиеся в сообщении, $B(v) := B(v) \setminus A(m)$ и запоминает пересылки из сообщения $A(v) := A(v) \cup A(m)$. Будем говорить, что сообщение m является *подтверждением* в вершине с номером v для исходящей дуги с номером j, если при получении этого сообщения множество $\{ (v,j,h) \in B(v) \mid h = 1.. \}$ неподтвержденных пересылок по этой дуге уменьшается. Алгоритм не нарушает емкость k дуги с номером j, исходящей из вершины с номером v, тогда и только тогда, когда вершина посылает сообщение по этой дуге только при условии, что число неподтвержденных пересылок по этой дуге меньше емкости дуги: $|\{ (v,j,h) \in B(v) \mid h = 1.. \}| < k$. Будем говорить, что *алгоритм следует правилу подтверждения для емкости дуги* k, если вершина, послав по дуге p сообщений и получив q подтверждений, посылает следующее сообщение по этой дуге только при условии $p\text{-}q < k$. Отсюда непосредственно следует следующее утверждение.

Утверждение 2. При произвольном времени прохождения сообщений по дугам алгоритм не нарушает емкость дуги k тогда и только тогда, когда алгоритм следует правилу подтверждения для емкости дуги k.

Рассмотрим класс графов $\boldsymbol{G} = \{ G_n \mid n \ge 2 \}$ на рис. 2 со всеми возможными способами нумерации вершин и дуг. Числа, записанные в вершинах на рис. 2 будем называть *индексами*. Когда мы будем говорить «вершина i», мы будем иметь в виду «вершина с индексом i». Если $n\text{-}1 \ge j > i \ge 1$, то будем говорить, что вершина i (с меньшим индексом) расположена правее вершины j, а вершина j (с большим индексом) – левее вершины i. Когда вершина $i = 1..n\text{-}2$ получает сообщение по дуге $i+1 \to i$, будем говорить, что она получает сообщение слева. Обозначим: $v(i)$ – номер вершины i, $a(i)$ – номер дуги $0 \to i$.

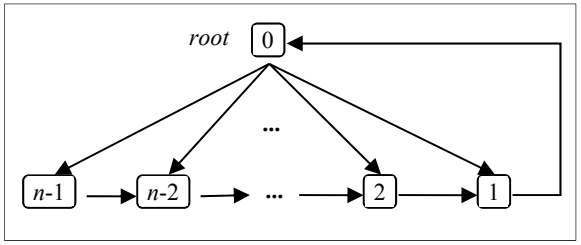

Рис. 2. Пример графа
Fig. 2. Example graph

Из утверждения 1 следует, что нам достаточно доказать, что алгоритм, не нарушающий емкости дуги k и покрывающий все дуги графа на классе $\boldsymbol{G}$ с емкостью дуг k, имеет оценку $T = \Omega(n^2)$. Из утверждения 2 следует, что нам достаточно рассматривать алгоритмы, которые следуют правилу подтверждения для емкости дуги k. Для оценки времени работы алгоритма будем рассматривать только случай, когда время пересылки единичное: каждое сообщение по каждой дуге пересылается за 1 такт. Итак, нам достаточно доказать, что алгоритм, следующий правилу подтверждения для емкости дуги k и покрывающий все дуги графа на классе $\boldsymbol{G}$ с емкостью дуг k и единичным временем пересылки, имеет оценку $T = \Omega(n^2)$. Далее по умолчанию мы будем рассматривать только такие алгоритмы. Через $t(n)$ будем обозначать время работы алгоритма.

Из устройства графа G_n следует, что подтверждением в вершине $i \ne 0$ является сообщение, которое прошло цикл, т.е. маршрут $i \to i\text{-}1 \to ... \to 1 \to 0 \to [... \to 0 \to] i$, где в квадратные скобки заключена часть маршрута, которая может отсутствовать. Сообщение будем называть $\{i\}$-сообщением, если $i \ne 0$ – это самая левая из вершин, через которые проходило сообщение. Очевидно, что вершина i получает $\{i\}$-сообщение по дуге $0 \to i$, т.е. от корня.

Будем говорить, что в вершине $i \ne 0$ происходит *склейка* $\{j`\}$-сообщения с $\{j\}$-сообщением, где $j` > j = i$, если первое сообщение, которое посылает вершина i, это $\{j`\}$-сообщение. Это происходит либо из-за того, что ранее полученное вершиной $\{i\}$-сообщение ожидает в вершине $\{j`\}$-сообщения, либо из-за того, что ранее полученное $\{j`\}$-сообщение ожидает в вершине $\{i\}$-сообщения, либо из-за того, что вершина, не получив $\{i\}$-сообщения, получает $\{j`\}$-сообщение и посылает его дальше. Во всех случаях будет послано $\{j`\}$-сообщение, с которым «склеено» $\{i\}$-сообщение.

Будем говорить, что в вершине $i \ne 0$ происходит *склейка* $\{j`\}$-сообщения с $\{j\}$-сообщением, где $j` > j > i$, если эта вершина посылает $\{j`\}$-сообщение тогда, когда она еще не послала $\{j\}$-сообщение. Это происходит из-за того, что ранее полученное $\{j\}$-сообщение ожидает в вершине $i$ $\{j`\}$-сообщения. Будет послано $\{j`\}$-сообщение, с которым «склеено» $\{j\}$-сообщение.

Утверждение 3. Для любого алгоритма можно так подобрать нумерацию дуг и вершин графа G_n, чтобы ни в какой вершине $i \neq 0$ не было склейки сообщений до получения этой вершиной первого подтверждения.

Доказательство: Рассмотрим работу алгоритма на графе G_n при некоторой нумерации вершин и дуг. Выберем самую левую вершину, в которой происходит склейка до получения этой вершиной первого подтверждения, и самую первую по времени склейку. Пусть это будет вершина $i \neq 0$, и она склеивает $\{j`\}$-сообщение с $\{j\}$-сообщением, где $j` > j$, в момент времени t.

Заметим, что любое сообщение, получаемое вершиной i слева до момента склейки в вершине i, не может быть уже склеено в некоторой вершине $x > i$. Действительно, поскольку мы выбрали самую левую вершину i, где происходит склейка до получения этой вершиной первого подтверждения, склейка в вершине x должна была происходить после получения вершиной x первого подтверждения. Но в этом случае склеенное сообщение, посланное вершиной x и дошедшее до вершины i, является подтверждением и для вершины i, а этого не может быть по правилу выбора вершины i.

Вершина i не может получить слева больше k сообщений до получения вершиной i+1 первого подтверждения. Но если вершина i+1 получает такое подтверждение, а потом посылает сообщение, то это сообщение будет подтверждением и для вершины i. Тем самым, вершина i не может получить слева больше k сообщений до получения вершиной i первого подтверждения.

Если бы было $j` > j$+1, то для некоторого $x$ такого, что $j` > x > j$, либо $\{j`\}$-сообщение склеивалось бы с $\{x\}$-сообщением в вершине левее $i$, либо $\{x\}$-сообщение склеивалось бы с $\{j\}$-сообщением в вершине $i$ до получения ею $\{j`\}$-сообщения. Поэтому, поскольку каждое сообщение, которое вершина i получает слева, не склеено ни с каким сообщением, и мы выбрали самую первую по времени склейку в вершине i, должно быть $j` = j$+1.

Итак, мы рассматриваем склейку $\{j$+1$\}$-сообщения с $\{j\}$-сообщением в вершине i. А поскольку мы выбрали первую по времени склейку, очевидно, j минимально. Поэтому возможны три типа склейки.

1) $j = i$, вершина i, еще не получив $\{i\}$-сообщение, получает $\{i$+1$\}$-сообщение.
2) $j = i$, вершина i сначала получает $\{i\}$-сообщение, но не посылает его дальше, а потом получает $\{i$+1$\}$-сообщение.
3) $j > i$, вершина i сначала посылает последовательно $\{x\}$-сообщения, где $x = i..j$-1, потом получает $\{j\}$-сообщение, но не посылает его дальше, а потом получает $\{j$+1$\}$-сообщение.

Тип 1 (рис. 3). Вершина i, еще не получив $\{i\}$-сообщение, получает $\{i$+1$\}$-сообщение. Тогда вершина i+1 получала $\{i$+1$\}$-сообщение (от корня) и посылала его дальше. Поскольку время пересылки единичное, корень посылал сообщение по дуге 0—$a(i$+1$)\rightarrow i$+1, но не посылал сообщение по дуге 0—$a(i)\rightarrow i$ или послал его позже (иначе оно пришло бы в вершину i раньше).

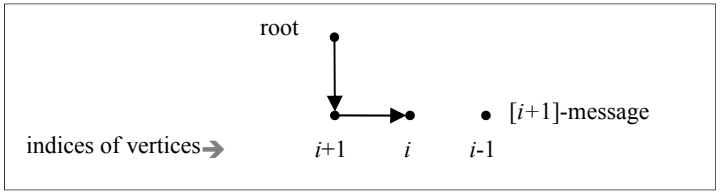

Рис. 3. Склейка сообщений типа 1
Fig. 3. Gluing messages of type 1

Рассмотрим момент времени $t_{0,i+1}$, когда корень посылает сообщение по дуге 0—$a(i$+1$)\rightarrow i$+1. К этому моменту времени корень мог получить сообщения от вершин самого правого отрезка в диапазоне индексов $[1..s_{i+1}]$ (если $s_{i+1} = 0$, диапазон пустой). Должно быть $s_{i+1} < i$, поскольку в противном случае $\{i$+1$\}$-сообщение будет подтверждением в вершине i, т.е. к моменту склейки вершина i получит подтверждение. Поэтому, учитывая, что на классе $\boldsymbol{G}$ у всех вершин, кроме корня, полустепень исхода равна 1, можно считать, что корню в момент времени $t_{0,i+1}$ известная следующая информация:

$I_{0,i+1}=(v(0), d(0), v(1), a(1), ..., v(s_{i+1}), a(s_{i+1}))$.

Поскольку $s_{i+1} < i$, если мы поменяем местами номера дуг $a(i)\leftrightarrow a(i$+1$)$, то информация $I_{0,i+1}$ не изменится, но сообщение от корня будет приходить не в вершину i+1, а в вершину i. Поэтому вершина i сначала получит $\{i\}$-сообщение, а не сообщение слева. После такого изменения нумерации вершин и дуг в вершине i не будет склейки типа 1.

Тип 2 (рис. 4). Вершина i сначала получает $\{i\}$-сообщение, но не посылает его дальше, а потом получает $\{i$+1$\}$-сообщение. Тогда вершина i+1 получала $\{i$+1$\}$-сообщение и посылала его дальше.

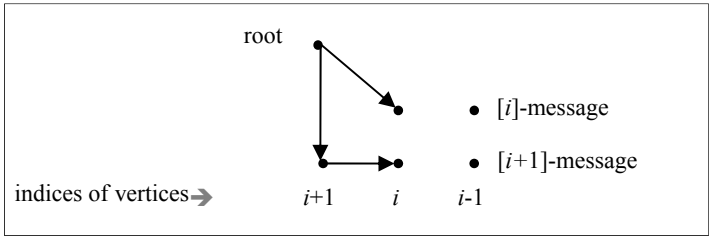

Рис. 4. Склейка сообщений типа 2
Fig. 4. Gluing messages of type 2

Рассмотрим моменты времени $t_{0,i+1}$ ($t_{0,i}$), когда корень посылает сообщение по дуге 0—$a(i$+1$)\rightarrow i$+1 (0—$a(i)\rightarrow i$). К моменту времени $t_{0,i+1}$ ($t_{0,i}$) корень мог получить сообщения от вершин самого правого отрезка в диапазоне индексов $[1..s_{i+1}]$ ($[1..s_i]$). Должно быть $s_{i+1} < i$ и $s_i < i$, поскольку в противном случае $\{i$+1$\}$-сообщение или $\{i\}$-сообщение будет подтверждением в вершине i, т.е. к

моменту склейки вершина i получит подтверждение. В моменты времени $t_{0,i+1}$ и $t_{0,i}$ корню известна следующая информация:

в момент времени $t_{0,i+1}$: $I_{0,i+1}$=($v(0), d^-(0), v(1), a(1), ..., v(s_{i+1}), a(s_{i+1})$),

в момент времени $t_{0,i}$: $I_{0,i}$ =($v(0), d^-(0), v(1), a(1), ..., v(s_i), a(s_i)$).

Вершина i+1 (i) получает {i+1}-сообщение ({i}-сообщение) от корня через один такт в момент времени $t_{0,i+1}$+1 ($t_{0,i}$+1). Поскольку в вершине i+1 не происходит склейки, а в вершине i не происходит склейки типа 1, это будет первое сообщение, получаемое вершиной. Поэтому в эти моменты времени вершинам известна следующая информация:

для вершины i+1: $I_{i+1,i+1}$ =($v(0), d^-(0), v(1), a(1), ..., v(s_{i+1}), a(s_{i+1}), v(i+1), a(i+1)$),

для вершины i: $I_{i,i}$ =($v(0), d^-(0), v(1), a(1), ..., v(s_i), a(s_i), v(i), a(i)$,).

Поскольку $s_{i+1} < i$ и $s_i < i$, если мы поменяем местами номера вершин $v(i) \leftrightarrow v(i+1)$ и номера дуг $a(i) \leftrightarrow a(i+1)$, то информация в корне $I_{0,i+1}$ и $I_{0,i}$ не изменится, но сообщение от корня будет приходить не в вершину i+1, а в вершину i. Поэтому информация для вершин i и i+1 поменяется местами $I_{i+1,i+1} \leftrightarrow I_{i,i}$. Поэтому теперь первое сообщение, которое получит вершина i, будет {i}-сообщением, и оно сразу посылается дальше. После такого изменения нумерации вершин и дуг в вершине i не будет склейки типа 1 и 2.

Тип 3 (рис. 5). Вершина i сначала посылает последовательно {x}-сообщения, где $x = i..j$-1, потом получает {j}-сообщение, но не посылает его дальше, а потом получает {j+1}-сообщение.

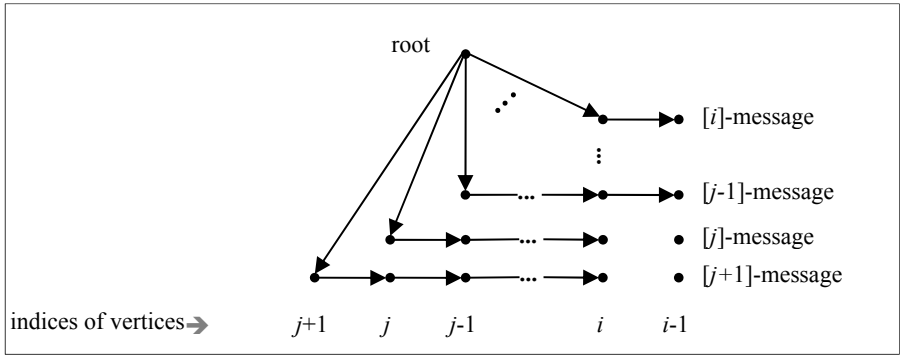

Рис. 5.Склейка сообщений типа 3
Fig. 5. Gluing messages of type 3

Для $y = i..j$+1 вершина y получала и посылала {y}-сообщение. Рассмотрим момент времени $t_{0,y}$, когда корень посылает сообщение по дуге $0—a(y)\rightarrow y$. К этому моменту времени корень мог получить сообщения от вершин самого правого отрезка в диапазоне индексов $[1..s_y]$. Должно быть $s_y < i$, поскольку в противном случае {y}-сообщение будет подтверждением в вершине i, т.е. к моменту склейки вершина i получит подтверждение. В момент времени $t_{0,y}$ корню известна следующая информация:

$I_{0,y}$=($v(0), d^-(0), v(1), a(1), ..., v(s_y), a(s_y)$),

Вершина y получает {y}-сообщение (от корня) через один такт в момент времени $t_{0,y}$+1. Поскольку в вершине $y > i$ не происходит склейки, а в вершине $y = i$ не происходит склейки типа 1 и 2, это будет первое сообщение, получаемое вершиной y. Поэтому в момент времени $t_{0,y}$+1 вершине y известна следующая информация:

$I_{y,y}$=($v(0), d^-(0), v(1), a(1), ..., v(s_y), a(s_y)$, $v(y), a(y)$).

Теперь рассмотрим момент времени $t_{y,j}$, когда вершина $y = i..j$ получает {j}-сообщение. Это сообщение проходит маршрут от вершины j до вершины y, поэтому в момент времени $t_{y,j}$ вершине y известна следующая информация:

$I_{y,j}$=($v(0), d^-(0), v(1), a(1), ..., v(s_y^{max}), a(s_y^{max})$, $v(y), a(y), ..., v(j), a(j)$),

где $s_y^{max} = max\{ s_j, ..., s_y \}$.

Поскольку $s_y < i$, если мы поменяем местами номера вершин по циклу $v(j+1) \rightarrow v(j) \rightarrow ... \rightarrow v(i) \rightarrow v(j+1)$ и номера дуг по циклу $a(j+1) \rightarrow a(j) \rightarrow ... \rightarrow a(i) \rightarrow a(j+1)$, то информация в корне $I_{0,y}$ не изменится для каждого $y = i..j$+1, но сообщение от корня будет приходить не в вершину $y > i$, а в вершину y-1, и не в вершину $y = i$, а в вершину j+1. Поэтому в момент времени $t_{0,y}$+1 в вершине $y = i..j$ окажется та информация, которая была в вершине y+1, т.е. $I_{y+1,y+1}$, а в вершине $y = j$+1 окажется та информация, которая была в вершине i, т.е. $I_{i,i}$. В результате в момент времени $t_{i,j}$ в вершине i окажется та информация, которая была в вершине i+1, т.е. $I_{i+1,j}$. Поэтому теперь вершина i, получая {j}-сообщение, посылает его дальше. После такого изменения нумерации вершин и дуг в вершине i не будет склейки типа 3 для данного j, а также всех меньших его, поскольку j выбиралось минимальным. Также вершина i, по-прежнему будет получать в качестве первого сообщения {i}-сообщение и сразу посылать его дальше (как до изменения делала вершина i+1), поэтому в вершине i не будет склеек типов 1 и 2.

Итак, мы нашли такие изменения нумерации дуг и вершин, которые приводят к отсутствию в вершине i склейки типов 1 и 2, а также к отсутствию склейки типа 3 или к увеличению минимального j для типа 3. Поскольку число вершин конечно, мы можем повторять эти изменения до тех пор, пока при работе данного алгоритма не останется склеек. Утверждение доказано.

Утверждение 4. Время работы алгоритма на графе G_n равно $\Omega(n^2)$.

Доказательство: По утверждению 3 можно считать, что в вершине $i = 1..n$-1 не происходит склеек сообщений до получения этой вершиной первого подтверждения. Очевидно, корень получит {1}-сообщение через $t_1 \geq 2$ такта. Обозначим $x_1 = 1$. Пусть вершина x_j посылает $1 \leq y_j \leq k$ сообщений до того, как она перестает посылать сообщения и начинает ждать первого подтверждения. Обозначим $x_{j+1} = x_j + y_j$, тогда $x_{j+1} \leq x_j + k$. Если $x_{j+1} \leq n$-1, то, поскольку склеек нет, $\{x_{j+1}\}$-сообщение не может быть послано из вершины x_j до получения ею первого подтверждения. Обозначим через t_j время от начала работы, через которое корень получает $\{x_j\}$-сообщение. Только после этого не менее чем

через 1 такт вершина x_j получит подтверждение, и только после этого $\{x_{j+1}\}$-сообщение может пройти путь до корня длиной не менее x_j. Следовательно, $t_{j+1} \ge t_j + 1 + x_j$. Имеем:

$x_1 = 1$, $t_1 \ge 2$,

$x_1+1 \le x_2 \le x_1+k$, $t_2 \ge t_1 + 1 + x_1$,

$x_2+1 \le x_3 \le x_2+k$, $t_3 \ge t_2 + 1 + x_2$,

...

$x_{u-1}+1 \le x_u \le x_{u-1}+k$, $t_u \ge t_{u-1} + 1 + x_{u-1}$.

Тогда $i \le x_i$ и $x_u \le 1+k(u-1)$. Пусть $n \ge k + 1$. Выберем u максимальное так, чтобы оставаться в диапазоне индексов, т.е. $0 \le n-1-k < x_u \le n-1$. Тогда $n-1-k < 1+k(u-1)$, что влечет $u > (n-2)/k$. Очевидно, $t(n) \ge t_u$. Имеем:

$t(n) \ge 2 + (1+x_1) + (1+x_2) + (1+x_3) + ... + (1+x_u) = 2 + u + (x_1+x_2+x_3+...+x_u) \ge$

$\ge 2 + u + (1+2+3+...+u) = 2 + u + u(u+1)/2 = 2+u(u+3)/2 >$

$> 2+((n-2)/k)(((n-2)/k)+3)/2 = (n-2)^2/(2k^2)+3(n-2)/2k+2 \ge n^2/(3k^2) = \Omega(n^2)$.

Утверждение доказано.

10. Синхронная модель

До сих пор мы рассматривали асинхронную модель, в которой время пересылки сообщения по дуге может быть произвольным в диапазоне [0..1] и даже меняться со временем. В синхронной модели время пересылки считается равным 1 такту и не меняется. В то же время следует отметить, что под синхронной моделью понимается не просто модель с единичным временем пересылки, а с дополнительным свойством: вершина принимает сразу все сообщения, дошедшие до нее по входящим дугам. Это соответствует понятию *раунда* как единицы измерения времени. За один раунд каждая вершина принимает все пришедшие к ней по входящим дугам сообщения и посылает все сообщения по исходящим из нее дугам.

Такой режим работы можно симулировать в модели, где вершина за одно «срабатывание» принимает только одно сообщение, если вместе с сообщением вершина получает булевский признак «последнего сообщения», указывающий, является ли это сообщение последним или не последним из сообщений, дошедших до вершины и еще не принятых ею.

«Быстрый» алгоритм построения отображения (см. 8.1) легко модифицируется в синхронной модели так, что получаются оценки $T = O(n)$, $A = O(n^2 logn)$, $M = O(n^2 logn)$, $N = 1$. Для этого достаточно, чтобы вершина принимала сообщения до тех пор, пока не получит сообщение с признаком «последнее сообщение». Если в оригинальном алгоритме, приняв сообщение m, вершина должна была послать по дуге j сообщение $m`$, то в модифицированном алгоритме сообщение не посылается, а в памяти вершины запоминается пара $(m`, j)$. После получения сообщения с признаком «последнее сообщение» по

каждой дуге j посылается не более чем одно сообщение, склеенное из тех сообщений $m`$, которые были запомнены в паре с дугой j.

При единичном времени пересылки все сообщения, посланные в вершину на предыдущем раунде, одновременно доходят до вершины на следующем раунде через 1 такт, время между приемом первого и последнего из этих сообщений равно нулю, и по каждой исходящей дуге посылается не более одного сообщения. Поэтому $N = 1$, и сохраняется оценка $T = O(n)$. Остальные оценки увеличиваются не более чем в n раз: $A = O(n^2 logn)$, $M = O(n^2 logn)$, что объясняется необходимостью хранить в памяти вершины несколько (но не более n) сообщений и склеивать из нескольких (но не более n) сообщений одно сообщение. Более точно, в оригинальном алгоритме на одной дуге может оказаться не более одного сообщения каждого типа от одной вершины, а число типов ограничено. Таким образом, в модифицированном алгоритме в вершине дополнительно хранятся оригинальные сообщения не более чем от n вершин с ограниченным числом сообщений от каждой из них, а «склеенное» сообщение состоит из оригинальных сообщений не более чем от n вершин с ограниченным числом сообщений от каждой из них. Поскольку в оригинальном алгоритме $A = O(nlogn)$ и $M = O(nlogn)$, в модифицированном алгоритме получаются оценки $A = O(n^2 logn)$ и $M = O(n^2 logn)$.

Другим способом симуляции синхронной работы является сигнал «освобождение дуги j», который вершина получает, когда на дуге j не остается сообщений, не дошедших до конца дуги. В этом случае вершина, принимая сообщения, не посылает сообщение по дуге j до тех пор, пока не получит сигнал «освобождение дуги j». Возможно также, что имеется только общий сигнал «освобождение всех исходящих дуг» [7]. Для соответствующей модификации «быстрого» алгоритма построения отображения при единичном времени пересылки получаем такие же оценки $T = O(n)$, $A = O(n^2 logn)$, $M = O(n^2 logn)$, $N = 1$. Более того, эти оценки остаются верными для произвольного и даже меняющегося времени пересылки в диапазоне [0..1]. В этом случае время ожидания между приемом сообщения вершиной и его посылкой по исходящей дуге не превышает времени пересылки по этой дуге, т.е. 1 такта, что ведет к увеличению времени T не более чем в 2 раза.

11. Заключение

В разд. 8 для асинхронной модели распределенной системы предложены два алгоритма построения отображения, позволяющего пересылать сообщения между вершинами по кратчайшим путям. «Быстрый» алгоритм имеет оценки $T = \boldsymbol{O}(n)$, $A = \boldsymbol{O}(nlogn)$, $M = \boldsymbol{O}(nlogn)$, $N = \boldsymbol{O}(n)$, «экономный» алгоритм имеет оценки $T = \boldsymbol{O}(n^2)$, $A = \boldsymbol{O}(nlogn)$, $M = \boldsymbol{O}(nlogn)$, $N = \boldsymbol{O}(1)$. В разд. 9 доказано, что при таких оценках числа N одновременно передаваемых по дуге сообщений оценки времени T не улучшаемы при неограниченных размерах памяти вершин и сообщений: $T = \boldsymbol{\Theta}(n)$ при $N = \boldsymbol{O}(n)$ и $T = \boldsymbol{\Theta}(n^2)$ при $N = \boldsymbol{O}(1)$.

В разд. 10 показано, как «быстрый» алгоритм можно модифицировать для синхронной модели при условии, что имеется либо признак «последнее сообщение», либо сигнал «освобождение дуги j» или «освобождение всех исходящих дуг». В этом случае $T = \boldsymbol{O}(n)$, $A = O(n^2 logn)$, $M = O(n^2 logn)$, $N = 1$.

В то же время, если время пересылки равно 1 такту и не меняется, но признака «последнее сообщение» и сигналов «освобождение дуги j» или «освобождение всех исходящих дуг» нет, вопрос об оценке времени работы алгоритма с ограниченной емкостью дуги $N = O(1)$ остается открытым. Другое направление дальнейших исследований – это поиск эффективных алгоритмов решения задач на ориентированных графах, не сводимых к симуляции алгоритмов для неориентированной распределенной системы.

Список литературы

[1]. Y. Afek and E. Gafni, Distributed Algorithms for Unidirectional Networks, SIAM J. Comput., vol. 23, No. 6, 1994, pp. 1152-1178.

[2]. И.Б.Бурдонов. Обход неизвестного ориентированного графа конечным роботом. Программирование, 2004, №4, стр.11-34.

[3]. И.Б.Бурдонов. Проблема отката по дереву при обходе неизвестного ориентированного графа конечным роботом. Программирование, 2004, №6, стр.6-29.

[4]. И. Бурдонов, А. Косачев. Общий подход к решению задач на графах коллективом автоматов. Труды Института системного программирования РАН, том 29, вып. 2, 2017 г., стр. 27-76. DOI: 10.15514/ISPRAS-2017-29(2)-2.

[5]. И. Бурдонов, А. Косачев. Распределённые алгоритмы на корневых неориентированных графах. Труды Института системного программирования РАН, том 29, вып. 5, 2017 г., стр. 283-310. DOI: 10.15514/ISPRAS-2017-29(5)-14.

[6]. И. Бурдонов, А. Косачев. Размер памяти для хранения упорядоченного корневого графа. Труды Института системного программирования РАН, том 29, вып. 2, 2017 г., стр. 7-26. DOI: 10.15514/ISPRAS-2017-29(2)-1.

[7]. И.Б. Бурдонов, А.С. Косачев, В.В. Кулямин. Параллельные вычисления на графе. Программирование, 2015, №1, стр. 3-20.

Directed distributed system: Backtracking problem

I.B. Burdonov <igor@ispras.ru>
A.S. Kossatchev <kos@ispras.ru>
Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. For a distributed system based on a directed graph without multiple edges and loops, the backtracing problem is considered: how to transfer a message from the final vertex of the arc to its initial vertex. The task is to create a structure on the graph that allows the message to be transmitted from the final vertex of the arc to its initial vertex in the minimum time, i.e. on the shortest path. Such a structure at each vertex a is given by mapping the number of vertex b to the number of the outgoing arc through which the shortest path passes from a to b. In particular, such a mapping makes it possible to simulate, in directed distributed systems,

algorithms for solving problems on a graph, developed for unditected distributed systems. This increases the running time of such algorithms by not more than k times, where k does not exceed the diameter of the graph, $k < n$, where n is the number of vertices of the graph. Section 2 describes the asynchronous model of the distributed system used. Section 3 contains the basic definitions and notation, and Section 4 – the statement of the problem. Section 5 describes two auxiliary algorithms for subtree correction, the application of which makes it possible to construct spanning trees of shortest paths: a out-tree and an in-tree. Section 6 contains a description of the various methods for transmitting messages over the graph. In Section 7, two algorithms are proposed for constructing in the memory of the graph root automaton the descriptions of spanning out- and in- shortest path trees, and in Section 8, the algorithms for constructing the required mapping based on them: a "fast" algorithm with $T = \boldsymbol{O}(n)$ and $N = \boldsymbol{O}(n)$ and an "economical" algorithm with $T = \boldsymbol{O}(n^2)$ and $N = \boldsymbol{O}(1)$, where T is the running time of the algorithm, N is the number of messages simultaneously transmitted along the arc. In Section 9 it is proved that these estimates of time are not improved. In Section 10, the "fast" algorithm is modified for a synchronous model with $N = 1$. The conclusion sums up and outlines directions for further research.

Keywords: directed graph; rooted graph; numbered graph; distributed algorithms; graph problems; backtracing problem; shortest paths.

DOI: 10.15514/ISPRAS-2018-30(2)-9

For citation: Burdonov I.B., Kossatchev A.S. Directed distributed system: Backtracking problem. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue. 2, 2018, pp. 167-194 (in Russian). DOI: 10.15514/ISPRAS-2018-30(2)-9

References

[1]. Y. Afek and E. Gafni, Distributed Algorithms for Unidirectional Networks, SIAM J. Comput., Vol. 23, No. 6, 1994, pp. 1152-1178.

[2]. I.B.Bourdonov. Traversal of an Unknown Directed Graph by a Finite Robot. Programming and Computer Software, vol. 30, No. 4, 2004, pp. 188-203.

[3]. I.B.Bourdonov. Backtracking Problem in the Traversal of an Unknown Directed Graph by a Finite Robot. Programming and Computer Software, vol. 30, No. 4, 2004, pp. 305-322.

[4]. I.Burdonov, A.Kossatchev. General approach to solving problems on graphs by collective automata. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 2, 2017, pp. 27-76 (in Russian). DOI: 10.15514/ISPRAS-2017-29(2)-2.

[5]. I.Burdonov, A.Kossatchev. Distributed algorithms on root undirected graphs. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 5, 2017, pp. 283-310 (in Russian). DOI: 10.15514/ISPRAS-2017-29(5)-14.

[6]. I.Burdonov, A.Kossatchev. The size of the memory for storing the ordered root graph. Trudy ISP RAN/Proc. ISP RAS, vol. 29, issue 2, 2017, pp. 7-26 (in Russian). DOI: 10.15514/ISPRAS-2017-29(2)-1.

[7]. I.B. Burdonov, A.S. Kossatchev, V.V. Kulyamin. Parallel Computations on a Graph. Programming and Computer Software, vol. 41, No. 1, 2015, pp. 1-13. DOI: 10.1134/S0361768815010028.