Вестник РУДН. Серия: Инженерные исследования
RUDN Journal of Engineering Research

2022;23(2):108–116

ISSN 2312-8143 (Print); ISSN 2312-8151 (Online)
journals.rudn.ru/engineering-researches

DOI 10.22363/2312-8143-2022-23-2-108-116
УДК 510.52:519.16

Научная статья / Research article

Необходимые и достаточные условия разделения структур алгоритмов на непересекающиеся множества: полиномиальные и переборные алгоритмы

Н.Л. Малинина

Московский авиационный институт (национальный исследовательский университет), *Москва, Российская Федерация*
✉ malinina806@gmail.com

История статьи
Поступила в редакцию: 28 января 2022 г.
Доработана: 11 марта 2022 г.
Принята к публикации: 15 апреля 2022 г.

Ключевые слова:
алгоритм, алфавит, граф, граф-схема, блок-схема, цикломатическое число, изоморфизм

Аннотация. Представлено строгое доказательство первой проблемы миллениума, а именно: $P \neq NP$, которая была озвучена в 1971 г. в статье Стивена Кука и положила начало долгой борьбе за ее осмысление и доказательство. Проблема тесно связана с понятием комбинаторного взрыва, возникшего в начале 1970-х гг. Она стала символом тех громадных трудностей, с которыми приходится сталкиваться разработчикам алгоритмов и программ, поскольку сложность решаемых задач с каждым днем растет. Предлагаемое доказательство основано на достижениях теории графов и теории алгоритмов. Обосновывается необходимое условие того, чтобы произвольный алгоритм мог быть решен с помощью машины Тьюринга и приводятся необходимые теоремы. Далее с помощью теории алгоритмов и теории графов доказывается, что нормализованные графы (визуализации алгоритмов) относительно такой характеристики их сложности, как цикломатическое число, распадаются на три непересекающихся множества, которые обладают различными свойствами. Эти свойства определяются структурными особенностями графов, их можно учесть при разработке алгоритмов и программ для решения массовых задач. Доказывается разделение алгоритмов массовых задач на непересекающиеся множества, которые соответствуют граф-схемам (блок-схемам) полиномиальных (P) или переборных (NP) алгоритмов. Этим обосновывается достаточное условие, которое, собственно, и подтверждает, что $P \neq NP$.

Для цитирования
Малинина Н.Л. Необходимые и достаточные условия разделения структур алгоритмов на непересекающиеся множества: полиномиальные и переборные алгоритмы // Вестник Российского университета дружбы народов. Серия: Инженерные исследования. 2022. Т. 23. № 2. С. 108–116. http://doi.org/10.22363/2312-8143-2022-23-2-108-116

Necessary and sufficient conditions for dividing the structure of algorithms into non-intersecting sets: polynomial and enumeration algorithms

Natalia L. Malinina

Moscow Aviation Institute (National Research University), *Moscow, Russian Federation*
✉ malinina806@gmail.com

Article history
Received: January 28, 2022
Revised: March 11, 2022
Accepted: April 15, 2022

Abstract. The article is devoted to a rigorous proof of the first millennium problem, which is named as $P \neq NP$. This problem was raised in 1971 by S. Cook and marked the beginning of a long struggle in order to understand and prove it. The problem is closely related to the concept of a combinatorial explosion, which

Keywords:
algorithm, alphabet, graph, graph-scheme, block-graph, cyclomatic number, isomorphism

concept was aroused in the early 1970s and became a symbol of the enormous difficulties that developers of algorithms and programs have to face, since the complexity of the tasks that have to be solved is growing every day. The presented proof is based on the achievements of graph theory and algorithm theory. Necessary conditions (normalizing), to which arbitrary algorithm must satisfy in order to be solved with a help of a Turing machine, are proved in the article. Further, using the theory of algorithms and graph theory, it is proved that normalized (necessary condition) graphs (visualization of algorithms) with respect to such a characteristic of their complexity as a cyclomatic number fall into three non-intersecting sets that have different properties. These properties are determined by the structural features of graphs, and they can be taken into account when developing algorithms and programs for solving mass problems. The division of algorithms of mass problems into three non-intersecting sets is proved. Such division corresponds with graph-schemes, or block-schemes of polynomial (P) or enumeration (NP) algorithms. This proves a sufficient condition, to which algorithms must satisfy in order to belong to different classes and actually confirm that $P \neq NP$.

For citation
Malinina NL. Necessary and sufficient conditions for dividing the structure of algorithms into non-intersecting sets: polynomial and enumeration algorithms. *RUDN Journal of Engineering Research*. 2022;23(2):108–116. http://doi.org/10.22363/2312-8143-2022-23-2-108-116

Введение

Проблема PvsNP, или проблема С. Кука [1; 2], как ее называют, сейчас стоит первой в топ-листе проблем миллениума. Настоящая статья продолжает проект двенадцатилетней давности, сообщение о котором было сделано автором на Международном математическом конгрессе в Хайдарабаде [3].

Отношения между классами P и NP рассматриваются в теории вычислительной сложности (раздел теории вычислений), изучающей ресурсы, необходимые для решения некоторой задачи. Начиная с 1971 г. попыткам ее решения посвящало свое время и усилия множество топологов, создателей алгоритмов и других ученых. Наиболее цитируемым автором по этой теме является А. Разборов [4]. Существует также специальный сайт, посвященный данной проблеме[1]. На нем представлены ссылки на 116 статей, посвященных ее возможному решению. Остановимся лишь на тех статьях, что опубликованы в реферируемых журналах. Одна из первых – статья М. Яннакакиса [5]. В ней нет решения самой проблемы, а только доказывается, что некоторый частный подход к доказательству не работает.

Ссылок, посвященных работам, доказывающим, что $P = NP$, несколько больше. В основном они посвящены удачным попыткам создать полиномиальные алгоритмы для некоторых частных случаев. Однако не следует забывать о том, что число массовых задач, которые мы не можем точно решить без применения переборных алгоритмов, ширится с каждым днем.

Чуть меньше работ написано в доказательство, что $P \neq NP$. Практически невозможно тщательно просмотреть все работы (их больше пятидесяти). При этом часть из них базируется на том факте, что встречаются модели (конкретные частные случаи), для которых не могут быть найдены полиномиальные алгоритмы и, соответственно, делается заключение, что $P \neq NP$. В частности, это работа Р. Валиева [6]. Интересна статья А. Анилла [7], в которой он приходит к доказательству $P \neq NP$, применяя принцип возрастания энтропии для исследования вычислительной сложности. В работе В. Иванова доказательство основано на более точных оценках нижних границ временной сложности, которые справедливы для всех алгоритмов решения задачи [8]. Самая последняя работа тоже посвящена доказательству $P \neq NP$ [9]; она выложена в интернете, но мнение математического сообщества еще не известно. В некотором числе статей найдены ошибки, в частности в доказательствах Ананда, Делиокара, Виана, Барбоса[2].

Следует отметить, что только доказательство того факта, что $P \neq NP$, ничего не даст нам для

[1] The P-versus-NP page. URL: https://www.win.tue.nl/~gwoegi/P-versus-NP.htm (дата обращения: 26.09.2016).

[2] The P-versus-NP page. URL: https://www.win.tue.nl/~gwoegi/P-versus-NP.htm (дата обращения: 26.09.2016).

решения задач из практической области. Необходимо разделить задачи по некоторым структурным признакам. Это позволит уже на первом этапе понимать, какой сложности будет алгоритм решения практической задачи.

Что самое главное мы знаем и не знаем о проблеме PvsNP [2]?

1. Знаем: для массовых задач, принадлежащих области P, мы можем создавать линейные или нелинейные алгоритмы и получать решения точные или практически точные.

2. Знаем: для массовых задач из области NP мы можем получить точные решения только с помощью переборных алгоритмов или получать приемлемые решения с помощью нелинейных или экспоненциальных алгоритмов.

3. Не знаем: как определить принадлежность задачи (алгоритма, который ее решает) той или иной области.

Представляется, что можно подобраться к решению проблемы PvsNP с другой стороны, с позиций необходимости решения практических задач. Это позволит применить достижения не только топологии, но также теории графов и теории алгоритмов.

1. Проблема вычислений

Итак, в проблеме PvsNP нас должна интересовать именно возможность вычислений, то есть создание алгоритмов и программ, решающих задачу. При этом не только желательно, но и необходимо, получать экономные алгоритмы и программы. С одной стороны, нужно сокращать временные ресурсы и ресурсы памяти на решения задач из области NP. С другой стороны, на создание и обслуживание мощных серверов, которые требуются для решения таких задач, уходит много энергетических ресурсов. Количество массовых задач из области NP множится с каждым годом. Это не только обязательные и жизненно важные задачи, но и развлекательные, такие как игры и социальные сети.

Вычисления мы производим с помощью компьютеров. Известно, что компьютер обрабатывает данные или решает только те задачи, для которых можно создать алгоритмы или программы, соответствующие тезису Черча – Тьюринга [10; 11]. Это означает, что программы должны обладать свойством эффективной рекурсивности.

Как добиться этого волшебного свойства? Оно предписано принципом нормализации Маркова: «алгоритм должен быть нормальным для того, чтобы его могла обрабатывать машина Тьюринга» [12]. В принципе, все алгоритмы нормализуемы, это подтверждается практикой разработки алгоритмов и программ [12]. Все известные алгоритмические схемы и их композиции (с точностью до эквивалентности) приводят к нормальным алгоритмам. Операторы в алгоритмах реализуются в определенном порядке или в порядке их нумерации. В свою очередь нумерация операторов может быть выполнена, если множество операторов рекурсивно. Однако ни одна из алгоритмических систем не имеет какого-либо наперед заданного способа нумерации операторов [12; 13]. Для создания у алгоритма или программы свойства рекурсивности необходимо расширить алфавит алгоритма, при этом достаточно добавить только одну букву [12]. Известно также, что алгоритмы могут быть визуализированы с помощью графов.

Возникает следующий вопрос: как добавить эту букву? Как сделать алгоритм нормальным или обладающим свойством эффективной рекурсивности? Данная задача была решена в 1972 г. в докторской диссертации Л. Малинина, но опубликована только в 2009 г. в книге «Изоморфизм графов в теоремах и алгоритмах» [14].

2. Расширение теории графов и связь с теорией алгоритмов

Обратимся к теории графов. До публикации [14], посвященной решению задачи изоморфизма графов, в теории графов не было решения проблемы четкого определения возможной двойственности графов. Все знают, что реберный граф (граф-схема) всегда имеет двойственный ему вершинный граф (блок-схема). А вот реберный граф далеко не всегда имеет двойственный ему вершинный [15] (рис. 1).

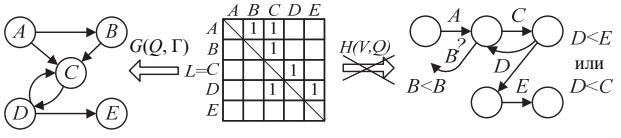

Рис. 1. Необязательность двойственности реберных и вершинных графов

Расширение теории графов в [14], позволяет решать подобные задачи. В этой работе проведено широкое исследование двойственности графов и доказаны необходимые и достаточные условия

того, чтобы матрица смежности одновременно была бы матрицей смежности как реберного, так и вершинного графа. Следует привести основные теоремы, которые ведут к понятиям двойственности графов и того, как добиться этой двойственности.

В основной, или главной, теореме доказываются условия, которым должна отвечать матрица смежности, чтобы граф, соответствующий ей, обладал свойствами двойственности.

Теорема 1. «Квазиканоническая матрица смежности»[3]. Теорема о квазиканонической матрице смежности устанавливает необходимые и достаточные условия того, чтобы матрица непосредственных путей L имела двойственный характер, то есть могла быть одновременно матрицей E смежности вершин графа G и матрицей R смежности дуг графа H при условии, что цикломатические числа этих графов могут быть различны. Теорема была доказана для случая ориентированных графов[4].

Дано: множество $Q = \{q_i\}$ и $L = QQ$ вида $q_i < q_j$, а $L = \|l_{ij}\|_1^n = \|e_{ij}\|_1^n$ для графа $G(Q, \Gamma)$ (рис. 1). Тогда $\|l_{ij}\|_1^n = \|r_{ij}\|_1^n = R$ для графа $H(V, Q)$ только в том случае, если соблюдаются условия (1):

$L = \|l_{ij}\|_1^n$ порождает $C_n = \|c_{ij}\|_1^n = [0]$.

Минор $|l_{ij}|_1^{n-1}$ каждого $l_{ij} = 1$ порождает

$$C_{n-1} = \|c_{ij}\|_1^n = [0], \qquad (1)$$

где

$$c_{ij} = l_{ij}(\Delta_{j/i} s_{ij} + \Delta_{i/j} s_{ij}),$$

$$\Delta_{j/i} s_{ij} = \left(s_{ij} - \min_j s_{ij}\right)_i,$$

$$\Delta_{i/j} s_{ij} = \left(s_{ij} - \min_i s_{ij}\right)_j,$$

$$k = n, (n-1),$$

$$s_{ij} = l_{ij}\left(\sum_{\substack{i=1 \\ j}}^{k} l_{ij} + \sum_{\substack{j=1 \\ i}}^{k} l_{ij}\right),$$

$$\left(\min_j s_{ij}\right)_i = \min_{j/i} s_{ij} \in \{s_{ij} \neq 0\},$$

$$\left(\min_i s_{ij}\right)_j = \min_{i/j} s_{ij} \in \{s_{ij} \neq 0\}.$$

Для доказательства теоремы необходимо показать, что матрица L, удовлетворяющая условиям (1) и рассматриваемая как матрица R – смежности дуг графа H, содержит всю информацию для однозначного составления матрицы F – матрицы смежности вершин графа H. Доказательство приведено в [14].

Цикломатические числа графов всегда удовлетворяют условию

$$\nu(G_q) \geq \nu(H_q). \qquad (2)$$

Условие (2) отражает определенную вырожденность двойственности (квазидвойственность) квазиканонической матрицы смежности. Кроме того, это условие отражает возможность присутствия в графе H_q сложных вершин.

Теорема 1 определяет условия существования квазиканонической матрицы смежности, хотя в практических приложениях такие готовые матрицы могут встречаться лишь случайно. Становится актуальным найти способ преобразования к необходимой форме любой произвольной матрицы непосредственных путей, которая не удовлетворяет требованиям теоремы 1. Преобразование матрицы L должно быть таким, чтобы система отношений между исходными элементами оставалась неизменной, то есть преобразование должно быть консервативным по отношению к системе бинарных отношений, заданной на множестве $Q = \{q_i\}$.

Преобразование матрицы непосредственных путей к квазиканонической форме. Определения:

1. Под *консервативным преобразованием бинарного отношения* двух элементов q_i и q_j условимся понимать такое преобразование, которое позволяет вводить или исключать дополнительные элементы из множества $Q = \{q_i\}$, не меняя самого отношения между элементами (q_i, q_j). Таким может стать преобразование, основанное

[3] Номера теорем в статье соответствуют номерам теорем в книге.

[4] Ориентированный граф может быть превращен в неориентированный граф путем удвоения дуг графа.

на свойстве транзитивности бинарного отношения. Например, исходная пара $(q_i, q_j) \in Q$. Пусть элементы q_i и q_j связаны отношением $q_i < q_j$. Примем два условия: $q_i < q_z$ и $q_z < q_j$ и преобразуем исходное выражение. Получим $q_i < q_z < q_j$. Очевидно, что отношения $q_i < q_j$ и $q_i < q_z < q_j$ эквивалентны относительно исходной пары элементов. Поэтому введение элемента q_z в отношение $q_i < q_j$ консервативно по отношению к этому отношению в исходной паре (q_i, q_j).

2. Под Δn-*преобразованием матрицы* L условимся понимать дополнение матрицы L одним рядом (строкой и столбцом) q_{n+1} при одновременной замене отношения $q_x < q_y$ парой бинарных отношений $q_x < q_{n+1}$ и $q_{n+1} < q_y$. Очевидно, что Δn-преобразование консервативно по отношению к бинарному отношению в исходной паре (q_i, q_j) и не нарушает такого критерия структурного подобия, как система бинарных отношений.

Теорема 2 «Квазинормализация матрицы бинарных отношений L». Любая матрица $\|l_{ij}\|_1^n$ непосредственных путей может быть приведена к квазиканонической (квазинормальной) форме $\|l_{ij}\|_1^{n+s_q}$, где $s_q \le n^2 - 1$, если применить Δn-преобразование к тем s_q-элементам матрицы $\|l_{ij}\|_1^n$, которые не удовлетворяют требованиям теоремы 1.

Доказана сходимость этого преобразования. Доказательство теоремы приведено в [14].

Если соотносить вышесказанное с теорией алгоритмов и тезисом Маркова, то доказанное преобразование автоматически добавляет к алфавиту произвольного алгоритма недостающую букву и делает его нормальным или рекурсивным. Доказано, что алгоритм преобразования (нормализации) является локальным, хотя с помощью некоторых ухищрений он может стать линейным.

В частном случае, когда $\nu(G_q) = \nu(H_q)$, матрица $L_q = R_q$ называется канонической или нормальной. Для случая строгой двойственности (цикломатические числа равны) была доказана теорема 4.

Теорема 4 «Каноническая матрица смежности». Пусть исходный связный граф G задан матрицей $\|e_{ij}\|_1^n$ – смежности вершин, которой соответствует квазиканоническая матрица $\|r_{ij}\|_1^{n+s_q}$ смежности дуг связного реберного графа H_q. Для того чтобы цикломатическое число $\nu(H_q)$ графа H_q было равно цикломатическому числу $\nu(G)$ исходного графа G, необходимо и достаточно, чтобы все вершины полученного графа H_q были простыми. Или, что то же самое, чтобы для всех $r_{xy} = 1$ соблюдались следующие условия:

$$\left.\begin{array}{l} \text{Если } \sum\limits_{\substack{i=1 \\ j=y}}^{n+s_q} r_{ij} \ge 1, \text{то } \sum\limits_{\substack{j=1 \\ i=x}}^{n+s_q} r_{ij} = 1 \\ \text{Если } \sum\limits_{\substack{j=1 \\ i=x}}^{n+s_q} r_{ij} \ge 1, \text{то } \sum\limits_{\substack{i=1 \\ j=y}}^{n+s_q} r_{ij} = 1 \end{array}\right\}. \qquad (3)$$

В результате нормализации графа и последующего упорядочения, получаем реберный граф в виде графа Кенига. В [14] доказано, что канонические графы без контуров, полученные в результате нормализации, обладают свойством рекурсивности, которое основано на разбиении канонических матриц смежности на взаимно непересекающиеся подматрицы. Это позволяет строить рекуррентные локальные алгоритмы их упорядочения. Такая возможность сообщает графам (алгоритмам и программам с такой структурой) свойство эффективной рекурсивности (нумерация осуществляется за один проход вдоль множества рядов матрицы).

Получение упорядоченного графа Кенига – условие необходимое, но не достаточное. Теоретически мы получаем возможность создать нормальный алгоритм в виде упорядоченного графа Кенига. Однако возможное присутствие в исходном графе контуров и в полученном графе H_q сложных вершин приводит к некоторым проблемам. Цикломатическое число некоторых графов при Δn-преобразовании растет, то есть не совсем понятно, что происходит с графами, в которых есть контуры. Таким образом, возможность автоматизации процесса нормализации алгоритма не решает проблему доказательства $P \text{vs} NP$.

Перед нами встает еще одна проблема – как разделить массовые задачи на классы, чтобы по определенным признакам графа задачи сразу было понятно, к какому классу принадлежит задача: P или NP. Здесь мы сталкиваемся с проблемой сложности графа. Характеристикой слож-

ности графа является цикломатическое число ν. Поэтому следует обратиться к такой характеристике сложности графа, как цикломатическое число, и разобраться является ли оно инвариантом графа и, если является, то в каких случаях.

3. Цикломатическое число и изоморфизм

Существует достаточное количество инвариантов, с помощью которых можно сравнивать графы между собой. Среди них есть такая характеристика, как цикломатическое число. Работы О. Оре [15], К. Бержа [16], А.А. Зыкова [17] и Ф. Харари [18–19] говорят о том, что цикломатическое число равно количеству независимых циклов в графе, поэтому оно является характеристикой сложности графа. Для понимания является ли цикломатическое число инвариантом, на который можно положиться, следует обратиться к изучению такого преобразования графов, как превращение вершинного графа в реберный граф, полученного реберного графа снова в вершинный граф и т. д. [14]. Назовем эту операцию конвертированием графов (рис. 2).

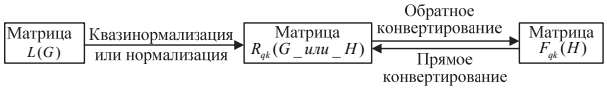

Рис. 2. Схема преобразования графов

Прямым конвертированием назовем операцию построения вершинного графа по реберному графу. Любой ориентированный граф может быть подвергнут операции прямого конвертирования любое число раз. Прямое конвертирование может быть каноническим ($\nu = \text{const}$) или квазиканоническим ($\nu \uparrow$).

Обратным конвертированием назовем операцию построения реберного графа по заданному вершинному графу. Граф может быть подвергнут операции обратного конвертирования исключительно в том случае, если матрица смежности его вершин имеет каноническую или квазиканоническую форму. Обе операции конвертирования подробно рассмотрены в [14]. Следует привести только две теоремы.

Теорема 9. Если граф H_1 в процессе его последовательного прямого конвертирования порождает только канонические графы H_{kj} $(j = 1, 2, 3 \ldots, M)$, то числа вершин этих последовательно получаемых графов определяются линейной зависимостью от номера операции конвертирования $(j-1)$, то есть

$$n_j = n_1 + \Delta n^*_{(j-1)}. \qquad (4)$$

Теорема 10. Если граф H_1 в процессе его последовательного прямого конвертирования порождает и канонические, и квазиканонические графы или только квазиканонические графы, то числа вершин этих последовательно получаемых графов определяются выражением

$$n_j = n_1 + \sum_{\xi=1}^{\xi=(j-1)} \Delta n_\xi, \qquad (5)$$

где

$$\Delta n_\xi = \Delta n_{(\xi-1)} + \Delta\nu(H_{(\xi-1)}, \qquad (6)$$

где $\xi = 1, 2, \ldots (j-1)$; $j = 1, 2, 3, \ldots M$.

В свою очередь значения $\Delta\nu\left(H_{(\xi-1)}\right)$ определяются структурой исходного графа H_1. Таким образом, показано, что возрастание числа вершин графов, получаемых при последовательном прямом конвертировании, связано с цикломатическим числом и видом конвертирования (каноническое или квазиканоническое), а прирост цикломатического числа при таком конвертировании зависит от структуры графа.

Доказанные теоремы [14] показывают, что все ориентированные графы могут быть разбиты на два класса:

1) графы, для которых цикломатическое число всегда является инвариантом прямого конвертирования;

2) графы, для которых цикломатическое число на части шагов или на всех шагах прямого конвертирования не является инвариантом конвертирования.

В результате тщательного исследования операций конвертирования и определения свойств структур графов относительно сочетаний различного рода вершин и ребер между ними выявлены необходимые и достаточные признаки графов, которые дали возможность определить необходимые и достаточные признаки для характеристики каждого из классов графов. Доказанные теоремы приведены в [14]. Также рассмотрено более точно понятие пути и контура в ориентированном графе. Реальным процессам могут соответствовать только такие контуры, которые имеют по крайней мере один «вход» и один «выход». Понятие пути и контура уже введено много ранее [15; 16; 18], но в [14] добавлена функция, позволяющая различать пути в графе один от другого. Этот признак определялся с помощью функции сумм степеней вершин, через которые проходит путь. Ока-

залось, что эти функции можно разделить на два класса, которые описаны в [14]. Соответственно, пути также можно разделить на два класса. Доказанные теоремы приведены в [14].

Изучение различных путей в графах привело к изучению различных комбинаций интервалов между вершинами графа [14]. Вершины графов определялись как положительные (один вход и много выходов) и отрицательные (много входов и один выход). Кроме того, вершины графа также разделялись на простейшие (один вход и один выход), простые (один/несколько входов и несколько/один выходов) и сложные (несколько входов и несколько выходов). Особенное внимание было обращено на интервал типа l_{31}, имеющий сложные вершины на обоих концах (рис. 3, *а, б*).

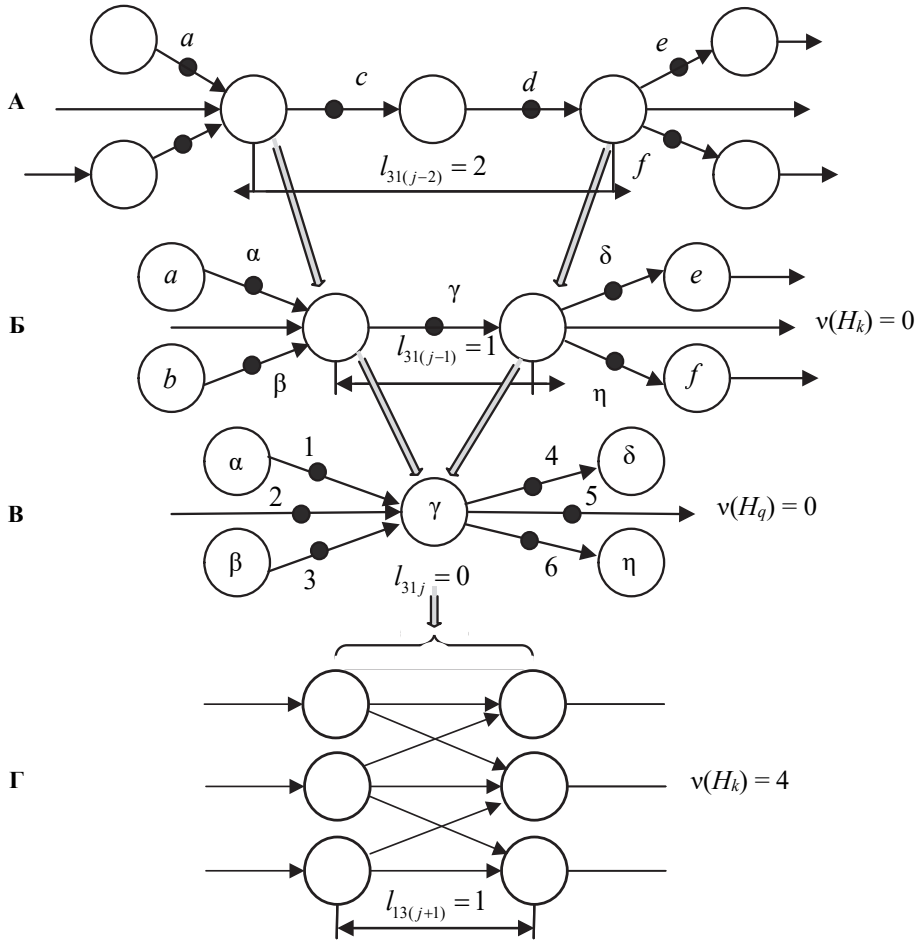

Рис. 3. Преобразование интервала l_{31} в сложную вершину и появление независимых циклов

Главной особенностью этого интервала является наличие отрицательной вершины на входе, а положительной вершины на выходе (рис. 3, *б*). Такой интервал на первом шаге последовательного конвертирования превращается в сложную вершину (рис. 3, *в*). Затем, на следующем шаге конвертирования, эта сложная вершина (одна) преобразуется в четыре независимых цикла (рис. 3, *г*). Появление новых циклов вызывает увеличение цикломатического числа. Увеличение цикломатического числа влечет рост и сложности графа и, соответственно, алгоритма или программы, структура которых определяется структурой графа.

В результате дальнейшего исследования оказалось, что ориентированные графы разбиваются на три непересекающихся класса:

1. ***Голономные графы.*** Для них цикломатическое число является регулярным инвариантом конвертирования, независимо от числа шагов последовательного конвертирования. Благодаря этому число вершин графов, полученных из исходного графа в результате его последовательного конвертирования, линейно зависит от числа шагов конвертирования. Эти графы не должны содержать контуры и интервалы типа l_{31} [14].

2. ***Ограниченно-гетерономные графы.*** Такие графы имеют границу гетерономности по числу шагов последовательного конвертирования. До достижения этой границы цикломатическое число не является регулярным инвариантом конвертирования. После достижения этой границы в результате очередного шага конвертирования порождается голономный граф и цикломатическое число становится регулярным инвариантом конвертирования независимо от числа шагов дальнейшего последовательного конвертирования. Структура подобных графов может содержать интервалы типа l_{31}, но не должна иметь контуры.

3. ***Прогрессивно-гетерономные графы*** не имеют границы гетерономности по числу шагов последовательного конвертирования. В результате цикломатическое число прогрессивно-гетерономного графа не становится регулярным инвариантом последовательного конвертирования ни при каком, сколь угодно большом, числе шагов конвертирования. В структуре таких графов присутствуют как контуры, так и интервалы типа l_{31}. В [14, глава 4] исследовано и представлено в виде таблицы 1 соответствие структур графов и различных блок-схем и граф-схем различных алгоритмов (таблица).

Сопоставление графов и алгоритмических схем

Вид исходного графа	Упорядоченный граф	№ в табл. 1 [14]	Полное наименование алгоритмической схемы
Гамильтонов граф $G^{(H)}$ при $\sum_j l_{ij} = 2$ $(j = 2,3,\dots,(n-1))$	$G^{(H)}\left(L^{(H)}\right)$	6	Нормальная одноканальная двухадресная алгоритмическая блок-схема
	$G_K^{(H)}\left(R_K^{(H)}\right)$	7	Нормальная одноканальная двухадресная алгоритмическая граф-схема Калужнина
	$H_K^{(H)}\left(F_K^{(H)}\right)$	8	Нормальная обыкновенная одноканальная двухадресная алгоритмическая граф-схема
		9	Обобщенная нормальная одноканальная двухадресная алгоритмическая граф-схема
	$D_K^{(H)}\left(F_K^{(H)}\right)$	10	Нормальная операторная одноканальная двухадресная алгоритмическая граф-схема
Произвольный граф $G^{(K)}$ при $\sum_j l_{ij} = 2$ $(j = 2,3,\dots,(n-1))$	$G^{(K)}\left(L^{(K)}\right)$	16	Двухадресная алгоритмическая блок-схема с произвольным числом каналов
	$G_K^{(K)}\left(R_K^{(K)}\right)$	17	Нормальная двухадресная алгоритмическая граф-схема Калужнина с произвольным числом каналов
	$H_K^{(K)}\left(F_K^{(K)}\right)$	18	Нормальная обыкновенная двухадресная алгоритмическая граф-схема с произвольным числом каналов
		19	Обобщенная нормальная двухадресная алгоритмическая граф-схема с произвольным числом каналов
	$D_K^{(K)}\left(F_K^{(K)}\right)$	20	Нормальная операторная двухадресная алгоритмическая граф-схема с произвольным числом каналов
Произвольный граф G	$G(L)$	26	N-адресная алгоритмическая блок-схема с произвольным числом каналов
	$G_K(R_K)$	27	Нормальная сопряженная N-адресная алгоритмическая граф-схема с произвольным числом каналов
	$H_K(F_K)$	28	Нормальная обыкновенная N-адресная алгоритмическая граф-схема с произвольным числом каналов
		29	Обобщенная нормальная N-адресная алгоритмическая граф-схема с произвольным числом каналов
	$D_K(F_K)$	30	Нормальная операторная N-адресная алгоритмическая граф-схема с произвольным числом каналов
Полный граф G_0^*	$G_0^*(L_0^*)$	36	Полная алгоритмическая блок-схема
	$G_{0u}^*(R_{0u}^*)$	37	Нормальная сопряженная полная алгоритмическая граф-схема
	$H_{0u}^*(F_{0u}^*)$	38	Нормальная обыкновенная полная алгоритмическая граф-схема
		39	Нормальная обобщенная полная алгоритмическая граф-схема
	$D_{0u}^*(F_{0u}^*)$	40	Нормальная операторная полная алгоритмическая граф-схема
$G^{(K)}, G, G_0^*$	$G^{(H)}\left(L^{(H)}\right)$	41	Нормальная алгоритмическая блок-схема (одноканальная двухадресная)

Заключение

Итак, множество всех алгоритмов разбивается на три класса (или на три непересекающихся множества) согласно приведенному выше разбиению множества ориентированных графов. После операции нормализации графы, а следовательно, и алгоритмы, имеющие подобную структуру, приобретают свойства рекурсивности.

Для голономных графов цикломатическое число становится инвариантом. Алгоритмы, которые после операции нормализации будут иметь подобную структуру, автоматически получают свойство эффективной рекурсивности и будут относиться к области полиномиальных.

Ограниченно-гетерономные графы должны быть подвергнуты некоторому (конечному) числу шагов прямого конвертирования, чтобы цикломатическое число стало инвариантом. Алгоритмы, имеющие такую структуру, можно назвать сводимыми к множеству полиномиальных алгоритмов.

Прогрессивно-гетерономные графы никогда не будут иметь цикломатическое число инвариантом. Поэтому алгоритмы, имеющие подобную структуру, всегда будут принадлежать к области NP, хотя в частных случаях операция нормализации может уменьшить число вариантов перебора.

И наконец, главное, что можно сказать в доказательство тезиса $P \neq NP$.

Необходимое условие – это получение упорядоченного графа Кенига с помощью нормализации произвольного графа.

Достаточное условие: поскольку произвольные графы разделяются на три непересекающихся класса соответственно их структурным характеристикам, то и алгоритмы, которые им соответствуют, также распадаются на три непересекающихся класса, и существует класс алгоритмов, которые не могут быть сведены к классу полиномиальных. Они всегда будут оставаться переборными, то есть NP-трудными.

Все это подтверждает тезис о том, что $P \neq NP$.

Список литературы

1. *Cook S.A.* The complexity of theorem-proving procedures // Conference Record of Third Annual ACM Symposium on Theory of Computing. New York: Association for Computing Machinery, 1971. Pp. 151–158. https://doi.org/10.1145/800157.8050472

2. *Гэри М., Джонсон Д.* Вычислительные машины и труднорешаемые задачи. Москва: Мир, 1982. 419 с.

3. *Malinina N.* On a principal impossibility to prove *P = NP* // International Congress of Mathematicians. Hyderabad: Hundistan Book Agency, 2010. Pp. 484–485.

4. *Razborov A.A.* Lower bounds for the polinomial calculus // Computational Complexity. 1998. Vol. 7. Pp. 291–324.

5. *Yannakakis M.* Expressing combinatorial optimization problems by liner programs // Journal of Computer and System Sciences. 1991. Vol. 43. Pp. 441–466.

6. *Valeyev R.* The lower border of complexity of algorithm of elementary *NP*-complete task // World Applied Science Journal. 2013. Vol. 8. Pp. 1072–1083.

7. *Annila A.* Physical portrayal of computational complexity // Computational Mathematics. 2009. Vol. 2012. https://doi.org/10.5402/2012/321372

8. *Ivanov V.* A short proof that NP is not P // International Journal of Pure and Applied Mathematics. 2014. Vol. 94. No 1. Pp. 81–88.

9. *Dowd M.* On the provability of P = NP. 2020. Pp. 1–13. Preprint. URL: https://www.researchgate.net/publication/339426546_On_the_Provability_of_PNP (accessed: 22.02.2020).

10. *Church A.* A note on the Entscheidungsproblem // The Journal of Symbolic Logic. 2014. Vol. 1. No. 1. Pp. 40–41. https://doi.org/10.2307/2269326

11. *Turing A.* On computable numbers, with an application to the Entscheidungsproblem // Proceedings of the London Mathematical Society. 1937. Vol. s2–42. Issue 1. Pp. 230–265.

12. *Марков А.А., Нагорный Н.М.* Теория алгорифмов. М.: Фазис, 1996.

13. *Глушков В.М.* Теория алгоритмов. Киев: КВИРТУ ПВО, 1961.

14. *Малинин Л.И., Малинина Н.Л.* Изомофизм графов в теоремах и алгоритмах. М.: ЛИБРОКОМ, 2009. 250 с.

15. *Оре О.* Теория графов. М.: Наука, 1968. 350 с.

16. *Берж К.* Теория графов и ее применения. М.: Иностранная литература, 1962. 319 с.

17. *Зыков А.А.* Теория конечных графов. Новосибирск: Наука, Сибирское отделение, 1969. 554 с.

18. *Harary F., Palmer E.* Graphical enumeration. London: Academic Press, 1973. https://doi.org/10.1016/c2013-0-10826-4

19. *Харари Ф.* Комбинаторные задачи перечисления графов / под ред. Э. Беккенбаха. М.: Мир, 1968. 363 с.

Сведения об авторе

Малинина Наталия Леонидовна, кандидат физико-математических наук, доцент кафедры 604, аэрокосмический факультет, Московский авиационный институт (национальный исследовательский университет), Российская Федерация, 125993, Москва, Волоколамское шоссе, д. 4; ORCID: 0000-0003-0116-5889, eLIBRARY AuthorID: 502378; malinina806@gmail.com