МАТЕРИАЛЫ ЗАДАНИЙ

Всероссийской междисциплинарной олимпиады школьников «Национальная технологическая олимпиада»

по профилю «Искусственный интеллект»

2021/22 учебный год

http://ntcontest.ru

Оглавление

Профиль «Искусственный интеллект»

Цель профиля «Искусственный интеллект» — развитие у школьников прикладных навыков в сфере искусственного интеллекта. Профиль знакомит школьников с методами и технологиями, применяемыми для решения актуальных для науки и бизнеса задач в сфере искусственного интеллекта.

В 2021/22 учебном году участникам НТО по профилю «Искусственный интеллект» было предложено погрузиться в задачу по созданию алгоритмов распознавания рукописного текста — это задача на стыке нескольких AI-технологий (компьютерное зрение, обработка естественного языка).

Основная задача отборочных этапов — выявить наиболее способных к решению подобных задач школьников и через образовательную составляющую развить у школьников прикладные навыки в области машинного обучения.

Первый отборочный дистанционный этап (индивидуальный) определял уровень подготовки школьников по предметам: математика и информатика. Задачи по информатике относятся к разделам: алгоритмы, программирование и методы оптимизации. Решая их, школьники должны были продемонстрировать простейшие навыки составления и отладки программ, обрабатывающих массивы данных, и понимание таких тем, как комбинаторика, операции со строками, матричный анализ, теория графов. Количество попыток при решении задач на программирование не ограничивалось. Задачи по математике проверяли у участников знания по алгебре, комбинаторике, теории вероятности и математической статистике. Таким образом, задачи первого этапа выявляли наличие у участников знаний, необходимых для решения задач последующих этапов.

В рамках **второго отборочного этапа** участникам было предложено решить две задачи, каждая из которых представляла собой один из этапов работы алгоритма, способного распознать рукописный текст на фотографии.

Первая задача (сегментация) решалась участниками индивидуально и предполагала создание алгоритма, способного определить, относится пиксель на фото к рукописному тексту или нет. Для работы участникам были предоставлены фотографии школьных тетрадей. Понимание решения данной задачи было необходимым условием для части решения задачи заключительного этапа.

Вторая задача (распознавание) решалась участниками в командах до 2 человек. Участникам были предоставлены сегментированные строки с рукописным текстом, их задачей была разработка алгоритма, способного определить содержание текста, т. е., что там написано.

Сочетание оценки индивидуальных и командных решений позволило выявить наиболее мотивированных и талантливых участников, а также убедиться, что необходимые компетенции развиты у каждого участника команды на должном уровне.

В целях подготовки финалистов к заключительному этапу дополнительно был проведен образовательный хакатон, где финалисты могли отточить необходимые для решения задачи заключительного этапа компетенции, а участники, не прошедшие в заключительный этап, — попробовать себя в решении задач более высокого уровня (хакатон.академияии.рф).

Заключительный этап НТО по профилю «Искусственный интеллект» состоял из командного практического и индивидуального предметного туров.

Предметный тур проводился по двум предметам, выбранных профилем: информатика и математика. Предметный тур формирует предметные знания, необходимые в решении задач на анализ данных. Баллы, набранные в индивидуальном предметном туре, т. е. знания и умения решать олимпиадные задачи в важных для направления искусственный интеллект разделах математики и информатики, влияют на индивидуальный результат участников.

Задача командного практического тура заключительного этапа предполагала создание алгоритма, который принимает на вход фото страницы с рукописным текстом и выдает распознанный рукописный текст. Качественное распознавание самых разных почерков является актуальной задачей для EdTech-сферы, в первую очередь в отношении школьного образования. Без ее решения невозможна автоматизация проверки тетрадей и проверочных работ, а значит и:

- снятие рутинной нагрузки с педагогов;
- персонализация образовательного процесса;
- глубокая аналитика прогресса отдельного ученика, группы, класса, параллели или даже всех школьников.

По сравнению с предыдущими этапами, данные для обучения алгоритмов задачи заключительного этапа были насыщены изображениями, содержащими не только русский язык, но и английский, что делало модель билингвальной. А также был полностью обновлен проверочный набор данных, в котором помимо английского языка значительно увеличилось количество почерков. Ни один из почерков не пересекался с наборами данных для обучения. Все это позволило выделить по-настоящему универсальные решения, которые способны распознавать самые разные почерки.

Таким образом учащиеся 8–11 классов, прошедшие все этапы НТО, демонстрируют понимание основных операций, необходимых для построения модели машинного обучения: подготовка и анализ данных с целью выявления признаков, необходимых для решения поставленной задачи, построение самой модели и интерпретация результатов ее работы для оценки качества. Для того чтобы сделать это возможным, в ходе олимпиады проводился цикл образовательных и отборочных мероприятий. С момента регистрации для подготовки участникам были доступны:

- бесплатный онлайн-курс по машинному обучению от Академии искусственного интеллекта для школьников: https://ml.ai-academy.ru/;
- задачи по машинному обучению прошлых лет для отработки навыков: https://ai-academy.ru/training/training-tasks/.

В ноябре 2021 года для участников дополнительно был проведен образовательный онлайн-интенсив, в рамках которого были разобраны темы от основ программирования на языке Python с разбором профильных библиотек для анализа данных и машинного обучения до популярных архитектур нейронных сетей, необходимых для качественного решения конкурсных задач: https://practicingfutures.org/autumn_bootcamp.

Первый отборочный этап

Задачи первого этапа. Информатика

Первая попытка. Задачи 8–11 класса

Задача I.1.1.1. Расчет скидки (20 баллов)

Темы: задачи для начинающих, простая математика.

Одна продуктовая сеть в рамках акции выдает скидочные купоны двух видов. По первому купону можно получить скидку в 8% от стоимости покупки, но не более 100 рублей. По второму купону можно получить скидку в 5% от стоимости покупки без других ограничений. Предъявлять можно только один купон, разделять покупку на части нельзя. Покупатель делает покупку на p рублей. У него есть оба купона. Напишите программу, которая вычислит максимальный размер скидки, которую покупатель сможет получить.

Формат входных данных

На вход подается одно целое число — размер покупки в рублях. Число не превосходит 10000.

Формат выходных данных

Вывести одно число — размер скидки в рублях. Ответ может оказаться не целым.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
`810`

Стандартный вывод
`64.8`

Пример №2

Стандартный ввод
1530
Стандартный вывод
100

Пример №3

Стандартный ввод
10000
Стандартный вывод
500

Решение

В этой задаче требуется рассмотреть два варианта. Обозначим сумму покупки за p.

При использовании купона со скидкой в 8% сумма скидки является минимумом из 100 и $0,08p$. При использовании купона со скидкой в 5% сумма скидки будет равна $0,05p$.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 p=int(input())
2 print(max(p*0.05,min(p*0.08,100)))
```

Задача I.1.1.2. Произведение многочленов (20 баллов)

Темы: реализация, простая математика.

Многочлены — это одни из самых распространенных математических объектов, которые используются практически во всех прикладных областях. Задан многочлен $a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$. От вас требуется написать программу, которая найдет произведение этого многочлена на $x+1$. Многочлен задан своими коэффициентами a_n, a_{n-1}, …, a_2, a_1, a_0. Обратите внимание, что многочлен степени n состоит из $n+1$ одночлена. Некоторые из одночленов могут отсутствовать. В этом случае соответствующий коэффициент считается равным нулю.

Например, многочлен $2x^3+3x^2+1$ будет задан набором коэффициентов 2 3 0 1. Результатом умножения будет многочлен четвертой степени с набором коэффициентов 2 5 3 1 1, что можно проверить, раскрыв скобки.

$$(2x^3 + 3x^2 + 1)(x + 1) = 2x^4 + 3x^3 + x + 2x^3 + 3x^2 + 1 = 2x^4 + 5x^3 + 3x^2 + x + 1$$

Формат входных данных

На вход программы в первой строке подается одно натуральное число n — степень многочлена. $1 \leqslant n \leqslant 100$. Далее во второй строке через пробел подается $n + 1$ целое число — коэффициенты многочлена a_n, a_{n-1}, …, a_2, a_1, a_0. Каждый из коэффициентов не превосходит 1000 по абсолютной величине. $a_n \neq 0$.

Формат выходных данных

Требуется вывести через пробел $n + 2$ коэффициента полученного многочлена.

Если вы программируете на Python, то убрать перенос строки в функции `print` можно при помощи именованного параметра `end`, например, `print(a,end=' ')`.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
3 2 3 0 1

Стандартный вывод
2 5 3 1 1

Решение

Найдем произведение многочлена $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0$ и $x + 1$.

$$\begin{gathered}(a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0)(x + 1) = \\ = (a_n x^{n+1} + a_{n-1} x^n + \cdots + a_2 x^3 + a_1 x^2 + a_0 x) + \\ + (a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0) = \\ = a_n x^{n+1} + (a_{n-1} + a_n) x^n + \cdots + (a_1 + a_2) x^2 + (a_0 + a_1) x + a_0\end{gathered}$$

Таким образом каждый коэффициент, кроме первого и последнего, является суммой двух соседних элементов исходного списка.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1  input()
2  x=map(int,input().split())
3  prev=int(next(iter(x)))
4  print(prev,end=' ')
```

```
for t in x:
    print(t+prev,end=' ')
    prev=t;
print(prev)
```

Ниже представлено решение в функциональном стиле на языке Python 3.

```
x=list(map(int,input().split()))
print(*[a+b for (a,b) in zip([0]+x,x+[0])])
```

Задача I.1.1.3. Очередь (20 баллов)

Темы: структуры данных.

Студенческая группа сдает зачет преподавателю. Он расположил студентов по некоторому неизвестному порядку и сообщил, кто после кого сдает зачет. Теперь студенты хотят выяснить, в каком порядке им приходить, чтобы не стоять всей группой за дверью.

Формат входных данных

На вход программы в первой строке подается одно натуральное число n — количество студентов в группе. $2 \leqslant n \leqslant 30$. Далее в $n-1$ строке через пробел подается по два имени: имя студента, сдающего зачет, и имя того студента, который будет сдавать перед ним. Имена не содержат пробелов и состоят только из строчных и прописных символов латиницы. Гарантируется, что очередь задана корректно, имена студентов в группе не повторяются.

Формат выходных данных

Требуется вывести имена всех студентов группы в том порядке, в котором они сдают зачет. Каждое имя выводится в отдельной строке.

Методика проверки и пояснение к тесту

В приведенном примере в группе 5 студентов. Первой сдает зачет Лиза, поскольку только для нее не указано, кто приходит раньше. После Лизы приходит Иван, далее Мария, затем Петр и последним сдает Игорь.

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
5 Petr Mariya Ivan Liza Mariya Ivan Igor Petr

Стандартный вывод
Liza Ivan Mariya Petr Igor

Решение

В этой задаче требуется продемонстрировать умение работать со структурами данных.

Наиболее простым способом решения будет создание ассоциативного массива (словаря), в котором для каждого студента будет указано имя следующего в очереди.

Дополнительной сложностью является нахождение имени первого студента. Для этого надо найти имя, которое есть среди ключей, но которого нет среди значений словаря. Такая операция может быть записана как разность множества ключей и множества значений словаря. По условию такая разность будет содержать только один элемент, который и будет искомым именем.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n=int(input())
nxt=dict()
for i in range(n-1):
    val,key=input().split()
    nxt[key]=val
cur=next(iter(nxt.keys()-nxt.values()))
print(cur)
for i in range(n-1):
    cur=nxt[cur]
    print(cur)
```

Задача I.1.1.4. Четырехугольник (20 баллов)

Темы: геометрия, неравенство треугольника, перебор, реализация.

Сергею на уроке геометрии задали следующее задание. Даны 5 чисел. Требуется нарисовать произвольный четырехугольник с одной диагональю так, чтобы длины

сторон и этой диагонали равнялись заданным числам. Сергею надо выбрать длину диагонали и каждую из сторон так, чтобы было возможно нарисовать требуемую фигуру. Если вариантов решения задачи несколько, можно выбрать любой. Нарисованная диагональ не должна лежать на одной из сторон. Возможно, что нарисовать требуемый четырехугольник не получится. В этом случае надо будет вывести ноль.

Формат входных данных

На вход через пробел подаются 5 натуральных чисел от 1 до 1000.

Формат выходных данных

Требуется вывести ответ в следующем порядке. В первой строке вывести число — длину диагонали четырехугольника. Во второй строке 2 числа — длины отрезков, лежащих с одной стороны от диагонали. В третьей строке еще 2 числа — длины отрезков, лежащих с другой стороны от диагонали. Если построить четырехугольник невозможно, то вывести 0.

В этой задаче можно выводить любой правильный ответ. В частности, можно переставить числа в одной строке или поменять местами вторую и третью строку.

Методика проверки и пояснение к тестам

В первом тесте в качестве диагонали можно взять отрезок длины 7. Тогда с одной стороны от диагонали будут стороны с длинами 3 и 5, а с другой — 9 и 3. Вторую и третью строку, а также числа в этих строках можно вывести в любом порядке. Также возможно нарисовать четырехугольник с диагональю 5 и длинами сторон 9, 7 и 3, 3. Кроме того, возможен вариант с диагональю 3 и длинами сторон 5, 3 и 7, 9 Любой из этих вариантов будет считаться верным.

Во втором тесте нарисовать четырехугольник невозможно. Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
`3 9 5 3 7`

Стандартный вывод

`7`
`3 5`
`9 3`

Пример №2

Стандартный ввод
`3 9 5 1 7`

Стандартный вывод
`0`

Решение

Для решения этой задачи требуется знать неравенство треугольника, которое говорит о том, что в невырожденном треугольнике сумма длин двух любых сторон больше, чем длина третьей. Четырехугольник с одной диагональю можно представить как два треугольника, смежных по одной стороне.

Решение должно заключаться в переборе нескольких вариантов. Этот перебор можно организовать разными способами. Один из способов заключается в использовании функций для генерации перестановок. В Python это функция `permutations` из модуля `itertools`. Можно получить все перестановки пяти заданных чисел и проверить их по неравенству треугольника.

Чтобы получить другой способ решения, можно заметить, что две самых длинных стороны в любом случае выгоднее использовать при построении одного треугольника. Тогда количество вариантов сокращается до трех.

Пример программы-решения

Ниже представлено решение на языке Python 3 с полным перебором.

```
from itertools import permutations
def triangle(a,b,c):
    return a<b+c and b<a+c and c<a+b
x=map(int,input().split())
for t in list(permutations(x)):
    if triangle(t[0],t[1],t[2]) and triangle(t[0],t[3],t[4]):
        print(t[0])
        print(t[1],t[2])
        print(t[3],t[4])
        break
else:
    print(0)
```

Ниже представлено решение на языке Python 3 с сортировкой.

```
x=sorted(list(map(int,input().split())),reverse=True)
if x[0]<x[1]+x[2] and x[2]<x[3]+x[4]:
    print(x[2])
    print(x[0],x[1])
    print(x[3],x[4])
elif x[0]<x[1]+x[3] and x[1]<x[2]+x[4]:
    print(x[1])
    print(x[0],x[3])
    print(x[2],x[4])
elif x[0]<x[1]+x[4] and x[1]<x[2]+x[3]:
    print(x[1])
    print(x[0],x[4])
    print(x[2],x[3])
else:
    print(0)
```

Задача I.1.1.5. Рыцари (20 баллов)

Темы: структуры данных, реализация, алгоритмическая сложность.

В королевстве Логрес за круглым столом собираются рыцари. Каждый рыцарь гордится своими победами, поэтому делает на своих доспехах царапины по количеству побежденных противников. Ранг рыцаря определяется по количеству царапин. Рыцари весьма горды и тот, чей ранг ниже, должен оказывать почтение тому, чей ранг выше. Если вдруг возникает ситуация, что подряд сидят рыцари с одинаковым рангом, то они устраивают между собой турнир, по итогам которого определяется один победитель. Он ставит на доспехи новые царапины (если в турнире участвовало k рыцарей, то победитель поставит $k-1$ новую царапину) и возвращается за стол на свое место. Остальные участники турнира уходят залечивать раны и уязвленное самолюбие. Если после этого вновь возникает аналогичная ситуация, то проводится новый турнир, и так далее.

За круглым столом собралось n рыцарей, и они предусмотрительно расселись так, чтобы ранги соседей были различными. Но опоздавший Галахад все испортил. Он сел на случайно выбранное место, и карусель турниров снова закрутилась.

Известны ранги всех n рыцарей, изначально сидевших за столом. Также известен ранг Галахада и место, на которое он сел. Напишите программу для определения количества рыцарей, которые останутся за столом после того, как все турниры завершатся.

Формат входных данных

В первой строке на вход подается число n — количество рыцарей без учета Галахада. $1 \leqslant n \leqslant 300000$. Во второй строке через пробел записаны n натуральных чисел $r_1, \ldots, r_n$ — ранги всех рыцарей. $r_i \leqslant 10^6$; $r_i \neq r_{i+1}$; $r_1 \neq r_n$. В третьей строке через пробел записаны два натуральных числа p и t. $1 \leqslant p \leqslant n$; $t \leqslant 10^6$. Число p задает место, на которое сел Галахад и означает, что он оказался между рыцарями с номерами p и $p+1$. Если $p = n$, то Галахад находится между первым и последним рыцарем. Число t задает исходный ранг Галахада.

Формат выходных данных

Требуется вывести одно число — ответ к задаче.

Методика проверки и пояснение к тесту

После того, как Галахад сядет за стол после второго рыцаря, ранги рыцарей за столом будут (7 5 5 6 8 9 10 12 10 8). Два рыцаря с рангом 5 проведут турнир, останется один из них, ранг которого повысится до 6 (7 6 6 8 9 10 12 10 8). Из двух рыцарей с рангом 6 вновь останется 1 с рангом 7 (7 7 8 9 10 12 10 8). После очередного турнира получим (8 8 9 10 12 10 8). Помня, что стол круглый, видим 3 рыцарей с рангом 8, сидящих подряд. Из них останется один с рангом 10 (10 9 10 12 10). Будет проведен еще один турнир, после которого оставшиеся 4 рыцаря не будут иметь соседей с одинаковым рангом (11 9 10 12).

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
9 7 5 6 8 9 10 12 10 8 2 5

Стандартный вывод
4

Решение

В этой задаче требовалось придумать достаточно простой в реализации способ решения, который при этом удовлетворял бы требованию по времени работы программы. Для этого необходимо, чтобы элементы не удалялись из массива, так как эта операция достаточно затратная по времени. Дополнительной сложностью является то, что последовательность элементов должна быть циклической.

Заметим, что все изменения могут происходить вокруг позиции Галахада, поэтому будет удобно, если его ранг не будет храниться в последовательности, а номера рыцарей слева от него будут начинаться с нуля. Тогда справа от Галахада будет последний элемент последовательности. Такого состояния можно добиться при помощи циклического сдвига всей последовательности.

Далее будем использовать метод двух указателей. Номер рыцаря слева от Галахада будем хранить в переменной i, а номер правого рыцаря — в переменной j. Вместо удаления элементов будем просто смещать значения этих переменных.

Пример программы-решения

Ниже представлено решение на языке Python 3.

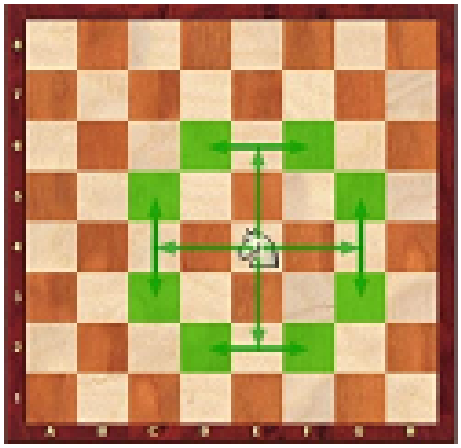
```
n=int(input())
lst=list(map(int,input().split()))
p,r=map(int,input().split())
lst=lst[p:]+lst[:p]
i=0
j=n-1
while i<j:
    if lst[i]==lst[j]==r:
        r+=2
        i+=1
        j-=1
    elif lst[i]==r:
        r+=1
        i+=1
    elif lst[j]==r:
        r+=1
        j-=1
    else:
        break
if i==j and lst[i]==r:
    i+=1
print(j-i+2)
```

Вторая попытка. Задачи 8–11 класса

Задача I.1.2.1. Конь (20 баллов)

Темы: задачи для начинающих, перебор.

Шахматный конь стоит на доске размером 8×8 в i-той строке и j-том столбце. Напишите программу, которая определит, сколько ходов он может сделать.



Конь ходит, как показано на рисунке. Из центральной части доски он может сделать 8 ходов, но если конь находится ближе к краю доски, то количество ходов уменьшится, так как он не может выйти за ее границы.

Формат входных данных

На вход подается два натуральных числа в диапазоне от 1 до 8 — номер клетки, в которой находится конь, по горизонтали и вертикали. Каждое число записано в отдельной строке.

Формат выходных данных

Вывести одно число — количество возможных ходов коня.

Методика проверки

Программа проверяется на 40 тестах. Прохождение каждого теста оценивается в 0,5 балла. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
5 3

Стандартный вывод
8

Решение

В предложенном решении этой задачи перебираются все клетки шахматной доски, и выполняется проверка того, что конь может сходить в эту клетку. Возможны и другие решения.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
a=int(input())
b=int(input())
ans=0
for i in range(1,9):
    for j in range(1,9):
         if a!=i and b!=j and abs(a-i)+abs(b-j)==3:
             ans+=1
print(ans)
```

Задача I.1.2.2. Королевство чисел (20 баллов)

Темы: задачи для начинающих, простая математика.

Алиса и Боб стали королями в королевствах на множестве натуральных чисел. Подданными Алисы являются все натуральные числа, которые делятся на 3 без остатка, а все остальные числа стали подданными Боба. Алиса дружит с Бобом, и они хотят, чтобы их подданные тоже дружили между собой. Они разбили все числа на пары, причем i-тое по порядку число из королевства Алисы будет дружить с i-тым по порядку числом из королевства Боба. Вам задан набор из n чисел. Напишите программу для нахождения друга каждого из чисел.

Первые 10 чисел из королевства Алисы — это $\{3, 6, 9, 12, 15, 18, 21, 24, 27, 30, \ldots\}$. Первые 10 чисел из королевства Боба — это $\{1, 2, 4, 5, 7, 8, 10, 11, 13, 14, \ldots\}$. Таким образом, парами друзей являются $(3, 1)$ $(6, 2)$ $(9, 4)$ и так далее.

Формат входных данных

На вход в первой строке подается натуральное число n — количество чисел в наборе. $1 \leqslant n \leqslant 10^5$. Во второй строке через пробел подается n натуральных чисел $a_1, a_2, \ldots, a_n$. Числа не превосходят 10^{18}. Обратите внимание, что для хранения таких чисел в программе на C++ вам потребуется тип `long long`. В программе на PascalABC такой тип называется `Int64`.

Формат выходных данных

Программа должна вывести через пробел n натуральных чисел $b_1, b_2, \ldots, b_n$. Число b_i должно быть другом числа a_i.

Если вы программируете на Python, то заменить перенос строки на пробел в функции `print` можно при помощи именованного параметра `end`, например `print(a,end=' ')`.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется. В первых 5 тестах $n \leqslant 10$, $a_i \leqslant 1000$. В следующих пяти тестах $n \leqslant 10^5$, $a_i \leqslant 10^6$. В последних 10 тестах $a_i \leqslant 10^{18}$.

Примеры

Пример №1

Стандартный ввод
10 1 2 3 4 5 6 7 8 9 10
Стандартный вывод
3 6 1 9 12 2 15 18 4 21

Решение

Можно заметить, что соответствующие числа в различных множествах различаются примерно в два раза. Поэтому требуемые формулы можно получить, умножая или деля заданное число на 2. В зависимости от четности к результату надо будет прибавить некоторую константу. С использованием свойств целочисленной арифметики программу можно упростить.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n=int(input())
for x in map(int,input().split()):
    if x%3!=0:
        print(3*(x-x//3),end=' ')
    else:
        print((x-1)//2,end=' ')
```

Задача I.1.2.3. Алфавит (20 баллов)

Темы: строки, символы, структуры данных.

Панграммой называется строка, в которой присутствуют все буквы алфавита. Однако при этом строка может содержать пробелы. У Алисы есть строка, состоящая из строчных и заглавных символов латиницы. Она хочет убедиться, что эта строка является панграммой. Алиса действует следующим образом. Для начала она удаляет из строки все пробелы, а также заменяет все заглавные буквы на строчные. Далее она вырезает из строки символы и составляет из них вторую строку по такому алгоритму. Она перебирает последовательно все буквы алфавита от «**a**» до «**z**» и пытается найти самое первое вхождение этой буквы в строке. Если вхождение нашлось, то Алиса вырезает из строки эту букву и добавляет ее справа ко второй строке и переходит с следующей букве.

Например, из строки «**A Bra Kada Bra**» будет получена строка «**abrakadabra**», далее из нее последовательно будут вырезаны самые первые вхождения букв **a**, **b**, **d**, **k**, **r**, после чего она превратится в строку **aaabra**.

Вы должны написать программу, которая определит содержимое обеих строк после преобразований Алисы.

Формат входных данных

На вход подается одна строка, содержащая не более 200 строчных и заглавных символов латиницы и пробелов. Гарантируется, что хотя бы один из символов латиницы входит в строку несколько раз.

Формат выходных данных

Программа должна вывести обе полученные строки без пробелов.

Методика проверки

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
`A  Bra    Kada   Bra`

Стандартный вывод
`aaabra` `abdkr`

Решение

В этой задаче проверяется знание некоторых функций Python для работы со строками. В частности, используются: метод `replace()` для удаления всех пробелов, метод `lower()` для смены регистра, метод `join()` для объединения символов в строку. Для хранения уникальных символов в приведенном решении используются множества.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
unique=set()
other=''
for x in input().replace(' ','').lower():
    if x in unique:
        other+=x
```

```
6       else:
7           unique.add(x)
8   print(other)
9   print(''.join(sorted(list(unique))))
```

Задача I.1.2.4. Велосипедисты (20 баллов)

Темы: математика, уравнения, двоичный поиск.

Два велосипедиста выехали одновременно из пункта A по одной дороге с различными скоростями u и v метров в секунду. Через t секунд им вдогонку выехал электромобиль и через некоторое время обогнал одного, а затем и другого велосипедиста. При этом интервал между моментами обгона составил d секунд.

Вы должны написать программу, которая вычислит скорость движения электромобиля.

Формат входных данных

На вход через пробел подаются четыре натуральных числа: u, v, t, d. При этом $u \neq v$; $u, v \leqslant 50$; $t, d \leqslant 10000$. Гарантируется, что введенные данные будут таковы, что ответ не превысит 200.

Формат выходных данных

Программа должна вывести одно вещественное число — скорость электромобиля.

Методика проверки и пояснение к тесту

Ответ участника считается верным, если он отличается от ответа жюри не более чем на 10^{-8}.

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. При этом в первых 5 тестах ответ обязательно будет целым числом. Тест из условия задачи при проверке не используется.

Рассмотрим тест из примера. Утверждается, что для заданных параметров ответом является 12. Проверим это. Можно вычислить, что электромобиль, двигаясь со скоростью 12 м/с, обгонит более медленного велосипедиста на расстоянии 480 метров, а более быстрого на расстоянии в 1200 метров. Действительно, электромобиль преодолеет 480 и 1200 метров за 40 и 100 секунд соответственно. Таким образом, интервал между моментами обгона действительно равен 60. Велосипедисты до моментов обгона будут двигаться на 20 секунд дольше, по 60 и 120 секунд соответственно. И, проверив пройденное расстояние $60 \cdot 8 = 480$ и $120 \cdot 10 = 1200$, убедимся, что ответ верен. **Обратите внимание, что это пояснение лишь показывает, как проверить правильность ответа, но не является алгоритмом решения**.

Примеры

Пример №1

Стандартный ввод
10 8 20 60

Стандартный вывод
12.0

Решение

Обозначим скорость электромобиля за x. Тогда до старта электромобиля велосипедист проехал tu метров. Электромобиль догнал велосипедиста через $\frac{tu}{x-u}$ секунд. Предполагая, что второй велосипедист двигался быстрее чем первый, получим уравнение:

$$\frac{tv}{x-v} - \frac{tu}{x-u} = d$$

Уравнение можно решить аналитически, а также тернарным или бинарным поиском, учитывая, что с ростом x интервал между моментами обгона будет сокращаться.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
u,v,t,d=map(int,input().split())
if u>v:
    u,v=v,u
d/=t
b=(d+1)*v+(d-1)*u
print((b+(b*b-4*d*d*u*v)**0.5)/(2*d))
```

Ниже представлено решение бинарным поиском на языке Python 3.

```
u,v,t,d=map(int,input().split())
if u>v:
    u,v=v,u
left=max(u,v)
right=200
for i in range(100):
    mid=(left+right)/2
    if t*v/(mid-v)-t*u/(mid-u)>d:
        left=mid
    else:
        right=mid
print(right)
```

Задача I.1.2.5. Много единиц (20 баллов)

Темы: системы счисления, битовые операции, реализация.

Алиса учится работать с двоичными числами. Она уже поняла, что число в двоичной записи получается в несколько раз длиннее, чем в десятичной. А еще она

поняла, что нули писать дольше, чем единицы. И теперь ее любимые числа — это те, двоичная запись которых содержит как можно больше единиц. Алисе дали задание — выбрать одно произвольное число из заданного закрытого интервала $[a; b]$ и перевести его в двоичную запись. И теперь Алиса просит, чтобы вы написали программу, которая найдет в этом интервале число, двоичная запись которого содержит наибольшее количество единиц. Если таких чисел будет несколько, то Алиса будет рада любому из них.

Формат входных данных

На вход через пробел подаются два натуральных числа a и b. При этом $1 \leqslant a \leqslant$ $\leqslant b \leqslant 10^{18}$. Обратите внимание, что для хранения таких чисел в программе на C++ вам потребуется тип **long long**. В программе на PascalABC такой тип называется **Int64**.

Формат выходных данных

Программа должна вывести одно целое число из заданного диапазона, двоичная запись которого содержит наибольшее количество единиц. Само число следует выводить в десятичной записи.

Методика проверки и пояснение к тесту

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. При этом в первых пяти тестах $1 \leqslant a \leqslant b \leqslant 1000$. Тесты из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
`150 200`
Стандартный вывод
`191`

Пример №2

Стандартный ввод
`1 255`
Стандартный вывод
`255`

Пример №3

Стандартный ввод
`127 200`
Стандартный вывод
`127`

Решение

Рассмотрим некоторое число в двоичной записи, которое содержит k единиц. Чтобы получить ближайшее сверху число в записи которого $k+1$ единица, требуется заменить на единицу самый правый ноль. Таким образом решение задачи состоит в том, чтобы взять число из начала интервала и заменять самые правые нули до тех пор, пока результат не превысит правую границу интервала.

Реализовать такой алгоритм можно, представляя число в виде строки, а также при помощи арифметических или битовых операций.

Пример программы-решения

Ниже представлено поиском на языке Python 3 с использованием строк.

```
a,b = map(int,input().split())
a=list(bin(a)[2:])
b=list(bin(b)[2:])
while len(a)<len(b):
    a=['0']+a
for i in range(len(a)-1,-1,-1):
    a[i]='1'
    if a>b:
        a[i]='0'
        break
print(int(''.join(a),2))
```

Ниже представлено поиском на языке Python 3 с использованием арифметических операций.

```
a,b = map(int,input().split())
p=1
while a+p<=b:
    if (a//p)%2==0:
        a+=p
    p*=2
print(a)
```

Ниже представлено поиском на языке Python 3 с использованием битовых операций.

```
a,b = map(int,input().split())
while a | (a+1) <= b:
    a = a | (a+1)
print(a)
```

Третья попытка. Задачи 8–11 класса

Задача I.1.3.1. Урожай (20 баллов)

Темы: задачи для начинающих.

Дядя Саша с сыном Колей копают картошку. Урожай выдался, как всегда, отменным, и они накопали n мешков. Дядя Саша пригнал грузовичок, в который может

поместиться не более a мешков картошки, а в Колин грузовичок поместится не более b мешков. Урожай они хотят поделить поровну. Если количество мешков не будет делится на 2, то лишний мешок на правах старшего заберет дядя Саша. Вместе с тем, никто не сможет забрать мешков больше, чем поместится в его грузовик. И, конечно же, они не оставят ни одного мешка на поле.

Напишите программу, которая определит, сколько мешков увезет дядя Саша, а сколько Коля.

Формат входных данных

На вход подаются натуральные числа n, a и b по одному числу в строке. Числа не превосходят 1000. Гарантируется, что $n \leqslant a + b$.

Формат выходных данных

Программа должна вывести в одной строке через пробел два числа — количество мешков, которое увезут дядя Саша и Коля на своих грузовичках.

Методика проверки и пояснение к тестам

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. Тесты из условия задачи при проверке не используются.

В первом тесте 59 мешков будут разделены почти поровну — 30 мешков дяде Саше и 29 — Коле. Во втором тесте Коля не сможет увезти причитающиеся ему 29 мешков и отдаст лишнее дяде Саше.

Примеры

Пример №1

Стандартный ввод
59 35 40
Стандартный вывод
30 29

Пример №2

Стандартный ввод
59 41 25
Стандартный вывод
34 25

Решение

Данную задачу можно решать различными способами. В предлагаемом варианте решения мешки сначала делятся поровну, а далее, в случае необходимости, перемещаются из одного грузовика в другой.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n=int(input())
a=int(input())
b=int(input())
x, y = (n+1)//2, n//2
if x>a:
    y+=x-a
    x=a
if y>b:
    x+=y-b
    y=b
print(x,y)
```

Задача I.1.3.2. Скайраннинг (20 баллов)

Темы: задачи для начинающих.

Скайраннингом называется бег в горной местности по неподготовленным трассам, которые обязательно проходят через одну или несколько вершин. Важной характеристикой маршрута в скайраннинге является набор высоты, который равен сумме перепада высот на всех участках подъема. Например, если на маршруте имеется три участка подъема, причем конечная точка каждого участка выше начальной на 200 метров, то набор высоты на маршруте будет равен 600 метров. Кроме того, весь маршрут может быть поделен на высотные зоны. В нашей задаче мы будем рассматривать две высотные зоны: ниже 2000 метров и выше 2000 метров. В этом случае все параметры, в том числе и набор высоты, рассчитываются для разных высотных зон.

Рассмотрим такой пример. Пусть профиль трассы содержит семь точек с высотами 1200, 2300, 2100, 2900, 3100, 1000, 1800 метров. На этой трассе есть четыре участка подъема: (1200; 2300), (2100; 2900), (2900; 3100), (1000;1800). На первом участке подъема 800 метров набирается в первой высотной зоне и 300 метров во второй. На втором и третьем участках набирается по 800 и 200 метров соответственно во второй высотной зоне. На четвертом участке набирается 800 метров в первой зоне. Таким образом в первой высотной зоне набирается 1600 метров, а во второй — 1300 метров.

Ваша задача — написать программу, которая по заданному профилю трассы найдет набор высоты для двух высотных зон: ниже 2000 и выше 2000 метров.

Формат входных данных

На вход в первой строке подается одно натуральное число n — количество точек в профиле высоты. $2 \leqslant n \leqslant 100$. Во второй строке через пробел записаны n целых чисел $a_1, \ldots, a_n$, задающих высоту каждой точки. $-416 \leqslant a_i \leqslant 8848$.

Формат выходных данных

Программа должна вывести в одной строке через пробел два числа — набор высоты для первой и второй высотной зоны.

Методика проверки

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
7 1200 2300 2100 2900 3100 1000 1800
Стандартный вывод
1600 1300

Пример №2

Стандартный ввод
5 2000 2675 2675 1215 -416
Стандартный вывод
0 675

Решение

В этой задаче необходимо перебрать все участки трассы, на которых происходит подъем, и рассмотреть три случая: весь участок ниже 2000 метров, весь участок выше 2000 метров, участок проходит через высоту в 2000 метров.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n = int(input())
h = list(map(int,input().split()))
a1,a2=0,0
for i in range(1,n):
    if h[i]>h[i-1]:
        if h[i-1]>=2000:
            a2+=h[i]-h[i-1]
        elif h[i]<=2000:
            a1+=h[i]-h[i-1]
        else:
            a1+=2000-h[i-1]
            a2+=h[i]-2000
print(a1,a2)
```

Задача I.1.3.3. Шоколадка (20 баллов)

Темы: перебор, сортировки, структуры данных.

У Аленки есть шоколадка прямоугольной формы, размером $n \times m$ долек. Аленка разламывает ее на две части по вертикали или по горизонтали и съедает одну из двух частей. Если Аленка чувствует, что не наелась, то она снова может разломить оставшийся кусок на две части и съесть одну из них, и так далее. Всю шоколадку Аленка есть не будет, а оставит про запас, как минимум, одну дольку.

Производителям стало известно о такой привычке девочки и они захотели узнать, какое количество долек может быть съедено при таком алгоритме поедания шоколада. Они хотят, чтобы вы написали программу, которая по известному размеру шоколадки найдет все возможные количества съеденных долек и выведет их в порядке возрастания.

Рассмотрим такой пример. Шоколадка имеет размер 3×3. Тогда Аленка может разломить ее на две части 3×1 и 3×2. Таким образом, она сможет съесть 3 или 6 долек и остановиться. Но также Аленка сможет съесть кусок из 6 долек, а от оставшегося отломить 1 или 2 дольки и съесть еще и их. Таким образом, она сможет съесть 7 или 8 долек. Наконец, она сможет съесть кусок 3×1, а от оставшегося куска 3×2 отломить и съесть еще 2 дольки, тогда количество съеденных долек будет равно 5. Очевидно, что съесть 1, 2 или 4 дольки Аленка не сможет.

Формат входных данных

На вход в одной строке подается два натуральных числа n и m — размеры шоколадки. $1 \leqslant n, m \leqslant 100$; $n + m \geqslant 3$.

Формат выходных данных

Программа должна вывести в одной строке через пробел все числа, являющиеся ответами к задаче. Числа должны выводиться в порядке возрастания без повторений.

Если вы программируете на Python, то убрать перенос строки в функции `print` можно при помощи именованного параметра `end`, например, `print(a,end=' ')`.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
`3 3`

Стандартный вывод
`3 5 6 7 8`

Решение

Заметим, что у Аленки всегда остается один кусок шоколадки прямоугольной формы. Поэтому можно перебрать все прямоугольные области меньшего размера и найти разность между количеством долек во всей шоколадке и в прямоугольниках. Полученные значения могут повторяться, поэтому потребуется отбросить одинаковые числа. Для этого можно использовать множество или булевский массив.

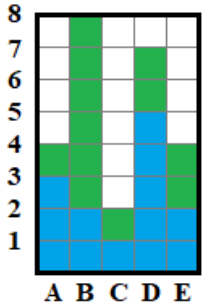
Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n, m = map(int,input().split())
k=[False]*(n*m)
for i in range(1,n+1):
    for j in range(1,m+1):
        k[n*m-i*j]=True
for i in range(1,n*m):
    if k[i]:
        print(i,end=' ')
```

Задача I.1.3.4. Стековая гистограмма (20 баллов)

Темы: реализация, структуры данных.



По определению Википедии, стековая гистограмма отображает зоны, представляющие свойства одной категории, друг поверх друга. Имеется n категорий с двумя свойствами. Требуется нарисовать стековую гистограмму без подписей в консоли с использованием символов «.*#», как показано в примере.

Формат входных данных

В первой строке на вход подается одно натуральное число n — количество категорий в гистограмме. $2 \leqslant n \leqslant 50$. Во второй строке через пробел записано n натуральных чисел $a_1, \ldots, a_n$ — значения первого свойства для каждой категории. В третьей строке аналогично записаны натуральные числа $b_1, \ldots, b_n$ — значения второго свойства. $1 \leqslant a_i, b_i \leqslant 20$.

Формат выходных данных

Требуется вывести в консоль текстовое представление гистограммы. Текст должен содержать ровно $\max(a_i + b_i)$ строк, каждая строка должна содержать ровно n символов и завершаться переводом строки. Если рассматривать текст по столбцам, то i-тый столбец должен содержать снизу ровно a_i решеток, далее b_i звездочек, а еще выше — точки.

Методика проверки и пояснение к тесту

Программа проверяется на 5 тестах. Прохождение каждого теста оценивается в 4 балла. Тест из условия задачи при проверке не используется.

На картинке изображена гистограмма из примера. Синие квадратики соответствуют решеткам, зеленые — звездочкам, белые — точкам. Количество строк равно высоте самого высокого столбика.

Примеры

Пример №1

Стандартный ввод
5 3 2 1 5 2 1 6 1 2 2

Стандартный вывод
.*... .*.*. .*.*. .*.#. **.#* #*.#* ##*## #####

Решение

Для решения этой задачи можно создать двумерный символьный массив и заполнить его требуемыми символами. Строки надо будет выводить на экран в обратном порядке. Можно обойтись и без двумерного массива. Для вывода можно использовать два вложенных цикла и определять вид символа по номеру строки и столбца. Чтобы определить количество строк, потребуется найти максимум из сумм значений по категориям.

Пример программы-решения

Ниже представлено решение на языке Python 3 с двумерным массивом.

```
1  n=int(input())
2  a=list(map(int,input().split()))
```

```
b=list(map(int,input().split()))
h=max([a[i]+b[i] for i in range(n)])
s=[['.']*n for i in range(h)]
for i in range(n):
    for j in range(a[i]):
        s[j][i]='#'
    for j in range(a[i],a[i]+b[i]):
        s[j][i]='*'
for x in reversed(s):
    print(''.join(x))
```

Ниже представлено решение на языке Python 3 без двумерного массива.

```
n=int(input())
a=list(map(int,input().split()))
b=list(map(int,input().split()))
h=max([a[i]+b[i] for i in range(n)])
for j in range(h):
    for i in range(n):
        if h-j <= a[i]:
            print('#',end='')
        elif h-j <= a[i]+b[i]:
            print('*',end='')
        else:
            print('.',end='')
    print()
```

Задача I.1.3.5. Префиксный код (20 баллов)

Темы: математика, строки, перебор, структуры данных.

Префиксное кодирование является очень распространенным способом сжатия информации для ее хранения и передачи. Каждый символ кодируется некоторым кодовым словом — строкой из нулей и единиц. Кодовые слова могут иметь различную длину, но при этом должно выполняться следующее свойство: ни одно кодовое слово не начинается с другого кодового слова. Например, множество кодовых слов $\{00, 011, 1000, 11\}$ подходит для префиксного кодирования, а $\{00, 101, 11, 1010\}$ — нет, так как 101 является началом 1010.

От вас требуется написать программу, которая найдет, какое максимальное количество кодовых слов заданной длины m можно добавить к некоторому уже построенному множеству слов для префиксного кодирования. Например $m = 4$, и уже построено множество $\{10, 001, 000, 0111, 11111, 11000, 111101\}$. Без нарушения условия префиксного кодирования можно добавить следующие слова длины 4: $\{0100, 0101, 0110, 1101, 1110\}$. Таким образом, ответ равен 5.

Формат входных данных

В первой строке на вход подается два натуральных числа n и m — количество уже известных кодовых слов и длина новых кодовых слов. $m \leqslant 60$. Во второй строке через пробел записано n кодовых слов. Все слова состоят из нулей и единиц и удовлетворяют условию префиксного кодирования. Суммарная длина всех слов не превосходит 10^6.

Формат выходных данных

Требуется вывести одно целое число — ответ к задаче. Ответ может быть равным нулю.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. При этом в 3 тестах $m \leqslant 10$ и $n \leqslant 100$, еще в 3 тестах длина всех заданных слов не превосходит m, еще в 4 тестах длина всех заданных слов не меньше чем m. Тест из условия задачи при проверке не используется.

Примеры

Пример №1

Стандартный ввод
7 4 10 001 000 0111 11111 11000 111101
Стандартный вывод
5

Решение

Сразу отметим, что существует 2^m различных кодовых слов длины m. Каждое кодовое слово, ранее добавленное в код, не позволит использовать некоторое количество возможных слов длины m. Так, если некоторое уже добавленное слово α имеет длину k, меньшую чем m, то любое слово, полученное приписыванием к α справа $m - k$ символов, будет недоступно. Существует 2^{m-k} способов дописать справа к α некоторую последовательность из нулей и единиц, поэтому такую величину надо будет вычесть из ответа.

Если кодовое слово имеет длину большую m, то его префикс длины m также нельзя использовать. Однако различные слова могут иметь одинаковые префиксы, поэтому среди них надо найти все различные, например, при помощи множеств.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n,m=map(int,input().split())
ans=2**m
wset=set()
for x in input().split():
    if len(x)>m:
        wset.add(x[:m])
    else:
        ans-=2**(m-len(x))
print(ans-len(wset))
```

Четвертая попытка. Задачи 8–11 класса

Задача I.1.4.1. Кросс нации (20 баллов)

Темы: математика, задачи для начинающих.

Некоторые измеряют скорость в километрах, пройденных за час. А вот люди, занимающиеся бегом, измеряют темп бега в секундах на километр, то есть указывают количество минут и секунд, за которые они пробегают ровно 1 километр. Например, скорость в 12,5 км/ч соответствует темпу бега в 4 минуты 48 секунд.

Ваша задача — написать программу, которая определит темп бега по скорости.

Формат входных данных

На вход подается единственное число u — скорость в километрах за час. Скорость является вещественным положительным числом не более чем с двумя знаками после точки. $5 \leqslant u \leqslant 50$.

Формат выходных данных

Вывести через пробел два целых числа — время в минутах и секундах, за которое бегун пробегает 1 километр. Секунды должны быть округлены до ближайшего целого числа по стандартным правилам.

Методика проверки

Программа проверяется на 20 тестах. Прохождение каждого теста оценивается в 1 балл. Тесты из условия задачи при проверке не используются.

Примеры

Пример №1

Стандартный ввод
`12.5`
Стандартный вывод
`4 48`

Пример №2

Стандартный ввод
`15.03`
Стандартный вывод
`4 0`

Пример №3

Стандартный ввод
15.04
Стандартный вывод
3 59

Решение

В одном часе 3600 секунд. Поэтому, если бегун пробегает за 3600 секунд a километров, то один километр он пробежит за $\frac{3600}{a}$ секунд. Это число надо округлить и привести к целому типу, далее выделить из него минуты и секунды, используя деление и остаток от деления на 60.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1 a=float(input())
2 a=int(round(3600/a))
3 print(a//60,s%60)
```

Задача I.1.4.2. Говорящий конь (20 баллов)

Темы: задачи для начинающих, реализация.

Говорящий конь Юлий шел по дороге и встретил трех богатырей.

— Здравствуй, богатырь номер 1,— сказал говорящий конь Юлий.

— Здравствуй, говорящий конь Юлий,— сказал богатырь номер 1.

— Здравствуй, богатырь номер 2,— сказал говорящий конь Юлий.

— Здравствуй, говорящий конь Юлий,— сказал богатырь номер 2.

— Здравствуй, богатырь номер 3,— сказал говорящий конь Юлий.

— Здравствуй, говорящий конь Юлий,— сказал богатырь номер 3.

— Здравствуй, лошадь богатыря номер 1,— сказал говорящий конь Юлий.

— Здравствуй, говорящий конь Юлий,— сказала лошадь богатыря номер 1.

— Здравствуй, лошадь богатыря номер 2,— сказал говорящий конь Юлий.

— Здравствуй, говорящий конь Юлий,— сказала лошадь богатыря номер 2.

— Здравствуй, лошадь богатыря номер 3,— сказал говорящий конь Юлий.

— Здравствуй, говорящий конь Юлий,— сказала лошадь богатыря номер 3.

— До свиданья, богатырь номер 1,— сказал говорящий конь Юлий.

— До свиданья, говорящий конь Юлий,— сказал богатырь номер 1.

— До свиданья, богатырь номер 2,— сказал говорящий конь Юлий.

— До свиданья, говорящий конь Юлий,— сказал богатырь номер 2.

— До свиданья, богатырь номер 3,— сказал говорящий конь Юлий.

— До свиданья, говорящий конь Юлий,— сказал богатырь номер 3.

— До свиданья, лошадь богатыря номер 1,— сказал говорящий конь Юлий.

— До свиданья, говорящий конь Юлий,— сказала лошадь богатыря номер 1.

— До свиданья, лошадь богатыря номер 2,— сказал говорящий конь Юлий.

— До свиданья, говорящий конь Юлий,— сказала лошадь богатыря номер 2.

— До свиданья, лошадь богатыря номер 3,— сказал говорящий конь Юлий.

— До свиданья, говорящий конь Юлий,— сказала лошадь богатыря номер 3.

И говорящий конь Юлий пошел по дороге дальше.

Маленькому Ванечке очень нравится эта сказка и он любит слушать ее в разных вариантах. Ему особенно нравится вариант, в котором говорящий конь Юлий встречает на дороге 40 разбойников, но почему-то усталые взрослые не хотят ее рассказывать.

Напишите программу, которая по номеру предложения сказки о встрече говорящего коня Юлия с 40 разбойниками будет выводить это предложение. Поскольку родители считают, что Ванечка с самого юного возраста должен изучать иностранные языки, от вас требуется вывести ответ по-английски.

Формат входных данных

Одно натуральное число n — номер предложения сказки. $1 \leqslant n \leqslant 322$.

Формат выходных данных

Вывести требуемое предложение сказки в одной строке. Слова должны быть разделены ровно одним пробелом. Перед знаками препинания пробел не ставится. Перевод всех предложений и формат вывода смотрите в примерах.

Методика проверки

Программа проверяется на 50 тестах. Прохождение каждого теста оценивается в 0,4 балла. **Первые 10 контрольных тестов совпадают с тестами из условия задачи**.

Примеры

Пример №1

Стандартный ввод
`1`

Стандартный вывод
`Talking horse Julius was walking along the road and met 40 robbers.`

Пример №2

Стандартный ввод
2
Стандартный вывод
- Hello, robber number 1,- said talking horse Julius.

Пример №3

Стандартный ввод
3
Стандартный вывод
- Hello, talking horse Julius,- said robber number 1.

Пример №4

Стандартный ввод
82
Стандартный вывод
- Hello, horse of robber number 1,- said talking horse Julius.

Пример №5

Стандартный ввод
83
Стандартный вывод
- Hello, talking horse Julius,- said horse of robber number 1.

Пример №6

Стандартный ввод
162
Стандартный вывод
- Goodbye, robber number 1,- said talking horse Julius.

Пример №7

Стандартный ввод
163
Стандартный вывод
- Goodbye, talking horse Julius,- said robber number 1.

Пример №8

Стандартный ввод
242
Стандартный вывод
- Goodbye, horse of robber number 1,- said talking horse Julius.

Пример №9

Стандартный ввод
243
Стандартный вывод
`- Goodbye, talking horse Julius,- said horse of robber number 1.`

Пример №10

Стандартный ввод
322
Стандартный вывод
`And talking horse Julius went on along the road.`

Решение

Текст сказки состоит из 10 различных предложений. Каждое предложение встречается в определенном диапазоне номеров на четных или нечетных позициях. Исходя из этого, можно записать программу через одну инструкцию ветвления с 10 вариантами работы.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n=int(input())
if n==1:
    print('Talking horse Julius was walking along the road and met 40 robbers.')
elif n<82 and n%2==0:
    print('- Hello, robber number '+str(n//2)+',- said talking horse Julius.')
elif n<82:
    print('- Hello, talking horse Julius,- said robber number '+str(n//2)+'.')
elif n<162 and n%2==0:
    print('- Hello, horse of robber number '+str((n-80)//2)+',- said talking horse
    ↪ Julius.')
elif n<162:
    print('- Hello, talking horse Julius,- said horse of robber number
    ↪ '+str((n-80)//2)+'.')
elif n<242 and n%2==0:
    print('- Goodbye, robber number '+str((n-160)//2)+',- said talking horse
    ↪ Julius.')
elif n<242:
    print('- Goodbye, talking horse Julius,- said robber number
    ↪ '+str((n-160)//2)+'.')
elif n<322 and n%2==0:
    print('- Goodbye, horse of robber number '+str((n-240)//2)+',- said talking
    ↪ horse Julius.')
elif n<322:
    print('- Goodbye, talking horse Julius,- said horse of robber number
    ↪ '+str((n-240)//2)+'.')
else:
    print('And talking horse Julius went on along the road.')
```

Задача I.1.4.3. Нумерация (20 баллов)

Темы: задачи для начинающих, реализация, структуры данных.

Власти города Байтленда решили построить новый микрорайон, который будет состоять из одной длинной улицы. Процесс постройки домов будет выглядеть следующим образом. Сначала будет построен самый первый дом, который получит номер 0. (Разумеется, в Байтленде все нумерации начинаются с нуля.) Далее все дома будут пристраиваться слева или справа от существующей застройки. Дома будут получать номера в порядке их ввода в эксплуатацию.

Рассмотрим пример. Пусть дом номер 1 был построен слева от дома номер 0. Тогда нумерация домов слева направо будет выглядеть как 1 0. Далее был построен дом номер 2 слева от существующих. Теперь дома на улице будут иметь номера 2 1 0. Далее был построен дом номер 3 справа от существующих. Теперь на улице будут стоять дома с номерами 2 1 0 3. Наконец, если дом номер 4 будет построен слева, а дома с номерами 5 и 6 справа, то последовательность номеров домов превратится в 4 2 1 0 3 5 6.

На улице построили n домов с номерами от 0 до $n - 1$. Для каждого дома с ненулевым номером нам стало известно с какой стороны от существующих он был построен. Теперь ваша задача — написать программу, которая перечислит номера домов в порядке движения по улице слева направо.

Формат входных данных

В первой строке на вход подается число n — количество построенных домов. $2 \leqslant n \leqslant 400000$. Во второй строке записана последовательность из $n - 1$ символов «L» или «R» (без кавычек). Символ «L» в i-той позиции означает, что дом с номером i был построен слева от предыдущих, а символ «R» — справа.

Формат выходных данных

Вывести через пробел в одной строке последовательность из n чисел — нумерацию домов после завершения строительства.

Методика проверки и описание тестов

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. Тест из условия задачи при проверке не используется.

В первых пяти тестах $n \leqslant 100$.

Примеры

Пример №1

Стандартный ввод
7 LLRLRR

Стандартный вывод
4 2 1 0 3 5 6

Решение

По условию задачи требуется составить последовательность чисел, добавляя их слева или справа к существующим. Для решения можно использовать структуру данных дек или изменить порядок выполнения команд добавления элементов. Требуемую последовательность можно получить, выполнив два прохода по строке, задающей команды. Во время первого прохода будем добавлять числа слева от нуля. Чтобы получить нужный порядок, по строке потребуется пройти справа налево. Далее необходимо вывести ноль и сделать второй проход для вывода чисел справа от нуля.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n=int(input())
x=input()
ans=''
for i in range(n-1,0,-1):
    if x[i-1]=='L':
        print(i,end=' ')
print(0,end=' ')
for i in range(1,n):
    if x[i-1]=='R':
        print(i,end=' ')
```

Задача I.1.4.4. Космическая связь (20 баллов)

Темы: реализация.

В этой задаче от вас потребуется написать программу для составления расписания сеансов связи со спутником. Каждый сеанс заключается в обмене короткими сообщениями, поэтому мы считаем, что он происходит мгновенно. Поскольку ресурсы оборудования ограничены, следует стремиться к минимизации количества сеансов. Вместе с тем, интервал времени между сеансами не должен превышать d миллисекунд. Кроме того, существуют промежутки времени, в течении которых связь невозможна. При этом на границах промежутка мгновенный сеанс связи возможен. Расписание составляется на t миллисекунд. Первый сеанс должен обязательно состояться в момент 0, а последний — в момент t.

Рассмотрим пример. Пусть $t = 100$, $d = 20$ и задано 3 промежутка недоступности связи: (5;25), (27;40), (75;90). Тогда потребуется восемь сеансов связи, которые можно провести в моменты времени 0, 5, 25, 45, 65, 75, 90, 1000, 5, 25, 45, 65, 75, 90, 100. Конкретное расписание может быть другим, но в любом случае количество сеансов не может быть меньше 8.

Ваша программа должна по имеющейся информации найти минимальное возможное количество сеансов связи.

Формат входных данных

В строке 1 через пробел записаны 3 натуральных числа n, d и t — количество интервалов недоступности связи, максимальный интервал между между сеансами

и время, на которое составляется расписание. $n \leqslant 200000$, $d, t \leqslant 10^9$. Далее в n строках заданы по два целых неотрицательных числа a_i и b_i — начало и конец каждого интервала недоступности связи. $b_i - a_i \leqslant d$. Интервалы недоступности связи не пересекаются, каждый следующий интервал начинается строго после окончания предыдущего. $0 \leqslant a_1 < b_1 < a_2 < b_2 < \ldots < a_n < b_n \leqslant t$.

Формат выходных данных

Вывести число — количество сеансов связи в графике.

Методика проверки

Программа проверяется на 25 тестах. Прохождение каждого теста оценивается в 0,8 балла. Тест из условия задачи при проверке не используется.

В первых 10 тестах $t \leqslant 1000$. В следующих 10 тестах $t \leqslant 10^6$.

Примеры

Пример №1

Стандартный ввод
3 20 100 5 25 27 40 75 90

Стандартный вывод
8

Решение

Для решения этой задачи можно использовать идею жадного алгоритма. Каждый следующий сеанс связи будем проводить как можно позже. Если время предыдущего сеанса было равно t, то следующий сеанс будем проводить в момент $t + d$. Однако, если $t + d$ попадет внутрь некоторого недоступного интервала $[a; b]$, то время сеанса потребуется сместить до левой границы интервала.

Для получения полных баллов за решение требуется составить алгоритм линейной сложности относительно количества недоступных для связи интервалов. Таким образом, на каждом шаге цикла будет рассматриваться один интервал, который будем называть текущим.

Решение этой задачи требует аккуратной реализации. Инвариантом предлагаемого решения является утверждение о том, что t никогда не попадет внутрь некоторого недоступного интервала, и $t + d$ будет больше, чем правая граница текущего интервала, что гарантирует правильность работы на следующих итерациях цикла.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
n,d,s=map(int,input().split())
m=1
t=0
for i in range(n):
    a,b=map(int,input().split())
    k=(a-t)//d
    t+=d*k
    m+=k
    if t+d<b:
        t=a
        m+=1
print(m+(s-t+d-1)//d)
```

Задача I.1.4.5. Простая задача (20 баллов)

Темы: реализация.

В этой задаче не будет длинного условия и простого решения. Все будет наоборот. Требуется провести непрерывную линию произвольного вида из точки с координатами (x_1, y_1) в точку с координатами (x_2, y_2) так, чтобы минимизировать количество точек на этой линии, в которых хотя бы одна координата является целым числом. Ответом к задаче будет являться количество таких точек. Если начальная или конечная точка линии будет иметь хотя бы одну целочисленную координату, то ее тоже надо учитывать.

Формат входных данных

Каждый тест в этой задаче будет содержать n запросов. $1 \leqslant n \leqslant 100$. Натуральное число n будет записано в первой строке. Далее в n строках записаны запросы. Каждый запрос располагается в отдельной строке и состоит из четырех чисел x_1, y_1, x_2, y_2, которые задают координаты двух точек. Точки не совпадают. Координаты могут быть целыми или вещественными числами не более чем с 2 знаками после точки. Координаты не превосходят 10^9 по абсолютной величине. Если координата является целым числом, то ее запись не содержит десятичной точки.

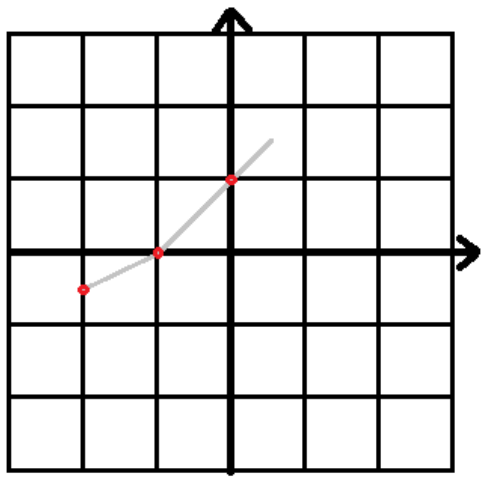
Формат выходных данных

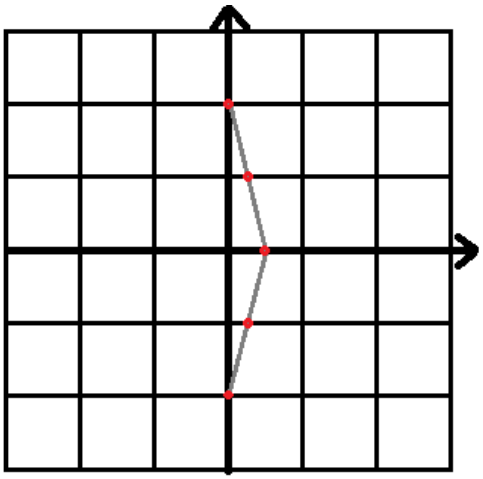
Требуется вывести ответы на запросы по одному ответу в каждой строке.

Методика проверки и пояснение к тесту

Программа проверяется на 10 тестах. Прохождение каждого теста оценивается в 2 балла. Тест из условия задачи при проверке не используется.

Следующие рисунки поясняют ответ к тесту. Обратите внимание, что никакой фрагмент линии не может лежать на сетке, так как в этом случае количество точек с целочисленной координатой будет бесконечно большим.





Примеры

Пример №1

Стандартный ввод
2 -2 -0.5 0.5 1.5 0 -2 0 2
Стандартный вывод
3 5

Решение

Для решения задачи потребуется найти количество горизонтальных и вертикальных прямых целочисленной сетки, которые будут пересекаться нарисованной линией. Поскольку одна вертикальная и одна горизонтальная прямая могут быть пересечены в одной точке, из полученных значений надо будет взять максимальное. Далее надо будет проверить, являются ли координаты концов линии целыми числами, и, в случае необходимости, учесть эти точки в ответе.

Для поиска количества точек пересечения в предлагаемом решении используется функция *bw*. Сначала границы диапазона смещаются на некоторую небольшую величину, чтобы их координаты перестали быть целочисленными. Далее верхняя граница диапазона округляется вниз, а нижняя вверх, после чего вычисляется количество целых чисел в диапазоне.

Пример программы-решения

Ниже представлено решение на языке Python 3.

```
1  import math
2  def bw(a,b):
3      if a>b:
4          a,b=b,a
```

```
    return max(0,math.floor(b-0.001)-math.ceil(a+0.001)+1)

n=int(input())
for i in range(n):
    x1,y1,x2,y2=map(float,input().split());
    print(max(bw(x1,x2),bw(y1,y2))+
int((x1==round(x1) or y1==round(y1))+
int((x2==round(x2) or y2==round(y2)))))
```

Задачи первого этапа. Математика

Первая попытка. Задачи 8–9 класса

Задача I.2.1.1. (15 баллов)

Темы: числа, сравнения, логика.

Пусть x — некоторое натуральное число. Среди утверждений:

- «$2x$ больше 70»;
- «x меньше 100»;
- «$3x$ больше 25»;
- «x не меньше 10»;
- «x больше 5»;

три верных и два неверных. Чему равно x?

Решение

Если $x \leqslant 8$, то как минимум три утверждения (первое, третье и четвертое) неверны, поэтому такие значения x не удовлетворяют условию.

Если $10 \leqslant x < 100$, то верны как минимум четыре утверждения (все, кроме первого), поэтому такие x также не удовлетворяют условию.

Если $x \geqslant 100$, то верны все утверждения, кроме второго, и такие x также не подходят.

Остается единственный вариант $x = 9$, при этом верны второе, третье и пятое утверждения и неверны первое и четвертое, т. е. условия задачи выполнены.

Ответ: 9.

Задача I.2.1.2. (15 баллов)

Темы: алгебра, преобразования.

Известно, что $\dfrac{2x+3y}{2x-3y}+\dfrac{2x-3y}{2x+3y}=\dfrac{50}{7}$. Чему равно значение выражения $\dfrac{x^2+2y^2}{x^2-2y^2}$?

Решение

Приведем исходное уравнение к общему знаменателю:

$$\frac{(2x+3y)^2+(2x-3y)^2}{(2x-3y)(2x+3y)}=\frac{50}{7},$$

откуда $7(8x^2 + 18y^2) = 50(4x^2 - 9y^2)$, или после упрощения $x^2 = 4y^2$. Подставим в искомое выражение: $\dfrac{x^2 + 2y^2}{x^2 - 2y^2} = \dfrac{6y^2}{2y^2} = 3$.

Ответ: 3.

Задача I.2.1.3. (20 баллов)

Темы: геометрия, периметр.

Кукурузное поле имеет форму прямоугольника, разделенного на девять меньших прямоугольников. Агроном Александр измерил периметры некоторых участков (в метрах) и отметил их на плане (см. рисунок). Помогите Александру без дополнительных измерений узнать периметр участка, обозначенный на схеме за x. Ответ укажите в метрах.

	600	1000
700		x
800	500	

Решение

Легко заметить, что периметры участков, отмеченные на схеме **жирным**, равны периметру всего поля, и то же касается периметров, отмеченных курсивом. Из уравнения $700 + 500 + 1000 = 600 + x + 800$ находим $x = 800$.

	600	**1000**
700		x
800	**500**	

Ответ: 800.

Задача I.2.1.4. (20 баллов)

Темы: числа, делимость.

Сергей перемножил шесть последовательных натуральных нечетных чисел и записал результат на бумаге, согнул листочек и передал его Саше. Саша долго носил его в кармане, а когда развернул, то увидел число 10530*075 («*» заменяет одну цифру, стершуюся на сгибе). На какую цифру надо Саше заменить «*», чтобы получился верный результат без повторения вычислений?

Решение

Среди шести последовательных нечетных натуральных чисел есть два числа, кратные 3. Значит, искомое произведение делится на 9, а, значит, и сумма цифр должна делиться на 9. Сумма оставшихся цифр числа равна 21, следовательно, «*» = 6.

Ответ: 6.

Задача I.2.1.5. (30 баллов)

Темы: комбинаторика, процессы.

Ослик Иа собрал кучу из 50 шишек и решил разложить их по кучкам следующим образом. Сначала кучу он произвольно разбивает на две кучи. Затем любую из имеющихся куч он снова разбивает на две кучи, и так далее до тех пор, пока каждая кучка не будет состоять из одной шишки. Чтобы еще заодно тренироваться в арифметике, Иа при каждом разбиении какой-либо кучи на две записывает произведение количеств шишек в двух получившихся кучках. Чему может быть равна сумма всех записанных чисел?

Решение

Изобразим шишки точками и соединим каждую пару точек отрезком. Получим $\frac{50 \cdot (50-1)}{2} = 1225$ отрезков. При каждом разбиении одной кучи шишек на две будем стирать все отрезки, соединяющие точки, соответствующие шишкам, оказавшимся в разных кучах. Пусть на некотором шаге мы разбили шишки одной из уже имевшихся куч на две кучки по x и y шишек. Тогда мы стираем xy отрезков. Это же число мы записываем. Таким образом, сумма записанных чисел — это количество всех стертых отрезков. Так как изначально было 1225 отрезков, а в итоге все отрезки стерты, то общее количество стертых отрезков равно 1225.

Ответ: 1225.

Первая попытка. Задачи 10–11 класса

Задача I.2.2.1. (15 баллов)

Темы: алгебра, преобразования, уравнения.

Известно, что $\frac{7x+3y}{x+5y} + \frac{3x-2y}{2x+y} = 4$. Чему может равняться значение выражения $t = \frac{x^2+2y^2}{x^2-2y^2}$? Если вариантов несколько, запишите в ответ наименьшее возможное значение t.

Решение

Приведем исходное уравнение к общему знаменателю:

$$\frac{(7x+3y)(2x+y)+(3x-y)(x+5y)}{(x+5y)(2x+y)}=4,$$

откуда после упрощения $x^2-2xy-3y^2=0$. Корнями уравнения являются $x=3y$ и $x=-y$. В первом случае $t=\dfrac{11}{7}$, во втором $t=-3$. В ответ записываем наименьшее значение $t=-3$.

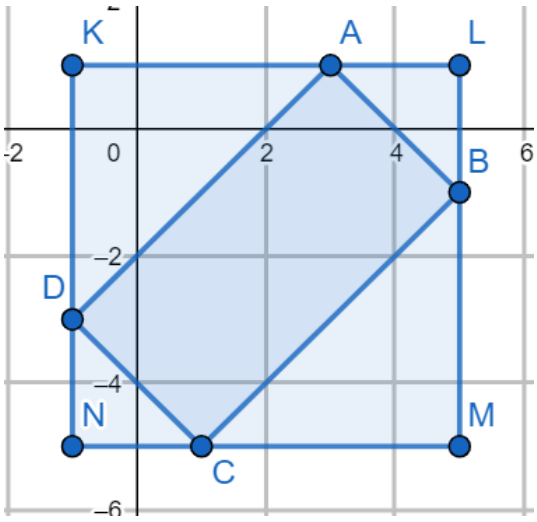
Ответ: -3.

Задача I.2.2.2. (15 баллов)

Темы: диофантовы уравнения, площадь.

Найдите площадь выпуклого многоугольника, координаты $(x;y)$ вершин которого являются целыми числами и удовлетворяют уравнению $xy-2y+2x=7$.

Решение



Преобразуем уравнение к виду $(x-2)(y+2)=3$. Поскольку x и y — целые числа, то возможны следующие четыре варианта: 1) $\begin{cases}x-2=1\\y+2=3\end{cases}$; 2) $\begin{cases}x-2=3\\y+2=1\end{cases}$; 3) $\begin{cases}x-2=-1\\y+2=-3\end{cases}$; 4) $\begin{cases}x-2=-3\\y+2=-1\end{cases}$. Решениями этих систем являются пары чисел, которые соответствуют точкам на плоскости $A(3;1)$; $B(5;-1)$; $C(1;-5)$; $D(-1;-3)$. Дополнениями четырехугольника $ABCD$ до прямоугольника $KLMN$ являются равнобедренные прямоугольные треугольники. Поэтому $ABCD$ — прямоугольник. Его стороны равны $2\sqrt{2}$ и $4\sqrt{2}$, площадь равна 16.

Ответ: 16.

Задача I.2.2.3. (20 баллов)

Темы: числа, делимость.

Сумма трех натуральных чисел равна 2021. Если в двух их них зачеркнуть цифры в разряде единиц и сложить полученные числа, то получится третье число. Найдите третье число.

Решение

Обозначим искомые числа A, B и C. Пусть $A = 10a + x$, $B = 10b + y$, где x и y — цифры единиц чисел A и B соответственно. Тогда:

$$\begin{cases} A + B + C = 2021 \\ a + b = C \end{cases} \Leftrightarrow \begin{cases} 10a + x + 10b + y + C = 2021 \\ a + b = C \end{cases} \Leftrightarrow \begin{cases} 11C = 2021 - (x + y) \\ a + b = C \end{cases}.$$

Поскольку $(x + y) \in [0; 18]$, то для делимости $2021 - (x + y)$ на 11 имеется ровно одна возможность $x + y = 8$. При этом $C = 183$.

Ответ: 183.

Задача I.2.2.4. (20 баллов)

Темы: вероятность.

У Сергея есть сундук с карточками, на которых написаны все семизначные числа (по одному на каждой карточке). Сергей называет число *зеркальным*, если оно одинаково читается слева направо и справа налево. Сергей случайным образом достает из сундука одну карточку. Какова вероятность, что Сергей достанет карточку, на которой записано зеркальное число, делящееся на 3? (Ответ запишите в виде десятичной дроби с точностью до 6 знаков после запятой).

Решение

Всего семизначных чисел 9 000 000.

Разобьем зеркальное число на центральную цифру Ц и две группы по 3 цифры. Левая группа образует трехзначное число Л, правая получается записью цифр левого в обратном порядке. Число N делится на 3 если его сумма цифр $s(N)$ делится на 3. Так как у правой и левой группы суммы цифр одинаковы, то общая сумма цифр равна $2s(\text{Л}) + \text{Ц}$. Она делится на 3 тогда и только тогда, когда $s(\text{Л})$ и Ц дают одинаковые остатки при делении на 3.

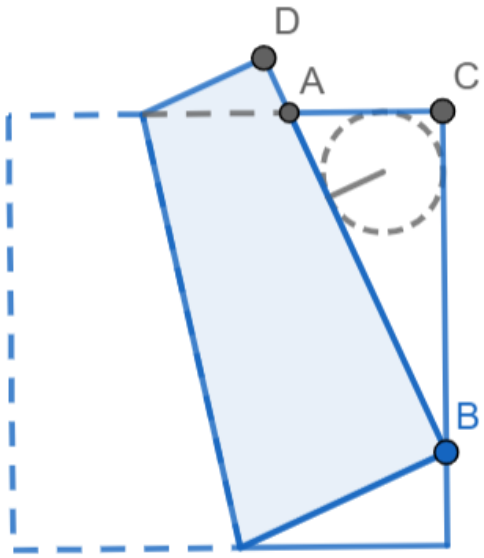
Всего есть 900 трехзначных чисел. Разобьем их на тройки так, чтобы в каждой встречались по разу остатки 0, 1, 2 от деления суммы цифр на 3. Самый простой пример разбиения — на тройки последовательных: $(100, 101, 102), ..., (997, 998, 999)$. В каждой такой тройке числа дают разные остатки, тогда, по свойству равноостаточности, и суммы цифр дают разные остатки. Каждый остаток $(0, 1, 2)$ имеет по $900 : 3 = 300$ чисел. Итак, для каждой цифры Ц есть 300 вариантов Л, дающих кратное 3 зеркальное число. Значит, всего искомых чисел 3000, а вероятность равна $3000/9000000 \approx 0,000333$.

Ответ: 0,000333.

Задача I.2.2.5. (30 баллов)

Темы: планиметрия.

Вася перегнул квадратный лист бумаги со стороной 20 см так, как показано на рисунке. Найдите радиус окружности, вписанной в треугольник ABC, если известно, что $AD : AB = 1 : 14$. Ответ (в см) округлите до двух знаков после запятой. Дробную часть от целой отделяйте запятой. В ответ вводите только число, без указания единиц измерения.



Решение

1. Перегибание листа бумаги с геометрической точки зрения означает симметрию относительно прямой GF. При этом точка E переходит в D, а точка K — в B. Следовательно, $EF = DF$, а $KG = BG$. По свойству равнобедренного треугольника $\angle KBG = \angle GKB$.
2. Опустим перпендикуляр KH на прямую BD. Тогда $\angle HKB = \angle GKB$ как накрест лежащие при параллельных прямых KH и GB и секущей BK.
3. Из пп. 1 и 2 следует равенство углов HKB и LKB. Тогда прямоугольные треугольники KLB и KHB равны по общей гипотенузе и равным острым углам. Отсюда $KH = KL$ и $HB = LB$.
4. Прямоугольные треугольники KEA и KHA равны по общей гипотенузе KA и равным катетам KE и KH ($KE = KL$ как стороны квадрата, а $KH = KL$ из п. 3). Следовательно, $EA = HA$.
5. Чтобы найти радиус вписанной окружности прямоугольного треугольника ABC, воспользуемся формулой $r = \dfrac{a+b-c}{2}$, где a и b — катеты, c — гипотенуза:

$$r = \frac{1}{2}(AC + BC - AB) = \frac{1}{2}(EC - EA + CL - LB - AH - BH) =$$

$$= EC - (AH + HB) = DA.$$

6. Поскольку $AD : AB = 1 : 14$, имеем: $r = AD = \dfrac{1}{15}AB = \dfrac{1}{15} \cdot 20 \approx 1,33$ (см).

Ответ: 1,33.

Вторая попытка. Задачи 8–9 класса

Задача I.2.3.1. (15 баллов)

Темы: уравнения, текстовая задача.

В 18 веке член Петербургской академии наук медик Иосий Вейтбрехт предложил свою шкалу измерения температуры. Нулевое значение соответствовало температуре кипения воды (100 градусов по используемой сейчас шкале Цельсия), температуре замерзания воды (0 градусов по шкале Цельсия) соответствовало 150 градусов по шкале Вейтбрехта. При какой температуре термометр со шкалой Вейтбрехта и термометр со шкалой Цельсия покажут одинаковые значения? Зависимость одной шкалы температуры от другой считайте линейной.

Решение

Температура по Вейтбрехту зависит от температуры по Цельсию по закону $t\text{В} = -1,5t_{\text{Ц}} + 150$. Если $t_{\text{В}} = t_{\text{Ц}}$, то это будет при $t_{\text{Ц}} = -1,5t_{\text{Ц}} + 150$, то есть при $t_{\text{В}} = t_{\text{Ц}} = 60$.

Ответ: 60.

Задача I.2.3.2. (15 баллов)

Темы: геометрия, площадь.

План сада имеет форму прямоугольника, разделенного дорожками на девять прямоугольных частей. Садовник Александр знает площади некоторых из частей (они указаны на рисунке). Помогите Александру, не выполняя измерений, узнать площадь левой верхней части сада. Считайте ширину дорожек нулевой.

?	240	
	112	84
108		144

Решение

Заметим, что если прямоугольник разрезан горизонтальным и вертикальным отрезками на четыре прямоугольника, то произведения площадей противоположных частей равны. Тогда последовательно находим площади нижней средней части ($112 \cdot 144 : 84 = 192$), средней левой части ($108 \cdot 112 : 192 = 63$) и, наконец, верхней левой части ($63 \cdot 240 : 112 = 135$).

135	240	
63	112	84
108	*192*	144

Ответ: 135.

Задача I.2.3.3. (20 баллов)

Темы: логика.

На юбилее Солнечного Города должны выступить Знайка, Винтик, Шпунтик, Пилюлькин и Незнайка. Сколькими способами их можно расположить в списке выступающих при условии, что до Незнайки должны выступать Винтик и Шпунтик (в каком-то порядке)?

Решение

Обозначим выступающих буквами З, В, Ш, П и Н. Н мог выступать третьим, четвертым или пятым. Разберем эти варианты.

1. Если Н выступает третьим, то перед ним выступают В и Ш — 2 варианта расстановки, после Н выступают З и П — 2 варианта. Всего 4 варианта в списке.
2. Если Н выступает четвертым, то перед ним выступают В и Ш и кто-то из З и П. Для выбора З или П — 2 варианта, расстановка этих трех докладчиков — 6 вариантов. Всего 12 вариантов списка.
3. Если Н завершает список, то перед ним 4 докладчика, для которых есть $4! = 24$ вариантов составить список.

Всего $4 + 12 + 24 = 40$ различных списков.

Ответ: 40.

Задача I.2.3.4. (20 баллов)

Темы: числа, делимость.

Саша записал на доске натуральное число $P \geqslant 2$, Сергей приписал к нему такое же число P. Оказалось, что полученное число $\overline{PP}$ кратно P^2. Найдите частное от деления $\overline{PP}$ на P^2.

(Поясним, что означает запись $\overline{PP}$, на примере: если $P = 9876$, то $\overline{PP} = 98769876$.)

Решение

Если P — однозначное число, то $\overline{PP} = 11 \cdot P \vdots P^2$ и, следовательно, $11 \vdots P$. Поскольку $P > 1$, этот случай невозможен.

Пусть теперь P — n-значное число, где $n \geqslant 2$. Тогда $\overline{PP} = P \cdot \underbrace{100...001}_{n+1\text{разряд}}$ и для делимости на P^2 необходимо, чтобы $\underbrace{100...001}_{n+1\text{разряд}}$ делилось на P. Частное может быть только однозначным числом, большим 1 и являющимся делителем числа $\underbrace{100...001}_{n+1\text{разряд}}$. Это число нечетно, поэтому оно не кратно 2, 4, 6, 8. Сумма его цифр равна 2, поэтому это число некратно 3 и 9. Последняя цифра числа 1, поэтому оно некратно 5. Единственный возможный делитель 7.

Пример: $P = 143$, *тогда* $\overline{PP} = 143\,143$, $\overline{PP} : P^2 = 143\,143 : 143^2 = 7$.

Ответ: 7.

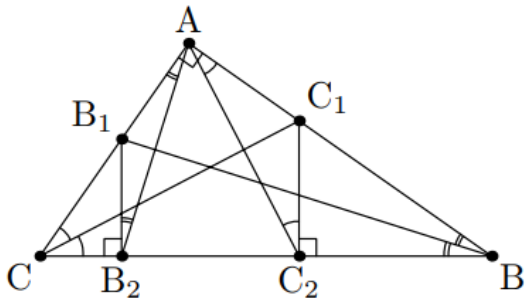
Задача I.2.3.5. (30 баллов)

Темы: геометрия, треугольники.

В прямоугольном треугольнике ABC с прямым углом A проведены биссектрисы BB_1 и CC_1. Из точек B_1 и C_1 на гипотенузу BC опущены перпендикуляры B_1B_2 и C_1C_2. Найдите величину угла B_2AC_2. Ответ запишите в градусах.

Решение

Заметим, что $C_1C_2 = C_1A$ (так как биссектриса CC_1 равноудалена от сторон AC и BC). Значит, треугольник AC_1C_2 равнобедренный, и, следовательно, $\angle C_1AC_2 = \frac{1}{2}\angle BC_1C_2 = \frac{1}{2}\angle ACB$, так как углы BC_1C_2 и ACB оба дополняют угол ABC до прямого. Аналогично получаем, что $\angle B_1AB_2 = \angle ACB$, откуда имеем: $\angle C_1AC_2 + \angle B_1AB_2 = \frac{1}{2}(\angle ACB + \angle ABC) = 45°$.



Ответ: 45.

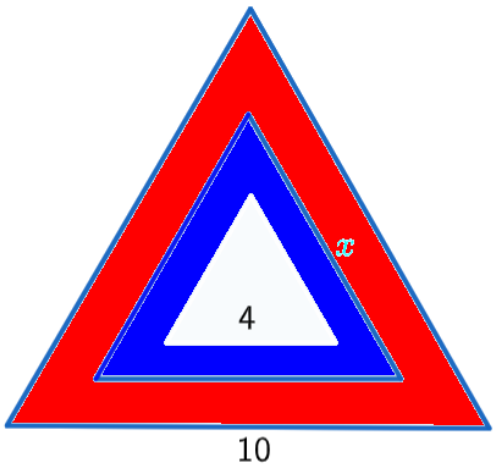
Вторая попытка. Задачи 10–11 класса

Задача I.2.4.1. (15 баллов)

Темы: планиметрия, площадь.

Сергей придумал эмблему нового прибора в виде трех правильных треугольников, имеющих общий центр и попарно параллельные стороны. При этом Сергей хочет

добиться, чтобы средний треугольник делил площадь фигуры, заключенной между внутренним и внешним, в отношении 3 : 4, считая от внутреннего. Какой длины должна быть сторона среднего треугольника, если стороны внутреннего и внешнего треугольников равны 4 см и 10 см соответственно? Ответ в сантиметрах округлите до двух знаков после запятой.



Решение

Как известно, площадь правильного треугольника со стороной a равна $\frac{a^2\sqrt{3}}{4}$. Если обозначить неизвестную сторону среднего треугольника за x см, получим уравнение:

$$\frac{(x^2-4^2)\sqrt{3}}{4} : \frac{(10^2-x^2)\sqrt{3}}{4} = 3:4.$$

Отсюда $4(x^2-16) = 3(100-x^2)$, тогда $x = \sqrt{52} \approx 7,21$ (см).

Ответ: 7,21.

Задача I.2.4.2. (15 баллов)

Темы: турниры, алгебра, неравенства.

Команда игроков набрала вместе в игре некоторое количество очков. Лучший игрок команды набрал 1/7 общего количества очков, а игрок, набравший наименьшее количество очков, набрал 1/9 от общего количества. Сколько игроков было в команде?

Решение

Пусть команда набрала x очков и было n игроков. Тогда каждый игрок набрал не менее, чем $\frac{x}{9}$ очков и не более, чем $\frac{x}{7}$ очков. Значит, все игроки набрали не менее, чем $\frac{x}{9}+\frac{x}{7}+\frac{x}{9}(n-2)$ очков, но не более, чем $\frac{x}{9}+\frac{x}{7}+\frac{x}{7}(n-2)$ очков. Таким образом,

выполняется неравенство $\frac{x}{9}+\frac{x}{7}+\frac{x}{9}(n-2) \leqslant x \leqslant \frac{x}{9}+\frac{x}{7}+\frac{x}{7}(n-2)$. Упростив его, получим $\frac{n}{9}+\frac{2}{63} \leqslant 1 \leqslant \frac{n}{7}-\frac{2}{63}$, то есть $7\frac{2}{9} \leqslant n \leqslant 8\frac{5}{7}$. Поскольку n — целое, то $n = 8$.

Ответ: 8.

Задача I.2.4.3. (20 баллов)

Темы: функции, экстремум.

Найдите наименьшее значение выражения:

$$x^2 + 5y^2 + 6z^2 - 4xy + 4yz - 2x + 6y + 4z + 5.$$

Решение

Рассмотрим данное выражение как квадратичную функцию относительно переменной x:

$$f(x) = x^2 - 2(2y+1)x + 5y^2 + 6z^2 + 4yz + 6y + 4z + 5.$$

Она принимает наименьшее значение при $x_{\text{в}} = 2y + 1$. Это наименьшее значение $f_{\text{мин}} = y^2 + 6z^2 + 4yz + 2y + 4z + 4$ рассмотрим как квадратичную функцию относительно переменной y: $g(y) = y^2 + 2(2z+1)y + 6z^2 + 4z + 4$. Эта функция принимает наименьшее значение при $y_{\text{в}} = -(2z+1)$. Это наименьшее значение $g_{\text{мин}} = 2z^2 + 3$ будет наименьшим при $z = 0$ и будет равно 3.

Ответ: 3.

Задача I.2.4.4. (20 баллов)

Темы: вероятность.

У Саши есть 9 карточек с написанными цифрами от 1 до 9 (на каждой карточке по одной цифре). Саша, не глядя, берет некоторые (может быть, одну или все) из них. Набор он считает хорошим, если сумма цифр в наборе четная. Найдите вероятность, что случайно выбранный набор окажется хорошим. Ответ запишите в виде десятичной дроби с 3 знаками после запятой.

Решение

Введем пустой набор. Пусть он будет хорошим. Тогда количество всех наборов равно 2^9. Заметим, что сумма всех цифр на карточках — нечетное число. Поэтому любому хорошему набору соответствует набор из оставшихся карточек с нечетной суммой. И наоборот. Поэтому хороших наборов ровно половина от всех, то есть $2^9/2 = 2^8$. Поскольку мы добавили один хороший набор, то настоящих хороших наборов $2^8 - 1 = 255$. $\frac{2^8-1}{2^9-1}$.

Ответ: 0,499.

Задача I.2.4.5. (30 баллов)

Темы: уравнения.

Решите в действительных числах уравнение:

$$x + \sqrt{x^2 - 16} = \frac{2(x+4)}{(x-4)^2}.$$

Если корней несколько, запишите наибольший из них. Ответ округлите до двух знаков после запятой.

Решение

Пусть $x + \sqrt{x^2-16} = t$. Тогда, $\sqrt{x^2-16} = t - x \Leftarrow \begin{cases} t \geqslant x \\ x^2 - 16 = t^2 - 2tx + x^2 \end{cases}$. Из второго уравнения следует, что $2tx = t^2 + 16$ и, в частности, $t \neq 0$. Отсюда $x = \dfrac{t^2+16}{2t}$, $x \pm 4 = \dfrac{t^2 \pm 8t + 16}{2t} = \dfrac{(t \pm 4)^2}{2t}$. Значит, исходное уравнение будет иметь вид $t = \dfrac{2(t+4)^2}{2t} \cdot \dfrac{4t^2}{(t-4)^4}$, или $(t-4)^4 = 4(t+4)^2$. Отсюда $\left[\begin{array}{l} (t-4)^2 = 2(t+4) \\ (t-4)^2 = -2(t+4) \end{array}\right. \Leftrightarrow \left[\begin{array}{l} t^2 - 10t + 8 = 0 \\ t^2 - 6t + 24 = 0 \end{array}\right.$. Второе уравнение действительных корней не имеет. Корни первого уравнения $t = 5 \pm \sqrt{17}$ при этом $x = \dfrac{t^2+16}{2t} = \dfrac{15 \mp \sqrt{13}}{2}$. Учитывая условие $t \geqslant x$, получаем, что корнем исходного уравнения является только $x = \dfrac{15 - \sqrt{17}}{2}$. С учетом округления по математическим правилам до двух знаков после запятой получаем $x \approx 5,44$.

Ответ: 5,44.

Третья попытка. Задачи 8–9 класса

Задача I.2.5.1. (15 баллов)

Темы: логика, принцип Дирихле.

В школьном кружке «Физики и лирики», где каждый участник или физик или лирик, 27 человек. Сергей провел исследование и выяснил, что у любых двух физиков из кружка количество друзей–лириков из этого кружка не совпадает. Какое наибольшее количество физиков может быть в этом кружке?

Решение

Если в кружке 14 физиков, то количество их друзей-лириков может быть любым целым числом от 0 до 13 (14 различных вариантов), что соответствует условию. Если же физиков больше 14, то лириков в кружке будет не больше 12, а значит, различных вариантов будет не больше 13 (от 0 до 12). Поэтому, по принципу Дирихле, хотя бы у

двух физиков окажется одно и то же количество друзей-лириков, что противоречит условию.

Ответ: 14.

Задача I.2.5.2. (15 баллов)

Темы: алгебра, текстовые задачи.

Удав решил измерить свой рост и попросил попугая ему помочь. Но лежать на месте удаву скучно, поэтому во время измерения он полз с постоянной скоростью. Попугай сначала прошел от хвоста удава к его голове и насчитал при этом 80 своих шагов, а затем немедленно развернулся и на обратном пути к хвосту удава насчитал 20 своих шагов (и от хвоста к голове удава, и обратно попугай шел с одной и той же скоростью). Чему равен рост удава в попугайских шагах?

(Под ростом удава мы, как и в известном мультфильме, понимаем его длину.)

Решение

За время движения туда и обратно попугай сделал $80+20 = 100$ шагов, а удав прополз расстояние в $80 - 20 = 60$ шагов. Следовательно, скорость удава в $\dfrac{100}{60} = \dfrac{5}{3}$ раз меньше скорости попугая, и за то время, что попугай дошел от хвоста до головы удава, удав переместился на $80 : \dfrac{5}{3} = 48$ шагов. Следовательно, рост удава (расстояние от головы до хвоста, посчитанное в момент, когда попугай повернул обратно) составляет $80 - 48 = 32$ попугайских шага.

Замечание. В общем случае, если по пути от хвоста к голове удава попугай сделал a шагов, а обратно — b шагов ($a > b$), рост удава составляет $\dfrac{2ab}{a+b}$ шагов, т. е. *среднее гармоническое* чисел a и b.

Ответ: 32.

Задача I.2.5.3. (20 баллов)

Темы: логика, графы.

Сергею досталась карта острова сокровищ. На ней изображены несколько сундуков с золотом, которые расположены в пещерах (в каждой пещере по одному сундуку). Пещеры соединены тоннелями, прорытыми гномами (каждый тоннель обозначен своим цветом, при этом известно, что тоннелей больше одного). Изучив карту, Сергей обнаружил, что от каждой пещеры до любой другой можно добраться, не меняя тоннеля. Кроме того, каждые два тоннеля пересекаются ровно в одной пещере. А вдоль каждого тоннеля расположены ровно три пещеры.

Сколько всего сундуков с золотом изображено на карте острова сокровищ?

Решение

Пусть a — один из тоннелей, а B — пещера, через которую тоннель a не проходит. Каждый тоннель, проходящий через пещеру B, пересекается с a, причем разные тоннели — в разных пещерах.

Следовательно, через пещеру B проходят ровно три тоннеля. Каждый из них проходит через две пещеры, отличные от B. Таким образом, всего на карте изображено $3 \cdot 2 + 1 = 7$ сундуков.

Замечание. Поскольку через каждую пару пещер (а их $C_7^2 = 21$) проходит один тоннель, а вдоль каждого тоннеля расположено $C_3^2 = 3$ пар пещер, то всего тоннелей 21 : 3 =7, т. е. столько же, сколько сундуков.

Пример. Если занумеровать сундуки/пещеры цифрами от 1 до 7, то семь тоннелей могут проходить через сундуки (каждый через три) с номерами: 123, 146, 157, 247, 256, 367 и 345.

Ответ: 7.

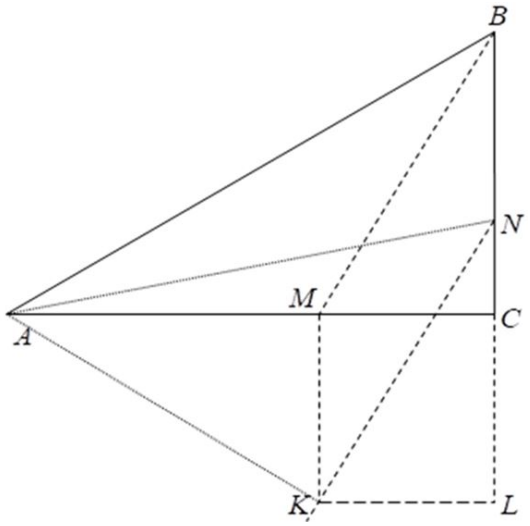
Задача I.2.5.4. (20 баллов)

Темы: геометрия, прямоугольный треугольник.

В прямоугольном треугольнике ABC с прямым углом при вершине C на катете AC выбрана точка M так, что $AM = BC$, а на катете BC — точка N так, что $BN = MC$. Найдите угол между прямыми AN и BM.

Ответ запишите в градусах.

Решение



На перпендикуляре к AC в точке M вне $\triangle ABC$ отложим отрезок $MK = MC$. Затем построим точку L, для которой $MCLK$ — квадрат. Тогда:

$$\triangle AMK = \triangle MCB = \triangle KNL.$$

Поэтому $AK = KN$ и:

$$\angle AKN = \angle AKM + \angle MKN = \angle AKM + \angle KNL = \angle AKM + \angle KAM = 90°.$$

Тогда $\angle ANK = 45°$, а это и есть искомый угол между прямыми, поскольку $BM \parallel KN$.

Ответ: 45.

Задача I.2.5.5. (30 баллов)

Темы: логика.

Саша на числовой прямой отметил все целые точки. Сергей соединяет числа a и b дугой, если расстояние между ними — простое число. Какое наименьшее количество цветов необходимо Саше для окраски всех точек, чтобы любые два числа, которые соединил дугой Сергей, были покрашены в разные цвета?

Решение

Покажем, что меньше чем 4 цветами не обойтись, так как числа $0; 2; 5; 7$ попарно соединены дугами, а поэтому все должны быть окрашены в разные цвета. Таким образом необходимо не менее 4 цветов. А теперь просто покрасим числа вида $4n + k$, $k \in \{0, 1, 2, 3\}$ в цвет k. Тогда разность между какими-либо двумя одноцветными числами равна $4m$ и не является простой, поэтому дугой они не соединены.

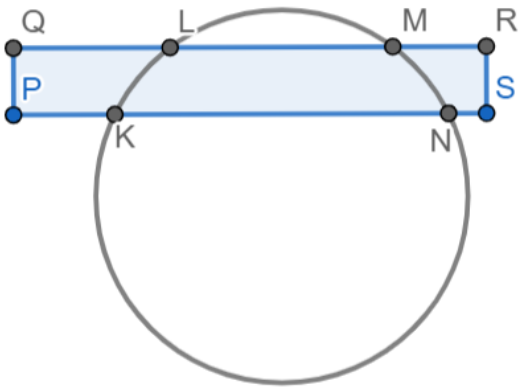
Ответ: 4.

Третья попытка. Задачи 10–11 класса

Задача I.2.6.1. (15 баллов)

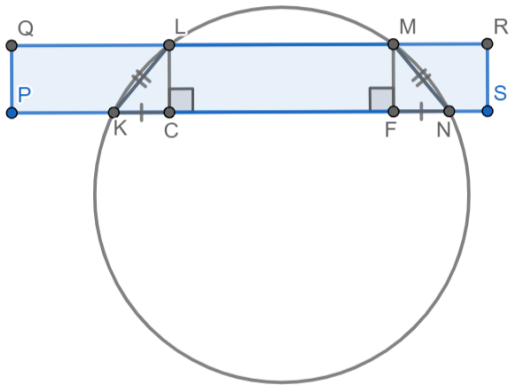
Темы: планиметрия.

Окружность пересекает прямоугольник $PQRS$ так, как показано на рисунке. Известно, что $PK = 5$, $QL = 7$ и $LM = 4$. Найдите KN.



Решение

$KLMN$ — вписанная трапеция, поэтому она равнобедренная. Опустим перпендикуляры LC и MF на прямую PS, тогда прямоугольные треугольники KCL и NFM равны (по гипотенузе и катету). Т. к. $PQLC$ — прямоугольник, то $QL = PC$, отсюда $KC = PC - PK = 7 - 5 = 2$. Окончательно $KN = KC + CF + FN = 8$.



Ответ: 8.

Задача I.2.6.2. (15 баллов)

Темы: числа.

Счетовод Сергей для каждого n от 1 до 2021 считает сумму первых n четных чисел и записывает на доске. Экономист Саша для экономии места после того, как Сергей записывает очередное число, стирает у него все цифры, кроме цифры в разряде единиц. Какова будет сумма чисел на доске?

Решение

Заметим, что Сергей вместе с Сашей для числа n напишет на доску последнюю цифру числа $n(n + 1)$. Эти последние цифры, как нетрудно убедиться перебором, периодичны с периодом 5. Период имеет вид 2, 6, 2, 0, 0. 2021 число содержит 404 периода с суммой в каждом 10 и еще число 2.

Ответ: 4042.

Задача I.2.6.3. (20 баллов)

Темы: функциональные уравнения.

Функция $f(x)$ при любых x принимает положительные значения, причем для любых x и y выполнено соотношение $f(x - y) = \dfrac{f(x)}{f(y)}$. Найдите значение $f(2021)$, если $f(-2021) = 25$.

Решение

Подставив в формулу $f(x-y)=\dfrac{f(x)}{f(y)}$ значения $x=y=1$, получим: $f(0)=\dfrac{f(1)}{f(1)}=1$.

Теперь подставим $x=0$ и $y=-2021$: $f(2021)=\dfrac{f(0)}{f(-2021)}=\dfrac{1}{25}=0,04$.

Ответ: 0,04.

Задача I.2.6.4. (20 баллов)

Темы: вероятность.

Вероятность того, что компьютер успешно загрузится в течение 10 секунд после включения, равна 0,7 (вероятность успешной загрузки при каждом включении одинакова). Саша и Сергей тестируют работу компьютера путем многократных включений/выключений и каждый раз записывают, успел ли компьютер загрузиться за 10 секунд. Тестирование идет до тех пор, пока число либо успешных, либо неуспешных загрузок достигнет шести (не обязательно подряд). В первом случае компьютер будет считаться годным, а во втором — неисправным.

После первых семи включений компьютер успешно загрузился три раза (и, соответственно, четырежды не загрузился) за отведенные 10 секунд.

Какова вероятность, что компьютер будет признан годным? Ответ запишите в виде десятичной дроби.

Решение

Обозначим за $p=0,7$ вероятность успешной загрузки компьютера при включении, тогда $q=1-p=0,3$ — вероятность того, что компьютер не загрузится при очередном включении.

Чтобы компьютер был признан годным, ему нужно еще 3 раза успешно загрузиться, а для признания неисправным нужно, чтобы он не загрузился хотя бы еще 2 раза. В любом случае до окончания тестирования остается не более 4 включений.

Рассмотрим возможные цепочки успешных ($У$) и неуспешных ($Н$) включений в течение следующих 4 включений (даже если результат тестирования будет известен раньше). Например, цепочка $НННУ$ приведет к забраковке компьютера (с вероятностью q^3p), а цепочка $УНУУ$ — к признанию годным (с вероятностью p^3q).

Для признания компьютера годным нужно, чтобы буква «$У$» в цепочке встретилась 3 или 4 раза. Вероятность этого равна:

$$C_4^4p^4+C_4^3p^3q=p^3(p+4q)=0,7^3\cdot(0,7+4\cdot0,3)=0,6517.$$

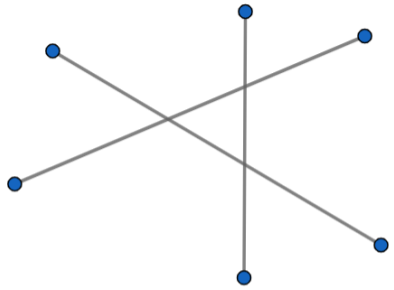
Ответ: 0,6517.

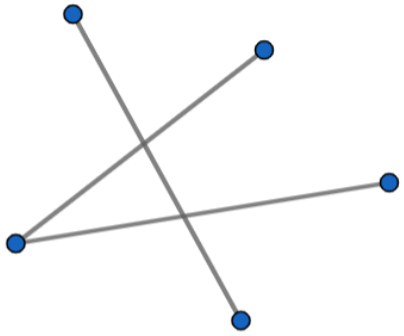
Задача I.2.6.5. (30 баллов)

Темы: планиметрия, комбинаторика.

Сергей отметил на плоскости вершины правильного 17-угольника. Сколько существует троек отрезков с концами в этих вершинах, таких, что каждый пересекает каждый (возможно, в концах)?

(На рисунках изображены некоторые возможные случаи взаимного расположения отрезков.)

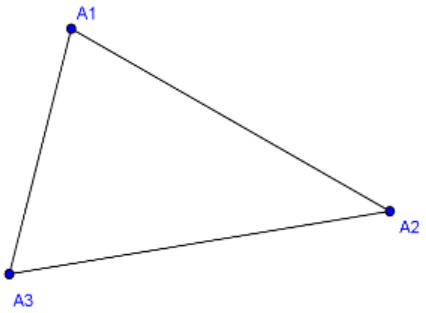




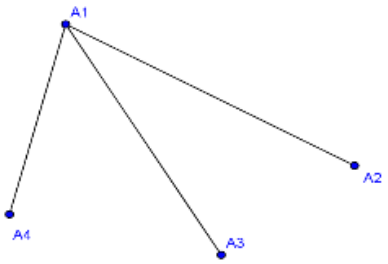
Решение

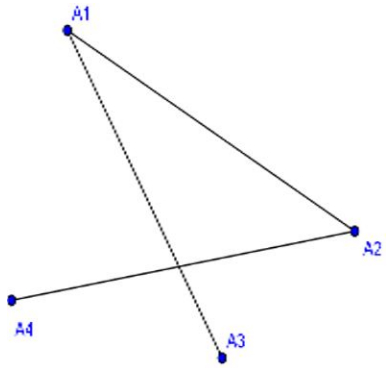
Три отрезка могут опираться на 3, 4, 5 или 6 точек. Разберем эти случаи.

1. Отрезки опираются ровно на 3 точки, которые можно выбрать C_{17}^3 способами. Соединить их отрезками можно единственным способом.

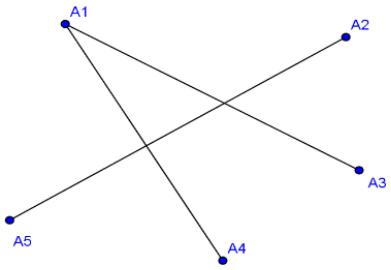


2. Отрезки опираются ровно на 4 точки. Из одной точки (пусть из точки A_1) обязательно должны выходить два отрезка, но есть две возможности для оставшихся отрезков, показанные на рисунке. Для каждого выбора четырех точек, которых C_{17}^4, существует по 8 способов их соединить отрезками.

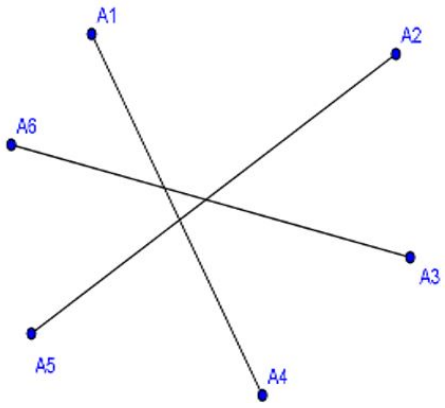




3. Отрезки опираются ровно на 5 точек. В этом случае ровно два отрезка имеют общую вершину, третий отрезок соединяет две оставшихся. Для каждого выбора пяти точек, которых C_{17}^5, существуют пять вариантов (по количеству точек, в которых сходятся два отрезка) проведения отрезков.



4. Отрезки опираются ровно на 6 точек. Для любого выбора шести точек, которых C_{17}^6, существует ровно один способ проведения отрезков, поскольку каждый отрезок должен оставлять по две точки по разные стороны от него.



Всего способов проведения отрезков будет

$$C_{17}^3 + C_{17}^4 \cdot 8 + C_{17}^5 \cdot 5 + C_{17}^6 = \frac{17 \cdot 16 \cdot 15}{2 \cdot 3} + \frac{17 \cdot 16 \cdot 15 \cdot 14 \cdot 8}{2 \cdot 3 \cdot 4} + \frac{17 \cdot 16 \cdot 15 \cdot 14 \cdot 13 \cdot 5}{2 \cdot 3 \cdot 4 \cdot 5} +$$

$$+ \frac{17 \cdot 16 \cdot 15 \cdot 14 \cdot 13 \cdot 12}{2 \cdot 3 \cdot 4 \cdot 5 \cdot 6} = \frac{17 \cdot 16 \cdot 15}{2 \cdot 3} \left(1 + 28 + \frac{14 \cdot 13}{4} + \frac{14 \cdot 13 \cdot 12}{4 \cdot 5 \cdot 6}\right) = 63036.$$

Ответ: 63036.

Четвертая попытка. Задачи 8–9 класса

Задача I.2.7.1. (15 баллов)

Темы: алгебра, преобразования.

Известно, что $\dfrac{3y+x}{3y-x}+\dfrac{3y-x}{3y+x}=\dfrac{41}{20}$. Чему равно значение выражения $\dfrac{2x^2+y^2}{x^2-y^2}$?

Ответ запишите в виде десятичной дроби.

Решение

Приведем выражение в левой части к общему знаменателю:

$$\frac{(3y+x)^2+(3y-x)^2}{9y^2-x^2}=\frac{41}{20}.$$

Отсюда $40(9y^2+x^2)=41(9y^2-x^2)$, или $y^2=9x^2$. Тогда:

$$\frac{2x^2+y^2}{x^2-y^2}=\frac{2x^2+9x^2}{x^2-9x^2}=-\frac{11}{8}=-1,375.$$

Ответ: $-1,375$.

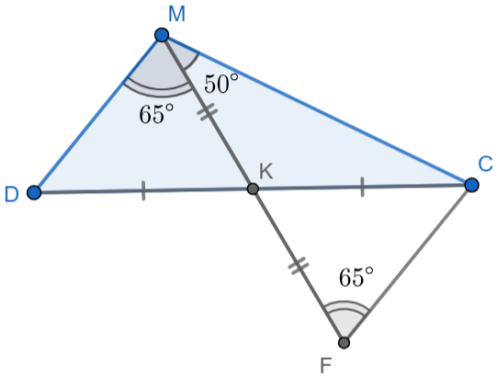
Задача I.2.7.2. (15 баллов)

Темы: геометрия, медиана.

В треугольнике MCD медиана MK вдвое меньше стороны CM и образует с ней угол 50°. Найдите разность углов DMK и CMK. Ответ запишите в градусах (знак градуса вводить не нужно, только число).

Решение

1. На продолжении луча MK за точку K отложим отрезок $KF=MK$. Тогда треугольник MCF равнобедренный ($MF=MC$), поэтому $\angle MCF=\angle MFC=$ $=\dfrac{180^\circ-\angle FMC}{2}=65^\circ$.
2. Треугольники DKM и CKF равны по двум сторонам и углу между ними ($DK=CK$ по условию, $MK=FK$ по построению, $\angle DKM=\angle CKF$ как вертикальные), поэтому $\angle DMK=\angle CFK=65^\circ$.
3. Окончательно имеем: $\angle DMK=\angle CMK=65^\circ-50^\circ=15^\circ$.



Ответ: 15.

Задача I.2.7.3. (20 баллов)

Темы: неравенства, алгебра.

Несколько бобров строили плотину на ручье из веток. Самый старательный бобр собрал $\frac{1}{10}$ всех веток, а самый ленивый — $\frac{1}{12}$ всех веток. Сколько бобров строили плотину?

Решение

Пусть бобров было x, и в сумме они собрали N веток. Тогда самый старательный собрал $\frac{N}{10}$ веток, а самый ленивый — $\frac{N}{12}$. В среднем каждый бобр принес $\frac{N}{x}$ веток. Очевидно, что $\frac{N}{12} < \frac{N}{x} < \frac{N}{10}$, тогда $10 < x < 12$. Поскольку x — целое число, $x = 11$.

Пример. Пусть всего было 540 веток, и самый результативный принес 54, самый ленивый — 45, а остальные девять — по 49.

Ответ: 11.

Задача I.2.7.4. (20 баллов)

Темы: комбинаторика.

В пятницу у Сергея 6 уроков: алгебра, геометрия, русский язык, история, химия и физкультура. Сколькими способами можно составить расписание на пятницу при условии, что до физкультуры должны быть алгебра и геометрия (в каком-то порядке)?

Решение

Обозначим предметы буквами А, Г, Р, И, Х и Ф соответственно. Ф может быть третьим, четвертым, пятым или шестым. Разберем эти варианты.

1. Если Ф будет третьей, то перед ней должны быть А и Г — 2 варианта расстановки, после Ф будут Р, И и Х — 6 вариантов. Всего 12 вариантов расписания.

2. Если Ф будет четвертой, то перед ней будут А и Г и что-то из Р, И и Х. Для выбора Р, И или Х — 3 варианта, расстановка этих трех предметов — 6 вариантов. Расстановка двух предметов после Ф — 2 варианта. Всего 36 вариантов расписания.
3. Если Ф будет пятой, то перед ней будут А и Г и еще пара предметов из Р, И и Х. Для выбора этой пары из Р, И и Х — 3 варианта, расстановка этих четырех предметов — 24 варианта. Всего 72 варианта расписания
4. Если Ф завершает день, то перед ней 5 предметов, для которых есть $5! = 120$ вариантов составить расписание.

Всего $12 + 36 + 72 + 120 = 240$ различных списков.

Ответ: 240.

Задача I.2.7.5. (30 баллов)

Темы: делимость, целые числа.

На какую наибольшую степень числа 2021 делится число $2021! = 1 \cdot 2 \cdot \ldots \cdot 2021$?

Решение

$2021 = 43 \cdot 47$ (43 и 47 — простые числа). Посмотрим, сколько раз в разложении числа 2021! на простые множители встречаются числа 43 и 47.

1. число 43 встречается 48 раз (47 раз за счет делимости чисел 43, 86, 129 и т. д. вплоть до 2021, и один дополнительный раз за счет числа 43^2).
2. число 47 встречается 43 раза.

Итак, $2021! = 43^{48} \cdot 47^{43} \cdot k$, где число k взаимно просто с 2021. Поэтому $2021! = (43 \cdot 47)^{43} \cdot t = 2021^{43} \cdot t$, где число t не кратно 2021.

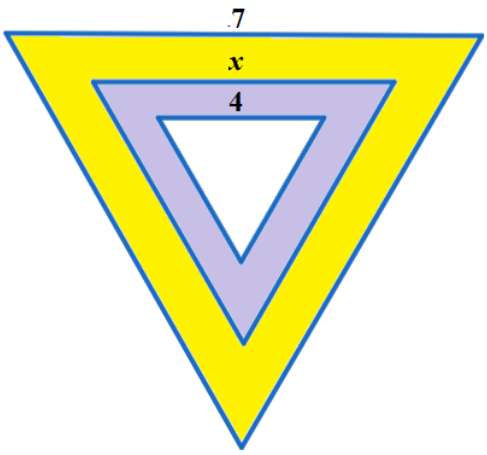
Ответ: 43.

Четвертая попытка. Задачи 10–11 класса

Задача I.2.8.1. (15 баллов)

Темы: планиметрия, площадь.

Сергей придумал эмблему нового прибора в виде трех правильных треугольников, имеющих общий центр и попарно параллельные стороны. При этом Сергей хочет добиться, чтобы отношение площади сиреневой (внутренней) части эмблемы к площади желтой (внешней) части составляло 2 : 5. Какой длины должна быть сторона среднего треугольника, если стороны внутреннего и внешнего треугольников равны 4 см и 7 см соответственно? Ответ в сантиметрах округлите до двух знаков после запятой.



Решение

Как известно, площадь правильного треугольника со стороной a равна $\frac{a^2\sqrt{3}}{4}$. Если обозначить неизвестную сторону среднего треугольника за x см, получим уравнение:

$$\frac{(x^2-4^2)\sqrt{3}}{4} : \frac{(7^2-x^2)\sqrt{3}}{4} = 2:5.$$

Отсюда $5(x^2-16) = 2(49-x^2)$, тогда $x = \sqrt{\frac{178}{7}} \approx 5,04$ (см).

Ответ: 5,04.

Задача I.2.8.2. (15 баллов)

Темы: вероятность, комбинаторика.

Саша забыл 4-значный код от банковской карты. Он помнит, что все цифры в коде различные и нечетные. Еще он помнит, что сумма каких-то двух цифр равна сумме оставшихся цифр. Какова вероятность, что Саша наберет верный код с первой попытки? Ответ запишите в виде десятичной дроби с округлением до двух знаков после запятой.

Решение

Существуют только три группы из четырех различных нечетных цифр, для которых сумма двух из них равна сумме оставшихся:

$$1,3,5,7\ (1+7=3+5)$$

$$1,3,7,9\ (1+9=3+7)$$

$$3,5,7,9\ (3+9=5+7)$$

Количество комбинаций в каждой группе равно $4! = 24$, то есть всего возможных комбинаций 72, из которых только одна правильная, то есть искомая вероятность равна $\frac{1}{72} \approx 0,013 \approx 0,01$.

Ответ: 0,01.

Задача I.2.8.3. (20 баллов)

Темы: функции.

Найдите наименьшее значение выражения:

$$8x^2 + 2y^2 + z^2 + 4xy - 2zy - 12x - 2y - 2z + 9.$$

Решение

Запишем исходное выражение в виде квадратного трехчлена относительно переменной z:

$$z^2-2z(y+1)+8x^2+2y^2+4xy-12x-2y+9 = (z-y-1)^2+y^2+8x^2+4xy-12x-4y+8 =$$

$$= (z - y - 1)^2 + (y + 2x - 2)^2 + (2x - 1)^2 + 3.$$

Минимальным это выражение будет в случае равенства 0 всех выражений под квадратами. Это достигается при $x = 0,5$, $y = 1$, $z = 2$. Минимальное значение равно 3.

Ответ: 3.

Задача I.2.8.4. (20 баллов)

Темы: делимость, целые числа.

Саша купил 20 пирожков с мясом и 21 пирожок с капустой, потратив все деньги, которые были у него в кошельке. Проанализировав покупку, Саша понял, что цены на пирожки могли быть только такими, чтобы он мог потратить все свои деньги и купить то же количество пирожков каждого вида (то есть если бы цены были какими-нибудь другими, то Саша бы не смог потратить то количество денег, что он потратил в итоге, и одновременно с этим купить то же самое количество пирожков каждого вида). Известно, что пирожок каждого вида стоит целое положительное число рублей.

Какое наибольшее количество рублей могло быть у Саши в кошельке?

Решение

Пусть пирожок с мясом стоит a рублей, пирожок с капустой — b рублей, а в кошельке у Саши s рублей. Тогда $s = 20a + 21b$, причем известно, что такие натуральные числа a и b единственны. Докажем, что наибольшее возможное значение s равно 840.

Оценка. Если $s > 2 \cdot 420 = 840$, то стоимость всех пирожков хотя бы одного из видов (с мясом или с капустой) больше 420 рублей. Пусть пирожки с мясом стоят больше 420 рублей, тогда $a > 21$. Если взять $c = a - 21$ и $d = b + 20$, мы получим альтернативный способ потратить все Сашины деньги. В самом деле, если пирожок с мясом будет стоить c рублей, а пирожок с капустой — d рублей, то общая сумма составит $20c + 21d = 20(a - 21) + 21(b + 20) = 20a + 21b = s$. Если же пирожки с капустой стоят больше 420 рублей, то $b > 20$ и можно взять $c = a + 21$, $d = b - 20$, причем $20c + 21d = 20a + 21b = s$.

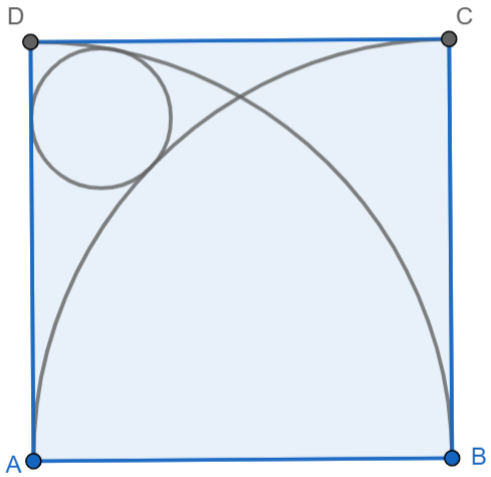
Пример. Если $s = 840$, то уравнение $20a + 21b = 840$ имеет ровно одно решение в натуральных числах: $a = 21$, $b = 20$. В самом деле, перепишем уравнение в виде $20k + b = 840$, где $k = a + b$, тогда $b = 20(42 - k)$, т. е. $b \vdots 20$, а поскольку $b \in \mathbb{N}$, то $b \geqslant 20$. Аналогично, переписав уравнение в виде $21k - a = 840$, получим $a = 21(k - 40)$, поэтому $a \vdots 21$, а с учетом того, что $a \in \mathbb{N}$, $a \geqslant 21$. Тогда $20a + 21b \geqslant 20 \cdot 21 + 21 \cdot 20 = 840$, причем равенство возможно только при наименьших значениях: $a = 21$, $b = 20$.

Ответ: 840.

Задача I.2.8.5. (30 баллов)

Темы: планиметрия, окружность.

В квадрате $ABCD$ проведены дуги с центрами A и B и радиусом, равным стороне квадрата. Окружность радиуса 2 касается стороны AD и двух данных дуг (см. рисунок). Найдите сторону квадрата.

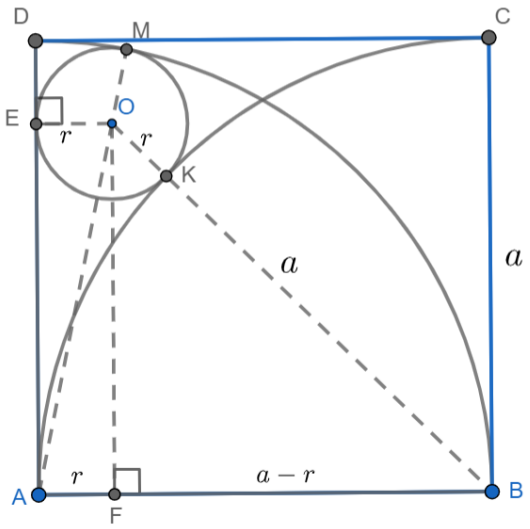


Решение

Обозначим сторону квадрата за a, радиус вписанной окружности за r. Пусть точки касания E, M, K. Также опустим перпендикуляр OF на сторону AB.

1. Поскольку точка касания двух окружностей находится на прямой, проходящей через центры окружностей, точки A, O и M лежат на одной прямой, поэтому $AO = AM - MO = a - r$. По теореме Пифагора для прямоугольного треугольника AFO: $OF^2 = AO^2 - AF^2$.

2. Аналогично, точки B, K и O лежат на одной прямой, поэтому $BO = BK + KO = a + r$. По теореме Пифагора для прямоугольного треугольника BFO: $OF^2 = BO^2 - BF^2$.
3. Из пп. 1 и 2 следует, что $AO^2 - AF^2 = BO^2 - BF^2$, или $(a-r)^2 - r^2 = (a+r)^2 - (a-r)^2$. Отсюда $a = 6r$. Подставляя значение $r = 2$, получим $a = 12$.



Ответ: 12.

Второй отборочный этап

Индивидуальная часть

Задача II.1.1. Сегментация рукописного текста на фото (100 баллов)

В финале НТО участникам необходимо создать алгоритм, который решает задачу распознавания рукописного текста на фото. Данная задача — первый шаг для создания итогового алгоритма.

Условие

Разработать алгоритм, который будет определять, относится пиксель изображения к тексту или нет. Участники работают с фотографиями разлинованных тетрадных листов с рукописным текстом.

Предсказанием для входного изображения является бинарная маска размера исходного изображения, в которой «1» означает, что данный пиксель относится к тексту, «0» — к фону. Таким образом, в датасете всего один класс — `text`.

Формат решения

В проверяющую систему необходимо отправить код алгоритма, запакованный в `ZIP-архив`. Решение запускается в изолированном окружении при помощи `Docker`. Время и ресурсы во время тестирования ограничены.

В корне архива обязательно должен быть файл `metadata.json` со структурой:

```
{
  "image": "<docker image>",
  "entry_point": "<entry point or sh script>"
}
```

Например:

```
{
   "image": "maloyan/ai-nto-task1:0.0.1",
   "entry_point": "python run.py"
}
```

Здесь:

- `image` — поле с названием `docker`-образа, в котором будет запускаться решение;
- `entrypoint` — команда, при помощи которой запускается скрипт инференса.

Решение запускается в `Docker`-контейнере. Можно воспользоваться образом от разработчиков «`maloyan/ai-nto-task1:0.0.1`».

Даны `dockerfile` https://clc.to/NTO21-22Dockerfile и `requirements.txt` https://clc.to/NTO21-22Requirements для сборки данного образа. В нем предустановлены CUDA 10.1 и актуальные версии Python библиотек для запуска бейзлайна. При необходимости участники могли использовать свой образ, выложив его на https://hub.docker.com.

Вебинар о том, как собрать докер: https://www.youtube.com/watch?v=KwVfwnlzLpE.

Доступные ресурсы:

- 8 ядер CPU;
- 48 Gb RAM;
- видеокарта NVidia Tesla V100.

Ограничения:

- 5 Gb на архив с решением;
- 15 минут на работу решения.

Структура данных

В контейнер помещается папка `images`, в которой находятся изображения, на которых необходимо сделать предсказания. Модель должна сформировать файл предсказания, например `prediction.npz`.

Пути к данным (полный путь к папке `images`) и файл, в котором необходимо сохранить результат (путь, куда нужно сохранить файл формата `.npz`), передаются как первые два аргумента при запуске решения участника. Их можно считать с помощью `sys.argv[1:]`.

Пример решения

Участникам доступны базовое решение от разработчиков задачи и пример архива для отправки на платформу соревнования для проверки.

Вебинар с разбором базового решения: https://youtu.be/mlm2xAg-JpY.

Для сегментации рукописного текста в бейзлайне используется фреймворк Detectron2 и архитектура Mask-RCNN. Для запуска бейзлайна необходимо скачать данные для обучения и положить их в папку data. Должна получиться структура вида:

- `baseline.ipynb`
- `data`
    - `train`
        - `images`
        - `annotations.json`
        - `binary.npz`

В архиве `sample_submit.zip` https://clc.to/NTO21-22submit.zip содержатся примеры загружаемого решения.

В загружаемом архиве должны быть следующие файлы:

- `metadata.json` — обязательный файл для каждого решения; в нем должны быть указаны пути к образу и скрипту выполнения модели;
- `run.py` — основной скрипт для инференса модели;
- `model_final.pth` — веса модели, которые подгружаются во время исполнения скрипта `run.py`.

Исходные данные

`baseline.ipynb`	https://clc.to/NTO21-22Baseline
`train_data.zip`	https://clc.to/NTO21-22train_data.zip
`evaluate.py`	https://clc.to/NTO21-22evaluate.py
`sample_submit_zeros.zip`	https://clc.to/NTO21-22zeros
`Dockerfile`	https://clc.to/NTO21-22Dockerfile
`requirements.txt`	https://clc.to/NTO21-22Requirements
`sample_submit.zip`	https://clc.to/NTO21-22submit.zip

Метрика

Оценка загружаемых решений проводится автоматически на платформе Open Data Science https://ods.ai/. Для оценки качества решений используется метрика `F1-score`:

$$F1 = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision}.$$

`F1-score` вычисляется на основе значений точности ($Precision$) и полноты ($Recall$), которые, в свою очередь, зависят от набора статистик по прогнозам — истинно-положительный результат ($TruePositive$), ложно-положительный результат ($FalsePositive$) и ложно-отрицательный результат ($FalseNegative$):

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive};$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}.$$

Сначала `F1-score` считается для каждого изображения из тестового набора. Для этого попиксельно сравниваются две маски — верная и предсказанная. Затем `F1-score` усредняются по всем изображениям.

Критерии оценивания

Тестовый датасет был разделен на 2 примерно равные части: публичную и приватную. В ходе соревнования участникам был доступен рейтинг, построенный на основании результатов проверки публичной части данных. В ходе соревнования участники могли выбрать 2 лучших, по их мнению, решения, которые шли для проверки на публичной части данных и включения в итоговый рейтинг. В случае, если участником не были выбраны никакие решения, автоматически выбиралось лучшее решение по публичной части данных.

После формирования итогового рейтинга на основании проверки лучших решений на приватной части данных по решениям с полным совпадением результатов проводилась проверка на плагиат путем запроса кода решения у участников и дальнейшего анализа присланного кода и загруженного на платформу архива. Результаты решений, которые не прошли проверку, были аннулированы.

Результаты полученного рейтинга были переведены в 100-балльную систему в соответствии с формулой:

$$Total = \frac{Result \cdot (RankParticipants + 1 - RankPlace) \cdot 100}{MaxResult \cdot Participants},$$

где $Total$ — итоговый балл за решение задачи;

$Result$ — результат решения задачи;

$RankParticipants$ — общее количество участников в рейтинге;

$RankPlace$ — место участника в рейтинге;

$MaxResult$ — максимальный результат решения задачи среди всех участников;

$Participants$ — общее количество участников.

Решение

Базовое решение от разработчиков: https://clc.to/NTO21-22Baseline.

Пример архива для загрузки на платформу: https://clc.to/NTO21-22submit.zip.

Сегментация тетрадей

Рассмотрим модель сегментации текста в школьных тетрадях с помощью фреймворка Detectron2. Можно также использовать другие модели (например, `UNET`, `mmdet`) или написать свою.

Установка библиотек

Устанавливаем библиотеки, под которыми запускается данное решение:

```
#!pip install torch==1.6.0+cu101 torchvision==0.7.0+cu101 -f
↪  https://download.pytorch.org/whl/torch_stable.html

#!python -m pip install detectron2 -f
↪  https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/torch1.6/index.html

#!pip install tensorflow==2.1.0

#!pip install opencv-python
```

Загружаем необходимые библиотеки для создания и обучения модели:

```
import cv2
import random
import json
```

```
import os

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.filterwarnings("ignore")
import ipywidgets as widgets
from ipywidgets import interact, interact_manual
import shutil

import tqdm

from matplotlib import pyplot as plt

import numpy as np

import torch, torchvision
import detectron2
from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog,DatasetCatalog
from detectron2.data.datasets import register_coco_instances,load_coco_json
from detectron2.data import detection_utils as utils
from detectron2.engine import DefaultTrainer
from detectron2.engine import HookBase

# from detectron2.utils.logger import setup_logger
# setup_logger()

import logging
logger = logging.getLogger('detectron2')
logger.setLevel(logging.CRITICAL)
```

Прежде чем переходить к загрузке данных проверим, доступны ли GPU-мощности:

```
print('GPU: ' + str(torch.cuda.is_available()))
```

Валидационный датасет

Для валидации моделей полезно создать из обучающих данных валидационный датасет. Для этого разделим исходный датасет на две части — для обучения и для валидации. С этой целью создадим два новых файла с аннотациями, куда раздельно запишем исходную информацию об аннотациях:

```
import json
#Подгрузим аннотации train
with open('data/train/annotations.json') as f:
    annotations = json.load(f)

#Пустой словарь для аннотаций валидации
annotations_val = {}
#Список категорий такой же как в train
annotations_val['categories'] = annotations['categories']

#Пустой словарь для аннотаций нового train
annotations_train = {}
```

```
#Список категорий такой же как в train
annotations_train['categories'] = annotations['categories']

#Положим в валидацию каждое 10 изображение из исходного train, а остальные - в новый
↪  train
annotations_val['images'] = []
annotations_train['images'] = []
for num,img in enumerate(annotations['images']):
    if num%10==0:
        annotations_val['images'].append(img)
    else:
        annotations_train['images'].append(img)

#Положим в список аннотаций валидации только те аннотации, которые относятся к
↪  изображениям из валидации.
#А в список аннотаций нового train - только те, которые относятся к нему
val_img_id = [i['id'] for i in annotations_val['images']]
train_img_id = [i['id'] for i in annotations_train['images']]

annotations_val['annotations'] = []
annotations_train['annotations'] = []

for annot in annotations['annotations']:
    if annot['image_id'] in val_img_id:
        annotations_val['annotations'].append(annot)
    elif annot['image_id'] in train_img_id:
        annotations_train['annotations'].append(annot)
    else:
        print('Аннотации нет ни в одном наборе')
```

Аннотации для валидации и новой обучающей выборки готовы, теперь сохраним их в формате `json` и положим в папке. Назовем аннотации `annotations_new.json`, чтобы новый набор аннотаций для `train` (без множества `val`) не перезаписал исходные аннотации:

```
if not os.path.exists('data/val'):
    os.makedirs('data/val')
if not os.path.exists('data/val/images'):
    os.makedirs('data/val/images')
```

Скопируем изображения, которые относятся к валидации, в папку `val/images`:

```
for i in annotations_val['images']:
    shutil.copy('data/train/images/'+i['file_name'],'data/val/images/')
```

Запишем новые файлы с аннотациями для `train` и `val`:

```
with open('data/val/annotations_new.json', 'w') as outfile:
    json.dump(annotations_val, outfile)

with open('data/train/annotations_new.json', 'w') as outfile:
    json.dump(annotations_train, outfile)
```

Регистрация датасета

Зарегистрируем выборки в Detectron2 для дальнейшей подачи на обучение модели:

```
for d in ['train','val']:
    DatasetCatalog.register("my_dataset_"+d, lambda d=d:
    ↪  load_coco_json("./data/{}/annotations_new.json".format(d),
    image_root= "./data/train/images",\
    dataset_name="my_dataset_"+d,extra_annotation_keys=['bbox_mode']))
```

После регистрации можно загружать выборки, чтобы иметь возможность посмотреть на них. Первой загрузим обучающую выборку в `dataset_dicts_train`:

```
dataset_dicts_train = DatasetCatalog.get("my_dataset_train")
train_metadata = MetadataCatalog.get("my_dataset_train")
```

И тестовую выборку в `dataset_dicts_val`:

```
dataset_dicts_val = DatasetCatalog.get("my_dataset_val")
val_metadata = MetadataCatalog.get("my_dataset_val")
```

Посмотрим на размер получившихся выборок — эта операция в Python осуществляется при помощи функции `len()`:

```
print('Размер обучающей выборки (Картинки): {}'.format(len(dataset_dicts_train)))
print('Размер тестовой выборки (Картинки): {}'.format(len(dataset_dicts_val)))
```

Имеем 588 изображений для тренировки и 66 — для проверки качества.

Посмотрим на размеченные фотографии из валидации:

```
import os
from IPython.display import Image
@interact
def show_images(file=range(len(dataset_dicts_val))):
    example = dataset_dicts_val[file]
    image = utils.read_image(example["file_name"], format="RGB")
    plt.figure(figsize=(3,3),dpi=200)
    visualizer = Visualizer(image[:, :, ::-1], metadata=val_metadata, scale=0.5)
    vis = visualizer.draw_dataset_dict(example)
    plt.imshow(vis.get_image()[:, :,::-1])
    plt.show()
```

Определяем конфигурацию модели

Прежде чем начать работать с самой моделью, нужно определить ее параметры и спецификацию обучения.

Создаем конфигурацию и загружаем архитектуру модели с предобученными на COCO весами для распознавания объектов. COCO — датасет, содержащий 80 популярных категорий объектов и более 300 000 изображений.

```
cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/ \
mask_rcnn_R_50_FPN_3x.yaml"))
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-InstanceSegmentation/ \
mask_rcnn_R_50_FPN_3x.yaml")
```

Можно посмотреть также другие архитектуры https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md.

Задаем параметры самой модели и обучения модели:

```
# Загружаем названия обучающией и тестовой выборок в настройки
cfg.DATASETS.TRAIN = ("my_dataset_train",)
cfg.DATASETS.TEST = ("my_dataset_val",)

# Часто имеет смысл сделать изображения чуть меньшего размера, чтобы
# обучение происходило быстрее. Поэтому мы можем указать размер, до которого будем
↪  изменяться наименьшая
# и наибольшая из сторон исходного изображения.
cfg.INPUT.MIN_SIZE_TRAIN = 300
cfg.INPUT.MAX_SIZE_TRAIN = 300

# Также мы должны сказать модели ниже какой вероятности определения она игнорирует
↪  результат.
# То есть, если она найдет на картинке еду, но вероятность правильного определения
↪  ниже 0.5,
# то она не будет нам сообщать, что она что-то нашла.
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5

# Также мы должны указать порядок каналов во входном изображении. Обратите внимание,
↪  что это Blue Green Red (BGR),
# а не привычный RGB. Это особенности работы данной модели.
cfg.INPUT.FORMAT = 'BGR'

# Для более быстрой загрузки данных в модель, мы делаем параллельную загрузку. Мы
↪  указываем параметр 4,
cfg.DATALOADER.NUM_WORKERS = 4

# Следующий параметр задает количество изображений в батче, на котором
# модель делает одну итерацию обучения (изменения весов).
cfg.SOLVER.IMS_PER_BATCH = 20

# Зададим также learning_rate
cfg.SOLVER.BASE_LR = 0.01

# Укажем модели, через сколько шагов обучения модели следует уменьшить learning rate
cfg.SOLVER.STEPS = (1500,)

# Фактор, на который уменьшается learning rate задается следующим выражением
cfg.SOLVER.GAMMA = 0.1

# Зададим общее число итераций обучения.
cfg.SOLVER.MAX_ITER = 3000

# Укажем количество классов в нашей выборке
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1

# Задаем через сколько  шагов обучения сохранять веса модели в файл. Этот файл мы
↪  сможем загрузить потом
# для тестирования нашей обученной модели на новых данных.
cfg.SOLVER.CHECKPOINT_PERIOD = 1000

# И указываем название папки, куда сохранять чекпойнты модели и информацию о процессе
↪  обучения.
cfg.OUTPUT_DIR = './output'

# Если вдруг такой папки нет, то создадим ее
```

```
os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)

# Если мы хотим удалить чекпойнты предыдущих моделей, то выполняем данную команду.
#%rm output/*
```

Обучаем модель

Процесс обучения модели запускают следующие строки кода. Возникающие предупреждения игнорируем — это информация об обучении.

```
trainer = DefaultTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()
```

Используем обученную модель для проверки качества на валидации:

```
cfg.MODEL.WEIGHTS = "output/model_0002999.pth"

cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.5
cfg.DATASETS.TEST = ("my_dataset_val", )
#Изменение размера исходных изображений для тестового датасета
cfg.INPUT.MIN_SIZE_TEST= 300
cfg.INPUT.MAX_SIZE_TEST = 300
cfg.INPUT.FORMAT = 'BGR'

#важно увеличить это значение (стандартное равно 100). Так как на листе тетради может
↪  быть довольно много слов
cfg.TEST.DETECTIONS_PER_IMAGE = 1000

predictor = DefaultPredictor(cfg)
```

Сделаем предсказания для тестового датасета и нарисуем его.

Можно выбрать из выпадающего списка номер изображения и посмотреть разметку на всем валидационном датасете:

```
@interact
def show_images(file=range(len(dataset_dicts_val))):

    example = dataset_dicts_val[file]
    im = cv2.imread(example["file_name"])
    outputs = predictor(im)
    fig, axs = plt.subplots(nrows=1,ncols=2,figsize=(4,4),dpi=200)
    v = Visualizer(im[:, :],
                   metadata=val_metadata,
                   scale=0.4 )
    v = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    axs[0].imshow(im[:, :, ::-1])
    axs[1].imshow(v.get_image()[:, :, ::-1])
    axs[0].axis('off')
    axs[1].axis('off')
    axs[0].set_title('Original')
    axs[1].set_title('Predict')
    plt.show()
```

Можно непосредственно в коде изменить номер изображения, которое хотим обработать:

```
id_image_selected = 3
example = dataset_dicts_val[id_image_selected]
im = cv2.imread(example["file_name"])
outputs = predictor(im)
plt.figure(figsize=(7,7))
v = Visualizer(im[:, :],
               metadata=val_metadata,
               scale=0.4 )
v = v.draw_instance_predictions(outputs["instances"].to("cpu"))
plt.imshow(v.get_image()[:, :, ::-1])
plt.axis('off')
plt.show()
```

В качестве предсказаний для каждого изображения из тестового набора требуется получить бинарную маску, в которой «1» означает, что данный пиксель относится к классу текста.

На примере одного изображения переведем формат выхода Detectron2 в требуемый формат.

`outputs` — результат предсказания модели на данном изображении из предыдущего блока с кодом:

```
prediction = outputs['instances'].pred_masks.cpu().numpy()
prediction.shape
```

В `prediction` находится массив бинарных матриц. Каждая матрица отвечает за отдельную задетектированную маску текста. В данном случае модель задетектировала 80 текстовых масок, визуализируем одну из них:

```
prediction[0]
```

```
plt.imshow(prediction[0])
```

Для того чтобы получить бинарную маску со всем задетектированным текстом для изображения, нужно объединить все маски в одну. С этой целью поэлементно сложим все матрицы. Там, где после сложения остались нули — модель не задетектировала никакого текста:

```
mask = np.add.reduce(prediction)
```

```
mask = mask > 0
```

```
plt.imshow(mask)
```

Получим такую маску для каждого изображения из валидационной выборки, а затем посчитаем метрику `F1-score`:

```
#Подгрузим аннотации train
with open('data/val/annotations_new.json') as f:
    annotations_val = json.load(f)

val_images = annotations_val['images']

val_predictions = {}
```

```
for val_img in tqdm.tqdm_notebook(val_images):
    file_name = val_img['file_name']
    img_path = os.path.join('data/val/images/',file_name)
    im = cv2.imread(img_path)
    outputs = predictor(im)
    prediction = outputs['instances'].pred_masks.cpu().numpy()
    mask = np.add.reduce(prediction)
    mask = mask > 0
    val_predictions[file_name] = mask
```

Для сохранения предсказаний и загрузки бинарных масок используем формат `.npz`. Он позволяет хранить большие массивы в компактном виде. Документация доступна по ссылке: https://numpy.org/doc/stable/reference/generated/numpy.savez_compressed.html.

```
np.savez_compressed('val_pred.npz',**val_predictions)
```

Загрузим бинарные маски для `train` и `val`. Так как в начале исходный набор `train` был разбит на новый трейн и валидацию, то информация по всем маскам из исходного `train` хранится в `binary.npz`.

Получившийся после подгрузки `np.load()` имеет структуру словаря. Его ключи можно получить с помощью метода `files` — `loaded_val.files`. В нашем случае ключами являются ключи исходного словаря `val_predictions`, то есть названия изображений:

```
loaded_train = np.load('data/train/binary.npz')

loaded_val_pred = np.load('val_pred.npz')
```

Используем среднюю метрика `F1-score`, то есть считаем `F1-score` для каждого изображения, а затем усредняем результаты:

```
def f1_loss(y_true, y_pred):

    tp = np.sum(y_true & y_pred)
    tn = np.sum(~y_true & ~y_pred)
    fp = np.sum(~y_true & y_pred)
    fn = np.sum(y_true & ~y_pred)

    epsilon = 1e-7

    precision = tp / (tp + fp + epsilon)
    recall = tp / (tp + fn + epsilon)

    f1 = 2* precision*recall / ( precision + recall + epsilon)

    return f1

f1_scores = []
for key in tqdm.tqdm_notebook(loaded_val_pred.files):
    pred = loaded_val_pred[key].reshape(-1)
    true = loaded_train[key].reshape(-1)

    f1_img = f1_loss(true,pred)
    f1_scores.append(f1_img)
```

Получившаяся метрика на валидации:

```
np.mean(f1_scores)
```

Запись submission

Как было сказано выше, правильные и предсказанные маски будем хранить в компактном формате `npz`.

Возьмем обученную модель и запишем предсказания в файл `prediction.npz`:

```
from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2.structures import Boxes, BoxMode

def run(test_images_path,predictions_output_path):
    threshold = 0.5
    model_path = "./output/model_0002999.pth"

    cfg = get_cfg()
    cfg.merge_from_file(model_zoo.get_config_file("COCO-InstanceSegmentation/ \
    mask_rcnn_R_50_FPN_3x.yaml"))
    cfg.MODEL.WEIGHTS = model_path
    cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = threshold   # set the testing threshold
    ↪  for this model
    cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1
    cfg.INPUT.MIN_SIZE_TEST= 300
    cfg.INPUT.MAX_SIZE_TEST = 300
    cfg.INPUT.FORMAT = 'BGR'
    cfg.TEST.DETECTIONS_PER_IMAGE = 1000

    predictor = DefaultPredictor(cfg)
    results = {}

    for img in os.listdir(test_images_path):
        img_path = os.path.join(test_images_path,img)
        im = cv2.imread(img_path)
        outputs = predictor(im)
        prediction = outputs['instances'].pred_masks.cpu().numpy()
        mask = np.add.reduce(prediction)
        mask = mask > 0

        results[img] = mask
    np.savez_compressed(predictions_output_path,**results)
```

В функцию передаются два аргумента:

- Путь к папке с изображениями, на которых будем делать предсказания. Сейчас это путь к валидационными изображениям. При загрузке образа стоит использовать путь `/home/jovyan/input/images`. Папку `test` поместим во время запуска контейнера.
- Путь к создаваемому файлу с предсказаниями. Во время локальной отладки можно использовать любое имя, для отправки в систему стоит использовать название `prediction.npz`.

Этот код вынесен в отдельные скрипты для удобства, они должны запускаться во время запуска контейнера:

```
run('data/val/images','prediction.npz')
```

Оценка качества

Оценка качества просходит по следующему скрипту — `evaluate.py`. Он принимает на вход путь к двум файлам формата `npz`:

- `ref_path` — путь к файлу с правильными ответами;
- `pred_path` — путь к файлу с предсказаниями.

Для даидации генерируем файл формата `npz`:

```
val_true = {}
for i in loaded_train.files:
    if i in val_predictions.keys():
        val_true[i] = loaded_train[i]

np.savez_compressed('val_true.npz',**val_true)

!python evaluate.py --ref_path val_true.npz --pred_path val_pred.npz
```

Командная часть

Задача II.2.1. Распознавание рукописного текста (100 баллов)

Темы: искусственный интеллект, машинное обучение, компьютерное зрение.

Задача финала подразумевает создание алгоритма, который получает на вход фото страницы с рукописным текстом, а выдает распознанный текст. Данная задача предполагает последовательное решение задач сегментации (задача индивидуальной части) и распознавания сегментированного текста.

Условие

Участникам предстояло разработать алгоритм, который сможет распознать текст на фото. Предсказание модели — текстовая строка, соответствующая тексту на картинке.

Формат решения

В проверяющую систему необходимо было отправить код алгоритма, запакованный в ZIP-архив. Решения запускаются в изолированном окружении при помощи `Docker`. Время и ресурсы во время тестирования ограничены.

В корне архива обязательно должен быть файл `metadata.json` со структурой:

```
{
    "image": "<docker image>",
    "entry_point": "<entry point or sh script>"
}
```

Например:

```
{
    "image": "skalinin1/baseline-htr:latest",
    "entry_point": "python run.py"
}
```

Здесь `image` — поле с названием docker-образа, в котором будет запускаться решение, `entry_point` — команда, при помощи которой запускается скрипт инференса. Решение запускается в `Docker`-контейнере. Участники могли воспользоваться готовым образом «`skalinin1/baseline-htr:latest`». Подготовленные файлы:

- https://clc.to/ai-21-22-dockerfile;
- https://clc.to/ai-21-22-requirements.

При желании можно было использовать свой образ, выложив его на https://hub.docker.com.

Доступные ресурсы:

- 8 ядер CPU;
- 48 Gb RAM;
- видеокарта NVidia Tesla V100.

Ограничения:

- 5 Gb на архив с решением;
- 15 минут на работу решения.

Критерии оценивания

Оценка качества решений участников производилась автоматически на основе метрики CER (character error rate). Она считается следующим образом:

$$CER = \frac{\sum_{i=1}^{n} dict_c(pred_i, true_i)}{\sum_{i=1}^{n} len_c(true_i)}.$$

Здесь $dict_c$ – это расстояние Левенштейна, посчитанное для токенов-символов (включая пробелы), len_c – длина строки в символах.

Метрика CER изменяется от 0 до 1, где 0 — наилучшее значение, 1 — наихудшее.

Тестовый датасет был разделен на 2 примерно равные части: публичную и приватную. В ходе соревнования участникам был доступен рейтинг, построенный на основании результатов проверки публичной части данных. В ходе соревнования участники могли выбрать 2 лучших, по их мнению, решения, который шли для проверки на публичной части данных и включения в итоговый рейтинг. В случае, если участником не были выбраны никакие решения, автоматически выбиралось лучшее решение по публичной части данных.

После формирования итогового рейтинга на основании проверки лучших решений на приватной части данных по решениям с полным совпадением результатов проводилась проверка на плагиат. Проверка выполнялась путем запроса кода решения у участников и дальнейшего анализа присланного кода и загруженного на платформу архива. Результаты решений, которые не прошли проверку, были аннулированы.

Результаты полученного рейтинга были переведены в 100-балльную систему в соответствии с формулой:

$$Total = \frac{(1 - Result) \cdot (RankParticipants + 1 - RankPlace) \cdot 100}{(1 - MaxResult) \cdot Participants},$$

где $Total$ — итоговый балл за решение задачи;

$Result$ — результат решения задачи;

$RankParticipants$ — общее количество команд в рейтинге;

$RankPlace$ — место команды в рейтинге;

$MaxResult$ — лучший результат решения задачи;

$Participants$ — общее количество участников.

Решение

Базовое решение от разработчиков: https://clc.to/ai-21-22-baseline:

Установка и подгрузка библиотек

Установка библиотек, под которым запускается данный бейзлайн.

```
# !pip install numpy==1.20.3
# !pip install torch==1.9.0+cu111 torchvision==0.10.0+cu111 -f
↪  https://download.pytorch.org/whl/torch_stable.html
# !pip install opencv-python==4.5.2.52
# !pip install matplotlib==3.4.2
```

```
import torch
import torch.nn as nn
import torchvision
from torch.utils.data import Dataset
from torch.nn.utils.rnn import pad_sequence

import numpy as np
import cv2
import os
import json
from matplotlib import pyplot as plt
```

Разделение трейн датасет на обучающую и валидационную подвыборки

```
with open('data/final_data/train/labels.json') as f:
    train_data = json.load(f)

train_data = [(k, v) for k, v in train_data.items()]
print('train len', len(train_data))

split_coef = 0.75
train_len = int(len(train_data)*split_coef)

train_data_splitted = train_data[:train_len]
val_data_splitted = train_data[train_len:]
```

```
print('train len after split', len(train_data_splitted))
print('val len after split', len(val_data_splitted))


with open('data/final_data/train/train_labels_splitted.json', 'w') as f:
    json.dump(dict(train_data_splitted), f)

with open('data/final_data/train/val_labels_splitted.json', 'w') as f:
    json.dump(dict(val_data_splitted), f)


train len 66599
train len after split 49949
val len after split 16650
```

Параметры обучения

Здесь мы можем поправить конфиги обучения — задать размер батча, количество эпох, размер входных изображений, а также установить пути к датасетам:

```
DEVICE = torch.device('cuda' if torch.cuda.is_available() else 'cpu')


config_json = {
    "alphabet": "
    ↪  !\"#$%&'()*+,-./0123456789:;<=>?@[\\]^_`{|}~«»ЁАБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШ \
    ЩЪЫЬЭЮЯабвгдежзийклмнопрстуфхцчшщъыьэюяёЈ№",
    "save_dir": "data/experiments/test",
    "num_epochs": 500,
    "image": {
        "width": 256,
        "height": 32
    },
    "train": {
        "root_path": "data/final_data/train/images/",
        "json_path": "data/final_data/train/train_labels_splitted.json",
        "batch_size": 64
    },
    "val": {
        "root_path": "data/final_data/train/images/",
        "json_path": "data/final_data/train/val_labels_splitted.json",
        "batch_size": 128
    }
}
```

Определение класса датасета (torch.utils.data.Dataset) и другие вспомогательные функции

```
# функция которая помогает объединять картинки и таргет-текст в батч
def collate_fn(batch):
    images, texts, enc_texts = zip(*batch)
    images = torch.stack(images, 0)
    text_lens = torch.LongTensor([len(text) for text in texts])
    enc_pad_texts = pad_sequence(enc_texts, batch_first=True, padding_value=0)
    return images, texts, enc_pad_texts, text_lens
```

```
def get_data_loader(
    transforms, json_path, root_path, tokenizer, batch_size, drop_last
):
    dataset = OCRDataset(json_path, root_path, tokenizer, transforms)
    data_loader = torch.utils.data.DataLoader(
        dataset=dataset,
        collate_fn=collate_fn,
        batch_size=batch_size,
        num_workers=8,
    )
    return data_loader


class OCRDataset(Dataset):
    def __init__(self, json_path, root_path, tokenizer, transform=None):
        super().__init__()
        self.transform = transform
        with open(json_path, 'r') as f:
            data = json.load(f)
        self.data_len = len(data)

        self.img_paths = []
        self.texts = []
        for img_name, text in data.items():
            self.img_paths.append(os.path.join(root_path, img_name))
            self.texts.append(text)
        self.enc_texts = tokenizer.encode(self.texts)

    def __len__(self):
        return self.data_len

    def __getitem__(self, idx):
        img_path = self.img_paths[idx]
        text = self.texts[idx]
        enc_text = torch.LongTensor(self.enc_texts[idx])
        image = cv2.imread(img_path)
        if self.transform is not None:
            image = self.transform(image)
        return image, text, enc_text


class AverageMeter:
    """Computes and stores the average and current value"""
    def __init__(self):
        self.reset()

    def reset(self):
        self.avg = 0
        self.sum = 0
        self.count = 0

    def update(self, val, n=1):
        self.sum += val * n
        self.count += n
        self.avg = self.sum / self.count
```

Определение Токенайзера — вспомогательного класса, который преобразует текст в числа

Разметка-текст с картинок преобразуется в числовое представление, на которых модель может учиться. Также может преобразовывать числовое предсказание модели обратно в текст:

```
OOV_TOKEN = '<OOV>'
CTC_BLANK = '<BLANK>'


def get_char_map(alphabet):
    """Make from string alphabet character2int dict.
    Add BLANK char fro CTC loss and OOV char for out of vocabulary symbols."""
    char_map = {value: idx + 2 for (idx, value) in enumerate(alphabet)}
    char_map[CTC_BLANK] = 0
    char_map[OOV_TOKEN] = 1
    return char_map


class Tokenizer:
    """Class for encoding and decoding string word to sequence of int
    (and vice versa) using alphabet."""

    def __init__(self, alphabet):
        self.char_map = get_char_map(alphabet)
        self.rev_char_map = {val: key for key, val in self.char_map.items()}

    def encode(self, word_list):
        """Returns a list of encoded words (int)."""
        enc_words = []
        for word in word_list:
            enc_words.append(
                [self.char_map[char] if char in self.char_map
                 else self.char_map[OOV_TOKEN]
                 for char in word]
            )
        return enc_words

    def get_num_chars(self):
        return len(self.char_map)

    def decode(self, enc_word_list):
        """Returns a list of words (str) after removing blanks and collapsing
        repeating characters. Also skip out of vocabulary token."""
        dec_words = []
        for word in enc_word_list:
            word_chars = ''
            for idx, char_enc in enumerate(word):
                # skip if blank symbol, oov token or repeated characters
                if (
                    char_enc != self.char_map[OOV_TOKEN]
                    and char_enc != self.char_map[CTC_BLANK]
                    # idx > 0 to avoid selecting [-1] item
                    and not (idx > 0 and char_enc == word[idx - 1])
                ):
                    word_chars += self.rev_char_map[char_enc]
            dec_words.append(word_chars)
        return dec_words
```

Accuracy в качестве метрики

Accuracy измеряет долю предсказанных строк текста, которые полностью совпадают с таргет-текстом:

```
def get_accuracy(y_true, y_pred):
    scores = []
    for true, pred in zip(y_true, y_pred):
        scores.append(true == pred)
    avg_score = np.mean(scores)
    return avg_score
```

Аугментации

Здесь мы задаем базовые аугментации для модели. Вы можете написать свои или использовать готовые библиотеки типа `albumentations`.

```
class Normalize:
    def __call__(self, img):
        img = img.astype(np.float32) / 255
        return img


class ToTensor:
    def __call__(self, arr):
        arr = torch.from_numpy(arr)
        return arr


class MoveChannels:
    """Move the channel axis to the zero position as required in pytorch."""

    def __init__(self, to_channels_first=True):
        self.to_channels_first = to_channels_first

    def __call__(self, image):
        if self.to_channels_first:
            return np.moveaxis(image, -1, 0)
        else:
            return np.moveaxis(image, 0, -1)


class ImageResize:
    def __init__(self, height, width):
        self.height = height
        self.width = width

    def __call__(self, image):
        image = cv2.resize(image, (self.width, self.height),
                           interpolation=cv2.INTER_LINEAR)
        return image



def get_train_transforms(height, width):
    transforms = torchvision.transforms.Compose([
        ImageResize(height, width),
        MoveChannels(to_channels_first=True),
```

```
        Normalize(),
        ToTensor()
    ])
    return transforms


def get_val_transforms(height, width):
    transforms = torchvision.transforms.Compose([
        ImageResize(height, width),
        MoveChannels(to_channels_first=True),
        Normalize(),
        ToTensor()
    ])
    return transforms
```

Определение модели — CRNN

Подробнее об архитектуре можно почитать в статье https://arxiv.org/abs/1507.05717.

```
def get_resnet34_backbone(pretrained=True):
    m = torchvision.models.resnet34(pretrained=True)
    input_conv = nn.Conv2d(3, 64, 7, 1, 3)
    blocks = [input_conv, m.bn1, m.relu,
              m.maxpool, m.layer1, m.layer2, m.layer3]
    return nn.Sequential(*blocks)


class BiLSTM(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, dropout=0.1):
        super().__init__()
        self.lstm = nn.LSTM(
            input_size, hidden_size, num_layers,
            dropout=dropout, batch_first=True, bidirectional=True)

    def forward(self, x):
        out, _ = self.lstm(x)
        return out


class CRNN(nn.Module):
    def __init__(
        self, number_class_symbols, time_feature_count=256, lstm_hidden=256,
        lstm_len=2,
    ):
        super().__init__()
        self.feature_extractor = get_resnet34_backbone(pretrained=True)
        self.avg_pool = nn.AdaptiveAvgPool2d(
            (time_feature_count, time_feature_count))
        self.bilstm = BiLSTM(time_feature_count, lstm_hidden, lstm_len)
        self.classifier = nn.Sequential(
            nn.Linear(lstm_hidden * 2, time_feature_count),
            nn.GELU(),
            nn.Dropout(0.1),
            nn.Linear(time_feature_count, number_class_symbols)
        )

    def forward(self, x):
        x = self.feature_extractor(x)
```

```
        b, c, h, w = x.size()
        x = x.view(b, c * h, w)
        x = self.avg_pool(x)
        x = x.transpose(1, 2)
        x = self.bilstm(x)
        x = self.classifier(x)
        x = nn.functional.log_softmax(x, dim=2).permute(1, 0, 2)
        return x
```

Циклы трейна и валидации

```
def val_loop(data_loader, model, tokenizer, device):
    acc_avg = AverageMeter()
    for images, texts, _, _ in data_loader:
        batch_size = len(texts)
        text_preds = predict(images, model, tokenizer, device)
        acc_avg.update(get_accuracy(texts, text_preds), batch_size)
    print(f'Validation, acc: {acc_avg.avg:.4f}')
    return acc_avg.avg


def train_loop(data_loader, model, criterion, optimizer, epoch):
    loss_avg = AverageMeter()
    model.train()
    for images, texts, enc_pad_texts, text_lens in data_loader:
        model.zero_grad()
        images = images.to(DEVICE)
        batch_size = len(texts)
        output = model(images)
        output_lenghts = torch.full(
            size=(output.size(1),),
            fill_value=output.size(0),
            dtype=torch.long
        )
        loss = criterion(output, enc_pad_texts, output_lenghts, text_lens)
        loss_avg.update(loss.item(), batch_size)
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), 2)
        optimizer.step()
    for param_group in optimizer.param_groups:
        lr = param_group['lr']
    print(f'\nEpoch {epoch}, Loss: {loss_avg.avg:.5f}, LR: {lr:.7f}')
    return loss_avg.avg


def predict(images, model, tokenizer, device):
    model.eval()
    images = images.to(device)
    with torch.no_grad():
        output = model(images)
    pred = torch.argmax(output.detach().cpu(), -1).permute(1, 0).numpy()
    text_preds = tokenizer.decode(pred)
    return text_preds


def get_loaders(tokenizer, config):
    train_transforms = get_train_transforms(
        height=config['image']['height'],
        width=config['image']['width']
```

```
    )
    train_loader = get_data_loader(
        json_path=config['train']['json_path'],
        root_path=config['train']['root_path'],
        transforms=train_transforms,
        tokenizer=tokenizer,
        batch_size=config['train']['batch_size'],
        drop_last=True
    )
    val_transforms = get_val_transforms(
        height=config['image']['height'],
        width=config['image']['width']
    )
    val_loader = get_data_loader(
        transforms=val_transforms,
        json_path=config['val']['json_path'],
        root_path=config['val']['root_path'],
        tokenizer=tokenizer,
        batch_size=config['val']['batch_size'],
        drop_last=False
    )
    return train_loader, val_loader


def train(config):
    tokenizer = Tokenizer(config['alphabet'])
    os.makedirs(config['save_dir'], exist_ok=True)
    train_loader, val_loader = get_loaders(tokenizer, config)

    model = CRNN(number_class_symbols=tokenizer.get_num_chars())
    model.to(DEVICE)

    criterion = torch.nn.CTCLoss(blank=0, reduction='mean', zero_infinity=True)
    optimizer = torch.optim.AdamW(model.parameters(), lr=0.001,
                                  weight_decay=0.01)
    scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(
        optimizer=optimizer, mode='max', factor=0.5, patience=15)
    best_acc = -np.inf
    acc_avg = val_loop(val_loader, model, tokenizer, DEVICE)
    for epoch in range(config['num_epochs']):
        loss_avg = train_loop(train_loader, model, criterion, optimizer, epoch)
        acc_avg = val_loop(val_loader, model, tokenizer, DEVICE)
        scheduler.step(acc_avg)
        if acc_avg > best_acc:
            best_acc = acc_avg
            model_save_path = os.path.join(
                config['save_dir'], f'model-{epoch}-{acc_avg:.4f}.ckpt')
            torch.save(model.state_dict(), model_save_path)
            print('Model weights saved')
```

Запускаем обучение!

```
train(config_json)
```

```
Validation, acc: 0.0000

Epoch 0, Loss: 5.43659, LR: 0.0010000
Validation, acc: 0.0364
Model weights saved
```

```
Epoch 1, Loss: 3.25091, LR: 0.0010000
Validation, acc: 0.0599
Model weights saved
```

Создание предсказаний для public-датасета

Сначала определим класс для создания предсказаний:

```
class InferenceTransform:
    def __init__(self, height, width):
        self.transforms = get_val_transforms(height, width)

    def __call__(self, images):
        transformed_images = []
        for image in images:
            image = self.transforms(image)
            transformed_images.append(image)
        transformed_tensor = torch.stack(transformed_images, 0)
        return transformed_tensor


class OcrPredictor:
    def __init__(self, model_path, config, device='cuda'):
        self.tokenizer = Tokenizer(config['alphabet'])
        self.device = torch.device(device)
        # load model
        self.model = CRNN(number_class_symbols=self.tokenizer.get_num_chars())
        self.model.load_state_dict(torch.load(model_path))
        self.model.to(self.device)

        self.transforms = InferenceTransform(
            height=config['image']['height'],
            width=config['image']['width'],
        )

    def __call__(self, images):
        if isinstance(images, (list, tuple)):
            one_image = False
        elif isinstance(images, np.ndarray):
            images = [images]
            one_image = True
        else:
            raise Exception(f"Input must contain np.ndarray, "
                            f"tuple or list, found {type(images)}.")

        images = self.transforms(images)
        pred = predict(images, self.model, self.tokenizer, self.device)

        if one_image:
            return pred[0]
        else:
            return pred
```

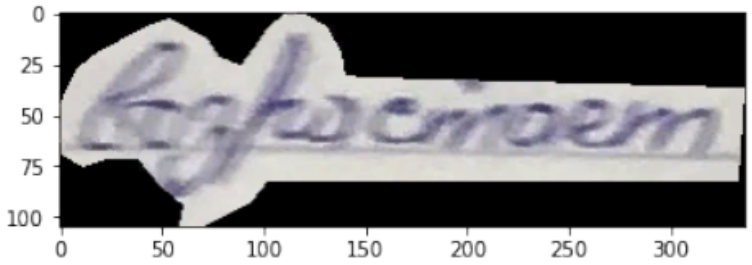
Инициализируем `OCR predictor`:

```
predictor = OcrPredictor(
    model_path='data/experiments/test/model-last.ckpt',
```

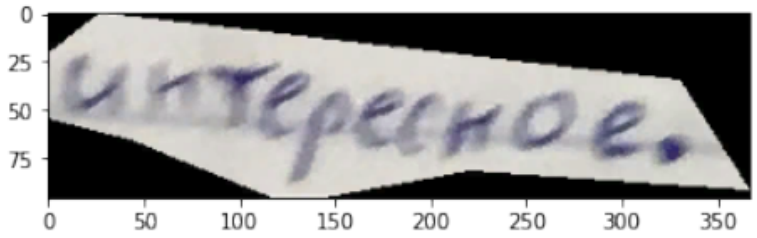
```
    config=config_json
)
```

Посмотрим несколько предсказаний и создадим финальный `json`:

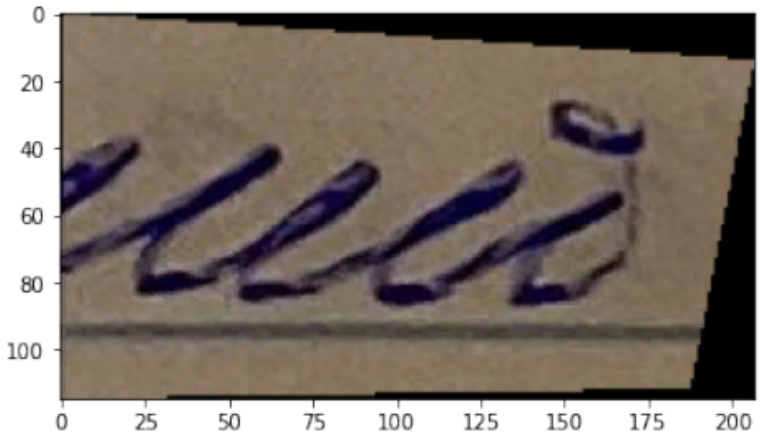
```
pred_json = {}

count = 0
print_images = True
for img_name in os.listdir('data/final_data/public/images/'):
    img = cv2.imread(f'data/final_data/public/images/{img_name}')

    pred = predictor(img)
    pred_json[img_name] = pred

    if print_images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        plt.imshow(img)
        plt.show()
        print('Prediction: ', predictor(img))
        count += 1

    if count > 3:
        print_images = False
```



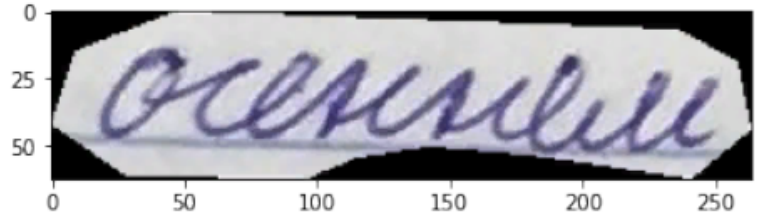

```
Prediction: взрастоет
```




```
Prediction: интересное.
```



Prediction: шй



Prediction: осеннем

Сохраням `submission json` с предсказаниями:

```
with open('data/prediction_HTR.json', 'w') as f:
    json.dump(pred_json, f)
```

Для распознавания рукописного текста в бейзлайне используется архитектура `CRNN` и `CTC-loss`. Для запуска бейзлайна скачайте данные для обучения https://clc.to/ai-21-22-train и положите их в папку data. Должна получиться такая структура:

- `OCR_baseline.ipynb`
- `train`
    - `images`
    - `labels.json`

Пример загружаемого на платформу решения на основе безлайна: https://clc.to/ai-21-22-sample-submission.

В архиве вы найдете следующие файлы:

- `metadata.json` — обязательный файл для каждого решения, в нем должны быть указаны пути к образу и скрипту выполнения модели;
- `run.py` — основной скрипт для инференса модели;
- `model-last.ckpt` — веса модели, которые подгружаются во время исполнения скрипта `run.py`.

Для корректной проверки решения сабмиты участников должны были иметь аналогичную структуру.

Заключительный этап

Индивидуальный предметный тур

Информатика. 8–11 класс

Задача III.1.1.1. (10 баллов)

Перед ИИ «Ваня» на работе стоит непростая задача, а именно, выявить самого счастливого клиента магазина. Будем считать, что счастье человека напрямую зависит от времени, проведенного в магазине. Самым счастливым человеком считается тот, кто провел в магазине наибольшее количество времени. Если несколько человек провели в магазине одинаковое количество времени, то самым счастливым является тот, кто покинул магазин позднее.

У вас есть информация о всех событиях за последние m минут. Каждую минуту происходило ровно одно из событий: человек с номером x зашел в магазин или человек с номером x вышел из магазина. Обучите ИИ «Ваня» определять самого счастливого клиента и время, которое он провел в магазине.

Формат входных данных

В первой строке задано два целых числа n и m ($0 < n,m \leqslant 10^5$) — количество людей, посетивших магазин и количество событий — входов и выходов.

В следующих m строках описаны события в хронологическом порядке. Интервал между событиями равен одной минуте. В каждой из строк через пробел записаны два целых числа T и X ($1 \leqslant T \leqslant 2$, $1 \leqslant X \leqslant n$) — тип события (1 — вход в магазин, 2 — выход из магазина) и идентификатор человека, совершившего данное действие, соотвественно. Гарантируется, что человек входит в магазин не более одного раза и каждый, кто посетил магазин, покинет его.

Формат выходных данных

В единственной строке выведите два числа, разделенные пробелом: идентификатор самого счастливого человека и время в минутах, которое он провел в магазине.

Примеры

Пример №1

Стандартный ввод
5 4 1 2 1 1 2 2 2 1
Стандартный вывод
1 2

Пример №2

Стандартный ввод
4 2 1 4 2 4
Стандартный вывод
4 1

Решение

Для каждого события первого типа будем запоминать его время входа в магазин. Пусть $t\_in[i]$ — время входа в магазин человека под номером i. Тогда при появлении события второго типа мы сможем легко вычислить время пребывания в магазине $(cur\_time - t\_in[x])$. Если продолжительность посещения, вышедшего только что человека, больше или равно текущего ответа, то обновим ответ.

Время работы алгоритма $O(m)$.

Задача III.1.1.2. (15 баллов)

Сегодня Вите с его другом ИИ «Ваня» предстоит анализировать граф связей людей. Есть неориентированный граф без петель и кратных ребер из n вершин (людей) и m ребер (связей). Пронумеруем всех людей от 1 до n. Пусть P — это перестановка людей, т. е. массив длины n, состоящий из элементов множества $1, 2, ..., n$ без повторений. Целью Вити является как можно быстрее найти количество таких пар индексов $l, r (1 \leqslant l \leqslant r \leqslant n)$, что люди из множества $P[l], P[l+1], ..., P[r]$ попарно незнакомы. Два человека i и j считаются знакомы, если в графе связей содержится ребро (i, j).

Формат входных данных

В первой строке задано два целых числа n, m, разделенных одним пробелом: $1 \leqslant n \leqslant 10^5$, $0 \leqslant m \leqslant 10^5$.

Во второй строке через пробел задана перестановка людей P. n различных чисел, разделенных одним пробелом $1 \leqslant P[i] \leqslant n$ для всех $1 \leqslant i \leqslant n$.

В следующих m строках заданы ребра графа связей. В каждой строке задано одно ребро в формате двух целых чисел, разделенных пробелом, обозначающие номера людей v_1, v_2, знакомых друг с другом. Гарантируется, что $1 \leqslant v_1 \leqslant n$, $1 \leqslant v_2 \leqslant n$, $v_1 \neq v_2$. Все ребра различны.

Формат выходных данных

В единственной строке выведите одно число — количество искомых пар чисел.

Примеры

Пример №1

Стандартный ввод
2 0 2 1
Стандартный вывод
3

Пример №2

Стандартный ввод
2 1 2 1 2 1
Стандартный вывод
2

Решение

Для каждой позиции r в заданной перестановке найдем максимальную левую позицию l ($l \leqslant r$), такую, что, начиная с нее и до r, все вершины не имеют ребер между собой. Обозначим такую позицию $left\_pos[i]$ для каждой позиции i. Тогда ответом на задачу будет $\sum_{i=1}^{n} (i - left\_pos[i] + 1)$.

Посчитать массив $left\_pos$ можно следующим способом. Для каждой позиции i найдем самую левую позицию j ($j \leqslant i$), такую, что вершина в позиции i не имеет связи ни с одной из вершин в интервале от j до i. Обозначим данное значение как $L[i]$. $L[i]$ — это еще не $left\_pos[i]$, так как надо учесть наличие ребер между вершинами в интервале между j и i. Легко заметить, что $left\_pos[i] = \max\limits_{1 \leqslant j \leqslant i} L[j]$.

Время работы алгоритма $O(n + m)$.

Задача III.1.1.3. (25 баллов)

Витя и его друг ИИ «Ваня» решают одну из важных проблем в теории графов. Представим, что у нас есть дорожная сеть, соединяющая города двунаправленными дорогами, которая представляет собой дерево с вершинами в городах и дорогами в качестве ребер. Требуется определить, какова вероятность появления циклической

дороги, состоящей из четного количества ребер, после соединения одной пары, еще не соединенных между собой дорогой, городов. Выбор пары городов для соединения происходит равновероятностно для всех удовлетворяющих условию пар.

Формат входных данных

В первой строке задано целое число n ($3 \leqslant n \leqslant 2 \cdot 10^5$) — число городов в стране.

В следующих $n-1$ строках заданы два целых числа a и b ($1 \leqslant a, b \leqslant n$) — номера городов, между которым существует дорога.

Гарантируется, что заданный граф является деревом.

Формат выходных данных

Выведите вероятность появления циклической дороги из четного количества ребер в виде несократимой дроби в формате: «P/Q» (без кавычек), где P — числитель, Q — знаменатель. Если вероятность равна нулю, выведите «0/1» (без кавычек).

Гарантируется, что вероятность можно представить в виде несократимой дроби.

Примеры

Пример №1

Стандартный ввод
5 1 2 2 3 3 4 3 5
Стандартный вывод
1/3

Пример №2

Стандартный ввод
3 2 1 2 3
Стандартный вывод
1/3

Решение

Данная задача имеет несколько различных подходов решения. Разберем один из них.

Так как по условию задачи дается дерево (т. е. это двудольный граф), следовательно, можно раскрасить вершины в два цвета (белые и черные) так, чтобы для любого ребра обе его вершины имели разные цвета.

Пусть w — это количество белых вершин, а b — количество черных. По свойству двудольного графа, цикл четной длины может возникнуть только при соединении вершин одного цвета, а нечетной — при соединении вершин различного цвета. Получается, что количество возможных способов получить цикл нечетной длины равно $odd = \frac{w \cdot (w-1)}{2} + \frac{b \cdot (b-1)}{2}$, а четной длины: $even = w \cdot b - (n-1)$. Тогда ответ на задачу равен: $even/(odd + even)$.

В условии просят вывести несократимую дробь. Для этого нужно числитель и знаменатель разделить на их наибольший общий делитель (НОД), который можно найти с помощью алгоритма Евклида.

Время работы алгоритма $O(n+m)$.

Задача III.1.1.4. (25 баллов)

Витя хочет расширить возможности своего друга ИИ «Ваня», чтобы возложить на него бремя выбора шариков для новогодней елки. У Вити есть множество A из n шариков. Шарики отличаются только двумя параметрами: цветом и весом. На новогодней елке умещается только k ($k \leqslant n$) шариков. Чтобы мама Вити была довольна тем, как Витя украсил елку, должны выполняться следующие условия:

- На елке должно находиться не больше k шариков.
- Пусть B это множество выбранных шариков. Если из этого множества можно выбрать два шарика с одинаковым цветом c и весами w_1 и w_2 ($w_1 \leqslant w_2$). То нужно, чтобы в множестве B существовал шарик цвета d и веса w_3 такой, что $c \neq d$ и $w_1 \leqslant w_3 \leqslant w_2$.

Так как елка не может выдержать произвольный вес, то Витя хочет удостовериться, что любой набор, который предложит ИИ «Ваня», не опрокинет елку. Пока Витя занят написанием навыка для своего друга, определите максимальный суммарный вес шариков, которые можно повесить на елку, чтобы мама Вити была довольна.

Формат входных данных

В первой строке заданы два целых числа n и k ($1 \leqslant k \leqslant n \leqslant 3 \cdot 10^5$) — всего шариков в распоряжении Вити и максимальное количество шариков, умещающихся на елке.

В i-ой из последующих n строк заданы два целых числа w_i и c_i — вес и цвет i-ого шарика ($0 \leqslant w_i, c_i \leqslant 10^9$).

Формат выходных данных

В единственной строке выведите максимальный суммарный вес для набора шариков, который удовлетворит маму Вити.

Примеры

Пример №1

Стандартный ввод
2 1 100 100 0 100
Стандартный вывод
100

Пример №2

Стандартный ввод
4 3 1 1 10 10 10 1 1 1
Стандартный вывод
21

Решение

Для начала разобьем все шары на группы по их весу. Если группа состоит из шаров одного цвета, то в ответе мы сможем использовать не более одного шара из этой группы, так как иначе не будет выполнено второе условие. Теперь группы шаров одного веса состоят либо из одного шарика (обозначим группы этого типа за A), либо в группе есть два шара разных цветов (группы типа B).

Давайте отсортируем по убыванию весов все шары. Тогда, если у нас есть более одной группы шаров типа A одного цвета подряд, то ответ может попасть только одна из таких групп (или один конкретный шар, т. к. группа типа A состоит из одного шара), иначе не будет выполнено второе условие. Чтобы максимизировать ответ выберем из этих групп первую, т. к. у этого шара максимальный вес.

Несложно доказать, что если отсортировать оставшиеся шары в порядке убывания весов и просуммировать первые k (или меньше, если шаров осталось меньше k) весов, то это будет максимальный возможный ответ, а также можно подобрать набор шаров, удовлетворяющий данной сумме весов.

Время работы алгоритма $O(nlogn)$.

Задача III.1.1.5. (30 баллов)

РИИ «Ваня» предложил Вите сыграть в очень известную игру «Крестики». Эта игра состоит из n клетчатых полей 3×3. Игра ведется одновременно на всех полях. Игроки поочередно делают ходы, во время своего хода игрок выбирает активное поле и ставит на нем крестик в любую свободную клетку. Поле считается активным, если на нем нет строки, столбца или диагонали, полностью заполненных крестиками. Проигравшим признается тот, кто не сможет сделать ход. Изначально некоторые

поля могут уже содержать крестики. Это делается для того, чтобы усложнить игру и не дать игрокам действовать по заранее известному сценарию. Однако гарантируется, что вначале игры все поля являются активными. ИИ «Ваня» всегда дает фору своему другу, поэтому первым ход в игре делает Витя. Подскажите Вите, сможет ли он выиграть эту игру, если начнет игру первым, или ИИ «Ваня» все просчитал и шанса на победу нет. Конечно же, и Витя, и ИИ «Ваня» хотят выиграть, поэтому делают только лучшие для себя ходы.

Формат входных данных

В первой строке задано число игровых полей n ($0 \leqslant n \leqslant 10^5$). Дальше идет описание полей, каждое поле описывается тремя строками. Каждая строка состоит из трех символов «.» или «X», где точка обозначает пустую клетку, а «X» занятую крестиком.

Формально на строке под номером $3 \cdot i - 1$ идет описание первой строки i-ого поля, на строке $3 \cdot i$ — второй строки i-ого поля, а на $3 \cdot i + 1$ — третьей строки i-ого поля.

Формат выходных данных

В единственной строке выведите игрока, который победит в игре. «First» (без кавычек) — в случаем победы Вити или «Second» (без кавычек) — в случае победы ИИ «Ваня».

Примеры

Пример №1

Стандартный ввод
1 ... .X. ...
Стандартный вывод
Second

Пример №2

Стандартный ввод
2 ... ...
Стандартный вывод
Second

Решение

Заметим, что данная игра представляет собой равноправную игру двух игроков, т. е. игру, где игроки располагают всей информацией о игре (правилах, ходах соперника, текущем состоянии). Проигрыш или выигрыш зависят только от состояния игры. От того, какой именно игрок ходит первым, не зависит ничего: т. е. игроки полностью равноправны. А также в каждой игре есть только выигрышные или проигрышные состояния, ничейных состояний нет.

Следовательно, по теореме Шпрага–Гранди данную игру можно свести к игре «Ним». Каждое игровое поле можно рассматривать как независимую игру, потому что действия на одном поле никак не влияют на состояния других полей, а их совокупность — как сумму независимых игр. Если мы каждой независимой игре поставим в соответствие число (функцию Гранди), то если XOR (побитовое ИЛИ) этих чисел будет отличен от нуля, победит первый игрок, иначе — второй.

Чтобы для конкретного состояния поля определить его значение функции Гранди, нужно из текущего состояния перебрать все возможные переходы и посчитать mex из их значений функции Гранди (где функция mex — от множества чисел возвращает наименьшее неотрицательное число, не встречающееся в этом множестве). Для проигрышного состояния значение функции равняется нулю.

Для того, чтобы избежать TL, нужно сохранять в памяти значения функции Гранди для уже посчитанных состояний.

Время работы алгоритма $O(n+2^k \cdot k)$, где k — размер поля (в данной задаче 9).

Математика. 8–9 классы

Задача III.1.2.1. (15 баллов)

Среди всех дробей вида $\frac{a}{b}$, где a и b — натуральные числа, а дробь удовлетворяет условию

$$\frac{2021}{2022} < \frac{a}{b} < \frac{2022}{2023}$$

найдите дробь с минимальным знаменателем b.

Критерии оценивания

- Только верный ответ — 5 баллов.
- Неправильная работа с дробями — не более 5 баллов.
- Запись числа вида $n/(n+1)$ в виде $1 - 1/(n+1)$ (даже без дальнейших продвижений) — 5 баллов.

Решение

Обозначим a как $b - x$ для целого x. Тогда имеем:

$$1 - \frac{1}{2022} < \frac{b-x}{b} < 1 - \frac{1}{2023}$$

$$\frac{1}{2022} > \frac{x}{b} > \frac{1}{2023}$$

$$2022 < \frac{b}{x} < 2023$$

Откуда имеем, что минимально возможное x равно 2, а b тогда не менее $2022 \cdot x + 1$, что не менее 4045. Для этого знаменателя дробь будет равна $\frac{4043}{4045}$ и будет удовлетворять условию задачи.

Ответ: $\frac{4043}{4045}$.

Задача III.1.2.2. (15 баллов)

Пусть AB и BC — равные стороны равнобедренного треугольника ABC. На сторонах AC, BC и AB выбраны такие точки P, Q и R соответственно, что PQ параллельна AB, RP параллельна BC и $RB = AP$. Найдите величины углов треугольника ABC, если $\angle AQB = 105^\circ$.

Критерии оценивания

- Только верный ответ — 5 баллов.
- Доказательство проведено для «другой картинки», но формально верное (все переходы верны и обоснованы) — баллы не снимаются.

- Проведены верные рассуждения, которые, тем не менее, неясно как довести до полного решения (дополнительные построения, вспомогательные утверждения и т. п.) — баллы за такие рассуждения не начисляются.

Решение

Из условия следует, что четырехугольник $RBQP$ — параллелограмм, значит $AP = RB = PQ$. Обозначим величину угла A треугольника ABC за α, тогда $\angle QCP = \alpha$ из равнобедренности. $\angle QPC = \angle BAC = \alpha$ из параллельности прямых AB и PQ. Значит, угол QPA равен $180^\circ - \alpha$, а в равнобедренном треугольнике AQP с этим углом два других угла равны друг другу, каждый из них равен $\alpha/2$. Таким образом, развернутый угол при точке Q равен:

$$180^\circ = \angle BQA + \angle AQP + \angle PQC = 105^\circ + \alpha/2 + (180^\circ - 2\alpha).$$

Отсюда получаем, что $\alpha = 50^\circ$, а значит:

$$\angle BAC = \angle ACB = 50^\circ,\ \angle ABC = 80^\circ.$$

Ответ: $50^\circ, 50^\circ, 80^\circ$.

Задача III.1.2.3. (20 баллов)

На доске нарисован клетчатый квадрат 5×5. Можно ли так расставить числа от 1 до 25 в клетки этого квадрата (по одному числу в клетку, все числа на доске должны быть различны), чтобы сумма чисел в любых двух соседних по стороне клетках равнялась бы точному квадрату?

Уточнение: точным квадратом называется число, которое является квадратом целого числа.

В этой задаче недостаточно дать только ответ – необходимо доказательство этого ответа.

Критерии оценивания

- Только верный ответ — 2 балла.
- Рассмотрение возможного положения числа 25 — 5 баллов, возможного положения числа 24 — 2 балла.
- Неполный перебор вариантов (переборного решения) — не более 10 баллов.

Решение

Предположим, такая расстановка существует. Тогда число 25 может быть дополнено не более чем до 49 (что является точным квадратом), но, с другой стороны, 25 в сумме с любым натуральным числом даст не менее 26. Таким образом, существуют всего две суммы, равные точному квадрату, в которых могло поучаствовать число 25: это $25 + 11 = 36$ и $25 + 24 = 49$. Заметим, что если число 25 стоит не в углу доски, то у него есть минимум три соседа, в которых будут стоять три различных

числа. Значит, минимум одна из сумм не является точным квадратом. Значит, в любой подходящей расстановке число 25 стоит в углу, а в соседних с ним клетках стоят 24 и 11.

Аналогичный анализ показывает, что рядом с 24 могут стоять числа 25 (уже найденное ранее), 12 (дополняющее до 36) и 1 (дополняющее до 25). Рассмотрим угловой квадрат 2×2 на этой доске, в который входят угловая клетка с числом 25, а также соседние клетки с 24 и 11. Рассмотрим последнюю, четвертую клетку этого квадрата: она соседняя с 24, поэтому в ней должно стоять либо число 1, либо 12 (но не 25, так как оно уже стоит в другой клетке). Но ни то, ни другое не дополняет число 11 до полного квадрата, хотя эти клетки также соседние. Противоречие с предположением, что такая расстановка существует.

Ответ: нет, так расставить нельзя.

Задача III.1.2.4. (25 баллов)

Двое играют в игру: первый игрок отмечает на прямой несколько точек, выстроенных в цепочку так, что каждые соседние две находятся на расстоянии 1 см. Второй игрок красит часть из этих точек в красный цвет, а остальные — в синий. После этого первый игрок проверяет, есть ли среди данных точек три точки A, B и C одного цвета, что расстояния AB и BC равны. Если такая тройка находится, то выиграл первый, если не находится — то второй. Найдите минимальное количество точек, которое в начале игры должен отметить первый игрок, чтобы гарантированно выиграть.

Критерии оценивания

- Только верный ответ — 5 баллов.
- Пример покраски вторым игроком 8-ми точек (для своей победы) — +7 баллов; нет покраски на 8-ми точках, но есть покраска «за второго» на 7 точек — +2 балла (если ответ при этом указан 8, то в целом за задачу не более 2 баллов); приведена покраска лишь на 6-ти или менее точках — продвижение дополнительно не оценивается.
- Доказано только, что на 9-ти точках всегда найдется искомая одноцветная тройка — +8 баллов; доказано что на 10-ти точках есть требуемая тройка — +3 балла; доказано, что на 11-ти или более точках есть требуемая тройка — продвижение дополнительно не оценивается.

Решение

Для каждой покраски точек рассмотрим соответствующую последовательность первых букв цветов, в которые они покрашены. Например, если отмечено всего три точки, самая левая из них красная, а две другие — синие, то им соответствует последовательность К С С. Легко видеть, что если между тремя раскрашенными точками были два равных расстояния (то есть можно было эти точки назвать А, В и С так, чтобы расстояние АВ равнялось расстоянию ВС), то и в последовательности букв будет выполнено аналогичное свойство: номера a, b и c букв в последовательности такие, что $b - a = c - b$. Поэтому будем решать задачу на последовательностях букв

вместо точек на прямой.

Покажем, что для 8-ми и менее точек на прямой второй игрок может сделать выгодную себе раскраску (и выиграть). Действительно, на 8-ми буквах можно рассмотреть последовательность К С С К К С С К. Легко проверить, что крайняя буква К не входит в тройку букв К, которые образуют два равных расстояния между собой. Обе крайние К находятся в сиимметричном положении, поэтому вывод верен для обеих. А без крайних К используются всего две другие буквы К, чего недостаточно для тройки. Аналогично можно рассмотреть крайнюю букву С — она также не входит ни в одну подобную тройку. А кроме крайних С остается всего две буквы С, чего недостаточно. Если букв в игре будет поставлено k меньшее 8-ми, то можно взять раскраску (распределение букв С и К) для как k самых левых букв в предъявленной последовательности на 8-ми буквах. Там тоже не будет искомой тройки одного цвета.

Докажем, что при 9-ти и более буквах уже гарантированно (при любом распределении букв, согласно выбору второго игрока) можно выбрать три буквы, между которыми есть два равных расстояния. Докажем от противного: пусть есть расстановка 9-ти букв, среди которых нет требуемой троицы одинаковых букв. Тогда пусть, не умаляя общности, центральная буква этой расстановки — это К. Если с этой буквой рядом также стоит буква К (не умаляя общности — левая соседняя), то есть расстановка имеет вид:

(* * *) (К К *) (* * *),

то третья буква центральной тройки букв — точно не К, то есть С. Кроме того, правая буква левой тройки также С:

(* * С) (К К С) (* * *).

Тогда самая правая буква должна быть не С, то есть К:

(* * С) (К К С) (* * К).

Но тогда на самой левой позиции стоит не К (то есть С), иначе первая, средняя и последняя буквы образуют такую тройку:

(С * С) (К К С) (* * К).

Откуда видно, что средняя буква левой тройки — это не С (то есть К):

(С К С) (К К С) (* * К).

Но тогда если средняя буква правой тройки — это К, то средние буквы всех троек образуют требуемую конструкцию из трех К, чего не должно быть по предположению. Значит, средняя буква правой тройки — это С:

(С К С) (К К С) (* С К).

Но на оставшуюся позицию нельзя поставить ни С (из-за соседних букв), ни К (из-за идущих через одну букву).

Значит, если требуемая конфигурация на 9-ти буквах существует, то в ее центральной тройке центральная буква не равна двум другим. Рассмотрим и такой вариант. Не умаляя общности, опять в центре К, а рядом, значит, две буквы С:

(* * *) (С К С) (* * *).

Тогда средняя буква левой тройки — не С (то есть К):

(* К *) (С К С) (* * *).

Так как средние буквы троек расположены «на равном расстоянии», то средняя буква правой тройки не К (то есть С):

(* К *) (С К С) (* С *).

Тогда левая буква правой тройки — не С (то есть К):

(* К *) (С К С) (К С *).

Из-за левой буквы К правой тройки и центральной К в центральной тройке, правая буква левой тройки не К, то есть С:

(* К С) (С К С) (К С *).

Так как правые буквы всех троек расположены «на равном расстоянии», то самая правая буква не С, то есть К. Но тогда эта буква, а также левая буква той же (правой) тройки и центральная буква центральной тройки — это три буквы К, между которыми два одинаковых расстояния. В любом из возможных случаев получили противоречия с предположением, что существует комбинация 9-ти букв, в которой никакие три одинаковые буквы не стоят на позициях, между которыми два одинаковых расстояния.

Если букв будет больше, то достаточно рассмотреть самые левые 9 из них.

Ответ: 9.

Задача III.1.2.5. (25 баллов)

На столе лежит 21 карта, на каждой карте написан свой уникальный номер от 1 до 21. Петя выбирает из них 4 карты и показывает Вере. Затем Вера забирает у Ивана одну из этих карт (любую, какую она захочет). Если сумма номеров трех карт, которые остались у Пети, кратна 3, то Вера выигрывает. Иначе побеждает Вера.

Определите, сколькими способами Петя может выбрать 4 карты, чтобы быть уверенным в победе, независимо от того, насколько хорошо играет Вера.

Уточнение: два набора карт, отличающихся лишь порядком карт, считаются одинаковыми.

Критерии оценивания

- Только верный ответ — 5 баллов.
- Приведено конструктивное описание требуемого множества («два числа из множества A и два числа из множества B», где в тех или иных терминах явно описано разделение всего множества на три подмножества: $\{1, 4, 7, \ldots, 19\}$, $\{2, 5, \ldots, 20\}$, $\{3, 6, \ldots, 21\}$ — 10 баллов.
- Решение переборное, в котором посчитано количество вариантов в некоторых группах (не менее одной нетривильной группы), но остались не посчитанными количество вариантов в других группах (не менее одной нетривильной группы осталось не посчитанной, и неясно, можно ли ее посчитать «в одно действие») — не более +5 баллов к баллам за ответ.

Решение

Назовем четверку карт, при которой Петя выиграет независимо от хода Веры, удачной. Рассмотрим, какие остатки при делении на 3 могут давать числа в такой четверке.

Во-первых, заметим, что если в четверке карт есть три с одинаковыми остатками при делении на 3, то, оставив эти три карты Пете, Вера выиграет. То есть в удачной четверке не должно быть трех чисел, дающих одинаковые остатки при делении на 3.

Во-вторых, если в четверке есть три числа, дающие все три разных остатка при делении на 3 (то есть 0, 1 и 2), то сумма этих трех чисел делится на 3. Если Вера оставит Пете эти три карты, то также она выиграет. Значит, в удачной четверке не могут присутствовать номера всех трех различных остатков при делении на 3.

Тогда из этих двух наблюдений следует, что в удачной четверке могут быть только два различных остатка при делении на 3 (один не может быть, так как тогда есть три карты с одинаковым остатком; три не могут быть, так как тогда все остатки присутствуют). Больше того, так как ни один остаток из этих двух не может быть представлен 3-мя или 4-мя числами, то второй остаток из двух не может быть представлен одним числом (нулем чисел также не может быть представлен, так как, по выбору, этот остаток в четверке есть).

Проверим, что любая четверка, в которой два числа дают один остаток при делении на 3, а другие два числа — другой, является удачной. Действительно, когда Вера заберет одну карту, то в оставшейся тройке будут два числа одного остатка (остатка r) и одно — другого (q). Заметим, что $r + r + r$ делится на 3, а q отличается от r на 1 или 2. Значит, $r + r + q$ отличается от числа кратного 3 на 1 или на 2, то есть само не является кратным 3. Что и требовалось.

Тогда для подсчета всех удачных четверок достаточно посчитать количество четверок, составленных из 4-х различных чисел (каждое от 1 до 21 включительно), два из которых имеют один остаток при делении на 3, а другие два — другой. Заметим, что чисел, представляющих каждый из остатков, всего 7. Тогда для каждой пары разных остатков (например, 0 и 1), количество таких наборов будет равно $C_7^2 \cdot C_2^7 = 21^2 = 441$ (тут C_7^2 — это количество сочетаний из 7 по 2, то есть число равное $7 \cdot 6/2$). Разных пар остатков при делении на 3 всего 3: $(0, 1), (0, 2), (1, 2)$. Значит и общее количество подходящих наборов $441 \cdot 3 = 1323$.

Ответ: 1323.

Математика. 10–11 классы

Задача III.1.3.1. (15 баллов)

Назовем натуральное число уравновешенным, если в его записи в двоичной системе счисления содержится четное число единиц. Если же количество единиц в двоичной записи числа нечетно, то будем называть его неуравновешенным. Найдите количество уравновешенных чисел в диапазоне от 1 до 2022.

Критерии оценивания

- Только ответ — 5 баллов.
- Упоминается идея разбиения на пары, но не указано, как это сделать — +5 баллов.

Решение

Рассмотрим пару идущих подряд чисел: n и $n+1$, из которых n нечетно. Тогда двоичная запись числа $n+1$ отличается от двоичной записи числа n только единицей в младшем разряде. Значит, из этих двух чисел одно уравновешенное, а другое — нет. Значит, уравновешенных чисел от 2 до 2021 поровну, т. к. эти числа разбиваются на пары вида $2k+1, 2k+2$.

Без пар такого вида остаются только числа 1 и 2022. Число 1, очевидно, не является уравновешенным. Числа 2022 переведем в двоичный код: 11111100110. В этом числе 8 единиц, то есть четное количество. Значит, оно является уравновешенным. Таким образом, все числа от 1 до 2022 можно разбить на пары, в каждой из которых ровно одно уравновешенное число. Итого 1011 уравновешенных чисел.

Ответ: 1011.

Задача III.1.3.2. (20 баллов)

Найдите все такие натуральные числа n, что $\left[\dfrac{n^2}{5}\right]$ является простым числом.

Уточнение: тут квадратные скобки обозначают операцию взятия целой части числа. То есть для произвольного числа a число $[a]$ равно наибольшему целому числу, которое не превосходит a. Например, $[1,3]=1; [3]=3$.

Критерии оценивания

- Только ответ — 7 баллов.
- Неполный перебор случаев (даже с полным ответом) вычитается не менее 5 баллов.
- Идея остатков по модулю 5 — +3 балла.
- Идея квадратичных вычетов — +2 балла.

Решение

Обозначим указанную в условии целую часть как p. Тогда $5p$ является целым числом, с одной стороны, не превосходящим n^2, с другой — строго большим числа n^2-5.

Отметим, что n^2 может давать остатки $0, 1$ или 4 при делении на 5 (и не может давать другие остатки, то есть 2 и 3). Тогда числа n^2-2 и n^2-3 не могут равняться $5p$.

Остаются три варианта: если $5p=n^2$, значит $n=5$ (из-за простоты p). Если $5p=n^2-1$, то $(n-1)(n+1)=5p$, откуда $n-1$ равно 1, либо 5, либо p. Первый вариант невозможен (т. к. $n=2$), второй дает $n=6$, третий означает, что $n+1=5$, то есть $n=4$.

Наконец, если $5p=n^2-4$, то $(n-2)(n+2)=5p$, откуда $n-2=1$ (не подходит — слишком мало), либо $n-2=5$ (не подходит, т. к. $n+2$ тогда равно 9, что не является простым), либо $n-2=p$, а тогда $n+2=5$, $n=3$, но при этом $p=1$, что неверно, т. к. 1 не простое).

Ответ: 4, 5, 6.

Задача III.1.3.3. (20 баллов)

Известно, что $\angle B$ — угол треугольника ABC — равен 40°. Известно, что на биссектрисе угла B есть точка P, для которой верно $BP = BC$ и $\angle BAP = 20°$. Определите размеры углов $\angle A$ и $\angle C$.

Критерии оценивания

- Только ответ 5 баллов.

Решение

Рассмотрим точку N, несовпадающую с P, такую, что треугольники ABP и ABN равны, но не совпадают (и лежат по разные стороны от прямой AB). Тогда угол CBN равен $20° \cdot 3 = 60°$, а также и условий: $BN = BP = BC$. Тогда треугольник NBC — равнобедренный с углом 60°. Значит, он равносторонний (например, счетом остальных двух углов равнобедренного треугольника).

Из равнобедренности имеем, что $\angle BCP = \angle BPC = 80°$ (так как в треугольнике CBP угол CBP равен 20°, а $CB = BP$). Так как в равностороннем треугольнике CBN углы по 60°, то $\angle PCN = 20°$.

Треугольник PBN является равнобедренным по построению точки N, угол PBN равен удвоенному углу 20°, то есть $\angle PBN = 40°$. Значит, два других его угла равны по 70°, а угол PNC равен тогда 10° (т. к. $\angle BNC = 60°$).

Треугольник CNA равнобедренный, т. к. $CN = BN = NA$, угол CNA равен $\angle BNA - \angle CNA = \angle BNA - 60°$. Угол BNA при этом равен 140°, т. к. треугольник BNA — равнобедренный, с углом при основании 20°. Отсюда, угол CNA равнобедренного треугольника CNA равен $140° - 60° = 80°$. Значит, другие два его угла (при основании CA) равны по 50°. Откуда $\angle PCA = \angle NCA - \angle NCP = 50° - 20° = 30°$. Аналогично, $\angle PAC = \angle NAC - \angle NAP = 50° - 40° = 10°$.

Тогда угол A исходного треугольника ABC равен $\angle BAP + \angle PAC = 20° + 10° = = 30°$, а угол C равен $\angle ACP + \angle PCB = 30° + 80° = 110°$.

Ответ: $\angle A = 30°, \angle C = 110°$.

Задача III.1.3.4. (20 баллов)

Представьте число 2022 в виде суммы 30 натуральных слагаемых так, чтобы произведение этих слагаемых было максимально возможным.

Критерии оценивания

- Только верный ответ — 5 баллов.
- Доказан шаг «оптимизации» текущей конструкции (наподобие «если одно слагаемое из списка Forbes увеличить, а другое — уменьшить» — +5 баллов).

Решение

Докажем вспомогательное утверждение: Пусть a и b — такие натуральные числа, что $a < b - 1$. Тогда $ab < (a + 1)(b - 1)$. Доказательство: раскроем скобки в правой части и сократим обе части на слагаемое ab. Оставшееся неравенство эквивалентно условию $a < b - 1$ из условия утверждения.

Отметим, что $2022/30 = 67,4$. Отсюда можно вычислить представление 2022 в виде суммы целых слагаемых, максимально близких к среднему арифметическому, т. е. к 67,4. Пусть $2022 = 67a + 68b$, где a, b — количества соответствующих слагаемых в таком представлении. Тогда $a + b = 30$, а значит:

$$2022 = 67(a + b) + b = 67 \cdot 30 + b = 2010 + b.$$

То есть $b = 12, a = 18$.

Так как все слагаемые натуральные, то существует лишь конечное количество сумм, удовлетворяющим условию задачи (без требования максимальности произведения). Значит, среди них есть один или несколько наборов, на котором (которых) достигается максимальное значение. Рассмотрим любой из таких наборов.

Если в этом наборе есть число a меньшее 67, то также в нем содержится и число b, не меньшее 68 (иначе все 30 чисел строго меньше 67, а общая сумма не превосходит 2010). Но тогда произведение чисел $a + 1$ и $b - 1$ будет больше, чем произведение ab (согласно вспомогательному утверждению). Но, значит, рассматриваемый набор давал не максимальное произведение — противоречие выбору набора. Значит, в этом наборе нет чисел меньших 67. Аналогично в этом наборе нет чисел больших 68. Значит, все числа такого набора равны либо 67, либо 68, а единственный такой набор мы уже нашли.

Ответ: $2022 = 67 \times 18 + 68 \times 12$ (то есть 18 слагаемых по 67, 12 слагаемых по 68).

Задача III.1.3.5. (25 баллов)

Число A получено как произведение факториалов первых 2022 натуральных чисел:

$$A = (1!)(2!)(3!) \cdot \ldots \cdot (2021!)(2022!)$$

Определите, можно ли вычеркнуть один из этих факториалов так, чтобы произведение оставшихся 2021-го факториала было точным квадратом.

Уточнение: факториал натурального числа n — это произведение всех целых чисел от 1 до n. Например, $5! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5$.

Целое число называется точным квадратом, если оно является квадратом целого числа. Например, 16 — точный квадрат, потому что $16 = 4^2$.

Критерии оценивания

- Только верный ответ — не более 5 баллов.
- Приведена формула Лежандра для степени вхождения — +5 баллов.

Решение

Перепишем A в виде:

$$(1!)(1!) \cdot 2 \cdot (3!)(3!) \cdot 4 \cdot \ldots \cdot (2021!)(2021!) \cdot 2202 = (1! \cdot 3! \cdot \ldots \cdot 2021!)^2 \times B,$$

где $B = 2{\cdot}4{\cdot}6{\cdot}\ldots{\cdot}2022$, то есть A является произведением точного квадрата (квадрата числа $1! \cdot 3! \cdot \ldots \cdot 2021!$) и целого числа B. Поэтому A является точным квадратом если и только если B является таковым. Проанализируем его.

Если из каждого множителя числа B вынести число 2, то получим:

$$B = 2^2 011 \times 1011!$$

Если это число, точный квадрат, то каждое простое число в его каноническое разложение должно входить в четной степени. Исследуем, в какой степени в это число входит число 5. По известной формуле Лежандра о степени вхождения простого числа в факториал, имеем: степень вхождения 5 в разложение 1011! равна:

$$\left[\frac{1011}{5}\right] + \left[\frac{1011}{5^2}\right] + \left[\frac{1011}{5^3}\right] + \ldots,$$

где суммирование ведется по степени знаменателя, пока очередное слагаемое не станет равным нулю. Тогда легко вычислить: $\left[\frac{1011}{5}\right] = 202$, $\left[\frac{1011}{5^2}\right] = 40$, $\left[\frac{1011}{5^3}\right] = 8$, $\left[\frac{1011}{5^4}\right] = 1$. Складывая эти слагаемые, имеем, что в сумме все слагаемые четные, кроме последней единицы. То есть в разложение $2^1 011 \cdot 1011!$ пятерка входит в нечетной степени, а значит, B не является точным квадратом, так же как и A.

Ответ: Нет, так вычеркнуть какой-то из факториалов-множителей не получится.

Командный практический тур

Задача III.2.1. Распознавание рукописного текста (100 баллов)

Введение

Финальная задача является продолжением задач отборочного этапа, она предполагает расширение возможностей разработанных участниками решений.

Требования к команде

Команда состоит из двух участников, деление на роли не предполагается, участники являются равноправными партнерами в команде.

Оборудование и программное обеспечение

Минимальные требования к рабочему месту участника:

- Компьютер: процессор не ниже Intel Core i5 (или аналогичный), 4Gb ОЗУ.
- Компьютер оснащен камерой, микрофоном, наушниками или динамиком в рабочем состоянии.
- Смартфон с установленным Telegram и доступом в интернет.
- Скорость доступа в интернет: не менее 100 Мбит/с.
- На компьютере установлен браузер последней версии.

Профильная задача заключительного этапа по профилю размещается на платформе `ods.ai` (далее — Платформа). Доступ к Платформе осуществляется посредством браузера, установка дополнительного программного обеспечения для пользования Платформой не требуется.

Для выравнивания вычислительных возможностей участникам будут предоставлены доступы для обучения моделей с использованием мощностей суперкомпьютера «Кристофари» на 23,5 часа с учетом времени, необходимого для проверки работы доступов. Данное время соответствует количеству часов, необходимому для решения задачи. Доступ к «Кристофари» осуществляется посредством браузера, установка дополнительного программного обеспечения (ПО) для пользования сервисом не требуется.

При желании участники могут использовать для решения задачи любое ПО, не противоречащее данным правилам.

Условие

Участникам необходимо разработать алгоритм, который способен распознать рукописный текст в школьных тетрадях. В качестве входных данных предоставлены фотографии целых листов. Предсказание модели — список распознанных строк с координатами полигонов и получившимся текстом.

Задача усложняется следующими факторами:

- больше стилей (почерков), а, значит, разработанный алгоритм должен «приспосабливаться» к абсолютно любому почерку;
- в выборку добавлен английский язык, т. е. разрабатываемая модель должна быть билингвальной.

Разработчики задачи видят итоговый алгоритм итоговый алгоритм как последовательность двух моделей: сегментации и распознавания. Сначала сегментационная модель предсказывает полигоны маски каждого слова на фото. Затем эти слова вырезаются из изображения по контуру маски и подаются в модель распознавания. В итоге получается список распознанных слов с их координатами.

Формат решения

В проверяющую систему необходимо отправить код алгоритма, запакованный в ZIP-архив. Решения запускаются в изолированном окружении при помощи Docker. Время и ресурсы во время тестирования ограничены.

В корне архива обязательно должен быть файл `metadata.json` со структурой:

```
{
    "image": "<docker image>",
    "entry_point": "<entry point or sh script>"
}
```

Например:

```
{
    "image": "<docker image>",
    "entry_point": "<entry point or sh script>"
}
```

Здесь `image` — поле с названием docker-образа, в котором будет запускаться решение, `entry_point` — команда, при помощи которой запускается скрипт инференса.

Решение запускается в Docker контейнере. Участники могут воспользоваться готовым образом «`skalinin1/baseline-ocr-segm:latest`» (создан на основе Dockerfile: https://clc.to/ai-21-22-final-dockerfile).

При желании можно использовать свой образ, выложив его на https://hub.docker.com.

Доступные ресурсы:

- 8 ядер CPU;
- 48Gb RAM;
- видеокарта NVidia Tesla V100.

Ограничения:

- 5Gb на архив с решением;
- 25 минут на работу решения.

Порядок выполнения командной задачи

1. В день старта решения задачи участникам предоставляется ссылка на Платформу.
2. В ходе соревнования участники обучают модель, формируют докер с обученной моделью и загружают его на Платформу. В сутки команда может загрузить решение 10 раз.
3. После загрузки решения определяется качество его работы путем проверки на тестовом наборе данных. Тестовый набор данных разделен на 2 примерно равные части: публичную и приватную.
4. Результаты работы по публичной части данных формируют рейтинг, доступный участникам на Платформе в ходе соревнования.
5. До завершения приема решений участник должен выбрать среди своих решений те, которые он считает наилучшими. Данные решения учитываются при формировании итогового рейтинга. Возможное количество выбранных решений — 2.
6. Если участник не выбирает решения, которые он считает наилучшими, автоматически выбираются решения, с наилучшим результатом на основании результатов по публичной части данных.
7. После завершения приема решений и их проверки выбранные решения проверяются на приватной части данных. На основе этих данных формируется итоговый рейтинг.
8. После формирования итогового рейтинга происходит расчет баллов за командную задачу.
9. После загрузки решения участником на Платформе отражается статус его обработки:
   9.1. Статус `Running` означает, что решение обрабатывается. Срок обработки не может превышать 1 час. Если обработка затягивается на больший срок, необходимо обратиться к организаторам или на почту `support@datasouls.com`, приложив к сообщению скриншот со статусом.
   9.2. Статус `Queued` означает, что решение находится в очереди на обработку. В среднем это может занимать 15–30 минут в зависимости от количества одновременно отправленных решений. В вечернее время и в последние дни перед закрытием лидерборда время обработки всегда увеличивается ввиду возросшего количества решений, учитывайте это при планировании работы.
   9.3. Различные статусы ошибки обработки решения.

Решение

Базовое решение от разработчиков: https://clc.to/ai-21-22-final-baseline.

В файле `baseline.ipynb` представлен подход разработчиков задачи к объединению моделей сегментации и распознавания текста. Данный алгоритм не содержит обучение, только объединение 2-х уже обученных моделей. Для запуска бейзлайна скачайте данные для обучения, должна получиться следующая структура:

- `baseline.ipynb` (основной бейзлайн для текущего хакатона — объединение двух моделей);

- `segm-model_final.pth` (веса модели сегментации);
- `ocr-model-last.ckpt` (веса модели детекции);
- `baseline_segmentation.ipynb` (бейзлайн из 1-ой части олимпиады);
- `baseline_recognition.ipynb` (бейзлайн из 2-ой части олимпиады);
- `train_segmentation`:
    - `images`;
    - `annotations.json`;
    - `annotations_extended.json`;
    - `binary.npz`.
- `train_recognition`:
    - `images`;
    - `labels.csv`.

Файл `baseline_segmentation.ipynb` https://clc.to/ai-21-22-final-segmentation содержит базовое решение для модели сегментации текста. Для запуска модели скачайте данные для обучения и положите их в папку `train_segmentation`.

Файл `baseline_recognition.ipynb` https://clc.to/ai-21-22-final-recognition содержит базовое решение для модели распознавания текста. Для запуска модели скачайте данные для обучения и положите их в папку `train_recognition`.

Пример загружаемого на платформу решения на основе безлайна: https://clc.to/ai-21-22-final-submit.

В архиве вы найдете следующие файлы:

- `metadata.json` — обязательный файл для каждого решения, в нем должны быть указаны пути к образу и скрипту выполнения модели;
- `run.py` — основной скрипт для инференса модели;
- `segm-model_final.pth` и `ocr-model-last.ckpt` — веса моделей сегментации и OCR, которые подгружаются во время исполнения скрипта `run.py`.

Для корректной проверки решения сабмиты участников должны были иметь аналогичную структуру.

Критерии оценивания

Метрика качества оценки решений представляет собой расчет CER для OCR-модели: https://clc.to/ai-21-22-final-evaluate.py.

Так как текст распознается на предсказанных моделью полигонах, чтобы понять, с каким текстом сравнивать предсказанный моделью, нужно соотнести предсказанные полигоны с `ground truth` полигонами.

Скрипт расчета метрики искал для каждого gt-полигона из тетради соответствующий ему предсказанный полигон. Из предсказанных полигонов выбирался тот, который имеет наибольшее пересечение по IoU с gt-полигоном (при этом IoU должно быть больше нуля). Таким образом, gt-текст из данного полигона соотносился с предсказанным текстом. Это true positive предсказания.

False negative случаи: если для gt-полигона не был сопоставлен предсказанный полигон, то предсказанный текст для такого полигона устанавливается как пустой «» (т. к. пайплайн не предсказал текст там, где он должен быть).

Для всех false positive предсказанных полигонов (т. е. тех, для которых отсутствуют gt-полигоны) gt-текст устанавливается как пустой «» (т. к. пайплайн предсказал текст там, где его нет).

CER считалась по следующей формуле:

$$CER = \frac{\sum\limits_{i=1}^{n} dict_c(pred_i, true_i)}{\sum\limits_{i=1}^{n} len_c(true_i)}.$$

Здесь $dict_c$ — это расстояние Левенштейна, посчитанное для токенов-символов (включая пробелы), len_c — длина строки в символах.

Метрика CER изменяется от 0 до 1, где 0 — наилучшее значение, 1 — наихудшее.

При расчете качества решений также учитывалась IoU — метрика, которая оценивает степень пересечения между двумя масками (предсказанной и правильной). Она вычисляется как отношение площади пересечения к площади объединения этих двух масок:

$$IoU = \frac{Intersection}{Union}.$$

Результаты полученного рейтинга были переведены в стобалльную систему в соответствии с формулой:

$$Total = \frac{(1 - Result) \cdot (RankParticipants + 1 - RankPlace) \cdot 100}{(1 - MaxResult) \cdot Participants},$$

где $Total$ — итоговый балл за решение задачи;

$Result$ — результат решения задачи;

$RankParticipants$ — общее количество команд в рейтинге;

$RankPlace$ — место команды в рейтинге;

$MaxResult$ — лучший результат решения задачи;

$Participants$ — общее количество участников.

Критерии определения победителей и призеров

Первый отборочный этап

В первом отборочном этапе участники решали задачи по двум предметам: математика и информатика, в каждом предмете максимально можно было набрать 100 баллов. Для того, чтобы пройти во второй этап участники должны были набрать в сумме по обоим предметам не менее 50 баллов, независимо от уровня.

Второй отборочный этап

Количество баллов, набранных при решении всех задач второго отборочного этапа, суммируется.

Победители второго отборочного этапа должны были набрать не менее 16 баллов, независимо от уровня.

Заключительный этап

Индивидуальный предметный тур

- Математика — максимально возможный балл за все задачи — 100 баллов;
- Информатика — максимально возможный балл за все задачи — 100 баллов.

Командный практический тур

Команды, участники заключительного этапа, получали за командный практический тур от 0 до 100 баллов. Команда, набравшая максимальное число балов, становилась командой-победителем.

В заключительном этапе олимпиады индивидуальные баллы участника складываются из двух частей, каждая из которых имеет собственный вес: баллы за индивидуальное решение задач предметного тура по предметам (математика, информатика), с весом 0,1 каждый за предмет, и баллы за командное решение инженерных задач командного тура в области технологий Искусственного интеллекта с весом 0,8.

Итоговый балл определяется по формуле:

$$S = 0,1 \cdot (S1 + S2) + 0,8 \cdot S3,$$

где:

- $S1$ — балл первой части заключительного этапа по математике в стобалльной системе ($S1\,\text{макс} = 100$);

- $S2$ — балл первой части заключительного этапа по информатике в стобалльной системе ($S2\,\text{макс} = 100$);
- $S3$ — итоговый балл командного тура в стобалльной системе ($S3\,\text{макс} = 589$).

Итого максимально возможный балл по условиям общего рейтинга:

$$0,1 \cdot 100 + 0,1 \cdot 100 + 0,8 \cdot 100 = 100\,\text{баллов}.$$

Критерий определения победителей и призеров (независимо от класса)

Для определения победителей и призеров (независимо от класса) был сформирован общий рейтинг, где 9-е классы участвовали на общих основаниях с 10–11 классами. С начала рейтинга были выбраны 7 победителей и 17 призеров (первые 25% участников рейтинга становятся победителями или призерами: первые 8% участников рейтинга становятся победителями, оставшиеся — призерами).

Критерий определения победителей и призеров (независимо от уровня):

Категория	Количество баллов
Победители	77,2 и выше
Призеры	от 53,38 до 77