УЧЕНЫЕ ЗАПИСКИ КАЗАНСКОГО УНИВЕРСИТЕТА.
СЕРИЯ ФИЗИКО-МАТЕМАТИЧЕСКИЕ НАУКИ

2020, Т. 162, кн. 3
С. 367–382

ISSN 2541-7746 (Print)
ISSN 2500-2198 (Online)

УДК 519.712.3

doi: 10.26907/2541-7746.2020.3.367-382

КВАНТОВЫЕ ОНЛАЙН-АЛГОРИТМЫ ДЛЯ МОДЕЛИ ИГРЫ ЗАПРОС-ОТВЕТ С БУФЕРОМ

К.Р. Хадиев[1,2], *Д.И. Лин*[2,3]

[1]*ООО «Квантовые интеллектуальные технологии», г. Казань, 420111, Россия*
[2]*Казанский (Приволжский) федеральный университет, г. Казань, 420008, Россия*
[3]*Компания АО «Барс Груп», г. Казань, 420012, Россия*

Аннотация

В статье онлайн-алгоритмы представляются в качестве игры «запрос-ответ». Это игра двух игроков: алгоритма и противника. Противник, у которого хранятся входные данные, отдает их по частям, затем делает запрос, а алгоритм отвечает на него, отправляя выходные данные. Мы рассматриваем обобщенную модель, в которую добавлен буфер ограниченного размера. Противник загружает данные в буфер, а алгоритм считывает данные из буфера в произвольном порядке. В работе рассматриваются квантовые и классические (детерминированные и вероятностные) алгоритмы в рамках данной модели.

Специально была сконструирована задача, для которой квантовый алгоритм работает эффективнее, чем любой классический. Эффективность алгоритмов в работе рассматривается с точки зрения конкурентного соотношения. Заметим, что при рассмотрении классических алгоритмов стандартная модель онлайн-алгоритмов эквивалентна расширенной модели с буфером.

Ключевые слова: квантовые вычисления, онлайн-алгоритмы, игра запрос-ответ, задача онлайн-минимизации, вычисления с буфером

Введение

Одно из применений онлайн-алгоритмов – это решение задач оптимизации [1]. В таком случае алгоритм выводит очередную порцию выходных данных сразу же после прочтения очередной порции входных данных. При этом выдаваемый результат должен минимизировать некоторую целевую функцию (функцию стоимости выходных данных). В качестве меры эффективности таких алгоритмов чаще всего используется *конкурентное соотношение* [2, 3].

Одна из возможных точек зрения на такие алгоритмы – это игра «запрос-ответ» [4]. Это игра двух игроков: онлайн-алгоритма и противника, у которого в полном распоряжении находится входной набор. Противник на своем раунде отправляет запрос в виде входной переменной, а онлайн-алгоритм возвращает в качестве ответа очередную выходную переменную. Рассмотрим развернутый вариант игры. Алгоритм просит у противника прислать ему входную переменную, но в качестве оплаты противник требует вернуть ему выходную переменную. Новая версия игры эквивалентна исходной, но ее можно обобщить. В настоящей работе предлагается новая модель, которая называется игра «запрос-ответ» с буфером. Это игра трех игроков: онлайн-алгоритма, противника и буфера ограниченного размера. Алгоритм может делать запросы двух видов:

– попросить противника загрузить очередной блок входных переменных в буфер;

– запросить у буфера одну из переменных, которая в нем сохранена.

Для некоторого целого числа R, выступающего параметром модели, противник запрашивает у алгоритма очередной блок выходных переменных каждые R раундов игры.

Если размер буфера равен 1 и $R = 1$, то модель в точности соответствует изначальной модели игры «запрос-ответ».

Мотивация. Онлайн-алгоритмы имеют различные применения. Одно из них – принятие решений в текущий момент в условиях отсутствия информации о данных, которые придут в будущем. Другое применение – обработка потока данных и вывод потока выходных данных в онлайн-формате, например, обработка потокового видео и другие. Многие языки программирования, такие как Java, C++ [5, 6] и др., используют потоки данных с буфером, которые сохраняют данные в быстрый буфер, а уже программа считывает данные из буфера. Предложенная модель схожа с потоками данных с буфером. Другими словами, рассматриваемая модель фокусируется на онлайн-поведении выходного потока данных в случае, когда алгоритм читает входные данные из потока, но при этом может пропускать часть входных данных.

Квантовая модель вычислений. В настоящей работе рассматривается квантовая версия модели игра «запрос-ответ» с буфером. Сами квантовые вычисления являются активно разрабатываемым направлением кибернетики [7–9]. Существует много задач, для которых квантовые алгоритмы демонстрируют превосходство перед лучшими классическими аналогами [11–15]. При этом исследователи рассматривали различные вычислительные модели, такие как модели запросов, модели обработки потока данных, коммуникационные модели и др. [16–26].

Различные версии квантовых онлайн-алгоритмов были рассмотрены в работах [21, 22], в том числе онлайн-алгоритмы, обрабатывающие потоки данных [27, 28], квантовые онлайн-алгоритмы с ограниченной памятью [29], квантовые онлайн-алгоритмы с возможностью повторно прочесть входной набор [30]. В этих работах авторы показали примеры задач, для которых квантовые онлайн-алгоритмы имели меньшее конкурентное соотношение в сравнении с классическими онлайн-алгоритмами.

Полученные результаты. В работе предлагается специально сконструированная задача и квантовый онлайн-алгоритм для модели игры «запрос-ответ» с буфером для ее решения. Покажем, что предлагаемый квантовый алгоритм эффективнее (имеет меньшее конкурентное соотношение), чем любой классический (вероятностный или детерминированный) аналог. Рассматриваемая задача называется «наиболее часто встречающаяся строка». При этом в задаче рассматриваются строки длины ровно k. Задача состоит в том, чтобы найти наиболее часто встречающуюся строку (из интересующих нас) и выдавать ее сразу же после каждой новой поступившей строки. Данная задача является разновидностью задачи поиска наиболее часто встречающегося элемента в последовательности [31], которая активно исследуется при изучении потоков данных [32–34]. Существует много приложений данной задачи для маршрутизации пакетов, ведения журнала телекоммуникаций и отслеживания запросов ключевых слов в поисковых машинах. Похожая задача для классических онлайн-алгоритмов рассматривалась в [35].

Статья организована следующим образом. Определения представлены в разд. 1. Описание задачи и квантового алгоритма для нее дано в разд. 2. Нижняя оценка сложности классических алгоритмов рассмотрена в разд. 3.

1. Определения

Задача онлайн-минимизации состоит из множества $\mathcal{I}$ допустимых входных наборов и функции стоимости. Каждый входной набор $I = (x_1, \ldots, x_n)$ – это набор

запросов, где n – размер входного набора $|I| = n$. Кроме того, с множеством I связано множество возможных выходных данных (решений) $\mathcal{O}(I)$; выходные данные – это последовательность ответов $O = (y_1, \ldots, y_n)$. Оптимальным решением для $I \in \mathcal{I}$ является $O_{\text{opt}}(I) = \underset{O \in \mathcal{O}(I)}{\operatorname{argmin}} \operatorname{cost}(I, O)$.

Дадим определение онлайн-алгоритма для такой задачи. **Детерминированный онлайн-алгоритм** A выдает выходную последовательность $A(I) = (y_1, \ldots, y_n)$ такую, что y_i вычисляется по $x_1, \ldots, x_i$. Скажем, что A является *c-конкурентным* в случае, если существует такая константа $\alpha \geq 0$ для каждого n и каждого выходного набора I размера n, что:

$$\operatorname{cost}(I, A(I)) \leq c \cdot \operatorname{cost}(I, O_{\text{opt}}(I)) + \alpha,$$

где c – это минимальное число, удовлетворяющее условию. Мы также говорим, что c – это конкурентное соотношение для алгоритма A. Если $\alpha = 0$ и $c = 1$, то алгоритм A называется оптимальным.

Вероятностный онлайн-алгоритм R вычисляет выходную последовательность $R^{\psi}(I) = (y_1, \ldots, y_n)$ такую, что y_i определяется по $\psi, x_1, \ldots, x_i$, где ψ – содержимое вероятностной ленты, то есть бесконечной бинарной последовательности, в которой значения всех битов выбираются равновероятно и независимо. Пусть $\operatorname{cost}(I, R^{\psi}(I))$ – это случайная величина, показывающая стоимость выходных данных, вычисленных R по набору I. R является c-конкурентным в ожидании, если существует такая константа $\alpha > 0$, что для любого I выполняется следующее неравенство:

$$\mathbb{E}\left[\operatorname{cost}(I, R^{\psi}(I))\right] \leq c \cdot \operatorname{cost}(I, O_{\text{opt}}(I)) + \alpha.$$

Говорят, что c – это математическое ожидание от конкурентного отношения алгоритма R.

1.1. Модель игры «запрос-ответ» с буфером. Стандартная модель онлайн-алгоритмов может быть рассмотрена как игра «запрос-ответ» [4].

Противник имеет доступ ко входным данным и отправляет запрос x_i алгоритму, а алгоритм выдает ответ y_i. В такой модели «активным» игроком, который управляет игрой, является противник, а алгоритм выступает в роли «пассивного», то есть только отвечающего на запросы.

Изменим точку зрения на эту игру. Рассмотрим модель, когда оба пользователя являются «активными» в некотором смысле.

> **Раунд** 1. Алгоритм запрашивает переменную x_1. (На этом раунде алгоритм является активным игроком.)
>
> **Раунд** 2. Противник запрашивает выходную переменную y_1. (На этом раунде противник является активным игроком.)
>
> ...
>
> **Раунд** $2i-1$. Алгоритм запрашивает переменную x_i. (На этом раунде алгоритм является активным игроком.)
>
> **Раунд** $2i$. Противник запрашивает выходную переменную y_i. (На этом раунде противник является активным игроком.)

Легко заменить, что новая модель эквивалентна исходной игре и стандартной модели.

В то же время новую модель можно обобщить. Рассмотрим третьего игрока, который будет выступать буфером или посредником между алгоритмом и противником. Пусть K – это положительное целое число, представляющее размер

буфера. Дополнительно рассмотрим целочисленный параметр $R \leq K$. Алгоритм делает запрос на загрузку в буфер блока данных из K переменных. К примеру, если i – это номер загружаемого блока, то на раундах, на которых алгоритм является активным, он может выполнить одно из двух следующих действий:

– алгоритм делает запрос на очистку буфера и загрузку в буфер следующего блока из K входных переменных, то есть переменных $x_{i\cdot K+1}, \ldots, x_{i\cdot K+K}$. После этого параметр i увеличивается на единицу $(i \leftarrow i+1)$;

– алгоритм запрашивает любую из переменных, которая загружена в буфер. В данном случае мы рассматриваем модель запросов для доступа к переменным буфера.

Таким образом, игра проходит по следующему сценарию:

Раунд 0. Инициализация переменной $i \leftarrow 0$.

Раунд 1. Алгоритм является активным игроком и выполняет одно из возможных действий, описанных раннее.

Раунд 2. Алгоритм является активным игроком и выполняет одно из возможных действий, описанных раннее.

...

Раунд R. Алгоритм является активным игроком и выполняет одно из возможных действий, описанных раннее.

Раунд $R+1$. Противник является активным игроком. Он запрашивает выходные переменные $y_1, \ldots, y_R$.

...

Раунд $(R+1)\cdot j+1$. Алгоритм является активным игроком и выполняет одно из возможных действий, описанных раннее.

Раунд $(R+1)\cdot j+2$. Алгоритм является активным игроком и выполняет одно из возможных действий, описанных раннее.

...

Раунд $(R+1)\cdot j+R$. Алгоритм является активным игроком и выполняет одно из возможных действий, описанных раннее.

Раунд $(R+1)\cdot j+R+1$. Противник является активным игроком. Он запрашивает выходные переменные $y_{j\cdot R+1}, \ldots, y_{j\cdot R+R}$.

Замечание. В случае, если $K=1$ и $R=1$, новая модель эквивалентна стандартной модели онлайн-алгоритмов.

В случае вероятностного алгоритма процесс запросов переменных из буфера является вероятностным и используется вероятностная модель запросов. Так же, как и для стандартной модели, в данном случае рассматривается математическое ожидание от конкретного отношения. В то же время запрос на загрузку следующего блока в буфер является детерминированной операцией. В случае квантового алгоритма процесс запросов переменных из буфера является квантовым и используется квантовая модель запросов. Из-за вероятностного поведения квантового алгоритма в данном случае также рассматривается математическое ожидание от конкурентного отношения. В то же время запрос на загрузку следующего блока в буфер является детерминированной операцией.

Так как основная часть рассмотренных алгоритмов классическая и только некоторые процедуры квантовые, то в настоящей работе детальное описание квантовой модели пропущено. Более подробную информацию о квантовой модели запросов можно получить из [7–9].

2. Квантовый алгоритм для задачи поиска наиболее часто встречающейся строки

Обсудим формальную постановку задачи.

Задача. Пусть m, d и k – некоторые положительные целые числа. Тогда входными данными является последовательность

$$I = (s^1, \ldots, s^d, x^1, \ldots, x^m).$$

Здесь $(s^1, \ldots, s^d)$ – это последовательность интересующих нас строк, $s^j = (s^j_1, \ldots, s^j_k) \in \{0,1\}^k$, где $j \in \{1, \ldots, d\}$. Строки $x^1, \ldots, x^m$ – это строки-запросы, где $x^j = (x^j_1, \ldots, x^j_k) \in \{0,1\}^k$ для $j \in \{1, \ldots, m\}$. При этом длина входного набора составляет $n = (m+d) \cdot k$. Частота строки $t \in \{0,1\}^k$ – это $f(t) = \dfrac{\#(t)}{m}$, где $\#(t) = |\{i : t = x^i, i \in \{1, \ldots, m\}\}|$ – это число появления t в наборе $(x^1, \ldots, x^m)$. Рассмотрим множество $F = \{i : x^i \in \{s^1, \ldots, s^d\}\}$. Индекс i_0 наиболее часто встречающейся строки x^{i_0} такой, что выполняется условие $f(x^{i_0}) = \max\limits_{i \in F} f(x^i)$, причем i_0 минимально из всех возможных. Необходимо выводить индекс i_0 после прочтения каждой из строк x^j. Верный ответ, который вычисляет оптимальный онлайн-алгоритм, – это последовательность $(z_1, \ldots, z_n)$, причем $z_{(j+d)\cdot k} = i_0$ для $j \in \{1, \ldots, m\}$, а остальные выходные переменные не важны.

Стоимость выходного набора $O = (y_1, \ldots, y_n)$ равна

$$\mathrm{cost}(I, O) = 1 + m - \sum_{j=1}^{m} \delta(y_{(j+d)\cdot l}, i_0)$$

Здесь $\delta(a, b) = 1$, если $a = b$, и $\delta(a, b) = 0$ в случае, если $a \neq b$

2.1. Квантовый алгоритм. Прежде всего обсудим квантовую процедуру, которая сравнивает две строки длины l для некоторого целого положительного l.

2.1.1. Квантовый алгоритм для сравнения двух строк. Рассмотрим реализацию процедуры, которая будет сравнивать строки $s = (s_1, \ldots, s_k)$ и $t = (t_1, \ldots, t_k)$ длины k в лексикографическом порядке. Функция будет возвращать

- -1, если $s < t$;
- 0, если $s = t$;
- 1, если $s > t$.

Обозначим данную квантовую процедуру за COMPARE_STRINGS(s, t, k). Алгоритм базируется на следующей идее. Для булевой функции находится минимальный аргумент, на котором функция принимает значение 1. Формально результат следующий.

Лемма 1 [10]. *Рассмотрим функцию $f : \{1, \ldots, N\} \to \{0,1\}$ для некоторого положительного целого N. Существует квантовый алгоритм, который находит $j_0 = \min\{j \in \{1, \ldots, N\} : f(j) = 1\}$. При этом алгоритм находит j_0 с запросной сложностью $O(\sqrt{N})$ и с вероятностью ошибки не более $1/2$.*

В качестве функции f выберем функцию $f(j) = (s_j \neq t_j)$. Таким образом будет найден j_0, являющийся наименьшим индексом несовпадающих символов двух строк. В этом случае можно утверждать, что s предшествует t в лексикографическом порядке тогда и только тогда, когда s_{j_0} предшествует t_{j_0} в алфавите. Если же неравных символов не найдено, то строки s и t равны. Будем считать, что если алгоритм нашел аргумент j' такой, что $f(j') = 0$, то он возвращает значение

$k+1$. Такое может произойти и в случае, если строки равны и у функции f нет аргументов, на которых она принимает значение 1; или же если алгоритм ошибся, то есть аргументы, на которых функция принимает значение 1, есть, а алгоритм его не нашел.

Для данного алгоритма применяется стандартная техника разгона вероятности успеха, которая используется, к примеру, в [13]. С этой целью выполнение алгоритма повторяется $3\log_2 m$ раз и возвращается минимальное из полученных значений. Напомним, что m – это число строк в последовательности s. После применения данного подхода вероятность ошибки станет равной $O\left(1/2^{3\log m}\right) = O\left(1/m^3\right)$. Рассмотрим реализацию этого алгоритма. В качестве процедуры, реализующей алгоритм из леммы 1, мы будем рассматривать THE_FIRST_ONE_SEARCH(f,k), где $f(j) = (s_j \neq t_j)$.

Алгоритм 1 COMPARE_STRINGS(s,t,k). Квантовый алгоритм сравнения двух строк

```
j_0 ← THE_FIRST_ONE_SEARCH(f, k)                    ▷ Начальное значение
for i ∈ {1, ..., 3 log_2 m} do
    j_0 ← min(j_0, THE_FIRST_ONE_SEARCH(f, k))
end for
if j_0 = k + 1 then
    result ← 0                                      ▷ Строки равны
end if
if (j_0 ≠ k + 1)&(s_{j_0} < t_{j_0}) then
    result ← −1                                     ▷ s предшествует t
end if
if (j_0 ≠ k + 1)&(s_{j_0} > t_{j_0}) then
    result ← 1                                      ▷ t предшествует s
end if
return result
```

Лемма 2. *Алгоритм* 1 *сравнивает две строки длины* k *в лексикографическом порядке. Алгоритм имеет запросную сложность* $O(\sqrt{k}\log m)$ *и вероятность ошибки* $O\left(1/m^3\right)$.

Доказательство. Алгоритм находит минимальный индекс неравных символов двух строк. Пусть индекс этого символа есть j_0. Таким образом, как обсуждалось ранее, для определения, какая из строк меньше в лексикографическом порядке, достаточно сравнить символы s_{j_0} и t_{j_0}. Это и делает алгоритм. Обсудим вероятность ошибки. Алгоритм допускает ошибку только в случае, если ошибку допустили все $3\log_2 m$ запусков алгоритма THE_FIRST_ONE_SEARCH. Вероятность такого события не более чем $0.5^{3\log_2 m} = O\left(1/m^3\right)$. □

2.1.2. Квантовый онлайн-алгоритм для модели игры «запрос-ответ» с буфером. В первую очередь обсудим идею алгоритма. Воспользуемся известной структурой данных – самобалансирующимся двоичным деревом поиска. В качестве реализации данной структуры воспользуемся АВЛ-деревьями [36, 37] или красно-черными деревьями [37, 38]. Обе реализации дерева позволяют осуществлять поиск и добавление элемента за время $O(\log N)$, где N – число вершин в дереве.

В каждой вершине дерева будет сохраняться тройка (i,s,c), где i – минимальный индекс строки из $\{x^1,\ldots,x^m\}$, такой, что $s = x^i$; c – количество появлений s среди $x^1,\ldots,x^m$. Будем считать, что тройка (i,s,c) меньше тройки (i',s',c') тогда и только тогда, когда s предшествует s' в лексикографическом порядке. При этом процедура COMPARE_STRINGS(s,s',k) будет использоваться в качестве функции сравнения вершин.

Во-первых, алгоритм загружает все строки $s^1, \ldots, s^d$ по очереди в буфер и добавляет в дерево вершину, соответствующую тройке $v = (NULL, s^j, 0)$ для строки s^j, где $j \in \{1, \ldots, d\}$. Значение $NULL$ в v является начальным значением и означает, что на данный момент мы не знаем минимального i, такого что $x^i = s^j$; 0 в v означает, что алгоритм еще не встретил s^j среди $(x^1, \ldots, x^m)$. При этом если есть две или более одинаковых строк среди $s^1, \ldots, s^d$, алгоритм создает в дереве только одну вершину для каждой из различных строк.

Во-вторых, алгоритм загружает строки от x^1 до x^m по очереди в буфер и ищет их в дереве. Рассмотрим случай, кода очередная строка x^i была найдена среди сохраненных в дереве и ей соответствует тройка (j, s, c). Тогда число c увеличивается на 1 и, если j равно $NULL$, то в качестве j записывается число i. Если же строка не найдена в дереве, тогда ее нет среди $s^1, \ldots s^d$ и алгоритм игнорирует ее. В то же время алгоритм в отдельной тройке переменных сохраняет строку, которая встречалась чаще всего, то есть

$$(i_{\max}, s, c_{\max}) = \underset{(i,t,c)\ в\ дереве}{\operatorname{argmax}}\ c\,,$$

и перевычисляет эти переменные после обработки очередной строки. В случае, когда противник запрашивает очередную выходную переменную, алгоритм возвращает $i_{\max}$.

Опишем алгоритм формально. Пусть BST – это самобалансирующееся бинарное дерево поиска, обладающее следующими операциями:

• FIND(BST, x^i) находит вершину, которой соответствует тройка (j, s, c) такая, что $s = x^i$, или выдает $NULL$ в случае, если x^i не найдено.

Стандартный алгоритм поиска x^i в дереве состоит в том, чтобы сравнивать x^i со строками, сохраненными в вершинах, и двигаться по дереву в соответствие с результатами сравнения. В том случае, когда алгоритм вызывает процедуру COMPARE_STRINGS, он запрашивает переменные из буфера для тестирования символов строки x^i и переменные из собственной памяти для проверки символов строки, сохраненной в вершине.

• ADD(BST, s^j) добавляет вершину с тройкой $(NULL, s^j, 0)$ в дерево, если таковой еще нет, и возвращает вершину в качестве результата.

• INIT(BST) инициализирует пустое дерево.

Теорема 1. *Алгоритм* 2 *c-конкурентен в ожидании, причем c не превышает* $\mathcal{C}_Q$*, где*

$$\mathcal{C}_Q = O\left(1 + \frac{(m \log m) \cdot (\log d)}{\sqrt{k}}\right).$$

Доказательство. Корректность алгоритма следует из его описания. Обсудим запросную сложность процедуры FIND(BST, x^i). Процедура выполняет $O(\log d)$ операций сравнения COMPARE_STRINGS$(x^i, s^{i'}, k)$. Согласно лемме 2, каждая операция сравнения совершает $O(\sqrt{k} \log m)$ запросов. Общая запросная сложность операции FIND составляет $O\left(\sqrt{k}(\log m) \cdot (\log d)\right)$.

Таким образом, алгоритм проверяет все строки из $x^1, \ldots, x^m$ за $O\left(m\sqrt{k}(\log m) \cdot (\log d)\right)$ раундов и после этого гарантированно сможет возвращать противнику корректный результат.

Следовательно, первые $O\left(\dfrac{m\sqrt{k}(\log m) \cdot (\log d)}{k}\right) = O\left(\dfrac{m(\log m) \cdot (\log d)}{\sqrt{k}}\right)$ «значимые» выходные переменные могут быть ошибочными, а оставшиеся уже верные. Здесь будем называть выходные переменные $y_{(j+d)\cdot k}$ (для $j \in \{1, \ldots, m\}$)

Алгоритм 2 Квантовый онлайн-алгоритм для задачи поиска наиболее часто встречающейся строки

```
INIT(BST)                                          ▷ Инициализация дерева
c_max ← 1                                          ▷ Максимальная частота
i_max ← 1                                          ▷ Индекс наиболее частой строки
step ← 0
for j ∈ {1, ..., d} do
    LOAD_TO_BUFFER                                 ▷ Загрузка s^j в буфер
    t ← ""                                         ▷ Изначально t – пустая строка
    for q ∈ {1, ..., k} do                         ▷ Чтение строки t
        t ← t + RQUEST(q)                          ▷ Запрос q-й переменной из буфера
                                                     и сохранение ее в переменную t
    end for
    ADD(BST, t)                                    ▷ Добавление строки t = s^j в дерево
                                                     в составе тройки (NULL, t, 0)
end for
for i ∈ {1, ..., m} do
    LOAD_TO_BUFFER                                 ▷ Загрузка x^i в буфер
    v = (j, t, c) ← FIND(BST, x^i)                 ▷ Поиске x^i в дереве
    if v ≠ NULL then                               ▷ Если x^i входит в (s^1, ..., s^d)
        if j = NULL then
            j ← i                                  ▷ Если x^j не встречалась раньше,
                                                     то сохраняется индекс текущей строки
        end if
        c ← c + 1                                  ▷ Увеличиваем частоту
        v ← (j, t, c)                              ▷ Обновляем значение в вершине новой тройкой
        if c > c_max then                          ▷ Обновляем максимальные значения
            c_max ← c
            i_max ← i
        end if
    end if
end for
if Противник запрашивает выходную переменную then return i_max
end if
```

«значимыми», потому что стоимость ответа зависит именно от этих переменных. В таком случае, стоимость выходного набора будет составлять не более $1 + {} + O\left(\dfrac{m(\log m)\cdot(\log d)}{\sqrt{k}}\right)$.

Обсудим вероятность ошибки. Ошибка может возникать только при вызове процедуры сравнения строк. Все события возникновения ошибок независимы. Таким образом, для корректности всего алгоритма необходимо, чтобы все вероятностные процедуры сработали корректно. Согласно лемме 2, вероятность корректности одного запуска процедуры сравнения строк равна $1-\left(1-1/m^3\right)$. Следовательно, вероятность корректности всех $O(m\log m)$ событий не менее $1-\left(1-1/m^3\right)^{\gamma\cdot m\log m}$ для некоторой константы γ.

Заметим, что

$$\lim_{n\to\infty}\frac{\left(1-1/m^3\right)^{\gamma\cdot m\log m}}{1/m}<1.$$

Следовательно, итоговая вероятность ошибки не превышает $O\left(1/m\right)$.

В случае ошибки все «значимые» выходные переменные могут оказаться ошибочными. Тогда математическое ожидание конкретного соотношения алгоритма

не превышает

$$\mathcal{C}_Q = \frac{O\left(\frac{m-1}{m}\right)\cdot\left(1+O\left(\frac{m(\log m)\cdot(\log d)}{\sqrt{k}}\right)\right)+O\left(m\cdot\frac{1}{m}\right)}{1} = \\ = O\left(1+\frac{m(\log m)\cdot(\log d)}{\sqrt{k}}\right).$$

□

3. Нижняя оценка сложности классического онлайн-алгоритма для задачи поиска наиболее часто встречающейся строки

Покажем, что существует такой входной набор I_B, что любой классический (детерминированный или вероятностный) алгоритм будет возвращать выходной набор стоимостью $O(m)$.

Теорема 2. *Математическое ожидание конкурентного соотношения с любого вероятностного онлайн-алгоритма не менее чем $\mathcal{C}_R = O(m)$. В то же время $\mathcal{C}_R > \mathcal{C}_Q$ в случае, если $(\log_2 m)\cdot(\log_2 d) = o(\sqrt{k})$.*

Доказательство. Покажем, что существует входной набор такой, что поставленная задача эквивалентна задачи поиска на неструктурированных данных для этого набора. Пусть $m = 2t$ для некоторого целого t.

Тогда пусть $x^{t+1}, \ldots, x^{2t} = 0^k$, где 0^k – это строка из k нулей. Другие строки такие, что выполняется только один из двух случаев:

случай 1: $x^1, \ldots, x^t = 1^k$;

случай 2: существует $z \in \{1, \ldots, t\}$ и $u \in \{1, \ldots, k\}$ такие, что $x^z_u = 0$ и $x^z_{u'} = 1$ для всех $u' \in \{1, \ldots, u-1, u+1, \ldots, k\}$. При этом $x^{z'} = 1^k$ для $z' \in \{1, \ldots, t\}\backslash\{z\}$.

Пусть $d = 2$, $s^1 = 0^k$ и $s^2 = 1^k$.

В первом случае ответом будет 1^k, во втором – 0^k. Следовательно, на этом наборе данных задача эквивалентна задачи поиска 0 среди $tk = mk/2$ переменных.

Согласно [39], вероятностная запросная сложность поиска среди неструктурированных данных размера $mk/2$ составляет $\Omega(mk)$.

Если m – это нечетное число, то тогда возьмем $x^m = 1^{k/2}0^{k/2}$ и не будем его учитывать в поиске. Тогда мы можем рассматривать только $m-1$ строку, а $m-1$ в этом случае четное число.

Предположим, что существует вероятностный онлайн-алгоритм A, который решает задачу за $o(mk)$ запросов к буферу при чтении $x^1, \ldots, x^m$. Тогда противник может сконструировать входной набор I_B так, что A получит неверный ответ. Следовательно, все «значимые» выходные переменные будут ошибочными и $\mathrm{cost}(I_B, A(I_B)) = 1 + m$. Конкурентное соотношение в таком случае будет $\mathcal{C}_R = m + 1$.

Если же алгоритм выполняет $O(mk)$ запросов к буферу для вычисления ответа, то тогда $O(m)$ «значимых» выходных переменных должны быть выведены до того, как алгоритм сможет выдавать верный ответ. Следовательно, $\mathrm{cost}(I_B, A(I_B)) = O(m)$ и $\mathcal{C}_R = O(m)$.

В случае, если $(\log_2 m)\cdot(\log_2 d) = o(\sqrt{k})$, получаем

$$\mathcal{C}_Q = O\left(1+\frac{m(\log_2 m)\cdot(\log_2 d)}{\sqrt{k}}\right) = o(m) < O(m) = \mathcal{C}_R.$$

□

Заключение

В работе была рассмотрена новая модель для онлайн-алгоритмов, которая может быть применена к реальным задачам. Было показано, что в случае $(\log_2 m) \cdot (\log_2 d) = o(\sqrt{k})$ квантовый алгоритм оказывается более эффективным с точки зрения конкурентного соотношения по сравнению с любым классическим (вероятностным или детерминированным). Заметим, что данное ограничение вполне разумно и встречается в реальных задачах.

Благодарности. Исследование выполнено за счет гранта Российского научного фонда (проект № 19-71-00149).

Авторы выражают благодарность Аблаеву Фариду Мансуровичу и Хадиевой Алие Ихсановне (Казанский федеральный университет) за полезные советы при обсуждении задачи.

Литература

1. *Komm D.* An Introduction to Online Computation: Determinism, Randomization, Advice. – Springer, 2016. – XV, 349 p. – doi: 10.1007/978-3-319-42749-2.
2. *Sleator D.D., Tarjan R.E.* Amortized efficiency of list update and paging rules // Communications of the ACM. – 1985. – V. 28, No 2. – P. 202–208. – doi: 10.1145/2786.2793.
3. *Karlin A.R., Manasse M.S., Rudolph L., Sleator D.D.* Competitive snoopy caching // 27th Annual Symposium on Foundations of Computer Science. – Toronto, Ont., Canada, 1986. – P. 244–254. – doi: 10.1109/SFCS.1986.14.
4. *Albers S.* Competitive Online Algorithms: A BRICS Mini-Course. – 1996. – URL: https://www.brics.dk/LS/96/2/.
5. Java Platform SE 8 documentation. – URL: https://docs.oracle.com/javase/8/docs/api/java/io/BufferedReader.html.
6. *Lippman S.B., Lajoie J.* C++ Primer. – Massachusetts: Addison-Wesley, 1998. – 1264 p.
7. *Nielsen M.A., Chuang I.L.* Quantum Computation and Quantum Information. – Cambridge Univ. Press, 2010. – 702 p. – doi: 10.1017/CBO9780511976667.
8. *Ambainis A.* Understanding quantum algorithms via query complexity // Proc. Int. Congr. of Mathematicians (ICM 2018). – 2019. – P. 3265–3285. – doi: 10.1142/9789813272880_0181.
9. *Ablayev F., Ablayev M., Huang J.Z., Khadiev K., Salikhova N., Wu D.* On quantum methods for machine learning problems part I: Quantum tools // Big Data Mining and Analytics. – 2019. – V. 3, No 1. – P. 41–55. – doi: 10.26599/BDMA.2019.9020016.
10. *Kapralov R., Khadiev K., Mokut J., Shen Y., Yagafarov M.* Fast classical and quantum algorithms for online k-server problem on trees. – 2020. – arXiv:2008.00270.
11. *Wolf de R.* Quantum computing and communication complexity: Acad. Proefschrift. – 2001. – 225 p. – URL: https://homepages.cwi.nl/∼rdewolf/publ/qc/phd.pdf.
12. *Jordan St.* Quantum Algorithm Zoo. – URL: https://quantumalgorithmzoo.org/.
13. *Khadiev K., Safina L.* Quantum algorithm for dynamic programming approach for DAGs. Applications for Zhegalkin polynomial evaluation and some problems on DAGs // McQuillan I., Seki S. (Eds.) Unconventional Computation and Natural Computation. UCNC 2019. Lecture Notes in Computer Science, V. 11493. – Cham: Springer, 2019. – P. 150–163. – doi: 10.1007/978-3-030-19311-9_13.
14. *Khadiev K., Kravchenko D., Serov D.* On the quantum and classical complexity of solving subtraction games // van Bevern R., Kucherov G. (Eds.) Computer Science – Theory and

Applications. CSR 2019. Lecture Notes in Computer Science, V. 11532. – Cham: Springer, 2019. – P. 228–236. – doi: 10.1007/978-3-030-19955-5_20.

15. *Khadiev K., Mannapov I., Safina L.* The quantum version of classification decision tree constructing algorithm C5.0 // Hölldobler S., Malikov A. (eds.) Proc. YSIP-3 Workshop. – 2019. – URL: http://ceur-ws.org/Vol-2500/paper_6.pdf.
16. *Ambainis A., Nahimovs N.* Improved constructions of quantum automata // Theor. Comput. Sci. – 2009. – V. 410, No 20. – P. 1916–1922. – doi: 10.1016/j.tcs.2009.01.027.
17. *Ablayev F.M., Vasilyev A.V.* On quantum realisation of Boolean functions by the fingerprinting technique // Discrete Math. Appl. – 2009. – V. 19, No 6. – P. 555–572. – doi: 10.1515/DMA.2009.037.
18. *Ablayev F., Gainutdinova A., Khadiev K., Yakaryilmaz A.* Very narrow quantum OBDDs and width hierarchies for classical OBDDs //Jürgensen H., Karhumäki J., Okhotin A. (Eds.) Descriptional Complexity of Formal Systems. DCFS 2014. Lecture Notes in Computer Science, V. 8614. – Cham: Springer, 2014. – P. 53–64. – doi: 10.1007/978-3-319-09704-6_6.
19. *Ablayev F. Gainutdinova A., Khadiev K., Yakaryilmaz A.* Very narrow quantum OBDDs and width hierarchies for classical OBDDs // Lobachevskii J. Math. – 2016. – V. 37, No 6. – P. 670–682. – doi: 10.1134/S199508021606007X.
20. *Ablayev F., Ambainis A., Khadiev K., Khadieva A.* Lower bounds and hierarchies for quantum memoryless communication protocols and quantum ordered binary decision diagrams with repeated test // Tjoa A., Bellatreche L., Biffl S., van Leeuwen J., Wiedermann J. (Eds.) SOFSEM 2018: Theory and Practice of Computer Science. SOFSEM 2018. Lecture Notes in Computer Science, V. 10706. – Cham: Edizioni della Normale, 2018. – P. 197–211. – doi: 10.1007/978-3-319-73117-9_14.
21. *Ablayev F., Ablayev M., Khadiev K., Vasiliev A.* Classical and quantum computations with restricted memory // Böckenhauer H.J., Komm D., Unger W. (Eds.) Adventures Between Lower Bounds and Higher Altitudes. Lecture Notes in Computer Science, V. 11011. – Cham: Springer, 2018. – P. 129–155. – doi: 10.1007/978-3-319-98355-4_9.
22. *Khadiev K., Khadieva A., Mannapov I.* Quantum online algorithms with respect to space and advice complexity // Lobachevskii J. Math. – 2018. – V. 39, No 9. – P. 1377–1387. – doi: 10.1134/S1995080218090421.
23. *Khadiev K., Khadieva A.* Reordering method and hierarchies for quantum and classical ordered binary decision diagrams // Weil P. (Ed.) Computer Science – Theory and Applications. CSR 2017. Lecture Notes in Computer Science, V. 10304. – Cham: Springer, 2017. – P. 162–175. – doi: 10.1007/978-3-319-58747-9_16.
24. *Ibrahimov R., Khadiev K., Prūsis K., Yakaryilmaz A.* Error-free affine, unitary, and probabilistic OBDDs // Konstantinidis S., Pighizzini G. (Eds.) Descriptional Complexity of Formal Systems. DCFS 2018. Lecture Notes in Computer Science, V. 10952. – Cham: Springer, 2018. – P. 175–187. – doi: 10.1007/978-3-319-94631-3_15.
25. *Le Gall F.* Exponential separation of quantum and classical online space complexity // Theory Comput. Syst. – 2009. – V. 45, No 2. – P. 188–202. – doi: 10.1007/s00224-007-9097-3.
26. *Khadiev K., Ilikaev A.* Quantum algorithms for the most frequently string search, intersection of two string sequences and sorting of strings problems // Martín-Vide C., Pond G., Vega-Rodríguez M. (Eds.) Theory and Practice of Natural Computing. TPNC 2019. Lecture Notes in Computer Science, V. 11934. – Cham: Springer, 2019. – P. 234–245. – doi: 10.1007/978-3-030-34500-6_17.
27. *Khadiev K., Khadieva A., Kravchenko D., Rivosh A., Yamilov A., Mannapov I.* Quantum versus classical online streaming algorithms with logarithmic size of memory. – 2019. – arXiv:1710.09595v3.

28. *Khadiev K., Khadieva A.* Quantum online streaming algorithms with logarithmic memory // Int. J. Theor. Phys. – 2019. – doi: 10.1007/s10773-019-04209-1. – URL: https://link.springer.com/article/10.1007%2Fs10773-019-04209-1.

29. *Khadiev K., Khadieva A.* Two-way quantum and classical machines with small memory for online minimization problems // Proc. SPIE V. 11022: International Conference on Micro- and Nano-Electronics 2018. – 2019. – Art. 110222T, P. 1–9. – doi: 10.1117/12.2522462.

30. *Yuan Q.* Quantum online algorithms: PhD Thesis. – Santa Barbara, US: University of California at Santa Barbara, 2009. – 88 p.

31. *Cormode Gr., Hadjieleftheriou M.* Finding frequent items in data streams // Proc. VLDB Endowment. – 2008. – V. 1, No 2. – P. 1530–1541. – doi: 10.14778/1454159.1454225.

32. *Muthukrishnan S.* Data streams: Algorithms and applications // Foundations and Trends in Theoretical Computer Science. – 2005. – V. 1, No 2. – P. 117–236.

33. *Aggarwal Ch.C.* Data Streams: Models and Algorithms. – Springer US, 2007. – XVIII, 354 p. – doi: 10.1007/978-0-387-47534-9.

34. *Becchetti L., Chatzigiannakis I., Giannakopoulos Y.* Streaming techniques and data aggregation in networks of tiny artefacts // Comput. Sci. Rev. – 2011. – V. 5, No 1. – P. 27–46. – doi: 10.1016/j.cosrev.2010.09.007.

35. *Boyar J., Larsen K.S., Maiti A.* The frequent items problem in online streaming under various performance measures // Int. J. Found. Comput. Sci. – 2015. – V. 26, No 4. – P. 413–439. – doi: 10.1142/S0129054115500239.

36. *Адельсон-Вельский Г.М., Ландис Е.М.* Один алгоритм организации информации // Докл. АН СССР. – 1962. – Т. 146, № 2. – С. 263–266.

37. *Cormen T.H., Leiserson C.E., Rivest R.L., Stein C.* Introduction to Algorithms. – Cambridge, Mass.: MIT Press, 2001. – 1180 p.

38. *Guibas L. J, Sedgewick R.* A dichromatic framework for balanced trees // 19th Annu. Symp. on Foundations of Computer Science – IEEE, 1978. – P. 8–21. – doi: 10.1109/SFCS.1978.3.

39. *Bennett Ch.H., Bernstein E., Brassard G., Vazirani U.* Strengths and weaknesses of quantum computing // SIAM J. Comput. – 1997. – V. 26, No 5. – P. 1510–1523. – doi: 10.1137/S0097539796300933.

Поступила в редакцию
04.08.2020

Хадиев Камиль Равилевич, кандидат физико-математических наук, научный сотрудник; старший научный сотрудник лаборатории «Квантовые методы обработки информации»

ООО «Квантовые интеллектуальные технологии»
ул. К. Маркса, д. 5, оф. 36, г. Казань, 420111, Россия
Казанский (Приволжский) федеральный университет
ул. Кремлевская, д. 18, г. Казань, 420008, Россия
E-mail: *kamilhadi@gmail.com*

Лин Дмитрий Игоревич, студент Института вычислительной математики и информационных технологий; разработчик

Казанский (Приволжский) федеральный университет
ул. Кремлевская, д. 18, г. Казань, 420008, Россия
Компания АО «Барс Груп»
ул. Некрасова, д. 9, г. Казань, 420012, Россия
E-mail: *dmitrijlin9@gmail.com*

ISSN 2541-7746 (Print)
ISSN 2500-2198 (Online)

**UCHENYE ZAPISKI KAZANSKOGO UNIVERSITETA.
SERIYA FIZIKO-MATEMATICHESKIE NAUKI
(Proceedings of Kazan University. Physics and Mathematics Series)**

2020, vol. 162, no. 3, pp. 367–382

doi: 10.26907/2541-7746.2020.3.367-382

Quantum Online Algorithms for a Model of the Request-Answer Game with a Buffer

K.R. Khadiev[a,b*], *D.I. Lin*[b,c**]

[a] *OOO Kvantovye intellektual'nye tekhnologii, Kazan, 420111 Russia*
[b] *Kazan Federal University, Kazan, 420008 Russia*
[c] *AO Bars Grup, Kazan, 420012 Russia*
E-mail: **kamilhadi@gmail.com*, ***dmitrijlin9@gmail.com*

Received August 4, 2020

Abstract

In this paper, we considered online algorithms as a request-answer game between two players: an adversary that generates input requests and an online algorithm that answers them. A generalized version of the game that has a buffer of limited size was studied. The adversary loads data to the buffer, while the algorithm has random access to elements of the buffer. For the model, quantum and classical (deterministic or randomized) algorithms were a focus of attention. A specific problem and a quantum algorithm that works better than any classical (deterministic or randomized) algorithm, in terms of competitive ratio, were provided. At the same time, for the problem, classical online algorithms in the standard model are equivalent to the classical algorithms in the request-answer game with a buffer model.

Keywords: quantum computation, online algorithms, request-answer game, online minimization problem, computation with buffer

Acknowledgments. The study was supported by the Russian Science Foundation (project no. 19-71-00149).

We are grateful to Farid Mansurovich Ablaev and Aliya Ikhsanovna Khadieva (Kazan Federal University) for their valuable advice during the discussion of the problem under consideration.

References

1. Komm D. *An Introduction to Online Computation: Determinism, Randomization, Advice*. Springer, 2016. XV, 349 p. doi: 10.1007/978-3-319-42749-2.
2. Sleator D.D., Tarjan R.E. Amortized efficiency of list update and paging rules. *Commun. ACM*, 1985, vol. 28, no. 2, pp. 202–208. doi: 10.1145/2786.2793.
3. Karlin A.R., Manasse M.S., Rudolph L., Sleator D.D. Competitive snoopy caching. *Proc. 27th Annu. Symp. on Foundations of Computer Science*. Toronto, Ont., Canada, 1986, pp. 244–254. doi: 10.1109/SFCS.1986.14.
4. Albers S. *Competitive Online Algorithms: A BRICS Mini-Course*. 1996. Available at: https://www.brics.dk/LS/96/2/.

5. *Java Platform SE 8 documentation.* Available at: https://docs.oracle.com/javase/8/docs/api/java/io/BufferedReader.html.

6. Lippman S.B., Lajoie J. *C++ Primer.* Massachusetts, Addison-Wesley, 1998. 1264 p.

7. Nielsen M.A., Chuang I.L. *Quantum Computation and Quantum Information.* Cambridge Univ. Press, 2010. 702 p. doi: 10.1017/CBO9780511976667.

8. Ambainis A. Understanding quantum algorithms via query complexity. *Proc. Int. Congr. of Mathematicians (ICM 2018)*, 2019, pp. 3265–3285. doi: 10.1142/9-789813272880_0181.

9. Ablayev F., Ablayev M., Huang J.Z., Khadiev K., Salikhova N., Wu D. On quantum methods for machine learning problems part I: Quantum tools. *Big Data Min. Anal.*, 2019, vol. 3, no. 1, pp. 41–55. doi: 10.26599/BDMA.2019.9020016.

10. Kapralov R., Khadiev K., Mokut J., Shen Y., Yagafarov M. Fast classical and quantum algorithms for online k-server problem on trees. 2020. arXiv:2008.00270.

11. Wolf de R. *Quantum Computing and Communication Complexity: Acad. Proefschrift.* 2001. 225 p. Available at: https://homepages.cwi.nl/∼rdewolf/publ/qc/phd.pdf.

12. Jordan St. *Quantum Algorithm Zoo.* Available at: https://quantumalgorithmzoo.org/.

13. Khadiev K., Safina L. Quantum algorithm for dynamic programming approach for DAGs. Applications for Zhegalkin polynomial evaluation and some problems on DAGs. In: McQuillan I., Seki S. (Eds.) *Unconventional Computation and Natural Computation. UCNC 2019. Lecture Notes in Computer Science.* Cham, Springer, 2019, vol. 11493, pp. 150–163. doi: 10.1007/978-3-030-19311-9_13.

14. Khadiev K., Kravchenko D., Serov D. On the quantum and classical complexity of solving subtraction games. In: van Bevern R., Kucherov G. (Eds.) *Computer Science – Theory and Applications. CSR 2019. Lecture Notes in Computer Science.* Cham, Springer, 2019, vol. 11532, pp. 228–236. doi: 10.1007/978-3-030-19955-5_20.

15. Khadiev K., Mannapov I., Safina L. The quantum version of classification decision tree constructing algorithm C5.0. In: Hölldobler S., Malikov A. (Eds.) *Proc. YSIP-3 Workshop*, 2019. Available at: http://ceur-ws.org/Vol-2500/paper_6.pdf.

16. Ambainis A., Nahimovs N. Improved constructions of quantum automata. *Theor. Comput. Sci.*, 2009, vol. 410, no. 20, pp. 1916–1922. doi: 10.1016/j.tcs.2009.01.027.

17. Ablayev F.M., Vasilyev A.V. On quantum realisation of Boolean functions by the fingerprinting technique. *Discrete Math. Appl.*, 2009, vol. 19, no. 6, pp. 555–572. doi: 10.1515/DMA.2009.037.

18. Ablayev F., Gainutdinova A., Khadiev K., Yakaryilmaz A. Very narrow quantum OBDDs and width hierarchies for classical OBDDs. In: Jürgensen H., Karhumäki J., Okhotin A. (Eds.) *Descriptional Complexity of Formal Systems. DCFS 2014. Lecture Notes in Computer Science.* Cham, Springer, 2014, vol. 8614, pp. 53–64. doi: 10.1007/978-3-319-09704-6_6.

19. Ablayev F. Gainutdinova A., Khadiev K., Yakaryilmaz A. Very narrow quantum OBDDs and width hierarchies for classical OBDDs. *Lobachevskii J. Math.*, 2016, vol. 37, no. 6, pp. 670–682. doi: 10.1134/S199508021606007X.

20. Ablayev F., Ambainis A., Khadiev K., Khadieva A. Lower bounds and hierarchies for quantum memoryless communication protocols and quantum ordered binary decision diagrams with repeated test. In: Tjoa A., Bellatreche L., Biffl S., van Leeuwen J., Wiedermann J. (Eds.) *SOFSEM 2018: Theory and Practice of Computer Science. SOFSEM 2018. Lecture Notes in Computer Science.* Cham, Edizioni della Normale, 2018, vol. 10706, pp. 197–211. doi: 10.1007/978-3-319-73117-9_14.

21. Ablayev F., Ablayev M., Khadiev K., Vasiliev A. Classical and quantum computations with restricted memory. In: Böckenhauer H.J., Komm D., Unger W. (Eds.) *Adventures Between Lower Bounds and Higher Altitudes. Lecture Notes in Computer Science*. Cham, Springer, 2018, vol. 11011, pp. 129–155. doi: 10.1007/978-3-319-98355-4_9.

22. Khadiev K., Khadieva A., Mannapov I. Quantum online algorithms with respect to space and advice complexity. *Lobachevskii J. Math.*, 2018, vol. 39, no. 9, pp. 1377–1387. doi: 10.1134/S1995080218090421.

23. Khadiev K., Khadieva A. Reordering method and hierarchies for quantum and classical ordered binary decision diagrams. In: Weil P. (Ed.) *Computer Science – Theory and Applications. CSR 2017. Lecture Notes in Computer Science*. Cham, Springer, 2017, vol. 10304, pp. 162–175. doi: 10.1007/978-3-319-58747-9_16.

24. Ibrahimov R., Khadiev K., Prūsis K., Yakaryilmaz A. Error-free affine, unitary, and probabilistic OBDDs. In: Konstantinidis S., Pighizzini G. (Eds.) *Descriptional Complexity of Formal Systems. DCFS 2018. Lecture Notes in Computer Science*. Cham, Springer, 2018, vol. 10952, pp. 175–187. doi: 10.1007/978-3-319-94631-3_15.

25. Le Gall F. Exponential separation of quantum and classical online space complexity. *Theory Comput. Syst.*, 2009, vol. 45, no. 2, pp. 188–202. doi: 10.1007/s00224-007-9097-3.

26. Khadiev K., Ilikaev A. Quantum algorithms for the most frequently string search, intersection of two string sequences and sorting of strings problems. In: Martín-Vide C., Pond G., Vega-Rodríguez M. (Eds.) *Theory and Practice of Natural Computing. TPNC 2019. Lecture Notes in Computer Science*. Cham, Springer, 2019, vol. 11934, pp. 234–245. doi: 10.1007/978-3-030-34500-6_17.

27. Khadiev K., Khadieva A., Kravchenko D., Rivosh A., Yamilov A., Mannapov I. Quantum versus classical online streaming algorithms with logarithmic size of memory. 2019. arXiv:1710.09595v3.

28. Khadiev K., Khadieva A. Quantum online streaming algorithms with logarithmic memory. *Int. J. Theor. Phys.*, 2019. doi: 10.1007/s10773-019-04209-1. Available at: https://link.springer.com/article/10.1007%2Fs10773-019-04209-1.

29. Khadiev K., Khadieva A. Two-way quantum and classical machines with small memory for online minimization problems. *Proc. SPIE, Vol. 11022: Int. Conf. on Micro- and Nano-Electronics 2018*, 2019, art. 110222T, pp. 1–9. doi: 10.1117/12.2522462.

30. Yuan Q. Quantum online algorithms. *PhD Thesis*. Santa Barbara, US, Univ. Calif. at Santa Barbara, 2009. 88 p.

31. Cormode Gr., Hadjieleftheriou M. Finding frequent items in data streams. *Proc. VLDB Endowment*, 2008, vol. 1, no. 2, pp. 1530–1541. doi: 10.14778/1454159.1454225.

32. Muthukrishnan S. Data streams: Algorithms and applications. *Found. Trends Theor. Comput. Sci.*, 2005, vol. 1, no. 2, pp. 117–236.

33. Aggarwal Ch.C. *Data Streams: Models and Algorithms*. Springer US, 2007. XVIII, 354 p. doi: 10.1007/978-0-387-47534-9.

34. Becchetti L., Chatzigiannakis I., Giannakopoulos Y. Streaming techniques and data aggregation in networks of tiny artefacts. *Comput. Sci. Rev.*, 2011, vol. 5, no. 1, pp. 27–46. doi: 10.1016/j.cosrev.2010.09.007.

35. Boyar J., Larsen K.S., Maiti A. The frequent items problem in online streaming under various performance measures. *Int. J. Found. Comput. Sci.*, 2015, vol. 26, no. 4, pp. 413–439. doi: 10.1142/S0129054115500239.

36. Adel'son-Vel'skii G.M., Landis E.M. An algorithm for organization of information. *Dokl. Akad. Nauk SSSR*, 1962, vol. 146, no. 2, pp. 263–266. (In Russian)

37. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. *Introduction to Algorithms*. Cambridge, Mass., MIT Press, 2001. 1180 p.

38. Guibas L. J, Sedgewick R. A dichromatic framework for balanced trees. *Proc. 19th Annu. Symp. on Foundations of Computer Science*. IEEE, 1978, pp. 8–21. doi: 10.1109/SFCS.1978.3.

39. Bennett Ch.H., Bernstein E., Brassard G., Vazirani U. Strengths and weaknesses of quantum computing. *SIAM J. Comput.*, 1997, vol. 26, no. 5, pp. 1510–1523. doi: 10.1137/S0097539796300933.

⟨ ***Для цитирования:*** *Хадиев К.Р., Лин Д.И.* Квантовые онлайн-алгоритмы для модели игры запрос-ответ с буфером // Учен. зап. Казан. ун-та. Сер. Физ.-матем. науки. – 2020. – Т. 162, кн. 3. – С. 367–382. – doi: 10.26907/2541-7746.2020.3.367-382. ⟩

⟨ ***For citation:*** Khadiev K.R., Lin D.I. Quantum online algorithms for a model of the request-answer game with a buffer. *Uchenye Zapiski Kazanskogo Universiteta. Seriya Fiziko-Matematicheskie Nauki*, 2020, vol. 162, no. 3, pp. 367–382. doi: 10.26907/2541-7746.2020.3.367-382. (In Russian) ⟩