УДК 519.7, 004.8

СЛОЖНОСТЬ АЛГОРИТМОВ ПЕРВОГО РОДА

Цветков Виктор Яковлевич,
д-р техн. наук, профессор, заместитель руководителя Центра стратегического анализа и развития АО «НИИАС»,
e-mail: cvj2@mail.ru,
Научно-исследовательский и проектно-конструкторский институт информатизации, автоматизации и связи на железнодорожном транспорте (НИИАС), АО «НИИАС», г. Москва

Статья проводит анализ сложности алгоритмов первого рода. Анализируются признаки, по которым алгоритм можно отнести к алгоритму первого рода. Вводится новое понятие «результативность алгоритма» в теорию сложности вычислений. Показано различие между алгоритмами первого и второго рода на примере детерминированной и недетерминированной машины Тьюринга. Вычислительная модель раскрывается на примере машины Тьюринга. Дана обобщенная модель алгоритмов первого рода. Раскрывается содержание и обосновывается необходимость понятия асимптотическая сложность. Основной анализ выполняется с классами и видами временной сложности. Приведены примеры и дан анализ алгоритмов постоянного времени и линейного времени. Отмечена условность некоторых видов сложности. Она состоит в том, что алгоритмы, которые относятся к одному классу или виду сложности физически используют разное время вычислений. Отмечена ситуация, при которой более простой вид сложности затрачивает больше времени на вычисления, чем более сложный. Отмечен недостаток существующей теории алгоритмической сложности – исключение из анализа сложности когнитивного фактора и когнитивной сложности. Это обусловлено исключением понятия результативности алгоритма при анализе или оценки сложности. Существующая теория сложности акцентирует внимание на вычислениях и времени вычислений. Но главным в вычислениях является результат. Если результат вычислений не качественный или нечеткий, то время вычислений теряет свою важность. Соответственно оценка классов сложности должна привязываться не только ко времени, но и к качеству результата. Намечены результаты дальнейших исследований.

Ключевые слова: алгоритмы, классы сложности, виды сложности, алгоритм первого рода, алгоритм второго рода, вычислительная модель, машина Тьюринга

THE COMPLEXITY OF THE FIRST KIND OF ALGORITHMS

Tsvetkov V.Ya.,
doctor of technical sciences, professor, professor Center for strategic analysis and development, the deputy head,
e-mail: cvj2@mail.ru,
Research and Design Institute of Informatization, Automation and Communication in Railway Transport (NIIAS), JSC «NIIAS», Moscow

The article analyzes the complexity of algorithms of the first kind. The features are analyzed according to which the algorithm can be classified as an algorithm of the first kind. A new concept is introduced in the theory of algorithmic complexity "algorithm efficiency". The difference between algorithms of the first and second kind is shown on the example of deterministic and non-deterministic Turing machines. The computational model is disclosed on the example of a Turing machine. A generalized model of algorithms of the first kind is given. The content is revealed and the necessity of the concept of asymptotic complexity is substantiated. Basic analysis is performed with classes and types of time complexity. Examples are given and the analysis of algorithms of constant time and linear time is given. The conventionality of some types of complexity is noted. It consists in the fact that algorithms that belong to the same class or type of complexity physically use different computation

times. A situation is noted in which a simpler type of complexity spends more time on calculations than a more complex one. The drawback of the existing theory of algorithmic complexity is noted – the exclusion of the cognitive factor and cognitive complexity from the analysis of complexity. This is due to the exclusion of the concept of the effectiveness of an algorithm when analyzing or assessing complexity. The existing theory of complexity focuses on computation and computation time. But the main thing in the calculations is the result. If the result of the calculations is not qualitative or unclear, then the computation time loses its importance. Accordingly, the assessment of complexity classes should be tied not only to time, but also to the quality of the result. Results of further research are outlined.

Keywords: algorithms, complexity classes, types of complexity, type I algorithm, type II algorithm, computational model, Turing machine

DOI 10.21777/2500-2112-2020-4-73-80

Введение

Следует разделять понятия “теория сложности” и “теория сложности вычислений”. Теория сложности вычислений является подмножеством теории сложности и связана, в первую очередь, с вычислительной сложностью алгоритмов. Алгоритмическая обработка информации соответствует трансформации входной информации (входные данные) в результат обработки [1]. В вычислениях алгоритм выполняет роль посредника в преобразовании информации. В современном понимании алгоритм представляет более широкое понятие, чем схема вычислений [2]. Существуют алгоритмы развития [3], алгоритмы познания [4], алгоритмы поиска [5], алгоритмы распознавания [6], алгоритмы идентификации [7], алгоритмы сепарации, например методом гиперплоскости [8], алгоритмы управления знаниями (KM Algorithm) [9], мультиагентные алгоритмы [10] и другие.

Опыт обработки информации показывает, что по типу получения решений можно выделить два типа алгоритмов. Первый тип алгоритмов называют алгоритмом первого рода или прямым алгоритмом [11]. Второй тип алгоритмов называют алгоритмом второго рода. С помощью алгоритма первого рода решение задачи можно получить с помощью одной вычислительной траектории. На рисунке 1 приведен обобщенный алгоритм первого рода.

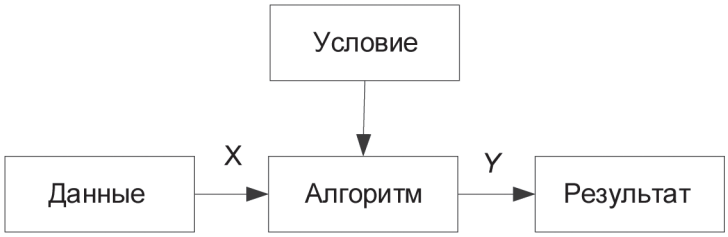

Рисунок 1 – Обобщенный алгоритм первого рода

Все алгоритмы первого рода характеризуются одним входом и одним выходом. Все алгоритмы первого рода можно упростить и объединить в один блок с входом (X) и выходом (Y). Большинство алгоритмов первого рода включают решения задач, которые могут быть решены детерминированной машиной Тьюринга (DTM) с ограниченными временными или пространственными ресурсами. Многие алгоритмы второго рода включают решения задач, которые могут быть решены недетерминированной машиной Тьюринга (NTM).

Для оценки сложности алгоритмов часто используют понятие «Класс временной сложности», который оценивают по временным ресурсам, необходимым для решения задачи. Для оценки сложности чаще всего применяют следующие процедуры: оценивают временные затраты для работы данного алгоритма; сравнивают два алгоритма и соотносят их либо к одному, либо к разным классам. Классы сложности алгоритмов рассмотрим как совокупность алгоритмов, решающих соразмерные вычисли-

тельные задачи. Класс сложности обычно определяется тремя факторами: тип вычислительной задачи, модель вычисления и ограниченный вычислительный ресурс. Например, класс сложности *P* (от англ. Polynomial), определяется как множество задач, решения которых может быть получено детерминированной машиной Тьюринга за полиномиальное время.

Асимптотическая сложность алгоритмов

В простой трактовке вычислительная проблема или вычислительная задача может быть представлена как сформулированный на формальном языке вопрос, на который компьютер может ответить. В этом аспекте с вычислительной проблемой и вычислительной сложностью возникает понятие информационного языка [12]. Понятия «задача» и «язык» в значительной степени синонимичны в теории вычислимости и алгоритмической сложности. Под машиной Тьюринга обычно понимается машина, которая определяет язык. С вычислительной сложностью связано понятие вычислительная модель. Вычислительная модель, в свою очередь, связана с вычислительными ресурсами: «время», «память». В теории алгоритмической сложности классы сложности определяются требованиями к ресурсам алгоритма, а не требованиями к физическим ресурсам. Основная вычислительная модель в теории алгоритмической сложности – это машина Тьюринга, хотя используют и другие модели. В машине Тьюринга вместо использования стандартных информационных единиц времени (секунда) используются информационные единицы – количество элементарных шагов, требуемых для решения задачи. В машине Тьюринга вместо использования стандартных информационных единиц объема (байты), используют информационные единицы типа количество ячеек, которые используются на ленте машины.

Время работы одного алгоритма может различаться для разных входных данных как по качественному набору данных, так и по объему. Для оценки сложности алгоритма обычно рассматривают временную сложность, которая задает максимальное количество времени, требуемое для входных данных. Такой подход приводит к понятию асимптотическая сложность [13]. Менее распространенной является усредненная сложность, которая представляет собой среднее время, затрачиваемое на входные данные определенного объема. В этих случаях временная сложность обычно выражается как функция размера входных данных [14]. Поскольку эту функцию, как правило, трудно вычислить точно, а время выполнения небольших входных данных обычно не имеет никакого значения, обычно сосредотачиваются на поведении сложности при увеличении размера входных данных, то есть на асимптотическом поведении сложности. Для сравнительной оценки сложности алгоритмов используют асимптотическую сложность, которая имеет следующие разновидности:

1) $f(n) \in O(g(n))$ – функция *f* ограничена сверху функцией *g* (с точностью до постоянного множителя) асимптотически:

$$\exists(C>0),\ n_0\colon \forall(n>n_0)\ |f(n)| \le |g(n)| \text{ или } \exists(C>0),\ n_0\colon \forall(n>n_0)\ |(n) \le Cg(n);$$

2) $f(n) \in \Omega(g(n))$ – функция *f* ограничена снизу функцией *g* (с точностью до постоянного множителя) асимптотически:

$$\exists(C>0),\ n_0\colon \forall(n>n_0)\ |f(n)| \ge C|g(n)|;$$

3) $f(n) \in \Theta(g(n))$ – функция *f* ограничена сверху и снизу функцией *g* (с точностью до постоянного множителя) асимптотически:

$$\exists(C, C'>0),\ n_0\colon \forall(n>n_0)\ C|g(n)| \le |f(n)| \le C'|g(n)|;$$

4) $f(n) \in o(g(n))$ – функция *g* доминирует над *f* асимптотически:

$$\forall(C>0),\ \exists n_0\colon \forall(n>n_0)\ |f(n)| <C|g(n)|;$$

5) $f(n) \in \omega(g(n))$ – функция *f* доминирует над *g* асимптотически:

$$\forall(C>0),\ \exists n_0\colon \forall(n>n_0)\ |f(n)| >C|g(n)|;$$

6) $f(n) \sim g(n)$ – функция *f* эквивалентна или соразмерна функции *g* асимптотически:

$$\lim f(n)/g(n)=1 \text{ при } n \to \infty.$$

Для асимптотической оценки используют нотацию: большое «О», которое используют для выражения верхней границы времени работы алгоритма. Большое *O* – одно из математических обозначений, называемых *асимптотическими обозначениями*, которые применяют для выражения поведения функ-

ции, когда ее аргумент увеличивается до бесконечности. Большое *O* было использовано математиком Полом Бахманном в конце XIX в., но иногда его называют *символом Ландау* (математика, Эдмунда Ландау). Его используют для описания ограничения сверху (см. п. 1 выше). Большое *ОМЕГА* используют для описания ограничения снизу (см. п. 2 выше).

Виды вычислительной сложности алгоритмов

Различают разные виды сложности алгоритмов. Традиционно в качестве основы используют временную сложность, которую называют вычислительной сложностью. Связывая сложность вычисления с временем вычисления. Время вычисления обозначают *T(n)*. Временная сложность обычно выражается с использованием большой буквы О: *O(n), O(n log(n)), O(n^a), O(2^n)* и другие, где *n* – размер входных данных в битах. Например, алгоритм с временной сложностью *O(n)* – это *линейный алгоритм времени* и алгоритм с временной сложностью *O(n^a)* для некоторой постоянной *a*>1, представляет собой *алгоритм* с *полиномиальным временем.*

Рассмотрим алгоритмы первого рода. Следует отметить, что разные авторы допускают противоречивые интерпретации сложности этих алгоритмов. Поэтому рассмотрим примеры и попытаемся обосновать сложность некоторых алгоритмов.

Простейший алгоритм – это алгоритм постоянного времени. Этот алгоритм осуществляет операцию за фиксированное время независимо от входных данных. Все машинные команды являются алгоритмами постоянного времени, что обозначается *O(1)*. На рисунке 2 приведен алгоритм постоянного времени на примере операции сложения.

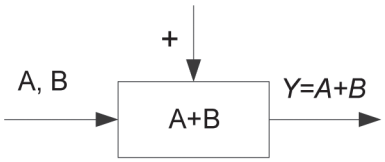

Рисунок 2 – Алгоритм постоянного времени

Рисунок 2 является полной реализацией схемы на рисунке 1. Входные данные представляют собой ограниченный набор *A*, *B*. Условие представляет собой единичную операцию, в данном случае сложение. В общем случае это может быть любая операция с фиксированным временем. Выход представляет собой результат действия операции или сумму. Следует отметить, что время операции на схеме рисунке 2 зависит от типа данных. Операции с целыми числами выполняются быстрее, чем с вещественными числами. Операции с вещественными числами выполняются быстрее, чем с числами двойной точности.

Некоторые авторы называют алгоритмом постоянного времени алгоритм вычисления среднего. Формально это допустимо, но физически некорректно. Время вычисления среднего зависит от объема выборки или массива, в котором это среднее определяется. Поэтому такой алгоритм имеет условно постоянное время вычислений. Его время зависит от объема выборки, от требуемой точности вычислений, а также от типов исходных данных. На этом примере можно подчеркнуть условность некоторых видов сложности. Она состоит в том, что алгоритмы, которые относятся к одному классу или виду сложности, физически используют разное время вычислений.

В качестве альтернативы данному алгоритму используют алгоритмы линейного времени *O(n)*. Алгоритм вычисления среднего является алгоритмом линейного времени. Если рассматривать не вычислительный алгоритм, а алгоритм решения некоторой задачи, то алгоритмом линейного времени будет алгоритм первого рода при постановке задачи или проблемы (рисунок 3).

В качестве решения может быть проект, инновационная разработка, управленческое решение и так далее. Принципиальным является то, что чем больше блоков, тем больше время выполнения алгоритма. Именно по этой причине в управлении и вычислениях осуществляют распараллеливание пото-

ков, которые участвуют в вычислениях. Алгоритм на рисунке 3 является линейным с одной единственной траекторией. Существуют модели нелинейных сетевых алгоритмов, которые допускают несколько возможных траекторий вычисления [15].

Рисунок 3 – Алгоритм линейного времени «идея – решение»

Рассмотрим еще один алгоритм линейного времени *O(n)*. Это алгоритм запроса к базе данных (DB) или поиска в базе данных (рисунок 4).

Рисунок 4 – Алгоритм запроса к базе данных

Для рисунка 4 условием решения является ∃Y(X) ∈ DB. В теории сложности считается, что в общем случае проблема решения не всегда имеет только два возможных выхода, да или нет (или попеременно 1 или 0) на любом входе. Следовательно, если решение имеет только два возможных выхода (рисунок 4), то это есть признак алгоритма первого рода. Особенностью алгоритма на рисунке 4 является однозначность результата. Оно может быть получен «да» или не получен «нет». При этом результат характеризуется единственностью значения Y.

Отметим понятие “результативность” алгоритма. Фактор результативности алгоритма не всегда оценивают при анализе его сложности. Алгоритм может работать короткое время и достичь нужного результата или поставленной цели. Алгоритм может работать долго, но ничего не достичь. Примером последнего является зацикливание. Результативность алгоритма есть сравнение или сравнительная оценка полученного результата и целевого результата. Рассмотрим алгоритм запроса к базе данных (рисунок 4). Его результативность оценивается достаточно просто, т.е.

$$X \wedge E_i \rightarrow Y, \quad (1)$$

$$X \cap E_i \rightarrow \varnothing. \quad (2)$$

Выражение (1) говорит, что результат в схеме на рисунке 4 достигнут. На основе запроса по входной информации X путем сравнения ее с неким элементом базы данных E_i, найдена нужна выходная информация Y, отвечающая информационным потребностям запроса. Выражение (2) говорит, что результат не достигнут. В результате перебора всех элементов базы данных не получена информация, отвечающая информационным потребностям запроса. Пересечение входной информации X с элементами базы данных E_i дает пустое множество.

Выражения (1), (2) характеризуют результативность и временную сложность. Она является линейной. Для выражения (1) время работы алгоритма $T_1<T(n)$. Для выражения (2) время работы алгоритма $T_2=T(n)$. В обоих случаях имеет место временная линейная сложность или сложность линейного времени *O(n)*. Напомним, что линейная сложность характеризуется не фактическим временем работы, а асимптотическим значением, которое не превосходит *O(n)*. В рассмотренном случае $T_1<T_2$. Но оба времени соизмеримы и относятся к одному алгоритму сложности.

Альтернативой и качественно отличным алгоритму запроса к базе данных является алгоритм поиска в информационной массиве (рисунок 5).

Отличие алгоритма на рисунке 5 от алгоритма на рисунке 4 заключается в результате, условии и времени работы. Результат поиска является множественным. Выдается не одно значение как при запросе к базе данных, а множество значений Y_m. Условием работы алгоритма является $X \Leftrightarrow Y$, где символ

⇔ обозначает соразмерность [16; 17]. В этих случаях имеет место перебор массива. Но при запросе к базе данных перебор заканчивается при нахождении соответствия запросу. Теоретически возможна ситуация, когда на первом же запросе получается нужный результат. При информационном поиске всегда обрабатывается весь массив.

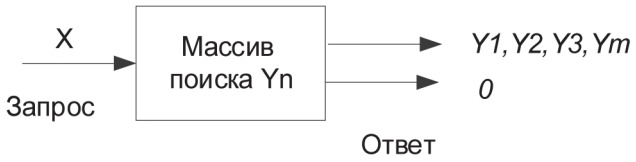

Рисунок 5 – Алгоритм поиска в информационном массиве

Дляосложности алгоритмов существуют классы и виды сложности. Для класса полиноминальной сложности видами являются все алгоритмы, которые можно описать полиномами *O(1), O(n), O(a(n)), O (n log(n)), O(n^a)* и др. Отметим некоторые алгоритмы: обратное время Аккермана *O(a(n))* – время на операцию с использованием непересекающихся множеств; логарифмическое время *O(log(n))* – бинарный поиск в массиве из *n* элементов; дробная мощность *O(n^c), где 0 <c <1* – поиск в *k-d* – дереве (англ. *k-d tree*, сокращение от *k-мерное дерево*).

Классы сложности группируют вычислительные задачи по требованиям к ресурсам. Для этого вычислительные задачи различаются по верхним границам максимального количества ресурсов, которое требуется наиболее эффективному алгоритму для их решения. В частности, классы сложности связаны со скоростью роста требований к ресурсам для решения вычислительной задачи по мере увеличения размера входных данных. Например, количество времени, необходимое для решения задач в классе сложности *P*, алгоритмы решения которых полиномиально зависят от размера входных данных, растет относительно медленно по мере увеличения размера входных данных, тогда как оно растет сравнительно быстро для задач в классе сложности экспоненциального времени EXPTIME (решаемых с помощью детерминированной машины Тьюринга за время *O(2$^{p(n)}$)*, где *p(n)* полиномиальная функция от *n*).

В теории сложности вычислений класс *P*, также известный как PTIME или DTIME(n$^{O(1)}$), является самым низким, содержит все задачи, которые могут быть решены детерминированной машиной Тьюринга за полиномиальное время. Многие алгоритмы первого рода имеют полиномиальную временную сложность (например, быстрая сортировка, сортировка вставкой, двоичный поиск).

Заключение

Многие классы сложности определяют с использованием концепции редукции. Редукция – это превращение одной проблемы в другую. Она отражает представление о том, что одна задача или алгоритм не менее сложна, чем другая задача. Например, если задача *X* может быть решена с помощью алгоритма для задачи *Z*, то следует, что *X* не сложнее, чем *Z*. Кроме редукции применяют сокращения. Есть много различных типов сравнения сложности, основанных на методах сокращений, таких как: сокращение Кука, сокращение Карпа, сокращение Левина. Метод сокращений включает оценку сложности сокращений. Хотя детерминированные и недетерминированные машины Тьюринга являются наиболее часто используемыми моделями вычислений, многие классы сложности определены в терминах других вычислительных моделей. Ряд классов определяется с помощью вероятностных машин Тьюринга, включая классы BPP, PP, RP и ZPP. Ряд классов определяется с помощью интерактивных систем доказательства, включая классы IP, MA и AM. Ряд классов определяется с использованием логических схем, включая классы P/poly и их подклассы NC и AC. Ряд классов определяется с помощью квантовых машин Тьюринга, включая классы BQP и QMA. Изучение отношений между классами сложности составляет предмет исследований в теории сложности вычислений. Делают регулярные попытки построения общих иерархий классов сложности. Например, известна попытка связать классы временной и пространственной сложности (выделены полужирным) следующим об-

разом: **L** ⊂ **NL** ⊂ P ⊂ NP ⊂ **PSPACE** ⊂ EXPTIME ⊂ **EXPSPACE**. Однако многие отношения еще не раскрыты явно, например, проблема отношений между классами P и NP (от англ. Non-deterministic polynomial), включающего множество задач, которые можно за время, не превосходящее полином от размера данных, решить на недетерминированной машине Тьюринга. Отношения между классами часто отвечают на вопросы о фундаментальной природе вычислений. Отношения между P и NP напрямую связаны с вопросами детерминированных и не детерминированных вычислений. Следует отметить условность деления на классы. Класс сложности не задает физическое время вычислений. Физическое время вычислений зависит от объема обрабатываемых данных. Может сложиться ситуация, при которой более простой вид сложности затрачивает больше времени на вычисления, чем более сложный. Недостатком существующей теории алгоритмической сложности является исключение из рассмотрения когнитивного фактора и фактора когнитивной сложности [18; 19]. Это является предметом дальнейших исследований. Существующая теория сложности акцентирует внимание на вычислениях и времени вычислений. Но главным в вычислениях является результат. Если результат вычислений не качественный или нечеткий, то время вычислений теряет свою важность. Соответственно оценка классов сложности должна привязываться не только ко времени, но и к качеству результата. В связи с этим целесообразно использовать параметр результативности вычислений при анализе классов сложности. Целесообразно при анализе сложности использовать понятие “информационные единицы”, придавая им смысл “информационные единицы вычислений”, “информационные единицы временных операций”. Алгоритмы первого рода – это алгоритмы относительно несложных классов. Это дает возможность более подробного их исследования, и данная статья является одним из этапов таких исследований.

Список литературы

1. *Tsvetkov V.Ya., Matchin V.T.* Information Conversion into Information Resources// European Journal of Technology and Design. – 2014. – № 2(4). – P. 92–104.
2. Математика. Большой энциклопедический словарь / Гл. ред. Ю.В. Прохоров. 3-е изд. – М.: Большая Российская энциклопедия, 2000. – 848 с.
3. *Моисеев Н.* Алгоритмы развития. – М.: Наука, 1987. – 304 с. Переиздана 2017, издательством Litres.
4. *Цветков В.Я.* Алгоритмы как средство познания // Информационные технологии. – 2018. – 8(24). – С. 507–515.
5. *Hippke M., Heller R.* Optimized transit detection algorithm to search for periodic transits of small planets // Astronomy & Astrophysics. – 2019. – Т. 623. – С. A39.
6. *Lynchenko A., Sheshkus A., Arlazarov V.L.* Document image recognition algorithm based on similarity metric robust to projective distortions for mobile devices //Eleventh International Conference on Machine Vision (ICMV 2018). – International Society for Optics and Photonics. – 2019. – Т. 11041. – С. 110411K.
7. *Xingyun Q., Shaobin C., Yanwei H.* An Algorithm for Identification of Inland River Shorelines based on Phase Correlation Algorithm //2019 Chinese Automation Congress (CAC). – IEEE, 2019. – С. 2047–2053.
8. *Аникина Г.А., Поляков М.Г., Романов Л.Н., Цветков В.Я.* О выделении контура изображения с помощью линейных обучаемых моделей // Известия академии наук СССР. Техническая кибернетика. – 1980. – № 6. – С. 36–43.
9. *Xiao Y.* et al. Non-Intrusive Load Identification Method Based on Improved KM Algorithm //IEEE Access. – 2019. – Т. 7. – С. 151368–151377.
10. *Zhang H., Su S.* A hybrid multi-agent Coordination Optimization Algorithm // Swarm and Evolutionary Computation. – 2019. – Т. 51. – С. 100603.
11. *Щенников А.Е.* Модели прямых алгоритмов // Славянский форум. – 2017. – 4(18). – С. 103–109.
12. *Иванников А.Д.* Проблема информационных языков и современное состояние информатики // Вестник МГТУ МИРЭА. – 2014. – № 4(5). – С. 39–62.
13. *Herold G., Kirshanova E., May A.* On the asymptotic complexity of solving LWE // Designs, Codes and Cryptography. – 2018. – Т. 86. – № 1. – С. 55–83.
14. *Sipser M.* Introduction To The Theory Of Computation, ser //Computer Science Series. Thomson Course Technology. – 2006.

15. *Цветков В.Я., Мордвинов В.А.* Подход к систематизации алгоритмов // Онтология проектирования. – 2018. – Т. 7. – № 4(26). – С. 388–397.
16. *Раев В.К.* Инфологические модели как инструмент исследования // Славянский форум. – 2020. – 3(29). – С. 56–66.
17. *Кудж С.А., Цветков В.Я.* Сравнительный анализ. – М.: МАКС Пресс, 2020. – 144 с.
18. *Кудж С.А., Цветков В.Я.* Факторы когнитивной сложности // ИТНОУ: Информационные технологии в науке, образовании и управлении. – 2018. – № 6 (10). – С. 34–41.
19. *Кудж С.А.* Оценка групповой когнитивной сложности // Славянский форум. – 2018. – 2(20). – С. 36–43.

References

1. *Tsvetkov V.Ya., Matchin V.T.* Information Conversion into Information Resources// European Journal of Technology and Design. – 2014. – № 2(4). – P. 92–104.
2. Matematika. Bol'shoj enciklopedicheskij slovar' /Gl. red. Yu.V. Prohorov. 3-e izd. – M.: Bol'shaya Rossijskaya enciklopediya, 2000. – 848 s.
3. *Moiseev N.* Algoritmy razvitiya. – M.: Nauka, 1987. – 304 s. Pereizdana 2017, izdatel'stvom Litres.
4. *Tsvetkov V.Ya.* Algoritmy kak sredstvo poznaniya // Informacionnye tekhnologii. – 2018. – 8(24). – S. 507–515.
5. *Hippke M., Heller R.* Optimized transit detection algorithm to search for periodic transits of small planets // Astronomy & Astrophysics. – 2019. – T. 623. – S. A39.
6. *Lynchenko A., Sheshkus A., Arlazarov V.L.* Document image recognition algorithm based on similarity metric robust to projective distortions for mobile devices // Eleventh International Conference on Machine Vision (ICMV 2018). – International Society for Optics and Photonics. – 2019. – T. 11041. – S. 110411K.
7. *Xingyun Q., Shaobin C., Yanwei H.* An Algorithm for Identification of Inland River Shorelines based on Phase Correlation Algorithm // 2019 Chinese Automation Congress (CAC). – IEEE, 2019. – S. 2047–2053.
8. *Anikina G.A., Polyakov M.G., Romanov L.N., Tsvetkov V.Ya.* O vydelenii kontura izobrazheniya s pomoshch'yu linejnyh obuchaemyh modelej // Izvestiya akademii nauk SSSR. Tekhnicheskaya kibernetika. – 1980. – № 6. – S. 36–43.
9. *Xiao Y.* et al. Non-Intrusive Load Identification Method Based on Improved KM Algorithm //IEEE Access. – 2019. – T. 7. – S. 151368–151377.
10. *Zhang H., Su S.* A hybrid multi-agent Coordination Optimization Algorithm //Swarm and Evolutionary Computation. – 2019. – T. 51. – S. 100603.
11. *Shchennikov A.E.* Modeli pryamyh algoritmov // Slavyanskij forum. – 2017. – 4(18). – S. 103–109.
12. *Ivannikov A.D.* Problema informacionnyh yazykov i sovremennoe sostoyanie informatiki // Vestnik MGTU MIREA. – 2014. – № 4(5). – S. 39–62.
13. *Herold G., Kirshanova E., May A.* On the asymptotic complexity of solving LWE // Designs, Codes and Cryptography. – 2018. – T. 86. – № 1. – S. 55–83.
14. *Sipser M.* Introduction To The Theory Of Computation, ser //Computer Science Series. Thomson Course Technology. – 2006.
15. *Tsvetkov, V.Ya., Mordvinov V.A.* Podhod k sistematizacii algoritmov // Ontologiya proektirovaniya. – 2018. – T. 7. – № 4(26). – S. 388–397.
16. *Raev V.K.* Infologicheskie modeli kak instrument issledovaniya // Slavyanskij forum. – 2020. – 3(29). – S. 56–66.
17. *Kudzh S.A., Tsvetkov V.Ya.* Sravnitel'nyj analiz. – M.: MAKS Press, 2020. – 144 s.
18. *Kudzh S.A., Tsvetkov V.Ya.* Faktory kognitivnoj slozhnosti // ITNOU: Informacionnye tekhnologii v nauke, obrazovanii i upravlenii. – 2018. – № 6 (10). – S. 34–41.
19. *Kudzh S.A.* Ocenka gruppovoj kognitivnoj slozhnosti // Slavyanskij forum. – 2018. – 2(20). – S. 36–43.