УДК 378:37.016:004
ББК 74.580

ЗАДАЧИ СПОРТИВНОГО ПРОГРАММИРОВАНИЯ ДЛЯ ДИСЦИПЛИНЫ «МАШИННОЕ ОБУЧЕНИЕ»

А. А. Никандров[1)], **К. Р. Пиотровская**[2)]

[1),2)] Российский Государственный педагогический университет им. А.И. Герцена, наб. р. Мойки, 48, 191186, Санкт-Петербург
[1)]nik190397@mail.ru, [2)]krp62@mail.ru

Обсуждается опыт создания и решения задач по программированию на языке Python 3 в рамках дисциплины «Машинное обучение» для бакалавров 3-го курса факультета математики в РГПУ им. А.И. Герцена, обучающихся по направлению «Прикладная математика и информатика». Разобраны примеры задач трех типов: жадный алгоритм, комбинаторика и сортировки.

Ключевые слова: олимпиадное программирование; жадный алгоритм; комбинаторика; сортировка; машинное обучение.

COMPETITIVE PROGRAMMING PROBLEMS FOR MACHINE LEARNING COURSE

Alexey Nikandrov[1)], **Xenia Piotrowska**[2)]

[1),2)] The Herzen State Pedagogical University of Russia, Moika Embankment, 48, 191186, Russia, [1)]nik190397@mail.ru, [2)]krp62@mail.ru

The creating and solving problems in Python 3 experience is discussed as part of the discipline "Machine Learning" for 3rd year bachelors of Math Department at the Herzen University. Examples of three types are analyzed: greedy algorithm, combinatorics and sorting.

Keywords: competitive programming; greedy algorithm; combinatorics; sorting; machine learning.

Введение

В данной статье представлен опыт отбора, создания и размещения задач по программированию в рамках дисциплины «Машинное обучение», которая в последние годы изучается студентами 3-го курса факультета математики в РГПУ им. А.И. Герцена направления «Прикладная ма-

тематика и информатика». Обсуждаемый в статье курс отличается возможностью выстраивания гибкой образовательной траектории для каждого студента благодаря задействованию современных средств: интерактивной вычислительной платформы Jupyter notebook для оформления и проведения лекций, которая позволяет проводить занятия по машинному обучению в современном формате, и многофункциональной и гибкой платформы для создания образовательных материалов Stepik – для добавления и автоматической проверки практических заданий [1, 2, 8]. Целесообразность применения описываемого нами подхода заключается в возможности расширения преподавательского банка задач за счет обращения к платформе спортивного программирования Codeforces. Все лекции записывались и выкладывались на платформе Stepik для повторного изучения студентами.

Обучающимся было предложено несколько вариантов развития собственной образовательной траектории. Студент имел возможность выбрать варианты обучения:

1) базовый вариант, выполнение практических заданий которого опиралось на лекции,

2) продвинутый вариант, включающий задачи на алгоритмы и структуры данных,

3) индивидуальное задание по машинному обучению, согласованное с преподавателем.

В течение семестра предусматривалась возможность вернуться к базовому варианту, опирающемуся на материал лекций, если студент осознавал свою безрезультативность.

Все задачи являются авторской разработкой и создавались с учетом междисциплинарного подхода, что позволило опираться на знания учащихся не только в области информатики, но и в области математики.

Отбор и анализ решений задач спортивного программирования

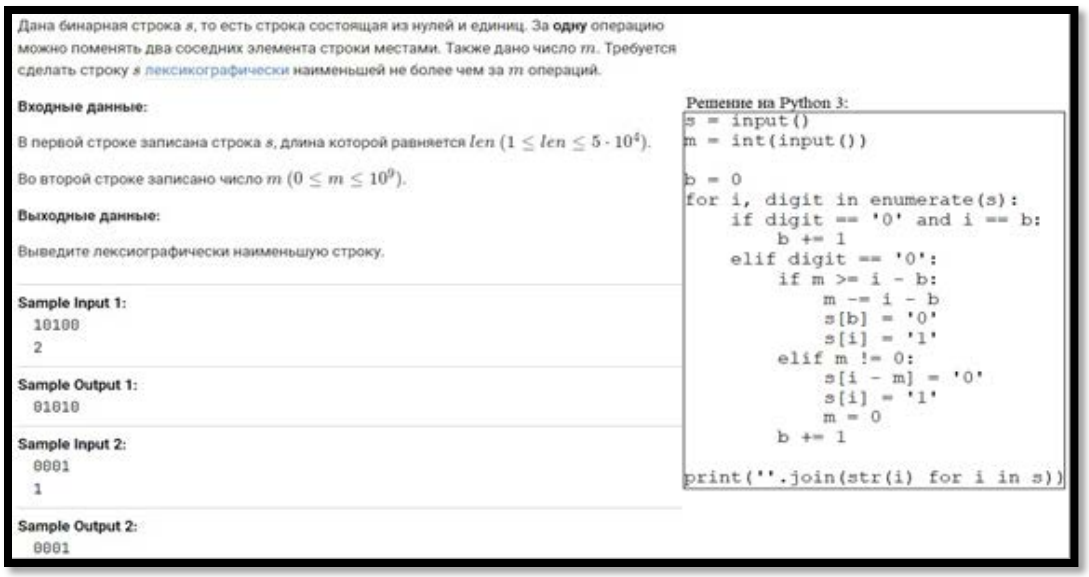
Оценка сложности предлагаемых в курсе задач проводилась следующим образом. Известная платформа Codeforces, созданная М. Мирзаяновым при поддержке Санкт-Петербургского университета информационных технологий ИТМО [3], размещает еженедельные соревнования (контесты) по спортивному программированию. Эта платформа широко известна профессионалам в области спортивного программирования по всему миру, что доказано присутствием именитых участников [4], а также её использованием при подготовке команд к олимпиадам по программированию [5].

На платформе Codeforces располагаются материалы в соответствии со следующей метрикой оценки сложности задач, согласно которой самые лёгкие задачи оцениваются в 800 условных единиц, а далее происходит прибавление по 100 условных единиц по мере усложнения до 3500 условных единиц сложности. При построении задач авторы ориентировались на материалы ресурса в пределах уровней сложности от 1200 до 1600, т.е. на умеренный уровень.

Рассмотрим семь задач из трех типов: жадный алгоритм, комбинаторика и сортировки. Разделение по типам достаточно условно, поскольку возможны пересечения и некоторые задачи могут быть отнесены одновременно к нескольким типам. Все рисунки построены единообразно: продемонстрировано условие задачи, входные и выходные данные, протокол вывода, справа программный код на языке Python 3. Для каждой задачи приводим ссылку на платформу Stepik, где можно посмотреть формулировку условия и комментарии к решению.

Рассмотрим первые две задачи на жадный алгоритм.

Идея решения задачи (https://stepik.org/lesson/671750/step/3?unit=669986), представленной на рис. 1, заключается в проходе по всем элементам строки s слева направо.



Дана бинарная строка s, то есть строка состоящая из нулей и единиц. За **одну** операцию можно поменять два соседних элемента строки местами. Также дано число m. Требуется сделать строку s лексикографически наименьшей не более чем за m операций.

Входные данные:

В первой строке записана строка s, длина которой равняется len ($1 \le len \le 5 \cdot 10^5$).

Во второй строке записано число m ($0 \le m \le 10^9$).

Выходные данные:

Выведите лексиографически наименьшую строку.

Sample Input 1:
10100
2

Sample Output 1:
01010

Sample Input 2:
0001
1

Sample Output 2:
0001

Решение на Python 3:

```
s = input()
m = int(input())

b = 0
for i, digit in enumerate(s):
    if digit == '0' and i == b:
        b += 1
    elif digit == '0':
        if m >= i - b:
            m -= i - b
            s[b] = '0'
            s[i] = '1'
        elif m != 0:
            s[i - m] = '0'
            s[i] = '1'
            m = 0
        b += 1

print(''.join(str(i) for i in s))
```

Рис 1. Задача 1

Если встретился ноль, то его, если позволяет значение переменной m меняем с самой крайней левой единицей. Когда значение m не позволяет совершить замену встреченного нуля с крайней левой единицей, то осуществляется замена на ту левую единицу, до которой позволяет это сделать оставшееся значение переменной m.

Вторая задача (https://stepik.org/lesson/675346/step/1?unit=673782) решается с использованием префиксных сумм, но с обязательным присутствием нуля в самом начале. Такой подход позволяет выявить подмассивы, суммы которых равны нулю. Это происходит в связи с тем, что если в префиксных суммах существуют хотя бы два одинаковых числа, то сумма подмассива, включающего эти два одинаковых числа, равна нулю. Поэтому, для решения этой задачи достаточно вставлять достаточно большое число, например, 10^{10}, на место второго одинакового числа. Следовательно, второе одинаковое число абстрактно сместится (это не прописывается в программном коде) на одну позицию вправо. Тем самым сумма подмассива уже не будет равна нулю. Далее повторяем описанный выше алгоритм, но уже с позиции второго одинакового числа.

Следующая группа задач (задачи 3-5) относится к комбинаторной математике, одна задача из этой группы представлена на рис. 2.

Дан массив l, состоящий из натуральных чисел $b_1, b_2, \ldots, b_n$. Назовём произвольный массив **изящным**, если найдётся такое число $const$, что каждое число в массиве содержится ровно 0 **или** $const$ раз.

Например, массивы $[1, 2, 2, 1]$, $[1, 2, 3]$, $[\,]$ являются изящными, но массив $[1, 2, 3, 3]$ - нет.

Вы можете удалять числа из массива l. Необходимо сделать массив l изящным за наименьшее количество удалений.

Входные данные:

В первой строке дано число n $(1 \leq n \leq 5 \cdot 10^4)$ - количество элементов в массиве.

Во второй строке записан массив l, заданный целыми числами $b_1, b_2, \ldots, b_n$ $(1 \leq b_i \leq 5 \cdot 10^4)$.

Выходные данные:

Выведите одно число - минимально возможное количество удалений.

Sample Input 1:

```
8
4 2 4 3 2 2 3 3
```

Sample Output 1:

```
2
```

Sample Input 2:

```
8
3 1 2 3 3 4 3 3
```

Sample Output 2:

```
3
```

Решение на Python 3:

```
from collections import Counter
from collections import defaultdict

n = int(input())
l = [
      int(num)
      for num in input().split()
]

frequency = Counter(l)

quantity = defaultdict(int)

for num in frequency:
    quantity[frequency[num]] += 1

quantity = {
    k: quantity[k]
    for k in sorted(quantity)
}

dels = float('inf')
m = len(quantity)
for k in quantity:
    l_del = 0
    q_copy = quantity.copy()
    for lk in q_copy:
        l_del += lk * q_copy[lk] \
            if lk < k else \
            (lk - k) * q_copy[lk]

    dels = min(dels, l_del)

print(dels)
```

Рис. 2. Задача 3

Для успешного решения задачи 4 (https://stepik.org/lesson/682434/step/ 2?unit=681248) достаточно знания всего одного из чисел исходной перестановки a, чтобы восстановить её целиком. Для этого находим позицию наибольшего числа исходной перестановки, что позволит однозначно определить крайнее левой значе-

ние. Для восстановления всей перестановки будем придерживаться условия задачи: $b_i = a_{i+1} - a_i$. Однако стоит учесть, что последнее возможно не всегда, поэтому присутствует проверка функции $check$.

Идея решения задачи 5 (https://stepik.org/lesson/682434/step/3?unit=681248) заключается в том, чтобы первым действием был проведен поиск наибольшего общего делителя для каждой пары с последующим делением на него. Это поможет в случае пар $(12, 16)$ и $(6, 8)$, где после деления будет соответственно $(3, 4)$ и $(3, 4)$, то есть для таких пар подойдёт одинаковое k. Стоит заметить, подойдёт аналогичное k для пары $(-3, -4)$. Наконец, для формирования итогового ответа нельзя забыть про пары $(0, 0)$.

В заключении рассмотрим две задачи на сортировку.

Для решения задачи 6, продемонстрированной на рис. 3 (https://stepik.org/lesson/671750/step/2?unit=669986), существует два подхода. Оба требуют сортировки и отличаются направлением прохода. Обратим внимание на способ справа налево, поэтому применяется процедура сортировки по убыванию: $a.sort(reverse = True)$.

Вы зашли в магазин игрушек. Каждая игрушка имеет свою цену (цены могут совпадать). Вы хотите купить максимальное количество игрушек при условии, что все цены попарно будут различны. Продавец обрадовался такому подходу и дал возможность **уменьшить** или **увеличить** на **единицу** стоимость отдельно взятой игрушки, но стоимость не должна быть равна **нулю.**

Надо узнать, какое максимальное количество игрушек можно купить.

Входные данные:

В первой строке записано число n $(1 \le n \le 5 \cdot 10^4)$.

Во второй строке записано n целых чисел $b_1, b_2, \ldots, b_n$ $(1 \le b_i \le 10^8)$.

Выходные данные:

Выведите единственное число - максимальное количество игрушек.

Sample Input 1:

```
4
1 1 3 3
```

Sample Output 1:

```
4
```

Sample Input 2:

```
3
7 8 6
```

Sample Output 2:

```
3
```

Решение на Python 3:

```
n = int(input())
a = [
      int(num)
      for num in input().split()
]

a.sort(reverse=True)
boxers = set()
for ai in a:
    if ai + 1 not in boxers:
        boxers.add(ai + 1)
    elif ai not in boxers:
        boxers.add(ai)
    elif ai - 1 != 0:
        boxers.add(ai - 1)

print(len(boxers))
```

Рис. 3. Задача 6

На первой позиции массива a находится наибольшая стоимость игрушки, которую можно увеличить, оставить без изменений или умень-

шить. В таком порядке нужно проводить проверку для получения окончательного ответа. Пытаемся добавить $a_i + 1$, a_i или $a_i - 1$ во множество $boxers$ на каждой итерации цикла в зависимости от их отсутствия в нём.

Общая идея (https://stepik.org/lesson/680868/step/3?unit=679557) решения задачи 7 состоит в том, что можно не использовать перебор отрицательных чисел в силу использования выражений по модулю (следует из условия). Далее осуществляется сортировка неотрицательных чисел с дальнейшей передачей результата в функцию $func$. Есть один момент, связанный с применением бинарного поиска, на который обратим внимание. Допустим, рассматривается некое число num в отсортированном массиве неотрицательных чисел. Из-за границ отрезка $|b_i - b_j|, |b_i + b_j|$ видно, нет смысла выходить за пределы $2 * num$, потому что это только сузит его, также необходимо учесть все отрезки, состоящие из одинаковых чисел (за что отвечает строка $c[num] * (c[num] - 1)\ //\ 2$).

Результаты и их обсуждение

Все задачи были опробованы в двух студенческих группах факультета математики РГПУ им. А.И. Герцена по направлению «Прикладная математика и информатика» в рамках дисциплины «Машинное обучение» в 2021-22 и 2022-23 учебных годах с наполняемостью групп 18 и 7 человек соответственно [7]. В таблице 1 приведена статистика решения обсуждаемых задач, выработанная платформой Stepik. Стоит отметить, что ни один студент не справился со всеми заданиями. Это можно объяснить тем, что существовала возможность сменить траекторию обучения, а следовательно, студенты шли по пути меньшего сопротивления и занимались более простыми заданиями.

Статистика успешности при решении задач

№ задачи	Количество студентов, предоставивших правильное решение	Количество правильных решений в % от общего числа попыток
1	8	67
2	5	93
3	5	80
4	1	100
5	1	100
6	8	92
7	1	100

На будущее запланирована корректировка, связанная с усложнением простых заданий с целью повышения интереса к вышеописанным задачам. Также будут добавлены новые типы задач и пополнены уже существующие.

Библиографические ссылки

1. Stepik [Электронный ресурс]. – URL: https://stepik.org/. (дата обращения: 29.03.2023).

2. *Johnson J. W.* Benefits and pitfalls of Jupyter Notebooks in the classroom: proceedings of the 21st Annual Conference on Information Technology Education, 2020 Oct 7. PP. 32-37 [Электронный ресурс]. URL: https://doi.org/10.1145/3368308.3415397

3. Codeforces [Электронный ресурс]. – Режим доступа: https://codeforces.com (дата обращения: 29.03.2023).

4. *Kostka B.* Sports programming in practice. Wroclaw: University of Wroclaw, 2021.

5. *Корабельщикова С. Ю., Толкачева Е. А., Бутин К. П.* О системе подготовки команды к олимпиадам по программированию. / С. Ю. Корабельщикова, Е. А. Толкачев, К.П. Бутин // Современные информационные технологии и ИТ-образование. – 2019. – С. 164-173.

6. *Фихтенгольц Г. М.* Курс дифференциального и интегрального исчисления. Т.1. М.: Физматлит, 2001.

7. *Никандров, А. А., Пиотровская К. Р.* Анализ образовательных данных дисциплины "Основы математической обработки информации" : Проблемы теории и практики обучения математике: сборник научных работ, представленных на Международную научную конференцию «73 Герценовские чтения», Санкт-Петербург, 21–25 апреля 2020 года. – Санкт-Петербург: Российский государственный педагогический университет им. А. И. Герцена, 2020. – С. 91-97.

8. *Никандров, А. А.* Использование платформы Stepik при обучении программированию в рамках дисциплины «Машинное обучение» // Современные проблемы математики и математического образования : сборник научных статей международной научной конференции к 225-летию Герценовского университета, Санкт-Петербург, 04–06 июня 2022 года / Под редакцией В.В. Орлова и М.Я. Якубсона. Санкт-Петербург: Российский государственный педагогический университет им. А. И. Герцена, 2022. – С. 159-162.