МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В.ЛОМОНОСОВА

На правах рукописи

Перпер Евгений Михайлович

Поиск подслова в множестве слов и его приложение к семантическому анализу текстов

01.01.09 — дискретная математика и
математическая кибернетика

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата физико-математических наук

Москва – 2018

Работа выполнена на кафедре математической теории интеллектуальных систем механико-математического факультета МГУ имени М.В.Ломоносова

Научный руководитель — *Гасанов Эльяр Эльдарович, доктор физико-математических наук, профессор*

Официальные оппоненты — *Колпаков Роман Максимович, доктор физико-математических наук, профессор кафедры дискретной математики механико-математического факультета МГУ имени М.В.Ломоносова*

Баранович Андрей Евгеньевич, доктор технических наук, доцент, профессор кафедры компьютерной безопасности и математических методов управления математического факультета Тверского государственного университета

Плетнев Александр Андреевич, кандидат физико-математических наук, ИП Плетнев Александр Андреевич

Защита диссертации состоится «6» апреля 2018 г. в 11 часов 30 минут на заседании диссертационного совета МГУ.01.16 Московского государственного университета имени М.В.Ломоносова по адресу: 119991, Москва, Ленинские горы, вл.1, Главное здание, механико-математический факультет, аудитория 1224.

E-mail: ilgova@cs.msu.su

С диссертацией можно ознакомиться в отделе диссертаций научной библиотеки МГУ имени М.В.Ломоносова (Ломоносовский просп., д.27) и на сайте ИАС «ИСТИНА»: http://istina.msu.ru/dissertations/99110840

Автореферат разослан «__» _____ 2018 г.

Ученый секретарь
диссертационного совета,
доктор физико-математических наук,
профессор Э.Э. Гасанов

Общая характеристика работы

Актуальность темы

В работе рассматривается задача поиска в базе данных, представляющей собой множество слов, всех слов, содержащих некоторое заданное слово в качестве подслова. Такая задача часто возникает на практике, например, когда необходимо найти слово в словаре. В частности, если рассматриваемое слово — имя существительное, находящееся не в именительном падеже, или глагол, находящийся не в инфинитиве, в словаре его может не оказаться. Тем не менее, в словаре найдутся слова, содержащие некоторое подслово рассматриваемого слова, например, его основу. Желание проводить поиск этих слов как можно быстрее приводит к задаче, которой посвящена работа. В диссертации рассматриваются две вариации данной задачи: собственно задача поиска подслова в множестве слов, в которой требуется найти все слова из множества, содержащие данное подслово, а также задача поиска вхождений подслова в множестве слов, требующая найти не только каждое слово из множества, содержащее данное подслово, но и позицию в этом слове, с которой данное подслово начинается.

Широко исследована вариация рассматриваемой задачи, в которой необходимо найти все вхождения некоторого слова («образца») в другое, более длинное слово («текст»). Алгоритмы, решающие задачу поиска вхождений образца в текст, можно условно разделить на 2 вида: осуществляющие предобработку текста и не осуществляющие ее. Характеристики алгоритмов этих двух видов существенно различаются.

Среди алгоритмов поиска, не выполняющих предобработку текста, можно упомянуть алгоритм Кнута – Морриса – Пратта[1] (1977), осуществляющий поиск за время, в худшем случае пропорциональное сумме длин текста и образца, а также алгоритм — Бойера – Мура[2] (1977). В худшем случае для решения задачи с помощью алгоритма Бойера – Мура потребуется время, пропорциональное произведению длины образца на разность длин текста и образца, к которой прибавили 1, однако в среднем этот алгоритм требует проверки числа символов, меньшего суммы длин текста и образца. Существует несколько известных модификаций алгоритма Бойера – Мура. Модификация, предложенная Галилем[3] в 1979 году, позволяет достичь времени работы, не превышающего (по порядку) длины образца. Модификация, предложенная Хорспулом[4] в 1980 году, упрощает алгоритм

[1]Knuth D.E., Morris J.H., Pratt V.B. Fast pattern matching in strings // SIAM J. Comput. — 1977. — Vol. 6, N 2. — P. 323—350.

[2]Boyer R.S., Moore J.S. A fast string searching algorithm // Commun ACM. — 1977. — Vol. 20, N 10. — P. 762—772.

[3]Galil Z. On improving the worst case running time of the Boyer-Moore string searching algorithm // Comm. ACM. — 1979. — Vol. 22, N 9. — P. 505—508.

[4]Horspool N. Practical fast searching in strings // Software Practice and Experience. — 1980. — Vol. 10. — P. 501—506.

Бойера – Мура и при этом увеличивает его среднюю скорость, но уменьшает его скорость в худшем случае; порядок времени работы алгоритма Бойера – Мура – Хорспула, однако, и в среднем, и в худшем случае такой же, как и у алгоритма Бойера – Мура.

Для данной работы, тем не менее, наибольший интерес представляют алгоритмы, осуществляющие предобработку текста. Эти алгоритмы удобно применять, когда происходит поиск разных образцов в одном и том же тексте (и поиск в словаре — именно этот случай). Время предобработки текста не учитывается при измерении времени поиска, так как предобработка осуществляется не при каждом поиске, а лишь один раз. В самых быстрых алгоритмах, осуществляющих предобработку текста, поиск происходит за время, в худшем случае пропорциональное сумме длины образца и количества вхождений образца в текст. К таким алгоритмам относятся суффиксные деревья и суффиксные массивы со словарями рангов.

П. Вейнер[5] предложил идею суффиксного дерева в 1973 году. Суффиксное дерево позволяло осуществлять поиск вхождений подслова в слово, используя $O(n \log n)$ битов памяти, где n — длина текста. Суффиксный массив, предложенный Манбером и Майерсом[6] в 1993 году, позволял осуществлять поиск вхождений подслова в слово и использовал на практике в 3-5 раз меньше памяти, чем суффиксное дерево. Манро, Раман и Рао[7] в 1998 году описали представление суффиксного дерева, требующее $n \log n + O(n)$ битов памяти. Суффиксный массив со словарями рангов (Мякинен, Наварро[8], 2007) использует $n \log n + kn + o(kn)$ битов памяти, где k — количество символов в алфавите.

Одним из наиболее значительных результатов является предложенный Гросси и Виттером[9] в 2000 году сжатый суффиксный массив: алгоритм на его основе использует $O(n \log k)$ битов памяти. Садакане[10] в 2003 году развил идею сжатого суффиксного массива: предложенный им алгоритм требует $n((1+\varepsilon')H_0(T)/\varepsilon+2\log(1+H_0(T))+3)+o(n)$ битов, где ε и ε' — любые константы такие, что $0 < \varepsilon < 1$ и $\varepsilon' > 0$. Функция $H_0(T)$ — это энтропия порядка 0 текста T, определенная как $-\sum_{i=1}^{k}(n_i/n)\log n_i/n$, где n_i — чис-

[5]Weiner P. Linear pattern matching algorithm // Proceedings of the 14th IEEE Annual Symposium on Switching and Automata Theory. — 1973. — P. 1—-11.

[6]Manber U., Myers G. Suffix Arrays: A New Method for On-Line String Searches // SIAM J. Comput. — 1993. — Vol. 22, N 5. — P. 935—-948.

[7]Munro J.I, Raman V., Rao S.S. Space Efficient Suffix Trees // Proceedings of the 18th Foundations of Software Technology and Theoretical Computer Science Conference, LNCS. — N. Y.; Berlin: Springer-Verlag, 1998. — P. 186—196.

[8]Navarro G., Mäkinen V. Compressed fulltext indexes // ACM Comput. Surv. — 2007. — Vol. 39, N 1. — Article 2.

[9]Grossi R., Vitter J.S. Compressed suffix arrays and suffix trees with applications to text indexing and string matching // Proc. of 32nd annual ACM symposium on Theory of computing, STOC'00. — 2000. — P. 397—406.

[10]Sadakane K. New text indexing functionalities of the compressed suffix arrays // Journal of Algorithms. — 2003. — Vol. 48, N 2. — P. 294—313.

ло вхождений буквы n_i в текст T. Алгоритм Садакане использует исходный текст при построении сжатого суффиксного массива, но не использует непосредственно при поиске. Ферраджина, Манцини, Мякинен и Наварро[11] в 2007 году улучшили результат Садакане для случаев $k = O(\text{polylog}(n))$, снизив объем используемой памяти до $nH_r(T)+o(n)$ битов, где $r \leq \alpha \log_k n$ для произвольной константы $\alpha, 0 < \alpha < 1$, а $H_r(T)$ — энтропия порядка r текста T, определяющаяся как $(1/n)\sum_{w\in\Sigma^r} |w_T|H_0(w_T)$, Σ — рассматриваемый алфавит, w_T — слово, получившееся в результате конкатенации символов, расположенных в тексте T сразу после вхождений подслова w. Надо заметить, что $H_r(T) \leq \log k$, а для некоторых текстов $H_r(T) = o(1)$.

В упомянутых работах Манро, Рамана и Рао[7]; Мякинена и Наварро[8]; Гросси и Виттера[9]; Садакане[10]; Ферраджины, Манцини, Мякинена и Наварро[11] используется модель вычислений word RAM[12]. В ней, в частности, предполагается, что каждая ячейка памяти состоит из w битов, причем арифметические и булевы операции над w-битными машинными словами осуществляются за константное время. Предполагается также, что длина текста меньше 2^w.

Не все алгоритмы, осуществляющие поиск вхождений образца в текст, можно обобщить на случай поиска вхождений образца в несколько текстов. Тем не менее, для суффиксного дерева это можно сделать: на его основе может быть построено обобщенное суффиксное дерево (см., например,[13]), позволяющее искать образец в массиве текстов за время, пропорциональное сумме длины образца и количества вхождений образца в массив текстов.

Интерес представляют не только сами алгоритмы поиска образца в тексте, но и нижние оценки памяти для алгоритмов, осуществляющие поиск за время, в худшем случае пропорциональное сумме длины образца и количества вхождений образца в текст. Демейн и Лопес-Ортис[14] в 2003 году показали, что такой алгоритм требует памяти, пропорциональной длине текста. В использованной ими модели предполагалось, что алгоритм имеет доступ к образцу, тексту и индексу — структуре данных, построенной для быстрого поиска образца в тексте. При вычислении времени поиска учитывались лишь запросы алгоритма к тексту, причем предполагалось, что целью каждого такого запроса было получение одного бита текста. Голынский[15] в 2009 году рассматривал модель, использующую ячейки па-

[11]Ferragina P., Manzini G., Mäkinen V., Navarro G. Compressed representations of sequences and full-text indexes // ACM Transactions on Algorithms. — 2007. — Vol. 3, N 2. — Article 20.

[12]Hagerup T. Sorting and Searching on the Word RAM // STACS 98, LNCS. — Berlin; Heidelberg: Springer, 1998. — Vol. 1373, P. 366—398.

[13]Gusfield D. Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology. — N. Y.: Cambridge University Press, 1997. — 556 p.

[14]Demaine E.D., Lopez-Ortiz A. A linear lower bound on index size for text retrieval // Journal of Algorithms - Special issue: Twelfth annual ACM-SIAM symposium on discrete algorithms. — 2003. — Vol. 48, N 1. — P. 2—15.

[15]Golynski A. Cell Probe Lower Bounds For Succinct Data Structures // Proc. of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms. — 2009. — P. 625—634.

мяти длиной w битов. Голынский показал, что для $w = \log n$ и образцов, имеющих длину $\log n / \log k$, алгоритм потребует $n \log k / \log n$ ячеек памяти. Тем не менее, общее число результатов, посвященных нижним оценкам объема памяти для задачи поиска подслова в слове, значительно меньше числа результатов, посвященных верхним оценкам объема памяти для той же задачи.

В настоящей работе используется информационно-графовая модель, предназначенная для формализации задач поиска в базах данных. В работах[16] [17] можно найти подробное описание модели и обоснование ее актуальности и полезности для построения и анализа алгоритмов поиска информации в базах данных. В данной модели алгоритм поиска описывается ориентированным графом. Вершины и ребра графа этого графа нагружены элементами данных, а также функциями, определенными на множестве возможных запросов к базе данных. Данная модель удобна для получения оценок (в том числе нижних) объема памяти, используемой при поиске, так как в этой модели объем памяти определяется числом ребер графа. При получении оценок предполагается, что при построении графа можно использовать только функции из некоторого заранее определенного множества. Для минимального по времени алгоритма поиска подслова в множестве слов получена нижняя оценка необходимой памяти порядка pn^2, где p — число слов в базе данных, а n — длина каждого из этих слов (считаем, что все слова в базе данных имеют одинаковую длину). Для случая, когда предполагается, что длина подслова подается на вход алгоритма вместе с самим подсловом, получена нижняя оценка сложности алгоритма, а также найдены верхние оценки сложности алгоритма при ограничениях на объем памяти. Для поиска вхождений подслова в множестве слов предложен алгоритм, фактически представляющий собой модификацию обобщенного суффиксного дерева. Он требует $O(pn)$ памяти. Показано, что время работы предложенного алгоритма превышает нижнюю оценку не более чем в три раза, а объем памяти, нужной для этого алгоритма, превышает нижнюю оценку не более чем в $2k + 11$ раз, где k — число букв в алфавите (k считается константой).

Далее в работе предложенные алгоритмы поиска подслова и вхождений подслова в множестве слов используются в процессе решения задачи автоматического выделения смысла некоторого текста на естественном языке. Эта задача требует многократного поиска слов в различных словарях, поэтому время работы алгоритмов ее решения существенно зависит от эффективности используемых алгоритмов поиска.

Как правило, для произвольного текста трудно формально определить, в чем состоит его смысл, но юридический документ описывает некоторые

[16]Гасанов Э.Э., Кудрявцев В.Б. Теория хранения и поиска информации. — М.: Физматлит, 2002. — 288 с.

[17]Кудрявцев В.Б., Гасанов Э.Э., Подколзин А.С. Основы теории интеллектуальных систем. — М.: МАКС Пресс, 2017. — 612 с.

алгоритмы действий в определенных ситуациях, и эти алгоритмы можно считать смыслом данного документа. В частности, закон, посвященный бухгалтерскому учету, описывает схемы вычисления значений объектов, упомянутых в законе. Можно считать, что смысл закона выделен правильно, если и только если вычисленные по этим схемам значения объектов совпадают со значениями, вычисленными в соответствии с текстом закона. Задача построения таких схем исследуется в данной работе. Предложен алгоритм, позволяющий строить такие схемы.

Первым этапом работы алгоритма является создание по каждому предложению текста его синтаксического дерева (дерева зависимостей) — дерева связей между словами в предложении. Определение дерева зависимостей см., например, на сайте Национального корпуса русского языка[18].

Большая часть парсеров — программ, осуществляющих синтаксический анализ — основывается либо на машинном обучении на синтаксически размеченных корпусах[19], либо на использовании строго определенных правил, позволяющих находить связи между словами[20].

Для программ первого типа необходима большая база синтаксически размеченных текстов (то есть, предложений, для которых синтаксический граф уже построен). Для русского языка можно выделить две базы синтаксически размеченных текстов — это Национальный корпус русского языка (доступ к этому корпусу ограничен) и проект Universal Dependencies[21]. Корпус русского языка проекта Universal Dependencies содержит более 66000 предложений, и его можно свободно использовать для обучения парсеров.

Что касается программ, работающих на основе правил, то они не требуют базы данных синтаксически размеченных текстов. Тем не менее, создание набора правил, который позволял бы проводить синтаксический анализ произвольных предложений — сложная задача. Одной из моделей языка, используемых программами, осуществляющими обработку естественного языка на основе правил, является модель «Смысл ⇔ Текст» (Мельчук[22]). Мельчук строит формальную модель языка, состоящую из нескольких уровней (в т.ч., семантического и синтаксического), и вводит наборы правил, с помощью которых должен совершаться переход обрабатываемого представления текста на другой уровень. В идеале эти наборы правил должны обеспечивать переход от текста к его смыслу и обратно.

У рассматриваемой задачи есть особенности, затрудняющие использо-

[18]Национальный корпус русского языка. Синтаксически размеченный корпус русского языка: информация для пользователей. — URL: http://www.ruscorpora.ru/instruction-syntax.html

[19]Chen D., Manning C.D. A Fast and Accurate Dependency Parser using Neural Networks // Proc. of EMNLP 2014. — 2014. — P. 740—750.

[20]Семантические словари в автоматической обработке текста (по материалам системы ДИАЛИНГ). — URL: http://www.aot.ru/docs/sokirko

[21]Universal Dependencies. — URL: http://universaldependencies.org

[22]Мельчук И. А. Опыт теории лингвистических моделей «Смысл ⇔ Текст». — М.: Школа «Языки русской культуры», 1999. — 368 с.

вание парсеров, имеющихся в свободном доступе. Первая из этих особенностей состоит в том, что достаточно одной неправильно определенной связи между словами в каком-либо предложении, чтобы формула, вычисляющая значение некоторого объекта, оказалась неверной. Другая особенность — наличие в тексте большого количества длинных сложных предложений, на которых парсеры делают значительно больше ошибок, чем на простых предложениях. Таким образом, возникает необходимость в надстройке, которая исправляла бы ошибки парсера. Число ошибок, совершаемых парсерами при разборе текста положения по бухгалтерскому учету ПБУ 6/01[23], оказалось достаточно велико, чтобы надстройка представляла собой практически полноценный парсер. В результате автором работы было принято решение отказаться от использования парсеров, имеющихся в свободном доступе, и создать собственный парсер. Он построен на основе правил, так как такой парсер значительно проще контролировать, чем парсер, созданный с помощью машинного обучения. При создании парсера учитывался тот факт, что в работе рассматриваются не произвольные тексты, а тексты нормативно-правовых актов, касающихся бухгалтерского учета, что значительно упростило создание необходимого набора правил.

После синтаксического разбора по каждому синтаксическому графу строится формула логики предикатов, которая определяет условия, накладываемые на рассматриваемые в тексте объекты. Наконец, по всем формулам вместе создается модель закона, представляющая собой совокупность схем вычисления значений объектов, упомянутых в законе. Каждая такая схема является своеобразным аналогом алгебраического дерева вычислений[24]. Построение формул логики предикатов и создание модели закона осуществляются, как и синтаксический анализ, на основе строго определенных правил.

Работоспособность предложенного алгоритма проверена на примере положения по бухгалтерскому учету ПБУ 6/01.

Цель работы

Целями работы являются:

- получение нижних оценок объема памяти алгоритмов поиска в базах данных в зависимости от времени поиска информации при некоторых ограничениях на используемые при поиске функции;

- построение алгоритма поиска слов и вхождений слов в множестве слов с характеристиками по объему и памяти, оптимальными при некоторых ограничениях на используемые при поиске функции;

[23]Приказ Министерства финансов Российской Федерации от 30.03.2001 N 26н «Об утверждении Положения по бухгалтерскому учету «Учет основных средств» ПБУ 6/01». — URL: http://base.consultant.ru/cons/cgi/online.cgi?req=doc;base=LAW;n=111056

[24]Ben-Or M. Lower Bounds For Algebraic Computation Trees // Proc. 15th ACM Annu. Symp. Theory Comput. — 1983. — P. 80—86.

- получение оценок сложности алгоритма поиска слов для случая, когда длина подслова подается на вход алгоритма вместе с самим подсловом, при ограничениях на объем памяти и используемые при поиске функции;

- создание алгоритма перевода текстов нормативных документов на русском языке в формальный язык логики предикатов;

- проверка работоспособности данного алгоритма на примере положения по бухгалтерскому учету ПБУ 6/01.

Методы исследования

В работе используются методы дискретной математики, теории сложности управляющих систем, математической логики, математической лингвистики.

Научная новизна

Для задачи поиска подслова в множестве слов найдено значение минимального времени поиска при некоторых ограничениях на используемые при поиске функции и получена нижняя оценка объема памяти, необходимого для алгоритмов, осуществляющих поиск за минимальное время. Для задачи поиска вхождений подслова в множестве слов найдено значение минимального времени поиска при некоторых ограничениях на используемые при поиске функции и построен алгоритм, осуществляющий поиск за минимальное по порядку время и требующий минимальной по порядку памяти. Для задачи поиска вхождений подслова в множестве слов в случае, когда длина подслова подается на вход алгоритма вместе с самим подсловом, и при некоторых ограничениях на используемые при поиске функции, найдена нижняя оценка сложности поиска, а также получены верхние оценки сложности поиска при ограничениях на объем памяти.

В работе получен перечень правил, позволяющий по тексту нормативно-правового акта построить его синтаксический граф, а затем — формулы логики предикатов, позволяющие вычислять значения объектов, о которых идет речь в нормативно-правовом акте.

Теоретическая и практическая значимость

Алгоритмы поиска, описанные в работе, можно использовать для быстрого нахождения, например, всех слов в словаре, содержащих данный корень, с использованием небольшого объема памяти.

Результаты работы могут быть использованы в ERP-системах (Enterprise Resource Planning — Управление ресурсами предприятия), частью которых являются программы, заполняющие различные формы отчетности. Сейчас эти программы пишутся вручную. При каждом изменении в законах, определяющих правила вычисления значений, которыми нужно заполнять формы отчетности, приходится находить, как именно эти изменения затронули программу, и соответствующим образом программу изменять. В данной работе описан способ автоматического построения схемы вычисления значений, которыми заполняются формы отчетности. Это позволило создать компьютерную программу, вычисляющую значения объектов в соответствии с инструкциями, описанными в нормативном акте. Последнее позволяет избавиться от труда программистов.

Описанные в работе методы обработки текста, написанного на естественном языке, можно использовать при решении различных задач математической лингвистики.

Апробация результатов

Результаты диссертации докладывались на следующих семинарах и конференциях:

1. Семинар «Теория автоматов» под руководством академика, профессора, д.ф-м.н. В.Б. Кудрявцева (2013–2017 гг., неоднократно);

2. Семинар «Вопросы сложности алгоритмов поиска» под руководством проф., д.ф-м.н. Э.Э.Гасанова (2010–2017 гг., неоднократно);

3. XI Международная конференция «Интеллектуальные системы и компьютерные науки» (2016, Москва, МГУ).

4. XII Международный семинар «Дискретная математика и ее приложения» (2016 г., Москва, МГУ);

5. Международная конференция студентов, аспирантов и молодых ученых «Ломоносов» (2013 г., 2014 г., 2015 г., Москва, МГУ);

6. Научная конференция «Ломоносовские чтения» (2013 г., 2014 г., 2016 г., Москва, МГУ);

7. XI Международный семинар «Дискретная математика и ее приложения», посвященный 80-летию со дня рождения академика О.Б. Лупанова (2012, Москва, МГУ);

8. X Международная конференция «Интеллектуальные системы и компьютерные науки» (2011, Москва, МГУ).

Публикации

Основное содержание диссертации опубликовано в 8 печатных работах, 5 из которых опубликованы в рецензируемых научных изданиях, определенных п.2.3 Положения о присуждении ученых степеней в Московском государственном университете имени М.В.Ломоносова. В работе, написанной в соавторстве, результаты автора диссертации являются определяющими. Список печатных работ приведен в конце автореферата.

Структура и объем работы

Диссертация состоит из введения, четырех глав, двух приложений и заключения. Объем диссертации — 114 страниц. Список литературы содержит 38 наименований.

Содержание работы

В главе 1 рассматривается задача поиска подслова в множестве слов, притом предполагается, что при поиске можно использовать лишь функции из определенного множества. Приводится нижняя оценка времени работы алгоритма, осуществляющего такой поиск. Также приводится нижняя оценка объема памяти, требуемого для поиска за наименьшее время, и описан алгоритм, производящий поиск за минимально возможное время и требующий минимальной по порядку памяти.

Обозначим через N_k множество, содержащее все натуральные числа от 1 до k. Будем считать, что слово длины s над алфавитом N_k представлено набором длины n, первые s элементов которого — из множества N_k, а оставшиеся элементы — нули. Длину слова w обозначим через $l(w)$.

Множество всех запросов (слов длины не более n над алфавитом N_k) обозначим через X. Назовем библиотекой множество, содержащее некоторые слова длины n над алфавитом N_k. Введем бинарное отношение ρ_1: для некоторых $x \in X$ и $y \in N_k^n$ тогда и только тогда выполнено $x\rho_1 y$, когда x — подслово y. Будем рассматривать задачу информационного поиска $I_1 =< X, V, \rho_1 >$, состоящую в перечислении для произвольного запроса $x \in X$ всех слов из некоторой библиотеки V, содержащих x в качестве подслова. Назовем эту задачу задачей поиска подслова.

При решении задачи поиска подслова в работе используется информационно-графовая модель данных[16] [17]. В данной модели база данных представлена в виде ориентированного графа. Некоторым вершинам графа сопоставлено одно слово из библиотеки. Кроме того, ребрам и вершинам графа сопоставлены функции из некоторого базового множества. Единственным аргументом каждой функции из базового множества является запрос. Для информационного графа задано функционирование: в

результате вычисления функции, сопоставленной вершине или ребру, запрос может перейти по ребру из вершины в другую вершину; изначально запрос находится в корне графа. Если запрос проходит в вершину, которой сопоставлено слово из библиотеки, то это слово попадает в ответ информационного графа на запрос. Если множество слов, оказавшихся в ответе информационного графа на запрос, совпадает с множеством слов из библиотеки, содержащих запрос в качестве подслова, то будем говорить, что информационный граф решает задачу поиска подслова.

Базовое множество $\mathcal{F}_1$, используемое при решении задачи поиска подслова, состоит из функции, тождественно равной 1, которая сопоставляется ребрам для безусловного перехода по ним, а также набора функций $g_i(x)$, сопоставляемых вершинам. Значение функции $g_i(x)$ равно i-й букве запроса x, а если длина запроса меньше i, то $k+1$. Обозначим через $\mathcal{U}(I_1, \mathcal{F}_1)$ множество всех информационных графов над базовым множеством $\mathcal{F}_1$, решающих задачу поиска подслова.

Объемом $Q(U)$ информационного графа U назовем число ребер в графе U. Эта величина соответствует объему памяти, используемой информационным графом. Еще одна характеристика информационного графа U — его сложность $T(U, x)$ на некотором запросе x — характеризует время, в течение которого граф обрабатывает запрос. Будем считать, что сложность вычисления функции $g_i(x)$ равна 1, а сложность вычисления функции, тождественно равной 1, равна некоторому числу t, причем $0 < t < 1$.

Обозначим через $d(I_1, x)$ количество слов из библиотеки, содержащих x как подслово. Величину $\min(l(x)+1, n) + t \cdot (d(I_1, x) - 1)$ будем обозначать как $R(I_1, \mathcal{F}_1, x)$.

Теорема 1. *При $k \geq 3$ среди всех библиотек $V \subseteq N_k^n$ из p слов, где $p \leq (k-1)^n$, найдется такая библиотека V, что для задачи I_1 для любых $U \in \mathcal{U}(I_1, \mathcal{F}_1)$ и $x \in X$ выполняется следующее условие: если $d(I_1, x) \geq 1$, то $T(U, x) \geq R(I_1, \mathcal{F}_1, x)$.*

Обозначим через $\mathcal{U}_{I_1}$ множество таких графов над базовым множеством $\mathcal{F}_1$, решающих задачу поиска подслова, что сложность каждого из них на любом запросе не превышает $R(I_1, \mathcal{F}_1, x)$.

Теорема 2. *Для любой задачи $I_1 =< X, V, \rho_1 >$ множество $\mathcal{U}_{I_1}$ непусто.*

Для каждой библиотеки из p слов выберем информационный граф наименьшего объема над базовым множеством $\mathcal{F}_1$, решающий задачу поиска подслова для этой библиотеки, а затем среди всех полученных графов выберем граф наибольшего объема; его объем обозначим через $Q_1(p, n)$.

Скажем, что функция $f(p, n) : \mathbb{N} \times \mathbb{N} \to \mathbb{R}_{>0}$ асимптотически не превосходит функцию $g(p, n) : \mathbb{N} \times \mathbb{N} \to \mathbb{R}_{>0}$ при $p \to \infty$, $n \to \infty$ и обозначим $f(p, n) \lesssim g(p, n)$, если существует такие функция $\alpha(p, n) : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$, $\alpha(p, n) \to 0$ при $n \to \infty,\ p \to \infty$ и числа $p_0 \in \mathbb{N}$, $n_0 \in \mathbb{N}$, что $f(p, n) \leq (1 + \alpha(p, n)) \cdot g(p, n)$, когда $p > p_0,\ n > n_0$.

Теорема 3. *Если* $k \geq 3$, $p \leq (k-1)^{[n/3]}$, *то* $pn^2/9 \lesssim Q_1(p,n) \lesssim (k+2)pn^2/2$ *при* $n \to \infty$, $p \to \infty$.

В главе 2 рассматривается задача поиска вхождений подслова в множестве слов при некоторых ограничениях на множество используемых при поиске функций. Приводится нижняя оценка времени работы алгоритма, осуществляющего такой поиск. Также приведена нижняя оценка объема памяти, требуемого для поиска за наименьшее по порядку время, и указан алгоритм, производящий поиск за минимально возможное по порядку время и требующий минимальной по порядку памяти.

Вхождением в слово будем называть пару, состоящую из слова и позиции в этом слове. Введем бинарное отношение ρ_2: для запроса $x \in X$ и вхождения $(y,i), y \in N_k^n, i \in N_n$ тогда и только тогда выполнено $x\rho_2(y,i)$, когда x — подслово y, начинающееся с i-й позиции слова y. Будем рассматривать задачу информационного поиска $I_2 =< X, V, \rho_2 >$, состоящую в нахождении в словах из библиотеки V каждой позиции, с которой начинается подслово, совпадающее с запросом. Назовем эту задачу задачей поиска вхождений подслова.

Понятие информационного графа вводится так же, как и для главы 1, с той разницей, что там, где для главы 1 рассматривалось множество слов из библиотеки, в настоящей главе будет рассматриваться множество вхождений в слова из библиотеки.

В качестве базового множества функций при решении задачи поиска вхождений подслова будем рассматривать множество $\mathcal{F}_2$, состоящее из всех функций множества $\mathcal{F}_1$, используемого в главе 1, а также множества функций $h_{v,i}(x)$. Функция $h_{v,i}(x)$ принимает значение 1, если подслово запроса x, начинающееся с его i-й позиции, является собственным префиксом слова v, либо длина запроса меньше i; функция принимает значение 2, если v — префикс этого подслова запроса; во всех остальных случаях значение функции равно 3. Сложность вычисления этой функции пропорциональна количеству сравнений букв слова v с буквами запроса при вычислении ее значения. Множество всех информационных графов над базовым множеством $\mathcal{F}_2$, решающих задачу поиска вхождений подслова, обозначим через $\mathcal{U}(I_2, \mathcal{F}_2)$.

Обозначим через $J_{I_2}(x)$ множество всех вхождений в слова из библиотеки, удовлетворяющих запросу x в задаче поиска вхождений подслова. Величину $\min(l(x)+1, n)+t\cdot(|J_{I_2}(x)|-1)$ будем обозначать как $R(I_2, \mathcal{F}_2, x)$. В теореме 4 показано, что если информационный граф решает задачу поиска вхождений подслова, то сложность обработки им запроса x, являющегося подсловом хотя бы одного слова из библиотеки, не может быть ниже величины $R(I_2, \mathcal{F}_2, x)$.

Теорема 4. *При* $k \geq 2$ *для каждой задачи* $I_2 =< X, V, \rho_2 >$, *для любых* $U \in \mathcal{U}(I_2, \mathcal{F}_2)$ *и* $x \in X$ *выполняется следующее условие: если* $|J_{I_2}(x)| \geq 1$, *то* $T(U,x) \geq R(I_2, \mathcal{F}_2, x)$.

Для каждой библиотеки из p слов выберем информационный граф наименьшего объема над базовым множеством $\mathcal{F}_2$, решающий задачу поиска вхождений подслова для этой библиотеки, а затем среди всех полученных графов выберем граф наибольшего объема; его объем обозначим через $Q_2(p,n)$.

Далее, для каждой библиотеки из p слов среди информационных графов над базовым множеством $\mathcal{F}_2$, решающих задачу поиска вхождений подслова для этой библиотеки, выберем графы, чья сложность на любом запросе x не превышает $3R(I_2,\mathcal{F}_2,x)$, а среди них — граф наименьшего объема. Затем среди всех полученных графов выберем граф наибольшего объема; его объем обозначим через $Q_{fast}(p,n)$.

.

Теорема 5. *Для любых натуральных $n \geq 2$, $k \geq 3$ и $p < (k-1)^{n-1}$ выполнено $p(2n-]log_{k-1}p[) - 1 \leq Q_2(p,n) \leq Q_{fast}(p,n) \leq (2k+11)pn$.*

В главе 3, как и в главе 1, рассматривается задача поиска подслова в множестве слов при условии, что при поиске можно использовать лишь функции из определенного множества, но, в отличие от главы 1, предполагается, что слово длины s над алфавитом N_k представлено набором длины s, а длина слова подается алгоритму, осуществляющему поиск, вместе с самим словом. Приведены нижняя оценка времени работы алгоритма, осуществляющего такой поиск, и верхние оценки времени работы алгоритма при различных ограничениях на объем памяти алгоритма.

Множество всех запросов (слов длины не более n над алфавитом N_k, представленных набором, чья длина совпадает с длиной слова) обозначим через X'. Введем бинарное отношение ρ_3: для некоторых $x \in X'$ и $y \in N_k^n$ тогда и только тогда выполнено $x\rho_3 y$, когда x — подслово y. Будем рассматривать задачу информационного поиска $I_3 =< X', V, \rho_3 >$, состоящую в перечислении для произвольного запроса $x \in X'$ всех слов из некоторой библиотеки V, содержащих x в качестве подслова, при условии, что сложность вычисления длины запроса равна 1. Назовем эту задачу задачей поиска подслова с известной длиной.

Понятие информационного графа вводится так же, как и в главе 1. Для построения информационных графов, решающих задачу поиска подслова с известной длиной, будем использовать функции из базового множества $\mathcal{F}_3$. Это множество состоит из функции $l(x)$, множества функций $\mathcal{F}_1$, описанного в главе 1, а также множества функций $g_i'(x)$, принимающих значение 1, если длина слова x больше i, и 2, если длина x меньше либо равна i. Сложность функции $l(x)$ и каждой функции $g_i'(x)$ полагается равной 1.

Сложностью информационного графа будем называть максимальную по всем запросам сложность обработки запроса этим графом.

Для некоторого числа q среди всех информационных графов объема не более q над базовым множеством $\mathcal{F}_3$, решающих задачу поиска вхождений подслова с известной длиной, выберем граф, чья сложность минимальна, и

обозначим его сложность через $T(I_3, \mathcal{F}_3, q)$. Назовем величину $T(I_3, \mathcal{F}_3, q)$ сложностью задачи I_3 при базовом множестве $\mathcal{F}_3$ и максимальном объеме q.

Обозначим через $d(I_3)$ наибольшее по всем запросам количество слов из библиотеки, удовлетворяющих запросу.

Теорема 6. *Пусть I_3 — задача поиска подслова с известной длиной, ε — произвольное положительное число, тогда для сложности $T(I_3, \mathcal{F}_3, q)$ задачи I_3 при базовом множестве $\mathcal{F}_3$ и максимальном объеме q справедливы оценки:*

$$n + t \cdot (d(I_3) - 1) \le T(I_3, \mathcal{F}_3, q);$$

$$T(I_3, \mathcal{F}_3, q) \le 2n - 1 + t \cdot (d(I_3) - 1), \text{ если } q \ge (1 + \varepsilon)pn^2(k + 3)/2;$$

$$T(I_3, \mathcal{F}_3, q) \le n + 1 + t \cdot (d(I_3) - 1), \text{ если } q \ge (1 + \varepsilon)pn^3k/6$$

при $n \to \infty$, $p \to \infty$.

Глава 4 посвящена построению модели юридического документа (закона, нормативно-правового акта) по тексту этого документа. Модель документа — это формальное представление совокупности схем вычисления значений объектов в соответствии с текстом документа. Построение модели документа осуществляется поэтапно.

Первый этап заключается в синтаксическом анализе текста, позволяющем по тексту каждого предложения получить его синтаксический граф. Вершинами этого графа являются слова, а ребрами — синтаксические отношения между словами. Граф, получившийся в результате синтаксического анализа, должен быть связным.

В главе описан алгоритм построения синтаксического графа предложения на основе информации, полученной в результате морфологического анализа этого предложения. Предлагается использовать (возможно, в несколько упрощенном виде) подход, применяемый А.С. Подколзиным для автоматического решения математических задач[25]. В применении к синтаксическому анализу этот подход состоит в следующем. Рассматривается список правил (также называемых приемами), которые позволяют находить синтаксические связи между словами. Для каждого слова (вместе с его морфологическими характеристиками) и каждого правила проверяется, применимо ли это правило к данному слову; если да, то создается продиктованное этим правилом синтаксическое отношение. Аналогичный подход используется и на следующих этапах построения модели документа.

На втором этапе происходит отождествление сущностей из различных предложений. В результате одну и ту же сущность во всех построенных на следующем этапе формулах будет выражать одна и та же переменная.

[25]Подколзин А.С. Самообучение интеллектуальной системы // Интеллектуальные системы. Теория и приложения (ранее: Интеллектуальные системы по 2014, вып. 2, ISSN 2075-9460). — 2014. — Т. 18, вып. 2. — С. 197—266

На третьем этапе по каждому предложению исходного текста и соответствующему синтаксическому графу строится формула логики предикатов. В результате связи между сущностями, о которых идет речь в предложении, оказываются выраженными с помощью математических операций.

На четвертом этапе по совокупности всех формул для каждого объекта строится модель вычисления этого объекта. Каждая такая модель фактически является схемой вычисления значения объекта в соответствии с необходимыми и достаточными для этого данными. Совокупность всех таких моделей и будет представлять собой модель закона.

Для каждого этапа построения модели документа, кроме синтаксического анализа, в главе 4 приведены приемы, позволяющие осуществить этот этап. Приемы синтаксического анализа вынесены в приложение 1. Приемов, описанных в главе 4 и приложении 1, достаточно для построения модели положения ПБУ 6/01. Приложение 2 содержит пример синтаксического анализа предложения с помощью приемов из приложения 1.

Заключение

В диссертации получены следующие основные результаты.

Получено значение минимального времени поиска подслова в базе данных, представляющей собой множество слов, при определенных ограничениях на используемые при поиске функции. Именно, используются только функции, каждая из которых для некоторого числа i возвращает i-ю букву запроса к базе данных, и функция, тождественно равная 1. Для алгоритмов, осуществляющих поиск за минимальное время, найдена нижняя оценка памяти. Кроме того, описан алгоритм, производящий поиск за минимальное время и требующий памяти, минимальной по порядку для алгоритмов, осуществляющих поиск за наименьшее время.

Найдено значение минимального времени поиска вхождений подслова в базе данных, представляющей собой множество слов, при некоторых ограничениях на множество используемых функций. Ограничения состоят в следующем: разрешено использовать функцию, тождественно равную 1, а также функции, каждая из которых для некоторого числа i возвращает i-ю букву запроса к базе данных. Кроме перечисленных, разрешены только функции, которые для некоторого слова и некоторого числа i проверяют, содержится ли префикс этого слова в запросе к базе данных, начиная с i-й буквы запроса, причем предполагается, что время вычисления такой функции пропорционально количеству произведенных при ее вычислении операций сравнения букв. Построен алгоритм, проводящий поиск за минимальное по порядку время и требующий минимальной по порядку памяти.

Для случая, когда длина подслова подается на вход алгоритма вместе с самим подсловом, получено значение минимального времени поиска при определенных ограничениях на используемые при поиске функции. Именно, разрешено использовать функцию, тождественно равную 1, и функции,

для некоторого числа i возвращающие i-ю букву запроса к базе данных; кроме того, можно использовать функцию, возвращающую длину запроса, и функции, проверяющие для некоторого числа, не превышает ли длина запроса это число. Найдены верхние оценки сложности поиска при ограничениях на объем используемой памяти.

Описан метод, позволяющий по тексту нормативно-правового акта, касающегося бухгалтерского учета, автоматически строить схемы вычисления значений объектов, упомянутых в данном нормативно-правовом акте. В частности, описаны приемы синтаксического анализа; упрощения синтаксического графа; отождествления сущностей; представления упрощенного синтаксического графа в виде формулы; построения по всем формулам схем вычисления значений объектов. С помощью данного метода созданы схемы вычисления значений объектов, описанных в положении по бухгалтерскому учету ПБУ 6/01.

В качестве перспектив дальнейшей разработки темы диссертации можно предложить распространение метода автоматического построения схем, вычисляющих значения объектов, на более широкий класс нормативно-правовых актов.

Благодарности

Автор выражает искреннюю благодарность заведующему кафедрой д.ф.-м.н., профессору, академику В.Б. Кудрявцеву и всему коллективу кафедры за опыт и знания, полученные во время обучения. Особую признательность хочется высказать научному руководителю д.ф.-м.н., профессору Э.Э. Гасанову за постановку задачи и научное руководство.

Публикации по теме диссертации

Статьи в рецензируемых научных изданиях, определенных п.2.3 Положения о присуждении ученых степеней в Московском государственном университете имени М.В.Ломоносова

[1]. Перпер Е.М. О сложности поиска подстроки // Интеллектуальные системы. Теория и приложения (ранее: Интеллектуальные системы по 2014, вып. 2, ISSN 2075-9460). — 2012. — Т. 16, вып. 1-4. — С. 299-320.

[2] Перпер Е.М. Нижние оценки временной и объемной сложности задачи поиска подслова // Дискретная математика. — 2014. — Т. 26, вып. 2. — С. 58—70.

[3] Кудрявцев В.Б., Гасанов Э.Э., Перпер Е.М. Автоматическая генерация компьютерной программы, моделирующей нормативно-правовой акт // Интеллектуальные системы. Теория и приложения (ранее: Интеллектуальные системы по 2014, вып. 2, ISSN 2075-9460). — 2014. — Т. 18, вып. 2. — С. 133—156.

[4] Перпер Е.М. Порядок сложности задачи поиска в множестве слов вхождений подслова // Интеллектуальные системы. Теория и приложения (ранее: Интеллектуальные системы по 2014, вып. 2, ISSN 2075-9460). — 2015. — Т. 19, вып. 1. — С. 99—116.

[5] Перпер Е.М. О синтаксическом анализе юридических текстов // Интеллектуальные системы. Теория и приложения (ранее: Интеллектуальные системы по 2014, вып. 2, ISSN 2075-9460). — 2016. — Т. 20, вып. 2. — С. 31—49.

Статьи в сборниках трудов научных конференций

[6] Перпер Е.М. Нижние оценки временной и объемной сложности задачи поиска подстроки // Тезисы докладов XX Международной научной конференции студентов, аспирантов и молодых ученых «Ломоносов–2013». — М., 2013.

[7] Перпер Е.М. Автоматическое построение модели нормативно-правового акта по его тексту // Тезисы докладов XXI Международной научной конференции студентов, аспирантов и молодых ученых «Ломоносов–2014». — М., 2014.

[8] Перпер Е.М. О синтаксическом анализе нормативных актов // Материалы XII Международного семинара «Дискретная математика и ее приложения» имени академика О. Б. Лупанова (Москва, МГУ, 20–25 июня 2016). — 2016. — М.: Изд-во механико-математического факультета МГУ. — С. 344—346.