

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Образовательная программа «Прикладная математика и информатика»

Отчет об программном проекте

на тему Подготовка задач для олимпиады «Высшая проба»

Выполнил:

Студент группы БПМИ181

М. Деб Натх

Дата _____ 2020

Принял:

Руководитель проекта

Густокашин Михаил Сергеевич

Директор
Центра студенческих олимпиад ФКН НИУ ВШЭ

Дата _____ 2020

Оценка _____

Москва 2020

1 Реферат

Проведённая работа имела своей целью подготовку алгоритмических задач для проведения школьных олимпиад «Высшая проба по информатике» и «Московская олимпиада школьников по информатике» (обе проводились центром студенческих олимпиад НИУ ВШЭ). Результатом работы являются три подготовленные задачи, каждая из которых была предложена участникам олимпиад, а именно, отборочного и финального туров московской олимпиады школьников и отборочного тура олимпиады «Высшая проба».

Каждая из этих задач была протестирована членами методической комиссии данной олимпиады, после чего была предложена участникам. Написанный код успешно справился со своей задачей, так как во время проведения туров олимпиад, на которых были даны приготовленные в рамках данного проекта задачи, не возникло технических трудностей на стороне организаторов олимпиады, связанных с подготовленными задачами. Результаты работы написанных программ были использованы при оценивании решений участников олимпиады, ранжирования их и в конечном итоге – награждения победителей дипломами ВСОШ.

Содержание

1	Реферат	2
2	Введение	4
2.1	Краткое описание предметной области	4
2.2	Актуальность проблемы	4
2.3	Цель и задачи проекта	4
3	Обзор и сравнительный анализ источников и аналогов	6
3.1	Аналоги	6
3.2	Выбранные методы для разработки проекта	6
3.3	Функциональные и нефункциональные требования к программному проекту	6
4	Теоретическая часть. Описание выбранных методов, алгоритмов и методик	8
4.1	Задача №1	8
4.1.1	Условие задачи	8
4.1.2	Решение задачи	9
4.2	Задача №2	11
4.2.1	Условие задачи	11
4.2.2	Решение задачи	11
4.3	Задача №3	12
4.3.1	Условие задачи	12
4.3.2	Решение задачи	12
5	Реализация проекта	14
5.1	Задача №1	14
5.2	Задача №2	15
5.3	Задача №3	16
6	Заключение	17

2 Введение

2.1 Краткое описание предметной области

Предметная область проекта — разработка задач для соревнований по программированию, в том числе состязаний для школьников по программированию, проводимых НИУ ВШЭ, а именно, «Высшая проба по информатике», «Московская олимпиада школьников по информатике» .

2.2 Актуальность проблемы

Создание качественных задач по программированию является актуальной задачей, так как с ростом популярности состязательного программирования растёт и нужда в разнообразных и оригинальных задачах, для решения которых могут потребоваться знания компьютерных наук, смежных им областей математики, знание языков программирования и умение написания программ. Проведение олимпиад с задачами, удовлетворяющими вышеописанным требованиям является важной задачей, так как олимпиады «Высшая проба» и «Московская олимпиада по информатике» включены в Перечень олимпиад школьников, дающих льготы при поступлении в высшие учебные заведения РФ, в том числе и НИУ ВШЭ. Статус победителя этих олимпиад даёт право на поступление на ФКН без вступительных испытаний.

2.3 Цель и задачи проекта

Целью проекта была подготовка пакета нескольких задач по спортивному программированию, включающих в себя, но не ограничивающими следующими составными частями:

1. Условие задачи – набранное в системе вёрстки L^AT_EX условие, описывающее преамбулу задачи, формат входных и выходных данных, ограничения на входные данные.
2. Проверяющая программа (checker) – программа, сверяющая вывод на конкретном тестовом примере ответ участника и ответ жюри, выводящая вердикт по данному тестовому примеру.
3. Главное авторское решение – программа, оптимально решающая данную задачу, с выводом которой будет сверяться вывод участника.
4. Частичные решения – либо правильные, но неоптимальные программы, либо неправильные программы, которые не проходят все тесты. Нужны для валидации корректности собранного пакета.
5. Валидатор (validator) – программа, проверяющая корректность формата входных данных

6. Тесты – набор тестовых входных данных и сгенерированный к ним набор выходных данных
7. Генераторы (generators) программы, генерирующие по входным аргументам тестовые примеры.

3 Обзор и сравнительный анализ источников и аналогов

3.1 Аналоги

Так как ежегодно проводятся сотни состязаний по спортивному программированию – как онлайн-соревнования на платформах Codeforces [3], Timus [7], Codechief [?], TopCoder[10], так и оффлайн-мероприятия, такие как Yandex.Algorithm [9], Google Code Jam [12], Всероссийская олимпиада школьников по программированию [11], Facebook Hacker Cup.

В силу специфики проекта не представляется возможным переиспользование уже использованных задач – это нарушит основополагающие правила проведения олимпиад и уничтожит соревновательный аспект состязания.

3.2 Выбранные методы для разработки проекта

В силу технологических особенностей олимпиада «Высшая проба» проводится на платформе ejudge, развёрнутой на сервере НИУ ВШЭ, а «Московская олимпиада школьников» проводится на платформе Yandex.Contest, развёрнутой на серверах Яндекса. Это ограничивает возможность использования иных форматов, кроме тех, что поддерживают эти системы. Поэтому было принято решение для хранения пакета использовать форматы систем Яндекс.Контекст [1] и Codeforces Polygon [2].

Codeforces Polygon – система подготовки задач, представленная командой разработки системы проведения соревнований codeforces.com [3]. Является де-факто стандартом подготовки олимпиадных задач, импортрование пакетов задач формате polygon поддерживается на основные решения для проведения олимпиад по программированию – ejudge [5], codeforces [3], Yandex Contest [1]. Именно широкая поддержка этого формата разными платформами и послужила причиной выбора этой системы для разработки пакетов задач.

3.3 Функциональные и нефункциональные требования к программному проекту

В связи с вышеописанными технологическими ограничениями складывается единственное функциональное требование – работоспособность при использовании системой подготовленного пакета задач, выражающаяся в возможности отправить произвольное решение в систему по подготовленной задаче и получить один из следующих вердиктов:

1. OK — OK, Решение является полным
2. CE — Compilation Error, Ошибка компиляции
3. RT — Run-Time Error, Ошибка при работе программы

4. TL — Time-Limit Exceeded, Ошибка превышения лимита времени
5. PE — Presentation Error, Ошибка неправильного формата результата
6. WA — Wrong Answer, Неправильный ответ

Описанные выше вердикты продиктованы функциональными особенностями систем и общепринятыми стандартами.

В качестве нефункциональных требований к подготавливаемым задачам можно выделить то, чтобы задача в своём решении требовала от участника знаний в использовании алгоритмов и структур данных, умения пользоваться языками программирования, а решение задачи было нетривиально и требовало как творческого, так и математического мышления.

4 Теоретическая часть. Описание выбранных методов, алгоритмов и методик

4.1 Задача №1

Первой задачей, подготовленной в рамках данного проекта была задача «С» (нумерация задач в мире спортивного программирования традиционно буквенная) со второго отборочного этапа олимпиады «Высшая проба». Этот этап проходил в качестве онлайн-тура длительностью 3 часа на платформе ejudge. Ниже приведены условия и решения задачи, важные аспекты реализации приведены в секции 5.1

4.1.1 Условие задачи

В условии этой задачи дана полоска (матрица) натуральных чисел размера $N \times 2$ и натуральное число K .

Требуется разместить на данной полоске ровно K непересекающихся доминошек (матриц 2×1 или 1×2) таким образом, чтобы сумма чисел на не покрытых доминошками клеток была минимально. Доминошки можно поворачивать горизонтально или вертикально, они не могут накладываться. Обязательно использовать все имеющиеся доминошки.

Требуется определить, как именно надо расположить доминошки для минимизации целевой функции и вывести любую найденную конфигурацию, на которой достигается этот минимум.

Ограничения на входные данные:

- $1 \leq N \leq 2 \cdot 10^5$;
- $0 \leq K \leq 2 \cdot 10^5$;
- $0 \leq N \times K \leq 2 \cdot 10^5$;
- $N \geq K$;

Ограничения на решение задачи:

- Ограничение по виртуальной памяти, которую использует решение участника: 256MB
- Ограничение по виртуальному времени, которое использует решение участника: 1 сек.

Оценка потестовая, а именно, в задаче ровно 30 тестов, за прохождение каждого из них участнику начисляется один балл. Решение, правильно работающее при ограничениях $N \leq 20$ будет набирать не менее 15 баллов.

4.1.2 Решение задачи

Для решения этой задачи участнику предлагалось либо реализовать полный перебор (для получения 15 баллов), либо воспользоваться методом динамического программирования для полного решения.

В первую очередь заметим, что задача минимизации суммы непокрытых клеток равносильна задаче максимизации суммы покрытых клеток, так как их общая сумма есть константа – сумма всех чисел в таблице.

По своей сути динамическое программирование – это метод математической индукции или рекурсивная формула, реализованная программно. Динамическое программирование в спортивном программировании – способ решения сложных задач путём разбиения их на более простые подзадачи. Динамическое программирование представляет из себя переформулирование сложной задачи в виде рекурсивной последовательности более простых подзадач. Простейшим примером применения динамического программирования можно считать алгоритм линейного вычисления n -го числа Фибоначчи.

В данном случае предлагалось использовать то, что называется многомерным динамическим программированием – подвидом д.п., при котором подзадача описывается более чем одним параметром.

Обозначим за $dp[k][i][j]$, где $0 \leq i \leq N$, $0 \leq j \leq K$, $0 \leq k \leq 3$ число, равное максимальной сумме покрытых клеток, если рассматривается задача покрытия первых i столбцов j доминошками, при этом в последнем столбце:

- ни одна клетка не покрыта, если $k = 0$;
- только верхняя клетка покрыта, если $k = 1$;
- только нижняя клетка покрыта, если $k = 2$;
- обе клетки покрыты, если $k = 3$;

Тогда нетрудно видеть рекурсивную формулу для этих величин: Если обозначить данный двумерный массив за A , то

$$dp[0][i][j] = \max(\begin{aligned} & dp[0][i-1][j], \\ & dp[1][i-1][j], \\ & dp[2][i-1][j], \\ & dp[3][i-1][j] \end{aligned});$$

$$dp[1][i][j] = \max(\begin{aligned} & dp[0][i-1][j-1], \\ & dp[2][i-1][j-1] \end{aligned}) + A[i][0] + A[i-1][0];$$

$$dp[2][i][j] = \max(\begin{aligned} & dp[0][i-1][j-1], \\ & dp[1][i-1][j-1] \end{aligned}) + A[i][0] + A[i-1][0];$$

$$\begin{aligned} & \text{dp}[0][i-1][j-1], \\ & \text{dp}[1][i-1][j-1] \\ &) + A[i][1] + A[i-1][1]; \end{aligned}$$

$$\begin{aligned} \text{dp}[3][i][j] = \max(\\ & \text{dp}[0][i-1][j-1], \\ & \text{dp}[1][i-1][j-1], \\ & \text{dp}[2][i-1][j-1], \\ & \text{dp}[3][i-1][j-1] \\ &) + A[i][0] + A[i][1]; \end{aligned}$$

С помощью такой рекурсивной формулы, положив изначально $\text{dp}[k][i][j] = +\infty$, а также $\text{dp}[0][0][0] = 0; \text{dp}[3][0][1] = A[0][0] + A[0][1];$, можем за $\mathcal{O}(NK)$ вычислить все значения dp и ответ в таком случае будет равен $\min_{0 \leq k \leq 3} \text{dp}[k][N][K]$. На языках C, C++, Java, Pascal такое решение будет укладываться в указанные ограничения по памяти и времени.

4.2 Задача №2

Задачи 2 и 3 являли собой задачи с открытыми тестами – такой формат часто используется на Московской олимпиаде по программированию. Это значит, что тесты известны участнику и его задача – лишь найти ответы к данным тестам. Часто задачи с открытыми тестами бывают оптимизационными задачами – когда точный ответ неизвестен и работа участника оценивается в сравнении с решениями жюри и других участников – чем ближе к максимальному известному ответу участник смог найти решение, тем выше его балл

Вторая задача была предложена на втором отборочном этапе олимпиады «Московская олимпиада школьников» для учащихся 10-11 классов. Раунд проходил на платформе Яндекс.Контест и длился несколько дней.

4.2.1 Условие задачи

Вторая задача была реализацией известной NP-полной задачи SETCOV, ниже приведена её формулировка:

Дано множество A из n элементов (все элементы – числа от 1 до n). Также есть семейство B из m подмножеств. i -е подмножество содержит ровно k_i элементов, равных $x_{i1}, x_{i2}, \dots, x_{ik_i}$. Разумеется, один элемент может быть выбран не более чем в одно множество. Задача – выбрать какой-то поднабор подмножеств из данного семейства B таким образом, чтобы каждый элемент из A присутствовал не более чем в одном множестве из B , при этом максимизировав число выбранных элементов t .

4.2.2 Решение задачи

Не было зафиксировано какое-то конкретное решение, так как задача носила оптимизационный характер – участник, нашедший решение, максимизирующее t и получал максимальный балл. Поэтому в решении данной задачи были использованы классические методы оптимизации – метод локальной оптимизации, метод имитации отжига, а в некоторых случаях – полный перебор.

Участнику предлагалось самому проанализировать входной набор тестов и заметить, что тесты, которые можно было гарантированно решить с помощью тривиального полного перебора за $\mathcal{O}(n2^m)$ составляли 30% тестов задачи, остальные тесты с достаточно большими ограничениями – остальные 70%

4.3 Задача №3

Третья задача, как и вторая носила оптимизационный характер и имела открытые тесты, хотя авторское решение этой задачи предполагало нахождение полного ответа, гарантированного корректного и оптимального.

Задача была предложена участникам заключительного этапа «Московской олимпиады школьников» для учащихся 10-11 классов. Раунд проходил на платформе Яндекс.Контест и длился 4 часа. Изначально запланированный как оффлайн-турнир с точкой проведения в Москве, его в связи с ограничительными эпидемиологическими мерами пришлось провести онлайн с системой прокторинга.

4.3.1 Условие задачи

Sokoban (Soko-Ban, яп. — «кладовщик») — логическая игра-головоломка, в которой игрок передвигает ящики по лабиринту, показанному в виде плана, с целью поставить все ящики на заданные конечные позиции.

Для этого пользователь перемещает человека, которого мы называем Сокобан. Сокобан может двигаться вверх, вниз, влево и вправо. Он не может проходить сквозь стены или ящики. Он может толкать только одну коробку за раз (никогда не тянуть). В любое время квадрат может быть занят только одной стеной, коробкой или Сокобаном.

Более формально, вам изначально известная конфигурация лабиринта — поля $n \times m$, состоящая из пустых клеток или стен. Также вам известна начальная позиция каждого из ящиков, конечные позиции, куда надо поставить ящики и начальное положение Сокобана.

Участнику предлагалось найти кратчайший способ решить головоломку — найти наиболее короткий способ поставить все ящики на позиции.

4.3.2 Решение задачи

Предполагалось, что эта задача требует умение пользоваться базовыми алгоритмами на графах и умение быстро писать код. Авторское решение использует алгоритм BFS (поиск в глубину) — рассматривается граф, состоящий из всевозможных конфигураций (конфигурация — это совокупность положений ящика и сокобана), где ребро ставится из вершины A в вершину B тогда и только тогда, когда из конфигурации A при шаге сокобана в одном из направлений, лабиринт перейдёт в состояние B .

Затем можно было воспользоваться общими идеями о данной игре — например, если ящик стоит в углу, то он останется там стоять и данная конфигурация не приведёт к решению и её можно выбросить.

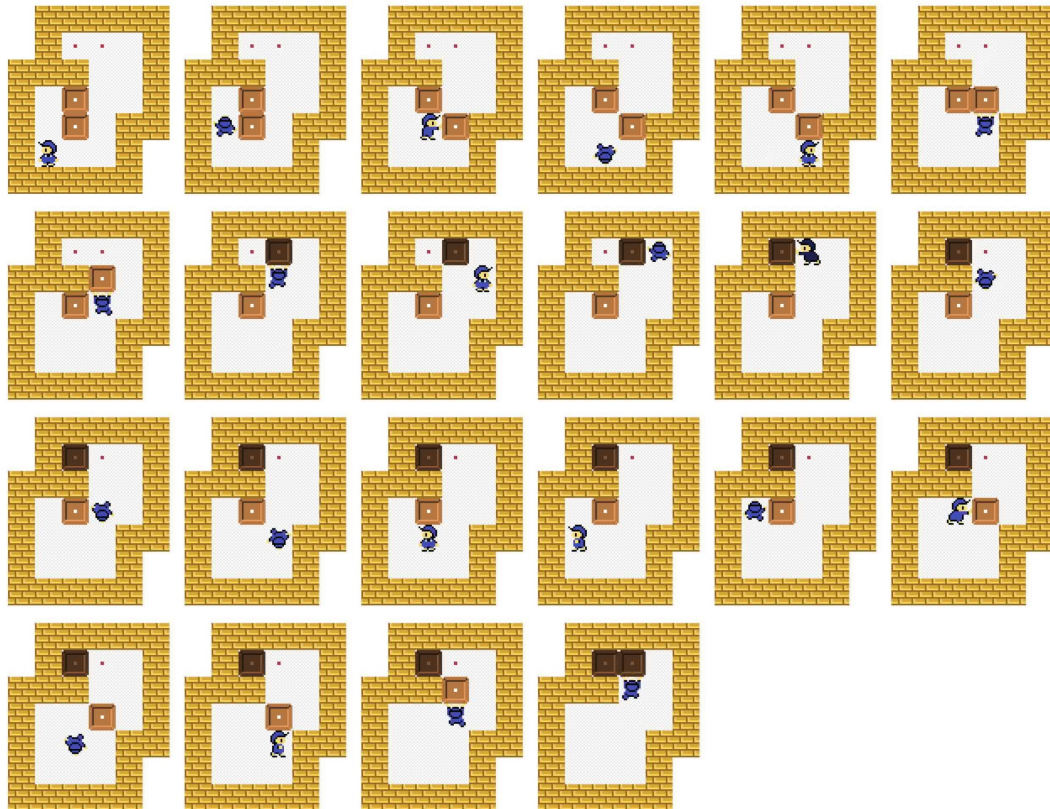
Такое решение, аккуратно написанное набирало 30% баллов.

Решением на 70% была оптимизация написанного выше с использованием оптимизации BFS, известной также как метод meet-in-the-middle — вместо запуска из стартовой конфигурации алгоритм запускается одновременно из стартовой и конечной конфигурации и два поиска работают параллельно, затем,

если они обе найдут кратчайший путь до какой-то вершины v , то будет найден кратчайший путь до вершины v из стартовой вершины и кратчайший путь до конечной вершины из вершины v , а значит, будет найден кратчайший путь из стартовой в конечную вершину.

Авторское решение работало около 8 минут и потребляло 1.5Гб оперативной памяти — предполагалось, что участники напишут решения, время работы которых займёт не более 30 минут и не более 3Гб оперативной памяти.

Ниже приведён пример решения головоломки Сокобан для лучшего понимания формулировки задачи



5 Реализация проекта

Все исходные файлы можно найти в github-репозитории <https://github.com/DebNatk/hse-project/>.

5.1 Задача №1

В данной задаче было написано решение, работающее за $\mathcal{O}(nk)$, описанное выше, а также были написана тривиальные валидаторы и чекеры – сравнивающие найденные ответы и проверяющие найденные ответы на валидность. Также был написан генератор случайных тестов `gen`.

Он принимает 4 аргумента N , K , $type$, M , где N , K соответствуют условию, а $type$ указывает, будут ли все элементы матрицы равны между собой, где M — максимально допустимое число в матрице. Генератор был запущен 30 раз для генерации 30 различных тестов:

```
gen 5 0 random
gen 5 3 random
gen 15 10 random
gen 20 5 random 300
gen 20 1 random
gen 20 3 random
gen 20 10 random
gen 20 13 random
gen 20 15 random
gen 20 15 random 1
gen 20 17 random 1
gen 20 19 random 300
gen 20 20 random
gen 20 20 random 300
gen 20 20 equal
gen 10000 5 random
gen 1000 30 random
gen 1000 200 random
gen 1000 200 random 5
gen 1000 200 equal
gen 2000 50 random
gen 2000 75 random
gen 2000 100 random
gen 5000 20 random
gen 5000 40 random
gen 100000 2 random
gen 50000 4 random
gen 50000 4 random 4
gen 50000 4 equal
gen 100000 1 random
```

5.2 Задача №2

Как сказано выше, было написано несколько решений от полного перебора до алгоритма имитации отжига, каждый из которых максимизировал целевой функционал. Для генерации тестов была использована написанная утилита `gen`, принимающая три параметра n , m , k , где n — число элементов, m — число множеств, а размер каждого из множеств выбирался из случайного распределения $\mathcal{N}(k, \text{rand})$, где rand — случайное целое число от 1 до 10.

Было сделано 6 тестов, решающихся с помощью полного перебора:

`gen 5 5 3`

`gen 10 10 5`

`gen 20 15 5`

`gen 100 20 5`

`gen 100 20 10`

`gen 100 20 15`

и 14 тестов, где необходимо было использовать методы оптимизации:

`gen 20 20 4`

`gen 20 40 6`

`gen 20 60 8`

`gen 20 100 10`

`gen 20 100 10`

`gen 24 100 10`

`gen 10000 10 12`

`gen 10000 11 40`

`gen 1000 30 10`

`gen 1000 60 10`

`gen 1000 60 2`

`gen 9999 100 78`

`gen 10000 100 30`

Формула $5 \times \left(\frac{\text{ParticipantSolution}}{\text{BestSolution}} \right)^3$ была выбрана в качестве формулы оценки за каждый из тестов, где `ParticipantSolution` — суммарное число занятых аудиторий в решении участника, а `BestSolution` — суммарное число занятых аудиторий в лучшем среди участников и жюри решении. При тестировании этой формулы и различных частичных решений использование именно такой формулы показалось целесообразным и справедливым в оценивании.

5.3 Задача №3

Для проверки задачи был написан код, который эмулировал логику игры sokoban и на каждое решение участника выдавал вердикт, верно ли был решён каждый из тестов.




Для решения задачи был написан код, реализующий BFS с meet-in-the-middle, затем для генерации тестов были выбраны уже известные головоломки сокобан, собранные в разные архивы, таких тестов было порядка 3 тысяч. Затем авторское решение было протестировано на каждом из них, замеряя время работы обычного BFS, двухстороннего BFS, а также объём потребляемой памяти. После этого в первую группу тестов (15 штук) были отобраны тесты, суммарное время работы на которых обычного BFS не превосходило 2 минут. Во вторую группу тестов (25 штук) были отобраны тесты, суммарное время работы полного решения на которых не превосходило 8 минут, и при этом решение с использованием обычного BFS не работало в связи с ограничениями по памяти или времени.

Формула $2 \times \left(\frac{\text{BestSolution}}{\text{ParticipantSolution}} \right)^4$ была выбрана в качестве формулы оценки за каждый из тестов, где ParticipantSolution — длина кратчайшего решения, найденного участником, а BestSolution — длина кратчайшего решения, найденного среди всех решений участников и жюри. При тестировании этой формулы и различных частичных решений использование именно такой формулы показалось целесообразным и справедливым в оценивании.

6 Заключение

Могу считать проект выполненным успешно – все задачи были предложены нескольким сотням участников. Например, последнюю задачу пытались, хоть и безуспешно, решить 45 участников, в решении второй задаче было предпринято свыше 1900 посылок. Все олимпиады состоялись без каких-либо значимых форс-мажоров, что считаю критерием удачно проделанной работы.

Ниже привожу фрагмент турнирной таблицы отборочного этапа «Московской олимпиады школьников», в которой задача D — это вторая приготовленная задача.

№	Участник	Я	0	A1	A2	B1	B2	C1	C2	D1	D2	E1	E2	Очки
		Ява2908	2843/2843	2649/2649	2189/2189	1126/1126	474/474	1499/1499	698/698	1127/1127	795/795	669/669	151/151	
208-210	nmachkova@mail.ru		0.00 2д. 15ч.	30.00 14д. 20ч.	70.00 14д. 20ч.	30.00 15д. 14ч.	70.00 15д. 18ч.	—	—	30.00 2м. 2д.	70.00 2м. 2д.	—	—	300.00
208-210	sanessansovic@gmail.com		—	30.00 27д. 19ч.	70.00 27д. 19ч.	30.00 1м. 12д.	70.00 1м. 21д.	—	—	30.00 1м. 26д.	70.00 2м. 2д.	—	—	300.00
208-210	mihail.Legend01		0.00 1м. 21д.	30.00 1м. 21д.	70.00 1м. 21д.	30.00 2м. 2д.	70.00 2м. 2д.	—	—	30.00 2м. 2д.	70.00 2м. 3д.	—	—	300.00
211	Артеми́й Казаков		0.00 10д. 16ч.	30.00 10д. 17ч.	70.00 10д. 17ч.	30.00 1м. 27д.	—	30.00 2м. 3д.	50.01 2м. 4д.	30.00 2м. 3д.	58.80 2м. 3д.	—	—	298.81
212-214	galtsev.mishail@gmail.com		0.00 1д. 15ч.	30.00 1д. 18ч.	70.00 1д. 18ч.	30.00 1м. 7д.	0.00 28д. 15ч.	30.00 1м. 12д.	61.79 1м. 14д.	30.00 1м. 14д.	46.73 1м. 21д.	—	—	298.52
212-214	mmaxim5555@gmail.com		0.00 1д. 16ч.	30.00 1д. 17ч.	70.00 1д. 17ч.	30.00 28д. 2ч.	0.00 1м. 7д.	30.00 1м. 18д.	61.79 1м. 18д.	30.00 1м. 14д.	46.73 2м. 20ч.	0.00 2м. 4д.	—	298.52
212-214	aleksey.co2017		0.00 1д. 20ч.	30.00 24д. 22ч.	70.00 24д. 23ч.	30.00 1м. 1д.	0.00 28д. 22ч.	30.00 1м. 13д.	61.79 1м. 14д.	30.00 1м. 13д.	46.73 2м. 2д.	0.00 2м. 4д.	—	298.52
215	naumovgm@mail.ru		0.00 1м. 19д.	30.00 1м. 19д.	70.00 1м. 19д.	30.00 1м. 21д.	—	30.00 2м. 3д.	70.00 2м. 3д.	30.00 2м. 4д.	27.84 2м. 4д.	10.00 2м. 4д.	—	297.84
216	онотолe		0.00 1м. 19д.	30.00 1м. 19д.	70.00 1м. 19д.	30.00 2м. 3д.	—	26.83 2м. 4д.	53.29 2м. 3д.	30.00 2м. 3д.	56.15 2м. 3д.	—	—	296.27
217	yurets04		0.00 11д. 17ч.	30.00 11д. 18ч.	70.00 11д. 18ч.	30.00 13д. 16ч.	—	30.00 2м. 1д.	64.23 2м. 1д.	30.00 1м. 23д.	41.48 2м. 2д.	—	—	295.71
218	Анна Алексеева		0.00 3д. 12ч.	30.00 3д. 13ч.	70.00 3д. 13ч.	30.00 1м. 10д.	70.00 1м. 15д.	—	—	30.00 1м. 22д.	65.14 1м. 22д.	—	—	295.14
219	lyect		0.00 1м. 4д.	30.00 1м. 7д.	70.00 1м. 7д.	30.00 1м. 4д.	70.00 1м. 7д.	—	—	30.00 2м. 20ч.	64.89 2м. 20ч.	—	—	294.89
220	kian2701		0.00 14д. 15ч.	30.00 14д. 15ч.	70.00 14д. 16ч.	30.00 26д. 18ч.	70.00 28д. 21ч.	30.00 1м. 12д.	34.77 1м. 25д.	30.00 1м. 14д.	—	—	—	294.77
221	Николай Плотников		0.00 1д. 15ч.	30.00 1д. 16ч.	70.00 1д. 16ч.	30.00 10д.	70.00 10д. 1ч.	0.00 2м. 4д.	—	30.00 1м. 21д.	64.56 2м. 4д.	—	—	294.56
222	ivang.27.09.03@gmail.com		0.00 12д. 18ч.	30.00 12д. 19ч.	70.00 12д. 19ч.	30.00 2м. 3д.	—	30.00 2м. 3д.	51.45 2м. 3д.	30.00 2м. 3д.	52.54 2м. 4д.	—	—	293.99

Список литературы

- [1] Яндекс.Контеcт / Yandex.Contest, <https://contest.yandex.ru/about/>
- [2] Codeforces Polygon beta, <https://polygon.codeforces.com/>
- [3] Codeforces, <https://codeforces.com>
- [4] CodeChief, <https://codechief.com>
- [5] Ejudge, <https://ejudge.ru/>
- [6] Архивы прошлых лет всероссийской олимпиады школьников «Высшая проба» <https://olymp.hse.ru/mmo/materials-it>
- [7] Timus (Online Judge <https://acm.timus.ru/?locale=ru>
- [8] Сайт дистанционной подготовки к олимпиадам по программированию <https://informatics.mccme.ru/>
- [9] Yandex.Algorithm <https://contest.yandex.com/algorithm2017/>
- [10] TopCoder <https://www.topcoder.com/community/competitive-programming/>
- [11] Официальная страница Всероссийской олимпиады школьников по информатике (РОИ) <http://rosoi.net/2020/>
- [12] Google Code Jam <https://codingcompetitions.withgoogle.com/codejam>
- [13] Московская олимпиада школьников по информатике <http://mos-inf.olimpiada.ru/>