МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра информатики и программирования

МАТЕМАТИКА В ОЛИМПИАДНОМ ПРОГРАММИРОВАНИИ. МЕТОДЫ И АЛГОРИТМЫ РЕШЕНИЯ ЗАДАЧ НА СТРОКИ. СОЗДАНИЕ ЭЛЕКТРОННОГО КУРСА

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

студента 2 курса 273 группы
направления 02.04.03 «Математическое обеспечение и администрирование информационных систем»
факультета КНиИТ
Левшунова Михаила Александровича

Научный руководитель
доцент, к. ф.-м. н., доцент _____ А. Г. Федорова

Заведующий кафедрой
доцент, к. ф.-м. н. _____ М. В. Огнева

Саратов 2021

ВВЕДЕНИЕ

Уже больше 50 лет активно развивается олимпиадное программирование – соревнования между лучшими программистами, которые решают сложнейшие задачи за сильно ограниченное время. В наши дни олимпиадное программирование уже завоевало мир – лучшие вузы вкладывают огромные деньги в тренировку сильнейших команд, которые смогут защитить честь университета на международных соревнованиях. Наиболее известное такое соревнование – международная студенческая олимпиада по программированию (ICPC) [1]. Крупнейшие ИТ-компании готовы идти на все, чтобы заполучить себе в штат победителей этой или других подобных олимпиад. Более того, многие из них самостоятельно их проводят. Примерами таких соревнований служат Google Code Jam, Facebook Hacker Cup, VK Cup (ВКонтакте) и многие другие. Очевидно, что невозможно выиграть такое соревнование за один день. Нужны годы подготовки, знание сотен различных алгоритмов и тысячи решенных задач.

Школьные и университетские курсы не дают достаточных знаний математики, необходимых для понимания некоторых наиболее сложных алгоритмов. Они используют значительно более глубокие области математики, с которыми, зачастую, знакомы только выпускники чисто математических специальностей. В свою очередь, на специальностях, связанных с компьютерными науками, студенты обычно лишь вскользь знакомятся с такими важными разделами математики как теория чисел, комбинаторика, теория игр, геометрия и многие другие. Тем не менее, все эти разделы очень широко представлены в олимпиадном программировании и любой студент, желающий добиться успехов в соревнованиях, обязан изучить эти темы более глубоко.

В связи с этим, студентам приходится прибегать к дополнительным источникам информации, таким как специализированные университетские курсы для олимпиадников (которые, однако, есть далеко не во всех университетах), онлайн-курсы, а также специализированная литература. Кроме того, необходима практика решения задач, которую предоставляют специализированные ресурсы, созданные олимпиадниками для подготовки к соревнованиям.

В процессе обучения студенты сталкиваются с совершенно непохожими друг на друга задачами и способами их решений. Чаще всего, распределение областей математики, алгоритмы из которых используются при решении задач, достаточно неравномерное в рамках различных олимпиад и олимпиад различ-

ных лет. Поэтому может возникнуть проблема, когда студент уже на довольно неплохом уровне подготовлен к задачам из одной области и с очень большим трудом решает задачи другого вида. В связи с этим, появляется необходимость время от времени фокусироваться на определенной области, чтобы подтянуть знания в ней.

Решением данной проблемы видится создание специализированных курсов, сосредоточенных на изучении применения алгоритмов из определенной области математики. Это задача комплексная и данная работа является лишь первым ее этапом. В качестве рассматриваемой области для первого курса были выбраны строковые алгоритмы, так как именно эта область является одной из лидирующих при возникновении проблем, описанных выше.

Следовательно, целью данной работы является разработка курса, описывающего строковые алгоритмы, используемые в олимпиадном программировании, а также способы их применения на реальных олимпиадных задачах.

Задачи, которые должны быть выполнены в ходе работы:

1. Провести стартовую работу по подготовке к разработке курса.
2. Изучить алгоритмы хеширования, префиксной и суффиксной обработки строк.
3. Подготовить список тем олимпиадных задач, которые могут быть решены с использованием алгоритмов из данных разделов.
4. Подобрать, решить и разобрать задачи по этим темам.
5. Объединить все наработки в итоговый курс.

Магистерская работа состоит из введения, 3 разделов, заключения, списка использованных источников и 6 приложений. Общий объем работы – 119 страниц, из них 66 страниц – основное содержание, список использованных источников – 44 наименования.

Краткое содержание работы

Теоретические основы разрабатываемого курса

Первый раздел «Теоретические основы разрабатываемого курса» посвящен теоретической подготовке к созданию курса. В нем изучаются уже существующие способы подготовки к соревнованиям по олимпиадному программированию и рассматриваются плюсы и минусы этих способов. Также в этом разделе описывается методика решения задач, которая применяется как на соревнованиях, так и во время решения задач при подготовке курса. Ниже приведена краткая выжимка данного раздела.

Изучение существующих решений

Процесс решения большинства задач можно разбить на 3 основных этапа: построение корректной математической модели, которая отражает все особенности поставленной задачи; разработка алгоритма, позволяющего решить задачу в условиях выбранной модели; написание программного кода для разработанного алгоритма. Первые два этапа могут повторяться до тех пор, пока не будет найдена подходящая пара модель-алгоритм, позволяющая решить поставленную задачу. В данном случае собственно программирование чаще всего оказывается наиболее простой частью работы, а на первый план выходит математический аспект.

Следовательно, как при разработке собственного курса, так и при изучении уже существующих, будет предполагаться, что студенты не испытывают проблем с написанием программного кода, если понимают теоретический аспект того, что они пишут. Соответственно ресурсы, предлагающие изучение языков программирования, не не рассматриваются в магистерской работе.

Литература. Для того, чтобы придумать способ применить тот или иной алгоритм к конкретной задаче, необходимо знать и понимать, как этот алгоритм работает. Одним из наиболее привычных способов изучения алгоритмов является специализированная литература.

Веб-сайты. Тем не менее, литература – далеко не единственный источник информации, доступный человеку, занимающемуся олимпиадным программированием. Другим немаловажным способом обучения являются различные веб-сайты, предоставляющие информацию об алгоритмах или их применениях.

Практика решения задач Однозначно наиболее важным аспектом при

подготовке к соревнованиям по программированию является прорешивание множества различных задач. Это позволяет:

- Отработать навыки написания изученных алгоритмов, позволяя с каждым разом писать их все быстрее.
- Привыкнуть к принятым в олимпиадном программировании способам ввода-вывода, вместе с определенными особенностями для разных типов задач.
- Разобраться в классификации видов задач, научиться быстро определять, что требует та или иная задача.
- Изучить паттерны, используемые как при описании условий задач, так и для их решений.
- Настроиться на быстрое соотнесение условия задачи и набора алгоритмов и структур данных, которые могут использоваться при ее решении.
- Найти свои сильные и слабые стороны, чтобы понимать, решение каких задач стоит придумывать самому, а какие лучше отдать своим сокомандникам.
- Ну и самое главное, просто научиться быстро составлять математические модели и решать с их помощью различные виды задач.

При разработке курса в первую очередь необходимо определить методику решения задач, которую можно будет использовать в примерах. Данная методика приведена в тексте магистерской работы.

Описание методов и алгоритмов

Второй раздел «Описание методов и алгоритмов» посвящен изучению методов и алгоритмов, используемых в работе. Для каждого алгоритма сначала приводится его общее описание, а затем в подробностях описывается само построение алгоритма в виде, пригодном для использования в олимпиадном программировании. Ниже приведены небольшие части из каждого подраздела.

Хеширование

Одним из наиболее используемых приемов для решения олимпиадных строковых задач является хеширование. В этом разделе будут рассмотрены его базовые определения, использование в компьютерных науках, а также различные алгоритмы, используемые при решении олимпиадных задач вместе с обоснованием, в каком случае какой алгоритм более эффективен.

Основным определения хеширования является хеш-функция. Именно

преобразование, производимое ей, называется хешированием.

Хеш-функция – функция, с помощью которой можно перевести массив входных данных произвольной длины в выходную строку заданной длины. Это производится с использованием специального алгоритма. Такое преобразование называется хешированием. Входные данные называют входным массивом, «сообщением» или «ключом». Выходные данные, полученные в результате данного преобразования, называют «хешем», «хеш-суммой» или «хеш-кодом».

Префикс-функция

Дана строка $s[0 \dots n-1]$. Необходимо вычислить для нее префикс-функцию, которая является массивом чисел $\pi[0 \dots n-1]$, в котором через $\pi[i]$ обозначается наибольшая длина максимального собственного суффикса подстроки $s[0 \dots i]$, совпадающего с ее префиксом. В данном случае значение $\pi[0]$ равно нулю.

В виде формулы это выглядит следующим образом:

$$\pi[i] = \max_{k=0 \dots i} \{k \ : \ s[0 \dots k-1] = s[i-k+1 \dots i]\} \tag{1}$$

Используя определение, можно написать алгоритм, который будет работать за $O(n^3)$. Для олимпиадного программирования необходим более быстрый алгоритм, для его получения необходимо использовать дополнительные оптимизации, которые описаны в работе.

Z-функция

Дана строка s длины n. Назовем Z-функцией [2] от этой строки массив длины n, в котором i-ый элемент равен максимальному длине подстроки, начинающейся в позиции i, которая совпадает с префиксом строки s.

Иначе говоря, $z[i]$ – это наибольший общий префикс строки s и ее i-го суффикса.

Возьмем за правило, что первый элемент Z-функции, $z[0]$ равен нулю. Чаще всего его значение не определяется, но так будет удобнее для реализации, при этом не повлияет на результат.

Для получения эффективного алгоритма, необходимо находить значения $z[i]$ по очереди – от $i = 1$ до $n-1$. В то же время, при нахождении следующего значения $z[i]$, выгодно по максиму использовать ранее вычисленные. В работе описано получение данного алгоритма.

Дан алфавит из k символов и набор строк, суммарная длина которых рав-

на m. Построим на этом наборе структуру данных «бор» [3]. Данная структура позволяет решать множество различных олимпиадных задач.

Структура данных Бор

Бор определяется как дерево, корень которого расположен в вершине Root, а на каждом ребре стоит метка одной из букв алфавита. Все ребра, спускающиеся из одной вершины имеют различные метки.

Посмотрим на некоторый путь из корня в боре и запишем метки ребер этого пути друг за другом. Получим строку, соответствующую данному пути. Аналогично, для каждой вершины бора, ставим в соответствие строку, которая получается на пути до этой вершины от корня.

Если в вершине бора заканчивается какая-либо строка из данного набора, установим для этой вершины флаг leaf равный **true**. Для всех остальных вершин, данный флаг будет равен **false** соответственно. В магистерской работе описан процесс построения Бора.

Алгоритм Ахо-Корасик

Алгоритм Ахо-Корасик – алгоритм поиска подстроки, который был разработан в 1975 году [4] Альфредом Ахо и Маргарет Корасик. Данный алгоритм находит в строке множество подстрок из заданного набора.

Данный алгоритм широко применяется не только в олимпиадном программировании, но и в системном программном обеспечении. Например, он используется в утилите поиска grep, которая применяется в командной строке UNIX систем.

Алгоритм Ахо-Корасик использует структуру «бор», описанную в предыдущем подразделе. Он строит по нему автомат. Данный автомат можно использовать для решения различных олимпиадных задач.

Суффиксный массив

Дана строка $s[0 \dots n-1]$ длины n. i-ым суффиксом строки называется подстрока $s[i \dots n-1]$, $i = 0 \dots n-1$.

Тогда суффиксным массивом [5] строки s называется перестановка индексов суффиксов $p[0 \dots n-1]$, $p[i] \in [0; n-1]$, которая задает порядок суффиксов в порядке лексикографической сортировки. Иными словами, нужно выполнить сортировку всех суффиксов заданной строки.

Например, для строки $s = abaab$ суффиксный массив будет равен: $(2, 3, 0, 4, 1)$.

Суффиксный автомат

Суффиксный автомат [6] (или ориентированный ациклический граф слов) – структура данных, позволяющая решать множество строковых задач.

Например, с помощью суффиксного автомата можно искать все вхождения одной строки в другую, или подсчитывать количество различных подстрок данной строки – обе задачи он позволяет решать за линейное время.

На интуитивном уровне, суффиксный автомат можно понимать как сжатую информацию обо всех подстроках данной строки. Впечатляющим фактом является то, что суффиксный автомат содержит всю информацию в настолько сжатом виде, что для строки длины n он требует лишь $O(n)$ памяти. Более того, он может быть построен также за время $O(n)$ (если мы считаем размер алфавита k константой; в противном случае – за время $O(n \log k)$).

Суффиксным автоматом для данной строки s называется такой минимальный детерминированный конечный автомат [7], который принимает все суффиксы строки s.

Суффиксное дерево

Суффиксное дерево [8] – структура данных, с помощью которой можно крайне эффективно решать множество сложных задач на поиск среди неструктурированных массивов данных.

Суффиксное дерево для строки s – это минимальное по числу вершин дерево, каждое ребро которого помечено непустой подстрокой s таким образом, что каждый суффикс $s[i \dots n-1]$ может быть прочитан на пути из корня до какого-нибудь листа и, наоборот, каждая строка, прочитанная на пути из корня до какого-нибудь листа, является суффиксом s.

Бор для произвольного набора строк строится за O(суммы длин этих строк). Понятно, что суммарная длина всех суффиксов строки пропорциональна квадрату от длины этой строки. Это приводит к тому, что построение суффиксного дерева тривиальным алгоритмом работает за $O(N^2)$. В этом случае возникает вопрос, каким образом можно построить суффиксное дерево быстрее?

Для этого можно воспользоваться алгоритмом Укконена [9].

Применение алгоритмов в олимпиадных задачах

Третий раздел «Применение алгоритмов в олимпиадных задачах» посвящен применению алгоритмов, описанных в предыдущем разделе. В нем

описываются олимпиадные задачи, которые можно решать с помощью того или иного алгоритма, их решения, а также дополнительные методы, которые могут использоваться вместе с данными алгоритмами. Ниже приведены списки задач, разобранных в каждом подразделе.

Применение хеширования в олимпиадных задачах

Разобрав используемые алгоритмы, необходимо теперь также разобрать, как применять данные алгоритмы для решения задач. Разбирать методы будем в том же порядке, в котором давались алгоритмы.

Хеширование тем или иным способом можно использовать во многих видах задач. В магистерской работе приведены решения следующих задач:

1. Сравнение подстрок за $O(1)$.
2. Поиск подстроки в строке за $O(n + k)$, где n – длина строки, в которой производится поиск, а k – длина искомой подстроки.
3. Поиск наибольшей общей подстроки двух строк длин n и m $(n \geqslant m)$ за $O(n \cdot (\log n)^2)$.
4. Поиск лексикографически минимального циклического сдвига строки за $O(n \cdot \log n)$.
5. Сортировка всех циклических сдвигов строки в лексикографическом порядке за $O(n \cdot (\log n)^2)$.
6. Определение числа палиндромов в строке за $O(n \cdot \log n)$.
7. Вычисление количества подстрок строки длины n, которые являются циклическими сдвигами строки длины k за $O((n + k) \cdot \log(n))$.
8. Вычисление наибольшего общего префикса двух строк длины n при обмене двух произвольных символов одной строки местами за $O(n \cdot \log(n))$.

Применение алгоритмов префиксной обработки в олимпиадных задачах

Перейдем к применению алгоритмов префиксной обработки. Тем или иным способом эти алгоритмы можно использовать во многих видах задач. В магистерской работе приведено описание решений следующих задач:

1. Поиск подстроки в строке за $O(n + m)$.
2. Количество различных подстрок в строке за $O(n^2)$.
3. Подсчет числа вхождений каждого префикса за $O(n)$.
4. Поиск всех строк из заданного набора в тексте за $O(Len + Ans)$, где Len

– длина текста, Ans – размер ответа.

Применение суффиксного массива

Используя алгоритмы и методы построения суффиксных массивов, можно решить многие виды задач. В магистерской работе приведено описание решений следующих задач:

1. нахождение наименьшего циклического сдвига строки;
2. поиск подстроки в строке;
3. сравнение двух подстрок строки;

Применение суффиксного автомата

Используя алгоритмы и методы построения суффиксных автоматов, можно решить многие виды задач. В магистерской работе приведено описание решений следующих задач:

1. Проверка вхождения строки в текст.
2. Нахождение количества различных подстрок.
3. Нахождение суммарной длины различных подстрок.

Применение суффиксного дерева

Используя алгоритмы и методы построения суффиксных деревьев, можно решить многие виды задач. В магистерской работе приведено описание решений следующих задач:

1. Нахождение включения одной строки в текст.
2. Наибольшая общая подстрока двух или более строк.
3. Нахождение суммы значений Z-функции строки на подстроке.

ЗАКЛЮЧЕНИЕ

В результате работы был создан курс, описывающий строковые алгоритмы, используемые в олимпиадном программировании, а также способы их применения на реальных олимпиадных задачах. С использованием данного курса студенты могут готовиться к успешным выступления на соревнованиях.

Курс содержит описания алгоритмов хеширования, префиксной и суффиксной обработки строк, способы их реализации и применения в олимпиадных задачах.

Также в рамках курса был подготовлен набор задач на перечисленные темы. Задачи используются как в качестве примеров для объяснения тех или иных методов и алгоритмов, так и в качестве материалов для самостоятельного решения. Ко многим представленным задачам также были подготовлены разборы и предоставлены решения на случай, если у студентов будут возникать проблемы.

Часть задач из набора была подготовлена специально для курса, часть взята из реальных соревнований или других источников, специализирующихся на подготовке студентов к соревнованиям по олимпиадному программированию.

Итоговый курс выложен на учебном портале СГУ и доступен по ссылке: `http://start.sgu.ru/course/view.php?id=1837`

Отдельные части магистерской работы были представлены на конференции: Студенческая научная конференция «Компьютерные науки и информационные технологии» 2021 г. Тема выступления: «Математика в олимпиадном программировании. Применение методов и алгоритмов решения задач на строки»

СПИСОК ОСНОВНЫХ ИСТОЧНИКОВ

1 ICPC – International Collegiate Programming Contest [Электронный ресурс]. — URL: `https://icpc.baylor.edu/` (Дата обращения 10.05.2021). Загл. с экр. Яз. англ.

2 *Gusfield, D.* Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology / D. Gusfield. — Cambridge University Press, 1997.

3 *Briandais, R.D.L.* File searching using variable length keys / R.D.L. Briandais // *IRE-AIEE-ACM '59 (Western): Papers presented at the the March 3-5, 1959, western joint computer conference*. — 1959. — Pp. 295–298.

4 *Aho, A.V.* Efficient string matching: An aid to bibliographic search / A.V. Aho, M.J. Corasick // *Communications of the ACM*. — 1975. — Vol. 18. — Pp. 333–340.

5 *Manber, U.* Suffix arrays: a new method for on-line string searches / U. Manber, G. Myers // *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms*. — 1990. — Pp. 319–327.

6 *Crochemore, M.* On compact directed acyclic word graphs / M. Crochemore, R. Verin // *Structures in Logic and Computer Science. Lecture Notes in Computer Science*. — 1997. — Pp. 192–211.

7 *Hopcroft, J.E.* Introduction to Automata Theory, Languages, and Computation / J.E. Hopcroft, R. Motwani, J.D. Ullman. — Addison Wesley, 2001.

8 *Weiner, P.* Linear pattern matching algorithms / P. Weiner // *Proceedings of the 14th Annual Symposium on Switching and Automata Theory*. — 1973. — Pp. 1–11.

9 *Ukkonen, E.* On-line construction of suffix trees / E. Ukkonen // *Algorithmica*. — 1995. — Vol. 14. — Pp. 249–260.